Lucerne University of Applied Science & Architecture

ARIS - Data Fusion for a Sounding Rocket

Bachelor Thesis

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

Declaration

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

Horw, May 10, 2018

Abstract

This is the Abstract

Contents

Chapter 1

Introduction

Figure 1.1: Official logo of the competition project 2018

The Academic Space Initiative Switzerland (ARIS) figure:1.1 is a student group which tries too compete in the yearly Intercollegiate Rocket Engineering Competition (IREC). The goal of this competition is to build a rocket which can fly autonomous at a predefined apogee (10000 feet). The competitors are graded by the documentation about the project, how much of the parts they engineered/produced itself and how precises the apogee was hit.
To aim for the right apogee a Control algorithm is implemented. This algorithm relays on the information of different sensors to determine the rockets actual state. Cause there are different sensors to measure the same value a algorithm which fuses those data would come in handy. With this fusion algorithm it should also be possible to be more accurate as with each sensor on its own. The purpose of this thesis is to implement a simulation and with its help find the algorithm which is most suitable for this task.
For this the problem as well as the desired solution will be defined in this chapter. After that the models for the different sensor that will be used as well as the pro and con of different state estimators are discussed at beginning of chapter 2. In addition, different possible system models are also described in this chapter. Chapter 3 will then show, how the simulation and the fusion algorithm are implemented in detail. To verify that the implementation is working as intended, the results of the simulation are discussed in chapter 4. In the last chapter 5 a summary of the achieved knowledge, as well as a comparison between the desired and the implemented solution will be stated.

1.1 Purpose

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

1.2 Research

Sensor/data fusion and state estimation is a well established engineering field. Especially since the 1960 when Rudolf A Kalman published his paper for the Kalman filter. Therefore there is already a lot of previous work which can be used in this thesis. For this thesis two books are used which provide the needed theory, this are by name David (2004) which contains basic theory about kalman filters. The second book Simon (2006) is more focused on different approaches of state estimation and provides also different solution to common problems that occur while implementing a state estimation. In addition the Master Thesis Bryan (2012) accesses more ore less the same issue as this thesis. Therefore it will be used mainly in the conceptional part of this paper.

