Lucerne University of Applied Science & Architecture

ARIS - Data Fusion for a Sounding Rocket

Bachelor Thesis

Author:	Michael Kurmann
Department:	Electrical Engineering and Information Technology
Supervisor:	Prof. Marcel Joss
Expert:	Werner Scheidegger
Industrial Partner:	ARIS (Akademische Raumfahrt Initiative Schweiz) Oliver Kirchhoff
Submission date:	June 8, 2018
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, June 8, 2018

Abstract

The Academic Space Initiative Switzerland (ARIS) is a student association which competes every year in the Intercollegiate Rocket Engineering Competition (IREC). in which the aim is to build a rocket that can fly autonomously to an exact apogee of 10 000 feet (= 3048 meter). To fly to this height a control loop has to be implemented which needs as accurate information as possible of the current height. To achieve this a set of different sensors are built into the rocket. The aim of this thesis is to find an algorithm and implement a simulation in which the data of the sensors can be fused to estimate the height as good as possible and prove the suitability of the algorithm with the simulation.

The first step to achieve this was to develop a validation concept. The developed concept uses a known trajectory to produce a perfect sensor dataset simulation. This perfect sensor dataset is then augmented with noise to a more realistic sensor dataset. For this the logging data from test flights has been used to produce AR models which are then used to generate realistic noise to be merged with the perfect sensor dataset. After that this produced realistic sensor dataset has been fet into different versions of a state estimator which represent different algorithms that try to reproduce the current height of the rocket as accurately as possible. The estimated heights from those state estimators have then been compared to the trajectory that was used in the beginning.

As the best fitting fusion algorithm a discrete Kalman filter was found. It works with dynamic noise matrices to compensate for different sampling speeds of the sensors. The Kalman filter itself does work with an rank 5 system which consists of the height, speed, acceleration, acceleration offset, and the air pressure. This solution provides an in perspective simple algorithm which with the help of the simulation is easy to adjust. That simulation showed that this algorithm will work on an embedded system and helps, that it can be used again in coming competitions with reasonable effort.

Contents

Shortcuts and Variables

Shortcuts

ARIS	Akademische Raumfahrt Initiative Schweiz
IREC	Intercollegiate Rocket Engineering Competition
SAC	Spaceport America Cup
ERT	EPFL Rocket Team
EPFL	École Polytechnique Fédérale de Lausanne
GPS	Global positioning system
FIR filter	Finite impulse response filter
IIR filter	Infinite impulse response filter
AR model	Auto Regressive model
ARMA model	Auto Regressive Moving Average model

Variables

A	System matrix
Ad	Discrete system matrix
B	Input matrix
Bd	Discrete input matrix
C	Output matrix
D	Throughput matrix
x	State vector
u	Input vector
y	Output vector
G	System noise input matrix
Gd	Discrete system noise input matrix
Q	System noise matrix
R	Measurement noise matrix
K	Kalman gain matrix
P	Error covariance matrix
P0	Pressure at ground level
T0	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
$a_1 \dots a_M$	FIR filter coefficients
σ_ω	Standard deviation of noise ω
γ_{yy}	Autocorrelation of y
h	Height
v	Speed
a	Acceleration
p	Pressure
φ	Pitch angle
KP_v	Pressure gain depending on speed

Chapter 1

Introduction

1.1 Task

Figure 1.1: Official logo of ARIS [1]

The Swiss Space Initiative ore Akademische Raumfahrt Initiative Schweiz (ARIS) is a student association which takes part in the yearly Spaceport America Cup (SAC). In detail they compete in the Intercollegiate Rocket Engineering Competition (IREC). The goal of this competition is to build a rocket that can fly autonomously to a predefined apogee (10 000 feets = 3048 meters) and after that return safely back to the ground with the help of parachutes. Additionaly the rocket has to be able to transport a specific amount of payload with it. There are a total of 1 000 points to achieve in the competition which are based on different parts depicted in Table 1.1.

Description	**Points**	**Percent**
Entry form and 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

It can be seen that just halve of the points are assigned to the performance at the competition itself. The other halve of the points can be achieved by teamwork, professional documentation and good engineering during the development and construction of the rocket. For the 500 points which are assigned to the flight performance, 350 are based on the error made between the targeted and approached apogee. These points are calculated like this:

$$Points = 350 - \frac{350}{0.3 \cdot TargetedApogee} \cdot |TargetedApogee - ApproachedApogee|$$

So there is a total of one point loss per 2.6 meters of deviation from the targeted apogee [3].

Figure 1.2 shows the rocket “Tell” which will be used in this year’s competition. To aim for the right apogee a control algorithm is implemented on a micro controller that is placed in the lower body of the rocket. This control algorithm predicts the to-be-achieved apogee depending on the current states of the rocket (height, speed and acceleration). It then drives the air breaks to adjust that predicted apogee to the aimed 10 000 feet. This algorithm relays on the information of different sensors to determine the actual state of the rocket. Because there are different sensors to measure basically the same states, an algorithm which fuses this data to only one state information set would come in handy. With this fusion algorithm it

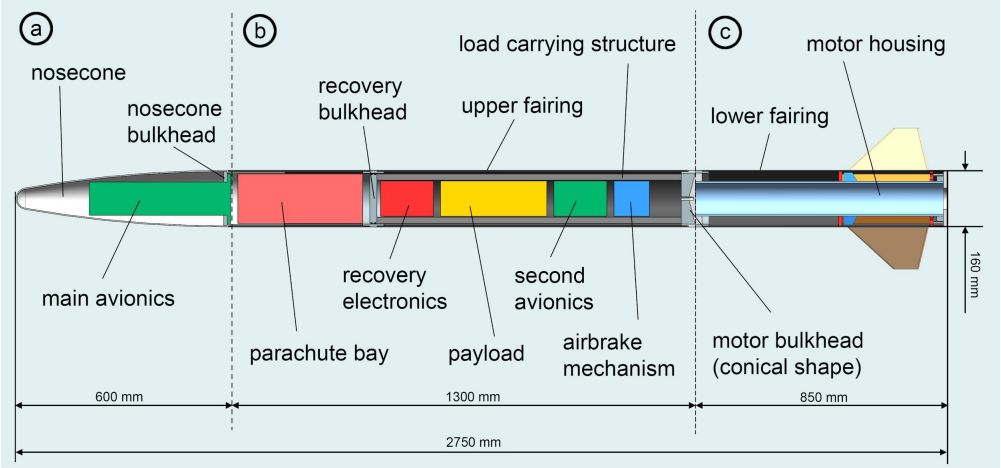

Figure 1.2: Concept of the 2018 competition rocket Tell

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 for the sensor data provided during flight and with this simulation's help find the algorithm which is best suited for this task.
For this the current situation of this year's project, the problem to be solved during this work and the desired solution will be described in this chapter. After that the models for the different sensors that will be used as well as the pros and cons of the different possible state estimators are discussed at the beginning of chapter 2. In addition, different possible system models are also described in that chapter. Chapter 3 will then show how the simulation and the fusion algorithm are implemented in detail. Furthermore will it discuss how the concepts that have been developed in the chapter before are integrated into the simulation. To verify that the implementation is working as intended, the results of the simulation and especially the performance of the different system models are discussed in chapter 4. Finally, in the last chapter 5 a summary of the achieved knowledge, a comparison between the desired and the implemented solution and an outlook for coming work on this topic will be given.

1.2 Purpose

The hardware as well as the most of the software parts that will be used for this competition have already been defined. In addition it is a suitable assumption that the sensors and the dynamics of the rocket will stay more or less the same for the competitions coming. Therefore this thesis will focus mainly on finding an algorithm for this given surroundings, raise the performance at the competition flight itself and with this optimise the points achieved in the design implementation part. On top of that this thesis will also try to find an as modular solution as possible, so that the achieved knowledge can be reused and refined in further competitions. For this the dependencies and possibilities of the different sensors should be developed such that they can be fused in an optimal way. The final product should consist of a suitable fusion algorithm as well as a simulation part which provides the tools needed to test and adjust this fusion algorithm.

1.3 Research

Sensor data fusion and state estimation is a well established engineering field. Especially since the 1960 when Rudolf A. Kalman first published his paper about the Kalman filter. Those fusion algorithms are especially established in rocket science since they were first developed for the tracking of flying objects. Therefore there is already a lot of previous work which can be used in this thesis. Especially two books provided most of the needed theory. The first one of them is [4] which contains the basic theory about state estimation especially with Kalman filters. The second book [6] is more focused on different approaches of state estimation and provides also different solutions to common problems that occur during the implementation of state estimations. In addition the Master Thesis [2] accesses more or less the same issue as this thesis does. Therefore in the conceptional part of this paper mainly this source will be used as a reference.

1.4 Sensors

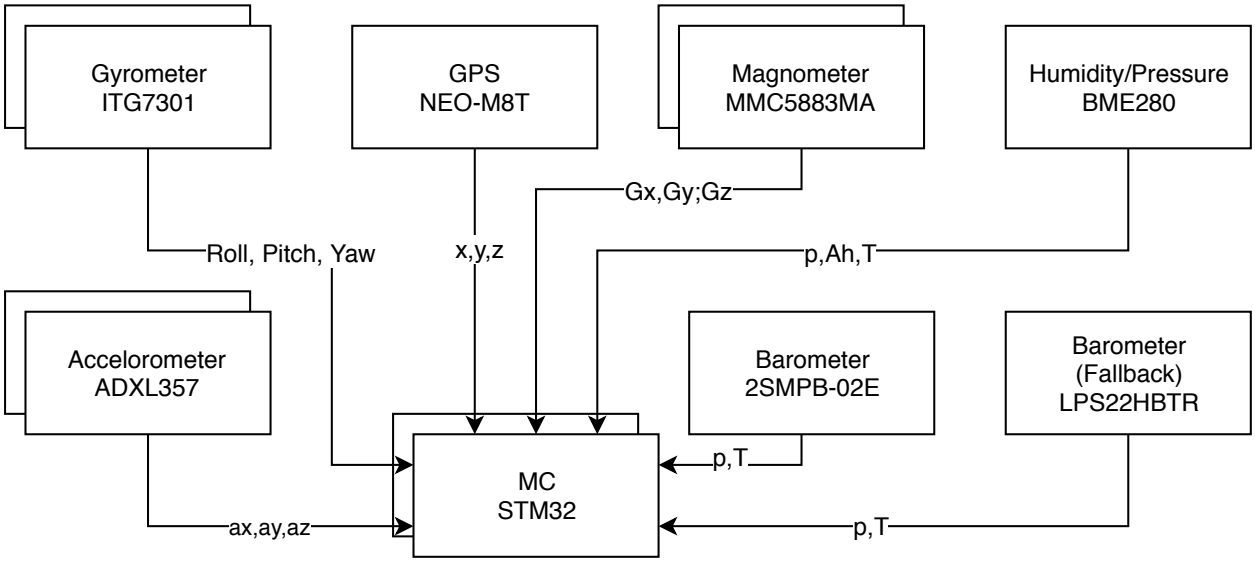

Figure 1.3: Sensor Network

As mentioned above, different sensors are used in this year's competition to measure the current state of the rocket. Based on the assumption that basically the same sensors will be used in the coming competitions too, they will act as the basis for the sensor fusion algorithm. This used sensors and their settings will be described in this chapter.

1.4.1 Accelerometer

First of all comes the accelerometer. Accelerometers are well established and widely used sensors which measure the force which is applied onto the sensor in the three space dimensions. This year's accelerometer is an ADXL357 which will be sampled at 1 000 Hz.

1.4.2 Gyrometer

The Gyrometer is needed to measure the rotational position 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 ITG3701 and it will also be sampled at 1 000 Hz.

1.4.3 Barometers

Barometers are widely used in aviation because with a common pressure model the height can be easily calculated out of the measurements that the barometer takes. In this year's competition two barometers are used. These are by model name a 2SMPB-02E and a LPS22HBTR. They will be used with a sampling rate of 100 Hz each. In addition the humidity and pressure sensor BME280 can be used as an additional source for the pressure too.

1.4.4 Temperature

The temperature needs to be taken into account to make the height calculated out of the barometer data more accurate. This is because most of the atmospheric models depend besides of the pressure on the temperature as well. This temperature will be provided directly by the different barometers which each posses a separate temperature sensor.

1.4.5 Magnometer

Also there are two Magnometers in the sensor network. They measure the strength of the surrounding magentic field. This can be used to determine the direction in relation to 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 in its calculations, this sensor will not be used for the sensor fusion algorithm.

1.4.6 GPS

For the next year's competition a differential GPS should be in place as well with the help of two μblock modules. This measurements are more precise as the rest of the sensors but are taken at a much slower sample rate around 0.5 - 2Hz. Therefore the algorithm should already take these measurements into account too and interpolate them with the data from the other sensors.

1.5 Problems

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

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 extract the strengths out of the different sensors and minimise their individual weaknesses. Additionally because this algorithm is system critical, it has to be reliable enough that it still works properly if any of the sensors are failing.

1.5.2 System Load

The cycling time will be around 1 ms on the embedded system. This time was chosen on behalf that it would be difficult to get the precisely needed cycling time to ensure the needed controllability of the rockets apogee. Therefore the system load that the algorithm will cause has to be strongly limited for it to be able to run on this given system. The system this year is a 32 bit ARM SoC which runs at 168 MHz. Assumed that the whole control software has at maximum half of a software cycle at his disposal the maximum clock cycles to be used for all calculations in one such software cycle is around 84 000. With this amount of cycles the processor can do around 10 000 simple calculations (addition, subtraction, multiplication or division), because its floating point unit needs on average around 8.5 cycles per operation (load, calculate and store). Of these 10 000 simple calculations at most 5 000 may be used by the sensor fusion algorithm. This number is just a rough assumption with the use of optimal performances Which means that the final system load should aim to not reach this value by an as great amount as possible or at least not exceed it.

1.5.3 Precision

The precision is after the system load the most critical attribute. If the algorithm fails to provide the required accuracy the control algorithm has no exact height estimation to rely on. The control requirements state that the maximum error between the estimated and the real height should not exceed two meters to be able to properly control the aim of the apogee. This accuracy is especially needed after the burnout when the apogee control with the air brakes will start.

1.5.4 Settling Time

The settling time is defined as the time span between the first reliable measurement after the burnout and the time the estimation reaches its required precision. This time span has to be small enough to ensure that the control has enough time to aim for the desired apogee. In the current system the burnout occurs around 3-3.5 seconds after ignition while the whole flight upwards only takes around 25 seconds. Therefore the settling should be finished at most one second after burnout so that the control has as much time as possible for the aiming. Best would be if the algorithm already achieves its target precision while the motor is still burning.

