# TTK4250 - Sensor Fusion

Group 42
Martin Albertsen Brandt
Martin Eek Gerhardsen

November 27th 2019

Department of Engineering Cybernetics

## Abstract

This report will highlight and discuss the results of the three graded assignments for the course TTK4250 Sensor Fusion. The code implemented was tested and tuned for both simulated and real datasets.

## Contents

## 1 Introduction

## 2 Graded Assignment 1

The IMM-PDAF combines Interactive Multiple Models (IMM) with Probabilistic Data Association Filter (PDAF). Relevant theory can be found in [1, p. 100-101, 120 – 122].

### Results

#### Simulated dataset

Tuning the full system for the simulated data required tuning of, in total, four systems. These were the Extended Kalman Filters, for the Constant Velocity and Constant Turn-rate modes, the IMM, and the IMM-PDAF.

For the CV-model, the measurement noise covariance parameter used was 4, while the acceleration noise covariance parameter used was $8 \times 10^{-2}$. Similarly, the same measurement noise covariance parameter was used for the CT-model, as the measurement noises should be similar. Furthermore, the acceleration noise covariance parameter used was $5 \times 10^{-3}$ and the turn-rate noise parameter used was $3 \times 10^{-4}$. Tuning these parameters would have some impact on the consistency, higher noise would lower the NEES, while lower noise increases the NEES, which is to be expected as we are essentially changing how much the filter should trust the measurements contra the estimates. From fig. 1c, it can be seen that both models handle their trajectories well.

The IMM was tuned using the transition matrix $\pi$, which was tuned as eq. (1). With the CV-model as the first model and the CT-model as the second, the argument is that if the IMM is in a given mode, it should most likely stay. Still, it may be beneficial to be more likely to change to, and stay as, the CT-model. When observing the trajectory, see fig. 1c, it seems to primarily use the CV-model, and then increasing the transition probability for changing to the CT-model would speed up this transition.

$$\pi = \begin{bmatrix} 0.92 & 0.05 \\ 0.08 & 0.95 \end{bmatrix} \tag{1}$$

Finally, for the IMM-PDAF, the clutter rate $\lambda$, detection probability $P_D$ and the validation gate were tuned. The clutter rate was tuned to $\lambda = 1 \times 10^{-4}$, and as there were very few false alarms, this could be chosen as a low value. Furthermore, as the only place the clutter rate is used in the IMM-PDAF is to scale the probability of no detection, there was really no use further decreasing $\lambda$, as it already would put the association likelihood ratio for no detection very low. The detection probability is tuned to be rather high, with $P_D = 0.95$, as it is expected that there is a good chance of detection for this dataset. A validation gate size of $5^2$ seemed also to give decent results. Increasing it would generally not make the tracker worse overall, but would increase the magnitude of certain errors, which would then spike. Decreasing it would of course make it more difficult for any measurements to be gated, such that this value seemed to be the best compromise between gating all and no measurements. The resulting consistency plot can be seen as fig. 1a, and the average NEESes were for position 1.9248, for velocity 1.1427 and overall 3.0887.

**Joyride dataset**

Initially, the IMM-PDAF for the Joyride dataset was tuned similarly to the tracker for the simulated dataset, and especially for the low noise CV model and the CT model, similar values were used. Furthermore, it should be noted that the Joyride dataset also was noisier, such that more lenient tuning was necessary. For this dataset, a high noise CV model was also included, which could *take over* if both the low noise CV model and the CT model were uncertain. Otherwise, the overall structure of the tracker is the same as for the simulated dataset.