1.3 Sensors

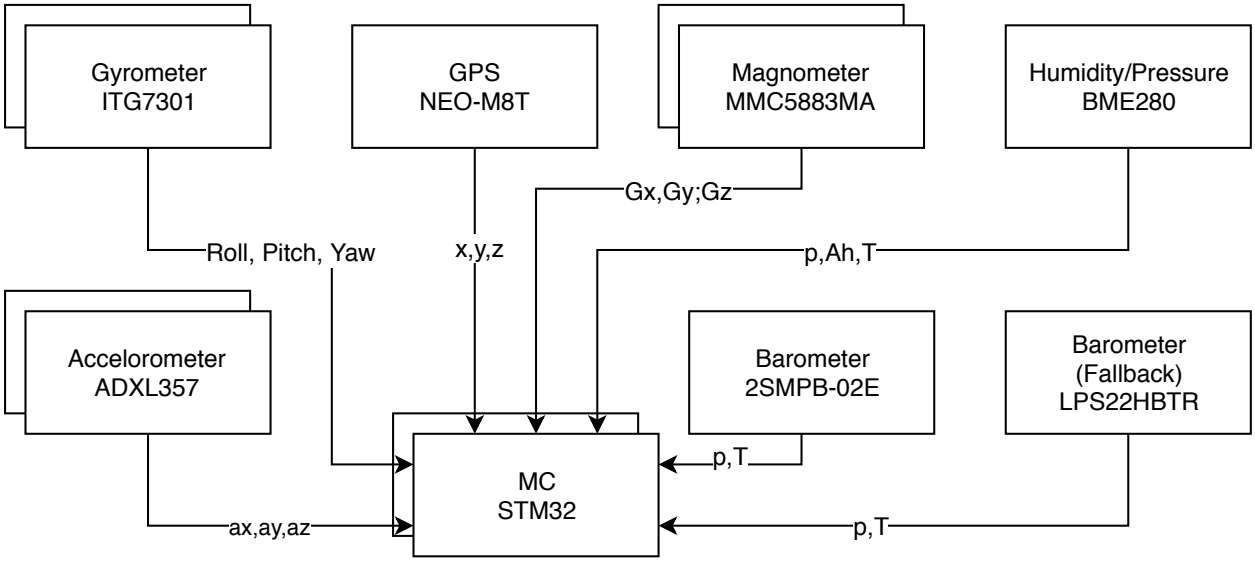

Figure 1.2: Sensor Network

As mentioned above different sensors are used in this years competition. This used sensors and their settings will be described in this chapter.

1.3.1 Accelerometer

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

1.3.2 Gyrometer

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

1.3.3 Barometers

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

1.3.4 Temperature

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

1.3.5 Magnometer

Also there are two Magnometer in the sensor network 1.2. These measure the strengths of the surrounding magentic field. This can be used to determine the direction regarding the north pole. Due to the fact that the algorithm that will be developed in this thesis does not include the X- and Y-Axis, these sensor will not be used for the sensor fusion.

1.3.6 GPS

For next years competition differential GPS will be implemented with the help of two μblocks modules. This taken measurements are more precise as the rest of sensors but are taken much slower on a rate like 1 Hz. Therefore the algorithm should takes those provided measurements and interpolate between them with the data from the other sensors.

1.4 Problems

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

1.4.1 Different Sensors

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

1.4.2 System Load

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

1.4.3 Precision

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

1.4.4 Settling Time

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

1.4.5 Reliability

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

1.4.6 Modularity

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

1.5 Requirements

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

Table 1.1: Requirements table

As seen in the table 1.1 five requirements were drown out of the problem analysis. First of all, there is critical requirement the system load. This is given as critical cause it is needed that the algorithm is small enough to be run on a embedded system. Any solution that would not fit this requirement would be pointless in the frame of this thesis. Secondly there are two requirements which are tied together, the precision and settling time. Where the precision describes what a optimal estimation is under the context of this thesis, the settling time relies on this to be defined. The desired precision will be needed to ensure a possible good control.

1.6 Desired Solution

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

Chapter 2

Approach

2.1 Verification

First of all the test concept has to be defined on which the developed algorithms will be tested. For this the following concept figure 2.1 was developed.

