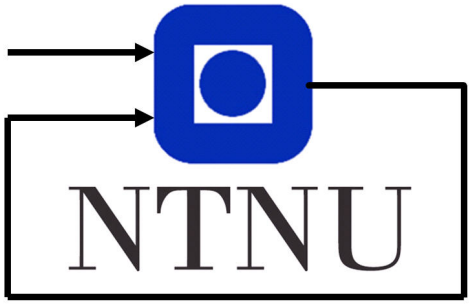
Helicopter Lab Report

TTK4115 – Linear System Theory

Group 68
Sandra Garder Løkken 507748
Andrea Schnell 507752

September 1, 2020



Department of Engineering Cybernetics

Contents

1 Part 1 - Monovariable control

1.1 Task 1 - Manual control

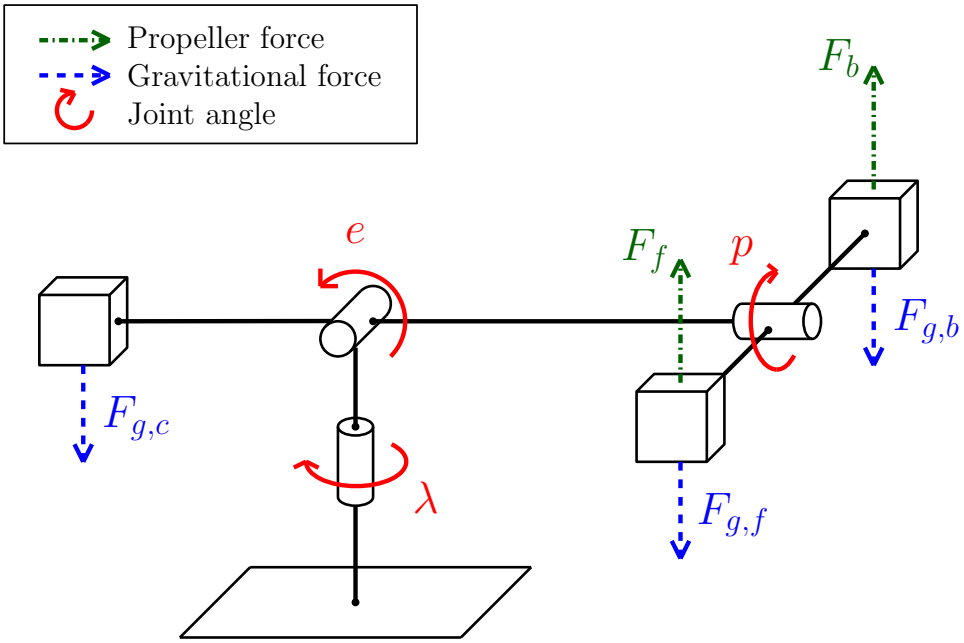


Figure 1: Helicopter model

Task 1 was to set up a manual control of the helicopter(Figure 34 and 35). This was done by using feedforward, and controlling it with the joystick. The x-axis signal from the joystick was connected to the voltage difference between the two propellers, V_d. The signal from the y-axis was connected to the voltage sum, V_s. By doing this, the helicopter would move up and down if the joystick moved along the y-axis because of the increase and decrease of the voltages to the two propellers, and tilt and move sideways if the joystick moved along the x-axis.

The voltage signal from the joystick itself was to small to move the helicopter, so it was necessary to include a gain block on each of the input signals. The signal from the y-axis to V_s needed to be multiplied by 9 to be able to lift the helicopter, while the signal from the x-axis to V_d only needed to be multiplied by 2. We obtained these values by testing with different gains, and saw that these worked the best when moving the helicopter.

The small gain to V_d was to ensure that the voltage difference did not change too fast since the motors only needed a small voltage difference to affect the pitch angle, p, of the helicopter. Increasing the gain of the voltage difference made the helicopter harder to control manually.

1.2 Task 2 - Parameter identification

By manually moving the helicopter and scoping the e, p and λ values, we identified that the positive directions given by the encoders matched the one defined in the task. Since the encoder values are set to zero when Simulink is connected to the helicopter, we initially have e equal to zero when the helicopter is laying still. We wanted e to represent the elevation angle, which is zero when the arm between the elevation axis and the helicopter head is horizontal. To ensure this, we scoped the elevation and saw that the difference between laying still and horizontal was 0.5 rad. We therefore subtracted 0.5 from the encoder value of e to get correct results for the elevation angle.

In the lab preparation we derived a linearized model, Equation 2, for the system showed in Figure 1 by using the variable transformation $\tilde{V_s} = V_s - V_{s,0}$. By elevating the helicopter with the joystick and scoping V_s, we found that the voltage value at elevation angle zero were approximately 8.5V. Setting this voltage as $V_{s,0}$, we obtained the equilibrium $e = 0$ as wanted. An equation found in the lab preparations gave the expression for K_f,

$$K_f = \frac{2m_p g l_h - m_c g l_c}{V_{s,0} l_h}. \tag{1}$$

This was calculated and used during the rest of the lab.

$$\ddot{p} = \frac{l_p K_f}{(2m_p m_p^2)} V_d \tag{2a}$$

$$\ddot{e} = \frac{K_f l_h}{(m_c l_c^2 + 2m_p l_h^2)} \tilde{V_s} \tag{2b}$$

$$\ddot{\lambda} = \frac{-2(m_c g l_c - 2m_p g l_h)}{m_c l_c^2 + 2m_p (l_h^2 + l_p^2)} p \tag{2c}$$

1.3 Task 3 - PD control

In the lab preparation we developed a PD controller for the pitch angel p, given Equation 3a. When we substituted this equation into Equation 2a and applied the Laplace transform, we ended up with a transfer function from p to p_c. From this we were able to develop two functions representing the systems poles, Equation 3b and Equation 3c.

The placement of the poles affects the stability of the system. In order to find the best poles for our system, we had to experiment with different pole placements. This was done by using a pulse generator as pitch reference, p_c, and plotting the achieved responses for the different values of the poles.

Table 1: Pole placement in Figure 5

Pole	Color	λ_1	λ_2
Complex conjugated	Green	$-2.2 + 1.05i$	$-2.2 - 1.05i$
Coinciding poles	Blue	-2.8	-2.8
Real poles	Red	-2	-3

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p} \tag{3a}$$

$$K_{pp} = \frac{\lambda_1 \lambda_2}{K_1} \tag{3b}$$

$$K_{pd} = -\frac{\lambda_1 + \lambda_2}{K_1} \tag{3c}$$

A problem that occurred when testing, was a drop in elevation when the helicopter had a large pitch to either side. This is caused by the force from the propellers working more horizontally than vertically, and consequently not enough force is provided to hold the helicopter at the reference elevation. A drop in elevation when there is a large pitch can be seen in several plots throughout the report, for example Figure 7.

Figure 2, Figure 3 and Figure 4 shows the responses of the system using different type of pole placements, compared with the pulse signal. Figure 5 compares the best results from each of the pole types. The values that gave the best responses are shown in Table 1. From the plots, we saw that coinciding poles on the real axis gave the fastest and most stable response.

There are three different types of responses; over-, under and critically damped [1, p. 141], where the latter is usually the one desired. The response type is determined by the pole placement. Each pole has a certain angle, θ, from the negative x-axis and a distance, r, from origin. The combination of these results in different pole types, and the combination of poles results in different responses. Since systems with poles in the right half plane are unstable, we only tested with poles in the left half plane.

Figure 4 shows the response of the system with different complex conjugated poles. These kind of poles leads to an underdamped response. The different values are to show how a system behaves differently for different complex conjugated poles. Large θ gives the system greater overshoot, while larger r means faster response. The response with $\lambda_{1,2} = -1 \pm 2i$ has a large complex value, which leads to a large theta. This is shown by the overshoot and oscillation in the red line in the plot. The green line in Figure 4 has oscillations due to the large real number. The best response with complex conjugated poles were with $\lambda_{1,2} = -2.2 \pm 1.05$. This gave a fast response, but without the overshoot and oscillation.

