TTK4115 Helicopter lab Report

Group 17
Karoline Nylænder 506889
Victoria Kenworthy 507766

February 2, 2021

Department of Engineering Cybernetics

Contents

1 Introduction

This report documents Helicopter lab in the course Linear System Theory at NTNU in the fall of 2020. The purpose of the lab is to experiment with pole placement, LQR, Luenberger observers and kalman filter. The report can be broken down into four main parts. The first part is mathematical modeling and PD control with pole placement. Next we will go into multivariable controllers through pole placement, LQR and LQR with integration. The third part entails the state estimation with IMU measurements and Luenberger observer. The final section consists of Kalman filter. Throughout the report we will associate the theory given in the course with our observations at the lab.

2 Part I - Monovariable control

To be able to control and analyze the behaviour of the helicopter we need to make a mathematical model of the system. The setup of the helicopter used in this project is illustrated below:

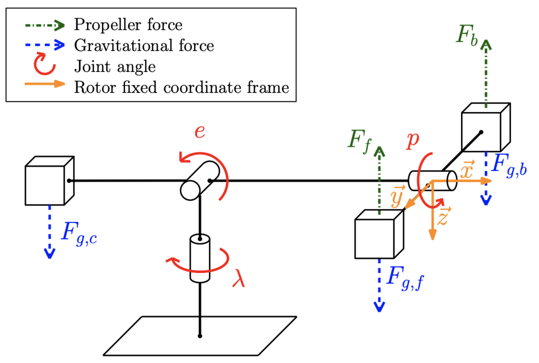


Figure 1: Model of system

From this setup we observe that there are three different joint angles we have to take into consideration when creating a model for controlling the system. These are pitch angle p, elevation angle e, and travel angle λ. From the preparation work, we get these equations of motion for the angles

$$J_p\ddot{p} = L_1V_d \tag{1a}$$
$$J_e\ddot{e} = L_2cos(e) + L_3V_scos(p) \tag{1b}$$
$$J_\lambda\ddot{\lambda} = L_4V_scos(e)sin(e) \tag{1c}$$

Where the constants L_1, L_2, L_3 and L_4 are given by

$$L_1 = K_fl_p \tag{2}$$
$$L_2 = g(l_cm_c - 2l_hm_p) \tag{3}$$
$$L_3 = K_fl_h \tag{4}$$
$$L_4 = K_fl_h \tag{5}$$

The controllers and observers we are using needs a linear system. Therefore, we linearize the system and get these new equations of motion

$$\ddot{p} = K_1 V_d \tag{6a}$$
$$\ddot{e} = K_2 \tilde{V_s} \tag{6b}$$
$$\ddot{\lambda} = K_3 p \tag{6c}$$

The helicopter is on equilibrium when the elevation axis is zero. To keep the helicopter stationary a constant voltage $V_{s,0}$ is needed.

$$\tilde{V_s} = V_s - V_{s,0} \tag{7}$$

By inserting (7) into (1b) we have an expression for the equilibrium voltage (8)

$$V_{s,0} = \frac{L_2}{L_3} \tag{8}$$

We want to be able to control the pitch angle, and for this we used a PD controller given by the equation

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p}, \tag{9}$$

Where p_c is the desired pitch angle. By applying Laplace we found an expression for the transfer function, G(s). When the denominator of G(s) equals zero, a pole is placed here. This provided us with expressions for K_{pd} and K_{pp} as a function of the poles, λ_1 and λ_2

$$K_{pd} = \frac{-(\lambda_1 + \lambda_2)}{K_1} \tag{10}$$

$$K_{pp} = \frac{\lambda_1 \lambda_2}{K_1}. \tag{11}$$

where

$$K_1 = \frac{K_f l_p}{J_p} \tag{12}$$

2.1 Equilibrium

To achieve stability we need to calculate the elevation angle offset. This was done by measuring the encoder value when the arm between the helicopter head and the elevation axis was horizontal. Then the angle was subtracted from the encoder so the elevation axis read zero when the helicopter is at linearization point. The elevation offset was measured to $e = 0.53[rad]$

From this we can calculate the motor force constant K_f. By rearranging (8) we get an expression for K_f, which is dependent on V_s. By measuring the voltage required to make the helicopter maintain equilibrium, we find $V_{s,0}$ and can calculate K_f. The measured voltage, $V_{s,0}$, is 7.5 and the calculated motor force constant is 0.1332.

2.2 Pole placements and PD control

By choosing different values for the poles we achieve different levels of stability. Where the imaginary part, β, gives an indication of how fast the system oscillate. While the real part, α, says something about how damped the waves are and how fast the system reacts. It is basic knowledge that when the poles are in the right half plane, the system is highly unstable, thus testing these values could damage the helicopter. Coincident eigenvalues on the imaginary axis gives an unstable response because they give integral effect. [1]

Table 1: Poles and regulator constants.

Poles	K_{pp}	K_{pd}	Predicted stability	Observed stability
$\lambda_1 = -1 \pm i$	3.784	3.784	Stable	Stable
$\lambda_2 = -2 \pm 1.5i$	11.826	7.5688	Stable	Stable
$\lambda_3 = -0 \pm 0i$	0	0	Unstable	Unstable
$\lambda_4 = -0 \pm 2i$	7.569	0	Marginal	Unstable
$\lambda_5 = -2 \pm 0i$	7.5688	7.5688	Stable	Stable
$\lambda_6 = -10 \pm 5i$	236.525	37.844	Unstable	Unstable
$\lambda_7 = -0.5 \pm 0.5i$	0.9461	1.8922	Stable	Stable

Table 1 shows the poles we chose to experiment with, their corresponding K_{pp} and K_{pd}, and our prediction of their stability and their stability we observed at the lab. We predicted λ_1, λ_2 and λ_7 to be stable since they consists of complex conjugated pair with imaginary values around -1 and small K_{pp} and K_{pd} values. We assume that λ_5 is stable since it has coincident poles on the negative real axis. λ_6 is complex conjugated but K_{pp} is so large that it will become unstable. λ_4 has complex conjugated poles on the imaginary axis, so we assume that it is marginally stable. Since λ_3 has coincident poles in the origin, we assume that it is unstable.

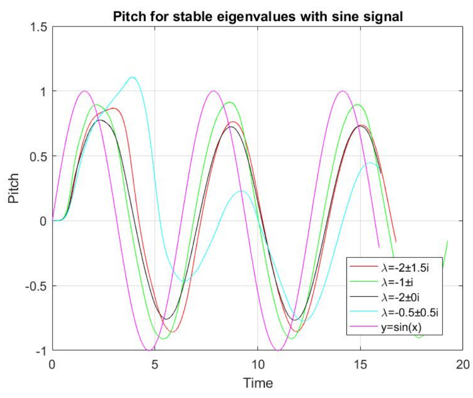


Figure 2: Pitch for stable eigenvalues. With sine signal as reference

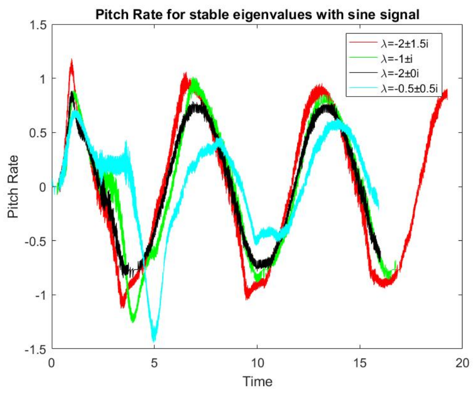


Figure 3: Pitch rate for stable eigenvalues. With sine signal as referance

In figure 2 we can see how the system with stable eigenvalues is able to follow a sine signal as reference. As we can see the blue line, with λ_7, struggles to follow the reference, and thus is not as stable as the others. We observe that in the beginning the red line, λ_2 also struggles to follow the signal, but it catches up after a few periods. The green one, λ_1 has a higher amplitude than the other poles and the phase is also less shifted because it reacts faster. The black one, λ_5 is also stable and is able to follow the sine signal, but its got a lower amplitude and the phase is more shifted than the green one. From pitch rate you can see that the blue, λ_7 struggles to follow, but the rest has the similar response.

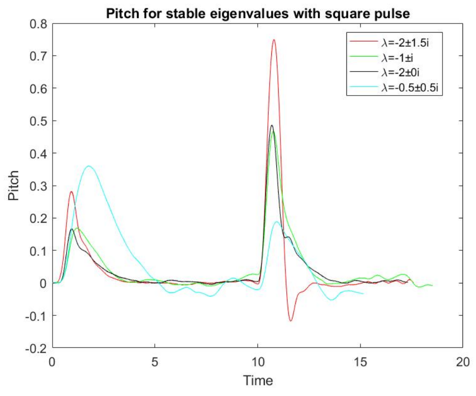


Figure 4: Pitch for stable eigenvalues. With square signal as reference

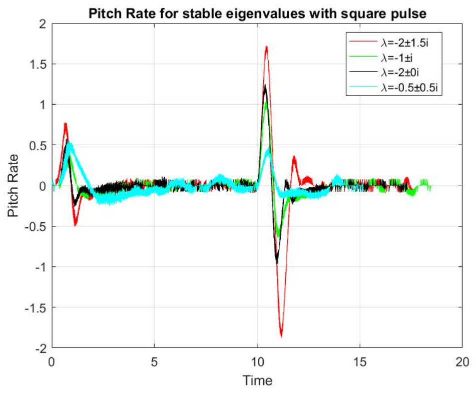


Figure 5: Pitch rate for stable eigenvalues. With square signal as reference

We wanted to check how our poles reacted to different reference signals, therefore we chose to test with a square signal as well. As we concluded with for the sine signal, the blue line, λ_7, and the red line, λ_2 has a worse reaction than the black, λ_5, and the green, λ_1. The blue is a little bit oscillating and wiggly but seems more stable than with the sine signal, therefore it was educational to test more signals. We observe in this figure that blue, λ_7, and black, λ_5, have similar response to the square signal, and they could both work as our poles in this case. But, from the sine signal we can see that the λ_1 is the better option in this case.

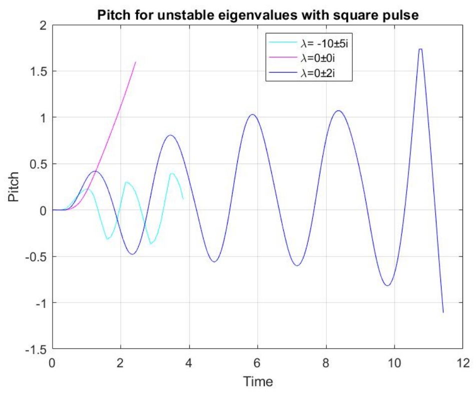


Figure 6: Pitch for unstable eigenvalues. With square signal signal as reference

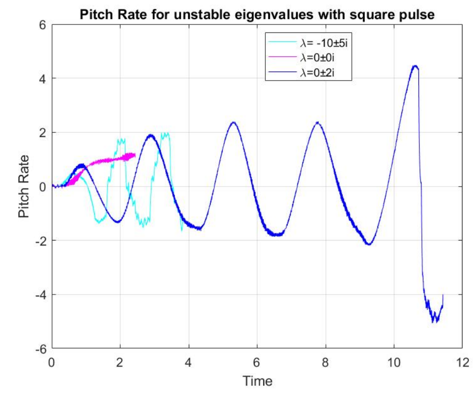


Figure 7: Pitch rate for unstable eigenvalues. With square signal signal as reference

In figure 6 the poles with an unstable outcome are illustrated. The pink line, λ_3, is the graph for coincident poles in origin. As mentioned, this results in an unstable system, which we can also observe from the figure. As we can see from the figure the amplitude for the blue line, λ_4, builds up to infinity, thus is unstable. This concur with the theory that systems with poles on the imaginary axis are unstable.

From the theory λ_6 is marginally stable, but since K_{pp} and K_{pd} is so large, the damping of the system makes it unstable and unable to rise.

After evaluating all the poles we conclude that we want λ_1 as our pole. This is because it is stable and has a fast response to references.

3 Part II – Multivariable control

Until now we have identified the equilibrium point and implemented a PD controller to control the pitch angle. Now we have to find a way to control and stabilize the remaining states of the system. In this part we are going to experiment with pole placement and LQR with and without integral effect, and discuss which one provides the best control. By letting a joystick provide the x-axis and the y-axis reference, respectively denoted p_c and $\dot{e_c}$, we can control the pitch angle and the elevation rate. From this we get the state space formulation:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{13}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices given by

$$\boldsymbol{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} \boldsymbol{u} = \begin{bmatrix} \tilde{V_s} \\ V_d \end{bmatrix} \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & \mathrm{K}_1 \\ \mathrm{K}_2 & 0 \end{bmatrix} \tag{14}$$

For a controller to work we have to check if the system is controllable. The method used is calculating the controllability matrix and check for full row rank. The matrix is calculated to be

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & 0 & K_1 & 0 & 0 \\ 0 & K_1 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{15}$$

.

