Tallinn University of Technology
Faculty of Information Technology

Caroliina Laantee 152996IAPM

# The suitability analysis and simulations of the state estimation algorithm for the TTÜ Mektory nanosatellite

Master's thesis

Supervisor: Martin Rebane
MSc
Lecturer

Tallinn 2017

Tallinna Tehnikaülikool
Infotehnoloogia teaduskond

Caroliina Laantee 152996IAPM

# TTÜ Mektory nanosatelliidi asendimääramise algoritmi sobivusanalüüs ja simulatsioonid

Magistritöö

Juhendaja: Martin Rebane
MSc
Lektor

Tallinn 2017

## Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Caroliina Laantee
05.05.2017

# Abstract

The aim of this thesis was to investigate which kind of algorithm would be most suitable for determining the attitude of the TTÜ Mektory nanosatellite. More specifically, it was necessary that the algorithm would manage filtering out noise from the nanosatellite sensor's measurements and also estimate its attitude.

In order to reach the goal, it was examined if the chosen Kalman filter algorithm would be suitable to be taken into use in the attitude determination system. To get more specific results, two variants of the algorithm were tested and compared to each other by implementing and executing simulations in MATLAB.

As a result of this thesis the chosen algorithm, simulations created in MATLAB and the thesis itself as documentation can be used for further development and research connected to the TTÜ Mektory nanosatellite.

This thesis is written in English and is 61 pages long, including 7 chapters, 13 figures and 3 tables.

## Annotatsioon

## TTÜ Mektory nanosatelliidi asendimääramise algoritmi sobivusanalüüs ja simulatsioon

Käesoleva töö eesmärgiks oli uurida, milline asendimääramise algoritm oleks kõige sobivam, et ennustada TTÜ Mektory nanosatelliidi asendit. Algoritmi puhul oli oluline, et see oleks võimeline nanosatelliidi sensorite mõõtetulemustest müra välja filtreerima kui ka ennustada selle asendit.

Töö käigus uuriti kas välja valitud Kalmani filtri algoritm on kasutatav antud nanosatelliidi asendimääramise süsteemis. Selleks, et saada täpsemaid tulemusi, uuriti kahte erinevat Kalmani filtri edasiarendust, mida testiti ja võrreldi MATLAB-is koostatud simulatsioone kasutades.

Töö tulemusena valiti välja sobiv algoritm ning loodi simulatsioonid MATLAB-is. Töö ise dokumentatsioonina ning välja valitud algoritm ja simulatsioonid on kasutatavad edasiseks arenduseks ja uurimiseks, mis on seotud TTÜ Mektory nanosatelliidiga.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 61 leheküljel, 7 peatükki, 13 joonist, 3 tabelit.

# List of abbreviations and terms

| | |
|---|---|
| **ADCS** | Attitude determination and control system |
| **ADS** | Attitude determination system |
| **AIS** | Automatic Identification System |
| **EKF** | Extended Kalman filter |
| **ISRO** | Indian Space Research Organisation |
| **P** | Error covariance |
| **R** | Measurement variance |
| **TTÜ** | Tallinn University of Technology |
| **UKF** | Unscented Kalman filter |
| **UT** | Unscented Transformation |

# Contents

# List of Tables

# List of Figures

# Listings

# 1 Introduction

As written on the project's homepage [22] TTÜ - Mektory Nanosatellite is an international, interdisciplinary and university wide program that is being carried out by students and professors from different nations. There is close cooperation with several national and international partners who come from a wide range of industries. The goals of this program is to put together a student nanosatellite and successfully launch it to the orbit. Also it must be possible to operate the satellite from the ground station. The building and testing processes are expected to take place between 2016 and 2017 and the launch is expected to happen in 2018.

The nanosatellite will be built according to CubeSat standards [1] which means that it has to be cube-shaped, with 10 cm sides and it has to weight below 1.33 kilograms. As brought out by Peeter Org [16], these kinds of satellites are usually launched as secondary payloads on launch vehicles or put in orbit by deployers on the International Space Station – the TTÜ nanosatellite will be launched as a secondary payload to a larger satellite.

The main goal of this thesis is to investigate which kind of algorithm would be most suitable for determining the attitude of the TTÜ nanosatellite. More specifically it is necessary to filter out the noise from the measurements that the sensors are providing and also estimate the attitude. The attitude determination software is an important part of the ADCS because in order to be able to control the satellite we first need to find out the current attitude of the object. To be able to suggest and test out an algorithm, it is necessary to research which algorithms have been successfully used before and what are the characteristics. It is important to look for a lightweight solution in means of power consumption because the attitude determination algorithm will be running on the nanosatellite board. Also it is needed to keep in mind the set of sensors that the TTÜ nanosatellite has been equipped with because they will be providing the output measurements that have to be filtered by using the suggested algorithm.

The first part of this thesis points out the requirements of the TTÜ nanosatellite and examine if the Kalman algorithm would be suitable to be taken into use in the system. Furthermore, in order to assure the reliability of the Kalman filter, brief descriptions of three successful previous CubeSat missions that used the Kalman algorithm will be provided. In the end of the first part a description of a basic Kalman algorithm process is presented.

The second part of this work provides information about attaining and presenting attitude by mainly focusing on how the raw measurements can be plugged into a previously introduced filtering and attitude estimation algorithm. Additionally, there are descriptions of each of the TTÜ nanosatellite sensors.

In order to get more specific results, the chosen algorithms will be tested out and compared to each other by implementing and executing simulations done in MATLAB. The results of the simulations will also be used to verify that the algorithms work properly and are able to handle the task. The data used for the simulations will be generated test data and also actual measurements from the satellite's gyroscope. Finally, the results will be analyzed and the performances of the algorithms are validated. In the end, based on the conclusions that will be made, one of the algorithms will be suggested to taken into use which can be used directly to be later converted into C code.

It is very important that everything would firstly be implemented in MATLAB because the simulations can be easily tested on a computer where everything can still be tuned, changed and tested. When all of the ADCS simulations in MATLAB, including the algorithm that will be analyzed in this study, have been linked together, tested and verified that they are stable on a computer, then they can be converted into C code.

In conclusion, the outcomes of this thesis will be an algorithm chosen and tested out for later development, simulations in MATLAB and also the thesis itself as documentation and basis for further research.

# 2 Kalman filter and its suitability

Dan Simon has pointed out in his article that signal filtering is important in many situations in engineering and embedded systems [26]. Signals are very often corrupted with noise but it is possible to remove it when using a relevant filtering algorithm. As a result, only the useful information will be outputted. For example the Kalman filter was originally developed for being used in spacecraft navigation but is actually very useful also for other applications. Mainly it is used to estimate system states that can not be attained directly or accurately.

The Kalman filter has become a fundamental tool for solving wide range of problems connected to estimation. Its first publicly known application was made at NASA Ames Research Center in the beginning of 1960s - it was used during feasibility studies for circumlinear navigation and control of the Apollo space capsule [24]. Later on, Kalman filter has also been used in many other space related projects including different student nanosatellites similar to TTÜ nanosatellite.

## 2.1 Requirements

There are several reasons why a signal filtering algorithm must be used in the attitude determination system of the TTÜ nanosatellite. These points are also important when choosing a suitable solution and some of them are pointed out here:

- The satellite will be equipped with several sensors and all the sensor measurements need to be included in the filtering algorithm
- It is important that the estimated values are within the precision of at least 2 degrees [16]

- The filtering algorithm should be light on memory and power consumption when processing the values
- It might happen at times that the system has to be reset. So it is important that the algorithm is able to quickly restart and continue working as efficiently as before.

Relying on the information provided by G. Welch and G. Bishop then the Kalman filter has many following advantages [29] that also satisfy the needs of the current system in hand:

- The filter is able to process all kinds of measurements, no matter of their precision.
- It will estimate the current values by using the knowledge about the system and measurement device dynamics.
- It will take into account measurement errors, uncertainty in the dynamics models and information about initial conditions.
- In the process of the Kalman filter algorithm, the previous data is not being kept in memory and the data is being recalculated every time a new measurement comes in.

This is why the algorithm is very suitable when running it in the on-board system of the nanosatellite because it does not expect a lot of memory and power. Also the raw measurements coming from the satellite will certainly be very noisy, inaccurate and in some cases, even missing. There are many reasons why the measurements are inaccurate, most of them connected to the shortcomings of the different sensors. Since almost every nanosatellite, including TTÜ nanosatellite, has its own combination of sensors then it is important to use an algorithm that can take into account the different types of systems - and the Kalman filter is able to do exactly that.

The Kalman filter could solely be used for signal filtering but since it can also handle attitude estimation then it beats some of its counterparts. It is more reasonable to use one algorithm that can handle both cases than to use different solutions for signal filtering and attitude estimation. For example, opposite to the Kalman algorithm, the TRIAD algorithm is not able to provide an optimal attitude estimate and also it is not able to take into account more than two measurements [9].

## 2.2 Previous CubeSat systems

Before the TTÜ Mektory nanosatellite, several other CubeSats have included the Kalman filtering algorithm in their ADCS systems. This paragraph will point out some similar nanosatellites and give a more detailed overview of three of them that have already successfully finished their missions and had enough mission data available. Some similar nanosatellite projects would include:

- ESTCube-1 - the first Estonian satellite, launched May 7, 2013 [2]
- Student-developed nanosatellites from Aalborg University, Denmark:
  - AAUSat2 - launched June 30, 2003 [3]
  - AAUSat3 - launched February 25, 2013 [4]
  - AAUSat4 - launched April 25, 2016 [5]
- Norwegian AIS (Automatic Identification System) satellites:
  - AISSat-1 - launched July 12, 2010 [6]
  - AISSat-2 - launched July 8, 2014 [7]

### 2.2.1 ESTCube-1

ESTCube-1 was an Estonian student CubeSat project from the University of Tartu that was launched aboard the European Space Agency's Vega launcher in 2013 [2].

In the first year, while in the orbit, the satellite was mostly focused on taking photos and downloading data and in 2014, the team started working towards to executing the E-sail experiment.

In the ESTCube-1 ADS (Attitude Determination System) the following set of sensors were used to gather measurements: Sun sensors, magnetometers and gyroscopic sensors [25]. For attitude estimation an Unscented Kalman filter was chosen that originally was developed for the AAUSat3 nanosatellite.

After an unsuccessful experiment of trying to unreel a small amount of the E-sail tether it was noticed that the solar panels were producing less energy that was needed for the satellite and the only thing to rely on were batteries that were already close to being empty [2]. On May 19, 2015 the batteries finally got empty and the satellite stopped giving out any signals. In the end the project was considered successful because it fulfilled its main goal: the satellite captured 300 photographs from space according to EstCube-1 homepage and some of them can be seen on Figure 2.1.