1.5.5 Reliability

Due to how a sensor package is placed into the rocket, the assumption has to be made that it will be possible that sensors fail in execution. Therefore the algorithm should be reliable enough to still provide the control system with the desired data with some of the sensors failed, so the functionality of the whole system reamins intact. But of course the provided data has not to be as accurate any more 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 the same over the next competitions, it is not certain that exactly the same sensors will be used. Therefore the presented algorithm should provide the possibility to adapt to new sensors in an easy way, as long as they resemble the old sensor in a feasible way. This will ensure a long term usability 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	< 2 m in height	High
Settling Time	Time from first reliable to optimal estimation	< 1 s after burnout	High
Reliability	Working estimation with # failed sensors	2-3 sensors	Medium
Modularity	Effort needed to change a sensor	< 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 one critical requirement, namely the system load. This one is given the importance "critical" because the algorithm needs to be efficient enough to run on the embedded system given. Any solution that would exceed the value given in this requirement in a great manner would be pointless in the frame of this thesis, because it will not be deployable to the target system.

Secondly there are two requirements which are tied together, the precision and the settling time. Where the precision describes what an optimal estimation is in regard to the scope of this thesis, the settling time relies on this precision beeing reached as soon as possible. The settling time is defined as the time span between the burnout until the best possible precision is achieved.

The modularity as well as the reliability requirements are more guidelines than hard requirements. This because they will still lead to a usable solution for this year's project and their aim is hard to define since it is depending on the coming competitions and their circumstances.

1.7 Desired Solution

The desired solution should meet all of the given requirements as good as possible. While doing this it should also not get more complicated than needed because an easy understandable solution helps to ensure its further use in coming competitions.

Chapter 2

Approach

This chapter discusses how the problem described in the chapter before will be approached by stating the concepts which were developed for the simulation of the desired solution. First the test concept and the concept to get the sensor models are described. After that, the different system models as well as the different state estimators are discussed.

2.1 Verification

First of all a test concept has to be defined by which the developed algorithms will be tested. This is also especially useful for future competitions to test the adjusted algorithms which will be used there.

For this the following concept which can be seen in figure 2.1 was developed.

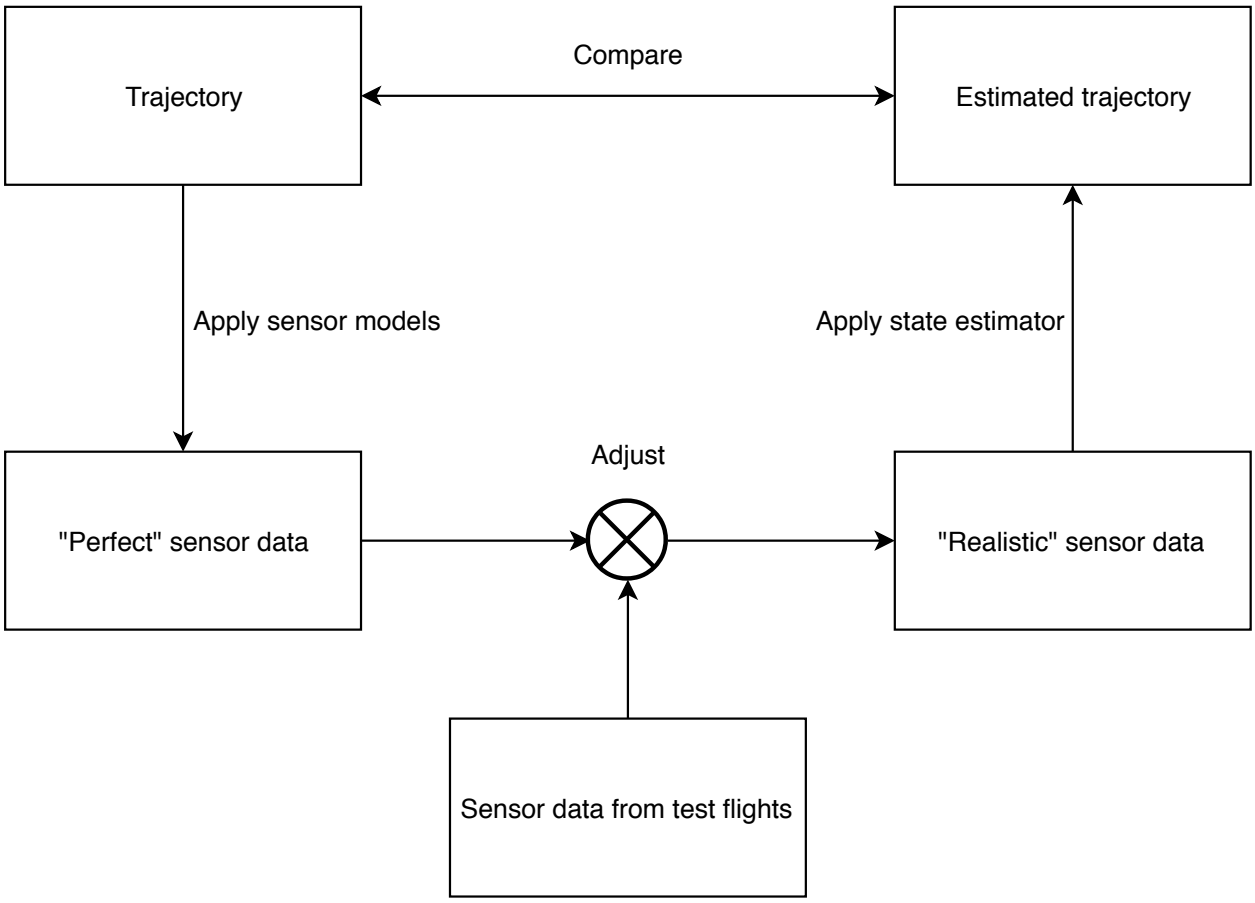

Figure 2.1: Verification Concept

The verification concept is based on a trajectory that is generated by a simulation, which should resemble a real trajectory as good as possible. This functionality was provided by the simulation team of ARIS from the last year's competition. The trajectory generated is then applied on the sensor models which are discussed below. The outcome is so called "perfect" sensor data which would represent the data provided by the sensor if they do not have any noise at all and the surroundings do perform as described. After that, noise is applied on this perfect sensor data which results in "realistic" sensor data. The inclusion of this noise will be discussed later in this chapter too. Basically this noise is drawn out of the sensor log data from the previous test flights. This "realistic" sensor data is then fed to the input of the different estimation algorithms which will provide their trajectory estimation at the output.

Finally, to verify the functionality of these algorithms the estimated trajectory is compared against the generated trajectory.

2.2 Sensor models

As stated above the trajectory will be generated by the simulation. This simulation includes just the information for the height, so the sensor models have to be adjusted to generate the different needed data. The models are defined in the following parts.

2.2.1 Accelerometer

Perfect measuring data from the accelerometer (a) is simply put just the two times deviated height (h) by time (dt).

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

This value equals to the straight upwards acceleration. To get the acceleration which would be provided taking into account the x- and y-axis, the pitch angle φ has to be calculated into this generated data like this.

$$a_{pitch} = \frac{a_z}{cos(\varphi)}$$

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 handedly by taking the gyrometer measurements from the testflights into account. It has also to be stated that only the pitch angle of the rocket which can be seen in figure 2.2 is of interest for this first sensor fusion implementation. Because of this, only the pitch angle will be generated with random values which drive towards a realistic value that was read out of the data from test flights.

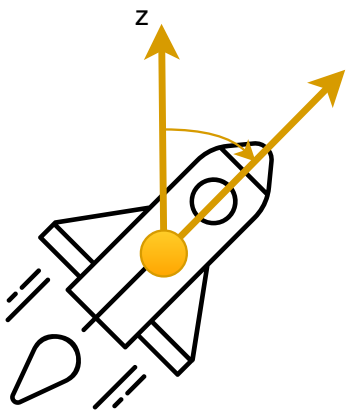

Figure 2.2: Pitch angle visualisation

2.2.3 Barometer

The barometer which are used in the aerospace usually provide the pressure in hP as well as the temperature in °C.

Pressure

To generated the pressure data the barometric height formula is used [5].

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

with: P0 = Pressure at ground level
T0 = 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 until 11 000 meters above ground under the condition that the temperature gradient is determined correctly.

Temperature

Calculating the temperature out of the height is not trivial because the temperature gradient depends 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 of the simulation with the right factor.

2.3 Noise Generation

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

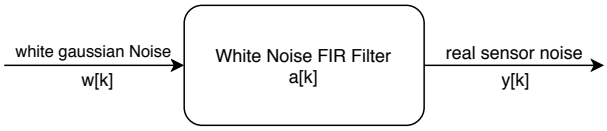

Figure 2.3: White noise filter concept

For this the Yule Walker Equations should come in handy to calculate a so called auto regressive (AR) model of the noise system. An AR model is an all pole finite impulse response (FIR) system $a_1 \dots a_N$ which generates a random sequence y[k] with a defined auto correlation value γ_{yy} out of white gaussian noise $\omega[k]$.

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

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 value γ_{yy} and therefore the same spectral capacities as the input sequence y. Because the AR model consists of an all pole filter the noise should have a steady mean value. Preferably it should be zero mean for the estimated AR model to best represent the real system. For this the mean value of the measurements should be calculated beforehand and then be subtracted from them to get zero mean noises. After the generation of the desired noise this mean value can be readded again 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 to prevent non linearities. The most important values to estimate are the vertical height and speed, so for a first implementation just variables that determine those both will be used. This values are mainly calculated from the height provided by the GPS, the vertical acceleration from the accelerometer as well as the pressure and temperature from the pressure sensors. In addition to that the pitch angle from the gyrometer is used to calculate the pure vertical acceleration. But even with these simplifications 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 quick view we have a look at the general notation of a state space system. First there is the update function:

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

The A matrix resembles how the system changes over time by itself where the B matrix describes how input u influences 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 affects the output while the D matrix describes how inputs directly affect the output. The D matrix is zero in most state space systems, because in reality there are not many systems where the output directly and immediately reacts to the input. The noise on the output (measurement accuracy) can be described with the R matrix.

2.4.2 Point Mass

The most simple possible model would be that the rocket is resembled by 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 general state space system to a 3x3 matrix with only two 1's in it. Also the input matrix would be set to a zero matrix in this model because the input process (power from the motor and drag force from the surrounding air) would be difficult to describe in such 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 following 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 usual 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 sensors could not be estimated correctly, because the value was transformed in a non-linear way before it entered the system. The same problem occurs with the accelerometer. If the rocket develops a pitch angle not equal to 0 while asccending, it would be measured wrong. To counter this error, the measurements of the accelerometer would have to be weighted by the angle of the gyrometer before entering the system. This weighting is also non-linear and the values of the gyrometer would not be filtered, which will make the estimation even more uncertain.

2.4.3 Point Mass with Pressure

To take into account the problems described above, the 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 problems 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 is shown in the matrices below where the deviation of the pressure is also depending on the current speed with a gain factor of KP_v which has to be defined depending on the temperature gradient. Based on such a linearisation the barometer measurements can be interpolated so the actual pressure is available at any loop iteration. This results in a 4x4 A matrix and also an all zero B vector for the dynamics equation which look like this:

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

This would also reflect on the output matrices which will directly contain the pressure. In addition the height calculated from the pressure can be inserted as an additional measurement 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 except of an additional dimension.

2.4.4 Point Mass with Angle and Pressure

The same solution as above can also be applied for the pitch angle. But its linearization and dependencies are more complicated and can therefore not be directly put into the system equation. This would change the state vector from above into the following.

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

In consequence the dynamic part for A is extended into a 5x5 matrix while the B matrix still would be a zero vector.

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

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

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

With this the pitch angle would be some sort of a real time low pass filter. This solution takes into account all measurements available for the needed values. It also keeps the calculation in the system as simple and linear as possible.

2.4.5 Discretisation

The systems stated above are all in continuous time state space description. To implement them on a discrete time system like a micro controller they have to be discretised. Those discrete matrices will be denoted with an additional lowercase d (A -> Ad). The general state space discrete description with noise looks as follows.

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

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

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

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

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 well 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 (the value is measured in an infinitely small instant). This calculation is rather far away from the reality because no sensor can measure with perfect dirac impluses. 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 as it 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 [4].

They can also be used in the same way to calculate Gd by changing B into G and Bd into Gd. Both version have to be tested in the simulation to see if the additional effort for the second version pays off enough.

2.5 State Estimator

There are several possibilities to generate the required 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 probable high precision and efficency, the time and knowledge needed for this task would exceed the resources given for this thesis by far. Also as stated in chapter 1 a lot of theoretical as well as practical prework has already been done and therefore should be used. So in the following different possible approaches with their pros and cons will be discussed.

2.5.1 Kalman Filter

First of all the traditional discrete Kalman filter has to be discussed as it provides the base for most known state estimators. 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 exactly there lies the problem: a physical system is seldom linear. Also the estimated system as well as its variances over the time have to be known to provide an optimal estimation. If the noise matrices of the system are static, the filter's gain matrices aim for a fix value and can therefore be calculated in beforehand. This reduces the computational effort by a significant amount [4]. It should be mentioned that even if the noise is not Gaussian the Kalman filter is still the best linear estimator as long as the system and its properties are well known [6]. To summarise it, the discrete 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 put in simple terms three Kalman filters in one. Where the main filter is used as a normal Kalman filter as 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 also to be estimated. But higher accuracy comes to a cost, this sensor needs more computational effort to perform this estimations, because of the three Kalman filter equations which have to be calculated each step[4].

2.5.3 Extended Kalman Filter

The extended Kalman filter provides additional parts to better access non linearities in the observed system. This by not estimating the state of the system but by estimating the linearised change of the state to the next coming state. For this the systems equations have to be derived around the current nominal point in every estimation state. This is some sort of bootstraping solution because the nominal points on which the derivation happens are estimated in the process and these estimates are then used to estimate the change between this estimation and the next. Therefore the computational effort needed raises even further because each function has to be deviated at each loop iteration [6].

2.5.4 Unscented Kalman Filter

The unscented Kalman filter uses the unscented transformation 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 the new state is estimated by weighting those new values depending on their probability of occuring. Using this approach the state function does not have to be linearized and therefore the estimation is most of the time better than that of the extended Kalman filter. But to achieve this the unscented Kalman filter needs to apply the unscented transformation onto the state vectors in each iteration. As explained for this it calculates the estimation for many different values and does therefore need even more computational effort [6].

2.5.5 H∞ Filter

The H∞ filter is a more diverse approach than the ones described above. It was developed if the observed system and especially its noise is not well known. Put into other words it was developed to resemble an as robust as possible state estimator. In addition its stability can be easier guaranteed than that of a Kalman filter. But because he consists of additional tuning parameters to achieve this, he is more complicated to use than a Kalman filer. On the plus side the H∞ filter minimises the worst-case estimation error in terms of the standard deviation of the estimation error compared to a Kalman filter [6].

