UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Tomás Lopes de Oliveira

State estimation for low-cost Remotely Piloted Aircraft (RPA) using Extended Kalman Filters

São Carlos

2019

Tomás Lopes de Oliveira

State estimation for low-cost Remotely Piloted Aircraft (RPA) using Extended Kalman Filters

Monografia apresentada ao Curso de Engenharia Aeronáutica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Aeronáutico.

Supervisor: Prof. Dr. Glauco Augusto de Paula Caurin
Co-supervisor: Prof. Dr. Marco Henrique Terra

São Carlos

2019

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Lopes de Oliveira, Tomás
L655s State estimation for low-cost Remotely Piloted Aircraft(RPA) using Extended Kalman Filters / Tomás Lopes de Oliveira; orientador Glauco Augusto de Paula Caurin; coorientador Marco Henrique Terra. São Carlos, 2019.

Monografia (Graduação em Engenharia Aeronáutica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.

1. Filtro de Kalman. 2. Aeronave Remotamente Pilotada (RPA). 3. Estimação de estados. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Candidato: Tomás Lopes de Oliveira
Título do TCC: State Estimation for low-cost Remotely Piloted Aircraft (RPA) using extended kalman filters
Data de defesa: 29/11/2019

Comissão Julgadora	**Resultado**
Professor Doutor Jorge Henrique Bidinotto	Aprovado
Instituição: EESC - SAA	
Professor Doutor Ricardo Afonso Angélico	APROVADO
Instituição: EESC - SAA	

Presidente da Banca: Professor Doutor Jorge Henrique Bidinotto

(assinatura)

Este trabalho é dedicado à minha familília, aos meus amigos e a todas os paulistas e brasileiros que contribuiram para meus estudos direta ou indiretamente.

ACKNOWLEDGEMENTS

First I would like thank professor Glauco Augusto de Paula Caurin, professor Marco Henrique Terra for all their support in guiding me through this project which concludes my path in becoming an engineer. I also would like to thank professor Roberto Santos Inoue of the federal university of São Carlos for his great contributions to this work.

Finally, I would like to thank Xmobots CEO Giovani Amianti for providing me with data from one of the company's Remotely Piloted Aircraft, which greatly served to complete this work with a real case application.

"O estudo, a busca da verdade e da beleza são domínios em que nos é consentido sermos crianças por toda a vida."

Albert Einstein

ABSTRACT

Oliveira, T.L. **State estimation for low-cost Remotely Piloted Aircraft (RPA) using Extended Kalman Filters**. 2019. 93p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

An important step in reducing the costs of Remotely Piloted Aircraft (RPA) is developing state estimation algorithms that are able to deal with low-cost sensors. This work focused in studying and implementing estimators for the navigation states (orientation, position and velocity) of low-cost RPAs by the use of error state kalman filter and an extended kalman filter. The orientation filter used an Error State Kalman Filter, while the position and velocity filter used a regular Extended Kalman Filter. All filters followed a coherent tuning methodology as Allan Variance method was applied to estimate sensor noise and thus more properly adapt the filter to the sensor characteristics. Both filters were coded in MATLAB language and had their results analyzed. Tests used simulated and real data and gave satisfactory results for a first developing phase of an estimator.

Keywords: State Estimation, Remotely Piloted Aircraft, Extended Kalman filter.

ABSTRACT

Oliveira, T.L. **State estimation for low-cost Remotely Piloted Aircraft (RPA) using Extended Kalman Filters**. 2019. 93p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2019.

Uma etapa importante na redução dos custos de uma aeronave remotamente pilotadas (RPA) é o desenvolvimento de algorítimos para estimação de estado capazes de lidar com sensores de baixo custo. Este trabalho focou-se no estudo e implementação de estimadores para os estados navegação de RPAs de baixo custo com o uso de técnicas de filtragem de Kalman. O filtro de orientação usou um filtro de kalman de erro de estado enquanto que o filtro de posição e velocidade usou um filtro de Kalman estendido. Todos os filtros seguiram uma metodologia coerente de ajuste de parâmetros, aplicando a técnica de variância de Allan para estimar os ruídos dos sensores. Desse modo os filtros puderam ser melhor adaptados às características dos sensores utilizados. Ambos os filtros foram implementados em linguagem MATLAB e tiveram seus resultados analisados. Em testes, dados simulados e dados reais de uma RPA foram utlizados. Os resultados se mostraram satisfatórios para uma primeira fase de desenvolvimento de um estimador.

Palavras-chave: Estimação de estados, Filtros de Kalman, Aeronaves Remotamente Pilotadas.

LIST OF FIGURES

Figure 1 – General Architecture of an RPA system: RPA + GS. Source (XMOBOTS, 2019b) . . . 27
Figure 2 – Body frame (ETKIN; REID, 1959) . . . 41
Figure 3 – ECEF *Earth-Centered, Earth fixed* (ECEF) frame (WIKIPEDIA, 2019) 42
Figure 4 – Definition of the navigation frame in the geographic frame. (LOTFI, 2015) . . . 42
Figure 5 – Allan Variance log-log plot for different types of error sources (BOARD, 1998). . . . 48
Figure 6 – Allan deviation log-log plot for one of the axis of the gyroscope sensor . 48
Figure 7 – Allan deviation bounding for x-axis of gyroscope sensor RPA sensor. . 50
Figure 8 – Xmobots fixed wing RPA Arator 5B. Source (XMOBOTS, 2019a). . . . 65
Figure 9 – Generated body rates . . . 67
Figure 10 – Generated magnetometer signal . . . 68
Figure 11 – Generated accelerometer signal . . . 68
Figure 12 – Roll angle comparison . . . 69
Figure 13 – Pitch angle comparison . . . 69
Figure 14 – Yaw angle comparison . . . 70
Figure 15 – Time angle error analysis for simulated trajectory for the ESKF orientation filter. . . . 70
Figure 16 – Standard deviation of the attitude error norm compared to two times the trace of the state covariance matrix $\mathbf{P}$ ($2\sigma = 2\sqrt{Tr(\mathbf{P}_k)}$), taken at a given realization. . . . 71
Figure 17 – Roll angle comparison . . . 72
Figure 18 – Pitch angle comparison as a function of time in seconds. . . . 72
Figure 19 – Yaw angle comparison as a function of time in seconds. . . . 73
Figure 20 – EKF position comparison with GPS . . . 74
Figure 21 – EKF position comparison with GNSS zoomed . . . 74
Figure 22 – EKF north velocity comparison with GNSS as a function of time in seconds. . . . 75
Figure 23 – EKF east velocity comparison with GNSS as a function of time in seconds. 75
Figure 24 – EKF down velocity comparison with GNSS as a function of time in seconds. . . . 76
Figure 25 – EKF accelerometer bias estimation as a function of time in seconds. . . 76
Figure 26 – Simulation of GNSS hazard. . . . 77
Figure 27 – Time averaging of inertial sensor output . . . 89

LIST OF TABLES

Table 1 – Xmobots Arator 5B technical specifications. Source (XMOBOTS, 2019a) 65
Table 2 – Values of σ^2 and τ for each sensor. 66

LIST OF ABBREVIATIONS AND ACRONYMS

EKF	Extended Kalman Filter
ESKF	Error State Kalman Filter
KF	Kalman Filter
RPA	Remotely Piloted Aircraft
RPAS	Remotely Piloted Aircraft System
TCC	Trabalho de Conclusão de Curso

LIST OF SYMBOLS

$\beta^2(\tau)$	Allan Variance
$\beta(\tau)$	Allan deviation
$\mathbf{x}_a$	Accelerometer bias
$\mathbf{x}_g$	Gyroscope bias
$\mathbf{x}_m$	Magnetometer bias
$\delta\mathbf{x}_a$	Accelerometer bias error
$\delta\mathbf{x}_g$	Gyroscope bias error
$\delta\mathbf{x}_m$	Magnetometer bias error
b_1	First order Gauss-Markov process
$\bar{\mathbf{x}}$	Nominal state
$\delta\mathbf{x}$	Error state
$\hat{\mathbf{x}}$	Estimated state
$\mathbf{y}_k$	sensor output or measurement at time index k
$\mathbf{y}_a$	Accelerometer output
$\mathbf{y}_m$	Magnetometer output
$\mathbf{u}_g$	Gyroscope output
$\delta\mathbf{y}$	Measurement residue
$\delta\mathbf{m}$	Earth magnetic field measurement residue
$\delta\mathbf{g}$	Earth gravitational field measurement residue
$\mathbf{g}^n$	Earth gravitational field represented in the navigation frame
$\mathbf{g}^b$	Earth gravitational field represented in the body frame
$\mathbf{m}^n$	Earth magnetic field represented in the navigation frame
$\mathbf{m}^b$	Earth magnetic field represented in the body frame
$\mathbf{p}$	Position vector, including latitude, longitude and height

$\mathbf{v}_e^g$	Velocity vector from Earth ECEF frame to geographic frame
$\mathbf{v}_{NED}$	Velocity vector with components along the North (v_n), East (v_e) and Down (v_d) directions
R_N	Earth normal radius
R_M	Earth meridian radius
a	Earth equatorial radius
e	Earth's elliptical eccentricity
ω_{ie}	Angular rate between the Earth fixed frame ECEF (e) and the inertial frame (i), represented in the inertial frame.
ϕ	Latitude
λ	Longitude
h	Height
θ	Pitch
ϕ	Roll
ψ	Yaw
p	Body angular rate around x axis
q	Body angular rate around y axis
r	Body angular rate around z axis
ω_{nb}^b	Body angular rate vector from the navigational frame (n) to the body frame (b), represented in the body frame.
$\mathbf{w}$	Body angular rate vector
$\mathbf{\Omega}$	Body rate matrix (see Appendix B.2)
$\mathbf{W}$	Body rate matrix integrated (see Appendix B.2)
$\mathbf{q}_a^b$	Quaternion representing rotation from frame a to frame b.
$\mathbf{R}_a^b$	Rotation matrix from frame a to frame b
$\mathbf{v}_k$	Measurement noise at instant t_k
$\mathbf{w}_k$	Process noise at instant t_k

$\mathbf{H}_k$	Measurement matrix calculated at instant t_k
$\mathbf{K}_k$	Kalman gain matrix calculated at instant t_k
$\mathbf{P}_k$	State covariance matrix calculated at instant t_k
$\mathbf{P}_k^-$	*a priori* state covariance matrix calculated at instant t_k
$\mathbf{P}_k^+$	*a posteriori* state covariance matrix calculated at instant t_k
$\mathbf{Q}$	Continuous process noise covariance matrix
$\mathbf{Q}_k$	Discrete process noise covariance matrix calculated at instant t_k
$\mathbf{R}_k$	Measurement noise covariance matrix calculated at instant t_k
$\mathbf{R}$	Continuous measurement noise covariance matrix
$\mathbf{S}_k$	Residue covariance matrix calculated at instant t_k
$\mathbf{S}_k$	Residue covariance matrix calculated at instant t_k
σ	Standard deviation
σ^2	Variance
$\bar{\mathbf{\Omega}}$	Sensor output rate
T	Sample time
t_k	time instant indexed k
τ	Correlation time.

CONTENTS

1 INTRODUÇÃO 27
1.1 Relevance of state estimation in RPA context 28
1.1.1 The Remotely Piloted Aircraft context 28
1.1.2 State estimation for RPA 29
1.2 Objectives 30
1.3 Literature review 30
1.3.1 Discrete Kalman Filters 30
1.3.1.1 Extended Kalman Filter (EKF) in aerospace applications 34
1.3.2 The Error State Extended Kalman Filter (ESKF) 35
1.3.3 Sensor models used in Kalman filters 37
1.3.3.1 Inertial sensors 38
1.3.3.2 Types of Kalman filters in respect to models 39
1.4 Scientific methodology 39

2 DEVELOPMENT 41
2.1 Reference Frames 41
2.2 Equations of motion 43
2.2.1 Orientation 43
2.2.2 Position 44
2.3 Measurement Models 46
2.3.1 Inertial Sensors 46
2.3.2 GNSS sensor model 46
2.3.3 Co-variance Matrix Estimation: Allan Variance method 47
2.4 Orientation Estimator 51
2.4.1 Nominal state equations 51
2.4.1.1 Quaternion propagation 51
2.4.1.2 Bias propagation 52
2.4.2 System Error Model 52
2.4.2.1 Tilt angle error model. 52
2.4.2.2 Bias error models. 54
2.4.2.3 Complete orientation error model. 54
2.4.3 Measurement Error Model 55
2.4.3.1 Accelerometer Error model 55
2.4.3.2 Magnetometer Error model 57
2.4.3.3 Complete Orientation Error model 57
2.4.4 The Algorithm 58

2.5 Position Estimator . . . 60
2.5.1 System Model . . . 60
2.5.1.1 Measurement Model . . . 61
2.5.2 The Algorithm . . . 62

3 RESULTS . . . 65
3.1 Co-variance matrix estimation for an RPA: test and results . . . 65
3.2 Orientation Estimator . . . 66
3.2.1 Simulated data test . . . 66
3.2.2 Real flight data . . . 71
3.3 Position Estimator . . . 71
3.3.1 Real flight data . . . 73

4 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK 79
4.1 Conclusion . . . 79
4.2 Future work . . . 79
4.2.1 Combining Position and Orientation . . . 80
4.2.2 Hard Iron and Soft Iron Estimation . . . 81

BIBLIOGRAPHY . . . 83

APPENDIX 87

APPENDIX A – ALLAN VARIANCE CALCULATION . . . 89

APPENDIX B – ROTATION ALGEBRA . . . 91
B.1 Quaternion small angular pertubations . . . 91
B.2 Quaternion integration . . . 91
B.3 Rotation matrix equivalent of quaternion . . . 92

1 INTRODUCTION

Overview

This project is part of the last year of aeronautical engineering program of the University of São Paulo (USP). It is thus an important moment for the senior student to apply his acquired knowledge to a engineering problem in a real context.

In this work, the word *drone* is used to refer generically to aerial unmanned vehicle (UAV). Following the definition found in (ANAC, 2017), Drones can be divided in to two classes according to their purposes: airmodels, used for recreational purposes and Remotely Piloted Aircraft (RPA) used for non recreational purposes, such as commercial and experimental (ANAC, 2017).

In many cases, such as in (DECEA, 2019), Remotely Piloted Aircraft are not referred as an aircraft alone, but as greater system including a Ground Station (GS) and the aircraft (Figure 1).

Figure 1: General Architecture of an RPA system: RPA + GS. Source (XMOBOTS, 2019b)

This report is divided in four main parts. In the first, the context of RPA is introduced and a bibliographical review of EKF estimation methods is conducted. It finishes describing the used methodology. In the second, the methods used to execute this work are studied and implementations are described and explained. In the third, the results of the methods are presented, following with observations. Finally in the fourth, possible improvements to the work are presented followed by the conclusion section with general remarks about the project.

1.1 Relevance of state estimation in RPA context

1.1.1 The Remotely Piloted Aircraft context

The growth of the usage of Remotely-Piloted Aircraft (RPA) is a worldwide trend as seen in recent forecasts (FAA, 2019) and (SESAR, 2016). In Brazil this trend is not different. In the year of 2018 there were 57 thousand registered drones in the registration system of Brazil's National Civil Aviation Agency (ANAC) (DECEA, 2018), with at least 25.000 aircraft for professional purposes which are operated by over 3.600 registered pilots (IBAS, 2019).

In order to understand this growth, it is necessary to consider some of the drone industry's particularities. Unlike the traditional aerospace industry, RPA industry also includes new economic players emerged from start-ups and academic environment (SESAR, 2016). By using cheaper sensors, these actors are capable of offering competitive prices for their drone platforms (SESAR, 2016). This paves the way for drone-based businesses, often specialized in services such as remote sensing, data processing and data analytics (SESAR, 2016). These businesses are able acquire data with lower operational costs when compared to airplanes, specially in smaller areas, and offer new data acquisition possibilities in remote and dangerous territory (EISENBEISS, 2009).

Another remarkable trend from the RPA industry is the use of open-source firmware and hardware. By openly sharing computer codes, large internet-based communities (i.g. (ARDUPILOT, 2019a)) have been developing firmware for RPAs open hardware (EBEID; SKRIVER; JIN, 2017). Through the efforts of these active collaborative projects, algorithms get implemented, tested and improved, serving as a usable platform for research, industry and hobbyists (EURE et al., 2013), (EBEID; SKRIVER; JIN, 2017), (ARDUPILOT, 2019c). Therefore, this paradigm is capable of adding another effect to the RPA industry which is the reduction of initial development costs. An inevitable consequence of the growth of the number of actors in this industry is the need to keep operations safe. This mobilizes on one side government institutions in creating certification requirements and laws and on the other side challenges the industry to respond with engineered solutions respecting those rules. Even though legislation in the RPA is relatively recent and in early development phases in the world (SESAR, 2016) and in Brazil (DECEA, 2015), it evolves at a fast pace, trailing the evolution of its technology. This has made the operation of many classes of RPAs today in many aspects similar to the operation of manned aircraft, having a direct effect shaping its industrial products.

For instance, in Brazil, operating a RPA class 3 (under 25 kg) over 400 ft in compliance with the countries current legislation, implies that the operator must be a ANAC certified pilot, the aircraft must be certified by ANAC and the mission must be authorized by the Brazilian airspace control department, DECEA (DECEA, 2015), (ANAC,

2017). For civilian purposes the airspace designated for drone applications is segregated in respect to the airspace for manned aircraft, meaning that both types of aircraft can not share the same airspace at the same time. This is a measure of security, since it is not yet possible to guarantee safety in both types of operations. A great immediate consequence is the need for each RPA operator to require a segregated airspace (NOTAM) to the DECEA (DECEA, 2015). Another general rule is that civil RPAs have no permission to fly less than 30m near any person who is not in accordance with the operation, which restricts even more the possibilities of flying over urban areas under the current legislation (ANAC, 2017).

Therefore, to expand the possibilities of RPA flights, enabling new types missions and applications, it is imperative that these systems evolve in order to become more reliable. In this sense, many efforts are being conducted to achieve the desired safety requirements, while respecting the low cost characteristics of this type of system. One particular domain is aircraft state estimation using low cost sensors, which is the theme of this work.

1.1.2 State estimation for RPA

Like many other controlled airspace vehicles, RPAs must accurately measure their physical states to correctly apply their commands for controlled flight. In this sense, the interested states can be represented by a vector $\mathbf{x}(t)$, given as a function of time, and governed by the equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1.1}$$

where $\mathbf{u}(t)$ is a known control input. In many cases the state vector is not directly observable, which means information about the system state must be inferred from another vector $\mathbf{y}(t)$ (output vector). The equation that establishes this relationship can be represented as:

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1.2}$$

Even though these equations describe in a great degree the system dynamics, they still represent a deterministic system (FARRELL, 2008), which is not the case of most of the systems found in navigation problems in the airspace field. These last systems are better described considering a stochastic model for the systems (FARRELL, 2008), in which the uncertainty about the models arise specially due to noise and imperfections of the used sensors (FARRELL, 2008). For example, the output of a sensor can be described by:

$$\mathbf{y}(t) = (1 + k)\mathbf{y}_t(t) + \mathbf{b}(t) + \mathbf{v}(t) \tag{1.3}$$

where $\mathbf{y}$ is the output measurement, k is a scale factor relating the true measurement $\mathbf{y}_t$ to $\mathbf{y}$, $\mathbf{v}$ is a fast varying error called noise and, $\mathbf{b}(t)$ is slow varying error called called bias. In practical applications, the stochastic characteristics of these two last types of error

are usually given by the sensor manufacturer (BOARD, 1998), along with information regarding the scale factor. These characteristics will be discussed further in the text.

In most RPA, the states used in navigation correspond to position, velocity, acceleration and orientation. For measuring them, many low-cost RPAs are equipped with three navigational sensors: a GPS, giving position and velocity; an accelerometer, measuring the linear acceleration in three orthogonal directions aligned with the aircraft's axes; a magnetometer, measuring the surrounding magnetic field in three orthogonal axes aligned with the aircraft's axes and; a gyroscope, measuring the three angular body rates. The gyroscope, the accelerometer and the magnetometer are often found to be in the same component, called Inertial Measurement Unit (IMU). Today, meany cheap IMUs can be found in the market, enabling the spread of its usage in low-cost RPAs.

Each of the measurements given by these sensors may be used in different ways to calculate the vehicle's physical states. However, in all cases, every calculated value will always be an estimation, since all of these sensors are not perfect, being corrupted by errors such as noise and bias.

Up until today, many well succeeded techniques for estimating states from corrupted sensors have been proposed (CRASSIDIS; MARKLEY; CHENG, 2007), such as nonlinear complementary filters (EURE et al., 2013) and various types of Kalman Filters ((CAVALLO et al., 2014), (BROWN; HWANG et al., 1992)). This work is dedicated to studying Kalman filter techniques since it is a well established method in the airspace industry (MARKLEY, 2003) and can be easily expanded to fit the use of multiple sensors, as it uses a state space representation methods (BROWN; HWANG et al., 1992).

1.2 Objectives

This work studies and implements a set of estimators, by the use of Extended Kalman Filters, to estimate a low-cost RPAs main navigational states: roll, pitch, yaw, position and velocity. The estimators must then use common RPA low-cost sensors such as magnetometer, an accelerometer, a gyroscope and a GNSS.

1.3 Literature review

1.3.1 Discrete Kalman Filters

The Kalman filter estimation techniques were first published in Dr. Kalman's seminal paper "A New Approach to Linear Filtering and Prediction Problems" (KALMAN, 1960) in 1960. In this paper, the novel techniques combined linear filtering to modern state space control concepts, being an important contribution to the Wiener filtering field (MCGEE; SCHMIDT, 1985).

In general, standard discrete Kalman filters work assuming two models which must be linear: a system model, which models the state's dynamics (the dynamics of $\hat{\mathbf{x}}$) and; a measurement model, which models the sensor's measurements. In the discretized form, the two equations describing these can be expressed, respectively, by:

$$\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{M}_k\mathbf{u}_k + \mathbf{w}_k, \tag{1.4}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k, \tag{1.5}$$

where $\mathbf{u}_k$ is a known input, $\mathbf{F}_k$ is the **transition matrix** and $\mathbf{y}_k$ is the sensor's output. $\mathbf{w}_k$ and $\mathbf{v}_k$ are, respectively, the **process noise** and the **measurement noise**, which are both assumed white and Gaussian.

