Contents

1 Means and Covariance Matrices

In this laboratory experience we were asked to perform four initial experiments in order to model the drone sensors and to identify the noise present in all the measurements. All the experiments were performed and the data were saved in a file called `navdata.mat`.

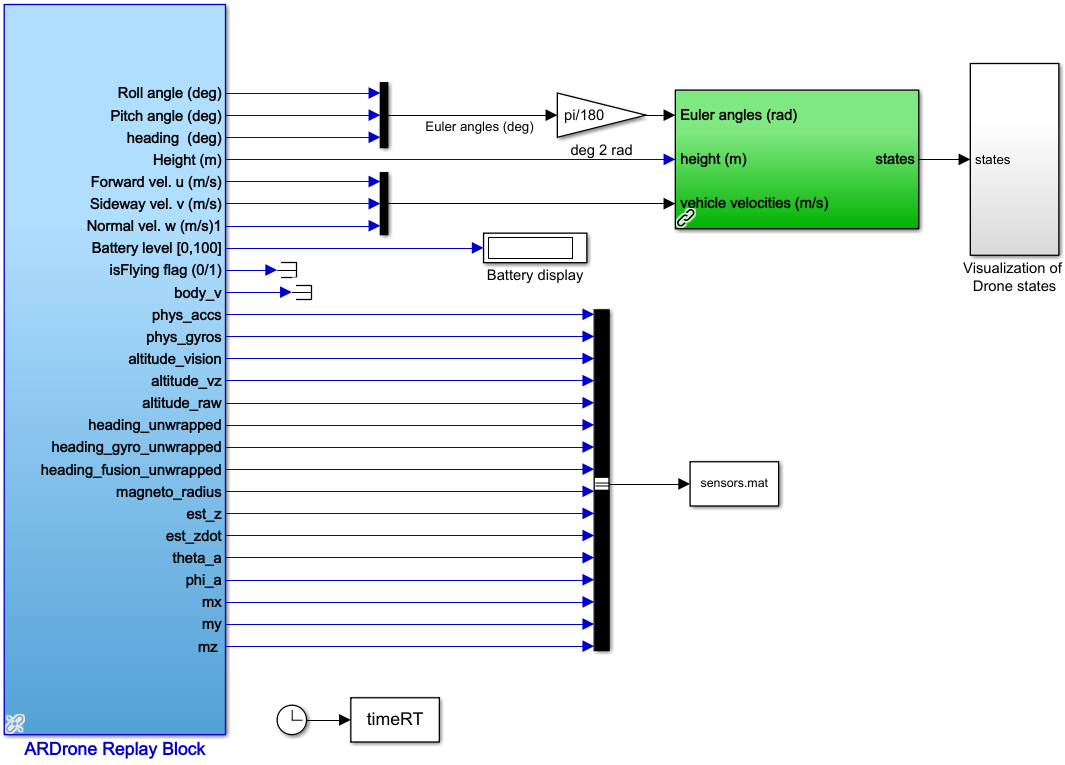
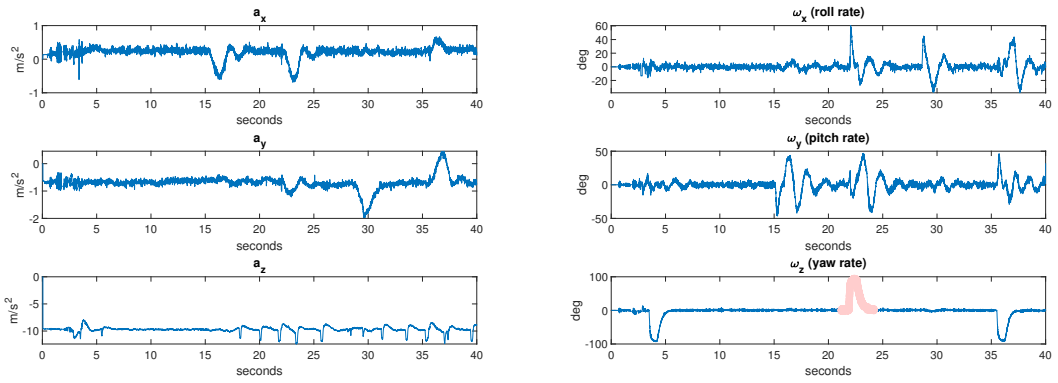


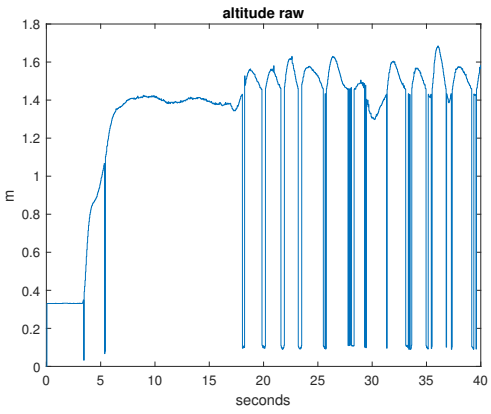
Figure 1: Modified Replay Model

Modifying the Replay script provided, as shown in Fig(1), we could collect all the raw data from the sensors decoded from the file `navdata`. Among all the available outputs we used the ones called `phys_gyro`, `phys_acc` and the `raw_altitude`. Having a first plot of these outputs we realised that both gyro and accelerometer were composed by three signals each: clearly the angular rate and the acceleration along the three main body reference system axes.
In order to identify which signal corresponds to which axis we had to compare data from different experiments: if the acceleration along z was clearly recognizable from the very first experiment, the identification of all the other data was possible only looking at Experiment D, where we had a clear matrix showing the imposed attitude and the drone movement sequence.
Another problem that we had to face was to identify the scale of the output: for the accelerometer

was easy to understand that all the signal should have been divided by one hundred, simply noting that the acceleration along z gave values around 981 instead of the expected 9.81. For the height signal we had to analyse the Experiment C in which the drone was supposed to hover at one meter: it turned out that we needed to scale the signal of a 10^{42} factor. It was a bit more complicated to understand the scale of the angular rate: knowing that the yaw variations in Experiment D were set at 90° each, we selected in the yaw rate signal one of them (i.e. the highlighted one in Fig(2)), integrating the signal, expecting a value of 90° with some possible scale factor. Luckily the integration result was exactly 90°, so for the `phys_gyro` signal we did not have to make any change.
In Fig(2) the plot of the sensors measurements for the Experiment D is shown.



(a) (b)



(c)

Figure 2: Experiment D Measurements

1.1 Experiment A

In the first experiment the drone was left on the floor without any input from the rotor. During a 30 second period datas from all the sensors on board were collected. The aim of this experiment was to have a first sight at the bias of our sensors at rest. We can recognise the gravity acceleration detected as the a_z measurement. As all the other measurements, it is not perfectly what we expected since, as we have already said, all the sensors have biases. The covariance matrix for the first experiment does not show considerably high value since the variation of the collected data are not connected to each other. Being the drone at rest on the floor the values present in this matrix are mainly due to the disturbances.

Means:

	a_x	a_y	a_z	ω_x	ω_y	ω_z	h
Mean	-0,103	-0,436	-9,92	0,0359	-0,0246	0,0131	0,332

Covariance Matrix:

	a_x	a_y	a_z	ω_x	ω_y	ω_z	h
a_x	0,000171	1,66e-05	0,00176	-2,56e-6	2,18e-06	-7,79e-07	-5,74e-05
a_y	1,66e-05	0,00160	0,00710	-7,49e-05	3,57e-05	-3,40e-05	-0,000241
a_z	0,00176	0,00710	0,165	-0,000617	0,000434	-0,000226	-0,00550
ω_x	-2,56e-06	-7,49e-05	-0,000617	0,00187	1,99e-06	-3,34e-05	1,98e-05
ω_y	2,18e-06	3,57e-05	0,000434	1,99e-06	0,00168	5,83e-05	-1,26e-05
ω_z	-7,79e-07	-3,40e-05	-0,000226	-3,34e-05	5,83e-05	0,00277	8,14e-06
h	-5,74e05	-0,000241	-0,00550	1,98e-05	-1,26e-05	8,14e-06	0,000185

1.2 Experiment B

The Experiment B was performed sending a take off input to the drone, but keeping it manually stuck at ground. In this case the sensor measurements are way more imprecise since all the environment surrounding the drone is full of disturbance due to the motion of the blades and to the close interaction between the floor and the air flow. For what does concern the measurement mean, more or less we can read the same order of magnitude of the A Experiment. The values of the covariance matrix are higher than in the Experiment A, because here it starts to appear some correlations between the sensors measurement (i.e. we notice high values between rate gyro measurements).

