Cape Peninsula
University of Technology

SELECTABLE PMU ALGORITHMS FOR SMART GRID APPLICATIONS

by

ISMAIL JASSIEM

Thesis submitted in fulfilment of the requirements for the degree

Master of Engineering: Electrical Engineering

in the Faculty of Engineering and the Built Environment

at the Cape Peninsula University of Technology

Supervisor: Dr Q Bart

Bellville
04 September 2024

CPUT copyright information
The dissertation/thesis may not be published either in part (in scholarly, scientific or technical journals), or as a whole (as a monograph), unless permission has been obtained from the University

DECLARATION

I, Ismail Jassiem, declare that the contents of this thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

04 September 2024

Signed **Date**

ABSTRACT

Modern day power grids have become more complex with the inclusion of renewable energy sources, increasing non-linear loads, and a growing demand for electricity by industry. This has resulted in dynamic power grids and a need for advanced techniques for improving support, protection, communication and control within the power network. Wide area measurement systems (WAMS) are utilised for monitoring these dynamic power networks, providing an overall status of the electrical grid in real time. They play a crucial role in maintaining grid stability and safeguarding against faults, blackouts, and damages to the network. The phasor measurement unit (PMU) is an essential component in a WAMS as they provide accurate phasor measurements of voltage and current propagating in the power grid. Each phasor measurement is synchronised with UTC in order to analyse and compare the magnitude, phase angle and frequency values of the electrical waveform, occurring at different locations across the power grid. This phasor information is relayed to devices such as phasor data concentrators (PDCs) for aggregation, storage, and further processing and analysis.

This research work presents the study of dynamic synchrophasor estimation techniques with focus on the Taylor Fourier Transform (TFT). The aim of this research project is to design and implement a low-cost phasor measurement prototype for performing both P class and M class measurements. This shall be achieved by utilising two selectable versions of the TFT phasor estimation algorithm deployed on a STM32 Nucleo development board. By combining P and M class measurements on a single measurement device enhances the PMU for applications in both protection and monitoring. This notion is extensible to other smart grid applications and illustrates the multi-functional potential of a PMU on a single hardware platform. The performance and accuracy of the phasor and frequency estimations are evaluated via simulation and actual testing. The results are analysed and verified according to the requirements stipulated by the IEEE C37.118.1-2011 standard.

Keywords - Dynamic phasor, IEEE C37.118.1-2011, Least Squares Method, M class, P class, Phasor Measurement Unit (PMU), STM32 Nucleo board, Synchrophasor Estimation, Taylor Fourier Transform (TFT)

ACKNOWLEDGEMENTS

I wish to thank:

- The Almighty for granting me the strength, ability, knowledge and prosperity to undertake this endeavour.
- My children, and my wife Zulfaa, for their unconditional love, support, sacrifice and belief in me during this journey.
- My parents for their love and support.
- SARAO and the NRF for their financial support during my third year of study.
- My functional manager at SARAO, Tyrone von Balla, for all his efforts, support and encouragement.
- I would like to express my heartfelt gratitude to my supervisor, Dr Quinton Bart, for his unwavering commitment, guidance, support, and encouragement. Throughout the years of this study your help and guidance has greatly contributed to the completion of this work.

The financial assistance of SARAO and the National Research Foundation towards this research is acknowledged. Opinions expressed in this thesis and the conclusions arrived at, are those of the author, and are not necessarily to be attributed to SARAO and the National Research Foundation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Acronyms/Abbreviations	Definition
AC	Alternating Current
ADC	Analog-to-Digital Converter
API	Application Programming Interface
CPU	Central Processing Unit
DAC	Digital-to-Analog Converter
DER	Distributed Energy Resources
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DMIPS	Dhrystone Million Instructions Per Second
FE	Frequency Error
FPGA	Field-Programmable Gate Array
FPU	Floating Point Unit
GPIO	General-Purpose Input/Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
Hz	Hertz
IEEE	Institute of Electrical and Electronics Engineers
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IED	Intelligent Electronic Device
MSPS	Mega Samples Per Second
MU	Merging Unit
NI	National Instruments
PMU	Phasor Measurement Unit
RAM	Random-Access Memory
RFE	Rate of Change of Frequency (ROCOF) error
ROCOF	Rate of Change of Frequency
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SSEG	Small-Scale Embedded Generation
TFT	Taylor Fourier Transform
TVE	Total Vector Error
TWLS	Taylor Weighted Least Squares
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
UTC	Coordinated Universal Time
WAMS	Wide Area Measurement Systems

CHAPTER ONE
INTRODUCTION

1.1 Introduction

The opening chapter of this thesis introduces the background and context of this research project, describing the evolution of modern day power grids and their increasing need for smart grid technology to ensure grid stability and the prevention of blackouts. The high cost of commercial PMUs due to the nature of this specialised field is described, and the impact of this on research and development is highlighted. This chapter outlines the methodology employed in developing a versatile low-cost measurement device, in the form of a P and M class phasor measurement prototype, as a possible solution to address the research problem. This aims to demonstrate the multi-functional potential of a PMU on a single hardware platform achieved through selectable phasor estimation algorithms, and extensible to other applications.

1.2 Background

Global warming and the depletion of fossil fuels has increased the usage of renewable energy sources to supplement the power grid by means of distributed energy resources (DER). This along with the growing demand for electricity, has resulted in power grids evolving and becoming more complex. In recent years, South Africa's on-going energy crisis together with the cost reductions in renewable energy technology, have prompted the introduction of municipal small-scale embedded generation (SSEG) programmes, allowing residents to feed excess solar energy back into the power grid (City of Cape Town, 2023). This has contributed to the complexity of modern day power grids now requiring wide area monitoring, control and advanced metering to sustain a stable power network and provide protection against faults, black outs and damage to the power network infrastructure. The integration of wide-area monitoring systems provides operators with a wide-area view and state of the entire power grid in real-time, enhancing decision-making processes, control and overall management of the power grid.

Synchrophasor estimation is an essential component in wide area measurement systems (WAMS), enabling protection, measurement and analysis of modern day power systems. The phasor measurement units (PMUs) are the hardware devices used for performing synchrophasor measurements of voltage and current waveforms present in the power grid. These PMUs are strategically placed across the grid, typically at substations, to provide an overview of the state of the power grid. Synchrophasors are defined as phasor measurements synchronised with Coordinated

Universal Time (or UTC time). Accurate time synchronisation methods are utilised for time-tagging each measurement in line with UTC. The time synchronisation provides a means to record the exact time a measurement was taken in order to analyse and compare the magnitude, phase angle and frequency values of the electrical waveform, occurring at different locations across the power grid.

The dynamic nature of modern day power grids presents a certain degree of difficulty and complexity in measurement, and can often contain inaccuracies. Renewable and DER initiatives and the increasing amounts of modern day power electronic devices and non-linear loads connected to the power grid often leads to the presence of waveforms with fluctuating voltages and varying frequency. These waveforms are termed ‘dynamic’ and refer to the signal amplitude, phase angle, and frequency that are constantly changing. These dynamic signal conditions can lead to inaccuracies and erroneous measurements, having a detrimental effect on various substation automation, measurement and protection devices utilising this PMU data. This has resulted in the use of several phasor estimation methods and techniques, with varying levels of accuracy, for meeting the measurement performance required by continuously evolving power systems.

The Institute of Electrical and Electronics Engineers (IEEE) has released several versions of standards for governing synchrophasor measurements for static and dynamic power systems. Currently the most established and recognised are the IEEE C37.118 series of standards.

1.3 Awareness of problem

The IEEE C37.118.1-2011 standard defines two performance classes, namely P Class and M Class. The P Class measurements are geared for protection applications and fault condition scenarios that require fast response times. The standard defines M class measurements for situations where accurate and precise measurement is required, as opposed to responsiveness. Each performance class consists of its own set of performance requirements for compliance with the standard, and is typically chosen by the user based on the required application. The processing of M class measurements are more complex in comparison to P class, requiring longer observation and processing intervals, advanced estimation techniques and improved rejection of unwanted interference signals in order to achieve the required accuracy and precision. These differences have resulted in many commercial PMUs being designed to perform either P class or M class measurements, hereby limiting their use across different power system applications. Commercial PMU equipment are costly

due to the niche industry they serve, and this presents a challenge for research and development in smart grid and substation automation fields. The situation is worsened by the fact that many applications in power systems requires specific types of devices for measurement. This hinders researchers and institutions with limited budgets from conducting comprehensive studies and experiments within the field.

1.4 Problem statement

Power systems present many types of measurement based on the application. A prime example are the P and M performance classes stipulated for PMU's, where P-class is designated for protection applications and M class for providing accurate measurements and monitoring. There is a growing need to provide a more flexible and versatile PMU that can cater for both protection and measurement applications. This notion can be applied and extended to other smart grid applications such as smart meters, substation automation, plug-in electrical vehicles, Distributed Energy Resources (DERs), and Advanced Metering Infrastructure (AMI). The measurement requirements of these applications are each unique, however the underlying principles of measurement remain the same employing similar hardware architectures consisting of a data acquisition system and a signal or data processing component. Given these similarities, the prospect exists for the development of a single dedicated hardware platform incorporating these core functions. This would enable a single hardware platform to be used across a variety of applications.

This research explores various phasor estimation algorithms and techniques for deployment on a single hardware platform. This will focus on the selection of suitable phasor estimation algorithms for performing P class and M class PMU measurements. This research aims to demonstrate the practicality of employing a single platform with the appropriate software, to be used for a variety of different power measurement applications. This prospect shall provide a cost-effective solution for the development of PMU devices and thereby aid and facilitate research and experimentation within this field.

1.5 Aims

The aim of this research project is to design and implement a phasor measurement prototype for P class and M class measurements. This shall be achieved with the deployment of two phasor estimation algorithms on a single hardware platform. This prospect is significant as it demonstrates the capability of performing both P class and M class measurements, hereby providing a more generic and versatile instrument. The measurements are intended to comply with the IEEE C37.118.1-2011 standard,

encompassing static and dynamic power system conditions. The phasor estimation algorithm shall be implemented on a low-cost embedded hardware platform in the form of a phasor measurement prototype, serving as a proof of concept and a means for evaluating and verifying measurements. The prototype is intended to operate at the distribution level and ideally serve as a research and development platform aimed at research groups specializing in smart grid technology, substation automation, protection and wide area measurement system (WAMS) applications.

1.6 Objectives

The key objectives for achieving the intended aim of this research project has been identified and are listed as follows:

1. A research study and literature review of the IEEE C37.118.1-2011 standard, the core concepts and principles of Phasor Measurement Units (PMUs), and various synchrophasor estimation methods and techniques proposed in research. This shall encompass the operation and performance of various synchrophasor estimation algorithms, as well as their practical implementation on appropriate embedded hardware platforms.
2. The selection of an appropriate P class and M class synchrophasor estimation algorithm for this project for computing the phasor parameters onboard a low-cost embedded hardware platform. This shall be guided by the findings obtained from the literature review.
3. The development and implementation of the selected synchrophasor estimation algorithms in MATLAB, and the evaluation of its performances through software simulations.
4. The development of a P class and M class phasor measurement prototype. This will consist of the implementation and deployment of the P and M class synchrophasor estimation algorithms on an appropriate embedded hardware platform.
5. The evaluation and compliance testing of the actual real time measurements produced by the phasor measurement prototype, in conformance with the IEEE C37.118.1-2011 standard.

1.7 Hypothesis

Based on the research problem and proposed solution, the following hypothesis has been formulated for this research project:

“A PMU can be developed to perform P class and M class measurements on a single device using two selectable phasor estimation algorithms, thus improving its versatility and multifunctional capabilities .

The hypothesis raises the following questions that are intended to be answered on completion of this project:

- Can a single PMU be practically implemented for both P class and M class measurements? Most PMU’s conform to either P class or M class measurements. This project will attempt to produce a PMU capable of performing both measurement classes.
- Which algorithms and techniques will be employed? A study examining the performance of various algorithms and techniques shall be pursued. Based on the findings, an appropriate phasor estimation method and technique will be identified and selected.
- What level of performance and accuracy will the PMU prototype provide? The phasor measurement produced by the prototype shall be evaluated for compliance according to requirements based on the IEEE C37.118.1-2011 standard.

1.8 Delimitation of research

This study explores several time and frequency domain synchrophasor estimation algorithms proposed in research, but focuses on the Taylor Fourier Transform (TFT) as the dedicated dynamic phasor estimation algorithm employed in a phasor measurement prototype. This study does not include any benchmarking or detailed comparison of different phasor estimation algorithm performances, but rather examines the TFT algorithm in greater detail and demonstrates the implementation and practicality of employing the TFT algorithm as a selectable phasor estimation algorithm on the STM32 Nucleo-F767ZI development board. The development of the phasor measurement prototype is carried out on the STM32 Nucleo-F767ZI development board, with all embedded development confined to this hardware platform. The phasor measurement prototype is intended for demonstrating the performance and accuracy of the TFT phasor estimation algorithms, and therefore does not contain additional functionality such as GPS synchronisation and Ethernet data transfer, which are typically included in commercial PMUs. The IEEE C37.118.1-2011 standard compliance tests are conducted for evaluating the performance of the dynamic phasor estimation produced by the phasor measurement prototype, however certain tests which apply to a fully-fledged PMU have been excluded. This research

project does not include a comparison with commercial PMUs, as its focus is on implementing a P and M class PMU as a proof of concept and the evaluation of its performance is based on the compliance requirements stipulated in the IEEE Std C37.118.1-2011.

The simulation test are conducted in a MATLAB environment. Actual experimental tests are conducted with discrete and continuous analog signals generated by a function generator and is therefore not sourced from an actual power grid. It should be noted that the function generator may not completely replicate and represent a real world power grid, nor capture all the typical challenges presented by it.

1.9 Assumptions

The following assumptions are made for this research project:

- The phasor measurement prototype is intended for the measurement of single phase power systems only.
- The usage of this device is intended for local power grids operating at a voltage of 230 V AC and a frequency of 50 Hz, and is assumed that this is known a priori.
- This research project assumes input signals are received at the appropriate voltage levels supplied by instrument transformers. High voltage AC waveforms are converted to suitable AC voltages for measurement by an ADC.

1.10 Significance and motivation of research

Modern day power grids have increased in capacity and complexity, and often require dynamic models and techniques for monitoring critical events and preventing faults or blackouts. Synchrophasor estimation performs a key function in the monitoring and protection of power grids, and in this regard is an important component within this field of research. Many research studies explore synchrophasor estimation utilising DFT-derived algorithms based on frequency domain analysis and processing. This study focuses on time domain based algorithms and provides insight into the performance, limitations and effectiveness of this approach. The study aims to provide a method and technique for performing both P class and M class measurements on a single measurement device. This allows for phasor measurements requiring both fast response and high accuracy to be carried out on a single unit, eliminating the need for two application-specific devices. Given the high cost of PMUs, another motivation for this research project is to develop a low-cost phasor measurement prototype platform

aimed at facilitating rapid prototyping. This shall provide students and researchers with an inexpensive hardware platform for carrying out practical phasor measurement experiments and aid further advancements within the field.

1.11 Research design and methodology

This section discusses the methodology undertaken in this research project for achieving the proposed aims and objectives, and ensuring results are reliable and valid. The research design and methodology adopted for this project consists of the following points:

1. Literature review and investigation - The research commences with a literature review to establish a comprehensive understanding of synchrophasor estimation, phasor measurement units (PMUs) and the associated IEEE C37.118-2011 standard for measurement applications in power systems. The various synchrophasor estimation methods, PMU architectures and hardware platforms presented in research papers and literature are investigated. The literature review consists of information primarily obtained from research papers, associated text books, IEEE standards, and online-based literature.

2. Software development and simulations - A suitable phasor estimation technique and hardware architecture is chosen based on the findings obtained from the aforementioned literature review and investigation. A corresponding phasor estimation algorithm based on the selected estimation technique is implemented in the MATLAB software environment. Software simulations are carried out in MATLAB to verify the operation of the algorithm and evaluate its performance and accuracy according to the required performance class.

3. Design – A design of a phasor measurement prototype is developed and presented. This is based on information obtained from the literature review and research investigation. In the design, a suitable PMU architecture, data acquisition system, and processing method for executing the phasor estimation algorithms are identified. Additional peripheral functions essential to the design are also highlighted.

4. Implementation – The phasor measurement prototype is realised by implementing the design on a suitable embedded hardware platform and carrying out the deployment of the phasor estimation algorithms. Hardware and software development tools are utilised in the development, debugging and programming process. The initial phasor estimation algorithm requires porting to a suitable programming language supported by the processing unit onboard the embedded hardware platform. This physical prototype serves as a proof of concept and a means of testing and verifying the software developments and hardware design.

5. Test and evaluation - The test and evaluation procedure conducted for the phasor measurement prototype is adopted from compliance tests and verification guidelines set out in the IEEE C37.118.1-2011 standard. An analysis of the test measurement results determines the accuracy and performance of the phasor measurement prototype, and its compliance with the IEEE C37.118-2011 standard. During this test a combination of continuous analog and discrete test signals are applied as input test sources. The test measurements are carried out using an oscilloscope for evaluating actual real time measurements produced by the phasor measurement prototype.

1.12 Expected outcomes

The primary expected outcome of this research consists of a functioning phasor measurement prototype implemented on an embedded hardware platform. The main goal of this prototype is to demonstrate the ability to perform both P class and M class phasor measurements by means of two selectable algorithms. The measurement results are recorded for static and dynamic conditions, and evaluated for conformance with the IEEE C37.118.1-2011 standard. The thesis in itself is also one of the major outcomes of this research project, which documents the theoretical principles and findings obtained in this research investigation, as well as the subsequent design, implementation and final evaluation results.

1.13 Contribution of research

This research is significant as it explores the prospect of a single phasor measurement device performing both P and M class measurements, in contrast to the many research papers proposing phasor measurements designated to only one performance class. The importance of this prototype lies in its versatility, as it is a single device capable of performing P class and M class measurements, making it suitable for protection and

measurement applications. The knowledge and information gained during this project serves as a contribution to the field of smart grid technology and substation automation. The final phasor measurement prototype can aid researchers in this field by utilising the prototype to perform phasor measurements in a laboratory environment, for testing signal quality, power analysis, protection systems and other substation automation functions.

1.14 Thesis outline

This thesis consists of a total of seven chapters. A brief overview of each chapter is provided in the following:

Chapter One presents an introduction to this research project. This chapter describes the background, definition and aims of this research problem, and discusses its significance, methodology and expected outcomes.

Chapter Two reviews literature pertaining to synchrophasor estimation, PMUs, associated industry standards, and the relevant algorithms, hardware platforms and techniques employed in this field. The insights and knowledge gained are presented for application in this research project.

Chapter Three establishes a theoretical foundation of the underpinnings of phasor estimation. The underlying principles and operation of the TFT algorithm for computing phasor parameters are discussed.

Chapter Four verifies the TFT algorithm as a phasor estimator through functional testing simulated in MATLAB. The simulation test results are evaluated and analysed.

Chapter Five presents the development process of a phasor measurement prototype. The practical implementation of the TFT algorithm on an embedded hardware platform is discussed and demonstrated.

Chapter Six describes the test and verification procedure for evaluating the performance of the phasor measurement prototype. The test results and observations are presented and analysed.

Chapter Seven concludes with a summary of the deliverables, findings and outcomes of this research project. Future work and recommendations in this field of study are discussed.

1.15 Conclusion

This chapter provides an introduction to this research project by clearly describing the background and definition of the research problem to be investigated. The significance and objectives of this research, and the methodology employed for achieving the expected outcomes are discussed. The following Chapter Two presents the commencement of the research investigation in the form of a literature review. This establishes a theoretical foundation to be applied throughout the rest of this research project.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

This chapter aims to provide a foundation of knowledge on the topic of synchronized phasor (synchrophasor) measurements and Phasor Measurement Units (PMUs) in power systems. The purpose is to gain a thorough understanding of relevant topics, methods and techniques available on this subject, and to determine the current state of research within this field. This knowledge and information is essential in the research, design and development of PMU instruments and synchrophasor estimation algorithms.

The field of PMUs and synchrophasor measurements is well-defined, regulated and supported by the IEEE C37.118 standard published by the Institute of Electrical and Electronics Engineers (IEEE). This is the prevailing standard within this industry which most research work and development is in accordance with. The IEEE C37.118 provides technical specifications and guidelines to maintain consistency in the design and operation of applicable equipment and systems.

The literature review commences with a brief overview of the history of the standard. Key concepts and terms are defined and the necessary performance criteria are stipulated for ensuring an acceptable level of quality.

A focus area in this literature review is the different types of methods and techniques used in synchrophasor estimation. These are broadly categorised into either time domain algorithms or frequency domain algorithms. Research papers presenting the development and evaluation of various synchrophasor estimation algorithms are explored as per these two categories. Synchrophasor estimation algorithms and the development of PMUs intended for a specific performance class are also explored. The performance of these methods and techniques are noted, as well as practical implementations of these algorithms on hardware platforms. The most current methods and software tools used in the development, simulation and evaluation of synchrophasor estimation algorithms and PMUs are explored to determine their applicability and usage in this research project.

2.2 History of PMU and Synchrophasor Standards

The initial development of a PMU in 1987 by Virginia Tech, and the first commercial PMU developed in 1990 by Macrodyne, prompted a need for a standard to regulate and support PMU development and the advancing synchrophasor industry (Phadke & Thorp, 2008).

In 1995, the IEEE Std 1344-1995 was published as the first standard for PMUs and synchrophasor measurements (IEEE, 1995). This standard introduced definitions for several synchrophasor measurement concepts, and provided guidelines for measurement, synchronisation, processing, and communication performed by PMUs. A growing need for a more comprehensive standard for synchrophasor phasor measurements resulted in it being superseded by the IEEE Std C37.118-2005 (IEEE, 2006). This standard included steady-state tests, performance indicators and compliance requirements for evaluating performance of PMU measurements. In 2011, the standard underwent another revision and was divided into two parts: the IEEE Std C37.118.1-2011 which contained measurement definitions and requirements, and the IEEE Std C37.118.2-2011 which focused on communication and data transfer (IEEE, 2011b; IEEE, 2011a). The IEEE Std C37.118.1-2011 also included measurement requirements and test conditions for dynamic power system conditions, as well as two performance classes. In 2014, the IEEE Std C37.118.1-2011 was amended with the IEEE C37.118.1a-2014 to address certain inconsistencies encountered by users and relax some performance requirements found to be difficult to meet (IEEE, 2014).

2.3 The IEEE C37.118.1-2011 Standard

The IEEE C37.118.1-2011 standard provides a set of compliance tests and requirements for evaluating the accuracy of four key measurements produced by the PMU, namely, the magnitude and phase angle of the input waveform, as well as the associated frequency and Rate of Change of Frequency (ROCOF). The ROCOF can also be expressed as the derivative of the frequency. Each measurement is synchronized with a time source, such as UTC, and contains a time tag. A reference design is provided by the standard, although different estimation methods and techniques are allowed and encourage, as long as all the measurement results fall within the given criteria.

The IEEE C37.118.1-2011 standard specifies three performance indicators for determining the accuracy of measurements. This consists of the Total Vector Error (TVE), frequency error (FE) and rate of change of frequency (RFE). The measured amplitude and phase angle values are evaluated together as a single vectorial quantity by means of the TVE. The TVE specifies the difference between the theoretical calculated values and the actual measured values produced by the PMU. The measured frequency and rate of change of frequency (ROCOF) values are evaluated based on the FE and the RFE. The FE and RFE indicate the magnitude of the error present in the resulting frequency and ROCOF measurement. Measurement error limits are also provided which specify the value performance indicators should not

exceed in order to comply with the standard. A detailed description of the TVE ,FE and RFE, along with defining equations are presented in Chapter 3.

The measurements produced by the PMU are required to be transmitted and reported at a specified number of times a second and is referred to as the reporting rate (F_s). The IEEE C37.118.1-2011 standard expresses the reporting rate as a sub-multiple of the system nominal frequency (f_o) and is selectable by the user. Table 1 provides examples of PMU reporting rates in frames per second for a nominal system frequency of 50 Hz.

Table 2.1: PMU reporting rates for 50 Hz system frequency

System frequency	**50 Hz**		
Reporting rate (Fs)	10	25	50

The IEEE C37.118.1-2011 standard specifies two performance classes, namely, P class for protection applications, and M class for precise and accurate measurements. P class measurements require fast response times and minimal latency, but no explicit filtering is needed as a certain amount of interference and aliasing is allowed. M class measurements on the other hand require more complex filtering to guard against out-of-band interference and aliasing. Each performance class is specified by a unique set of compliance limits. The user selects the required performance measurement class based on the intended application. Measurement compliance consists of steady-state conditions as well as dynamic conditions. Steady-state conditions are where the magnitude, phase angle and frequency of the input signal remain constant within a given observation interval, while dynamic conditions are where these quantities vary with time.

For compliance testing, the standard provides each test with a set of independent requirements for meeting P class and M class measurements respectively. The dynamic compliance testing involves applying modulation and step changes to the input signal and the ramping of the system frequency, and measuring the output result. Out-of-band interference testing is a specific requirement for M class devices, evaluating the accuracy of measurements in the presence of unwanted frequency signals.

The measurement reporting latency is defined as the time interval from when a PMU is presented with an input signal to when a measurement value is produced at the

output. Factors such as the complexity of the estimation algorithm, length of observation windows, and filtering techniques have an influence on the resulting latency. The maximum reporting latency for both performance classes are governed by the reporting rate (F_s). M class allows for a longer duration in comparison to P class, as the higher level of accuracy for M class measurements requires more processing time. The maximum reporting latency for P class and M class are shown in Table 2.

Table 2.2: Measurement reporting latency (IEEE, 2011a)

Performance class	Max. measurement reporting latency (s)
P class	$2/F_s$
M class	$5/F_s$

The IEEE C37.118 remains the primary standard for regulating synchrophasor technology and the PMU industry. All functions of a PMU are defined, along with test procedures and performance requirements for evaluating measurements and designs. Consequently, all research proposed in this field are conducted in accordance with this standard.

2.4 Introduction to Phasor Measurement Units

The following provides a brief introduction of a Phasor Measurement Unit (PMU) and its architecture. PMUs are real-time instruments used for measuring the magnitude, phase, frequency and ROCOF of AC voltage or current waveforms in a power system. A PMU is typically located in a substation where the AC waveforms coming from feeders and buses are converted to suitable measurement levels via instrument transformers, and applied to the PMU input (Phadke & Thorp, 2008). The AC input signals are sampled by an Analog-to-Digital Converter, and the measurements are synchronised and time-tagged to UTC (Universal Time Coordinated). A Pulse-per-second signal supplied by an accurate time source, such as a GPS receiver, provides the synchronisation of measurements. This allows for all measurements from different locations across the power grid to be compared with each other as they all share a common time reference. The digitised signal is processed by a synchrophasor estimation algorithm and calculates the signal amplitude, phase angle, frequency and ROCOF. This phasor measurements are packaged into an ethernet packet and distributed on a network to Phasor Data Concentrators (PDC) for collecting, processing and storing of phasor information (Phadke & Thorp, 2008). A basic block diagram of the architecture of a PMU is shown in Figure 2.1.

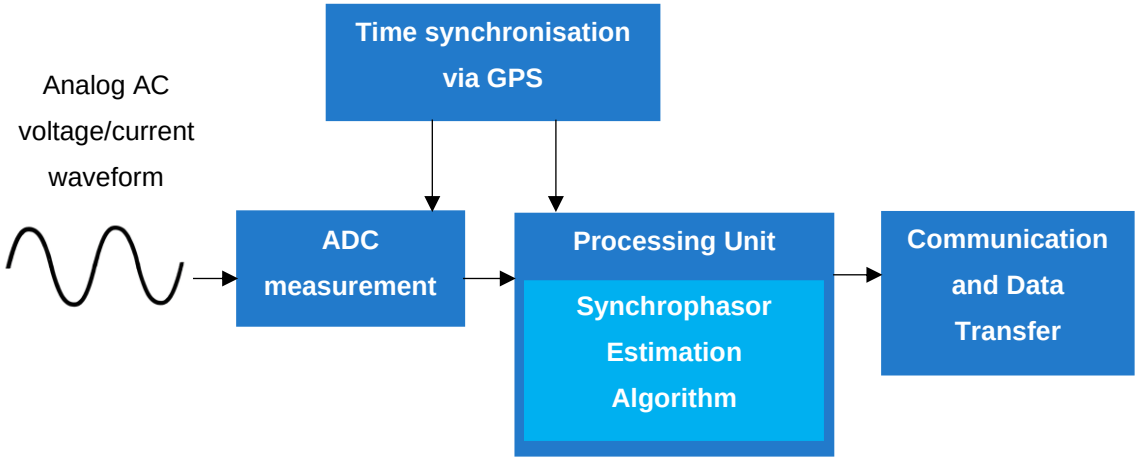

Figure 2.1: Block diagram of PMU architecture

2.5 Phasor Estimation Algorithms and Techniques

At the heart of a PMU lies the phasor estimation algorithm which is responsible for extracting the phasor parameters of an AC signal. The phasor estimation algorithm is a special mathematical method or technique for computing the amplitude and phase angle of a sampled signal. This operation contributes significantly to the overall accuracy of a PMU. The method involves applying a sequence of sampled values representing an AC signal to the input of the phasor estimation algorithm which computes the phasor parameters. A wide range of phasor estimation algorithms have been proposed and presented in research literature. To name a few, these include methods such as the discrete Fourier transform (DFT), interpolated DFT (IpDFT), sine-fitting algorithms, Kalman filters, phase-locked loops (PLL), the Taylor Fourier transform (TFT), and the wavelet transform (Monti, Muscas, Ponci, 2016). Each method differs in accuracy and complexity, and is usually targeted at specific applications and signal conditions. Even though each phasor estimation algorithm employs a unique technique for extracting phasor parameters, many methods share common underlying principles, and can thus be broadly categorized into either time domain or frequency domain algorithms. Frequency domain algorithms are composed of frequency analysis methods based on the DFT and it's variants, while time domain algorithms are based on either model matching or demodulation techniques (Monti, Muscas, Ponci, 2016). Furthermore, the mathematical technique of each phasor estimation algorithm is built upon an underlying static or dynamic signal model for computing the corresponding phasor parameters. Static models are based on purely sinusoidal signals that exhibit minimal to no transients, interference or phasor

fluctuations. As a result, static models are unable to accurately track fluctuations in the amplitude, phase angle, and frequency exhibited by modern day power grids and further degrades during off-nominal conditions and transients. To overcome this limitation, phasor estimation algorithms based on dynamic phasor models are employed to cater for these type of conditions. The dynamic signal model employed in dynamic phasor estimation algorithms allow for more movement and flexibility in the acquired signal, and thus is able to accurately estimate phasors that are subject to variations and interference.

The following two sections presents the basic principles and fundamentals of frequency domain and time domain algorithms, and explores several phasor estimation algorithms and methods proposed in research.

2.5.1 Frequency-domain DFT-based Algorithms

One of the more widely used frequency domain algorithms for static phasor estimation is the Discrete Fourier Transform (DFT). The DFT is renowned for its low computational complexity and rejection of unwanted harmonics. One point to note is the DFT relies on a static phasor model which assumes a signal with constant amplitude and frequency, and thus produces inaccurate results during off-nominal frequency deviations. During the measurement of off-nominal frequency signals, an integer number of periods does not fit uniformly within the observation interval, and thus causes the input signal energy in the DFT to stretch and smear across adjacent frequency channels. This phenomenon is known as spectral leakage and causes difficulty in determining the actual input signal frequency. The spectral leakage also causes the reduction of the frequency component of the measured signal, and is referred to as the scalloping loss. Windowing is a well-known technique employed to mitigate the effects of spectral leakage. Windowing consists of multiplying a discrete sequence with a finite-length window that tapers off to zero at each end, and hereby reducing any sharp transitions and discontinuities. In research literature, numerous frequency domain algorithms based on the DFT are presented and proposed for carrying out the phasor estimation . Many of these studies examines limitations of the DFT and explores different techniques and enhancements in order to improve its accuracy and performance as a phasor estimator. The following examines and discusses several frequency domain algorithms and techniques proposed in research papers for estimating the phasor parameters of an AC signal.

In a research paper presented by MacIi, Petri and Zorat (2012), simulation results confirm that the accuracy of a single cycle DFT can be significantly improved with the

inclusion of a window function. The scalloping loss can be compensated for using a technique known as the Interpolated Discrete Fourier Transform (IpDFT) conceived by Jain et al. (1979). This DFT-based algorithm extracts the parameters of the input waveform and interpolates the highest frequency components in the spectrum. According to Belega and Petri (2012), an IpDFT together with a Hann window yields results boasting TVE values which satisfy M-class accuracy thresholds for two-cycle and four-cycle observation intervals. In a study conducted by Derviškadić, Romano and Paolone (2017), an improvement to the IpDFT is presented which claims to be the first algorithm compliant with the IEEE std. C37.118.2011 for M-class PMUs. The technique is referred to as the iterative-Interpolated DFT (i-IpDFT) and iteratively estimates the spectral interference caused by the negative image and inter-harmonics, and subtracts the result from the frequency spectrum. In a later study by Derviškadić, Romano and Paolone (2018), an improved method of the i-IpDFT is proposed and is shown to be compliant with both P class and M class measurements. It should be noted that the aforementioned DFT and the IpDFT algorithms and variants thereof are all based on static phasor models, which presents a low computational burden and are simple to implement. These DFT-based methods provide good harmonic rejection but are sensitive to off-nominal frequency conditions.

2.5.2 Time-domain Algorithms

A significant amount of phasor estimators presented in research are based on time-domain techniques, and are thus categorised according to this approach. This category of phasor estimation techniques are also sometimes referred to as non-DFT type methods. As previously mentioned, this category of time domain techniques can be further subdivided into methods employing either demodulation or model matching.

The IEEE C37.118 standard provides a reference signal processing model for estimating the phasor parameters of a signal. The method is based on a static model employing demodulation and filtering. The standard states the reference signal model is provided merely for information purposes and is by no means intended as the preferred or recommended method for estimating phasors. Many other methods successfully developed by PMU manufactures and proposed by researchers are permitted and encouraged. In the reference signal processing model, the input signal is multiplied by the sine and cosine terms of a complex exponential generated by a quadrature oscillator operating at the nominal frequency. The real and imaginary components produced by the complex demodulator are filtered by either a P class or M class low pass filter, depending on the application. From this result, the magnitude and phase angle of the input signal can be determined. A block diagram of the

reference signal processing model provided by the IEEE C37.118 is shown in Figure 2.2.

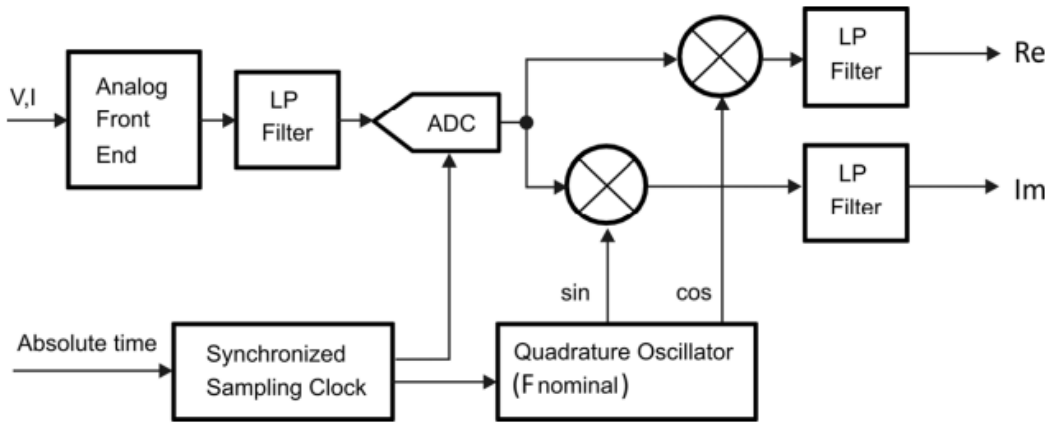

Figure 2.2: Reference signal processing model for phasor estimation. (IEEE, 2011)

Another widely used time domain technique commonly found in phasor measurement research are curve-fitting methods. These methods are also referred to as sine fitting methods, as they fit a sampled signal to a sinusoidal function and compute the phasor parameters by means of least squares regression. These methods are based on either static or dynamic phasor models. A detailed description of a trivial sine fitting algorithm is presented in Chapter 3.

In the IEEE 1057 standard for Digitizing Waveform Recorders, a three-parameter and four-parameter sine-fit algorithm is provided for the analysis and evaluation of waveform recorder equipment (IEEE, 2018). The three-parameter sine-fit algorithm estimates the amplitude, phase angle and DC offset of the input AC signal, provided that the signal frequency is known. In the case where the signal frequency is unknown and needs to be determined, the four-parameter sine fit algorithm is employed which follows an iterative procedure for calculating the parameters for amplitude, phase angle, frequency and DC offset. These techniques have been adopted from the IEEE 1057 standard and are now being utilised in phasor estimation algorithms. In a study conducted by Belega and Petri (2012a), the performance and accuracy of the three-parameter and four-parameter sine-fit algorithms are evaluated and analysed as synchrophasor estimators under steady-state and dynamic conditions. Both algorithms are analysed for half-cycle and one-cycle observation intervals. Overall, the simulation results produced in MATLAB indicate that the one-cycle algorithms provide more accurate estimates and comply with both P-class and M-class measurements. The

processing time of the three-parameter sine-fit algorithm is faster, as an accurate value of the signal frequency is assumed to be provided and therefore not needed to be calculated.

The types of sine-fit algorithms previously mentioned all rely on a static phasor model, where the sinusoidal parameters are assumed to be constant. An algorithm for determining a sinusoidal waveform as a dynamic quantity was first introduced in a research paper by de la O Serna (2006) and is referred to as the Taylor Fourier Transform (TFT). This method approximates the variations in amplitude and phase with a second-order Taylor polynomial, and uses the least squares method for calculating the result. The TFT is suited for both static and dynamic signals, with the former employing a zero-order Taylor model, and the latter requiring terms up to the second-order. An enhanced and more accurate version of the TFT is proposed in a research study by Platas, Garza and De La O Serna (2010). This method determines the Taylor parameters with windows to weight the errors of the least squares solution, and is known as the Taylor Weighted Least Squares (TWLS) method. The performance of the TWLS method is analysed in a research paper by Belega and Petri (2013) and indicates accurate measurement results, although the computation complexity is much higher in comparison to the traditional DFT. The reason for this is the TWLS requires computations with complex valued data. The real-valued TWLS algorithm (Belega & Petri, 2014) and the fast-TWLS algorithm (Belega & Petri, 2019) propose faster alternatives for implementing the TWLS method. The computational complexity of the Fast Taylor Weighted Least Squares Algorithm is reduced, and is suitable to employ on low-cost microprocessor platforms for real time synchrophasor applications (Belega & Petri, 2019) . The TFT algorithm and it's variants have become a popular choice as a synchrophasor estimator in PMUs. This method is favourable due to its performance under dynamic conditions. A detailed description of the TFT algorithm is presented in Chapter Three.

2.6 Performance Class Algorithms and Techniques

As mentioned in section 2.3, the IEEE C37.118 standard provides a set of performance criteria for evaluating PMUs. These requirements are categorised into two distinct measurement classes, namely P class and M class. Many PMU designs and phasor estimation algorithms developed and proposed in research are aimed at a specific performance class. The following takes a look at several phasor estimation algorithms and PMU designs presented in research specifically for meeting the requirements for either P class, M class or both performance classes.

2.6.1 P Class Algorithms and Techniques

P class measurements are specified for protection applications which typically require fast measurement response times, but less filtering capabilities. A particular phasor estimation algorithm for P-class PMUs is presented by Affijulla and Tripathy (2018) aimed at protection applications for smart grids. The algorithm is referred to as the Hilbert dynamic phasor estimator (HDPE) as it employs auto-correlation techniques and the Hilbert transform for estimating the phasor parameters of dynamic signals. The proposed phasor estimator is tested and verified under different dynamic conditions specifically for P class requirements as stipulated in the IEEE C37 standard. The result successfully indicated measurement accuracy and performance within the maximum TVE limit. In a second study conducted by Affijulla and Tripathy (2018a) a dynamic dictionary-based phasor estimation (DDPE) algorithm is presented also targeted at P-class applications. The algorithm utilises a primary and secondary dictionary matrix constructed from sinusoidal functions for coarse and fine phase angle estimation. The Taylor series approximation and a least squares technique are used for estimating the magnitude, frequency and ROCOF. The same set of dynamic tests conducted for the HDPE algorithm were used for evaluating the accuracy and performance of the DPPE which allowed for a comparison between the two methods. The test results of the DDPE demonstrated an overall improved performance in comparison to the HDPE and hereby confirmed its suitability for power system protection applications in compliance with the IEEE C37.118 standard.

2.6.2 M class algorithms and techniques

M class measurements are aimed at providing greater accuracy and precision in environments affected by interference and aliased signals, and thus allows for longer response times. A study by Bi et al. (2015) proposes a synchrophasor estimation algorithm based on the Taylor series approximation of a dynamic phasor for producing M class measurements. The algorithm uses the linear relationship between the second-order coefficients of the Taylor series expansion and the errors produced by the DFT averaging effect for achieving highly accurate phasor measurements. Two digital filters are also included for minimizing the effects of spectral leakage and out-of-band interference. Simulation and experimental testing conducted according to the IEEE C37.118 standard confirmed performance and accuracy suitable for M class measurements. As discussed in section 2.5.1, the iterative-Interpolated DFT (i-IpDFT) method is employed specifically for M class measurements and claims to be the first i-IpDFT algorithm satisfying M class PMU requirements (Derviškadić et al., 2017).

2.6.3 P and M class algorithms and techniques

This literature review has shown thus far that the majority of the PMU designs and phasor estimation algorithms proposed in research are developed for either P class or M class measurements. As synchrophasor technology is constantly being advanced and improved, newer studies have been presented which aim at developing phasor estimation methods for a single PMU device satisfying both performance classes. This allows for a single PMU to be deployed in the field for both measurement and protection applications. The following discusses a novel PMU design presented in research for achieving this outcome.

A research study conducted by Castello et al. (2013) presents the development of a P and M class compliant synchrophasor estimation algorithm based on an adaptive version of the Taylor-Fourier Weighted Least Squares (TFT-WLS) method. The study was aimed at developing a design for a single PMU capable of providing both accurate measurements and fast responses to transients for protection applications. The proposed method consists of a step change detector which monitors the AC signal and identifies any fast transient conditions based on the estimated phasor derivatives. Then a selection between two different configurations of the TFT-WLS algorithm is made for producing the best estimation results based on the signal conditions. Computed frequency estimations are fed back into the algorithm for improving measurement accuracy through frequency tuning. To reduce the computational burden caused by the inclusion of the frequency input parameter to the algorithm, dedicated transformation matrices are precomputed for frequency values deviating from the fundamental. The test results confirmed compliance with the IEEE C37.118 standard for P class and most M class requirements.

In another study presented by Castello et al. (2014), an enhancement to the aforementioned P class and M class PMU design is proposed. This design also utilises a step change detector and two different configurations of the TFT-WLS algorithm which run concurrently. A static and dynamic measurement channel allow for the sampled signal to be processed simultaneously by the two algorithms. A transient detector selects the output of the algorithm which is the most appropriate and provides the best performance for static or dynamic signal conditions. Each algorithm is customized for the measurement of either static or dynamic signals, and is defined by its order in the Taylor series approximation, window profile, and window length. The design is evaluated under static and dynamic conditions, and successfully meets all IEEE C37.118 standard compliance requirements, with the exception of the M-class ROCOF measurements which degrade in the presence of out-of-band interference.

