



كلية الهندسة  
College of Engineering  
QATAR UNIVERSITY جامعة قطر

BACHELOR OF SCIENCE DEGREE / DEGREE WITH HONOURS IN  
ELECTRICAL ENGINEERING

Senior Design Project Report  
Department of Electrical Engineering  
Qatar University

## Development of Instrumentation for High-Precision Angular Position and Speed Measurement

---

Report by

Asem Khattab

Sharief Saleh

Mohammed El-Jaam

Mahmoud Mesleh

Supervisors

Prof. Mohieddine Benammar

Dr. Faycal Bensaali

Date



14/06/2016

# DECLARATION STATEMENT

We, the undersigned students, confirm that the work submitted in this project report is entirely our own and has not been copied from any other source. Any material that has been used from other sources has been properly cited and acknowledged in the report.

We are fully aware that any copying or improper citation of references/sources used in this report will be considered plagiarism, which is a clear violation of the Code of Ethics of Qatar University.

In addition, we have read and understood the legal consequences of committing any violation of the Qatar University's Code of Ethics.

	Student Name	Student ID	Signature	Date
1	Asem Khattab	201208108		13/06/2016
2	Mahmoud Mesleh	201104499		13/06/2016
3				
4				

# ABSTRACT

Being able to measure the angular displacement of a rotary device is one of the pillars of today's innovation, and tomorrow's advancement. Trigonometric sensors are used heavily in many sensitive applications such as industrial, biomedical and military applications where accurate measurements are required. A particular type of these sensors provide two sinusoidal outputs that are proportional to the measured angle. The two main aims of this project are to design an online correction method to correct the imbalances within these sinusoids as well as designing a powerful low-cost converter that is able to extract the angle from the outputs of trigonometric sensors at high accuracy and speed. Within this report, literature review will be undergone for imbalance correction techniques as well as for open loop and closed loop converters. Building on that, a novel online correction technique will be presented as well as a novel open loop converter that was given the name "Phase Shifted Tangent (PST)". Then, a chosen closed loop converter from the literature will be re-implemented to compare and contrast its results with the PST method. All of these methods are described in full details along with extensive computer simulation using MATLAB and SIMULINK. After that, all methods are practically tested with the designed Test Bench with the aid of both emulated and real sensors under different operating conditions to test their reliability.

# ACKNOWLEDGEMENTS

The team is very thankful to the supervisors of this project as well as Eng. Antonio for their contentions support and help, furthermore, we appreciate the help of Eng. Ahmed Zahran, Eng. Ayman Ammar and Eng. Mohammed Ayad for their help to build the Test Bench.

## TABLE OF CONTENTS

DECLARATION STATEMENT .....	i
ABSTRACT .....	ii
ACKNOWLEDGEMENTS .....	iii
LIST OF FIGURES.....	vi
LIST OF TABLES.....	x
GLOSSARY .....	xi
Chapter 1 Introduction.....	1
1.1 Sinusoidal Encoders and the Resolver .....	2
1.2 Objectives of the Project .....	3
Chapter 2 Literature Review .....	4
2.1 Signals Correction .....	4
2.2 Open-Loop Conversion .....	5
2.3 Closed-Loop Conversion.....	10
2.4 Constraints .....	11
Chapter 3 Imbalances Correction .....	12
3.1 Mathematical Background.....	12
3.1.1 The Proposed Algorithm .....	13
3.1.2 Validity of the Test Points.....	15
3.2 Simulation.....	15
3.3 Implementation and Practical Testing .....	17
3.3.1 Microcontroller Implementation .....	17
3.3.2 Emulation Results .....	19
3.3.3 Practical Usage .....	24
Chapter 4 Open-Loop Converter .....	25
4.1 Theoretical Explanation.....	25
4.1.1 Logic Circuit Design .....	29
4.1.2 The Number of Sections and the Computational Cost.....	31

---

Development of Instrumentation for High-Precision Angular Position and Speed Measurement

---

4.1.3	Converterwith16 Sections .....	32
4.2	Simulation.....	33
4.3	Implementation and Practical Testing .....	36
4.3.1	Microcontroller Implementation .....	37
4.3.2	Emulation Results .....	39
4.3.3	Test Bench Results .....	43
Chapter 5 Closed-Loop Converter .....		45
5.1	Theory .....	45
5.2	Simulation.....	46
5.3	Implementation and Practical Testing .....	48
5.3.2	Test Bench Results .....	49
Chapter 6 Test Bench Implementation .....		51
6.1	Introduction.....	51
6.2	Rotary Table.....	51
6.3	Test Bench Introduction .....	52
6.4	Equipment .....	52
6.5	Hardware Implementation .....	53
6.6	Software Implementation .....	55
6.7	Test Bench Final Assembly.....	62
Chapter 7 Conclusion.....		63
7.1	Achievements.....	64
7.2	Further Development .....	64
REFERENCES.....		66

# LIST OF FIGURES

Figure 1-1: Robotic arms in an automated factory, (Picture: Paul Sakuma/AP).....	1
Figure 1-2: A doctor performing a robotic surgery, (Picture: Visage Technologies).....	1
Figure 1-3: A Simplified model of a resolver and associated signals demodulating unit.....	2
Figure 2-1: Example of transducer signals before and after balancing. ....	4
Figure 2-2: Lissajous curves of an encoder's output with different imbalances. ....	5
Figure 2-3: A simplified block diagram of the method proposed in [15].....	6
Figure 2-4: Alternation between tangent and cotangent.....	6
Figure 2-5: Getting pseudo-linear signal by alternating between sensor's outputs .....	7
Figure 2-6: The logic circuit used in [18]. ....	7
Figure 2-7: Phase Shifted Sinewaves.....	8
Figure 2-8: MATLAB Simulink Simulation of the method in [18].....	8
Figure 2-9: The logic circuitry used in [19]. ....	9
Figure 2-10:MATLAB Simulink Simulation of the method in [19].....	9
Figure 3-1: A flowchart of the suggested algorithm. ....	15
Figure 3-2: A flowchart of the simulation program. ....	16
Figure 3-3: Maximum absolute conversion error using arctangent method. ....	17
Figure 3-4: Pseudo code implementation of the correction method function. ....	18
Figure 3-5: receiving the data of the test though a terminal software.....	19
Figure 3-6: Result of the simulation done inside the microcontroller.....	19
Figure 3-7: The emulation setup. ....	20
Figure 3-8: The emulation program to test the correction technique. ....	20
Figure 3-9: The estimated imbalances and the correction parameters for case 1. ....	21
Figure 3-10: The <i>US</i> and <i>UC</i> signals (yellow and green respectively) and the corrected <i>UC</i> signal (blue) for case 1.....	21
Figure 3-11: The <i>US</i> and <i>UC</i> signals (yellow and green respectively), the estimated angle before correction (blue) and the estimated angle after correction signal (pink) for case 1. ....	22

Figure 3-12: The estimated imbalances and the correction parameters for case 2. ....	22
Figure 3-13: The <i>US</i> and <i>UC</i> signals (yellow and green respectively) and the corrected <i>UC</i> signal (blue) for case 2.....	22
Figure 3-14: The <i>US</i> and <i>UC</i> signals (yellow and green respectively), the estimated angle before correction (blue) and the estimated angle after correction signal (pink) for case 2. ....	23
Figure 3-15: The estimated imbalances and the correction parameters for case 3. ....	23
Figure 3-16: The <i>US</i> and <i>UC</i> signals (yellow and green respectively) and the corrected <i>UC</i> signal (blue) for case 3.....	23
Figure 3-17: The <i>US</i> and <i>UC</i> signals (yellow and green respectively), the estimated angle before correction (blue) and the estimated angle after correction signal (pink) for case 2. ....	24
Figure 3-18: Conversion Error for a fixed angle before (Up) and after (bottom) applying the signal correction algorithm.....	24
Figure 4-1: Block diagram of the proposed Phase Shifted Tangent method applied to a resolver. ....	25
Figure 4-2: Simulation results for converter with N=4.....	27
Figure 4-3: Maximum Error as a function of the number of sections N for small angle approximation option and the LUT option with different number of elements. ....	28
Figure 4-4: Stages of Calculating <b>BM</b> – 4.....	30
Figure 4-5: The logic circuit for four, eight and sixteen sections.....	31
Figure 4-6: Simulation result for converters with number of sections N=16. ....	34
Figure 4-7: The error of the proposed method compared to the error of other similar methods in presence of an accurate measurement of the amplitude of the SCS. ....	35
Figure 4-8: The error of the proposed method compared to the error of other similar methods in presence of 0.1% error in the amplitude of the SCS. ....	35
Figure 4-9: The error (in degrees) of the proposed method compared to the error of other similar methods in presence of amplitude and phase imbalances.....	36
Figure 4-10: Pseudo-code for calculating the needed parameters of the converter. ....	37
Figure 4-11: Pseudo-code of the PST converter function.....	38
Figure 4-12: Receiving the estimated angle (left column) and the input angle (left column).....	39
Figure 4-13: Simulation done inside the microcontroller for the proposed method. ....	39
Figure 4-14: Simulation done inside the microcontroller for similar methods. ....	39
Figure 4-15: The block diagram of the emulation program.....	40



Figure 4-16: One revolution error signals of the proposed converter with sixteen sections and similar methods.....	41
Figure 4-17: One revolution error signals for the proposed converter with lower number of sections.	42
Figure 4-18: the <i>US</i> and <i>UC</i> signals (yellow and green respectively) and the estimated angle (blue).	42
Figure 4-19: The block diagram (left) and the front panel (right) of the step change emulation.....	43
Figure 4-20: the <i>US</i> (yellow), <i>UC</i> (green) signals and the estimated angle (blue) for a step change from 50 to 80 degrees. ....	43
Figure 4-21: Error signal from the test bench for the proposed converter with 16 sections using small angle approximation.....	44
Figure 5-1: The schematics of the proposed method of the PLL converter [22].....	45
Figure 5-2: Simulink module representing the PLL.....	47
Figure 5-3:The output of the Simulink module in figure 5-2. (a) shows a step input of 195° at 0.5 seconds. (b) shows a ramp input. Yellow is the estimated angle, Pink is the input angle and Blue is the error..	47
Figure 5-4: The error signal of the estimated angle (a), and the estimated angle with respect to the real angle.....	49
Figure 5-5: The estimated angle and the conditioned sine and cosine. ....	49
Figure 5-6: The estimated angle with respect to the real angle and the error of the estimated angle plot. ....	50
Figure 6-1: Rotary Table setup .....	51
Figure 6-2: Final Assembly of Test Bench .....	52
Figure 6-3: Test bench base .....	54
Figure 6-4: Version 1 SolidWorks brackets design.....	54
Figure 6-5: Version 2 SolidWorks brackets design.....	54
Figure 6-6: Test Bench Version 2 SolidWorks assembly.....	55
Figure 6-7: Test Bench Hardware Assembly .....	55
Figure 6-8: NI my-DAQ .....	56
Figure 6-9: front and back panels of the LabVIEW program .....	57
Figure 6-10: second version front and back LabVIEW panels.....	58
Figure 6-11:Initial angle error first solution front and back LabVIEW panels.....	60
Figure 6-12: Initial angle error Second solution front and back LabVIEW panels .....	60

Figure 6-13: Pulse Encoder pulse per revolution front and back LabVIEW panels ..... 61

# LIST OF TABLES

Table 4-1: Comparison between our algorithm with sixteen sections and equivalent logic circuits in other methods.....	30
Table 4-2: Computational cost of the logic circuit as a function of the number of sections. ....	31
Table 4-3: List of the parameters needed for a converter with number of sections $N=16$ .....	33

# GLOSSARY

<b>Symbol</b>	<b>Meaning</b>
A	Maximum amplitude of the
ADC	Analog-to-digital converter
DAC	Digital-to-analog converter
Emulation	Producing the same sensor's outputs using another device
LSB	Least significant bit
MSB	Most significant bit
PSS	Phase shifted sinewaves
PST	Phase shifted tangents

# Chapter 1 INTRODUCTION

In today's modern technology, the need for more precision becomes critical in many fields. Control algorithms are becoming more advanced and depending on very accurate feedback. In this context, precise angular position and speed measurements are very important in many applications. This includes, for example, industrial applications like factories automation in which large robotic arms are used in production lines as seen in Figure 1-1.



**Figure 1-1: Robotic arms in an automated factory, (Picture: Paul Sakuma/AP).**

A more sensitive application can be biomedical robotic surgery (as in Figure 1-2) where a slight error can result in a much undesired consequence. Other applications may be found in military, avionics, communications and other fields.



**Figure 1-2: A doctor performing a robotic surgery, (Picture: Visage Technologies).**

This senior design project focuses on sinusoidal encoders, a popular category of rotational position sensors. The objective of this project, as will be clarified in details later, is achieving more precise and stable estimation of the measured angle based on the outputs of a sinusoidal encoders. The project

---

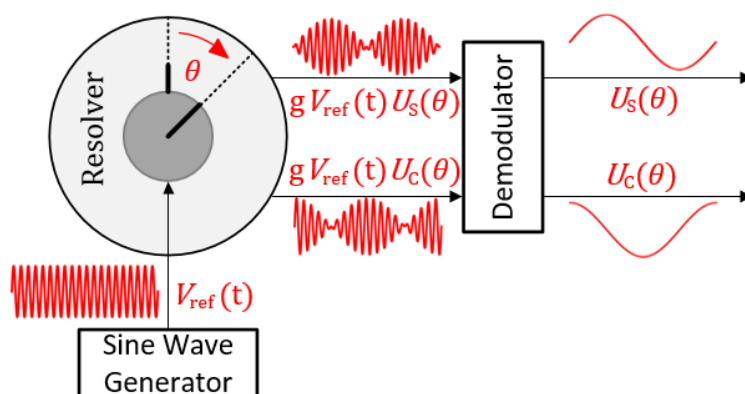
Development of Instrumentation for High-Precision Angular Position and Speed Measurement

does not deal with the design of the sensors themselves nor does it consider the whole systems in which these sensors are employed. The only concentration of the project is the level of angle estimation methods based on the sensor outputs and the performance of such methods.

The report is divided into seven chapters. The first one is the introduction, where the topic is introduced. After that, there is the literature review where the previous work connected to the topic of this project is reviewed and evaluated. Next, a new method of correcting sensors' signals imbalances is presented and discussed in addition to simulation and practical testing. After that, also, a new open-loop conversion method is introduced. The quality of its performance is proved using both simulation and practical testing. The fifth chapter discusses the theory and implementation of a closed-loop conversion method. Furthermore, the next chapter shows how these conversion methods can be integrated in a closed loop servo control system. Finally, the report is concluded with an overview of what was achieved, what can be improved and what is the next step.

## 1.1 Sinusoidal Encoders and the Resolver

Sinusoidal encoders are very popular absolute position and speed transducers. Operating on different principles, including magnetoelectric, inductive, Hall effect or optical techniques, these transducers produce electrical analog quadrature signals that encode the unknown angle allowing an absolute estimation of the angular position [1]-[5]. Resolvers (Figure 1-3), for example, are rotating transducers that are used widely due to their reliability and ability to give precise angular measurements in harsh environments.



**Figure 1-3: A Simplified model of a resolver and associated signals demodulating unit.**

In its basic form, a resolver is similar to an electric motor with the primary winding in the rotor and two secondary mechanically  $90^\circ$  apart stator windings. Modern resolvers are brushless, a feature that significantly increases their reliability and life-time. A sinusoidal excitation signal is applied to the primary winding. The excitation voltage is also referred to as the reference or the carrier  $V_{ref}(t) = V_m \sin 2\pi f_c t$ , where  $V_m$  is the peak amplitude, typically a few Volts, and  $f_c$  is the frequency of the signal, typically a few kHz and must be considerably higher than the maximum rotational speed of the sensor. This excitation causes the two secondary windings to generate two output voltages that are modulated by the excitation signal. Because of the mechanical phase shift between the windings, one output depends on the sine while the other on the cosine of the rotor angle (i.e.  $gV_{ref} \sin \theta$  and  $gV_{ref} \cos \theta$ )

where  $g$  is a constant that represents the transformation ratio between the primary and secondary windings and  $\theta$  is the measured mechanical angle. The two signals are then demodulated resulting ideally in two sine and cosine signals (SCS) of the form:

$$\begin{aligned} V_s(\theta) &= A \sin \theta \\ V_c(\theta) &= A \cos \theta \end{aligned} \quad (1-1)$$

Where  $A$  is the amplitude of the two SCS and depends on the excitation signal amplitude  $V_m$ , the transformation ratio  $g$  and, sometimes on the demodulation technique. Actually, the signals shown in (1-1) represent the ideal outputs of all kinds of sinusoidal encoders not only the resolver. Though, other sinusoidal encoders may not require any demodulation and directly output these signals.

In reality, however, the sensors' output signals are not perfectly described by (1-1), and contain errors from various sources. The most significant imperfections reported in the literature are: amplitude imbalance and phase imbalance. As will be seen in the literature review, taking the transducer's sine signal as a reference and ignoring other sources of imperfections, the actual output can be modelled by:

$$\begin{aligned} U_s(\theta) &= A \sin \theta \\ U_c(\theta) &= (1 + E_a)A \cos(\theta + E_p) \end{aligned} \quad (1-2)$$

where  $E_a$  and  $E_p$  are the amplitude and phase errors, respectively. These imbalances must be detected and corrected before estimating the measured angle since all conversion techniques (techniques to extract the measured angle from the outputs) assume the ideal output shown in (1-1). The main previous techniques to correct these imbalances are reported in section 2.1 and a novel correction method that is simple and computationally efficient is presented in section 3.

After correcting the outputs, a suitable converter is needed to extract the measured angle  $\theta$  from the output signals. Conversion techniques reported in the literature can be classified into two main categories: open-loop converter and closed-loop converters. Closed loop converters obviously require a feedback of some kind to track the measured angle. They are more robust but much slower than open loop converter in terms of dynamic response. Main previous open-loop and closed-loop converters are reviewed in sections 2.2 and 2.3 respectively. In addition to that, a novel open-loop converter that produces a very linear output, independent of the amplitude of the sensor signals  $A$  and allows compromising computational complexity with accuracy is introduced in section 4. Moreover, a recent closed-loop converter from the literature is presented and implemented in section 5, while section 6 shows the performance of these different converters in a closed-loop servo control system.

## 1.2 Objectives of the Project

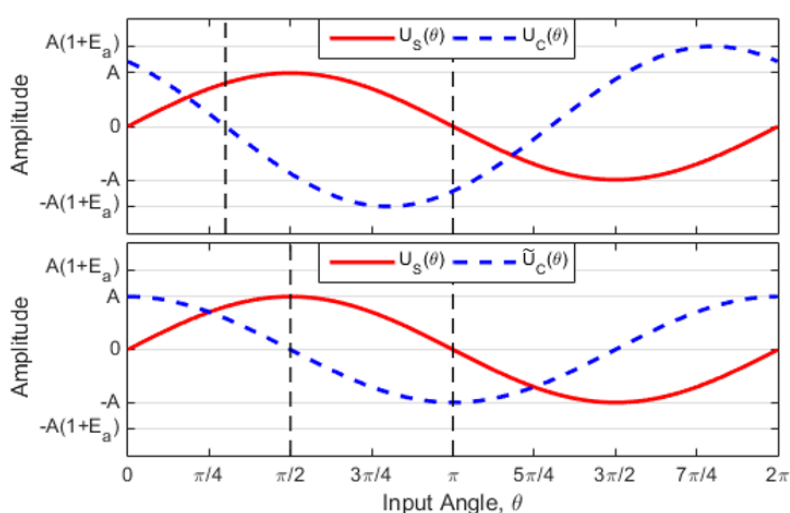
The objectives of our project can be summarized in the following points:

- 1- Design, simulate and test an open-loop resolver's signals correction technique.
- 2- Design, simulate and test an open-loop resolver-to-digital conversion technique.
- 3- Implement a recent closed-loop resolver-to-digital converter and compare the results with the designed open-loop converter.

# Chapter 2 LITERATURE REVIEW

## 2.1 Signals Correction

As seen in the previous chapter, the main imperfections reported in the literature are: amplitude imbalance, phase imbalance, offsets and harmonics [6]. However, the most significant causes of errors are amplitude and phase imbalances [6]. Figure 2-1 shows an example of transducers signals that present exaggerated phase and amplitude imbalances; correction of these signals results in balanced versions as shown in the lower part of Figure 2-1. To show the significance of amplitude and phase imbalances, either 0.63% amplitude imbalance or  $0.18^\circ$  phase imbalance is enough to cause 1/2 LSB position error for a 10-bit PLL converter [6]. The other errors are small and have minor impact on the performance of sensors converters [7]. Amplitude imbalance is a result of unequal inductances of the resolver secondary windings [8]. On the other hand, phase imbalance between the two SCS results from errors in the mechanical positioning of these winding (i.e. imperfect quadrature) [6]-[8]. Taking the transducer's sine signal as a reference, these errors can be reflected in the imbalanced signals as in (1-2).



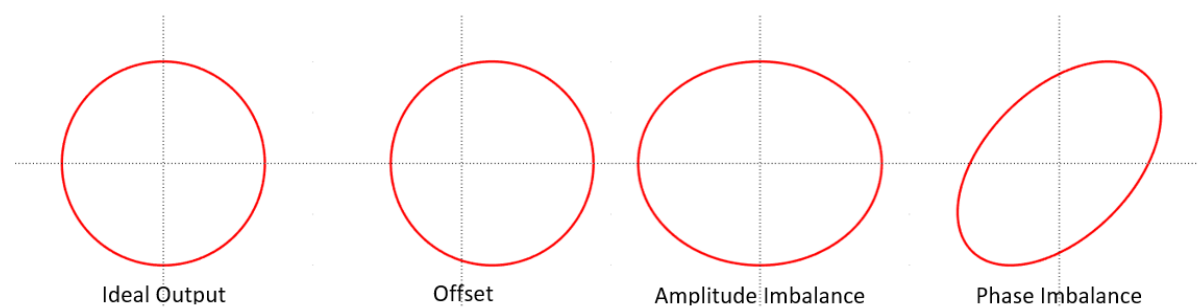
**Figure 2-1: Example of transducer signals before and after balancing.**

**Upper: amplitude and phase imbalances of  $U_C(\theta)$  compared to  $U_S(\theta)$ . Lower: the corrected cosine  $\tilde{U}_C(\theta)$  signal compared to  $U_S(\theta)$ .**

Several methods have been suggested to compensate for these imbalances. In [6], the correction could not be done for the two errors together. Instead, the amplitude imbalance had to be solved manually first, which required examining the outputs of the sensor for the full  $360^\circ$  interval. After that, the phase imbalance was resolved by replacing  $U_S(\theta)$  and  $U_C(\theta)$  by their sum and difference respectively, which would eliminate the phase error, but will also impose another amplitude imbalance that has to be corrected again which makes the correction lengthy and inefficient. It is also not easy to be done frequently in order to account for the long term drifts in errors due to aging. Furthermore, the method did not offer amplitude and phase errors estimation. Another method was proposed in [9] that



depends on the Lissajous curve of the encoder outputs (where  $U_s(\theta)$  is the Y-axis and  $U_c(\theta)$  is the X-axis). The areas and radii of the four quadrants of the curve were evaluated to detect and get a fair estimation of the amplitude and phase imbalances and offset as in Figure 2-2. It is obvious that to be able to plot the Lissajous curve, many samples are needed to cover the whole rotation interval requiring the sensor to rotate  $360^\circ$ . Furthermore, the method is only able to correct for the amplitude imbalance and offset error ignoring the more significant phase error.



**Figure 2-2: Lissajous curves of an encoder's output with different imbalances.**

Also, an interesting algorithm was presented in [10] in which an iterative linear search method known as “steepest descent” is utilized to correct for different non-ideal characteristics of encoders using samples of the  $U_s(\theta)$  and  $U_c(\theta)$  signals. The main disadvantage, however, is the relatively long time (up to 0.5 seconds) required for the algorithm to converge because of the needed small learning rate, which does not allow online correction at the same time of conversion.

The previous methods are all open-loop methods where no feedback is required. Closed-loop schemes were also proposed as in [7] where amplitude and phase errors were corrected using an adaptive PLL with synchronous demodulators, which requires more computational power and increases the time of processing. Moreover, the correction is done continuously with the conversion which adds an unnecessary delay since the errors are not expected to vary continuously. Another closed-loop method was presented in [11] and required evaluation of the square magnitude of the encoder outputs and the mean values over a sufficient number of periods. Because of the feedback required in these methods, they are not valid for certain scenarios such as fast point-to-point positioning where the data are not enough for the loop to correct the errors. Other closed-loop correction techniques were presented in [8], [12] for the specific application of PMSM drives utilizing some parameters from the drive itself. In addition to that, there are methods that utilize multiple encoders to perform correction like in [13] where two resolvers' outputs are compared to estimate the errors and correct them. A method that is, of course, not feasible in many situations. In chapter 3, we present a new simple and effective method of detecting and correcting amplitude and phase imbalances.

## 2.2 Open-Loop Conversion

After ensuring that the output signals are balanced, they need to be processed by a converter in order to extract the unknown angle  $\theta$  from the SCS signals. The converters that have been suggested in literature can be classified into two main categories: closed loop and open loop types. Closed loop converters require a feedback and generally utilize techniques like the phase locked loop (PLL) and the

angle tracking observer (ATO). Several closed-loop conversion techniques will be reviewed in the next subsection. Generally, open loop converters have a better dynamic performance and are able to operate at much higher sensor speeds.

Many open loop converters have been proposed. Some rely on lookup tables (LUT) to get  $\theta$  from the ratio between the two SCS [14][15]. A simplified block diagram of the method in [15] is shown in Figure 2-3. The method utilizes a software based converter that generates the carrier signal and samples the sensor outputs at the peaks of the carrier to demodulate these outputs, a technique that is often referred to as synchronous demodulation.

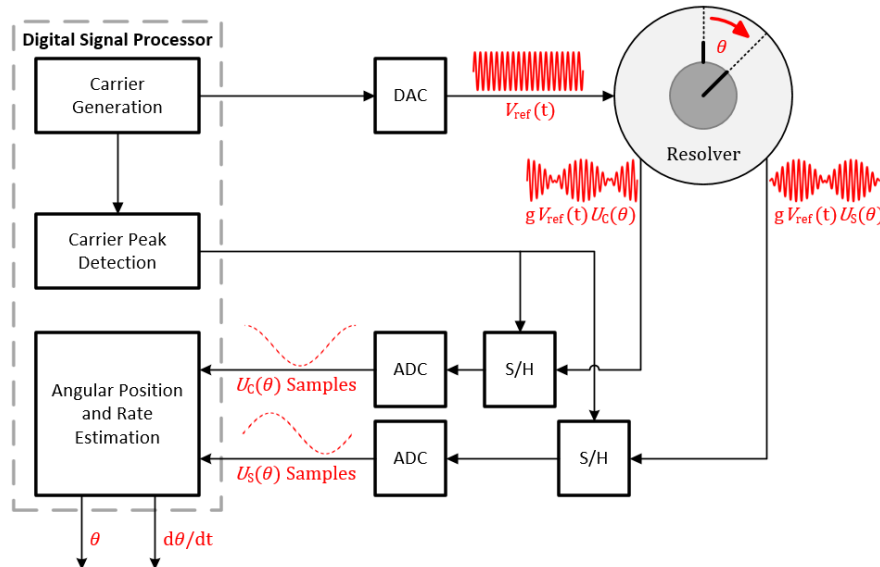


Figure 2-3: A simplified block diagram of the method proposed in [15].

Because any both ratios between the two SCS have singularities ( $\tan \theta$  for example is not defined at  $\theta = \pi/2$  and  $\theta = 3\pi/2$ ), this method and similar ones usually alternate between  $\tan \theta$  and  $\cot \theta$  as in Figure 2-4 requiring LUTs of relatively large sizes. A logic circuit is needed to determine which of the two ratios has to be used (and hence which part of the LUT). Of course the error of such method is heavily dependent on the size of the LUT and the interpolation method used between the points of the LUT. Despite that, the method takes advantage of the Digital Signal Processor (DSP) to effectively reduce the complexity of the system by minimizing the number of components used. The output of this method is totally independent of the maximum amplitude  $A$  as it is ratiometric. However, it is highly non-linear as seen in Figure 2-4.

