Detection of GNSS Faults Using Receiver Clock Drift Estimates

by

Joshua Wood

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 7, 2021

Keywords: GNSS Integrity, Clock Drift, GNSS/INS

Approved by

Dr. Scott Martin, Chair, Assistant Research Professor of Mechanical Engineering
Dr. David Bevly, Bill and Lana McNair Endowed Professor of Mechanical Engineering
Dr. Chad Rose, Assistant Professor of Mechanical Engineering

This thesis is dedicated to my grandparents, who always supported and encouraged me along this journey. I miss you both.

Abstract

This thesis develops a GNSS interference detection method using clock drift estimate monitoring by combining GNSS measurements with inertial navigation systems. During normal operation, GNSS satellites have negligible clock drift, allowing the receiver to estimate the drift of the local oscillator present in the user hardware. When additional transmitters using unsynchronized or low quality oscillators transmit counterfeit GNSS signals, the estimated clock drift will reflect not only the receiver clock drift but also the drift of the transmitter clock. Many unsophisticated or accidental spoofing incidents occur when an individual unknowingly transmits signals, potentially leading to interference on nearby GNSS receivers. Prior work has demonstrated that estimates of receiver clock drift provide a viable option for integrity monitoring at a static location, however the assumptions shown do not hold for dynamic platforms. By incorporating additional sensors into the algorithm, the clock drift can be estimated for dynamic platforms such as autonomous vehicles or commercial drones. The proposed clock drift monitoring algorithm is analyzed using various quality receiver clocks and inertial sensors, expanding the capabilities beyond that of a standard receiver. The developed algorithm utilizes time-differenced carrier phase measurements and mechanized INS measurements to provide estimates of the receiver clock drift prior to using the measurements to update the navigation solution, allowing for the detection of measurement faults before the faults influence the state estimate. Two methods of GNSS/INS coupling are compared, which are used to provide position estimates for use in the clock drift estimation algorithm.

Performance evaluation was conducted using both software simulations and hardware testing to evaluate the detection capabilities of the proposed algorithm under varying conditions. Software simulations were performed with various quality oscillators and inertial sensors, and the resulting detection and mitigation performance was analyzed. In addition to software simulation, hardware-in-the-loop (HWIL) testing of a GNSS/INS system is shown. Data from the HWIL tests was compared to the expected results based on the simulations performed. Results

show that when the interference source is driven by a low quality oscillator, the interference is detectable and harmful effects can be mitigated. As the quality of oscillator driving the interference source increases, however, some effects do not get mitigated by the proposed clock monitoring algorithm and corrupt the navigation solution.

Acknowledgments

Some people may think that success comes from a single person's hard work, but the work done here shows that is not the case. Without the help and support of numerous faculty and lab members from the Auburn GAVLAB, this thesis would have never been completed. Both my advisors, Dr. Bevly and Dr. Martin, played a large role in the work shown here. Dr. Martin was always willing to schedule a last minute meeting to sit down and work through problems and discuss various topics to further this work, and Dr. Bevly provided useful insights and feedback on numerous drafts that helped shape this thesis. Both Dr. Bevly and Dr. Martin go out of their way to help their students succeed, whether it is reading through a presentation late at night or talking to potential employers. Another faculty member, Dr. Howard Chen, also helped this work come together. Howard was always willing to sit down and bounce ideas around, especially during a game of ping pong. I would also like to thank my third committee member, Dr. Chad Rose, for taking the time out of his schedule to sit down and talk about this work and the feedback he provided throughout the thesis writing process. Without the help of these faculty members, this work would not have been possible.

Faculty support is critical to the success of the students, but the support and help from other members of the GAVLAB made a world of difference. This thesis would never been completed if it weren't for the help of Patrick Carter, Scott Burchfield and Sterling Thompson. This group would always be willing to try off the wall experiments and explore ideas as well as take breaks to help get the focus back after a while. Many long nights were spent working with them, and they kept me motivated and awake even with very little sleep. Numerous other members of the lab were invaluable to the completion of this thesis as well. Anderson Givhan and Tanner Watts provided significant insight into the inner workings of the signal side of GNSS receivers, and were always willing to have good conversations about the most random stuff possible. John David Sprunger provided the networking know-how to this uneducated mechanical engineer, allowing me to create the setup I needed for this work. His IT support was not the only thing he

provided, with many laughs being shared over videos or other things. Brendan Schretter, Jake Ward, and Will Bryan were always willing to sit and talk Indy Challenge and trucks, providing a much needed break from the seemingly endless GNSS work that was being conducted. I will forever cherish the friendships I made during graduate school and know that I will continue to have these friendships well into the future.

Outside of the lab, the support and encouragement of my family was instrumental to the success of this thesis. My mom was always willing to sit on the phone and talk to me and take my mind off of work for a while. My sister was always willing to come over and watch Hoshi while I stayed at work many late nights. Hoshi would also provide lots of support without knowing it, but she was always overjoyed when I came home and would tackle me with love. Most importantly, my fiancée Bridget kept me sane and provided all the love and comfort one could need. Not only did she support me at home, she read endless drafts and revisions of this thesis and incredibly did not fall asleep, providing invaluable feedback and edits. Without her, I could not have make this journey.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analog to Digital Converter
ADEV	Allan Deviation
AHRS	Attitude Heading Reference System
AoA	Angle of Arrival
ASIC	Application Specific Integrated Circuit
AVAR	Allan Variance
BOC	Binary Offset Carrier
bps	Bits Per Second
BPSK	Binary Phase Shift Keying
C/A	Coarse Acquisition
CDDIS	Crustal Dynamics Data Information System
CDMA	Code Division Multiple Access
CG	Center of Gravity
CLI	Command Line Interface
COTS	Commercial Off The Shelf
CRPA	Controlled Reception Pattern Array

CS Commercial Service

Cs Cesium

CSAC Chip Scale Atomic Clock

DARPA Defense Advanced Research Projects Agency

dBFS dB Full Scale

DCM Direction Cosine Matric

DDC Direct Downconversion

DLL Delay Lock Loop

DOCXO Double Oven Controlled Crystal Oscillator

DUC Direct Upconversion

ECEF Earth Centered Earth Fixed

ECI Earth Centered Inertial

EKF Extended Kalman Filter

ENU East North Up

ESA European Space Agency

EU European Union

FDE Fault Detection and Exclusion

FFM Flicker Frequency Modulation

FLL Frequency Lock Loop

FN False Negative

FOC Full Operational Capability

FOG Fiber Optic Gyroscope

FP False Positive

FPGA Field Programmable Gate Array

FPM Flicker Phase Modulation

FPR False Positive Rate

GLONASS Global Navigation Satellite System (Globalnaya Navigatsionnaya Sputnikovaya Sistema)

GNSS Global Navigation Satellite System

GPS Global Positioning System

GPSDO GPS Disciplined Oscillator

GPST GPS System Time

GST Galileo System Time

HAS High Accuracy Service

HWIL Hardware In The Loop

IF Intermediate Frequency

IMU Inertial Measurement Unit

INS Inertial Navigation System

IOC Initial Operational Capability

IQ Inphase/Quadrature

ITRF International Terrestrial Reference Frame

LC Loosely Coupled

LEO Low Earth Orbit

LO Local Oscillator

LOS Line of Sight

LS Least Squares

LSB Least Significant Bit

MAC Miniature Atomic Clock

MC Monte Carlo

MCS Master Control Station

MEMS Microelectromechanical Systems

MEO Medium Earth Orbit

MSPS Megasamples Per Second

NBS National Bureau of Standards

NED North East Down

NIST National Institute of Standards and Technology

NLOS Non Line of Sight

NLWLS Nonlinear Weighted Least Squares

OADEV Overlapping Allan Deviation

OCXO Oven Controlled Crystal Oscillator

OS Open Service

P(Y) Precision (encrypted)

PDR Pedestrian Dead Reckoning

PLL Phase Lock Loop

PNT Position, Navigation, and Timing

PRN Pseudorandom Noise

PVA Position, Velocity, Attitude

PVT Position, Velocity, and Time

QPSK Quadrature Phase Shift Keying

RAIM Receiver Autonomous Integrity Monitoring

Rb Rubidium

RF Radio Frequency

RFI Radiofrequency Interference

RINEX Receiver Independent Exchange Format

RLG Ring Laser Gyroscope

RLS Recursive Least Squares

RMS Root Mean Square

RWFM Random Walk Frequency Modulation

SA Selective Availability

SDR Software Defined Radio

SNR Signal To Noise Ratio

SOP Signal of Opportunity

SPP Single Point Position

SQM Signal Quality Monitoring

SSB Single Sideband

SVN Satellite Vehicle Number

SWaP Size, Weight, Power

SWaP-C Size, Weight, Power, and Cost

TC Tightly Coupled

TCXO Temperature Compensated Crystal Oscillator

TOD Time of Day

TP True Positive

TPR True Positive Rate

UAS Unmanned Aircraft Systems

UHD USRP Hardware Driver

USRP Universal Software Radio Peripheral

UTC Coordinated Universal Time

VCXO Voltage Controlled Crystal Oscillator

WERL Woltosz Engineering Research Laboratory

WFM White Frequency Modulation

WGS84 World Geodetic Survey 1984

WLS Weighted Least Squares

WPM White Phase Modulation

XO Crystal Oscillator

XOR Exclusive or

Chapter 1

Introduction

1.1 Background and Motivation

Since the early 2000s, the Global Positioning System (GPS) and other similar Global Navigation Satellite Systems (GNSS) has exploded in popularity as a source of accurate navigation and timing. Each GNSS boasts position accuracy of better than 15 meters RMS and timing synchronization to Coordinated Universal Time (UTC) better than $50ns$ [1]. This performance combined with relatively low cost receivers make GNSS an ideal candidate for many applications that rely on position and timing, such as autonomous vehicles and fleet tracking, the power grid, cellular and first responder communication systems, and many others. One major use of GPS is the navigation of autonomous platforms in various settings such as factories, warehouses, or in the air. Many commercial and government entities have expanded into drone operations, ranging from package delivery to search and rescue [2],[3]. Autonomous unmanned aircraft systems (UAS) rely on the precise navigation to maintain safe operation and complete the desired goal. With the increase in popularity, however, has also come an increase in threats to the accuracy and reliability of GPS. GPS signals are very low power at the surface of the Earth, making them vulnerable to both intentional (jamming, spoofing) and unintentional (multipath, RFI) interference.

Intentional interference presents a large threat to systems that rely on GNSS for precision measurements. In regards to GNSS signals, spoofing is described as [4]:

> *"RF waveforms that mimic true signals in some ways, but deny, degrade, disrupt, or deceive a receiver's operation when they are processed."*

The primary objective of spoofing is to mislead a receiver by transmitting inauthentic GNSS signals with additional delays or modifications to the signal, however spoofing can occur unintentionally with no malicious intent. Successful spoofing of a receiver can be much more difficult to detect, depending on the type of attack. GPS signals can be replicated using various methods, including commercial off the shelf (COTS) GNSS simulators such as the one shown in Figure 1.1 and open source software libraries such as GPS-SDR-SIM [5] combined with software defined radios (SDRs) like the one shown in Figure 1.2. Simulators can be relatively expensive, however SDRs can cost as little as a few hundred dollars, making them obtainable for many individuals.

Figure 1.1: Jackson Labs CLAW GPS Simulator

Figure 1.2: GreatScottGadgets HackRF One

SDRs can be used for meaconing (record and playback) attacks, or constructed into a larger network that is able to generate replicas of live sky signals by using a system such as that described in [6] as well as the simplistic spoofing attacks discussed above. More complicated spoofing events have been demonstrated in several studies, such as [6] and [7]. These attacks are typically intended to disrupt or cause harm to the operator or equipment relying on the GNSS signals, which is often not the case when GNSS spoofing is detected. Unintentional spoofing can occur by amateurs experimenting with software radios or new and emerging RF infrastructure. More recently, spoofing attacks were analyzed and demonstrated against a Tesla Model 3 sedan operating in autonomous mode [8]. The tests in [8] demonstrated that autonomous technology must be resilient to GNSS vulnerabilities prior to wide scale adoption. These attacks, however, are not limited to autonomous technology. Studies conducted in [9] show that human drivers were able to be fooled into following map directions from a smartphone app while the smartphone location was spoofed by a low cost, portable GNSS spoofer. Intentional spoofing is not the only threat that GNSS equipment faces, however. Spoofing has recently come to the forefront of cellular gaming, allowing players to travel to different locations in the world without leaving their homes. Players of the popular mobile game Pokemon Go have taken to spoofing their phone location using low cost SDRs, making the phone think they are in different locations to accomplish location based goals [10]. These actions can lead to spoofing of nearby receivers which, while unintentional, can still cause issues for systems relying on accurate position and time.

Not only can GNSS receivers fall victim to GNSS spoofers, but new uses for GNSS repeaters can have unintended consequences for receivers in the area where they are used. Indoor navigation methods are being explored where GNSS repeaters are implemented, taking live sky signals and retransmitting these signals indoors [11]. While indoor navigation systems using the repeaters do not directly use the measurements, the live sky signals are still rebroadcast and have the potential to interfere with receivers if signal leaks from the building where the system is in use [12]. One such case of a GNSS repeater inadvertently affecting unassuming systems occurred at the Hannover Airport in Langenhagen, Germany [13]. In this situation, the receiver on an aircraft taxiing past a hangar locked onto false signals being transmitted by a repeater

inside of the hangar, causing the aircraft to report an incorrect position location. These systems are marketed for various applications, such as bus transportation infrastructure and emergency response station infrastructure to ensure that GNSS dependent hardware continues to operate while indoors [14], [15]. While these repeaters have benefits, the resulting effects on outside receivers that are not the target for the repeated signals need to have methods to detect and avoid interference from these unintentional threats.

Development of spoofing detection and mitigation methods haven been explored in detail, and a method that can successfully detect all attacks has yet to be developed. Many applications, such as commercial drones or autonomous vehicles, are cost and power limited, making many multielement array methods impractical for detection of spoofing. Spoofing detection using existing hardware and maintaining computational efficiency lend to practical implementations that can benefit consumers of autonomous platforms as they become more readily available in the coming years.

1.2 Prior Art

Several methods have been explored to detect the presence of GPS spoofing, including the monitoring of C/N_0, signal quality (SQM), and clock bias [16]. These methods have the benefit of operating on a single antenna system, however may not prove to be effective against more complex attacks. Other methods which rely on external or larger hardware, such as angle of arrival (AoA) discrimination or comparing the navigation solution to one from an inertial navigation system (INS), add additional robustness, but at increased cost [17]. Multiantenna detection methods are not explored in this thesis, as many controlled reception pattern arrays (CRPA) are too large and too heavy for a commercial drone to carry. Low size, weight, power, and cost (SWaP-C) detection methods are desired for many applications in the commercial sector, with drones having limited power supplies and carrying capacities. Delivery organizations aim to have the farthest delivery range possible, driving low SWaP-C requirements for antispoofing hardware.

Several studies have been done analyzing the impact of transmitter clocks on the received signal. A transmitter clock introduces additional errors to the received signal, which were analyzed by observing additional errors in the position solution in [18]. Methods for monitoring receiver clock bias and drift are explored in [19] and [20]. These methods have been shown to be successful for some typical scenarios. In [20], time differenced measurements are used to estimate the receiver clock drift on an individual channel basis. This method allows for detection of individual faults and is less computationally intensive than other methods discussed. Shang et al. explore the detection of erroneous signals using a static receiver at a surveyed location, simplifying the problem of not knowing where the user is. In addition to the use of a receiver at a known location, Shang et al. make use of a high quality ovenzed oscillator for their experimentation. While large, high quality local oscillators (LOs) are useful for static locations, newer oscillators have been developed with reduced SWaP, making them ideal candidates for mobile or autonomous applications. These improved LOs range from ovenized crystal oscillators to compact atomic oscillators, providing the end user a wide range of performance and cost.

Standalone GNSS receiver methods provide the lowest SWaP for interference detection, but often do not offer the same performance as integrated navigation systems. Coupled navigation authentication methods have shown promise with the ability to bridge GNSS interruptions and provide backup navigational data to the system or user. These methods typically rely on a comparison of the GPS navigation estimates to those of an external source, such as INS or vision based navigation [21],[22]. Both systems can be used to provide estimates of user position, however both have inherent errors and can not provide a perfectly accurate standalone solution without periodic calibration or estimation of the errors. Inertial systems are frequently installed on unmanned aircraft systems (UAS) as well as many modern vehicles, allowing the technology to be leveraged as a redundant navigation solution. As more vehicles become equipped with cameras, this may prove to be beneficial to integrity monitoring as well.

1.3 Research Contributions

Spoofing detection is an area of high interest. This thesis aims to analyze several common methods of signal spoofing and to evaluate a detection method that relies on the statistical properties of the receiver oscillator and inertial sensors as well as the frequency stability characteristics of transmitter oscillators. The proposed algorithm is used against several interference scenarios and the effectiveness is characterized. In addition to an analysis of spoofing methods, this thesis details methods of satellite and inertial navigation and coupled navigation methods. Simulation and estimation of clock errors is discussed in detail and the performance of the algorithm with various grades of transmitter and receiver oscillators is analyzed. A proposed integrity monitoring algorithm is discussed based on clock drift estimation using time differenced measurements, allowing for detection of faults on individual channels. Simulation results using the proposed integrity monitoring architecture are shown and discussed, and hardware-in-the-loop (HWIL) experiments are shown and compared to the simulation results.

1.4 Thesis Outline

This chapter introduced the concept of GNSS integrity and the many threats that face users of GNSS systems for navigation and timing, and gave the motivation for the need for GNSS integrity monitoring systems for dynamic applications. The remainder of the thesis is laid out as follows: Chapter 2 explores current GNSS constellations, receiver architectures, measurements, and position, velocity, and time (PVT) estimation. Chapter 3 details the use of inertial sensors, sensor errors, mechanization of measurements, and coupling of GNSS and INS systems to leverage the advantages of each system. Chapter 4 discusses oscillators and their relationship to GNSS receiver performance as well as the modeling and simulation of timing errors for use in GNSS simulations. Chapter 5 lays the basis for the proposed algorithm and shows the implementation of the algorithm using coupled GNSS/INS architectures. Chapter 6 discuss the testing conducted in software and hardware and analyzes the results of indvidual tests and Monte Carlo simulations. Performance metrics are calculated based on the results from the simulations and a comparison of the different implementation methods is discussed.

Finally, Chapter 7 draws conclusions on the effectiveness of the proposed algorithm and discusses several areas of potential improvement and future research.

Chapter 2

Global Navigation Satellite Systems

In the early days of navigation, sailors and explorers relied on the position of known landmarks or the orientation of celestial bodies to find their way. Modern satellite navigation systems vastly expanded access to precise navigation, greatly reducing the need for expensive user equipment and technical skill required to navigate great distances successfully. Several of these satellite systems are discussed in this section.

2.1 Global Positioning System

The Global Positioning System (GPS) was an expansion of earlier radionavigation satellite systems such as Transit. Designed as a method of trilateration using radio signals transmitted from orbiting satellites, GPS was originally intended for military applications. The system was designed to be a reliable source of positioning with full coverage requiring anywhere from 24-36 satellites in medium earth orbit (MEO). The GPS constellation was declared operational in 1995, providing access to precise position and time to authorized users. During this period Selective Availability (SA) was enabled, which added additional errors into the measurements from the satellites. This was eventually disabled in 2000, allowing civilian users to achieve similar positioning accuracy as the original military users.

GPS is made up of three distinct segments: the control segment, the space segment, and the user segment [23]. The control segment is centered around the GPS Master Control Station (MCS), which monitors and controls the satellites that make up the space segment. The space segment consists of a baseline constellation of 24 satellites orbiting with a radius of 26560km.

Currently, there are 31 operational satellites, not including decommissioned spares that remain in orbit as well [24]. The satellites are arranged into six orbital planes, each with up to six space vehicles per plane [25]. This arrangement, seen in Figure 2.1, allows for up to 36 satellites to be available at a time. Each satellite has a unique satellite vehicle number (SVN) as well as a pseudorandom noise (PRN) identifier that is utilized by the user segment. Over the years, several blocks of satellites have been launched to modernize the GPS constellation. Each new block provided additional capabilities and design features, improving the system over time. The newest satellites, designated Block III/IIIF, were first launched in 2018 and introduced new frequencies and modulations to enhance signal accuracy and integrity [26].

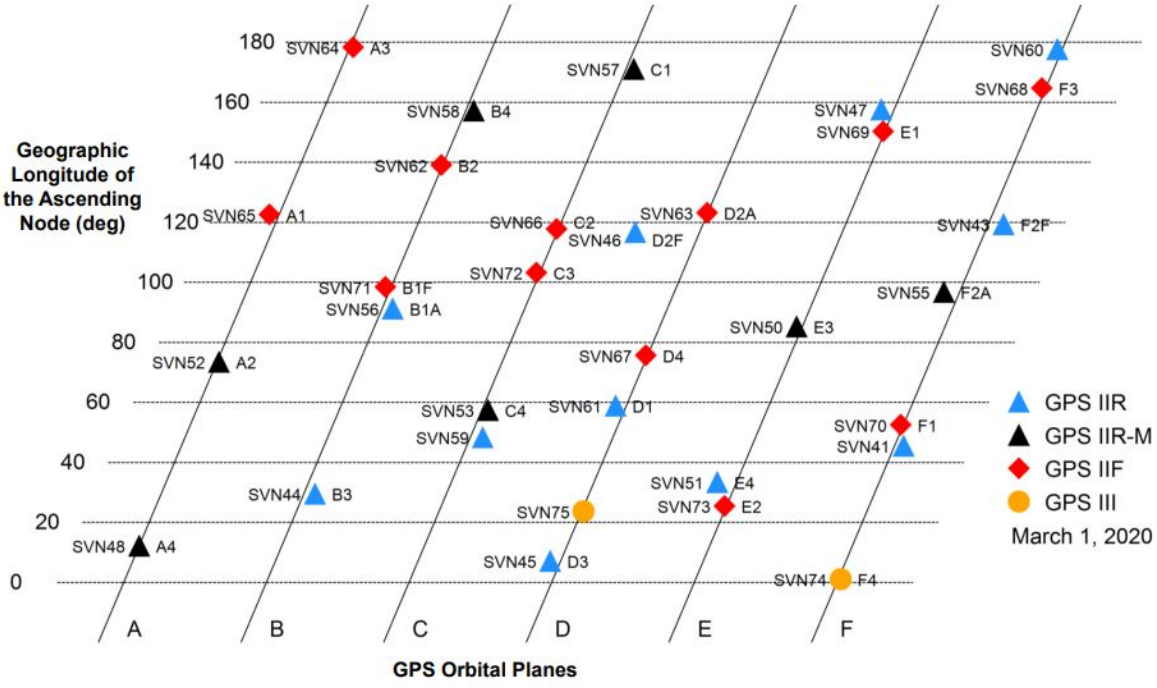

Figure 2.1: Current Arrangement of GPS Satellites [25]

The final segment of the GPS system is the user segment. This consists of the end user GPS receivers that receive and decode the signals transmitted from the satellites. This segment is populated with countless commercial off the shelf (COTS) receivers that range from several tens of dollars to thousands of dollars depending on the performance. GPS receivers are ubiquitous in modern technology, with their applications ranging from personal navigation in

smartphones to time synchronization for telecom antennas. The user segment operates by receiving measurements and other information from the signals transmitted by overhead satellites and processes the data to estimate the user position, velocity, and time (PVT).

2.1.1 GPS Signal Structure

GPS signals are made up of several different components: the carrier, the ranging code, and the navigation data message [23]. Each component has a specific purpose, which will be discussed presently.

The carrier wave is a sinusoidal radio frequency (RF) signal that the other two components are modulated onto for data transmission. Several carrier frequencies are used depending on which signal is being broadcast. Table 2.1 provides an overview on the frequencies transmitted by current GPS satellites, but the discussion here will focus primarily on the civilian L1 C/A signal. Later work will use both the L1 C/A and L2C civilian signals for estimating receiver clock drift.

Table 2.1: GPS Carrier Frequencies

Frequency Band	**Carrier Frequency** (MHz)
L1/L1C	1575.42
L2/L2C	1227.60
L5	1176.45

The transmitted ranging code is a unique binary sequence for each satellite allowing for distinction between each SV. This sequence, called the pseudorandom noise (PRN) code, identifies each satellite and allows for the transmission of the signals on a single frequency without interference [23]. This method of identifying the satellites is known as code division multiple access (CDMA). The PRN codes used for the CDMA scheme provide for a high autocorrelation while maintaining low crosscorrelation with the PRNs from other satellites. By having a high autocorrelation, when the incoming signal is correlated to a local replica which accurately replicates the incoming signal with a certain threshold, the magnitude of the autocorrelation is

significantly higher than the magnitude of the correlation when the replica is not aligned. Additionally, when a signal replica of a satellite that is not visible is correlated with the signal data, the magnitude remains low across the search space. An example of these properties are shown in Figure 2.2, with the left hand plot showing that PRN5 is present in the signal data while PRN6 is not [27]. For the civilian L1 signal, this PRN is also known as the coarse acquisition (C/A) code. The length of the C/A code is 1023 unique bits called chips that are transmitted at a rate of $1.023MHz$. The C/A code is repeated every millisecond where each chip is roughly $1\mu s$ in duration. The wavelength, or chip width, of the C/A code is about $300m$, which is less accurate than the precision GPS signal. In contrast to the large chip width of the civilian C/A code, the military (P(Y)) code has a chip width of about $30m$, much finer than the C/A code. Currently, 32 unique PRN codes are used to identify GPS satellites in the receiver acquisition algorithm.

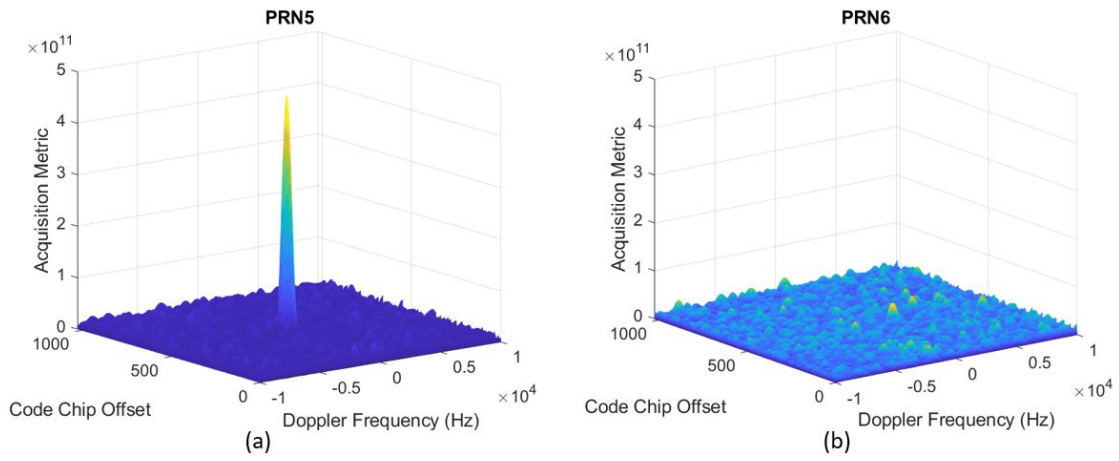

Figure 2.2: GPS L1 Acquisition Plane (a) PRN5 Present (b) PRN6 Not Present

The final component of the GPS signal is the navigation data. The navigation data contains the satellite orbital parameters (ephemeris), clock information, and historical orbital information for all active satellites (almanac). This message is transmitted at a much slower rate than the ranging code, with a transmit rate of 50 bits per second (bps). Because of the slow transmit rate, it takes 12.5 minutes for the entire navigation message to be received. The information critical to GPS navigation, however, is repeated every 30 seconds. By broadcasting the critical navigation data more frequently than the almanac data, receivers can estimate a position without having to wait the entire duration to receive the full message.

Prior to transmission by the satellite, the three signal components must be combined into a single transmitted signal. This is done in two steps: the code and navigation message are combined, then the resulting signal is modulated onto the carrier signal using binary phase shift keying (BPSK) [23]. The code and navigation data are combined using modulo-2 addition, that is, when both bits are the same the result is a 0 and when the bits are different the result is a 1. BPSK modulation does as the name implies, and modulates the phase of the carrier signal depending on current binary value. When a 0 bit occurs, the carrier signal is unchanged, and when a 1 bit occurs, the carrier is multiplied by -1, which phase shifts the signal by 180°. Shown in Figure 2.3, the generated signal (representing L1 C/A only) can be modeled as shown in Equation (2.1), where P_{C1} is the C/A code power, $x^{(k)}$ is the C/A sequence for the k^{th} satellite, $D^{(k)}$ is the navigation data, and f_{L1} and θ_{L1} are the frequency and phase of the carrier wave.

$$S_{L1}^{(k)}(t) = \sqrt{2P_{C1}}x^{(k)}(t)D^{(k)}(t)\cos(2\pi f_{L1}t + \theta_{L1}) \tag{2.1}$$

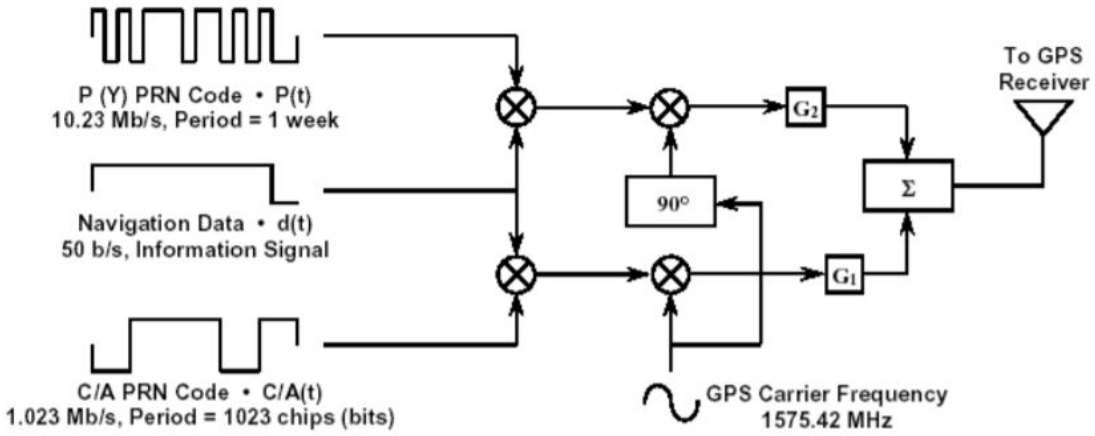

Figure 2.3: GPS L1 Signal Generation [28]

By modulating the carrier with the code and navigation data, the signal energy is spread out over a frequency band instead of being concentrated at a single frequency [23]. The C/A code distributes the power of the L1 signal across a bandwidth of $2MHz$, as shown in Figure 2.4. To de-spread the signal, the C/A code must be removed. The receiver does this by generating local replicas of the known C/A codes and matching them to the incoming signal data. If the code

used to spread the signal is unknown and unable to be reverse engineered, the signal can not be de-spread, and is unusable to an individual receiving the data. This concept is the background for the P(Y) codes transmitted by GPS, where only authorized users have the ability to generate the replica codes to de-spread the signal and use it for navigation. Codes such as the P(Y) code are unable to be tracked and reverse engineered due to their extremely slow repetition rate, adding to their security.

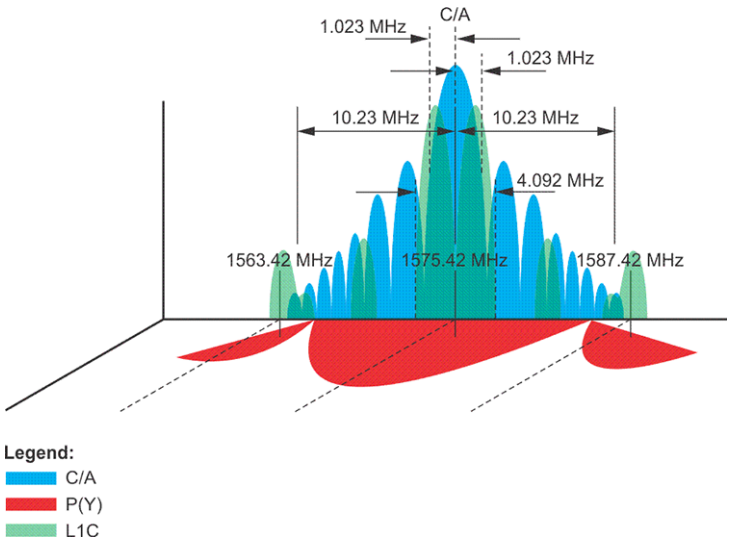

Figure 2.4: GPS L1 Signal Spectrum [29]

Once the signal is received, the receiver goes through the processes of acquisition and tracking. Signal tracking is used to generate the measurements of pseudorange and Doppler as well as identify the bits of the navigation message that are received. The measurements and decoded satellite navigation data are then used to estimate the user position, velocity, and receiver clock errors.

2.2 Galileo

Unlike GPS, which is operated by the United States Air Force and controlled by the Department of Defense, Galileo is operated and controlled by the European Space Agency (ESA), a civilian entity with members from across the European Union (EU). By operating a civilian GNSS system, there is no possibility for a government to intentionally degrade the measurements as was the case with SA early in the life of GPS. Galileo is designed to be interoperable with

other GNSS systems, providing users with the ability to obtain improved coverage in many environments. Each of the three orbital planes will contain eight primary and two reserve satellites, comprising a constellation of 30 total satellites. As of October 2020, 22 satellites are available for navigation and two are available for equipment testing only [30]. The constellation was anticipated to transition from the Initial Operational Capability (IOC) status to the Full Operational Capability (FOC) status by the end of 2020, however the full constellation of FOC satellites has not been completed as of this writing.

Galileo operates in a similar manner as GPS. The system is divided into three segments: ground monitoring and control, space, and user. Like GPS, the ground segment monitors the signals transmitted by the orbiting satellites and calculates orbital parameters and space vehicle clock corrections as needed. These corrections are uploaded to the satellites, which control the orbits and broadcast ephemeris data that are received by the user segment. The user segment receives the signals and utilizes the measurements to calculate PVT. An overview of the Galileo system is shown in Figure 2.5.

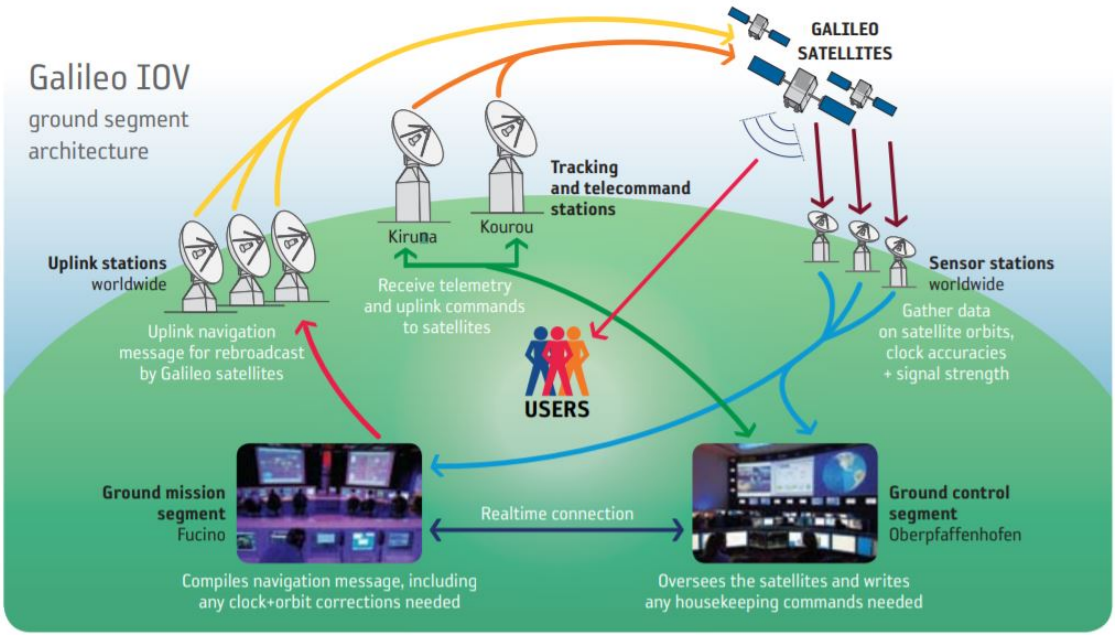

Figure 2.5: Galileo System Overview [31]

On the outside, Galileo appears very similar to GPS, however there are several variations between the two systems. First, as mentioned previously, Galileo is completely commercial instead of military leading to the need for funding to continually operate the system. In addition

to offering an unencrypted open service (OS), Galileo also offers the commercial service (CS), also known as the high accuracy service (HAS) is available for users who pay for it [32]. At the signal level, there are several differences from GPS. Galileo uses a different spreading technique as well as different methods for generating PRN codes. These components will be briefly discussed in the following section. For more information, refer to [33].

2.2.1 Galileo Signal Structure

Currently, Galileo broadcasts at three frequencies: E1, E5, and E6 . The E5 band can further be broken down into the E5a and E5b signals, which are combined prior to transmission creating a single E5 signal. The E5a signal overlaps with the GPS L5 signal, and E1 overlaps with GPS L1. The E5b and E1 signals are of interest to this thesis as several low cost commercial receivers, such as the ublox ZED-F9P chipset and the SwiftNav Piksi Multi, are able to process these signals [34], [35]. The frequencies for each signal are shown in Table 2.2 below [33]. The E6 signal is not discussed further as the ranging code is encrypted and not available for open use [36].

Table 2.2: Galileo Carrier Frequencies

Frequency Band	**Carrier Frequency** (MHz)
E1	1575.42
E6	1278.75
E5	1191.795
E5a	1176.45
E5b	1207.14

Galileo signals transmitted on E1 and E5 make use of a binary offset carrier (BOC) modulation scheme. As opposed to the BPSK techniques employed for GPS L1 and L2, BOC modulation creates two distinct main lobes and spreads the power across each side of the main lobes. An example of the Galileo E1 signal is shown in Figure 2.6.

The BOC modulation approach creates a different autocorrelation function than that of the BPSK modulation used by GPS. This autocorrelation function is discussed in [37]. The advantages of the BOC modulation are that robustness is improved in harsh environments and self

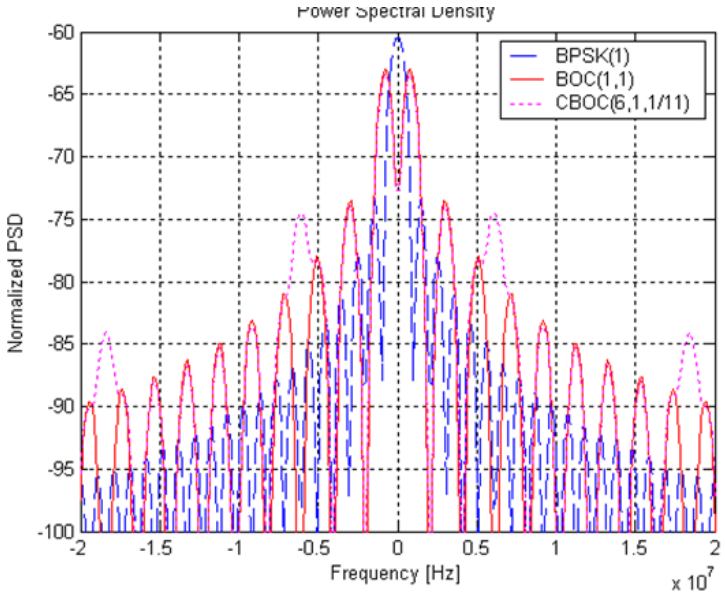

Figure 2.6: GPS L1 vs Galileo E1 PSD [37]

interference is reduced when signals are processed [37]. To successfully track the signal, it must be ensured that the main peak is tracked and not a side peak of the autocorrelation function. A comparison of the GPS BPSK autocorrelation function and the Galileo BOC autocorrelation function is shown in Figure 2.7.

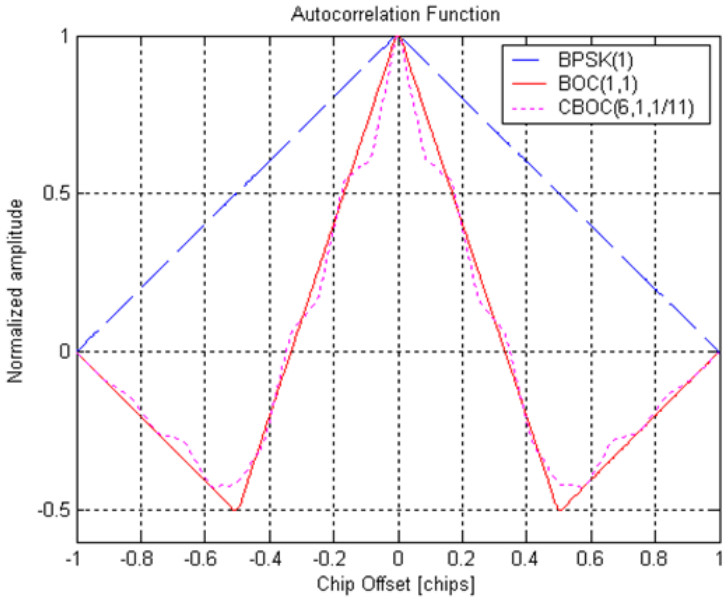

Figure 2.7: BPSK vs BOC Autocorrelation Function [37]

In addition to differing modulation schemes, each broadcast signal has a structure that resembles the QPSK modulation of GPS L5. Each channel provides a data signal and a pilot

signal. Both signals have a primary PRN code that is longer than GPS, which increases the acquisition time for the receiver but also improves the crosscorrelation properties of the signal [37]. To aid acquisition, a shorter secondary PRN code is modulated onto the signal. To generate the tiered ranging code, the secondary code is sequentially compared to the primary code with an exclusive or (XOR) gate to determine the current bit as shown in Figure 2.8 [33]. This code is then combined with the navigation data and carrier signal for transmission.

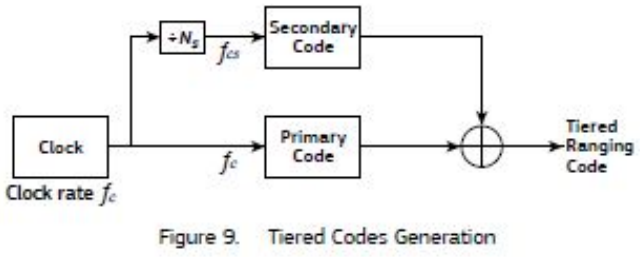

Figure 9. Tiered Codes Generation

Figure 2.8: Galileo Tiered Ranging Code Generation [33]

When the Galileo signals are received, the receiver performs the acquisition and tracking process to decode the navigation data provided in the signals. Each Galileo satellite broadcasts navigation data which is very similar to that transmitted by GPS satellites, with the data containing orbital parameters used to calculate the satellite states. The calculations used to determine the ECEF satellite position, velocity, and satellite clock correction terms from broadcast ephemeris values can be found in [33].

2.2.2 Multiconstellation Navigation Considerations

When navigating with a single constellation of satellites (i.e. standalone GPS), the receiver estimates the user position and time according to the system's reference coordinate frame and time scale. When using several GNSS constellations, there can be variations in the navigation coordinate frame and system time specifications. These differences need to be considered when designing algorithms that position with multiconstellation GNSS signals.

An example of differing references is the GPS and Galileo coordinate frames. Positions computed using GPS are referenced to the World Geodetic Survey 1984 (WGS84) reference frame, whereas Galileo estimates are referenced to the International Terrestrial Reference

Frame (ITRF) [33]. Conveniently, the WGS84 frame and the ITRF are identical, so no transformation in position is needed when estimating a combined solution [38]. However, other systems such as the Russian Global Navigation Satellite System (GLONASS) require a transformation from it's system coordinates to WGS84 to allow for proper use of the estimates.

Not only do various satellite systems use unique reference frames, each system is controlled to a realization of coordinated time determined by the respective master control station. For GPS, this is the GPS System Time (GPST), and for Galileo is Galileo System Time (GST) [33]. Each satellite system is steered by the master control station to within a threshold of universal coordinated time (UTC). Because of the individual system times, the clock bias estimated from GPS measurements would not match the clock bias estimated from Galileo measurements. To counter this, one satellite system (i.e. GPS) is chosen to be a reference system, and time offsets between the reference system and each subsequent system (i.e. Galileo, GLONASS) are estimated. This increases the number of states to be estimated, and requires at least one satellite from each non-reference system to estimate the offsets.

2.3 Receiver Front End Operations

GNSS operates by using measurements gathered by a ground based receiver to estimate the user position, velocity, and time. The observables that are collected by the receiver are derived from the received RF signal and the associated errors induced by satellite and user motion, environmental effects, and others. To obtain these measurements, the RF signals must be collected by the receiver for processing.

In RF operations, the RF front end is a collection of components designed to receive and process the signals prior to digitization. Typical front ends include an antenna, filters, mixers, amplifiers, oscillators, and analog to digital converters (ADCs). An example of a conventional GNSS front end is shown in Figure 2.9, which is known as a superheterodyne front end [39].

Once the signals enter the receiver through the antenna, they are passed through a bandpass filter with the center frequency at the desired signal frequency. The initial bandpass filter limits the bandwidth of the signal entering the remainder of the RF processing chain, preventing out of band noise from being introduced into the receiver. Once the signal is filtered, an amplifier

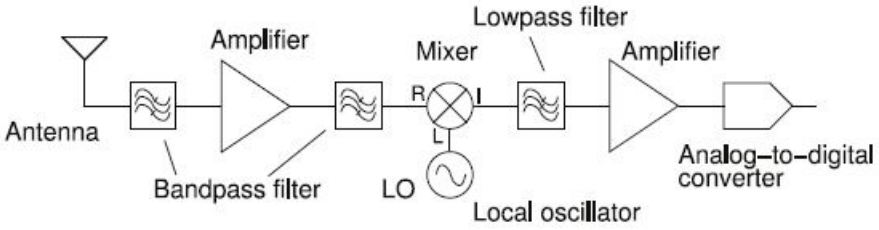

Figure 2.9: Conventional GNSS RF Front End [39]

increases the signal power, allowing the ADC to better quantize the analog values. When received at the Earth's surface, GNSS signals are extremely weak, and are frequently compared to a 25W lightbulb viewed from 10,000km away [40]. By passing the signal through an amplifier, this power is increased, allowing for improved acquisition later in the signal processing block.

GNSS signals operate in the L-band, which spans frequencies from $1 - 2GHz$ [41]. In order to reduce the computational load on the receiver, this RF signal is downconverted to an intermediate frequency (IF). The IF allows samples to be taken at a slower rate than would be required if directly processing the RF signal. To convert the RF signal to an IF signal, the RF signal is mixed with a local oscillator (LO) signal. The RF and LO signals have the form of a sine wave, shown in Equations (2.2) and (2.3).

$$s_{RF}(t) = A_{RF}\cos(2\pi f_{RF}t) \tag{2.2}$$

$$s_c(t) = A_c\cos(2\pi f_{LO}t) \tag{2.3}$$

A mixer is an analog component that performs a time domain multiplication on the two input signals. When two sinusoidal signals are multiplied together, the trigonometric identity shown in Equation (2.4) describes the product.

$$\cos(2\pi f_0 t) * \cos(2\pi f_1 t) = \frac{1}{2}\cos(2\pi(f_0 - f_1)t) + \frac{1}{2}\cos(2\pi(f_0 + f_1)t) \tag{2.4}$$

Mixing the sinusiods produces two signals equal to the sum and the difference of the frequencies of the input signals, which generates the desired IF signal as well as an image signal. As

described in [42], the bits of navigation data message and PRN are shown as a time varying coefficient, giving the expression in Equation (2.5).

$$D(t)\cos(2\pi f_{RF}t)*\cos(2\pi f_{LO}t) = \frac{D(t)}{2}[\cos(2\pi(f_{RF}-f_{LO})t)+\cos(2\pi(f_{RF}+f_{LO})t)] \quad (2.5)$$

In this case, the RF signal is shown to be at a higher frequency than the LO, known as low-side mixing, however a higher frequency LO can be used and is known as high-side mixing [43].

If this signal were to be sampled directly after the mixing the high frequency image would become aliased onto the IF signal, adding erroneous frequency components into the sampled spectrum. To avoid this, the output of the mixer is low pass filtered to remove the image frequency from the signal. Once filtered, the remaining signal can be digitally sampled with an ADC and the digital samples can be passed to the acquisition function in the case of a software receiver, or continue through application specific integrated circuits (ASICs) for hardware correlation. More information on software receivers can be found in [42], [44], and [45]. The desired measurements - pseudorange, Doppler, and carrier phase - can be extracted from the information provided from the tracking loops and passed to the navigation process.

2.4 Receiver Measurements

GPS receivers provide measurements from the signals received from overhead satellites that are used for estimation of the receiver PVT. The two primary measurements are generated from tracking the code phase and the carrier phase. These measurements, however, are biased due to the fact that the receiver clock and the satellite clocks are not synchronized. In addition, the generation of the Doppler frequency between the received and transmitted signals is biased by the clock drift of the receiver oscillator. These measurements will be discussed further in this section.

2.4.1 Pseudorange Measurements

The primary ranging observable used by the receiver is known as the pseudorange. Pseudoranges are defined as the difference between the apparent receive time of the signal at the

receiver and the transmit time from the satellite. The transmission time is encoded onto each signal when it is sent from the satellite, ensuring that this time is synchronized to GPS time. The receive time, however, is dependent on the time indicated by the receiver clock, which is typically a low quality crystal oscillator that is not perfectly synchronized with GPS time, if it is synchronized at all. Equation (2.6) shows the measurement of the pseudorange using the time differences, where ρ is the measured pseudorange, τ is the transit time of the signal between the satellite and receiver, $t^s(t-\tau)$ is the time of transmission from the satellite, and $t_u(t)$ is the reception time of the signal as determined by the receiver clock.

$$\rho = c[t_u(t) - t^s(t-\tau)] \tag{2.6}$$

In terms of a geometrical definition, the pseudorange is a combination of the true line of sight (LOS) range from the satellite to the receiver, the error in the receiver time, atmospheric propagation errors and random noise. Figure 2.10 shows the true LOS range as well as the pseudorange with atmospheric disturbances, and the geometric pseudorange equation is shown in Equation (2.7).

$$\rho^k = r^k + cb + I + T + \eta^k \tag{2.7}$$

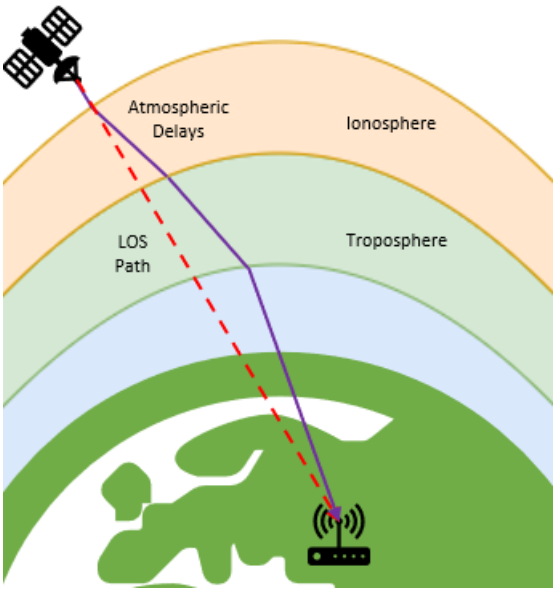

Figure 2.10: LOS and Atmospheric Distortion of Pseudorange

Pseudorange measurements are generated from the code phases of the received GNSS signals. The receiver delay lock loop (DLL) tracks the code phase of the received signal, which provides a fraction of a cycle but not total cycles to the satellites [46]. By combining the partial cycle estimate from the DLL and the number of whole cycles provided by the navigation messages, the distance from the receiver to the satellite can be determined. Code phase measurements are relatively coarse, as the wavelength of the code is $293.05m$. More detail on pseudorange measurement can be found in [46].

2.4.2 Pseudorange Rate Measurements

Doppler shift measurements are generated in the carrier tracking loops [42]. During signal tracking, the phase lock loop (PLL) estimates the Doppler shift of the received signal in order to correctly generate a local replica used in the receiver. The received signal frequency depends on both the velocity of the satellite and the receiver, which correspond to the line of sight (LOS) velocity between the receiver and satellite. To use the measured Doppler frequency to estimate user velocity, the observables must be converted from Hertz to meters per second. This is done by multiplying the measured Doppler values by the carrier wavelength, as is done in Equation (2.8). For a receiver using the L1 signal, this corresponds to a wavelength of about $19cm$.

$$\dot{\tilde{\rho}} = -f_{Doppler}\lambda_{L1} \tag{2.8}$$

The measured pseudorange rate consists of the true line of sight velocity between the receiver and satellite, error due to the receiver clock drift, and errors due to other factors such as motion during the measurement interval and atmospheric effects [23]. The clock drift of the receiver makes it appear as if there is a larger Doppler shift than is actually present, and must be estimated as well to accurately calculate the user velocity. Equation (2.9) shows the combination of these terms into the measured value.

$$\dot{\tilde{\rho}}^k = \dot{r}^k + c\dot{b} + \dot{\epsilon}^k \tag{2.9}$$

Using the equation above, a measurement model can be derived and used to estimate the user velocity and receiver clock drift. This methodology is described in Section 2.5.2. If the receiver is known to be static, the user velocity calculation is not necessary, and the estimation with the pseudorange rates can be omitted.

2.4.3 Carrier Phase Measurements

Code phase measurements are the primary observable used for positioning in GNSS receivers due to their unambiguous nature. Carrier phase measurements, which are generated from the beat phase between the received signal and the local carrier replica. The carrier phase is generated by counting the cycles of the carrier signal received since a reference point. Receivers make this measurement using the phase lock loop in the receiver front end, initially synchronizing the phase between the incoming and replica signals [23]. Once the phase is synchronized, any changes in the phase can be tracked, generating a measurement of full cycles since synchronization as well as fractional cycles. This measurement, however, does not provide the full number of cycles between the receiver and the satellite. This number of full cycles is known as the integer ambiguity, denoted as N. To generate the measurements of carrier phase, the measured phase is related to the transmitted phase using the transit time, similar to code phase measurements.

The measured carrier phase can be related to the geometric range and clock error as was done with the pseudorange measurement. The carrier phase is a function of the true range between the satellite and receiver, receiver clock error, integer ambiguity, the atmospheric disturbances, and unmodeled or random errors. The carrier wavelength is used to convert from units of meters to units of cycles for consistency where the magnitude depends on the transmitted carrier frequency. This formulation is shown in Equation (2.10).

$$\phi = \frac{1}{\lambda}[r + I + T] + \frac{cb}{\lambda} + N + \eta \tag{2.10}$$

Carrier phase measurements are inherently more precise than code phase measurements, with a wavelength of $19cm$ for L1. Positioning with the carrier phase measurement would

result in far more accurate position estimates, however direct positioning is not possible due to the integer ambiguity that is present in the measurements. Several methods are available for resolving the ambiguity, but these methods are not explored in this work. The precise nature of the carrier phase will be used, however, to generate a more accurate estimate of the receiver clock drift as discussed later in Chapter 5.

2.5 Position, Velocity, and Time Estimation

GPS receivers provide the user the ability to accurately know their position and velocity. Pseudorange and Doppler measurements are related to the receiver states through a linearized measurement model, H, discussed in the following sections. The direct relation between the states and the measurements is nonlinear, so the model is linearized about the best position estimate for use in the linear estimators that are discussed later in this section. In some cases, such as a static receiver, there is no desire to estimate the receiver velocity, and as such the related states can be omitted. In addition to position and velocity estimates, the receiver clock bias and drift are also estimated. This provides not only information about the time biases in the measurements, but also allows the receiver to synchronize the internal clock to GPS time, allowing for an accurate timing source which can be used for synchronization of other systems.

2.5.1 Position Measurement Model

Position can be estimated with the pseudorange measurements generated by the receiver as discussed in Section 2.4.1. When enough satellites are in view, these measurements constrain the receiver location using a method called trilateration. As shown in Figure 2.11, each measurement of pseudorange constrains the user to a sphere of constant radius around the satellite. By having four or more measurements from separate satellites, the user position can be constrained to the correct area. The pseudorange measurements can be related to the states shown in Equation (2.11) using the geometry matrix, H, shown in Equation (2.12).

$$\hat{\mathbf{x}} = [x \quad y \quad z \quad b]^T \tag{2.11}$$

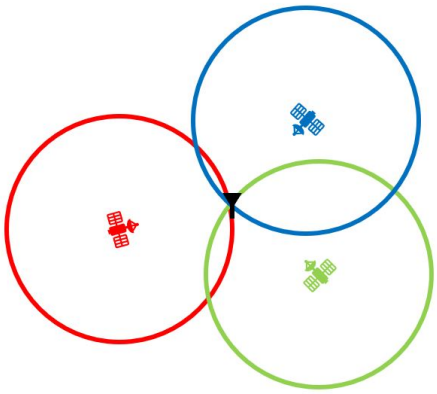

Figure 2.11: 2D Representation of GPS Range Constraints

$$\mathbf{H} = \begin{bmatrix} \frac{-x^1-x_0}{\|\mathbf{x^1}-\mathbf{x_0}\|} & \frac{-y^1-y_0}{\|\mathbf{y^1}-\mathbf{y_0}\|} & \frac{-z^1-z_0}{\|\mathbf{z^1}-\mathbf{z_0}\|} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{-x^i-x_0}{\|\mathbf{x^i}-\mathbf{x_0}\|} & \frac{-y^i-y_0}{\|\mathbf{y^i}-\mathbf{y_0}\|} & \frac{-z^i-z_0}{\|\mathbf{z^i}-\mathbf{z_0}\|} & 1 \end{bmatrix} \tag{2.12}$$

The geometry matrix relates the pseudorange measurements to the position and clock bias states using the pointing vectors from the estimated user position to each respective satellite. The geometry matrix shown assumes a static receiver, with $[x^i, y^i, z^i]$ as the position of the i^{th} satellite and $[x_0, y_0, z_0]$ as the current best estimate of the user position in the Earth centered Earth fixed (ECEF) reference frame [23].

The number of rows in the geometry matrix corresponds to the number of pseudorange measurements from unique satellites. To estimate a position, at least four measurements are needed. If more than four measurements are available, the system is said to be overdetermined, and methods of error minimization can be used to calculate the best solution. Two methods are commonly used for position (and velocity) estimation: Gauss-Newton Nonlinear Least Squares (LS) and an Extended Kalman Filter (EKF). Each method is discussed in Sections 2.5.3 and 2.5.4 respectively.

2.5.2 Velocity Measurement Model

The relation of the pseudorange rates to the user velocity and clock drift is similar to that of the pseudorange and user position and clock bias. The pseudorange rate can be related to the user velocity and clock drift terms as shown in Equation (2.13).

$$\dot{\tilde{\rho}} = \mathbf{H} \begin{bmatrix} \mathbf{v} \\ \dot{b} \end{bmatrix} + \tilde{\epsilon}_{\dot{\phi}} \tag{2.13}$$

The geometry matrix, $\mathbf{H}$, is the same as described in the previous section, consisting of the unit pointing vectors to each satellite. As was mentioned in the discussion concerning the estimation of user position from pseudoranges, both methods can be used to estimate the user velocity. Typically, the estimation of user position and velocity are combined into a single step using a combined geometry matrix and state vector. The combined geometry matrix is shown in Equation (2.14), and the state vector is shown in (2.15).

$$\mathbf{H_c} = \begin{bmatrix} \frac{-x^1-x_0}{\|\mathbf{x^1}-\mathbf{x_0}\|} & 0 & \frac{-y^1-y_0}{\|\mathbf{y^1}-\mathbf{y_0}\|} & 0 & \frac{-z^1-z_0}{\|\mathbf{z^1}-\mathbf{z_0}\|} & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{-x^i-x_0}{\|\mathbf{x^i}-\mathbf{x_0}\|} & 0 & \frac{-y^i-y_0}{\|\mathbf{y^i}-\mathbf{y_0}\|} & 0 & \frac{-z^i-z_0}{\|\mathbf{z^i}-\mathbf{z_0}\|} & 0 & 1 & 0 \\ 0 & \frac{-x^1-x_0}{\|\mathbf{x^1}-\mathbf{x_0}\|} & 0 & \frac{-y^1-y_0}{\|\mathbf{y^1}-\mathbf{y_0}\|} & 0 & \frac{-z^1-z_0}{\|\mathbf{z^1}-\mathbf{z_0}\|} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \frac{-x^i-x_0}{\|\mathbf{x^i}-\mathbf{x_0}\|} & 0 & \frac{-y^i-y_0}{\|\mathbf{y^i}-\mathbf{y_0}\|} & 0 & \frac{-z^i-z_0}{\|\mathbf{z^i}-\mathbf{z_0}\|} & 0 & 1 \end{bmatrix} \tag{2.14}$$

$$\hat{\mathbf{x}} = [x \quad \dot{x} \quad y \quad \dot{y} \quad z \quad \dot{z} \quad b \quad \dot{b}]^T \tag{2.15}$$

Using the combined geometry matrix, the receiver states from Equation (2.15) can be related to the measurements through Equation (2.16).

$$\begin{bmatrix} \tilde{\rho} \\ \dot{\tilde{\rho}} \end{bmatrix} = \tilde{\mathbf{z}} = \mathbf{H_c}\hat{\mathbf{x}} \tag{2.16}$$

The measurement model constructed in Equation (2.14) can be used to estimate the user PVT by using various estimation algorithms. GNSS receivers most commonly use a least squares or a Kalman filter based estimation algorithm, which will be discussed in the following sections. Derivations of the algorithms can be found in [23] and [42]. The weighted least squares (WLS) algorithm is a straightforward estimation technique allowing for calculation of PVT, but has the disadvantage that information from the previous estimation epoch is discarded. This limits the accuracy of the WLS estimator and does not provide a useful basis for monitoring of states over time as is desired for GNSS integrity. A recursive least squares (RLS) or extended Kalman filter (EKF) both account for information from previous epochs, however, the EKF provides a kinematic model for periods of GNSS outages that can be used to propagate the estimate uncertainty, making it a good foundation for integrity monitoring algorithms.

2.5.3 Weighted Least Squares Estimation

In the previous sections, linear measurement models were derived relating the states to the measurements collected by the receiver. During normal operations, the receiver must estimate four states using the pseudorange measurements and four more using pseudorange rate measurements. If the case arises when less than four measurements are available, the solution is underdetermined, and no estimate can be generated using standard methods. If exactly four measurements are available, the solution is straightforward and can be solved easily through a matrix inversion. Typically there are more than four satellites visible at a given time, leading to an overdetermined solution. It is desired to get the best estimate of PVT, so frequently a best fit solution incorporating all available measurements is preferred. In the case of a least squares (LS) solution, the objective is to minimize the sum of the residuals squared [23].

To minimize the errors, an error state vector, $\delta\mathbf{x}$, must be defined. The states are the error in the position estimate and clock bias, and are calculated from the difference between the measured and estimated pseudoranges. The measurement model is shown in Equation (2.17), where the geometry matrix is unchanged from the one defined in the previous sections.

$$\delta\mathbf{z} = \mathbf{H}\delta x + \tilde{\epsilon}_z \tag{2.17}$$

When the solution is overdetermined (m >4), the geometry matrix is not square and the direct inverse can not be taken. To solve this, a pseudoinverse is taken [47]. The values solved for in Equation (2.18) are corrections to the current estimate of the user states, and are added to the current state estimate once estimated.

$$\delta\hat{\mathbf{x}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\delta\mathbf{z} \tag{2.18}$$

Once the corrections are added to the current state estimate the geometry matrix is recalculated using the new best guess. This process is iterated until the corrections estimated are below a specified threshold, at which point it is stated that the estimate has converged, and a new batch of measurements can be processed. The process of updating the state estimate with the error states is shown in Equation (2.19),

$$\hat{\mathbf{x}} = \mathbf{x}_0 + \delta\hat{\mathbf{x}} \tag{2.19}$$

A variation on the standard least squares solution is the use of a weighted least squares (WLS) algorithm. By including weights into the solution, the relative qualities of the measurements are taken into account, weighting poor measurements less than good measurements [47]. The weights used can vary depending on the methods employed, and weights used here are based on the received signal C/N_0 values. The weighting matrix, shown in Equation (2.20), is a $2m \times 2m$ diagonal matrix where m is the number of measurements at each epoch.

$$W = \begin{bmatrix} \frac{1}{\sigma_1^2} & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \frac{1}{\sigma_i^2} \end{bmatrix} \tag{2.20}$$

The weights used can be calculated using the variances of the PLL and DLL shown in [48].

Estimates of the error states are calculated using the same concepts as discussed previously, only with the inclusion of the weighting matrix. Equation (2.18) is modified to include the weighting matrix, and is shown in Equation (2.21).

$$\delta\hat{\mathbf{x}} = (\mathbf{H}^T W\mathbf{H})^{-1}\mathbf{H}^T W\delta\mathbf{z} \tag{2.21}$$

Once the state errors are estimated, they are applied to the previous estimate of the states as was done in Equation (2.19). After the states are updated, the next batch of received measurements are used to perform PVT estimation for the following epoch. Standard LS algorithms are also known as single point position (SPP) algorithms as they do not carry any information over from one epoch to the next. Algorithms such as the recursive least squares or EKF incorporate an estimate of confidence of the previous estimates into the current estimation of PVT.

2.5.4 Extended Kalman Filter Estimation

The linear Kalman filter provides an optimal estimate of the specified states in the same manner as a least squares algorithm, but includes additional benefits such as time propagation of uncertain states as well. A nonlinear estimator, the EKF is used for GNSS navigation and introduces some suboptimality such as local minima as opposed to the global minima of a linear estimator [47]. The Kalman filter architecture consists of five main steps, as seen in Figure 2.12. These steps involve a time update, or prediction step during which the previous best estimate and estimate uncertainty are propagated to the current epoch, and a measurement update or correction, during which new measurements are used to correct the predicted estimates.

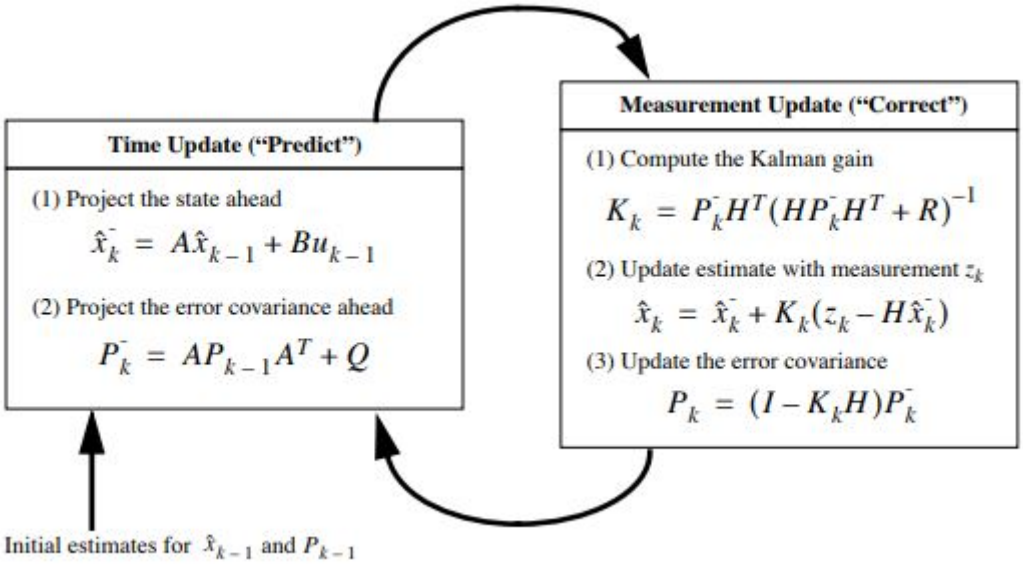

Figure 2.12: Diagram of Kalman Filter Process [49]

During the time update, the state matrix propagates the previous estimates of position, velocity, and clock errors forward to the current time. Many EKF based estimators employ a kinematic constant velocity model, as shown in Equation (2.22).

