Lucerne University of Applied Science & Architecture

ARIS - Data Fusion for a Sounding Rocket

Bachelor Thesis

Author:	Michael Kurmann
Supervisor:	Prof. Marcel Joss
Expert:	Werner Scheidegger
Industrial Partner:	ARIS (Akademische Raumfahrt Initiative Schweiz) Oliver Kirchhoff
Submission date:	22.12.2017
Classification:	Access

Declaration

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from sources are properly denoted as such. This paper has neither been previously submitted to another authority nor has it been published yet.

Horw, May 24, 2018

Abstract

This is the Abstract

Contents

Chapter 1

Introduction

1.1 Task

Figure 1.1: Official logo of the competition project 2018

The Academic Space Initiative Switzerland (ARIS) figure 1.1 is a student group which competes in the yearly Intercollegiate Rocket Engineering Competition (IREC). The goal of this competition is to build a rocket which can fly autonomous at a predefined apogee (10000 feet = 3048 meter) and after that return safe to the ground. There are a total of 1000 points to achieve in the competition which are split into different parts.

Description	**Points**	**Percent**
Entry form an progress update	60	6 %
Technical report	200	20 %
Design implementation	240	24 %
Flight performance	500	50 %
Total	1000	100 %

Table 1.1: Calculation of the points of the IREC

Table 1.1 shows how those points are divided in detail. It can be seen that just the halve of the points are assigned for the performance at the competition itself. The other halve of the points can be achieved by teamwork, professional documentation and engineering during the developing and construction of the rocket. For the 500 points which are assigned for the flight performance, 350 are assigned for the error made between the targeted and approached apogee. These points are calculated like this:

$$Points = 350 - \frac{350}{0.3 \cdot TargetApogee} \cdot |TargetApogee - ActualApogee|$$

So there is a total of one point loss per 2.6 meter error [2].
Figure 1.2 shows the rocket Tell which will be used in this years competition. To aim for the right apogee a Control algorithm is implemented in the lower body micro controller. This control algorithm predicts the apogee which would be achieved depending on the actual states of the rocket (height. speed. acceleration). It then drives the air breaks to adjust that predicted apogee to the aimed 10000 feet. This algorithm relays

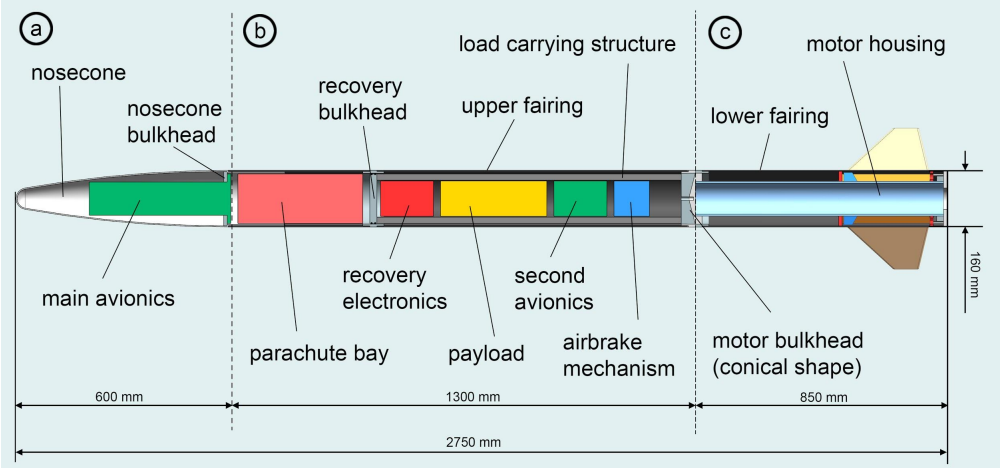

Figure 1.2: Concept of the 2018 competition rocket Tell

on the information of different sensors to determine the rockets actual state. Because there are different sensors to measure the same value a algorithm which fuses those data would come in handy. With this fusion algorithm it should also be possible to be more accurate as with each sensor on its own. So the aim of this thesis is to implement a simulation and with its help find the algorithm which is most suitable for this task. For this the problem as well as the desired solution will be defined in this chapter. After that the models for the different sensor that will be used as well as the pro and con of different state estimators are discussed at beginning of chapter 2. In addition, different possible system models are also described in this chapter. Chapter 3 will then show, how the simulation and the fusion algorithm are implemented in detail. To verify that the implementation is working as intended, the results of the simulation are discussed in chapter 4. In the last chapter 5 a summary of the achieved knowledge, as well as a comparison between the desired and the implemented solution will be stated.

1.2 Purpose

The hardware as well as the most of the software parts that will be used for this competition are already defined. Also it is a suitable assumption that the sensors and the dynamics of the rocket will be stay more or less the same for the competitions coming. Therefore this thesis will mainly focus on finding a algorithm for this given surroundings, but it is also will try to find as modular solution as possible, so that achieved knowledge can be used in further competition. This knowledge will then be used to for better performance at the competition flight itself as well as to optimise the points achieved in the implementation part.

1.3 Research

Sensor/data fusion and state estimation is a well established engineering field. Especially since the 1960 when Rudolf A Kalman published his paper for the Kalman filter. Therefore there is already a lot of previous work which can be used in this thesis. For this thesis two books are used which provide the needed theory, this are [3] which contains basic theory about state estimation especially with kalman filters. The second book [4] is more focused on different approaches of state estimation and provides also different solution to common problems that occur while implementing a state estimation. In addition the Master Thesis [1] accesses more ore less the same issue as this thesis. Therefore it will be used mainly in the conceptional part of this paper.

1.4 Sensors

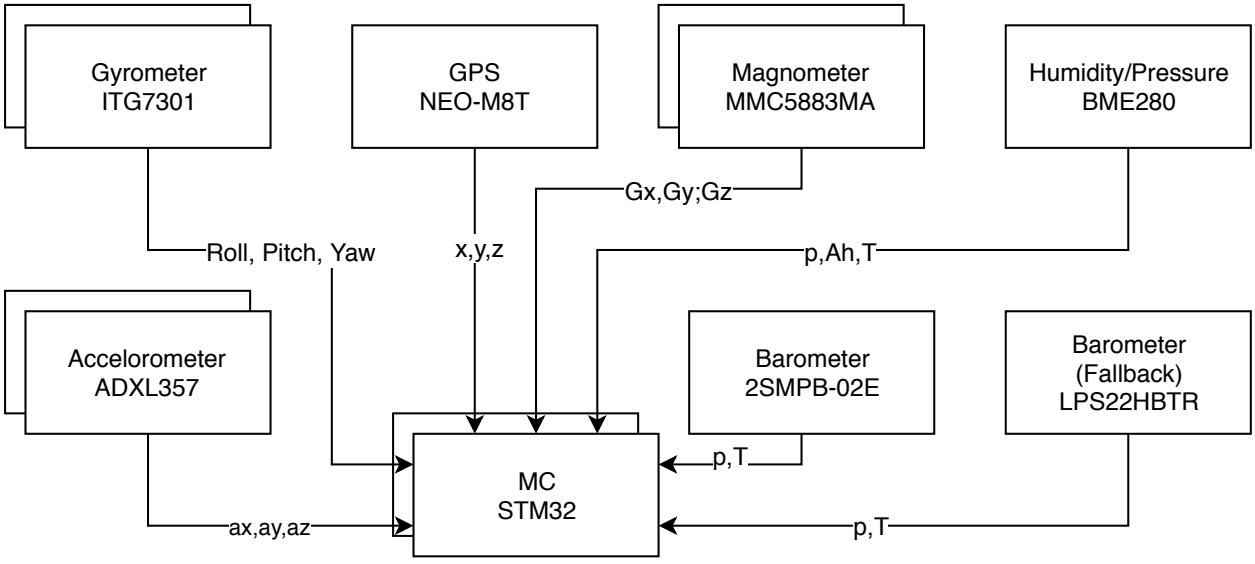

Figure 1.3: Sensor Network

As mentioned above different sensors are used in this years competition and due to the assumption that in concept the same sensors will be used in the competitions coming they will act as the basis for the sensor fusion- This used sensors and their settings will be described in this chapter.

1.4.1 Accelerometer

First of all, comes the accelerometer. This is a well established and widely used sensor. It measures the force which is applied on the sensor in the three space dimensions. This years accelerometer is adxl357 which will be sampled at 1000 Hz.

1.4.2 Gyrometer

The Gyrometer is needed to measure the posture of the Rocket. This is especially needed to determine if the rocket has a pitch angle. If so the pure acceleration on the z-axis can be calculated. The used gyrometer is ITG-3701 and it will also be sampled at 1000 Hz.

1.4.3 Barometers

Barometers are widely used in aviation, cause with a common pressure model the height can be calculated out of the measurements that the barometer takes. In this years competition three barometers are used. This are by name 2SMPB-02E and LPS22HBTR. They will be used with a sampling rate 100Hz and 50Hz. In addition the Humidity/Pressure sensor BME280 does also measure the pressure which can be used in addition.

1.4.4 Temperature

The temperature is maybe needed to make the height out of the barometer better because most of the atmospheric model depend on the pressure as well as the temperature. This temperature will be provided by the different barometers which each posses a separate temperature sensor.