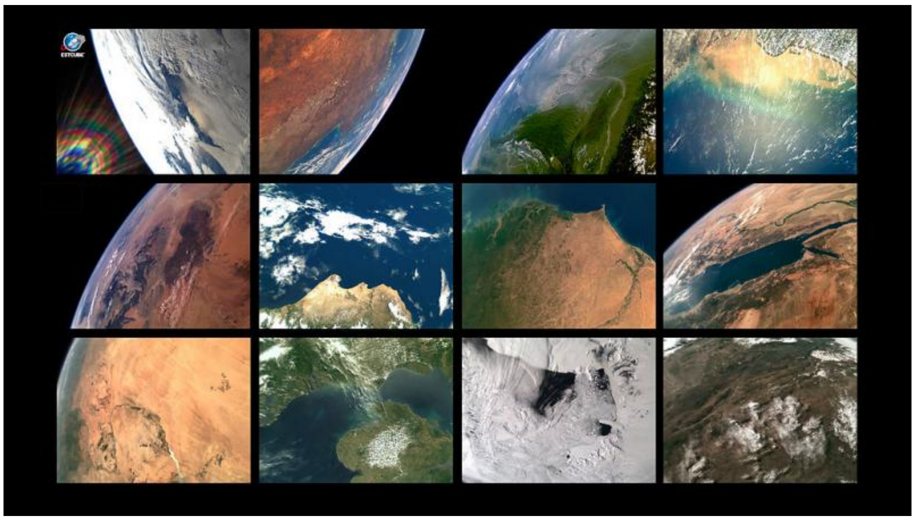

Figure 2.1: Some images from the EstCube-1 mission, taken from EstCube-1 homepage

### 2.2.2 AAUSat3

AAUSat3 was the third CubeSat created by the students from the Department of Electronic Systems at Aalborg University, Denmark [4]. The satellite was a successor to AAUSAT-II that was launched in April 2008. AAUSat3 was launched February 25, 2013 as a secondary payload on on the PSLV-C20 launcher of the Indian Space Research Organisation (ISRO). The goal of the mission was to investigate the quality of ship monitoring from space. In the attitude determination system two different types of sensors were used: magnetometers and gyroscopes. As previously mentioned under ESTCube-1, an Unscented Kalman filter was used for attitude determination.

At the 100th day in space during two evening passes, more than 2400 ships were downloaded and the results can be seen on Figure 2.2 taken from the AAUSat3 homepage [4].

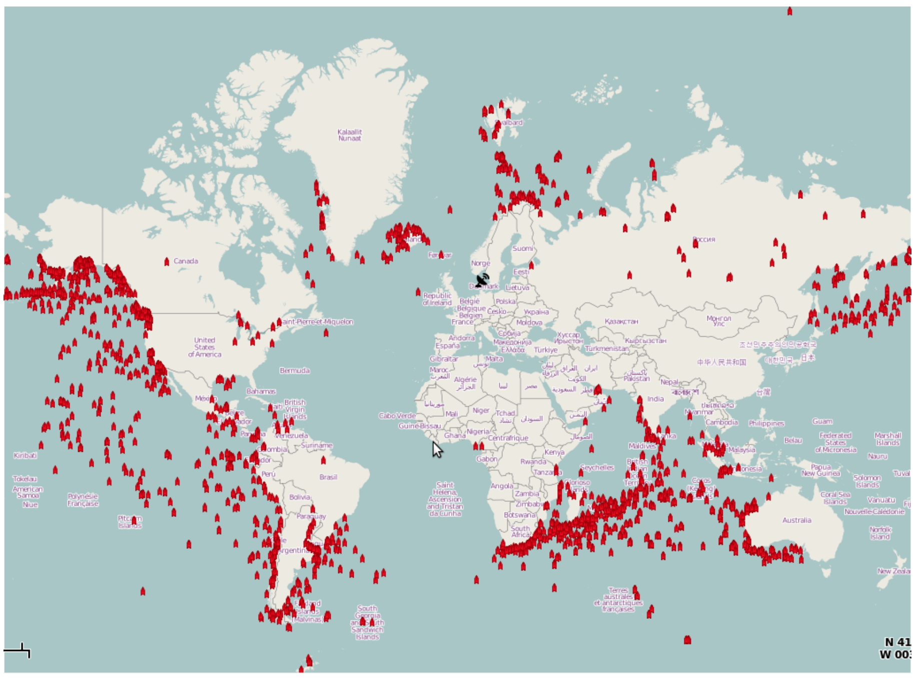

Figure 2.2: Plot from AAUSat3 homepage showing more than 2400 downloaded ships

The end of the mission was declared on October 1, 2014 because of battery problems - power production was constantly decreasing [4].

## 2.2.3 AISSat-1

AISSat-1 was a Norwegian nanosatellite which was launched as a secondary payload on July 12, 2010 on a PSLV launcher of ISRO (PSLV-C-15) from the Satish Dhawan

Space Center at Sriharikota [6]. The satellite was constructed on behalf of the government of Norway by University of Toronto, Institute for Aerospace Studies/Space Flight Laboratory in Toronto, Canada. The purpose of the satellite was to make a maritime situational picture available to Norwegian authorities. In the attitude determination system three types of sensors were used: Sun sensors, magnetometer and gyroscopes. For the attitude estimation an Extended Kalman filter was used.

According to an online article published by the Norwegian Defence Research Establishment (FFI) [8] the launch of AISSat-1 was a complete success and already the first AIS messages showed that the mission improves the Maritime Situational Awareness in Norwegian areas. The maritime traffic situation in the High North was already mapped within a 10 minute pass of the satellite. On Figure 2.3 the first data gathered by AISSat-1 can be seen.

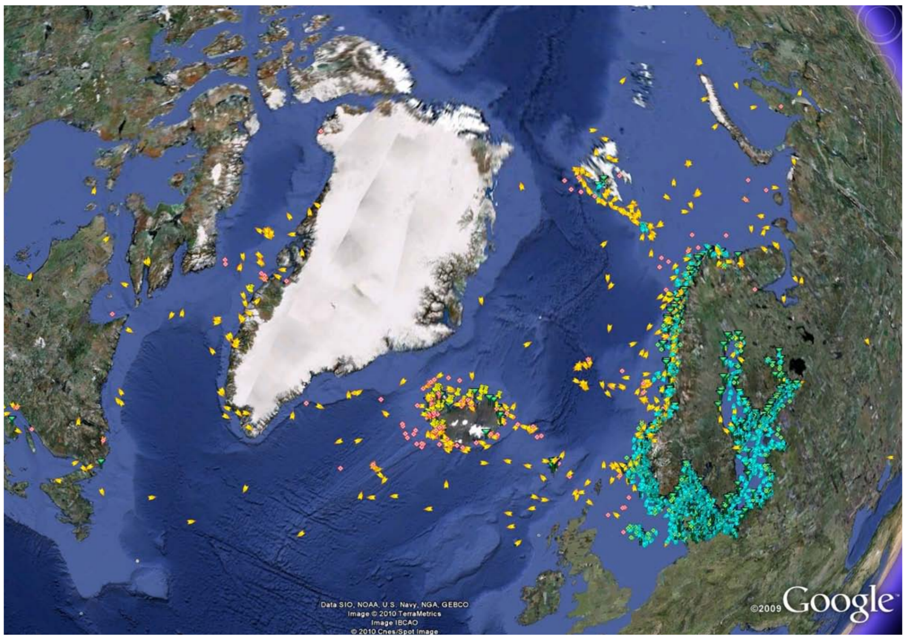

Figure 2.3: Image of first AIS data gathered with AISSat-1 taken from the FFI article. Yellow and pink symbols show the contribution of AIS data from AISSat-1.

## 2.3 Kalman filter algorithm

In the following section, the main idea of the Kalman filter will be presented.

The idea of the Kalman filter is to estimate the state of a certain process that can be expressed by a linear stochastic equation [29]:

$$x_k = A_{x_{k-1}+_{w_{k-1}}} \tag{2.1}$$

with a measurement

$$z_k = H_{x_k+_{v_k}} \tag{2.2}$$

where the variables $w_k$ and $v_k$ are respectively the process and measurement noise. They are assumed to be white noises that are independent of each other and have Gaussian (normal probability) distribution. The noises in the state model of the Kalman filter are represented through the two covariance matrices $Q$ and $R$. Noise is a value that can not be predicted but only estimated statistically and that is why it is necessary to use statistics when expressing it [13].

The matrix $A$ in equation (2.1) is a transition matrix that relates the state at a previous time step $k-1$ to the state at the current step $k$ [29] and represents how the system changes along time since it contains the equations of motion of the system [13].

The Kalman filtering algorithm process could be divided into four parts [13]:

1. Predicting state and error covariances
2. Computing the Kalman gain
3. Computing the estimate
4. Computing the error covariance

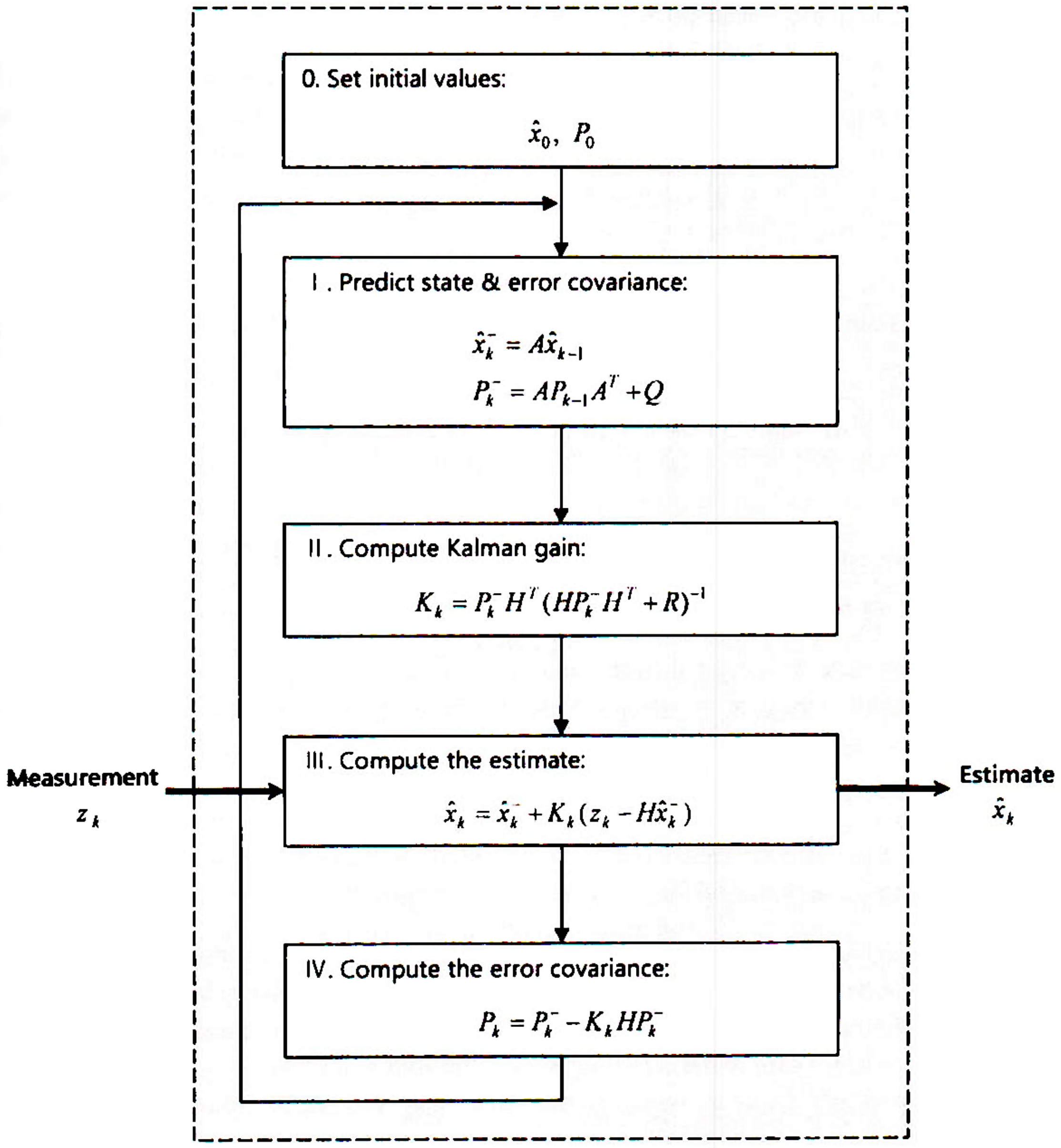
0. Set initial values:
$\hat{x}_0, P_0$
I . Predict state & error covariance:
$\hat{x}_k^- = A\hat{x}_{k-1}$
$P_k^- = AP_{k-1}A^T + Q$
II . Compute Kalman gain:
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
III . Compute the estimate:
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
IV. Compute the error covariance:
$P_k = P_k^- - K_k HP_k^-$
Measurement
$z_k$
Estimate
$\hat{x}_k$