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.22}$$

In some models, system inputs can be included in the state propagation step, however this is not required for the GNSS model. During periods between measurement updates, the model is used to maintain an estimate of the filter states. In addition, the state uncertainty matrix P is also propagated. The state uncertainty grows over time and is reflected in the uncertainty covariance matrix until a measurement update occurs. The EKF time update steps can be seen in Equations (2.23) - (2.24).

$$\hat{x}_k^- = \Phi \hat{x}_{k-1} \tag{2.23}$$

$$P_k^- = \Phi P_{k-1} \Phi^T + Q \tag{2.24}$$

The process noise covariance, Q, is a diagonal matrix that contains the expected dynamic errors for each of the states. Often times the position and velocity noise is set to a fixed value for simplicity. The clock state uncertainty can be derived from the power law noise model,

discussed in Section 4. The process noise covariance matrix is shown in Equation (2.25) [50].

$$Q = \begin{bmatrix} Q_{PV} & 0 & 0 & 0 \\ 0 & Q_{PV} & 0 & 0 \\ 0 & 0 & Q_{PV} & 0 \\ 0 & 0 & 0 & Q_{clk} \end{bmatrix} \tag{2.25}$$

The uncertainty in the dynamics drive the errors in the time update step, with changes in the true dynamics not being reflected in the propagated estimates until a measurement update occurs. The blocks Q_{PV} and Q_{clk} are defined in Equations (2.26) and (2.27), respectively.

$$Q_{PV} = \begin{bmatrix} \frac{S_p \Delta t^3}{3} & \frac{S_p \Delta t^2}{2} \\ \frac{S_p \Delta t^2}{2} & S_p \Delta t \end{bmatrix} \tag{2.26}$$

$$Q_{clk} = \begin{bmatrix} q_1 \Delta t + \frac{q_2 \Delta t^3}{3} & \frac{q_2 \Delta t^2}{2} \\ \frac{q_2 \Delta t^2}{2} & q_2 \Delta t \end{bmatrix} \tag{2.27}$$

When new measurements are received from the GNSS front end, the propagated states are corrected using the differences between the measured and predicted states. To update the states, the Kalman gain is calculated using Equation (2.28).

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{2.28}$$

The Kalman gain calculation is a function of the error covariance, measurement model, and measurement noise covariance to capture the noises and uncertainty in the received GNSS measurements. The R matrix, shown in Equation (2.29) is a diagonal matrix which uses the same model for pseudorange and pseudorange rate uncertainties as the least squares weighting

matrix, but does not take the inverse of the value.

$$R = \begin{bmatrix} \sigma_1^2 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \sigma_i^2 \end{bmatrix} \tag{2.29}$$

This matrix, known as the measurement noise covariance matrix, accounts for the uncertainty or error in the received measurements as a function of the receiver dynamics as well as the measured C/N_0. Once the Kalman gain is calculated, the states are corrected and a new measure of uncertainty is calculated. The Kalman gain is applied to the difference in measured and estimated observables, given by δz and shown in Equation (2.30).

$$\delta z = z_k - H\hat{x}_k^- = \begin{bmatrix} \tilde{\rho}_k - \hat{\rho}_k \\ \tilde{\dot{\rho}}_k - \hat{\dot{\rho}}_k \end{bmatrix} \tag{2.30}$$

The estimated pseudoranges and pseudorange rates are calculated using the satellite position and velocity as well as the estimated user position and velocity at the time of measurement transmission and reception, respectively. In addition to the satellite and user states, user clock error corrections are applied to the estimated observables to accurately reflect the measured pseudorange between the satellite and user. A full derivation of this process can be seen in [23] and various other GNSS textbooks. The measurement update is added to the propagated state vector shown in Equation (2.31), and the error covariance update is shown in Equation (2.32).

$$\hat{x}_k = \hat{x}_k^- + K_k \delta z \tag{2.31}$$

$$P_k = (I - K_k H) P_k^- \tag{2.32}$$

One significant advantage to the EKF formulation of the GNSS navigator is that the last best estimate can be propagated forward in time during periods of interruption. This allows the processor to maintain an estimate of position and velocity until measurements return, aiding both reacquisition and tracking. For short periods of time the dead reckoned solution can be

accurate so long as no major changes to the dynamics occur, however the PVT errors grow rapidly if the constant velocity model does not accurately capture the dynamics of the vehicle.

2.6 GNSS Spoofing Methods

GNSS spoofing can be done in several different ways, ranging from extremely simplistic to highly complex attacks. Several different attack types are detailed in [51], and an analysis of threats is discussed in [52]. This thesis explores the single transmitter, non-cooperative spoofing methods. Two primary methods are explored here: meaconing and signal generation. Multi-antenna attacks and cooperative attacks are not explored in this thesis, however they have been explored in other sources such as [53].

2.6.1 Meaconing

Meaconing attacks operate on the principle of record and replay. A simple meaconer consists of a receive antenna, hardware to amplify the incoming signal, and a transmit antenna. A simple meaconer introduces additional time delay into the signal that is equivalent to the time delay for the processing of the signal and the time of flight between the transmit antenna and the user receiver [51]. When processed by the user receiver, the position is estimated to be the location of the meaconer receive antenna, and the clock states will be estimated incorrectly. A typical meaconing attack is shown in Figure 2.13, where the receive and transmit antenna of the meaconer are at the same location. In some cases, the two antennas are not co-located, leading to additional signal delay due to transmission of the signal data from one location to another.

Simulation of a meaconing event can be represented by the reception of signals with additional delays based on the distance between the meaconer and the user as well as processing delays introduced by the meaconer. In a live scenario, the distance from the meaconer to the user would be unknown except in highly advanced attacks. In simulations, the range between the transmitter and user is able to be known, allowing for the calculation of the additional signal delay. As stated previously, the distance between the transmitter and receiver and the signal processing time will introduce additional delay in the signal. When the authentic signal is repeated by a meaconer, frequency error will be added to the transmitted signal due to the

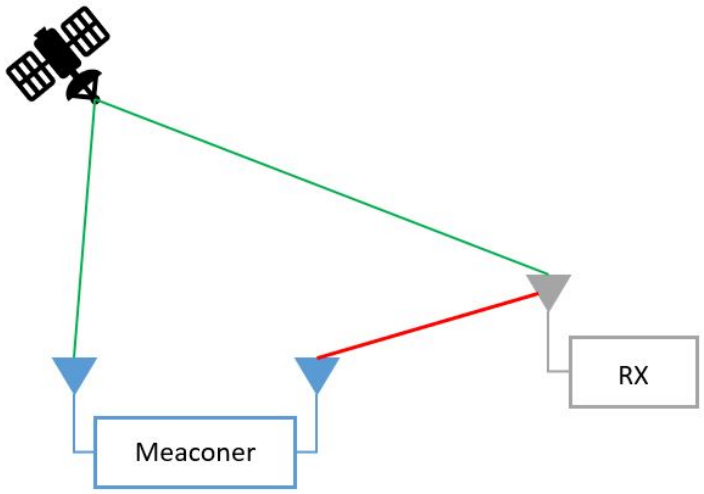

Figure 2.13: Diagram of Simple Meaconing Attack

clock drift of the meaconer reference oscillator. The frequency received at the victim receiver will become not only a function of the satellite and user clock drifts, but also the drift of the meaconer clock as well. Although no clock bias term is directly added, the phase noise of the meaconer oscillator will affect the replayed signal, and would primarily be observed on the phase measurements at the victim receiver. A basic model for the received pseudorange after passing through a meaconer can be shown in Equation (2.33).

$$\rho_m = r_{s,m} + r_{m,u} + cb_u + ct_{proc} + I + T + \eta \tag{2.33}$$

In Equation (2.33), $r_{s,m}$ is the range between the satellite and meaconer, and $r_{m,u}$ is the range between the meaconer and victim receiver. The meaconer clock bias does not play a part because the meaconer simply replays the signal, so the offset between the meaconer time and GPST does not matter. The processing time of the meaconer does influence the measured range, and is represented by t_{proc} and converted to a range by multiplying by the speed of light. At the correlation level, a meaconer will appear to be noise if the additional delays cause the signal to be delayed outside of a chip spacing. For additional errors to affect the tracking loops, the receiver would have to be within about one half chip spacing and have a higher received power level than the true signals. When the meaconer is outside of the one chip separation, the erroneous signal causes little error as the tracking loops continue to track the authentic signal

[54]. If the receiver were to lose tracking however, it is possible that the repeated signal would be acquired so long as the power level was higher than that of the authentic signal. When the victim receiver estimates a position using the repeated signals, the estimated position will be that of the meaconer receive antenna [51]. The receiver clock bias will be greater than the true clock bias, creating a solution that appears to occur earlier in time than the true solution.

2.6.2 Spoofing

Spoofing attacks are much more sophisticated than meaconer (or repeater) attacks. Most GNSS spoofers are synchronized to authentic GNSS signals and in some cases can track the position of the victim receiver. These attacks commonly start as aligned events, where the code phase is aligned with the authentic signals received at the victim. The spoofer can raise the power levels of the inauthentic signals and capture the tracking loops of the victim receiver, then drags the code and carrier away from the authentic signal to create a solution that is controlled by the spoofer. Like the meaconer, there are delays based on the distance between the spoofer and the victim receiver, the time for the spoofer to receive the signal, and the processing delay. Unlike the meaconer, however, spoofers are able to predict signal components for a future time based on the currently received authentic signals, allowing for total control of the signal that is transmitted to the victim receiver [6]. A measurement level model for a spoofing signal is shown in Equation (2.34) [55].

$$\rho_{sp} = r_{s,sp} + r_{sp,u} + cb_u + ct_{proc} + ct_{ctrl} + I + T + \eta \tag{2.34}$$

In Equation (2.34), the same delays based on the distance between the receiver and spoofer positions are introduced, however an additional term, ct_{ctrl} is introduced. This term allows the spoofer to advance or delay the signal to generate the desired measurements, allowing for control of the estimated position and time of the victim receiver. Additionally, the spoofer may also have the ability to alter the navigation data of the received signal, causing the victim receiver to believe the satellites are in a different location from where they actually are. These types of attacks are highly complex, and are less likely than a simulator attack, which simply generates

navigation data for a given location. Signal generation can be done with various combinations of devices, such as COTS signal simulators like the Spirent GSS9000 or Spectracom GSG-6. These signal simulators, however, can be costly and impractical for someone who may want to fool their receiver in a fleet tracked vehicle or other mobile platform. An example of this occurred in 2019, when attendees at the Geneva Motor show noticed that the cars on display all showed a location in Buckingham, England, in the year 2036 [56]. It is believed that this event was the result of someone broadcasting generated GPS signals from a GNSS simulator, however this could have been done just as easily with an inexpensive software defined radio (SDR). Open source packages such as GPS-SDR-SIM are available that generate GPS signal data for a given location and time based on satellite navigation data, as shown in Figure 2.14 [5].

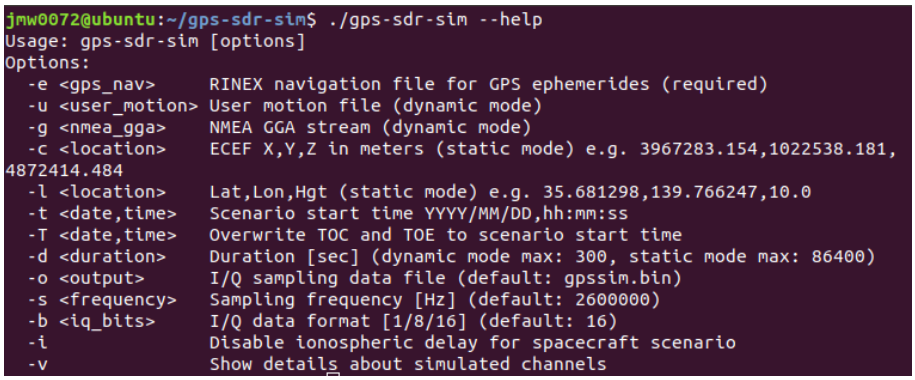

Figure 2.14: GPS-SDR-SIM Simulation Options

Not only does GPS-SDR-SIM provide the ability to generate GNSS data, it also provides the ability to directly interface with a software defined radio to broadcast the generated data. The broadcast data can cause unprotected GNSS receivers in the area to degrade performance or malfunction, causing problems for systems relying on GNSS availability. SDRs are discussed in the following section, with a focus on GNSS recording and transmission.

2.7 Software Defined Radios

Most COTS GNSS receivers rely on application specific integrated circuits (ASICs), which are computationally efficient and typically compact [57]. These GNSS specific chips are designed

with a specific objective, and do not perform tasks outside of the designed goal. For commercial or industrial use, these receivers fill the role as needed, providing access to GNSS position and time services at various price points depending on the application. For research and development, however, more flexibility is frequently desired. Software defined radios, or SDRs, provide a flexible RF platform that can be used for various applications. SDRs are available as inexpensive receiver chips for hobbyists such as the RTL-SDR to high end SDRs designed for research use such as the Ettus Universal Software Radio Peripheral (USRP) [58],[59].

USRPs provide an extremely flexible RF receiver and transmitter that can be used for any frequency supported by the daughterboard. The N210 platform uses an FPGA processor to perform digital operations on signals received by the interchangeable daughterboards available from Ettus Research. The overall schematic for the USRP N210 SDR is shown in Figure 2.15. The FPGA board controls the up and downconversion of the transmitted/received signals and the interface between the USRP and the host computer. By implementing interchangable daughterboards, a single USRP can be adjusted to different applications, adding further flexibility to the unit. Two daughterboards used for this work are the WBX and SBX boards, both of which cover the entire GNSS spectrum. A comparison of the two daughterboards is shown in Table 2.3.

Table 2.3: Typical Inertial Measurement Unit Parameters

Daughterboard	Frequency Range	Analog Bandwidth	Channels
WBX	$50 - 2200MHz$	$40MHz$	TX/RX RX2
SBX	$400 - 4400MHz$	$40MHz$	TX/RX RX2

The USRP daughterboard acts as the GNSS front end, receiving the incoming RF signal and passing the filtered signal to the ADC. USRPs are direct downconversion (DDC) receivers, where the RF is mixed directly with an LO at the specified incoming frequency to convert the signal directly to a zero IF signal, also known as baseband signal. The baseband signal is mixed with an inphase (cosine) and quadrature (sine) LO signal during the downconversion process, which generates the baseband inphase and quadrature (IQ) samples. These samples represent

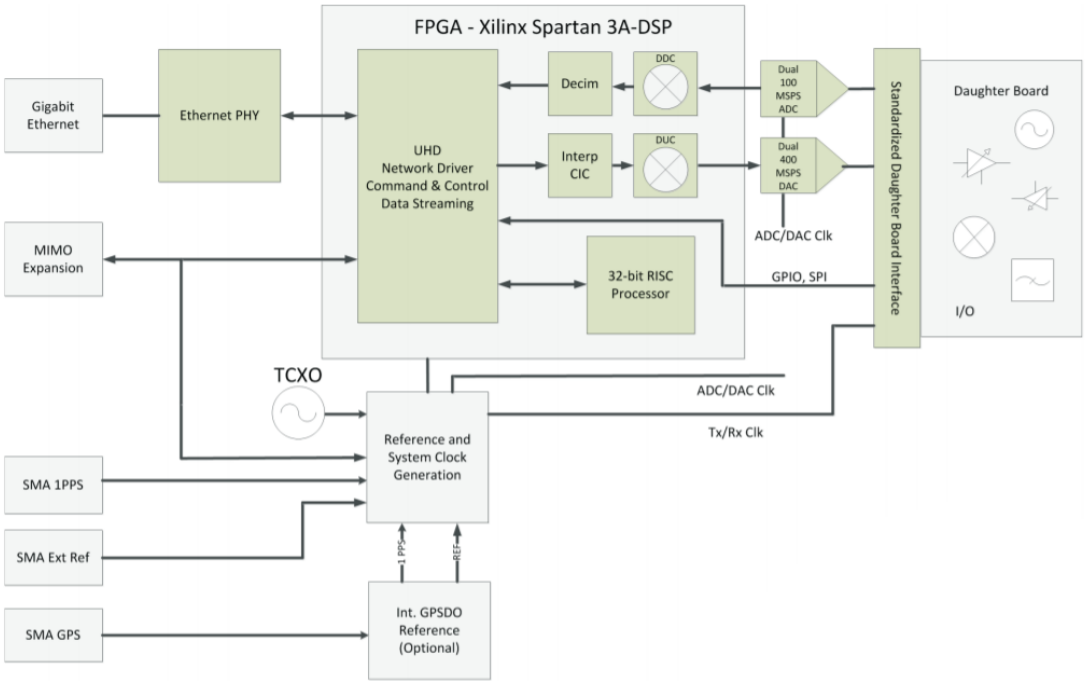

Figure 2.15: USRP FPGA and Host Interface [60]

the received signal, with the inphase samples representing the real component of the complex signal and the quadrature samples representing the imaginary component. A block diagram for a WBX front end is shown in Figure 2.16.

To acquire and track GNSS signals, COTS GNSS receivers implement several amplifier and filter banks. Since the USRP is not specifically designed for GNSS applications, only a small amount of amplification is available on board the N210. To properly quantize the received signal when sampling and converting to digital, GNSS signals must be amplified higher than is possible with the on-board amplifiers. To accomplish this, an external signal conditioning devices are frequently used as a substitute for a dedicated GNSS front end. An example of these devices, shown in Figure 2.17, provides 60dB of amplification as well as up to 110dB of attenuation. In addition to the signal conditioning, the device also contains a DC bias tee, allowing for the use of active antennas to further increase the amplification of the received signal.

Once the incoming signal is mixed down to baseband and separated into the I and Q components, it can be either streamed directly into a software receiver or stored as a complex array

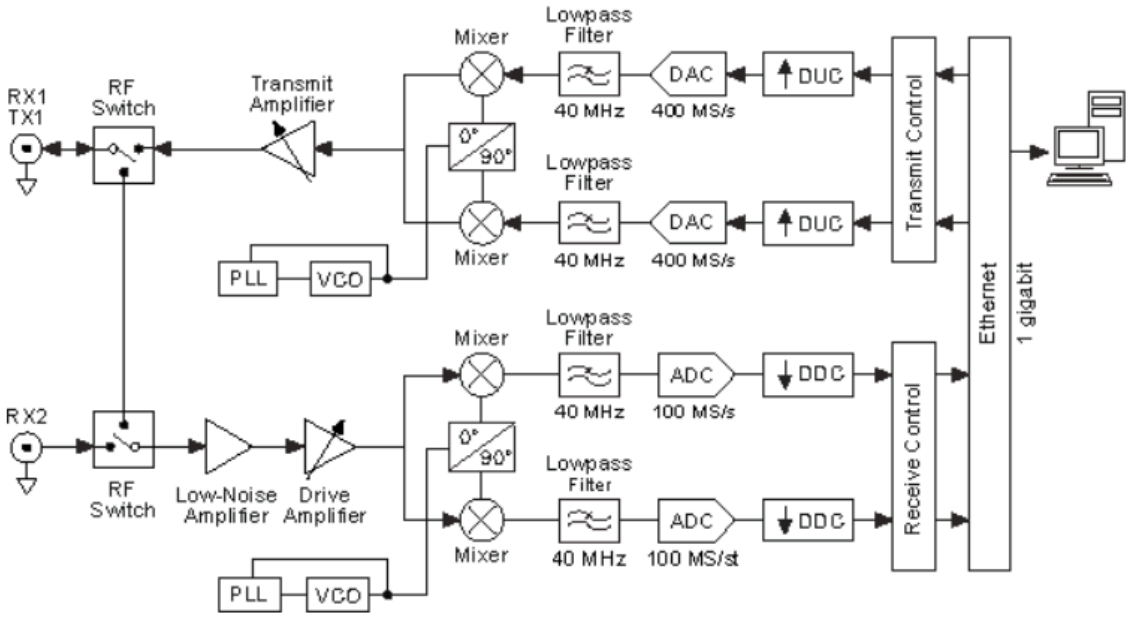

Figure 2.16: USRP Daughterboard Blockchain [61]

on the file system of the host computer. Recorded IQ files can be postprocessed using various software receivers in C++ and MATLAB, allowing for repeated use of test scenarios on various receivers using the replay functionality that the USRPs also offer. The errors introduced by the replaying of the signals are relatively low, allowing USRPs to be used for testing receivers in environments that may not have access to live sky signals [62].

Since the N210s provide the ability for both the reception and transmission of GNSS signals, they were used for both the role of the receiver as well as the meaconer/spoofer. The specifics of the testing configuration are detailed in Section 6.2. USRPs, along with various other SDRs, are supported by a software interface called GNURadio, an open source software that allows for control of the SDRs as well as DSP operations on a host computer. The USRP Hardware Driver (UHD) library provides the interface in GNURadio for configuration and data streaming of the N210 radios used for this thesis.

Figure 2.17: External Signal Amplification Box

2.8 Conclusion

GPS and other GNSS provide cheap and accurate access to geolocation and timing services for users across the globe. These services are used everywhere from financial institutions and telecommunications to autonomous vehicles and UAS platforms. Despite recent advances in GNSS integrity, the constellations are still as vulnerable as ever to intentional and unintentional interference such as jamming and spoofing. Spoofing mitigation methods have been explored, however no single method has proven to work in every scenario. The integration of inertial sensors to create a more robust navigation solution along with a method of spoofing mitigation using coupled external sensors is explored in the following sections.

Chapter 3

Inertial Navigation

With the advent of GNSS systems, many applications rely solely on a GNSS receiver for accurate position, navigation, and timing (PNT). In some cases, however, GNSS solutions are unavailable or inaccurate. Environments where tall buildings or other obstructions are present can prevent the use of GPS by limiting satellite visibility and increasing multipath, creating poor satellite geometry or introducing erroneous signal reflections into the system. These conditions lead to positioning errors and may create unsafe conditions for systems relying on accurate positioning. Non-GNSS navigation solutions are frequently explored for systems in challenging environments, such as lidar, radar or visual navigation. One solution for vehicles that operate in both good and poor environments is the use of inertial navigation systems (INS) that can be used in a standalone implementation or integrated into a combined architecture.

Inertial navigation utilizes self-contained sensors that do not rely on external signals to generate measurements. Since no external signals are needed, these units can operate in a multitude of conditions. IMU measurements have inherent errors, called biases, which causes the navigation solution to drift relative to the truth if uncorrected. This contrasts GNSS positioning, where the solution does not drift but relies on external signals that are not always available. Inertial sensors typically fall into two categories: accelerometers and gryoscopes (gyros), which measure specific forces and angular rates, respectively [63]. Inertial measurement units (IMUs) combine accelerometers and gyroscopes to provide six axis measurements ($x, y,$ and z specific forces and angular rates) of the vehicle. This style of IMU is also known as a strapdown inertial system, meaning that the sensor is rigidly mounted to the body whose dynamics are being

measured. Once the inertial measurements are output by the IMU, they must be rotated into the desired coordinate frame for navigation as described in Section 3.1.

Various grades of inertial sensors are available for navigational use. Typically, higher grade IMUs have smaller sensor errors associated with the measurements and can be used for longer periods of time without calibration or estimation of the errors. The most common grades of inertial sensors are navigation, tactical, and automotive. Other grades of IMUs are available, but are costly and typically reserved for specialized applications. Table 3.1 provides an overview of the sensor grades and typical errors that are seen with each.

Table 3.1: Typical Inertial Measurement Unit Parameters

IMU Grade	**Cost**	**Typical Error Magnitude**	**Typical Usage**
Automotive	$10	Large	Cars/Industrial Robots
Tactical	$10k	Moderate	UAS
Navigation	$100k	Small	Commercial aircraft

Navigation grade IMUs are commonly used in military and commercial aircraft applications, and can be purposed for precise standalone navigation due to the small errors present in the system. This solution can be accurate for much longer than lower grade IMUs, but over time the solution will drift, and the sensor will need to be recalibrated. These units are usually extremely expensive and large, limiting their use for automotive or UAS applications.

Tactical grade IMUs can also provide accurate standalone navigation, but only for a few minutes prior to recalibration. The errors present in tactical grade units have a significant impact after a short period of time, causing the solution to deviate from the true path. To improve this solution, these IMUs can be integrated with other devices through sensor fusion techniques. The most common approaches are visual-inertial and GPS/INS coupling. GPS/INS coupling techniques are discussed in more detail in the following sections. Tactical grade IMUs can be found in some military vehicles and unmanned aircraft applications.

Automotive grade IMUs are a low quality IMU, and are extremely cheap and widely used. These sensors have large measurement errors and are typically used in applications such as

airbags, pedometers, and other small devices where the solution is not used for overall navigation of a system [63]. Typically automotive grade IMUs are not combined with sensor aiding methods because the error growth is so rapid the solution drifts between corrections. These sensors are often integrated into attitude heading reference systems (AHRS) or pedestrian dead reckoning (PDR) systems, which require small, lightweight sensors [64]. These small, inexpensive IMUs have found their way into many other technologies such as mobile phones and entry level drones, that frequently couple the inertial and GPS solutions to correct the inertial errors.

This thesis explores the performance of automotive and tactical grade IMUs combined with GPS corrections using different integration techniques. The loose and tight coupling methods are discussed in Sections 3.4.1 and 3.4.2, respectively. Navigation grade IMUs are not considered here due to their significant cost and availability.

3.1 Coordinate Frames

Integrating multiple navigation systems presents a new challenge: determining the reference coordinate frame for the overall navigation system. Many sensors operate in unique navigation frames, and in order to to provide a common reference for navigation algorithms a common coordinate frame is typically used. There are several reference frames that are commonly used in navigation algorithms: the Earth Centered Inertial (ECI) frame, Earth Centered Earth Fixed (ECEF) frame, the geodetic frame, the local navigation frame, and the body frame. Each of these coordinate frames are discussed in further detail in the following sections, as well as methods for converting between frames. Some coordinate frames, like the wander azimuth frame, are not discussed in this thesis, but can be found in [63].

3.1.1 Earth Centered Inertial

An inertial frame is a nonaccelerating reference frame in which Newton's laws apply, denoted with the superscript i [65]. The axes of an inertial frame are three orthogonal direction vectors originating from an arbitrary point, with the x axis lying along the plane of the equator. In the context of navigation, an inertial frame is defined with the origin at the center of the Earth, and

the x axis begins at a specified reference point in time that does not rotate with the Earth. There are two conventional methods of initializing the x axis for navigation: initializing the axis at the beginning of navigation and initializing the axis at a reference epoch. A common reference used is the direction from the Earth to the Sun at the vernal equinox in the northern hemisphere [63]. The z axis is perpendicular to the x axis, out of the northern pole of the Earth, and the y axis is orthogonal to the x and z axes. Inertial sensors measure the motion of the body in an inertial frame, and utilizing the ECI frame allows straightforward inertial navigation algorithms to be employed.

3.1.2 Earth Centered Earth Fixed

Another commonly used reference frame is the Earth Centered Earth Fixed (ECEF) frame. Similar to the ECI frame, The origin of the ECEF frame is at the center of the Earth, and the z axis points out of the true North Pole. The x axis points out of the equator along the Prime Meridian, and the y axis is orthogonal to the other axes. The difference between the ECEF and the ECI frames is that, as the name implies, the ECEF frame is fixed to the Earth, rotating with the planet. An example of the ECEF frame is shown in Figure 3.1. The rotational rate of the Earth, given in Equation (3.1), indicates that the frame has acceleration, therefore it is not an inertial reference frame.

$$\omega_{ie}^{i} = \begin{bmatrix} 0 \\ 0 \\ 7.292115E-5 \end{bmatrix} rad/s \tag{3.1}$$

3.1.3 Geodetic Coordinates

Navigation systems such as COTS GPS receivers most frequently report the user position in terms of latitude, longitude, and altitude. Latitude, ϕ, is defined as the angle from the equator towards the North or South poles. Longitude, λ, is the angle from the Prime Meridian to the East or West of the meridian. The grid structure created by the lines of latitude and longitude are shown in Figure 3.2, with the latitude varying from $\pm 90^{\circ}$ and longitude varying from

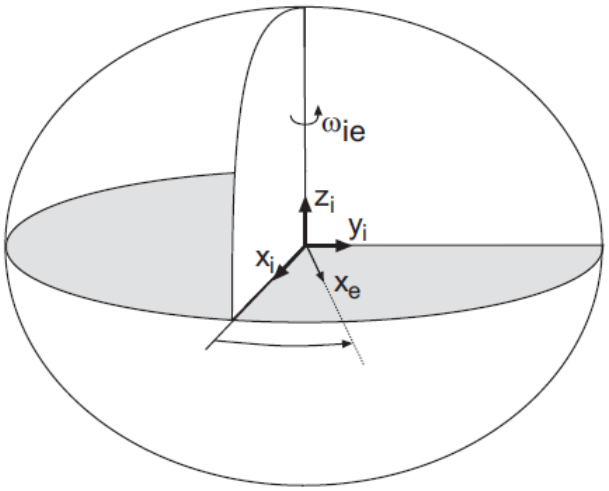

Figure 3.1: Earth Centered Earth Fixed Reference Frame [65]

$\pm 180°$. Finally, altitude is the distance above the ellipsoid, normal to the curvature of the surface. These coordinates, known as geodetic coordinates, are based on a reference system that accounts for the shape of the Earth. A reference ellipse approximates the surface of the Earth, or geoid, which has significant variation across different areas. The ellipsoid used to approximate the surface has a semi-major axis along the equator and the semi-minor axis across the poles. The geodetic model used most commonly in navigation is the World Geodetic Survey 1984 (WGS84) system. WGS84 defines four parameters, two of which are used to define the reference ellipse. The remaining parameters are calculated from the two ellipsoid variables, as shown in Equations (3.2) - (3.5).

$$\text{semi-major axis:} \quad a \tag{3.2}$$

$$\text{semi-minor axis:} \quad b \tag{3.3}$$

$$\text{eccentricity:} \quad e = \sqrt{\frac{a^2 + b^2}{a^2}} = \sqrt{f(2 - f)} \tag{3.4}$$

$$\text{flattening:} \quad f = \frac{a - b}{a} \tag{3.5}$$

Calculation of the latitude, longitude, and height from ECEF coordinates is performed using an iterative computation that requires an initial guess of parameters. This solution is

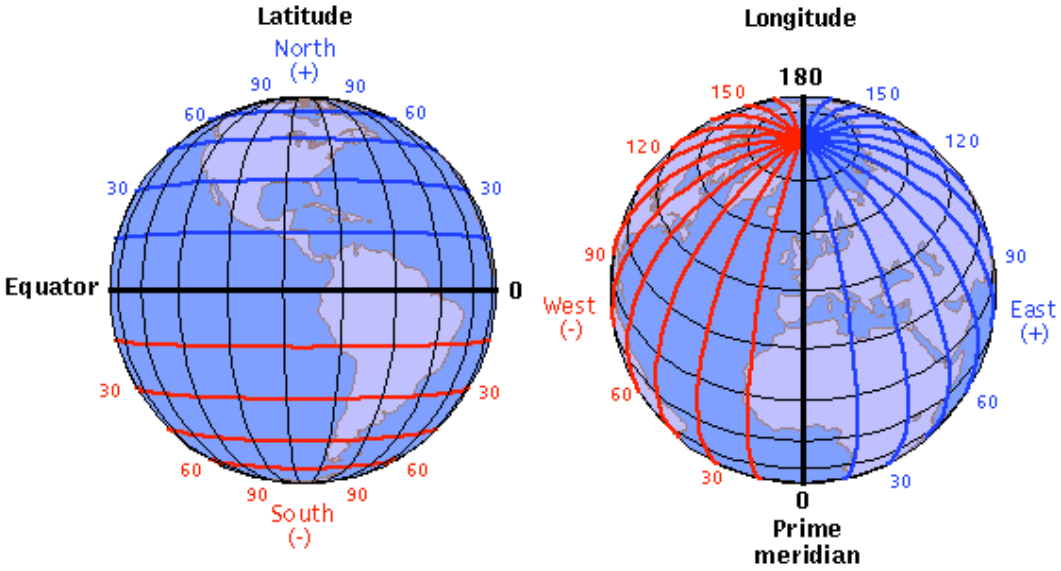

Figure 3.2: Geodetic Reference Frame [66]

iterated upon until the change in value is under a given threshold as determined by the user. Details for this calculation, along with several other frame transformations, are found in [63].

3.1.4 Local Frame Coordinates

A local navigation frame is used for applications that do not travel great distances from the origin of the frame or in enclosed spaces such as warehouses. The local frame is defined based on the Earth's reference ellipsoid in use. A conventional definition is the North-East-Down (NED) frame [63]. This defines the Down axis as perpendicular to the reference ellipsoid, pointing into the Earth. The North axis points in the direction of the geographic north pole, perpendicular to the down axis. The East axis is orthogonal to both of the other axes to ensure a right handed coordinate frame is used. An example of a local navigation frame is shown in Figure 3.3.

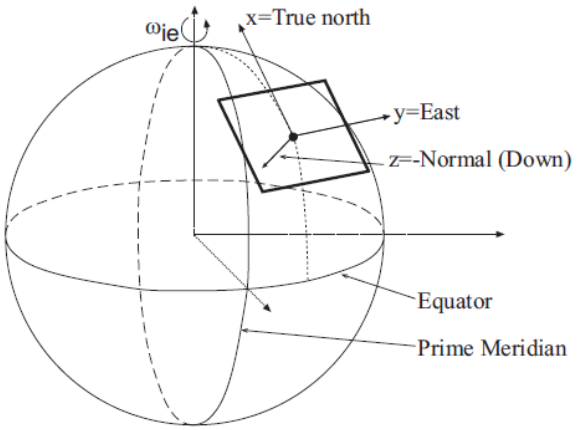

Figure 3.3: North-East-Down Reference Frame [65]

Other local navigation frames also exist and can be related to each other using rotation matrices in the same way as done to transform between global and local frames. The East-North-Up (ENU) frame, for example, is related to the NED frame as shown in Equation (3.6).

$$\vec{r}^{ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \vec{r}^{NED} \tag{3.6}$$

These frames provide users with navigation information that is referenced to logical directions. The local frames have several drawbacks however. One major assumption for a local navigation frame is that the gravity vector is aligned with the down axis. For areas close to the origin, this assumption holds. When the user gets farther away from the origin, however, the curvature of the Earth results in the gravity vector no longer aligning with the down axis, which can create issues if attempting to use inertial navigation in the local frame.

3.1.5 Body Frame Coordinates

In all navigation systems, sensors are mounted to a platform that is being localized. This could be a car, truck, bus, UAS, or any other object that can move relative to a reference frame. This object, known as the body, has a coordinate frame that is rigidly attached with the origin at the body center of gravity (CG) [65]. Any direction can be defined for the orthogonal set of axes, but typically the axes are defined with the x-axis out the nose of the body, the y-axis out the

right hand side, and the z-axis pointing down perpendicular to the u and v axes. In the example of a passenger car, the origin would be at the CG of the car, with the axes pointing out the hood, out the passenger side, and out of the floor. An example of a body frame coordinate system is shown in Figure 3.4.

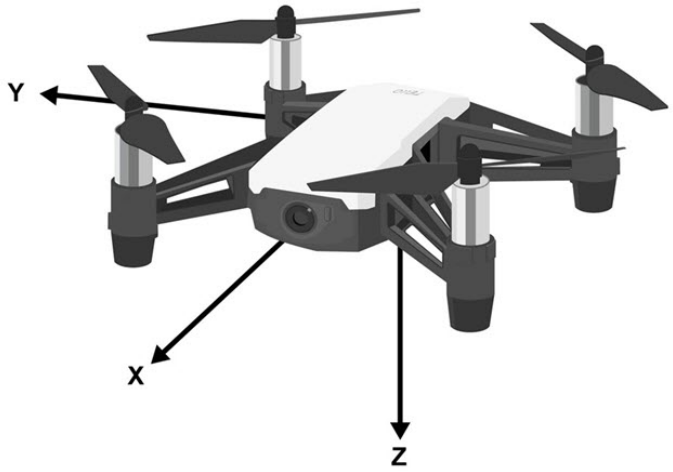

Figure 3.4: Body Reference Frame [67]

Rotations about the three axes (x, y, z) are defined as roll, pitch, and yaw, respectively. Rotation rates about these axes are commonly measured by an IMU when the sensor frame is aligned with the body frame, as discussed in the following section.

3.1.6 Sensor Frame Coordinates

Sensor frames are defined by the manufacturer and most often are indicated on the sensor enclosure. These indicated axes align with the sensitive axes in each sensor for which measurements are taken. Frequently the sensor frame and the body frame are aligned for simplicity, but this may not be possible in some cases. In the event that it is not possible to align the two frames, a fixed rotation matrix is determined to define the angles between the sensor axes and the body axes. The sensor is rigidly mounted to the body, so this rotation does not change once attached. Defining this orientation prior to system use allows for the necessary fixed rotations to be implemented in the navigation processor. An example where the sensor frame is not aligned with the body frame is shown in Figure 3.5.

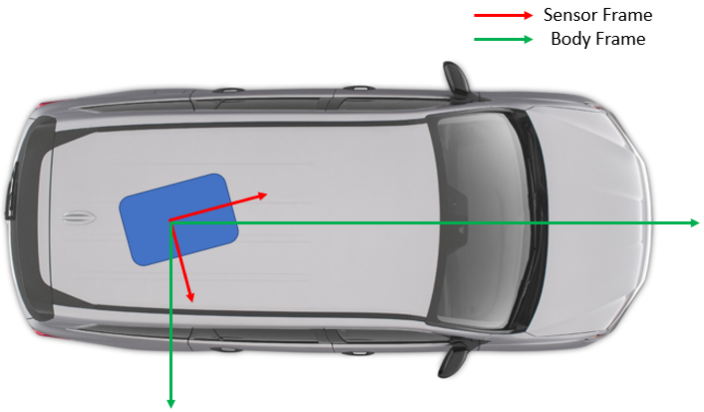

Figure 3.5: Sensor Reference Frame [68]

3.1.7 Coordinate Frame Transforms

In many navigation systems, sensor data will not all be reported in common coordinate frames. In the case of GPS and INS systems, GPS position solutions are commonly resolved in the ECEF frame, whereas IMU measurements are in the body frame. Methods for transforming measurements into different frames are needed to properly integrate the information from each sensor. To accomplish this, rotation matrices (also known as direction cosine matrices (DCM)) are used. The rotation matrix, R_a^b, denotes a rotation of a vector from initial frame a to final frame b as shown in Equation (3.7)

$$x^b = R_a^b x^a \tag{3.7}$$

The elements of the rotation matrix R relate to the angles that the vector x is rotated through to transform from one coordinate frame to the other. In a simple one rotation case as shown in Figure 3.6, the frame $[x', y', z']$ is rotated about the y axis by angle θ . The appropriate rotation matrix for this is shown in Equation (3.8).

$$R_{xyz}^{x'y'z'} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.8}$$

A unique principle of rotation matrices are that they are orthonormal, meaning that the inverse and the transpose of the matrix are identical. This property is particularly useful for

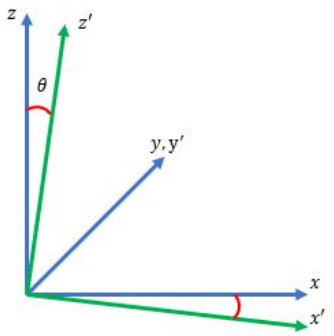

Figure 3.6: Single Axis Rotation

inverse rotations. In the case of the previous example, $R_{xyz}^{x'y'z'}$ was given as the rotation between the nonprime and prime reference frames. To transform a vector from the prime to the nonprime frame, the transpose of $R_{xyz}^{x'y'z'}$ is used, as shown in Equations (3.9).

$$\begin{aligned} \mathbf{v}' &= R_{xyz}^{x'y'z'}\mathbf{v} \\ \mathbf{v} &= (R_{xyz}^{x'y'z'})^T\mathbf{v}' \end{aligned} \tag{3.9}$$

Rotation matrices provide a mathematical method of determining the orientation of one frame with respect to another, but are not the most intuitive to look at or understand. To provide an intuitive method of describing orientation, Euler angles are frequently used. Euler angles, commonly denoted as roll (θ), pitch (ϕ), and yaw (ψ), describe the three body rotations about given axes. One key element to consider when using Euler angles to denote rotations is the order of the rotations. If a rotation is specified to be a 1-2-3 rotation (roll-pitch-yaw), the resulting orientation is not the same as if the body was rotated through a 3-2-1 rotation (yaw-pitch-roll). Therefore, care is needed when specifying rotations using Euler angles in navigation systems. Derivations of coordinate transformations and examples can be found in [63], [65].

3.2 Types of Inertial Sensors

When referring to a system as an inertial measurement unit, it indicates that the sensor contains either an accelerometer, a gyroscope, or both. These devices provide measurements of the

movement of the body with no external references, unlike a GNSS system or optical navigation system [63]. Strapdown INS have become the most common system, with the sensors rigidly mounted to the body in a known orientation, allowing for measurement of the body motion. Accelerometers, contrary to the name, measure specific force applied to the sensor, and gyroscopes measure angular rates of the body. As discussed in the beginning of the chapter, IMU errors (or the lack thereof) drive the performance of the sensor. Noise, bias, and random walk are a few of the errors present in inertial sensors, making them inadequate for long term standalone navigation. More details on each sensor and typical errors are discussed in the following sections.

3.2.1 Accelerometers

Accelerometers come in various configurations, but all serve the same purpose: measure the displacement of a proof mass with respect to a nominal position. In the example of a simple spring-mass accelerometer, the mass is suspended by two springs as shown in Figure 3.7 [63]. When the mass moves, the springs are stretched or compressed depending on the direction of motion. The displacement of the proof mass relative to the enclosure provides a measure of the applied acceleration. Gravitational forces, however, act on both the mass and the case, causing no motion between the two components [63]. Because of this, the resulting measurements are of specific force that do not include gravitational forces.

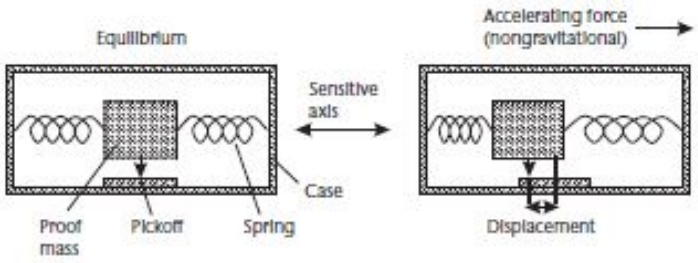

Figure 3.7: Basic Accelerometer [63]

Since accelerometers (and gyroscopes, as will be seen in the following section) are sensitive along a single axis, three sensors are mounted in orthogonal directions to provide observability along each sensor axis. Most accelerometers are marketed as a three-axis accelerometer

for this reason. While spring-mass accelerometers were discussed in this section, standard microelectromechanical systems (MEMS) sensors use crystal components to allow for reduced size and cost.

3.2.2 Gyroscopes

Gyroscopes, like accelerometers, come in various configurations to achieve the goal of measuring the angular rate of the sensor. The most basic spinning mass gyroscope is discussed here, while other variants include optical (ring laser gyroscope (RLG), fiber optic gyroscope (FOG)) and vibratory MEMS gyroscopes. More details on these sensors can be found in [63].

Spinning mass gyroscopes are constructed of a mount with a pivot fixed to the case, a spinning mass attached to the mount, springs to constrain the motion of the mount, and an apparatus to measure the displacement of the spinning mass. An example of a spinning mass gyroscope is illustrated in Figure 3.8. These simple gyroscopes operate based on the conservation of angular momentum. When the case is rotated, the mass inside will remain pointed in the original direction, creating a displacement between the nominal and rotated direction. In small enclosures, the spinning mass is kept aligned with the case by springs that provide a countertorque to an input torque applied to the case. The spinning mass and mount rotate about the sensitive axis until the torque is balanced by the springs in the case, at which point there is an angular displacement between the nominal direction and the final direction. This difference in direction is proportional to the angular rate about the input axis, as shown in Figure 3.8.

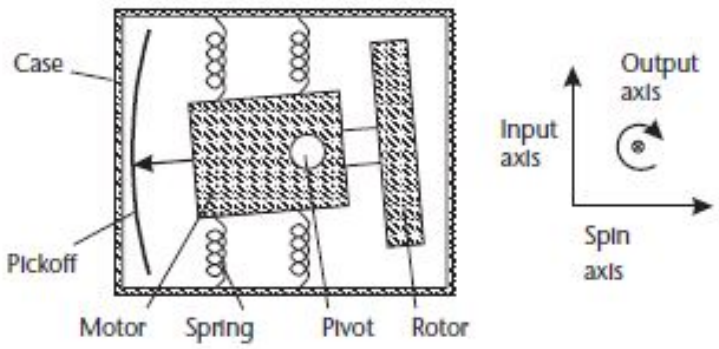

Figure 3.8: Basic Gyroscope [63]

In this work, MEMS gyroscopes are primarily used as they are the most commonly available gyroscope. A drawback to MEMS technology is that while inexpensive and compact, accuracies are degraded from significantly higher noise levels. Typical errors for both accelerometers and gyroscopes are discussed in the next section.

3.2.3 IMU Errors

Inertial sensors have several error sources that cause variations in the measurements output by the sensor. There are two different types of errors: deterministic and nondeterministic. The deterministic errors can be measured directly in a labratory or factory setting and can be reduced by calibration [63]. Nondeterministic errors are random errors that can vary from run to run or during the run, preventing calibration and removal of these errors.

Bias errors are often the largest present in the measurements from an IMU [69]. Biases can contain several different components: a deterministic fixed bias, bias stability, and bias instability. The fixed bias, b_{fb}, is typically modeled as a function of temperature and can be removed with sensor calibration. Bias stability, b_{bs}, also known as turn on bias, is constant during sensor operation but varies each time the sensor is powered on [63]. The bias instability, b_{bi}, can vary during the run, and is usually modeled as a random walk or a Markov Process [69]. The output of the inertial sensors are a combination of these biases, as shown in Equations (3.10) and (3.11).

$$\Delta f_b = b_a = b_{a,fb} + b_{a,bs} + b_{a,bi} \tag{3.10}$$

$$\Delta \omega_b = b_g = b_{g,fb} + b_{g,bs} + b_{g,bi} \tag{3.11}$$

Bias error magnitudes vary significantly based on the IMU grade being used. Units for accelerometer biases are commonly in milli-g (mg) and gyroscopes biases are reported in degrees per hour (deg/hr). The specifications typically encompass the nondeterministic errors, as the deterministic errors are calibrated and removed from the output by calibrations. Table 3.2 show typical IMU errors based on grade [63].

In addition to bias errors, inertial sensors can also have scale factor errors present. The scale factor is a linear approximation of the sensor input to output over a given input range.

Table 3.2: Typical IMU Biases [63]

IMU Grade	**Accelerometer Bias** (mg)	**Gyroscope Bias** (deg/hr)
Automotive	$>$10	$>$100
Tactical	1-10	1-100
Navigation	0.03-0.1	0.01

Scale factor error is the difference between the true relationship of the input and output and the reported relationship. The scale factor is represented as a coefficient on the true output to give the measurement. An example of sensor scale factor is shown in Figure 3.9 [69]. Misalignment and cross coupling errors also occur due to the angular offset between the ideal measurement axis and the true measurement axis, shown in Figure 3.10. This deterministic error is caused by manufacturing tolerances, and is generally given in milliradians [69]. Because the axis is not perfectly aligned, measured values would not strictly represent one direction as is the case in the ideal sensor. The scale factor and misalignment errors are represented in the matrix M, shown in Equation (3.12).

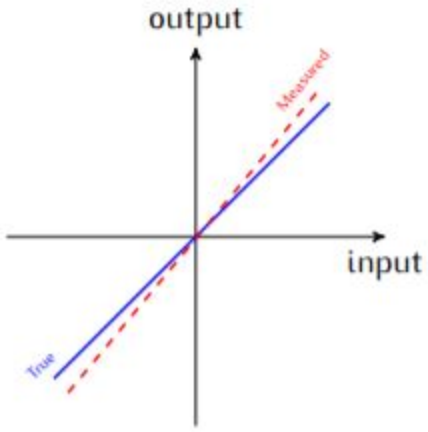

Figure 3.9: Scale Factor Error [69]

$$\Delta\vec{f} = \begin{bmatrix} s_{a,x} & m_{a,xy} & m_{a,xz} \\ m_{a,yx} & s_{a,z} & m_{a,yz} \\ m_{a,zx} & m_{a,zy} & s_{a,z} \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = M_a \vec{f}_{ib}^{b} \tag{3.12}$$

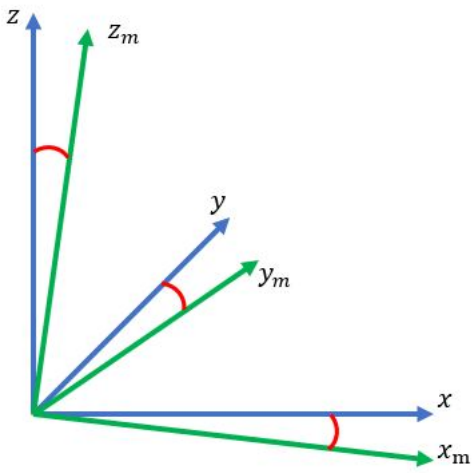

Figure 3.10: Misalignment Errors

Other noise associated with electrical components and ADCs is also present in IMU outputs. The random noise is modeled as white noise on the measured values [69]. White noise can come from several sources, such as electrical components, mechanical instabilities, and others [63]. The random noise on the sensors is denoted here as η, and is an additive to the measured output of each sensor. Other noises may be present, such as flicker noise, but are not explored here. Typical noise values for inertial sensors are shown in Table 3.3. In some references, the white noise errors are referred to as random walk errors. These noises are not a random walk on the measurements, but on the integrals of the measurements [63]. In this case, white noise on a measured specific force will be a random walk on velocity, and white noise on angular rates will be a random walk on the Euler angles. A more in depth discussion on power law noises in Chapter 4 with a focus on oscillators, however the same concepts apply to inertial sensors.

Table 3.3: Typical IMU Noises [63]

IMU Grade	**Accelerometer Noise** $(\mu g/\sqrt{Hz})$	**Gyroscope Noise** (deg/hr)
Automotive	1000	1
Tactical	100	0.03-0.1
Navigation	20	0.002

Other sources of noise include quantization from the IMU ADCs. Digital outputs are limited by the least significant bit (LSB) resolution, which depends on the range of values the sensor is configured to measure. Converting the analog measurements to digital representations causes error as the analog measurements are more precise than what is possible with the digital outputs.

Inertial sensor measurements are the sum of the true measured values and each of the error sources described above, outlined in Equations (3.13) and (3.14).

$$\tilde{\vec{f}}_{ib}^{b} = \vec{f}_{ib}^{b} + \Delta\vec{f}_{ib}^{b} = \vec{b_a} + (I_3 + M_a)\vec{f}_{ib}^{b} + \vec{\eta}_a \tag{3.13}$$

$$\tilde{\vec{\omega}}_{ib}^{b} = \vec{\omega}_{ib}^{b} + \Delta\vec{\omega}_{ib}^{b} = \vec{b_g} + (I_3 + M_g)\vec{\omega}_{ib}^{b} + \vec{\eta}_g \tag{3.14}$$

Here, the tilde is used to represent the measurements obtained from the inertial sensors, and the values with no tilde represent the true values sensed by the IMU. Without any method of predicting the errors present in the INS, the standalone estimate will diverge from the true solution at a rate that depends on the error magnitudes present.

INS mechanization and error estimation using sensor fusion is discussed in the following sections. These sections explain how IMU errors are estimated and compensated for to improve the output of the combined navigation system.

3.3 Sensor Mechanization

INS outputs provide a navigation solution based on the measurements from an IMU [63]. An INS contains an IMU which provides the measurements used for navigation and a navigation computer, which is responsible for mechanizing the IMU measurements to provide a dead reckoned estimate of position, velocity, and attitude (PVA). The mechanization process consists of a four step operation that involves updating the attitude estimate, rotating the specific force measurements into the desired navigation frame, updating the velocity estimate, and finally updating the position estimate [63]. An additional step for the transformation of specific force measurements to accelerations requires a gravity model and will be discussed as well. The

mechanization process varies slightly depending on the navigation reference frame, and in this thesis is done in the ECEF frame for reasons to be discussed in the following section. Other variations, such as the ECI or local frame mechanization, can be found in [63] and [69]. Figure 3.11 shows a block diagram representation of the ECEF mechanization process that has been adapted from [69].

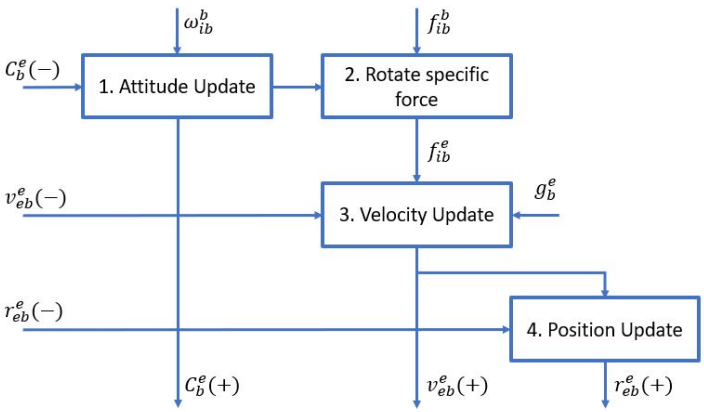

Figure 3.11: ECEF IMU Mechanization Process [69]

Prior to the start of the mechanization process, the initial attitude alignment must be determined. There are numerous research topics exploring the best methods for attitude initialization, and several methods are discussed in [63], [65], [69]. For this work, it is assumed that successful initialization was done prior to the start of the mechanization for simplicity.

To correctly mechanize the accelerometer measurements, a gravity model is needed to account for the force of gravity in the transformation of the specific forces to accelerations. Several models are available, and the Somigliana model is applied in this thesis. This model uses the WGS84 reference to provide a gravity vector as a function of the current latitude, as shown in Equation (3.15) [63].

$$g_0(\Lambda) \approx 9.7803253359 \frac{(1 + 0.001931853 \sin^2(\Lambda)}{\sqrt{1 - e^2 \sin^2(\Lambda)}} \tag{3.15}$$

Once the gravity has been computed, a gravitational model provides the desired values of gravitational acceleration. The gravitational model is different from the gravity model as it accounts for the centrifugal acceleration of the Earth, giving the total acceleration due to the

Earth at a given latitude. The geocentric radius, r^e_{eS}, is calculated as a function of latitude as shown in Equation (3.16), and is used to calculate the gravitational acceleration, γ_0, as shown in Equation (3.17).

$$\mathbf{r}^e_{eS}(\Lambda) = R_E(\Lambda)\sqrt{\cos^2(\Lambda) + (1-e^2)^2\sin^2(\Lambda)} \tag{3.16}$$

$$\gamma_0(\Lambda) = g_0(\Lambda) + \Omega_{ie}\Omega_{ie}\mathbf{r}^e_{eS}(\Lambda) \tag{3.17}$$

After the gravitational acceleration has been determined, it is scaled with height as shown in Equation (3.18) [63].

$$\gamma_{ib} = \frac{(r^3_{eS}(\Lambda))^2}{(r^3_{eS}(\Lambda)+h)^2}\gamma_0(\Lambda) \tag{3.18}$$

For vehicles on the surface of the Earth, this scaling is not necessary, but for aerial platforms such as UAS, the scaling can change the calculated value significantly depending on the altitude of operation. Once this has been calculated, the acceleration due to gravity is computed using Equation (3.19).

$$g_b = \gamma_{ib} - \Omega_{ie}\Omega_{ie}\mathbf{r}_{eb} \tag{3.19}$$

Before the acceleration due to gravity can be added to the specific forces, the specific forces must be rotated into the desired frame. To do this at a given time period, the rotation matrix C^e_b must first be updated using the rotational rates measured by the gyroscope. The rotation rates are integrated to determine the change in angle over a given time period, Δt. Assuming the measured rotation rates do not change over the integration period, the attitude update step can be done using Equation (3.20) [63].

$$C^e_b(+) = C^e_b(-)(I_3 + \Omega^b_{ib}\Delta t) - \Omega^e_{ie}C^e_b(-)\Delta t \tag{3.20}$$

The skew symmetric matrix of the body rotation rates, Ω_{ib}^{b}, is given in Equation (3.21), and the skew symmetric matrix Ω_{ie}^{e} contains the rotational rate of the Earth.

$$\Omega_{ib}^{b} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.21}$$

Once the rotation from the body frame to the Earth frame has been updated, the specific forces are transformed into the Earth frame using the new rotation matrix. This step is straightforward, and only requires the multiplication of the rotation matrix and the specific forces in the body frame. To increase accuracy, the current and previous rotation matrices can be averaged and utilized to rotate the specific forces into the Earth frame, as shown in Equation (3.22).

$$f_{ib}^{e} = C_{b}^{e}(+) f_{ib}^{b} \approx \frac{1}{2}[C_{b}^{e}(-) + C_{b}^{e}(+)] f_{ib}^{b} \tag{3.22}$$

Updating the velocity in the Earth frame requires the revisitation of the gravitational acceleration model discussed earlier in this section. Now that the body accelerations have been rotated into the Earth frame, the gravitational accelerations can be added to get the overall body acceleration in the Earth frame and integrated to update the velocity estimate. The body acceleration in the Earth frame is shown in Equation (3.23), and the numerical integration to update velocity is shown in Equation (3.24) [70].

$$a_{eb}^{e} = -2\Omega_{ie}^{e} v_{eb}^{e} + f_{ib}^{e} + g_{b}^{e} \tag{3.23}$$

$$v_{eb}^{e}(+) = v_{eb}^{e}(-) + a_{be}^{e} \Delta t \tag{3.24}$$

The position update step is again a simple numerical integration, however it must be noted which values are being used for the integration. If the updated velocity value is used to update the position, the new acceleration has already been accounted for and an acceleration term is not added to this step of the update. Both methods of updating position are shown in Equations

(3.25) and (3.26) [70].

$$r^e_{eb}(+) = r^e_{eb}(-) + v^e_{eb}(-)\Delta t + a^e_{eb}\frac{\Delta t^2}{2} \tag{3.25}$$

$$r^e_{eb}(+) = r^e_{eb}(-) + v^e_{eb}(+)\Delta t \tag{3.26}$$

In a perfect world where IMUs reported errorless measurements, the INS outputs would be the exact motion and velocity of the body it was attached to. However, as discussed previously, sensor noise corrupts the IMU measurements, leading to errors throughout the mechanization process. The errors are also double integrated, causing the position estimates to drift away from the true solution at a rate proportional to the magnitude of the errors. Figure 3.12 shows a comparison of INS outputs with varying grades of inertial sensors using simulated inertial measurements. The truth data shown in black was taken from Airsim and errors were added as is discussed in Chapter 6. It can be seen that the high performance navigation grade IMU lies closely to the true solution, and as IMU quality decreases the mechanized solution drifts further from the truth over time.

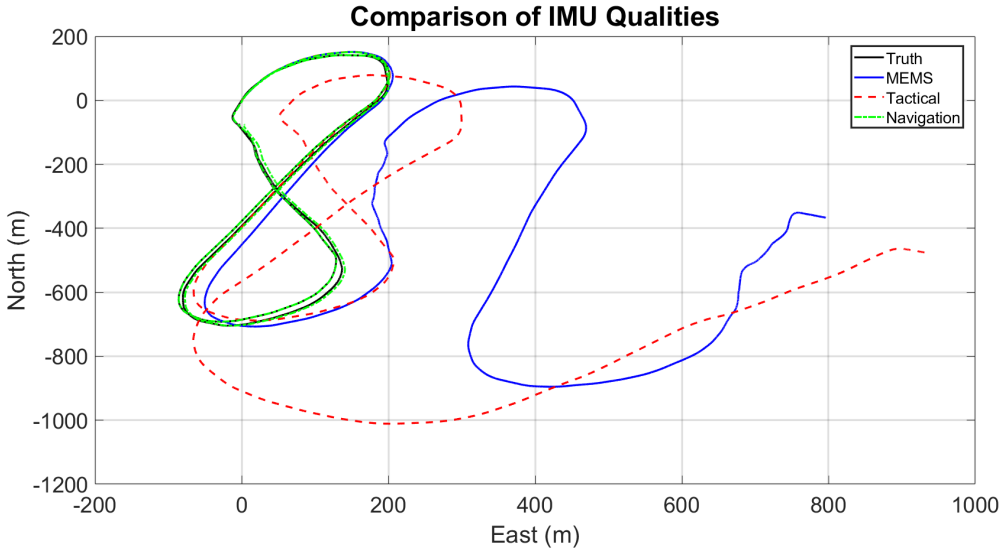

Figure 3.12: INS Mechanized Navigation Solution With Various Sensor Grades

Without correction of the errors, the INS solution will degrade over time and become unusable. Often times, an integrated navigation system is used to constrain and correct inertial errors using outside sources such as GNSS. Other methods include vision-aided inertial systems

or systems that use non-GNSS signals of opportunity such as WiFi or 4G/5G as ranging signals. GNSS/INS integration is discussed in detail in the next section.

3.4 GNSS-Aided Inertial Navigation

As mentioned in the previous section, standalone INS solutions can be accurate for a period of time after alignment, but drift away from the true solution unless constrained or corrected by an external sensor. GNSS receivers are able to provide a position solution to within a few meters under most conditions and do not drift over time [63]. GNSS signals are extremely weak, however, and solutions can be degraded or have outages in urban environments or areas such as tunnels. By combining GNSS/INS systems, the long term errors of the INS can be constrained while outages in the GNSS solution can be filled by standalone INS over short time periods. In addition to these benefits, conventional GNSS receivers do not provide a measure of the body attitude, so the INS provides additional state information as well.

To formulate the GPS/INS integration, an error state Kalman filter is constructed. The errors in the position, δr^e_{eb}, velocity, δv^e_{eb}, and attitude, $\delta \Psi^e_{eb}$ are augmented with the estimates of the accelerometer and gyroscope biases, δb_a and δb_g. The complete error state vector is shown in Equation (3.27), which contains the errors in the PVA of the INS output and are used to correct the INS output.

$$x^e_{INS} = \begin{bmatrix} \delta \Psi^e_{eb} & \delta v^e_{eb} & \delta r^e_{eb} & \delta b_a & \delta b_g \end{bmatrix}^T \tag{3.27}$$

During periods between measurements from the GNSS receiver, the estimated errors are propagated forward in time using the Kalman Filter state transition matrix for the system. To derive the state transition matrix, the system dynamics are modeled as shown in [63]. The

continuous time system matrix is given in Equation (3.28).

$$F^e_{INS} = \begin{bmatrix} -\Omega^e_{ie} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \hat{C}^e_b \\ F^e_{21} & -2\Omega^e_{ie} & F^e_{23} & \hat{C}^e_b & \mathbf{0}_3 \\ \mathbf{0}_3 & I_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \tag{3.28}$$

The components F^e_{21} and F^e_{23} are described in Equations (3.29) and (3.30), respectively, where the $\wedge$ represents the skew symmetric form.

$$F^e_{21} = \frac{2g_0(\Lambda)}{r^e_{eS}(\Lambda)} \frac{\hat{r}^e_{eb}}{|\hat{r}^e_{eb}|} (\hat{r}^e_{eb})^T \tag{3.29}$$

$$F^e_{21} = [-(\hat{C}^e_b] f^e_{ib})\wedge] \tag{3.30}$$

The system matrix is a continuous time representation of how the states progress over time. to discretize the state transition matrix, a power series expansion is used [63]. Once the matrix is represented in discrete time, it can be implemented in the Kalman filter model. For this thesis, the first term in the expansion is used, given by Equation (3.31).

$$\Phi_{INS} = I_{15} + F^e_{INS} dt \tag{3.31}$$

The Kalman filter architecture also requires a process noise covariance matrix to be specified. The process noise covariance matrix, Q, consists of the power spectral densities of the gyro and accelerometer random noise as well as gyro and accelerometer bias variations [63]. These quantities are given by $\eta_{rg}, \eta_{ra}, \eta_{ba}$, and η_{bg}, respectively. In this case, it is assumed that

all three axes have equivalent errors. The Q matrix is given in Equation (3.32).

$$Q_{INS} = Q_c dt = \begin{bmatrix} \eta_{rg}^2 I_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \eta_{ra}^2 I_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \eta_{ra}^2 I_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \eta_{ra}^2 I_3 \end{bmatrix} dt \tag{3.32}$$

Using the state transition matrix and the process noise covariance matrix, the state vector x is propagated from the point of the last measurement update to the current epoch. The time update step discussed here is common across both the loose and tight coupling methods, which are discussed in the following section. Measurement updates are conducted when GNSS measurements are received and new estimates of errors can be generated. Loosely coupled GNSS/INS relies on estimates of position and velocity from the GNSS navigation filter, whereas tightly coupled GNSS/INS relies on pseudorange and pseudorange rate residuals to correct the error states. More details about each integration method are discussed in the following sections.

3.4.1 Loosely Coupled GNSS/INS

The basic structure for a loosely coupled (LC) GNSS/INS algorithm uses position estimates from both the GNSS and INS estimators. This integration is useful when working with "black box" standalone navigation solutions - that is, the only outputs available are the current position estimates. By comparing the accurate but infrequent GNSS solution with the high frequency but drifting inertial solution, position errors and inertial sensor biases can be estimated, allowing for accurate positioning between GNSS updates. The loosely coupled GNSS/INS algorithm architecture used in this thesis is shown in Figure 3.13 [71].

