Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Irfan Ljevo

Pose Estimation of Aquaculture Crane Using IMUs

Master's thesis in Subsea Technology
Supervisor: Christian Holden, NTNU
Co-supervisor: Sveinung Ohrem, SINTEF & Bent Oddvar Arnesen Haugaløkken, SINTEF
July 2022

NTNU
Norwegian University of Science and Technology

Irfan Ljevo

Pose Estimation of Aquaculture Crane Using IMUs

Master's thesis in Subsea Technology
Supervisor: Christian Holden, NTNU
Co-supervisor: Sveinung Ohrem, SINTEF & Bent Oddvar Arnesen Haugaløkken, SINTEF
July 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of Science and Technology

NTNU

Departement of Mechanical and Industrial Engineering,
Faculty of Engineering

TPK4960 – Robotics and Automation

Pose Estimation of Aquaculture Crane Using IMUs

Author: Irfan Ljevo

Supervisor: Christian Holden, NTNU
Co-supervisors: Sveinung Ohrem, SINTEF
Bent Oddvar Arnesen Haugaløkken, SINTEF

Trondheim
July 2022

Preface

This report documents the work performed in connection with my Master's thesis during the spring semester of 2022. It represents one full semester of work and completes my two-year Master of Science program in Sub-sea technology (MIUVT) at the Norwegian University of Science and Technology (NTNU).

Automation and robotics have always been fields that have fascinated me growing up, and provided motivation for choosing this both as an object of my research, and as a choice of my studies. With a background in electrical engineering, specialized in automation, I realized importance and potential in exploring such topic, which made me even more interested in performing research regarding this specific problem. There are still many issues that need to be resolved in this field, and I am happy to have been given the chance to contribute.

I would like to thank my supervisor and co-supervisors at NTNU and SINTEF, Professor Holden, Dr. Haugaløkken, and Dr. Ohrem, for providing insight, research material and hardware needed in order to carry out a successful research. I would also like to extend my gratitude for all the time and effort my supervisors have generously shared in order to help me in my research, both in theoretical and practical aspects. Lastly, I would like to express my gratitude to the closest friends and family for all the encouragement, motivation and support that they have provided, especially my mother and father who have led the way all these years and convinced me that five years can pass in an instance.

Trondheim, July, 2022

Irfan Ljevo

Abstract

Salmon is one of Norway's most exported goods, therefore aquaculture sector is among the most significant in the country. As a result, creating changes in this area will be essential to obtain more success, which will then give enhanced competitive advantage over other industries in other regions of the world. It is essential to continue making investments in technologies and strategies that will address the industry's existing issues and difficulties. One of the main challenges is the dynamic environment where the work is to be performed. Ships performing their tasks out in the open seas will be heavily influenced by the wind and the motion of the ocean. For the reasons mentioned above, it is crucial to stabilize crane operations when working in an offshore environment. Operating an offshore crane is a difficult task that requires the operator to simultaneously regulate the load's position, anticipate vessel motion, and account for load sway. For many years, there have been various proposals that would partly solve problems caused by the dynamic environment of the ocean. Systems like heave compensation, anti-rolling tanks, bilge keels, active fins and more, have been implemented in order to stabilize the entire vessel. These systems are proven to be effective and are well tested in the field. The downside is that they are relatively expensive and are not easily retrofitted on to the ship. Therefore, another approach has to be taken in order to reduce the cost, and to implement solutions which will make crane operations both easier and safer.

The research done in this report was based on exploitation of the Inertial Measurement Unit (IMU). Capabilities of the relatively expensive industrial IMU were examined in order to give insight into whether this sensor is capable of providing reliable measurements which could help to make crane operations either fully or semi-automated. The reliability of the data, and the overall performance of the sensor, was tested in hypothetical situations. The IMUs are known to suffer from various errors like white noise and bias, all of which will have to be minimized as much as possible if the IMU is to provide quality measurements. In addition to errors, motion of the vessel and the tasks performed on the deck of the ship will also greatly influence performance of the IMU. The results show that the IMU is capable of estimating pitch and roll angles of the crane links, if it is positioned properly and is isolated from the outside disturbances like man made vibrations, electromagnetic fields, etc. Unfortunately, the IMU is not capable of providing direct estimations of the position, due to the fact that none of IMU's sensors are capable of reliably estimating the position of the desired object, directly. Integration of the accelerometer or gyroscope readings will lead to a completely false position estimations due to the fact that both readings contain bias offset which will cause position estimations to drift over time. With this said, the work performed in this thesis shows that the dynamic environment of the ocean would create numerous disturbances that would have a great impact on the IMU. This fact, in conjunction with inherent IMU weaknesses, shows that IMU alone is not capable of delivering desired results.

Sammendrag

Et av norges største eksport varer er laks, derfor er akvakultur sektoren en av de mest betydelige sektorene i landet. Det vil derfor være avgjørende å skape endringer innenfor dette området for å oppnå størst mulig suksess, noe som vil da gi økt konkurransefortrinn i forhold til andre bransjer i andre regioner av verden. I tillegg er det viktig å fortsette å investere i teknologi og strategier som vil påvise bransjens eksisterende problemer og utfordringer. En av de primære utfordringene er det dynamiske miljøet hvor arbeidet skal utføres. Skip som utfører deres oppgaver ute på havet vil være sterkt påvirket av vinden og havbevegelsen. Grunnet de overnevnte utfordringene er det avgjørende å stabilisere krandriften når arbeidet utføres i et offshoremiljø. Offshore krandrift er en kompleks oppgave som krever at operatøren må samtidig kontrollere posisjonen til lasten, forutsi fartøyets bevegelse og kompensere for lastens svai. Det har vært ulike forslag opp gjennom årene som har delvis løst problemer forårsaket av det dynamiske miljøet i havet. Systemer som heave compensation, anti-rolling tanks, bilge keels, active fins og mer, er implementert for å stabilisere hele fartøyet. Disse systemene har vist seg å være effektive og er godt testet i feltet. Ulempen er at de er relativt dyre og kan ikke lett etter-monteres på skipet. Derfor, må et annet alternativ bli tatt i bruk for å redusere kostnadene, og implementere løsninger som vil gjøre kranoperasjoner både enklere og sikrere.

Forskingen i denne rapporten baserte seg på utnyttelse av Inertial Measurement Unit (IMU). Funksjonaliteten til den relativt dyre industrielle IMU ble undersøkt for å gi innsikt i om denne sensoren er i stand til å gi pålitelige målinger som kunne bidra til å gjøre kranoperasjoner enten hel- eller halvautomatiserte. Påliteligheten til dataene, og sensorens generelle ytelse, ble testet i hypotetiske situasjoner. IMUene er kjent for å avvike på grunn av ulike feil som hvit støy og bias, som alle må minimeres mest mulig dersom IMU skal gi kvalitetsmålinger. I tillegg til feil, vil bevegelse av fartøyet og oppgavene som utføres på skipets dekk også ha stor innflytelse på ytelsen til IMU. Resultatene viser at IMU er i stand til å estimere roll and pitch vinklene av kranlenkene, hvis den er riktig plassert og er isolert fra ytre forstyrrelser som menneskeskapte vibrasjoner, elektromagnetiske felt, osv. IMU er ikke i stand til å gi direkte estimeringer av posisjonen, grunnet at ingen av IMUs sensorer gir pålitelige estimeringer av posisjonen til det ønskede objektet. Integrasjon av akselerometeret eller gyroskopmålinger vil føre til fullstendig falske posisjonsestimater på grunn av at begge avlesningene inneholder bias forskyvning som vil føre til at posisjonsestimater avviker over tid. Arbeidet utført i denne oppgave viser til at det dynamiske miljøet i havet vil skape mange forstyrrelser som har stor innvirkning på IMUen. Dette viser til at i tillegg til de underliggende IMU-svakhetene så vil ikke denne sensoren være i stand til levere resultatene som ønskes oppnådd.

Contents

List of Figures

List of Tables

Listings

Nomenclature

IMU	Inertial Measurement Unit
MCU	Microcontroller Unit
MEMS	Microelectromechanical system
RTOS	Real time operating system
I2C	Inter - Integrated Circuit
Roll(θ)	Rotation around x-axis
Pitch(ϕ)	Rotation around y-axis
Yaw(ψ)	Rotation around z-axis
Dof	Degrees of freedom
IO	Input Output
a_x	Measured x axis acceleration
a_y	Measured y axis acceleration
a_z	Measured z axis acceleration
ω_x	Measured x axis angular velocity
ω_y	Measured y axis angular velocity
ω_z	Measured z axis angular velocity
A_x	Accelerometer x sensitivity axis
A_y	Accelerometer y sensitivity axis
A_z	Accelerometer z sensitivity axis
G_x	Gyroscope x sensitivity axis
G_y	Gyroscope y sensitivity axis
G_z	Gyroscope z sensitivity axis

Chapter 1

Introduction

1.1 Problem statement and objective

This report will investigate capabilities provided by the Inertial Measurement Unit in order to deliver pose estimation of the crane joints and load. Previous master's thesis [1] written on this subject documented research regarding use of expensive and cheap IMUs. More specifically, difference between obtained measurements and their ability to provide reliable pose estimation and control of the suspended crane load. This research showed that both cheap and expensive IMU will provide data which can be used in order to estimate rotation angles and use this knowledge in control structure. Obvious differences between low and high-end IMU were in the precision of the sensor measurements and the level of corruptions in the form of bias, drift and white noise.

The basis of this report is examination of the exploitation of IMU sensors. Sensors will be used in order to determine how reliable they are to provide pose estimation of the crane end-effector and load suspended on it (crane hook). Inertial measurement unit used in this research is Bosch XDK110, which is relatively expensive, having current price of 200 USD. Being more expensive and with far more capabilities than standard low end IMUs, it still suffers from the same corruptions and irregularities which are also occurring with the cheap 100 NOK IMUs. Problems like drift, bias, scale factor and white noise are all typical issues faced by the inertial units regardless of their cost and quality. Therefore, this report will also look into methods for removing or minimizing such problems, as well as methods for minimizing external disturbances coming from the environment.

The nature of the dynamic environment provided by the ocean will introduce numerous difficulties which will have great effect on the performance of the IMU. This fact will also be taken into consideration, because IMUs will be heavily affected by the disturbances coming from the wind and sea currents. All of the mentioned above has to be taken into consideration in order to fully understand inertial measurement units and their limitations. Obtaining an understanding of the capabilities, limitations and physical motions will provide the basis for possible solutions in instances where IMU fails to deliver the desired results.

If ship mounted cranes are to be stabilized, the control algorithm would require sensors that provide reliable data regarding the crane's orientation and position. All of the necessary information can be obtained if the crane had preinstalled sensors provided by a manufacturer, for example a rotary encoder connected to the crane's revolute joint, which measures rotation directly. If the sensors are not preinstalled on the crane, the alternative to this would be to retrofit the crane with the desired sensors afterwards. Due to the diversity in deck mounted cranes, which are as diverse as the task they are performing, it is highly unlikely for all cranes to have preinstalled sensors that are already positioned and calibrated for the task. Additionally, it is not always the case that preinstalled sensors are of desired quality and complexity. Sensor failures and malfunctions are always possible, thus it would not help to have an inbuilt sensor which can be difficult to replace or to calibrate.

For this reason, it is worth investing time and resources in researching methods to automate crane operations with the sensors that are retrofitted on the links and joints. Another important aspect are the finances. Cranes with the inbuilt sensor systems are more expensive than the ones that are not incorporating such systems. Manufacturing cranes with built-in sensors would require multiple expenses in different stages of development. Crane's links and joints would have to be modified to host sensors. This would potentially include increasing the size of the component and developing special electromagnetic protection for the sensor. In order to make the sensor system more user-friendly, companies would have to invest in the development of a simpler software. All of the mentioned above would increase the cost of buying a crane with inbuilt sensors. Thus, it is natural to assume that a retrofitted sensor system would be significantly more cost-efficient. This approach would also allow for greater flexibility when considering different types of sensors. One would have much greater freedom to choose methods and approaches when deriving and implementing control strategy. The crane that is mentioned through this research is SINTEF owned, and is mounted on the SINTEF ship, used for the aquaculture purposes. As of the writing of this report, the ship seen in Figure 1.1 is stationed in Frøya. The SINTEF crane that is to be automated can be seen in Figure1.2.

Figure 1.1: SINTEF ship.

Figure 1.2: SINTEF crane, from [1].

To summarize, the objectives of this research are:

- Understand IMU's method of operation and its inner workings.
- Investigate problems that will occur when using an IMU in the determined environment.
- Learn procedures used to either eliminate or minimize effects of those problems as much as possible.
- Understand limitations of the IMU when used to obtain pose estimation. Provide suggestions for the possible solution to those limitations.
- Evaluate different types of filters which can be used in order to improve upon the quality of the measurements.
- Compare different filtering methods.
- Derive equations describing position of the suspended load.

1.2 Literature study

Cranes are machines that are used for transporting heavy loads or hazardous materials from one place to another [2]. Diversity in tasks that are to be executed in harbors, or on the open seas, have caused the need for cranes to be both land fixed and ship mounted. Through history, ships were one of the main means of transportation of people and goods. With the industrial revolution in the 19th century and globalization in the late 20th and early 21st century, ships have become one of the main means of transporting various merchandise around the world. With this expansion came also the need for ships to have on board crane, which would allow for loading and offloading in harbors which are not equipped with adequate crane systems.

Historically, cranes were predominantly stationed on land, mostly in harbors and construction sites. The first historical mention of the use of hoisting mechanism in Ancient Greece goes back to 530 BC, mainly concerning construction of the temple of Artemis in Ephesus. With this said, it is false to assume that this is the first use of the crane system, taking into consideration that great structures like the ones in Egypt and Mycenae are much older [3]. Deck mounted cranes, as they are known today, first started appearing in the late 19th and early 20th century. Battleship USS Kearsarge was the first military vessel to be decommissioned and modernized to host a deck mounted crane capable of lifting 250 tons [4]. In the second half of 20th century with the advancements in the oil industry, like sub-sea production and exploitation, ship mounted cranes became crucial as they were the main tool for the heavy lifting construction operations.

In order to achieve maximum flexibility and operational area, cranes are built using different components which allow for multiple motions and degrees of freedom(Dof). Cranes usually incorporate revolute and prismatic joints, which allow for rotational and translational type of motion, as it can be seen in Figure 1.3.

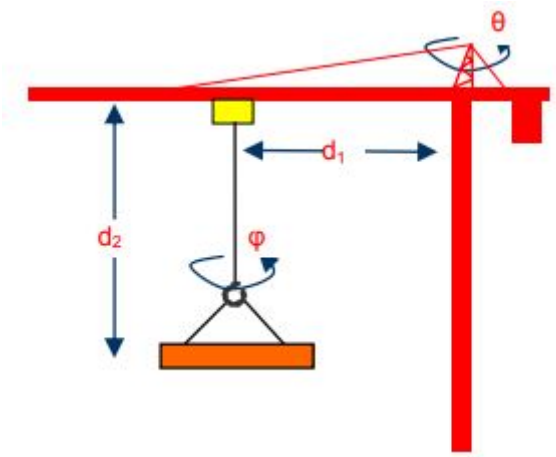

Figure 1.3: Crane degrees of freedom, from [5]

Typical cranes, as seen in Figure 1.3, are in general considered as under-actuated systems. Meaning that they have more degrees of freedom than actuators [6]. In practice, this means that deriving a control scheme for such systems is more demanding, considering the fact that there are different degrees of freedom which cannot be controlled directly with an actuator. Under-actuated systems are nonlinear systems that lack feedback and are more complex [6]. The reason for this property of the crane system is due to the fact that cranes have to be light, easily maintained, robust and cheap. Therefore, cranes usually have as few actuators as possible. With this said, not all cranes are constructed with their simplicity as the main strategy. Crane construction and functionality is as various as tasks that are to be carried out, regardless of where the crane is stationed.

Most cranes that are land based experience very little to no disturbances. The only disturbance which can influence crane operations to a noticeable degree is the wind. However, offshore crane operations are exposed to continuous disturbances like wind and motion caused by the sea waves and currents [7]. All these disturbances will cause the ship to experience motion which will have an effect on the crane in heave elevation and in roll, pitch and yaw movement, as seen in Figure 1.4. On board cranes are mainly operated by human operators using manual control input in order to control motion of the crane. This also means that any unwanted motion caused by the disturbances will have to be eliminated using operator input. Naturally, success in eliminating the unwanted motions is solely based on the operator's training and intuition, which is acquired through experience. With this said, it can be concluded that the onboard safety conditions are completely dependent on the skills of the crane operator and his/hers ability to predict and counteract the motion caused by the ocean and wind. Fully or semi-automated cranes would increase the safety of the crane operations and allow for easier crane operations. If automation of the crane is to be achieved, the control algorithm would require a mathematical model of the system. The crane, seen in Figure1.2, is consisting of two revolute and one prismatic joint. The position of the crane load, can be modeled using Euler-Lagrange equations [2]. The position of the crane hook, also known as the end-effector, can be obtained from the mathematical model describing the crane kinematics. The kinematics equations can be obtained by using robot theory and the Denavit Hartenberg convention [8].

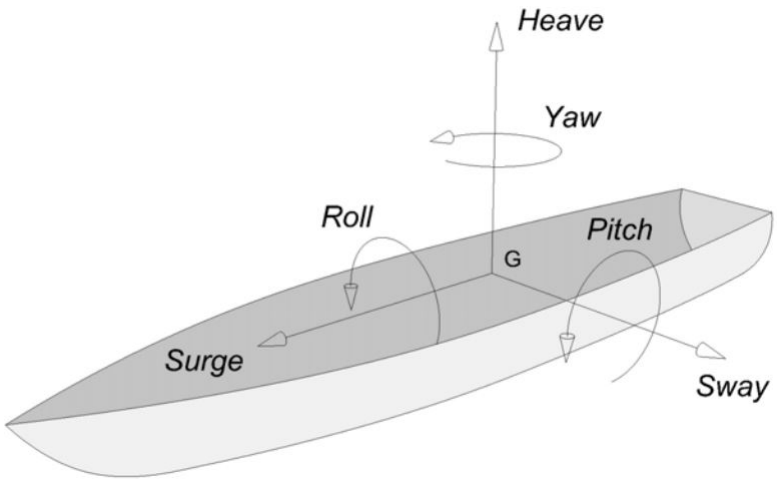

Figure 1.4: Representation of heave, roll, pitch and yaw, from [9].

The mathematical model describing the crane load, requires sensors to calculate the angle between the wire and the suspension point. This angle can be provided by an IMU, connected to the crane hook. This is achievable by using a wireless IMU connected to the hook, in conjunction with the mathematical model of the hook and the wire [10]. Low cost IMU will suffer from various error terms, therefore different filtration methods have to be implemented in order to achieve the greatest possible precision of the estimated parameters.

Since the crane will be operating in the dynamic environment, it is crucial for the control algorithm to have an estimator tasked with estimating the motion of the ship caused by the waves. The waves will cause the load to sway with the specific frequency which must be included in the system so that the control algorithm can compensate for the motion. This can be performed using a feed-forward controller, where disturbances caused by the waves are compensated for before they can affect the system.

The IMU will be able to provide orientation estimations for the crane links. Direct roll and pitch values will be provided by the accelerometer, in addition to the angular rate measurements provided by the gyroscope. The IMU will have to be used together with other sensor in order to ensure the best possible quality of the estimated parameter. If the IMU is to be used alone, the noisy accelerometer and gyroscope would quickly cause perceived motion to radically deviate from the true motion [11].

1.2.1 Related work

History of the IMU begins in the 1930s, where it was used in aircraft navigation. Because of its constraints mainly in size, cost and power consumption, IMU usage at the time was restricted to bulk applications and thus, unpopular for smaller size devices and consumer applications [12]. Further innovations like the gyroscope based compass [13] allowed for navigation and heading calculations. As early as 19th century, ships started using anti-roll bilge keels as the mean for roll stabilization [14]. After that came a series of innovations like ballast tanks (antiroll tanks) and active fins [15], all of them with the intent to stabilize the ship's motion during travel. Unfortunately, such systems are not cheap and cannot be easily installed on a ship which does not have this type of stabilization system from before. In addition to the fact that the vessel is still going to experience motion due to disturbances.

Offshore cranes are the major actor on-board of platforms and vessels in transporting and lifting operations. Under rough sea conditions, offshore crane operations result in many problems such as load sway, positioning accuracy, collision avoidance and manipulation security, etc. To monitor ships movement, commercial offshore cranes usually adopt some motion detection unit like IMU and Motion Reference Unit (MRU). Then, according to this data input, a control system calculates how actuators have to react to such movements [16].

Yingguang Chu, Filippo Sanfilippo, Vilmar Æsøy and Houxiang Zhang at the Aalesund University College [16], presented anti-sway and anti-heave algorithm which would provide reliable control of vessel during potential sub-sea lifting operations. In 2016 Ning Xianliang, Zhao Jiawen, and Xu Jianan [17] investigated use of IMU in order to estimate heave velocity by motion transformation. This velocity is used in conjunction with the heave filter in order to derive a mathematical equation describing the ship's vertical movement. The equation is later used as a model in the active heave compensation system.

Florentin Rauscher, Samuel Nann and Oliver Sawodny [10] investigated crane motion control using an IMU. They presented a method to estimate angle of the wire using a wireless IMU mounted on the hook. In addition, methods for filtering IMU data were presented, as well as modeling of the hook and the wire. A complimentary filter was used in order to improve upon data quality, while a Kalman filter was used for simple estimations of an angle between the wire and the crane assembly.

Article written by Liyana Ramli, Zaharuddin Mohamed and others [2] gave comprehensive review of the crane control strategies based upon different crane types. Anti-sway systems available on the market are also presented. The master thesis written by O. Gjelstenli [18], at NTNU, provides in depth insight into mathematical models describing crane and load movement needed for an effective anti-sway control.

Norhafizan Ahmad, Raja Ariffin Raja Ghazilla, and Nazirah M. Khairi [12] wrote the article describing IMU. The article describes various aspects of the IMU like data accuracy, size, response rate, and IMU Dof. It provides information regarding several application fields for the IMU. According to the article, IMUs are used in robotics, medical rehabilitation, sports, navigation systems etc. In 2008, Chris C. Ward and Karl Iagnemma investigated use of the slip detector algorithm which uses measurements from the IMU, GPS, and front wheel encoders in order to detect wheel slip of the robot car [19].

Manon Kok, Jeroen D. Hol and Thomas B. Schön wrote about the use of inertial sensor for the position and orientation estimation [20]. The article gives detailed description of the probabilistic models, models for estimation of the position and orientation, as well as calibration and filtering methods. Weixin Yang, Alexandr Bajenov and Yantao Shen implemented IMU on the snake robot with the intention of measuring translation and providing trajectory tracking [21].

Shenghai Wanga, Yuqing Suna, Haiquan Chena and Jialu Du proposed an improvement to the three-post direct ship motion compensation (DSMC) designed by Barge Master [22]. This system mounts the crane on top of a three-post stabilization platform, which stabilizes crane operations under rough seas conditions. The system measures ship's motion and directly compensates in roll, pitch and heave by a 3-axis motion platform. Performance of such platforms is still limited due to size of the components and their functionality in the dynamic environment.

Thor I. Fossen in his book [23], gives detailed information regarding mathematical modeling, rigid body kinetics and kinematics of the marine craft. Models for ships, offshore structures and underwater vehicles are described in detail, together with sensors and signal filtering methods. The book also provides historical information regarding various innovations significant for the marine vessels and motion control.

1.3 Report structure

Chapter 2 will provide the reader with the information regarding the IMU used in this thesis. IMU's inner workings are presented together with the code used to extract the data using Matlab and Python script. Additionally, the IMU limitations and potential disturbances identified. In Chapter 3, the equations used to obtain Euler angles are presented and typical problems like the gimbal lock are explained. Chapter 4 deals with the digital processing of the IMU measurements. Digital filtering methods like the low-pass filters are shown together with the method for the sensor fusion. Chapter 5 shows mathematical modeling of the suspended crane load using the Lagrange equations. Additionally, Runge-Kutta iterative integration method is presented. Chapter 6 is the final chapter providing results obtained from the methods described in Chapters 2-6.