Figure 2.4: Kalman filter algorithm steps by Phil Kim [13]

Basically the filter receives one input which is the measurement $z_k$ and returns one output which is the estimate $\hat{x_k}$. In the first step a predicted state estimate and error covariance will be calculated (this can also be called a prediction step). In step two, the Kalman gain will be computed and it also takes into account the previously calculated error covariance. In addition it takes into account the matrices $H$ and $R$, $H$ representing the relationship between the measurement and state variable (it defines how each state variable is mapped into the measurement). In the third step an estimate of the state will be calculated by taking into account the previous state, previously calculated Kalman gain and a measurement that comes in as an input. The Kalman gain influences the contribution of the measurement taking into account the initially set matrices $R$ and $Q$ which means that it is important how to initialize them - this will be looked at more precisely in a later chapter. Finally, in the last the step an error covariance $P_k$ is calculated which shows how accurate the estimate is. The same process is depicted on Figure 2.4. The described steps can also be classified as the time update ("predict") and measurement update ("correct") steps. The previously mentioned first step would be the "prediction" step and steps 2-4 would be the "correction" steps.

In conclusion, the variables mentioned before are as follows:

- $\hat{x_k}$ - estimated state, vector
- $\hat{x_k^-}$ - predicted state, vector
- $z_k$ - measurement, vector
- $A$ - state transition matrix
- $H$ - state-to-measurement matrix
- $w_k$ - state transition noise
- $v_k$ - measurement noise
- $P_k$ - estimate of the error covariance, matrix
- $P_k^-$ - prediction of the error covariance, matrix
- $K_k$ - Kalman gain

The variables $A$, $H$, $Q$ and $R$ should be initialized before the start of the algorithm, $z_k$ is the actual measurement (input), $\hat{x_k}$ is the estimate (output) and the remaining variables are used for internal calculations.

## 2.4 Conclusion

Since the Kalman filter has proved to be the most popular signal filtering and attitude estimation algorithm and has a good history of successful usages when determining the attitudes of spacecrafts then there is no need focusing on alternative filtering algorithms. Also, it proves to be able to handle all the signal filtering problems that are also connected to the TTÜ nanosatellite.

Unfortunately, the previously presented Kalman filter only applies to linear systems. When dealing with satellite attitude determination, a linear Kalman filter is not applicable. A nonlinear version of the Kalman filter must be used when dealing with a three-dimensional systems. Therefore, in the fourth chapter, two nonlinear Kalman filter variants will be further analysed to find out if they can be modified to suit the needs of the TTÜ nanosatellite. The Extended Kalman filter (EKF) and Unscented Kalman filter (UKF) will be further investigated and compared.

Before the comparison of the two nonlinear versions, the following paragraph will include a description of the attitude determination, representing the attitude and the types of sensors in the TTÜ nanosatellite. It is important because it is needed to know what are the inputs for the algorithm.

# 3 Attitude determination

In order to use a filtering and attitude estimation algorithm that was previously introduced in the previous chapter, it is necessary to understand what is received as inputs. The following chapter will give an overview of the main idea of attitude determination, how it can be attained and plugged into the attitude estimation process. Furthermore, it includes the descriptions of the attitude sensors that the TTÜ nanosatellite has been equipped with.

## 3.1 Attitude determination

Attitude is a three-dimensional orientation of an object compared to a specific reference frame [17]. When it is needed to control an object then first it is necessary to determine its attitude because otherwise it would be impossible to know how or what way it should be controlled. In order to achieve this, attitude systems use sensors, actuators, avionics, algorithms, software, and ground support equipment to determine and control the object's attitude. The whole process consists of getting inputs from the set of sensors a vehicle might have and combining it with the knowledge of spacecraft's dynamics. This will result in an attitude state as a function of time. Luckily nowadays the available microprocessors are powerful enough to be able to run attitude algorithms on-board. For the TTÜ nanosatellite a STM32F microcontroller will be used [16] and the attitude determination algorithm is also expected to run on it.

Knowing the precise attitude of the satellite makes it possible to do accurate calculations in order to find out how much and in what way it is needed to rotate the satellite. The results of the calculations will be the amounts of thrust and torque that the satellite's actuators need to apply. Since the main goal of the TTÜ Mek-

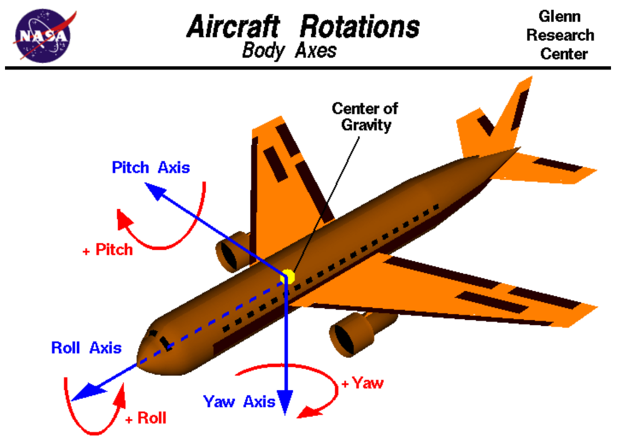

Figure 3.1: Representation of yaw, pitch and roll angles [18]

tory nanosatellite is to take pictures then there will be many commands related to rotating the satellite.

The initial result of attitude determination can be attained by using the data from sensors that relate information about external references to the orientation of the spacecraft [17]. The external references might be the stars (star tracker), the Sun (sun sensors), the Earth (Earth sensor, horizon sensor, magnetometer), or other celestial bodies. A single sensor always has some kind of noise or other drawbacks and that is why it is essential to equip the satellite with several different sensors to weigh out the error of one sensor with another's. The filtering algorithm also takes care of combining the different types of sensors' readings.

Determining and controlling the attitude of a satellite should be done in all three dimensions [18]. In flight, any aircraft or spacecraft is rotating about its own center of gravity which is the point of the average location of the mass of the object. Through the same point (center of gravity) a three dimensional coordinate system can be imagined where each axis of that system is perpendicular to two other axes. Finally, the orientation of the satellite can be defined by how much it is rotated along each axis.

The most common way for representing the attitude would be using a set of three Euler angles [23]. These have become popular because they are easy to understand

and easy to use. Some sets of Euler angles are being so widely used that they have gotten the names *yaw*, *pitch* and *roll*. The representation of the three angles can be seen on Figure 3.1 [18]. They are typically denoted respectively: $\phi$ as yaw, $\theta$ as pitch and $\psi$ as roll.

An alternative orientation representation format is called quaternions. A quaternion consists of $x$, $y$ and $z$ components that represent the axises about which the rotations occur but it also has a fourth element, $w$, that represents the amount of rotation that occurs around one axis [19]. With these four elements it is possible to build a matrix that will represent the rotations. The quaternions are not easy to understand intuitively but they are compact, do not have a problem with gimbal lock and can be interpolated without troubles [20].

If necessary, it is always possible to convert from Euler angles format to quaternions and vice versa. There are even existing MATLAB functions in order to do so, for example *eul2quat* [21], a function that takes in an Euler angles matrix and outputs a quaternion matrix. Since the Euler angles are a more common, straightforward and intuitive way of representing attitude then, hereinafter, in this thesis, it will be considered that the attitude is represented through Euler angles.

## 3.2 Attitude determination sensors in TTÜ satellite

The TTÜ nanosatellite will be equipped with three different kinds of sensors: sun sensors, gyroscopes and magnetometers [16]. To implement the Kalman filtering for attitude estimation it is needed to retrieve the angular velocities from the gyroscope and three Euler angles (yaw, pitch and roll) from the sun sensors and magnetometer for error calibration. To get the best result, we need to have several sensors which would complement one another since each sensor has its faults. In the following paragraphs the descriptions of the three sensors will be provided.

On Figure 3.2 it can be seen that the inputs for the attitude determination module come from sun sensors, magnetometers and gyroscopes. That is also the module where the Kalman filtering algorithm will be integrated. The output of the attitude determination module is inputed to the current calculation algorithm which is responsible for controlling the satellite.

### 3.2.1 Sun sensors

Sun sensor is an optical attitude sensor that can be used to determine the orientation of a spacecraft [10]. It outputs the spacecraft orientation relative to the sun by determining how the sun vector positions in comparison to the satellite's coordinate system. When the satellite changes its position then the sun vector in the satellite orbit coordinate system also differs. The sun vector is mainly influenced by the yaw angle so the vector could be used to determine the yaw attitude of the satellite. It should be kept in mind that the sun sensors can only function in case the satellite is in the sun-lit phase of the orbit in all other cases it is not able to observe the Sun [11]. This has to be taken into account when choosing the attitude determination algorithm or choosing the set of sensors for a satellite.

More specifically a sun sensor has a thin slit on top of a rectangular chamber where in the bottom part there are light-sensitive cells [12]. Next, an image of a thin line will be casted on the cells and the distance of the projected image and the centerline can be measured. Finally, the refraction angle can be measured by using the height of the chamber.