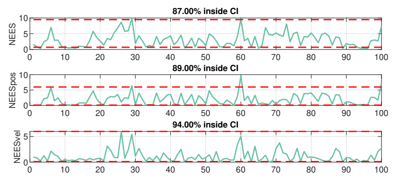
The measurement noise covariance parameter was again the same for the CT and two CV models, and was set to $25^2$. Though this seemed rather higher than necessary, however lowering it would especially push higher the positional NEES. Similar responses were generated from tuning the acceleration noise covariance parameters and turn-rate noise covariance parameter for the different models. Then, for the low noise CV model, the acceleration noise covariance parameter was tuned to 0.5, for the high noise CV model, the acceleration noise covariance parameter was tuned to 20 and for the CT model, the acceleration noise covariance parameter was tuned to $1 \times 10^{-4}$ and the turn-rate noise covariance parameter was tuned to $2 \times 10^{-4}$. Similarly to the results from the simulated data, these models handle their trajectories well, as can be seen from fig. 1d.

As for the IMM used in the simulated dataset, the transition matrix $\pi$ was tuned to prefer to stay in the CT-mode, so that the tracker should prefer to continue a turn if it has begun. Since the high noise CV model was included primarily to handle high noise areas where it could be unambiguous whether to use the low noise CV or CT model, this transition probability was set lower than the others, as those should be preferred. From observing the track and measurements, there also seemed to be unambiguous which of the other modes each should transition to, and as such these transitions were set to be equally likely, see eq. (2).

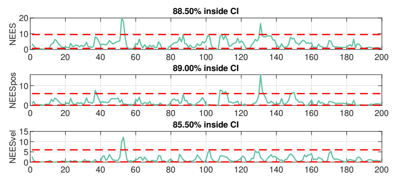
$$\pi = \begin{bmatrix} 0.900 & 0.025 & 0.075 \\ 0.050 & 0.950 & 0.075 \\ 0.050 & 0.025 & 0.850 \end{bmatrix} \tag{2}$$

The IMM-PDAF was tuned to have a lower clutter rate and higher detection probability than for the simulated data, while keeping the gate size. The clutter rate was tuned to be $\lambda = 5 \times 10^{-5}$. Higher values would inevitably make the tracker loose track and converge to one mode, rendering the tracker useless. As the clutter rate affects the likelihood ratio for no detection, a higher clutter rate implies a higher likelihood compared to a lower clutter rate, and a higher value would be given to the association probability for no detection. The detection probability was tuned to $P_D = 0.97$, which is a rather high value. It was possible to reduce $P_D$ and, specifically, improve the percentage where the NEES is inside CI, but at a cost of the NEES for the velocity, and a higher $P_D$ was chosen so that the NEESes would be more similar.

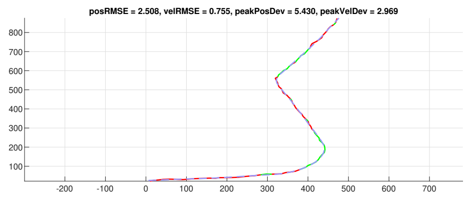
The resulting consistency plot can be seen fig. 1b, and the average NEESes were found as 1.6653 for position, 1.3966 for velocity and 3.4346 overall. Both from the plots and the averages tend to be closer to the lower bound of the confidence interval, though as can be noted, there were certain spikes in the plot. These were difficult to as they seemed to originate from especially bad measurements, and a compromise was made where the consistency would tend to be low, in exchange for limiting the spikes.



