

Universität der Bundeswehr
Professur für Satellitennavigation
Institut für Raumfahrttechnik und Weltraumnutzung
Univ.-Prof. Mag. Dr. habil. Thomas Pany

Low-cost Inertial Measurement Unit (IMU) calibration with Shallow Neural Networks

Ian Mambea Solomon

Master Thesis

Master's Course in Earth Oriented Space Science and Technology


Supervisors: 1. Univ.-Prof. Mag. Dr. habil. Thomas Pany
Universität der Bundeswehr, München
2. Dipl.-Ing. Mohamed Bochkati
Universität der Bundeswehr, München
3. Christian Lichtenberger M.Sc.
Universität der Bundeswehr, München
4. Univ.-Prof. Dr.phil.nat. Urs Hugentobler
Technische Universität München, München

April, 2024

Statement of Authorship

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

Munich, th 29 April, 2024 (date)

Author  (signature)

Acknowledgements

I would like to express my heartfelt gratitude to the Almighty Father in Heaven, whose guidance and blessings have been the cornerstone of my journey. Your grace has illuminated my path and filled my endeavors with purpose.

I extend my sincere appreciation to the German Academic Exchange Service (DAAD) ST 32, for their invaluable support through the scholarship, which made my academic pursuit possible. The opportunity provided by DAAD, under Leadership for Africa, has been instrumental in shaping my scholarly endeavors and fostering cultural exchange.

I am indebted to the esteemed members of the Universität der Bundeswehr from the Institute of Space Technology & Space Applications for entrusting me with such an intriguing topic and offering their expertise throughout my research. Their mentorship and encouragement have significantly enriched my academic experience.

Lastly, to my beloved family, your unwavering love, encouragement, and understanding have been my source of strength and motivation.

Abstract

Inertial Measurement Units (IMUs) have become ubiquitous in various applications, ranging from navigation systems to motion tracking. However, low-cost IMUs often suffer from significant errors, including bias, scale factor, and drift. This thesis presents a comprehensive study on the calibration of low-cost micro-electro-mechanical system (MEMS) IMUs utilizing a shallow neural network approach within the framework of strapdown technique.

The research begins with an exploration of different error sources affecting low-cost IMUs and compares traditional calibration methods with neural network-based approaches. Simulation signals, generated using MATLAB, are employed for calibration purposes, utilizing Long Short-Term Memory (LSTM) and Feedforward neural networks. Both static and dynamic signals are considered to simulate real-world scenarios accurately.

The primary focus of the study lies in addressing gyroscope signal errors. Real static step signals are collected using a turntable at Universität der Bundeswehr in Munich, and strapdown technique is employed to compensate for constant bias, scale factor, non-orthogonality, misalignment and cross-coupling errors. Despite encountering minor drifts attributed to angle random walk (ARW), the strapdown technique proves to be successful in producing accurate gyroangles from gyro rates.

This thesis contributes to the advancement of IMU calibration methodologies by demonstrating the effectiveness of shallow neural networks in mitigating errors associated with low-cost IMUs. The findings offer insights into improving the accuracy and reliability of IMU measurements, thus enhancing their applicability in various fields, including navigation, robotics, and virtual reality.

Zusammenfassung

Inertiale Messeinheiten (IMUs) sind in verschiedenen Anwendungen allgegenwärtig, von Navigationssystemen bis hin zur Bewegungsverfolgung. Kostengünstige IMUs leiden jedoch oft unter erheblichen Fehlern, einschließlich Biase, Skalierungsfaktor und Drift. In dieser Arbeit wird eine umfassende Studie über die Kalibrierung kostengünstiger mikroelektromechanischer Systeme (MEMS) mit Hilfe eines flachen neuronalen Netzwerks im Rahmen der Strapdown-Technik vorgestellt.

Die Forschung beginnt mit einer Untersuchung verschiedener Fehlerquellen, die sich auf kostengünstige IMUs auswirken, und vergleicht traditionelle Kalibrierungsmethoden mit auf neuronalen Netzen basierenden Ansätzen. Für die Kalibrierung werden mit MATLAB generierte Simulationssignale verwendet, wobei Long Short-Term Memory (LSTM) und neuronale Feedforward-Netzwerke zum Einsatz kommen. Es werden sowohl statische als auch dynamische Signale berücksichtigt, um reale Szenarien genau zu simulieren.

Das Hauptaugenmerk der Studie liegt auf der Behebung von Fehlern im Gyroskop-Signal. Reale statische Winkelpositionen werden mit einem Drehtisch an der Universität der Bundeswehr in München erfasst, und die Strapdown-Technik wird eingesetzt, um konstante Biase, Skalierungsfaktor, Nicht-Orthogonalität, Fehlausrichtung und Kreuzkopplungsfehler zu kompensieren. Trotz geringfügiger Drifts, die auf einen zufälligen stochastischen Prozess, z.B., den Angle Random Walk (ARW) zurückzuführen sind, erweist sich die Strapdown-Technik als erfolgreich bei der Erzeugung genauer Kreiselwinkel aus den entsprechenden Drehraten.

Diese Arbeit trägt zur Weiterentwicklung der IMU-Kalibrierungsmethoden bei, indem sie die Wirksamkeit von flachen neuronalen Netzen bei der Abschwächung

von Fehlern im Zusammenhang mit kostengünstigen IMUs demonstriert. Die Ergebnisse bieten Einblicke in die Verbesserung der Genauigkeit und Zuverlässigkeit von IMU-Messungen, wodurch ihre Anwendbarkeit in verschiedenen Bereichen, einschließlich Navigation, Robotik und virtuelle Realität, erhöht wird.

Table of Contents

Statement of Authorship	i
Acknowledgements	ii
Abstract	iii
Zusammenfassung	iv
List of Figures	viii
List of Tables	xi
1. Introduction	1
2. Inertial Sensors	5
2.1 Accelerometer	5
2.2 Gyroscope	8
2.3 Error Budget	11
2.3.1 Deterministic Errors	12
2.3.2 Stochastic Errors	12
3. Classical Sensor Calibration	17
3.1 Six-Position Static Test (SPST)	17
3.2 Angle Rate Test	18
3.3 Least Squares Method	19
4. Artificial Neural Networks	22
4.1 Supervised And Unsupervised Learning	22
4.2 Data Handling	24
4.3 Feedforward NN	25
4.4 Long-Short Term Memory Network	27
4.4.1 Structure	28
4.4.2 Back Propagation	31

4.4.3	Loss Function	33
5.	Methodology	34
5.1	Hardware And Software.....	35
5.1.1	Turntable	35
5.1.2	Computer Specifications	36
5.1.3	Software	36
5.2	SPST	37
5.3	Gyrorate To Gyrorate AND Accelerometer To Accelerometer Calibration With LSTM and FNN	38
5.3.1	Static Test With LSTM.....	38
5.3.2	Dynamic Test With LSTM	44
5.3.3	Dynamic Test With Feedforward NN	51
5.3.4	Static Step Simulation Test Considering Constant Angle Position Using LSTM and Feedforward NN.....	55
5.4	Real Dynamic Simulated Data Test	62
5.5	Gyro-rate To Gyro-angle Test With Strapdown	63
5.5.1	Gyro-rate To Gyro-angle Test With Simulation using Strapdown Technique NN	65
5.5.2	Gyro-rate To Gyro-angle Test With Real Data Using Strapdown Technique NN	70
6.	Conclusion and Outlook	73
	Bibliography	75

List of Figures

Figure 1.1: Low-cost MEMS IMU MTi-G-710-2A8G4 (small orange box) mounted on the turntable at Universität der Bundeswehr (UniBw) Munich	3
Figure 2.1: Accelerometers measure acceleration in 3 axes: x, y and z axes [2].....	5
Figure 2.2: Operation principle of MEMS' accelerometer [13].....	7
Figure 2.3: Gyroscope [18].....	8
Figure 2.4: Angular motion on the 3 axes in MEMS, where φ is in the x axis, θ is in the y axis and ψ is in the z axis [2]	9
Figure 2.5: Schematic architecture of a single proof mass vibrating gyroscope [16].....	10
Figure 2.6. $\sigma\tau$ vs cluster time τ , Allan variance analysis noise terms results [19]	16
Figure 4.1: Training-Validate-Test Error [26].....	25
Figure 4.2: A single layer FNN [33]	26
Figure 4.3: A multi-layer FNN [33]	26
Figure 4.4: The structure of the LSTM network [22]	29
Figure 4.5: Concept of backpropagation [11].....	32
Figure 5.1: Acuitas Turntable at UniBw.....	36
Figure 5.2: Static Test structure with LSTM	38
Figure 5.3: Gyroscope simulation plots	39
Figure 5.4: Accelerometer simulation plots.....	39
Figure 5.5: Training results on single CPU for accelerometer static results with LSTM	42
Figure 5.6: Training results on single CPU for gyroscope static results with LSTM	43
Figure 5.7 (a), (b), (c), (d): (a) Accelerometer static results with LSTM; (b) Zoomed accelerometer static results with LSTM; (c) Gyroscope static results with LSTM; (d) Zoomed gyroscope static results with LSTM	44
Figure 5.8: Dynamic test structure with LSTM.....	45
Figure 5.9 (a), (b), (c), (d): (a) Dynamic simulation plots. (b) Zoomed dynamic simulation plots. (c) Normalized dynamic simulation plots. (d) Zoomed normalized dynamic simulation plots.	47
Figure 5.10: Training results on single CPU for accelerometer dynamic results with LSTM	48

Figure 5.11(a), (b): (a) Accelerometer dynamic results with LSTM. (b) Accelerometer dynamic results showing the difference between predicted and reference with respect to the bias	49
Figure 5.12: Training results on single CPU for gyroscope dynamic results with LSTM.....	50
Figure 5.13 (a), (b): (a) Gyroscope dynamic results with LSTM. (b) Gyroscope dynamic results showing the difference between predicted and reference with respect to the bias in LSTM	51
Figure 5.14: FNN Training Structure	52
Figure 5.15 (a), (b): (a) LSTM to FNN. (b) Predicted minus reference results of LSTM to FNN	53
Figure 5.16 (a), (b): (a) Gyroscope dynamic results with FNN only. (b) Gyroscope dynamic results showing the difference between predicted and reference with respect to the bias threshold in FNN.	54
Figure 5.17 (a), (b): (a) Gyroscope dynamic results from LSTM after first passing it through FNN. (b) Gyroscope dynamic difference plot of predicted and reference data from LSTM computation after first passing it through FNN	55
Figure 5.18: Training results on single CPU for gyroscope dynamic results with LSTM after first passing it through FNN	56
Figure 5.19: Step simulation data from -1 to +1 rad/s	57
Figure 5.20 (a), (b), (c), (d): (a) and (b) LSTM ONLY static step results. (c) and (d) FNN ONLY static step results	58
Figure 5.21 (a), (b), (c), (d): First scenario test with 48000 points from +1 to -1 rad/s. (a) and (b) LSTM ONLY static step results. (c) and (d) FNN ONLY static step results.....	59
Figure 5.22 (a), (b), (c), (d): Second scenario test with 24000 points from -1 to +2 rad/s. (a) and (b) LSTM ONLY static step results. (c) and (d) FNN ONLY static step results	60
Figure 5.23 (a), (b), (c), (d): Third scenerio test with dynamic data of 1000 points. (a) and (b) LSTM ONLY dynamic results. (c) and (d) FNN ONLY dynamic results.....	61
Figure 5.24: Averaged step static points used for training	61
Figure 5.25 (a), (b), (c), (d): Forth scenerio test with static step data from -1 to +1 rad/s using averaged static step points as trained data. (a) and (b) LSTM ONLY (c) and (d) FNN ONLY	62
Figure 5.26: Real dynamic data collected from the turntable	63
Figure 5.27: Strapdown Technique structure with NN	64
Figure 5.28 (a), (b): (a) Heading data collected as error free from 0 deg to 360 deg. (b) Noisy data collected as static positioning from 0 deg to 360 deg	66

Figure 5.29 (a), (b): (a) Results from Strapdown technique, with testing on 20% of the unseen data.	
(b) Difference between predicted and reference gyro angles.	67
Figure 5.30 (a), (b), (c), (d): New gyrorate to gyroangle trajectory 1. (a) heading (target) in rad and rad/s. (b) gyrorate plots in x,y,z axes. (c) output from prediction. (d) difference between predicted and reference.....	68
Figure 5.31 (a), (b), (c), (d): New gyrorate to gyroangle trajectory 2. (a) heading (target) in rad and rad/s. (b) gyrorate plots in x,y,z axes. (c) output from prediction. (d) difference between predicted and reference.....	69
Figure 5.32 (a), (b), (c), (d): New gyrorate to gyroangle trajectory 3. (a) heading (target) in rad and rad/s. (b) gyrorate plots in x,y,z axes. (c) output from prediction. (d) difference between predicted and reference.....	70
Figure 5.33: Raw step gyrorate data and heading (targets), 0 deg to 360 deg, each with a hold of 120 seconds.....	71
Figure 5.34 (a), (b): (a) Prediction results on real data. (b) Difference between predicted and heading (target) on real data	72

List of Tables

Table 1.1: Main technical specifications of Xsens MTi-G-710-2A8G4 [14].....	4
Table 5.1: Previously computed Xsens IMU error specifications	37
Table 5.2: Bias and scale factor calculation results based on SPST	37
Table 5.3: Static Test Training Options	41
Table 5.4: Coupling effect on axes with respect to the artificial rotation reference rate and earth's gravitational acceleration of gyroscope and accelerometer sensors in Xsens MTi-G-710-2A8G4 IMU	45
Table 5.5: Dynamic test training options.....	46
Table 5.6: Gyro-rate to gyro-angle training options	66
Table 5.7: Summary of performance of the strapdown technique with new simulated trajectories	67
Table 5.8: Real data trained twice to improve the performance of updated data	71

1. Introduction

In recent years, the advancement of positioning systems has improved because of the constellation of satellites available that give positioning, navigation and timing (PNT). Global Positioning System (GPS), for example, which originates from United States of America, is commonly known to serve this purpose. However, even though GPS is good in positioning, it gets affected by occlusions such as buildings or even poor weather conditions that puts its signal tracking to be inaccurate [6]. For this reason, complementary devices are being used to overcome such challenges from satellites. Inertial Measurement Units (IMUs) are a great example of navigation devices used today to help in positioning.

An IMU is an electronic device that combines accelerometers, gyroscopes, and often magnetometers to provide real-time information about an object's acceleration, angular rate, and magnetic field orientation. They are known for their ability to provide continuous navigation information, even in environments where GPS signals may be unreliable or unavailable. They are very useful in various applications from aerospace to wearable devices in healthcare by providing invaluable data on an object's orientation, acceleration, and angular velocity. Among the myriad of IMUs available, those leveraging Micro-Electro-Mechanical Systems (MEMS) technology, such as the Xsens IMU, stand out for their low cost and high performance. The disadvantage of MEMS is that they have low accuracy and require scale and calibration before use [35].

Some calibration techniques have been employed to find solutions to compensate for the errors that degrade the performance of the MEMS IMU. These are the classical and neural network-based algorithms that help in compensating errors associated with low-cost IMUs. Static calibration [9], for instance, involves placing the IMU in different static orientations to characterize sensor biases and

misalignments. By collecting data in various positions and averaging measurements, static calibration compensates for deterministic errors. Another is temperature compensation, because IMU sensors are sensitive to temperature variations, which can affect their accuracy. Temperature sensors are often integrated into IMUs to monitor environmental conditions, and compensation algorithms are applied to adjust sensor outputs accordingly, such as the soak method [4]. In this method, the thermal effect of sensors is investigated to get the local temperature drift compensation models such as the linear interpolation over a range of temperature from -25 degrees Celsius to +70 degrees Celsius (deg C).

Deep learning models are currently being explored as they can learn complex mappings between raw sensor data and calibrated outputs. By training on large datasets containing both raw sensor measurements and corresponding reference values obtained from high-precision instruments, neural networks can effectively model and correct for sensor errors. An example is Calib-Net [23] which can achieve the accurate calibration of low-cost IMU gyroscope measurements via a simple deep convolutional neural network. It employs dilation convolution for spatio-temporal feature extraction of IMU measurements. Calib-Net is driven by a carefully designed loss function to output the compensation for raw gyroscope measurements.

In this thesis, Xsens IMU MTi-G-710-2A8G4 [14] was used for data acquisition and the technical specifications are summarized in Table 1.1. The Xsens IMU incorporates MEMS-based sensors, which are miniaturized mechanical and electrical components fabricated using semiconductor manufacturing techniques. These sensors, including accelerometers and gyroscopes, are integral to the IMU's ability to precisely measure linear acceleration and angular velocity in three-dimensional space in relation to a starting point.

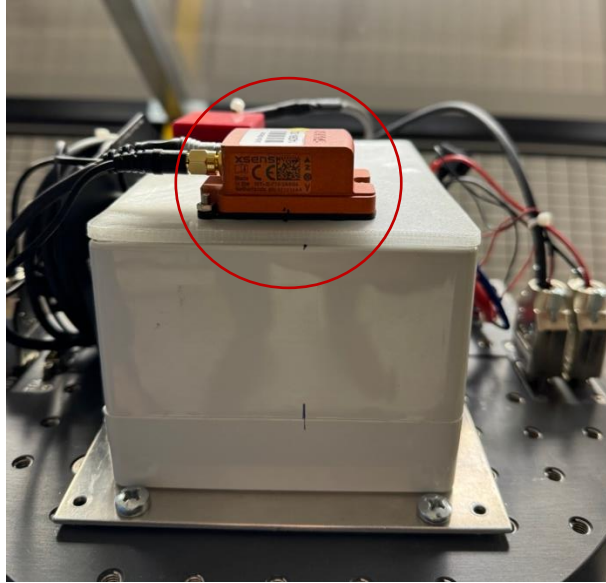


Figure 1.1: Low-cost MEMS IMU MTi-G-710-2A8G4 (small orange box) mounted on the turntable at Universität der Bundeswehr (UniBw) Munich

Figure 1.1 shows an Xsens IMU (small orange box) at UniBw, that was used to collect data for this thesis. What distinguishes the Xsens IMU from conventional counterparts is its cost-effectiveness without compromising on performance. By leveraging MEMS technology, Xsens achieves a delicate balance between affordability and precision, making it accessible to a wide range of applications, from consumer electronics to industrial automation. This introduction sets the stage for a deeper exploration of the capabilities and applications of the Xsens IMU in various fields.

The workflow of the thesis follows a systematic progression towards achieving comprehensive insights and practical solutions in sensor calibration and bias compensation methodologies. Chapter 1 gives a brief introduction of MEMS IMU, why they are being used today and ways error compensation techniques are currently being employed to calibrate the sensors. It also gives the specifications of the Xsens IMU that is being investigated in this thesis. Chapter 2 starts off with introducing the sensors used in IMUs, together with the errors affecting these sensors. Chapter 3

elucidates classical sensor calibration techniques, laying a foundation of fundamental principles. In Chapter 4, the focus shifts towards exploring advanced artificial neural networks, including Feedforward NN (FNN) and Long Short-Term Memory (LSTM) architectures, as potential tools for enhancing sensor performance. Chapter 5 encompasses the rigorous testing and evaluation of various methodologies devised for constant error compensation, aiming to optimize sensor accuracy and reliability. Finally, Chapter 6 serves as the culmination, offering a conclusive summary of findings and an outlook on future research directions, encapsulating the thesis's contributions to the field of sensor calibration.

Table 1.1: Main technical specifications of Xsens MTi-G-710-2A8G4 [14]

Manufacturer	Movella
Operating Temperature	- 40 deg C to + 85 deg C
Sensing Axis	X, Y, Z
Gyroscope	
Standard full range	+/- 450 deg/s
In-run bias stability	10 deg/h
Bandwidth (-3dB)	415 Hz
Noise density	0.01 deg/s/g
g-sensitivity	0.003 <i>deg/s/g</i>
Accelerometer	
Acceleration	20 g
In-run bias stability	15 μ g
Bandwidth (-3dB)	375 Hz
Noise density	60 μ g/ $\sqrt{\text{Hz}}$

2. Inertial Sensors

IMUs based on MEMS utilize a combination of inertial sensors to measure motion-related parameters. These sensors are crucial components in various applications such as navigation systems, robotics, virtual reality, and wearable devices.

2.1 Accelerometer

Accelerometer is an automatic tool for detecting and measuring vibration and measuring acceleration due to the body (inclination) [15]. They are used to measure electronic equipment like 3-dimensional games, telephones and to measure vibrations in cars, engines, buildings.

To understand accelerometers, the fundamentals in physics are needed, where their units are defined based on the International System of Units (SI), and acceleration is termed as a derived unit that is formed by combining base units of length and time. It is also termed as change in velocity over time, where an increase is acceleration, and a decrease is deceleration. Therefore, as velocity is a vector quantity, that has magnitude and direction, acceleration (or even deceleration) is a vector quantity.

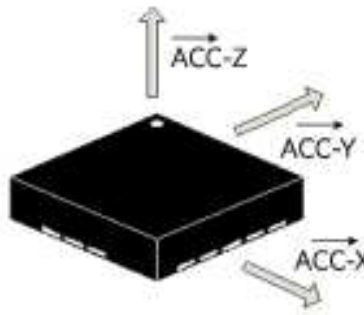


Figure 2.1: Accelerometers measure acceleration in 3 axes: x, y and z axes [2]

The basic working principle of an accelerometer is based on Newton's second law as shown in Equation (2.1), where the specific force is derived from, and takes

measurements in the x, y and z axes as shown in Figure 2.1. If in a stable position, it will measure the acceleration based on freefall, the force of gravity, which is equivalent to 1g (9.81 m/s²).

$$F = m \cdot a \quad (2.1)$$

where:

F : gravitational force

m : mass

a : acceleration

Equation (2.1) then derives the specific force that shows the relationship between gravitational acceleration and gravitational force, which is expressed in Equation (2.2).

$$SF = \frac{F}{m} = a \quad (2.2)$$

where:

SF : specific force

Therefore, based on the configuration in Figure 2.1, assuming it to be a stable position and noise available, the acceleration recorded will be:

$$\begin{aligned} x &= 0g \\ y &= 0g \\ z &= 1g \end{aligned} \quad (2.3)$$

It will only measure 0g in all directions when it is in freefall in a vacuum. Accelerometers within the Xsens IMU also detect changes in velocity over time along the three axes of movement. Utilizing the principles of Newtonian physics, these sensors measure the force exerted on tiny internal masses as the IMU

accelerates, allowing for the calculation of linear acceleration. As expressed in Equation (2.1), the working principle boils down to changing the position of a known mass suspended on springs [13]. One end of the spring is attached to the capacitors, as shown in Figure 2.2 as C1 and C2, while the other end to the mass. From the force acting on the sensor, the mass moves, causing changes in displacement of plates, therefore changing the capacitance. The capacitance will then be measured and processed to a particular acceleration value.

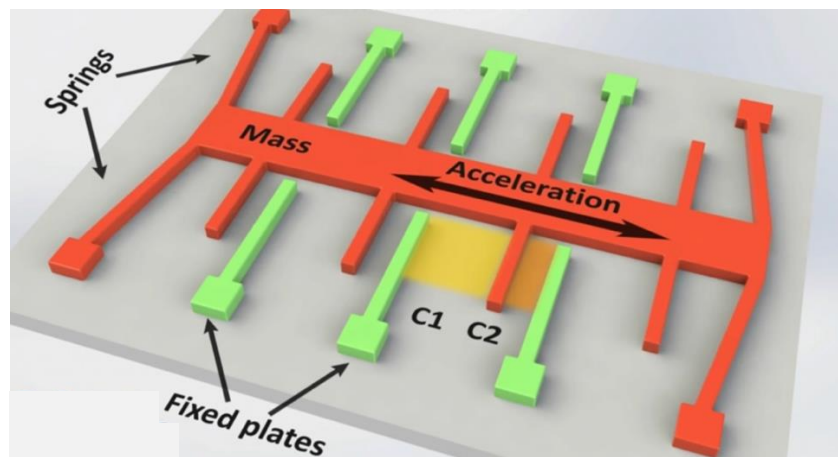


Figure 2.2: Operation principle of MEMS' accelerometer [13]

By integrating these acceleration measurements over time, the IMU can determine changes in velocity and accurately track the device's motion. Also, to buy distance data from the accelerometer sensor, a dual integral process is required for the output sensor [15]. An important note to consider when you choose an accelerometer for application, is some characteristics such as bandwidth, unit is Hertz (Hz), where it indicates the range about vibration of frequency that respond by accelerometer. Other characteristics are amplitude of vibrations, communication interface, environmental conditions like temperature, humidity, electro-magnetic interference, and many more [15].

