DEGREE PROJECT IN SPACE TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2017

Attitude Navigation using a Sigma-Point Kalman Filter in an Error State Formulation

PERIKLIS-KONSTANTINOS DIAMANTIDIS

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING

Attitude Navigation using a Sigma-Point Kalman Filter in an Error State Formulation

Periklis-Konstantinos Diamantidis

TRITA 2017:107
ISSN 1653-5146

Department of Space and Plasma Physics
School of Electrical Engineering
KTH Royal Institute of Technology
SE–100 44 Stockholm
Sweden

Abstract

Kalman filtering is a well-established method for fusing sensor data in order to accurately estimate unknown variables. Recently, the unscented Kalman filter (UKF) has been used due to its ability to propagate the first and second moments of the probability distribution of an estimated state through a non-linear transformation. The design of a generic algorithm which implements this filter occupies the first part of this thesis. The generality and functionality of the filter were tested on a toy example and the results are within machine accuracy when compared to those of an equivalent C++ implementation.

Application of this filter to the attitude navigation problem becomes non-trivial when coupled to quaternions. Challenges present include the non-commutation of rotations and the dimensionality difference between quaternions and the degrees of freedom of the motion. The second part of this thesis deals with the formulation of the UKF in the quaternion space. This was achieved by implementing an error-state formulation of the process model, bounding estimation in the infinitesimal space and thus de-coupling rotations from non-commutation and bridging the dimensionality discrepancy of quaternions and their respective covariances.

The attitude navigation algorithm was then tested using an IMU and a magnetometer. Results show a bounded estimation error which settles to around 1 degree. A detailed look of the filter mechanization process was also presented showing expected behavior for estimation of the initial attitude with error tolerance of 1 mdeg. The structure and design of the proposed formulation allows for trivially incorporating other sensors in the estimation process and more intricate modelling of the stochastic processes present, potentially leading to greater estimation accuracy.

Keywords: unscented Kalman filtering, information filtering, quaternions, attitude navigation, gyroscope modelling, error state formulation, sensor fusion

Sammanfattning

Kalmanfiltering är en väletablerad metod för att sammanväga sensordata för att erhålla noggranna estimat av okända variabler. Nyligen har den typ av kalmanfilter som kallas unscented Kalman filter (UKF) ökat i popularitet på grund av dess förmåga att propagera de första och andra momenten för sannolikhetsfördelningen för ett estimerat tillstånd genom en ickelinjär transformation. Designen av en generisk algoritm som implementerar denna typ av filter upptar den första delen av denna avhandling. Generaliteten och funktionaliteten för detta filter testades på ett minimalt exempel och resultaten var identiska med de för en ekvivalent C++-implementation till den noggrannhet som tillåts av den finita maskinprecisionen.

Användandet av detta filter för attitydnavigering blir icke-trivialt när det används för kvaternioner. De utmaningar som uppstår inkluderar att rotationer inte kommuterar och att de finns en skillnad i dimensionalitet mellan kvaternioner och antalet frihetsgrader i rörelsen. Den andra delen av denna avhandling behandlar formuleringen av ett UKF för ett tillstånd som inkluderar en kvaternion. Detta gjordes genom att implementera en så kallad error state-formulering av processmodellen, vilken begränsar estimeringen till ett infinitesimalt tillstånd och därigenom undviker problemen med att kvaternionmultiplikation inte kommuterar och överbryggar skillnaden i dimensionalitet hos kvaternioner och deras motsvarande vinkelosäkerheter.

Attitydnavigeringen testades sedan med hjälp av en IMU och en magnetometer. Resultaten visade ett begränsat estimeringsfel som ställer in sig kring 1 grad. Strukturen och designen av den föreslagna formuleringen möjliggör på ett rättframt sätt tillägg av andra sensorer i estimeringsprocessen och mer detaljerad modellering av de stokastiska processerna, vilket potentiellt leder till högre estimeringnoggrannhet.

To my parents, Menelaos and Eleni

Acknowledgement

I want to thank Prof. Mykola Nickolay Ivchenko for his support as KTH supervisor during the duration of my thesis. I am also grateful to Dr. Anders Ericsson for arranging the project and welcoming me at Acreo, making sure my stay there was smooth and fruitful. I would like to thank other colleagues at Acreo namely Kaies Daoud and Dimitar Kolev who provided crucial support in the software and hardware front respectively of the experimental setup. Special thank goes to Dr. Pontus Johannisson for his daily commitment to guiding and teaching me which has proven to be instrumental both in the evolution of this project and of myself as an engineer.

Contents

Acronyms

EKF	extended Kalman filter
IMU	inertial measurement unit
LKF	linear Kalman filter
SRUIF	square-root unscented information filter
SRUKF	square-root uncented Kalman filter
UIF	unscented information filter
UKF	unscented Kalman filter
UT	unscented transformation

Chapter 1

Introduction

1.1 Background and Motivation

Attitude and position determination systems are an integral part of launch systems and sounding rockets, as they provide crucial information regarding their operations. Accurate position estimation results in prior knowledge of the impact area which in turn contributes to enhanced safety and efficient recovery process [1]. Position and attitude information is also provided to scientific payloads and can be the basis for triggering flight events while enhancing overall performance [1]. Sophisticated instrumentation for measuring acceleration and angular rates has been developed and a dead reckoning approach on integrating the obtained data could be a solution; albeit no matter how accurate the measurement instruments are, an inherent noise is always present and its cumulative effect over time results in a divergence between estimation and reality. An optimal approach to attitude determination is desired and the design of a system that implements it is of prime focus in the current thesis.

Design of such systems for space vehicles can be traced back to Wahba's problem [2], a proposed cost function of an optimization problem which determines spacecraft attitude. A general approach to optimal estimation was proposed by Kalman [3] and is known today as the Kalman filter (KF). The latter has evolved from a technique to be used in linearly modelled systems to efficiently incorporating non-linear dynamics in the form of *Extended* KF (EKF) and *Unscented* KF (UKF) [4]. UKF especially is of great interest since non-linear dynamics can be directly used via the *Unscented Transformation* (UT). At the same time, for attitude determination, a plethora of different mathematical tools is present to describe rotations, namely Euler angles, quaternions, axis-angle vectors and rotation matrices. Out of all those, quaternions is determined to be the most powerful tool for the numerical robustness and low computational cost it offers.

The aim of this thesis is to combine the non-linear modelling of the UKF with the mathematical soundness of the quaternions in the design of a filtering system which determines attitude with a very low error of less than 1 degree. Work on this topic, however, proves to be non trivial as mathematically sound concepts on their own, collide in logic when put together due to a host of reasons. While they will be presented in

great detail below, the zest of the challenge comes down to the following facts. The concept of the barycentric mean, a core concept of the UT, does not have a physical meaning in the quaternion space and covariances cannot be derived directly from a state vector containing quaternions. Sensors available for the project include an Inertial Measurement Unit (IMU) and a magnetometer.

1.2 Related Work

Previous work directly related to UKF is that of van der Merwe & Wan [5] where the basics of the UT are presented along with the algorithmic structure of the filter and its square-root variant. Van der Merwe [6] reiterates this work by applying the proposed filter in, among else, attitude determination of an Unmanned Air Vehicle (UAV), a sensor fusion of a Global Positioning System (GPS) and Inertial Navigation System (INS). This implementation, despite using quaternions, does not take into account the theoretical challenges stated above. To solve this issue, Kraft [7] proposes a modification of the algorithm presented by van der Merwe, while similar formulations to Kraft which contain ad-hoc quaternion normalizations are that of Cheon & Kim [8] and Challa et al. [9]. Crassidis [10] proposes a reformulation of the state vector by using Rodrigues parameters instead of quaternions, formulating an error-state approach. Crassidis' approach is based on the multiplicative error quaternion formulation from Shuster et al. [11]. This filter, originally an EKF is known as the *multiplicative* EKF (MEKF). Sola [12] presents in great detail the theoretical foundations of the error quaternion formulation and despite, like Shuster, not applying it to UKF but rather to EKF, he has greatly influenced the outcome of this thesis.

1.3 Thesis Outline

This thesis is organised as follows:

Chapter 2. MATLAB Implementation of an Unscented Kalman Filter
This chapter describes the design of a generic UKF and the results obtained when it is applied in a toy example. Further investigation into the numerical robustness is also presented.
Chapter 3. Quaternion Based Attitude Navigation Algorithm
This chapter contains a theoretical overview of the quaternions, the gyroscope and magnetometer modelling used and the error-state formulation of the attitude navigation problem.
Chapter 4. Results
This chapter contains results of the main filtering algorithm when used in the specific case researched in the present thesis.
Chapter 5. Conclusion
This chapter contains the conclusion of the current thesis along with further discussion on the evolution of the proposed method.

Chapter 2

MATLAB Implementation of an Unscented Kalman Filter

In systems design it is generally favorable to determine the object of interest as a state vector $\{x_k\}_{k=1}^n$. In the case of a 3DOF robot, for example, the state vector may contain its position in a two dimensional plane and its orientation angle. In general, by observing how the state vector evolves, one is able to determine its trajectory. The evolution of states can be approached in two ways. One can:

- Use the dynamic system that describes the physical process taking place, coupled with some initial conditions and user defined control signal input, to forward induct/predict state trajectory. One should note that imperfections like, e.g., in actuator operation cause deviations from the predicted path that the dynamic model cannot account for.
- Use sensors and get actual measurements either directly on the states or other quantities from which the states can be deduced. Noise present in sensors does not allow for perfect measurements.

The challenge posed is, therefore, to combine in an efficient and optimal manner dynamic model and measurements in order to determine with a great degree of certainty the true evolution of states.

2.1 Linear Kalman Filter

The filtering problem can be tackled by implementing a *linear* KF (LKF) [3], whose iterative process is shown in Fig. 2.1.

- $x_k = Ax_{k-1} + Bu_{k-1}$ is the linear system dynamics.
- $z_k = Hx_k$ is the output(measurement) linear relation to the state vector.
- P_k, Q and R are the system, process noise and measurement noise covariance matrices, respectively.

– K_k is the Kalman gain.

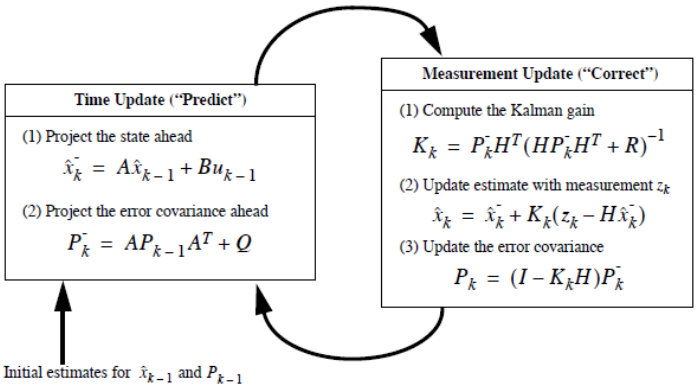


Figure 2.1: The Kalman filtering procedure (figure taken from [13]).

What one can deduce is that initial estimates on state (x_{k-1}) and covariance (P_{k-1}) can both be propagated, as shown in the Time Update part of Fig. 2.1. This produces predictions or *a priori* estimates on both quantities (x_k^-, P_k^-). These predictions are then corrected by incorporating measurements in a process shown in the Measurement Update part of Fig. 2.1. The result is that one obtains a mean value and a covariance matrix ($\hat{x_k}$, P_k^-), called the *a posteriori* estimates for the state vector at each time-step, obtaining thus with a fair degree of certainty a bounded region where the state resides. Fig. 2.2 shows how the mean value/ covariance update takes place in the state space. The validity of formulas derived for use in the Kalman filter has been reaffirmed through least squares [14] and bayesian [15] approaches. A core concept in deriving them is that of the *expected value*. The expected value of a random variable $\mathbf{x}$, E[$\mathbf{x}$], is the first moment of its distribution $f(\mathbf{x})$ or its *mean*. KF is an optimal estimator on condition that the expected value of the estimation error, i.e., difference between true and estimated state, is zero. This condition is known as unbiasedness and when it is met, LKF is the *Best Linear Unbiased Estimator* (BLUE) [16].

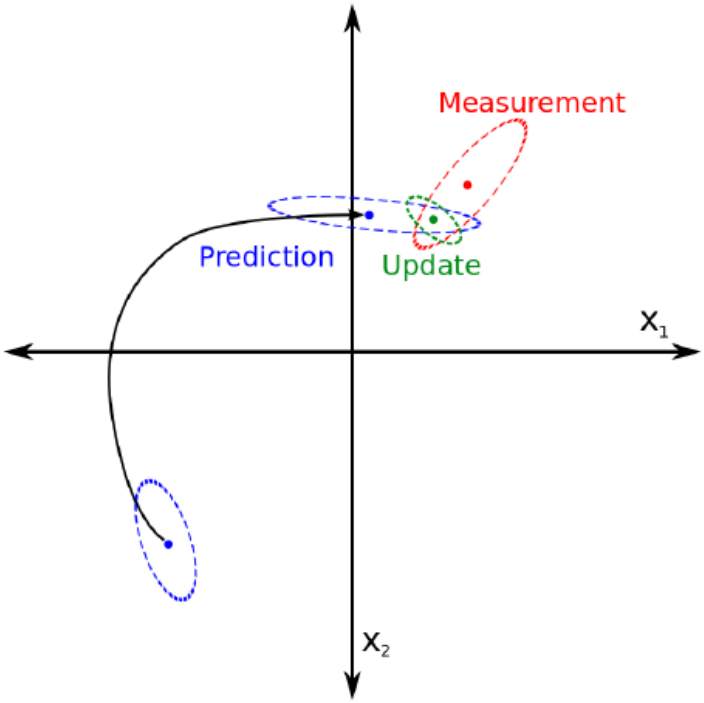


Figure 2.2: Covariance (oval) and mean value (dot) propagation/correction for a two dimensional state. The green dot and oval represent a region where the state vector will be present based on the estimation technique (figure taken from [15]).

2.2 Unscented Kalman Filter

While systems of linear dynamics can be approached efficiently by the LKF such is not the case for non-linear ones. The need arises, therefore, to modify the filtering procedure to account for non-linearities. One way to approach this, is to extend the LKF by using first order linearisations of non-linear system dynamics, around the estimated state vector, at each time-step in a process implemented in the EKF. Another option is to use directly the non-linear model, something made possible by the UKF. Instead of relying on first order linearisation for the construction and propagation of the covariance matrix, a minimal set of carefully selected sample points (known as sigma points) is used which undergo the Unscented Transformation. This process allows for calculating the mean value and covariance of these sample points through a true non-linear transformation. Assume a state vector x (of dimension L) has a mean $\bar{x}$ and a covariance matrix P_x. A

matrix $\mathcal{X}$ of $2L+1$ sigma points $\{\chi_i\}_{i=0}^{2L}$ can be obtained via the following.

$$\chi_0 = \bar{x}, \tag{2.1}$$

$$\chi_i = \bar{x} + \left(\sqrt{(L+\lambda)P_x}\right)_i \quad \text{where} \quad i = 1, \ldots, L, \tag{2.2}$$

$$\chi_i = \bar{x} - \left(\sqrt{(L+\lambda)P_x}\right)_i \quad \text{where} \quad i = L+1, \ldots, 2L, \tag{2.3}$$

where $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter. The constant α determining the spread of the sigma points around $\bar{x}$ is usually set to a small positive value ($10^{-4} \leq \alpha \leq 1$). The constant κ is a secondary scaling parameter usually set to 0 or $3-L$ [17]. $\left(\sqrt{(L+\lambda)P_x}\right)_i$ is the ith column of the matrix square root.

The sigma points are subsequently propagated via a non-linear function

$$\mathcal{Y}_i = f(\chi_i) \quad i = 0, ..., 2L, \tag{2.4}$$

to obtain the *a posterior* sigma points $\{\mathcal{Y}_i\}_{i=0}^{2L}$, which in turn can be used to find a new mean

$$\bar{\mathbf{y}} = \sum_{i=0}^{2L} \mathcal{W}_i^{(m)} \mathcal{Y}_i \tag{2.5}$$

and covariance

$$\mathbf{P}_\mathbf{y} = \sum_{i=0}^{2L} \mathcal{W}_i^{(c)} (\mathcal{Y}_i - \bar{\mathbf{y}})(\mathcal{Y}_i - \bar{\mathbf{y}})^{\mathrm{T}}, \tag{2.6}$$

with weights given by

$$\mathcal{W}_0^{(m)} = \lambda/(L+\lambda), \tag{2.7}$$

$$\mathcal{W}_0^{(c)} = \lambda/(L+\lambda) + (1-\alpha^2+\beta), \tag{2.8}$$

$$\mathcal{W}_i^{(c)} = \mathcal{W}_i^{(m)} = 1/2(L+\lambda), \tag{2.9}$$

where β is used to denote prior knowledge of the distribution of $\mathbf{x}$ ($\beta = 2$ is optimal for Gausian distribution). Fig. 2.3 shows this process as a flowchart with values being colour defined.

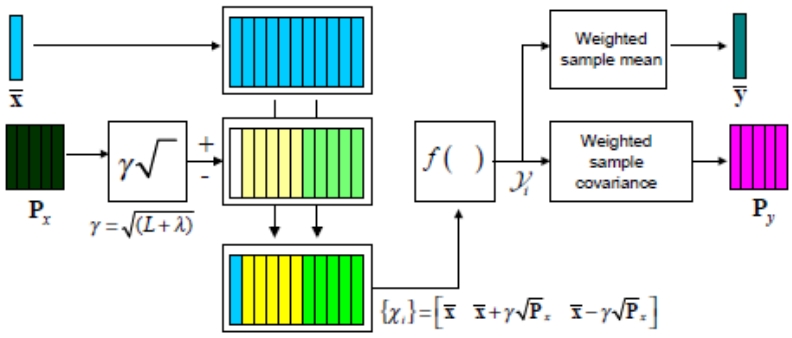


Figure 2.3: Block diagram of the UT (figure taken from [4]). Using the original mean (blue) and covariance (black), a vector of sigma points is generated (blue/yellow/green). This is transformed, through the UT, into the new mean (gray) and covariance (purple).

2.3 MATLAB Implementation

An algorithmic structure of the UKF is presented below (as laid out in Algorithm 2.1 of [5]) which is the base for the MATLAB implementation. Note that in (2.12) the matrix square root of the covariance is needed, $\sqrt{P_{k-1}}$, which is equivalent to the *lower triangular* matrix obtained through a Cholesky factorization. Parallel to the standard UKF, a variation on it exists in the form of *square-root* UKF (SRUKF). Instead of refactorizing the covariance at every time-step, a set of linear algebra tools are used to update the Cholesky factor instead. This allows for better numerical properties, namely ensuring the positive definiteness of the covariance matrix. The sequential steps need be taken to implement the UKF algorithm are as follows. It all starts with initialising the mean and covariance,

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \tag{2.10}$$

$$\mathbf{P}_0 = \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^{\mathrm{T}}]. \tag{2.11}$$

(2.12) – (2.15) show in sequence the generation of sigma points followed by the forward induction of them to get a new mean and covariance. This constitutes the *Prediction*

step of the UKF algorithm.

