ANGULAR MOTION ESTIMATION AND ITS APPLICATION TO THE STABILIZATION OF A BALLBOT

by

FIRAT YAVUZ

Submitted to
the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

SABANCI UNIVERSITY

August 2016

ANGULAR MOTION ESTIMATION AND ITS APPLICATION TO THE STABILIZATION OF A BALLBOT

APPROVED BY

Prof. Dr. Mustafa Ünel
(Thesis Advisor) .....

Assoc. Prof. Dr. Kemalettin Erbatur .....

Assoc. Prof. Dr. Şeref Naci Engin .....

DATE OF APPROVAL: ...08.08.2016.....

ABSTRACT

ANGULAR MOTION ESTIMATION AND ITS APPLICATION TO THE STABILIZATION OF A BALLBOT

FIRAT YAVUZ

Mechatronics Engineering, M.Sc. Thesis, August 2016

Thesis Advisor: Prof. Dr. Mustafa Ünel

Keywords: angular motion estimation, Euler angles, angular rates, IMU, sensor fusion, Kalman filter, ballbot, stabilization, acceleration control, balancing control

Reliable angular motion estimation have received significant attention in recent years due to remarkable advances in sensor technologies and related requirements in many control applications including stabilization of robotic platforms. The goal of the stabilization control is to maintain the desired orientation by rejecting external disturbances.

In this thesis, a novel master-slave Kalman filter is proposed where an extended Kalman filter (EKF) and a classical Kalman filter (KF) are integrated in a master-slave configuration to estimate reliable angular motion signals including Euler angles, rates and accelerations by fusing measurements of an inertial measurement unit (IMU). Estimated angular motion signals are used as feedback in both balancing and position control of a ballbot, which is a single spherical wheeled mobile platform driven with three omniwheels. An experimental ballbot system is designed and constructed for implementing estimation and control algorithms. Furthermore, nonlinear dynamical model of the ballbot is derived using Euler-Lagrange formulation, and balancing and position controllers are designed. Robustness of the controllers is achieved by employing cascaded control loops enhanced with acceleration feedback (AFB) to provide higher stiffness to the system. Effectiveness of the proposed fusion and control algorithms are validated by several simulations and experiments where performance comparison with a conventional PD controller is also made.

ÖZET

AÇISAL HAREKET KESTİRİMİ VE BİR BALLBOTUN STABİLİZASYONUNDA KULLANIMI

FIRAT YAVUZ

Mekatronik Mühendisliği, Yüksek Lisans Tezi, Ağustos 2016

Tez Danışmanı: Prof. Dr. Mustafa Ünel

Anahtar Kelimeler: açısal hareket kestirimi, Euler açıları, açısal hızlar, IMU, sensör füzyonu, Kalman filtresi, ballbot, stabilizasyon, ivme kontrolü, dengeleme kontrolü

Güvenilir açısal hareket kestirimi, sensör teknolojilerindeki dikkate değer gelişmeler ve robotik platformların stabilizasyonu gibi birçok kontrol uygulamasındaki ihtiyaçlardan dolayı son yıllarda ciddi ilgi toplamıştır.

Bu tezde, ataletsel ölçüm birimi (AÖB) ölçümlerinin füzyonu ile Euler açılarını, hızlarını ve ivmelerini içeren güvenilir açısal hareket kesitirimi için, genişletilmiş bir Kalman filtresi (GKF) ve klasik bir Kalman filtresinin (KF) bir usta-yamak biçiminde bütünleştirildiği özgün bir usta-yamak Kalman filtresi sunulmuştur. Kestirilen açısal hareket sinyalleri, üç adet her yöne hareket edebilen çarklar ile sürülen tek bir küresel tekerlekli seyyar bir platform olan ballbotun dengeleme ve konumlama kontrolünde geribildirim olarak kullanılmıştır. Kestirim ve kontrol algoritmalarını uygulamak için deneysel bir ballbot sistemi tasarlanıp üretilmiştir. Ayrıca, ballbotun doğrusal olmayan dinamik modeli Euler-Lagrange formülasyonu kullanılarak türetilmiştir ve dengeleme ve konumlama kontrolcüleri tasarlanmıştır. Kontrolcülerin gürbüzlüğü, sisteme yüksek sertlik sağlamak için ivme geribildirimi ile güçlendirilmiş iç içe geçmiş kontrol döngülerinin kullanımı ile sağlanmıştır. Önerilen füzyon ve kontrol algoritmalarının etkinliği benzetim ve deneylerle onaylanmış olup, konvensiyonel PD kontrolörü ile performans karşılaştırması da yapılmıştır.

$\ll$ *Aileme ve dostlarıma* $\gg$

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor Prof. Dr. Mustafa Ünel for his guidance, constructive suggestions and support throughout this research. I am grateful to him for helping me to broaden my horizon, not only in academic field but also in personal manner.

I would gratefully thank Assoc. Prof. Dr. Kemalettin Erbatur and Assoc. Prof. Dr. Şeref Naci Engin for their feedbacks and spending their valuable time to serve as my jurors.

I would like to acknowledge the financial support provided by The Scientific and Technological Research Council of Turkey (TÜBİTAK) through BİDEB 2228-A scholarship.

I would like to thank all the members of Control, Vision and Robotics (CVR) research group, Sanem Evren Han, Gökhan Alcan and Hammad Zaki for their pleasant team-work and friendship. I am also grateful to all of the other members of the Mechatronics Laboratory, especially to Yusuf Mert Şentürk, Wisdom Chukwunwike Agboh, Hammad Munawar, Gökay Çoruhlu, Mehrullah Soomro, Osman Saygıner, Sezen Yağmur Günay, Aykut Özgün Önol and Shoaib Imtiyaz Shaikh. They all made my journey of master education pleasurable and joyful. Also many thanks to M. İlker Sevgen and Cüneyt Genç for their positive attitude and companionship. Furthermore, I am highly indebted to my bicycle RR 8.1. Every day, it accompanied me on my trip to the university without minding.

I would like to thank Bedriye Salman and Hüseyin Hışıl for all their support in all my choices. They have always encouraged me to pursue my dreams and follow my heart. Through all challenges, they have been there for me.

Finally, I would like to thank my precious ones, my parents Asya and Munir Yavuz and my brother Ömer Yavuz for all their love, caring and never ending support in every instant of my life.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The attitude estimation of rigid body systems from sensor measurements has been popular over the years and been used in many applications such as unmanned vehicle control, platform stabilization, human motion tracking and underwater navigation. Modern attitude heading reference systems (AHRS) are combination of strapped-down multi-axis inertial sensors and microprocessors to provide an accurate measurement of the orientation of a vehicle with respect to inertial frame by using estimation algorithms to compute the attitude from several sensor measurements. The recent developments in integrated circuits and micro-electro-mechanical systems (MEMS) technology has facilitated production of low cost, light weight and highly accurate inertial measurement units (IMU) and processors to be used in the development of small and reliable AHRS devices for both research and industrial applications. A typical IMU consists of a triad of orthogonal gyroscopes, accelerometers and magnetometers on all 3-axis, that measure rotation rates, accelerations and earth's magnetic field respectively. Measuring the attitude angles by integrating angular rates measured by gyroscope is not feasible due to the drift problem resulted from the accumulation of the bias errors. Since the individual use of these

sensors is not sufficient for reliable attitude estimation, usually a sensor fusion algorithm is employed to combine individual sensor measurements to achieve estimation results that are far better than those of a single sensor.

In recent years, inertial sensors have gained a great interest by the robotics society. These sensors are key components for a robot that operates in its environment where the robot must have accurate information about its current state such as position, velocity, distance, altitude, attitude. In most type of robots, especially in autonomous mobile robots, sensors empower the robot to work self-sufficiently, and robustly. One of the fundamental necessities of an autonomous mobile robot is to have good mobility and maneuverability in order to accomplish its task. The fact is that wheels design plays an important role for these capabilities. Recent research showed that more groups have been interested in improving the autonomous navigation capability of mobile robotic systems, especially for omnidirectional mobile robots. Research on omnidirectional and self-balancing mobile platforms has been quite active in robotics and control communities in the last decade due to their ability to freely move in all directions. There are two main types of omnidirectional robots, the conventional wheeled structure and the special wheeled structure.

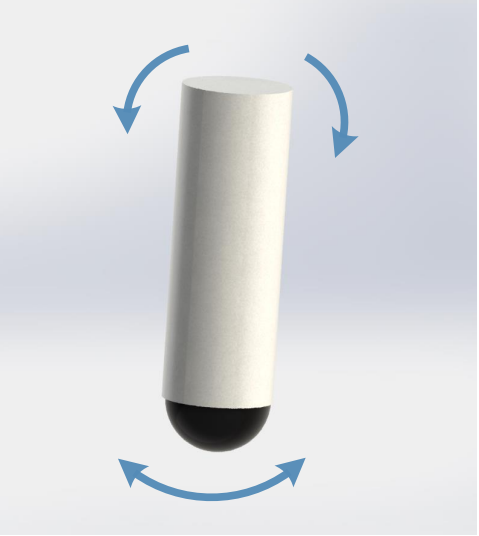

FIGURE 1.1: General representation of a ballbot.

Self-balancing mobile platforms with single spherical wheel, also known as ballbots, are suitable example of special wheeled systems. In such systems, a body balances and moves on a single ball which used as a replacement of conventional wheels to achieve omnidirectional motion. Ballbots can be described as an inverted pendulum mounted on a spherical wheel (see Figure 1.1). The spherical wheel is mostly manipulated through rollers or wheels attached to the actuators on the body.

In summary, a ballbot can move spontaneously in any direction and must actively balance on a single point of contact with the ground which reduces energy for motion due to less friction but makes system to be inherently unstable. The developments on ballbot broke out in a question: "How to preserve stability of the robot despite all disturbances?". A key component of this task is that the ballbot must have accurate information about its current orientation which means that it should use sufficient variety of sensors and fusion algorithms for extracting meaningful feedback from those sensor measurements.

1.1 Motivation

The success of stabilization control largely depends on reliable angular motion feedback. In the literature, there are several research efforts done for robust stabilization problem using angular motion signal, in particular acceleration feedback (AFB) signals. The primary objective of acceleration feedback control is rejecting disturbance which manifests itself directly in the acceleration signal. However, the essence of the task is obtaining a reliable angular motion feedback which is still a challenging issue. Nowadays, obtaining reliable orientation angle is easily walkable. On the other hand, obtaining angular velocity and acceleration signal by taking time derivatives of measured position signals is not feasible to due to sensor noise amplification problems. Angular accelerometers directly measure angular accelerations; however they are costly and produced by only a few manufacturers. Therefore, it is important to develop novel angular motion estimation techniques which also estimates angular velocities and accelerations.

In order to show the usability of estimated angular motion, ballbots are suitable platforms due to their inherently unstable and profoundly nonlinear under-actuated structure. After 2006, interest and research studies about ballbot platforms rose rapidly and became an important constituent in both academic and commercial applications. Although much progress has been made on the development on such platforms, controlling motion while maintaining balance is a challenging task because their under-actuated structure has fewer control inputs than their degrees of freedom. Therefore, it is crucial to develop robust approaches for controlling the ballbot.

In the light of these discussions, the main motivation of this thesis is to estimate reliable angular position, velocity and accelerations using newly formulated sensor fusion algorithm, hereby to provide a solution to the ballbot stabilization problem for improving the disturbance rejection properties and therefore the overall stabilization accuracy by utilizing estimated values. In doing so, demonstration of the

effectiveness of the proposed fusion and control methods in both simulation and experimental environments constitutes an important part of the motivation. The ballbot model used in this work is motivated and inspired in part by the early work of three omniwheeled ballbot designs for better control performance and reduced friction [1, 2].

1.2 Contributions of the Thesis

Contributions of the thesis can be summarized as follows:

- A new master-slave type Kalman filter which employs both an extended Kalman filter (EKF) and a classical Kalman filter (KF) is developed for estimating reliable angular motion including Euler angles, Euler rates and Euler accelerations.
- An experimental ballbot system is designed and constructed where sensor fusion and control algorithms are assessed on the prototype ballbot.
- Cascaded position, velocity and current control loops enhanced with angular acceleration feedback (AFB) signals are developed to provide robust performance against the disturbances acting on the ballbot.
- 3D dynamic model of the designed ballbot is derived by considering the highly nonlinear couplings and uncertainties in order to implement proposed control algorithms in simulation environment.
- A high fidelity simulator is created which consists of the accurate models for the ballbot system, controllers, inertial sensors (gyroscopes, accelerometers and magnetometers), motion estimator and external disturbances.

1.3 Outline of The Thesis

Chapter 2 presents the literature survey and theoretical background for the angular motion estimation techniques and ballbot platforms as well as a summary of existing models and control methods. **Chapter 3** details the derivation of the conventional attitude and heading reference system and the proposed master-slave Kalman filter algorithm, also the sensor models. **Chapter 4** starts with the explanation of design and construction of a ballbot platform and details the other components of the experimental setup. **Chapter 5** details the derivation of dynamic model of a ballbot and explains the fundamental design of the acceleration based cascaded controller, PD controller and the position controller schemes. **Chapter 6** presents simulation and experimental results to validate the effectiveness of the proposed angular motion estimation algorithm by using the estimated inertial angles, velocities and accelerations as feedback signals in the stabilization control of a ballbot. Finally, **Chapter 7** includes an overall discussion of the work done and concluding remarks. Possible future work that could not be covered inside the scope of this thesis is also discussed.

1.4 Publications

- F. Yavuz, M. Unel, "Robust Balancing and Position Control of a Single Spherical Wheeled Mobile Platform", *The 42nd Annual Conference of IEEE Industrial Electronics Society* (IECON 2016), Florence, Italy, October 24-27,2016

Chapter 2

Literature Survey and Background

In this chapter, the literature survey about angular motion estimation techniques and ballbot platforms are presented. Topics that are closely related to the subject of sensor fusion and working principles of ballbot examples from literature, which serve as basic foundations of this thesis, are detailed here.

2.1 Sensor Fusion

In order to control a system for a specific task, it is necessary to know the internal states that can not be measured directly. State estimation covers the theory and tools to accurately determine these internal states from measurable sensor signals using sensor fusion. This comes along with a major challenge, where the sensor measurements are corrupted by noise. Sensor fusion covers various techniques and algorithms, where the Kalman filter and its variations are most commonly preferred in the literature. Kalman approaches also formed the basis of the proposed sensor fusion work covered in this thesis.

In the following subsections, the principle of Kalman filtering are introduced. Quick reviews, properties and equations for the generalized versions of Kalman Filter (KF) and Extended Kalman Filter (EKF) are presented. Reference [3] can be consulted for a detailed derivation and further information.

2.1.1 Kalman Filter

Kalman filter is a linear recursive mean squared error estimator that produces an optimal estimate of a system state represented by a stream of noisy data. If the modeled noise processes are Gaussian, it is the optimal state estimator, concerning minimizing the error variance between true and estimated states. It is a well known method for fusing sensor data in Attitude Heading and Reference System (AHRS) applications, as it can be designed to estimate the orientation of a body as well as sensor biases. In order to accomplish this task, it requires to know the measurement noise, the noise of the input to the filter, also the noise of the process by assuming all noises to be Gaussian distributed and with zero mean.

A standard Kalman filter requires a discrete time linear dynamic system model in state space form, for x_k is $n \times 1$ state vector, given by:

$$x_k = Fx_{k-1} + Bu_k + w_k \tag{2.1}$$

where F is a square symmetric state transition matrix, B is the input matrix and u_k is the input to the system. Also, w_k is the Gaussian distributed process noise with zero mean and with covariance Q_k, i.e. $w_k \sim N(0, Q_k)$.

Observation or measurement z_k of the true state is given as $m \times 1$ vector

$$z_k = Hx_k + v_k \tag{2.2}$$

H is called measurement matrix to map the true state into the measured states and v_k is the measurement noise similarly Gaussian distributed with zero mean and with covariance of R_k, i.e. $v_k \sim N(0, R_k)$.

In order to proceed to Kalman filter stages, (F, H) must be a observable pair where the observability matrix must have full rank of n, i.e. $rank(O) = n$.

$$O = \begin{bmatrix} F & FH & \dots & FH^{n-1} \end{bmatrix}^T \tag{2.3}$$

In order to complete estimation task to achieve optimal state $\hat{x}_{k|k}$, Kalman filter algorithm can be constructed as follows:

- **Prediction Stage:** We predict the state as

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + Bu_k \tag{2.4}$$

  where $\hat{x}_{k|k-1}$ is the predicted state at time k and $\hat{x}_{k-1|k-1}$ is the previously estimated state.

- **A Priori Covariance Stage:** Then, a priori error covariance matrix $P_{k|k-1}$ is calculated based on the previous error covariance matrix $P_{k-1|k-1}$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_k \tag{2.5}$$

  and Q_k can be taken as constant diagonal matrix for most cases.

- **Kalman Gain:** The next step is to calculate the Kalman gain which is used to exude how much we trust the innovation comes from measurements.

$$K_k = P_{k|k-1}H^T \left(HP_{k|k-1}H^T + R_k\right)^{-1} \tag{2.6}$$

Where R_k is measurement noise covariance matrix, similarly can be taken as constant diagonal matrix.

- **Update Stage:** To perform update, which corrects the state estimates based on measurements, lets compute the residual between the measured state z_k and the predicted state $\hat{x}_{k|k-1}$, which is also called as innovation:

$$\bar{y}_k = z_k - H\hat{x}_{k|k-1} \tag{2.7}$$

Then, update the state as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\bar{y}_k \tag{2.8}$$

where $\hat{x}_{k|k}$ is the optimal state at time k.

- **A posteriori Covariance Stage:** Finally, the last thing we will do is update the a posteriori error covariance matrix,

$$P_{k|k} = (I_{n\times n} - K_k H)\, P_{k|k-1} \tag{2.9}$$

Initial value for the state $x_{0|0}$ can be selected arbitrarily and estimation covariance matrix $P_{0|0}$ can be can be selected as positive definite diagonal square matrix with fairly large values.

2.1.2 Extended Kalman Filter

Many practical systems have nonlinear process and/or measurement dynamics due to their nature. In order to employ Kalman Filters to those nonlinear models, the Extended Kalman Filter (EKF) was developed. The EKF is based on linearization of process and measurement models in terms of the current estimated state.

The most common representation of nonlinear systems in the continuous-time state space form at the continuous time t. given as

$$\dot{x}(t) = f\left(x(t), u(t)\right) + w(t) \tag{2.10}$$

$$z(t) = h\left(x(t)\right) + v(t) \tag{2.11}$$

where f is a nonlinear state dynamics as a function of the current state $x(t)$ and the input $u(t)$, h representing a nonlinear measurement model. And $w(t)$ and $v(t)$ are the Gaussian distributed process noise with zero mean and with covariances $Q(t)$ and $R(t)$, respectively.

$$w(t) \approx N\left(0, Q(t)\right), \quad Q(t) = E[w(t)w(t)^T] \tag{2.12}$$

$$v(t) \approx N\left(0, R(t)\right), \quad R(t) = E[v(t)v(t)^T] \tag{2.13}$$

Then, the system is linearized about the estimated state vector $x(t)$. Hence the continuous-time linearized system can be represented as:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + w(t) \tag{2.14}$$

$$z(t) = C(t)x(t) + v(t) \tag{2.15}$$

where:

- $A(t)$: Jacobian of $f\left(x(t), u(t)\right)$ wrt $x(t)$.
- $B(t)$: Jacobian of $f\left(x(t), u(t)\right)$ wrt $u(t)$.
- $C(t)$: Jacobian of $h\left(x(t)\right)$ wrt $x(t)$.

In plain words, $B(t)$ is not need to be calculated, because $\dot{x}(t)$ will be formed from $f\left(x(t), u(t)\right)$.

Finally, Kalman Filters are generally implemented in the discrete-time as in Section 2.1.1, thus the above continuous-time process model must be converted to discrete time. For this purpose continuous-time linearized dynamics model and process noise matrices, $A(t)$ and $Q(t)$ are converted into the discrete-time state transition matrix, F_k, and discrete-time process noise covariance matrix, Q_k by using the Van Loan method [4].

Linearized discrete-time model obtained as

$$x_k = F_k x_k + B_k u_k + w_k, \quad E\left(w_k w_k^T\right) = Q_k \tag{2.16}$$

$$z_k = H_k x_k + v_k, \quad E\left(v_k v_k^T\right) = R_k \tag{2.17}$$

Estimation task can be done as in Section 2.1.1, by calculating Jacobian matrices F_k and H_k evaluated at the current state estimate at each step.

2.2 Sensor Fusion for Attitude Estimation

The attitude estimation of rigid body systems from sensor measurements has been popular over the years and been used in many applications such as unmanned vehicle control, platform stabilization, human motion tracking and underwater navigation [5–7]. Modern attitude heading reference systems (AHRS) are combination of strapped-down multi-axis inertial sensors and microprocessors to provide an accurate measurement of the orientation of a vehicle with respect to inertial frame by using estimation algorithms to compute the attitude from several sensor measurements. The recent developments in integrated circuits and micro-electro-mechanical systems (MEMS) technology has facilitated production of low cost, light weight and highly accurate inertial measurement units (IMU) and processors to be used in the development of small and reliable AHRS devices for both research and industrial

applications. A typical IMU consists of a triad of orthogonal gyroscopes, accelerometers and magnetometers on all 3-axis, that measure rotation rates, accelerations and earth's magnetic field respectively. Since the individual use of these sensors is not sufficient for reliable attitude estimation, usually a sensor fusion algorithm is employed to combine individual sensor measurements to achieve estimation results that are far better than those of a single sensor. For example, gyroscopes have high bandwidth and does operate in a fast manner. Measuring the attitude angles by integrating angular rates measured by gyroscope is not feasible due to the drift problem resulted from the accumulation of the bias errors. On the other hand, accelerometers have low bandwidth and therefore they provide relatively accurate roll and pitch angles from the components of the gravity vector in a slow manner. Similarly, determining yaw angle from the components of the earth's magnetic field using a magnetometer is a drift-free, but, slow process. To obtain fast and accurate attitude angles, outputs of inertial sensors must be fused by sensor fusion.

The development of efficient estimation algorithms that can accurately estimate the orientation of a rigid body from a strapped-down IMU sensors has attracted the attention of many researchers. One of the first studies for attitude estimation provided by Wahba in 1965, was a mathematical problem related to finding the optimal rotation matrix where the solution gives an algebraic estimate based on the vector observations using a least squares technique without filtering process [8]. Later, various techniques such as the Singular Value Decomposition (SVD) [9] and Quaternion Estimation (QUEST) [10] were developed to tackle the attitude estimation problem.

Kalman filters are the most extensively used sensor fusion methods for the majority of attitude estimation algorithms that employ IMU. Kalman filter has so many variations such as Extended Kalman filter (EKF), Unscented Kalman filter (UKF) and Adaptive Kalman filter (AKF). EKF is the nonlinear version of the classical Kalman filter [11, 12]. Kim and Golnaraghi used an Extended Kalman Filter to fuse signals from a low cost IMU to estimate the orientation. A quaternion based process

model is used to avoid the problem of singularities in Euler angle representations. Simulation and experimental results show that the filter tracked the roll, pitch and yaw angles quite accurately and significantly corrected the yaw angle error drift [13]. Sabatini presented a quaternion based Extended Kalman Filter for estimating the three dimensional orientation of a rigid body. Simulations and experiments are done to evaluate the algorithm performance especially under the critical observability conditions [14]. Li and Wang proposed an effective Adaptive Kalman Filter to integrate low cost MEMS accelerometers, MEMS gyroscopes and magnetometers with an Attitude and Heading Reference System (AHRS) [15]. Literature includes several studies where the signals from the same sensors can be processed and fused properly to get angular motion information. One of the common approach is the estimation of angular motion using gyro-free IMU, where the angular motion vector is produced by using 12 separate single axis linear accelerometers [16]. In addition to the cited works, attitude estimation includes many alternatives in the current literature as discussed in the survey [17], where virtually all techniques have a similar structure with rotational rate gyroscope data being fused with vector observations of the gravitational and/or magnetic fields.

2.2.1 Representing Attitude

This section explains necessary layout of coordinate systems required for representing attitude. The attitude or orientation generally indicates how a coordinate system is aligned with respect to another one. In literature, most common representations are Euler angles and quaternions.

The most well-known approach to represent the attitude of a body is vector of Euler angles. The Euler angles are three dimensional attitude representation presented by Leonhard Euler to portray the attitude of a rigid body. A few arrangements of Euler angles are so generally utilized that they have names and notations that have

turned out to be a piece of the normal speech such as the symbols ϕ, θ and ψ are used for names of roll, pitch, and yaw of a body [18]. However, the three dimensional attitude parameterizations have singularity and Euler angles where ϕ and ψ along with their derivatives are not well-defined for $\theta = \pm\frac{\pi}{2}$. These insufficiencies in the Euler angles have driven researchers to utilize quaternions as a parametrization of the orientation. Where the unit quaternions have no singularities and well defined for the integration of the angular velocity of a body.

In spite of the singularity problem, in our work, we will define the orientation of a body by using Euler angles. Because the working range of the controlled body pitch angle will not reach to $\theta = \pm\frac{\pi}{2}$ for simulations and experiments. Additionally, the accompanying definitions and properties for these representations are utilized from [19].

2.2.1.1 Coordinate Systems

We consider the relationships between data expressed in two different coordinate systems.

- A **body-fixed** coordinate system is need to be defined in order to describe the orientation of an object. It is rigidly attached to the objects geometry and moves with the object as it moves or rotates.
- Also, an **inertial coordinate** system is usually defined as a non-moving, non-rotating coordinate. A common choice is an earth-fixed coordinate system with North-East-Down (NED) convention.

2.2.2 Inertial Measurement Unit (IMU)

In recent years, advances in the development of micro-electromechanical systems (MEMS) have significantly improved the cost-performance ratio of inertial sensors such as gyroscopes, accelerometers and magnetometers that measure angular velocities, linear accelerations and earth's magnetic field, respectively. An inertial measurement unit (IMU) is a package of those inertial sensor that consists of a triad of orthogonal gyroscopes, accelerometers and magnetometers on all 3-axis (see Figure 2.1).

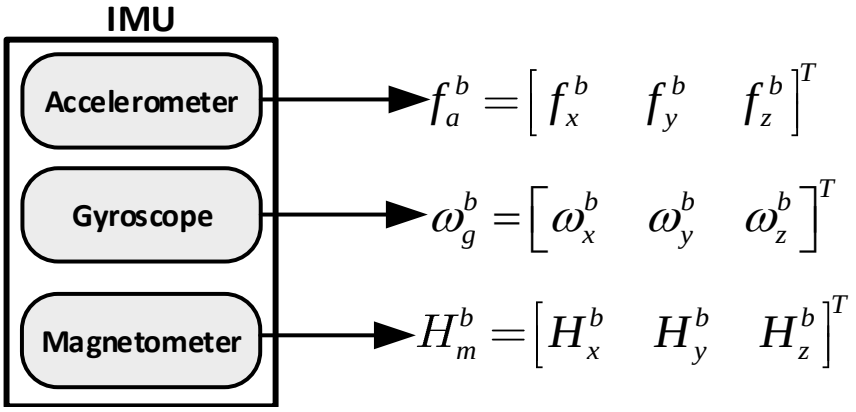

FIGURE 2.1: Block diagram showing the IMU assembly and its signals.

In practice, due to their mechanical and electrical natures, MEMS inertial sensors are frequently mixed with two types of errors, deterministic and stochastic errors [20]. Deterministic errors mostly include stable and repeatable biases, which can be eliminated through some suitable calibration. However, stochastic errors are based on multiplicative or additive measurement noises that can not be eliminated via simple calibration. The bias is commonly constant error that observed in the accelerometer and gyroscope measurements when there is no input acceleration or rotation. In the following subsections, sensor subsystems of an IMU are briefly described, however the detailed mathematical modeling of those sensors are presented in Section 3.1.

2.2.2.1 Gyroscope

Gyroscopes or angular rate sensors measures rotation rate around fixed axes with respect to inertial frame then expressed in the sensor coordinate system. They are used extensively in the applications of inertial navigation, robotics, aerospace and automotive. There are several factors to consider when evaluating a gyroscope, such as bias, scale factors, random angular walk and alignment error. Bias error occurs when the sensor is laying still and if it has not got zero mean on all axes. Scale factors are related to the conversion from analog sensor measured voltages to rotation rate. Random angular walk is the phenomena that is experienced when the angular rate measurements are integrated to get the angle.

2.2.2.2 Accelerometer

Accelerometers are one the main component of an inertial measurement unit (IMU). Accelerometers measure specific forces though a sensitive axis in the body frame. Working principle of an accelerometer is based on Newton's first law, where an object at rest stays at rest and a moving object stays in motion unless attracted by an unbalanced force. The accelerometer can be utilized to evaluate orientation of the body. Beside orientation, in mobile robotic applications, accelerometer measurements can be used to determine the position of the robot by dead-reckoning for indoor implementations. Bias error occurs in accelerometer measurements as well.

2.2.2.3 Magnetometer

A magnetometer measures the heading and the intensity of the magnetic field around the sensor. Although the magnetometers are mentioned as an additional sensor for IMU, nowadays they are necessary part of an IMU used for detection of yaw angle. In mobile robotic applications, a magnetometer can be utilized to calculate heading

of the robot where the heading is determined by measuring the horizontal component of Earths magnetic field vector.

2.3 Sensor Fusion and Control

Attitude estimation methods, mentioned in previous sections, are generally used in conventional control of the stabilized platforms which are becoming increasingly popular in different application areas such as target identification, security and defense, gun-turret control and mobile omnidirectional robots. All of these applications require highly precise stabilization because small angular displacements may cause large position errors or failures. The goal of the stabilization control is to maintain the desired orientation by rejecting external disturbances due to terrain changes, high-frequency vibrations and sudden shocks, wind and other environmental factors. Stabilization control performance largely depends on reliable angular motion feedback. Sensor fusion techniques can be used to enhance the robustness and stability of those systems against sensor failure.

2.4 Construction of Inertial Angular Velocity and Acceleration

The success of stabilization control largely depends on reliable angular velocity and acceleration feedback, besides angular position. However, obtaining reliable velocity and acceleration signal is difficult. In the straightforward method, obtaining an angular velocity and acceleration signals by taking time derivatives of measured position and velocity signals. Although it is proposed in papers [21, 22], obtaining acceleration from position signal with double differentiation creates exceptionally weak results due to sensor noise amplification problems. On the other hand, there

are some angular accelerometers to directly measure angular accelerations; however they are costly and produced by only a few manufacturers.

Several methods have been proposed in the literature to estimate angular acceleration. Han et al. [23] proposed a Newton Predictor Enhanced Kalman Filter (NPEKF) to estimate angular accelerations. This estimator provides a wide bandwidth and a small phase lag of the estimated acceleration while attenuating noises. Moreover, some estimation techniques are provided by using gyro-free IMU [16, 24–26]. Those methods only use linear accelerometer measurements have been developed to estimate angular velocities and accelerations. However, they rely only on low bandwidth linear accelerometers and do not employ high bandwidth gyro measurements.