Chapter 2

Inertial Measurement Unit (IMU)

2.1 Bosch XDK110 Cross-Domain Developement Kit

The goal of this research is to use Bosch XDK110 cross-domain development platform in order to read data which is provided by the numerous sensors available within the platform. The XDK is a universal programmable sensor device and IoT (Internet of Things) prototyping platform with an open SDK. This device is an industrial platform which combines multiple sensor communication protocols in order to achieve maximum flexibility and application use. Bosch XDK combines multiple MEMS sensors like accelerometer, gyroscope, thermometer, magnetometer, humidity sensor etc. Bosch XDK is an industrial unit which currently costs 250 USD. In addition to sensors, it also possesses multiple communication protocols like Bluetooth 4.0 low energy and wireless LAN. XDK is equipped with the 32-Bit microcontroller (ARM Cortex M3), which has 1 MB Flash and 128 kB RAM. In order to achieve maximum flexibility when used, XDK has an internal 560 mAh Li-Ion rechargeable battery. It also has SD card slot which allows for data to be logged on the SD card while performing measurements. Full capabilities of the XDK are listed in Tables 2.1, 2.2 and 2.3.

Table 2.1: Bosch XDK110 sensors.

Name	Type
BMA280	Accelerometer
BMG160	Gyroscope
BMM150	Magnetometer
BMI160	Accelerometer/Gyroscope
BME280	Humidity/Pressure/Temperature
AKU340	Ambient Noise
MAX44009	Ambient Light

Table 2.2: Bosch XDK110 communication.

Name	Type
Cable	USB 2.0
Wireless	Bluetooth; Wireless LAN
LED	1x green, 1x yellow, 1x orange, 1x red

Table 2.3: Bosch XDK110 technical specifications.

Name	Value
Temperature Range	-20 - 60 °C operating, 0 - 45 °C charging
Humidity	10 - 90 %, non condensing
IP Rating	IP 30 (IEC 60529)
Flammability classification	HB (IEC 60695-11-10/-20; CSA C 22.2)
Voltage	5 V DC
Charging Current	500 mA maximum
Communication (cable)	USB
Wireless LAN	IEEE 802.11 b/g/
Bluetooth 4.0	IEEE 802.15.1
Cable length	<3m

Figure 2.1: Bosch XDK110 Cross-Domain Development Kit.

2.2 Software

In order to program the XDK, Bosch has supplied a software called XDK Workbench, which also includes a direct interface to the XDK community and the XDK API documentation. Workbench offers two programming languages, Eclipse Mita and C. Eclipse Mita is a programming language made in order to simplify programming of the embedded platforms for the users without embedded development background. This makes it much more user-friendly and much easier to learn than the standard C. Mita is event-based, e.g. events triggered by sensors, timers or connectivity.
This development style makes it fit perfectly for IoT applications where devices spend most of the time either in hibernation or some other form of power saviour mode. Thus it is possible for sensors to wake up, perform the operation and go back to power saving mode. Listing 2.1 gives an example of accelerometer data being printed to the terminal using Eclipse Mita code:

```
every 100 milliseconds {
    var x = accelerometer.x_axis.read();
    var y = accelerometer.y_axis.read();
    var z = accelerometer.z_axis.read();
    println(`Accelerometer output:
            \n x: ${x}mg\n y: ${y}mg\n z: ${z}mg`);
}
```

Listing 2.1: Printing accelerometer readings.

Due to its simplicity and available examples on the Bosch and Eclipse Mita sites, it was decided that Mita will be used as a programming language. As mentioned in Section 2.1, the XDK offers multiple sensors which can easily be configured and extracted using Mita script. When the Mita script is compiled and executed, it is converted to the C code. This is performed in order to leverage debugging tools, efficient compilers and existing infrastructure provided by the C programming language. Meaning that Mita still has possibility to use C libraries. In Mita script, every sensor has to be configured before it can be practically used. When being configured, user defines different parameters like the bandwidth, range, mode of operation etc.

Figure 2.2: XDK Workbench software.

2.3 Choice of sensors

For this research, 3-axis accelerometer and 3-axis gyroscope are going to be primary sensors used for the purpose of obtaining needed data. Pose estimation is a task that implies finding or estimating position and orientation of the desired object. With this said accelerometer, gyroscope and magnetometer are the three sensors which will provide orientation in the form of roll, pitch and yaw. Using trigonometry equations together with the physical dimensions of the crane, it is also possible to obtain the position of the tracked object.

For the purpose of finding angle of the joints, more specifically roll and pitch values, gravitational acceleration will be used together with the angular velocity provided by the gyroscope. When finding yaw angle, magnetometer is the sensor which would be used in order to achieve such a task. Magnetometers are sensors that measures the strength and sometimes direction of the Earth's magnetic field [24]. It detects fluctuations in Earth's magnetic field, by measuring magnetic flux density at the sensor's point. Using those fluctuations, it is capable of finding a vector pointing at the Earth's magnetic North pole. This vector is later used in order to obtain yaw angle, a task which can't be effectively executed by neither accelerometer nor gyroscope.

The magnetometer sensor will not be used in this research. This is due to the great uncertainty whether the sensor would be reliable and useful in the scenario where IMU is mounted on a ship crane. This concern is due to the fact that magnetometer measures magnetic field which can be distorted by both soft and hard iron sources, thus corrupting the measurements. A soft iron source is a source that cannot generate magnetic field, while hard iron sources can generate magnetic field. Magnetometer must be calibrated in order to eliminate as much of those corruptions as possible, but this does not solve the problem. Ship deck is an area where magnetic field is influenced by the soft iron sources like deck floor, railings and crane component, all of which are made from various metals. In addition, ships also host hard iron sources capable of generating its own magnetic field, like power cables and electrical motors. It is also important to emphasize that magnetometers also have to be calibrated to compensate for the changes in the Earth's magnetic field, which will occur if the ship changes its geographical location.

As shown in Table 2.1, the XDK posesses multiple accelerometers and gyroscopes. BMI160 Accelerometer/Gyroscope is an IMU which is included in the XDK. This means that the user of the XDK practically has the ability to choose which accelerometer and gyroscope to use. Either to choose BMG160 and BMA280 gyroscope and accelerometer, or to choose BMI160 gyroscope accelerometer package. After some trials it was concluded that there is no noticeable difference between individual accelerometer-gyroscope (BMA280-BMG160) and IMU package BMI160 containing another set of accelerometer-gyroscope. For this reason it was chosen to use individual accelerometer (BMA280) and gyroscope (BMG160). As of now, notation IMU will only be used when refering to the XDK as a whole.

2.4 Measurement output and limitations

XDK110 BMA280 accelerometer gives scaled acceleration measurements in gravitational force (mG). When IMU is at rest on a steady surface like a table top, with the axis pointing as shown in Figure 2.3, the sensor should transmit measurements of 1000 mG for A_z and 0 mG for both A_x and A_y. These are the perfect measurements that will not be achievable in reality due to different errors and imperfections present in the measurements.

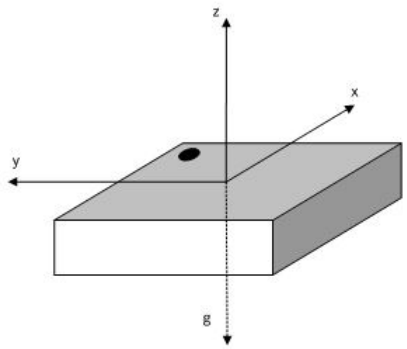

Figure 2.3: Orientation of sensing axis for accelerometer and gyroscope.

BMG160 gyroscope transmits scaled angular rate measurements in degrees per second. Sensitivity axes for the gyroscope are the same as for the accelerometer. If gyroscope is positioned as described before, on the table top at rest without any disturbances, it should measure angular velocity of 0 degrees per second for all sensitivity axis. Maximum sampling frequency of both BMA260 and BMG160 is in theory 2 kHz but in reality it reaches maximum of 1 kHz. This is due to the limitations in the I2C protocol which is used as a communication between accelerometer/gyroscope and the MCU. Practically, an application which tries to read data from the two sensors, which both are set to sample rate of 1kHz, cannot be read by the MCU with 1kHz each. This means that a programmer has to be careful when setting the sample rate of every sensor. Additionally, the sampling rate is also influenced by interrupts, which are performed by the freeRTOS operating system[25].

For this research, it was concluded that the sampling frequency of 100 Hz is going to be chosen. This is derived from the assumption that 100 Hz sampling frequency is more than sufficient for measuring movement of the crane during operation. When performing lifting operations, the deck mounted crane will be speed limited and will therefore not be able to perform rapid movement due to the safety concerns. For this reason, it is assumed that the sampling frequency of 100 Hz will be more than sufficient for the task.

2.5 Measurement errors

With the advancements in electronics, inertial measurement units became smaller and cheaper. This allowed hobbyists to have access to MEMS IMU components for as low as 75NOK.
Every sensor suffers from certain errors caused either by the environment or by the imperfections during manufacturing process. In the case of a IMU, it suffers from measurement errors which are manifested in form of bias, scaling factor, white noise etc. By integrating measurements in the potential navigation algorithm, these errors will be accumulated leading to the drift in output. Regardless of cost, all IMU sensors will suffer from errors like bias, white noise and bias drift. This is the case for every IMU, from the cheapest IMU to the military grade IMUs used on nuclear submarines for the purpose of navigation when submarines are under surface for months at the time. In general, IMU errors can be categorised into two categories: stochastic and deterministic. Deterministic errors are the ones that can be detected in their entirety by monitoring the output, and can be attributed to manufacturing process and differences in components. Stochastic errors cannot be exactly measured. They can be approximated statistically. Such errors are usually electric noise interfering with the system's output and is assumed to be Gausian in nature [26]. Deterministic errors can be removed by calibration, while stochastic errors like white noise can be dampened with the use of filters.

2.5.1 Bias

Bias is deviation from the true measured value in form of a constant offset. Bias can be caused by a variety of factors. Some of them are imperfections during manufacturing process and environmental temperature. As seen of Figure 2.4, bias is a constant offset from the true value which has to be removed if accelerometer and gyroscope are to be used in some type of integration algorithm like the dead reckoning. The bias is typically measured in units of gravity *G* for an accelerometer, and change in angle over time for gyroscope, expressed in radians per second (rad/s) [27]

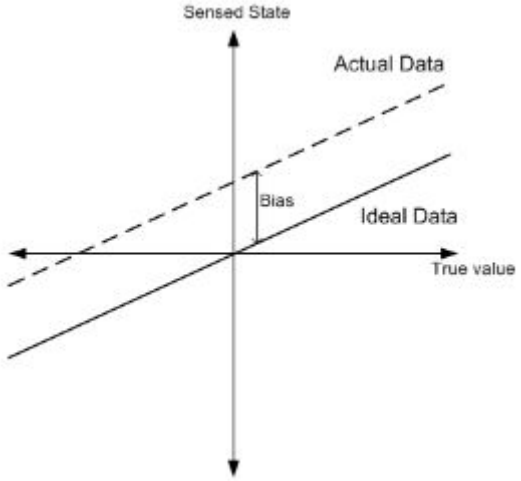

Figure 2.4: Illustration of bias, from [27, p. 17].

2.5.2 White noise

White noise is a random error signal that is present with the given sensor measurements. Quality of the components used to build an IMU is one of the factors which contributes to the presence of white noise in the measurements. Proximity of electrical motors or power lines is an external factor which has to be taken into consideration when using IMU or any other sensors. Electrical current, either in power cables or coils of the electric motors, will produce changing magnetic field which will have an effect on the white noise quantity. As seen in Figure 2.5, white noise is a fluctuation about ideal set of data [27]. This type of error can be removed with the use of digital filters.

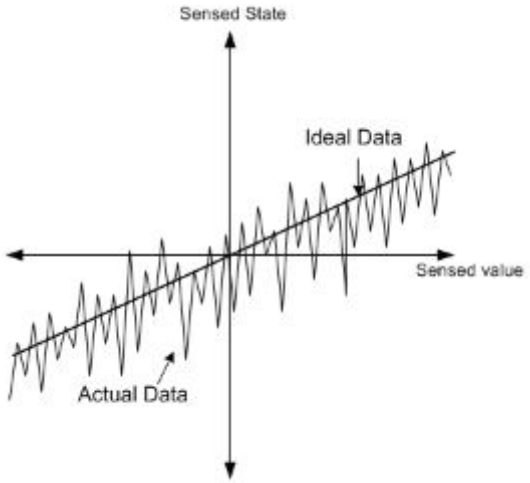

Figure 2.5: Illustration of white noise, from [27, p. 18].

2.5.3 Bias drift

Bias drift, or moving bias as it is also called, is the drift in bias over time. The most important aspect with the bias drift is that it drifts depending on time, environmental factors and stochastic factors. Bias drift is a serious problem since it creates more issues than the white noise. To illustrate, after the first integration (e.g. to estimate angle from angular velocity) the error due to noise grows proportionally to the square root of time. The error due to bias drift will grow proportionally to the time itself. This type of error is usually present in gyroscopes and is attributed to changes in drive frequency and structural asymmetries [28]. Bias drift must not be confused with random walk. Random walk is caused by integration of the errors like the bias drift and white noise which causes long term growth in the output measurements [29, p. 31]. One of the potential solutions to this problem is to implement a zero update to the gyroscope. In case of an aquaculture crane, if the crane's starting position (zero position) is known, than a control algorithm would cancel accumulated error every time the crane is stationary in its zero position. Of course, implementation of such solution is completely determined by the tasks that are to be executed by the crane.

2.6 Obtaining measurements

By using Mita script, one can extract sensor measurements. Before extracting the measurements, sensors have to be configured, as shown in the Listing 2.2. After importing packages, the user must configure sensors and determine the values of the detectable range and bandwidth of the inner AD/DA low-pass filter, using two setup functions. After reviewing the content from the Bosch Developer website [30, 31] and reviewing the accelerometer and gyroscope response, it was decided that the configuration seen in Listing 2.2 below will be used for the further research.

```
package main;
import platforms.xdk110;

setup Gyroscope_BMG160 {
  range = Range_250s;
  bandwidth = Bw_12Hz;
}
setup accelerometer {
    bandwidth = BW_500Hz;
    range = Range_8G;
}

every 10 milliseconds {
    var gyro_x = gyroscope.x_axis.read();
    var gyro_y = gyroscope.y_axis.read();
    var gyro_z = gyroscope.z_axis.read();

    var accel_x = accelerometer.x_axis.read();
    var accel_y = accelerometer.y_axis.read();
    var accel_z = accelerometer.z_axis.read();

    println(`${gyro_x} ${gyro_y} ${gyro_z}
            ${accel_x} ${accel_y} ${accel_z}`);
}
```

Listing 2.2: Configuration and printing of the accelerometer and gyroscope readings.

For the accelerometer configuration, after inspection and trial, it was concluded that the bandwidth of 500 Hz was to be used for the inner low-pass filter. For the gyroscope, after inspection and trial, it was concluded that the bandwidth of 12 Hz gave the best gyroscope response with the least ripples and high sensitivity spikes. Range configuration is specially important for both the accelerometer and gyroscope. The reason for this is due to the fact that in order to achieve the greatest range sensitivity, one must sacrifice resolution of the sensor. Bosch Developer web site[31] gives detailed description on the relationship between range and resolution. As described, 500 deg/s range for the gyroscope will result in resolution of 0.25 deg/s. For this reason, accelerometer range was set to 8G, while gyroscope range was set to 250 deg/s. Lower resolution for the accelerometer could also be chosen due to the assumption that the IMUs connected to the crane will never experience force greater than 4G, even under possible high seas conditions.

When compiling and running code presented in Listing 2.2, XDK Worbench gives command to the IMU to start sending desired data. This data is then sent via USB cable to the USB port which is then printed on the Workbench terminal. XDK Workbench does not have functionalities like Python or Matlab, thus in order to work with the data coming from the XDK, one must be able to capture measurements with either Python or Matlab scripts.

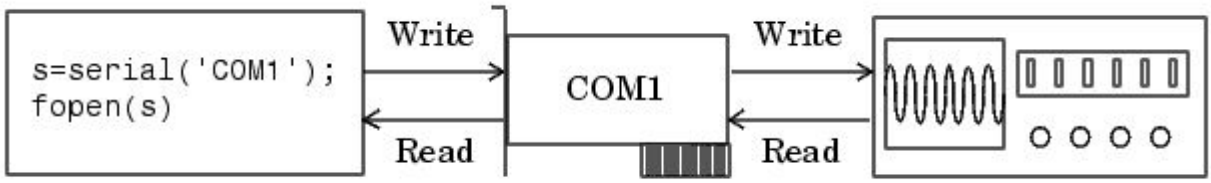

Figure 2.6: Illustration of serial port communication.

The accelerometer and gyroscope data is sent to the USB port where it is captured by either Matlab or Python and is then interpreted and used by the script, as seen in Figure 2.6. Important to note is that two pieces of software or peripheral devices cannot listen to the same USB port at the same time. For this reason, one must compile and run IMU code in XDK Workbench, when compilation is finished and IMU is transmitting the desired data, the user must terminate the Workbench software so that other programs can access the data that is being sent to the USB port. Accessing the IMU data via USB port is shown in Appendix A using Matlab and Python. Matlab offers *serialport* function for the purposes of creating object of the serial port. This object gives access to the information sent via USB port, as it can be seen in Listing 2.3. Function *serialport* takes four input arguments, from which two are mandatory and two are optional. Name of the USB port and baudrate are mandatory to specify. In Listing 2.3, the com port name is "COM6" and baudrate is 9600 bits/sec. The com port name will be different for each computer. In Windows, com port name is designated COM#, while in Linux name is designated as /dev/ttyACM#. Baudrate is very important to specify, since it is a measure of how fast data is moving between instruments that use serial communication [32].

```
clear BoschIMU;
rate = 9600;
BoschIMU = serialport("COM6", rate)

gyro_accel_mat=zeros(1000,7); %Empty matrix
n=1; %Counter
tStart=clock;

while n<=10000
    data = readline(BoschIMU);
    str=split(data);

    for i = 1:6
        gyro_accel_mat(n,i) = str2double(str(i));
    end
    tStop=etime(clock,tStart);
    gyro_accel_mat(n,7)=tStop;

    if(tStop > 50)
        break;
    end
    n=n+1;
end
```

Listing 2.3: Extracting IMU measurements from the USB port using Matlab script.

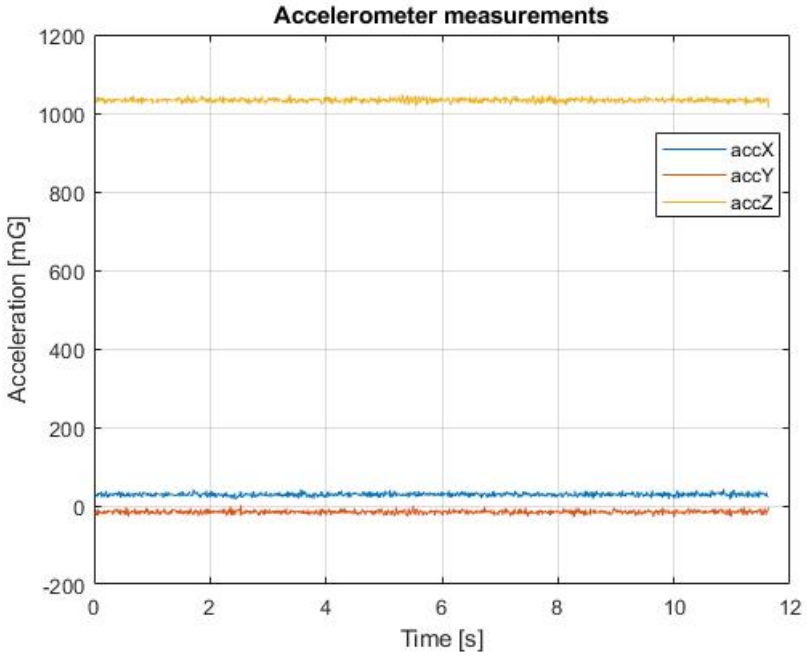

Figure 2.7: BMA280 accelerometer measurements.

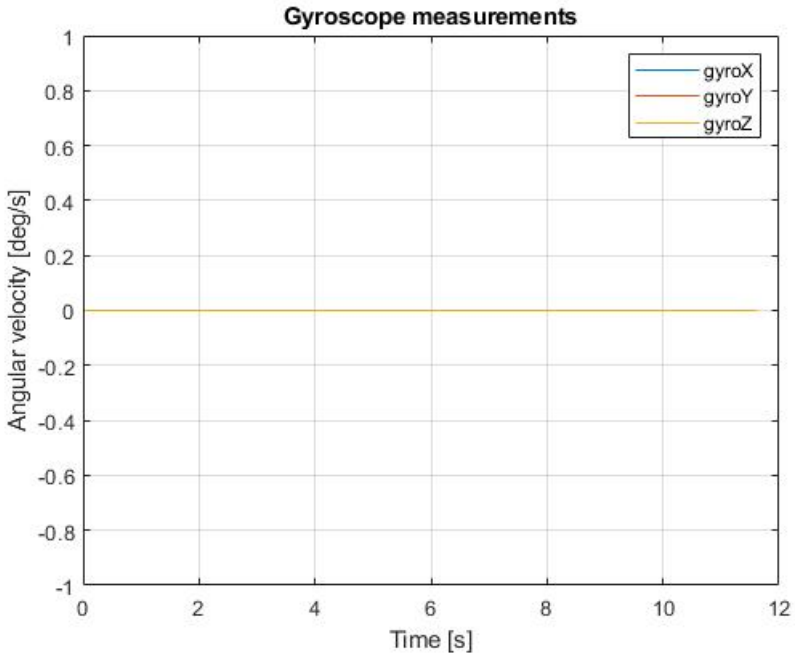

Figure 2.8: BMG160 gyroscope measurements.

2.7 Allan analysis

As mentioned in Chapter 2.5, every inertial measurement unit regardless of its quality and cost will have a certain degree of error in the output measurements. Inertial systems design and performance prediction depends on the accurate knowledge of the sensors' noise model. This can be achieved with the frequency-domain approach of modeling by using power spectral density (PSD) to estimate transfer function of the noise model. Unfortunately, this is difficult for nonsystem analysts to understand. The Allan variance is a time-domain- analysis technique originally developed to study the frequency stability of precision oscillators. Being a directly measurable quantity, it can provide information on the types and magnitude of various noise terms [33].

For this reason, Allan analysis is crucial to perform, specially in cases when noise model is necessary to derive in order to enhance performance of the inertial unit. Kalman filter, especially Extended Kalman Filter (EKF), requires estimator of state space signal model of the process. The signal model dynamics describe how the process may be evolving. Once the system has been identified, Kalman filter theory requires noise covariance matrices Q and R [34]. These matrices are used by the algorithm to determine how much original measurements are to be trusted according to the noise information provided by the noise covariance matrices. In this case Allan investigation would be beneficial since it would provide information regarding which noise is present in the measurements, and the magnitude of that noise.

Before performing Allan analysis, it is beneficial to calculate standard deviation of the accelerometer and gyroscope data. Standard deviation provides information regarding noise level of the dataset. If data consists of random spikes and dips, it will have a high deviation value and will therefore be considered as noisy data. Standard deviation can be used in order to get a general understanding of the white noise magnitude. But it will not provide sufficient results if it is to be implemented on a set of data which is collected over a long period of time. As the dataset is increasing with time, the values for the standard deviation will increase as well due to the fact that the standard deviation diverges with the square root of the data length. Additionally, the standard deviation is significantly dependent on the filter form, e.g., linear least squares, as well as the clock deviations [35]. For the purpose of calculating variance and standard deviation, IMU data will be logged for approximately 30 minutes with the frequency of 100 Hz.

When performing Allan analysis, IMU measurements will be logged for a long period of time, in this case it was chosen to collect the measurements for 5 hours. For this purpose, IMU was placed on a flat surface with the A_z axis pointing towards the sky. Data was logged with the sampling frequency of 100 Hz for 5 hours which accounted for 1.8 million data samples. If the test was to be successful, it is important for IMU to be completely undisturbed for the duration of the data logging. For a vector A made up of N scalar samples, according to [36], the standard deviation is defined as:

$$S = \sqrt{\frac{1}{N-1}\sum_{i-1}^{N}|A_i - \mu|^2} \tag{2.1}$$

where μ is mean of A:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \tag{2.2}$$

Importing the file to the `BoschIMU_script.m` Matlab script, as seen in Appendix A.2, described in and performing calculations using inbuilt standard deviation and variance functions gives:

Table 2.4: Variance values for each axis.