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{k-1} + \gamma\sqrt{\mathbf{P}_{k-1}}, \hat{\mathbf{x}}_{k-1} - \gamma\sqrt{\mathbf{P}_{k-1}}\right], \tag{2.12}$$

$$\boldsymbol{\mathcal{X}}^*_{k|k-1} = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}, \mathbf{u}_{k-1}], \tag{2.13}$$

$$\hat{\mathbf{x}}^-_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}, \tag{2.14}$$

$$\mathbf{P}^-_k = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}^-_k][\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}^-_k]^{\mathrm{T}} + \mathbf{R}^{\mathbf{v}}. \tag{2.15}$$

Measurement sigma points, their mean and covariance are computed mirroring the previous procedure in (2.16) – (2.19). Cross-covariance is then computed, $\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}$, followed by the Kalman gain, $\mathcal{K}_k$, to get to the corrected estimate on mean and covariance as shown in (2.20) – (2.23). This constitutes the *Correction* step of the UKF algorithm.

$$\boldsymbol{\mathcal{X}}_{k|k-1} = \left[\hat{\mathbf{x}}^-_k, \hat{\mathbf{x}}^-_k + \gamma\sqrt{\mathbf{P}^-_k}, \hat{\mathbf{x}}^-_k - \gamma\sqrt{\mathbf{P}^-_k}\right], \tag{2.16}$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}], \tag{2.17}$$

$$\hat{\mathbf{y}}^-_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \tag{2.18}$$

$$\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}^-_k][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}^-_k]^{\mathrm{T}} + \mathbf{R}^{\mathbf{n}}, \tag{2.19}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}^-_k][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}^-_k]^{\mathrm{T}}, \tag{2.20}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}(\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k})^{-1}, \tag{2.21}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}^-_k + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}^-_k), \tag{2.22}$$

$$\mathbf{P}_k = \mathbf{P}^-_k - \mathcal{K}_k\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}\mathcal{K}^{\mathrm{T}}_k. \tag{2.23}$$

The SRUKF algorithm is presented in the same manner. The first difference is that in this case, the Cholesky factor is initialised instead of the covariance,

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \tag{2.24}$$

$$\mathbf{S}_0 = \mathrm{chol}\left\{\mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^{\mathrm{T}}]\right\}. \tag{2.25}$$

The Prediction step is similar to the one described in the UKF algorithm. The weighted sampled covariance computation shown in (2.15), is substituted by (2.29) - (2.30).

$$\boldsymbol{\mathcal{X}}_{k-1} = [\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{k-1} + \gamma \mathbf{S}_{k-1}, \hat{\mathbf{x}}_{k-1} - \gamma \mathbf{S}_{k-1}], \tag{2.26}$$

$$\boldsymbol{\mathcal{X}}^*_{k|k-1} = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}, \mathbf{u}_{k-1}], \tag{2.27}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}, \tag{2.28}$$

$$\mathbf{S}_k^- = \mathrm{qr}\left\{\left[\sqrt{W_1^{(c)}}(\boldsymbol{\mathcal{X}}^*_{1:2L,k|k-1} - \hat{\mathbf{x}}_k^-), \sqrt{\mathbf{R}^{\mathbf{v}}}\right]\right\}, \tag{2.29}$$

$$\mathbf{S}_k^- = \mathrm{cholupdate}\left\{\mathbf{S}_k^-, \mathcal{X}^*_{0,k|k-1} - \hat{\mathbf{x}}_k^-, W_0^{(c)}\right\}. \tag{2.30}$$

The Correction step of the SRUKF algorithm follows. The main differences from the UKF algorithm are the substitution of the measurement covariance computation, (2.19), by (2.34) - (2.35), and that of the corrected covariance, (2.23), by (2.39) - (2.40).

$$\boldsymbol{\mathcal{X}}_{k|k-1} = \left[\hat{\mathbf{x}}_k^-, \hat{\mathbf{x}}_k^- + \gamma \mathbf{S}_k^-, \hat{\mathbf{x}}_k^- - \gamma \mathbf{S}_k^-\right], \tag{2.31}$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}], \tag{2.32}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \tag{2.33}$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \mathrm{qr}\left\{\left[\sqrt{W_1^{(c)}}(\boldsymbol{\mathcal{Y}}_{1:2L,k} - \hat{\mathbf{y}}_k), \sqrt{\mathbf{R}^{\mathbf{n}}}\right]\right\}, \tag{2.34}$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \mathrm{cholupdate}\left\{\mathbf{S}_{\tilde{\mathbf{y}}_k}, \mathcal{Y}_{0,k} - \hat{\mathbf{y}}_k, W_0^{(c)}\right\}, \tag{2.35}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^{\mathrm{T}}, \tag{2.36}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}(\mathbf{S}_{\tilde{\mathbf{y}}_k}^{\mathrm{T}})^{-1}(\mathbf{S}_{\tilde{\mathbf{y}}_k})^{-1}, \tag{2.37}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-), \tag{2.38}$$

$$\mathbf{U} = \mathcal{K}_k \mathbf{S}_{\tilde{\mathbf{y}}_k}, \tag{2.39}$$

$$\mathbf{S}_k = \mathrm{cholupdate}\{\mathbf{S}_k^-, \mathbf{U}, -1\}. \tag{2.40}$$

2.3.1 Standard to Square-Root Equivalence

At this point, it is helpful to directly derive the SRUKF out of the UKF. This not only helps to gain deeper insight in the underlying process but also to faithfully implement it in MATLAB, taking into account the special attributes that MATLAB functions possess. What should be kept in mind at all times is that for any positive definite matrix A, a Cholesky factor L (in lower triangular form) can be produced so that $A = LL^*$, where L^* is the conjugate transpose. If A is changed to $\tilde{A} = A + xx^*$, where x is a column vector, then instead of computing a new factor for $\tilde{A}$, one can update the existing Cholesky factor, L, to the new one, $\tilde{L}$, in a process called *Cholesky update*.

Looking back at both formulations, when updating the sigma points as shown in (2.12) and (2.26), a *lower triangular Cholesky factor* of the full covariance matrix should be input in both cases. Furthermore, the two processes are interchangeable up to (2.14) and (2.28) respectively. The first major difference comes when updating the a priori estimate of the covariance matrix, P_k^-, and its Cholesky factor, S_k^-. More specifically,

$$\begin{aligned} P_k^- &= \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{x}_k^-][\mathcal{X}_{i,k|k-1} - \hat{x}_k^-]^T + R^v = \\ &= \sum_{i=1}^{2L} \sqrt{W_i^{(c)}}(\mathcal{X}_{i,k|k-1} - \hat{x}_k^-)\sqrt{W_i^{(c)}}(\mathcal{X}_{i,k|k-1} - \hat{x}_k^-)^T + \sqrt{R^v}\sqrt{R^v}^T + \\ &+ W_0^{(c)}(\mathcal{X}_{0,k|k-1} - \hat{x}_k^-)(\mathcal{X}_{0,k|k-1} - \hat{x}_k^-)^T \end{aligned} \tag{2.41}$$

for

$$\mathcal{J}_i = \sqrt{W_i^{(c)}}(\mathcal{X}_{i,k|k-1} - \hat{x}_k^-) \tag{2.42}$$

becomes

$$P_k^- = \left[\mathcal{J}_1, \mathcal{J}_2 \cdots \mathcal{J}_{2L}, \sqrt{R^v}\right] \begin{bmatrix} \mathcal{J}_1{}^T \\ \mathcal{J}_2{}^T \\ \vdots \\ \mathcal{J}_{2L}{}^T \\ \sqrt{R^v}^T \end{bmatrix} + W_0^{(c)}(\mathcal{X}_{0,k|k-1} - \hat{x}_k^-)(\mathcal{X}_{0,k|k-1} - \hat{x}_k^-)^T. \tag{2.43}$$

By substituting

$$S = \left[\mathcal{J}_1, \mathcal{J}_2 \cdots \mathcal{J}_{2L}, \sqrt{R^v}\right] \tag{2.44}$$

and

$$q = \sqrt{W_0^{(c)}}(\mathcal{X}_{0,k|k-1} - \hat{x}_k^-) \tag{2.45}$$

(2.41) becomes

$$P_k^- = SS^T + qq^T. \tag{2.46}$$

It would make sense to propagate the Cholesky factor of the form of S, as defined in (2.44), using a Cholesky update due to the addition of q, as defined in (2.45). However, S is a non square, non triangular matrix and therefore cannot be validly considered a Cholesky factor. By using a *QR decomposition* on S, an orthogonal Q and a triangular R matrix are obtained such that

$$S^T = QR \tag{2.47}$$

and with the help of (2.47), (2.46) becomes

$$P_k^- = SS^T + qq^T = R^T Q^T QR + qq^T = R^T R + qq^T \tag{2.48}$$

It is evident now, as (2.48) indicates, that the update of the covariance matrix P_k^- can be converted into a Cholesky update of a triangular matrix R which is "posing"

as a Cholesky factor after decomposing S. By sustaining this process throughout the algorithm it is possible to alter the "update covariance → find Cholesky factor" of the UKF to "update directly the Cholesky factor" of the SRUKF.

2.3.2 MATLAB Implementation Considerations

S_k, S_k^- and $S_{\tilde{y}_k}$ are lower triangular matrices which come in contrast to the way `qr` and `cholupdate` functions operate in MATLAB.

- `qr` function accepts as input a square matrix or rectangular matrix where there are more rows than columns and returns an upper triangular matrix.
- `cholupdate` function accepts as input an upper triangular matrix and a vector of appropriate length and returns an upper triangular matrix.

In order to adhere to the conventions followed by the MATLAB functions and the expected result demanded by the algorithm, the following modifications to the UKF-SRUKF formulations take place.

- In step (20) the input to the QR decomposition is the transpose of the one shown, to fulfill the "more rows than columns" requirement.
- In step (20) the output is an upper triangular matrix and is kept as such in steps (21) and (29) in order for `cholupdate` to operate as stated above.
- A step is added after step (29), where the output of `cholupdate` is transposed before being input in step (17) as part of the iterative process. This is to fufill the requirement brought forth by the algorithm that the Cholesky factor must be a lower triangular matrix.
- The input and outputs of steps (24) and (25) are treated in the same fashion. By doing this, $S_{\tilde{y}_k}$ becomes upper triangular.
- To conform to the algorithm requirements, the output of step (25) is transposed before entering steps (27) and (28).

2.3.3 MATLAB Implementation Design Principles

The main concept behind the design of the model is *case independence*. Since the filtering procedure is developed with the purpose of being used in different application cases, two distinct ecosystems are defined, as shown by the vertical line in Fig. 2.4. One, the filter ecosystem, can be used without modification, for every application case and contains (i) a generic UKF, (ii) a generic SRUKF and (iii) the main "hub" function where all the other functions are called. The other can be viewed as the identity of each application case, comprising of (i) a file which initialises properties and values specific to the application, (ii) the model of the physical process/system examined, (iii) the model

of the control system present and (iv) the model of the measurement system(s) used. The two ecosystems, in spite of coming in contact during simulation, maintain their respective independence in the sense that changes in the underlying functions/scripts of one ecosystem pertain to its connections in the other and vice versa.

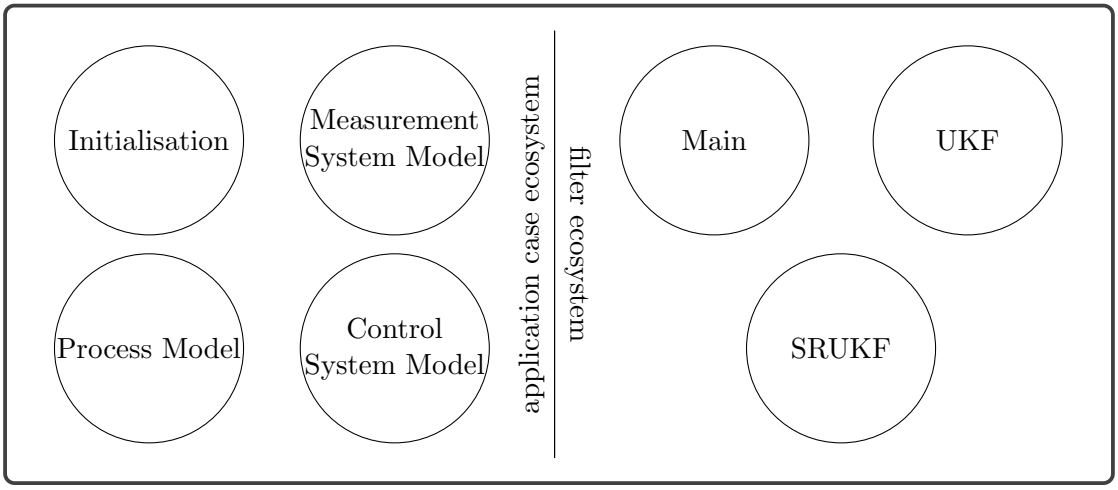


Figure 2.4: High-level MATLAB model design.

2.3.4 Application on a 3DOF Robot

After the filter ecosystem was implemented, an application case was used to test it in the form of a 3DOF robot. Its dynamic model is using a three dimensional state vector $\mathbf{x} \equiv (x, y, \phi)^{\mathrm{T}}$, which contains the position in x-axis, y-axis and the orientation angle. The robot is controlled through a two dimensional control vector $\mathbf{u} \equiv (v, \psi)^{\mathrm{T}}$, which contains the velocity and the orientation angle. Given that the system is iteration-based (no time-step given), one could assume that the velocity is the distance covered per unit iteration than per unit time. With this is mind, at iteration k, the following model is used

$$\phi_k = \phi_{k-1} + \psi_k, \tag{2.49}$$
$$x_k = x_{k-1} + \cos(\phi_k) \cdot v_k, \tag{2.50}$$
$$y_k = y_{k-1} + \sin(\phi_k) \cdot v_k. \tag{2.51}$$

Control signal generation is arbitrarily chosen to be

$$v_k = 1 + \sin\left(\frac{2\pi}{N}\right), \tag{2.52}$$
$$\psi_k = \sin\left(\frac{2\pi}{N}\right) \left\lfloor k - \frac{2}{(N/2)+1} \right\rfloor \tag{2.53}$$

where

$$\left\lfloor k - \frac{2}{(N/2)+1} \right\rfloor \tag{2.54}$$

is the integer part of

$$\left[k - \frac{2}{(N/2)+1} \right] \tag{2.55}$$

and N is the total number of iterations.

As far as measurement modelling, a three dimensional vector is used $\mathbf{y} \equiv (z, w, \theta)^{\mathrm{T}}$ where (z, w) are not the position in x and y-axis like the system vector above, but the distance from two landmarks $\{l_i\}_{i=1}^{2}$, while the third element is the orientation angle. For $p_k \equiv (x_k, y_k)$ the position at any given iteration k, the distance is given by

$$d_1 = (p_k - l_1), \tag{2.56}$$

$$d_2 = (p_k - l_2) \tag{2.57}$$

and therefore the measurement vector becomes

$$z_k = \sqrt{d_1 \cdot d_1}, \tag{2.58}$$

$$w_k = \sqrt{d_2 \cdot d_2}, \tag{2.59}$$

$$\theta_k = \phi_k. \tag{2.60}$$

The system and measurement models must be augmented with the addition of the actuator and measurement noise in order to complete the formulation.

2.3.4.1 Comparison to an Equivalent C++ Implementation

The MATLAB implementation was compared to an equivalent C++ one [18]. As the direct comparison of the obtained results between the two implementations is of great interest, a link between the two is established. As of now, any input data necessary for the replication of a test case along with the output data of the C++ code are saved in a .csv file and subsequently loaded in the MATLAB environment. Possible changes to the way the link operates might need to be made if the volume of data need be saved exceeds the functional limits of the current setup.

2.4 Results

As shown in the following figures, by implementing the model in the way analyzed above, almost identical results between the C++ and the MATLAB models are obtained (Fig. 2.5 - 2.7); the difference between the two is of the order of 10^{-5}. Process and noise covariance matrices are for unknown reasons set to identity for the C++ implementation. By constructing them as diagonal, with the elements being the squared standard deviations of the noise (variances), one can get a better response as can be clearly seen in Fig. 2.8 and Fig. 2.9.

Almost identical are the results between UKF and SRUKF, validating their interchangeability but for better numerical properties for the latter (Fig. 2.10 and Fig. 2.11). Note that the difference is on the order of 10^{-14}. Computationally they are not so different either. After 2000 iterations of the code in MATLAB, the mean difference between them was 1.04 ms or almost 5% of the total time one iteration of the code takes to complete.

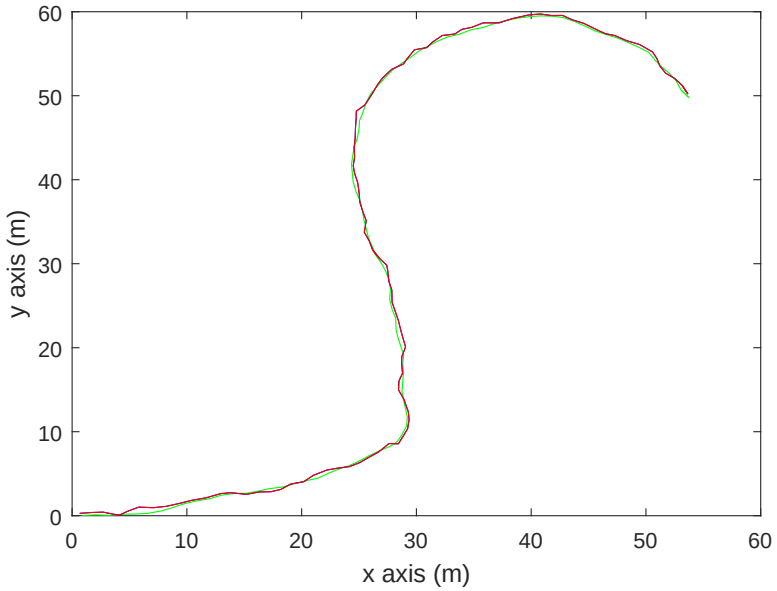


Figure 2.5: The real trajectory of the robot (green curve) and how it is estimated by the MATLAB implemented UKF (blue curve) and its C++ equivalent (red curve).

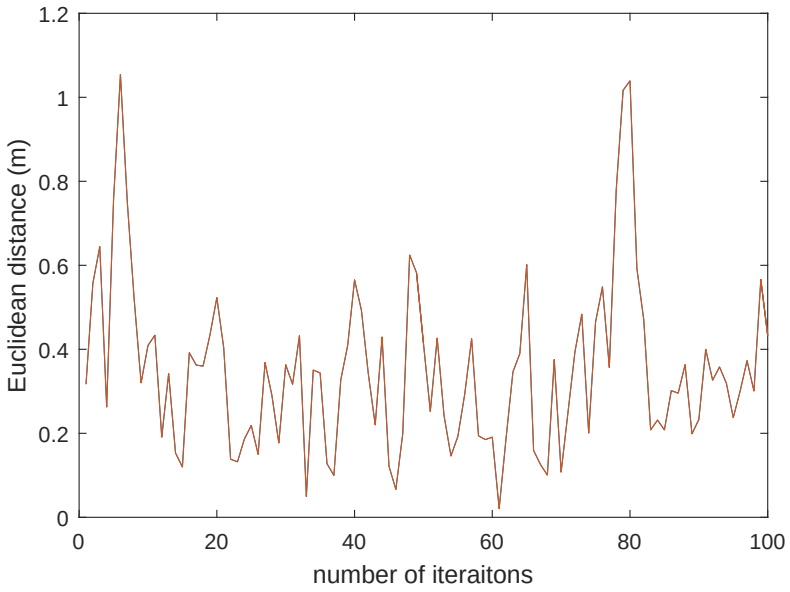


Figure 2.6: Estimation error (Euclidean distance) of the MATLAB UKF (blue curve) and the C++ UKF (red curve).