2.2 Gyroscope

Complementing the accelerometer are the gyroscopes, for example in the Xsens IMU, that provide crucial data regarding rotational movement. A gyroscope is a device used for measuring or maintaining orientation based on the principles of angular momentum. The working principle of a gyroscope relies on the conservation of angular momentum, which states that a spinning object will maintain its axis of rotation unless acted upon by an external torque [18].

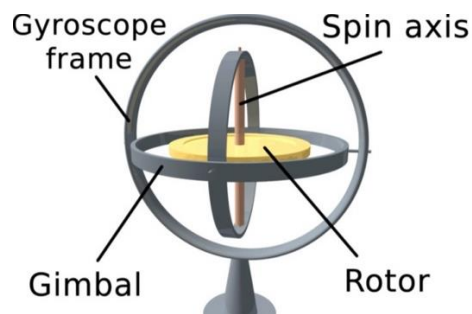


Figure 2.3: Gyroscope [18]

The gyroscope parts as shown in Figure 2.3 are [18]:

- Gimbal: provides support for the spinning mass while allowing it to maintain its orientation relative to the Earth's gravitational field.
- Rotor (spinning mass): central component of the gyroscope. It provides the gyroscopic effect, resisting changes in its orientation once it's set in motion.
- Gyroscope frame: provides stability and holds the gimbal system in place, allowing the gyroscope to be mounted and oriented as needed.
- Spin axis: it is fixed in space once the gyroscope is set in motion to provide the basis for gyroscopic stability and orientation sensing.

Therefore, the rotor is mounted on a set of gimbals, allowing it to rotate freely in three degrees of rotational directions. When this happens, it possesses angular

momentum, which is a vector quantity that points in the direction of the axis of rotation and is proportional to the rate of rotation. From the principle of conservation of angular momentum, this angular momentum tends to remain constant unless an external torque is applied.

$$F_c = -2 \cdot m \cdot (\omega \cdot v) \quad (2.4)$$

where:

F_c : coriolis force

m : mass

ω : angular velocity

v : velocity

When external forces i.e. torques are applied, the rotor's axis of rotation begins to precess and because of the conservation of angular momentum [18], the gyroscope resists this change and tries to maintain its original orientation, which is termed as the gyroscope effect. This effect allows gyroscopes to be used for various applications including navigation, stabilization and measuring angular motion.

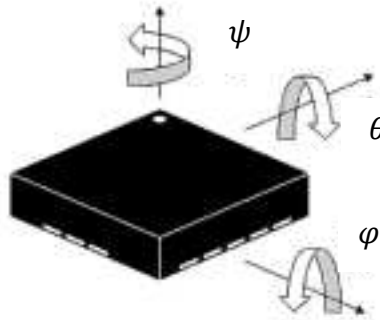


Figure 2.4: Angular motion on the 3 axes in MEMS, where ϕ is in the x axis, θ is in the y axis and ψ is in the z axis [2]

Therefore, MEMS gyroscopes within the IMU sense changes in angular velocity by measuring the Coriolis effect induced by rotational motion [15], which states that in a frame of reference rotating at angular velocity, a mass m moves with velocity and experiences a force, as shown in Equation (2.4).

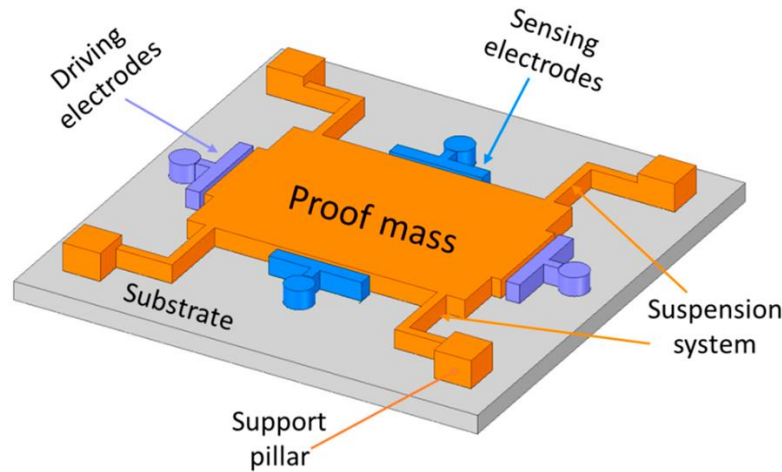


Figure 2.5: Schematic architecture of a single proof mass vibrating gyroscope [16]

As shown in Figure 2.5, The gyroscope operates in two modes: drive and sense [16]. The vibrating proof mass is typically etched onto a silicon wafer using microfabrication techniques, and it is oscillated at a specific frequency after exerted by an electrostatic force during the drive mode. This vibration is maintained by applying a periodic force to the mass. Sense mode comes into play when the gyroscope experiences angular rotation, which results to the Coriolis effect. According to this effect, when an object moves in a rotating frame of reference, it appears to experience a force perpendicular to both its velocity and the axis of rotation. As the frame of the gyroscope rotates, the proof mass remains in its original plane of oscillation due to its inertia. However, from the perspective of the rotating frame (due to the Coriolis effect), the mass appears to move perpendicular to its oscillation plane. This movement of the proof mass is detected using various sensing

mechanisms, such as capacitive sensing, like in the accelerometer. These mechanisms can measure the displacement of the proof mass caused by the Coriolis force.

The detected displacement is converted into an electrical signal, which is proportional to the angular rate of rotation. This signal can then be processed and integrated over time to determine the orientation changes of the device. This enables the IMU to precisely determine changes in orientation and angular velocity around each axis as shown in Figure 2.4, termed as roll, pitch and yaw angles.

2.3 Error Budget

The low accuracy is due to error sources caused by calibration drift. Understanding these errors is essential, and furthermore, modeling these errors is important so that one can compensate them to be able to improve the accuracy of the navigation outcome. One needs to understand how the compensation of the error works beforehand. The output [28] of accelerometers and gyroscopes can be written as:

$$\hat{f} = [I + sf_a] \cdot f + N_a \cdot f + b_a + w_a \quad (2.5)$$

$$\hat{\omega} = [I + sf_g] \cdot \omega + N_g \cdot \omega + b_g + w_g \quad (2.6)$$

where:

\hat{f} : accelerometer-derived specific force error vector

$\hat{\omega}$: gyroscope-derived angular velocity error vector

f : accelerometer true specific force vector

ω : gyroscope true angular velocity vector

N : skew-symmetric matrix containing the non-orthogonality

I : Identity matrix

b : bias

w : noise

sf : scale factor

As shown in Equations (2.5) and (2.6), the derived accelerometer and gyroscope values are affected by different errors. These errors amount to poor performance of the IMU. There are two main types of error sources in IMUs; first are deterministic errors, that include sensor bias, scale factors, misalignment, cross-coupling. Second are stochastic errors, that include random based errors such as Random Walk, Bias Instability and many more. Other errors are environmental errors such as vibration, shock, humidity, pressure, and temperature variation, which also cause drift or changes in calibration over time.

2.3.1 Deterministic Errors

For the deterministic errors, also known as constant errors, sensor bias and scale factors are the major sources of error in gyroscopes and accelerometers, which are related to non-orthogonality of the axes [12].

- The bias is said to either start constant and stay that way, zero drift or it might increase with time due to aging or other factors, like non-zero drift [1].
- The scale factor of the IMU, which relates the analog or digital signal of the IMU to linear acceleration or angular velocity, might have an error which either linearly, or non-linearly, depend on the IMU measurement [1].
- Misalignment errors would then appear when assembling multiple sensors into a three-dimensional system, which bring about negative effects on the system accuracy [25].
- Cross-coupling effects include the coupling errors among the measurements of three-axis accelerations and three-axis angular rates [25].

2.3.2 Stochastic Errors

For the stochastic errors [29], there are different types of random noises involved:

- Quantized noise (QN) is the small difference between the real amplitude of the point under sampling and the bit resolution of the Analog-to-Digital Converter (ADC)
- Random Walk (also referred to as White Noise) is the random amount added to the actual signal and with a long-term average equal to zero. In gyroscopes, it is referred to as Angle Random Walk (ARW) while in accelerometers, it is referred to as Velocity Random Walk (VRW).
- Bias Instability (BI), which is also referred to as Flicker Noise, is related to the instability of the bias offset for a sensor output measured in ideal environment.
- Rate Random Walk (RRW) or Acceleration Random Walk (AccRW) in gyroscopes and accelerometers respectively is a random process of uncertain origin, possibly known as a very low frequency noise term. Mathematically, it is the integral of white noise.
- Drift Ramp (DR) is a very low frequency noise process that is often called Drift Rate/Acceleration Ramp (DRR/DAccR) for gyros and accelerometers, respectively.

Various compensation techniques, often referred to as calibration techniques, have been used to identify these errors. Allan Variance is a method for modelling the random noise and instability in inertial sensors, including those found in IMUs. It's a statistical analysis method commonly used in the field of sensor calibration and performance evaluation. While Allan variance analysis doesn't directly calibrate the IMU, it provides valuable insights into the sensor's behavior, which can be used to improve calibration procedures. For example, it helps in determining appropriate sensor integration times, evaluating sensor performance specifications, and designing calibration algorithms to mitigate sensor errors.

2.3.2.1 Stochastic Error Modelling – Allan Variance (AV)

This is the method of representing the Root-Mean-Square (RMS) [29] random drift error as a function of averaging times. AV is a time domain analysis technique [19] originally developed to study the frequency stability of oscillators. It can be used to determine the character of the underlying random processes that give rise to the data noise. This is done by estimating the magnitude of each noise source covariance from the data. As it is not a standalone calibration technique, it complements classical calibration methods by providing a quantitative assessment of sensor performance.

ARW, RRW, BI, QN and DR are the five noise terms found using AV [19]. Consider N samples of data from a gyro with a sample time of τ_0 . Form data clusters of durations $\tau_0, 2\tau_0, \dots, m\tau_0$ (where $m < (N-1)/2$) and obtain averages of the sum of the data points contained in each cluster over the length of that cluster. AV is defined as the two-sample variance of the data cluster averages as a function of cluster time. How AV is calculated:

Log N stationary gyroscope samples with a sample period τ_0 . Let Ω be the logged samples. AV can be defined either in terms of the output rate or the output angle (the quantity measured), therefore, for each sample, calculate the output angle θ :

$$\theta(t) = \int^t \Omega(t') dt' \quad (2.7)$$

The lower integration limit is not specified as only angle difference are employed in the definitions. For discrete samples, the cumulative sum multiplied by τ_0 can be used. Next, calculate the AV:

$$\sigma^2(\tau) = \frac{1}{2\tau^2} < (\theta_{k+2m} - 2\theta_{k+m}\theta_k)^2 > \quad (2.8)$$

where $\tau = m\tau_0$ and $\langle \cdot \rangle$ is the ensemble average. The ensemble average can be expanded to:

$$\sigma^2(\tau) = \frac{1}{2\tau^2(N-2m)} \sum_{k=1}^{N-2m} (\theta_{k+2m} - 2\theta_{k+m}\theta_k)^2 \quad (2.9)$$

Finally, the AV deviation $\sigma(t) = \sqrt{\sigma^2(t)}$ is used to determine the gyroscope noise parameters. To obtain the noise parameters for the gyroscope, AV is related to the two-sided power spectral density (PSD) of the noise parameters in the original dataset Ω :

$$\sigma^2(\tau) = 4 \int_0^\infty S_\Omega(f) \frac{\sin^4(\pi f \tau)}{(\pi f \tau)^2} df \quad (2.10)$$

where $S_\Omega(f)$ is the two-sided PSD.

From equation (2.10), AV is proportional to the total noise power of the gyroscope when passed through a filter with a transfer function of $\sin^4(x)/(x)^2$. This transfer function arises from the operations done to create and operate on the clusters. Using this transfer function interpretation, the filter bandpass depends on τ . This means that different noise parameters can be identified by changing the filter bandpass, or varying τ . Each noise parameter magnitude is computed [19] based on the slope of the relation between AV and cluster time at specific τ values as shown in Figure 2.6.

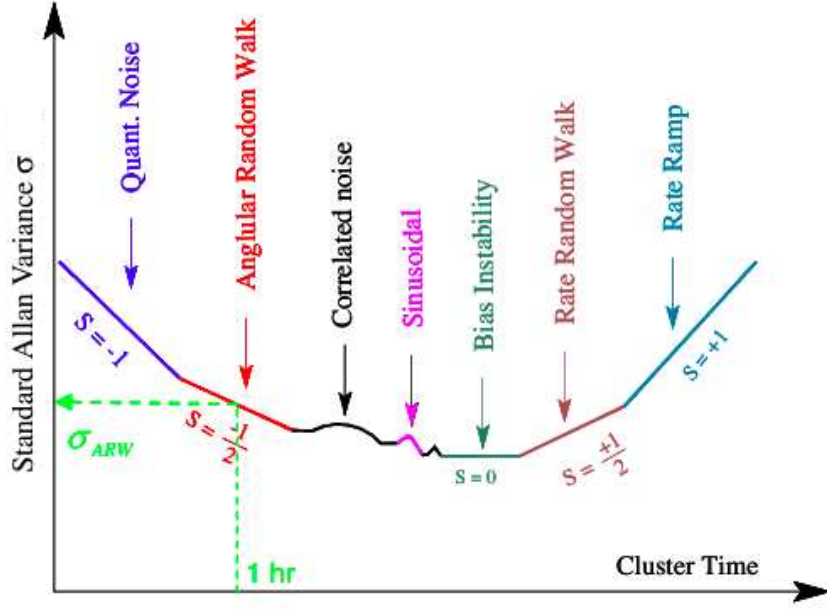


Figure 2.6. $\sigma(\tau)$ vs cluster time (τ), Allan variance analysis noise terms results [19]

Therefore, stochastic errors can be identified using the AV analysis, however, classical, and neural network calibration techniques can only compensate the deterministic errors. Thus, in this thesis, the sensor noises are neglected, making Equations (2.5) and (2.6) be transformed [28] into Equations (2.11) and (2.12) respectively:

$$\hat{f} = [I + sf_a + N_a] \cdot f + b_a \quad (2.11)$$

$$\hat{\omega} = [I + sf_g + N_g] \cdot \omega + b_g \quad (2.12)$$

The compensated sensor outputs can be represented as:

$$f = [I + sf_a + N_a]^{-1} \cdot (\hat{f} - b_a) \quad (2.13)$$

$$\omega = [I + sf_g + N_g]^{-1} \cdot (\hat{\omega} - b_g) \quad (2.14)$$

The purpose is to minimize the difference to get the compensated values.

3. Classical Sensor Calibration

Classical IMU sensor calibration techniques represent a cornerstone in ensuring the accuracy and reliability of IMU data. These techniques typically involve a series of calibration steps aimed at mitigating errors induced by sensor imperfections and environmental factors. Calibration often begins with sensor bias estimation, where offset errors are determined and corrected. Scale factor errors are then addressed through scaling factors adjustment. Additionally, temperature dependencies are accounted for to maintain consistency across varying environmental conditions. Gyroscope and accelerometer calibration are commonly performed separately to optimize performance. Classical IMU calibration techniques, while traditional, remain essential for achieving precise measurements in navigation, robotics, and various other applications reliant on IMU data accuracy.

3.1 Six-Position Static Test (SPST)

It consists of mounting the inertial system [9] on a leveled surface with sensitive x-, y-, and z-axes of the IMU pointing alternatively up and down which results in a total of 6 different positions. The bias (b) and scale factors (sf) can then be calculated using Equations (3.1) and (3.2) respectively:

$$b = \frac{l_f^{up} + l_f^{down}}{2} \quad (3.1)$$

$$sf = \frac{l_f^{up} - l_f^{down} - 2 \cdot K}{2 \cdot K} \quad (3.2)$$

Where l_f^{up} and l_f^{down} are the sensor measurement when the sensitive axis is pointed upwards and downwards respectively. K is the known reference signal which can either be the local gravity constant or the magnitude of the Earth's rotation rate at the given latitude. However, since the low-cost MEMS IMU are low grade

gyroscopes [4], they are often affected by external signals including the Earth's rotation, which can mask or interfere with the gyroscope's measurements. This is so because:

- The Earth's rotation rate is constant, causing a constant angular velocity in the horizontal plane at any given location on the Earth's surface. The constant velocity creates a constant bias or offset in the gyroscope's measurements making it difficult to distinguish between true motion and the Earth's rotation.
- Cross-axis sensitivity from vibrations caused by the earth's rotation and environmental factors like magnetic interference may lead to additional noise and drift in the measurements.
- Gyroscope mounting in certain orientations may align with (or oppose) the earth's rotation, making the sensitive axes affect the measured angular velocity.

These interferences can lead to inaccuracies and errors in the gyroscope's output, particularly in applications where precise measurements are required. Therefore, artificial reference signal of rotation rate is used to be able to compute the gyroscope's constant errors. In this thesis, 1 rad/s is used as a reference signal of the rotation rate.

3.2 Angle Rate Test

Gyroscopes are often assumed to have their axes orthogonal to each other, but due to manufacturing imperfections or mechanical stresses, the axes may not be perfectly orthogonal. Non-orthogonality can introduce errors in navigation systems and affect the accuracy of measurements. For the non-orthogonality [9] errors of the gyroscope, $N_{g,ij}$, the artificial reference signal K is also used. In an angle rate test, the IMU is rotated about each axis independently, clockwise (cw) and counterclockwise (ccw), while measuring the angular rate output from the gyroscope.

$$N_{g,ij} = \frac{l_{ij_cw} - l_{ij_ccw}}{2 \cdot K} \quad (3.3)$$

where l_{ij_cw} and l_{ij_ccw} are the two axes levelled during the constant rotation around the rotation axis. One should make sure to capture data for a sufficient duration to observe steady-state behavior around the rotation axis. For instance, if the rotation is around the z-axis, therefore, the non-orthogonality between the x- and z-axis ($N_{g,xz}$) and the one between y- and z-axis ($N_{g,yz}$) can be estimated.

Given the three deterministic errors found, Equation (2.14) can be solved to get the compensated gyroscope angular velocity vectors.

3.3 Least Squares Method

Using the linear model [4] relationship in Equation (3.4), the estimations of the bias, scale factor and non-orthogonality are done.

$$\mathbf{y} = \mathbf{m} \cdot \mathbf{x} + \mathbf{b} \quad (3.4)$$

where:

\mathbf{y} : derived specific force error vector/ specific angular velocity vector

\mathbf{x} : true specific force vector/ angular velocity vector

\mathbf{m} : scale factor and non-orthogonality

\mathbf{b} : bias

Thus the compensated accelerometer values can be calculated using the linear model:

$$\begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \\ \tilde{a}_z \end{bmatrix} = \begin{bmatrix} sf_x & N_{xy} & N_{xz} \\ N_{yx} & sf_y & N_{yz} \\ N_{zx} & N_{zy} & sf_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \quad (3.5)$$

where:

$\tilde{\mathbf{a}}$: accelerometer derived specific force vector

\mathbf{a} : accelerometer true specific force error vector

sf : scale factor

N : non-orthogonalities

b : bias

In the calibration process, each accelerometer's axis is oriented alternately upwards and downwards for a specific duration.

$$\underbrace{\begin{bmatrix} \tilde{a}_x \\ \tilde{a}_y \\ \tilde{a}_z \end{bmatrix}}_{\tilde{\mathbf{A}}} = \underbrace{\begin{bmatrix} sf_x & N_{xy} & N_{xz} & b_x \\ N_{yx} & sf_y & N_{yz} & b_y \\ N_{zx} & N_{zy} & sf_z & b_z \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} a_x \\ a_y \\ a_z \\ 1 \end{bmatrix}}_{\mathbf{A}} \quad (3.6)$$

The expected acceleration under ideal conditions can be written as Equation (3.7):

$$a_{xup} = \begin{bmatrix} +g \\ 0 \\ 0 \end{bmatrix}, a_{xdown} = \begin{bmatrix} -g \\ 0 \\ 0 \end{bmatrix}, a_{yup} = \begin{bmatrix} 0 \\ +g \\ 0 \end{bmatrix}, a_{ydown} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix}, a_{zup} = \begin{bmatrix} 0 \\ 0 \\ +g \end{bmatrix}, a_{zdown} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \quad (3.7)$$

Therefore, matrix \mathbf{A} is a 4x6 matrix written as Equation (3.8):

$$\begin{bmatrix} a_{xup} & a_{xdown} & a_{yup} & a_{ydown} & a_{zup} & a_{zdown} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (3.8)$$

and the corresponding matrix $\tilde{\mathbf{A}}$ is a 3x6 matrix written as Equation (3.9):

$$\tilde{a}_{xup} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tilde{a}_{xdown} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tilde{a}_{yup} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tilde{a}_{ydown} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tilde{a}_{zup} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \tilde{a}_{zdown} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad (3.9)$$

To find the calibration parameters in \mathbf{M} , the measured acceleration are associated with the ground truth.

$$\mathbf{M} = (\mathbf{A} \mathbf{A})^{-1} \mathbf{A}^T \tilde{\mathbf{A}} \quad (3.10)$$

Matrix \mathbf{M} can be separated to scale factor with non-orthogonalities and bias as shown in Equations (3.5) and (3.6). Then, using Equation (2.13), the compensated accelerometer vectors can be calculated.

4. Artificial Neural Networks

Artificial neural networks (ANNs) [31] are the fundamental building blocks of both machine learning (ML) and deep learning (DL), which are subfields of artificial intelligence (AI). ML encompasses a broader range of algorithms and techniques such as decision trees, for teaching machines to learn from data and make predictions or decisions without being explicitly programmed. DL focuses specifically on training deep neural networks such as convolution neural networks (CNNs) and recurrent neural networks (RNNs) with multiple layers. This enables them to automatically discover intricate patterns and features from large datasets to achieve tasks such as computer vision, speech recognition or translation. Together, these fields drive advancements in artificial intelligence and enable machines to perform increasingly complex tasks with high accuracy. ANN is characterized by three types of parameters [32]; (a) based on its interconnection property (as feed forward network and recurrent network); (b) on its application function (as classification model, association model, optimization model and self-organizing model) and (c) based on the learning rule (supervised/ unsupervised /reinforcement etc.,)

4.1 Supervised And Unsupervised Learning

Learning can refer to either acquiring or enhancing knowledge [32] and in ML, it explains the changes in the system that are adaptive. This means that they enable the system to do the same task, or tasks drawn from the same population more efficiently and more effectively the next time. Referring to ANN, learning occurs by adjusting the free parameters of the network that are adapted where the ANN is embedded.

In supervised learning [32], data is trained from a data source with correct classification already assigned. Such techniques are utilized in feed forward or multilayer perceptron (MLP) models. The objective is to learn a mapping from inputs to outputs based on the provided examples. During training, the ANN adjusts

its parameters (weights and biases) to minimize the difference between its predicted outputs and the true targets. This process typically involves optimizing a predefined loss function, such as mean squared error (MSE) for regression tasks or cross-entropy loss for classification tasks. Supervised learning with ANNs is commonly used for tasks such as classification (e.g., image recognition, spam detection) and regression (e.g., predicting house prices, stock prices).

In unsupervised learning [32], ANNs are trained on unlabeled data, where the objective is to discover hidden patterns, structures, or representations within the data without explicit guidance from labeled examples. The lack of direction for the learning algorithm in unsupervised learning can sometimes be advantageous, since it lets the algorithm to look back for patterns that have not been previously considered. ANNs in unsupervised learning may employ various techniques, including autoencoders, generative adversarial networks (GANs), and self-organizing maps (SOMs). Autoencoders [5], for example, learn to reconstruct input data from a compressed representation, effectively learning a compressed, meaningful representation of the input data. GANs [17] consist of two neural networks, a generator and a discriminator, which are trained simultaneously to generate realistic data samples. Unsupervised learning with ANNs is used for tasks such as clustering (e.g., grouping similar data points together), dimensionality reduction (e.g., reducing the number of features while preserving relevant information), and generative modeling (e.g., generating new data samples similar to the training data).