This matrix has full row rank, $rank(\mathcal{C}) = 3$, thus the system i controllable. This means that we are able to directly alter the states, within a finite time interval, from changing the input signal. The only needed information is retrieved from the current system states, therefore there is no need for information of the history of states. Now that we know that the system is controllable we have to use feedback to obtain the states. The state-feedback controller we are going to use is given by

$$\boldsymbol{u} = \boldsymbol{F}\boldsymbol{r} - \boldsymbol{K}\boldsymbol{x} \tag{16}$$

Where $\boldsymbol{u}$ is given by

$$\boldsymbol{u} = [(\boldsymbol{B}\boldsymbol{K} - \boldsymbol{A})^{-1}\boldsymbol{B}]^{-1}\boldsymbol{r} - \boldsymbol{K}\boldsymbol{x} \tag{17}$$

We want the system output to converge to the references, p_c and $\dot{e_c}$. From this information we can find an expression for the feedforward matrix $\boldsymbol{F}$. By comparing equation 16 and 17 one can observe that $\boldsymbol{F}$ is given by

$$(\boldsymbol{BK} - \boldsymbol{A})\boldsymbol{x_\infty} = \boldsymbol{BFr_0} \tag{18}$$

Where $\boldsymbol{r}$ is the reference, $\boldsymbol{K}$ is the feedback gain matrix and $\boldsymbol{F}$ is feedforward matrix as follows:

$$\boldsymbol{r} = \begin{bmatrix} p_e, & \dot{e_c} \end{bmatrix}^\top \tag{19}$$

$$\boldsymbol{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix} \tag{20}$$

$$\boldsymbol{F} = [\boldsymbol{C}(\boldsymbol{BK} - \boldsymbol{A})^{-1}\boldsymbol{B}]^{-1} = \begin{bmatrix} k_{11} & k_{13} \\ k_{21} & k_{23} \end{bmatrix} \tag{21}$$

3.1 Pole placement

The system is now implemented and set for testing out the different control methods. First we tested out pole placement, a method used in feedback control where you place poles in a predetermined location. Doing this is beneficial because the pole placement is directly tied to the systems eigenvalues, which determines the response and stability. Hence, good pole placement can ensure a stable system with desired response.

The poles of a system with full state feedback are given by $\det[s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{BK}] = 0$. Based on this we know that the values of $\boldsymbol{K}$ forces the poles to be located at the desired position.

With different poles the system changes stability and rate of oscillation. We decided not to experiment with poles in the right half plane because this will generate an unstable system and damage the helicopter. Also, poles placed on the imaginary axis would only give us a marginally stable system, therefore we skipped experimentation with these as well[1]. Because of this we only tested out complex conjugated poles and poles on the negative real axis as they are able to generate a stable system. We experimented with having the poles smaller than -1, close together and far apart, and complex conjugated. As you can see in table 2.

Table 2: Poles

Vector of poles
$p_1 = \begin{bmatrix}-1 & -1.5 & -2\end{bmatrix}$
$p_2 = \begin{bmatrix}-1.2 & -1.3 & -1.4\end{bmatrix}$
$p_3 = \begin{bmatrix}-1 & -5 & -10\end{bmatrix}$
$p_4 = \begin{bmatrix}-0.1 & -1.3 & -2.5\end{bmatrix}$
$p_5 = \begin{bmatrix}-1 & -2+i & -2-1i\end{bmatrix}$
$p_6 = \begin{bmatrix}-0.3 & -1.1+i & -1.1-i\end{bmatrix}$
$p_7 = \begin{bmatrix}-0.7 & -1.1+i & -1.1-i\end{bmatrix}$

From the theory we can predict what kind of response the experimented poles will have. The poles with close spacing, p_2 will have a slow response and it will take some time for it to reach the desirable stationary value. Poles with large spacing, like p_3, have fast response but could end up with overshooting or reach satiation.

This experimentation gave a lot of different responses, some were desirable and some worked poorly. We decided to mostly look at pitch and elevation rate, because these are the references of the system. But while plotting we realized that the elevation rate was very noisy and it was difficult to tell the different poles apart, therefore we decided to instead plot elevation as this gave similar response with less noise. The responses of the pole placement are plotted beneath, in figures 8 and 9.

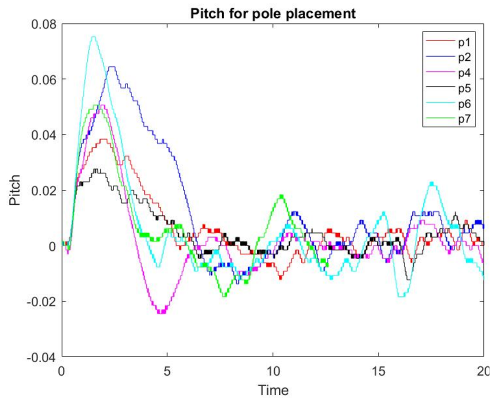


Figure 8: Pitch Pole Placement Without Largest Pole

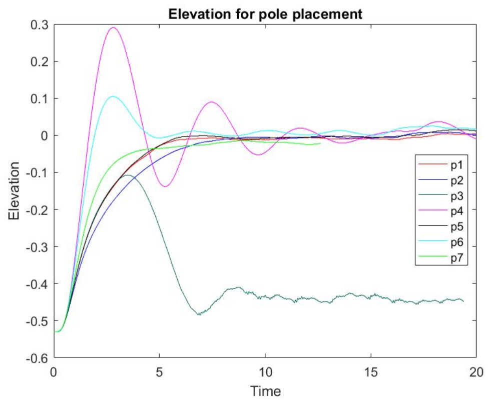


Figure 9: Elevation Pole Placement

We did not include pole p_3 in figure 8, because the overshoot was so big that it scaled down the other measurements. From 9 you see that it struggles to reach desired elevation. This corresponds with our expectation of a pole with large spacing. Also, the pole with the largest spacing besides p_3, pole p_4, has a fast response, but oscillates before stabilizing, which corresponds with the theory.

Our expectations about a pole with small spacing, like p_2, having slow reaction is confirmed in figure 9 where it is lagging behind the others.

We observe that p_1 and p_5 have similar response for elevation, but the pitch is quite different. This is because the pole closest to the imaginary axis is the same for both of them, -1. This is the pole that decides how fast the system is, hence they have the same speed/response. But they still are a little bit slow, so we will not use these poles.

Because we prioritize a fast system, we preferred the response of p_6 but we wanted it without the overshoot. Therefore we increased the negative value of the closest pole so that we got a fast response but not as aggressive. Therefore, we ended up with using pole p_7.

3.2 LQR

Finding the right pole combination in pole placement is challenging. Therefore we implemented and tested out the linear-quadratic regulator(LQR), which is another feedback controller where you can control each state directly. This controller demands a linear and controllable system, which from section 3 we know our system fulfills. The method is based on optimizing the cost function (22) under the constraints of the system.

$$\boldsymbol{J} = \int_0^\infty \boldsymbol{x}^\top \boldsymbol{Q_{LQR}} \boldsymbol{x}(t) + \boldsymbol{u}^\top(t) \boldsymbol{R}_{LQR} \boldsymbol{u}(t) dt \tag{22}$$

This provides us with two weighting matrices, $\boldsymbol{Q_{LQR}}$ and $\boldsymbol{R_{LQR}}$, representing the cost for state error and the cost for input errors, respectively. For simplicity we will operate with diagonal weighting matrices, where each of the diagonal elements corresponds to a different state, e.g. in $\boldsymbol{x}$, Q_{22} affects pitch angle error, $x_2 = \dot{p} - \dot{p_c}$. By increasing the value of $\boldsymbol{Q_{LQR}}$ you penalize the state error more, and the response will be closer to the reference value. To get a well tuned system we had to optimize these weighting matrices. [3]

We experimented with different diagonal values for $\boldsymbol{Q_{LQR}}$ and $\boldsymbol{R_{LQR}}$. It is interesting to look at high values versus low values to see the direct effect of variations in values. By experimenting with different $\boldsymbol{R_{LQR}}$ values we noticed that it had less effect on the system than $\boldsymbol{Q_{LQR}}$ and thus we focuses on experimenting more with different $\boldsymbol{Q_{LQR}}$ matrices. Regardless $\boldsymbol{R_{LQR}}$ affected how much power the motors had, and too small values made the system reach saturation and large values did not give the motors enough power.

The result of our experimentation is shown in figures 10 and 11, with the matrices given in table 3.2.

Diagonal of $\boldsymbol{Q_{LQR}}$
$\boldsymbol{Q_1} = \begin{bmatrix}2 & 0.5 & 4\end{bmatrix}$
$\boldsymbol{Q_2} = \begin{bmatrix}15 & 1 & 7\end{bmatrix}$
$\boldsymbol{Q_3} = \begin{bmatrix}75 & 3 & 25\end{bmatrix}$
$\boldsymbol{Q_4} = \begin{bmatrix}100 & 40 & 100\end{bmatrix}$
$\boldsymbol{Q_5} = \begin{bmatrix}0.3 & 0.2 & 0.2\end{bmatrix}$
$\boldsymbol{Q_6} = \begin{bmatrix}5 & 5 & 10\end{bmatrix}$

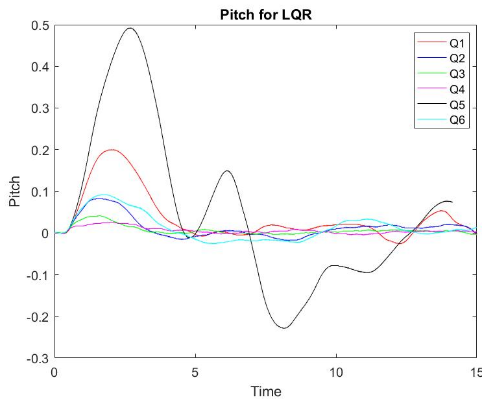


Figure 10: Pitch for LQR

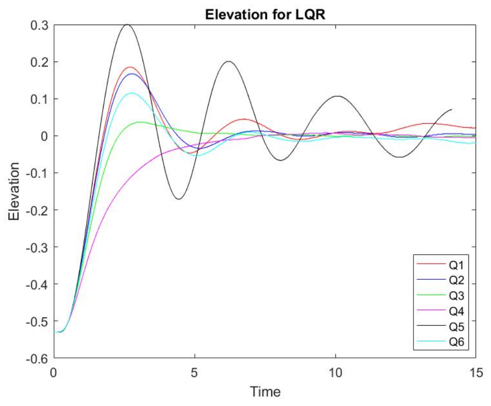


Figure 11: Elevation for LQR

As mentioned different values of $\boldsymbol{Q_{LQR}}$ affects the helicopter behaviour massively, which the plots validates. We observe that quite small values for $\boldsymbol{Q_{LQR}}$, as for $\boldsymbol{Q_5}$, gave large overshoot from desired value. This is because there is almost no penalizing of the error. We observe that as the magnitude of $\boldsymbol{Q_{LQR}}$ increased the states came closer to the desired value. This corresponds with the errors being more penalized, and the acceptance of error decreasing.

If we compare $\boldsymbol{Q_1}$ and $\boldsymbol{Q_2}$ we can directly observe how the magnitude of $\boldsymbol{Q_{LQR}}$ affects the different states of the system. Q_{11} which affects pitch error is scaled by a lot. From this observation we can see that the pitch is overshooting a lot less in $\boldsymbol{Q_2}$ than $\boldsymbol{Q_1}$, this is because the error is more penalized and does not have the ability to overshoot more. While Q_{33}, affecting elevation, is scaled by less than two. For the elevation the scaling was smaller and we can see that the two looks similar, which we also observed at the lab. This gives the conclusion that $\boldsymbol{Q_{LQR}}$ affects how much room there is to go over the desired value. This kind of behaviour is desirable, but we can see that it needs to be penalized further.

Therefore, the remaining alternatives for the optimal LQR are $\boldsymbol{Q_3}$ and $\boldsymbol{Q_4}$. Both of these have positive properties. $\boldsymbol{Q_3}$ has a faster response, but it has a small overshoot before it stabilizes at the stationary value. While $\boldsymbol{Q_4}$ has a slower, but more accurate response, with less overshoot. These are both properties that we require for an optimal system, and a combination of the two would be ideal. Nonetheless, we have to decide what we prioritize for this system, and based on what we observed in the lab, the faster response is prioritized and would work better for this helicopter. However, because of safety the slower and more accurate response would be the best choice for a real helicopter.