Means:

	a_x	a_y	a_z	ω_x	ω_y	ω_z	h
Mean	0,136	-0,560	-9,86	0,0165	-0,301	0,117	0,331

Covariance Matrix:

	a_x	a_y	a_z	ω_x	ω_y	ω_z	h
a_x	0,00497	0,00134	-0,00441	0,0201	-0,0418	-0,00791	4,178e-05
a_y	0,00134	0,00749	0,00975	0,0363	-0,0406	-0,0548	-0,000469
a_z	-0,00441	0,00975	0,215	-0,0198	0,0412	0,0167	-0,00702
ω_x	0,0201	0,0363	-0,0198	2,75	-0,564	-0,371	-0,000247
ω_y	-0,0418	-0,040589	0,0412	-0,564	4,87	1,74	-0,000244
ω_z	-0,00791	-0,0548	0,0167	-0,371	1,74	5,33	-0,000409
h	4,18e-05	-0,000469	-0,00702	-0,000247	-0,000245	-0,000409	0,000983

1.3 Experiment C

The last experiment required to collect the measurement during a certain time period of hovering. In order to compute mean and covariance of the drone sensors we decided to analyse data after the take off, just when the drone was already at one meter height.
We noticed that data from height sensor were in some way discontinuous: they suddenly jump from the expected one meter measurement to just some decimals for a very short period of time (i.e. from 1.0701 to 0.0703 for some Δt). Since this jump was apparently a conversion or a transmission error of missing or exchanging 1 to 0 we decided to correct all the measurement close to zero adding the missing number (i.e. we selected 0.0703 and convert it to 1.0703). This procedure is obviously applicable only in case of hovering, neglecting the transient take off phase, otherwise we could not identify, between all the height data, which were the mistake and which are the real recordings at the very beginning of the experiment. Moreover this correction can just be implemented after collecting the height measurement, not during the effective experiment in real time, which will be always affected by this imprecision.
The means of the measurement show a general decrease in magnitude except from a_y (case that can derive from some particular disturbance). The covariance matrix shows the same order of magnitude of the Experiment B.

Means:

	a_x	a_y	a_z	ω_x	ω_y	ω_z	h
Mean	-0,0857	-0,539	-9,87	-0,00550	-0,0504	-0,0496	1,058

Covariance Matrix:

	a_x	a_y	a_z	ω_x	ω_y	ω_z	h
a_x	0,00252	-0,000241	-0,000724	0,00355	-0,00950	-0,00329	7,29e-06
a_y	-0,000241	0,00469	-0,000411	0,0248	-0,0172	-0,0196	-0,000103
a_z	-0,000724	-0,000410	0,00657	-0,00437	0,0299	0,0182	9,95e-05
ω_x	0,00355	0,0248	-0,00437	4,28	-0,298	-0,512	0,000701
ω_y	-0,00950	-0,0172	0,0299	-0,298	4,94	0,886	-0,000269
ω_z	-0,00329	-0,0196	0,0182	-0,512	0,886	2,79	-0,000412
h	7,29e-06	-0,000103	9,95e-05	0,000701	-0,000269	-0,000412	0,000112

1.4 Pitch & Roll Estimation

Recalling the Rotation Matrix, seen in the first laboratory, between Inertial Reference Frame and Body Reference Frame, we can easily connect the gravity acceleration seen in IRF to the same vector seen in BRF.

$$\boldsymbol{a}_g^B = \begin{bmatrix} \dots & \dots & -\sin\theta \\ \dots & \dots & \cos\theta\sin\phi \\ \dots & \dots & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix} \tag{1}$$

with $g = 9.81\ m/s^2$.

Supposing we know these three values we could compute exactly the value of pitch and roll angle. Unfortunately the available measurement that we can use to compute those angles are only the one comes from the linear accelerometer. The accelerometer model provide us:

$$\boldsymbol{a}_{acc}^B = -\begin{bmatrix} \dot{u} - ru + qw \\ \dot{v} + ru - pw \\ \dot{w} - qu + pv \end{bmatrix} - \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix} \tag{2}$$

Where the minus signs are necessary since the accelerometer outputs are the inertial accelerations ($a_{z,acc}$ measurement is -9.81 when θ and ϕ are zero).

The only way to extract the second vector from our acceleration measurement is to neglect the first part of the acceleration, supposing the system to be always in a steady state. This is obviously a big approximation that in general is false for every working condition. However for flight condition featuring small linear accelerations and small angular velocities this procedure can be accepted. We can also think that both pitch and roll are calculated from accelerometer in order to be used as measurement inside a Kalman filter: this will assure us that even though they are not extremely close to the reality, they will be improved using the prediction based on the system model implemented inside the filter, model that is not bound with the part of the acceleration that we are neglecting here.

Going through the formulas we have:

$$\boldsymbol{a}_{acc}^B = -\begin{bmatrix} \cancel{\dot{u} - ru + qw} \\ \cancel{\dot{v} + ru - pw} \\ \cancel{\dot{w} - qu + pv} \end{bmatrix} - \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix} \tag{3}$$

$$\boldsymbol{a}_g^B \simeq -\boldsymbol{a}_{acc}^B \tag{4}$$

Solving this simple equations we can get the desired variables

$$\begin{cases} \phi = \arctan(\frac{a_y}{a_z}) \\ \theta = \arctan(\frac{-a_x cos(\phi)}{a_z}) \end{cases} \tag{5}$$

2 Kalman Filters

Kalman filters are based on a prediction of the possible state at the step $k+1$ merged together with measurements coming from sensors. It is important to underline that every Kalman filter needs a mathematical model of the variable(s) that is going to predict in order to operate properly. This kind of filter is very useful in the control techniques and is mainly used for state estimation reducing noise and providing a mean to get unknown parameters present in the system (such as biases).

2.1 Design

First of all we had to choose our prediction equation for θ, ϕ and h:

$$\theta_{k+1} = \theta_k + \Delta t(\omega_{y,k}) \tag{6}$$

$$\phi_{k+1} = \phi_k + \Delta t(\omega_{x,k}) \tag{7}$$

$$h_{k+1} = h_k + \frac{1}{2}\Delta t^2(a_z) \tag{8}$$

The values for the angular rates ω_x and ω_y are given to the model from the rate gyro sensor, as well as a_z. For this last estimation it should be highlighted that the value of a_z according to our prediction model should be $a_z = -(a_{z,acc} + g)$. In this way a positive value of a_z will imply a positive variation of height.
The second step is the fusion of this result with the sensors data through an optimal gain calculated for every time step:

$$\theta_{k+1} = \theta_{k+1} + K(\theta^M_{k+1} - \theta_{k+1}) \tag{9}$$

$$\phi_{k+1} = \phi_{k+1} + K(\phi^M_{k+1} - \phi_{k+1}) \tag{10}$$

$$h_{k+1} = h_{k+1} + K(h^M_{k+1} - h_{k+1}) \tag{11}$$

The complete general core loop is reported below:

```
for k=1:length(input)-1
|  states_pred=Ad*states(:,k)+Bd*input(:, k);
|  P_pred=Ad*P*Ad'+Q;
|  K = P_pred*H'*inv(H*P_pred*H'+R);
|  states(:,k+1)=states_pred+K*(y(:, k+1)-H*states_pred);
|  P=P_pred-K*H*P_pred;
end
```

This loop can be expanded to how many states we want being able to provide the $\boldsymbol{A_d}$, $\boldsymbol{B_d}$ and the $\boldsymbol{H}$ matrix of our system:

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{A_d}\boldsymbol{x}_k + \boldsymbol{B_d}\boldsymbol{u}_k \\ \boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{x}_k \end{cases} \tag{12}$$

where $\boldsymbol{y}$ are the sensors measurements.