Axis	Variance
A_x	22.05
A_y	34.85
A_z	28.59
G_x	0.00
G_y	0.00
G_z	0.00

Table 2.5: Standard deviation values for each axis.

Axis	Standard deviation
A_x	4.69
A_y	5.90
A_z	5.34
G_x	0.00
G_y	0.00
G_z	0.00

Since standard deviation cannot be used as a measurement of the performance, the method which is independent of the data length has to be used. In this analysis, Allan's definition and results are related to five basic noise terms. Those terms are quantization noise, angle random walk, bias instability, rate random walk and rate ramp. Allan variance will be calculated and later used to obtain Allan deviation which will provide information regarding noise characteristics.
It is assumed that there are N consecutive data points, each having a sample time of t_0. Associated with each cluster, is a time T, which is equal to nt_0. If the instantaneous output rate of the accelerometer is $a(t)$, the cluster average is defined as [37]:

$$\bar{a}(\tau) = \frac{1}{\tau}\int_{t_k}^{t_k+\tau} a(t)dt \tag{2.3}$$

where $\bar{a}(t)$ represents the cluster average of the output rate for a cluster which starts from kth data point and contains the n data points. Deffinition of the subsequent cluster average is:

$$\bar{a}_{k+\tau}(\tau) = \frac{1}{\tau}\int_{t_k+\tau}^{t_k+2\tau} a(t)dt \tag{2.4}$$

The difference between the two cluster averages $d_k(\tau)$ is:

$$d_k(\tau) = \bar{a}_{k+\tau}(\tau) - \bar{a}(\tau) \tag{2.5}$$

Here the quantity of interest is the variance of $d_k(\tau)$ over all the cluster of the same size that can be formed from the entire data. Thus, the Allan variance of length τ is defined as [34]:

$$\sigma^2(\tau) = \frac{1}{2(N-2n)}\sum_{k=1}^{N-2n}\Big[\bar{a}(\tau) - \bar{a}(\tau) - \mu_k\Big]^2 \tag{2.6}$$

Allan variance is calculated using equation:

$$\sigma^2(\tau) = \frac{1}{2}\Big\langle\Big(\bar{y}_{n+1} - \bar{y}_n^2\Big)^2\Big\rangle = \frac{1}{2\tau^2}\Big\langle\Big(x_{k+2M} - 2x_{k+M} + x_k\Big)^2\Big\rangle \tag{2.7}$$

where $\tau = M\tau_0$ and $\langle\rangle$ is the ensemble average. The ensemble average can be expanded to give Allan variance equation, also known as overlapping method [38]:

$$\sigma^2(\tau) = \frac{1}{2\tau^2(N-2M)}\sum_{k=1}^{L-2M}\Big(x_{k+2M} - 2x_{k+M} + x_k\Big)^2 \tag{2.8}$$

Finally, Allan deviation is used to examine noise characteristics:

$$\sigma(\tau) = \sqrt{\sigma^2(\tau)} \tag{2.9}$$

2.8 Frequency domain analysis

Performing noise analysis is extremely important when considering sensors for different purposes. Sensor noise will corrupt the measurements and will lead to distortion of the useful data. For this reason, in addition to the Allan analysis, it is important to perform frequency domain analysis. Allan analysis will provide information regarding the type of noise and its amplitude, but it will not be sufficient since it is the time domain analysis. This is also true for variance σ^2 and standard deviation σ. Frequency domain analysis will provide information regarding different frequency components found in the measured data. These frequencies can be pure white noise, but they can also be disturbances measured by the IMU.

2.8.1 Fourier analysis

In order to obtain an understanding of the signal and its components, it is important to perform Fourier analysis of the signal. Fourier transform transforms a functions in space or time domain into sinusoidal function in frequency domain [39]. This approach will allow for identification of different periodical disturbance signals that are present in the output measurement. The frequency of the disturbances will be identified and will provide additional understanding of the different frequency components that are included in the IMU output measurements. This knowledge is extremely important since various disturbances can be identified and later destroyed using for example notch filter (Band-stop filter). Accelerometer and gyroscope measurements were imported in Python script, which performed calculations using numpy Fast Fourier Transform algorithm. Calculations were performed in `noise_density.py` script, described in Appendix A.1.9. The results can be seen in Figure 2.9.

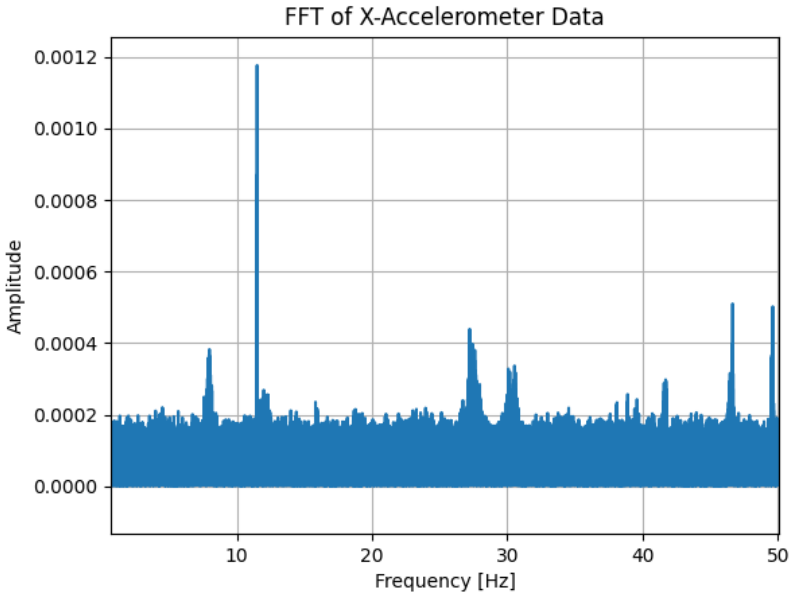

Figure 2.9: Single-sided amplitude spectrum of the A_x.

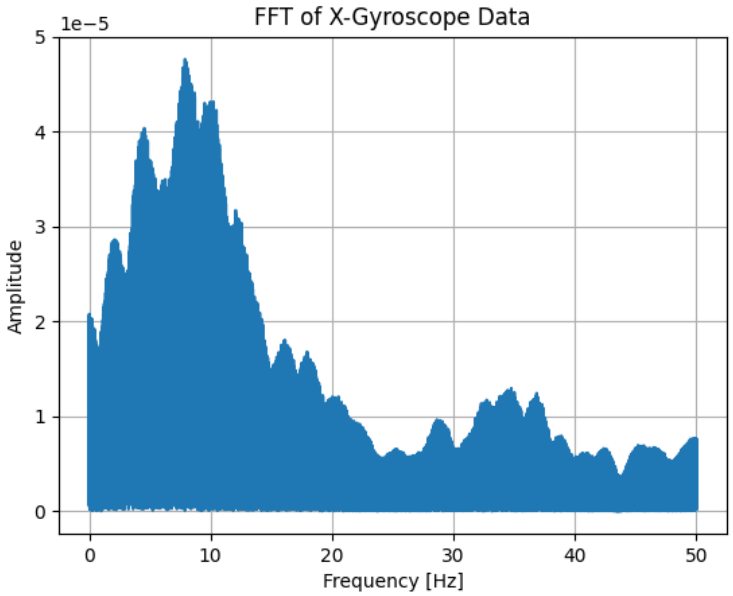

Figure 2.10: Single-sided amplitude spectrum of the G_x.

The Fast Fourier Transform is the algorithm used to compute the Fourier transform of the signal. The Fourier transform states that every signal is the sum of multiple sinusoids, each with specific frequency and amplitude. The plots in Figures 2.9 and 2.10 show multiple peaks that represent sinusoidal waves contained within the measurements captured by the IMU. The Figures 2.9 and 2.10 have both a frequency range from 0 to 50 Hz. Frequency of 50 Hz is the Nyquist frequency (folding frequency) of the IMU's sampling frequency. This is calculated using the equation:

$$F_N = \frac{F_{sampling}}{2} \tag{2.10}$$

Figure 2.9 shows six major spikes at the frequencies of approximately 8 Hz, 12 Hz, 27 Hz, 30 Hz, 46 Hz and 49 Hz. The plot in Figure 2.10, shows multiple spikes spread out across the entire 50 Hz range. The spike with the largest amplitude, at the frequency of approximately 8 Hz, seems to be the only sinusoidal component affecting the gyroscope measurements. Other spikes are most likely products of the white noise, taken into consideration that the 8 Hz spike is present in both accelerometer and gyroscope measurements. The Fourier transform gives detailed information regarding amplitude and frequency of the sinusoidal component present within the measured data. In order to obtain an understanding of the power possessed by the each sinusoidal component, one must perform the power spectral density analysis.

2.8.2 Power spectrum density analysis

Power spectrum density analysis (PSD) will provide insight into the power possessed by each frequency component. In the `noise_density.py` Python script, described in Appendix A.1.10, ScyPy's *signal.welch*() function will be used in order to compute the PSD. The results from the alculation results can be seen in the Figure 2.11.

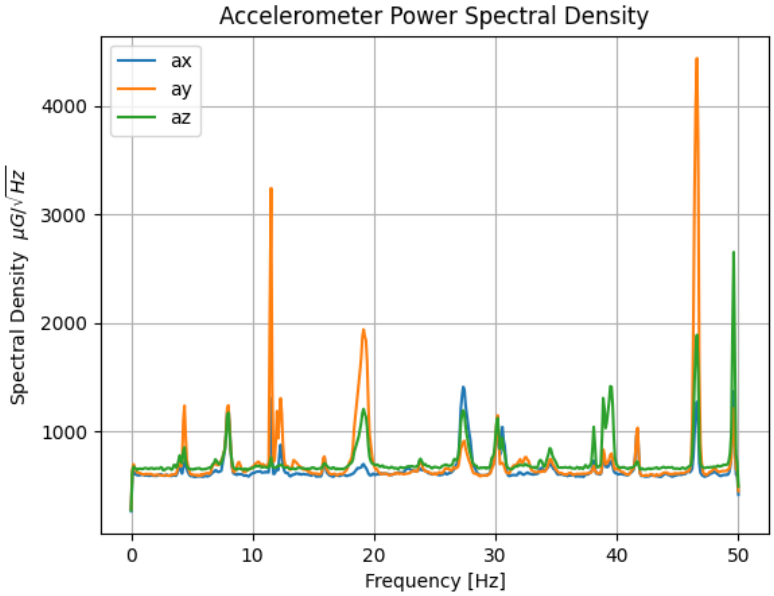

Figure 2.11: Power spectral density of the accelerometer data.

Figure 2.11 shows the power of different frequency components contained by the accelerometer measurements. The largest spikes in the spectral density occur at 12 Hz, 18 Hz, 46 Hz and 49 Hz. The frequencies which have power values below 1500 $\mu G\sqrt{Hz}$ can be ignored since they are most likely not the real part of the transform. The 12 Hz frequency component was identified to be the frequency of the CPU cooling fan. While collecting the data for the analysis, IMU was placed on a table, close to the stationary PC. Due to the proximity of the two, IMU was able to capture the frequency of the CPU fan, which was operating at the average rotational speed of 700 RPM. Therefore, this frequency component can be considered to be the disturbance coming from the environment. The sources of other frequency components, most notably the ones at 18 Hz, 46 Hz and 49 Hz, are difficult to determine. These are also suspected to be disturbance sources coming from the environment. The presence of the white noise is also apparent and can be seen in Figure 2.11.

The same procedure was performed with the gyroscope measurements. The results are seen in Figure 2.12

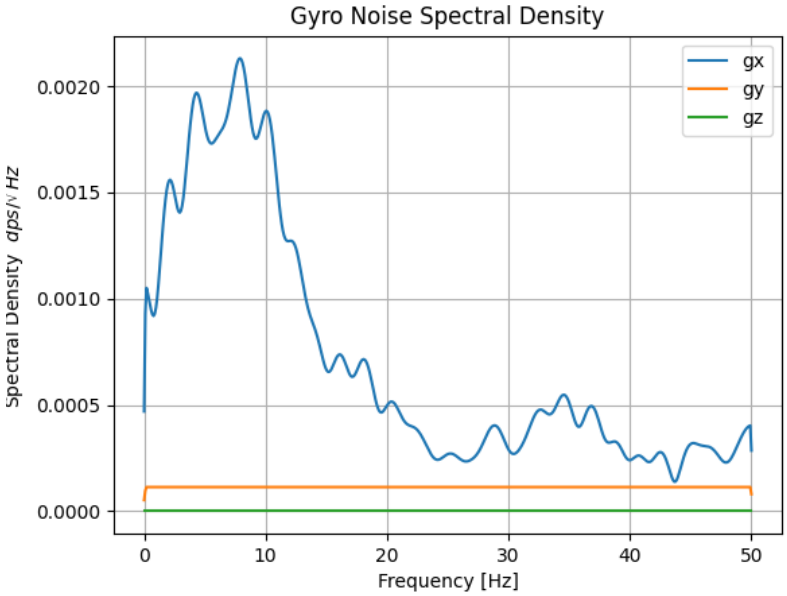

Figure 2.12: Power spectral density of the gyroscope data.

The power spectral density analysis of the gyroscope data shows that it is only the G_x axis measurement that is mostly influenced by the noise. The measurements captured by the G_y and G_z axis do not possess any disturbances, thus the graphs plotted in Figure 2.12, are flat and do not show any frequency components. Noise density values for both accelerometer and gyroscope data are presented in Tables 2.6 and 2.7.

Table 2.6: Standard deviation values for each axis.

Accelerometer axis	Noise density $[\mu G\sqrt{Hz}]$
A_x	649.5801
A_y	728.1151
A_z	730.0412

Table 2.7: Standard deviation values for each axis.

Gyroscope axis	Noise density $[dps\sqrt{Hz}]$
G_x	0.000718
G_y	0.000113
G_z	0.000000

2.9 Calibration

After inspecting Figure 2.7, it is apparent that the accelerometer requires removal of the bias error. Figure 2.8 shows that gyroscope readings seem to be perfect without any constant bias. Accelerometer readings are noticeably off their true value, that should be 1000 mG for A_z and 0 mG for A_x and A_y. For the purpose of finding accelerometer bias, the same data file was used as the one described in Section 2.7. The XDK was oriented with A_z axis pointing at the sky and was recording the data undisturbed for 5 hours. The sensor bias can be removed either by calibration or by estimation using for example Kalman filter. In this task it was chosen to perform calibration due to the complexity of Kalman filters. After processing the data from the text file, the values shown in Table 2.8 were obtained:

Table 2.8: Bias values for each axis, A_z pointing towards the sky

Axis	Measured mean	True val.	Bias	Unit
A_x	36.0	0.0	36.0	mG
A_y	25.0	0.0	25.0	mG
A_z	1044.0	1000.0	44.0	mG
G_x	0.0	0.0	0.0	deg/s
G_y	0.0	0.0	0.0	deg/s
G_z	0.0	0.0	0.0	deg/s

The same procedure was performed two more times, once with A_x axis pointing towards the sky and then with A_y pointing towards the sky, as seen in Tables 2.9 and 2.10.

Table 2.9: Bias values for each axis, A_x pointing towards the sky

Axis	Measured mean	True val.	Bias	Unit
A_x	1057.0	1000.0	57.0	mG
A_y	12.0	0.0	12.0	mG
A_z	-15.0	0.0	-15.0	mG
G_x	0.0	0.0	0.0	deg/s
G_y	0.0	0.0	0.0	deg/s
G_z	0.0	0.0	0.0	deg/s

Table 2.10: Bias values for each axis, A_y pointing towards the sky

Axis	Measured mean	True val.	Bias	Unit
A_x	21.0	0.0	21.0	mG
A_y	1035.0	1000.0	35.0	mG
A_z	-4.0	0.0	-4.0	mG
G_x	0.0	0.0	0.0	deg/s
G_y	0.0	0.0	0.0	deg/s
G_z	0.0	0.0	0.0	deg/s

Results showed that the maximum bias value is at 5.7% of the gravitational acceleration. This indicates that the accelerometer requires calibration. It is important to note is that the bias values seen in Tables 2.8, 2.9 and 2.10 don't necessarily require calibration. The need for calibration is determined according to the task specifications and desired precision. Depending on the precision requirements and by consulting the data quality from the IMU, the need for calibration can be decided. Majority of the crane operations do notr require great precision, thus one can decide if the calibration is required at all. Meanwhile, gyroscopes have to be calibrated before they are used. Measurements provided by the gyroscope can suffer from bias and bias drift. These errors will be extremely problematic if gyroscope data is to be used in some type of integration algorithm like the dead reckoning. The simplest accelerometer model for single axis sensors considers only a scale factor and constant offset, is given by the equation [40]:

$$a_m(t) = Aa(t) + b \tag{2.11}$$

where $a_m(t)$ is the raw accelerometer output, A is the scale factor, b denotes the bias and $a(t)$ stands for the actual acceleration measurement which does not include bias error. For pose estimation purposes, tri-axial accelerometers, composed of three, single-axis, orthogonally mounted linear accelerometers, are employed. The generalization of the simplest single-axis model for tri-axial accelerometers, which accounts for scale factor and bias, reads as [40]:

$$\mathbf{a}_m(t) = \mathbf{A}\mathbf{a}(t) + \mathbf{b} \tag{2.12}$$

where $\mathbf{A} \in \mathbb{R}^{3x3}$ is a matrix that includes the scale factors and non-orthogonality corrections, $\mathbf{b} \in \mathbb{R}^3$ is the matrix containing bias terms, $\mathbf{a}(t)$ is the matrix containing acceleration measured in the absence of bias, from now on also called true value and:

$$\mathbf{a}_m(t) = \begin{bmatrix} a_x(t) \\ a_y(t) \\ a_z(t) \end{bmatrix} \tag{2.13}$$

From (2.12), one can obtain the calibration equation that will provide calibrated accelerometer measurements. The equation states:

$$\mathbf{a}(t) = \mathbf{A}^{-1}\mathbf{a}_m(t) - \mathbf{b} \tag{2.14}$$

Calculation of the matrices will be performed in the calibration software. After $\mathbf{A}^{-1}$ and $\mathbf{b}$ matrices are calculated, Equation 2.14 will be imported in Python and Matlab code in order to calibrate the accelerometer.

2.9.1 Procedure

Raw measurements of the three accelerometer axes were recorded to a text file using `recor_uncalib_data.py` script, described in Appendix A.1.5. Afterwards, $\mathbf{A}^{-1}$ and $\mathbf{b}$ matrices will be calculated using a software called Magneto [41]. In order to ensure best possible quality, each outputted measurement was the average of 50 measurements taken while the IMU is stationary. Function performing this task can be seen in Listing 2.4:

```
def RecordDataPt(ser: SerialPort) -> tuple:
    ax = ay = az = 0.0

    for _ in range(AVG_MEAS): #AVG_MEAS=50
        try:
            data = ser.Read().split()
            ax_now = float(data[3]) / 1000 #from mG to G
            ay_now = float(data[4]) / 1000
            az_now = float(data[5]) / 1000
        except:
            ser.Close()
            raise SystemExit("[ERROR]: Error serial connection.")
        ax += ax_now
        ay += ay_now
        az += az_now

    return (ax / AVG_MEAS, ay / AVG_MEAS, az / AVG_MEAS)
```

Listing 2.4: Function for recording accelerometer measurements.

RecordDataPt function was called in the main loop to calculate the average of the measurements.
During the measurement procedure, the IMU had to be fully stationary. Even the slightest movement would have corrupted the measurements and the calibration procedure would have to be performed again. The main loop prompts the user for the command to either perform measurements or to quit and log recorded data to the text file. Function *List2DelimFile* takes the data stored in the *data* vector and prints it to the text file designated by the *FILENAME* input parameter. This procedure was performed as many times as it was necessary to capture different orientations. In this particular case, 277 orientations were recorded.

For the calibration to be successful, one must capture both positive and negative orientations for all three sensitivity axis. Performing less measurements is also possible and would also result in successful calibration as long as positive and negative values are captured. One of the biggest advantages of this calibration method is that it has to be performed only once. If the IMU is to be used for a prolonged period of time, it is recommended to calibrate the IMU again due to the drift and aging of the components [42]. Once enough orientations have been recorded, the text file will be imported to Magneto, which will then calculate the matrices, as seen in Figure 2.13.

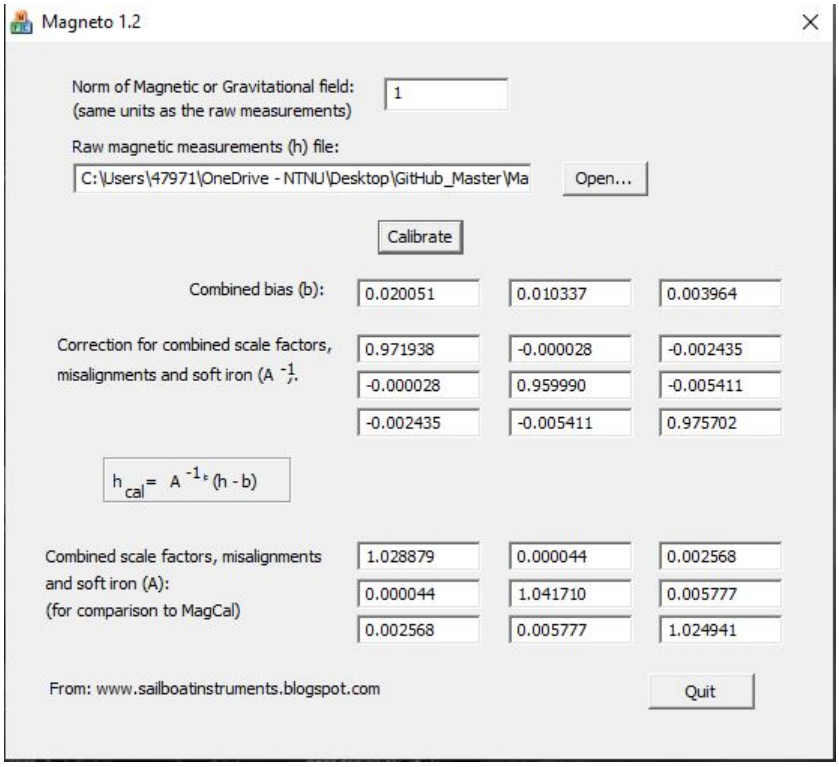

Figure 2.13: Magneto calibration software.

Matrix values of $\mathbf{A}^{-1}$ and $\mathbf{b}$, seen in Figure 2.13, are copied to the `plot_calib_data.py` Python script, described in A.1.5, which plots results of the calibration. As it can be observed in Figures 2.15, 2.16 and 2.17, calibrated values are positioned closer to the center of the plot. Additionally, calibrated data is positioned closer to the horizontal and vertical axis running through origo. This is demonstrated in Figure 2.17. Unfortunately, the calibration method fails to produce same results for all sensitivity axis. Raw measurements in the upper part of the plot seen in Figure 2.15, are grouped closer to the vertical axis than the calibrated measurements.

Before performing the calibration, it was decided that the procedure will be considered successful if it improves the accelerometer performance. Meaning that it is expected to remove as much bias as possible, bringing the measurements closer to their true value. The method can also be considered as successful in cases where the calibration fails, this will only happen if such instances are rare when compared to the overall performance of the calibration. Due to the non-orthogonality of the accelerometer sensitivity axis and the environmental influences, it is impossible to completely remove the bias. It is therefore expected that in some instances, calibrated data will fail to provide an improvement over the raw data. Since the implemented calibration method is relatively simple to perform, it is expected that, in some cases, it will not provide high-end results as would be the case with a more complex method.

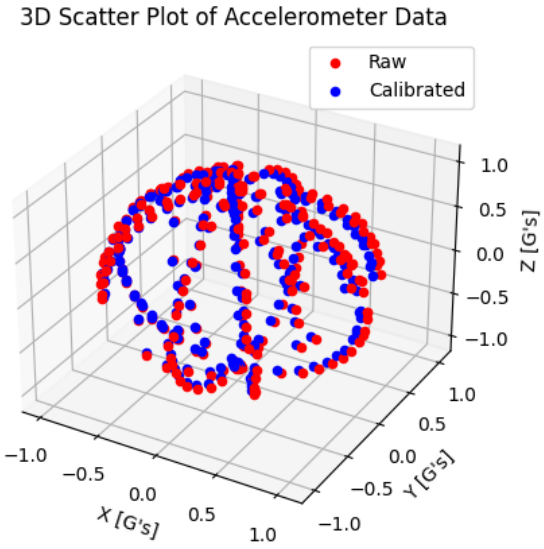

Figure 2.14: 3D representation of the calibration results.