To introduce the idea of the algorithm it is useful to recall the definition of a state observer (OGATA; YANG, 2010). In its canonical form in discrete time, a state observer can be described by equations:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{F}_k\hat{\mathbf{x}}_k + \mathbf{M}_k\mathbf{u}_k + \mathbf{K}_k\delta\mathbf{y}_k = \mathbf{F}_k\hat{\mathbf{x}}_k + \mathbf{M}_k\mathbf{u}_k + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k) \tag{1.6}$$

$$\hat{\mathbf{y}}_k = \mathbf{H}_k\hat{\mathbf{x}}_k \tag{1.7}$$

where the gain $\mathbf{K}_k$ is the observer gain and the term $\delta\mathbf{y}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k$ is often referred to as the residual of the measurement (i.e. the difference between the measurement $\mathbf{y}_k$ and last estimated value $\hat{\mathbf{y}}_k$).

As seen in the above equations, the state observer corrects the propagation of the state estimation dynamics by making adjustments depending on the residual of the measurement. Therefore, one may realize that equation can be divided in two parts: a first which dealing with system dynamics, and a second one correcting the dynamics with a recent measurement.

Similarly, the Kalman filter algorithm (algorithm 1) can be divided in to two steps: the **predict step**, where the state estimation $\hat{\mathbf{x}}$ is propagated using its dynamic model (1.8) and an *a priori* estimate ($\hat{\mathbf{x}}^-$) is computed and; the **update step**, where a *a posteriori* estimate ($\hat{\mathbf{x}}^+$) is calculated by correcting $\hat{\mathbf{x}}^-$ through equation 1.9, as the correction term would do in a state observer.

$$\hat{\mathbf{x}}^-_{k+1} = \mathbf{F}_k\hat{\mathbf{x}}^+_k + \mathbf{M}_k\mathbf{u}_k \tag{1.8}$$

$$\hat{\mathbf{x}}^+_{k+1} = \hat{\mathbf{x}}^-_{k+1} + \mathbf{K}_k\delta\mathbf{y}_k = \hat{\mathbf{x}}^-_{k+1} + \mathbf{K_k}(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}^-_{k+1}) \tag{1.9}$$

One way to understand the Kalman filter algorithm is by analyzing the statistical properties of the state vector, such as mean and covariance. This will be done in the

following, however the reader is encouraged to see a more complete discussion in (BROWN; HWANG et al., 1992) and (FARRELL, 2008).

A first analysis is in regards to the state predict equation. Assuming $E[\mathbf{w}_k] = 0$ in equation 1.4, it is straight forward to see that, by taking the expectation over this equation, the mean of the state $E[\mathbf{x}_k]$ is in fact propagated the same way as $\hat{\mathbf{x}}_k$ in equation 1.8:

A second analysis sheds light on the update equation gain $\mathbf{K}_k$. Particularly in the Kalman filter, this gain $\mathbf{K}_k$ depends on the state covariance matrix (algorithm 1) defined as the positive definite matrix:

$$\mathbf{P}_k = E[(\mathbf{x}_k - \mu_k)(\mathbf{x}_k - \mu_k)^T], \quad \mu_k = E[\mathbf{x}_k] \tag{1.10}$$

This matrix holds the state's uncertainty information and should thus vary in time along with the state estimation $\hat{\mathbf{x}}_k$. The propagation of this matrix in the predict phase (algorithm 1) can be deduced by applying the definition of the covariance as in the following equation

$$\mathbf{P}^-_{k+1} = E[(\mathbf{F}_k(\mathbf{x}_k - \mu_k) + \mathbf{w}_k)(\mathbf{F}_k(\mathbf{x}_k - \mu_k) + \mathbf{w}_k)^T] \tag{1.11}$$

$$\mathbf{P}^-_{k+1} = \mathbf{F}_k\mathbf{P}^+_k\mathbf{F}^T_k + \mathbf{Q}^T_k \tag{1.12}$$

where the positive definite matrix $\mathbf{Q}_k = E[\mathbf{w}_k\mathbf{w}^T_k]$ is the process noise covariance matrix.

In the update phase, $\mathbf{P}_k$ gets corrected along with the estimated state. Before discussing this correction further note that by subtracting equation 1.4 by equation 1.8 one obtains the linear relationship for the *a priori* error $\delta\mathbf{x}^-_{k+1}$ as:

$$\delta\mathbf{x}^-_{k+1} = \mathbf{x}_{k+1} - \hat{\mathbf{x}}^-_{k+1} = \mathbf{F}_k(\mathbf{x}_k - \hat{\mathbf{x}}^+_k) + \mathbf{w}_k = \mathbf{F}_k\delta\mathbf{x}^+_k + \mathbf{w}_k \tag{1.13}$$

Proceeding in an analogous manner with equations 1.5 and 1.7, it can be seen that the measurement residue has as well a linear relationship to the *a priori* error:

$$\delta\mathbf{y}^-_k = \mathbf{H}_k\delta\mathbf{x}^-_k + \mathbf{v}_k \tag{1.14}$$

One also notes that by subtracting equation 1.9 by $\mathbf{x}_k$ on both sides and using 1.14, one obtains a linear relationship between the *a posteriori* error and the *a priori* error:

$$\delta\mathbf{x}^+_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\delta\mathbf{x}^-_k - \mathbf{K}_k\mathbf{v}_k \tag{1.15}$$

By taking the covariance of the residue in equation 1.14, one obtains:

$$\mathbf{S}_k = E[\delta\mathbf{y}_k\delta\mathbf{y}^T_k] = E[(\mathbf{y}_k - E[\mathbf{y}_k])(\mathbf{y}_k - E[\mathbf{y}_k])^T] = \mathbf{H}_k\mathbf{P}^-_k\mathbf{H}^T_k + \mathbf{R}_k \tag{1.16}$$

where $\mathbf{S}_k$ is called the measurement error covariance matrix, and the positive definite matrix $\mathbf{R}_k = E[\mathbf{v}_k\mathbf{v}^T_k]$ is the measurement covariance matrix.

By taking the covariance of the *a posteriori* error in equation 1.15, one deduces the covariance matrix for the *a posteriori* state:

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{1.17}$$

Now, it is necessary to establish the relationship for calculating $\mathbf{K}_k$. There are many ways to define this expression (FARRELL, 2008), but here this will be done by the solution of an optimization problem, given by the cost function:

$$J(\mathbf{K}_k) = Tr(\mathbf{P}_k^+) \tag{1.18}$$

where $\mathbf{P}_k^+$ is given by 1.17. Note that this cost function seeks to minimize the sum of the variance of each element of the state vector.

By differentiating equation 1.18 in respect to $\mathbf{K}_k$ and equaling the derivative to zero, one can derive the expression for $\mathbf{K}_k$ as (FARRELL, 2008):

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{S}_k^{-1} \tag{1.19}$$

As mentioned in (FARRELL, 2008), the inverse of $\mathbf{S}_k$ exists since $\mathbf{P}^-$ and $\mathbf{R}_k$ are positive definite matrices.

Finally, by combining equations 1.17 and 1.19, one can deduce a simpler relationship for the *a posteriori* state covariance (FARRELL, 2008):

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \tag{1.20}$$

Algorithm 1 Kalman Filter. **Input: y**. **Output: $\hat{\mathbf{x}}$**

procedure INITIALIZATION
    Set $\hat{\mathbf{x}}_0^- = E[\mathbf{x}_0]$ and $\mathbf{P}_0 = var(\mathbf{x}_0^-)$
while $t_k \neq t_{END}$ **do**
    procedure PREDICT
        Propagate state $\hat{\mathbf{x}}_{k+1}^- = \mathbf{F}_k\hat{\mathbf{x}}_k^+ + \mathbf{M}_k\mathbf{u}_k$
        Propagate state covariance $\mathbf{P}_{k+1}^- = \mathbf{F}_k\mathbf{P}_k^+\mathbf{F}_k^T + \mathbf{Q}_k^T$
    procedure UPDATE
        Calculate residual covariance $\mathbf{S}_k = \mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k+1}^-\mathbf{H}_k^T$
        Calculate Kalman gain $\mathbf{K}_k = \mathbf{P}_{k+1}^-\mathbf{H}_k^T\mathbf{S}_k^{-1}$
        Correct $\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_{k+1}^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k+1}^-)$
        Correct $\mathbf{P}_{k+1}^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k+1}^-$

Remark 1

As seen in the previous subsection, the Kalman filter equations are recursive, in the sense that the solutions of the problem depend on the solutions of smaller instances

of the same problem (GRAHAM et al., 1989). Thus, a quick analysis of the algorithm can show that much computer memory can be spared. This characteristic is one of the relevant aspects of this type of estimator for airspace systems. For example, in the early phases of the use of this technology, it suited well the navigation calculations in the Apollo program which had to meet the storage constraints of NASA's IBM 704 computer (MCGEE; SCHMIDT, 1985).

A clever way to implement the filter is by using functions that call themselves in their own code (BAPTISTE; MARANGET, 2006). However, in this project, the Kalman filter codes were implemented as an iterative algorithm, as shown in algorithm 1.

Remark 2

Several important properties of the estimated state calculated by the standard Kalman filter algorithm are explained in (FARRELL, 2008). These properties will not be shown here, but are relevant to understanding the statistical properties of the estimation. Important properties of the estimated state are:

- it is an unbiased estimate;
- it is optimal in the sense of minimum mean-squared error;
- it is optimal in the sense of maximum likelihood (MLE).

It is also important to keep in mind that these properties hold under the three previously mentioned assumptions: the models used are linear, the noises are white and Gaussian. For a more complete development of the mathematical background of the Kalman filter estimation please refer to (BROWN; HWANG et al., 1992).

1.3.1.1 Extended Kalman Filter (EKF) in aerospace applications

After publication of Dr. Kalman's paper, it was not long for engineers to find that the linearity assumption was not a hindrance in applications inserted in nonlinear contexts. This was the case of many problems in the airspace field context. During the Apollo space program, at the NASA Ames Research Center in the 1960s, it was realized that Kalman's linear filters could be combined to linear perturbation concepts to solve the nonlinear systems in navigation problems (MCGEE; SCHMIDT, 1985).

In the case of a nonlinear dynamical system $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k)$ ($\mathbf{f}$ nonlinear in its arguments) or a nonlinear measurement model $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, t_k)$ ($\mathbf{h}$ nonlinear in its arguments), the regular Kalman Filter is not adequate, leading to other variations such as the Linearized Kalman Filter and the Extended Kalman Filter (BROWN; HWANG et al., 1992). These filters deal with nonlinearity by approximating the error dynamics of

equations 1.13 to a first order linearization of $\mathbf{f}$. In this sense, the transition matrix becomes $\mathbf{F}_k = \frac{\mathbf{d}}{\mathbf{dx}}\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ and the measurement matrix $\mathbf{H}_k$ in 1.14 becomes $\mathbf{H}_k = \frac{\mathbf{d}}{\mathbf{dx}}\mathbf{h}(\mathbf{x}_k)$.

In the Linearized Kalman Filter, the nonlinear equations are linearized with respect to the state $\mathbf{x}_k$ of a pre-programmed known trajectory. In the Extended Kaman Filter (EKF), the linearization is calculated using the most recently estimated state $\hat{\mathbf{x}}_k$ (see algorithm 2). As stated in (BROWN; HWANG et al., 1992) and (JULIER; UHLMANN, 1997), this last type of linearization may be riskier if the models are not sufficiently close to reality or if the measurement errors grow, resulting in divergence. However, in many aerospace navigational problems, the linearity approximation is usually valid, as was in the Apollo case (MCGEE; SCHMIDT, 1985), one of the first applications of EKFs. According to (FARRELL, 2008), in navigation aided by GNSS this assumption usually holds making these methods suitable for RPA navigational purposes.

Algorithm 2 Extended Kalman Filter (EKF). Example where dynamic and measurement models are nonlinear. **Input: y**. **Output: $\hat{\mathbf{x}}$**

procedure INITIALIZATION
    Set $\hat{\mathbf{x}}_0^- = E[\mathbf{x}_0]$ and $\mathbf{P}_0 = var(\mathbf{x}_0^-)$
while $t_k \neq t_{END}$ **do**
    procedure PREDICT
        Propagate state $\hat{\mathbf{x}}_{k+1}^- = \mathbf{f}(\hat{\mathbf{x}}_k^-, \mathbf{u}_k, t)$
        Propagate state covariance $\mathbf{P}_{k+1}^- = \mathbf{F}_k\mathbf{P}_k^+\mathbf{F}_k^T + \mathbf{Q}_k,\ \mathbf{F}_k = \frac{\mathbf{d}}{\mathbf{dx}}\mathbf{f}(\hat{\mathbf{x}}_k^-, \mathbf{u}_k, t_k)$
    procedure UPDATE
        Calculate residual covariance $\mathbf{S}_k = \mathbf{R}_k + \mathbf{H}_k\mathbf{P}_{k+1}^-\mathbf{H}_k^T,\ \mathbf{H}_k = \frac{\mathbf{d}}{\mathbf{dx}}\mathbf{h}(\hat{\mathbf{x}}_k^-, t_k)$
        Calculate Kalman gain $\mathbf{K}_k = \mathbf{P}_{k+1}^-\mathbf{H}_k^T\mathbf{S}_k^{-1}$
        Correct $\hat{\mathbf{x}}_{k+1}^+ = \hat{\mathbf{x}}_{k+1}^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k+1}^-)$
        Correct $\mathbf{P}_{k+1}^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k+1}^-$
    procedure OUTPUT($\hat{\mathbf{x}}_k$)

1.3.2 The Error State Extended Kalman Filter (ESKF)

In many aerospace applications ((BROWN; HWANG et al., 1992), (FARRELL, 2008), (INOUE et al., 2017), (SOLA, 2017), (TRAWNY; ROUMELIOTIS, 2005), (MARKLEY, 2003)) it is possible to find the use of a specific formulation of regular Kalman Filter and EKFs: the error state Kalman Filter (ESKF). In the error-state formulation three state values are considered: true ($\mathbf{x}$), nominal ($\bar{\mathbf{x}}$) and error-state ($\delta\mathbf{x}$) (SOLA, 2017). The nominal state contains the values of the estimated state without noise or model imperfections and therefore must be corrected by the error-state in order to not diverge from the true state (SOLA, 2017).

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \delta\mathbf{x} \tag{1.21}$$

In algorithm 3 both nominal state and error-state are propagated by equations 1.22 and 1.23, respectively. However the correction could be understood to be done only in the error state, differing from regular Kalman filters.

$$\bar{\mathbf{x}}_k = \mathbf{f}(\bar{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \tag{1.22}$$

$$\delta\mathbf{x}_k^- = \mathbf{F}_{k-1}\delta\mathbf{x}_{k-1}^+, \quad \mathbf{F}_k = \frac{\mathbf{d}}{\mathbf{dx}}\mathbf{f}(\bar{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \tag{1.23}$$

At k=0, let the estimated state value coincide with the nominal state such that $\bar{\mathbf{x}}_0 = E[\mathbf{x}(0)]$. There is then no need for correction, and thus $\delta\mathbf{x}_0^+$ should be zero and the nominal trajectory is updated by summing zero to it (by equation 1.26). In the next step $\delta\mathbf{x}_1^-$ is calculated from equation 1.23, however, since $\delta\mathbf{x}_0^+ = 0$, it will also be $\delta\mathbf{x}_1^- = 0$.

$$\delta\mathbf{x}_k^+ = \delta\mathbf{x}_k^- + \mathbf{K}_k(\mathbf{y}_k - \bar{\mathbf{y}}_k) \tag{1.24}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{R}_k + \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T)^{-1}, \quad \mathbf{H}_k = \frac{\mathbf{d}}{\mathbf{dx}}\mathbf{h}(\bar{\mathbf{x}}_k) \tag{1.25}$$

Now in the next correct phase of k=1, a new measurement has arrived, and thus $\delta\mathbf{x}_1^-$ will be corrected by equation 1.24, where $\mathbf{K}_k$ is given by equation 1.25. Once the correction is realized in the error state, the nominal state is updated (equation 1.26), the expectation of the state goes back to being the nominal state and the expectation of the error state becomes zero. In practice, this means that the error state is set to zero right after the nominal state is updated (equation 1.27).

$$\bar{\mathbf{x}}_{k+1} \leftarrow \bar{\mathbf{x}}_k + \delta\mathbf{x}_k^+ \tag{1.26}$$

$$0 \leftarrow \delta\mathbf{x}_k^+ \quad \textit{or equivalently} \quad 0 \leftarrow \delta\mathbf{x}_{k+1}^-. \tag{1.27}$$

As seen, in each iteration's predict step the $\delta\mathbf{x}_{k+1}$ arrives with a zero value, which at a first look seems that it is not propagated. However, this is not true since the propagation does happen through equation 1.23, but it is trivial (FARRELL, 2008).

Another detail in this algorithm is how the nominal update occurs. Note that here the error state updates the nominal state by sums (equation 1.26), but the plus sine is meant to mean a composition (SOLA, 2017). For example, in case of quaternions, the composition can be chosen to be a quaternion sum (like the additive EKF in (MARKLEY, 2003)) or a quaternion product (like the multiplicative EKF in (MARKLEY, 2003)). Reference (MARKLEY, 2003) discusses further the difference between both.

Algorithm 3 ESKF filter. Error state $\delta\mathbf{x}$. **Input**: $\mathbf{y}$. **Output**: $\hat{\mathbf{x}}$

procedure INITIALIZATION
    Set $\bar{\mathbf{x}}_0$, $\mathbf{P}_0$ and $0 \leftarrow \delta\mathbf{x}_0$
while $t_k \neq t_{END}$ **do**
    procedure PREDICT
        Propagate $\bar{\mathbf{x}}_{k+1} = \mathbf{f}(\bar{\mathbf{x}}_k, \mathbf{u}_k)$
        Propagate $\delta\mathbf{x}^-_{k+1} = \mathbf{F}_k \delta\mathbf{x}^+_k$.
        Propagate $\mathbf{P}^-_{k+1} = \mathbf{F}_k \mathbf{P}^+_k \mathbf{F}^T_k + \mathbf{Q}_k$.
    procedure UPDATE
        Calculate residual covariance $\mathbf{S}_k = \mathbf{R}_k + \mathbf{H}_k \mathbf{P}^-_{k+1} \mathbf{H}^T_k$, $\mathbf{H}_k = \frac{\mathbf{d}}{\mathbf{dx}} \mathbf{h}(\hat{\mathbf{x}}^-_k, t_k)$
        Calculate Kalman gain $\mathbf{K}_k = \mathbf{P}^-_{k+1} \mathbf{H}^T_k \mathbf{S}^{-1}_k$
        Calculate residue $\delta\mathbf{y}_k = \mathbf{y}_k - \bar{\mathbf{y}}_k$
        Correct error state $\delta\mathbf{x}^+_{k+1} = \delta\mathbf{x}^-_{k+1} + \mathbf{K}_k \delta\mathbf{y}_k$.
        Correct matrix $\mathbf{P}^+_{k+1} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}^-_k$
        Update $\bar{\mathbf{x}}_{k+1} \leftarrow \bar{\mathbf{x}}_{k+1} + \delta\mathbf{x}^+_{k+1}$
        Reset error state $0 \leftarrow \delta\mathbf{x}^+_{k+1}$
    procedure OUTPUT($\bar{\mathbf{x}}_k$)

It is interesting to note that in the ESKF the assumed dynamic model is the dynamic model of the errors, which are in fact linear. According to (SOLA, 2017) some advantages of the ESKF applied to IMU-driven systems in particular is that the error-state second order products are negligible and it operates close to the origin, avoiding parameter singularities. Another aspect is that the the orientation error-state is minimal ((SOLA, 2017), (MARKLEY, 2003)) reducing a four component state to a three component state.

1.3.3 Sensor models used in Kalman filters

An appropriate implementation of the Kalman filter update phase depends on the development of a measurement model for each one of the sensors used. One of the great difficulties with low-cost sensors is the lower accuracy of their measurements, which can impose challenges when modeling. In many cases low-cost sensors may be difficult to calibrate than more sophisticated sensors. For example, it can be much more difficult to identify noise sources (XING; GEBRE-EGZIABHER, 2008).

To cope with this issue, a common approach is to create an augmented system ((BROWN; HWANG et al., 1992), (FARRELL, 2008), (SOLA, 2017), (TRAWNY; ROUMELIOTIS, 2005)), which means that the sensor errors are also taken as an estimated state in the Kalman filter. The most common estimated error is the bias. This approach used in literature is described in the following subsection.

1.3.3.1 Inertial sensors

According to (XING; GEBRE-EGZIABHER, 2008), an inertial sensor output y_m can be modeled by

$$y_m = (1 + k)y + b(t) \tag{1.28}$$

where y is the true value, k is a scale factor which can be modeled by a random constant and $b(t)$ is a time-varying bias. $b(t)$ is also decomposed in:

$$b(t) = b_0 + b_R(t), \quad b_R(t) = b_w(t) + \sum_{i=1}^{N} b_i(t) \tag{1.29}$$

where b_0 is a random constant ("turn-on to turn-off bias), $b_w(t)$ is a wide-band process and $b_i(t)$ are correlated time domain error processes. As stated by (XING; GEBRE-EGZIABHER, 2008), one may choose to keep only one term of correlated noise ($b_1(t)$), as a trade-off between usability and accuracy, and model it as a first order Gauss-Markov process (XING; GEBRE-EGZIABHER, 2008). One way to understand a Gauss-Markov process is to recognize it as the output of a linear system with a white noise input (BROWN; HWANG et al., 1992). In this sense, the power spectral density (PSD) of the Gauss-Markov process can be obtained by (BROWN; HWANG et al., 1992)

$$S_{GMkv}(j\omega) = |G(j\omega)|^2 S_w \tag{1.30}$$

$|G(j\omega)|$ represents the transfer function of a first order low pass filter and S_w is the PSD of the white noise. A time domain description of this process can also be given by the first order stochastic differential equation:

$$\dot{b}_1(t) = -\frac{1}{\tau_{b_1}} b_1(t) + w_{b_1} \tag{1.31}$$

where τ_{b_1} is the correlation time of the process and w_{b_1} is the driving white process noise.