1.4.5 Magnometer

Also there are two Magnometer in the sensor network 1.3. These measure the strengths of the surrounding magentic field. This can be used to determine the direction regarding the north pole. Due to the fact that the algorithm that will be developed in this thesis does not include the X- and Y-Axis, these sensor will not be used for the sensor fusion.

1.4.6 GPS

For next years competition differential GPS will be implemented with the help of two μblocks modules. This taken measurements are more precise as the rest of sensors but are taken much slower on a rate like 0.5 - 2Hz. Therefore the algorithm should takes those provided measurements and interpolate between them with the data from the other sensors.

1.5 Problems

Out of the research and the previous competition, different problems appeared that need to be addressed in this thesis to ensure an as good solution as possible.

1.5.1 Different Sensors

First of all there are different sensors which all do measure different values and have different parameters (precision, sampling time). So the algorithm has to use out the strengths of the different sensors to cancel out their individual weaknesses. Additionally, because this algorithm is system critical, it has to be reliable enough that it still is working properly if sensors failing.

1.5.2 System Load

The cycling time will be around 1 ms on a embedded system. This time was chosen on the behalf that it would be difficult to get the exact needed cycling time on ensure the needed controllability of the rockets apogee. Therefore the system load that the algorithm can cause, has to be strongly limited, so that it can be run on this given system. The system this year is an 32 bit Arm Processor which runs on 168 MHz, assumed that the algorithm has at maximum the half of a software cycle, the maximum given clock cycles are around 84 000. With this cycles the processor can do around 10000 simple calculation (addition, subtraction, multiplication, division), cause with its floating point unit it needs on average around 8.5 cycles per operation (load, calculate, store). This number is just a rough assumption, which means that the final system load should not exceed this value by a great manner.

1.5.3 Precision

The Precision is after the system load the most critical attribute, if the algorithm does not get into the required accuracy the whole thing is more or less for nothing. The Control stated that the maximal error between the estimated and ground truth height should not exceed two meter. This especially after the burnout at which the control with the air breaks will start. This accuracy is needed to proper control the aim of the apogee.

1.5.4 Settling Time

The settling time defines the time span when the first reliable measurements arrive after burnout until the estimation is into the required precision. This time span has to be small enough to ensure that the controlling has enough time to aim for the desired apogee. In the current system the burnout occurs occurs around 3-3.5 seconds after ignition, whereas the whole flight upwards only takes around 23 seconds. Therefore the settling time needs to be around just one second so that the control has as much time as possible for the controlling.

1.5.5 Reliability

Due to given surroundings that come if a sensor package is placed into a rocket, the assumption has to be made that it will be possible that sensors fail in execution. Therefore the algorithm should provide the reliability of still working in a proper manner with some sensors failed. So that the execution of the controlling software in terms of functionality, but locally it has not to be as accurate as it would be with all sensors working.

1.5.6 Modularity

Although it can be assumed that the sensors will stay more or less they same over the next competitions, it is not ensured that exactly this sensors will be used. Therefore the presented algorithm should provide the possibility to exchange the sensors, as long as they resemble the old sensor in a feasible way. This will ensure a long term use of the provided algorithm.

1.6 Requirements

Requirement	**Rating**	**Aim**	**Importance**
System Load	# Calculation steps per loop	< 5000	Critical
Precision	Error between estimation and ground truth	$<$ 2m in Z	High
Settling time	Time from first reliable to optimal estimation	$<$ 1 s	High
Reliability	Functioning Estimation with #failed sensors	2-3 sensors	Medium
Modularity	Effort needed to change a sensors	$<$ 10 h work	Desirable

Table 1.2: Requirements table

As seen in the table 1.2 five requirements were drown out of the problem analysis. First of all, there is critical requirement the system load. This is given as critical cause it is needed that the algorithm is small enough to be run on a embedded system. Any solution that would not fit this requirement would be pointless in the frame of this thesis. Secondly there are two requirements which are tied together, the precision and settling time. Where the precision describes what a optimal estimation is under the context of this thesis, the settling time relies on this to be defined. The desired precision will be needed to ensure a possible good control.

1.7 Desired Solution

The desired solution should met the given requirements as optimal as possible. While doing this it should also not be more complicated than needed to make a future use as easy as possible. Cause of this the modularity is an important requirement to ensure this.

Chapter 2

Approach

2.1 Verification

First of all the test concept has to be defined on which the developed algorithms will be tested. This is also be specially useful for the future competitions to test the adjusted state estimator which are used there. For this the following concept figure 2.1 was developed.

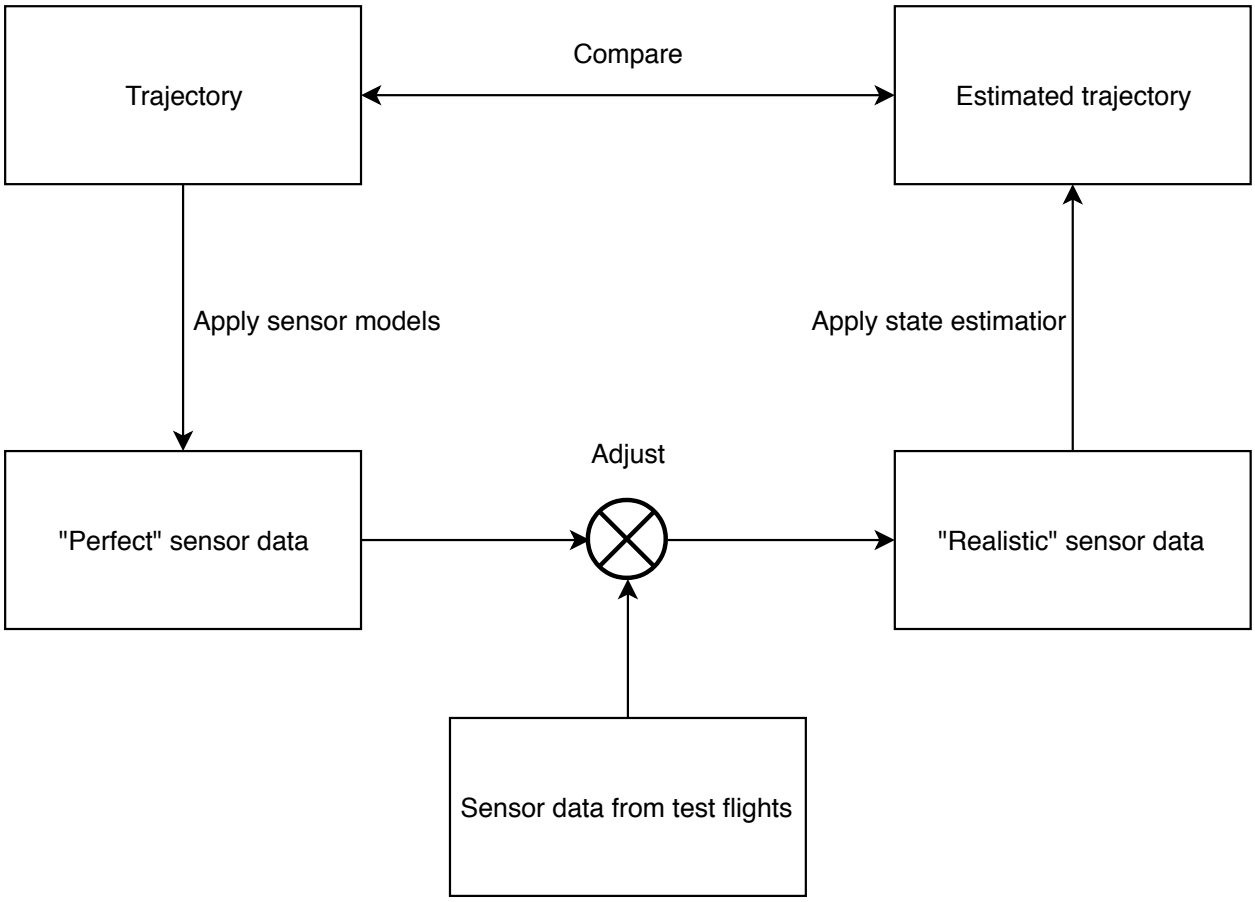

Figure 2.1: Verification Concept

The theory behind this is that a trajectory is generated by the simulation, which should resemble a real trajectory as good as possible. This function was provided by the simulation team of ARIS from the last years competition. Then this trajectory is applied on the sensor models which are developed in chapter 3. This generates so called perfect sensor data. After this the noise is applied which will also be developed later in this chapter, which will result in real sensor data. This noise is drawn out of the sensor log data from the previous test flights. After the realistic sensor data is generated, it serves as the input to the different estimation algorithms.

Then to verify the functionality of those algorithms, the estimated trajectory is then compared against the generated trajectory.

2.2 Sensor models

As stated above the trajectory will be generated by the simulation. This are just the information of the height, so the different sensor models have to be adjusted to get to the different needed data. The models are defined as follow.

2.2.1 Accelerometer

Perfect measuring from the accelerometer is in simple terms the two times deviation of the height.

$$a = \frac{d^2h}{dt^2}$$

This equals in the straight up acceleration. To get the acceleration which would be provided by an accelerometer, the pitch angle has to be calculated into this generated data.