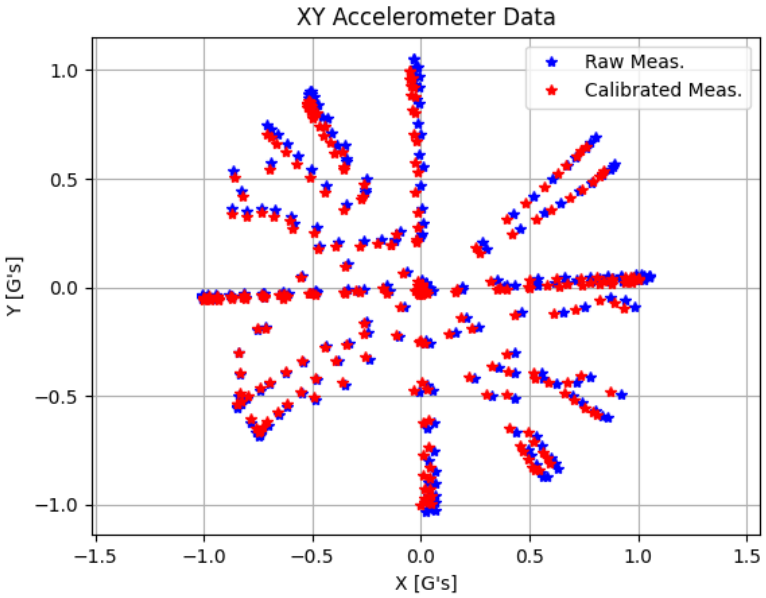

Figure 2.15: XY axis calibration plot.

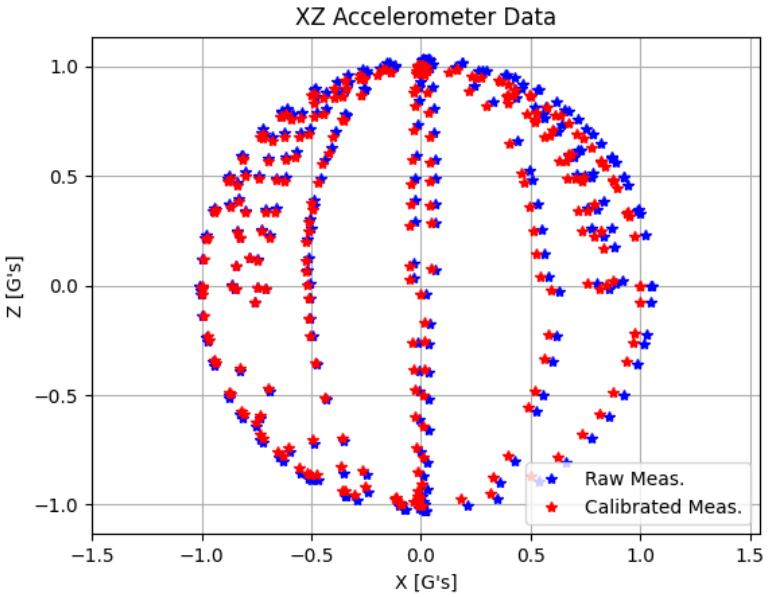

Figure 2.16: XZ axis calibration plot.

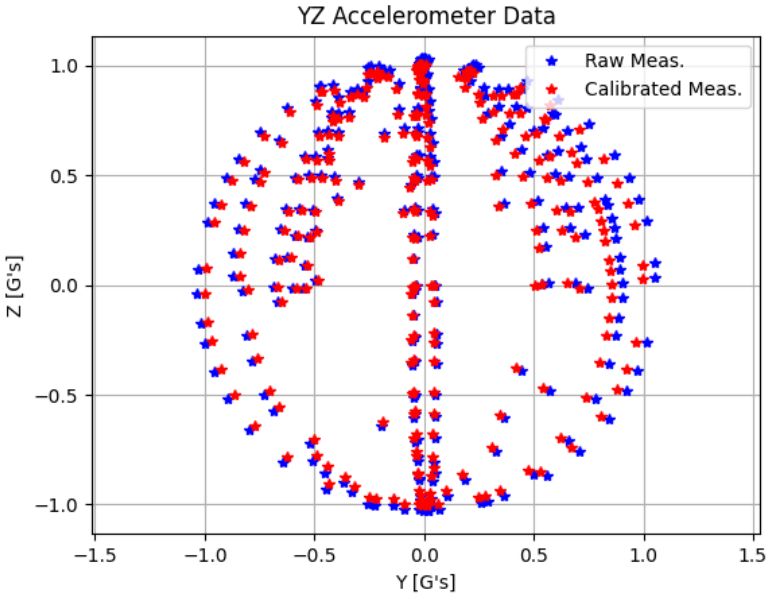

Figure 2.17: YZ axis calibration plot.

Chapter 3

Orientation representation

3.1 Euler angles

Accelerometers and gyroscopes measure gravitational linear velocity and angular velocity. The raw measured data does not provide an intuitive understanding of the orientation of the IMU. With the use of trigonometry, the measured angular velocities ω_x, ω_y, and ω_z and the linear acceleration a_x, a_y, and a_z, will be utilized to generate a representation of the orientation using more comprehensible Euler angles.

According to Euler's rotation theorem, any rotation may be described using three angles. If the rotations are written in terms of rotation matrices, which represent rotation [43]. As explained in Section 2.3, in this report magnetometer will not be used. Therefore, it will not be possible to measure yaw angles, due to the fact that neither accelerometer nor gyroscope can measure the yaw angle reliably. The accelerometer cannot measure yaw angle because rotation around its sensitivity axes does not result in a change of acceleration relative to the gravity, as the gravity vector and a sensitivity axis are aligned. The gyroscope could theoretically measure yaw angle based upon angular velocity measured by the G_x axis. In practice, this is not possible because the gyroscope data would have to be integrated in order to obtain absolute yaw angle. Due to bias drift in the measurements, the gyroscope yaw measurements would drift off from the true value. Additionally, integration of the bias would lead to the error accumulation which would render the method completely useless after certain period of time.

The main technique for converting three-axis specific force from an IMU to roll and pitch angles is based on the idea that the angle between the acceleration and gravity vectors can be calculated using trigonometry. This is a static mapping that suffers from inaccuracies when performing high-acceleration maneuvers [44, p. 21].

$$a^b_{imu} = \mathbf{R}^{bT}_n(\Theta)(a^n_{nm_I} - g^n) + b^b_{acc} + w^b_{acc} \tag{3.1}$$

Where $\Theta = [\theta, \phi, \gamma]^T$ is a vector of Euler angles and $\mathbf{R}^{bT}_n(\Theta)$ is the rotation matrix from $\{b\}$ to $\{n\}$. Accelerometer bias vector and Gaussian white noise are donated as b^b_{acc} and w^b_{acc} respectively [23, p. 330]. The b term in the exponent indicates that the vectors are expressed in IMU's body frame $\{b\}$

From (3.1), roll and pitch angles can be calculated by assuming that the IMU at rest measures $a^n_{nm_I} = 0$. The initial accelerometer biases are generally eliminated by calibrating the accelerometer for temperatures fluctuations. It is also vital to eliminate dynamic drift, which may be accomplished by recalibrating the sensor when the vessel is still. Low-pass filtering should also be used to eliminate measurement noise such that

$$a^b := a^b_{imu} - b^b_{acc} \approx -\mathbf{R}^{b^T}_n(\Theta_{nb})g^n \tag{3.2}$$

From here

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = -\mathbf{R}^{b^T}_n(\Theta_{nb}) \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} gsin(\phi) \\ -gcos(\phi)sin(\theta) \\ -gcos(\phi)cos(\theta) \end{bmatrix} \tag{3.3}$$

Equation (3.3) gives the following ratios

$$\frac{a_y}{a_z} \approx tan(\theta), \quad \frac{a_x}{g} \approx sin(\phi), \quad \frac{a_y^2 + a_z^2}{g^2} \approx cos^2(\phi)$$

This gives static roll and pitch angles as a function of specific force

$$\theta = arctan\left(-\frac{a_y}{a_z}\right) \tag{3.4}$$

and

$$\phi = arctan\left(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}\right) = arctan\left(\frac{a_x}{a_y}\right) \tag{3.5}$$

Alternatively, pitch angle can also be obtained by

$$\phi = sin(a_x) \tag{3.6}$$

Euler angles can also be obtained through the integration of the gyroscope measurements

$$\theta = \omega_x(0) + \int_0^t \omega_x(t)dt \tag{3.7}$$

where $\omega_x(0)$ is the angular velocity value.

3.2 The gimbal lock

Because of the definition of Euler angles, certain orientations of the IMU will lead to gimbal lock. Gimbal lock occurs when one of the IMU axis is aligned with the gravitational vector, thus one of the degrees of freedom is lost. Meaning that the axis is parallel with the gravitational vector and can no longer measure the difference in the linear acceleration. Singularity and gimbal lock are two distinct phenomena that are collectively referred to as the gimbal lock. Despite the fact that the two phenomena are called by the same name, they are different from each other. At the Euler angle singularity, there is an obvious loss of a degree of freedom. However, there is no obvious loss of physical degree of freedom. With this said, the gimbal lock is the term that can be associated with the physical loss of a degree of freedom, while singularity is the term used to describe the gimbal lock mathematically [45, p. 34], that is, when the Jacobian of the system is no longer full rank due to the orientation of the system.

Chapter 4

Digital filtering methods

4.1 Low-pass filter

The results shown in Section 2.8 demonstrate that the accelerometer measurements suffer from the white noise. This is even more apparent when for example observing the plot in Figure 6.5 In addition to the white noise, the IMU proved itself to be extremely sensitive to the disturbances coming from the environment in the form of various vibrations. To improve the quality of accelerometer measurements, it is important to implement a digital filter that will filter out any unwanted frequencies contained within the output data. In case of the IMU, the accelerometer will measure all linear accelerations, not only gravitational acceleration. Thus, any artificial vibrations induced by, for example, a hammer hitting the crane link, will be picked up by the accelerometer and will corrupt the measurements. This also means that if the accelerometer is to be used to find the orientation of the crane link, the IMU must be positioned in the center of the rotational axis. If the IMU is not positioned in the center of the rotational axis, even the slightest rotational movements will induce linear acceleration, which will be recorded by the accelerometer.

The accelerometer's sensitivity was investigated by placing the IMU on the table and recording accelerometer data. The plot in Figure 4.1 demonstrates the obtained results. In the interval between 3 and 8 seconds, the IMU was subjected to the man made vibrations. Within the interval of 3 to 6 seconds, vibrations were created by slightly tapping the table with the metal bar. The vibrations seen in the interval between 7 and 8 seconds are the result of shaking the IMU from side to side. From the Figure 4.1, it can be seen that the accelerometer is very sensitive and that it registers even the slightest vibrations induced on the table surface. In the environment like the one found on the ship deck, vibration induced by humans and other machines are certain to occur. A low pass filter therefore needs to be implemented in order to smooth out accelerometer measurements as much as possible. The plot in Figure 4.2 indicates that the gyroscope will also capture the same vibrations. Thus, it is also possible that the filtering algorithm will have to be implemented to the gyroscope measurements as well. In addition to the digital low-pass filter, it would also be beneficial to physically isolate the IMU from any vibration sources found in the near vicinity.

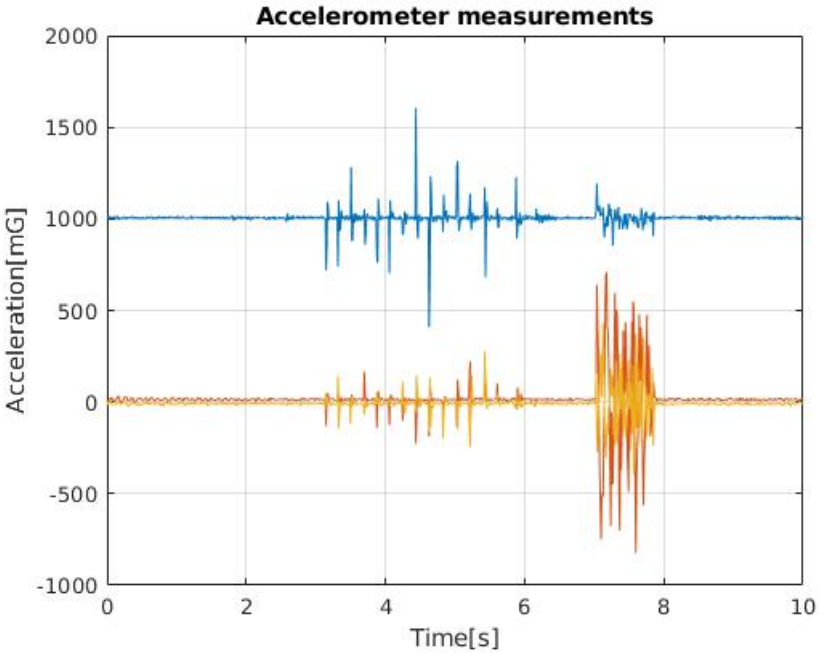

Figure 4.1: Man made vibrations captured by the accelerometer.

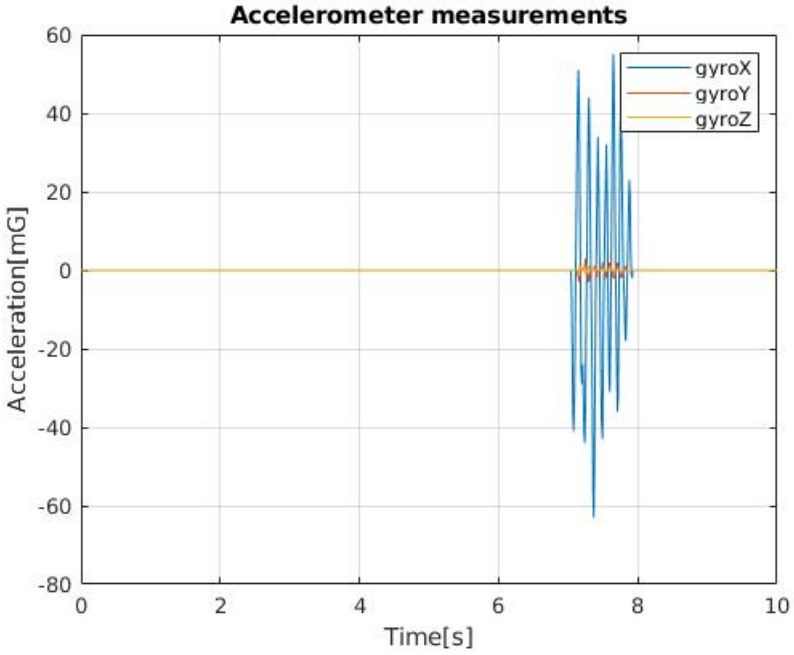

Figure 4.2: Man made vibrations captured by the gyroscope.

First order low-pass filter is the first choice of the filter due to its simplicity. The first order low-pass filter will have continuous transfer function:

$$H(s) = \frac{U(s)}{E(s)} = \frac{\omega_0}{s+\omega_0} \tag{4.1}$$

In order to implement this transfer function in the digital environment, it has to be discretized using discretization methods. For this task, multiple discretization methods can be used. Some of them are zero order hold, bilinear method, trapezoidal approximation method, pole-zero matching etc. In the case of this low-pass filter, it was decided to use the bilinear method also known as the Tustin's method.

4.1.1 Tustin's method

As it can be seen in Figure 4.3, Tustin's method is the discrete integration method which approximates numerical integration by taking a straight line between sampling points [46, p. 26].

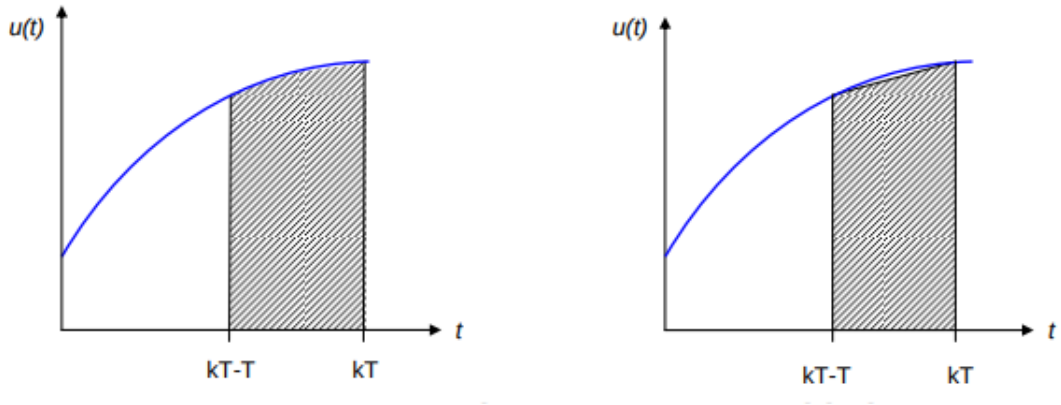

Figure 4.3: Numerical integration using Tustin's method, from [46, p. 26]

If (4.1) is to be used to design a filter with the equivalent behaviour, the system can be defined by the first order differential equation

$$u(t)(s+\omega_0) = e(t)\omega_0 \tag{4.2}$$

this gives

$$\dot{u}(t) + \omega_0 u(t) = e(t)\omega_0 \tag{4.3}$$

Equation 4.3 can be solved by evaluating the integral

$$u(t) = \omega_0 \int_0^t (e(\tau) - u(\tau))d\tau \tag{4.4}$$

The output of the continuous system will be observed at the time points $t \in kT$, where $k = \mathbb{Z}$ and T is the sampling interval. To determine the integral, area of the shaded region has to

be determined. The area of the shaded trapezoid depicted in the Figure 4.3 is given by the equation:

$$u(kT) = \omega_0 \int_0^{kT} (e(t) - u(t))dt \tag{4.5}$$

Equation (4.3) can be devided into two parts

$$u(kT) = \omega_0 \int_0^{kT-T} (e(t) - u(t))dt + \omega_0 \int_{kT-T}^{kT} (e(t) - u(t))dt \tag{4.6}$$

Solving the integral one gets

$$u(kT) = \omega_0(e(kT-T) - u(kT-T)) + \frac{T\omega_0}{2}(e(kT-T) - u(kT-T) + e(kT) - u(kT)) \tag{4.7}$$

This gives

$$u(kT) = u(kT-T) + \frac{\omega_0 T}{2}\left[e(kT-T) - u(kT-T) + e(kT) - u(kT)\right] \tag{4.8}$$

Equation 4.8 is converted to the Z-domain using Z-transform.

$$U(z) = U(z)z^{-1} + \frac{T\omega_0}{2}\left[E(z)z^{-1} - U(z)z^{-1} + E(z) - U(z)\right] \tag{4.9}$$

$$U(z)\left(1 - z^{-1} + \frac{T\omega_0}{2}z^{-1} + \frac{T\omega_0}{2}\right) = E(z)\left(\frac{T\omega_0}{2}z^{-1} + \frac{T\omega_0}{2}\right) \tag{4.10}$$

After rearranging (4.10), the transfer function formulation is obtained

$$H(z) = \frac{U(z)}{E(z)} = \frac{\omega_0}{\frac{T}{2}(\frac{z+1}{z-1}) + \omega_0} = \frac{\omega_0}{\frac{2}{T}(\frac{1-z^{-1}}{1+z^{-1}}) + \omega_0} \tag{4.11}$$

When inspecting (4.1) and (4.11), it can be concluded that the Z-domain equivalent to the continuous transfer function, could be obtained by simple substitution:

$$s \leftarrow \frac{T}{2}\frac{z+1}{z-1} = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}} \tag{4.12}$$

Substituting the s term in (4.1) with the (4.12) gives the following equations:

$$H(z) = \frac{\omega_0}{\frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})} + \omega_0} \tag{4.13a}$$

$$H(z) = \frac{\omega_0 \frac{T}{2}(1+z^{-1})}{(1-z^{-1}) + \omega_0 \frac{T}{2}(1+z^{-1})} \tag{4.13b}$$

$$H(z) = \frac{\omega_0 \frac{T}{2}(1+z^{-1})}{1 - z^{-1} + \omega_0 \frac{T}{2} + \omega_0 \frac{T}{2} z^{-1}} \tag{4.13c}$$

$$H(z) = \frac{\omega_0 \frac{T}{2}(1+z^{-1})}{(1+\omega_0 \frac{T}{2}) - (1-\omega_0 \frac{T}{2})z^{-1}} \tag{4.13d}$$

Equation (4.13d can be represented with the constant coefficient difference equation, which has the form

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} \tag{4.14}$$

Equation (4.13d) has the following coefficients:

$$b_0 = \omega_0 \frac{T}{2}$$
$$b_1 = \omega_0 \frac{T}{2}$$
$$a_1 = (1 - \omega_0 \frac{T}{2})$$

Taking the coefficients from (4.14), the difference equation can be derived

$$y[n] = a_1 y[n-1] + b_0 x[n] + b_1 x[n-1] \tag{4.16}$$

Equation (4.16) will be later used in the `lowPassfilter.py` script, described in Appendix A.1.7, in order to perform real time filtering of the input signal. In order to ensure precision and reduction of the fail margin, Python's *numpy* and *scipy* libraries will be used to discretize the continuous transfer function of the first order low-pass filter. The same equation can be used in order to perform post-processing of the prerecorded data.

4.1.2 Stability of the method

As mentioned in Subsection 4.1.1, there are multiple methods that perform discretization of the continuous transfer functions. Each method has its positives and negatives, taking into consideration complexity of the method, computational cost and stability. Due to the fact that the first order low-pass filter equation is relatively simple, computational cost of the discretization method can be ignored. Therefore, it was decided to base the choice of the discretization method on the stability characteristics. Tustin's method maps the entire left-half plain (LHP) of the Laplace domain, to the unit circle in z-domain, as seen in Figure 4.4.

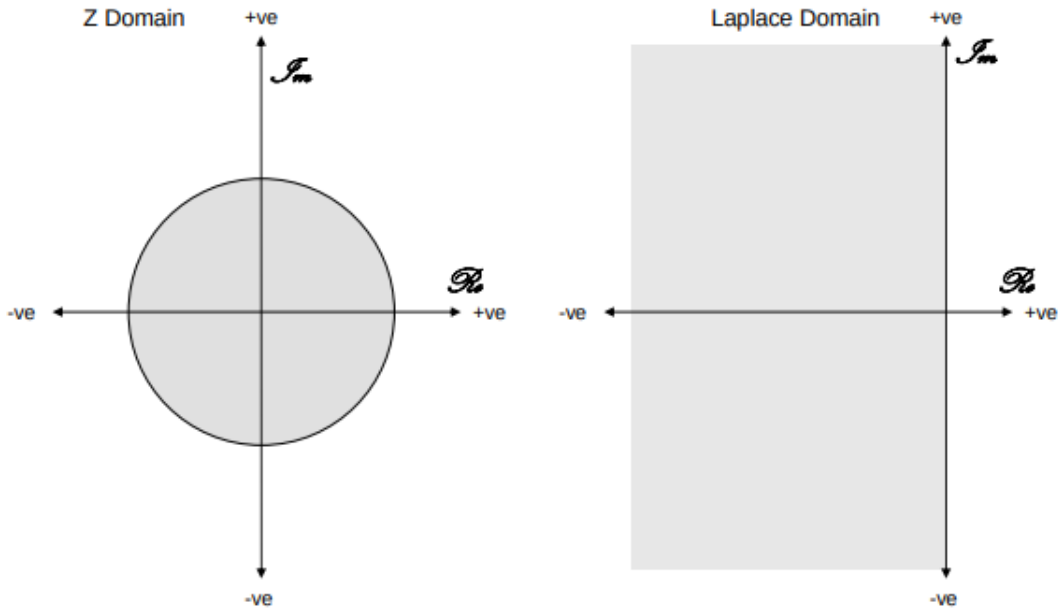

Figure 4.4: Mapping of the LHP to the unit circle in z-domain using Tustin's method, from [46].