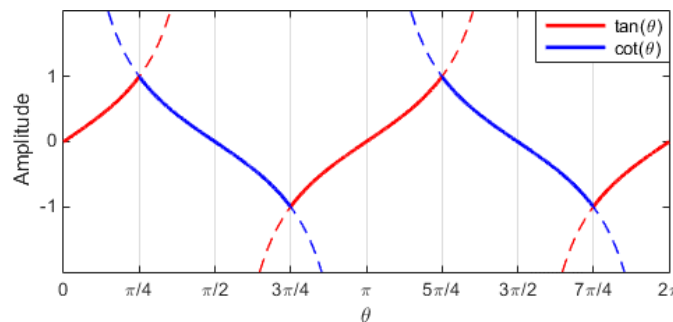
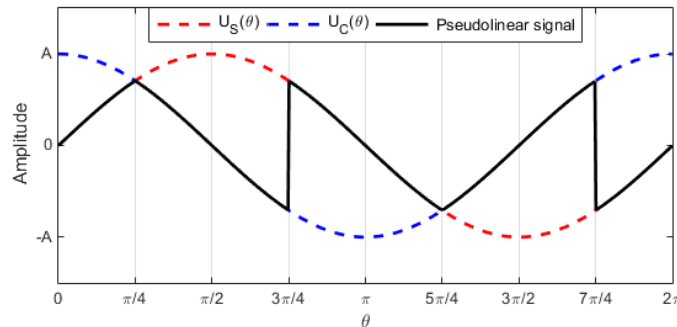


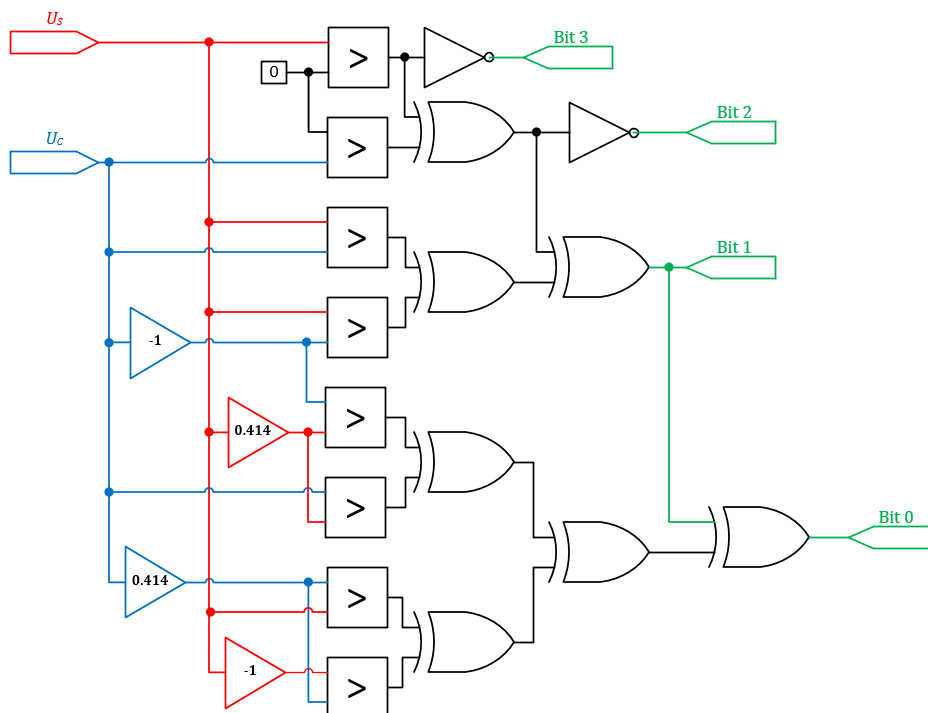
Figure 2-4: Alternation between tangent and cotangent.

Other open loop converters utilize the pseudo-linearity of the SCS by multiplexing between the two SCS signals together with linearization techniques [16][17]. Figure 2-5 shows the selected pseudo-linear segments from the SCS throughout the 360° interval in solid line. The resultant signal is then linearized and corrected by different techniques and using it the input angle can be determined by simple mathematical relationships provided that the quadrant in which the input angle is in has been determined using an appropriate logic circuit.

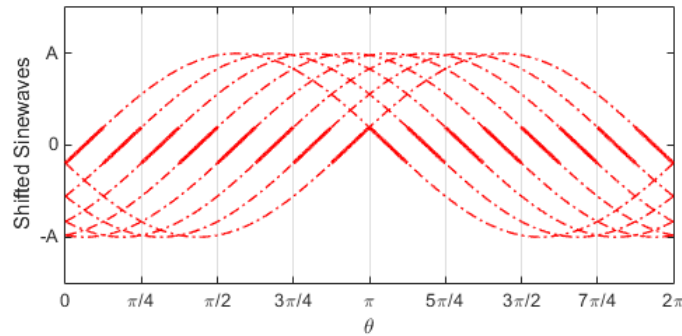


**Figure 2-5: Getting pseudo-linear signal by alternating between sensor's outputs**

Another open loop conversion technique has been first introduced by Benammar *et al.* [18] and consists of generating phase shifted sine waves (PSS) together with dedicated multiplexing between the PSS in order to make use of their alternating pseudo-linear segments. In this method, the whole rotation interval was divided into 16 equal length segments. A logic circuit was used to locate the angle in one of these segments as shown in Figure 2-6. The output of this logic circuit was used to choose one of 8 PSS shown in solid line in Figure 2-7. It is clear that this output is much more linear than that shown in Figures 2-4 and 2-5. One of the disadvantages of this method, however, is that it was implemented using analog electronics, which required many components and increased the cost and size of the converter.

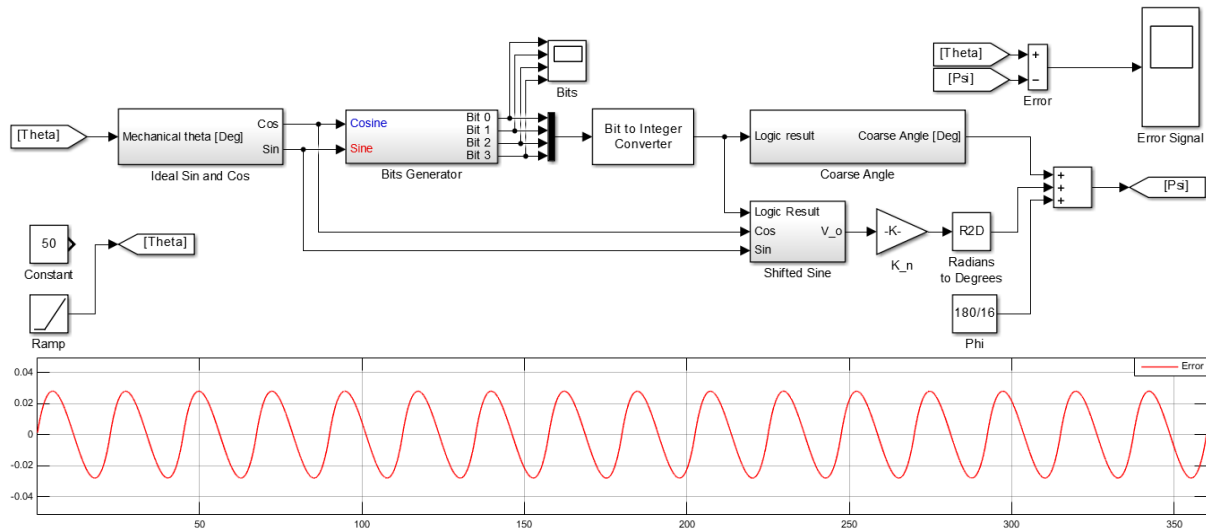


**Figure 2-6: The logic circuit used in [18].**



**Figure 2-7: Phase Shifted Sinewaves.**

To get a better understanding of this method and looking for the possibility to improve it, it was simulated using MATLAB Simulink as well as MATLAB code. The simulation diagram and the simulated error signal (in degrees) are shown in Figure 2-8. The resultant error signal is typical to the simulation results shown in the paper itself.



**Figure 2-8: MATLAB Simulink Simulation of the method in [18].**

**Up: the simulation block diagram. Down: The error signal.**

In the same context, a new method by Wang *et al* was published very recently in 2016 [19]. The method also takes advantage of the pseudo-linearity of sine waves by generating 4 PSS from the SCS and their sum and difference, followed by linearization techniques. Although, the method is similar to that in [18] and uses a logic circuit to produce four bits, the proposed logic circuit is unnecessarily and unjustifiably complex involving large number of logic gates, comparators and absolute value operations as in Figure 2-9.

Like the previous method, the method in [19] was also simulated using MATLAB Simulink as well as MATLAB code and we were able to get the same simulation results shown in the original paper. The Simulink diagram and the simulated error signal (in degrees) can be seen in Figure 2-10. These simulations of these two method were used to compare them to the novel method we proposed in section 4.

It can be seen that all the methods depending on pseudo-linearity [16]-[19] use logic circuitry to locate the measured angle in one of equal-length sections in the  $[0, 2\pi]$  rotation interval. They are

capable of giving a significantly more linear output than that given by methods that rely on  $\tan \theta$  and  $\cot \theta$  like in [15]. However, these methods share a serious drawback; they require an accurate knowledge of the peak amplitude  $A$  of the SCS. This is mainly because these methods produce outputs proportional to  $A\theta$ . In practice, however,  $A$  may vary for one or more reasons, including changes in the amplitude of the carrier signal, ageing of the sensor, and inaccuracy in the demodulation of the sensor signals. This may lead to significant increase in the previously reported converters errors as will be shown later.

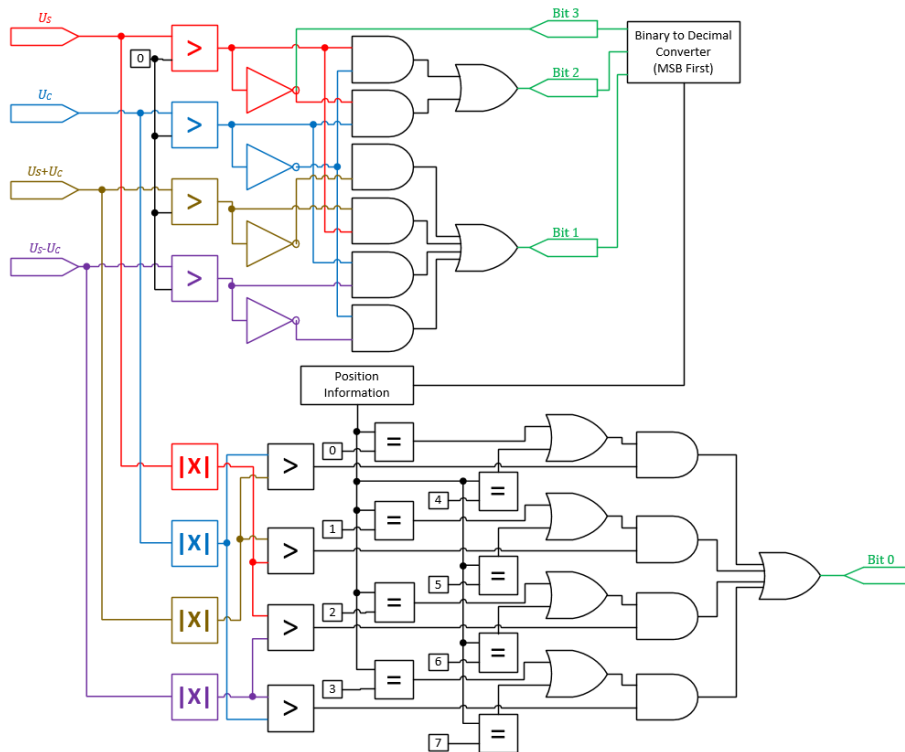


Figure 2-9: The logic circuitry used in [19].

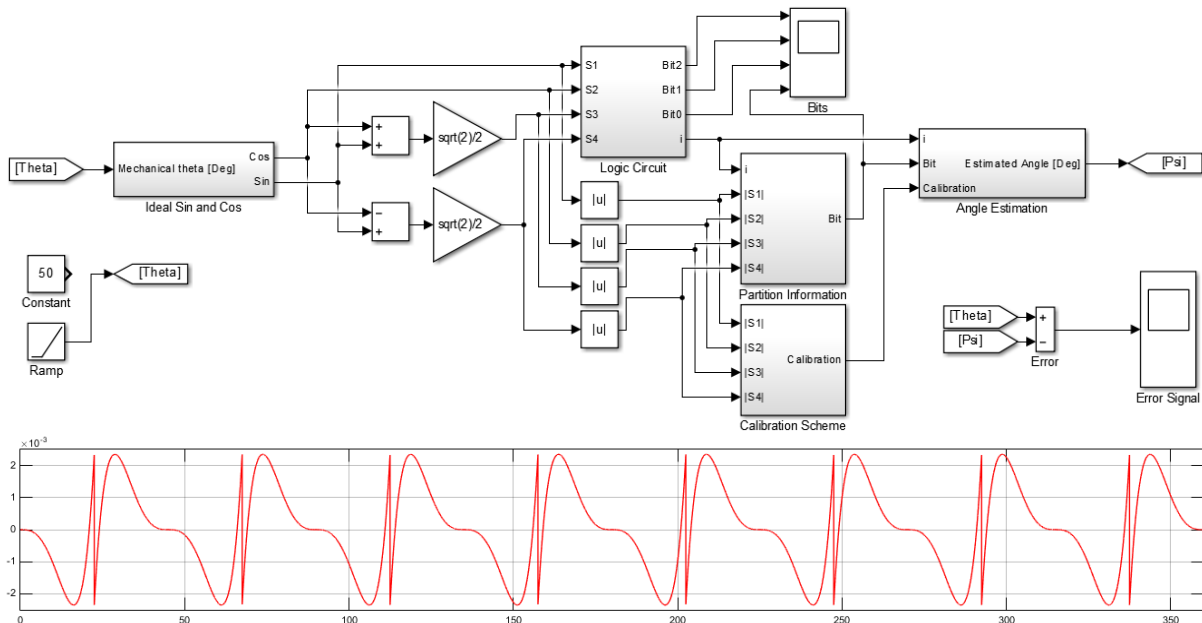


Figure 2-10: MATLAB Simulink Simulation of the method in [19].

Up: the simulation block diagram. Down: The error signal.

In section 4, we introduce a novel open-loop converter that can give a very linear output at low computational cost. The converter is totally independent of the maximum amplitude of the SCS and is adaptive to the required accuracy and the available computational power. Moreover, it comes with a new logic circuit algorithm that is greatly simpler than these presented in this review and is able to not only give precise results in our converter but also improve the performance of the previous methods.

## 2.3 Closed-Loop Conversion

Many closed loop converters have been proposed in literature. As the name suggests, all of them have a sort of feedback system and a controller. Most of these converters utilize the Phase Locked Loop “PLL” technique as mentioned earlier. PLLs in general consist of three main stages, a Phase detector PD, where the input SCS signals get converted to an error signal. Followed by a loop filter; Integrator, and sometimes a PI / PID controller that will result in the estimated angle, and at last, a stage that converts the estimated angle to SCS signals back again to be feedback to the PD stage and close the loop. Every paper focuses in one of these stages to improve some aspects of the overall converter such as the steady state error, the dynamic response and the overall complexity.

In classical PLLs, the PD stage is simply performing a multiplication and subtraction between the input and the feedback sinusoids to get a signal proportional to the difference between the real angle and estimated angle as shown in (2-1):

$$\sin(\theta) \times \cos(\psi) - \cos(\theta) \times \sin(\psi) = \sin(\theta - \psi) \quad (2-1)$$

Followed by, the filter which is a simple integrator of transfer function  $\frac{1}{s}$ . Also, the classical way to convert the angle to sinusoidal signals is done via Voltage Controlled Oscillator “VCO” or Look Up Tables “LUTs” which in fact consumes much time and affects the dynamic response of the whole converter negatively. Since the input signal might be modulated signal in the case of resolvers as discussed before, a demodulator is usually added at the beginning of the loop.

One of the proposed closed loop converters is altering the PLL by removing the demodulation part entirely; which hinders the dynamic response of the loop, and replacing it by either full-wave or half-wave rectifiers to take the absolute value of the input signals [20]. In addition, they utilized a polynomial rational approximation to mimic the sine and cosine feedback signals; which in term replaces the LUT part and hence, improves the dynamic response of the loop so that it can handle 3000 RPM compared to 150 RPM for the classical PLL. Yet, the drawback here is the accuracy, as the maximum absolute error remained a big issue reaching 0.1 degrees, which is not acceptable for precise applications like military and biomedical applications.

Another modified version of the PLL method was proposed in [21] where they focused on the feedback part of the PLL. In this method, the LUT was replaced by a 12 bit pseudo Sine wave generator, this generates stair-shaped sine/cosine signals. However, this needs matched resistors and MOS switches which adds lots of cost. Moreover, in this work,  $\omega t + \theta$  (the carrier frequency plus the input angle) was tracked instead of tracking  $\theta$  only, it was actually a solution to salvage the quantization error introduced in this proposal. That's because quantization error in time domain will be translated into

---

Development of Instrumentation for High-Precision Angular Position and Speed Measurement

harmonics in frequency domain. It can be shown that while tracking  $\theta$  only, the harmonics generated by the quantization process will be very close to the PLL band, which will require higher order filters to be sufficiently eliminated. Yet if  $\omega t + \theta$  was tracked instead, these harmonics will be sufficiently pushed away from the PLL band, and thus low order filters can be employed to remove these harmonics. The major drawbacks of this method is its complexity and its high implementation cost while in return, the accuracy is improved slightly.

## **2.4 Constraints**

**Error and accuracy:** the quality of any resolver-to-digital converter is assessed by two main properties: computation speed and the maximum conversion error. Conducting a literature review, the team has not found any specific standard of the acceptable conversion error. This is mainly because the acceptable conversion error depends mainly on the application and also because this an active field of research with continues developments. Technical handbooks like [22] considers a  $0.01^\circ$  error as a very small one. In this project, the team compares the maximum conversion error of designed method with that of the similar and recent methods. A good measure is the maximum error of the method in [18] which is  $0.028$  degrees and that of the method in [19] at  $0.00235$  degrees. Moreover, an imbalance correction technique was designed and added to the converter, which plays a very important rule in significantly decreasing the error.

**Time and speed:** when dealing with real-time systems, time is one of the most significant constraints. In this project, a signal imbalance correction technique was designed. To make it run as smoothly as possible and to allow online correction, the design was limited to use an open-loop approach with minimum number of sample points needed to perform imbalances. Time was also taken into consideration in designing the conversion method. Which is the reason the design was limited to open-loop. A simplified logic circuit algorithm was suggested and simple conversion operations were used.

**Cost:** the microcontrollers as well as the DACs used are low cost components that has many limitations in terms of their specifications. This includes microcontrollers speed, ADC and DAC precision and sample rate. These limitations are discussed in the following chapters. Although a budget was available, shipping took long time and some of the components that were ordered couldn't be received, so the best choice was to work with what is available. The presented correction method and conversion methods were both designed so that they can be implemented easily and run smoothly on low-cost microcontrollers available on-shelf.

# Chapter 3 IMBALANCES CORRECTION

It can be noted from the literature review in section 2.1 that most of the previous correction techniques suffered from one of two problems: requiring too many samples or taking a long time to execute. In the present work, a novel analytical algorithm, that is able to precisely detect and correct both phase and amplitude imbalances, is presented. The algorithm does not require any additional hardware and uses a minimum computational power and processing time allowing it to be implemented on low-cost 16bit or 32bit microcontrollers and be easily used online during conversion. Furthermore, the proposed correction technique is not tied to any conversion method and can be used without restrictions with both open-loop and closed-loop converters.

## 3.1 Mathematical Background

The idea of the algorithm is to use  $U_s(\theta)$  and  $U_c(\theta)$  to generate a new cosine signal replacing  $U_c(\theta)$  which has phase and amplitude errors. The algorithm requires an accurate knowledge of  $A$ , the maximum amplitude of the  $U_s(\theta)$  signal, as well as three valid test points at three different angles. A test point is described by a pair of values of the transducer outputs  $U_s(\theta)$  and  $U_c(\theta)$  at a specific angle  $\theta$ . The angles at which these test points have been sampled, however, are not needed. The requirements are very simple and easy to satisfy. An accurate knowledge of the maximum amplitude  $A$  is a common requirement in most of the reviewed conversion and correction methods.  $A$  can be determined by different ways. One of them is to measure the peak-to-peak value of the  $U_s(\theta)$  signal and divide it by two. The  $U_s(\theta)$  signal is taken as a reference in terms of amplitude and phase. Even if the phase error was in  $U_s(\theta)$  not in  $U_c(\theta)$ ,  $U_c(\theta)$  will be corrected to be 90 degrees apart from  $U_s(\theta)$ . In this particular case the mechanical reference angle (the angular position at which the encoder should read a zero degree) will be shifted by the phase error present in  $U_s(\theta)$ . The following straightforward equation can be easily proven [23]:

$$\gamma \cos(\theta + \alpha) + \beta \sin(\theta) = \sqrt{\gamma^2 + \beta^2 - 2\delta\beta \sin(\alpha)} \times \cos\left\{\theta + \tan^{-1}\left[\frac{\gamma \sin(\alpha) - \beta}{\gamma \cos(\alpha)}\right]\right\} \quad (3-1)$$

where  $\gamma$  and  $\beta$  are scaling factors and  $\alpha$  is a phase shift. Using this formula, it is possible to add a scaled version of  $U_s(\theta)$  to  $U_c(\theta)$  to correct the phase imbalance and obtain a scaled cosine signal as follows:

$$U_c(\theta) + BU_s(\theta) = A[(1 + E_a) \cos(\theta + E_p) + B \sin(\theta)] = AC \cos(\theta) \quad (3-2)$$

By comparing (3-2) to (3-1), it is clear that  $B$  must be chosen such that:

$$\tan^{-1}\left[\frac{(1 + E_a) \sin(E_p) - B}{(1 + E_a) \cos(E_p)}\right] = 0 \quad \therefore B = (1 + E_a) \sin(E_p) \quad (3-3)$$

Then according to (3-1), (3-2) and (3-3),  $C$  can be expressed by:

$$C = \sqrt{(1 + E_a)^2 - B^2} = |1 + E_a| \cdot |\cos(E_p)| = (1 + E_a) \cos(E_p) \quad (3-4)$$



The absolute values of terms in (3-4) are equal to the terms themselves because, in practice,  $E_a$  is much smaller than 1 and  $E_p$  is a small angle around 0. A corrected cosine signal  $\tilde{U}_C(\theta)$  (shown in Figure 2-1) may then be determined as follows:

$$\tilde{U}_C(\theta) = A \cos(\theta) = \frac{U_C(\theta) + BU_S(\theta)}{C} \quad (3-5)$$

The corrected signal requires online determination of suitable values for  $B$  and  $C$  as shown below. Using  $B$  and  $C$ , amplitude and phase errors can be quantified from (3-3) and (3-4):

$$E_a = \sqrt{B^2 + C^2} - 1 \quad (3-6)$$

$$E_p = \tan^{-1} \frac{B}{C} \quad (3-7)$$

### 3.1.1 The Proposed Algorithm

A flowchart of the proposed algorithm can be seen in Figure 3-1. As mentioned before, three valid test points are needed to find the two unknown correction parameters  $B$  and  $C$ . Examining and ensuring the validity of the test points will be discussed in the next subsection. The values of the test points are arranged in two separate vectors: the vector of the  $U_S(\theta)$  values  $\mathbf{P}_S$  and the vector of the  $U_C(\theta)$  values  $\mathbf{P}_C$ , where  $P_S[i]$  and  $P_C[i]$  represent the values of  $U_S(\theta)$  and  $U_C(\theta)$  of the  $i$ th test point respectively. The absolute value vector of the corrected cosine signal  $|\mathbf{P}_{AC}|$  can be calculated from:

$$|\mathbf{P}_{AC}| = \sqrt{A^2 - \mathbf{P}_S^2} \quad (3-8)$$

It can be noticed from (3-5) and (3-6) that the phase error  $E_p$  can be completely corrected by only controlling the variable  $B$ . At the correct value of  $B$ , there will be a certain value of  $C$  such that:

$$P_C[i] + B \cdot P_S[i] = C \cdot P_{AC}[i] \quad (3-9)$$

is valid for any test point  $i$ . Taking two different test points,  $i$  and  $j$ :

$$C = \frac{P_C[i] + B \cdot P_S[i]}{P_{AC}[i]} = \frac{P_C[j] + B \cdot P_S[j]}{P_{AC}[j]} \quad (3-10)$$

which leads to:

$$B = \frac{P_{AC}[i] \cdot P_C[j] - P_{AC}[j] \cdot P_C[i]}{P_{AC}[j] \cdot P_S[i] - P_{AC}[i] \cdot P_S[j]} \quad (3-11)$$

However,  $P_{AC}[i]$  and  $P_{AC}[j]$  are not known. What is known is their absolute values from (3-8). This results in two solutions for (3-11), the first solution  $B_1$  is true when both  $P_{AC}[i]$  and  $P_{AC}[j]$  have the same sign, and the second solution  $B_2$  is when  $P_{AC}[i]$  and  $P_{AC}[j]$  have different signs:

$$\begin{aligned} B_1 &= \frac{|P_{AC}[i]| \cdot P_C[j] - |P_{AC}[j]| \cdot P_C[i]}{|P_{AC}[j]| \cdot P_S[i] - |P_{AC}[i]| \cdot P_S[j]} \\ B_2 &= -\frac{|P_{AC}[i]| \cdot P_C[j] + |P_{AC}[j]| \cdot P_C[i]}{|P_{AC}[j]| \cdot P_S[i] + |P_{AC}[i]| \cdot P_S[j]} \end{aligned} \quad (3-12)$$

Each solution will give a corresponding  $C$  value, which can be calculated using:

$$\begin{aligned} C_1 &= \min\left(\frac{|P_C[i] + B_1 \cdot P_S[i]|}{|P_{AC}[i]|}, \frac{|P_C[j] + B_1 \cdot P_S[j]|}{|P_{AC}[j]|}\right) \\ C_2 &= \min\left(\frac{|P_C[i] + B_2 \cdot P_S[i]|}{|P_{AC}[i]|}, \frac{|P_C[j] + B_2 \cdot P_S[j]|}{|P_{AC}[j]|}\right) \end{aligned} \quad (3-13)$$

In fact, the two terms in the minimum function used to calculate  $C_1$  and  $C_2$  should be equal except in the case when either  $P_{AC}[i]$  or  $P_{AC}[j]$  is zero, in which one of the terms will go to infinity. This will happen if one of the test points is taken on an angle of  $90^\circ$  or  $270^\circ$ . The minimum function is used to account for this case. Now, unless either  $P_{AC}[i]$  or  $P_{AC}[j]$  is zero, there will be two different  $B_1$  and  $B_2$  values. One is right and the other is false. For this reason, a third point  $k$  is needed to test each solution and determine the correct one. If the  $B$  and  $C$  values are correct, then according to (3-9):

$$\left| \frac{P_C[k] + B \cdot P_S[k]}{C} \right| = |P_{AC}[k]| \quad (3-14)$$

So, the error of each solution can be calculated by:

$$\begin{aligned} E_1 &= \left| \left| \frac{P_C[k] + B_1 \cdot P_S[k]}{C_1} \right| - |P_{AC}[k]| \right| \\ E_2 &= \left| \left| \frac{P_C[k] + B_2 \cdot P_S[k]}{C_2} \right| - |P_{AC}[k]| \right| \end{aligned} \quad (3-15)$$

The solution that has the minimum error is the correct one. It can be seen that the accuracy of the solution directly depends on the accuracy of the test points. In the ideal case of perfect measurement of test points, the detection and correction will be perfect. In order to increase accuracy, the steps of the algorithm will be repeated three times for all possible combinations of two test points out of three to be used as the  $i$ th and  $j$ th test points in (3-12) and (3-13). The test point left in each time is used as the  $k$ th test point in (3-15). The  $B$  and  $C$  values of the three solutions will be then averaged to obtain the final one. To do that in software, the indices  $i$ ,  $j$  and  $k$  are saved in three 3-elements vectors. The values of first elements of these vectors represent the values of the indices in the first iteration as shown in the flowchart in Figure 3-1.