Therefore, ANNs can be applied to both supervised and unsupervised learning where supervised learning focuses on learning mapping from inputs to outputs based on labeled data while unsupervised focuses on discovering hidden patterns or representations within unlabeled data.

4.2 Data Handling

In ML, there are different categories of data involved to help the neural network model perform. Here, we have, training, testing, validation, and target data. Train-test-validate technique is used to reduce model fitting. Training a network is basically getting the best values for weights and biases inside the network, thus they play a crucial role. In neural networks [26], weights represent the strength of connections between neurons while biases allow neural networks to fit more complex patterns in the data.

The data collected is divided into training, testing and validation. Target data, which is mostly ground truth when dealing with positioning, helps in mapping the collected data outcomes by predicting the datasets to converge to it. Usually, training and testing is enough for the neural network [26], where the data is divided into, for example 80% and 20% respectively. However, because of overfitting, which occurs when the training algorithm runs too long, the data is divided into training, validation and testing, for example as 60%, 20% and 20% respectively. After training the data in the network, the accuracy of the model is compared with new, unseen data, in this case, validation data to see the performance.

Here, the previously adjusted weights are applied to the validation dataset. Depending on whether the validation error starts to increase, you should end the training and use the weight values at the epoch change point as shown in Figure 4.1. Test data is used to make another final comparison with unseen data into the network to see how it will perform, given the optimized weights from validation.

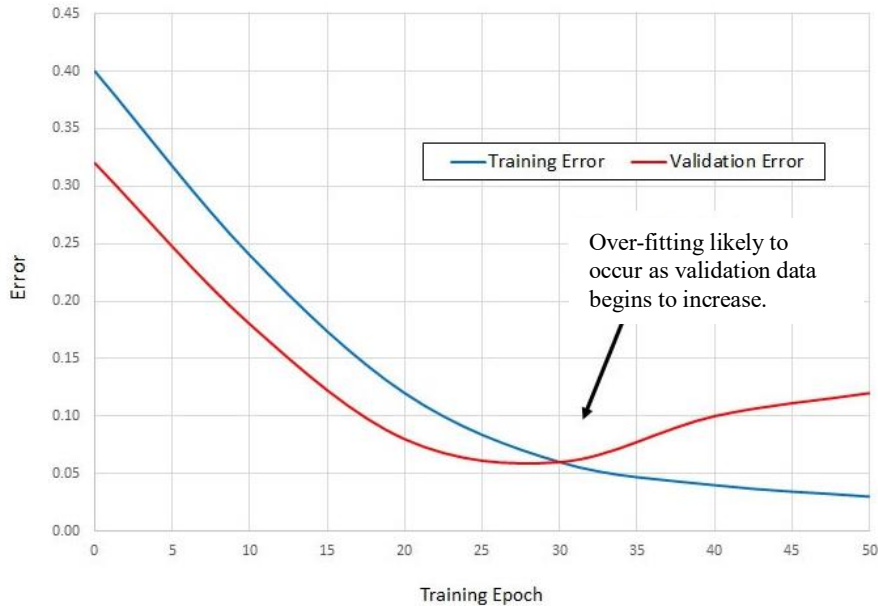


Figure 4.1: Training-Validate-Test Error [26]

4.3 Feedforward NN

In feedforward neural network (FNN) [33], often called Multi-Layer Perceptron (MLP), information flows in one direction: from the input layer through one or more hidden networks to the output layer. There are no cycles or loops in the network, meaning, there are no feedback connections. FNNs or MLPs are the essentials of deep learning models and can approximate virtually any function accuracy [7], provided enough hidden units are available. There are two categories depending on the number of the layers, either single layer or multi-layer as shown in Figure 4.2 and Figure 4.3. From the two figures [33], the difference can be seen that in a multi-layer FNN, there is (at least) one layer of hidden neurons between the input and output layers. These hidden neurons help the network extract higher-order statistics. On both figures, each neuron is connected to every neuron in the next forward layer, and thus the network is said to be ‘fully connected’ otherwise if some connections were missing, it would be ‘partially connected’.

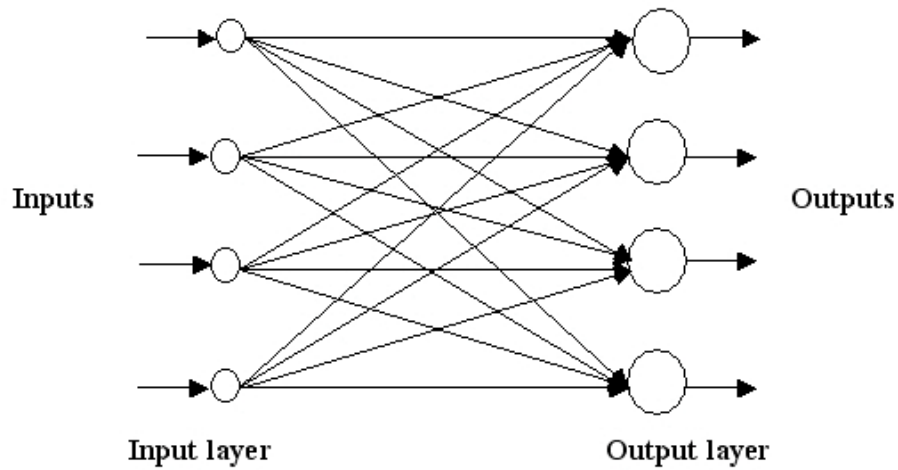


Figure 4.2: A single layer FNN [33]

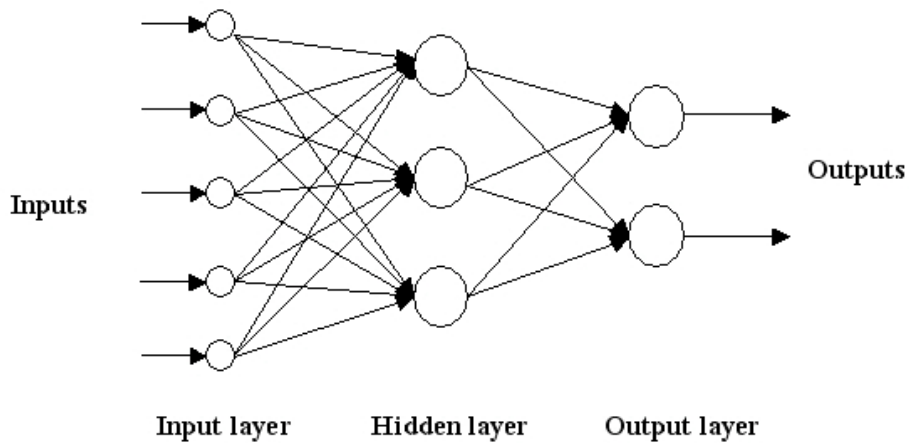


Figure 4.3: A multi-layer FNN [33]

Therefore, FNNs have different parts:

- Input layer that consists of neurons representing the features or dimensions of the input data. Each neuron corresponds to a feature, and the values of these neurons represent the input data.
- Output layer consisting of neurons representing the desired outputs of the network, which depend on the nature of the task.

- Hidden layers between the input and output layer, and each layer consists of neurons that receive input from the previous layer and produce output of the next layer. Neurons in these layers apply a weighted sum of inputs, followed by an activation function to produce their output. The purpose of the activation function is to introduce non-linearity, allowing the network to learn complex relationships in the data.
- Weights and biases that determine the strength of the connection and allows the network to learn different decision boundaries respectively.
- Forward propagation where each neuron computes a weighted sum of its inputs, applies an activation function to produce its output and passes this output to neurons in the next layer.
- Training using supervised learning algorithms like gradient descent.

4.4 Long-Short Term Memory Network

To better understand long-short term memory (LSTM) networks, one needs to understand recurrent neural network (RNN).

Recurrent Neural Networks (RNNs) are deep learning algorithms [31] that process sequential or time-series data by feeding the output from the previous step as input to the current stage. They learn from training input but distinguish themselves by their 'memory'. This enables them to influence current input and output by utilizing information from previous inputs, thereby producing predictive results in sequential data that other algorithms cannot achieve, like how human brains function. The output of a RNN depends on prior elements within the sequence [10]; however, standard RNNs encounter issues like vanishing gradients, making learning long data sequences challenging.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) [31] architecture designed to overcome the limitations of traditional RNNs in capturing

long-term dependencies in sequential data. LSTM has become one of the most popular and widely used architectures in deep learning, particularly for tasks involving sequential data such as natural language processing, time series prediction, and speech recognition.

4.4.1 Structure

Long Short-Term Memory (LSTM), addresses the vanishing gradient [22] problem by using special units. The structure of an LSTM network as shown in Figure 4.4 consists of several components:

- **Memory Cell:** At the core of the LSTM is the memory cell, which maintains a memory state throughout the sequential data processing. The memory cell can selectively remember or forget information based on the input and the current state. This memory cell allows the LSTM to retain information over long sequences, thus addressing the vanishing gradient problem.
- **Three Gates:**
 - **Forget Gate:** The forget gate determines what information from the previous time step should be discarded or forgotten from the memory cell. It takes as input the previous hidden state h_{t-1} and the current input x_t and produces a forget gate activation vector f_t through a sigmoid activation function σ as shown in Equation (4.1). This gate decides which information is no longer relevant and should be removed from the memory cell.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (4.1)$$

where:

W_f : weight matrices of forget gate

b_f : bias of forget gate

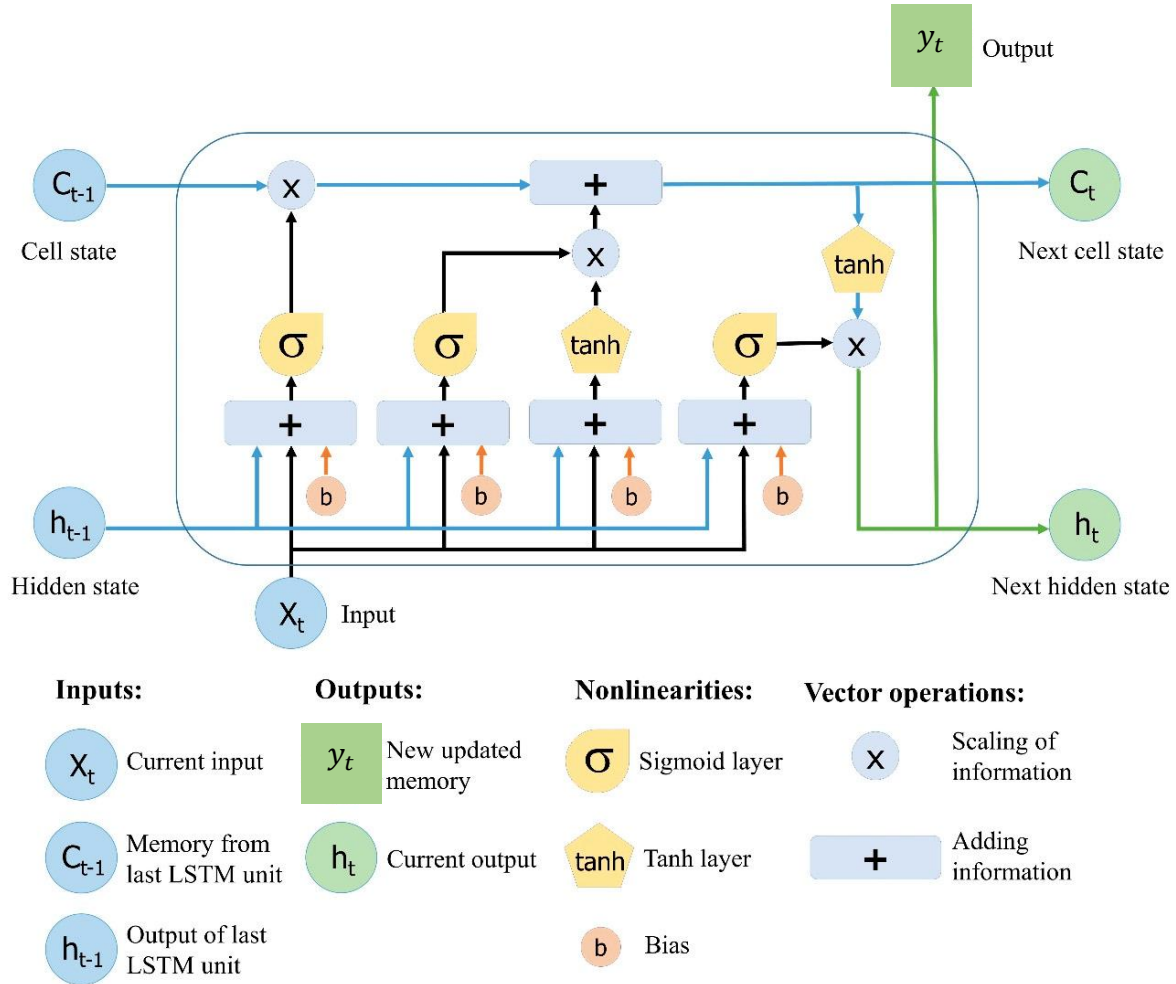


Figure 4.4: The structure of the LSTM network [22]

- Input Gate: The input gate controls the flow of new information into the memory cell. It consists of two parts:
 - A sigmoid layer σ that decides which values of the input x_t should be updated.
 - A tanh layer that generates a vector of new candidate values N_t that could be added to the memory cell.

The input gate activation i_t and the candidate values N_t are then combined to determine the update to the memory cell state as shown in Equations (4.2), (4.3) and (4.4).

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (4.2)$$

$$N_t = \sigma(W_n[h_{t-1}, x_t] + b_n) \quad (4.3)$$

$$C_t = C_{t-1}f_t + N_ti_t \quad (4.4)$$

where:

W : weight matrices of their respective cell state

b : bias of their respective cell state

- Output Gate: The output gate determines which information from the current memory cell state C_t should be output as the hidden state h_t to the next time step. Similar to the forget and input gates, it uses the current input x_t and the previous hidden state h_{t-1} to compute the output gate activation o_t through a sigmoid function. The current memory cell state C_t is then passed through a tanh function and multiplied by the output gate activation to produce the updated hidden state h_t as shown in Equations (4.5) and (4.6).

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (4.5)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (4.6)$$

where:

W_o : weight matrices of output gate

b_o : bias of output gate

- **Cell State Update:** The cell state C_t is updated based on the information from the forget gate, input gate, and the current cell state. The forget gate decides what information to discard, the input gate decides what new information to store, and the output gate controls how much information from the cell state is passed to the next time step.

Therefore, in this gating system [24], LSTM prevents past information from being entirely discarded, filters irrelevant input information, and achieves higher precision in modeling time-variant behavior.

Standard DL algorithms assign a weight matrix to its inputs [11] and then produces the output. These weights are tweaked during training for both gradient descent and backpropagation to reduce the error margins involving the predicted data and target (reference data). Various hyper-parameters are tweaked to find the optimum conditions for training and knowing how the datasets are handled in the LSTM is important to help in adjusting these parameters.

4.4.2 Back Propagation

Backpropagation is nothing but going backwards through your neural network [11], as shown in Figure 4.5, enabling the network to learn from its mistakes by adjusting its parameters, such as weights and biases, to minimize the difference between its predictions and the actual targets. To understand back propagation (also known as backward pass), one needs to understand the forward pass.

During the forward pass [24], input data is fed into the network sequentially, e.g. LSTM, one time step at a time. At each time step, the LSTM processes the input x_t along with the previous hidden state h_{t-1} and cell state C_{t-1} . The LSTM updates its internal states based on these inputs and produces an output y_t along with a new hidden state h_t for the current time step as shown in Figure 4.4. After processing all time steps, the network's output sequence is compared to the target sequence to

calculate the loss using a predefined loss function, such as mean squared error (MSE) for regression tasks or cross-entropy loss for classification tasks.

The backpropagation process starts with computing the gradients of the loss with respect to the parameters of the LSTM network. Gradients are calculated recursively through time using the chain rule of calculus. Since the LSTM is unrolled across time steps, gradients are backpropagated through the entire sequence, which is why this process is often referred to as Backpropagation Through Time (BPTT). At each time step the gradients of the loss with respect to the output $\frac{\partial L}{\partial y_t}$ and the hidden state $\frac{\partial L}{\partial h_t}$ are computed. These gradients are then used to compute the gradients of the loss with respect to the LSTM parameters [11], including the weights and biases in the gates and memory cell. The gradients are accumulated across all time steps and used to update the parameters of the LSTM network using an optimization algorithm such as stochastic gradient descent (SGD) or adaptive moment estimation (Adam).

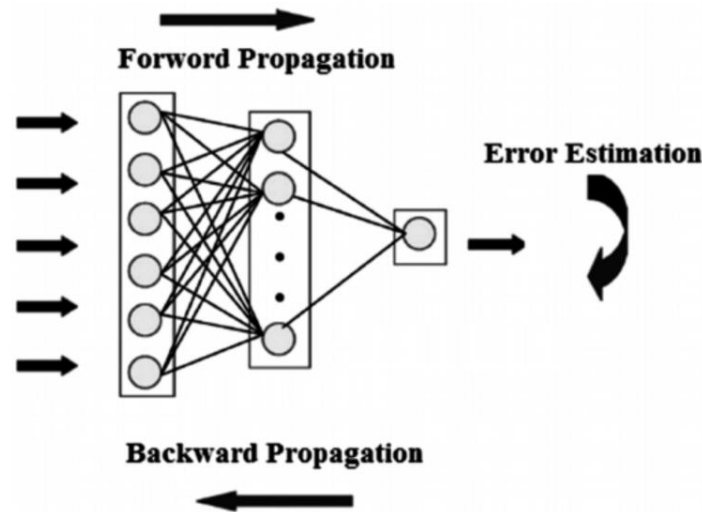


Figure 4.5: Concept of backpropagation [11]

Once the gradients have been computed, they are used to update the parameters of the LSTM network in the direction that minimizes the loss. The learning rate [25],

which controls the size of the parameter updates, is a hyperparameter that needs to be carefully tuned to ensure stable and efficient training. The forward pass, loss calculation, backward pass, and parameter update are then repeated iteratively over multiple epochs until the model converges to a satisfactory solution, or until a stopping criterion is met. These process enables the LSTM to effectively learn from its mistakes and improve its performance over time.

4.4.3 Loss Function

This is simply how well a given machine learning model fits the specific data set. The loss function quantifies [23] the difference between the predicted output of the network and the actual target values. The goal of training is to minimize this loss, thereby improving the accuracy of the model's predictions. The choice of loss function depends on the nature of the task being performed by the network and in regression tasks, as it is in this thesis, the loss is typically computed as the average of the loss values across all time steps in the sequence. Common loss functions used are mean square error (MSE) and mean absolute error (MAE). Depending on the type of data available, these loss functions may not be sufficient and customized loss function would be necessary to tackle some data, especially one that has too much noise, which can make compensation very difficult.

LSTM, therefore, remembers important information in the past [35] by memory and forgetting gates in the neural network. It discards non-critical information suitable for processing information that has correlation in the predicted time series. This can overcome the shortage of the traditional methods in calibration and provide guarantee for subsequent applications.

5. Methodology

This chapter delves into the systematic approach used to gather and analyze data pertaining to gyroscopic and accelerometer measurements. Before embarking on data collection from the turntable available at UniBw, simulation data in MATLAB was utilized to generate static and dynamic gyro rate and acceleration data. This initial step provided insights into the direction of the simulation work once data was acquired. Notably, specific target values for gravitational and rotational rates of low-cost Micro-Electro-Mechanical Systems (MEMS) were established, with 9.81 m/s^2 and 1 rad/s respectively serving as benchmarks.

The methodology proceeds with two primary tests: the static test and the dynamic test. In the static test, biases and scale factors are validated through a six-position static test, which were meant to be removed from the signals using the neural network. The dynamic test introduces varying amplitudes of gyroscope and accelerometer readings, exploring the coupling effects between their axes. Utilizing computed biases and scale factors, the network is then tasked with identifying and rectifying these factors. The same procedures are then conducted on simulated static step gyro rates with constant angle positioning as this can also be recreated in the turntable.

However, challenges arose during the collection of dynamic data from the turntable due to the absence of timestamps to align the reference signal from the turntable with the signal of the IMU, hindering the recording of corresponding reference values. This setback necessitated a return to simulation, to simulate gyrorate to gyroangle tests and employ a strapdown technique. The focus shifted primarily to gyroscope values due to time constraints, with the incorporation of coarse alignment facilitating successful results. Subsequently, real data collection was conducted, mirroring the simulation procedures to validate the neural network's performance. This

methodology chapter outlines the comprehensive steps taken to ensure the accuracy and reliability of the data collected and the subsequent analysis conducted in this study.

5.1 Hardware And Software

Here, the hardware used in Xsens IMU MTi-G-710-2A8G4 [14] calibration includes specialized equipment such as motion-vibration turntable, computer, and temperature monitors. For instance, the temperature was monitored by a thermo-hygrometer device that showed the temperature and humidity variations inside the laboratory. During the experiment, +22 deg C was maintained for collection of data. On the software side, calibration algorithms and software tools are utilized to process the raw sensor data collected during calibration sessions. This involves MATLAB with neural network support libraries that can run the algorithms.

5.1.1 Turntable

The turntable available at UniBw, shown in Figure 5.1 is made by Acuitas [3], which is a company known for its innovative turntable design. Their turntables typically feature precision engineering and high-quality materials, resulting in exceptional sound reproduction. Acuitas turntables often incorporate advanced technologies such as direct drive systems, high-torque motors, and low-resonance platters, ensuring accurate playback of vinyl records. Additionally, Acuitas turntables may offer customizable features and aesthetics, appealing to audiophiles and collectors alike.



Figure 5.1: Acuitas Turntable at UniBw

5.1.2 Computer Specifications

The computer used to run these processes was with a processor of 1.8 GHz Dual-Core Intel Core i5, with a memory of 8 GB, 1600 MHz DDR3 and an intel graphics 6000 card with 1536 MB.

5.1.3 Software

MATLAB was the computer programming software used to write neural network algorithms. Also, this helped to simulate the sensor error values that helped in creating diverse scenarios that may occur in the turntable. It also helped in data augmentation that can enhance the robustness and generalization ability of machine learning models trained on limited real data.

From previously computed error values based on KVH CG-5100 IMU that combines highly accurate fibre optic gyro (FOG) with MEMS [8], the simulation code is built

to have a realistic setup of the Xsens IMU using the data in Table 5.1. These simulation codes were important to understand the performance of neural networks in this thesis using Xsens IMU MTi-G-710-2A8G4.

Table 5.1: Previously computed Xsens IMU error specifications

Parameter	Accelerometer	Gyroscope
Bias	5000 μg	700 deg/h
Scale Factor	1 %	0.15 %
VRW/ARW	60 $\mu\text{g}/\sqrt{\text{Hz}}$	1 deg/\sqrt{h}
BI	15 μg	10 deg/h

The simulation is able to create a realistic up and down sensor measurement of all axes as shown in Figure 5.3 and Figure 5.4. To achieve this, the IMU error specifications from Table 5.1, combined with random values within a defined time-span and frequency are used to generate the up and down signals of the x-, y- and z-axes for gyroscope and accelerometer.

5.2 SPST

Given the data setup in Table 5.1, Equations (3.1) and (3.2) are used to validate the simulation values.

Table 5.2: Bias and scale factor calculation results based on SPST

Parameter	Accelerometer	Gyroscope
Bias	5009.17 μg	707.46 deg/h
Scale Factor	1 %	0.15 %

From Table 5.2, the simulation data is correct based on Equations (3.1) and (3.2). Referring to Equations (2.13) and (2.14), one can remove the scale factor and bias values from the accelerometer and gyroscope values that contain the deterministic and stochastic errors. Given these results, gyrorate to gyrorate & accelerometer to accelerometer tests are done on static and dynamic simulated data using neural networks.

5.3 Gyrorate To Gyrorate AND Accelerometer To Accelerometer Calibration With LSTM and FNN

The calculated bias and scale factors need to be compensated using neural networks. Using the simulation data produced, the static and dynamic tests are done to be able to compensate the bias and scale factor using LSTM and FNN.

5.3.1 Static Test With LSTM

Here, x_{up} component of gyroscope from Figure 5.3 and z_{up} component of accelerometer from Figure 5.4 are used to compensate the simulated data. In here, the IMU sampling rate is undersampled to 10 Hz.