Thus, from equation 1.31 it can be seen that in order to complete the sensor measurement model, the bias is considered as a new state. Moreover, note that the time-varying correlated component of the bias becomes then a state with linear dynamics and its own corrupting white noise, suiting the Kalman filter assumptions.

Before being able to model b_1 as a state of the Kalman Filter, this approach demands the determination of τ_{b_1} and the covariance of w_{b_1}. One method is discussed in (FARRELL, 2008) and uses PSD to describe the noises sources. Another method described in (XING; GEBRE-EGZIABHER, 2008) uses time domain equivalent of the PSD called Allan Variance method (ALLAN, 1966). In (XING; GEBRE-EGZIABHER, 2008), the method is applied in order to determine bounds for the covariances of the different components of noise present in the sensor measurements.

1.3.3.2 Types of Kalman filters in respect to models

In airspace applications one can also distinguish two categories of Kalman filters with respect to the system dynamic model:

- **Model-free Kalman Filter**: which do not require a system model as they work essentially as a sensor fusion algorithm. Examples of this kind of filter can be found in (EURE et al., 2013), (FARRELL, 2008), (BROWN; HWANG et al., 1992), (SOLA, 2017), (INOUE, 2011)). In navigation problems, the predict phase of these filters integrate gyroscope measurements, accelerometer measurements and velocity measurements to obtain *a priori* estimates of orientation, velocity and position, respectively. And in the update phase, these estimations are compared to sensor measurements from which orientation, velocity and position can be observed, respectively. An important advantage of this type of filter is that it can be applied to different systems with barely any changes - exceptions would be certain parameters such as sensor positions that could change from vehicle to vehicle.

- **Model-based Kalman Filter**: which require a description of the aircraft's dynamics through an accurate and representative model. An example of a model-based Kalman filter in aerospace applications is (BORUP; FOSSEN; JOHANSEN, 2016). If one were to conceive a model-based filter for the same navigation problem discussed in the last item, the aircraft dynamics would be modeled, which would include its forces and moments. By modeling forces, *a priori* estimates of accelerations could be calculated, which can be compared with accelerometer measurements in the update phase. By modeling torques, angular acceleration estimates could be calculated, which can be integrated to obtain a a priori estimate of body rates. The body rates, in their turn, can be compared in the update phase to gyroscope measurements. The difficulty of this type of Kalman filter lies in developing an accurate model of the system. In the case of fixed wing drones, like the ones here used in tests, this means accurately modeling aerodynamic forces and moments at given flight conditions.

Even though the second set of filters would be more complete, the first set of filters have a wide variety of literature asserting its efficiency (see (BROWN; HWANG et al., 1992),(SOLA, 2017), and (FARRELL, 2008), (EURE et al., 2013)) and is frequently chosen by low cost RPA developers, as seen in (ARDUPILOT, 2019b), (EURE et al., 2013).

1.4 Scientific methodology

The methods used to arrive at the objectives are enumerated in following items:

1. Model-free Kalman filter techniques were chosen since they do not require an accurate dynamic model of the aircraft. The cost of developing such a model can increase the cost and the complexity of a RPA project.

2. The technique used in the orientation estimator is the Error State Kalman Filter, since the error state respects the Kalman assumption of linearity.

3. In the estimator for position and velocity a regular Extended Kalman filter is implemented, since the adopted position and velocity equations are nonlinear.

4. The Allan Variance method was used as in (XING; GEBRE-EGZIABHER, 2008) to describe each sensor's noise source and more appropriately tune the filters. In order to accelerate the tuning, a MATLAB based tool described in (INOUE, 2011) was used.

5. All implementations were done in MATLAB language, since algorithms could be easily implemented and altered. MATLAB is also an ideal environment for analyzing the results from numerical method calculations and plotting data.

6. MATLAB tools were used for implementing simulated data by generating random signals, since Kalman Filters assume the corrupting noise is white and Gaussian.

7. Real flight data was used in the algorithm tests by replaying the acquired sensor data.

2 DEVELOPMENT

2.1 Reference Frames

In this section several reference frames used in the following text will be presented: the inertial frame, the body frame, the ECEF (*Earth-Centered, Earth fixed*) frame, the geographic frame and the navigation frame. The inertial frame is the frame where the laws of Newton apply, and therefore is not accelerating or rotating. The inertial sensors will produce measurements in respect to this frame.

As defined in (FARRELL, 2008), the body frame is rigidly fixed to the vehicle of interest in its center of gravity. The x-axis is defined pointing towards the front of the vehicle, the z-axis is defined pointing downwards in the vehicle, while the y-axis is defined in matter to complete a right-hand orthogonal frame.

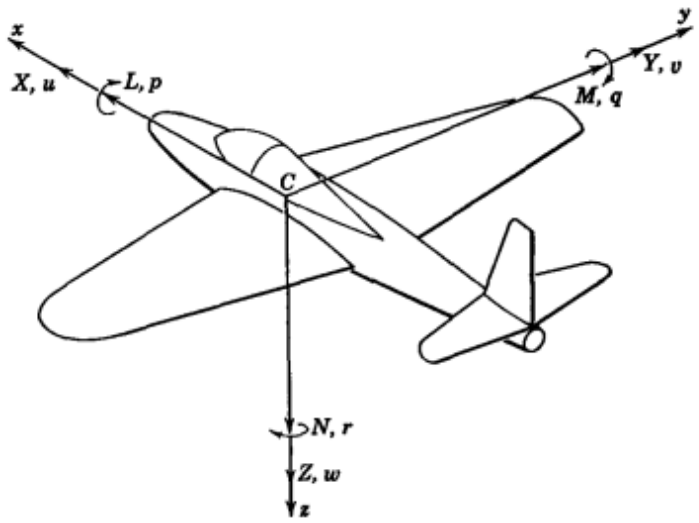

Figure 2: Body frame (ETKIN; REID, 1959)

The ECEF frame has its origin in Earth's center of mass and is fixed relative to Earth. As shown in figure 3, the rectangular coordinate frame of the ECEF has its x-axis traversing the equator at zero latitude and zero longitude, and z-axis points towards the North pole, coinciding with Earth's spin axis. It is also possible to represent ECEF coordinates with the triplet $[\phi, \lambda, h]^t$, the geodetic coordinates. The geodetic surface is defined to be everywhere normal to the gravity vector (FARRELL, 2008).

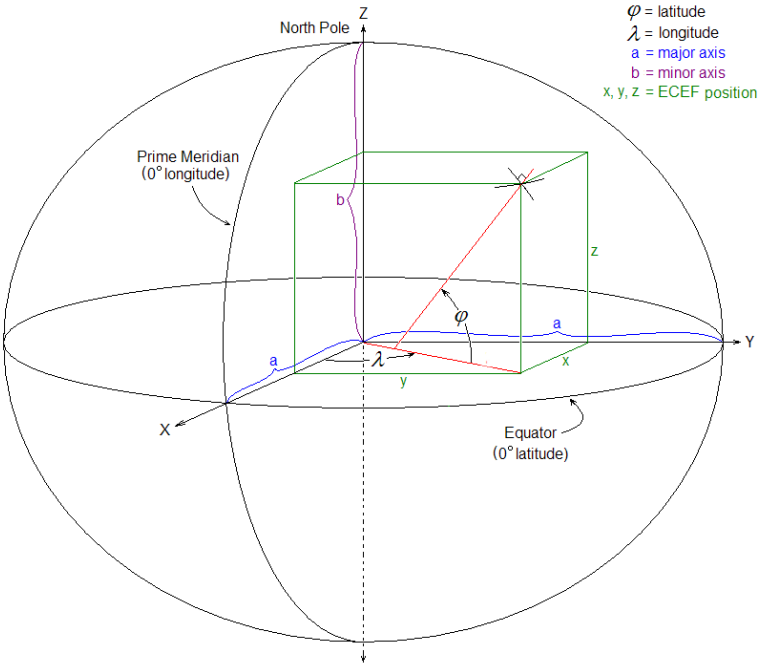

Figure 3: ECEF *Earth-Centered, Earth fixed* (ECEF) frame (WIKIPEDIA, 2019)

The geographic frame is the local frame which origin is projected on the reference Earth ellipsoid ((FARRELL, 2008))and moves moves with the vehicle. The z-axis points downward to the ellipsoid, while the x-axis points north and the y-axis points east, completing an orthogonal right-hand axis (North East Down, as seen in figure 4).

The vehicle's latitude ϕ and the longitude λ define the position of the geographic frame in the reference ellipsoid. Note that the relative position of the body frame in the geographic frame is defined as $\mathbf{x}^g = [0, 0, -h]^T$, where h is vehicle's altitude in respect to the ground surface. The components of the vehicle's relative velocity to the ground (or to the ECEF frame) is represented in the geographic frame as $\mathbf{v}_e^g = \mathbf{v}_{NED} = [v_n, v_e, v_d]^T$.

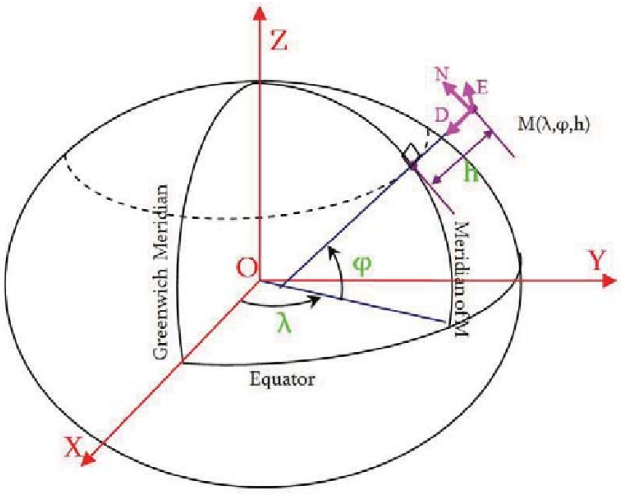

Figure 4: Definition of the navigation frame in the geographic frame. (LOTFI, 2015)

In this work, the navigational frame (or the global frame) will be defined as the geographic frame. This frame, along with body (or local) frame will be the most frequently used frames in this work.

2.2 Equations of motion

2.2.1 Orientation

The transformation between a vector in the navigation frame $\mathbf{x}^n$ and a vector in the body frame $\mathbf{x}^b$ is given by

$$\mathbf{x}^n = \mathbf{R}_b^n \mathbf{x}^b. \tag{2.1}$$

$\mathbf{R}_b^n$ is a rotation matrix which depends on the vehicle's Euler angles $[\phi, \theta, \psi]$ (roll, pitch and yaw) of the body frame in respect to the navigation frame:

$$\mathbf{R}_b^n = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} = (\mathbf{R}_n^b)^{-1} = (\mathbf{R}_n^b)^T \tag{2.2}$$

where $cx = \cos(x)$ and $sx = \sin(x)$.

Among the three standard choices of representing orientation in space (Euler angles, Rotation matrices and unit quaternions), the unit quaternion representation has been chosen for this work. The choice is due to its advantages (see (FARRELL, 2008)) such as: the lack of trigonometric functions in integration routines, the lack of singularities, when compared to Euler representation and; the smaller amount of parameters, when compared to direction cosine matrices.

Between the 12 different combinations of quaternions (SOLA, 2017), this work uses the a right-handed-rule product quaternion with a local-to-global representation (or the Hamilton representation), that is, the quaternion $\mathbf{q}_l^g = [q_0 \ \vec{q}^{\,T}]^T$ represents the rotation from a local frame (in this case, the body frame), to a global frame (in this case, the navigation frame). With this definition one may write a vector in the navigational frame $\mathbf{x}^b$ as:

$$\mathbf{x}^n = \mathbf{q}_b^n \otimes \mathbf{x}^b \otimes \mathbf{q}_b^{n*} = \mathbf{R}_b^n(\mathbf{q}_b^n)\mathbf{x}^n \tag{2.3}$$

where $\mathbf{q}_b^{n*}$ is the conjugate of $\mathbf{q}_b^n$ and $\mathbf{R}_b^n(\mathbf{q}_b^n)$ is the rotation matrix that produces the same rotation effect as $\mathbf{q}_b^n$ (SOLA, 2017). This matrix can be related to the quaternion by (see (SOLA, 2017)):

$$\mathbf{R}_b^n = \mathbf{I} + 2q_0[\vec{\mathbf{q}}\times] + 2[\vec{\mathbf{q}}\times]^2. \tag{2.4}$$

A definition of a perturbation quaternion $\Delta\mathbf{q}$ is also needed in order to carry on the orientation dynamics description. In this case two perturbations are defined: a local one, represented in the body frame $\Delta\mathbf{q}^b$ and; a global one $\Delta\mathbf{q}^n$, represented in the global frame.

As a consequence of the definition of the quarternion, the composition of the quaternion $\bar{\mathbf{q}}_b^n$ with its pertubation $\Delta\mathbf{q}$ must be done in the following orders:

$$\mathbf{q}_b^n = \bar{\mathbf{q}}_b^n \otimes \Delta\mathbf{q}^b \tag{2.5}$$

$$\mathbf{q}_b^n = \Delta\mathbf{q}^n \otimes \bar{\mathbf{q}}_b^n \tag{2.6}$$

An equivalent formulation of the above equations can be done with rotation matrices. This is presented in appendix B.3.

Now let $\omega_{nb}^b = [\omega_1, \omega_2, \omega_3]^T$ be the body rate of the body frame in respect to the navigation frame, represented in the body frame. With this definition, one can write the kinematic equations that govern the orientation quaternion (see (SOLA, 2017) and (FARRELL, 2008)):

$$\dot{\mathbf{q}}_b^n = \frac{1}{2}\mathbf{q}_b^n \otimes \mathbf{q}_{\omega_{nb}^b} = \frac{1}{2}\mathbf{\Omega}(\omega_{nb}^b)\mathbf{q}_b^n = \frac{1}{2}\begin{bmatrix} 0 & -\omega_{nb}^{b\,T} \\ \omega_{nb}^b & -[\omega_{nb}^b\times] \end{bmatrix}\mathbf{q}_b^n \tag{2.7}$$

where $[\omega_{nb}^b\times]$ is the skew symmetric matrix of the vector ω_{nb}^b. The above relationship can also be defined for a global representation of the body rate $\omega_{nb}^n = \mathbf{R}_b^n\omega_{nb}^b$:

$$\dot{\mathbf{q}}_b^n = \frac{1}{2}\mathbf{q}_{\omega_{nb}^n} \otimes \mathbf{q}_b^n = \frac{1}{2}\mathbf{\Omega}(\omega_{nb}^n)\mathbf{q}_b^n = \frac{1}{2}\begin{bmatrix} 0 & -\omega_{nb}^{n\,T} \\ \omega_{nb}^n & [\omega_{nb}^n\times] \end{bmatrix}\mathbf{q}_b^n \tag{2.8}$$

In order to simplify the notation in the following text, the quatenion $\mathbf{q}_b^n$ will be referred to as $\mathbf{q}$.

2.2.2 Position

As shown in (FARRELL, 2008) the equations of motion governing the vehicles position in the ECEF frame can be related to $\mathbf{v}^n$ by:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{v_n}{R_M+h} \\ \frac{v_e}{cos\phi(R_M+h)} \\ -v_d \end{bmatrix} \tag{2.9}$$

where the Earth meridian radius R_M and the Earth earth normal radius R_N are expressed as a function of Earth's elliptical eccentricity ($e = 0.0818$) and Earth equatorial radius ($a = 6378137.0$):

$$R_M = \frac{a(1-e^2)}{(1-e^2sin^2\phi)^{1.5}}, \quad R_N = \frac{a}{(1-e^2sin^2\phi)^{0.5}}$$

The same reference shows that the equation of velocity $\mathbf{v}^n$ is given by:

$$\begin{bmatrix} \dot{v}_n \\ \dot{v}_e \\ \dot{v}_d \end{bmatrix} = \begin{bmatrix} f_n^n \\ f_e^n \\ f_d^n \end{bmatrix} + \mathbf{g^n} - \begin{bmatrix} 0 & -\omega_d & \omega_e \\ -\omega_d & 0 & -\omega_n \\ -\omega_e & \omega_n & 0 \end{bmatrix} \begin{bmatrix} v_n \\ v_e \\ v_d \end{bmatrix} \tag{2.10}$$

In the above expression, $\mathbf{g}^n = [0\ 0\ g_e]^T$ with $g_e > 0$, and f^n is the specific force, which is the inertial force per unit mass, as defined in (FARRELL, 2008). The vector $\omega_{ig}^g = [\omega_n, \omega_e, \omega_d]^T$ is the rotation of the inertial frame to the geographic frame, and is given by:

$$\begin{bmatrix} \omega_n \\ \omega_e \\ \omega_d \end{bmatrix} = \begin{bmatrix} (\dot{\lambda} + 2\omega_{ie})cos\phi \\ -\dot{\phi} \\ -(\dot{\lambda} + 2\omega_{ie})sin\phi \end{bmatrix} \tag{2.11}$$

where $\omega_{ie} = 7.292115 \times 10^{-5} rad/s$, is the rotation of the inertial frame to the ECEF frame, and represents Earth's spin rate.

2.3 Measurement Models

2.3.1 Inertial Sensors

According to (FARRELL, 2008), the gyroscope output $\mathbf{u}_g$ can be expressed by the following equation:

$$\mathbf{u}_g = \omega_{ib}^b + \mathbf{x}_g + \mathbf{v}_g. \tag{2.12}$$

ω_{ib}^b is the inertial-to-body rate measurement in the body frame, $\mathbf{v}_g$ is the rate measurement's white Gaussian noise and the rate bias $\mathbf{x}_g$ is modeled by a scalar Gauss-Markov process as defined in the beggining of section 1.3.3.

$$\dot{\mathbf{x}}_g = \mathbf{F}_g\mathbf{x}_g + \mathbf{v}_{b_g}, \quad \mathbf{F}_g = -diag(\tau_{g_x}^{-1}, \tau_{g_y}^{-1}, \tau_{g_z}^{-1}) \tag{2.13}$$

Analogously, the accelerometer measurement model can be described by the equation (see reference (FARRELL, 2008)):

$$\mathbf{y}_a = \mathbf{a}_{ib}^b - \mathbf{g}^b + \mathbf{x}_a + \mathbf{v}_a' \tag{2.14}$$

where $\mathbf{a}_{ib}^b$ is the inertial-to-body acceleration measurement in the body frame, $\mathbf{v}_a'$ is the accelerometer measurement white Gaussian noise and the acceleration bias $\mathbf{x}_a$ is modeled as well by a scalar Gauss-Markov process given in equation 2.15. Note from equation 2.14 that at zero acceleration ($\mathbf{a}_{ib}^b = 0$) and assuming zero bias, the value measured by the accelerometer is $\mathbf{y}_a = -\mathbf{g}^b$, and not zero.

$$\dot{\mathbf{x}}_a = \mathbf{F}_a\mathbf{x}_a + \mathbf{v}_{b_a}, \quad \mathbf{F}_a = -diag(\tau_{a_x}^{-1}, \tau_{a_y}^{-1}, \tau_{a_z}^{-1}) \tag{2.15}$$

For the magnetometer measurement output, the following equation (found in (FARRELL, 2008)) can be writen:

$$\mathbf{y}_m = \mathbf{m}_e^b + \mathbf{x}_m^b + \mathbf{v}_m. \tag{2.16}$$

$\mathbf{v}_m$ is a Gaussian white noise, $\mathbf{m}_e^b = \mathbf{R}_n^b\mathbf{m}_e^n = \mathbf{R}_n^b[m_e\ 0\ 0]^T$ is the Earth's magnetic local field represented in the body frame, and $\mathbf{x}_m^b$ is the magnetometer bias due to the magnetic field produced by the own vehicle, also known as the Hard Iron bias affect (GUO et al., 2008). Differently from the bias of the other two sensors, $\mathbf{x}_m^b$ can be modeled as a random constant (GUO et al., 2008). Usually the magnetometer is calibrated for $\mathbf{x}_m^b$ after manufacturing. It is known to vary only in the long term usage of the RPA or after it has gone through maintenance, when a new calibration is often needed (ARDUPILOT, 2019a).

2.3.2 GNSS sensor model

In this work, the GNSS sensor measurements (position and velocity in the navigation frame) were assumed to be corrupted only by a zero mean, uncorrelated, Gaussian noise,

as in (INOUE, 2011):

$$\mathbf{p}_{GNSS} = \mathbf{p}^e + \mathbf{v}_{p_w} \tag{2.17}$$

$$\mathbf{v}_{GNSS} = \mathbf{v}^n + \mathbf{v}_{v_w} \tag{2.18}$$

with $\mathbf{v}_{p_w} \in R^{3\times1}$ and $\mathbf{v}_{v_w} \in R^{3\times1}$. The covariance matrices we estimated using the same method of the other sensors here described. However, this assumption can not be considered accurate, since the nature of the GNSS measurement model is quite different from the previous sensors. GNSS receivers often have their own filters which even may be Kalman filters. This implies time correlation between measurements which are not considered in this model. A more rigorous solution for the GNSS sensor would be therefore to model the parameters it uses in its own position calculations, such as pseudo-range and carrier phase ((FARRELL, 2008)).

According to (FARRELL, 2008), the time-correlation issue can be mitigated if the GNSS measurements output the position solutions point-wise, meaning that measurements are only calculated at each time only considering the available satellites at that time. Since this was the case for GNSS signal of the used RPAs, the only pertubation described was the white noise, estimated by the Allan Variance method

2.3.3 Co-variance Matrix Estimation: Allan Variance method

Correctly tuning a Kalman filter is an essential step to obtain the expected performance. The approach taken here sought to estimate the Kalman filter parameters for each of the sensor's used in tests, instead of relying solely in nominal information from manufacturers.

The covariance matrix estimation was realized using the Allan Variance method since it can be used to study noise from any instrument's data time series and has a good trade-off between precision and ease in usage (HOU, 2004). The description of how to calculate the Allan Variance from the data time series is briefly presented in appendix A, while the following text discusses how it was used to arrive at the desired covariance matrices.