2.7 Hardware Platforms

An important aspect in the development of a PMU prototype is identifying a suitable hardware platform for implementing the PMU design. Due to the present-day technological advancements, several hardware platforms and architectures are available and capable of performing the functions of a PMU. In the following, a survey is conducted encompassing the various types of hardware platforms found in literature for implementing phasor measurements and PMUs. A brief description of the hardware platforms and development methods employed in the design are presented in a chronological order. Focus is placed upon low cost devices which enables rapid development and prototyping which are desirable for research environments. The methods employed in developing these PMU prototypes are also included. The information obtained in this review is intended to guide the selection of a suitable hardware platform for this research project.

In a research paper presented by Romano and Paolone in (2017), a high performance low-cost PMU prototype is developed which implements the iterative-Interpolated DFT (i-IpDFT) algorithm on a NovTech IoT Octopus development board. The development board consists of an AD770 24-bit analog-to-digital converter and an Intel Cyclone V system-on-chip (SoC) which integrates a FPGA together with a dual-core ARM Cortex-A9 processor. The main components of the design consists of the i-IpDFT algorithm, the Modulated Sliding DFT (MSDFT) algorithm, and an ADC controller which are implemented using the programable logic of the FPGA. The development of the FPGA hardware description language (HDL) code is achieved through a model-based design flow using Simulink. The ARM processor is tasked with packaging output data and sending it out via Ethernet. The performance of the PMU prototype demonstrated low processing latency and good scalability in terms of the utilisation of FPGA resources, and overall confirms the successful implementation of the FPGA-based prototype.

A research study by Avalos-Almazan et al. (2018) showcases the development of a low-cost real-time phasor estimation prototype for measuring dynamic signals. This design utilises a Taylor Fourier Transform algorithm implemented in Python. The phasor estimation computation is executed on an ARM Cortex-A53 processor onboard a Raspberry Pi 3 Model B. The analog input signals are digitised by an AD7606 analog-to-digital converter and transferred to the Raspberry Pi for processing via a parallel communication protocol. The estimated phasor parameters are displayed on a graphical user interface also implemented in Python using PyQT. The developed prototype was tested with actual and simulated signals and produced reliable results demonstrating the successful tracking of dynamic signal behaviour.

An extensive study conducted by Adhikari, Hooshyar and Vanfretti (2019) explores the implementation of a single PMU design on Multiple Xilinx FPGA targets hosted on National Instruments (NI) Compact Reconfigurable I/O (cRIO) devices. The study proposes a metric for predicting the hardware and resource requirements for the FPGA-based PMU designs. The PMU designs are implemented in NI LabVIEW and deployed to the corresponding FPGAs hosted on several cRIO model devices. The digitisation of the analog input signal is achieved using NI C-series data acquisition modules. The cRIO onboard real-time processor and communication interfaces are used to broadcast the PMU output data to an Ethernet network. The design work flow commences with the PMU design implemented in a NI LabVIEW environment and converted to HDL. The HDL code is subsequently converted to bit-streams by Xilinx Vivado and uploaded to the relevant FPGA device. The development of the proposed predictive metrics are based on synthesis reports generated by Xilinx during compilation of designs. Verification and compliance tests in accordance with the IEEE C37.118 standard produced values with in specification, hereby confirming the validity of the PMU implementations.

In another study concerned with the low-cost implementation of a PMU presented by Delle Femine et al. (2019), a design approach utilising a STM32F407V microcontroller as a hardware platform is demonstrated. The input voltage or current waveform is digitised by the onboard 12-bit ADC, and the resulting samples are processed on the 32-bit ARM Cortex-M4 processor by means of the Interpolated Discrete Fourier Transform (IpDFT). The IpDFT phasor estimation algorithm as well as the ancillary functions of the PMU prototype are developed in the C programming language and deployed to the STM32 microcontroller. The estimated output phasor parameters are tested using the Fluke 6135A/PMUCAL measurement device and evaluated according to some of the P class criteria stipulated in the IEEE C37.118 standard. The test results produced for steady-state conditions at off-nominal frequencies and added harmonic interference demonstrated good accuracy and performance and were within compliance limits.

In a research paper presented by Garcia et al. (2020), the functionality of a PMU is incorporated into a commercial utility smart meter intended for low voltage distribution networks. The smart meter is based on the Texas Instruments MSP430f6776A Polyphase Metering System-on-Chip (SoC) microcontroller and is responsible for handling all metering functions. The phasor estimation is computed by a DFT-based algorithm running independently on a MT7688 32-bit microprocessor embedded on a Widora Bit hardware module. The two embedded devices interface via a UART-based

multitasking interface bus. The DFT-based algorithm as well as data transfer tasks are written and implemented in the C programming language. The performance of the PMU prototype is evaluated against a commercial RPV311 Digital fault recorder and PMU manufactured by General Electric. The test results of the PMU prototype during nominal and off-nominal conditions closely match the corresponding measurements produced by the RPV311. This study demonstrates the prospect of augmenting PMU functionality to existing smart grid infrastructure at a relatively low cost.

A study conducted by Da Silva et al. (2022) presents the investigation and development of a low-cost real-time PMU research prototype intended for the measurement of dynamic signals. The prototype employs the Taylor Fourier Transform (TFT) algorithm implemented on a Beaglebone Black development platform for computing the estimated phasor parameters. The Beaglebone Black consists of an ARM Cortex-A8 processor running a Linux operating system, and accompanied by two programmable real-time units (PRUs). Signal acquisition is achieved by interfacing with the AD7606-F4 16-Bit Data Acquisition System by Analog Devices, which is responsible for digitising the input analog signal. In this study, the verification of the PMU prototype is achieved by conducting a selection of dynamic tests facilitated through the Fluke 6135A/PMUCAL test equipment. The test results are within the maximum limits for compliance with the IEEE C37.118 standard and hereby validates the hardware platform as a viable option as a PMU prototype in a research environment.

A recent study conducted by Artale et al. (2023) investigates the use of a microcontroller for the measurement and analysis of harmonics in non-stationary signals for smart grid environments. The study is carried out by implementing a chirp-Z transform (CZT) algorithm on a STM32H723ZG Nucleo development board and assesses it's performance and accuracy when subjected to different non-linear load conditions. The measurement results are evaluated and compared against the same algorithm and test conditions implemented in a LabVIEW environment. The results indicated maximum percentage errors below 1.5% and a high spectral resolution, hereby validating and confirming the feasibility of the proposed solution.

2.8 Discussion

The following discussion highlights key findings obtained from this literature review on phasor estimation techniques and algorithms. This includes various PMU implementations based on the performance class, as well as the tools and hardware platforms utilised in development.

The traditional DFT and its variants remain a popular choice among PMU developers and researchers, due to their low computational complexity. However, these methods suffer in the presence of off-nominal frequency, out-of-band interference, and dynamic conditions. Many improvements to the DFT as a phasor estimator have been proposed to compensate for these limitations, one of which is the IpDFT, found in a number of recent studies. Another popular choice is the TFT-based methods, which rely on time-domain sine-fitting techniques. These are based on a dynamic phasor model and demonstrate excellent performance in both static and dynamic conditions. Although this method possesses a high computational complexity, this is no longer an issue with the advancement of processing hardware. The research and development of a P and M class PMU design is also observed. In this design, a TFT-WLS method is utilised, and successfully demonstrates its ability to meet P and M class compliant measurements simultaneously.

Several methods and approaches for implementing phasor measurements onboard an embedded hardware platform are presented. The selected hardware platforms in these research projects consists of a variety of development boards such as the Raspberry Pi, BeagleBone, CompactRio and STM32 Nucleo board. These development boards are based on different architectures which employ either FPGAs, microprocessors or microcontrollers as their central processing units for executing tasks and performing the phasor estimation computations. Signal acquisition is achieved through sampling on an onboard ADC or interfacing with an external ADC data acquisition module. The implementation of the phasor estimation algorithm on the hardware platform is largely dependent on the development process and workflow associated with the specific type of hardware and processing technology, as well as the semiconductor manufacturer. In the research papers we see development tools such as Xilinx Vivado, Simulink and LabVIEW being used for FPGA development in HDL, while microprocessor and microcontroller development is carried out in MATLAB, C and Python programming languages using a range of suitable vendor or third-party integrated development environments (IDE). The phasor estimation algorithms employed in these research projects include the Discrete Fourier transform (DFT), iterative-Interpolated DFT (i-IpDFT), Taylor Fourier transform (TFT) and Chirp-Z transform (CZT). Each method is targeted for certain types of signal conditions and presents its own unique performance and computational complexity. The accuracy and performance of the measurements produced by the developed PMU prototypes are assessed according to the IEEE C37.118 standard using various test methods and test

equipment. Overall, the IpDFT-based and TFT-based algorithms have shown to be a popular choice as a phasor estimator among researchers.

Based on the information obtained in this literature review, the TFT method was selected as the dedicated phasor estimation algorithm for computing the phasor parameters in this research study. The TFT algorithm is built upon the dynamic phasor model and offers good performance and accuracy in estimating dynamic signals commonly found in power grids. It is envisaged to employ a one-cycle TFT algorithm for performing the P class measurements, and a two-cycle TFT algorithm designated for M class measurements. This design scheme will be used in the development of the selectable PMU prototype. The selection of the hardware platform is provided in Chapter Five.

2.9 Conclusion

This chapter consists of a literature review which provides an overview of synchrophasor measurements with its regulatory standards, synchrophasor estimation algorithms and techniques presented in research papers, and the practical development implementation thereof on suitable hardware platforms. The fundamental concepts and principles of synchrophasor estimation for power systems are presented as per the IEEE C37.118 standard. Synchrophasor estimation algorithms based on time domain and frequency domain phasor methods, as well as performance class are introduced. Several papers presenting various phasor estimation algorithms based on these techniques are reviewed, highlighting their performance and compliance according to the IEEE C37.118 specification. The practical application of these algorithms is observed, identifying the types of hardware platforms utilised, as well as the methods and software tools employed to achieve this goal. The information gathered in this literature review will assist in the selection of an appropriate synchrophasor estimation algorithm and a hardware platform. In the following Chapter Three, the theoretical principles of phasor measurement and the mathematical underpinnings of phasor estimation algorithms are presented.

CHAPTER THREE
THEORY

3.1 Introduction

This chapter aims to provide a theoretical foundation of the fundamental principles and concepts concerning the phasor estimation and measurement of sinusoidal signals. The underlying mathematical theory and relevant equations supporting this research is presented. The representation of sinusoidal signals as phasors remains a core principle in the field of phasor estimation. In this chapter, we explore this concept applied as static and dynamic signal models and illustrate it's usage in the parameter estimation of AC sinusoidal signals. A technique provided in the IEEE C37.118.1-2011 standard for evaluating the performance and accuracy of phasor measurements is discussed, highlighting the key performance indicators and associated equations required for calculating these values. As previously indicated in the literature review of Chapter Two, the survey of the various phasor estimation techniques led to the selection of the TFT algorithm as the favourable phasor estimation method for extracting the phasor parameters from the input signal. This renders the TFT algorithm as an important and central topic within this research project. As the TFT algorithm falls within the time domain category of phasor estimation algorithms, the related theoretical concepts of least squares sinusoidal curve fitting, Taylor series expansion and matrix algebra are presented. All these concepts form part of the underpinnings of the TFT algorithm and are essential in understanding the fundamental operation of this phasor estimation method.

3.2 Introduction to the Phasor

The concept of the phasor was first conceived in 1893 by Charles Proteus Steinmetz who proposed a method of representing a steady-state sinusoidal signal by a constant complex quantity (Steinmetz, 1894). This approach allowed for easier calculations of AC voltage, current and power through the use of complex algebra. This consequently simplified the analysis of AC circuits. Until today this is regarded as the standard method of representing and analysing AC power systems.

A phasor is defined as a complex-valued vector rotating in the complex plane, and can also be viewed as a complex exponential signal. The complex value in polar form indicates the amplitude and phase angle of the sinusoidal AC voltage or current. In Figure 3.1, a phasor diagram of a complex phasor $\underline{A}$ and $\underline{B}$ is illustrated rotating counter-clockwise in the complex plane. The magnitude of the phasor $\underline{A}$ is indicated

by the symbol |$\underline{A}$|, the corresponding phase angle is represented by ϑ, and ω is the angular velocity.

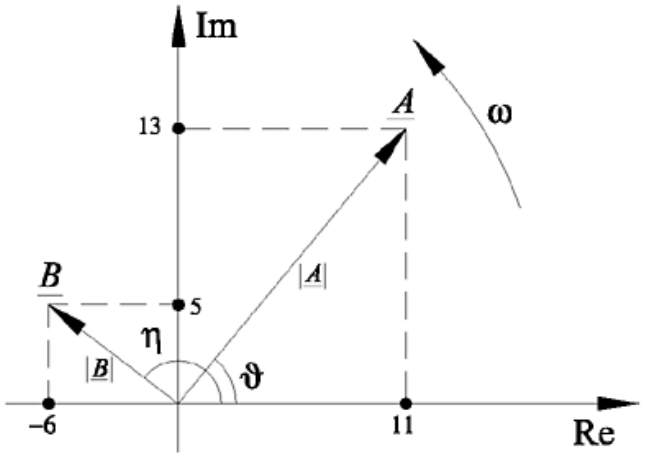

Figure 3.1: A phasor diagram of two phasors rotating in the complex plane (Mitolo, 2009)

In the following section, the mathematical relationship between the sinusoidal function and the phasor is examined.

3.3 Static Signal Model and Phasor Representation

In AC power systems, a steady-state electrical waveform can be represented by a sinusoidal signal expressed by the following trigonometric equation:

$$x(t) = X_m \cos(\omega t + \varphi) = X_m \cos(2\pi f t + \varphi) \tag{3.1}$$

where X_m is the peak amplitude of the signal, φ is the phase angle, ω is the angular system frequency, and f is the nominal system frequency. For this model, the amplitude and phase angle remain constant over the observation interval. To allow for analysis of the electrical waveform, the sinusoidal signal is represented as a phasor, containing the magnitude and phase angle of the signal. By using Eulers identity, a sinusoidal signal can be represented as a phasor, by taking the real component of the complex vector. The relationship between the time domain sinusoidal signal and phasor representation is shown in the following Equation (3.2) and Equation (3.3):

$$x(t) = Re\{X_m e^{j(\omega t+\varphi)}\} = Re\{(e^{j\omega t})X_m e^{j\varphi}\} = Re\{X_m e^{j\varphi}\} \tag{3.2}$$

where X_m is the peak amplitude and φ is the phase angle of the signal. The time factor term $e^{j\omega t}$ in Equation (3.2) is suppressed when moving from time to phasor domain, however it should be noted that ω represents the angular frequency of the signal.

Following from Equation (3.2), the signal $x(t)$, can be expressed by the phasor representation as shown in Equation (3.3).

$$x(t) \leftrightarrow X = \left(X_m/\sqrt{2}\right) e^{j\varphi} = \left(X_m/\sqrt{2}\right)(\cos\varphi + j\sin\varphi) = X_r + jX_i \tag{3.3}$$

where X denotes the phasor representation in complex form, $X_m/\sqrt{2}$ is the root-mean-square value and magnitude of the signal, and X_r and X_i are the real and imaginary components of the complex value in rectangular form. This aforementioned phasor representation is based on the static phasor model, and assumes a stationary signal with a constant amplitude, phase angle and frequency. In section 3.4 it is discussed how this model is adapted to accommodate for dynamic signals.

3.4 Dynamic Signal Model and Phasor Representation

The traditional signal model for the steady-state signal shown in Equation (3.1) assumes a constant amplitude, phase angle and frequency across the entire observation interval. To accommodate for the analysis of transient and dynamic behaviour presented by modern power systems, a dynamic phasor model is required. The dynamic phasor provides an enhancement to the static phasor model whereby the phasor parameters are considered as varying quantities. To represent the dynamic behaviour of signals present in modern power systems more accurately, a modulated signal or band pass signal is employed and expressed as

$$x(t) = X_m g\,(t) cos(\omega t + \varphi(t)) \tag{3.4}$$

where $X_m g(t)$ is the time-varying amplitude of the dynamic signal, $\varphi(t)$ is the time-varying phase angle, and $\omega = 2\pi f$ is the angular system frequency. Similarly to static signals, a dynamic signal can also be represented in phasor form, and is expressed by the following Equation (3.5):

$$X(t) = \left(X_m g(t)/\sqrt{2}\right) e^{j\varphi(t)} \tag{3.5}$$

where $X(t)$ represents the complex-valued phasor of the dynamic signal, and $X_m g(t)$ and $\varphi(t)$ is the varying amplitude and phase angle respectively. The steady-state and dynamic phasor models are the fundamental concepts for analysing electrical waveforms. All phasor estimation techniques and methods rely on these two models.

3.5 The Synchrophasor

The synchronous phasor, or more commonly referred to as the synchrophasor, is based on the same concept of a phasor as discussed in the previous section, however each measurement is synchronised to a common time source. In the case of PMUs, the Coordinated Universal Time (UTC) is taken as the absolute time reference and is obtained from systems such as GPS which broadcast the time information. Each measurement obtained from PMUs located in different regions across the power grid are synchronised and time-tagged according to UTC, allowing the measurements to provide a better overall view of the condition and state of the power grid. The magnitude and phase angle of the synchrophasor is expressed with reference to a cosine function at the nominal system frequency. This implies a sampled signal with a maximum value occurring at t=0 corresponds to a zero degree phase angle, and likewise, a positive zero-crossing occurring at t=0 indicates a phase angle of -90 degrees. A plot and phasor representation of a signal with a phase angle of zero degrees and -90 degrees is provided in the IEEE C37.118.1-2011standard and shown in Figure 3.2.

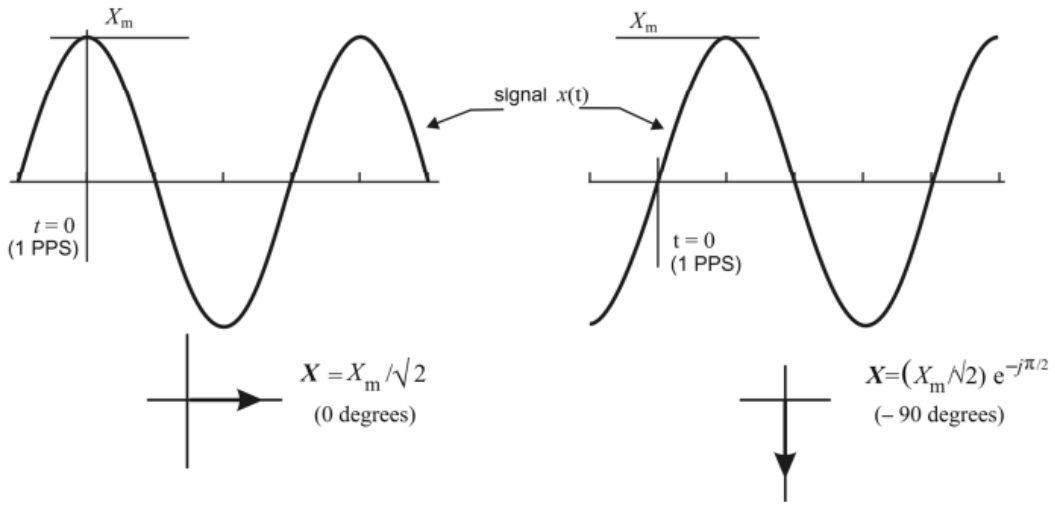

Figure 3.2: An illustration of a signal and phasor at a phase angle of zero degrees and -90 degrees (IEEE, 2011a)

3.6 Frequency Estimation and ROCOF

Traditionally, the phasor representation of a sinusoidal signal consisted solely of an amplitude and phase angle quantity. Two additional signal parameters are now included in the analysis of modern day power systems. These are the frequency and ROCOF quantities. The importance of these two parameters in AC analysis led to their definitions and requirements being included in the IEEE C37.118 standard (IEEE, 2011a). In the IEEE C37.118 standard, the frequency is defined as a time-varying quantity based upon the concept of instantaneous frequency. This model corresponds to the behaviour of power systems which often experiences frequency variations and fluctuations. Consider the sinusoidal signal in Equation (3.1) being rewritten as

$$x(t) = X_m \cos(\psi(t)) \tag{3.6}$$

where the cosine argument $\psi(t)$ denotes the instantaneous phase angle of the sinusoidal signal. The instantaneous frequency is expressed as the first derivative of the instantaneous phase angle, and is given by

$$f(t) = \frac{1}{2\pi}\frac{d\psi(t)}{dt} \tag{3.7}$$

where $f(t)$ is the instantaneous frequency and $\psi(t)$ is the instantaneous phase angle. Following from Equation (3.7), the ROCOF, measured in Hz/s, can be expressed as the first derivative of the frequency, or the second derivative of the instantaneous phase angle, and is given by

$$ROCOF(t) = \frac{1}{2\pi}\frac{df(t)}{dt} = \frac{1}{2\pi}\frac{d^2\psi(t)}{dt^2} \tag{3.8}$$

From Equation (3.6), the argument of the cosine, $\psi(t)$, can be expanded further, and expressed as

$$\begin{aligned} \psi(t) &= \omega_o t + \varphi(t) \\ \psi(t) &= 2\pi f_o t + \varphi(t) \\ \psi(t) &= 2\pi[f_o t + \varphi(t)/2\pi] \\ \psi(t) &= 2\pi[f_o t + \varphi(t)] \end{aligned} \tag{3.9}$$

By substituting Equation (3.9) into Equation (3.7), the following new equation for the frequency is produced:

$$\begin{aligned} f(t) &= \frac{1}{2\pi}\frac{d}{dt}2\pi[f_o t + \varphi(t)] \\ f(t) &= f_o + \frac{1}{2\pi}\frac{d\varphi(t)}{dt} \\ f(t) &= f_o + \frac{1}{2\pi} + \Delta f(t) \end{aligned} \tag{3.10}$$

where $\Delta f(t)$ is the frequency of deviation from nominal. In Equation (3.10) it is shown that the frequency of deviation is obtained from the first derivative of the phase angle, and that when summed with the fundamental frequency, produces the instantaneous frequency. Similarly, the ROCOF is now expressed as the second derivative of the phase angle, and is given by

$$ROCOF(t) = \frac{1}{2\pi}\frac{d\Delta f(t)}{dt} = \frac{1}{2\pi}\frac{d^2\varphi(t)}{dt^2} \tag{3.11}$$

The aforementioned principles and equations are significant as they are used in the TFT algorithm for determining the frequency and ROCOF of the input signal.

3.7 Measurement Evaluation

In the literature review, the performance indicators provided by the IEEE C37.118.1-2011 standard are introduced for evaluating and benchmarking phasor measurements. This consists of the TVE, FE and RFE. In the following section, the method for calculating these performance values and the equations involved are discussed.

3.7.1 Total Vector Error

An important factor in phasor measurement is the evaluation of the measured result for determining the performance, quality and accuracy of the phasor estimation. The IEEE C37.118.1-2011 standard provides a set of criteria and test procedures for bench marking the phasor measurement. The measurements are evaluated for different test cases and are based on a performance index called the Total Vector Error (TVE). The TVE is expressed as a percentage, indicating the vectorial difference between the true actual value and the measured or estimated value at a given point in time, and is given by

$$TVE(n) = \frac{\left|\hat{X} - X\right|}{|X|} \tag{3.12}$$

where $\hat{X}$ is the estimated phasor value and X is the actual phasor value.

A combination of the errors present in the measured amplitude, phase angle and timing, all influence and contribute collectively to a single TVE value. The IEEE C37.118.1-2011 standard provides the following equation for calculating the TVE:

$$TVE(n) = \sqrt{\frac{(\hat{X}_r(n) - X_r(n))^2 + (\hat{X}_\iota(n) - X_i(n))^2}{((X_r(n))^2 + ((X_i(n))^2}} \tag{3.13}$$

where $\hat{X}_r(n)$ and $\hat{X}_\iota(n)$ denotes the real and imaginary components of the estimated phasor in rectangular form, and $X_r(n)$ and $X_i(n)$ are the real and imaginary components of the actual or reference phasor. For most test cases, the TVE is required to be below 1% in order to meet compliance with the standard. To put this into perspective, a single magnitude error of 1% or a single phase error of 0.573° will produce a TVE value of 1%. In Figure 3.3, a phasor representation of a reference phasor is shown, along with the TVE region represented by a circle illustrating the maximum allowable magnitude and phase errors for meeting the 1% TVE limit. For 50 Hz systems, any timing error of 31.8us will also incur an error of 1%. This illustrates the high degree of accuracy and precision required to meet the IEEE C37.118.1-2011 standard compliance requirements. In Chapter Four and Six, the TVE is used extensively in the evaluation of the simulation and experimental test results.

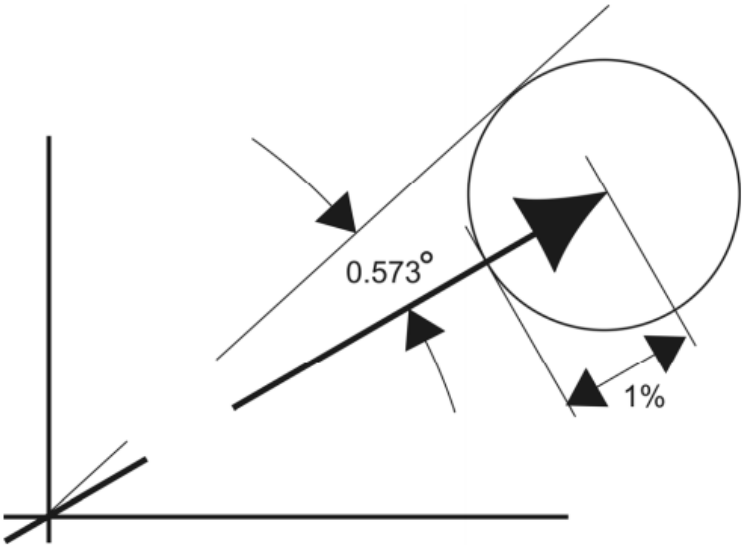

Figure 3.3: The TVE criterion for magnitude and phase angle errors illustrated in a phasor diagram (IEEE, 2011a)

3.7.2 Frequency Error and ROCOF Error

For measurement evaluation, two additional performance indices are provided by the IEEE C37.118.1-2011 standard, namely the frequency measurement error (FE) and the ROCOF measurement error (RFE). As the name suggests, these values provide an indication of the accuracy and quality of the estimated frequency and ROCOF measurement. The FE is defined as the absolute value of the difference between the actual frequency of the signal and the estimated or measured frequency. The FE is given in Hz and expressed as

$$FE = |f_{true} - f_{measured}| \tag{3.14}$$

Similarly, the RFE is specified as the absolute value of the difference between the actual ROCOF and the measured ROCOF given in Hz/s and expressed as

$$RFE = |RFE_{true} - RFE_{measured}| = (\frac{df}{dt})_{true} - (\frac{df}{dt})_{measured} \tag{3.15}$$

These equations are significant as they are used to evaluate the frequency estimation in the MATLAB simulations and experimental testing presented in Chapter Four and Six.

3.8 Least Squares Sine Fitting

In Chapter Two, a survey of several methods for phasor estimation are conducted. These are broadly categorised into time-domain and frequency domain algorithms. As time domain algorithms are largely based on curve-fitting techniques, the following introduces the trivial least squares sine-fitting algorithm to illustrate the basic principles and fundamentals of estimating signal parameters with time-domain algorithms.

Curve-fitting techniques operate by fitting the measured or sampled signal values to a given sinusoidal function, then estimates the parameters of the waveform. The simplest type of sine fitting algorithm makes use of the least squares method which minimizes the sum of the squared differences between the measured and fitted waveform, and is given by

$$SSE = \sum_{i=1}^{n} (e_i)^2 \tag{3.16}$$

where SSE denotes the sum of squares error, and e is the difference between the measured value and predicted value.

The IEEE 1057 Standard for Digitizing Waveform Recorders provides a standardised three parameter Sine fitting least squares algorithm for estimating three signal parameters using matrix operations (IEEE, 2018). The predefined mathematical function representing the sinusoidal signal for fitting the data to is expressed as

$$x(t) = X_m \cos(\omega t + \varphi) + C \tag{3.17}$$

where the three parameters of interest are the amplitude X_m, phase angle φ, and DC offset C. By using the trigonometric angle sum identity, Equation (3.17) can be expanded to

$$x(t) = X_m \cos\varphi \cdot \cos\omega t - X_m \sin\varphi \cdot \sin\omega t + C \tag{3.18}$$

To solve for the three parameters of interest, Equation (3.18) is represented in the matrix form as follows:

$$B = AX \tag{3.19}$$

$$\begin{pmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{pmatrix} = \begin{pmatrix} cos\omega(0) & -sin\omega(0) & 1 \\ cos\omega(1) & -sin\omega(1) & 1 \\ cos\omega(2) & -sin\omega(2) & 1 \\ \vdots & \vdots & \vdots \\ cos\omega(N-1) & -sin\omega(N-1) & 1 \end{pmatrix} \cdot \begin{pmatrix} X_m cos\varphi \\ -X_m sin\varphi \\ C \end{pmatrix}$$

where B is a vector of the signal samples, A is a matrix containing components of the signal model, and X is the vector of unknown coefficients. The least squares solution to the matrix is expressed as:

$$X = [A^T A]^{-1} A^T X \tag{3.20}$$

where A^T is the transpose of matrix A. The pseudo-inverse $[A^T A]^{-1} A^T$ is used for overdetermined systems where the number of equations exceeds the number of unknowns, rendering matrix A non-invertible. Finally, the signal amplitude X_m, and phase angle φ can be determined by the following equations:

$$X_m = \sqrt{(X_m cos\varphi)^2 + (X_m sin\varphi)^2} \tag{3.21}$$

$$\varphi = tan^{-1}\left(\frac{X_m \sin\varphi}{X_m cos\varphi}\right) \tag{3.22}$$

This method demonstrates the basic process followed for estimating and extracting the signal parameters using a time domain technique. Several other phasor estimation algorithms have been developed in this field and are all based on this method, however these contain variations such as employing different signal models and functions. The following section discusses how these signal models and functions are approximated using a well-known mathematical technique.

3.9 Taylor Series Approximation

The Taylor series approximation is an important concept used in phasor estimation to approximate the value of any type of signal as a function, especially in cases where it is difficult to compute directly. The approximation is formulated by an infinite sum of terms that represents the function at a given point. The series of terms consists of the function itself as well as it's derivatives. The Taylor series expansion can be truncated at any finite number of terms, with a higher-order series resulting in an improved accuracy, but more terms to compute. Typically this will involve a trade-off between

the accuracy required and computational resources available. The equations for the Tayor series is given by

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots \quad (3.23)$$

where $f(x)$ is the function to approximate, a is the point where $f(x)$ is evaluated at, and n is the number of terms or the order of the Taylor series. The term $\frac{f^{(n)}(a)}{n!}$ is also commonly referred to as the Taylor coefficients. The next section illustrates how the Taylor series approximation is incorporated in a phasor estimation algorithm and utilised in this research project.

3.10 TFT Algorithm

In the Chapter Two literature review, the TFT algorithm is selected as the phasor estimation method for extracting the phasor parameters of a dynamic signal. In this section the theoretical principles of the TFT algorithm for performing the phasor estimation are explored and discussed.

The Taylor Fourier Transform was initially proposed by de la O Serna and is based on the dynamic phasor model for estimating signals that possess time varying amplitudes, phase angles, and frequencies (de la O Serna, 2006; de la O Serna, 2007). The TFT is based intrinsically on a dynamic signal model composed of a modulated sinusoidal or bandpass signal as previously shown in Equation (3.4). By applying Eulers identity to Equation (3.4), and substituting the cosine function, the bandpass signal can be manipulated and expressed as a complex exponential given by

$$x(t) = \frac{1}{2}\left[a(t)e^{j\omega t}e^{j\varphi(t)} + a(t)e^{-j\omega t}e^{-j\varphi(t)}\right]$$

$$x(t) = \frac{1}{2}\left[p(t)e^{j\omega t} + p(t)^* e^{-j\omega t}\right]$$

$$x(t) = Re\left\{p(t)e^{j\omega t}\right\} \quad (3.24)$$

where the real component of the complex exponential represents the signal $x(t)$, and $p(t) = a(t)e^{j\varphi(t)}$ is the complex envelope of the band -pass signal, with $a(t)$ and $\varphi(t)$ being the amplitude and phase angle modulations acting upon the signal. The complex envelope is also commonly referred to as the dynamic phasor, and is approximated by

a k[th] order Taylor series expansion about the center of the observation interval. From Equation (3.24), the dynamic phasor $p(t)$ is approximated at t=0 and is represented by the following Taylor polynomial:

$$p(t) = p_0 + p_1 t + p_2 t^2 + \cdots + p_k t^k \tag{3.25}$$

where the Taylor series coefficients $(p_0, p_1, p_2, , p_k)$ are the derivatives of the dynamic phasor at the center of the observation interval. Higher order Taylor series polynomials produce more accurate approximations, however the increasing number of terms is computationally expensive, and a trade-off between accuracy and computational complexity needs to be made. When utilizing a zeroth-order Taylor approximation, static phasor estimates are produced which are inadequate for estimating dynamic signals . A second-order Taylor approximation is a suitable choice as it offers a good balance between accuracy and computational complexity and is used in (de la O Serna, 2007). With a second-order Taylor polynomial now approximating the complex envelope, Equation (3.24) can be expanded to

$$x(t) = \frac{1}{2}\left[(p_0 + p_1 t + p_2 t^2)e^{j\omega t} + (p_0^* + p_1^* t + p_2^* t^2)e^{-j\omega t}\right] \tag{3.26}$$

where p_0, p_1, p_2 are the second-order Taylor coefficients and p_0^*, p_1^*, p_2^* are the respective conjugates. By sampling the continuous signal $x(t)$ at a constant sampling interval T_s and renaming it to $s(n)$, a sequence of values are produced representing the equivalent discrete signal and is expressed as

$$s(n) = \frac{1}{2}\left[(p_0 + p_1 n T_s + p_2 n^2 T_s)e^{j\omega n T_s} + (p_0^* + p_1^* n T_s + p_2^* n^2 T_s)e^{-j\omega n T_s}\right] \tag{3.27}$$

From Equation (3.27), a system of linear equations are represented in matrix format for solving the coefficients by least squares method, and is shown in the following Equation (3.28)

$$\begin{pmatrix} s(0) \\ \vdots \\ s(N_h) \\ \vdots \\ s(n) \\ \vdots \\ s(N-1) \end{pmatrix} = \begin{pmatrix} N_h^2 e^{jN_h\omega} & -N_h e^{jN_h\omega} & e^{jN_h\omega} & e^{-jN_h\omega} & -N_h e^{-jN_h\omega} & N_h^2 e^{-jN_h\omega} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n^2 e^{-jn\omega} & n e^{-jn\omega} & e^{-jn\omega} & e^{jn\omega} & n e^{jn\omega} & n^2 e^{jn\omega} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ N_h^2 e^{-jN_h\omega} & N_h e^{-jN_h\omega} & e^{-jN_h\omega} & e^{jN_h\omega} & N_h e^{jN_h\omega} & N_h^2 e^{jN_h\omega} \end{pmatrix} \frac{1}{2} \begin{pmatrix} p_2 \\ p_1 \\ p_0 \\ p_0^* \\ p_1^* \\ p_2^* \end{pmatrix} \tag{3.28}$$

where N is the number of samples of the discrete signal $s(n)$, and N_h denotes the middle sample of the observation window, such that $N = 2N_h + 1$. This ensures N is always even. The observation window can be chosen to contain either a portion or multiple cycles of the input signal. The number of cycles employed in the calculation is usually appended to the name of the phasor estimation algorithm. For a second-order Taylor approximation, Equation (3.28) can be expressed by the matrix equation

$$\boldsymbol{S} = \boldsymbol{B}^{(2)} \cdot \boldsymbol{P}^{(2)} \tag{3.29}$$

where $\boldsymbol{S}$ is a vector of samples of the input signal, $\boldsymbol{B}^{(2)}$ is a matrix with column vectors of the form $n^2 e^{jn\omega}, ne^{jn\omega}, e^{jn\omega}$ and their complex conjugates, and $\boldsymbol{P}^{(2)}$ is a vector containing the Taylor coefficients and their complex conjugates (de la O Serna, 2006). The phasor estimates are obtained by determining the Taylor coefficients by least squares method, and is given by the equation

$$\widehat{\boldsymbol{P}} = (\boldsymbol{B}^{\boldsymbol{H}} \cdot \boldsymbol{B})^{-1} \cdot \boldsymbol{B}^{\boldsymbol{H}} \cdot \boldsymbol{S} \tag{3.30}$$

where $\widehat{\boldsymbol{P}}$ is the phasor estimate and $\boldsymbol{H}$ is the Hermitian transpose operator. It should be noted here that when computing the least squares solution of an overdetermined system, the pseudoinverse matrix $(\boldsymbol{B}^{\boldsymbol{H}} \cdot \boldsymbol{B})^{-1} \cdot \boldsymbol{B}^{\boldsymbol{H}}$ is required, as $\boldsymbol{B}$ is non-invertable.

The solution produces the estimated coefficients representing the first three terms of the second-order Taylor approximation and contains essential information, such as the speed and acceleration of the dynamic phasor. With these coefficients (p_0, p_1, p_2) the amplitude, phase angle, and their derivatives can be determined, and are given by the following equations:

$$\begin{aligned}
\hat{a}(t) &= 2|\hat{p}_0| \\
\hat{\varphi}(t) &= \angle \hat{p}_0 \\
\hat{a}'(t) &= 2Re\{\hat{p}_1 e^{-j\hat{\varphi}(t)}\} \\
\hat{\varphi}'(t) &= \frac{2}{\hat{a}(t)} Im\{\hat{p}_1 e^{-j\hat{\varphi}(t)}\} \\
\hat{a}''(t) &= 4Re\{\hat{p}_2 e^{-j\hat{\varphi}(t)}\} + \hat{a}(t)[\hat{\varphi}'(t)]^2 \\
\hat{\varphi}''(t) &= \frac{4Im\{\hat{p}_2 e^{-j\hat{\varphi}(t)}\} - 2\hat{a}'(t)\hat{\varphi}'(t)}{\hat{a}(t)}
\end{aligned} \tag{3.31}$$

When utilising the TFT algorithm for phasor estimation applications, the equations in (3.31) are used to determine the phasor parameters of the input signal. It should be noted that when employing a zero-order model only the amplitude and phase angle values are produced, whereas a second-order model includes the first and second derivative of the amplitude and phase angle. For a typical phasor estimation application, $\hat{a}(t)$ and $\hat{\varphi}(t)$ correspond to the estimated amplitude and phase angle of the input signal. The first derivative of the phase angle $\hat{\varphi}'(t)$, produces the frequency deviation with reference to the fundamental and nominal frequency, while the second derivative $\hat{\varphi}''(t)$ determines the ROCOF. The IEEE C37.118 standard states the frequency estimation can be reported as either the frequency deviation from nominal or the actual measured frequency. In this case the frequency deviation is estimated. The estimated frequency deviation and ROCOF are given by the following equations:

$$\widehat{FD} = \frac{\hat{\varphi}'(t)}{2\pi} \tag{3.32}$$

$$\widehat{ROCOF} = \frac{\varphi''(t)}{2\pi} \tag{3.33}$$

where the first derivative and second derivative of the phase angle are divided by 2π to obtain the frequency deviation $\widehat{FD}$ and the rate of change of frequency $\widehat{ROCOF}$ respectively. With the frequency of deviation now known and the nominal frequency of the power grid taken to be 50 Hz, the actual measured frequency of the signal can be determined.

3.11 Variations of the TFT algorithm as a phasor estimator

In the field of synchrophasor estimation and PMUs, several versions and implementations of the TFT algorithm have been developed for functioning as a phasor estimator. In this section a brief description of the zero-order two-cycle TFT, first-order two-cycle TFT, and second-order two-cycle TFT are introduced. Two-cycle refers to the amount of cycles of the input signal acquired and used in the TFT computation. The differences between the three aforementioned TFT versions mainly lies in the order of the Taylor polynomial for estimating the input signal. This determines the amount of terms used in the expression and has a direct bearing on the size of the coefficient matrix and the amount of computations taking place within the TFT algorithm. The computation time can be reduced by selecting a lower order Taylor polynomial, while a higher order will improve measurement accuracy for dynamic signals. The significance of the zero-order two-cycle TFT algorithm lies in the compact

size of the coefficient matrix utilised within the pseudo inverse matrix operation. In this method the Taylor polynomial expansion used in estimating the phasor is constrained to a single term. This minimizes the size of the coefficient matrix and hereby reduces the computation and processing time of the algorithm. As this method is based on a static signal model, the estimation is limited to amplitude and phase values only, and does not include rate of change and acceleration information about the signal. These values are calculated by the algorithm using the first two equations in (3.31). This method is suitable for the measurement of stationary signals that does not undergo any transients and oscillations. The remaining versions of the TFT algorithm are all based on the dynamic signal model which employ higher order terms in the Taylor polynomial. These versions are appropriate for the measurement of power systems that exhibit variations in amplitude, phase angle and frequency. For the first-order two-cycle TFT algorithm, the Taylor polynomial consists of the zero and first order terms. This increases the size of the coefficient matrix. This method utilises the first four equations in (3.31) to calculate the amplitude and phase angle of the input signal, including their first derivatives. The frequency deviation of the input signal can therefore also be determined. Finally the second-order two-cycle TFT algorithm consists of a Taylor polynomial composed of the zero, first, and second order terms. This method is an improvement to the previous but employs all six equations in (3.31) and includes acceleration information for determining the ROCOF.

3.12 Conclusion

In this chapter, a theoretical foundation for this research project is established. The intent is to provide a comprehensive understanding of the fundamentals of phasor estimation and the operation of the TFT algorithm. These concepts are essential for the development and testing of the TFT algorithm in a MATLAB simulation environment, as well for implementing the algorithm on a hardware platform. The start of this chapter illustrates the relationship between sinusoidal signals and the dynamic phasor model. This forms the basis of the TFT algorithm which allows for dynamic signals of modern power systems to be estimated accurately and effectively. A description of the performance indices for evaluating the accuracy and performance of phasor measurements are provided along with the corresponding equations. This information is of considerable importance as it is used extensively in the simulation and experimental testing in Chapter Four and Chapter Six. The underlying principles of the TFT algorithm which include the least squares sine fitting, Taylor series approximation and matrix algebra are explored. Using these concepts, the mathematical operation of the TFT algorithm is presented, illustrating the method of

computing the phasor parameters of a sinusoidal signal. The specific equation for calculating the amplitude, phase angle, frequency and ROCOF are included. Mathematically, this computation process comprises primarily of matrix operations for determining the least squares solution of the Taylor coefficients representing the dynamic phasor. The next step in this research is to provide a software implementation of the TFT algorithm. In Chapter 4, the mathematical process of the TFT algorithm is implemented in MATLAB code and evaluated. Thereafter compliance test are conducted via simulation to verify and evaluate the accuracy and performance of the TFT algorithm.