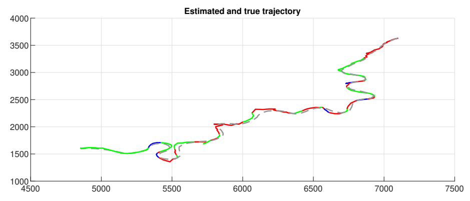
(a) NEES for simulated data



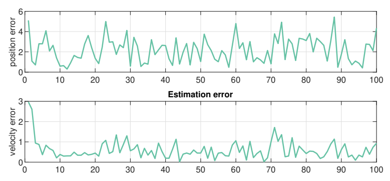
(b) NEES for Joyride



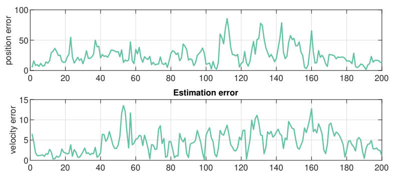
(c) Trajectory for simulated data



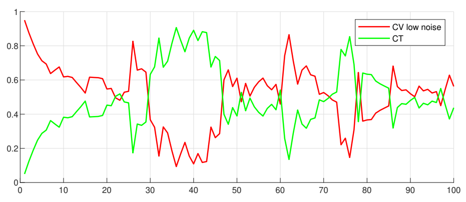
(d) Trajectory for Joyride



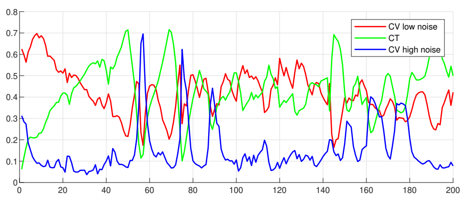
(e) Errors for simulated data



(f) Errors for JoyriThe results of this tuning gave de



(g) Probabilities simulated data



(h) Probabilities for Joyride

Figure 1: The same colour code was utilised for visualising the active modes in the trajectories as the probabilities. For the trajectories, ground truth is represented as a dashed line.

## Assumptions

The IMM-PDAF works under the single-target assumptions, as seen from [1, p. 105], which all seem reasonable considering both the simulated and real datasets. In the simulated dataset, there is initially good reason to suspect that these assumptions were considered when the dataset was generated, and it may therefore be more interesting to consider the real dataset. As there are a smaller number of measurements for each time step than there seemed to be for the simulated dataset. Therefore, it may seem like the job of the tracker for this dataset is, primarily, to consider whether there is a detection or not. In certain cases, the measurements are also pretty far away from each other and the track, which could also make it easier for the tracker. Since it is likely that several raw measurements may originate from the sensor, like a RADAR, it is common to apply some form of clustering to reduce the measurements from the target to one single measurement. Depending on the method for clustering, this is very likely to make the tracker work, and it is computationally a good idea to use the single target assumption, though this clustering may also cluster bad measurements as well, which may make the tracking more difficult. This was not further explored here, as there doesn't seem to be a need for it.

There are certain simplifications done in this algorithm to make it more viable, and as mentioned above, assuming only one measurement may originate from the target allows for a more computationally efficient algorithm.

Additionally, as the associations (and modes) are seen as mixtures, then as to not end up with an extremely large number of values, the mixtures can be reduced by moment matching. This also makes the tracker a more computationally efficient algorithm, and keeps the memory usage manageable.

Gating of the measurements, is done, for each measurement, by checking the NISes for all modes given this measurement. If one is gated, it is gated for all. The assumption seems to be good, as it then seems like at least one of the models can *accept* this measurement. There may still be cases where a stricter gating rule could've been considered, but for these datasets, there haven't been any specific cases where this has caused problems.

Certain simplifications were also made to make the code easier to implement, which probably slows the tracker slightly down, as when calculating the log-likelihood ratios, the *update*-method from the IMM was utilised, and most of the results were discarded. If there was a need to save a some small amount of time, or reduce the number of operations done, there are some places in the code where one could save time. This could be necessary if there was a need to run the tracker online.

Using the EKF, with the CV and CT models, is also a simplification, especially for the joyride dataset. For the simulated dataset, these models were probably used to generate the dataset, it is possible to get very good result using these models, as seen from fig. 1c. The joyride, however, is different, as there is not *really* an underlying model which generated this dataset. Still, by using several simpler models, decent tracking was achieved for the joyride dataset as well, as can be seen from fig. 1d.

# 3 Graded Assignment 2

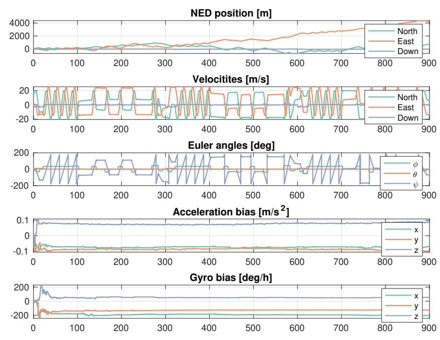
An error-state Kalman filter (ESKF) for a GNSS-aided fixed-wing UAV was implemented in MATLAB. The implementation is based on the standard formulation in [2]. But most notably, IMU sensor correction matrices has been added to counteract any mounting errors, scaling errors and orthogonality errors in the accelerometer and rate gyro. Furthermore, leverarm componsation for the GNSS receiver is also implemented.