Using the obtained  $B$  and  $C$  values, phase and amplitude errors can be estimated using (3-6) and (3-7) and can be corrected in run time using (3-5). Since amplitude and phase errors are not expected to change rapidly under the normal conditions, it is not recommended that the algorithm runs repeatedly during the conversion (i.e., while the converter associated with the transducer is determining the mechanical position  $\theta$  from the transducer signals). Instead, it could be run every few hours or even days, depending on the anticipated rate of change of the unbalance.

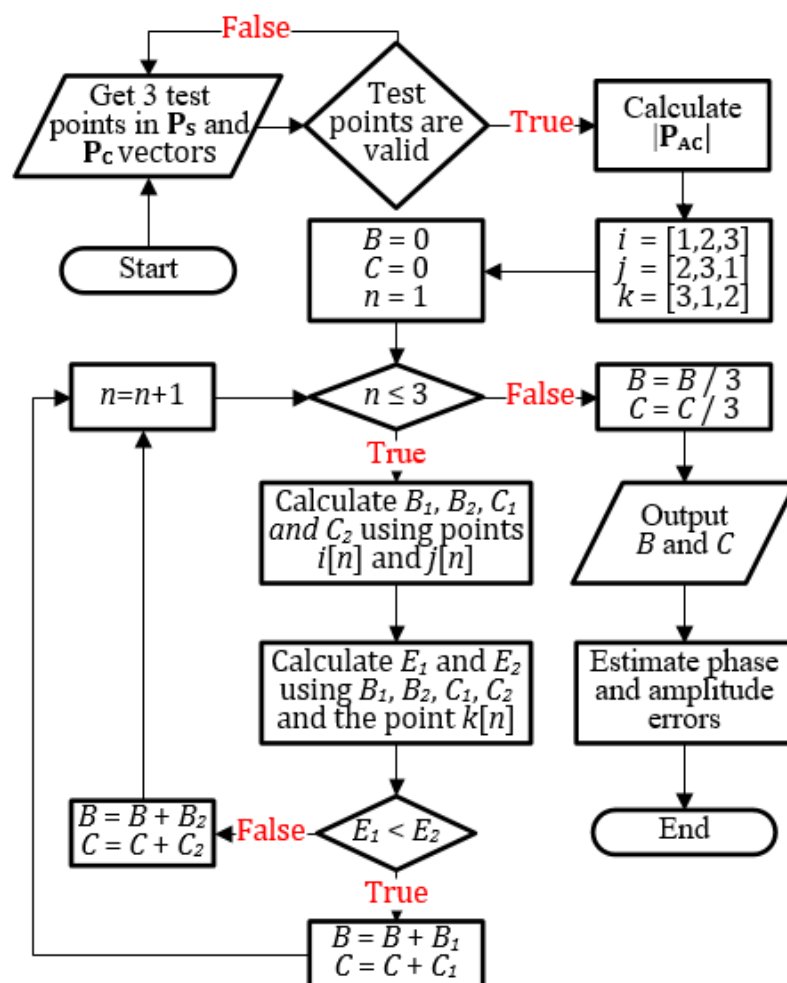


Figure 3-1: A flowchart of the suggested algorithm.

### 3.1.2 Validity of the Test Points

It can be seen from (3-12) and (3-13) that the three test points must be different. This is because if two are at the same angle,  $B_1$  would be  $0/0$ , which is undefined, and  $B_2$  would be equal to  $-P_C[i]/P_S[i]$ , which will result in  $C_2 = 0$ . Similarly, no two test points should be taken at angles  $180^\circ$  apart. In the latter case, because  $\sin(\theta + 180) = -\sin \theta$  and also  $\cos(\theta + 180) = -\cos \theta$ ,  $B_1$  would be equal to  $-P_C[i]/P_S[i]$  leading to  $C_1 = 0$ , and  $B_2$  would be undefined. To avoid this scenario, the algorithm performs a very simple check on the three test points. The check basically ensures that the absolute values of  $P_S[i]$  and  $P_C[i]$  are not equal to the corresponding absolute values of any other test point, which also does not require knowing the angles at which the test points were taken.

## 3.2 Simulation

In order to test the algorithm and verify the theory behind it, a simulation was done using MATLAB. A flowchart of the simulation is shown in Figure 3-2. The simulation imposes both amplitude and phase errors to the SCS and then the algorithm is run to estimate the errors and calculate the correction parameters  $B$  and  $C$ . Assuming that the amplitude error  $E_a$  is less than  $\pm 10\%$  of the maximum amplitude  $A$  and that the phase error  $E_p$  is between  $\pm 10^\circ$ , which is a very reasonable assumption, the simulation

program iterates over the range of phase error with a small step size. In each iteration, another iteration over the range of amplitude error with a small step size also occurs. For every combination of amplitude and phase errors, three test points are chosen at three random angles. The points are checked for validity and then the algorithm runs to produce the needed correction parameters. Then these parameters are used to correct the unbalanced signal and give  $\tilde{U}_c(\theta)$  as in (3-5). The corrected signal is entered to a function in MATLAB named "atan2" that estimates the angle from its sine and cosine values using arc tangent operation combined with logic based on the sign of the two orthogonal components.

The absolute error is then calculated as:  $|\text{atan2}(U_s(\theta), \tilde{U}_c(\theta)) - \theta|$ . This conversion error is calculated for  $\theta$  from 0 to  $360^\circ$  and the maximum is taken at each amplitude and phase errors combination. The result of simulation is a three-dimensional graph of the Maximum absolute conversion error after correction as in Figure 3-3(b) which shows the error for a total of 40401 amplitude and phase errors combinations. The performance of the algorithm is seen when comparing the conversion error after correction with the error before correction shown in Figure 3-3(a). It can be seen that there is almost no error due to the algorithm. The present error can be completely attributed to the digital rounding of the computer processor.

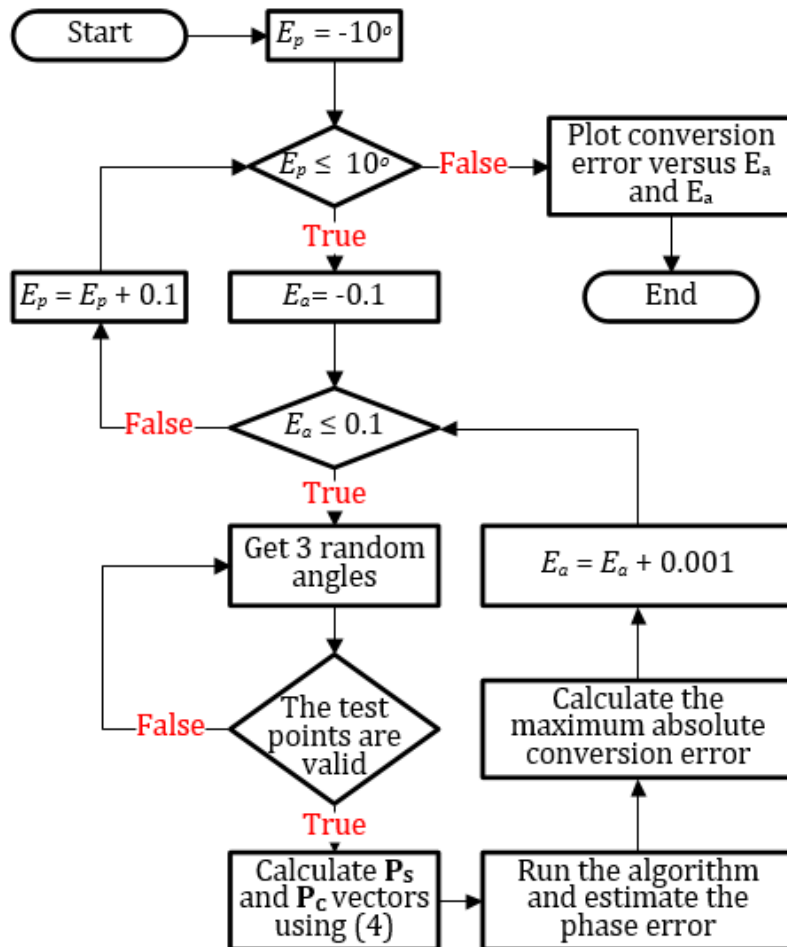
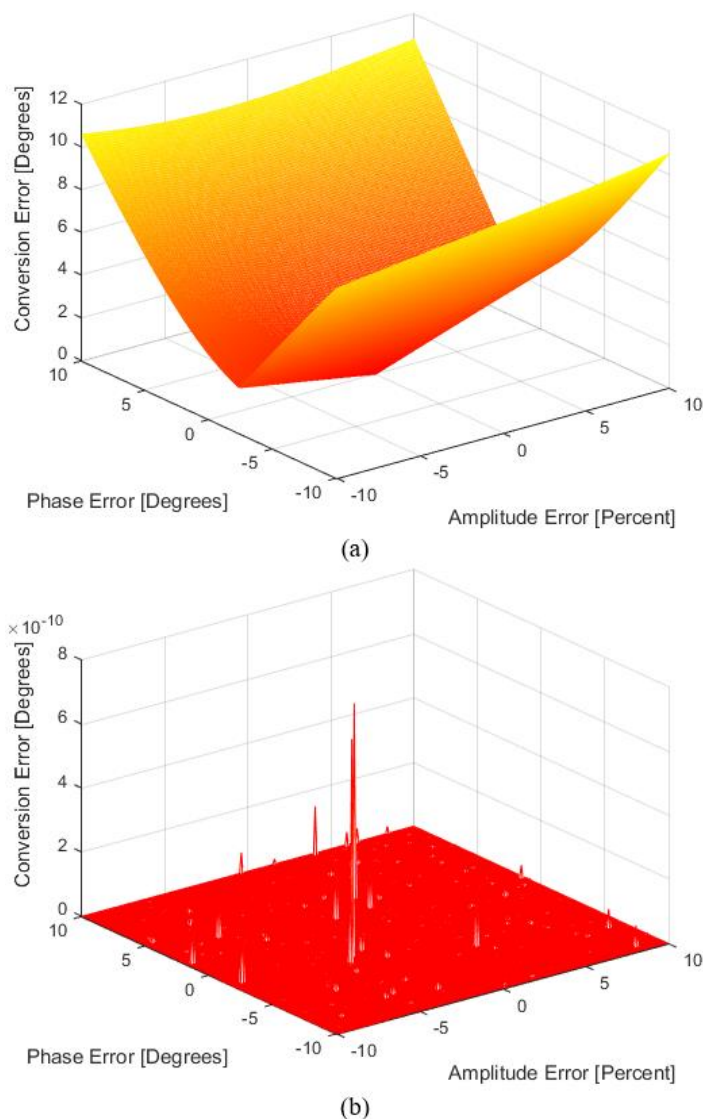


Figure 3-2: A flowchart of the simulation program.



**Figure 3-3: Maximum absolute conversion error using arctangent method. (a) before imbalances correction. (b) after imbalance correction.**

### 3.3 Implementation and Practical Testing

The proposed correction technique was implemented and tested in several ways. It was first written in 'C' language that was used on a microcontroller. The validity of the written code was first tested by simulating an imbalanced sinusoidal encoder's outputs inside the microcontroller and then correct it using the written code to compare between the imposed imbalances and the detected imbalances. After that, the implemented technique was used on an emulated imbalanced sinusoidal encoder and finally on a real sensor. The details of these tests and their results are shown in the following subsections.

#### 3.3.1 Microcontroller Implementation

The proposed correction method was successfully written in 'C' language and implemented on the ARM mbed microcontroller. For more details about the microcontroller see chapter 6. Figure 3-4 shows the pseudo-code of the implemented correction function. As discussed before, the function needs to know the maximum amplitude of the  $U_s$  output accurately. First, the function acquires three sample

points. And test them to ensure that they are valid according to what was shown in section 3.1.2. After that, the function does the calculations according to what is presented in section 3.1.1. At the end, the function returns  $B$  and  $C$  the two parameters necessary to estimate the amplitude and phase imbalances according to (3-6) and (3-7) and correct them as in (3-5).

**Require: A** --the maximum amplitude of  $U_S$

```

1: function Corrector(A)
2:    $P_S[3], P_C[3]$ 
3:    $valid \leftarrow false$ 
4:   while  $valid = false$  do
5:     Acquire three  $U_S$  and  $U_C$  samples and store them in  $P_S$  and  $P_C$ 
6:      $valid \leftarrow NOT((|P_S[0]| = |P_S[1]|) OR (|P_S[0]| = |P_S[2]|) OR (|P_S[1]| = |P_S[2]|))$ 
7:      $|P_{AC}| \leftarrow \sqrt{A^2 - P_S^2}$ 
8:      $i \leftarrow [1,2,3]$ 
9:      $j \leftarrow [2,3,1]$ 
10:     $k \leftarrow [3,1,2]$ 
11:     $B \leftarrow 0$ 
12:     $C \leftarrow 0$ 
13:    for  $n = 1 \rightarrow 3$  do
14:       $B_1 \leftarrow \frac{|P_{AC}[i[n]] \cdot P_C[j[n]] - |P_{AC}[j[n]] \cdot P_C[i[n]]|}{|P_{AC}[j[n]] \cdot P_S[i[n]] - |P_{AC}[i[n]] \cdot P_S[j[n]]|}$ 
15:       $B_2 \leftarrow -\frac{|P_{AC}[i[n]] \cdot P_C[j[n]] + |P_{AC}[j[n]] \cdot P_C[i[n]]|}{|P_{AC}[j[n]] \cdot P_S[i[n]] + |P_{AC}[i[n]] \cdot P_S[j[n]]|}$ 
16:       $C_1 \leftarrow \min\left(\frac{|P_C[i[n]] + B_1 \cdot P_S[i[n]]|}{|P_{AC}[i[n]]|}, \frac{|P_C[j[n]] + B_1 \cdot P_S[j[n]]|}{|P_{AC}[j[n]]|}\right)$ 
17:       $C_2 \leftarrow \min\left(\frac{|P_C[i[n]] + B_2 \cdot P_S[i[n]]|}{|P_{AC}[i[n]]|}, \frac{|P_C[j[n]] + B_2 \cdot P_S[j[n]]|}{|P_{AC}[j[n]]|}\right)$ 
18:       $E_1 \leftarrow \left| \frac{|P_C[k[n]] + B_1 \cdot P_S[k[n]]|}{C_1} - |P_{AC}[k[n]]| \right|$ 
19:       $E_2 \leftarrow \left| \frac{|P_C[k[n]] + B_2 \cdot P_S[k[n]]|}{C_2} - |P_{AC}[k[n]]| \right|$ 
20:      if  $E_1 < E_2$  then
21:         $B \leftarrow B + B_1$ 
22:         $C \leftarrow C + C_1$ 
23:      else
24:         $B \leftarrow B + B_2$ 
25:         $C \leftarrow C + C_2$ 
26:       $B \leftarrow B/3$ 
27:       $C \leftarrow C/3$ 
28:    return  $B, C$ 

```

**Figure 3-4: Pseudo code implementation of the correction method function.**

To make sure that the code is running correctly,  $U_S$  and  $U_C$  signals were generated inside the microcontroller with embedded phase and amplitude imbalances in them. Then the function was run to detect and correct the errors. The data of the of the two  $U_S$  and  $U_C$  signals before correction as well as the  $\tilde{U}_C$  after correction was then transferred to the computer via the serial port and received by a terminal software as in Figure 3-5 after which it was used to plot the result in MATLAB as in Figure 3-6. The results prove the functionality of the code and shows a very accurate detection and correction abilities. There is almost no error in the detection and correction. This is because the accuracy of detection and correction in this technique depends only on the accuracy of the used samples, and since the signals were originally generated inside the microcontroller there were almost no error in them.

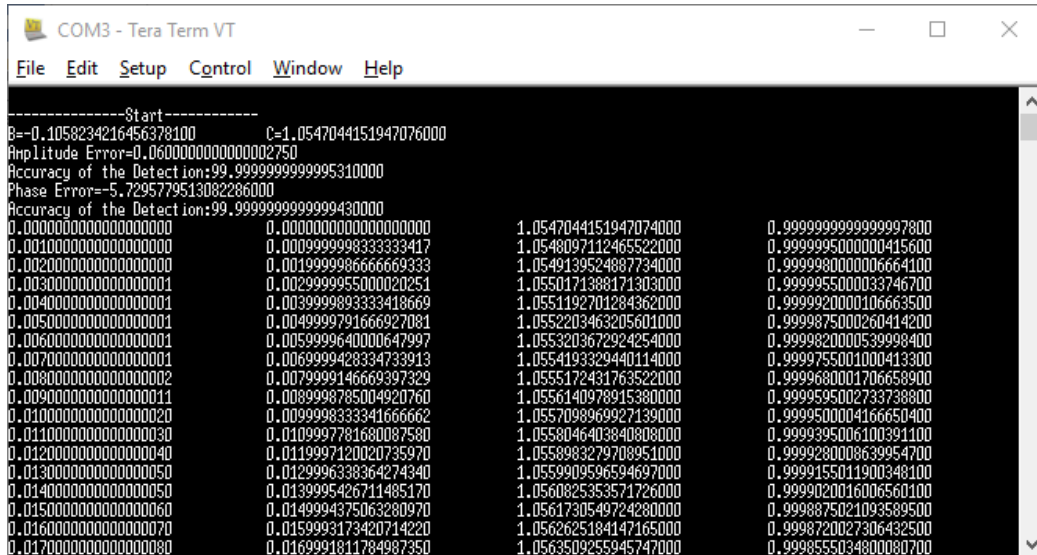


Figure 3-5: receiving the data of the test though a terminal software.

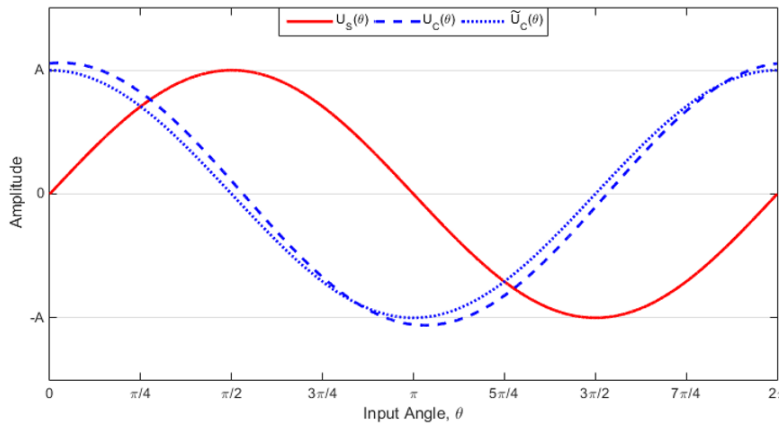


Figure 3-6: Result of the simulation done inside the microcontroller.

### 3.3.2 Emulation Results

An emulation program was designed using NI LabVIEW software to be used on the NI myDAQ platform allowing it to output two signals resembling  $U_s$  and  $U_c$  outputs of an imbalanced sinusoidal encoders with controlled amplitude and phase imbalances. More details about NI myDAQ and NI LabVIEW can be found in chapter 6. Figure 3-7 shows the emulation setup used. The setup consists of the myDAQ platform for emulation as well as two ARM mbed LPC1768 microcontrollers. Two were used because each one has only one DAC port and in some oscilloscope plots, such as in Figure 3-11, two different output signals were needed to be compared. The myDAQ platform as well as the two microcontrollers were connected to a computer that was used to control the emulator using NI LabVIEW and also to load codes to the microcontrollers and receive data from them using serial communication. Moreover, a 4-channel advanced oscilloscope was used to show the emulator and microcontrollers outputs in real time.

Figure 3-8 shows the emulation block diagram and front panel. The designed emulation allowed for setting certain amplitude and phase imbalances in the output signals of a sinusoidal encoder moving in a constant speed. The microcontroller then receives these signals through its ADC ports, performs the

imbalances detection and correction and produces the corrected signal. The emulation program also has options to shape the output signals so that they suit the receiving ADC. The ADC of the mbed LPC1768 accepts inputs from 0 to 3.3 volts. So the sinusoidal outputs were multiplied by an amplitude of 1.5V to increase the precision of the output and then shifted by an offset of 1.65V to keep adequate margin between the values of the emulator outputs and the limit values of the microcontroller's ADC. The sinusoidal outputs in this case takes a value in a range of 3 volts.

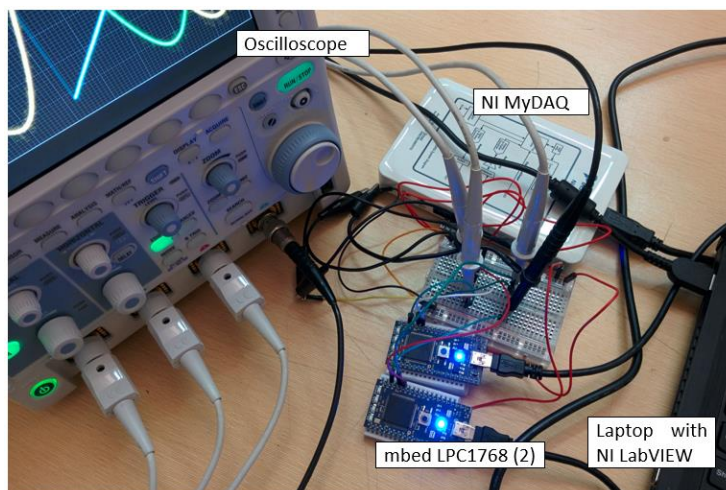
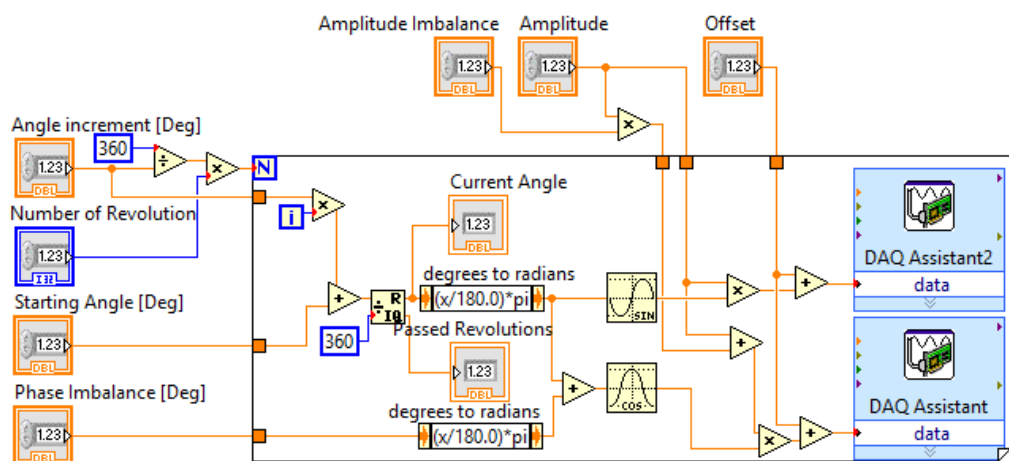
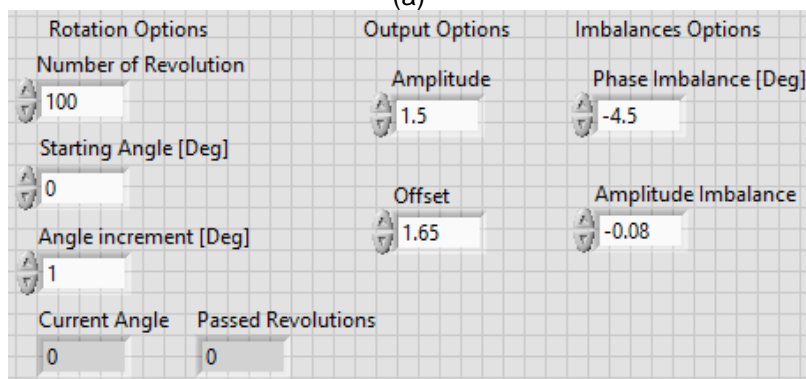


Figure 3-7: The emulation setup.



(a)



(b)

Figure 3-8: The emulation program to test the correction technique. (a) the block diagram. (b) the front panel.



The emulator was set to give  $U_S$  and  $U_C$  signals with 5.473 degrees phase imbalance and -0.025 amplitude imbalance. The code of the correction technique was added to the code of the conversion method presented in chapter 4. The correction function runs for one time to estimate the error and calculate the correction parameters. It then sends these information through serial communication to the computer and then these parameters are used to correct the signals in each conversion iteration.

In the first test case, a phase imbalance was introduced to the signals without an amplitude error. For instance, the phase imbalance was set on 9 degrees, which is a relatively large imbalance. Figure 3-9 shows the detected errors by the algorithm running on the microcontroller. It can be seen that the error in detecting the amplitude imbalance is very low. The accuracy of detecting the phase imbalance is 95.02%. Which is very acceptable given that detection depends heavily on the accuracy of the samples which is affected greatly by the DAC and ADC resolution discussed in section 4.3.2. Figure 3-10 shows the imbalance signals outputted from the NI myDAQ and the corrected signal outputted from the microcontroller as seen from an oscilloscope. Figure 3-11 shows the estimated angle using the method presented in chapter 4 ( $N=8$  using small angle approximation) before and after signal correction. The estimated angle was outputted through the microcontroller DAC as a voltage linearly proportional to the angle. Note how the estimated angle curve is not very linear when the signal is not corrected and how it is greatly more linear after the correction.

```

COM3 - Tera Term VT
File Edit Setup Control Window Help
-----Start-----
B=0.1484441582076434200
C=0.9871942243425612100
Amplitude Error=-0.0017074052757443114
Phase Error=8.5514839623444967000
  
```

Figure 3-9: The estimated imbalances and the correction parameters for case 1.

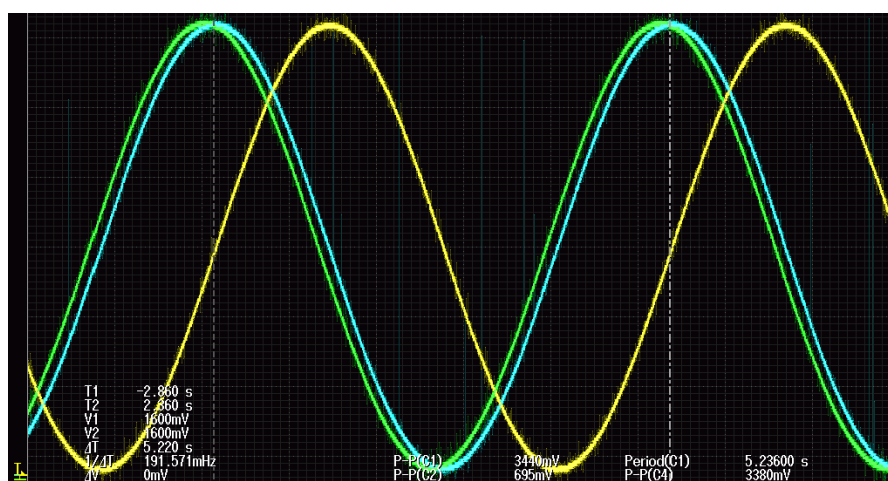


Figure 3-10: The  $U_S$  and  $U_C$  signals (yellow and green respectively) and the corrected  $\tilde{U}_C$  signal (blue) for case 1.