As an example we report here the loop used for the pitch Kalman filter without bias estimation:

```
for k=1:length(om_y)-1
|  theta_pred(k+1)=theta_filtered(k)+dt*om_y(k);
|  P_pred(k+1)=P(k)+Q;
|  K=P_pred(k+1)*H'*inv(H*P_pred(k+1)*H'+R);
|  theta_filtered(k+1)=theta_pred(k+1)+K*(theta_meas(k+1)-H*theta_pred(k+1));
|  P=P_pred-K*H*P_pred;
end
```

2.2 Transfer Functions

In order to analytically get the transfer functions from the measurements to the filtered state and from the input to the filtered state we have to consider the following system:

$$\begin{cases} \boldsymbol{x}^P = \boldsymbol{A_d}\boldsymbol{x}_k^F + \boldsymbol{B_d}\boldsymbol{u}_k \\ \boldsymbol{x}_{k+1}^F = \boldsymbol{x}^P + \boldsymbol{K}(\boldsymbol{y}_{k+1}^M - \boldsymbol{H}\boldsymbol{x}^P) \\ \boldsymbol{K} = constant \end{cases} \tag{13}$$

combining the first and the second equation we get:

$$\boldsymbol{x}_{k+1}^F = \boldsymbol{A_d}\boldsymbol{x}_k^F + \boldsymbol{B_d}\boldsymbol{u}_k + \boldsymbol{K}(\boldsymbol{y}_{k+1}^M - \boldsymbol{H}\boldsymbol{A_d}\boldsymbol{x}_k^F - \boldsymbol{H}\boldsymbol{B_d}\boldsymbol{u}_k) \tag{14}$$

$$(z\boldsymbol{I} - \boldsymbol{A_d} + \boldsymbol{K}\boldsymbol{H}\boldsymbol{A_d})\boldsymbol{x}^F = z\boldsymbol{K}\boldsymbol{y}^M + (\boldsymbol{B_d} - \boldsymbol{K}\boldsymbol{H}\boldsymbol{B_d})\boldsymbol{u} \tag{15}$$

This allow us to write the transfer functions required as:

$$\frac{\boldsymbol{x}^F}{\boldsymbol{y}^M} = (z\boldsymbol{I} - \boldsymbol{A_d} + \boldsymbol{K}\boldsymbol{H}\boldsymbol{A_d})^{-1}z\boldsymbol{K} \tag{16}$$

$$\frac{\boldsymbol{x}^F}{\boldsymbol{u}} = (z\boldsymbol{I} - \boldsymbol{A_d} + \boldsymbol{K}\boldsymbol{H}\boldsymbol{A_d})^{-1}(\boldsymbol{B_d} - \boldsymbol{K}\boldsymbol{H}\boldsymbol{B_d}) \tag{17}$$

This expressions declined to our three systems give us the six transfer functions we were looking for:

$$\frac{\theta^F}{\theta^M} = \frac{Kz}{z-1+K} \qquad \frac{\theta^F}{\omega_y} = \frac{\Delta t(1-K)}{z-1+K}$$
$$\frac{\phi^F}{\phi^M} = \frac{Kz}{z-1+K} \qquad \frac{\phi^F}{\omega_x} = \frac{\Delta t(1-K)}{z-1+K}$$
$$\frac{h^F}{h^M} = \frac{Kz}{z-1+K} \qquad \frac{h^F}{a_z} = \frac{\Delta t^2(1-K)}{2(z-1+K)}$$

2.3 Tuning Q and R

The **Q** matrix describe the uncertainty on the state estimation, so it is mainly influenced from the accuracy of the model and of the input and also from the timestep Δt. It can be convenient to use a simpler model for our variable evolution since the computation will be quicker, having on the other hand higher values for **Q** coefficients.
The **R** matrix describe the uncertainty on the sensors measurements, so the identification of its coefficient can be performed offline by different kind of experiments. It is common to take as **R** the covariance matrix of the measurements present inside the Kalman filter.
Both matrices in general can vary during an experiment according to the condition in which our system works: for simplicity we decided to take **Q** and **R** constant, set at the beginning of each simulation. Decreasing **Q** or increasing **R**, the filter will trust more the state estimation from prediction model than the measurements. On the other hand, decreasing **R** the filter will trust more measurements than the estimation from prediction model.
In all the experiments in which we have tested our Kalman filter script we kept a unitary value for **R** while a very small value for **Q**, around 0.0001. A different approach was used in the extended Kalman filter, where we used the values of the covariance matrix for **R** while keeping the same low value for **Q**, which in that case was a diagonal matrix. For the functions we used to close the loop of the Simulink block the values where $\boldsymbol{Q} = 0.01$, $\boldsymbol{R} = 5$.

2.4 Results from Simulation

We implemented Kalman Filters for θ, ϕ and h and run offline simulations with the measurements collected thanks the modifications applied to the replay script. The results of the estimation are shown in the following Figures (3),(4),(5),(6), where the estimated variable is compared to the measured one and to the one estimated by the Simulink model provided. After the Kalman filter has been properly tuned, the output is considerably less affected by the noise. The shape of the estimated states is really similar to the one estimated by the original model. We can just see that there are some offset errors that shift our prediction away from the yellow lines. This offset is probably mainly due to the bad measurements of accelerations and angular velocities that we used to estimate the state.

2.5 Bias Estimation

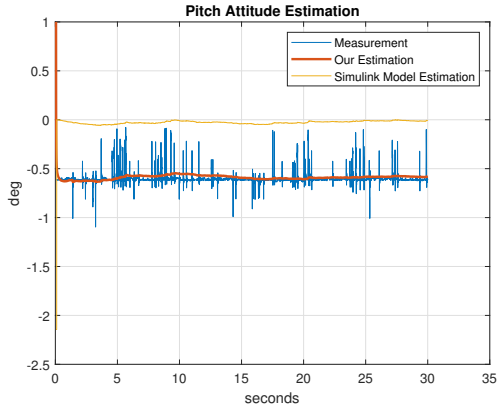
As already said before, Kalman filters are useful to estimate unknown parameters of the system: in this case we want to extract the information about the input bias. With reference to the previous general core loop with matrices $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$, it will be easy to insert the bias as a state of the system. The prediction model will be thus modifyed for pitch, roll and height:

$$\begin{bmatrix} \theta \\ b_y \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ b_y \end{bmatrix}_k + \begin{bmatrix} dt \\ 0 \end{bmatrix} \omega_{y,k} \tag{18}$$

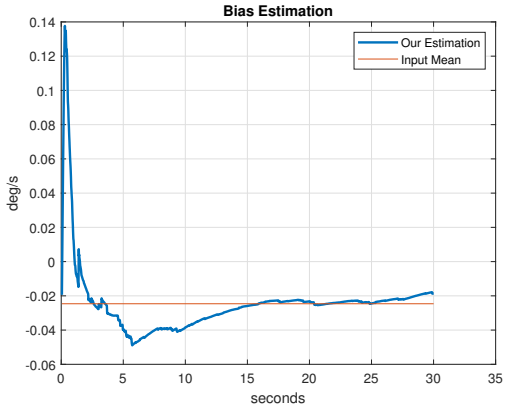
$$\begin{bmatrix} \phi \\ b_x \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \phi \\ b_x \end{bmatrix}_k + \begin{bmatrix} dt \\ 0 \end{bmatrix} \omega_{x,k} \tag{19}$$

$$\begin{bmatrix} w \\ h \\ b_h \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & -dt \\ 1 & dt & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w \\ h \\ b_h \end{bmatrix}_k + \begin{bmatrix} dt \\ 0 \\ 0 \end{bmatrix} a_{z,k} \tag{20}$$