2.5 Ballbots: Single Spherical Wheeled Mobile Platforms

Research on omnidirectional and self-balancing mobile platforms has been quite active in robotics and control communities in the last decade due to their ability to freely move in all directions on the horizontal plane. Self-balancing mobile platforms with single spherical wheel, also known as ballbot platforms, are suitable example of special wheeled omnidirectional robots. In such systems, a body balances on a single ball which used as a replacement of conventional wheels to achieve omnidirectional motion. The spherical wheel is mostly manipulated through rollers or wheels attached to the actuators on the body. In summary, single spherical wheeled mobile platforms must actively balance on a single point of contact with the ground which reduces energy for motion due to less friction but makes system to be inherently unstable. Beside being a research valuable topic, single spherical wheeled platforms are aimed to perform missions as accompanying handicapped or elder people to

move around (e.g. in hospitals, museums, supermarkets), furthermore as a vehicle for personal transportation [27].

Single spherical wheeled platforms have been popular over the years, and many researchers have already paid attention on modeling, design and construction of such systems. The first ball balancing robot called ERROSphere (Equilibrating Robot Rolling On Sphere) [28], was presented by Havasi in 2005, merely can balance on the ball without any further functionality. B.B.Rider [27] was another omnidirectional robot stabilizes on a basketball developed by Endo et al. However, they provided only the design and did not presented any experimental results. Initial implementation on ballbot type robots, actually an inverted pendulum mounted on a spherical wheel, started in 2006.

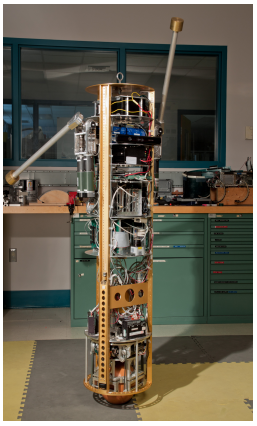

FIGURE 2.2: CMU

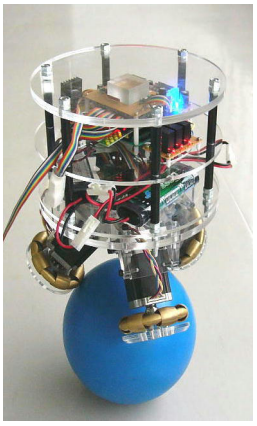

FIGURE 2.3: TGU

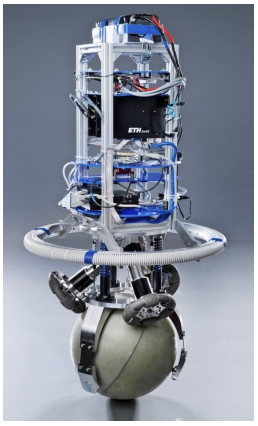

FIGURE 2.4: Rezero

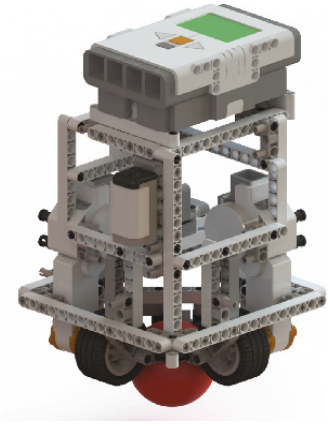

FIGURE 2.5: NXT

FIGURE 2.6: Overview of different Ballbots.

The first example of the ballbot morphology constructed in Carnegie Mellon University in the United States by Lauwers et. al [29] as human size ballbot which has an inverse mouse-ball drive mechanism to actuate the ball and can interact with humans. Although the robot with the inverse mouse-ball driven mechanism worked well, it could not rotate around yaw axis. During the following years, many other version of the inverse mouse-ball driven ballbots were developed by adding

yaw control and different features such as stabilizer legs [30] and functional arms [31, 32] (see Figure 2.2). Later on, many other configurations are developed by using different actuating systems. Most commonly, omniwheels driven by motors are used in many applications for better control performance. Researchers in the Tohoku Gakuin University (TGU) in Japan developed a much more smaller ballbot (see Figure 2.3) as compared to CMU's one with three omniwheels and [1]. It is main difference was that it can also perform yaw motion. In 2009, a 20 cm length small ballbot [33] developed in The University of Adelaide (UA) in Australia by using LEGO™ Mindstorms NXT components (see Figure 2.5) which has only two wheels to drive the ball like the inverse mouse-ball driven ones. Among all, the Rezero [2, 34] (see Figure 2.4) is the most famous one, to the best of our knowledge, developed in 2010 at ETH Zurich in Switzerland with three omniwheels which has a high dynamic robustness.

Moreover, novel actuating mechanisms are presented in [35] as partially sliding rollers and in [36] spherical induction motor ball wheel designed exclusively for a ball balancing mobile robots. Initial results for the ballbot with a special induction motor ball wheel are presented in [37]. Additionally, several commercial ventures were exhibited from research to commercialization with the Rezero [34] for omniwheel structure and the mObi for inverse mouse-ball drive structure.

2.5.1 Design and Modeling

Essentially, all ballbot designs are based on the same principle where the manipulation of the ball through rollers or wheels attached to the actuators on the body standing on the ball. As mentioned before, there were mainly two types of the ballbot design: the inverse mouse-ball driven mechanism and omniwheels driven mechanism.

The first proposed actuation idea for such systems was the inverse mouse-ball design, either using two rollers [33] or four rollers [29]. Although the ballbot with the inverse mouse-ball driven mechanism worked well, it could not rotate around yaw axis. During the following years, many other version of the inverse mouse-ball driven ballbots were developed by adding yaw control and different features such as stabilizer legs [30] and functional arms [31, 32]. In other versions, omniwheels are used to actuate the ball for better control performance and reduced friction [1, 2, 34]. In these omniwheeled ballbots,the body includes three or four single-row omniwheel driven by independent dc motors which are equally placed each other at the bottom.

Modeling of a ballbot platform is crucial in order to be able to rapidly develop and tune the controllers and also test them without actually using the experimental setup. Planar system modeling has been commonly used in the initial studies on the ballbot by dividing the 3D system into the independent planar models by neglecting coupling effects between these models [29, 30, 32, 33, 38]. Two of those planar models are identical and the third one describes the rotation around the z-axis in the body fixed reference frame. When modeling these 2D planes, the vertical planes are assumed to be independent. The dynamical model obtained by neglecting coupling effect is much simpler than a 3D model. In order to provide pure nonlinear model of a ballbot, more realistic mathematical models are presented by taking into account all coupling effects. In [38], besides derivation of 2D model, a 3D model of a ballbot is also presented where the calculations are separated into multiple, smaller computations and evaluated by using Mathematica. Inal et al. [39] developed a nonlinear 3D model for a ballbot using unit quaternions to represent rigid body rotations.

2.5.2 Control Approaches

Although much progress has been made on the development on spherical wheeled mobile platforms, controlling motion while maintaining balance is a challenging task because their underactuated structure has fewer control inputs than their degrees of freedom. In the literature, various balancing and positioning control algorithms for ballbot platforms were presented. The most common approach for controlling both the orientation and the position of the ballbot is LQR (Linear-quadratic regulator) based state feedback control. In [29], the ballbot controller consisted of an inner PI (Proportional+Integral) control loop for controlling angular velocity of the ball and an outer LQR based full state feedback control loop. Later on, in [30, 40] these controllers were replaced by PID (Proportional+Integral+Derivative) controller for both balancing and tracking set points. In [1], inputs of the plant were decoupled so that simple linear PD controllers were used to control virtual wheels. Then virtual accelerations were converted into the velocities of three omniwheeled stepping motors by using kinematic relations. Differently, Rezero is controlled by quasi-nonlinear feed-forward control using the idea of gain scheduling [34]. In [39, 41] inverse dynamics controller is used to cancel out accelerations on body attitude degrees of freedom, in combination with a PD controller to stabilize the body angles and the position. Another important issue is the path following control for the ballbot, which ensures that the robot to follow a path while maintaining balance. Initial research in position control of the ballbot focused on station keeping where the postural stabilization by an outer loop using LQR [29] or PID [30]. These approaches were successful in realizing station keeping and slow line following. In [31, 40] dynamic constraint-based optimal shape planner is used to plan shape trajectories which are used as reference for the body angles during tracking for the inverse mouse-ball driven ballbot. In [42], the tuning of linear controllers by LMIs is presented to provide robustness against some parametric uncertainties in the system by using Matlab™ model of LEGO™ Mindstorms NXT ballbot.

In the aforementioned methods, LQR and PID controllers can also get the optimal results but were not sufficiently robust for balancing purpose to handle sudden deviations due to the external disturbances or highly inclined initial conditions. On the other hand using nonlinear controller is quite complicate because of the complex structure of the dynamic equation of the ballbot. Therefore, the development of robust approaches for controlling the ballbot type platforms is important.

Chapter 3

Sensor Fusion Model

In this chapter, detailed derivation and equations of fusion algorithms and modeling of inertial sensors will be presented.

3.1 Sensor Modeling

As mentioned earlier, in this work, a three-axis IMU will be considered, composed of three axis accelerometers, three axis rate gyroscopes and, finally, three axis magnetometers. MEMS inertial sensors (gyroscopes, accelerometers and magnetometers) are modeled by corrupting the true sensor measurements with sensor errors. In practice, due to their mechanical and electrical natures, MEMS inertial sensors are frequently mixed with two types of errors: deterministic and stochastic errors [20]. The development of the deterministic and stochastic error model for an inertial sensor is one of the most important steps for building a reliable navigation system. Deterministic errors mostly include stable and repeatable biases, which can be eliminated through some suitable calibration. However, stochastic errors are based on multiplicative or additive measurement noises that can not be eliminated via simple

calibration. In this work, random noises are assumed to be drawn from a normal distribution.

3.1.1 Gyroscope Modeling

Gyroscopes measure angular rates in the body frame. The true angular rates about the body axes, denoted as vector $\omega_0^b = \begin{bmatrix} \omega_{0_x}^b & \omega_{0_y}^b & \omega_{0_z}^b \end{bmatrix}^T$, can be simply determined from true Euler angles and rates using the inverse velocity transformation matrix $\mathbb{E}$ as:

$$\omega_0^b = \begin{bmatrix} \omega_{0_x}^b \\ \omega_{0_y}^b \\ \omega_{0_z}^b \end{bmatrix} = \mathbb{E}\Omega_0 = \begin{bmatrix} 1 & 0 & \sin\theta_0 \\ 0 & \cos\phi_0 & \cos\theta_0 \sin\phi_0 \\ 0 & -\sin\phi_0 & \cos\phi_0 \cos\theta_0 \end{bmatrix} \begin{bmatrix} \dot{\phi}_0 \\ \dot{\theta}_0 \\ \dot{\psi}_0 \end{bmatrix} \tag{3.1}$$

Then, gyroscope output can be modeled as:

$$\omega_g^b = \omega_0^b + b_g + \eta_g \tag{3.2}$$

where b_g and η_g represent the gyro biases and noises.

3.1.2 Accelerometer Modeling

Accelerometers measure specific forces in the body frame. These forces are the total accelerations relative to free-fall and represented by f_a. It is assumed that IMU is attached to the body center and earth rotation effects are neglected. Then, the true specific forces are computed in the inertial frame as follows:

$$f_0^n = \frac{d}{dt}V^n + g, \qquad g = \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} \tag{3.3}$$

where $\dot{V}^n$ denotes a vector of the translational accelerations of the body and g is the acceleration due to gravity. Since accelerometers measure the specific forces in the body frame, f_0^n is multiplied by the rotation matrix, R_n^b, to transform from *inertial* to *body* frame as shown in (3.4).

$$f_0^b = R_n^b f_0^n = R_n^b \frac{d}{dt} V^n + R_n^b g \tag{3.4}$$

Usually V^n is assumed to be constant and therefore $\dot{V}^n = 0$. Then, (3.4) is expressed as:

$$f_0^b = R_n^b g = \begin{bmatrix} -g \sin\theta \\ g \cos\theta \sin\phi \\ g \cos\phi \cos\theta \end{bmatrix} \tag{3.5}$$

where ϕ and θ are the roll and pitch angles. The output of the accelerometer is modeled as:

$$f_a^b = f_0^b + b_a + \eta_a \tag{3.6}$$

where b_a and η_a define the accelerometer biases and noises.

3.1.3 Magnetometer Modeling

Magnetometers measure the strength of the magnetic fields in the body frame. The magnetometer output is modeled as:

$$H_m^b = H_0^b + b_m + \eta_m \tag{3.7}$$

where H_0^b defines the true magnetometer measurements in the body frame, b_m and η_m represent the magnetometer biases and noises.

3.2 EKF-based AHRS Using Euler Angles

Conventional sensor fusion methods based on Kalman filter estimate Euler angles and the gyroscope biases [17]. The state vector is defined as follows:

$$x = \begin{bmatrix} \phi & \theta & \psi & b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}^T = \begin{bmatrix} \Theta^T & b_g^T \end{bmatrix}^T \tag{3.8}$$

where $\Theta \equiv \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ represent Euler angles (roll, pitch and yaw) and the gyroscope biases are denoted by $b_g \equiv \begin{bmatrix} b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}^T$.

The relationship between the Euler rates $\Omega = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$ and the angular velocity of the body $\omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ is formed by velocity transformation matrix. Multiplying this matrix by the angular velocity in the global coordinates results in the Euler rates vector [19]. The nonlinear process dynamics is described using this kinematic relationship as:

$$\Omega = \mathbb{B}\omega \tag{3.9}$$

where $\mathbb{B}$ is the velocity transformation matrix and defined as:

$$\mathbb{B} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\theta & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \tag{3.10}$$

With respect to the state vector (3.8), the following continuous-time process dynamics is used as

$$\frac{d}{dt}x = \begin{bmatrix} \dot{\Theta} \\ \dot{b}_g \end{bmatrix} = \begin{bmatrix} \mathbb{B}\left(\omega_g^b - b_g\right) \\ \\ 0_{3\times 1} \end{bmatrix} + w \tag{3.11}$$

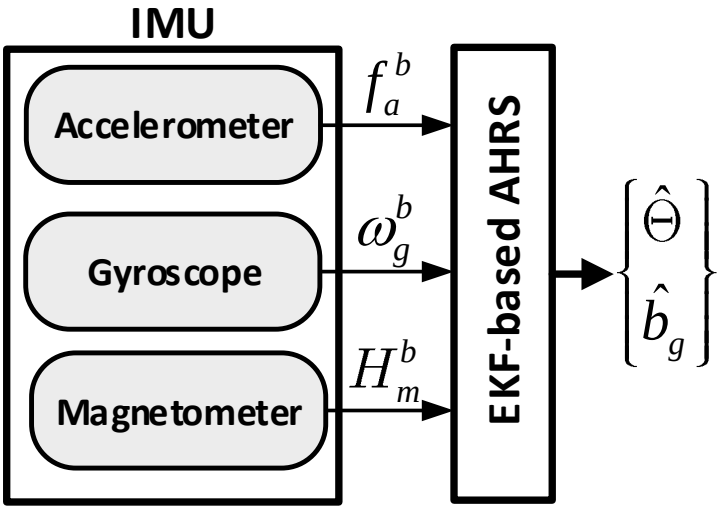

FIGURE 3.1: Block diagram of the EKF-based AHRS.

where

$$\omega_g^b = \left[\omega_{g_x}^b \;\; \omega_{g_y}^b \;\; \omega_{g_z}^b\right]^T \tag{3.12}$$

$$b_g = \left[b_{g_x} \;\; b_{g_y} \;\; b_{g_z}\right]^T \tag{3.13}$$

and the measurement vector provides sufficient observability as

$$z = \left[f_a^b \quad \psi_m\right]^T \tag{3.14}$$

where

$$f_a^b = \left[f_{a_x}^b \;\; f_{a_y}^b \;\; f_{a_z}^b\right]^T \tag{3.15}$$

and ψ_m is calculated from magnetometer measurements as

$$\psi_m = \tan^{-1} \frac{H_{m_y}^b}{H_{m_x}^b} \tag{3.16}$$

The overall algorithm for EKF-based AHRS is provided in Algorithm 3.1.

Algorithm 3.1 EKF-based AHRS

if $t = 0$ **then initialize**
$n = 6$
$\hat{x}_{0|0} = rand(n, 1), \quad P_{0|0} = 100 \times I_{n \times n}$
while $f_a^b \neq \emptyset \parallel \omega_g^b \neq \emptyset \parallel H_m \neq \emptyset$ **do**
procedure $\hat{x}_m = \text{EKF}\ (f_a^b,\ H_m,\ \omega_g^b)$
prediction at $t = t_k$
$\hat{\dot{x}}_{k|k} = f\left(\hat{x}_{k|k}, \omega_g^b - \hat{b}_g\right)$
$\hat{x}_{k+1|k} = \hat{x}_{m_{k|k}} + \hat{\dot{x}}_{k|k} dt$
$P_{k+1|k} = \Phi_k P_{k|k} \Phi_k^T + Q_k$
update
$K_k = P_{k+1|k} \Gamma_k^T \left(\Gamma_k P_{k+1|k} \Gamma_k^T + R\right)^{-1}$
$z_k = \begin{bmatrix} f_a^b & \psi_m \end{bmatrix}_k^T$
$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_k \left(z_k - \hat{y}_{k+1|k}\right)$
$P_{k+1|k+1} = \left(I_{n \times n} - K_k \Gamma_k\right) P_{k+1|k}$
$\hat{b}_g = \hat{x}_{k+1|k+1}(3:6)$

3.3 Master-Slave Kalman Filter

In this section, a novel sensor fusion method for reliable angular motion estimation using a master-slave Kalman filter is presented. It employs both an extended Kalman filter (EKF) and a classical Kalman filter (KF) in a master-slave configuration. In order to estimate the angular position, velocity and acceleration, the state of the EKF is extended to include both Euler rates and accelerations in addition to Euler angles. Gyro biases are also included into the state vector. While the EKF feeds the KF with the estimated gyro biases, the KF estimates bias compensated angular velocity, acceleration and jerk signals in the body frame and sends back to the EKF.

The state vector in (3.8) is extended to include angular velocities and accelerations; i.e.

$$x = \begin{bmatrix} \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} & \ddot{\phi} & \ddot{\theta} & \ddot{\psi} & b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}^T = \begin{bmatrix} \Theta^T & \Omega^T & \Gamma^T & b^T \end{bmatrix}^T \quad (3.17)$$

where $\Gamma \equiv \begin{bmatrix} \ddot{\phi} & \ddot{\theta} & \ddot{\psi} \end{bmatrix}^T$ defines Euler accelerations.

In accordance with the new state vector (3.17), the following continuous-time process dynamics is obtained by differentiating the nonlinear dynamics in (3.9):

$$\frac{d}{dt}x = \begin{bmatrix} \dot{\Theta} \\ \dot{\Omega} \\ \dot{\Gamma} \\ \dot{b}_g \end{bmatrix} = \begin{bmatrix} \mathbb{B}\omega \\ \dot{\mathbb{B}}\omega + \mathbb{B}\alpha \\ \ddot{\mathbb{B}}\omega + 2\dot{\mathbb{B}}\alpha + \mathbb{B}\gamma \\ 0_{3\times 1} \end{bmatrix} + w \quad (3.18)$$

where angular accelerations and jerks are denoted by $\dot{\omega} \equiv \alpha = \begin{bmatrix} \alpha_x & \alpha_y & \alpha_z \end{bmatrix}^T$ and $\dot{\alpha} \equiv \gamma = \begin{bmatrix} \gamma_x & \gamma_y & \gamma_z \end{bmatrix}^T$ in the body coordinate frame.

Also, the measurement vector extended by using angular velocities, $\hat{\omega}$, and accelerations, $\hat{\alpha}$ estimated by KF which provides sufficient observability as:

$$z = \begin{bmatrix} f_a^b & \psi_m & \hat{\omega} & \hat{\alpha} \end{bmatrix}^T \quad (3.19)$$

Non-deterministic effects and modeling errors are represented by the process noise, w. In (3.18), gyro biases are assumed to be constant. This model is known as a *Wiener process* and can be considered as a special case of *Gauss-Markov process* [43].

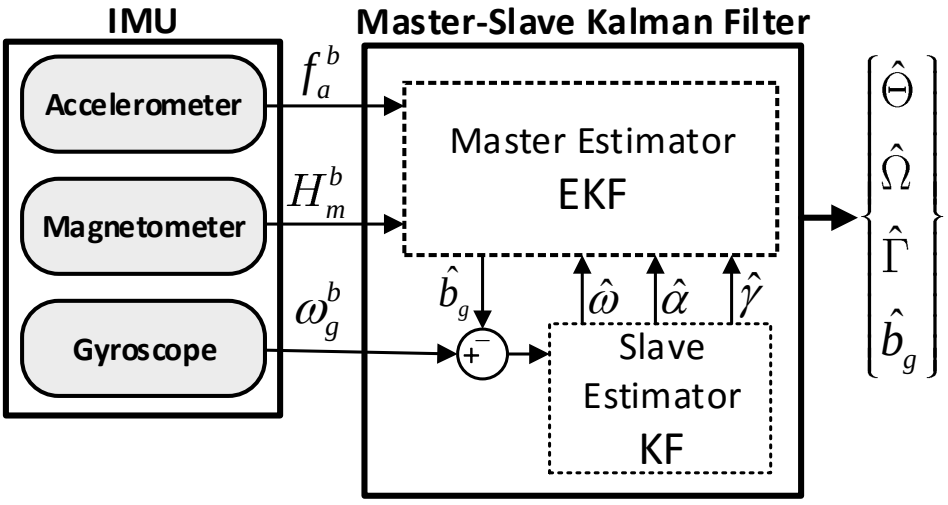

FIGURE 3.2: Block diagram of the master-slave Kalman filter.

Remark 1. Gyro biases can also be modeled using *Singer Model*. This model assumes that the gyro bias is a zero-mean stationary *first order Markov process* [26]. The continuous time bias model is defined as:

$$\dot{b}_g = -\beta b_g + w \tag{3.20}$$

where w is a zero mean white noise and β is the reciprocal of the time constant. Note that $\beta = 0$ implies constant bias model.

To estimate the state vector x in (3.17), an extended Kalman filter that utilizes sensor measurements will be implemented. To run the EKF, we need to compute ω, α and γ that appear on the right hand side of (3.18). Since there are no additional sensors to measure angular accelerations and jerks, they need to be estimated from gyro measurements. To this end, a slave type Kalman filter (KF) is introduced to estimate ω, α and γ using bias compensated gyro readings. Since biases will be estimated by EKF and used as inputs to the KF, we have a master-slave configuration in Figure 3.2 where the master estimator (EKF) feeds the slave estimator (KF) with bias estimates and the slave estimator returns estimated angular velocity, accelerations and jerks to the master estimator.

3.3.1 Master Estimator

Process dynamics of the master estimator is given by (3.18). Applying Euler's forward discretization to the process dynamics leads to:

$$\begin{bmatrix} \Theta \\ \Omega \\ \Gamma \\ b_g \end{bmatrix}_{k+1} = \begin{bmatrix} \Theta \\ \Omega \\ \Gamma \\ b_g \end{bmatrix}_{k} + T \begin{bmatrix} \mathbb{B}\omega \\ \dot{\mathbb{B}}\omega + \mathbb{B}\alpha \\ \ddot{\mathbb{B}}\omega + 2\dot{\mathbb{B}}\alpha + \mathbb{B}\gamma \\ 0_{3\times 1} \end{bmatrix}_{k} + w_k \tag{3.21}$$

where T is the sampling period. Measurement vector of the master estimator contains the specific force measurements, f_a^b, from accelerometer and the yaw angle, ψ_m, determined from the resolved components of the magnetic field measurements, H_m^b, in the horizontal plane along the heading axis [44]. In order to increase the observability of the state vector, the measurement vector of EKF is also extended by using angular velocities, $\hat{\omega}$, and accelerations, $\hat{\alpha}$ estimated by KF:

$$z_k = \begin{bmatrix} f_a^b & \psi_m & \hat{\omega} & \hat{\alpha} \end{bmatrix}_k^T = \begin{bmatrix} R_n^b(\Theta)g \\ \psi \\ \mathbb{E}\Omega \\ \mathbb{E}\Gamma + \dot{\mathbb{E}}\Omega \end{bmatrix}_k + v_k \tag{3.22}$$

where R_n^b is the rotation matrix from the inertial frame to the body frame and is given as:

$$R_n^b = \left[\begin{array}{c|c|c} \cos\psi\cos\theta & \cos\theta\sin\psi & -\sin\theta \\ \hline \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\theta\sin\phi \\ \hline \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi & \cos\psi\cos\theta \end{array}\right]$$

g is the gravitational acceleration vector and v_k is the measurement noise. The inverse velocity transformation matrix $\mathbb{E}$ is defined as:

$$\mathbb{E} = \mathbb{B}^{-1} = \begin{bmatrix} 1 & 0 & \sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{3.23}$$

As it is seen from (3.21) and (3.22), the process and measurement models of the master estimator are nonlinear. Since the EKF is based on the linearization of the nonlinear system, the functions (3.21) and (3.22) are linearized around the current estimated state vector x using the partial derivatives of the process and measurement functions. The state transition and measurement matrices F and H were found by a direct computation, which is performed by using the symbolic Matlab™ computational tool to find the matrices in terms of the states. F and H matrices are provided in Appendix A.

3.3.2 Slave Estimator

The slave estimator provides estimates of angular velocities, accelerations and jerks to the master estimator. The following process dynamics is constructed in discrete time, based on the classical laws of motion using Taylor series where angular jerks

are assumed to be constant:

$$\begin{bmatrix} \omega \\ \alpha \\ \gamma \end{bmatrix}_{k+1} = \begin{bmatrix} I_{3\times3} & TI_{3\times3} & 0.5T^2I_{3\times3} \\ 0_{3\times3} & I_{3\times3} & TI_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix} \begin{bmatrix} \omega \\ \alpha \\ \gamma \end{bmatrix}_{k} + w_k^s \tag{3.24}$$

where T is the sampling period and w_k^s is the process noise. The gyro bias estimated by the EKF is subtracted from the gyro readings to obtain bias compensated body angular velocity, $\omega_g^b - \hat{b}_g = \begin{bmatrix} \omega_{g_x}^b - \hat{b}_{g_x} & \omega_{g_y}^b - \hat{b}_{g_y} & \omega_{g_z}^b - \hat{b}_{g_z} \end{bmatrix}^T$, which is used as the measurement for the slave estimator:

$$y_k^s = \omega_{g_k}^b - \hat{b}_{g_k} = \begin{bmatrix} I_{3\times3} & 0_{3\times6} \end{bmatrix} \begin{bmatrix} \omega \\ \alpha \\ \gamma \end{bmatrix}_{k} + v_k^s \tag{3.25}$$

where v_k^s is the noise vector associated with the measurement process. A classical Kalman filter (KF) is employed to estimate angular velocity (ω), acceleration (α) and jerk (γ) using process and measurement models given by (3.24) and (3.25), respectively.

The overall algorithm for Master-Slave Kalman filter is provided in Algorithm 3.2.

Algorithm 3.2 Master - Slave Kalman Filter

if $t = 0$ **then initialize**

$n_m = 12,\ n_s = 9$

$\hat{x}_{m_{0|0}} = rand(n_m, 1), \quad P_{m_{0|0}} = 100 \times I_{n_m \times n_m}$

$\hat{x}_{s_{0|0}} = rand(n_s, 1), \quad P_{s_{0|0}} = 100 \times I_{n_s \times n_s}$

while $f_a^b \neq \emptyset \parallel \omega_g^b \neq \emptyset \parallel H_m \neq \emptyset$ **do**

procedure $\hat{x}_m = \text{EKF}\ (f_a^b,\ H_{mag},\ \omega_g^b)$

master prediction at $t = t_k$

$\omega = \hat{x}_{s_{k|k}}(1:3),\ \alpha = \hat{x}_{s_{k|k}}(4:6),\ \gamma = \hat{x}_{s_{k|k}}(7:9)$

$\dot{\hat{x}}_{m_{k|k}} = f\left(\hat{x}_{m_{k|k}}, \omega, \alpha, \gamma\right)$

$\hat{x}_{m_{k+1|k}} = \hat{x}_{m_{k|k}} + \dot{\hat{x}}_{m_{k|k}} dt$

$P_{m_{k+1|k}} = \Phi_{m_k} P_{m_{k|k}} \Phi_{m_k}^T + Q_{m_k}$

master update

$K_{m_k} = P_{m_{k+1|k}} \Gamma_{m_k}^T \left(\Gamma_{m_k} P_{m_{k+1|k}} \Gamma_{m_k}^T + R_m\right)^{-1}$

$z_{m_k} = \begin{bmatrix} f_a^b & \psi_m & \hat{\omega} & \hat{\alpha} \end{bmatrix}_k^T$

$\hat{x}_{m_{k+1|k+1}} = \hat{x}_{m_{k+1|k}} + K_{m_k}\left(z_{m_k} - \hat{y}_{m_{k+1|k}}\right)$

$P_{m_{k+1|k+1}} = \left(I_{n_m \times n_m} - K_{m_k} \Gamma_{m_k}\right) P_{m_{k+1|k}}$

$\hat{b}_g = \hat{x}_{m_{k+1|k+1}}(10:12)$

procedure $\hat{x}_s = \text{KF}(\omega_g^b,\ \hat{b}_g)$

slave prediction at $t = t_k$

$\hat{x}_{s_{k+1|k}} = \Phi_{s_k} \hat{x}_{s_{k|k}}$

$P_{s_{k+1|k}} = \Phi_{s_k} P_{s_{k|k}} \Phi_{s_k}^T + Q_s$

slave update

$K_{s_k} = P_{s_{k+1|k}} \Gamma_{s_k}^T \left(\Gamma_{s_k} P_{s_{k+1|k}} \Gamma_{s_k}^T + R_s\right)^{-1}$

$\hat{x}_{s_{k+1|k+1}} = \hat{x}_{s_{k+1|k}} + K_{s_k}\left(z_{s_k} - \hat{y}_{s_{k+1|k}}\right)$

$P_{s_{k+1|k+1}} = \left(I_{n_s \times n_s} - K_{s_k} \Gamma_{s_k}\right) P_{s_{k+1|k}}$

Chapter 4

Design and Construction of a Ballbot

This chapter details the structural design and electro-mechanical construction of a ballbot platform. The design process includes mechanical design of the ballbot by means of CAD (Computer-aided design) tools, investigation of the components and materials as well as testing the compatibility of each component. Additionally, construction of the ballbot includes manufacturing of the precisely modeled parts, testing each electronic hardware and building up the full scale ballbot by integration of all subcomponents in accordance with the CAD design.

Design objectives are given as follows:

- Drive mechanisms with omniwheels have to be placed with 120° between each other to form a triangle from top view.
- Each omniwheel should touch to the ball on a single point with 45° angle with respect to the center of the ball.

- Body has to be symmetric and its center of the mass should be close enough to the ball.
- The overall design should have durable structure and feasible dimensions.