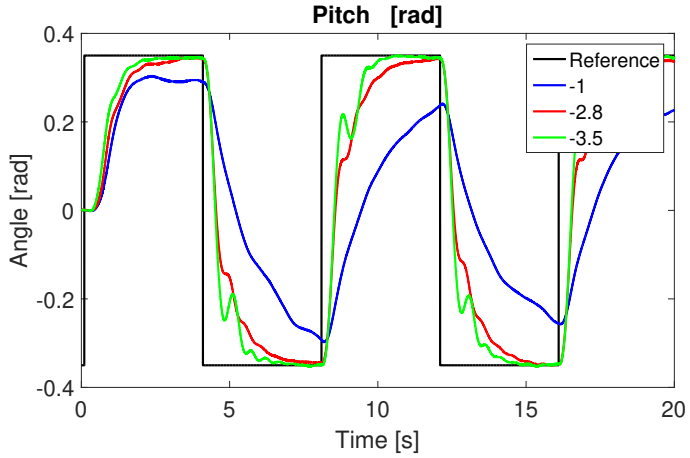


Figure 2: The response from coinciding real poles

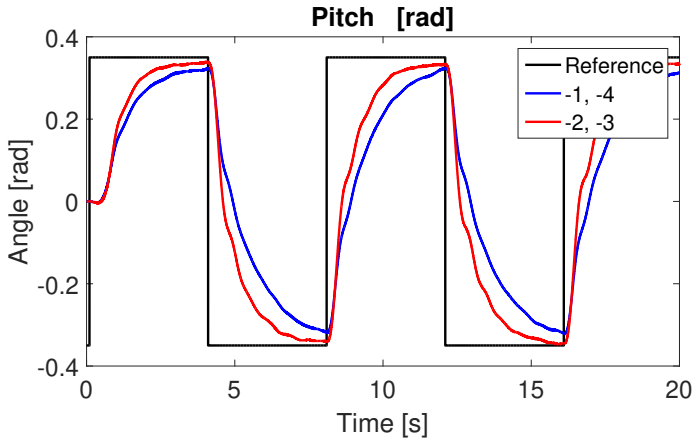


Figure 3: The response from different, real poles

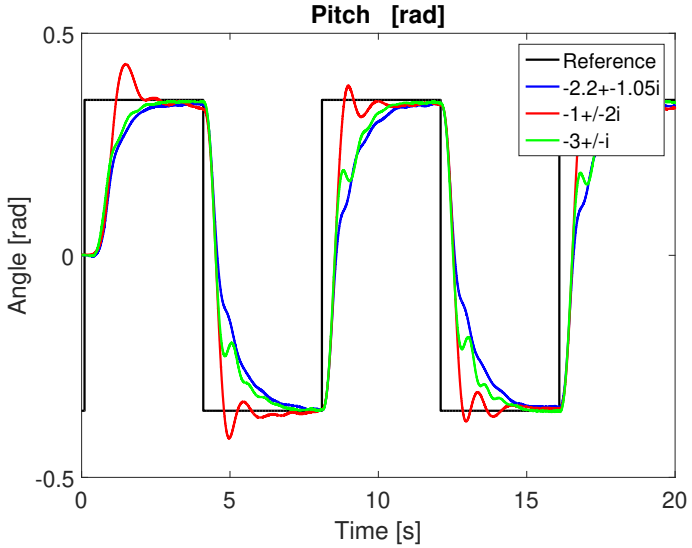


Figure 4: The response from complex conjugated poles

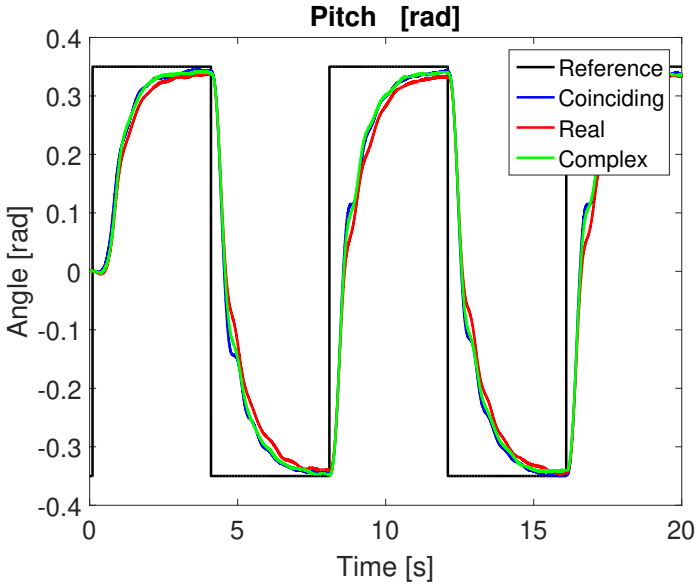


Figure 5: The best response from each pole type

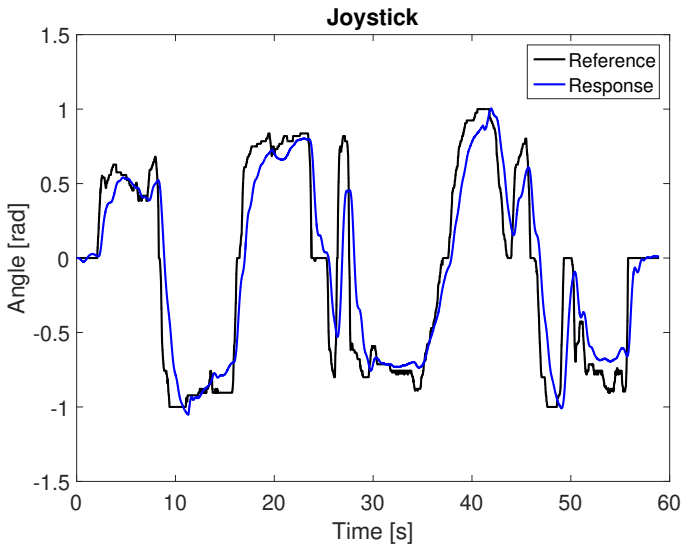


Figure 6: The pitch response when using coinciding real poles, with joystick as reference

For real, different poles the response will be overdamped, resulting in a slow, but non-oscillating response. Figure 3 shows this type of response. As expected, it is the slowest of the responses, as well as having no oscillations. Poles located close to each other will result in a faster response, which is possible to see from the figure.

According to the theory, coinciding real poles, i.e. $\theta = 0$, results in a critically damped response. This is the fastest response without oscillations. For poles placed further to the left, the response will be faster. However, because of physical restrictions in the helicopter motors, poles placed far left will saturate the motor actuation and not give the desired response. This is reflected in our result, where the pole pair with the highest absolute value, the green line in Figure 2, gave oscillations. For poles placed close to origin, the response will be slow, as the test with poles in -1 (the blue line) showed. The best response was at $\lambda_{1,2} = -2.8$, shown with the red line in Figure 2.

The best results of the three different pole placement types are plotted in Figure 5 to comparison, with pole values as described in Table 1. It shows that the coinciding, real poles gave a slightly faster response than the rest. Figure 6 shows the response with coinciding poles when using the joystick as reference pitch p_c. This shows that it was possible to use the joystick to control the helicopter with the tuned PD regulator .

When linearizing a system, the solution will be approximately equal to the nonlinear system in proximity of a chosen linearization point, in this case equilibrium point $e = p = \lambda = 0$. When moving sufficiently far from this point, the approximation will be increasingly inaccurate. In Figure 6, there

are more inaccuracy in the extremes of the reference graph. This can be explained by the distance from the linearization point.

2 Part 2 - Multivariable control

2.1 Task 1 - Implementation

The first task was to implement the controller, Equation 5, that we developed in the lab preparation in simulink. The implementation is shown in Figure 36 and Figure 37. We let the x-axis of the joystick be the pitch reference, and the y-axsis be the elevation rate reference.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{4a}$$

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \ddot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix} \mathbf{u} \tag{4b}$$

$$\mathbf{u} = \mathbf{F}\mathbf{r} - \mathbf{K}\mathbf{x} \tag{5a}$$

$$\mathbf{u} = \begin{bmatrix} k_{11} & k_{13} \\ k_{21} & k_{23} \end{bmatrix} \begin{bmatrix} p_c \\ \dot{e}_c \end{bmatrix} + \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} \tag{5b}$$

In the preparation work we examined the controllability of the system, and concluded that the system was controllable. This means that with the right sequence of inputs, it is capable of moving any state space to any other state in a finite time [3, s. 144].

2.2 Task 2 - Pole placement

The next task was to use pole placement to design the state-feedback matrix, **K**. This was done by setting `K=place(A,B,p)`, where `place()` is a Matlab function that generates the matrix, making the system poles the elements of **p**. **A** and **B** are the matrices, Equation 4b, developed in the lab preparations.

The intention of the controller is for the response $\mathbf{y}(t) \to \mathbf{r}$ when $t \to \infty$. When we varied the value of the poles in **p**, we got different values in the matrix **K**. This matrix does not only affect the feedback gain but also the value of the feedforwad gain **F**, as stated in Equation 5b. Different values of pole placements does therefore affect the behavior of the system by changing the input value, **u**.

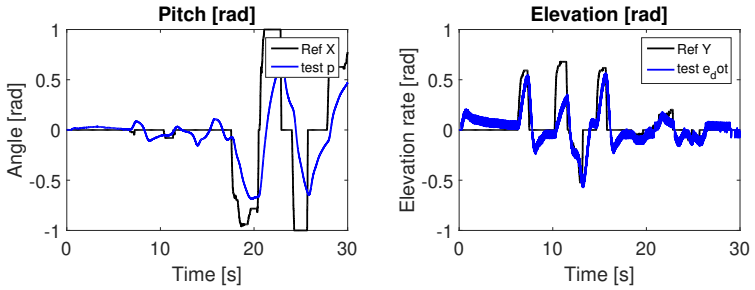
When experimenting with different pole values, we noticed that the system behaved worse for larger values. That is, when we tested with large pole-values, the pitch were unstable and the response had overshoot. This type of response could probably be caused by saturation of the signal. Meaning we had reached the systems physical limit, hence the unstable behaviour. Because the pitch was unstable, it was difficult to examine the behaviour of the elevation rate. The behaviour of the system depends both on the pitch

and elevation rate. Therefore, when the pitch was unstable because of the high values of the poles, it affected the other states and made the whole system unstable, and there was no way to control the elevation rate.

When we tested with small pole-values, the response were slow, but stable. We noticed a stationary deviation in both pitch and elevation rate. The best response we were able to achieve with this system were with small pole-values, $\mathbf{p} = \begin{bmatrix} -2 & -3 & -1 \end{bmatrix}$, resulting in:

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 24.46 \\ 5.90 & 7.87 & 0 \end{bmatrix}.$$

The response is shown in Figure 7.



(a) Pitch (b) Elevation rate

Figure 7: Response in task 2.2 with joystick as reference

2.3 Task 3 - LQR

In this task, we were to use linear quadratic regulator (LQR) algorithm, instead of pole placement, to design the matrix $\mathbf{K}$. This was done by using Matlab command `K=lqr(A, B, Q, R)`. Where matrices A and B were the same as in task 2, and with $\mathbf{Q}$ and $\mathbf{R}$ as diagonal weighting matrices. We then experimented with different values of $\mathbf{Q}$ and $\mathbf{R}$ to see how the system behavior varied with the different values.

$$J = \int_0^\infty \mathbf{x}^\mathbf{T}(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^\mathbf{T}(t)\mathbf{R}\mathbf{u}(t)dt \tag{6}$$

For optimal control with LQR, the cost equation 6 have to be minimized. Tuning is done by varying the weighting matrices, $\mathbf{Q}$ and $\mathbf{R}$. These matrices are on the forms

$$\mathbf{Q} = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \textit{and } \mathbf{R} = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix}.$$

The relative size of $\mathbf{Q}$ and $\mathbf{R}$ decide the tradeoff between the control action and how fast the system converges. Each of the elements q_i and r_i

represents the weighting to respectively x_i and u_i, e.g. if the response of the pitch angle is slow, increasing q_1 will improve it.

When testing the system, we started out with identity matrices for both $\mathbf{Q}$ and $\mathbf{R}$. We found that q_i and r_i corresponded to x_i and u_i. For each element, we increased or decreased the value until we observed a good response for the corresponding state. We observed that if the value of q_1, corresponding to pitch, were to high compared to the elevation weighting, it made the controlling of pitch very sensitive. Consequently, the elevation became hard to control. Therefore, we wanted elevation rate to have a higher cost than pitch. However, increasing the elevation rate too much resulted in the pitch responding slow.

Choosing too large values of the elements in the matrix $\mathbf{R}$, causes the actuations V_s and V_d to be weighted too much compared to the states, leading to a slow system. Both elements of $\mathbf{R}$ could be decreased, because the value of the control signal is not important compared to how fast the system converge.

The result of the tuning was the matrices

$$\mathbf{Q} = \begin{bmatrix} 45 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 100 \end{bmatrix},$$

$$\mathbf{R} = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix},$$

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 11.9523 \\ 8.0178 & 9.5672 & 0 \end{bmatrix}.$$

2.4 Task 4 - LQR with integral action

In the lab preparations we developed an integral controller with integral effect for the elevation rate and pitch angle. This lead to two additional states γ and ζ, with differential equations $\dot{\gamma} = p - p_c$ and $\dot{\zeta} = \dot{e} - \dot{e_c}$, and a system on the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{r}, \tag{7a}$$

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \ddot{e} \\ \dot{\gamma} \\ \dot{\zeta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \\ \gamma \\ \zeta \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_c \\ 0 \\ \dot{e_c} \\ 0 \\ 0 \end{bmatrix}, \tag{7b}$$

with $\mathbf{u}$ as shown in Equation 5.

When testing, we saw that this made the stationary deviation disappear, and that the weighting followed the same principle as for the other states.

Because of the integral effect, the controller could provide zero steady state error as long as the states were within relative limits from the linearization point. Figure 9 shows an example where the elevation rate were able to reach the reference signal because of the integral action, but since the reference were far from the linearization, the elevation rate decreased again after a short time. Despite this, the LQR system with integral action responded much better than the system without this effect, which is especially shown in Figure 8, where the system had a great deviation before introducing the integral effect.

In the lab preparations, we derived that the states will converge to the reference independently of the values of $\mathbf{F}$. When testing this in the lab, we came to the same conclusion. The difference in the results were in context of how the states converged to the reference. Using smaller values of $\mathbf{F}$ resulted in a slower convergence, while lager values resulted in a faster response but with overshoot. Therefore we concluded that the $\mathbf{F}$ used in the LQR system without integral effect was the one resulting in the best response.

2.5 Task 5 - tuning

The last task was to tune in a good response from the LQR system with integral action, Equation 7. The tuning was based on observations made during the execution of the previous tasks through experimentation with different values for the two additional diagonal elements in Q. At last we ended up with:

$$\mathbf{Q} = \begin{bmatrix} 45 & 0 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 6 \end{bmatrix}, \tag{8a}$$

$$\mathbf{R} = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}, \tag{8b}$$

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 14.8623 & 0 & 2.9277 \\ 12.2915 & 10.4815 & 0 & 4.1404 & 0 \end{bmatrix}. \tag{8c}$$

The tuned responses with and without integral action is shown in Figure 8 and Figure 9.