CHAPTER FOUR
SIMULATIONS TESTS

4.1 Introduction

In this chapter, the primary objective is the verification and evaluation of the Taylor Fourier Transform (TFT) algorithm as a phasor estimator, as previously discussed in Chapter Three. This involves implementing and translating the TFT algorithm to MATLAB code for conducting functional and compliance tests by means of MATLAB simulations. The one-cycle and two-cycle second-order TFT algorithms are the specific algorithm versions developed, verified, and evaluated in the MATLAB simulation environment. The simulation tests consists of a functional test for verifying the basic operation of the TFT algorithm, and compliance tests for evaluating the measurement performance and accuracy in accordance with the IEEE C37.118.1-2011 standard. The compliance tests are adopted from the IEEE C37.118.1-2011 standard and include static and dynamic test conditions. A test description for each test is provided detailing the necessary test conditions, test procedure, requirements and the relevant equations for conducting the simulation tests. The test results are recorded and the associated TVE, FE, and RFE values are calculated to determine the performance and accuracy of the phasor and frequency estimation. These values are examined to determine whether they fall within the maximum allowable limits stipulated for compliance with the IEEE C37.118.1-2011 standard. The estimation results obtained for the one-cycle and two-cycle TFT algorithm are also assessed to determine which is compliant and appropriate for P class and M class measurements respectively.

4.2 Development of TFT algorithm

In this research project, the programming and numeric computer platform, MATLAB, was selected as the primary tool for the development and simulation of the TFT algorithm. The algorithm is developed in MATLABs high-level programming language and undergoes several simulation tests in order to verify its operation and evaluate its performance.

As discussed in Chapter Three, the TFT algorithm is fundamentally based on a linear algebra operation for solving a system of linear equations, where the solution represents the phasor values of the input signal. A MATLAB implementation of the TFT algorithm for estimating the dynamic phasor is adopted from a research paper presented by Khodaparast and Khederzadeh (2015) which investigates the detection of faults during power swings. This particular MATLAB code was used as a foundation in the development of the TFT algorithm employed in this project.

In the following paragraph, a brief description of the developed MATLAB implementation of the TFT algorithm is discussed. An accompanying flow diagram is presented in Figure 4.1 which visually illustrates the step-by-step process, logic and flow of the algorithm.

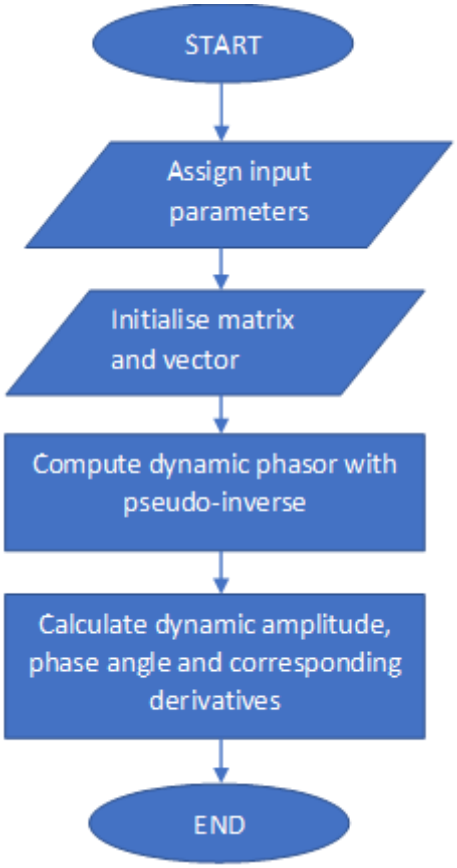

Figure 4.1: Flow diagram of the MATLAB implementation of the TFT algorithm

The algorithm commences by assigning values to the following input parameters required for the computation:

- The nominal frequency of the input signal,
- The number of samples per cycle,
- A vector representing the discrete samples of the input signal.

With these values provided, a column vector consisting of the discrete samples of the input signal, as well as a coefficient matrix are formulated. These form part of the system of linear equations and are represented in MATLAB in the form of a matrix and vector equation. The Pseudo inverse method is used to compute the best fit solution to the system of linear equations and thus determines the value of the phasor.

Consequently with this result, the phasor parameters of the input signal can be estimated by using the Equations in (3.31) presented in Chapter Three.

4.3 Two-cycle and one-cycle second-order TFT algorithm

The second-order two-cycle TFT is the primary MATLAB implementation of the TFT algorithm for this project. This version is based on a more accurate estimation, but will also introduce a longer computation time. By employing the second-order Taylor expansion of the dynamic phasor allows for the amplitude, phase angle, and their first and second derivatives to be estimated using all six equations in Equation (3.31). The first derivative of the phase angle corresponds to the difference between the fundamental frequency and the actual frequency of the input signal, while the second derivative corresponds to the rate of change of frequency (ROCOF). This version possesses the largest coefficient matrix due to the amount of terms in the second-order Taylor polynomial, and therefore requires the most computations to execute.

Additionally a second-order one-cycle TFT is developed in MATLAB by reducing the observation interval to one cycle in order to speed up the latency of the measurement. In this method, only one cycle of the input signal is sampled and processed. A shorter observation interval and fewer samples to process results in a quicker and more responsive measurement, however this may result in a reduction in accuracy and sensitivity to unwanted frequency components.

In Appendix A, the source code of the MATLAB implementation of the one-cycle and two-cycle TFT algorithm is provided.

4.4 Functional testing of the TFT algorithm

The phasor estimation algorithm is an essential component in the synchrophasor estimation process and therefore needs to be thoroughly verified. The functional testing forms part of the initial testing of the TFT algorithm. The main objective of the functional testing is to verify by simulation that the overall functionality of the developed TFT algorithm is operating correctly. In this section, the functional testing is conducted by carrying out simulation tests in MATLAB and examining the output results produced by the TFT algorithm for a given input. For the functional testing, only the second-order two-cycle TFT version is tested as this is based on the same principles and mathematical operations as the one-cycle TFT, and functions in a similar manner. The input data is a static sinusoidal signal based on Equation (3.1) provided in Chapter Three, and is generated in MATLAB with a given amplitude, phase angle, and frequency. This is applied to the input of the TFT algorithm for processing and computation. The observation window is taken across two cycles of the input signal,

resulting in 33 samples, and thus represents a sampling frequency of 800 Hz for a 50 Hz input signal. The output results of the TFT algorithm are examined to ensure that they match the corresponding values of the input signal, based on the specified amplitude, phase angle, and frequency. The functional testing is conducting using various values of amplitude, phase angle and frequency, to ensure the test encompasses a range of test values.

An example of the input test signal is shown in Figure 4.2 and is based on a stationary cosine function expressed by Equation (3.1). In this instance, an amplitude of 2 and a phase angle of 0 radians is applied to the input signal and corresponds with the values indicated by Test 1 shown in Table 4.1. From Figure 4.2, the observation window of two cycles of the discrete input signal is illustrated and amounts to a total number of 33 samples.

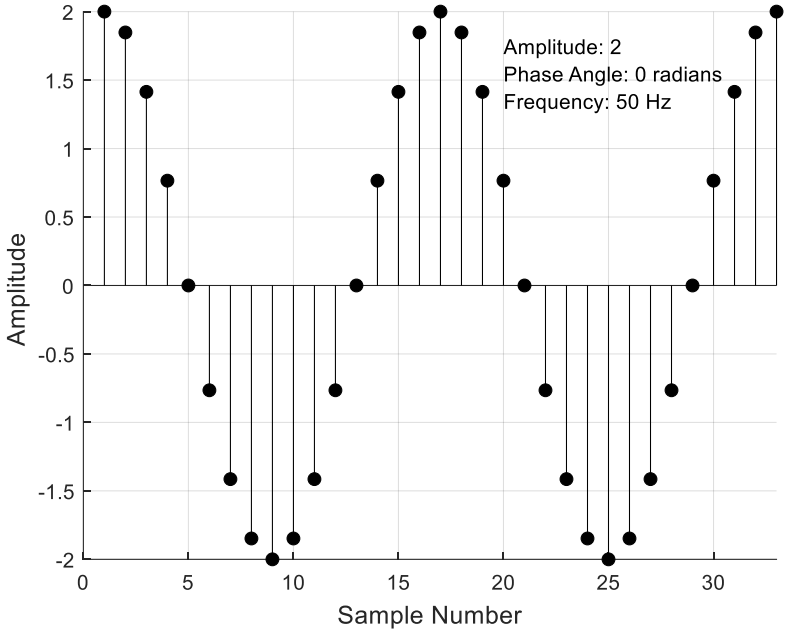

Figure 4.2 Discrete input test signal based on a stationary cosine function

4.4.1 Functional testing results

The functional testing results of the TFT algorithm are recorded and presented in Table 4.1. Based on these results, it is evident that the estimated amplitude, phase angle and frequency values computed by the TFT algorithm closely match the range of values applied to the input test signal. The results in Table 4.1 confirm and verify that the MATLAB implementation of the TFT algorithm performs successfully as a phasor estimator. It should be noted that an increasing error in the estimated phase angle is observed exclusively during off-nominal frequency conditions, especially for signal frequencies which deviate further from the nominal frequency of 50 Hz. This adverse

effect is addressed in section (4.7.1) by employing a phase angle compensation method which introduces an additional step in the phase angle estimation and hereby reduces the phase angle error caused by the off-nominal frequency.

Table 4.1: Functional testing results of the TFT algorithm

Test No.	Input signal parameters			Estimated signal parameters			Pass/Fail
	Amplitude	Phase Angle (radians)	Frequency (Hz)	Amplitude	Phase Angle (radians)	Frequency (Hz)	
1	2	-0	50.0	2	0	50.0	Pass
2	5	-0.524(-π/6)	50.0	5	-0.5236	50.0	Pass
3	10	-1.047(-π/3)	50.0	10	-1.0472	50.0	Pass
4	1	-1.571(-π/2)	49.5	1	--1.6336	49.5002	Pass
5	3	-2.094(-2π/3)	50.5	3	-2.0316	50.4998	Pass
6	4	-2.618(-5π/6)	49.0	4	-2.7437	49.0014	Pass
7	6	-3.142(-π)	51.0	6	-3.0159	50.9986	Pass

4.5 Overview of compliance testing

In the previous test, the basic functionality of the TFT algorithm as a phasor estimator is tested and verified through computer simulation. This is followed by a compliance testing stage, which is a more rigorous and in-depth test for evaluating the performance and estimation accuracy of the TFT algorithm. This is significant as there are several phasor estimation methods and techniques employed in research and industry, each providing their own unique level of performance and accuracy. The performance of both one-cycle and two-cycle second-order TFT algorithms developed for this project are evaluated by conducting a series of compliance tests obtained from the IEEE C37.118.1-2011 standard. The compliance tests are also conducted by means of simulation in a MATLAB environment, and are carried out according to the test description, conditions and accompanying test signals defined in the IEEE C37.118.1-2011standard. In this study, only specific tests relating to the performance and accuracy of the phasor estimation are considered. Lastly, the phasor estimation results are examined and evaluated according to the defined criteria stipulated in the IEEE C37.118.1-2011 standard.

4.5.1 IEEE C37.118.1-2011 compliance testing

The following section provides a description of the compliance tests stipulated in the IEEE C37.118.1-2011 standard. A set of requirements are specified for each test, which must be fulfilled in order to comply with the standard. This ensures that different phasor estimation methods and techniques all conform to a single industry standard. The requirements are categorised according to the performance class, with P class intended for protection applications requiring a fast response time, and M class

measurements providing an improved accuracy. The compliance testing consists of test scenarios which present and exhibit both steady-state and dynamic conditions, however the IEEE C37.118.1-2011 standard does not specify any test scenarios relating to extreme dynamic changes or extreme fault conditions. The steady state and dynamic compliance tests are defined as two independent testing events, each conducted separately.

4.5.2 Steady-state compliance tests

The steady-state compliance tests are specifically intended for evaluating the measurement of signals that exhibit a constant amplitude, phase angle and frequency. Each test configures a specific influence quantity of the input signal, then assesses the estimated output by evaluating the resulting TVE, FE, and RFE values. The TVE, FE, and RFE values must fall within the maximum specified limits in order to comply with the standard. Table 4.2 and Table 4.3 provides the conditions and requirements of the steady-state compliance tests along with the influence quantities and error limits, where F_s is the phasor reporting rate, f_0 is the nominal system frequency and f_{in} is the fundamental frequency of the input test signal.

Table 4.2: Steady-state measurement requirements (IEEE, 2011a)

Influence quantity	**Reference condition**	**Minimum range of influence quantity over which PMU shall be within given TVE limit**			
		P class		**M class**	
		Range	**Max TVE (%)**	**Range**	**Max TVE (%)**
Signal frequency range	$F_{nominal}$ (f_o)	± 2.0 Hz	1	± F_s/5 for 10 ≤ F_s < 25	1
Signal magnitude (Voltage)	100% rated	80% to 120% rated	1	10% to 120% rated	1
Phase angle with $\|f_{in} - f_o)\|$ <0.25 Hz	Constant or slowly varying angle	±π radians	1	±π radians	1
Harmonic distortion (single harmonic)	<0.2% (THD)	1%, each harmonic up to 50th	1	10%, each harmonic up to 50th	1
Out-of-band interference as described below	<0.2% of input signal magnitude		None	10% of input signal magnitude for F_s ≥ 10	1.3

Table 4.3: Steady-state frequency and ROCOF measurement requirements (IEEE, 2011a)

Influence quantity	Reference condition	Error requirements for compliance			
		P class		M class	
Signal frequency	Frequency = f_o ($F_{nominal}$) Phase angle constant	Range: f_o ± 2.0		Range: f_o ± 2.0 Hz for F_s ≤ 10	
		Max FE	**Max RFE**	**Max FE**	**Max RFE**
		0.005 Hz	0.01 Hz/s	0.005 Hz	0.01 Hz/s
Harmonic distortion (single harmonic)	<0.2% (THD) F_s ≤ 20	0.005 Hz	0.01 Hz/s	0.005 Hz	2 Hz/s
Out-of-band interference	<0.2% of input signal magnitude	No requirements		Interfering signal 10% of signal magnitude	
				0.01 Hz	0.1 Hz/s

The collection of steady-state compliance tests as defined in Table 4.2 and Table 4.3 are formulated into several individual tests. In the following, a brief description of each individual test is provided.

4.5.2.1 Frequency variation test

In this test, the frequency of the input reference test signal is varied from 2 Hz below the fundamental to 2 Hz above the fundamental. For both P and M performance classes, the maximum TVE value is specified at 1%, and the maximum FE and RFE values are specified at 0.005 Hz and 0.01 Hz/s respectively.

4.5.2.2 Magnitude variation test

During this test, the magnitude of the input test signal is varied from 80% to 120% for P class, and from 10% to 120% for M class. Both classes require TVE values that do not exceed 1%.

4.5.2.3 Phase angle variation test

In this test, the fundamental frequency of the input test signal is offset by 0.25 Hz. This consequently produces a slowly varying phase angle which can effectively be tested. Alternatively a phase angle from $-\pi$ to $+\pi$ can be applied to the input test signal. The resulting TVE value should not exceed 1%.

4.5.2.4 Harmonic distortion test

This test evaluates the performance of the estimation when subjected to an input signal containing harmonic interference. A single harmonic from the 2nd to the 50th order is added to the test signal. The test signal is expressed by the following Equation (4.1):

$$X_1 = X_m \cos(2\pi f_o t) + X_h \cos(2\pi f_h t) \quad (4.1)$$

where X_m and f_o are the amplitude and frequency of the fundamental signal, and X_h and f_h are the amplitude and frequency of the harmonic component. A harmonic level of 1% is specified for P class, and 10% for M class. A maximum TVE value of 1%, and a maximum FE value of 0.005 Hz is specified for both performance classes. The RFE values differ, with P class limited to 0.01 Hz/s, and M class limited to 2 Hz/s.

4.5.2.5 Out-of-band interference test

Interharmonic signals at frequencies above the Nyquist reporting rate (F_s/2) can alias into the passband. This test evaluates the capability of the front-end anti-aliasing filter to reject interharmonic signals that occur outside the passband. This test is similar to the harmonic distortion tests but instead pertains to the interharmonic (and subharmonic) frequency components present in the input signal. The test signal is represented by Equation (4.2):

$$X_1 = X_m \cos(2\pi f_o t) + X_i \cos(2\pi f_i t) \quad (4.2)$$

where X_m and f_o are the amplitude and frequency of the fundamental signal, and X_i and f_i are the amplitude and frequency of the out-of-band component. The passband is specified as half the reporting rate, below and above the fundamental frequency. For determining the performance of the out-of-band rejection, the test requires the input test signal to contain an external sinusoidal tone at 10% of the input signal magnitude, which varies from 10 Hz to the start of the pass band, and continues from the end of the pass band up to the second harmonic. This test is only required for M class measurements with reporting rates of 10 Hz and above, and specifies a maximum value of 1.3% for TVE, 0.01 Hz for FE, and 0.01 Hz/s for RFE.

4.5.3 Dynamic compliance tests

The dynamic compliance suite of tests provided by the IEEE C37.118.1-2011 standard are specifically intended for evaluating the measurement of input signals that present variations in amplitude, phase angle and frequency. These dynamic signals can occur in power systems during fault conditions, power swings and oscillations. The dynamic compliance tests are intended to replicate the aforementioned dynamic conditions by applying modulation, frequency ramps and step changes to the input test signal. The performance evaluation is conducted similarly to that of the steady-state tests, where the resulting TVE, FE and RFE values should be within the stipulated reference limits.

A brief description of each dynamic test procedure is provided below. All tests assume a single-phase input signal, a nominal power system frequency (f_0) of 50 Hz and a reporting rate (F_s) of 10 frames per second (fps).

4.5.3.1 Modulation test

The modulation test (also referred to as the measurement bandwidth test) assesses the measurement of input signals comprising of a sinusoidal modulation, where the amplitude and phase angle is modulated at a factor of 0.1. The modulated reference test signal is defined by Equation (4.3):

$$X_1 = X_m[1 + k_x\, cos(\omega t)] \times \cos[\omega_0 t + k_a\, cos(\omega t - \pi)] \quad (4.3)$$

where X_m is the amplitude of the input signal, ω_0 is the nominal power system frequency, ω is the modulation frequency in radians/s, k_x is the amplitude modulation factor, and k_a is the phase angle modulation factor.

The test requires the modulation frequency to be varied in steps of 0.2 Hz, ranging from 0.1 Hz to 1 Hz for P class measurements, and 0.1 Hz to 2 Hz for M class measurements. A maximum TVE of 3% is specified for all mentioned conditions. The FE and RFE limits are stipulated as 0.01 Hz and 0.2 Hz/s for P class, and 0.06 Hz and 2 Hz/s for M class. For evaluating the measurement of the modulated test signal, the expected theoretical phasor value, frequency, frequency deviation, and ROCOF values are given by the following Equation (4.4), Equation (4.5), Equation (4.6) and Equation (4.7) respectively:

$$X(nT) = \{X_m/\sqrt{2}\}[1 + k_x \cos(\omega nT)] \angle \{k_a \cos(\omega nT - \pi)\} \quad (4.4)$$

$$f(nT) = \omega_0/2\pi - k_a(\omega/2\pi)\sin(\omega nT - \pi) \quad (4.5)$$

$$\Delta f(nT) = -k_a(\omega/2\pi)\sin(\omega nT - \pi) \quad (4.6)$$

$$ROCOF(nT) = d/dt[f(nT)] = -k_a(\omega^2/2\pi)\cos(\omega nT - \pi) \quad (4.7)$$

where n is an integer and T is the phasor reporting interval.

4.5.3.2 Frequency ramp test

The frequency ramp test is intended for evaluating the measurement of signals experiencing frequency excursions, potentially caused by the loss of a generator or load. During this test the input system frequency is increased linearly at a rate of 1.0 Hz/s. The frequency ramp for both performance classes ranges from 48 Hz to 52 Hz, and specifies a maximum TVE value of 1%. The FE and RFE limits for P class is 0.01

Hz and 0.1 Hz/s respectively, and 0.005 Hz and 0.1 Hz/s for M class. The input test signal for this test is defined by the following Equation (4.8):

$$X_1 = X_m \cos(\omega_0 t + \pi R_f t^2) \tag{4.8}$$

where X_m is the amplitude of the input signal, ω_0 is the nominal power system frequency, and $R_f = df/dt$ is a fixed value for the frequency ramp rate in Hz/s.

The theoretical values of the test signal amplitude, phase angle, frequency, frequency deviation, and ROCOF are given by the following equations:

$$X(nT) = \left\{\frac{X_m}{\sqrt{2}}\right\} \angle\{\pi R_f (nT)^2\} \tag{4.9}$$

$$f(nT) = \omega_0/2\pi + (R_f)(nT) \tag{4.10}$$

$$\Delta f(nT) = (R_f)(nT) \tag{4.11}$$

$$d/dt[f(nT)] = R_f \tag{4.12}$$

where n is an integer and T is the phasor reporting interval.

4.5.3.3 Magnitude and phase step change test

The step change test evaluates the measurement performance of signals which present abrupt changes in amplitude and frequency. These are potentially caused by power system faults and switching operations. The test signal defined in Equation (4.13) is utilised in the magnitude and phase step change test and is expressed as:

$$X_1 = X_m[1 + k_x f_1(t)] \times cos[\omega_0 t + k_a f_1(t)] \tag{4.13}$$

where X_m is the amplitude of the input signal, ω_0 is the nominal power system frequency, $f_1(t)$ is a unit step function, k_x is the magnitude step size, and k_a is the phase step size.

This test assesses the response time, delay time and overshoot in the measurement. The response time is defined as the time from when the initial applied step change is observed, to the time when the stepped quantity has settled. The overshoot is considered as the maximum instantaneous value estimated during an amplitude or phase step change. The delay time is the measured interval from the instant a step change is applied to the time when this value reaches 50% of its final stepped value. One of the reasons for evaluating the delay time is to ensure that the group delay of any filtering system employed in the phasor estimation has been properly

compensated for. In Table 4.4, the maximum values allowed for the response time, delay time and, overshoot or undershoot are specified.

Table 4.4:Phasor, frequency and ROCOF performance requirements for input step change, where F_s = phasor reporting rate, f_o = nominal system frequency (IEEE, 2011a)

Step change specification	Reference condition	Maximum response time, delay time, and overshoot					
		P class			M class		
		Response Time (s)	|Delay time| (s)	Max overshoot/ undershoot	Response Time (s)	|Delay time| (s)	Max Overshoot/ undershoot
Magnitude = ± 10%, k_x = ± 0.1, k_a = 0	All test conditions nominal at start or end of step	1.7/f_o	1/(4×F_s)	5% of step magnitude	0.595	1/(4×F_s)	10% of step magnitude
Angle ± 10°, k_x = 0, k_a = ± π/18	All test conditions nominal at start or end of step	1.7/f_o	1/(4×F_s)	5% of step magnitude	0.595	1/(4×F_s)	10% of step magnitude
		Frequency response time (s)	**ROCOF response time (s)**		**Frequency response time (s)**	**ROCOF response time (s)**	
		3.5/f_o	4/f_o		0.869	1.038	

Measurement reporting latency compliance

The measurement reporting latency is defined as the time taken for a signal applied at the input of a system to be measured and it's parameters to be reported successfully. The measurement reporting latency is the total time delay incurred during a phasor measurement, and is contributed by many factors such as analog-to-digital conversion, window length, filtering, algorithm processing and computation time. This test ensures that the total delay is kept as short as possible, to prevent any measurement and reporting from being adversely affected or degraded. The maximum measurement reporting latency is dependent on the reporting rate. For compliance with the standard, the maximum measurement reporting latency for P class is specified as $(2/F_s)$, and $(5/F_s)$ for M class. By assuming a reporting rate of 10 fps, the latency limit results in a value of 200 ms for P class, and 500 ms for M class.

4.6 Compliance test simulations

In this section, the simulation of the compliance tests performed in MATLAB, are described and discussed. The second-order one-cycle TFT and the second-order two-cycle TFT algorithms are evaluated through the compliance test simulation, to gauge their performance and accuracy in accordance with the IEEE C37.118.1-2011 standard. The results of the two TFT algorithm versions are analysed in comparison, and their compliance with the IEEE C37.118.1-2011 are scrutinized.

The compliance test procedure provided by the IEEE C37.118.1-2011 standard, along with the associated input sinusoidal test signals are implemented and executed in

MATLAB by means of simulation. Only selected compliance tests from the IEEE C37.118.1-2011 standard have been chosen and implemented in MATLAB. The selection of compliance tests encompasses steady-state and dynamic conditions, and are listed as follows:

1. Frequency variation test
2. Magnitude and phase angle variation test
3. Harmonic distortion test
4. Out-of-band interference test
5. Modulation test
6. Frequency ramp test

The magnitude and phase step change test has not been performed, as this specific test is more applicable for evaluating the response time, delay time and overshoot of a fully developed PMU functioning as a complete system. The measurement reporting latency test has been excluded from the test simulation, however the test is conducted in Chapter Six, where the performance of the phasor estimation implemented on a hardware platform is evaluated and assessed.

4.7 Test simulation and evaluation procedure

A dedicated sinusoidal reference test signal is generated in MATLAB for the simulation of each compliance test. This test signal is based on a sinusoidal function and is defined in the IEEE C37.118.1-2011 standard. In the prior sections (4.5.3.1) to (4.5.3.3), a description and equation of the test signals for each compliance test is provided. This signal represents the sampled values or digitised version of the input test signal to be measured and estimated. A fundamental frequency of 50 Hz was selected for this test simulation. The simulation assumes a fixed sampling frequency of 800 Hz, resulting in a total number of 16 samples per cycle. The digitised test signal is applied to the input of the TFT algorithm which performs the processing and computing of the phasor parameters. A reporting rate of 10 fps is assumed for this test simulation. The test simulation applies for single phase conditions only.

The TVE, FE, and RFE performance indicators described in Chapter Three are employed to quantify the accuracy and benchmark the performance of the phasor and frequency estimation. Firstly, the performance and accuracy of the phasor estimation is determined by recording the estimated phasor parameters along with the expected phasor parameters, and calculating the resulting TVE value. As discussed in Chapter Three, the TVE provides an indication of the vectorial difference between the reference phasor value and the estimated value, and is expressed by Equation (3.13). A similar

approach is adopted for evaluating the estimated frequency and the rate of change of frequency (ROCOF). The reference and estimated frequency and ROCOF values are recorded and the resulting frequency error (FE) and ROCOF error (RFE) is calculated using Equations (3.14) and (3.15). For each test a maximum TVE, FE and RFE value is specified which should not be exceeded in order to meet compliance with the IEEE C37.118.1-2011 standard. These requirements vary according to the performance class (P or M class) desired.

4.8 Simulation test results

In this section, the test results obtained from the compliance tests simulated in MATLAB are showcased, providing an indication of the performance, accuracy and limitations of the one-cycle and two-cycle TFT algorithms. The steady-state compliance test results are presented initially, followed by the dynamic compliance test results. The results captured for each tests is examined and evaluated according to the IEEE C37.118.1-2011 standard.

4.8.1 Frequency variation test

The frequency variation test evaluates the performance of the phasor estimation algorithm when subjected to an input signal exhibiting an off-nominal frequency. When employing a fixed sampling rate, this measurement condition often results in inaccuracies in phase due to the mismatch between the input frequency and fundamental frequency. The frequency variation test is implemented in a MATLAB simulation where the input reference signal is generated for different off-nominal frequencies. For this test, the test conditions and requirements for P class and M class are effectively the same. The test input signal is applied at distinct off-nominal frequencies ranging from 48 Hz to 52 Hz. The accuracy of the phasor estimation is indicated by the TVE values calculated from the measurements produced by the one-cycle and two-cycle TFT algorithms. The resulting TVE values are presented in Figure 4.3 over the given frequency range. It can be seen that both algorithms are well below the specified TVE limit of 1%, with the best accuracy measurements coinciding with signal frequencies close to 50 Hz. The two-cycle TFT produces the highest TVE value of 0.17% at 48 Hz, and is still comfortably within the TVE limit. It must be noted that for the frequency variation test, the estimated phase angle is compensated for, due to a phase angle offset caused by the resulting off-nominal frequency. This effectively improves the accuracy of the estimated phase angle and thus the overall TVE value. The phase angle compensation is calculated and expressed by the following Equation (4.14) and (4.15).

$$\varphi_{comp} = \varphi_{estimated} + \Delta\varphi \tag{4.14}$$

$$\varphi_{comp} = \varphi_{estimated} + 2\pi\left(\frac{\Delta f}{f_o}\right) \tag{4.15}$$

where φ_{comp} is the compensated estimated phase angle, $\varphi_{estimated}$ is the estimated phase angle without any compensation, Δf is the frequency of deviation, and $f_o$is the fundamental frequency.

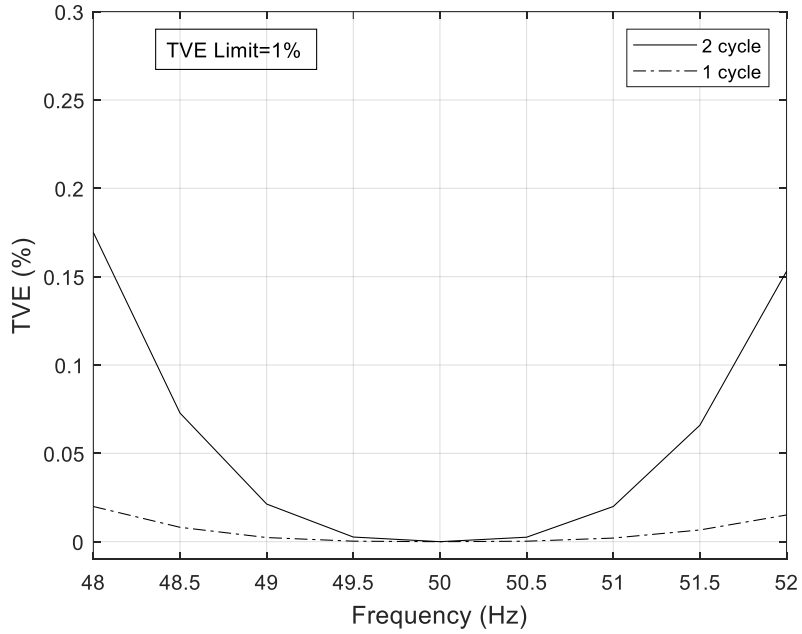

Figure 4.3: TVE results of one-cycle and two-cycle TFT for frequency variation test

The FE values presented in Figure 4.4 show that the one-cycle TFT algorithm is within the maximum error constraints across the entire frequency test range, however the FE values produced by the two cycle TFT are only admissible across a narrower band of frequencies.

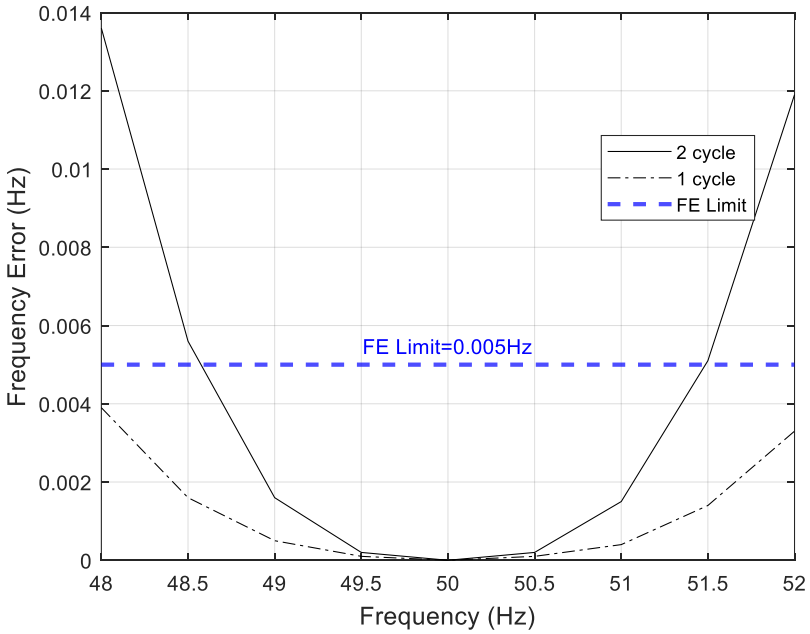

Figure 4.4: FE results of one-cycle and two-cycle TFT for frequency variation test

The RFE values for both TFT algorithm versions are presented in Figure 4.5. The results indicate that the 0.01Hz/s RFE limit is also only satisfied by the two algorithms around frequencies in the vicinity of 50 Hz.

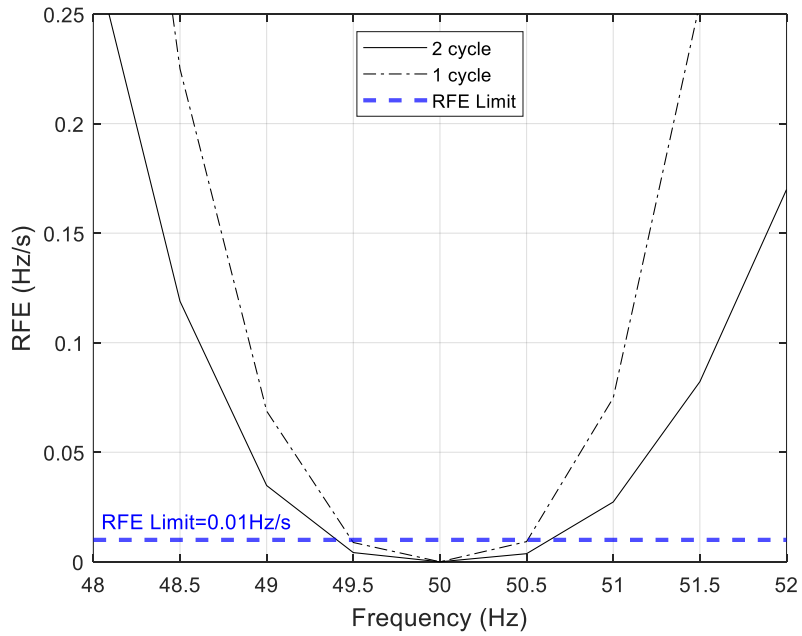

Figure 4.5: RFE results of one-cycle and two-cycle TFT for frequency variation test

4.8.2 Signal magnitude and phase angle variation test

The magnitude and phase angle variation tests are two separate tests conducted independently from each other, but are grouped together because of their common test conditions and requirements. In these static tests, the performance of the TFT algorithms are evaluated for estimating signals composed of different values of magnitude and phase angle. For both tests, the IEEE C37.118 standard only specifies the TVE as the performance indicator. The FE and RFE values are not required and are therefore omitted.

The magnitude variation test assesses the accuracy of the estimation when the input signal amplitude is set to values ranging from 10% to 120% of the nominal value for M class, and 80% to 120% for P class. This range of amplitude values are defined specifically for voltage measurement applications, and include compliance requirements for P class and M class measurements. The phase angle variation test is evaluated for the measurement of phase angle values ranging from 0 to $7\pi/4$ radians (0 to 315°).

The simulation results of the magnitude and phase angle variation test for the one-cycle and two-cycle TFT estimation methods are presented in Figure 4.6 and 4.7 respectively. In Figure 4.6, the TVE values produced by both one-cycle and two-cycle TFT algorithms are shown to be negligibly small for all amplitude values applied in this test (10% to 120%).

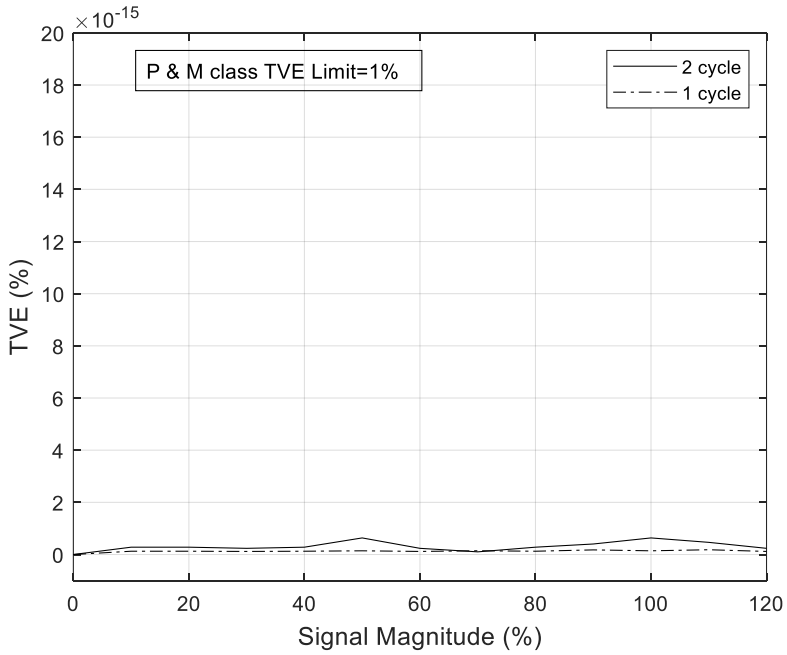

Figure 4.6: TVE results of one-cycle and two-cycle TFT for signal magnitude variation test

A similar outcome for the phase angle variation test is shown in Figure 4.7, where very small TVE values are produced and maintained for all applied phase angle values. The results produced for both magnitude and phase angle variation tests are well below the P and M class TVE limit of 1%, and thus successfully meets the compliance criteria. In conclusion, this test illustrates the high level of accuracy in the estimated amplitude and phase angle values produced by the one-cycle and two-cycle TFT algorithm versions.

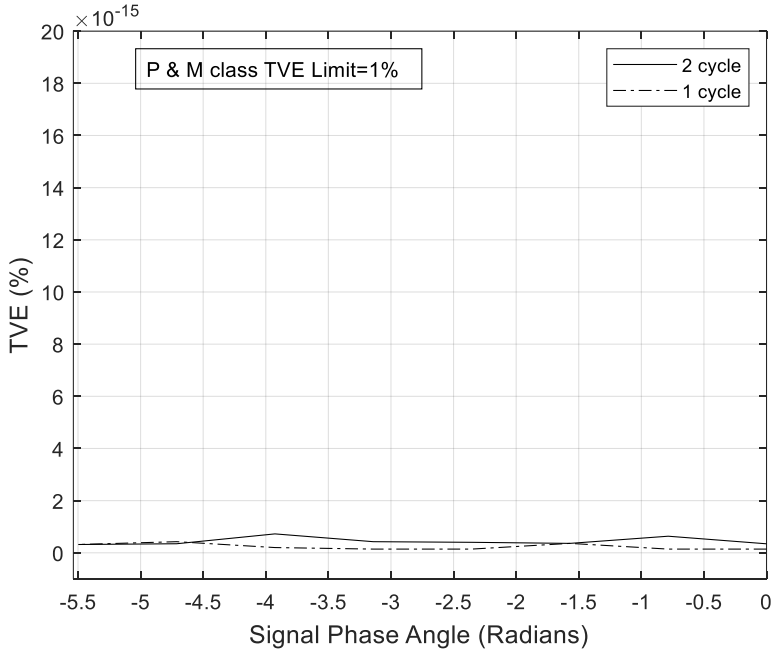

Figure 4.7: TVE results of one-cycle and two-cycle TFT for phase angle variation test

4.9 Harmonic distortion test

The harmonic distortion test is conducted by evaluating the accuracy of the phasor estimation produced by the TFT algorithm, when applied with an input test signal containing harmonic components. The simulation test is conducted for a harmonic level of 1%, which is the required condition for P class, and a harmonic level of 10% for M class. The harmonic frequencies are applied at 100 Hz, 150 Hz, 200 Hz, 250 Hz, 500 Hz, 1250 Hz and 2500 Hz. Figure 4.8 displays the TVE results of the phasor estimation for P class which applies a harmonic interference level of 1%. It can be seen here that the one-cycle and two-cycle TFT versions are both within the allowable TVE limits of 1%, and this is applicable up to the 50th harmonic.

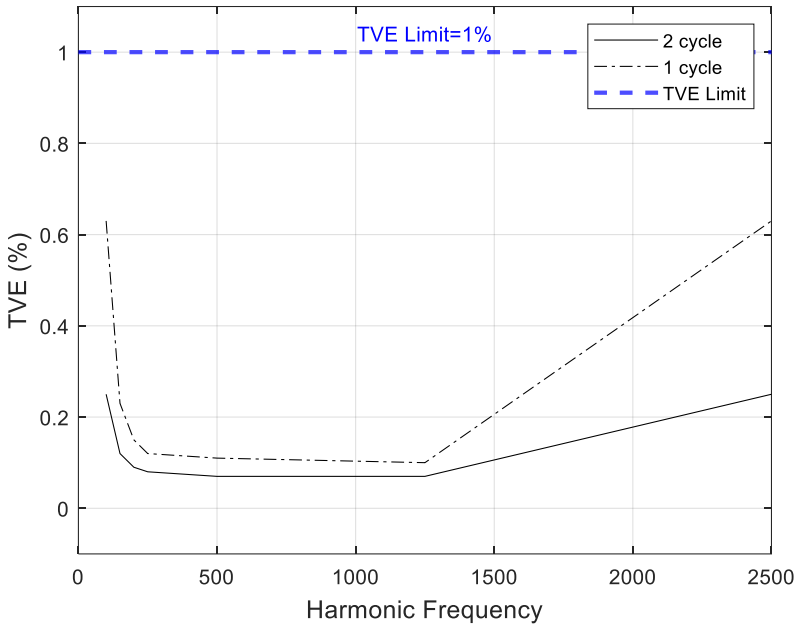

Figure 4.8: TVE results of one-cycle and two-cycle TFT for 1% harmonic interference test

Figure 4.9 presents the TVE results for M class where the harmonic levels are increased to 10%. The results indicate that the TVE values produced only by the two-cycle TFT algorithm are admissible, and are limited to the harmonic frequencies ranging from 200 Hz to 1250 Hz. Overall the two-cycle TFT demonstrates the better performance in phasor estimation.

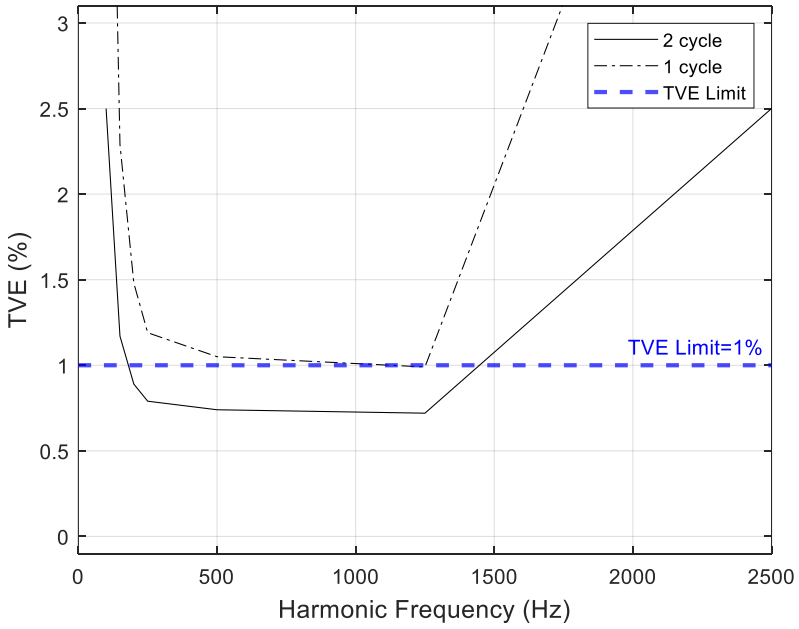

Figure 4.9: TVE results of one-cycle and two-cycle TFT for 10% harmonic interference test.