In summary, the design and construction of the ballbot are conducted as:

- Building a CAD design of the prototype ballbot
- Selection of a ball and drive mechanism components.
- Checking whether if the selected components are dimensionally compatible in CAD software.
- Completing the CAD model to analyze modeling parameters (mass, inertia, dimensions) and revising the CAD model if necessary.
- Purchasing the decided components.
- Design the body structures (plate diameters and rod lengths) and connectors (motor bracket and pins) by considering ball and drive mechanism dimensions to meet design objectives.
- Manufacturing of the structural components using subtractive manufacturing tools. (e.g. CNC)
- Assembling the final ballbot by mounting all the components.

4.1 Mechanical Design

The mechanical design of the ballbot starts with CAD design that will be used as a guide for construction process. Especially, with the aid of CAD modeling, the size

of the subcomponents will be calculated precisely and thus manufactured easily. Besides, CAD model makes it easy to precisely calculate the moment of inertia of each component. To this end, a 3D CAD model of the ballbot is created in SolidWorks. The ball is assumed to be an homogeneous hollow sphere and the omniwheels are modeled as solid disks. The structural part of the body consists of three parallel circular plates and rods to connect these plates to each other. The sensors, dc motors, gear heads, brackets and pins are also modeled. The material of each part, from screws to beams are properly defined for components from SolidWorks material library.

In CAD design, several possible ball sizes were considered and dimensioning of other parts are done by considering the ball driving mechanism dimensions. Following this, structural components of the body are designed in a way that to meet the angle specifications as well as exactly fit all driving mechanism components into the body. The body of the ballbot itself is alike cylinder, which is designed to have same moment of inertias on both x and y axis. As a most important structure of the body, lower plate is modeled with a specific diameter of 174.46 mm to make the zenith angle of the omniwheel to be center of the ball. Two plates were used for the upper part of the body have 250 mm diameter, as this was determined to be compatible with overall appearance. Also, the linkages between the plates were designed using 130 and 100 mm length rods. The design guarantees that the ballbot does not seem too short or too long. In fact, it has suitable size to with those plates which can be used to carry the other required components.

It is important to note that the selected Maxon DC motor has a shaft of diameter 5 mm, and length of 12.8 mm. which are not enough to support the omniwheels. Therefore, a pin extension must be designed to connect the motor shaft to the omniwheel in a way that the motor can drive the omniwheel without slipping. The motor bracket are made of a 5 mm sheet that has the holes for the mounting of lower plate of the body and the each dc motor. Furthermore, it is bended with an

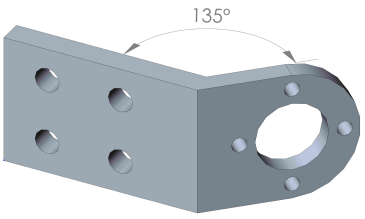

FIGURE 4.1: Motor bracket.

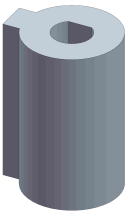

FIGURE 4.2: Motor-omniwheel pin.

angle of 135° to make the omniwheel contact points to be on zenith angle α (see Figure 4.1).

Finally, the arrangement of these parts was completed and assembled. In the following Figure 4.3, the final CAD design is presented,

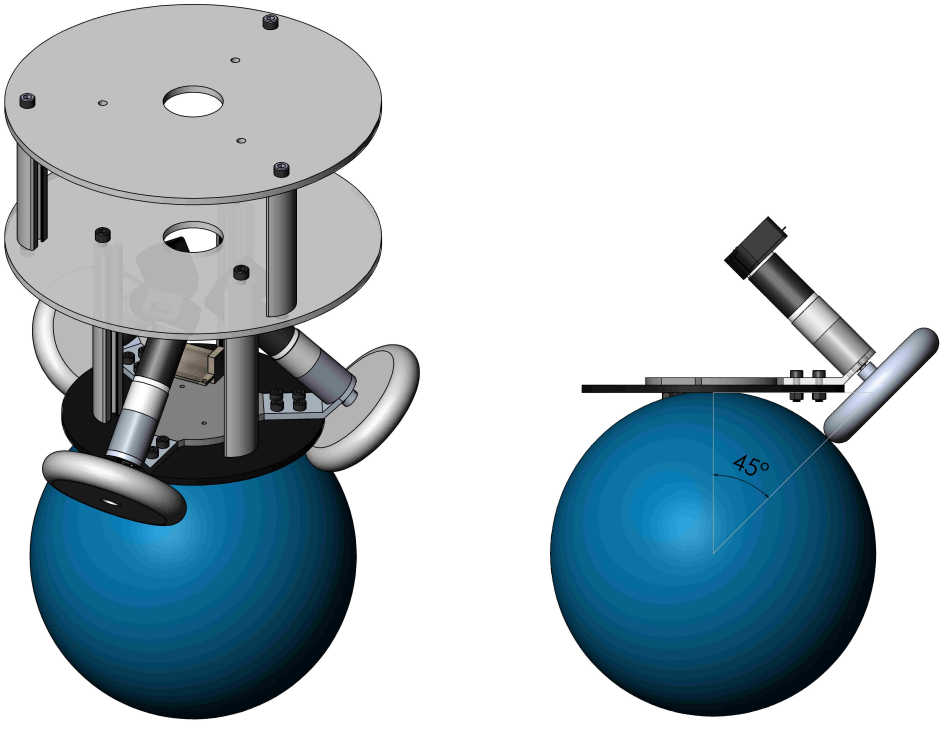

FIGURE 4.3: Final CAD design of the ballbot.

4.2 Component and Material Selection

We selected components to fulfill the following criteria:

- The structural components should have enough load capacity, light weight, durable material.

- The driving mechanism should have sufficient torque to drive the ball.

4.2.1 Ball Selection

As a starting point, the balls used in ballbot systems are investigated, then an appropriate one was searched in order to suffice high stiffness, weight and homogeneity. In literature, different types of balls have been employed in ballbot constructions. In most of the applications [1, 27], a basketball was used as spherical wheel which was not convenient due to insufficient weight and stiffness. In further ballbots, a bowling ball with or without rubber-coated is used [1, 33] but a bawling ball is generally quite heavy and not homogenized. In successfully implemented ballbots [29, 34], they produced their own balls either by coating a spun aluminum shell or two milled aluminum hollow half spheres with polyurethane which is extremely costly as compared to others. Finally, a ball type called “medicine ball” is found in a sports market with suitable size and mass. In our ballbot design, the medicine ball (see Figure 4.6), around 3 kg and with 10.8 cm radius, is preferred which provides better inertia and resistance than basketball.

4.2.2 Drive Mechanism

Drive mechanism of the ballbot consists of an omniwheel, a dc motor with gearbox and mechanical connectors such as pins and motor brackets. Starting point for

design of drive mechanism was to find dc motors with sufficient torque. High performance brushed Maxon RE26 DC Motor is selected, which has 27.7 mNm maximum continuous torque and 24 V nominal voltage. In order to increase maximum torque, the low torque level of those dc motors has required to usage of a gearbox, which is selected as Planetary Gear-head GP 26B with $19:1$ gear ratio.

Omniwheels, the interfaces between the driving motors and the ball, are critical to the success of a ballbot. In our ballbot, a special type of omniwheel produced by Nexus Robot is selected with diameter of 100 mm and aluminum structured. The omniwheel has a series of rollers connected with an angle to its circumference. The rotation axes of the rollers are parallel to the circumference of the omniwheel. This arrangement of rollers makes possible the omniwheel to move in both the rotational and the perpendicular direction of the omniwheel. Particularly, the selected omniwheel has smaller rollers in the gaps between the main larger rollers which gives a smooth transition between the rollers. As seen in Figure 4.4, each wheel will directly attach to the shaft of the motors.

FIGURE 4.4: Omniwheel-actuator connection.

FIGURE 4.5: Omniwheel.

4.3 Construction of the Prototype Ballbot

Final structure of the ballbot is assembled by mounting all the components using screws and bolds. In order to place the motors properly and connect the omniwheels to motor shafts, the motor brackets and pins are precisely assembled with specific angle from 7075-T6 aluminum alloy by a CNC machine. This forms a very durable structure, as at least a few crashes were expected during balancing experiments. The rest of design strongly depends on ball size. Following this, the main part of the ballbot, is called body, is constructed using manufactured subcomponents. The structural part of the body consists of three parallel poly methyl methacrylate (PMMA) (also known as Plexiglas) circular plates with 5 mm thickness and aluminum rods to connect these plates to each other. Plates were manufactured on laser cutting machine by considering the the drawings and aluminum rods were cut from sigma profiles with 20 mm thickness. To put it clearly, the lower plate, which the motors will be fixed, will be placed at a specific height to make the zenith angle of the omniwheel to be center of the ball. The inertial measurement unit is rigidly connected to the center of the weight of the body, i.e. strapped down. Later, a data acquisition board equipped with the micro-controller and drivers were connected to the drive mechanism and encoders by flexible cables. In our design the controllers and the drivers are placed separate from the ballbot body. However, in most of the ballbot implementations in the literature, the controllers and drivers are mounted to the main body, which is planned for future work.

The overall constructed ballbot is obtained by including all components as shown in Figure 4.6.

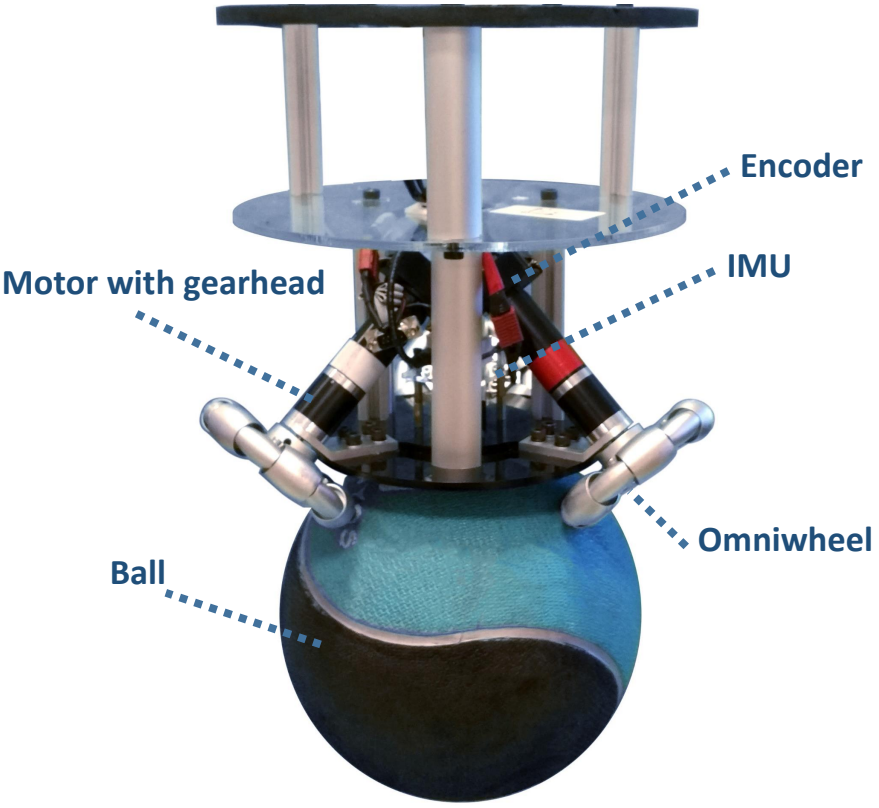

FIGURE 4.6: Final structure of the ballbot.

Overall dimensions of the ballbot are given in Table 4.1.

TABLE 4.1: Dimension of the ballbot and its subcomponents.

Description	**Value / Unit**
Overall ballbot length and width	445 mm - 300 mm
Overall ballbot weight	5400 g
Diameter of the ball	216 mm
Diameter of each omniwheel	100 mm
The height of the center of mass of body	185 mm
Diameter of the lower plate	174.46 mm
Diameter of the upper plates	250 mm
Length of the rods	130 mm - 100 mm
Omniwheel-ball configuration angles	$\alpha = 45°$, $\beta = 120°$

4.4 Data Acquisition Hardware and Drivers

Quanser Q8 is the data acquisition card used with the real-time computer to establish the interface between the computer and the actuators/sensors. It has enough number of Digital/Analog inputs/outputs and encoder inputs with 16-bit resolution.

FIGURE 4.7: Quanser Q8 Data Acquisition Card.

FIGURE 4.8: Maxon LSC 30/2, 4-Q-DC Servo-amplifier in module housing.

Additionally, Maxon LSC 30/2, 4-Q-DC Servo-amplifiers are used in order to drive the dc motors used in the assembly. These drivers are used in the torque mode in our ballbot system, in which the driver takes analog reference signal in between ± 10 V and proportionally generates current in the motor windings, therefore acts as a torque controller. This configuration enables implementation of speed and position loops outside of the servo drive.

The encoders are connected through RS232 on the motor side and 5 pin DIN connector on the data acquisition card side. Furthermore, communication between data acquisition card and the motor drivers is achieved by means of RCA connectors where the outer part of the connector is connected to ground and the inner portion of the connector carries the signal. All the connections to the real-time computer, from both the data acquisition card and IMU, were done via Type A USB 2.0 connector. The power source used for motor drivers is Kesupower S201-24 24 V DC power supply with maximum 8.5 A.

4.5 Sensors and Calibration

In the ballbot, there are two main types of sensors as encoders and inertial measurement unit (IMU). The encoders are selected simply to be compatible with the selected dc motors as Avago HEDS-5540 with resolution up to 1024 counts per revolution. Selection of the IMU was done according to two main criteria where the IMU must be cost-effective and resistant to the external disruptive magnetic sources such as dc motors.

4.5.1 Inertial Measurement Unit

For the present work the MEMS inertial sensor, TinkerForge IMU Brick 2.0 is used (see Figure 4.9), which is very compact and cost-effective inertial measurement unit that equipped with a 3-axis accelerometer, magnetometer (compass) and gyroscope with 100Hz sampling rate. Its main advantages are reduced cost, cost-effective resolution, easier recalibration and to be directly readable by USB. Besides being cheap and cost-effective, it has a wide range of programming interface tools.

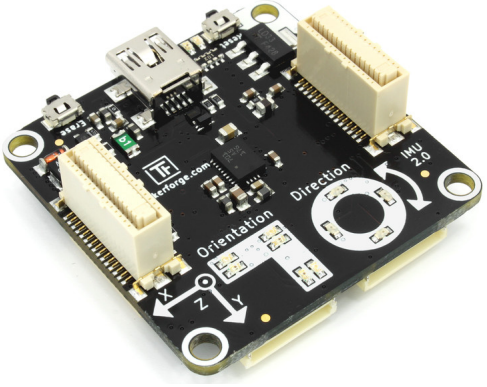

FIGURE 4.9: IMU Brick 2.0

4.5.2 Calibration and Tuning

Calibration procedure for the inertial measurement unit is done using corresponding software as follows:

- Gyroscope: The IMU is placed in a single stable position for a period of several seconds.
- Accelerometer: The IMU is placed in six different stable position for a period of a few seconds by using slow movement between stable positions where at least one position that is perpendicular to the inertial x, y and z axis.
- Magnetometer: Calibration procedure is done by calibrating magnetometer by making random movements (e.g. write a figure 8 in air).

It is necessary to readjust and tune the Kalman filter when it is associated with the experimental ballbot system. The process noise covariance Q and the measurement noise covariance R are main parameters for the tuning. Fortunately, the selection of R is relatively easy that can be calculated from a variance of sample series of measurement. Q is a covariance matrix associated with the noise in states but finding out Q is not so obvious. It is generally determined intuitively but there are some points that need to be regarded choosing it, that is why it is done by trial and error.

4.6 Real-time Control and Monitoring Software

A PC with D2700 dual core 2.13 GHz processor and 2 GB RAM, running under Microsoft Windows 7 operating system is used for both the real-time control and signal monitoring. Visual Studio™ software is used to create a kernel for real-time

applications on a windows based system and enabling the user to monitor the signals. Also, Matlab/Simulink™ is used for data inspection and creates plots from saved data. Additionally, in order to access to the IMU brick 2.0 and Quanser Q8, Brick Deamon v2.2.2 and Quarc 2.4 softwares are installed.

Chapter 5

Modeling and Control of the Ballbot

Nonlinear dynamics of the ballbot and control schemes are briefly described in this chapter. Planar system modeling has been commonly used in the initial studies of the ballbot by dividing the 3D system into the independent planar models by neglecting coupling effects between these models [29, 30, 32, 33, 38]. Be that as it may, it is still valuable to derive the 2D models to get a subjective comprehension of the dynamics of a ballbot. In this thesis, all coupling effects are taken into account in the 3D mathematical model of the ballbot.

5.1 Model Description

Generally, a ballbot consists of two main components: a spherical ball and an upper body placed on top of it. The model used in this thesis is three omniwheeled morphology where the body includes three single-row omniwheel driven by three independent dc motors which are placed with β degree between each other at the

bottom as presented in Figure 5.2. In this arrangement, each omniwheel has a contact point with the ball, as shown in Figure 5.1, where the contact points are located at the zenith angle, α.

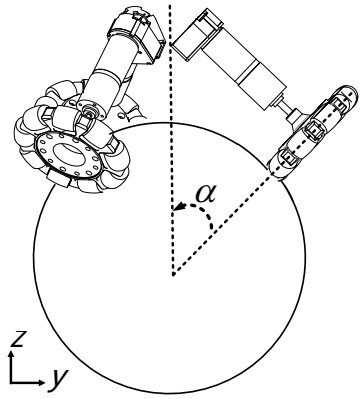

FIGURE 5.1: Side view.

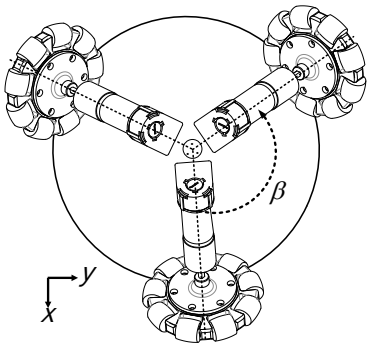

FIGURE 5.2: Top view.

FIGURE 5.3: Ballbot-omniwheels configuration.

It is assumed that omniwheels can only apply forces to the ball in the tangential direction of its own rotation in the body reference system. Omniwheels are wheels with small rollers around the periphery which are perpendicular to the turning direction, where the ball and the body interact with each other through those omniwheels.

5.1.1 Inputs and Outputs

The inputs of the system are the three control torques $\tau = \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 \end{bmatrix}^T$ which are generated by the motors and transformed by the omniwheels to the ball and external disturbances $\tau_d = \begin{bmatrix} \tau_{d_x} & \tau_{d_y} & \tau_{d_z} \end{bmatrix}^T$ acting on the body.

The system has ten outputs which include the body angular position $\Theta_b = \begin{bmatrix} \phi_b & \theta_b & \psi_b \end{bmatrix}^T$ and velocity $\Omega_b = \begin{bmatrix} \dot{\phi}_b & \dot{\theta}_b & \dot{\psi}_b \end{bmatrix}^T$, and the ball angular position $\Theta_k = \begin{bmatrix} \phi_k & \theta_k \end{bmatrix}^T$ and velocity $\Omega_k = \begin{bmatrix} \dot{\phi}_k & \dot{\theta}_k \end{bmatrix}^T$. Since the system has several inputs and outputs, it can be regarded as a Multiple Input, Multiple Output (MIMO) system.

5.1.2 Assumptions

In order to derive the complete dynamical model of the ballbot, several assumptions have to be made. In literature, existing works [32, 33, 38] has assumptions which are commonly based on the same principles as listed below.

In our model, the 3D mathematical model of the ballbot is derived by taking into account all coupling effects under the following assumptions:

- The model consists of five rigid bodies (see Figure 5.5):
  - 1 body with dc motors and gear heads
  - 1 spherical ball
  - 3 omniwheels
- The ball moves only horizontally in a pure rolling motion without jumping.
- Contacts between the ball and omniwheels are assumed to be slippage-free.
- Contact between the ball and floor is assumed to be slippage-free.
- Omniwheels are assumed to be simplified without rolling or kinetic friction.
- The floor is assumed to be perfect flat without any deformation.

5.1.3 Coordinates

Coordinate frames are used to describe the position of the ball and the orientation of the body. As seen in Figure 5.4, the inertial reference frame is denoted by N, middle reference frame at the center of the ball M, ball fixed frame as K, and finally the body fixed frame as B.

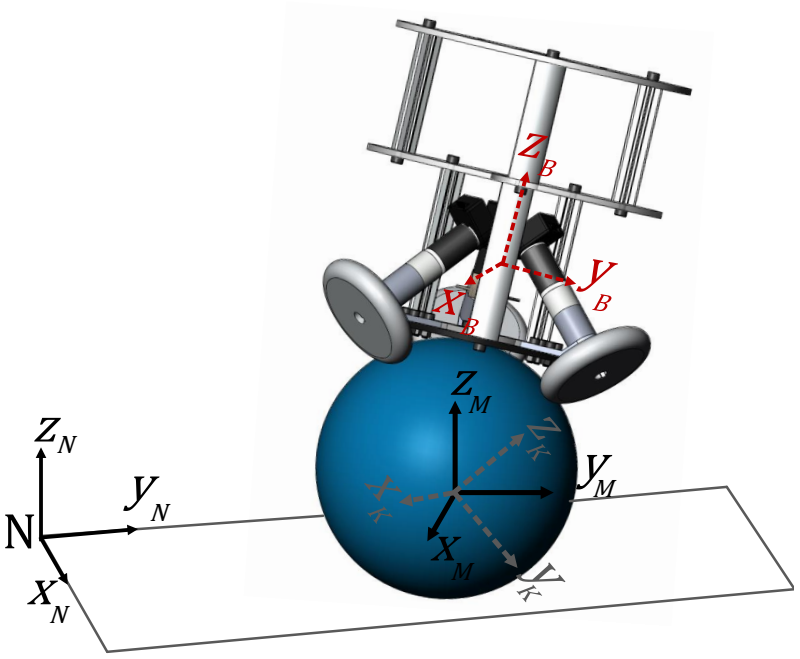

FIGURE 5.4: Coordinate systems and body frames.

The representations of the rotation between coordinate frames are defined by using three dimensional attitude parameterizations, i.e., Euler angles. The system can be described with five DoF (Degrees of Freedom). 3 DoF for the body rotation $\Theta_b^N = (\phi_b, \theta_b, \psi_b)$ represent the roll, pitch and yaw angles of the body respectively with respect to N. 2 DoF for the the ball angles $\Theta_k^M = (\phi_k, \theta_k)$ with respect to fixed frame M at the center of the ball. Since the working range of the ballbot platform pitch angle will not reach to $\theta_b = \pm\frac{\pi}{2}$, the singularity problem associated with Euler angle parametrization does not have to be considered.

5.2 Dynamic Equations

The dynamics of the ballbot can be derived using the Euler-Lagrange approach simply by calculating the kinetic and potential energy of each body which has previously been performed by [29, 30, 39]. In order to derive the Euler-Lagrange equations of motion, a minimal coordinates q is selected to include following set of variables that fully defines the system:

$$q = \begin{bmatrix} \phi_b & \theta_b & \psi_b & \phi_k & \theta_k \end{bmatrix}^T \tag{5.1}$$

This method defines the Lagrangian $L(q, \dot{q})$ which is simply the sum of the kinetic energies minus the potential energy of the system, as shown in (5.2).

$$L(q, \dot{q}) = T_b + T_k + T_{w_1} + T_{w_2} + T_{w_3} - V_b \tag{5.2}$$

Where T_b and T_k represent the sum of the translational and rotational kinetic energies of the body and the ball respectively. T_{w_i} represents rotational kinetic energies of omniwheels and motors, and V_b is the sum of the potential energy of the body includes motors and omniwheels.

5.2.1 Energy Calculations

For each rigid body, translational, rotational and potential energies must be calculated. The reference plane with zero potential energy is assumed to be at the center of the ball. The numerical results of these calculations are only valid if the dynamics are a very good approximation of the actual ballbot dynamics. On account of this, the physical parameters of the ballbot are presented in Table 5.1 and Figure 5.5 shows the corresponding rigid bodies, where these parameters are selected to be consistent with sizing and material selection of our ballbot design in Chapter 4.

The mass and moment of inertia of the rigid parts of the ballbot are calculated and verified from the CAD model mentioned in Section 4.1 and relevant data sheets.

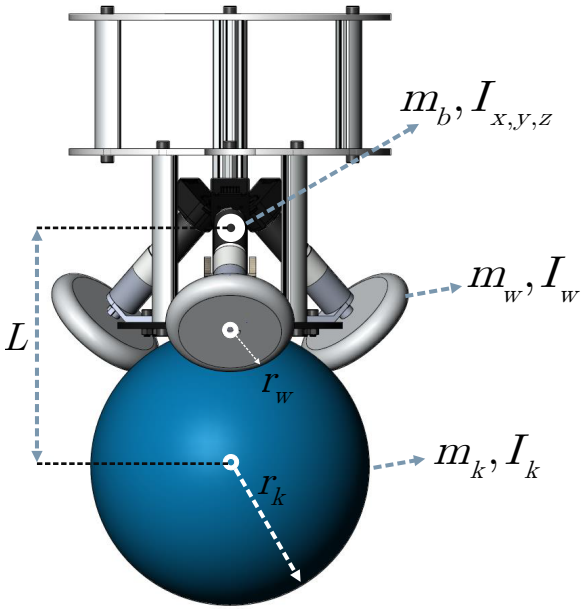

FIGURE 5.5: Parameters of the ballbot.

Body

The kinetic energy of the body consists of a translational, a coupling and a rotational part. The kinetic energy of the body is defined as:

$$T_b = \underbrace{\frac{1}{2} m_b v_b^2}_{\text{translational}} + \underbrace{m_b v_b \left(\omega_b \times \overrightarrow{L}\right)}_{\text{coupling}} + \underbrace{\frac{1}{2} I_{x,y,z} \omega_b^2}_{\text{rotational}} \tag{5.3}$$

where $\overrightarrow{L}$ denotes the vector from the center of the ball to the center of the body. The potential energy of the body is calculated as follow

$$V_b = -m_b \overrightarrow{g}\, \overrightarrow{L} \quad \text{for} \quad \overrightarrow{g} = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T \tag{5.4}$$

At this point it is important to note that above potential energy includes the potential energies of motors and omniwheels.

TABLE 5.1: Definitions and values of all the parameters used for modeling.

Parameter	**Description**	**Value**
m_b	Mass of the body	2.39615 kg
m_k	Mass of the ball	3 kg
m_w	Mass of each omniwheel	0.25 kg
L	The height of the center of mass of body	0.185 m
r_k	Radius of the ball	0.108 m
r_w	Radius of each omniwheel	0.05 m
I_x, I_y	Inertia of the body around x and y axis	0.10354 kgm^2
I_z	Inertia of the body around z axis	0.020929 kgm^2
I_k	Inertia of the ball	0.019814 kgm^2
I_w	Inertia of omniwheels	0.003329 kgm^2
α	Omniwheel-ball configuration angle	45°
β	Angle between omniwheels	120°

Ball

The kinetic energy of the ball consists of a translational and a rotational part. The kinetic energy of the ball is defined as:

$$T_k = \underbrace{\frac{1}{2} m_k v_k^2}_{\text{translational}} + \underbrace{\frac{1}{2} I_k \omega_k^2}_{\text{rotational}} \tag{5.5}$$

Since the ball moves horizontally, its potential energy is always zero, i.e. $V_k = 0$.

Wheels

Lastly, rotation of the omniwheels and the motor shafts around their axis is considered in the following kinetic energies

$$T_{w_i} = \frac{1}{2} I_w \omega_{w_i}^2 \quad \text{for} \quad i = 1,\ 2,\ 3 \tag{5.6}$$

5.2.2 Equation of Motion by Euler-Lagrange Derivation

The equations of motion are derived by solving the Lagrange equation in time t to find the torques that directly control the minimal coordinates τ_q which are related to the motor torques τ with kinematic relations.

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}}\right)^{T} - \left(\frac{\partial L}{\partial q}\right) = \tau_q\left(q, \tau\right) \tag{5.7}$$

Overall, resulting in non-linear equations of motion in compact form is obtained as

$$M\left(q\right)\ddot{q} + C\left(q, \dot{q}\right)\dot{q} + G(q) = \tau_q \tag{5.8}$$

At this point it is important to note that these calculations could not be done directly. All energy equations described above are calculated, as well as solution for the Lagrange equation is found by using a dynamic symbol manipulation software package Autolev™ [45]. The equations of motion for the system described above are obtained from software as a set of nonlinear, differential equations consist of many terms. Therefore, the overall Autolev model and detailed explanations are referred in Appendix B.

5.3 Control Schemes

In this section, the design of a balancing controller is discussed by using two different approaches: cascaded acceleration feedback (AFB) control and conventional Proportional-Derivative (PD) control for the system developed in previous section. Also a brief description of the position controller is provided.

5.3.1 Balancing Control

The balancing controller is obviously the most important controller for the ballbot platforms due to their inherently unstable and profoundly nonlinear under-actuated structure. The balancing controller maintains the body at reference body angles. In most of the applications, the body reference angles are zero for achieving complete balancing task. It is unfortunate that the ballbot orientation angles are not directly actuated, but the ballbot accomplishes the orientation tracking circuitously by actuating the ball with desired torques. These desired control torques may be considered as outputs of the virtual actuating wheels.

In the case of three omniwheeled mechanism, so as our situation, since the controllers do not control the omniwheels directly, instead control virtual wheels that exist in the same plane as the sensors. A kinematic conversion is need to be done to find corresponding torques of the omniwheels, which will be discussed later in Section 5.4.

5.3.1.1 Balancing Control: Acceleration Feedback Approach

In this work, the inner acceleration control approach detailed in [21] is implemented as part of cascaded position, velocity and current control loops as depicted in Figure 5.6. The goal of acceleration control is to improve the stabilization performance of the ballbot system by rejecting external disturbances. The position loop produces reference signals for the velocity loop, which in turn creates reference signals for the current loop. The success of stabilization control largely depends on reliable acceleration feedback, noise-adjusted angular accelerations, provided by sensor fusion algorithm, are used as feedback in the inner current control loops.

In control loops following notations are used to describe motion vectors, $\Theta = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ represent Euler angles, Euler rates described by $\Omega = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$, and Euler accelerations are denoted by $\Gamma = \begin{bmatrix} \ddot{\phi} & \ddot{\theta} & \ddot{\psi} \end{bmatrix}^T$. The block diagram of the AFB

controller is depicted in Figure 5.6, Θ_{ref} is the reference attitude for balancing controller. Additionally, τ_d is the unknown disturbance acting directly on body and K_γ is the acceleration feedback gain.

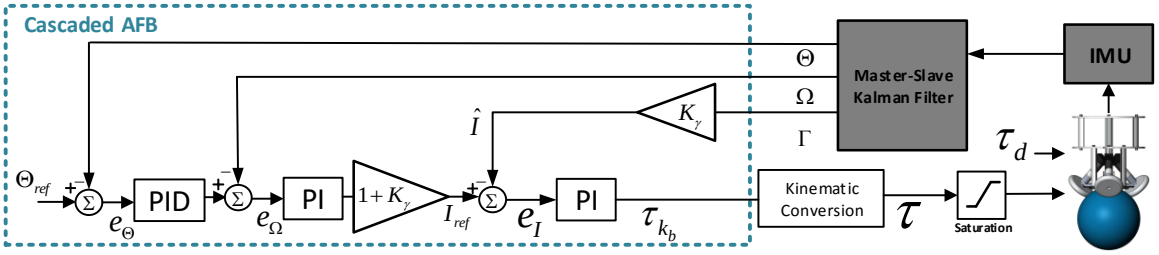

FIGURE 5.6: Block diagram of the balancing controller with cascaded AFB.