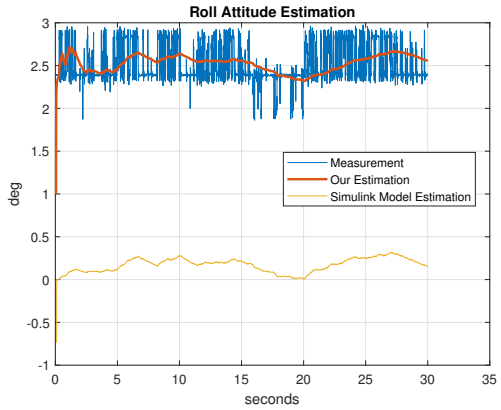
In the prediction model the bias is supposed to stay unchanged but thanks to the fusion at the end of the loop this variable will be refreshed every time step, converging much or less quickly depending on the $\mathbf{Q}$ and $\mathbf{R}$ tuning.



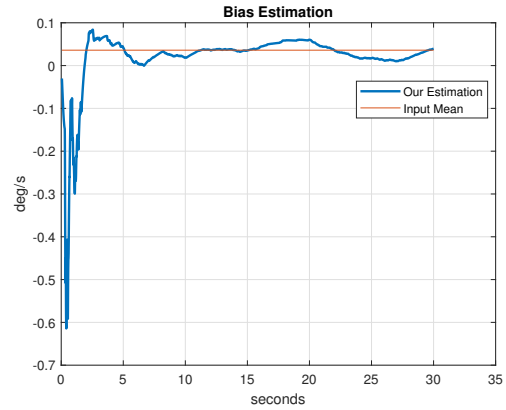
(a) θ



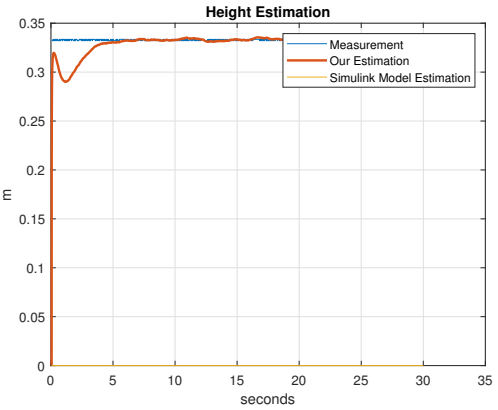
(b) ω_y bias



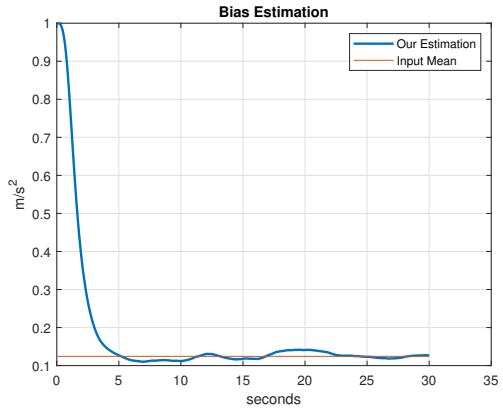
(c) ϕ



(d) ω_x bias

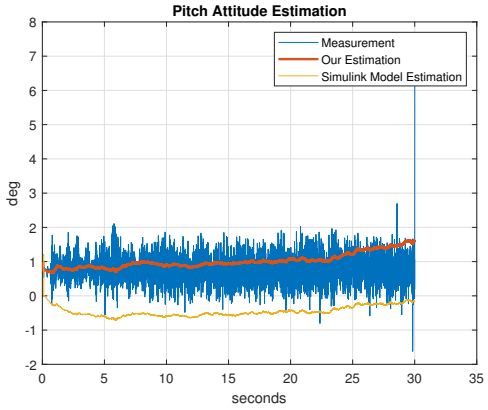


(e) h

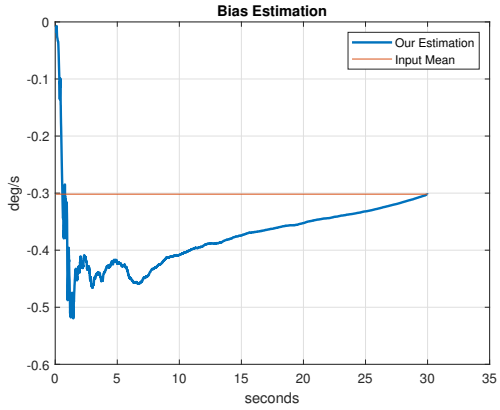


(f) a_z bias

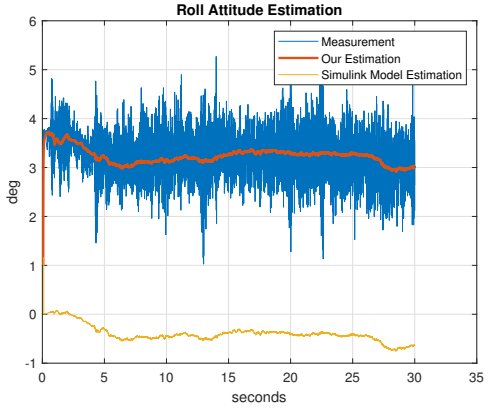
Figure 3: Experiment A



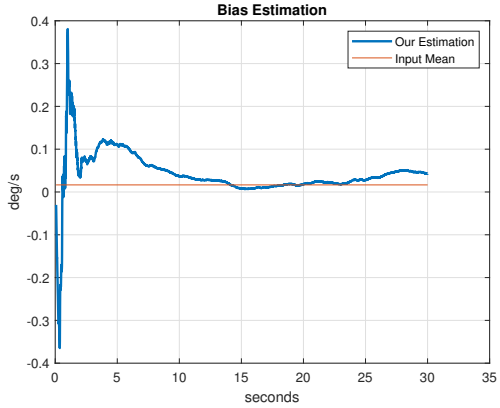
(a) θ



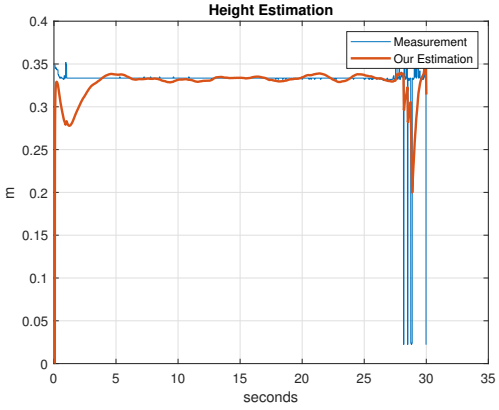
(b) ω_y bias



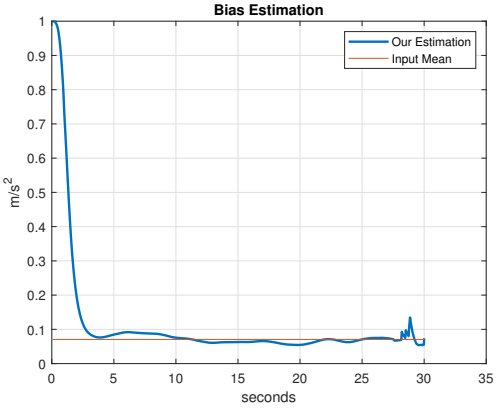
(c) ϕ



(d) ω_x bias

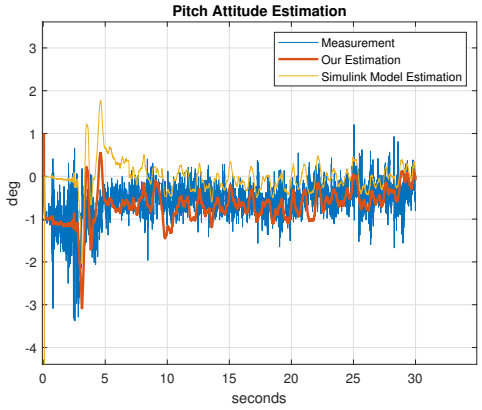


(e) h

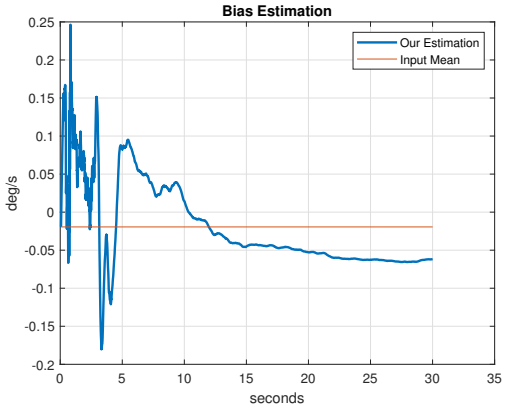


(f) a_z bias

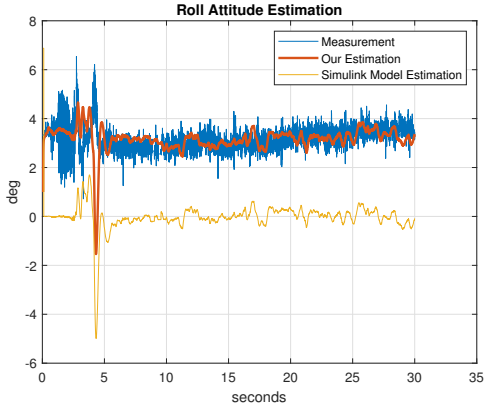
Figure 4: Experiment B



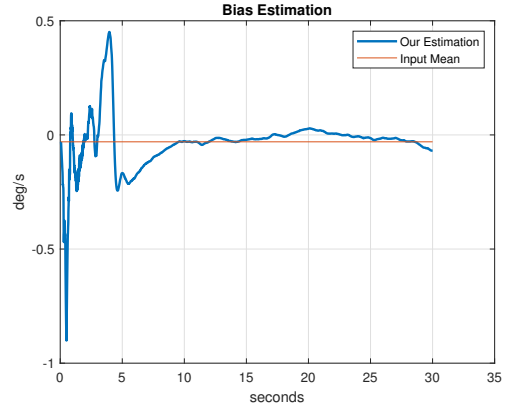
(a) θ