In the architecture shown above, each navigation system operates independently and the position solutions are blended in the GNSS/INS filter. Prior to estimating the error states given in Equation (3.27), the current position is estimated by each system at the current epoch. For the GNSS filter, this can be done using a least squares solution as described in Chapter 2. The inertial position estimate is propagated using the mechanization equations described in the

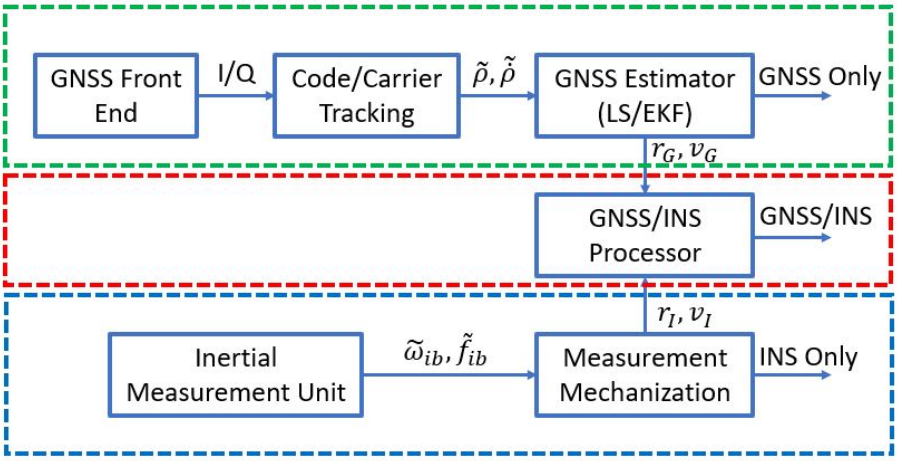

Figure 3.13: Loosely Coupled GNSS/INS Architecture

previous sections in this chapter. In this thesis the ECEF frame is used to allow for a simple comparison to the solution estimated by the GNSS processor. Since the INS has a much higher output rate than the GNSS system, the IMU measurements are mechanized between periods where a GNSS update is used to re-estimate the corrections to the INS.

GNSS/INS filters are most often implemented using an EKF architecture. During the periods between measurements, the estimates of INS errors and state uncertainty are propagated using the state transition matrix given in Equation (3.28). When GNSS measurements are received, the position estimated by the GNSS processor and the position estimated by the INS mechanization are differenced. To account for the difference in position between the GNSS antenna and the IMU location, the lever arm l^b_{ba} between the two positions is included in the calculation [63]. The lever arm is rotated from the body frame to the Earth frame using the rotation matrix calculated in the attitude update step of the INS mechanization. Once the measured difference has been calculated, the difference between the measured and the estimated error states is taken. This innovation model seen in Equation (3.33), is multiplied by the Kalman gain calculated to estimate the new corrections based on the confidence bounds for the INS and GNSS systems [63].

$$\delta z = \begin{bmatrix} \hat{r}^e_{eaG} - \hat{r}^e_{eb} - \hat{C}^e_b l^b_{ba} \\ \hat{v}^e_{eaG} - \hat{v}^e_{eb} - \hat{C}^e_b(\hat{\omega}^b_b \times l^b_{ba}) + \mathbf{\Omega}^e_{ie}\hat{C}^e_b l^b_{ba} \end{bmatrix} \tag{3.33}$$

For the loosely coupled architecture, the state measurement model is straightforward. As the loosely coupled integration uses only position differences, the measurement model is derived by taking the Jacobian of the GNSS measurements. The full derivation of the measurement model can be found in [63], however it can be simplified as lever arm coupling between attitude errors and gyroscope biases are weak and can be neglected. The simplified measurement model is shown in Equation (3.34).

$$\mathbf{H} = \begin{bmatrix} 0_3 & 0_3 & \mathbf{I}_3 & 0_3 & 0_3 \\ 0_3 & \mathbf{I}_3 & 0_3 & 0_3 & 0_3 \end{bmatrix} \tag{3.34}$$

The measurement noise covariance, R_G is described in several ways depending on the information that is available from the GNSS receiver in use. Using the estimated uncertainty from the GNSS processor, P_{GNSS}, allows the GNSS/INS algorithm to weight the GNSS information according to the confidence output by the receiver [63]. This information is available from some COTS receivers or if the user has designed a navigation filter that does not rely on the position outputs of a COTS receiver but rather uses the raw observables. In the case that this metric is unavailable, the noise covariance can be defined as a function of the carrier to noise ratio of each measured GNSS signal, satellite geometry, and in high dynamic applications, acceleration. In this thesis, the measurement noise covariance is constructed using the first method, and is generated based on the uncertainty in the GNSS measurements as given by the GNSS positioning algorithm.

3.4.2 Tightly Coupled GNSS/INS

As opposed to the loosely coupled integration described above, the tightly coupled (TC) algorithm does not use position and velocity outputs from the GNSS and INS algorithms. Instead, the tightly coupled algorithm joins the GNSS positioning algorithm with the inertial mechanization process. Described briefly in [71], the GNSS algorithm estimates a standalone position, instead providing the pseudorange and pseudorange rate measurements directly to the tightly coupled GNSS/INS processor. A graphical overview of the tightly coupled algorithm is shown

in Figure 3.14 below. In addition to estimating position, velocity, and INS errors directly in the coupled algorithm, receiver clock states must be estimated as well. The clock states, cb and $c\dot{b}$ are appended to the end of the state vector shown in Equation (3.27), creating the new state vector given in Equation (3.35).

$$x^e_{INS} = \begin{bmatrix} \delta\Psi^e_{eb} & \delta v^e_{eb} & \delta r^e_{eb} & \delta b_a & \delta b_g & cb & c\dot{b} \end{bmatrix}^T \tag{3.35}$$

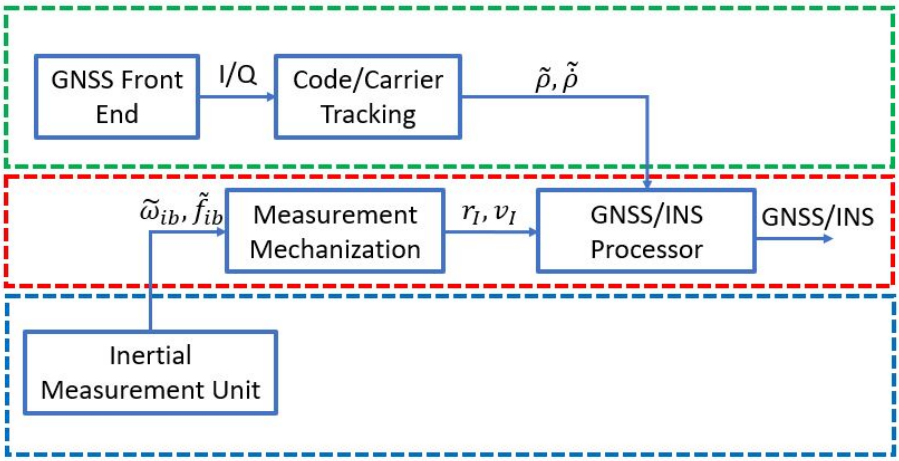

Figure 3.14: Tightly Coupled GNSS/INS Architecture

To form the innovation vector for the tightly coupled algorithm, measured pseudoranges and pseudorange rates are compared to the estimated observables for each satellite[63]. This process is identical to that employed for standalone GNSS positioning. The measured pseudoranges are corrected using the satellite clock correction terms calculated from the broadcast ephemerides as discussed previously, and the estimated pseudoranges are calculated using the satellite position and the current user position and clock bias estimate from the coupled algorithm. Estimates of pseudorange rate and clock drift are calculated using the coupled estimate of velocity in addition to user position. The process of calculating the estimated values is shown in Equation (3.36) and Equation (3.37), and the innovation vector is given by Equation (3.38).

$$\hat{\rho}_I = r_{SV} - r_{u,I} + cb \tag{3.36}$$

$$\hat{\dot{\rho}}_I = (v_{SV} - v_{u,I}) \cdot \bar{\mathbf{u}}_I \tag{3.37}$$

$$\delta z = \begin{bmatrix} \tilde{\rho}_G - \hat{\rho}_I \\ \tilde{\dot{\rho}}_G - \hat{\dot{\rho}}_I \end{bmatrix} \tag{3.38}$$

Values denoted with a subscript G are calculated from GNSS parameters, and values denoted with a subscript I are calculated from the INS estimates. The unit vector, $\bar{\mathbf{u}}_I$, is the pointing vector from the estimated user location to the satellite location, calculated for each satellite in view.

The tightly coupled measurement matrix is obtained by taking the partial derivative of the measurements with respect to the states. The full derivation can again be found in [63], with the final form sharing properties with a standard GNSS measurement model given in Equation (2.14) with additional columns for the heading and inertial error states. Similar to the loosely coupled model, the attitude and gyroscope errors are neglected, leading to the model shown in Equation (3.39) [63].

$$\mathbf{H} = \begin{bmatrix} 0_{1,3} & 0_{1,3} & \bar{\mathbf{u}}_{I,1} & 0_{1,3} & 0_{1,3} & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0_{1,3} & 0_{1,3} & \bar{\mathbf{u}}_{I,n} & 0_{1,3} & 0_{1,3} & 1 & 0 \\ 0_{1,3} & \bar{\mathbf{u}}_{I,1} & 0_{1,3} & 0_{1,3} & 0_{1,3} & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0_{1,3} & \bar{\mathbf{u}}_{I,n} & 0_{1,3} & 0_{1,3} & 0_{1,3} & 0 & 1 \end{bmatrix} \tag{3.39}$$

The size of the observation matrix be $2n \times 17$, where n depends on the number of GNSS observations that are received at each time period.

Unlike the loosely coupled implementation, there is no output of receiver confidence in the GNSS measurements as no position solution has been calculated prior to the coupling. Due to this, the measurement noise covariance is based on the measured C/N_0 as well as body acceleration if under high dynamics. This matrix, R_G, is be identical to that used in the GNSS Kalman filter derivation, and can be seen in Equation (2.29).

3.5 Conclusion

Inertial navigation systems provide redundant backups to GNSS receivers for varying durations of time based on the quality, however they are not able to provide a standalone navigation solution without periodic corrections. By coupling GNSS with INS, the navigation processor can span periods of GNSS outages while correcting inertial errors periodically. By combining the GPS/INS algorithm with the spoofing detection method described in Chapter 5, faults can be detected and excluded from the overall navigation process.

Chapter 4

Oscillators and Timing

As long as history has been recorded, time has been measured in some way by humans. In ancient times, the seasons told the farmers when crops needed to be planted or harvested. Timekeeping devices ranged from the simple sundial for daily time to the massive Taosi site in ancient China used to tell seasons [72]. These early devices relied primarily on the angle of the sun to tell the time of day or the months of the year. It was quickly determined, however, that the sun was not reliable enough as the need for accurate time grew. Now, GNSS timing is used for references in industry ranging from the financial sector to the power grid and telecommunications. Various types of oscillators are used in nearly every electronic instrument constructed, with the application determining the required frequency stability. This section will discuss types of common oscillators and methods of frequency stability analysis, as well as the simulation of oscillators for GNSS or other applications.

4.1 Types of Oscillators

The most common types of oscillators in use today by far are crystal oscillators. These oscillators are present in nearly any electronic circuit requiring timing, from calculators to watches to GPS receivers. These crystal oscillators rely on the piezoelectric vibrations of a specially cut piece of quarts to keep track of time. For more precise timing applications, atomic clocks such as Rubidium (Rb) or Cesium (Cs) standards are commonly implemented, which rely on properties of the atoms used to measure time. Both of these types of oscillators are discussed further in this section, with details on the performance of each discussed later.

4.1.1 Crystal Oscillators

As radios and other electronic communications began to expand into everyday use, problems with frequency references began to appear. The National Bureau of Standards (NBS, now National Institute of Standards and Technology, or NIST) began to study oscillators using quartz crystals as the driving element. It was discovered that the quartz oscillators would be able to keep the time required for radio frequency use, and a new program was launched into researching high quality crystal oscillators.

As the uses for precise time continued to grow and evolve, so did the types of crystal oscillators that were available for use. The original TCXOs delivered to NBS were large and stationary, but as electronics began to shrink and become portable, so did the reference oscillators used in the devices. Mobile electronics such as cell phones, GPS receivers, and more rely on compact crystal oscillators that can be synchronized to external sources. Table 4.1 provides an overview on different types of crystal oscillators. Typically, lower end crystal oscillators are used in consumer electronics due to their low size, weight, power, and cost (SWaP-C), whereas oven controlled crystal oscillators (OCXOs) are found in precision timing equipment and occasionally as telecommunication standards where size and power limitations are not a consideration.

Table 4.1: Types of Crystal Oscillators

Crystal Oscillator Type	**Operation Principle**
Crystal Oscillator (XO)	Voltage applied to quartz crystal induces piezoelectric vibrations
Voltage Controlled XO (VCXO)	Voltage applied to crystal is variable to allow for tuning of frequency
Temperature Compensated XO (TCXO)	Voltage is controlled by a circuit with thermal compensation to account for ambient conditions
Oven Controlled XO (OCXO)	Crystal is enclosed in an enclosure where temperature is raised above ambient to reduce thermal effects
Double Oven Controlled XO (DOCXO)	Same as an OCXO, but with a second oven enclosing the first

Crystal oscillators are able to achieve good stability and maintain accurate time and frequency references, but are not able to match the performance of atomic frequency standards.

Discussed in Section 4.1.2, atomic oscillators have come to the forefront of precision time and frequency references.

4.1.2 Atomic Oscillators

In 1949, NIST announced the creation of the first atomic clock using a quartz oscillator as a frequency reference to interrogate ammonia atoms in a resonant chamber. The original atomic clocks were able to achieve performance that was comparable to existing crystal standards, but did not offer much over the quartz standards already in production [73]. Soon after, work began on a cesium standard that would become known as NBS-1, the first atomic frequency standard used by NIST.

Atomic clocks work on the theory of observing the hyperfine transition between two ground states of an atom, typically Rubidium, Cesium, or Hydrogen [74]. A crystal oscillator generates a reference signal for a microwave cavity or laser, and the energy is applied to the atoms. The error in the frequency reference is measured by a physics package, and periodic corrections are applied to the reference to lock it to the atomic resonant frequency. This creates an extremely stable frequency output as the crystal reference is driven by the accuracy of the atomic frequency, not by the accuracy of the crystal reference [74].

By locking the output frequency to the generated frequency of the atomic hyperfine transitions, atomic clocks are able to achieve stability levels that can not be achieved by conventional quartz references. Table 4.2 shows the most common types of atomic oscillators and the timing performance that can be achieved compared to a standard TCXO.

Table 4.2: 1-Day Timing Errors of Common Oscillators [74]

Oscillator Type	**1-Day Timing Error** (μs)
TCXO	100000
Rubidium	100
Cesium	0.1
Hydrogen Maser	0.01

A newer method of interrogation of atoms, known as coherent population trapping, has led to the miniaturization of atomic oscillators [74]. Oscillators such as the SA.5X Miniature

Atomic Clock (MAC) or the SA.45s Chip Scale Atomic Clock (CSAC) are two such examples. Both of these oscillators have low SWaP that can be comparable or better than low cost crystal oscillators, but have performance that can exceed many of the same oscillators. These new atomic oscillators allow for battery operable atomic performance, expanding the availability of accurate time to mobile applications where power requirements previously prevented their use.

Prior to 2001, atomic clocks were large, power hungry, and difficult to transport. If accurate timing was wanted in portable applications, the task primarily fell on OCXOs. These oscillators exhibit exceptional short term stability, but can use as much as 13W during warm up, and warm up times can last more than ten minutes [75]. For many applications requiring portable oscillators, the size, weight, and power (SWaP) plays a major role. Applications such as autonomous vehicles or underwater seismographs require reference oscillators to be compact, lightweight, and use minimal power. In 2001, the Defense Advanced Research Projects Agency (DARPA) issued a call for a battery operable atomic clock. Several companies made progress on this development, but in 2011, the first commercially available chip scale atomic clock was released [76].

The miniaturization of atomic timekeeping allows users to obtain accurate time while mantaining low SWaP in the deployed systems. CSACs are able to operate for extended periods of time off battery power thanks to the extremely low power consumption during operation [77]. The short and long term stability of the atomic physics package makes the CSAC an ideal candidate for systems which require accurate time at a low cost, and are frequently deployed in underwater applications where GPS timing is not available [78],[79]. Shown in Figure 4.1, the CSAC package is only $1.6" \times 1.39" \times 0.46"$, making for an extremely compact package when compared to other atomic frequency references.

Figure 4.1: Microsemi Chip Scale Atomic Clock [77]

In addition to the size and power improvements of the CSAC, the stability of the clock is a large benefit to many users as well. When compared to other oscillators in the same size and power category, the CSAC outperforms nearly all contenders. Table 4.3 provides a comparison between a typical TCXO, OCXO, CSAC, and Rb standard. The frequency stability of the CSAC is much better than standard TCXOs, while using far less power than an OCXO. In addition to good short term stability, the CSAC exhibits superior long term stability compared to OCXOs.

Table 4.3: Performance Comparison of Common Oscillators

Oscillator	Volume (in^3)	Power (Operating) (W)	ADEV (@1s)	Aging
TCXO	0.08	0.01	N/A	4.2E-7 /mo
OCXO	58.86	3.5	3E-13	1.5E-9 /mo
CSAC	1.02	0.125	3E-10	9E-10 /mo
Rubidium	386.75	50	2e-11	5e-11 /mo

In addition to the properties listed above, the CSAC has much better performance under conditions where temperature changes or vibrations occur. TCXOs are highly sensitive to ambient thermal conditions, and both TCXOs and OCXOs are influenced by vibrations and oscillator orientation. Atomic clocks are much less susceptible to these effects, giving the CSAC another advantage over conventional oscillators. The CSAC is a good blend of all the desired characteristics of oscillators, and is the ideal candidate for aiding mobile GPS receivers. Due to the improved frequency stability, modeling techniques can be used to predict the clock errors during periods of GPS outages, allowing for navigation with three satellites [80] as well as using the receiver oscillator for detection of anomalous signals. Methods for modeling oscillators will be discussed in the following sections, and the discussed methods will be implemented in the GNSS simulations shown in Chapter 6.1.

4.2 Oscillator Stability In The Time Domain

Modern oscillators keep time by measuring the number of cycles of a source (crystal or atomic) and creating a “tick” every time a certain number of cycles is counted. In a perfect world,

the frequency of the source would never change, and the clock would keep time perfectly. Unfortunately, there is no such thing as a perfect clock, and the frequency of the source is dependent on numerous factors. The stability of any given oscillator can depend on factors such as temperature fluctuations, power fluctuations, vibration, aging, and inherent noise processes in the oscillator. Frequency drift, or the change in frequency over time, is a term that refers to the algebraic sum of all the frequency errors combined [81]. In order to model oscillators accurately for simulation, an understanding of the frequency errors and frequency stability must first be developed. To do so, the fundamentals of oscillator stability are explored. Several resources, such as [82] and [83] explore numerous aspects of frequency stability and have additional details that are not covered here.

As a frequency reference, the ideal oscillator output can be described using a generic sinusoid oscillating at nominal frequency v_0 in Hz and amplitude V_p as shown in Equation (4.1) [84].

$$V(t) = V_p sin(2\pi v_0 t) \tag{4.1}$$

Equation (4.1), however, assumes that there are no amplitude or phase deviations from the nominal values, which is not the case in real oscillators. A more realistic model accounts for both the amplitude fluctuations $\epsilon(t)$ and the phase fluctuations $\phi(t)$, shown in Equation (4.2) [84].

$$V(t) = [V_p + \epsilon(t)]sin(2\pi v_0 t + \phi(t)) \tag{4.2}$$

The errors shown in Equation (4.2) are randon in nature, but can be studied and quantified using measurements of frequency stability. One point of confusion which is often brought to the reader's attention is that, while most of the literature in the area refers to frequency stability, the values measured are actually measures of frequency instability [84]. Typically, these refer to the phase error $x(t)$ and the fractional frequency $y(t)$. The phase error, $x(t)$ can be related to the phase fluctuations $\phi(t)$ by Equation (4.3) [82].

$$\phi(t) = 2\pi v_0 x(t) \tag{4.3}$$

Equation (4.4) relates the frequency error of an oscillator to the phase error, where $\frac{d\phi}{dt}$ is the derivative of the phase error shown in Equation (4.3)

$$v(t) = v_0 + \frac{1}{2\pi}\frac{d\phi}{dt} \tag{4.4}$$

When discussing measures of oscillator instability, it is important to keep in mind that the measurements are relative to two oscillators, the oscillator under test and the reference oscillator driving the measurement system. The reference oscillator typically has significantly better performance than the oscillator under test, allowing for the assumption that the noise measured is only of the oscillator being tested. This assumption allows for the calculation of the fractional frequency, $y(t)$, which is commonly used to describe deviation from a nominal frequency v_0. The derivation of the fractional frequency is shown in Equation (4.5).

$$y(t) = \frac{\Delta f}{f} = \frac{v(t) - v_0}{v_0} = \frac{1}{2\pi v_0}\frac{d\phi}{dt} = \frac{dx}{dt} \tag{4.5}$$

Frequency stability of an oscillator can be represented in either the time domain or the frequency domain [82]. Presently, the time domain representation will be discussed, with details of the frequency domain analysis available in Appendix A.1. In most cases where random processes are concerned, a mean and a standard deviation are used to describe the behavior of the random samples. Due to the nature of the noise processes relevant to oscillators, however, these measures typically do not converge, negating their usefulness in describing the behavior of the oscillator.

Generally, time domain stability is represented using a $\sigma - \tau$ plot, with the X axis representing a measure of time and the Y axis representing the calculated statistical measure [82]. The most common method used to quantify the statistics of oscillator noise is the Allan deviation, which is discussed in detail later in this section. These plots allow for the visualization of various noise types that are present in oscillators, and can be used to derive statistical models for simulations. An example of a $\sigma - \tau$ plot is shown in Figure 4.2. Each portion of the plot has

a distinct slope which can be correlated to a specific type of noise that is present in the oscillator. Table 4.4 shows the relation of the power law slope to each type of noise, respectively. The values of the slopes represented by μ correspond with the $\sigma - \tau$ plot as shown in Figure 4.2, and the slopes represented by α correspond to the frequency domain representation of stability, discussed in Appendix A.1. The relationship between the power law slopes and the regions on the Allan Deviation curve is shown in Equation (4.6).

$$\sigma_y(\tau) \sim \tau^{\frac{\mu}{2}} \tag{4.6}$$

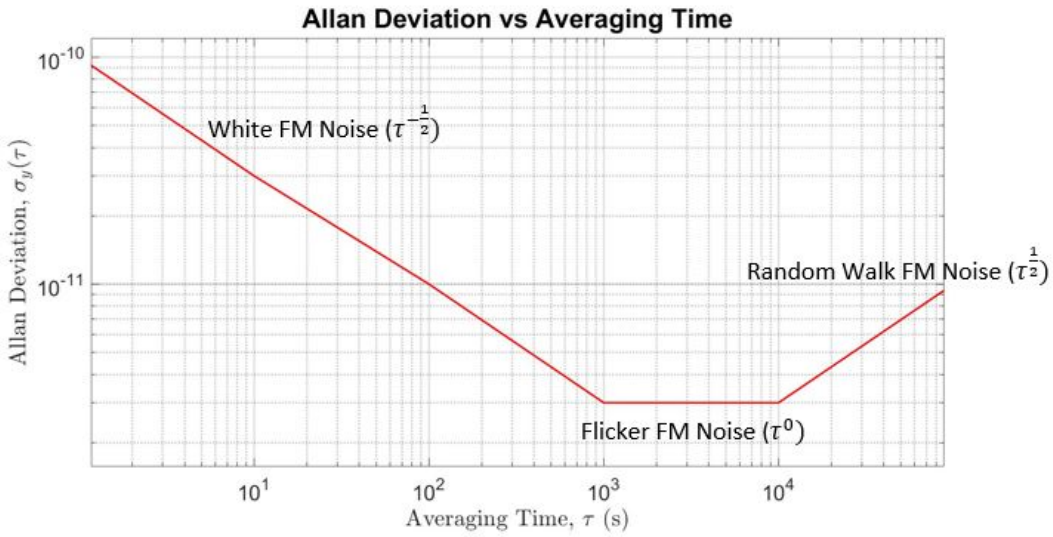

Figure 4.2: Example of $\sigma - \tau$ Plot With Noise Processes

Table 4.4: Power Law Slopes

Noise Type	α	μ
White Phase Modulation	2	-2
Flicker Phase Modulation	1	-2
White Frequency Modulation	0	-1
Flicker Frequency Modulation	-1	0
Random Walk Frequency Modulation	-2	1

As mentioned previously, the classical sample variance does not converge for some noise processes present in oscillators. For certain noises, the mean of the data varies depending on

the length of the data set, causing the variance to vary with the number of samples [85]. The classical sample variance is shown in Equation (4.7), where M is the number of samples, y_i is the i^{th} sample, and $\bar{y}$ is the sample mean of the set.

$$\sigma^2_{std} = \frac{1}{M-1}\sum_{i=1}^{M}(y_i - \bar{y})^2 \tag{4.7}$$

To allow for a more universal description of frequency stability, the Allan variance (AVAR) was introduced by David Allan. Also known as the two sample variance, this method uses a difference between subsequent samples with no dead time in between, allowing the variances to converge for the types of noise present in oscillators. As is the case with the standard variance and deviation, the Allan variance is the square of the Allan deviation (ADEV), which is commonly plotted to represent the short term stability of an oscillator. An example of the divergent nature of the standard deviation compared to the Allan Deviation can be seen in Figure 4.3, where the standard deviation increases as a function of sample size. Equation (4.8) shows the formal definition of the Allan Variance, and Equation (4.9) shows the method that can be easily implemented to calculate the AVAR.

$$\sigma^2_y(\tau) = \left\langle \frac{1}{2}[y(t+\tau) - y(t)] \right\rangle^{\frac{1}{2}} \tag{4.8}$$

$$\sigma^2_y(\tau) = \frac{1}{2(M-1)}\sum_{i=1}^{M}[y_{i+1} - y_i]^2 \tag{4.9}$$

The Allan Variance is calculated for a given averaging time, τ. The sampled data is said to be taken with an averaging of τ_0, and each successive averaging time is described using the multiplier m, giving averaging times of $m\tau_0$. To resample the data, the original sampled data can be averaged using windows of the desired value of $m\tau_0$. For example, if $\tau_0 = 1s$, an averaging time of $\tau = 2s$ can be calculated by taking adjacent pairs of measurements and averaging them, then using that average as a new sample [85].

By calculating the Allan variance over different averaging times, the different noise processes that are present in the oscillator can be discerned. As shown in Figure 4.2, the slopes

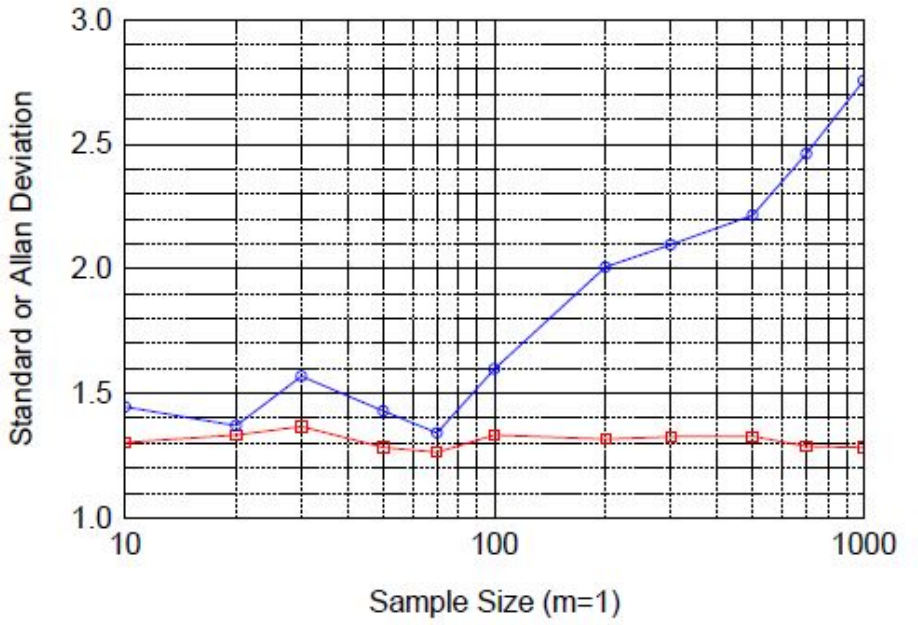

Figure 4.3: Comparison of Standard Deviation (blue) to Allan Deviation (red) [82]

of the Allan Deviation vary with the averaging time used to calculate the deviation. Figure 4.4 shows the frequency errors over different averaging times. As expected for this oscillator, at short averaging times ($\tau = 1s$), the frequency error is dominated by white noise. When the averaging time is increased ($\tau = 1000s$), the underlying random walk of the frequency becomes observable.

The standard Allan Variance (AVAR) is simple to implement using frequency data samples, however results in relatively low confidence for longer averaging times. This lower confidence is due to the number of samples effectively being reduced by the averaging that is done, leaving fewer samples to calculate the variance. This effect can be seen in Figure 4.4, where at $\tau = 1s$ there are 10×10^6 samples, but at $\tau = 1000s$ there are only 10000 samples. To overcome this limitation, the overlapping Allan variance is often used, which uses fully overlapping samples to increase the number of measurements used for each averaging time. Typically, the overlapping Allan variance is the standard for calculation of frequency stability values for oscillators. Figure 4.5 depicts the differences between the two methods of calculating the Allan variance.

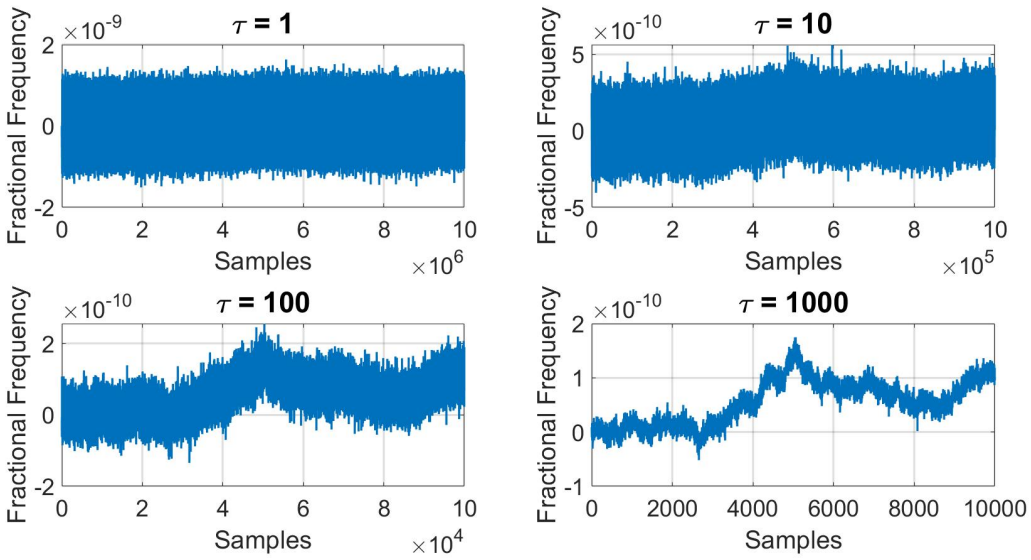

Figure 4.4: Frequency Errors Over Different Averaging Times

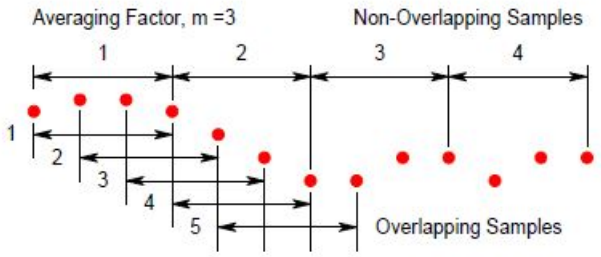

Figure 4.5: Comparison of Standard and Overlapping Allan Variance Samples [82]

The calculation of the overlapping Allan deviation (OADEV) can be done with either phase or frequency samples, as is the case with the standard Allan Variance. In the case of the OADEV, however, it is much simpler to implement using measurements of phase. Equation (4.10) shows the method of calculating the OADEV using frequency samples, which involves a somewhat computationally heavy double summation.

$$\sigma_y^2(\tau) = \frac{1}{2m^2(M-2m+1)} \sum_{j=1}^{M-2m+1} \left\{ \sum_{i=j}^{j+m-1} [y_{i+m} - y_i] \right\}^2 \tag{4.10}$$

Computing the OADEV using phase samples reduces the computational load if available, as only one summation is requred. For this reason, the phase is typically used to calculate the

OADEV, as shown in Equation (4.11).

$$\sigma_y^2(\tau) = \frac{1}{2(N-2m)\tau^2} \sum_{i=1}^{N-2m} [x_{i+2m} - 2x_{i+m} + x_i]^2 \tag{4.11}$$

As mentioned previously in the chapter, x_i is the phase deviation of the oscillator signal, and can be calculated by integrating the fractional frequency values over time. Integrating the frequency values results in $N = M + 1$ samples. To calculate the OADEV over averaging times, the expression $\tau = m\tau_0$ is used, where τ_0 is the base sample rate of the data. In many cases when phase data is used, $\tau_0 = 1s$ as this is the fastest the $1PPS$ phase difference can be sampled. Allan deviation functions were verified by using a 1000-point data set containing frequency data samples [86]. This data set, along with the 9 sample NBS data set from [87] provide the correct Allan deviation values as well as phase and frequency values and are invaluable tools when implementing Allan deviation calculations.

Figure 4.6 shows a comparison between the standard and overlapping Allan deviations It can be seen in the figure that, while initially aligned, the values calculated for the Allan deviation become increasingly noisy as averaging time increase. As shown in Figure 4.5, the standard Allan deviation uses significantly less samples than the overlapping implementation, leading to a decrease in confidence at higher averaging times. By overlapping samples, the OADEV has more samples available to work with at larger averaging times, increasing the confidence in the calculated ADEV. Both calculations of Allan deviation are shown overlaid on the power law noise model that was used to calculate the spectral density coefficients h_0 and h_{-2}.

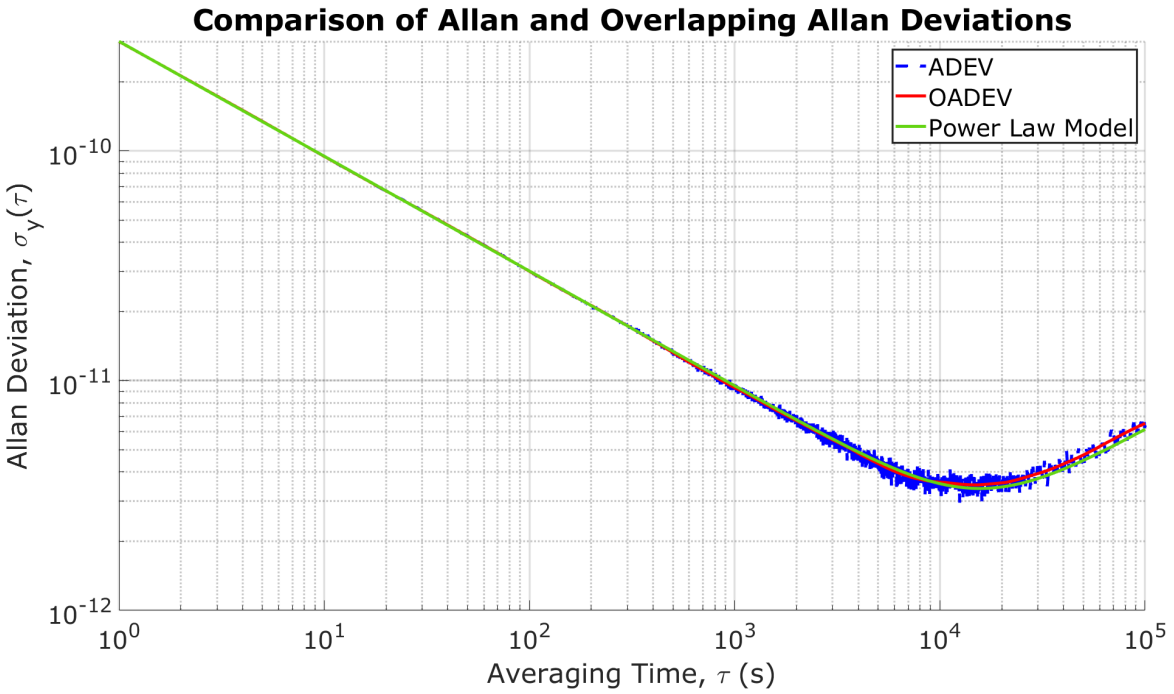

Figure 4.6: Comparison of Standard and Overlapping Allan Deviation Using Simulated Clock Data

Several free software packages, such as Stable32 and TimeLab are available that can read in .csv files which contain frequency or phase data and calculate Allan variances and other statistics. Stable32 has various other built in functions, including the simulation of power law noises using models similar to those discussed here [88],[89].

Performing an Allan variance calculation on phase or frequency data provides the ability to analyze noise processes present and can be related to several simulation models which are discussed in Section 4.3. One question still remains, however: how are measurements of phase or frequency data obtained? The answer to this question can be found in [90], which describes several methods, including the beat frequency method and the time difference method. Both of these methods can be performed using modern instruments such as the Keysight 53230A frequency/time interval counter, which was used for the frequency analysis conducted in this thesis [91].

The first method which can be used to analyze frequency stability is the beat frequency method. The beat frequency is the difference between the reference oscillator and the oscillator under test. The output signals of each clock are mixed using a double balanced mixer, then low

pass filtered to remove the higher frequency image [90]. If necessary, the filtered signal is amplified then passed to a frequency counter, which records the beat frequency. To calculate the fractional frequency error $y(t)$, the measured beat frequency f_b is divided by the nominal frequency v_0 of the oscillator. The typical hardware setup for measurement of the beat frequency is shown in Figure 4.7, which is adapted from [81]. As noted in the figure, in addition to the two oscillators and frequency counter, several RF components are required for this measurement as well.

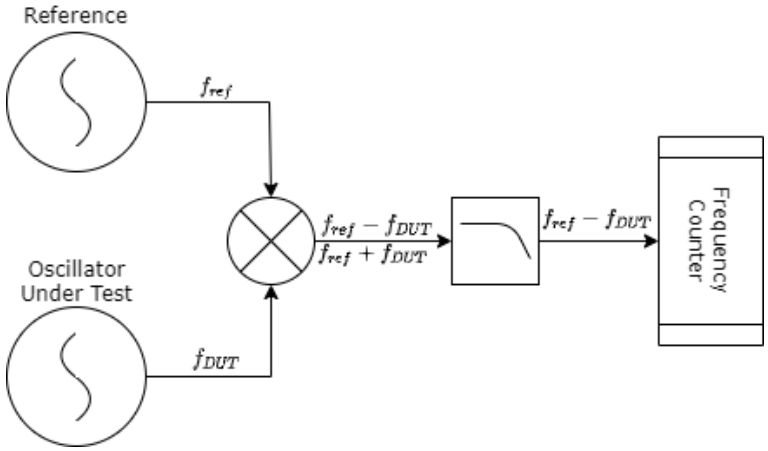

Figure 4.7: Beat Frequency Measurement Method

The beat frequency measurement method provides excellent performance and can be used for virtually any oscillator that has a frequency output available to the user. In some situations, however, a device may only have a time reference output or the additional hardware required for the above method may not be readily available. The second method, measuring the time interval between two signals, can be implemented in these cases.

The time difference method is not as precise as the beat frequency method, however with modern time interval counters the precision is comparable. This method is frequently used for devices that do not provide a frequency reference, such as a GPS receiver with only a $1PPS$ output. A comparison between the $1PPS$ output from a reference oscillator and an oscillator under test provides a direct measurement of the phase error $x(t)$ of the test oscillator, which can be used to calculate the Allan variance as discussed previously [90]. A hardware diagram of the time difference method is shown in Figure 4.8, adapted from [81].

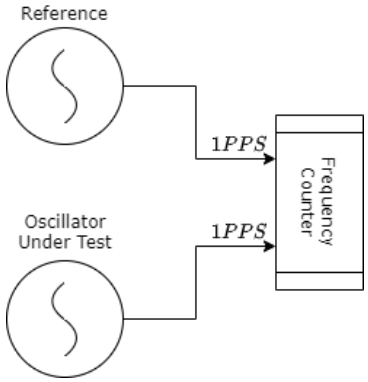

Figure 4.8: Time Difference Method

For this work, the time difference method was used for its simplicity as well as the unavailability of RF mixers and filters used for the beat frequency implementation. An example of phase data recorded from this method is shown in Figure 4.9, where the oscillator under test was an Ettus Research Octoclock-G with the GPS disconnected, and the reference was a Stanford Reasarch Systems FS725 Rb standard. When tested, the Octoclock had a frequency offset which introduced a linear drift in phase, which was removed to allow the phase deviations to be observed clearly.

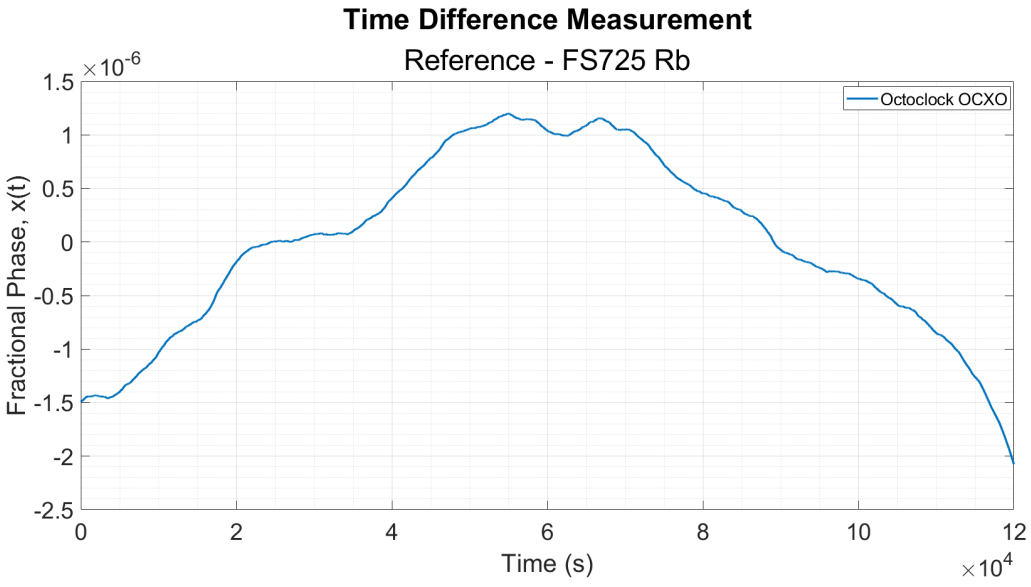

Figure 4.9: Octoclock Phase vs FS725

This phase data shown in the above figure was processed using the overlapping Allan deviation calculation to generate the Allan deviation plot shown in Figure 4.10. One aspect of the calculation of Allan variances that can be particularly time consuming is the data collection

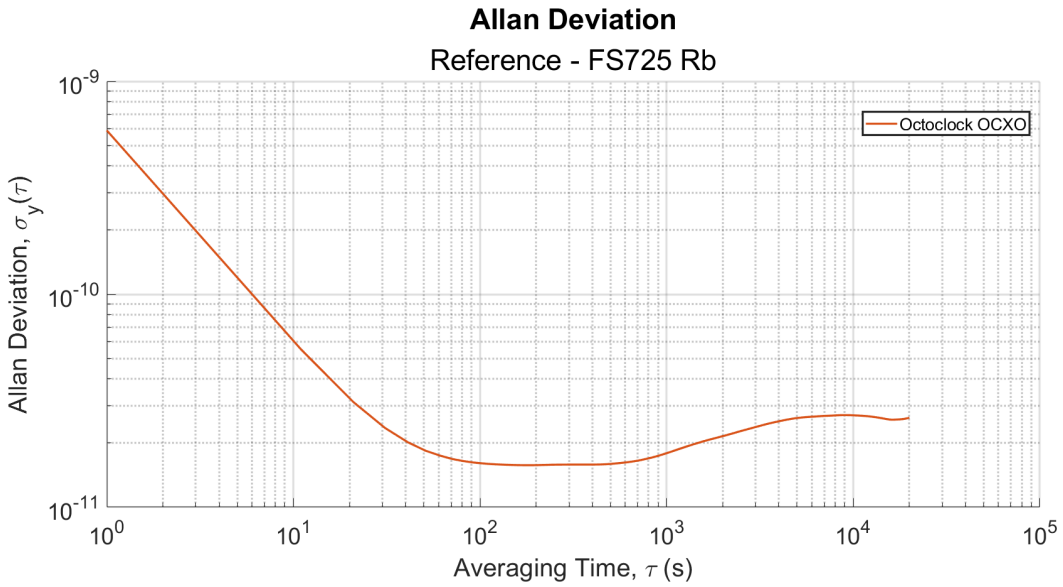

Figure 4.10: Octoclock-G (Jackson Labs LCXO) Overlapping Allan Deviation

for high stability oscillators. The Allan deviation shown in Figure 4.10 clearly shows the white and flicker frequency errors, however the random walk has not yet become apparent in the curve shown, which covers $20000s$ worth of averaging times. A general rule is to have $5-10\times$ the number of samples as the highest averaging time of interest, and in this case the phase data shown, data was logged for a total of $120000s$, or just under a day and a half. Test time becomes a serious bottleneck when attempting to generate a complete statistical analysis of high precision oscillators, which can limit the modeling of oscillators described in the following section.

4.3 Clock Modeling

In addition to providing insight into the performance level of the oscillator in use, the values calculated using the Allan variance can be used to model the process noise of the oscillator. This is particularly useful in applications where prediction of time error is of interest. Two methods of clock simulation are commonly used: a two state discrete time model, and a continuous time model. Each simulation method will be discussed in the following sections, and the benefits and drawbacks of each method will be discussed.

For both models, the noise variances are characterized by the Allan Deviation of the desired oscillator. Returning to the plot of the Allan variance shown in Figure 4.2, the values of the Allan variance can be related to the power law coefficients as shown by Equation (4.12), where τ is the averaging time and f_h is the high frequency cutoff of the measurement system [92].

$$\sigma_y^2(\tau) = h_2 \frac{3f_h}{(2\pi)^2\tau^2} + h_1 \frac{1.038 + 3\ln(2\pi f_h \tau)}{2\pi^2\tau^2} + h_0 \frac{1}{2\tau} + h_{-1} 2\ln(2) + h_{-2} \frac{2\pi^2}{3}\tau \qquad (4.12)$$

Typically, phase modulation terms are neglected when using this method, as the value for f_h is frequently unknown. The relationships between the h_α values and the noise processes are shown in Table 4.5. Typically, the values representing the flicker and white phase modulation are ignored, as they are not typically distinguishable using the ADEV or OADEV, as both have the same slope. In addition to neglecting these parameters, the flicker frequency modulation parameter is typically neglected as the flicker noise cannot be modeled by a finite order process.

Table 4.5: Power Law Coefficients

Noise Type	**Coefficient**
White Phase Modulation (WPM)	h_2
Flicker Phase Modulation (FPM)	h_1
White Frequency Modulation (WFM)	h_0
Flicker Frequency Modulation (FFM)	h_{-1}
Random Walk Frequency Modulation (RWFM)	h_{-2}

4.3.1 Two State Clock Model

Most applications rely on a straightforward two state clock model to represent the errors in the phase and frequency for both simulation and estimation. Shown in Equation (4.13), the primary driving forces for the oscillator are the noise processes represented by η. The noise processes

have a covariance matrix Q which is comprised of the variance and covariance of the phase and frequency. The elements of the covariance matrix can be seen in Equation (4.15) to Equation (4.17) [93].

$$\underline{\mathrm{x}}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \underline{\mathrm{x}}_k + \eta_k, \quad \eta \sim N(0, Q) \tag{4.13}$$

where

$$Q = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \tag{4.14}$$

and

$$q_{11} = E[x^2(\Delta t)] = S_f \Delta t + \frac{S_g \Delta t^3}{3} \tag{4.15}$$

$$q_{12} = E[x(\Delta t)y(\Delta t)] = S_g \Delta t \tag{4.16}$$

$$q_{22} = E[y^2(\Delta t)] = \frac{S_g \Delta t^2}{2} \tag{4.17}$$

The q_{11} term can also be represented by the power law coefficients shown in Equation (4.12) previously. In terms of the power law coefficients, Equation (4.15) can be written as Equation (4.18):

$$q_{11} = \frac{h_0}{2}\Delta t + 2h_{-1}\Delta t^2 + \frac{2\pi^2}{3}h_{-2}\Delta t^3 \tag{4.18}$$

Equation (4.18) can be compared to Equation (4.15) and the like terms can be matched. The results of the comparison are shown in Equations (4.19) and (4.20).

$$S_f = \frac{h_0}{2} \tag{4.19}$$

$$S_g = 2\pi^2 h_{-2} \tag{4.20}$$

The h_{-1} term in Equation (4.18) represents the flicker noise process, and is ignored for the analysis done here. By neglecting the flicker noise term the model over approximates the

flicker floor of the oscillator [93]. This can be mitigated by slightly elevating the entire curve, which provides a compromise between the flicker floor and the noise at the beginning and end of the curve, which are increased from the appropriate value as well.

Each clock has a unique set of values for the power law coefficients depending on the stability of the oscillator in question. Typically, lower stability oscillators have larger values, corresponding to a larger noise variance. Table 4.6 shows some typical values for common types of oscillators [93]. When combining a clock model with GPS, it is common to specify the clock error in units of meters instead of seconds. In this case, the parameters given in Table 4.6 must be multiplied by the speed of light squared.

Table 4.6: Power Law Coefficients

Oscillator Type	h_0	h_{-1}	h_{-2}
TCXO	2×10^{-19}	7×10^{-21}	2×10^{-20}
OCXO	2×10^{-25}	7×10^{-25}	6×10^{-25}
Rubidium	2×10^{-22}	4.5×10^{-26}	1×10^{-30}
Cesium	2×10^{-22}	5×10^{-27}	1.5×10^{-33}

Power law coefficients can also be calculated for oscillators by taking the Allan deviation of phase or frequency data and comparing the plot to the power law given by Equation (4.12). This is done by identifying the slopes that relate to the h_0 and h_{-2} noise types and pulling the values of both τ and $\sigma_y^2(\tau)$ from the plot. After these values are know, the power law coefficients can be solved for and used in the two state model given in Equations (4.19) and (4.20). A detailed example of this process can be found in Appendix A.

As mentioned previously, the two state model is a fairly simple and lightweight model that generates the white frequency and random walk frequency modulation (WFM and RWFM) noises for the oscillator at hand. This model is frequently used for the estimation of clock states and is commonly used as the noise variance model for clock bias and drift with GPS navigation filters. The model can also be used for simulation of clock errors, however as stated only white FM and random walk FM noises are simulated. In addition to only simulating these noises, other noise types such as flicker FM or any phase noises are not simulated, limiting the usefulness of the model for simulation purposes. For some clocks, the assumptions made in

this model are valid, and can accurately represent the behavior, but for others, the model may fail to reflect the true behavior of the clock.

4.3.2 Continuous Time Clock Model

While the two state clock model can be used for simulation of oscillators, the continuous time model is frequently used for simplicity. Clock errors can be described by the continuous time algebraic expression shown in Equation (4.21), which is analogous to the constant acceleration model commonly used in particle kinematics [94].

$$x(t) = x_0 + y_0 t + 0.5at^2 + \int_0^t y_e(t)dt + \eta(t) \tag{4.21}$$

When compared with Equation (4.13), it is observed that the continuous time model and the x_1 expression represent the same errors if the aging term is neglected. The continuous time model includes both aging (a) and additional environmental frequency error terms (y_e), however these could be added into the two state model if desired as shown in [95]. These errors can be added into the continuous time model with a simple integral as shown in Equation (4.21), making this model desirable for the ease of implementation.

One difference between the two error models is the additive noise. The two state model is limited to the two noise terms defined in the Q matrix, the white and random walk frequency noise. Many oscillators, however, have other noise types present. The two state model does an acceptable job estimating these errors when used in a Kalman filter, but is less adequate when generating clock noises for simulation. The additive noise shown in Equation (4.21) can be simulated independently, allowing for the simulation of all five noise types shown in Figure 4.11. Even coefficient power law noises ($\alpha = -2, 0, 2$) are straightforward to simulate, as one can simply integrate or differentiate the noises to decrease or increase the power, respectively. To generate the even power law noises, either Gaussian white noise with the appropriate variance can be simulated then integrated or Brownian motion can be generated and differentiated [96].

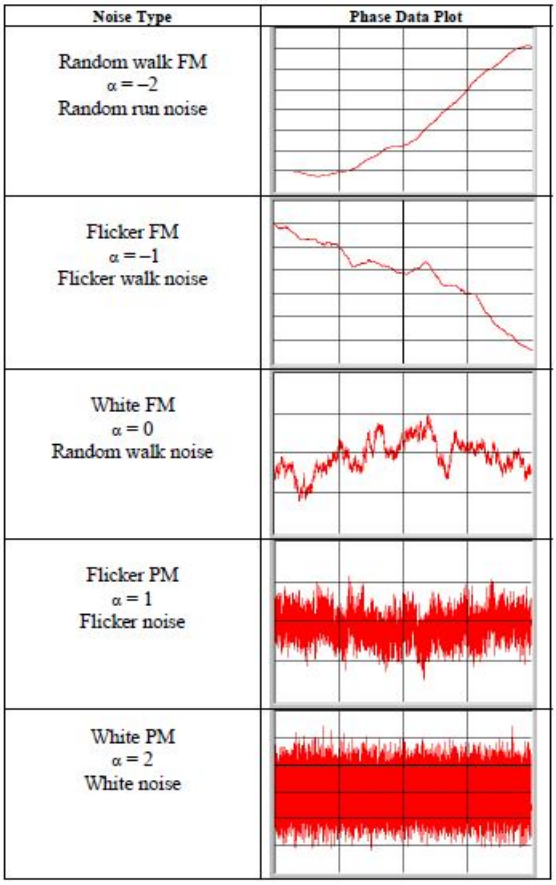

Figure 4.11: Power Law Noises

Flicker noise, however, can not be represented by a simple equation or finite expression. To generate flicker noise, Gaussian white noise is passed through a cascade of first order filters as described in [97] and [98]. By filtering the white noise, the desired flicker noise is generated. Once generated, this noise can be added to the desired clock state to simulate colored noise on the phase and frequency outputs.

To use the generated noise processes, the frequency noise terms must be converted to phase noise terms. This is done by simply integrating the white, flicker, and random walk frequency noises. Once integrated, the noises can be added to the white and flicker phase noise to get the total stochastic phase error. If it is desired to simulate frequency error, the phase noise processes can be differentiated. To verify that each noise type appears correctly, the Allan variance of the noise can be computed. Figure 4.12 shows an example of a process where each of the three noise types of interest were simulated. As mentioned previously, the flicker and white phase modulation noises are neglected, as these are typically unobservable on clock

outputs due to their high frequency nature. The Allan variance values calculated in Figure 4.12 were calculated using the standard Allan deviation method. As expected, the uncertainty in the Allan deviation increases as the averaging time increases due to the decreased number of samples available for the calculation. The purple curve on the plot is the sum of the three independent noise types, and demonstrates the behavior of a typical Allan deviation plot with a U-shape. As expected, the white FM noise has a slope of $\tau^{-\frac{1}{2}}$, the flicker noise has a slope of τ^{0}, and the random walk noise has a slope of $\tau^{\frac{1}{2}}$, indicating that the clock model is generating the desired noise types and is valid for simulation of timing errors in GNSS receivers.

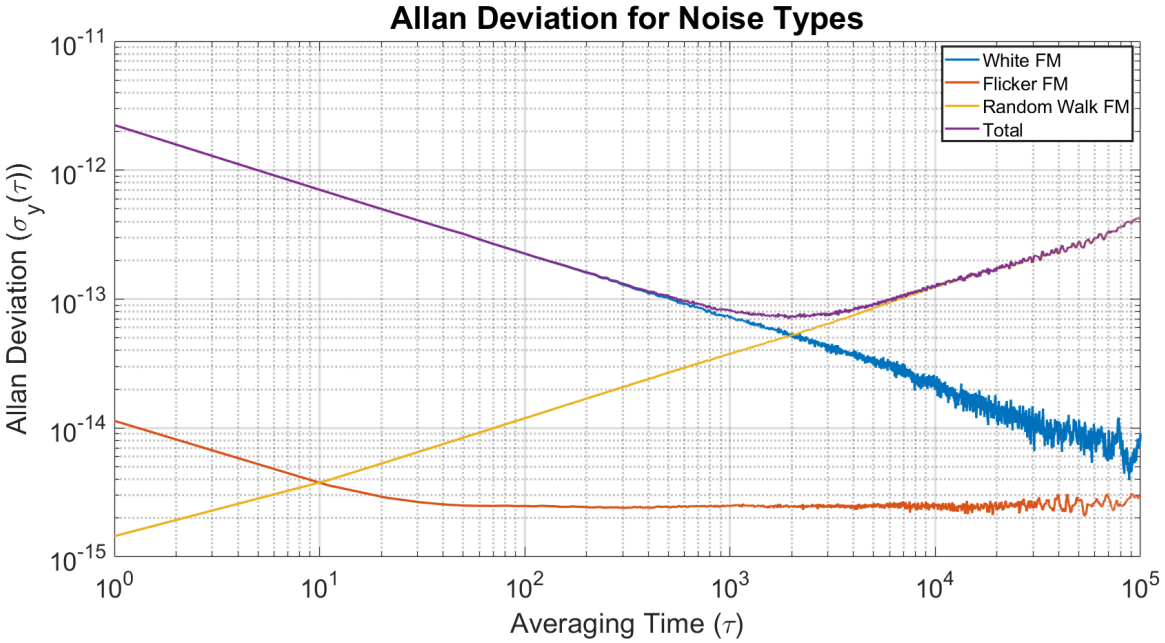

Figure 4.12: Allan Deviation of Simulated Noises

To simulate the desired clock errors, the appropriate variances must be calculated for use in the model as was done with the two-state model above. For the continuous time model, each noise process is simulated directly using random vectors multiplied by the appropriate value of the Allan variance. An example of the clock bias of several common oscillators is shown in Figure 4.13. In this figure, the log of the absolute value of the bias was taken to show the large difference between the timing error of a TCXO compared to an Rb standard. To calculate the appropriate value of the Allan variance used for each oscillator, the slope of each noise process is extended to intersect with an averaging time of $\tau = 1s$. The Allan variance value $\sigma_y^2(\tau = 1s)$

is used as the variance for the noise processes described above. A detailed exploration of this method can be found in Appendix A.

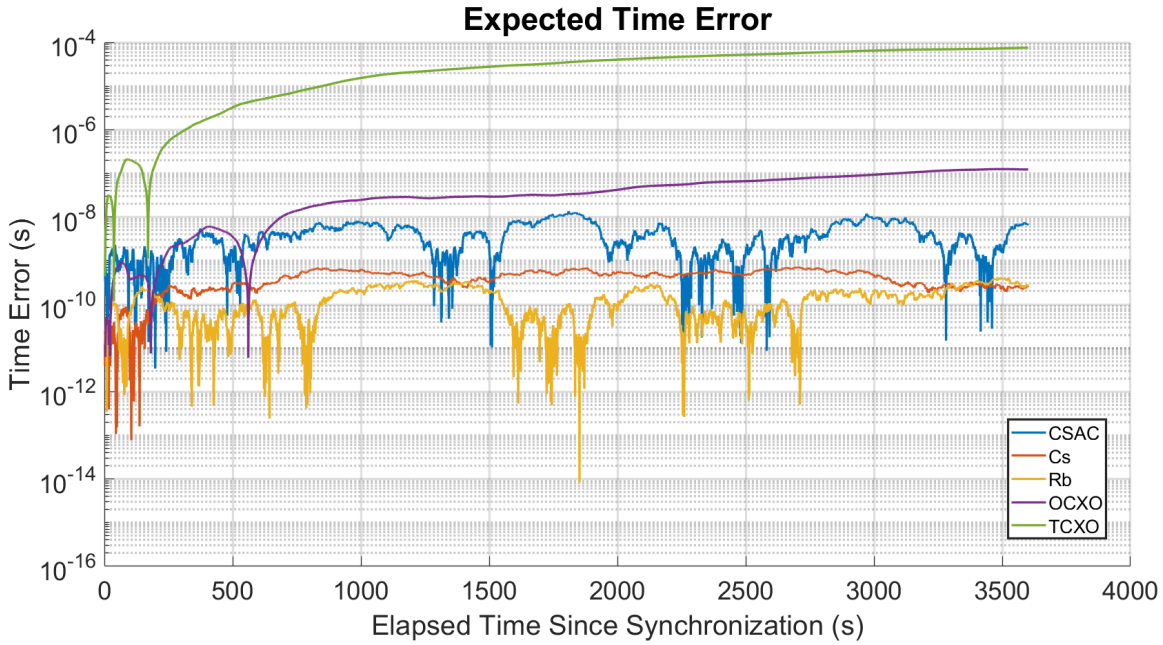

Figure 4.13: Expected Time Error After 1 Hour For Typical Oscillators

Deterministic environmental effects such as thermal changes or vibrations can be simulated and added to the clock states on top of the noise processes, introducing additional error. These environmental effects directly impact the frequency error, which increases the overall time error reported by the clock [94]. The ability to include these errors, along with all desired noise types and aging, make the model an effective tool for simulating receiver clock errors in GNSS simulations. For this work, aging is neglected due to the short periods of evaluation, which are sufficiently short to the point where aging affects would not be observed. Thermal and vibrational errors are neglected as well, as the exact characteristics of these errors vary widely from application to application. These errors can be added if a specific application is desired to be simulated, but this is neglected to focus on the detection methods proposed.

4.4 Conclusion

Clock modeling provides a preliminary method for testing of algorithms in simulation that are representative of real hardware environments. Oscillators are inherently noise driven processes, and modeling and simulation allows for running Monte Carlo simulators to observe the random

characteristics which may not be possible in hardware testing due to limited time or hardware. By accurately representing transmitter and receiver clocks in GNSS networks, an understanding of the effects of clock bias, drift, and phase noise can be developed without the need for costly user equipment and many hours of testing. Methods of determining the appropriate values for model parameters were discussed, which can be derived from hardware testing if available or referenced from a manufacturer's data specification sheet. In addition to modeling capabilities, clock estimation provides smoothing to least squares clock state estimates as well as provides models for Kalman filter implementations. Having an accurate clock model also allows for clock states to be propagated across outages, providing an estimate of the uncertainty of what the clock states should be when signals return. This aspect is leveraged in the spoofing detection algorithm explored in Chapter 5 to bridge outages in signal coverage that attackers may attempt to take advantage of.

Chapter 5

Clock Drift Estimation and Fault Detection

In addition to estimating the receiver clock bias and drift using a Kalman filter, the receiver clock drift can be estimated by taking the time difference of incoming measurements. Both pseudorange and carrier phase measurements are influenced by the receiver clock bias, and when differenced over time, provide an estimate of the instantaneous receiver clock drift. This technique, described in [20], is straightforward for a static receiver with a known position. When a receiver is static, the location can be surveyed leaving the receiver clock drift as the only unknown parameter. Expanding to a dynamic platform the requires the use of an external sensor to get an estimate of position prior to computing the expected clock drift, adding complexity and increasing the possible error in the estimate depending on the error in the computed position. This technique, when paired with a stable receiver clock with little drift, can be used to detect errors due to multipath or spoofed GPS signals, improving the integrity of receivers used in challenging environments.

Changes in the clock bias and drift can be observed when the transmitter moves or adjusts signal parameters, as demonstrated in [20]. Transmitters using low quality TCXO and OCXO LOs can be readily identified by the increase in drift over what is present due to the high quality receiver clock and satellite clocks. This paper performed an analysis on the detection of a spoofer, and the results demonstrated changes in the estimated clock drift when the transmitter moved, changed the transmit time of the signal, or was driven by an unstable oscillator. This analysis was done at a static site, allowing the receiver position to be known. The work done in this paper formed the basis for the proposed integrity monitoring methods described in this thesis.

When high stability oscillators are used as a receiver reference, the receiver drift can be predicted and estimated bounds from the EKF covariance matrix can be implemented. If estimates of clock drift exceed the expected uncertainty, the corresponding measurements can be marked as faulty and removed from the navigation processor input. Additional redundancy is provided by calculating clock drift using multiple constellations and frequencies. By using multiple frequencies, the chances of a repeater/simulator overtaking all available signals decreases significantly. Low end meaconers created using devices such as a HackRF or BladeRF have transmit/receive channel bandwidths that do not encompass multiple GNSS frequencies, limiting the range of signals which they effect [99],[100]. In more sophisticated setups, all GNSS signals may be corrupted, preventing the mitigation of the errors if an initial change in bias/drift is not flagged.

5.1 Time-Differenced Pseudorange

As discussed in Chapter 2, pseudorange measurements are derived from the true range between the receiver and satellite along with atmospheric errors, clock errors, and random noise. By taking the time difference of two consecutive pseudorange measurements from a single satellite as shown in Equation (5.1) and dividing by the time step Δt between the two measurements, the clock drift of the receiver can be estimated.

$$\tilde{\rho}^i_{t_2} - \tilde{\rho}^i_{t_1} = r^i_{t_2} - r^i_{t_1} + c(b_{t_2} - b_{t_1}) + (I_{t_2} - I_{t_1}) + (T_{t_2} - T_{t_1}) + \eta_{t_2} - \eta_{t_1} \tag{5.1}$$

Over small time steps, it can be assumed that ionospheric and tropospheric errors are constant and cancel when differenced. Shown in Equations (5.2) and (5.3), the clock drift becomes a function of the speed of light, the change in measured pseudorange, $\Delta\rho^i$, and the change in estimated range from the satellite to the receiver, Δr^i.

$$\frac{\Delta\tilde{\rho}^i}{\Delta t} = \frac{\Delta r^i + c\Delta b}{\Delta t} = \frac{\Delta r^i}{\Delta t} + c\dot{b} + \eta \tag{5.2}$$

$$c\dot{b} = \frac{\Delta\tilde{\rho}^i}{\Delta t} - \frac{\Delta r^i}{\Delta t} + \eta \tag{5.3}$$

The basis for this method is described in [20] for a static receiver in a surveyed location, which allows the estimated range to become a function of known user position and satellite position calculated from decoded ephemeris parameters. When dynamic or in an unknown location, additional error is introduced to the estimate by way of the user position estimate used to calculate the estimated range between the satellite and antenna. Estimation of clock drift is done prior to estimation of the current user position to prevent errors from entering the position estimate, driving the requirement for a current estimate of the user position when implemented on a dynamic platform. This can be achieved using two methods: only using the time update of the GNSS EKF, or using a coupled GNSS/INS navigation filter to update the position during periods between GNSS measurements. The performance of using the GNSS time update is severely limited as any dynamics that are unmodeled by the state transition matrix will create a difference in the actual and estimated position, so this method is not explored further.

For a static known location, the unknowns in Equation (5.1) can be reduced to only the satellite positions as compared to the satellite and user positions in a unknown location or dynamic scenario. The satellite positions can be calculated using broadcast ephemerides from each satellite, allowing for the estimation of the receiver clock drift using Equation (5.3). One shortcoming of the $\Delta\rho$ method described is the noise on the received pseudorange measurements, which is roughly 1% of a chip or better [101]. For GPS L1, with a chip width of $293.1m$, this corresponds to a noise magnitude of about $3m$. This noise makes the $\Delta\rho$ estimate extremely noisy with a mean about the estimated clock drift value, effectively hiding any errors from repeaters or other interference sources. Figure 5.1 shows the estimation of receiver clock drift using a static simulation representing the roof of Woltosz Engineering Research Laboratory (WERL). GPS L1 signal data was simulated using a Spirent GSS9000 GNSS simulator and was recorded using an Ettus Research USRP N210 Software Defined Radio (SDR) driven by a Stanford Research Systems FS725 Rubidium standard. Measurements are generated from the baseband samples using GNSS-SDR, an open source C++ software receiver [102]. The stable frequency generated by the FS725 can be observed by the constant clock drift, with an offset due to either the Spirent OCXO or the FS725. The noise on the time differenced pseudoranges becomes apparent, with errors as large as $\pm 10m/s$.