## 3.1 INS for simulated fixed-wing UAV

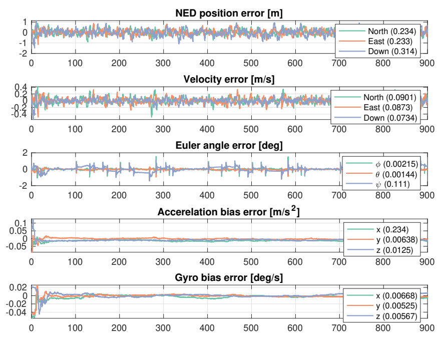
The ESKF was first tuned to a simulated dataset. The GNSS measurement standard deviation was tuned to 0.4 in each degree of freedom. The measurement noise covariance is therefore $R = 0.16I^2$. For the accelerometer the measurement noise covariance and bias driving noise covariance was tuned to be $q_a = (4 \times 10^{-2})^2$ and $q_{ab} = (1 \times 10^{-3})^2$ respectively. Similarily for the rate gyro the measurement noise covariance and bias driving noise covariance was tuned to be $q_\omega = (8 \times 10^{-4})^2$ and $q_{\omega b} = (1 \times 10^{-6})^2$ respectively. Finally the time constants in the Gauss-Markov bias processes were both tuned to be $T_b = 1 \times 10^8$ s.

Firstly, the tuning was based on common sense. For instance the GNSS measurement covariance was initially based on a reasonable guess from a physical perspective and then more thoroughly tuned. This more thorough tuning was based on calculating and plotting the errors in position, velocity, attitude and bias. Furthermore the corresponding NEES in position, velocity, attitude and biases, as well as the overall NEES and NIS was calculated and analysed during the tuning process.

The resulting position estimate is plotted in fig. 3a, the state and state errors are presented in fig. 2a and fig. 2b and finally the consistency analysis is presented in fig. 3b. Note that the consistency figures are presented in logarithmic scale. Furthermore, the average NEES and NIS is approximately 18.57 and 2.27 respectively.

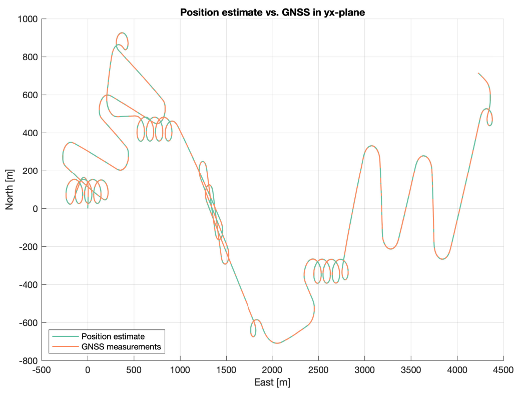


(a) UAV ESKF states

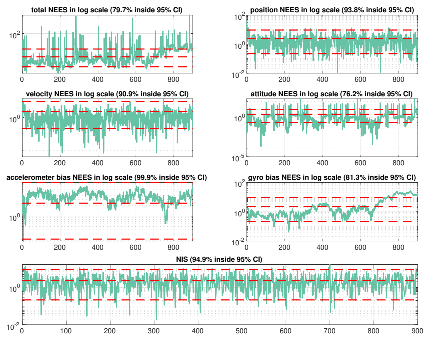


(b) UAV ESKF state errors

The ESKF was in particular tuned such that the bias states converged nicely, such that the errors propagated in the system from sensor drift were minimised. It should be emphasised that one of the main features that makes the ESKF robust is the IMU bias estimation, and it was therefore also emphasised in the tuning process.