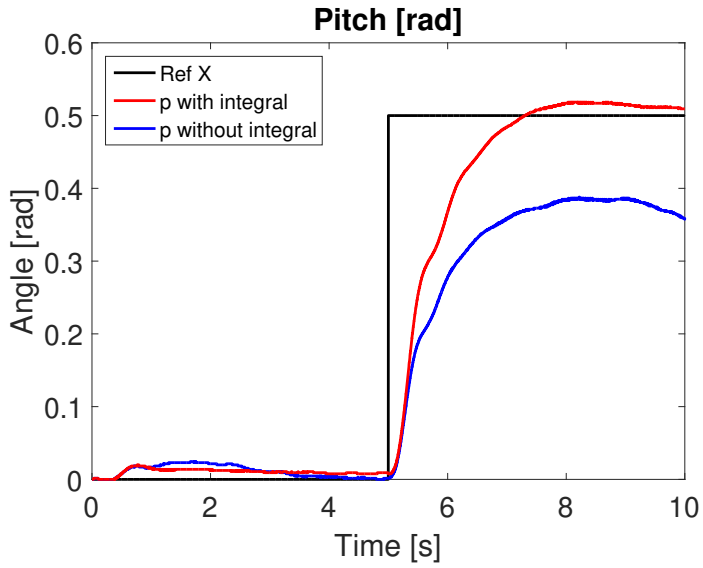


Figure 8: Comparison of the best tuning of the pitch with LQR

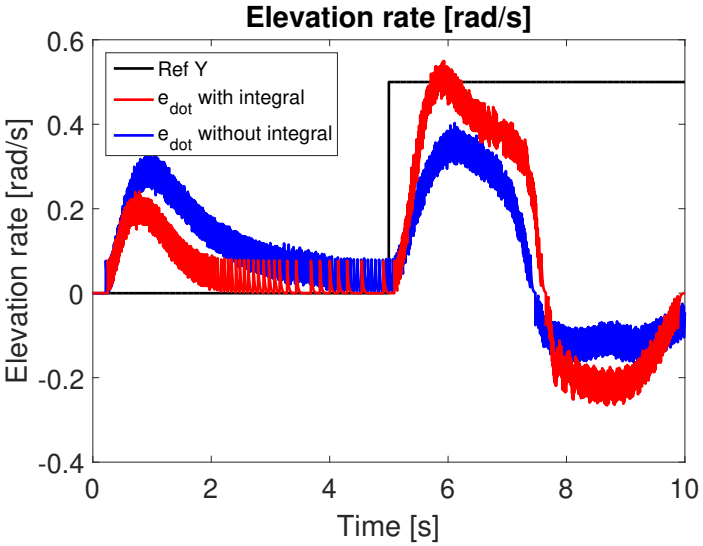


Figure 9: Comparison of the best tuning of the elevation rate with LQR

3 Part 3 - Luenberger Observer

3.1 Task 1 - IMU Characteristics

The encoder used for measurements in the previous labs requires that the helicopter is stuck to the ground. To be able to control a free flying helicopter, the encoder was switch out with an inertial measurement unit(IMU). The first task was to compare the IMU- and encoder-measurements, while manually moving the helicopter. This was done by disconnecting the voltage signals in simulink, and scoping the corresponding measurements from the encoder and IMU.

First we tried rotating one axis at the time. When plotting the encoder rates against the gyroscope rates, we observed that the gyroscope produced more noise than the encoder, as well as having a small offset in the magnitude of centi rad on all of its measurements. The more noisy IMU measurements are more noticeable in plots of the helicopter flying, e.g. Figure 19, while in the plots from manually moving the helicopter, e.g. Figure 12, the difference in noise is not as big. However, the figure shows that the noise increased during rapid changes in the helicopter movements. The fact that the encoder had less noise than the IMU is caused by the IMU being located on the helicopter, experiencing all the movement and noise that the helicopter is experiencing.

The offset on the IMU measurements in all of the tasks, are probably caused by some physical aspects. The IMU is placed on the helicopter, and often has a bias to its measurements that the Arduino tries to calculate and subtract. This is not always accurate, and is probably why there is offsets to the measurements. This was fixed when we compared the Luenberger observer and the Kalman filter in Lab 4, by finding the approximate offset on each measurement and adding the offset such that the IMU and encoder have the same zero.

Apart from this, the measurements from the IMU were about the same as the encoder's. However, when rotating around two axes at the time, one of the three rates measured by the gyroscope had great discrepancies from the encoder. Figure 10, Figure 11 and Figure 12 are the scopes of respectively elevation rate, pitch rate and travel rate when rotating pitch and elevation at the same time, making the gyroscope measure travel rate incorrectly. This error is because the measurements made by the IMU have a different coordinate system (x, y, z) than the coordinate system we use in the rest of the lab (e, p, λ). When rotating about two axis, the coordinate systems will not coincide, which results in a wrong measurement on the third axis.

When observing the accelerometer, we found that the magnitude on a_x, Equation 9a, was greater than the two others. When multiplying sine and cosine, the product will never have an absolute value greater than either one of the factors. a_y and a_z, Equation 9b and Equation 9c, are both dependent

on sin/cos to e and p, making the magnitude of these smaller than a_x. We also observed a_x only being dependent on the elevation. In addition to elevation, a_y and a_z were both dependent on the pitch. This coincides with the equations derived in the preparations, Equation 9. Similar to the gyroscope measurements, the accelerometer reading were noisy, probably caused by the placement of the IMU.

$$a_x = N\sin(e) \tag{9a}$$
$$a_y = N\cos(e)\sin(p) \tag{9b}$$
$$a_z = N\cos(e)\cos(p) \tag{9c}$$

3.2 Task 2 - Transformations

To be able to use the gyro-measurements directly and get correct results, we had to apply a rotation that used the pitch and elevation angles to calculate the correct pitch-, elevation-, and travel-rate. This was done by including a given Simulink block that performs the rotation, done in the `gyro_vec_to_euler_rates` sub system (Figure 38). We connected the encoder angles as inputs to this block.

Next, we implemented the equations for pitch and elevation derived the preparation (equation 10 and 11). This was so we could use the accelerometer measurements to indirectly measure the orientation of the helicopter. Since the program measurements from the IMU during start-up is zero, we had to implement some logic to the sub-system, shown in Figure 39. This was to avoid a division by zero.

$$p = \arctan\left(\frac{a_y}{a_x}\right) \tag{10}$$

$$e = \arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \tag{11}$$

After implementing this, we compared the transformed gyro output with rates from the encoder by manually moving the helicopter around all axis. Additionally, we compared the transformed accelerometer measurements with the elevation and pitch angle from the encoders.

The results of rotating the helicopter manually around all axis, are shown in figure 13, figure 14 and figure 15. We observed that all the measurements from the gyro matched the corresponding measurements from the encoder independent of the number of axes we rotated about. However, there was some discrepancies. As mentioned in task 3.1, there was some offset deviations in the measurements. Shown in figure 13, the elevation rate had a

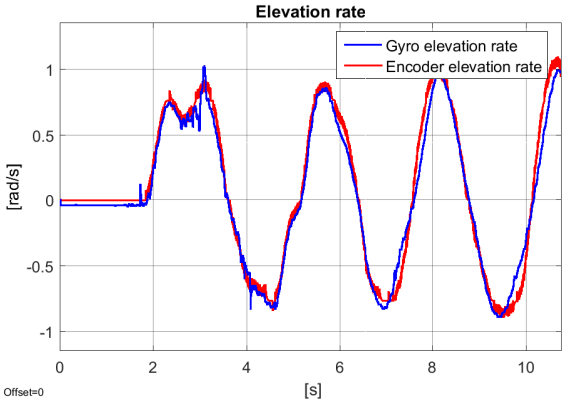


Figure 10: Comparison of encoder and gyroscope measurements of elevation rate when rotating pitch and elevation

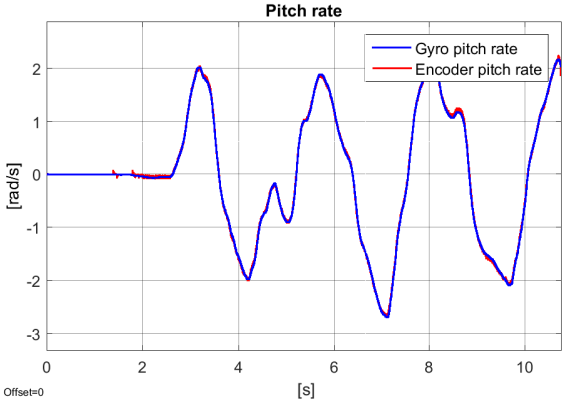


Figure 11: Comparison of encoder and gyroscope measurements of pitch rate when rotating pitch and elevation

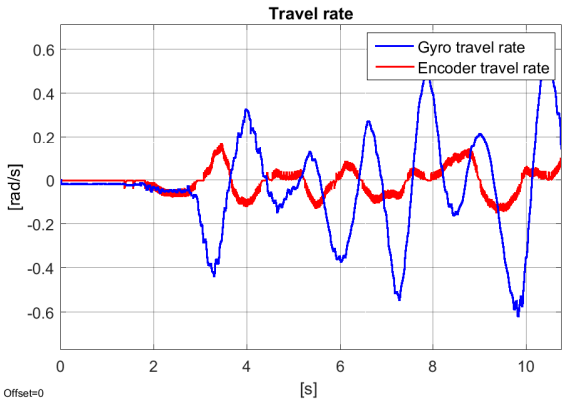


Figure 12: Comparison of encoder and gyroscope measurements of travel rate when rotating pitch and elevation

deviation of approximately -0.04[rad/s] between the gyro and encoder. The travel rate also had a small deviation of -0.02[rad/s], shown in Figure 15. The figures also show that there was noise from the measurements of the elevation- and travel rate, both from the gyroscope and encoder. However, the gyroscope noise was amplified when using the motors instead of moving it manually.

The measurements we got of the elevation and pitch angle from the accelerometer compared to the encoder are shown in figure 16 and figure 17. The signals followed the encoder outputs well, but as we noticed from the gyro measurements, the accelerometer measurements also had an negative offset compared to the encoder on approximately 0.06 and 0.04 for the pitch and elevation angle respectively. We also experienced more noise from the accelerometer measurements, which were expected due to the results in task 3.1. Especially, we experienced more noise in the extremes of the graphs, which we will come back to in the next task.

3.3 Task 3 - Theoretical discussion

A state is said to be observable if the knowledge of the input u and the output y over the interval $[0, t_1]$ suffices to determine uniquely the initial state $x(0)$ [3, p. 153]. When calculating the observability of the system in the preparations, we found that for the system to be observable, we had to at least measure $x_3 = e$ and $x_5 = \dot{\lambda}$. By examining the system matrix $\mathbf{A}$, you see that there is only zeros in the columns 3 and 5. The transfer function exclude all information on unobservable states. Consequently, those two states has to be measured in order to be observable, while the others are already included in the transfer function due to them having values in their corresponding columns.

When using the angle measurements based on the accelerometer readings, there are a few assumptions underlying the conversion from the measurements to the angles. The deriving of the equations assumed that the helicopter was standing still. This means that all behavior different from that, is not taken into account. Especially when there are a lot of movement or vibrations, the angle measurement will be uncertain. This is probably why, in Figure 16 and Figure 17, there are more noise in the extremes of the graph, where there are physically a lot of movement and possible vibrations. In addition, the functions Equation 10 and Equation 11 is derived by dividing on cosine of the angles, which introduce the zero division problematic when the angles is $\pi/2$. We implemented a logic that avoids this problem shown in Figure 39. However in those cases, the measurements is not correct, though, the angles are rarely as much as $\pi/2$.