As described in (XING; GEBRE-EGZIABHER, 2008), the Allan Variance $\beta^2(\tau)$ can be understood as a time domain equivalent of the PSD, as it is a function of the averaging time of the signal τ instead of it being a function of its frequency (see original paper on the method (ALLAN, 1966)). From the Allan Variance log-log plot one can identify the different error sources by their respective Allan-Variance slope (Figure 5).

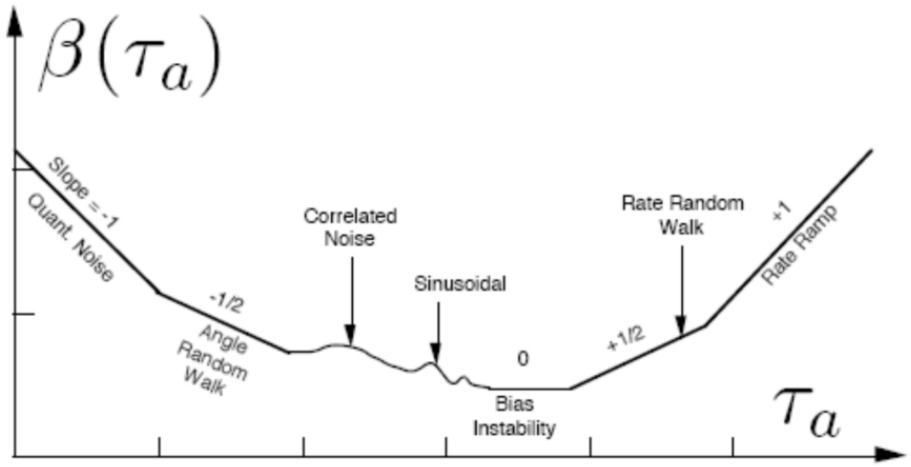

Figure 5: Allan Variance log-log plot for different types of error sources (BOARD, 1998).

Figure 6 shows a typical Allan deviation plot for data of an inertial sensor. From the figure, on can see that for low time correlations an angle random walk type slope prevails. As correlation times τ increase, the slope increases while the correlated noise starts to dominate. For very big correlation times, the Allan deviation plot becomes again a strait line with slope approximating that of a rate random walk.

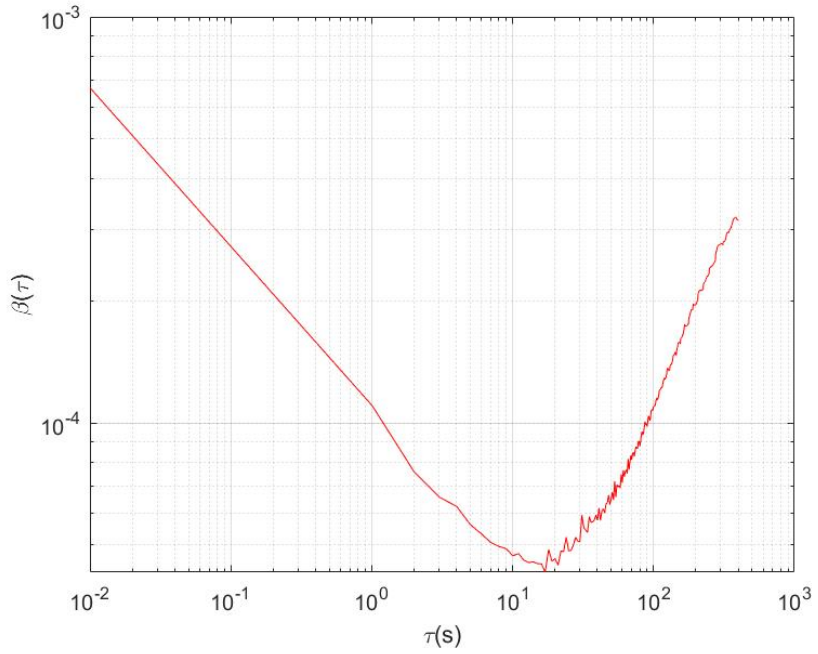

Figure 6: Allan deviation log-log plot for one of the axis of the gyroscope sensor

Once the sources are identified, the signal's variance can be obtained by determining the signal's PSD $S_{\mathbf{\Omega}}(f)$ which is uniquely related to the Allan variance values through the

relationship (BOARD, 1998):

$$\beta^2(\tau) = 4\int_0^\infty S_\Omega(f)\frac{\sin^4(\pi f\tau)}{(\pi f\tau)^2}df. \tag{2.19}$$

For a wide band noise, like the uncorrelated source $b_w(t)$ acting in equation 1.29, the PSD is $S_\Omega(f) = N^2$ and the Allen Variance is then

$$\beta^2(\tau) = \frac{N^2}{\tau}. \tag{2.20}$$

Thus the expected slope of the Allan variance is $-1/2$ and the estimated variance σ_{b_w} can be directly obtained by reading out the Allan variance in the plot for $\tau = T$ ((XING; GEBRE-EGZIABHER, 2008)).

For the scalar Gauss-Markov process, like the correlated process b_1 in equation 1.31, the noise description requires extra steps. Remember that in section 1.3.3.1 it was mentioned that the goal is to approximate the correlated noise terms by this single Gauss-Markov process b_1. Therefore, it would be more appropriate in this estimation to obtain a bound for the variance, rather then modeling it exactly ((XING; GEBRE-EGZIABHER, 2008)). Therefore, the sequence of this text will explain the method presented by (XING; GEBRE-EGZIABHER, 2008) to arrive at these bounds.

From equation 1.30 it can be deduced that the PSD of this process is ((XING; GEBRE-EGZIABHER, 2008)):

$$S_{b_1}(f) = \frac{Q_{w_{b_1}}\tau_{b_1}^2}{1+(2\pi\tau_{b_1}f)^2}, \quad Q_{w_{b_1}} = \frac{2\sigma_{b_1}^2}{\tau_{b_1}} \tag{2.21}$$

where τ_{b_1} is the correlation time and $Q_{w_{b_1}}$ is the PSD of the white noise ω_{b_1}, which both appeared in equation 1.31. It can be shown that the corresponding Allan Variance expression for 2.21 has two limiting cases ((GYROS, 2004)):

$$\beta^2(\tau) = \frac{Q_{w_{b_1}}}{3}\tau, \quad \tau << \tau_{b_1}, \tag{2.22}$$

$$\beta^2(\tau) = \frac{Q_{w_{b_1}}\tau_{b_1}^2}{\tau}, \quad \tau >> \tau_{b_1} \tag{2.23}$$

which correspond respectively to a random walk and to a rate random walk (white noise) with PSDs of $Q_{w_{b_1}}$ and $Q_{w_{b_1}}\tau_{b_1}^2$ (XING; GEBRE-EGZIABHER, 2008). Note it is possible to use two parameters to fit the Gauss-Marcov model to a desired correlated curve, being a more adaptive model then the random walk described previously (BROWN; HWANG et al., 1992), (XING; GEBRE-EGZIABHER, 2008).

From equation 2.22 and from the fact that the white noise Allan variance is a linear decreasing function of τ, it can be seen that for small values of τ the total Allan Variance is given by (XING; GEBRE-EGZIABHER, 2008):

$$\beta_{Total}^2(\tau) = \beta_w^2(\tau) + \beta_{b_1}^2(\tau). \tag{2.24}$$

Thus, an overbound $\bar{Q}_{w_{b_1}}$ for the parameter $Q_{w_{b_1}}$ can be obtained by isolating $\beta^2_{b_1}(\tau)$ in the previous equation and applying equation 2.22.

In this next step, a first estimate of $\sigma^2_{b_1}$ is obtained in order to derive τ_{b_1} from equation 2.21 and the recently computed value of $\bar{Q}_{w_{b_1}}$. Since the variance of the uncorrelated noise is often much greater than the variance of the correlated noise and thus could make its extraction from the total noise less precise, (XING; GEBRE-EGZIABHER, 2008) proposes to realize a 1Hz average of the $b_R(t)$ in order to mitigate its effect. After filtering, the uncorrelated component of the noise will remain uncorrelated with a variance of $\sigma^2_{b_w}T$, where T is sample time. Filtering the correlated component will not change much its value since the processes being captured now are those with correlation grater time. With this reasoning, a first estimation of $\sigma^2_{b_1}$ will be given by

$$\sigma^2_{b_1} = \sigma^2_{total} - \sigma^2_{b_w}T \tag{2.25}$$

With $\sigma^2_{b_1}$, τ_{b_1} is then calculated and a Allan variance for the b_1 process can be plotted. This plot can then be moved by altering these parameters until the correlated total Allan variance is bounded, as shown in figure 7.

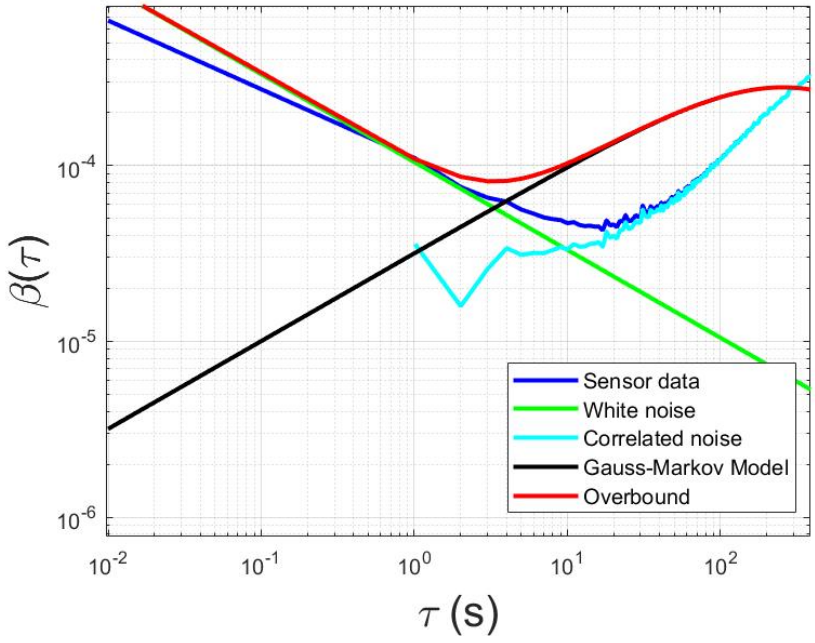

Figure 7: Allan deviation bounding for x-axis of gyroscope sensor RPA sensor.

The values found for $\sigma^2_{b_1}$, τ_{b_1} and $\sigma^2_{b_w}$ can then be finally integrated to the Kalman filter, in order to define matrices matrices Q and R.

2.4 Orientation Estimator

As mentioned in section 2.2.1, the objective of this estimator is to obtain an estimation for the orientation unit quaternion $\mathbf{q}$ which will be obtained by the use of an ESKF (section 1.3.2). There is a extensive amount of reference in this type of filter. In particular this work was based on (INOUE et al., 2017), (SOLA, 2017), (TRAWNY; ROUMELIOTIS, 2005) and (FARRELL, 2008). As it will be discussed in the following sections, the error quaternion can be approximated to $\mathbf{q} \approx [1 \ \delta\theta_G/2]^T$, where $\delta\theta_G$ is a global representation ((SOLA, 2017), (FARRELL, 2008)) of the vector part of the quaternion, and will be also referred to as the tilt angle. Thus, as will be shown further in this section, the corresponding part of the estimated vector $\hat{\mathbf{x}}$ that keeps track of the quaternion error will be a three component vector.

This estimator will dispose of four sensors: a gyro, which will provide body rates for quaternion propagation in the predict phase; an accelerometer that will be used to correct the predicted orientation using the measurement of the gravitational force in respect to the body frame; a magnetometer which will also correct orientation by giving the relative body frame measurement of the local magnetic field and; a GNSS, which will compensate the gyro's measurements for Earth's rotation.

In order to increase the precision of the estimation, the estimator will also keep track of the gyro bias $\mathbf{x}_g$ and the accelerometer bias $\mathbf{x}_a$. Moreover their respective errors, $\delta\mathbf{x}_g$ and $\delta\mathbf{x}_a$, will be included in the estimated state vector. Therefore, the ESKF will estimate the error vector $\delta\mathbf{x} = [\delta\theta_G, \delta\mathbf{x}_g, \delta\mathbf{x}_a]^T$.

2.4.1 Nominal state equations

The nominal state in the orientation vector will be propagated in each predict step along with the error state. In this filter, the nominal states are $\bar{\mathbf{x}} = [\bar{\mathbf{q}}, \bar{\mathbf{x}}_g, \bar{\mathbf{x}}_a]^T$.

2.4.1.1 Quaternion propagation

The first of these states, the quaternion, had its dynamics described previously by equation 2.8, which is repeated below:

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(t)\mathbf{q}. \tag{2.26}$$

It is now necessary to derive an expression to for the integration of this equations. This is done in detail in appendix B.2, following the derivation presented in (FARRELL, 2008).

In appendix B.2 one arrives at the closed form for the integration of the quaternion under the assumption of constant rates during the interval T:

$$\bar{\mathbf{q}}_k = \left(\cos(||\mathbf{w}||)\mathbf{I} + \frac{\sin(||\mathbf{w}||)}{||\mathbf{w}||}\mathbf{W}\right)\bar{\mathbf{q}}_{k-1} \tag{2.27}$$

As mentioned in (FARRELL, 2008), it can be shown that this algorithm is second order in T.

The above expression is what reference (TRAWNY; ROUMELIOTIS, 2005) defines as a zero order integration of the quaternion, since it considers constant rates during an interval of size T. The above equation may also be numerically unstable when the rates are near zero (TRAWNY; ROUMELIOTIS, 2005). This can be avoided by not propagating the quaternion when angular rates are under a threshold, however this will not be analyzed in depth in this work.

2.4.1.2 Bias propagation

Both nominal state biases are propagated through the linear equations presented in equations 2.15 and 2.13. As mentioned in section 1.3.2, the nominal state integration does not take into account noise, bias or other erros. Therefore a first order (Euler) integration of these equations will then be

$$\bar{\mathbf{x}}_{g(k)} = \mathbf{F}\bar{\mathbf{x}}_{g(k-1)}, \quad \mathbf{F} = \mathbf{I} - T\mathbf{F}_g \tag{2.28}$$

for the gyroscope nominal bias and

$$\bar{\mathbf{x}}_{a(k)} = \mathbf{F}\bar{\mathbf{x}}_{a(k-1)}, \quad \mathbf{F} = \mathbf{I} - T\mathbf{F}_a \tag{2.29}$$

for the accelerometer nominal bias.

2.4.2 System Error Model

2.4.2.1 Tilt angle error model.

The first error state equation that will be derived here is the angular error θ_G, which is represented in the navigation frame.

Recalling equation 2.7, let $\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q}\otimes\mathbf{q}_\omega$ be the derivative of the true-state quaternion and $\dot{\bar{\mathbf{q}}} = \frac{1}{2}\bar{\mathbf{q}} \otimes \mathbf{q}_{\bar{\omega}}$ represent the derivative of the nominal state.

The rate ω represents ω^b_{nb} and is adopted to simplify the notation in the following deduction. It will be here referenced as the true rate. Its nominal rate will then be $\bar{\omega} = \bar{\omega}^b_{nb}$. The relationship between both is given by:

$$\omega = \bar{\omega} + \delta\omega. \tag{2.30}$$

$\bar{\omega}$ is also defined to be

$$\bar{\omega} = \bar{\omega}^b_{ib} - \bar{\omega}^b_{in}, \tag{2.31}$$

where $\bar{\omega}^b_{ib} = \mathbf{u}_g - \mathbf{x}_g$ is the nominal inertial-to-body rate and $\bar{\omega}^b_{in}$ was defined in equation 2.11. The error $\delta\omega$ is given as

$$\delta\omega = -\delta\mathbf{x}_g - \mathbf{v}_g. \tag{2.32}$$

By writing the true quaternion as a function of its local pertubation and its normal value (equation 2.6), it can be seen that:

$$\dot{\mathbf{q}} = \frac{d}{dt}(\Delta\mathbf{q}^n \otimes \bar{\mathbf{q}}) = \Delta\dot{\mathbf{q}}^n \otimes \bar{\mathbf{q}} + \Delta\mathbf{q}^n \otimes \dot{\bar{\mathbf{q}}} \tag{2.33}$$

Replacing $\dot{\bar{\mathbf{q}}} = \frac{1}{2}\bar{\mathbf{q}} \otimes \mathbf{q}_{\bar{\omega}}$, in the above relationship gives:

$$\dot{\mathbf{q}} = \Delta\dot{\mathbf{q}}^n \otimes \bar{\mathbf{q}} + \frac{1}{2}\Delta\mathbf{q}^n \otimes \bar{\mathbf{q}} \otimes \mathbf{q}_{\bar{\omega}} \tag{2.34}$$

Now replacing $\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \mathbf{q}_{\omega} = \frac{1}{2}(\Delta\mathbf{q}^n \otimes \bar{\mathbf{q}}) \otimes \mathbf{q}_{\omega}$ in the last relationship gives

$$\frac{1}{2}\Delta\mathbf{q}^n \otimes \bar{\mathbf{q}} \otimes \mathbf{q}_{\omega} = \Delta\dot{\mathbf{q}}^n \otimes \bar{\mathbf{q}} + \frac{1}{2}\Delta\mathbf{q}^n \otimes \bar{\mathbf{q}} \otimes \mathbf{q}_{\bar{\omega}} \tag{2.35}$$

By recalling equation 2.30, it can be written that $\omega = \bar{\omega} + \delta\omega$, thus this previous equation becomes

$$\frac{1}{2}\Delta\mathbf{q}^n \otimes \bar{\mathbf{q}} \otimes \mathbf{q}_{\delta\omega} = \Delta\dot{\mathbf{q}}^n \otimes \bar{\mathbf{q}}. \tag{2.36}$$

Multiplying both sides by the right by $\bar{\mathbf{q}}^*$ results in

$$\Delta\dot{\mathbf{q}}^n = \frac{1}{2}\Delta\mathbf{q}^n \otimes \bar{\mathbf{q}} \otimes (\mathbf{q}_{\delta\omega} \otimes \bar{\mathbf{q}}^*) = \frac{1}{2}\Delta\mathbf{q}^n \otimes (\mathbf{R}_b^n\delta\omega). \tag{2.37}$$

For small angular perturbations (see appendix B.1), one may write

$$\Delta\mathbf{q}^n \approx [1 \quad \delta\theta_G^T/2]^T \tag{2.38}$$

where θ_G is defined as the tilt angle represented in the global frame. Substituting this definition in equation 2.37 finally gives

$$\begin{bmatrix} 0 \\ \delta\dot{\theta}_G/2 \end{bmatrix} = \frac{1}{2}\Delta\mathbf{q}^n \otimes (\mathbf{R}_b^n\delta\omega) = \begin{bmatrix} 0 & -(\mathbf{R}_b^n\delta\omega)^T \\ (\mathbf{R}_b^n\delta\omega) & -[(\mathbf{R}_b^n\delta\omega)\times] \end{bmatrix} \begin{bmatrix} 1 \\ \delta\theta_G/2 \end{bmatrix} + \mathbf{O}(||\theta_G||^2) \tag{2.39}$$

From the above equation, it can be seen that

$$(\mathbf{R}_b^n\delta\omega)^T\delta\theta_G = \mathbf{O}(||\theta_G||^2) \tag{2.40}$$

and, therefore, after neglecting second order terms, $\dot{\delta\theta}_G$ can be written as

$$\dot{\delta\theta}_G = \mathbf{R}_b^n\delta\omega = -\mathbf{R}_b^n\delta\mathbf{x}_g - \mathbf{R}_b^n\mathbf{v}_g. \tag{2.41}$$

where $\mathbf{R}_b^n = \mathbf{R}_b^n(\bar{\mathbf{q}}))$ can be calculated through

$$\mathbf{R}_b^n = \mathbf{I} + 2q_0[\vec{\mathbf{q}}\times] + 2[\vec{\mathbf{q}}\times]^2 \tag{2.42}$$

according to (TRAWNY; ROUMELIOTIS, 2005).

2.4.2.2 Bias error models.

The equations for the gyro bias error state $\delta\mathbf{x}_g$ is obtained by subtracting equation 2.13 written for the true bias state $\mathbf{x}_g$ by equation 2.13 written for the nominal state: $\bar{\mathbf{x}}_g$.

$$\dot{\delta\mathbf{x}}_g = \dot{\mathbf{x}}_g - \dot{\bar{\mathbf{x}}}_g = \mathbf{F}_g\ (\mathbf{x}_g - \bar{\mathbf{x}}_g) + \mathbf{v}_{b_g} = \mathbf{F}_g\ \delta\mathbf{x}_g + \mathbf{v}_{b_g} \tag{2.43}$$

Where we see the nature of the equation remains the same as for the bias state.