### 3.2.2 Gyroscope

Gyroscope is an inertial navigation sensor that returns a measurement with respect to the inertial frame [13]. The advantage of a gyroscope is that it can provide the angular displacement and/or angular velocity of the satellite's roll, pitch and yaw angles directly. However, gyroscopes have an error due to drifting, meaning that their measurement error increases with time [14]. But the measurements can

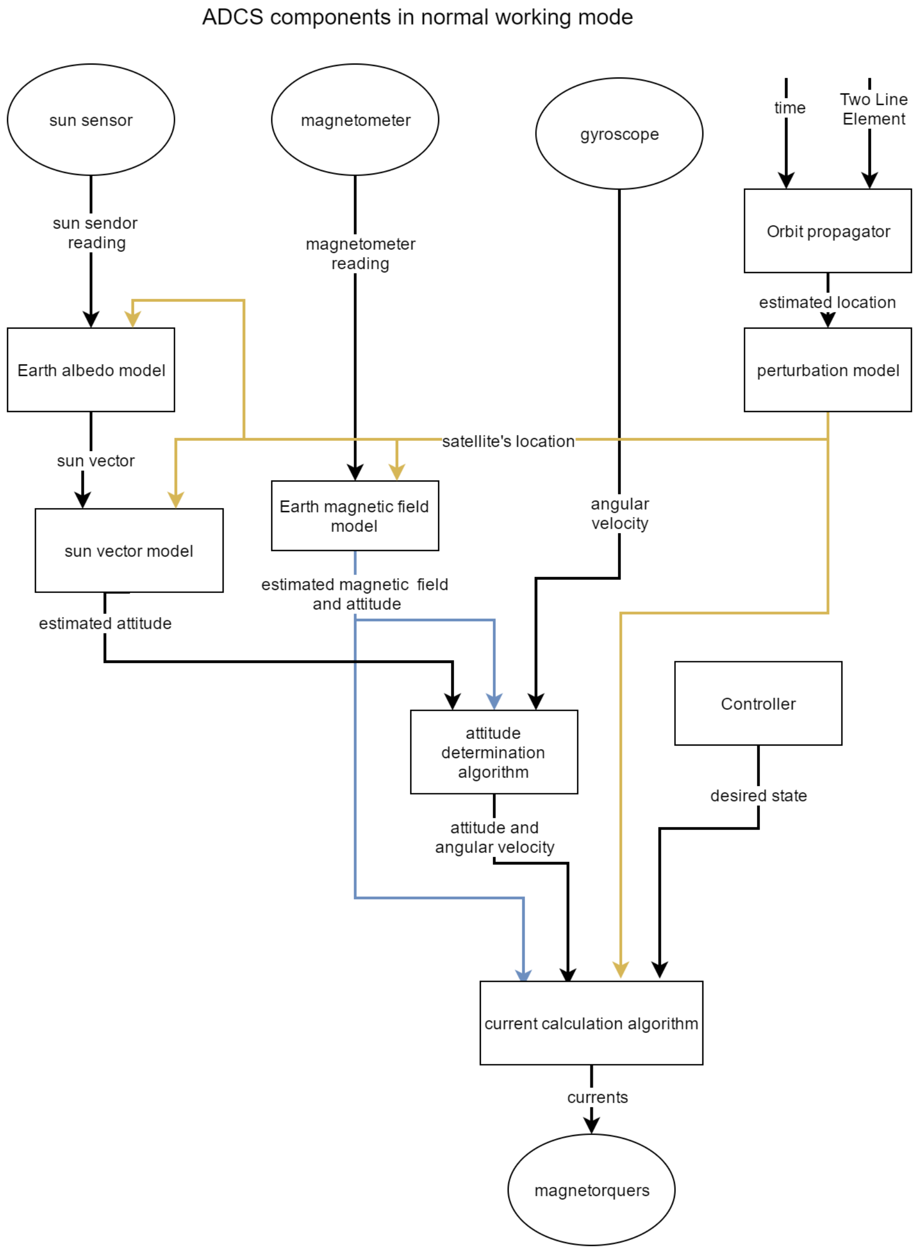

Figure 3.2: Org's drawing: ADCS components in normal working mode [16]

be corrected by having measurements from different sensors and applying sensor fusion. It is not possible to get the Euler angles by integrating the angular velocities measured from the gyroscope because they are not the rate of change in the Euler angles but the angular rates of the satellite [13]. So the measurement from the gyroscope should be first transformed into the rate of change in the Euler angles and then integrate them. The relationship between Euler angles and angular velocities is well known in kinematics. The equation can be seen in (3.1) [13].

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi/cos\theta & cos\phi/cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.1}$$

The attitude can be obtained by applying the angular velocities measured from the gyroscope to this formula and integrating the result with respect to time. When applying the Kalman filtering the angular velocities from the gyroscope are needed as inputs. The previously mentioned steps for extracting the Euler angles will be a part of the algorithm's implementation.

### 3.2.3 Magnetometer

Magnetometers measure the projections of acceleration generated by gravity and the magnetic field of the Earth on the aerial vehicle [15]. These projections are used to compute the angular position of the vehicle, which is described by Euler angles.

In this work, it is assumed that by combining the measurements from the magnetometers and sun sensors, all three Euler angles can be calculated and used as inputs for the algorithm. Calculating the Euler angles from the sun sensors and magnetometers was not in the scope of this thesis and needs to be included in future work. If it proves to be impossible, then it will have to be taken into account and the algorithms need to be altered to adapt with the inputs.

## 3.3 Conclusion

Inputs of the attitude determination module are therefore expected to be the attitude estimated by the measurements from the sun sensors and magnetometer, angular velocities from the gyroscope. Because the Sun is not visible at all times, the module needs to be able to perform its tasks with data from the magnetic field model and gyroscopes only [16]. In conclusion, it is important for the filtering algorithm that we would get three Euler angles (yaw, pitch, roll) and angular velocities as inputs.

# 4 Analysis of two non-linear Kalman filter implementations

As mentioned in the end of the second paragraph, in order to estimate the attitude of the satellite, a non-linear version of the Kalman filter should be used. There are two possible options that will be analysed in this chapter:

- The extended Kalman filter (EKF)
- The unscented Kalman filter (UKF)

The EKF is based on linearizing the system's dynamics but it continues to endure some issues connected to the linearization process [28]. The UKF skips the linearization step and introduces the use of setting sample points instead.

For comparing the two Kalman filter variants, code examples implemented in MATLAB [13] by Phil Kim were used and modified to suit the needs of the project in hand. The mentioned implementations of EKF and UKF were chosen for further comparison and simulations for the following reasons:

- The implementations were created in MATLAB and it was not necessary to convert them in order to conduct simulations
- The implementations were easily readable and had documentation
- Both implementations were created with similar style which made comparing them more convenient
- Both implementations already took into account the usage of Euler angles and angular velocities

- Both implementations took into account sensor fusion in order to merge measurements from several different sensors

In the following two paragraphs the algorithms will be used for describing the extended Kalman filter and the unscented Kalman filter to find out how different they are from each other. It will be looked at if one of the implementations is significantly more complex than the other or not. Since MATLAB code can not be used directly in the satellite system due to being substantially slower than C code then the complexity of converting the code is important. Altogether, the main reason for comparing the two implementations is to ascertain if one of them is more suitable for the current nanosatellite system.

Furthermore, the description of the changes made in the algorithms will be provided. Some modifications were made in order to use the specified inputs and that the algorithms could be later directly used in the satellite's attitude determination and control system. The descriptions in the following paragraphs can also be taken as documentation when starting to convert the MATLAB code.

## 4.1 EKF

The extended Kalman filter considers a nonlinear system model which is the following [13]:

$$x_{k+1} = f(x_k) + w_k \tag{4.1}$$

$$z_k = h(x_k) + v_k \tag{4.2}$$

The difference with the linear model is that the state variable is not separable from the coefficients and that the linear matrix has changed into a nonlinear function. To be able to implement the extended Kalman filter, the Jacobian of the system model must be known. Usually it is not difficult to get the derivatives but in complex systems there are high chances for errors in the process.

First of all we will be expecting as inputs three angular velocities from a gyroscope and three Euler angles calculated from sun sensors and magnetometer readings. So the function will be called as follows:

```
[yaw pitch roll] =
EulerEKF([yaw_a pitch_a roll_a]', [p q r], dt);
```

Listing 4.1: EKF initialization

The function's parameters are the Euler angles and three angular velocities plus the rate of change which can be initialized as needed. The output is three filtered Euler angles. Initially it only had two Euler angles instead of three but for the satellite we will be needing to take into account all three to get more accurate results. After calling the method the actual algorithm will be initialized. Respectively, the matrices $R$ and $H$ had to be changed accordingly from 2x2 matrices to 3x3 matrices since the state consists of three angles. They were initialized as follows:

```
  H = [ 1 0 0;
        0 1 0;
        0 0 1 ];


  R = [  0.01  0  0;
         0  0.01  0;
         0  0  0.01 ];
```

Listing 4.2: EKF state-to-measurement and measurement variance matrix

The R matrix being the measurement error covariance matrix. When changing the values in the matrix it also influences how fast the filter responds to changes in the measurements [29]. The measurement variance was fixed as $R = (0.1)^2 = 0.01$ for running the first simulation, it will be further explained later. Q matrix is the process noise matrix. When decreasing Q, the filter will be less affected by the measurements which is the opposite of R [13]. The remaining matrices were initialized as follows:

```
Q = [ 0.00001 0 0;
      0 0.00001 0;
      0   0  0.00001 ];

x = [20 20 20]';

P = 1*eye(3);
```

Listing 4.3: EKF process noise, state and error covariance matrices

The following code is the main part of the algorithm. First, it calculates the Jacobians to get the transition matrix $A$. The variable $xp$ corresponds to the predicted state variable which is returned by the function $fx$. Next, the predicted error covariance, Kalman gain, estimate and the estimated error covariance are calculated.

```
A = Ajacob(x, rates, dt);
xp = fx(x, rates, dt);
Pp = A*P*A' + Q;
K = Pp*H'*inv(H*Pp*H' + R);
x = xp + K*(z - H*xp);
P = Pp - K*H*Pp;
```

Listing 4.4: EKF main part

Finally, the new estimated state can be extracted from the estimate matrix:

```
phi   = x(1);
theta = x(2);
psi   = x(3);
```

Listing 4.5: EKF new state

## 4.2 UKF

The unscented Kalman filter is a substitute for the extended Kalman filter. In the UKF the linearization process is eliminated which also discards the problem of divergence that could happen in the process of obtaining linear models through the calculation of the Jacobians [13]. On the downside, the UKF is harder to understand and implement because it uses the unscented transformation algorithm. Instead of linearizing with the use of Jacobians, the UKF uses a rational deterministic sampling approach to retrieve the mean and covariance estimates with a set of sample points [14]. The function is applied to each point to produce a cloud of transformed points and the statistics of the points can then be calculated to form an estimate of the mean and covariance.

The system model for the UKF remains the same as it was in the previous, EKF algorithm. The algorithm was modified similarly to the EKF algorithm. It is needed to take into account the same inputs as for the EKF: three Euler angles and angular velocities. The algorithm can be called similarly:

```
[yaw pitch roll] =
EulerUKF([yaw_a pitch_a roll_a]', [p q r], dt);
```

Listing 4.6: UKF initialization

Also in this case, the error covariance matrix $R$ had to be changed from a 2x2 matrix to a 3x3 matrix and was initialized as follows:

```
R = [  0.01  0  0;
         0  0.01  0;
         0  0  0.01 ];
```

Listing 4.7: UKF error covariance matrix

The initialization of the remaining variables looked like this:

```
Q = [ 0.00001 0 0;
```

```
        0 0.00001 0;
        0   0   0.00001 ];
  x = [20 20 20]';
  P = 1*eye(3);
```

Listing 4.8: UKF process noise, state and error covariance matrices

Following is the main part of the algorithm. The function *SigmaPoints* computes the sigma points and weights for the previous estimate and error covariance is called.

```
[Xi W] = SigmaPoints(x, P, 0);
```

Next, the sigma points are applied to $f(x)$ in order to calculate new sigma points. Following that, the mean *xp* and error covariance *Pp* are calculated, utilizing the function *UT* (unscented transformation). The same function is also used to calculate the mean and covariance of $h(x)$.

```
fXi = zeros(n, 2*n+1);
for k = 1:2*n+1
  fXi(:, k) = fx(Xi(:,k), rates, dt);
end

[xp Pp] = UT(fXi, W, Q);

hXi = zeros(m, 2*n+1);
for k = 1:2*n+1
  hXi(:, k) = hx(fXi(:,k));
end

[zp Pz] = UT(hXi, W, R);

Pxz = zeros(n, m);
```

```
for k = 1:2*n+1
  Pxz = Pxz + W(k)*(fXi(:,k) - xp)*(hXi(:,k) - zp)';
end
```

Listing 4.9: UKF main part

Finally, the Kalman gain, estimate of the state and error covariance are computed.

```
K = Pxz*inv(Pz);
x = xp + K*(z - zp);
P = Pp - K*Pz*K';
```

Listing 4.10: UKF calculating Kalman gain, state estimate and error covariance

## 4.3 Conclusions

In conclusion, by just looking at the algorithms it can be seen that the UKF is a lot more complex and more difficult to implement or convert to other programming languages. While the EKF has a step for linearization by taking advantage of computing Jacobians then the UKF skips that step and uses the unscented transformation that includes calculating the sigma points and their weights. Despite that, both of the algorithms can be initialized in the same way and give the outputs in the same format - as Euler angles. The differences of the algorithms can be analyzed further when looking at the simulation results in the next chapter.