Higher acceleration gain, K_γ, adds more electronic inertia to the physical inertia of the total system. Thus, the overall system exhibits high dynamic stiffness and has better disturbance rejection. The increase in effective inertia reduces the speed of the system's response. In order to preserve the loop gain, one should scale up the control loop gains by the factor $(1 + K_\gamma)$ in (5.10).

The following PID controllers generate reference velocities for the velocity control loops, for the orientation errors are defined as e_Θ:

$$\Omega_{ref} = K_{p_\Theta} e_\Theta + K_{i_\Theta} \int e_\Theta dt + K_{d_\Theta} \dot{e}_\Theta \tag{5.9}$$

Velocity and current controls are designed as PI controllers and reference currents are generated as follows, where e_Ω represents the velocity errors in the Euler rates.

$$I_{ref} = (1 + K_\gamma) K_{p_\Omega} e_\Omega + (1 + K_\gamma) K_{i_\Omega} \int e_\Omega dt \tag{5.10}$$

Herewith, the balancing torque is calculated as

$$\tau_{k_b} = K_{p_I} e_I + K_{i_I} \int e_I dt \tag{5.11}$$

where the acceleration feedback comes into picture and current error is defined as:

$$e_I = I_{ref} - \hat{I} \tag{5.12}$$

$$\hat{I} = K_\gamma \Gamma \tag{5.13}$$

5.3.1.2 Balancing Control: Conventional PD Control

In conventional PD controller [1, 41], a ballbot is controlled by two PD controllers, that control the pitch and roll angles, where the problem of stabilization is reduced to one regulation problem. For the orientation errors are defined as e_Θ, the balancing torque is calculated as:

$$\tau_{k_b} = K_{p_\Theta} e_\Theta + K_{d_\Theta} \dot{e}_\Theta \tag{5.14}$$

The block diagram of the PD controller is depicted in Figure 5.7.

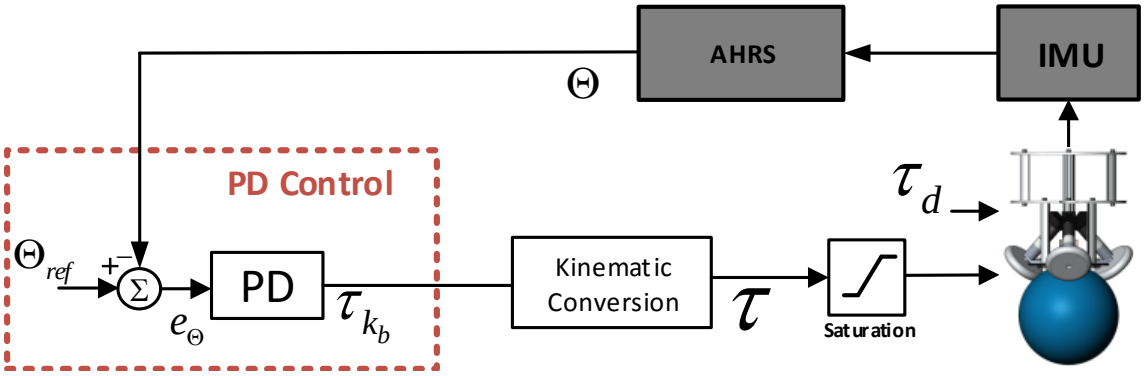

FIGURE 5.7: Block diagram of the balancing controller with conventional PD.

5.3.2 Position Control

In the proposed approach, the position controller is constructed according to the absolute position of the ball in the inertial coordinates. Tracking control is mainly focused on the controlling the position and velocity of the ball center. Since the ball moves only horizontally in a pure rolling motion without slippage between the ball and the ground, the horizontal positional dynamics of the ball in the inertial coordinates can be written in terms of ball angular velocity as

$$v = r_k \begin{bmatrix} s_{\psi_b} & c_{\psi_b} \\ c_{\psi_b} & s_{\psi_b} \end{bmatrix} \Omega_k \tag{5.15}$$

where $\Omega_k = \begin{bmatrix} \dot{\phi}_k, \dot{\theta}_k \end{bmatrix}^T$ is angular velocity of the ball, r_k is the radius of the ball, and c_{θ_i} and s_{θ_i} represent cosine and sine of the θ in frame i, respectively. By taking time derivative of both side and rearranging to find angular acceleration of the ball as $\Gamma_k = \begin{bmatrix} \ddot{\phi}_k, \ddot{\theta}_k \end{bmatrix}^T$,

$$\Gamma_k = \frac{1}{r_k} \begin{bmatrix} s_{\psi_b} & -c_{\psi_b} \\ c_{\psi_b} & s_{\psi_b} \end{bmatrix} a - \begin{bmatrix} 0 & c_{\psi_b}^2 - s_{\psi_b}^2 \\ 1 & -2c_{\psi_b} s_{\psi_b} \end{bmatrix} \Omega_k \dot{\psi}_b \tag{5.16}$$

Desired acceleration of the ballbot is constructed from the outputs of position controller as in the typical motion control systems contain cascaded position and velocity loops. The position loop produces reference signals for the velocity loop, which in turn creates desired acceleration a_{ref} signal for the system defined in the inertial frame. Position and velocity controls are designed as PID and PI respectively.

$$v_{ref} = K_{p_p} e_p + K_{i_p} \int e_p dt + K_{d_p} \dot{e}_p \tag{5.17}$$

$$a_{ref} = K_{p_v} e_v + K_{i_v} \int e_v dt \tag{5.18}$$

where e_p and e_v represent the position and velocity errors, respectively. K_p, K_i and K_d are the proportional, integral and derivative control gains for corresponding variables. Then, desired torques on the ball coming from tracking controller can be calculated after doing the transformation detailed above and multiplying by the inertias of the ball, I_k, as

$$\tau_{k_p} = I_k \Gamma_k \tag{5.19}$$

The main block diagram of the position controller with balancing controller is depicted in Figure 5.8.

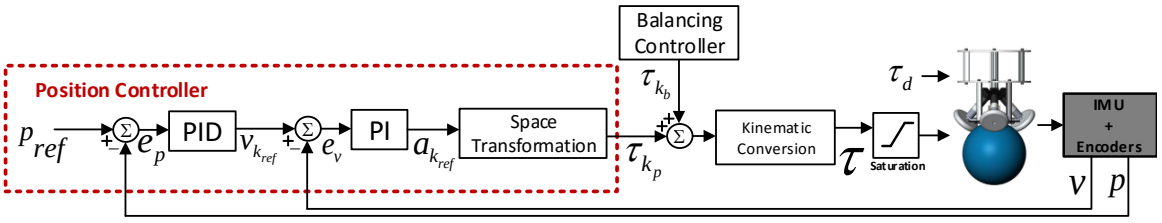

FIGURE 5.8: Block diagram of the position controller.

5.4 Torque Conversion

In this section a kinematic relation between the control torques of the virtual actuating wheels and the torques of the real omniwheels is provided. This relation can be derived using the conservation of the resulting torque on the body [1, 38]. Above calculated torques are actuating the ball like a virtual wheel. The real system has three motors. In order to be able to control the real system, the torques on the virtual motors have to be converted into the torques for the real motors. A simple mathematical relationship can be used to convert the signals from the virtual wheels to the actual wheels using the angular configuration in Figure 5.3.

The command torques of virtual wheels, τ_{k_i}, derived from proposed linear controllers were converted into the torques of the three real wheels, τ_i, as follows:

$$\tau = \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{3}\left(\tau_{k_3} + (\tau_{k_1}\cos\beta - \tau_{k_2}\sin\beta)\frac{2}{\cos\alpha}\right) \\ -\frac{1}{3}\left(\tau_{k_3} + \left(\sin\beta(-\sqrt{3}\tau_{k_1} + \tau_{k_2}) - \cos\beta(\tau_{k_1} + \sqrt{3}\tau_{k_2})\right)\frac{1}{\cos\alpha}\right) \\ \frac{1}{3}\left(\tau_{k_3} + \left(\sin\beta(\sqrt{3}\tau_{k_1} + \tau_{k_2}) + \cos\beta(-\tau_{k_1} + \sqrt{3}\tau_{k_2})\right)\frac{1}{\cos\alpha}\right) \end{bmatrix}$$

where α and β are the angles of the motor configuration which values given in Table 5.1.

Chapter 6

Simulation and Experimental Results

The following sections describe and discuss the verification of the derived fusion and control algorithms using both simulations and experimental ballbot system. Initially, in Section 6.1, the algorithms were programmed in Matlab/Simulink™ for simulation purpose. Subsequently, Section 6.2 deals with the implementation of the filter equations with real IMU data and control algorithms on the experimental ballbot system.

6.1 Simulation Results

A simulation model of the actual system is built in order to be able to rapidly develop and tune the controllers and also test them without actually running the experimental ballbot system, therefore avoiding the overhead associated with setting up and calibrating the system and the test equipment each time. Matlab/Simulink™ is used for the simulation model, where the proposed fusion algorithm is incorporated

into a simulator, which includes sensor models, to test the stabilization performance of the ballbot model.

6.1.1 Sensor Fusion Simulator

Accuracy of the designed algorithm highly depends on the sensor quality, especially time-varying error characteristics. In order to be able to evaluate the filter, data from inertial sensors were needed to construct a high fidelity simulation model, therefore realistic sensor dynamics models given in Section 3.1 are utilized. Biases and noises that corrupt sensor outputs are tabulated in Table 6.1 where SNR denotes signal to noise ratio.

Table 6.1: Sensor simulator parameters.

Parameter	**Description**	**Value**
b_g	Gyro bias	$[1 \ -1 \ 0.5]^T \ deg/s$
b_a	Accelerometer bias	$[0.01 \ -0.01 \ 0.005]^T \ g$
b_m	Magnetometer bias	$[0.01 \ -0.01 \ 0.02]^T \ \mu T$
S_{η_g}	SNR of gyro	29.31 dB
S_{η_a}	SNR of accelerometer	25.45 dB
S_{η_m}	SNR of magnetometer	40 dB

Hereby, in this implementation, true Euler angles generated based on unbiased and noise-free gyro data by using transformation matrix $\mathbb{B}$ (3.10) that transforms body angular rates to Euler rates, then random noise and constant biases are included to generate sensor measurements from the true data.

In Matlab environment, master and slave Kalman filters are modeled independently. Both algorithms are compiled in the same recursive loop to work simultaneously. Initial conditions for all states are selected as random number. After several implementations and configuration adjustments, results in Section 6.1.3 are obtained for generated simulation data.

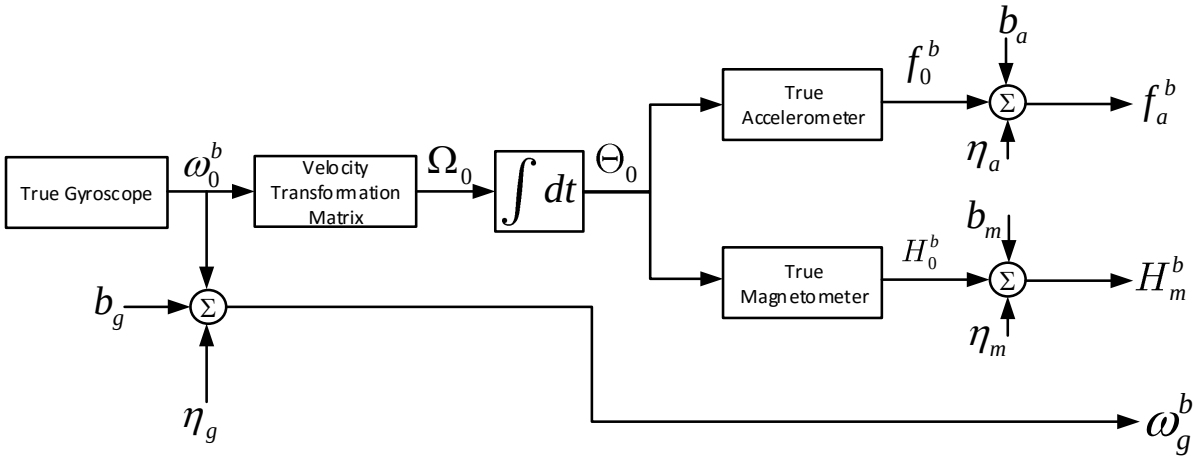

FIGURE 6.1: Block diagram of the IMU simulator.

Numerical values for noise processes have been determined iteratively, instead of analyzing them empirically. The presented results of simulation in this thesis were achieved by setting the system covariance matrices of the master-slave Kalman filter as follows:

The process and the measurement covariance matrices of the slave filter are

$$Q_{s_{9\times 9}} = diag\left(\begin{bmatrix} 10^{-6} & 10^{-6} & 10^{-6} & 10^{-5} & 10^{-5} & 10^{-5} & 10^{-4} & 10^{-4} & 10^{-4} \end{bmatrix}\right)$$
$$R_{s_{3\times 3}} = diag\left(\begin{bmatrix} 10 & 10 & 10 \end{bmatrix}\right)$$

The process and the measurement covariance matrices of the master filter are

$$Q_{m_{12\times 12}} = diag\left(\begin{bmatrix} 10^{-2} & 10^{-2} & 10^{-2} & 10^{2} & 10^{2} & 10^{2} & 10^{3} & 10^{3} & 10^{3} & 10^{-8} & 10^{-8} & 10^{-8} \end{bmatrix}\right)$$
$$R_{m_{9\times 9}} = diag\left(\begin{bmatrix} 0.962 & 0.962 & 0.962 & 0.687 & 0.1 & 0.1 & 0.1 & 0.003 & 0.003 & 0.003 \end{bmatrix}\right)$$

Noise-free and unbiased gyro data selected as quadratic polynomial to satisfy slowly varying damped sinusoidal angular jerk assumption. All data were generated at 1 kHz frequency for 40 seconds duration.

True angular motion values in body axes are generated using the bounded harmonic wavelets as

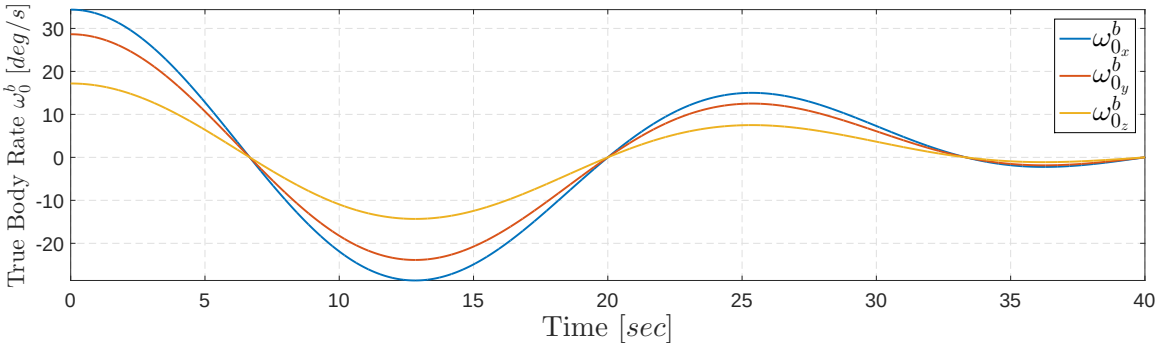

FIGURE 6.2: True angular velocity in body frame.

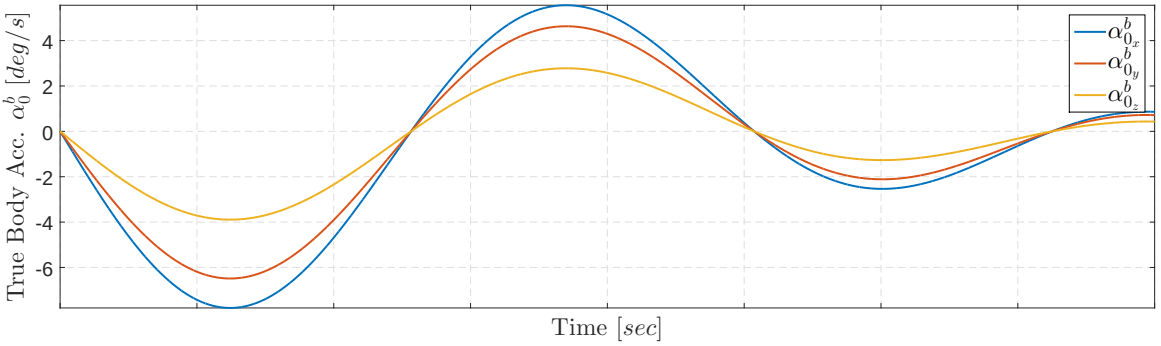

FIGURE 6.3: True angular acceleration in body frame.

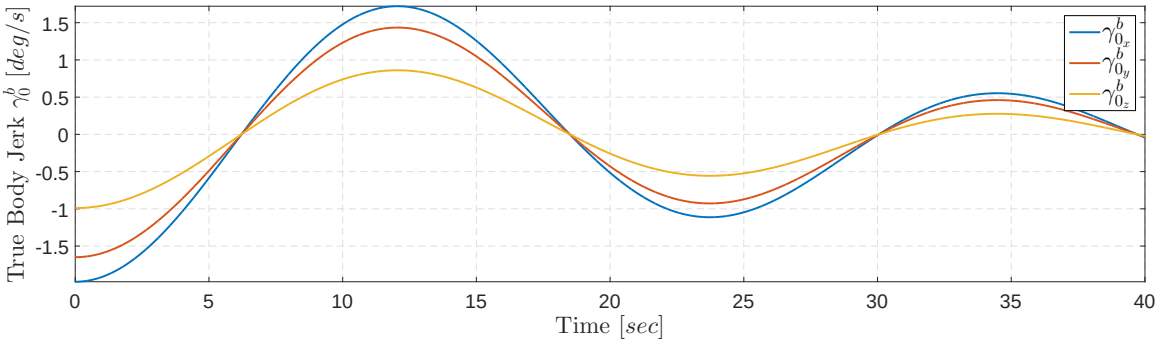

FIGURE 6.4: True angular jerk in body frame.

and, true Euler angles, rates and accelerations are formed by assuming initial positions as 0.

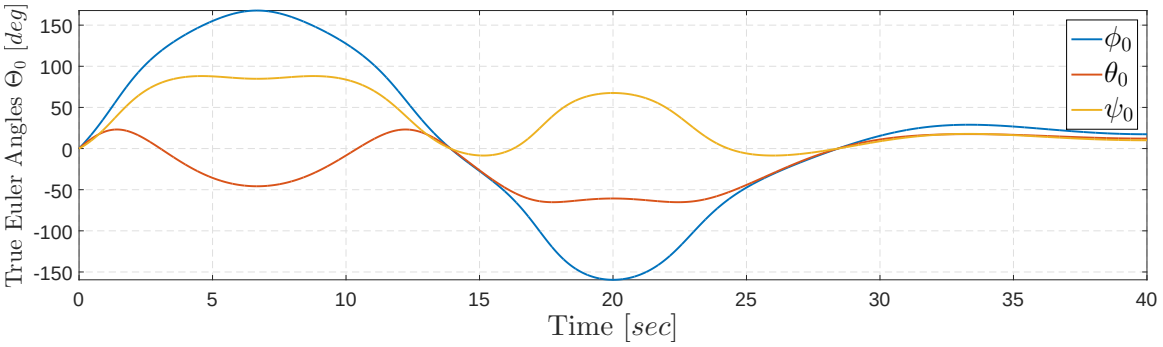

FIGURE 6.5: True Euler angles in inertial frame.

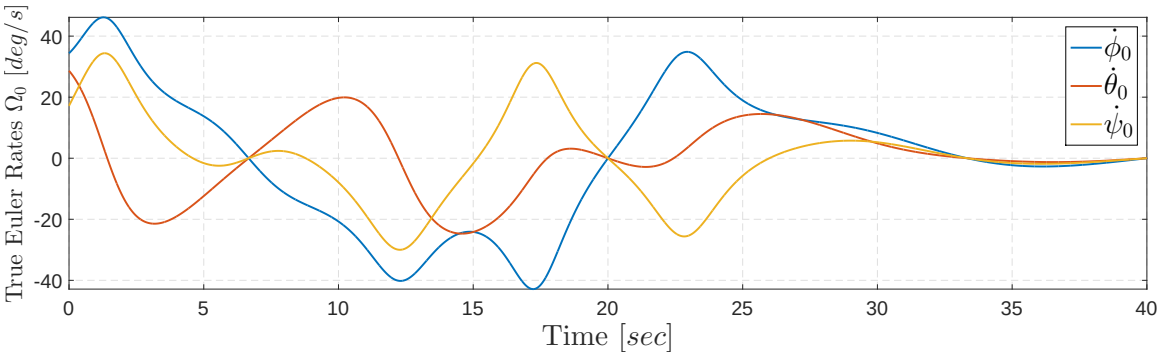

FIGURE 6.6: True Euler rates in inertial frame.

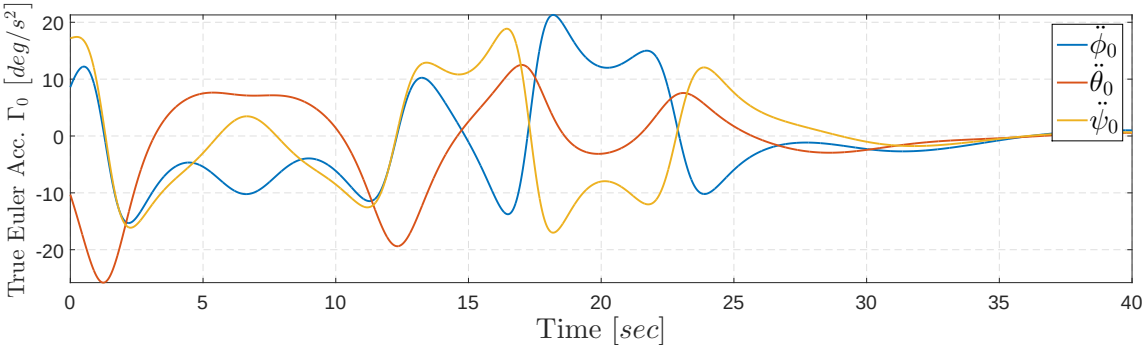

FIGURE 6.7: True Euler accelerations in inertial frame.

Hereby, aiding sensor measurements are generated from true Euler angles

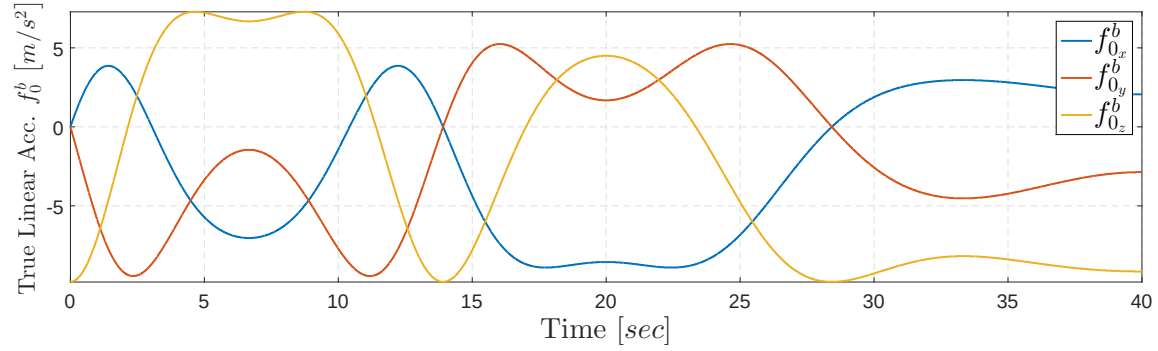

FIGURE 6.8: True linear acceleration in body frame.

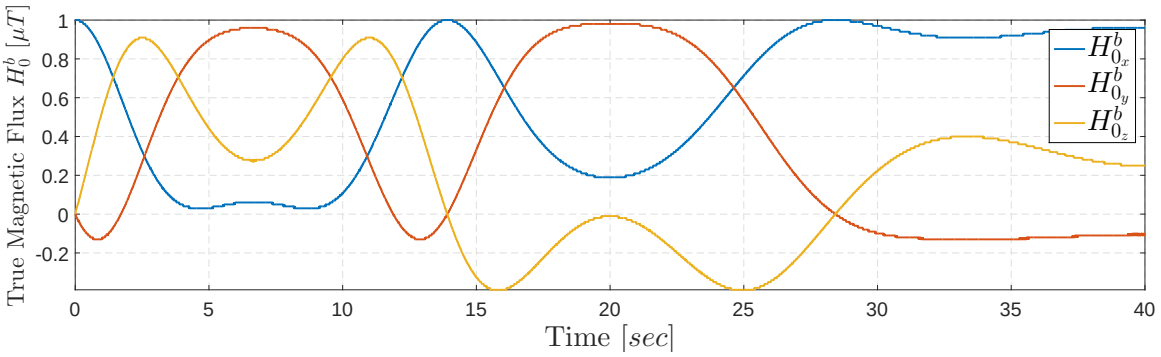

FIGURE 6.9: True magnetic flux in body frame.

After obtaining clean measurement data, random noise and constant biases are included to generate sensor measurements from the true data as follows.

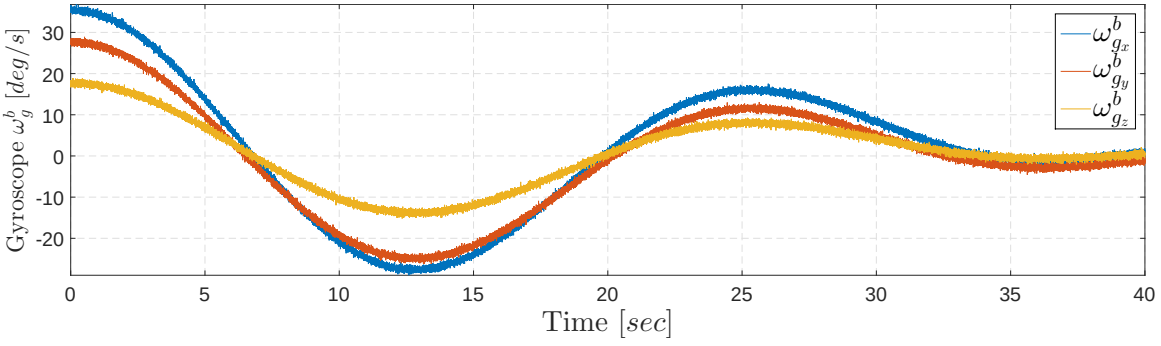

FIGURE 6.10: Generated gyroscope measurement by IMU simulator.

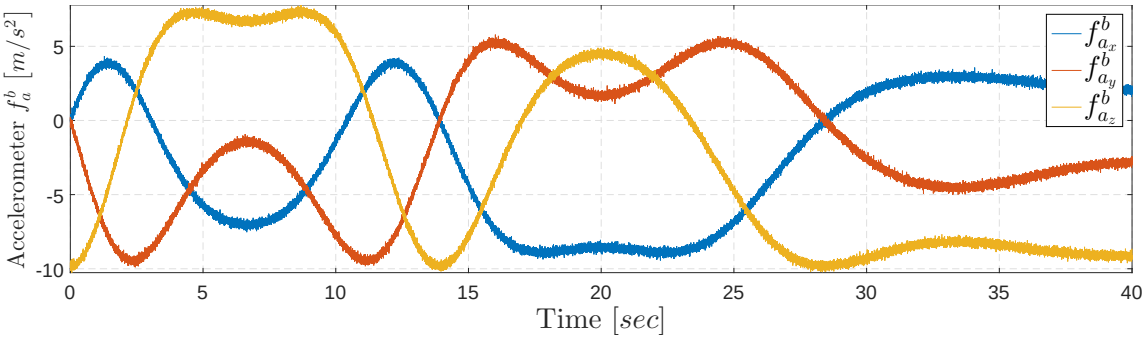

FIGURE 6.11: Generated accelerometer measurement by IMU simulator.

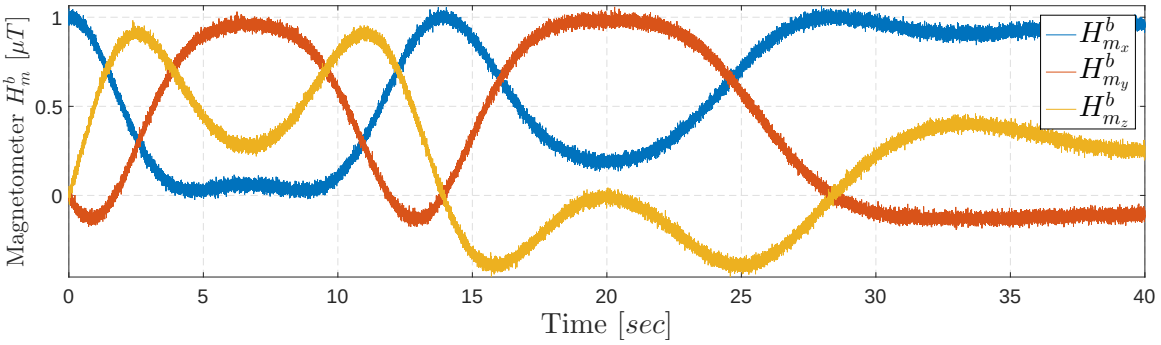

FIGURE 6.12: Generated magnetometer measurement by IMU simulator.

6.1.2 Ballbot Simulator

Finally, calculations of the equations of motion, derived in Section 5.2.2 by solving the Lagrange equation, are performed by a dynamic symbol manipulation software package Autolev™ [45]. Because of the size of the equations of motion, we used generated C code to build a MEX function based MATLAB function. In this way, the execution speed of simulation process is increased. The speedup is around 25 times using the generated MEX over the baseline MATLAB function. To make sure that calculated torques are in the range of the ballbot capability, the saturation block is used to bound torque inputs to the system. It is important to note that, the simulator does not include friction and/or backlash terms that are present in real ballbots. The physical parameters of the ballbot for the simulation purpose are

presented in Table 5.1, where these parameters are calculated from a CAD software model by considering appropriate sizing and material selection of our ballbot design.

The performance of the proposed sensor fusion algorithm is evaluated by using the estimated Euler angles, velocities and accelerations as feedback signals in the stabilization control of the ballbot simulator. For the PD and AFB controllers, controller gains are tuned manually by considering convergence and stability properties of system trajectories. Additionally, all integrations are realized as numerical integration in forward Euler form.

6.1.3 Simulation: Sensor Fusion Results

For the slave Kalman filter, the bias compensated angular velocities are used as measurement. As seen in Figure 6.13, estimated angular velocities were smoothed and unbiased version of gyroscope readings.