The FE results of the 1% and 10% P and M class harmonic interference levels are shown in Figure 4.10 and Figure 4.11 respectively. Both the one-cycle and two-cycle TFT algorithms fail to meet the compliance criteria for P class as well as M class, as their results exceed the 5 mHz allowable frequency error margin. The FE results suggest that the presence of harmonic interference tends to affect the performance of the frequency estimation.

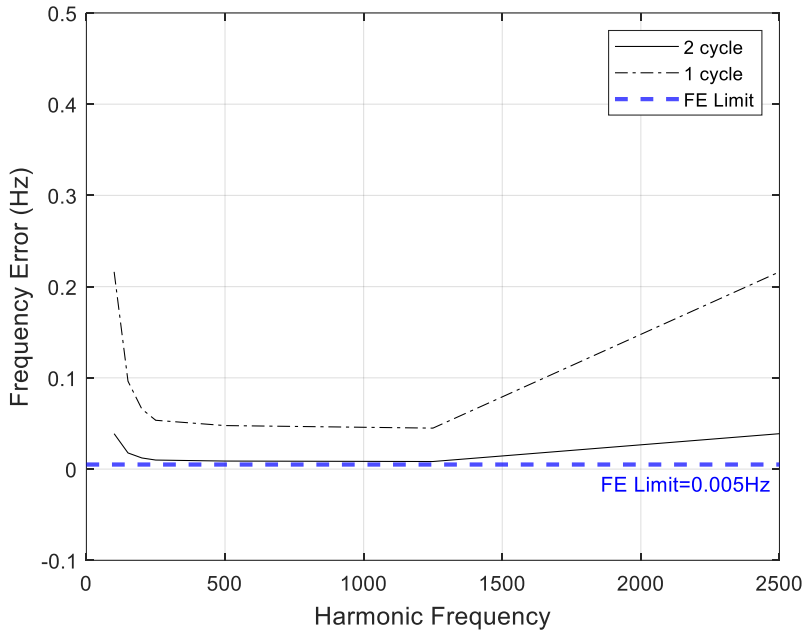

Figure 4.10: FE results of one-cycle and two-cycle TFT for 1% harmonic interference signal

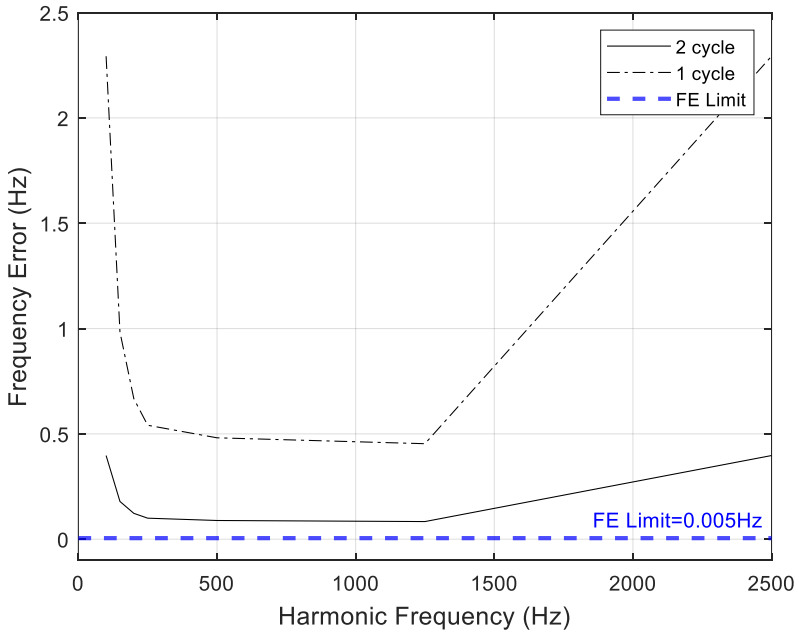

Figure 4.11: FE results of one-cycle and two-cycle TFT for 10% harmonic interference signal

The RFE results produced for the harmonic interference levels of 1% and 10% are shown in Figure 4.12 and Figure 4.13 respectively. The one-cycle and two-cycle TFT algorithms for both interference conditions are shown to produce RFE values negligibly small and near zero, and thus clearly within the P class and M class limits of 0.01 Hz/s and 2 Hz/s.

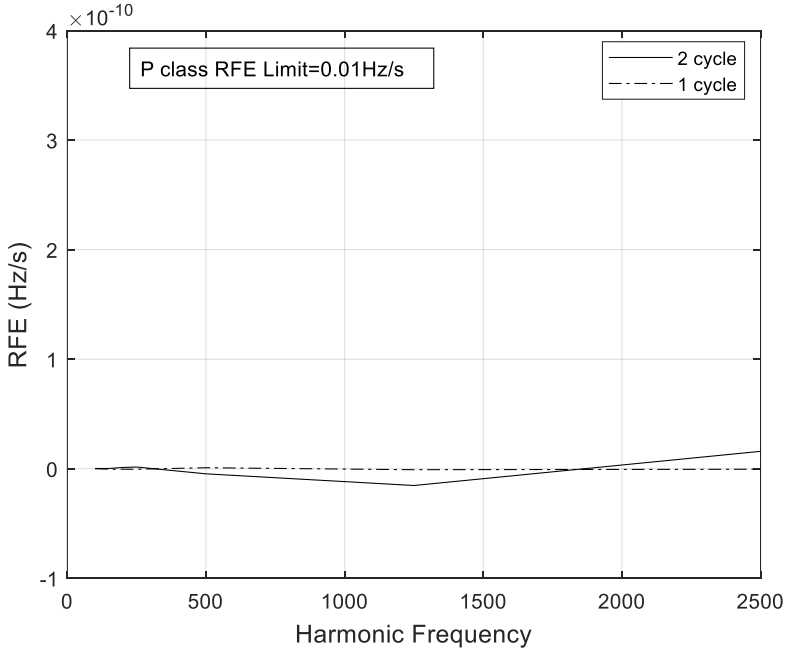

Figure 4.12: RFE results of one-cycle and two-cycle TFT for 1% harmonic interference signal

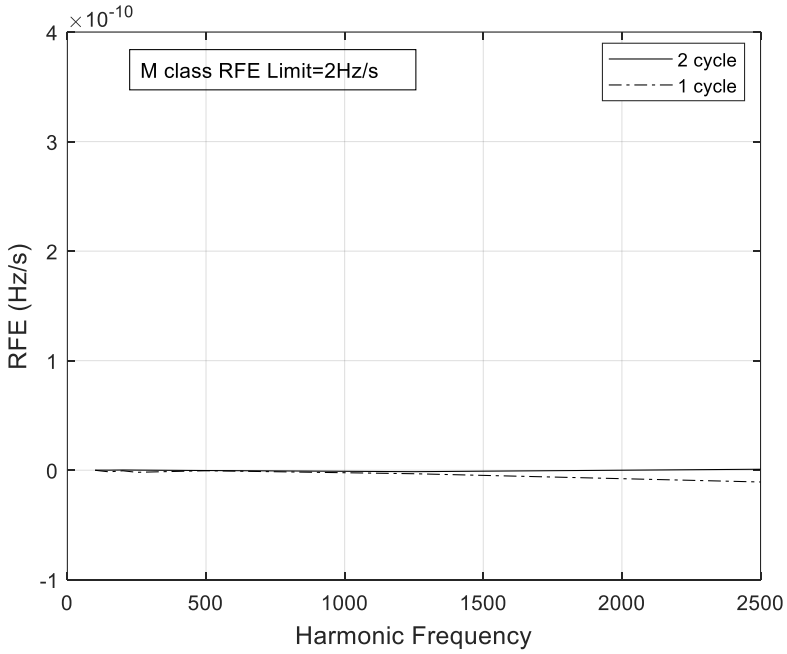

Figure 4.13: RFE results of one-cycle and two-cycle TFT for 10% harmonic interference signal

4.10 Out-of-band interference test

The Out-of-band interference (also referred to as interharmonic) test applies only to M class measurements and is not defined in the standard for P class measurements. The input reference test signal utilised for this test is composed of a 50 Hz continuous

waveform and an added interharmonic component at 10% of the magnitude of the fundamental. The interharmonic components are selected at distinct frequency values, ranging from 10 Hz to 100 Hz, and increasing in intervals of 10 Hz. The synthesized test signal is processed by the TFT phasor estimation algorithm and the resulting measurement performances are presented in Figure 4.14, 4.15 and 4.16. The TVE results of the one-cycle and two-cycle TFT algorithms are displayed in Figure 4.14, and are shown to not be within the compliance criteria as they are above the TVE limit of 1.3% for all interharmonic frequencies.

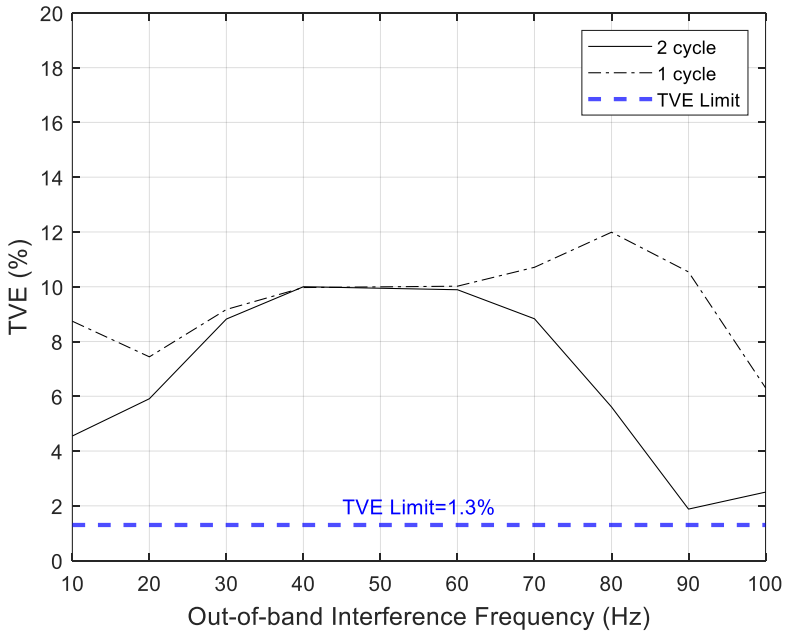

Figure 4.14: TVE results of one-cycle and two-cycle TFT for 10% Out-of-band interference test

The FE and RFE values of the one-cycle and two-cycle TFT algorithms are illustrated in Figure 4.15 and Figure 4.16 respectively. The values are shown to exceed the maximum FE and RFE limits and therefore do not meet the compliance requirements.

The results suggest that the out-of-band or interharmonic interference corrupts the fundamental sinusoidal signal to the point where inaccuracies in the measurement start to develop. The phasor estimation algorithm struggles to reject the interharmonic signals and this may require a low pass or anti-aliasing filter at the input stage in order to remove unwanted interharmonic components.

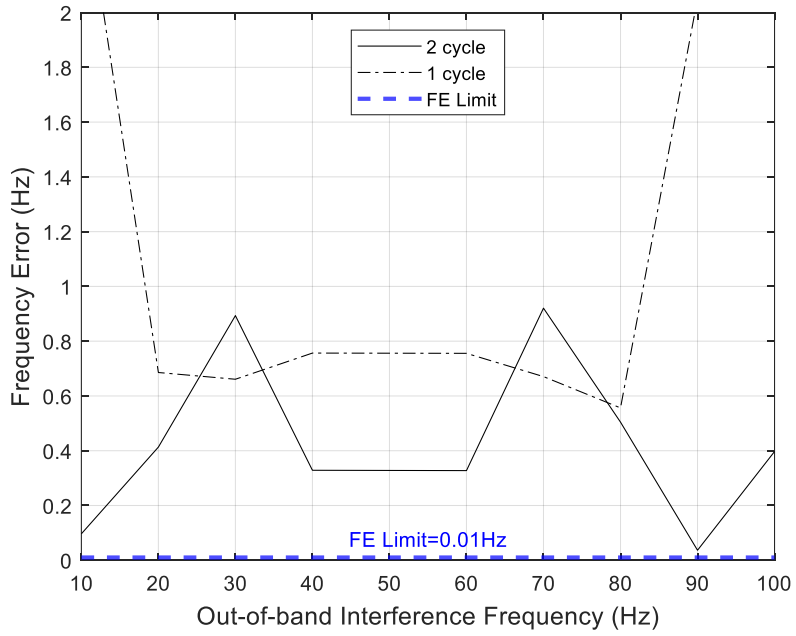

Figure 4.15: FE results of one-cycle and two-cycle TFT for 10% interharmonic signal

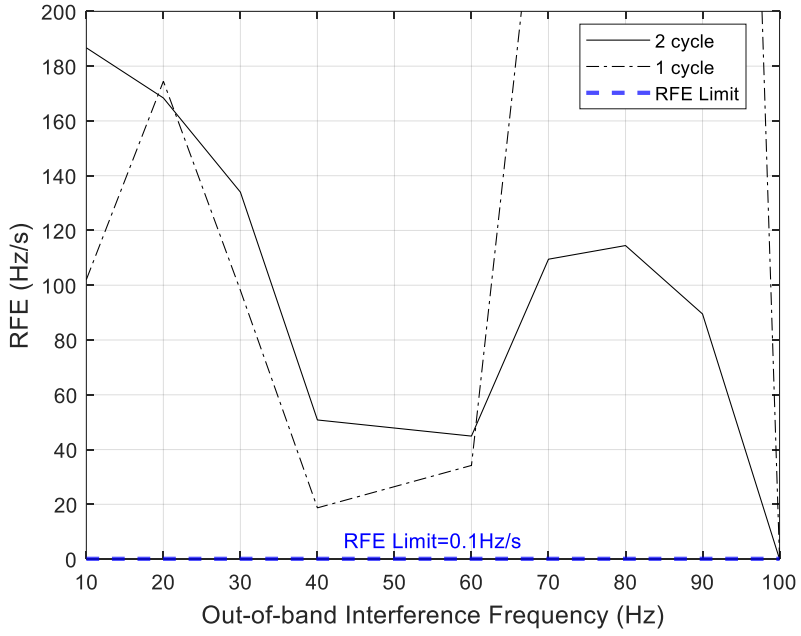

Figure 4.16: RFE results of one-cycle and two-cycle TFT for 10% interharmonic signal

4.11 Modulation test

The first of the dynamic tests to be conducted is the modulation test (also known as measurement bandwidth test). This test applies a modulating amplitude and a modulating phase angle to the reference input test signal to be estimated. The

modulating signal frequency ranges from 0.1 Hz to 1 Hz for P class, and 0.1 Hz to 2 Hz for M class. During this test, the modulating frequency is incremented in steps of 0.1 Hz, and the modulating signal is applied at 10% of the fundamental amplitude. This test mimics amplitude and phase variations in the input signal, which can occur during a power system fault condition. This test is evaluated for phase angle modulation, as well as amplitude modulation and phase angle modulation combined.

The performance results of the phasor estimation produced by the one-cycle and two-cycle TFT algorithm for amplitude and phase modulation are shown in Figure 4.17. The performance results for phase modulations only, are shown in Figure 4.18. The TVE results in both Figure 4.17 and Figure 4.18 are shown to be well below the compliance limit of 3% for P and M class.

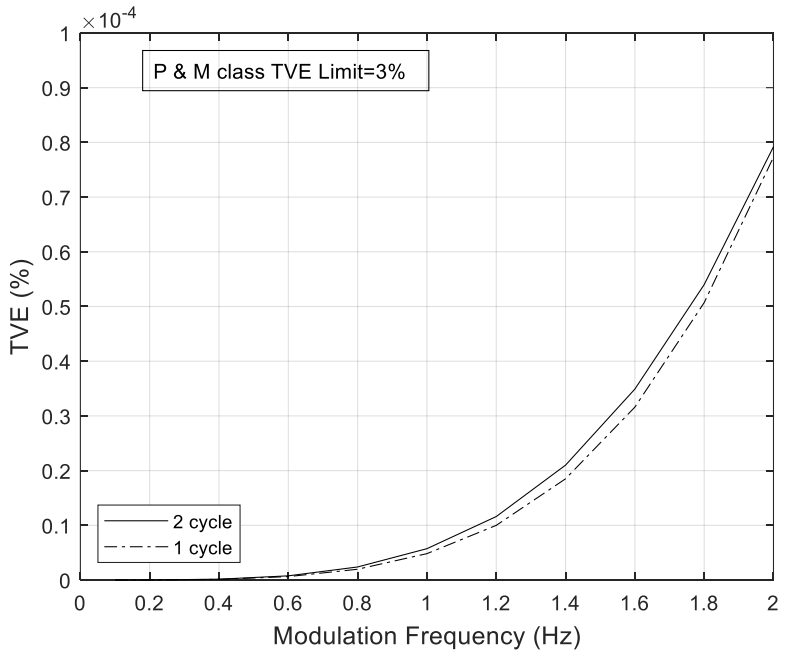

Figure 4.17: TVE results of one-cycle and two-cycle TFT for amplitude and phase modulation of 10%

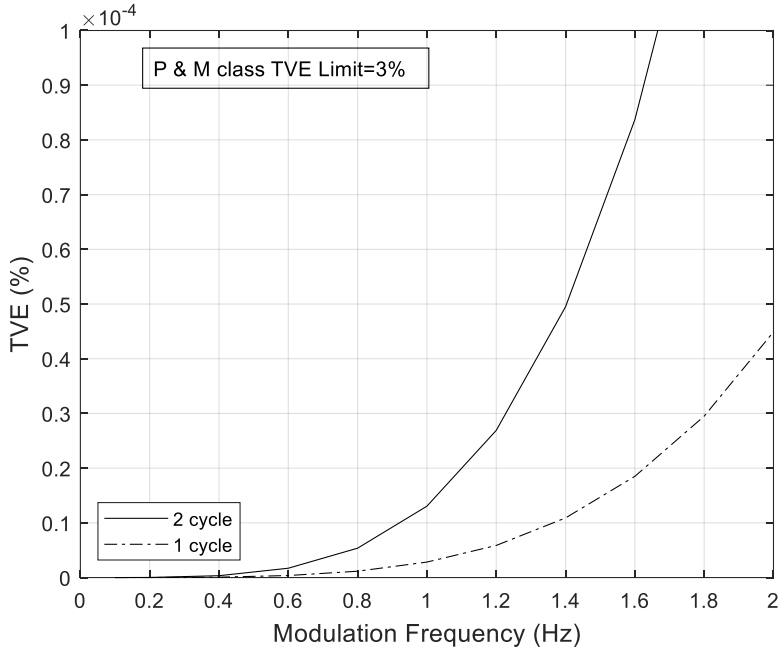

Figure 4.18: TVE results of one-cycle and two-cycle TFT for phase modulation of 10%

The FE values are recorded in Figure 4.19 and 4.20 for single and combined modulation conditions respectively. It can be seen here that the P class and M class error limits of 0.01 Hz and 0.06 Hz are easily met by both one-cycle and two-cycle TFT versions.

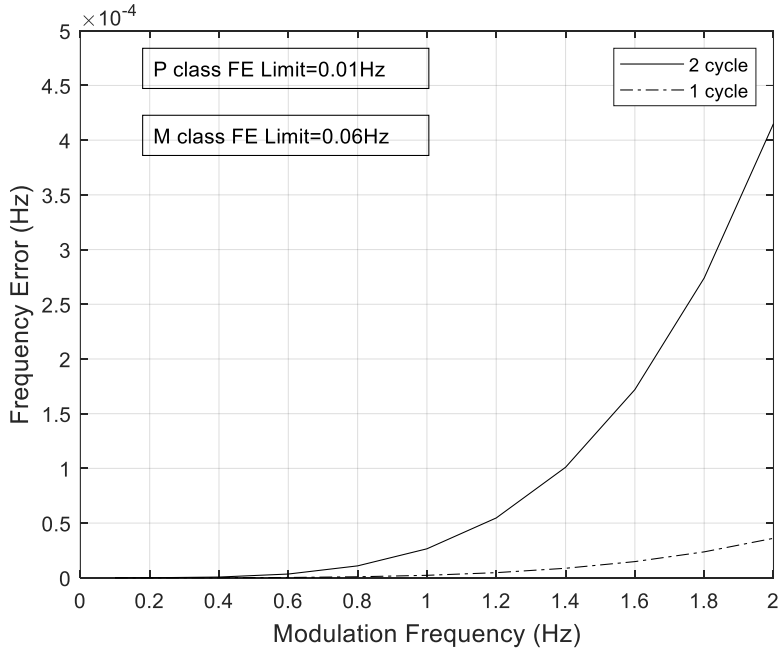

Figure 4.19: FE results of one-cycle and two-cycle TFT for amplitude and phase modulation of 10%

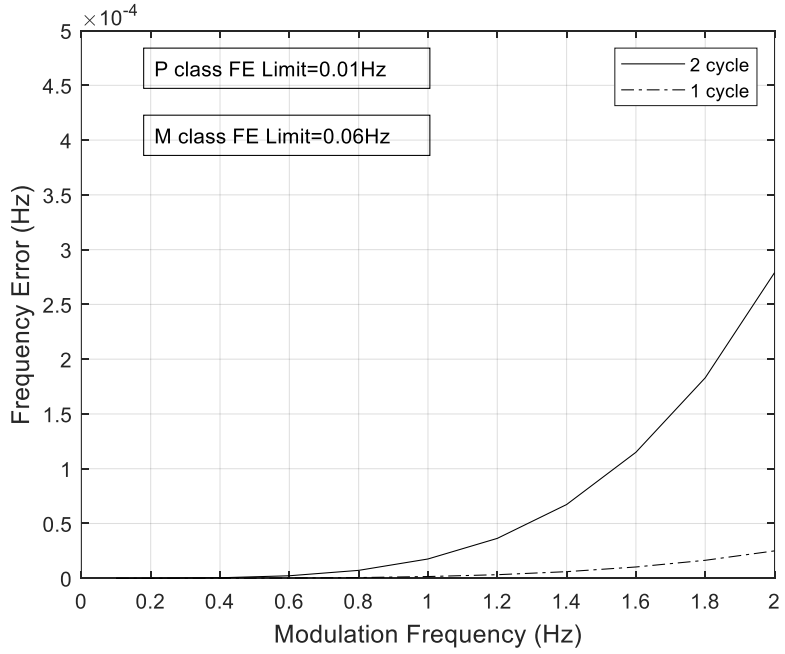

Figure 4.20: FE results of one-cycle and two-cycle TFT for phase modulation of 10%

The test results displaying the RFE values for amplitude and phase modulation is shown in Figure 4.21. The RFE values produced for independent phase modulation only, is shown in Figure 4.22. The RFE values are shown to be well below the 0.2 Hz/s and 2 Hz/s P class and M class limit, and hereby successfully meets the compliance standard.

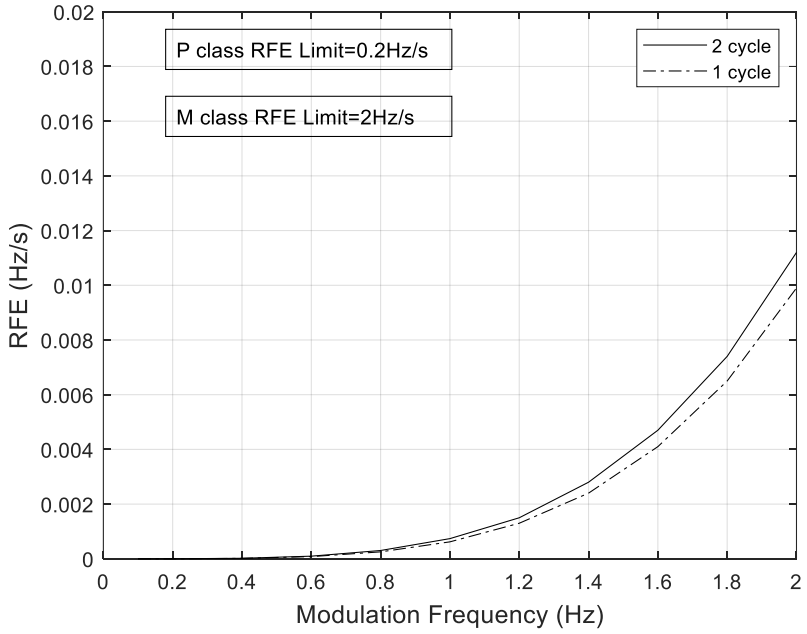

Figure 4.21: RFE results of one-cycle and two-cycle TFT for amplitude and phase modulation of 10%

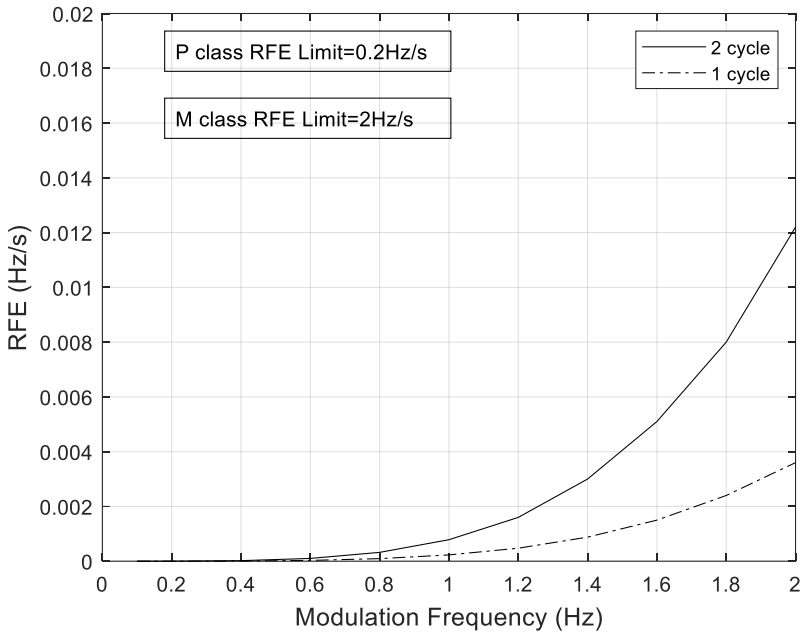

Figure 4.22: RFE results of one-cycle and two-cycle TFT for phase modulation of 10%

4.12 Frequency ramp test

In the simulation of the frequency ramp test, the frequency of the reference test signal is ramped linearly from 48 Hz to 52 Hz, at a rate of 1 Hz/s. The reference equation of the test signal defined in Equation (4.8) applies a continuous variation in phase which results in a linear frequency ramp. During this test the signal amplitude remains constant.

The performance results of the frequency ramp test produced by the one-cycle and two-cycle TFT algorithms are shown in Figure 4.23, 4.24 and 4.25. In Figure 4.23, we observe the TVE results are well below the TVE limit of 1%, this being the requirement for P class and M class measurements. It should be noted that the TVE values of the one-cycle and two-cycle TFT algorithms differ by a small margin, with the latter offering a slightly better performance with an overall maximum TVE value of 0.01% over the given frequency range.

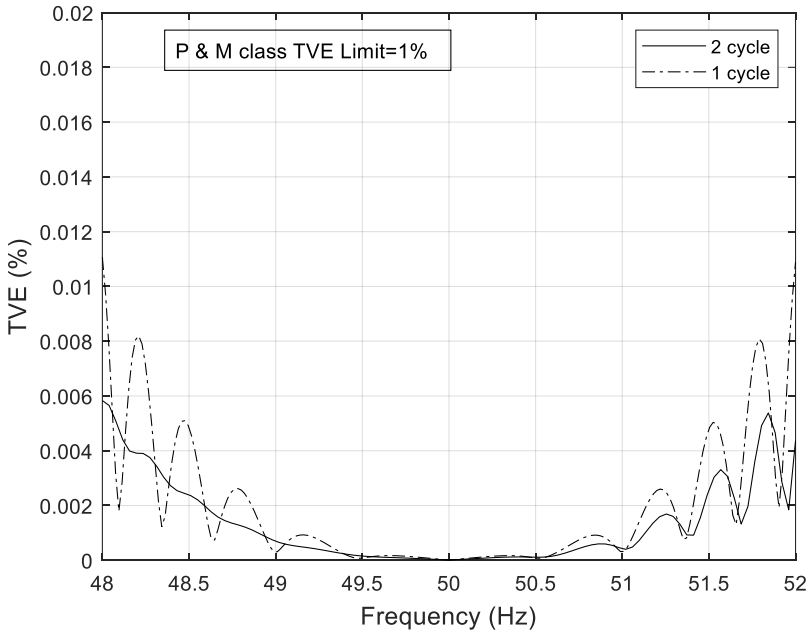

Figure 4.23: TVE results of one cycle and two cycle TFT for frequency ramp test

Figure 4.24 show the FE results of the one-cycle and two-cycle TFT versions. The one-cycle TFT offers a more accurate frequency estimation when deviating from the fundamental frequency, with the resulting frequency errors falling within the respective P class and M class error limits of 0.005 Hz and 0.01 Hz. The two-cycle TFT struggles to meet the FE requirements across the entire frequency ramp range, and is only able to produce admissible frequency error margins across a narrower band of frequencies for P class, which spans from 48.6 Hz to 51.5 Hz.

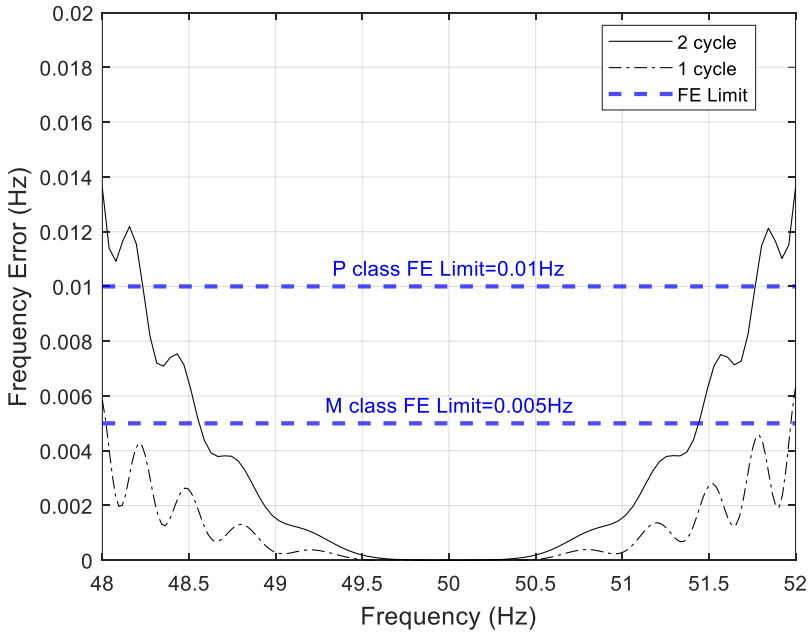

Figure 4.24: FE results of one-cycle and two-cycle TFT for frequency ramp test

The simulation test results of the RFE values are shown in Figure 4.25, and indicate both one-cycle and two-cycle TFT versions only satisfy compliance requirements for a narrow band of frequencies within 1 Hz of the fundamental frequency. We observe the frequency measurements outside this 1 Hz frequency band all exceed the P and M class RFE limit of 0.01Hz/s.

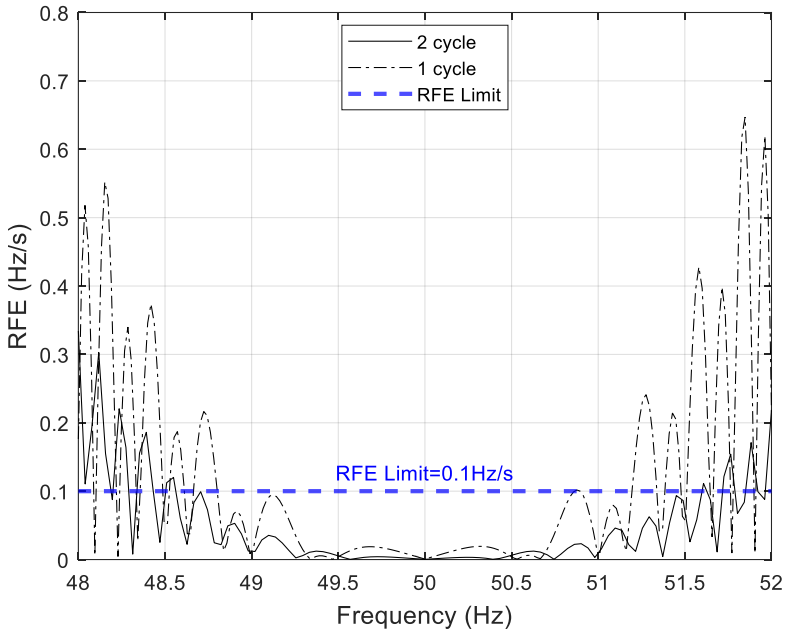

Figure 4.25: RFE results of one-cycle and two-cycle TFT for frequency ramp test

4.13 Conclusion

In this chapter, the performance and accuracy of the TFT algorithm is evaluated through a series of tests simulated in a MATLAB environment. A functional test is conducted via simulation and successfully verifies the basic operation of the TFT algorithm. Thereafter a compliance test is performed, comprised of a selection of tests obtained from the IEEE C37.118.1-2011 standard. The compliance test evaluates and determines the measurement performance and accuracy of the one-cycle and two-cycle second-order TFT algorithm in accordance with the IEEE C37.118.1-2011 standard. Steady-state and dynamic test conditions are included, which represent common conditions and scenarios occurring in power systems. The estimated phasor parameters and frequencies are recorded and the corresponding TVE, FE, and RFE values are calculated, providing a measure of the accuracy of the estimation, as well as determining whether compliance with the IEEE C37.118.1-2011 standard is met. These results also indicate the capability and suitability of the one-cycle and two-cycle TFT algorithm for performing P class and M class measurements. Similar measurement performances and accuracy between the one-cycle and two-cycle TFT algorithm were found, with the exception of the following observations. The two-cycle TFT algorithm has demonstrated a slightly better accuracy over the one-cycle TFT version with regards to phasor estimation, however the one-cycle TFT has shown to produce more accurate frequency estimation results, be it by a small margin. The latter

observation was especially noticeable at off-nominal frequencies as the signal frequency deviated farther from the fundamental. Most requirements and compliance limits stipulated by the IEEE C37.118.1-2011 standard are satisfied by the one-cycle and two-cycle TFT algorithm. The simulation tests has demonstrated an acceptable performance by the TFT algorithm as a phasor estimator, and has hereby validated its use in this project. The next chapter presents the deployment of the TFT algorithm on an embedded hardware platform for the development of a phasor measurement prototype for the real world environment.

CHAPTER FIVE
HARDWARE IMPLEMENTATION

5.1 Introduction

In Chapter Four, the estimated phasor parameters produced by the TFT algorithm have been successfully evaluated and verified through simulations in MATLAB in accordance with the IEEE C37.118.1-2011 standard. The next stage of this project involves implementing the TFT algorithms for deployment on a hardware platform for performing actual real life phasor measurements. The aim is to utilise the TFT algorithm on an actual phasor measurement prototype and verify and evaluate its operation and performance in a real world environment. In the simulation tests presented in Chapter Four, a second order one-cycle and second order two-cycle TFT algorithm are employed for performing the phasor estimation. For the development of the phasor measurement prototype, the intention is to also deploy and utilise these two algorithms on an embedded hardware platform. The prototype shall have the option of selecting which algorithm to execute based on whether a P-class or M-class measurement performance is required. One of the main objectives of the phasor measurement prototype is to perform phasor estimations in compliance with the IEEE C37.118.1-2011 standard. The most important factors that contribute to the success and realisation of this goal is accuracy and processing speed. Accuracy is primarily dependant on the efficacy of the employed phasor estimation algorithm, while the speed is largely reliant on the hardware platform used for computing the experimental phasor estimations. In order to effectively implement the phasor measurement on an embedded hardware platform, there are several basic hardware components that are essential for the successful operation of the device. This consists of a processing unit, data acquisition system and a communication interface.

In this chapter, we discuss the requirements and selection of the embedded platform, and describe the development process followed for building the phasor measurement prototype. The development and implementation of the prototype includes a hardware and software component which both form an integral part of this design.

5.2 Processing platform selection

When selecting a processing platform for this research project, two important factors were considered, namely the type of processor for computing the Taylor–Fourier transform (TFT) algorithm with, and the data acquisition system for measuring and sampling the input signal. The initial decision was whether to use a Central Processing Unit (CPU) or a Graphics Processing Unit (GPU) for performing the data processing in

the phasor measurement. The parallel processing of data with GPU's were investigated, however it was discovered that this type processing was geared more towards applications with large datasets that require high throughput rates. The nature of our application in phasor measurement requires frequent sampling of the input data and transferring it to the processing stage. This movement of data to and from the GPU incurs numerous memory transfers which is an expensive operation and can lead to bottlenecks and an increase in overall latency. Furthermore, because our dataset is not very large, the parallel processing power of the GPU cannot be utilised to its full potential. The literature review in Chapter Two has also shown the use of FPGAs as the processing platform for PMUs in several projects. While this remains a good choice, FPGA development can prove to be costly and complex, requiring specialised expertise. CPUs on the other hand offer an effective solution for performing the computations and processing required by the phasor measurement prototype. The transmitting of phasor measurement data at the specified reporting rate as stipulated in the IEEE C37.118.1-2011 standard renders this task CPU-bound. It is essential the phasor estimation is computed within the required latency before the next report transmission is due. This type of operation is well-suited for CPUs, as it requires tasks to be executed sequentially and is ideal for low latency applications. One such type of embedded device is the microcontroller, which integrates an onboard CPU, RAM and additional peripherals for handling data acquisition and communication. They are also cost-effective and have a low power consumption, overall making them an excellent choice for this embedded real-time application. It should be noted that when employing a CPU-based device for processing the data, the selected reporting rate will significantly influence the processor speed, as phasor measurements are required to be transmitted more frequently, ultimately defining the necessary processing capabilities of the system.

5.3 Selection of STM32 Nucleo-F767ZI development board

The STM32 Nucleo-F767ZI development board was selected as the dedicated hardware platform for implementing the TFT algorithm and developing the phasor measurement prototype. The Nucleo-F767ZI development board is intended for high performance applications and includes the required processing power, necessary features and onboard peripherals to develop the phasor estimation and measurement prototype. The development boards ease of use, rapid prototyping capabilities, and affordability are also significant factors which has resulted in its selection as the hardware platform for this project. A top view of the STM32 Nucleo-F767ZI development board is shown in Figure 5.1. In the next section we introduce some

important features of the STM32 Nucleo-F767ZI which makes it the ideal development platform for this project and application.

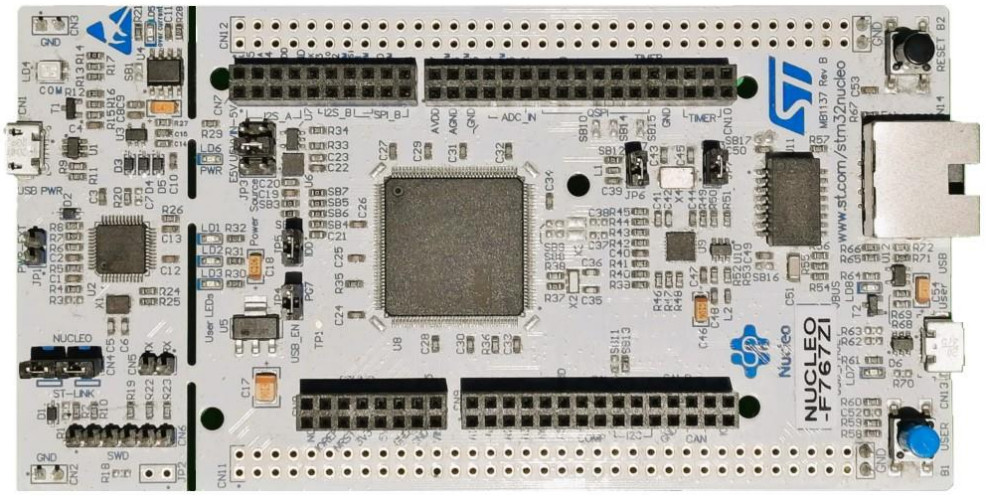

Figure 5.1: The STM32 Nucleo-F767ZI development board

5.4 Overview and specification of STM32 Nucleo-F767ZI

The following specification information presented in this section has been sourced from the STM32F767xx microcontroller datasheet (STMicroelectronics, 2021). The STM32 Nucleo development boards are a family of affordable development boards produced by STMicroelectronics and are intended as a rapid prototyping platform for industrial electronic, automation and embedded applications. The STM32 Nucleo development boards hosts a range of 32-bit ARM-based microcontrollers featuring a variety of onboard peripherals, such as General Purpose Input/Output (GPIO), Universal Synchronous/Asynchronous Receiver-Transmitter (USART), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), Analog-to-Digital Converter (ADC), Digital-to-Analog Converter (DAC), Timers and Direct Memory Access (DMA), which are all essential for embedded applications. All STM32 Nucleo boards are equipped with an onboard ST-LINK/V2-1 programmer and debugger for facilitating development and debugging, thus eliminating the need for additional development kit. A free version of the STM32CubeIDE is provided by STMicroelectronics as a software development tool for compiling, programming and debugging source code. This includes a comprehensive list of free software libraries and example code. The following highlights important hardware specifications of the Nucleo-F767ZI development board required for the development of the hardware prototype.

5.4.1 Processor and memory

The development board is driven by a STM32F767 32-bit microcontroller which is based on the ARM Cortex-M7 processor core. This processor can operate at a

maximum frequency of 216 MHz, hereby achieving a performance of 2.14 DMIPS. This amount of computation power is well-suited for performing intensive computations found in the TFT algorithm. The Cortex-M7 core features a hardware floating point unit (FPU), for speeding up floating-point arithmetic operations using 32-bit single-precision and 64-bit double-precision data types. The microcontroller has 2 Mbytes (MB) of flash program memory for storing developed user code, as well as 512 Kbytes of SRAM for data storage.

5.4.2 Onboard analog-to-digital converter

The STM32F767 microcontroller includes three onboard 12-bit successive approximation analog-to-digital converters (ADC) with a maximum sampling rate of 2.4 mega samples per second (MSPS), and an input voltage range of 0V to 3.3V. To accelerate the data acquisition stage further, a Direct Memory Access (DMA) controller is used in conjunction with the ADC. The DMA peripheral is utilised to speed up operations through direct data transfers of sampled data from the ADC to memory. The DMA controller also operates independently of the processor, thus freeing up valuable CPU resources and usage for executing other tasks.

5.4.3 Communication interface

The Nucleo-F767ZI development board is integrated with an Ethernet interface and UART serial communications interface via virtual (USB) serial port. These interfaces enable connectivity and communication to external devices. The virtual serial port allows the STM32 Nucleo board to communicate with a computer over a USB interface, but retains the functionality of a serial connection. For this project, the UART peripheral was selected as the dedicated communication interface on the Nucleo board, for providing a readout of the estimated phasor parameters.

5.4.4 Software development

The STM32 family of 32-bit ARM-based microcontrollers supports a range of programming languages, however C/C++ is the most popular and widely used among embedded developers and enthusiasts. As a result of this, most STM32 software libraries, example code and development tools all support C/C++ development. C/C++ has also been found to be the popular choice among the STM32 community which provides useful resources, tutorials and valuable knowledge relating to the microcontroller. These factors all contribute to making C/C++ the clear programming language of choice for development on the STM32 microcontroller.