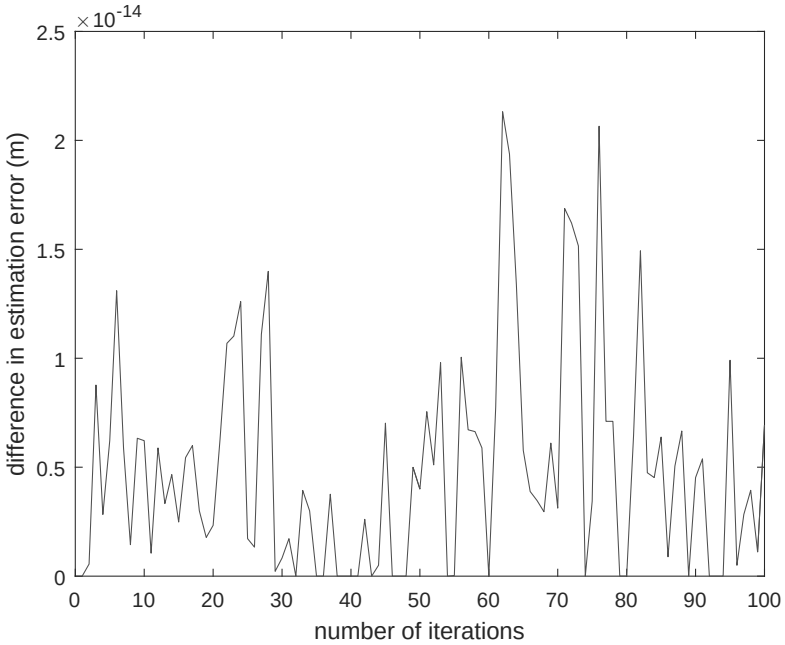


Figure 2.7: Estimation error difference between MATLAB UKF and C++ UKF.

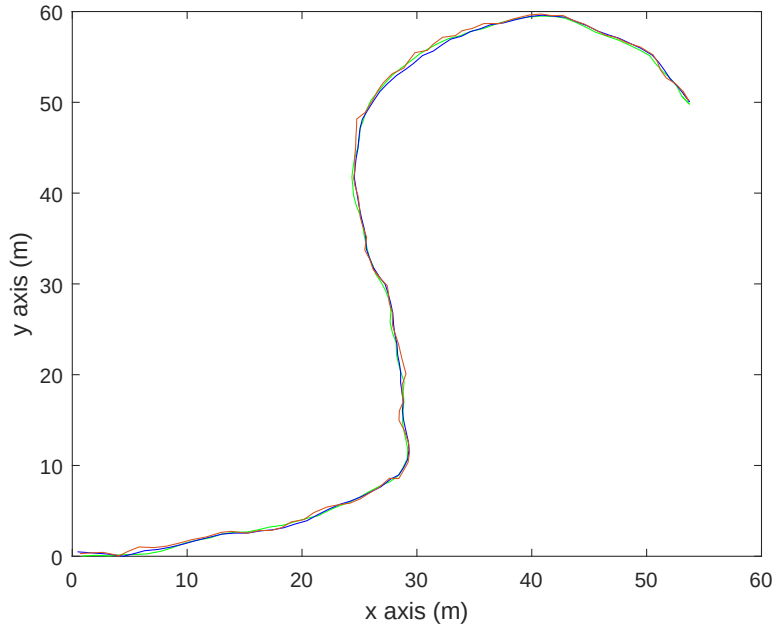


Figure 2.8: The real trajectory of the robot (green curve) and how it is estimated by UKF with calibrated (blue curve) and uncalibrated (red curve) covariances.

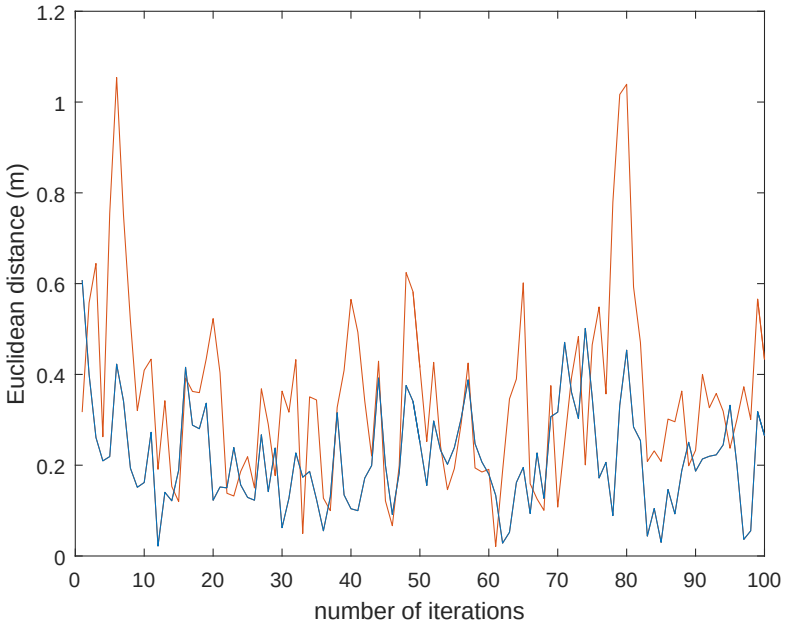


Figure 2.9: Estimation error (Euclidean distance) between UKF with calibrated (blue curve) and uncalibrated (red curve) covariances.

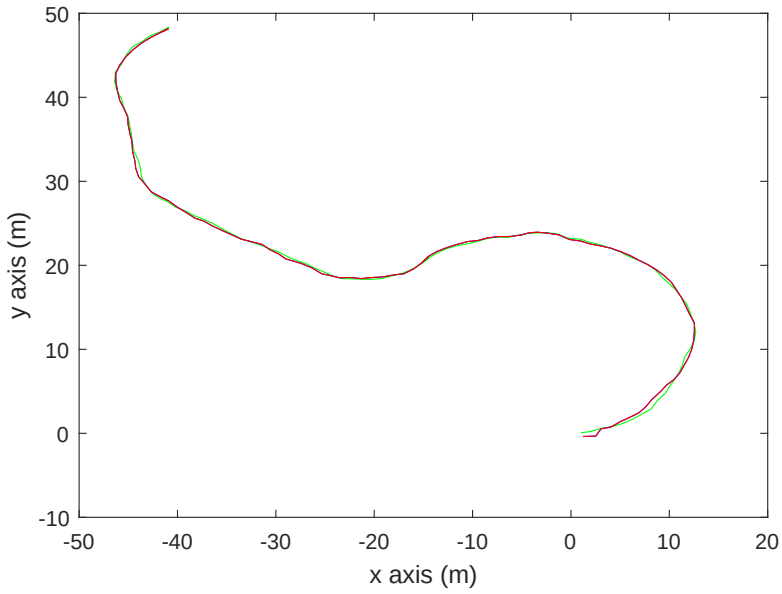


Figure 2.10: The real trajectory of the robot (green curve) and how it is estimated by UKF (blue curve) and SRUKF (red curve).

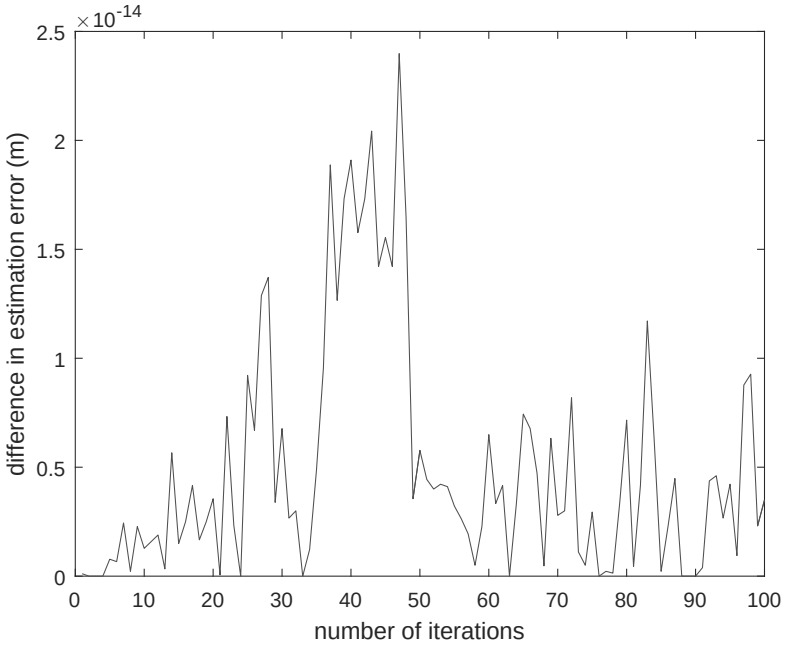


Figure 2.11: Estimation error difference between UKF and SRUKF.

2.5 Unscented Information Filter

Work has been done thus far in implementing a UKF and an SRUKF according to the algorithms proposed by van der Merwe. However, robustness of the numerical methods has always been an issue.

- Updating the covariance matrix in the UKF does not guarantee its positive definiteness. This is of utmost importance since it is a necessary condition for the Cholesky factor to exist.
- For a positive definite matrix A and a column vector x, Cholesky update, as described in chapter 2.2.1, can be performed as long as A is changed to $\tilde{A} = A + xx^*$ (rank 1 update). If $\tilde{A} = A - xx^*$(downdate), as is the case in (2.40) for $A \equiv \mathbf{S}_k^-$ and $x \equiv \mathbf{U}$, then $\tilde{A} \equiv \mathbf{S}_k$ may not be positive definite and the process will fail.

A solution to both issues may be the formulation of the dual problems to UKF and SRUKF known as *Unscented Information filter* (UIF) and *square-root* UIF(SRUIF).

2.5.1 Formulation

In this case, instead of the covariance matrix and the state vector, what is being updated through the prediction/correction process is what is known as the *information matrix*

$$\mathbf{Z}_k = \mathbf{P}_k^{-1} \tag{2.61}$$

and *information vector*

$$\mathbf{z}_k = \mathbf{P}_k^{-1}\mathbf{x}_k = \mathbf{Z}_k\mathbf{x}_k. \tag{2.62}$$

Basically, there is a duality in the relationship, since the information matrix is the direct inverse. What makes this method interesting are a couple of beneficial properties.

- It can be proven [19] that the square root formulation consists only of rank 1 Cholesky updates thus eliminating the danger of a possible downdating leading to negative eigenvalues.
- What must be done to achieve great accuracy in the Kalman procedure is to pursue a modular design, where each time before the input of a sensor reading the sigma points and observation points must be recalculated, based on the previous best estimate. In information filtering, a very good approximation of it can be achieved by only calculating the sigma points and observation points in the beginning, and augmenting them with a linear combination of the local sensor contribution (see below), potentially lowering the computational cost.

The algorithms in use are the following, as proposed in [19]. The $\mathbf{\Phi_k}$ and $\phi_\mathbf{k}$ defined below, are information contribution terms [19]. At first the UIF algorithm is presented in its sequential steps, starting from initialisation,

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \tag{2.63}$$

$$\mathbf{P}_0 = \mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^{\mathrm{T}}]. \tag{2.64}$$

The Prediction steps follows with the insertion of the information matrix and vector.

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{k-1} + \gamma\sqrt{\mathbf{P}_{k-1}}, \hat{\mathbf{x}}_{k-1} - \gamma\sqrt{\mathbf{P}_{k-1}}\right], \tag{2.65}$$

$$\boldsymbol{\mathcal{X}}^*_{k|k-1} = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}, \mathbf{u}_{k-1}], \tag{2.66}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}, \tag{2.67}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-]^{\mathrm{T}} + \mathbf{R}^{\mathbf{v}}, \tag{2.68}$$

$$\mathbf{Z}_k^- = (\mathbf{P}_k^-)^{-1}, \tag{2.69}$$

$$\mathbf{z}_k^- = \mathbf{Z}_k^- \hat{\mathbf{x}}_k^-. \tag{2.70}$$

Finally the Correction step is shown with the update procedures of the quantities that are now being tracked.

$$\boldsymbol{\mathcal{X}}_{k|k-1} = \left[\hat{\mathbf{x}}_k^-, \hat{\mathbf{x}}_k^- + \gamma\sqrt{\mathbf{P}_k^-}, \hat{\mathbf{x}}_k^- - \gamma\sqrt{\mathbf{P}_k^-}\right], \tag{2.71}$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1},] \tag{2.72}$$

$$\hat{\mathbf{y}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \tag{2.73}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^{\mathrm{T}}, \tag{2.74}$$

$$\phi_{\mathbf{k}} = \mathbf{Z}_k^- \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} (\mathbf{R}^{\mathbf{n}})^{-1} [(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) + \mathbf{P}^{\mathrm{T}}_{\mathbf{x}_k\mathbf{y}_k} (\mathbf{Z}_k^-)^{\mathrm{T}} \hat{\mathbf{x}}_k^-], \tag{2.75}$$

$$\boldsymbol{\Phi}_{\mathbf{k}} = \mathbf{Z}_k^- \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} (\mathbf{R}^{\mathbf{n}})^{-1} \mathbf{P}^{\mathrm{T}}_{\mathbf{x}_k\mathbf{y}_k} (\mathbf{Z}_k^-)^{\mathrm{T}}, \tag{2.76}$$

$$\mathbf{z}_k = \mathbf{z}_k^- + \phi_{\mathbf{k}}, \tag{2.77}$$

$$\mathbf{Z}_k = \mathbf{Z}_k^- + \boldsymbol{\Phi}_{\mathbf{k}}. \tag{2.78}$$

The SRUIF algorithm follows in the same manner. Initialisation begins with the mean and the Cholesky factor,

$$\hat{\mathbf{x}}_0 = \mathbb{E}[\mathbf{x}_0], \tag{2.79}$$

$$\mathbf{S}_0 = \mathrm{chol}\left\{\mathbb{E}[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^{\mathrm{T}}]\right\}. \tag{2.80}$$

The Prediction[1] step follows, with all the introduced changes.

$$\boldsymbol{\mathcal{X}}_{k-1} = [\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{k-1} + \gamma \mathbf{S}_{k-1}, \hat{\mathbf{x}}_{k-1} - \gamma \mathbf{S}_{k-1}], \tag{2.81}$$

$$\boldsymbol{\mathcal{X}}^*_{k|k-1} = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}, \mathbf{u}_{k-1}], \tag{2.82}$$

$$\hat{\mathbf{x}}^-_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}, \tag{2.83}$$

$$\mathbf{S}^-_k = \mathrm{qr}\left\{\left[\sqrt{W_1^{(c)}}(\boldsymbol{\mathcal{X}}^*_{1:2L,k|k-1} - \hat{\mathbf{x}}^-_k), \sqrt{\mathbf{R}^{\mathbf{v}}}\right]\right\}, \tag{2.84}$$

$$\mathbf{S}^-_k = \mathrm{cholupdate}\left\{\mathbf{S}^-_k, \mathcal{X}^*_{0,k|k-1} - \hat{\mathbf{x}}^-_k, W_0^{(c)}\right\}, \tag{2.85}$$

$$\mathbf{z}^-_k = (\mathbf{S}^-_k)^{\mathrm{T}} \setminus (\mathbf{S}^-_k \setminus \hat{\mathbf{x}}^-_k), \tag{2.86}$$

$$\mathbf{S}^-_{z_k} = \mathrm{qr}\{\mathbf{S}^-_k \setminus I\}. \tag{2.87}$$

Finally, the Correction step takes place providing the corrected estimates for the information vector and the Cholesky factor of the information matrix.

$$\boldsymbol{\mathcal{X}}_{k|k-1} = \left[\hat{\mathbf{x}}^-_k, \hat{\mathbf{x}}^-_k + \gamma \mathbf{S}^-_k, \hat{\mathbf{x}}^-_k - \gamma \mathbf{S}^-_k\right], \tag{2.88}$$

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}], \tag{2.89}$$

$$\hat{\mathbf{y}}^-_k = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Y}_{i,k|k-1}, \tag{2.90}$$

$$\mathbf{S}_{\tilde{\mathbf{y}}_k} = \mathrm{qr}\left\{\left[\sqrt{W_1^{(c)}}(\boldsymbol{\mathcal{Y}}_{1:2L,k} - \hat{\mathbf{y}}_k), \sqrt{\mathbf{R}^{\mathbf{n}}}\right]\right\}, \tag{2.91}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}^-_k][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}^-_k]^{\mathrm{T}}, \tag{2.92}$$

$$\mathbf{U} = (\mathbf{S}^-_k)^{\mathrm{T}} \setminus (\mathbf{S}^-_k \setminus \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k})/\mathbf{S}_n, \tag{2.93}$$

$$\mathbf{z}_k = \mathbf{z}^-_k + \mathbf{U}/\mathbf{S}_n^{\mathrm{T}}[\mathbf{y}_k - \hat{\mathbf{y}}^-_k + \mathbf{P}^{\mathrm{T}}_{\mathbf{x}_k\mathbf{y}_k}((\mathbf{S}^-_k)^{\mathrm{T}} \setminus (\mathbf{S}^-_k \setminus \hat{\mathbf{x}}^-_k))], \tag{2.94}$$

$$\mathbf{S}_{z_k} = \mathrm{cholupdate}\{\mathbf{S}^-_{z_k}, \mathbf{U}, +1\}. \tag{2.95}$$

Proof of the UIF can be found in [21] and the SRUIF can be derived directly in a straightforward way. For multiple sensors instead of calculating the state and covariance at each subsequent observation, one can calculate only the *local* information matrix/vector contribution of each sensor and at the end of the time-step add them all together. For the UIF it means computing only the $\{\phi_k, \Phi_k\}$ pair, for each sensor and at the end of the

[1] '\' backlash is MATLAB notation referring to *efficient least squares* [20].

time step, do the final update

$$\begin{aligned}
\mathbf{z}_k &= \mathbf{z}_k^- + \sum_{k=1}^{N} \phi_k, \\
\mathbf{Z}_k &= \mathbf{Z}_k^- + \sum_{k=1}^{N} \Phi_k, \\
\mathbf{P}_k &= {\mathbf{Z}_\mathbf{k}}^{-1}, \\
\mathbf{x}_k &= \mathbf{P}_k \mathbf{z}_k.
\end{aligned}$$

For the SRUIF it means computing only the $\{\phi_k, \mathbf{S}_{i,\Phi_k}\}$ pair, where $\mathbf{S}_{i,\Phi_k} = \mathbf{U}$ of i^{th} sensor and at the end of the time-step, do the final update

$$\begin{aligned}
\mathbf{z}_k &= \mathbf{z}_k^- + \sum_{k=1}^{N} \phi_k, \\
\mathbf{S}_{z_k} &= \text{cholupdate}\{\mathbf{S}_{z_k}^-, [\mathbf{S}_{\mathbf{1},\mathbf{\Phi_k}} \cdots \mathbf{S}_{\mathbf{N},\mathbf{\Phi_k}}], +1\}, \\
\mathbf{S}_k &= \text{qr}\{(\mathbf{S}_{z_k})^{\mathrm{T}} \setminus I\}, \\
\mathbf{x}_k &= (\mathbf{S}_k)^{\mathrm{T}} \mathbf{S}_k \mathbf{z}_k.
\end{aligned}$$

This has the potential to speed up the process without losing in accuracy as it will be shown later.