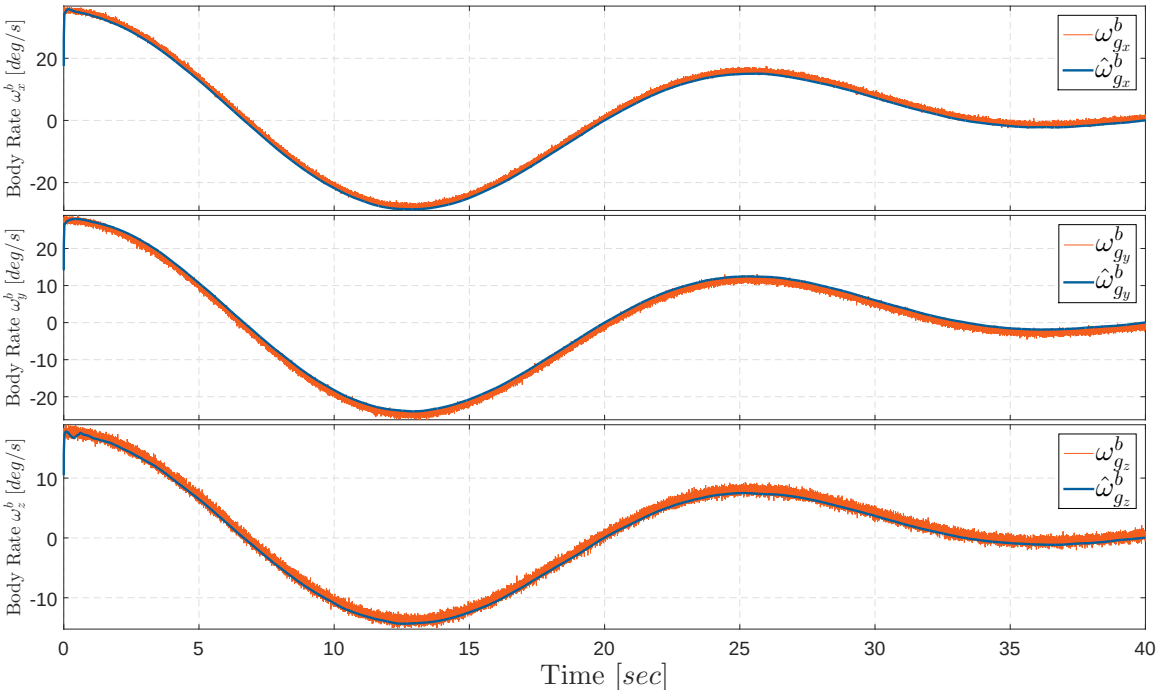

FIGURE 6.13: Estimated angular velocity in the body coordinate system.

The slave filter has a high accuracy in body angular acceleration estimation. Estimated body accelerations are compared with calculated true body angular acceleration in Figure 6.14.

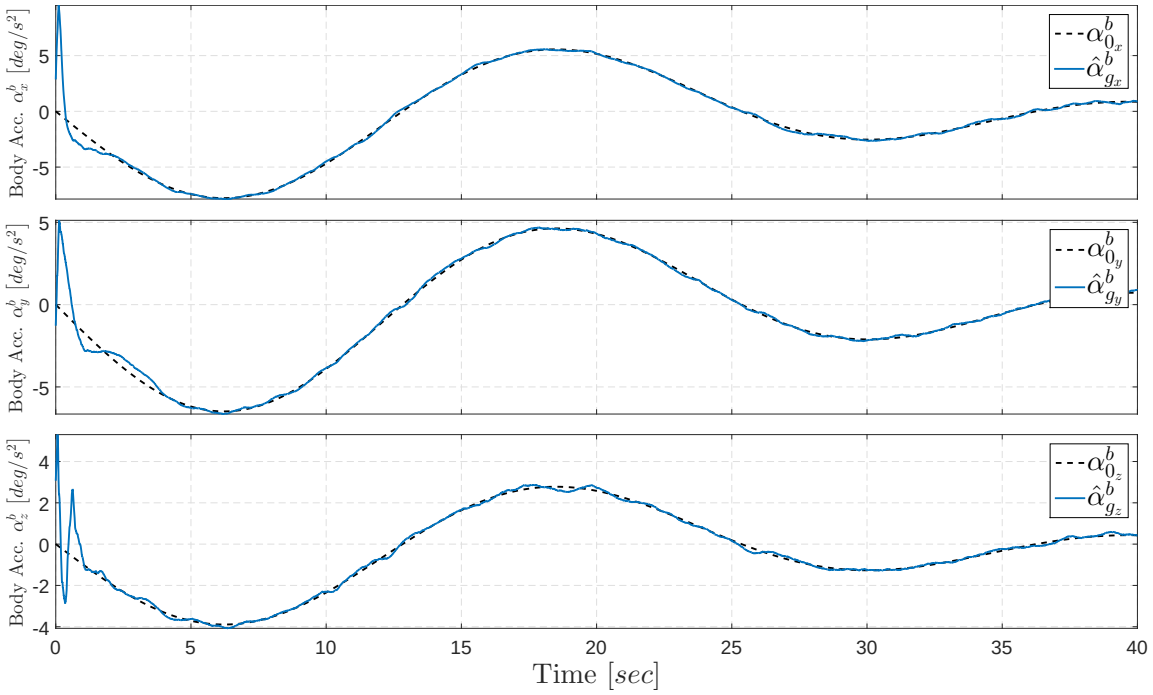

FIGURE 6.14: Estimated angular acceleration the body coordinate system.

TABLE 6.2: RMS and maximum values of body angular acceleration estimation errors after $t = 5$ sec.

Body Angular Acceleration	**RMS Error** $[deg/s^2]$	**Maximum Error** $[deg/s^2]$
α_{g_x}	0.105	0.285
α_{g_y}	0.102	0.237
α_{g_z}	0.132	0.243

Slave Kalman filter estimated angular acceleration in body frame which is consistent with the true acceleration trajectory where the maximum and rms of the estimation error shown in Table **??**.

Also, estimated body jerks are compared with calculated true body angular jerks in Figure 6.15. It can be seen from both estimated body angular jerks $\hat{\gamma}_g^b$ results that the slave estimator provided a fast convergence rate with high accuracy.

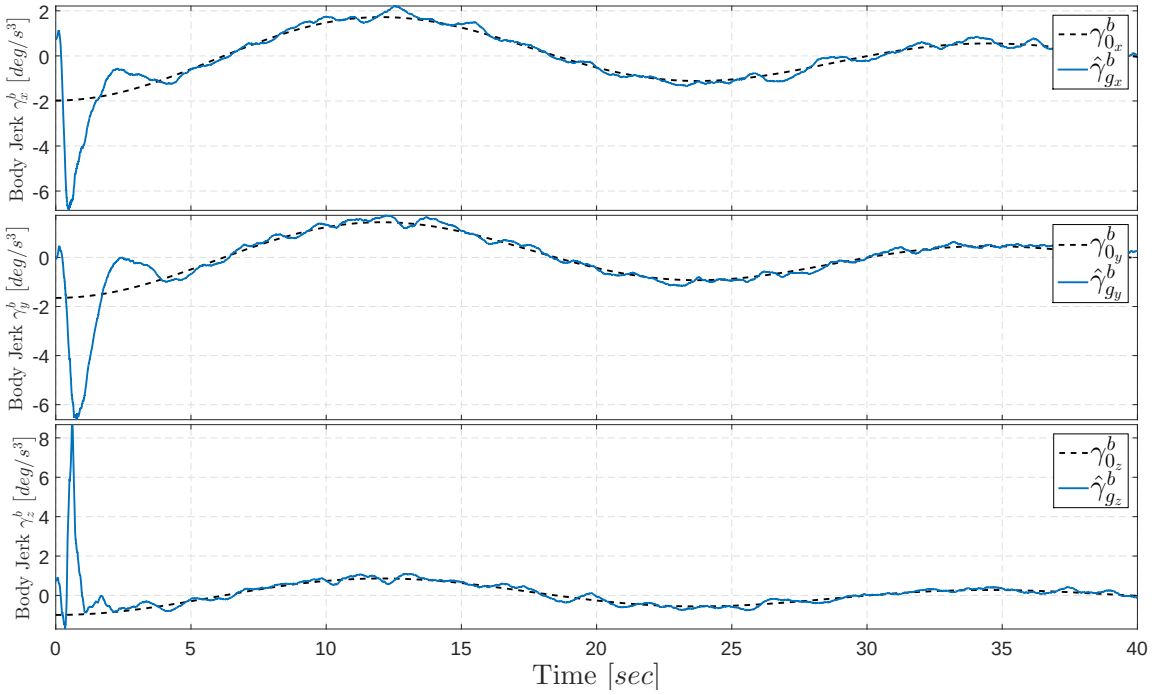

FIGURE 6.15: Estimated angular jerk the body coordinate system.

TABLE 6.3: RMS and maximum values of body angular jerk estimation errors after $t = 5$ sec.

Body Angular Jerk	**RMS Error** $[deg/s^3]$	**Maximum Error** $[deg/s^3]$
γ_{g_x}	0.157	0.507
γ_{g_y}	0.146	0.345
γ_{g_z}	0.141	0.361

Before going further, let us see how bad attitude results can be obtained from directly gyroscope measurements. Body rate measurements are multiplied with transformation matrix $\mathbb{B}$ and integrated to show drifted results of direct integration of gyroscope data. There are noticeable drifts in Euler angles as seen in Figure 6.16.

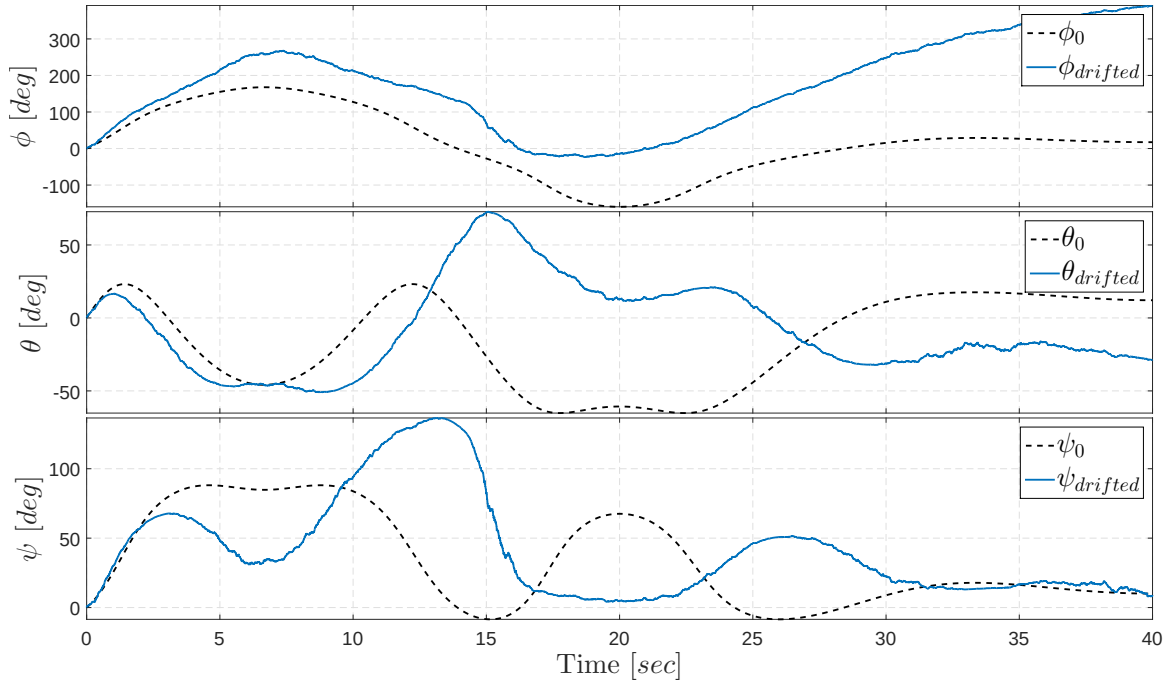

FIGURE 6.16: Comparison of drifted and true Euler angles.

It is clear that, without aiding sensors, integration of gyroscope measurement noise brings seriously poor performance. Figure 6.16 frankly shows the importance of sensor fusion for the accurate estimate of attitude.

The results of the simulations for the master estimator are provided in the following figures. The master-slave Kalman filter has a high accuracy in Euler angles estimation (see Figure 6.17). The roll and pitch attitude errors are approximately 0.04 degree. The yaw error is relatively bigger as 0.06 degree, which is a results of the inaccuracy in measuring the magnetic vector.

TABLE 6.4: RMS and maximum values of Euler angles estimation errors.

Euler Angle	**RMS Error** $[deg]$	**Maximum Error** $[deg]$
Roll - ϕ	0.0462	0.817
Pitch - θ	0.0498	0.955
Yaw - ψ	0.0643	1.106

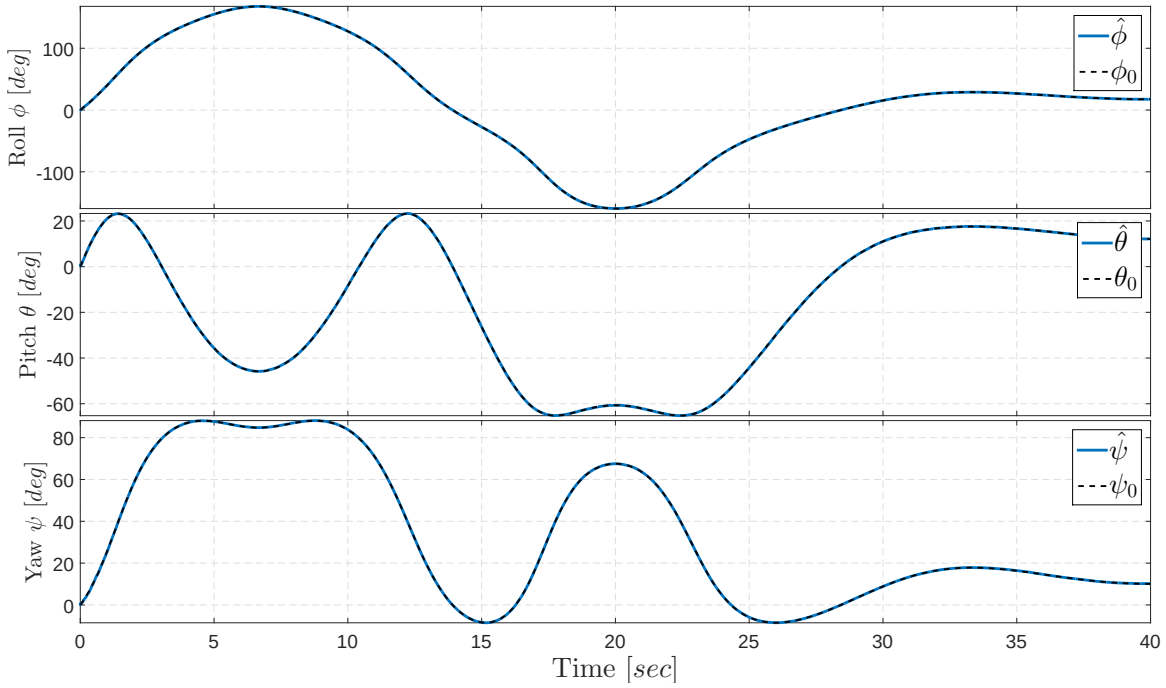

FIGURE 6.17: Comparison of estimated and true Euler angles.

It is important to estimate gyroscope biases where they have a key role to compensate slave estimator's measurement and provide unbiased smoothed body rates (see Figure 6.13). The implemented filter was successful enough to estimated constant bias of gyroscope measurements. In Figure 6.18, the gyroscope biases are presented and in Table **??** the rms and maximum values of the bias estimation error in the time interval of 10-15 seconds are provided.

TABLE 6.5: RMS and maximum values of gyroscope bias estimation errors between 10-15 sec.

Bias	**RMS Error** $[deg/s]$	**Maximum Error** $[deg/s]$
b_{g_x}	0.0396	0.0814
b_{g_y}	0.0354	0.0736
b_{g_z}	0.0391	0.0982

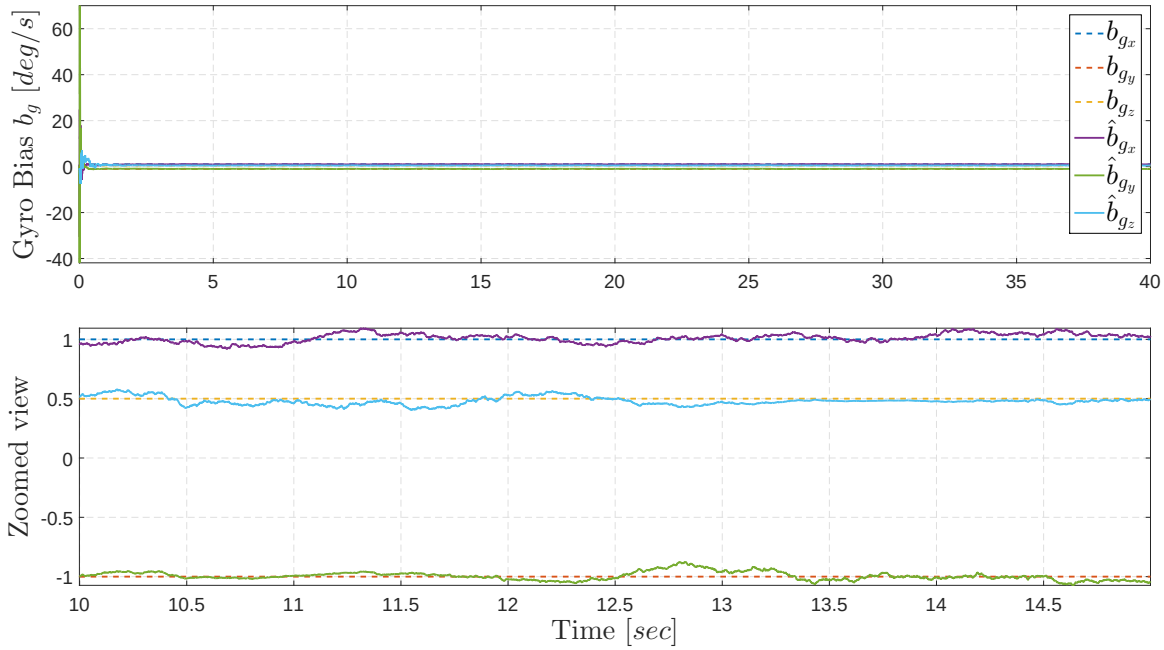

FIGURE 6.18: Estimated gyroscope biases and zoomed views.

The performance of the proposed sensor fusion algorithm largely depends on how good can it estimate Euler velocity and acceleration in the inertial frame. Since those estimations will be used as feedback for stabilization control, they have to be not only consistent but also smooth enough.

Estimated Euler rates are presented in Figure 6.19 and in Table **??** the rms and maximum values of the Euler rate estimation error are provided.

TABLE 6.6: RMS and maximum values of Euler angular rates estimation errors between 1-40 sec.

Euler Rate	**RMS Error** $[deg/s]$	**Maximum Error** $[deg/s]$
Roll Rate - $\dot{\phi}$	0.389	3.572
Pitch Rate - $\dot{\theta}$	0.414	3.198
Yaw Rate - $\dot{\psi}$	0.453	3.814

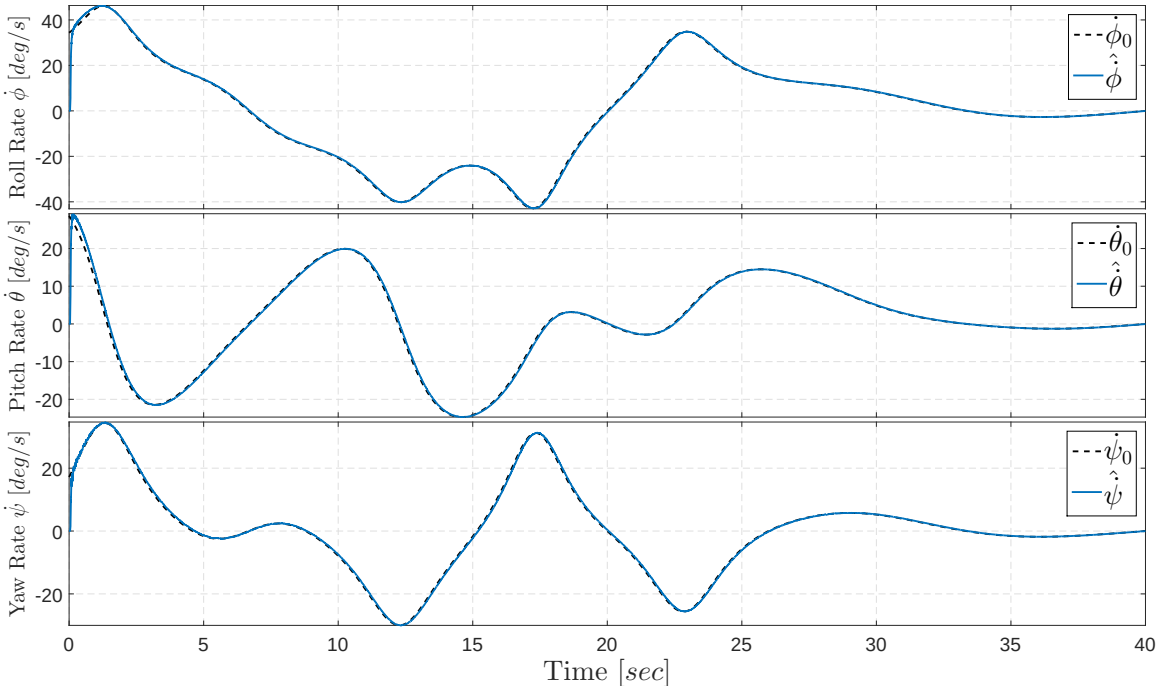

FIGURE 6.19: Comparison of estimated and true Euler rates.

Lastly, as depicted in Figure 6.20 master-slave Kalman filter estimates Euler accelerations successfully. Table **??** shows the rms and maximum values of the Euler acceleration estimation error.

TABLE 6.7: RMS and maximum values of Euler angular acceleration estimation errors between 1-40 sec.

Euler Acceleration	**RMS Error** $[deg/s^2]$	**Maximum Error** $[deg/s^2]$
Roll Acceleration - $\ddot{\phi}$	0.463	4.092
Pitch Acceleration - $\ddot{\theta}$	0.381	2.84
Yaw Acceleration - $\ddot{\psi}$	0.474	4.775

As it can be seen from Figure 6.19 and Figure 6.20, the accuracy of the developed method is sufficient enough to use its outputs in control loops.

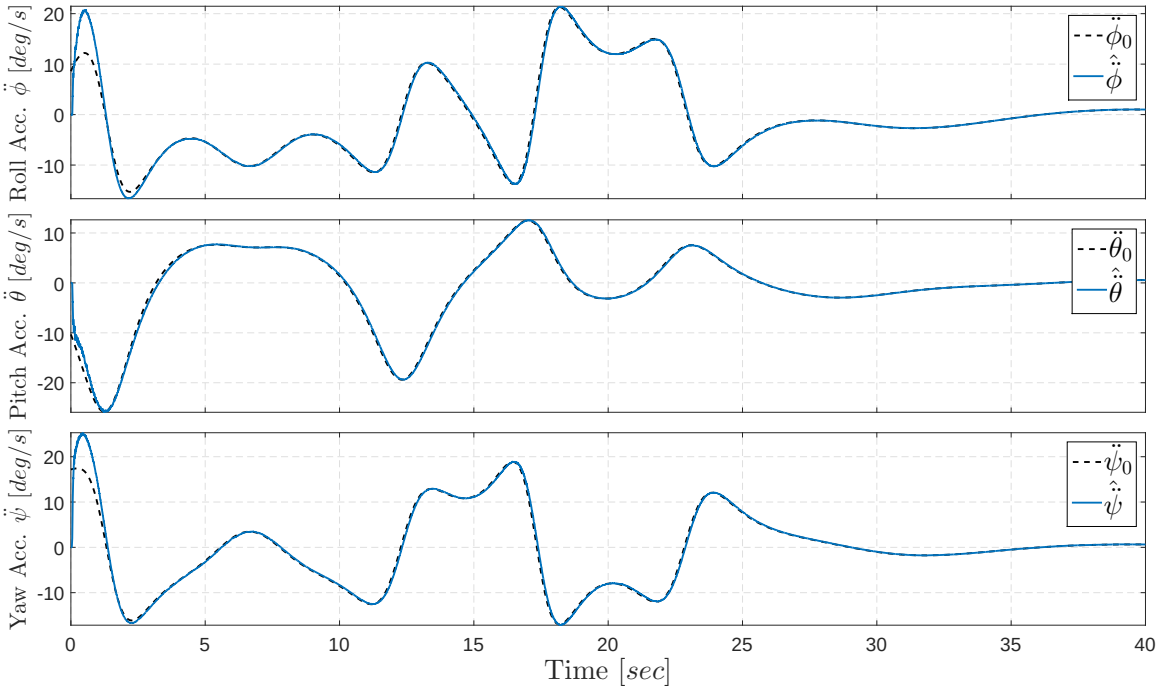

FIGURE 6.20: Comparison of estimated and true Euler accelerations.

6.1.4 Simulation: Control Results

Simulation results for two different scenarios are presented in this section where the proposed balancing and tracking controls are incorporated. In the first one, self-balancing performance is investigated, while in the second simulation a horizontal cartesian trajectory is tracked by the ballbot. All simulations are implemented in Simulink. Furthermore, external disturbances are generated to illustrate realistic effects of environmental factors in both scenarios.

6.1.4.1 Self-Balancing Results

In the first simulation, the body is initially tilted from its vertical position as setting roll angle as $\phi_b = -20\ deg$. During the balancing, the ballbot is exposed to sudden shocks on the body.

Disturbance acting on the ballbot is modeled as one point force like human touch. In order to illustrate this case, disturbance torques are modeled as short duration pulses with a magnitude of $\pm 20\ Nm$ that are applied on the system between 5 and 10 seconds as shown in Figure 6.21.

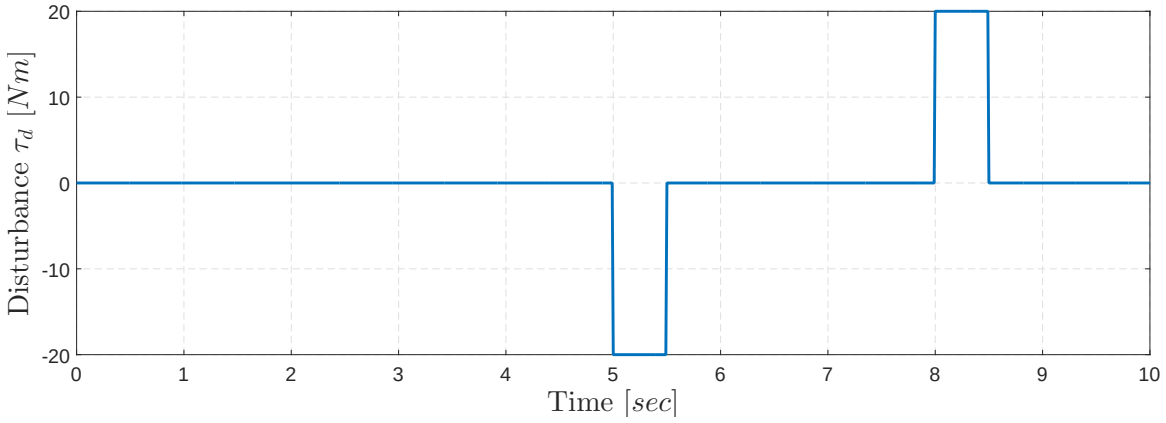

FIGURE 6.21: External disturbances applied on the body through roll axis.

Balancing performances of the both cascaded AFB control and PD control are shown in Figures 6.22 and 6.23.

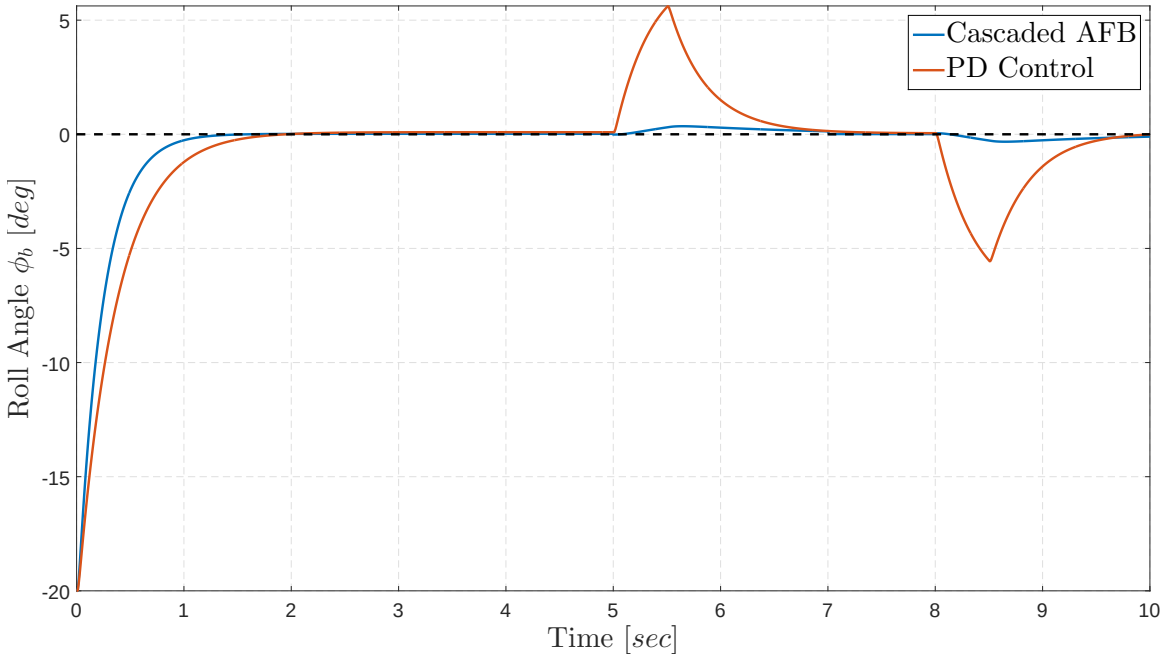

FIGURE 6.22: Simulation results of the self-balancing: Body roll angle ϕ_b.

TABLE 6.8: Rise time and maximum deviation of body roll angle for both controllers.

Controller	**Rise time** $[sec]$	**Maximum Deviation** $[deg]$
Cascaded AFB	0.678	0.35
PD	1.03	5.6

Body roll angle converged to zero from initially tilted position with fast transient response for both controller but PD controller resulted was a bit slower. While the rise time for cascaded AFB controller to 95% with respect to a given reference value is 0.678 second, the rise time of PD controller is 1.03 second. When the acceleration feedback is not used, body orientation can not be preserved due to sudden shocks between $5 - 10$ sec. The plots show that with AFB controller the ballbot has successfully stayed upright when subjected to the push during balancing. Body roll angle remained around zero with an maximum error 0.35 *deg*. However, PD controller could not reject disturbances and cause the ballbot to incline through roll and pitch axes, then it takes around 2 seconds to settle back to upright position. The disturbance resulted in a maximum deviation from upright of approximately 5.6 *deg* for roll and 0.45 *deg* for pitch angles.