2.6 Choosing

By taking the requirements table 1.2 into the consideration for finding the optimal solution, two main requirements define this decision. The first one is the system load which is a critical requirement and has therefore to be achieved. The second one to specially take into account here is the modularity, so the algorithm should be as simple as possible. Taken into account that the system is more or less well known and that the noise can be determined with the simulation and the log data from previous test flights, a normal Kalman filter seems to be the most fitting solution. A more complicated approach should be avoidable, because the performance of the rocket and the sensor should stay the same during each flight. Because of this decision only the normal Kalman filter well be discussed from here on to fit the time limitations given for this thesis. Nonetheless the developed simulation should provide the possibility to implement and test the other state estimators in a later instalment.

2.6.1 Functionality

Since the best fitting state estimator has now been chosen, its detailed functionality shall be described here. This algorithm works in four main equations which can be divided into prediction and correction steps, as depicted in Figure 2.4.

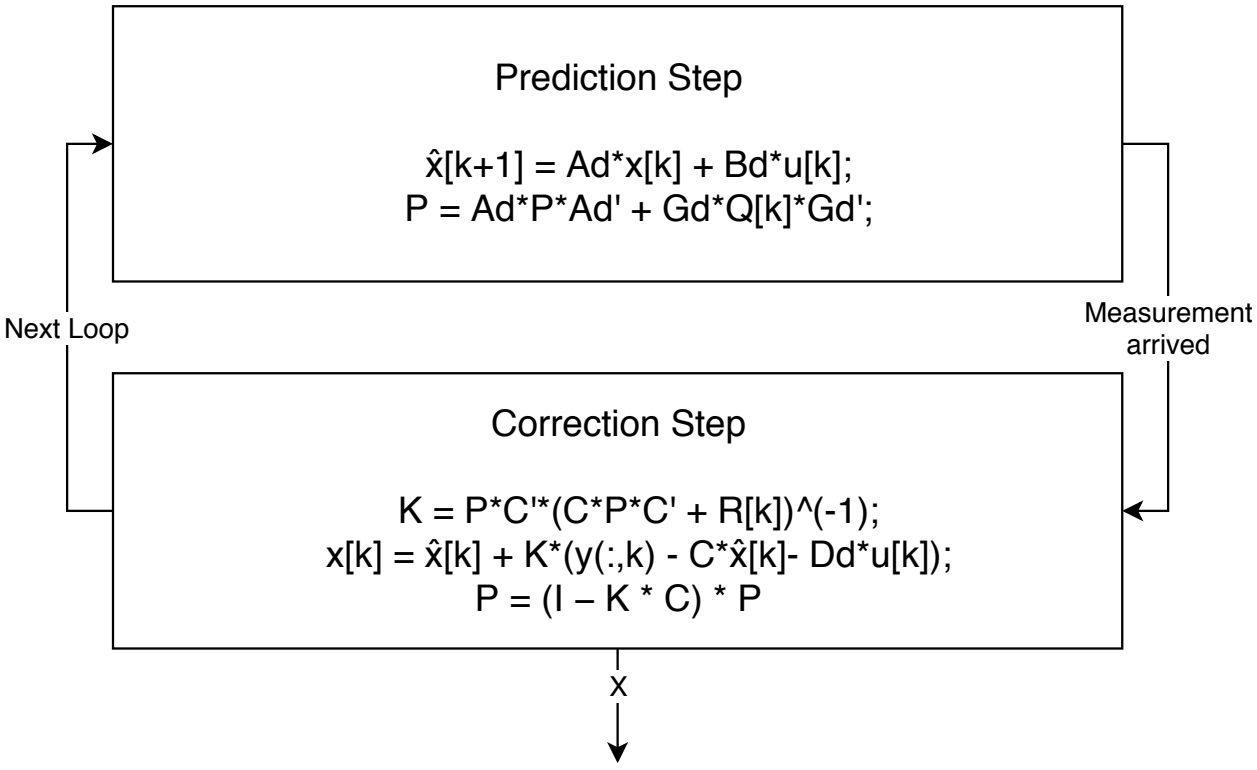

Figure 2.4: Kalmanfilter

Prediction Step

The prediction equations take the current values of the state vector ($x[k]$) and uses the time depending system model part (Ad) to predict the state values for the next time step $\hat{x}[k+1]$ with the equation:

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

The hat denotes that this value is an assumption.

In addition the certainty matrix (P) is estimated with the same tactic, 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 in the correction step to correct the prediction. First the Kalman gain (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 noise on the measurements, meaning how accurate the values from the measurements are. K is then used in the following equation to calculate $x[k]$:

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

By this, the measurement corrects the predicted value of the state vector with there uncertainties taken into account [4].

In the last equation

$$P = (I - K * C) * P$$

the certainty matrix is corrected with the help of the Kalman gain and the estimated certainty matrices.

To be able to run this Kalman filter the matrices for the system models (Ad,Bd,C,D), the measurements noise (Q,Gd) as well as the system noises (R) have to be defined.

Chapter 3

Implementation

This chapter describes how the simulation is implemented in detail. The development of the simulation has been done in Matlab. The first part of the chapter mainly discusses the sensor model implementations while the second part is about the implementation of the state estimator itself.

3.1 Sensor Models

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

3.1.1 Perfect Sensor

To calculate the prefect sensor data the trajectory is put into the equations which were defined before. In Figure 3.1 those generated sensor data as well as the trajectory used for this can be seen.

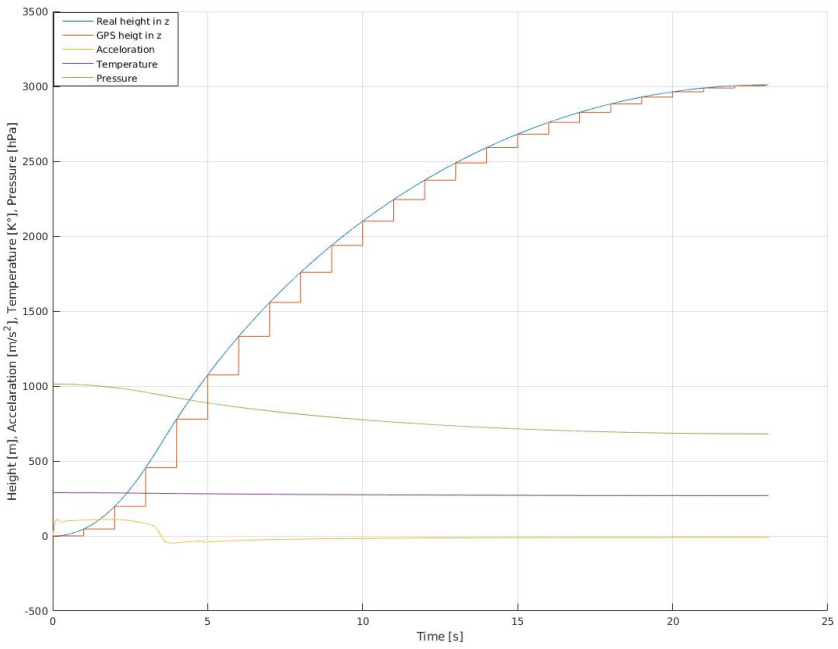

Figure 3.1: Generated sensor data

Accelorometer

Due to the fact that the whole simulation works with discrete time stamps the derivative of the height can not be done formally. So it is done by calculating the difference between each data point to the next and then weighting 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 meters per second squared.

Gyrometer

As stated before the pitch angle can not be directly generated. But as can be seen from the data of the test flights 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 determined direction and stabilises 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 simulate this the values are generated randomly and then 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 raised afterwards to higher values.

Barometer

The measurements from the barometers are depending on the atmospheric model which is used in this simulation. For reasons of simplicity the start pressure is chosen as the mean pressure at sea level which is 1013.35 hP. Also the temperature at the beginning is chosen as 288.15 Kelvin (15 °C) which also represents the mean value on the sea level. Last but not least the temperature gradient is chosen as - 0.0065 K/m, which is a commonly used value. For the state estimation in a test flight those values have to be determined before the start, especially the temperature gradient.

Since the barometers do sample at 100 Hz the measurements have to be down sampled. As with the GPS sensor below this is achieved by a zero order hold conversion instead of direct down sampling

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, it has first to be extracted. It is assumed that the noise of the data is different depending on the state of the rocket (before ignition, during motor burning, after burnout untill parachute ejection), but it should have more or less the same properties between those events. Based on this, the data vector has to be separated in those different sections first. For this the accelerometer measurements are iterated to find the time stamps on which those events happen, as can be seen in Figure 3.2.

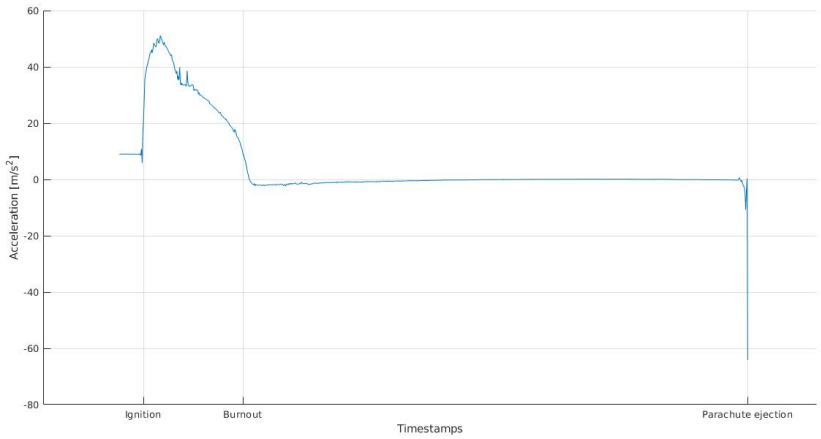

Figure 3.2: Timestamps drawn out of acceleration measurements

If done so, polynomials of the order two are fitted on this measurements with the least squared error method. Those polynominals represent the functions / the values which are assumed to be the noiseless data with possible offsets. So if now these polynominals are subtracted from the test flight data the result are the measurements with zero mean. From this point on this noise can be examined on its parameters, like the power density, the probability distribution and the variance, as can be seen in Figure 3.3.

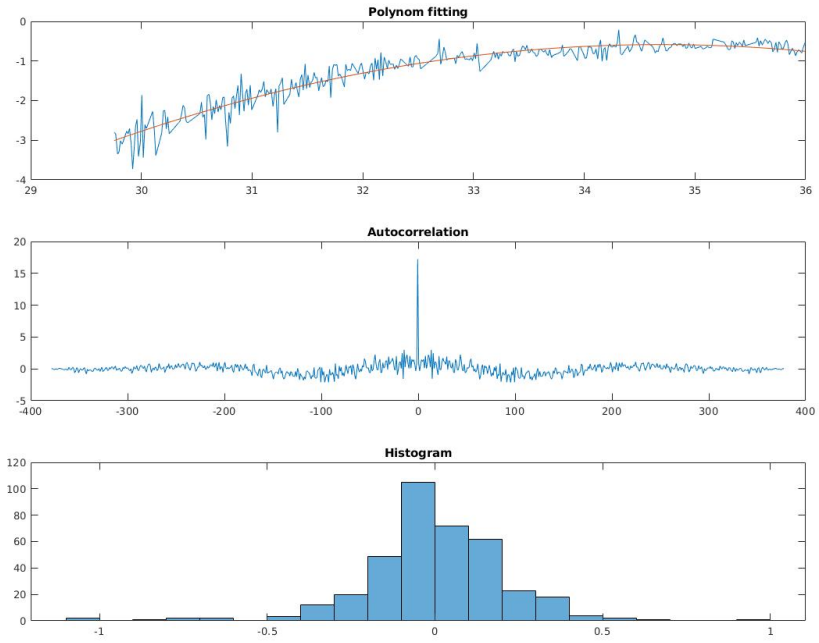

Figure 3.3: Polyfit Autoccorellation and Histogramm

This noise data can now be used to solve the Yule Walker Equation to get an AR model. For this the aryule() function can be used which estimates an AR model of the order N as well as the variance directly out of the noise vector. To achieve this it estimates the autocorrelation out of the given noise vector. For the best possible results the corresponding data from all test flights were put together and used as one great noise vector. Before doing this, the data had to be resampled so that the AR models can be properly used 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 from the test flight data.

As can be seen in figure 3.4 both noises 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 is helpful, because the value that should be measured is known.

Gyrometer

For the gyrometer noise a separate script has been 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 degrees per second and have therefore to be integrated before they resemble the correct pitch angle. In addition to this 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 regenerate noise with the same characteristics. Because of that, the noise has to be low pass filtered another time to resemble the real noises better. For this task an IIR filter of order 200 was found best. With this also the Gyrometer measurements could be properly handled.

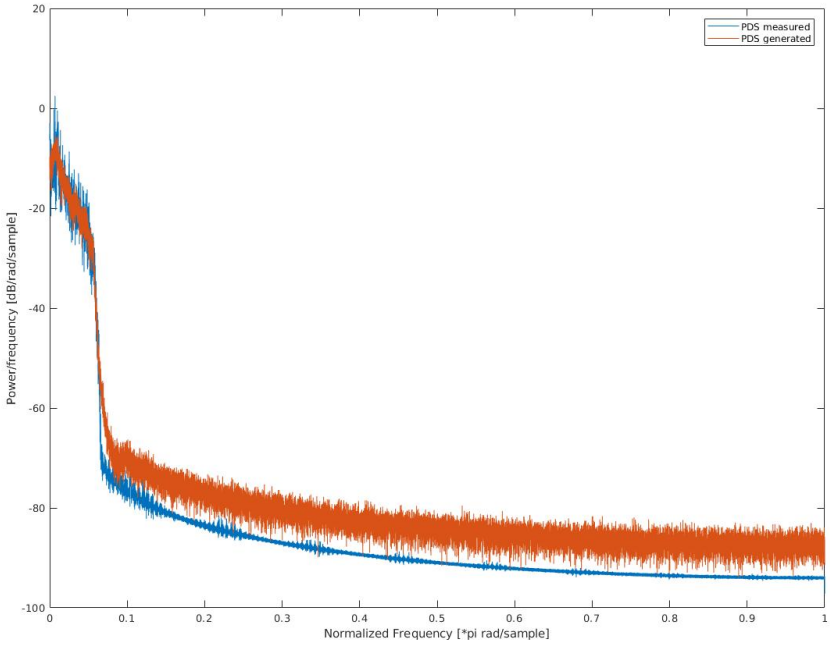

Figure 3.4: PDS from the measured and generated noise

Barometer

The barometer noise itself has no real capacities which were not already discussed before.