The structure is as follows:

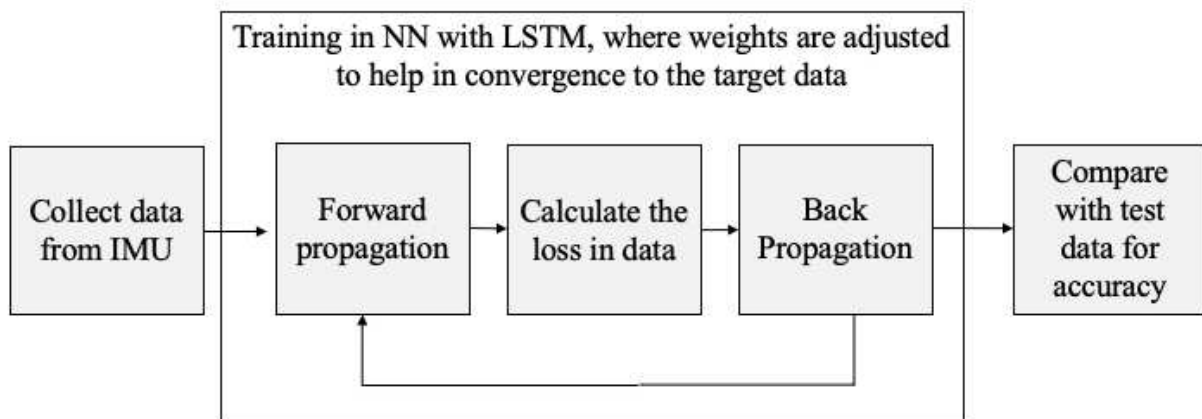


Figure 5.2: Static Test structure with LSTM

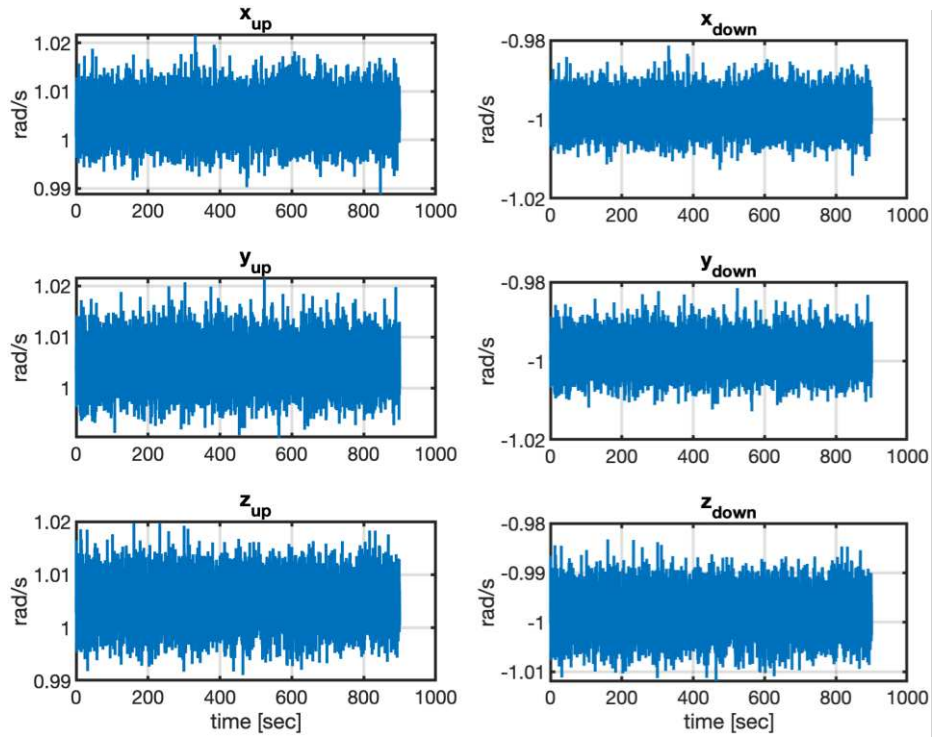


Figure 5.3: Gyroscope simulation plots

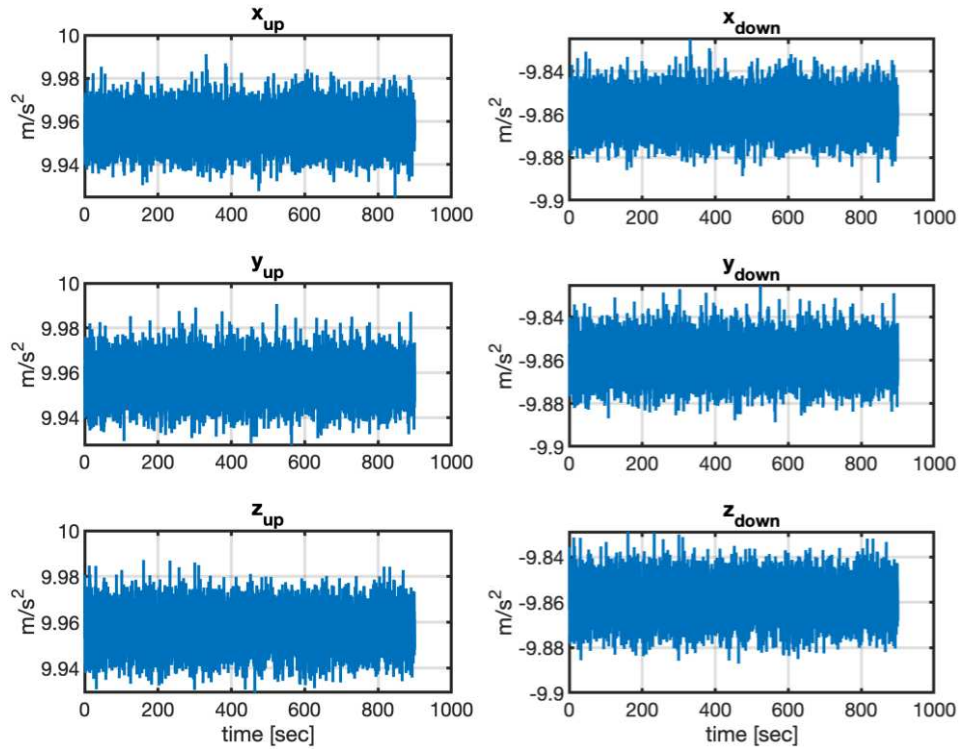


Figure 5.4: Accelerometer simulation plots

- i. Data from the simulation is used as the input data to the NN. For the static test, 9000 data points of each sensor was used where 80% was for training and 20% was for testing. No validation set was used in this setup.
- ii. Inside the NN, forward propagation occurs when the input data is fed through the network in a forward direction. Each hidden layer accepts the input data, processes it based on the dropout layer regularization, which helps with the overfitting problem, and later passes it to the next layer. More training options are shown in Table 5.3.
- iii. As explained in Section 4.4.3, calculating the loss helps in model fitting. Here, MSE regression layer is enough for the loss computation, shown in Equation (5.1).

$$MSE = \sum_{i=1}^R \frac{(Y_i - T_i)^2}{R} \quad (5.1)$$

where:

R : number of responses

T_i : target output

Y_i : network's prediction for response i

- iv. For the backpropagation, this helps in updating the weights of the neural network by calculating the gradient as explained in Section 4.4.2. Equation (5.1) is then transformed to the first derivative as shown in Equation (5.2).

$$\frac{\delta L}{\delta Y_i} = \frac{2}{R} (Y_i - T_i) \quad (5.2)$$

- v. After minimizing the losses, where the difference between the predicted and the target (which is the error free data) are very small, new data, unknown to the neural network, is used to check the accuracy of the trained data. This will

help to check if the updated weights are best to help in convergence as shown in Figure 4.1, to avoid overfitting.

Table 5.3: Static Test Training Options

Training Option	Value
Optimization Algorithm	Adam [21]
Dropout Layer	0.2
Regression Layer	Mean Squared Error (MSE)
Number of Hidden Layers	16
MaxEpochs	15
MiniBatchSize	2^5
Initial Learn Rate	0.001

The performance of the network is represented in Figure 5.5 and Figure 5.6, to monitor the training progress for the network in real-time and to assess the convergence of the model. Here, the root mean square error (RMSE) graph is used as a metric to evaluate the performance of regression models [30], including neural networks. It measures the average magnitude of the errors between predicted values and actual values. The loss graph represents the value of the loss function which measures [23] how well the neural network's predictions match the actual target values during training. The goal is to minimize this value, as a lower loss indicates better performance. This is also supported by Figure 5.7, where the predicted plots (which represents the calibrated values), are now closer to the reference data, indicating a good improvement of the data, which is compensation of constant errors. For both sensors, the noise seems to be suppressed but not completely removed.

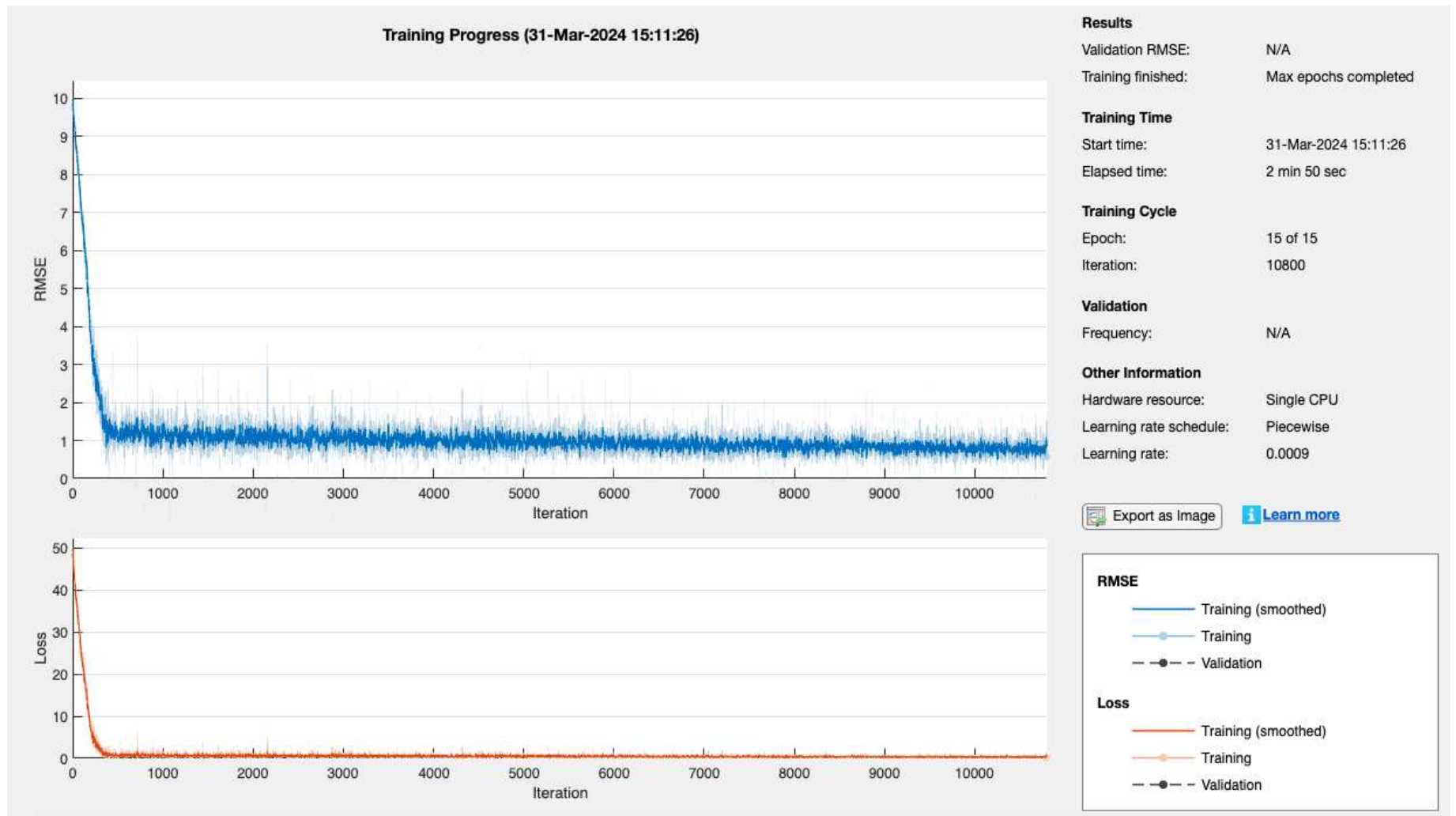


Figure 5.5: Training results on single CPU for accelerometer static results with LSTM

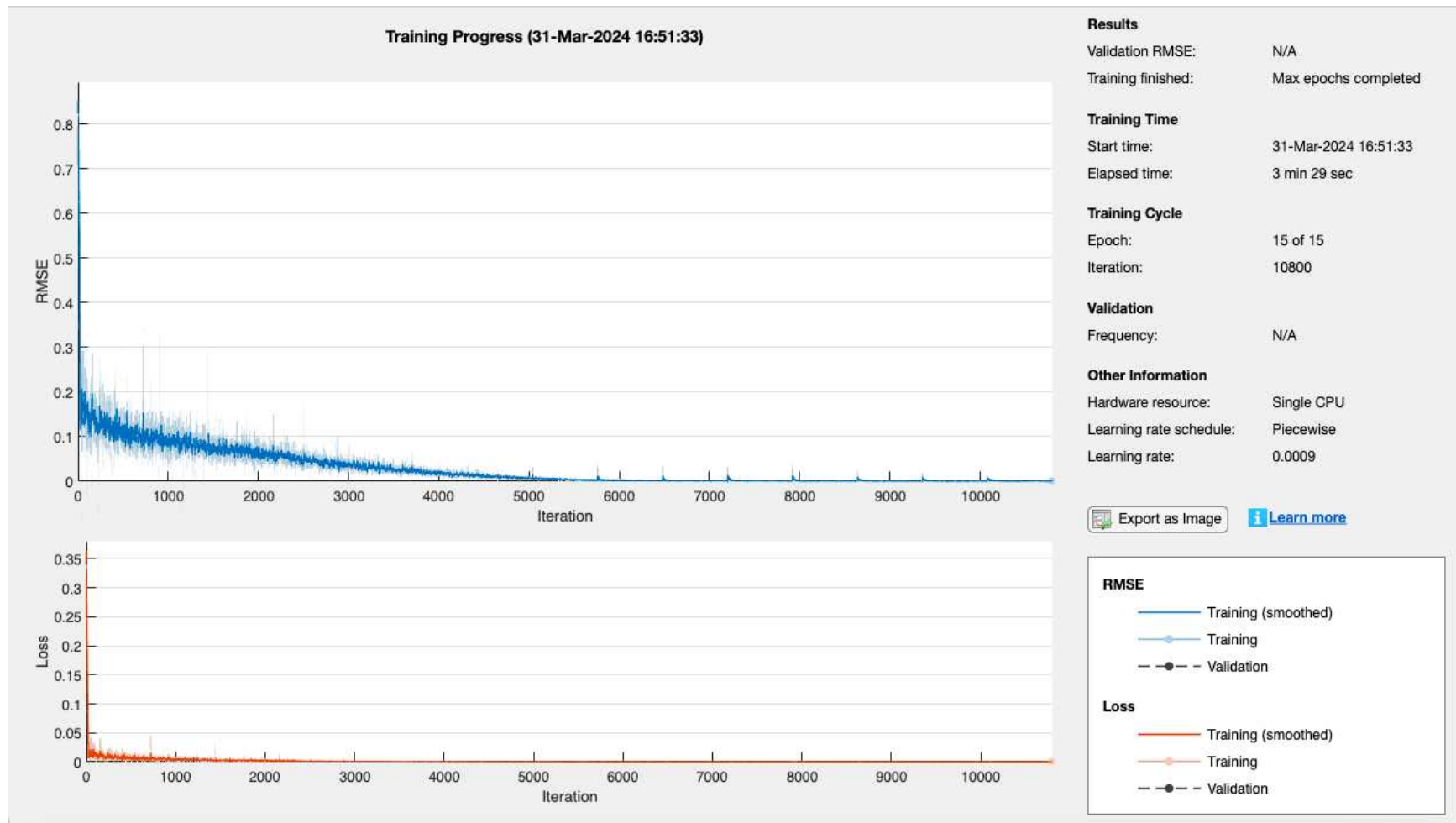


Figure 5.6: Training results on single CPU for gyroscope static results with LSTM

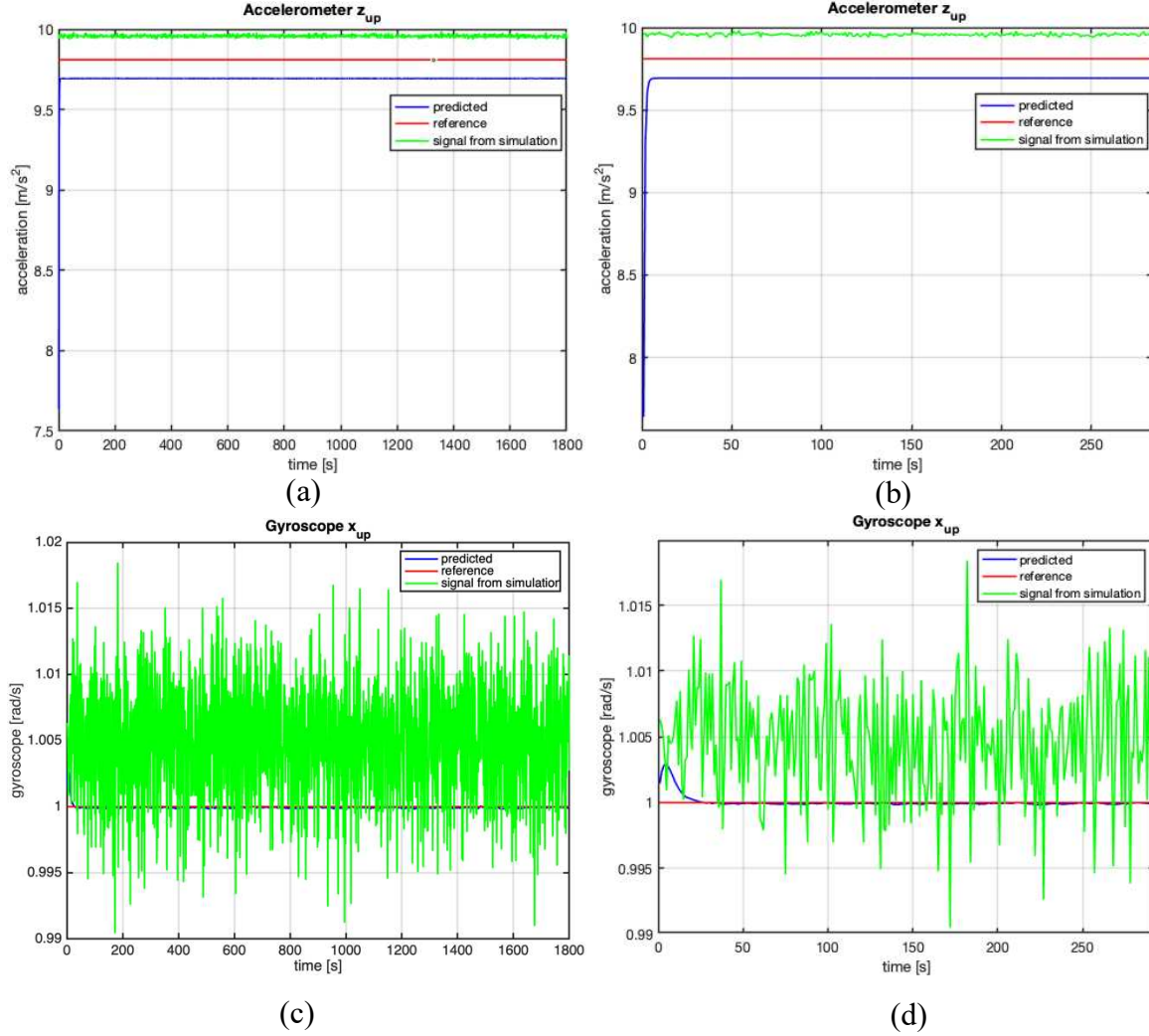


Figure 5.7 (a), (b), (c), (d): (a) Accelerometer static results with LSTM; (b) Zoomed accelerometer static results with LSTM; (c) Gyroscope static results with LSTM; (d) Zoomed gyroscope static results with LSTM

5.3.2 Dynamic Test With LSTM

In the dynamic data, frequency of accelerometer sensor together with the amplitude of gyroscope and accelerometer are added to the simulation. The IMU frequency of 200 Hz is used as the bandwidth in the simulation to be able to collect the data. Similar to how the collection of real data from the turntable, due to vibrations, other axes will be affected. Axes that are larger and show magnitudes closer to the artificial reference rotation rate and the earth's gravity acceleration in both the gyroscope and accelerometer sensors respectively indicate that these are closely aligned to the

reference data. In the Xsens, these coupling of sensors closer to the reference data are shown in Table 5.4. In this simulation, similar stochastic and deterministic errors used in the static test are implemented to simulate the uncompensated data.

Table 5.4: Coupling effect on axes with respect to the artificial rotation reference rate and earth's gravitational acceleration of gyroscope and accelerometer sensors in Xsens MTi-G-710-2A8G4 IMU

Gyroscope	Accelerometer
X	Y
Y	Z
Z	X

The structure is as follows:

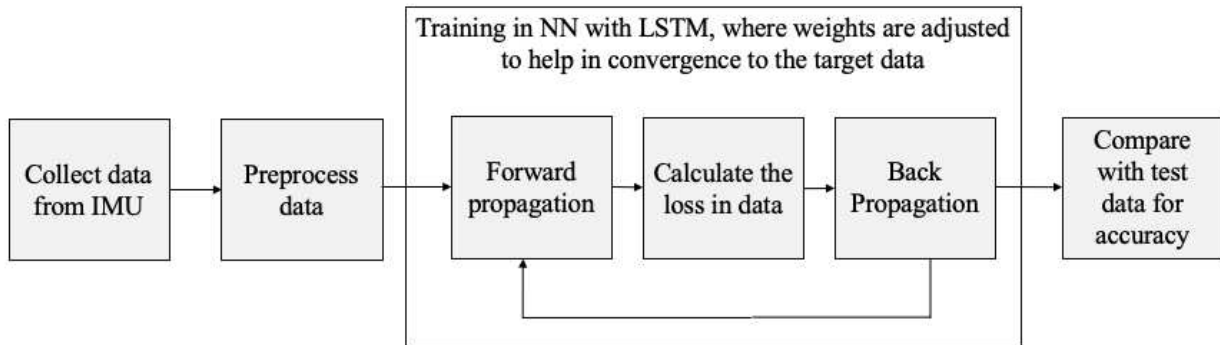


Figure 5.8: Dynamic test structure with LSTM

The steps in Figure 5.8 are quite similar to the static test in Figure 5.2, with the addition of the preprocessing of data before putting it into the network. This helps in eliminating anomalies by making the errors have a similar format to be easier to interpret and use. This is basically standardizing the data to have them in the same range making it easier to train and improve the learning speed. Here, the x component of the gyroscope and y component of the accelerometer are used as inputs

to the neural network shown in Figure 5.9 (a) and (b) and their corresponding normalized data are shown in Figure 5.9 (c) and (d). The training options used for the dynamic simulation are shown in Table 5.5. Validation data is also included to ensure that the trained neural network generalizes well to new, unseen data and helps in optimizing the model's performance and generalization ability. Therefore, for the dynamic test, 1000 data points of each accelerometer and gyroscope are used and are split to 60% training, 20% validation and 20% testing.

Table 5.5: Dynamic test training options

Training Option	Value
Optimization Algorithm	Adam [21]
Dropout Layer	0.2
Activation Layer	tanh
Regression Layer	Mean Squared Error (MSE)
Number of Hidden Layers	90
MaxEpochs	50
MiniBatchSize	2^{10}
Learn Rate Drop Factor	0.2
Learn Rate Drop Period	5
Initial Learn Rate	0.001

Similar to the static test, the performance of the neural network is seen in Figure 5.10 and Figure 5.12 for both the accelerometer and gyroscope respectively.