The plot in Figure 4.4 indicates that the entire LHP of the Laplace domain is projected to the stability unit circle of the discrete domain. This property is extremely important, since it ensures that the stability characteristics of the continuous system are preserved when converted to the discrete domain. This is not always an instance with other discretization methods. Forward approximation method does not ensure complete mapping of the stable poles and zeros from the continuous to the discrete domain. The reason for this property is due to the working principle of the forward approximation method. Tustin's method evaluates the function at the current sampling point kT and the previous sampling point $kT - T$. The forward approximation method evaluates the function at the current sample kT and the next sampling point $kT + T$. Since Tustin's method evaluates the function at the two known sampling points, it will remain stable even if the sampling frequency is low. Meaning that Tustin's method allows for much lower sampling frequency while the method still remains stable. This is not the case with the forward approximation method due to the fact that it requires estimated value of the function at the current step kT and the next step $kT + T$. Quality of the estimation for the next step is dependent on the sampling frequency and the derivative of the function. If the sampling frequency is too low or the derivative of the function is large, the forward approximation method will fail to produce sufficient estimation and the method will become unstable.

4.1.3 System frequency

Design of the digital filter requires the cutoff frequency to be specified. Filter cutoff frequency is the frequency at which damping is equal to -3 dB. The cutoff frequency has to be chosen in order to ensure damping of the unwanted frequencies. With this said, it is important to have an insight into the frequency at which the system operates. This is crucial to understand, since the filter must not filter out the system's frequencies. The Nyquist-Shannon criterion states [47]:

$$\omega_s \geq 2\omega_{max} \tag{4.17}$$

Where ω_s is the sampling frequency and ω_{max} is the maximum frequency of the system. The same principle, as seen in (4.17), can be applied to the filter's cutoff frequency. If aliasing is to be avoided, the filter's cutoff frequency has to be at least two times greater than the maximum frequency of the system. In practice, due to imperfections in the filter's attenuation rate, the cutoff frequency should be larger than suggested.

$$\omega_s \gg \omega_{max} \tag{4.18}$$

Unfortunately, it was not possible to measure motion of the aquaculture crane, since access to the crane itself was not possible. Measuring the crane's movement is crucial to obtaining precise information regarding the crane's bandwidth. This information would later be used to determine a suitable cutoff frequency for the low-pass filter. Since this is not possible, the crane's movement has to be assumed, as shown in the project report [48, p. 24]. In the project report, the IMU was rotated around an imaginary axis, assuming that this motion of the IMU would correspond to the motion of the crane link. Data captured by the accelerometer was imported to Matlab, where Figure 4.5 was obtained.

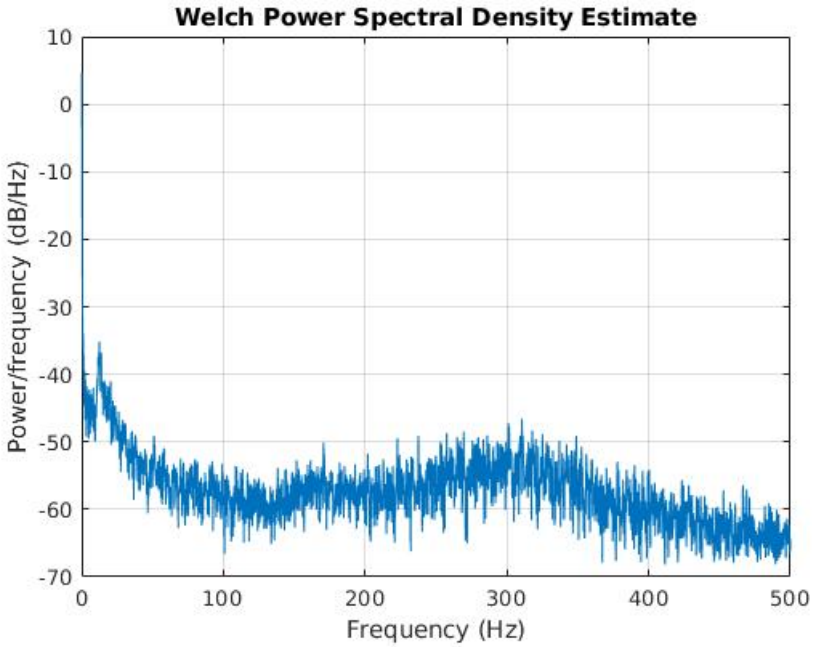

Figure 4.5: Frequency domain analysis, from [48, p. 24]

The plot in Figure 4.5 shows one dominant frequency component. The sudden spike at 12.5 Hz corresponds to the frequency of the motion performed by the IMU. Therefore, it is assumed that the crane system operates at a frequency of 12.5 Hz, with an assumed maximum frequency of 15 Hz. With this said, it is decided that the suitable cutoff frequency would be 75 Hz. This cutoff frequency is five times larger than the estimated maximum frequency, thus obeying the Nyquist-Shannon criterion and leaving sufficient margin if the maximum frequency is larger than 15 Hz.

4.2 Butterworth filter

The low-pass filter described in Section 4.1 performs the duty of filtering out unwanted frequencies, most notably the ones coming from the white noise and the vibrations from the environment. Due to the fact that access to the crane was not allowed, it is very likely that some other type of low-pass filter would be needed. The low-pass filters like Butterworth, Chebyshev and Bessel are popular filter types used when more advanced filtering method has to be implemented.

Butterworth filter has the maximally flat response within the filter's pass-band, optimized for gain flatness. The attenuation is -3dB at the cutoff frequency. Additionally, it has a moderate phase distortion. Chebyshev filters are designed with the ripple in the pass-band amplitude response, but steeper roll-off at the cutoff frequency. Steeper roll-off will also result in monotonicity in the pass-band region, along with poor transient response. Bessel filter is optimized for maximally flat time delay, meaning that this type of filter has linear phase response and excellent transient response to a pulse input. Cutoff frequency is defined as the -3dB point [49]. Due to their respective characteristics, it was decided that the Butterworth filter would be the most optimal choice of filter. Maximal flatness of the passband region, in addition to moderate phase delay, are the reasons why Butterworth filter would be the most optimal filter type. The IMU measurements are not pulses, therefore Bessel filter will not be used. According to Dessen in [50], (4.19) can be used in order to calculate order of the Butterworth filter based solely upon the filter's passband and stopand requirements.

$$n \geq \frac{\log \frac{D_s^2-1}{D_d^2-1}}{2\log \frac{\omega_s}{\omega_d}} \tag{4.19}$$

where D_s and D_d are attenuation values for the passband and stopband, ω_s and ω_d are passband and stopband frequencies. For the purpose of filtering the IMU measurements, it is decided that the filter will have less than 2 dB attenuation in the passband for the frequencies below the cutoff frequency of 75 Hz, while attenuation above the stopband frequency of 350 Hz is going to be more than 50 dB. The values are inserted in (4.19) and the filter order is obtained

$$n \geq \frac{\log \frac{10^{\frac{50\cdot 2}{20}}-1}{10^{\frac{2\cdot 2}{20}}-1}}{2\log \frac{2199.11}{471.23}} = 3.91$$

The order has to be an integer which is equal or greater than the n. Dessen [50] proposes that the 5th order Butterworth filter is the best choice of filter order. A 5th order filter would provide the best trade-off between phase delay and the dampening factor. Due to the implementation procedure for the digital Butterworth filter, it was decided that a 4th order Butterworth filter is going to be designed. Continuous 4th order Butterworth filter has the transfer function.

$$H(s) = \frac{1}{(s^2 + 0.765367s + 1)(s^2 + 1.847759s + 1)} \tag{4.20}$$

The 4th order Butterworth filter, seen in the Bode plot in Figure 4.6, is designed with the cutoff frequency f_c of 75 Hz and sampling frequency f_s of 500 Hz.

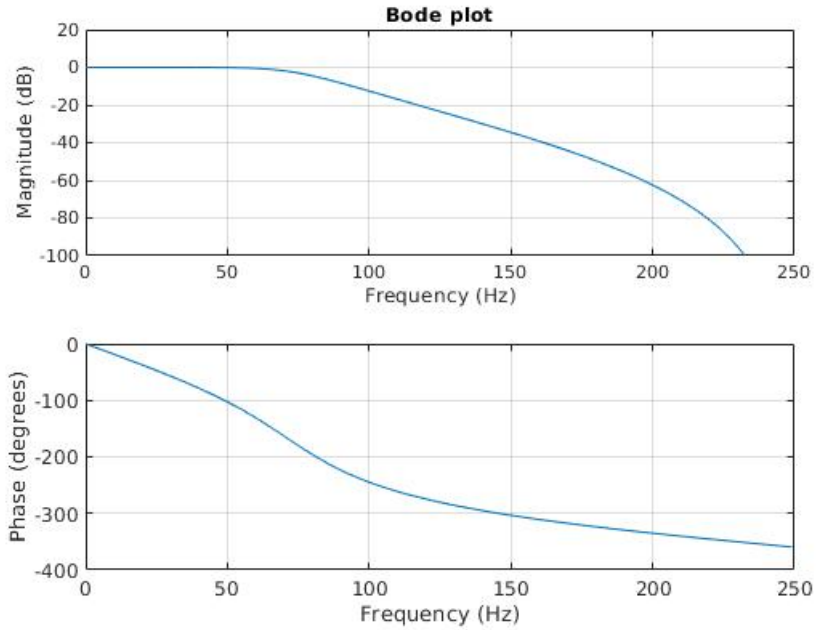

Figure 4.6: Bode plot of 4th order Butterworth filter.

4.2.1 Digital Butterworth filter

A higher order Butterworth filter will provide higher attenuation rate and will allow for more precise filtering of the frequencies above the cutoff frequency. Higher order filters will have to be implemented as a cascade of multiple low order filters. In the case of a 4th order Butterworth filter, it will be implemented as a cascaded sequence of two 2nd order Biquad filters [51, p. 512]. For the higher order filters, digital implementation, as seen in Section 4.1, would be difficult to implement. This is because direct implementation, as seen in (4.16), would have a and b coefficients that differ in several orders of magnitude. A digital filter, implemented using a difference equation, would result in the stability of the filter being dependent on the floating point precision of the computing algorithm. Single point floating precision will cause quantization error, which would result in the filter becoming unstable. High order filters are sensitive to quantization error, particularly for lower cutoff frequencies. Cascaded implementation of the higher order filters has coefficients which are less different in orders of magnitude, thus single point precision will not cause rounding errors.

Biquad filter

The implementations of the digital Butterworth filter will be performed using Biquad filters. Biquad filter is a 2nd order recursive filter. It realizes an arbitrary biquadratic transfer function, thus the name Biquad filter. A generic linear second order filter which may be described in terms of its action on a stream of samples by the following recurrence relation, given by [52] as:

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} - a_1 y_{n-1} - a_2 y_{n-2} \tag{4.21}$$

where x_n denotes the current input sample, x_{n-1} is the previous sample and y_n is the output. Taking the Z-transform of (4.21) gives a general discretized transfer function for the Biquad filter

$$H(z) = K \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.22}$$

Equation (4.22) can be represented by the difference equation:

$$y[n] = K_1 b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2] \tag{4.23}$$

The difference equation (4.23) can be implemented in the digital environment using Direct form I approach [53], as seen in Figure 4.7.

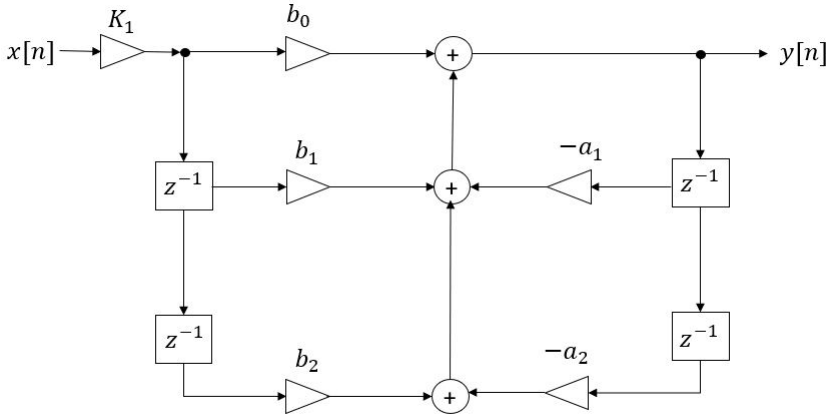

Figure 4.7: Direct form I block diagram.

The discretized transfer function, for the Biquad filter, is given by [54] as:

$$B(z) = K \frac{(1+z^{-1})^2}{(1-p_k z^{-1})(1-p_k^* z^{-1})} = K \frac{1 + 2z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.24}$$

where p_k^* is the complex conjugate of p_k. Gain of each Biquad section has to be equal to 1 at frequency of $\omega = 0$. Letting $z = e^{j\omega}$, then the $z = 1$. This gives:

$$H_k(z) = 1 = K_k \frac{\sum b}{\sum a} \tag{4.25}$$

this can further be simplified to give the expression for the gain values

$$K_k = \frac{\sum a}{4} \tag{4.26}$$

where $a = [1 \quad a1 \quad a2]$ are the denominator coefficients of the Biquad section. This gives the values for each of the coefficients:

$$b = [1 \quad 2 \quad 1] \tag{4.27a}$$
$$a = [1 \quad -2real(p_k) \quad |p_k|^2] \tag{4.27b}$$
$$K = \frac{\sum a}{4} \tag{4.27c}$$

The coefficients and gain values seen in (4.27) are calculated using `biquad_sinth.m` script, described in Appendix A.2. The transfer functions obtained for the two Biquad filters produce the magnitude response as seen in the Figure 4.8. The response of the first filter has the desired attenuation of -3dB at the cutoff frequency, while the magnitude response of the second filter shows that it is slightly underdamped when compared to the first one. For a stable digital filter, all poles of the $B(z)$ have to be placed inside the unit circle, which implies the following conditions for a_1 and a_2;

$$|a_1| < 2, \quad |a_1| - 1 < a_2 < 1 \tag{4.28}$$

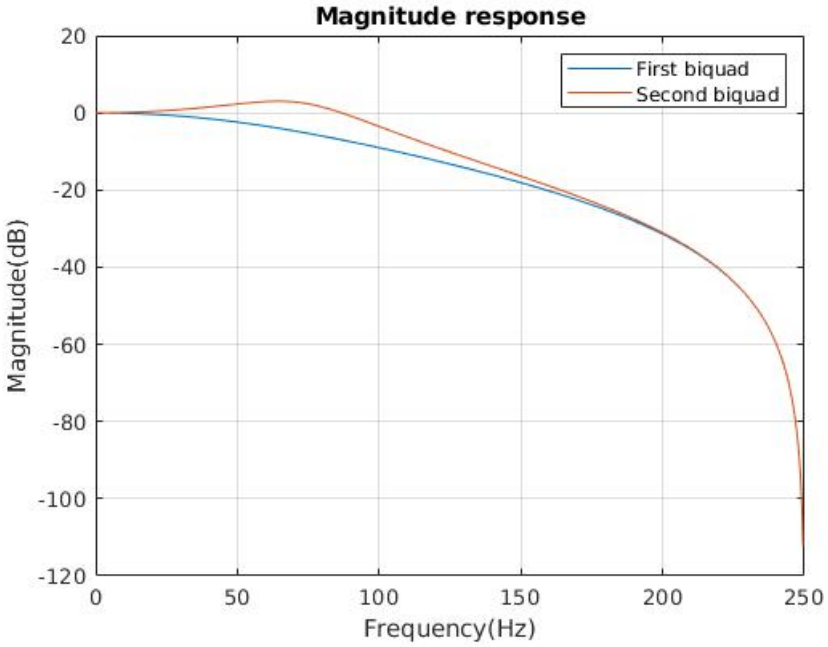

Figure 4.8: Individual magnitude response for each Biquad section.

Implementing two cascaded Biquad filters will allow for -3dB attenuation at the cutoff frequency, as seen in Figure 4.9.

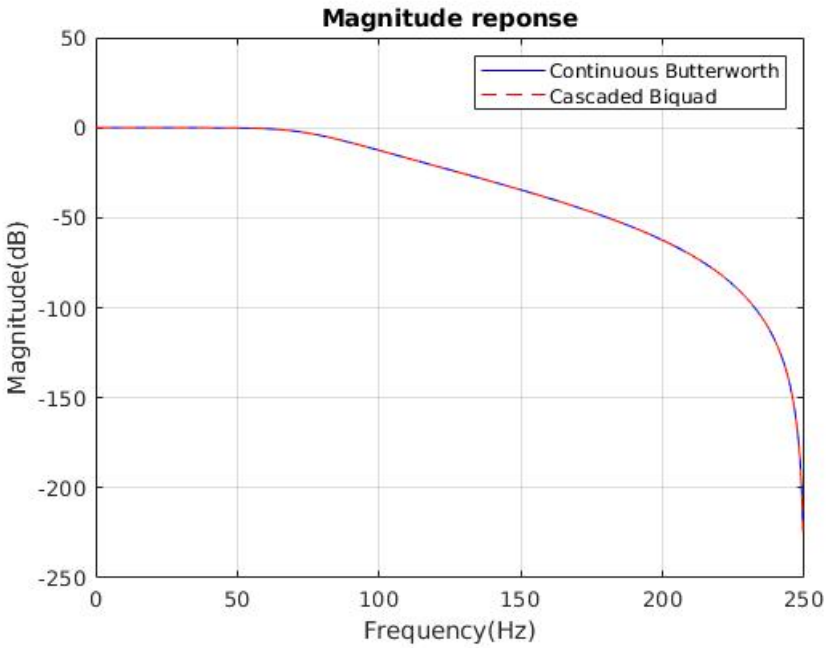

Figure 4.9: Magnitude response of the cascaded Biquad filters and the continuous Butterworth filter

The cascaded implementation of the two 2nd order Biquad filters results in the combined magnitude response, which is completely identical to the one obtained from the continuous Butterworth filter. The Biquad filters allow for the correct implementation of the Butterworth filter. Cascading two 2nd order Butterworth filters would be an incorrect implementation, since attenuation at the cutoff frequency would be -6 dB.

4.3 Complementary filter

4.3.1 Methodology

The most crucial aspect of any autonomous vehicle operating in an unknown environment is the ability to estimate the vehicle's orientation in space. For this purpose, a vast variety of sensors and signal methods can be used, and most of them have one common modality, which is reliance on the inertial sensors. Triaxial accelerometers and gyroscopes are extensively used for the purpose of providing orientation determined by roll, pitch and yaw angle. There are a large variety of methods for orientation estimation, but in general two major groups can be distinguished [55]. First, the Kalman filter based algorithms use a series of data observed over time, which contain noise and other inaccuracies, to estimate unknown variables with more accuracy [56]. Second, the complimentary filter based solutions, which offer more direct solution that is not computationally demanding as it is the case with Kalman filter.

As mentioned in Section 2.5, all sensors will suffer from stochastic and deterministic errors. This is also the case for accelerometers and gyroscopes. Both sensors are capable of providing orientation with respect to some frame of reference. Unfortunately, individual use of accelerometer or gyroscope will not provide reliable orientation values due to their inherent weaknesses. In general, accelerometers will suffer from the short term disturbances like white noise, while gyroscopes will suffer from the long term disturbances like bias drift. If a gyroscope alone is to be used to obtain orientation, angular velocity will have to be integrated. Due to bias drift, orientation estimates will drift over time, causing the method to become almost useless after a long period of time. Accelerometer angle estimations will suffer from the short term weaknesses like white noise, which will cause angle output to fluctuate. Unlike the gyroscope, the accelerometer will provide long term stability due to the fact that the gravity vector does not change with time.

To improve the performance of both sensors, the sensor fusion algorithm must be implemented. Sensor fusion is a method of integrating signals from multiple sources into a single signal or information [57]. Since each sensor output contains noise and measurement errors, multiple sensors can be used to determine the same property, but with the consensus of all sensors. This way, sensor uncertainty can be reduced [58]. In the case of accelerometer and gyroscope, sensor fusion would allow for merging of both sensors into one measurement unit. The resulting output of the sensor fusion would not suffer from the pre-mentioned weaknesses. The complimentary filter algorithm performs the merge of the gyroscope and accelerometer in a way that the accelerometer compensates for the weaknesses of the gyroscope and vice versa. The principal of the operational method for the complementary filter can be described by the block diagram in Figure 4.10.

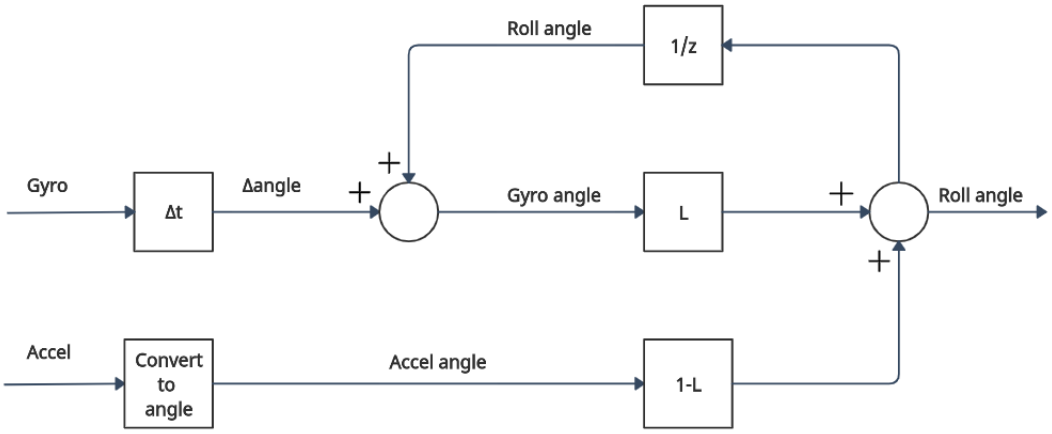

Figure 4.10: Complimentary filter block diagram, from [59].

In Figure 4.10, the constant L represents the fixed fraction determining which sensor is going to perform low frequency, short term corrections. Since the accelerometer suffers from the short term weaknesses, this constant would be greater for the gyroscope part. As seen in `complimentaryFilter_preRec.py` script, described in Appendix A.1.4, the constant L is set to 0.90 which corresponds to the gyroscope performing 90% of angle estimation in the short term. The block diagram in Figure 4.10, will be implemented in the code in the form of the equation given by [59] as:

$$\text{angle} = \left(1 - L\right) \cdot \text{accel angle} + L \cdot \left(\text{gyro angle} \cdot \Delta t + \frac{\text{angle}}{z}\right) \tag{4.29}$$

The complementary filter combines two measurements that complement one another. Since the accelerometer suffers from high frequency white noise, and the gyroscope suffers from low frequency bias drift, the complimentary filter runs these two measurements through the low-pass and high-pass filters, thus eliminating low and high frequency disturbances. The L tunes the cutoff frequency of the low-pass and high-pass filters, thus determining which frequencies will be present in the output signal. The working principle can be seen in Figure 4.11.

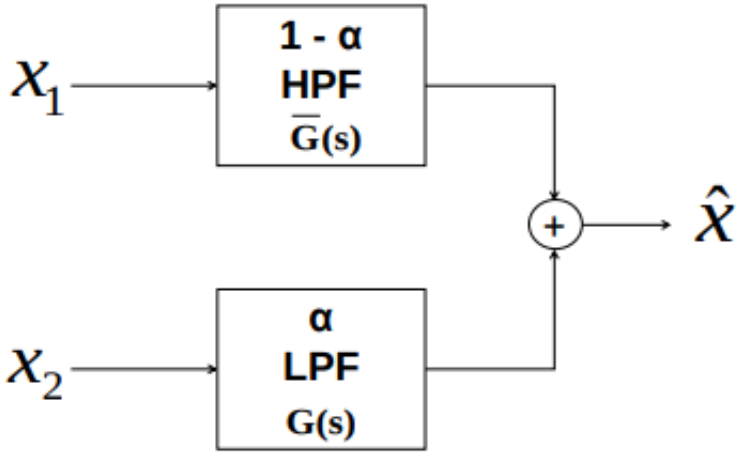

Figure 4.11: Structure of the complimentary filter.