5.5 Functional description of system

In this section, the overall design envisaged for the phasor measurement prototype is discussed. The functional block diagram in Figure 5.2 illustrates the basic design of the phasor estimation prototype to be implemented on the STM32 Nucleo board. The diagram highlights four key elements in the design of the phasor measurement prototype. A brief description of each stage is discussed next.

The first stage in the design is the selection of the input source and is responsible for configuring the embedded device based on the type of input signal applied. The input signal options consists of a continuous analog signal generated by a signal generator, and a discrete test signal in the form of a test vector embedded in the source code. These represent the actual and simulated input test signals respectively. Furthermore, either one cycle or two cycles of the input signal are sampled, depending on the TFT algorithm selected. This is followed by the data acquisition stage, which carries out the conversion of the analog input signal and produces a set of discrete sampled values. It should be noted that this section applies to continuous analog input signals only. When using the discrete test signal as an input to the system, the data acquisition section is bypassed as the input is already in a discrete format. The discrete test signal or sampled values produced by the ADC are subsequently transferred to the phasor estimation stage, where the TFT algorithm computes the estimated phasor parameters. The output phasor parameters include the amplitude, phase angle, frequency and ROCOF. There are two algorithms available to choose from for computing the phasor estimation, namely the one-cycle TFT and two-cycle TFT algorithm. This selection is based on the performance class required (ie. P class or M class measurements). Finally, the estimated phasor parameters are stored in memory and read out to the user via a serial console.

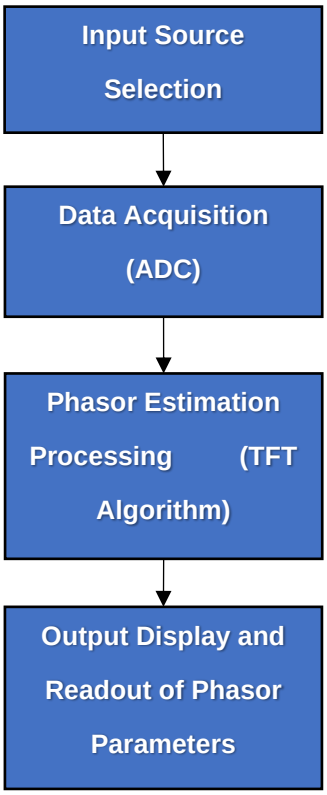

Figure 5.2: Functional block of phasor measurement design/system

5.6 Embedded software development

The embedded software development plays in integral role in allowing the STM32 hardware platform to function and operate as a phasor estimation device as described in the functional overview. The following provides a brief overview of how the functionality of all four major components of the phasor estimation prototype indicated in Figure 5.2 are implemented and realised on the STM32 hardware platform through developed software. All embedded software development is carried out in the STM32CubeIDE.

The firmware developed for carrying out the phasor estimation onboard the STM32 microcontroller are composed of several source code files written in the C programming language. The main.c file contains the initialisation of the microcontroller and its peripherals, as well as the primary control and logic of the program. The phasor_estimation.c file is the actual TFT algorithm in the form of a C function, generated and exported from the MATLAB code. The porting process of this function is described in section 5.9.1. This function is called from the main C function for computing the phasor parameters. To facilitate and simplify development, the STM32CubeIDE provides low-level peripheral drivers and a Hardware Abstraction Layer (HAL) for accessing and controlling hardware peripherals at a high level of

abstraction. These APIs are used to generate initialisation code for the microcontroller when a project is created in the STM32CubeIDE.

For this project, the approach of employing two TFT algorithms for performing the phasor estimation computation was adopted. This consisted of the one-cycle TFT and the two-cycle TFT version. Although these two variants of the TFT algorithm are fundamentally alike, each version is generated independently in MATLAB and Simulink. This is due to the amount of input samples required by each TFT computation, which has a direct impact on the size of the matrices and vectors utilised in the mathematical computation. When porting the MATLAB and Simulink code to C, these sizes are accounted for by being hardcoded and assigned up front as they are responsible for setting the size of the vectors and matrices. The final outcome are two separately generated c files for deployment on the STM32 microcontroller, one for the one-cycle TFT and another for the two-cycle TFT computation.

5.7 Design considerations

There are a few design aspects to consider when developing the embedded software for the STM32 microcontroller to perform phasor estimation. The following delves into the software implementation of the phasor estimation prototype and examines some key aspects and considerations in the software which are pertinent to the design.

The design utilises several essential hardware peripherals integrated onboard the STM32F767 microcontroller. These include General Purpose Input/Output (GPIO), Timers, Analog-to-Digital Converter (ADC), Direct Memory Access (DMA), and the Universal Asynchronous Receiver-Transmitter (UART). Each peripheral is required to perform a specific task for the phasor estimation prototype. The ADC is responsible for capturing the input analog signal and transfers the ADC output directly to a memory buffer via DMA for further data manipulation. The UART provides a readout of the estimated phasor parameters by printing the results to serial console. The timer ensures the ADC measurement is started every 500 mS. For debugging purposes the GPIO is used to mark specific events or indicate certain conditions. The microcontroller's hardware peripherals are configured via a graphical interface provided by the STM32CubeIDE. From this, the source code for initialising the hardware peripheral is generated accordingly.

5.8 Data acquisition

5.8.1 A/D Conversion and configuration

The first task to consider for the phasor measurement is the digitizing of the input signal using the onboard ADC. When selecting a suitable sampling frequency for our application, the fundamental frequency of the input signal must be known to ensure the Nyquist criterion is met. For this project, a nominal power system frequency of 50 Hz is assumed, and thus requires the sampling frequency to be at least twice this value ie. 100 Hz. For the MATLAB simulations presented in Chapter Four, a sampling frequency of 800 Hz was employed, however the sampling frequency of the STM32's onboard ADC was configured at a frequency of 32 kHz. Power grids often present dynamic changes which can manifest as frequency fluctuations, modulated signal components, or harmonic or interharmonic interference. These conditions contain frequency components that are higher than the 50 Hz nominal power system frequency and therefore require a higher sampling frequency in order to be detected. With a sampling frequency of 32 kHz, a maximum frequency component of 16 kHz is capable of being observed. The 32 kHz high sampling frequency also produces a better time resolution which can lead to more accurate phase angle estimations. A higher sampling frequency does however result in an increased amount of data to be processed which in turn requires more powerful hardware to handle the additional load. Conversely, lower sampling frequencies reduce the amount of data and computational demands, but this may compromise the precision and fidelity of the phasor measurements, especially during rapid signal variations. The chosen sampling frequency of 32 kHz produces 640 samples per cycle for an expected input signal of 50 Hz. The ADC output sampled data is transferred directly to a memory buffer by means of a Direct Memory Access (DMA) peripheral-to-memory transfer, on completion of each ADC conversion. The onboard DMA controller allows for high speed transfers of sampled ADC data that runs independently of the CPU, thus conserving CPU usage and processing time. The DMA continues to transfer ADC data until the memory buffer is full, at which point the ADC complete callback is raised, initiating the pre-processing and formatting of the data. The size of this buffer is dependent on the TFT version selected, as each method requires a specific amount of samples from the ADC to process. The STM32F767 on-chip ADC is operated in single conversion mode for a single input channel. In this mode, the start of each conversion is triggered and synchronized every 500 ms by an on-chip internal timer. This requires the ADC conversion as well as the TFT processing to complete in under 500 ms in order to be ready to proceed with the next ADC measurement.

5.8.2 Pre-processing of data

On completion of the ADC conversion, the raw ADC samples are stored in a buffer and pre-processed into a predefined format required by the TFT algorithm. The size of the memory buffer for storing the converted ADC samples for pre-processing is based on the number of samples in the Taylor observation interval and is determined by Equation (5.1),

$$N = (T/T_1)N_1 \tag{5.1}$$

where N is the number of samples within a given Taylor observation interval T, and N_1 is the number of samples within one cycle (T_1) of the input signal. N is rounded up to the closest odd integer to ensure that the value of the TFT window length is always odd, such that one sample is always located at the centre of the observation interval (de la O Serna, 2007). The sampling frequency determines the number of samples per cycle for a given input signal. As the sampling frequency of the ADC is run at 32kHz, the number of samples per cycle produced for a 50 Hz input signal amounts to a value of N_1 = 640 samples per cycle. When rounding up to the closest odd number, a Taylor window length (N) of 641 samples is obtained for the one-cycle TFT, and 1281 samples for the two-cycle TFT algorithm. The above description is illustrated in the following calculation for the two-cycle TFT algorithm.

$$N = \left(\frac{40\ ms}{20\ ms}\right) \times 640 \text{ samples}$$

$$N = 1280 \text{ (Rounded up to } 1281)$$

Each discrete sample produced by the ADC is represented in a 12-bit unsigned integer format, allowing for a maximum of 4096 discrete levels to be represented by this value. During this pre-processing stage, this value is subsequently converted into a floating point number representing the instantaneous voltage of the input signal measured by the ADC. With an ADC reference voltage of +3.3V, the smallest voltage that can be measured by the ADC is 805 µV. The conversion of the ADC digital value to the corresponding voltage level is defined by the formula indicated in Equation (5.2).

$$V_{in} = \frac{ADC\ Value\ \times\ V_{ref}}{2^{N-1}} - DC\ Offset \tag{5.2}$$

Where V_{in} is the voltage of the input signal, V_{ref} is the ADC reference voltage of 3.3V, N is the 12-bit ADC resolution, and DC Offset is an applied DC bias in volts. As the onboard ADC reference voltage ranges from 0V to +3.3V, the input signal must be applied with a DC offset in order to capture both positive and negative half cycle values of the input signal. This applied DC bias is removed digitally as indicated in Equation

(5.2) to obtain the measured input signal voltage centred around zero. This is the required format of the input to the TFT algorithm for computing and extracting the estimated phasor parameters of the input signal.

In DSP applications, it is common practice to apply a tapered window to the sampled input signals for improved accuracy and performance, however this was omitted from this implementation to conserve resources and minimize latency. In essence, the exclusion of a tapered window can be likened to applying a rectangular window to the digitized input signal.

5.9 TFT algorithm

5.9.1 Porting of MATLAB/Simulink code to C

The initial implementation of the one-cycle and two-cycle TFT algorithm was written in MATLAB, and evaluated through simulations as discussed in Chapter Four. In order to deploy and utilise the TFT algorithm on the STM32 hardware platform for performing the phasor estimation, the MATLAB code needs to be ported to a C function. This function is to be invoked by the main C program when required to extract the phasor parameters of the input signal. The code export is executed using Embedded Coder, a tool offered with the MATLAB/Simulink software suite for generating optimised C/C++ code for deployment on embedded devices.

The TFT algorithm written in MATLAB employs the default 64-bit double-precision data types for computing the phasor estimation. When ported to C for deployment on the STM32 microcontroller, the double data types are retained for maintaining the precision, accuracy, and wider range of values associated with the double-precision format. As the STM32 microcontroller features a double-precision FPU and 512 Kbytes of SRAM, floating point calculations can be executed successfully and timeously.

5.9.2 Code porting procedure

There are several ways of exporting MATLAB code to C, however for this project the algorithm export workflow was followed. This process is demonstrated in a MathWorks video and webinar series (MathWorks, 2024). This process uses the MATLAB and Simulink Embedded coder app to generate customizable standalone C code which can be added to larger STM32 software projects and compiled collectively with a standard toolchain such as STM32CubeIDE.

This workflow requires the MATLAB implementation of the TFT algorithm to firstly be converted into a Simulink model prior to code generation. The Simulink model is created by converting the original MATLAB TFT algorithm function into a MATLAB

function block named phasor_estimation_method4, and adding it to the Simulink design. Input and output ports are added to the model for passing data to and from the function block. In Figure 5.3, a screenshot of the Simulink model shows the input and output ports connected to the phasor_estimation_method4 function block. The three Simulink inports on the lefthand side of the design are assigned as inputs to the TFT algorithm, and are used for passing the digitised signal of interest as an input (inport3), as well as specifying the input signal frequency (inport2), and the associated number of samples per cycle (inport1). The estimated phasor parameters are computed and provided at the outports as shown on the righthand side of the design. The outputs include the estimated amplitude (outport2), phase angle (outport3), and corresponding first and second derivatives of the amplitude (outport4 and outport6) and phase angle (outport5 and outport7).

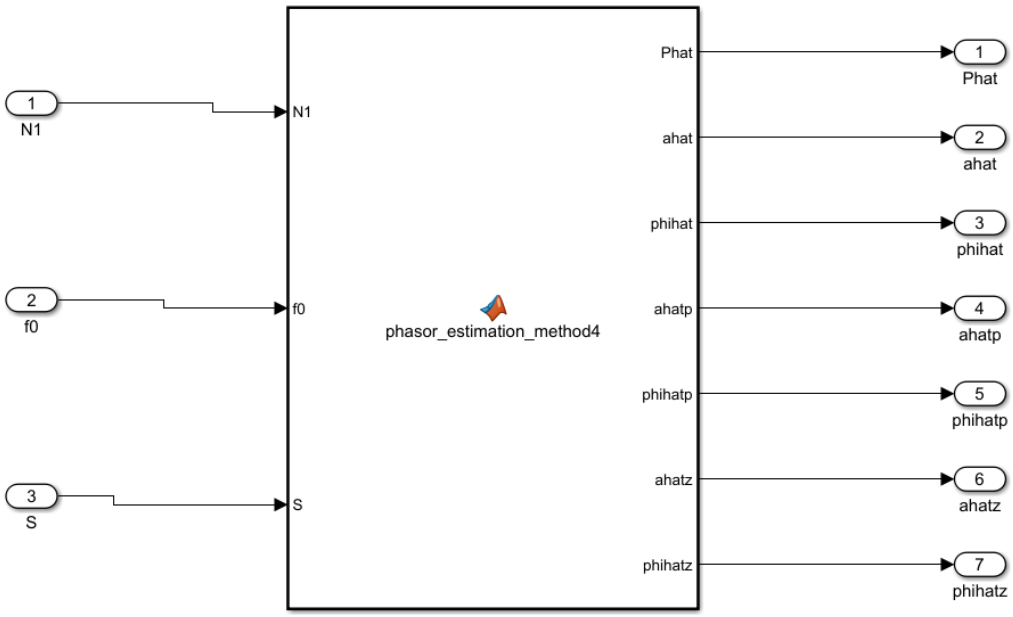

Figure 5.3: Simulink design of TFT algorithm

The embedded coder app provides a quick start wizard for setting up the configuration for embedded code generation and there after commences with the code export. The embedded coder generates a phasor_estimation.c file and associated header files. Two callable C functions namely, phasor_estimation_initialize() and phasor_estimation_step(), are contained in this generated file and are responsible for initialisation and execution of the TFT phasor estimation. The generated source code

files are transferred collectively to the STMCubeIDE where the algorithm is integrated with the main program and built for deployment on the STM32 Nucleo board.

5.9.3 Description of source code

Once the TFT algorithm is successfully ported from MATLAB/Simulink to C, the generated source code files can be utilised as a callable function for performing the phasor estimation.

The following is a description of the source code implementing the phasor estimation written in the C programming language for deployment on the STM32 Nucleo-F767ZI development board. The flowchart in Figure 5.4 is a graphical representation of the developed source code and depicts the overall process and logical flow of the embedded program. The main C function initiates the start of the program's execution and serves as the core of the embedded program where all the primary tasks are structured around. In the flowchart in Figure 5.4, the main C function is the entry point to the program’s execution and begins with the initialisation of the STM32 microcontroller’s clock and peripheral configuration. This is followed by the declaration and assignment of user-defined variables, as well as buffer lengths for specifying the number of input samples required for the design. As mentioned in the pre-processing section, the buffer lengths are determined by the number of samples in the Taylor observation interval, and are specified as either 641 samples for the one-cycle TFT or 1281 samples for the two-cycle TFT.

The primary execution of the program takes place within an infinite while loop. This is common practice in embedded systems and allows for the microcontroller to continuously execute a set of instructions and handle interrupt events. One of the primary tasks contained within the infinite while loop is the readout of the digitised input signal and estimated phasor values. The corresponding values are printed to a serial console from a memory buffer which continuously gets updated by the TFT phasor estimation function as the output values get processed. The readout continues to get executed until either an interrupt occurs, or the loop is terminated externally by a hardware reset or the removal of power.

Two interrupt sources and accompanying callback functions are incorporated in the design and are tasked with handling timing events and ADC-DMA functionality. The callback functions are designated as the HAL_TIM_PeriodElapsedCallback and the HAL_ADC_ConvCpltCallback, and are shown on the right side of the flowchart in Figure 5.4. The first interrupt is assigned to Timer 3 which is responsible for triggering the start of the ADC measurement every 500ms. The Timer 3 period and prescaler is

configured to rollover after a duration of 500ms and sets off an interrupt periodically. The interrupt is handled by the interrupt service routine facilitated by the HAL library, which subsequently calls the HAL_TIM_PeriodElapsedCallback function where in which the start of the ADC-DMA is initiated. The second interrupt is tasked with handling the ADC conversions and DMA transfers from ADC to memory. The ADC1 peripheral is used in conjunction with the DMA2 controller and raises an interrupt whenever the ADC buffer is full. The interrupt is managed by the HAL DMA interrupt handler which calls the HAL_ADC_ConvCpltCallback for execution. The source code in the callback function is customised and ensures the sampled input values are in the correct format for computation by the TFT algorithm. Finally the estimated phasor parameters are computed by the TFT algorithm and the results are stored in a memory buffer which are accessible by the main C function for readout to the serial console. An extract of the source code written in C is provided in Appendix B.

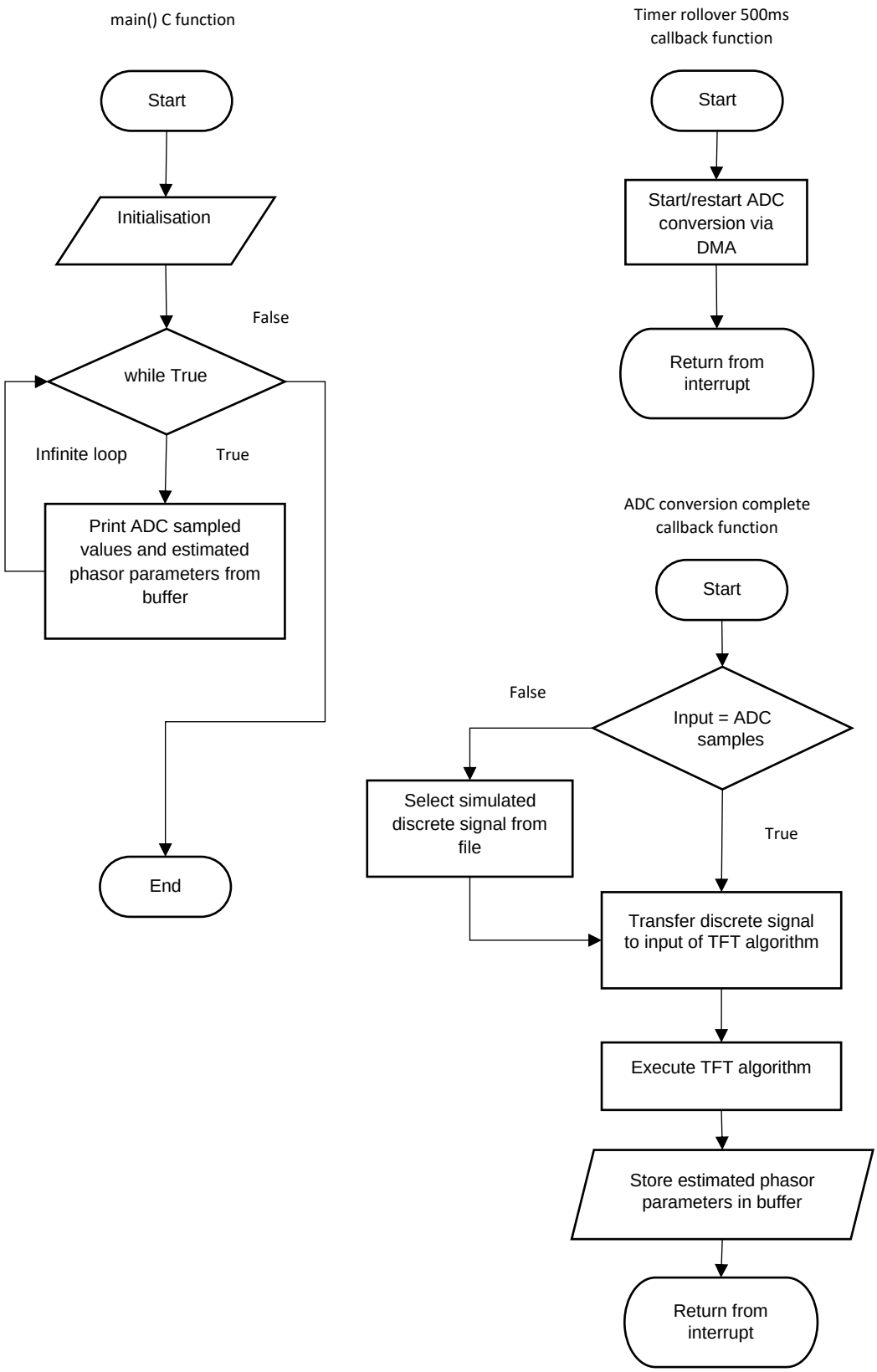

Figure 5.4: Flowchart of phasor estimation implemented in C

5.10 Estimated phasor parameter readout

5.10.1 Serial communication interface

In order to read out the estimated phasor parameters computed by the TFT algorithm, each corresponding output parameter is printed to the STM32CubeIDE command shell via the UART serial interface. This task is executed in the main C program and prints the estimated amplitude, phase angle, frequency and ROCOF values from their corresponding memory buffers using the virtual (USB) serial port interface. For testing and verification purposes, the discrete sampled input signal and input to the TFT algorithm is included and also printed to the serial console . This printing process is iterated repeatedly in the main C function, and displays the updated phasor parameters as they progress through each measurement. In Figure 5.5 a screenshot of the STM32CubeIDE shows the ADC sampled values and estimated phasor values printed to the serial console.

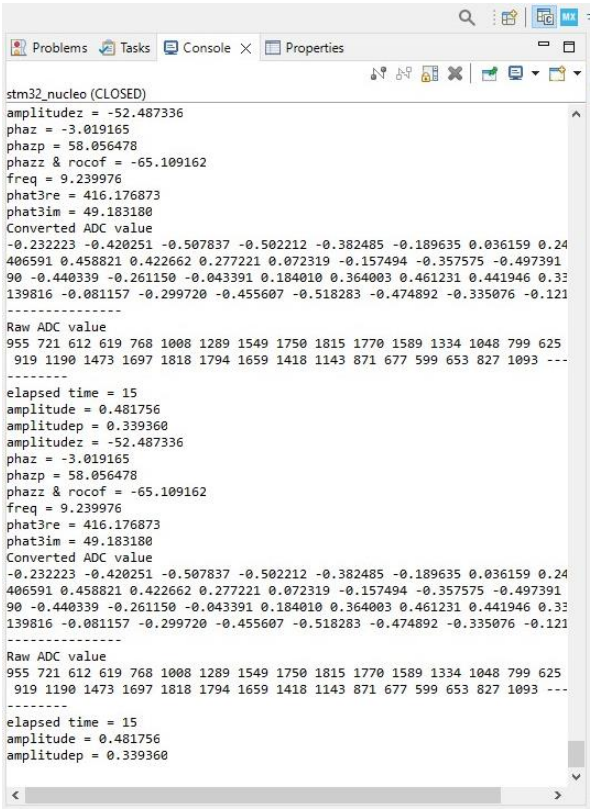

Figure 5.5: Screenshot of the estimated phasor values displayed in the serial console of the STM32Cube IDE

The configuration parameters specified for the serial communication connection between the STM32 device and the serial terminal are shown in Table 5.1.

Table 5.1: Serial communication interface configuration

Baud rate	115200
Data size	8
Parity	None
Stop bits	1

5.11 Other hardware configuration

5.11.1 System clock configuration

For the STM32 prototype, a phase-locked loop (PLL) clock was selected as the main system clock source (SYSCLK) for driving the microcontroller. The PLL clock is derived from the High Speed External (HSE) oscillator which is passed through a number of frequency multipliers in the PLL to finally produce a clock frequency of 192 MHz. The high clock frequency is required to ensure the phasor estimations are processed and completed within the reporting time interval. The STM32CubeIDE provides a graphical tool for configuring the clock of the STM32F767, and with this generates the associated source code for initialising the clocks. In Figure 5.6, the graphical tool provided by the STM32CubeIDE is shown for configuring the clocks of the STM32F767.

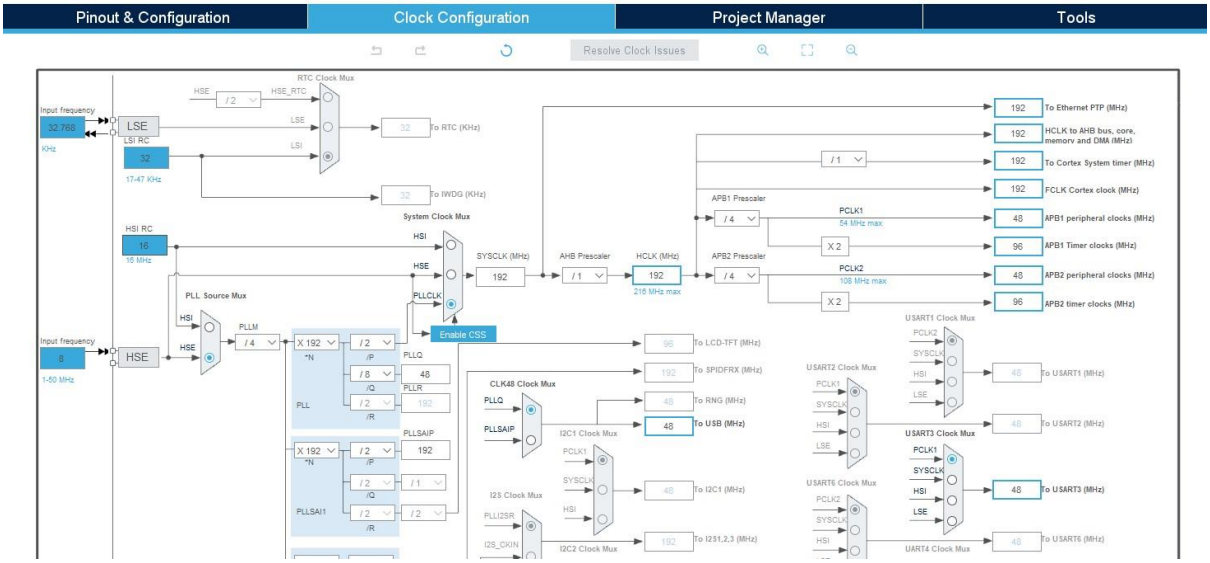

Figure 5.6: Screenshot of the clock configuration in the STM32CubeIDE

5.11.2 Floating Point Unit

The Nucleo-F767ZI development board is targeted at DSP and high performance applications and includes a floating point unit (FPU) integrated to the Cortex-M7 core of the STM32F767 microcontroller (STMicroelectronics, 2021). This hardware feature is advantageous and essential for this project as the TFT algorithm primarily consists

of floating point arithmetic operations for computing the output phasor parameters. These operations are computationally intensive and require a FPU in order to complete the phasor estimation within the given reporting rate. The hardware FPU facilitates and accelerates all floating point operations and supports both 32-bit single-precision and 64-bit double-precision data types (STMicroelectronics, 2016). Most floating point operations are able to be carried out by the hardware FPU in a single instruction cycle. Processors without a FPU utilise software libraries for performing floating point arithmetic operations, however this approach is inefficient, requiring several instruction cycles to execute and results in extensive delays in execution time. Additionally the use of floating point data types also allows for improved precision and a wider range of values to work with. The embedded FPU is enabled via the STM32CubeIDE, by selecting the appropriate configuration in the microcontroller and compiler settings.

5.12 Conclusion

This chapter concludes the development process and successful implementation of the TFT algorithm on an embedded hardware platform for performing the phasor estimation. In the design of the system, four key components are identified which are integral to the successful operation and implementation of the hardware platform as a phasor measurement device. By following this criteria, the STM32 Nucleo-F767ZI development board is selected as the dedicated hardware platform for implementing the phasor estimation for carrying out actual real-world phasor measurements. The hardware specification of the STM32F767 is carefully examined, identifying the key features and onboard peripherals required for developing the phasor estimation prototype. This device does present limitations in processing power and advanced hardware features, however the digital implementation can easily be ported to other hardware platforms to accommodate future enhancements. This chapter looks at the software implementation of the phasor measurement device, as the control logic, functionality and operation of the phasor measurement prototype is primarily implemented in software. The source code for the phasor estimation prototype is written in C and deployed to the STM32F767 microcontroller. The TFT algorithm employed in the MATLAB simulations is ported to C and utilised for performing the phasor estimation computation on the STM32 hardware platform. In this chapter, the design considerations required for the phasor estimation are presented , as well as a description of the developed source code. An input test source is applied at the input of the STM32 development board, and the corresponding phasor parameters are successfully produced at the output. In the next chapter, the output phasor parameters are read out and recorded for the evaluation of the hardware prototype.

CHAPTER SIX
MEASUREMENT RESULTS

6.1 Introduction

In this chapter , the implementation of the one-cycle and two-cycle TFT algorithm deployed on the STM32 hardware platform is tested and verified in the form of a phasor measurement prototype, whereby its phasor and frequency measurement performances are evaluated. This experimental verification and evaluation exercise is carried out by conducting a series of compliance tests obtained from the IEEE C37.118.1-2011 standard to determine the performance and accuracy of the estimation results produced by the phasor measurement prototype. The compliance tests incorporate both static and dynamic conditions. Since the prototype is not a fully-fledged PMU, only applicable tests obtained from the IEEE C37.118.1-2011 standard are carried out. The intent of these tests are to verify the implementation of the TFT algorithm deployed onboard the STM32 hardware platform and evaluate the accuracy of the resulting phasor and frequency estimations in contrast to the MATLAB simulations performed in Chapter Four. The accuracy of the STM32 phasor estimation results are evaluated according to the TVE, FE and RFE criteria, and are defined as per the class of performance. The results and performance of the one-cycle and two-cycle TFT algorithm are examined for each test to determine which is compliant and appropriate for P class and M class measurements respectively.

6.2 Test and evaluation procedure

The following compliance tests obtained from the IEEE C37.118.1-2011 standard have been selected for conducting the experimental verification and evaluation of the phasor measurements produced by the STM32 hardware platform:

1. Frequency variation test
2. Magnitude and phase angle variation test
3. Harmonic distortion test
4. Out-of-band interference test
5. Modulation test
6. Frequency ramp test
7. Measurement reporting latency test

These selected compliance tests consist of the same collection of tests utilised in the MATLAB simulations presented in Chapter Four. The MATLAB simulation results provides a mechanism for benchmarking and verifying the estimation outputs of the STM32 phasor measurement prototype. Each test requires a dedicated test signal

applied as an input to the test system. This provides a means to evaluate and verify the output estimation result against. The test signals are defined by mathematical formula and are provided in the IEEE C37.118.1-2011 standard. For the experimental testing and verification of the STM32 phasor measurement prototype, a continuous analog signal and discrete signal are utilised as the input references to the tests. This is discussed in greater detail in the following section 6.3 and 6.4. To remain consistent with the MATLAB simulations, the same reporting rate of F_s = 10 frames per second (fps) is retained. The reporting rate has an influence on the given test range conditions and also allows for dynamic conditions to be evaluated. The embedded software implementing the one-cycle and two-cycle TFT algorithm is deployed to the STM32 development board and its performance is evaluated through the compliance tests. Once the appropriate input test signal (as per the compliance test) is supplied to the input of the TFT algorithm in either analog or digital form, the estimated phasor and frequency parameters are processed and computed on the STM32 hardware platform. The computed phasor parameters consist of the estimated amplitude, phase angle, frequency, and rate of change of frequency (ROCOF) of the applied input signal. These estimated signal parameters are transmitted via the STM32 UART Serial Communication module and printed to a serial console provided by the STM32CubeIDE. The estimated phasor values are recorded for each compliance test and are used to calculate the TVE, FE and RFE values. These performance indicators are defined by Equations (3.13), (3.14), and (3.15) in Chapter Three, and expresses the accuracy and performance of measurement. The evaluation and analysis of the results are discussed in each compliance test section.

6.3 Continuous analog test signal

The continuous analog test signal has been specifically reserved for conducting the magnitude and phase angle variation test. As this test is considered for steady-state conditions, the specified test signal is simple and easy to produce. The signal is composed of a fundamental sinusoidal waveform with constant amplitude, phase angle and frequency. The RS Pro 2205A-20 PC-based oscilloscope with built-in function generator was utilised for generating the continuous analog test signal. The control of the oscilloscope and displaying of the measured waveforms are facilitated through the PicoScope 6 PC Oscilloscope software. Some important hardware specifications of the built-in function generator are as follows (RS Components, n.d.):

- Output Waveforms Types: Sine, Square, Triangle, Sinc, Arbitrary
- Standard Signal Frequency: DC to 100 kHz

- Output Frequency Resolution: 0.02 Hz
- Output Voltage Range: +/-2 V
- Output adjustments: Any amplitude and offset within ±2 V range

The sinusoidal analog signal is generated by the RS Pro 2205A-20 oscilloscopes built-in function generator and is physically connected to the input ports of the STM32 onboard analog-to-digital converter, where the sampling and quantization take place. Thereafter the digitised signal is transferred to the input of the TFT algorithm for computation of the amplitude, phase angle, frequency and ROCOF estimation.

6.3.1 Phase angle of continuous analog signal

The compliance tests require a sinusoidal test signal configured at a specific amplitude, phase angle, and frequency, in order to conduct, evaluate, and verify the phasor estimation. The RS Pro 2205A-20 built-in function generator only allows for the setting of amplitude and frequency values, however the phase angle cannot be set as simply as with a discrete signal. The phase angle is defined as the horizontal positioning or shifting of a waveform in relation to a cosine function, and this cannot be set in real time for a continuous analog signal. The phase angle of the continuous analog test signal is therefore taken as the angle in radians that occurs at the start of the ADC measurement, and is determined by measurement with an oscilloscope. This point effectively references the start of the continuous analog test signal that is captured and digitised by the ADC. This phase angle value is used as the input or expected phase angle value in the TVE calculation. In Figure 6.1, the phase difference of the test signal in relation to the 0 radian reference point of a cosine function is shown. The blue channel displays the input analog signal, and the red channel shows the rising edge as the ADC begins the conversion. For measuring the time delay between these two points, the oscilloscope rulers are placed at the start of the ADC conversion and at the cosine 0 radian reference point, and the time delay is obtained.

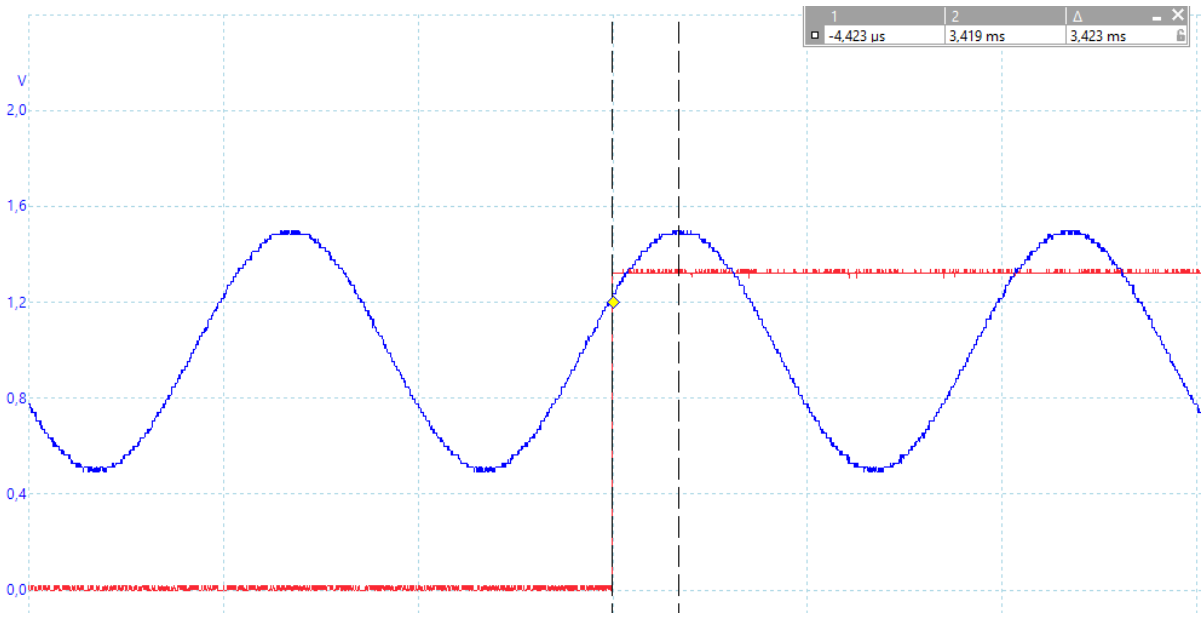

Figure 6.1: Measurement of time delay with oscilliscope for determining phase angle of analog test signal

The phase angle is obtained by measuring this time delay in seconds, and converting it to an angle in radians using Equation (6.1).

$$\varphi = 2\pi f \Delta t \tag{6.1}$$

where φ is the phase angle in radians, f is the signal frequency in Hertz, and Δt is the time shift or time delay in seconds.

The calculation of the TVE, FE and RFE performance indicators are effectively a process whereby the estimated signal parameter is compared against the corresponding input test signal parameter. For the dynamic compliance test, the input test signal parameters are constantly changing over time. This requires a signal generator with time-tagging capabilities in order to confirm and verify that the input signal parameters at a specific time matches the corresponding time-tagged output estimation values. For the purposes of this experimental verification and evaluation, the continuous analog test signal is only applied to the magnitude and phase angle variation test, as the signal parameters remain constant throughout an observation interval and therefore does not require high-end test instruments for determining these input values.

6.4 Discrete test signal

The discrete test signals were specifically chosen for conducting the noise interference and dynamic tests onboard the STM32 development board. These account for a large

portion of the compliance tests. One of the benefits of a discrete test signal is its parameters can be controlled easily and noise can be added or eliminated. In this test scenario, the onboard analog-to-digital conversion is bypassed. This prevents noise, as well as sampling and quantisation errors from being introduced into the system. This ensures an ideal test signal is free from noise and isolates the testing to solely encompass the performance of the TFT algorithm. A discrete test signal is also more versatile, as any signal expressed mathematically in MATLAB can be reproduced. This feature better suits the compliance tests as several reference test signals with special features including modulation, frequency ramps and added interference are required. The signal parameters of dynamic input signals for evaluating estimated parameters can also be easily obtained through the equation defining the test signal. The discrete test signal is based on a 50Hz sinusoidal waveform generated in MATLAB at a fixed sampling rate of 32kHz. This matches the sampling of the ADC onboard the STM32 development board for digitising an analog input signal. The total length of the test signal amounts to 100 cycles. This allows for multiple adjacent observation windows to be processed consecutively, ensuring any dynamic phasor behaviour occurring over time is observed and detected. Further details of each test signal prescribed for the compliance tests are described in the corresponding test sections. The resulting discrete test signal generated in MATLAB takes the form of a test vector representing the specified test signal and is applied to the input of the TFT algorithm. The generated test vector is transferred to the STM32 by entering it in the main C code and assigning it as a variable. When the firmware executes, the test vector is initiated in STM32 memory and transferred to the input of the TFT algorithm, where the phasor and frequency estimations get processed.

6.5 Frequency variation test

The frequency variation test assesses the accuracy of the phasor and frequency estimation for a range of input signal frequencies, and examines the effect this test scenario has on the estimated output result. This test is conducted by generating a collection of individual sinusoidal discrete test signals ranging from 48 Hz to 52 Hz, and applying these as individual inputs to the test. The discrete test signal is generated in MATLAB and is defined by the cosine function indicated by Equation (3.1) in Chapter Three. For this test, the signal parameters are specified as follows:

- Amplitude (X_m) = 1V
- Phase angle (φ) = π/4
- Frequency (f) = 48Hz to 52Hz (in intervals of 0.5 Hz)

The resulting phasor and frequency estimations produced by the STM32 development board are recorded for each test frequency. Based off these results, the TVE, FE and RFE values are calculated to determine the performance and accuracy of the estimation.

6.5.1 TVE results

The calculated TVE values of the one-cycle and two-cycle TFT estimations produced at the specified test frequencies are shown in Figure 6.2. The plot shows the phasor estimation for frequencies at 50 Hz yielding very small TVE values, illustrating the high accuracy of the estimated amplitude and phase angle outputs at nominal frequencies. As the signal frequency deviates further from the nominal frequency, we begin to observe an increase in the TVE value. The increasing TVE values spanning across the entire frequency test range are minimal, and remain below the 1% compliance limit as specified for P class and M class measurements. The phasor estimation of the one-cycle TFT produces a slightly improved accuracy in comparison to the two-cycle TFT version. In most cases, the two-cycle TFT version would be expected to provide the better performance. However, off-nominal frequency conditions causes the input signal to no longer fit uniformly within the observation interval, resulting in errors in the estimated phase angle. These errors in phase angle are further exacerbated when the observation interval spans over two cycles. It should be noted that this test includes phasor angle compensation as demonstrated in section 4.8.1 in Chapter 4. For this test, the phasor estimation and TVE results of the STM32 reflect the same performance and accuracy (TVE below 0.2%) as illustrated in the MATLAB simulation test results. This hereby confirms that the phasor estimation performed by the TFT algorithm has been successfully implemented onboard the STM32 hardware platform.

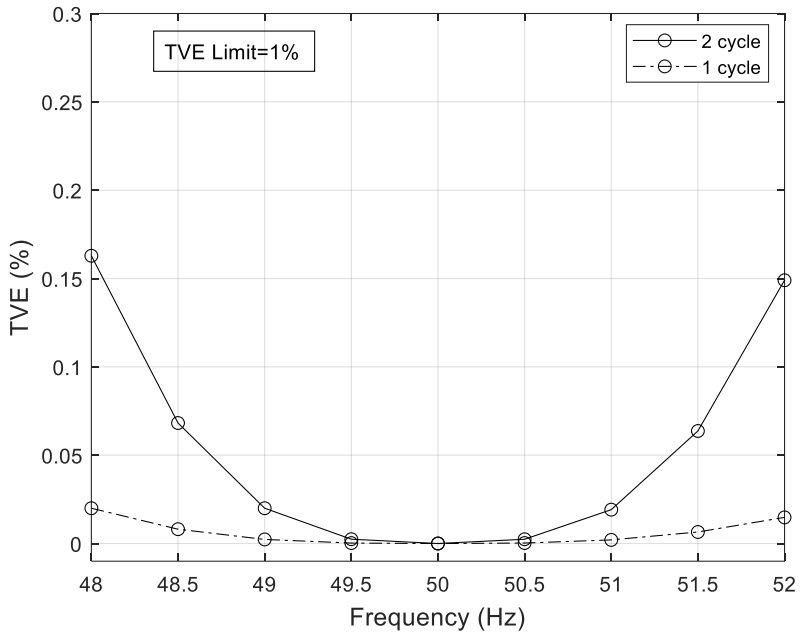

Figure 6.2: TVE values of the one-cycle and two-cycle TFT algorithm shown at discrete frequencies for frequency variation test

6.5.2 FE and RFE results

The performance and accuracy of the frequency estimation can be determined by examining the calculated FE and RFE values shown in Figure 6.3 and Figure 6.4 respectively. In Figure 6.3, the difference between the estimated and actual frequency are negligible at the fundamental frequency, however the error tends to expand as the frequency deviates further from this point. The one-cycle TFT algorithm offers the better accuracy in frequency estimation, and as a result meets the compliance error limit of 0.005 Hz (for P class and M class) throughout the entire frequency test range. On the other hand, the two-cycle TFT version begins to exceed the compliance limit as the signal frequency deviates beyond 1.5 Hz of the fundamental frequency. This is presumably a result of the extended observation interval. The frequency estimation results produced by the STM32 for this test maintains a similar trend as demonstrated in the MATLAB simulations.