2.5.2 Results

The results obtained by the UIF using sensor fusion as performed in the C++ equivalent case are presented in Figs. 2.12 – 2.14. On the other hand, Figs. 2.15 – 2.17 represent the response for the same test case but by performing the sequential update as proposed by [19] and described above. One can see that there is comparable accuracy. Initial runs show a speed up of the procedure by almost 5% but with room for improvement for further algorithmic refinement.

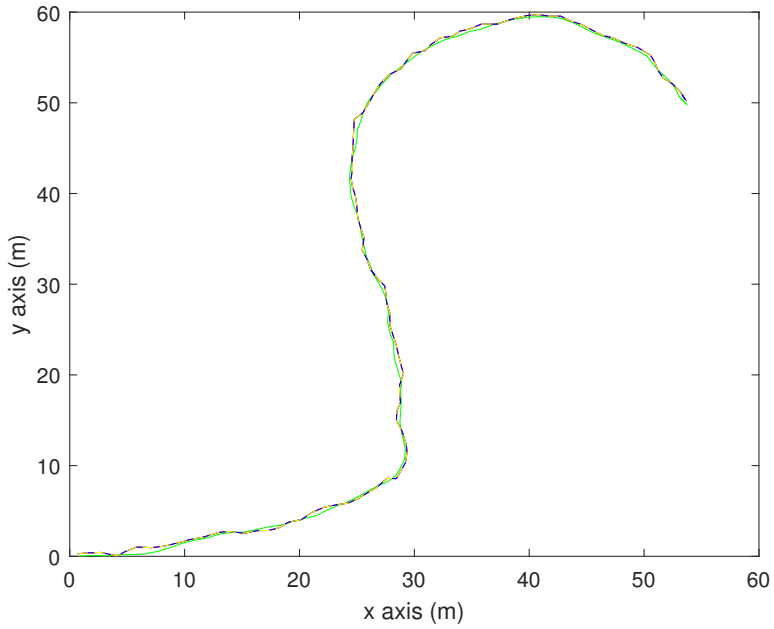


Figure 2.12: The real trajectory of the robot (green curve) and how it is estimated by UIF (blue curve) and SRUIF (red curve) and the C++ UKF (yellow dotted curve).

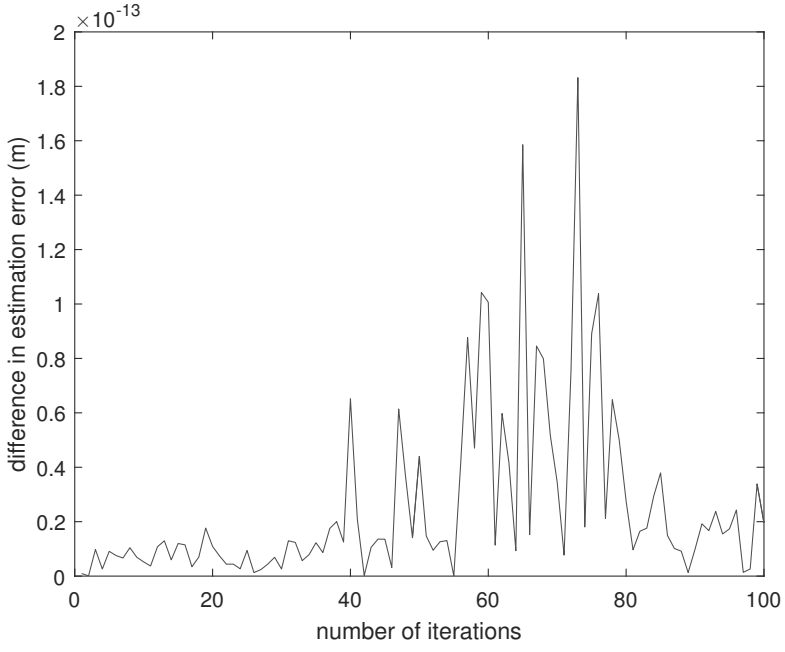


Figure 2.13: Estimation error difference between UIF and SRUIF.

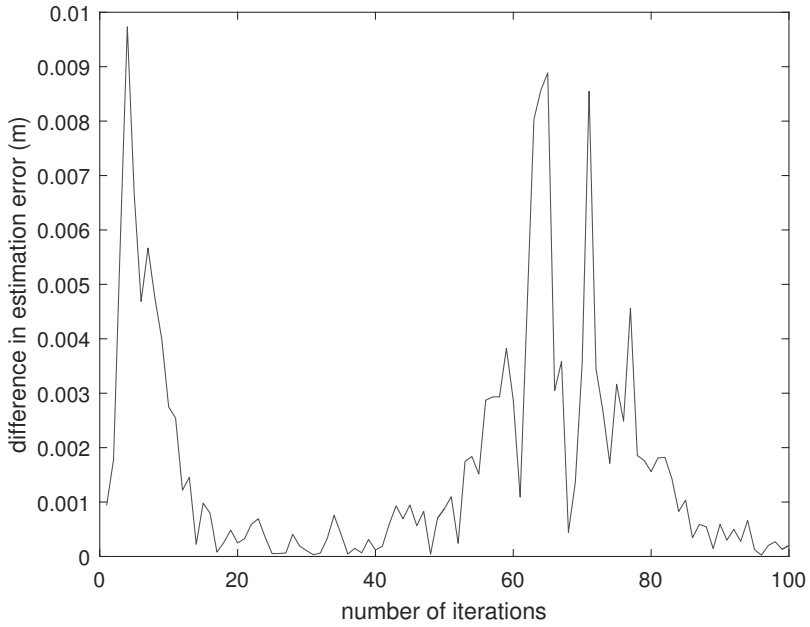


Figure 2.14: Estimation error difference between UIF and C++ UKF.

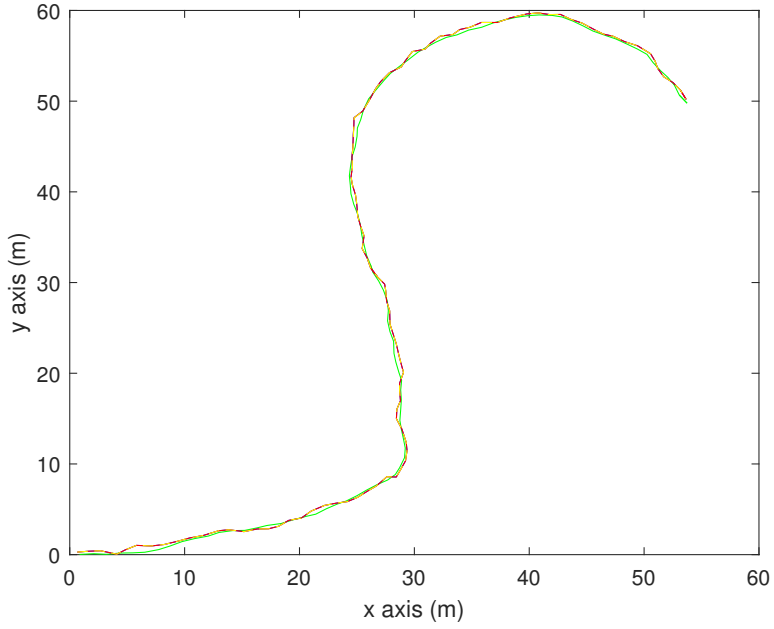


Figure 2.15: The real trajectory of the robot (green curve) and how it is estimated by UIF (blue curve) and SRUIF (red curve) and the C++ UKF (yellow dotted curve).

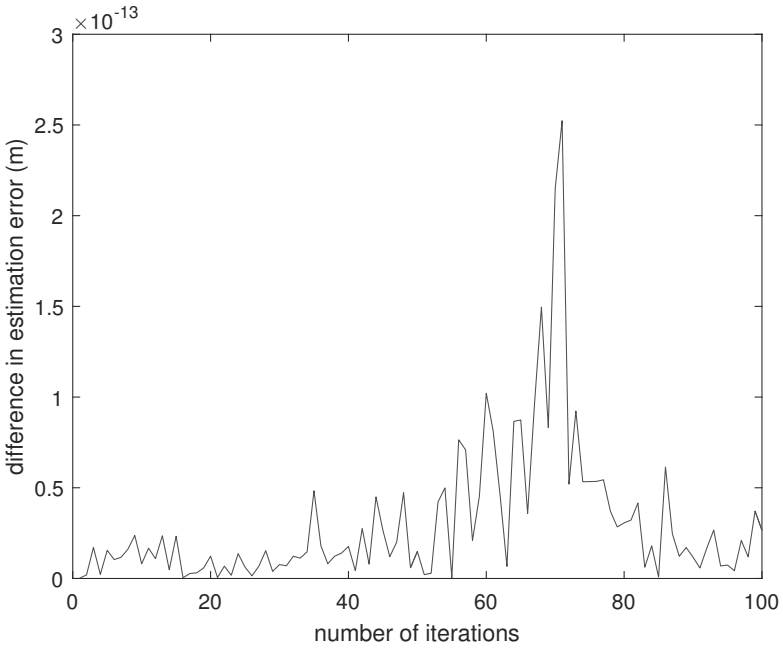


Figure 2.16: Estimation error difference between UIF and SRUIF.

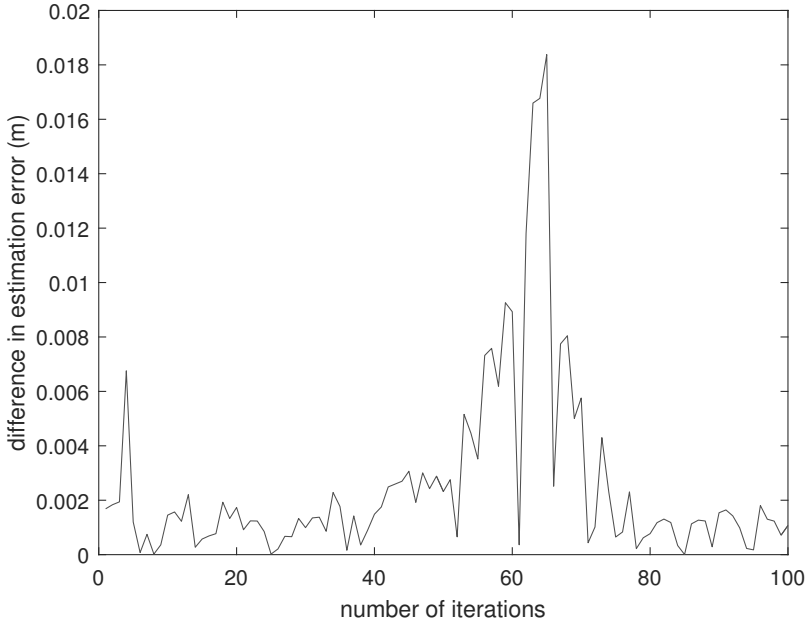


Figure 2.17: Estimation error difference between UIF and C++ UKF.

Chapter 3

Quaternion-Based Attitude Navigation Algorithm

3.1 Orientation Basics

The goal of an INS is to determine the position and orientation of an object. While the scope of the current study only revolves around the second one, a comparison between the two is deemed useful. This is because orientation is a non-trivial subject to grasp and looking at it in parallel to position provides intuition and insight into its inner workings.

3.1.1 Position Determination

In order to determine position in a three dimensional space, a coordinate system is established. It consists of orthonormal axes which enable one to specify uniquely each point through the use of a vector (a, b, c). This is especially true in the type of motions examined below where, as far as position is concerned, the object can be considered to be perfectly stiff, leading it to the possibility to treat it as a point object. This representation of position in space, while fundamental, has some properties which often go unmentioned.

- Every point/vector in space can be analyzed in three elemental vectors, each one parallel to a respective constituent axis of the coordinate system. These vectors are thus linearly independent to each other and directly correspond to the degrees of freedom of the motion.
- Updating position with an (a, b, c) vector is commutative in and of itself. The order of addition of each elemental vector is irrespective to the resulting point/vector in space, the result being the same.
- An UKF involves the averaging of sigma points, which for position can be adequately described by their barycentric mean.

3.1.2 Orientation Determination

3.1.2.1 Euler Angles

The second quantity of interest in an INS, the orientation of an object, obviously cannot be described using a Cartesian representation. In this case, one's goal is to determine the attitude/angular position at any given time with respect to a global fixed coordinate system. Mimicking position determination, any rotation can be composed out of three elemental rotations, which in turn define three angles (ϕ, θ, ψ), known as *Euler angles*; the difference being that *these rotations do not commute*, i.e., changing the order of the same set of elemental rotations, gives a different resulting orientation. It is evident that commutation is non-applicable to rotations and a three dimensional vector of angles does not fully describe what is taking place. Fig. 3.1 shows the non-commutation stated above. In order to get to $(\mathrm{A}, \mathrm{B}, \mathrm{C})$, one has to first rotate around the z axis, then around the y axis of the perturbed coordinate system $(x^{'}, y^{'}, z^{'})$, and finally around the z axis of the $(x^{''}, y^{''}, z^{''})$ system. The $z-y-z$ or $3-2-3$ sequence is one of the many that can possibly be used when describing rotations via Euler angles [22].

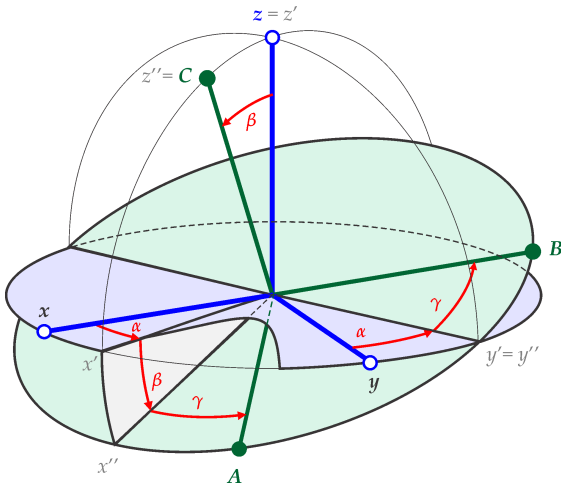


Figure 3.1: The change in orientation from (x, y, z) to $(\mathrm{A}, \mathrm{B}, \mathrm{C})$ using elemental rotations (figure taken from [23]).

Given a selection of axes and a rotation order, Euler angles can be used to express orientation. This is done through the construction of a rotation matrix R which can be used to update the orientation of an arbitrary vector, r, through $r^{'} = Rr$. A detailed presentation of the possible rotation matrices is given in [22]. The insight that is gained through this process can lead to the following observations.

- Converting from rotation matrices back to Euler angles can result in singularities for certain values of the angles (ϕ, θ, ψ)
- It is computationally expensive to construct a 3×3 matrix every time an orientation update is being done.

- There might be cases where the rotations are not well defined, i.e., two different sets of elemental rotations giving the same resulting rotation [24].

The reasons stated above put forth the need of expressing orientation in a more powerful way. A way that can ensure a compact form of expressing rotations that avoids singularity issues is to use *quaternions*.

3.1.2.2 Quaternions

Quaternions, from a mathematical perspective, can be viewed as an extension to the complex numbers by adding two more dimensions to the imaginary part, of the form

$$\begin{aligned} &q = \alpha + \beta i + \gamma j + \delta k \quad \text{where} \quad [\alpha, \beta, \gamma, \delta] \in \mathbb{R}, \\ &i^2 = j^2 = k^2 = ijk = -1, \\ &ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j. \end{aligned}$$

The special case quaternion where $|q| = \sqrt{\alpha^2 + \beta^2 + \gamma^2 + \delta^2} = 1$ is called *unit quaternion*. To examine how the unit quaternion can be used to represent attitude, it is useful to consider the following.

- Given an arbitrary vector r and a unit quaternion q, $r^{'} = qr$ presents a linear transformation of r where $|r^{'}| = |q||r| = |r|$.
- Given a unit quaternion q, two arbitrary vectors r_1 and r_2, and their linear transformations $r_1^{'} = qr_1$, $r_2^{'} = qr_2$, it can be proven that the dot product is preserved, i.e., $r_1 \cdot r_2 = r_1^{'} \cdot r_2^{'}$.

Multiplying vectors by a unit quaternion causes, thus, for them to transform while retaining their length and the vector space between them, or in other words, *rotate*. According to Euler's rotation theorem, any change in orientation of a rigid body can be expressed as a rotation around a fixed axis $\hat{n}$, for some angle θ. Therefore, a systematic way must exist, to express this rotation using quaternion representation. The $(\hat{n}, \theta)$ axis-angle representation can be composed into a *rotation vector* of the form

$$u = \frac{\theta}{2}\hat{n} \tag{3.1}$$

whose exponential can be analyzed using a Taylor series expansion [12] as

$$q = e^u = e^{\frac{\theta}{2}\hat{n}} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2})\hat{n} \end{bmatrix} = \begin{bmatrix} q_w \\ \mathbf{q_v} \end{bmatrix}. \tag{3.2}$$

It is evident that q is a unit quaternion since the following are valid.

- It is a four dimensional quantity with a real part q_w and an (expanded by two dimensions) imaginary part $\mathbf{q_v}$.

- $|q| = \sqrt{\sin\left(\frac{\theta}{2}\right)^2 + \cos\left(\frac{\theta}{2}\right)^2}$ is always equal to 1.

If it is furthermore multiplied to an arbitrary vector, r, as shown in (3.3), it performs the rotation that the $(\hat{n}, \theta)$ axis-angle representation necessitates. The result is that the rotated vector $r^{'}$ is obtained. It should be noted that in quaternion space, quaternion multiplication is defined by the $\otimes$ sign.

$$r^{'} = q \otimes r \otimes q^{*} \tag{3.3}$$

The conjugate of a quaternion is

$$q^{*} = \begin{bmatrix} q_w \\ -\mathbf{q_v} \end{bmatrix}. \tag{3.4}$$

There are two other important properties. The product of two unit quaternions is always a unit quaternion as shown in (3.5) and a quaternion can be constructed out of any rotation vector, as defined in (3.1), using (3.6).

$$q_3 = q_1 \otimes q_2 \Rightarrow |q_3| = 1 \tag{3.5}$$

$$q = \begin{bmatrix} \cos\left(\frac{|u|}{2}\right) \\ \sin\left(\frac{|u|}{2}\right) \frac{u}{|u|} \end{bmatrix}. \tag{3.6}$$

Quaternions, therefore, can indeed express relative attitude, avoiding the singularity issues that Euler angles suffer from and having a more compact form (no need for the construction of a rotation matrix to perform an update). However, despite the fact that a quaternion can describe a rotation fully by itself, combining rotations(quaternions) still carries the non-commutative property. Compared to position determination, attitude determination through quaternions suffers from the following.

- Quaternions, a four dimensional entity, is used to describe a three degree of freedom motion.
- Averaging quaternions, through their barycentric mean, lacks physical foundation and can violate the unit norm constraint on them.

The challenge posed is, therefore, to design an algorithm which implicitly retains the unit norm while efficiently tackles the difference in dimensionality.

3.2 Gyroscope Modelling

The current report studies attitude navigation in an IMU driven system where instrumentation includes a STIM300 IMU provided by *Sensonor AS*. While process models in general can be constructed through the combination of system dynamics and kinematics, when systems are IMU driven, system dynamics is replaced by gyroscope data which are being used directly in the kinematics equation to retrieve attitude. This dead reckoning process accumulates error over time, due to instrument noise and systematic error such

as bias drift, giving rise to the need of sensor fusion via KF. Gyroscope modelling is a useful tool, while developing said filters since it (i) enables to get simulated measurements along experimental ones and (ii) It gives the ability for large contributing errors to be estimated and removed, improving performance.