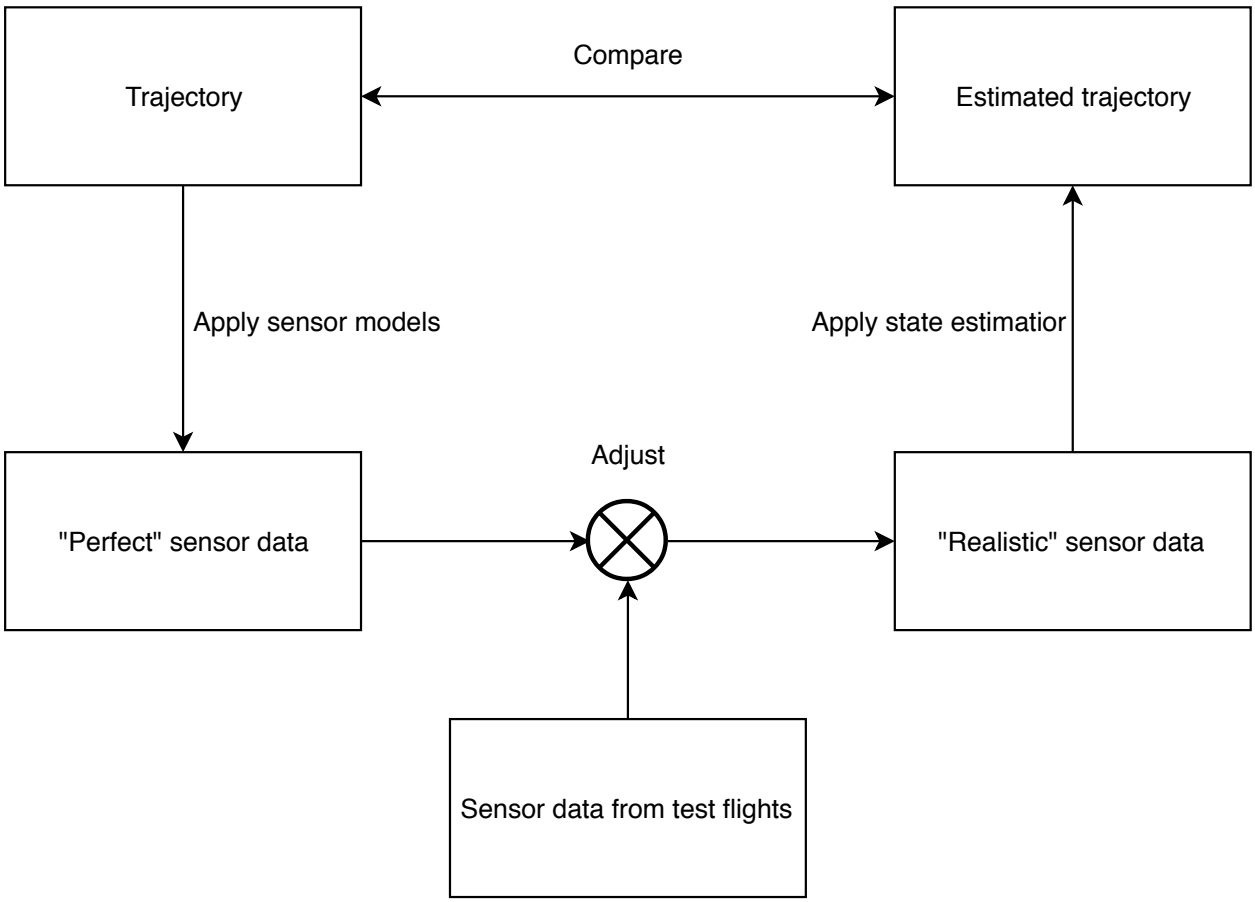

Figure 2.1: Verification Concept

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

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

2.2 Sensor models

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

2.2.1 Accelerometer

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

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

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

2.2.2 Gyrometer

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

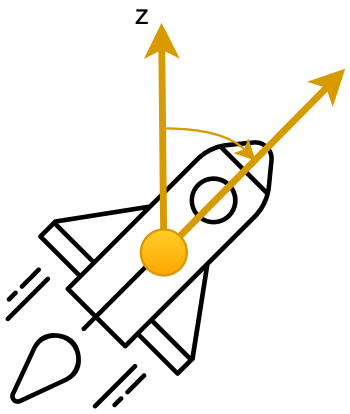

Figure 2.2: Pitch angle visualisation

2.2.3 Barometer

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

Pressure

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

$$P = P0 * (1 - \frac{TCoeff * h}{T0})^{5.255}$$

with: PO = Pressure at ground level
TO = Temperature at ground level This is more or less accurate till 11 000 meter under the condition that the Temperature gradient is determined correct.

Temperature

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

$$T = T0 - Tgrad * h$$

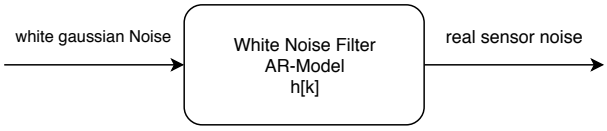

Figure 2.3: White noise filter concept

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 downsampling the height vector with the right factor.

2.3 Noise generation

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

For this the yule walker equation should come in handy to generate a so called AR (auto regressive) model of the noise system. But to get the correct coefficients, the noise has to have a steady mean value preferably it should be zero mean.

2.4 State estimator

Which ones are there and what are their positives and negatives, in short how do they work I will use kalman filter with time depending system and sensor noise, but i have yet to define how i do this in peticullar. I should also make some pictures and such things.