Analogously we obtain the model equation for the $\delta\mathbf{x}_a$; this time, using equation 2.15:

$$\dot{\delta\mathbf{x}}_a = \dot{\mathbf{x}}_a - \dot{\bar{\mathbf{x}}}_a = \mathbf{F}_a\ (\mathbf{x}_a - \bar{\mathbf{x}}_a) + \mathbf{v}_{b_a} = \mathbf{F}_a\ \delta\mathbf{x}_a + \mathbf{v}_{b_a} \tag{2.44}$$

2.4.2.3 Complete orientation error model.

After the discussion in the above subsections, the error state space equation can finally be written:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{or}(t)\mathbf{x}(t) + \mathbf{B}_{or}(t)\mathbf{w}(t) \tag{2.45}$$

where the state $\mathbf{x}(t) = [\delta\theta_G^T, \delta\mathbf{x}_g^T, \mathbf{x}_a^T]^T$, the input white Gaussian noise is $\mathbf{w}(t) = [\mathbf{w}_g^T, \mathbf{w}_{b_g}^T, \mathbf{w}_{b_a}^T]^T \in R^{9\times 1}$ and covariance matrix $\mathbf{Q} = diag([\sigma_g^2, \sigma_{b_g}^2, \sigma_{b_a}^2])$,

$$\mathbf{A}_{or}(t) = \begin{bmatrix} \mathbf{O}_3 & -\mathbf{R}_b^n(\mathbf{q}) & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{F}_g & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{F}_a \end{bmatrix}, \quad \mathbf{B}_{or}(t) = \begin{bmatrix} -\mathbf{R}_b^n(\mathbf{q}) & \mathbf{O}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{I}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{I}_3 \end{bmatrix} \tag{2.46}$$

Integrating equation 2.46 in the interval of time k to time $k+1$ gives:

$$\mathbf{x}(t_{k+1}) = \mathbf{e}^{(t_{k+1}-t_k)\mathbf{A}_{or}(s)}\mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{e}^{(t_{k+1}-s)\mathbf{A}_{or}(s)}\mathbf{B}(s)\mathbf{w}(s)ds \tag{2.47}$$

where one can recognize the separation between a state transition term and a noise term. In the transition term one can define the transition matrix:

$$\mathbf{F}_k = \mathbf{e}^{T\mathbf{A}_{or}(t_k)} \tag{2.48}$$

and the discretized noise term $\mathbf{w}_k$:

$$\mathbf{w}_k = \int_{t_k}^{t_{k+1}} \mathbf{e}^{(t_{k+1}-s)\mathbf{A}_{or}(s)}\mathbf{B}(s)\mathbf{w}(s)ds \tag{2.49}$$

An approximation for equation 2.48 can be obtained by a first order Taylor expansion in T (FARRELL, 2008),

$$\mathbf{F}_k \approx \mathbf{I} + \mathbf{A}_{or(k)}T \tag{2.50}$$

Thus the discretized version of equation 2.47 becomes

$$\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{w}_k \tag{2.51}$$

It is important to remember that by using equation 2.49 the covariance matrix $\mathbf{Q}_k = var(\mathbf{w}_k)$ can be derived from the integral (FARRELL, 2008):

$$\mathbf{Q}_k = \int_{t_k}^{t_{k+1}} \mathbf{e}^{(t_{k+1}-s)\mathbf{A}_{or}(s)} \mathbf{B}(s)\mathbf{Q}(s)\mathbf{B}^T(s) \left(\mathbf{e}^{(t_{k+1}-s)\mathbf{A}_{or}(s)}\right)^T ds \tag{2.52}$$

In the case where $\mathbf{F}_k$ is constant, $\mathbf{Q}_k$ is constant as well an is accumulated to $\mathbf{P}_k$ at each prediction step. $\mathbf{Q}_k$ in this case can be approximated by

$$\mathbf{Q}_k = \mathbf{G}_k\mathbf{Q}\mathbf{G}_k^T \quad \mathbf{G}_k = \sqrt{T}\mathbf{B}_k \tag{2.53}$$

which is valid if $||\mathbf{F}T|| << 1$ (FARRELL, 2008).

A special observation can be made in respect to $\mathbf{Q}_k$ when $\mathbf{F}_k$ is time varying. In order to take this into account (FARRELL, 2008) proposes to accumulate $\mathbf{Q}_k$ separately while measurements don't arrive

$$\mathbf{Q}(\tau_i, \tau_0) = \mathbf{F}(\tau_i, \tau_{i-1})\mathbf{Q}(\tau_{i-1}, \tau_0)\mathbf{F}(\tau_i, \tau_{i-1})^T + \mathbf{G}_k\mathbf{Q}\mathbf{G}_k^T\tau \tag{2.54}$$

and then update $\mathbf{P}_k$ at the end, right before the next measurement arrives, through the formula:

$$\mathbf{P}_{k+1} = \mathbf{F}_k\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{Q}_k, \quad \mathbf{Q}_k = \mathbf{Q}(\tau_N, \tau_0). \tag{2.55}$$

In all these expressions, τ is the time step of the prediction which runs at frequency $1/\tau$; τ_0 is the time where the last measurement arrived; τ_N is the time where the next measurement arrives; $\mathbf{F}(\tau_i, \tau_{i-1})$ is the transition matrix from τ_{i-1} to τ_i and; $\mathbf{Q}$ is the continuous covariance matrix.

In the above formula, one can see then that $\mathbf{Q}_k$ has the same structure as $\mathbf{P}_k$ propagation in algorithm 1 of the regular Kalman filters. This is not a coincidence. Moreover, as long as the rate of prediction is as fast as $1/T$ and $\mathbf{F}_k$ is calculated at each step, the accumulation of $\mathbf{Q}$ can be substituted by the accumulation of $\mathbf{P}$ (FARRELL, 2008). Therefore the structure of the state covariance propagation equations can remain the same in respect to those presented in algorithm 1.

Thus, in this case, the propagation equation for the state covariance is given by

$$\mathbf{P}_{k+1} = \mathbf{F}_k\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{G}_k\mathbf{Q}\mathbf{G}_k^T \quad \mathbf{G}_k = \sqrt{T}\mathbf{B}_k. \tag{2.56}$$

where even though the accumulation of the noise covariance is time varying, it is taken constant at each prediction step.

2.4.3 Measurement Error Model

2.4.3.1 Accelerometer Error model

In this estimator, the goal of using the accelerometer is to measure the Earth's gravity in respect to the body frame and use that information to correct the $\delta\theta_G^-$ estimate

obtained in the predict phase. More precisely, Earth's known (true-)gravitational field can be subtracted by the nominal gravitational field $\bar{\mathbf{g}}^n = \bar{\mathbf{R}}_n^b \bar{\mathbf{g}}^b$ to obtain a measurement of the gravitational residue, which can be used in θ_G^- correction.

The residue of the gravitational field vector in the body frame can be written as $\delta\mathbf{g}^n = \mathbf{g}^n - \bar{\mathbf{g}}^n$, where $\bar{\mathbf{g}}^n = \bar{\mathbf{x}}_a - \mathbf{y}_a$ is defined as the nominal value of the gravity vector. Moreover, one can express this residual as

$$\delta\mathbf{g}^n = \mathbf{g}^n - \bar{\mathbf{g}}^n = \mathbf{g}^n - \mathbf{R}_b^n(\bar{\mathbf{q}})(\bar{\mathbf{x}}_a - \mathbf{y}_a) \tag{2.57}$$

From the angular perturbation relationships for matrices (see appendix B.3) it is possible to write the local gravity field as a function of the the tilt angle θ_G (see (TRAWNY; ROUMELIOTIS, 2005), (FARRELL, 2008)). As a consequence of the definition of θ_G, the θ_G is related to the matrix $\mathbf{R}_b^n(\mathbf{q})$ and the matrix $\mathbf{R}_b^n(\bar{\mathbf{q}})$ by

$$\mathbf{R}_b^n(\mathbf{q}) = (\mathbf{I} + [\delta\theta_G\times])\mathbf{R}_b^n(\bar{\mathbf{q}}) \tag{2.58}$$

Transposing both sides of this equation yields

$$\mathbf{R}_b^n(\bar{\mathbf{q}}) = (\mathbf{I} - [\delta\theta_G\times])\mathbf{R}_b^n(\mathbf{q}) \tag{2.59}$$

Substituting this equation in the residue relationship,

$$\delta\mathbf{g}^n = \mathbf{g}^n - (\mathbf{I} - [\delta\theta_G\times])\mathbf{R}_b^n(\mathbf{q})(\bar{\mathbf{x}}_a - \mathbf{y}_a) \tag{2.60}$$

and by considering the model equation of the accelerometer (section 1.3.3) and defining $\mathbf{v}_a = \mathbf{v}'_a + \mathbf{a}_{ib}^b$:

$$\delta\mathbf{g}^n = \mathbf{g}^n - (\mathbf{I} - [\delta\theta_G\times])\mathbf{R}_b^n(\mathbf{q})(\bar{\mathbf{x}}_a + \mathbf{g}^b - \mathbf{x}_a - \mathbf{v}_a) \tag{2.61}$$

$$= \mathbf{g}^n - (\mathbf{I} - [\delta\theta_G\times])\mathbf{R}_b^n(\mathbf{q})(\mathbf{g}^b - \delta\mathbf{x}_a - \mathbf{v}_a). \tag{2.62}$$

After neglecting the second order error terms, one arrives at

$$\delta\mathbf{g}^n = [\delta\theta_G\times]\mathbf{g}^n + \mathbf{R}_n^b(\mathbf{q})(\delta\mathbf{x}_a + \mathbf{v}_a) \tag{2.63}$$

$$= -[\mathbf{g}^n\times]\delta\theta_G + \mathbf{R}_n^b(\mathbf{q})(\delta\mathbf{x}_a + \mathbf{v}_a). \tag{2.64}$$

From the above relationship, one deduces that

$$\mathbf{H}_a = \frac{\mathbf{d}}{\mathbf{dx}}\delta\mathbf{g}^n = [-[\mathbf{g}^n\times] \quad \mathbf{O}_3 \quad \mathbf{R}_b^n]^T \tag{2.65}$$

Since $\mathbf{g}^n = [0\ 0\ g_e]^T$, it is possible to see that the gravity vector residue only corrects $\delta\theta_G$ in the first and second components.

In wishing to use the accelerometer only to measure the gravity field in the body frame, it is important to note that the covariance matrix of $\mathbf{v}_a$ is not the one estimated in section 3.1, since this random variable also is also composed of the acceleration $\mathbf{a}_{ib}^b$. The treatment of this inconvenience will later be explained in the section dedicated to the algorithm presentation.

2.4.3.2 Magnetometer Error model

As mentioned in the last section, the gravity vector does not add useful information in respect to the third component (yaw) of the tilt angle vector θ_G. In order to fill in that gap, the magnetometer is also used in the update phase to provide corrections.

In a similar manner to the accelerometer sensor model deduction, the magnetometer sensor model can be obtained by first writing down the error residue for the Eath's magnetic field, defined by

$$\delta\mathbf{m}^n = \mathbf{m}^n - \bar{\mathbf{m}}^n \tag{2.66}$$

where the nominal magnetometer state is defined as $\bar{\mathbf{m}}^n = \mathbf{R}_b^n\mathbf{y}_m$ and the constant magnetometer bias $\mathbf{x}_m$ (equation 2.16) was considered known and accounted for, canceling out in the above equation. Writing an equivalent equation for (2.60) in the magnetometer case yields

$$\delta\mathbf{m}^n = \mathbf{m}^n - (\mathbf{I} - [\delta\theta_G\times])\mathbf{R}_b^n(\mathbf{q})\mathbf{y}_m = \mathbf{m}^n - (\mathbf{I} - [\delta\theta_G\times])\mathbf{R}_b^n(\mathbf{q})(\mathbf{m}^b + \mathbf{v}_m) = \tag{2.67}$$

$$= [\delta\theta_G\times]\mathbf{m}^n - \mathbf{R}_b^n(\mathbf{q})\mathbf{v}_m \tag{2.68}$$

$$= -[\mathbf{m}^n\times]\delta\theta_G - \mathbf{R}_b^n(\mathbf{q})\mathbf{v}_m \tag{2.69}$$

where it is noted that the magnetometer residue provides correction for the third component of $\delta\theta_G$. The second component affected by the residue correction has been eliminated for greater precision, since the magnetic vertical component is often a source of uncertainty (FARRELL, 2008).

Finally, one can then write

$$\mathbf{H}_m = \frac{\mathbf{d}}{\mathbf{dx}}\delta\mathbf{m}^n = [-[\mathbf{m}^n\times] \quad \mathbf{O}_3 \quad \mathbf{O}_3] \tag{2.70}$$

Rejecting the influence on the second component of $\delta\theta_G$ in this equation yields

$$\mathbf{H}_m = \frac{\mathbf{d}}{\mathbf{dx}}\delta\mathbf{m}^n = [H_M \quad \mathbf{O}_3 \quad \mathbf{O}_3], \quad H_M = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & m_e \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.71}$$

2.4.3.3 Complete Orientation Error model

From the preceding subsections, it can be concluded that the measurement model equations for the orientation filter are

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t)) + \mathbf{N}(t)\mathbf{v}(t), \quad \mathbf{h}(\mathbf{x}) = [\delta\mathbf{m}^{nT} \quad \delta\mathbf{g}^{nT}]^T, \quad \mathbf{N}(t) = \begin{bmatrix} -\mathbf{R}_b^n(\mathbf{q}) & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{R}_b^n(\mathbf{q}) \end{bmatrix} \tag{2.72}$$

$$\frac{\mathbf{dh}}{\mathbf{dx}}(\mathbf{x}) = \mathbf{H} = [\mathbf{H}_m \quad \mathbf{H}_a]^T \tag{2.73}$$

Where $\mathbf{v} = [\mathbf{v}_m, \mathbf{v}'_a]^T \in R^{6\times1}$ is a white, zero mean, Gaussian process, with covariance matrices defined in section 3.1.

The discretization of the above equations leads to

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{N}_k\mathbf{v}_k, \quad \mathbf{H}_k = \mathbf{H}. \tag{2.74}$$

with the covariance matrix correction of $\mathbf{P}_k$ given by

$$\mathbf{P}^+_{k+1} = (\mathbf{I}_9 - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}^-_{k+1}(\mathbf{I}_9 - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{2.75}$$

where the Joseph's form was used in (2.75) to preserve the symmetry of $\mathbf{P}_k$ (FARRELL, 2008). Here $\mathbf{K}_k$ is given by

$$\mathbf{K}_k = \mathbf{P}^-_{k+1}\mathbf{H}_k^T\mathbf{S}_k^{-1} \tag{2.76}$$

with

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}^-_{k+1}\mathbf{H}_k^T + \mathbf{R}_k. \tag{2.77}$$

$\mathbf{R}_k$ is equal to the continuous time covariance matrix $\mathbf{R}$ divided by T, as shown in (BROWN; HWANG et al., 1992).

2.4.4 The Algorithm

The orientation ESKF algorithm can be illustrated by the following pseudo-algorithm. The frequency of the predict phase is 100Hz (accelerometer frequency), while the frequency of the update phase is of 50Hz. The initial nominal state was set considering the an estimate given by another estimator or by a known value (when this was the case in simulation). As a consequence the state covariance matrix was intialized with a diagonal matrix with low values.

Algorithm 4 ESKF filter for orientation estimation with error state $\delta\mathbf{x} = [\delta\theta_G, \delta\mathbf{x}_g, \delta\mathbf{x}_a]^T$ and measured state $\mathbf{y} = [\mathbf{g}^b, \mathbf{m}^b]^T$. **Inputs**: $\mathrm{u}_g, \mathbf{y}_a, \mathbf{y}_m, \mathbf{v}^n, \mathbf{p}$. **Outputs** :$\phi$, θ, ψ

procedure INITIALIZATION
    Set $\mathbf{q}_0$, $\mathbf{P}_0$ and $0 \leftarrow \mathbf{x}_0$
while $t_k \neq t_{END}$ **do**
    procedure PREDICT
        Propagate $\bar{\mathbf{q}}_k$, $\bar{\mathbf{x}}_{g(k)}$ and $\bar{\mathbf{x}}_{a(k)}$ with Eqs. 2.27, 2.28 and 2.29.
        Propagate $\delta\mathbf{x}_k^-$ by Eq. 2.51 with $\mathbf{R}_b^n(\bar{\mathbf{q}}_k)$.
        Propagate $\mathbf{P}_k^-$ by Eq. 2.56.
    procedure UPDATE
        Calculate $\mathbf{R}_b^n(\bar{\mathbf{q}}_k)$ by Eq. 2.42
        Calculate $\mathbf{H}_k$ by Eq. 2.73 from $\bar{\mathbf{g}}_k^b$ and $\bar{\mathbf{m}}_k^b$
        Calculate residues $\delta\mathbf{g}_k^n$ and $\delta\mathbf{m}_k^n$ from Eqs. 2.57 and 2.66.
        Calculate error state $\delta\mathbf{x}_k^+$ using Eq.2.76.
        Calculate $\mathbf{P}_k^+$ by Eq.2.75.
        Extract $\delta\theta_{G(k)}^+$, $\delta\mathbf{x}_{g(k)}^+$, $\delta\mathbf{x}_{a(k)}^+$ from $\delta\mathbf{x}_k^+$.
        Calculate $\Delta\mathbf{q}_k^{n+}$ by Eq.2.38
        Update $\bar{\mathbf{q}}_k$, $\bar{\mathbf{x}}_{g(k)}$ and $\bar{\mathbf{x}}_{a(k)}$ by Eqs. 2.6 and 1.26.
        Reset error state $0 \leftarrow \delta\mathbf{x}_k^+$
    procedure QUAT2EULER($\bar{\mathbf{q}}$)
    procedure OUTPUT(ϕ, θ, ψ)

It is important to say that the correction by the gravity residual $\delta\mathbf{g}^n$ does not always occur. It depends on a condition suggested in (FARRELL, 2008):

$$\mu_k = |||\mathbf{y}_a|| - g_e| < \beta_\mu \tag{2.78}$$

This is because the accelerometer is expected to always measure the gravity field. Therefore, when the body is accelerating and, thus, the gravity vector can not be deduced from $\mathbf{y}_a$, no corrections are realized by the acceleromter.

2.5 Position Estimator

The objective of this position estimator is to localize the body frame in respect to the Earth's surface. In addition to position (latitude, longitude and height), the navigation estimator also estimates the body's velocity vector. In this formulation, an Extended Kalman Filter (EKF) was chosen, where the estimated states are the entire states. In this case, the states are position in navigational frame $\mathbf{p} = [\phi, \lambda, h]$, velocity vector $\mathbf{v} = [v_n, v_e, v_d]^T$, and the accelerometer bias $\mathbf{x}_a$, which role will be further explained.

The estimator essentially combines GNSS information (i.e. position and velocity) with Inertial Navigation System information (acceleration) in order to obtain an estimate of position and velocity (INOUE, 2011). In the predict step the acceleration from the INS is integrated in order to provide an estimation of velocity; and the last estimate of velocity is integrated in order to estimate position. However, before being integrated, the acceleration vector gets corrected by the the last estimated bias $\mathbf{x}_a$ in order to increase the precision of this step, which is likely to diverge very quickly due to round off errors and other model imperfections. The predic step runs at 100Hz, which is the frequency in which the accelerometer measurements arrive.

In the update step, the estimation of position and velocity gets corrected by the GNSS measurements, which arrive at a lower rate of 10Hz.

2.5.1 System Model

As seen in section 2.2.2, the equations that govern position in the navigation frame are nonlinear and therefore, at each prediction step, the nonlinear numerical integration of these equations will be realized. In this filter, a Euler integration was considered as a first level of precision.

It is important to state that the acceleration applied in equation 2.10 is not the output of the accelerometer, but the best estimate of the acceleration, which can be obtained by equation 2.14, that is:

$$\bar{\mathbf{a}}_{ib}^n = \mathbf{R}_b^n(\mathbf{y}_a - \bar{\mathbf{x}}_a) + \mathbf{g}^n \tag{2.79}$$

To complete the EKF model formulation it is necessary to linearize equations 2.9 and 2.10 in respect to each one of the considered states: $\mathbf{p}$, $\mathbf{v}_e^g$ and $\mathbf{x}_a$. The resultant linearization gives:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_{pos}(t)\mathbf{x}(t) + \mathbf{B}_{pos}(t)\mathbf{w}(t) \tag{2.80}$$

where the state $\mathbf{x}(t) = [\mathbf{p}^T, (\mathbf{v}_e^g)^T, \mathbf{x}_a^T]^T$, the input white Gaussian noise is $\mathbf{w}(t) = [\mathbf{w}_a^T, \mathbf{w}_{b_a}^T]^T \in R^{6\times 1}$ with covariance matrix defined by $\mathbf{Q} = diag([\sigma_a^2, \sigma_{b_a}^2])$, (INOUE,

2011),

(2.81)

$$\mathbf{A}_{pp} = \begin{bmatrix} 0 & 0 & \frac{-\bar{v}_n}{(R_M+\bar{h})^2} \\ \frac{\bar{v}_e \sin\bar{\phi}}{(R_N+\bar{h})\cos^2\bar{\phi}} & 0 & \frac{-\bar{v}_e}{(R_N+\bar{h})^2\cos\bar{\phi}} \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A}_{pv} = \begin{bmatrix} \frac{1}{R_M+\bar{h}} & 0 & 0 \\ 0 & \frac{1}{(R_N+\bar{h})\cos\bar{\phi}} & 0 \\ 0 & 0 & -1; \end{bmatrix} \tag{2.82}$$

$$\mathbf{A}_{vp} = \begin{bmatrix} -2\Omega_N\bar{v}_e - \frac{\bar{v}_e^2}{(R_N+\bar{h})\cos^2\bar{\phi}} & 0 & \frac{\tan\bar{\phi}\bar{v}_e^2}{(R_N+\bar{h})^2} - \frac{\bar{v}_n\bar{v}_d}{(R_M+\bar{h})^2} \\ 2(\Omega_N\bar{v}_n + \Omega_D\bar{v}_d) + \frac{\bar{v}_e\bar{v}_n}{(R_N+\bar{h})\cos^2\bar{\phi}} & 0 & \frac{-\tan\bar{\phi}\bar{v}_e\bar{v}_n}{(R_N+\bar{h})^2} - \frac{\bar{v}_e\bar{v}_d}{(R_N+\bar{h})^2} \\ -2\bar{v}_e\Omega_D & 0 & \frac{\bar{v}_e^2}{(R_N+\bar{h})^2} + \frac{\bar{v}_n^2}{(R_M+\bar{h})^2} \end{bmatrix} \tag{2.83}$$

$$\mathbf{A}_{vv} = \begin{bmatrix} \frac{\bar{v}_d}{R_M+\bar{h}} & 2(\Omega_D - \frac{\bar{v}_e\tan\bar{\phi}}{R_N+\bar{h}}) & \frac{\bar{v}_n}{R_M+\bar{h}} \\ \frac{\bar{v}_e\tan\bar{\phi}}{R_N+\bar{h}} - 2\Omega_D & \frac{\bar{v}_n\tan\bar{\phi}}{R_N+\bar{h}} + \frac{\bar{v}_d}{R_N+\bar{h}} & \frac{\bar{v}_e}{R_N+\bar{h}} + 2\Omega_N \\ -2\frac{\bar{v}_n}{R_M+\bar{h}} & -2(\frac{\bar{v}_e}{R_N+\bar{h}} + \Omega_N) & 0 \end{bmatrix} \tag{2.84}$$

with $\Omega_E = -\omega_{ie}\sin\bar{\phi}$ and $\Omega_D = \omega_{ie}\cos\bar{\phi}$.