# 5 Simulations

To be able to validate the two modified algorithms, verify that they perform as expected and compare them, simulations had to be conducted. These simulations are also important because all the parts of the ADCS need to have implementations in MATLAB in order to make sure that everything works stable on a computer. The simulations conducted in this work would later be a part of all the ADCS simulations in order to make them work together.

In this paragraph it will be explained how the simulations were done and what the results were. For running the simulations MATLAB was used.

## 5.1 Methodology and initialization

When implementing the following simulations, the simulations presented in [29] were taken as an example for creating personalized scripts in MATLAB. For all the following simulations the measurements were generated with the error distributed around 0 using Gaussian distribution. For each test a different set of measurements was generated but both of the algorithms (EKF and UKF) were ran with the same data to be able to compare them more efficiently. The standard deviation was chosen to be 0.1 and based on that, the measurement variance $R = (0.1)^2 = 0.01$. The measurements, in radians, were generated for the Euler angles as well as for the gyroscope inputs, since the goal was to run the simulation with the previously described algorithms.

The author chose the initial state $x$ to be 20 degrees in each angle. Since the actual state is expected to be 0 degrees which is distinguishably far from 20, then it will be visually clear to see how the algorithm manages to get closer and closer to the actual state. The process noises were set to be very small: $Q = 0.00001$. A small process variance results in the filter being less affected by new measurements. The bigger the process variance the more the filter is influenced by measurements. The error covariances were set to $P = 1$ because in case of $P = 0$ the convergences to the actual states would not be visible and the algorithm would understand that the initial state is absolutely correct. The measurement variance was initialized to be $R = 0.01$ which is the same that was used for generating the measurements and should give "best" performance from the algorithms.

As the results of the simulations it is expected that the filtered measurements converge to 0 degrees since the measurements were created accordingly. Also, it is stated [16] that the attitude should be determined with the accuracy of 2 degrees because the main idea of the satellite is to be able to take pictures from a desired angle.

## 5.2 Data analysis of first simulation results

The simulation was repeated 10 times and the averages were calculated for each parameter. Before each run, 60 distinct measurements were generated. The results are displayed on Table 5.1 and Figures 5.1 and 5.2.

On Table 5.1 are the execution speeds of the algorithms and error covariance values for yaw, pitch and roll angles. The execution speed is the time that it took for both algorithms to process the 60 measurements. It can be seen that the execution time for the UKF algorithm is slightly bigger than EKF's which might be due to the fact that the UKF is more complex and calculates sigma points for the estimate. After this simulation the author came to a conclusion that it will be more important how fast either algorithm reach a certain error covariance - which of the algorithms reach faster to a result that is close to the actual state.

Also there are slight differences in the error covariance values: for EKF they are smaller which means that the first algorithm reached closer to the actual state than the second algorithm. Those differences are not major but it will be investigated closer in the next section which of the algorithms is faster in getting a more precise result.

| EKF | | UKF | |
|---|---|---|---|
| Execution time [s] | Error covariance (P) | Execution time [s] | Error covariance (P) |
| 0,0323 | 0,000325835 (Yaw) | 0,0735 | 0,000335836 (Yaw) |
| | 0,000325817 (Pitch) | | 0,000335817 (Pitch) |
| | 0,000325815 (Roll) | | 0,000335815 (Roll) |

Table 5.1: First simulation average results from 10 executions presenting execution times and error covariance values

On the Figures 5.1 and 5.2 are the yaw, pitch and roll results of EKF and UKF. It is clear to see that the estimated state ends up very near to the actual state and the measurements affect the estimated states less and less while the algorithm is running. Visually, no major differences can be seen between the EKF and UKF results - the plots are almost equal. Already from the results of the first simulation it can be concluded that the filters are working as they should because in both cases the convergence is very identifiable.

By looking at the visualizations on 5.1 and 5.2, it is possible to determine that the filtered states get close to the actual states already with roughly about 10 measurements. How fast the algorithms actually reach a certain state will be more closely investigated in the next section. The following visualizations are presenting the results of the very first run out of the 10 simulations.

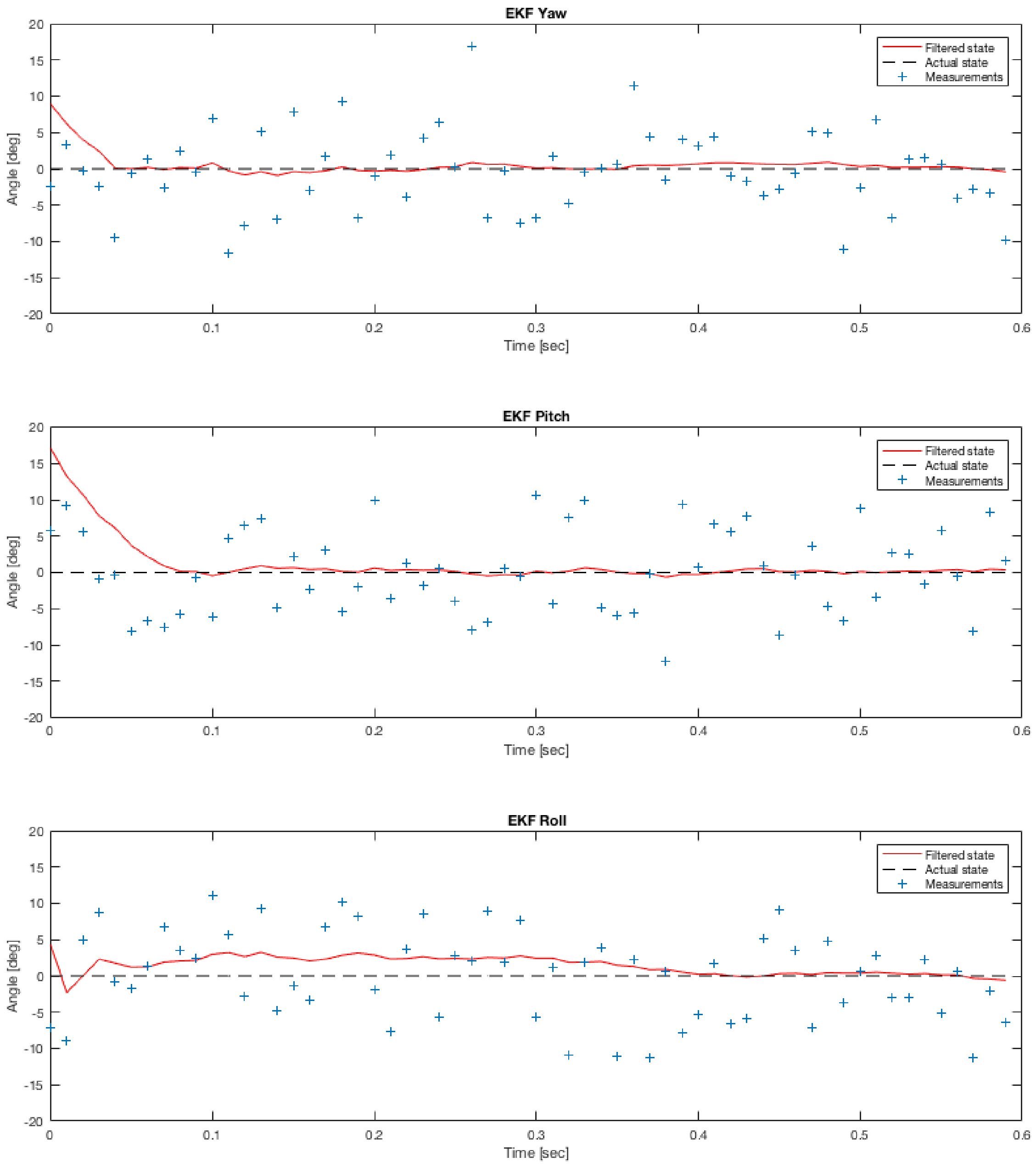

Figure 5.1: Yaw, pitch and roll angles from first EKF simulation

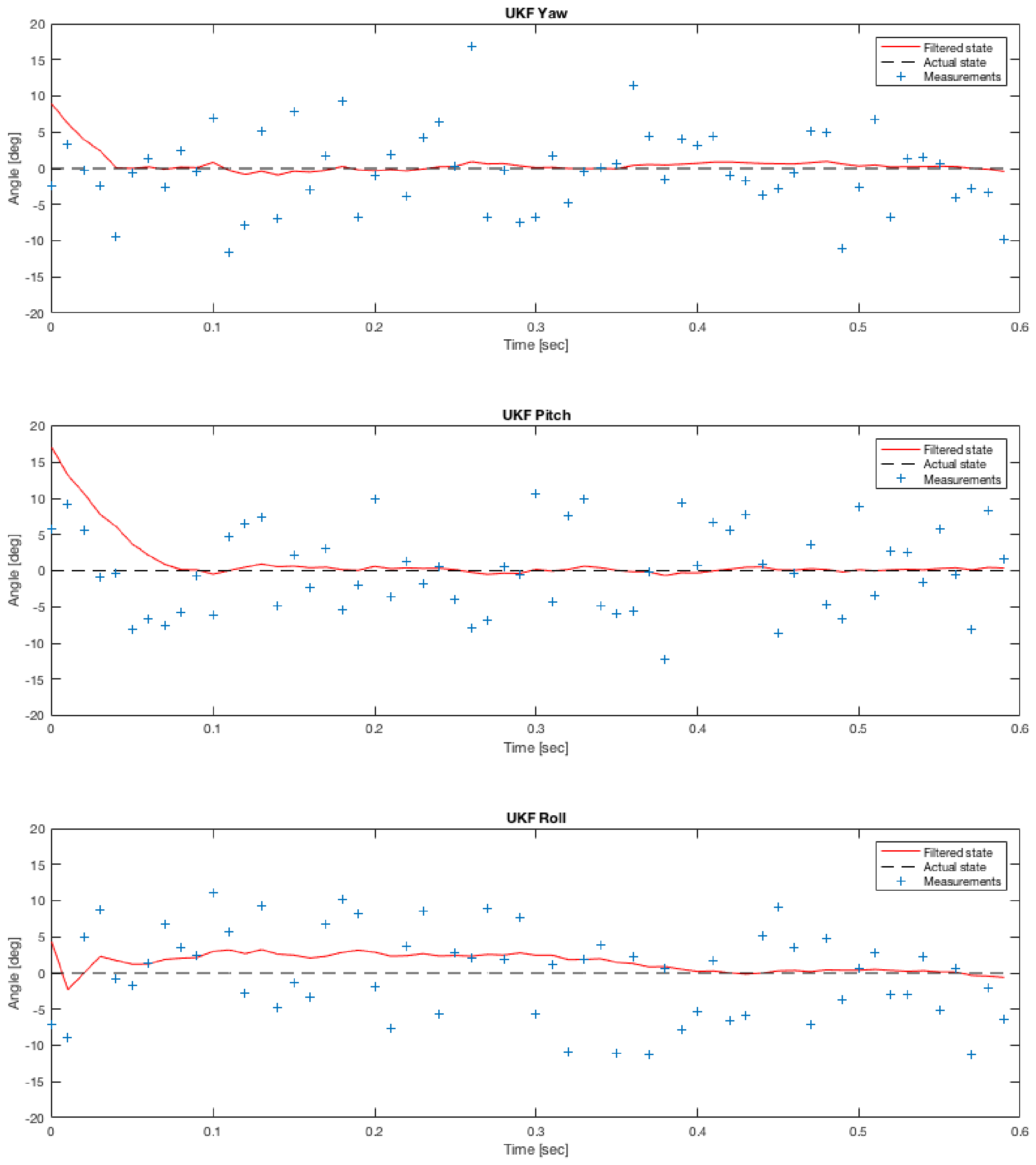

Figure 5.2: Yaw, pitch and roll angles from first UKF simulation

On Figures 5.3 and 5.4 the same results have been zoomed in showing the last measurements and how they affect the filtered results. From these visualizations it can be seen that the filtered results range inside the interval of -2 until 2 degrees and it was one of the requirements for the TTÜ Mektory nanosatellite attitude determination system that it should estimate the attitude within the precision of -2 to 2 degrees.