GPS

The GPS noise capacities were only found with measurements that were taken for a longer time period while the GPS receiver stayed at the same place. So based on the assumption that the GPS measurements are independent from the motor vibrations and the rocket posture changes the noise will have the same capacity over the whole flight. This should be suitable as long as the receiver does not lose its fix.

3.1.3 Real Sensor

To now generate the real sensor data, the different noises have to be generated with the calculated AR models and added to the perfect sensor data. For this a vector of normal distributed random values is generated and multiplied by the square root of the corresponding variance (hence the standard deviation). This white noise is now filtered by the corresponding AR-model and can then be added onto the corresponding perfect sensor data, which results in the real sensor data.

Accelorometer

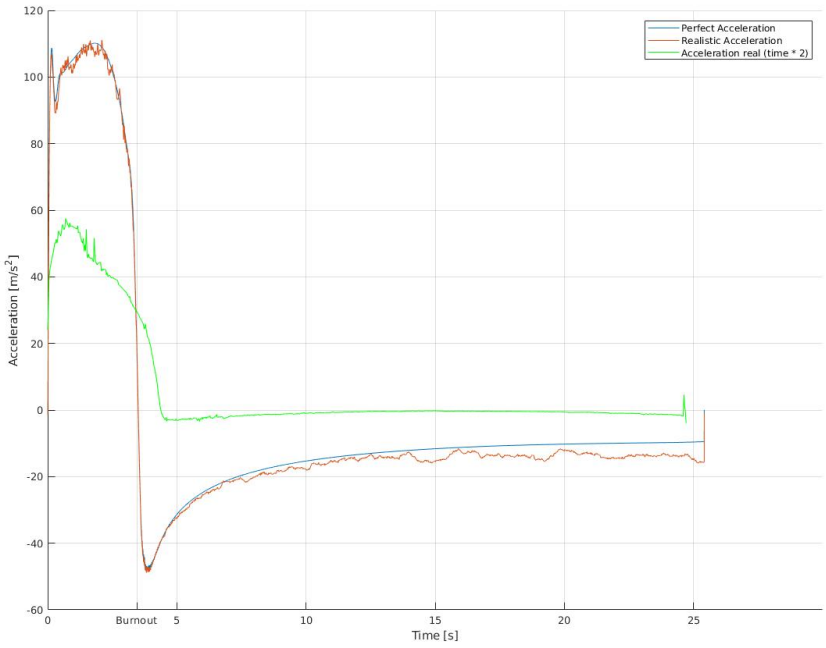

Figure 3.5: Plot of perfect acceleration vs realistic vs measured

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 not 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 generated 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 that due to the property of the pitch angle (more or less random depending on air current etc) the generated realistic pitch angle must not resemble the measured angle in its specific value. Important to show is that the noises have the same capacities which they do. Also it can be seen that the assumption that the angle does not change much during the burning of the motor is appropriate despite a quick change at the start.

Barometer

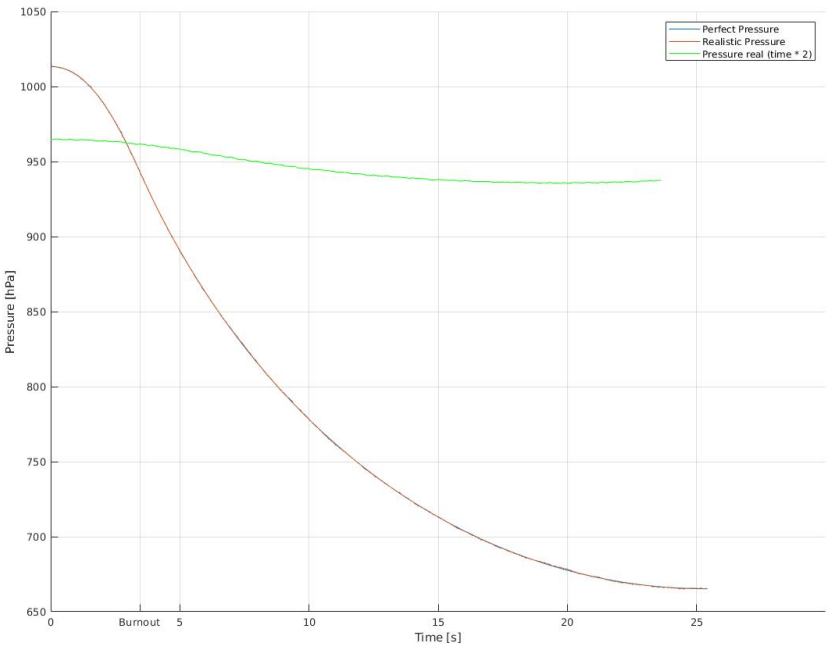

Figure 3.7: Plot of perfect barometer vs realistic vs measured

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

GPS

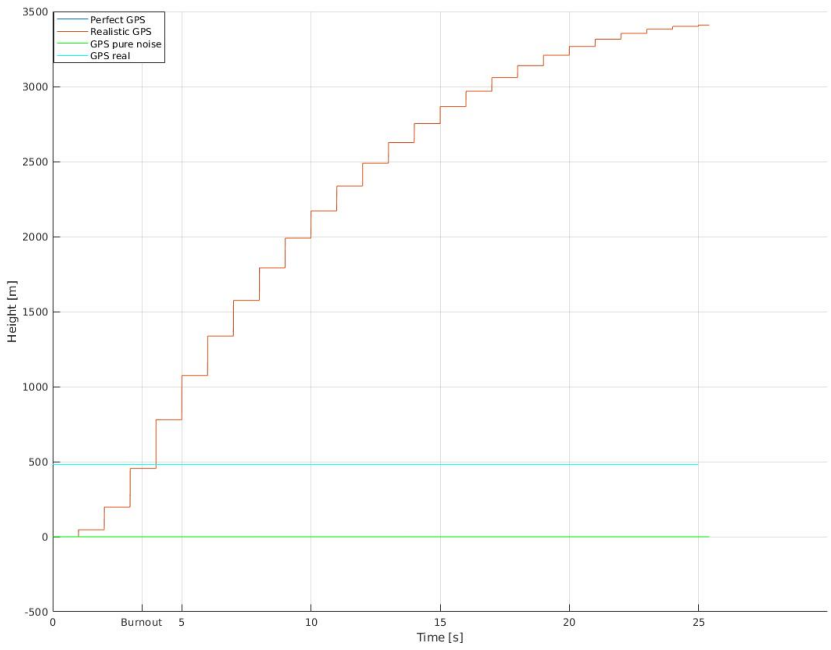

Figure 3.8: Plot of perfect GPS vs realistic vs measured

Last but not least the generated GPS measurements can be observed in Figure 3.8. In this plot the pure generated noise was also plotted because this way it can be best shown that it has real slow properties. For 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 in the last chapter the to be used state estimator is a discrete Kalman filter with dynamic noises on the measurements as well as on the system. For this the noise matrices as well as the used loop is described below.

3.2.1 System Model

Based on the models stated in chapter 2 several different implementations can be derived. For this simulation 8 models were implemented with each different capabilities. To find the best suiting model all of them have been tested. The test results are discussed and evaluated in the next chapter.

3.2.2 Adjustment

Taking into account the noise characteristics of the measurements stated above, some adaptions to the model presented in chapter 2 have been made, 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 to minimise the impact of the offset. This works because by adding the offset to the state vector the state estimator can also estimate the actual offset which can then be used to correct the offset value [4]. 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 from 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 and acceleration offset 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 Measurement Noise

The matrix on each time stamp for the measurement noise 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 separately. After that those values are used to generate a noise vector for each measurement. In addition the implementation of the noise vector has the same length as all other used vectors. These noise measurements are then concatenated into diagonal noise matrices as shown in the listing below:

```
R_dyn = [GPSvar;ACLvar;HBM1var;HBM2var];          %Add the noise vectors into one matrix
R_dyn_m = diag(R_dyn(:,1)');                      %Make a diagonal first diagonal matrix
for n = 2:length(TimeVec)
    R_dyn_m = cat(3,R_dyn_m,diag(R_dyn(:,n)'));   %Concatogenate all noise vectors
end
```

Listing 3.1: Mesurements noise matrix conagonating

If the noise matrix is displayed as a diagonal matrix, it takes into account the assumption that the noises from the measurements are independent from each other. This assumption can be made because of the fact that each measurement except those from the barometer (pressure and temperature) are made from different sensors. 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 Times

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. This adjustment 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,

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

maximum values in the R matrix result in nearly zero values in the corresponding K matrix. Those near or exactly zero values result in ignoring the corresponding measurements from the y vector as can be seen in the measurement update equation.

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

With this vectors with the same length as the other vectors in the state estimation can be generated out of the measurements. When implementing in an embedded system this can simply be achieved with an "if" statement that switches the R matrix value to the normal variance if a measurement arrives.

3.2.4 System Noise

The system noise describes how uncertain the system model is in comparison to the real system. For this each entry in the diagonal matrix 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 angle, pressure if linearised). For example Figure 3.9 shows the pitch angle. In the system models there are no influences on this state

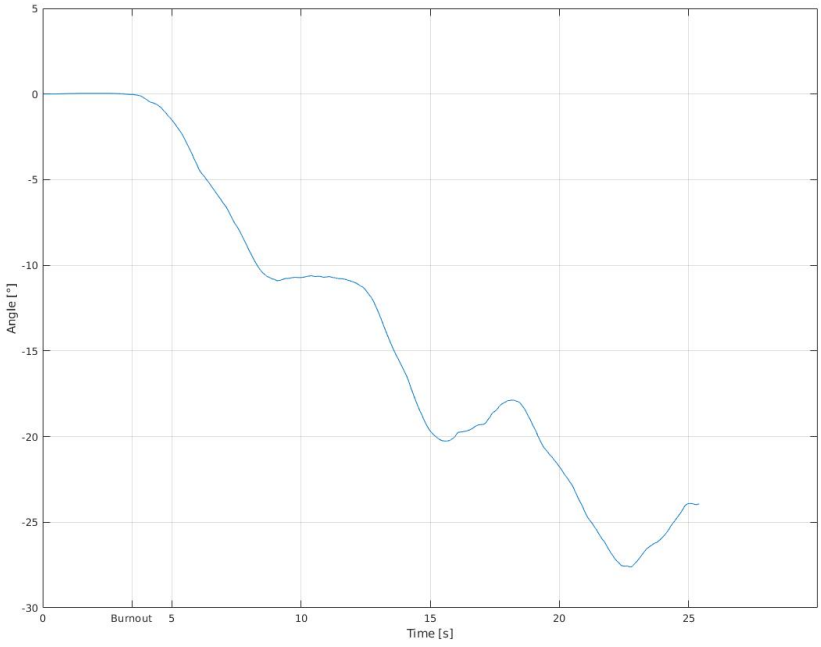

Figure 3.9: Plot of pitch angle

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 all over the 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 by 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 3.2: System noise generation with deviation

This shows that after the deviation the absolute value from the deviation result is taken since a variance cannot or should not be negative. After that the values are low pass filtered to make a 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 if 2-3 sensor fail, accepting that the estimation results will not be as accurate any more. 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 for the GPS signal 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 most system model versions the u vector remains zero while all measurements are brought into the estimation loop trough the y vector. But there are some others where the acceleration and the pitch angle are brought into the estimation loop over the u vector only and the remaining measurements through the y vector. That means that the current state vector x has to be initialised with the value of the corresponding sensors at the start, which is presumably zero for all states except pressure and temperature. The loop itself calculates the equations 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 only with a linearising 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. This model 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);
x_est_loop = zeros(size(x,1),length(TimeVec));         %Vector for the state estimation values

% Estimation loop
for k = 1:length(TimeVec)
    K = P*C'*pinv(C*P*C' + R_dyn_m(:,:,k));
    % Calculate height out of pressure measurements.
    Height1 = CalcHeight(Po,p_mes_1(k),Temp,0,true,TgradSimu);
    Height2 = CalcHeight(Po,p_mes_2(k),Temp,0,true,TgradSimu);
    % Calculate the pure vertical acceleration with the pitch angle.
    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 3.3: State Estimation Loop

Chapter 4

Tests

In this chapter the test results of the tests that were taken to gain insight of the different system models and their performances are discussed. First a general analysis is made by plotting the whole estimated flight as well a table which displays the errors of the different estimated values which were taken from multiple simulation cycles. For this all estimations have been made with a mean acceleration offset of $-1.31m/s^2$ which was taken out of measurements from a test flight. Also for the general analysis a perfectly linear temperature gradient of $-0.0065°/m$ was used. After that the different problematic properties of each system model is discussed in detail. At the end the best system from the first tests has been further tested under different circumstances.

4.1 Point Mass

The first system model to test was the simple point mass model as described in chapter 2. While this implementation is a simple version it does already work surprisingly good. In Figure 4.1 the overall performance of its estimation of the different values can be seen.

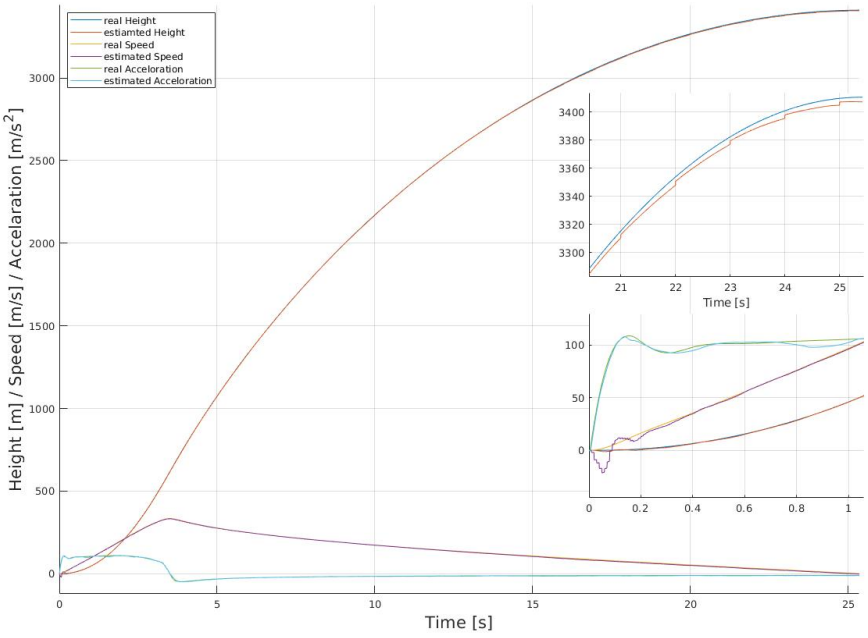

Figure 4.1: The performance of the point mass system model over time

Especially the first second (lower right corner) and the last five seconds (upper right corner) are interesting. It can be seen that the system does not really have a settling time at all, because the height is estimated at the minimal error at the beginning of the estimation. On the other hand the speed needs around 0.2

seconds to settle on the right value. Also a clear error on the estimation of the error can be seen after the speed has settled, this is due to the fact that the estimator can trust the GPS and pressure measurements much more at the beginning than the acceleration measurements. Therefore the estimator does adapt the acceleration estimation on the height and speed estimation at the start and not the other way around. In the last five second it can be seen that the height is far more off as it was at the start. Also the steps the estimation makes there are coming from the GPS measurements. In addition Table 4.1 shows the maximum, minimum, mean and median error of the estimated values.