After discretization, these equations become

$$\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{w}_k \tag{2.85}$$

where

$$\mathbf{F}_k \approx \mathbf{I} + \mathbf{A}_{pos(k)}T, \quad \mathbf{G}_k \approx \mathbf{B}_{pos(k)}\sqrt{T} \tag{2.86}$$

In this discretization, a first order approximation in T was used to calculate the transition matrix $\mathbf{F}_k$ and $\mathbf{G}_k$ accounts for the discretization of the process noise covariance matrice obtained in section 3.1. The condition $||FT|| << 1$ for which $\mathbf{G}_k$'s expression in 2.86 holds (see (FARRELL, 2008)) was found to be true in practice in this implementation.

As a consequence, the propagation equation for the $\mathbf{P}_k$ state covariance matrix is then

$$\mathbf{P}_{k+1}^- = \mathbf{F}_k\mathbf{P}_k^+\mathbf{F}_k^T + \mathbf{Q}_k \tag{2.87}$$

where $\mathbf{Q}_k$ can be defined as

$$\mathbf{Q}_k = \mathbf{G}_k\mathbf{Q}\mathbf{G}_k^T \tag{2.88}$$

2.5.1.1 Measurement Model

Differently from the previous model, the measurement model is linear, since the measurements are related to the states in a linear manner:

$$\mathbf{z}(t) = \mathbf{h}(\mathbf{x}) + \mathbf{v}(t), \quad \mathbf{h}(\mathbf{x}) = [\mathbf{p}^T \quad \mathbf{v}_{NED}^T]^T \tag{2.89}$$

$$\frac{\mathbf{dh}}{\mathbf{dx}}(\mathbf{x}) = \mathbf{H} = [\mathbf{I}_6 \quad \mathbf{O}_{6\times 3}] \tag{2.90}$$

Where $\mathbf{v}(t) \in R^{6\times 1}$ is a uncorrelated, zero mean, Gaussian process, with its covariance matrices defined through results in section 3.1. In the above equations we also have $\mathbf{y}(t) = [\mathbf{p}_{GNSS}^T, \quad \mathbf{v}_{GNSS}^T]^T$.

The discretization of the above equations leads to

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{H}_k = \mathbf{H}. \tag{2.91}$$

with the covariance matrix correction of $\mathbf{P}_k$ given by

$$\mathbf{P}_{k+1}^+ = (\mathbf{I}_9 - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k+1}^-(\mathbf{I}_9 - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{2.92}$$

where the Joseph's form was used in 2.75 in order to preserve symmetry of $\mathbf{P}_k$ ((FARRELL, 2008)). The Kalman gain $\mathbf{K}_k$ is given by

$$\mathbf{K}_k = \mathbf{P}_{k+1}^-\mathbf{H}_k^T\mathbf{S}_k^{-1} \tag{2.93}$$

where

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k+1}^-\mathbf{H}_k^T + \mathbf{R}_k. \tag{2.94}$$

$\mathbf{R}_k$ is equal to the continuous time covariance matrix $\mathbf{R}$ devided by T, as shown in (BROWN; HWANG et al., 1992).

2.5.2 The Algorithm

The position EKF can be illustrated by the following pseudo-algorithm. The frequency of the predict phase is 100Hz (accelerometer frequency), while the frequency of the update phase is of 10Hz (GNSS frequency). The initial state was set considering the known GNSS position and velocity of the vehicle, which is a good first estimate. As a consequence the state covariance matrix was intialized with a diagonal matrix with low values.

Algorithm 5 EKF filter for position and velocity estimation with estimated state $\mathbf{x}(t) = [\mathbf{p}^T, (\mathbf{v}_e^g)^T, \mathbf{x}_a^T]^T$, and measured state $\mathbf{y}(t) = [\mathbf{p}_{GNSS}^T, \mathbf{v}_{GNSS}^T]^T$. **Inputs**: $\mathbf{y}_a$, $\mathbf{R}_b^n(\mathbf{q})$, $\mathbf{p}_{GNSS}$ and $\mathbf{v}_{GNSS}$. **Outputs**: $\mathbf{p}$ and $\mathbf{v}_e^g$

procedure INITIALIZATION
    Set $\mathbf{P}_0$ and $\mathbf{x}_0$
while $t_k \neq t_{END}$ **do**
    procedure PREDICT
        Propagate $\hat{\mathbf{x}}_k$ by Eqs. 2.9, 2.10 and 2.79.
        Propagate $\mathbf{P}_k$ by Eq. 2.87.
    procedure UPDATE
        Calculate residues $\delta\mathbf{p}_k = \mathbf{p}_{GNSS} - \hat{\mathbf{p}}$ and $\delta\mathbf{v}_k^n = \mathbf{v}_{GNSS}^n - \hat{\mathbf{v}}_k^n$.
        Correct state $\hat{\mathbf{x}}_k$ using Eq.2.93.
        Correct $\mathbf{P}_k$ by Eq.2.92.
    procedure OUTPUT($\hat{\mathbf{p}}$, $\hat{\mathbf{v}}_e^g$)

3 RESULTS

After formulating the filters in the last chapter, it is now time to present the results found in tests where the estimators were applied.

In the following, the tests with real flight data were realized with an Arator 5B, a fixed wing RPA manufactured by Xmobots (Figure 8). This RPA was equipped with a system capable of transmitting sensor outputs to a computer, where the data could be processed after flight using the estimators here developed. As means of comparison, the Kalman filter outputs were compared to the RPA's own estimator outputs.

Figure 8: Xmobots fixed wing RPA Arator 5B. Source (XMOBOTS, 2019a).

Some characteristics of the Arator 5B are presented in the following table:

Name	MTOW (Maximum Takeoff Weight)	Wingspan	Cruise Speed	Nominal Autonomy
Xmobots Arator 5B	3.5 kg	1.2 m	57.6 km/h	66 minutes

Table 1: Xmobots Arator 5B technical specifications. Source (XMOBOTS, 2019a)

3.1 Co-variance matrix estimation for an RPA: test and results

In order to be able to test the Kalman Filter estimators here developed with real flight data, the inertial sensors of the experiment RPA (Arator 5B) had its noise characterized by the Allan Variance method mentioned above. The test consisted of steadily

placing the RPA in a non perturbed environment with all systems on, except the motor and collecting the sensor data in a computer running MATLAB.

As explained in (XING; GEBRE-EGZIABHER, 2008), the sensor time data series for the noise characterization does not need to be too long, since these sensors are not meant to be used for large time intervals without corrections, being unnecessary to catch the characteristics of long time correlations. A 40 min interval was then chosen and the Allan Variance method presented in appendix A was calculated and plotted with the help of a MATLAB tool developed by professor Roberto Santos Inoue of the Federal University of São Carlos (see reference (INOUE, 2011)). The tool implements the equations in (XING; GEBRE-EGZIABHER, 2008) to give more agility to the characterization process as it permits to vary the Gauss-Markov parameters (section 2.3.3) and implement the curves immediately.

The method was applied to the magnetometer, the accelerometer and the gyroscope in each one of their 3 axes. The table 2 below shows the obtained values for σ^2 and τ_{b_w}.

Accelerometer	$\sigma^2_{\mathbf{w}}[(\mathbf{m^2/s})]$	$\sigma^2_{\mathbf{b_1}}[(\mathbf{m^2/s^2})]$	$\tau_{\mathbf{b_1}}[\mathbf{s}]$
$x-axis$	5.9×10^{-5}	1.1×10^{-7}	237
$y-axis$	4.4×10^{-5}	4.9×10^{-7}	576
$z-axis$	10.0×10^{-5}	4.0×10^{-8}	772
Gyroscope	$\sigma^2_{\mathbf{w}}[(\mathbf{rad^2/s})]$	$\sigma^2_{\mathbf{b_1}}[(\mathbf{rad^2/s^3})]$	$\tau_{\mathbf{b_1}}[\mathbf{s}]$
$x-axis$	4.5×10^{-7}	8.1×10^{-10}	521
$y-axis$	3.9×10^{-7}	9.6×10^{-8}	518
$z-axis$	3.3×10^{-7}	4.4×10^{-10}	278
Magnetometer	$\sigma^2_{\mathbf{w}}[(\mathbf{T^2s})]$	$\sigma^2_{\mathbf{b_1}}[\mathbf{T^2/s}]$	$\tau_{\mathbf{b_1}}[\mathbf{s}]$
$x-axis$	3.2×10^{-13}	–	–
$y-axis$	1.2×10^{-14}	–	–
$z-axis$	4.1×10^{-15}	–	–

Table 2: Values of σ^2 and τ for each sensor.

3.2 Orientation Estimator

The test of the orientation filter was done in two parts: a first one using simulated data and a second one using real flight data. As a means of comparison, the orientation filter response was also compared to Xmobots RPA estimated angles in the real flight data tests.

3.2.1 Simulated data test

For this test a desired trajectory was created in terms of Euler angles (ϕ, θ, ψ) for each time step t_k in a discretized interval of sampling time $T = 0.01s$. The simulated trajectory describes a square in the xy plane, which is the type of trajectory that mapping drones usually fly by following each way-point at a time. During each straight portion

there are also some sinusoidal oscillations. The trajectory can thus be divided in four portions with one different heading each: north, east, south and west.

After trajectory generation, the Euler angles were derived in time in order to obtain their respective rates $(\dot{\phi}_k, \dot{\theta}_k, \dot{\psi}_k)$ at each t_k. These rates were then converted to their corresponding body rates (p_k, q_k, r_k), to which known white Gaussian noise signals and Gauss-Markov process type biases were added (Figure 9). All noises were generated using MATLAB *randn* function.

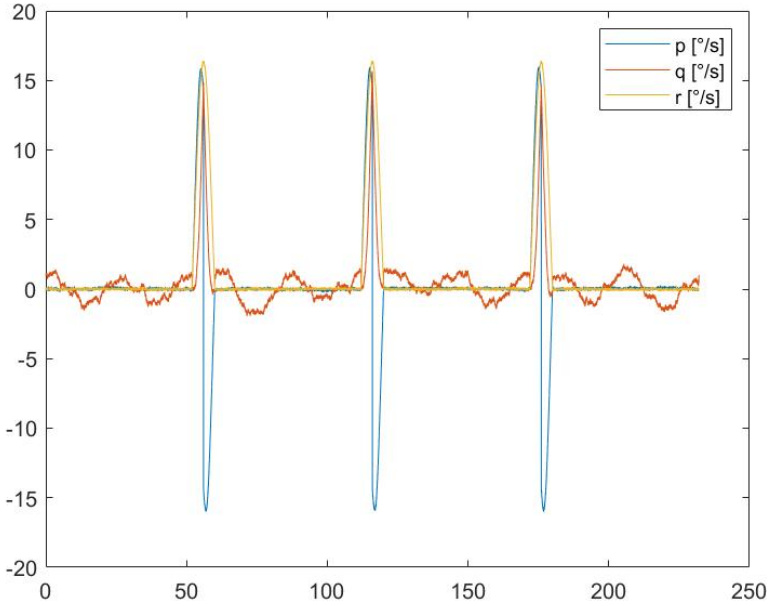

Figure 9: Generated body rates

The magnetometer and the accelerometer signals were generated by calculating

$$\mathbf{g}^b(t_k) = \mathbf{R}_n^b(t_k)\mathbf{g}^n(t_k) \tag{3.1}$$

$$\mathbf{m}^b(t_k) = \mathbf{R}_n^b(t_k)\mathbf{m}^n(t_k) \tag{3.2}$$

for each time step t_k using equation 2.1. After these rotations, white Gaussian noises and a Gauss-Markov process were added to create the biases. Figure 10 below shows the generated signals.

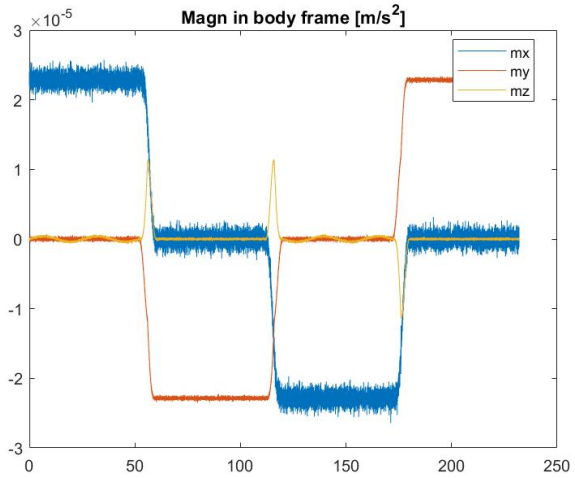

Figure 10: Generated magnetometer signal

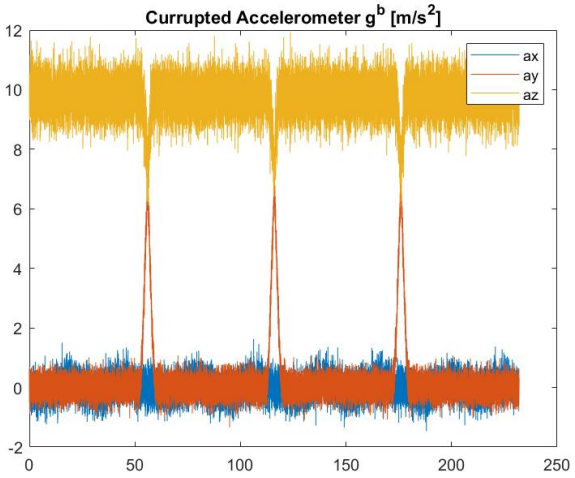

Figure 11: Generated accelerometer signal

In the tests, two different implementations for the orientation filter were used which differed only by how the nominal quaternion $\bar{\mathbf{q}}$ was updated. The fist one used only quaternions, in the sense that it followed exactly what was described in section 2.4.2 and algorithm 4. The second, replaced updating the quaternion in equation 2.6, by first updating its corresponding matrix $\mathbf{R}_b^n$ by equations B.19 and B.17, and only calculating its matrix afterwards.

In the second case a inconvenience appeared in respect to the matrix to quaternion conversion. As indicated in equation B.20, the term in the denominator $4q_0$ approached zero for the 3rd part of the simulated trajectory. This caused discontinuities in the orientation estimation as the matrix was numerically unstable. Although some methods exist to counter the problem (see (SARABANDI; THOMAS, 2018)), the quaternion update formulation still showed to be less unstable and therefore was maintained.

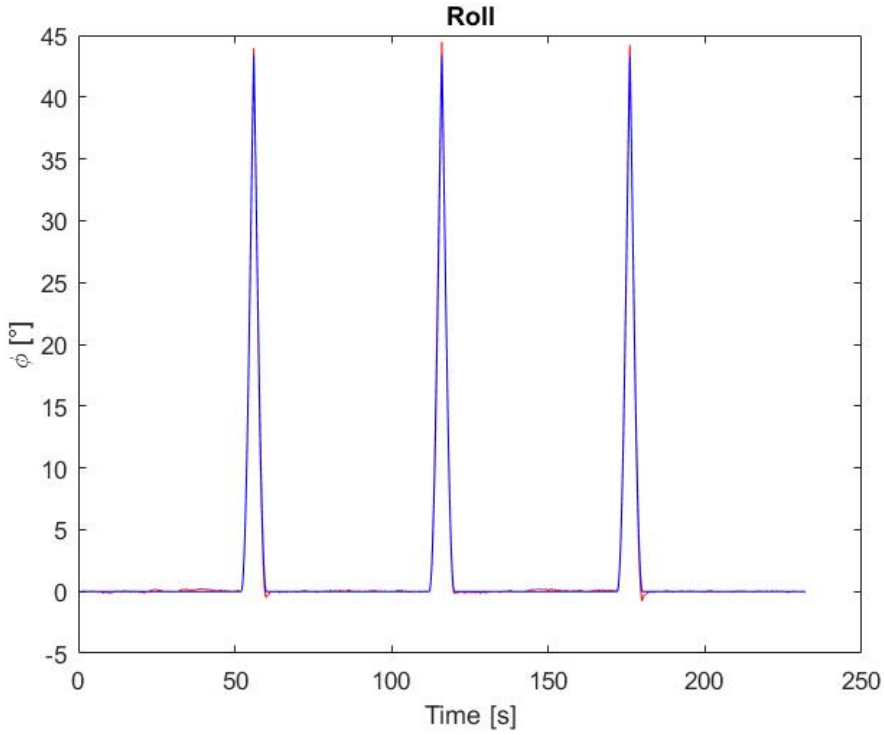

Figure 12: Roll angle comparison

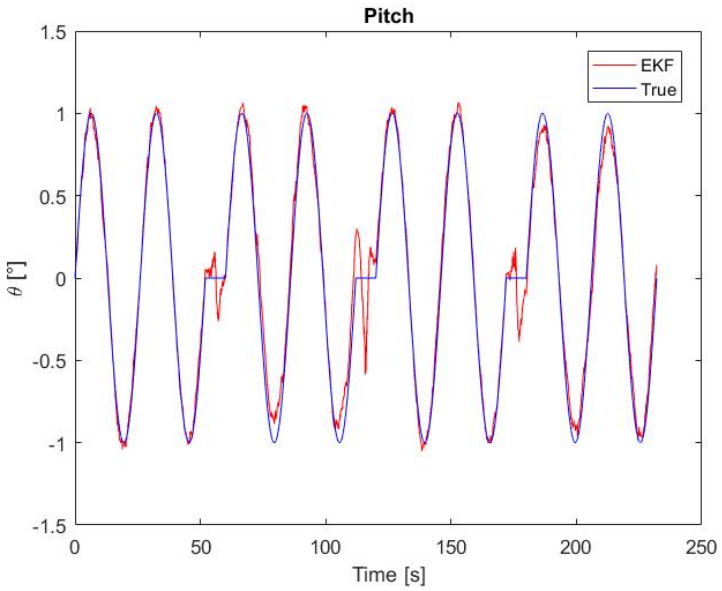

Figure 13: Pitch angle comparison

The filtered Euler angles are shown in figures 12-14, where they are compared to their corresponding true value. In all three cases, the filter converges and tracks the realized angles.

To define parameters for a performance analysis, the Euler error vector $\boldsymbol{\Theta}_{\mathbf{e}}(t) = \boldsymbol{\Theta}_{true}(t) - \hat{\boldsymbol{\Theta}}(t)$ is defined as a function of time. The 2-norm for this vector at each time

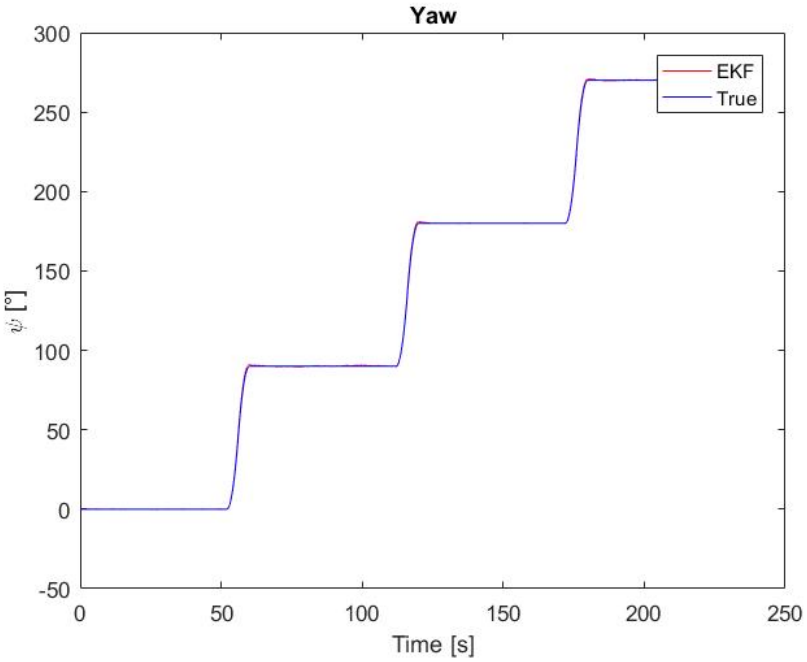

Figure 14: Yaw angle comparison

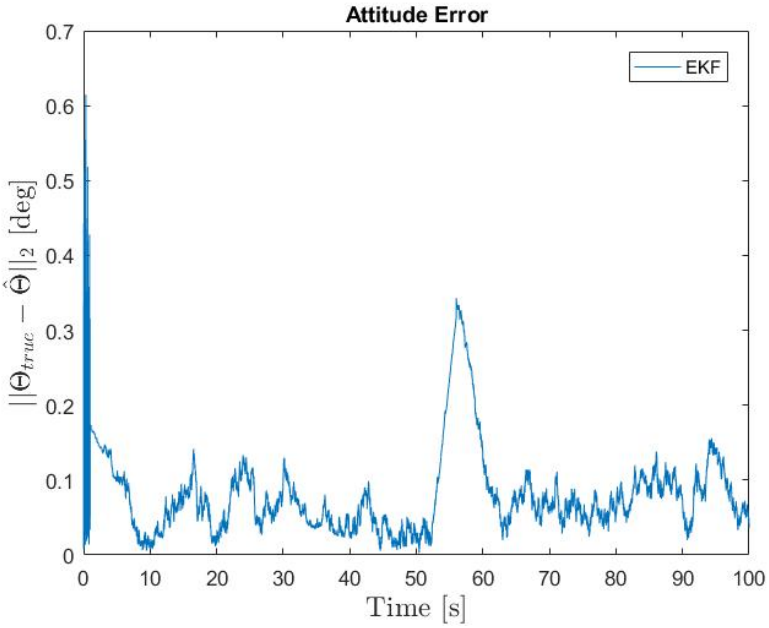

Figure 15: Time angle error analysis for simulated trajectory for the ESKF orientation filter.

step is presented in the plot of figure 15 for a given realization. To analyze the estimator's performance, 30 simulations were performed. By taking the standard deviation of the estimated time-series in respect to the true time-series, it was possible to analyze the state covariance estimation derived from the state covariance matrix P. The plot in figure 16.