Two main types of errors are present in gyroscopes, namely (i) *deterministic* that are constant in value, any time the gyroscope operates and (ii) *stochastic* that change constantly, which can be accounted for by incorporating their estimation in the filter. A general flow chart of the gyroscope process is found in Fig.3.2.

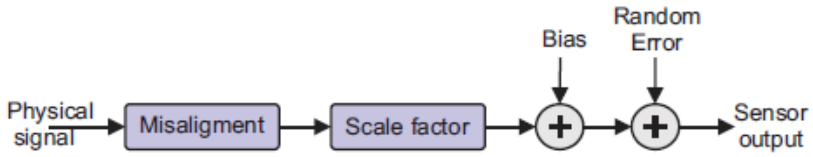


Figure 3.2: Gyro process (figure taken from [30]).

The equivalent mathematical model is

$$\underbrace{\widetilde{\omega_g}}_{\text{measured rate}} = \overbrace{(I_{3\times3} + M_g)}^{\text{scale factor+misalignment}} \underbrace{\omega_g}_{\text{true rate}} + \overbrace{b_g}^{\text{bias}} + \underbrace{w_g}_{\text{random error}} . \tag{3.7}$$

Scale factor(s_g) and *misalignment* (m_g) errors are deterministic and can be combined into a single matrix

$$M_g = \begin{bmatrix} s_{g_x} & m_{g_{xy}} & m_{g_{xz}} \\ m_{g_{yx}} & s_{g_y} & m_{g_{yz}} \\ m_{g_{zx}} & m_{g_{zy}} & s_{g_z} \end{bmatrix}. \tag{3.8}$$

A way to estimate the scale factor error is presented in [25]. The gyroscope is mounted on a rotation table which is spun for a known rate $\vec{\omega}$ and a set of two different measurements is taken, sequentially for each axis of rotation. The two measurements differ in the way the gyroscope is mounted; its axes of rotation aligned to positive and negative gravity. The errors can then be determined by

$$\begin{bmatrix} s_{g_x} & 0 & 0 \\ 0 & s_{g_y} & 0 \\ 0 & 0 & s_{g_z} \end{bmatrix} = \begin{bmatrix} \widetilde{\omega_{g_x}}^+ - \widetilde{\omega_{g_x}}^- & 0 & 0 \\ 0 & \widetilde{\omega_{g_y}}^+ - \widetilde{\omega_{g_y}}^- & 0 \\ 0 & 0 & \widetilde{\omega_{g_z}}^+ - \widetilde{\omega_{g_z}}^- \end{bmatrix} \frac{1}{2\vec{\omega}} - I_{3\times3}. \tag{3.9}$$

The misalignment error follows a similar path, only in this case the recorded data during a test are not the angular rates on the principal axis of rotation, but on the other two axes. The error is determined to be

$$\begin{bmatrix} 0 & m_{g_{xy}} & m_{g_{xz}} \\ m_{g_{yx}} & 0 & m_{g_{yz}} \\ m_{g_{zx}} & m_{g_{zy}} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \widetilde{\omega_{g_y}}^+ - \widetilde{\omega_{g_y}}^- & \widetilde{\omega_{g_z}}^+ - \widetilde{\omega_{g_z}}^- \\ \widetilde{\omega_{g_x}}^+ - \widetilde{\omega_{g_x}}^- & 0 & \widetilde{\omega_{g_z}}^+ - \widetilde{\omega_{g_z}}^- \\ \widetilde{\omega_{g_x}}^+ - \widetilde{\omega_{g_x}}^- & \widetilde{\omega_{g_y}}^+ - \widetilde{\omega_{g_y}}^- & 0 \end{bmatrix} \frac{1}{2\vec{\omega}}. \tag{3.10}$$

The random error can be treated as a zero-mean Gaussian white noise with standard deviation

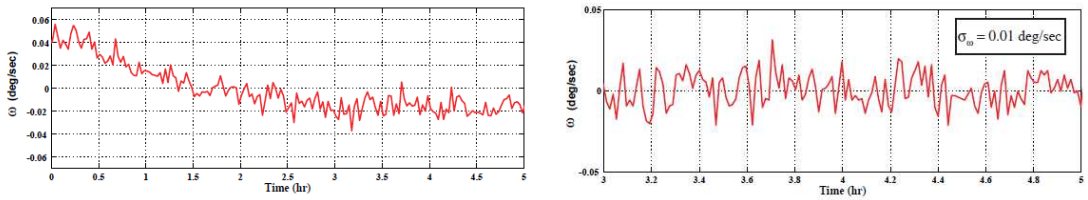
$$\sigma_{w_g} = \frac{\text{ARW}}{\sqrt{\Delta T}} \tag{3.11}$$

where *Angular Random Walk*, ARW, is specified in the gyroscope datasheet. All the values must be converted to rad/s.

3.2.1 Bias Model

Bias error can be analysed into three components, fixed bias, run-run bias and in-run bias.

- Fixed bias is a deterministic error due to external factors like temperature. Fig. 3.3 [26] shows such a bias, which is compensated for in the calibrated gyroscope used [27] and thus is neglected.
- Run-run bias is a constant in-run value which changes from run ro run. It is essentially the mean value of a no-rate measurement scenario.
- In-run bias is a stochastic error which can be modelled either as a random walk [28]. a Gauss-Markov correlated noise ([29], [30]) or neglected [31] if it has negligible effect on the total instrument noise.



(a) Uncalibrated gyroscope

(b) Calibrated gyroscope

Figure 3.3: Effect of temperature on bias (figure taken from [26]). (a) The offset drift with temperature without compensation. (b) The improved offset stability as obtained using temperature compensation.

3.2.1.1 Random Walk Model

The discrete-time random walk model is

$$b_{g_k} = b_{g_{k-1}} + w_{b_g} \tag{3.12}$$

where b_{g_0} is the run-run bias as described above and w_{b_g} is a zero-mean Gaussian white noise. The standard deviation of this noise is presented below with σ_{BI} being obtained from the datasheet and τ_c being the correlation time.

$$\sigma_{w_{b_g}} = \sigma_{BI}\sqrt{\frac{\Delta T}{\tau_c}} \tag{3.13}$$

3.2.1.2 Gauss-Markov Model

The continuous-time model of a first-order Gauss-Markov process is presented in (3.14) [29], along with its equivalent discrete-time form

$$\tau_c \dot{b_g} + b_g = w_{b_g} \Rightarrow b_{g_k} = \alpha_d b_{g_{k-1}} + g_d w_{b_g} \tag{3.14}$$

where $\alpha_d = e^{-\frac{1}{\tau_c}\Delta T}$, $g_d = \int_0^{\Delta T} e^{-\frac{1}{\tau_c}t} dt$, b_{g_0} is the run-run bias as described above and w_{b_g} is a zero-mean Gaussian white noise. The standard deviation of this noise is as follows

$$\sigma_{w_{b_g}} = \sqrt{1 - \frac{\alpha_d^2 \sigma_{BI}}{0.664 b_d}}. \tag{3.15}$$

3.2.2 Allan Variance Analysis

Gyroscope Model Determination

A tool to determine what are the dominant disturbances in a gyroscope is the Allan Variance plot. A typical plot has the form of Fig.3.4. This plot follows the slope of a curve, which in this case has three distinct parts. A part where the slope is $-1/2$, a part where it is 0 and a part where it is $1/2$. What this means is that the stochastic errors will be modelled, reflecting the three distinct parts in the following way. A zero-mean Gaussian white noise accounting for the random error, a random walk or Gauss-Markov model accounting for in-run bias, and a random walk model accounting for random error in angular acceleration.

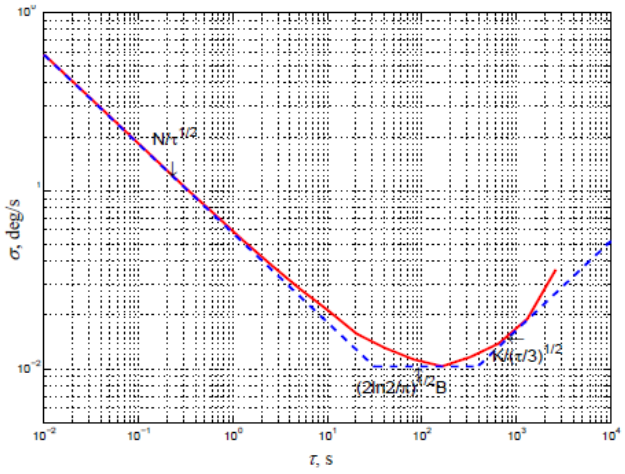


Figure 3.4: Generic Allan Variance plot (figure taken from [29]).

As shown in [31] and [26] in tactical grade IMUs, Allan Variance plot is dominated by the $-1/2$ slope curve, indicating that all stochastic models, other than that of a white

noise can be neglected. By removing all the negligible stochastic processes, keeping only the random error and the run-run bias, (3.7) can be written as

$$\widetilde{\omega_g} = (I_{3\times3} + M_g)\omega_g + b_{g0} + w_g \Rightarrow \omega_g = (I_{3\times3} + M_g)^{-1}(\widetilde{\omega_g} - b_{g0} - w_g). \tag{3.16}$$

Alternatively by taking bias into account, (3.16) is written as

$$\omega_g = (I_{3\times3} + M_g)^{-1}(\widetilde{\omega_g} - b_g - w_g) \tag{3.17}$$

where b_g is found using (3.12).

Correlation Time Determination

The correlation time, τ_c, physically refers to how quickly the stochastic process "establishes" the bounds in uncertainty as characterized by the standard deviation. Fig. 3.5 [26] shows a Gauss-Markov process with a correlation time $\tau_c = 200$ s.

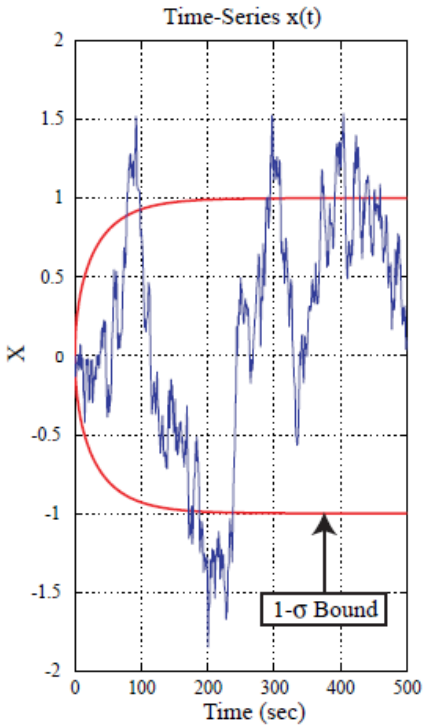


Figure 3.5: Standard Deviation in a Gauss-Markov process (figure taken from [26]). The red curve is the (1-σ) bound and the blue curve is the random process itself.

An approximation on it can be given from the Allan Variance plot as cited by [30] and [28]. According to [30] it is the average time where the flat-line asymptote of Fig. 3.4 is present, while [28] takes it as the time where the Allan Variance curve goes to its lowest point. More systematic ways of calculating include the autocorrelation function and autoregressive processes, details on both can be found in [30].

3.2.3 Gyroscope Model Validation

A 30 min static measurement of the gyroscope was made, and its Allan Variance plot is presented using the NaveGo Algorithm [32]. Instead of consulting the datasheet for parameter values, the algorithm made it feasible to obtain them through the experimental data. These parameters were subsequently plugged in the two different gyroscope models (3.16), (3.17). The 30 min static measurement was emulated using the two models and their Allan Variance plots were recomputed. It can be seen (Figs. 3.6 – 3.8) that both of them provide a fairly good approximation of the true plot. For the rest of the analysis the simpler model as presented in (3.16) will be used. Validation of this model will be further verified when emulated and experimental data will be used to obtain attitude through dead reckoning; their respective accumulated attitude errors must be really close.

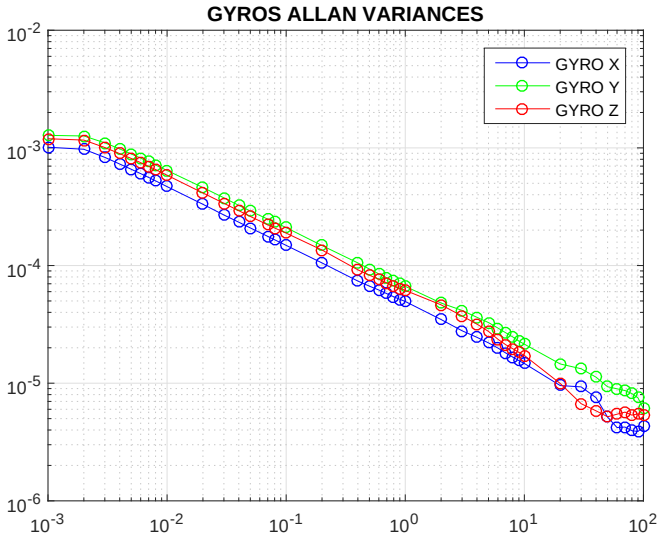


Figure 3.6: Allan Variance of the STIM300 IMU. The x-axis is in s and the y-axis in rad/s.

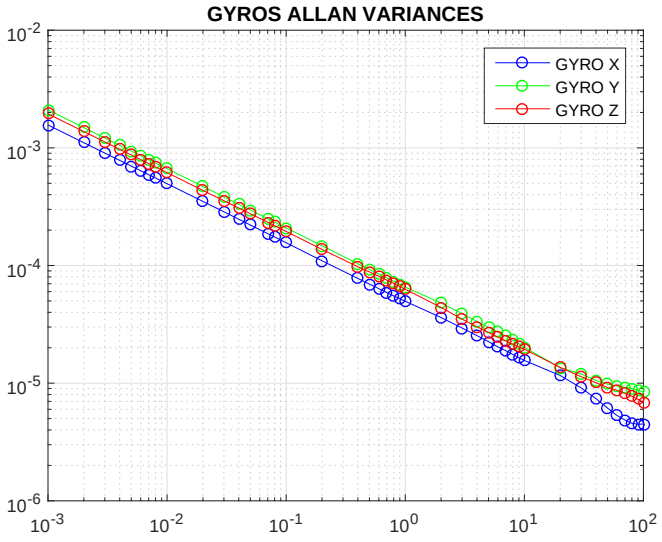


Figure 3.7: Allan Variance of the emulated data using the model indicated in (3.16). The x-axis is in s and the y-axis in rad/s.

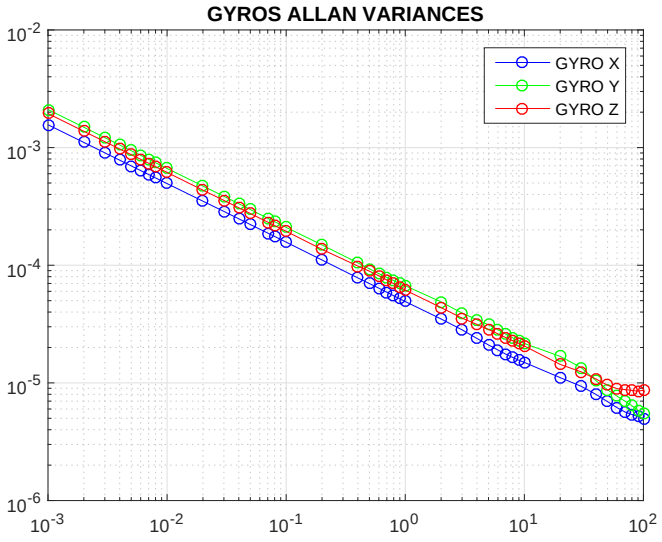


Figure 3.8: Allan Variance of the emulated data using the model indicated in (3.17). The x-axis is in s and the y-axis in rad/s.

3.3 Magnetometer Modelling

In a similar fashion to what was described above, magnetometer modelling is also implemented to follow the LEMI-020A magnetometer provided by *Lviv Centre of Institute for Space Research*. Since, for the purposes of the current report, the attitude navigation algorithm is developed to be used in sounding rockets, using coordinates of a launch site and consulting [33], one can see that for altitude between 0 and 10 km the magnetic field reduces by some 0.5% with a constant direction. The background field, thus, can to sufficient accuracy be assumed constant. For a magnetic field, B_n, in the navigation frame

$$B_b = R_b^n B_n \tag{3.18}$$

is how it evolves in the local/body coordinate frame. The model is augmented with bias and random processes as discussed above taking the form of

$$B_b = R_b^n B_n + b_m + w_m \tag{3.19}$$

where R_b^n is the rotation matrix from the navigation frame to the body frame, b_m is the bias of the magnetometer and w_m a zero-mean Gaussian white noise with standard deviation provided by the magnetometer datasheet.

3.3.1 Magnetometer Model Validation

Measurements with the magnetometer mounted on a rate table for different angular rates around one axis were carried out. Based on the model described in (3.19), one would expect for the magnetometer data to remain constant around the axis of rotation and follow a sinusoidal form on the other two axes. This was not the case and what became visible instead is that the presence of various electronic devices caused for the magnetic field to lose *uniformity*. The magnetometer readings, thus, despite being periodic were not sinusoidal (Fig. 3.9).

To counter this effect the magnetometer was moved so that one of its axis was as close as possible to matching the axis of rotation. Since the precision of the positioning of the magnetometer was not ideal some form of periodicity albeit greatly reduced remained present (Fig. 3.10). The bias in (3.19) was treated as a constant value being measured for an obtained set of measurements and subsequently removed. Fig. 3.11 shows how simulated and experimental data compare to each other following the analysis stated above.

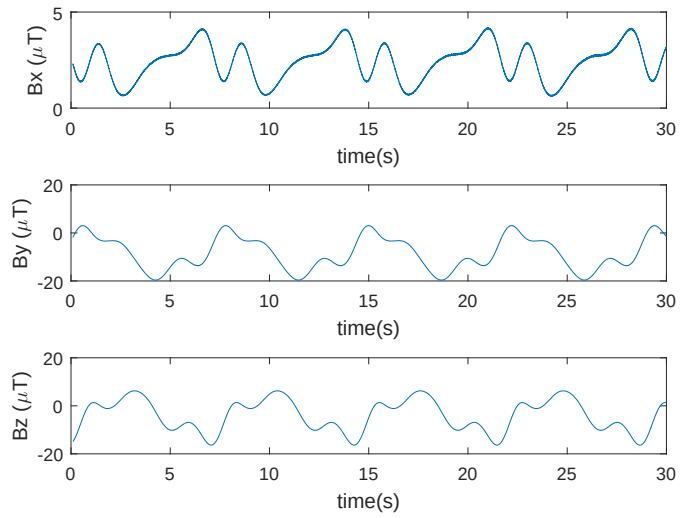


Figure 3.9: Magnetometer readings in x, y and z axes.

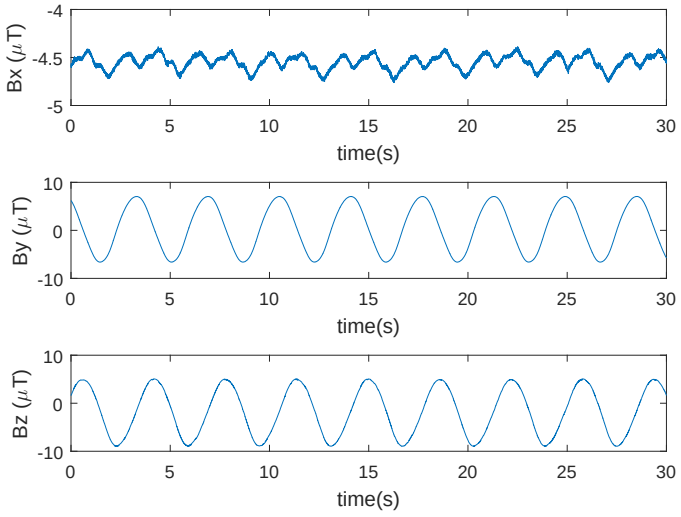


Figure 3.10: Magnetometer readings in x, y and z axes for differently positioned magnetometer.