State Variable	Unit	Max	Min	Mean	Median
Height	m	20.05	1.18e-05	4.72	2.20
Speed	m/s	3.42e+03	0	3.40	2.43
Acceleration	m/s^2	17.79	2.55e-05	1.67	1.55

Table 4.1: Error of estimated state variables

The most interesting value is the median from the height error because it is free from outliers. It shows that the height error is with 2.2 meter slightly above the aimed 2 meter. The 20 meter maximum error occurs around second 15 of the flight just before a new GPS measurement comes in. At this time stamp the offset on the acceleration measurements does have a greater impact because the real value is in comparison to the start rather small. In addition the height does change more between two GPS measurements than it does at the end of the flight, therefore greater errors in the interpolation between can occur. Additionaly the minimum speed error occurs at the beginning of the estimation loop where the speed still zero.

4.1.1 Greater Offset

It has to be said that this only works while no sensor has a bigger 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 estimations. 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 4.2 which shows the state estimation error with different offsets.

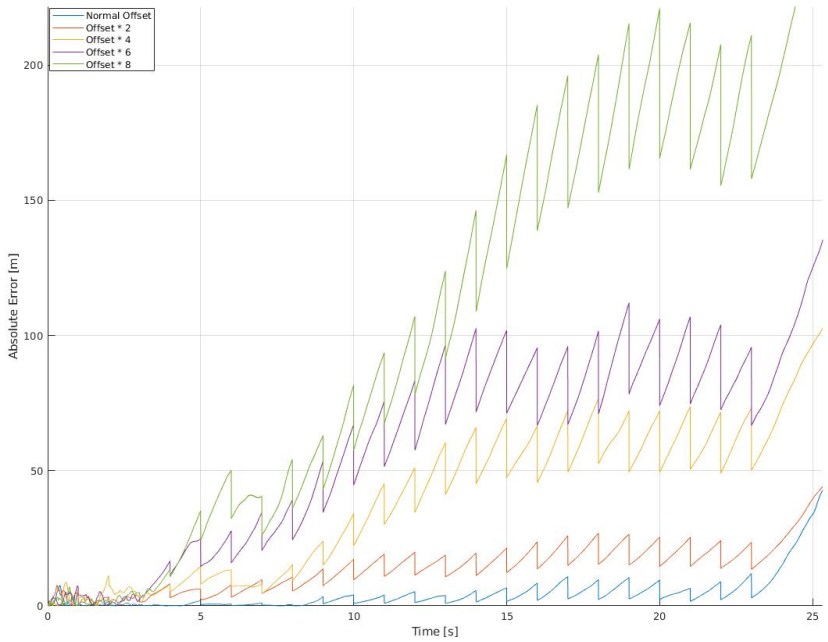

Figure 4.2: Error during flight time with different offsets

Table 4.2 shows the mean and median of the error in the height depending on the offset on the accelerometer.

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

Table 4.2: Error of the height in meter with changing offset

This shows that the error which the estimator makes does rise exponentially. So this is an issue which should be assessed by including an acceleration offset in the state vector.

4.2 Point Mass with Acceleration Offset

The overall performance of this system model can be seen in the figure 4.3.

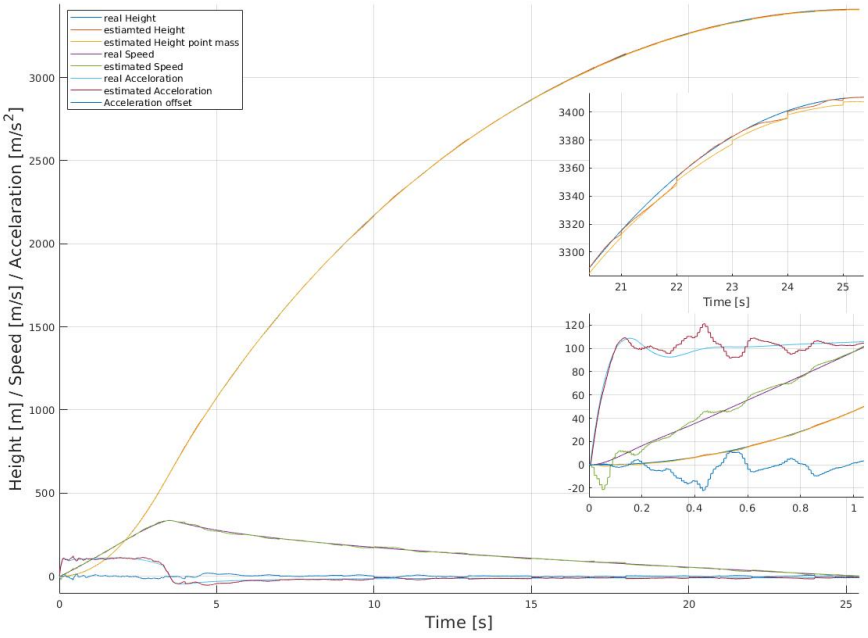

Figure 4.3: Performance of point mass with acceleration offset over time

While the speed and height of it performs more or less the same way at the first second like the point mass system model, a clear difference in the estimated acceleration can be seen. This is possible due to the fact that the estimator can describe the difference in the system description between the speed and acceleration as the acceleration offset. These dependencies can be seen after the settling of the speed where the estimator starts to change the acceleration offset value to adjust the acceleration. The great advantage of this can be seen in the plot of the last five seconds of the height estimation (upper right plot). Due to the better estimation of the acceleration over the whole time, the dependencies of the height on the acceleration is much more trustworthy. When the rocket does rise at further height and the measurements of the barometers lose their accuracy, the better acceleration measurements can be used to interpolate between the good GPS measurements. This results in a much better height estimation at greater height.

Table 4.3 shows the estimation errors of this model. The entries that draw the attention are the maximum of the estimation errors from the acceleration and speed which is bigger than that of of the point mass system model. These occurred due to one complete false estimation during one of the simulation and should not be seen as normal. The interesting thing about this is that despite those big maxima, the mean value of the estimated height and speed are still better than the ones from the point mass system model. Also can be seen that the median of the height error is just 0.83 meter.

State Variable	Unit	Max	Min	Mean	Median
Height	m	9.78	2.98e-06	1.33	0.83
Speed	m/s	78.79	0	3.18	2.22
Acceleration	m/s^2	165.87	1.35e-05	4.52	2.64
Acceleration Offset	m/s^2	167.46	2.66e-05	4.44	2.61

Table 4.3: Error of estimated state variables point mass with acceleration offset

As with the point mass system model Figure 4.4 shows the error during a flight with different sensor offsets. It can clearly be seen that while the mean error does rise by some value, it always gets back to zero and therefore overall this system preforms better as the simple point mass.

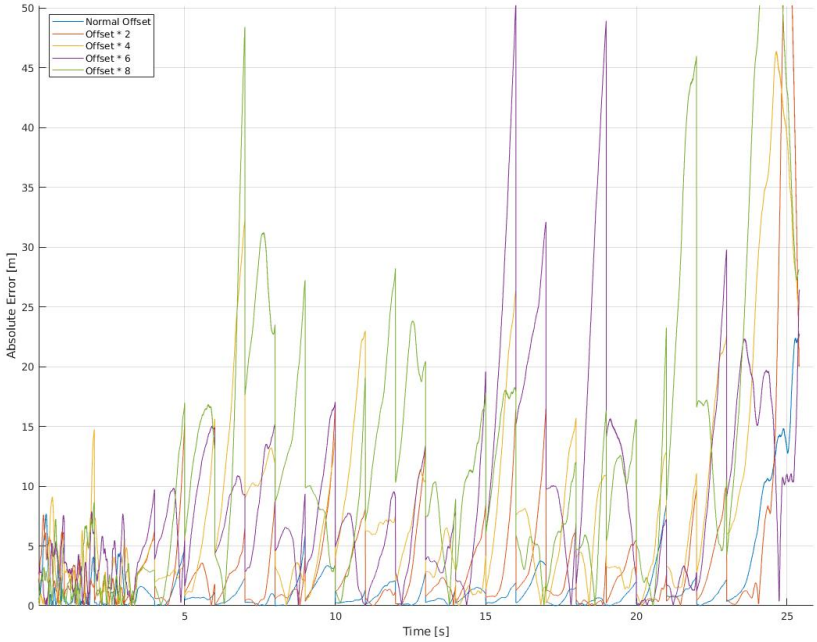

Figure 4.4: Error during flight time with different offsets

4.3 Point Mass with Pressure

As with the ones above the overall performance of the system model which uses the pressure as an additional state variable can be seen in Figure 4.5.

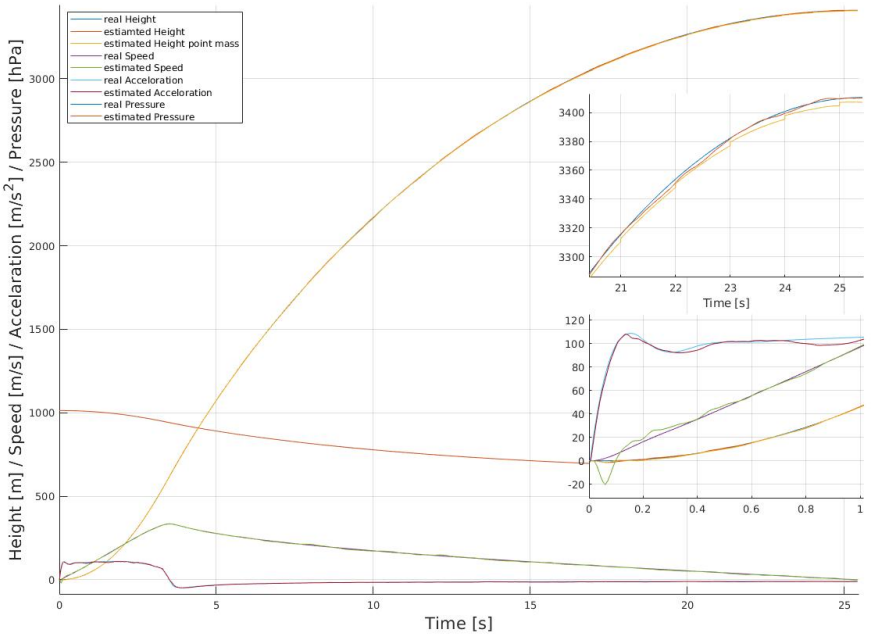

Figure 4.5: Performance of point mass with pressure over time

This shows that it performs as good as a system which uses acceleration offset as a state value. It can be seen that the pressure is so well estimated that it fully covers the real pressure curve in the plot. Also in the plot of the first second it can be seen that there is no settling time for either of the estimation values. The height in the last 5 second shows too that the impact from the GPS measurements are much smaller than in the first model (no staircase like adjustments of the height as seen in the point mass system model).

State Variable	Unit	Max	Min	Mean	Median
Height	m	5.16	1.47e-06	1.20	0.96
Speed	m/s	8.80	0	1.89	1.61
Acceleration	m/s^2	18.14	3.90e-05	1.80	1.63
Pressure	hPa	0.72	4.69e-06	0.13	0.11

Table 4.4: Error of estimated state variables point mass with pressure

Like above Table 4.4 shows the error over the simulations. The error on the height is nearly as good as with the acceleration offset, while the maxima of the differences are quite smaller than above. These results therefore have much better mean values of the errors.

4.3.1 Wrong Temperature Gradient

While it does increase the accuracy it does also increase the needed computational effort due to the fact that to get a real added value from this, the height out of the pressure has to be calculated at each time step with the help of the estimated pressure. In addition when the temperature gradient is chosen wrong it deeply affects the estimation as can be seen in Figure 4.6. The errors there were low pass filtered with a moving average filter for better visualisation.

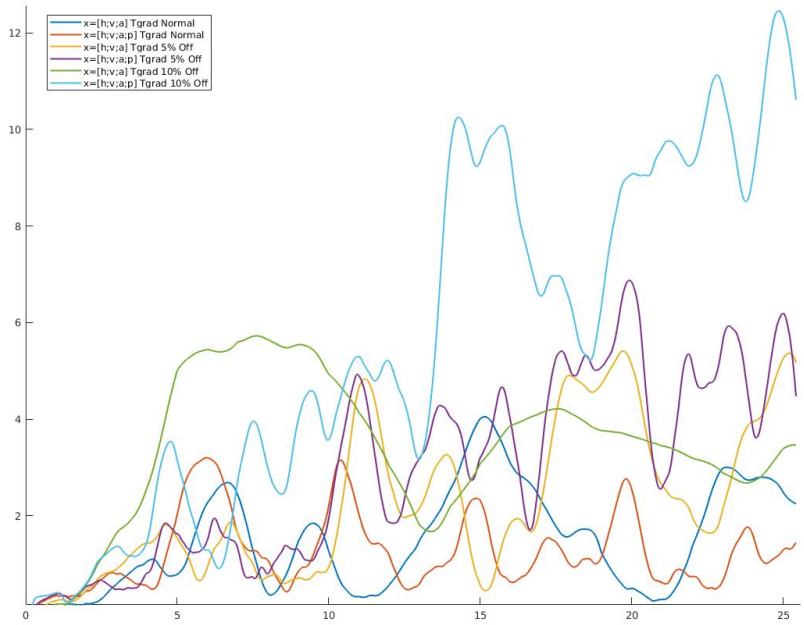

Figure 4.6: Plot of point mass vs the pressure low pass filtered

In this figure the error for both, a system with acceleration offset (which therefore can depend more on the accelerometer measurements) and a system model with the pressure in the state vector are compared against each other. While they preform equaly good as long as the temperature gradient is determined correctly, if the gradient is determined wrong by just 5 percent it already performs less accurate than all of the other tested system models. This is a problem due to the fact that the temperature gradient will most certainly not be correct during the whole flight. With this it can be seen that the overall system performance of this model is more or less the same as with the normal point mass system. On the other hand if the error on the temperature gradient is exactly known, the noise onto those measurements can be adjusted which results in a gain of robustness for the whole estimation.

To summarize it, the pressure as a state variable is a risky but if done right a helpful adjustment of the system model.

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 on those measurements. This can be seen in Figure 4.7. The estimated pitch angle there changes slower than the generated measurements value which can be seen in chapter 3.