3.3 LQR with integral action

LQR alone might not generate desired response and to improve it we can include an integral effect. This introduces two new states

$$\dot{\gamma} = p - p_c \tag{23}$$

$$\dot{\zeta} = \dot{e} - \dot{e}_c. \tag{24}$$

With this new effect, the system has to be modified and gets the following form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{G}\boldsymbol{r} \tag{25}$$

To get the right dimension of the matrices, they have to be modified as well. We then get

$$\boldsymbol{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \\ \gamma \\ \zeta \end{bmatrix}, \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{G} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{26}$$

With this LQR controller we will get steady state error. By introducing integral effect we prevent this from happening and ensure that any steady state error will be eliminated and that process disturbances will be compensated for. By now adjusting LQR we can decide at which rate we want the errors to be reduced. This makes the helicopter approach the equilibrium point at a faster rate. As mentioned we get two new states and therefore $\boldsymbol{Q}$ and $\boldsymbol{R}$ will get two new parameters. These will penalize the accumulated error for $\int p - p_c dt$ and $\int \dot{e} - \dot{e}_c dt$.

We experimented with different diagonal values. The response is shown in figure 12 and 13, with the matrices given in table 3.3. Our implementation of the LQR with integral effect is given in 57 in the appendix.

Diagonal of $\boldsymbol{Q_{LQR}}$
$\boldsymbol{Q_{i1}} = \begin{bmatrix} 0.1 & 5 & 20 & 0.8 & 0.8 \end{bmatrix}$
$\boldsymbol{Q_{i2}} = \begin{bmatrix} 0.41 & 1 & 2.04 & 2.04 & 0.86 \end{bmatrix}$
$\boldsymbol{Q_{i3}} = \begin{bmatrix} 90 & 20 & 17 & 87 & 12 \end{bmatrix}$
$\boldsymbol{Q_{i4}} = \begin{bmatrix} 14 & 0.1 & 5 & 11 & 13 \end{bmatrix}$
$\boldsymbol{Q_{i5}} = \begin{bmatrix} 75 & 50 & 75 & 4 & 5 \end{bmatrix}$
$\boldsymbol{Q_{i6}} = \begin{bmatrix} 5 & 5 & 10 & 40 & 90 \end{bmatrix}$

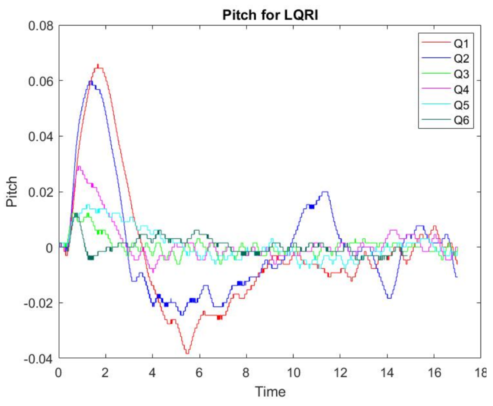


Figure 12: Pitch for LQRI

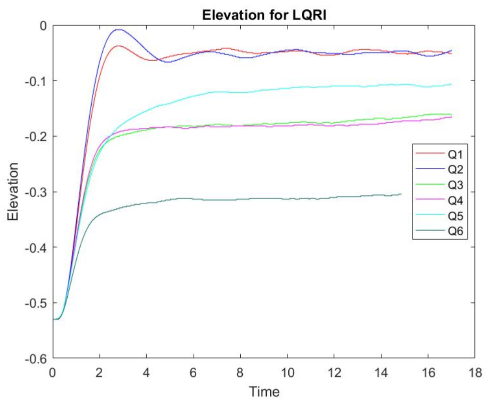


Figure 13: Elevation for LQRI

We observed that using this integral effect caused the helicopter to respond more precisely when we increased Q_{44} and Q_{55}, especially the latter. The response to the reference improved and it was easier to get the responses we wanted. As the references are for p and $\dot{e}$ the desired response when the joystick is at the center point is that the system should stabilize at the last given height. To achieve this the penalizing of γ and ζ is significant.

We quickly realized that a low, especially less than one, $\boldsymbol{Q_{LQRI}}$ value for ζ would not cause the helicopter to stop at desired height. Instead the system wanted to stabilize at linearization point. This is the case for $\boldsymbol{Q_{i1}}$ and $\boldsymbol{Q_{i2}}$. This is not the desired response, and the value needed to be increased. As we increased it the helicopter responded faster to the reference and tried to stay at the point the joystick was released. Significantly large values made the helicopter stop almost immediately after the joystick was released to neutral position, as for $\boldsymbol{Q_{i6}}$. This happens because the steady state error, $\dot{e} - \dot{e_c}$, will be penalized more and thus have a smaller area it can occur in. This forces the helicopter to be closer to the stationary value and therefore it is able to stop at the desired height.

As for the accumulated error for pitch, γ, there was no need for an equally large value. We observe this by comparing $\boldsymbol{Q_{i3}}$, $\boldsymbol{Q_{i4}}$ and $\boldsymbol{Q_{i5}}$, where the values are significantly different, but the pitch has similar response. This also correlates to the $\boldsymbol{Q_{LQR}}$ value for pitch, where larger penalizing of pitch also makes the helicopter reach the linearization point faster. Still, for the pitch to quickly go back to the linearization point after a change in reference, the Q_{44} value needed to be quite large. The same argument for penalizing ζ is valid for γ.

The integral effect also caused the helicopter to react more rapidly and we needed to adjust the previous values in $\boldsymbol{Q_{LQR}}$ to get a slower and more precise system. This caused a large decrease in both Q_{11} and Q_{33}, and a small increase in Q_{22}. We got the best response from system $\boldsymbol{Q_{i6}}$, as it was fast and stabilized more precisely.

As the penalizing of ζ increased the helicopter would initially fly lower, as you can see in figure 13. This is acceptable response, and the reason why this happens is because the reference is elevation rate and not elevation. When the joystick is in neutral position the reference would be zero, therefore the elevation rate should be zero as well. The reason why it flies at all is because it gets a pulse in the beginning as it tries to reach the equilibrium, as you can see in figure 15, and this is an error compared to the reference. Because ζ is connected to this error a large penalizing of it will cause it to eliminate the error faster. This is the reason why helicopters with larger penalizing of ζ will reach a lower height initially than helicopters with less penalizing.

To see the direct effect of the integral joint, we need to compare the system with and without it for the preferred tuning, $\boldsymbol{Q_{i6}}$. In this comparison we had the joystick in neutral position, thus the references are zero and the value should converge to this. This is shown in figures 14 and 15.

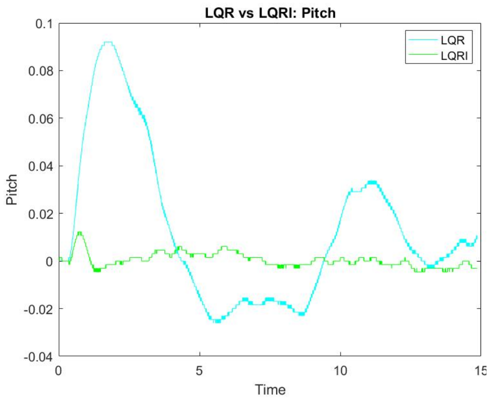


Figure 14: LQR Vs LQRI: Pitch Angle

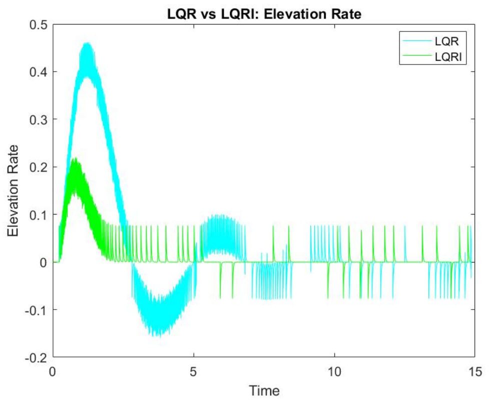


Figure 15: LQR VS LQRI: Elevation Rate

We observe that the integral effect has a massive impact on the system response. We can see that the overshoot from the LQR without integral effect is significantly larger and with a longer duration than with integral effect, both for pitch and for elevation rate. The fact that the elevation rate takes longer to stabilize at zero without integral effect explains why the helicopter reaches the linearization point here and not with integral. From the preparation work we know that the system should work independent of what the $\boldsymbol{F}$ matrix was, because p and $\dot{e}$ would be equal to p_c and $\dot{e_c}$ due to the integral effect. It is also worth mentioning that the pitch stabilizes around zero with integral effect at a fast pace, while it takes a long time to stabilize without. The explanation is that LQR without integral effect does not integrate the steady state error, which LQRI does, and thus it is not eliminated. With integral effect you ensure that any steady state error is eliminated and makes sure that you reach the desired value, but without there is no way to guarantee that it will be reached.

4 Part III – Luenberger observer

An internal measurement unit (IMU) is practical device because one can measure the internal state $\boldsymbol{x}$, given in equation (31), of the helicopter, without being stuck to the ground.

4.1 IMU characteristics

The IMU measureres the acceleration of its local system as seen in figure 1. A gyroscope is used to measure the rotational velcoity in $\frac{rad}{s}$, whilst a accelormeter measures the proper acceleration in $\frac{m}{s^2}$.

The following figures shows the measurements versus the encoder output. The measurements are for the pitch rate, $\dot{p}$, elevation rate, $\dot{e}$ and travel rate $\dot{\lambda}$, while the helicopter is rotated around each axis. When the helicopter is rotated about one axis at the time, the gyroscope output follows the encoder-rate. As seen in figure 16, 17 and 22, the gyroscope measurements and encoder values are pretty much the same. On the other hand when the helicopter is rotated around two or more axis the measurements diverges from the encoder values.

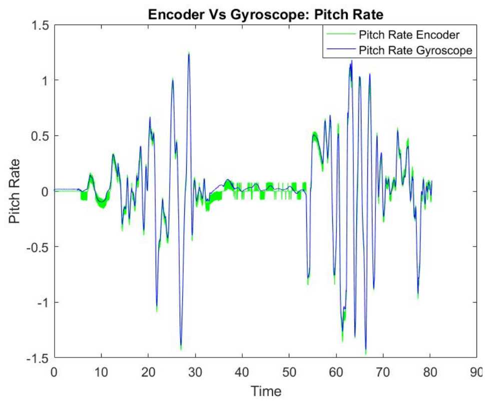


Figure 16: Pitch Rate Encoder vs IMU

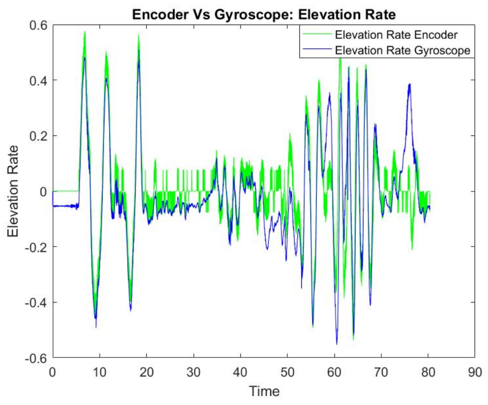


Figure 17: Elevation Rate Encoder Vs IMU

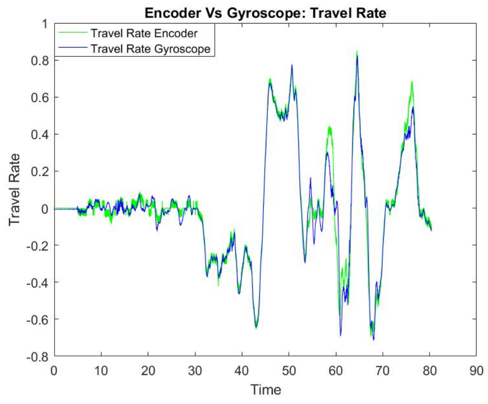


Figure 18: Travel Rate Encoder vs IMU

The following figure, figure 19, shows the accelerometer output while the helicopter is rotated around each axis. The output of the accelorometer is the acceleration in the x-, y- and z- given in figure 1, which we denote a_x, a_y and a_z. From the figure we can observe that a_y and a_z responds in sync. This make sense since a_y and a_z is both dependent on the pitch and elevation angle, while a_x is only dependent on the elevation angle. This can be calculated by using newtons laws and trigonometric identities on the forces given in figure 1.