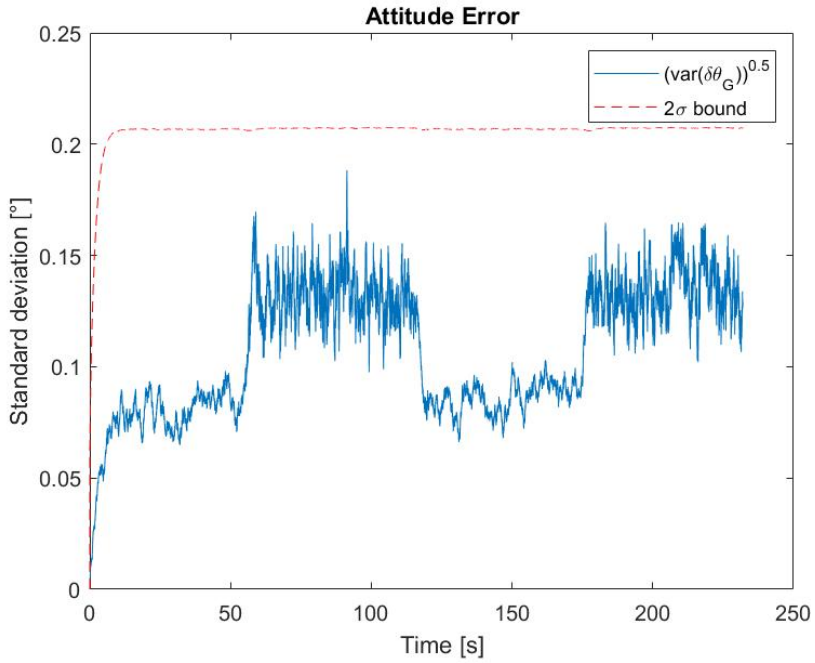

Figure 16: Standard deviation of the attitude error norm compared to two times the trace of the state covariance matrix $\mathbf{P}$ ($2\sigma = 2\sqrt{Tr(\mathbf{P}_k)}$), taken at a given realization.

The $\mathbf{L}_2$ norm

$$\mathbf{L}_2(\mathbf{\Theta_e}) = \frac{1}{t_f - t_i} \int_{t_i}^{t_f} ||\mathbf{\Theta_e}(t)||_2^2 dt \tag{3.3}$$

computed for 30 simulations showed to be smaller than 1 degree.

A last analysis was carried in respect to the mean time for convergence. The defined criteria for convergence was to observe the time the algorithm took to arrive a 2-norm attitude error 0.2 degrees. In 30 runs, the mean time for convergence was found to be 5.23 seconds.

3.2.2 Real flight data

The test with real flight data used sensor data from Xmobots Arator 5B in a quick flight test. The executed trajectory can been seen in figure 20 of the position estimator section, since it was the same flight. It is possible to distinguish several curves in the roll angle data, separated by portions of level flight.

In general, it is possible to see that results agree through many portions of flight.

3.3 Position Estimator

The position estimator was tested only using real flight data. As a means of comparisson, the GNSS position and velocity were used, as they are known to be precise (see Figure 20).

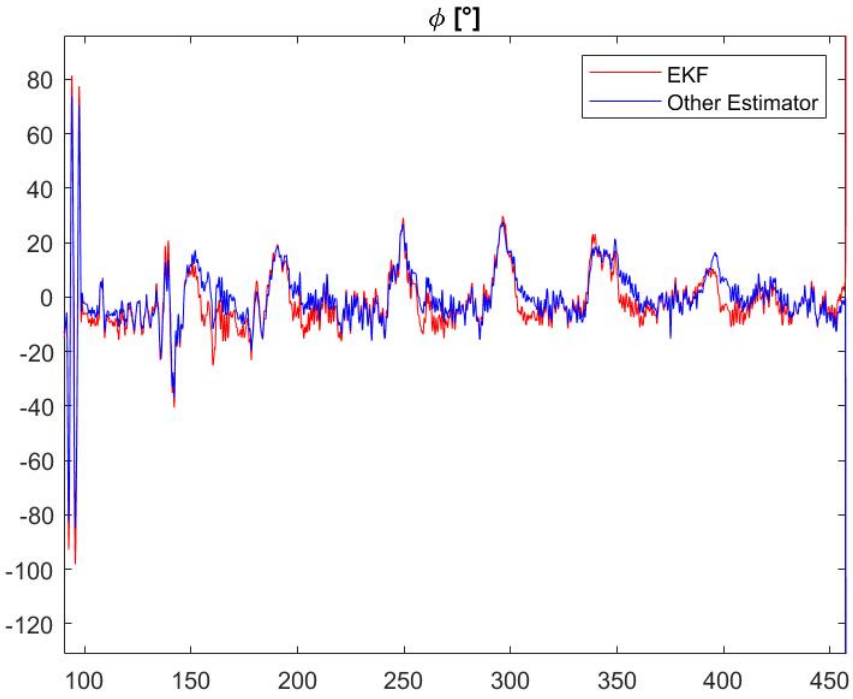

Figure 17: Roll angle comparison

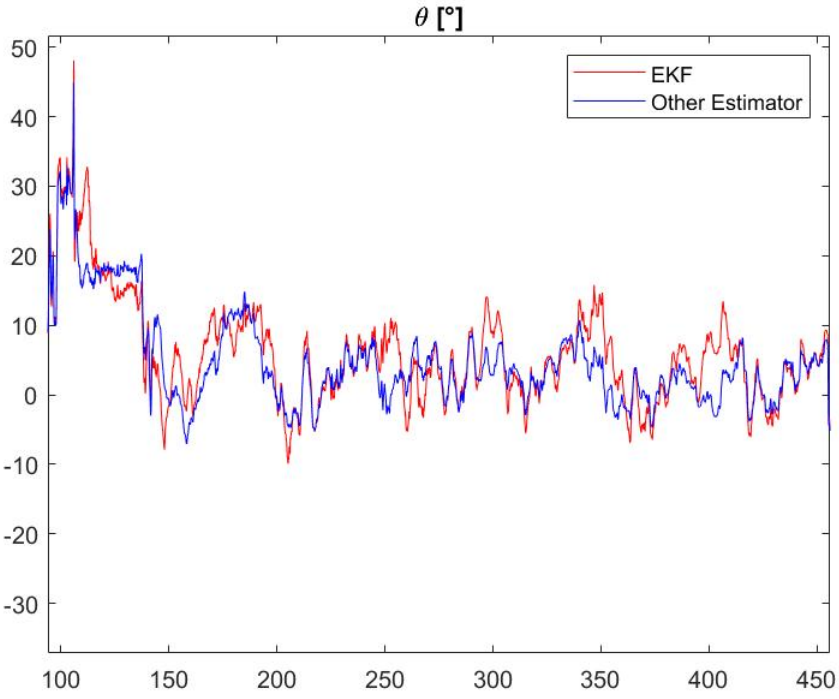

Figure 18: Pitch angle comparison as a function of time in seconds.

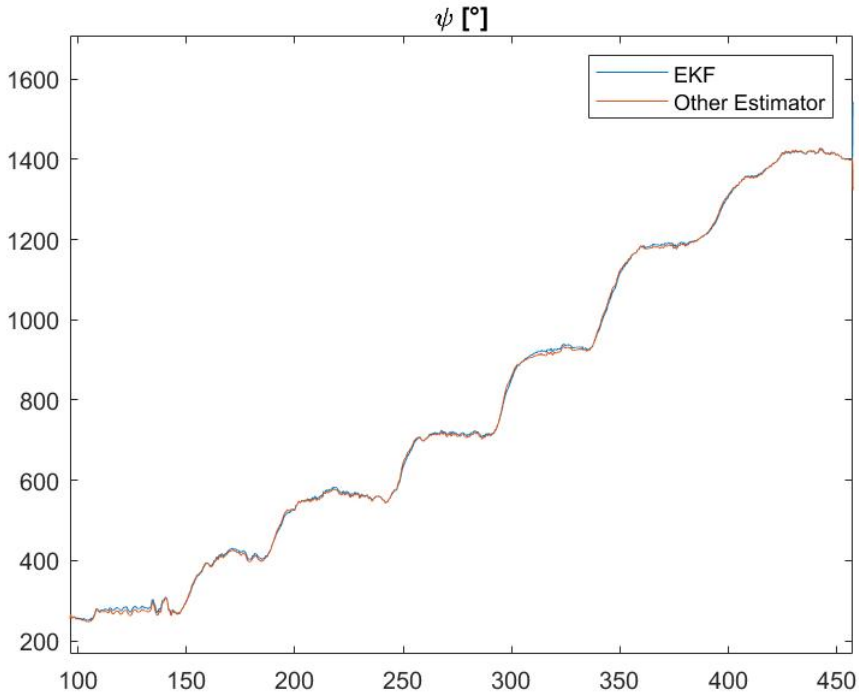

Figure 19: Yaw angle comparison as a function of time in seconds.

3.3.1 Real flight data

An advantage of sensor fusion, that is, combining sensor measurement, is that errors of one sensor can be corrected by the other. In the graph below it is possible to see that the EKF estimation in yellow has smoothed a GNSS perturbation in one of the vertical traces in figure 21.

Figures from 22 to 24 show the results for velocity also compared to the GNSS measurements. In order to validate this implementation, a Monte Carlo analysis could be realized creating a false signal as was suggested for the orientation filter in the previous section. However, in this project it was not realized.

The estimated acceleration bias for the test is plotted in Figure 25. It is possible to see estimations reach higher values during curves, which is when the gravity vector $\mathbf{g}^b$ has its values most perturbed by centripetal accelerations ($\mathbf{a}^{\mathbf{b}}_{\mathbf{ib}} \neq 0$)

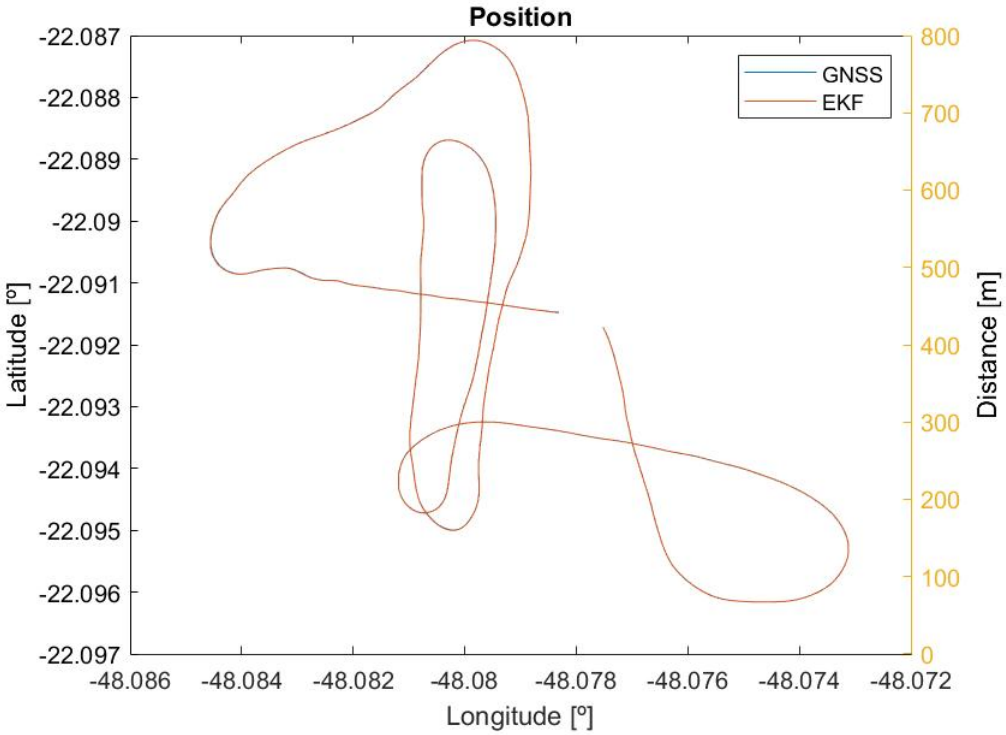

Figure 20: EKF position comparison with GPS

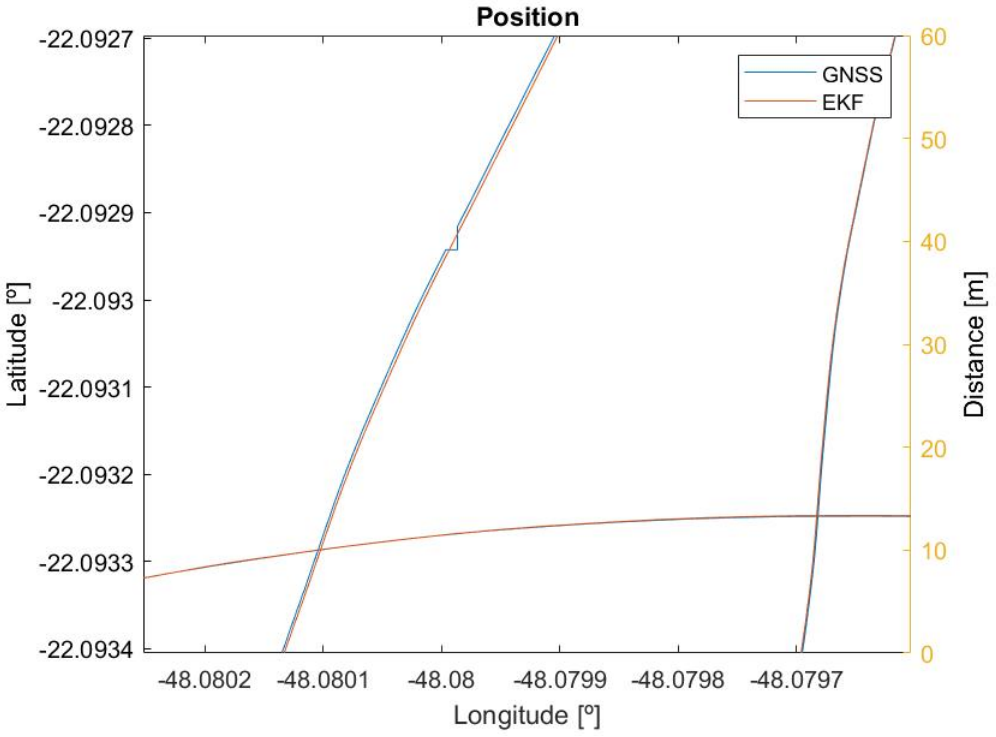

Figure 21: EKF position comparison with GNSS zoomed

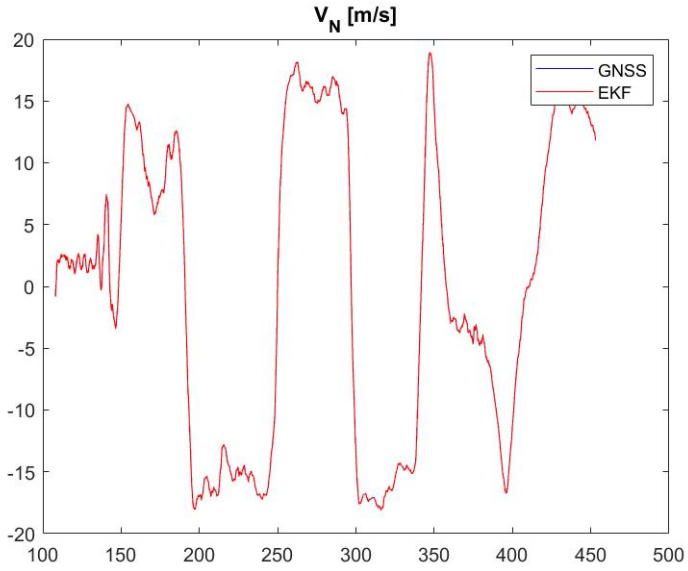

Figure 22: EKF north velocity comparison with GNSS as a function of time in seconds.

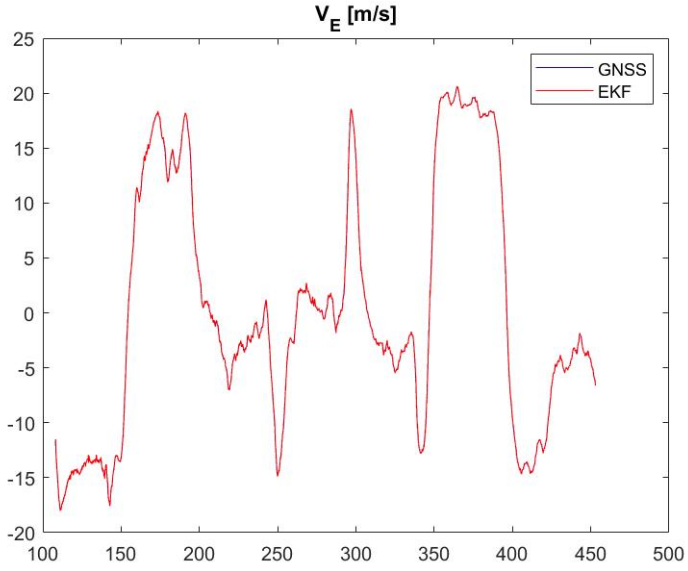

Figure 23: EKF east velocity comparison with GNSS as a function of time in seconds.

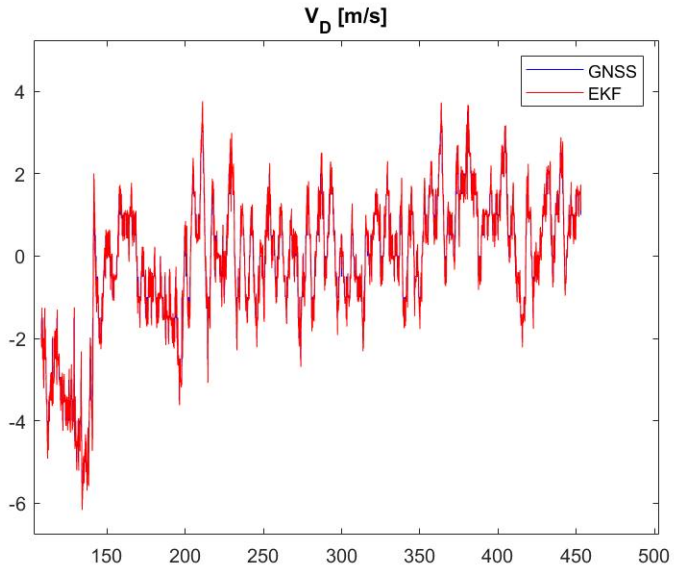

Figure 24: EKF down velocity comparison with GNSS as a function of time in seconds.

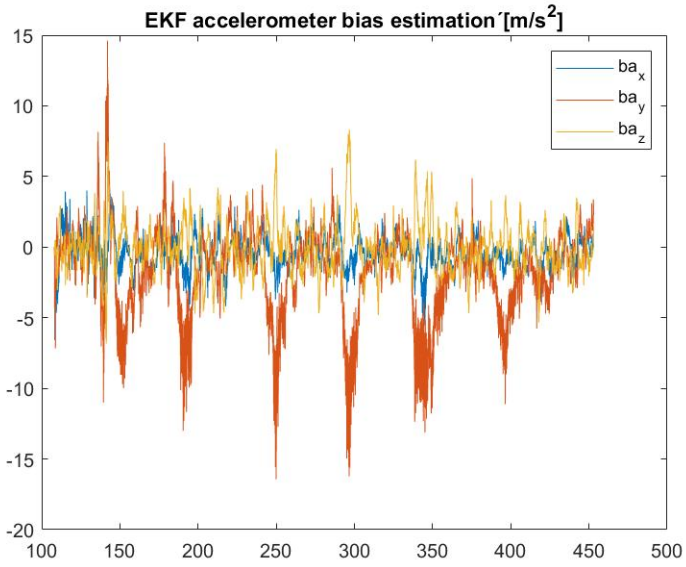

Figure 25: EKF accelerometer bias estimation as a function of time in seconds.

As a last test, the GNSS had its position and velocity measurement data corrupted by a greater white Gaussian noise at half of the trajectory. For this test, the amplitude of the Gaussian noise for latitude and longitude was set to 25 degrees, and the altitude white noise error to an amplitude of 10 m. Velocity was corrupted with a $25m/s$ amplitude white noise in all three axes.

As seen in figure 26, if this failure could be identified and the covariance matrix for the GNSS redefined, the EKF would not have diverged, although the errors would be large.

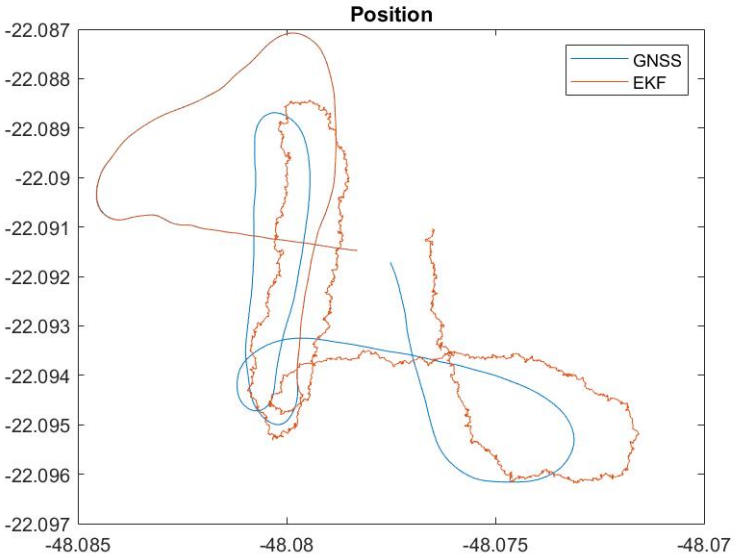

Figure 26: Simulation of GNSS hazard.

4 CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

4.1 Conclusion

This work consisted in developing estimators for the navigation states (orientation, position and velocity) by the use of Error State Kalman Filter and an Extended Kalman Filter. The filters were developed separately, however a solution in combining them was also discussed. The orientation filter used an Error State Kalman Filter, while the position and velocity filter used a regular Extended Kalman Filter.