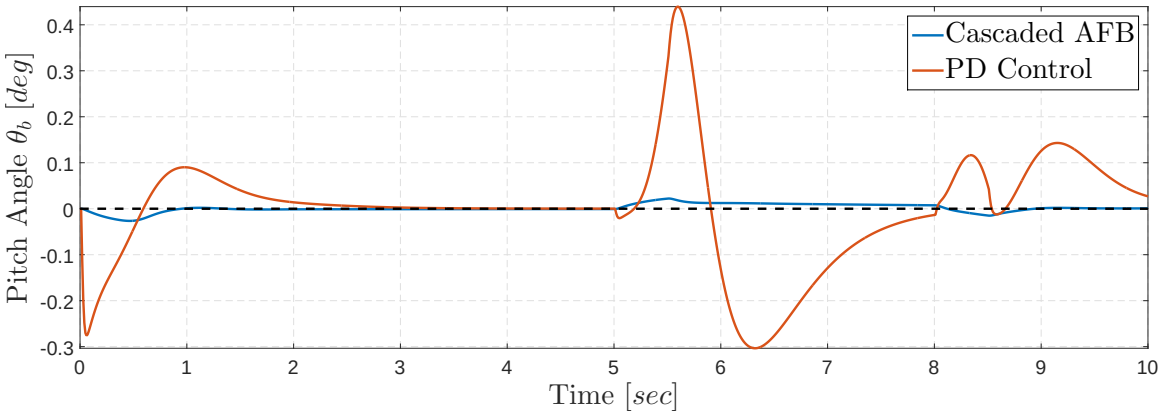

FIGURE 6.23: Simulation results of the self-balancing: Body pitch angle θ_b.

6.1.4.2 Trajectory Tracking Results

In the second simulation, the ballbot tracked a time-varying trajectory starting from the origin of the inertial frame. During trajectory tracking, the ballbot has to perform self-balancing. In order to prevent rapid acceleration that makes the body to attempt aggressively maneuvers toward the reference, desired trajectories are generated by using virtual time. In doing so, velocity and acceleration references can be slowed down and smoothed trajectories are generated with zero initial velocities and accelerations, where the maximum velocity is 0.1 m/s and maximum acceleration is 0.014 m/s^2. With a well designed reference trajectory, the ballbot can accomplish a given task more stable. In this scenario, initial conditions for all states are selected as zero. During trajectory tracking, the ballbot is exposed to random generated small external disturbances to show effectiveness of the proposed AFB controller.

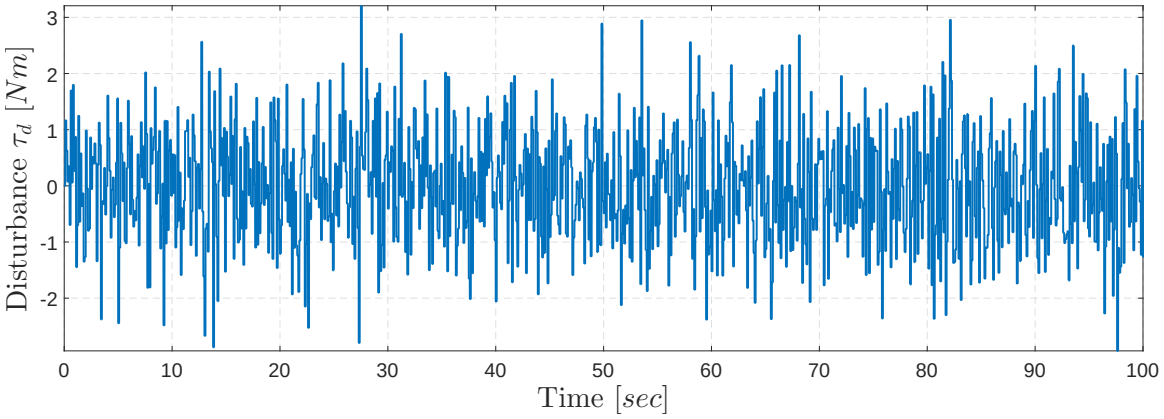

FIGURE 6.24: External disturbances applied on the body through roll axis.

Tracking response on XY plane is presented in the following figure. The ballbot tracked the reference trajectory in a much smoother way when the cascaded AFB is used with the cascaded position controller. In other words, random external disturbances that are given in Figure 6.24 are rejected. In contrast, PD controller is failed to follow reference properly and resulted in a drift error almost up to 18 cm.

Rms of the drift error is found as 5 cm when PD control is used, for cascaded AFB control it decreased about 80% to 1 cm.

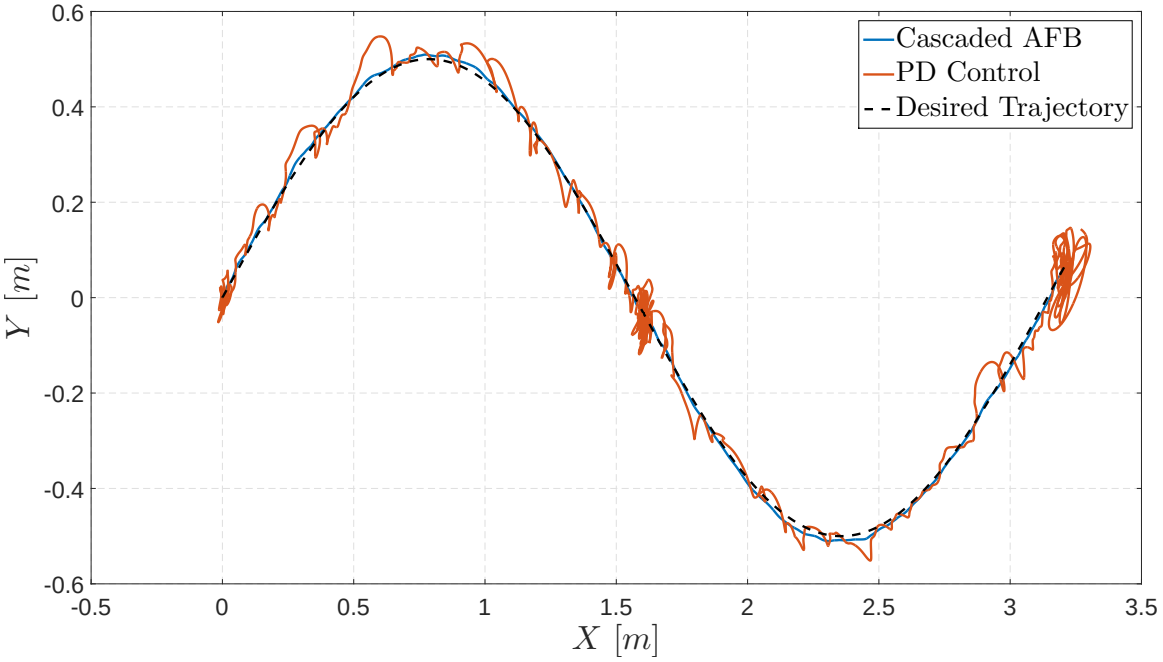

FIGURE 6.25: Simulation results of the tracking: 2D position.

TABLE 6.9: RMS and maximum values of positional drift errors for both controllers.

Controller	**RMS Error** $[m]$	**Maximum Drift** $[m]$
Cascaded AFB	0.01	0.02
PD	0.05	0.18

6.2 Experimental Results

Experimental results have been considered as being the most important part of this thesis. Therefore obtaining convincing results to be consistent with simulation results is the most important outcome of this work. In order to investigate the derived Kalman Filter equations and control algorithms on the designated real-time software, they were coded in C language in Visual Studio environment.

6.2.1 Experimental: Sensor Fusion Results

As stated earlier, the MEMS inertial measurement unit being used in this study is the IMU Brick 2.0 from TinkerForge Inc. Master-Slave Kalman filter is implemented for 20 seconds duration by moving the IMU randomly. To check the performance the proposed sensor fusion method, Euler angles provided directly by IMU's embedded system are also presented.

First of all, let us examine the inertial sensor readings are provided by IMU as follow:

Gyroscope measurements are obtained as follows:

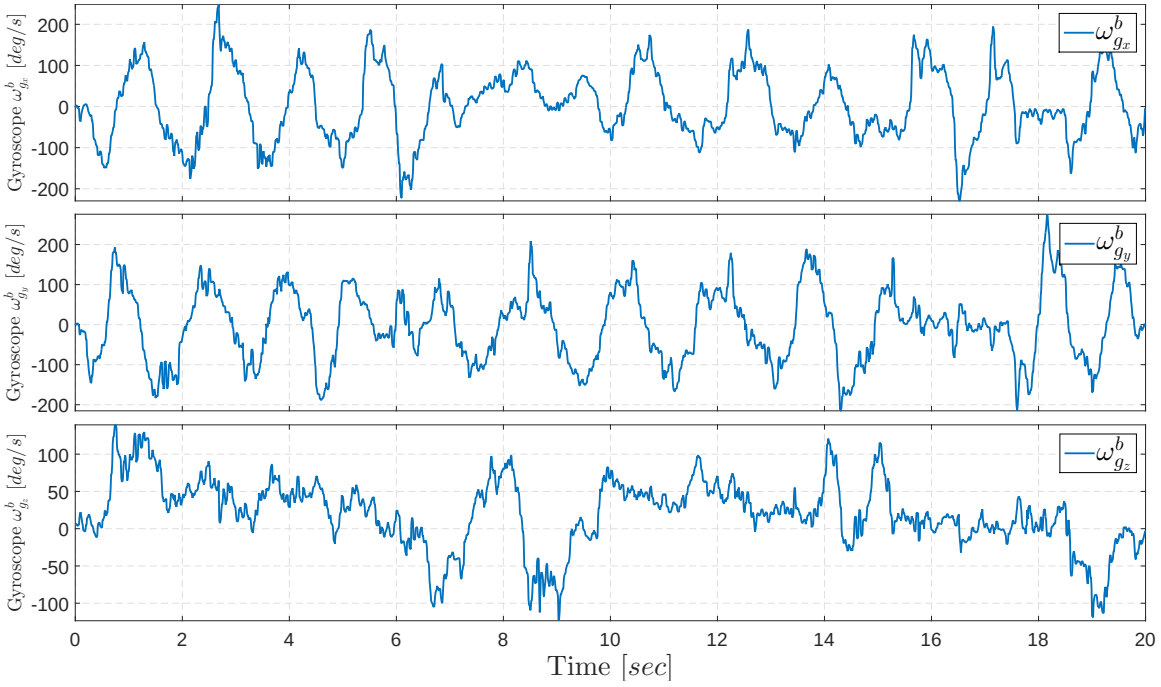

FIGURE 6.26: Gyroscope measurement by IMU.

Accelerometer measurements are obtained as shown in Figure 6.27.

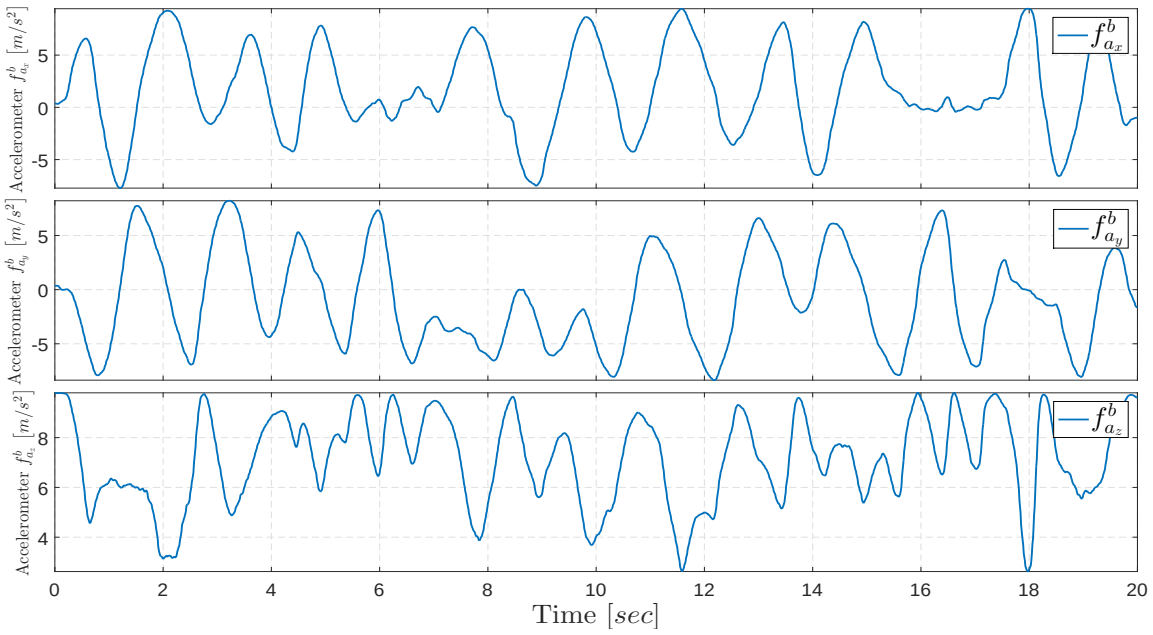

FIGURE 6.27: Accelerometer measurement by IMU.

And, magnetometer measurements are obtained as

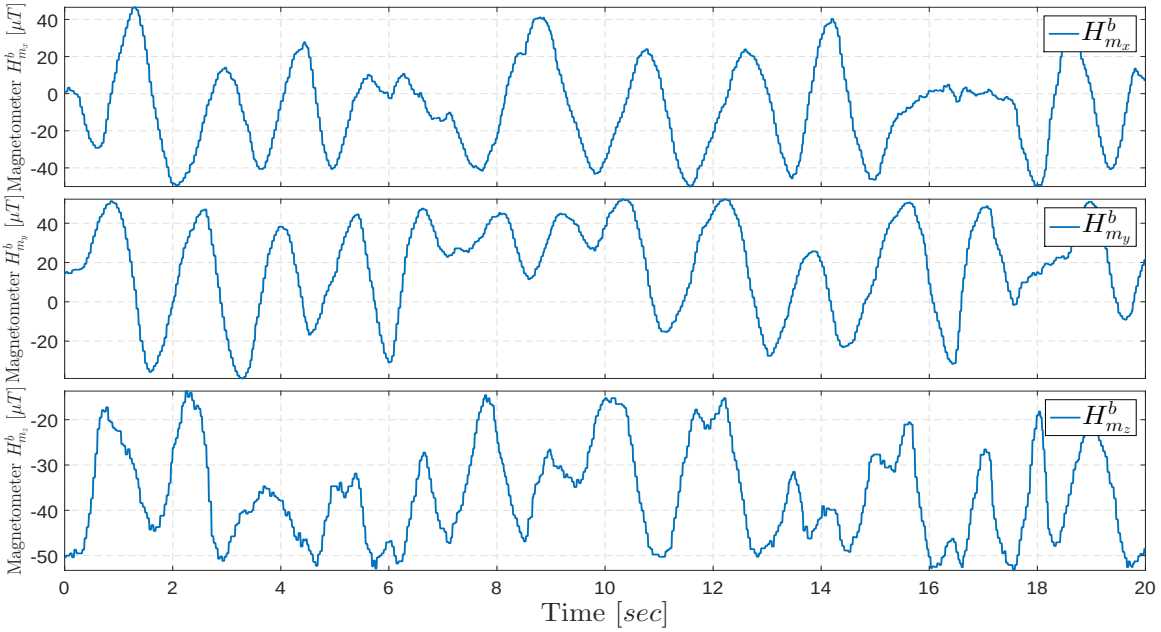

FIGURE 6.28: Magnetometer measurement by IMU.

Estimated body rates by slave Kalman filter are obtained as smoothed version of gyro readings as shown in the following figure for x, y, and z axes respectively.

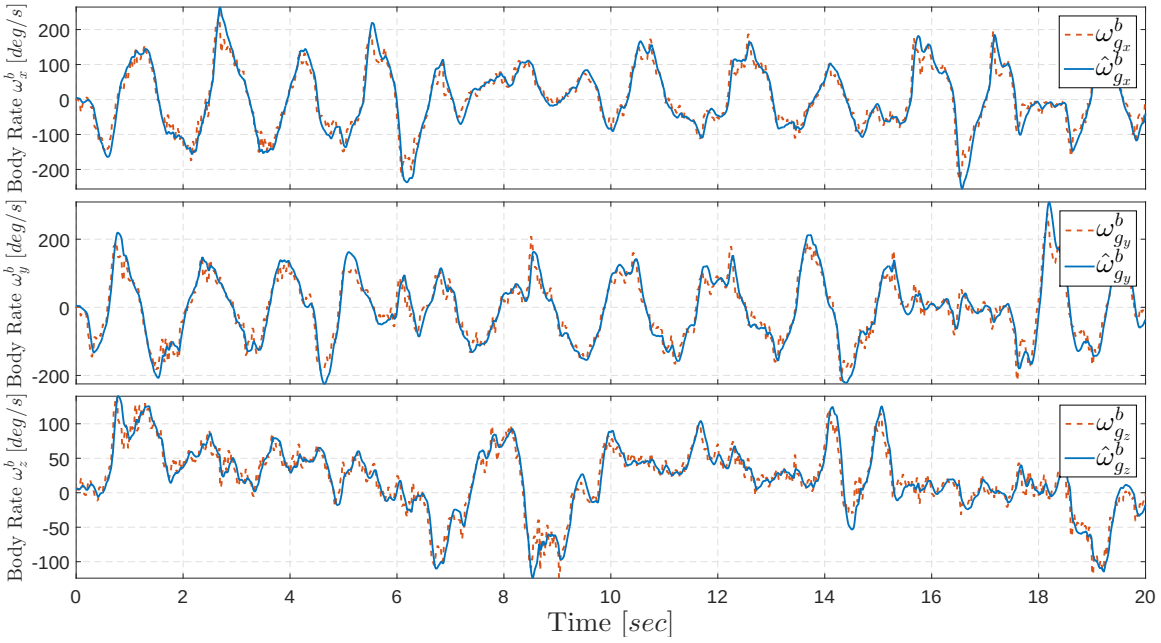

FIGURE 6.29: Estimated and measured angular velocity in the body coordinate system.

Secondly, body accelerations are obtained as follows:

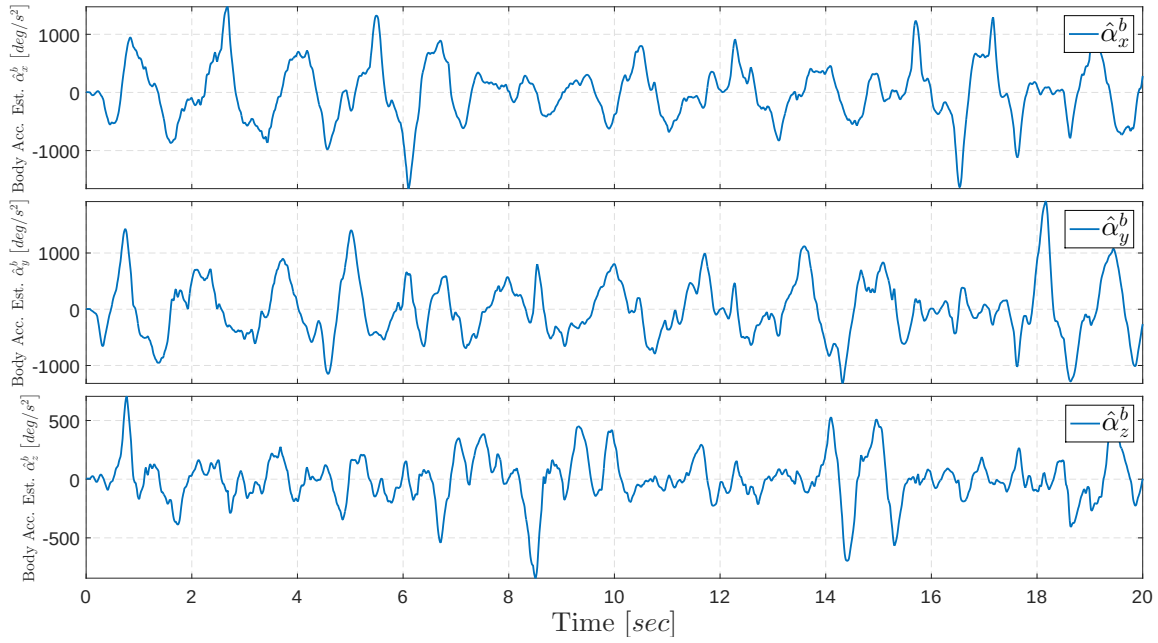

FIGURE 6.30: Estimated angular acceleration in the body coordinate system.

And, body angular jerks are obtained as follows.

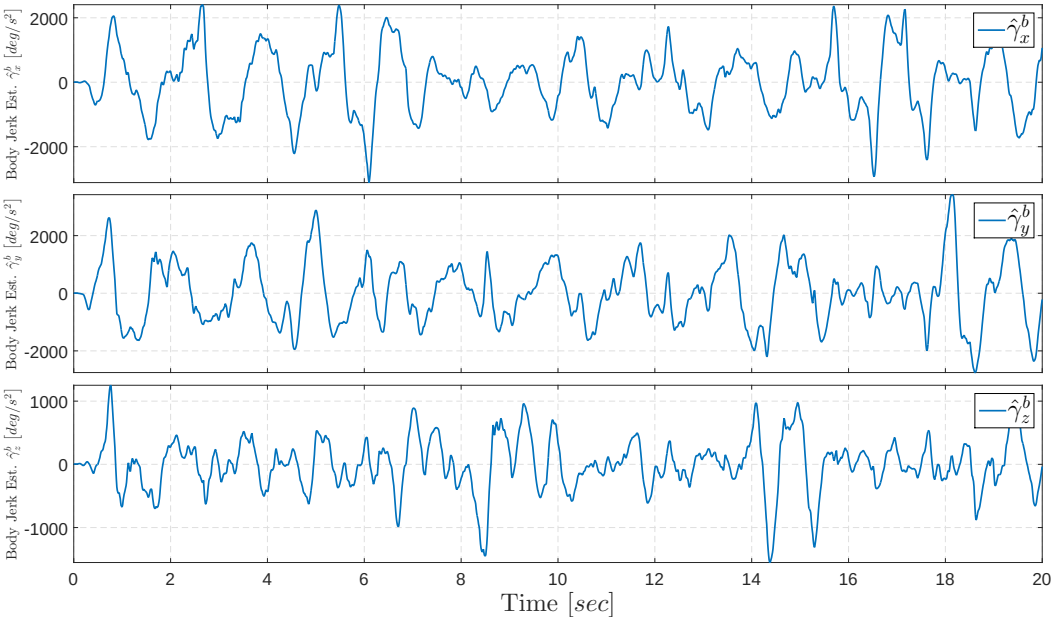

FIGURE 6.31: Estimated angular jerk in the body coordinate system.

In order to observe the drift problem, attitude results obtained by integration of the real gyroscope measurements are shown in Figure 6.32. Even for this short duration, integration of sensor errors cause a noticeable drift in Euler angles.

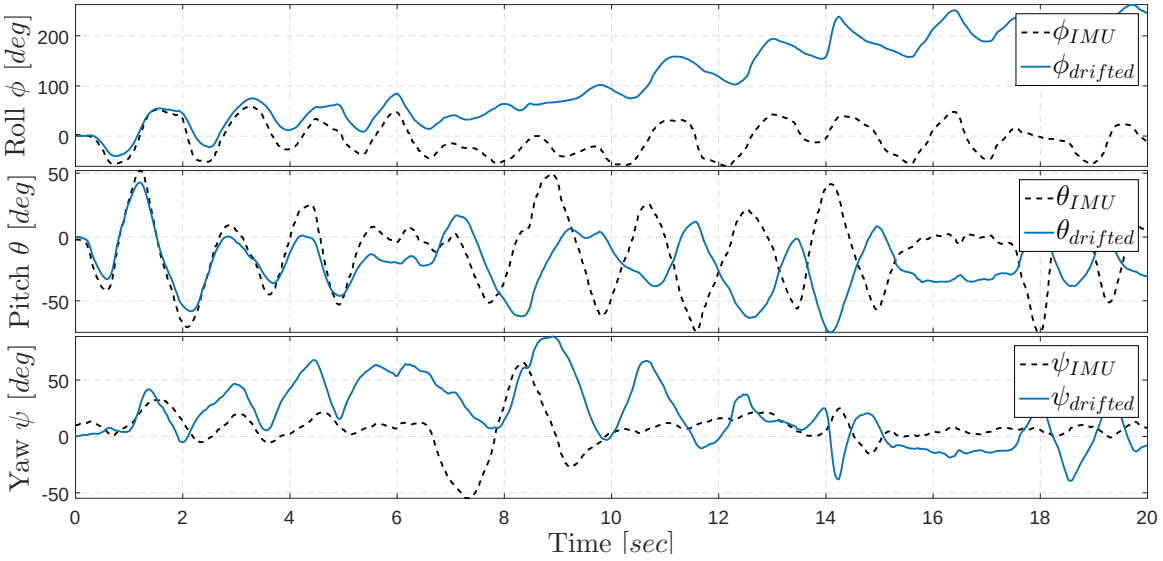

FIGURE 6.32: Comparison of drifted and measured Euler angles.

Experimental results for the master estimator are provided in the following figures. The master Kalman filter has a high accuracy in Euler angles estimation, where the estimated attitude angles are perfectly same as measured ones (see Figure 6.33). If we assume the orientation angles that directly measured from IMU as reference

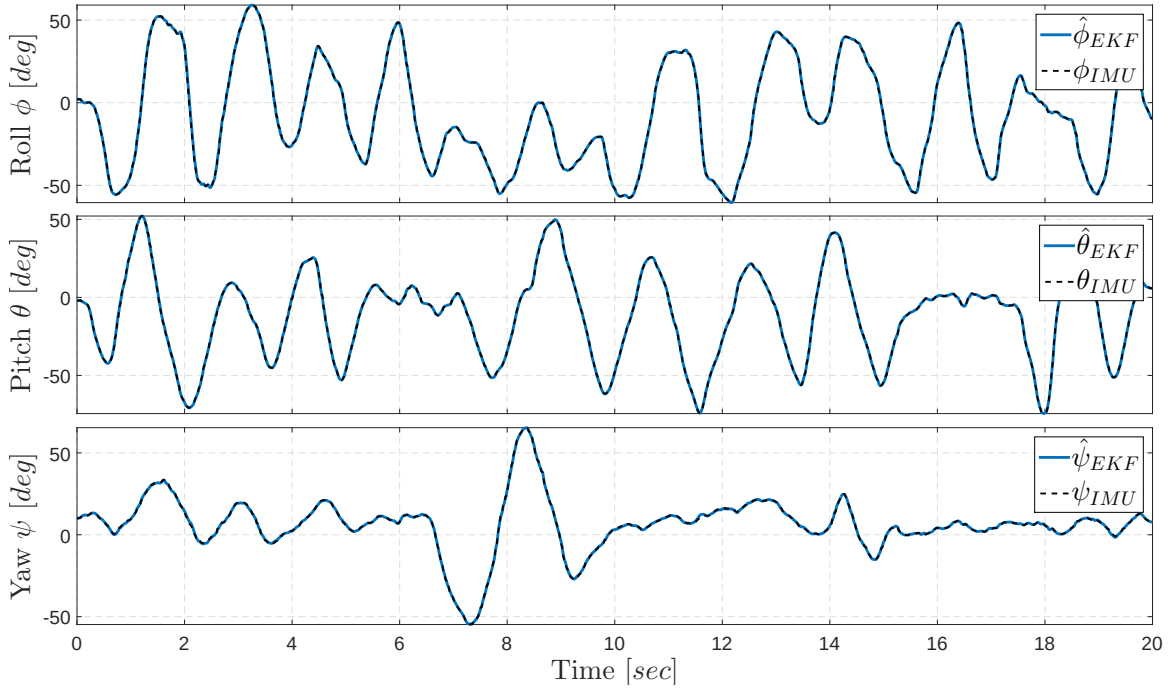

FIGURE 6.33: Comparison of estimated and measured Euler angles.

angles, estimation errors were obtained as follows: The roll error is approximately 0.585 degree, the pitch error is 0.478 degree and for the yaw angle error is around 0.309, as detailed in Table **??**.

TABLE 6.10: RMS and maximum values of Euler angles estimation errors.

Euler Angle	**RMS Error** $[deg]$	**Maximum Error** $[deg]$
Roll - ϕ	0.585	4.54
Pitch - θ	0.478	2.305
Yaw - ψ	0.309	2.678

In simulation data set, gyroscope measurement were created with constant bias assumption (see Table 6.1), where the estimated biases in simulation results were inherently almost constant as depicted in Figure 6.18. In contradistinction to simulation results, the estimated gyroscope biases were considerably varying, in spite of the fact that they were assumed to be constant. The implemented filter was successful enough to estimated those biases of gyroscope measurements. In Figure 6.34, the gyroscope biases are presented and in Table **??** the maximum, mean and variation of estimated bias values are provided.

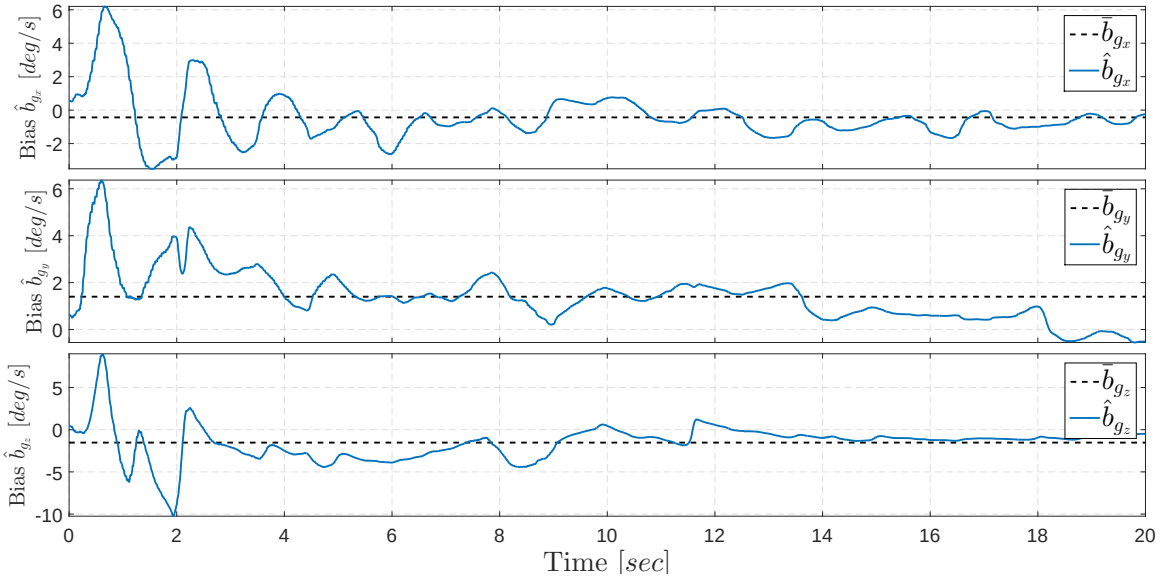

FIGURE 6.34: Estimated gyroscope biases.

TABLE 6.11: Maximum, mean and variance of values of the estimated gyroscope biases.