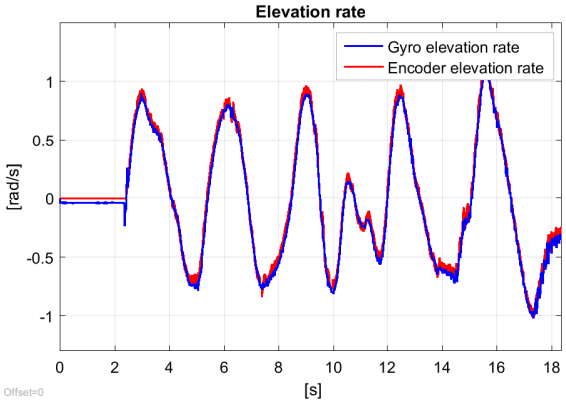


Figure 13: The elevation rate measured from the transformed gyroscope output compared to the rate from the encoder

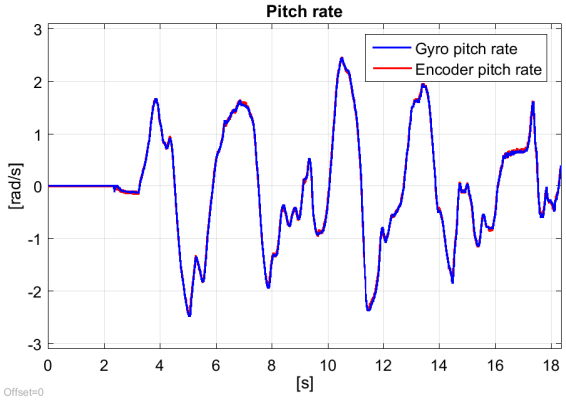


Figure 14: The pitch rate measured from the transformed gyroscope output compared to the rate from the encoder

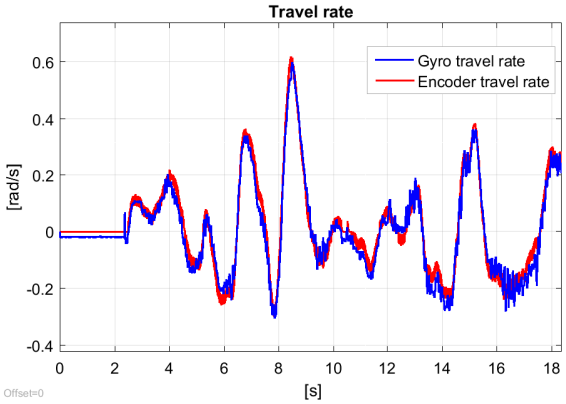


Figure 15: The travel rate measured from the transformed gyroscope output compared to the rate from the encoder

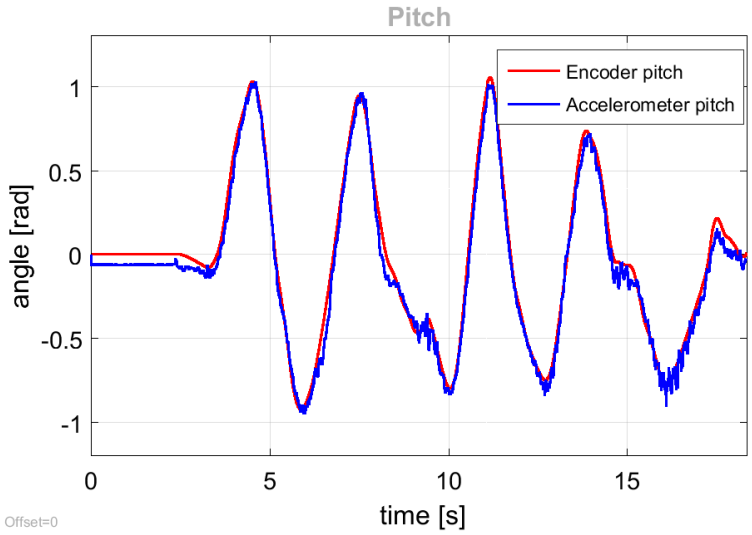


Figure 16: Accelerometer measurements of the pitch compared to the encoder

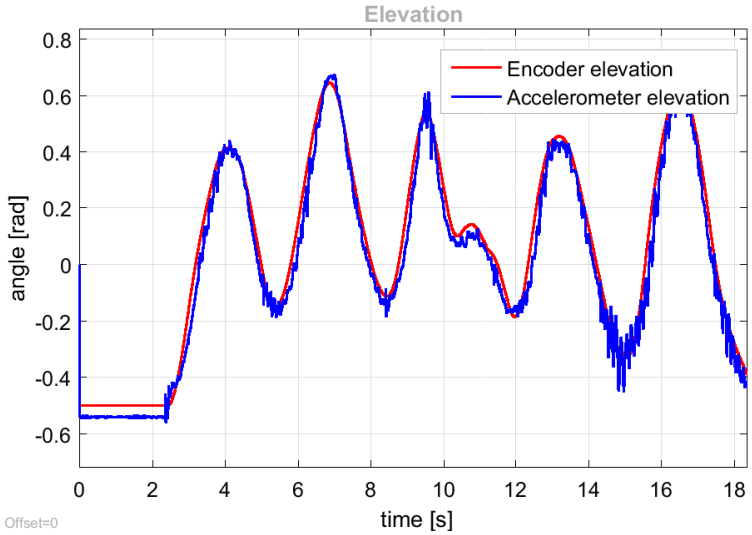


Figure 17: Accelerometer measurements of the elevation compared to the encoder

3.4 Task 4 - State estimator

To reduce noise from the IMU-measurement, the state estimator in Equation 12, derived in the lab preparations, was implemented(see figure 40).

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}), \tag{12a}$$

$$\begin{bmatrix} \dot{\hat{p}} \\ \ddot{\hat{p}} \\ \dot{\hat{e}} \\ \ddot{\hat{e}} \\ \dot{\hat{\lambda}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ K_3 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \dot{\hat{p}} \\ \hat{e} \\ \dot{\hat{e}} \\ \dot{\hat{\lambda}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u} + \mathbf{L} \left(\mathbf{y} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{p} \\ \dot{\hat{p}} \\ \hat{e} \\ \dot{\hat{e}} \\ \dot{\hat{\lambda}} \end{bmatrix} \right) \tag{12b}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} \tag{12c}$$

We used the tuned LQR with integral action, Equation 8, as the controller. While using the encoders for feed-back, we compared the state estimates with the encoder- and IMU-measurements. Since we were to use all the measurements available from the IMU, we chose C to be the identity matrix. We then tuned the observer by using the Matlab-function `L=place(A',C',p)'` and pole placement, by varying the poles in the vector **p**.

The state estimator were used to smooth the signal, since there were a lot of noise in the measurements. When using an open-loop estimator in conjunction with a feedback from the measurement error, we got an closed-loop estimator. This feedback corrects a certain degree of noise due to measurement errors and disturbances. Since the pair of matrices **A** and **C** were observable, we could assign all eigenvalues of the gain matrix **L** arbitrarily [3, p. 250]. As already mentioned, this were done by using Matlab function `place`, and varying the poles in vector **p**. When **L** were properly designed, the estimated state were driven to the actual state.

While we experimented with different pole-placements, we noticed that small values led to large discrepancies from the actual state. Because of the low values, the observer gain was too small for the observer to be able to approach the actual state. When using an estimator in state feedback, the poles of the error dynamics should be faster than the plant itself in order to achieve a large enough observer gain [3, p. 250]. This will ensure that the estimation is updated adequately. The observer was therefore too slow when testing with low values. However, the observer reduces noise by low-passing the noise contribution in the estimate. A low observer gain therefore gives a

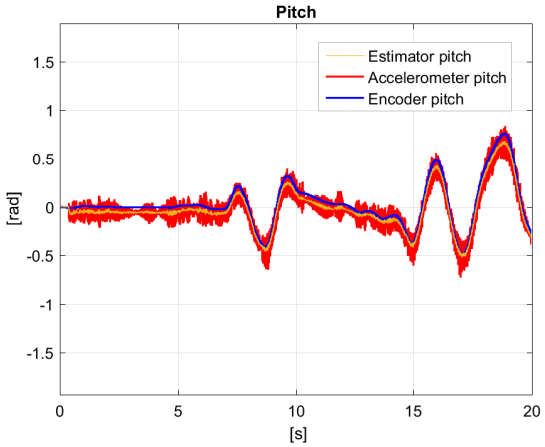


Figure 18: Comparison of the pitch with the tuned observer

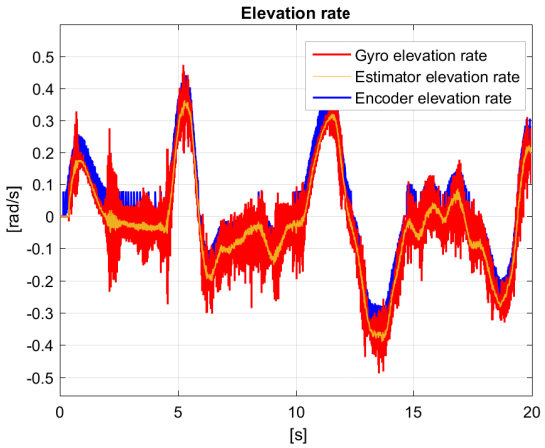


Figure 19: Comparison of the elevation rate with the tuned observer

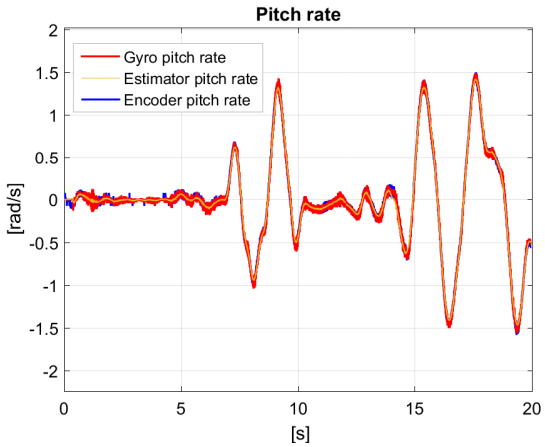


Figure 20: Comparison of the pitch rate with the tuned observer

low cut-off frequency. Meaning that with low enough pole values, an observer can reduce approximately all the noise, but at the expense of following the actual state.

With large pole values, we noticed that the observer followed the actual state extremely well. However, the noise had been amplified. This were caused by the large pole value leading to a large gain matrix **L**. Since observers low-passes the noise contribution in the estimate, the cut-off frequency increases when choosing a higher observer gain, leading to a noisier signal than with lower pole-values.

The pole placement came down to us choosing what degree of noise canceling we wanted at the expense of good estimation. We wanted a fairly exact estimation, hence allowed some noise in the signal. We then got a signal that followed the actual state almost prefect, still having less noise than the measurements. The tuned observer had

$$P = \begin{bmatrix} -40 & -40 & -80+5i & -80-5i & -5 \end{bmatrix},$$

which resulted in

$$L = \begin{bmatrix} 40 & -4 & 0 & 0 & 0 \\ 5 & 40 & 0 & 0 & 0 \\ 0 & 0 & 40 & 1 & 0 \\ 0 & 0 & 0 & 40 & 0 \\ 0.6117 & 0 & 0 & 0 & 5 \end{bmatrix}.$$

The state estimates, compared to both the gyro outputs and encoder outputs, are shown in figure 18, 19 and 20. We see that we were able to tune the Luenberger observer in a way that reduced a big part of the measurement noise from the IMU, as well as still following the main signal as wanted.