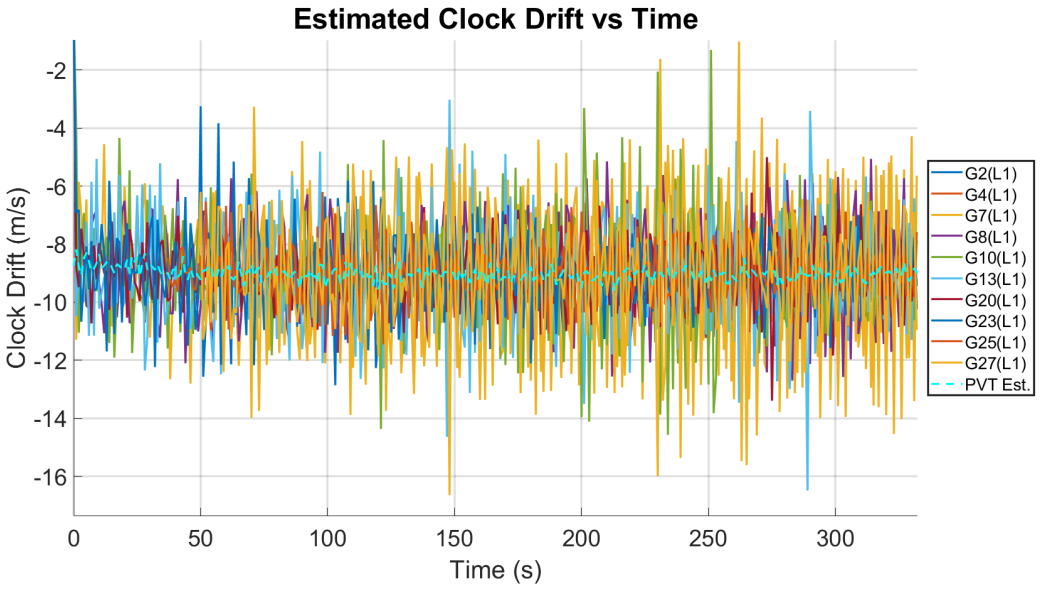

Figure 5.1: Static Clock Drift Estimation using $\Delta\rho$

Clock drift estimation can provide a useful metric for detecting the presence of additional or erroneous signals based on the clock effects of the additional source, however this usefulness is degraded when the estimates contain significant noise. When a meaconer is used to repeat GNSS signals, additional delays are introduced as was discussed in Chapter 2. Additionally, clock drift errors will appear as the meaconer LO drift will be added to the frequency error of the received signal. When a simulator is used to generate GNSS signals, the clock bias and drift will be derived from the simulator LO instead of strictly the receiver clock, and will appear in the estimates of these values. All of the additional clock errors appear in the receiver estimate of the clock states, however may be masked by the noise on the drift estimates. For that reason, carrier phase measurements were explored as a finer resolution method of estimating receiver clock drift.

5.2 Time-Differenced Carrier Phase

Calculating receiver clock drift using carrier phase measurements follows the same process as is shown in the previous section. The method detailed in [20] uses received carrier phase measurements to calculate the clock drift for a static receiver, and this method provides the basis for this work. Carrier phase measurements provide a range estimate to each satellite,

which are corrupted by ionospheric and tropospheric errors as well as the receiver clock bias and random noise. Additionally, the carrier phase measurements include an integer ambiguity, which was discussed in Section 2.4.3. The difference equation for carrier phase measurements is shown in Equation (5.4).

$$\phi^i_{t_2} - \phi^i_{t_1} = r^i_{t_2} - r^i_{t_1} + c(b_{t_2} - b_{t_1}) + (I_{t_2} - I_{t_1}) + (T_{t_2} - T_{t_1}) + (N_2 - N_1) + \eta_{t_2} - \eta_{t_1} \quad (5.4)$$

As shown in the previous section, time differencing over short time steps allows Equation (5.4) to be simplified. When differenced over short time intervals, it can be assumed that the atmospheric errors are constant and the integer ambiguity does not change, allowing these values to cancel. By dividing the difference equation by the time step, an estimate of clock drift can be obtained, as shown in Equations (5.5) and (5.6).

$$\frac{\Delta\phi^i}{\Delta t} = \frac{\Delta r^i + c\Delta b}{\Delta t} = \frac{\Delta r^i}{\Delta t} + c\dot{b} \quad (5.5)$$

$$c\dot{b} = \frac{\Delta\phi^i}{\Delta t} - \frac{\Delta r^i}{\Delta t} \quad (5.6)$$

Carrier phase measurements are far more accurate than code phase measurements with the noise magnitude being roughly 1% of the wavelength. For pseudorange measurements, this was fairly large due to large the chip width of the L1 signal. For the GPS L1 carrier wave with a wavelength of $19.03cm$, this noise only a few millimeters. As a result of more accurate measurements, the drift estimates more accurately reflect the true drift value. Figure 5.2 shows the estimated receiver drift from the same static data shown in Figure 5.1. Compared to the noise on the differenced range estimate of drift, the noise on the carrier measurements is roughly $\pm 0.1m/s$. The finer resolution of the carrier phase measurements lends to a better candidate for estimation of receiver clock drift.

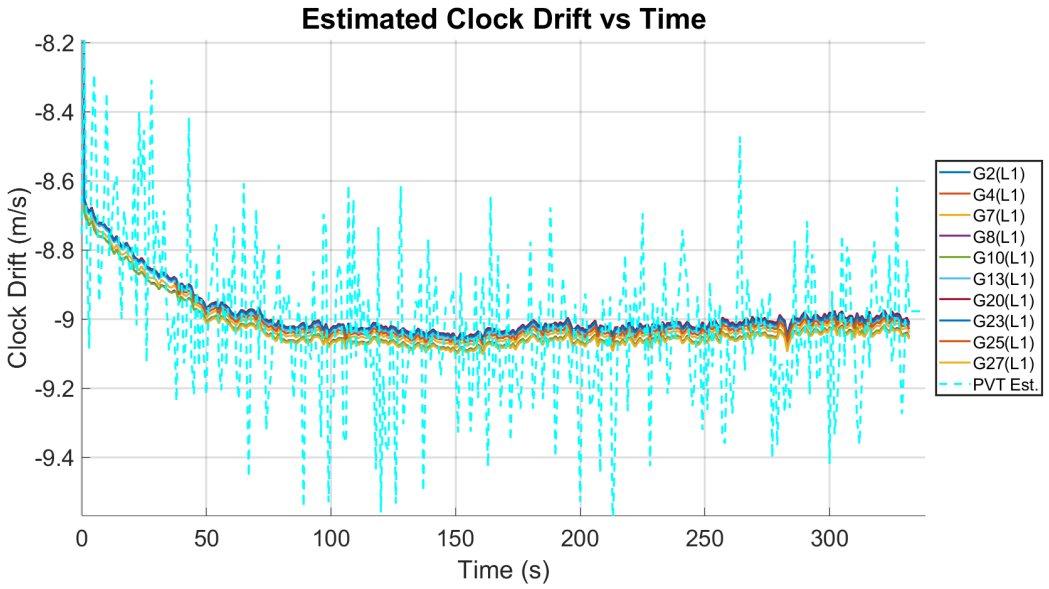

Figure 5.2: Static Clock Drift Estimation Using $\Delta\phi$

5.3 Detection Algorithm Construction

Two methods were used to test the detection algorithm described above: GNSS EKF time update and coupled GNSS/INS. Both methods rely on by-channel clock drift estimation as well as residual and residual sequence monitoring to improve the probability of detection. Once measurements have been checked by the drift estimation process, fault detection and exclusion (FDE) methods are implemented to check any that may have been missed or have slowly growing faults that are within the detection threshold for the drift estimation process. Each method is described in the following sections, then software and hardware simulations are explored.

Coupled GNSS/INS relies on mechanized measurements from the IMU to provide an estimate of the user location at the current time period. The IMU estimate of position is required because the movement of the dynamic platform after the last measurement update is not captured otherwise. This method increases computational complexity and system cost, but provides a better estimate of the current position. The fast update rate of IMUs provide measurements of the user dynamics between GNSS updates and INS errors can be estimated using GNSS updates when signals are available and valid.

The IMU mechanization process is detailed in Chapter 3. Once the measurements have been mechanized, the estimated user position can be used to calculate the range between the

satellite and user as shown in Equation (5.7) using the mechanized positions estimate from the INS.

$$\begin{aligned}\Delta r_k^i = &\sqrt{(x^i - x_{u_{INS}}^-)^2 + (y^i - y_{u_{INS}}^-)^2 + (z^i - z_{u_{INS}}^-)^2}_k \\ &-\sqrt{(x^i - x_u^+)^2 + (y^i - y_u^+)^2 + (z^i - z_u^+)^2}_{k-1}\end{aligned} \tag{5.7}$$

In Equation (5.7), the previous estimate of position calculated from the corrected GPS/INS states is used for the $k-1$ epoch.

Clock drift values estimated from the differenced carrier phase measurements are compared to the fault detection threshold shown later in this chapter to determine which measurements to process at the current epoch. If a channel's drift value exceeds the expected drift bounds, the measurements for that channel are removed from the measurement matrices. Individual faults can be handled using this method, however if over half of the measurements on a given frequency are marked as faulty, the entire frequency is removed from the filter update at that epoch. For example, if seven L1 measurements are used to estimate the receiver clock drift and five out of the seven are flagged as faulty, the remaining two unflagged measurements will be removed as well to prevent undetected errors from entering the navigation solution.

After a gap in GNSS availability, measurements are checked prior to updating the integrated navigation solution to ensure that no faults enter the receiver upon reacquisition of the signals. During the outage period, the INS output is mechanized and GNSS states are dead reckoned. When measurements return, the first batch is skipped to allow for a time difference to be calculated without potential errors influencing the navigation filter. The clock drift is estimated using the skipped measurement and the subsequently received measurement and compared to the drift before the outage as well as the propagated error bounds. Two measurements after the outage are used to estimate clock drift due to the integer ambiguity in the carrier phase measurements. When differenced over measurements without cycle slips, the integer ambiguity drops out of the equation. Over an outage, however, the integer ambiguity is not tracked and could change between the start of the outage and return of the signals. If the difference between the measurement prior to the outage and the first measurement after the

outage was calculated, the additional error from a change in integer ambiguity would be represented as additional clock drift at the receiver and the measurements could be rejected even if there is no interference. To avoid this, the method described above is implemented.

Once measurements have been validated, the desired coupling can be implemented in the filter measurement update. Figure 5.3 shows a block diagram of the generic system implementation for both loosely and tightly coupled filters. The system design is similar to a standard GPS/INS filter with the addition of the clock drift detection algorithm placed between the filter time update and measurement update. By placing the detection block before the measurement update, faulty measurements can be detected and removed from the filter to maintain solution accuracy and protect against the receiver being captured by transmitters.

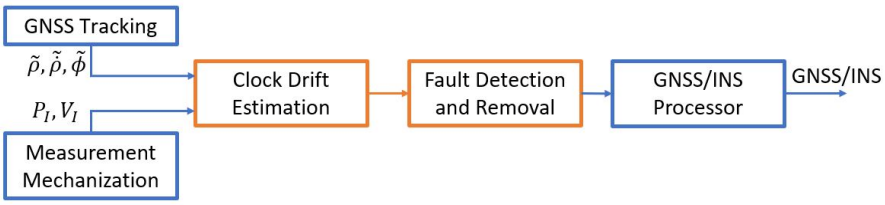

Figure 5.3: Complete Algorithm Block Diagram

5.3.1 Loosely Coupled Implementation

The loosely coupled GNSS/INS implementation adds several key components to the basic loosely coupled algorithm. The first component, the clock drift estimation and fault detection algorithm, uses the position estimates from the mechanized INS as well as the measured carrier phase from the receiver as described previously. After measurements are checked, the standard loosely coupled update occurs, during which a GNSS single point position (SPP) estimate is calculated and the GPS/INS filter is updated. To obtain a smoothed estimate of the clock drift, the SPP estimate of bias and drift are used as measurements for a federated clock model Kalman filter. The clock Kalman filter implements the two state model discussed in Chapter 4, with the Q matrix derived from the appropriate power law coefficients and the R matrix a function of the GNSS receiver confidence in the clock state estimates [80].

The smoothed clock state estimates are used to generate the confidence thresholds for the subsequent measurement check by the fault detection algorithm proposed in this thesis. By

using an additional filter to smooth the drift estimates, large spikes in drift due to the receiver clock noise are not propagated in the drift bounds used for fault detection. A complete block diagram for this implementation is shown in Figure 5.4.

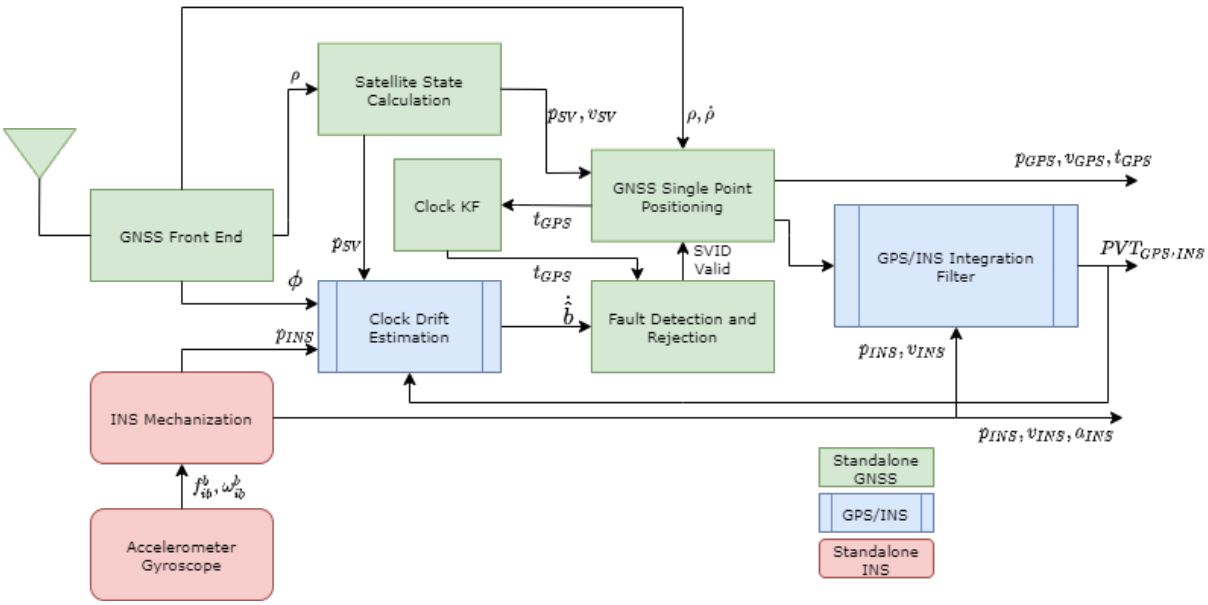

Figure 5.4: Loosely Coupled GNSS/INS Detection Algorithm Implementation

5.3.2 Tightly Coupled Implementation

When implementing a tightly coupled GNSS/INS algorithm with the proposed fault detection method, additional federated filters are not necessary. The tightly coupled measurement update includes the clock model as shown in Chapter 3, which smooths the clock state estimates preventing noise from affecting the drift bounds. The removal of the federated clock filter is the primary difference between the implementations of the loosely and tightly coupled drift fault detection implementations, aside from the inherent differences of the basic coupling algorithms. The block diagram for the tightly coupled implementation is shown in Figure 5.5

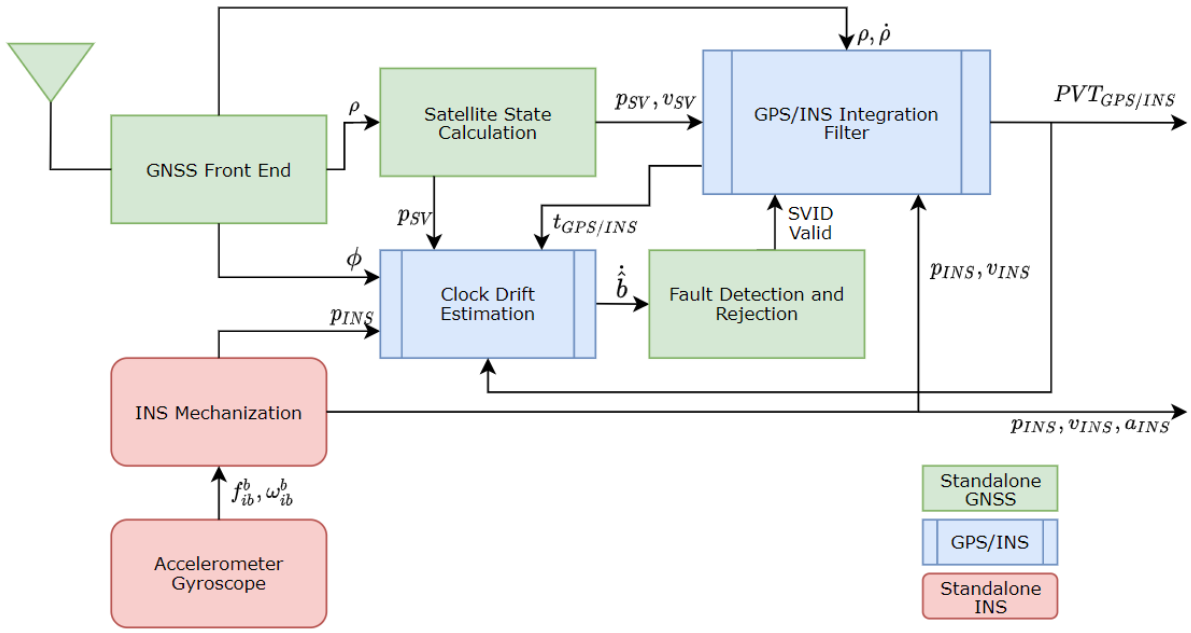

Figure 5.5: Tightly Coupled GNSS/INS Detection Algorithm Implementation

5.4 Fault Detection Thresholds

To determine whether the individual channels are faulty, the estimated drift values must be compared to a metric or threshold. The chosen threshold is calculated by taking the uncertainty in the clock drift estimate as well as the uncertainty in the current and previous position estimates. The position estimate uncertainty is incorporated as both positions are used to estimate the clock drift as shown in Equation (5.1) or Equation (5.4). The estimate of the uncertainty at the previous time step $k-1$ is obtained from the state covariance matrix obtained from the measurement update of the coupled filter., P_{k-1}^{+}. The clock drift uncertainty and current position uncertainty are obtained from the propagated state covariance, P_{k}^{-}. To account for the uncertainty in the range estimate, the position uncertainties are mapped onto the unit vectors between the satellite and the receiver as shown in Equation (5.8).

$$\sigma_{pos}^{2} = H_{k-1} P_{k-1}^{+} H_{k-1}^{T} + H_{k} P_{k}^{-} H_{k}^{T} + R \tag{5.8}$$

The measurement noise for the given states is captured by the uncertainty as well. The total variance at the current time k, given by Equation (5.9), is the combined position and drift

uncertainty.

$$\sigma^2_{drift} = P_{\dot{b}} + \sigma^2_{pos} \tag{5.9}$$

For the loosely coupled implementation, clock drift bounds are calculated from the clock drift covariance calculated by the federated clock Kalman filters as well as the position variances calculated from the integration filter. Since the federated filter relies on the accuracy of the GNSS solution to determine the variance on the clock drift, the bounds are slightly increased from those calculated by the tightly coupled solution. This increase can be attributed to the method of deriving the coupled filter measurement variance, which in this case was calculated from the diagonals of the GNSS LS filter confidence estimate. Since only the diagonal elements from the GNSS confidence estimate were used, the drift bounds were larger than they would potentially be if the full matrix was used. When including the off-diagonal elements from the GNSS confidence estimate, the loosely coupled confidence bounds decrease and closely mirror those of the tightly coupled implementation. This difference does not drastically impact the results, as the magnitude of the errors from the TCXO driven transmitters are typically greater than any of the bounds and the magnitude from the high quality oscillators typically fall within the bounds. More detail on these results is discussed in the following chapter. The GNSS SPP solution is noisier than the Kalman filter implementation in the tightly coupled filter, so the overall position variance calculated from the coupling filter is slightly increased as well. Figures 5.6 and 5.7 show the variances calculated for a loosely and tightly coupled data set, respectively.

As expected, the tightly coupled implementation achieves a tighter bounding of the states due to the use of previous information to inform the confidence at the current time step instead of calculating an individual solution at each step. This increase in confidence leads to a tighter fault detection threshold, indicating that errors in the drift estimate have to be close to what the true drift value is to corrupt the receiver. Additionally, the drift bounds grow more slowly over time when the receiver is paired with a higher quality IMU during signal outages, allowing the receiver to survive longer periods of time before faults can potentially enter the receiver.

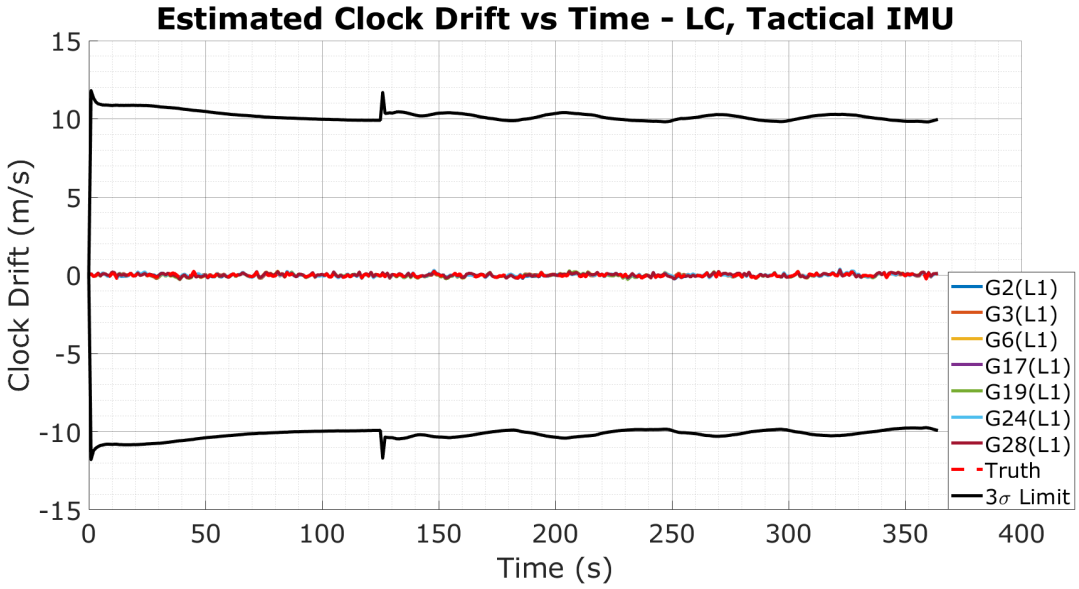

Figure 5.6: Drift Bounds for Loosely Coupled Implementation

This threshold is the metric against which estimated clock drift values are compared against to determine if there is a fault on each individual channel. For the bounding of errors in the proposed algorithm, a 3σ threshold was used. By using a 3σ threshold, 99.7% of authentic drift values fall inside of the calculated bounds, leaving only a small margin that may be flagged as a false positive. This threshold is a good compromise of accuracy and detection capability. If the threshold used is too small, such as a 1σ bound, the false positive rate will be significantly higher, and a large threshold, such as 6σ, will lead to an increase in false negative cases. In this application, both false negatives and false positives impact the performance of the receiver, however false negatives have the potential for far greater impacts depending on the source of interference. A false positive may cause GNSS availability to degrade or be dropped, leading to an unnecessary interruption of GNSS availability. A false negative would potentially introduce faulty measurements into the navigation processor, allowing the position to be controlled by the source of the spoofed signals and corrupting the integrated navigation solution. Both of these issues can be compensated by the inclusion of additional sensors and integrity checks, which are not explored in this thesis.

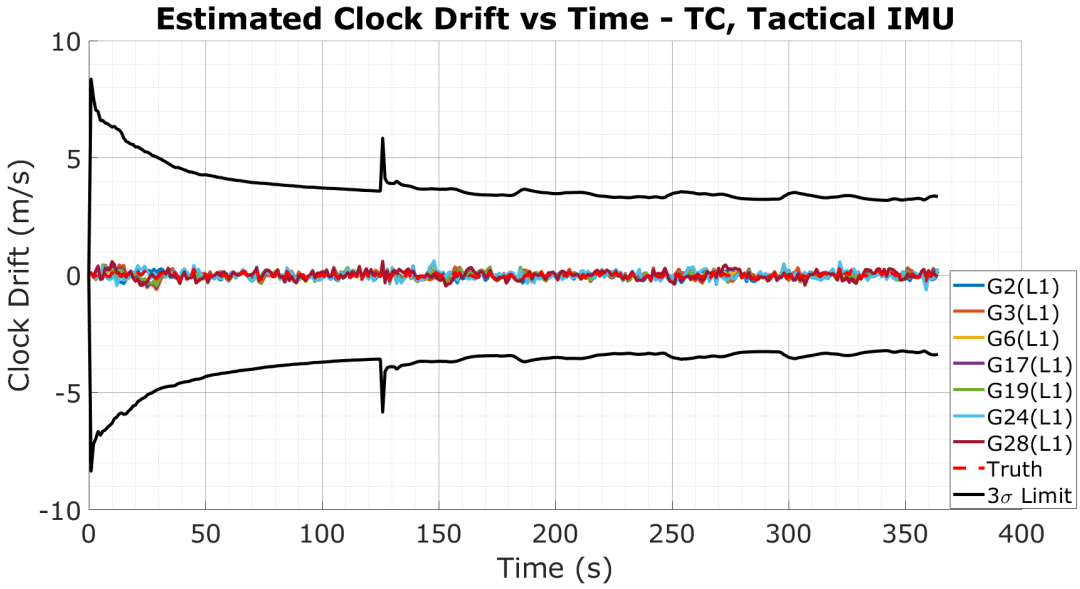

Figure 5.7: Drift Bounds for Tightly Coupled Implementation

5.5 Sensitivity Analysis

As shown in Equation (5.7), the change in estimated range depends upon both the previous and current estimates of the user position. For a static location, this value can be surveyed and held constant, simplifying the calculation and removing errors introduced by an incorrect user position. For a dynamic receiver, the position will never be perfectly known, so the uncertainty in the position is an additional source of error in the estimate. Figures 5.6 and 5.7 show the threshold during a benign environment for the loosely and tightly coupled implementations, respectively. Clock drift estimates within the bounds shown will be marked as valid and used to update the navigation filter, while estimates outside of the bounds will be rejected. If an outage occurs, both the position error and the fault threshold will grow as a result of the INS mechanization and the covariance propagation. Without corrections, the INS solution may drift significantly depending on the IMU performance, leading to the potential of the algorithm calculating an incorrect drift value due to position error. To determine the potential of this occurrence, a sensitivity analysis was conducted by generating measurements for a static receiver at a known location and varying the position error across each axis. This was done as a comparison to the results shown in [20], which showed the a position error of $50km$ would introduce $5m/s$ error in the clock drift estimates. For the sensitivity analysis conducted here, each axis

was varied from $-50000m$ to $50000m$ of position error and the drift was calculated using the same method shown in Equation (5.6). The results for the analysis are shown in Figure 5.8.

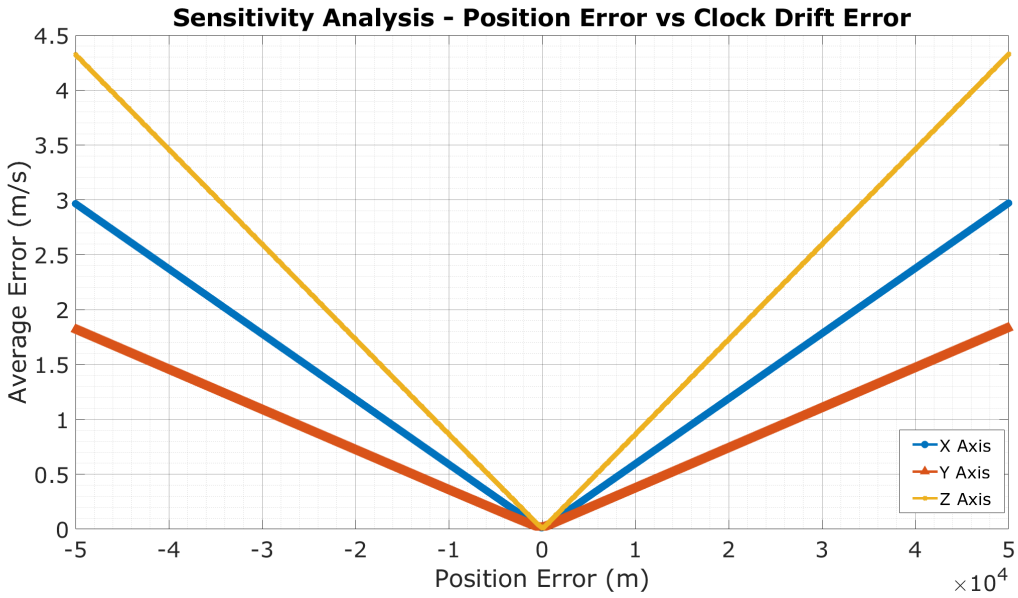

Figure 5.8: Analysis of Position Error Influence on Clock Drift Estimates

For this analysis, it is assumed that the satellite position error is negligible. Given the particular location and satellite geometry shown in Figure 5.8, it can be seen that the Z direction contributes the maximum error. Additionally, at a value of $50km$ ($29km$ in each direction), the magnitude of the clock drift error is equal to $5.3m/s$, which reflects what is shown in [20]. If the error is in a single linear direction, however, the error magnitudes are lower than what is observed if there is error in all three directions.

Simulating a second batch of satellites, it is observed that the Z error contributes the most to the clock drift estimate error while the contributions from the X and Y components have flipped which is larger. In the second example, the clock drift with a total position error of $50km$, the total drift error is equal to $5.5m/s$, which remains consistent with both the initial analysis and the previously cited paper. As was observed in the first study, the linear errors in each individual direction are lower than if the error is in all three directions. The results for the second analysis are shown in Figure 5.9.

In both analyses, the total error when the position estimate is close to the true value is well under $0.5m/s$, indicating that the clock drift estimates are predominantly affected by the

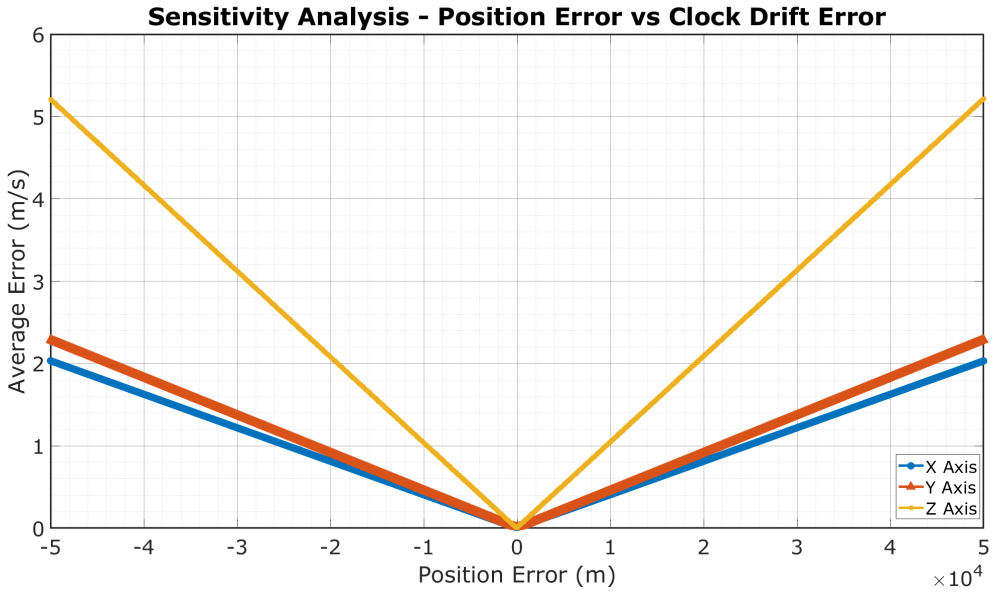

Figure 5.9: Analysis of Position Error Influence on Clock Drift Estimates - New Satellite Geometry

differences in the received measurements. The position error will introduce some small errors into the estimate of drift calculated using differenced carrier phase measurements, however the error will not be greater than the drift threshold in a nominal case and will not significantly impact the drift estimate after an outage when true signals return. This is critical to determining which signals may be faulty and which signals may be authentic after a signal outage, either artificially created or due to an obstruction such as a bridge or urban canyon.

5.6 Conclusion

By calculating an estimate of receiver clock drift independently of a GNSS measurement update, fault detection can be performed prior to introducing erroneous measurements into the overall navigation algorithm. Using the uncertainty in the EKF propagation as well as the clock drift propagation, a fault detection threshold can be determined and clock drift estimates that exceed this bound can be rejected prior to updating the GPS or GPS/INS states. This method, however, is highly dependent on the confidence in the estimated user position and clock drift at the end of a signal outage. This confidence is directly related to the quality of inertial sensor and receiver clock that are in use.

By providing faster measurements than GNSS systems, the INS can span the gaps between GNSS measurements and capture the dynamics that occur during these gaps. These measurements are used to estimate the position at the current epoch prior to introducing potential errors into the navigation solution. One drawback to the coupled integration is if corrupted measurements are passed into the navigation filter, the error estimates for the INS will become corrupted as well. This has the potential to further derail the navigation system, as subsequent errors will not be detected as the filter is influenced by the false GNSS signals.

In the following sections, software simulation and detection method implementation will be discussed, followed by software results. Results from hardware tests conducted using the same algorithm will then be shown and compared to the simulation results. The construction of the simulation is described in Chapter 6 along with the Monte Carlo simulations and HWIL tests performed.

Chapter 6

Fault Detection Algorithm Testing and Results

To test the developed integrity monitoring algorithm, several tests were conducted. These tests, discussed in the following sections, included several software simulations as well as hardware tests of the proposed algorithm. The implementation and analysis of the results will be further discussed in this chapter.

6.1 Software Simulations

Prior to testing on hardware systems, simulations were developed to test the feasibility of the proposed integrity algorithm. Once the simulation architecture was developed, a navigation algorithm was implemented to process the measurements generated in the simulation and to test the performance of the detection and mitigation methods discussed.

6.1.1 Simulation Environment

To test the detection algorithm, a measurement level simulation was implemented in MATLAB. The simulation consisted of a measurement level simulation of GNSS signals as well as error models for IMU values based on a simulated trajectory. The traveled trajectory was used for both the GNSS and IMU measurements, as discussed presently.

To create the needed IMU measurements, accelerometer and gyroscope measurements were generated using Airsim, an open source simulation environment used for interfacing with aircraft and other vehicles [103]. Modifications to the Airsim source code were made to provide the true body accelerations and rotational rates with no noise as an output that could be recorded

[104]. Once these measurements were logged, the accelerations and rotations were mechanized to generate a truth trajectory used for the IMU and GNSS measurement simulation. The truth trajectory generated can be seen in Figure 6.1, which consists of two laps around a figure eight shaped path. The true values were output at $100Hz$ to allow for simulation of IMU output rates up to this rate.

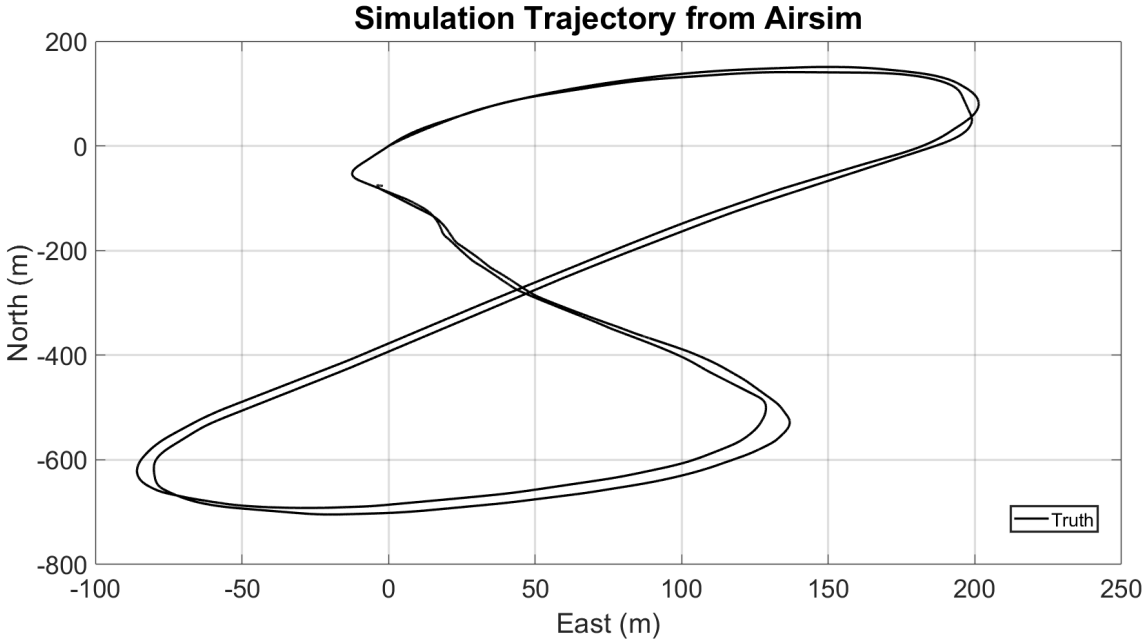

Figure 6.1: Ground Truth Trajectory From Mechanized Values

Once the truth positions were established, the IMU and GNSS measurements were simulated. IMU measurements were created by adding the desired noise processes to the IMU outputs. For all simulations, no bias term was added, emulating a calibrated unit with turn on bias removed. Additionally, misalignment and scale factor errors were neglected for this work. Both MEMS and tactical IMUs were simulated for testing of the GNSNS/INS algorithms. The noise and random walk standard deviations were referenced from GPS-INS-SIM, an open source IMU simulator [105]. These values can be seen in Table 6.1.

Gravitational and rotational forces related to the Earth were neglected in the simulation. Due to this omission, simulated body specific forces were a direct function of the true accelerations and rotation rates taken from Airsim with bias, white noise, and random walk errors

Table 6.1: IMU Noise Model Parameters

IMU Grade	Accel Noise $(m/s/\sqrt{hr})$	Accel Random Walk (m/s^2)	Gyro Noise $(^\circ/\sqrt{hr})$	Gyro Random Walk $(^\circ/hr)$
MEMS	0.05	2e-4	0.75	10
Tactical	0.03	5e-5	0.25	3.5
Perfect	0	0	0	0

added. The simulated measurements are shown in Equation (6.1) and Equation (6.2).

$$f_{ib}^b = f_{ib_{true}}^b + \eta_a + rw_a \tag{6.1}$$

$$\omega_{ib}^b = \omega_{ib_{true}}^b + \eta_g + rw_g \tag{6.2}$$

GNSS measurements were simulated for GPS and Galileo satellites following the same trajectory described above. Satellite states were generated using receiver independent exchange format (RINEX) files containing satellite ephemerides downloaded from the NASA Crustal Dynamics Data Information System (CDDIS) database [106]. A current time of day (TOD) is specified to allow for the proper simulation of satellite locations above the user position. For this simulation, all visible satellites were simulated, allowing the user algorithms to determine which satellites to use when processing. Often times an elevation mask is applied, removing low elevation satellites from navigation updates. Figure 6.2 shows a single epoch of the simulation with LOS vectors drawn from each satellite to the user position.

Once the satellite positions were simulated, measurements were generated using the known receiver and satellite locations as well as the desired error parameters. Clock errors were generated using the models discussed in Chapter 4, with the quality of oscillator varied for each simulation. Noise was added to each measurement, with the 1σ values used shown in Table 6.2. The measurements for each satellite were generated using the same observation equations that were shown in Chapter 2.

The carrier wavelength, λ_c, is determined by the signal that is generated for the simulation. For this thesis, GPS L1 and L2 and Galileo E1 and E5b measurements were generated. Other measurements, such as Galileo E5a or GLONASS measurements could be simulated, but were

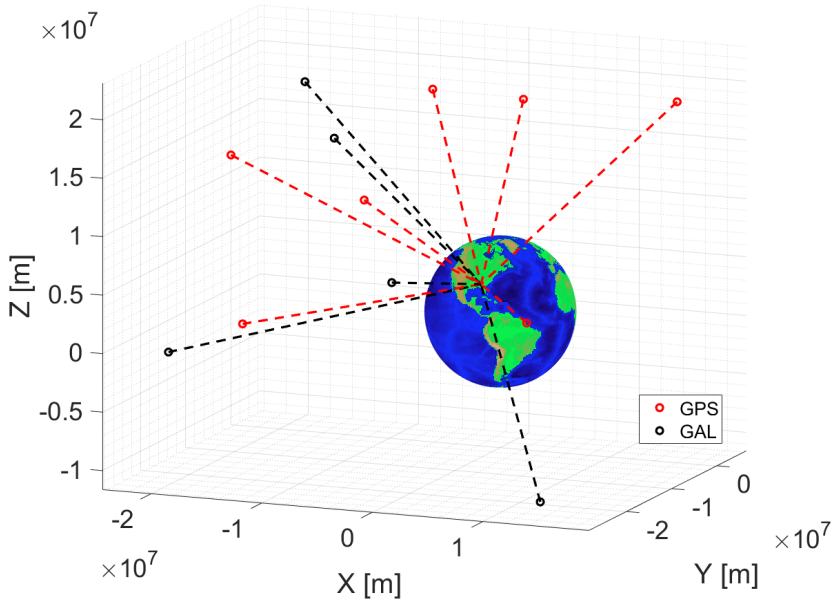

Figure 6.2: Simulated GNSS Satellite Positions

Table 6.2: GPS Noise Model Parameters

Measurement	Noise Standard Deviation (1σ)
Pseudorange	$4m$
Doppler	$0.05m/s$
Carrier Phase	$0.05m$

not included in this work. By simulating two distinct frequencies from several GNSS systems, cross-constellation and cross-frequency checks can be implemented if interference is detected in an attempt to determine the type of attack. As discussed in Chapter 2, low end meaconers typically operate at a single frequency band, i.e. the L1 band. In this case, if the L1 and E1 measurements were faulty, but L2 and E5b measurements were not, it could be determined that the L1 band had some sort of interference and measurements from this band should not be used in the navigation solution. These generated measurements were tested to ensure the correct position and velocity were calculated, and the estimation errors are shown in Figures 6.3 and 6.4. A nonlinear weighted least squares (NLWLS) estimation algorithm was used for this testing.

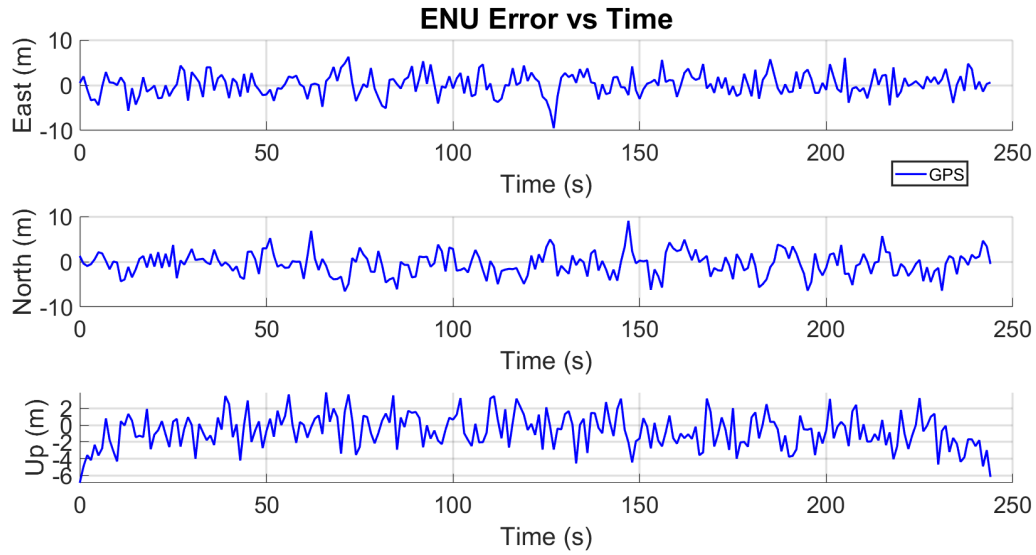

Figure 6.3: GNSS Solution Position Error Using NLWLS

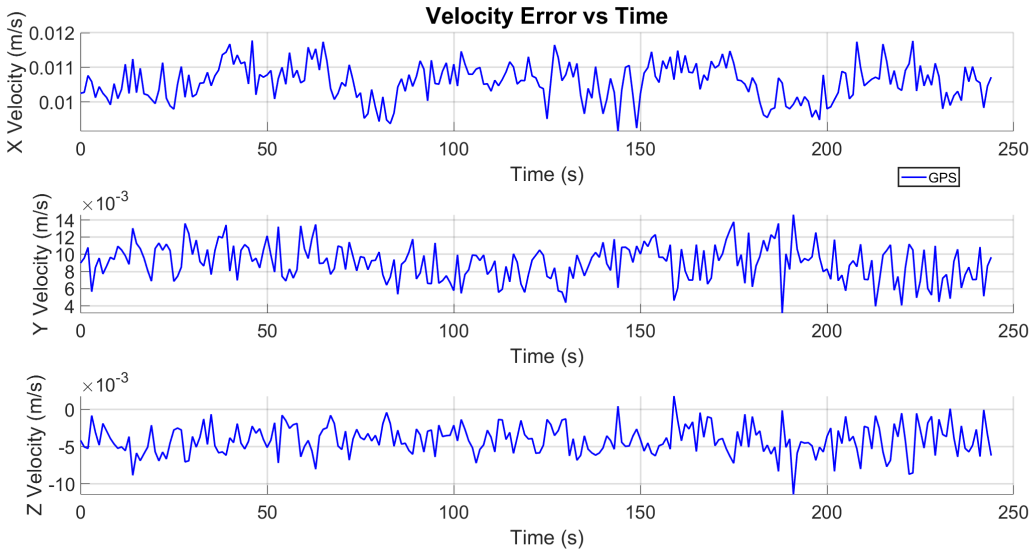

Figure 6.4: GNSS Solution Velocity Error Using NLWLS

Meaconer and spoofer measurements were generated in the GNSS simulation as well, with different methods for each. Meaconer measurements were generated using a multipath model described in [54] with several modifications to account for software delays and additional clock dynamics. In addition to the extra range from the meaconer to the receiver, signal processing delay from the computer hardware and RF blockchain is introduced. This delay further shifts the received code phase, tricking the receiver into thinking time is earlier than it actually is. The errors on the meaconer measurements are generated using the same variances shown in Table

6.2 above, and the meaconer clock is varied from simulation to simulation. As is the case with multipath, the carrier phase measurement is affected in the same manner as the code phase. The spoofed code phase measurements are a function of the range from the satellite to the meaconer and then to the receiver as well as the signal processing delay. The spoofed pseudorange and carrier phase measurements were simulated using the formulation shown in Section 2.6

Typical multipath models assume that signals will be reflected from static objects such as buildings in urban canyons. This assumption concludes that since the object that is causing the reflection is static, the received Doppler shift is the same as that of the true signal. In the case of a meaconer, this is not the case. The meaconer LO frequency error will have an effect on the measured Doppler shift at the receiver when receiving counterfeit signals. In the case of a dynamic meaconer, the frequency error of the meaconer LO and the motion of the receive antenna both play a part in the frequency shift of the retransmitted signals. The measurement model for the spoofed Doppler measurement at the receiver is shown in Equations (6.3) - (6.5).

$$f_t = \frac{1 - \dot{r}_s tc}{1 + \frac{\dot{b}_t}{c}} \tag{6.3}$$

$$f_r = \frac{1 - \frac{\dot{r}_t r}{c}}{1 + \frac{\dot{b}_r}{c}} \tag{6.4}$$

$$\dot{\rho} = (f_t + f_r) * f_c - f_c + \eta_{\dot{\rho}} \tag{6.5}$$

To test the meaconer measurements, a simulation was conducted which started with clean measurements, followed by a period of no signal, followed by a period of spoofed measurements. When captured by a meaconer, the receiver estimates of position should show the meaconer receive antenna and the estimated clock bias should be larger than the true clock bias, indicating the receiver thinks that time is earlier than it actually is [51]. For this simulation, a static meaconer was simulated with a TCXO, and a dynamic receiver driven by a CSAC was simulated using the figure eight trajectory shown above. The results of this simulation are shown in the figures below, with Figure 6.5 showing the comparison of the true and estimated

trajectories and Figure 6.6 showing the true and estimated receiver clock states. This simulation confirms that the meaconer model is working as expected and is representative of what has been observed in academic testing of meaconers.

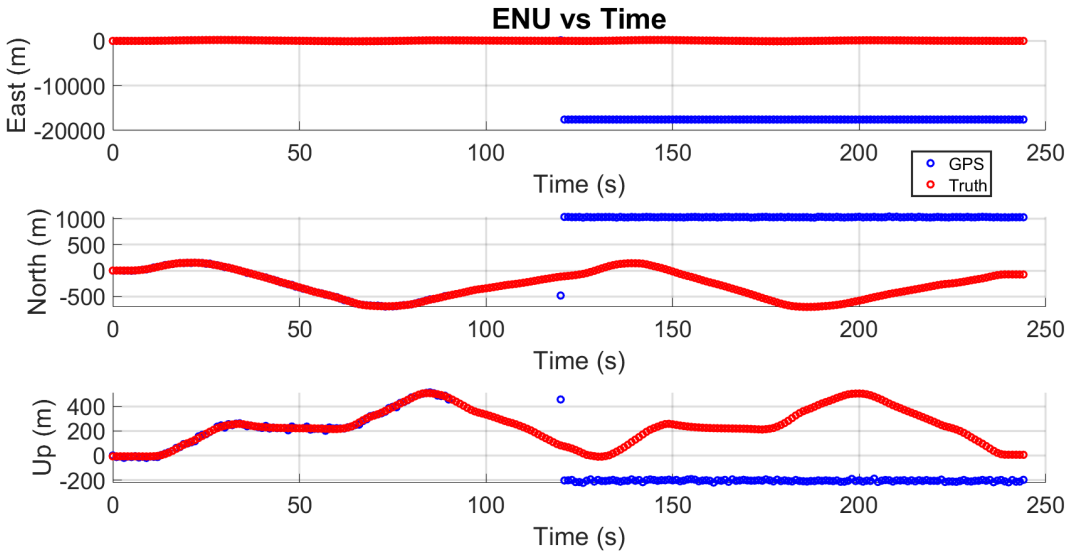

Figure 6.5: Comparison of Spoofed and True Positions - Meaconer

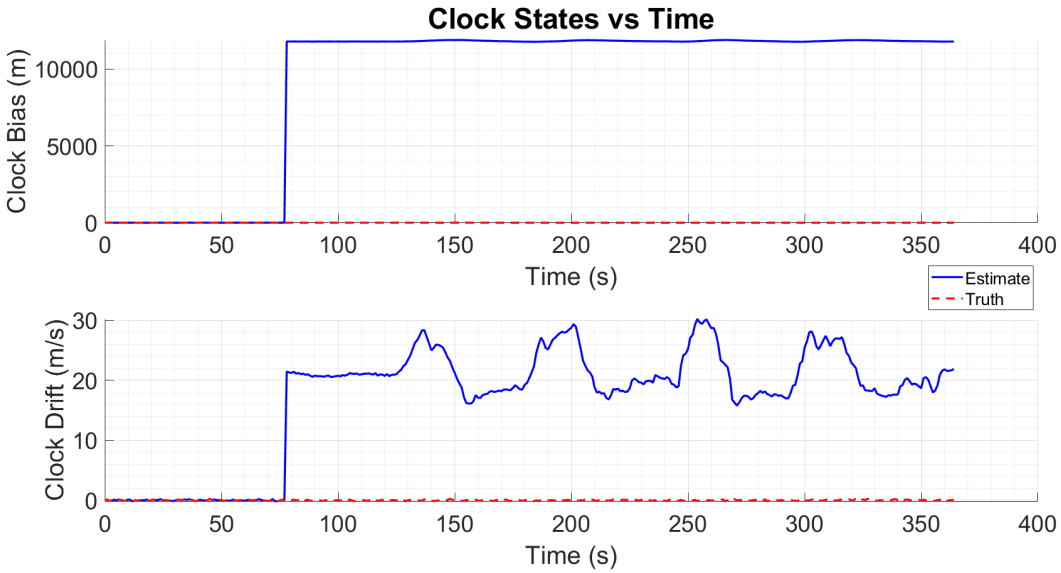

Figure 6.6: Comparison of Spoofed and True Clock States

In live sky environments, it is common for a meaconer to only capture a few of the channels in the receiver. This behavior was reflected in the simulation environment, allowing the

user to determine the number of signals to switch from true measurements to spoofed measurements. This implementation allows for the more thorough testing of the detection and mitigation effectiveness of the proposed algorithm in a variety of situations.

Simple spoofers mimic many of the aspects of a meaconer, except the resulting position solution can be anywhere in the world and the time solution can be arbitrary based on the generated data. To simulate the measurements generated from spoofed signals, the measurement models used for truth data were used with the receiver position at the desired spoofing location. In addition to the true measurements, errors from both the transmitter and receiver oscillators were introduced. In each simulation, a period of GNSS unavailability occurred prior to the spoofing. This emulated forcing the receiver into reacquisition, and after the outage the spoofed measurements were passed to the navigation algorithm. A comparison of both meaconer and spoofer simulation results to HWIL testing will be discussed in Section 6.2.

In both the case of the meaconer and the case of the spoofer, the transmitter clock will introduce additional clock dynamics based on the varying quality of the clock. To account for variability in different oscillators, the bias and drift of the transmitters were initialized using zero mean normally distributed random variables with a variance equal to the variance of the clock type after one day of continuous operation. For atomic standards and OCXOs, warm up periods are typically required, and one day was chosen to allow for any oscillator to be considered sufficiently warmed up. Using the two state clock model, a Monte Carlo simulation was conducted for each clock type, assuming perfect initial phase and frequency. Each clock type was simulated 1000 times, and the clocks were allowed to vary randomly according to the process noise covariance disccusssed in Chapter 4 during each run. The variance at $t = 86400s$ of both the phase and frequency was taken across all iterations, and was used as the variance for the initial phase and frequency for the each simulation that was generated for testing. Figures 6.7 - 6.9 show the simulated clock states for each Monte Carlo test.

As expected, the TCXO varied the greatest during the time shown, and the Rb standard varied the least. For the simulations, the initial drift is of particular interest because the transmitter clock drift introduces frequency error in the transmitted signal, which is used to monitor

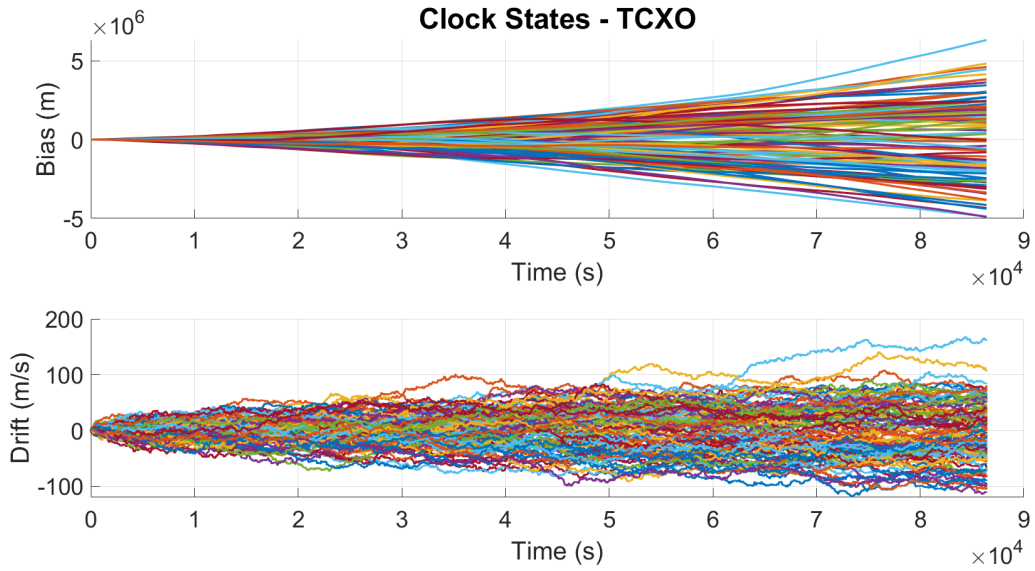

Figure 6.7: TCXO Monte Carlo Results

faults. Each drift value is zero mean, and the magnitude of frequency error significantly decreases as the quality of oscillator increases. The standard deviation for each transmitter clock phase and frequency is shown in Table 6.3, with the values shown used to generate random phase and frequency values used for each individual simulation and the Monte Carlo simulations discussed in Section 6.1.3.

Table 6.3: Oscillator State Standard Deviations at $t = 1$ Day

	TCXO	OCXO	Rb
Bias (m)	$2.806e6$	$1.495e4$	19.071
Drift (m/s)	57.682	0.297	0.003

The standard deviations indicate that the simulated TCXO varies significantly over the course of a day, while the Rb standard frequency error is extremely stable. Because of this, it is hypothesized that a transmitter using a TCXO will be relatively easy to detect, while a transmitter with a Rb standard will be virtually impossible to discern from satellite clock error due to the lack of frequency error introduced by the LO. Results showing how the algorithm performed will be analyzed in detail in the following sections.

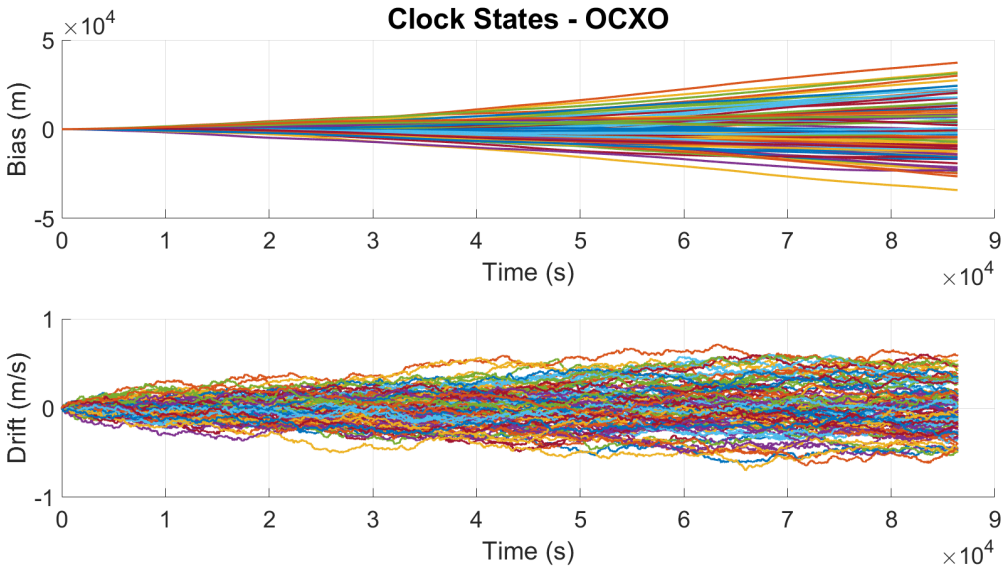

Figure 6.8: OCXO Monte Carlo Results

6.1.2 Meaconing Results - Single Tests

For each scenario, a transmitter driven by a TCXO is tested against a receiver driven by a CSAC. Each configuration of IMU and integration is tested for both the full capture and partial capture, and the results are shown in Sections 6.1.2.1 and 6.1.2.2. Following this section, a more in depth analysis of the algorithm is conducted using a Monte Carlo simulation where more variations of hardware are tested and the results analyzed.

One interesting aspect to note is that while during nominal operation the value calculated in Equation (5.6) represents the receiver clock drift, this is not the case when the the equation is used with values received from a meaconer or spoofer. The estimate of the change in range, $\Delta\hat{r}$, remains consistent with the estimate of change in range prior to the outage as the position estimates rely on the INS output, not a GNSS position. The differenced carrier phase measurement, $\Delta\phi$, no longer depends solely on the change in receiver clock bias but also the change in position of the satellites above the repeater and the change in range between the repeater and the receiver (or in the case of a spoofer, only the change in range between the spoofer and receiver). Since Equation (5.6) assumes a line of sight to each satellite, the drift estimates calculated when using non-line of sight (NLOS) measurements are not consistent with the true clock drift of either the receiver or transmitter. An example of this is shown in Table 6.4, where

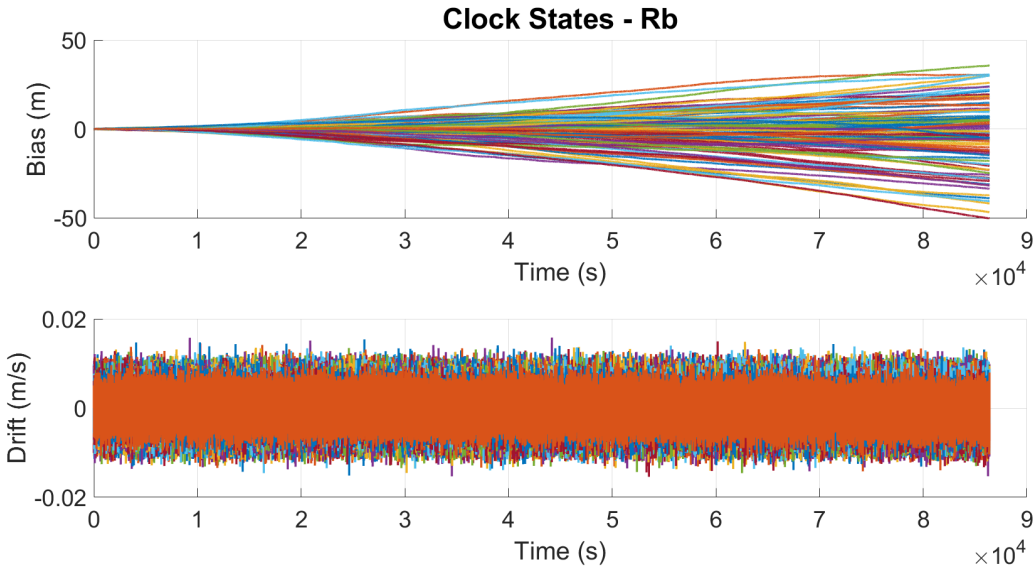

Figure 6.9: Rb Standard Monte Carlo Results

the estimated range difference and differenced carrier phase for both authentic and meaconed measurements are shown. The calculated delta range and delta carrier values are taken at the same epoch from a simulated data set where one does not have any interference signals present and the second has all channels captured by a meaconer.

Table 6.4: Comparison of Authentic and Meaconed Measurement Differences

PRN	$\Delta\hat{r}(m)$	$\Delta\tilde{\phi}(m)$ (True)	$\Delta\tilde{\phi}(m)$ (Meaconer)
2	-385.7060	-384.5664	-375.7359
3	343.6340	344.7979	376.6498
6	-78.1410	-76.9891	-51.6137
17	80.74944	81.9050	117.8887
19	-52.8797	-51.7337	-19.7858
24	46.7580	47.8971	56.3759
28	322.6777	323.8573	352.0608

Table 6.4 shows the large difference between the estimated ranges and the measured carrier phase values that occur as a result of the position errors. When the receiver clock drift is non-zero mean as it is in the case shown above, there will be an offset between the estimated range difference and the difference in carrier measurements. This offset, however, will be essentially constant based on the value of the clock drift and the time interval between the two measurements. An example of this, taken from recorded RF data, is shown in Table 6.5 below.

The interval between measurements for this data set is $\Delta t = 1s$, which means the difference between the differenced carrier and the differenced range estimates is equal to the clock drift estimate. In this case, the estimated clock drift from the least squares is roughly $-19m/s$, which corresponds to what the estimate is when calculated using differenced carrier phase.

Table 6.5: Delta Range Estimate and Delta Carrier Measurement - Nonzero Clock Drift

PRN	$\Delta\hat{r}(m)$	$\Delta\tilde{\phi}(m)$ (True)
2	-360.9230	-340.8510
3	357.7636	377.3149
6	-58.7514	-38.5390
12	-677.5849	-658.1280
17	139.1920	159.1338
19	-9.9466	10.0364
24	101.9888	121.6914
28	346.5384	366.7233

In the above example, each differenced carrier measurement is roughly $20m/s$ greater than the estimated delta range. This difference is consistent across all measurements, and indicates that the receiver clock has a drift rate which is estimated to be about $-20m/s$, since the delta range measurements are subtracted from the delta carrier measurements, which is consistent with the least squares and EKF estimates of drift when using a standalone navigation algorithm. If the differences between the estimated ranges appear random, as is the case in Table 6.4, it can be assumed that there are faults in the measurements. By using the confidence bounds shown previously, the proposed algorithm looks for any jumps in the clock drift as a result of both transmitter clock influence as well as position error introduced by the false measurements. For each individual scenario analyzed here, it is assumed that the transmitter is an inexpensive software radio driven by a low quality TCXO. Various combinations of transmitter and receiver clocks are analyzed in the Monte Carlo simulations discussed later in this chapter.

This same effect is observed when a spoofer generates a random position, similar to the behavior of a meaconer. In this case, the drift estimate results would look very similar to what is shown in Table 6.4, where each channel estimate is not consistent. In the case of a spoofer that is able to track the receiver position, however, the drift estimates become the combination

of the receiver and transmitter clock alone. In this case, higher quality oscillators with little to no clock drift will frequently go undetected when using the detection algorithm proposed. More details of this case will be analyzed in Section 6.1.4.

6.1.2.1 Full Capture

The full capture scenario is the best case for detection of faults, however is not necessarily the most common occurrence unless the received signals are at a significantly higher power level than live sky signals. In the following tests, the MEMS IMU tests were conducted with a single data set, and the tactical IMU tests were conducted with a second data set. While two different data sets were used, both had TCXO transmitters with clock drift magnitudes which were significantly larger than the nominal clock drift in the clear sky environment, which is shown at the beginning of each scenario. For each test, a signal outage is initiated at $t = 200s$ into the run, followed by the meaconer broadcasting on the L1/E1 frequency ($f_0 = 1575.42e6$) at $t = 210s$. The L2/E5a frequencies are unaffected in these scenarios, as it is assumed that a radio with a limited bandwidth such as a USRP is used to implement the meaconer. The L1/E1 meaconer remains on for the remainder of the scenario after the outage ends, with changes in the clock drift estimate due to a combination of actual change in the drift, position error, and differenced noise from the received carrier phase measurements. In each plot, the contributing values to the drift bounds are shown during the outage, with the position uncertainty primarily contributing to the error growth over the time of the outage. The clock drift uncertainty grows a small amount, but because a stable receiver oscillator was used, does not grow nearly as much as the IMU errors. For each case, the MEMS implementation grow at a common rate and the tactical errors grow at a common rate, with the only difference occurring from the implementation of the measurement noise in the filter update, which was discussed previously. The drift bounds during an outage are shown in Figure 6.10. As expected, the tactical bounds grow at a much slower rate than the MEMS bounds, improving the ability of the tactical implementations to survive longer outages.