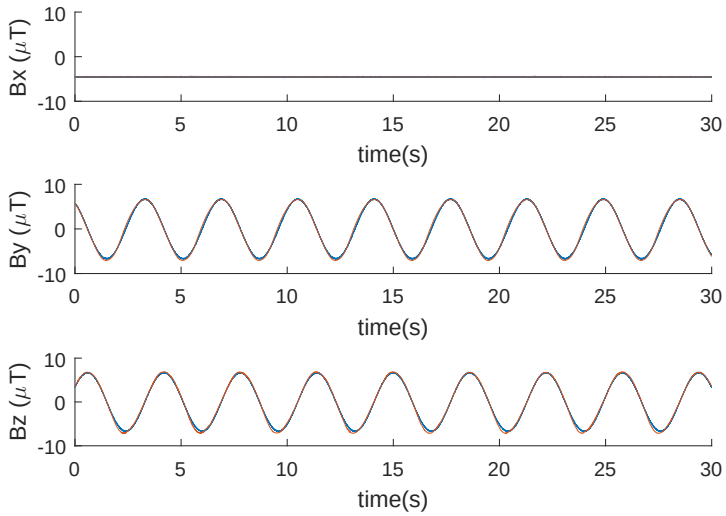


Figure 3.11: Magnetometer readings in x, y and z axes. Emulated (blue curve) and experimental (red curve) data comparison.

3.4 Quaternion-Based Unscented Kalman Filtering

3.4.1 Process Model

Given that, in this case, an IMU-driven system is studied, it is important to note that the gyroscope-measured angular rate is

$$\omega_g = \omega_{nb}^b. \tag{3.20}$$

Following the notation in (3.20), the angular rate given by the IMU is that of the body (b of the subscript) with respect to the navigation frame (n of the subscript) in local(body-frame) coordinates (b of the superscript). It is deemed, thus, convenient to maintain this locality in forming the process model in order to use directly the measured gyroscope data without reverting them to their global value

$$\omega_G = \omega_{nb}^n = R_b^n \omega_{nb}^b = R_b^n \omega_g. \tag{3.21}$$

The process model is formed by treating the change in orientation at each timestep as infinitesimal, in essence a perturbation Δq, an assumption which holds ground given the high sampling rate of the gyroscope. The locality of this perturbation is maintained by placing the quaternions in the following order

$$\widetilde{q} = q \otimes \Delta q. \tag{3.22}$$

It is evident, as (3.22) shows, that at every timestep, the previous local coordinate frame, q, is retrieved before being perturbed by Δq. Following (3.2) a generic perturbation

quaternion is formed [7].

$$\Delta q = \begin{bmatrix} \cos\left(\frac{\delta\theta}{2}\right) \\ \sin\left(\frac{\delta\theta}{2}\right)\hat{n} \end{bmatrix} = \begin{bmatrix} \Delta q_w \\ \mathbf{\Delta q_v} \end{bmatrix} \tag{3.23}$$

$$\delta\theta = |\omega_g|\Delta T \tag{3.24}$$

$$\hat{n} = \frac{\omega_g}{|\omega_g|} \tag{3.25}$$

The operation of (3.22) can be performed as a normal matrix product

$$\widetilde{q} = [\Delta q]q \tag{3.26}$$

where

$$[\Delta q] = \begin{bmatrix} \Delta q_w & -\mathbf{\Delta q_v}^\intercal \\ \mathbf{\Delta q_v} & \Delta q_w \mathbf{I}_{3\times 3} - [\mathbf{\Delta q_v}]_\times \end{bmatrix}. \tag{3.27}$$

$[a]_\times$ is the skew-symmetric matrix of a vector, generally computed as

$$[a]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \tag{3.28}$$

Replacing $\widetilde{q} \to q_k, \quad q \to q_{k-1}, \quad \Delta q \to \Delta q_k, \quad \delta\theta \to \delta\theta_k$ gives the discrete-time process model based on the quaternion kinematics. A 5 min measurement test was performed on a rotation table with angular rate of 100 deg/s around the x-axis. The same test was also emulated via the gyroscope model. Using the process model described above the dead reckoning approach to obtaining attitude was used, and the attitude error in both cases is presented in Fig.(3.12) and Fig.(3.13).

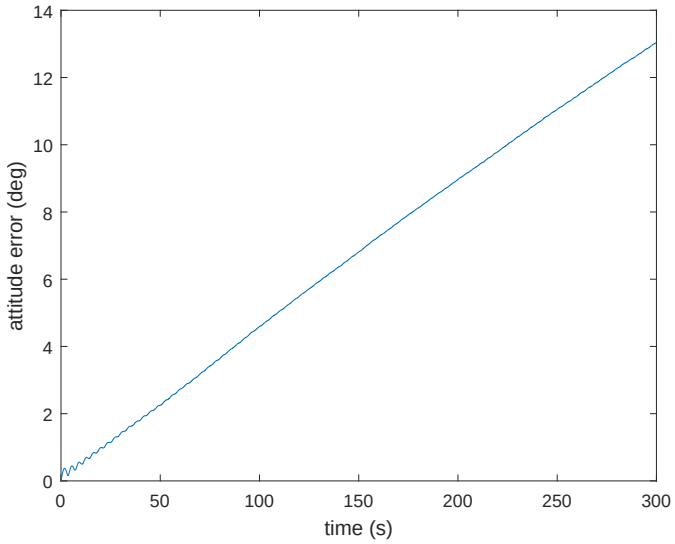


Figure 3.12: Attitude error using IMU-obtained data.

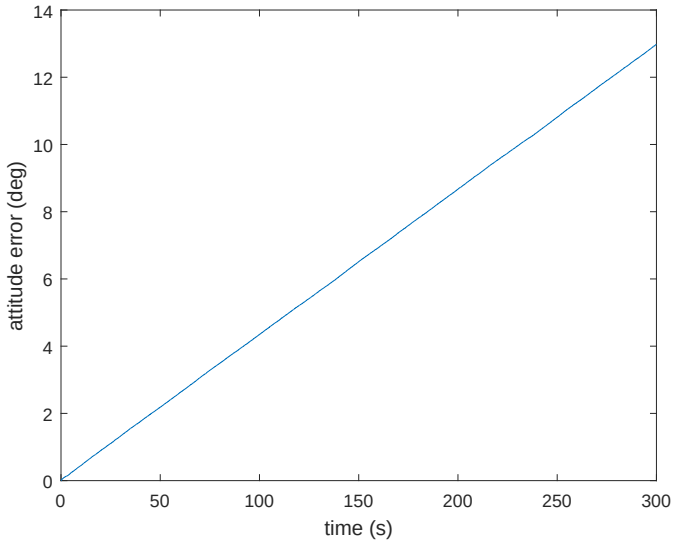


Figure 3.13: Attitude error using emulated data.

It is clear that the filtering procedure is necessary to keep the attitude error under the desired threshold and that the gyroscope model is closely approximating the real process. Using (3.26) to propagate attitude quaternion, in theory should maintain the unity norm. However numerical errors accumulate over time, as evident in Fig.(3.14) and can corrupt the result for longer flight times. A formulation of (3.26), as proposed by [6] where correction factors η, λ are present is

$$\lambda = 1 - ||q||^2, \tag{3.29}$$

$$\eta = \frac{0.5}{\Delta T}, \tag{3.30}$$

$$\widetilde{q} = (I_{4\times 4}(\eta \cdot \lambda \cdot \Delta T) + [\Delta q])q \tag{3.31}$$

which allows for the norm to stay close to unity at all times, as shown in Fig.(3.15)

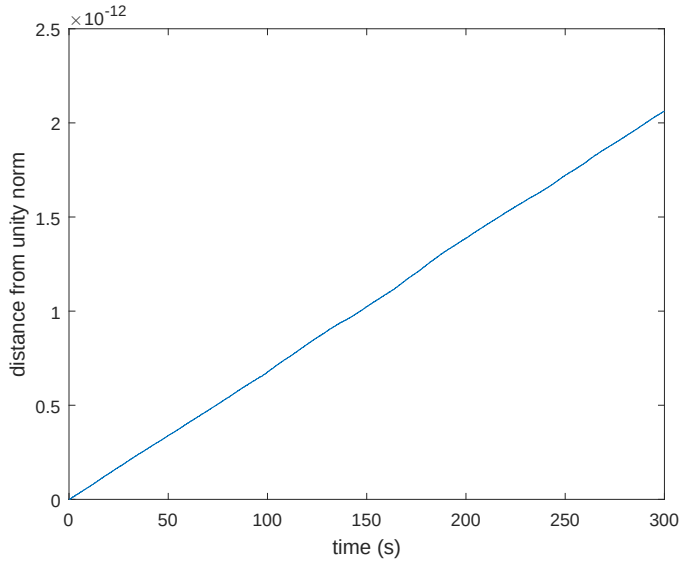


Figure 3.14: Absolute value of $(||q|| - 1)$.

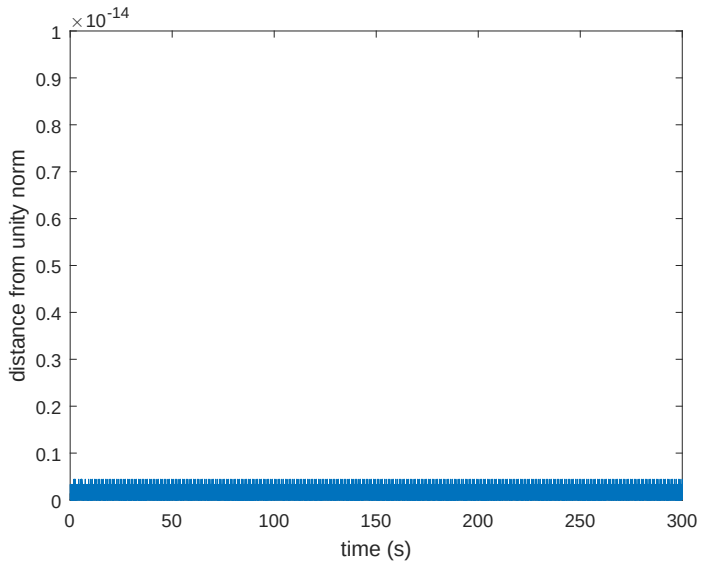


Figure 3.15: Absolute value of $(||q|| - 1)$.

3.4.2 Challenges Posed by the Presence of Quaternions in UKF

The trivial way to tackle the problem is to formulate it with a state vector of the form

$$\mathbf{x} = q = \begin{bmatrix} q_w \\ \mathbf{q_v} \end{bmatrix}. \tag{3.32}$$

At this point it is useful to stress the physical importance of maintaining unity in the norm. A quaternion *rotates* and *scales* a vector. The magnitude of scaling depends on the quadrance of the quaternion $|q|^2$. To maintain the size of the vector as it is, keeping the rotation process faithful to reality, the quaternion must keep its quadrance and implicitly its norm equal to 1.

Sigma-Point Generation

Generation of sigma points cannot follow the normal route as described in [6] which is

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[q_{k-1}, q_{k-1} + \gamma\sqrt{\mathbf{P}_{k-1}}, q_{k-1} - \gamma\sqrt{\mathbf{P}_{k-1}} \right]. \tag{3.33}$$

There are two main reasons for this.

- The covariance matrix dimensions correspond to the degrees of freedom of the motion, in this case being a 3×3 matrix. It cannot be added to a 4-dimensional state vector.
- The additive property, even if it was possible, causes violation of the unit norm constraint of the quaternion.

The difference in dimensionality is resolved following a procedure described in [7]. Unit quaternions, $q_{\mathcal{W}}$, are constructed using (3.2), off the columns of

$$\boldsymbol{\mathcal{W}}_{i,i+n} = \text{columns}\left(\pm\gamma\sqrt{(\mathbf{P}_{k-1} + \mathbf{R}^{\mathbf{v}})}\right). \tag{3.34}$$

Sigma points are then formulated via multiplication

$$\boldsymbol{\mathcal{X}}_{k-1} = [q_{k-1}, q_{k-1} \otimes q_{\mathcal{W}}]. \tag{3.35}$$

The additive property as stated in (3.33) can be used, on condition that it is followed by normalization [8]. However this is an ad-hoc solution which is not desirable, since it is "nesting" in the scalar part of the quaternion, information originally present in the vector part and vice-versa. Using the process model as described above the sigma points can be propagated to obtain

$$\boldsymbol{\mathcal{X}}^*_{k|k-1} = F(\boldsymbol{\mathcal{X}}_{k-1}). \tag{3.36}$$

Averaging Quaternions

The process of averaging or weighted averaging is crucial for the implementation of the UKF. As far as quaternions are concerned, however, non-commutation poses a challenge. Sigma-point generation provides a set of rotations that reflect the uncertainty bounds as laid forth in the covariance matrices. This set is not a sequence of rotations and no order in it exists, therefore, averaging lacks physical sense. In [34] it is claimed that averaging rotations can be approximated by the barycentric mean of the quaternions followed by renormalization,

$$\underline{q}_k = \frac{\sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}}{|\sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}|}. \tag{3.37}$$

An issue with this approach is that possible quaternions q and $-q$ as obtained from the sigma-point generation cancel each other out, despite the fact that they describe the exact same rotation. The problem can be treated as one of averaging the squared Frobenius norm of the attitude matrix differences as stated in [35],

$$\underline{q}_k = \underset{\underline{q}_k \in \mathbb{S}^3}{\text{argmin}} \sum_{i=0}^{2L} W_i^{(m)} ||A(q) - A(\mathcal{X}^*_{i,k|k-1})||^2_F. \tag{3.38}$$

where $\mathbb{S}^3$ denotes the unit 3-sphere. (3.38) is further shown in [35] to transform into a maximization problem of the form,

$$\underline{q}_k = \underset{\underline{q}_k \in \mathbb{S}^3}{\text{argmax}}\, q^{\mathrm{T}} M q, \tag{3.39}$$

for

$$M = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1} {\mathcal{X}^*_{i,k|k-1}}^{\mathrm{T}}. \tag{3.40}$$

The solution to (3.39) is the eigenvector of M which corresponds to its maximum eigenvalue. For positive eigenvalues the power iteration can be used to provide a solution.

Reconstructing the Covariance Matrix

A dominant process in the UKF algorithm is the construction of a covariance matrix after obtaining the mean and sigma-points. It is calculated as follows,

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k][\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k]^{\mathrm{T}} + \mathbf{R}^{\mathbf{v}}. \tag{3.41}$$

In the case of quaternion-based UKF, the dimensionality difference between the covariance matrix and quaternions, as well as the physical properties of the quaternions, prohibit the use of the normal procedure described above. In rotation space, the difference between two rotations q_1 and q_2 cannot be found via subtraction, instead it is found as,

$$e_{12} = d(q_1, q_2) = q_1 \otimes q_2^{-1}. \tag{3.42}$$

A set of rotation quaternions $\{e_i\}_{i=0}^{2L}$ is calculated between the newly computed mean $\underline{q}_k$ and the sigma points $\mathcal{X}^*_{i,k|k-1}$,

$$e^*_{i,k|k-1} = d(\mathcal{X}^*_{i,k|k-1}, \underline{q}_k) = \mathcal{X}^*_{i,k|k-1} \otimes \underline{q}_k^{-1}. \tag{3.43}$$

This is then transformed into a set of rotation vectors $\{\tilde{e}^*_{i,k|k-1}\}_{i=0}^{2L}$, using (3.6), to tackle the dimensionality problem [7]. The covariance is now computed as

$$\mathbf{P}_k = \sum_{i=0}^{2L} W_i^{(c)} \tilde{e}^*_{i,k|k-1} \tilde{e}^{*\mathrm{T}}_{i,k|k-1}. \tag{3.44}$$

Measurement Update

On obtaining a new mean $\underline{q}_k$ and covariance, sigma points $\boldsymbol{\mathcal{X}}_{\boldsymbol{k|k-1}}$ can be computed using (3.35). They are subsequently transformed to sigma measurement points via

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}]. \tag{3.45}$$

For measurement instruments such as magnetometers or sunsensors which measure vectors the rest of the UKF algorithm can follow the procedure as stated in [6] with the distinction that the cross-covariance is calculated as

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} \tilde{e}_{i,k|k-1} [\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^{\mathrm{T}} \tag{3.46}$$

where $\tilde{e}_{i,k|k-1}$ is calculated similarly to (3.43). This concludes the direct method. It is evident that many steps of the generic Unscented Formulation need to be altered to account for the unique characteristics of the quaternions. In order for the algorithm to maintain its structure a marginalised approach is presented below.

3.4.3 Error State UKF

The two basic properties of the quaternions that dictate the changes in the UKF formulation presented above are the following.

- non-commutation
- dimensionality difference with the unity norm constraint that comes with it.

Both of them can be disregarded in infinitesimal rotation space, without loss of generality. This leads to an alternative large signal-small signal approach as shown in Fig. 3.16.

- in the first phase (large signal dominant), nominal state kinematics are used to provide an estimate of the attitude. This phase lasts as long as the high frequency IMU data are gathered without being interrupted by other types of data (such as the magnetometer ones). During this phase the error-state is continuously *predicted* via the error state kinematics of (3.48).
- in the second phase (small signal dominant), upon gathering data from other sensors, the error state prediction is corrected and the error state estimate is used to correct the attitude estimate.
- after the correction of the attitude estimate, the error state is reset and the process is repeated in a cyclic fashion.

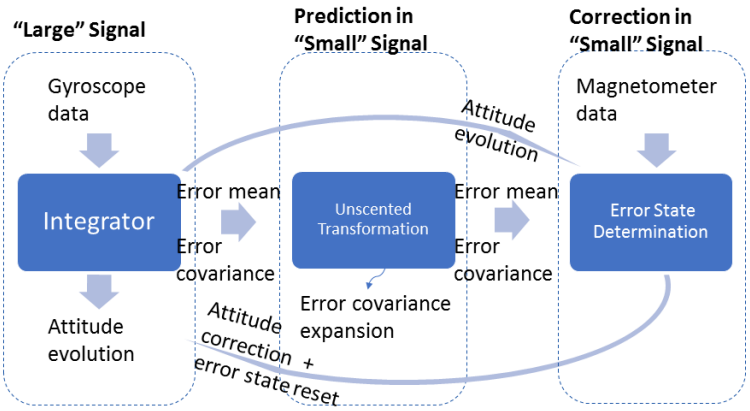


Figure 3.16: Error State Filtering Procedure.

The error state kinematics as will be presented below, due to their infinitesimal nature can be decoupled of commutation; furthermore, using the quaternion exponential approach the error quaternion can be faithfully recreated from a 3-dimensional error angle vector. The result is that by incorporating the error angle vector in the state vector one is able to follow the generic UKF approach without any custom changes to it. The distinction between nominal and error states is clearly presented in Table 3.1. A more detailed look on the error state approach as well as the theoretical derivations behind it can be seen in [12].

Table 3.1: True, Nominal & Error States.