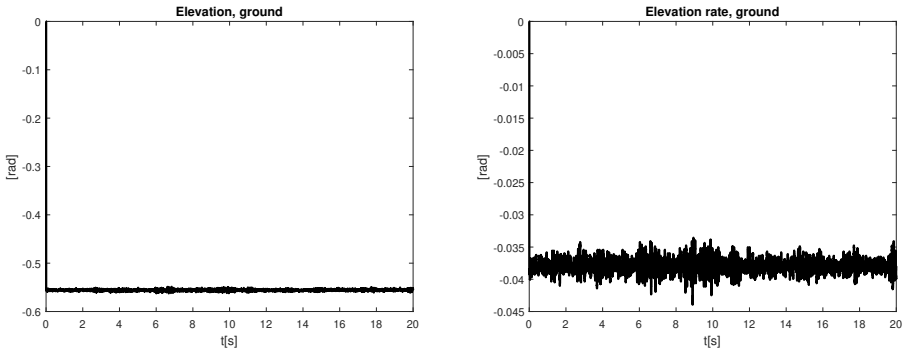
4 Part 4 - Kalman Filter

4.1 Task 1 - Noise estimate

In order to experimentally estimate the measurement noise, $\mathbf{R_d}$, we saved all measurements from the IMU in an array when the helicopter was laying still on the ground and when it was stabilized at the linearization point. We used the transposed of the measurements in order to get the matrices on the right form. The $\mathbf{R_d}$ matrix were then derived by taking the covariance of the noise, and used in further tasks.

The measurement noise while the helicopter was still on the ground was much smaller than when the helicopter was flying at the linearization point, as seen in Figure 21, Figure 22, Figure 23 and Figure 24.

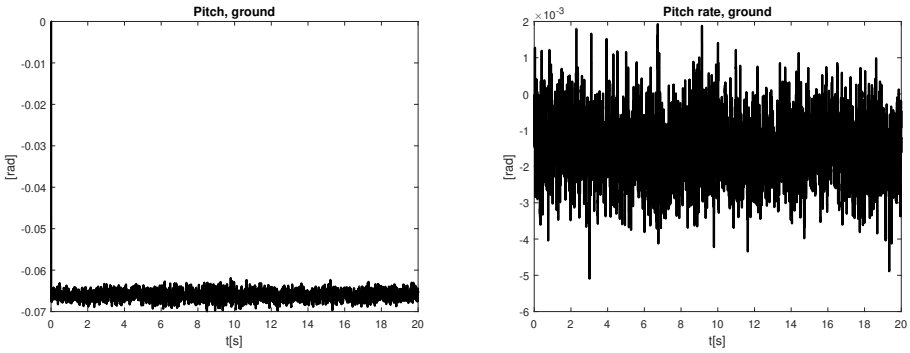
Measures from the helicopter laying still on the ground does not represent the actual noise in the system when it is flying. This is because when the system is flying, the measurements are affected by small movements in the helicopter that do not occur when the helicopter is grounded. This difference leads to greater noise. Therefore, to get the most correct measurement of the actual noise, we used the covariance value from the flying helicopter in the Kalman filter, shown in Equation 13.



(a) (b)

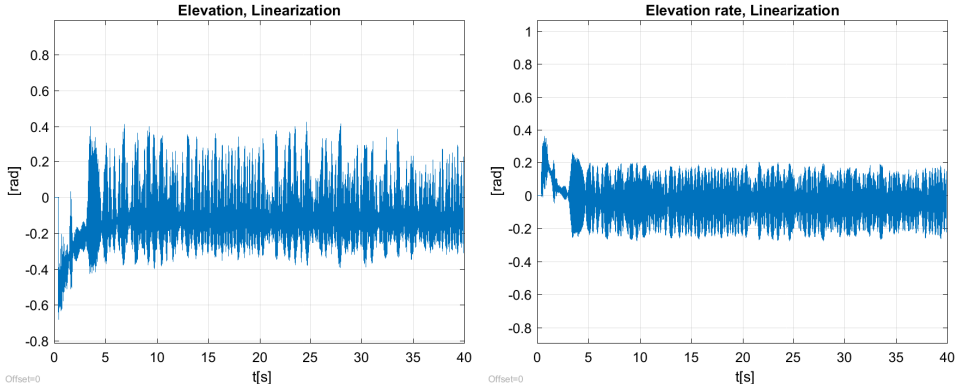
Figure 21: Elevation and elevation rate, ground

The covariance matrices describe the covariance between each pair of elements in a random signal. We found that the matrix corresponding to the helicopter flying at the linearization point has larger elements than the helicopter laying still, which means that the variables of the flying system are related to a larger degree. A variation in the pitch will most likely make the other states change due to physical dependencies. However, when the helicopter is grounded, noise is the only thing making the states vary. This makes noise the only factor affecting the covariance matrix, while the flying system is also affected by the physical dependencies. Consequently, the grounded system resulted in smaller elements in the covariance matrix.



(a) (b)

Figure 22: pitch and pitch rate, ground

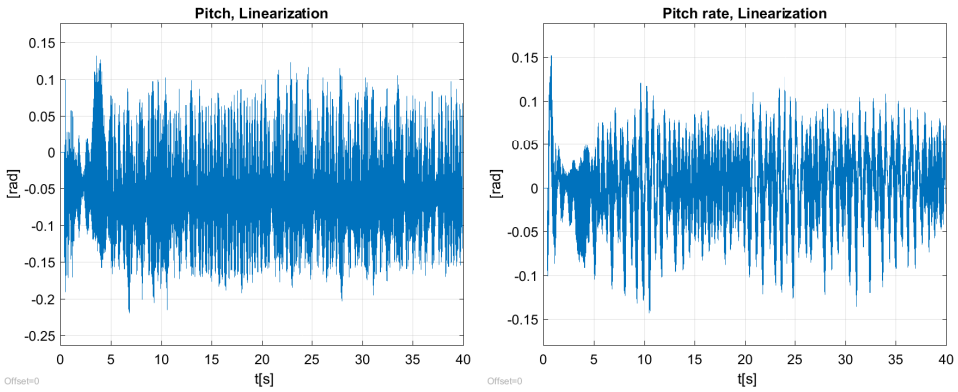


(a) (b)

Figure 23: Elevation and elevation rate, linearization

$$\mathbf{R}_d = \begin{bmatrix} 0.0052 & 0.0014 & 0.0020 & -0.0033 & -0.0005 \\ 0.0014 & 0.0017 & 0.0004 & -0.0012 & 0.0016 \\ 0.0020 & 0.0004 & 0.0297 & -0.0065 & -0.00465 \\ 0.0033 & -0.0012 & -0.0065 & 0.0079 & -0.0076 \\ -0.0005 & 0.0016 & 0.0465 & -0.0076 & 0.1229 \end{bmatrix} \tag{13}$$

Kalman Filters are used to correct noisy time series data, and this is done by expressing the data measured as a state space model and applying probabilistic estimation to this model. This method represents the noise by using a zero mean white Gaussian signal. White noise is a stationary random process that have a constant spectral density function [2, p. 75]. However, from the signals showing the noise, e.g Figure 23 and Figure 24, it is possible to see that the noise had a pattern. This indicates that the spectral density was varying, and not constant as expected for white noise.



(a) (b)

Figure 24: pitch and pitch rate, linearization

Hence, we concluded with the noise being colored. Even though Kalman filter assume white noise, we used the colored noise because it did not affect the system.

4.2 Task 2 - Discretization

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{14a}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}, \tag{14b}$$

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \\ \dot{\lambda} \\ \ddot{\lambda} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \\ \dot{\lambda} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u}, \tag{15a}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \\ \dot{\lambda} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{15b}$$

In the second task, we derived a full state-space model, Equation 14, and discretized it, as seen in appendix B.3.1, by using the Matlab function `c2d(sysc, Ts)`, where `sysc` is the continuous system $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$, and `Ts`

is the sampling period. This resulted in a system on the form:

$$\mathbf{x}[k+1] = \mathbf{A}_d\mathbf{x}[k] + \mathbf{B}_d\mathbf{u}[k] + \mathbf{w}_d[k], \tag{16a}$$

$$\mathbf{y}[k] = \mathbf{C}_d\mathbf{x}[k] + \mathbf{v}_d[k], \tag{16b}$$

$$\mathbf{w}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_d), \mathbf{v}_d \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_d) \tag{16c}$$

We used a timestep of $0.002s$. The covariance of the disturbance $\mathbf{Q}_d$ served as a tuning factor in task 4.4.

4.3 Task 3 - Implementation

Next, we implemented the Kalman functions, derived in preparations, for the prediciton- and correction step in two separate Matlab function blocks in Simulink (See appendix B.3.1, Figure 41 and 42). Equations 17, 18 and 19 were implemented in the correction block, while equations 20 and 21 were implemented in the prediction block. Because the prediction equations is at time step $k+1$, and the correction at k, we had to implement a unit delay block in simulink to delay the signal one time step. The correction function block also takes newdata from the IMU-block as input. This signal is 1 if new data has arrived, and 0 otherwise. We implemented a code such that the corrected state estimate, $\bar{\mathbf{x}}[k]$, were equal to the predicted state estimate, $\hat{\mathbf{x}}[k]$, and the corrected error covariance, $\bar{\mathbf{P}}[k]$, were equal to the predicted error covariance, $\hat{\mathbf{P}}[k]$, if newdata $= 0$.

$$\mathbf{K}[k] = \bar{\mathbf{P}}[k]C_d^T(C_d\bar{\mathbf{P}}[k]C_d^T + \mathbf{R}_d)^{-1} \tag{17}$$

$$\hat{\mathbf{x}}[k] = \bar{\mathbf{x}}[k] + \mathbf{K}[k](\mathbf{y}[k] - \mathbf{C}_d\bar{\mathbf{x}}[k]) \tag{18}$$

$$\hat{\mathbf{P}}[k] = (\mathbf{I} - \mathbf{K}[k]\mathbf{C}_d)\bar{\mathbf{P}}[k](\mathbf{I} - \mathbf{K}[k]\mathbf{C}_d)^T + \mathbf{K}\mathbf{R}_d\mathbf{K}^T \tag{19}$$

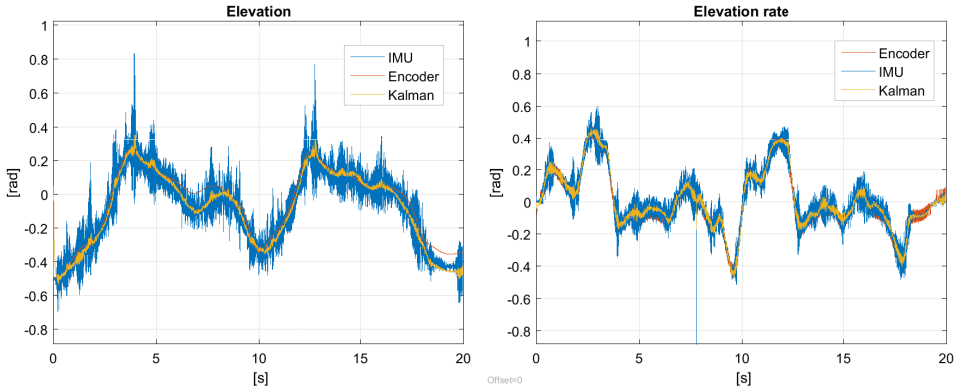
$$\hat{\mathbf{x}}[k+1] = \mathbf{A}_d\hat{\mathbf{x}}[k] + \mathbf{B}_d\mathbf{u}[k] \tag{20}$$

$$\bar{\mathbf{P}}[k+1] = \mathbf{A}_d\hat{\mathbf{P}}[k]\mathbf{A}_d^T + \mathbf{Q}_d \tag{21}$$

4.4 Task 4 - Experimentation

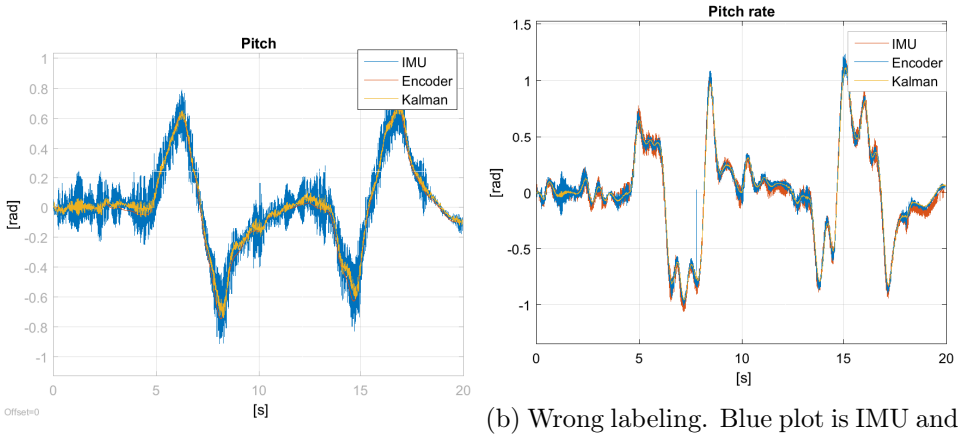
In order to experiment with the Kalman filter, we used the encoders for feedback, and the same LQR-regulator as we did with the Luenberger observer. We tested the Kalman filter with different values in the diagonal matrix $\mathbf{Q}_d$ and then compared the corrected state estimates, $\hat{\mathbf{x}}$, with the encoder signals and IMU measurements. Figure 25 and Figure 26 show the comparison. The estimates are similar to the IMU values except for having filtered out a large amount of the noise. The estimation for both the rates is also less noisy than

the encoder, however the encoder measurements of the pitch and elevation is less noisy. The spikes in the IMU measurements around $t = 8s$ is due to poor wire connection, something that happened on and off throughout the lab.