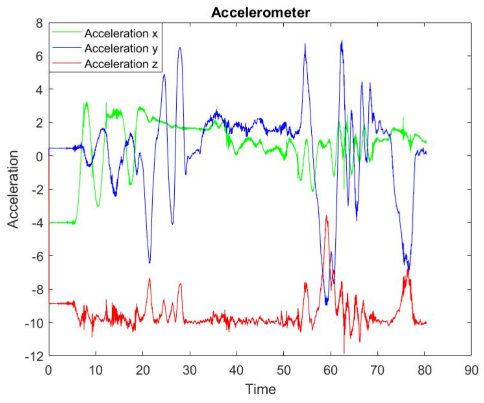


Figure 19: Accelerometer Output

4.2 Transformations

To counteract the faulty gyroscope measurements, we can indirectly measure the pitch-, elevation and travel-rate by using it as a input in "$gyro\_vec\_to\_euler\_rate$"

in the simulink diagram 55. The output is vector with the correct measurements of pitch-, elevation- and travel-rate, given in unit $\frac{rad}{s}$ The following figures show how the transformed gyroscope measurements, which we will call the euler measurements, responds when the helicopter is rotated around each axis.

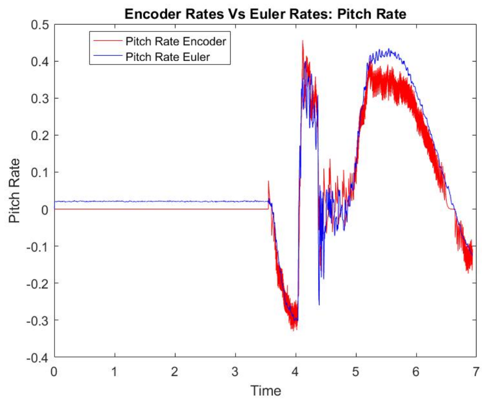


Figure 20: Pitch Rate Encoder vs Euler

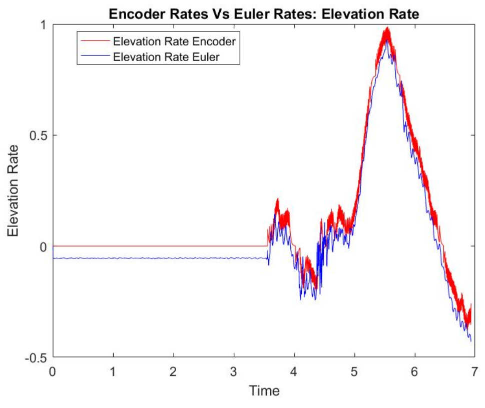


Figure 21: Elevation Rate Encoder vs Euler

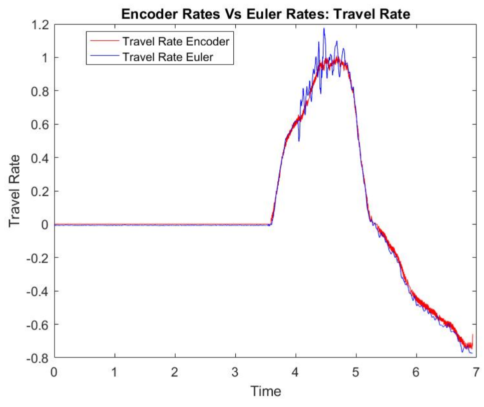


Figure 22: Travel Rate Encoder vs Accelerometer

We can also use the accelerometer to measure the pitch- and elevation-angle, because the acceleration in each axis is dependent on the angle of the pitch and elevation. The equations for the pitch, p, and elevation, e, is given in equation 27 and 28

$$p = \arctan\left(\frac{a_y}{a_z}\right) \tag{27}$$

$$e = \arctan\left(\frac{a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right) \tag{28}$$

The implementation of equations is given in the appendix in section A.3. Since a_y and a_z can both be zero, we could end up with zero in the denominator. To avoid this we implemented the equations so that p is zero when a_z is zero and e is zero when both a_y and a_z is zero. The following figures 23 and 24 show the measured pitch and elevation angle measured indirectly from the accelerometer. The reason of the delay in the measurements figures is that the measurements started a couple of seconds before we moved the helicopter.

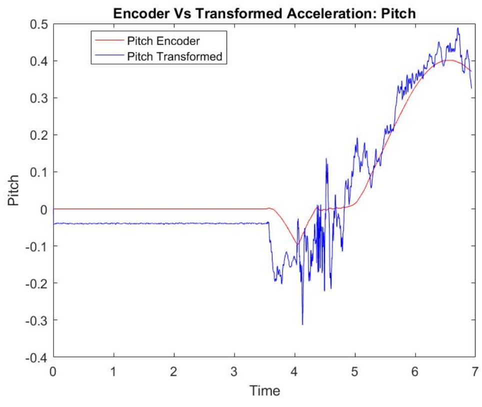


Figure 23: Pitch Encoder vs Accelerometer

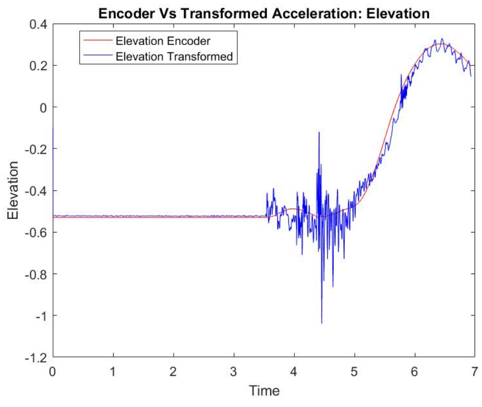


Figure 24: Elevation Encoder vs Accelerometer

By combining the euler rates from figure 20, 21 and 22 and the transformed accelerations from figure 23 and 24, we get measurements for each state in $\boldsymbol{x}$ given in the matrix equation (31).

With the new $\boldsymbol{x}$ we can extend the state space formulation for the system:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{29}$$
$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{30}$$

Where the matrices $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are given by

$$\boldsymbol{x} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \end{bmatrix}, \boldsymbol{u} = \begin{bmatrix} V_s \\ V_d \end{bmatrix}, \boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ K_3 & 0 & 0 & 0 & 0 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \end{bmatrix} \tag{31}$$

4.3 Theoretical discussion

The estimated state given by the euler rates and transformed accelerometer measurements contains a lot of noise, compared to the encoder rates in the previous sections. An observer can be used to smooth out the measurements. Before an observer can be implemented we have to check if the system is observable. This is easily done by calculating the observability matrix and check if it has full column rank, $rank(\mathcal{O}) = 5$.

If we only observe the system through the transformed gyroscope measurements, $\dot{p}$, $\dot{e}$ and $\dot{e}$, we get a rank of $rank(\mathcal{O}) = 4$. Hence, by only observing the rates, we get no information about the angles and thus the system is not observable.

From equation (6c) we can see that by deviating $\dot{\lambda}$ we get information about p. This means that by observing travel rate, $\dot{\lambda}$, we know the state of the pitch angle and rate. We only have to measure the elevation angle, e, to get information about both elevation angle and rate. Therefore, it is not sufficient to only use the gyro measurements, some angle measurements are required as well. The transformed accelerometer measurements can provide us with the angle measurements. To conclude the minimum set of states measured that makes the system observable is 2. The states are e and $\dot{\lambda}$, which indirectly gives measurements of the entire system. Since these two states come from the transformed accelerometer and gyroscope, we have to use both sensors to observe the system. Our implementation of the estimated state is given in figure 56 in the appendix.

4.4 State estimator

Since our system is linear, we can use a linear observer. This means that we can use the Luenberger observer, with equation given by (32). Since $\{\boldsymbol{A}, \boldsymbol{C}\}$ are observable, we can choose the eigenvalues of $\boldsymbol{A} - \boldsymbol{LC}$. We experimented with different pole placement which gave us different values for the gain matrix $\boldsymbol{L}$. [3]

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}(\boldsymbol{y} - \boldsymbol{C}\hat{\boldsymbol{x}}) = (\boldsymbol{A} - \boldsymbol{LC})\hat{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{L}\boldsymbol{y} \tag{32}$$

Where $\hat{\boldsymbol{x}}$ is the state estimator and $\boldsymbol{L}$ is the Luenberger gain matrix. When we trust that our estimate $\hat{\boldsymbol{x}}$ is correct we want it to affect our system more than our model $\boldsymbol{A}$. The gain matrix $\boldsymbol{L}$ says something about how much we want to amplify the estimated measurements. When the gain matrix $\boldsymbol{L}$ is large, the previous estimated measurement have more effect on the new estimated value. The higher observer gain, the more rapidly it converges the system states. If $\boldsymbol{L}$ is small we have less trust in our measurement and more trust in our system model. With small observer gain, the system converges sluggish and with insignificant variance in measurement. To summarize, when $\boldsymbol{L}$ is large the variance in $\hat{\boldsymbol{x}}$ is large and vice versa when $\boldsymbol{L}$ is small.

From matrix properties we know that it is the columns in the $\boldsymbol{L}$ matrix that affects the values of each state. For example the values in first column will effect the pitch angle, while the values in the third column will effect the travel angle.

Table 3 shows the different poles we chose to experiment with. We found the Luenberger observer by using the matlab function $L = place(A', C', p)'$;. From the table you can see that the real values of the poles $\boldsymbol{p}$ sets the values along the diagonal of $\boldsymbol{L}$. Depending if the poles has zero, one or two pairs of complex conjugated poles, the off-diagonal elements is set with regards to the complex value or sum of them. The amount of coincident poles also sets the off-diagonal elements. The 0.0816 value is constant and independent from the poles, the value comes from the A and C matrix. Our implantation of the Luenberger is given in figure 58 in the appendix.

Table 3: Luenberger poles

Vector of poles	Corresponding Luenberger matrix
$p_1 = \begin{bmatrix}-8 & -8\pm i & -8 & -8\end{bmatrix}$	$L_1 = \begin{bmatrix} 8 & 0 & 0 & 0 & 0 \\ 1 & 8 & 0 & 0 & 0 \\ 0 & 0 & 8 & 1 & 0 \\ 0 & 0 & 0 & 8 & 0 \\ 0.0816 & 0 & 0 & 0 & 8 \end{bmatrix}$
$p_2 = \begin{bmatrix}-5 & -50\pm 5i & -25\pm 2i\end{bmatrix}$	$L_2 = \begin{bmatrix} 50 & -4 & 0 & 0 & 0 \\ 5 & 50 & 0 & 0 & 0 \\ 0 & 0 & 25 & -1 & 0 \\ 0 & 0 & 2 & 25 & 0 \\ 0.0816 & 0 & 0 & 0 & 5 \end{bmatrix}$
$p_3 = \begin{bmatrix}-65 & -50\pm 5i & -75\pm 10i\end{bmatrix}$	$L_3 = \begin{bmatrix} 50 & -4 & 0 & 0 & 0 \\ 5 & 50 & 0 & 0 & 0 \\ 0 & 0 & 75 & -9 & 0 \\ 0 & 0 & 10 & 75 & 0 \\ 0.0816 & 0 & 0 & 0 & 65 \end{bmatrix}$
$p_4 = \begin{bmatrix}-80 & -80\pm 10i & -80 & -80\end{bmatrix}$	$L_4 = \begin{bmatrix} 80 & -9 & 0 & 0 & 0 \\ 10 & 80 & 0 & 0 & 0 \\ 0 & 0 & 80 & 1 & 0 \\ 0 & 0 & 0 & 80 & 0 \\ 0.0816 & 0 & 0 & 0 & 80 \end{bmatrix}$
$p_5 = \begin{bmatrix}-1 & -2\pm 3i & -4 & -5\end{bmatrix}$	$L_5 = \begin{bmatrix} 2 & -2 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0.0816 & 0 & 0 & 0 & 5 \end{bmatrix}$
$p_6 = \begin{bmatrix}-1 & -1 & -1 & -1 & -1\end{bmatrix}$	$L_6 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.0816 & 0 & 0 & 0 & 1 \end{bmatrix}$

From the table 3 you can see that the observer gain "pitch angle and rate columns" for observer 2 and 3, respectively matrix $\boldsymbol{L_2}$ and $\boldsymbol{L_3}$, have the same values, but different values for the other states. From this we can expect observer 2 and 3 to have the same pitch characteristic, but different elevation. Since observer 3's elevation rate column is three times larger than observer 2's column, we expect observer 3's column to be more variant and noisy.

The gain matrix for observer 6 contains small values less or equal to 1.

From the theory we can expect the measurements to have small variance, but they might be so small that its too sluggish and unable to follow the encoder values. Observer 5 value are larger than observer 6, but smaller than the rest. The pitch rate column contains -2 and 2, and the sum of them will give a pitch rate of zero. Since the pitch is dependent on the pitch rate, we expect the pitch angle and rate to be different from the encoder values.

The first observer gain matrix, $\boldsymbol{L_1}$, contains positive values that ranges from 1 to 8, excluding the constant value of 0.0816. The matrix has less and smaller values on the off-diagonal than the other observes. The values are larger than observer 5 and 6, but not as large as observer 3, 4 and 5. Observer 4 is 10 times larger than observer 1, except that it contains a negative value on the pitch rate column. From this we expect the measurements from observer 4 to be more noisy than observer 1, but it might be better at following the encoder values.