Both filters were coded in MATLAB language and were analyzed using simulated data and flight data of an Xmobots RPA. The tuning of the estimators followed a coherent methodology, as each sensor used in the aircraft had its noises described and modeled in order to appropriately apply the Kalman filter algorithms.

The orientation filter showed to function properly with simulated data, as the standard deviation of the estimated time-series taken in 30 simulations showed to be inferior to the estimated variance. Moreover, a small mean time for convergence was found, as the filter attitude error converged to less than 0.2 degrees in attitude error before 6 seconds. When applied to real flight data, the estimator showed results in agreement to the estimation provided by the RPA's own estimator. Differences are attributed to imprecision in sensor covariance matrix characterization.

The position filter was compared to the GNSS measurements for means of comparison, showing it converged to the expected trajectory. Since the GNSS sensor model is not accurate, future improvements of the position filter may include a more complete GNSS model, contemplating the sensor's internal calculations instead of simply assuming it's noise as white and Gaussian.

Although other tests are required in order to validate the filters here presented, the satisfactory results motivate further efforts in the direction of ameliorating these estimators.

4.2 Future work

At the end of this project, a couple of other filter developments were studied. This section will then comment on these other projects and the future work that can be done to complete them.

4.2.1 Combining Position and Orientation

Ultimately, the orientation filter and the position filter can be combined in order to form a single ESKF navigation filter as proposed in (FARRELL, 2008). The states of this filter would then be $\hat{\mathbf{x}} = [\delta\mathbf{p}, \delta\mathbf{v}, \delta\theta_G, \delta\mathbf{x}_a, \delta\mathbf{x}_g]^T$, i.e. a 15-state filter.

One of the advantages of this filter is to estimate as well the errors of position and velocity, using the ESKF formulation. Moreover, it can also be used as a *truth model* as suggested in (FARRELL, 2008), since it estimates all states at once and the influences between error states can be properly modeled.

The model equations for this filter would then be

$$\begin{bmatrix} \dot{\delta\mathbf{p}} \\ \dot{\delta\mathbf{v}} \\ \dot{\delta\theta_G} \\ \dot{\delta\mathbf{x}_a} \\ \dot{\delta\mathbf{x}_g} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{pp} & \mathbf{A}_{pv} & \mathbf{A}_{p\theta} & \mathbf{O}_3 & \mathbf{O}_3 \\ \mathbf{A}_{vp} & \mathbf{A}_{vv} & \mathbf{A}_{v\theta} & -\hat{\mathbf{R}}_b^n & \mathbf{O}_3 \\ \mathbf{A}_{\theta p} & \mathbf{A}_{\theta v} & \mathbf{A}_{\theta\theta} & \mathbf{O}_3 & \hat{\mathbf{R}}_b^n \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{F}_a & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{F}_a & \mathbf{F}_g \end{bmatrix} \begin{bmatrix} \delta\mathbf{p} \\ \delta\mathbf{v} \\ \delta\theta_G \\ \delta\mathbf{x}_a \\ \delta\mathbf{x}_g \end{bmatrix} + \begin{bmatrix} \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 \\ -\hat{\mathbf{R}}_b^n & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \hat{\mathbf{R}}_b^n & \mathbf{O}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{I}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} w_a \\ w_g \\ w_{b_a} \\ w_{b_g} \end{bmatrix} \tag{4.1}$$

where $\mathbf{A}_{pp}$, $\mathbf{A}_{pv}$, $\mathbf{A}_{vp}$, $\mathbf{A}_{vv}$, $\mathbf{A}_{pv}$, $\mathbf{A}_{\theta\theta}$ were already presented in sections 2.4.2 and 2.5.1 and;

$$\mathbf{A}_{p\theta} = \mathbf{O}_3 \tag{4.2}$$

$$\mathbf{A}_{v\theta}, = [f_{ned}\times] \quad f_{ned} = \mathbf{R}_b^n(\mathbf{a}_{ib}^b - \mathbf{x}_a) \tag{4.3}$$

$$\mathbf{A}_{\theta p} = \begin{bmatrix} -\Omega_D & 0 & \frac{\bar{v}_e}{(R_N+\bar{h})^2} \\ 0 & 0 & \frac{-\bar{v}_n}{(R_M+\bar{h})^2} \\ \Omega_N + \frac{\bar{v}_e}{(R_N+\bar{h})\cos^2\bar{\phi}} & 0 & \frac{-\bar{v}_e \tan\bar{\phi}}{(R_N+\bar{h})^2} \end{bmatrix} \tag{4.4}$$

$$\mathbf{A}_{\theta v} = \begin{bmatrix} 0 & \frac{-1}{R_N+\bar{h}} & 0 \\ \frac{1}{R_M+\bar{h}} & 0 & 0 \\ 0 & \frac{\tan\bar{\phi}}{R_N+\bar{h}} & 0 \end{bmatrix} \tag{4.5}$$

with $\Omega_E = -\omega_{ie}\sin\bar{\phi}$ and $\Omega_D = \omega_{ie}\cos\bar{\phi}$.

Thus, in this filter, the predict phase would comprise propagation of this error state and the propagation of the nominal states of both position estimator ($\bar{\mathbf{v}}^n$ and $\bar{\mathbf{p}}$) and orientation estimator ($\bar{\mathbf{q}}^n$). And in the update phase, residues for position, velocity, tilt angle and biases would be calculated following the same procedures as described in the previous sections.

4.2.2 Hard Iron and Soft Iron Estimation

In section 2.3, it was mentioned that the magnetometer constant bias $\mathbf{x}_m$ is accounted for in calibration in order to avoid biased measurements from the magnetometer due to its own magnetic field. There is another type of parameter that is affected by this effect: the scale factor. The scale factor is responsible for correcting incongruences between the three magnetometer axes.

For instance, if all axes measure the magnetic field equally, a 3D plot of the time series data of $\mathbf{m}_k^b$ in an experiment, where the magnetometer rotates in various senses, would describe a sphere. However, if one or more axes measure with the field differently, the described curve will be an ellipsoid. In the latter case, three scale factors, one for each axis, must be applied to the sensor output in order to scale the magnetometer output. Like the bias, the scale factor does not vary much, but needs to be estimated from time to time.

Therefore magnetometer calibration comprises both bias and scale factor, since both are affected by the vehicles own magnetic field. An alternative approach for calibrating the magnetometer for this bias and scale factor is to estimate them in flight, that is to consider it as other states. To accomplish this, a new Kalman filter can be developed. This can be then a subject of another project which complements the work done here as it ameliorates magnetometer measurement precision and enhances yaw estimation. For this project two approaches were imagined. The estimation can be done separately from the main state estimation (as done in (GUO et al., 2008)) or be incorporated as other states in equation 4.1 (see (FARRELL, 2008)). The latter case may be undesirable in case of low computer power, which is critical aspect in embarked systems, but can be still an approach if a *truth model* is to be developed.

BIBLIOGRAPHY

ALLAN, D. W. Statistics of atomic frequency standards. **Proceedings of the IEEE**, IEEE, v. 54, n. 2, p. 221–230, 1966.

ANAC, A. N. de A. C. **Regras da ANAC para uso de drones entram em vigor**. 2017. <https://www.anac.gov.br/noticias/2017/regras-da-anac-para-uso-de-drones-entram-em-vigor>. [Online; accessed 23-October-2019].

ARDUPILOT. **Acknowledgements**. 2019. <http://ardupilot.org/dev/docs/common-acknowledgments.html>. [Online; accessed 23-October-2019].

_____. **Extended Kalman Filter (EKF)**. 2019. <http://ardupilot.org/copter/docs/common-apm-navigation-extended-kalman-filter-overview.html>. [Online; accessed 23-October-2019].

_____. **Partners**. 2019. <http://ardupilot.org/about/Partners>. [Online; accessed 23-October-2019].

BAPTISTE, P.; MARANGET, L. **Programmation et Algorithmique**. [S.l.]: Ecole Polytechnique, 2006.

BOARD, I. Ieee standard specification format guide and test procedure for single-axis interferometric fiber optic gyros. **IEEE Std**, p. 952–1997, 1998.

BORUP, K. T.; FOSSEN, T. I.; JOHANSEN, T. A. A nonlinear model-based wind velocity observer for unmanned aerial vehicles. **IFAC-PapersOnLine**, Elsevier, v. 49, n. 18, p. 276–283, 2016.

BROWN, R. G.; HWANG, P. Y. et al. **Introduction to random signals and applied Kalman filtering**. [S.l.]: Wiley New York, 1992. v. 3.

CAVALLO, A. et al. Experimental comparison of sensor fusion algorithms for attitude estimation. **IFAC Proceedings Volumes**, Elsevier, v. 47, n. 3, p. 7585–7591, 2014.

CRASSIDIS, J. L.; MARKLEY, F. L.; CHENG, Y. Survey of nonlinear attitude estimation methods. **Journal of guidance, control, and dynamics**, v. 30, n. 1, p. 12–28, 2007.

DECEA, A. de C. S. **RPA/drone em aeroporto é assunto sério para o DECEA, que demonstra preocupação com o impacto na navegação aérea**. 2018. <https://www.decea.gov.br/?i=midia-e-informacao&p=pg_noticia&materia=rpadrone-em-aeroporto-e-assunto-serio-para-o-decea-que-demonstra-preocupacao-com-o>-impacto-na-navegacao-aerea. [Online; accessed 23-November-2019].

DECEA, D. de Controle do E. A. **Voos de RPAS (drones). Entenda a nova legislação do DECEA!** 2015. <https://www.decea.gov.br/?i=midia-e-informacao&p=pg_noticia&materia=voos-de-rpas-drones-entenda-a-nova-legislacao-do-decea>. [Online; accessed 23-October-2019].

_____. **Drone**. 2019. <https://www.decea.gov.br/drone/>. [Online; accessed 23-October-2019].

EBEID, E.; SKRIVER, M.; JIN, J. A survey on open-source flight control platforms of unmanned aerial vehicle. In: IEEE. **2017 Euromicro Conference on Digital System Design (DSD)**. [S.l.], 2017. p. 396–402.

EISENBEISS, H. **UAV photogrammetry**. 2009. Tese (Doutorado) — ETH Zurich, 2009.

ETKIN, B.; REID, L. D. **Dynamics of flight**. [S.l.]: Wiley New York, 1959. v. 2.

EURE, K. W. et al. An application of uav attitude estimation using a low-cost inertial navigation system. 2013.

FAA, F. A. A. Unmanned aircraft systems fy 2019-2039. **FAA aerospace forecast: Fiscal Years 2019-2039**, 2019.

FARRELL, J. **Aided navigation: GPS with high rate sensors**. [S.l.]: McGraw-Hill, Inc., 2008.

GRAHAM, R. L. et al. Concrete mathematics: a foundation for computer science. **Computers in Physics**, AIP, v. 3, n. 5, p. 106–107, 1989.

GUO, P. et al. The soft iron and hard iron calibration method using extended kalman filter for attitude and heading reference system. In: IEEE. **2008 IEEE/ION Position, Location and Navigation Symposium**. [S.l.], 2008. p. 1167–1174.

GYROS, V. Ieee standards. 2004.

HOU, H. **Modeling inertial sensors errors using Allan variance**. [S.l.]: University of Calgary, Department of Geomatics Engineering, 2004.

IBAS. **Drone industry expected to grow** 25% **by 2019 and challenges will be debated at IBAS**. 2019. <https://internationalbrazilairshow.com.br/en/2019/07/24/setor-de-drones-deve-crescer-25-em-2019-e-os-desafios-serao-debatidos-no-ibas/>. [Online; accessed 21-October-2019].

INOUE, R. S. **Controle robusto descentralizado de movimentos coordenados de robôs heterogêneos**. 2011. Tese (Doutorado) — Universidade de São Paulo, 2011.

INOUE, R. S. et al. Markovian jump linear systems-based filtering for visual and gps aided inertial navigation system. In: IEEE. **2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2017. p. 4083–4089.

JULIER, S. J.; UHLMANN, J. K. New extension of the kalman filter to nonlinear systems. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Signal processing, sensor fusion, and target recognition VI**. [S.l.], 1997. v. 3068, p. 182–193.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Journal of basic Engineering**, American Society of Mechanical Engineers, v. 82, n. 1, p. 35–45, 1960.

LOTFI, B. Attitude estimation control of autonomous aerial vehicles. 06 2015.

MARKLEY, F. L. Attitude error representations for kalman filtering. **Journal of guidance, control, and dynamics**, v. 26, n. 2, p. 311–317, 2003.

MCGEE, L. A.; SCHMIDT, S. F. Discovery of the kalman filter as a practical tool for aerospace and industry. 1985.

OGATA, K.; YANG, Y. **Modern control engineering**. [S.l.]: Pearson Upper Saddle River, NJ, 2010. v. 17.

SARABANDI, S.; THOMAS, F. Accurate computation of quaternions from rotation matrices. In: SPRINGER. **International Symposium on Advances in Robot Kinematics**. [S.l.], 2018. p. 39–46.

SESAR, J. European drones outlook study unlocking the value for europe. **Siebert, JU, Nov**, 2016.

SOLA, J. Quaternion kinematics for the error-state kalman filter. **arXiv preprint arXiv:1711.02508**, 2017.

TRAWNY, N.; ROUMELIOTIS, S. I. Indirect kalman filter for 3d attitude estimation. **University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep**, v. 2, p. 2005, 2005.

WIKIPEDIA. **ECEF — Wikipedia, The Free Encyclopedia**. 2019. <http://en.wikipedia.org/w/index.php?title=ECEF&oldid=918493914>. [Online; accessed 21-October-2019].

XING, Z.; GEBRE-EGZIABHER, D. Modeling and bounding low cost inertial sensor errors. In: IEEE. **2008 IEEE/ION Position, Location and Navigation Symposium**. [S.l.], 2008. p. 1122–1132.

XMOBOTS. **Arator 5B**. 2019. <https://xmobots.com.br/arator-5b/>. [Online; accessed 23-October-2019].

_____. **Drone**. 2019. <https://xmobots.com.br/#drones>. [Online; accessed 23-October-2019].

Appendix

APPENDIX A – ALLAN VARIANCE CALCULATION

As described in (BOARD, 1998) and (HOU, 2004), if the instantaneous output of the inertial sensor $\mathbf{\Omega}(t)$ is obtained with a sampling frequency $1/T$, one can write its k^{th} time average for a period $\tau = NT$ as

$$\bar{\mathbf{\Omega}}_k = \frac{1}{\tau}\int_{t_k}^{t_k+\tau} \mathbf{\Omega}(t)dt, \tag{A.1}$$

and next time average is defined as

$$\bar{\mathbf{\Omega}}_{k+1} = \frac{1}{\tau}\int_{t_{k+1}}^{t_{k+1}+\tau} \mathbf{\Omega}(t)dt, \tag{A.2}$$

where $t_{k+1} = t_k + \tau$ (Figure 27). A new random variable ξ relatable to each τ can then be defined as $\xi_{k+1,k} = \bar{\mathbf{\Omega}}_{k+1}(\tau) - \bar{\mathbf{\Omega}}_k(\tau)$. The Allan Variance will then be the variance of this variable ξ for each partition τ of the data time-series (of size M):

$$\beta^2(\tau) = \frac{1}{2(M-2N)}\sum_{k=1}^{M-2N}(\bar{\mathbf{\Omega}}_{k+1}(\tau) - \bar{\mathbf{\Omega}}_k(\tau))^2. \tag{A.3}$$

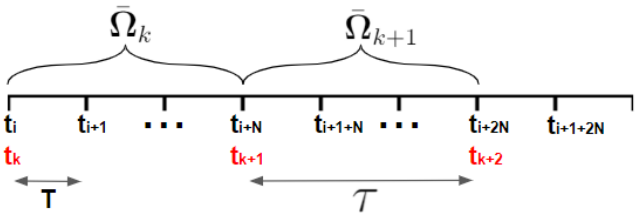

Figure 27: Time averaging of inertial sensor output

APPENDIX B – ROTATION ALGEBRA

B.1 Quaternion small angular pertubations

According to (SOLA, 2017), the unit quaternion can be used to represent a rotation ϕ through an axis with unit vector $\mathbf{u}$. This quaternion can then be written as:

$$\mathbf{q} = \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2)\vec{\mathbf{u}} \end{bmatrix} \tag{B.1}$$

For $\phi << 1$, a first order approximation of this equation is

$$\delta\mathbf{q} \approx \begin{bmatrix} 1 \\ (\phi/2)\vec{\mathbf{u}} \end{bmatrix} \tag{B.2}$$

B.2 Quaternion integration

The following derivation of the quaternion integration closed form was taken from (FARRELL, 2008).

Let

$$\dot{\mathbf{q}}(t) = \mathbf{Q}(t)\mathbf{q}(t) = \frac{1}{2}\mathbf{\Omega}(t)\mathbf{q}(t). \tag{B.3}$$

Defining

$$\Sigma(t) = e^{-\int_{t_1}^{t} \mathbf{Q(s)}ds}, \tag{B.4}$$

one can left-multiply both sides of B.3 by $\Sigma(t)$ and arrive at

$$\frac{d}{dt}\Big(\Sigma(t)\mathbf{q}(t)\Big) = 0 \tag{B.5}$$

Integrating both sides over the interval $[t_1, t_2]$ yields

$$e^{-\int_{t_1}^{t_2} \mathbf{Q(s)}ds}\mathbf{q}(t_2) = e^{-\int_{t_1}^{t_1} \mathbf{Q(s)}ds}\mathbf{q}(t_1) \tag{B.6}$$

and thus,

$$\mathbf{q}(t_2) = e^{\int_{t_1}^{t_2} \mathbf{Q(s)}ds}\mathbf{q}(t_1) \tag{B.7}$$

Now, define

$$\mathbf{W} = \begin{bmatrix} 0 & -\mathbf{w}^T \\ \mathbf{w} & [\mathbf{w}\times] \end{bmatrix} \tag{B.8}$$

where $\mathbf{w} = [W_1, W_2, W_3]^T$ is a constant vector with $W_i = \frac{1}{2}\int_{t_1}^{t} \omega_i(s)ds$, for $i = 1, 2, 3$. One may then rewrite B.7 as

$$\mathbf{q}(t_2) = e^{\mathbf{W}}\mathbf{q}(t_1) \tag{B.9}$$

Note $\mathbf{W}$ has the property

$$\mathbf{W}^2 = -||\mathbf{w}||^2\mathbf{I} \tag{B.10}$$

By the use of this identity, one can expand the $e^{\mathbf{W}}$ Taylor series and see that

$$e^{\mathbf{W}} = \left(\mathbf{I} + \frac{\mathbf{W}^2}{2!} + \frac{(\mathbf{W}^2)^2}{4!} + ...\right) + \mathbf{W}\left(\mathbf{I} + \frac{\mathbf{I} + \mathbf{W}^2}{3!} + \frac{(\mathbf{W}^2)^2}{5!} + ...\right) \tag{B.11}$$

$$= \cos(||\mathbf{w}||)\mathbf{I} + \frac{\sin(||\mathbf{w}||)}{||\mathbf{w}||}\mathbf{W} \tag{B.12}$$

and thus

$$\mathbf{q}(t_2) = \left(\cos(||\mathbf{w}||)\mathbf{I} + \frac{\sin(||\mathbf{w}||)}{||\mathbf{w}||}\mathbf{W}\right)\mathbf{q}(t_1) \tag{B.13}$$

According to (FARRELL, 2008), it is possible to show that the above expression is second order in T.

B.3 Rotation matrix equivalent of quaternion

The quaternion perturbed equations shown in 2.2.1 can have their equivalent forms in rotation matrices given in (SOLA, 2017). In general, its is possible to see that the order of the operations is kept the same.

$$\mathbf{q}_b^n = \bar{\mathbf{q}}_b^n \otimes \Delta\mathbf{q}^b \tag{B.14}$$

$$\mathbf{q}_b^n = \Delta\mathbf{q}^n \otimes \bar{\mathbf{q}}_b^n \tag{B.15}$$

is equivalent to

$$\mathbf{R}_b^n = \bar{\mathbf{R}}_b^n \Delta\mathbf{R}^b \tag{B.16}$$

$$\mathbf{R}_b^n = \Delta\mathbf{R}^n \bar{\mathbf{R}}_b^n \tag{B.17}$$

It can also be noted, from left-to-right, that all the above matrix compositions take a vector from body frame to navigational frame, as $\Delta\mathbf{R}^n$ and $\Delta\mathbf{R}^b$ can be read as $\Delta\mathbf{R}_b^b$ and $\Delta\mathbf{R}_n^n$ respectively.

As shown in (FARRELL, 2008), a quaternion may be converted to a rotation matrix by

$$\mathbf{R}_b^n = \mathbf{I} + 2q_0[\vec{\mathbf{q}}\times] + 2[\vec{\mathbf{q}}\times]^2. \tag{B.18}$$

From this relationship, one can see that for small quaternion pertubations $\Delta q = [1, \vec{\theta}/2]^T$, substituting Δq in 2.4 yields

$$\Delta\mathbf{R}_b^n = \mathbf{I} + [\vec{\theta}\times] \tag{B.19}$$

after neglecting second order terms.

According to (FARRELL, 2008), a rotation matrix can also be transformed to a quaternion by the equation:

$$\mathbf{q} = \begin{bmatrix} \frac{1}{2}\sqrt{1 + \mathbf{R}(1,1) + \mathbf{R}(2,2) + \mathbf{R}(3,3)} \\ \frac{\mathbf{R}(3,2) - \mathbf{R}(2,3)}{4q_0} \\ \frac{\mathbf{R}(1,3) - \mathbf{R}(3,1)}{4q_0} \\ \frac{\mathbf{R}(2,1) - \mathbf{R}(1,2)}{4q_0} \end{bmatrix} \tag{B.20}$$

When term $4q_0$ approaches zero, the above expression will be affected by numerical stability. Fortunately, there are alternatives to this relationship, which write the quaternion as a function of its other terms (not only q_0), permiting one to deal with this issue.

A detailed explanation of the unit quaternion space relationship to rotation matrix space is made in (SOLA, 2017).