2.2.2 Gyrometer

For the gyrometer there does not really exist a model with which those measurements could be generated. Therefore it has to be generated free hand by taking the gyrometer measurements from the testflights into account. It has also to be sayed that only the pitch angle of the rocket which can be seen in figure 2.2 is for interest for this first sensor fusion implementation, So only this angle will be generated

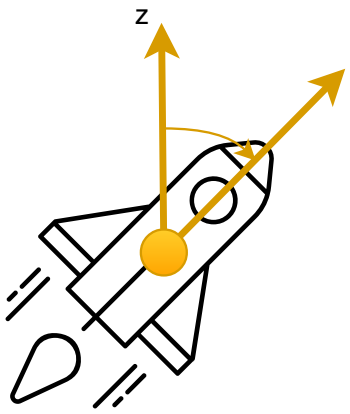

Figure 2.2: Pitch angle visualisation

2.2.3 Barometer

The barometer which are used in the aerospace usually provide the pressure in hecto Pascal as well as the temperature in degree Celsius.

Pressure

To generated the pressure data, the barometric height formula is used

$$P = P0 \cdot (1 - \frac{Tgrad \cdot h}{T0})^{\frac{M \cdot g}{R \cdot Tgrad}}$$

with: PO = Pressure at ground level
TO = Temperature at ground level
Tgrad = Temperature gradient for the actual weather condition
M = Molar mass of Earth's air: 0.0289644 kg/mol
g = Gravitational acceleration: 9.80665 m/s^2
R = Universal gas constant: 8.3144598 J/mol/K

This is more or less accurate till 11 000 meter under the condition that the Temperature gradient is determined correct.

Temperature

The temperature depending on the height is a difficult subject because the temperature gradient is depending on the actual weather and the capacities of the air. So this gradient has to be determined before the start for each flight.

$$T = T0 - Tgrad * h$$

2.2.4 GPS

The perfect GPS data is seen as the accurate height but with a slow sample rate around 0.5 to 2 Hz. This can simply be achieved by down sampling the height vector with the right factor.

2.3 Noise generation

To generate the different noises, first the noise from the test flight has to be extracted. If done so, a system can be calculated which represents a white noise filter that generates noises that has the same spectral power density as original noise. This process is visualised in figure 2.3.

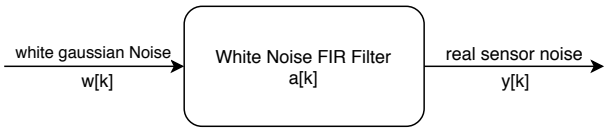

Figure 2.3: White noise filter concept

For this the yule walker equation should come in handy to calculate a so called AR (auto regressive) model of the noise system. An AR model is a FIR (all pole) system which generates a random sequence with defined auto correlation γ_{yy} out of white gaussain noise.

$$y[n] = \omega[n] - \sum_{k=1}^{N} a_k \cdot y[n-k]$$

The yule walker equations

$$\begin{bmatrix} \gamma_{yy}[0] & \gamma_{yy}[-1] & \dots & \gamma_{yy}[-N] \\ \gamma_{yy}[1] & \gamma_{yy}[0] & \dots & \gamma_{yy}[-N+1] \\ \vdots & \vdots & & \vdots \\ \gamma_{yy}[N] & \gamma_{yy}[N-1] & \dots & \gamma_{yy}[0] \end{bmatrix} \cdot \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} \sigma_\omega^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

estimate the coefficient $a_1 \dots a_N$ of the FIR system and the variance of the noise σ_ω^2 which can be used to generate a random sequence which has the same auto correlation γ_{yy} as the input sequence y. Because the AR model consists of an all pole filter, the noise should have has to have a steady mean value preferably it should be zero mean. This way the estimated AR model from the yule walker equation can represent the real system at best. For this the mean value of the measurements should be calculated beforehand and then be subtracted to get there zero mean noises. The part of the mean values that represent noise and can then be added after the noise in generated to represent the real sensors as good as possible.

2.4 System Model

The system model to represent the rocket will be hold simple to reduce the system load as well as prevent non linearities. Also the important values to estimate are at first hand the vertical height and speed, so for a first implementation just variables that can bed used do determine those both will be used. This are mainly the height and speed from the GPS, the vertical acceleration from the accelerometer as well as the pressure and temperature from the pressure sensors. In addition the pitch angle from the gyrometer is used to calculate the pure vertical acceleration. But even with such simplification there are different possible system description which have to be taken into account to find the best suitable.

2.4.1 General State Space System

For this a quick view at general donation of a state space system. First there is the update function.

$$\dot{x} = A \cdot x + B \cdot u + G \cdot q$$

Here the A matrix resambles how the system transmits over time by itself where the B matrix describes how input u does influence the system. In addition noise on the system can be described with the G matrix and the noise input q. The second equation is the output equation.

$$y = C^T \cdot x + D \cdot u + R$$

Here the C^T matrix describes how the state value from x effect the output while the D matrix describes how input directly effects output. The D matrix zero in the most state space systems, because in reality there are not a lot of systems where the output directly and immediately reacts to input. Also the noise on the output (measurements) can be described with the R matrix.

2.4.2 Point Mass

The most simple possible model would be, that the rocket would be resembled as a simple point mass which flies perfectly vertical upwards. For this only three state variables would be necessary, the vertical acceleration, the vertical speed and the height.

$$x = \begin{bmatrix} h_z \\ v_z \\ a_z \end{bmatrix}$$

This would reduce the A matrix of the system to a 3x3 matrix with with only two 1 in it. Also the input matrix would be a zero matrix in this system because the input process (power from the motor and drag force from the surrounding air) would be difficult to describe in a linear system.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

In addition the measurements which are taken from the sensor will be described as the system output (y vector). So if the pressure from for example two barometers is calculated into the height beforehand it would result in the flowing output matrices.

$$y = \begin{bmatrix} h_{GPS} \\ h_{p1} \\ h_{p2} \\ a_z \end{bmatrix} \qquad C^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

As normal in engineering such a simplification comes with a cost. With this system description the output from the barometer would have to be transformed into the height before they could be taken into the system. Due to this the properties of these sensor could not be estimated correct because the value was transformed in a non linear way before it entered the system. In addition the same problem occurs with the accelerometer. If the rocket develops a pitch angle not equal to 0 during the asccending, it would be measured wrong. To counter this error, the measurements of the accelerometer would have to be weighted with the angle of the gyrometer before entering the system. This weighting is also non linear and the values of the gyrometer are not filtered, which would make the estimation even more uncertain.

2.4.3 Point Mass with Pressure

To taken into account to problem stated above, pressure can be taken into the state vector and therefore be estimated.

$$x = \begin{bmatrix} h_z \\ v_z \\ a_z \\ p \end{bmatrix}$$

While this solves the problem stated above, it also produces a new. The system model can only describe linear dependencies between the state variables, but the relation between the pressure and the height is clearly non linear in each atmospheric model. This dependency can be linearized, but if done so, it does

resemble the atmospheric model with less accuracy. This linearisation can be used to interpolate between the barometer measurements so the actual pressure is available at any loop iteration. This results in a 4x4 A matrix and also a zero B vector for the dynamics equation which look like this:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & KP_h & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

This would also inflect on the output matrices which do then contain the pressure directly. In addition the height calculated from the pressure can be inserted as additional measurements to include them into the estimation.

$$y = \begin{bmatrix} h_{GPS} \\ h_p \\ a \\ p_1 \\ p_2 \end{bmatrix} \qquad C^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The input matrix stays the same as above.

2.4.4 Point Mass with Angle and Pressure

The same solution as above can also be applied for the pitch angle. But its linearisation is more complicated This would change the state vector from above into the following.

$$x = \begin{bmatrix} h_z \\ v_z \\ a_z \\ p \\ \varphi_{pitch} \end{bmatrix}$$

This would extend the dynamic part for A into a 5x5 matrix while the B matrix still would be a zero vector.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & Kp_h & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

In addition the output matrix and the y vector will also change to adjust for the gyrometer measurements.

$$y = \begin{bmatrix} h_GPS \\ h_p \\ a \\ p_1 \\ p_2 \\ \varphi_{pitch} \end{bmatrix} \qquad C^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

This solution would take all measurements available for the needed values into account. It would also keep the calculation in the system as simple and linear as possible.

2.4.5 Discretisation

The system stated above are all in continuous time state space description. For implementation on a discrete time system like a micro controller they have to be discredited. Those discrete matrices which will be denoted with an additional lowercase d (A -> Ad). The general state space discrete description with noise looks as follow.

$$x[k+1] = Ad \cdot x[k] + Bd \cdot u[k] + Gd \cdot Q[k]$$

$$y = C \cdot x[k] + D \cdot u[k] + R[k]$$

As it can see only the A, B and G matrices have to be discredited this because the matrices used in the second equation are only there for scalar adjustment for the output and should there not contain any time depending calculation. The Ad matrix can be calculated with the the following formula.

$$Ad = e^{A \cdot Td}$$

Where Ts stands for the sampling time of the system which would be 1 millisecond on the system developed for this thesis. In addition there are two known ways to calculate the other needed discrete matrices Bd and Gd.