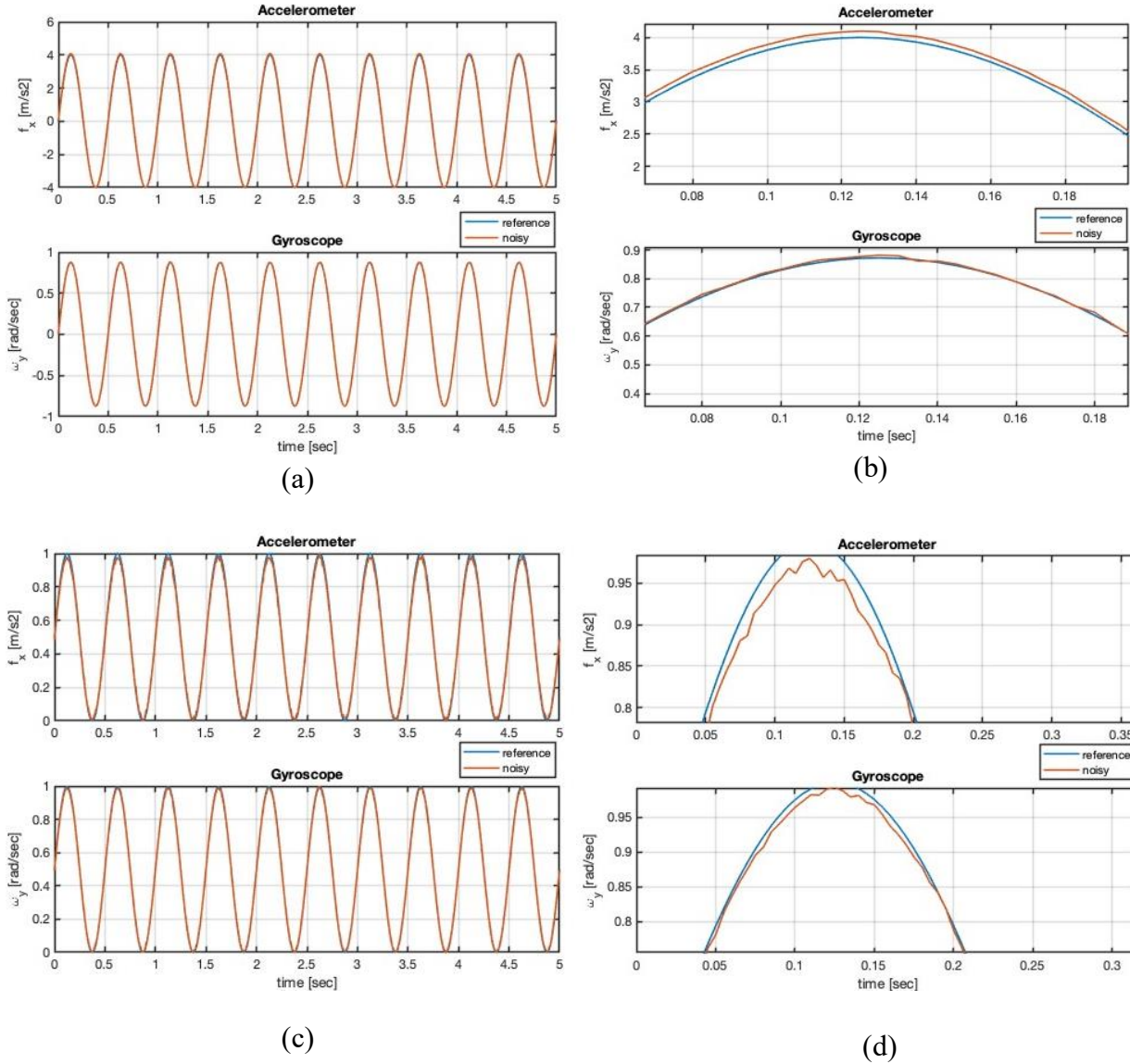


Figure 5.9 (a), (b), (c), (d): (a) Dynamic simulation plots. (b) Zoomed dynamic simulation plots. (c) Normalized dynamic simulation plots. (d) Zoomed normalized dynamic simulation plots.

After training the accelerometer data, the output of the network, represented as predicted plot in Figure 5.11, converges towards the targets. The accelerometer bias is then used as error bounds to check if the network compensated for them. Ideally, both bias and scale factor need to be checked, but as the scale factor only changes the size of the signal while bias shifts the signal away from the reference, it's easier to make better conclusions with the bias.

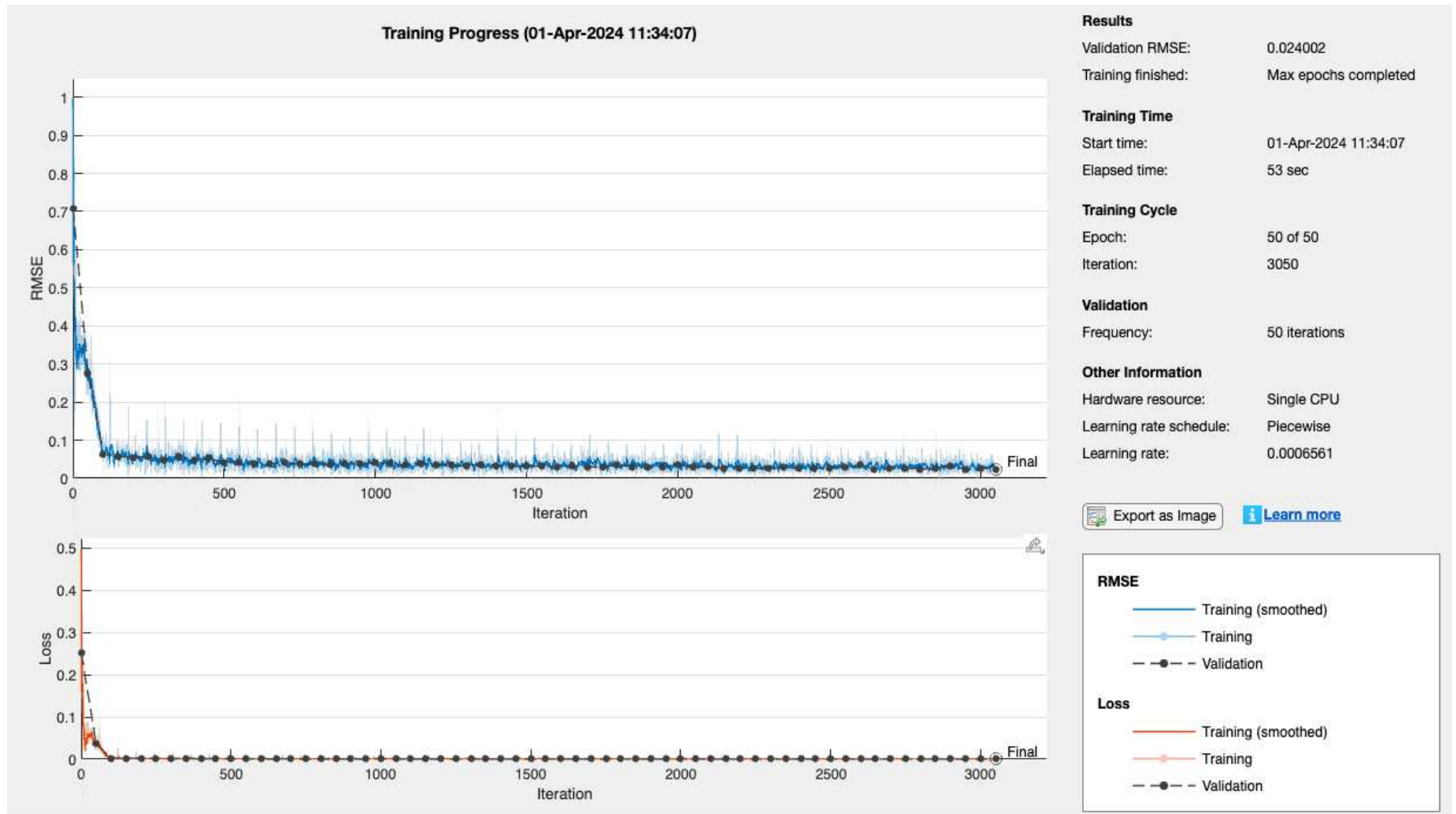


Figure 5.10: Training results on single CPU for accelerometer dynamic results with LSTM

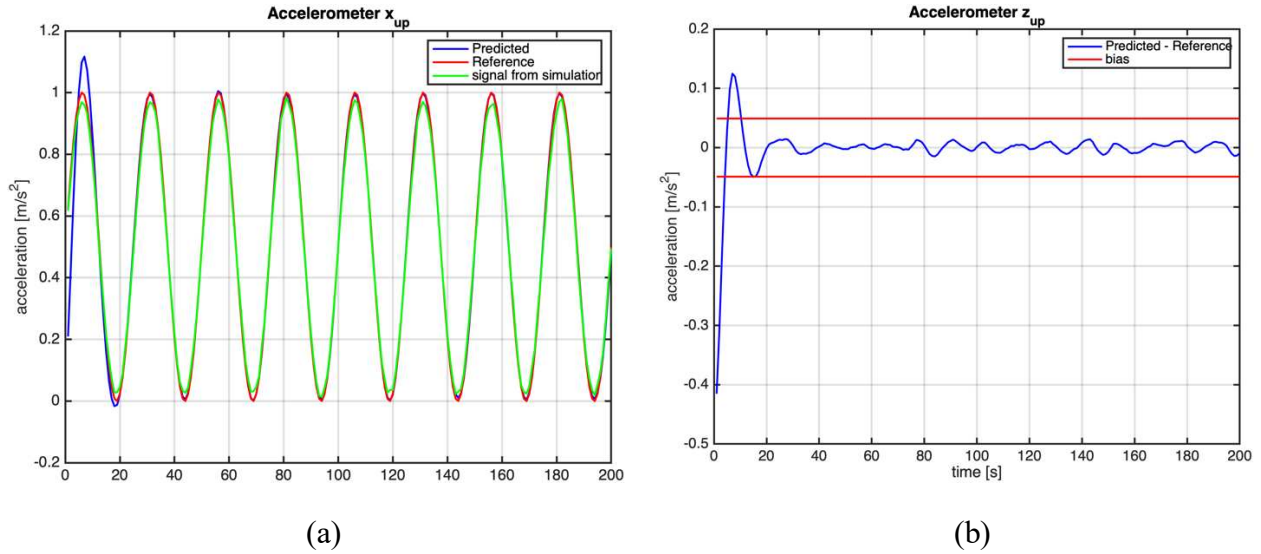


Figure 5.11(a), (b): (a) Accelerometer dynamic results with LSTM. (b) Accelerometer dynamic results showing the difference between predicted and reference with respect to the bias

This bias is the same from Table 5.1, that is used to check if the compensated accelerometer and gyroscope simulated data are free from bias after passing the uncompensated simulated data into the neural network.

In Figure 5.11(b), $\pm 0.049 \text{ m/s}^2$ (equivalent to $5000 \mu\text{g}$) is seen bounding the difference between predicted and reference. The difference should go to zero to show that the bias has been completely removed and that the compensated data is free from bias.

The same analysis is seen in Figure 5.13 for gyroscope, with the bias being $\pm 0.0034 \text{ rad/s}$ (equivalent to 700 deg/h). Comparing the two sensors, the network didn't seem to be able to remove the bias fully in both, however, the performance of the accelerometer is better than the gyroscope. This is because, in the gyroscope, the difference is seen to be crossing the bias bounds, indicating that bias still has a significant influence on the predicted output. Further analysis must be done on gyroscope to understand how the bias can be compensated.

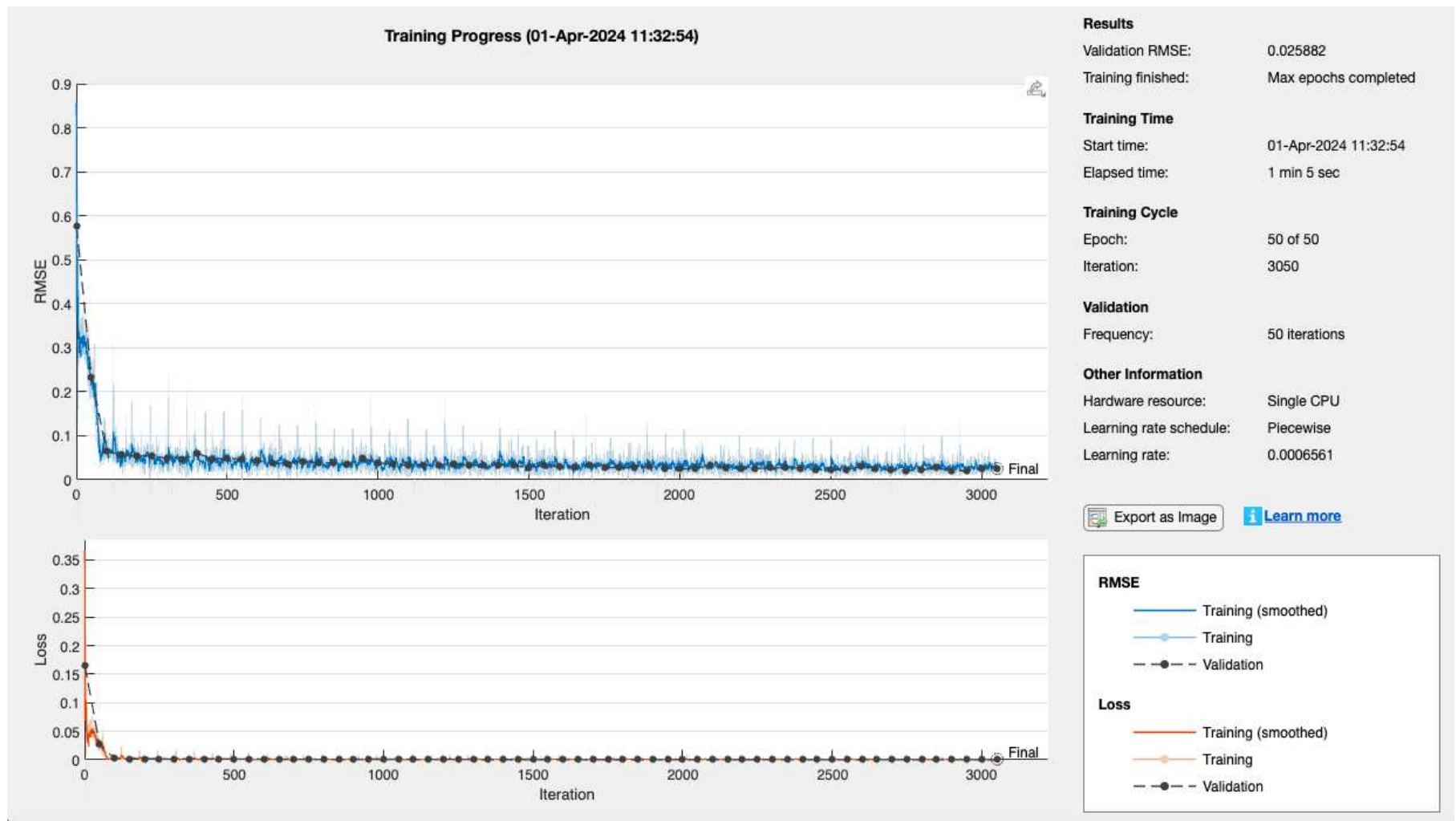


Figure 5.12: Training results on single CPU for gyroscope dynamic results with LSTM

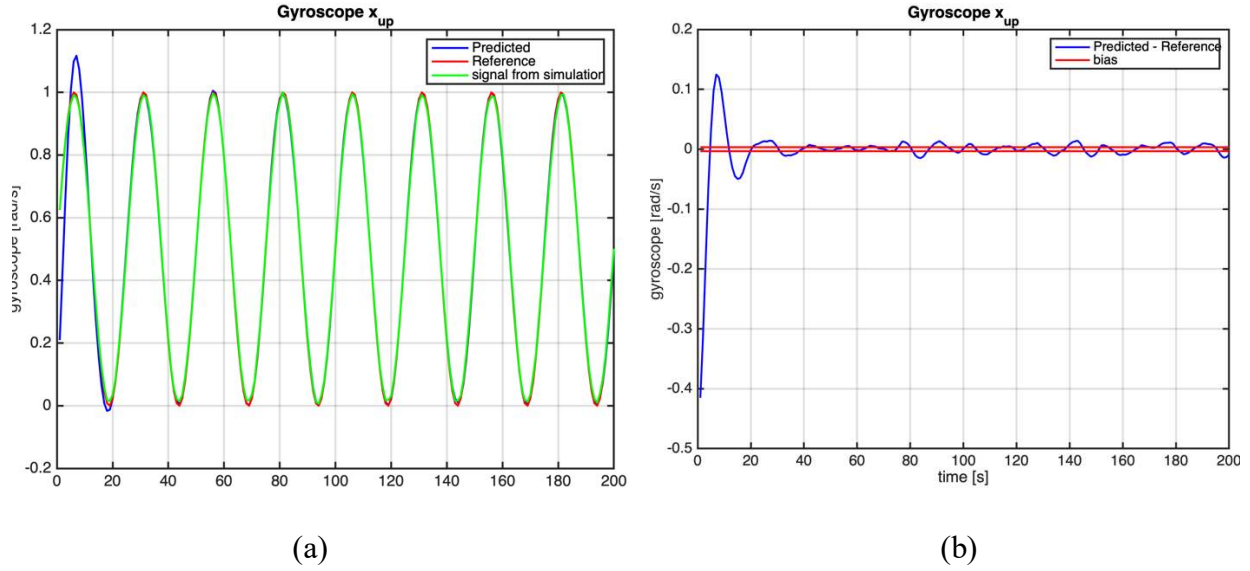


Figure 5.13 (a), (b): (a) Gyroscope dynamic results with LSTM. (b) Gyroscope dynamic results showing the difference between predicted and reference with respect to the bias in LSTM

5.3.3 Dynamic Test With Feedforward NN

Hereon in the thesis, only the gyroscope sensor is checked to find ways to improve the compensation. Different scenarios are done to make such improvements:

- i. After LSTM, pass the predicted values to FNN. It has a single hidden layer, but varying number of neurons in the hidden layer, which are determined based on the size of input and output data, as well as a user-specified parameter known as the 'hidden layer size'.
- ii. Start of by passing the data from the simulation to FNN. Later, pass the predicted output of FNN into LSTM to check how it performs.

The basic structure of FNN used is shown in Figure 5.14, where the hidden layer has 60 neurons and the training algorithm used is Levenberg-Marquardt, that helps to reduce the losses. The Levenberg-Marquardt algorithm [27] is an optimization technique used primarily for solving nonlinear least squares problems. It combines the features of gradient descent methods and Gauss-Newton methods to efficiently converge towards the minimum of a nonlinear objective function.

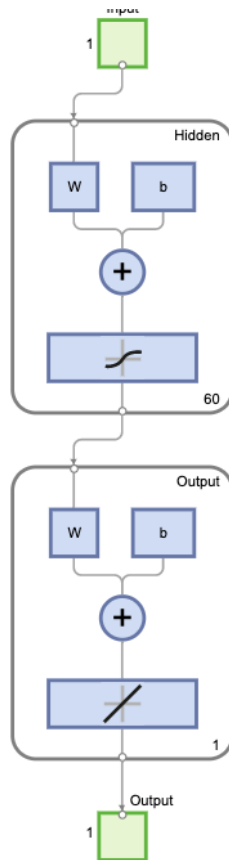


Figure 5.14: FNN Training Structure

5.3.3.1 LSTM To Feedforward NN

The first FNN test is done using the results from LSTM and pass it into FNN. Therefore, from Figure 5.13, take those results and place them into FNN. This is just to find whether with a pretrained network, the results will improve once you combine with another network structure. As shown in Figure 5.15, FNN results converges to zero, thus the bias is being compensated. However, the costs are too high as the data seems to be overfitting.

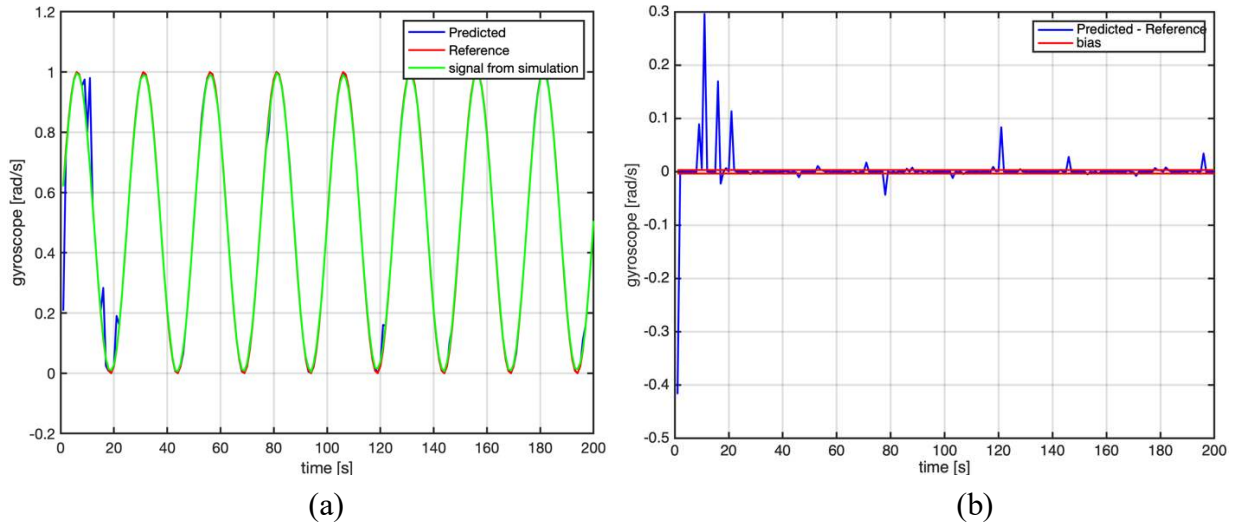


Figure 5.15 (a), (b): (a) LSTM to FNN. (b) Predicted minus reference results of LSTM to FNN

5.3.3.2 Feedforward NN To LSTM

Another try would be to first pass the original dynamic data from the simulation in Figure 5.9 into FNN, maintaining the same settings of using 60 neurons and a training algorithm of Levenberg-Marquardt [27]. This way, we can understand how FNN will perform as a standalone structure. From Figure 5.16, the output results from FNN are shown. At first glance, it may show that the network has fitted well to the reference as shown in Figure 5.16(a), but the difference between predicted (calibrated data) and reference data as shown in Figure 5.16(b) explains that there is an overfit. This may result into problems when testing with different data unseen to the network and it will not predict well. The crossing of the bias threshold as shown in Figure 5.16(b) explains that compensation isn't fully achieved. If you further decide to put the results from FNN into LSTM and check the performance of the data, there seems to be an improved convergence and the bias seems to be compensated, but not perfectly as shown in Figure 5.17.

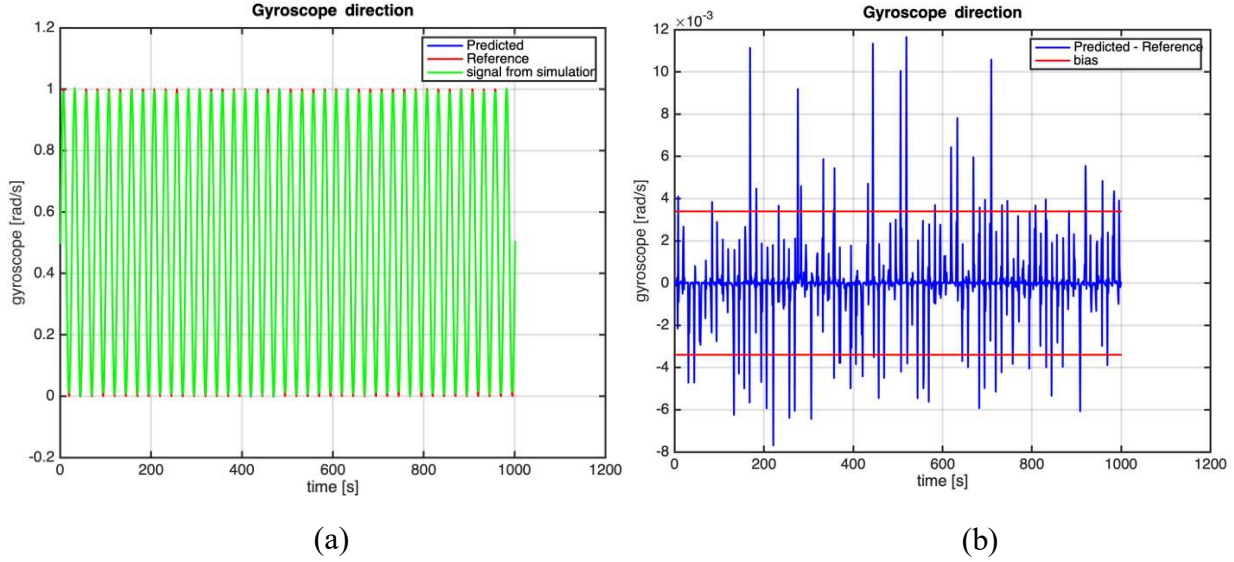


Figure 5.16 (a), (b): (a) Gyroscope dynamic results with FNN only. (b) Gyroscope dynamic results showing the difference between predicted and reference with respect to the bias threshold in FNN.