(a) Estimated UAV trajectory



(b) UAV ESKF consistency analysis

The consistency was also emphasised when tuning the ESKF. We see from fig. 3b that most of the different normalized square errors are all in the 80% or 90% range inside the calculated confidence intervals, which was found to be satisfactory. One should especially note that the attitude NEES is only about 76% inside the confidence interval. Studying the heading error in fig. 2b can give some insight into why this might be the lowest NEES. It is observed that the heading is at certain time intervals far off the ground truth, which is also confirmed by the RMSE of 0.111. This is about two magnitudes higher error than for pitch and roll. This is caused by the fact that pitch and roll are directly observable from the gravitational force acting on the accelerometer. Heading, on the other hand, is only observable when the system is exited by a manoeuvre. The heading error, therefore, has certain sections where it is unobservable and the update step is not able to correct the constant offset from the true value. This is further supported by studying the time intervals when the UAV is doing a spiral manoeuvre e.g. from about 300 to 400 seconds or from 600 to 700 seconds. In these intervals the heading is observable and as a consequence, we observe that the error drops to the same magnitude as the pitch and roll, before it starts acting unexpectedly again when the UAV starts flying straight. This issue might be why both the attitude NEES and total NEES is somewhat worse than the others. So the heading estimation would therefore not work if the UAV was standing still.

When letting the sensor correction matrices be identity matrices, and thereby removing any sensor correction, the ESKF estimation performance is noticeably worse. While the filter is still able to reliably estimative velocity and position with the aid of the GNSS, the estimates of heading and sensor biases are much worse. Most importantly, the rate gyro and accelerometer biases no longer converge to the true sensor biases. This means we are not able to remove the bias in the measurements, and as a consequence, they are integrated and propagated through the Kalman filter. This makes all the RMSEs significantly worse, and especially the heading now drifts quickly when the UAV is not turning.

Then the NEES and NIS naturally also worsens significantly. While the position and velocity NEES is still somewhat within the confidence intervals at 85.2% and 64.9% respectively, the attitude, accelerometer bias and gyro bias is at 3.09%, 14.6% and 2.89% respectively. This further supports the claim above about how important IMU bias compensation is when trying to design a robust attitude estimator. It is then evident that correction for sensor misalignment, scaling errors and orthogonality errors are a critical part of designing such an estimator. Neglecting the GNSS lever arm, on the other hand, has no noticeable effect on the estimator performance.

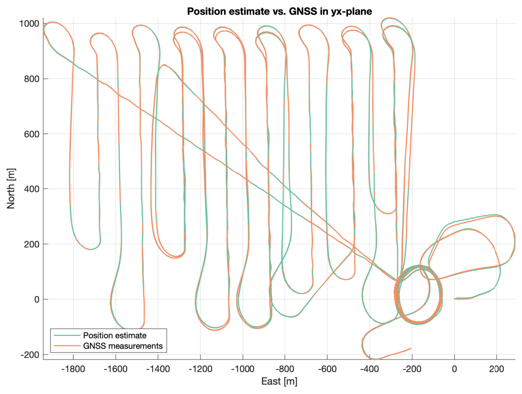
Finally, it must be said that finding the sensor correction matrices is not trivial, and the results will be empirically based and not absolutely correct. Therefore we will always have some sensor misalignment and miss-scaling in a physical system, especially since these matrices will, in reality, be time-varying.

## 3.2 INS for real fixed-wing UAV

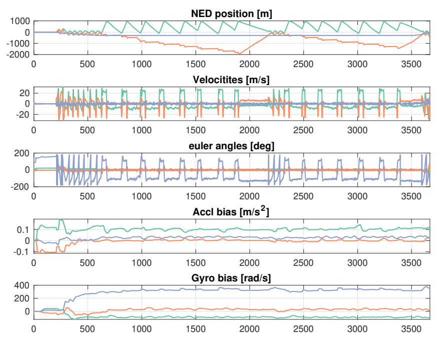
The ESKF was then tuned to a real dataset from a fixed-wing UAV flight. The sensors onboard was a STIM300 IMU and an u-blox M8 GNSS receiver. The GNSS receiver produces an accuracy estimate in standard deviation while operating. This was scaled by a gain parameter, squared and used as a time-varying GNSS measurement noise covariance signal. The gain was tuned to be $k_x = 0.5$.