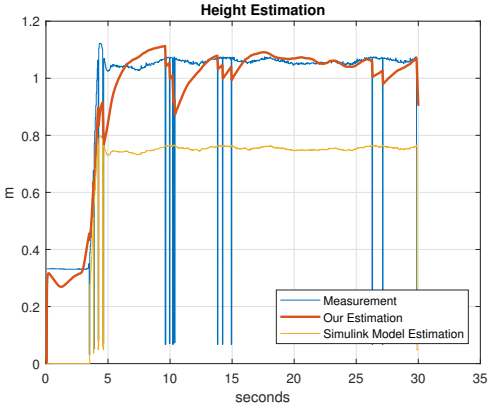
(b) ω_y bias



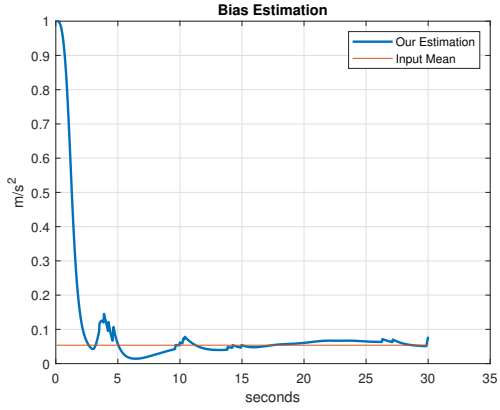
(c) ϕ



(d) ω_x bias

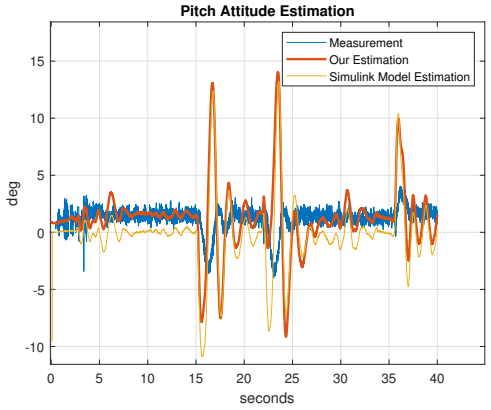


(e) h

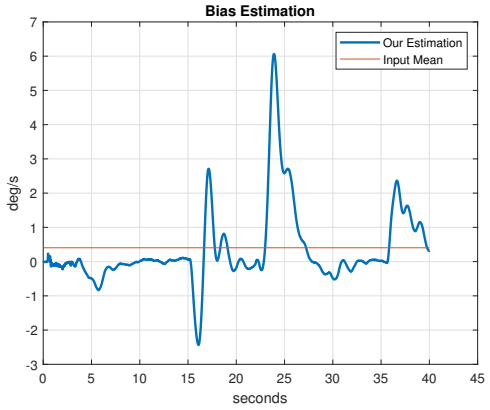


(f) a_z bias

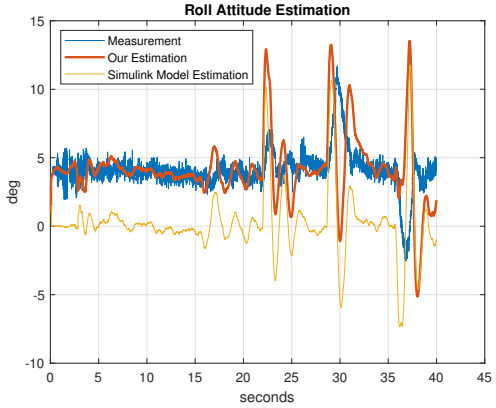
Figure 5: Experiment C



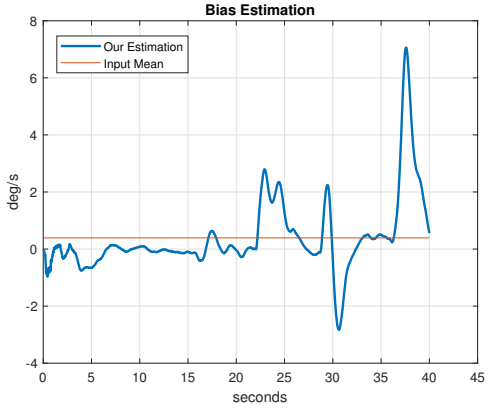
(a) θ



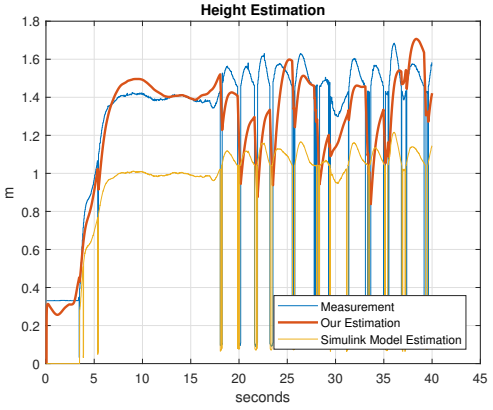
(b) ω_y bias



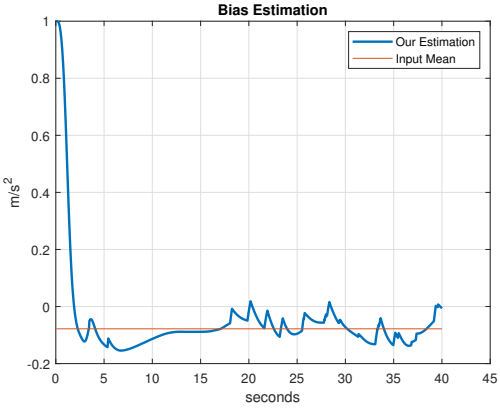
(c) ϕ



(d) ω_x bias



(e) h



(f) a_z bias

Figure 6: Experiment D

Remarks: in Experiment B the plot of the estimated height does not include the Simulink estimation because it had some extremely high values (around 10^6). This mistake in the Simulink state estimation is probably due to the fusion of various sensors measurement, including the sonar.
The bias estimation converges only if the state is steady: we can clearly see that in Experiment D, where we have several transient part, the bias estimation has many peaks and is not as precise as in the previous experiments.