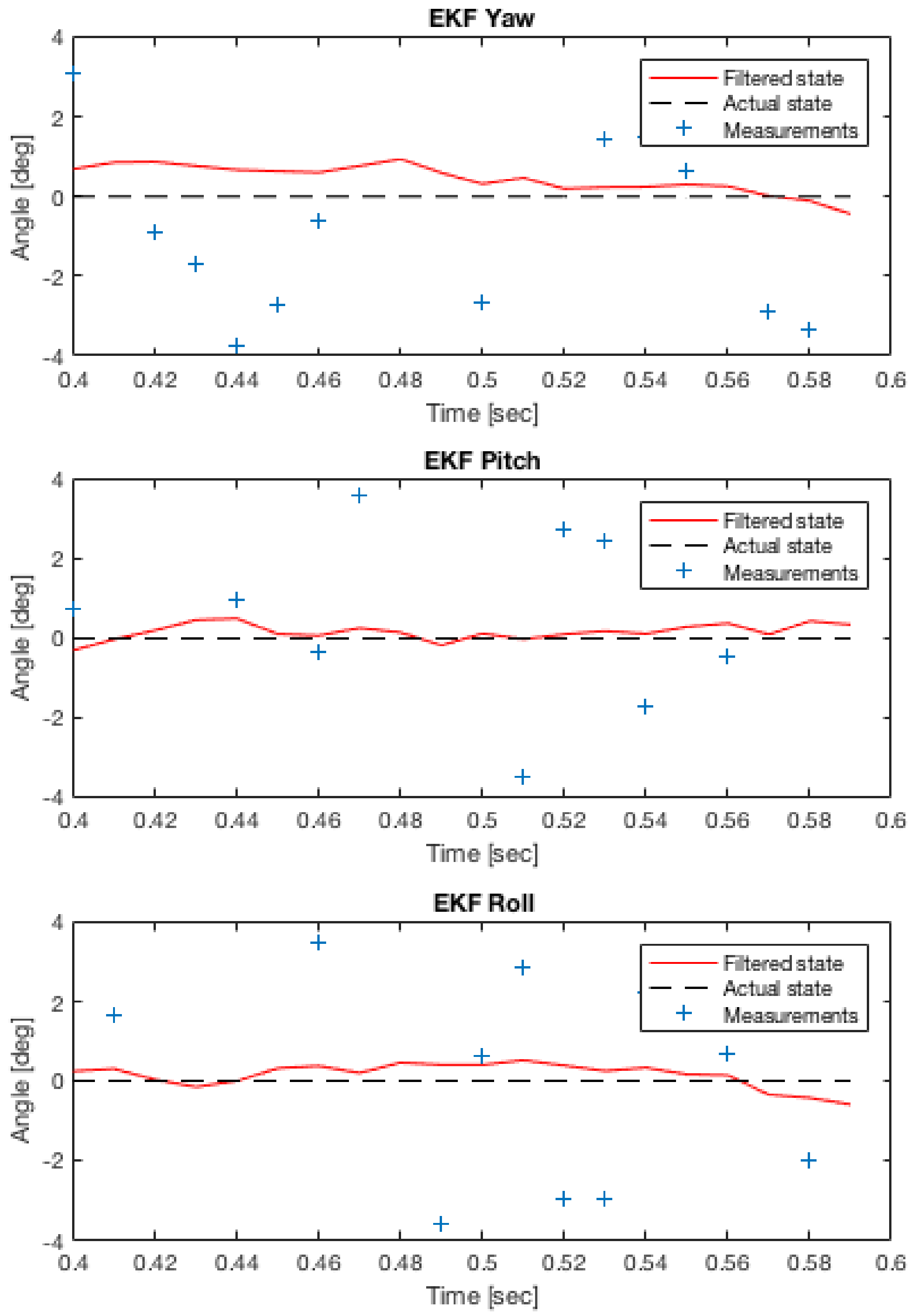

Figure 5.3: Last measurements from first EKF simulation zoomed in

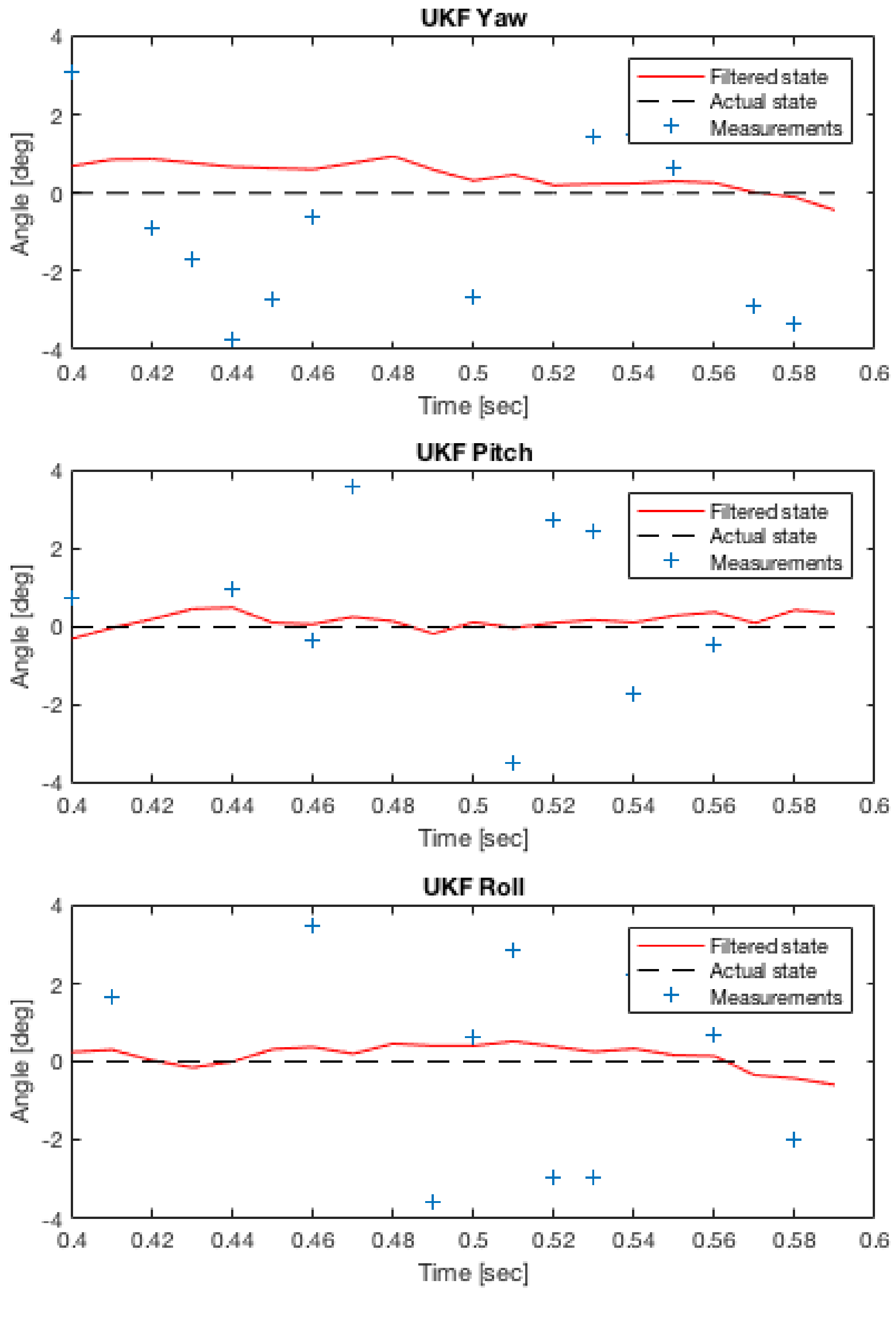

Figure 5.4: Last measurements from first UKF simulation zoomed in

In a similar work conducted by Shaobo Ni and Cui Zhang [10], simulations were ran with magnetometer, sun sensor and gyroscope data by using EKF algorithm.

The differences with this work were that instead of the Euler angles the quaternions were used and also the initialization values were different. In their work it was found that the estimated attitude angle error converged to 0.05 degrees within 20 minutes. In this current work the simulation times were considerably shorter and also the generated test data and initialization values were different, so the reached accuracy of the filters seems feasible to the author.

Another work conducted by O. Diaz where the EKF and UKF filters were compared and simulated [28], it was found that the UKF performed better in case of nonlinearities but performed 2.4 times slower than the EKF. It was also possible to conclude in this current work that the UKF performed slower than the EKF in all simulated cases. Furthermore, it was brought out that the UKF has 2.5 times the cost in computational time of the EKF. The conclusion that was made in that work was that the UKF could be used to avoid worst case scenarios but if the spacecraft has relaxed attitude control requirements and low computational power, EKF should be sufficient to use.

The main reason for the differences in the computational times of the EKF and UKF come from the fact that the UKF uses the Unscented Transformation that needs to handle all the sigma points. It has been also pointed out in a previous study where the EKF and UKF were compared in order to use for human motion tracking in a virtual reality application [27]. It also reaches a conclusion that if calculating the Jacobians is not an issue based on the structure of the process then the UKF does not give many benefits. The main difficulty with the Jacobian matrix evaluations is usually that they are non-trivial and lead to implementation difficulties.

From the results of the first simulations it is possible to conclude that the results of the EKF algorithm were better than UKF. The execution speed for the EKF was always faster and the reached error covariance values were always smaller, meaning that the process got closer to the actual states. The author also concluded that the execution speed do not play a significant role - more important is how fast the algorithm would reach a certain error covariance value because in real life the actual states are constantly changing and the process needs to be able to follow along. Because of this, in the next simulation it will be measured how fast either algorithm reaches a fixed error covariance.