The IMU measurement noise covariances and bias driving noise covariances were tuned from the Allan variances in the STIM300 datasheet [3]. Specifically, the gyro noise and bias noise root Allan variance is $0.15\,\mathrm{deg}/\sqrt{\mathrm{h}}$ and $0.5\,\mathrm{deg}/\mathrm{h}$ respectively. The accelerometer noise and bias noise root Allan variance is $0.06\,\mathrm{m/s}\sqrt{\mathrm{h}}$ and $0.05\,\mathrm{mg}$ respectively. These values were scaled to the correct units and used as initial parameters in the ESKF. The values were then tuned such that a satisfactory estimate and NIS were produced. The final tuning parameters were then $q_a = (9.8 \times 10^{-4})^2$, $q_{ab} = (4.9 \times 10^{-4})^2$, $q_\omega = (4.9 \times 10^{-6})^2$ and $q_{\omega b} = (2.4 \times 10^{-6})^2$. When comparing to the simulated case, one sees that the bias noise standard deviations are of about the same magnitude, while the measurement noise standard deviations are about two magnitudes larger in the simulated case.

The resulting UAV trajectory estimate is presented in fig. 4a and the estimated states are presented in fig. 4b. Finally the NIS is presented in fig. 5. The average NIS was approximately 3.01. Note that since this is a real dataset, we do no longer have the ground truth, and can therefore not calculate the estimation errors and NEES. This certainly makes tuning more challenging, even though access to the IMU datasheet is useful. This shows how it is often helpful to test filter implementations on simulated data first, as has been done in all of the assignments discussed in this report. The Gauss-Markov bias time constants were both tuned to $T_b = 1 \times 10^{16}\,\mathrm{s}$, effectively making the bias processes random walks processes.



(a) Estimated UAV trajectory



(b) UAV ESKF states

We observe how the estimated trajectory fits nicely with the GNSS data as one would expect. The estimates are also nicely behaved, while the state trajectories are naturally

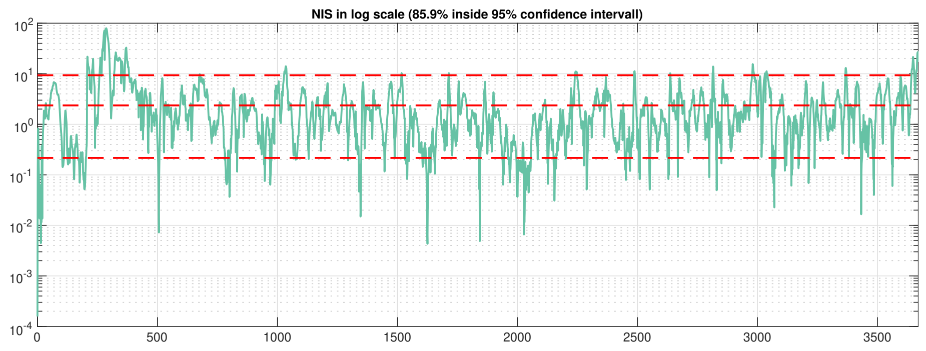


Figure 5: UAV GNSS logarithmic NIS

a lot less smooth in this case compared to the simulation, which is to be expected. In particular, the biases converge over time, with some corrections throughout the flight when the UAV turns and the system is excited. From the heading plot it seems like the bias estimates work well, as there is no noticeable drift in heading. For most of the flight, we see that the heading changes between two almost constant values. This is reflected by the UAV manoeuvres in fig. 4a, where the UAV flies straight north, then quickly does a U-turn, then flies straight south, then does a U-turn and so on.