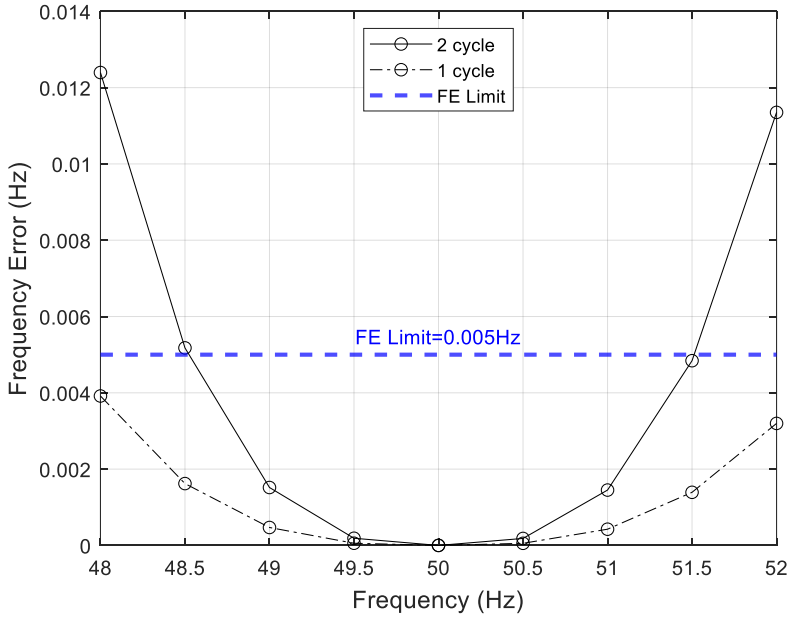

Figure 6.3: FE values of the one-cycle and two-cycle TFT algorithm shown at discrete frequencies for frequency variation test

The actual ROCOF and estimated ROCOF values produced for this test are recorded, and the resulting RFE is calculated. In Figure 6.4, the RFE values for both TFT versions are presented. Similarly to the FE values, the RFE values for both TFT versions produced at the nominal frequency are minimal and fall within the 0.01 Hz/s RFE limit. For signal frequencies outside 0.5 Hz of the fundamental, an increasing error is observed exceeding the RFE limit. The same observation is made in the MATLAB simulations tests.

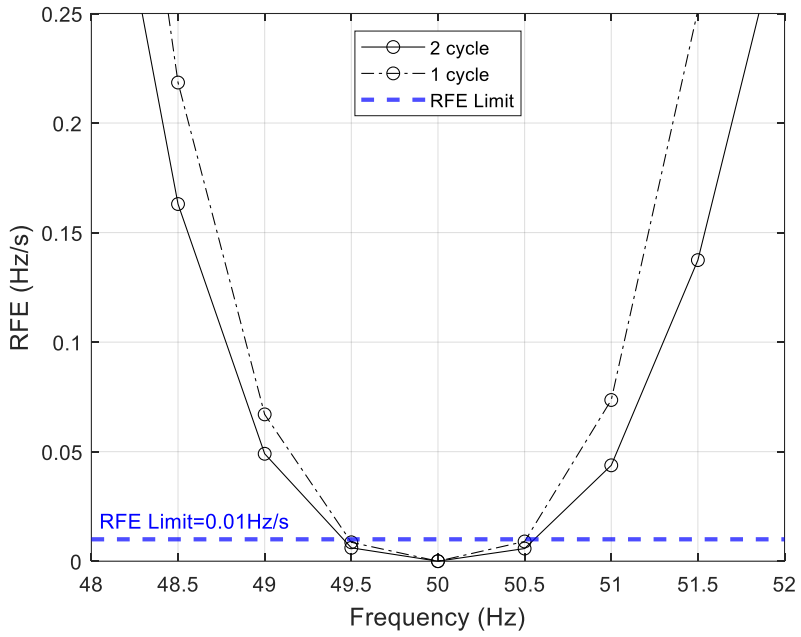

Figure 6.4: RFE values of the one-cycle and two-cycle TFT algorithm shown at discrete frequencies for frequency variation test

6.5.3 Discussion of results

The frequency variation test results produced by the STM32 development board are favourable. Firstly, the TVE results of the one-cycle and two-cycle TFT algorithms are well below the 1% stipulated compliance limit for P class and M class. For phasor estimation, the TFT algorithm displays the ability to maintain compliance for individual frequency tones spanning from 48 Hz to 52 Hz. Based on the low TVE values observed, this frequency range can most likely be extended to a wider range of frequencies. The one-cycle TFT presents slightly more accurate phasor estimates compared to the two-cycle version, although this is by a small margin. For the two-cycle TFT algorithm, a greater decline in accuracy is observed for the estimation of signals at off-nominal frequencies. This degradation is mainly contributed by error in the estimated phase angle. At off-nominal frequencies, the phase angle rotates after each cycle and any phase angle errors incurred get compounded over two cycles when employing the two-cycle TFT algorithm. The frequency and ROCOF estimates meet the FE and RFE compliance limits, however this is mostly achieved at signal frequencies close to the 50 Hz fundamental. The one-cycle TFT does however meet the 0.005 Hz FE limit for the entire frequency test range. When examining the FE and RFE values, an increasing sensitivity proportional to the off-nominal frequency is observed. This is due to the FE and RFE being first and second derivatives of the

estimated phasor. This result highlights a limitation of the TFT whereby at off-nominal frequencies inaccuracies in frequency and ROCOF are introduced. As for the RFE results, the one-cycle and two-cycle TFT are found to be similar. Finally, all the aforementioned results reflect the outcomes observed in the corresponding MATLAB simulation, and thus hereby verifies the implementation and operation of the TFT algorithm onboard the STM32 development board.

6.6 Magnitude and phase angle variation test

The magnitude and phase angle variation test are individual tests grouped together as they both form part of the steady-state compliance tests and follow a similar test procedure. The static test signal and test conditions allow these tests to be conducted utilising a continuous analog waveform as the input test signal. This is generated by the RS Pro 2205A-20 built-in function generator and is used to evaluate the measurement performance of the STM32 development board when deployed with the one-cycle and two-cycle TFT algorithm. The test signals for the magnitude and phase angle variation test are both expressed by Equation (3.1), as in the previous test. It should be noted that the magnitude and phase angle variation test forms an integral part of the experimental verification, as it is the only test that includes the digitisation of the input test signal carried out by the ADC onboard the STM32 prototype.

6.6.1 Magnitude variation test

The magnitude variation test evaluates the accuracy of the phasor estimation for different amplitude values of the input signal ranging from 10% to 120% of the nominal value. This range covers P class and M class measurements. It should be noted that this range of amplitudes are specific to voltage measurement only, and does not include current measurement conditions. For this test, the nominal value is selected as 500mV. The input test signal takes the form of a 50 Hz sinusoidal waveform with amplitudes of 50mV, 250mV, 500mV and 600mV, respectively, and thereby covers the required amplitude test range. The phase angle of the input signal is taken as the arbitrary angle that occurs at the moment the input signal is sampled, and is determined by measurement with an oscilloscope as described in section 6.3.1. In Figure 6.5, a screenshot of the oscilloscope measurement of the 50 Hz input test signal at the nominal amplitude of 500mV is shown. The input test signal is DC offset at 1V to allow for the ADC to capture and quantize both positive and negative half cycles of the continuous waveform.

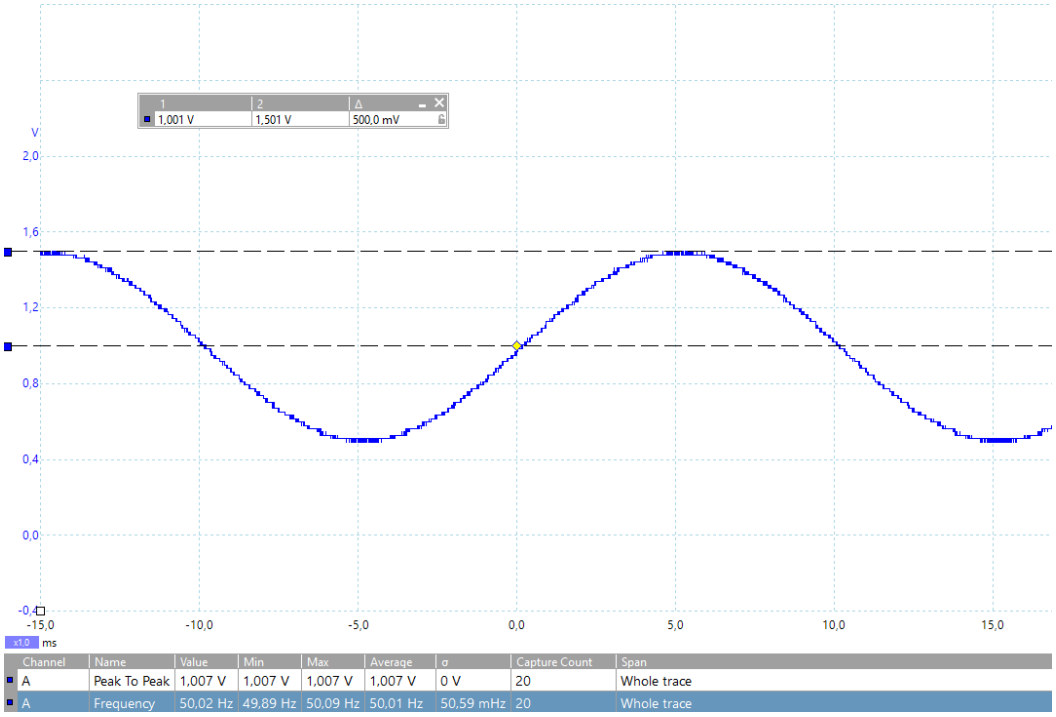

Figure 6.5: A screenshot of the oscilloscope measurement of the input test signal utilised in the magnitude variation test

The actual phasor parameters based on the input test signal, and the estimated phasor results are recorded and the respective TVE values are calculated. It should be noted that the evaluation of the frequency and rate of change of frequency estimations are omitted for this test as they are not a requirement in the IEEE C37.118.1-2011 standard.

6.6.2 TVE results

The one-cycle and two-cycle TFT phasor estimation results displaying the calculated TVE values for the magnitude variation test are presented in Table 6.1.

Table 6.1: Measurement results of one-cycle and two-cycle TFT for magnitude variation test

	Input signal parameters		**Measured signal parameters**		**TVE (%)**
	Amplitude (V)	**Phase angle (radians)**	**Amplitude (V)**	**Phase angle (Radians)**	***(TVE Limit = 1%)***
One-cycle TFT	0.050	-0.879960102	0.0483	-0.725345	5.55
	0.250	-2.673495348	0.2488	-2.652836	2.12
	0.500	-2.288336089	0.495	-2.273403	1.79
	0.600	-1.105840614	0.5929	-1.076583	2.25
Two-cycle TFT	0.050	-0.833150372	0.048204096	-0.836222	3.6
	0.250	-1.807672413	0.246571313	-1.806269	1.38
	0.500	-2.372216613	0.494097751	-2.367556	1.27
	0.600	-1.937420189	0.593712651	-1.933715	1.11

The results show the calculated TVE values being marginally over the 1% compliance limit stipulated for P class and M class. The best test result is produced by the two-cycle TFT algorithm for a nominal input signal amplitude at 600mV. Overall the phasor estimation results produced by the two-cycle TFT yields slightly more accurate results than the one cycle TFT. This is to be expected due to the observation interval being double in length. The largest TVE values are observed for the smaller input signal amplitudes of 50mV, indicating a decline in accuracy as the input signal amplitude is decreased. This is to be expected as any noise present in the system will cause greater signal distortion and degradation at lower signal levels. When evaluating the overall TVE results of the STM32 prototype against the results of the MATLAB simulations tests, we notice a significant decline in accuracy. This is primarily caused by unwanted noise infiltrated into the analog input test signal, and affecting the accuracy of the TFT computation. It should be noted that in contrast to the MATLAB simulation tests, the input of the TFT algorithm is non-ideal and susceptible to noise as the test signal generation and the sampling and quantization process all takes place in real time.

6.6.3 Phase angle variation test

In the phase angle variation test, the performance and accuracy of the phasor estimation is assessed differently by keeping the input signal amplitude constant at a nominal value and varying the phase angle only. As the input test signal is a continuous analog waveform, the test is conducted using several arbitrary phase angle values, as specific phase angle values are difficult to implement in a real time test environment. The phase angle is based on the point in the cycle at which the input signal is sampled, and this value is verified by measurement with an oscilloscope referenced at the time the analog-to-digital conversion is started. With all the parameters of the input test signal now confirmed, as well as the phasor estimation results computed on the STM32 development board captured, the resulting TVE values of the one-cycle and two-cycle TFT algorithms can be calculated.

6.6.4 TVE results

In Table 6.2, the resulting calculated TVE values along with the corresponding input signal parameters and estimated output signal parameters are presented.

Table 6.2: Measurement results of one-cycle and two-cycle TFT for phase angle variation test

	Input signal parameters		Measured signal parameters		TVE (%) *(TVE Limit = 1%)*
	Amplitude (V)	**Phase angle (radians)**	**Amplitude (V)**	**Phase angle (Radians)**	
One-cycle TFT	0.500	-0.974522041	0.4975	-0.982619	0.95
	0.500	-2.351482101	0.4979	-2.335594	1.64
	0.500	-1.256637061	0.4918	-1.247648	1.87
	0.500	-2.630769688	0.489	-2.618834	2.5
	0.500	-1.879614885	0.4903	-1.855305	3.09
	0.500	-2.69831393	0.49342	-2.653602	4.59
	0.500	-2.727216583	0.4939	-2.706401	2.4
	0.500	-1.008451242	0.4975	-0.993976	1.53
Two-cycle TFT	0.500	-0.301718558	0.492996	-0.309829	1.62
	0.500	-0.716911444	0.493448	-0.73329	2.09
	0.500	-2.263517507	0.494219	-2.235749	2.99
	0.500	-1.130030877	0.493216	-1.127947	1.37
	0.500	-2.983570543	0.493303	-2.973153	1.69
	0.500	-3.091013012	0.493701	-3.071829	2.28
	0.500	-1.832805154	0.493949	-1.834578	1.22
	0.500	-2.766172331	0.494131	-2.748418	2.12

By examining the overall TVE results, it can be seen that both one-cycle and two-cycle TFT algorithms are slightly outside the 1% compliance threshold, and therefore do not meet the requirement for P class and M class measurements. Similarly to the magnitude variation test, the phase angle variation test also reveals a decline in accuracy when compared to the corresponding MATLAB simulations. This is primarily attributed to noise infiltrated and present in the continuous analog test signal. The TVE results do however show more consistency across the calculated TVE values. This is due to the input signal amplitude being kept constant at a nominal value of 500mV. This observation also suggests that the performance and accuracy of the TFT algorithm is not significantly affected by different phase angle values of the input signal, with this being the given influence quantity for this test. Overall the TVE values produced by the two-cycle TFT algorithm offers a small improvement in accuracy in comparison to the one-cycle TFT algorithm. The same result is observed in the magnitude variation test, and is due to the digitised input test signal spanning over two cycles.

6.6.5 Discussion of results

The magnitude and phase angle variation test evaluates the performance and accuracy of the phasor measurement produced onboard the STM32 prototype for a range of signal amplitude and phase angle values. This test is of particular importance as it is the only test where a continuous analog waveform is applied as the input test signal and shows the effects there of. By comparison of the TVE results of the one-cycle and two-cycle TFT algorithm, we observe the two-cycle TFT providing a slightly better performance and accuracy in phasor estimation. The two-cycle TFT estimation produced the lowest TVE values, however the one-cycle TFT TVE values are only larger by a small margin. This result demonstrates and confirms the influence that the length of the observation interval has on the accuracy of the measurement.

Overall the TVE results for both TFT algorithms exceed the 1 % TVE limit and therefore does not meet the compliance requirements for the IEEE C37.118.1-2011 standard. This result differs from that obtained in the MATLAB simulations with a reduction in accuracy. This is primarily attributed to the differences between the simulation and the real-world test environment. In the MATLAB simulations, a perfectly sampled sinusoidal waveform of the input signal is provided as the input to the TFT algorithm. However in real-world scenarios, such an ideal sinusoidal test signal is not possible to produce, as the digitised signal provided to the TFT algorithm will be subject to inaccuracies introduced in the analog-to-digital conversion, as well as noise contained in the generated input signal. These factors produce inaccuracies in the estimation result that negatively affect the TVE value. In Figure 6.6, a sampled portion of the input signal after being digitised by the STM32 ADC is plotted alongside the corresponding MATLAB simulated test signal. These signals are provided as the input to the TFT algorithm. In this plot we observe the jagged peaks and edges that are present in the ADC sampled signal, in contrast to the ideal simulated signal generated in MATLAB. This distortion in the digitised input signal has an impact on the estimated phasor output when computed by the TFT algorithm and results in a degradation in accuracy that is observed in the test results. To improve the performance and accuracy of the phasor measurement, additional front-end filtering can be employed to suppress any unwanted noise present in the input test signal, however this will increase the overall measurement latency.

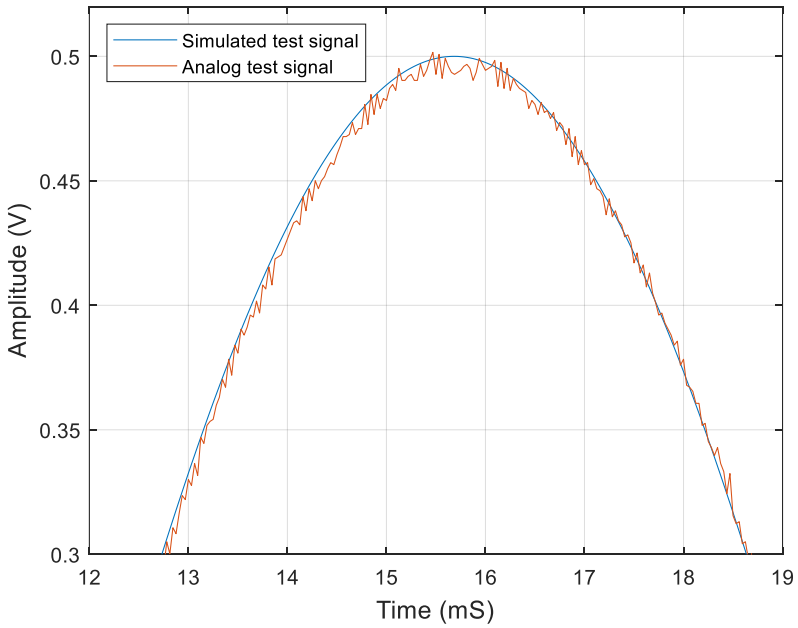

Figure 6.6: A portion of the continuous analog test signal digitised by the ADC, plotted against the discrete simulated test signal generated in MATLAB

6.7 Harmonic distortion test

The harmonic distortion test evaluates the performance of the phasor estimation when subjected to a fundamental signal corrupted by harmonic interference. The test is conducted by applying a discrete test signal (as described in section 6.4) as an input to the test. This discrete test signal is comprised of a 50 Hz sinusoidal signal added with a single 4th harmonic component, at a level of either 1% or 10% the nominal amplitude. The discrete test signal is expressed by Equation (4.1) indicated in Chapter Four. The signal parameters for this test are specified as follows:

- Amplitude (X_m) = 1V
- Nominal power system frequency: (f_o) = 50 Hz
- Amplitude of harmonic component at a level of 1% (P class) and 10% (M class): (X_h) = 0.01V or 0.1V
- Frequency of 4th harmonic component: (f_h) = 200 Hz

According to the IEEE C37.118.1-2011 standard, the harmonic distortion test requires the testing of individual interfering harmonic components up to the 50th harmonic. For the purpose of verifying the phasor measurement produced by the STM32 for the harmonic test, only the 4th harmonic component is considered. An harmonic interference level of 1% the nominal amplitude is applied for P class measurements, and 10% for M class measurements. An illustration of the discrete test signal specified

for M class containing the required harmonic component of 10% is shown in Figure 6.7. The harmonic distorted test signal is indicated by the blue solid line and plotted alongside the fundamental (orange dashed line) and harmonic components (maroon dash-dotted line). Here the effect of the harmonic interference can be seen distorting the input signal and causing the harmonic component to be superimposed on the 50 Hz sinusoidal waveform.

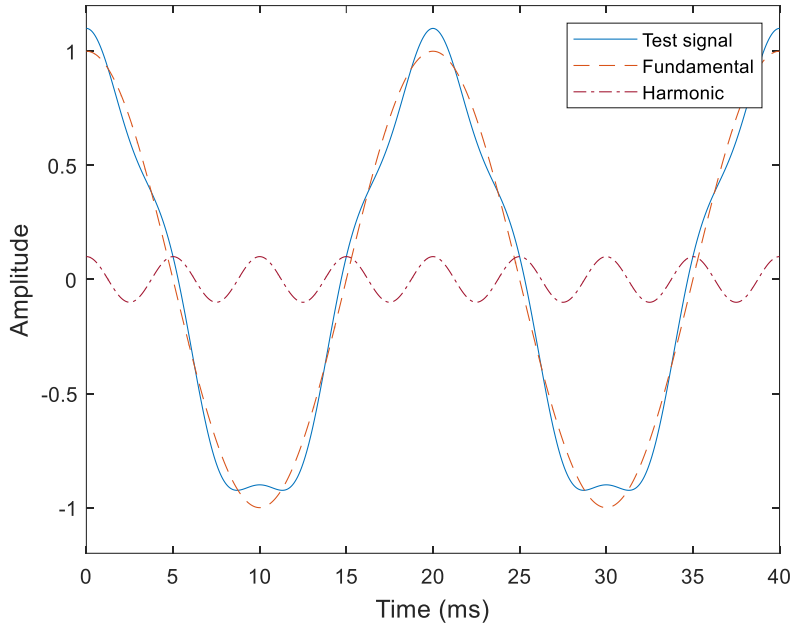

Figure 6.7: Input test signal composed of the 50 Hz fundamental signal and the 4th harmonic component at a level of 10%

This test is conducted with two types of input test signals, the first being a signal at a 1% harmonic distortion level required for P class, and the second at a level of 10% for M class. The harmonic distorted test signals are applied to the input of the one-cycle and two-cycle TFT algorithms deployed on the STM32 development board, and the computed phasor and frequency estimated values are recorded and assessed.

6.7.1 TVE results

The estimated amplitude and phase angle values produced by the STM32 board are recorded and the resulting TVE values are calculated. The calculated TVE results for the harmonic (4^{th}) distortion level of 1% for P class, and 10% for M class, are shown separately in Figure 6.8 and Figure 6.9 respectively. In Figure 6.8, the TVE values shown for P class measurements are minimal and comfortably meet the 1% compliance limit. Proceeding to the M class TVE results presented in Figure 6.9, we observe an increase in the TVE value, however the results are still within compliance limits. This is to be expected as the increased harmonic distortion specified for M class

measurements causes an increase in the TVE value, and adversely affects the accuracy. The two-cycle TFT achieves the better performance for both P class and M class, by producing the lower TVE values in comparison to the one-cycle TFT. A similarity between the TVE results produced by the STM32 prototype and the MATLAB simulations are apparent for an interference of the 4th harmonic component (200 Hz).

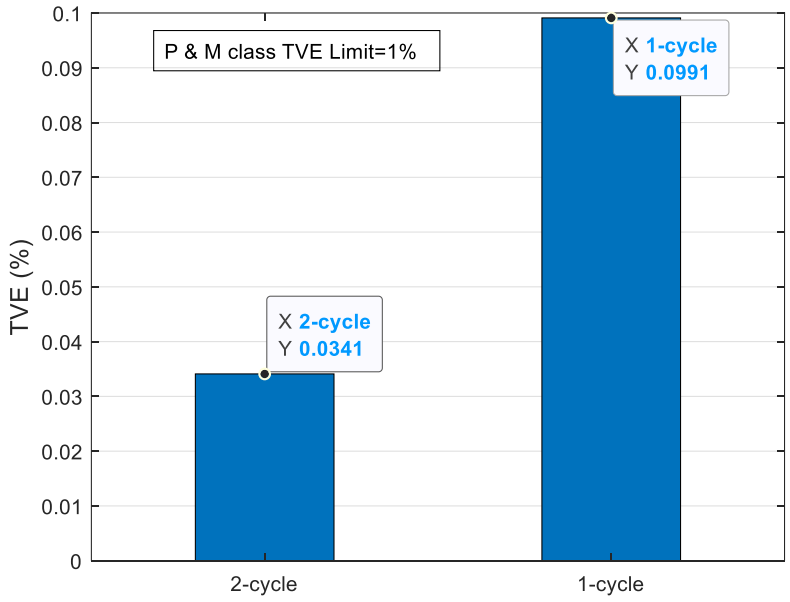

Figure 6.8: TVE results of harmonic distortion test for input signals containing a harmonic component of 1%

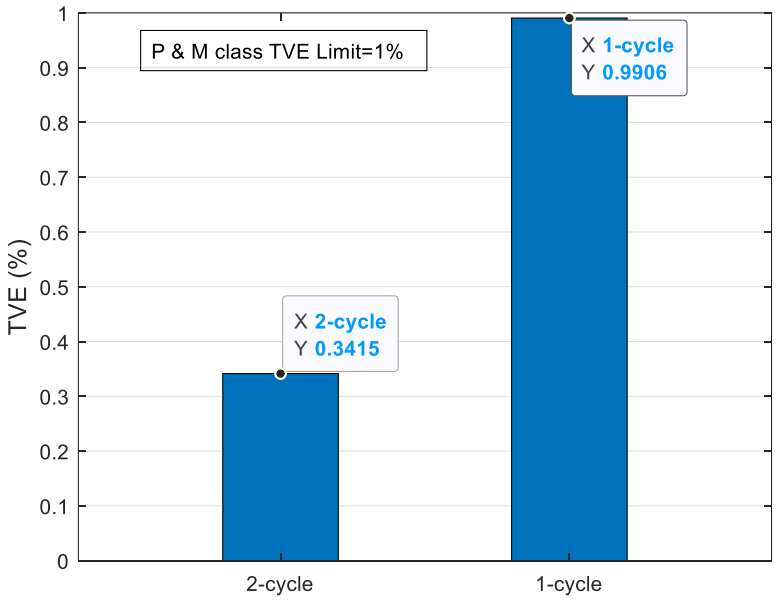

Figure 6.9: TVE results of harmonic distortion test for input signals containing a harmonic component of 10%

6.7.2 FE and RFE results

The frequency estimation and ROCOF results of the STM32 development board is also evaluated separately for P class (1%) and M class (10%). The FE results for P class are shown in Figure 6.10, followed by the M class results shown in Figure 6.11. Both P class and M class FE limits are not met by neither one-cycle nor two-cycle TFT algorithm, as the FE values exceed the 0.005 Hz limit. It is only the two-cycle TFT that comes close to meeting the P class (1%) FE compliance limits and offers the better performance in frequency estimation. The same outcome has also been observed in the MATLAB simulation tests for a harmonic interference of 200 Hz.

The ROCOF results are not illustrated in a graph as a zero frequency deviation is estimated, indicating no error as the tests are conducted at the fundamental frequency and no change in frequency is expected. In the simulation tests we see very small ROCOF values produced, but when implemented on the STM32 these values are truncated to zero due to the lower resolution of 32 bits as opposed to 64 bits used in the MATLAB simulations.

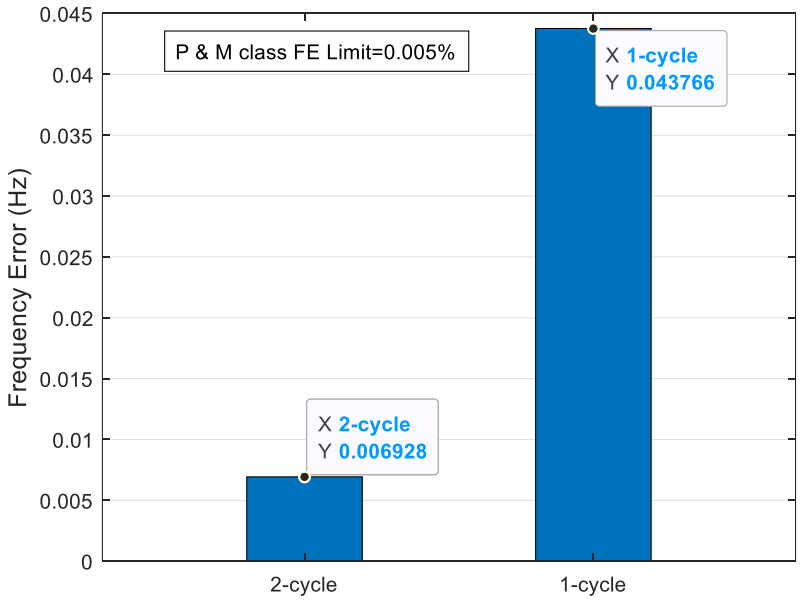

Figure 6.10: FE results of harmonic distortion test for input signals containing a harmonic component of 1%

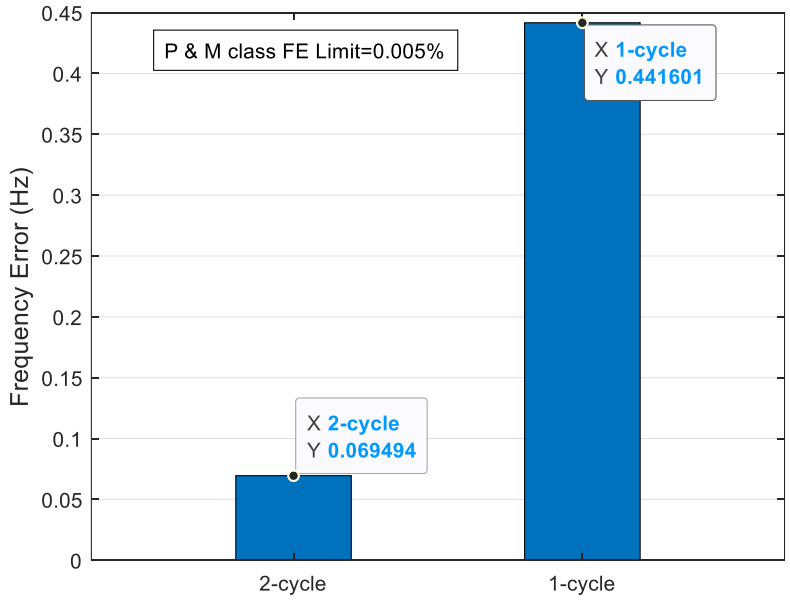

Figure 6.11: FE results of harmonic distortion test for input signals containing a harmonic component of 10%

6.7.3 Discussion of results

The harmonic interference test results confirm the ability of the STM32 to produce accurate phasor estimates in the presence of harmonic disturbances. The one-cycle and two-cycle TFT achieves the TVE compliance limits for P class as well as M class measurements. This demonstrates the TFT algorithms immunity to harmonic interference and successful suppression of harmonic components contained in the input signal without any additional filtering. The TVE values are found to be proportional to the harmonic level, which is apparent as the harmonic level is increased between P class and M class. Overall the two-cycle TFT produces the lower TVE values, and therefore offers the better performance over the one-cycle TFT version. The TFT algorithm struggles to meet FE compliance limits for both P class and M class. This outcome has also been observed in the MATLAB simulations of the harmonic interference test. The FE limits stipulated by the IEEE C37.118.1-2011 is quite stringent and difficult to meet. Some front-end filtering to further suppress harmonic infiltration may be required in order to improve frequency estimation performance and accuracy.

6.8 Out-of-band interference test

The out-of-band interference test assesses the ability of the TFT algorithm to reject and suppress unwanted interharmonic and subharmonic components, and hereby reduce the impact this interference has on the phasor estimation result. The test

evaluates the accuracy of the phasor and frequency estimation computed by the TFT algorithm when subjected to input signals containing a single interfering component at interharmonic or subharmonic frequencies occurring outside the passband. This test is carried out by generating a discrete test signal in MATLAB and applying it as an input to the test. The test signal consists of a main sinusoidal signal at 50 Hz, added with an out-of-band component at a level of 10% the nominal amplitude. For this test, a single subharmonic component occurring at 20 Hz was selected to represent the signal disturbance originating outside the passband. The discrete test signal is expressed by Equation (4.2) provided in Chapter Four. The discrete test signal parameters for this test are specified as follows:

- Amplitude (X_m) = 1V
- Nominal power system frequency: (f_o) = 50 Hz
- Amplitude of out-of-band component at a level of 10% : (X_i) = 0.1V
- Frequency of out-of-band component: (f_i) = 20 Hz

In Figure 6.12, a visual representation of the discrete test signal with the aforementioned parameters is shown indicated by the solid blue line. The plot also illustrates the 50 Hz fundamental signal (black dashed line) and the 20 Hz sub-harmonic interference signal (maroon dash-dotted line) which the discrete test signal is composed of. This illustrates the extent to which the input test signal deviates from the fundamental signal, and undergoes distortion caused by the interfering sub-harmonic component.

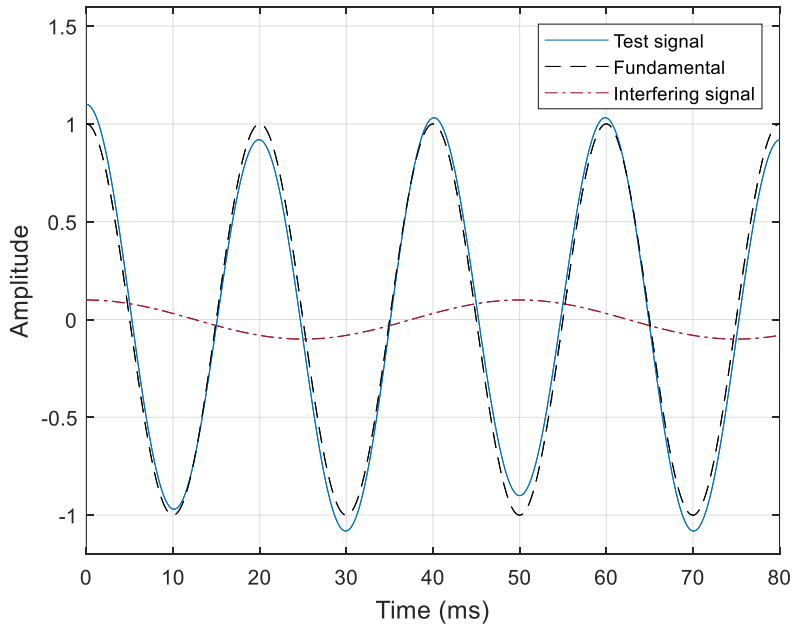

Figure 6.12: Input test signal for out-of-band interference test, composed of a 50 Hz fundamental signal and a 20 Hz sub-harmonic component

In Chapter Four, it has already been established that the TFT algorithm is unable to meet the compliance requirements for the out-of-band interference test and requires additional front-end filtering. Despite of these findings, the out-of-band interference test is still conducted to determine and demonstrate the impact this input conditions has on performance and how accuracy is compromised when deployed on the STM32 platform. The discrete test signal is applied to the input of the one-cycle and two-cycle TFT algorithms, and the resulting estimated amplitude, phase angle, frequency and ROCOF values produced onboard the STM32 are recorded for evaluation.

6.8.1 TVE Results

The IEEE C37.118.1-2011 standard specifies the out-of-band interference test for M class measurements only, and therefore is assessed solely for this performance class. The TVE performance indicators for this test are calculated according to the phasor estimation results produced by the STM32 hardware platform. The resulting TVE values of the one-cycle and two-cycle TFT algorithms are shown in Figure 6.13. Both TFT algorithm versions are shown exceeding the 1.3% compliance limit, with the two-cycle TFT offering the more accurate estimate among the two. These results correspond with the findings obtained in the MATLAB simulation tests, and thus confirm the successful implementation of the TFT algorithm onboard the STM32 platform.

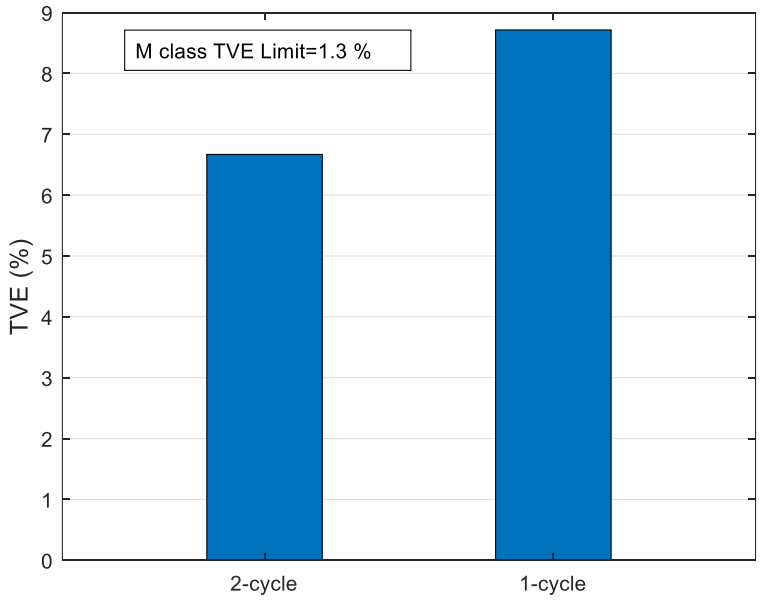

Figure 6.13: TVE results of the out-band-interference test

6.8.2 FE and RFE Results

The performance of the frequency estimation produced by the STM32 during the out-of-band interference test is examined by calculating the difference between the estimated frequency and expected frequency. The resulting FE values of the one-cycle and two-cycle TFT versions are presented in Figure 6.14. As expected, the FE values surpass the maximum allowable error and the compliance requirement is not met. Once again the two-cycle TFT out performs the one-cycle TFT by incurring much smaller error values in the frequency estimation, although this is still not low enough to meet compliance levels required by the IEEE C37.118.1-2011 standard.

In Figure 6.15, very large error values in the ROCOF estimation is also shown exceeding the RFE limit. This has however been anticipated for and previously observed in the MATLAB simulation tests.

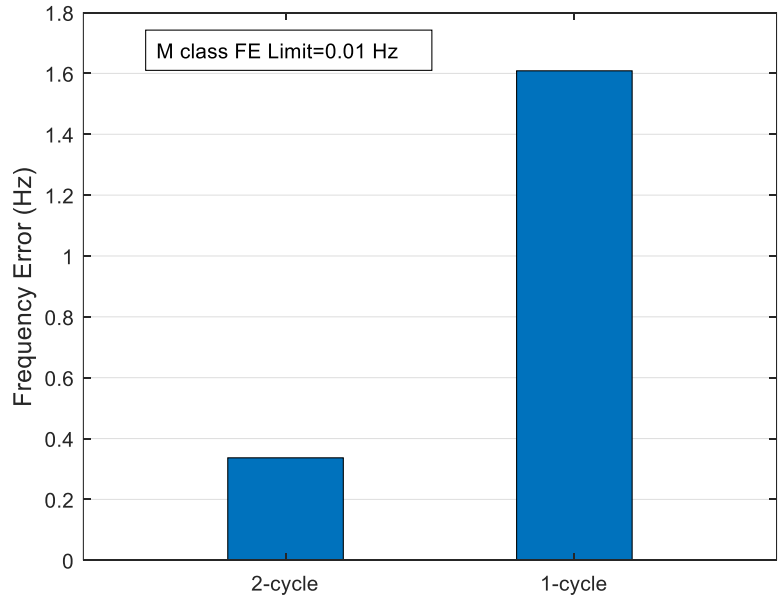

Figure 6.14: FE results of the out-band-interference test

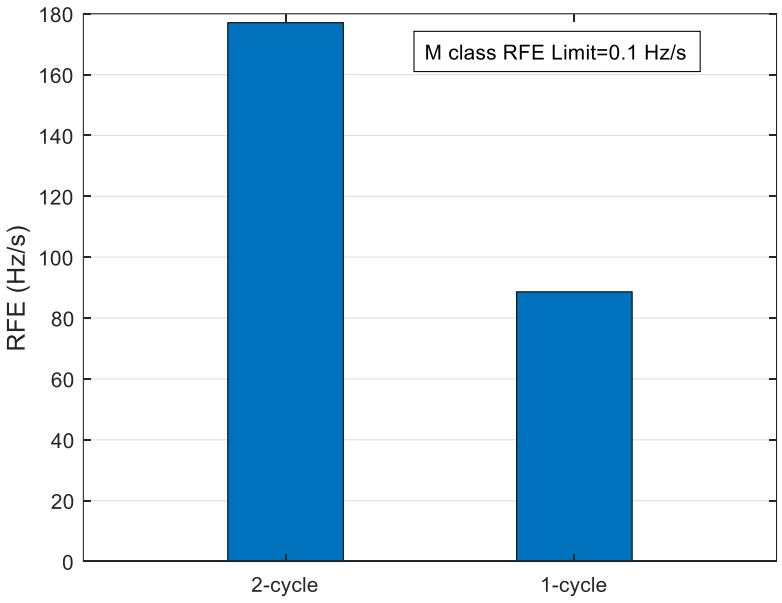

Figure 6.15: RFE results of out-band-interference test

6.8.3 Discussion of results

In conclusion, the overall STM32 results for the out-of-band interference test is consistent with the findings obtained for the corresponding test simulated in MATLAB. The phasor and frequency estimation of the TFT algorithm when deployed on the STM32 hardware platform performs as expected and this confirms its successful operation and implementation. The two-cycle TFT is seen to provide the better

performance and accuracy in comparison to the one-cycle version. An important result to note is that the TFT algorithm is unable to achieve the compliance requirements for the out-of-band interference test. This reveals a limitation of the TFT algorithm whereby the input signals containing the out-of-band interference are unable to be suppressed and has a detrimental effect on accuracy. These type of input conditions are seen to negatively impact the ROCOF estimation value the most, as this output parameter is computed by means of derivative and is sensitive to noise. In order to utilise the TFT algorithm successfully when subjected to out-of-band interference, front-end filtering is essential to improve accuracy and bring estimated values within compliance limits.