## 5.3 Data analysis of second simulation results

In the second simulation the tests were run with the same initial values as in the first simulation. To be more certain in the differences in the algorithms the tests were run 10 times and 60 distinct measurements were generated each time. This time it was more closely looked at error covariance values. Since in the real situation it is important that the process would be fast enough to determine the actual state then the author fixed a value for the error covariance $P = 0.001$. This error covariance was considered precise enough by the author to state that the process has reached close enough to the actual state and the conclusion was made after running simulations and looking at the accuracies. In order to compare the two algorithms, time was measured to find out how fast either algorithm reached the goal value of the error covariance.

| EKF | | | UKF | | |
|---|---|---|---|---|---|
| Mean time [s] | | | Mean time [s] | | |
| Yaw | Pitch | Roll | Yaw | Pitch | Roll |
| 0,035646 | 0,036214 | 0,036716 | 0,087611 | 0,088775 | 0,089602 |

Table 5.2: Second simulation average results from 10 executions presenting times to reach a fixed error covariance value

On Table 5.2 it can be seen from the average values of the 10 simulations that the UKF reached the fixed error covariance value slower than the EKF. Since the attitude needs to be acquired fast and accurately then the faster the actual state is reached the better the performance.

From these results it can be concluded that with these certain initial conditions and generated test data in the previous simulations the EKF algorithm proved to perform slightly better than the UKF, although the differences in these algorithms are fairly small.

## 5.4 Data analysis of gyroscope simulation results

The first and second simulations were ran with the initial conditions that should be "ideal". Since every system is a bit different, then the initial values should be tuned accordingly. In this section the author generated measurements by using a gyroscope that was built in a STM32F microcontroller (based on ARM Cortex M4 architecture) and that will also be used in the nanosatellite [16]. The connection between the board and computer had to be established and for viewing the measurements an existing MATLAB code was used. The measurements were generated while the author was rotating the board around and changing its attitude. The existing code had to be edited a bit to extract the needed values and save them into variables. Since there was currently no possibility to get values from other sensors then the Kalman algorithms were also modified to only take into account the gyroscope measurements.

Since in the previous simulations it was concluded that the results of the EKF and UKF were very similar but EKF performed slightly better then the following simulations with gyroscope measurements will be shown only from EKF. The author compared the EKF and UKF also for these simulations but they proved to be very identical and decided to show only results from the EKF.

Followingly, three different simulations will be presented with the same data from the gyroscope but having different initialization data to show how much effect the different initial values affect the whole process. All the figures will show the gyroscope measurements along with filtered states.

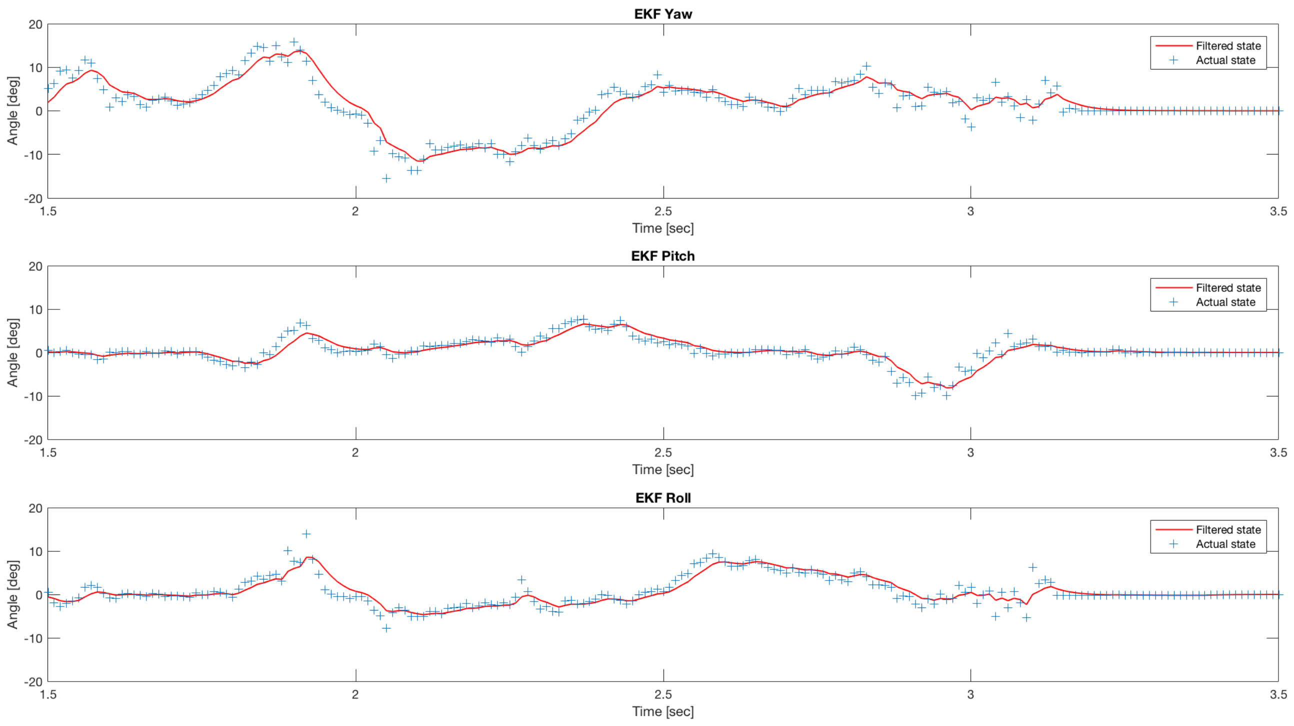

Figure 5.5: Measurements with optimal measurement variance

On Figure 5.5 it is seen that the filtered states follow the measurements but not very precisely. The measurement variances were fixed to be $R = 0.01$. Out of all the three simulations with gyroscope data, this one is the most optimal one.

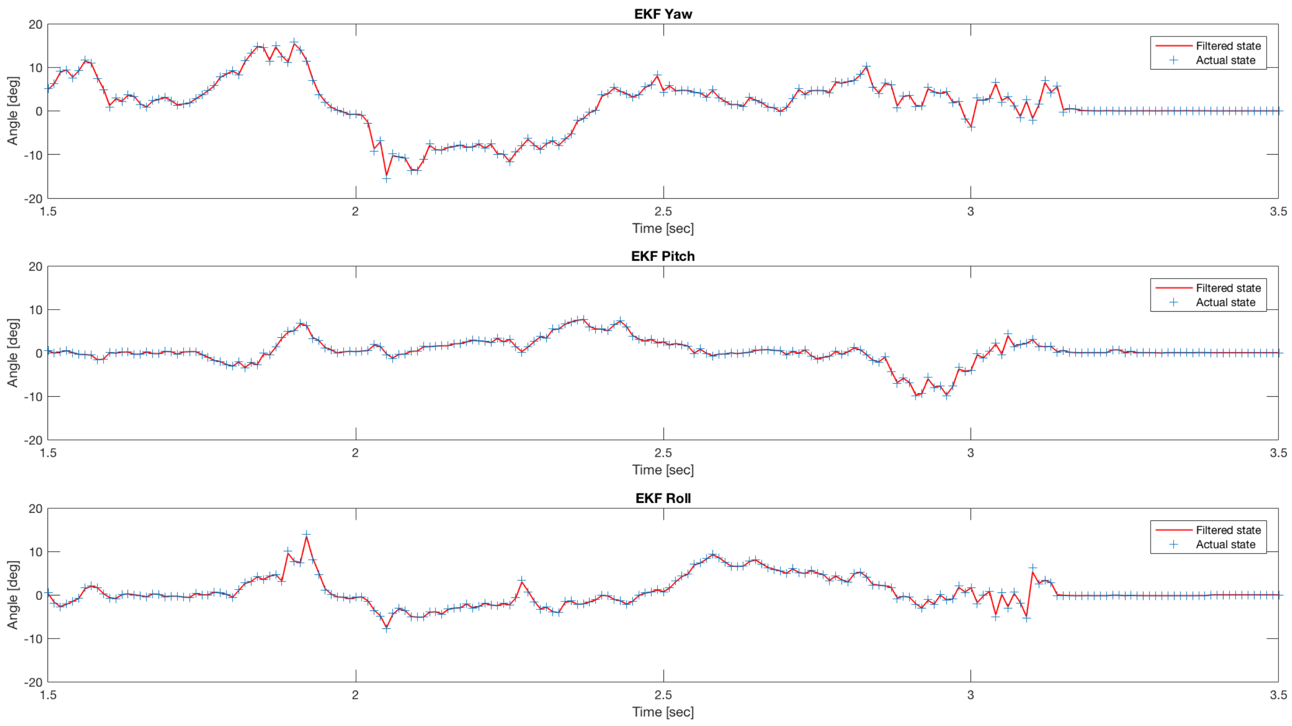

Figure 5.6: Measurements with small measurement variance

On Figure 5.6 the measurement variances were fixed to $R = 0.0001$. In this case, when decreasing the variable, the actual measurements affect the filtered states a lot because they are being taken into account as very precise. Almost no state estimation is done because the filtered states follow along the measurements and the filtered states also might end up very noisy. Compared to the first figure, the visualizations for the angles are a lot less smooth.

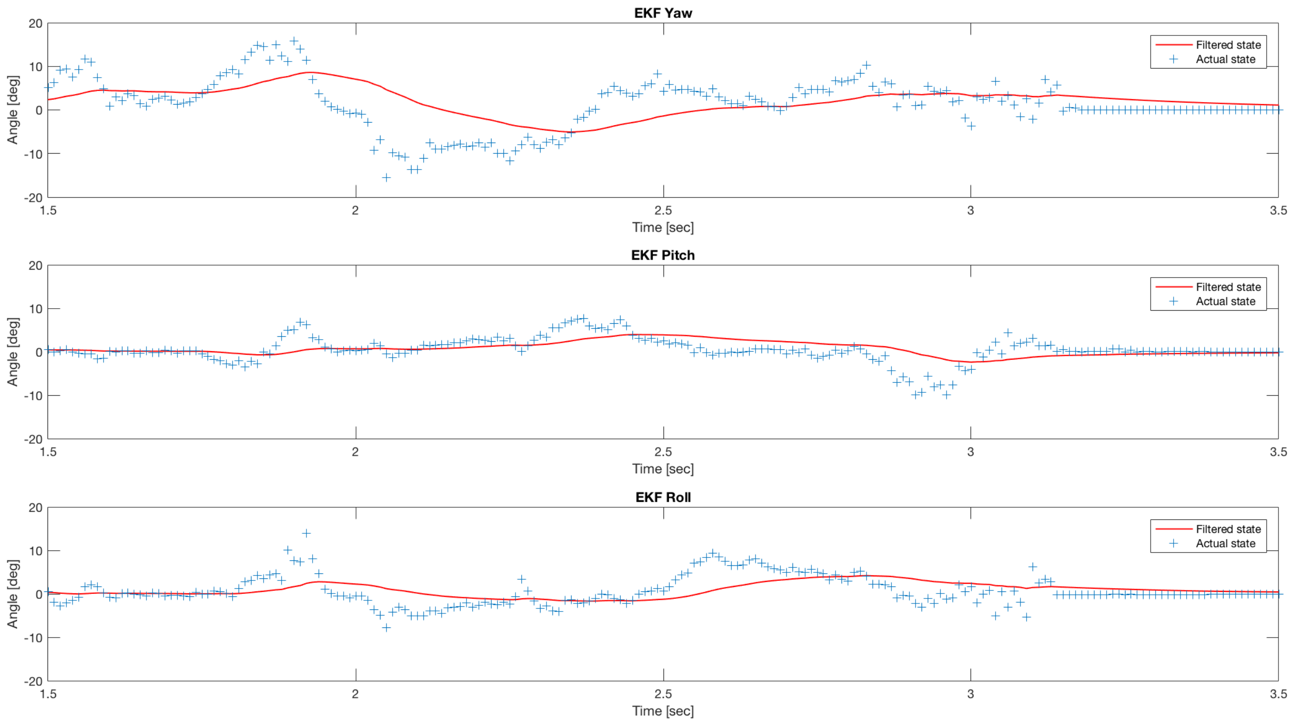

Figure 5.7: Measurements with large measurement variance

On Figure 5.6 the measurement variances were set to be $R = 1$ at initialization. This visualization shows that when increasing the variances the filter starts to respond a lot slower to the measurements and the filtered states are being smoothened a lot. Compared to the first simulation it filters out too much and also loses a lot of important information.