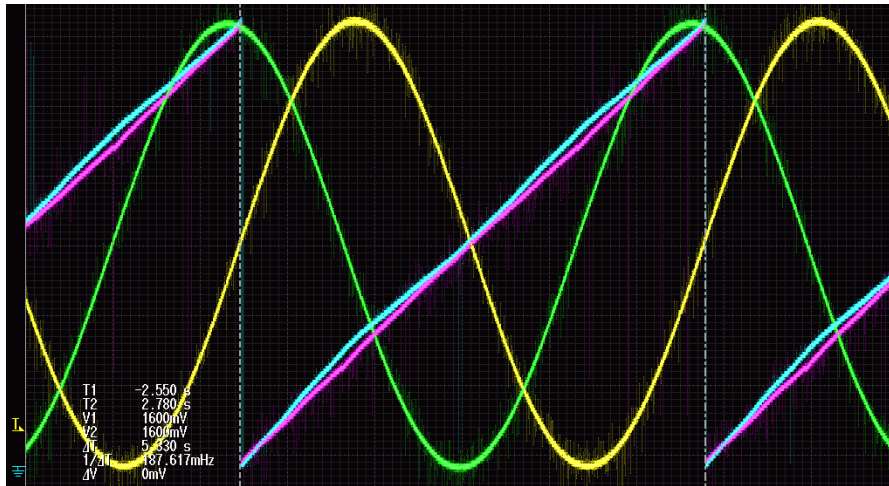


Figure 3-11: The  $U_S$  and  $U_C$  signals (yellow and green respectively), the estimated angle before correction (blue) and the estimated angle after correction signal (pink) for case 1.

On the second test case, an amplitude error was added to the emulator signals without adding a phase error. In one test, the amplitude imbalance was set to be -0.05. Figure 3-12 shows the detected errors by the algorithm running on the microcontroller. Here also, the error in detecting the phase imbalance is very low. The accuracy of detecting the amplitude imbalance is high enough at 92.06%. Figure 3-13 shows the imbalance signals compared to the corrected signal and Figure 3-14 shows the estimated angle using the method presented in chapter 4 (N=8 using small angle approximation) before and after signal correction. Again, the nonlinearity caused by the presence of the amplitude imbalance can be clearly seen, but it is not as high as that caused by the phase imbalance above.

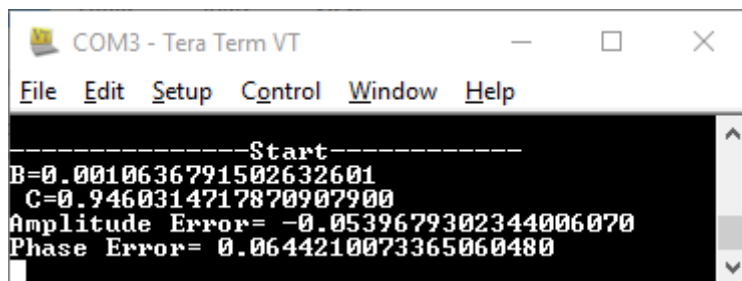


Figure 3-12: The estimated imbalances and the correction parameters for case 2.

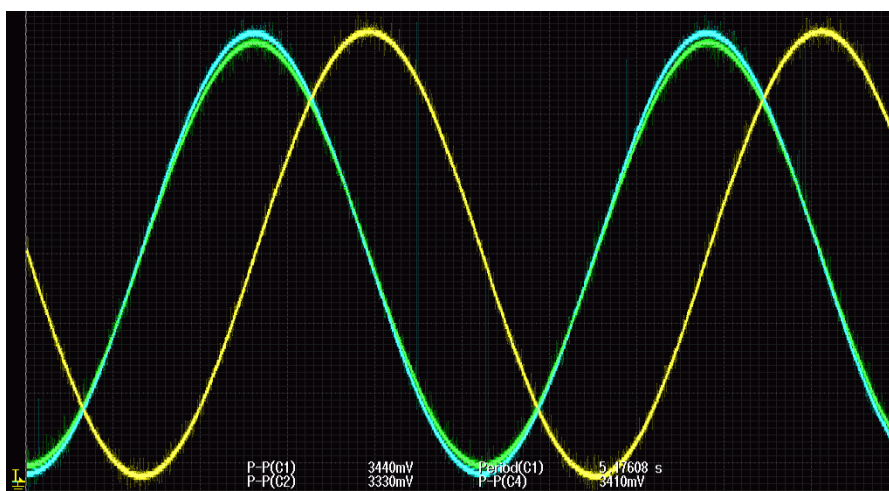


Figure 3-13: The  $U_S$  and  $U_C$  signals (yellow and green respectively) and the corrected  $\tilde{U}_C$  signal (blue) for case 2.



Figure 3-14: The  $U_S$  and  $U_C$  signals (yellow and green respectively), the estimated angle before correction (blue) and the estimated angle after correction signal (pink) for case 2.

Finally, both amplitude and phase imbalances were added to the emulator signals, shown in Figure 3-15 are the detection results when an amplitude imbalance of -0.08 and a phase imbalance of -4.5 degrees were added to the signals. The detection accuracy is high for both imbalances at 98.75% and 99.76 for phase and amplitude errors respectively. Figure 3-16 shows the imbalance signals compared to the corrected signal and Figure 3-17 shows the estimated angle using the method presented in chapter 4 (N=8 using small angle approximation) before and after signal correction. Figures 3-10, 3-13 and 3-15 practically demonstrate the importance of correcting phase and amplitude imbalances to avoid significant errors in the estimated angle.

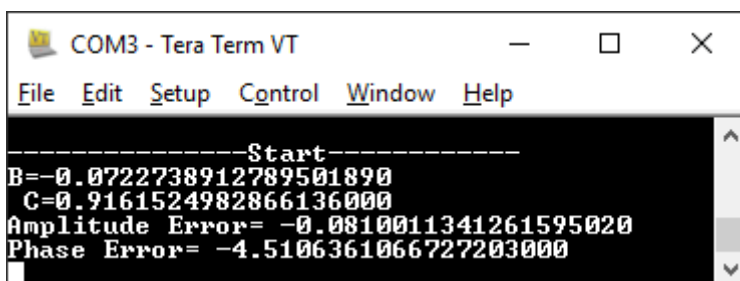


Figure 3-15: The estimated imbalances and the correction parameters for case 3.

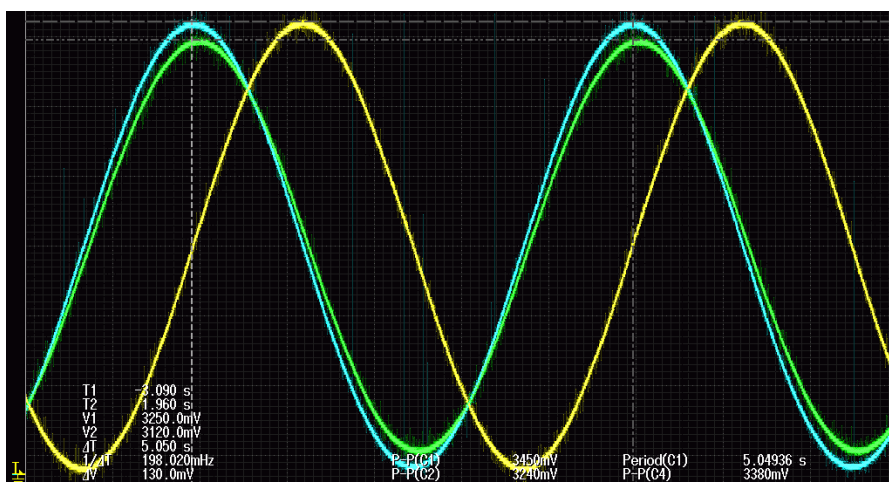


Figure 3-16: The  $U_S$  and  $U_C$  signals (yellow and green respectively) and the corrected  $\tilde{U}_C$  signal (blue) for case 3.

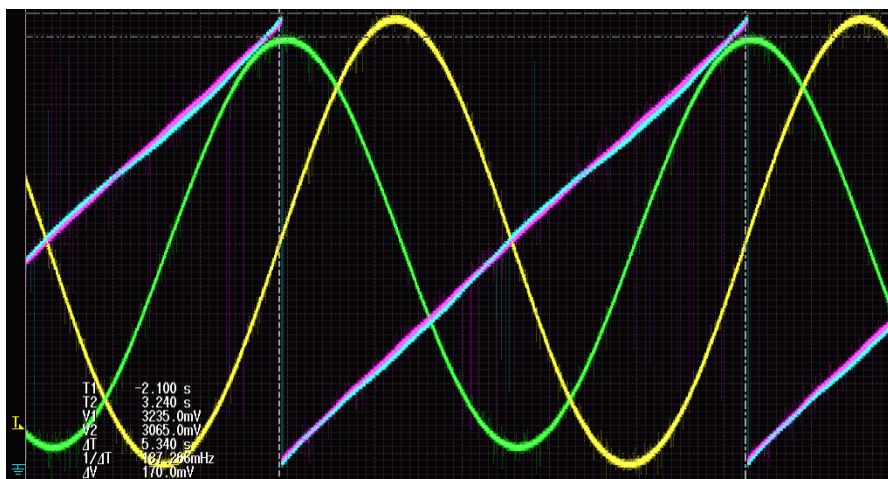


Figure 3-17: The  $U_S$  and  $U_C$  signals (yellow and green respectively), the estimated angle before correction (blue) and the estimated angle after correction signal (pink) for case 2.

### 3.3.3 Practical Usage

The correction technique was also used on the test bench setup described in chapter 6 which contains a hall-effect sinusoidal encoder. The algorithm detected an amplitude imbalance of 1.22% and a phase imbalance of 1.126 degrees. Although the accuracy of this detection cannot be determined because the true imbalances are not known, the correction parameters greatly improved the conversion error of the system as seen in Figure 3-18. It shows the conversion error of a fixed angle (about 120 degrees) before and after applying the correction algorithm. It can be seen that before correction, the error signal has an offset of around -1 degrees. The offset totally disappeared after the correction and the error signal became centered at zero as it should be.

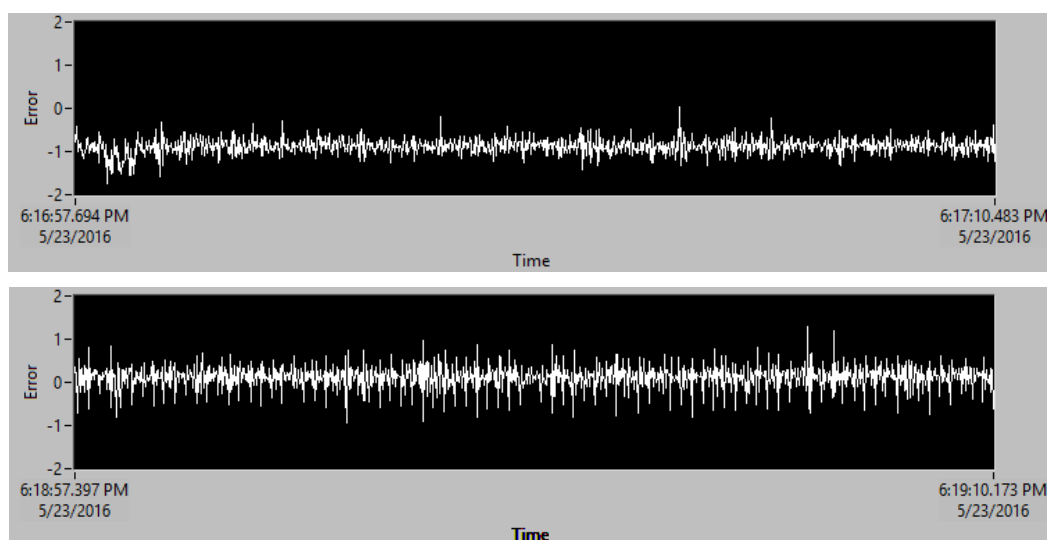


Figure 3-18: Conversion Error for a fixed angle before (Up) and after (bottom) applying the signal correction algorithm.

# Chapter 4 OPEN-LOOP CONVERTER

In this section, a new method for resolver-to-digital conversion that is completely independent of the amplitude of the SCS is introduced. The method employs the pseudo-linear part of  $\tan \theta$  by shifting it to an angle close to the measured angle. The error is then significantly and efficiently reduced by one of two options: using the small angle approximation principle, or adding a very small LUT (as small as a 4-element LUT). In addition, an algorithm is presented for designing a digital circuit that is able to locate the measured angle  $\theta$  in one of  $N$  equal segments in the circle angular interval. This makes it possible to increase the complexity in order to get a higher accuracy. Furthermore this algorithm may be adapted to other previously-reported methods that rely on the principle of pseudo-linearity in order to simplify their associated logic circuitry.

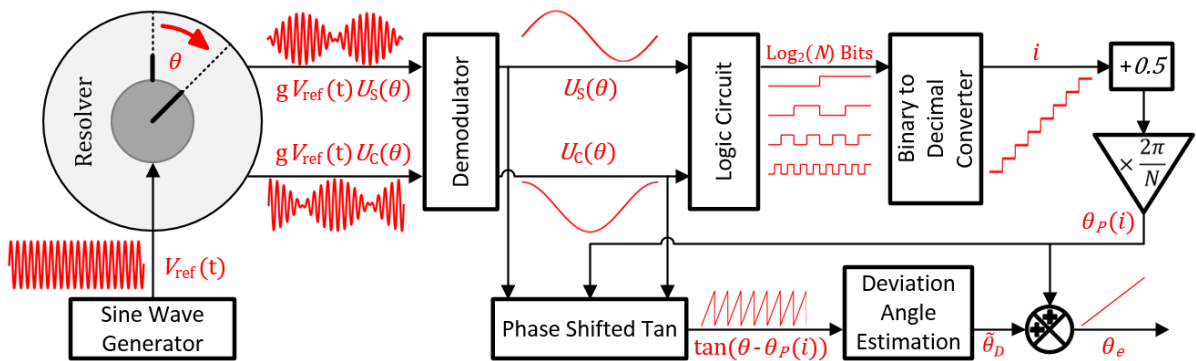
## 4.1 Theoretical Explanation

Figure 4-1 depicts the block diagram of the proposed method when used with resolvers which require demodulation of its output signals; for other types of sinusoidal encoders, the excitation and demodulation parts are not required. In order to estimate the measured angle  $\theta$  from (1-1) assuming that the sensor's outputs are balanced, the circle angular interval (from 0 to  $2\pi$ ) is divided into  $N$  equal sections, where  $N$  is a power of 2 greater than or equal to 4 for reasons that will become evident later. The width of one section is represented by an angle equals to  $2\pi/N$ . We define the principal angle  $\theta_p(i)$  as the angle at the center of each section such that:

$$\theta_p(i) = 2(i + 0.5)\pi/N ; i = 0,1,2 \dots, N - 1 \quad (4-1)$$

where  $i$  is the section number to which  $\theta$  belongs;  $i$  is identified through a digital circuit that will be described in the next section. After knowing the principal angle  $\theta_p(i)$  which represents a "coarse" measure of  $\theta$ , the next step is to estimate the "deviation angle" which is the difference between the true angle and the principal angle  $\theta_D = \theta - \theta_p(i)$ . This is done by generating the phase shifted tangent (PST) simply by dividing two weighted sums of  $U_S(\theta)$  and  $U_C(\theta)$ :

$$\tan(\theta_D) = \tan(\theta - \theta_p(i)) = \frac{\cos \theta_p(i) \times U_S(\theta) - \sin \theta_p(i) \times U_C(\theta)}{\sin \theta_p(i) \times U_S(\theta) + \cos \theta_p(i) \times U_C(\theta)} \quad (4-2)$$



**Figure 4-1: Block diagram of the proposed Phase Shifted Tangent method applied to a resolver.**

The term in the numerator represents a shifted sine, and conversely the term in the denominator represents a shifted cosine; both are determined using trigonometric identities. It is evident that the ratiometric technique results in a normalized signal (4-2) as the resultant tangent function is totally independent of  $A$ . Because the tangent function is a periodic function with a period equal to  $\pi$ :

$$\tan(\theta - \theta_p(i)) = \tan\left(\theta - \theta_p\left(i - \frac{N}{2}\right)\right); \quad N/2 \leq i \leq N - 1 \quad (4-3)$$

This means that for  $N$  sections,  $\tan\theta$  is shifted only to one of  $N/2$  angles, namely  $\theta_p(0)$  to  $\theta_p(N/2 - 1)$ . The output of this shifted tangent is within the interval  $[-\tan \pi/N, \tan \pi/N]$ , so  $N$  must be more than 2 to avoid the singularities. This will guarantee that the segments of the shifted cosine used to determine the tangent are never equal to zero as illustrated in Figure 4-2. Note that converters based on the classical tangent method suffer from this problem and resort to the determination of tangent and cotangent as described in section 2. Now the angle  $\theta_D$  represents a “fine” measure of the unknown angle  $\theta$  within a given section, and when combined with “coarse” measure yields a measure of the angle within  $[0, 2\pi]$ :

$$\theta = \theta_p(i) + \theta_D = \theta_p(i) + \tan^{-1}[\tan(\theta - \theta_p(i))] \quad (4-4)$$

In order to avoid the need for determining the arctangent in (4-4), an approximation of the deviation angle  $\tilde{\theta}_D$  from the PST may be determined in one of two ways: (a) using small angle approximation, and (b) using LUT. This leads to an estimated measure  $\theta_e$  of the angle whose deviation from the true angle  $\theta$  depends on the approximation method and its complexity as discussed below.

#### a. Small Angle Approximation Method

If we use the small angle approximation of the tangent of small angles (i/e., when  $N$  is sufficiently large), the pseudo-linear PST signal can be used to calculate a simplified estimation  $\theta_{es}$  of the measured angle:

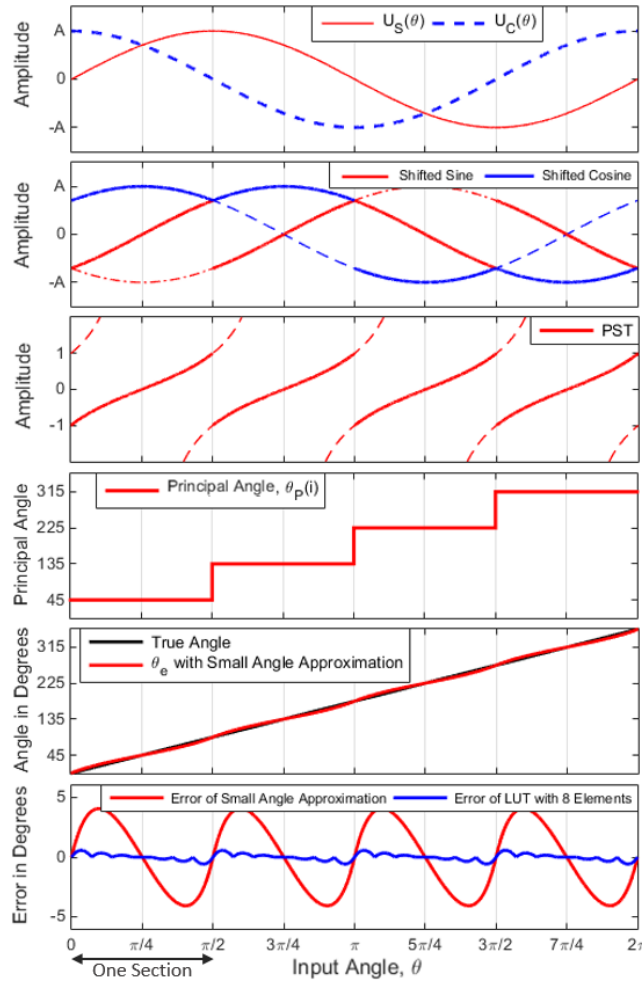
$$\theta_{es} = \theta_p(i) + \tilde{\theta}_D = \theta_p(i) + K_N \tan(\theta - \theta_p(i)) \quad (4-5)$$

Where  $K_N$  is a constant whose value is determined as described below. The accuracy of the linear estimation of  $\theta_D$  from the PST depends on the linearity of its pseudo-linear segments; this linearity decreases significantly as we increase the interval (i.e., decrease  $N$ ). In other words, the error in determining the angle  $\theta$  is heavily dependent on the number  $N$  of sections. Due to the periodic nature of the segments of the PST, the error analysis can be limited to the first interval  $[0, 2\pi/N]$ . In this interval (identified by  $i = 0$ ), the error is given by:

$$\begin{aligned} E_{es}(\theta, N) &= \theta_{es} - \theta = \theta_p(0) + K_N \tan(\theta - \theta_p(0)) - \theta \\ &= \pi/N + K_N \tan(\theta - \pi/N) - \theta \end{aligned} \quad (4-6)$$

With  $K_N = 1$ , the maximum error happens at the border angles (multiples of  $2\pi/N$ ) at which furthest deviation from linearity occurs and is equal to  $\tan(\pi/N) - \pi/N$ . To decrease this maximum error, we force a zero error at the border angles by choosing  $K_N$  to be:

$$K_N = \pi \cot(\pi/N) / N \tag{4-7}$$



**Figure 4-2: Simulation results for converter with N=4.**

From top to bottom: The SCS, shifted sine and cosine waves where the solid line portions represent the segments used to determine PST, PST, principal angle in degrees, estimated angle and the residual converter error.

Now, the angle at which the maximum error occurs can be calculated by differentiating (4-6) and equating the result to zero. This gives:

$$\theta_{me} = \cos^{-1} \sqrt{K_N} - \pi/N \tag{4-8}$$

Substituting (4-8) in (4-6) yields the absolute maximum error:

$$E_{es,max}(N) = |K_N \tan(\cos^{-1} \sqrt{K_N}) - \cos^{-1} \sqrt{K_N}| = \left| \sqrt{K_N - K_N^2} - \cos^{-1} \sqrt{K_N} \right| \tag{4-9}$$

Figure 4-3 shows the absolute maximum error for different values of  $N$ . At low values of  $N$ , the error is not acceptable in most applications; for example for  $N = 4$  the maximum error may exceed  $4^\circ$ . In order to decrease the maximum error, the number of sections  $N$  must be increased. It can be seen in Figure 4-3 that the maximum error versus the number of sections is represented by nearly a straight line in the log-log scale. Doubling the number of sections  $N$  will reduce the maximum error by approximately

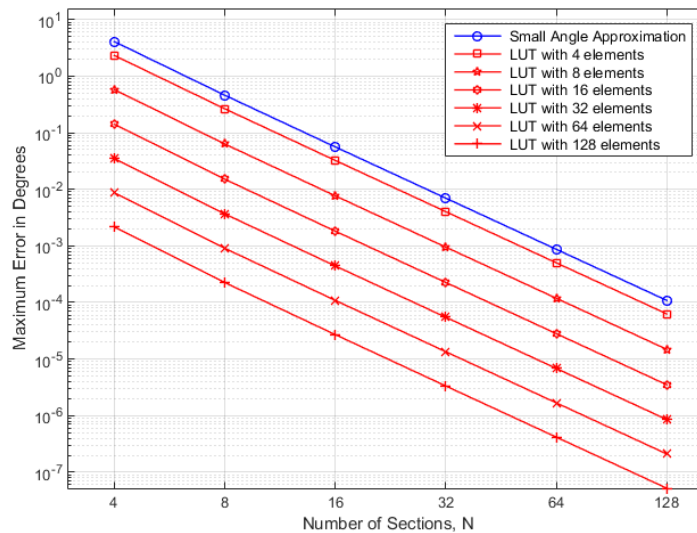
dividing it by a factor of 8. The advantages of using the small angle approximation is that no LUT or processor is used and the converter can be completely realized even using analog electronics.

### b. Look-Up Table Method

The second option for determining  $\tilde{\theta}_D$  is to combine the proposed technique involving a limited number of sections (i.e., reasonably low value for  $N$ ) with a very small one-dimensional LUT that takes the value of the PST expressed in (4-2), which is limited in the interval  $[-\tan \pi/N, \tan \pi/N]$ , and linearly interpolates between the arctangent of  $L$  equally-spaced elements in that interval. Since the error is maximum at the borders of the interval as mentioned above and the tangent function is almost linear throughout the interval, the linear interpolation error will be almost maximum at the point in the middle between the first two elements in the table as in Figure 4-2. The maximum error can then be very effectively estimated using the linear interpolation equation to get:

$$(\theta_e - \theta)_{\max} = E_{est,\max}(N, L) \cong \left| \frac{\pi}{(L-1)N} \times \left( 1 - 2 \frac{\tan \frac{(L-2)\pi}{(L-1)N} - \tan \frac{(L-3)\pi}{(L-1)N}}{\tan \frac{\pi}{N} - \tan \frac{(L-3)\pi}{(L-1)N}} \right) \right| \quad (4-10)$$

Where  $L$  is greater than 2. Figure 4-3 shows the approximated maximum error  $E_{est,\max}$  as a function of the number of sections  $N$  for different numbers of elements  $L$ . It is observed that the lines in Figure 4-3 are almost straight and parallel in the log-log scale. Doubling the number of elements  $L$  is approximately equivalent to dividing the maximum error by a factor of 4. In order to illustrate the difference between the two described methods for determining  $\tilde{\theta}_D$ , for example for  $N = 4$ , the LUT method with  $L = 8$  results in a converter error of approximately  $0.6^\circ$  compared to  $4.1^\circ$  when using the small angle approximation method.



**Figure 4-3: Maximum Error as a function of the number of sections N for small angle approximation option and the LUT option with different number of elements.**



### 4.1.1 Logic Circuit Design

For the logic circuit to indicate the position of  $\theta$  in one of  $N$  possible segments, the number of needed digital bits  $M = \text{ceil}(\log_2(N))$ . This means that it is not computationally efficient to choose  $N$  to be a non-power of two. Doing so will result in losing precision without reducing complexity.

An important requirement for the proposed converter is that it must retain the independence of measurement from the peak amplitude  $A$  of the SCS. This is guaranteed by comparing either of the sensor signals  $U_s(\theta)$  or  $U_c(\theta)$  only with zero or with a weighted version of the other. In this way, the results of such comparisons will always be independent of the maximum amplitude of the sensor signals. The bits are calculated starting from  $B_{M-1}$  to  $B_0$ . The most significant bit (MSB)  $B_{M-1}$  should divide the total circular interval by 2. This can easily be done by comparing  $\sin \theta$  with zero:

$$B_{M-1} = (\sin \theta < 0) \quad (4-11)$$

The next bit should divide  $B_{M-1}$  by 2. This may be obtained as follows:

$$B_{M-2} = B_{M-1} \oplus (\cos \theta < 0) \quad (4-12)$$

where  $\oplus$  denotes the exclusive-or logic operation. These two bits are enough for  $N = 4$ , but for higher values, additional bits are needed. With each additional bit,  $N$  doubles and therefore the maximum error decreases as shown in Figure 4-3. In the following we describe an algorithm for calculating any additional number of bits by comparing weighted versions of  $U_s(\theta)$  and  $U_c(\theta)$ .