Magnitude	True	Nominal	Error	Composition
Full State	$\mathrm{x_t}$	x	δx	$\mathrm{x} \oplus \delta\mathrm{x}$
Quaternion	$\mathrm{q_t}$	q	δq	$\mathrm{q} \otimes \delta\mathrm{q}$
Rotation Matrix	$\mathrm{R_t}$	R	δR	RδR
Angles Vector			$\delta\theta$	$\delta\mathrm{q} = e^{\frac{\delta\theta}{2}}$
Angular Rate	ω_g	ω	$\delta\omega$	$\omega \oplus \delta\omega$

The nominal and error state kinematics are also derived in [12] and are as follows,

$$q \leftarrow q \otimes q\{(I_{3\times3} + M_g)^{-1}(\widetilde{\omega_g} - b_{g0})\Delta T\}, \tag{3.47}$$

$$\delta\theta \leftarrow R^{\mathrm{T}}((I_{3\times3} + M_g)^{-1}(\widetilde{\omega_g} - b_{g0})\Delta T)\delta\theta + \theta_i, \tag{3.48}$$

with θi a random impulse equal to

$$\theta_i = (I_{3\times3} + M_g)^{-1} w_g \Delta T. \tag{3.49}$$

Prediction Step

While (3.47) is used in the large signal dominant phase to provide updates on the attitude quaternions, in the background a UKF prediction step takes place where

$$\hat{\mathbf{x}} = \delta\theta = R^{\mathrm{T}}((I_{3\times3} + M_g)^{-1}(\widetilde{\omega_g} - b_{g0})\Delta T)\delta\theta = F(\delta\theta), \tag{3.50}$$

$$\mathbf{R^v} = \theta_i\theta_i^{\mathrm{T}} = (I_{3\times3} + M_g)^{-1}\sigma_{w_g}^2\Delta T^2(I_{3\times3} + M_g)^{-\mathrm{T}} \tag{3.51}$$

are plugged in the known algorithm,

$$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{k-1} + \gamma\sqrt{\mathbf{P}_{k-1}}, \hat{\mathbf{x}}_{k-1} - \gamma\sqrt{\mathbf{P}_{k-1}}\right], \tag{3.52}$$

$$\boldsymbol{\mathcal{X}}^*_{k|k-1} = \mathbf{F}[\boldsymbol{\mathcal{X}}_{k-1}, \mathbf{u}_{k-1}], \tag{3.53}$$

$$\hat{\mathbf{x}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{X}^*_{i,k|k-1}, \tag{3.54}$$

$$\mathbf{P}_k^- = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{X}^*_{i,k|k-1} - \hat{\mathbf{x}}_k^-]^{\mathrm{T}} + \mathbf{R^v}. \tag{3.55}$$

What one expects from (3.50)-(3.51) is for the a priori estimate of the error state, $\hat{\mathbf{x}}_k^- = \delta\hat{\theta}_k^-$, during this prediction step to remain zero and the a priori estimate of the state covariance, $\mathbf{P}_k^-$, to grow over time.

Correction Step

It is important, for the sake of keeping the integrity of the UKF Algorithm, to formulate the measurement sigma points in a way that it keeps the error-state vector, $\delta\theta$, in the UT loop. In [12], for example, this is not the case; the measurement vector is dependent on normal quaternions which means that the sigma measurement points would have to be formulated via quaternion multiplication as stated in (3.35). Instead, for calculated $\mathbf{P}_k^-$, $\hat{\mathbf{x}}_k^- = \delta\hat{\theta}_k^-$ and nominal attitude, q_k^-, the sigma measurement points are expected to have a weighted average equal to the measurement vector calculated using the nominal attitude,

$$\hat{\mathbf{y}}_k^- = H(q_k^-). \tag{3.56}$$

The sigma measurement points are scattered *around* this mean in distances that are calculated by

$$\boldsymbol{\mathcal{X}}_{k|k-1} = \left[\hat{\mathbf{x}}_k^-, \hat{\mathbf{x}}_k^- + \gamma\sqrt{\mathbf{P}_k^-}, \hat{\mathbf{x}}_k^- - \gamma\sqrt{\mathbf{P}_k^-}\right], \tag{3.57}$$

$$\boldsymbol{\mathcal{Y}}^*_{k|k-1} = \mathbf{H}[\boldsymbol{\mathcal{X}}_{k|k-1}] \tag{3.58}$$

so that the sigma measurement points can be found via element-wise addition,

$$\boldsymbol{\mathcal{Y}}_{k|k-1} = \hat{\mathbf{y}}_k^- \oplus \boldsymbol{\mathcal{Y}}^*_{k|k-1}. \tag{3.59}$$

Using the measurement instrument datasheet a measurement noise covariance can be determined, $\mathbf{R^n}$, and the rest of the process is as follows.

$$\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)}[\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^{\mathrm{T}} + \mathbf{R^n}, \tag{3.60}$$

$$\mathbf{P}_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)}[\mathcal{X}_{i,k|k-1} - \hat{\mathbf{x}}_k^-][\mathcal{Y}_{i,k|k-1} - \hat{\mathbf{y}}_k^-]^{\mathrm{T}}, \tag{3.61}$$

$$\mathcal{K}_k = \mathbf{P}_{\mathbf{x}_k\mathbf{y}_k}(\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k})^{-1}, \tag{3.62}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathcal{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k^-), \tag{3.63}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathcal{K}_k\mathbf{P}_{\tilde{\mathbf{y}}_k\tilde{\mathbf{y}}_k}\mathcal{K}_k^{\mathrm{T}}. \tag{3.64}$$

For $\hat{\mathbf{x}}_k = \delta\theta_k$, the attitude correction can be made as,

$$q_k^+ = q_k^- \otimes e^{\frac{\delta\theta_k}{2}}. \tag{3.65}$$

Error State Reset

The error-state $\delta\theta$ after the correction step must be reset to zero, which also carries some modification onto the covariance. According to [12] this can be done as

$$\delta\theta = 0, \tag{3.66}$$

$$P_k^+ \leftarrow G P_k^+ G^{\mathrm{T}}, \tag{3.67}$$

$$G = I_{3\times 3} - [\frac{1}{2}\delta\theta]_\times. \tag{3.68}$$

In most cases $G \approx I_{3\times 3}$ and the reset is trivial. If this is the case in the attitude navigation implementation it will further diminish the customization of the algorithm.

Chapter 4

Results

The attitude navigation algorithm was tested in the following simulation setups.

(A) A gyroscope model in MATLAB with similar characteristics to the one present in the STIM300 IMU was implemented, as described in chapter 3.2. Data were emulated for a user-defined angular rate and combined with similarly acquired magnetometer ones to estimate the attitude.

(B) The STIM300 IMU was mounted in a rate table and spun for a user-defined angular rate. Data from the gyroscope were collected and inserted in the MATLAB implementation of the attitude navigation algorithm. They were combined with emulated magnetometer data for the same angular rate to estimate the attitude.

(C) Experimental data from both the gyroscope of the STIM300 IMU and the LEMI-020A magnetometer were combined to provide attitude estimation.

The rate table was considered to be highly accurate and to be spinning with the user-defined angular rate, i.e. its errors being negligible. By inputting this angular rate in place of ω_g in (3.22) – (3.25) the *true* evolution of attitude, q_t was obtained. This was, subsequently, compared to the one estimated by the algorithm, q_e, the results being plotted below.

In quaternion space, the correction of the estimated attitude due to the presence of the filter is not trivially visible. It most prominently manifests in the evolution of the *attitude error angle* which is obtained as follows. For every time-step, the q_t and the q_e differ by some

$$\delta q = \begin{bmatrix} \delta q_w \\ \delta \mathbf{q_v} \end{bmatrix} \tag{4.1}$$

which can be computed as shown in (3.42). The δq practically describes the rotation needed to go from the estimated attitude to the true one. Given this δq, the attitude error angle is

$$\theta = 2 \arccos(\delta q_w). \tag{4.2}$$

The three mathematical tools described above namely (i) the cumulative quaternions q_t and q_e (ii) the attitude error quaternion δq and (iii) angle θ are used to gain insight in

the filter mechanization process and how the deterministic and stochastic errors affect the performance of the UKF.

4.1 Effect of Gyroscope Errors in Attitude Determination

The effect of gyroscope errors is shown below using the first simulation setup as presented above. However, for a complete review of how everything operates, the analysis begins with the dead reckoning approach (3.22)-(3.25) both for the true angular rate and the one measured by the gyroscopes. In quaternion space the attitude evolution is presented in Fig. 4.1. Deterministic and stochastic errors on the gyroscope cause for the estimation to diverge or drift from the true attitude. This drift is clearly seen (Fig. 4.2) by magnifying the top plot of Fig. 4.1.

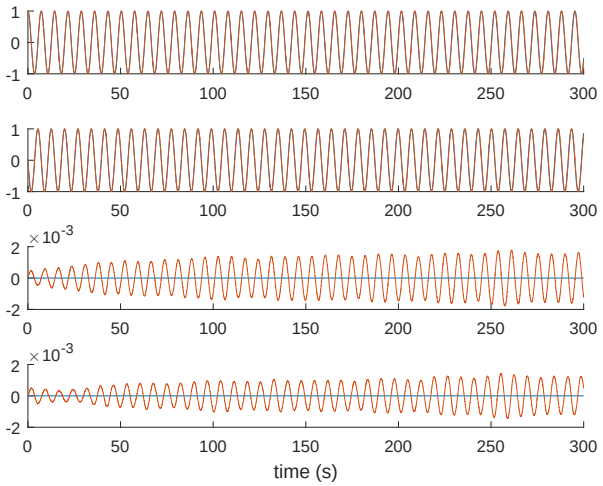


Figure 4.1: Evolution of q_t (blue curve) and q_e (red curve) over a period of 300 s at a rate of 100 deg/s. The top plot is the scalar part of the quaternions q_w while the rest three the vector part $\mathbf{q_v}$. The presence of errors is especially visible in the state evolution on the two latter plots.

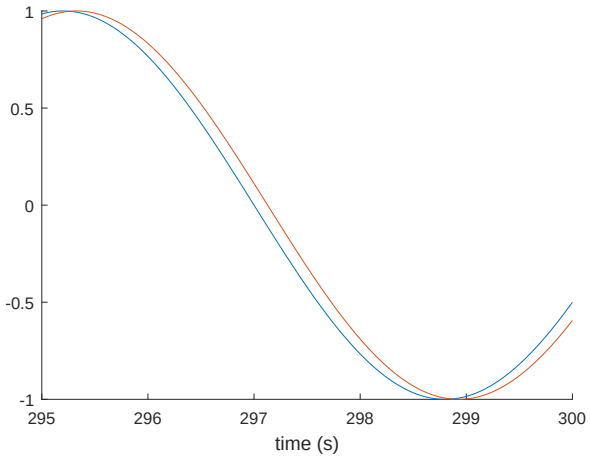


Figure 4.2: Evolution of the scalar part (q_w) of q_t (blue curve) and q_e (red curve) over a period of 300 s at a rate of 100 deg/s, limiting the timespan between 295 and 300 s. The blue curve is drifting away from the red curve as time goes thereby increasing the discrepancy between the true and estimated attitude.

After the dead reckoning approach was presented, the main filtering algorithm was tested without prefiltering or accounting for the bias the results being shown in Figs. 4.3 – 4.4.

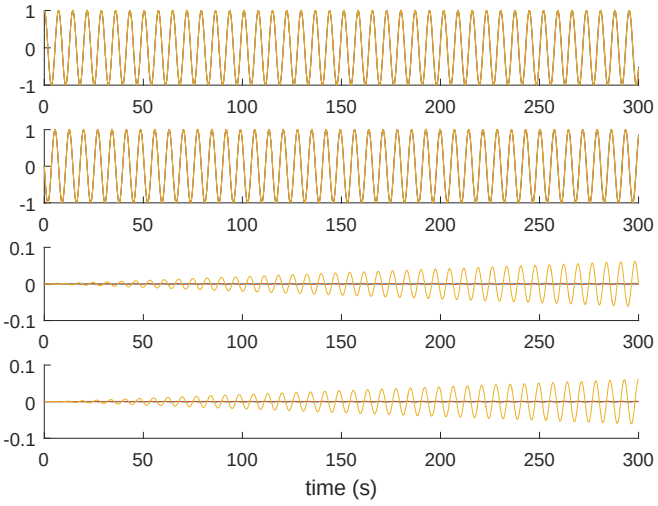


Figure 4.3: Evolution of q_t (blue curve), q_e with no filter enabled (red curve) and q_e with the error-state UKF enabled (yellow curve) over a period of 300 s at a rate of 100 deg/s. The top plot is the scalar part of the quaternions q_w while the rest three the vector part $\mathbf{q_v}$.

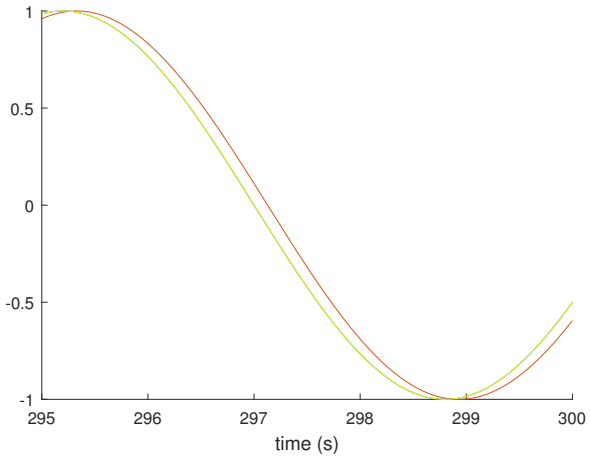


Figure 4.4: Evolution of the scalar part (q_w) of q_t (blue curve), q_e with no filter enabled (red curve) and q_e with the error-state UKF enabled (yellow curve), over a period of 300 s at a rate of 100 deg/s, limiting the timespan between 295 and 300 s. The yellow curve closely follows the red curve indicating that the filter is partly working.

By using the error-state UKF as described in chapter 3.4 the behavior of the estimated quaternion changes as shown in Fig. 4.3. Despite the drift in the top plot being largely corrected, something visible also in Fig. 4.4, errors intensify in the two latter plots which is a physical manifestation of the suboptimality of the UKF in the presence of unfiltered or unaccounted for in the model, deterministic bias. This is a direct consequence of the violation of the unbiased assumption which is a necessary condition for the UKF to be optimal as described in chapter 2.1.

This becomes clear in Figs. 4.5 – 4.6, where the correction is visible in the scalar part of the error quaternion and consequently the error angle. While the filter is managing to attenuate the discrepancy between estimation and reality, it is not able to provide a bounded response.

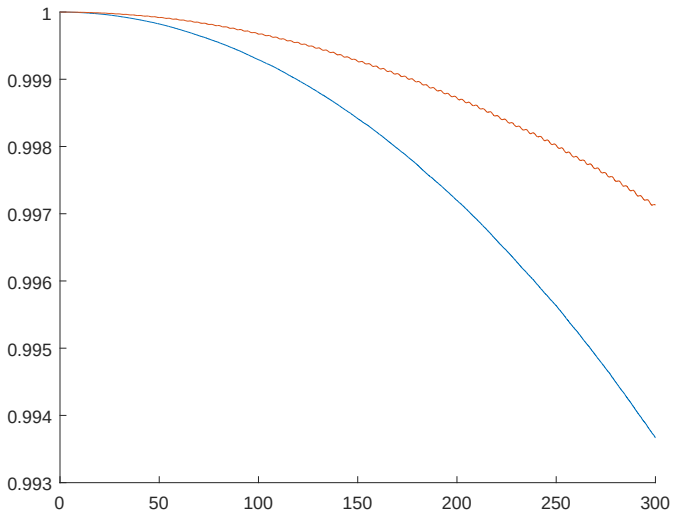


Figure 4.5: Evolution of the scalar part (δq_w) of the attitude error quaternion, δq, with no filter enabled (blue curve) and with the error-state UKF enabled (red curve), over a period of 300 s at a rate of 100 deg/s. A bounded curve close to 1 would be the desirable behavior as evident in (4.2)

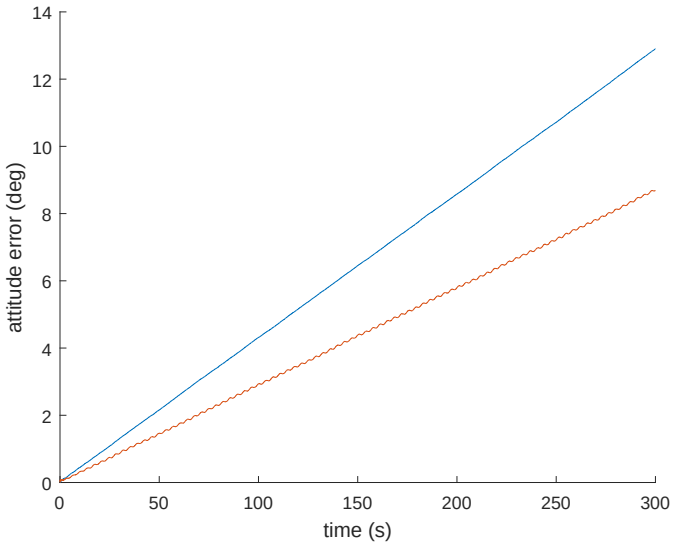


Figure 4.6: Attitude error angle at a rate of 100 deg/s, with UKF disabled (blue curve) and enabled (red curve)

4.2 Attitude Determination with Bias Compensation

In the current implementation of the algorithm, (3.16) is used to model the gyroscope, out of which ω_g which enters the filter is computed, with bias being taken as a constant value $b_g = b_{g0}$. The reason for this is the ease of implementation and the fact that in-run bias drift was found to only be present in a small scale. The value of this constant bias term, usually referred to as *turn on* bias was computed by finding the mean on data collected from the true gyroscope after a 30 min static measurement test. By compensating for it in this manner in the attitude navigation algorithm, results were obtained regarding the performance of the estimator, in a 5 min simulation run, using the first simulation setup (Fig. 4.7 and Fig. 4.8), the second simulation setup (Fig. 4.9 and Fig. 4.10) and the third simulation setup (Fig. 4.9 and Fig. 4.10).

Results for the two latter cases where experimental data are being used (Figs. 4.9 – 4.12) show that the error although being higher has a bounded character close to a limit of 1 degree. Not accounting for bias instability, incorporating a rudimentary modelling instead may be the cause of the higher error, the response although has the properties that one seeks through the sensor fusion implementation.

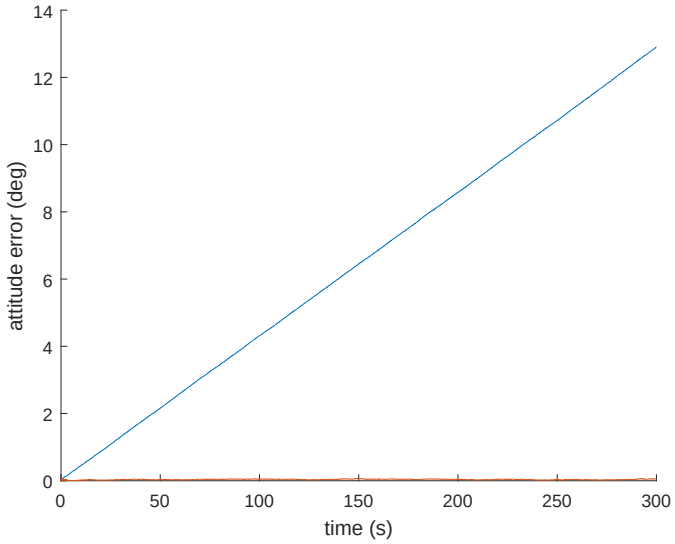


Figure 4.7: The attitude error angle, θ, using the error-state UKF (red curve) stays bounded unlike that of the dead reckoning approach (blue curve), determining the validity of the procedure.