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

2.4.1 Kalmanfilter

Its structure provides the optimal estimation of the standard deviation estimation error as long as the noise is Gaussian. But there lies the problem, a physical system is most often not linear. Also the estimated system as well as its variances over the time have to be known to provide this optimal estimation. If the noise matrices of the system are static, the filters gain matrices aim for a fix value and can therefore be calculated in beforehand. This reduces the computational effort by a significant amount. David (2004). It should be mentioned that even if the noise is not Gaussian, the kalmanfilter is still the best linear estimator as long as the system and its properties are well known Simon (2006).

2.4.2 ROSE

The ROSE(rapid ongoing stochastic estimator) is in simple terms three kalman filters in one. Where the main filter is used 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 be estimated. Due to this, this sensor needs more computational effort David (2004).

2.4.3 Extended Kalmanfilter

The extended kalmanfilter provides additional parts to better access nonlinearity in the observed system. This by not estimating the state of the system but by estimating the linearized change of the state to the past sate. For this the systems equations have to be derived around the current working point in every estimation state. This is some sort of Bootstrap solution because the nominal point on which the derivation happens are estimated in the process and this estimates are then used to estimate the change between this estimation and the next. Therefore the computational effort exceeds even further Simon (2006).

2.4.4 Unscented Kalmanfilter

The unscented kalmanfilter takes the unscented transformation in use to calculate the different interpolating steps. The unscented transformation uses different points on which the mean and the covariance of a state is known to estimate the change in the next iteration much better than the normal linearized approach. But for this the unscented Kalmanfilter needs also to apply the unscented transformation onto the state vectors in each iteration and does therefore need even more computational effort.

2.4.5 H∞ filter

The H∞ filter is a more diverse approach then the ones described above. It was developed to access the problem when the to be observed system espacially its noise is not well known. Also in contrast to the kalman filters the H∞ filter minimizes the worst-case estimation error in stand of the standard deviation of the estimation error.

2.5 Choosing

If the requirements table 1.1 is taken into the consideration of finding the optimal solution, two main requirements occur that define this decision. First the system load is a critical requirements and has therefore to be addressed in this process. Also for the requirement of modularity the algorithm should be as simple as possible. If taken in regard that the system is more or less well known and that the noise can be determened with the simulation and the log data from previous test flights, a normal kalmanfilter seems to be the most fitting solution. This because the performance of the rocket and the sensor should stay the same during each flight.

2.6 System Model

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

2.6.1 Point Mass

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

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

This would reduce the dynamics matrix of the system to a 3x3 matrix with with only two 1 in it:

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

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

2.6.2 Point Mass with Pressure and Temperature

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

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

While this solves the problem stated above, it also produces a new. The system model can only describe linear dependencies between the state variables, but the relation between the pressure and the height is clearly non linear in each atmospheric model. This dependency can be linearized, but if done so, it does resemble the athmosperic model with less accuracy. This as to be taken into account an has to be 'told' to the state estimator. The change of the temperature depending on the height is a difficult manor. But to make it simple I use KT_h = -0.0065 °/m in this.

This results in a 5x5 dynamic system matrix which looks like this:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ KP_h & 0 & 0 & KP_T & 0 \\ KT_h & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.6.3 Point Mass with Angle, Pressure and Temperature

The same solution as above can also be applied for the pitch angle. The linearization of small angle changes in the sinus is the angle itself. The assumption that the pitch angle of the rocket wont be to great during the ascention is suitable due to its flight caracterices. This would change the state vector from above into the following.

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

But now how to do this ? describe here how the system description looks in particular
Those described different system will no be implemented in matlab to find the best possible.

Chapter 3

Implementation

3.1 Sensor Model

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

3.1.1 Perfect Sensor

In general, the perfect sensor data are calculated like stated in chapter 2.

3.1.2 Accelorometer

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

3.1.3 Gyrometer

The gyrometer measurements are difficult, because it can not be directly calculated from the height. Therefore the best option to generate this data, is to fit a polynom onto the measured data for the test flights. Which is done like this in the implementation: bla bla bla bla

3.1.4 Barometer

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

3.1.5 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.