The first test conducted represents the most likely sensor suite for commercial applications, which uses a CSAC receiver clock and MEMS IMU in a loosely coupled navigation

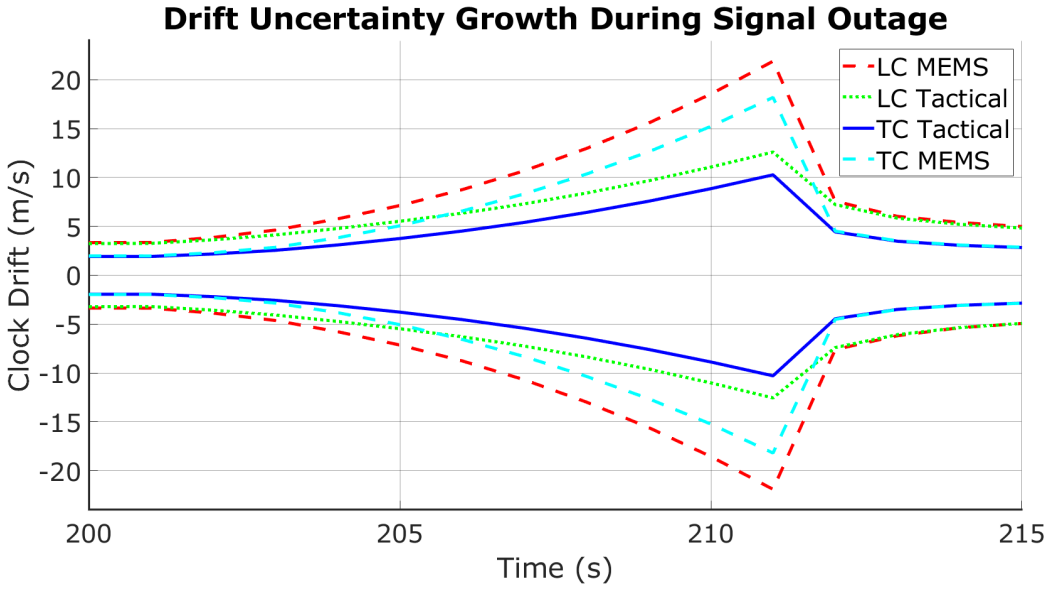

Figure 6.10: Clock Drift Uncertainty Bounds During Outage

filter. The loosely coupled implementation is the most common implementation as the filter does not require access to the GNSS receiver, only the PVT outputs as was discussed in Chapter 3. The results shown in Figure 6.11 show that the algorithm successfully detects and rejects the meaconer interference after the outage initially. After a short time, the drift estimates vary due to the changing position error, which causes several of the estimates to fall within the uncertainty bounds. When half of the meaconer channels are within the bounds, the measurements are used for a measurement update, which corrupts the navigation solution as can be seen in the figure. This could be avoided by comparing the drift estimates to each other as well as the drift bounds to see if the drift estimate variance is consistent with what it is expected to be. This method would take advantage of the fact that in a benign scenario, all channels would have the same offset which shows the drift measurement. This method, however, was not explored in this work.

As stated previously, the MEMS and tactical IMU scenarios used different inertial and GNSS errors, leading to differing clock drifts at the receiver as well as different tuning values for the INS filter. In the tactical IMU case, the outage and meaconer start points were the same as the MEMS case, with the events starting at $t = 200s$ and $t = 210s$, respectively, shown in Figure 6.12. As was the case in the first simulation, the faulty drift values are excluded from the drift bounds and are not used for updates to the state estimates, allowing the navigator to

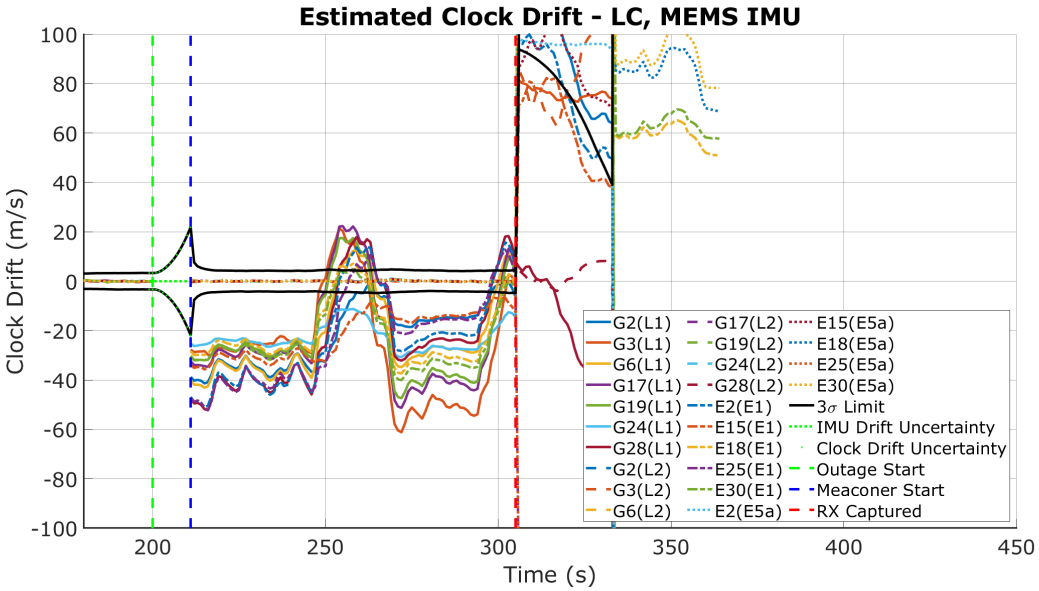

Figure 6.11: Simulation Result - LC, MEMS IMU, Full Capture

function normally. At around $t = 270s$, one channel enters the drift bounds, however the associated frequency band (in this case, L1) is still marked as faulty as over half the measurements on that frequency are indicated as bad. This allows the receiver to prevent capture when single channels may erroneously enter the drift bounds. The same trend of large changes in the drift estimate occur in this case as a result of the changing position of the receiver relative to the transmitter, which influences the drift estimate shown here.

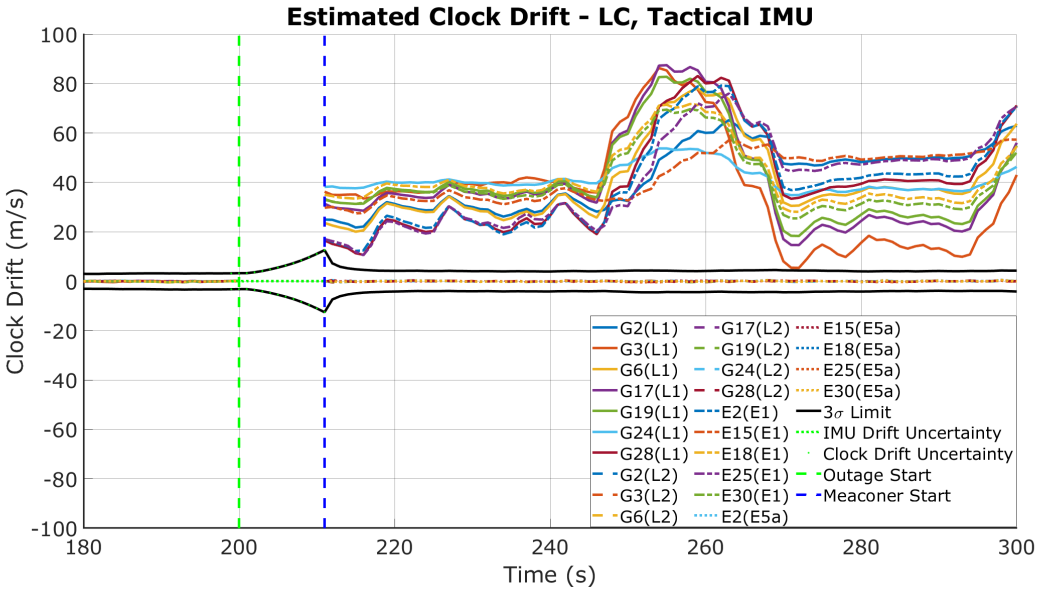

Figure 6.12: Simulation Result - LC, Tactical IMU, Full Capture

When processing the MEMS scenario with a tightly coupled GNSS/INS implementation, the navigator performs better than when a loosely coupled filter is used. This is due to the fact that the TC bounds on the estimated clock drift are tighter than in the LC case. As a result, when the drift estimates cross the confidence bounds as they did in the LC case at $t = 240s$. not enough channels are included at a single time to allow the faults to enter the receiver. Later in the simulation, however, enough faults are marked as valid within the confidence bounds and the receiver updates the states using faulty measurements. This causes the navigation algorithm to fail, as shown in Figure 6.13.

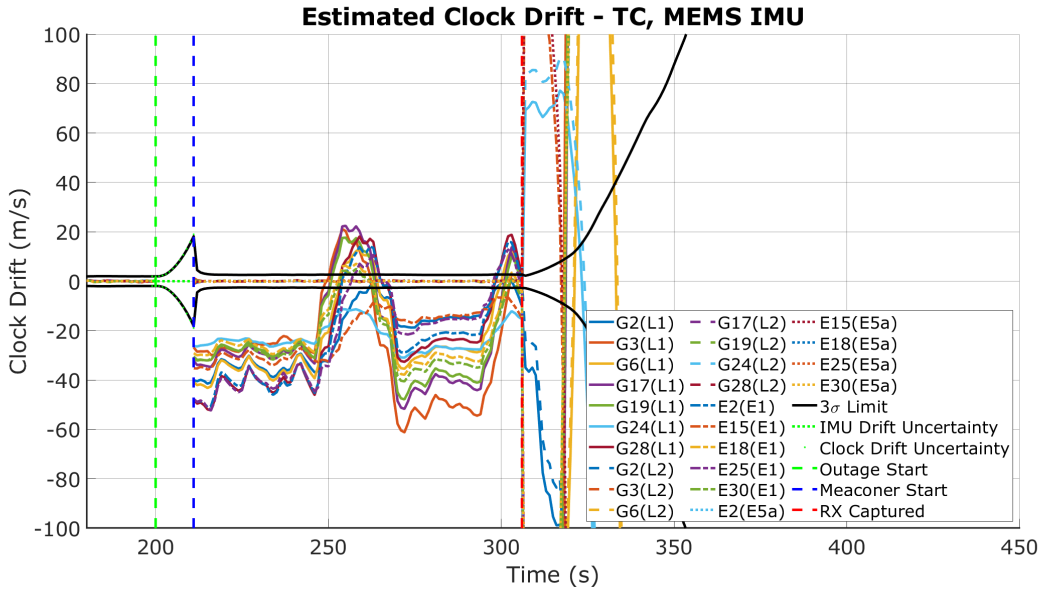

Figure 6.13: Simulation Result - TC, MEMS IMU, Full Capture

The tightly coupled implementation, shown in Figure 6.14, has the best performance of all the implementations as expected. By combining the improved estimation of the TC filter with the higher quality tactical INS, the confidence bounds and position estimates are superior to the other implementations. This can be seen by the faulty drift estimates remaining fully excluded across the entire scenario as opposed to the single channel entering the drift bounds which was shown in Figure 6.12. The TC tactical implementation was able to identify and reject the faulty measurements throughout the duration of the scenario, allowing the navigation algorithm to successfully maintain an accurate navigation solution in the presence of a single frequency meaconer.

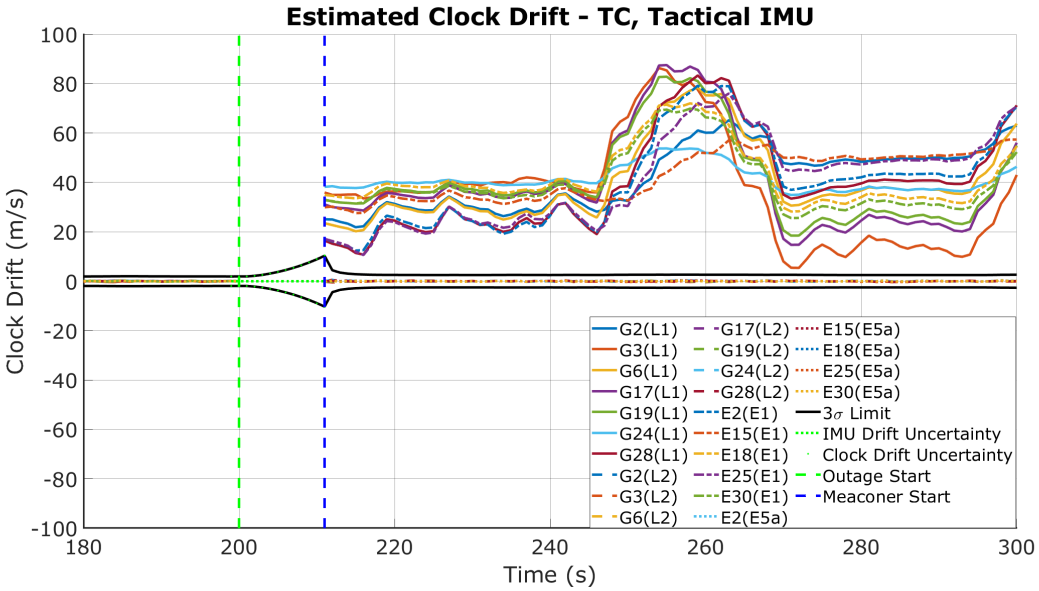

Figure 6.14: Simulation Result - TC, Tactical IMU, Full Capture

In each case where the entire L1/E1 band was fully captured by a meaconer, the drift estimation and detection algorithm was able to detect the faults present in the measurements. The MEMS cases did not perform as well as the tactical implementations, however this is mainly due to the inauthentic drift estimates crossing across across the authentic drift estimates. Because the MEMS IMU implementation confidence bounds are larger, there is a higher likelihood of a fault entering the confidence bounds if the estimates cross zero. The estimates in the MEMS implementations cross zero due to the combination of transmitter clock drift (which is negative) and the change in position between the spoofer position and the estimated receiver position. In the tactical IMU implementation, the transmitter clock error is positive, so the errors introduced by the change in position cause the drift estimates to grow larger instead of crossing the authentic estimates. The difference in the transmitter clock drift causes the scenarios to have significantly different results as shown in the results above. If both the MEMS and tactical scenarios had drift estimates that crossed zero, it is expected that the tactical IMU would outperform the MEMS implementation due to the tighter confidence bounds. When enough faults enter the bounds simultaneously, the receiver becomes captured and the navigation solution becomes corrupted, as was seen in Figures 6.11 and 6.13. If the confidence bounds are tighter,

there is less likelihood of enough measurements entering the threshold at once, preventing the receiver from being captured.

6.1.2.2 Partial Capture

Often times meaconers may not capture an entire frequency band, but instead capture several channels while the remainder of the channels continue to receive live sky or uncorrupted signals. To replicate this, a simulation was designed where a random subset of the visible satellites on a given frequency were corrupted with the rest remaining unchanged. The simulation was configured to introduce false signals on less than one half of the available channels for the desired frequency, which prevented the entire channel from being excluded due to faults. This test represented the a significantly more difficult case to detect and mitigate, and the results discussed below represent the increased difficulty.

For these tests, the scenarios were examined in the same order. Starting with the LC MEMS implementation in Figure 6.15, a similar result as observed in the full capture scenario can be seen. The faults are initially excluded after the outage ends and the meaconer is introduced, however once a single fault crosses to the inside of the fault threshold the measurement is used to update the states. When the single fault enters the navigation solution, the bounds vary and more of the faulty measurements are included in the measurement update, further degrading the solution. Once the faults are introduced into the measurement update, the navigation algorithm breaks down as was the case in the full capture and the navigation solution is no longer valid. Even though the navigation solution is corrupted, the measurements are still indicated as faulty and an alert is still able to be passed to the operator indicating that the GNSS/INS navigation solution is untrustworthy and would be able to intervene with appropriate actions to prevent system failure.

The LC tactical implementation again performed better than the MEMS implementation in the case of a partial capture of the receiver. The performance was similar to the full capture scenario, where a single faulty channel crosses the drift bounds after the meaconer is started. In this case, however, since less than half the signals on a single channel are faulty, several L1 measurements remain valid. Because of this, when the single fault crosses into the valid

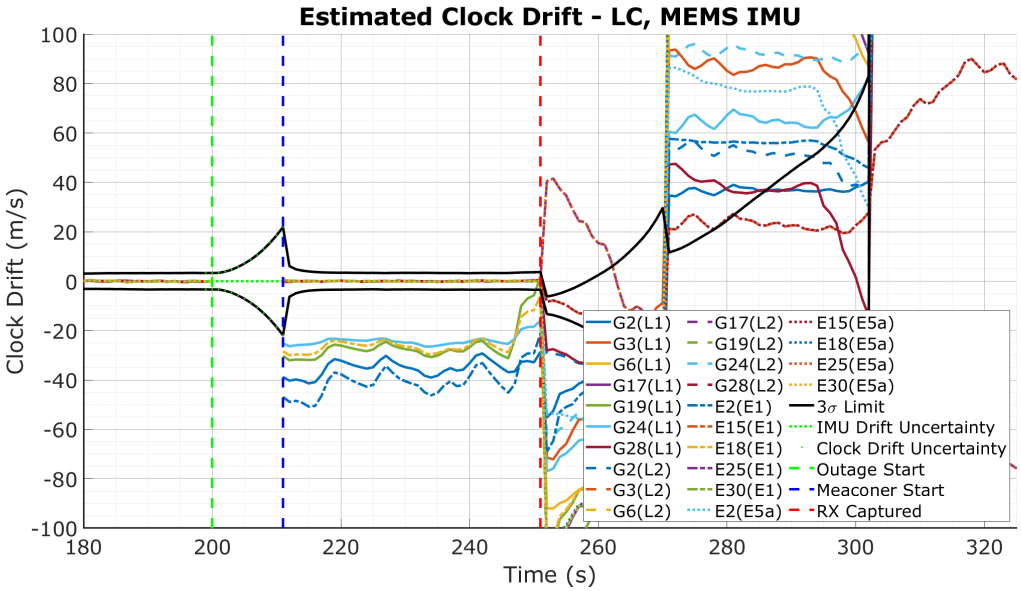

Figure 6.15: Simulation Result - LC, MEMS IMU, Partial Capture

range, it is not excluded as was done in the full capture scenario, allowing the fault to influence the navigation solution. Unlike the LC MEMS implementation shown in Figure 6.15, the uncertainty bounds do not grow as large and the remainder of the faulty measurements remain outside of the drift bounds. When the single faulty channel estimate exits the drift bounds after several seconds, the navigation solution reconverges to the correct values and the receiver is able to continue navigating for the duration of the event. The results for this scenario are shown in Figure 6.16 below.

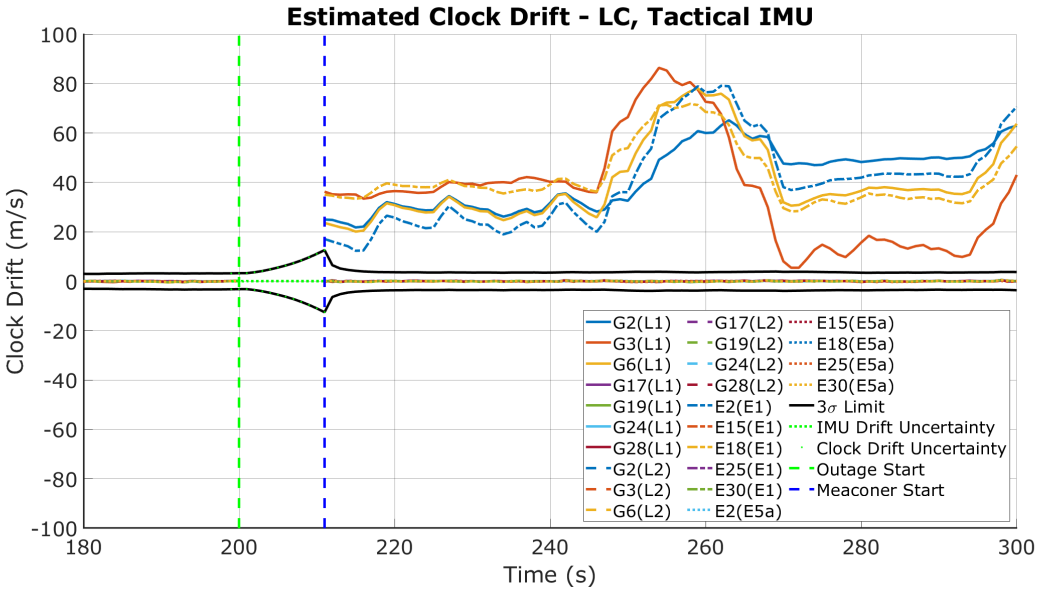

Figure 6.16: Simulation Result - LC, Tactical IMU, Partial Capture

Moving to the tightly coupled implementation, the TC MEMS scenario has similar performance to the full capture and MEMS partial capture scenarios. As was the case for the previous scenarios, when a faulty measurement crosses into the bounds as occurs in Figure 6.17, the drift confidence bounds expand rapidly and allow the remainder of the faulty measurements to corrupt the navigation solution. The resulting state estimates diverge rapidly, making the remainder of the navigation solution unusable for any applications. When comparing the TC MEMS partial result to the LC partial result, both receivers were captured at the same point in the scenario, indicating that one is not better than the other in the partial scenario.

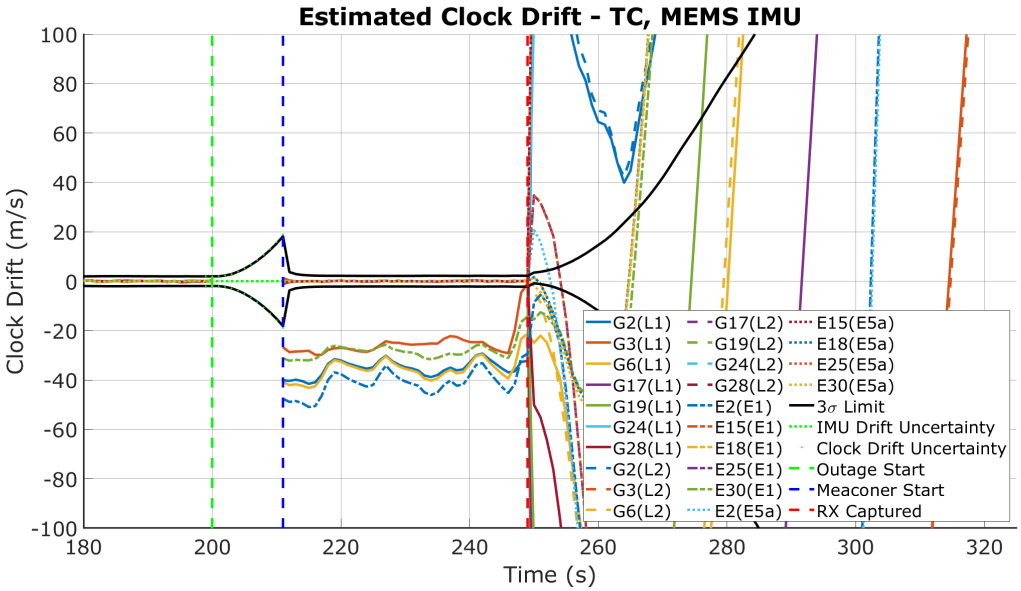

Figure 6.17: Simulation Result - TC, MEMS IMU, Partial Capture

The final test case for the partial capture event was the tightly coupled tactical IMU implementation, which is shown in Figure 6.18. The performance of this filter is again the best when the four implementations are compared. The drift bounds reconverge quickly after the outage ends using the true measurements, and the faulty estimates never enter the drift bounds. Since the faults never cross the bounding value, no degredation of the navigation solution occurs and the platform is able to maintain a valid solution throughout the event. This performance shows that the algorithm is able to successfully detect and mitigate a threat while alerting a user to the presence of faults that have the potential to degrade the navigation solution.

The scenarios analyzed here show a sampling of the performance of the proposed algorithm in a particular scenario. However, as discussed previously the algorithm performance not only depends on the clock drift of both the transmitter and receiver, but also the difference in position of the meaconer. To provide a more thorough analysis of the response to a meaconer, a Monte Carlo simulation was developed using the same detection algorithms. The algorithm and resulting performance is discussed in the following section.

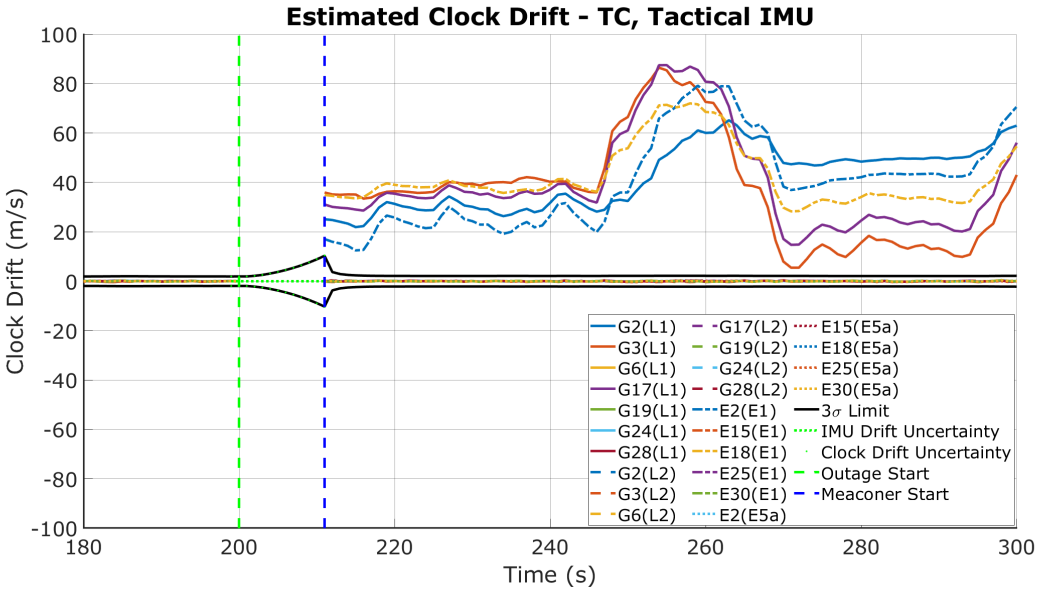

Figure 6.18: Simulation Result - TC, Tactical IMU, Full Capture

6.1.3 Meaconing Results - Monte Carlo

To provide a more thorough evaluation of the detection and mitigation performance of the algorithm against a meaconer, a Monte Carlo (MC) simulation was developed. Statistics for the performance of the algorithm were calculated across 250 iterations for each configuration of hardware and estimators. For each of the 250 iterations, a flag was set using a normally distributed random variable and rounded to the closest whole number. If the flag was set to true, the simulation involved a varying length GNSS signal outage followed by the capture of receiver channels by false signals. If the flag is set to false, the scenario is a benign scenario with no additional signals introduced into the receiver. This architecture allows for the observation of how the algorithm performs in both scenarios with interference and scenarios with no interference. Table 6.6 below shows the parameters that are varied for each simulation.

Table 6.6: Simulation Parameter Options

Parameter	Integration	IMU Grade	RX Clock	TX Clock	Outage	Capture Type
Options	Loosely Tightly	MEMS Tactical	TCXO OCXO CSAC	TCXO OCXO Rb	10s 20s 30s	Full Partial

To quantify the performance of each configuration, several statistical metrics were used. The first metrics that was used was a true positive rate (TPR) and false positive rate (FPR). In an ideal scenario, the TPR would be equal to one, and the FPR would be equal to zero, implying that the proposed algorithm was able to detect all faults and did not flag any good measurements as faulty. The TPR is calculated using the ratio of indicated true positive scenarios and the true number of positive scenarios, which is also comprised of the number of true positive (TP) and false negative (FN) scenarios. This metric is shown in Equation (6.6) [107].

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{6.6}$$

The false positive rate is a metric of how many benign scenarios are indicated as faulty, disrupting GNSS use incorrectly. Ideally, the FPR of any scenario should be zero. An FPR greater than zero indicates the detection method had a fault that was not caused by interference signals. The FPR is calculated in a similar manner to the TPR, but instead using the total number of false positives compared to the total of number of true negative scenarios, which is comprised of both indicated false positive and true negative scenarios. The total negative rate, N, is comprised of the false positive cases, FP, and the true negative cases, TN.This calculation is shown in Equation (6.7).

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{6.7}$$

In terms of an ideal detection algorithm, the TPR would be equal to 100% and the FPR would be equal to 0%. This is never achievable in a realistic scenario, however. The objective of the proposed detection algorithm here is to get as close to these metrics as possible. In this application, false negative results ($FNR = 1 - TPR$) are much more harmful than false positive results, so the main objective is to detect the maximum number of faults possible. False positive results are still harmful to the navigation algorithm, potentially dropping GNSS when there are no faults present, however this can be compensated with additional sensors or comparison of several interference detection algorithms, such as vestigial peak detection

or signal quality monitoring (SQM) algorithms that are commonly implemented at the signal level.

The final metric that scenarios were evaluated by is the mitigation accuracy, calculated as a function of the number of true positive scenarios that are detected and errors are not introduced into the navigation solution. In each simulation, the meaconer duration is set to on for a fixed period of time. To determine the success of the algorithm in each scenario, the status flag of the detection algorithm is compared to what the true status of the scenario is. If the proposed algorithm detects and rejects the faulty measurements for the duration of the interference, the simulation is said to have successfully mitigated the effects of the interference. On the other hand, if the algorithm detects the interference but only after errors have been introduced into the navigation processor or the measurements are marked as valid prior to the end of the scenarios the algorithm is classified as not successfully mitigating the interference. The mitigation accuracy is calculated as a function of the successfully mitigated true positive scenarios, TP_M compared to the total number of true positive scenarios, TP. The calculated mitigation accuracy percentage will reflect on how effective the algorithm is at preventing errors from corrupting the output navigation solution compared to simply detecting the faults that are present. The calculation to determine the mitigation accuracy, $\%M$, is shown in Equation (6.8).

$$\%M = \frac{TP_M}{TP} \tag{6.8}$$

The results for each metric are shown on a bar plot, with time on the x axis, transmitter and receiver combination on the y axis, and the performance metric plotted on the z axis. An example of the results plot is shown in Figure 6.19 below. The color at the top of each bar indicates the maximum value of that bar. For both the TPR and $\%M$, a higher value is desired, with a perfect performance equal to 100. For the FPR, the ideal value is zero, indicating that no benign scenarios were indicated as faulty and measurements removed erroneously.

The following sections analyze the performance of the proposed algorithm with an array of hardware combinations described in Table 6.6. For the meaconer simulations below, a full and partial capture scenario is analyzed in the same manner as was done above with the individual

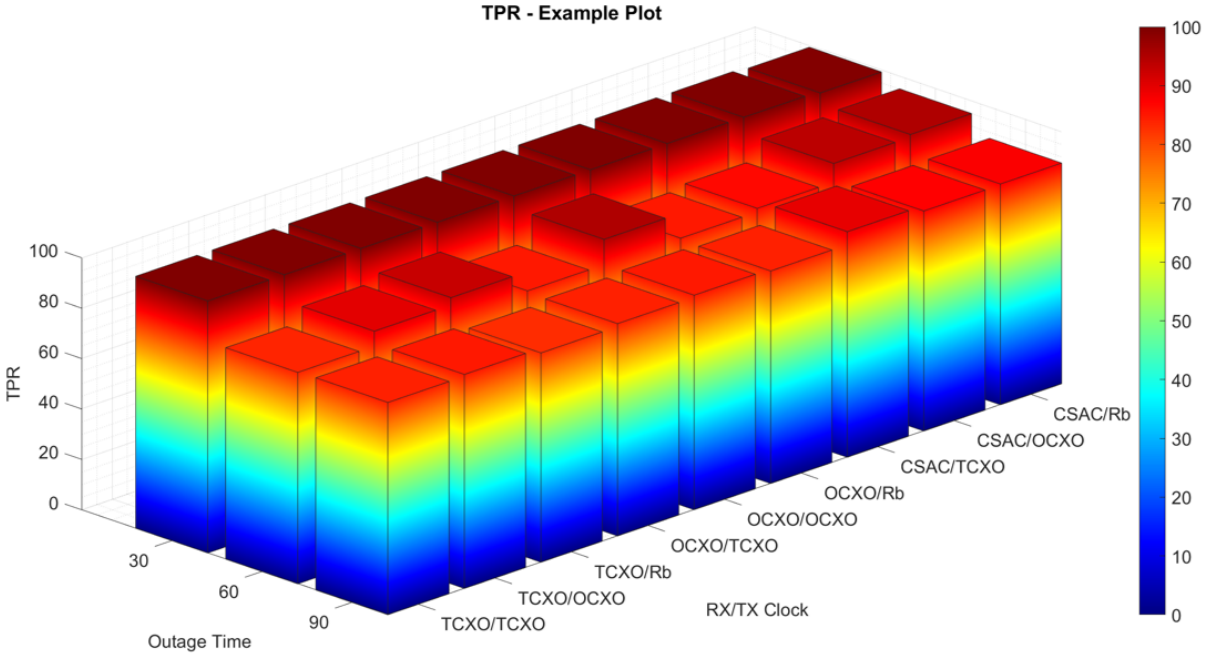

Figure 6.19: TPR Example Plot

test cases. The $\%M$ is shown for each scenario as it reveals the main differences between the performance of each algorithm configuration. The TPR for each scenario simulated was equal to 100%, so these plots are not included in this section as they do not add any additional information. The excellent detection performance is due to the errors in clock drift that occur when the estimated position from the GNSS filter no longer matches the estimate from the INS measurements, causing the clock drift values to diverge as was shown in Figure 6.11. The TPR plots, along with numerical results for the mitigation percentage for all scenarios are provided in Appendix B.

6.1.3.1 Full Capture

To examine the performance of the detection and mitigation algorithm in the presence of a meaconer that has captured all signals on a single frequency, a MC simulation is conducted using four test cases shown in the individual scenarios. The four test cases are various combinations of a loose and tight coupling integration filter and a MEMS and tactical grade IMU. These four base configurations are then combined with various quality oscillators for both the transmitter and receiver clocks as well as the outage time that occurred prior to the meaconer starting. For

all of the scenarios shown here, the FPR was zero, so no plots depicting the calculated FPR values are included. The FPR of zero shows that the algorithm is able to accurately estimate the clock drift during nominal conditions where no signals are corrupted.

The first test case is the LC integration with a MEMS IMU. The mitigation percentage for the LC MEMS scenario is shown in Figure 6.20 below. As expected, the algorithm performs very well against transmitters driven with low quality oscillators, as the drift is often large and remains outside of the detection threshold. When the higher quality oscillators are used, the drift is typically negligible, leaving only the error in position to be detected by the algorithm. While this position error is detectable, it varies as the vehicle travels along the trajectory, and the estimates may cross into the detection bounds and be marked as valid, as is the case here. Since the LC MEMS implementation has the largest confidence bounds, the chance of enough faulty measurements entering the valid region is increased, and the receiver is unable to mitigate against higher quality threats.

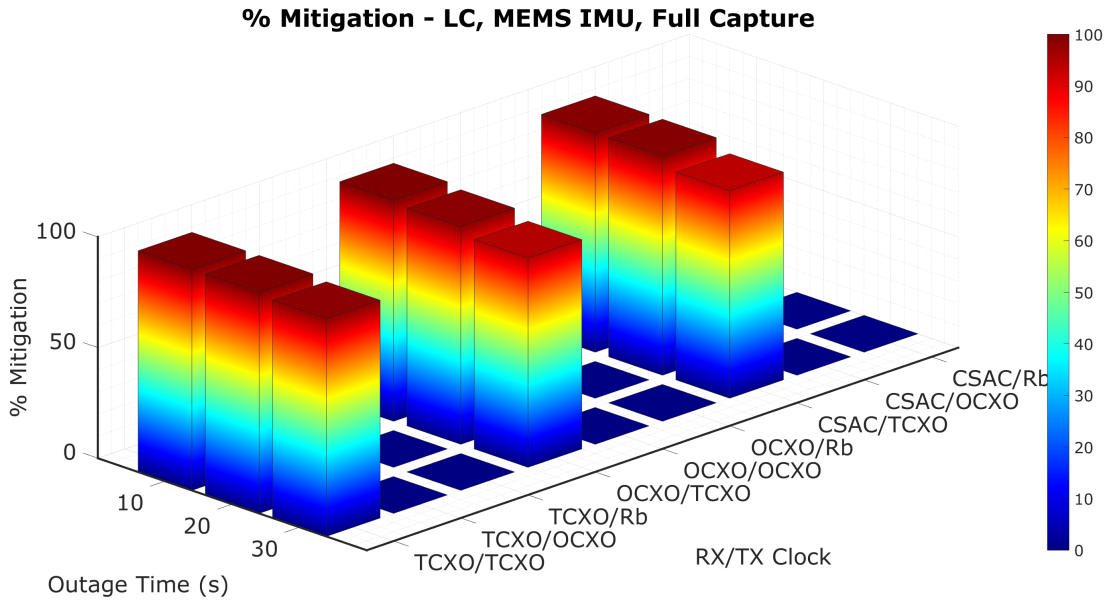

Figure 6.20: Mitigation Percentage - Full Capture, Loosely Coupled, MEMS IMU

When the receiver is paired with a tactical grade IMU, the mitigation percentage was more consistent across outage times however did not mitigate as high of a percentage as the MEMS implementation. The difference in mitigation percentage is rather low, however, with about a 5% difference in performance at the shortest outage time. The results shown in Figure 6.21

show that the mitigation percentage remains at about 95% across the outage times shown here, whereas the MEMS mitigation percentage begins to decrease at longer outage times. This is expected as the confidence bounds with the tactical implementation grow less rapidly than the MEMS implementation, allowing faults to be detected at longer periods of time. When the transmitter is driven by a higher quality oscillator, the algorithm is again unable to mitigate the effects of the capture but is still able to detect that faults are present and the solution is untrustworthy.

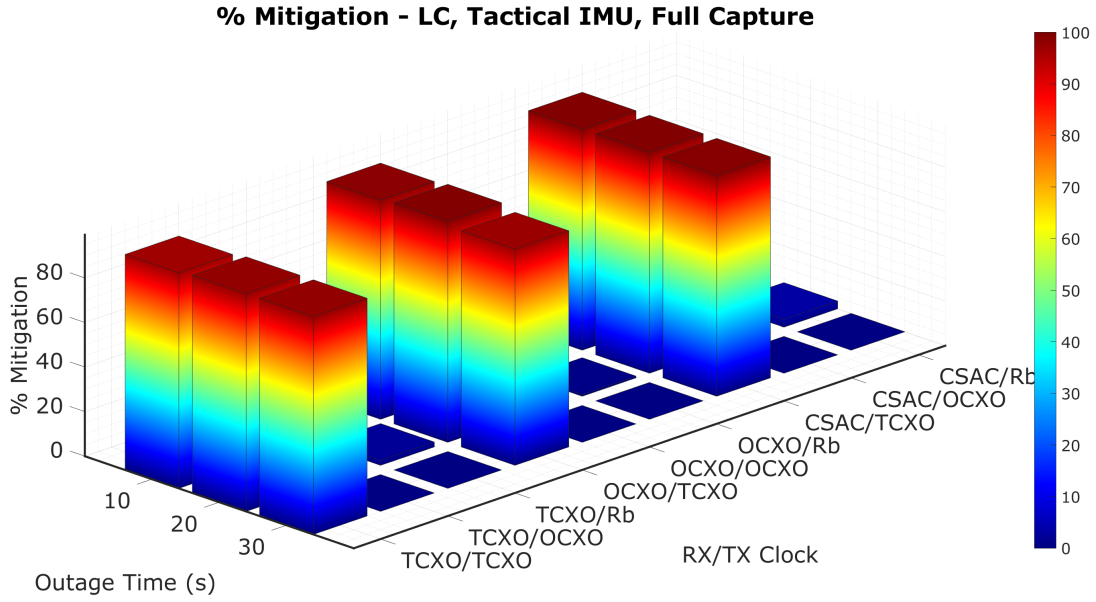

Figure 6.21: Mitigation Percentage - Full Capture, Loosely Coupled, Tactical IMU

The mitigation percentage of the TC MEMS implementation, shown in Figure 6.22, shows that the described filter configuration significantly improves upon the performance of the LC filter. The primary reason for this is the combination of the varying drift estimates due to position error along with tighter threshold bounds as a result of the TC filter. By reducing the filter bounds, the region that more than half of the measurements must fall into is decreased, which increases the mitigation performance of the algorithm as shown below. If the position error is decreased, the spread of the drift estimates (which can be seen in the single simulation results) would decrease and likely reduce the ability of the algorithm to detect the transmitters driven by a higher quality oscillator. As the position difference between the transmitter and receiver

decreases, the drift estimate becomes closer and closer to the combination of transmitter and receiver clock drift only.

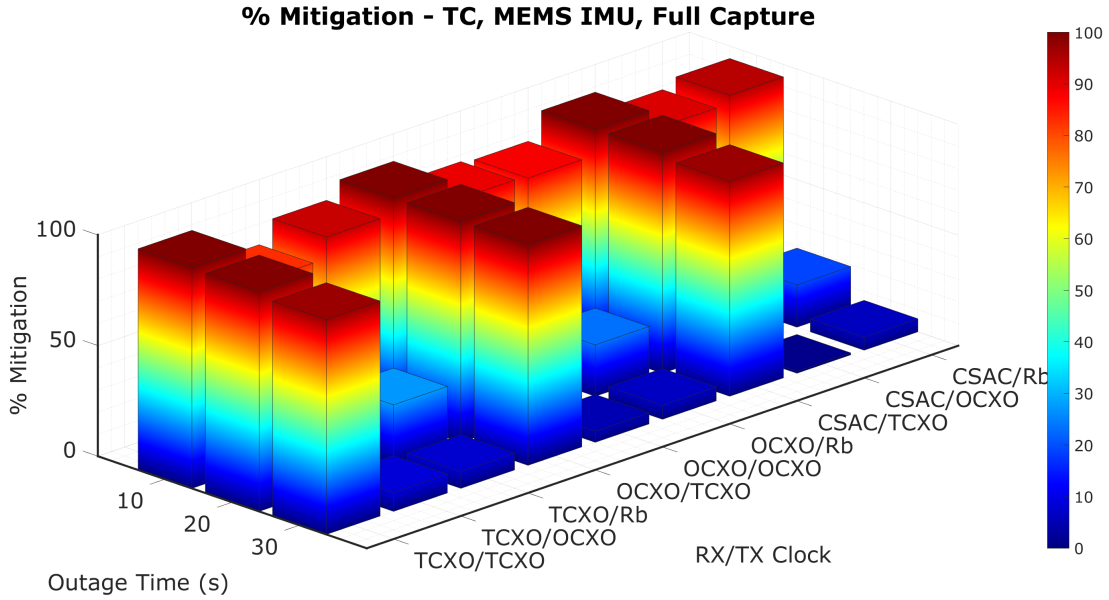

Figure 6.22: Mitigation Percentage - Full Capture, Tightly Coupled, MEMS IMU

The final iteration of the hardware tested was the tightly coupled implementation with a tactical IMU, which is expected to offer the best performance when compared to other configurations. Comparing the mitigation percentage for this scenario, shown in Figure 6.23, indicates that the performance of this combination does indeed outperform the other configurations shown above. As stated above discussing the TC MEMS mitigation percentage, the tighter drift bounds provide the ability to exclude channels because of the spread of the drift estimates as a result of the position difference between the meaconer and the receiver. This is improved upon by the TC tactical implementation, where the drift bounds grow more slowly than the MEMS implementation and the tight coupling provides good confidence for the clock drift estimates. As the outage time increases, the drift bounds grow as expected and allow more faults into the receiver, however at short outage times the mitigation performance of the algorithm is quite good when compared to the other implementations.

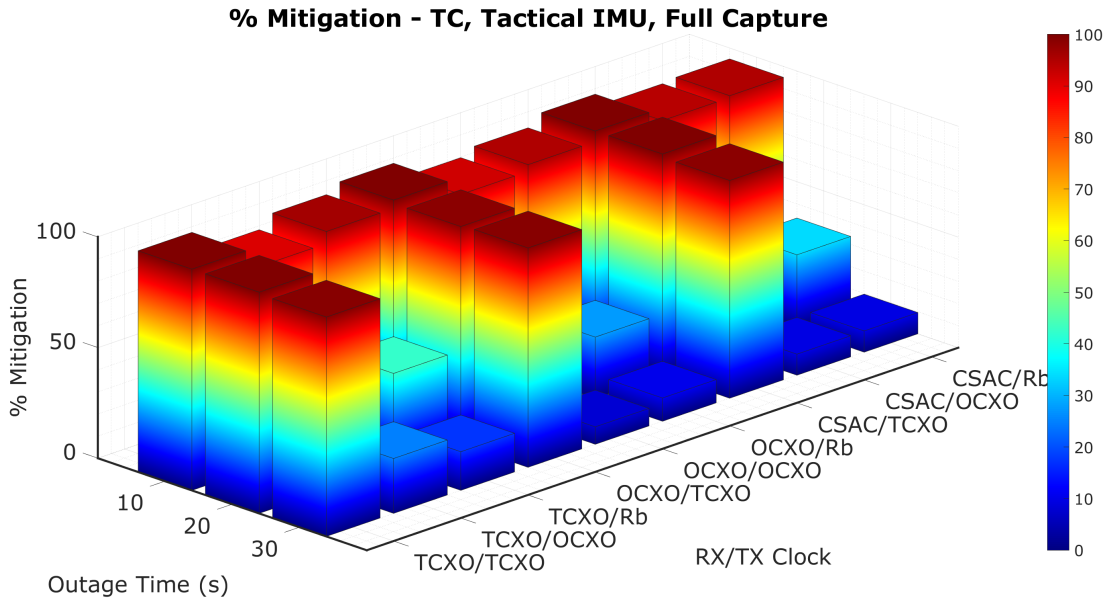

Figure 6.23: Mitigation Percentage - Full Capture, Tightly Coupled, Tactical IMU

This section shows that the proposed algorithm has the potential to detect faults that enter a GNSS receiver and coupled algorithm effectively as well as mitigate faults at various performance levels depending on the algorithm implementation. Some slight variation in the mitigation percentage was observed due to the relatively low number of scenarios that were simulated (250 iterations per configuration) and a larger simulation would likely show more consistent results across the board.

6.1.3.2 Partial Capture

Simulation of a partial capture event reveals the weaknesses in the algorithm as currently implemented. Described in Chapter 5, when half or more channels on a single frequency band are flagged as faulty, the entire band is dropped from the measurement update. For the partial capture simulation, one less than half the channels on the specified frequency were captured by the meaconer to intentionally avoid the criteria described above, making the algorithm have to individually detect and remove each channel with faults. In these tests, if even a single fault enters the receiver, it degrades the position solution and is classified as a failure to mitigate the interference. Results are shown for the same four base cases, with no FPR results shown as again the algorithm did not indicate any faults during clear sky scenarios.

The mitigation percentage for the partial outage using the LC MEMS implementation is significantly decreased when compared the full capture scenarios. Shown in Figure 6.24, the peak mitigation percentage was just over 50%, with the percentages decreasing as the outage length increased. Some fluctuation is observed in the percentages, with the TCXO receiver case increasing slightly before decreasing at the longer outage time. This is due to the randomized initial clock states, with some scenarios having much larger drift values than others which mmake the meaconer significantly easier to detect. Mitigation of errors does not occur when the transmitter is driven by high quality oscillators as the spread of the drift estimates is no longer enough to detect the faults as it was with the full capture scenarios. This decrease in performance is due to the decision criteria to reject a full band of measurements, where previously half of the signals had to be marked valid to allow the faults to enter the receiver. As stated above, in these scenarios a single faulty measurement entering the receiver would degrade or disrupt the navigation solution, causing the iteration to not be marked as mitigated.

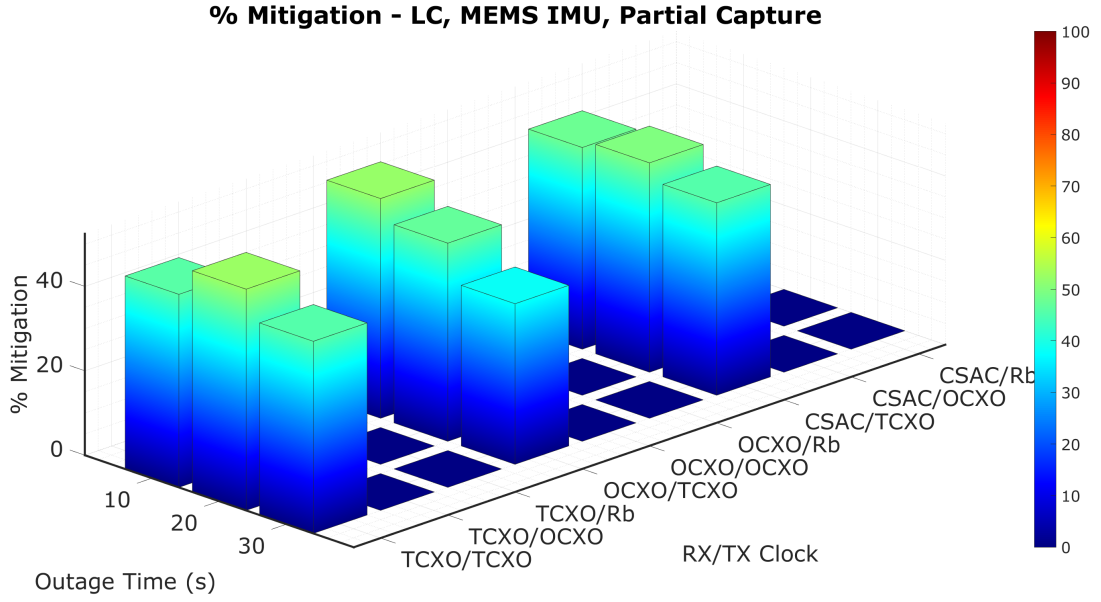

Figure 6.24: Mitigation Percentage - Partial Capture, Loosely Coupled, MEMS IMU

Compared to the MEMS implementation, the tactical implementation has marginally improved mitigation ability. The results, shown in Figure 6.25, show the highest mitigation percentage closer to 55% as opposed to 50% with the MEMS IMU. This difference is slight, however it is an improvement. There is again fluctuation in percentage due to the methods used

to simulate transmitter clock states, leading to some cases having less mitigation effectiveness than others.

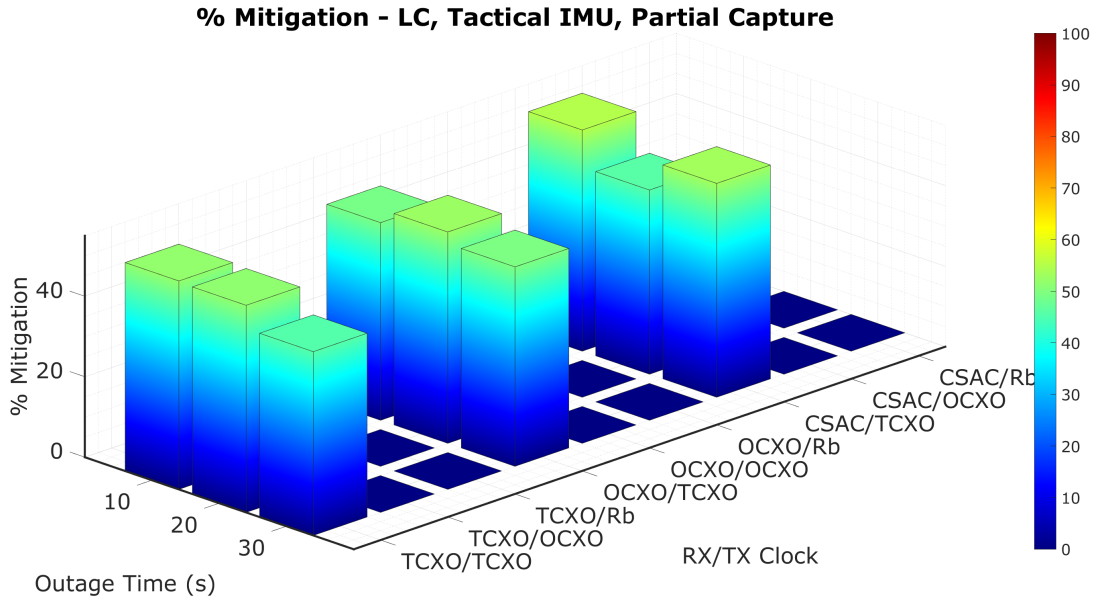

Figure 6.25: Mitigation Percentage - Partial Capture, Loosely Coupled, Tactical IMU

Pairing the MEMS IMU with the TC implementation does not offer many benefits over the LC implementation, with the mitigation percentage coming out about the same for both. The TC results in Figure 6.26 show that the mitigation percentage again lies around 50% with the amount decreasing as the outage duration increased. Some fluctuation is also visible in this scenario, with the same reasoning behind it - randomized initial clock states lead to some having large drift values and some having small drift values, with the small values being detected but not mitigated. This is also the case with the high quality transmitter clocks, where the drift is not far enough outside of the bounds and there are not enough faults to reject the entire frequency band.

The tightly coupled tactical IMU results are shown in Figure 6.27 below. This final scenario shows results similar to that of the LC tactical implementation, with the mitigation percentages in the mid-50%, a few points higher than the MEMS implementations. The mitigation percentage remains more constant across longer outage times instead of dropping due to increased confidence mounds as was seen with the MEMS implementations as well. The tighter

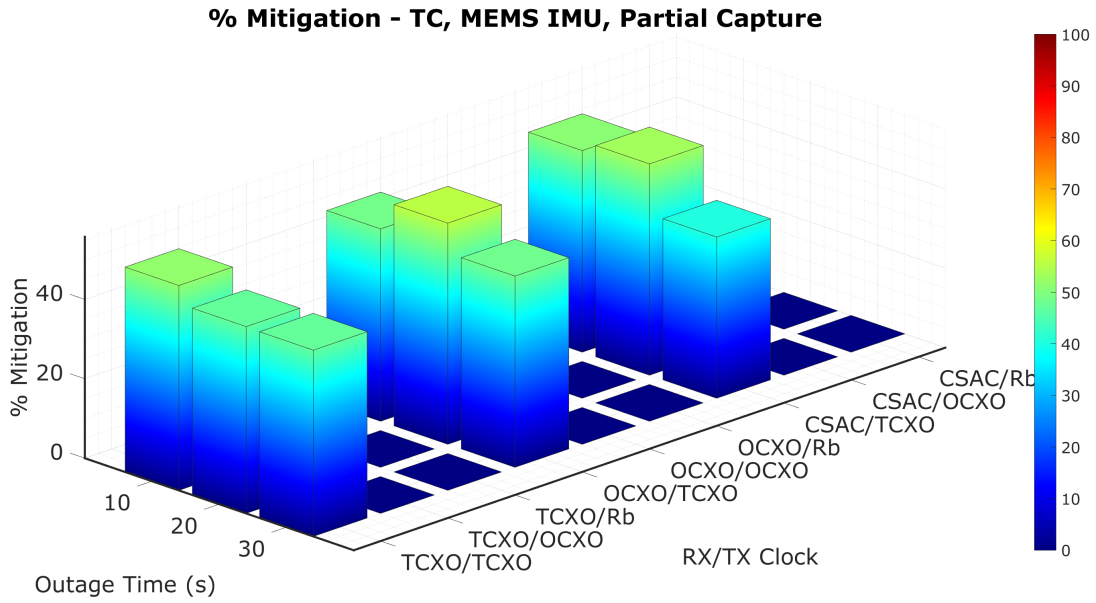

Figure 6.26: Mitigation Percentage - Partial Capture, Tightly Coupled, MEMS IMU

confidence bounds lead to the rejection of measurements that may have been on the threshold in the MEMS scenario, as discussed before.

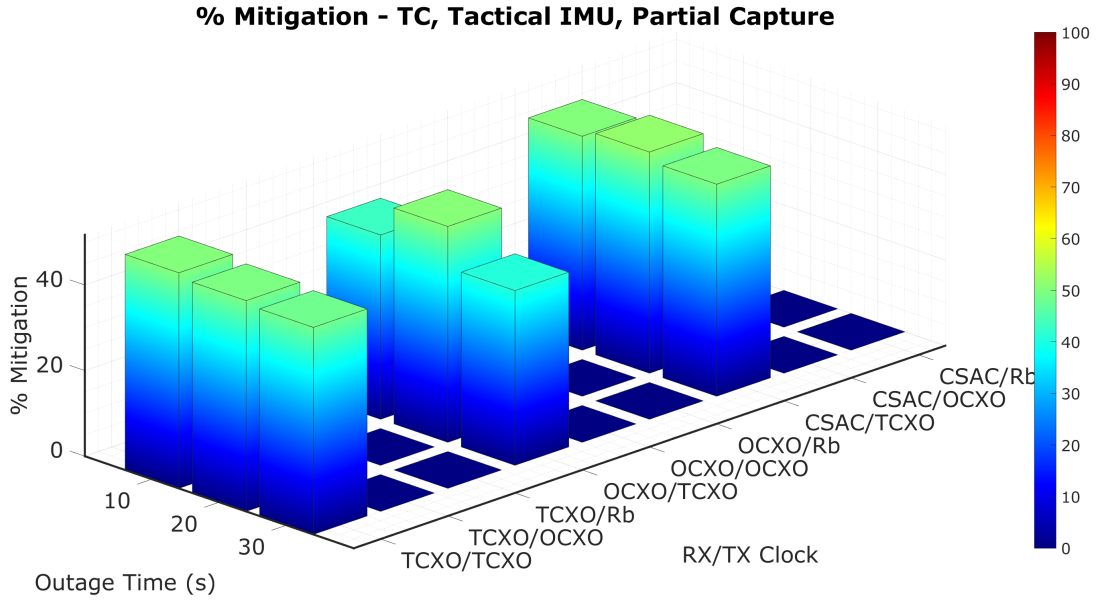

Figure 6.27: Mitigation Percentage - Full Capture, Tightly Coupled, Tactical IMU

The results shown in this section depict a more realistic scenario, where a receiver only locks on to several channels from a meaconer and the remainder continue to track the live sky signals. This scenario proves to be significantly more difficult to mitigate than the full

capture scenario, and transmitters with high quality oscillators go largely unmitigated because of the low to negligible clock drift errors introduced by the transmitter. The algorithm can fairly consistently mitigate partial outages induced by transmitters with low quality oscillators about half the time, which is much lower than the full outage but still effective. The detection performance shown in the TPR plots show that even if the faults can not be mitigated, the algorithm is able to detect a meaconer capturing the receiver based on the errors introduced into the clock drift estimates from the inauthentic position fix.

6.1.4 Spoofing Results - Single Tests

As opposed to a meaconer, which simply records and retransmits GNSS signals, spoofers have the ability to generate various desired navigation solutions depending on the complexity of the implementation. In this thesis, a simple spoofer which generates a static position solution similar to that of a meaconer as well as a spoofer that replicates the true trajectory are analyzed. More complex scenarios, such as dragoff or push attacks are not analyzed here. The static position spoofer closely mimics the behavior of the meaconer, where the drift estimates contain both position and drift errors from the transmitter. The dynamic spoofed trajectory mimics the trajectory of the receiver exactly, which removes the position error from the drift estimate, leaving only the difference in clock drifts. This final scenarios becomes extremely hard, if not impossible, to detect using the proposed method when the transmitter is driven by a high stability clock. The performance of the algorithm in each of these scenarios are analyzed in the following sections. As opposed to the meaconer tests, the spoofer scenarios are analyzed with a full capture event only to limit the number of test cases. For all the cases shown in this section, a signal outage occured from $t = 120s$ to $t = 130s$, with the spoofer operating on the GPS band (L1/L2) starting at the end of the outage at $t = 130s$.

6.1.4.1 Static Position Spoofer

Much like the meaconer, a receiver captured by a static position spoofer will display a false position fix at the location generated by the spoofer, and the clock states will be relative to the spoofer clock rather than GPS time. In this scenario, the drift estimates calculated by

the differenced carrier phase measurements will again reflect errors based on the transmitter clock as well as the simulated satellite geometry and the spoofer location. As was seen in the meaconer case, the clock drift estimates had a larger variance between each channel as opposed to a constant offset between the estimated delta range and measured delta carrier, indicating that the received measurements are faulty. When these received measurements fall outside of the clock drift confidence bounds, they are again rejected as was the case with the meaconer testing.

Each test case was again conducted using a combination of loose and tight integration of the measurements, as well as the MEMS and tactical grade IMUs. The simulations used for these test cases here again used a transmitter driven by a low quality TCXO while the receiver was driven by a higher stability CSAC, allowing the transmitter drift errors to be better observed.

The first scenaro shown is the LC MEMS IMU scenario, shown in Figure 6.28 below. As expected, the estimated drift values have a large spread between the values and are all excluded from the measurement updates. This closely mimics the performance of the meaconer simulations as expected, where the position errors heavily influence the drift estimates. One interesting aspect that did not appear in the meaconer scenarios is the drift bounds do not decrease to the levels they were at prior to the measurement outage and spoofing. The primary difference between the spoofing and meaconing scenarios is the signals which remain after the outage, where the L2/E5a were still valid in the meaconing scenario and E1/E5a remain available in the spoofing outage. Using only Galileo, only five satellites are available on two frequencies compared to twelve satellites available on L2/E5a for the meaconing scenarios. The reduced number of satellites that appear in the spoofing scenario potentially leads to the decreased confidence in the GNSS position update which is used for the coupled filter update.

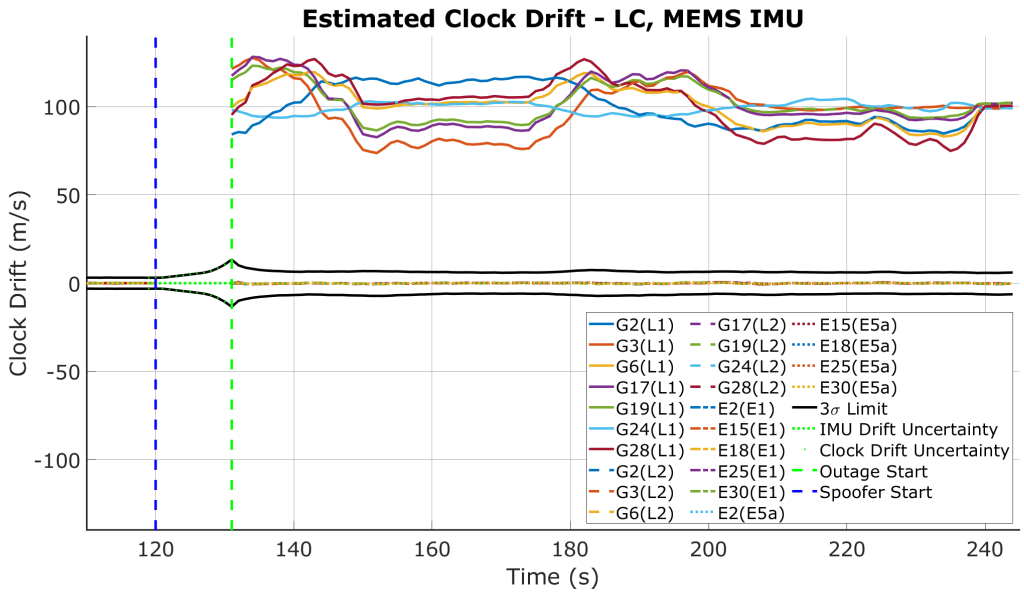

Figure 6.28: Simulation Result - LC, MEMS IMU, Static Spoof Location

The scenario used for the tactical simulation had significantly lower clock drift rates, leading to different results than shown in the MEMS scenario. This scenario, shown in Figure 6.29, shows that the mitigation algorithm has the same weaknesses against a static spoofed position as it does against a meaconer. The results shown for this test indicate that the faulty measurements are again initially detected and are rejected throughout the run. Shortly after the spoofer starts, two signals cross into the confidence bounds but are still rejected because the rest of the channels on those frequencies are still marked invalid. If fewer channels were identified as faulty, the estimates crossing into the threshold would allow the navigation solution to be corrupted.

Using a tightly coupled filter to process the same data as used above provided improved results, as is expected when comparing a loosely and tightly coupled filter. The drift estimates are the same between the LC and TC MEMS implementations, with the largest difference being the confidence bounds of the TC implementation. Similar to the LC filter, the TC implementation drift bounds decrease after the outage ends, though still not to the level of before the outage. The TC filter successfully mitigates the interference and is able to provide a valid navigation solution throughout the event. This is mainly due to the large clock drift of the transmitter combined with the position error that is introduced into the drift estimation calculation. The

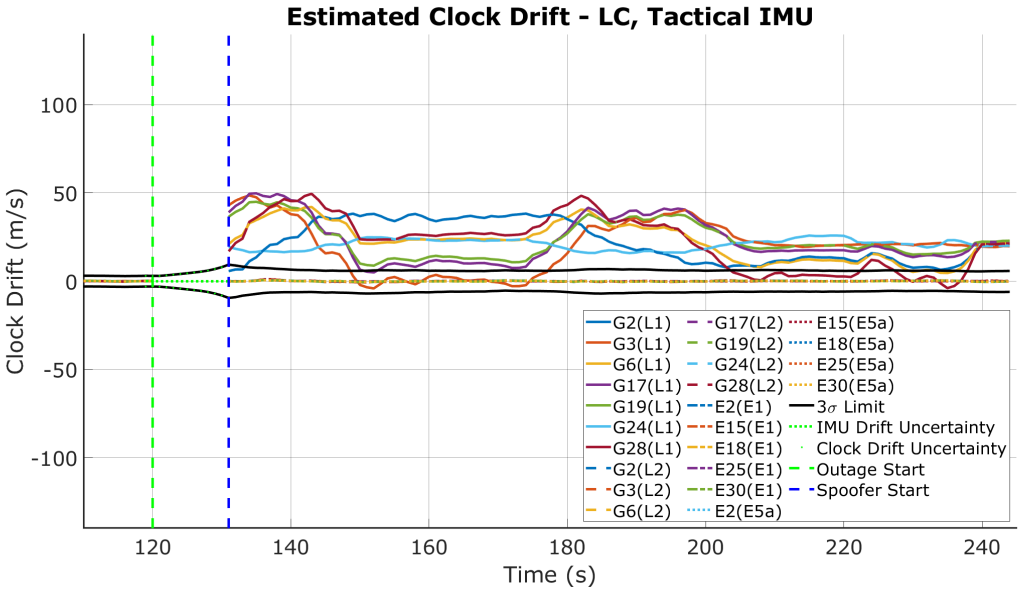

Figure 6.29: Simulation Result - LC, Tactical IMU, Static Spoof Location

larger the drift of the transmitter clock, the farther shifted away the drift estimates are, which is beneficial when being spoofed or meaconed to a static location where the position error causes fluctuations in the drift estimates as well.