6.9 Modulation test

The modulation test evaluates the performance of the phasor and frequency estimation produced by the STM32 phasor measurement prototype when presented with a dynamic input signal exhibiting amplitude and phase angle modulation. The test is conducted with a discrete test signal generated in MATLAB and is composed of a 50 Hz sinusoidal signal with its amplitude and phase angle modulated at a frequency of 2 Hz. A modulation level of 10% the nominal signal amplitude is applied. The modulated test signal is defined by Equation (4.3) as indicated in Chapter Four, and is assigned with the following input parameters for both P class and M class compliance:

- Amplitude: 1V
- Nominal power system frequency: f_o= 50 Hz
- Modulation frequency: f = 2 Hz
- Amplitude modulation factor: k_x = 0.1
- Phase angle modulation factor: k_a = 0.1

For this test, the IEEE C37.118.1-2011 standard states several input test conditions spanning over the entire modulating frequency range, however this is exhaustive for the purpose of this research project, and therefore only the aforementioned input test parameters are considered. A visual representation of the discrete test signal generated in MATLAB is shown in Figure 6.16. We observe the 2 Hz envelope of the modulating signal causing the amplitude of the fundamental to oscillate around the 1V nominal amplitude.

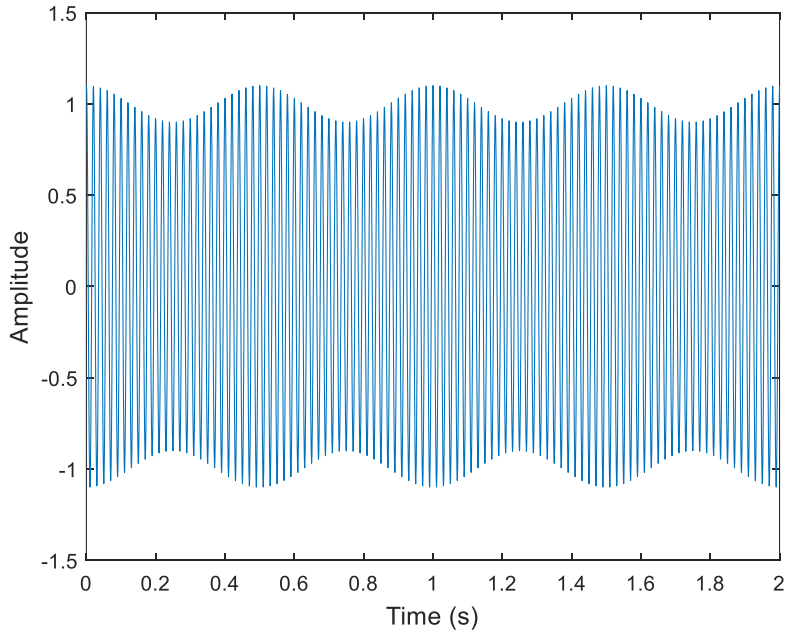

Figure 6.16: Amplitude modulated input test signal

This test is conducted for the one-cycle and two-cycle TFT algorithms deployed on the STM32 development board. The estimated amplitude, phase angle and frequency results are captured and the performance indicators are calculated.

6.9.1 TVE results

According to the reference test signal indicated by Equation (4.3), the amplitude and phase angle of the input test signal are expressed as modulating signals. From this equation, we obtain the modulating components $X_m[1 + k_x\, cos(\omega t)]$ and $k_a\, cos(\omega t - \pi)$, which defines the modulating amplitude and phase angle of the input signal. These are used to calculate the theoretical or expected instantaneous amplitude and phase angle values of the input signal. These theoretical values are used as a benchmark to evaluate the accuracy of the estimated amplitude and phase angle. To further illustrate this, the modulating amplitude of the input reference test signal is plotted together with the corresponding estimated amplitude produced by the STM32 board, and is shown in Figure 6.17. The estimated amplitude of the input signal is denoted by the orange circular markers, while the solid blue line represents the theoretical or expected amplitude. These plots demonstrate the manner in which the estimated amplitude successfully tracks the modulated envelope of the input test signal. A similar plot of the modulating phase angle of the input reference test signal plotted alongside the corresponding estimated phase angle is shown in Figure 6.18.

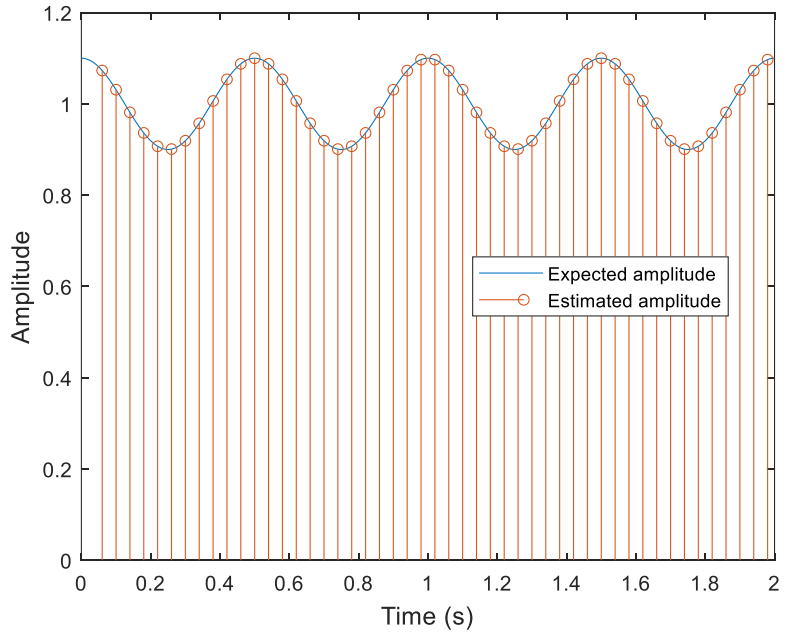

Figure 6.17: Estimated amplitude of modulated input test signal

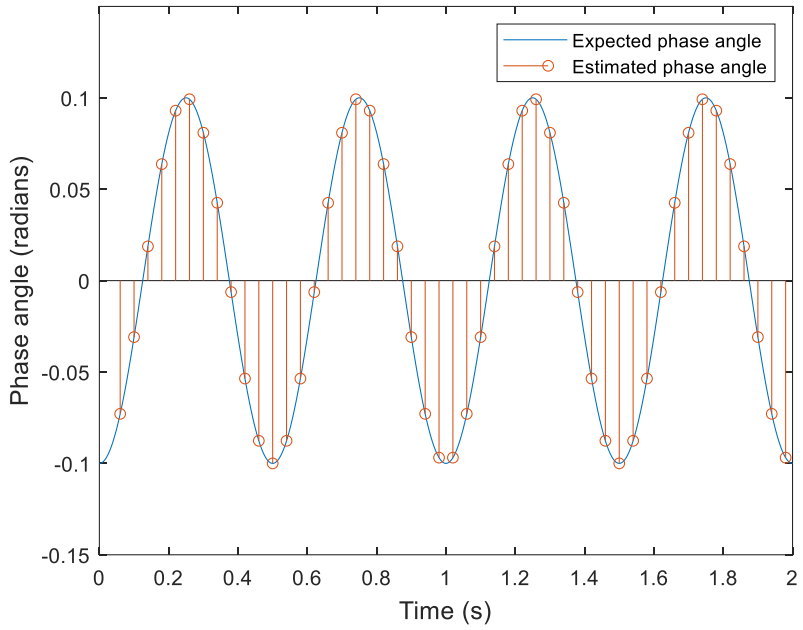

Figure 6.18: Estimated phase angle of modulated input signal

To benchmark the performance and accuracy of the phasor estimation produced by the one-cycle and two-cycle TFT algorithm, the corresponding TVE values are calculated. The calculated TVE results are presented in Figure 6.19. The resulting TVE values for both the one-cycle and two-cycle TFT algorithm are minimal and comfortably fall within the 3% compliance limit for P class and M class. This test result demonstrates the accuracy and proficiency of the TFT algorithm in estimating and tracking dynamic signals and conditions. Furthermore, these findings match the results

of the combined amplitude and phase angle modulation tests obtained from the corresponding MATLAB simulations, and demonstrate the algorithms capability of accurately measuring practical power system conditions.

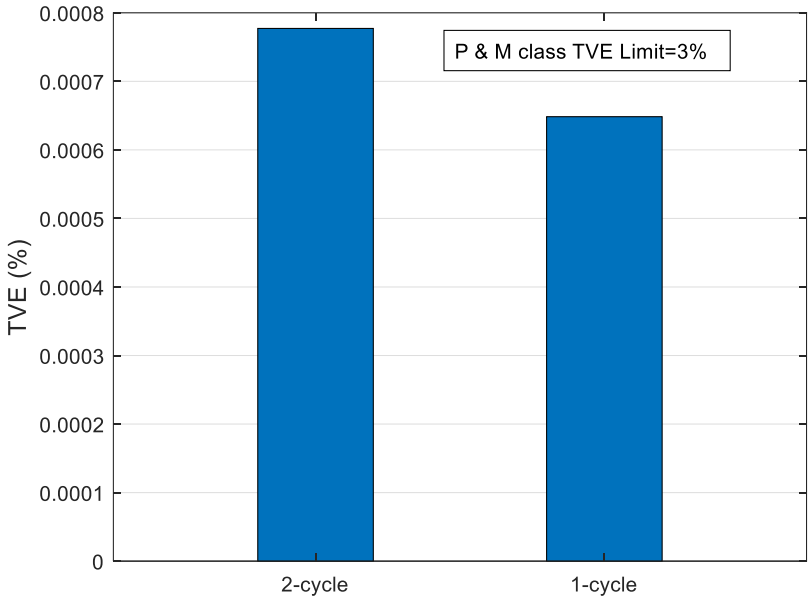

Figure 6.19: TVE results for modulation test

6.9.2 FE and RFE Results

The estimated frequency values for the modulation test are recorded and the FE and RFE values are calculated to determine the performance and accuracy of the frequency estimation produced by the TFT algorithm onboard the STM32 hardware platform. The calculated FE values for the one-cycle and two-cycle TFT algorithm are shown in Figure 6.20. The FE values remain comfortably below the prescribed compliance limits, in fact the compliance for both P class and M class are met by the one-cycle and two-cycle TFT algorithm. The introduction of modulation to the input signal does not compromise the accuracy, as precise frequency estimations are maintained even in the presence of modulations. It is also worth noting that while the two-cycle TFT error values are marginally higher than those of the one-cycle version, they still remain well within the P class and M class compliance limits.

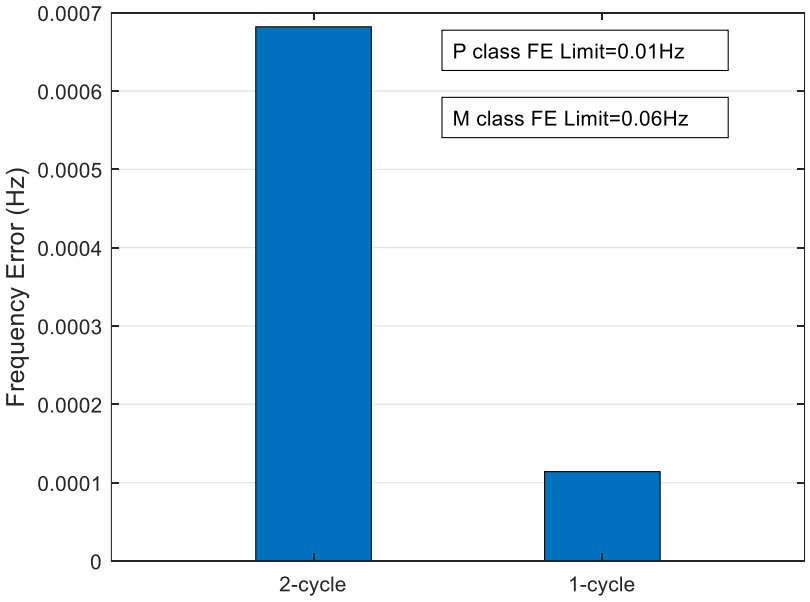

Figure 6.20: FE results for modulation test

The RFE results for the one-cycle and two-cycle TFT algorithm are shown in Figure 6.21. It was found that the two-cycle TFT produces smaller RFE values compared to the one-cycle version, however both variants fall within the maximum allowable limit for P class and M class measurements. These RFE results are consistent with the results produced in the MATLAB simulations.

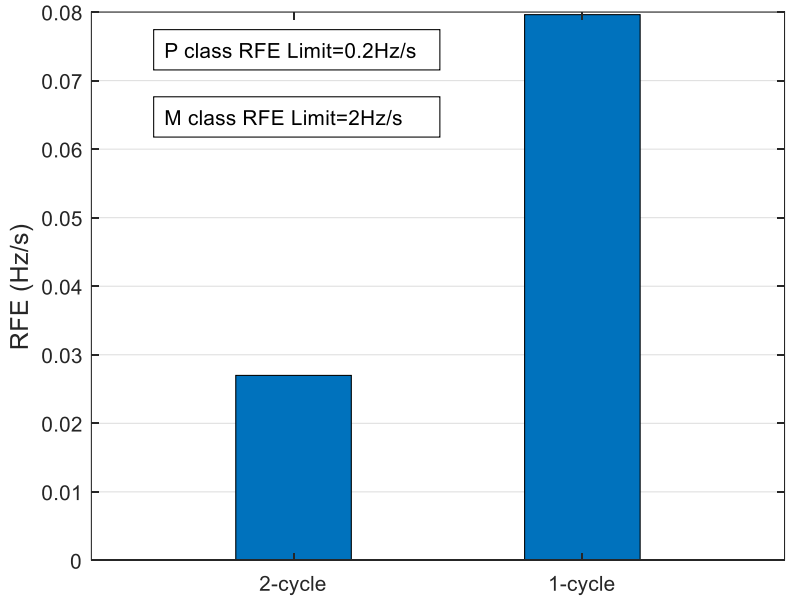

Figure 6.21: RFE results for modulation test

6.9.3 Discussion of Results

The results of the modulation test confirm that both one-cycle and two-cycle TFT algorithms satisfy the performance criteria for both P class and M class measurements. In this test, the TFT algorithm demonstrates the ability to successfully track amplitude and phase angle variations that occur within an observation window. The frequency estimation has also produced results meeting the IEEE C37.118.1-2011 compliance requirements. These findings concludes that the TFT algorithm can successfully be employed for the measurement of power systems exhibiting modulation signals.

6.10 Frequency Ramp Test

The frequency ramp test assesses the performance of the TFT algorithm when estimating an input test signal that undergoes a frequency ramp. During this test, the frequency of the input signal is increased linearly from 50 Hz to 52 Hz at a positive ramp rate of 1 Hz/s, while maintaining a constant amplitude. A discrete test signal exhibiting this frequency ramp is generated in MATLAB and applied as the input to the test. The test signal is defined in the IEEE C37.118.1-2011 standard and is represented by Equation 4.8 provided in Chapter Four.

The input parameters of the discrete test signal specified for the frequency ramp test are as follows:

- Amplitude: 1V
- Nominal power system frequency: f_o= 50 Hz
- Ramp rate: Rf =+ 1Hz/s
- Frequency ramp range: 50Hz to 52Hz

The provided input test parameters and conditions are in accordance with the IEEE C37.118.1-2011 standard, and are specified for both P class and M class measurements. The test is conducted with the one-cycle and two-cycle TFT algorithms and the estimation results are recorded for evaluation and analysis.

6.10.1 TVE Results

The computed amplitude and phase angle estimation values produced on the STM32 development board, utilising the one-cycle and two-cycle TFT algorithms, are recorded for evaluation. These phasor estimated results, along with the given reference input test signal values, are used to calculate the TVE, and consequently determine the accuracy of the measurement. In Figure 6.22, a frequency plot of the calculated TVE values of the one-cycle and two cycle TFT estimations are presented. The resulting TVE values of the one-cycle and two-cycle TFT algorithm indicate a minimal error,

much like the results produced in the MATLAB simulation tests. An increase in the TVE value is observed as the frequency ramps from 50 Hz to 52 Hz, however this remains well within the 1% compliance limit. In summary, the TVE results are favourable, with both one -cycle and two-cycle TFT versions meeting the compliance requirements for P class and M class measurements.

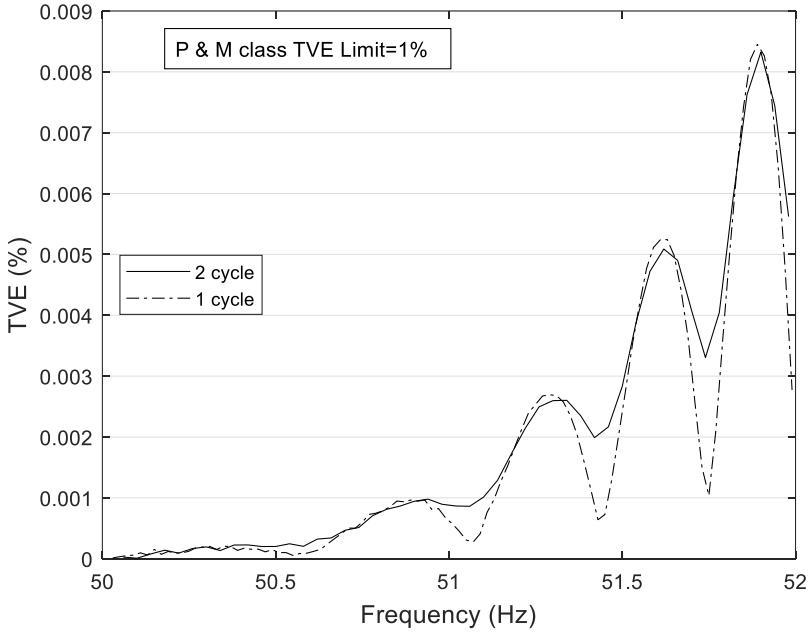

Figure 6.22: TVE results of one-cycle and two-cycle TFT for frequency ramp test

6.10.2 FE and RFE Results

For this test, the frequency estimation is of particular importance as it relates directly to the system frequency that undergoes a positive linear ramp. This test assesses the ability of the TFT algorithm to accurately track a frequency ramp occurring in the input signal. The estimated frequency and ROCOF values are computed by the one-cycle and two-cycle TFT algorithm deployed on the STM32 development board, and their results are recorded for evaluation. In Figure 6.23 , the expected change in frequency given by the reference signal in Equation 4.11, is plotted against the estimated frequency of deviation produced by the TFT algorithm.

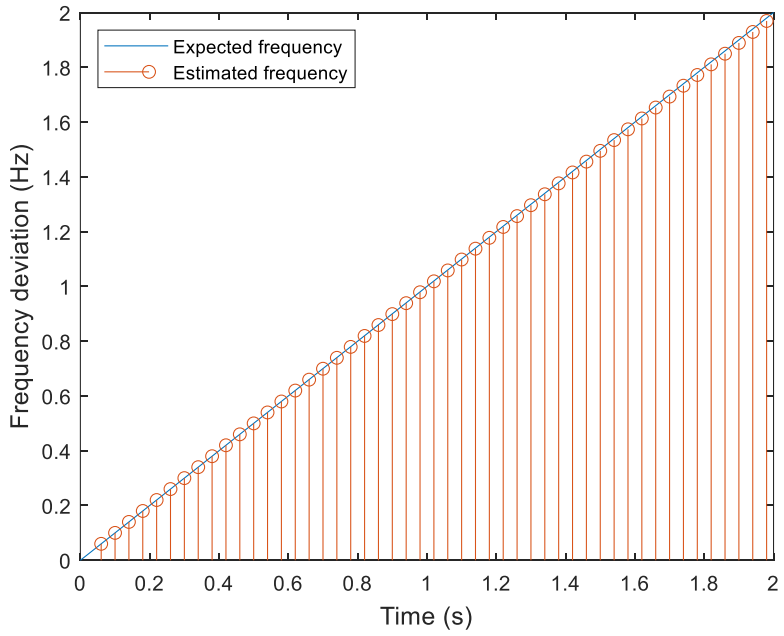

Figure 6.23: Expected and estimated frequency during frequency ramp

This illustrates the frequency estimation successfully tracking the ramping frequency of the input signal between 50 Hz to 52 Hz. The difference between the estimated and expected change in frequency yields the FE. In this manner, the FE and RFE values of the one-cycle and two-cycle TFT are calculated and presented in Figure 6.24 and Figure 6.25 respectively.

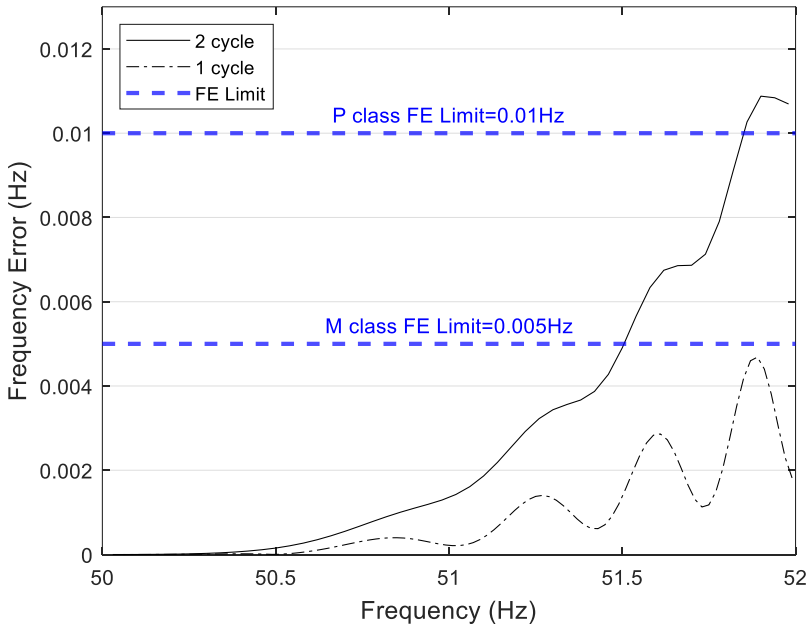

Figure 6.24: FE results of one-cycle and two-cycle TFT for frequency ramp test

In Figure 6.24, it can be seen that the one-cycle TFT estimation produces FE values below 0.005Hz, hereby satisfying the P class and M class compliance limit. The two-cycle TFT version however is only able to maintain frequency estimation errors below the P class limit of 0.01Hz, and even then this limit is exceeded as the frequency ramp reaches 51.5Hz. These results clearly indicate that the frequency estimation of the one-cycle TFT provides an improved performance in comparison with the two-cycle TFT variant, and this outcome has also been observed in the simulation tests.

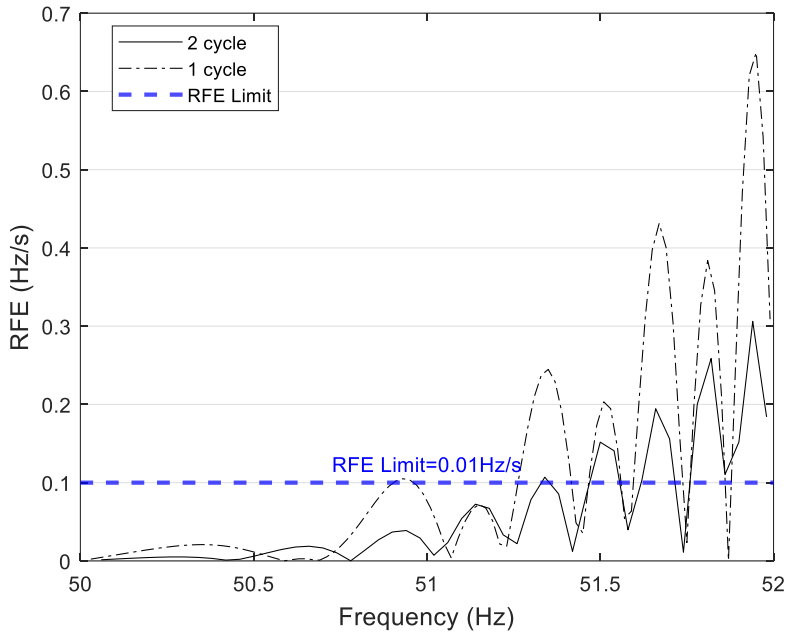

Figure 6.25: RFE results of one-cycle and two-cycle TFT for frequency ramp test

In Figure 6.25, the RFE values of the one-cycle and two-cycle TFT algorithms are presented. Here we observe the RFE results for both TFT versions satisfying the compliance limit of 0.1Hz/s for P class and M class, however this only applies to frequency values close to the nominal system frequency. It should be noted as the frequency ramp approaches 51 Hz, the RFE values tend to exceed this limit and thus no longer meet the compliance requirements. By comparison the two-cycle TFT algorithm provides slightly lower RFE values over the one-cycle TFT version.

6.10.3 Discussion of Results

The frequency ramp test results indicate that the one-cycle and two-cycle TFT algorithm offers similar phasor estimation performances, with the TVE values produced by both TFT versions easily meeting the compliance requirements for P class and M class measurements. There is a distinction observed in the frequency estimation results, whereby the FE values of the one-cycle version meets M class limits, whereas the two-cycle version only satisfies P class limits. Lastly, the ROCOF estimation results

of the one-cycle and two-cycle TFT version together follow a similar trend of RFE values which exceed the compliance limit as the frequency ramps above 51 Hz, and therefore only partially satisfies the performance requirements. Overall, the results of the STM32 development board for the frequency ramp test reflect similar outcomes as to that of the simulation tests conducted in MATLAB.

6.11 Measurement Reporting Latency Test

The measurement reporting latency test measures the total duration for a signal applied at the input of the STM32 hardware prototype to be digitised and processed by the TFT algorithm, and the estimated signal parameters to be reported. The measurement reporting latency test was conducted for a reporting rate of F_s = 10 frames per second (fps), hereby remaining consistent with the conditions prescribed in the MATLAB simulations. A continuous analog waveform is used as the input signal during the reporting latency test to ensure the processing time of the analog-to-digital conversion carried out on the STM32 development board is also included in the test. The IEEE C37.118.1-2011 standard specifies a maximum measurement reporting latency of $2/F_s$ for P class, and $5/F_s$ for M class. For our implementation these latency limits equates to a maximum duration of 200 ms for P class and 500 ms for M class. To determine if the measurement reporting latency of this hardware prototype falls within these limits, the duration from the start of the ADC conversion, to the moment the phasor estimation output is reported, is measured. This latency measurement includes the ADC sampling and conversion time, as well as the computation time of the TFT algorithm. At the start of the ADC conversion and at the end of the TFT computation, two corresponding STM32 GPIO pins are toggled in order to mark these events for measurement with an oscilloscope. Figure 6.26 and Figure 6.27 are screenshots of the oscilloscope measurements for the two-cycle and one-cycle TFT algorithm respectively, and illustrate the transitions of the GPIO pins that coincide with the start and the end of the measurement.

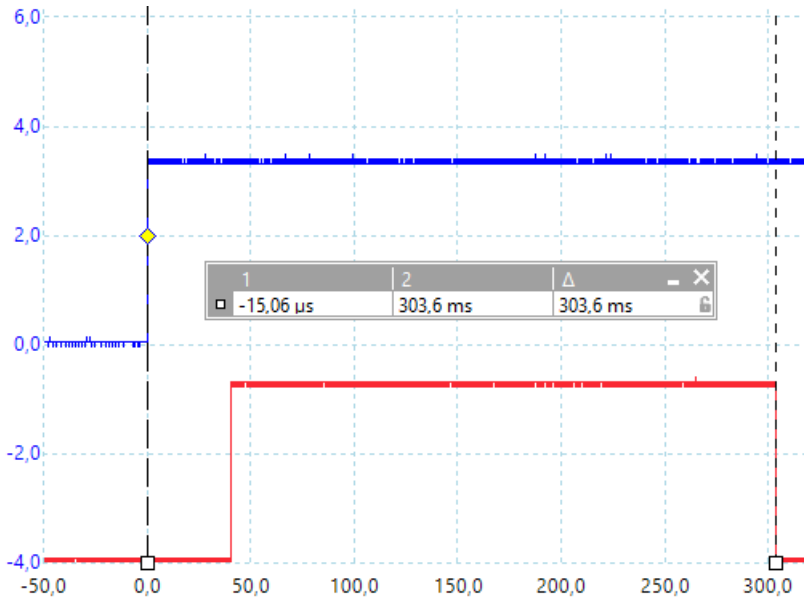

Figure 6.26: Oscilloscope measurement of reporting latency for two-cycle TFT algorithm

In Figure 6.26, the rising edge of the blue trace indicates the start of each ADC measurement. On the red trace, the falling edge indicates when the phasor estimated output is reported. The oscilloscope measurement of the time duration between these two points represents the measurement reporting latency, and results in a value of 303.6 ms. This result concludes that the measured delay of the two-cycle TFT falls within the reporting latency limits of 500 ms and successfully meets the M class requirements.

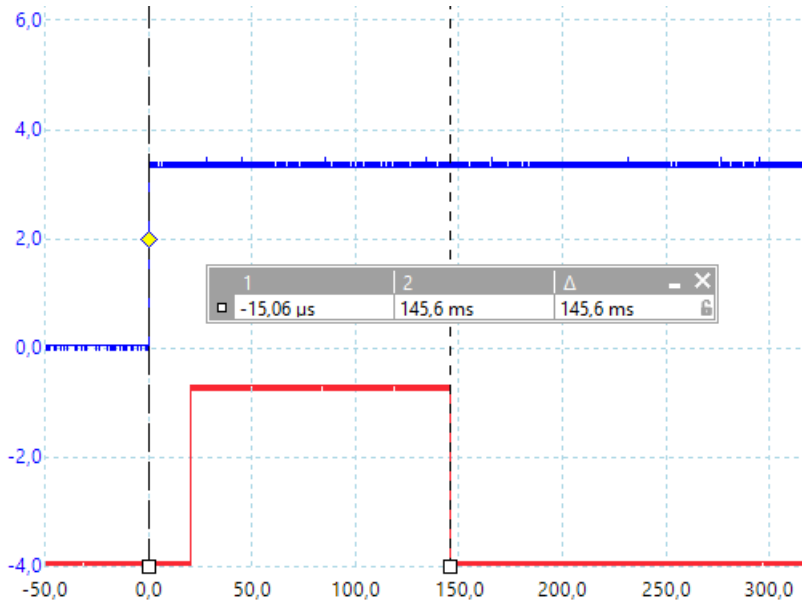

Figure 6.27: Oscilloscope measurement of reporting latency for one-cycle TFT algorithm

In Figure 6.27, the reporting latency of the one-cycle TFT is shown. It is observed that the reporting latency is reduced to a value of 145.6 ms. This is a result of the input signal only being acquired for one cycle, as well as a faster TFT computation time due to the smaller input data set. The resulting reporting latency of the one-cycle TFT is well within the limits for both P class and M class measurements and complies with the IEEE C37.118.1-2011 standard.

These measurement reporting results confirm that the time duration for acquiring the input signal and computing the phasor and frequency estimation onboard the STM32 hardware platform are within the time limits set out in the IEEE C37.118.1-2011 standard. The one-cycle TFT algorithm provides the shortest latency as expected, hereby meeting the time constraints stipulated for P class, while the two-cycle TFT algorithm complies with the requirements for M class measurements.

6.12 Conclusion

In this chapter, the experimental verification and evaluation of the phasor estimations onboard the STM32 prototype is carried out by conducting a series of IEEE C37.118.1-2011 compliance tests comprised of steady-state and dynamic conditions. These compliance tests are performed by applying a continuous analog and discrete test signal to the input of the TFT algorithm and assessing the output results.

In summary , the developed STM32 prototype has demonstrated its ability to successfully produce phasor measurements employing the TFT algorithm. The accuracy and performance of the one-cycle and two-cycle TFT algorithm were evaluated for P class and M class measurements based on the IEEE C37.118.1-2011 compliance requirements. The measurements produced by the STM32 prototype have shown to be in line with and reflect performances shown in the MATLAB simulations. Many of the compliance requirements are met, however there are instances where certain conditions were difficult to meet. The TFT algorithm has shown to perform well in dynamic conditions. Phasor parameters are successfully tracked during frequency excursions, and accuracy is maintained when subjected to amplitude and phase angle modulations.

The TFT algorithm also demonstrates its ability to suppress harmonic interference, but struggles with interharmonic and subharmonic components, and may require additional filtering for these type of scenarios. The TFT algorithm has displayed its best performances and accuracy for signal frequencies occurring at the nominal power system frequency. At off-nominal frequencies, a decline in performance and accuracy is observed. This is mainly attributed to inaccuracies incurred in the phase angle due

to the signal no longer fitting uniformly within the observation interval. For phasor estimation, the two-cycle TFT algorithm has shown a slight improvement in performance and accuracy in contrast to the one-cycle TFT algorithm, illustrating the benefit of adopting longer observation windows. The measurement latency of the one-cycle and two-cycle TFT were also found to successfully satisfy P class and M class measurements respectively. These outcomes supports the proposition of having the two-cycle TFT algorithm dedicated for M class measurements, while utilising the one-cycle TFT algorithm for P class measurements.

The effect of the ADC measurements on the phasor estimations are also examined. When a continuous analog input signal is sampled and digitised by the STM32 onboard ADC, a non-ideal sinusoidal signal with imperfections is produced and applied to the input of the TFT algorithm. This causes a degradation in the accuracy of the phasor measurement when compared to results found in the MATLAB simulations, and demonstrates the typical difference found between real world test and simulation test environments. The frequency and ROCOF estimations are found to be sensitive to any noise present in the input signal as these values are based-on derivatives. Performance and accuracy can be improved by employing front-end filtering to ensure the test signal is as close to the ideal sinusoidal signal as possible. High-end test instruments also play an important role as these determine the precision of the reference test signal, as well as ensuring the accurate measurement of input signal parameters. These factors all have an impact on the calculated performance indicators and are crucial for ensuring an accurate evaluation result.

CHAPTER SEVEN
CONCLUSION

7.1 Introduction

The final chapter of this thesis discusses the deliverables produced by this research project, and concludes with a summary of the findings, closing remarks, and recommendations for future work.

7.2 Thesis deliverables

The thesis deliverables are derived from the aim and objectives of this research project indicated in Chapter One. The following section outlines the deliverables that have been achieved during this research work.

7.2.1 Literature review

A literature review has been conducted providing essential knowledge and information required for application in this research project. The literature review establishes a theoretical basis and clear understanding of synchrophasor measurement, PMUs, the IEEE C37.118 standard, and related concepts currently within this field. The study focuses on time and frequency domain synchrophasor estimation algorithms and techniques proposed in published research papers, as well as the practical implementations thereof through appropriate hardware and software components. This lays the groundwork for the design and development of the P and M class phasor measurement prototype.

7.2.2 Theoretical framework

A theoretical framework is formulated providing a background to the mathematical principles of static and dynamic signal and phasor models, and illustrates its importance in phasor estimation. The underlying principles and fundamental operation of the second-order TFT phasor estimation algorithm are presented, encompassing matrix algebra, least squares sine fitting, and Taylor series approximation, upon which the phasor estimator is built. The relevant equations for computing the phasor and frequency parameters of a sinusoidal waveform are provided, as well as the performance indicators for evaluating the estimated results. The theoretical framework sets the stage for the MATLAB implementation of the TFT algorithm and the verification of its operation via simulation.

7.2.3 MATLAB- implementation of TFT algorithm

A MATLAB-implementation of the one-cycle and two-cycle second-order TFT algorithm is provided, adopted from a research paper presented by Khodaparast and Khederzadeh (2015). An extract of the MATLAB code is provided in Appendix A.

7.2.4 STM32 embedded source code

The embedded source code written in C for deployment on the STM32 Nucleo-F767ZI development board is provided. This consists of a ported version of the TFT algorithm, as well as the serial communication data transfer and ADC data acquisition functionality carried out onboard the STM32 development platform. An extract of the STM32 embedded C code is included in Appendix B.

7.2.5 Phasor measurement prototype

The P and M class phasor measurement prototype is one of the primary deliverables of this research project. The development of this phasor measurement device demonstrates and facilitates the following key aspects within this research project:

- The design and development of a P and M class phasor measurement device on a single hardware platform.
- The porting of the one-cycle and two-cycle TFT algorithm to C code, and the deployment thereof on a STM32 Nucleo development board.
- The verification and performance evaluation of the P and M class phasor measurements produced by the prototype in accordance with the IEEE C37.118.1-2011 standard.

7.3 Conclusions

In this section, a summary of the findings and conclusions of this research project are provided.

7.3.1 Summary of test results and findings

The compliance test results produced by the phasor measurement prototype are assessed and verified against the corresponding tests simulated in MATLAB. The two sets of results have confirmed similar performance levels for discrete test signals, however typical noise introduce during analog-to-digital conversion severely degrades the performance and accuracy of the TFT algorithm. This drawback has also been observed for out-of-band interference, highlighting the importance of front-end filtering and the suppression of unwanted interference signals when employing this phasor estimation technique.

The phasor estimation performed by the two-cycle TFT algorithm has shown to be marginally more accurate compared to that of the one-cycle TFT, given its longer observation interval. The one-cycle and two-cycle TFT algorithm has also successfully met P class and M class latency requirements respectively. These results support the prospect of having the one-cycle TFT algorithm dedicated to P class measurements,

and reserving the two-cycle TFT algorithm solely for M class measurements. A degradation in the accuracy of the frequency estimation has been observed specifically at off-nominal frequencies, revealing one of the limitations of the TFT phasor estimation technique. Overall, the performance and accuracy of the phasor measurement prototype have fulfilled many of the compliance requirements stipulated by the IEEE C37.118.1-2011 standard, however certain instances have proven to be challenging to meet.

7.3.2 Concluding remarks

This research project has successfully showcased the development of a low-cost P class and M class measurement device on a single hardware platform. The cost is primarily based on the price of the STM32 Nucleo-F767ZI development board which provides the key components required to implement the PMU prototype. The phasor measurement device successfully performs static and dynamic phasor and frequency estimations by means of the one-cycle and two-cycle TFT algorithm. Since both functions are performed on a single platform, demonstrates its flexibility and versatility by supporting two applications. This outcome successfully addresses and supports the initial research problem and hypothesis indicated in Chapter One. The concept, methodology and design employed in this research project can be adopted for developing measurement devices for various smart grid application, thereby facilitating and aiding research and experimentation in this field.

7.4 Future work and recommendations

This research project focuses on the implementation and evaluation of the TFT algorithm as a phasor estimator on the STM32 Nucleo board. This implementation could be extended to include a broader range of phasor estimation algorithms, allowing for a comparative analysis of their measurement performance and accuracy. Additional development could be undertaken to produce a fully-fledged PMU which would require additional functionality incorporating UTC synchronisation via GPS, and the ethernet data transfer of phasor measurements in compliance with the IEEE C37.118.2-2011. To accommodate for multiphase systems, the design could possibly be implemented using multiple STM32 Nucleo boards, with each device assigned to a specific phase. Another interesting research area to investigate is the development of other types of measurement devices such as Merging Units (MU) or Intelligent Electronic Devices (IED) implemented on the same STM32 hardware platform. Further research could explore alternative processing platforms for deploying these smart grid measurement devices to, taking into account measurement performance, accuracy, cost, ease of use, and development time.

REFERENCES

Adhikari, P.M., Hooshyar, H. & Vanfretti, L. 2019. Experimental Quantification of Hardware Requirements for FPGA-Based Reconfigurable PMUs. *IEEE Access*, 7: 57527–57538.

Affijulla, S. & Tripathy, P. 2018a. Development of Dictionary-Based Phasor Estimator Suitable for P-Class Phasor Measurement Unit. *IEEE Transactions on Instrumentation and Measurement*, 67(11): 2603–2615.

Affijulla, S. & Tripathy, P. 2018b. Development of Phasor Estimation Algorithm for P-Class PMU Suitable in Protection Applications. *IEEE Transactions on Smart Grid*, 9(2): 1250–1260.

Artale, G., Panzavecchia, N., Cosentino, V., Cataliotti, A., Ben-Romdhane, M., Benazza-Ben Yahia, A., Boscaino, V., Ben Othman, N., Ditta, V., Fiorino, M., Del Mastro, G., Guaiana, S., Tinè, G. & Di Cara, D. 2023. CZT-Based Harmonic Analysis in Smart Grid Using Low-Cost Electronic Measurement Boards. *Energies*, 16(10).

Avalos-Almazan, G., Castillo-Garcia, G., Paternina, M.R.A., Zamora, A., Fuentes, J.G., Rodriguez-Rodriguez, J.R. & Guillen, D. 2018. Real-Time Phasor Estimation via the Taylor-Fourier's Subspace. *North American Power Symposium*.

Belega, D. & Petri, D. 2014. A Real-Valued Taylor Weighted Least Squares Synchrophasor Estimator. *IEEE International Workshop on Applied Measurements for Power Systems Proceedings*.

Belega, D. & Petri, D. 2012a. Accuracy of synchrophasor measurements provided by the sine-fit algorithms. *IEEE International Energy Conference and Exhibition*.

Belega, D. & Petri, D. 2012b. Accuracy of the synchrophasor estimator provided by the interpolated DFT algorithm. *IEEE International Instrumentation and Measurement Technology Conference Proceedings*: 2700–2705.

Belega, D. & Petri, D. 2019. Fast Taylor Weighted Least Squares Algorithm for Synchrophasor Estimation. *IEEE 5th International forum on Research and Technology for Society and Industry*.

Belega, D. & Petri, D. 2013. Performance of Synchrophasor Measurements Provided by the Weighted Least Squares Approach. *IEEE International Instrumentation and Measurement Technology Conference*.

Bi, T., Liu, H., Feng, Q., Qian, C. & Liu, Y. 2015. Dynamic Phasor Model-Based Synchrophasor Estimation Algorithm for M-Class PMU. *IEEE Transactions on Power Delivery*, 30(3): 1162–1171.

Castello, P., Liu, J., Monti, A., Muscas, C., Pegoraro, P.A. & Ponci, F. 2014. A Fast and Accurate PMU Algorithm for P+M Class Measurement of Synchrophasor and Frequency. *IEEE Transactions on Instrumentation and Measurement*, 63(12): 2837–2845.

Castello, P., Liu, J., Monti, A., Muscas, C., Pegoraro, P.A. & Ponci, F. 2013. Toward a class "P + M" Phasor Measurement Unit. *IEEE International Workshop on Applied Measurements for Power Systems*.

City of Cape Town. 2023. *REQUIREMENTS FOR SMALL-SCALE EMBEDDED GENERATION Application and approval process for small-scale embedded generation in the City of Cape Town*. https://resource.capetown.gov.za/documentcentre/Documents/Procedures,%20guidelines%20and%20regulations/Requirements%20for%20Small-Scale%20Embedded%20Generation.pdf 3 September 2024.

Delle Femine, A., Gallo, D., Landi, C. & Luiso M. 2019. A Design Approach for a Low Cost Phasor Measurement Unit. *IEEE International Instrumentation and Measurement Technology Conference*.

Derviškadić, A., Romano, P. & Paolone, M. 2018. Iterative-Interpolated DFT for Synchrophasor Estimation: A Single Algorithm for P-and M-Class Compliant PMUs. *IEEE Transactions on Instrumentation and Measurement*, 67(3): 547–558.

Derviškadić, A., Romano, P. & Paolone, M. 2017. Iterative-Interpolated DFT for Synchrophasor Estimation in M-class compliant PMUs. *IEEE Manchester PowerTech*.

Garcia, M.C., Dotta, D., Pereira, L., de Almeida, M.C., Santos, O.L.D. & da Silva, L.C.P. 2020. Design and Development of D-PMU Module for Smart Meters. In *IEEE Power and Energy Society General Meeting*. IEEE Computer Society.

IEEE. 2018. *IEEE Standard for Digitizing Waveform Recorders. IEEE Std 1057-2017*. IEEE.

IEEE. 1995. *IEEE Standard for Synchrophasers for Power Systems. IEEE Std 1344-1995*. IEEE.