From these visualizations it can be concluded that tuning the filter parameters is very important and affects the process a lot. Since at the moment it was not possible to test the sensor fusion with actual measurements from all sensors, then further simulations can be done in the future. The variables can be tuned accordingly to make the process fit the needs and get more informative results.

# 6 Conclusions

It was needed that the satellite ADCS system would include signal filtering and state estimation. To find a solution for that, research had to be done and the chosen options needed to be analyzed to verify that the chosen solution is suitable. The Kalman filter was chosen for further investigations since its characteristics and history proved to be suitable for the problem in hand. In the end two variants of the Kalman filter were picked out and both implementations in MATLAB were taken as the basis for conducting comparisons and simulations.

It was also found out that for representing the attitude, the Euler angles would be suitable because they were easily applicable and widely spread in the aerospace field. It was important to specify what the inputs for the algorithm have to be because the measurements from different sensors would have to be converted to the same format in order to achieve sensor fusion in the algorithm.

From the two previous chapters, code comparison and simulations, several conclusions were made. When comparing the two algorithms, EKF and UKF, and looking at the differences in the way they worked, it was possible to see that EKF had easier and more straightforward structure than the UKF. Since the MATLAB code would have to be converted into C code then it would be better to choose an algorithm that is more understandable. Also, when converting, less mistakes would occur when the algorithm's code complexity is not very high.

Unfortunately, it is not possible to only choose an algorithm by its complexity because the most important things are accurate results and good performance. In order to find out the differences in the performances of the algorithms, they had to be modified and simulations needed to be created in MATLAB. The following simulations were created and conducted:

| | |
|---|---|
| First simulation | • Measuring execution speeds and comparing differences in the error covariances<br>• Simulations with EKF and UKF<br>• Measurements generated by script |
| Second simulation | • Measuring the times to reach a certain error covariance value<br>• Simulations with EKF and UKF<br>• Measurements generated by script |
| Third simulation | • Consisting of three smaller simulations with EKF<br>• Simulations showing how different initialization values affect the process<br>• Measurements originated from a satellite gyroscope |

Table 6.1: Overview of the simulations

The author created simulations (overview on Table 6.1) to measure the execution speeds, see the differences in the error covariance values and measure the time to achieve a certain error covariance. After looking at the results of the simulations, it was concluded that the EKF performed slightly better at all times. Both of the algorithms reached an accuracy that was needed for the nanosatellite pointing.

Finally, a simulation was conducted with the nanosatellite's gyroscope that showed three different options which were achieved by changing the initialization values. The last simulation was important to show how the initialization affects the whole process and that in the future, when testing with all the sensors, the filter should be tuned accordingly. Since in previous simulations EKF had proven to give better results then the last simulation results were only presented with EKF.

The simulations conducted in MATLAB are important for the whole ADCS system in order to make all the parts work together for testing purposes. All the simulations for the ADCS will be used to verify the stability of the system on a computer before converting the implementations into C code and transferring to the nanosatellite's board.

The author would propose using the extended Kalman filter for further development since it proved to perform better with Euler angles and the current set of sensors. For further research the algorithms could also be tested with data from all the sensors if possible. The current work can be taken as documentation or basis for conducting further simulations. In this work it was also validated that the chosen algorithm and its implementation were working as expected and is suitable for using in the ADCS system of the TTÜ Mektory nanosatellite.

# 7 Summary

The main goal of this thesis was to propose a signal filtering and state estimation algorithm that can be used in the TTÜ Mektory nanosatellite. In order to validate the efficiency and suitability of the algorithm, different simulations were conducted.

In the first half of this thesis it was examined why the Kalman filter fulfills the needs of the TTÜ Mektory nanosatellite and why it is reliable enough to take into use. Furthermore, a description of the basic Kalman filter was provided followed by descriptions about attaining and presenting attitude in the TTÜ nanosatellite. It was specified what kind of sensors the nanosatellite has been equipped with and what kind of attitude representation could be used in the current system.

This study helped to gain knowledge about nanosatellite systems, the Kalman filter and attitude representation. Furthermore, the author got the experience working with MATLAB and conducting different simulations in order to get needed results.

As a result of this thesis, it can be concluded the chosen algorithm satisfies the needs of the TTÜ nanosatellite system, is reliable and efficient and is easily integrable to the satellite's ADCS system. The simulations showed that both of the algorithm's that were examined performed well but the extended Kalman filter slightly better than the unscented Kalman filter. In the end, the simulations conducted with the extended Kalman filter and measurements from the gyroscope showed how the process responds to different initialization data which is important for further simulations and developments with all the sensors when they will are available.

The chosen algorithm, simulations in MATLAB and the thesis itself as documentation can be used for further development and research connected to the TTÜ Mektory nanosatellite. The algorithm implementations and simulation codes are accessible in GitLab, the link can be found in appendix A. Since at the time of writing this thesis it was only possible to run simulations with the gyroscope data then further work could include running simulations with all the sensors data to test the sensor fusion with actual measurements.

# Bibliography

[1] CubeSat Design Specification Rev. 13. The CubeSat Program. California Polytechnic State University, SLO. [WWW] https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf (Accessed: 04.05.2017)

[2] ESTCube-1 homepage. [WWW] https://www.estcube.eu/en/estcube-1 (Accessed: 17.02.2017)

[3] AAUSat-2 in eoPortal Directory. [WWW] https://directory.eoportal.org/web/eoportal/satellite-missions/a/aausat-2 (Accessed: 04.05.2017)

[4] AAUSat3 homepage. [WWW] http://www.space.aau.dk/aausat3/ (Accessed: 17.02.2017)

[5] ESA. (2016). Sentinel-1B launched to complete radar pair. [WWW] http://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-1/Sentinel-1B_launched_to_complete_radar_pair (Accessed: 04.05.2017)

[6] AISSat-1 and 2 in eoPortal Directory. [WWW] https://directory.eoportal.org/web/eoportal/satellite-missions/a/aissat-1-2 (Accessed: 17.02.2017)

[7] Aasen, C., Moen M. (2017). Norway's Satellites. [WWW] https://www.romsenter.no/eng/Norway-in-Space/Norway-s-Satellites (Accessed: 04.05.2017)

[8] Norwegian Defence Research Establishment. (2015). AISSat-1 – Norway's first observation satellite. [WWW] http://www.ffi.no/no/Publikasjoner/

Documents/AISSAT-1_Norways%20first%20observation%20satellite.pdf (Accessed: 04.05.2017)

[9] Shuster, M. D. (2007). The Optimization of TRIAD. — The Journal of the Astronautical Sciences. Vol. 55, No 2. [WWW] http://www.malcolmdshuster.com/Pub_2007f_J_OptTRIAD_AAS.pdf (Accessed: 04.05.2017)

[10] Ni, S., Zhang, C. (2011). Attitude Determination of Nano Satellite Based on Gyroscope, Sun Sensor and Magnetometer. [WWW] http://www.sciencedirect.com/science/article/pii/S187770581101678X (Accessed: 04.05.2017)

[11] Unhelkar, V. V. (2012) Satellite Attitude Estimation using Sun Sensors, Horizon Sensors and Gyros. Indian Institute of Technology, Bombay. [WWW] http://docshare01.docshare.tips/files/29916/299167552.pdf (Accessed: 04.05.2017)

[12] AZoSensors. (2013). What is a Sun Sensor? [WWW] http://www.azosensors.com/article.aspx?ArticleID=223 (Accessed: 04.05.2017)

[13] Kim, P. (2010). Kalman Filter for Beginners with MATLAB Examples. Republic of Korea: A-JIN Publishing company.

[14] Veloso Garcia, R., Koiti Kuga, H., F. P. S. Zanardi, M. C. (2011). Unscented Kalman Filter Applied to the Spacecraft Attitude Estimation with Euler Angles. Volume 2012 (2012). [WWW] https://www.hindawi.com/journals/mpe/2012/985429/ (Accessed: 04.05.2017)

[15] Jagadish, C., Chang, B.-C. (2007). Diversified redundancy in the measurement of Euler angles using accelerometers and magnetometers. [WWW] http://ieeexplore.ieee.org/document/4434730/ (Accessed: 04.05.2017)

[16] Org, P. (2016). Software Architecture and Development Tools API for Attitude Control System of TUT Nanosatellite: BSc thesis. Tallinn University of Technology, Tallinn.

[17] Starin, S. R., Eterno, J. (2011). Attitude Determination and Control Systems. [WWW] https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110007876.pdf (Accessed: 04.05.2017)

[18] Hall, N. (2015). Aircraft rotations. [WWW] https://www.grc.nasa.gov/WWW/K-12/airplane/rotations.html (Accessed: 04.05.2017)

[19] Using Quaternion to Perform 3D rotations. [WWW] http://www.cprogramming.com/tutorial/3d/quaternions.html (Accessed: 04.05.2017)

[20] Unity. (2017). Documentation: Quaternion. [WWW] https://docs.unity3d.com/ScriptReference/Quaternion.html (Accessed: 04.05.2017)

[21] MathWorks. Documentation: eul2quat. [WWW] https://se.mathworks.com/help/robotics/ref/eul2quat.html (Accessed: 04.05.2017)

[22] TUT Mektory Nanosatellite programme homepage. [WWW] http://www.ttu.ee/?id=115635 (Accessed: 10.02.2017)

[23] Diebel, J. (2006). Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. Stanford University, Stanford. [WWW] https://www.astro.rug.nl/software/kapteyn/_downloads/attitude.pdf (Accessed: 04.05.2017)

[24] McGee, L. A., Schmidt, S. F. (1985). Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry. — NASA Technical Memorandum. Ames Research Center. [WWW] https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19860003843.pdf (Accessed: 04.05.2017)

[25] Slavinskis, A. (2015). EstCube-1 attitude determination. Estonia: University of Tartu Press.

[26] Simon, D. (2001). Kalman Filtering. — Embedded Systems Programming. [WWW] https://c.forex-tsd.com/forum/29/kalman.pdf (Accessed: 04.05.2017)

[27] LaViola Jr., J. A Comparison of Unscented and Extended Kalman Filtering for Estimating Quaternion Motion. Brown University Technology Center for Advanced Scientific Computing and Visualization. [WWW] `https://ai2-s2-pdfs.s3.amazonaws.com/df70/ce9aed1d8b6fe3a93cb4e41efcb8979428c0.pdf` (Accessed: 04.05.2017)

[28] Diaz, O. X. (2010). Analysis and comparison of extended and unscented Kalman filtering methods for spacecraft attitude determination. Naval Postgraduate School, Monterey. [WWW] `http://calhoun.nps.edu/bitstream/handle/10945/5010/10Dec/_Diaz.pdf?sequence=1&isAllowed=y` (Accessed: 04.05.2017)

[29] Welch, G., Bishop, G. (2006). An Introduction to the Kalman Filter. University of North Carolina, Chapel Hill. [WWW] `http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf` (Accessed: 04.05.2017)

## A Link to git repository

https://gitlab.com/martinrebane/ADCS/tree/Kalman