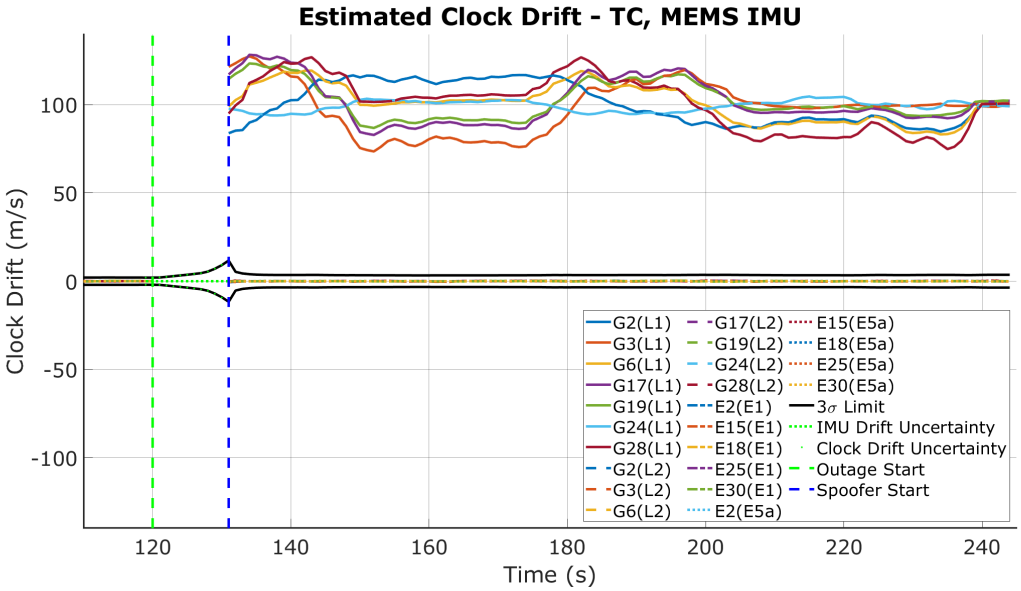

Figure 6.30: Simulation Result - TC, MEMS IMU, Static Spoof Location

The TC implementation using the tactical IMU proved to be more resilient than the LC implementation, as is shown in Figure 6.31 below. After the outage ended, the confidence bounds decreased close to the level that was present prior to the outage. Because of the decrease in the

confidence bounds, when a single faulty channel crosses into the boundary the measurements are still rejected as the rest of the measurements on the frequency are still indicated as faulty. In this implementation, only one faulty estimate crosses into the valid region throughout the entire simulation, whereas several faults crossed into the valid region in the LC implementation allowing the navigation to become corrupted. Since the measurements continue to be rejected throughout the spoofing event, the receiver is able to maintain a navigation solution, indicated by the drift estimates of the valid signals staying inside the fault bounds and lying along the truth value.

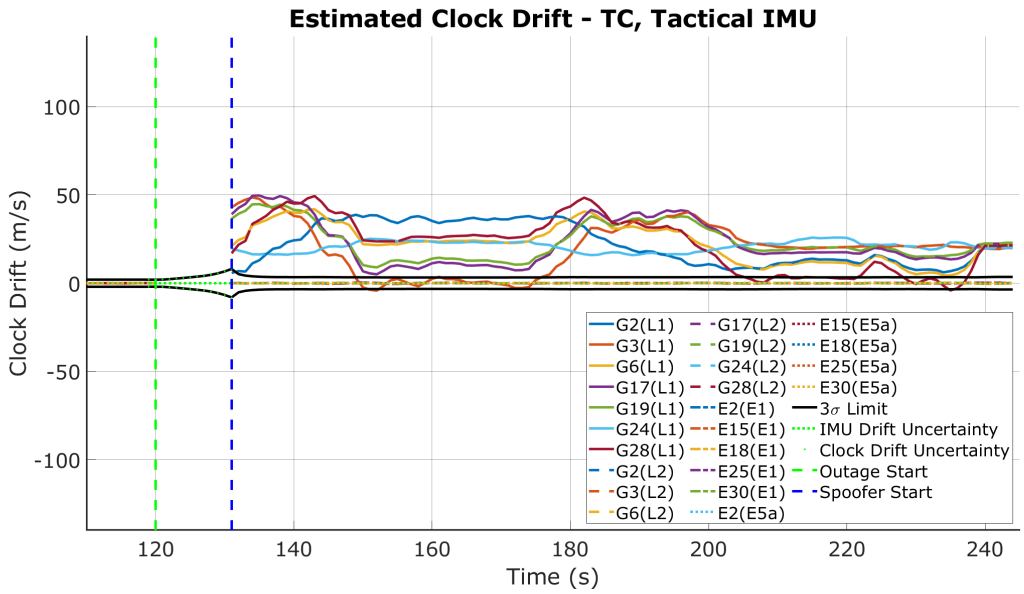

Figure 6.31: Simulation Result - TC, Tactical IMU, Static Spoof Location

A spoofer generating a static position offers the same type of results as a static meaconer does, which can be observed by comparing the results of the meaconer with the results of the static location spoof above. Both sets of results show that the clock drift estimates of the inauthentic signal depend not only on the clock drift of the transmitter, which determines the offset from the truth, but also the position error between the true navigation solution used to estimate the drift and the actual location that the meaconer or spoofer generates. When the receiver is captured and faulty GNSS measurements enter the coupled filter update, the state estimates explode due to the large change in position from where the INS output believes the location is and where the GNSS position believes it is. The large jump in estimates allow all

faults to be detected, as subsequent estimates of drift are heavily corrupted and lie outside of the propagated bounds. This method allows for the detection of nearly all faults involving a spoofed position, and allows for the mitigation of many of the received faults enabling the receiver to continue normal operations.

6.1.4.2 Dynamic Trajectory Spoofer

The most complex spoofing attack examined here involves a spoofed trajectory that matches the exact trajectory of the vehicle in question. By mimicking the exact trajectory with the spoofed measurements, the large deviation in clock drift estimates which has been shown for the meaconer and previous spoofer results no longer is visible. By spoofing the accurate location of the vehicle, the drift estimate no longer contains errors due to the difference in INS estimated position and the spoofed position. This change leaves only the difference in clock drift between the true receiver drift and the transmitter and receiver drift combined. These types of spoofers can be generated by an individual who may have direct access to the GNSS receiver antenna, such as a boat captain or truck driver who may want to make their monitoring system believe that they started in the same location but at a different time or the vehicle traveled a different route than was actually taken. The simulations shown here are conducted with the receiver using a synchronized CSAC, and the transmitter clock was simulated as an unsynchronzied TCXO using the same normal distribution described previously to assign initial bias and drift states. These results only show a very small subset of potential scenarios, with more combinations analyzed in the MC results section below.

The four tests detailed in this section follow the same process as the tests discussed previously. First, the loosely coupled integration is tested with a MEMS and tactical IMU, followed by the tightly coupled integration using the same scenarios analyzed in the first two tests. Again, the MEMS and tactical IMU data sets were simulated independently, leading to different clock drift values for the two tests. For each, an outage began at $t = 120s$ and ended when the spoofer became active at $t = 130s$.

The first test out of the four is shown in Figure 6.32, using a MEMS IMU and LC filter. As described above, the spoofer generated position matches exactly the true position of the

vehicle, leading to no spread across the estimated drift values as was seen when the spoofed position was different than the true user position. As a result of this, the clock drift estimate for the captured channels becomes the combination of the transmitter and receiver clock drift, with the receiver clock drift being negligible in this scenario. Since the transmitter clock drift from the TCXO is far greater than the confidence bounds at the end of the outage, leading to the rejection of all spoofed measurements. Since the spoofed position is the same as the true position, the estimated clock drift does not deviate as it did in the previously analyzed cases, leading to the successful mitigation of the spoofer for the duration of the simulation.

As was seen in the previous spoofer case, the receiver confidence bounds do not decrease to the pre-spoofing levels when the GPS satellites are removed from the measurement update. The spoofed signals here encompass the L1/L2 band, which is the same as the previously analyzed static spoofed location in which the same trends were observed for the confidence bounds. In the simulated scenarios, eight GPS satellites were available while only five Galileo satellites were available. When the GPS satellites were removed from the state update, less measurements were available and the confidence was not as good as it was when all satellites were available for measurement updates. This same trend is observed in the LC tactical simulation as well.

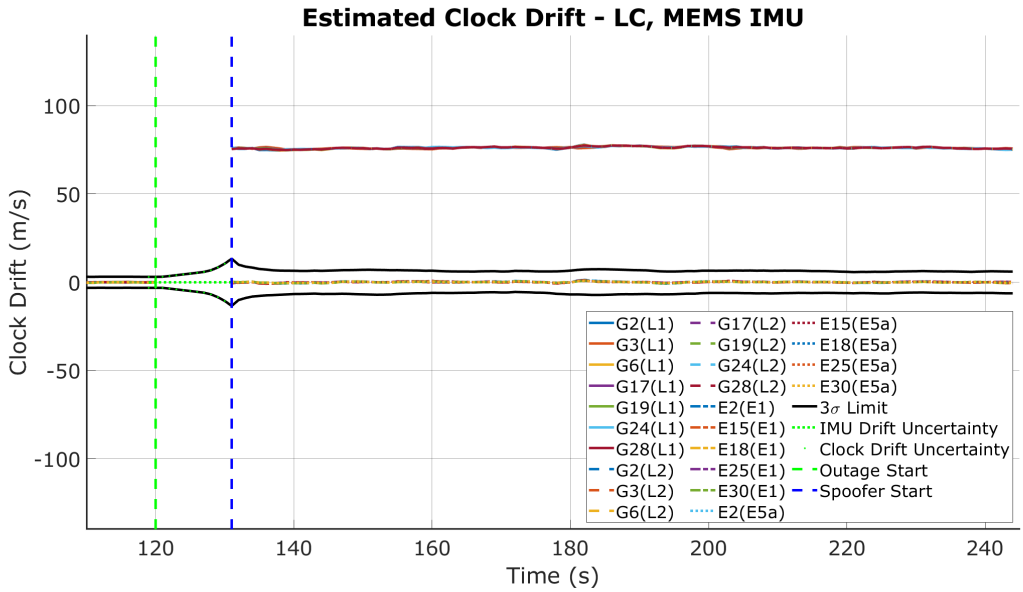

Figure 6.32: Simulation Result - LC, MEMS IMU, Spoofed Trajectory Matched to Truth

Figure 6.33 shows the results of the tactical IMU used with the LC integration. The transmitter clock drift again is significantly outside of the bounds at the end of the outage, allowing the receiver to identify and reject the faults and maintain a valid navigation solution in the presence of the spoofer. The increase in confidence bounds is also seen in this scenario, which has the possibility to be detrimental to the detection and mitigation performance if the transmitter drift was only just outside of the confidence bounds at the end of the outage then entered the bounds when they expanded due to the reduced number of measurements. This performance could be improved by incorporating additional GNSS constellations or other augmentation systems into the filter.

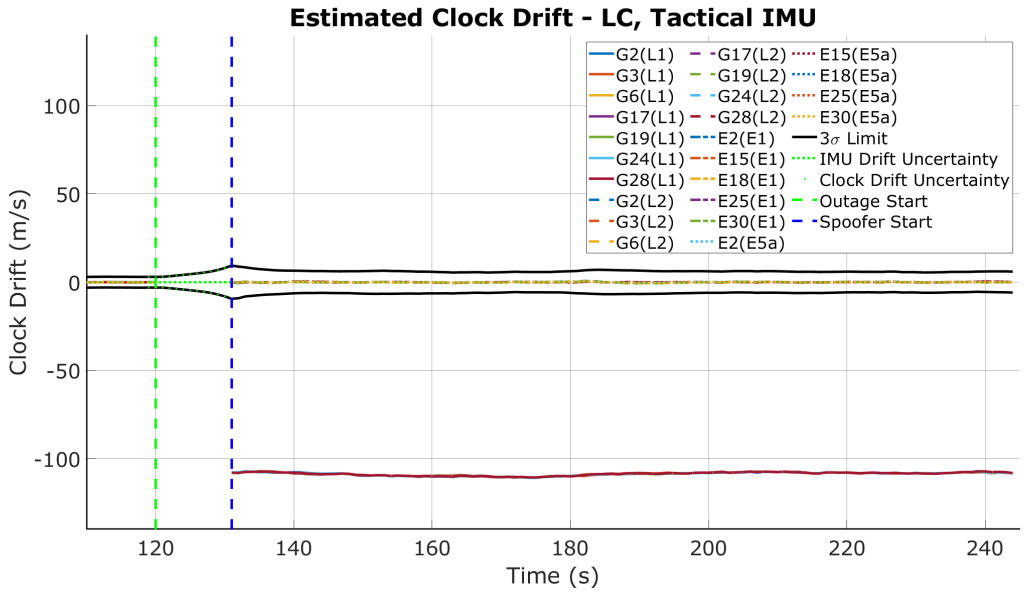

Figure 6.33: Simulation Result - LC, Tactical IMU, Spoofed Trajectory Matched to Truth

The first of the tightly coupled tests is shown in Figure 6.34, which again used a MEMS grade IMU. The transmitter clock drift in this simulation is the same as in the LC MEMS simulation, with the clock drift being just under $80m/s$. Since the receiver clock is assumed to be a high quality CSAC and is synchronized, the drift of the receiver clock is zero mean and does not age over the course of the small window of observation. This again allows the receiver to identify the signals with large clock drift values and remove those faults from the state update, ensuring an error free navigation solution during the spoofing event. For the TC implementation, the drift bounds decrease after the outage ends but do not reach the same level

as they were prior to the outage. The decrease is due to the implementation of the TC algorithm, where the INS measurements are directly coupled to the estimated range and range rate values. By directly coupling the measurements to update the state estimates, the covariance matrix is directly derived from the Kalman update instead of incorporating a LS estimate of confidence from the GNSS update. The TC Kalman update allows for prior knowledge to be incorporated into the uncertainty, allowing the bounds to be improved over those of the LC implementation.

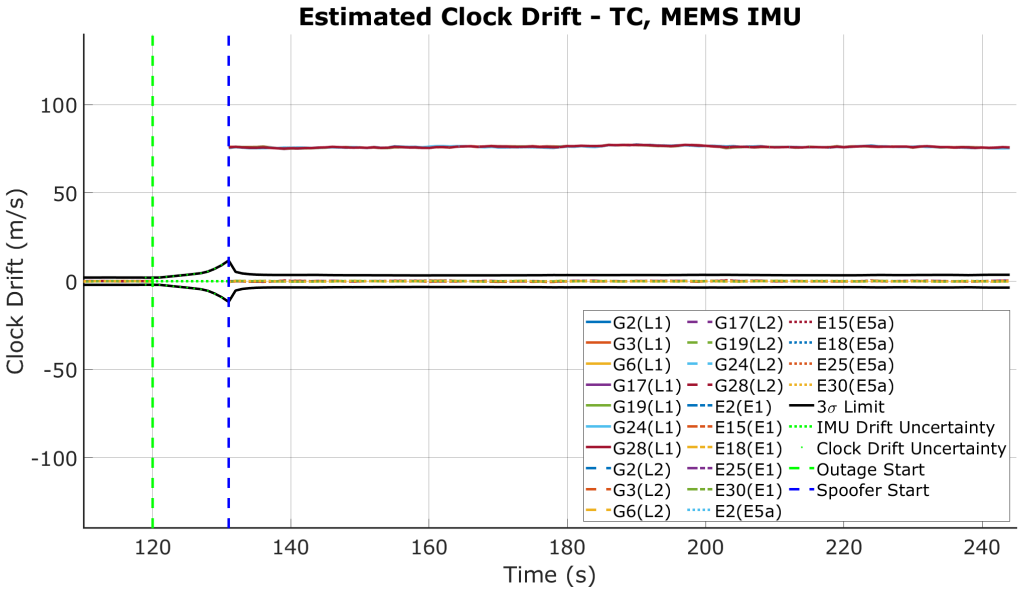

Figure 6.34: Simulation Result - TC, MEMS IMU, Spoofed Trajectory Matched to Truth

The final individual test that was conducted was the TC implementation with a tactical IMU, shown in Figure 6.35 below. Since the transmitter clock drift is the same as that shown in the LC implementation, it comes as no surprise that the TC implementation is able to detect and reject the spoofed measurements for the duration of the spoofing event. The confidence bounds shown here are tighter both before and after the outage, which is expected for the highest quality IMU tested with the higher performing integration filter. Once again, the confidence bounds do not decrease back to the level which they were at prior to the outage due to the reduced number of satellites that are used for the measurement updates. The results shown here again place the TC integration with tactical grade IMU as the top performer for the proposed algorithm, as was the case with the meaconer and static position spoofer tests as well.

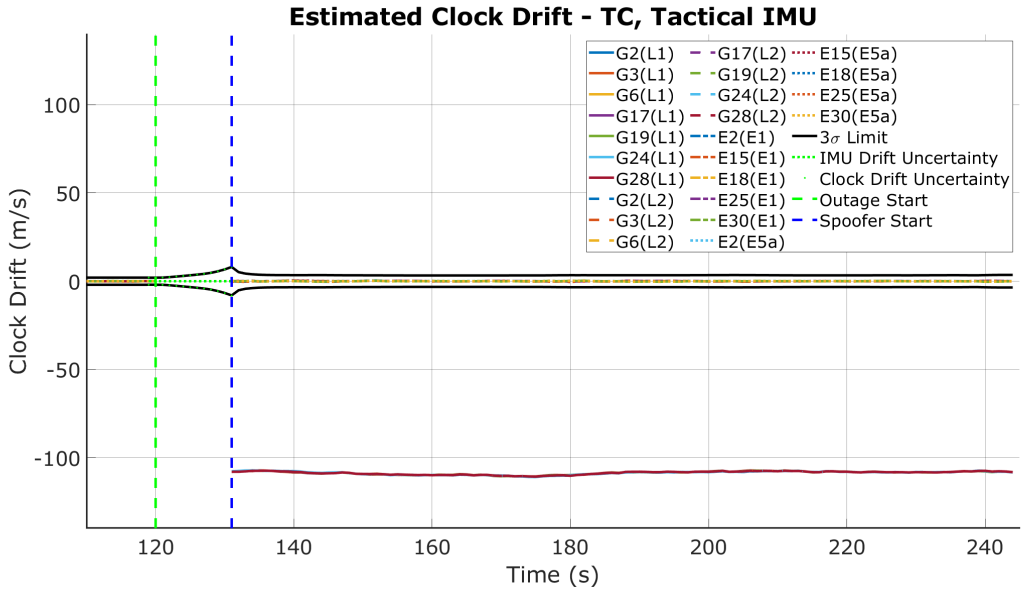

Figure 6.35: Simulation Result - TC, Tactical IMU, Spoofed Trajectory Matched to Truth

When a spoofer accurately replicates the position of the receiver, the only observable difference is the transmitter clock drift, which is shown in the results above. For a TCXO which has been on for any period of time while running freely, this difference will be detectable, allowing the receiver to detect and remove the faults. The scenarios shown here are only a single example, however, with numerous other transmitter clock possibilities available. If a transmitter clock has negligible drift, such as a Rb or Cs standard, or is GPS disciplined, the interference will likely go undetected. To analyze the possible combinations and initial frequency errors, a Monte Carlo simulation was conducted, and the results are discussed in the following section.

6.1.5 Spoofing Results - Monte Carlo

The Monte Carlo simulation conducted for the spoofer took the same approach as the the meaconer simulation discussed earlier in this chapter. The same random distributions were used to initialize the transmitter clock states, allowing more variables to be analyzed when examining the performance of the detection and mitigation algorithm. The results for both the static position spoofer and the dynamic trajectory spoofer are discussed presently. Plots of TPR and tabulated mitigation percentage results are provided in Appendix B, with the TPR also included for the dynamic position spoof in the section below.

6.1.5.1 Static Position Spoofer

The static position spoofer, as has been discussed, is similar to the meaconer in that the position solution calculated when using the false measurements is a static point which is corrupted by the clock errors of the transmitter as well as the receiver. Because of the similarities between the meaconer and the static position spoofer, the results from the Monte Carlo simulations for the static spoofer show the same trends as those from the meaconer.

Results from the LC MEMS test are shown in Figure 6.36. The mitigation percentages show results that closely replicate the meaconer LC MEMS results, with very good performance when the transmitter is driven by a low quality oscillator and poor mitigation performance when the transmitter is driven by a high quality oscillator. As mentioned previously, the clock drift introduced from high quality oscillators is negligible, leading to the primary errors to be from the difference in satellite geometry between the spoofed position and the true position. The drift estimates from this frequently enter the valid region, causing the algorithm to mark them as valid and allow the measurements to be used to update the states, disrupting the navigation solution.

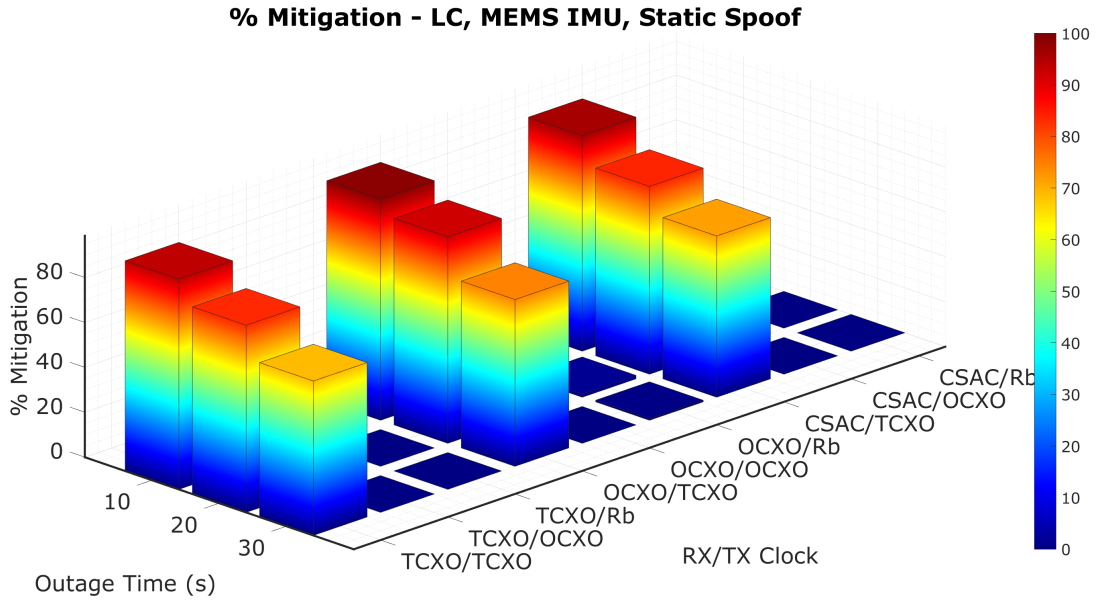

Figure 6.36: Mitigation Percentage - LC, MEMS IMU, Static Position Spoof

Moving to the tactical IMU LC implementation, the mitigation percentage given in Figure 6.37 shows good improvement over the MEMS implementation, with the percentage staying higher for longer due to the decrease in growth of the fault bounds. Additionally, there was a small improvement in mitigation percentage at the shortest time steps when using a tactical IMU, with 100% of faults being mitigated at the shortest time step by when a low quality transmitter was used. The decreased fault bounds also allow for the increase in detection of transmitters with higher quality oscillators. This is again due to the spread of the drift estimates from spoofed measurements, which was shown and discussed in the meaconer section as well as with the individual results above. In this scenario, the receiver is able to successfully mitigate the effects of transmitters with higher grade clocks at short outage times. This is likely due to the spreading of the drift estimates combined with the motion of the receiver relative to the transmitter, causing the drift estimates to exceed the detection threshold.

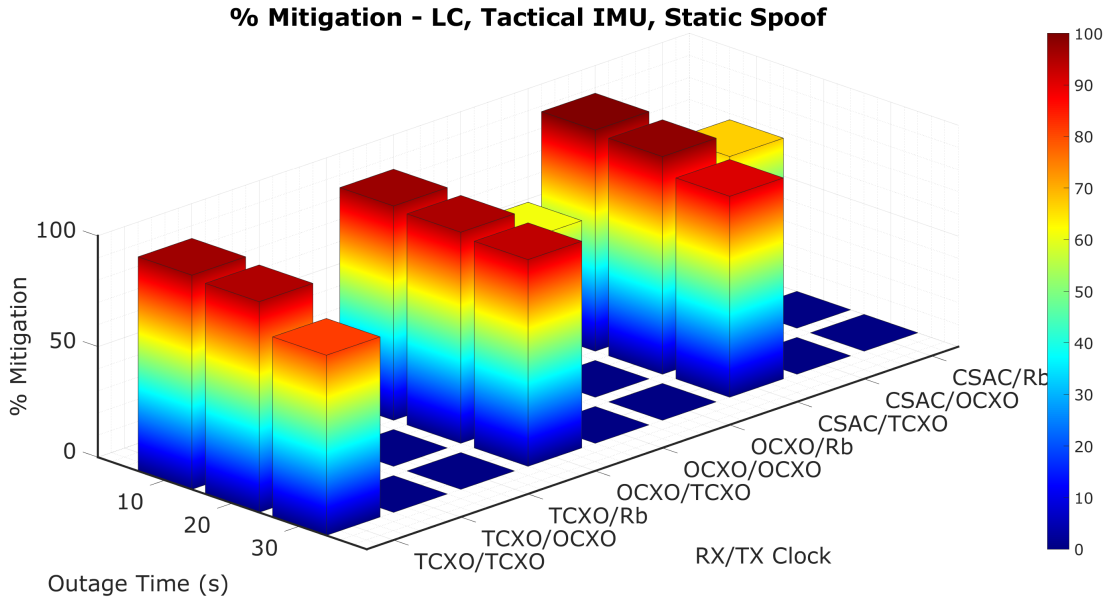

Figure 6.37: Mitigation Percentage - LC, Tactical IMU, Static Position Spoof

The tightly coupled implementation offers some slight improvement over the loosely coupled implementation, however the improvement in mitigation percentage is marginal at best in this case. The mitigation percentage shown in Figure 6.38 does not offer significant improvement when compared to the LC implementation. Spoofers driven by a low quality oscillator are readily detected at short outage times, and the percentage drops off as the outage time increases

due to the growth of the error bounds. This performance is very similar to that shown in the LC spoofing results above. When a high quality transmitter clock is used, the algorithm is again unable to successfully detect the spoofer.

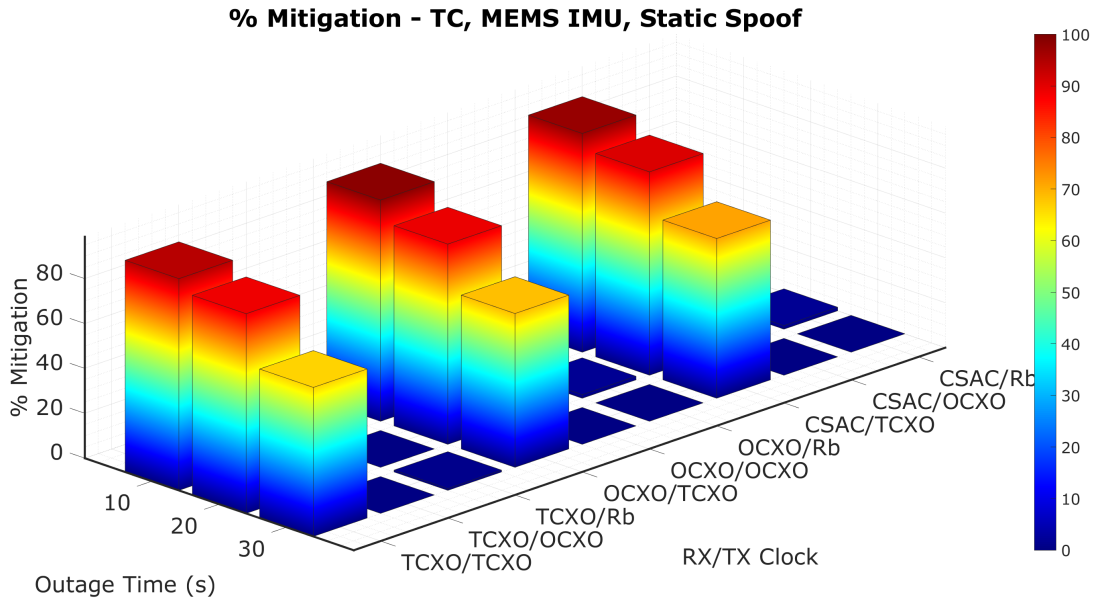

Figure 6.38: Mitigation Percentage - TC, MEMS IMU, Static Position Spoof

In previous scenarios, the tightly coupled tactical IMU combination showed the best performance with regards to mitigation ability. The mitigation percentage for this combination is shown in Figure 6.39 below. The mitigation percentage closely matches that of the loosely coupled implementation, with the percentages differing by only about 1%. As expected, the tactical IMU allows the algorithm to survive and mitigate more events at longer outage durations as a result of the slower growth of uncertainty bounds. The mitigation percentage at an outage time of $30s$ is somewhat lower than the LC tactical implementation, with the LC implementation staying around 90% and the TC implementation dropping to around 85%. The slight difference may be chalked up to the random initialization of clock states for each simulation, which may have given a slight edge to the LC test that was conducted. Running longer MC simulations would provide additional information on this difference.

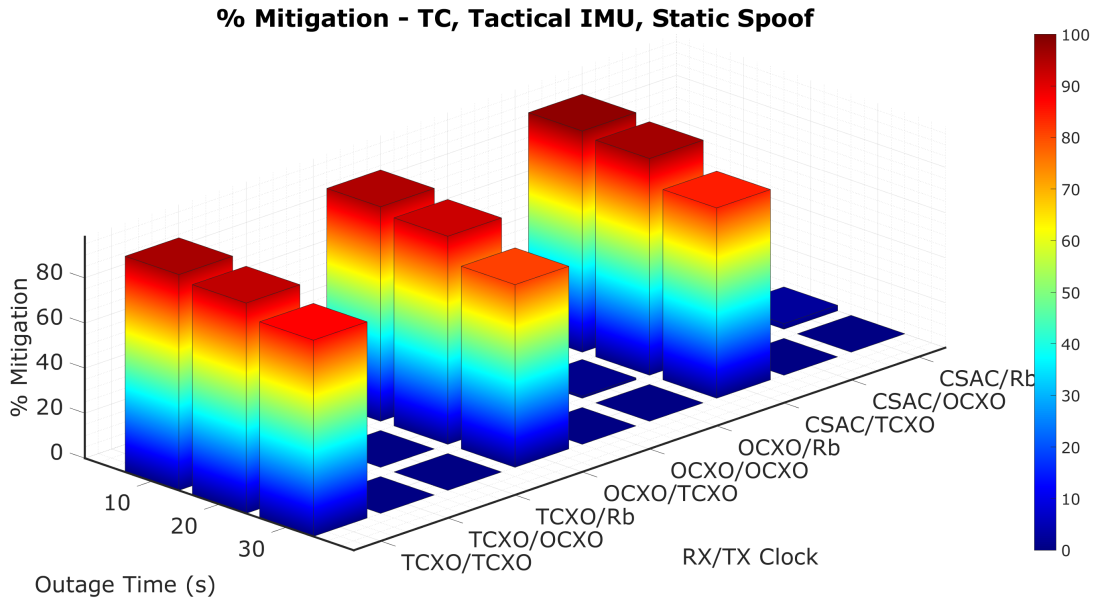

Figure 6.39: Mitigation Percentage - TC, Tactical IMU, Static Position Spoof

The results discussed in this section show that the static spoof location simulations perform similarly to those of the meaconer. This result is expected as both the meaconer and the spoofer generate a false position solution, with the meaconer generating the solution of the receive antenna and the spoofer generating a position anywhere in the world. In some scenarios, the spoofer may be easier to detect depending on the difference in position between the true and spoofed location, whereas the meaconer is limited in range as it must be able to rebroadcast signals to directly corrupt the receiver. The proposed algorithm was able to successfully mitigate nearly all faults when the transmitter was driven with a low quality oscillator like those which come standard in most COTS SDR platforms. The algorithm performance was degraded significantly when higher quality oscillators were used, however as stated previously additional factors such as position error may also influence the resulting mitigation performance.

6.1.5.2 Dynamic Trajectory Spoofer

The results from the individual test cases in Section 6.1.4.2 show that when the position error is eliminated from the drift estimation equation, the resulting estimate is that of either the receiver drift or the receiver and transmitter drift combined. As expected, for all combinations of integration method and IMU tested, the higher quality oscillators with negligible or low

clock drift went undetected, limiting the effectiveness of the proposed algorithm. For each scenario, the FPR was zero across the board, indicating that the detection algorithm did not identify spoofing when only valid signals were present. Since the FPR plots would show no additional information, these plots are omitted for these results.

The TPR for the LC MEMS case is shown in Figure 6.40. When the transmitter clock is a low quality TCXO, the mitigation algorithm is able to detect and remove the faults at varying outage lengths, not only the $10s$ outage shown in the individual tests. When the transmitter is driven by and OCXO or Rb oscillator, however, the added clock drift remains within the drift bounds and the receiver is captured. The receiver is captured and the spoofing is unable to be detected, as there is no noticable change in the clock drift due to the transmitter. Since there is no change and the position is spoofed to be the same as the true position, no change in drift occurs and the receiver continues to believe that only true values are present. Once the receiver is captured in this manner, other attacks could be conducted that may lead to the detection of the spoofing, however other detection methods are not examined in this thesis.

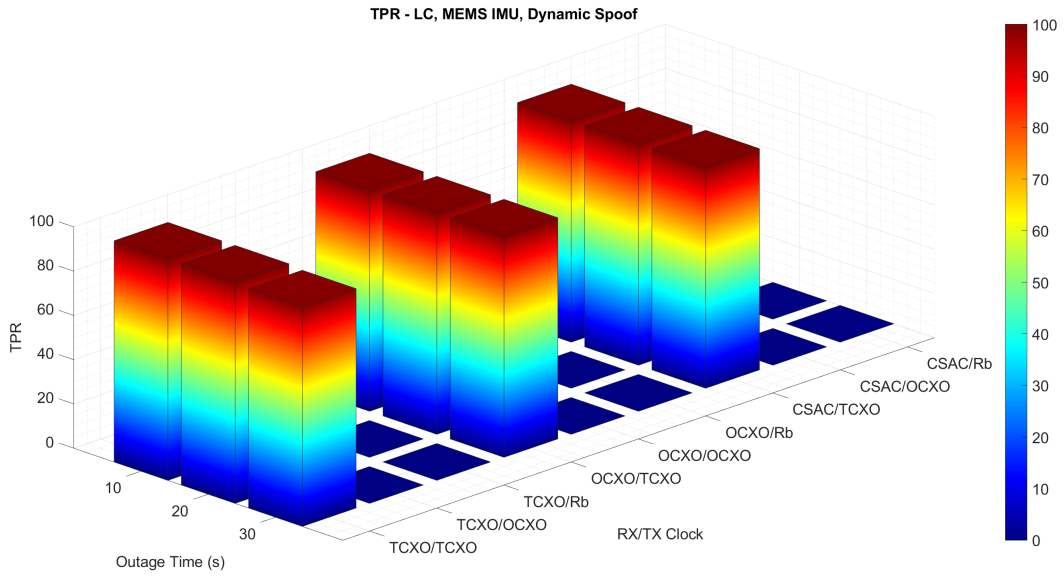

Figure 6.40: TPR - LC, MEMS IMU, Spoofed Trajectory Matched to Truth

Following the trend of the TPR shown above, the mitigation percentage shown in Figure 6.41 is zero for the OCXO and Rb transmitter clock scenarios. This is because no spoofing was ever detected, making the receiver unable to mitigate something that was never detected.

The mitigation percentage decreases as the duration of the outage increases as well, with the maximum mitigation percentage occurring at the shortest outage time. The TCXO, OCXO, and CSAC receivers all perform nearly equally with an outage time of $10s$, mitigating nearly 95% of the faults that were present. The results show that the TCXO receiver is able to mitigate more interference with an outage time of $20s$, however this may simply be a result of the randomized transmitter clock states and the relatively low number (250) of iterations conducted for each Monte Carlo. Overall, the results follow the expected trend of decreasing effectiveness at longer outage times, where the clock drift is more likely to fall within the drift bounds after a longer outage.

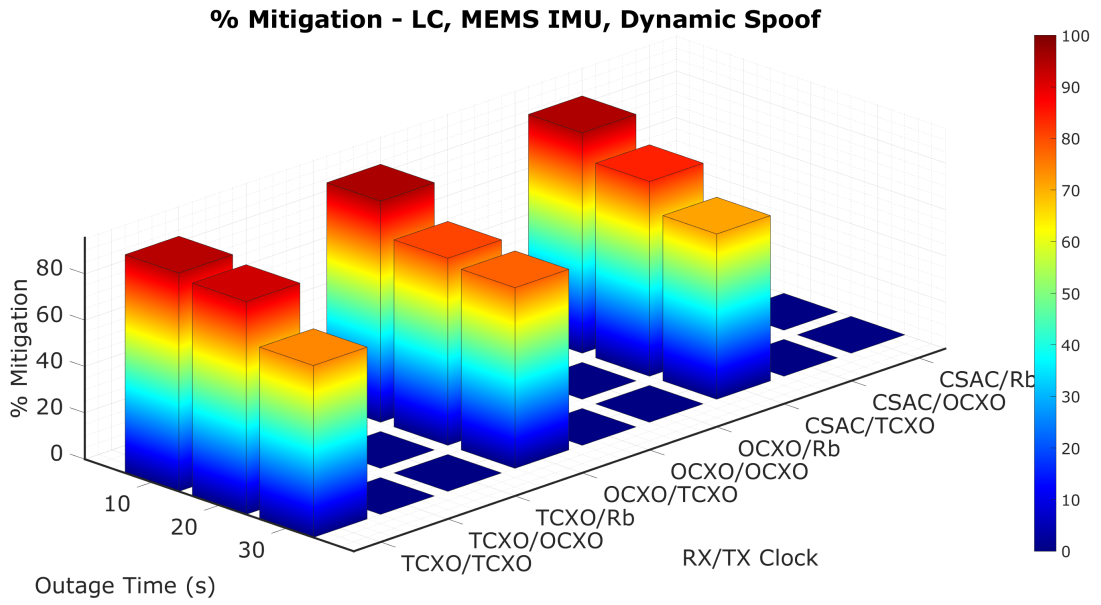

Figure 6.41: Mitigation Percentage - LC, MEMS IMU, Spoofed Trajectory Matched to Truth

The tactical IMU implementations performed just as well as the MEMS implementations in terms of TPR, with the transmitters driven by TCXOs being detected easily and the remainder of the transmitters going undetected. The results for this case are shown in Figure 6.42, and look identical to those shown in Figure 6.40. Even though the confidence bounds are improved when using a tactical IMU, if the transmitter clock drift is small enough as is the case when using a quality OCXO or atomic standard, the increase in drift is well within the expected bounds.

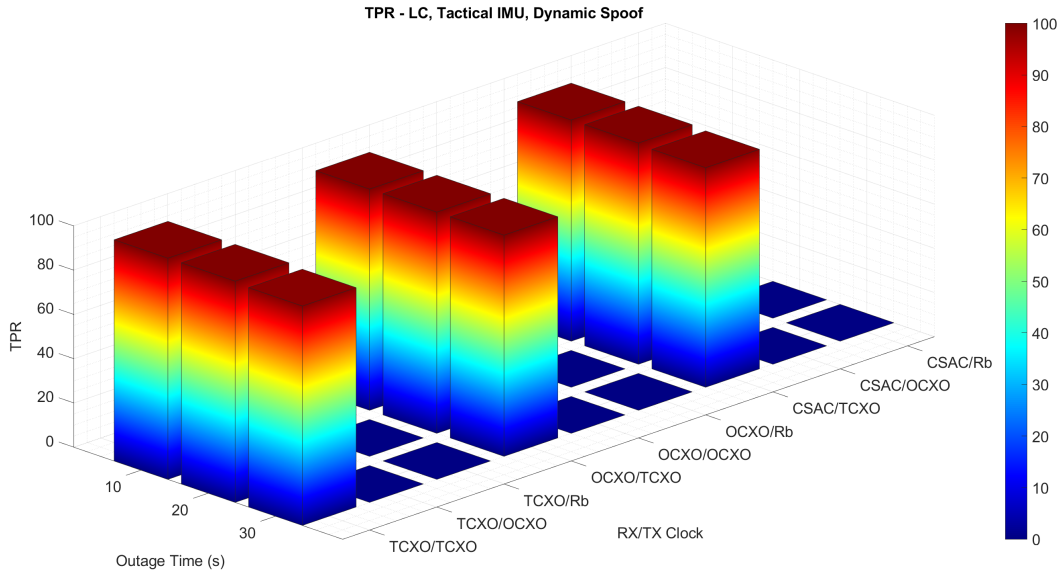

Figure 6.42: TPR - LC, Tactical IMU, Spoofed Trajectory Matched to Truth

The largest difference between the MEMS and tactical implementations comes from the mitigation percentage. The tactical implementation is able to mitigate the interference for longer periods of time, which has been shown in scenarios in prior sections. In Figure 6.43, not only does the mitigation percentage at the shortest outage length match that of the MEMS implementation, the mitigation percentage does not drop off as rapidly as it did previously. This is attributed to the less rapid growth of the error bounds with the tactical IMU, which would reach the same level as the MEMS error bounds at a later time, allowing more drift errors to be mitigated at longer outage times. If the simulations were conducted for longer outage times, the plot would begin to reflect the same trend as Figure 6.41 where the percentage decreased at longer outage times. One thing to note is that even with the higher quality IMU, the receiver is unable to mitigate all of the interference in any scenario. Since the transmitter clock states are randomized, some drift values may lie within the error bounds, allowing the receiver to be captured.

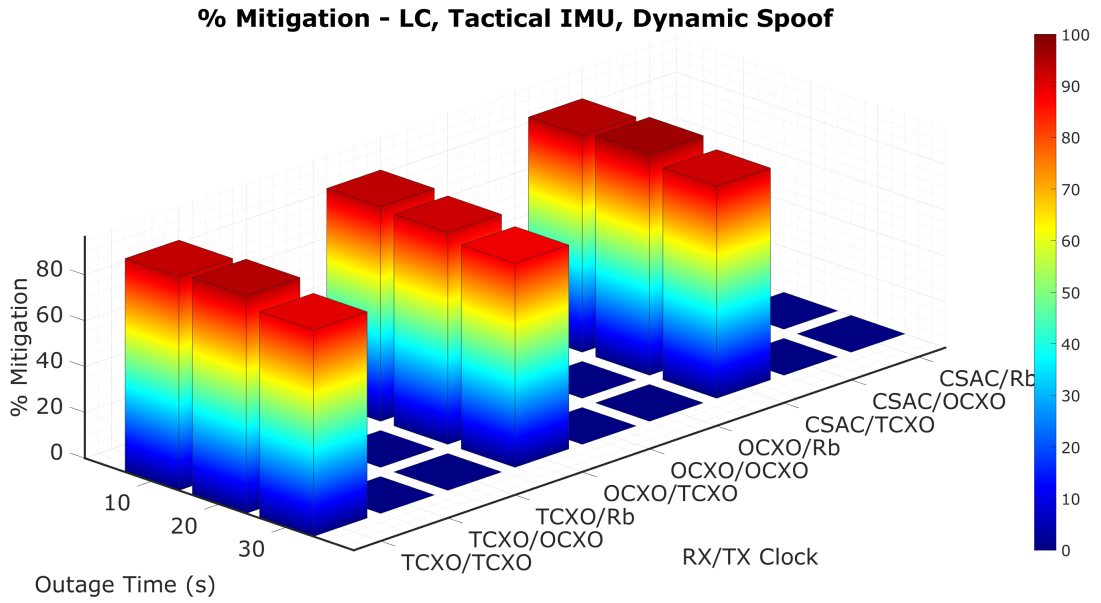

Figure 6.43: Mitigation Percentage - LC, Tactical IMU, Spoofed Trajectory Matched to Truth

The final spoofer simulation analysis was conducted with the tightly coupled implementation. Shown in Figure 6.44, the MEMS TC simulation has the same performance as the LC simulation, with all faults being detected for transmitters driven by low quality TCXOs such as those found in the HackRF or BladeRF transceivers. When the transmitter clock is improved to a mid-grade OCXO or an atomic standard, the algorithm again fails to detect any interference and is captured by the spoofer.

The mitigation percentage for the TC MEMS case shown in Figure 6.45 shows the same trends as the LC MEMS mitigation percentage, decreasing as the outage time increases. The TC implementation does not perform significantly better or worse than the LC implementation with the MEMS IMU, with performance being comparable across the board. The OCXO results show that the mitigation percentage was lower at an outage time of $10s$ than that of $20s$, likely due to the limited number of simulations that were run to generate the results. More simulations may show more conclusive results, however are time prohibitive to run for this work.

Pairing a tactical IMU with a tight integration filter again offered the same detection performance as the other algorithm implementations. Having a higher quality transmitter provides too small of a change in clock drift to raise an alarm at the receiver to reject measurements, and

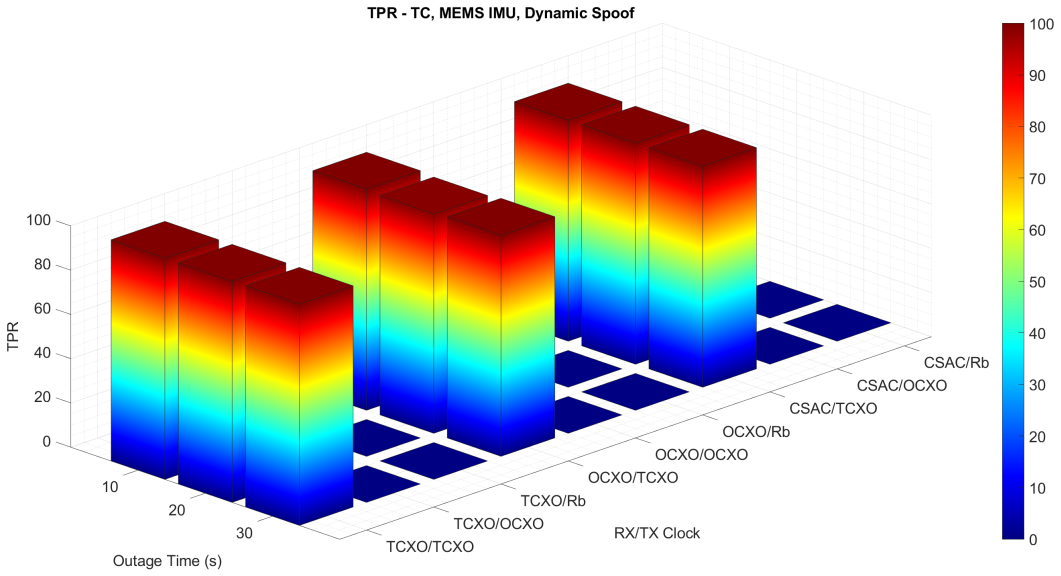

Figure 6.44: TPR - TC, MEMS IMU, Spoofed Trajectory Matched to Truth

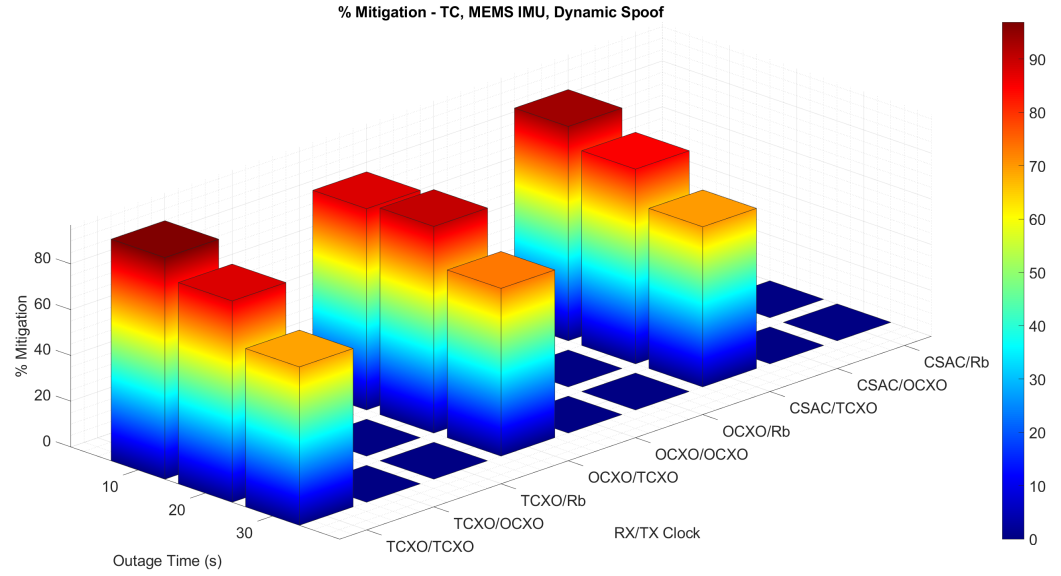

Figure 6.45: Mitigation Percentage - TC, MEMS IMU, Spoofed Trajectory Matched to Truth

since the trajectory does not change the drift estimates do not vary as was seen in the meaconer and static spoofer results. The TC tactical TPR results are shown in Figure 6.46 below.

The tactical IMU offers the same benefits as were discussed in the LC tactical scenario, with the receiver able to detect spoofing across longer outage times. This offers improvement over the MEMS implementation, but there is no appreciable difference between the LC and TC tactical implementations. For this scenario in particular, the benefits of the TC algorithm do not

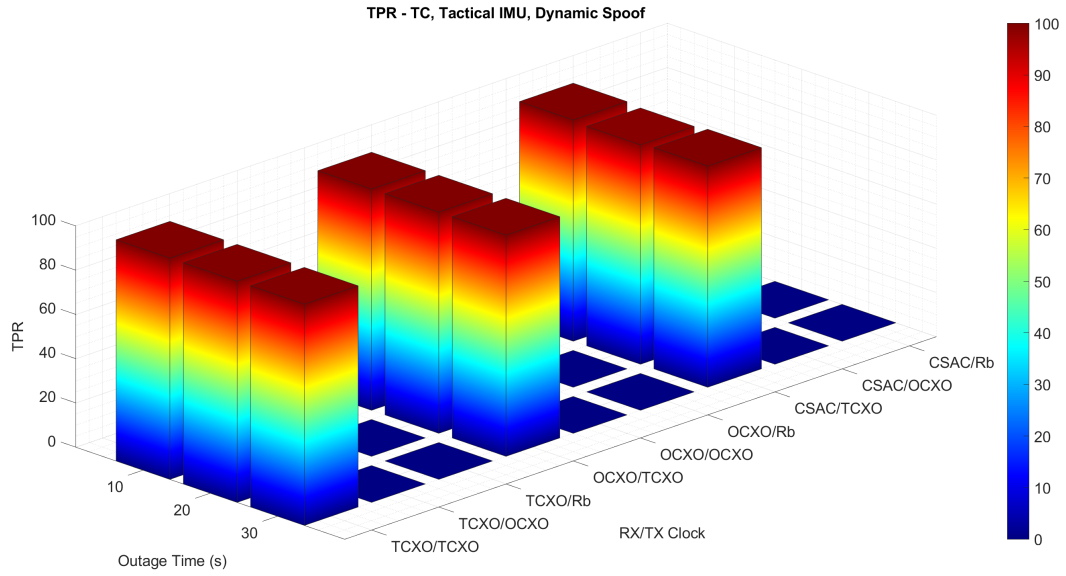

Figure 6.46: TPR - TC, Tactical IMU, Spoofed Trajectory Matched to Truth

shine over that of the LC implementation, however the benefits were observed in the meaconer and static spoofer scenarios. The mitigation percentages, shown in Figure 6.47, again show the decreasing performance as outage time increases, and the percentage is significantly improved at $t = 30s$ when compared to the MEMS implementations.

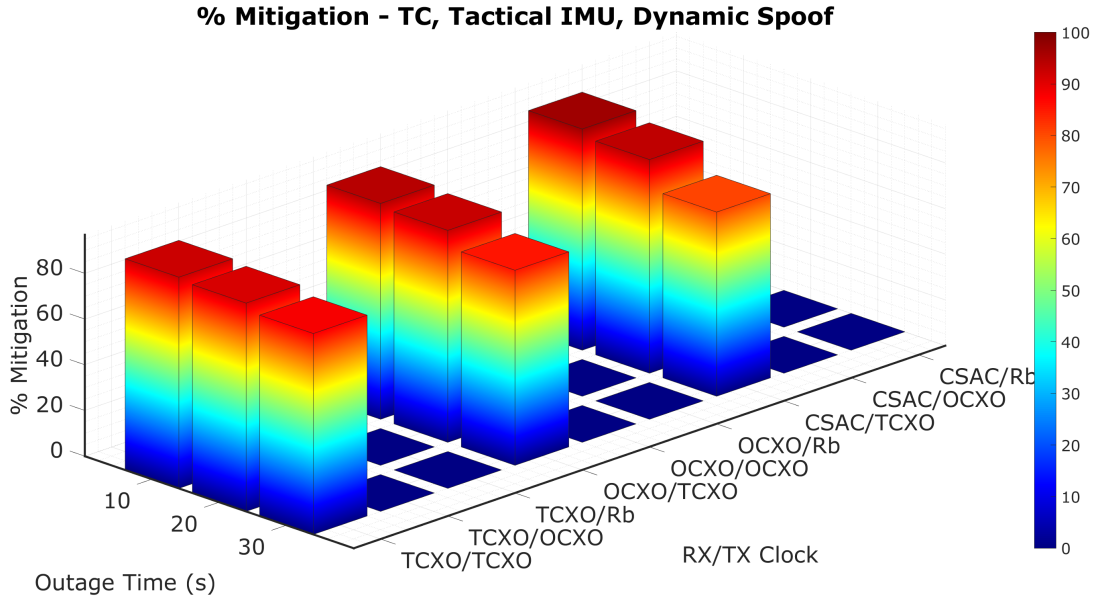

Figure 6.47: Mitigation Percentage - TC, Tactical IMU, Spoofed Trajectory Matched to Truth

The results shown in this section bring up several performance questions about the algorithm and the effectiveness against transmitters driven by high quality oscillators. When driven by a low quality clock such as a TCXO, the spoofing is readily detectable, however clocks with negligible drift can go unnoticed. Once the receiver is captured, an attempted time or position push may allow the spoofing to be detected, raising an alert and notifying the operatior not to trust the GNSS solution available. Comparing the LC and TC implementations, there was not a significant difference in the mitigation percentage between the two. The largest performance change was seen when switching from a MEMS grade IMU to a tactical grade IMU, allowing for longer outages to be bridged with higher mitigation effectiveness. Tabulated results for mitigation percentage can be found in Appendix B.

6.2 Hardware Simulations

To further validate the effects of GNSS spoofing and meaconing attacks as well as test the proposed algorithm, hardware in the loop (HWIL) testing of GNSS interference systems were used. To generate repeatable testing, a Spirent GSS9000 signal simulator was used as the live sky truth data. By using the Spirent simulator, the same scenario could be run numerous times with the exact same trajectory as was done with the software simulations in MATLAB. Additionally, the simulator allows for total control over the generated signal parameters and satellites in view, allowing for more flexibility in testing the algorithm.

Testing was conducted for both meaconing and spoofing scenarios using the same methodology. Each scenario was started with only valid signals from the Spirent simulator being transmitted into the receiver, a USRP N210 SDR. At a predetermined time, the valid signals were turned off to replicate a loss of GNSS signals and the spoofed signals were turned on. This method was implemented for its relative simplicity and can be representative of a vehicle traveling in an urban canyon or tunnel where GNSS outages occur. More complex spoofing attacks may initially align the false signals with the true signals and slowly drag the receiver off of the true signal, however this capability was not available at the time of the hardware experimentation.

To ensure that the Spirent and receiver oscillators would not influence the change in clock drift at the receiver, phase data was logged using a Keysight 53230A frequency/time interval counter. The $1PPS$ output of the receiver clock, an Ettus Octoclock-G, and the simulator clock, an unknown OCXO, were compared against a Stanford Research Systems FS725 Rb standard. The FS725 was used as a reference due to the superior stability of the outputs when compared to the oscillators under test. Phsae measurements were taken at a rate of one sample per second for a duration of 120000 seconds. The Allan deviation was calculated to determine the short term stability of both oscillators in question, and the resulting curve is shown in Figure 6.48.

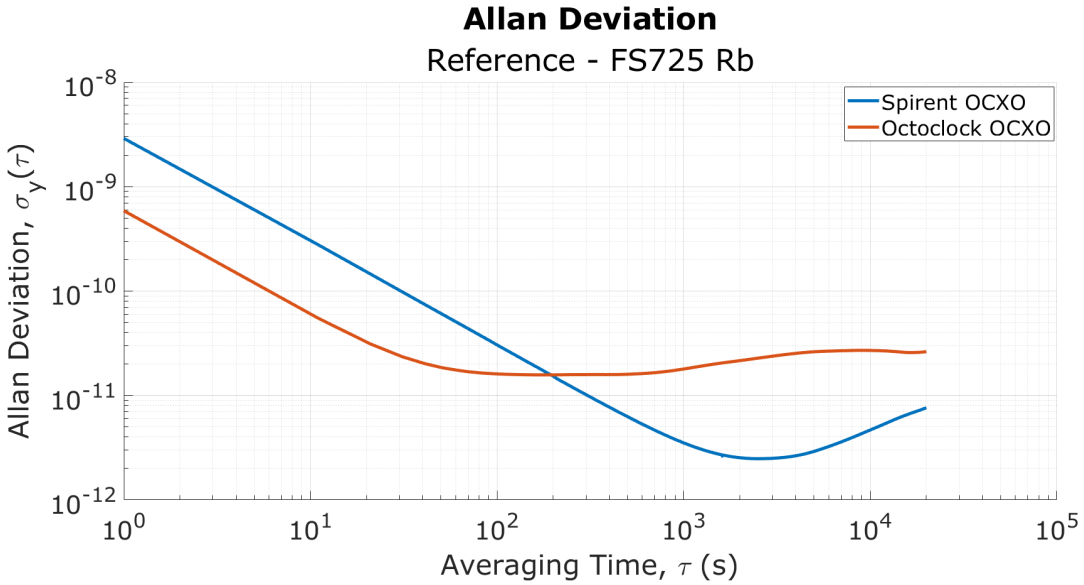

Figure 6.48: Spirent and Octoclock Allan Deviations

The resulting Allan deviation show that both the Spirent OCXO and the Octoclock have relatively stable frequencies until much higher durations than are tested in the simulations. For the Octoclock, the flicker floor begins at just over $100s$ and the random walk noise is not observed on the plot. The Spirent OCXO has a much lower flicker floor than the Octoclock, with the floor occurring at around $2 \times 10^3 s$, which is significantly longer than the durations observed in the hardware simulations. The random walk of the Spirent OCXO would not become apparent until much later, indicating that neither the Octoclock or the Spirent OCXO performance will degrate the testing. For better results in the future, the Spirent and receiver can be synchronized to an external GPS disciplined oscillator (GPSDO) to eliminate frequency error nearly completely. The TCXO that is used in the USRP transmitter has significantly

lower frequency stability than the receiver and simulator clocks, leading to it introducing the most frequency error into the system. This will be observed at the receiver as clock drift, which is used to determine when the receiver is observing spoofed measurements.

A difference between the hardware simulations and software simulations is that only the GPS L1 signal was used in the hardware testing. This was done because of the analog bandwidth of the USRP as well as the limited sample rate of the USRPs. USRP N210s have an analog bandwidth of $40MHz$, which removes any possibility of recording multiple GNSS frequencies at a single time. The true bandwidth is a function of the sample rate, where the bandwidth is limited to one-half the sample rate according to the Nyquist sampling criteria [108]. This criteria is shown in Equation (6.9), where f_s represents the sample rate of the signal and Ω_0 is the Nyquist frequency.

$$2\pi f_s \geq 2\Omega_0 \tag{6.9}$$

To record GPS data that can be processed by GNSS-SDR, a sampling rate that encompasses the GPS L1 bandwidth must be used. For this work, a sample rate of 5 megasamples/s (MSPS) was used. This allowed the L1 bandwidth to be covered as well as reduced the data rate to prevent overflows, which occur when the computer is unable to write samples to a file fast enough, or underflows, which occur when the computer can not provide samples to a USRP at a fast enough rate. An overflow or underflow represent missing samples, creating a gap in what should be continuous data. This gap causes the receiver tracking loops to fail, corrupting the recorded data and making it useless. Each configuration required the streaming of data to two or three USRPs, depending on the test. The following sections go into more detail about the hardware configuration for each test conducted.

6.2.1 Meaconing Hardware Implementation

In the most basic form, a meaconer can be implemented with a single two channel SDR, with one channel receiving the incoming signal and the second channel rebroadcasting the signal at a higher power level towards the target receiver. This configuration was not implemented in testing for this thesis due to poor isolation between the receive and transmit channels on

the N210, allowing incoming signal to directly leak across to the transmit antenna output. In a scenario where the objective is to directly rebroadcast a signal at the same location as the meaconer receiver, this would be less of an issue, however testing for this thesis was conducted with direct RF connections. Since the receiver, meaconer, and Spirent simulator are all directly connected, leakage between the RX and TX ports of the N210 cause signal interference outside of the control of the test as well as negate some of the anticipated effects of the transmitter oscillator.

To avoid the low isolation between the N210 channels, a second USRP was added to the meaconer side of the hardware. This configuration represents a situation where the meaconer receiver is in one location and the transmitter is in another. This configuration may be used to make the attack harder to defeat, as the position estimate of a receiver captured by a meaconer is the position of the meaconer receive antenna. The signal processing delay combined with the additional distance creates the false pseudorange measurements, and if driven by an unsynchronized or low quality oscillator, the meaconer introduces additional frequency error to the retransmitted signal. A diagram of the test configuration is shown in Figure 6.49.

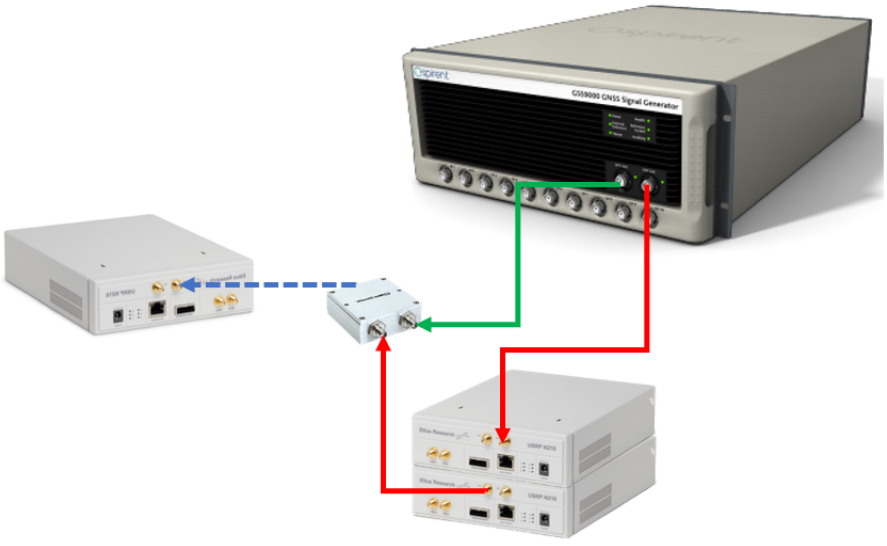

Figure 6.49: Meaconer Hardware Implementation

The USRPs were controlled using GNURadio, with one flowgraph controlling the receiver and another controlling the meaconer. These blockchains can be seen in Figures 6.50 and 6.51, respectively. GNURadio compiles the blocks into Python executables, allowing the flowgraphs

to be run from a terminal window, reducing the load on the computer during operation of the radios. GNURadio allowed for quick configuration of the N210 hardware as well as several fixes for issues that involved the LOs on each USRP. The receiver flowgraph specified a single USRP Source block which recorded all incoming signal data to the specified file shown in the File Sink block. The values for each are set using the Parameter block, which act as variables that are specified as command line arguments.

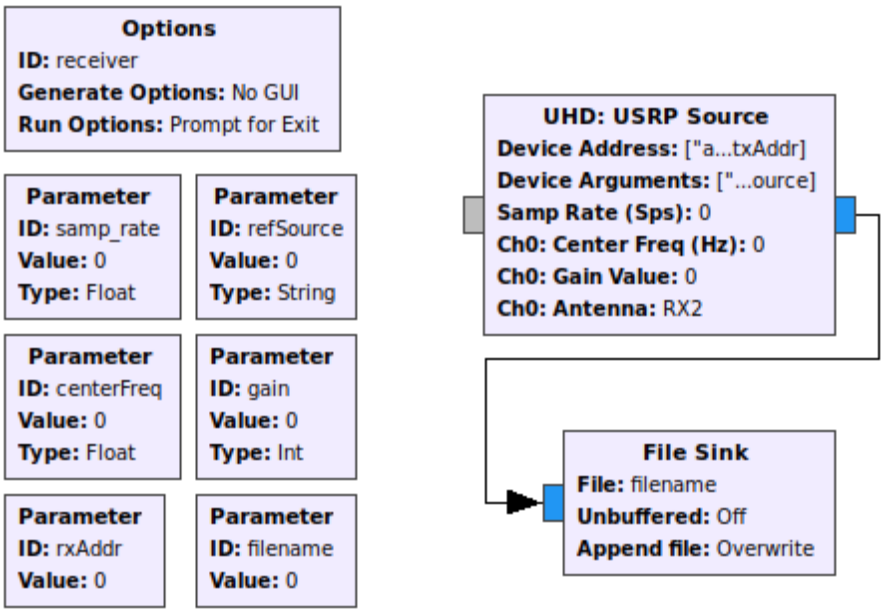

Figure 6.50: Receiver GNURadio Flowgraph

The meaconer implementation used two USRP blocks, one sink and one source as shown in Figure 6.51. The source served the same purpose as the USRP source in the receiver and recorded all incoming signal data. The incoming signal data was then passed from the USRP source through a variable delay block that could be used to add additional phase delay to the signal then to a USRP sink, which specifies the transmitter for the flowgraph. In this implementation, each value was passed as a command line argument, however each individual USRP can be specified by calling the assigned IP address in the "Device Arguments" line of the USRP blocks. By specifying which USRP performed which task, consistency was maintained throughout all hardware testing.