3.1.6 Noise

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

If done so, polynoms are fitted on this measurements with the least squared error method. Those polynoms represent the function which is assumed to be the noiseless data with possible offsets. So if now

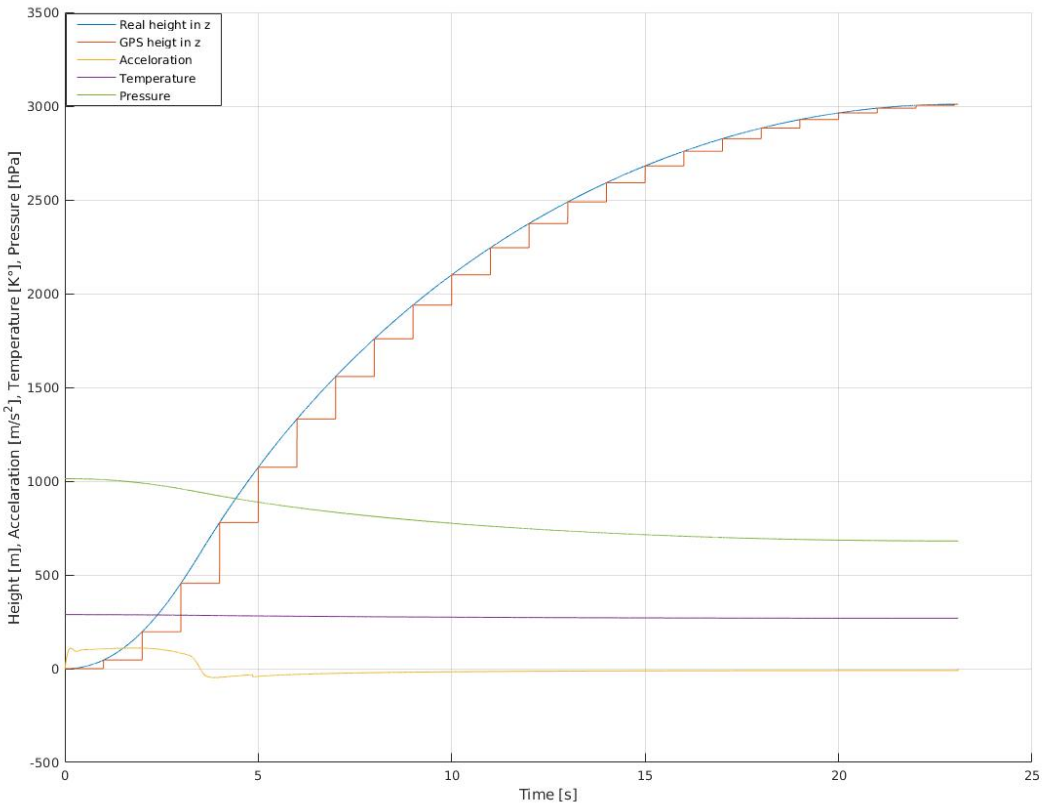

Figure 3.1: Generated sensor data

the test flight data is subtracted by those polynomial curves which results in the noise without a mean. From this point on this noise can be examined on its parameters, like the power density, the probability distribution and the variance figure bla.

This noise data can now used tho solve the yule walker equation to get an AR-model. But first, the data has to be resampled so that the AR-models can be used proper in the simulation. With those AR-models, the noise can be regenerated by filtering white noise with the correct variance. This generated noise can now be compared to the noise form the test flight data.

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

3.1.7 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 icognition are help full, because the value that should be measured is known.

3.1.8 Gyrometer

3.1.9 GPS

3.1.10 Real Sensor

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

3.1.11 Accelorometer

To generate the real acceleration data, in addition to the noise, also an offset has to be laied of it.

3.1.12 Gyrometer

3.1.13 Barometer

3.1.14 GPS

3.2 State Estimator

As stated the used state estimator is a kalman filter with dynamic noises on the system as well as on the measurements. This algorithm works in 4 main equations which can be devided into prediction and correction steps figure 3.2.

3.2.1 Prediction

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

$$\hat{x}[k+1] = AD * x[k] + Bd * u[k]$$

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

$$P = Ad * P * Ad' + Gd * Q[k] * Gd'$$

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.