(a) (b)

Figure 25: Tuned kalman Filter



(a)

(b) Wrong labeling. Blue plot is IMU and red is encoder.

Figure 26: Tuned kalman Filter

Next, we tested our hypothesis that if all the elements in $\mathbf{Q}_d$ were large, we chose 1 000 000, the Kalman filter would follow the input measurements completely, and if $\mathbf{Q}_d$ were 0, the filter would not be able to follow the input. Testing the Kalman filter with $\mathbf{Q}_d = 0$ gave a response shown in Figure 27. The result were that the Kalman filter signal did not follow the input signal at all, and were unusable. On the other side, when using the much larger values in $\mathbf{Q}_d$, we got a response shown in Figure 28. The Kalman filter signal then followed the input signal, but did not reduce any noise.

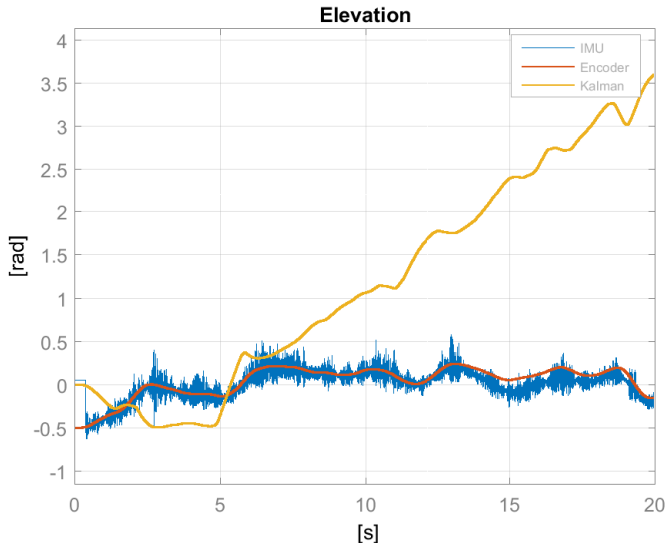


Figure 27: Result of setting $\mathbf{Q}_d$ matrix to 0

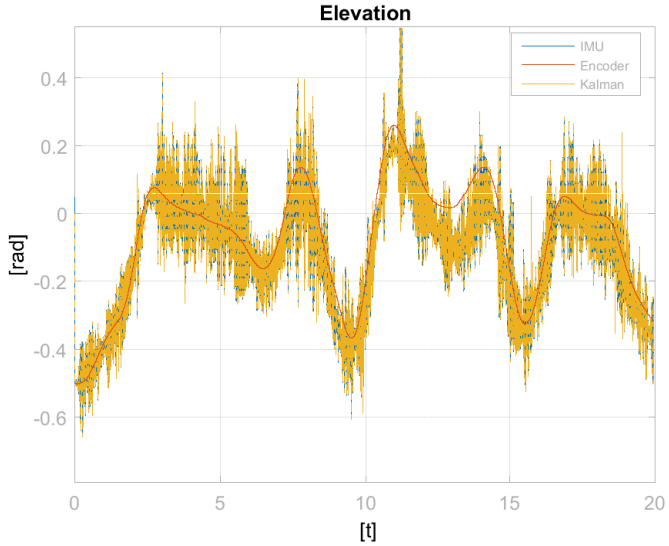


Figure 28: Result of setting $\mathbf{Q}_d$ matrix to 1M

Our hypothesis was based on Kalman filter theory. The matrix $\mathbf{Q}_d$ has a part in deciding the Kalman gain $\mathbf{K}$. A large $\mathbf{Q}_d$ leads to a large gain $\mathbf{K}$, resulting in a stronger measurement based updating of the state estimates. Further explained, large elements in $\mathbf{Q}_d$ states that the variations in the real state variables are assumed to be big. Higher Kalman gain therefore gives more noise, because it has more emphasis on the measurement than the model. This is what we saw when testing with $\mathbf{Q}_d$ equal to 1 000 000, where the signal from the Kalman filter was the same as the measurement input, $\mathbf{y}$.

When testing with $\mathbf{Q}_d$ equal to 0, we saw that the signal from the Kalman filter drifted from the input measurement. This is because a low Kalman gain means that the filter has low emphasis on the measurement, meaning trusting the model more. So when $\mathbf{Q}_d$ was zero, it trusted the model completely with no consideration on what the measurement actually was. The filter therefore had to be tuned by setting the diagonal elements of the $\mathbf{Q}_d$ matrix as large as possible without the state estimations becoming too noisy.

The next task was to experiment with what would happen if the the new data signal switched on and off. We looked at what happened and how the filter reacted to this change. To show how the corrected state estimate, $\hat{\mathbf{x}}$, reacted when switching new data on and off, we plotted the pitch response together with the new data signal. In order to show how the $\hat{\mathbf{P}}$ matrix changed, we plotted the sum of the diagonal, which corresponds to the sum of the corrected estimation error variances. Before doing this, we excluded the variance of travel, because this is not a measured state and will therefore never receive new data. By doing that, we get the true variance sum of the five other states.

The reason for us scoping the sum of the diagonal of the $\hat{\mathbf{P}}$ matrix, is because this is where the variances of the corrected error is. When the new data switch is turned off, the filter has to predict several time steps with no correction. For each prediction the estimate becomes progressively uncertain, which translates to the variance increasing, as we saw in Figure 30. Figure 29 shows that the pitch increased rapidly from the linearization point when the correction was turned off. This can be explained by the prediction having to be made on previous predictions, resulting in the continuous movement pattern that occurs. The filter keeps predicting movement in the same manner as in the last time step. When new data was turned back on, the state estimate were corrected back to the linearization point and the variance immediately dropped to approximately zero which means that the uncertainty in next prediction is almost zero again.

4.5 Task 5 - Tuning

Lastly, we tuned the filter. We started with the identity matrix as $\mathbf{Q}_d$, and for each element of the diagonal, we decreased the value if the correspond-

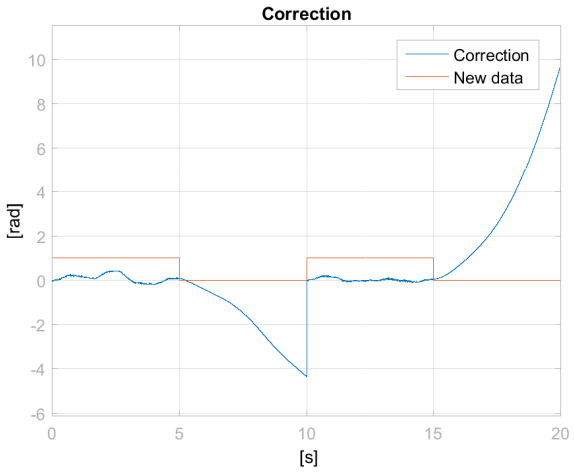


Figure 29: Corrected state estimate pitch when switching new data on and off

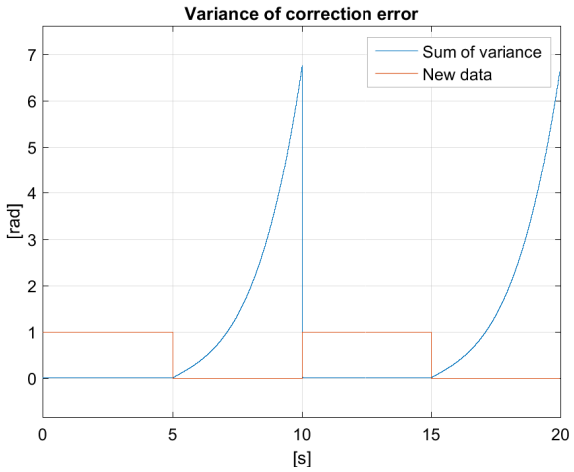


Figure 30: Sum of variance of the correction error when switching new data on and off

ing state gave a too noisy signal, and increased it if the response were too inaccurate. After tuning, we compared the Kalman filters performance with the Luenberger observer from lab 3.

The tuning of the filter resulted in

$$\mathbf{Q}_d = \begin{bmatrix} 0.00005 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.00001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0002 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0001 \end{bmatrix},$$

which made the signal accurate while not gaining to much noise. Figure 25 and Figure 26 shows the tuned filter compared with the encoder and IMU measurements.

The plots in Figure 31, Figure 32 and Figure 33 compares the performance from the tuned Kalman filter with the tuned Luenberger observer. The Kalman filter was not able to reduce as much noise as the Luenberger observer, however we did get that the Kalman filter followed the small changes in the system more accurate than Luenberger. We did try to tune Kalman to reduce more noise by lowering values in $\mathbf{Q}_d$, but that resulted in worse following of the signal.

Both Luenberger and Kalman are methods for reducing noise in the measurements. While Kalman filters models and considers the noise when choosing how much it can trust the measurements and predictions, the Luneberger observer corrects a constant amount of uncertainties due to disturbances. As we saw from the results, the Luenberges observer sometimes cut off the small changes in the measurements. This could be caused by having too low values when tuning the observer, leading to filtering away the small movements as if it was noise. We saw that the Kalman filter followed these small movements well, but with more noise than the observer, as a result of the Kalman filter "trusting" the measurements more than the prediction. To conclude, the Kalman filter is preferable, because of its precise following of the main signal, as well as reducing a big part of the measurement noise. Luenberger observer did reduce a lot of the noise and disturbances, but does not take previous data into account as the Kalman filter do, and were therefore less accurate.