- $$Bd = Ad \cdot B$$

  The first one resembles a perfect sampling which is equal to a multiplication with a dirac impulse. This calculation is rather further from the reality because no sensor can measure with perfect dirac impluse. The great advantage of this method on the other hand is its simple calculation.

- $$Bd = \int_0^{Ts} e^{A \cdot v} \cdot B \; dv$$

  This version does resemble a zero order hold likewise measurement which is done by most sensors so it is more realistic than the first version. It comes at the cost that the calculation is more difficult [3].

Both version have to be tested in the simulation to see if the additional effort for the second version pays of enough.

2.5 State estimator

There are many possibilities to do sensor fusion. first of all an all new algorithm could be developed which accesses the stated problems directly. While this solution would be preferable regarding the efficency, the time and knowledge needed for this task would exceed the resources given in this thesis by far. Also as stated in chapter 1 a lot of theoretical as well as practical pre work is already done and therefore should be used. So following different possible approaches with there pro and cons will be discussed.

2.5.1 Kalmanfilter

First of all the traditional discrete kalmanfilter has to be discussed as it provides the base for the most known state estimator. Its structure provides the optimal estimation of the standard deviation estimation error as long as the noises are Gaussian and the observed system can be described by linear differential equations. But there lies the problem, a physical system is most often not linear. Also the estimated system as well as its variances over the time have to be known to provide this optimal estimation. If the noise matrices of the system are static, the filters gain matrices aim for a fix value and can therefore be calculated in beforehand. This reduces the computational effort by a significant amount. [3]. It should be mentioned that even if the noise is not Gaussian, the kalmanfilter is still the best linear estimator as long as the system and its properties are well known [4]. To summarise it, the normal kalman filter is a simple to understand and adjust state estimator in comparison to the following.

2.5.2 ROSE

The ROSE(rapid ongoing stochastic estimator) is in simple terms three kalman filters in one. Where the main filter is used as a normal kalman filter like stated above, the additional two are used to estimated the the system noise as well as the measuring noise. Therefore this sensor preforms better than the traditional kalman filter if those noises change over time in a not known fashion and has therefore be estimated. Due to this, this sensor needs more computational effort [3].

2.5.3 Extended Kalmanfilter

The extended kalmanfilter provides additional parts to better access nonlinearity in the observed system. This by not estimating the state of the system but by estimating the linearized change of the state to the past sate. For this the systems equations have to be derived around the current nominal point in every estimation state. This is some sort of Bootstrap solution because the nominal point on which the derivation happens are estimated in the process and this estimates are then used to estimate the change between this estimation and the next. Therefore the computational effort exceeds even further because each function has to be deviated at each loop iteration [4].

2.5.4 Unscented Kalmanfilter

The unscented kalmanfilter takes the unscented transformation in use to calculate the different interpolating steps. The unscented transformation uses a test set of points around the current state and calculates the new values for them using the functions which represent the system. Out of these new values calculated from the test set is now the new state estimated by weighting those new values depending on there probability of occuring. With this the state function do not have to be linearized and therefore the the estimation is most time better than that of the extended kalmanfilter. But for this the unscented Kalmanfilter needs also to apply the unscented transformation onto the state vectors in each iteration and does therefore need even more computational effort [4].

2.5.5 H∞ filter

The H∞ filter is a more diverse approach then the ones described above. It was developed to access the problem when the to be observed system especially its noise is not well known. In other words it was developed to resemble an as robust as possible state estimator. In addition its stability can be easier guaranteed that with a kalman filter. For this it consists of additional tuning parameters which makes it more complicated ti use. Also in contrast to the kalman filters the H∞ filter minimises the worst-case estimation error in stand of the standard deviation of the estimation error [4].

2.6 Choosing

If the requirements table 1.2 is taken into the consideration of finding the optimal solution, two main requirements occur that define this decision. First the system load is a critical requirements and has therefore to be addressed in this process. Also for the requirement of modularity the algorithm should be as simple as possible. If taken in regard that the system is more or less well known and that the noise can be determened with the simulation and the log data from previous test flights, a normal kalmanfilter seems to be the most fitting solution. This because the performance of the rocket and the sensor should stay the same during each flight.

2.6.1 Functioning

Since the best fitting state estimator was chose. Its detailed functioning shall be described here. This algorithm works in four main equations which can be devided into prediction and correction steps figure 2.4.

Prediction

The prediction equations take the currently values of the state vector ($x[k]$) and uses the time depending system model part (A) to predict the state values for the next time step $\hat{x}[k+1]$. The hat denotes that this value is an assumption. This with the equation:

$$\hat{x}[k+1] = Ad \cdot x[k] + Bd \cdot u[k]$$

In addition the certainty matrix (P) is calculated which means that it is calculated how trustworthy those predictions are.

$$P = Ad \cdot P \cdot Ad^T + Gd \cdot Q[k] \cdot Gd^T$$

For this the system noise is used, so with the help of the Q matrix it can be stated how well known the system is in this time step.

Correction Step

If the measurements arrive those will be used can be used in the correction step to correct the prediction. First the Kalmangain (K) is calculated with the equation.

$$K = P \cdot C^T \cdot (C \cdot P \cdot C^T + R[k])^{(-1)}$$

This uses the P matrix from the prediction step as well as the R matrix which represents the measurments noise (how certain the values from the measurements are). K is then used in the last equation.

$$x[k] = \hat{x}[k] + K \cdot (y[k] - C \cdot \hat{x}[k] - Dd \cdot u[k])$$

With this the measurement is used to correct the predicted value of the state vector with there uncertainties taken into account [3].

For this the matrices for the system models (A,B,C,D), the measurements noise (Q) as well as the system noises (R) have to be defined.

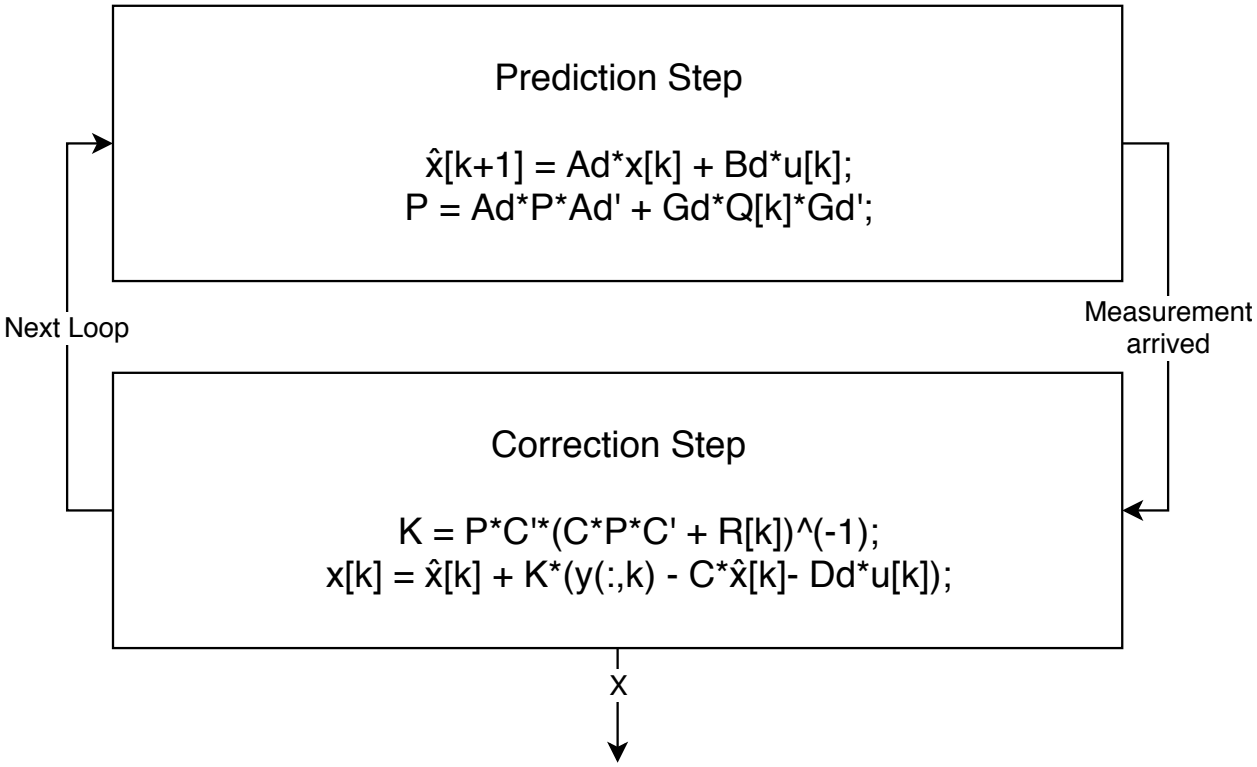

Figure 2.4: Kalmanfilter

Chapter 3

Implementation

3.1 Sensor Models

How the concept of the different sensor models work is described in chapter 2. Following here, the implementation which is used in the simulation will be stated in detail. First in general for all sensors, following by the different characteristics of each sensor.

3.1.1 Perfect Sensor

In general, the perfect sensor data are calculated like stated in chapter 2. In figure 3.1 those generated sensor data as well as the trajectory used for this can be seen.