This is because the difference between predicted (calibrated data) and reference seems to be having some biases all the way, as shown in Figure 5.17(b), indicated by the crossing of the difference plot to the bias bounds. The training options used in Table 5.5 were repeated, with only changes to the number of hidden to 120. The performance of the network is shown in Figure 5.18, where validation data is added and the data is split to 60% training, 20% validation and 20% testing.

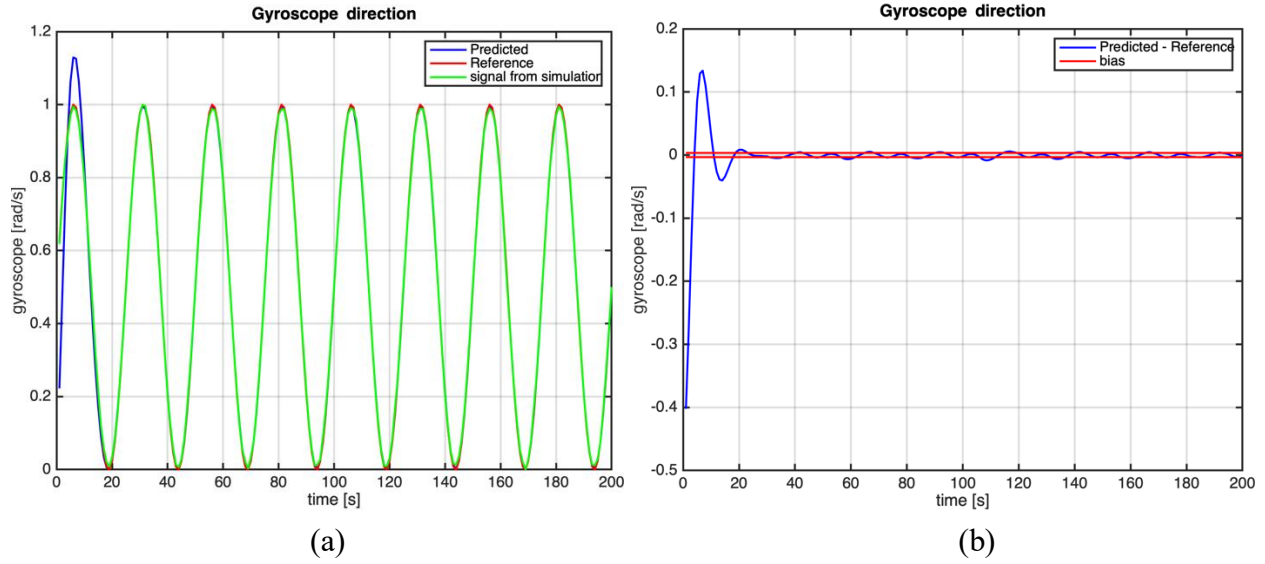


Figure 5.17 (a), (b): (a) Gyroscope dynamic results from LSTM after first passing it through FNN. (b) Gyroscope dynamic difference plot of predicted and reference data from LSTM computation after first passing it through FNN

5.3.4 Static Step Simulation Test Considering Constant Angle Position Using LSTM and Feedforward NN

The same procedures done before are conducted with static step data, simulated from -1 rad/s to +1 rad/s with 60 steps. Here, the simulated data involves static angle rate positions simulated for a constant rate of 120 seconds with an IMU sample rate of 200 Hz, producing 24000 data points as shown in Figure 5.19.

For LSTM, the training options from dynamic test in Table 5.5 are used, only changing hidden layers to 240 and MaxEpochs to 90 while for the FNN, the hidden layer size is now 100 while the training algorithm is maintained as Levenberg-Marquardt [27]. The data is split to 60% training, 20% validation and 20 % testing on both neural network tests and the results are shown in Figure 5.20 (a), (b), (c) and (d). It is seen that the constant errors are compensated in both networks.

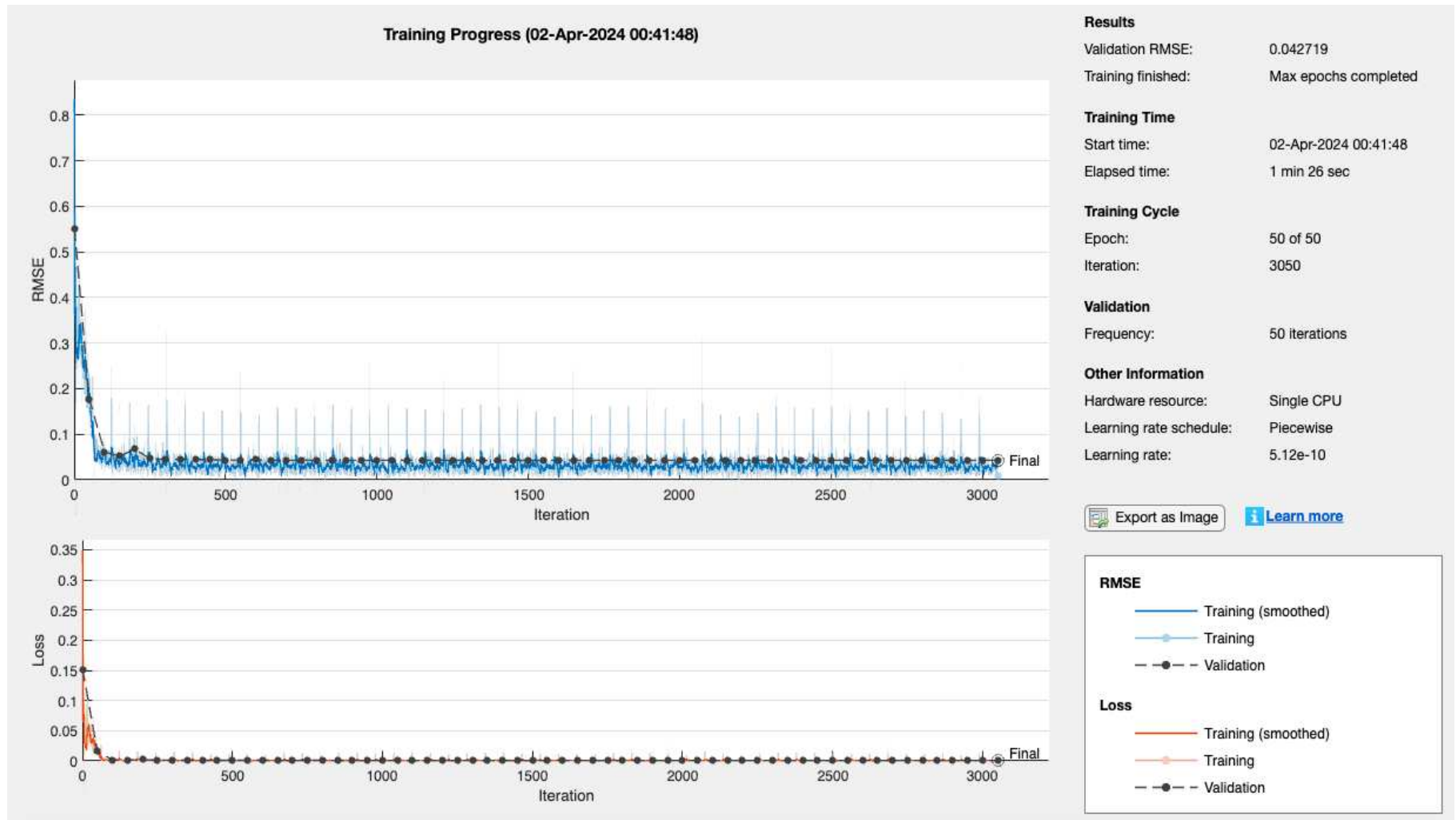


Figure 5.18: Training results on single CPU for gyroscope dynamic results with LSTM after first passing it through FNN

However, trying on another set on new unseen testing data, using the same networks with their individual weights and biases in each neuron, the performance changes.

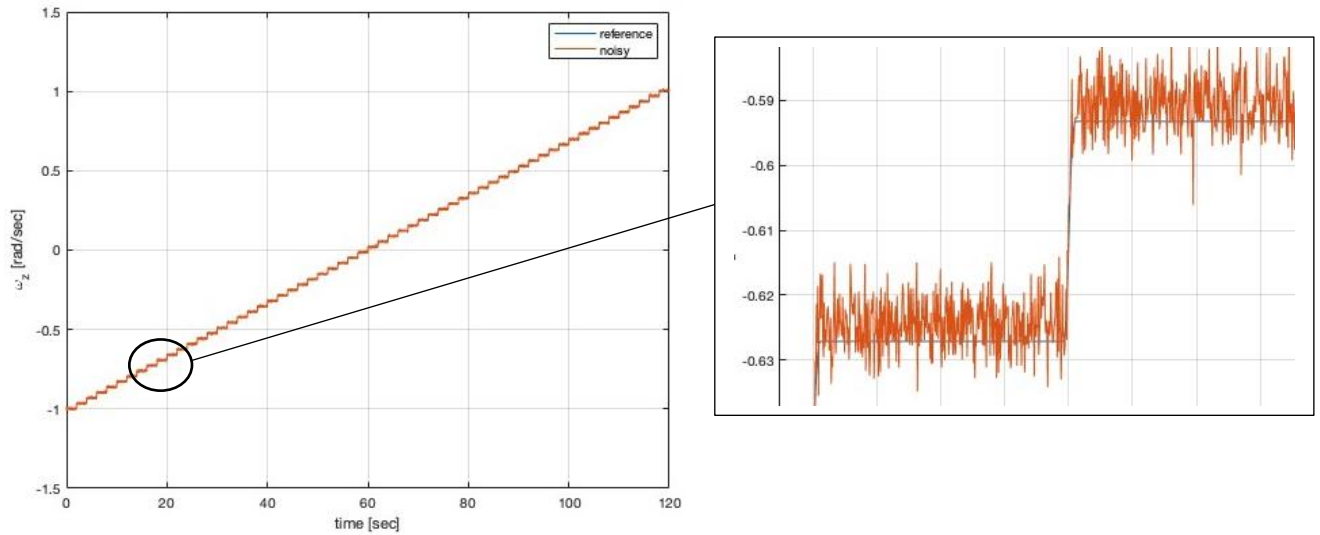
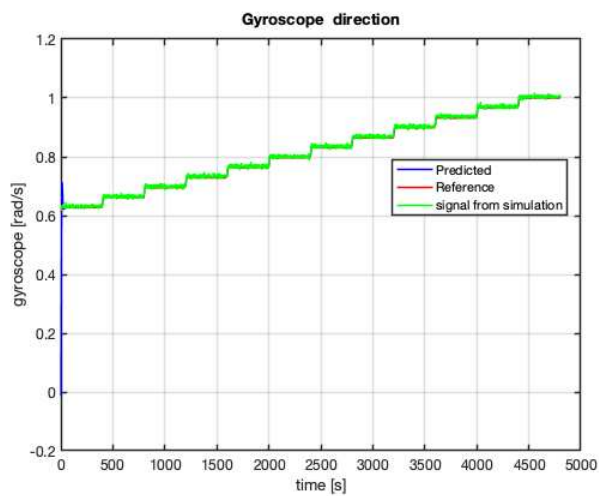
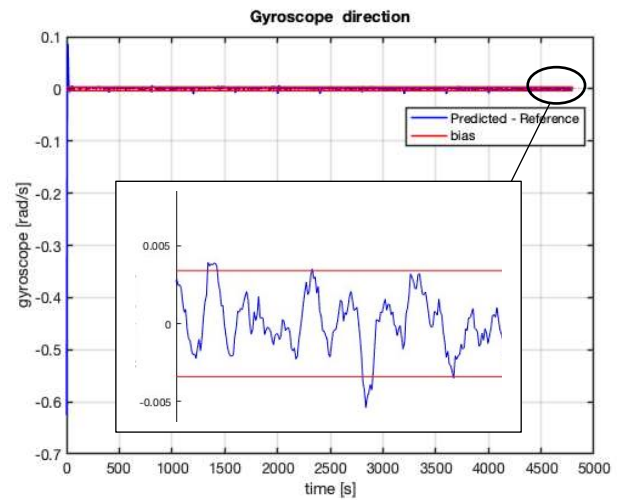


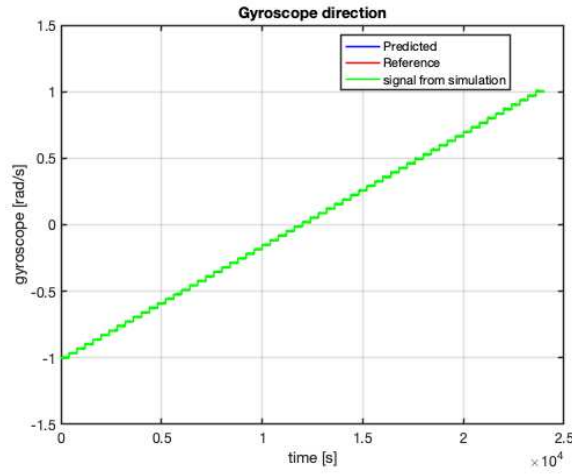
Figure 5.19: Step simulation data from -1 to +1 rad/s



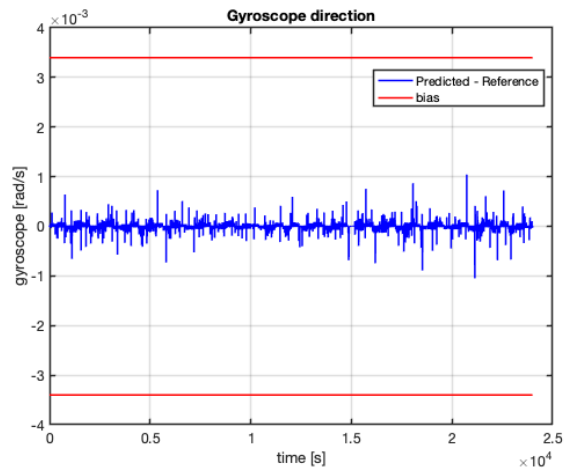
(a)



(b)



(c)

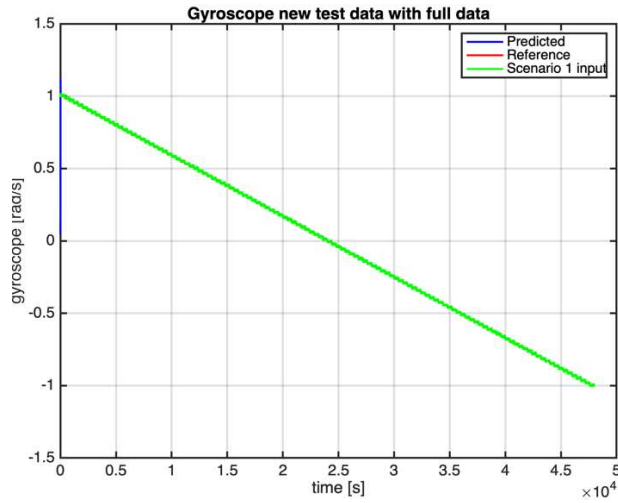


(d)

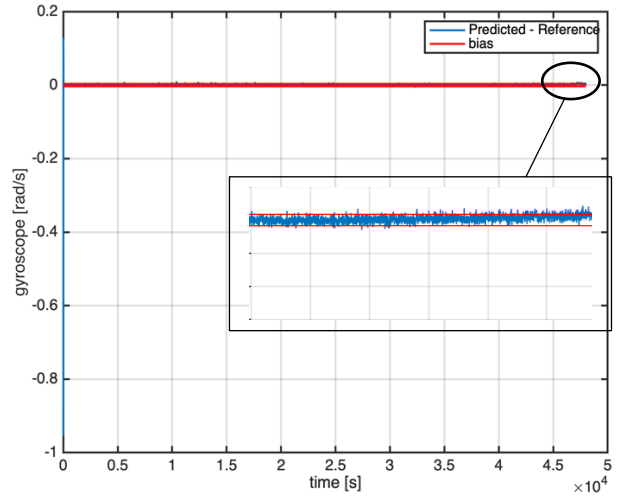
Figure 5.20 (a), (b), (c), (d): (a) and (b) LSTM ONLY static step results. (c) and (d) FNN ONLY static step results

First try of testing data is simulating a set of step static data running from +1 to -1 rad/s with halved step sizes, having more data points of 48000, shown in Figure 5.21. A second try of test data is simulating data from -1 to +2 rad/s, with the same number of steps as the original simulated data, having a total of 24000 data points, shown in Figure 5.22. Third try results, shown in Figure 5.23, is using dynamic data from Figure 5.9 to test on the network and see the performance. On all three, LSTM and FNN worked poorly on newly unseen data.

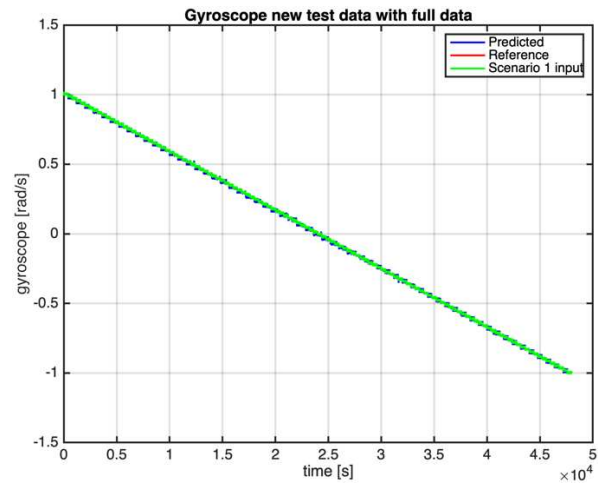
As these tries weren't successful, the next trial was averaging the static regions of constant angle positions (each step) as shown in Figure 5.24 and training the data in the networks. This was done to see if the network would learn the characteristics of the data and check the performance with new unseen step data from -1 to +1 rad/s. The results of this is shown in Figure 5.25, where the performance of the networks is poor.



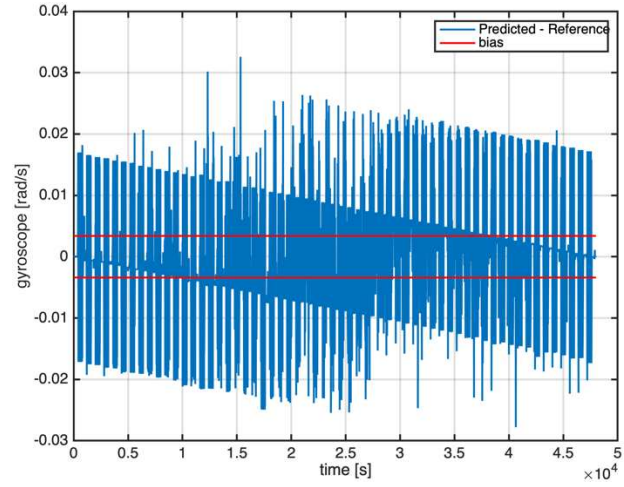
(a)



(b)

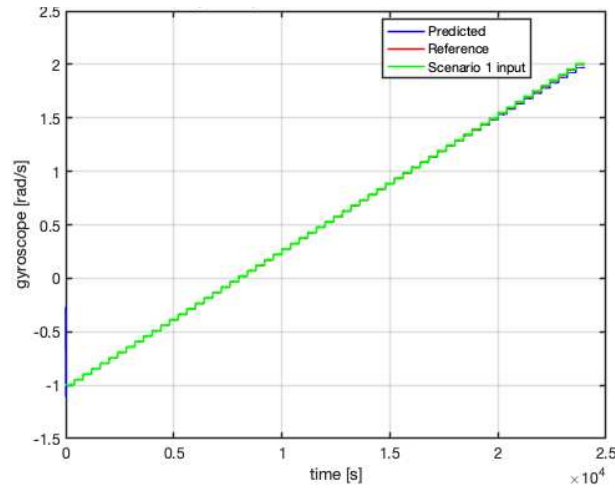


(c)

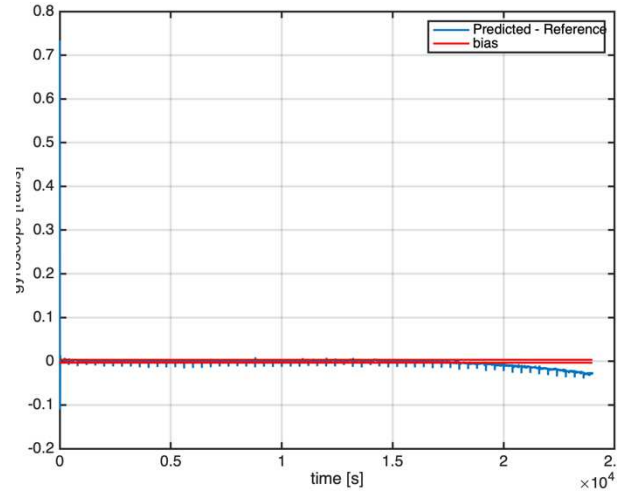


(d)

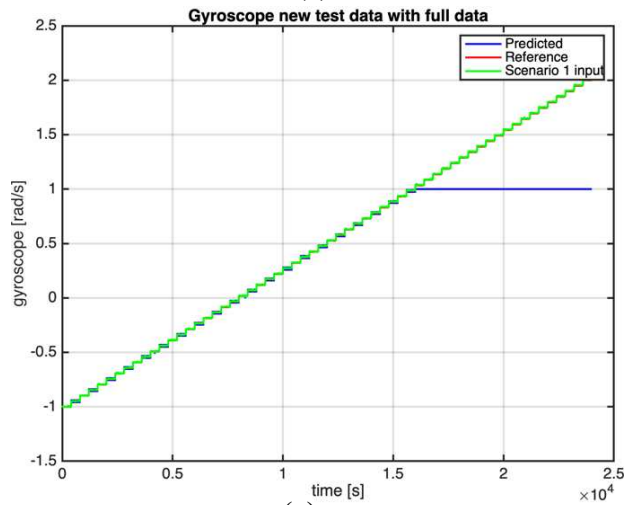
Figure 5.21 (a), (b), (c), (d): First scenario test with 48000 points from +1 to -1 rad/s. (a) and (b) LSTM ONLY static step results. (c) and (d) FNN ONLY static step results



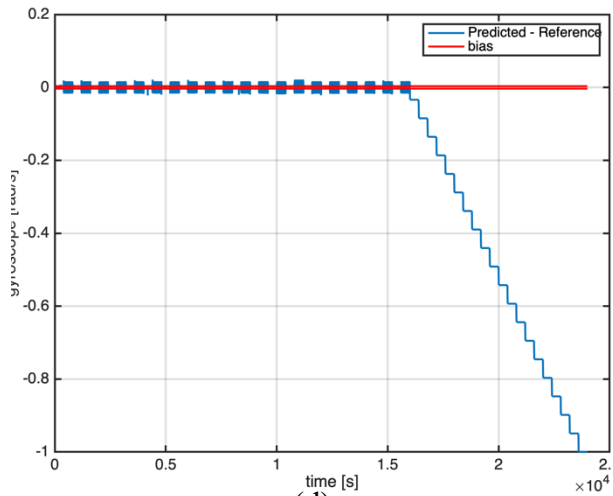
(a)



(b)