2.6 Complementary Filters

Complementary filter fuses the integrated rate gyro with the accelerometer. We have the following equation:

$$\theta^F = \frac{Kz}{z-1+K}\theta_{acc} + \frac{\Delta t(1-K)}{z-1+K}\frac{z}{z-1}\dot{\theta}_{gyro} \tag{21}$$

If $K = 1$ then the two filters will sum to 1 and we have complimentary filter. The advantage of complementary filters is that the it is simpler because it involves less computations. The complementary filter will handle high frequency noise that effects θ_{acc}. And a high pass filter will take care of low frequent drifting in the θ_{gyro} measurement. The downside is that it is not as accurate as the Kalman filter.

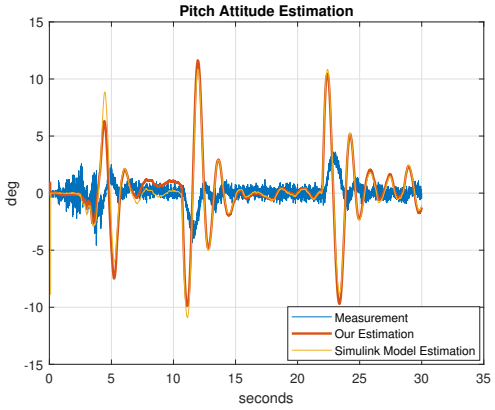
2.7 Additional Experiments (E)

In order to verify the action of the various Kalman filters we decided to perform three different additional experiments. Since the three filters were developed separately one to each other, any dependence of one measurement on another is neglected, so each one of these experiment has the purpose to excite only the estimated variable of the filter under investigation. Following this path we defined three 30 seconds missions as:

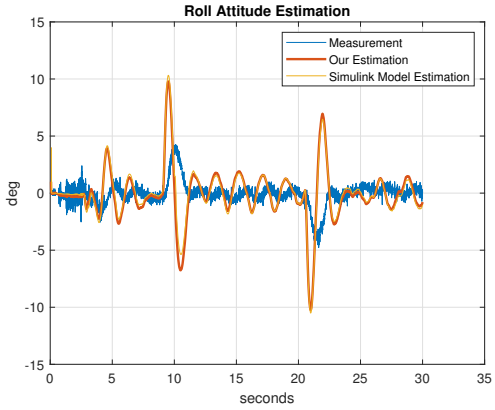
- **Pitch - E1**: take off at 1 meter at $P = (0, 0)$ position, go forward to $P = (1, 0)$ meter, come backward at $P = (0, 0)$.

- **Roll - E2**: take off at 1 meter at $P = (0, 0)$ position, go right at $P = (0, 1)$ meter, come back at $P = (0, 0)$.

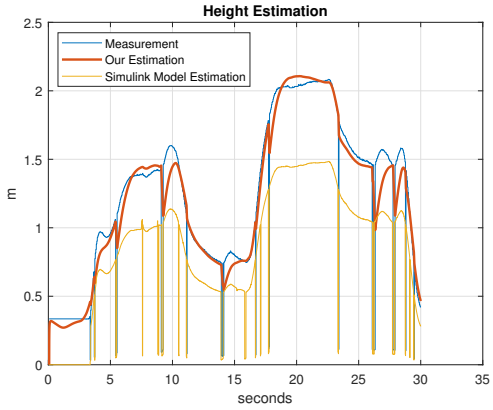
- **Height - E3**: take off at 1 meter, descend at 0.5 meter, climb at 1.5 meter, come back at 1 and end landing at 0.



(a) θ estimation in E1



(b) ϕ estimation in E2



(c) h estimation in E3

Figure 7: Experiments E

Using the same state estimation script we could obtain the results shown in Figures (7a, b, c). We notice that in this case there is no offset between our estimation and the Simulink estimation. Since the only thing that changes between A, B, C, D experiments and E experiments was the drone itself, we have here a confirmation that the offset present in the previous experiments was only due to the bad sensor measurements of the first drone we were given. In this case we have almost a perfect matching between the red and the yellow line for pitch and roll estimation. Height estimation remains a bit critical since we used just one measurement to filter our state: this seems to be not enough to estimate correctly the drone height.

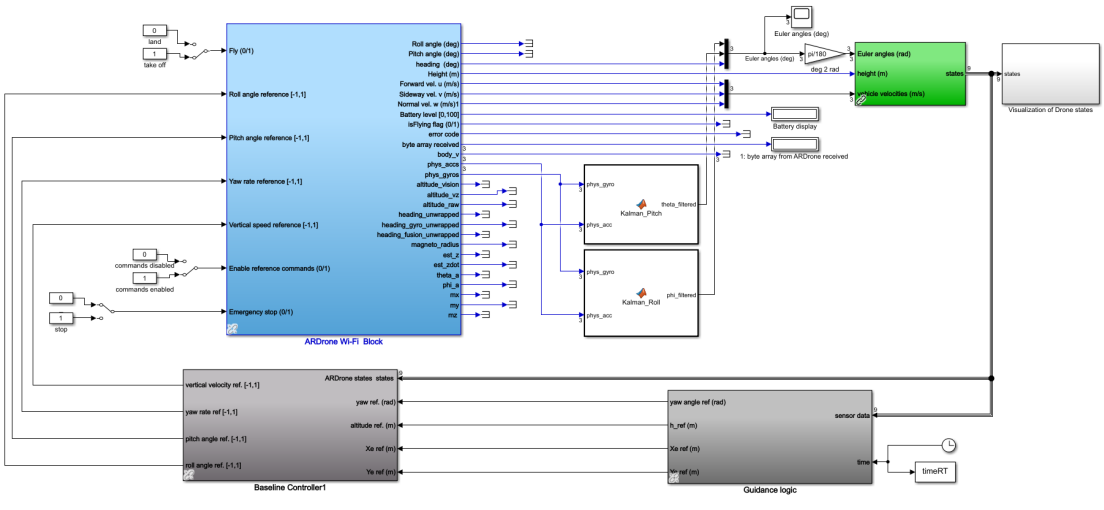


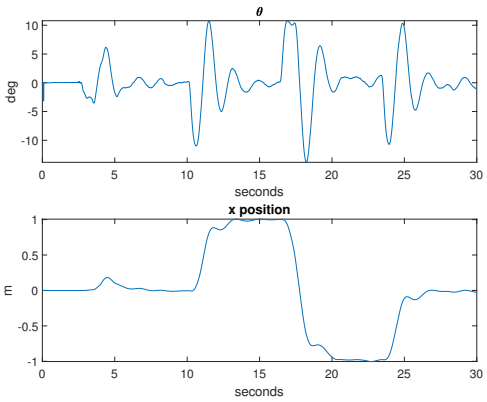
Figure 8: Modified Simulink Model

3 Closing the Loop with our Kalman Filter

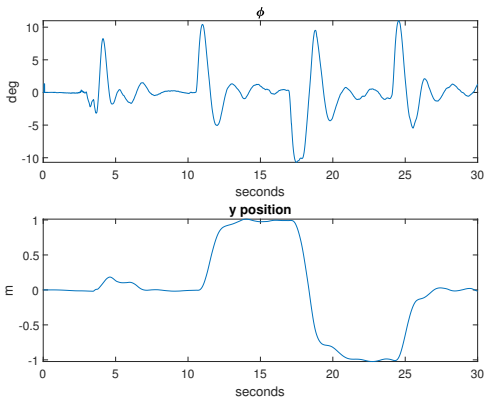
In order to practically test the implementation of our Kalman Filters we decided to try to convert our scripts for pitch and roll into two functions that could estimate that states in real time.