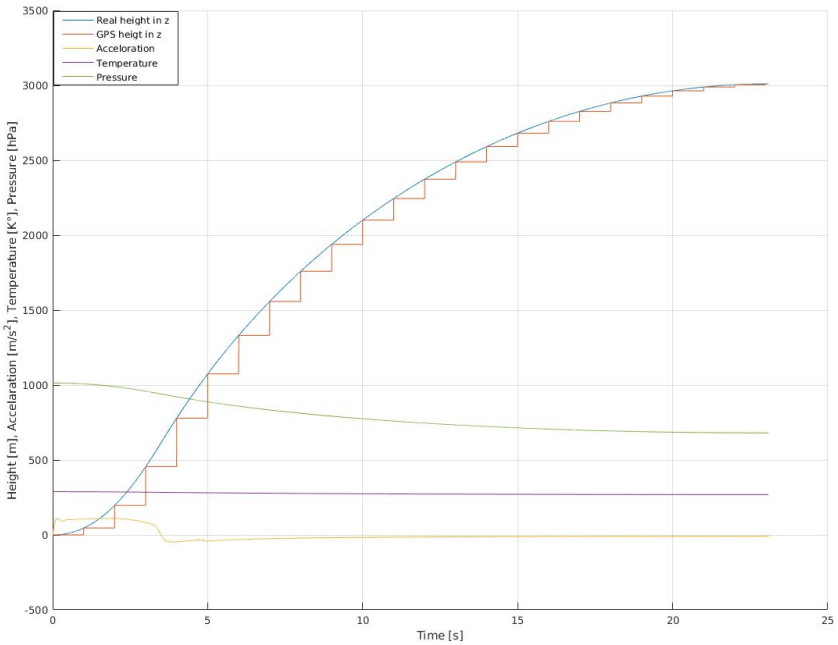

Figure 3.1: Generated sensor data

Accelorometer

Due to the fact that whole simulation works with discreet time stamps, the derivative of the height can not be done formally. So it is done by calculating the difference in each data point to the next and then weight those by the delta in time between them. This has to be done two times to get from the height to the acceleration. The unit for the acceleration in this simulation is meter per second squared.

Gyrometer

As stated before the pitch angle can not be directly generated. But if the data from the test flights are visited. It can be seen that the angle stays more or less the same wile the motor is burning. This make sense because during this time the main acceleration comes from one determinet direction and stabilices the rocket. After the burnout the pitch angle does change more or less randomly depending on strength and direction of the wind that hits the rocket. To reanimate this the values are generated randomly and the low pass filtered with a moving average filter to represent that behaviour. While doing this the random values are kept small during the burning of the motor and exceed afterwards to higher values.

Barometer

The measurements from the barometer are depending on the formula stated in chapter 2. For reasons of simlicity the start pressure is choosen as the mean pressure at sea level which is 1013.35 hPa. Also the temperature at the beginning is chosen as 288.15 degree Kelvin (15 degree Celcius) which also represents the mean value on the sea level. At least the temperature gradient is - 0.0065 °/m which is a common used value. For the state estimation in a test flight those values have to be determined before the start.

GPS

As stated in chapter 2 the GPS signal is just the height with a different sampling time. To maintain the vectors length which simplifies the later use in the estimation algorithm, the signal is acquired with a zero order hold conversion instead of a down sampling.

3.1.2 Noise

To generate the noise out of the data from the test flight, this has first to be extracted. It is assumed that the noise on the data is different depending on the state of the rocket (before Icognition, during motor burning, after burnout till parachute ejection), but it should have more or less the same properties between those events. Depending on this, the data vector have first to be separated in those different sections. For this the accelorometer measurements are iterated to find the time stamps on which those events happen like in figure 3.2.

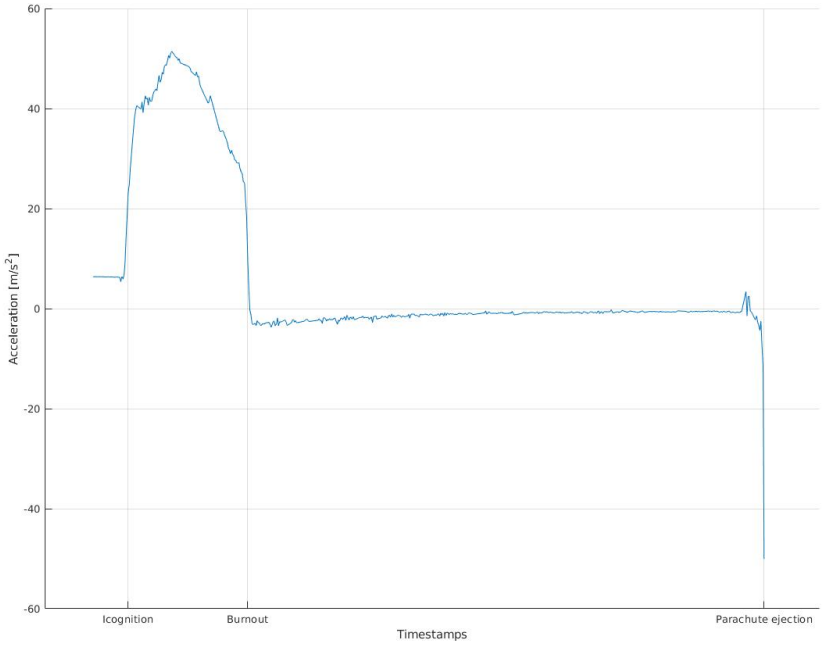

Figure 3.2: Timestamps drawn out of acceleration measurements

If done so, polynoms are fitted on this measurements with the least squared error method. Those polynoms represent the function which is assumed to be the noiseless data with possible offsets. So if now the test flight data is subtracted by those polynomial curves which results in the noise without a mean.

From this point on this noise can be examined on its parameters, like the power density, the probability distribution and the variance figure 3.3.

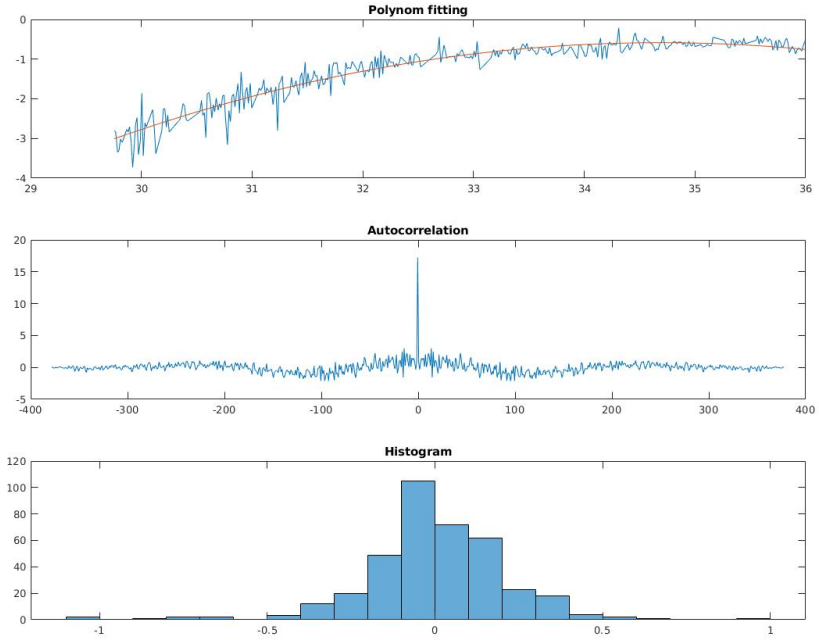

Figure 3.3: Polyfit Autoccorellation and Histogramm

This noise data can now used tho solve the yule walker equation to get an AR-model. For this the aryule function in can be used which estimates an AR-model of the order N as well as the variance directly out of the noise vector. But first, the data has to be resampled so that the AR-models can be used proper in the simulation. With those AR-models, the noise can be regenerated by filtering white noise with the correct variance. This generated noise can now be compared to the noise form the test flight data.

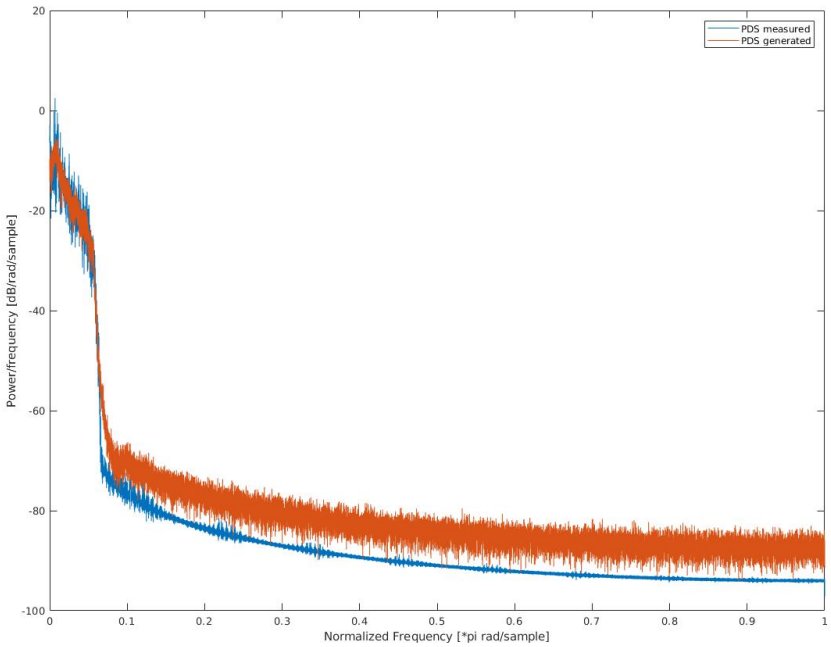

Figure 3.4: PDS form the measured and generated noise

As seen in figure 3.4 those noise resemble each other in their power density spectrum much more than the white noise would. So this AR-model is exported to the simulation script and can be used there to generate the real sensor data.