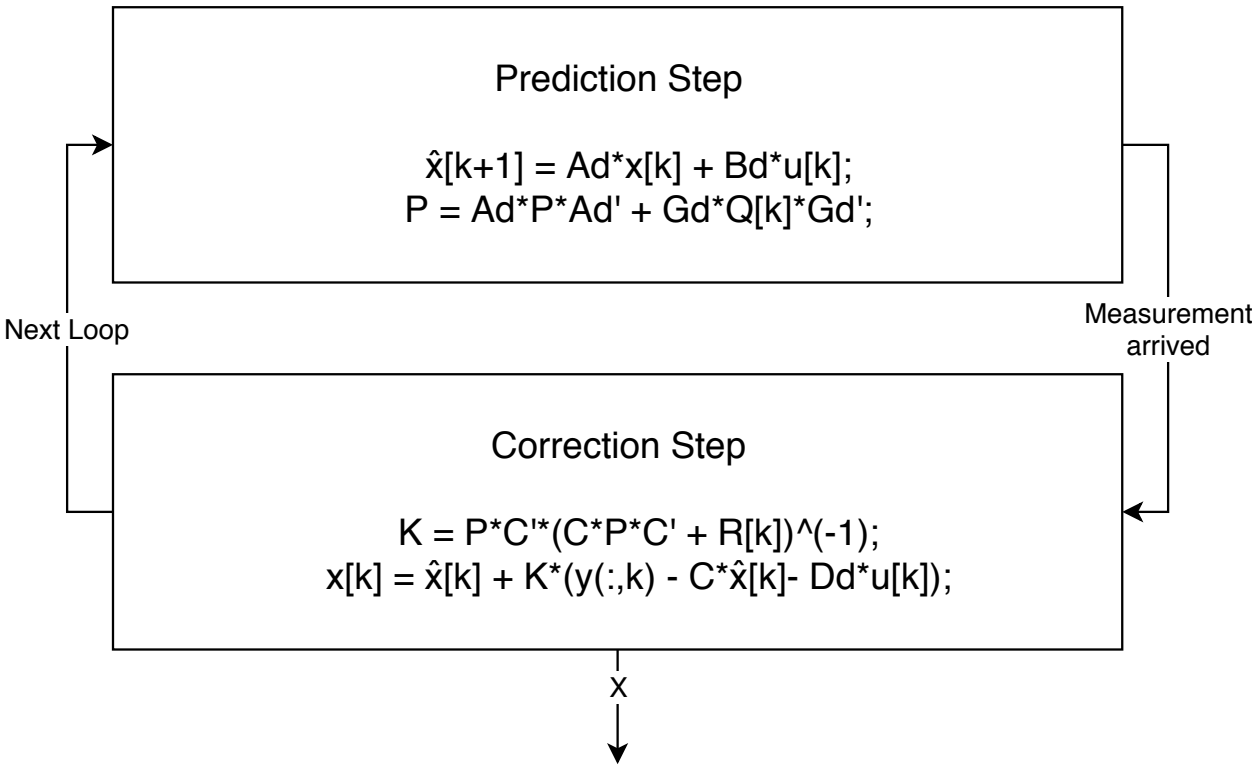

Figure 3.2: Kalmanfilter

3.2.2 Correction Step

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

$$K = P * C' * (C * P * C' + R[k])^(- 1)$$

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

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

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

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

3.3 System Model

The models are modelled as stated in chapter 2 but from this different uses can be made out of this. Due to this, these different implementation are tested and evaluated in the next chapter

3.4 Measurements noise

3.5 System noises

For the system noise the behavior of the system during the flight has to be examinated.

3.5.1 Loop

Chapter 4

Tests

Here come the tests
    For the different plots of the different approaches

Chapter 5

Conclusion

5.1 Derived Solution

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

5.2 Comparison Solution/Requirements

Compare the derived solution to the requirements

5.3 Thanks

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

Bibliography

Bryan, T. M. (2012).

David, W. (2004). Application of the kalman filter to rocket apogee detection.

Simon, D. (2006). *Optimal state estimation : Kalman, H∞, and nonlinear approaches*.

Appendices

This is the Appendix