3.1 Real Time Implementation

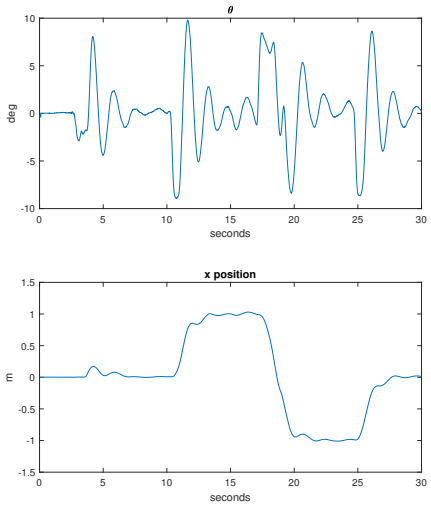
In order to face with real time streaming data we had to make several adjustment to the already existing script. First of all we converted it into a function with two input (the two measurements of accelerators and rate gyros) and one output (the filtered state).
We immediately realised that we had to drop the idea of having a loop inside the function, but we needed to keep a sort of memory of the variables calculated at $k-1$ step: for this aim we chose to use the `persistent` mode of Matlab. This mode keeps a certain parameter in memory even after a function has come to the end in such a way that at the next time step, when the function is called, it has the previous value.
We also had to include in the function the process of approximating the states through the accelerometer measurements: since we were working with equations (5) we also had to implement an `if` control to check whether a_z would be null. In that particular case equations (5) give a `NaN` result, being a zero on the denominator. Since this case only appeared at the very beginning of the experiment we set the state filtered to be zero if this case would have occurred.

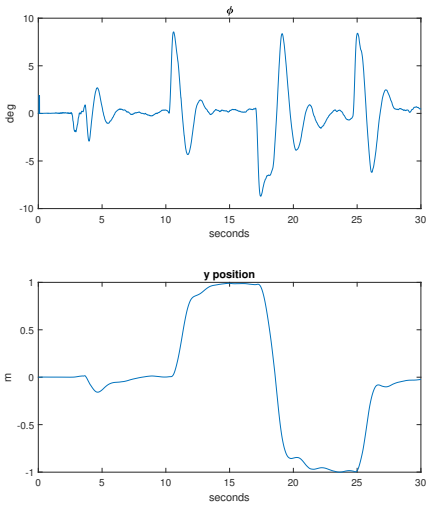


(a) Pitch



(b) Roll





(c) Pitch and Roll Combined

Figure 9: Drone States with Closed Loop using our Kalman Filter

In the laboratory, after fixing some errors in our initial functions, we succeeded in running the wi fi control script imposing some particular waypoints to the drone. We first tested the pitch Kalman filter (Fig(9a)) then tested the roll filter (Fig(9b)) and in the end we closed the loop with both filter imposing to the drone a diagonal trajectory (Fig(9c)). Results of estimation seems reasonable especially if we compare the first part of roll and pitch estimation in Figures (9a and b) with the first part of the experiments E1 and E2 in Figure (7a and b) where the input given to the drone is the same: we can see in both graph a first part due to the take off stabilisation and the first input to the angles. The shape of the first part of the curves, if compared, is very similar.

4 Extended Kalman Filter

The next step has been to try to implement an Extended Kalman filter for the pitch and roll attitude estimation.
We based our prediction model on the non linear equations (22):

$$\boldsymbol{f} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} p + r\cos\phi\tan\theta + q\sin\phi\tan\theta \\ q\cos\phi - r\sin\phi \end{bmatrix} \tag{22}$$

Here the $\boldsymbol{A_d}$ matrix for the refreshing of $\boldsymbol{P}$ was calculated at each step time as the Jacobian matrix of the non linear relation $\boldsymbol{f}$:

$$\boldsymbol{A_d} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^P} \tag{23}$$

The relation that binds the predicted state to the acceleration along the three axis is:

$$\boldsymbol{h} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = g \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \tag{24}$$

with $g = 9.81\, m/s^2$
On the other hand, as already underlined the accelerometer measurement vector is:

$$\boldsymbol{y} = -\begin{bmatrix} a_{x,acc} \\ a_{y,acc} \\ a_{z,acc} \end{bmatrix} \tag{25}$$

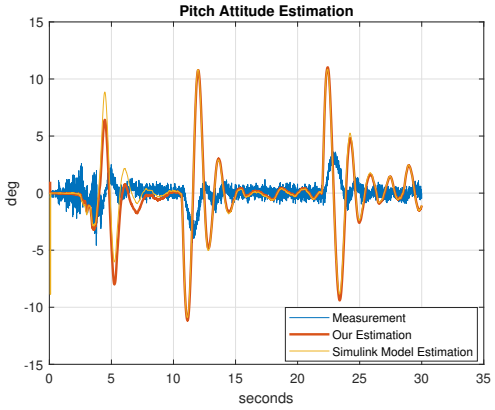
Here the $\boldsymbol{H}$ matrix for the refreshing of the Kalman gains $\boldsymbol{K}$ was calculated at each step time as the Jacobian matrix of the non linear relation $\boldsymbol{h}$:

$$\boldsymbol{H} = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^P} \tag{26}$$

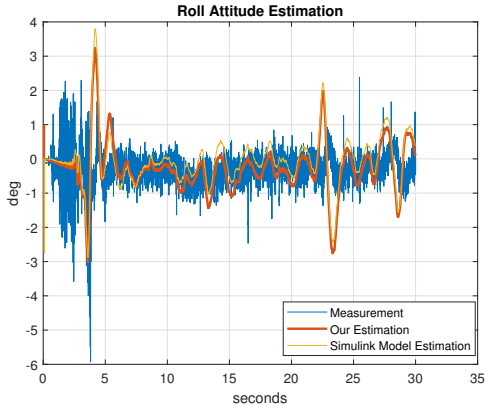
In this case the filtered state is calculated as following:

$$\boldsymbol{x}^F = \boldsymbol{x}^P + \boldsymbol{K}\left(\boldsymbol{y} - \boldsymbol{h}|_{\boldsymbol{x}=\boldsymbol{x}^P}\right) \tag{27}$$

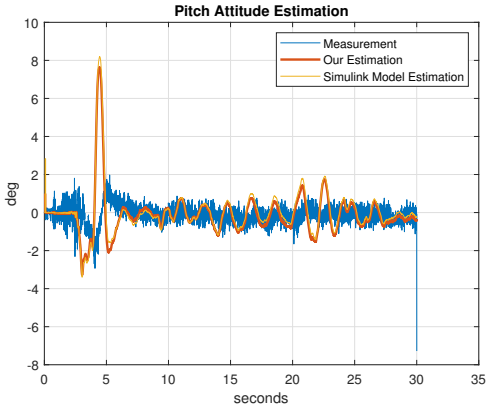
We can see that this kind of approximation is very similar to the Simulink block one and we can only see some difference in few areas of the graphs. This precision imply a bigger computational cost on the other side.