Accelorometer

The noise which is on the accelorometer is special because it often has a drift which results in a more or less constant offset. To recreate this, the offset can be estimated from the test flight data. Especially the data before the ignition are help full, because the value that should be measured is known.

Gyrometer

For the gyrometer noise a separate script was written to calculate the proper pitch angle and filter out the offset before generating the AR-model. This because the gyrometer measurements which are available are in degree per second and have therefore to be integrated before they resamble the correct pitch angle. In addition it is complicated to define which part of the measurements are noise and which is the ground truth. It was found that the estimated AR- model could not regenerated the noise with proper capacities. Because of that, the noise has to be low pass filter afterwards to resemble the noises better. For this task a IIR filter of order 200 was found best. Therefore those measurements have to be properly handled.

Barometer

First there are two or more barometer which sample on different frequencies ant have therefore also different accuracy's. This is represented in the way that the variance of the noise from the slower sampled barometer is kept on a smaller value, than that of the sensor which is faster sampled.

GPS

The GPS noise capacities were found with measurements that were taken for a longer time period while the GPS receiver at the same place. So the noise will have the same capacities over the whole flight. This due to the assumption that the GPS measurements should be independent from the motors vibration and the rockets posture changes. This should be suitable as long as the receiver does not lose its fix.

3.1.3 Real Sensor

To now generated the real sensor data, the different noises have to be generated like stated above. For this a vector of normal distributed random values is generated and multiplied by the square root of the corresponding variance. This white noise is now filtered by the corresponding AR-model and can then be added onto the corresponding perfect sensor data. This now results in the real sensor data.

Accelorometer

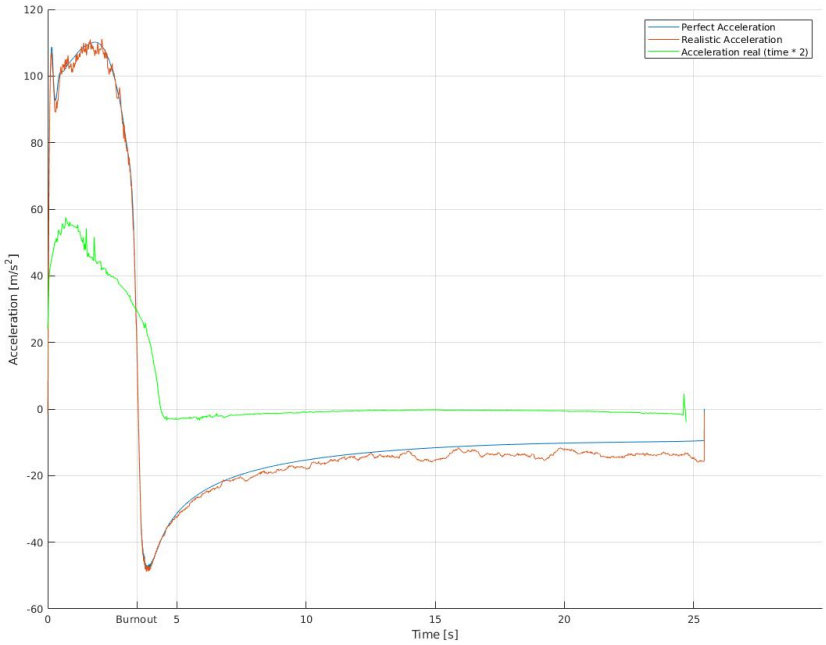

Figure 3.5: Plot of perfect acceleration vs realistic vs measured

In figure 3.5 the different generated values as well as the data from a test flight can be seen. The time vector from the test flight was stretched by the factor two to make the observation easier. Also it has to be said that the test flight was with a smaller rocket which flew only at an apogee of around 300 meters. This explains why the acceleration is as great as in the generated data and why the time vector had to be stretched. But the plot shows that the noise as well as the perfect data resemble the acceleration from the test flight in a appropriate way.

Gyrometer

Figure 3.6: Plot of perfect gyrometer vs realistic vs measured

Figure 3.6 shows the generate gyrometer data. Like in the acceleration plot the time vector from the test flight data was adjusted for better observability. It should also be explained due to the proberty of the pitch angle (more or less random depening ond air current etc) that the generated realistic pitch angle must not resamble the measured angle in its specific value. Important is in first hand that the noises have the same capacities which they do. Also it can be seen that the assumption that the angle does no change great during the burning of the motor is despite a quick change at the start appropriate.

Barometer

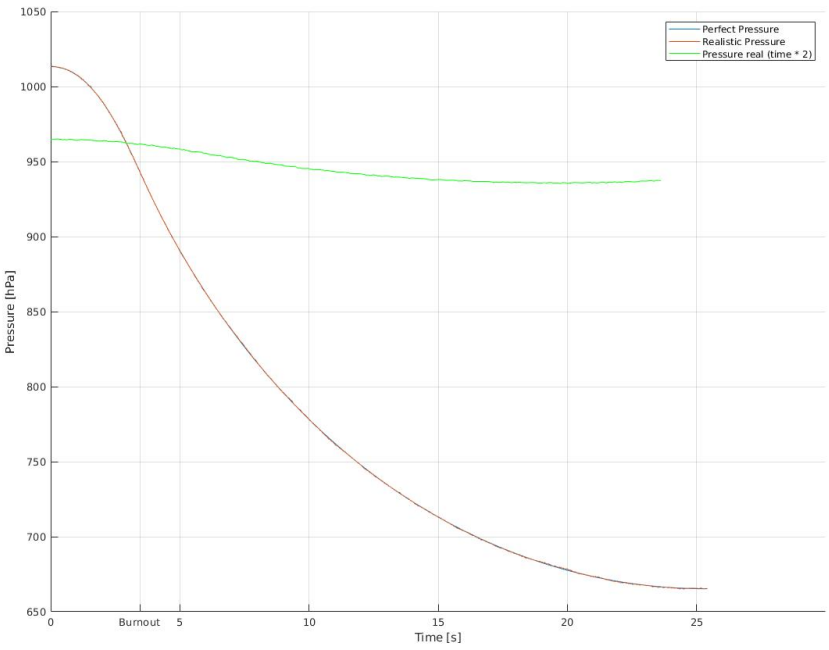

Figure 3.7: Plot of perfect barometer vs realistic vs measured

The realistic pressure measurements from a barometer are shown in the figure 3.7. The noise itself does not to see as it would have a great impact on the perfect data. But this decieves because the pressure does change by around 350 hecto Pascal during the upflight and therefore the changes from the noise which are around 1 to 3 hecto Pascal can not be seen that good. The comparison with the real measured data (in the figure also with a stretched time vector) shows that generated realistic measurement data does resemble real measurements.

GPS

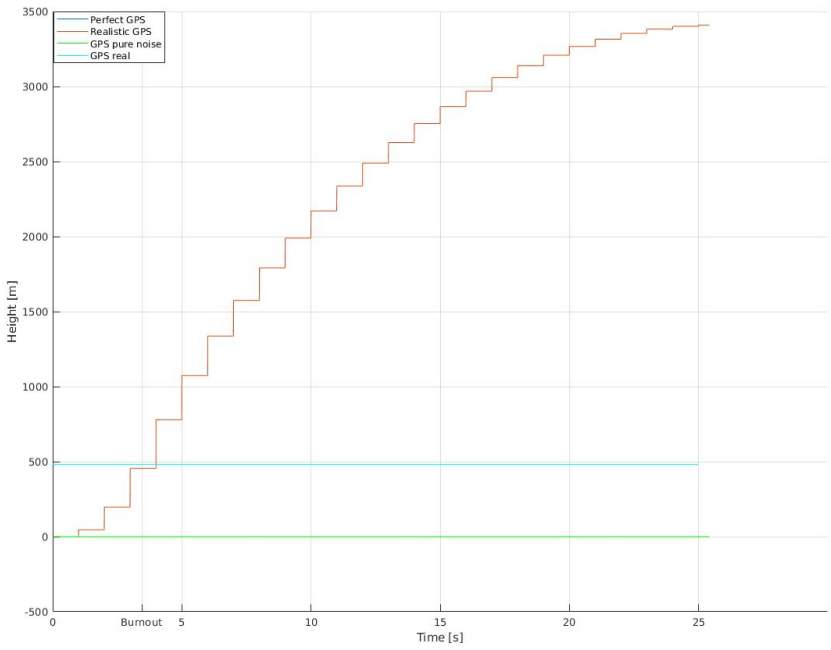

Figure 3.8: Plot of perfect GPS vs realistic vs measured

At least the generated GPS measurements can be observed in the figure 3.8. In this plot the pure generated noise was also plotted because as it can be seen it has real slow properties. For the comparison the measurements which were taken from a fixed position are plotted (because there were no usable GPS data from a test flight avaiable during the term of this thesis). It can be seen that the test data does also resemble the noise from the generated data.