Chapter 5

Mathematical modeling

5.1 Crane load

In this section, a mathematical model is derived for the load suspended on the crane hook, using Lagrange mechanics. This entire chapter will be based on the mathematical equations described in [60]. Lagrange mechanics is a powerful tool used to derive mathematical models of the mechanical systems based upon kinetic and potential energy of the system, conventionally labeled as T and V, respectively. A mathematical model of the system is described by a vector of time-varying generalized coordinates donated $q(t) \in \mathbb{R}^{n_c}$, that must be capable of describing the 'configuration' of the system at a given time t. Coordinates at a given time t are providing information regarding the state of the system at a particular point in time.

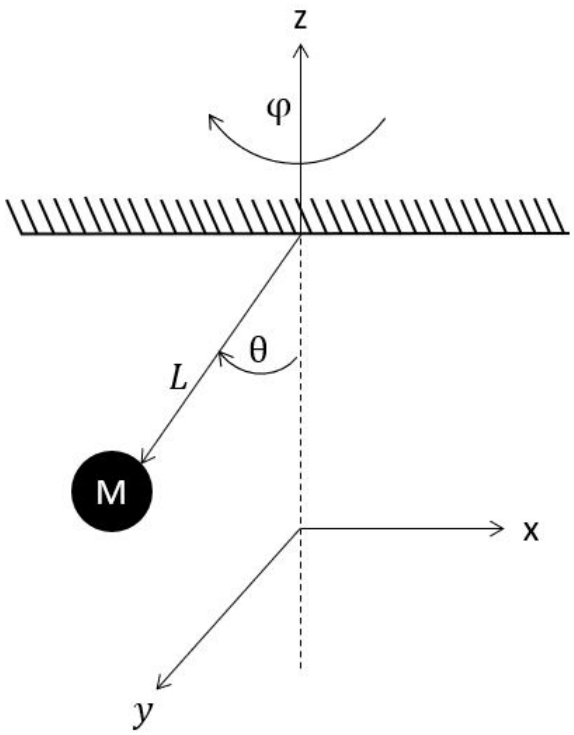

Figure 5.1: Sketch of spherical pendulum.

Before mathematical equations were derived, the following assumptions were made:

- Crane load is connected to the crane hook using a massless rod.
- The position of the crane hook (end-effector) is known.
- No external forces are acting upon the load.
- All friction is neglected.

Table 5.1: Notation for the spherical pendulum model.

Notation	Description
θ	Load sway angle
ϕ	Load yaw angle
M	Mass of the load
L	Length of the hoisting rod

The assumption given above provided the basis for choosing the generalized coordinates vector $q(t) \in R^3$

$$q(t) = [\phi \quad \theta \quad L]^T \tag{5.1}$$

To simplify, explicit dependence of the generalized coordinates and other time-varying objects, e.g. time-varying generalized coordinates $q(t)$, will be simply noted as q.

5.1.1 Position

As seen in the plot in Figure 5.1, the position of the load can be described as

$$\mathbf{p} = \begin{bmatrix} Lsin\theta cos\phi \\ Lsin\theta sin\phi \\ Lcos\theta \end{bmatrix} \tag{5.2}$$

5.1.2 Kinetic energy

The equation for the kinetic energy is given by

$$\mathbf{T} = \frac{1}{2}M||\mathbf{p}||^2 = \frac{1}{2}M\dot{\mathbf{p}}^T\dot{\mathbf{p}} \tag{5.3}$$

where

$$\dot{\mathbf{p}} = \frac{\partial \mathbf{p}}{\partial \mathbf{q}}\dot{\mathbf{q}} \tag{5.4}$$

Performing derivation of the position, described in 5.4, the velocity matrix is obtained

$$\dot{\mathbf{p}} = \begin{bmatrix} -Lsin\theta sin\phi\dot{\phi} + Lcos\theta cos\phi\dot{\theta} + cos\phi sin\theta\dot{L} \\ Lcos\phi sin\theta\dot{\phi} + Lcos\theta cos\phi\dot{\theta} + sin\phi sin\theta\dot{L} \\ cos\theta\dot{L} - Lsin\theta\dot{\theta} \end{bmatrix} \tag{5.5}$$

The derivative of the position seen in (5.5) is inserted into (5.3) and gives

$$\begin{aligned} T = \frac{1}{2}M\Bigg[\Big(\dot{L}cos\theta - L\dot{\theta}sin\theta)^2 + \Big(\dot{L}cos\phi sin\theta + L\dot{\theta}cos\phi cos\theta - Lp\dot{h}isin\phi sin\theta\Big)^2 \\ + \Big(\dot{L}sin\phi sin\theta + L\dot{\phi}cos\phi sin\theta + L\dot{\theta}cos\theta sin\phi\Big)^2\Bigg] \end{aligned} \tag{5.6}$$

After cancelation of the terms

$$T = \frac{1}{2}M\Big(L^2\dot{\theta}^2 + L^2\dot{\phi}^2 sin^2\theta\Big) \tag{5.7}$$

5.1.3 Potential energy

From Figure 5.1 it can be seen that the potential energy is described as

$$V = -Mg\begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} \tag{5.8}$$

5.1.4 Lagrangian

Lagrange function is defined as

$$\mathscr{L}(q,\dot{q}) = T(q,\dot{q}) - V(q) \in \mathbb{R} \tag{5.9}$$

The Euler-Lagrange equation then reads as:

$$\frac{d}{dt}\frac{\partial\mathscr{L}}{\partial\dot{q}} - \frac{\partial\mathscr{L}}{\partial q} = 0 \tag{5.10}$$

After performing derivation and partial derivation, the equation of position is obtained

$$\frac{d}{dt}\frac{\partial\mathscr{L}}{\partial\dot{q}} - \frac{\partial\mathscr{L}}{\partial q} = \begin{bmatrix} \ddot{\theta} - \dot{\phi}^2 cos\theta sin\theta + \frac{g}{L}sin\theta \\ \ddot{\phi}sin^2\theta + 2\dot{\phi}sin\theta cos\theta\dot{\theta} \end{bmatrix} = 0 \tag{5.11}$$

Equation (5.11) gives the equation of the system acceleration.

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{\phi}^2 cos\theta sin\theta - \frac{g}{L} sin\theta \\ -2\dot{\phi}\frac{cos\theta}{sin\theta}\dot{\theta} \end{bmatrix} \tag{5.12}$$

5.2 Explicit 4th order Runge-Kutta method

Ordinary differential equations (ODE) presented in Section 5.1 will have to be solved in order to obtain values for the system acceleration, derived in (5.12). Due to the nature of the ODE, it is not possible to compute the solution numerically or otherwise, from the initial conditions to the arbitrary time. ODEs have a unique solution from the initial conditions to the arbitrary time. Therefore, if approximation of the solution is to be obtained, one must implement one of the numerical integration methods. The most commonly used method for the numerical integration is the Runge-Kutta (RK) integration method. Due to their relative computational simplicity, it was decided to use explicit Runge-Kutta method.

The explicit Runge-Kutta method uses the iterative Euler method in order to approximate solution to the ODE. Efficiency and the computational cost is dependent on the order of the method, which is defined by the stages s. Taking into consideration computational cost and efficiency, it was decided to implement the 4th order explicit Runge-Kutta method, also known as RK4. The number of stages of an explicit RK method defines the computational complexity of evaluating one step $x_k \rightarrow x_{k+1}$ in the integrator. Each integration step performed by the RK requires evaluation of the equations in (5.12). Therefore, it can be concluded that the lower order methods are more preferable, due to their relative balance between efficiency and the computational complexity. RK4 offers the best trade-off between computational complexity (CPU time) and accuracy. It has $s = 4$ stages, thus integration order of this scheme is

$$||x_n - x(T)|| \leq c||\Delta t^o|| \tag{5.13}$$

where $c > 0$ is a constant and o is the order of the method, in this case $o = 4$. The Δt should be small, thus becoming exponentially smaller as the order increases. By increasing order of the method, the computational complexity rises proportionally, but the accuracy is gained exponentially. It is also possible to use higher order methods when estimating the solution of the ODE. After the observation of (5.13), it may appear that the increase in stages will produce better approximations. With the explicit RK methods, after the 4th order RK, higher number of stages does not produce exponentially higher rate of error elimination with the respect to computational complexity. For a greater number of stages, the order stops increasing as quickly as the number of stages. This results in increased computational cost that is not justified by the exponential increase in performance of the method. The RK4 can be defined by the Butcher tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

The RK4 method approximates solutions to a differential equation of the form

$$\dot{x} = f(x,u), \quad x(0) = x_0 \tag{5.14}$$

over a time interval $[0,T]$. Approximation of the solution is performed by assembling the RK step

$$x_{k+1} = x_k + \Delta t\left(\frac{1}{6}K_1 + \frac{1}{3}K_2 + \frac{1}{3}K_3 + \frac{1}{6}K_4\right) \tag{5.15}$$

where

$$K_1 = f\left(x_k, u(t_k)\right) \tag{5.16a}$$

$$K_2 = f\left(x_k + \frac{\Delta t}{2}K_1, u\left(t_k + \frac{\Delta t}{2}\right)\right) \tag{5.16b}$$

$$K_3 = f\left(x_k + \frac{\Delta t}{2}K_2, u\left(t_k + \frac{\Delta t}{2}\right)\right) \tag{5.16c}$$

$$K_4 = f\left(x_k + \Delta t K_3,\ u\left(t_k + \Delta t\right)\right) \tag{5.16d}$$

Chapter 6

Results

6.1 Allan analysis

Equations (2.8) and (2.9) were implemented in `AllanDevAnalsis.py`, described in Appendix A.1.1 , which calculated Allan variance and deviation of the IMU accelerometer measurements. Calculation results are plotted in Figure 6.1:

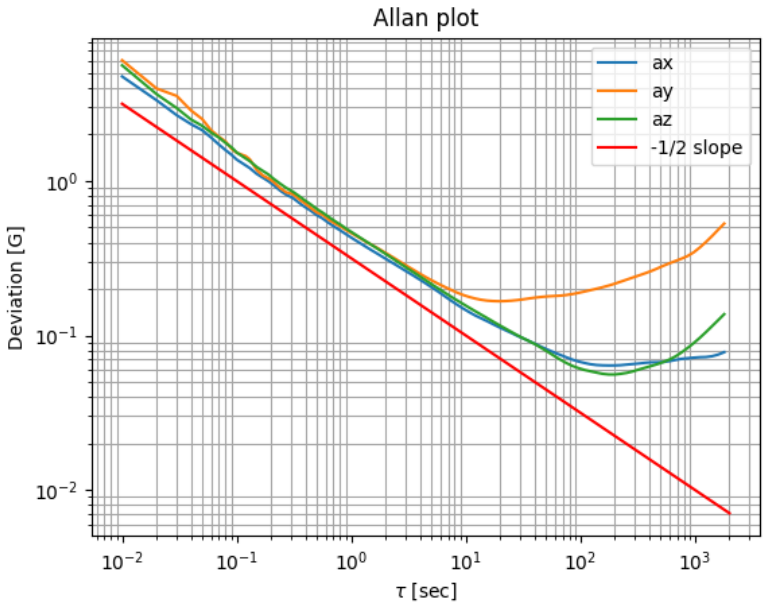

Figure 6.1: Allan deviation plot for the accelerometer data.

The type of noise present in the measurements can immediately be determined just by inspecting the slope of the plotted graphs. As a reference, in addition to three graphs for three acceleration axes, a red line with the slope of $-\frac{1}{2}$ was plotted. Slope of $-\frac{1}{2}$ or -1 indicates that output measurements contain quantization noise. The quantization noise is one of the errors introduced into an analog signal by encoding it in digital form. That noise is caused by the small differences between the actual amplitudes of the points being sampled and the bit resolution of the AD converter [34]. This analysis confirms that the accelerometer output suffers from Gaussian white noise, which is also apparent when examining Figure 2.7.

The same procedure with the same rules can be performed on gyroscope readings. According to MathWorks article [38], the gyroscope measurement model is as follows:

$$\Omega(t) = \Omega_{Ideal}(t) + \text{Bias}_N(t) + \text{Bias}_B(t) + \text{Bias}_K(t) \tag{6.1}$$

Where three noise parameters are N (angle random walk), K (rate random walk), and B (bias instability).

For each sample, gyroscope output angle θ has to be calculated:

$$\theta(t) = \int^t \Omega(t')dt' \tag{6.2}$$

Using (2.8) and (2.9), Allan deviation of the gyroscope data can be plotted.

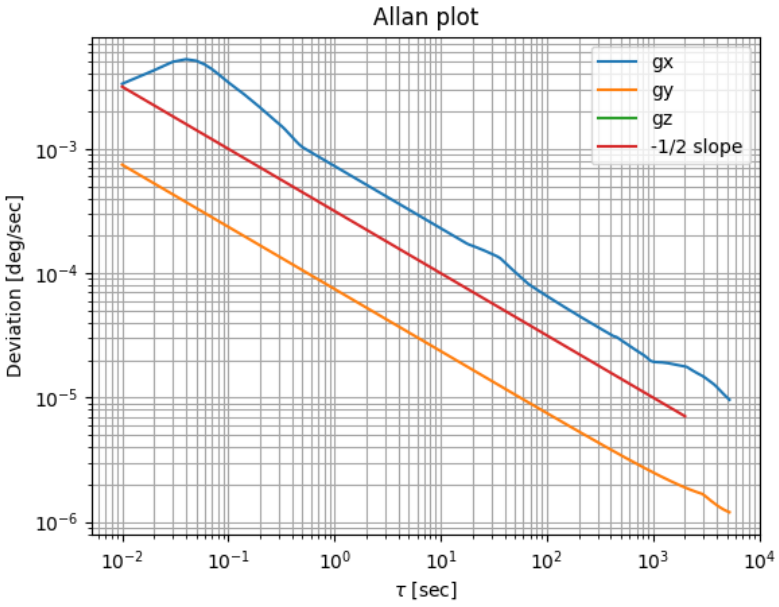

Figure 6.2: Allan deviation plot of the gyroscope data.

Figure 6.2 shows only two plots for the G_x and G_y sensitivity axis. This is because for the duration of 5 hours, the G_z axis did not record a single deviation value. As in Figure 6.1, the red line indicates slope of $-\frac{1}{2}$. It can be seen that the plots indicate the presence of the white noise in the gyroscope measurements. From the plots in Figure 6.2, it is possible to calculate values for the angle random walk and bias instability. In order to calculate angle random walk, Allan deviation at $\tau = 1$ has to be determined. For the G_x axis, the Allan deviation value at time $\tau = 1$ is 0.00069, while for the G_y axis, the deviation value is $7.243 \cdot 10^{-5}$. These values are multiplied by 60 to get angle random walk in $deg/\sqrt{hr}$, as shown in [61]:

$$N_x = 0.00069 \frac{\text{deg}}{s} \cdot 60 \frac{s}{\sqrt{hr}} = 0.0414 \frac{\text{deg}}{\sqrt{hr}} \tag{6.3a}$$

$$N_y = 7.24 \cdot 10^{-5} \frac{\text{deg}}{s} \cdot 60 \frac{s}{\sqrt{hr}} = 0.0043 \frac{\text{deg}}{\sqrt{hr}} \tag{6.3b}$$

where N is the angle random walk coefficient for each of the respected axis. Computing bias instability requires Allan deviation value at the point where the graph's value stops declining at the same rate as before. For the G_x the value $1.91 \cdot 10^{-5}$ is at $\tau = 988$, while for the G_y the value $1.16 \cdot 10^{-5}$ is at $\tau = 5123$, as shown in [61]:

$$B_x = 1.91 \cdot 10^{-5} \cdot \frac{\text{deg}}{s} \cdot \frac{3600}{0.664} \frac{s}{\sqrt{hr}} = 0.1 \frac{\text{deg}}{hr} \tag{6.4a}$$

$$B_y = 1.16 \cdot 10^{-6} \frac{\text{deg}}{s} \cdot \frac{3600}{0.664} \frac{s}{hr} = 0.0063 \frac{\text{deg}}{\sqrt{hr}} \tag{6.4b}$$

6.2 Calibration

As seen in Figure 2.13, Magneto gives the following matrix values:

$$\mathbf{A}^{-1} = \begin{bmatrix} 0.971938 & -0.000028 & -0.002435 \\ -0.000028 & 0.959990 & -0.005411 \\ -0.002435 & -0.005411 & 0.975702 \end{bmatrix} \tag{6.5}$$

and

$$\mathbf{b} = \begin{bmatrix} 0.020051 \\ 0.010337 \\ 0.003964 \end{bmatrix} \tag{6.6}$$

Content of both matrices is imported to the (2.14) in `BoschIMU_script.m` script, where calibrated and uncalibrated data is plotted and compared. Figures 6.3, 6.4 and 6.5, show accelerometer measurements when the IMU is positioned with the A_z pointing towards the sky. The same procedure is performed for the other two axes. Axis A_x and A_y are oriented towards the sky and data is logged for 10 seconds each. The results shown in Tables 6.1, 6.2 and 6.3 indicate that the calibration method is successful, due to the lesser presence of the bias term in the measurements. As mentioned earlier, some axis orientations are better calibrated than the others. This is apparent when the specific sensitivity axis is pointed towards the sky and is supposed to measure an acceleration value of 1000 mG. In these positions calibration method removes over 50 % of the bias value, as seen in Table 6.2. In Table 6.1 and 6.3, calibration removes more than 80 % of the bias value. The method is not always successful to the same extent, as seen for A_y axis in Table 6.2. In these instances, the calibrated A_y measurement has 35 % less bias than the raw data. For A_z and A_x axis in Figure 6.3, the calibration method removed 17 % and 5 % of the bias term, respectively. Small removal values, like the 5 %, can be considered as a negligible improvement over the uncalibrated measurements.

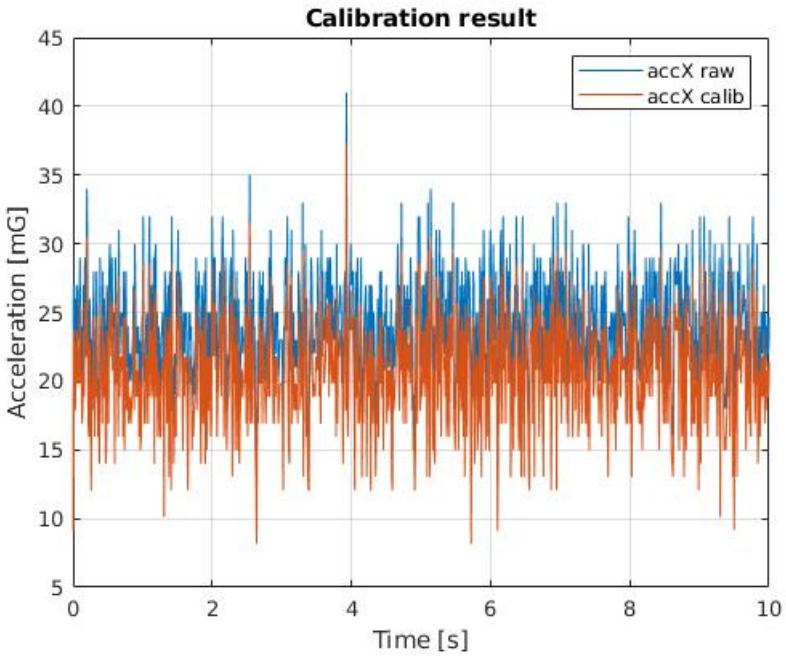

Figure 6.3: Calibration results for A_x.

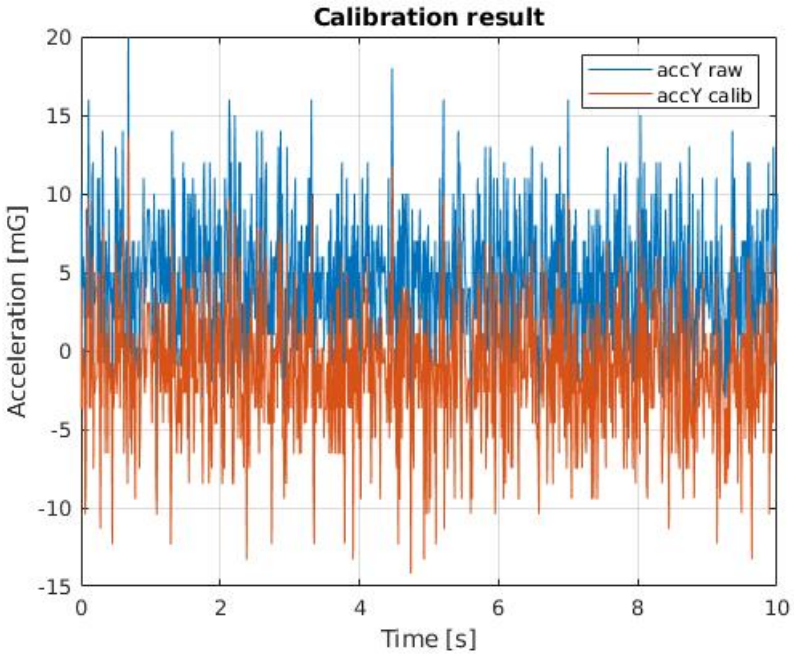

Figure 6.4: Calibration results for A_y.

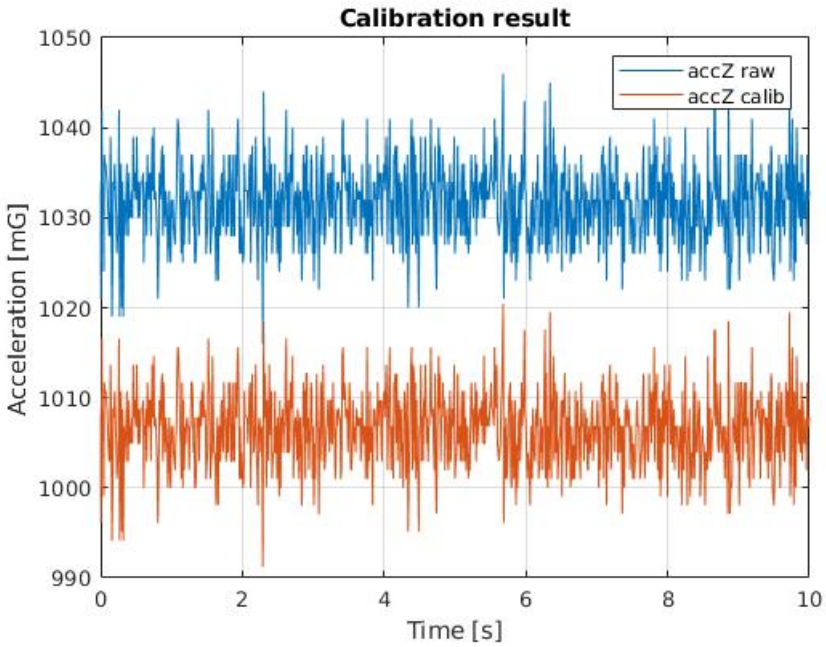

Figure 6.5: Calibration results for A_z.

Table 6.1: Comparison of raw and calibrated measurements for A_z

Axis	True	Calibrated		Raw		Unit
		Measured	Bias	Measured	Bias	
A_x	0	20.64	20.64	23.84	23.84	mG
A_y	0	-1.29	-1.29	4.47	4.47	mG
A_z	1000	1006.50	6.05	1031.70	31.70	mG

Table 6.2: Comparison of raw and calibrated measurements for A_x

Axis	True	Calibrated		Raw		Unit
		Measured	Bias	Measured	Bias	
A_x	1000	1024.70	24.70	1054.30	54.30	mG
A_y	0	2.04	2.04	2.21	2.21	mG
A_z	0	5.01	5.01	7.78	7.78	mG

Table 6.3: Comparison of raw and calibrated measurements for A_y

Axis	True	Calibrated		Raw		Unit
		Measured	Bias	Measured	Bias	
A_x	0	23.56	23.56	24.88	24.88	mG
A_y	1000	1006.90	6.90	1050.40	50.40	mG
A_z	0	32.47	32.47	38.86	38.86	mG

6.3 Low-pass filter

Equation (4.16) was implemented in `lowPassfilter_realTime.py` script, described in Appendix A.1.7. The results can be seen in Figure 6.6.