The following figures shows the pitch angle and elevation rate given for each observer given in 3. The plot contains the measurement from the encoder, euler and estimated states.

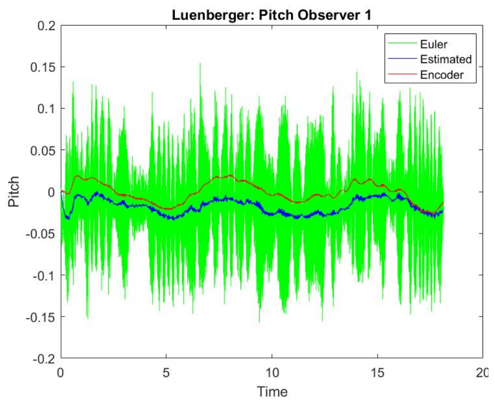


Figure 25: Observer 1: Pitch Angle

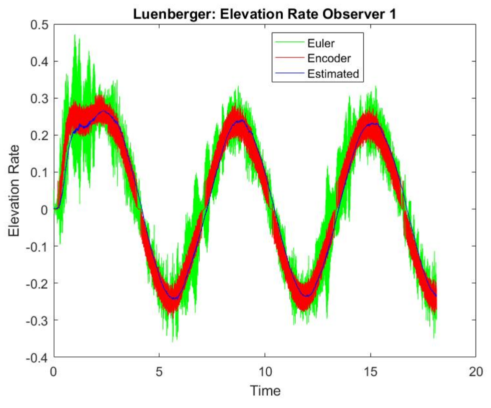


Figure 26: Observer 1: Elevation Rate

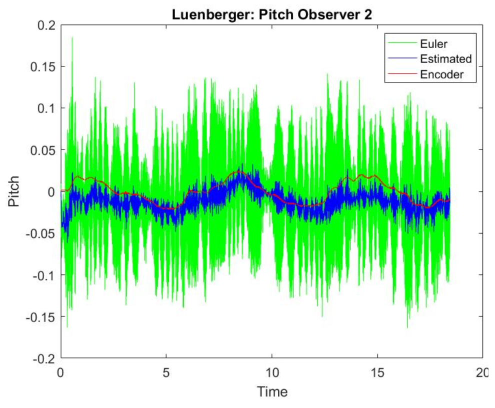


Figure 27: Observer 2: Pitch Angle

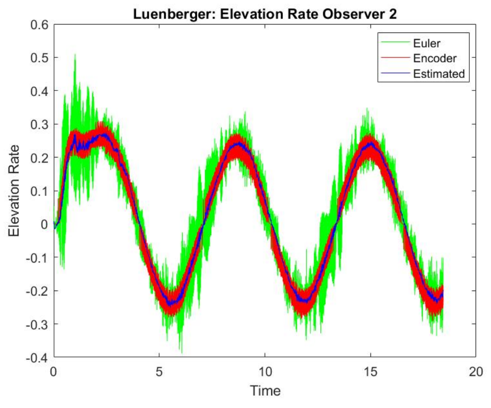


Figure 28: Observer 2: Elevation Rate

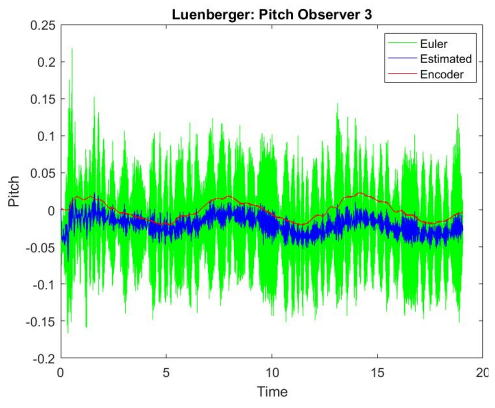


Figure 29: Observer 3: Pitch Angle

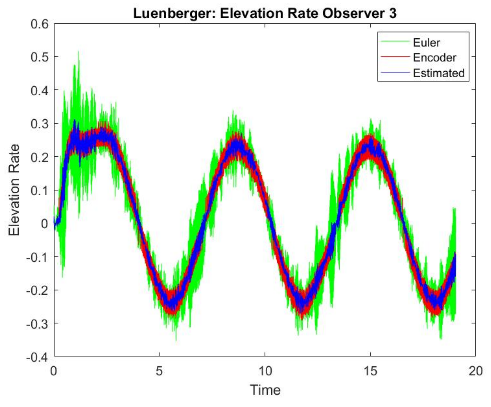


Figure 30: Observer 3: Elevation Rate

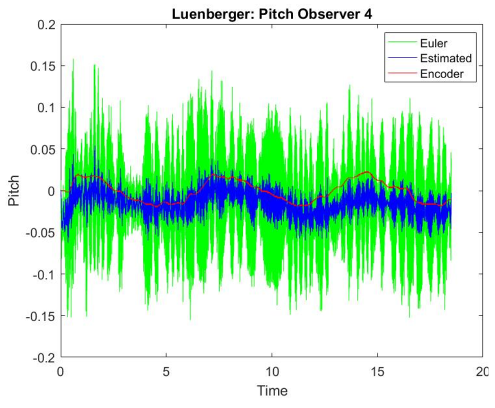


Figure 31: Observer 4: Pitch Angle

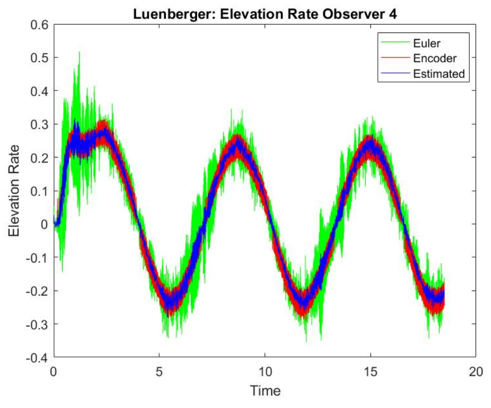


Figure 32: Observer 4: Elevation Rate

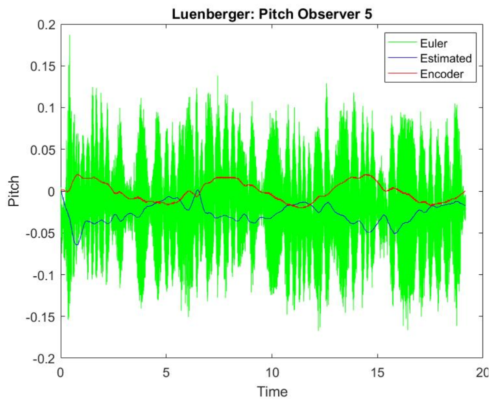


Figure 33: Observer 5: Pitch Angle

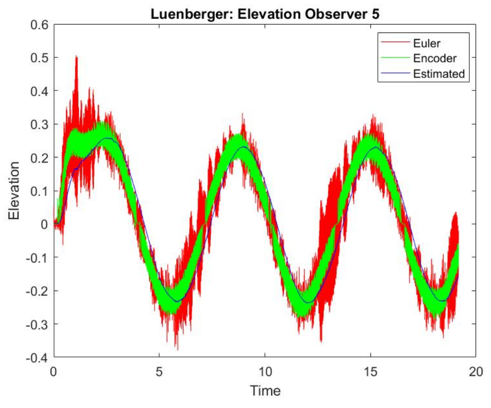


Figure 34: Observer 5: Elevation Rate

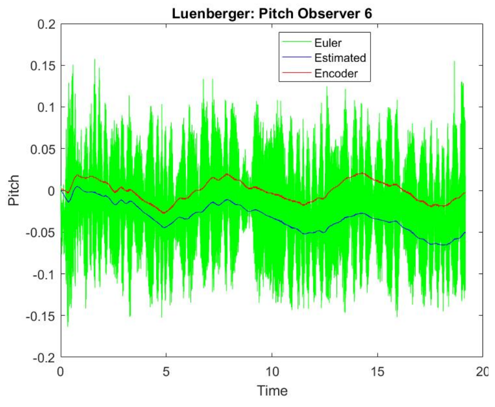


Figure 35: Observer 6: Pitch Angle

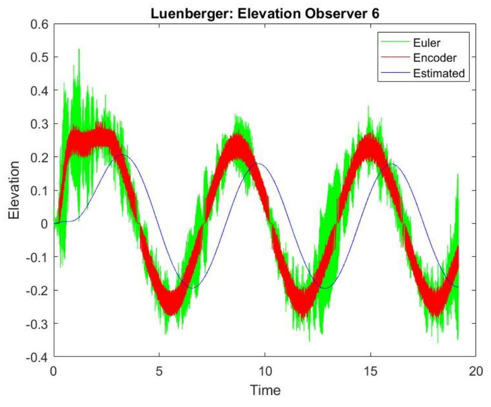


Figure 36: Observer 6: Elevation Rate

From the figures above you can observe that the green signal, the euler measurements, are rather noisy, but has a mean around the encoder value. The variance of the estimated value varies for each observer. By comparing figure 25, for observer 1, and figure 37, for observer 4, you can see that observer 1 has a smaller variance than observer 4. This is due to observer 1 having smaller values on the pitch column than observer 4, which has large values.

If we compare figure 27 and 29, which is the pitch for observer 2 and 3, you see that they almost seem like the same plot. When we compare figure 28 and 30, which is the elevation for the same observes, the variance of observer 3 is larger than for observer 2. This corresponds to our theory of the values of the columns affecting the noise and measurement of each state.

Figure 33 shows the pitch rate for observer 5. The estimated values are really off from the encoder values, and the two signals seems uncorrelated. The elevation rate for the same observer, given in 34, has ha time lag but is otherwise able to follow the encoder. The reason for the strange pitch angle is that the pitch rate always measures to zero, this is due to the gain of zero in the pitch rate from the gain matrix.

Observer 6's figure for the pitch, figure 35, shows that it has low variance but for each measurements it diverges further and further from the encoder values. The small values in the gain matrix $\boldsymbol{L_1}$ gives the measurement low variance but this also means that we trust the measurement too much, and several incorrect measurements makes it unable to follow the encoder values.

Observer 5 and 6 does not manage to follow the encoder, and therefore will give us a wrong estimate of our helicopters state. Observer 1, 2, 3 and 4 all manages to follow the encoder values, but they have a pitch offset. Since observer 1 has less variance and achieves the same estimate as the other observes, it is the better option. Even though observer 1's pitch has an offset from encoder, its pitch rate, elevation angle and travel rate is better than the other observers. We didn't include them in the report because it would be too many plots and would distract from our discussion about how the Luenberger observer works.

When using the estimated state we noticed that due to the offset the helicopter "believed" that is was at a lower value than it was, and it did not manage to lift itself higher than equilibrium. To counteract the offset we added an initial value on the gyro so when the helicopter started it would have the same values as the encoder rates. The initial value vector is given in 33.

$$gyro_{init} = \begin{bmatrix} -0.018 & 0.054 & 0.005 \end{bmatrix} \tag{33}$$

The following figure shows the estimated measurements as feedback. We chose observer 4 to estimate our feedback measurement. Since the estimated values now would be calculated based on our previous estimated values, rather than the encoder values, we cannot trust our estimate as much as we did with observer 1.

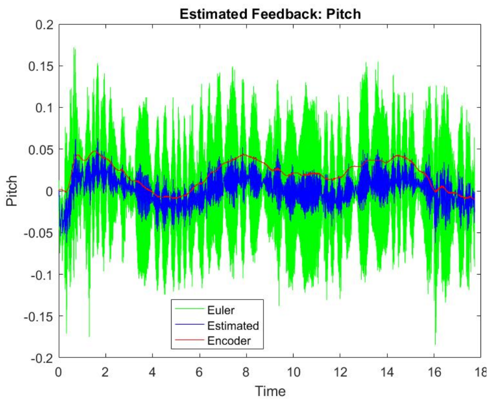


Figure 37: Estimated Feedback: Pitch Angle

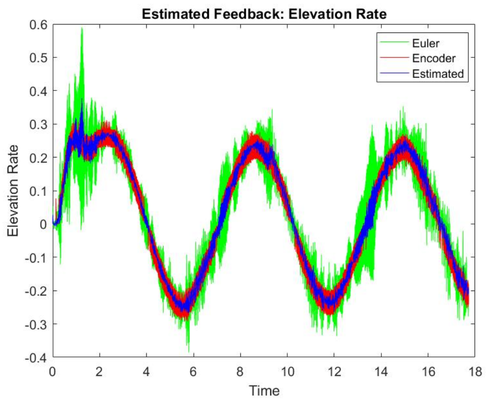


Figure 38: Estimated Feedback: Elevation Rate

5 Part IV – Kalman filter

Now we have experimented with a Luenberger observer, but we also want our observer to consider the system noise. Therefore we are going to implement and experiment with a Kalman filter.