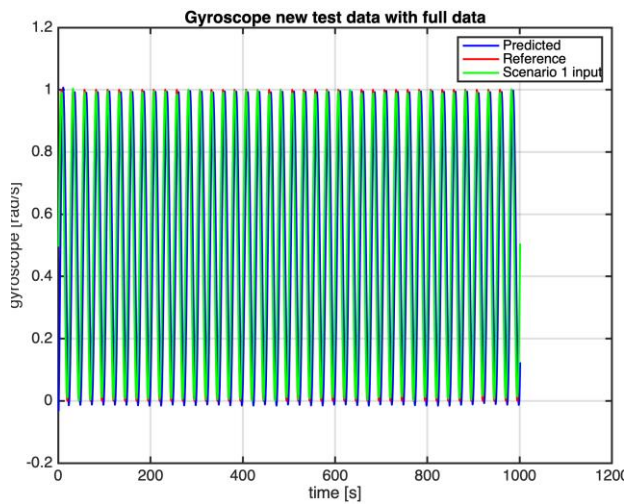


(c)

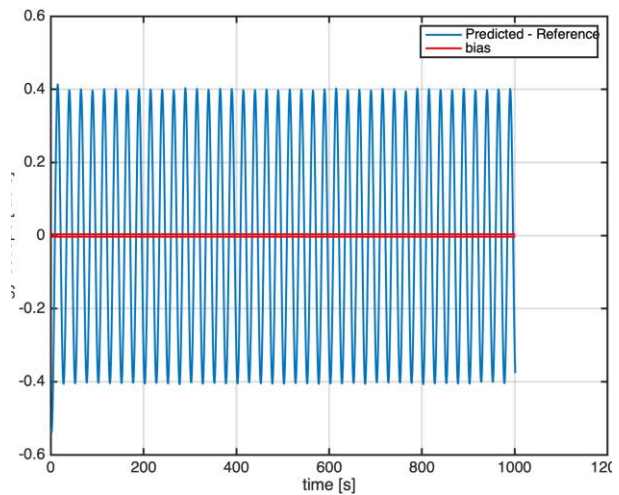


(d)

Figure 5.22 (a), (b), (c), (d): Second scenario test with 24000 points from -1 to +2 rad/s. (a) and (b) LSTM ONLY static step results. (c) and (d) FNN ONLY static step results



(a)



(b)

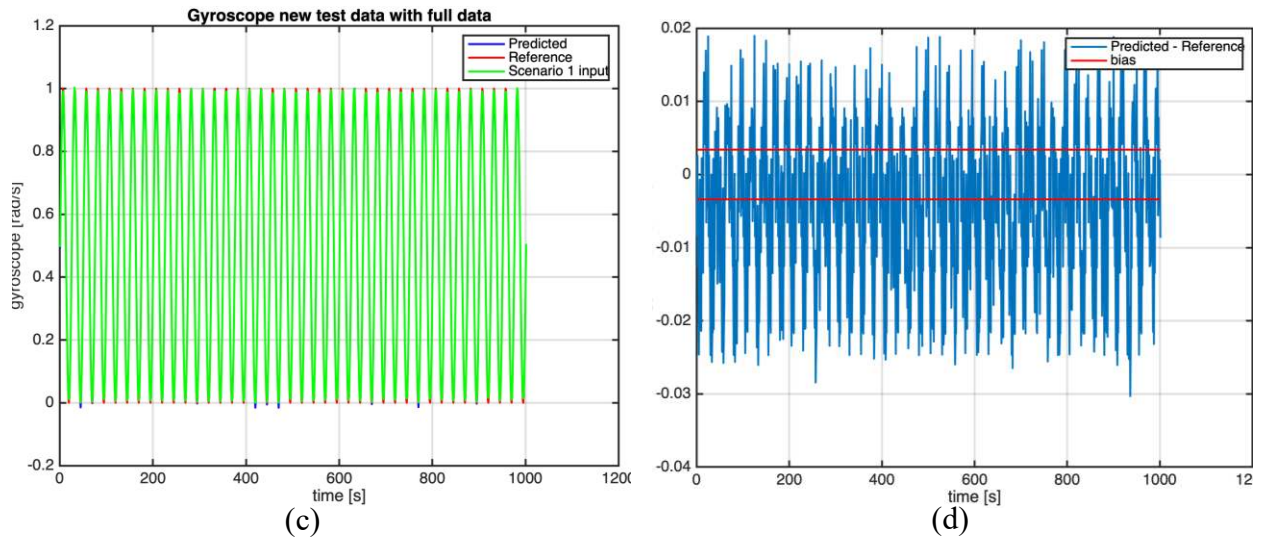


Figure 5.23 (a), (b), (c), (d): Third scenerio test with dynamic data of 1000 points. (a) and (b) LSTM ONLY dynamic results. (c) and (d) FNN ONLY dynamic results

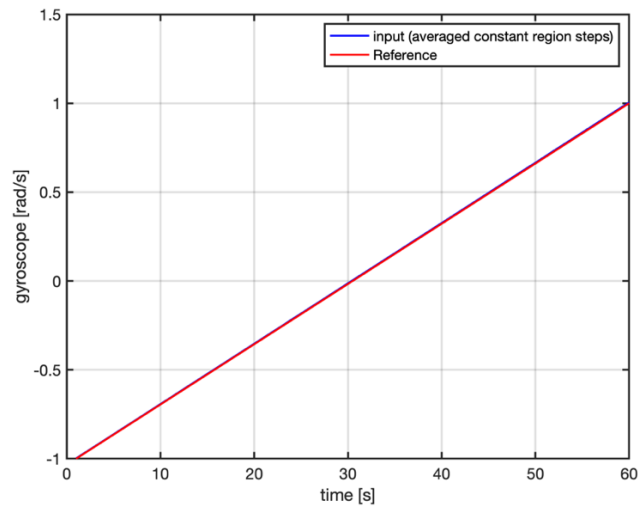


Figure 5.24: Averaged step static points used for training

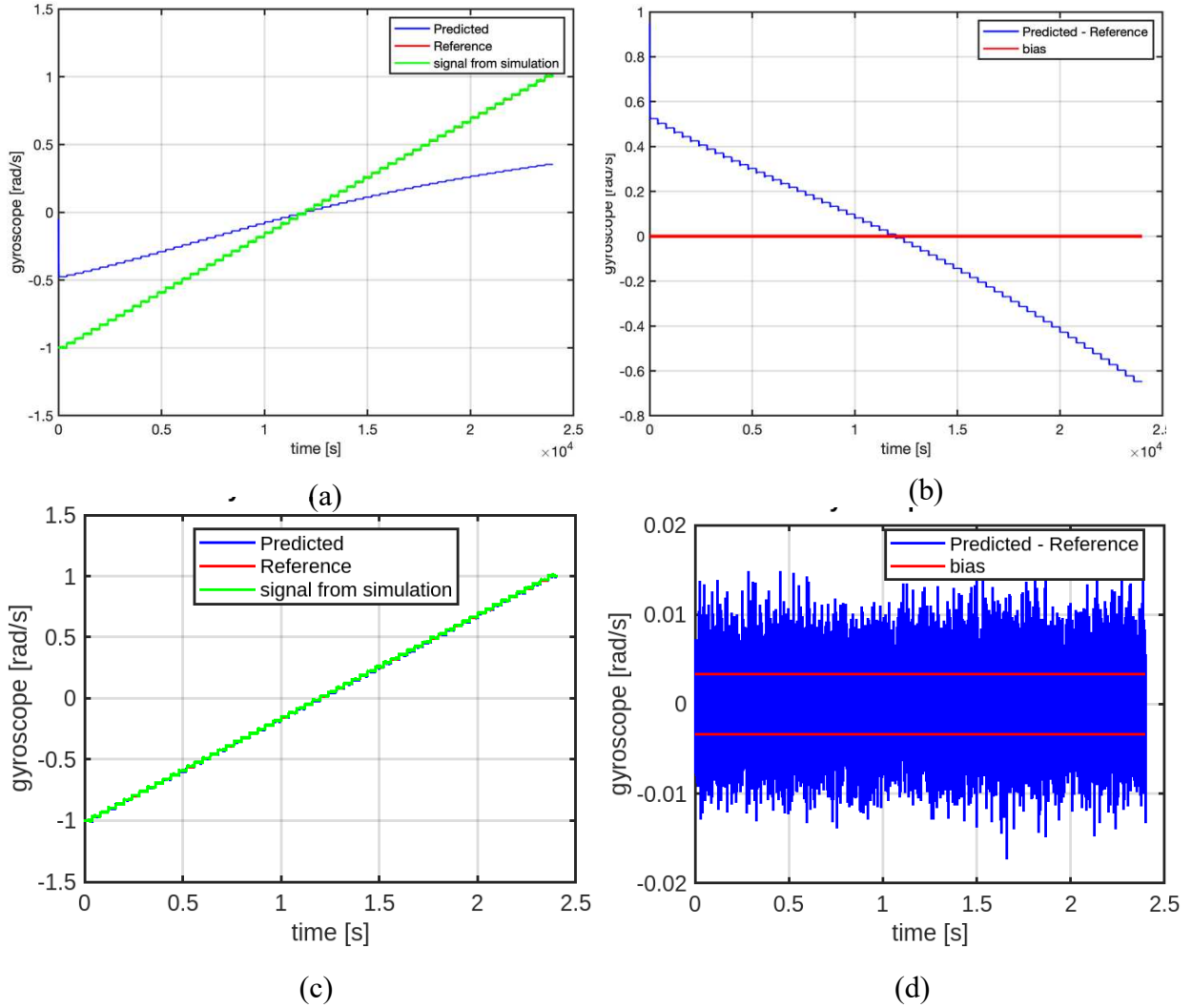


Figure 5.25 (a), (b), (c), (d): Forth scenerio test with static step data from -1 to +1 rad/s using averaged static step points as trained data. (a) and (b) LSTM ONLY (c) and (d) FNN ONLY

5.4 Real Dynamic Simulated Data Test

After seeing how LSTM and FNN perform on dynamic simulated test, the next trial was to use real data from the turntable. The Acuitas Motion Control (AMC) [3] software was used to run the turntable while the Movella MT [34] software was used to collect data from the Xsens IMU. As shown in Figure 5.26, the rotation is about the y-axis of the IMU, since the y-axis of gyroscope and z-axis of accelerometer have their ranges closer to the artificial rotation rate of 1 rad/s and the reference

earth's gravitational of 9.81 m/s^2 acceleration. About 65000 data points were collected, but the only problem was, time wasn't recorded, therefore, matching an error free signal with the raw dynamic data would be a challenge.

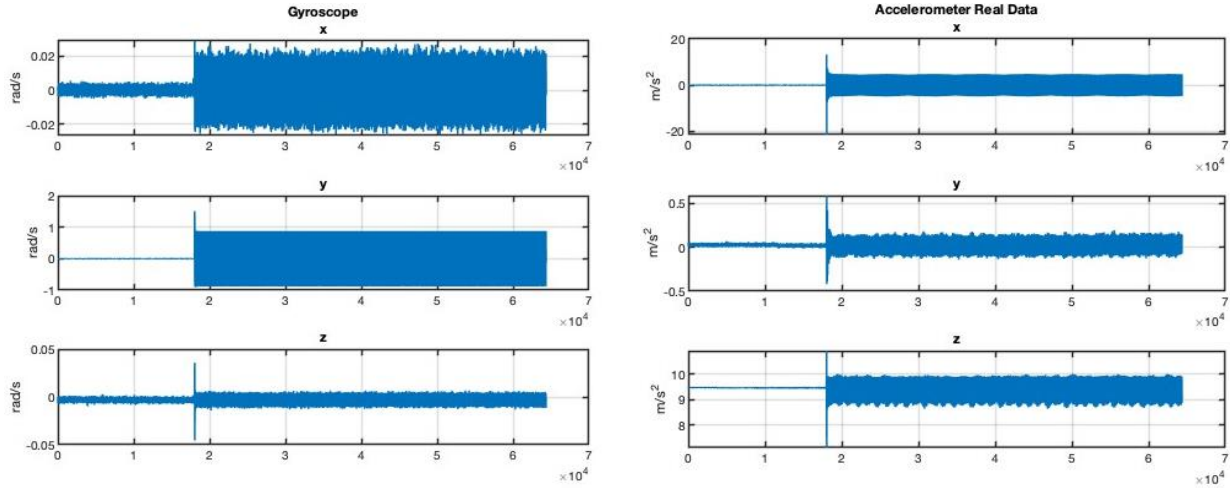


Figure 5.26: Real dynamic data collected from the turntable

5.5 Gyro-rate To Gyro-angle Test With Strapdown

As there was no consistency with the performance of LSTM and FNN on new unseen data, strapdown technique was the next alternative to use on the gyro-rates. So far, one keynote seen with neural networks is that they don't have the capability to mathematically integrate data correctly. They will only try to match the input data to the targets, and if the units are different, then ideally the result isn't correct. Now, to change from gyro-rate to gyro-angles, one needs to add an integration calculation on the data, in order to be able to have consistent units in training, validation and testing. The main purpose of calibration in gyroscopes is to have error free angles that would be useful in getting the roll, pitch and yaw movements. Rotation about the z-axis based on the configuration in Figure 2.4, having the x-axis be the direction of movement is called yaw angle while the rotation about the x and y axes are roll and pitch respectively.

The investigation now would focus on yaw angle [23] as achieving the accurate yaw angle estimation is mostly difficult for IMU. A successful test was done with simulation data and later, real data was collected to check on the performance.

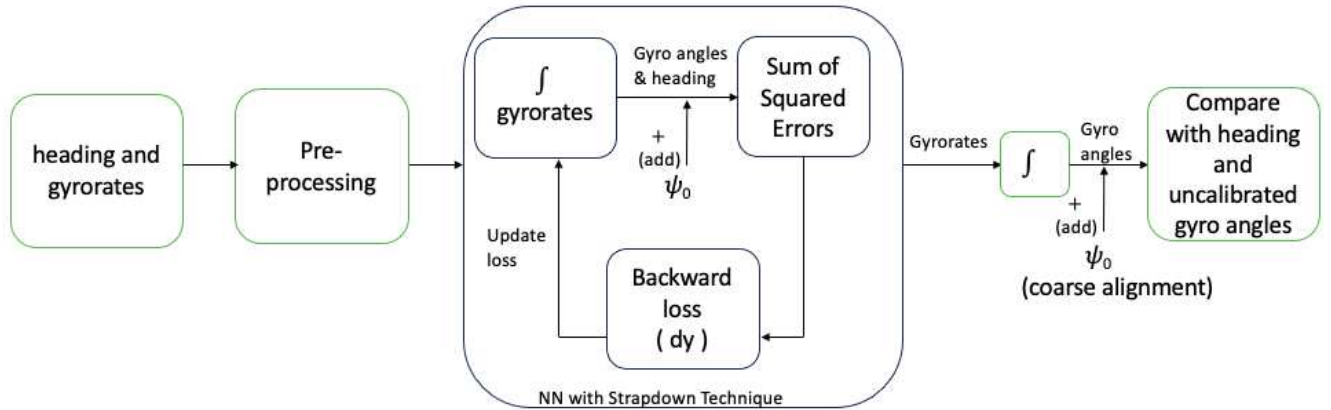


Figure 5.27: Strapdown Technique structure with NN

The structure of the strapdown technique is shown in Figure 5.27.

- i. Gyro-rates and heading angles are used as inputs to the network, as training and targets respectively. Here, pre-processing (normalization) is also important to better condition the data to equally contribute to the training process.
- ii. Inside the NN, the gyro-rates are first integrated to angles so that comparison is done with the targets. The reason is because the network cannot explicitly integrate the data to provide consistent units. Therefore, one needs to assist the network in implementing mathematical integration of the data.
- iii. As data in the loss computation in the network is already different based on the initial weights and biases of the network, coarse alignment is needed with the heading values inside the loss computation. Coarse alignment is aligning the IMU's coordinate frame with respect to an external reference frame. Here, a sum of squared errors (SSE) is used in the strapdown technique as shown in

Equation (5.3). The difference between MSE and SSE is the division with the number of responses, see Equations (5.1) and (5.2).

$$SSE = \sum_{i=1}^R (Y_i - T_i)^2 \quad (5.3)$$

where:

R : number of responses

T_i : target output

Y_i : network's prediction for response i

- iv. For the backpropagation, the first derivative of SSE is done as shown in Equation (5.4).

$$\frac{\delta L}{\delta Y_i} = 2 \cdot (Y_i - T_i) \quad (5.4)$$

- v. After minimizing the losses, the output of strapdown is still in rates. This is because, as the input of the network was gyro-rates, the output is also gyro-rates. Therefore, integration to convert them to gyro-angles is needed to easily compare with the targets. Coarse alignment is again necessary on the predicted values to improve the calibration accuracy.

5.5.1 Gyro-rate To Gyro-angle Test With Simulation using Strapdown Technique NN

As before, simulation data is first used to understand the capabilities of strapdown technique before collecting real data from the turntable, as shown in Figure 5.28.

The total number of data points simulated for both raw noisy data and error free data are 480,000 points.

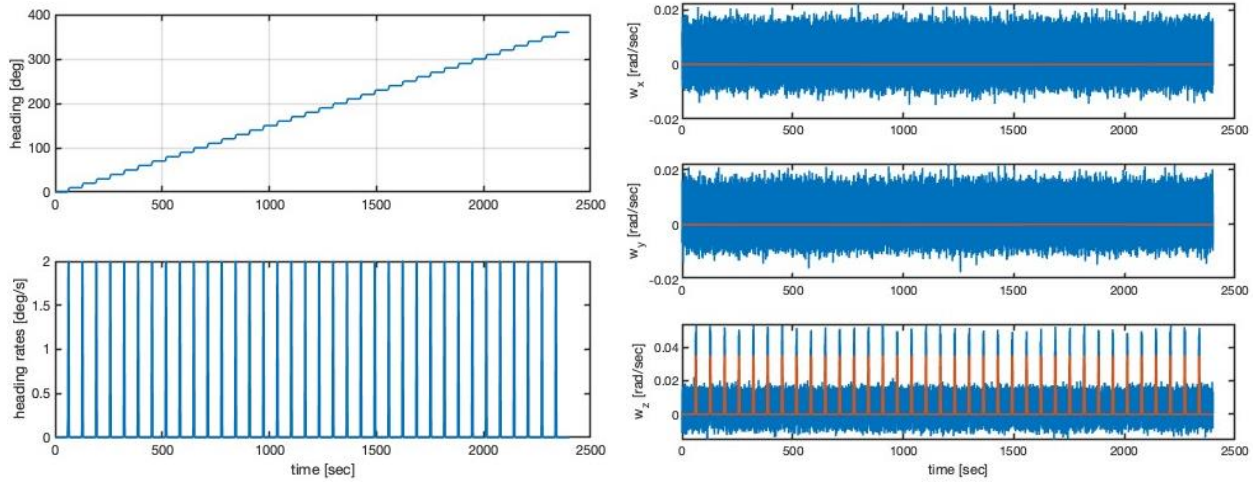


Figure 5.28 (a), (b): (a) Heading data collected as error free from 0 deg to 360 deg. (b) Noisy data collected as static positioning from 0 deg to 360 deg

The training options used in the strapdown technique are shown in Table 5.6, where the data is split to 80% training and 20% testing.

Table 5.6: Gyro-rate to gyro-angle training options

Training Options	Value
Strapdown Regression Layer	Sum of Squared Error (SSE)
MaxEpochs	225
MiniBatchSize	2^5
SequenceLength	6000
Learn Rate Drop Factor	0.9
Learn Rate Schedule	Piecewise
Initial Learn Rate	0.0003

Strapdown technique did well under the training options as shown in Figure 5.29. The difference between predicted and reference, after coarse alignment is close to zero. Given the performance of the network is promising, different tests with different dynamic simulated trajectories are conducted to understand if the network will perform well on new unseen data, and the summary is briefly shown in Table 5.7.

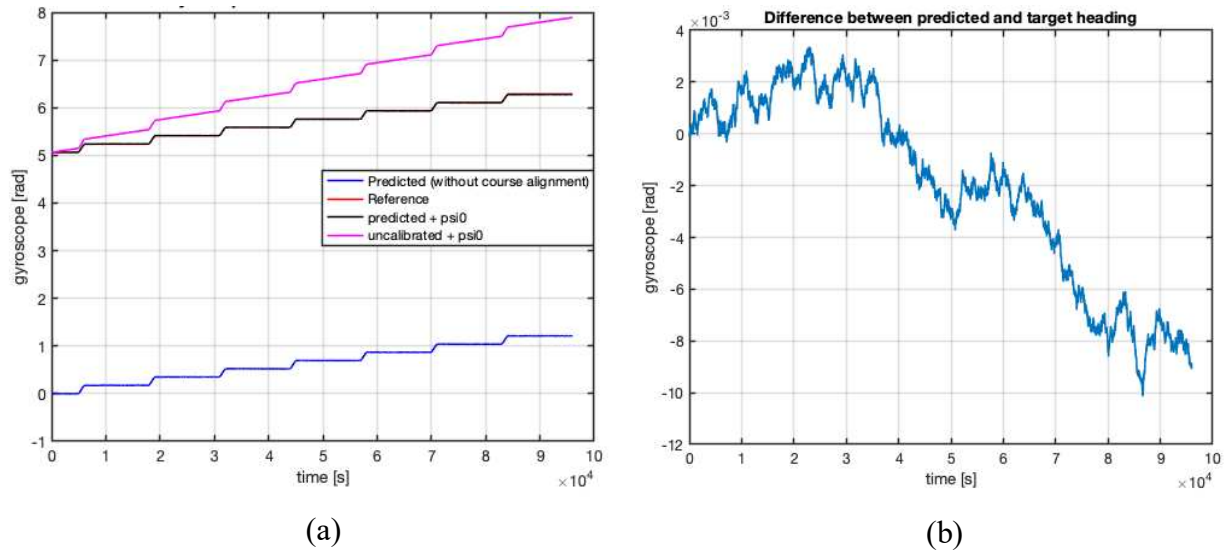


Figure 5.29 (a), (b): (a) Results from Strapdown technique, with testing on 20% of the unseen data. (b) Difference between predicted and reference gyro angles.