Bias	**Maximum Bias** $[deg/s]$	**Mean** $[deg/s]$	**Variance** $[deg/s]$
b_{g_x}	6.20	-0.434	2.0
b_{g_y}	6.37	1.398	1.22
b_{g_z}	8.99	-1.531	4.17

Estimated Euler rates and accelerations are presented in Figure 6.35 and in Figure 6.36 respectively.

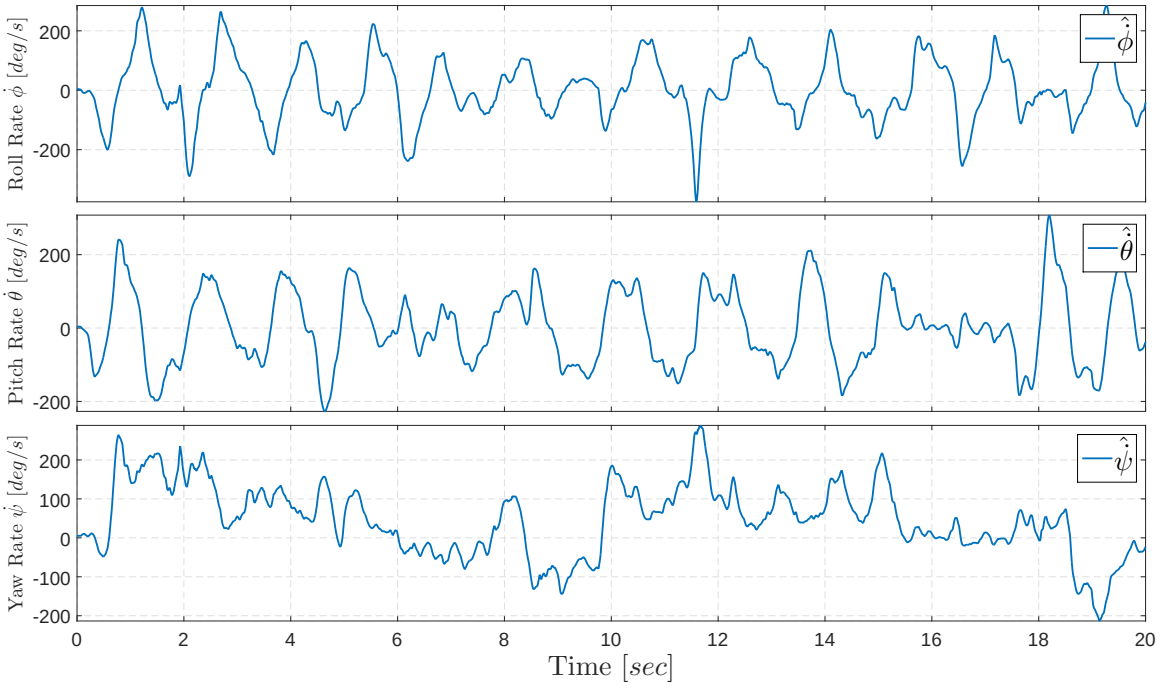

FIGURE 6.35: Estimated Euler rates from IMU measurements.

These results seem consistent because in this case it is not possible to compare the estimated Euler rate and acceleration values with true values, since these values are not known.

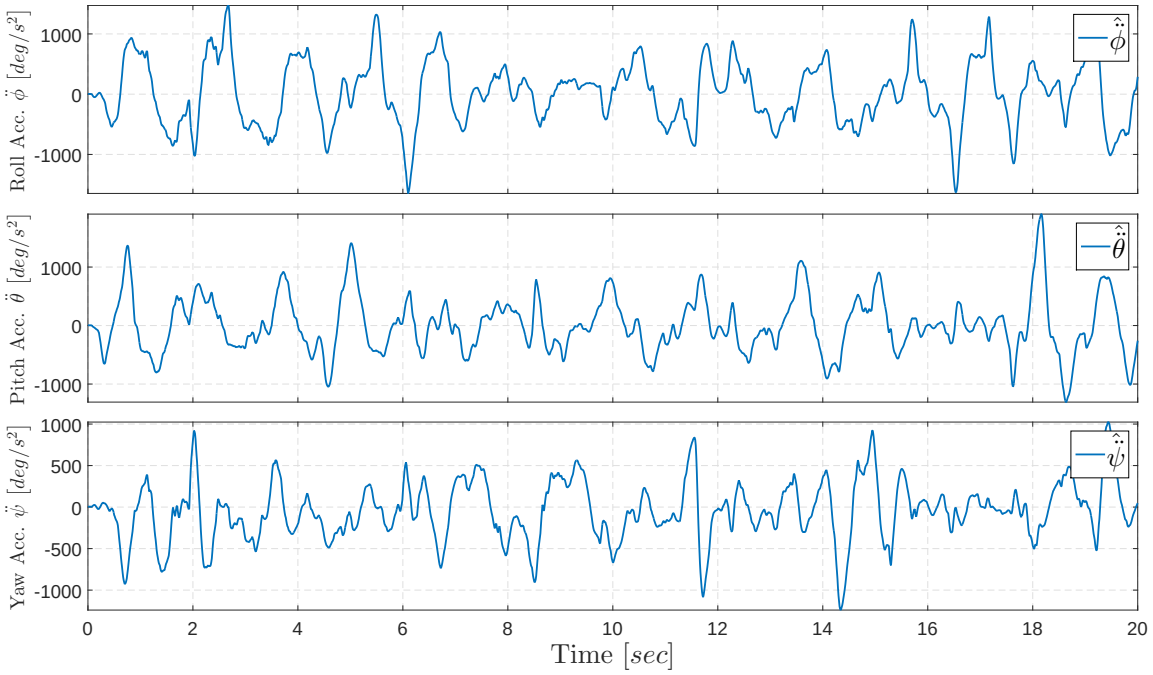

FIGURE 6.36: Estimated Euler accelerations from IMU measurements.

6.2.2 Experimental: Control Results

In this section, we evaluate the performance of the presented sensor fusion and control methods on real ballbot system, in particular of balancing performance. Ballbot experiments in this thesis only focus on investigating the balancing performances of PD and cascaded AFB controllers. In the performed experiments, the body is initially tilted from its vertical position. After the settling down, the ballbot is exposed to sudden shocks on the body as one point force human touch as short duration pulses.

In order to compare the cascaded AFB control and PD control, same procedures were followed from initial angle setting to applied disturbances. The applied distorting effects are maintained at the same magnitude and duration as far as possible. For the experiments, initial attitude is set as tilted towards roll angles approximately as $\phi_b = -20\ deg$. This initial arrangement is done by real-time monitoring of the sensor fusion outputs. After that, selected controller is activated manually by keyboard

input. Each experiment took 40 seconds and disturbance is exerted between 5-35 seconds.

Balancing performances of the both cascaded AFB control and PD control on real ballbot system are shown in Figures 6.37 and 6.38.

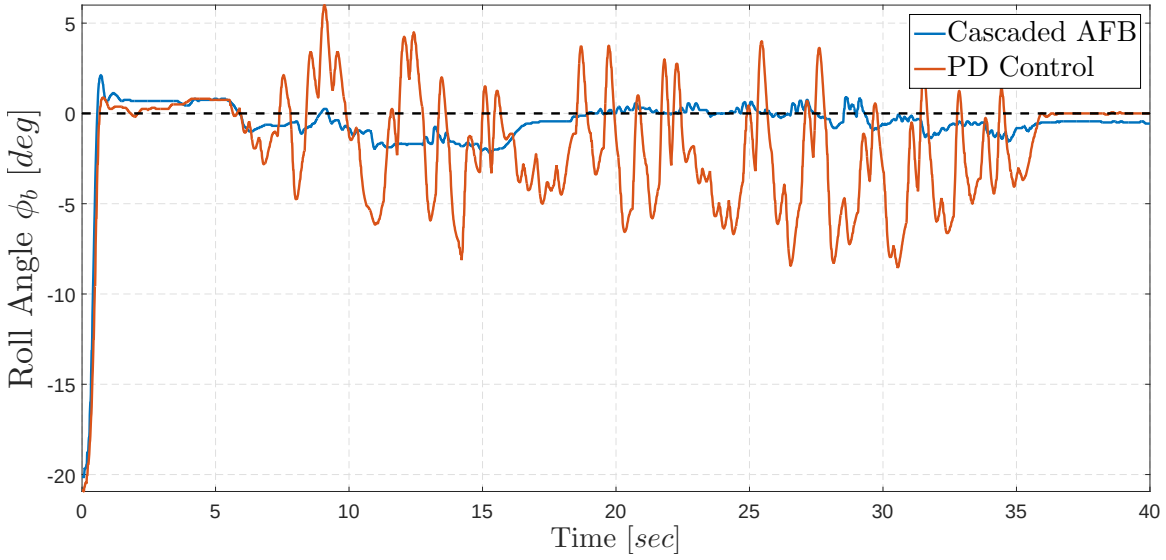

FIGURE 6.37: Experimental results of the self-balancing: Body roll angle ϕ_b.

Body roll angle converged to zero from initially tilted position with fast transient response for both controller but PD controller resulted was a bit slower. While the rise time (for 95% with respect to a given reference value) of cascaded AFB controller is 0.36 second, the rise time of PD controller is 0.411 second. When the acceleration feedback is not used, body orientation can not be preserved due to sudden shocks between $5 - 35$ seconds. The Figures 6.37 and 6.38 show that with AFB controller the ballbot behaved more robust when subjected to the push during balancing. Body roll angle remained around zero with an maximum error 2.188 *deg*. However, PD controller failed to reject disturbances and cause the ballbot to incline through roll and pitch axes. The disturbance resulted in a maximum deviation from upright of approximately 8.563 *deg* for roll angle.

TABLE 6.12: Rise time and maximum deviation of the body roll angle ϕ_b for both controllers.

Controller	**Rise time** $[sec]$	**Rms** (5-35 sec) $[deg]$	**Max. Deviation** $[deg]$
Cascaded AFB	0.360	0.855	2.188
PD	0.411	3.487	8.563

It can also be observed that the body pitch angle also preserved much more stable when the AFB is introduced as shown in Figure 6.38. RMS error in the θ_b is significantly smaller when AFB is used. While the rms of the pitch angle is 0.586 deg for PD control, the AFB contributed to decrease it to 0.248 deg. Consequentially, the maximum deviation is reduced from 1.625 deg to 0.487 deg.

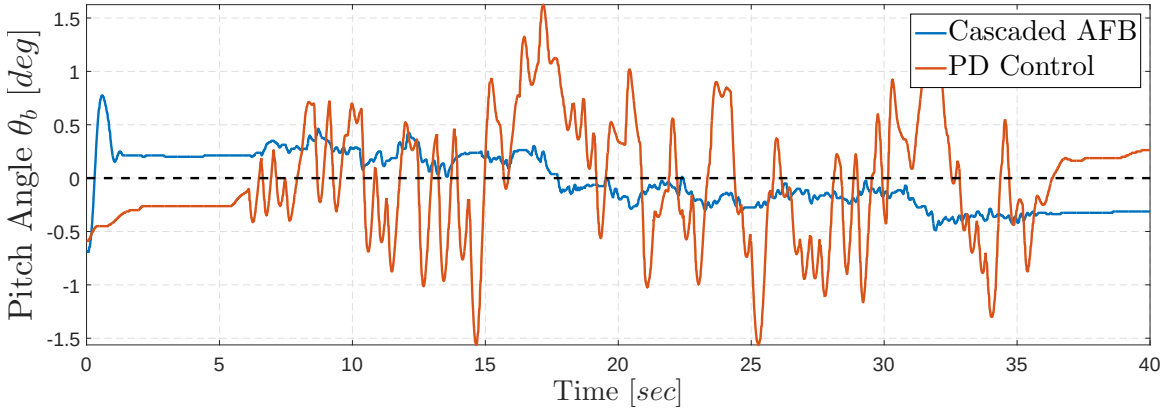

FIGURE 6.38: Experimental results of the self-balancing: Body pitch angle θ_b.

TABLE 6.13: Rms and maximum of the body pitch angle between 5-35 seconds for both controllers.

Controller	**Rms** (5-35 sec) $[deg]$	**Maximum Deviation** $[deg]$
Cascaded AFB	0.248	0.487
PD	0.586	1.625

Body Euler velocities are given in Figure 6.39. For the PD control case, rms values of estimated roll and pitch velocities are 17.01 deg/s and 3.105 deg/s. When the AFB is used, rms values of estimated roll and pitch velocities become 1.949 deg/s and 0.396 deg/s as detailed in Table **??**.

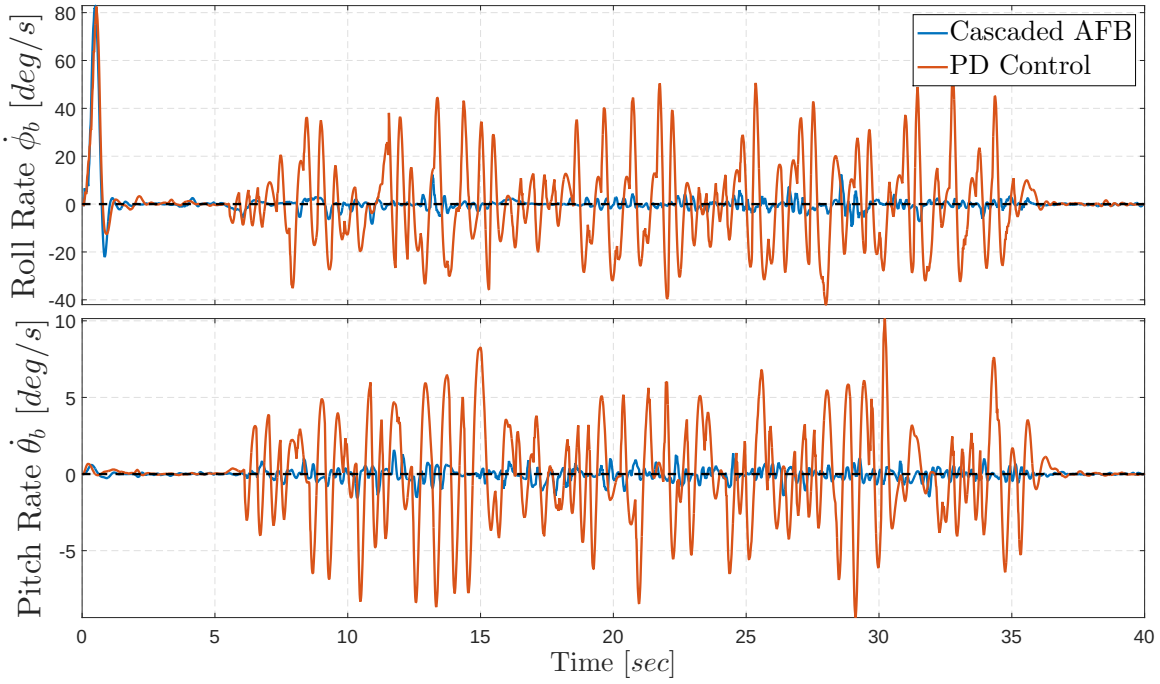

FIGURE 6.39: Experimental results of the self-balancing: Euler rates, $\dot{\phi}_b$, $\dot{\theta}_b$.

TABLE 6.14: Rms and maximum values of the body Euler rates between 5-35 seconds for both controllers.

Controller	**Euler Rate** $[deg/s]$	**Rms** $[deg/s]$	**Maximum Rate** $[deg/s]$
Cascaded AFB	Roll rate - $\dot{\phi}_b$	1.949	10.147
	Pitch rate - $\dot{\theta}_b$	0.396	1.568
PD	Roll rate - $\dot{\phi}_b$	17.01	53.479
	Pitch rate - $\dot{\theta}_b$	3.105	10.1472

Additionally, Figure 6.40 shows estimated Euler accelerations. When the AFB is utilized in the cascaded control loop, rms values of the estimated acceleration signals are 11.89 deg/s^2 and 2.50 deg/s^2 respectively for roll and pitch axes. As the acceleration feedback is not used, RMS values of the Euler accelerations increase to 165.64 deg/s^2 and 27.44 deg/s^2.

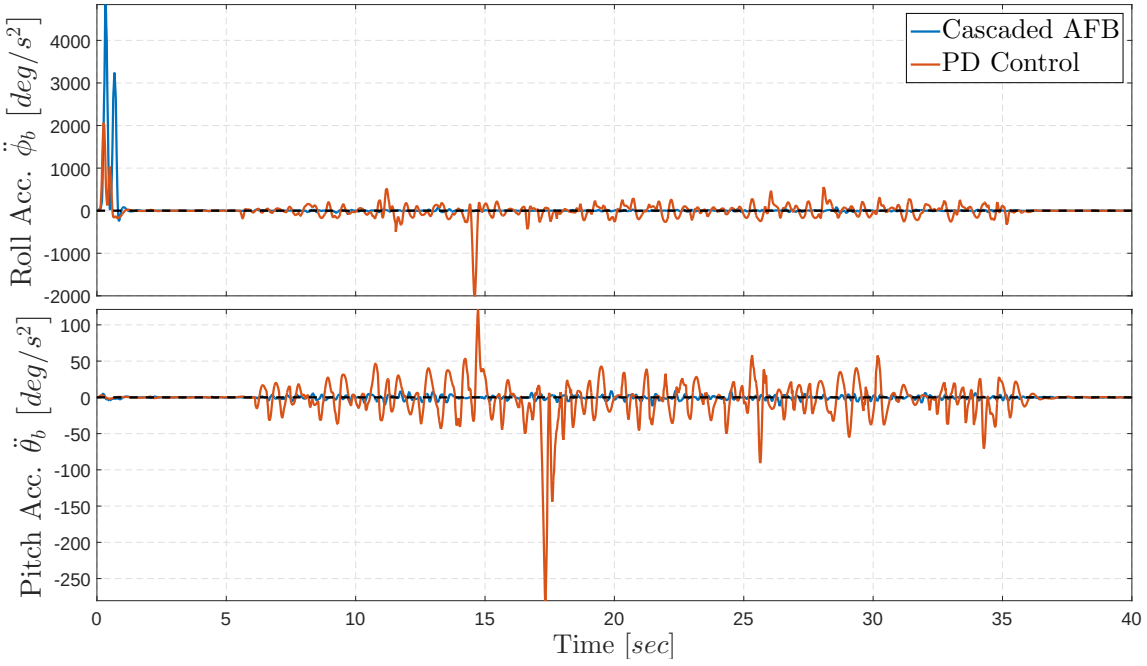

FIGURE 6.40: Experimental results of the self-balancing: Euler accelerations, $\ddot{\phi}_b$, $\ddot{\theta}_b$.

TABLE 6.15: Rms and maximum values of the body Euler accelerations between 5-35 seconds for both controllers.

Controller	**Euler Acc.** $[deg/s^2]$	**Rms** $[deg/s^2]$	**Maximum Acc.** $[deg/s^2]$
Cascaded AFB	Roll Acc. - $\ddot{\phi}_b$	11.89	81.29
	Pitch Acc. - $\ddot{\theta}_b$	2.50	11.77
PD	Roll Acc. - $\ddot{\phi}_b$	165.64	2000
	Pitch Acc. - $\ddot{\theta}_b$	27.44	280

Chapter 7

Conclusion and Future Works

In this thesis, a novel master-slave Kalman filter was proposed where an extended Kalman filter (EKF) and a classical Kalman filter (KF) were integrated in a master-slave configuration to estimate reliable angular motion signals by fusing measurements of an inertial measurement unit (IMU). The proposed fusion method reliably estimated not only Euler angles but also rates and accelerations in both simulations and experimental ballbot system. The performance of proposed fusion algorithm was evaluated by using the estimated angular motion signals as feedback in the control of a self-balancing and position of a single spherical wheeled mobile platform driven by three omniwheels, i.e. ballbot. For this purpose, the design and construction of a ballbot platform with MEMS sensors and real-time control hardware were conducted, followed by the analytical derivation of the ballbot dynamics by means of Euler-Lagrange formulation. Balancing and position controllers that utilize acceleration feedback (AFB) were then designed to achieve stabilization of the ballbot and tracking of a desired trajectory.

Performance of the proposed sensor fusion method and the developed controllers were assessed by using a high fidelity simulation platform. First, inertial sensor measurements were generated and sensor fusion simulations were performed, then

performance of the proposed fusion method was examined by comparing estimated values with true values. The master-slave Kalman filter accurately estimated angular motion values with rms error for Euler angles around 0.05 deg, for Euler rates 0.4 deg/s, and for Euler accelerations as 0.5 deg/s^2. Secondly, ballbot simulator was used to compare cascaded AFB and PD controllers in two scenarios subject to different external disturbances. Much better stabilization performance was achieved in the both scenarios with acceleration feedback (AFB) controller. In the first one, self-balancing results were quite satisfying where the maximum deviation of the body orientation angles decreased by 95% in the presence of AFB. In the second scenario, the desired trajectory was tracked in a much smoother way when the cascaded AFB was used with the cascaded position controller where the rms of the drift error was decreased about 80% to 1 cm.

Additionally, several experiments were conducted on the prototype ballbot system. Initial experiments focused on the implementation of master-slave Kalman filter with real sensor data and reasonable results were obtained. Developed fusion algorithm was then incorporated into the implementation of the ballbot self-balancing control under external disturbances where the ballbot exposed to forces exerted on top of the upper plate. With the PD controller, the ballbot failed to preserve its equilibrium and deviated up to 8.5 deg. The same scenario was repeated with the developed controller that utilizes estimated acceleration feedback, and the results showed better performances with the decrease of 75% of maximum deviation for roll angle and 70% for pitch angle. Success of the cascaded AFB controller has also been observed by Euler rate and acceleration responses against disturbances. Cascaded AFB successfully accomplished disturbance rejection to prevent rapid acceleration where the rms values of the estimated Euler rates and accelerations were reduced by 90%. However, in experiments, only balancing performance of the proposed control method was investigated due to disruptions and deficiencies in the structure such as excessive friction and neglected slippage.

Possible future works may include the experiments of robust position control of the ballbot enhanced with AFB which can be done by utilizing additional external feedback, e.g. vision or laser ranger measurements, and overcoming friction and slippage problems. Moreover, the controllers and drivers can be mounted to the main body to make the system fully mobile.