The Kalman filter is an optimizer estimator, and as for any other estimator, it is used to get an estimation of the system states. In this project we will use this method to get an estimation of the states from the IMU measurements. This method uses estimates, with statistical noise, used over time to produce new estimates that tend to be more accurate than estimates which only uses single measurements alone, like in Luenberger. The reason is that it estimates a probability distribution over the states for each timestep [2]. The filter also minimizes the covariance estimate to find an optimal gain matrix $\boldsymbol{K}$, which reduces the uncertainty in the estimate, $tr(\boldsymbol{P})$, at the fastest rate, just as in Luenberger.

In physical systems model uncertainty, such as disturbances and measurement noise, is so large that it cannot be neglected. Therefore, they require proper treatment. The Kalman filter is dependent on finding the systems disturbance and noise, such that the uncertainty can be treated and the random properties are taken into account. $\boldsymbol{Q_d}$ is the stochastic disturbance and $\boldsymbol{R_d}$ is the stochastic measurement covariance. Higher value of $\boldsymbol{R_d}$, higher measurement noise, gives lower Kalman gain indicating less emphasis on measurements. Higher value of $\boldsymbol{Q_d}$, stronger disturbance gives higher Kalman gain, thus less emphasis on the model, and vice versa.

5.1 Noise estimate

We had to find an estimate for the measurement noise so that we could compensate for it. To be able to do this we also have to classify the noise, since different noise types need to be handled differently.

White noise is a theoretical concept where the observed noise have zero-mean and have a constant spectral density function, because it has equal intensity at different frequencies. It also requires that the random variables of the system are completely uncorrelated. [2]

We measured the noise at two different time-series, one stabilized at the linearization point, and one when the helicopter was lying still on the ground.

Looking at our measurements of the noise we observed that it looks similar to white noise. So, even though white noise is a theoretical concept, we can use an approximation and work with our system as if it consisted of white noise. Because of this we can use the Kalman filter with the assumption that we are working with white noise. Comparing the two noise measurements the demand for uncorrelatedness is more fulfilled for the ground noise than for the flying noise. This might be because the signal can be polluted by the regulation of the values, or other physical factors.

From the flying noise we calculated the measurement noise, $\boldsymbol{R_d}$ using the matlab function $\boldsymbol{cov(A)}$ on our measurement. This matrix, along with $\boldsymbol{Q_d}$ provides information about how much we trust the measurement versus our model. If $\boldsymbol{Q_d}$, the disturbance on the system, is larger than $\boldsymbol{R_d}$ we trust the measurement prediction of the system. This is important to consider when tuning the system, because if we have a lot of sensor noise we need to weight the measurements and estimates differently than with low noise.

The following is the measurement noise, $\boldsymbol{R_d}$, we will consider throughout the task

$$\boldsymbol{R_d} = \begin{bmatrix} 0.0033 & 0.0017 & 0.0027 & -0.00200 & 0.0003 \\ 0.0018 & 0.0265 & 0.0002 & 0.0005 & 0.0005 \\ 0.0027 & 0.00024 & 0.0036 & -0.0017 & 0.0003 \\ -0.0020 & 0.0005 & -0.0019 & 0.0030 & -0.0001 \\ 0.0003 & 0.00049 & 0.0003 & -0.0001 & 0.0005 \end{bmatrix} \tag{34}$$

5.2 Discretization

Since we are going to operate with a discrete kalman filter we need to discretize our continuous-time deterministic system, with all of the six states. This was done using the matlab function **c2d** on our system. With timestep 0.002 seconds this resulted in these matrices.

$$\boldsymbol{A_d} = \begin{bmatrix} 1 & 0.0020 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1.6e-07 & 1.1e-10 & 0 & 0 & 1 & 0.0020 \\ 0.00016 & 1.6e-07 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

$$\boldsymbol{b_d} = \begin{bmatrix} 0 & 1.06e-06 \\ 0 & 0.0011 \\ 1.7e-07 & 0 \\ 0.00017 & 0 \\ 0 & 2.8e-14 \\ 0 & 5.7e-11 \end{bmatrix} \tag{36}$$

$$\boldsymbol{c_d} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{37}$$

Where $\boldsymbol{A_d}$ is a $6x6$ matrix, $\boldsymbol{B_d}$ is a $6x2$ matrix and $\boldsymbol{A_d}$ is a $5x6$ matrix.

5.3 Implementation

We are going to use Kalman filter in discrete time, and this consists of two steps. The first is the prediction step which produces estimates for the current states, containing their uncertainties. The next step is to correct the estimates by using a weighted average where more weight is given to the estimates with higher certainty. The correction step is given by these equations:

$$\boldsymbol{K}[k] = \bar{\boldsymbol{P}}[k]\boldsymbol{C_d}^\top(\boldsymbol{C_d}\bar{\boldsymbol{P}}[k]\boldsymbol{C_d}^\top + \boldsymbol{R_d})^{-1} \tag{38}$$

$$\hat{\boldsymbol{x}}[k] = \bar{\boldsymbol{x}}[k] + \boldsymbol{K}[k](y[k] - \boldsymbol{C_d}\bar{\boldsymbol{x}}[k]) \tag{39}$$

$$\hat{\boldsymbol{P}}[k] = (\boldsymbol{I} - \boldsymbol{K}[k]\boldsymbol{C_d})\bar{\boldsymbol{P}}[k](\boldsymbol{I} - \boldsymbol{K}[k]\boldsymbol{C_d})^\top + \boldsymbol{K}[k]\boldsymbol{R_d}\boldsymbol{K}^\top[k] \tag{40}$$

,

Where $\boldsymbol{K}$ is Kalman gain, $\bar{\boldsymbol{x}}$ is predicted state estimate, $\bar{\boldsymbol{P}}$ is error covariance matrix and $\boldsymbol{R_d}$ is stochastic measurement. The prediction step is given by these equations:

$$\bar{\boldsymbol{P}}[k+1] = \boldsymbol{A_d}\hat{\boldsymbol{P}}[k]\boldsymbol{A_d}^\top + \boldsymbol{Q_d} \tag{41}$$

$$\bar{\boldsymbol{x}}[k+1] = \boldsymbol{A_d}\hat{\boldsymbol{x}}[k] + \boldsymbol{B_d}\boldsymbol{u}[k] \tag{42}$$

Where $\hat{\boldsymbol{x}}$ is corrected state estimate, $\hat{\boldsymbol{P}}$ is corrected error covariance and $\boldsymbol{Q_d}$ is stochastic disturbance matrix. These equations are implemented as simulink functions shown in section A.4 in the appendix.

The diagonal elements in the estimate error covariance $\hat{\boldsymbol{P}}$[k] contains the error variance for each state, and the off-diagonal elements contains the error covariance between states. The kalman gain, given in equation 38, is found by minimizing the trace of the estimate error covariance. Our implementation of the Kalman filter is given in figure 56 in the appendix.

5.4 Experimentation

To find a good estimator using the Kalman filter method, we had to experiment with different $\boldsymbol{Q_d}$ matrices. Because $\boldsymbol{Q_d}$ consists of process noise, the values should be quite low and far under 1, but we wanted to test out values which were a little bit larger as well. It is interesting to look at significantly small values, large values and some varying, random values. The $\boldsymbol{Q_d}$ matrices we chose to experiment with are shown in table 4.

Our hypothesis for an infinitely large $\boldsymbol{Q_d}$ is as follows: The estimates would be equal to the measurements. This hypothesis came from the theory that high $\boldsymbol{Q}$ will provide a higher Kalman gain and the filter completely trusts the measurements.

For the zero $\boldsymbol{Q_d}$ we formulated this hypothesis: The signal will be completely deterministic and clear of any noise. We decided to go with this hypothesis because the covariance would be zero which indicates that we do not think that process noise affects our model. Because process noise always will affect physical systems, neglecting it would cause faulty values. Also, the Kalman gain would be low and we will have no emphasis on the measurements and the filter exclusively trusts our model.

Table 4: Kalman

Diagonal of $\boldsymbol{Q_d}$
$Q_{d1} = \begin{bmatrix}120 & 200 & 3000 & 5600 & 66000 & 66000\end{bmatrix}$
$Q_{d2} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$
$Q_{d3} = \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}$
$Q_{d4} = \begin{bmatrix}1 & 3 & 5 & 7 & 5 & 3\end{bmatrix}$
$Q_{d5} = \begin{bmatrix}0.05 & 0.05 & 0.05 & 0.05 & 0.05 & 0.05\end{bmatrix}$
$Q_{d6} = \begin{bmatrix}0.01 & 0.1 & 1 & 0.1 & 1 & 1\end{bmatrix}*10^{-4}$

When plotting we saw a initial spike in several of the Kalman states. We realized that this was because the Kalman filter needed an initial state when producing the first prediction and correction step. From observation and calculation we got these initial values:

$$\boldsymbol{P_{init}} = \begin{bmatrix} 0.0122 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0114 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0234 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0151 & 0 & 0 \\ 0 & 0 & 0 & 0 & 63.6004 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01646 \end{bmatrix} \tag{43}$$

$$\boldsymbol{x_{init}} = \begin{bmatrix} 0 \\ 0 \\ -0.54 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{44}$$

Before testing out any of the other $\boldsymbol{Q_d}$ matrices, we tried to predict what kind of response we would see from each of them. We started out with the identity matrix and wanted to see what happened when we increased and decreased the values. From theory we predict that $\boldsymbol{Q_d}$ with higher magnitude than the identity matrix, like $\boldsymbol{Q_{d4}}$ would cause similar response as the infinity matrix. This is because the process noise of the system is likely to be far under 1, and values over 1 would be hundreds maybe thousands times larger than the actual noise. We think that this might happen for the identity matrix, $\boldsymbol{Q_{d3}}$, as well.

As mentioned, we want $\boldsymbol{Q_d}$ to be small to get a system with more precise and less varying estimates. However, we did not know exactly how small. We tested out hundredth values, $\boldsymbol{Q_{d5}}$ to see if this was sufficiently small. From the result of this we found that we needed significantly smaller values, and scaled it down further. This resulted in $\boldsymbol{Q_{d6}}$. The result of the experimentation is plotted in figures 39 to 50.