A key aspect to consider during hardware testing is the behavior of a direct down/up conversion (DDC/DUC) radio. The DDC/DUC nature of the N210 brings two additional aspects

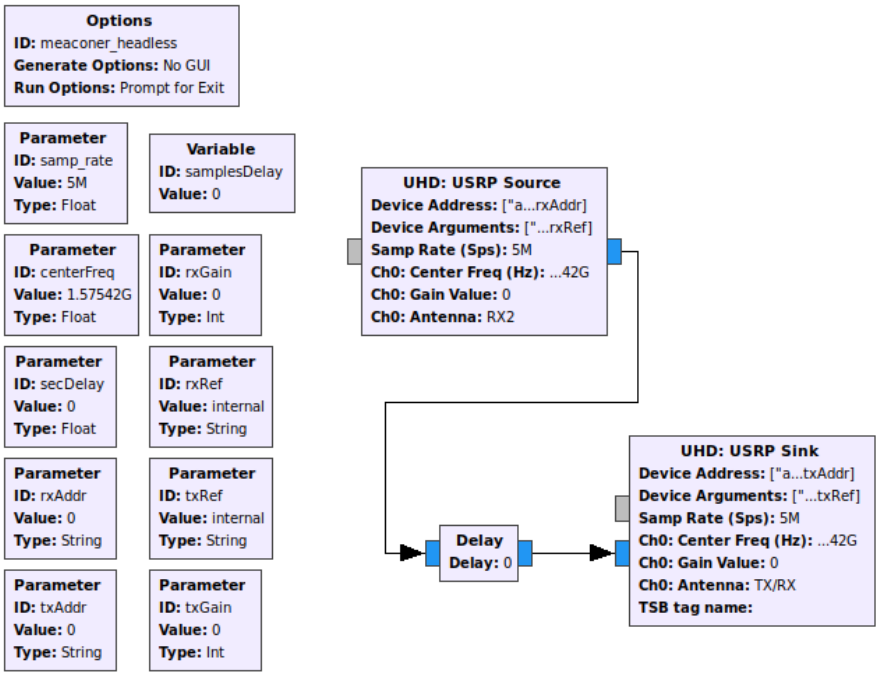

Figure 6.51: Meaconer GNURadio Flowgraph

into consideration, both which involve the behavior of the LO. As described in Section 2.7, DDC receivers directly convert a signal from the received RF frequency to baseband and samples the signal to I and Q samples. The upconvert side of the radio operates in the opposite way, taking baseband IQ signals and directly converting them to the RF to be transmitted. Both of these operations, shown in Figures 6.52 and 6.53, respectively, require the LO to operate at the center frequency of the signal being received or transmitted. Having the LO at the frequency of interest has the potential to introduce significant noise if the LO power leaks into the RX or TX channels, essentially acting as a jammer at the center frequency being recorded. To avoid this issue, the UHD interface allows the N210 LO to be operated at a frequency near the desired center frequency and the remaining frequency offset is removed in digital processing. For each USRP block, an argument specifying the center frequency is used to configure the SDRs. By changing this from the exact value of the center frequency to the command `>>uhd.tune_request(centerFreq,loOffset)`. This command issues the request to offset the center frequency by a specified amount, which allows the LO spike to be moved outside of the sampling bandwidth. This improves the sampled signal quality significantly and prevents the LO from acting as a jammer.

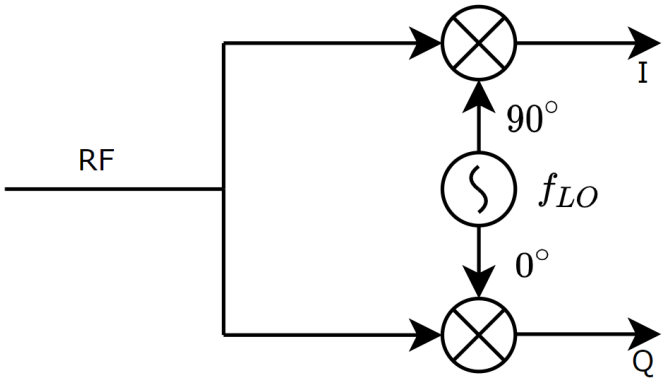

Figure 6.52: Direct Downconversion of RF Signal to Baseband

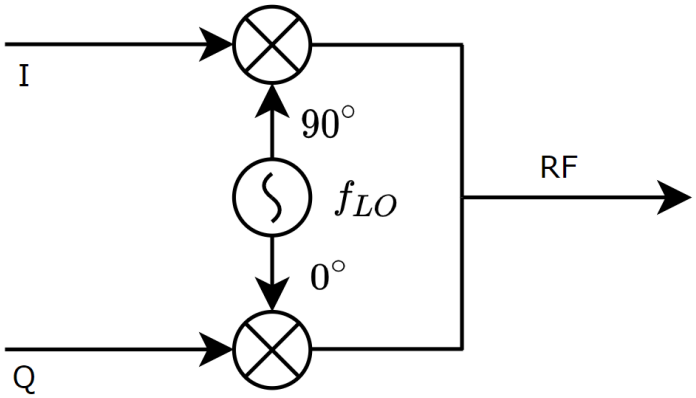

Figure 6.53: Direct Upconversion of Baseband Signal to RF

The second issue that arises in a DDC/DUC receiver is related to the effect of the LO frequency offset on the received and transmitted signals. In a normal receiver, the measured Doppler shift at the receiver is the combination of the transmitted frequency error, motion between the transmitter and receiver, and receiver clock frequency error. As discussed previously, when a signal is being transmitted, the IQ samples are mixed with the LO and transmitted at the desired RF. In a scenario where the receiver and transmitter are driven by the same reference clock and the samples are retransmitted quickly after they are received, the frequency of the LO will not have changed much if at all. Since the samples would be downconverted and then upconverted with the same frequency offsets, the transmitted signal would appear to have no additional frequency error introduced. In the hardware configuration shown in Figure 6.49, the original test had both meaconer USRPs driven by a single reference clock. Then the signal was processed at the receiver, there was no noticeable additional clock drift, causing the mitigation

algorithm to fail to detect any interference. The results of this first test are shown in Figure 6.54, where the meaconer was turned off after an outage and the least squares estimate of clock drift remains constant before and after the outage.

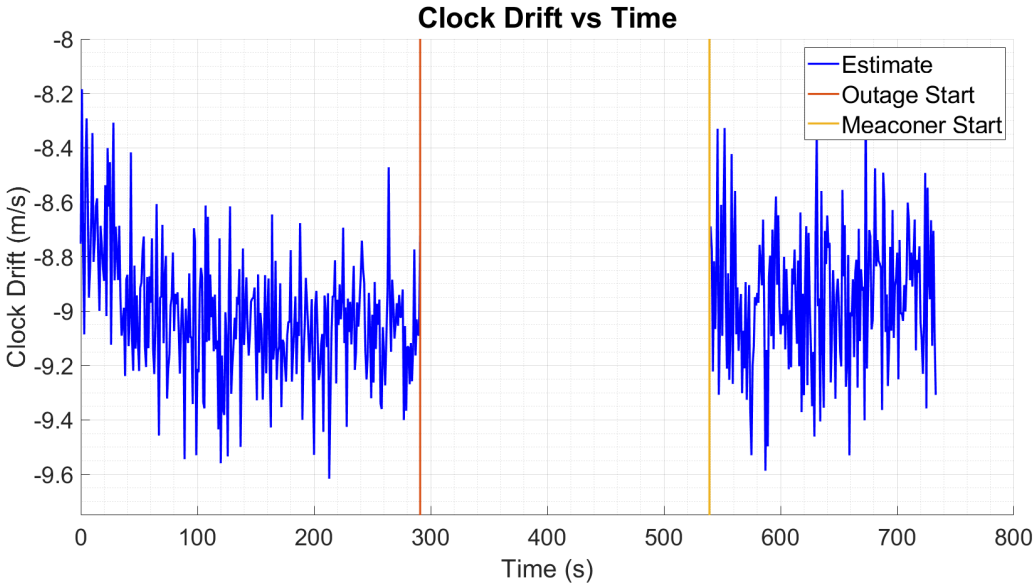

Figure 6.54: LS Drift Estimate Before and During Receiver Capture - Single Clock

To mimic a more realistic scenario where a recorder and transmitter are separated in position and therefore driven by separate clocks, the recorder was driven by the Ettus Octoclock and the transmitter was driven by the internal TCXO. This configuration could be reversed as well with the same final affects still being observed. The receiver was driven by an internal GPSDO, ensuring that the majority of the frequency error that was observed was a result of the transmitter oscillator as was done in the MATLAB simulations previously. As expected, the additional frequency error becomes observable in a jump in clock drift at the receiver when the receiver is captured by the meaconer, as shown in Figure 6.55. Testing was done using the internal oscillator for both the recorder and the transmitter, however GNSS-SDR was unable to track the data due to extremely poor phase noise and frequency error as a result of the TCXOs. The configuration described above was used for all meaconer testing conducted for this thesis.

Once the USRP hardware implementation was finalized, the Spirent GSS9000 simulator was configured for use with the hardware scenario. To generate the truth trajectory, the same data used in the MATLAB simulations was written to a `.umt` file, which passes reference data to the Spirent simulation. The `.umt` file provides the user the ability to specify a timestamp and

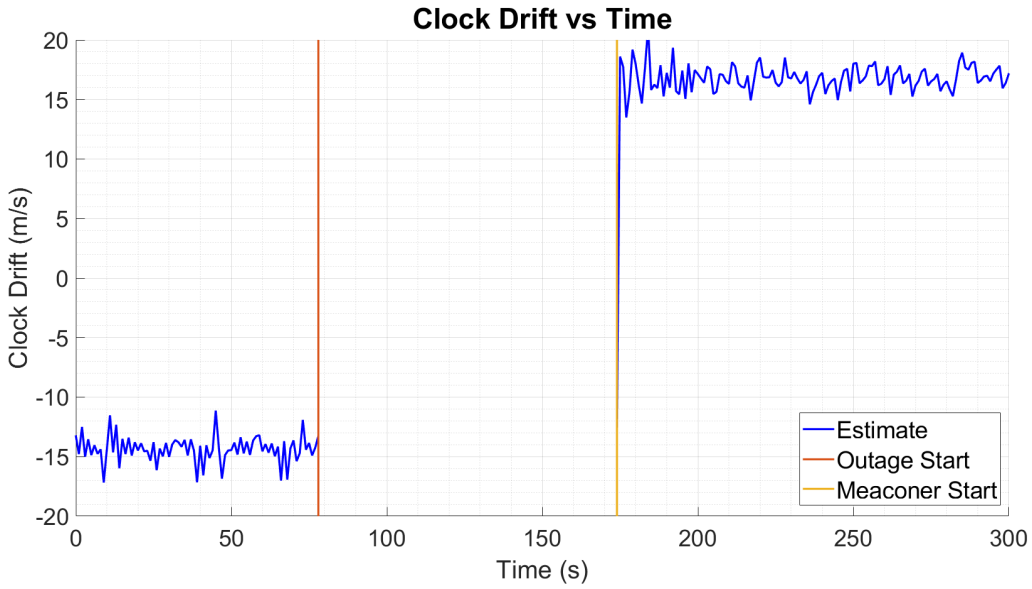

Figure 6.55: LS Drift Estimate Before and During Receiver Capture - Separate Clocks

a variety of dynamic properties of the vehicle up to the third derivative of position and angle (translational jerk and angular jerk, respectively). This enabled the INS data to be simulated in the same manner as was done with the MATLAB simulation and synchronized with the GPS data recorded by the N210 receiver. The $RF1$ output on the simulator was connected directly to the splitter as shown in Figure 6.49 and was simulating the truth GPS trajectory. The meaconer was implemented using the $RF2$ output on the simulator, with a static position chosen as the meaconer receiver location. The $RF2$ output was fed into the N210 meaconer then replayed using the second N210, with the RF output connected to the second channel of the splitter. The splitter was connected to the receiver N210, where the RF data was recorded for processing in GNSS-SDR and MATLAB. The Spirent configuration is shown in Figure 6.56, where the ground track of the truth trajectory is seen in the bottom left, and the meaconer location is shown on the bottom center.

Using the meaconer RF data collected by the N210 receiver, the spoofing detection and mitigation algorithm was again analyzed using a MEMS and tactical IMU, along with the loose and tight coupling of the GPS/INS hardware. The results for these tests are shown in the following section.

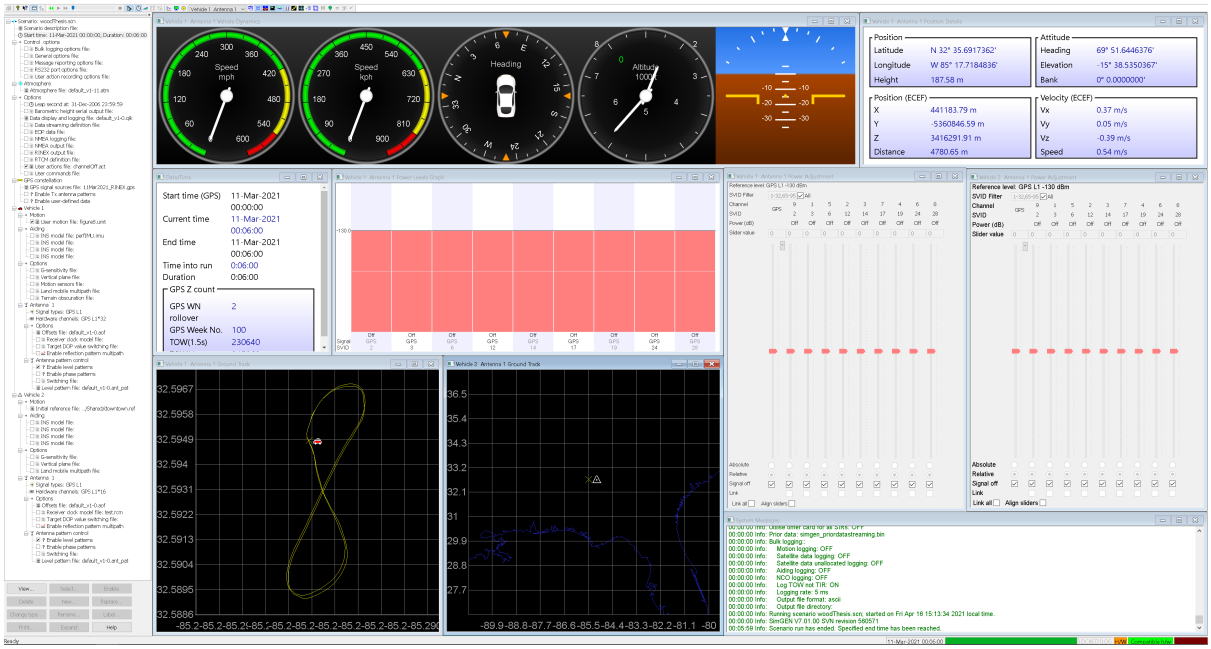

Figure 6.56: Spirent Simulation Window

6.2.2 Meaconing Results

As stated previously, each test was conducted using a MEMS and tactical grade IMU with a loosely and tightly coupled integration filter. The same RF data was used for each test, and the IMU measurements were simulated using the same model as the MATLAB simulations discussed previously. For these tests, a signal outage began at $t = 112s$ into the run, and the outage continued until $t = 157s$, when the meaconer signals were tracked by GNSS-SDR. Unlike in the MATLAB simulations, only the L1 signal was processed for the hardware results for reasons discussed in the previous section. Because of this limitation, the receiver was eventually captured in all scenarios, however if additional frequencies were used this would have been prevented, reflecting the results shown in simulation.

The first test conducted was with a MEMS IMU and a loosely coupled integration. The results for this test are shown in Figure 6.57 below. The outage for the hardware testing was significantly longer than the individual tests shown for simulation, and the Monte Carlo results show mixed performance at longer outage durations. The results shown here show that the algorithm, even with the low quality IMU, is able to detect the interference for a short period of time prior to capture. The clock drift of the N210 transmitter is $160m/s$, which is greater

than the range of clock drifts in the Monte Carlo simulations. Since the transmitter drift is larger, the duration of outage the algorithm can successfully survive is also increased. The MEMS LC implementation was able to survive the 45 second outage until the propagated drift threshold increased to a value that was larger than the faulty drift measurements at $t = 158s$, only one sample after the meaconer measurements were introduced. When the faulty measurements are inside of the drift bounds, they are marked as valid and used to update the navigation solution, causing the receiver to estimate an incorrect PVT solution, which then corrupts the INS error estimates in the integration filter. As stated previously, however, if other frequencies such as L2/L5 or E5a were used in addition to L1, the receiver would be able to estimate a valid position after the outage and the uncertainty bounds would decrease again, continuing to exclude the faulty measurements. With such a small window between the fault bounds and the drift estimate, the performance of the LC MEMS filter is not ideal for most applications, however this would likely be the most common implementation on commercial drones or autonomous vehicles.

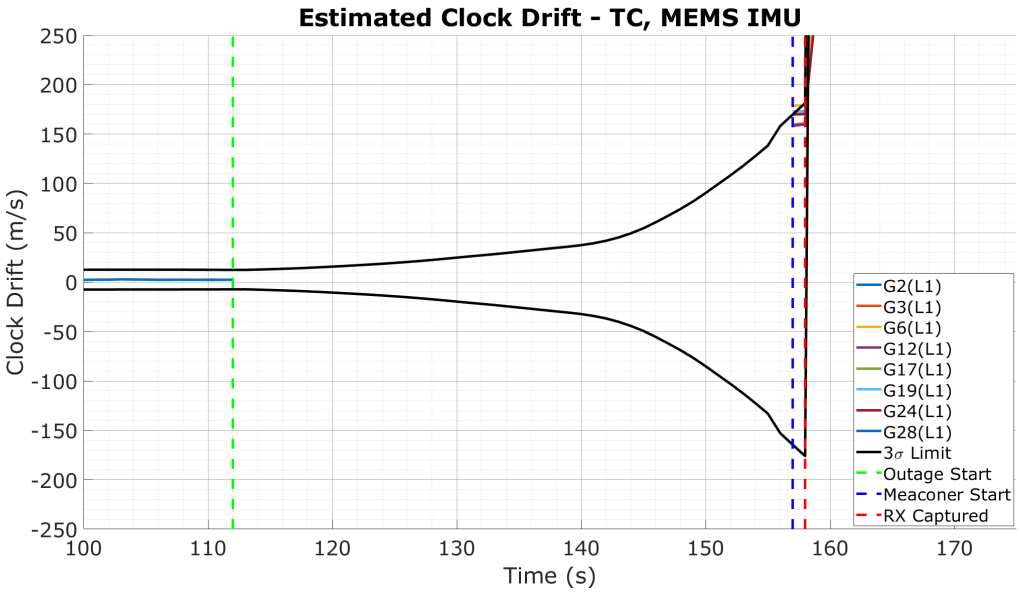

Figure 6.57: Hardware Meaconer Results - LC, MEMS IMU

Continuing with the loose integration of the GPS/INS, a tactical grade IMU was simulated and tested with the same hardware measurements. As expected, the tactical IMU uncertainty grows more slowly than the MEMS IMU uncertainty, leading to the clock drift bounds to grow

at a slower rate. As a result, the LC tactical implementation is able to remain unspoofed for a longer period of time after the meaconer measurements are introduced into the system. As was the case in the first test, the meaconer measurements are introduced at $t = 157s$, and the receiver is able to reject the faults until $t = 176s$, which was $14s$ longer than the MEMS LC implementation was able to survive the event. This longer duration would allow more time for the filter to reconverge after an outage if other GNSS measurements were available, ensuring that the receiver would not be captured by the meaconer in this situation.

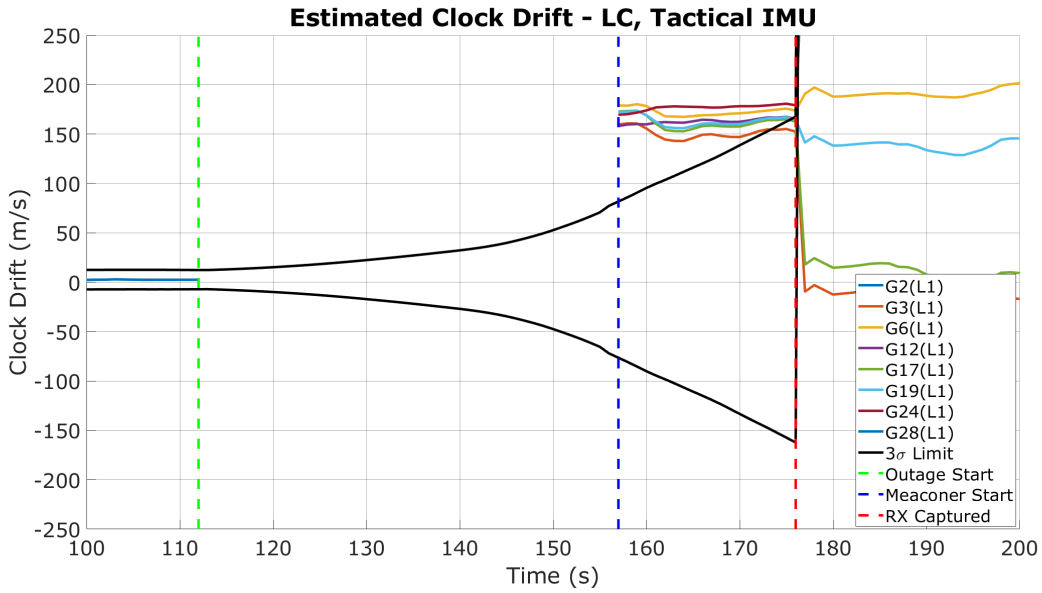

Figure 6.58: Hardware Meaconer Results - LC, Tactical IMU

The tightly coupled implementation was evaluated using both grades of IMU as well. As was seen in Figure 5.7, the tightly coupled filter drift error bounds are tighter than that of the loosely coupled filter, which leads to the expectation that the TC filter will be able to avoid capture by an interference source across longer outage durations. The difference between the LC and TC error propagation is fairly minor, as the clock model and IMU model used in both filters are the same. The increased confidence comes from the Kalman filter's use of previous information instead of using a single position estimate to update the INS states at each point, which reduces the error bounds. Since the error rates grow at the same rate for both cases, the scenario with lower error bounds at the start of an outage will reach the fault point later than the scenario with increased error bounds. The tightly coupled results shown in Figures 6.59 and

6.60 reflect this conclusion, where both the MEMS and tactical IMU with the TC integration survive the meaconing attack longer than the respective LC implementations.

In Figure 6.59, the outage and meaconer start at the same time as in the LC cases ($t = 112s$ and $t = 157s$, respectively). When the meaconer is turned on, all of the false signals are outside of the MEMS confidence bounds, enabling the receiver to detect the interference and continue to reject the faults. At $t = 160s$, more than half of the faulty signals are contained within the error bounds, leading the receiver to use several of the false measurements to estimate a position solution. At this point, the receiver is considered captured, as the state estimates are corrupted by the faulty data. Compared to the LC MEMS scenario, the TC scenario was able to last two seconds longer. This does not seem like a significant improvement, however this means that there are two additional potential measurement updates to use other frequencies to calculate an accurate position solution, which would prevent the faulty measurements from entering the receiver and allow the navigation algorithm to successfully mitigate the attack.

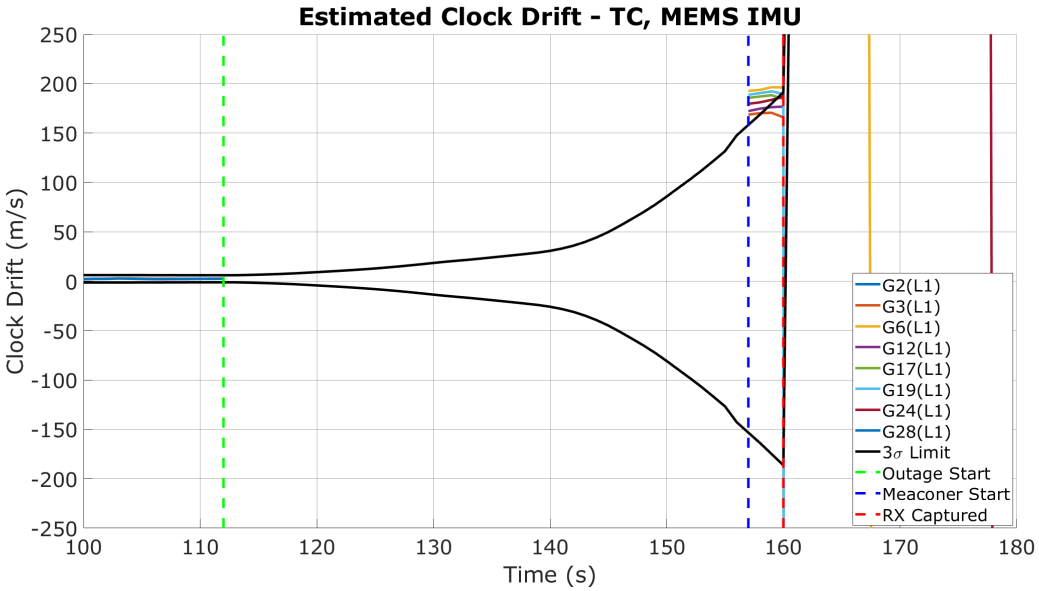

Figure 6.59: Hardware Meaconer Results - TC, MEMS IMU

The TC implementation using a tactical IMU shows the same trend as the MEMS TC implementation, with the algorithm rejecting the faulty measurements for a longer duration of time than the LC tactical implementation. In the LC case, the receiver was able to avoid capture until $t = 176s$, whereas in the TC case the receiver avoids capture until $t = 180s$, four seconds

longer than the LC case. As has been discussed with all the other cases, any time after the meaconer begins where the receiver rejects the measurements allows the receiver to track other frequencies/signals and calculate a solution using valid measurements.

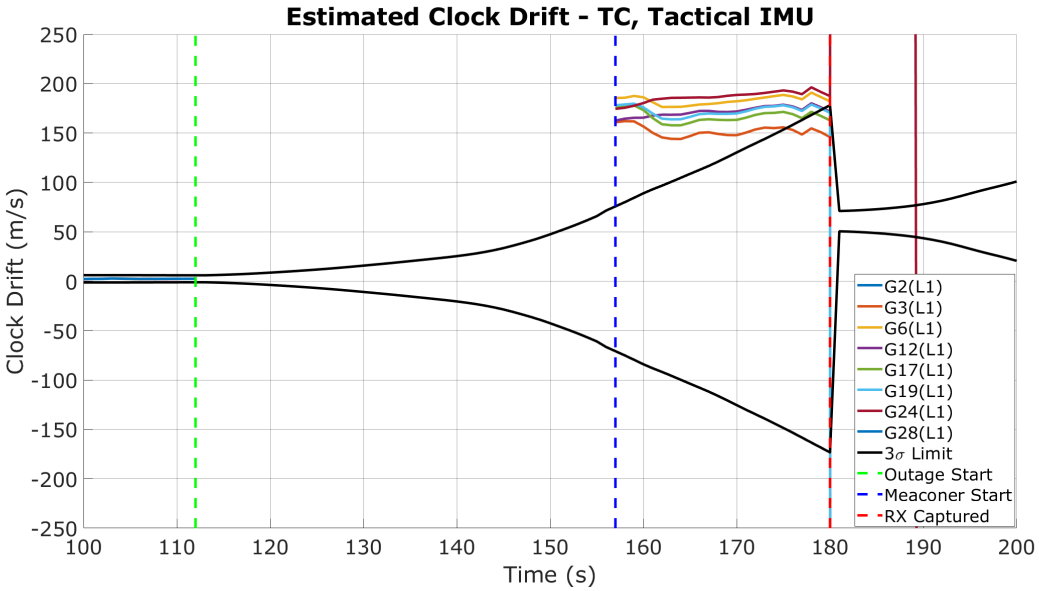

Figure 6.60: Hardware Meaconer Results - TC, Tactical IMU

The results shown for the hardware meaconer represent only a single data point which used a low quality TCXO for the transmitter. As was observed, the TCXO has significant drift error, leading to the simple meaconer to be readily detected using the algorithm. If the transmitter was driven by a different oscillator, such as a more accurate OCXO or atomic frequency standard, or synchronized to a GPSDO, the detection rate would decrease significantly. An example of a transmitter synchronized to an Abracon AOCJY2 OCXO is shown in Figure 6.61. The change in clock drift from the live sky scenario to the captured scenario is still significant and can be observed. If the outage duration was much longer than several seconds, the receiver would fail to adequately detect the jump and become captured, making the navigation solution useless for the vehicle in question and could lead to disastrous results.

6.2.3 Spoofing Hardware Implementation

Implementing a simple spoofer using the USRP hardware was straightforward, requiring only the generation of the spoofer data prior to the test as would likely be done by an individual

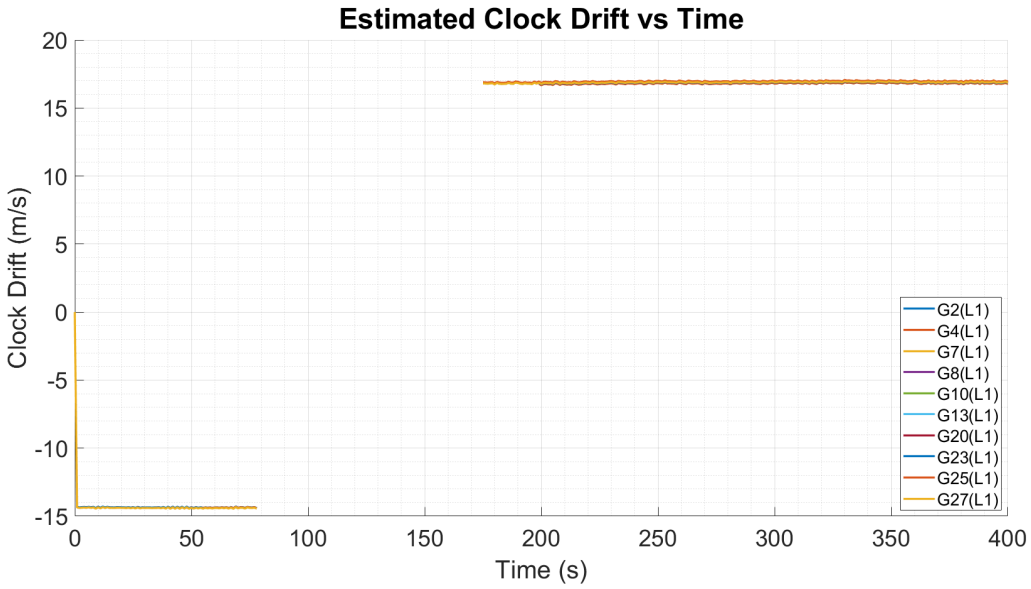

Figure 6.61: Clock Drift Estimate - OCXO Transmitter Clock

attempting to travel the world to capture Pokemon in different regions. As was the case with the meaconer tests, the $RF1$ output on the Spirent GSS9000 signal generator was used for the truth signal, and was fed to an RF combiner. The spoofer signal, generated using GPS-SDR-SIM, was specified as a static point near the receiver to allow for the same satellites to be observed. In reality, an individual generating a signal at a completely random location would create a scenario where the satellites before and after capture would not match, allowing the event to be easily detected. In notable spoofing examples, however, the receivers are fooled to believe they are within the same region, allowing the satellites to remain unchanged [9], [7], [109]. These attacks are much more complex than a simple signal generation attack, however by using the same satellites the hardware simulation is more consistent with what may be experienced by commercial receivers.

Once the spoofer signal was generated, it was combined into the second RF port of the signal combiner, which is connected to the receiver N210. The complete hardware setup is shown in Figure 6.62. At a point in the scenario, the Spirent truth signal was turned off, representing an outage due to a drone or car passing through an urban canyon or under a bridge, and the spoofer was turned on. The spoofer time solution is drastically different than the true receiver offset, which would create a detectable spike in clock bias, however this is not ideal to use for detection as the jump would occur after the measurement faults have entered the receiver. By

observing the estimate of drift, an unsynchronized transmitter can be detected in the same way it was detected with the meaconer case.

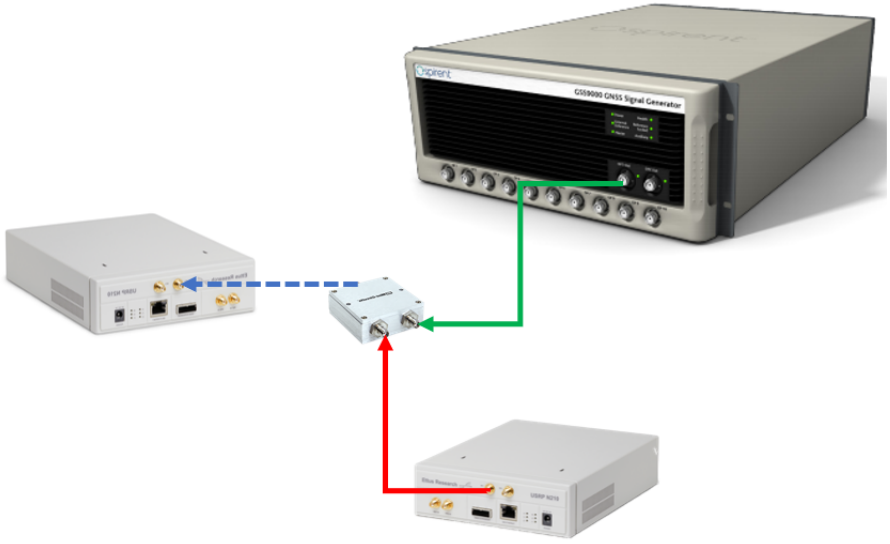

Figure 6.62: Spoofer Hardware Implementation

To allow the USRPs to transmit the data generated by GPS-SDR-SIM, either the command line interface (CLI) had to be used or the GNURadio blocks adjusted, as the data generated was baseband int16 IQ data as opposed to the float32 data more commonly used by GNURadio. Both the CLI and GNURadio offer simple implementations, and GNURadio was chosen for the ease interfacing with other USRPs. To allow the flowgraph to transmit the samples, the file source was adjusted to read the correct data type from the specified file, which was then passed to a USRP which was changed to expect two 16-bit complex values for each IQ pair. Once these changes were implemented, the spoofing case was executed in a manner that was similar to the meaconer case. The USRP transmitter used the internal TCXO as the LO, and the receiver was synchronized to the Ettus Octoclock, as was done in the meaconer experiments.

Once the RF data was recorded, the IMU data was simulated with the same figure eight trajectory that was used for all previous software and hardware simulations. The LC and TC integrations as well as the MEMS and tactical IMU combinations were all analyzed as was done with the meaconer hardware testing. The outage duration for the spoofer test is $51s$, with the outage starting at $t = 142s$ and the spoofer measurements received at $t = 193s$. This outage is slightly longer than the $45s$ outage seen with the meaconer, which was at the upper limit of

the detection capabilities of the MEMS integration. The spoofing case results will be discussed further below.

6.2.4 Spoofing Results

As mentioned in the previous section, the results from the spoofing tests conducted look much like the results from the meaconer. This is due to the simple implementation of the spoofer, where the spoofer LO is a TCXO and the position is not dragged off, but instead a position jump is introduced. This is the simplest spoofing attack and mimics the meaconing attack, but can be done with any position in the world as discussed. Using the same TCXO and trajectory as in the meaconer case, the LC and TC as well as MEMS and tactical IMU combinations are again analyzed and the results shown.

Figure 6.63 below shows the first test case, the loosely coupled integration with a MEMS IMU. In the meaconer case, the outage lasted a total of $45s$ and the LC MEMS filter was able to survive a single second prior to being captured by the spoofer. In this test, the outage lasted for $51s$ total, causing the estimated clock drift to be within the error bounds when measurements returned. This can be seen by the fact that the navigator does not reject any measurements and the drift estimates inside the bounds diverge immediately after the first estimated value. Since the estimates diverge and exit the confidence bounds, the spoofing is detected, however the state estimates have been corrupted and the solution is no longer valid. This could be avoided by using a local buffer of previous valid state estimates, however this quickly becomes computationally intensive and unwieldy for low SWaP-C systems such as UAS or autonomous vehicles.

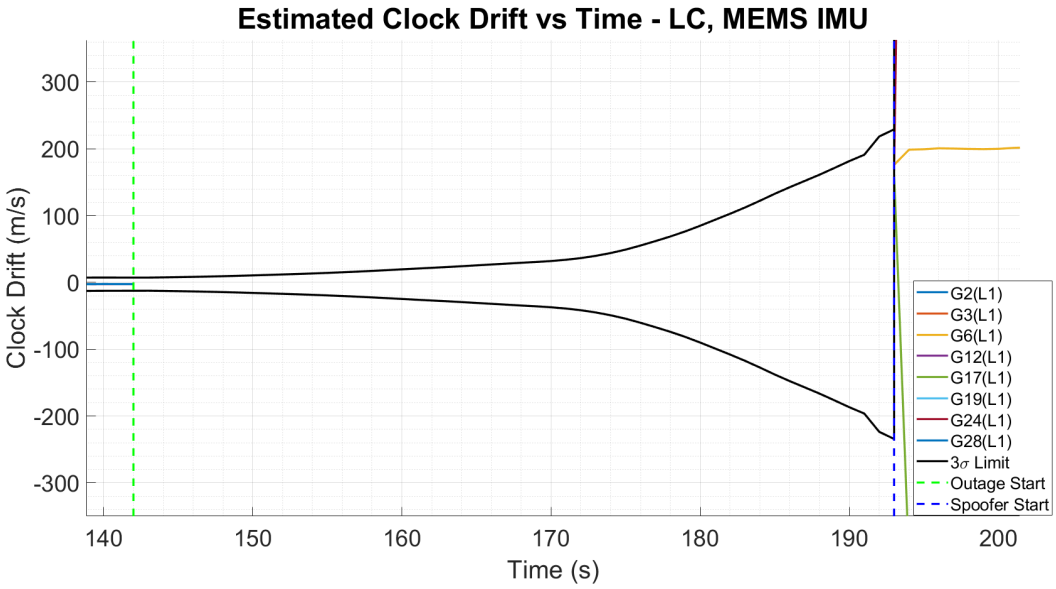

Figure 6.63: Hardware Spoofer Results - LC, MEMS IMU

As expected, with the longer duration of the outage, the MEMS implementation fails to successfully mitigate the attack before the receiver is captured. The tactical implementation, on the other hand, is able to prevent faulty measurements from entering the navigation algorithm as shown in Figure 6.64. When the spoofer measurements are received at $t = 193$, the estimates are outside of the drift bounds as the tactical bounds grow more slowly than the MEMS IMU. The slower growth allows the LC tactical implementation to identify and avoid capture until $t = 207s$, or $14s$ after the spoofer measurements are introduced.

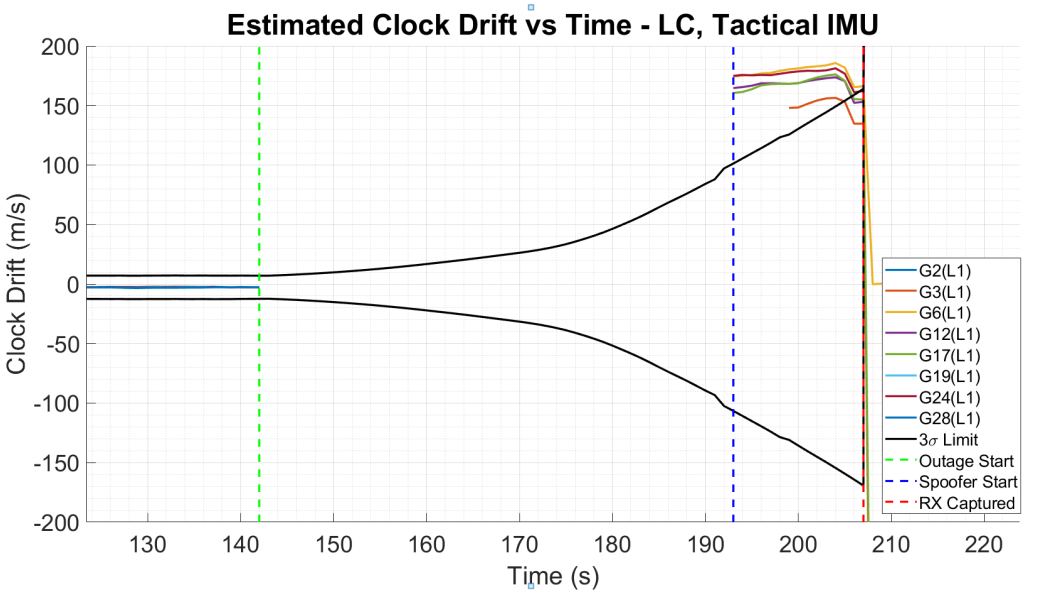

Figure 6.64: Hardware Spoofer Results - LC, Tactical IMU

Using a tightly coupled implementation with the MEMS IMU does not provide better results than the LC MEMS implementation, with the navigator still being captured and corrupted by the spoofed measurements. These results can be seen in Figure 6.65, where the spoofer measurements were introduced at $t = 193s$. The drift estimates appear as a vertical line again due to the jumps in drift estimates once the navigator is captured, which allow the spoofing to be detected but only after corrupting the navigation solution. these same results were observed in the LC implementation. If the duration of the outage was reduced or the TCXO drift was increased, the algorithm would have a better chance to detect the errors, however the low quality MEMS IMU drives the reduced capabilities of these implementations.

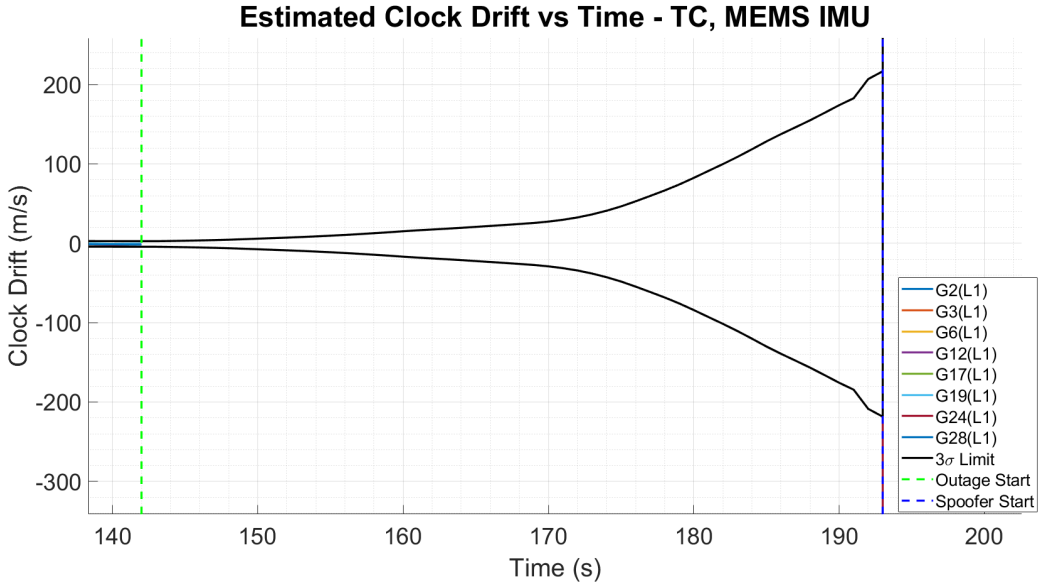

Figure 6.65: Hardware Spoofer Results - TC, MEMS IMU

As was seen in the previous test cases, the tactical IMU implementation has better performance than the MEMS IMU, as is expected for a higher grade and higher cost sensor. In the spoofing test, the TC tactical implementation was able to detect the interference and prevent the faults from entering the receiver for the longest time frame, as shown in Figure 6.66. This implementation was able to prevent measurements from corrupting the receiver for $23s$, with the error bounds exceeding the drift estimate at $t = 216s$, a full $9s$ longer than the LC tactical implementation. As expected from the meaconing results, the TC tactical implementation performs the best compared to all other algorithm implementations, and provides the largest resistance to longer outages.

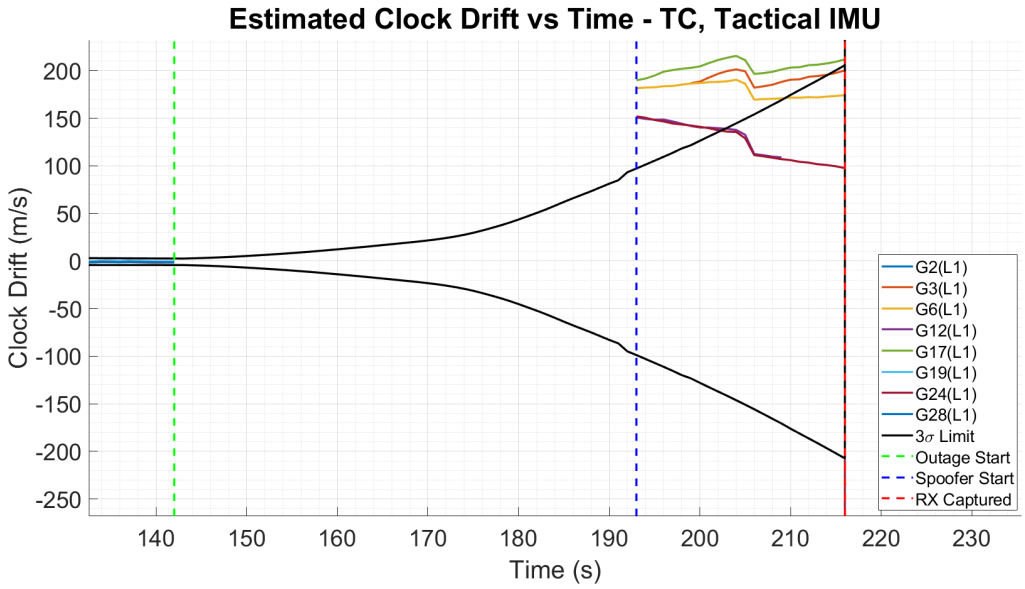

Figure 6.66: Hardware Spoofer Results - TC, Tactical IMU

6.3 Conclusions

The results shown in this chapter demonstrate that the proposed algorithm is able to detect simple spoofing attacks as well as it is able to detect simple meaconing attacks. Simple spoofing typically causes unintended interference, where an individual or organization may not realize the transmitted signals are creating problems for nearby systems. The proposed algorithm can successfully detect spoofing generated by low cost software receivers with no external hardware, which introduce additional frequency error into the navigation solution beyond what is expected from the receiver oscillator and satellite transmit oscillator. Transmitters with higher quality oscillators are significantly harder to mitigate, with many of the scenarios unable to remove the faults introduced by these types of transmitters.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis presented an expanded spoofing detection and mitigation algorithm based off a previously documented static method which focused on the receiver clock drift [20]. By taking the time difference of received carrier phase measurements and the estimated range updated by an INS, an estimate of clock drift was able to be obtained. INS integration allowed the drift to be estimated after signal outages without the need for a measurement update, allowing this estimate to be used for integrity monitoring purposes. The method proposed here did not require additional sensors beyond those present on many commercial UAS platforms or autonomous vehicles, allowing the algorithm to be implemented with relative ease. Detailed information about oscillator simulation was presented and the discussed simulation methods were implemented in simulation to test the feasibility of the proposed algorithm in simulation. An overview of various inertial sensor grades was also presented, and sensor fusion architectures for GNSS/INS filters were shown and implemented to examine the performance of each when paired with the proposed algorithm. The algorithm proposed here was shown to have good performance in many scenarios, even when the transmitter was driven by a stable oscillator.

One main issue with using INS to bridge outages is that the position solution will drift further away from the truth the longer the outage is. A sensitivity analysis conducted using the clock drift estimation algorithm on a static point was conducted to show that the drift estimate

after an outage will lie within the drift bounds propagated forward in time. The sensitivity analysis showed that a position error of $50km$ introduced about $5m/s$ error into the drift estimate, which was at the threshold of the drift bounds while valid measurements were available. After an outage, the drift bounds consistently grew larger than the $5m/s$ drift error, indicating that the position error at the end of the outage would not adversely affect the drift estimation algorithm.

Testing of the algorithm was conducted using several simulations which replicated the received measurements from a meaconer or spoofer. While individual tests showed good performance results with the algorithm able to detect and mitigate the effects of meaconing or spoofing attacks, the stochastic nature of INS errors and clock errors lent itself to the need for a Monte Carlo simulation. The Monte Carlo simulations conducted showed that the algorithm had a high success rate in detecting spoofing attacks and alerting the operator to the degraded or disrupted status of the navigation solution available to the platform. In terms of mitigation, the algorithm was able to successfully mitigate the majority of interference when the transmitter was driven by a low quality oscillator such as the TCXOs commonly found on commercial SDR platforms. When transmitters were driven by higher quality oscillators, such as more expensive OCXOs or atomic standards, the mitigation performance decreased significantly however the detection performance remained the same.

Not only did the simulation results show the performance of the proposed algorithm, they also showed the effect of the position difference between the receiver and the meaconer or spoofed location. When the true and spoofed location were identical, the clock drift estimate was the combination of the receiver clock drift and the transmitter clock drift. This was represented in the dynamic location spoofing results, where the proposed algorithm was able to detect and mitigate nearly all spoofing from transmitters with low quality LOs and was unable to even detect transmitters with high quality oscillators that perfectly spoofed the receiver location to the true location. When there was a difference between the receiver and transmitter location, however, the clock drift estimates no longer represented the combination of drift alone. The position error caused the drift estimates to spread out and not lie along the same value due to the difference in geometry between the satellites and spoofed position compared to the geometry between the satellites and true position. This spreading of the estimates allowed

meaconers and static position spoofers to be detected even when driven by higher quality oscillators, however at a significantly reduced rate as when driven by a low quality oscillator.

Simulation results showed that at short outage times of $t = 10s$ allowed the receiver to detect many of the faults that occurred from low quality transmitter oscillators. When the outage times increased to $t = 30s$, the mitigation percentage dropped between $10 - 20\%$ compared to the ten second outage time. This trend would continue until as the outage time increased until the receiver was unable to detect any spoofing effects at the end of a signal outage. When the transmitter was driven by a high quality oscillator and the spoofed location was adequately far from the receiver location, the receiver was able to detect the effects due to the spread of the drift estimates. At outage times of $t - 20s$ or greater, the spreading effect was not enough to cause the drift estimates to exceed the drift bounds.

Experimental hardware tests were conducted to verify the performance that was observed in simulation. Using commercial software radios and open source software as well as commercial simulators, both the static spoofing and full capture meaconing scenario were replicated and the results analyzed. The results from both HWIL tests showed the same trends as were observed in the software simulations, with the clock drift estimates spreading out due to the geometry error between the estimated ranges and measured carrier phases. The clock drift introduced from the USRP TCXO was much larger than the simulated TCXO error, leading to better detection performance compared to the simulated individual scenarios. This difference was primarily due to the simulations taking the drift variance at $t = 24hrs$, while the USRP had been powered on for several days prior to conducting testing. Since the transmitter drift value was larger, the length of the outage after which the faults were detected was longer. Compared to the $10s$ of the software simulations, the hardware experiments showed the receiver was able to detect the faults after a $45s$ outage. If the transmitter clock drift was not as large, the results would reflect the performance that was seen in the software simulation results. The hardware results showed that the algorithm was able to successfully detect and reject faulty measurements until the fault threshold grew larger than the drift estimates, which would be avoided if more frequencies/constellations were used as was shown in the software simulations. As expected, the implementations that relied on a tactical grade IMU survived $2 - 5s$ longer than the

MEMS implementations. This is a relatively small difference, with the MEMS implementation performing adequately in all the hardware tests conducted.

Overall, both the software simulations and hardware experiments demonstrated that the proposed spoofing detection and mitigation method performs well in many scenarios, however it is not without limitations. When transmitters are driven by higher quality oscillators, the detection performance significantly degrades. In addition, the tightly coupled implementation may be prohibitive to augment onto systems that do not provide the ability to integrate the measurements at the receiver level. The loosely coupled implementation demonstrated promising performance when paired with a tactical grade IMU, while the loosely coupled MEMS IMU implementation performing marginally worse. The LC MEMS implementation is most likely to be implemented on low cost commercial UAS and autonomous systems, and while the performance was adequate, it was not a catch all solution. The proposed algorithm accomplishes the desired objectives in many scenarios, and performance could be further improved by integrating additional spoofing detection algorithms.

7.2 Future Work

Analysis of the results and conclusions from the proposed spoofing detection and mitigation algorithm showed several areas of potential future research and improvement. First, additional testing of the algorithm using a full HWIL setup simulating both inertial and GNSS measurements would provide a high fidelity and repeatable simulation environment that could be used for in depth testing. For this work, the internal TCXO on the USRP N210 SDR was used as the transmitter clock. Additional hardware testing with higher quality transmitter clocks would provide additional comparison to the software simulations and expand on the results that were observed in each simulation. Testing various SDRs as transmitters would again provide more data points in hardware testing as well as expand the number of low quality oscillators tested. A final step for hardware testing would be to collect RF signal data from sanctioned test events where GNSS repeaters or spoofers were being legally operated. This data would provide the most thorough analysis of the proposed algorithm, however the feasibility of this test method is unknown.

In addition to further hardware testing, several simulation aspects could be expanded upon. Specific vehicle platforms could be tested, allowing the environmental effects such as thermal changes or vibrational profiles to be simulated and the response of the receiver clock as well as the detection and mitigation performance analyzed. For spoofing tests where the proposed algorithm failed to detect the event and was captured without an alert, the effects of a time or position push could be analysed and determine the limits of errors that could be introduced to the receiver by a spoofer before the algorithm was able to detect the faults that were present.

Several areas relating to the detection of faults also showed areas of improvement. The first area that could be improved is the method of validating the drift estimates against the expected values. Using additional sensors or nonholonomic constraints for platforms such as ground vehicles could provide the means to decrease the drifting position error and provide limits on the drift bounds over longer outage times. Additionally, the variance of the received measurements has the potential to be analyzed and used for spoofing detection. During a nominal, live sky test case, the clock drift estimates from each channel have very low variance and all lie near the true receiver clock drift that was propagated using a clock model. When the meaconer or static spoofer measurements are received, the difference between channels is much greater. This difference has the potential to be monitored and used to determine if the estimates are valid or if there are outliers, similar to how a RAIM algorithm works when calculating a solution with $n-1$ satellites and comparing each for outliers. Drift estimates from additional sources such as signals of opportunity (SOP) could be incorporated into the comparison, with numerous SOPs available from low earth orbit (LEO) satellites and cellular networks.

A detection algorithm that uses estimates of both receiver clock bias and drift could provide additional improvements in detection and mitigation of faults. By identifying potential faults in the drift prior to the measurement update and observing jumps in the receiver clock bias, a RAIM-like algorithm could be implemented to further identify the faulty measurements present. Removing detected faults prior to the RAIM algorithm would increase computational efficiency by reducing the number of iterations that are needed at each time step. The analysis of clock bias and drift could be further extended, relying on a network of connected GNSS

receivers with synchronized or precision oscillators operating in the affected area. An algorithm that uses drift estimates from each receiver could compare indicated faulty satellites and determine whether the faults are valid and potentially identify faults that are missed by some receivers. This method would rely on additional hardware, such as networked radios and cooperative navigation systems, which may be achieved in the near future with autonomous road vehicles.

Bibliography

[1] (Apr. 22, 2020). GPS accuracy, [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/#:~:text=Learn%20more-,How%20accurate%20is%20GPS%20for%20speed%20measurement%3F,interval%2C%20with%2095%25%20probability. (visited on 03/17/2021).

[2] Amazon. (). Amazon prime air, [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

[3] Department of Homeland Security. (Apr. 2, 2020). Snapshot: First responders assess drones for search and rescue missions, [Online]. Available: https://www.dhs.gov/science-and-technology/news/2020/04/02/snapshot-first-responders-assess-drones-search-and-rescue-missions#:~:text=Law%20enforcement%20responders%20use%20drones,hidden%20in%20an%20apartment%20complex. (visited on 09/12/2020).

[4] Department of Homeland Security, *Improving the operation and development of global positioning system (GPS) equipment used by critical infrastructure*, 2016. [Online]. Available: https://ics-cert.us-cert.gov/sites/default/files/documents/Improving_the_Operation_and_Development_of_Global_Positioning_System_%28GPS%29_Equipment_Used_by_Critical_Infrastructure_S508C.pdf.

[5] T. Ebinuma, *GPS-SDR-SIM*, Sep. 9, 2018. [Online]. Available: https://github.com/osqzss/gps-sdr-sim (visited on 03/17/2021).

[6] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via GPS spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, Jul. 2014, ISSN: 15564959. DOI: 10.1002/rob.21513. [Online]. Available: http://doi.wiley.com/10.1002/rob.21513 (visited on 09/05/2020).

[7] J. Bhatti and T. E. Humphreys, "Hostile control of ships via false GPS signals: Demonstration and detection," *NAVIGATION*, vol. 64, no. 1, pp. 51–66, Mar. 2017, ISSN: 0028-1522, 2161-4296. DOI: 10.1002/navi.183. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/navi.183 (visited on 03/17/2021).

[8] R. Mit, Y. Zangvil, and D. Katalan, "Analyzing tesla's level 2 autonomous driving system under different GNSS spoofing scenarios and implementing connected services for authentication and reliability of GNSS data," presented at the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020), Oct. 28, 2020, pp. 621–646. DOI: 10.33012/2020.17687. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=17687 (visited on 03/17/2021).

[9] S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang, "All your GPS are belong to us: Towards stealthy manipulation of road navigation systems," p. 18,

[10] M. Walter. (). POKEMON GO CHEAT FOOLS GPS WITH SOFTWARE DEFINED RADIO, [Online]. Available: https://hackaday.com/2016/07/19/pokemon-go-cheat-fools-gps-with-software-defined-radio/.

[11] A. Fluerasu, N. Jardak, A. Vervisch-Picois, and N. Samama, "GNSS repeater based approach for indoor positioning: Current status," p. 12,

[12] M. Coulon, A. Chabory, A. Garcia-Pena, J. Vezinet, C. Macabiau, P. Estival, P. Ladoux, and B. Roturier, "Characterization of meaconing and its impact on GNSS receivers," presented at the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020), Oct. 28, 2020, pp. 3713–3737. DOI:

10.33012/2020.17713. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=17713 (visited on 04/21/2021).

[13] J. Fish, *GMCA 641613 white paper*.

[14] Faltech GPS. (). GPS repeaters for police stations, FalTechGPS: Indoor GPS Coverage Solutions, [Online]. Available: https://www.gps-repeaters.com/gps-repeater-applications/gps-repeaters-for-police-stations/ (visited on 05/26/2021).

[15] (). GPS repeaters for bus stations, FalTechGPS: Indoor GPS Coverage Solutions, [Online]. Available: https://www.gps-repeaters.com/gps-repeater-applications/gps-repeater-for-bus-stations/ (visited on 04/26/2021).

[16] Inside GNSS, "Nobody's fool: Spoofing detection in a high-precision receiver," *Inside GNSS*, Jul. 30, 2020. [Online]. Available: https://insidegnss.com/nobodys-fool-spoofing-detection-in-a-high-precision-receiver/ (visited on 08/19/2020).

[17] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, P. M. Kintner, and P. M. Kintner, "Assessing the spoofing threat: Development of a portable GPS civilian spoofer," *Proceedings of ION GNSS, The Institute of Navigation, Savanna, Georgia*, p. 12, 2008.

[18] A. Hennigar, "Analysis of record and playback errors of GPS signals caused by the USRP," Auburn University, Auburn, Alabama, Dec. 13, 2014, 86 pp.

[19] Z. Chen, H. Li, J. Wen, and M. Lu, "Moving receiver PVT solution authentication based on monitoring the combination of clock bias and drift," p. 11,

[20] S. Shang, H. Li, Y. Wei, and M. Lu, "GNSS spoofing detection and identification based on clock drift monitoring using only one signal," presented at the 2020 International Technical Meeting of The Institute of Navigation, San Diego, California, Feb. 13, 2020, pp. 331–340. DOI: 10.33012/2020.17147. [Online]. Available: https://www.

ion.org/publications/abstract.cfm?articleID=17147 (visited on 03/09/2020).

[21] S. Khanafseh, N. Roshan, S. Langel, F.-C. Chan, M. Joerger, and B. Pervan, "GPS spoofing detection using RAIM with INS coupling," p. 8,

[22] Y. Qiao, E. C. N. University, Y. Zhang, E. C. N. University, X. Du, and E. C. N. University, "A vision-based GPS-spoofing detection method for small UAVs," p. 5,

[23] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*, Revised Second Edition. Ganja-Jamuna Press, 2012.

[24] National Coordination Office for Space-Based Positioning, Navigation, and Timing. (Mar. 16, 2021). Space segment, [Online]. Available: https://www.gps.gov/systems/gps/space/ (visited on 03/17/2021).

[25] U.S. Coast Guard Navigation Center. (Mar. 17, 2021). GPS CONSTELLATION STATUS, [Online]. Available: https://www.navcen.uscg.gov/?Do=constellationStatus (visited on 03/17/2021).

[26] USAF. (). GPS III, [Online]. Available: https://www.losangeles.af.mil/About-Us/Fact-Sheets/Article/343728/gps-iii/ (visited on 08/15/2020).

[27] GMV. (2011). CDMA FDMA techniques, [Online]. Available: https://gssc.esa.int/navipedia/index.php/CDMA_FDMA_Techniques.

[28] D. Sessions, "GPS tutorial #2: Signals and messages," [Online]. Available: https://www.theairlinepilots.com/forumarchive/rnav/gps2.pdf.

[29] J. Van Sickle. (). Spread spectrum and code modulation of l1 GPS carrier, PennState College of Earth and Mineral Sciences, [Online]. Available: https://www.e-education.psu.edu/geog862/node/1753.

[30] European GSA. (). Galileo constellation information, European GNSS Service Centre, [Online]. Available: https://www.gsc-europa.eu/system-service-status/constellation-information (visited on 10/05/2020).

[31] S. Blair, "Birth of the european satellite navigation constellation: Galileo in-orbit validation," *ESA Communications*, vol. BR-297, 2011, ISSN: 0250-1589. [Online]. Available: https://www.dlr.de/dlr/en/Portaldata/1/Resources/documents/2011_1/BR-297_Galileo_EN_web.pdf.

[32] R. Constantine, "GPS and galileo: Friendly foes?" Air University Press, Maxwell Air Force Base.

[33] E. GSA, *Galileo-OS-SIS-ICD.pdf*, Dec. 2016.

[34] Swift Navigation, *Piksi datasheet*, Mar. 28, 2016.

[35] ublox, *ZED-f9p*, Jun. 4, 2020.

[36] J. Á. Rodríguez. (2011). Galileo signal plan, [Online]. Available: https://gssc.esa.int/navipedia/index.php/Galileo_Signal_Plan.

[37] GMV. (2011). Correlators, [Online]. Available: https://gssc.esa.int/navipedia/index.php/Correlators.

[38] B. Wiley, *GPS geodetic reference system WGS 84*, Sep. 16, 2009. [Online]. Available: https://www.gps.gov/multimedia/presentations/2009/09/ICG/wiley.pdf.

[39] M. Steer, *Fundamentals of Microwave and RF Design*, Third Edition. NC State University, 2019.

[40] J. S. Warner, "A simple demonstration that the global positioning system (GPS) is vulnerable to spoofing," *The Journal of Security Administration*, p. 9, 2012.

[41] A. Campbell. (). What are the spectrum band designators and bandwidths? NASA, [Online]. Available: https://www.nasa.gov/directorates/heo/scan/communications/outreach/funfacts/txt_band_designators.html.

[42] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A Software-Defined GPS and Galileo Receiver: A Single Frequency Approach*. Birkhauser, 2007.

[43] H. Zumbahlen, *Linear Circuit Design Handbook*. Analog Devices, 2007.

[44] T. Watts, "A GPS and GLONASS L1 vector tracking software-defined receiver," Auburn University, Auburn, Alabama, Feb. 19, 2019, 276 pp.

[45] M. Braasch and A. Dempster, "Tutorial: GPS receiver architectures, front-end and baseband signal processing," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 2, pp. 20–37, Feb. 2019, ISSN: 0885-8985, 1557-959X. DOI: 10.1109/MAES.2018.170117. [Online]. Available: https://ieeexplore.ieee.org/document/8666693/ (visited on 09/27/2020).

[46] M. Petovello, "GNSS solutions: Code tracking & pseudoranges," *Inside GNSS*, vol. January/February 2012, pp. 26–33, 2012.

[47] R. Stengel, *Optimal Control and Estimation*, Dover. Dover Publications, Inc., 1994.

[48] S. Martin, "Closely coupled GPS/INS relative positioning for automated vehicle convoys," Auburn University, May 9, 2011.

[49] G. Welch and G. Bishop, *An introduction to the kalman filter*, 2006.

[50] M. Wickert and C. Siddappa, "Exploring the extended kalman filter for GPS positioning using simulated user and satellite track data," *PROC. OF THE 17th PYTHON IN SCIENCE CONF. (SCIPY 2018)*, 2018.

[51] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, Jun. 2016, ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2016.2526658. [Online]. Available: http://ieeexplore.ieee.org/document/7445815/ (visited on 09/05/2020).

[52] D. Schmidt, K. Radke, S. Camtepe, E. Foo, and M. Ren, "A survey and analysis of the GNSS spoofing threat and countermeasures," *ACM Computing Surveys*, vol. 48, no. 4, pp. 1–31, May 2, 2016, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2897166. [Online]. Available: https://dl.acm.org/doi/10.1145/2897166 (visited on 03/10/2021).

[53] J. R. v. d. Merwe, A. Rügamer, A. Popugaev, X. Zubizarreta, and W. Felber, "Cooperative spoofing attack detection using multiple antennas and a snapshot receiver," presented at the 32nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2019), Miami, Florida, Oct. 11, 2019, pp. 4011–4025. DOI: 10.33012/2019.17112. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=17112 (visited on 02/28/2021).

[54] T. L. Bitner, "Detection and removal of erroneous GPS signals using angle of arrival," Auburn University, Dec. 2013, 103 pp.

[55] Y. Liu, S. Li, Q. Fu, and Z. Liu, "Impact assessment of GNSS spoofing attacks on INS/GNSS integrated navigation system," *Sensors*, vol. 18, no. 5, p. 1433, May 4, 2018, ISSN: 1424-8220. DOI: 10.3390/s18051433. [Online]. Available: http://www.mdpi.com/1424-8220/18/5/1433 (visited on 09/19/2020).

[56] G. Buesnel. (Mar. 12, 2019). Was GPS spoofing responsible for a "weird incident" at the geneva motor show? [Online]. Available: https://www.linkedin.com/pulse/gps-spoofing-responsible-weird-incident-geneva-motor-show-guy-buesnel.

[57] G. W. Hein, T. Pany, S. Wallner, and J.-H. Won, "Platforms for a future GNSS receiver: A discussion of ASIC, FPGA, and DSP technologies," *Inside GNSS*, pp. 56–62, Mar. 2006.

[58] (). About RTL-SDR, RTL-SDR.com, [Online]. Available: https://www.rtl-sdr.com/about-rtl-sdr/.

[59] Ettus Research. (). USRP products, [Online]. Available: https://www.ettus.com/products/.

[60] Ettus Research, *USRP n200/n210 networked series*. [Online]. Available: https://kb.ettus.com/About_USRP_Bandwidths_and_Sampling_Rates#:~:text=The%20FPGA%20processing%20bandwidth%20is,the%20DACs%20and%20ADCs%20respectively. (visited on 04/25/2021).

[61] Ettus Research. (May 2016). About USRP bandwidths and sampling rates.

[62] R. Di, Y. Morton, and E. Vinande, "Performance analysis of a USRP based GPS and GLONASS signal recording and playback system," *San Diego*, p. 10, 2013.