As each bit divides the previous more significant bit by 2 throughout the circular period (see bits in Figure 4-5), it can be easily realized that  $B_{M-j}$  will switch its state at all the positions  $B_{M-j-1}$  switches at. It can be shown that each bit  $B_{M-j}$  may be determined from  $B_{M-j+1}$ :

$$B_{M-j} = B_{M-j+1} \oplus C_{M-j} \quad (4-13)$$

where  $C_{M-j}$  is a logical expression that needs to be determined. For instance, (4-12) shows that  $C_{M-2} = (\cos \theta < 0)$ . For  $C_{M-j}$  to fulfil (4-13), it must be symmetric around  $\pi$  (see  $C_{M-4}$  in Figure 4-4) and it must switch at  $2\pi m/2^j$ , where  $m = 1, 3, 5, \dots, 2^j - 1$ , which means that the total number of switching events is  $2^{j-1}$ . To achieve this,  $U_s(\theta)$  and  $U_c(\theta)$  are weighed so that they alternately cross each other at successive values of  $m$  by doing the following logic operation:

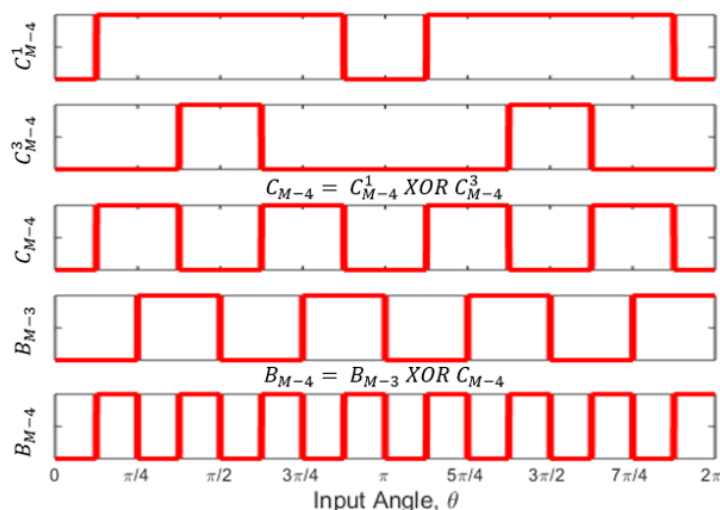
$$C_{M-j}^m = (U_s > \tan \frac{2\pi m}{2^j} U_c) \oplus (-U_s > \tan \frac{2\pi m}{2^j} U_c); \text{ where } m = 1, 3, \dots, 2^{j-2} - 1 \quad (4-14)$$

It can be seen in Figure 4-4 that  $C_{M-j}^m$  is symmetric around  $\pi$  and has four switching events at:  $\frac{2\pi m}{2^j}, \pi \pm \frac{2\pi m}{2^j}, 2\pi - \frac{2\pi m}{2^j}$ . Since the number of needed switching events in  $C_{M-j}$  is  $2^{j-1}$ , a total of  $2^{j-3}$  terms of  $C_{M-j}^m$  are XORed together to form  $C_{M-j}$  as seen in Figure 4-4:

$$C_{M-j} = C_{M-j}^1 \oplus C_{M-j}^3 \oplus \dots \oplus C_{M-j}^{2^{j-2}-1} \quad (4-15)$$

Figure 4-4 shows the steps of calculating  $B_{M-4}$ . After all the  $M$  bits are calculated, converting the resultant binary number with  $B_{M-1}$  as the MSB and  $B_0$  as the LSB to a decimal integer gives the section

number  $i$ . Figure 4-5 shows the logic circuit for  $N = 16$ . It is useful to compare this result to the logic circuits used in [18]-[19]. This is because the methods proposed in [18]-[19] and reviewed in section 2 are both very similar to the method introduced in this report and both use a digital circuits that give 4 bits typical to what the circuit shown in Figure 4-5 gives. In addition to that, they are very recent and published in well-respected journals. In terms of the logic circuit, comparing the circuit shown in Figure 4-5 with that shown in Figures 2-6 and 2-9, it can be seen that our logic circuit uses less number of components and executes in less number of clock cycles. Figure 4-5 also shows how each bit is used in calculating the next less significant bit which increases efficiency of calculations. It should be noted here that it is not possible to compare this algorithm to the circuits reported before. This is because the algorithm is able to produce the logic needed to locate the measured angle in any number of sections  $N$ , a feature that is not available any other method. So, this comparison is only for the result the algorithm gives for sixteen sections.



**Figure 4-4: Stages of Calculating  $B_{M-4}$ .**

Table 4-1 shows a detailed comparison between our our algorithm with sixteen sections and equivalent logic circuits in other methods in terms of the economic cost (number of components needed) and the computational cost (number of operations to generate the output). The merit of the proposed algorithm in both aspects can be clearly observed.

**Table 4-1: Comparison between our algorithm with sixteen sections and equivalent logic circuits in other methods.**

	Number of components						
	Scaling operations	Comparators	Absolute Value Operations	NOT Gates	AND Gates	OR Gates	XOR Gates
Our Method, $N=16$	3	8	0	0	0	0	7
Method in [18]	4	8	0	2	0	0	7
Method in [19]	0	16	4	4	10	7	0

**Number of operations needed to generate the output (Parallel Computing)**

Development of Instrumentation for High-Precision Angular Position and Speed Measurement

	$B_3$	$B_2$	$B_1$	$B_0$
Our Method, $N=16$	1	2	4	5
Method in [18]	2	3	4	5
Method in [19]	2	5	5	10

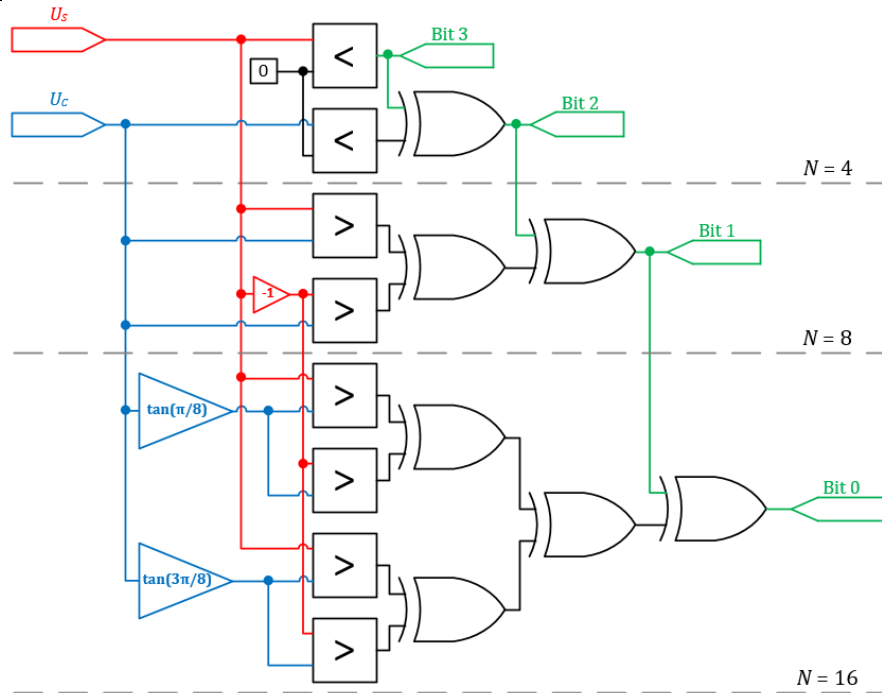


Figure 4-5: The logic circuit for four, eight and sixteen sections.

### 4.1.2 The Number of Sections and the Computational Cost

By now, the reader can see that the proposed converter solves two different problems. The first one is a determination problem, in which the converter determines the principal angle as one of finite  $N$  values. And the second one is an estimation problem, in which the converter estimates the difference between this principal angle and the measured angle. The computational cost of the converter is totally dependent on two parameters: the number of sections  $N$  and the number of elements in the look-up table  $L$  (in case that a LUT is used). It is clear that increasing the number of sections  $N$ , will result in more bits to be calculated. Each additional bit will require an increasing number of scaling operations, comparators and logic gates, which will sharply increase the computational cost and complexity. It can be seen in Figure 4-5 how the number of operations needed in the logic circuit increases with an additional bit required. Table 4-2 shows the computational cost of the logic circuit as a function of the number of sections  $N$ .

Table 4-2: Computational cost of the logic circuit as a function of the number of sections.

Number of needed bits	Scaling operations	Comparators	XOR Gates	Number of operations in Parallel implementation (FPGA)	Number of operations in series implementation (Microcontrollers)

$\log_2(N)$	$N/4 - 1$	$N/2$	$N/2 - 1$	$2$ ; $N = 4$ $\log_2(N) + 1$ ; $N > 4$	$5N/4 - 2$
-------------	-----------	-------	-----------	--	------------

The number of operations needed to calculate the PST according to (4-2) is fixed. It needs four multiplications, two additions and one division. In case of using small angle approximation, one additional multiplication is needed as in (4-5). If a LUT with  $L$  elements is used however, then the linear search will need  $2L$  comparators and  $L$  AND gates followed by the interpolation calculation that requires a fixed number of operations. The time of doing the linear search and interpolation is fixed if this implemented in parallel, where all comparator can work on the same time, but when this is implemented on a microcontroller, the time required will be the sum of time required for all operations, which is linearly proportional to  $L$ . So, in summary, the number of operations the proposed algorithm require when implemented in parallel is proportional to  $\log_2(N)$ , and when implemented in a microcontroller is linearly proportional to both  $N$  and  $L$ .

Increasing the number of sections  $N$  should decrease the maximum error as in Figure 4-3. However, it should be noted that there are some unavoidable errors, such as the quantization error. Increasing  $N$  to reach a conversion error below that would be useless. Following are all the considerations to be taken when choosing the number of segments  $N$ :

- It must be more than 2 to avoid the singularities of the tangent function.
- It must be a power of 2 as the identification of the segments of the PST is achieved using the described logic circuitry.
- It should be chosen reasonably (i.e., not too high) based on the processing power available and the precision of the other components for the reasons stated above.

### 4.1.3 Converter with 16 Sections

In order to further illustrate the proposed conversion method, a converter with  $N = 16$  (i.e., using  $\pi/8$  wide segments of  $\theta$ ) is described below. The parameters needed to calculate PST, which are the sine and cosine of the first 8 principal angles as in (4-2) and (4-3), are calculated in the first part of Table 4-3. The number of bits needed to determine the section number is  $M = \log_2(16) = 4$ . The first two most significant bits  $B_3$  and  $B_2$  can be calculated simply using (4-11) and (4-12). The second part of Table 4-3 contains the parameters needed to calculate the remaining third and fourth bits  $B_1$  and  $B_0$  according to (4-14). The two bits can then be calculated as follows:

$$B_1 = B_2 \oplus C_1 = B_2 \oplus [(U_s > U_c) \oplus (-U_s > U_c)] \quad (4-16)$$

$$\begin{aligned} B_0 &= B_1 \oplus C_0 = B_1 \oplus (C_0^1 \oplus C_0^3) \\ C_0^1 &= (U_s > 0.414U_c) \oplus (-U_s > 0.414U_c) \\ C_0^3 &= (0.414U_s > U_c) \oplus (-0.414U_s > U_c) \end{aligned} \quad (4-17)$$

The steps of calculating  $B_0$  are shown in Figure 4-4 and the logic circuit needed to calculate the four bits is shown in Figure 4-5. These four bits can then be converted to the section number  $i = B_3 \times 2^3 + B_2 \times 2^2 + B_1 \times 2 + B_0 \times 1$ . The maximum error when utilizing the small angle approximation

as in (4-5) is  $0.0561^\circ$ . Using a very small one-dimensional 8-element LUT, the maximum error reduces to  $0.007661^\circ$ , which is excellent for most practical applications.

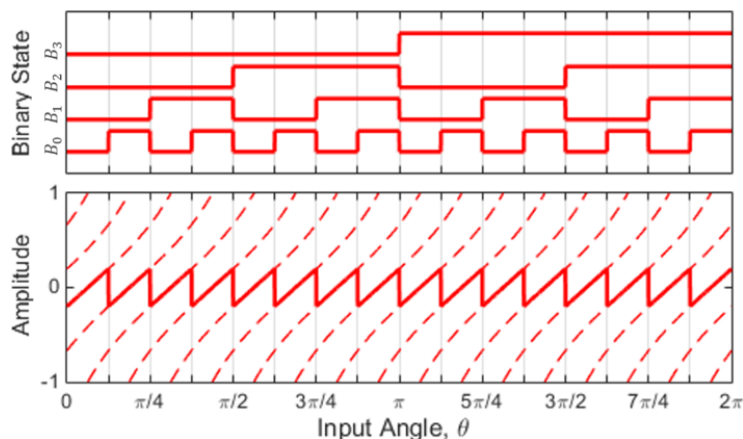
**Table 4-3: List of the parameters needed for a converter with number of sections  $N=16$ .**

<b>Part 1: Phase Shifted Tan Parameters</b>				
$i$	$\theta_p(i)$	$\tan(\theta - \theta_p(i))$	$\sin \theta_p(i)$	$\cos \theta_p(i)$
0, 8	$\pi/16, 17\pi/16$	$\tan(\theta - \pi/16)$	0.1951	0.9808
1, 9	$3\pi/16, 19\pi/16$	$\tan(\theta - 3\pi/16)$	0.5556	0.8315
2, 10	$5\pi/16, 21\pi/16$	$\tan(\theta - 5\pi/16)$	0.8315	0.5556
3, 11	$7\pi/16, 23\pi/16$	$\tan(\theta - 7\pi/16)$	0.9808	0.1951
4, 12	$9\pi/16, 25\pi/16$	$\tan(\theta - 9\pi/16)$	0.9808	-0.1951
5, 13	$11\pi/16, 27\pi/16$	$\tan(\theta - 11\pi/16)$	0.8315	-0.5556
6, 14	$13\pi/16, 29\pi/16$	$\tan(\theta - 13\pi/16)$	0.5556	-0.8315
7, 15	$15\pi/16, 31\pi/16$	$\tan(\theta - 15\pi/16)$	0.1951	-0.9808
<b>Part 2: Logic Circuit Parameters</b>				
Bit	$j$	$m$	$\tan 2\pi m/2^j$	
$B_1$	3	1	1	
$B_0$	4	1	0.4142	
$B_0$	4	3	2.4142	

Note that  $\sin \theta_p(i)$  and  $\cos \theta_p(i)$  are only for the first section number  $i$  in each row since the shifted tan is the same for the two sections.

## 4.2 Simulation

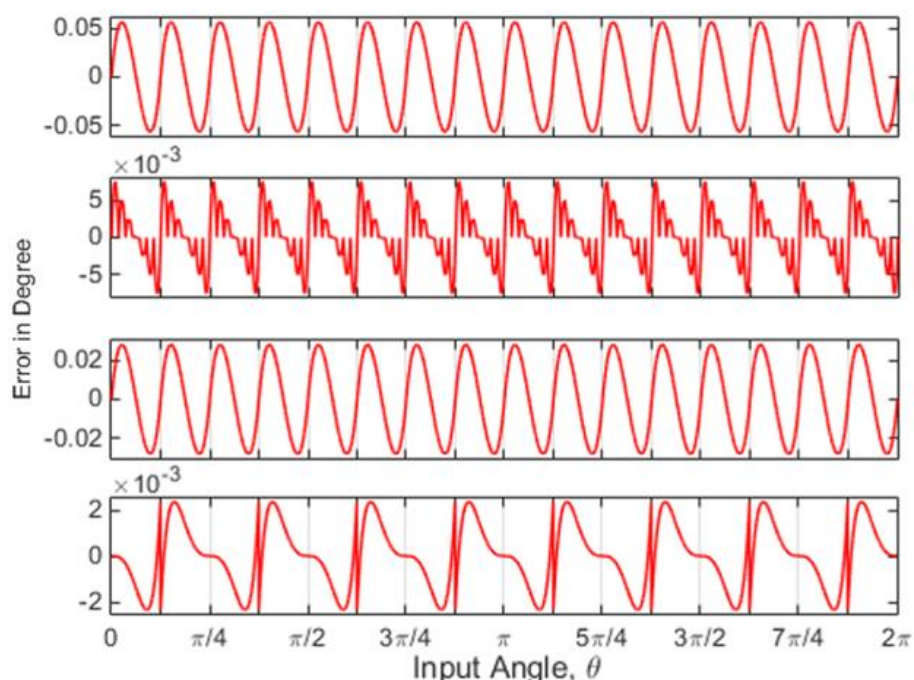
The method was successfully simulated using MATLAB/Simulink for number of sections  $N = 16$  with small angle approximation and with a LUT that has a number of elements  $L = 8$ . The four bits as well as the PST signal are shown in Figure 4-6 as a function of the input angle  $\theta$ .



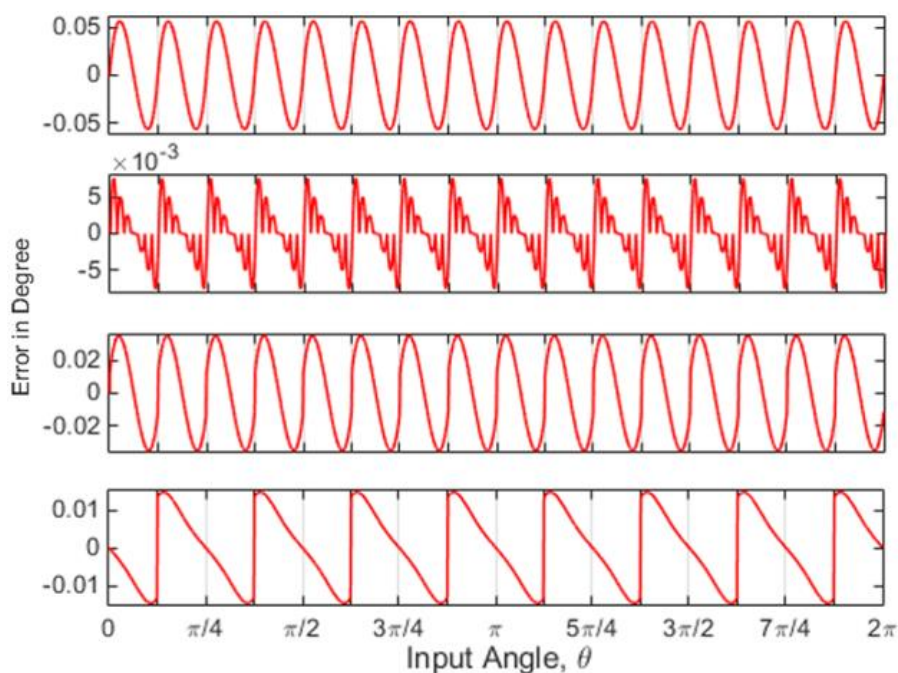
**Figure 4-6: Simulation result for converters with number of sections  $N=16$ . From top to bottom: The four bits and the PST in solid line as a function of the input angle  $\theta$ .**

Furthermore, the error of the converter was compared with two other similar methods that were simulated using the same software to emphasize the amplitude independence of the proposed method. The first one is the method proposed in [18] which was reviewed in section 2.2. In this method, the  $2\pi$  interval is also divided to 16 sections. The authors pointed to the generality of the method for higher numbers of sections, but they didn't offer a way to do so. The method has a very similar concept as the method presented in this paper and has a slightly more complex logic circuit than what is given here to spot the measured angle in one of the 16 sections. However it relies on shifted sinewaves instead of tangents, which makes it very sensitive to amplitude errors. The absolute error of the method is very low provided that the amplitude is accurately known. The error is then  $0.0279^\circ$ , like in Figure 4-7, which is still much higher than the error of the proposed LUT-based method. When an error of 0.1% was introduced to the amplitude of the SCS, the absolute error increased by approximately 24% of the original error to  $0.0347^\circ$  as shown in Figure 4-8.

The second method in this comparison is the very recent one proposed in [19]. It divides the  $2\pi$  interval to 8 segments, but then treats each half segment differently, so it is equivalent to dividing to 16 segments. For that reason, four bits are calculated as in our method. However, the number of logic gates used is two times higher than the number of logic gates we use as shown in section 4.1.1. In addition to that, calculating the bits required determination of the absolute values of four signals together with the determination of the minimum and maximum of them. At first, it appears that the method has a tremendous maximum error of  $0.00235^\circ$  when the amplitude is free of error as in Figure 4-7, but because this method also depends on the pseudo-linearity of the sine function, it is very sensitive to fluctuation of  $A$ . In fact, it is much more sensitive than the method in [18] as shown in Figure 4-8. For a balanced error of 0.1% in the amplitude of the SCS, the maximum error of the method increased significantly to  $0.0147^\circ$  which represents an increase of the original error by 525%.



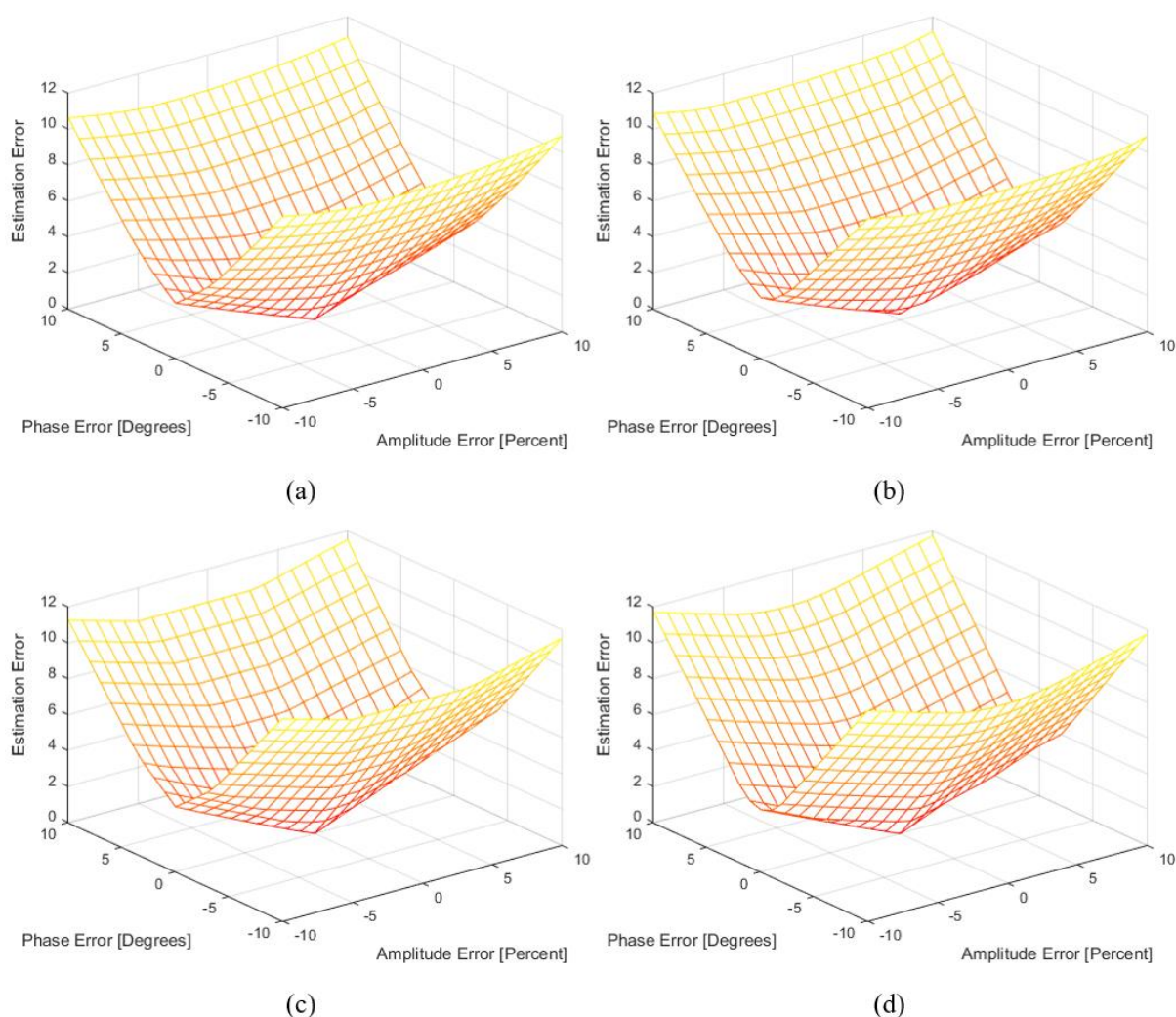
**Figure 4-7: The error of the proposed method compared to the error of other similar methods in presence of an accurate measurement of the amplitude of the SCS. From top to bottom: the proposed method with number of sections  $N=16$  with small angle approximation, the proposed method with number of sections  $N=16$  with an 8-element LUT, the method presented in [18] and the method presented in [19].**



**Figure 4-8: The error of the proposed method compared to the error of other similar methods in presence of 0.1% error in the amplitude of the SCS. From top to bottom: the proposed method with number of sections  $N=16$  with small angle approximation, the proposed method with number of sections  $N=16$  with an 8-element LUT, the method presented in [18] and the method presented in [19].**

Another simulation has been done using MATLAB codes to test the response of the three methods (the proposed one, the one in [18] and that in [19]) to amplitude and phase imbalances. The simulation

was done in the same way as the flowchart in Figure 3-3. However, instead of using the classical arctangent method, the three conversion methods were written as MATLAB functions and were used. Figure 4-9 shows the result of this simulation. It can be seen that the error curve of three curves are similar and have a “folded paper shape” with maximum conversion error occurring at the corners of the graphs (when both amplitude and phase imbalances are maximum). Both amplitude and phase imbalances affects the conversion of all methods greatly and therefore must be corrected first as shown in section 3. However, it appears that the proposed method with small angle approximation has the least worst-case conversion error of  $10.9^\circ$ . After it comes the proposed method with LUT of 8 elements at  $11.21^\circ$ . The methods proposed in [18] and [19] have slightly higher worst-case errors at  $11.53^\circ$  and  $11.76^\circ$  respectively.



**Figure 4-9: The error (in degrees) of the proposed method compared to the error of other similar methods in presence of amplitude and phase imbalances. (a) the proposed method with number of sections  $N=16$  with small angle approximation. (b) the proposed method with number of sections  $N=16$  with an 8-element LUT. (c) the method presented in [18]. (d) the method presented in [19].**

### 4.3 Implementation and Practical Testing

The proposed method was implemented and tested using several approaches. First it was implemented on the ARM mbed Microcontroller using 'C' language. Section 3.3.1 shows the Development of Instrumentation for High-Precision Angular Position and Speed Measurement



considerations taken to choose this particular microcontroller. The practical testing was performed in three levels. The first level was implementing the converter on the microcontrollers and testing it using emulated sinusoidal encoder's outputs generated internally inside the microcontroller. The second level was to emulate a sinusoidal encoder's outputs externally using the same NI myDAQ emulation platform described in section 3.3.2. The final level was to test the converter on a real sensor using a test bench designed specifically for this purpose as will be shown soon.

### 4.3.1 Microcontroller Implementation

A lot of time was invested to optimize the converter code and make it run as fast as possible on the microcontroller. The first step taken by the program is a one-time calculation of the needed parameters for the converter. These parameters are then stored in memory and accessed when needed. Accessing these parameters is a much faster operation than calculating them at each conversion execution. The parameters calculated in this step include  $N$  values of  $\theta_p(i)$  calculated for every possible  $i$  according to (4-1),  $N/2$  values of  $\cos \theta_p(i)$  and  $\sin \theta_p(i)$  to be used in calculating the PST as in (4-2) and (4-3),  $N/4 - 1$  values of  $\tan 2\pi m/2^j$  to be used as scaling factors as in (4-14) and two arrays of length  $L$  for the look-up table. Figure 4-10 shows the pseudocode of calculating these parameters. Note that the indices of arrays in this pseudo-code and the following one begin from 0 to the size of the array minus one, just like how they are in the 'C' language.