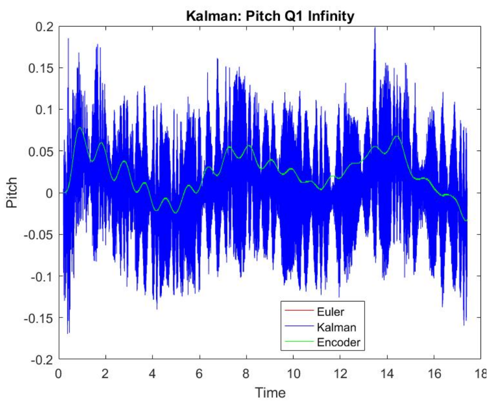


Figure 39: Kalman Infinity: Pitch Angle

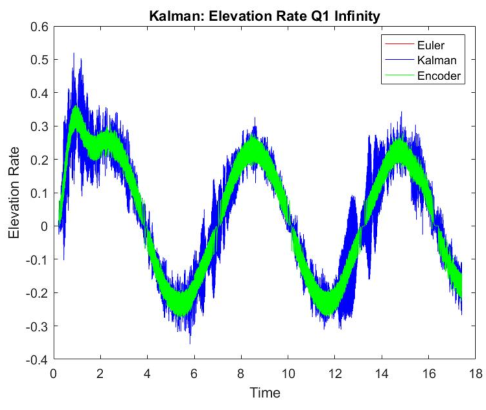


Figure 40: Kalman Filter Infinity: Elevation Rate Infinity

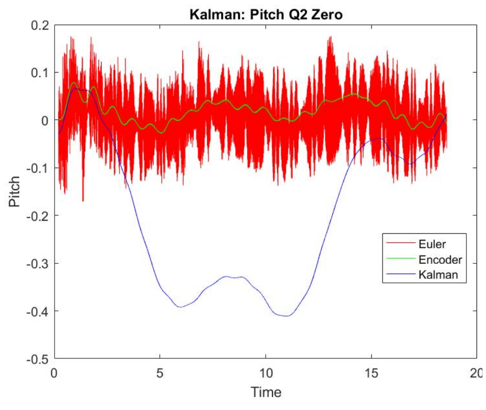


Figure 41: Kalman Filter Zero: Pitch Angle

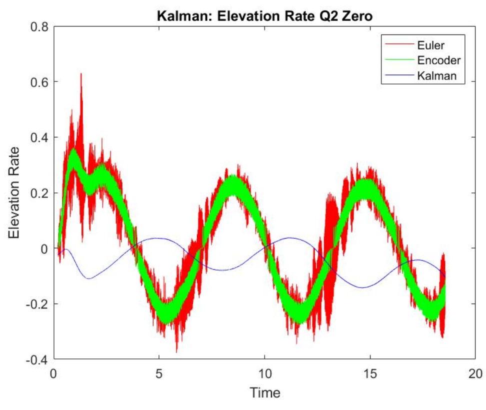


Figure 42: Kalman Filter Zero: Elevation Rate

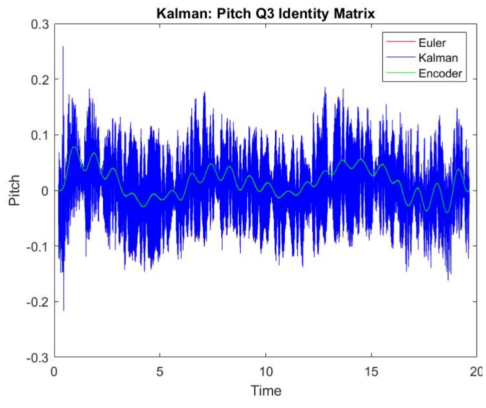


Figure 43: Kalman Filter Q3: Pitch Angle

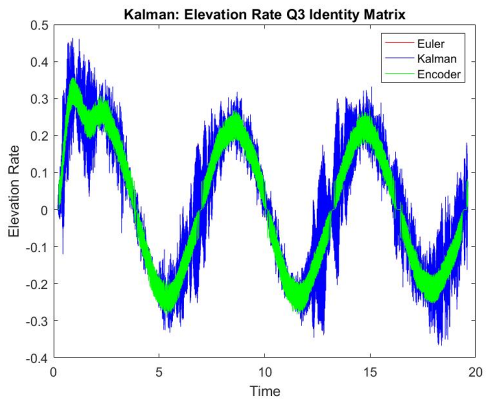


Figure 44: Kalman Filter Q3: Elevation Rate

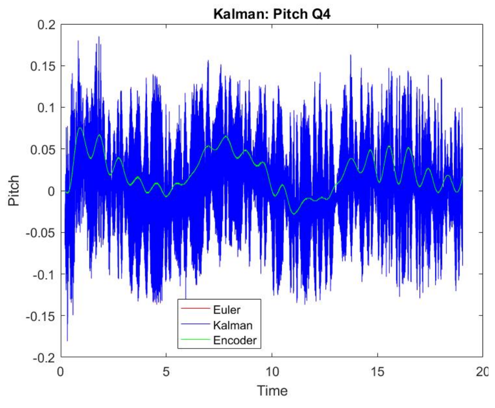


Figure 45: Kalman Filter Q4: Pitch Angle

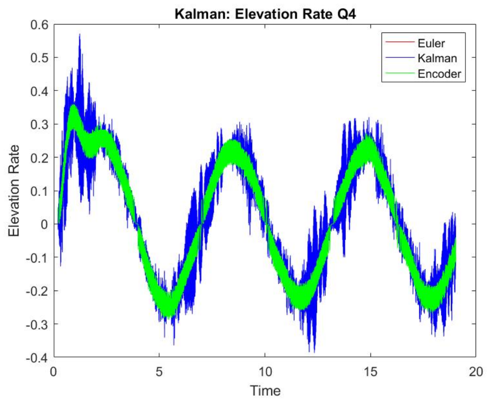


Figure 46: Kalman Filter Q4: Elevation Rate

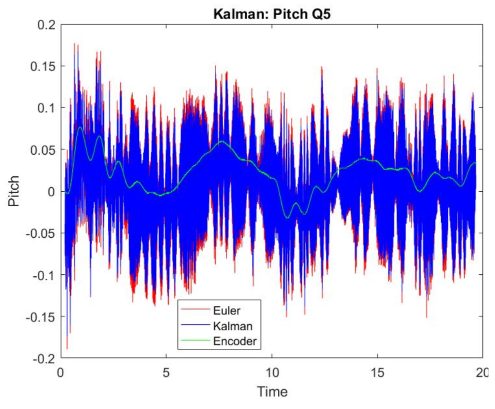


Figure 47: Kalman Filter Q5: Pitch Angle

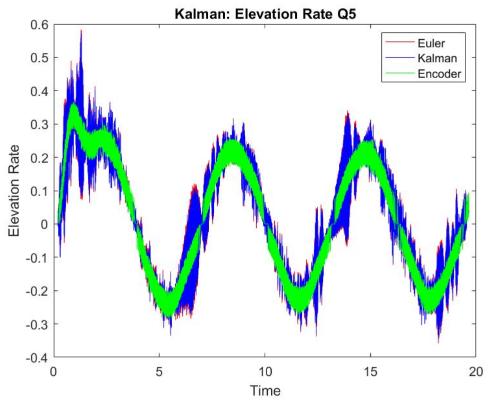


Figure 48: Kalman Filter Q5: Elevation Rate

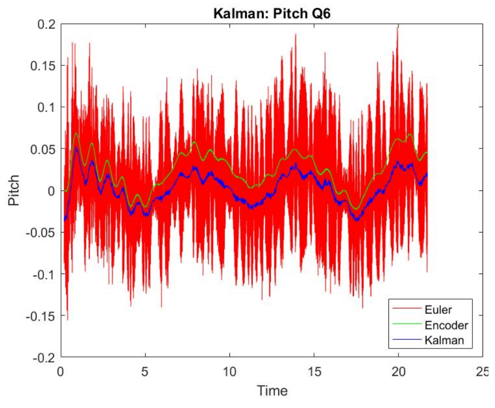


Figure 49: Kalman Filter Q6: Pitch Angle

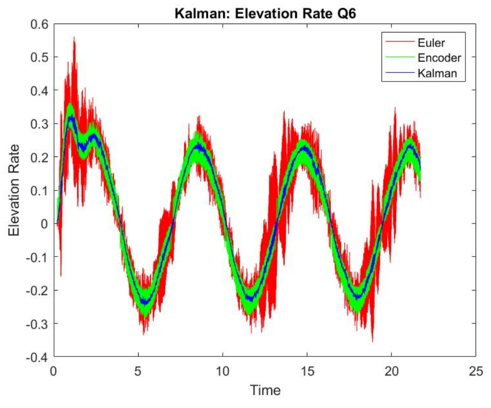


Figure 50: Kalman Filter Q6: Elevation Rate

We observe in figures 39 and 40 that our hypothesis for the infinity matrix in fact was correct. The estimates were exactly the same as the IMU measurements. As mentioned, this is because the covariance is infinite and we completely trust the measurements. Therefore, it covers the whole area and in theory, the helicopter can be anywhere. This is not desired response because we want estimates that reduces the noise of the measurements and gives us a more exact signal. Therefore, this Kalman filter with infinite $\boldsymbol{Q_d}$ is not suited for our use.

Furthermore, looking at figures 41 and 42 validates our hypothesis for zero $\boldsymbol{Q_d}$. The estimates are completely of and the correction step will only weight the model. When this happens, the helicopter will think it is in the perfect spot and there will be bad tracing of the signal. Consequently, once one of the estimates are off the rail, it will just keep getting more of. Hence, we will not use this kind of Kalman filter.

As discussed, the responses of $\boldsymbol{Q_{d4}}$ and $\boldsymbol{Q_{d5}}$ looks similar to the infinity matrix. We realize that this is because the actual process noise is significantly smaller than 1, therefore 1 behaves as a large value. For this reason we will not operate with these Kalman filters either.

The remaining observers are $\boldsymbol{Q_{d5}}$ and $\boldsymbol{Q_{d6}}$, but as we observe in figures 47 and 48 values of hundredths were insufficient. They gave a better estimate than values over 1, but still generated a way too large estimate, hence $\boldsymbol{Q_{d5}}$ is unsatisfactory. Therefore, we came to the conclusion that $\boldsymbol{Q_{d6}}$ gave us the best Kalman filter, plotted in figures 49 and 50. The small value of $\boldsymbol{Q_d}$ indicates that we trust our model and indicates that the actual motion of the helicopter barely deviates from the assumed motion model. Therefore these values gave us an estimation that looked quite similar to the encoder values, apart from the small offset that we should have included. Also, this estimation has little noise, and provides us with a signal that is more exact and will work better in a physical system.

Because of our implementation, the Kalman filter is dependent on new data arriving continuously. By adding a switch we can see what happens to the signal if there is no new data received. The results are shown in 51 and 52.

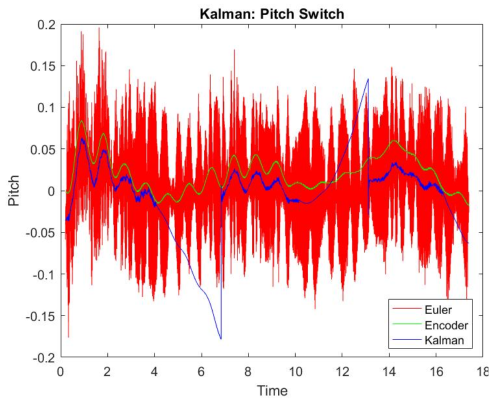


Figure 51: Pitch Angle when using Switch

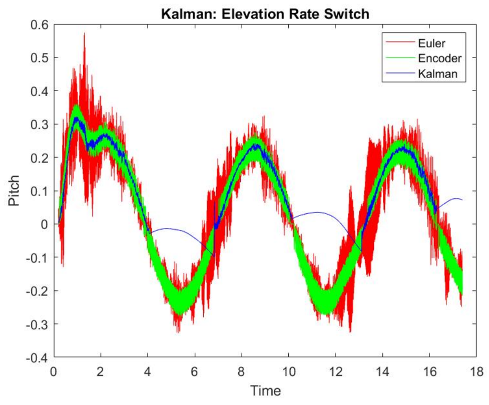
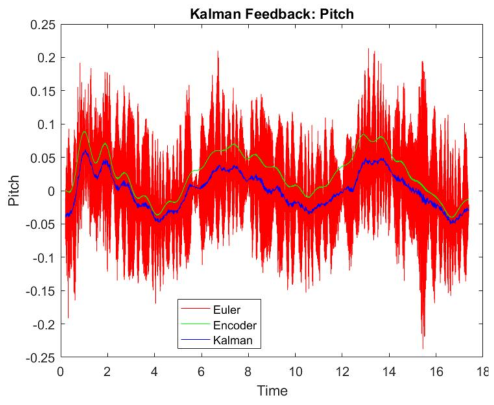


Figure 52: Elevation Rate when using Switch

We can see that when there is no new data the system will constantly go in the direction it was going when new data stopped arriving. The reason why this is, is that when no new measurements are available the corrected state estimate and error covariance are equal to the corresponding prediction. This implies that the correction step, which weights the predicted states against the weighted average of the measurements, is turned of. This causes the system to only use the past predicted states to predict the next, without validating the result with the measurements. The result is a prediction that sums up the errors and causes the values to diverge, as illustrated in figures 51 and 52.

5.5 Tuning

Until now the feedback was given by the encoder values, but we wanted our system to use the Kalman filter estimates instead. We found that the tuning we had for the encoder feedback worked just as well for the Kalman feedback. This is expected because we can observe from previous figures that the encoder values and estimated values looks very similar with our tuning. However, we did observe that the encoder feedback had a somewhat faster response, and was a bit more accurate. The result of the tuning is shown in figures 53 and 54.



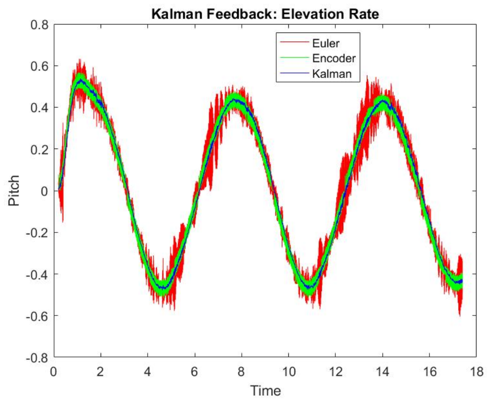


Figure 53: Pitch Angle Kalman Feedback

Figure 54: Elevation Rate Kalman Feedback

As mention for the Luenberger observer, a feedback from the Kalman filter, will make the estimated values be calculated base on our previous estimated values rather than the encoder values. Thus we cannot trust our estimate as much as we did when we had a feedback of encoder measurements.

5.6 Kalman versus Luenberger

One of the differences between the Kalman filter and the Luenberger observer is that the Kalman filter models and considers the noise when choosing how much to trust the measurement and the prediction with noise matrices $\boldsymbol{R_d}$ and $\boldsymbol{Q_d}$. This is a positive property because all physical systems are affected by noise, and we will get more accurate estimates for the states.