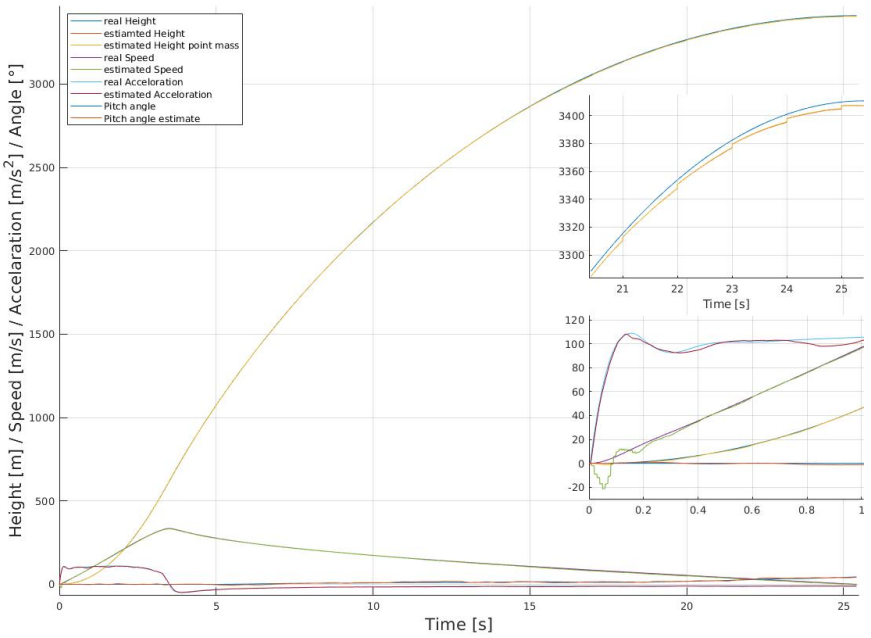

Figure 4.7: Performance of point mass with pitch angle over time

The overall performance is more or less the same than that of the normal point mass system. This can espacially be seen in the last five seconds of the height estimation where both estimations are equal to each other (the estimation of the point mass system model does overlap the one of the system model which includes the pitch angle estimation). The one big difference which can be seen in the plot of the first second is that the estimator tries to correct the wrong acceleration measurements (due to the offset) with the change of the pitch angle. But because the system noise on the pitch angle states that it does not change in a great manner until the burnout the influence is restricted.

State Variable	Unit	Max	Min	Mean	Median
Height	m	20.24	4.46e-06	4.75	2.24
Speed	m/s	78.57	0	3.42	2.44
Acceleration	m/s^2	17.79	6.26e-05	1.68	1.55
Angle	°	12.26	2.46e-05	2.41	1.90

Table 4.5: Error of estimated state variables point mass with pressure

In Table 4.5 the errors of the estimation can be seen once again. It shows that the values are nearly the same as the ones from the point mass system.

4.4.1 Small Influence

This results in the conclusion that the influence of the pitch angle is much smaller than thought. This mainly because the pitch angle does only start to get to a greater value after the burnout. After this the vertical acceleration is much smaller and therefore an error of the angle by a few degrees does not have a big impact. In other words for example if the real angle is 10 degrees, the measurement at this point is around 18 degrees while the estimation is 11 degrees. Then this means the measured acceleration of for example $10.15m/s^2$ should be corrected to $10m/s^2$. With the estimated angle it is corrected to $9.97m/s^2$, while with the measured angle it is corrected to $9.66m/s^2$. So for this example an error of 8 degrees would only result in a false estimation of $0.31m/s^2$. Although this error does not develop in a linear fashion especially when the angle has a greater value, due to the rather small accelerations after the burnout the impact of the noisy measurements is still strongly limited. This can be seen in the plot on Figure 4.8 where the estimation which uses the pitch angle as an additional state variable makes the performance just slightly better.

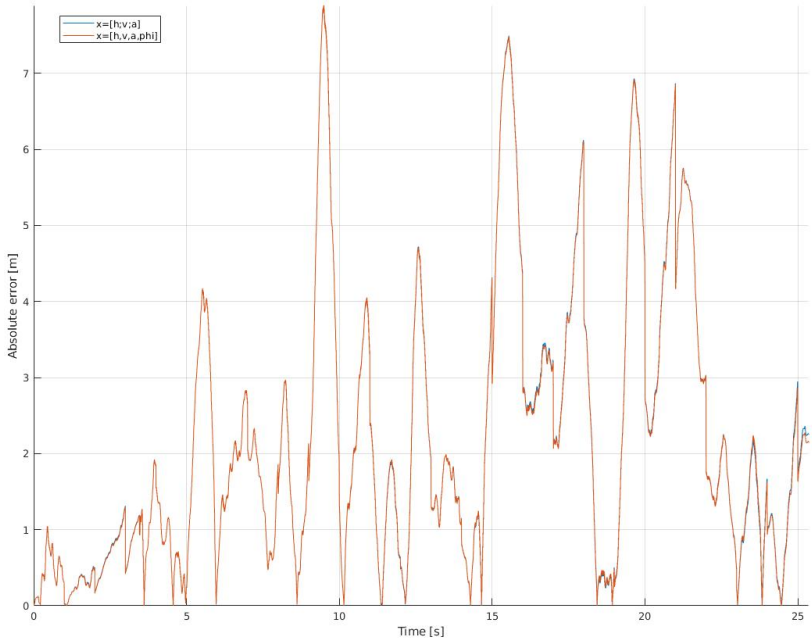

Figure 4.8: Estimation error over time from point mass system with and without pitch angle inclusion

4.5 Point Mass with Acceleration as Input

The overall performance of this system model is shown in Figure 4.9.

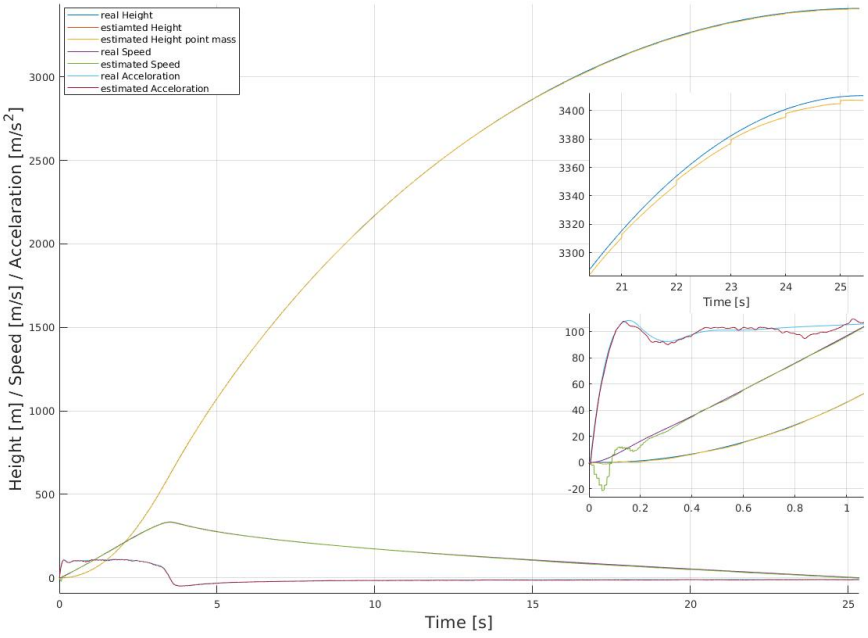

Figure 4.9: Performance of point mass with acceleration as input over time

Its performance does also resemble the point mass system model quite well but without any significant improvement. As above with the pitch angle the estimated height of both systems do overlap over more or less the whole flight. This can be seen in the upper right corner of Figure 4.9 where due to the overlap only the values from the point mass system model is visible. The difference in the errors between this system model and the normal point mass model occurs more due to rounding errors of the simulation than due to really better estimation. This can be concluded by fact that this system model performs sometimes slightly better and sometimes slightly worse than the point mass system, therefore there is no real gain in the implementation this way.

In addition the plot of the first second of the estimation shows that the acceleration estimation has more noise on it than in the other system models. This is because the acceleration has no model dependencies in this implementation, it is taken directly as system input. With this the system noise on the acceleration has to resemble either the measurement or the system noise. In this implementation it was used to bring the measurement noise into the system and therefore the low pass characteristics of the system noise is lost. In other words with a measurement as input a tuning parameter (either system or measurement noise) which could compensate the corresponding state value cannot be used. So there is no real gain in this implementation and it will therefore not be implemented in the final system model.

State Variable	Unit	Max	Min	Mean	Median
Height	m	20.05	7.21e-06	4.71	2.18
Speed	m/s	78.49	0	3.38	2.40
Acceleration	m/s^2	10.65	0	1.60	1.43

Table 4.6: Error of estimated state variables point mass with acceleration as input

4.6 Point Mass with Offset and Better Calculated System Noise

The better calculated system noise for this estimation was calculated as stated in chapter 3. This has been done by calculating the discrete system noise matrix Gd with the integration method as well as derive the perfect measurements and then low pass filter them to get better system noise vectors. This improvement had to be done on a system which uses the acceleration offset as well in the state vector to get the maximum possible gain out of this implementation. This results in an overall system performance as seen in Figure 4.10

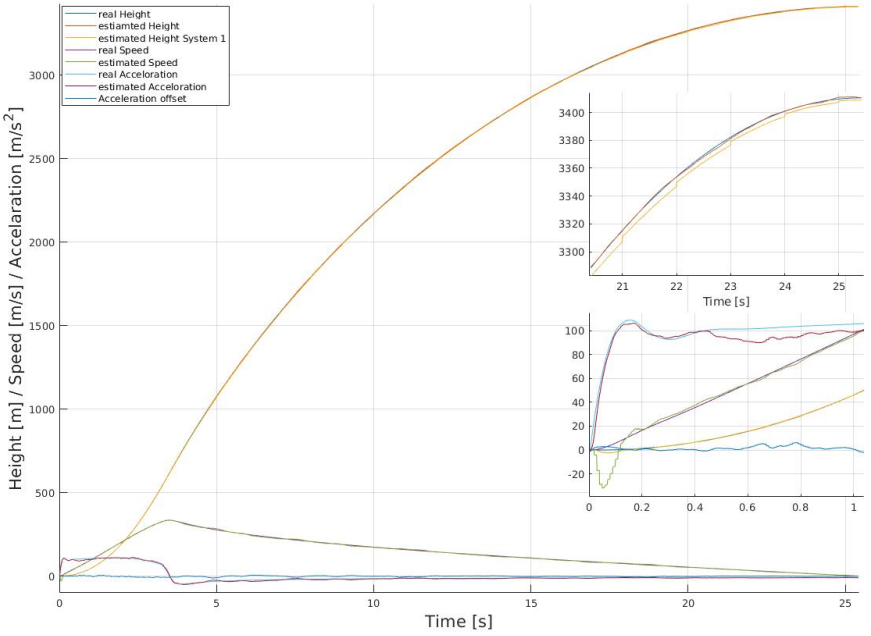

Figure 4.10: Performance of point mass with better system noise over time

This system performs much like the one with acceleration offset as a state vector. The greatest difference can be seen in the first half second where the acceleration is nearly perfectly estimated. Also does the acceleration offset not change as much as in the system model without the improved system noise calculation. This is because with the better system noise an optimal estimation can be achieved with less effort since the influence of the acceleration offset is more clearly defined.

State Variable	Unit	Max	Min	Mean	Median
Height	m	9.74	8.11e-06	1.28	0.73
Speed	m/s	78.58	0	2.79	1.85
Acceleration	m/s^2	86.93	4.07e-05	3.57	1.96
Acceleration Offset	m/s^2	88.60	2.52e-05	4.59	2.59

Table 4.7: Error of estimated state variables point mass with acceleration offset and better system noise

Table 4.7 shows that the system performs pure error wise better than each of the other system models discussed before. While it also contains big maxima in speed, the acceleration and acceleration offset values are smaller than the ones of the system with only acceleration offset. Due to that the mean values (except the offset) are still better than most other.

This shows that an overall better estimation can be achieved with this tactic. This is mostly due to the better system noise vector which can be much better estimated this way. Also the additional effort to access this system model only occurs on the preparation, while it does have no effect on the computational effort during the flight itself. So this approach of the state estimation should be used any time if available.

4.7 With and Without GPS

It is not certain that the GPS will be working in the next competitions as already stated in the introduction. In addition the GPS sensor data needs more time to be measured and can therefore arrive too late to be included correctly. There is the possibility of back calculation to include such too late arrived measurements into the state estimation but this needs a lot of computational effort [6].

Because of this, the estimation without the GPS measurements are tested with a point mass system model to find its direct impacts. Figure 4.12 shows the plot of different estimations with and without working GPS.

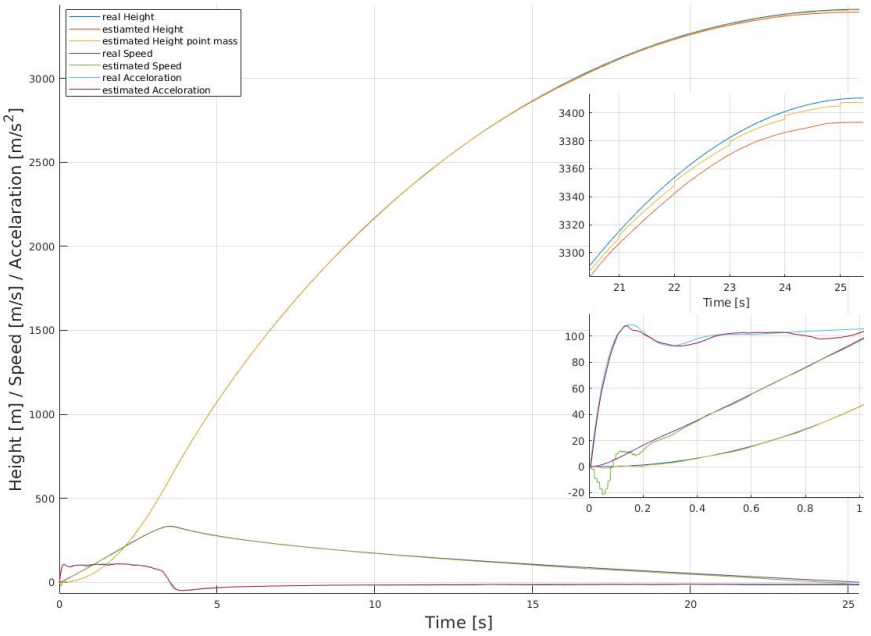

Figure 4.11: Performance of point mass without GPS over time

As expected the height is further away from the ground truth in the last five seconds plot than the one estimated with GPS measurements. Also the staircase like correction steps which come from the GPS measurements are missing. This behaviour was expected because the the GPS measurements are most important at great heights were the remaining measurements lose their credibility.