Table 5.7: Summary of performance of the strapdown technique with new simulated trajectories

New Trajectory	Size of Data	Figure Number	Performance Summary
1	19600	Figure 5.30	Very Good
2	434000	Figure 5.31	Good
3	382000	Figure 5.32	Good

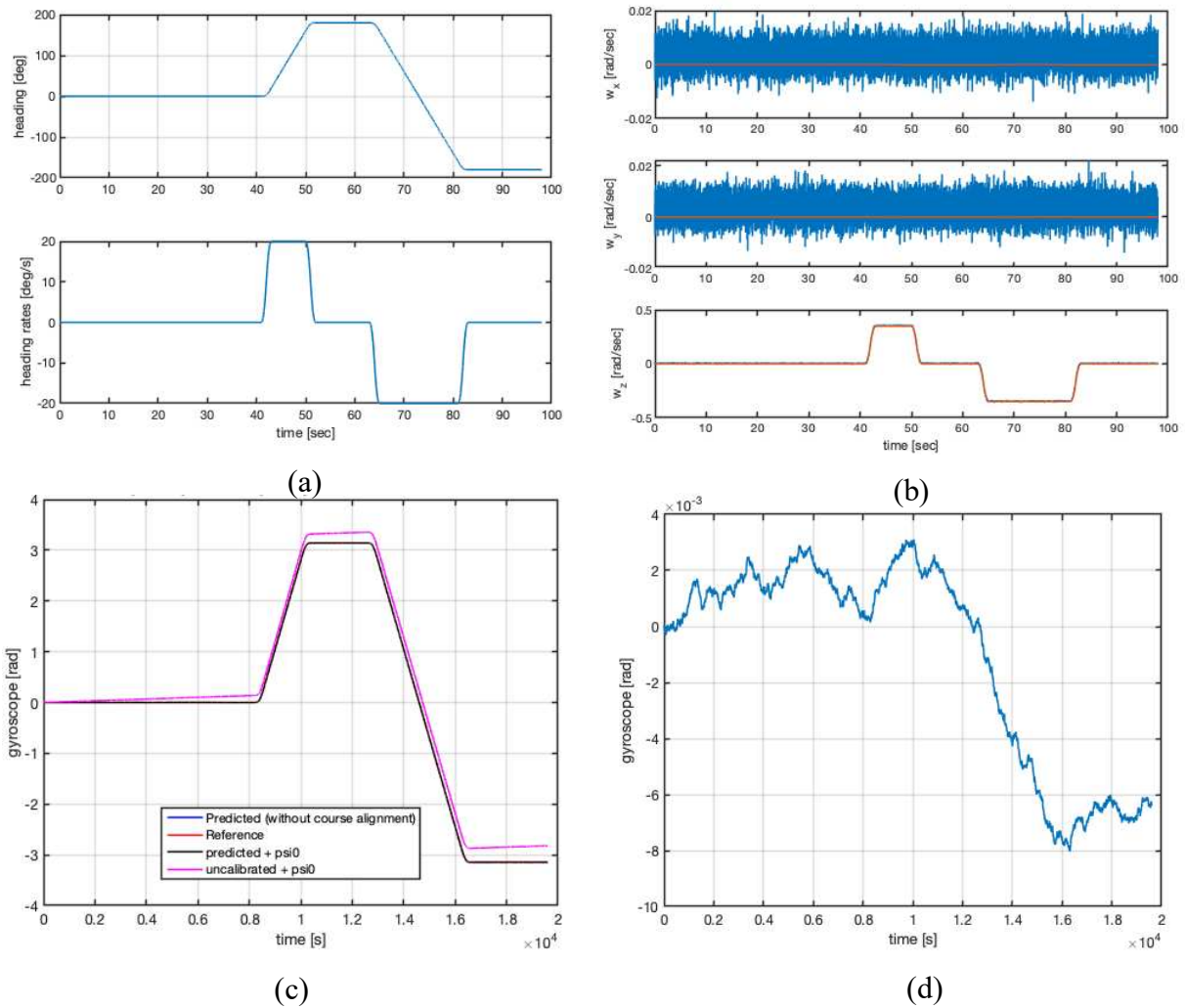
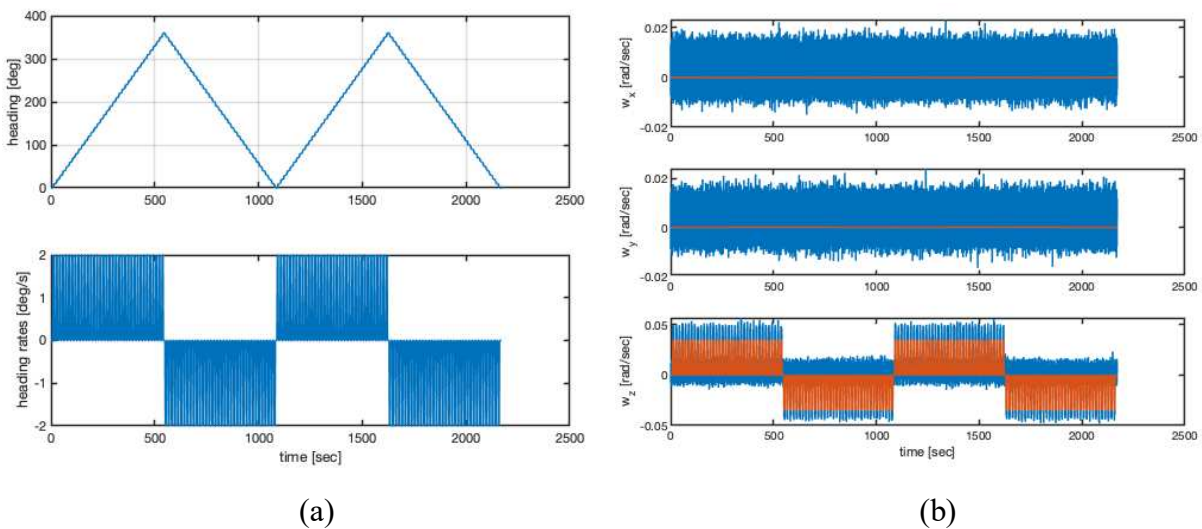
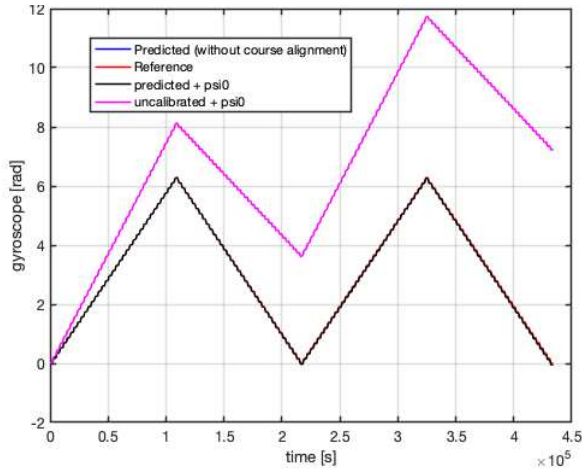
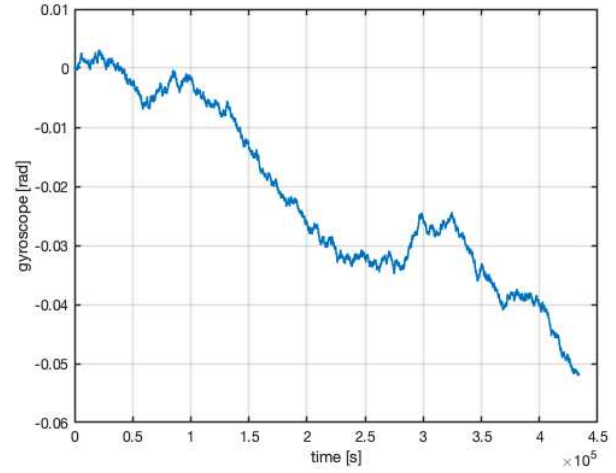


Figure 5.30 (a), (b), (c), (d): New gyrorate to gyroangle trajectory 1. (a) heading (target) in rad and rad/s. (b) gyrorate plots in x,y,z axes. (c) output from prediction. (d) difference between predicted and reference.



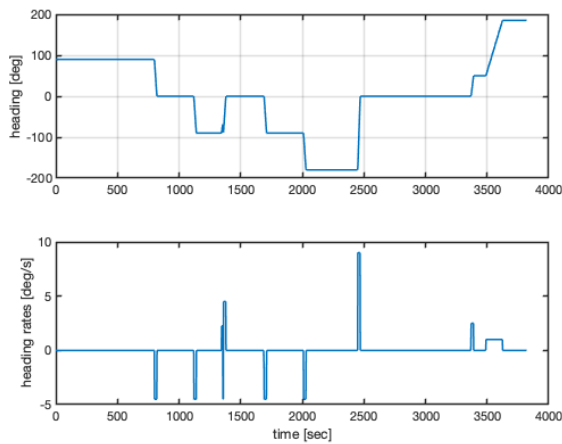


(c)

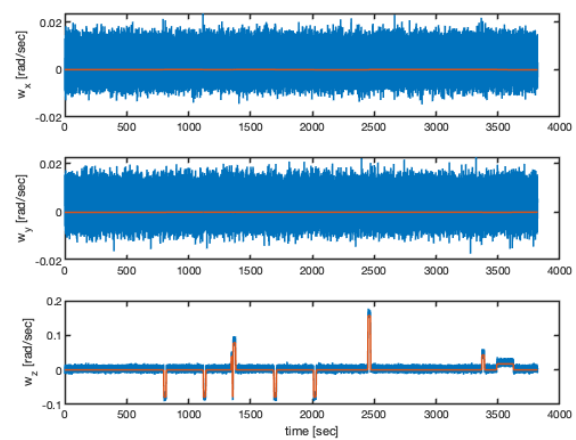


(d)

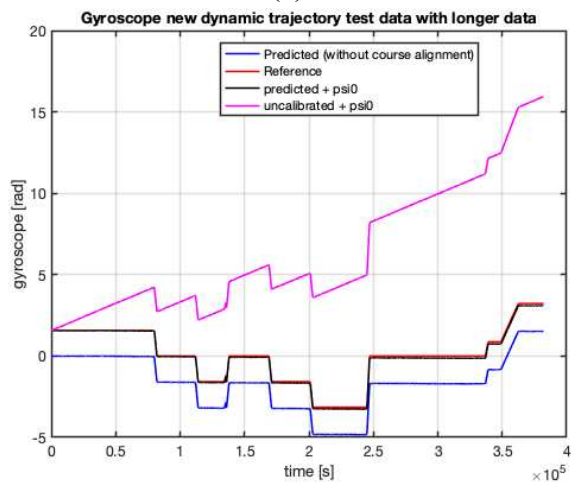
Figure 5.31 (a), (b), (c), (d): New gyrorate to gyroangle trajectory 2. (a) heading (target) in rad and rad/s. (b) gyrorate plots in x,y,z axes. (c) output from prediction. (d) difference between predicted and reference.



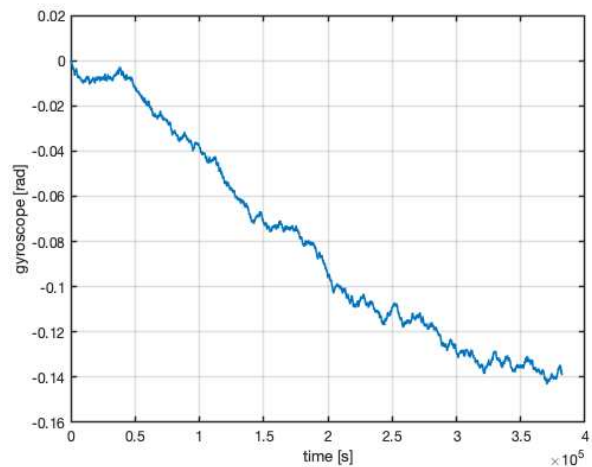
(a)



(b)



(c)



(d)

Figure 5.32 (a), (b), (c), (d): New gyrorate to gyroangle trajectory 3. (a) heading (target) in rad and rad/s. (b) gyrorate plots in x,y,z axes. (c) output from prediction. (d) difference between predicted and reference.

5.5.2 Gyro-rate To Gyro-angle Test With Real Data Using Strapdown Technique NN

The same procedure done in Section 5.5.1 is repeated with real data. The static data points of the IMU were collected in constant angle position, where the setup is done on the turntable, and it rotates from 0 deg to 360 deg, with an increment of 10 deg, maintaining each angle for roughly 120 seconds. The size of the data is 486956 used for both raw noisy gyro rate data and heading gyro angle data, shown in Figure 5.33, and split to 80% as training and 20% as testing.

As shown in Figure 5.33, the long vertical plotted lines represent the transition points from one angle to the next. Therefore, just as in Figure 5.27, both are put into the network for computation of the loss. Since we want a 1-1 relation, the transition points should not be considered while computing losses. This is because, the network wouldn't know that these regions aren't the constant angle regions and may give poor results. These are the blue vertical long lines in Figure 5.33.

The data was trained twice to improve the model's performance and address the changes in the data distribution as shown in the training options in Table 5.8. Retraining enables incremental learning, where the model is updated gradually over time as new data becomes available. This allows the model to stay up-to-date with the latest information without requiring periodic re-training from scratch. It also improves the performance since updating the model parameters with fine-tuning the training options makes the network capture complex relationships better within the data, leading to better generalization.

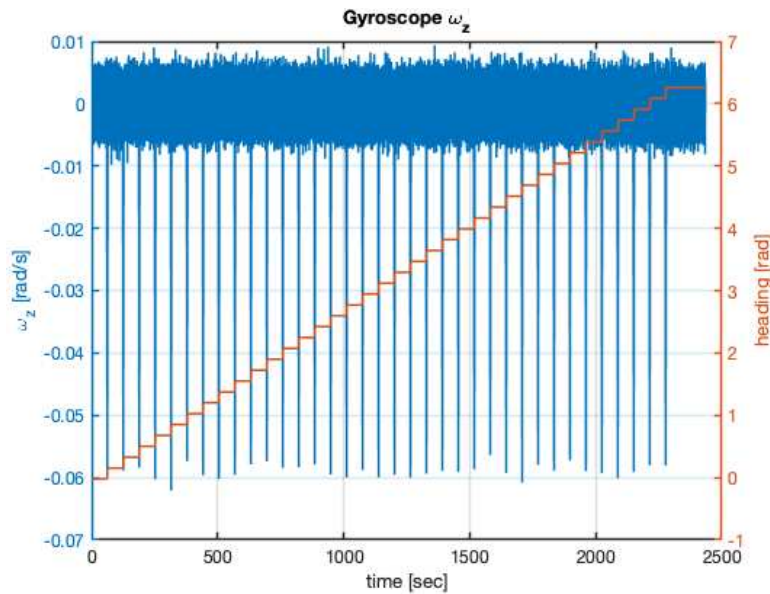


Figure 5.33: Raw step gyrorate data and heading (targets), 0 deg to 360 deg, each with a hold of 120 seconds

Table 5.8: Real data trained twice to improve the performance of updated data

Regression Layer	SSE
Training Options 1	
MaxEpochs	10^3
MiniBatchSize	4000
InitialLearnRate	0.0003
Training Options 2 - Retrain	
MaxEpochs	8000
MiniBatchSize	10^5
InitialLearnRate	0.065

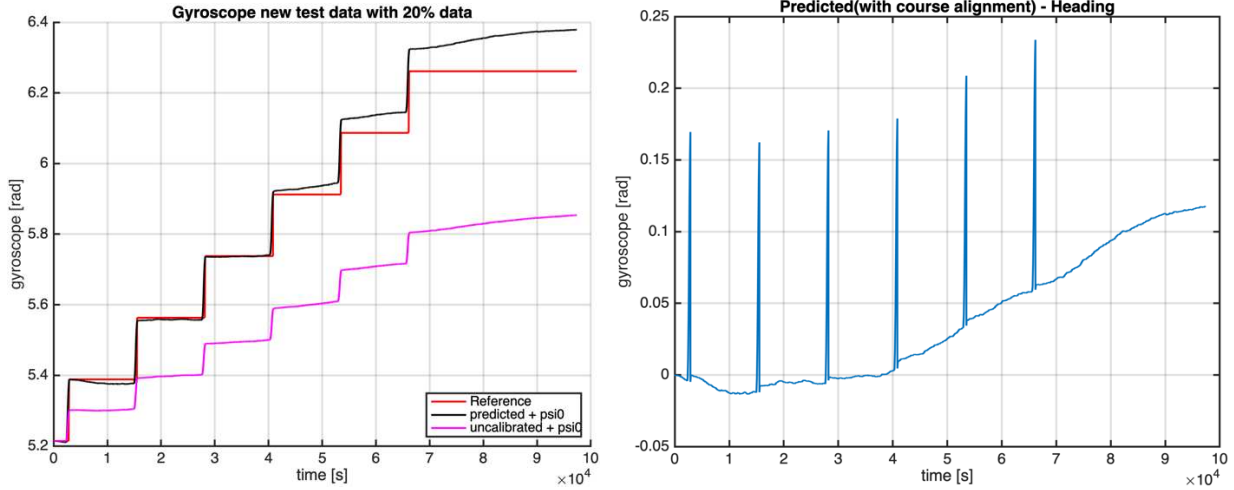


Figure 5.34 (a), (b): (a) Prediction results on real data. (b) Difference between predicted and heading (target) on real data

The output of the network is shown in Figure 5.34. This shows that the network understood the characteristics of the data and compensated the errors on the data. In Figure 5.34(b), the difference between the predicted and reference is close to zero, and it confirms the good performance of the network. In the same figure, the vertical lines represent the transition points to different constant angles from the test data, which are irrelevant as these weren't the regions considered during loss computation.

6. Conclusion and Outlook

In this thesis, the main goal was to compensate the constant errors from Xsens IMU, using neural networks. Other error sources are mentioned such as stochastic errors. Properties of these errors and how they affect the IMU readings are seen in detail and the classical and artificial neural networks calibration schemes are explained.

A series of simulation tests are done in MATLAB for various purposes such as algorithm development and testing, validation and verification, parameter tuning, performance evaluation, and education. Computer graphics became a core characteristic in running the networks, with respect to the libraries installed in the MATLAB software. This boils down to properties like parallel computing, especially as these tests use huge sets of complex data that needs to be broken down for easier computations.

While doing the tests, LSTM and FNN, performed well on the simulated data that was split into training and testing, however, ended up doing poorly on new trajectories of unseen simulated data. FNN for instance, forces an exact fit onto the data, and adjusting the number of neurons in the hidden layer only works well on the current data that has been split into testing. This is because the network seemed to already know its characteristics. However, the performance degrades with another set of new simulated data.

These tests became good foundation blocks for understanding how the real data would behave. Given a frequency of 200 Hz, the training options were adjusted to accommodate the high frequency for example by having more training epochs. At the end, strapdown technique is used to compensate for the constant errors. From the results, network performs well on both simulated and real data, having the difference of the predicted and target showing approximately the angle random walk, which cannot be compensated fully by the neural network.

Comparing with the interferometric fiber optic gyroscope (iFOG) [20], as the bandwidth sampling frequency is 100 Hz, less computational effort is needed to compensate the deterministic errors. The training options used such as maximum epochs, were less as compared to the low-cost MEMS IMU as its frequency was 200 Hz. This means that one needs to customize the training parameters depending on the type of IMU being calibrated.

Comparing classical and the modern neural network algorithms, neural networks is recommended for calibration of IMUs. This is because, they save on computational resources that could benefit the usage of IMUs.

Further investigation is though necessary, as drift errors from the stochastic errors can cause the result to move in the wrong direction but within the scope of this thesis, the outcome is okay as they were ignored. A trial with real dynamic data to compare the outcome with the static step data of constant angle position can also be done.

Bibliography

- [1] Abdel-Hafez, M. F. (2011). On the development of an inertial navigation error-budget system. *Journal of the Franklin Institute*, 348(1), 24-44.
- [2] Accelerometer and Gyroscopes Sensors: Operation, Sensing, and Applications. (2015, May 17). From Analog Devices: <https://www.analog.com/en/resources/technical-articles/accelerometer-and-gyroscopes-sensors-operation-sensing-and-applications.html>.
- [3] Acuitas Motion Control Software. (2023, April 18). From acuitas: <https://www.acuitas.ch/news/acuitas-motion-control-software>
- [4] Aggarwal, P., Syed, Z., Niu, X., & El-Sheimy, N. (2008). A Standard Testing and Calibration Procedure for Low Cost MEMS Inertial Sensors and Units. *THE JOURNAL OF NAVIGATION*(61), 323-336.
- [5] Bank, D., Koenigstein, N., & Giryes, R. (2023). Autoencoders. In *Machine Learning for Data Science Handbook* (pp. 353-374). Springer, Cham.
- [6] Barber, D., Mills, J., & Smith-Voysey, S. (2008). Geometric validation of a ground-based mobile laser scanning system. *ISPRS journal of photogrammetry and remote sensing*(63), 128-141.
- [7] Bebis, G., & Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE*, 13(4), 27-31.
- [8] Bochkati, M. A. (2014). Simulation and processing of IMU observations based on various sensor characteristics. Munich: Chair of Astronomical and Physical Geodesy - Technical University of Munich.
- [9] Bochkati, M., & Pany, T. (2021). Does the Android operating system provide what the MEMS-IMU manufacturers promise? *DGON Inertial Sensors and Systems - Symposium Gyro Technology* (pp. 1-17). Piscataway, NJ: IEEE.
- [10] DiPietro, R., & Hager, G. (2020). Deep Learning: RNNs and LSTM. In *Handbook of Medical Image Computing and Computer Assisted Intervention* (pp. 503-530). Baltimore, MD, United States: Elsevier Inc.
- [11] Donges, N. (2023, February 28). A Guide to Recurrent Neural Networks: Understanding RNN and LSTM Networks. From builtin: <https://builtin.com/data-science/recurrent-neural-networks-and-lstm>
- [12] El-Diasty, M., & Pagiatakis, S. (2008). Calibration and Stochastic Modelling of Inertial Navigation Sensor Errors. *Journal of Global Positioning Systems*, 7(2), 170-182.
- [13] Electronics-lab.com. (2017, July 10). From MEMS - A 22-Billion-Dollar-Worth Industry by 2018: <https://www.electronics-lab.com/mems-22-billion-dollar-industry-2018/>
- [14] Electronics, M. (2023, May). MTi-G-710-2A8G4 Datasheet. From Mouser Electronics: https://eu.mouser.com/ProductDetail/Movella-Xsens/MTI-G-710-2A8G4?qs=B6kkDfuK7%2FCmOmOoLHoXVg%3D%3D&_gl=1*tekcio*_ga*NDk1NDA3MD

M1LjE3MTA1NDE5OTQ.*_ga_15W4STQT4T*MTcxMDkzMzI5NC4yLjAuMTcxMDkzMzI5OS41NS4wLjA.*_ga_1KQLCYKRX3*MTcxMDkzMzI5NS4yLjAuMTcxMDkzMzI5OS

[15] Faisal, I. A., Purboyo, T. W., & Ansori, A. S. (2020). A Review of Accelerometer Sensor and Gyroscope Sensor in IMU Sensors on Motion Capture. *Journal of Engineering and Applied Sciences*, 15(3), 826-829.

[16] Gill, W. A., Howard, I., Mazhar, I., & McKee, K. (2022). A Review of MEMS Vibrating Gyroscopes and Their Reliability in Harsh Environments. *Sensors*, 22(19), 1-36.

[17] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). *Generative Adversarial Networks*. Montréal.

[18] Gyroscope: Learn Definition, Diagram, Parts, Working, Types & Uses. (2023, November 10). From testbook: <https://testbook.com/physics/gyroscope>

[19] IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Laser Gyros. (1996). IEEE Std 647-1995, 1-88.

[20] Kiema, E. (2024). Navigation Grade Inertial Measurement Unit (IMU) calibration using Neural Networks. [Unpublished manuscript].

[21] Kingma, D. P., & Ba, J. L. (2015). Adam: A method for Stochastic Optimization. 3rd International Conference for Learning Representations, (pp. 1-15). San Diego.

[22] Le, X.-H., Ho, H. V., Lee, G., & Jung, S. (2019). Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. *water*, 1387-1406.

[23] Li, R., Fu, C., Yi, W., & Yi, X. (2022). Calib-Net: Calibrating the Low-Cost IMU via Deep Convolutional Neural Network. *Frontiers in Robotics and AI*, 1-8.

[24] Lindermann, B., Müller, T., Vietz, H., Jazdi, N., & Weyrich, M. (2020). A survey on long short-term memory networks for time series prediction. 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME '20 (pp. 650-655). ELSEVIER.

[25] Liu, S. Q., & Zhu, R. (2016). System Error Compensation Methodology Based on a Neural Network for a Micromachined Inertial Measurement Unit. *Sensors*, 16(2), 1-14.

[26] McCaffrey, J. (2015, May 13). Neural network Train-Validate-Test Stopping. From Visual Studio Magazine: <https://visualstudiomagazine.com/articles/2015/05/01/train-validate-test-stopping.aspx>

[27] Moré, J. J. (2006). The Levenberg-Marquardt algorithm: Implementation and theory. *Numerical Analysis* (pp. 105-116). SPRINGER.

[28] Niu, X., Wang, Q., Li, Y., & Liu, J. (2015). Using Inertial Sensors in Smartphones for Curriculum Experiments of Inertial Navigation Technology. *Education Sciences*, 26-46.

[29] Radi, A., Nassar, S., & El-Sheimy, N. (2018). Stochastic Error Modeling of Smartphone Inertial Sensors for Navigation in Varying Dynamic Conditions. *Gyroscopy and Navigation*, 9(1), 76-95.

- [30] Root Mean Square Error MSE. (2024). Retrieved from C3.ai: <https://c3.ai/glossary/data-science/root-mean-square-error-rmse/>
- [31] Sarker, I. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. SN Computer Science, 2(420), 1-20.
- [32] Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. International Journal of Advanced Research in Artificial Intelligence, 2(2), 33-38.
- [33] Sazli, M. (2006). A bried review of feed-forward neural networks. Communications Faculty of Science University of Ankara, 50, 11.17.
- [34] Software & Documentation. (2024). From Movella: <https://www.movella.com/support/software-documentation>
- [35] Wang, J., Lou, W., Liu, W., & Liu, P. (2019). Calibration of MEMS Based Inertial Measurement Unit Using Long Short-Term Memory Network. 14th Annual IEEE International Conference on Nano/Micro Engineered and Molecular Systems (pp. 118-121). Bangkok, Thailand: IEEE.