Appendix A

Jacobian Matrices for Master-Slave Kalman Filter

This appendix includes the state transition and measurement matrices F and H of the master estimator detailed in Section 3.3.1. Additionally, the state derivative and measurement vectors in terms of the states is also provided.

```
%% In below expressions:
% phi: roll angles
% theta: pitch angle
% psi: yaw angle
% phid: roll rate
% thetad: pitch rate
% psid: yaw rate
% phidd: roll acceleration
% thetadd: pitch acceleration
% psidd: yaw acceleration
% gx: gyro rate around x axis
% gy: gyro rate around y axis
% gz: gyro rate around z axis
% wx: body rate slave estimator x axis
% wy: body rate slave estimator y axis
% wz: body rate slave estimator z axis
% wxd: body acceleration slave estimator x axis
% wyd: body acceleration slave estimator y axis
```

```
% wzd: body acceleration slave estimator z axis
% wxdd: body jerk slave estimator x axis
% wydd: body jerk slave estimator y axis
% wzdd: body jerk slave estimator z axis
% bx: bias x axis
% by: bias y axis
% bz: bias z axis

% 12x1 state vector
x(1) = phi;
x(2) = theta;
x(3) = psi;
x(4) = phid;
x(5) = thetad;
x(6) = psid;
x(7) = phidd;
x(8) = thetadd;
x(9) = psidd;
x(10) = bx;
x(11) = by;
x(12) = bz;

% State derivatives
xd(1) = gx - bx - cos(phi)*tan(theta)*(bz - gz) - sin(phi)*tan(theta)*(by - gy);
xd(2) = sin(phi)*(bz - gz) - cos(phi)*(by - gy);
xd(3) = - (sin(phi)*(by - gy))/cos(theta) - (cos(phi)*(bz - gz))/cos(theta);
xd(4) = wxd + wy*(thetad*sin(phi)*(tan(theta)^2 + 1) + phid*cos(phi)*tan(theta))
        + wz*(thetad*cos(phi)*(tan(theta)^2 + 1) - phid*sin(phi)*tan(theta))
        + wzd*cos(phi)*tan(theta) + wyd*sin(phi)*tan(theta);
xd(5) = wyd*cos(phi) - wzd*sin(phi) - phid*wz*cos(phi) - phid*wy*sin(phi);
xd(6) = wy*((phid*cos(phi))/cos(theta) + (thetad*sin(phi)*sin(theta))/cos(theta)^2)
        - wz*((phid*sin(phi))/cos(theta) - (thetad*cos(phi)*sin(theta))/cos(theta)^2)
        + (wzd*cos(phi))/cos(theta) + (wyd*sin(phi))/cos(theta);
xd(7) = wxdd + wy*(phid*(thetad*cos(phi)*(tan(theta)^2 + 1) - phid*sin(phi)*tan(theta))
        + thetad*(phid*cos(phi)*(tan(theta)^2 + 1) + 2*thetad*sin(phi)*tan(theta)*
        (tan(theta)^2 + 1)) + thetadd*sin(phi)*(tan(theta)^2 + 1) + phidd*cos(phi)*
        tan(theta)) - wz*(phid*(thetad*sin(phi)*(tan(theta)^2 + 1) + phid*cos(phi)*
        tan(theta)) + thetad*(phid*sin(phi)*(tan(theta)^2 + 1) - 2*thetad*cos(phi)*
        tan(theta)*(tan(theta)^2 + 1)) - thetadd*cos(phi)*(tan(theta)^2 + 1) + phidd*
        sin(phi)*tan(theta)) + wyd*(2*thetad*sin(phi)*(tan(theta)^2 + 1) + 2*phid*
        cos(phi)*tan(theta)) + wzd*(2*thetad*cos(phi)*(tan(theta)^2 + 1) - 2*phid*
        sin(phi)*tan(theta)) + wzdd*cos(phi)*tan(theta) + wydd*sin(phi)*tan(theta);
xd(8) = wydd*cos(phi) - wzdd*sin(phi) - wy*(cos(phi)*phid^2 + phidd*sin(phi))
```

```
        + wz*(sin(phi)*phid^2 - phidd*cos(phi)) - 2*phid*wzd*cos(phi)
        - 2*phid*wyd*sin(phi);
xd(9) = wz*(thetad*((thetad*cos(phi))/cos(theta) + (2*thetad*cos(phi)*sin(theta)^2)
        /cos(theta)^3 - (phid*sin(phi)*sin(theta))/cos(theta)^2) - phid*((phid*cos(phi))
        /cos(theta) + (thetad*sin(phi)*sin(theta))/cos(theta)^2) - (phidd*sin(phi))
        /cos(theta) + (thetadd*cos(phi)*sin(theta))/cos(theta)^2) + wy*(thetad*((thetad*
        sin(phi))/cos(theta) + (2*thetad*sin(phi)*sin(theta)^2)/cos(theta)^3 +
        (phid*cos(phi)*sin(theta))/cos(theta)^2) - phid*((phid*sin(phi))/cos(theta) -
        (thetad*cos(phi)*sin(theta))/cos(theta)^2) + (phidd*cos(phi))/cos(theta) +
        (thetadd*sin(phi)*sin(theta))/cos(theta)^2) + wyd*((2*phid*cos(phi))/cos(theta)
        + (2*thetad*sin(phi)*sin(theta))/cos(theta)^2) - wzd*((2*phid*sin(phi))/cos(theta)
        - (2*thetad*cos(phi)*sin(theta))/cos(theta)^2) + (wzdd*cos(phi))/cos(theta)
        + (wydd*sin(phi))/cos(theta);
xd(10) = 0;
xd(11) = 0;
xd(12) = 0;

% State transition matrix
F = zeros(12,12);
F(1,1) = sin(phi)*tan(theta)*(bz - gz) - cos(phi)*tan(theta)*(by - gy);
F(2,1) = cos(phi)*(bz - gz) + sin(phi)*(by - gy);
F(3,1) = (sin(phi)*(bz - gz))/cos(theta) - (cos(phi)*(by - gy))/cos(theta);
F(4,1) = wy*(thetad*cos(phi)*(tan(theta)^2 + 1) - phid*sin(phi)*tan(theta))
        - wz*(thetad*sin(phi)*(tan(theta)^2 + 1) + phid*cos(phi)*tan(theta))
        + wyd*cos(phi)*tan(theta) - wzd*sin(phi)*tan(theta);
F(5,1) = phid*wz*sin(phi) - wyd*sin(phi) - phid*wy*cos(phi) - wzd*cos(phi);
F(6,1) = (wyd*cos(phi))/cos(theta) - wz*((phid*cos(phi))/cos(theta) + (thetad*
         sin(phi)*sin(theta))/cos(theta)^2) - wy*((phid*sin(phi))/cos(theta) -
         (thetad*cos(phi)*sin(theta))/cos(theta)^2) - (wzd*sin(phi))/cos(theta);
F(7,1) = wyd*(2*thetad*cos(phi)*(tan(theta)^2 + 1) - 2*phid*sin(phi)*tan(theta))
        - wz*(phid*(thetad*cos(phi)*(tan(theta)^2 + 1) - phid*sin(phi)*tan(theta))
        + thetad*(phid*cos(phi)*(tan(theta)^2 + 1) + 2*thetad*sin(phi)*tan(theta)
        *(tan(theta)^2 + 1)) + thetadd*sin(phi)*(tan(theta)^2 + 1) + phidd*cos(phi)
        *tan(theta)) - wy*(phid*(thetad*sin(phi)*(tan(theta)^2 + 1) + phid*cos(phi)
        *tan(theta)) + thetad*(phid*sin(phi)*(tan(theta)^2 + 1) - 2*thetad*cos(phi)
        *tan(theta)*(tan(theta)^2 + 1)) - thetadd*cos(phi)*(tan(theta)^2 + 1) +
        phidd*sin(phi)*tan(theta)) - wzd*(2*thetad*sin(phi)*(tan(theta)^2 + 1) +
        2*phid*cos(phi)*tan(theta)) + wydd*cos(phi)*tan(theta) - wzdd*sin(phi)*tan(theta);
F(8,1) = wy*(sin(phi)*phid^2 - phidd*cos(phi)) - wydd*sin(phi) - wzdd*cos(phi)
        + wz*(cos(phi)*phid^2 + phidd*sin(phi)) - 2*phid*wyd*cos(phi) + 2*phid*wzd*sin(phi);
F(9,1) = wy*(thetad*((thetad*cos(phi))/cos(theta) + (2*thetad*cos(phi)*sin(theta)^2)
        /cos(theta)^3 - (phid*sin(phi)*sin(theta))/cos(theta)^2) - phid*((phid*cos(phi))
        /cos(theta) + (thetad*sin(phi)*sin(theta))/cos(theta)^2) - (phidd*sin(phi))/cos(theta)
```

```
        + (thetadd*cos(phi)*sin(theta))/cos(theta)^2) - wz*(thetad*((thetad*sin(phi))
        /cos(theta) + (2*thetad*sin(phi)*sin(theta)^2)/cos(theta)^3 + (phid*cos(phi)*
        sin(theta))/cos(theta)^2) - phid*((phid*sin(phi))/cos(theta) - (thetad*cos(phi)
        *sin(theta))/cos(theta)^2) + (phidd*cos(phi))/cos(theta) + (thetadd*sin(phi)
        *sin(theta))/cos(theta)^2) - wyd*((2*phid*sin(phi))/cos(theta) - (2*thetad*
        cos(phi)*sin(theta))/cos(theta)^2) - wzd*((2*phid*cos(phi))/cos(theta)
        + (2*thetad*sin(phi)*sin(theta))/cos(theta)^2) + (wydd*cos(phi))/cos(theta)
        - (wzdd*sin(phi))/cos(theta);
F(1,2) = - cos(phi)*(bz - gz)*(tan(theta)^2 + 1) - sin(phi)*(by - gy)*(tan(theta)^2 + 1);
F(3,2) = - (cos(phi)*sin(theta)*(bz - gz))/cos(theta)^2 - (sin(phi)*sin(theta)*
        (by - gy))/cos(theta)^2;
F(4,2) = wy*(phid*cos(phi)*(tan(theta)^2 + 1) + 2*thetad*sin(phi)*tan(theta)
        *(tan(theta)^2 + 1)) - wz*(phid*sin(phi)*(tan(theta)^2 + 1) - 2*thetad*
        cos(phi)*tan(theta)*(tan(theta)^2 + 1)) + wzd*cos(phi)*(tan(theta)^2 + 1)
        + wyd*sin(phi)*(tan(theta)^2 + 1);
F(6,2) = wz*((thetad*cos(phi))/cos(theta) + (2*thetad*cos(phi)*sin(theta)^2)/cos(theta)^3
        - (phid*sin(phi)*sin(theta))/cos(theta)^2) + wy*((thetad*sin(phi))/cos(theta)
        + (2*thetad*sin(phi)*sin(theta)^2)/cos(theta)^3 + (phid*cos(phi)*sin(theta))
        /cos(theta)^2) + (wzd*cos(phi)*sin(theta))/cos(theta)^2 + (wyd*sin(phi)*
        sin(theta))/cos(theta)^2;
F(7,2) = wz*(thetad*(2*thetad*cos(phi)*(tan(theta)^2 + 1)^2 - 2*phid*sin(phi)*
        tan(theta)*(tan(theta)^2 + 1) + 4*thetad*cos(phi)*tan(theta)^2*(tan(theta)^2 + 1))
        - phid*(phid*cos(phi)*(tan(theta)^2 + 1) + 2*thetad*sin(phi)*tan(theta)*(tan(theta)^2
        + 1)) - phidd*sin(phi)*(tan(theta)^2 + 1) + 2*thetadd*cos(phi)*tan(theta)*(tan(theta)^2
        + 1)) + wy*(thetad*(2*thetad*sin(phi)*(tan(theta)^2 + 1)^2 + 2*phid*cos(phi)*tan(theta)
        *(tan(theta)^2 + 1) + 4*thetad*sin(phi)*tan(theta)^2*(tan(theta)^2 + 1)) - phid*(phid
        *sin(phi)*(tan(theta)^2 + 1) - 2*thetad*cos(phi)*tan(theta)*(tan(theta)^2 + 1))+ phidd
        *cos(phi)*(tan(theta)^2 + 1) + 2*thetadd*sin(phi)*tan(theta)*(tan(theta)^2 + 1))+ wyd
        *(2*phid*cos(phi)*(tan(theta)^2 + 1) + 4*thetad*sin(phi)*tan(theta)*(tan(theta)^2 + 1))
        - wzd*(2*phid*sin(phi)*(tan(theta)^2 + 1) - 4*thetad*cos(phi)*tan(theta)*(tan(theta)^2
        + 1)) + wzdd*cos(phi)*(tan(theta)^2 + 1) + wydd*sin(phi)*(tan(theta)^2 + 1);

F(9,2) = wzd*((2*thetad*cos(phi))/cos(theta) + (4*thetad*cos(phi)*sin(theta)^2)/cos(theta)^3
        - (2*phid*sin(phi)*sin(theta))/cos(theta)^2) + wyd*((2*thetad*sin(phi))/cos(theta) +
        (4*thetad*sin(phi)*sin(theta)^2)/cos(theta)^3 + (2*phid*cos(phi)*sin(theta))
        /cos(theta)^2)+ wy*(phid*((thetad*cos(phi))/cos(theta) + (2*thetad*cos(phi)*
        sin(theta)^2)/cos(theta)^3 - (phid*sin(phi)*sin(theta))/cos(theta)^2) + thetad
        *((phid*cos(phi))/cos(theta) + (2*phid*cos(phi)*sin(theta)^2)/cos(theta)^3
        + (6*thetad*sin(phi)*sin(theta)^3)/cos(theta)^4 + (5*thetad*sin(phi)*sin(theta))
        /cos(theta)^2) + (thetadd*sin(phi)) /cos(theta) + (2*thetadd*sin(phi)*sin(theta)^2)
        /cos(theta)^3 + (phidd*cos(phi)*sin(theta))/cos(theta)^2) - wz*(phid*((thetad*
        sin(phi))/cos(theta) + (2*thetad*sin(phi)*sin(theta)^2)/cos(theta)^3 +
        (phid*cos(phi)*sin(theta))/cos(theta)^2) + thetad*((phid*sin(phi))/cos(theta)
```

```
        - (6*thetad*cos(phi)*sin(theta)^3)/cos(theta)^4 + (2*phid*sin(phi)*sin(theta)^2)
        /cos(theta)^3 - (5*thetad*cos(phi)*sin(theta))/cos(theta)^2) - (thetadd*cos(phi))
        /cos(theta) - (2*thetadd*cos(phi)*sin(theta)^2)/cos(theta)^3 + (phidd*sin(phi)
        *sin(theta))/cos(theta)^2) + (wzdd*cos(phi)*sin(theta))/cos(theta)^2
        + (wydd*sin(phi)*sin(theta))/cos(theta)^2;
F(4,4) = wy*cos(phi)*tan(theta) - wz*sin(phi)*tan(theta);
F(5,4) = - wz*cos(phi) - wy*sin(phi);
F(6,4) = (wy*cos(phi))/cos(theta) - (wz*sin(phi))/cos(theta);
F(7,4) = wy*(2*thetad*cos(phi)*(tan(theta)^2 + 1) - 2*phid*sin(phi)*tan(theta))
        - wz*(2*thetad*sin(phi)*(tan(theta)^2 + 1) + 2*phid*cos(phi)*tan(theta))
        + 2*wyd*cos(phi)*tan(theta) - 2*wzd*sin(phi)*tan(theta);
F(8,4) = 2*phid*wz*sin(phi) - 2*wyd*sin(phi) - 2*phid*wy*cos(phi) - 2*wzd*cos(phi);
F(9,4) = (2*wyd*cos(phi))/cos(theta) - wz*((2*phid*cos(phi))/cos(theta) +
         (2*thetad*sin(phi)*sin(theta))/cos(theta)^2) - wy*((2*phid*sin(phi))/cos(theta)
         - (2*thetad*cos(phi)*sin(theta))/cos(theta)^2) - (2*wzd*sin(phi))/cos(theta);
F(4,5) = wz*cos(phi)*(tan(theta)^2 + 1) + wy*sin(phi)*(tan(theta)^2 + 1);
F(6,5) = (wz*cos(phi)*sin(theta))/cos(theta)^2 + (wy*sin(phi)*sin(theta))/cos(theta)^2;
F(7,5) = wy*(2*phid*cos(phi)*(tan(theta)^2 + 1) + 4*thetad*sin(phi)*tan(theta)*(tan(theta)^2
        + 1)) - wz*(2*phid*sin(phi)*(tan(theta)^2 + 1) - 4*thetad*cos(phi)*tan(theta)*
        (tan(theta)^2 + 1)) + 2*wzd*cos(phi)*(tan(theta)^2 + 1) + 2*wyd*sin(phi)
        *(tan(theta)^2 + 1);
F(9,5) = wz*(thetad*(cos(phi)/cos(theta) + (2*cos(phi)*sin(theta)^2)/cos(theta)^3)
        + (thetad*cos(phi))/cos(theta) + (2*thetad*cos(phi)*sin(theta)^2)/cos(theta)^3
        - (2*phid*sin(phi)*sin(theta))/cos(theta)^2) + wy*(thetad*(sin(phi)/cos(theta)
        + (2*sin(phi)*sin(theta)^2)/cos(theta)^3) + (thetad*sin(phi))/cos(theta)
        + (2*thetad*sin(phi)*sin(theta)^2)/cos(theta)^3 + (2*phid*cos(phi)*sin(theta))
        /cos(theta)^2) + (2*wzd*cos(phi)*sin(theta))/cos(theta)^2 + (2*wyd*sin(phi)
        *sin(theta))/cos(theta)^2;
F(7,7) = wy*cos(phi)*tan(theta) - wz*sin(phi)*tan(theta);
F(8,7) = - wz*cos(phi) - wy*sin(phi);
F(9,7) = (wy*cos(phi))/cos(theta) - (wz*sin(phi))/cos(theta);
F(7,8) = wz*cos(phi)*(tan(theta)^2 + 1) + wy*sin(phi)*(tan(theta)^2 + 1);
F(9,8) = (wz*cos(phi)*sin(theta))/cos(theta)^2 + (wy*sin(phi)*sin(theta))/cos(theta)^2;
F(1,10) = -1;
F(1,11) = -sin(phi)*tan(theta);
F(2,11) = -cos(phi);
F(3,11) = -sin(phi)/cos(theta);
F(1,12) = -cos(phi)*tan(theta);
F(2,12) = sin(phi);
F(3,12) = -cos(phi)/cos(theta);

% Measurement vector
y = zeros(10,1);
```

```
y(1) = g*sin(theta);
y(2) = -g*cos(theta)*sin(phi);
y(3) = -g*cos(phi)*cos(theta);
y(4) = psi;
y(5) = bx + phid - psid*sin(theta);
y(6) = by + thetad*cos(phi) + psid*cos(theta)*sin(phi);
y(7) = bz - thetad*sin(phi) + psid*cos(phi)*cos(theta);
y(8) = phidd - psidd*sin(theta) - psid*thetad*cos(theta);
y(9) = psid*(phid*cos(phi)*cos(theta) - thetad*sin(phi)*sin(theta)) + thetadd*
       cos(phi) - phid*thetad*sin(phi) + psidd*cos(theta)*sin(phi);
y(10) = psidd*cos(phi)*cos(theta) - thetadd*sin(phi) - phid*thetad*cos(phi)
        - psid*(phid*cos(theta)*sin(phi) + thetad*cos(phi)*sin(theta));

% Measurement matrix
H = zeros(10,12);
H(2,1) = -g*cos(phi)*cos(theta);
H(3,1) = g*cos(theta)*sin(phi);
H(6,1) = psid*cos(phi)*cos(theta) - thetad*sin(phi);
H(7,1) = - thetad*cos(phi) - psid*cos(theta)*sin(phi);
H(9,1) = psidd*cos(phi)*cos(theta) - thetadd*sin(phi) - phid*thetad*cos(phi)
        - psid*(phid*cos(theta)*sin(phi) + thetad*cos(phi)*sin(theta));
H(10,1) = phid*thetad*sin(phi) - thetadd*cos(phi) - psid*(phid*cos(phi)*cos(theta)
        - thetad*sin(phi)*sin(theta)) - psidd*cos(theta)*sin(phi);
H(1,2) = g*cos(theta);
H(2,2) = g*sin(phi)*sin(theta);
H(3,2) = g*cos(phi)*sin(theta);
H(5,2) = -psid*cos(theta);
H(6,2) = -psid*sin(phi)*sin(theta);
H(7,2) = -psid*cos(phi)*sin(theta);
H(8,2) = psid*thetad*sin(theta) - psidd*cos(theta);
H(9,2) = - psid*(phid*cos(phi)*sin(theta) + thetad*cos(theta)*sin(phi))
         - psidd*sin(phi)*sin(theta);
H(10,2) = - psid*(thetad*cos(phi)*cos(theta) - phid*sin(phi)*sin(theta))
          - psidd*cos(phi)*sin(theta);
H(4,3) = 1;
H(5,4) = 1;
H(9,4) = psid*cos(phi)*cos(theta) - thetad*sin(phi);
H(10,4) = - thetad*cos(phi) - psid*cos(theta)*sin(phi);
H(6,5) = cos(phi);
H(7,5) = -sin(phi);
H(8,5) = -psid*cos(theta);
H(9,5) = - phid*sin(phi) - psid*sin(phi)*sin(theta);
H(10,5) = - phid*cos(phi) - psid*cos(phi)*sin(theta);
```

```
H(5,6) = -sin(theta);
H(6,6) = cos(theta)*sin(phi);
H(7,6) = cos(phi)*cos(theta);
H(8,6) = -thetad*cos(theta);
H(9,6) = phid*cos(phi)*cos(theta) - thetad*sin(phi)*sin(theta);
H(10,6) = - phid*cos(theta)*sin(phi) - thetad*cos(phi)*sin(theta);
H(8,7) = 1;
H(9,8) = cos(phi);
H(10,8) = -sin(phi);
H(8,9) = -sin(theta);
H(9,9) = cos(theta)*sin(phi);
H(10,9) = cos(phi)*cos(theta);
H(5,10) = 1;
H(6,11) = 1;
H(7,12) = 1;
```

Appendix B

Ballbot Dynamics with AutoLev

This appendix includes the code used to generate the equations of motion for the ballbot designed in Chapter 4, modeled in Chapter 5. Additionally, the code also prints out and displays the equations in a suitable file format.

```
% Autolev Code for the BallBot with Lagrange method
% Date:  August 2016
% Problem:  Kinematics / Dynamics of the BallBot
%--------------------------------------------------------------------
%       Default Settings
Digits       7               % Significant digits
AutoEpsilon 1.0E-14
AutoZ ON
%--------------------------------------------------------------------
%       Newtonian(Inertial), bodies, frames, particles, points
Newtonian    N                       % Newtonian reference frame
Bodies       K, W{3}, A              % Ball, wheels , body
% Point fixed in N, O center of N, PK ball-ground touching
% PW ball omniwhell touching, M intersection of Wi3 vectors
Points       O, PK, PW{3}, M
Frames       OW{3},L,Asub % Declares masless reference frames
%--------------------------------------------------------------------
%       Variables, constants, and specified
%       phi: angle of ball, theta: angle of body, q: motor angles
MotionVariables' x'', y'', phi{3}'' , theta{3}'', q{3}''
```

```
Constants MA            % Mass of body and omniwheels
Constants MK            % Mass of ball
Constants MW            % Mass of each omniwheel
Constants rK            % Radius of the ball
Constants rW            % Radius of the Omniwheel
Constants rA            % Radius of the body
Constants l             % Distance between center of the ball an center of the body
Constants g             % Gravitational acceleration
Constants IK            % Inertia of ball in the intertial reference frame I or L
Constants IAX           % Inertia of body and omniwheels in the body reference frame A
Constants IAY           % Inertia of body and omniwheels in the body reference frame A
Constants IAZ           % Inertia of body and omniwheels in the body reference frame A
Constants IW            % Inertia of each omniwheel and motor about the motor axis
Specified T{3}          % Declares T as a func of time, constants and variables
Specified Td{3}         % Disturbance force acting on the body

Constants a=pi*45/180   % Motor arrangement angle alpha
Constants b1=0, b2=2*pi/3, b3=4*pi/3  % Angle of the motors

%-----------------------------------------------------------------------
%       Mass and Inertias
Mass            K=MK, W1=MW, W2=MW, W3=MW, A=MA
Inertia         A,IAX, IAY, IAZ
Inertia         K, IK, IK, IK
Inertia         W1, 0, 0, IW
Inertia         W2, 0, 0, IW
Inertia         W3, 0, 0, IW
%-----------------------------------------------------------------------
%       Geometry relating unit vectors
Simprot(N,K,1,0)
Simprot(N,L,3,theta3)
Simprot(L,Asub,2,theta2)
Simprot(Asub,A,1,theta1)
Simprot(A,W1,2,a+pi/2)
Simprot(W1,W1,2,b1)
Simprot(W1,W2,A3>,b2)
Simprot(W1,W3,A3>,b3)

%-----------------------------------------------------------------------
%       Position vectors
P_O_Ko> = x*N1> + y*N2> + rK*N3> % Position of the center of the ball
P_Ko_Ao> = l*A3>                % Position of center of body wrt ball center
P_Ko_M> = (rW+rK)/cos(a)*A3>    % Position of motor intersection wrt ball center
```

```
P_M_W1o> = (rK+rW)*tan(a)*W13>   % Position of omniwheel center wrt M
P_M_W2o> = (rK+rW)*tan(a)*W23>
P_M_W3o> = (rK+rW)*tan(a)*W33>
P_Ko_PK> = -rK*N3>               % Position of the Ball-ground touching point
P_Ko_PW1> = rK/(rK+rW)*P_Ko_W1o> % Vector from ball center to omniwheel touching points
P_Ko_PW2> = rK/(rK+rW)*P_Ko_W2o>
P_Ko_PW3> = rK/(rK+rW)*P_Ko_W3o>

%----------------------------------------------------------------------
%       Angular velocities
Angvel(N,A)  % Rotation frames ang vel declaration
Angvel(N,L)
Angvel(N,Asub)

W_K_L> = phi1'*L1> + phi2'*L2> % +phi3'*L3> Ball rotation on z axis is ignored

W_W1_A> = q1'*W13>
W_W2_A> = q2'*W23>
W_W3_A> = q3'*W33>

q1d = dot(W_W1_A>, W13>)  % Motor speed with omniwheel
q2d = dot(W_W2_A>, W23>)
q3d = dot(W_W3_A>, W33>)

ALF_W1_A> = dt(W_W1_A>,W1)
ALF_W2_A> = dt(W_W2_A>,W2)
ALF_W3_A> = dt(W_W3_A>,W3)

q1dd = dot(ALF_W1_A>, W13>) % Motor acceleration with omniwheel
q2dd = dot(ALF_W2_A>, W23>)
q3dd = dot(ALF_W3_A>, W33>)

wbx=dot(W_A_N>, A1>)  %  body axes angular velocities
wby=dot(W_A_N>, A2>)
wbz=dot(W_A_N>, A3>)

ALF_A_N> = dt(W_A_N>,N)

abx=dot(ALF_A_N>, A1>)  %  body axes angular acceleration
aby=dot(ALF_A_N>, A2>)
abz=dot(ALF_A_N>, A3>)

%----------------------------------------------------------------------
```

```
%       Velocities
V_Ko_N> = dt( P_O_Ko>, N )  % Ball center inertial velocity
V_PK_N> = dt( P_O_PK>, N )  % Bal-ground touching point inertial velocity
V_PK_K> = dt( P_Ko_PK>, K)

V_W1o_N> = dt( P_O_W1o>, N )  % Wheel center inertial velocity
V_W2o_N> = dt( P_O_W2o>, N )
V_W3o_N> = dt( P_O_W3o>, N )

v2pts(N, A, Ko, Ao)

%-----------------------------------------------------------------------
%       Motion Constrains
Dependent[1]= dot(V_PK_N>-V_PK_K>,N1>)  % = 0
Dependent[2]= dot(V_PK_N>-V_PK_K>,N2>)  % = 0
ad1>=A2>
ad2>=-sin(b2)*A1> + cos(b2)*A2>
ad3>=-sin(b3)*A1> + cos(b3)*A2>
Dependent[3]=dot(cross(W_K_A>,P_Ko_PW1>),ad1>)- q1'*rW
Dependent[4]=dot(cross(W_K_A>,P_Ko_PW2>),ad2>)- q2'*rW
Dependent[5]=dot(cross(W_K_A>,P_Ko_PW3>),ad3>)- q3'*rW

Constrain(Dependent[x',y',q1',q2',q3'])
%-----------------------------------------------------------------------
%       Forces & Torques
Gravity( -g*N3> )
Torque(A/W1,T1*W13>)
Torque(A/W2,T2*W23>)
Torque(A/W3,T3*W33>)

Torque_A> += Td1*A1> + Td2*A2> + Td3*A3>

%-----------------------------------------------------------------------
%       Energies
KE= KE()
PE= (dot(N3>,P_Ko_Ao>)*mA +
    (dot(N3>,P_Ko_W1o>)+dot(N3>,P_Ko_W2o>)+dot(N3>,P_Ko_W3o>))*mW)*g
Vir_W=dot(Torque_W1>,W_W1_A>)+dot(Torque_W2>,W_W2_A>)
      +dot(Torque_W3>,W_W3_A>)+dot(Torque_A>,W_A_N>)
VirW_phi1=d(Vir_W,phi1')
VirW_phi2=d(Vir_W,phi2')
VirW_theta1=d(Vir_W,theta1')
VirW_theta2=d(Vir_W,theta2')
```

```
VirW_theta3=d(Vir_W,theta3')

%--------------------------------------------------------------------
%       Equations of motion
Digits 3
Lag[1] =  dt(d(KE,phi1')) - d(KE, phi1) + d(PE,phi1) - VirW_phi1
Lag[2] =  dt(d(KE,phi2')) - d(KE, phi2) + d(PE,phi2) - VirW_phi2
Lag[3] =  dt(d(KE,theta1')) - d(KE, theta1) + d(PE,theta1) - VirW_theta1
Lag[4] =  dt(d(KE,theta2')) - d(KE, theta2) + d(PE,theta2) - VirW_theta2
Lag[5] =  dt(d(KE,theta3')) - d(KE, theta3) + d(PE,theta3) - VirW_theta3

Lag := Evaluate(Lag, L=0.185090, MA=2.39615, MK=3, MW=0.25, g=9.81,&
                     rK=0.108225361302, rW=0.05, IK=0.019814, IW=0.003329,&
                     IAX=0.10354, IAY=0.10354, IAZ=0.020929) % Real Values

Solve(Lag,[phi1'',phi2'',theta1'',theta2'',theta3''])

Encode phi1'',phi2'',theta1'',theta2'',theta3'', wbx, wby, wbz,&
       abx, aby, abz, q1d, q2d, q3d, q1dd, q2dd, q3dd

CODE Algebraic() bb_Lagrange_RT.m % Save Resulting Code as m file
CODE Algebraic() bb_Lagrange_RT.c % Save Resulting Code as C Code

%*********************************************************************
%                        LINEAR MODEL
%*********************************************************************

%       Linearization: Perturbation variables
Variables  dphi{2}''    % Perturbations of phi, phi', phi''
Variables  dthe{3}''    % Perturbations of theta, theta', theta''
Variables  dT{3}        % Perturbations of Torques
Imaginary  i            % Imaginary number

Perturb = Taylor(Lag, 1,   phi1=0:dphi1,phi2=0:dphi2,&
                           phi1'=0:dphi1',phi2'=0:dphi2',&
                           phi1''=0:dphi1'',phi2''=0:dphi2'', &
                           theta1=0:dthe1,theta2=0:dthe2,theta3=0:dthe3, &
                           theta1'=0:dthe1',theta2'=0:dthe2',theta3'=0:dthe3', &
                           theta1''=0:dthe1'',theta2''=0:dthe2'',theta3''=0:dthe3'', &
                           T1=0:dT1, T2=0:dT2,T3=0:dT3)
Solve( Perturb, dphi1'',dphi2'', dthe1'',dthe2'',dthe3'')

%--------------------------------------------------------------------
```

```
%       Form matrix of peturbations and its time-derivative
%       Matrix of perturbations
Xm = [dphi1,dphi2,dthe1,dthe2,dthe3,dphi1',dphi2',dthe1',dthe2',dthe3']
%       Time derivative of Xm
Xp = [dphi1';dphi2';dthe1';dthe2';dthe3';dphi1'';dphi2'';dthe1'';dthe2'';dthe3'']
Ut = [dT1,dT2,dT3]
%-----------------------------------------------------------------------
%       Form matrices A and B such that Xm' = A * Xm + B * u
A = D(Xp, Xm)
B = D(Xp, Ut)
Encode A,B
Code Algebraic() ABmatrix_RT.m % Save A & B matrices

% End
```

Bibliography

[1] M. Kumagai and T. Ochiai, "Development of a robot balancing on a ball," in *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, pp. 433–438, IEEE, 2008.

[2] L. Hertig, D. Schindler, M. Bloesch, C. D. Remy, and R. Siegwart, "Unified state estimation for a ballbot," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 2471–2476, IEEE, 2013.

[3] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[4] R. G. Brown and P. Y. Hwang, "Introduction to random signals and applied kalman filtering: with matlab exercises and solutions," *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions, by Brown, Robert Grover.; Hwang, Patrick YC New York: Wiley, c1997.*, vol. 1, 1997.

[5] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, "A high integrity imu/gps navigation loop for autonomous land vehicle applications," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 572–578, 1999.

[6] J. Wendel, O. Meister, C. Schlaile, and G. F. Trommer, "An integrated gps/mems-imu navigation system for an autonomous helicopter," *Aerospace Science and Technology*, vol. 10, no. 6, pp. 527–533, 2006.

[7] S. H. Jeong, S. Jung, and M. Tomizuka, "Attitude control of a quad-rotor system using an acceleration-based disturbance observer: An empirical approach," in *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 916–921, IEEE, 2012.

[8] G. Wahba, "A least squares estimate of satellite attitude," *SIAM review*, vol. 7, no. 3, pp. 409–409, 1965.

[9] F. L. Markley, "Attitude determination using vector observations and the singular value decomposition," *The Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, 1988.

[10] M. D. Shuster and S. Oh, "Three-axis attitude determination from vector observations," *Journal of Guidance, Control, and Dynamics*, vol. 4, no. 1, pp. 70–77, 1981.

[11] J. L. Farrell, "Attitude determination by kalman filtering," *Automatica*, vol. 6, no. 3, pp. 419–430, 1970.

[12] F. L. Markley, "Attitude error representations for kalman filtering," *Journal of guidance, control, and dynamics*, vol. 26, no. 2, pp. 311–317, 2003.

[13] A. Kim and M. Golnaraghi, "A quaternion-based orientation estimation algorithm using an inertial measurement unit," in *Position Location and Navigation Symposium, 2004. PLANS 2004*, pp. 268–272, IEEE, 2004.

[14] A. M. Sabatini, "Estimating three-dimensional orientation of human body parts by inertial/magnetic sensing," *Sensors*, vol. 11, no. 2, pp. 1489–1525, 2011.

[15] W. Li and J. Wang, "Effective adaptive kalman filter for mems-imu/magnetometers integrated attitude and heading reference systems," *Journal of Navigation*, vol. 66, no. 01, pp. 99–113, 2013.

[16] E. Edwan, S. Knedlik, and O. Loffeld, "Constrained angular motion estimation in a gyro-free imu," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 1, pp. 596–610, 2011.

[17] J. L. Crassidis, F. L. Markley, and Y. Cheng, "Survey of nonlinear attitude estimation methods," *Journal of guidance, control, and dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[18] J. Stuelpnagel, "On the parametrization of the three-dimensional rotation group," *SIAM review*, vol. 6, no. 4, pp. 422–430, 1964.

[19] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[20] P. Petkov and T. Slavov, "Stochastic modeling of mems inertial sensors," *Cybernetics and information technologies*, vol. 10, no. 2, pp. 31–40, 2010.

[21] Y. Hori, "Disturbance suppression on an acceleration control type dc servo system," in *Power Electronics Specialists Conference, 1988. PESC'88 Record., 19th Annual IEEE*, pp. 222–229, IEEE, 1988.

[22] P. B. Schmidt and R. D. Lorenz, "Design principles and implementation of acceleration feedback to improve performance of dc drives," *Industry Applications, IEEE Transactions on*, vol. 28, no. 3, pp. 594–599, 1992.

[23] J. Han, Y. He, and W. Xu, "Angular acceleration estimation and feedback control: An experimental investigation," *Mechatronics*, vol. 17, no. 9, pp. 524–532, 2007.

[24] J.-C. Lu and P.-C. Lin, "State derivation of a 12-axis gyroscope-free inertial measurement unit," *Sensors*, vol. 11, no. 3, pp. 3145–3162, 2011.

[25] J.-H. Chen, S.-C. Lee, and D. B. DeBra, "Gyroscope free strapdown inertial measurement unit by six linear accelerometers," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 2, pp. 286–290, 1994.

[26] E. Edwan, S. Knedlik, and O. Loffeld, "Angular motion estimation using dynamic models in a gyro-free inertial measurement unit," *Sensors*, vol. 12, no. 5, pp. 5310–5327, 2012.

[27] T. Endo and Y. Nakamura, "An omnidirectional vehicle on a basketball," in *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on*, pp. 573–578, IEEE, 2005.

[28] L. Havasi, "Errosphere: an equilibrator robot," in *Control and Automation, 2005. ICCA'05. International Conference on*, vol. 2, pp. 971–976, IEEE, 2005.

[29] T. Lauwers, G. A. Kantor, and R. L. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2884–2889, IEEE, 2006.

[30] U. Nagarajan, A. Mampetta, G. A. Kantor, and R. L. Hollis, "State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 998–1003, IEEE, 2009.

[31] U. Nagarajan, B. Kim, and R. Hollis, "Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 130–135, IEEE, 2012.

[32] U. Nagarajan, *Fast and Graceful Balancing Mobile Robots*. CARNEGIE MELLON UNIVERSITY, 2012.

[33] J. Fong, S. Uppill, and B. Cazzolato, "899: Ballbot," 2009.

[34] S. Doessegger, P. Fankhauser, C. Gwerder, J. Huessy, J. Kaeser, T. Kammermann, L. Limacher, and M. Neunert, "Rezero," *Focus Project Report, Autonomous Systems Lab., ETH Zurich, Switzerland*, 2010.

[35] M. Kumagai, "Development of a ball drive unit using partially sliding rollersan alternative mechanism for semi-omnidirectional motion," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 3352–3357, IEEE, 2010.

[36] A. Bhatia, M. Kumagai, and R. Hollis, "Six-stator spherical induction motor for balancing mobile robots," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 226–231, IEEE, 2015.

[37] G. Seyfarth, A. Bhatia, O. Sassnick, M. Shomin, M. Kumagai, and R. Hollis, "Initial results for a ballbot driven with a spherical induction motor," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3771–3776, May 2016.

[38] S. Leutenegger and P. Fankhauser, "Modeling and control of a ballbot," *Bachelor thesis, ETH Zurich*, 2010.

[39] A. N. Inal, Ö. Morgül, and U. Saranlı, "A 3d dynamic model of a spherical wheeled self-balancing robot," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5381–5386, IEEE, 2012.

[40] U. Nagarajan, G. Kantor, and R. L. Hollis, "Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 3743–3748, IEEE, 2009.

[41] A. N. Inal, O. Morgul, and U. Saranli, "Path following with an underactuated self-balancing spherical-wheel mobile robot," in *Advanced Robotics (ICAR), 2015 International Conference on*, pp. 194–199, IEEE, 2015.

[42] R. A. Garcia-Garcia and M. Arias-Montiel, "Linear controllers for the nxt ball-bot with parameter variations using linear matrix inequalities [lecture notes]," *IEEE Control Systems*, vol. 36, no. 3, pp. 121–136, 2016.

[43] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[44] R. P. Collinson, *Introduction to avionics systems*. Springer Science & Business Media, 2013.

[45] T. R. Kane and D. A. Levinson, *Solution Manual for Problem Sets for Dynamics Online: Theory and Implementation with AUTOLEV*. OnLine Dynamics, 2000.