State Variable	Unit	Max	Min	Mean	Median
Height	m	31.20	5.77e-05	8.97	5.20
Speed	m/s	78.58	0	6.35	4.86
Acceleration	m/s^2	17.78	7.60e-05	3.48	2.34

Table 4.8: Error of estimated state variables point mass without GPS measurements

Table 4.8 which displays the errors from the estimations shows that as expected the error is bigger in each value. Also the maximum height error of over 30 meter shows that the normal state estimator is really depending on the GPS measurements.

4.7.1 Wrong Temperature Gradient

This system has to depend strongly on the barometer measurements to calculate the height. Therefore the problem of a wrong temperature gradient arises once again. To show this the estimation results with different temperature gradients are plotted in Figure 4.12.

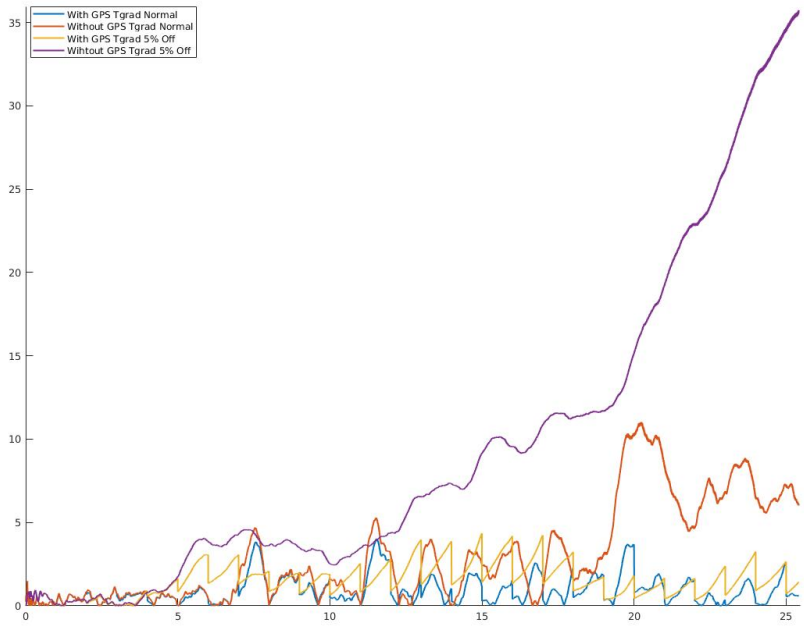

Figure 4.12: Absloute error over time with and without GPS

It shows that especially if the temperature gradient is chosen wrong (by just 5 percent) and therefore the height out of the pressure measurements is calculated wrong the error of the estimation rises significantly. With this the error increases with the ascending rocket if it is not corrected by the GPS measurements. So the aim of the best performance system has to be that it should not depend too much on those GPS measurements for a good state estimation.

4.8 Best Performing System

Based on the results from the tests above the best performing system can be defined. The main goal is to achieve an as robust state estimation as possible that does also match the given accuracy requirements. The pitch angle did not have a useful positive effect on the estimation and would increase the computational effort due an additional rank, because of this it is not implemented. Also the acceleration as input does not significantly improve the estimation. Therefore the best system for the stated problem consist of the additional acceleration offset as a state vector as well as better calculated system noise. Despite the fact that the pressure is risky to implement as a state variable it is used in this implementation because it does increase the robustness significantly. This does therefore result in a rank 5 system which should fit the given calculation steps requirement at first glance, but it will also be analyzed in detail in the next chapter.

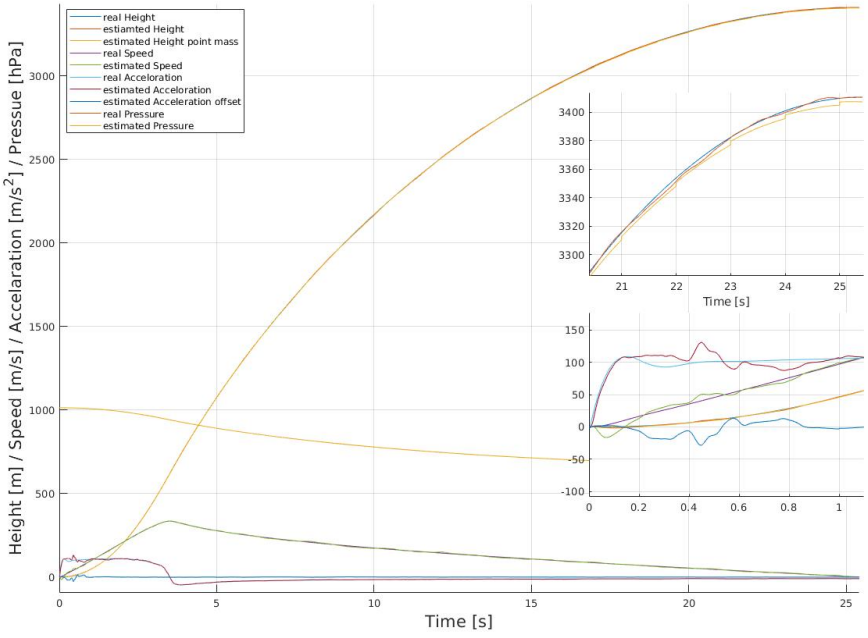

Figure 4.13: Performance over time

Figure 4.13 shows again the performance over one whole flight. In the plot in the lower right corner which shows the first second of the estimation it can be seen that this system has a clearly visible settling time in the acceleration and speed. While this seems quite big at the first sight, it is well in the given requirements of one second settling time after the burnout. On the other hand it shows that the performance increases significantly after the acceleration and speed have settled. This increase in the accuracy of the estimation can also be seen in the table 4.9.

State Variable	Unit	Max	Min	Mean	Median
Height	m	7.39	2.01e-06	1.26	0.92
Speed	m/s	24.87	0	1.61	1.26
Acceleration	m/s^2	41.66	4.86-06	1.28	0.70
Acceleration Offset	m/s^2	41.70	5.36e-06	1.01	0.61
Pressure	hPa	0.80	2.49e-06	0.14	0.12

Table 4.9: Error of estimated state variables of the best found system

While the median of the error from the estimated height shows a slight loose in the height accuracy in comparison to the system model with better system noise alone, the accuracy of all other values has increased. This is especially impressive in the mean of the speed and acceleration, because they have big errors at the beginning with the shown settling problem which has been shown in the plot above. Also the height is with a median of just 0.82 meter and a maximum of 7.39 meter still most of the time in the aimed error of 2 meter maximal error.

4.8.1 Robustness

The slight loss in accuracy was traded for an increase in robustness which is seen as more important due to different uncertainties, which are still being able to function properly if a sensor fails as well as being robust against wrong system modelling and especially falsely detected temperature gradients. The impact of those possible errors on the found best performing system will be shown in the following sections.

Without GPS

As stated above this system should estimate the height without a significant raise in the error without the GPS measurements. Figure 4.14 shows the height error of the best performing system model without GPS measurements. For comparison there is also the point mass system model estimation with and without the GPS measurements plotted.

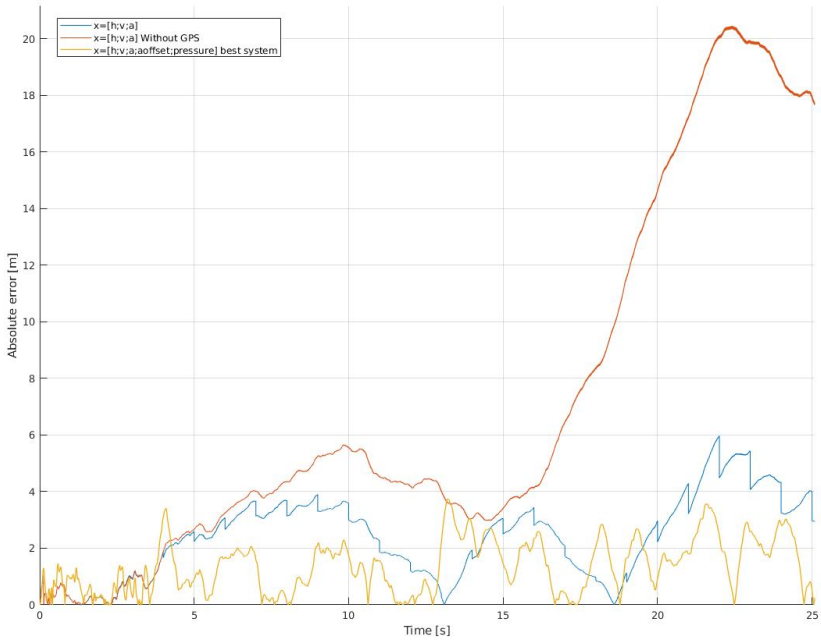

Figure 4.14: Error of point mass system and best system with and without GPS measuements

This shows that the best found system model can estimate the height as good as the point mass system without the GPS measurements. On the other hand the point mass system model without the GPS measurements does loose accuracy as the rocket rises higher above ground.

Wrong Temperature Gradient

This system model should also have some robustness against a falsely defined temperature gradient. It has to be said that in the simulation the difference of the real and the used temperature gradient is known and the noises are adjusted properly. This will maybe not be the case in the real flight. Table 4.10 shows the errors which the state estimation makes when the temperature gradient is wrong with and without working GPS.

First of all it can be seen that with the correct temperature gradient and no GPS measurements, the error of the height estimation is still clearly below the two meter margin. In addition the median of the height error does only increases by around one meter if no GPS is available. But also the robustness of this system model can be seen, if there are no GPS measurements and the temperature gradient is 10 percent off over the whole flight it only results in a median error of the height estimation of 7.61 meter.

Temperature gradient correctness	Mean	Median
Normal with GPS	1.26	0.92
Normal without GPS	1.43	1.11
5% off with GPS	2.77	2.46
5% off without GPS	3.39	3.24
10% off with GPS	6.32	7.04
10% off without GPS	7.74	7.61

Table 4.10: Error of the height in meter by changing temperature gradient with and without GPS measurements

4.9 Sensor Outfall

In addition to the tests for the different system model implementation the possibility to simulate outfalls of the different sensors at start or during the flight was implemented for the last system model. For this the corresponding measurement noise of this sensor values can be set to infinity at a given time step. The most probable way was found that a sensor fails during the burning of the motor because of to the vibrations. Therefore the here discussed scenarios are all the same with the corresponding sensor failing at the third second of the flight while the motor is still burning.

4.9.1 GPS Outfall

The outfall of the GPS has already been discussed above for a state estimation with no GPS measurements at all. If the GPS fails during the flight the behaviour of the state estimation should resemble the one above after the outfall. The tests have shown that this is the case. If the GPS fails at second three the mean of the height error rises to 1.42 meter while the median rises to 1.11 meter. The other state variable errors rise by a small amount too, but as expected the pressure estimation error stays the same. These values show that the estimation is just slightly better than without any GPS at all.

4.9.2 Barometer Outfall

The barometer is special because if all barometers are lost, the height which is calculated out of the state vector pressure has to be completely max'ed out.

One Barometer Fails

The loss of one barometer does already make a significant change into the estimation. For example the mean of the height error rises to 1.78 meter while the median also rises to 1.39 meter. The difference between the mean and the median shows that there occur more outliers if just one barometer is active. This can also be seen in the error of the barometer estimation which rises to 0.2 hP for the mean and 0.16 hP for the median.

All Barometer Fail

More interesting is how the state estimator is performing when no barometer measurements are available at all. As expected the error of the barometer estimation rises to high values which are around 12 hP for the mean and 11 hP for the median. But on the other hand the accuracy of the height estimation does actually rise to more or less the same values as with two working barometers. This behaviour can be explained with the still working GPS. If both barometers fall out and therefore there is no height calculated out of the pressure, the height from the GPS sensor gets a higher weight due to it being the only remaining measurements on the height.

4.9.3 Accelerometer Outfall

The influence of an outfall of the accelerometer can be well shown in the plots of Figure 4.15.

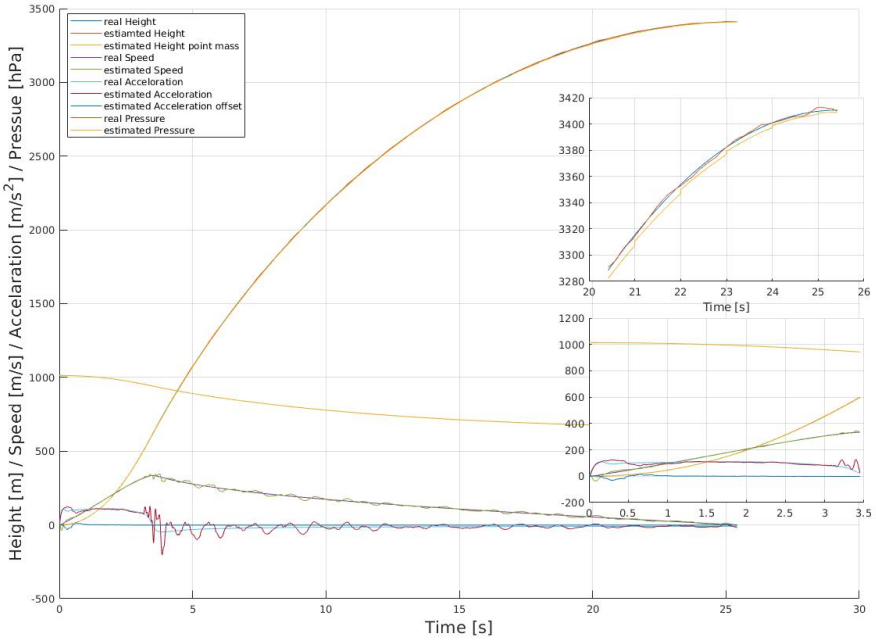

Figure 4.15: Performance over a whole flight with failing accelerometer at second 3

It shows how the acceleration estimation starts to change in a great manner after the third second. But as can be seen in the plot of the last five seconds the height estimation does not change in a great manner compared to the estimation with acceleration measurements during the whole flight. The calculated height errors do confirm this by values of 1.54 meter for the mean and 1.20 meter for the median. The pressure estimation is also not impacted as it was expected it would be. On the other hand the error of the acceleration does rise by around 10 m/s^2 and does therefore affect the speed estimation.

4.9.4 Gyrometer Outfall

Due to the fact that the pitch angle does not have such a big impact onto the acceleration as already stated above, the effect onto the state estimation if the gyrometer fails is also quite small. First of all the error which is made in the estimation of the acceleration should be examined. Against the expectation it does rise by around $1m/s^1$ for both the mean and the median value. Due to that the estimation of the height does also loose some accuracy and resembles with 1.39 meter for the mean and 1.11 meter for the median the same situation as if there would be no GPS measurements. This shows that while it is not necessary to estimate the pitch angle the measurements of the gyrometer should still be included for an optimal estimation.