---

```

Require:  $N, L$ 
Ensure:  $N \geq 4$  AND ( $L = 0$  OR  $L > 2$ )
1: global  $M \leftarrow \log_2 N$ 
2: global  $K_N \leftarrow \pi/N/\tan(\pi/N)$ 
3: global  $\theta_p[N], \text{sines}[N/2], \text{cosines}[N/2], \text{tans}[N/4 - 1], X[L], Y[L]$ 
4: for  $j = 0 \rightarrow N - 1$  do
5:    $\theta_p[j] \leftarrow 2(j + 0.5)\pi/N$ 
6:   if  $j < N/2$  then
7:      $\text{sines}[j] \leftarrow \sin(\theta_p[j])$ 
8:      $\text{cosines}[j] \leftarrow \cos(\theta_p[j])$ 
9:    $k \leftarrow 0$ 
10:  for  $j = 3 \rightarrow M$  do
11:    for  $m = 1, 3 \rightarrow 2^{j-2} - 1$  do
12:       $\text{tans}[i] \leftarrow \tan(2\pi m/2^j)$ 
13:  for  $j = 0 \rightarrow L - 1$  do
14:     $Y[i] \leftarrow -\pi/N + 2j\pi/(L - 1)N$ 
15:     $X[i] \leftarrow \tan(Y[i])$ 

```

---

**Figure 4-10: Pseudo-code for calculating the needed parameters of the converter.**

Figure 4-11 shows the pseudo-code of the converter function assuming that all the needed parameters are already calculated as in Figure 4-10. First, the logical bits are calculated to get the section number  $i$  which is then used as an index to access the corresponding principal angle  $\theta_p(i)$  value. The value of  $i$  is then replaced by the remainder of  $i$  when divided by  $N$  and used as indices to access the scaling factors used to calculate the PST. This is done so to take advantage of the periodicity of the tangent function as in (4-3). After calculating the PST, there is two options, if the value of number of elements in the LUT  $L = 0$ , then the small angle approximation is used to estimate the deviation

angle. Else, a linear search is performed to determine the position of the PST relative to the tangents values of the LUT ( $X$  array) followed by a linear interpolation to estimate the corresponding deviation angle based on the angles values saved in the  $Y$  array. At the end the estimated angle is returned as the sum of the principal angle and the deviation angle.

After writing the code and compile it on the microcontroller, its validity was tested by generating the  $U_S(\theta)$  and  $U_C(\theta)$  internally in the microcontroller for the whole rotation interval and input them to the converter function then measuring the error between the input angle and the estimated angle. Both the input angle and the estimated angle throughout the rotational interval were transferred to the computer through serial communication and received in a terminal software as in Figure 4-12 and then plotted using MATLAB as shown in Figure 4-13. The resultant error signals are identical to the simulation results shown in Figure 4-7. In addition to that, the methods in [18] and [19] were implemented on the same microcontroller for the sake of practical comparisons. The same validation test were done on them to ensure that the code is written well. The resultant error between the internally generated input angle and the estimated angle is shown in Figure 4-14. Again, the results are identical to the simulation results in Figure 4-7, which proves that the written codes are ready to be used.

---

**Require:**  $U_S, U_C, N, L$   
**Ensure:**  $N \geq 4$  AND ( $L = 0$  OR  $L > 2$ )

```

1: function PSTConverter( $U_S, U_C, N, L$ )
2:    $Bits[M]$  --Boolean array
3:    $Bits[M - 1] \leftarrow U_S < 0$ 
4:    $Bits[M - 2] \leftarrow Bits[M - 1] \oplus (U_C < 0)$ 
5:    $k \leftarrow 0$ 
6:   for  $j = 3 \rightarrow M$  do
7:      $Bits[M - j] \leftarrow Bits[M - j + 1]$ 
8:     for  $l = 0 \rightarrow 2^{j-3}$  do
9:        $Bits[M - j] \leftarrow Bits[M - j] \oplus (U_S > \tan[k] U_C) \oplus (-U_S > \tan[k] U_C)$ 
10:       $k \leftarrow k + 1$ 
11:    $i \leftarrow 0$ 
12:   for  $j = 1 \rightarrow M$  do
13:      $i \leftarrow i + 2^{M-j} Bits[M - j]$ 
14:    $\theta_p \leftarrow \theta_{p}[i]$ 
15:    $i \leftarrow \text{mod}(i, N)$  --Modulus operation
16:    $PST \leftarrow (\cosines[i] U_S(\theta) - sines[i] U_C(\theta)) / (sines[i] U_S(\theta) + \cosines[i] U_C(\theta))$ 
17:   if  $L = 0$  then
18:      $\theta_D \leftarrow K_N PST$ 
19:   else
20:     for  $j = 1 \rightarrow L - 1$  do
21:       if  $PST \leq X[j]$  then
22:          $\theta_D \leftarrow (PST - X[j - 1])(Y[j] - Y[j - 1]) / (X[j] - X[j - 1]) + Y[j - 1]$ 
23:       break
24:   return  $\theta_p + \theta_D$ 

```

---

Figure 4-11: Pseudo-code of the PST converter function.

```

COM3 - Tera Term VT
File Edit Setup Control Window Help
-----Start-----
0.00000000000000000000000000000000 0.00000000000000000000000000000000
0.0010015593194828767 0.00100000000000000000000000000000
0.0020030222445192106 0.00200000000000000000000000000000
0.0030043907916645005 0.00300000000000000000000000000001
0.0040056669767167263 0.0040000000000000000000000000001
0.0050068528147326277 0.0050000000000000000000000000001
0.0060079503200439477 0.0060000000000000000000000000001
0.0070089615062736907 0.0070000000000000000000000000001
0.0080098883863523734 0.0080000000000000000000000000002
0.0090107329725343105 0.00900000000000000000000000000011
0.0100114972764137680 0.01000000000000000000000000000020
0.0110121833089412010 0.01100000000000000000000000000030
0.0120127930804394410 0.01200000000000000000000000000040
0.0130133286006199330 0.01300000000000000000000000000050
0.0140137918785988720 0.01400000000000000000000000000050
0.0150141849229133780 0.01500000000000000000000000000060
0.0160145097415376680 0.01600000000000000000000000000070
0.0170147683418991890 0.01700000000000000000000000000080
0.0180149627308947950 0.01800000000000000000000000000090
0.0190150949149068240 0.01900000000000000000000000000100
0.0200151668998192690 0.02000000000000000000000000000110
0.0210151806910338740 0.021000000000000000000000000120
    
```

Figure 4-12: Receiving the estimated angle (left column) and the input angle (left column).

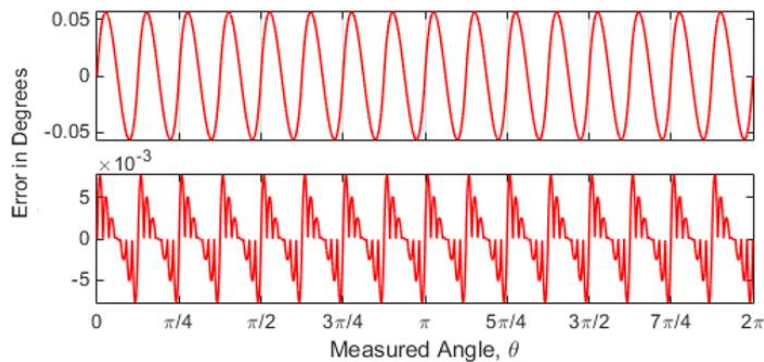


Figure 4-13: Simulation done inside the microcontroller for the proposed method. Up: the proposed method with 16 sections using the small angle approximation. Down: the proposed method with 16 sections using an 8-elements LUT.

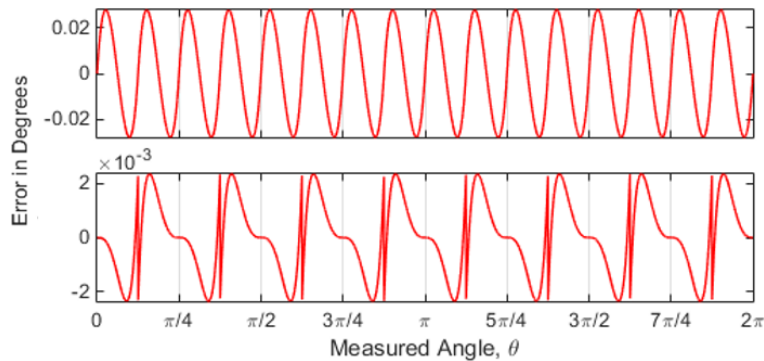
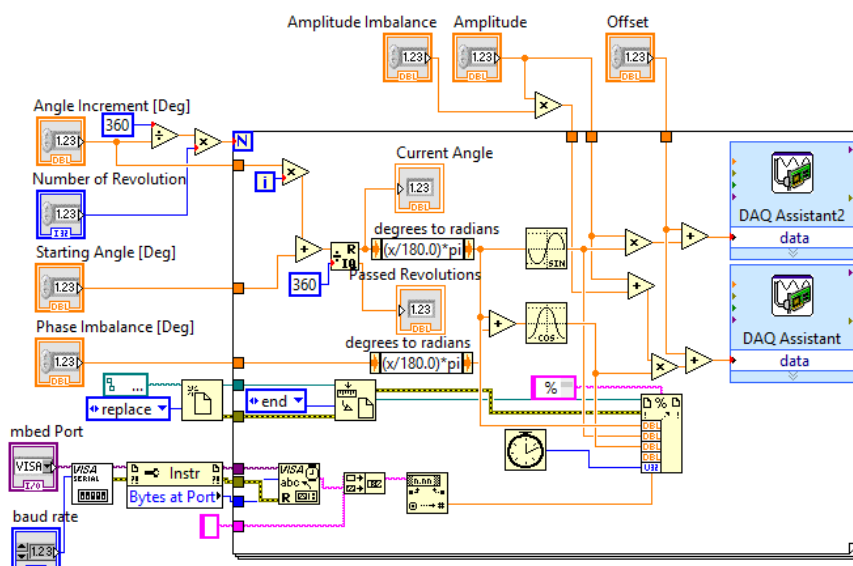


Figure 4-14: Simulation done inside the microcontroller for similar methods. Up: the method in [18]. Down: the method in [19].

### 4.3.2 Emulation Results

After ensuring that the implemented codes work well on the microcontroller, an emulation program was designed using NI LabVIEW software to be used on the NI myDAQ platform described in details in chapter 6. The designed program is in fact a developed version of the program used in emulation in section 3.3.2. The front panel of the emulation program is the same as that in Figure 3-7. Few parts were added, however, to the block diagram as shown in Figure 4-15.



**Figure 4-15: The block diagram of the emulation program.**

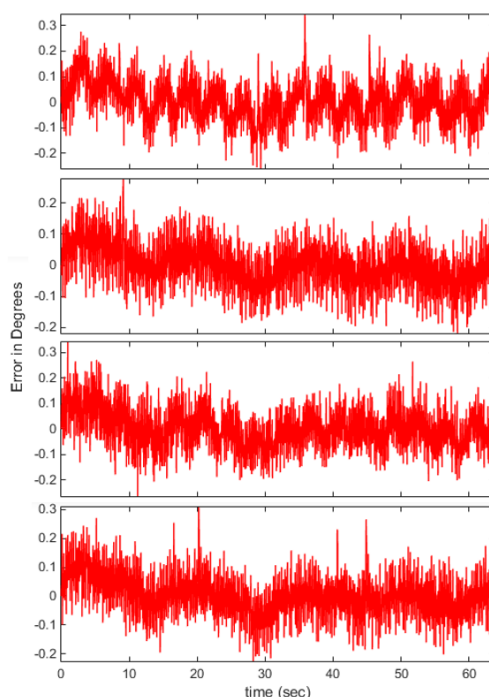
In this program  $U_S$  and  $U_C$  values of a sinusoidal encoder rotating at a constant speed are emulated. These two outputs are received by the microcontroller through its ADC. The conversion algorithm runs on the microcontroller then the estimated angle is returned by the microcontroller to the computer and received by the LabVIEW software via the serial port. The LabVIEW software writes the data of the generated and received signals including the true angle value, the estimated angle returned by the microcontroller and the elapsed time to a text file in the computer. This txt file is then read by MATLAB to extract the information and plot the error signals as shown in Figure 4-16. The emulation setup also included an oscilloscope to plot the data in real time and allow debugging any error.

What is shown in the four graphs of Figure 4-16 is random noise that cannot be used to compare these different open loop methods. This can be justified by studying the DAC and ADC errors. The DAC used in myDAQ is a 16-bit one. According to the datasheet [24], it has an absolute error of about 5 mV. The two signals outputted through the DAC are sine and cosine signals within 3V (their amplitude is 1.5V as shown in section 3.3.2). These signals are highly nonlinear. The worst case, at which a precision error would result on maximum deviation from the real value, is where they are most linear. Given this resolution of the DAC with this interval of voltage values of the sent data, this may result in worst-case error of 0.2701 degrees. On the other side, the ARM mbed LPC1768 has a 12-bit ADC with an LSB precision of 806  $\mu$ V which, in worst case scenario mentioned above, gives a maximum error of 0.0435 degrees. The total error duo to both DAC and ADC is around 0.3 degrees. The theoretical and simulation error of the four converter is lower than that and hence cannot be tested with the available precision. Unfortunately, we did not have access to higher precision equipment to perform this test.

However, the proposed converter gives higher errors when used with lower number of section. This gave the opportunity to test the converter with lower number of sections and see how close the emulation results to the theoretical calculations. Figure 4-17 shows the resultant error graphs for the converter with four and eight sections using small angle approximation and a LUT with eight elements. It can be seen that the result of the 4 sections converter is very close to the error theoretical calculations shown in Figure 4-3 and the simulation results shown in Figure 4-2. In the emulation results, the

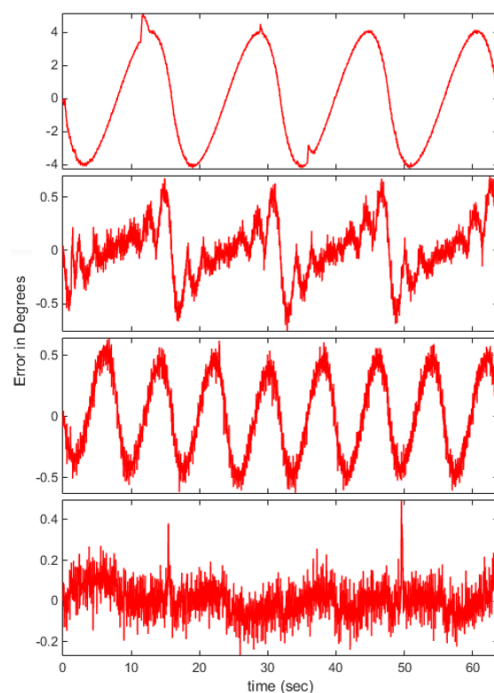
maximum error for the 4-sections converter with small angle approximation is  $5.114^\circ$  compared to a theoretical maximum error of  $4.075^\circ$ . The same converter gives a maximum error of  $0.753^\circ$  in the emulation after adding an 8-element LUT to it compared to a maximum theoretical error of  $0.578^\circ$ . The maximum error of the 8-sections converter also comes at  $0.6367^\circ$  close to the theoretical one at  $0.4594^\circ$ . The error signal of the same converter after adding an 8-element LUT, however, is noisy because the maximum theoretical error in this case is  $0.06316^\circ$  much lower than the precision of the system.

The estimated angle was also outputted through the DAC of the microcontroller as voltage linearly proportional to the angle and was plotted through oscilloscope for the 4-sections converter with small angle approximation and an 8-element LUT as shown in Figure 4-18. The effect of adding this small size LUT can be seen clearly as the estimated angle curve is much more linear in this case. In section 3.3.2, the effect of having amplitude or phase imbalances on the estimated angle produced by the converter is emulated and analyzed.



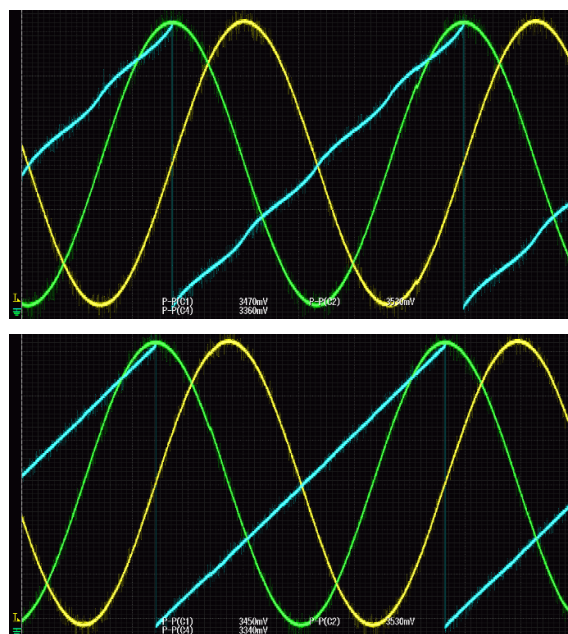
**Figure 4-16: One revolution error signals of the proposed converter with sixteen sections and similar methods.**

From top to bottom: the proposed method with number of sections  $N=16$  with small angle approximation, the proposed method with number of sections  $N=16$  with an 8-element LUT, the method presented in [18] and the method presented in [19].



**Figure 4-17: One revolution error signals for the proposed converter with lower number of sections.**

From top to bottom: the proposed converter with four sections using small angle approximation, the proposed method with four sections using an 8-element LUT, the proposed converter with eight sections using small angle approximation, the proposed method with eight sections using an 8-element LUT.



**Figure 4-18: the  $U_s$  and  $U_c$  signals (yellow and green respectively) and the estimated angle (blue).**

**Top: the proposed converter with four sections using small angle approximation. Bottom: the proposed method with four sections using an 8-element LUT.**

Another test that was performed was the step response test to see how the proposed converter responds to a step change in the angle of the sinusoidal encoder. Another simple simulation program was designed to make the emulator output the  $U_s$  and  $U_c$  of the specified angle in the front panel. The microcontroller computes the estimated angle and outputs it through its DAC as a voltage linearly

proportional to the angle. Figure 4-19 shows this very simple program. It was soon found that this test cannot be done using the emulator. This is due to the fact that the emulator updates the two DAC outputs sequentially not in parallel. The time lag between updating one of the outputs and the other is much higher than the step response needed to be measured. Figure 4-20 shows this for the proposed converter with 4-sections using small angle approximation. The lag between the two updates is about 1.5ms.

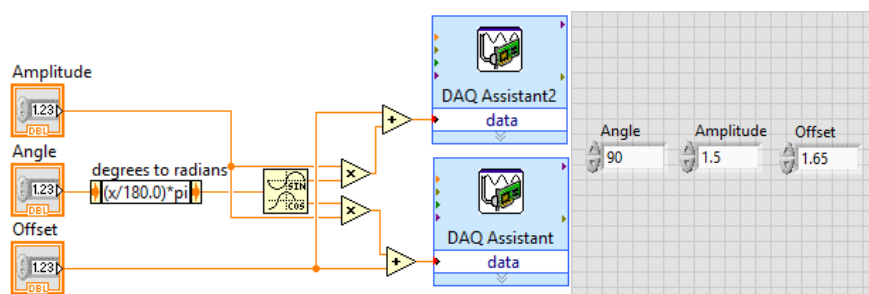


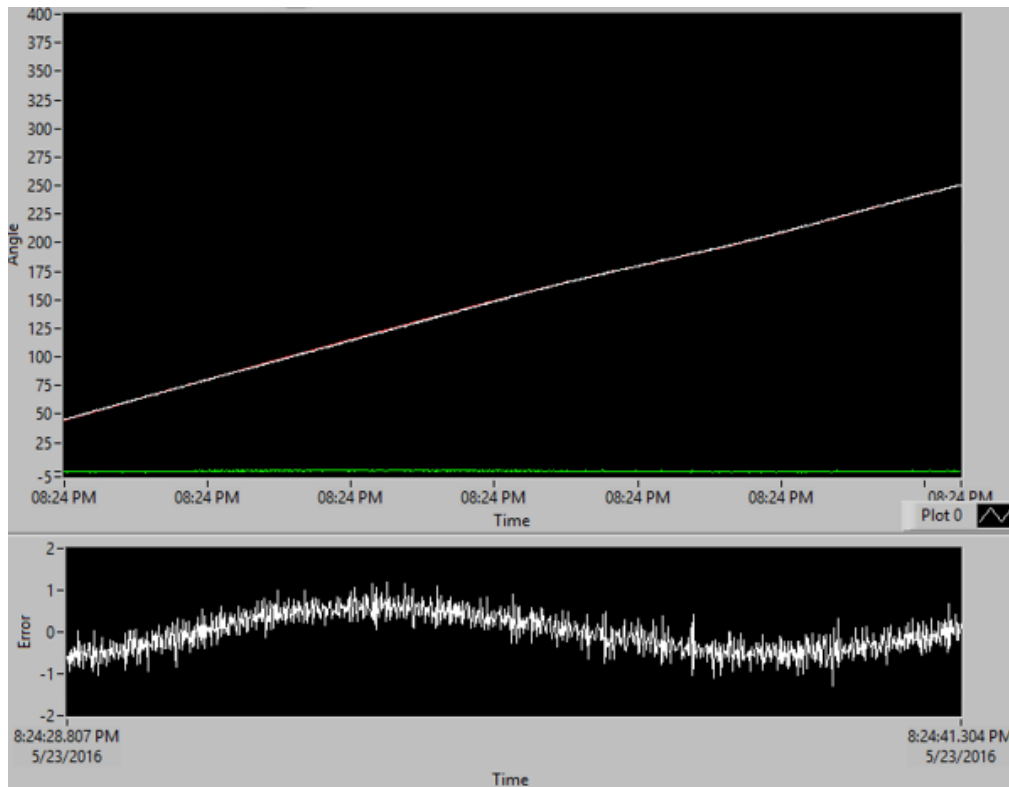
Figure 4-19: The block diagram (left) and the front panel (right) of the step change emulation.



Figure 4-20: the  $U_s$  (yellow),  $U_c$  (green) signals and the estimated angle (blue) for a step change from 50 to 80 degrees.

### 4.3.3 Test Bench Results

Figure 4-21 shows the error signal coming from the test bench setup described in chapter 6, which contains a hall-effect sinusoidal encoder, for the proposed converter with sixteen sections using the small angle approximation option for about one revolution with constant speed. For reasons that will become evident in chapter 6 the precision of the system did not allow testing the conversion error of the proposed converter which has a maximum conversion error much smaller than the precision of the test bench. However, Figure 4-21 shows clearly that the converter flows the angle smoothly.



**Figure 4-21: Error signal from the test bench for the proposed converter with 16 sections using small angle approximation.**



# Chapter 5 CLOSED-LOOP CONVERTER

A recent paper was published in IEEE about PLL [22] was chosen to be implemented and tested, in order to compare the performance and accuracy between the open loop converter and the closed loop converter. In this section, the theory of the paper will be explained in details, simulated using SIMULINK, then implemented practically.

## 5.1 Theory

The purpose of this paper is to improve the dynamic response of a PLL converter, to determine the estimated angle of the encoders and resolvers. This paper uses analog triangle-sine conversion technique to approximate the sine and the cosine using the estimated output angle signal. The advantages of this method is that it introduces an analogue implementation of a PLL which does not need an AC reference signal. Figure 5-1 shows the schematics of the proposed method.

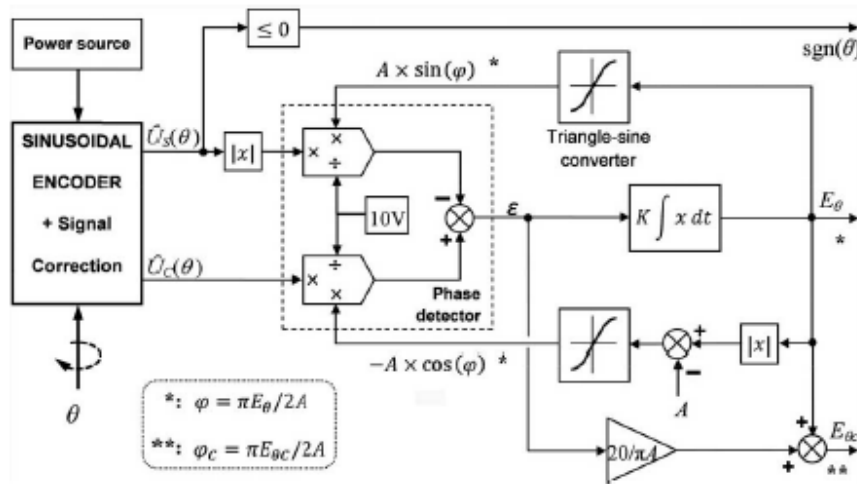


Figure 5-1: The schematics of the proposed method of the PLL converter [22]

As seen from figure 5-1, the outputs of the sinusoidal encoder is the input of the converter, which is the conditioned sine ( $U_s(\theta)$ ) and cosine ( $U_c(\theta)$ ). The amplitude of the input signal is represented as "A" and the output is represented as  $E_\theta$  (which is the estimated angle).  $|U_s(\theta)|$  and  $U_c(\theta)$  enters a phase detector (PD) where they are mixed with  $A * \sin(\varphi)$  and  $-A * \cos(\varphi)$  (where  $\varphi = \frac{\pi E_\theta}{2A}$ , 2A is the peak to peak). The output of the PD is the loop error signal  $\varepsilon(\theta)$ .

$$\varepsilon(\theta) = -\frac{A^2}{10} \left( (\cos(\theta) * \cos(\varphi)) + (|\sin(\theta)| * \sin(\varphi)) \right) = \frac{A^2}{10} \sin \left( \left| \theta \right| - \frac{\pi}{2} - \left( \frac{\pi}{2A} E_\theta \right) \right) = \frac{A^2}{10} \sin(\Delta\theta) \quad (5-1)$$

The loop error signal then enters an integrator where the estimated angle is extracted.

$$E_\theta = K * \int \varepsilon(\theta) dt \quad (5-2)$$

Where K is the loop gain of the converter. At zero speed, the error signal converges to zero. Thus:

$$E_\theta \approx -A + \left( \frac{2A}{\pi} \right) |\theta| \quad (5-3)$$

Where  $\theta$  is in the range of  $[-\pi, \pi]$ . Hence  $E_\theta$  tracks the input angle  $\theta$ . During the steady state, the PLL forces the lower multiplier to match the upper multiplier to maintain the result of  $\varepsilon(\theta) \approx 0$ . Thus, from figure 5-1, the lower triangular-Sine converter tends to determine the peak of  $E_\theta$ . Hence, to obtain the maximum amplitude of the converter,  $|E_\theta|$  is shifter by  $A$ . For example, when  $\theta = -\frac{\pi}{2}, 0, \frac{\pi}{2}$ ,  $U_s(\theta) = 0$ . Since the PLL forces the lower multiplier to be zero,  $|E_\theta| - A$  is equal to zero.

It can be observed that the closed loop behaves as a first order system, because of the integrator. Thus, the time constant of the integrator must equal to  $\frac{20}{\pi AK}$  in order to maintain the condition of having a small error. The maximum rate of change of  $E_\theta$  depends on of the first order system. Thus, the time constant must be chosen with respect to the maximum rate of change of  $\theta$  to ensure that  $E_\theta$  tracks the changes of  $\theta$ . In case of a constant ramp input, the time constant must be much smaller than half the period of the full rotation of the sensor shaft. This is because of the absolute value in equation 5-1 and 5-3, resulting in a triangular output.