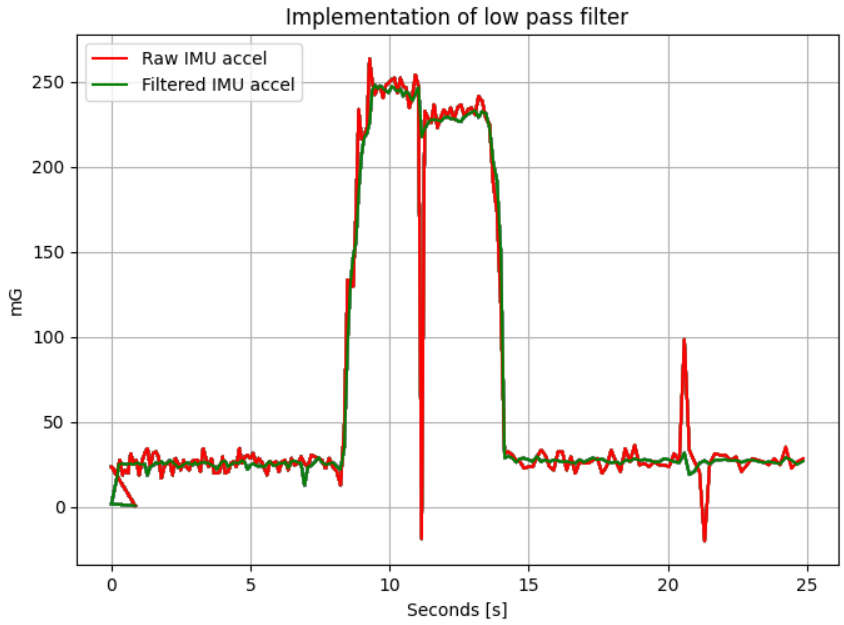

Figure 6.6: Implementation of the real time low-pass filtering.

The low-pass filter algorithm performs real time filtering of the raw accelerometer data. As seen in the Figure 6.6, filtered data are much less noisy than the raw accelerometer data. The low-pass filter removes the white noise and smooths the accelerometer output. From the plot in Figure 6.6, it can be seen that the filter is efficient in removing white noise and the disturbances coming from the environment. During the time interval 8 to 25 seconds, the IMU was oriented at different orientations, in addition to being exposed to vibrations. The vibrations occurring at 11 and 21 seconds are suppressed by the filter, ensuring that the accelerometer output is stable and not susceptible to vibrations. The filter is designed with a sampling frequency of 1000 Hz. This sampling frequency is more than sufficient for the task of sampling the accelerometer output. The acceptable sampling frequency, in conjunction with the successful discretization, results in a stable digital filter capable of filtering out disturbances. As seen in Figure 6.6, the small delay is present in the filter output, which is typical of the first order process.

6.4 Butterworth filter

The Butterworth coefficients, seen in (4.27), are implemented in `butter_lowPass_comp.py` script, described in Appendix A.1.3. The script performed filtering using both Butterworth and low-pass filter on the a_x measurements. The results can be seen in Figure 6.7.

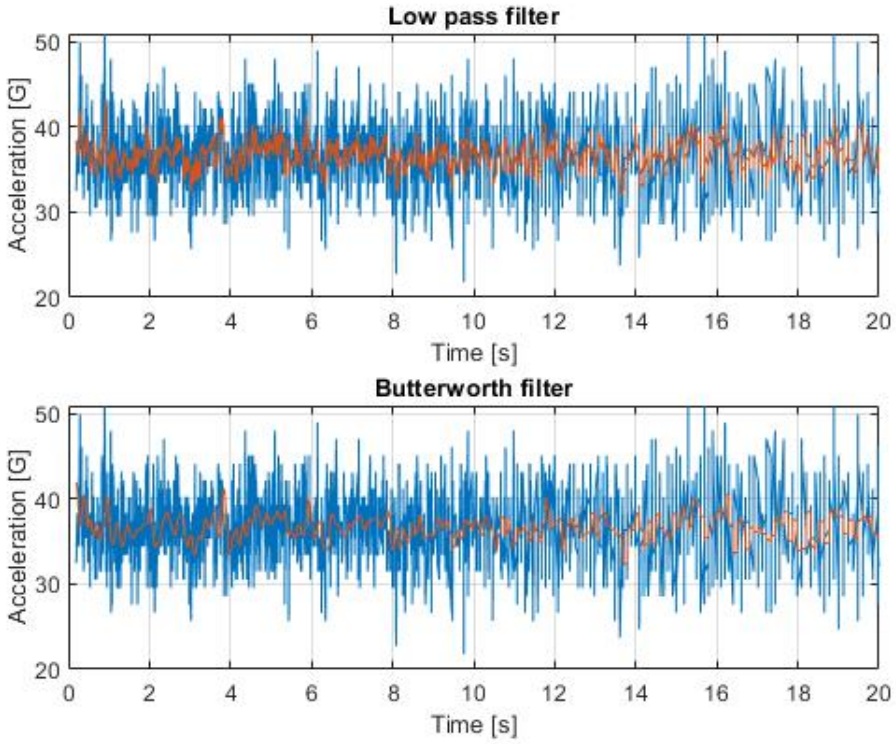

Figure 6.7: Comparison of the Butterworth and low-pass filter.

The IMU was recording acceleration measurements for 20 seconds, while being undisturbed on the table-top. The data set was filtered in real time and recorded to a .csv file that was processed in the `BoschIMU_script.m` script. The results show that the 4th order Butterworth filter performs better filtering than the first order low pass-filter. In the two plots seen in Figure 6.7, the cutoff frequency for both filters was chosen to be at 50 Hz. After calculating standard deviation for the filtered data, the results show that the Butterworth filter performs better filtering, as seen in Table 6.4. Lower standard deviation values, as well as plots in Figure 6.7, indicate that the Butterworth filter is the better filtration method when compared to the 1st order low-pass filter. This is expected since 4th order filters offer steeper attenuation rates.

Table 6.4: Standard deviation values for the two filters

Name	Standard deviation.
Butterworth filter	1.29
Low-pass filter	1.44

6.5 Complementary filter

Testing of the complimentary filter implementation was performed using Reach Alpha 5 manipulator [62], as seen in Figure 6.8. Reach Alpha 5 manipulator is assumed to be the crane, tasked of picking up an object and lifting it in the air. The IMU was attached to the third link, on the axis of orientation of the third rotary joint, as seen in Figure 6.9.

Figure 6.8: Reach Alpha 5 underwater manipulator.

Figure 6.9: Positioning of the IMU on the Reach Alpha 5 manipulator.

The Reach Alpha 5 manipulator performed sinusoidal movement, which was captured by the accelerometer and gyroscope. The sensors measured the acceleration and angular velocity, that were then utilized to produce reliable angle estimates. The plot in Figure 6.10 shows the pitch angle estimation. The three graphs plotted represent angle estimations using accelerometer, gyroscope and complimentary filter. By observing the figure, it is apparent that the complimentary filter provides the best angle estimates. The angle estimates are both correct and less noisy than the ones provided by the accelerometer and gyroscope. The accelerometer is providing correct angle estimations, but the estimates are noisy and are susceptible to vibrations. Due to bias offset present in the gyroscope measurements, the integration algorithm causes angle estimation to drift noticeably off the true value, thus rendering this method completely incorrect. The integration algorithm, described in (3.7), accumulates the error from the gyroscope measurements and produces an incorrect angle estimates.

As mentioned in Section 4.1, the equations used to calculate the angle assume that the IMU is positioned in the center of the rotational axis. If the IMU is not positioned in the center of the rotational axis, the angle estimates provided by the complimentary filter will be incorrect. In Figure 6.11, it can be seen that the angle estimates are unstable. The complimentary filter produces incorrect estimations due to the linear accelerations induced by the movement of the manipulator arm. Since the IMU is no longer positioned in the center of the rotational axis, the accelerometer is no longer capable of measuring only the gravitational acceleration, thus the angle estimates are no longer precise. Plots describing position and velocity of the Reach Alpha 5 manipulator, when performing maneuvers, can be seen in Figures 6.12 and 6.13.

The plots in Figure 6.13 indicates that the manipulator arm is experiencing slight vibrations when performing the movements. This is most noticeable in the situations when the joints are changing direction of the movement. These vibrations are also one of the reasons for the failure of the complimentary filter, as seen in Figure 6.11. The vibrations in combination with the incorrect placement of the IMU leads to the incorrect angle estimations and an instability of the sensor fusion algorithm.

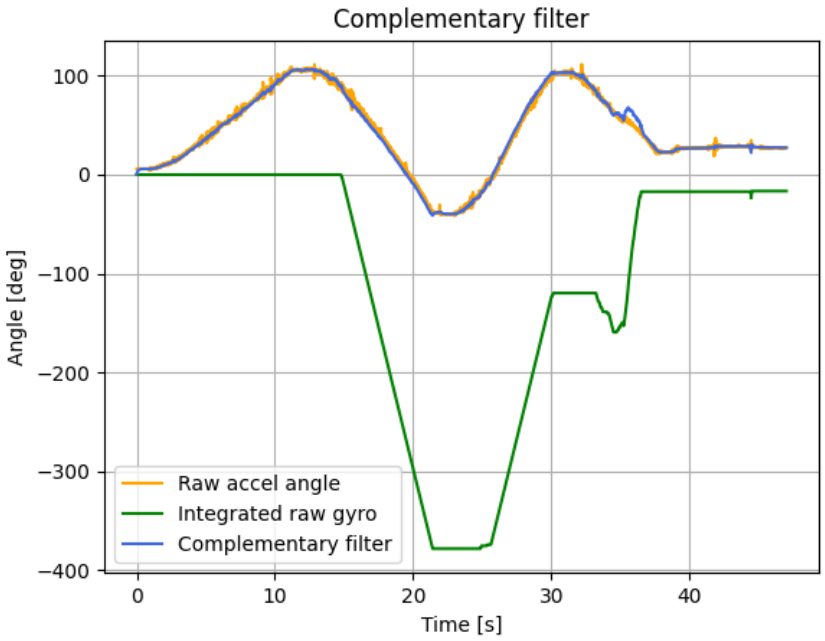

Figure 6.10: Complimentary filter angle estimates.

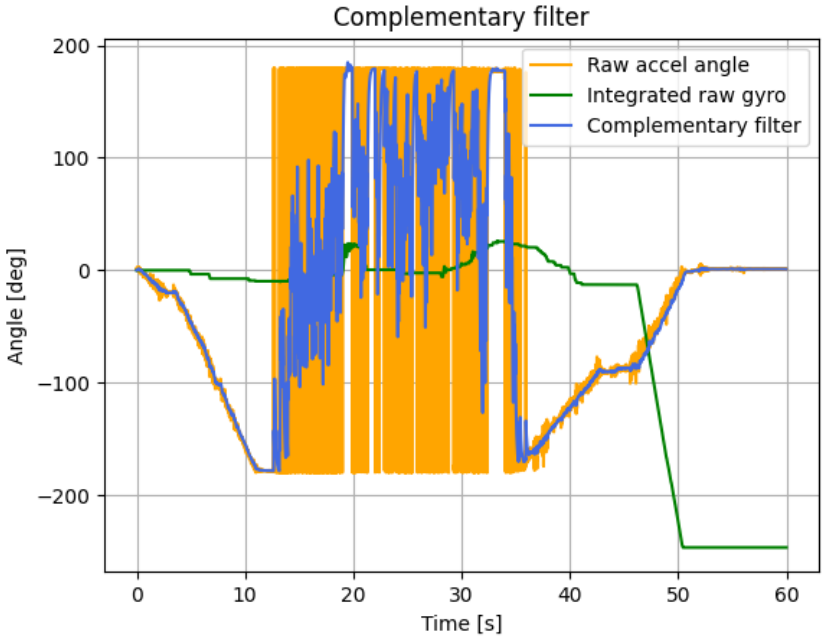

Figure 6.11: Failure of the complimentary filter angle estimates.

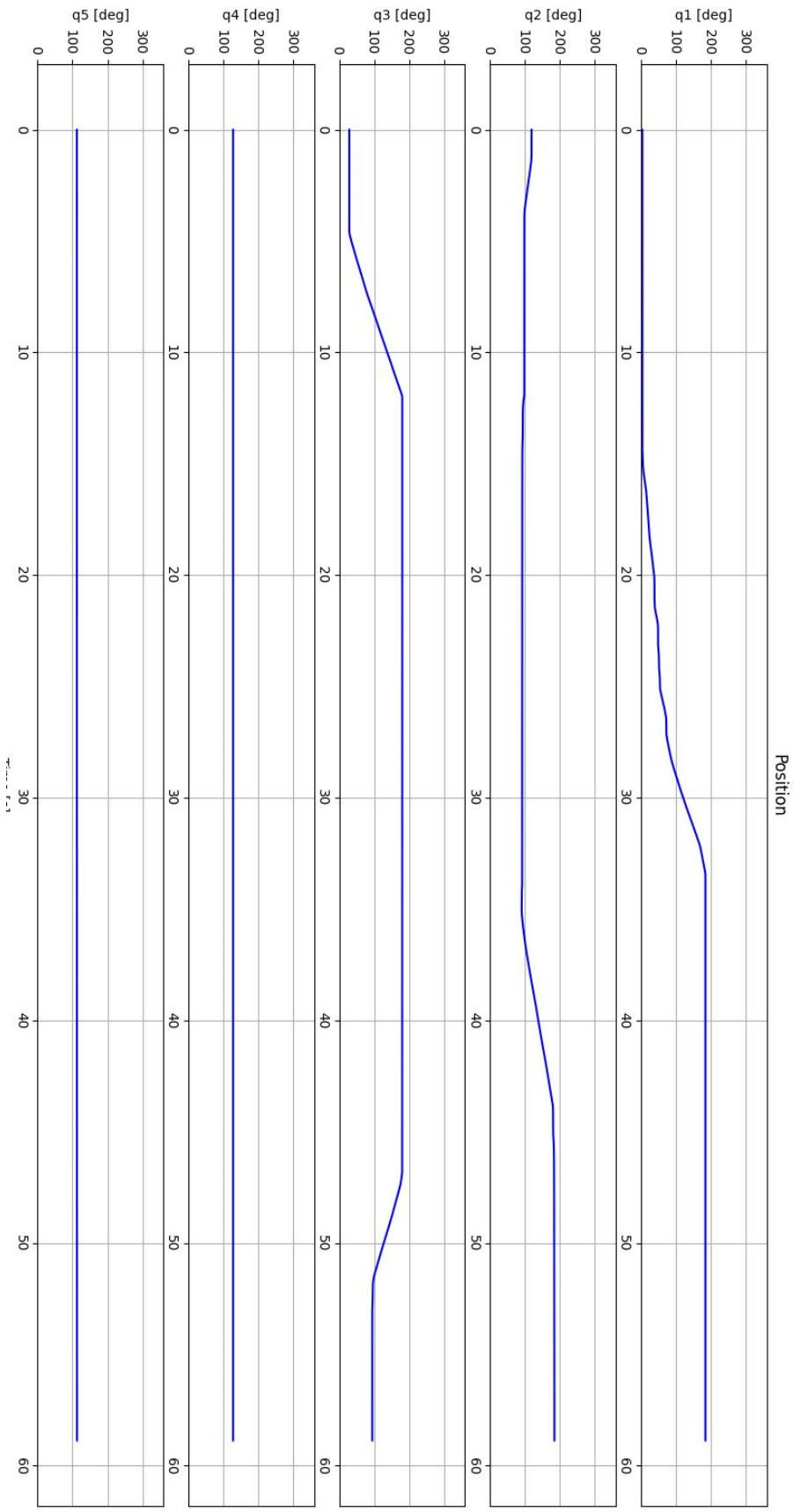

Figure 6.12: Position of the Reach Alpha 5 joints.

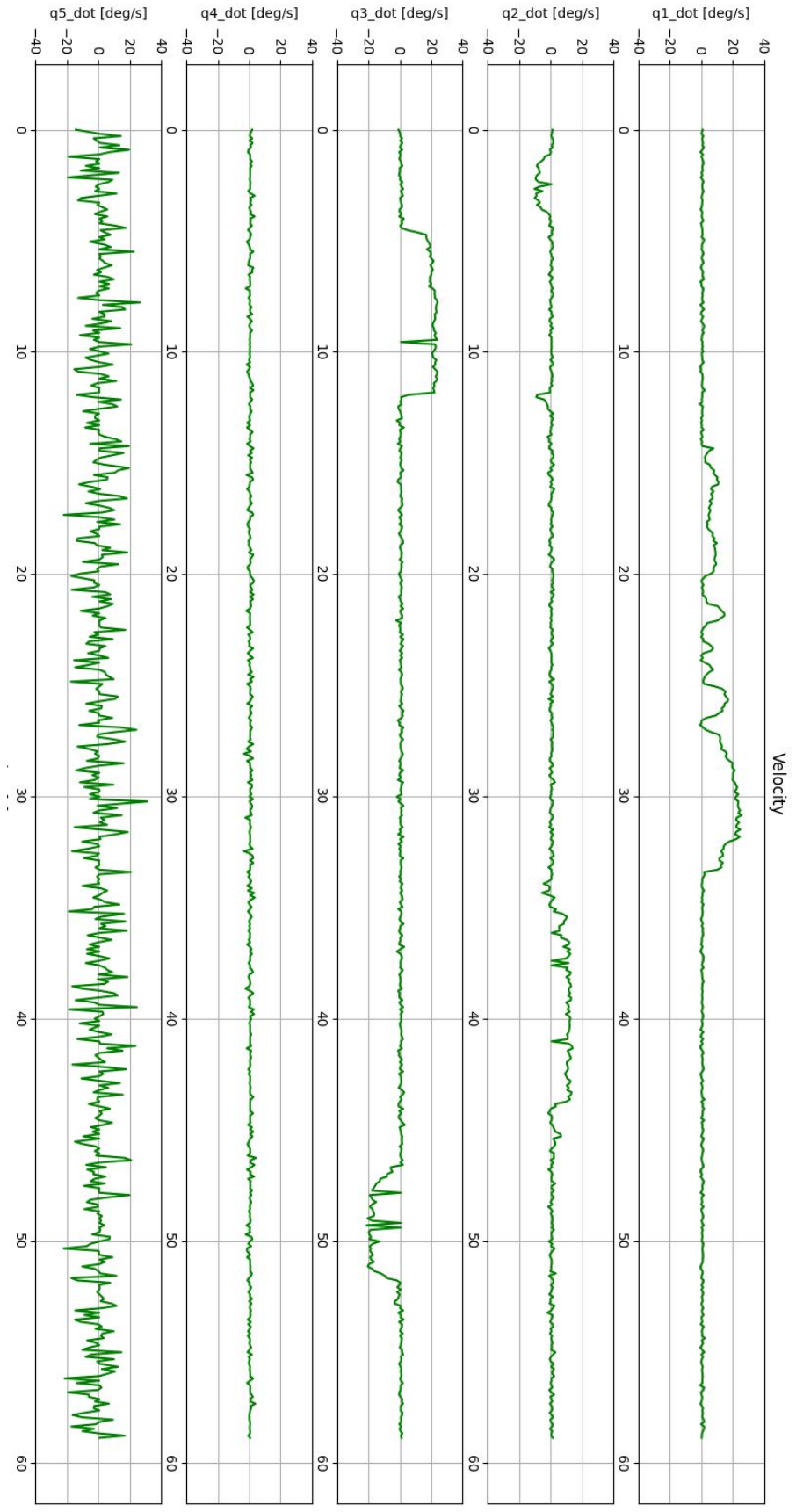

Figure 6.13: Velocity of the Reach Alpha 5 joints.

6.6 Integration of the accelerometer measurements

Theoretically, the position of the object can be determined by the double integration of the accelerometer measurements, as shown in (6.7). In practice, this is not achievable due to the problems introduced by the bias and the white noise, present in the measurements. Integrated accelerometer readings can be seen in Figure 6.14. In this experiment, the IMU was lifted 30 cm into the air with the A_z axis pointing upwards towards the sky. When the IMU reached the top position, it was rapidly lowered to the original starting position. Acceleration measurements were recorded and integrated in order to examine if the integration will produce the correct displacement of 30 cm in both directions. As expected, integrated accelerometer readings provide the position estimate that is not correct. Due to the accumulation of the bias error, the position estimate indicates that the IMU has traveled a distance of 5 meters.

$$x(t) = \iint a(t)dt \tag{6.7}$$

The incorrect position estimate is the product of the error term being rapidly accumulated through the double integration. Calibration methods and low-pass filters will not remove the entire bias offset, thus it will not be possible to perform successful integration of the accelerometer measurements.

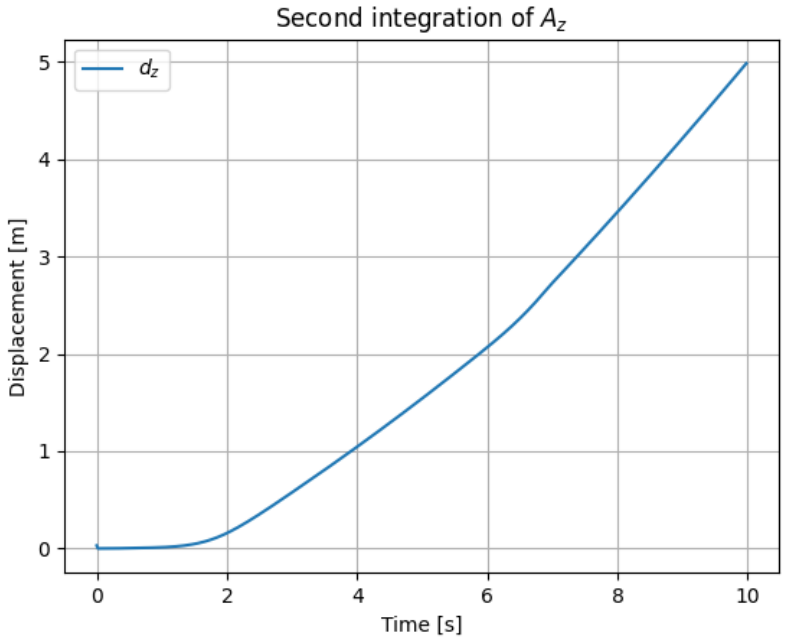

Figure 6.14: Double integration of the accelerometer measurements.

6.7 Mathematical model

The equations in (5.11), together with the RK method, are implemented in `pendulum_calc.py` script, described in Appendix A.1.2. The calculation results can be seen in Figure 6.15.

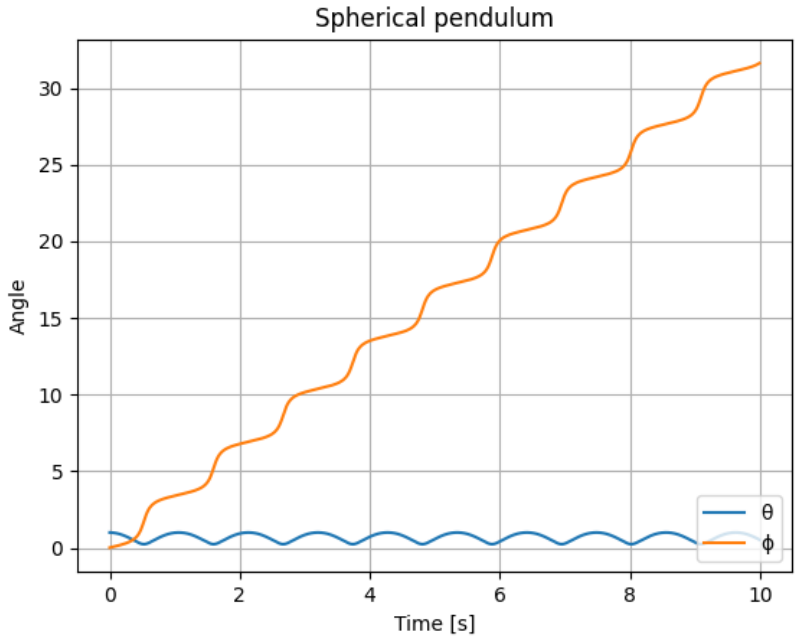

Figure 6.15

The simulation is performed with the initial conditions of $\dot{\phi} = 1$ and $\theta = 1$. The plot in Figure 6.15 shows movement of the spherical pendulum when no damping is present in the system. The starting conditions for this calculation, have to be chosen in a way that θ is never zero, due to the fact that this mathematical model is singular when $\theta = 0$. The second equation in (5.12) is singular when $\theta = 0$, because the sine term becomes 0 and the entire model goes to infinity, thus crushing the Python scrip.