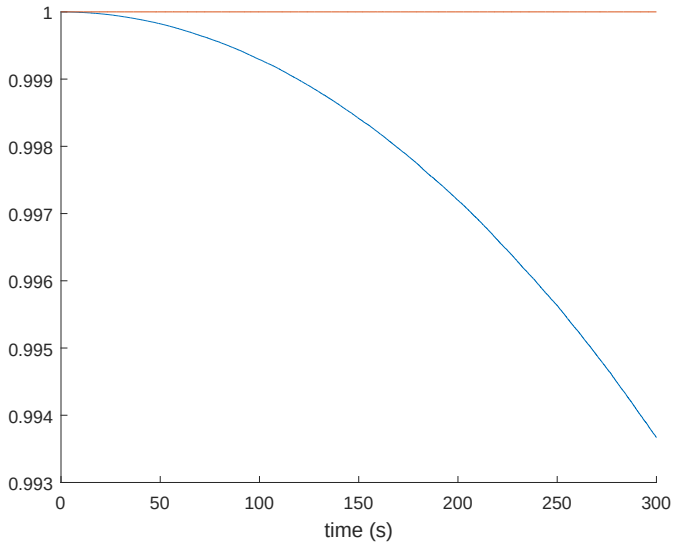


Figure 4.8: The scalar part, δq_w, of the attitude error quaternion, δq, also shows little deviation from 1 in the presence of the filter (red curve) which is not the case without it (blue curve).

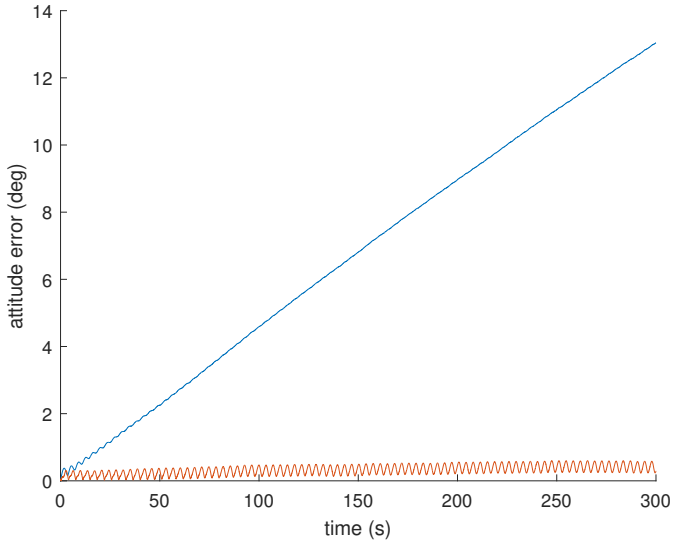


Figure 4.9: The attitude error angle, θ, is presented in a similar fashion to Fig. 4.7 but for the second simulation setup.

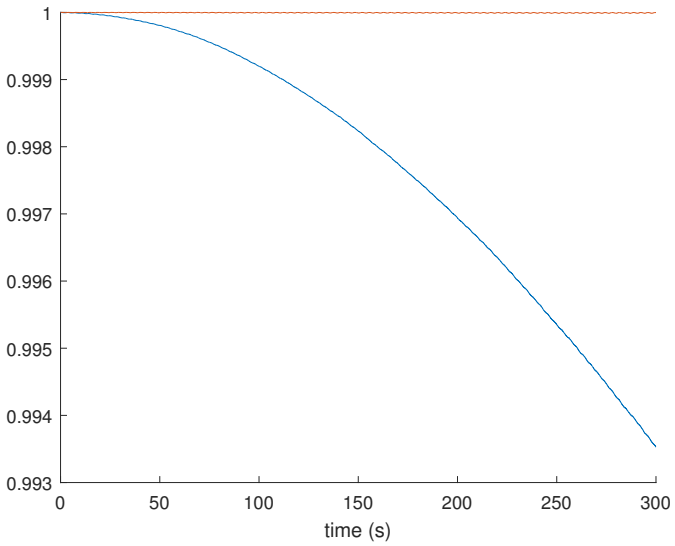


Figure 4.10: The scalar part, δq_w, of the attitude error quaternion, δq, is presented in a similar fashion to Fig. 4.8 but for the second simulation setup.

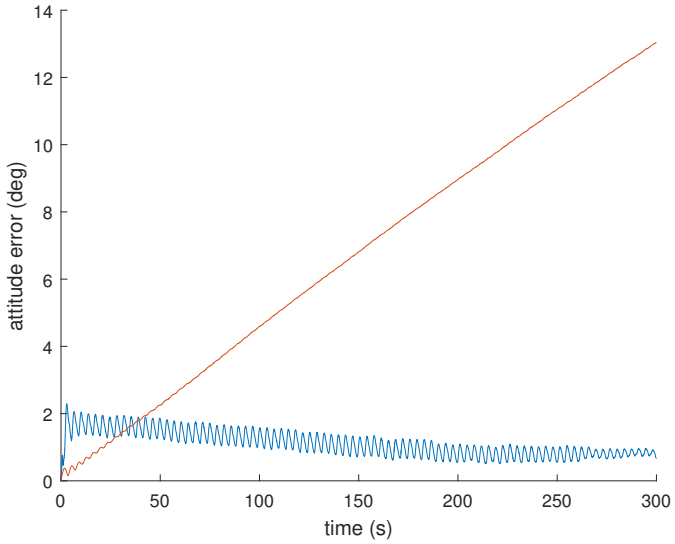


Figure 4.11: The attitude error angle, θ, is presented in a similar fashion to Fig. 4.7 but for the third simulation setup.

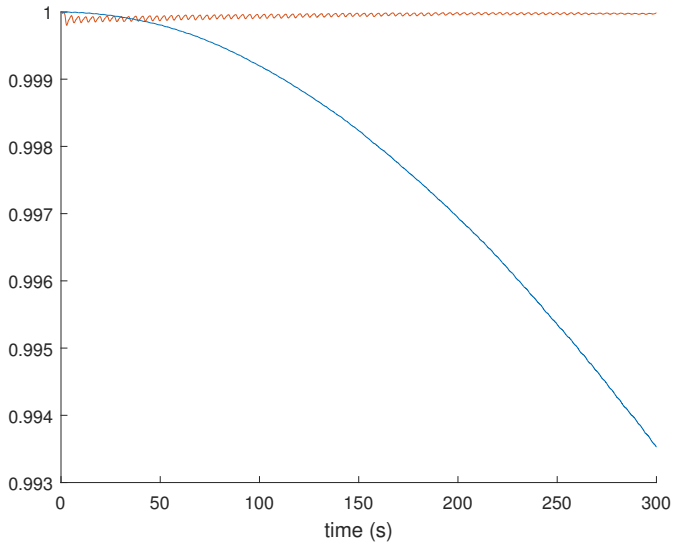


Figure 4.12: The scalar part, δq_w, of the attitude error quaternion, δq, is presented in a similar fashion to Fig. 4.8 but for the third simulation setup.

4.3 Filter Structural Characteristics

At this point it is beneficial to look at the structure of the filter in order to gain deeper understanding of how it works and what is the error tolerance in the initial estimation. Usually in the presence of filters correction of the state trajectory takes place every time the correction step of the filtering algorithm is completed. This becomes visible in the results in the form of the estimated state trajectory following the true one. In the current case, where the magnetometer operates at 1/4th of the frequency of the gyroscope, the correction step takes place every 4th timestep. This behavior, though, is not visible in the evolution of the angle θ as it is not present in the state vector. The filter, as described in chapter 3.4, contains infinitesimal (error) quaternions and it is there where the filter can be seen acting. Fig. 4.13 shows clearly this process.

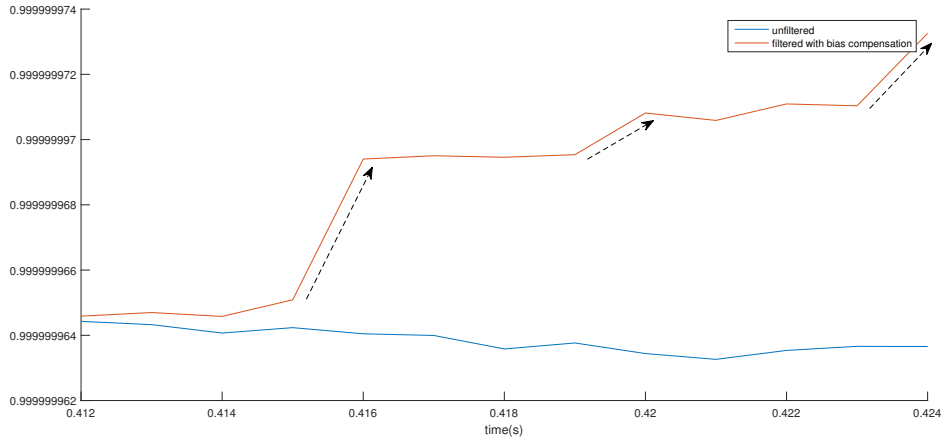


Figure 4.13: The y-axis is the scalar part, δq_w, of the attitude error quaternion, δq. As it is seen the estimated state trajectory using the error-state UKF (red curve) is corrected every 4 timesteps (actions denoted by the black arrows). This is not present in the dead reckoning approach (blue curve).

A way to investigate if the filter is indeed working as expected is to look at the evolution of the variances (Fig. 4.14) of the state vector elements. These should converge denoting that the confidence of the estimation grows over time since the region around the estimated state (where the true state resides) becomes smaller.

Apart from the evolution of the variances, their initial value is also of great importance. This value shows how close the estimated initial orientation is to the real one and was chosen to be 10^{-6}deg^2 . What this practically means is that one can derive the tolerance between true and estimated initial attitude by computing the standard deviation taken from the initial variance, i.e., its square root. In this case this is $\sqrt{10^{-6}} = 10^{-3}$ deg or 1 mdeg. This is on the same order of magnitude of the tolerance taken by [36].

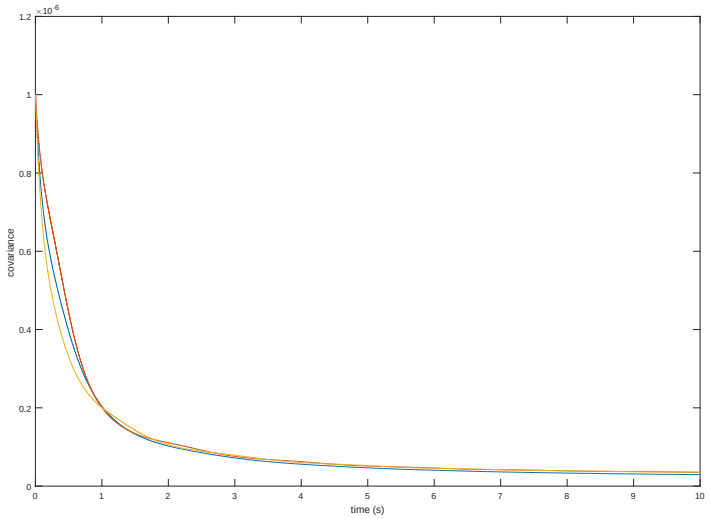


Figure 4.14: Variance evolution over time for the error angle denoting the rotation around the x-axis (blue curve), the y-axis (red curve) and the z-axis (yellow curve).

Chapter 5

Conclusion

Through the duration of this thesis a number of objectives were met. At first a generic UKF algorithm was implemented in MATLAB and tested on a 3DOF robot with performance being compared to an equivalent C++ implementation. The results showed a near identical response for the two, validating the MATLAB design as an accurate one. The SRUKF variant was implemented as well to provide numerical robustness to the filter, with results showing the desirable behavior. The research extended and eventually encompassed other formulations (the UIF and the SRUIF) as well whose implementations were also validated on the same 3DOF robot application.

The gyroscope used in the experiments was modeled in order for the user to be able to generate emulated data. An Allan Variance analysis was done to compare the instrument to its model counterpart to determine important quantities in the stochastic processes used. These quantities were later used in the model whose performance was determined to closely resemble that of the actual gyroscope. The magnetometer was also modelled using datasheet for determining quantities in the stochastic processes present and the comparison between the data obtained by model and the measurement instrument showed equivalent performance.

The object of quaternion-based UKF was thoroughly researched, resulting in the proposal of an extension of the error-state approach previously seen in EKF to the UKF one. In this way no need for special alterations to the filtering ecosystem are required The concept of this quaternion-based error-state UKF was validated using real and emulated gyroscope data and magnetometer data.

The results concerning the attitude navigation algorithm show that the filter is achieving sound operation given a faithful initialisation procedure with error tolerance of about 1 mdeg. The attitude error angle gives a bounded response staying below 1 degree for the 5 min of the expected flight time of a sounding rocket in the emulated data setup. Experimental results showed it to settle below 1 deg after 2.5 min of flight time.

The error-state formulation maintains the genericness of the filtering algorithm, an attribute that should be noted as most quaternion-based UKF demand special modifications to run efficiently. The structure of the algorithm gives the ability for the state

vector to be trivially augmented in order to estimate other interesting quantities, e.g., the position, the velocity etc.

5.1 Future Work

The attitude navigation algorithm can be further refined in the following ways

- The state vector can be augmented with bias, scale factor or other error terms and use a more intricate stochastic process, e.g., a random walk or a Gauss-Markov process for a more robust modelling
- The same should be done for the magnetometer and possible fusion with other sensors, like sun-sensors.
- A C++ equivalent application should be developed and test the algorithm in real time. Considerations that come with it like data acquisition and scheduling should be taken into account.
- The initialization phase should be determined and put into the model. More specifically, how the initial attitude and bias terms will be obtained.
- The numerically robust SRUIF should be thoroughly examined based on the preliminary results shown here.

Bibliography

[1] B. Tong Minh, "Real-time Position and Attitude Determination of the Stratos II Sounding Rocket." `https://repository.tudelft.nl/islandora/object/uuid%3A9ea7cdf8-ae8b-4131-8647-c07794fe4a20`, 2012.

[2] M. D. Shuster, "The generalized Wahba problem," *Journal of the Astronautical Sciences*, vol. 54, no. 2, p. 245, 2006.

[3] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[4] E. Wan and R. Van Der Merwe, "Chapter 7: The unscented kalman filter," *Kalman Filtering and Neural Networks*, pp. 221–280, 2001.

[5] R. Van Der Merwe and E. A. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, vol. 6, pp. 3461–3464, IEEE, 2001.

[6] R. Van Der Merwe, E. A. Wan, S. Julier, *et al.*, "Sigma-point Kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation," in *Proceedings of the AIAA Guidance, Navigation & Control Conference*, pp. 16–19, 2004.

[7] E. Kraft, "A quaternion-based Unscented Kalman Filter for Orientation Tracking," in *Sixth International Conference of Information Fusion, 2003. Proceedings of the*, vol. 1, pp. 47–54, July 2003.

[8] Y.-J. Cheon and J.-H. Kim, "Unscented filtering in a unit quaternion space for spacecraft attitude estimation," in *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pp. 66–71, IEEE, 2007.

[9] M. S. Challa, J. G. Moore, and D. J. Rogers, "A simple attitude unscented kalman filter: Theory and evaluation in a magnetometer-only spacecraft scenario," *IEEE Access*, vol. 4, pp. 1845–1858, 2016.

[10] J. L. Crassidis, "Sigma-point Kalman filtering for integrated GPS and inertial navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 2, pp. 750–756, 2006.

[11] M. Shuster, E. Lefferts, and F. Markley, "Kalman filtering for spacecraft attitude estimation," in *AIAA 20th Aerospace Sciences Meeting, Orlando, Florida*, vol. 232, 1982.

[12] J. Sola, "Quaternion kinematics for the error-state KF." `http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf`, 2017.

[13] G. Bishop and G. Welch, "An introduction to the kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, no. 27599-23175, p. 41, 2001.

[14] H. W. Sorenson, "Least-squares estimation: from Gauss to Kalman," *IEEE spectrum*, vol. 7, no. 7, pp. 63–68, 1970.

[15] C. S. Adam, "Kalman Filtering: A Bayesian Approach," 2010. Princeton University.

[16] H. Durrant-Whyte *et al.*, "Introduction to estimation and the Kalman filter," *Australian Centre for Field Robotics*, vol. 28, no. 3, pp. 65–94, 2001.

[17] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference, Proceedings of the 1995*, vol. 3, pp. 1628–1632, IEEE, 1995.

[18] M. Herb, "kalman." `https://github.com/mherb/kalman`, 2015.

[19] G. Liu, F. Wörgötter, and I. Markelic, "The Square-root Unscented Information Filter for State Estimation and Sensor Fusion.," in *SENSORNETS*, pp. 169–173, 2012.

[20] L. DO Q, "Numerically efficient methods for solving least squares problems." `http://math.uchicago.edu/~may/REU2012/REUPapers/Lee.pdf`, 2012.

[21] D.-J. Lee, "Nonlinear estimation and multiple sensor fusion using unscented information filtering," *IEEE Signal Processing Letters*, vol. 15, pp. 861–864, 2008.

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

[23] `http://easyspin.org/documentation/eulerangles.html`.

[24] J. Chai, "Computer graphics rotation representation and interpolation." `http://slideplayer.com/slide/5157731/`. Texas A&M University.

[25] J. Ferguson, "Calibration of Deterministic IMU Errors." `http://commons.erau.edu/pr-honors-coe/2`, 2015.

[26] D. Gebre-Egziabher, "Design and performance analysis of a low-cost aided dead reckoning navigator," *PhD Thsis, Standford University*, 2004.

[27] "STIM300 IMU specifications." `http://www.sensonor.com/media/91313/ts1524.r8%20datasheet%20stim300.pdf`.

[28] O. J. Woodman, "An introduction to inertial navigation," tech. rep., University of Cambridge, Computer Laboratory, 2007.

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

[30] A. G. Quinchia, G. Falco, E. Falletti, F. Dovis, and C. Ferrer, "A comparison between different error modeling of MEMS applied to GPS/INS integrated systems," *Sensors*, vol. 13, no. 8, pp. 9549–9588, 2013.

[31] W. Flenniken, J. Wall, and D. Bevly, "Characterization of various IMU error sources and the effect on navigation performance," in *Ion Gnss*, pp. 967–978, 2005.

[32] R. Gonzalez, J. I. Giribet, and H. D. Patiño, "NaveGo: a simulation framework for low-cost integrated navigation systems," *Journal of Control Engineering and Applied Informatics*, vol. 17, no. 2, pp. 110–120, 2015.

[33] "Magnetometer calculator." `https://www.ngdc.noaa.gov/geomag-web/`.

[34] C. Gramkow, "On averaging rotations," *Journal of Mathematical Imaging and Vision*, vol. 15, no. 1, pp. 7–16, 2001.

[35] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance Control and Dynamics*, vol. 30, no. 4, p. 1193, 2007.

[36] J. K. Bekkeng, "Prototype Development of a Low-Cost Sounding Rocket Attitude Determination System and an Electric field Instrument," *Article, UiO*, vol. 5, 2007.

TRITA 2017:107
ISSN 1653-5146