IEEE. 2014. *IEEE Standard for Synchrophasor Measurements for Power Systems. Amendment 1: Modification of Selected Performance Requirements. IEEE Std C37.118.1a-2014*.

IEEE. 2011a. *IEEE Standard for Synchrophasor Measurements for Power Systems. IEEE Std C37.118.1-2011*.

IEEE. 2011b. *IEEE Standard for Synchrophasor Data Transfer for Power Systems. IEEE Std C37.118.2-2011*.

IEEE. 2006. *IEEE Standard for Synchrophasors for Power Systems. IEEE Std C37.118-2005*. IEEE.

Jain, V.K., Collins, W.L. & Davis, D.C. 1979. High-Accuracy Analog Measurements via Interpolated FFT. *IEEE Transactions on Instrumentation and Measurement*, 28(2): 113–122.

Khodaparast, J. & Khederzadeh, M. 2015. Three-Phase Fault Detection during Power Swing by Transient Monitor. *IEEE Transactions on Power Systems*, 30(5): 2558–2565.

de la O Serna, J.A. 2007. Dynamic phasor estimates for power system oscillations. *IEEE Transactions on Instrumentation and Measurement*, 56(5): 1648–1657.

de la O Serna, J.A. 2006. Dynamic phasor estimates for power system oscillations and transient detection. *IEEE Power Engineering Society General Meeting*: 1–7.

Macii, D., Petri, D. & Zorat, A. 2012. Accuracy analysis and enhancement of DFT-based synchrophasor estimators in off-nominal conditions. *IEEE Transactions on Instrumentation and Measurement*, 61(10): 2653–2664.

MathWorks. 2024. Deploy C Code to STM32 Nucleo Using Embedded Coder. https://www.mathworks.com/videos/deploy-c-code-to-stm32-nucleo-using-embedded-coder-overview-of-workflows-1631526406921.html 14 July 2024.

Mitolo, M.A.G. 2009. *Electrical safety of low-voltage systems*. McGraw-Hill.

Monti, A., Muscas, C. & Ponci, F. 2016. *Phasor Measurement Units and Wide Area Monitoring Systems*. Elsevier. A. Monti, C. Muscas, & F. Ponci, eds.

Phadke, A.G. & Thorp, J.S. 2008. *Synchronized Phasor Measurements and Their Applications*. Springer.

Platas-Garza, M.A. & de la O Serna, J.A. 2010. Dynamic Phasor and Frequency Estimates Through Maximally Flat Differentiators. *IEEE Transactions on Instrumentation and Measurement*, 59(7): 1803–1811.

Romano, P., Paolone, M., Chau, T., Jeppesen, B. & Ahmed, E. 2017. A High-performance, Low-cost PMU Prototype for Distribution Networks based on FPGA. *IEEE Manchester PowerTech*.

RS Components. RS PRO 2205A PC Based Oscilloscope datasheet. https://docs.rs-online.com/e065/0900766b81680f25.pdf 11 August 2024.

da Silva, P.R.D.R., Prym, G.C.S., de Lima, G.P., de Carvalho, C.C.A., Dotta, D. & Martins, M.B. 2022. Proposal of Prototype of PMU Based on Dynamic Phasors. In *14th Seminar on Power Electronics and Control*. Institute of Electrical and Electronics Engineers Inc.

Steinmetz, C.P.,. 1894. Complex Quantities and their use in Electrical Engineering. *Proceedings of the International Electrical Congress*: 33–74.

STMicroelectronics. 2016. *AN4044 Application note: Floating point unit demonstration on STM32 microcontrollers*. https://www.st.com/resource/en/application_note/an4044-floating-point-unit-demonstration-on-stm32-microcontrollers-stmicroelectronics.pdf 14 July 2024.

STMicroelectronics. 2021. STM32F767xx microcontroller datasheet (DS11532). https://www.st.com/resource/en/datasheet/stm32f767zi.pdf 14 July 2024.

APPENDICES

Appendix A: MATLAB code of one-cycle and two-cycle TFT algorithm

```
%% One-cycle TFT algorithm %%
function
[amplitude,phaz,amplitudep,phazp,amplitudez,phazz,P]=phasor_estimation_method4_1cy
cle(N1,f0,S)

B=zeros((1*N1+1),6);
size_B = size(B)
Nh=N1/2;
w1=2*pi/N1;
delT=1/(f0*N1);
size_B = size(B)

for n=-8:8
    B(n+8+1,1)=(n^2)*exp(1i*n*w1);
    B(n+8+1,2)=(n)*exp(1i*n*w1);
    B(n+8+1,3)=(1)*exp(1i*n*w1);
    B(n+8+1,4)=(1)*exp(-1i*n*w1);
    B(n+8+1,5)=(n)*exp(-1i*n*w1);
    B(n+8+1,6)=(n^2)*exp(-1i*n*w1);
end

teta=2*pi/N1;
SN1=zeros((1*N1+1),1);
SN1(1:(1*N1+1))=S(1:(1*N1+1));
size_SN1 =size(SN1)
size_S = size(S)
Phat(:,1)=(inv(B'*B)*B'*SN1);
Phat_size = size(Phat)

ahat=2*abs(Phat(3,1));
phihat=atan2(imag(Phat(3,1)),real(Phat(3,1)));

Phat(2,1)=Phat(2,1)/delT;
ahatp=2*real(Phat(2,1)*exp(-1i*phihat(1,1)));
phihatp=(2/ahat(1,1))*imag(Phat(2,1)*exp(-1i*phihat(1,1)));

Phat(1,1)=Phat(1,1)/(delT^2);
ahatz=4*real(Phat(1,1)*exp(-1i*phihat(1,1)))+(ahat(1,1)*((phihatp(1,1))^2));
phihatz=((4*imag(Phat(1,1)*exp(-1i*phihat(1,1))))-
(2*ahatp(1,1)*phihatp(1,1)))/ahat(1,1);

amplitude=ahat;
phaz=phihat-pi;

amplitudep=ahatp;
phazp=phihatp;

amplitudez=ahatz;
phazz=phihatz;
P=Phat;

B_size = size(B)
S_size = size(S)
SN1_size = size(SN1)
Phat_size = size(Phat)
SNN = SN1
```

```
frequency = phihatp/(2*pi)
rocof = phihatz/(2*pi)

%% Two-cycle TFT algorithm %%
function
[amplitude,phaz,amplitudep,phazp,amplitudez,phazz,P]=phasor_estimation_method4(N1,
f0,S)

B=(zeros((2*N1+1),6));
Nh=N1;
w1=2*pi/N1;
delT=1/(f0*N1);

for n=-Nh:Nh
    B(n+Nh+1,1)=(n^2)*exp(1i*n*w1);
    B(n+Nh+1,2)=(n)*exp(1i*n*w1);
    B(n+Nh+1,3)=(1)*exp(1i*n*w1);
    B(n+Nh+1,4)=(1)*exp(-1i*n*w1);
    B(n+Nh+1,5)=(n)*exp(-1i*n*w1);
    B(n+Nh+1,6)=(n^2)*exp(-1i*n*w1);
end

teta=2*pi/N1;
SN1=(zeros((2*N1+1),1));
SN1(1:(2*N1+1))=S(1:(2*N1+1));
Phat(:,1)=(inv(B'*B)*B'*SN1);
Phat_size = size(Phat)

ahat=2*abs(Phat(3,1));
phihat=atan2(imag(Phat(3,1)),real(Phat(3,1)));

Phat(2,1)=Phat(2,1)/delT;
ahatp=2*real(Phat(2,1)*exp(-1i*phihat(1,1)));
phihatp=(2/ahat(1,1))*imag(Phat(2,1)*exp(-1i*phihat(1,1)));

Phat(1,1)=Phat(1,1)/(delT^2);
ahatz=4*real(Phat(1,1)*exp(-1i*phihat(1,1)))+(ahat(1,1)*((phihatp(1,1))^2));
phihatz=((4*imag(Phat(1,1)*exp(-1i*phihat(1,1))))-
(2*ahatp(1,1)*phihatp(1,1)))/ahat(1,1);

amplitude=ahat;
phaz=phihat;

amplitudep=ahatp;
phazp=phihatp;

amplitudez=ahatz;
phazz=phihatz;
P=Phat;

B_size = size(B)
S_size = size(S)
SN1_size = size(SN1)
Phat_size = size(Phat)
frequency = phihatp/(2*pi)
rocof = phihatz/(2*pi)
```

Appendix B: STM32 embedded source code

```
//============================================================================//
/* main.c file */
//============================================================================//

#include "main.h"
#include "phasor_estimation.h"
#include "rtwtypes.h"
#include <string.h>
#include <stdio.h>
#include <stdbool.h>

#define ADC_BUF_LEN 641 //1-cycle
//#define ADC_BUF_LEN 1281 //2-cycle
#define M_CNT 50    // set amount of estimation outputs stored in array
#define ADC_CNT 50  // set number of ADC measurements triggered by 500mS timer

ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim3;

UART_HandleTypeDef huart3;

PCD_HandleTypeDef hpcd_USB_OTG_FS;

/* Global Variables */
uint16_t adc_buf[ADC_BUF_LEN];
float adc_store[ADC_BUF_LEN];
uint16_t m_cnt=0;
uint16_t adc_cnt=0;
int global_tb2, global_tb1;
int t2, t1;
int proc_time=0;
int global_proc_time_b=0;
float amplitude_store[M_CNT];
float phase_store[M_CNT];
float freq_store[M_CNT];
float rocof_store[M_CNT];
uint16_t j=0;

//******Load discrete signal from file******//
const float input1281[] = {
             #include "test_signal_2.txt"
             };

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_DMA_Init(void);
static void MX_TIM1_Init(void);
static void MX_ADC1_Init(void);
static void MX_USB_OTG_FS_PCD_Init(void);
static void MX_TIM3_Init(void);

int main(void)
{
```

```
  HAL_Init();

  phasor_estimation_initialize();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_USART3_UART_Init();
  MX_DMA_Init();
  MX_TIM1_Init();
  MX_ADC1_Init();
  MX_USB_OTG_FS_PCD_Init();
  MX_TIM3_Init();

  HAL_Delay(1000);
  HAL_TIM_Base_Start(&htim1); // start timer
  HAL_TIM_OC_Start(&htim1, TIM_CHANNEL_1); //start debug line ie. when timer
triggers adc
  HAL_Delay(500);
  HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buf, ADC_BUF_LEN); //enable adc via
dma, clocked by compare register
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_8);// this is to sync with oscilloscope
measurement
  // Start timer
  HAL_TIM_Base_Start_IT(&htim3);

  char msg[100]; // variable for printing data
  double amplitude, amplitudep, amplitudez, phat3re, phat3im;
  double phaz, phazp, phazz, freq;

  rtU.f0 = 50;
  rtU.N1 = 640;

  int loop_cnt = 0;

  while (1)
  {

      loop_cnt++; //counter for while loop
        HAL_Delay(100);
        HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);//toggle red led
        HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_15);

        //**********PRINT STORED ADC OUTPUT**********//
        sprintf(msg, "Converted ADC value\r\n");
        HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 2000);
        for (int i = 0; i < ADC_BUF_LEN; i++)
        {
             sprintf(msg, "%f ", adc_store[i]);
             HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);
        }
        sprintf(msg, "------------------\r\n");
        HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 2000);

        sprintf(msg, "m_cnt = %d\r\n", m_cnt);
        HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);

        for (int i = 0; i < M_CNT; i++) // print 50 amplitude outputs
        {
```

```
                sprintf(msg, "amplitude_store(%d) = %f\r\n", i+1,
amplitude_store[i]);
                HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);
        }

        sprintf(msg, "\r\n");
        HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);

        for (int i = 0; i < M_CNT; i++) // print 50 phase angle outputs
        {
                sprintf(msg, "phase_store(%d) = %f\r\n", i+1, phase_store[i]);
                HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);
        }
        sprintf(msg, "\r\n");
        HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);

        for (int i = 0; i < M_CNT; i++) // print 50 frequency outputs
        {
                sprintf(msg, "freq_store(%d) = %f\r\n", i+1, freq_store[i]);
                HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);
        }
        sprintf(msg, "\r\n");
        HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);

        for (int i = 0; i < M_CNT; i++) // print 50 rocof outputs
        {
                sprintf(msg, "rocof_store(%d) = %f\r\n", i+1, rocof_store[i]);
                HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);
        }
        sprintf(msg, "\r\n");
        HAL_UART_Transmit(&huart3, (uint8_t *)msg, strlen(msg), 1000);

  }

}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET); //SET HIGH AT START
        global_tb1=0;
        global_tb2=0;
        global_tb1 = HAL_GetTick(); //Provides a tick value in millisecond.

        int array_length = ADC_BUF_LEN; // copy adc values to store and tft input

        for (int i = 0; i < array_length; i++)
        {
                adc_store[i] = (adc_buf[i]*(3.3/4096))-0.98;
                rtU.S[i] = adc_store[i]; // copy store to tft input
        }

        j+=640; //1-cycle
        //j+=1280; //2-cycle

        phasor_estimation_step();

        global_tb2 = HAL_GetTick();
        global_proc_time_b = global_tb2-global_tb1;
```

```
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_RESET); // set low when
complete

	if (m_cnt < M_CNT)

	{
		amplitude_store[m_cnt] = rtY.ahat; // store each measurement in
array
		phase_store[m_cnt] = rtY.phihat-3.141592653;
		freq_store[m_cnt] = rtY.phihatp/(2*3.141592653);
		rocof_store[m_cnt] = rtY.phihatz/(2*3.141592653);
		//HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buf, ADC_BUF_LEN); //re-
enable adc via dma
		m_cnt++;
		HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_7);//toggle blue led - PB7
	}

}
//ISR executes every 500mS. Starts ADC via DMA and toggles pin
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
	if (htim==&htim3)
	{
		if (adc_cnt < ADC_CNT) // remove to continuously re-enable adc.
		{
			HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buf, ADC_BUF_LEN);
//re-enable adc via dma
		    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0); //green led
		    adc_cnt++;
		}
	}
}

//=========================================================================//
/* phasor_estiamtion.c file */
//=========================================================================//


#include "phasor_estimation.h"
#define NumBitsPerChar                 8U

/* Block signals and states (default storage) */
DW rtDW;

/* External inputs (root inport signals with default storage) */
ExtU rtU;

/* External outputs (root outports fed by signals with default storage) */
ExtY rtY;

/* Real-time model */
static RT_MODEL rtM_;
RT_MODEL *const rtM = &rtM_;
extern real_T rt_hypotd_snf(real_T u0, real_T u1);
extern real_T rt_atan2d_snf(real_T u0, real_T u1);

/* Forward declaration for local functions */
static void inv(const creal_T x[36], creal_T y[36]);
static real_T rtGetNaN(void);
```

```
static real32_T rtGetNaNF(void);

/*===========*
 * Constants *
 *===========*/
#define RT_PI                          3.14159265358979323846
#define RT_PIF                         3.1415927F
#define RT_LN_10                       2.30258509299404568402
#define RT_LN_10F                      2.3025851F
#define RT_LOG10E                      0.43429448190325182765
#define RT_LOG10EF                     0.43429449F
#define RT_E                           2.7182818284590452354
#define RT_EF                          2.7182817F

/*
 * UNUSED_PARAMETER(x)
 *   Used to specify that a function parameter (argument) is required but not
 *   accessed by the function body.
 */
#ifndef UNUSED_PARAMETER
#if defined(__LCC__)
#define UNUSED_PARAMETER(x)                                   /* do nothing */
#else

/*
 * This is the semi-ANSI standard way of indicating that an
 * unused function parameter is required.
 */
#define UNUSED_PARAMETER(x)            (void) (x)
#endif
#endif

extern real_T rtInf;
extern real_T rtMinusInf;
extern real_T rtNaN;
extern real32_T rtInfF;
extern real32_T rtMinusInfF;
extern real32_T rtNaNF;
static void rt_InitInfAndNaN(size_t realSize);
static boolean_T rtIsInf(real_T value);
static boolean_T rtIsInfF(real32_T value);
static boolean_T rtIsNaN(real_T value);
static boolean_T rtIsNaNF(real32_T value);
typedef struct {
  struct {
    uint32_T wordH;
    uint32_T wordL;
  } words;
} BigEndianIEEEDouble;

typedef struct {
  struct {
    uint32_T wordL;
    uint32_T wordH;
  } words;
} LittleEndianIEEEDouble;

typedef struct {
  union {
```

```
    real32_T wordLreal;
    uint32_T wordLuint;
  } wordL;
} IEEESingle;

real_T rtInf;
real_T rtMinusInf;
real_T rtNaN;
real32_T rtInfF;
real32_T rtMinusInfF;
real32_T rtNaNF;
static real_T rtGetInf(void);
static real32_T rtGetInfF(void);
static real_T rtGetMinusInf(void);
static real32_T rtGetMinusInfF(void);

/*
 * Initialize rtNaN needed by the generated code.
 * NaN is initialized as non-signaling. Assumes IEEE.
 */
static real_T rtGetNaN(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T nan = 0.0;
  if (bitsPerReal == 32U) {
    nan = rtGetNaNF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0xFFF80000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    nan = tmpVal.fltVal;
  }

  return nan;
}

/*
 * Initialize rtNaNF needed by the generated code.
 * NaN is initialized as non-signaling. Assumes IEEE.
 */
static real32_T rtGetNaNF(void)
{
  IEEESingle nanF = { { 0.0F } };

  nanF.wordL.wordLuint = 0xFFC00000U;
  return nanF.wordL.wordLreal;
}

/*
 * Initialize the rtInf, rtMinusInf, and rtNaN needed by the
 * generated code. NaN is initialized as non-signaling. Assumes IEEE.
 */
static void rt_InitInfAndNaN(size_t realSize)
{
  (void) (realSize);
```

```
  rtNaN = rtGetNaN();
  rtNaNF = rtGetNaNF();
  rtInf = rtGetInf();
  rtInfF = rtGetInfF();
  rtMinusInf = rtGetMinusInf();
  rtMinusInfF = rtGetMinusInfF();
}

/* Test if value is infinite */
static boolean_T rtIsInf(real_T value)
{
  return (boolean_T)((value==rtInf || value==rtMinusInf) ? 1U : 0U);
}

/* Test if single-precision value is infinite */
static boolean_T rtIsInfF(real32_T value)
{
  return (boolean_T)(((value)==rtInfF || (value)==rtMinusInfF) ? 1U : 0U);
}

/* Test if value is not a number */
static boolean_T rtIsNaN(real_T value)
{
  boolean_T result = (boolean_T) 0;
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  if (bitsPerReal == 32U) {
    result = rtIsNaNF((real32_T)value);
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.fltVal = value;
    result = (boolean_T)((tmpVal.bitVal.words.wordH & 0x7FF00000) == 0x7FF00000 &&
                         ( (tmpVal.bitVal.words.wordH & 0x000FFFFF) != 0 ||
                          (tmpVal.bitVal.words.wordL != 0) ));
  }

  return result;
}

/* Test if single-precision value is not a number */
static boolean_T rtIsNaNF(real32_T value)
{
  IEEESingle tmp;
  tmp.wordL.wordLreal = value;
  return (boolean_T)( (tmp.wordL.wordLuint & 0x7F800000) == 0x7F800000 &&
                      (tmp.wordL.wordLuint & 0x007FFFFF) != 0 );
}

/*
 * Initialize rtInf needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
static real_T rtGetInf(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T inf = 0.0;
```

```
  if (bitsPerReal == 32U) {
    inf = rtGetInfF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0x7FF00000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    inf = tmpVal.fltVal;
  }

  return inf;
}

/*
 * Initialize rtInfF needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
static real32_T rtGetInfF(void)
{
  IEEESingle infF;
  infF.wordL.wordLuint = 0x7F800000U;
  return infF.wordL.wordLreal;
}

/*
 * Initialize rtMinusInf needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
static real_T rtGetMinusInf(void)
{
  size_t bitsPerReal = sizeof(real_T) * (NumBitsPerChar);
  real_T minf = 0.0;
  if (bitsPerReal == 32U) {
    minf = rtGetMinusInfF();
  } else {
    union {
      LittleEndianIEEEDouble bitVal;
      real_T fltVal;
    } tmpVal;

    tmpVal.bitVal.words.wordH = 0xFFF00000U;
    tmpVal.bitVal.words.wordL = 0x00000000U;
    minf = tmpVal.fltVal;
  }

  return minf;
}

/*
 * Initialize rtMinusInfF needed by the generated code.
 * Inf is initialized as non-signaling. Assumes IEEE.
 */
static real32_T rtGetMinusInfF(void)
{
  IEEESingle minfF;
  minfF.wordL.wordLuint = 0xFF800000U;
```

```
  return minfF.wordL.wordLreal;
}

/* Function for MATLAB Function: '<Root>/MATLAB Function' */
static void inv(const creal_T x[36], creal_T y[36])
{
  creal_T A[36];
  real_T A_im_tmp;
  real_T A_im_tmp_0;
  real_T bim;
  real_T br;
  real_T smax;
  real_T temp_im;
  int32_T b_ix;
  int32_T c_j;
  int32_T d_k;
  int32_T ijA;
  int32_T ix;
  int32_T iy;
  int32_T jj;
  int8_T ipiv[6];
  int8_T p[6];
  for (b_ix = 0; b_ix < 36; b_ix++) {
    A[b_ix] = x[b_ix];
    y[b_ix].re = 0.0;
    y[b_ix].im = 0.0;
  }

  for (b_ix = 0; b_ix < 6; b_ix++) {
    ipiv[b_ix] = (int8_T)(b_ix + 1);
  }

  for (c_j = 0; c_j < 5; c_j++) {
    jj = c_j * 7;
    iy = 0;
    ix = jj;
    smax = fabs(A[jj].re) + fabs(A[jj].im);
    for (d_k = 2; d_k <= 6 - c_j; d_k++) {
      ix++;
      A_im_tmp_0 = fabs(A[ix].re) + fabs(A[ix].im);
      if (A_im_tmp_0 > smax) {
        iy = d_k - 1;
        smax = A_im_tmp_0;
      }
    }

    b_ix = jj + iy;
    if ((A[b_ix].re != 0.0) || (A[b_ix].im != 0.0)) {
      if (iy != 0) {
        iy += c_j;
        ipiv[c_j] = (int8_T)(iy + 1);
        for (d_k = 0; d_k < 6; d_k++) {
          ix = d_k * 6 + c_j;
          smax = A[ix].re;
          temp_im = A[ix].im;
          b_ix = d_k * 6 + iy;
          A[ix] = A[b_ix];
          A[b_ix].re = smax;
          A[iy + d_k * 6].im = temp_im;
```

```
    }
  }

  iy = (jj - c_j) + 6;
  for (ix = jj + 1; ix < iy; ix++) {
    smax = A[ix].re;
    temp_im = A[ix].im;
    br = A[jj].re;
    A_im_tmp = A[jj].im;
    if (A_im_tmp == 0.0) {
      if (temp_im == 0.0) {
        A[ix].re = smax / br;
        A[ix].im = 0.0;
      } else if (smax == 0.0) {
        A[ix].re = 0.0;
        A[ix].im = temp_im / br;
      } else {
        A[ix].re = smax / br;
        A[ix].im = temp_im / br;
      }
    } else if (br == 0.0) {
      if (smax == 0.0) {
        A[ix].re = temp_im / A_im_tmp;
        A[ix].im = 0.0;
      } else if (temp_im == 0.0) {
        A[ix].re = 0.0;
        A[ix].im = -(smax / A_im_tmp);
      } else {
        A[ix].re = temp_im / A_im_tmp;
        A[ix].im = -(smax / A_im_tmp);
      }
    } else {
      A_im_tmp_0 = fabs(br);
      bim = fabs(A_im_tmp);
      if (A_im_tmp_0 > bim) {
        A_im_tmp_0 = A_im_tmp / br;
        A_im_tmp = A_im_tmp_0 * A_im_tmp + br;
        A[ix].re = (A_im_tmp_0 * temp_im + smax) / A_im_tmp;
        A[ix].im = (temp_im - A_im_tmp_0 * smax) / A_im_tmp;
      } else if (bim == A_im_tmp_0) {
        br = br > 0.0 ? 0.5 : -0.5;
        A_im_tmp = A_im_tmp > 0.0 ? 0.5 : -0.5;
        A[ix].re = (smax * br + temp_im * A_im_tmp) / A_im_tmp_0;
        A[ix].im = (temp_im * br - smax * A_im_tmp) / A_im_tmp_0;
      } else {
        A_im_tmp_0 = br / A_im_tmp;
        A_im_tmp += A_im_tmp_0 * br;
        A[ix].re = (A_im_tmp_0 * smax + temp_im) / A_im_tmp;
        A[ix].im = (A_im_tmp_0 * temp_im - smax) / A_im_tmp;
      }
    }
  }
}

iy = jj;
ix = jj + 6;
for (d_k = 0; d_k <= 4 - c_j; d_k++) {
  if ((A[ix].re != 0.0) || (A[ix].im != 0.0)) {
    int32_T d;
```

```
      smax = -A[ix].re - A[ix].im * 0.0;
      temp_im = A[ix].re * 0.0 + -A[ix].im;
      b_ix = jj + 1;
      ijA = iy + 7;
      d = (iy - c_j) + 12;
      while (ijA + 1 <= d) {
        A_im_tmp = A[b_ix].re;
        A_im_tmp_0 = A[b_ix].im;
        A[ijA].re += A_im_tmp * smax - A_im_tmp_0 * temp_im;
        A[ijA].im += A_im_tmp * temp_im + A_im_tmp_0 * smax;
        b_ix++;
        ijA++;
      }
    }

    ix += 6;
    iy += 6;
  }
}

for (b_ix = 0; b_ix < 6; b_ix++) {
  p[b_ix] = (int8_T)(b_ix + 1);
}

for (c_j = 0; c_j < 5; c_j++) {
  int8_T ipiv_0;
  ipiv_0 = ipiv[c_j];
  if (ipiv_0 > c_j + 1) {
    jj = p[ipiv_0 - 1];
    p[ipiv_0 - 1] = p[c_j];
    p[c_j] = (int8_T)jj;
  }
}

for (c_j = 0; c_j < 6; c_j++) {
  jj = p[c_j] - 1;
  b_ix = 6 * jj + c_j;
  y[b_ix].re = 1.0;
  y[b_ix].im = 0.0;
  for (iy = c_j; iy + 1 < 7; iy++) {
    b_ix = 6 * jj + iy;
    if ((y[b_ix].re != 0.0) || (y[b_ix].im != 0.0)) {
      for (ix = iy + 1; ix + 1 < 7; ix++) {
        ijA = 6 * iy + ix;
        smax = A[ijA].re;
        temp_im = A[ijA].im;
        A_im_tmp = y[b_ix].re * temp_im + y[b_ix].im * smax;
        d_k = 6 * jj + ix;
        y[d_k].re -= y[b_ix].re * smax - y[b_ix].im * temp_im;
        y[d_k].im -= A_im_tmp;
      }
    }
  }
}

for (c_j = 0; c_j < 6; c_j++) {
  jj = 6 * c_j;
  for (iy = 5; iy >= 0; iy--) {
    ix = 6 * iy;
```

```
b_ix = iy + jj;
if ((y[b_ix].re != 0.0) || (y[b_ix].im != 0.0)) {
  smax = y[b_ix].re;
  temp_im = y[b_ix].im;
  br = A[iy + ix].re;
  A_im_tmp = A[iy + ix].im;
  if (A_im_tmp == 0.0) {
    if (temp_im == 0.0) {
      b_ix = iy + jj;
      y[b_ix].re = smax / br;
      y[b_ix].im = 0.0;
    } else if (smax == 0.0) {
      b_ix = iy + jj;
      y[b_ix].re = 0.0;
      y[b_ix].im = temp_im / br;
    } else {
      b_ix = iy + jj;
      y[b_ix].re = smax / br;
      y[b_ix].im = temp_im / br;
    }
  } else if (br == 0.0) {
    if (smax == 0.0) {
      b_ix = iy + jj;
      y[b_ix].re = temp_im / A_im_tmp;
      y[b_ix].im = 0.0;
    } else if (temp_im == 0.0) {
      b_ix = iy + jj;
      y[b_ix].re = 0.0;
      y[b_ix].im = -(smax / A_im_tmp);
    } else {
      b_ix = iy + jj;
      y[b_ix].re = temp_im / A_im_tmp;
      y[b_ix].im = -(smax / A_im_tmp);
    }
  } else {
    A_im_tmp_0 = fabs(br);
    bim = fabs(A_im_tmp);
    if (A_im_tmp_0 > bim) {
      A_im_tmp_0 = A_im_tmp / br;
      A_im_tmp = A_im_tmp_0 * A_im_tmp + br;
      b_ix = iy + jj;
      y[b_ix].re = (A_im_tmp_0 * temp_im + smax) / A_im_tmp;
      y[b_ix].im = (temp_im - A_im_tmp_0 * smax) / A_im_tmp;
    } else if (bim == A_im_tmp_0) {
      br = br > 0.0 ? 0.5 : -0.5;
      A_im_tmp = A_im_tmp > 0.0 ? 0.5 : -0.5;
      b_ix = iy + jj;
      y[b_ix].re = (smax * br + temp_im * A_im_tmp) / A_im_tmp_0;
      y[b_ix].im = (temp_im * br - smax * A_im_tmp) / A_im_tmp_0;
    } else {
      A_im_tmp_0 = br / A_im_tmp;
      A_im_tmp += A_im_tmp_0 * br;
      b_ix = iy + jj;
      y[b_ix].re = (A_im_tmp_0 * smax + temp_im) / A_im_tmp;
      y[b_ix].im = (A_im_tmp_0 * temp_im - smax) / A_im_tmp;
    }
  }

  for (d_k = 0; d_k < iy; d_k++) {
```

```
          ijA = iy + jj;
          b_ix = d_k + ix;
          smax = y[ijA].re;
          temp_im = A[b_ix].re;
          A_im_tmp = y[ijA].im;
          A_im_tmp_0 = A[b_ix].im;
          b_ix = d_k + jj;
          y[b_ix].re -= smax * temp_im - A_im_tmp * A_im_tmp_0;
          y[b_ix].im -= smax * A_im_tmp_0 + A_im_tmp * temp_im;
        }
      }
    }
  }
}

real_T rt_hypotd_snf(real_T u0, real_T u1)
{
  real_T a;
  real_T y;
  a = fabs(u0);
  y = fabs(u1);
  if (a < y) {
    a /= y;
    y *= sqrt(a * a + 1.0);
  } else if (a > y) {
    y /= a;
    y = sqrt(y * y + 1.0) * a;
  } else if (!rtIsNaN(y)) {
    y = a * 1.4142135623730951;
  }

  return y;
}

real_T rt_atan2d_snf(real_T u0, real_T u1)
{
  real_T y;
  if (rtIsNaN(u0) || rtIsNaN(u1)) {
    y = (rtNaN);
  } else if (rtIsInf(u0) && rtIsInf(u1)) {
    int32_T u0_0;
    int32_T u1_0;
    if (u0 > 0.0) {
      u0_0 = 1;
    } else {
      u0_0 = -1;
    }

    if (u1 > 0.0) {
      u1_0 = 1;
    } else {
      u1_0 = -1;
    }

    y = atan2(u0_0, u1_0);
  } else if (u1 == 0.0) {
    if (u0 > 0.0) {
      y = RT_PI / 2.0;
    } else if (u0 < 0.0) {
```

```
      y = -(RT_PI / 2.0);
    } else {
      y = 0.0;
    }
  } else {
    y = atan2(u0, u1);
  }

  return y;
}

/* Model step function */
void phasor_estimation_step(void)
{
  creal_T B_0[36];
  creal_T tmp[36];
  creal_T rtb_Phat[6];
  real_T B_im;
  real_T c_r;
  real_T d_r;
  real_T delT;
  real_T re_tmp;
  real_T re_tmp_0;
  real_T re_tmp_1;
  real_T rtb_Phat_im;
  real_T w1;
  int32_T B_re_tmp;
  int32_T b;
  int32_T b_n;
  int32_T i;
  int32_T tmp_0;

  /* MATLAB Function: '<Root>/MATLAB Function' incorporates:
   *  Inport: '<Root>/N1'
   *  Inport: '<Root>/S'
   *  Inport: '<Root>/f0'
   */
  memset(&rtDW.B_m[0], 0, 7686U * sizeof(creal_T));
  w1 = 6.2831853071795862 / rtU.N1;
  delT = 1.0 / (rtU.f0 * rtU.N1);
  b = (int32_T)((1.0 - (-rtU.N1)) + rtU.N1);
  for (b_n = 0; b_n < b; b_n++) {
    B_im = -rtU.N1 + (real_T)b_n;
    re_tmp_1 = w1 * B_im;
    if (re_tmp_1 == 0.0) {
      re_tmp = exp(B_im * 0.0 * w1);
      re_tmp_0 = 0.0;
    } else {
      c_r = exp(B_im * 0.0 * w1 / 2.0);
      re_tmp = c_r * cos(re_tmp_1) * c_r;
      re_tmp_0 = c_r * sin(re_tmp_1) * c_r;
    }

    c_r = B_im * B_im;
    i = (int32_T)((B_im + rtU.N1) + 1.0);
    rtDW.B_m[i - 1].re = c_r * re_tmp;
    rtDW.B_m[i - 1].im = c_r * re_tmp_0;
    if (re_tmp_1 == 0.0) {
      re_tmp = exp(B_im * 0.0 * w1);
```

```
    re_tmp_0 = 0.0;
  } else {
    c_r = exp(B_im * 0.0 * w1 / 2.0);
    re_tmp = c_r * cos(re_tmp_1) * c_r;
    re_tmp_0 = c_r * sin(re_tmp_1) * c_r;
  }

  rtDW.B_m[i + 1280].re = B_im * re_tmp;
  rtDW.B_m[i + 1280].im = B_im * re_tmp_0;
  if (re_tmp_1 == 0.0) {
    rtDW.B_m[i + 2561].re = exp(B_im * 0.0 * w1);
    rtDW.B_m[(int32_T)((B_im + rtU.N1) + 1.0) + 2561].im = 0.0;
  } else {
    c_r = exp(B_im * 0.0 * w1 / 2.0);
    i = (int32_T)((B_im + rtU.N1) + 1.0) + 2561;
    rtDW.B_m[i].re = c_r * cos(re_tmp_1) * c_r;
    rtDW.B_m[i].im = c_r * sin(re_tmp_1) * c_r;
  }

  re_tmp_1 = w1 * -B_im;
  if (re_tmp_1 == 0.0) {
    i = (int32_T)((B_im + rtU.N1) + 1.0) + 3842;
    rtDW.B_m[i].re = exp(B_im * 0.0 * w1);
    rtDW.B_m[i].im = 0.0;
  } else {
    c_r = exp(B_im * 0.0 * w1 / 2.0);
    i = (int32_T)((B_im + rtU.N1) + 1.0) + 3842;
    rtDW.B_m[i].re = c_r * cos(re_tmp_1) * c_r;
    rtDW.B_m[i].im = c_r * sin(re_tmp_1) * c_r;
  }

  if (re_tmp_1 == 0.0) {
    re_tmp = exp(B_im * 0.0 * w1);
    re_tmp_0 = 0.0;
  } else {
    c_r = exp(B_im * 0.0 * w1 / 2.0);
    re_tmp = c_r * cos(re_tmp_1) * c_r;
    re_tmp_0 = c_r * sin(re_tmp_1) * c_r;
  }

  i = (int32_T)((B_im + rtU.N1) + 1.0);
  rtDW.B_m[i + 5123].re = B_im * re_tmp;
  rtDW.B_m[i + 5123].im = B_im * re_tmp_0;
  if (re_tmp_1 == 0.0) {
    re_tmp = exp(B_im * 0.0 * w1);
    re_tmp_0 = 0.0;
  } else {
    c_r = exp(B_im * 0.0 * w1 / 2.0);
    re_tmp = c_r * cos(re_tmp_1) * c_r;
    re_tmp_0 = c_r * sin(re_tmp_1) * c_r;
  }

  c_r = B_im * B_im;
  rtDW.B_m[i + 6404].re = c_r * re_tmp;
  rtDW.B_m[i + 6404].im = c_r * re_tmp_0;
}

for (i = 0; i < 6; i++) {
  for (b = 0; b < 6; b++) {
```

```
    b_n = 6 * b + i;
    B_0[b_n].re = 0.0;
    B_0[b_n].im = 0.0;
  }
}

for (i = 0; i < 6; i++) {
  for (b = 0; b < 6; b++) {
    for (b_n = 0; b_n < 1281; b_n++) {
      B_re_tmp = 1281 * i + b_n;
      w1 = rtDW.B_m[B_re_tmp].re;
      B_im = -rtDW.B_m[B_re_tmp].im;
      B_re_tmp = 1281 * b + b_n;
      tmp_0 = 6 * b + i;
      c_r = rtDW.B_m[B_re_tmp].im;
      re_tmp = rtDW.B_m[B_re_tmp].re;
      B_0[tmp_0].re += re_tmp * w1 - c_r * B_im;
      B_0[tmp_0].im += c_r * w1 + re_tmp * B_im;
    }
  }
}

inv(B_0, tmp);
for (i = 0; i < 6; i++) {
  for (b = 0; b < 1281; b++) {
    b_n = 6 * b + i;
    rtDW.dcv[b_n].re = 0.0;
    rtDW.dcv[b_n].im = 0.0;
  }
}

for (i = 0; i < 6; i++) {
  for (b = 0; b < 1281; b++) {
    for (b_n = 0; b_n < 6; b_n++) {
      B_re_tmp = 1281 * b_n + b;
      w1 = rtDW.B_m[B_re_tmp].re;
      B_im = -rtDW.B_m[B_re_tmp].im;
      B_re_tmp = 6 * b_n + i;
      tmp_0 = 6 * b + i;
      c_r = tmp[B_re_tmp].re;
      re_tmp = tmp[B_re_tmp].im;
      rtDW.dcv[tmp_0].re += c_r * w1 - re_tmp * B_im;
      rtDW.dcv[tmp_0].im += c_r * B_im + re_tmp * w1;
    }
  }
}

for (i = 0; i < 1281; i++) {
  rtDW.dcv1[i].re = rtU.S[i];
  rtDW.dcv1[i].im = 0.0;
}

for (i = 0; i < 6; i++) {
  w1 = 0.0;
  B_im = 0.0;
  for (b = 0; b < 1281; b++) {
    b_n = 6 * b + i;
    c_r = rtDW.dcv[b_n].re;
    re_tmp = rtDW.dcv1[b].im;
```

```
    re_tmp_0 = rtDW.dcv[b_n].im;
    re_tmp_1 = rtDW.dcv1[b].re;
    w1 += c_r * re_tmp_1 - re_tmp_0 * re_tmp;
    B_im += c_r * re_tmp + re_tmp_0 * re_tmp_1;
  }

  rtb_Phat[i].re = w1;
  rtb_Phat[i].im = B_im;
}

w1 = 2.0 * rt_hypotd_snf(rtb_Phat[2].re, rtb_Phat[2].im);
B_im = rt_atan2d_snf(rtb_Phat[2].im, rtb_Phat[2].re);
if (rtb_Phat[1].im == 0.0) {
  re_tmp_1 = rtb_Phat[1].re / delT;
  rtb_Phat_im = 0.0;
} else if (rtb_Phat[1].re == 0.0) {
  re_tmp_1 = 0.0;
  rtb_Phat_im = rtb_Phat[1].im / delT;
} else {
  re_tmp_1 = rtb_Phat[1].re / delT;
  rtb_Phat_im = rtb_Phat[1].im / delT;
}

rtb_Phat[1].re = re_tmp_1;
rtb_Phat[1].im = rtb_Phat_im;
if (-B_im == 0.0) {
  re_tmp = exp(B_im * 0.0);
  re_tmp_0 = 0.0;
} else {
  c_r = exp(B_im * 0.0 / 2.0);
  re_tmp = c_r * cos(-B_im) * c_r;
  re_tmp_0 = c_r * sin(-B_im) * c_r;
}

c_r = (re_tmp_1 * re_tmp - rtb_Phat_im * re_tmp_0) * 2.0;
if (-B_im == 0.0) {
  re_tmp = exp(B_im * 0.0);
  re_tmp_0 = 0.0;
} else {
  d_r = exp(B_im * 0.0 / 2.0);
  re_tmp = d_r * cos(-B_im) * d_r;
  re_tmp_0 = d_r * sin(-B_im) * d_r;
}

d_r = (re_tmp_1 * re_tmp_0 + rtb_Phat_im * re_tmp) * (2.0 / w1);
delT *= delT;
if (rtb_Phat[0].im == 0.0) {
  re_tmp_1 = rtb_Phat[0].re / delT;
  rtb_Phat_im = 0.0;
} else if (rtb_Phat[0].re == 0.0) {
  re_tmp_1 = 0.0;
  rtb_Phat_im = rtb_Phat[0].im / delT;
} else {
  re_tmp_1 = rtb_Phat[0].re / delT;
  rtb_Phat_im = rtb_Phat[0].im / delT;
}

rtb_Phat[0].re = re_tmp_1;
rtb_Phat[0].im = rtb_Phat_im;
```

```
  if (-B_im == 0.0) {
    re_tmp = exp(B_im * 0.0);
    re_tmp_0 = 0.0;
  } else {
    delT = exp(B_im * 0.0 / 2.0);
    re_tmp = delT * cos(-B_im) * delT;
    re_tmp_0 = delT * sin(-B_im) * delT;
  }

  /* Outport: '<Root>/ahatz' incorporates:
   *  MATLAB Function: '<Root>/MATLAB Function'
   */
  rtY.ahatz = (re_tmp_1 * re_tmp - rtb_Phat_im * re_tmp_0) * 4.0 + d_r * d_r *
    w1;

  /* MATLAB Function: '<Root>/MATLAB Function' */
  if (-B_im == 0.0) {
    re_tmp = exp(B_im * 0.0);
    re_tmp_0 = 0.0;
  } else {
    delT = exp(B_im * 0.0 / 2.0);
    re_tmp = delT * cos(-B_im) * delT;
    re_tmp_0 = delT * sin(-B_im) * delT;
  }

  /* Outport: '<Root>/Phat' */
  memcpy(&rtY.Phat[0], &rtb_Phat[0], 6U * sizeof(creal_T));

  /* Outport: '<Root>/ahat' incorporates:
   *  MATLAB Function: '<Root>/MATLAB Function'
   */
  rtY.ahat = w1;

  /* Outport: '<Root>/phihat' incorporates:
   *  MATLAB Function: '<Root>/MATLAB Function'
   */
  rtY.phihat = B_im;

  /* Outport: '<Root>/ahatp' incorporates:
   *  MATLAB Function: '<Root>/MATLAB Function'
   */
  rtY.ahatp = c_r;

  /* Outport: '<Root>/phihatp' incorporates:
   *  MATLAB Function: '<Root>/MATLAB Function'
   */
  rtY.phihatp = d_r;

  /* Outport: '<Root>/phihatz' incorporates:
   *  MATLAB Function: '<Root>/MATLAB Function'
   */
  rtY.phihatz = ((re_tmp_1 * re_tmp_0 + rtb_Phat_im * re_tmp) * 4.0 - 2.0 * c_r *
                 d_r) / w1;
}

/* Model initialize function */
void phasor_estimation_initialize(void)
{
  /* Registration code */
```

```
  /* initialize non-finites */
  rt_InitInfAndNaN(sizeof(real_T));
}
```