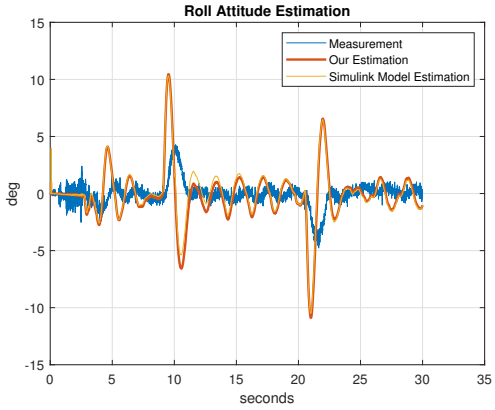
(a) Experiment E1



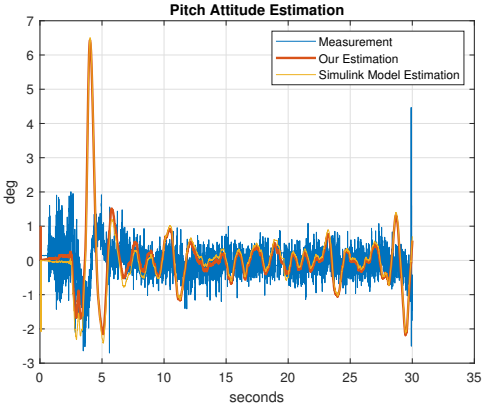
(b) Experiment E1



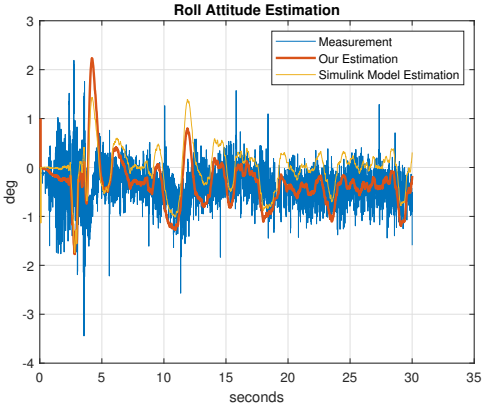
(c) Experiment E2



(d) Experiment E2



(e) Experiment E3



(f) Experiment E3

Figure 10: Extended Kalman Filter applied at Experiments E

5 Partial Attitude and Rate Gyro Bias Estimation

5.1 Implementation

In order to create a Kalman filter for the system (2) in the *"Partial Attitude and Rate Gyro Bias Estimation: Observability Analysis, Filter Design, and Performance Evaluation"* we had to create a function that could implement the creation of skew symmetric matrix, calling it at each step time in order to compose the $A_d(t)$ matrix of our system. The main core loop of our Kalman filter was created according to *A. Gelb, Applied Optimal Estimation (1974)* pag. 111.
We ran our Kalman script with data from the previous experiment.

5.2 Simulation

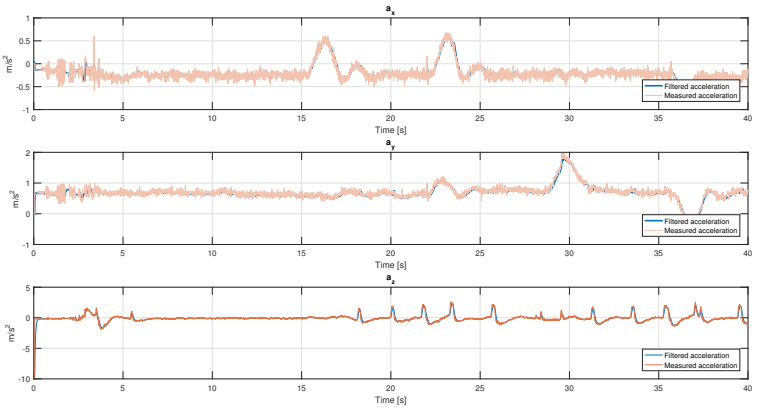


Figure 11: Filtered acceleration vs measured acceleration, Experiment D

We notice from Fig(11) that the filtered signal is less affected by the noise of the accelerometer. We also notice an offset from zero in the x- and y-axis. From the figure Fig(12) we see that the bias does not converge to any number. From Fig(13) we see the same smoothing of the noise, and also no offset from zero. From Fig(14) we see the value vary slightly around a constant value.

5.3 Results

The simulation of the implemented Kalman filter shows a smoothing result of the noisy measurement, even though still having some high frequent signals. The offset in experiment D is not present in experiment E1, and may be assumed as difference in the sensors, where the two experiments where done with different vehicles. The estimation of the bias did not quite converge to a constant value. With more exact tuning, and a longer simulation with a more consistent behaviour, one would possibly obtain a better result for the bias estimation.

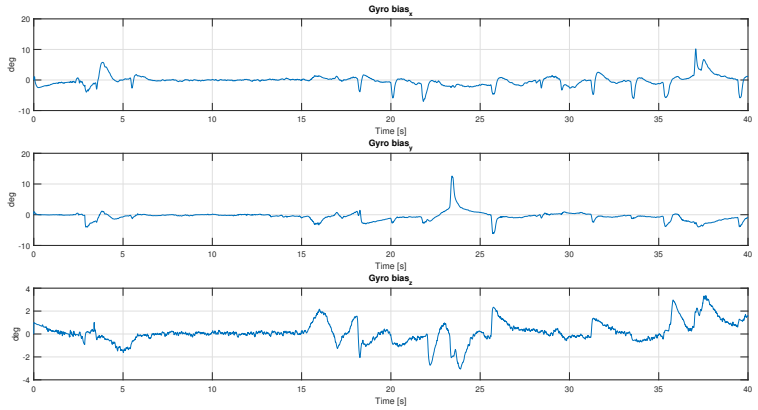


Figure 12: Estimated bias for the accelerometer, Experiment D

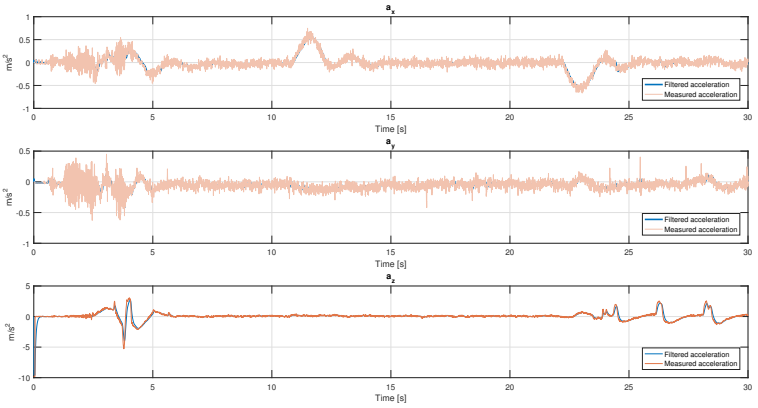


Figure 13: Filtered acceleration vs measured acceleration, Experiment E1

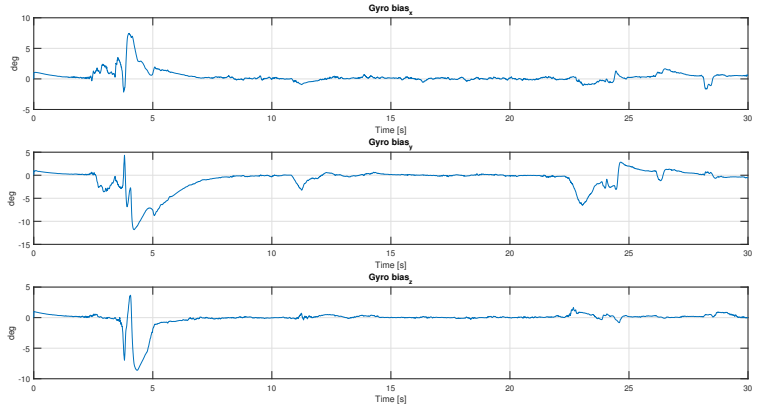


Figure 14: Estimated bias for the accelerometer, Experiment E1