When the sensor correction matrices were set to identity for the real data, it was observed again that thanks to the GNSS measurements the position and velocity estimates are still viable. But the biases do not converge to the same values as before. Interestingly, the heading quickly becomes the opposite of what it was before, and the roll and pitch estimates "switch places", as can be seen in fig. 6. This is due to the fact that the sensor frame is no longer approximately aligned with the body frame. While in the simulated case the sensor correction matrices were close to identity, for this data set they are

$$S_g = \begin{bmatrix} -0.0243 & 0.9981 & 0.0454 \\ -0.9998 & -0.0242 & -0.0255 \\ -0.0272 & -0.0532 & 0.9949 \end{bmatrix}, \quad S_a = \begin{bmatrix} -0.0503 & 1.0021 & 0.0476 \\ -1.0243 & -0.0695 & -0.0197 \\ -0.0387 & -0.0483 & 1.0005 \end{bmatrix}. \tag{3}$$

Now notice that $S_g \approx R_z(-\frac{\pi}{2}), S_a \approx R_z(-\frac{\pi}{2})$. This means that in addition to small scaling, misalignment and orthogonality errors there is a 90° rotation that is needed in order to convert the measurements from sensor to body frame. When implementing an ESKF, and even more generally some sensor fusion algorithm in a physical application such as this, one obviously needs to consider that very rearily will sensors be mounted along the body axes, and a sensor to body transformation is therefore absolutely necessary for the system to work. Luckily, it is easy to see from the estimates or even the measurements themselves that they do not correspond to what common sense it telling you e.g. in this case that the heading estimate points southwards when the plane is flying north and vice-versa.

One could argue that the benefits of the ESKF are twofold. Firstly, since the error dynamics are approximately linear, the optimality of the linear Kalman filter, which is lost when extending it to nonlinear process and measurement models with the EKF, are (approximately) retained. Secondly, since the IMU is treated as a control input and the IMU biases are estimated, we achieve robustness, as has previously been discussed. In the test implementations in this assignment, with both simulated and real data sets, this has been observed. By masking modelling errors, sensor errors, uncertainties and actual sensor biases within these states, the filter system is able to operate in a dynamic environment

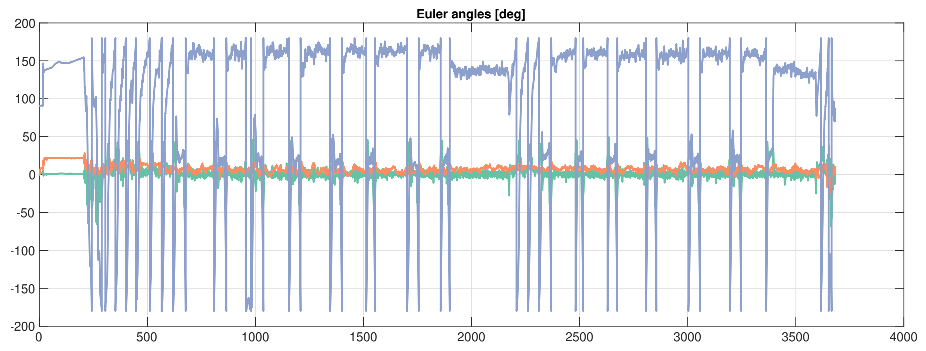


Figure 6: UAV estimated attitude without sensor error correction

where the plant performs various different manoeuvres, even with wind disturbance and temperature fluctuations.

One could argue that the main limitation of the ESKF for inertial navigation is its slow convergence for deficient initial attitude values. In these tests of the filter, this has not been an issue, as good initial values were provided. One should note that this is not always the case. In many real-time applications, you cannot expect to turn on your INS and have good initial estimates of the attitude. If one tries to give an arbitrary attitude initialization to the ESKF, one observes poor estimates that are not at all usable. Due to length limitations this cannot be further explored here, but nevertheless this is a problem that could be further analysed and extensions to the ESKF could be added to handle this e.g. with optimization methods.

## 4 Graded Assignment 3

## 5 Conclusion

## References

[1] E. Brekke. *Fundamentals of Sensor Fusion*. 2019.

[2] J. Solà. *Quaternion kinematics for the error-state KF*. `http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf`. 2017.

[3] *STIM300 Inertia Measurement Unit*. Rev. 9. Sensonor. May 2013.