[63] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2008.

[64] A. Thopay, "Initialization of a pedestrian navigation system using a transfer alignment approach," Auburn University, Auburn, Alabama, 2020, 190 pp.

[65] J. A. Farrell, *Aided Navigation: GPS with High Rate Sensors*. McGraw Hill, 2008, 553 pp.

[66] Journey North. (2019). Understanding latitude and longitude, [Online]. Available: https://journeynorth.org/tm/LongitudeIntro.html (visited on 03/21/2021).

[67] Mathworks. (). MathWorks help center - move, [Online]. Available: https://www.mathworks.com/help/supportpkg/ryzeio/ref/move.html.

[68] Kelly Blue Book. (). 2021 honda pilot.

[69] A. El-Osery, K. Wedeward, and S. Bruder, "EE 570 location and navigation: Sensor technology," New Mexico Tech.

[70] A. El-Osery, K. Wedeward, and S. Bruder, "EE 570 location and navigation: Navigation equations: ECEF mechanization," New Mexico Tech.

[71] G. T. Schmidt and R. E. Phillips, "INS/GPS integration architectures," p. 19, 2010.

[72] X. Lu, Z. Dou, Y. Lilu, and L. Chen, "Ancient china navigation - from compass to BeiDou," presented at the 2020 International Technical Meeting of The Institute of Navigation, San Diego, California, Feb. 13, 2020, pp. 71–115. DOI: 10.33012/2020.17205. [Online]. Available: https://www.ion.org/publications/abstract.cfm?articleID=17205 (visited on 03/20/2021).

[73] D. B. Sullivan, *Time and frequency at NIST.pdf*, Jun. 8, 2001.

[74] R. Lutwak, "Atomic clocks tutorial," Proceedings of the 50th Annual Precise Time and Time Interval Systems and Applications Meeting, Reston, VA, Jan. 2019.

[75] Microsemi. Inc., *DS1000 OCXO data sheet*, 2014.

[76] P. Cash, W. Krzewick, P. Machado, K. R. Overstreet, M. Silveira, M. Stanczyk, D. Taylor, and X. Zhang, "Microsemi chip scale atomic clock (CSAC) technical status, applications, and future plans," in *2018 European Frequency and Time Forum (EFTF)*, Turin: IEEE, Apr. 2018, pp. 65–71, ISBN: 978-1-5386-5620-4. DOI: 10.1109/EFTF.2018.8408999. [Online]. Available: https://ieeexplore.ieee.org/document/8408999/ (visited on 07/11/2020).

[77] Microsemi. Inc., *CSAC user guide*, 2017.

[78] A. T. Gardner and J. A. Collins, "Advancements in high-performance timing for long term underwater experiments: A comparison of chip scale atomic clocks to traditional microprocessor-compensated crystal oscillators," in *2012 Oceans*, Hampton Roads, VA: IEEE, Oct. 2012, pp. 1–8. DOI: 10.1109/OCEANS.2012.6404847. [Online]. Available: http://ieeexplore.ieee.org/document/6404847/ (visited on 07/11/2020).

[79] A. Wells, "FairfieldNodal seismic data collection advances oil and gas," *Boss Magazine*, pp. 58–71, Apr. 2018. [Online]. Available: https://thebossmagazine.com/fairfieldnodal/ (visited on 07/11/2020).

[80] R. Ramlall, J. Streter, and J. Schnecker, "Three satellite navigation in an urban canyon using a chip-scale atomic clock," in *Proceedings of the 24th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2011)*, Portland, OR: ION, Sep. 2011.

[81] G. Weaver, "The application of frequency stability analysis and the use of time domain statistics for clock and oscillator performance assessment," Proceedings of the 50th Annual Precise Time and Time Interval Systems and Applications Meeting, Reston, VA, Jan. 2019.

[82] W. J. Riley, *Handbook of frequency stability analysis*, Jul. 2008.

[83] IEEE Standards Coordinating Committee 27, "IEEE standard definitions of physical quantities for fundamental frequency and time metrology—random instabilities," IEEE, 1139, ISBN: 9780738168555 9780738168562. DOI: 10.1109/IEEESTD.2008.4797525. [Online]. Available: http://ieeexplore.ieee.org/document/4797525/ (visited on 04/10/2021).

[84] D. Sullivan, D. Allan, D. Howe, and F. Walls, *Characterization of clocks and oscillators*, 1990.

[85] D. Howe, D. Allan, and J. Barnes. (). Analysis of time domain data, NIST Time and Frequency Division, [Online]. Available: https://tf.nist.gov/phase/Properties/four.htm.

[86] W. Riley. (). Test suite data, [Online]. Available: https://www.wriley.com/tst_suit.dat (visited on 03/21/2021).

[87] W. Riley. (). NBS data set, [Online]. Available: https://www.wriley.com/nbs.dat.

[88] W. J. Riley, *Stable32 - software for frequency stability analysis*. [Online]. Available: https://ieee-uffc.org/frequency-control/frequency-control-software/stable32/.

[89] J. Miles, *TimeLab*, version 1.51, Jun. 2020. [Online]. Available: http://www.ke5fx.com/timelab/readme.htm.

[90] J. A. Barnes, A. R. Chi, L. S. Cutler, D. J. Healey, D. B. Leeson, T. E. McGunigal, J. A. Mullen, W. L. Smith, R. L. Sydnor, R. F. C. Vessot, and G. M. R. Winkler, "Characterization of frequency stability," *IEEE Transactions on Instrumentation and Measurement*, vol. IM-20, no. 2, pp. 105–120, May 1971, ISSN: 0018-9456, 1557-9662. DOI: 10.1109/TIM.1971.5570702. [Online]. Available: http://ieeexplore.ieee.org/document/5570702/ (visited on 03/21/2021).

[91] K. Technologies, *Keysight 53230a user guide*, Sep. 2017.

[92] Keysight Technologies, *Analyzing frequency stability in the frequency and time domains - application note*, 2014.

[93] R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, Fourth. John Wiley & Sons, 2012.

[94] Microsemi. Inc., *Microsemi CSAC performance during rapid temperature change*, 2018.

[95] L. Galleani, "A tutorial on the two-state model of the atomic clock noise," *Metrologia*, vol. 45, no. 6, S175–S182, Dec. 2008, ISSN: 0026-1394, 1681-7575. DOI: 10.1088/0026-1394/45/6/S23. [Online]. Available: https://iopscience.iop.org/article/10.1088/0026-1394/45/6/S23 (visited on 03/21/2021).

[96] Z. Yang, *Brownian motion simulation project in r*. [Online]. Available: https://www.stat.berkeley.edu/~aldous/Research/Ugrad/ZY1.pdf.

[97] J. Barnes and S. Jarvis Jr., *Efficient numerical and analog modeling of flicker noise processes*, Jun. 1971.

[98] C. A. Greenhall, "Initializing a flicker-noise generator," in *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. IM-35, NO.2*, IEEE, Jun. 1986, pp. 222–224.

[99] Nuand, *bladeRF x40 datasheet*. [Online]. Available: https://www.nuand.com/bladeRF-brief.pdf.

[100] J. Edgcombe. (Feb. 2019). HackRF wiki, [Online]. Available: https://github.com/mossmann/hackrf/wiki.

[101] J. S. Subirana, J. M. J. Zornoza, and M. Hernández-Pajares, *GNSS Data Processing: Fundamentals and Algorithms*, 2 vols. ESA Communications, May 2013, vol. 1, 238 pp.

[102] C. Fernández-Prades, J. Arribas, L. Esteve, P. Closas, M. Majoral, Jordi Vilà-Valls, Álvaro Cebrián Juan, and Damian Miralles, *GNSS-SDR*. [Online]. Available: https://gnss-sdr.org/.

[103] *AirSim*. [Online]. Available: https://microsoft.github.io/AirSim/.

[104] S. Thompson, *Airsim modifications for IMU TruthOutput*, 2020.

[105] Aceinna, *GNSS-INS-sim*, 2019. [Online]. Available: https://github.com/Aceinna/gnss-ins-sim.

[106] NASA. (). NASA CDDIS EarthData repository, [Online]. Available: https://urs.earthdata.nasa.gov/.

[107] R. Wang, "Classification: Performance measurements," HMC E161 Course Notes, HMC E161 Course Notes, Apr. 4, 2016, [Online]. Available: http://fourier.eng.hmc.edu/e161/lectures/classification/node5.html.

[108] M. H. Hayes, *Schaum's Outline of Theory and Problems of Digital Signal Processing*, ser. Schaum's Outline Series. McGraw-Hill, 1999.

[109] M. Jones, "Spoofing in the black sea: What really happened?" *GPS World*, Oct. 11, 2020. [Online]. Available: https://www.gpsworld.com/spoofing-in-the-black-sea-what-really-happened/ (visited on 09/02/2020).

[110] W. Kester, "Converting oscillator phase noise to time jitter," p. 10,

Appendix A

Oscillator Stability Analysis

This Appendix provides additional detail on frequency stability analysis methods. Additionally, domain conversion methods are discussed, which allow the derivation of the necessary power law coefficients from phase noise specifications if Allan variance parameters are unknown or unavailable to the user.

A.1 Frequency Domain Stability

In addition to methods of describing stability in terms of the time domain as was shown in Chapter 4, stability can also be described in the frequency domain. This measure is typically represented as single sideband (SSB) phase noise. Shown in Figure A.1, phase noise is the uncertainty in the zero crossing of the sinusoidal signal generated by the oscillator. This measure is defined as the power density at a given frequency offset from the center frequency relative to the total power of the carrier [92].

Figure A.1: Ideal and Noisy Oscillator Output [92]

In an ideal world, the signal generated by an oscillator would have an exact center frequency and no noise. Since there is no such thing as ideal, variations in the frequency of the

signal cause sidebands in the signal, or areas where the center frequency fluctuates. An example of a SSB phase noise plot is shown in Figure A.2. Like the ADEV plot, several distinct slopes can be identified. These slopes are related to each type of noise process present in the oscillator, similar to the slopes on the ADEV curve. The better the oscillator, the smaller the values of the SSB phase noise will be indicating that the frequency of the oscillator remains closer to the center frequency instead of deviating.

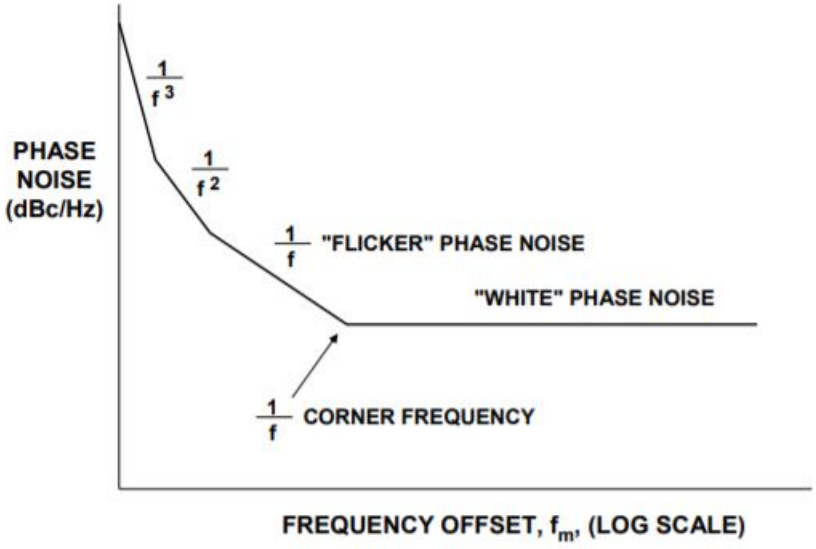

Figure A.2: Single Sideband Phase Noise Plot [110]

The value of the phase noise is important to designers of DSP hardware as the sampling process consists of mixing the incoming RF signal with the local oscillator signal. This mixing, a simple multiplication in the time domain, equates to a convolution in the frequency domain [110]. The convolution of the spectrum of the phase noise can smear the input signal across the bandwidth of the phase noise, degrading the signal resolution that is digitized. This effect is shown in Figure A.3 [110].

When discussing high speed analog to digital converters (ADCs), the value that is most frequently discussed is clock jitter. The jitter in the oscillator signal creates problems for the ADC as it can cause samples to be taken too early or too late, adding noise to the data that is being collected. Phase noise can be integrated across specified bandwidths to determine the jitter of the oscillator. Figure A.4 provides an example of a SSB phase noise plot with each section clearly marked. To calculate the RMS jitter of the oscillator, the area of each section

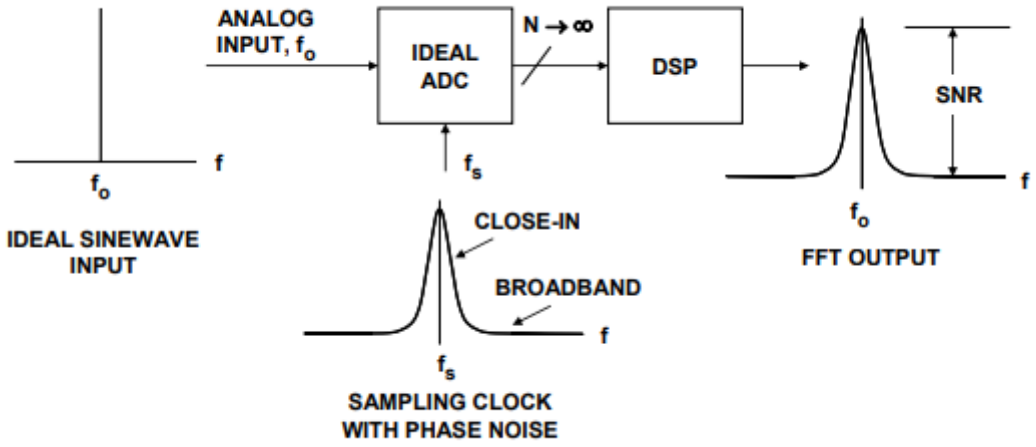

Figure A.3: Effect of Phase Noise on Sampled Signal [110]

(A_i) must be calculated, then converted to seconds using the method shown in Equation (A.1) [110].

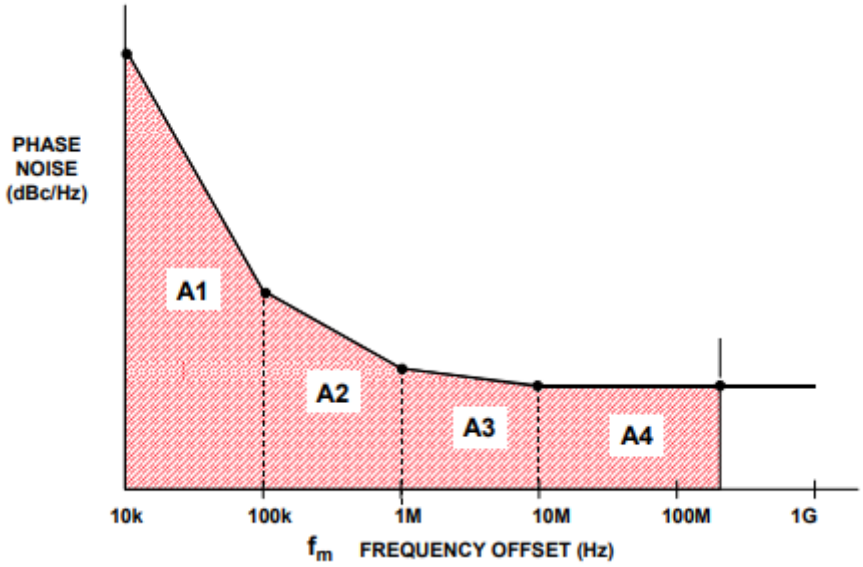

Figure A.4: SSB Phase Noise Curve Areas [110]

$$\sigma_{RMS} = \sum_{i=1}^{n} \frac{\sqrt{2 \times 10^{A_i/10}}}{2\pi f_0} \tag{A.1}$$

Oscillator jitter is an important measure in terms of ADCs as the higher the jitter, the lower the signal to noise ratio (SNR) can be. Equation (A.2) shows the limit on the SNR in decibels relative to full scale (dBFS).

$$SNR = -20\log(2\pi f_{in}\sigma_{RMS}) \tag{A.2}$$

To allow for the best possible SNR in a digital system, an oscillator which low jitter should be used. This maximizes the performance of the system and does not limit the performance of the other components unnecessarily.

A.2 Conversion Between Time and Frequency Domain

In some scenarios, oscillator documentation omits the Allan deviation or only has limited sample points. Without enough points along the full Allan deviation curve, it can be difficult or impossible to calculate the power law coefficients that are used in the two state model discussed in Chapter 4. Many data sheets, however, include information on the SSB phase noise, a frequency domain stability measure described above. The phase noise can be related to the power law model in the same way the Allan deviation can, albeit with a few more steps in between. The simplest method is to directly convert the SSB phase noise values, given as $S_\phi(f)$, to the Allan variance values, given as $\sigma_y^2(\tau)$, and relate the Allan variance values to the power law as was shown previously.

To convert the phase noise values to Allan variance values, the equations shown in Table A.1 are used. These equations come from [84], which provides additional background into the conversion between domains as well as a wealth of other information related to oscillators. To convert from the phase noise to Allan variance, the b coefficients shown in the third column are used, with τ representing the averaging time of interest. The second column relates the Allan variance to the power law in the reverse of what was shown in Chapter 4. Previously, the relation was given as $\sigma_y^2(\tau) = \frac{S_y(f)}{a}$, whereas here it is given as $S_y(f) = a\sigma_y^2(\tau)$, where $S_y(f)$ denotes the power law model for oscillators.

In the table above, ν_0 represents the nominal frequency of the oscillator and f_h and ω_h represent the high frequency cutoff in either Hz or rad/s, respectively. The high frequency cutoff can be determined from the phase noise plot, and represents the highest frequency of interest on the plot. For example, Figure A.5 gives a cutoff frequency of $10^4 Hz$. To calculate

Table A.1: Coefficients for Domain Conversion of Clock Noises

$S_y(f) = h_\alpha f^\alpha$ $\alpha =$	$S_y(f) = a\sigma_y^2(\tau)$ $a =$	$\sigma_y^2(\tau) = bS_\phi(f)$ $b =$
2 (white phase)	$\frac{(2\pi)^2\tau^2 f^2}{3f_h}$	$\frac{3f_h}{(2\pi)^2\tau^2\nu_0^2}$
1 (flicker phase)	$\frac{2\pi^2\tau^2 f}{1.038+3\ln(2\pi f_h\tau)}$	$\frac{1.038+3\ln(\omega_h\tau)f}{(2\pi)^2\tau^2\nu_0^2}$
0 (white frequency)	2τ	$\frac{f^2}{22\tau\nu_0^2}$
-1 (flicker frequency)	$\frac{1}{2\ln(2)f}$	$\frac{2\ln(2)f^3}{\nu_0^2}$
-2 (random walk frequency)	$\frac{6}{2\pi^2\tau f^2}$	$\frac{(2\pi)^2\tau f^4}{6\nu_0^2}$

the Allan variance from the phase noise plot shown, the slopes on the phase noise plot need to be related to the given noise types, which is shown in Figure A.2 above. For this example, there is a section with a slope of $\frac{1}{f^3}$ from $10Hz$ to $100Hz$, and a slope of f^0 from $100Hz$ to $10kHz$, which is the high frequency cutoff point.

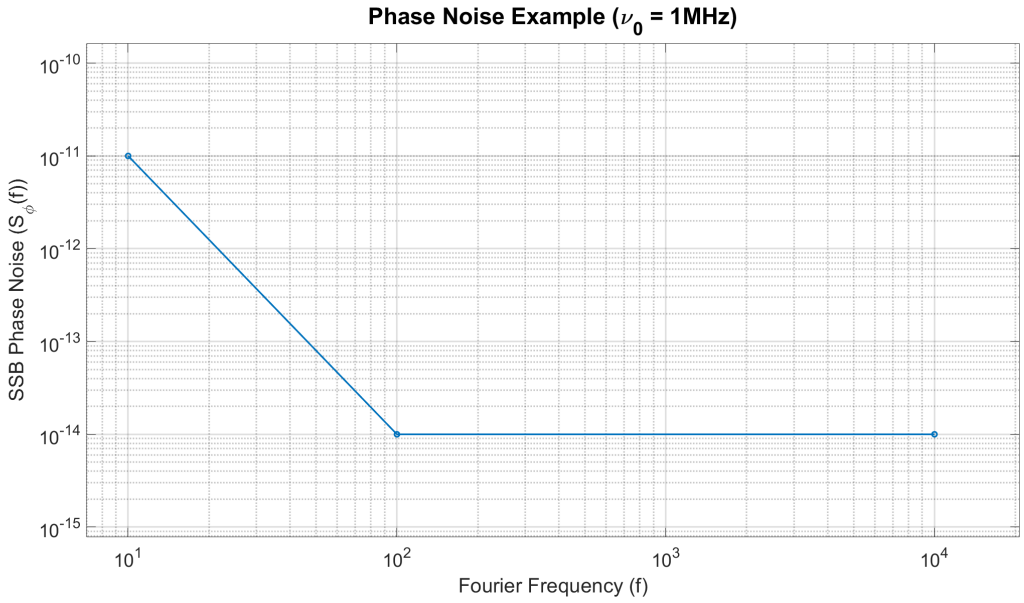

Figure A.5: Phase Noise Example

To calculate the Allan variance, the appropriate conversion coefficients are taken from Table A.1. In this example, flicker frequency and white phase noise are present, so those are

the coefficients used to calculate the Allan variance. The value of f and $S_\phi(f)$ are chosen at a point that lies along the slope for the appropriate noise type. For the flicker noise, a value of $S\phi(f) = 10^{-11}$ at $f = 10Hz$ is used. The calculation of the Allan variance is shown in Equation (A.3).

$$\sigma_y^2(\tau) = \frac{2\ln(2)f^2}{\nu_0^2} S_y(f) = \frac{2\ln(2)(10^2)}{(1e6)^2}(10^{-11}) = 1.39e - 20 \tag{A.3}$$

The same process can be followed for the white phase noise using the second line from the table. For this calculation, a phase noise value of $S_y(f) = 10^-14$ at $f = 10^4$ is used. The calculation is shown in Equation (A.4) below. For this calculation, the value of τ is not specified but rather left a variable. To get the value of the Allan variance at a given averaging time, the desired τ value can be substituted, or a range of averaging times can be used to generate the curve and determine where it intersects the flicker noise Allan variance calculated previously.

$$\sigma_y^2(\tau) = \frac{3f_h}{(2\pi)^2\tau^2\nu_0^2} = \frac{3(10^4)}{(2\pi)^2(1e6)^2\tau^2}(10^-14) = 7.59e - 24\frac{1}{\tau^2} \tag{A.4}$$

The resulting Allan deviation curve from the calculated values is shown in Figure A.6. As expected, the white phase noise has a slope related to $\frac{1}{2\tau}$ and the flicker noise has a slope of zero. The total Allan deviation is a combination of the two noise processes plotted, with the final curve reflecting both noise types present.

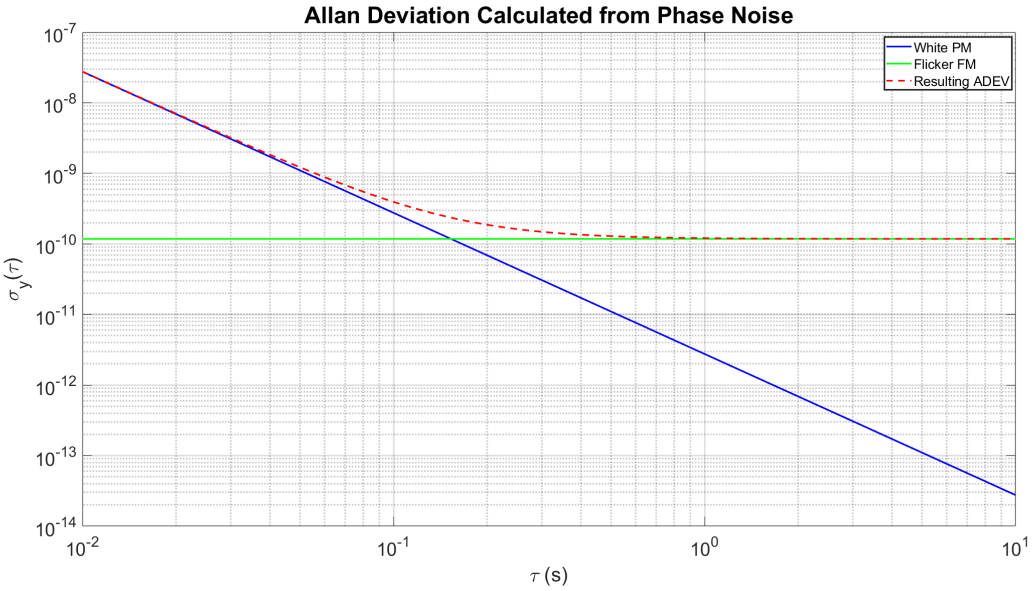

Figure A.6: Allan Deviation from Phase Noise

Appendix B

Additional Monte Carlo Results

This appendix contains the TPR plots and tabulated $\%M$ results for all Monte Carlo simulations shown in this thesis. Each section shows the TPR bar plot and mitigation percentage tables for the Monte Carlo of the same name shown in Chapter 6. The TPR for all but the dynamic spoofing scenario was 100% and the FPR for all scenarios was 0%. The only exception to the TPR was the dynamic spoofing scenario, where the TPR when the transmitter was driven by a high quality oscillator (OCXO, Rb) was equal to 0%. For this case, the TPR is shown both in this appendix and the results shown in Chapter 6, as it provides additional insight into the performance of the algorithm when dealing with a more sophisticated transmitter configuration.

B.1 Meaconer - Full Capture

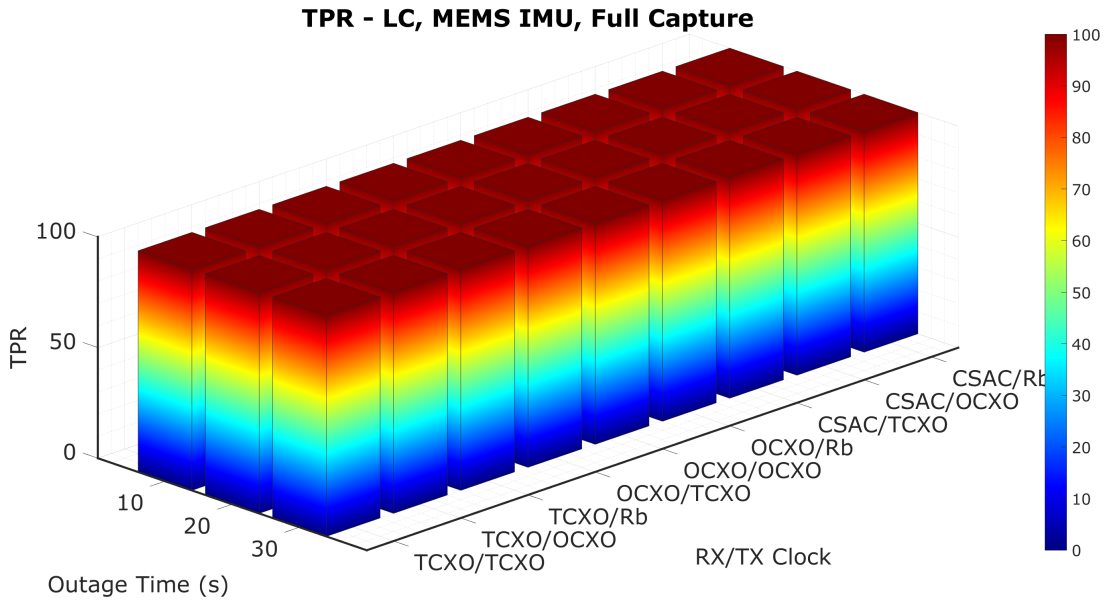

Figure B.1: TPR - Full Capture, Loosely Coupled, MEMS IMU

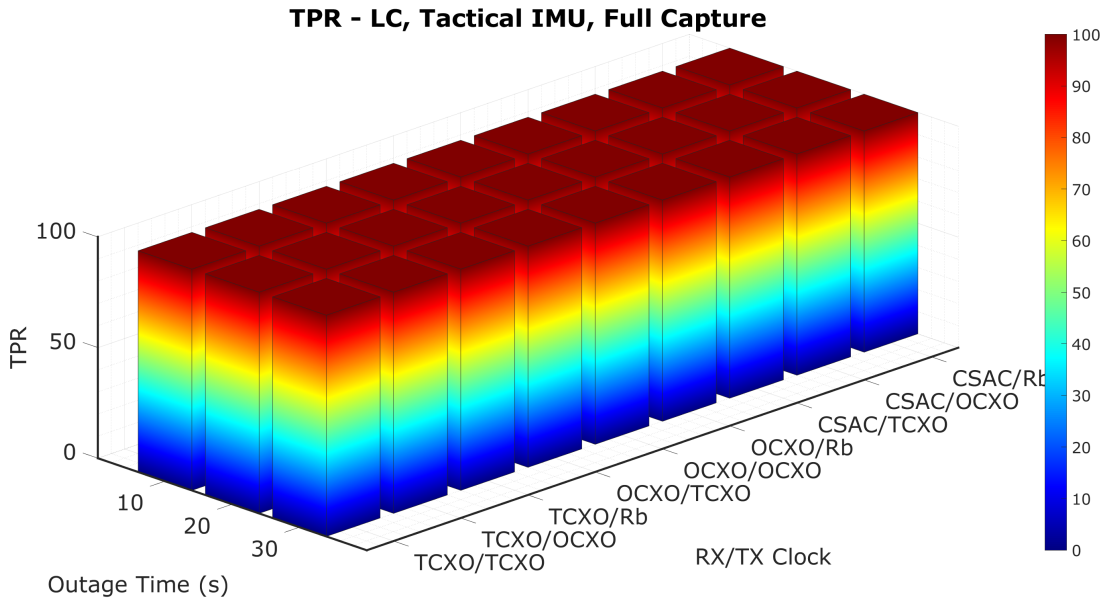

Figure B.2: TPR - Full Capture, Loosely Coupled, Tactical IMU

LC, MEMS, Full Capture			
	10	20	30
TCXO/TCXO	100.00%	100.00%	98.47%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	100.00%	98.43%	94.61%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	99.21%	99.09%	93.91%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(a) MEMS IMU

LC, Tactical, Full Capture			
	10	20	30
TCXO/TCXO	96.72%	97.73%	97.74%
TCXO/OCXO	0.00%	3.22%	0.00%
TCXO/Rb	0.00%	2.20%	0.00%
OCXO/TCXO	98.44%	99.27%	97.01%
OCXO/OCXO	0.00%	4.80%	0.00%
OCXO/Rb	0.00%	0.77%	0.00%
CSAC/TCXO	99.24%	99.24%	99.21%
CSAC/OCXO	0.00%	3.44%	0.00%
CSAC/Rb	0.00%	3.47%	0.00%

(b) Tactical IMU

Figure B.3: Full Capture - Loose Coupling

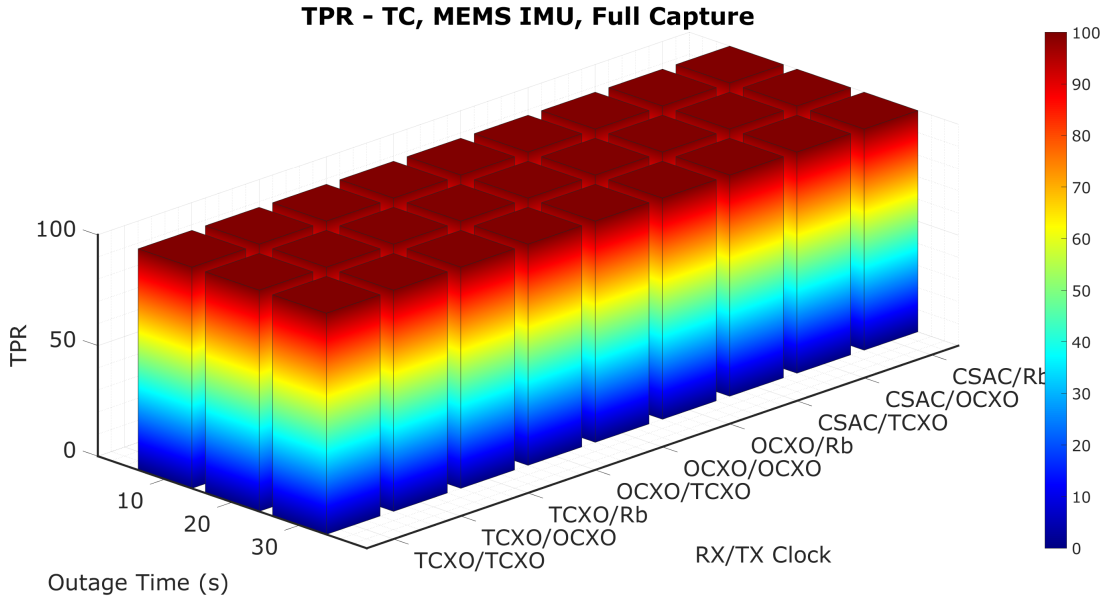

Figure B.4: TPR - Full Capture, Tightly Coupled, MEMS IMU

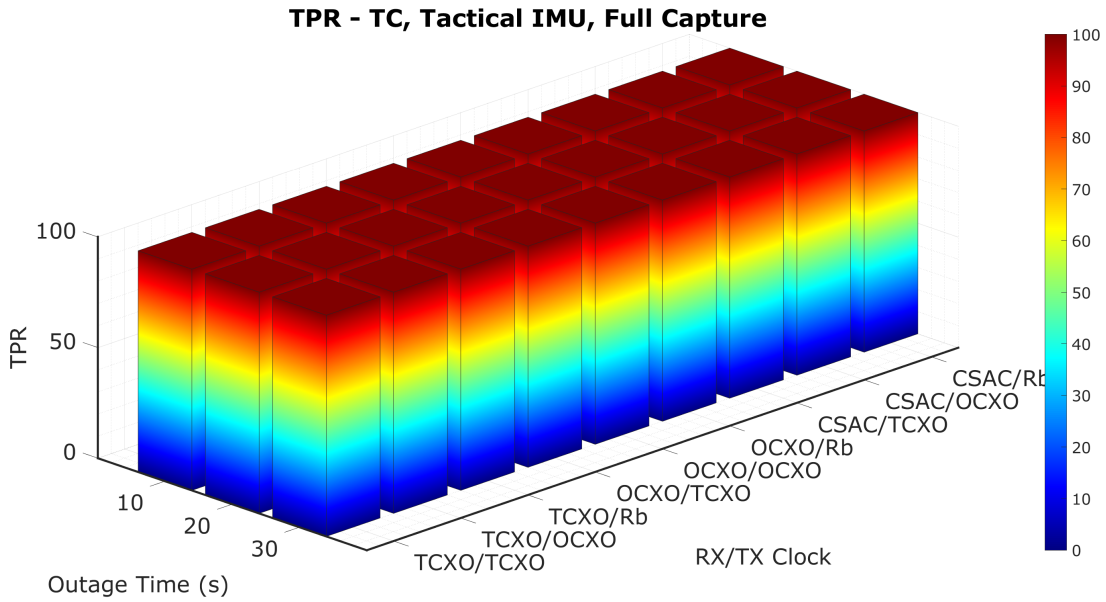

Figure B.5: TPR - Full Capture, Tightly Coupled, Tactical IMU

TC, MEMS, Full Capture			
	10	20	30
TCXO/TCXO	100.00%	99.24%	97.17%
TCXO/OCXO	83.09%	22.13%	8.53%
TCXO/Rb	92.74%	27.33%	7.58%
OCXO/TCXO	100.00%	100.00%	100.00%
OCXO/OCXO	89.43%	20.49%	4.96%
OCXO/Rb	88.19%	23.21%	6.30%
CSAC/TCXO	100.00%	100.00%	97.12%
CSAC/OCXO	90.71%	14.84%	0.86%
CSAC/Rb	94.41%	19.09%	5.88%

(a) MEMS IMU

TC, Tactical, Full Capture			
	10	20	30
TCXO/TCXO	99.19%	100.00%	98.36%
TCXO/OCXO	90.76%	43.07%	24.81%
TCXO/Rb	96.21%	42.44%	17.46%
OCXO/TCXO	100.00%	98.43%	99.18%
OCXO/OCXO	92.13%	41.96%	7.89%
OCXO/Rb	95.20%	27.64%	10.53%
CSAC/TCXO	100.00%	100.00%	98.44%
CSAC/OCXO	94.31%	34.09%	9.52%
CSAC/Rb	95.12%	33.60%	9.60%

(b) Tactical IMU

Figure B.6: Full Capture - Tight Coupling Coupling

B.2 Meaconer - Partial Capture

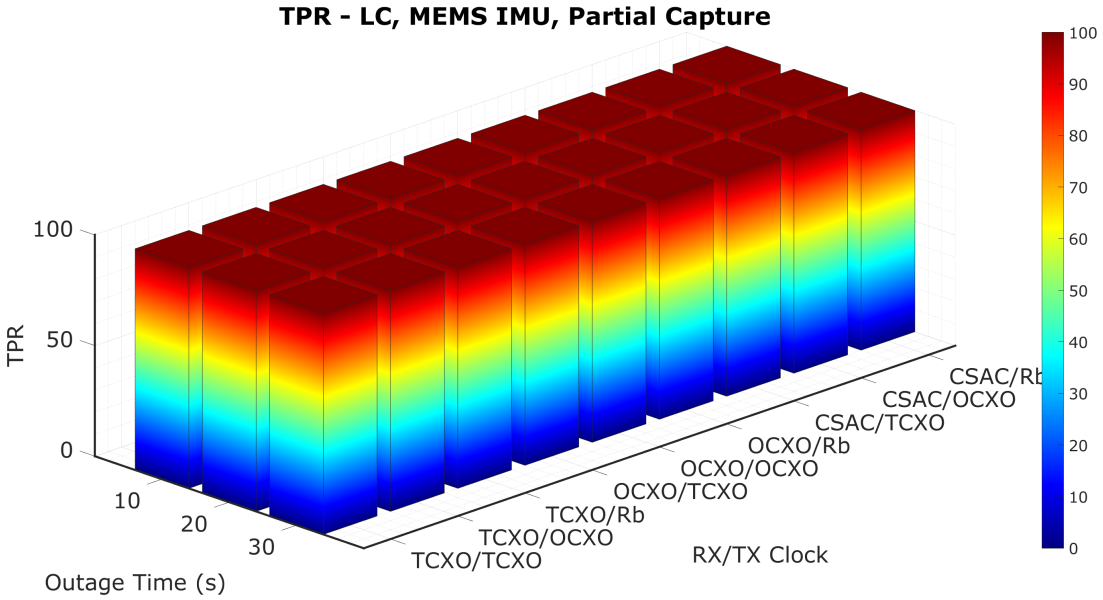

Figure B.7: TPR - Partial Capture, Loosely Coupled, MEMS IMU

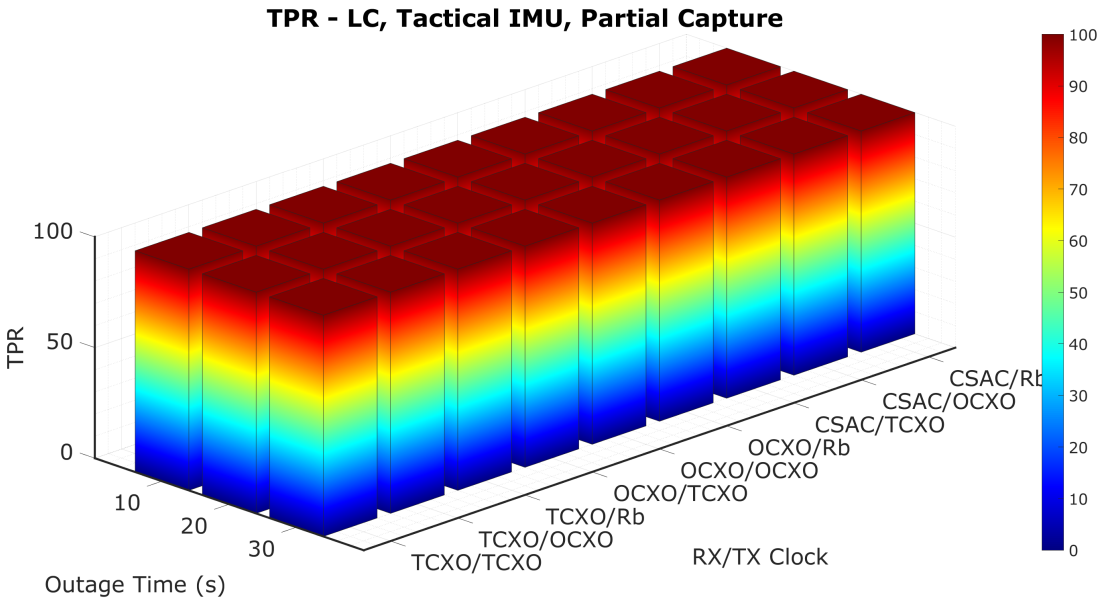

Figure B.8: TPR - Partial Capture, Loosely Coupled, Tactical IMU

LC, MEMS, Partial Capture			
	10	20	30
TCXO/TCXO	45.74%	52.42%	45.52%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	52.07%	46.96%	38.05%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	47.79%	49.61%	45.60%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(a) MEMS IMU

LC, Tactical, Partial Capture			
	10	20	30
TCXO/TCXO	51.82%	51.49%	45.69%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	49.13%	52.54%	49.59%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	55.00%	45.80%	53.17%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(b) Tactical IMU

Figure B.9: Partial Capture - Loose Coupling

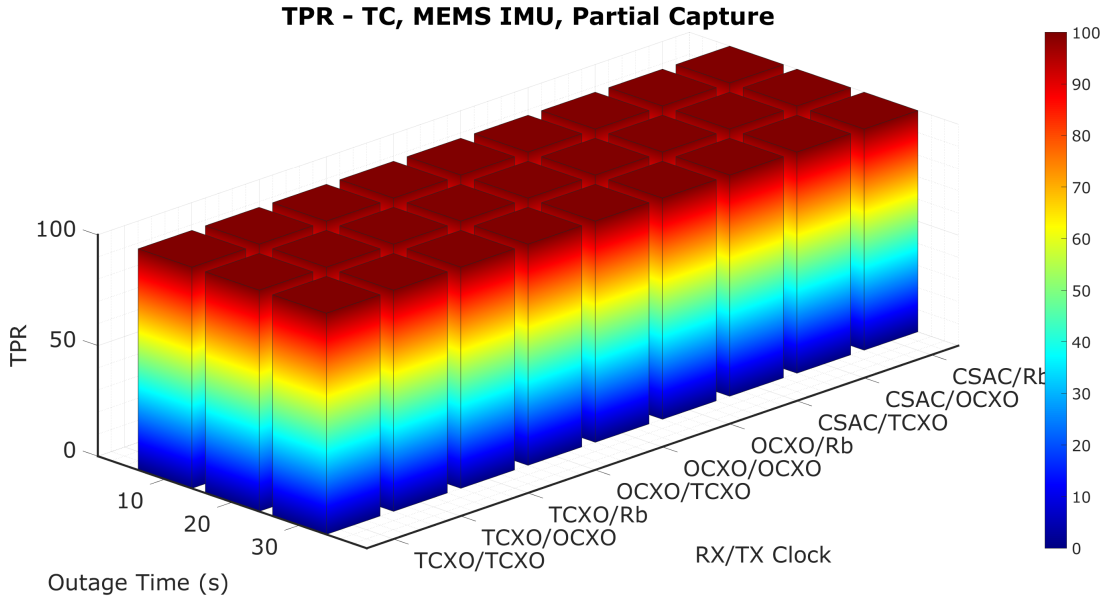

Figure B.10: TPR - Partial Capture, Tightly Coupled, MEMS IMU

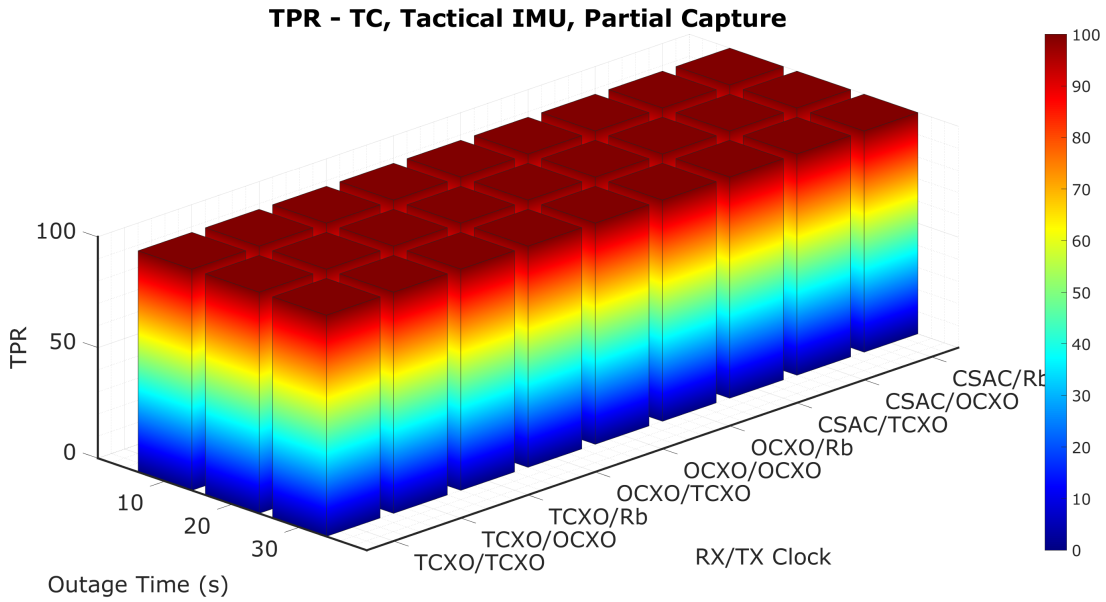

Figure B.11: TPR - Partial Capture, Tightly Coupled, Tactical IMU

TC, MEMS, Partial Capture			
	10	20	30
TCXO/TCXO	51.61%	47.06%	46.92%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	48.46%	55.72%	48.18%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	50.79%	53.23%	40.60%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(a) MEMS IMU

TC, Tactical, Partial Capture			
	10	20	30
TCXO/TCXO	50.38%	49.25%	48.38%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	43.07%	50.44%	40.83%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	50.00%	51.69%	49.55%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(b) Tactical IMU

Figure B.12: Partial Capture - Tight Coupling

B.3 Spoofer - Static Position

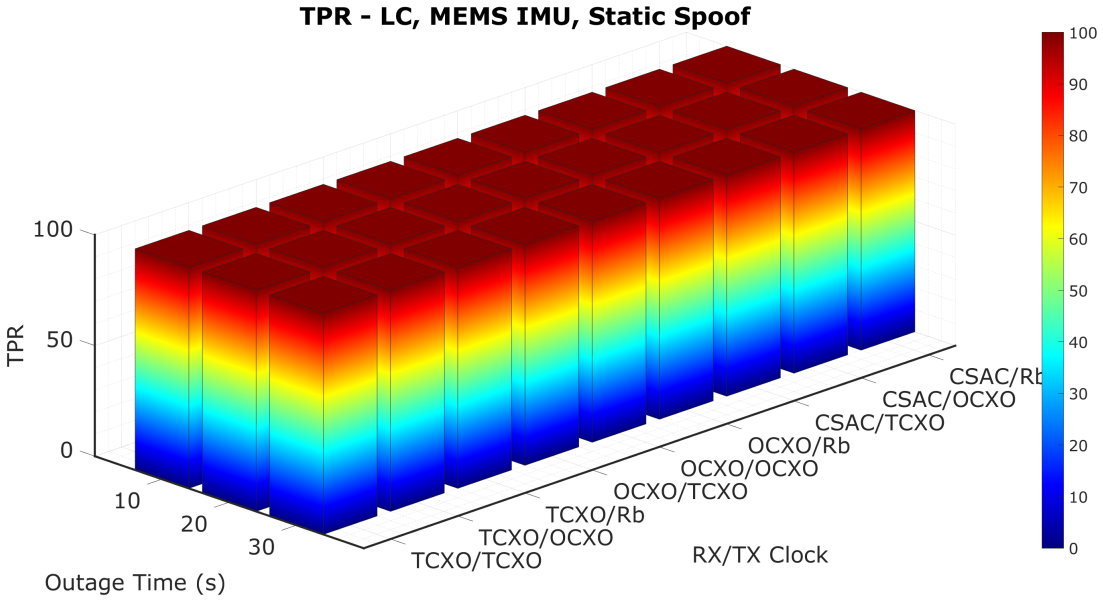

Figure B.13: TPR - LC, MEMS IMU, Static Position Spoof

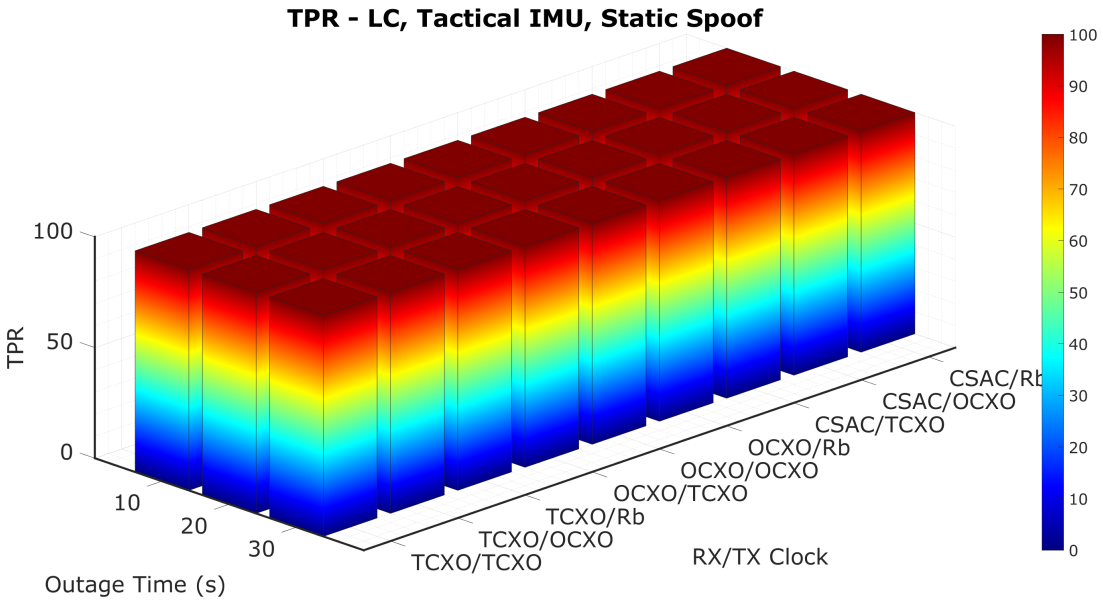

Figure B.14: TPR - LC, Tactical IMU, Static Position Spoof

LC, MEMS, Static Spoof Location			
	10	20	30
TCXO/TCXO	93.60%	83.33%	68.75%
TCXO/OCXO	0.00%	0.81%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	98.50%	91.94%	74.26%
OCXO/OCXO	0.85%	0.00%	0.00%
OCXO/Rb	0.83%	1.56%	0.74%
CSAC/TCXO	96.03%	83.61%	71.82%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(a) MEMS IMU

LC, Tactical, Static Spoof Location			
	10	20	30
TCXO/TCXO	96.83%	95.24%	81.45%
TCXO/OCXO	27.27%	0.00%	0.00%
TCXO/Rb	36.76%	0.00%	0.00%
OCXO/TCXO	96.87%	95.42%	93.44%
OCXO/OCXO	45.71%	0.69%	0.00%
OCXO/Rb	61.11%	0.00%	0.00%
CSAC/TCXO	100.00%	98.39%	90.84%
CSAC/OCXO	53.45%	2.50%	0.00%
CSAC/Rb	67.18%	0.00%	0.00%

(b) Tactical IMU

Figure B.15: Static Spoof - Loose Coupling

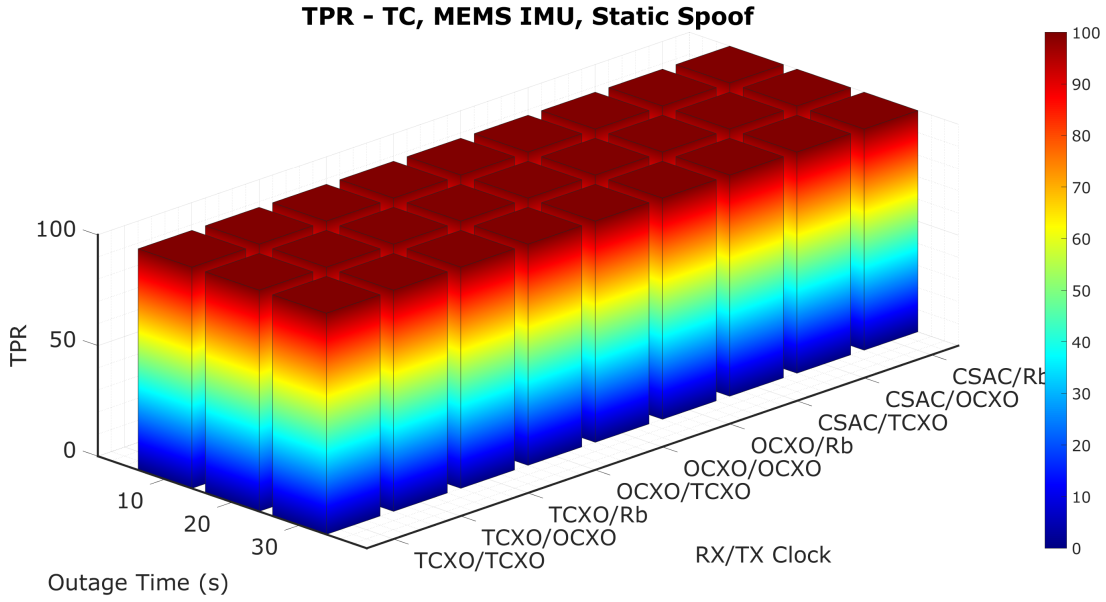

Figure B.16: TPR - TC, MEMS IMU, Static Position Spoof

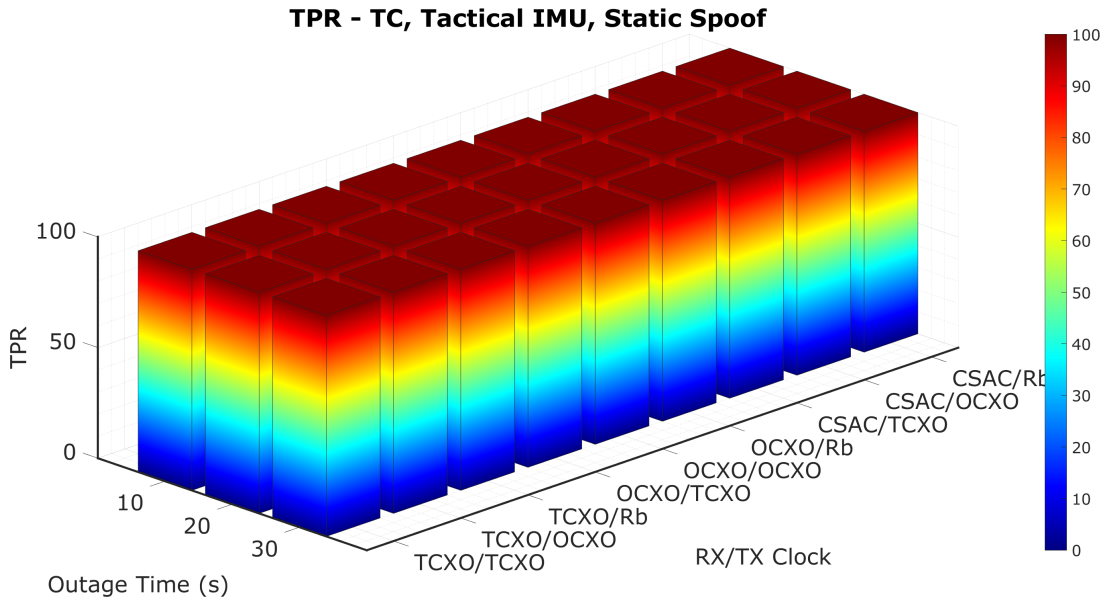

Figure B.17: TPR - TC, Tactical IMU, Static Position Spoof

TC, MEMS, Static Spoof Location			
	10	20	30
TCXO/TCXO	94.35%	88.98%	66.41%
TCXO/OCXO	0.86%	0.83%	0.00%
TCXO/Rb	0.00%	0.00%	0.83%
OCXO/TCXO	98.58%	89.23%	68.55%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	2.50%	1.60%	0.00%
CSAC/TCXO	97.48%	90.63%	71.19%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	1.64%	1.57%	0.00%

(a) MEMS IMU

TC, Tactical, Static Spoof Location			
	10	20	30
TCXO/TCXO	95.80%	93.39%	87.12%
TCXO/OCXO	0.86%	0.82%	0.00%
TCXO/Rb	3.10%	0.00%	0.00%
OCXO/TCXO	95.24%	92.37%	81.10%
OCXO/OCXO	0.00%	0.76%	0.00%
OCXO/Rb	0.00%	0.88%	0.00%
CSAC/TCXO	98.21%	96.21%	84.51%
CSAC/OCXO	2.31%	0.00%	0.00%
CSAC/Rb	0.77%	2.12%	0.00%

(b) Tactical IMU

Figure B.18: Static Spoof - Tight Coupling

B.4 Spoofer - Dynamic Position

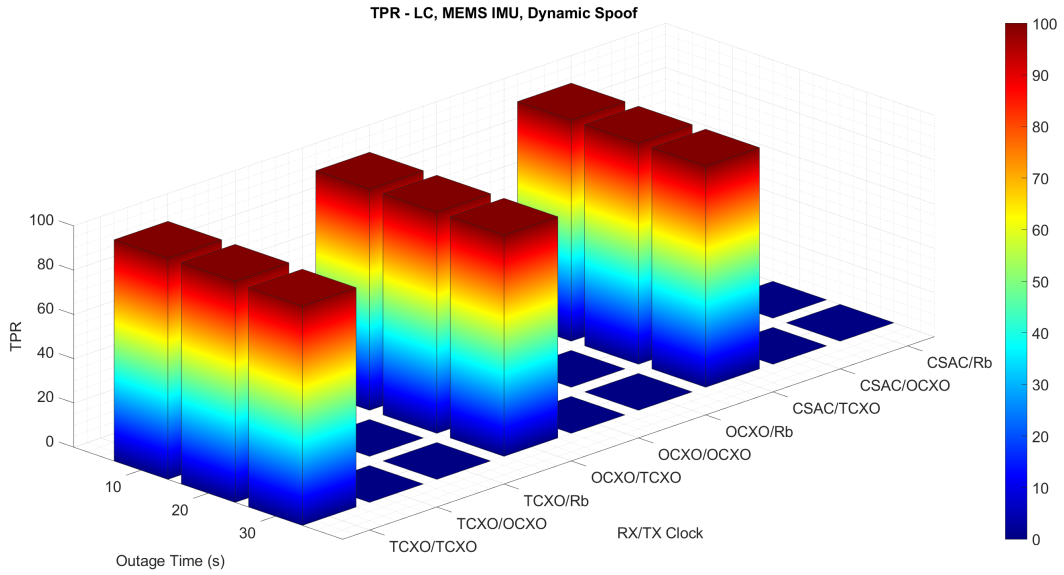

Figure B.19: TPR - LC, MEMS IMU, Spoofed Trajectory Matched to Truth

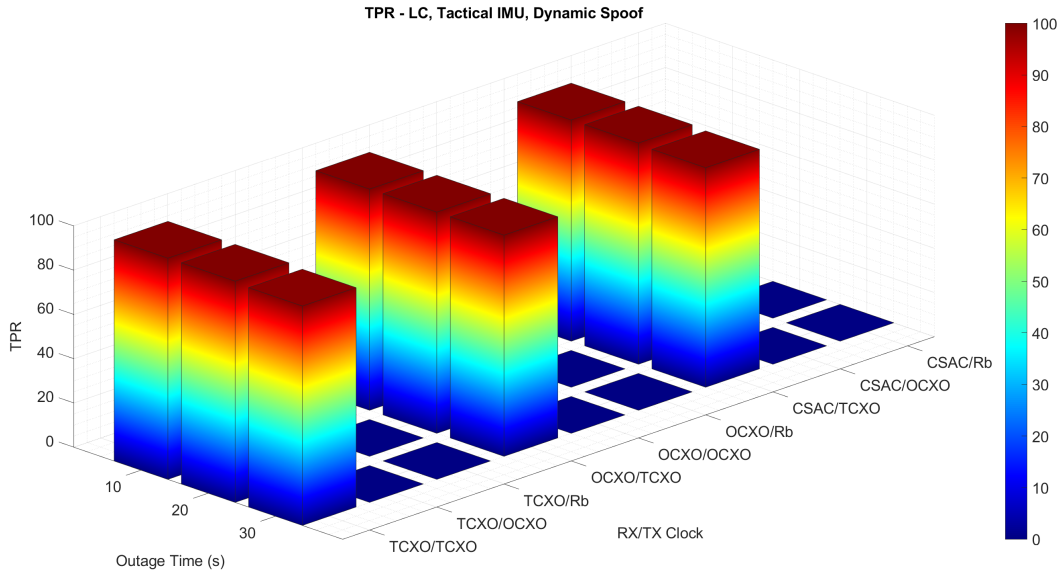

Figure B.20: TPR - LC, Tactical IMU, Spoofed Trajectory Matched to Truth

LC, MEMS, Dynamic Spoof Location			
	10	20	30
TCXO/TCXO	94.21%	91.87%	74.17%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	95.45%	80.73%	77.94%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	95.12%	84.00%	71.30%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(a) MEMS IMU

LC, Tactical, Dynamic Spoof Location			
	10	20	30
TCXO/TCXO	93.13%	94.87%	90.00%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	93.60%	92.31%	88.98%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	94.57%	96.52%	92.42%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(b) Tactical IMU

Figure B.21: Dynamic Spoof - Loose Coupling

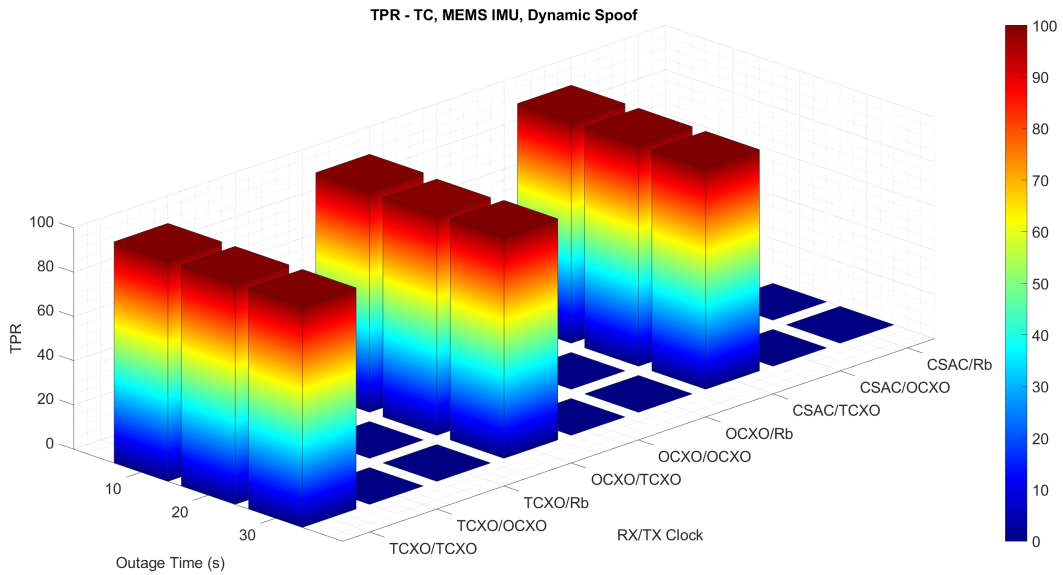

Figure B.22: TPR - TC, MEMS IMU, Spoofed Trajectory Matched to Truth

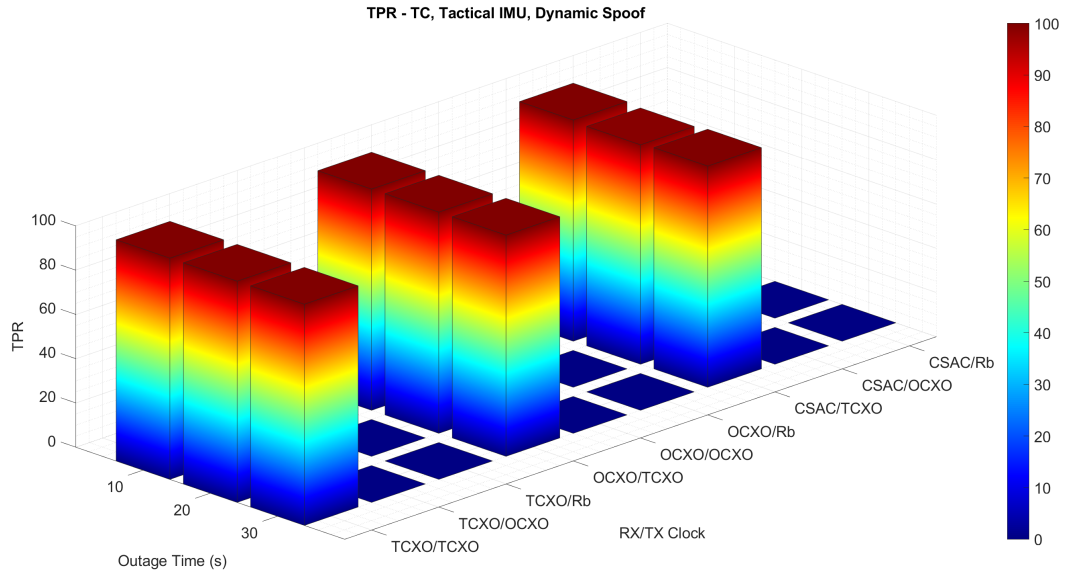

Figure B.23: TPR - TC, Tactical IMU, Spoofed Trajectory Matched to Truth

TC, MEMS, Dynamic Spoof Location			
	10	20	30
TCXO/TCXO	96.88%	88.00%	69.23%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	88.00%	90.32%	73.33%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	93.70%	85.12%	70.00%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(a) MEMS IMU

TC, Tactical, Dynamic Spoof Location			
	10	20	30
TCXO/TCXO	92.42%	91.15%	87.90%
TCXO/OCXO	0.00%	0.00%	0.00%
TCXO/Rb	0.00%	0.00%	0.00%
OCXO/TCXO	94.44%	92.74%	85.36%
OCXO/OCXO	0.00%	0.00%	0.00%
OCXO/Rb	0.00%	0.00%	0.00%
CSAC/TCXO	96.80%	93.50%	80.60%
CSAC/OCXO	0.00%	0.00%	0.00%
CSAC/Rb	0.00%	0.00%	0.00%

(b) Tactical IMU

Figure B.24: Dynamic Spoof - Tight Coupling