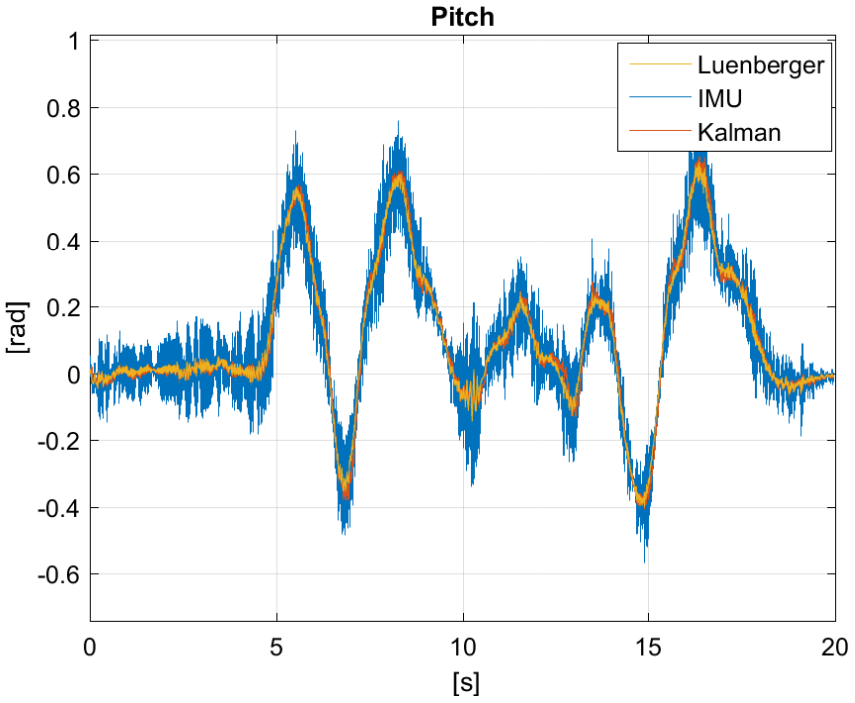


Figure 31: Compared pitch from Luenberger observer and Kalman filter

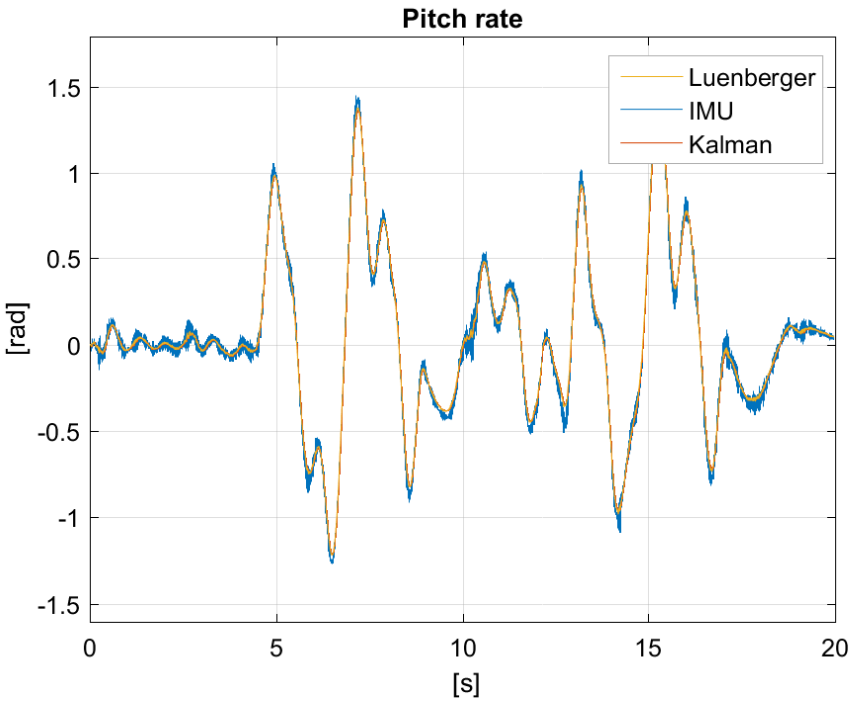


Figure 32: Compared pitch rate from Luenberger observer and Kalman filter

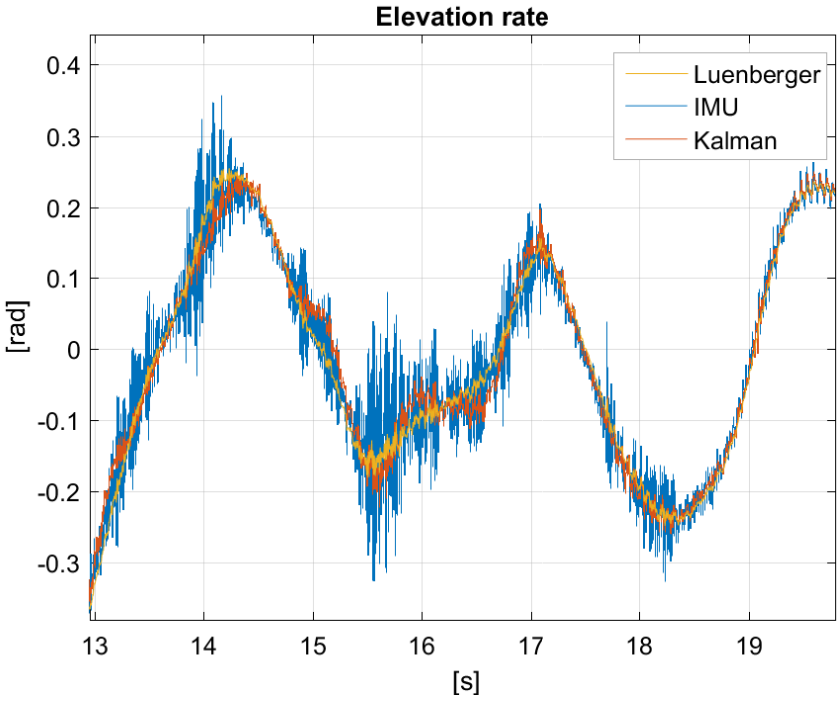


Figure 33: Compared elevation rate from Luenberger observer and Kalman filter

A Simulink diagram

A.1 Simulink Diagram, lab 1

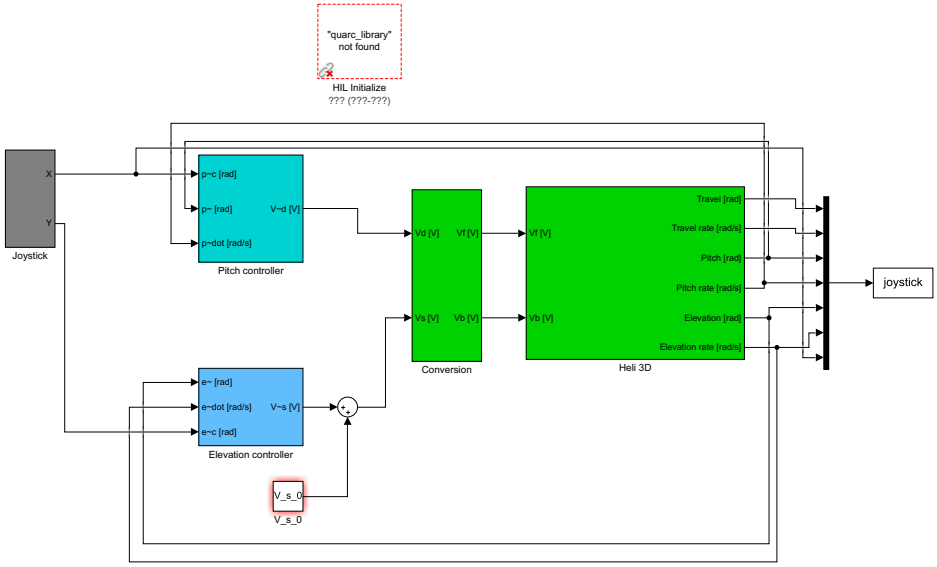


Figure 34: Simulink diagram, lab 1

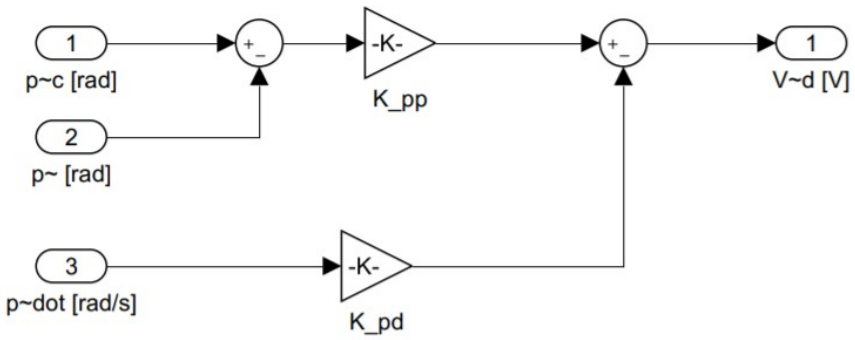


Figure 35: Diagram of pitch controller, lab 1

A.2 Simulink Diagram, lab 2

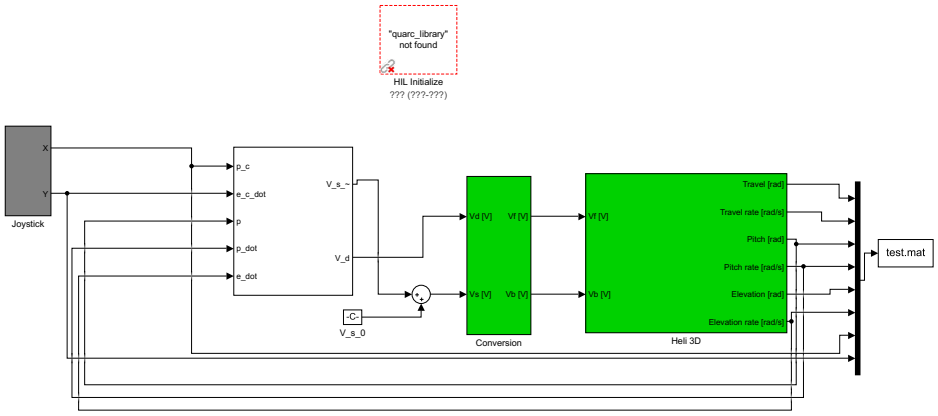


Figure 36: Simulink diagram of lab 2

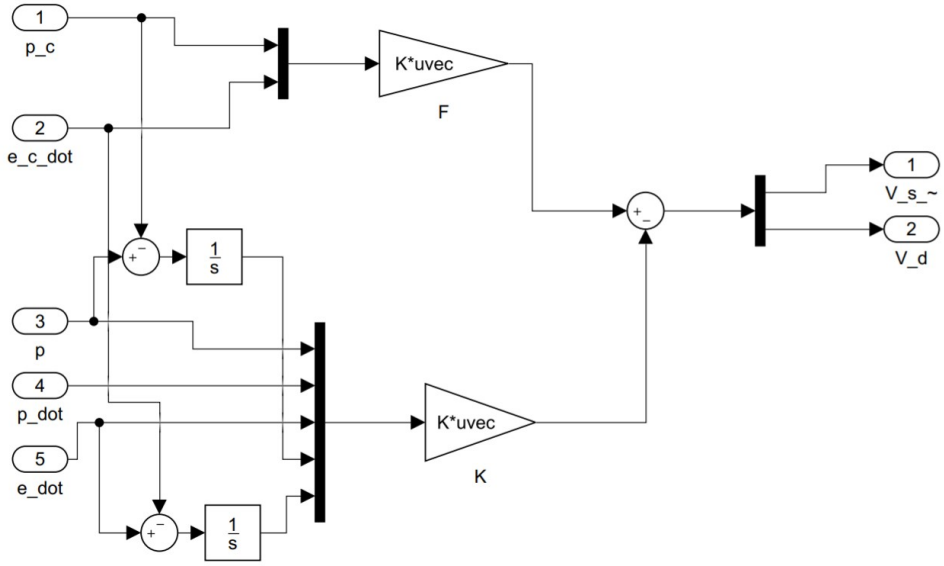


Figure 37: Simulink diagram of the controller in lab 2, with integral action

A.3 Simulink Diagram, lab 3

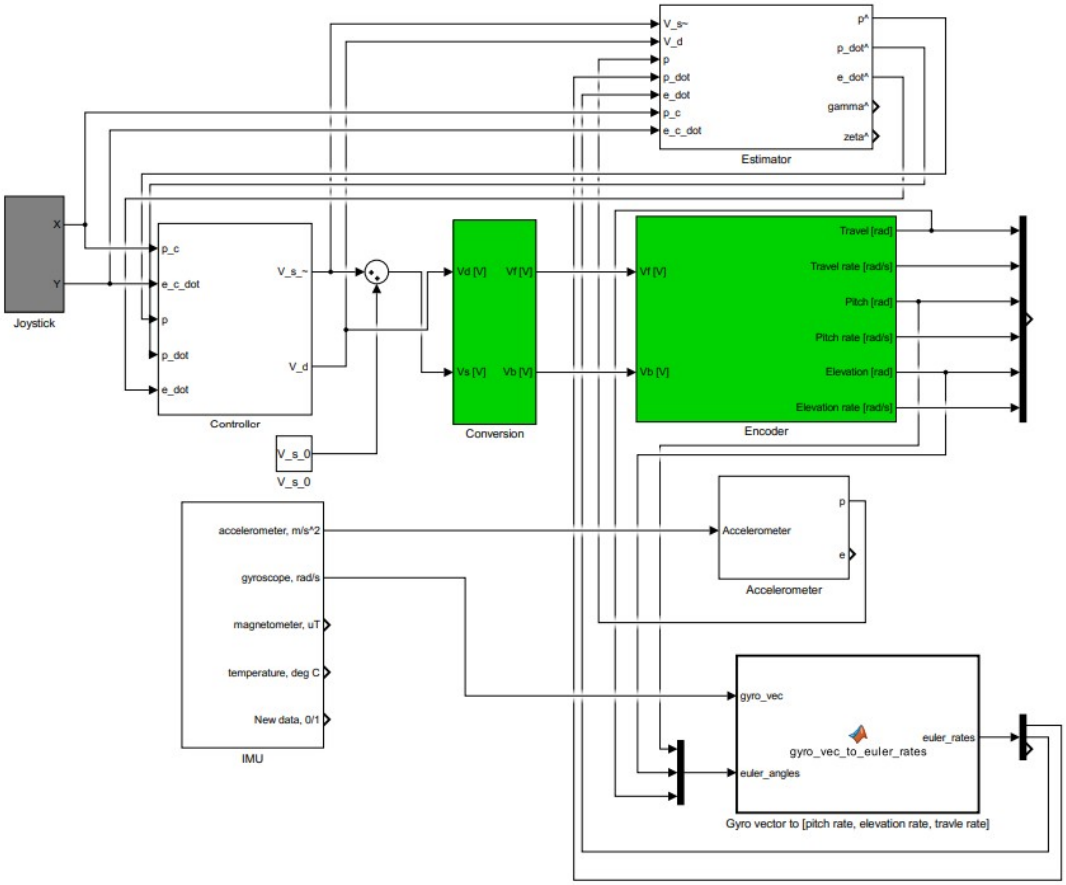


Figure 38: Simulink diagram of lab 3

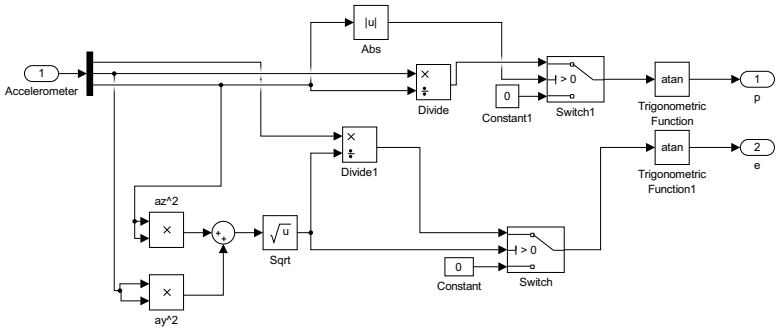


Figure 39: Simulink diagram of the accelerometer subsystem, lab 3

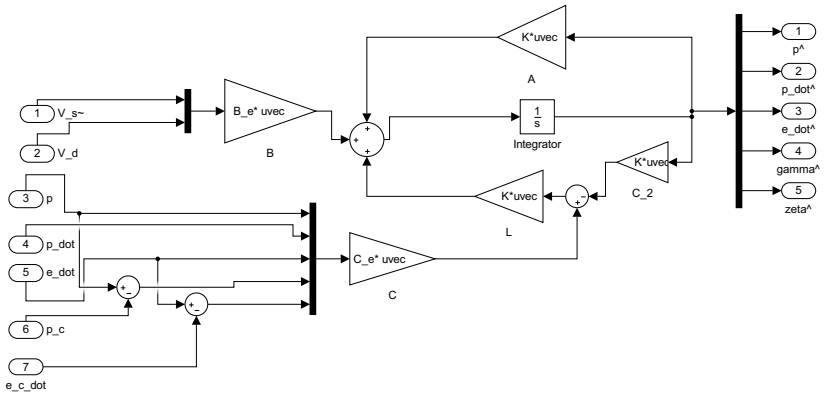


Figure 40: Simulink diagram of the observer subsystem, lab 3

A.4 Simulink Diagram, lab 4

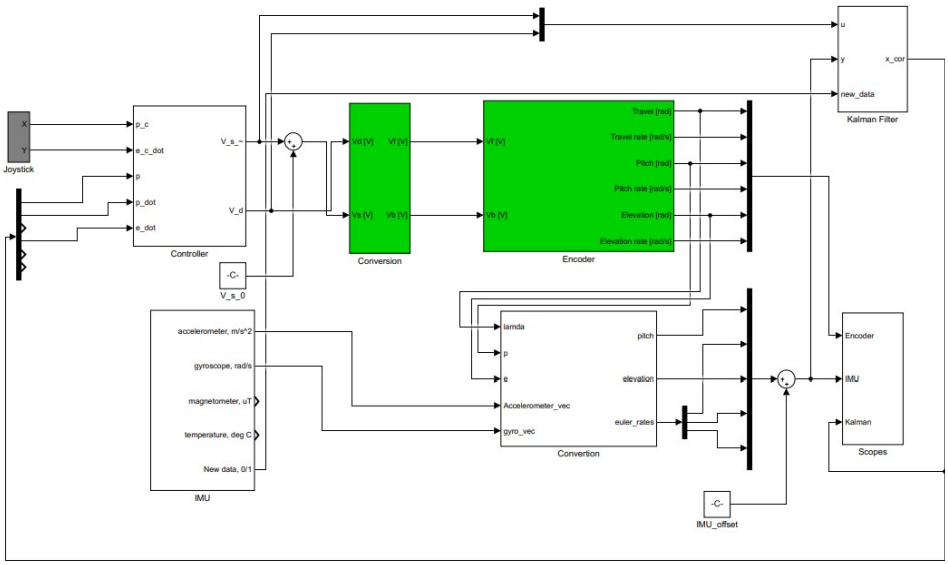


Figure 41: Simulink diagram lab 4

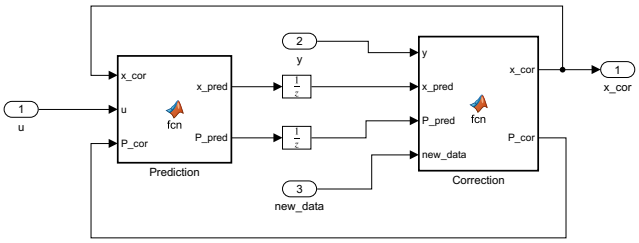


Figure 42: Simulink diagram of Kalman Filter subsystem, lab 4

B MATLAB Code

B.1 Matlab code lab 1

```
% Calibration of the encoder and the hardware for the
%specific helicopter
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_s_0 = 8.5;
K_f = -(m_c*g*l_c-2*m_p*g*l_h)/(V_s_0*l_h);

lamda_1 =-2.8;
lamda_2 = -2.8;

L_1 = l_p*K_f;
K_1 = L_1/(2*m_p*(l_p)^2);
K_pp = lamda_1*lamda_2/K_1;
K_pd = -(lamda_1+lamda_2)/K_1 ;
```

B.2 Matlab code lab 2 and 3

```
PORT = 5;

%Physical constants
V_s_0 = 8.5;
K_f = -(m_c*g*l_c-2*m_p*g*l_h)/(V_s_0*l_h);

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);
L_1 = l_p*K_f;
L_2 = m_c*g*l_c-2*m_p*g*l_h;
L_3 = K_f*l_h;
L_4 = K_f*l_h;
K_1 = L_1/(2*m_p*(l_p)^2);
K_2 = L_3/J_e;
K_3 = -L_2/J_l;

% LQR-matrices with tuned vales
A = [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 = [0 0; 0 K_1; K_2 0; 0 0;0 0];