Moreover, what separates the Kalman filter from other linear observer is that i minimizes the covariance estimate to find an optimal timedependent $\boldsymbol{K}[k]$. Whilst, in Luenberger we had to use pole placement to find the gain $\boldsymbol{L}$. Since pole placement is a challenging method, the Kalman filter is more practical to work with, because it gives us an already optimized gain.

6 Conclusion

The result from this lab is a functioning helicopter which we are able to fly smooth and controlled. There are several ways we could have improved the stability and controll of the helicopter. If we had a better understanding of LQR before the lab we could have chosen our $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices more wisely and would have spent less time on tuning and more on perfecting the luenberger observer and Kalman filter. Our LQR with integral effect is far from perfect, but from endless hours at the lab and from writing this rapport we have an understanding of which elements in the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrix we have to change to tune the helicopter.

A MATLAB Code

This section should contain our MATLAB code

A.1 Inital values

```
Joystick_gain_x = 1;
Joystick_gain_y = -1;


%%%%%%%%%%% Physical constants
g = 9.81; % gravitational constant [m/s^2]
l_c = 0.46; % distance elevation axis to counterweight [m]
l_h = 0.66; % distance elevation axis to helicopter head [m]
l_p = 0.175; % distance pitch axis to motor [m]
m_c = 1.92; % Counterweight mass [kg]
m_p = 0.72; % Motor mass [kg]
V_s0 = 7.5; %stable voltate when e=0 [V]

%Task 1.1 Mathematical modeling %
K_f = (-l_c*m_c*g + 2*l_h*m_p*g)/(l_h*V_s0); %Motor force constant
J_p = 2*m_p*(l_p)^2;
J_e = m_c*(l_c)^2 + 2*m_p*(l_h)^2;
J_l = m_c*(l_c)^2 + 2*m_p*((l_h)^2 + (l_p)^2);
K_1 = K_f*l_p/J_p;
K_2 = l_h*K_f/J_e;
K_3 =l_h*K_f/J_l;
lambda_1 = -1 + 1i;
lambda_2 = -1 - 1i;
K_pd = -(lambda_1 + lambda_2)/K_1;
K_pp = lambda_1*lambda_2/K_1;
```

A.2 LQR

```
% Pole placement %
A = [0 1 0 ;
      0 0 0 ;
      0 0 0];

B = [0 0;
      0 K_1;
      K_2 0];

CT = [B A*B A*A*B];
```

```

C = [1 0 0; 0 0 1];

pole = [-1.2 -1.3 -1.4];

%K = place(A, B, pole);

% LQR without integral effect %

Q = [ 0.3 0 0;
      0 0.2 0;
      0 0 0.2];


R = [1 0;
     0 1];
K = lqr(A, B, Q, R);

F = inv([C*inv(B*K - A)*B]);

%LQR with integral effect

A_i = [0 1 0 0 0;
       0 0 0 0 0;
       0 0 0 0 0;
       1 0 0 0 0;
       0  0 1 0 0];

B_i = [0 0;
       0 K_1;
       K_2 0;
       0 0;
       0 0];

CT = [B_i A_i*B_i A_i*A_i*B_i];

C_i = [1 0 0 0 0; 0 0 1 0 0];

G_i = [0 0;
         0 0;
         0 0;
        -1 0;
        0 -1];

```

```
Q_i = [ 0.1 0 0 0 0;
         0 5 0 0 0;
         0 0 10 0 0;
         0 0 0 0.8 0;
         0 0 0 0 0.8];

R_i = [1 0;
       0 0.5];


K_i = lqr(A_i, B_i, Q_i, R_i);

F_i = inv([C_i*inv(B_i*K_i - A_i)*B_i]);
% %F_i = [K_i(1,1) K_i(1,3);
%         K_i(2,1) K_i(2,3)];
% F_i = [0 1;
%         1 0];
```

A.3 Luenberger observer

```
% Pitch function %
function p = pitchFcn(ay, az)
    if az == 0
        p = 0;
    else
        p = atan(ay/az);
    end
end

%Elevation function %
function e = ElevFcn(ax, ay, az)
    if (sqrt(ay.^2 + az.^2) == 0)
        e = 0;
    else
        e = atan(ax /sqrt(ay.^2 + az.^2));
    end
end
```

A.4 Kalman filter

```
%Corretion step function %
function [x_hat, P_hat] = correction_step(x, P, y, new_data, Cd, R_d)

    if new_data
        Kalman_K = P*Cd'*(Cd*P*Cd' +R_d)^(-1);
        x_hat = x + Kalman_K*(y-Cd*x);
        n = size(Kalman_K*Cd, 1);
        P_hat = (eye(n)-Kalman_K*Cd)*P*(eye(n)-Kalman_K*Cd)
                + Kalman_K*R_d*Kalman_K';
    else
        x_hat = x;
        P_hat = P;
    end


%Kalman prediction function%
function [x_next, P_next]
        = prediction_step(x_hat, P_hat, u, Ad, Bd, Q_d)
    x_next = Ad*x_hat +Bd*u;
    P_next = Ad*P_hat*Ad' + Q_d;
end
```

B Simulink Diagrams

This section should contain our Simulink diagrams.

B.1 Simulink Diagram for the full system

Figure 55 shows the Simulink diagram for the helicopter.

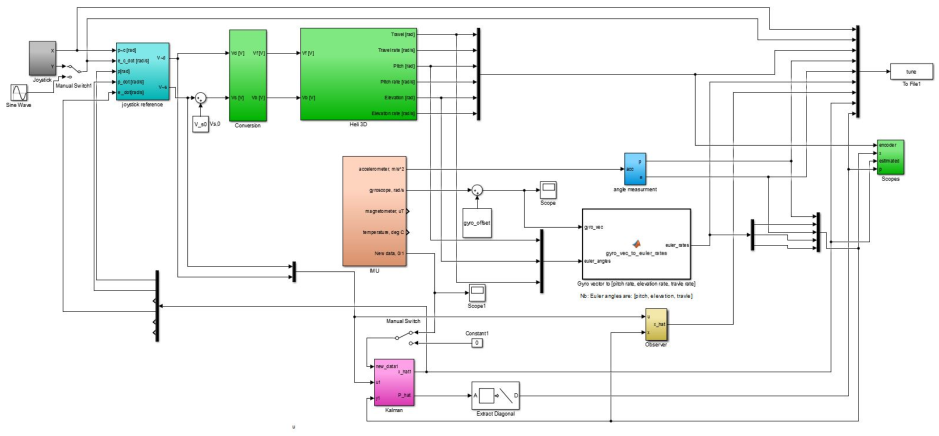


Figure 55: The full system with Kalman Filter.

B.2 Simulink Diagram for the estimated state

Figure 56 shows the Simulink implementation of the estimated state from the IMU measurements.

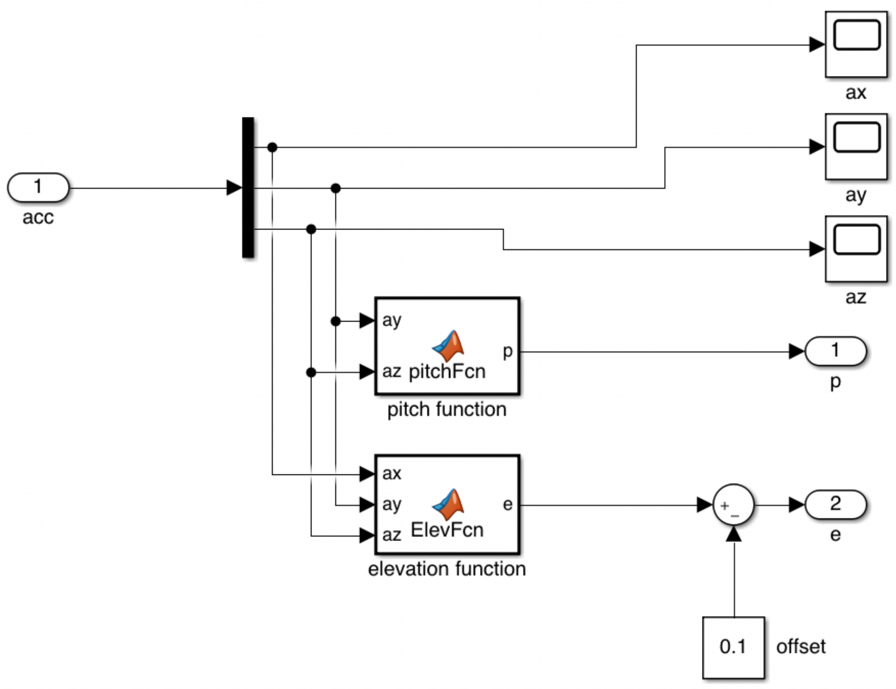


Figure 56: Estimated values

B.3 Simulink Diagram for the LQRI

Figure 57 shows the Simulink implementation of the LQR with integral effect

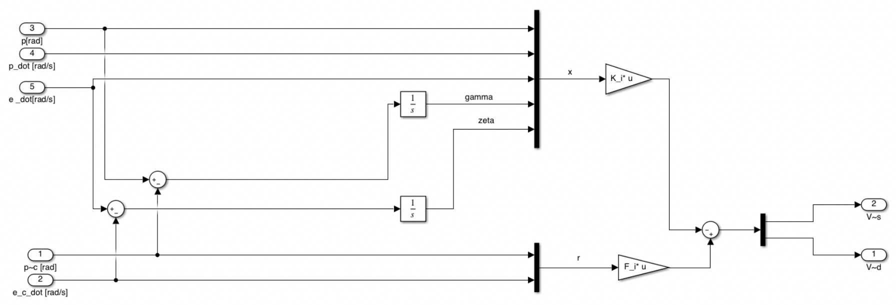


Figure 57: LQR with integral

B.4 Simulink diagram for the Luenberger observer

Figure 58 shows the Simulink implementation of the luenberger observer

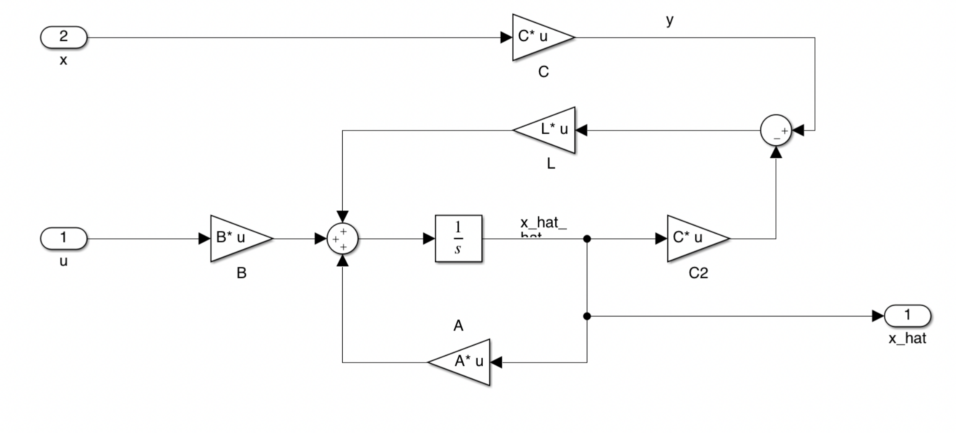


Figure 58: Luenberger

B.5 Simulink Diagram for the Kalman Filter

Figure 59 shows the Simulink implementation for the Kalman filter

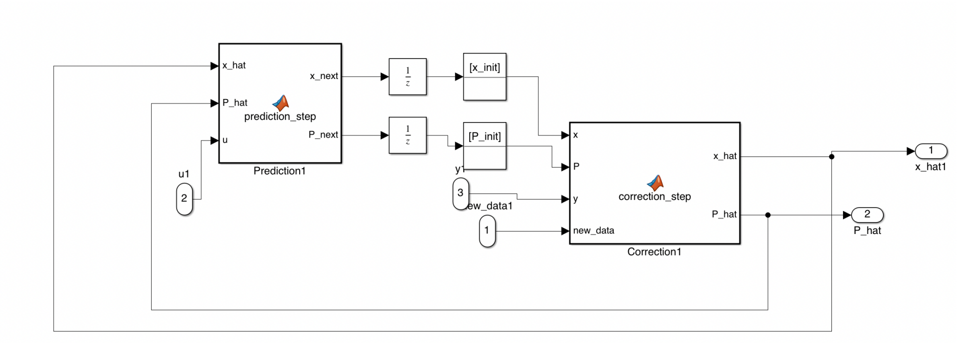


Figure 59: Kalman filter

References

[1] Balchen et. al. *Reguleringsteknikk*. NTNU Grafisk senter, 2016.

[2] Hwang P. Y Brown R. G. *Introduction to random signals and applied kalman filtering*. John Wiely: sons, Inc, 2012.

[3] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, Incorporated, 2014.