The rotational speed is represented by  $\frac{d\theta}{dt}$ , which is a direct measurement of  $\varepsilon(\theta)$ . When  $\theta$  is constant (zero speed condition  $\frac{d\theta}{dt} = 0$ ), the error signal  $\varepsilon(\theta) \approx 0$  and  $\varphi \approx |\theta|$ . Thus when  $\frac{d\theta}{dt} \neq 0$ , a dynamic error will occur in calculating  $\theta$ . Hence, by generating an auxiliary compensated output, the dynamic error can be minimised. The compensated auxiliary output ( $E_{\theta c}$ ) is a combination of  $E_\theta$  and  $\varepsilon(\theta)$ . Thus:

$$E_{\theta c} = E_\theta + \frac{20}{\pi A} \varepsilon(\theta) = E_\theta + \frac{2A}{\pi} \text{Sin}\left(|\theta| - \frac{\pi}{2} - \frac{\pi}{2A} E_\theta\right) \quad (5-4)$$

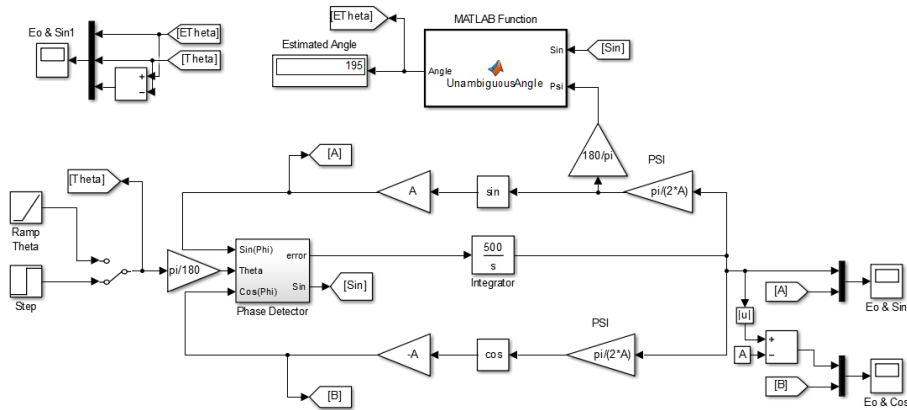
In the condition of small error,  $\text{Sin}(x) = x$ . Thus, from equation 5-4.

$$E_{\theta c} = E_\theta + \frac{2A}{\pi} \text{Sin}\left(|\theta| - \frac{\pi}{2} - \frac{\pi}{2A} E_\theta\right) = E_\theta + \frac{2A}{\pi} \left(|\theta| - \frac{\pi}{2} - \frac{\pi}{2A} E_\theta\right) = \frac{2A}{\pi} \left(|\theta| - \frac{\pi}{2}\right) \quad (5-5)$$

Note, that the equation 5-3 is only valid when there is no angular displacement (no motion). However, equation 5-5 is valid for both no motion and in motion. Thus, the dynamic response is improved by implementing the auxiliary output ( $E_{\theta c}$ ). For example, if a step change occurs in the input angle, where the input angle moves from  $\theta_1$  to  $\theta_2$ . at  $\theta_1$ , the error  $\varepsilon_1 = 0$  (steady-state), and the converters output would be  $\varphi_1 = \varphi_{1c} = -\frac{\pi}{2} + |\theta_1|$ . at  $\theta_2$ , the error signal would change to  $\varepsilon_2 = \left(\frac{A^2}{10}\right) * \text{Sin}\left(-\frac{\pi}{2} + |\theta_2| - \varphi_1\right) = \frac{A^2}{10} * \sin(|\theta_2| - |\theta_1|)$ . Thus, the integrator prevents  $E_\theta$  from the immediate tracking of the input angle  $\theta$ . However, using the auxiliary output  $E_{\theta c}$ , an immediate result would be obtained as  $\varphi_2 = \varphi_1 + \text{Sin}(|\theta_2| - |\theta_1|)$ . Therefore, the method of using the compensated auxiliary output can be used for both, digital and analogue PLL.

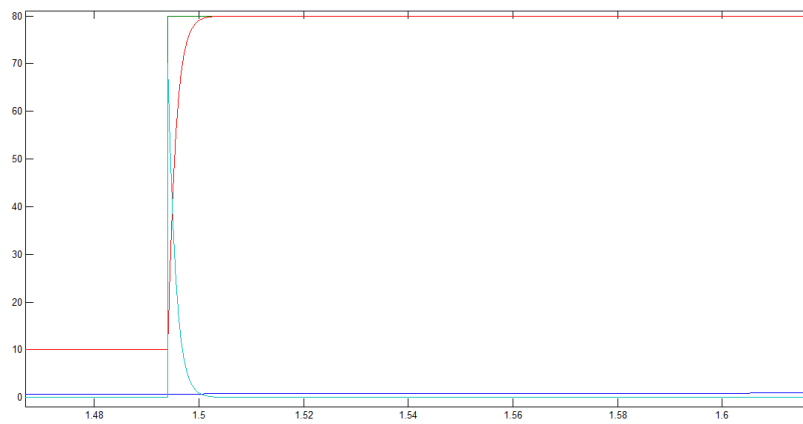
## 5.2 Simulation

A Simulink module is made to analyse and test the PLL without the auxiliary compensation output. Figure 5-2 shows the Simulink module.

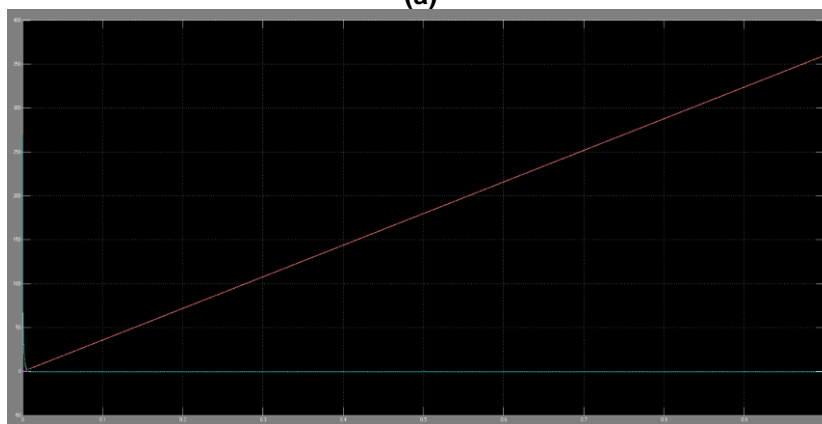


**Figure 5-2: Simulink module representing the PLL.**

Figure 5-2 follows the same logic as explained in the theory without the auxiliary output. The "Unambiguous Angle" in the Matlab function checks the sign of the conditioned  $U_s(\theta)$ , if the value of  $U_s(\theta)$  is bigger than 0, 90 degrees will be added to the output (angle). Else, the  $Angle = 360 - (PSI + 90)$ . Figure 5-3 shows a step input and a ramp input. for a step input, the error is zero, and the response time is around 0.01 seconds. For a ramp input, there is always a relative error of around 0.127%.



**(a)**



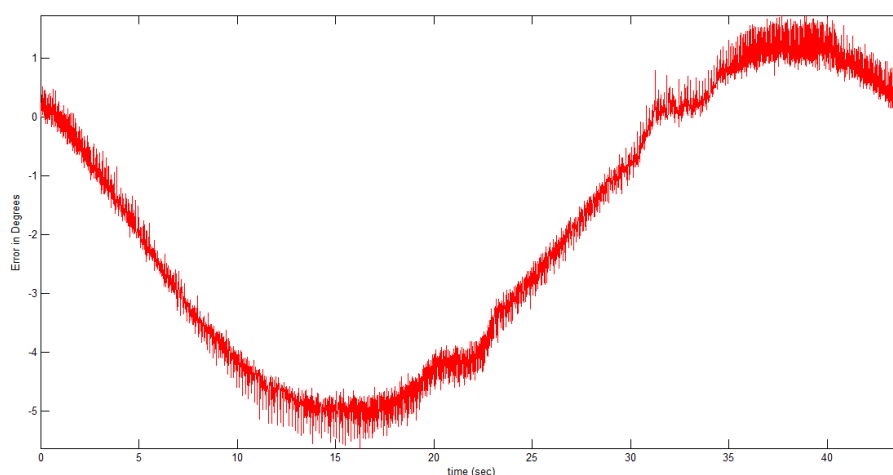
**(b)**

**Figure 5-3: The output of the Simulink module in figure 5-2. (a) shows a step input of 195° at 0.5 seconds. (b) shows a ramp input. Yellow is the estimated angle, Pink is the input angle and Blue is the error.**

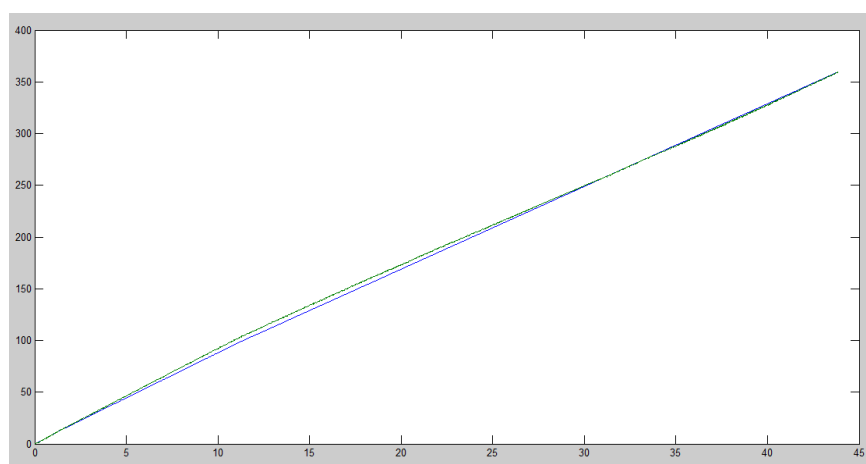
### 5.3 Implementation and Practical Testing

The same procedure that was done on the open loop converter was implemented on the closed loop converter. As a consequence, the closed loop converter was tested practically using the emulator sensor and the test bench.

As seen from the simulation section, the expected error for the ramp angle is expected to be extremely small, and it converges to zero as the integrator gain gets bigger. However, for the practical implementation, using the emulator, the error is expected to be noisy as the ADC adds an error to the signal. The converter is implemented on mbed microcontroller where it follows the same logic as explained in the theory. However, if the integrator gain exceeds 1 only, the system becomes unstable! Thus, the gain for the practical work is extremely small compare to that of the simulation. This is because there are a lot of algorithms that the mbed microcontroller is executing sequentially, which introduces a small delay from now and then. Therefore, the execution of the mbed code cannot keep up with the generation of the angle theta. Figure 5-4 shows the estimated angle with respect to the real angle, and the error signal of the estimated angle.



(a)

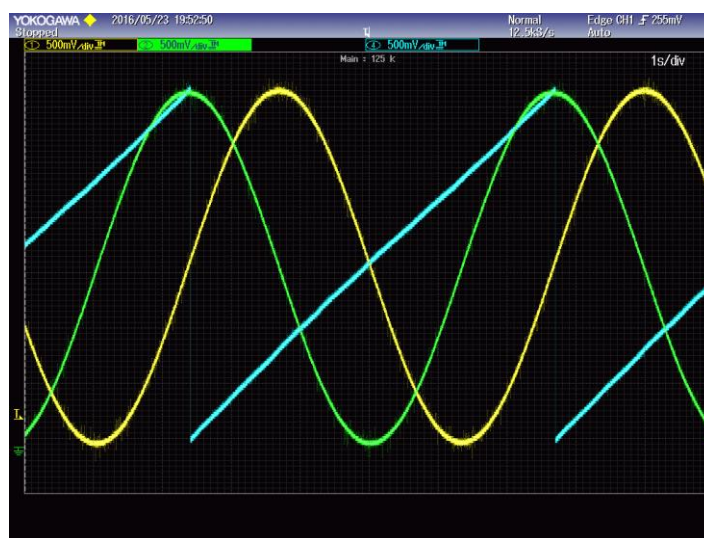


(b)

**Figure 5-4: The error signal of the estimated angle (a), and the estimated angle with respect to the real angle.**

Notice from figure 5-4 (b) that the estimated angle is not exactly following the real angle. In fact, it looks like the estimated angle is oscillating around the real angle! Thus from figure 5-4 (a), the error seems to be oscillating while it is fluctuating. The oscillation is due to the estimated angle going around the real angle (integrator gain is 1), while the fluctuating is mainly due to the ADC error.

The previous plotting of the estimated angle were made using a computer software. Figure 5-5 shows the plot of the estimated angle and the conditioned sine and cosine.

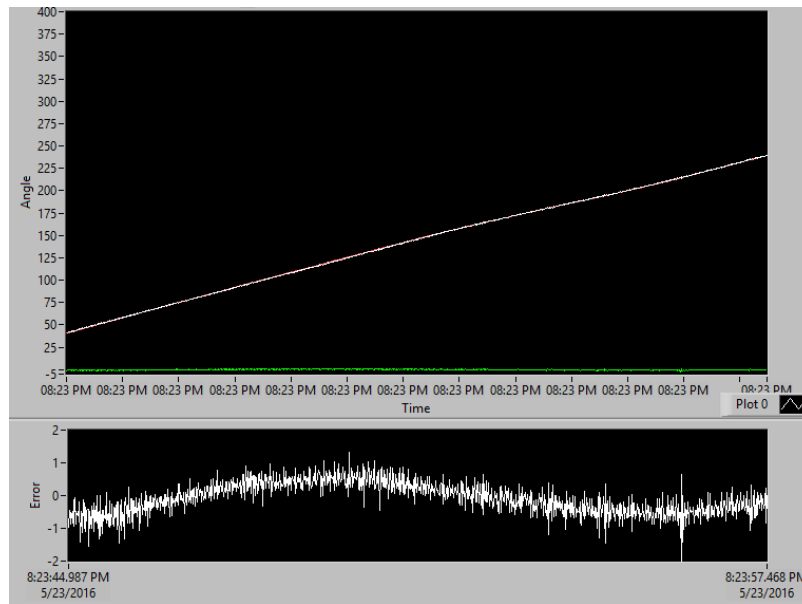


**Figure 5-5: The estimated angle and the conditioned sine and cosine.**

Observing figure 5-5, it seems as the estimated angle is a perfect ramp just like the real angle, and the conditioned sine and cosine seems well plotted. However, by zooming in the figure, the estimated angle will look similar to figure 5-4 (fluctuating and oscillating).

### 5.3.2 Test Bench Results

Theoretically, the error for both, using the emulator and the test bench should approach zero. However, as mentioned before, due to the ADC error, mbed sequential execution delay and the lack of the integral gain, the practical results does not match the expected results. Figure 5-6 shows the practical plot using the test bench.



**Figure 5-6: The estimated angle with respect to the real angle and the error of the estimated angle plot.**

As seen from figure 5-4 and 5-6, the test bench and the emulator results are similar. Conversely, the error in the test bench seems smaller than the emulator. This is because the emulator DAC is not used anymore. Thus, less error is contributing to the estimated angle.

# Chapter 6 TEST BENCH IMPLEMENTATION

## 6.1 Introduction

Within the previous chapters, lots of intensive work was done investigating and designing signal correction, open loop and closed loop techniques. Yet all of them were tested through simulations, which usually don't include much of the practical world constraints such as noise, vibrations, etc... Testing these techniques practically would eventually give a more genuine measure of their accuracy, dynamic response, reliability and robustness if the test was done correctly. Towards that, three methods will be used for practical testing, namely, Sensor Emulator, Rotary Table with and real sensor equipped on a designed test bench.

Each and every test will test the converters under certain conditions, for example, the Emulator is going to produce a controlled SCS output to insure the quality of the SCS signal's amplitude and frequency, also, to generate a perfect error signal since the real angle is generated internally. The emulator setup was discussed in details in section 4.3.2. With regards to the Rotary table, it is needed to test the steady state error of the converters with a real sensor. The error graph is generated by knowing the real angle manually from the table, this section will be briefly discussed within chapter. Last but not least, the Test Bench is used to mimic the environment of a real motor system and to test the reliability of the converters under different rotational speeds. The converter's output will be then compared to another high precision sensor fixed on the same platform to generate the error graph.

## 6.2 Rotary Table

The rotary table is a simple platform that consists of two main components, first, the hall-effect sensor. Second, the rotational disk. The rotational disk is composed of a knob, that when rotated will result in a precise rotational displacement of the disk of 0.1 *degree* resolution. The disk has a shaft fixed on its axis of rotation. The setup is implemented by simply fixing the Hall Effect sensor on the shaft of the rotational disk as shown in figure 6-1. And thus, when the disk is rotated, the shaft of the sensor will rotate as well and generate the SCS signals accordingly.



**Figure 6-1: Rotary Table setup**

To test the converters, the knob was rotated/incremented with a step of one degree. Moreover, to semi-automate the process of data collection, a switch was implemented, so that whenever the knob is turned to the needed position, the switch is turned on, and the microcontroller would output the estimated angle accordingly. Along with that, a counter was initiated to stamp every reading with its intended real angle. Thus, by the end of the 360 rotation, the microcontroller would eventually output an array of 360 estimated angles along with their real angle. Further to that, the error can be calculated by subtracting both values. The error is expected to be higher than usual since the angle is set manually, and thus lots of errors will be involved.

### 6.3 Test Bench Introduction

The test bench will consist of two main parts, Hardware and software. Intuitively speaking, each part will introduce some error to the overall system, and here lies the challenge to minimize all of these errors as possible while taking every step towards implementing the test bench. Throughout this chapter, the designing steps done towards making the Test Bench found in figure 6-2 will be discussed in details in both hardware and software.

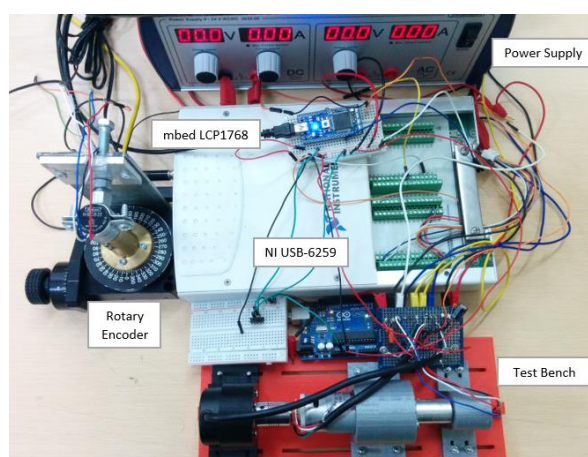


Figure 6-2: Final Assembly of Test Bench

### 6.4 Equipment

To conduct both accuracy and dynamic test, the test bench should include a rotating object with a shaft pointing out of it. That shaft will be connected to a test sensor which will output the SCS signals. These signals are then fed to the microcontroller to get corrected / converted. From the other side, another high precision sensor should be attached to measure the angle of the shaft in parallel with the converter. After that, both of the converted angle and the high precision angle will be transferred to a software program to get the error between them. To do that, these items are needed:

- 1) DC-Geared motor: Is the rotating object, and it has two shafts pointing out of it. It has a gear with 100:1 gear ratio. Having such gear will increase the accuracy of the measurement, as each degree read by the converter will be magnified into 100 degrees on the other side.

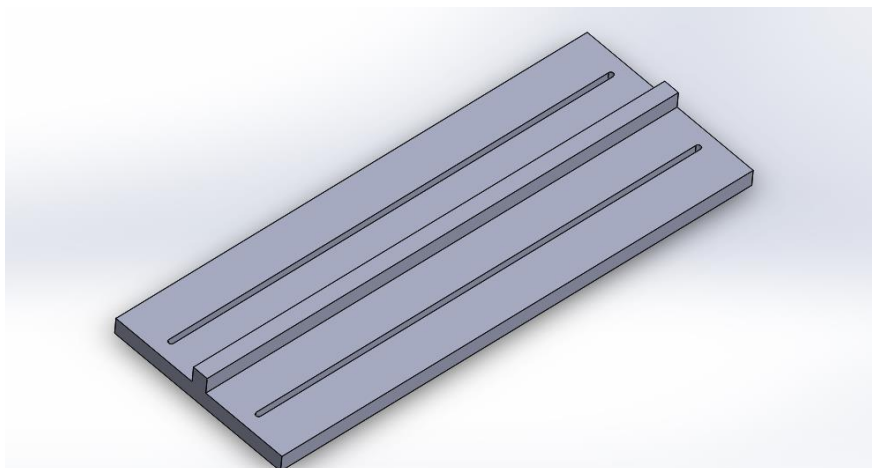


- 2) Hall effect sensor: Is the test sensor that will provide the microcontroller with the SCS angles, it was chosen over resolvers to eliminate the need for a demodulator which in fact will add errors and will slow the system dynamics.
- 3) Pulse Encoder: Is the high precision sensor. The chosen encoder has two outputs A & B. each produce 500 pulse per revolution, and they are 90° shifted of each other. Hence, if A and B outputs are XORed together, the output will be 1000 pulse per revolution. Also, from the gear transformation, the encoder will experience 100 revolution versus 1 revolution on the converter side, therefore, the pulse encoder will produce 100,000 pulse per converter revolution which leads to a resolution equivalent to  $\frac{360}{100,000} = 0.0036 \text{ degree/pulse}$ .
- 4) Couplers and L-Shaped brackets: In order to connect between shafts, a coupler is needed, yet, to reduce the vibrations generated when any of the shafts is displaced, a flexible shaft coupler is used instead. Also, to support the motor and sensors and to reduce the torsion forces acting on the system, an L-shaped bracket is needed.
- 5) Base: Is the foundation that holds and align all of the previously mentioned parts together.

## 6.5 Hardware Implementation

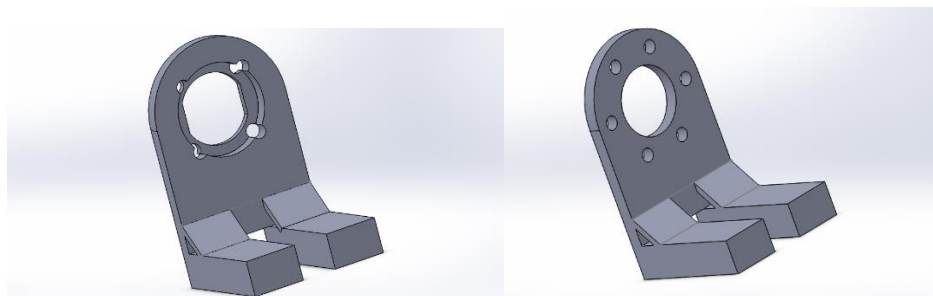
To implement the previously mentioned parts together, the team thought of two solutions. Either building it using metallic parts, which usually come in standard dimensions, or to design a customized Test bench and print it via 3D-Printers. The first solution was undergone at first, until the team found out that the last part; the Pulse Encoder, does have some non-standard dimensions, also, the available parts in shops will not satisfy the alignment constrains demanded, which drove the team to go for the second solution and start with designing customized equipment. To do so, the team learned the basics of SolidWorks which is the program that will be used design models for 3D printing later.

The most important aspect that needs to be respected while designing the Test Bench is the X, Y and Z axes alignment of all components. As mentioned before, the key to eliminate the misalignment issues is the Base. The base was designed in a way to eliminate the X and Y misalignment issue by adding a slider in the middle with adequate width, and add a groove at the middle of each component, and thus, once the component is installed on the designed base, it will be deprived from the X and Y degrees of freedom. Also, to solve the Z axis degree of freedom, some ideas was generated, like adding screws to the sides of each part and then fix it to the Base slider. Yet, this design will limit team to fixing all parts at one place only, which is not good for debugging and re-editing at assembly phase. Hence, another design was introduced, which adds the screws from top instead, and to have two long sweeping holes on both sides of the base, and thus the parts position can be altered by releasing the screws, the base designed is shown in figure 6-3.



**Figure 6-3: Test bench base**

The first version of the test bench brackets were designed to mimic most of the online found brackets. But since the materials used in 3D printing are more fragile than the metallic parts, supports were added on both legs to prevent the holder from collapsing under high rotational speeds. Accordingly, the following designs shown in figure6-4 were created.



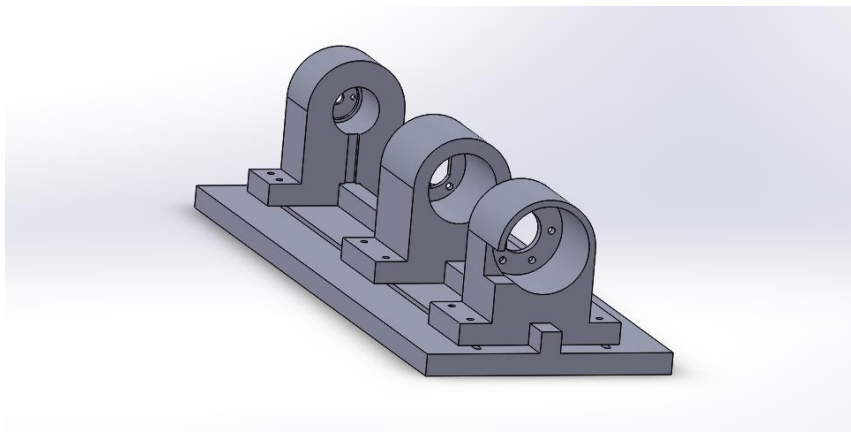
**Figure 6-4: Version 1 SolidWorks brackets design**

After that, it was thought to alter the last design and to add more support to the holded parts as to damp the vibrations that eventually result from having loose contact between the holded object and the bracket. Hence, version one was edited to have an extended holding pipe, that will sufficiently support the motor and the other parts as well as damp their vibrations. Version two designs are as shown in figure6-5.



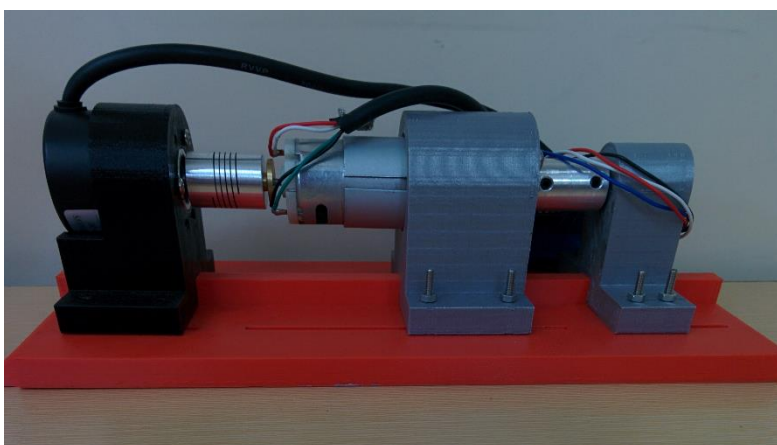
**Figure 6-5: Version 2 SolidWorks brackets design**

After designing the base and the brackets, all parts were assembled on SolidWorks to sort out assembly errors as seen in figure 6-6.



**Figure 6-6: Test Bench Version 2 SolidWorks assembly**

After that, one piece was printed for testing purposes. It was found that the printed part was much tight to have the object going in smoothly, and thus, all parts were altered accordingly to have a tolerance of  $0.2\text{mm}$  to allow the objects to get inside easily, yet, not loose. Next, all parts were printed and assembled as shown in the following figures 6-7.



**Figure 6-7: Test Bench Hardware Assembly**

During the assembly phase, the shafts were pushed against each other to check the alignment and it passed, then the test bench was tested under high speed to check if any misalignments would happen. The test bench passed the second test as well, yet, during that test, the whole base was vibrating, and thus displacing the whole test bench. To solve that problem, it was thought to add a rubber mat underneath the base, which will absorb the vibrations and eventually would prevent the test bench from displacing.

## 6.6 Software Implementation

Software implementation is as important as the hardware implementation, and thus needs to be designed carefully to minimize its error contribution. The software implementation would consist of three main parts, reading the pulses generated from the pulse encoder and processing them further, reading

the converted angle from the microcontroller, assembling both together and solve any errors found. LabVIEW was chosen to be the platform that will carry out these tasks. That's because it is a well-known software package by National Instruments. Also, It provides the user with different reliable tools to read from ADCs and Serial communication as well as outputting the data in numerical, graphical, and output voltage through DACs formats utilizing NI-Elvis boards. After choosing LabVIEW as the testing and development platform for the software test bench, the team started going through the process of building the testing program. This resulted in many iterations and versions of the test bench, all of which will be went through later in the chapter.

Using LabVIEW, the team started creating a graphical user interface (GUI) that included different settings and had visual outputs to indicate different results like the output angle, voltage, etc....The first device used to gather data in order to be used inside LabVIEW was National Instrument my-DAQ.