Q=[45 0 0 0 0;0 40 0 0 0;0 0 100 0 0;0 0 0 12 0;0 0 0 0 6];
R=[0.7 0;0 0.7];

K = lqr(A,B,Q,R);
F = [K(1,1) K(1,3); K(2,1) K(2,3)];

IMU_offset = [0.055; 0.002; 0.05; 0.04; 0.02];
% Matrices for estimator implementation, with tuned p
p=[-40+5i -40 -40 -50 -40-5i];
A_e = [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];
B_e =[0 0; 0 K_1; 0 0;K_2 0;0 0];
C_e=[1 0 0 0 0;0 1 0 0 0;0 0 1 0 0;0 0 0 1 0;0 0 0 0 1];
L_e = place(A_e', C_e', p)';
```

B.3 Matlab code lab 4

```
%Same code here as in lab 3

%Kalman
Ts = 0.002;

A_c = [0 1 0 0 0 0;
    0 0 0 0 0 0;
    0 0 0 0 0 0;
    1 0 0 0 0 0;
    0 0 0 0 0 1;
    0 0 1 0 0 0];

B_c = [0 0;
    0 K_1;
    K_2 0;
    0 0;
    0 0;
    0 0];

C_c = [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];

D_c = [0 0;
    0 0;
    0 0;
    0 0;
    0 0];

load('Ground.mat')

load('linearization.mat')

ground = transpose(gr(2:6,:));
linearization = transpose(lin(2:6,:));

R_d = cov(linearization);
ground_cov = cov(ground);

sysc = ss(A_c, B_c, C_c, D_c);
```

```
sysd= c2d(sysc, Ts);

A_d = sysd.A;
B_d = sysd.B;
C_d = sysd.C;

% Tuned
Q_d =diag([0.00005 0.00001 0.0001 0.0002 0.0001 0.0001]);
```

B.3.1 Matlab code for Kalman Filter from Simulink Function block, lab 4

```
% Code inside Kalman Filter subsystem - Prediction

function [x_pred, P_pred]= fcn(x_cor,u, P_cor, A_d, B_d, Q_d)

x_pred = A_d*x_cor+B_d*u;
P_pred = A_d* P_cor*(A_d)'+ Q_d;

% Code inside Kalman Filter subsystem - Correction

function [x_cor, P_cor]= fcn(y, x_pred,P_pred, new_data, C_d, R_d)
if(new_data)
    K_cor = P_pred*(C_d)'*(C_d*P_pred*(C_d)'+R_d)^(-1);
    x_cor = x_pred + K_cor*(y-C_d*x_pred);
    P_cor=(eye(6)-K_cor*C_d)*P_pred*(eye(6)-K_cor*C_d)
    +K_cor*R_d*K_cor';
else
    P_cor = P_pred;
    x_cor = x_pred;
end
```

References

[1] J. Balchen, T. Andresen, and B. Foss. *Reguleringsteknikk*. Institutt for teksninsk kybernetikk, NTNU, 2016.

[2] R. G. Brown and P. Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. fourth. Don Fowler, 2012.

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