Figure 6.15 shows θ and ϕ angle of the spherical pendulum when no damping is present in the system. Since there is no damping, the θ angle never goes to zero. This is because, as θ gets closer to zero, the angular velocity $\dot{\phi}$ increases, and it propels the mass out, thus increasing the θ angle and not allowing it to become zero or negative. This can be seen from the plot in Figure 6.15, as every time the angle θ gets closer to zero, the angular velocity $\dot{\phi}$ experiences a rapid increase in the angular velocity, thus throwing the mass out.

Chapter 7

Discussion

The main hardware component used in this thesis was the Bosch XDK110 which proved to be a reliable tool for obtaining inertial measurements. The physical dimensions of the XDK and the fact that it incorporates multiple sensors, allowed for much easier handling and greater precision of the measurements. The fact that the XDK110 is an industrial IMU, created significant problems, since industrial IMUs are not always user-friendly and can be quite difficult to work with. This became prominent in the early stages of the research. Unlike with other, cheaper IMUs, the producer of the XDK did not provide information on how to capture the data transmitted by the XDK sensors via USB port. Availability of multiple programming languages, like Mita and C, allowed for easier programming of the XDK. Simplicity of the Mita script and availability of the online sources simplified the process of configuring the sensors and exploiting their full potential.

Due to the manufacturing imperfections, the accelerometer measurements included a noticeable bias offset, which had to be removed in order to gain the greatest possible precision from the accelerometer. This was performed using a simple calibration procedure, which was to the greatest degree successful. One possible improvement to the calibration coefficients could be to simply record as many orientations as possible, specially in the regions lacking these measurements, like the third quadrant of the YZ plane in Figure 2.17. More advanced calibration methods could have been used, in order to provide greater bias removal. This idea was rejected due to the nature of the job that is to be performed at the ship deck. The majority of crane operations do not require extreme precision, therefore it was assumed that the greater precision of the accelerometer measurements will not be crucial.

Bosch developer page offers information on the inside calibration procedure for the XDK. This procedure is performed in the XDK Workbench using C script with the calibration values provided by the manufacturer. This script performs better calibration than the calibration procedure described in the Section 2.9. Unfortunately, C code was too complicated, and the time limitations did not allow for further research. Gyroscope measurements were precise without the major error terms. This was apparent after the gyroscope data was recorded to a file for five hours. The measurements captured by the G_x and G_y axis did possess small amount of error, while the G_z axis measurements were perfect without any drift or bias. Since the XDK also provides another set of accelerometer and gyroscope sensors, it would be beneficial to conduct more thorough experiments in order to determine more precisely which of the sensors are better to use.

Regardless of their cost and quality, all IMUs will suffer from bias offset and white noise. For this reason, it was decided to perform frequency analysis of the accelerometer and gyroscope measurements. Looking at the results, Allan analysis showed that both the accelerometer and the gyroscope suffer from white noise. Analysis of the gyroscope readings provided the information regarding angle random walk and bias instability, both of which proved that the gyroscope contains minimal bias and drift values. Frequency domain analysis provided information regarding the frequency spectrum of the accelerometer and gyroscope measurements. This demonstrated that the IMU is extremely sensitive to outside influences. The plot in Figure 2.9 demonstrates this observation, as it shows the 11 Hz frequency component which was identified as the CPU cooling fan. Performing the frequency analysis using the Fourier transform is extremely important since it allows for identification of the various frequency components present within the signal. This knowledge can later be used when designing filters, e.g. notch filter, tasked with eliminating unwanted frequency components.

High frequency white noise present in the raw accelerometer measurements would potentially create problems for the control structure, therefore the raw measurements have to be filtered using some type of low-pass filter. The simple first order low-pass filter performed successful filtering of the raw data. The standard deviation value calculated from the filtered data indicates that the performance of the first order low-pass filter is slightly worse than that of a Butterworth filter. In the report, due to the cascaded implementation procedure, it was decided to used 4th order Butterworth filter. Implementing the Butterworth filter using the difference equation led to the Python script crashing due to the rapid instability in the filter output, caused by the quantization error. For this reason, cascaded Biquad implementation had to be implemented. The filter's cutoff frequency must obey Nyquist-Shannon sampling criterion shown in (4.17). Since access to the crane was not permitted, the frequency of the crane had to be assumed, therefore it is possible that the cutoff frequency for the filter is incorrect. An incorrect cutoff frequency could result in inclusion of unwanted frequencies in the output signal, thus the cutoff frequency could be chosen in a way that the filter provides the best possible filtration of the raw data.

The Euler angles were chosen as the angle representation for the crane links and the suspended load. This angle representation is known to suffer from singularity problems. Problems arising from singularity can be resolved by using different type of angle representation like the quaternions. The quaternions would resolve singularity problems and would be much more precise than the Euler angles. A four dimensional representation of the rotation angle, performed by the quaternions, would also introduce other problems like the increased computational time and the double mapping of the angles. Rotation matrices were also one of the possible choices of the representation. This approach would introduce more complexity since rotation matrices have 9 elements, and they have to obey $SO(3)$ rules. For the angle representation of the crane links and load, it was decided that the use of Euler angles would be sufficient. It was assumed that the orientation of the crane links will never reach singularity positions, and if the crane link is to reach e.g. singularity orientation of 90 degrees, then such orientation could only be reached due to some serious malfunction.

The accelerometer and gyroscope have complementary errors, i.e., both are erroneous but in different bands of the frequency spectrum and hence can be fused to get much more accurate measurements than one could ever get with either of the sensors separately. The computationally simplest, and the most intuitive method was the complimentary filter. It performed fusion of the accelerometer and gyroscope, which produced a correct angle estimate. The complimen-

tary filter code used to process prerecorded data worked as intended and was able to confirm that the method provides a correct estimate. The code used for real-time filtering was much more susceptible to vibrations and other outside disturbances. High sensitivity of the accelerometer and gyroscope created problems for the real-time complimentary filter, often causing instability in the output measurements. For this reason, low-pass and Butterworth filters were used to filter out accelerometer and gyroscope measurements prior to passing them to the complimentary filter. This approach had limited success due to the fact that filtering the data prior to passing it to the complimentary filter will create lag in the system.

In general, it is difficult to achieve pose estimation using only IMUs. Considering the dynamic environment of the ocean, in addition to the working principle of the IMU, it can be concluded that the IMU cannot provide reliable pose estimations. The IMU is capable of providing orientation estimates with a certain degree of success, but it is not capable of providing position estimations of the entire crane. Three IMUs have to be used in order to provide estimation of all three degrees of freedom. Using trigonometry, the IMU is capable of providing position estimation of the arbitrary point on the crane links, but it cannot provide the position of the end-effector due to the fact that the IMU cannot measure linear displacement caused by the prismatic joint. The accelerometer could theoretically provide absolute position estimations using double integration of the acceleration measurements captured during the linear translation. In practice, this is almost impossible to achieve due to the various error terms in the accelerometer measurements. Therefore, other sensors have to be used in addition to the IMUs.

The position of the crane end-effector could be determined by using different sensors like encoders, digital switches, photo-electric switches etc. [63]. Optical sensors and encoders would provide information regarding displacement caused by the crane's prismatic joint, thus directly describing the position of the end-effector [64]. Given the known joint angles and displacement of the prismatic joint, provided by other sensors, the forward kinematics can be calculated, thus providing position and velocity of the end-effector. Cameras and other types of vision systems could also be extremely effective in providing the position of the crane end-effector and the load attached to it. A camera mounted on the crane captures the images of the ship deck, which are used to directly calculate the crane's motion. The crane's position relative to the desired position is estimated based on the visual information encoded in the spatio-temporal derivatives of the image function [65].

The mathematical model of the crane load describes the position of the load relative to the crane's end-effector. Unfortunately, it would not be possible to obtain the load position due to the singularity. As seen in (5.12), the second differential equation $\ddot{\phi}$ goes to infinity when θ is equal to 0, thus, the equation fails to deliver a unique solution. The Python script manages to perform calculation and simulation of the load motion, despite the fact that the script should crash in instances when θ is close to 0. This is most probably due to the Python time step being too large, thus the θ angle never reaches singular values.

Chapter 8

Conclusion and future work

This thesis concludes that the Inertial Measurement Unit cannot provide reliable pose estimation of the crane links and the suspended load. The research showed that the IMU is capable of providing orientation estimations. Using the obtained orientations, dimensions of the crane and the trigonometrical analysis, it is possible to obtain the position of the desired point on the crane body. Obtaining the position of the end-effector is not possible because the IMU is not capable of estimating linear displacement performed by the prismatic joint. Furthermore, the IMU is extremely sensitive to the disturbances caused by the environment, thus any angle and position estimations will be heavily dependent on the amount of disturbances acting on the IMU. For this reason, it is an imperative to protect the IMU from external influences. Potential future plans for the use of the IMU have to consider if it will be possible to protect the IMU from the outside factors like vibration, electromagnetic fields, changing temperature, weather conditions etc.

Since the IMU cannot provide estimation of the end-effector position, other sensors capable of performing the task have to be used. The additional information, required for the position and orientation estimation, would be provided by GNSS and a computer vision system. Therefore, it would be beneficial to use these sensors in combination with the IMU. Due to the time constraints, it was not possible to obtain a full mathematical model of the crane and the load. The mathematical model describing the position of the load should be revised. The inferior mathematical model is singular when $\theta = 0$, therefore it should be examined whether different angle representation will produce a better model. Sensor fusion algorithms are essential for the successful pose estimation, regardless of which sensor combination is used.

References

[1] Aslak Johannes Strand. "Control and Sensor Systems for Offshore and Aquaculture Cranes". MA thesis. Norway: Norwegian Univesity of Science and Technology, 2021.

[2] Liyana Ramli, Zaharuddin Mohamed, Auwalu M. Abdullahi, Hazriq Izzuan Jaafar, and Izzuddin M. Lazim. "Control strategies for crane systems: A comprehensive review". In: *Mechanical Systems and Signal Processing* 95 (2017), pp. 1–23.

[3] Nenad Zrnić, Kent Hoffmann, and Srđan Bošnjak. "A note on the history of handling in ports: from ancient to medieval cranes". In: *12th IFToMM World Congress, Besançon (France), June18–21*. 2007.

[4] Office of the Chief of Naval Operations. Naval History Division. *USS Crane Ships No. 1 (AB-1)*. 2009. URL: `https://www.ibiblio.org/hyperwar/USN/ships/dafs/AB/ab1.html` (visited on 02/15/2022).

[5] Shih-Chung Kang and Eduardo Miranda. "Physics Based Model for Simulating the Dynamics of a Tower Crane". In: June 2004.

[6] Dong Liu, Xuzhi Lai, Yawu Wang, and Xiongbo Wan. "Position Control of a Planar Four-Link Underactuated Manipulator". In: *2018 37th Chinese Control Conference (CCC)*. 2018, pp. 929–932.

[7] Yuzhe Qian and Yongchun Fang. "Dynamics analysis of an offshore ship-mounted crane subject to sea wave disturbances". In: *2016 12th World Congress on Intelligent Control and Automation (WCICA)*. IEEE. 2016, pp. 1251–1256.

[8] Ronny Landsverk and Jing Zhou. "Modeling and simulation of an offshore crane". In: *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE. 2018, pp. 713–718.

[9] José Miguel Varela and Carlos Guedes Soares. "Interactive Simulation of Ship Motions in Random Seas based on Real Wave Spectra." In: *GRAPP*. 2011, pp. 235–244.

[10] Florentin Rauscher, Samuel Nann, and Oliver Sawodny. "Motion control of an overhead crane using a wireless hook mounted IMU". In: *2018 Annual American Control Conference (ACC)*. IEEE. 2018, pp. 5677–5682.

[11] Hanspeter Schaub. "Rate-based ship-mounted crane payload pendulation control system". In: *Control Engineering Practice* 16.1 (2008), pp. 132–145.

[12] Norhafizan Ahmad, Raja Ariffin Raja Ghazilla, Nazirah M. Khairi, and Vijayabaskar Kasi. "Reviews on various inertial measurement unit (IMU) sensor applications". In: *International Journal of Signal Processing Systems* 1.2 (2013), pp. 256–262.

[13] Eli Gai. "The century of inertial navigation technology". In: *2000 IEEE Aerospace Conference. Proceedings (Cat. No. 00TH8484)*. Vol. 1. IEEE. 2000, pp. 59–60.

[14] Linkun Deng, Hang Guo, Nikolas Xiros, and Min Yu. "A research on roll angle calculations based on IMU/GPS compass for ships". In: *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE. 2016, pp. 976–980.

[15] Roderick A. Barr and Vladimir Ankudinov. “Ship rolling, its prediction and reduction using roll stabilization”. In: *Marine Technology and SNAME News* 14.01 (1977), pp. 19–41.

[16] Yingguang Chu, Filippo Sanfilippo, Vilmar Æsøy, and Houxiang Zhang. “An effective heave compensation and anti-sway control approach for offshore hydraulic crane operations”. In: *2014 IEEE International Conference on Mechatronics and Automation*. IEEE. 2014, pp. 1282–1287.

[17] Ning Xianliang, Zhao Jiawen, and Xu Jianan. “The heave motion estimation for active heave compensation system in offshore crane”. In: *2016 IEEE International Conference on Mechatronics and Automation*. IEEE. 2016, pp. 1327–1332.

[18] Oddvar Gjelstenli. “Anti-Sway Control and Wave Following System for Offshore Lattice Crane”. MA thesis. Norway: Norwegian University of Science and Technology, 2012.

[19] Chris C. Ward and Karl Iagnemma. “A dynamic-model-based wheel slip detector for mobile robots on outdoor terrain”. In: *IEEE Transactions on Robotics* 24.4 (2008), pp. 821–831.

[20] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. “Using inertial sensors for position and orientation estimation”. In: *arXiv preprint arXiv:1704.06053* (2017).

[21] Weixin Yang, Alexandr Bajenov, and Yantao Shen. “Improving low-cost inertial-measurement-unit (IMU)-based motion tracking accuracy for a biomorphic hyper-redundant snake robot”. In: *Robotics and biomimetics* 4.1 (2017), pp. 1–9.

[22] Wang Shenghai, Sun Yuqing, Chen Haiquan, and Du Jialu. “Dynamic modelling and analysis of 3-axis motion compensated offshore cranes”. In: *Ships and Offshore Structures* 13.3 (2018), pp. 265–272.

[23] Thor I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[24] Encyclopædia Britannica. *Magnetometer*. 2022. URL: `https://www.britannica.com/technology/magnetometer`.

[25] *Cross-Domain Development Kit XDK110 Platform for Application Development*. BCDS-XDK110-SENSOR-GUIDE. Bosch Connected Devices and Solutions. 2017, p. 7.

[26] Andreas Grammenos, Cecilia Mascolo, and Jon Crowcroft. “You Are Sensing, but Are You Biased? A User Unaided Sensor Calibration Approach for Mobile Sensing”. In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* (2018).

[27] Jia Qi Zhou. “Simplified Analysis of IMU Sensor Corruptions on Existing Pendulation Control System For Ship-Mounted Crane”. MA thesis. United States: Virginia State University, 2006.

[28] Farid Gulmammadov. “Analysis, modeling and compensation of bias drift in MEMS inertial sensors”. In: *2009 4th International Conference on Recent Advances in Space Technologies*. IEEE. 2009, pp. 591–596.

[29] Anthony Lawrence. *Modern inertial technology: navigation, guidance, and control*. Springer Science & Business Media, 2001.

[30] Bosch Developer Portal. *Mita Accelerometer Sensor*. 2022. URL: `https://developer.bosch.com/products-and-services/sdks/xdk/develop/mita/sensors-mita/accelerometer`.

[31] Bosch Developer Portal. *Mita Gyroscope Sensor*. 2022. URL: `https://developer.bosch.com/products-and-services/sdks/xdk/develop/mita/sensors-mita/gyroscope`.

[32] Jan Machacek and Jiri Drapela. “Control of serial port (RS-232) communication in LabVIEW”. In: *2008 International Conference-Modern Technique and Technologies*. IEEE. 2008, pp. 36–40.
[33] Naser El-Sheimy, Haiying Hou, and Xiaoji Niu. “Analysis and modeling of inertial sensors using Allan variance”. In: *IEEE Transactions on instrumentation and measurement* 57.1 (2007), pp. 140–149.
[34] Barbara F. La Scala and Robert R. Bitmead. “Design of an extended Kalman filter frequency tracker”. In: *IEEE Transactions on Signal Processing* 44.3 (1996), pp. 739–742.
[35] David W. Allan et al. “Time and frequency(time-domain) characterization, estimation, and prediction of precision clocks and oscillators”. In: *IEEE transactions on ultrasonics, ferroelectrics, and frequency control* 34.6 (1987), pp. 647–654.
[36] MathWorks. *MATLAB std - Standard deviation*. 2019. URL: `https://se.mathworks.com/help/matlab/ref/std.html` (visited on 05/02/2022).
[37] Marin Marinov and Zhivo Petrov. “Allan variance analysis on error characters of low-cost MEMS accelerometer MMA8451Q”. In: *International conference of scientific paper AFASES*. 2014, pp. 22–24.
[38] MathWorks. *Inertial Sensor Noise Analysis Using Allan Variance*. 2019. URL: `https://se.mathworks.com/help/fusion/ug/inertial-sensor-noise-analysis-using-allan-variance.html` (visited on 04/25/2022).
[39] Ronald N. Bracewell. “The fourier transform”. In: *Scientific American* 260.6 (1989), pp. 86–95.
[40] Pedro Batista, Carlos Silvestre, Paulo Oliveira, and Bruno Cardeira. “Accelerometer calibration and dynamic bias and gravity estimation: Analysis, design, and experimental evaluation”. In: *IEEE transactions on control systems technology* 19.5 (2010), pp. 1128–1137.
[41] Michael Wrona. *How to Calibrate an Accelerometer*. Youtube. 2021. URL: `https://www.youtube.com/watch?v=-1tmYPE7MAQ&ab_channel=MicWroEngr`.
[42] B HEBB. *A General Calibration Algorithm for 3-Axis Compass/Clinometer Devices*. 2009.
[43] Eric W. Weisstein. “Euler angles”. In: *https://mathworld. wolfram. com/* (2009).
[44] Thor I. Fossen. *Lecture Notes TTK 4190 Guidance, Navigation and Control of Vehicles. Chapter 14 – Inertial Navigation Systems*. 2021. URL: `https://www.fossen.biz/wiley/pdf/Ch14.pdf`.
[45] Evan G. Hemingway and Oliver M. O’Reilly. “Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments”. In: *Multibody System Dynamics* 44.1 (2018), pp. 31–56.
[46] Richard Poley. *Introduction to Control Using Digital Signal Processors*. 2010.
[47] Emiel Por, Maaike van Kooten, and Vanja Sarkovic. “Nyquist–Shannon sampling theorem”. In: *Leiden University* 1 (2019), p. 1.
[48] Irfan Ljevo. *Specialization Project. Aquaculture crane*. Tech. rep. 2021.
[49] Jim Karki. “Active low-pass filter design”. In: *Texas Instruments application report* (2000).
[50] Fredrik Dessen. “Optimizing order to minimize low-pass filter lag”. In: *Circuits, Systems, and Signal Processing* 38.2 (2019), pp. 481–497.
[51] Steven W. Smith et al. *The scientist and engineer’s guide to digital signal processing*. 1997.
[52] Martin Vicanek. “Matched Second Order Digital Filters”. In: (2016).

[53] Julius O. Smith. *Direct form I*. 2007. URL: https://ccrma.stanford.edu/~jos/fp/Direct_Form_I.html.

[54] Niel Robertson. *Design IIR Filters Using Cascaded Biquads*. dspelated.com. 2018. URL: https://www.dsprelated.com/showarticle/1137.php.

[55] Vladimir Kubelka and Michal Reinstein. "Complementary filtering approach to orientation estimation using inertial sensors only". In: *2012 IEEE international conference on robotics and automation*. IEEE. 2012, pp. 599–605.

[56] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. "Kalman filter and its application". In: *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. IEEE. 2015, pp. 74–77.

[57] Jurek Z Sasiadek. "Sensor fusion". In: *Annual Reviews in Control* 26.2 (2002), pp. 203–228.

[58] Jay K Hackett and Mubarak Shah. "Multi-sensor fusion: a perspective". In: *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE. 1990, pp. 1324–1330.

[59] Brian Douglas. *Drone Control and the Complementary Filter*. Youtube. 2018. URL: https://www.youtube.com/watch?v=whSw42XddsU&ab_channel=BrianDouglas.

[60] Sebastien Gros. *Modelling And Simulation*. Lecture notes for the NTNU/ITK course TTK4130. NTNU, 2021.

[61] MathWorks. *Inertial Sensor Noise Analysis Using Allan Variance*. 2021. URL: https://se.mathworks.com/help/nav/ug/inertial-sensor-noise-analysis-using-allan-variance.html.

[62] Blueprint Lab. *Reach Alpha*. 2022. URL: https://blueprintlab.com/products/manipulators/reach-alpha/.

[63] Rick Rice. *Linear translation and measuring motion*. 2020. URL: https://www.controldesign.com/articles/2020/linear-translation-and-measuring-motion/.

[64] Sooyong Lee and Jae-Bok Song. "Robust mobile robot localization using optical flow sensors and encoders". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 1. IEEE. 2004, pp. 1039–1044.

[65] Kok-Meng Lee and Debao Zhou. "A real-time optical sensor for simultaneous measurement of three-DOF motions". In: *IEEE/ASME Transactions on Mechatronics* 9.3 (2004), pp. 499–507.

Appendix A

List of external files

In this Appendix, the files used in the research are listed and presented with the description of their functionality. The files are sorted into folders for easier navigation. The file Appendix.zip contains Python and Matlab code used in this thesis. Additionally, the specialization project report and the IMU measurements are also included. Files marked with (∗) are the files that require the IMU input in order to perform the calculations. Files without the (∗) often process prerecorded data or perform calculations which do not require IMU data as an input.

A.1 Python

A.1.1 Allan Analysis

- AllanAnalysis.py (Performs Allan analysis of the gyro data)
- AllanDevAnalysis.py (Performs Allan analysis of the accelerometer data)

A.1.2 Accelerometer integration

- `*accel_integration.py` (Integrates accelerometer measurements)

A.1.3 Butterworth Filter

- `*butter_lowPass_comp.py` (Comparison of the low-pass filter and Butterworth filter)
- `butterworthFilter_preRec.py` (Butterworth low-pass filtering of the prerecorded data)
- `*butterworthFilter_realTime.py` (Butterworth low-pass filtering of the real time data)
- `filterDesign.py` (Design of the digital Butterworth filter)

A.1.4 Complimentary filter

- `*comFilter_butter_realTime.py` (Filtration of the real time complimentary filter using Butterworth)
- `complementaryFilter_preRec.py` (Complimentary filter for the prerecorded data)

- `*complementaryFilter_realTime.py` (Complimentary filter for the real time filtering)

A.1.5 Calibration

- `recor_uncalib_data.py` (Records uncalibrated accelerometer measurements)
- `plot_calib_data.py` (Plots calibration results)

A.1.6 Obtaining angles

- `*getAngleAccel_Pitch.py` (Estimates pitch angle using accelerometer)
- `*getAngleAccel_Roll.py` (Estimates roll angle using accelerometer)
- `*getAngleGyro.py` (Estimates angle using filtered gyro data)

A.1.7 Low-pass filter

- `*lowPassfilter_realTime.py` (Performs real time low-pass filtering)

A.1.8 Pendulum simulation/Mathematical model

- `pendulum_calc.py` (Performs calculation of the pendulum equations)
- `pendulum_sim.py` (Performs simulation of the spherical pendulum)

A.1.9 Noise density

- `noise_density.py` (Calculates noise density of the prerecorded dataset)

A.1.10 General

- `*functions.py` (Script containing SerialPort class used to extract IMU data from the USB port)
- `bent_log_ra5_reader.py` (Plots position and velocity of the Reach Alpha 5 manipulator)

A.2 Matlab

A.2.1 General script

- `*BoschIMU_script.m` (General Matlab script with various subsections. Used to extract IMU data from the USB, plotting, calculations, etc.)
- `biquad_sinth.m` (Calculates Biquad coefficients)

A.3 Measurement file

Contains various IMU measurements written to the text and csv file.

A.4 Reference

Specialization project report (PDF).

Irfan Ljevo

Pose Estimation of Aquaculture Crane Using IMUs

NTNU
Norwegian University of
Science and Technology