4.9.5 Multiple Sensor Outfall

A requirement which has been given is that the sensor fusion should still be able to work with 2-3 sensors failing. Two sensors was already discussed by tailking about both barometer or the accelerometer (which would also be an outfall of the gyrometer, since those measurements would not be included either) failing. An additional test should therefore be on what happens when three sensors fail. For this scenario the two barometers and the gyrometer were chosen to fail because if they fail the observability of the point mass system still remains.

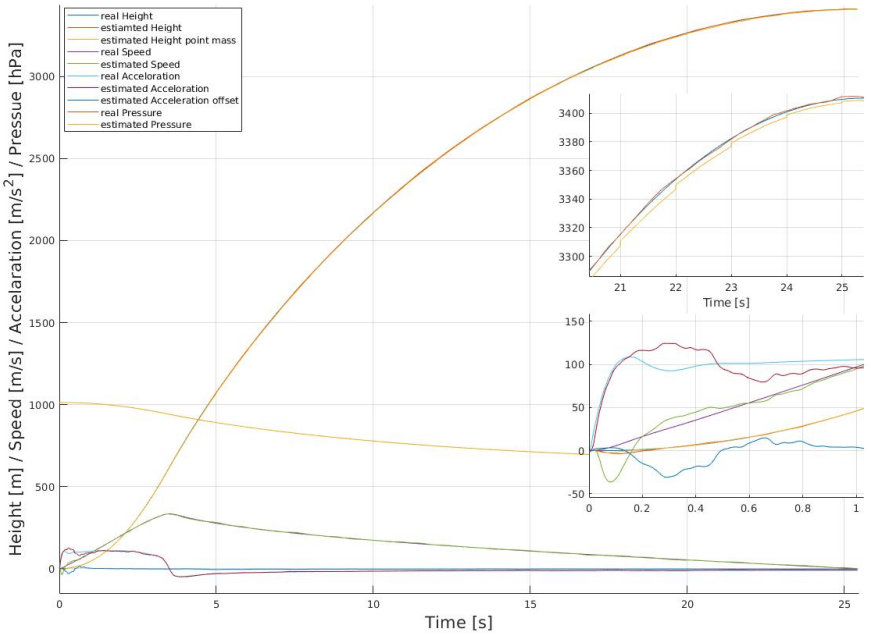

Figure 4.16: Performance over a whole flight with both barometer and the gyrometer failed

With those failed sensors the estimation does still work quite well which can be seen in the plots of Figure 4.16. This is because the GPS which provides the most accurate measurements is still working. Due to that the error of the height estimation does just rise slightly to 1.54 meter for the mean and 0.97 meter for the median. For comparison if the GPS sensor would also fail the height estimation does loose accuracy by a big amount so the error rises to 63 meters for the mean and 26 meters for the median. Also it should be said that the height error does end at 531.78 meter which is quite far off.

Chapter 5

Conclusion

After the test have been made to find the best performing solution for the state estimation, the found solution will now be compared to the requirements. After that follows an outlook for further development and a brief reflection on this project itself.

5.1 Derived Solution

The solution for the stated problem which was found during this thesis can be summarised as follows.

- The algorithm uses a discrete Kalman filter which has dynamic noise matrices for the measurements as well as the system.
- The state vector which describes the estimated system consists of the height, the speed, the acceleration, the acceleration offset and the pressure.
- The measurement noise matrices contain the noises for the GPS and the pressure set to infinity if no measurements from the GPS module or the Barometers are available. The same concept is used if it is detected that another sensor is not working properly.
- To work in the best possible way additional test flights with the really used sensors have to be made and the noise matrices have to be calculated out of that measured data.

This solution provides an optimal trade off between robustness and accuracy and an as small computational effort as possible.

5.2 Comparison Solution/Requirements

How good the given requirements are met can be described by filling out the requirements table from the chapter 1. The outcome can be seen in Table 5.1.

Requirement	**Rating**	**Aim**	**Result**
System Load	# Calculation steps per loop	$<$ 5000	$\approx$ 2500
Precision	Error between estimation and ground truth	$<$ 2m in height	Median $\approx$ 0.92 m
Settling Time	Time from first reliable to optimal estimation	1 s after burnout	0.9 s after ignition
Reliability	Working Estimation with # failed sensors	2-3 sensors	3 sensors
Modularity	Effort needed to change a sensor	$<$ 10 h work	5-10 hours

Table 5.1: Requirements table filled out

5.2.1 System Load

The system load is calculated by finding the simple calculations (addition, subtraction, multiplication and division) which are needed for each equation of the state estimator to solve. Since the used system model consist out of five state variables and there are 5 measurements at the input (GPS, Accelerometer, Barometer 1, Barometer 2 and height out of pressure) most matrices used are 5x5 matrices. Therefore the multiplication with each other needs 225 calculation steps (4 additions and 5 multiplications for each of the 25 values in

the resulting matrix). Also the matrix inversion with the Gauss-Jordan elimination has a complexity of 2^3 which results in about 125 calculations for a 5x5 matrix. In addition two equations which do not come from the Kalman filter equations itself have to be calculated. These are the inclusion of the pitch angle on the acceleration measurements to get the pure vertical acceleration and the calculation of the height out of the state vector which contains the actual pressure. With this the following calculation steps per equation can be estimated as shown in Table 5.2.

Equation	Steps	Total
$\hat{x}[k] = Ad \cdot x[k-1]$	45	45
$P[k] = Ad \cdot P[k-1]Ad^T + Gd \cdot Q[k] \cdot Gd^T$	225+225+25+75+125	775
$a_z[k] = a[k] \cdot cos(phi[k] \cdot \frac{\pi}{180})$	1+2+1	4
$h_{pres} = \frac{((1-p_x[k]/p0)^{Coeff} \cdot T0}{Tgrad}$	1+1+1+3+1+1	8
$K = P[k] \cdot C^T \cdot (C \cdot P[k] \cdot C^T + R[k])^{-1}$	225+225+225+225+25+125	1050
$x[k] = \hat{x}[k] + K \cdot (y[k] - C * \hat{x}[k])$	5+45+5+45	100
$P[k] = (I - K \cdot C) \cdot P[k])$	25+225+225	475
Total calculation steps	45+775+4+8+1050+100+475	2457

Table 5.2: Calculation steps per estimation cycle

The table shows that a rough estimation of the needed calculation steps is around 2500 calculations per cycle. This is half of the maximum given by the requirements and does therefore fit the requirement.

5.2.2 Precision

The tests have shown that the mean and the median of the height error are with 1.29 and 0.92 meters clearly under the aimed 2 meters. Despite of that there are peaks during the estimation where the error rises above the aimed 2 meters. This results in errors around 2 to 4 meters and can sometimes even rise to 6 or 7 meters. While this does not exactly match the requirement at a deeper sight, some improvement with sensors adjustment (faster and more accurate GPS and pressure sensors) should be possible.

5.2.3 Settling Time

No real settling time could have been defined in the tests because the error peaks rise above the aimed two meters during the whole flight. But if the settling time is used in terms of the time until the optimal possible estimation is working properly, it can be seen in the plot from Figure 4.13 that this is around 0.9 seconds after the ignition. After this moment the estimation for the acceleration and speed are settled which means the height estimation should be at its best possible estimation.

5.2.4 Reliability

The found sensor fusion algorithm does exceed the reliability. It does work quite well with one or two failed sensors and can also estimate the height properly if the "right" three sensor do fail. If the system loses all its height measurements the error does rise above 500 meters at the end and is therefore no more useful. But by having three height measurements this still remains inside the given borders of the requirement.

5.2.5 Modularity

The modularity is difficult to evaluate in clear values. With the derived simulation it should be possible to adjust the algorithm for a new sensor quite fast (5 to 10 hours of work), given that the simulation is known and enough tests with the new sensors have been made and logged. This test data can then be used to derive the needed sensor model and the corresponding noise matrices will be calculated automatically during the simulation.

5.3 Outlook

As always there are points which still can be developed further for better results. These are stated here.

5.3.1 More and Better Documented Test Flights

First of all more and better documented test flights with the used sensors should be made. These would result in better sensor models and could therefore increase the accuracy of the derived noise matrices. Also the found algorithm should be ported into an embedded system and then be tested in those flights. With this the algorithm itself can be validated and also adjusted, which should result in an overall more reliable state estimation.

5.3.2 Slower Cycle Time

A quite interesting change could be to reduce the frequency of the control cycle. This would open the door for other state estimation algorithms which require more computational power but also provide more accurate estimations.

Extended Kalman Filter

For example an extended Kalman filter could be implemented and with its help the non-linear dependencies of the height from the pressure could be described directly inside the system model. In addition the force from the motor as well as the drag from the air breaks could be included into the system as system inputs, because they are also non-linear and time depending (since the weight of the rocket does change over the flight).

Back Calculation for Measurements That Arrive Out of Time

Also if there is more computing time for the sensor fusion the principle of including too late arrived measurements could be used. With this the GPS measurements could even more positively affect the whole estimation. This would also come in handy when the algorithm is implemented on a real system because it is possible that too late arriving measurements are a common problem in the real system.

5.3.3 Temperature Gradient

Last but not least the temperature gradient which was referenced during the whole thesis is a topic which can be more thoroughly researched. This would lead in a better understanding in the interaction between the pressure and the height. From there either the simulation and/or the algorithm itself could be optimised with this knowledge.

5.4 Thanks

First of all credits have to be given to the whole ARIS team which supported this thesis with help and advice and great parts of an already working simulation which could be used to generate the trajectories. Special thanks goes to Thomas Lew and Fabian Lyck which provided their knowledge and the logging data from the test flights. But also the EPFL Rocket Team (ERT) team of the Matterhorn competition has to be thanked which saved the success of this thesis by providing freely their logging data. Finally I would like to thank Prof. Marcel Joss and Prof. Thomas Hunziker for providing their knowledge.

5.5 Reflection

First of all, all the requirements could be met which of course is satisfying. But it has to be said that the tests were made in an optimised simulation environment and if this algorithm will preform likewise in reality has yet to be tested. Also since the final product consist mainly of concepts and simulation the functionality and suitability for the target system was hard to proof. This circumstance was even increased in the first weeks by the lack of usable sensor data from test flights, which was fortunately solved by the ERT team which provided their log data.

Personally this thesis was a frightening new experience in the beginning because it covered a whole new theoretical field for me (sensor fusion/ Kalman filter). Also the range of this thesis was scattered wide by the multiple concepts form different electric engineering fields that had to been developed and used. On the other hand because of this I have earned more theoretical knowledge during this thesis than in any other before.

Bibliography

[1] Akademische raumfahrt initiative schweiz. Retrieved from `https://aris-space.ch/`, 2018.

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

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

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

[5] Nancy Hall. Earth atmosphere model. Retrieved from `https://www.grc.nasa.gov/WWW/K-12/airplane/atmosmet.html`, 2015.

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

Appendices

Appendix A

Assignment

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

Diplomarbeit im Fachbereich Elektrotechnik und Informationstechnologie

Aufgabe für Herrn Michael Kurmann

ARIS – Data Fusion für eine Sounding Rocket

Fachliche Schwerpunkte
(Mit X markierten Text von Projektskizze im Abschnitt „Fachliche Themenbereiche“ übernehmen)

Einleitung
Eine Sounding Rocket hat in der Regel zahlreiche Sensoren an Bord, die für einen sicheren Flug benötigt werden. Im Rahmen dieser Arbeit soll untersucht werden, wie diese verfügbaren Daten möglichst optimal für einen gewünschten Flugstatus und Flugverhalten verwendet werden können.

Aufgabenstellung
Im Rahmen dieser Arbeit soll untersucht werden, welche Sensoren wie miteinander verknüpft werden können, um während des Fluges der Rakete mögliche Abweichungen von der nominalen Flugbahn zu erkennen. Damit die Information der verschiedenen Sensoren verknüpft werden können (Data Fusion), sollen…

- …Übereinstimmung und Abhängigkeiten zwischen ihnen aufgezeigt werden,
- …und allfällige Störgrössen identifiziert und charakterisiert werden.

Ausgehend von diesen Ergebnissen soll für einen exemplarischen Flugparameter, z.B. Höhe, ein Lösung erarbeitet werden und mit einem Funktionsmuster die Zweckmässigkeit dieser Lösung aufgezeigt werden.

Termine

Start der Arbeit:	Montag 19.2.2018
Zwischenpräsentation:	Zu vereinbaren im Zeitraum 9.4. -4.5.2018
Abgabe Broschüre-Doku:	Freitag 25. Mai 2018, per Mail an Betreuer und H. R Andrist
Abgabe Schlussbericht:	Freitag 8. Juni 2018, vor 15:00 im Sekretariat
Abgabe Poster-File:	Montag 18. Juni 2018 per Mail an Betreuer und H. R. Andrist
Abschlusspräsentation:	Zu vereinbaren im Zeitraum 11.6. – 30.6.2018

Dokumentation

Der gebundene Schlussbericht ist in 4-facher Ausführung zu erstellen. Er enthält zudem zwingend

- die folgende Selbstständigkeitserklärung auf der Rückseite des Titelblattes:
  „Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche verwendeten Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.
  Horw, Datum, eigenhändige Unterschrift"
- einen englischen Abstract mit maximal 2000 Zeichen.
- Ein Titelblatt mit: Name des Studierenden, Titel der Arbeit, Abgabedatum, Dozent, Experte, Studiengang, Klassifikation (Einsicht/Rücksprache/Sperre)
- Eine CD-Hülle, innen, auf der Rückseite des Berichtes

Alle Exemplare des Schlussberichtes müssen termingerecht abgeben werden. Zusätzlich muss zu jedem Exemplar eine CD mit dem Bericht (inkl. Anhänge), dem Poster und den Präsentationen, Messdaten, Programmen, Auswertungen, usw. unmittelbar nach der Präsentation abgeben werden.

Ein Poster sowie Unterlagen für eine Diplomarbeitsbroschüre sind gemäss den offiziellen Layout-Vorgaben termingerecht einzureichen.

Fachliteratur/Web-Links/Hilfsmittel

Geheimhaltungsstufe: Einsicht

Verantwortlicher Dozent/Betreuungsteam, Industriepartner

Dozent Prof. Marcel Joss marcel.joss@hslu.ch

Industriepartner ARIS (Akademische Raumfahrt Initiative Schweiz)
Oliver Kirchhoff
kioliver@student.ethz.ch

Experte

Werner Scheidegger
werner.scheidegger@sbb.ch Tel. +41 79 611 76 04

Hochschule Luzern
Technik & Architektur

Prof. Marcel Joss

Appendix B

Lists

List of Figures

List of Tables