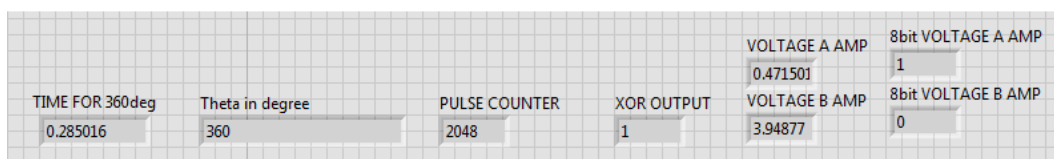


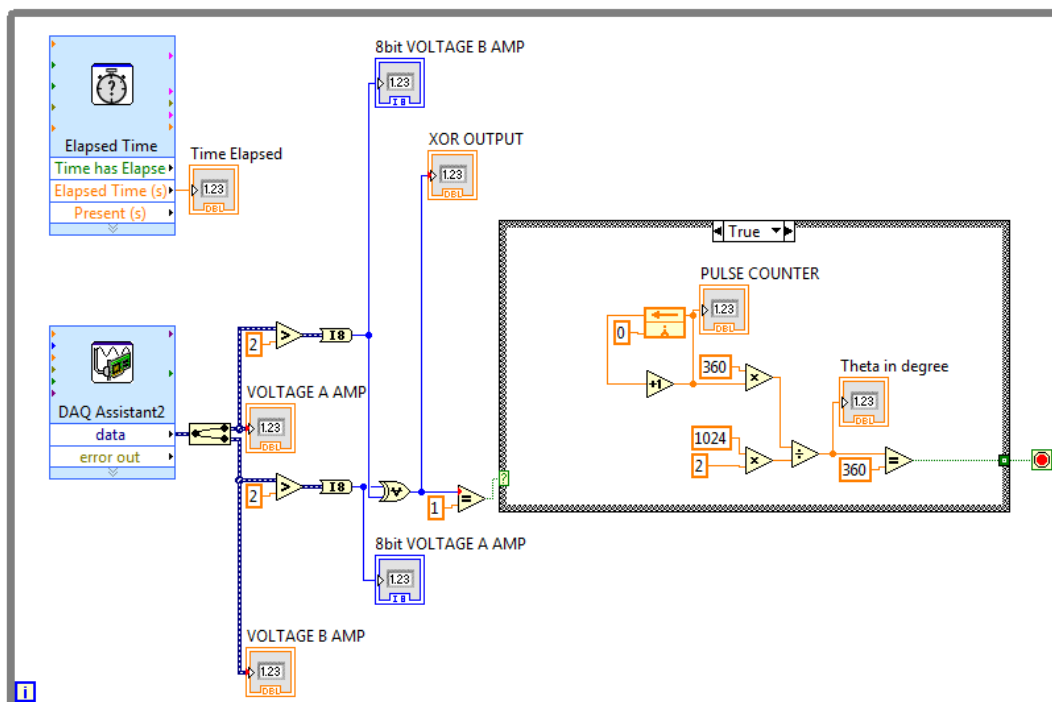
**Figure 6-8: NI my-DAQ**

This data acquisition device (DAQ) allowed the user to use the different analog/digital inputs and outputs to perform several tasks. The first version of the LabVIEW test bench utilized both A and B outputs of the pulse encoder, using two analog input channels to receive the data, then, the program would convert the analog input into digital high/low output which is done by setting a certain threshold depending on the incoming amplitude of the input analog signals.

**6.6.1 Digitized XOR**

In this early version of the LabVIEW test bench, the program would perform digital XOR on the digitized input. The number of HIGH logic output for the XOR will then be counted, and stored. This allowed the user to know the number of pulses, then using a certain algorithm the program would output the angle corresponding to the number of pulses registered by the program. In addition to that, the program would keep track of the running time, allowing the user to find the rounds per minute RPM if needed. An overview of the GUI of the first version of the LabVIEW test bench can be seen in the following figure.





**Figure 6-9: front and back panels of the LabVIEW program**

The main issue with this program was the fact that digitizing the XOR would introduce more errors in the system, and would increase the delay. Which would not suit this type of application for high speed angle detection that the project is aiming for.

### 6.6.2 Analog XOR

Building on the knowledge gained from previous iterations of the test bench, the next step was to use the analog output of the pulse encoder directly, without digitizing the output. This would reduce the time delay, which is a very important criterion for the design of the test bench. This eventually led the team into using the embedded counter in specific DAQ devices.

The counter would allow using the XOR output of the pulse encoder directly, and using LabVIEW, the constant count of pulses would be provided for the user to use. This allowed the team to build this version of the LabVIEW test bench. Where in this version, the number of pulse counts is taken from the DAQ assist and just plugged into different blocks that result in the output angle in degrees. The program can be seen in the following figure.

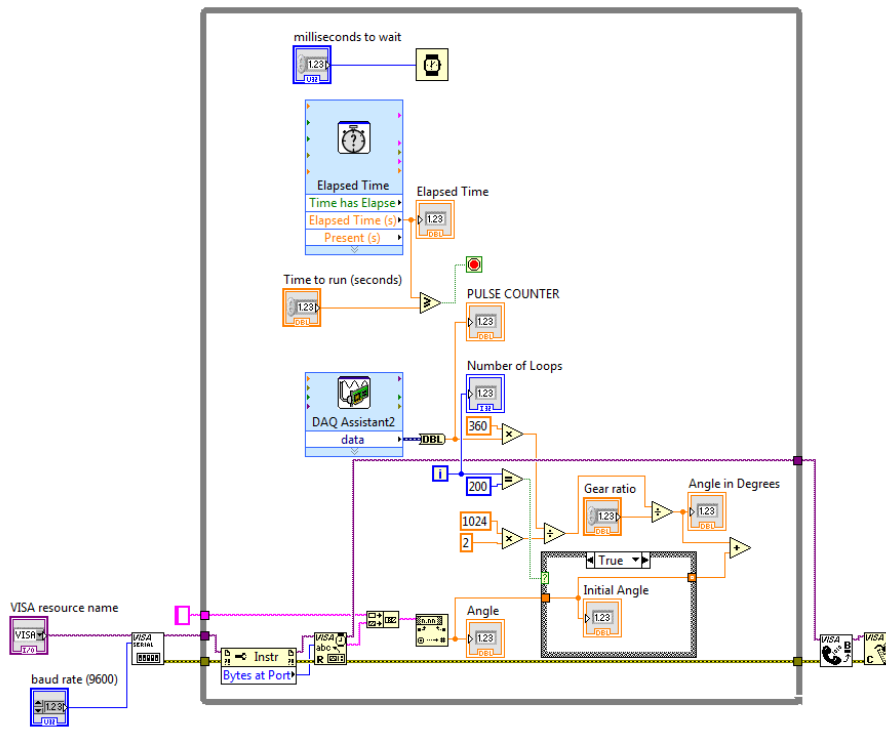
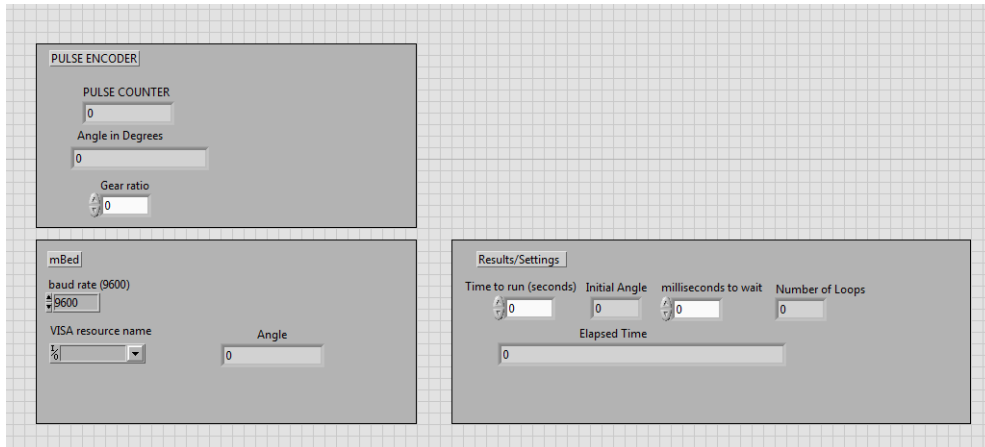


Figure 6-10: second version front and back LabVIEW panels

### 6.6.3 Sampling Frequency Problems

Throughout the building process of the LabVIEW test bench, one of the frequent issues faced by the team is the hardware limitation of the DAQ device. This limitation was in the form of sampling frequency, as the capability of some DAQs was not enough to keep up with the fast changing input.

The first DAQ used in the development of the LabVIEW test bench was the NI my-DAQ seen earlier in Figure X, the main issue with using this DAQ is the fact the maximum sampling frequency for the two analog input channels is 200kSamples/Second. This resulted in many errors when the XOR output from the pulse encoder was at its highest speed, resulting in an XOR output frequency of 150 kHz, which cannot be sampled by this DAQ according to Nyquist criteria.

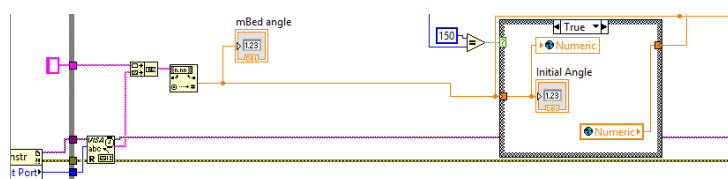
To fix this, a different DAQ was used, this time it was the NI 6259 DAQ, which has a multi-channel sampling frequency of 1MSample/Second. This resolved the problem of the sampling frequency.

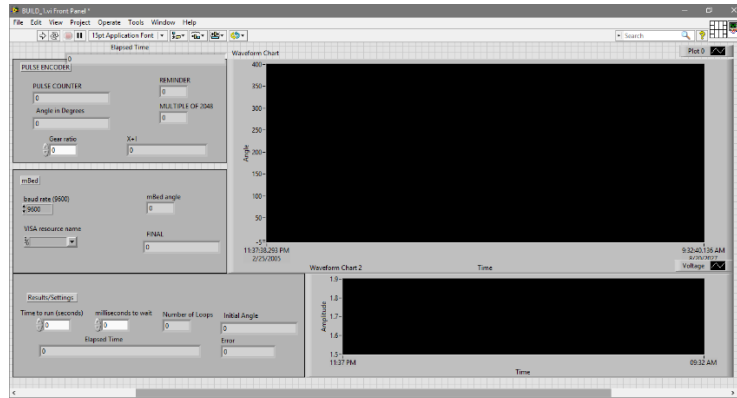
#### 6.6.4 Assembly problems and solutions

After being successful with reading the pulse encoder signals and choosing the right DAQ to work with, the next step was acquiring the converted angles from the converter; mbed in this case, to compare them together and get the error. To do that, serial communication was needed to transfer the data to LabVIEW. It can be observed from the previous figures that Serial communication block “VISA” was included in that version of the LabVIEW test bench. This included the ability to select the serial COM port and the baud rate for any microcontroller programmed to find and serially print the angle.

Throughout the software implementation, two major sources of errors were found and tackled. First, synchronizing the Pulse Encoder start angle with the converter’s start angle, it can be observed here that if any errors happen in this stage, it will be added as a constant error in all readings later. Second, the difference of the rate of change of the angle between both platforms, it can be seen that if the rate of change of the angle is different between the two, the error that will be added will not be constant, yet, it will be increasing and getting accumulated over time, and thus, the Test Bench will not reflect the real error of the converter which would defeat the purpose of the Test bench.

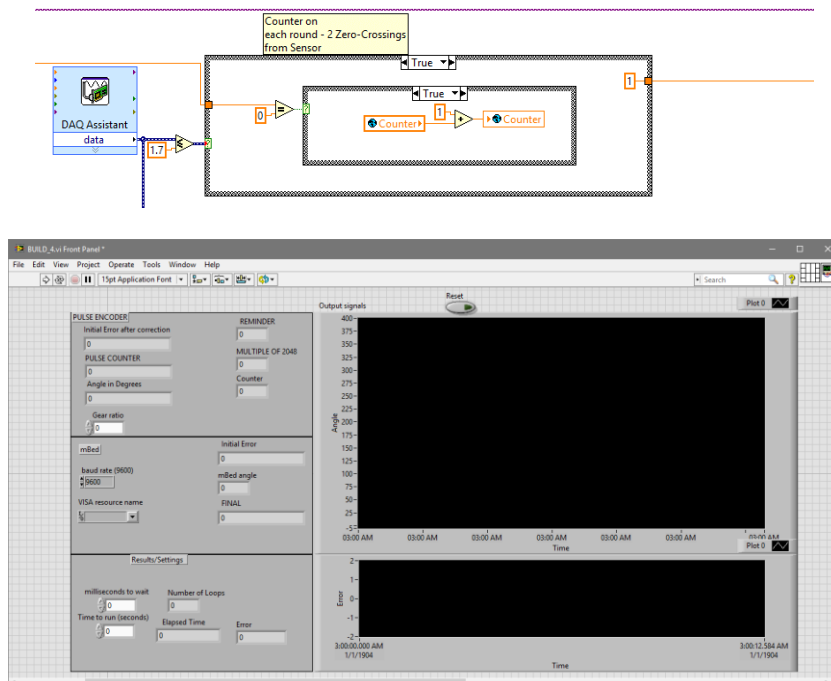
To tackle the first problem, two methods were implemented. First, setting the speed of the motor to Zero, so that the angle will not change, then read the angle given by the converter and assign it to the Pulse encoder as an initial angle, and then excite the motor to start rotating and the LabVIEW would then start reading the pulses from the pulse encoder and count/convert the pulses to angles as shown previously. The VI of this method is shown in figure 6-11 partially, as it shows that the VI will wait 1.5 sec then reads the initial value from the converter and saves it to a global variable that will be used later while processing the Pulse Encoder angle. This method however was not reliable, since the converter itself has an error which lead every time to a smaller initial error, which is not eliminated. Thus, it can’t be used to assess the converter error.





**Figure 6-11: Initial angle error first solution front and back LabVIEW panels**

The second solution was about utilizing the Hall-Effect sensor’s sine and input it to the LabVIEW through ADC. Then, generate a bit that is logic HIGH when the sine is positive and logic LOW when the sine is negative. Then, a condition is made to trigger the DAQ to start reading the Pulse encoder’s pulses. The condition is true if the bit was changed from logic LOW to logic HIGH, which is the instance were the angle should be Zero within the converter too. With this method, the starting error is independent on the converter’s error. Also, it can be used while the motor is already rotating. But a problem would evolve if the power supply of the sensor is unstable. Since the program will highly depend on the instance of the zero crossings of the sine generated from the sensor, which is highly dependent on the power supply. Yet, the error formed from this method is much better than the previous method. The VI of the second method is partially shown in figure 6-12.

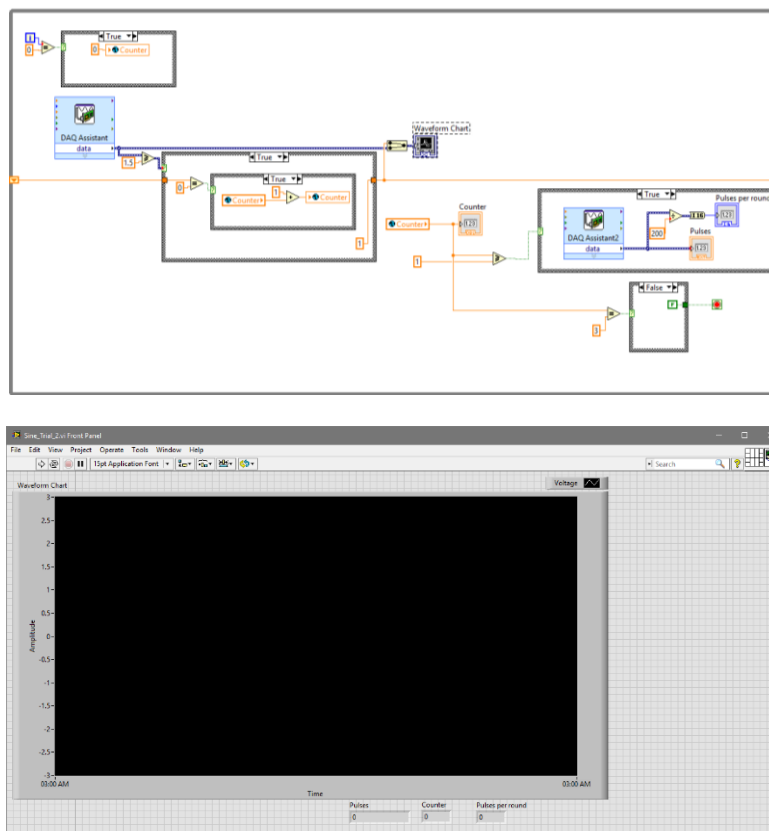


**Figure 6-12: Initial angle error Second solution front and back LabVIEW panels**

For the second type of error, the deference of the rate of change of angle between both platforms, it was hard to know the reason behind the error. At first, there was a doubt about a problem within the Development of Instrumentation for High-Precision Angular Position and Speed Measurement



Pulse Encoder itself with regards to the number of pulses it generates per revolution. Which lead the team to create a new VI that would read how many pulses per revolution does the Pulse encoder generate by utilizing the same technique of sine zero crossing utilized in the previous problem. Then, the program is run until it reads specific number of revolutions and outputs the number of pulses per revolution. The VI is shown in figure 6-13.



**Figure 6-13: Pulse Encoder pulse per revolution front and back LabVIEW panels**

The results of this VI showed that the pulse encoder's pulses per revolution is indeed changing. Yet, after debugging, it was observed that the error was caused from two external sources. First, the screws within the Test Bench were not tight enough, which lead to vibrations and a slip between the shafts and the couplers. Second, the analog XOR gate was examined again using an oscilloscope, where it showed that the XOR skips some of the pulses. After debugging, it was observed that the source of the problem was the power supply. Thus, a separate power supply was used to power the Pulse encoder and the XOR gate which solved the problem. When the test was done again to check the number of pulses generated per second, the result was the same number as indicated within the datasheet and it was repeatable.

## **6.7 Test Bench Final Assembly**

After solving most of the major problems that was faced in hardware and software implementation, the system was tested with different converters, to check their practical error. The error shown in most of the trials was around one-degree peak which is a bit higher than expected for all converters. Yet, this error was expected since the test bench is not yet fully developed. These errors can be attributed to many minor problems acting on the test bench, such as minor vibrations, and the initial angle problem and delays in serial communication as well as the fluctuations of the power supplies which would in fact change the SCS signals of the Hall-Effect sensor. Although the current test bench is not as perfect as it should be in a professional work, yet working on it gave the team lots of practical experience and debugging skills.

## Chapter 7 CONCLUSION

Through this report, the topic of the project was introduced and its importance was clarified. The significance of the rotary position and speed measurements as an important element of many modern advanced control systems was shown. There are several types of rotational sensors, however, the project concentrated on sinusoidal encoders.

Sinusoidal encoders output two signals, one is proportional to the sine and the other is proportional to the cosine of the measured angle. These two signals often contain imbalances. The most significant imbalances are amplitude imbalance, where the two signals do not have the same amplitude, and phase imbalance, where the two signals are not exactly 90 degrees out of phase. After ensuring that these signals are balanced by performing the necessary detection and correction, they are inputted into a suitable converter to estimate the input angle. The aim of the project is to design a new open-loop imbalance correction technique as well as an open-loop converter, to implement them together with a closed-loop converter from the literature and compare results.

A detailed literature review of the most important and recent works related to sinusoidal encoders' imbalances correction was presented. In addition to that, a review of the literature related to converters was given. It was shown that there are two main categories of converters: open loop and closed loop. Selected Methods related to both categories were briefly mentioned. The design constraints of the project were stated after that according to the state-of-the-art imbalance correction techniques and converters as well as the technical standards.

After that, novel methods were introduced for amplitude and phase imbalance correction as well as resolver-to-digital conversion in consecutive chapters to achieve these constraints and objectives. In each chapter the theoretical calculations were presented followed by several simulation tests in addition to emulation and practical implementation.

The new imbalance correction method depended on only three samples points to effectively detect and correct for the amplitude and phase imperfections in the sensors' signals. The method was successfully implemented on a low-cost microcontroller and the tests performed on the method showed how significant is the effect of correction in decreasing conversion error.

This presented conversion method, on the other hand, consisted of two main stages. In the first stage, the location of the input angle is located in one of  $N$  equal segments in the  $360^\circ$  interval utilizing a dedicated logic circuit. In the second stage, the deviation angle or the fine angle was determined by one of two options: relying on small angle approximation of the tangent function or using a very small LUT as small as a four-element LUT. The method was tested and compared to other very recent methods using extensive simulation and emulation followed by practical implementation and proved to satisfy all the objectives and constraints of the project.

In addition to that, a recent closed-loop conversion method was explained and successfully simulated and implemented. It was found that the performance of the presented open-loop method is

superior. Following that was a chapter describing in details the great effort spent on designing and assembling the test bench used to test the correction technique and different types of converter. It was found that the precision of the system with the current components does not allow precise testing of the conversion methods. However, the system was proved to work correctly and can be developed further.

The project has successfully achieved its objectives and goals, but more significantly, the amount of the acquired knowledge and experience was priceless. In addition to that, valuable skills were honed, like efficient time management and effective teamwork. Believing in the critical importance of such project in acquiring interdisciplinary knowledge and applying it to real life applications, we would like to thank everyone helped us during this long journey.

## **7.1 Achievements**

The team was able to add valuable scientific contributions to this active field of research and was able to compete, in terms of the results, with recent works published in very respectful journals. Because of that, the team was able to submit a paper describing its work on the novel open-loop converter method presented in chapter 4 to a major IEEE Instrumentation and Measurement Society Conference. The conference is named "Sensors Applications Symposium". The paper was accepted and was presented by the team in Italy in April 2016. Furthermore, the paper was awarded a merit-based travel grant that was given to the six best student papers among nearly 40 student papers in the conference.

In addition to that the team submitted a second paper to highly respected IEEE Industrial Electronics conference named "International Conference on Power Electronics and Motion Control" about the work on the new imbalance correction technique the paper was accepted with good feedback from the reviewers and should be presented next September in Bulgaria.

Currently, the team is working hard to extend these works into a journal paper as it was invited to submit an extended version of the presented paper in Italy to a journal paper in a special edition of the IEEE Transactions on Instrumentation and Measurements.

## **7.2 Further Development**

Further development can be done in three main aspects:

The correction technique: though it is novel and very computationally efficient as it uses only three sample points, it is highly dependent on the maximum amplitude of the sine signal. Advance techniques such as machine learning can be utilized to eliminate this drawback.

The emulation testing: because of the emulator and microcontroller limits in terms of precision, update rate and sample rate, the error signals of the proposed method with 16 sections and that of the other similar methods couldn't be tested. In addition to that, the step response couldn't be obtained. Using more advanced components will solve this problem.

The test bench: more work needs to be done to make the test bench able to perform more advanced and precise tests on converters. Part of this job is to use different components and part of it to improve the software interface.

# REFERENCES

- [1] G. Ye, S. Fan, H. Liu, X. Li, H Yu, Y. Shi, L. Yin, B. Lu, "Design of a precise and robust linearized converter for optical encoders using a ratiometric technique," *Meas. Sci. Technol.*, vol. 25, 125003-125011, 2014.
- [2] K. Bienczyk, "Angle measurement using a miniature hall effect position sensor," in *Proc. IEEE Electrodyn. Mechatron.*, Opole, Poland, May 19–21, 2009, pp. 21–22.
- [3] D. Wang, J. Brown, T. Hazelton, and J. Daughton, "360° angle sensor using spin valve materials with SAF structure," *IEEE Trans. Magn.*, vol. 41, no. 10, pp. 3702–3705, Oct. 2005.
- [4] C. S. Anoop and B. George, "Electronic Scheme for Computing Inverse-Cosine and its Application to a GMR Based Angle Sensor," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 7, pp. 1991-1999, July 2012.
- [5] N.H. Duc, B.D. Tu, N.T. Ngoc, V.D. Lap and D.T.H. Giang, "Metglas/PZT-Magnetolectric 2-D Geomagnetic Device for Computing Precise Angular Position," *IEEE Transactions on Magnetics*, vol.49 , no. 8, pp. 4839 – 4842, 2013.
- [6] D.C. Hanselman, "Techniques for improving resolver-to-digital conversion accuracy," *IEEE Trans. Ind. Electron.*, vol. 38, no. 6, pp. 501-504, Dec. 1991.
- [7] S.Sarma and A.Venkateswaralu, "Systematic error cancellations and fault detection of resolver angular sensors using a DSP based system," *Mechatronics*, vol. 19, no. 8, pp. 1303-1312, Dec. 2009.
- [8] S.H. Hwang, Hyun-Jin Kim, Jang-Mok Kim, Liming Liu, and Hui Li. "Compensation of amplitude imbalance and imperfect quadrature in resolver signals for PMSM drives," *IEEE Trans. Ind. Appl.*, vol. 47, no. 1, pp. 134-143, Jan.-Feb. 2011.
- [9] J.J. Moon, H.J.Heo, W.S.Imand J.M. Kim, "Classification and compensation of amplitude imbalance and imperfect quadrature in resolver signals," *16th European Conference on Power Electronics and Applications*, pp. 1-7, Aug. 2014.
- [10] J. Lara and A. Chandra, "Position error compensation in quadrature analog magnetic encoders through an iterative optimization algorithm," *40th Annual Conference of the IEEE Industrial Electronics Society IECON*, pp. 3043-3048, Oct. 2014.
- [11] A. Bunteand S.Beineke, "High-performance speed measurement by suppression of systematic resolver and encoder errors," *IEEE Trans. Ind. Electron.*, vol. 51, no. 1, pp. 49-53, Feb. 2004.
- [12] J. Lara, J. Xuand A. Chandra, "A Novel Algorithm based on Polynomial Approximations for an Efficient Error Compensation of Magnetic Analog Encoders in PMSMs for EVs," *IEEE Trans. Ind. Electron.*, to be published. 2016.
- [13] V.D. Aksenenko and S.I. Matveyev, "Digital Angle Sensor Self-Calibration: Two Approaches to Accuracy Increasing", *IEEE Proc. IMTC Conf.*, pp.543-547, 16-19 May 2005.
- [14] C. Attaianesse and G. Tomasso, "Position measurement in industrial drives by means of low-cost resolver-to-digital converter," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 6, pp. 2155–2159, Dec. 2007.

- [15] S. Sarma, V. K. Agrawal, and S. Udupa, "Software-based resolver-to-digital conversion using a DSP," *IEEE Trans. Ind. Electron.*, vol. 55, no. 1, pp. 371–379, Jan. 2008.
- [16] M. Benammar, "A novel amplitude-to-phase converter for sine/cosine position transducers," *International journal of electronics* 94, no. 4 (2007): 353-365.
- [17] Lukić, Jelena R., Dragan B. Živanović, and Dragan B. Denić. "A compact and cost-effective design of a linearization circuit used for angular position sensors," *FactaUniversitatis, Series: Automatic Control and Robotics*, vol. 14, no. 2, pp. 123-134. 2015.
- [18] M. Benammar, L. Ben-Brahim, M. A. Alhamadi, and M. El-Naimi, "A novel method for estimating the angle from analog co-sinusoidal quadrature signals," *Sens. Actuators A, Phys.*, vol. 142, no. 1, pp. 225–231, Mar. 2007.
- [19] Y. Wang, Z. Q. Zhu, and ZongyuZuo, "A novel design method for resolver-to-digital conversion," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3724 - 3731, June. 2015.
- [20] M. Benammar and A. Gonzales, "A Novel PLL Resolver Angle Position Indicator", *IEEE Trans. Instrum. Meas.*, vol. 65, no. 1, pp. 123-131, 2016.
- [21] M. Katakura, A. Toda, Y. Takagi, N. Suzuki, T. Kadoyama and H. Kushihara, "A 12-bits resolver-to-digital converter using complex twin PLL for accurate mechanical angle measurement", *Digest of Technical Papers. 2005 Symposium on VLSI Circuits*, 2005.
- [22] *Synchro/Resolver Conversion Handbook*, 4<sup>th</sup> ed., DDC, New York, 1994.
- [23] R.G. Lyons, "Sum of two sinusoids," For free publication by IOWEGIAN, 2011.
- [24] National Instruments, "NI myDAQ Specifications," myDAQ datasheet.