3.2 State Estimation

As stated the used state estimator is a kalman filter with dynamic noises in the measurements as well in the systems. For this the noise matrices as well as the used loop is described below.

3.2.1 System Model

The models are modelled as stated in chapter 2 but from this different uses can be made out of this. For this simulation 8 models were implemented with each different capability.

Due to this, these different implementation are tested and evaluated in the next chapter.

3.2.2 Adjustment

Taken into account the above stated noise capacities of the measurements there were new systems developed which should hopefully result in better results.

Offset

The main adjustment is the inclusion of acceleration offset into the state vector. This is a common tactic used so that the state estimator can estimate the actual offset and therefore the impact of the offset can be minimised [3]. Therefore the state vector of a point mass would look like this.

$$x = \begin{bmatrix} h_z \\ v_u \\ a_z \\ a_{offset} \end{bmatrix}$$

While the dynamic matrices A and B would stay the same apart form an additional dimension of zeros for the acceleration offset state variable.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The y vector would stay the same while the output matrix C^T will be adjusted so that both acceleration in the state vector are added into the accelerometer measurements.

$$y = \begin{bmatrix} h_{GPS} \\ h_{p1} \\ h_{p2} \\ a \end{bmatrix} \qquad C^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Acceleration as Input

An additional adjustment would be to consider the measured acceleration of the rocket as input into the system. This should result into a system which could react faster to changes in the acceleration. For this the measurement noise of the accelerometer would have to be placed in the system noise matrix and therefore no additional system noise can be modulated. For a point mass this would result in the following system matrices. While the state vector and the A matrix would stay the same, the B vector would have to be adjusted like this.

$$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

In addition the y vector would lose its acceleration measurements and the output matrix C^T the corresponding dependencies.

$$y = \begin{bmatrix} h_{GPS} \\ h_{p1} \\ h_{p2} \end{bmatrix} \qquad C^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

3.2.3 Measurements noise

The matrix on each timestamp is rather easy to get in the simulation because the perfect measurements are known.

First the variance over the burning of the motor as well as over the up flight is calculated seperatly. After that those values are used to generate a noise vector for each measurements. In addition in the implementation the noise vector have the same length as all other used vectors. These noise measurements are then catognated into a diagonal noise matrices.

If the noise matrix is displayed as a diagonal matrix, it equals in the assumption that the noises from the measurements are independent from each other. This assumption can be made cause of the fact that each measurements expected those from the barometer are made from different sensors. The barometer measurements are the pressure as well as the temperature. Due to the fact that the temperature is not used for the state estimation, the measurements matrix can still be assumed as diagonal.

Different Sampling time

In addition the measurement noise matrix can be used to adjust for the different sampling times of the sensors. This is used for the barometers as well as the GPS sensors which are sampled slower as the state estimation itself loops. It is achieved by maxing out (setting to the highest possible value) the corresponding variance in the measurement noise matrix R if no actual measurements are available. As it can be seen in the formula to calculate the kalman gain K,

$$K = P \cdot C^T \cdot (C \cdot P \cdot C^T + R)^{-1}$$

maximal values in the R matrix result in relative zero value in the corresponding K matrix value. Those near or exactly zero values results in ignoring the corresponding measurements from the y vector as it can be seen in the measurement update equation.

$$x = \hat{x} + K \cdot (y[k] - C^T \cdot \hat{x})$$

To achieve this the noise vector from those measurements are generated with this taken into account to achieve the same vector length as all other vector used in the state estimation. In the implementation in a embedded system this can simply be achieved by an if statement which switches the R matrix value to the normal variance if a measurements arrives.

3.2.4 System noises

The system noise describes how uncertain the system model is in comparison to the real system. For this each entry in the diagonal resembles the variance of the noise on the corresponding state variable. In other words the system noise describes how far away from the predicted value the actual value can get in the next loop iteration. For the system noise the behaviour of the system during the flight has to be examined. This can be done in different ways. The first way would be to view the different ground truth curves of those state variables, which do have system noise acting on them (acceleration, pitch angel, pressure if linearized). For example figure 3.9 which represent the pitch angel. In the system models there are no influences on this

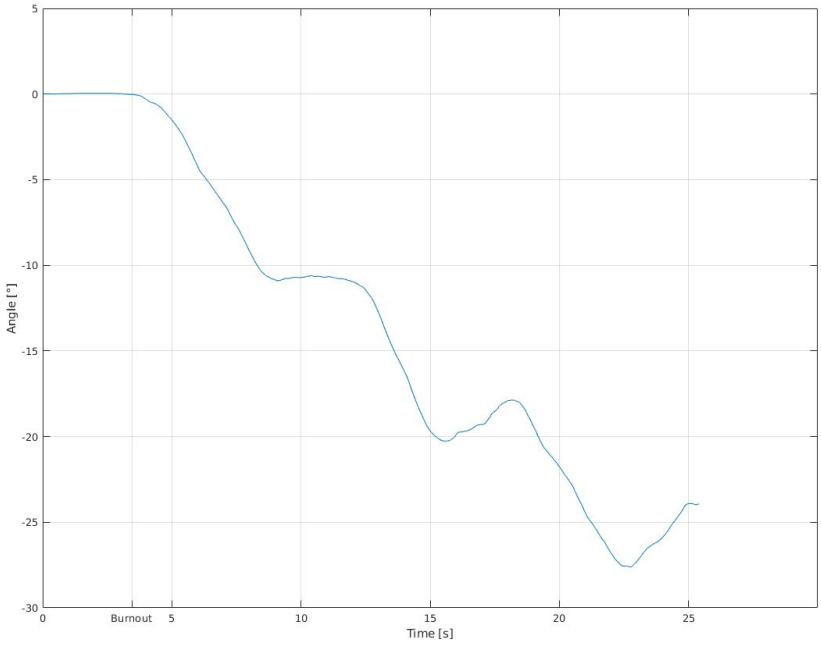

Figure 3.9: Plot of pitch angle

state value except for the measurements. So to describe the changes of the value which are observed with the measurements a system noise has to have an impact on the pitch angle. As it can be seen in the plot the angle does not change in a great manner until the burnout so the system noise till the burnout should also be rather small. After the burnout the angle changes a lot more but keeps changing in the same way so this would reassemble a greater noise as before the burnout but the noise should be more or less the same over time.

As a second attempt since the system noise describes the capability of the value to change over time independent from the dynamic system description it can also be achieved due deviating the ground truth value vector. For this the following matlab code can be used.

```
ACEL = abs(diff(a));          % Deviating the gournd truth acceleration
ACEL = filter(ones(1,100)*1/100,1,ACEL);  % Low pass filtering
ACEL = [ACEL ACEL(end)];       % Maintain vector length
```

Listing 3.1: System noise generation with deviation

This shows that after the deviation the absolute value from those is taken since a variance cannot or should not be negative. After that the values are low pass filtered to smooth out more steady system noise description.

3.2.5 Sensor Outfall

An additional interesting scenario which can be observed with the simulation is the outfall of sensors. This is needed to test the reliability requirements which states that the algorithm should still be working (with less accuracy) if 2-3 sensor fail. For this it has to be said that it has to be detected that a sensor fails to adjust the estimation algorithm. If done so the variance can be adjusted for this sensor in the same way as stated above, by maxing its variance out.

It is achieved by a simple if statement which does exactly that if a sensor fail is recognised.

3.2.6 Loop

Finally the state estimation is implemented in a simple loop which iterates trough each given time stamp. First the needed vectors and matrices have to be initialised with the right value. In the most system model versions the u vector remains zero while all measurements are brought into the estimation loop trough the y vector. In the others the acceleration and the pitch angle are brought into the estimation loop over the u vector an the remaining measurements trough the y vector. Also the current state vector x has to be initialised with the value that those states have at the start, which is presumably zero for all states except pressure and temperature.
The loop itself calculates the equation as they were stated in chapter 2. In addition if values from the measurements can not be transformed into the state vector values directly or with a linear calculation, they have to be transformed first before entering the system. For example pressure and temperature into height or acceleration and pitch angle into pure vertical acceleration.
Below is an example for the estimation loop for a rank five system. It contains height, speed, acceleration, acceleration offset and pitch angle as state variables.

```
% Initalzation
u = zeros(1,length(TimeVec));                          %Input vector is zero
y = [h_mes_GPS;a_mes;p_mes_1;p_mes_2;phi_mes];          %Output are the measurements
x = [0;0;0;0;0];                                       %Start Vector should be like this
P = eye(5);                                            %Standart can maybe be increased
Height1 = 0;
Height2 = 0;
Temp = T(1);

% Estimation loop
x_est_loop = zeros(size(x,1),length(TimeVec));         %Vector for the SE values
for k = 1:length(TimeVec)
    K = P*C'*pinv(C*P*C' + R_dyn_m(:,:,k));
    Height1 = CalcHeight(Po,p_mes_1(k),Temp,0,true,TgradSimu);
    Height2 = CalcHeight(Po,p_mes_2(k),Temp,0,true,TgradSimu);
    acc = a_mes(k) * cos(x(5)*pi/180);
    x = x + K*([h_mes_GPS(k);acc;Height1;Height2;phi_mes(k)] - C*x);
    P = (eye(5)-K*C)*P;

    x_est_loop(:,k) = x;                               %Save data from the Sensor fusion

    x = Ad*x + Bd*u(k);
    P = Ad*P*Ad' + Gd*Q_dyn_m(:,:,k)*Gd';

end
```

Listing 3.2: State Estimation Loop

Chapter 4

Tests

4.1 Point Mass

The first system model to test was the simple point mass as stated in chapter 2. While this implementation is a simple version it does already work surprisingly good. The mean error over the whole flight is normally around 4-5 meter. It has to be said that this only works while no sensor has a greater offset. Therefore these values come at the cost that they are not that trustworthy, because the system noise on the acceleration has to be set to a greater value to get those good estimation. An additional factor is also the pitch angle which does hinder this estimation if it changes in a great manner. This can be seen in figure 4.1 which shows the state estimation error with different offsets.

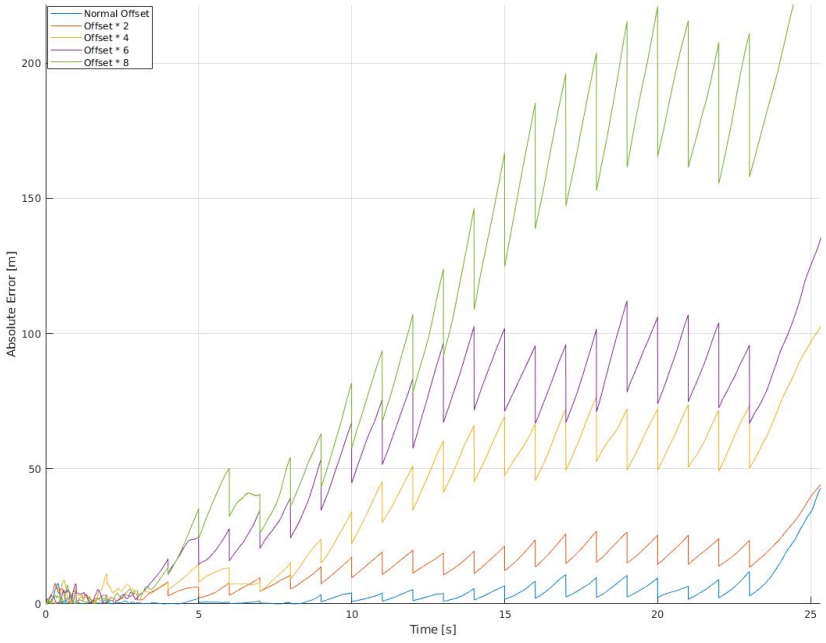

Figure 4.1: Error during flight time with different offsets

The table shows the mean and median of the error in the height depending on the offset on the accelerometer.

	Mean	Median
Normal	4.57	2.80
2Times	13.97	14.66
4Times	39.31	46.99
6Times	59.35	71.90
8Times	107.21	102.15

This shows that the error which the estimator makes does rise exponentially.

4.1.1 Performance

4.2 Point Mass with Acceleration Offset

While this system works also sometimes no so good as the first most times it works much better and therefore this should be implemented in the final system. Like above the figure 4.2 shows the error during a flight with different sensor offsets. It can clearly be seen that while the mean error does rise about some value, it always get back to zero and therefore overall this system preforms better as the simple point mass.

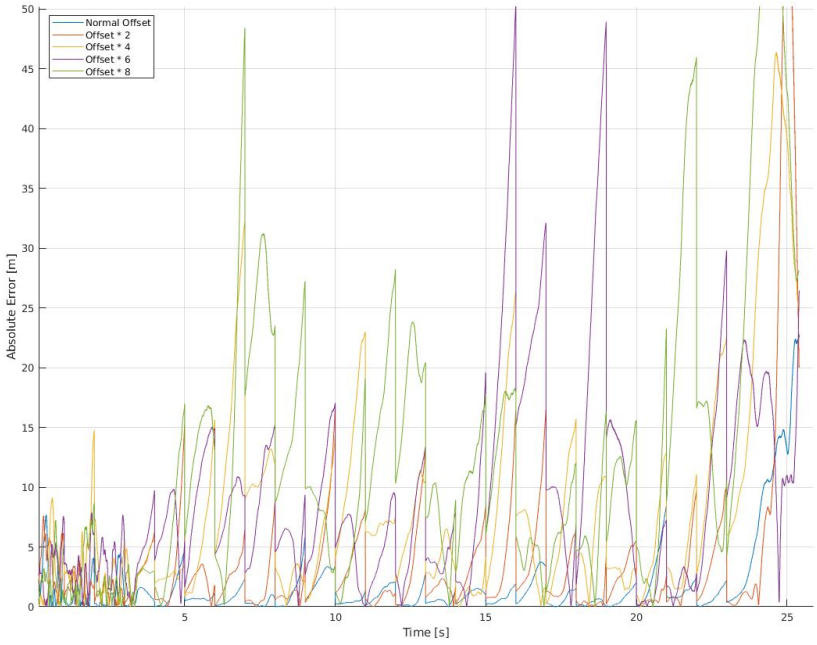

Figure 4.2: Error during flight time with different offsets

4.2.1 Performance

While it performes much better than the only point mass system it should be much room for above.

4.3 Point Mass with Pressure

The pressure used as state vector has to be shown that it is not that good as firstly tough. While it does increase the accuracy it does also increase the needed computational effort. In addition when the temperature gradient is chosen wrong it deeply effects the estimation like seen in figure. This is a problem due to the fact that the temperature gradient will most certanly not be correct during the whole flight. The assumption here is that it will be off around 5 percent. With this can be seen that the system performance is more ore less the same as the normal point mass system. Therefore the cost ond problems that can occur outweight the gain here.

It is the opposite a system which uses the pressure in the state vector does normally perform. This mainly due that the noise on the measurements with the interpolation from the linearized factor is worse than the interpolation with just the integrated accelerometer measuremnets. Therefore this will not be used in the best performance system.

4.4 Point Mass with Pitch angle

The pitch angle is difficult to estimate because it has no measured dependencies on its own. Therefore the kalman filter does just something like a real time low pass filtering. So the system noise on the pitch angle has to be as good calculated as possible to optimise its estimation. On the other hand in this implementation

the pitch angle does more of less the same thing as the acceleration offset because of it properties. Due to that, it should be enough to just implement one of them I think

4.5 Point Mass with Acceleration as input

It can be seen in the plot in figure bla, that the estimated height estimated when the acceleration measurements are taken as a input that they reseamble more or less the exact values as from the normal system. The difference between int the errors between this system model and the normal point mass model is mostly due rounding errors of the simulation then due to really better estimation. This can be assumed due to the fact that this system model performs some time slightly better and some time slightly worse than the point mass system- Therefore there is no real gain in the implemenation this way. In addition with this version the additional adjustment factor which was the system noise onto the acceleration can not be used. So it does not really make sense to use this in the final version.

Makes no real difference while losing the possibilitey to make system noise because it is used for the measurment noise.

4.6 Point Mass with offset and better calculated system noise

The better calculated system noise first by accesing it in the second was stated in **??** This by calculating the discrete system noise matrix with the integration as well as derive the perfect measurements and then low pass filter them to get better system noise vectors. This had to be used on a system which uses the acceleration offset as well in the state vector to get fully possible gain out of this implemenation. Figure bla shows that an overhall better estimation can be achieved with this tactic. This mostly due the better system noise vector which are much better estimated this way. Also the additional effort to access this system model only occurs on the preperation, while it does have no effect on the computational effort during the flight itself. Non the less this expansion of the state estimation should be used any time if available.

4.7 Best Performance System

Works really great I think It was tested with and without GPS because it is not sure that this would function

4.7.1 With GPS

4.7.2 Without GPS

4.8 Sensor Outfall

4.8.1 GPS Outfall

4.8.2 Barometer Outfall

4.8.3 Accelerometer Outfall

4.8.4 Gyrometer Outfall

Chapter 5

Conclusion

5.1 Derived Solution

Here comes the best finded state estimator loop/solution and how an why it works Again

5.2 Comparison Solution/Requirements

Compare the derived solution to the requirements

5.3 Outlook

5.4 Thanks

Thanks to the Aris Team espacially Thomas and Fabian Also Thanks to Lukas and Papa Moll
And to the EPFL team for the test flight data

Bibliography

[1] Tong Minh Bryan. Real-time position and attitude determination of the stratos ii sounding rocket, 2012.

[2] Spaceport America Cup. Intercollegiate rocket engineering competition rule and requirements document. Retrieved from `http://www.soundingrocket.org/sa-cup-documents--forms.html`, 2018.

[3] W.Schultz David. Application of the kalman filter to rocket apogee detection. 2004.

[4] Dan Simon. *Optimal state estimation : Kalman, H∞, and nonlinear approaches*. 2006.

Appendices

This is the Appendix