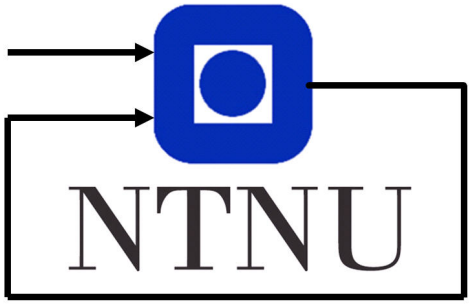
Helicopter lab report

Group 62
Jananni Johanraj, 507707
Åshild Berg Rosland, 507756

September 1, 2020



Department of Engineering Cybernetics

Contents

1 Model of helicopter

To be able to control the helicopter, the helicopter has to be modeled mathematically. A model of the helicopter is depicted in figure 1. The figure illustrates the forces affecting the helicopter, as well as the rotations of the joints, where p denotes the pitch angle of the helicopter head, e denotes the elevation angle, and λ denotes the travel angle of the helicopter.

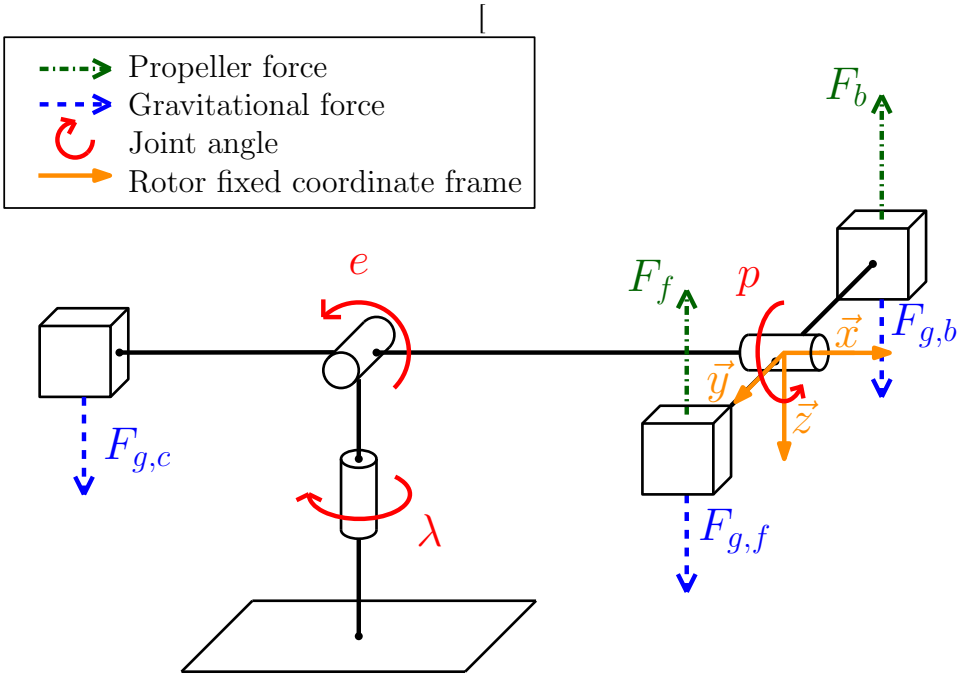


Figure 1: Model of the helicopter setup.

By using Newtons 2. law of rotational motion, the equations of motion can be obtained as

$$J_p\ddot{p} = L_1 V_d \tag{1a}$$
$$J_e\ddot{e} = L_2\cos(e) + L_3 V_s\cos(p) \tag{1b}$$
$$J_\lambda\ddot{\lambda} = L_4 V_s\cos(e)\sin(p), \tag{1c}$$

where

$$L_1 = K_f l_p \tag{2a}$$
$$L_2 = g m_c l_c - 2g m_p l_h \tag{2b}$$
$$L_3 = K_f l_h. \tag{2c}$$

By linearizing the equations above, we achieve the linearizing equations as

$$\ddot{p} = K_1 V_d \tag{3a}$$

$$\ddot{e} = K_2 \tilde{V}_s \tag{3b}$$

$$\ddot{\lambda} = K_3 p, \tag{3c}$$

where

$$K_1 = \frac{K_f}{2 m_p l_p} \tag{4a}$$

$$K_2 = \frac{L_3}{J_e} \tag{4b}$$

$$K_3 = -\frac{L_4 L_2}{L_3 J_\lambda}. \tag{4c}$$

2 Part 1 - Monovariable Control

In this section we are going to examine the use of monovariable control for the helicopter.

2.1 Task 1 - Manual control

The helicopter is being controlled by using feedforward. The signal from the x-axis of the joystick is directly connected to the voltage difference V_d, while the y-axis of the joystick is directly connected to the voltage sum V_s. This makes it possible to control both the pitch angle and the elevation angle, although it was difficult to control. To make the helicopter easier to control, we added a gain of 0.1 into V_d and a gain of 10 to V_s. However, it is still hard to control by only using feedforward.

2.2 Task 2 - Parameter identification

Assuming that the helicopter head rests on the table when Simulink is connected, the encoder outputs are not the same as the elevation angle, e. In order for the encoder value to correspond to the elevation angle, we measured the encoder value while the helicopter head rested on the table, which was equal to 0.5. For this reason, we subtracted this value from the elevation output of the encoder.

With the aim of making the helicopter easier to control, a PD-regulator using feedback will be implemented. Before we can implement the controller developed in the preparations, the motor force constant K_f must be calculated. In order to do this, $V_{s,0}$ is required to be found. $V_{s,0}$ is the value necessary to add to V_s, with the purpose of maintaining the equilibrium value $e = 0$ of the helicopter. This was done by lifting the helicopter manually, until it reached the linearization point while scoping V_s. From this, we obtained $V_{s,0} = 7.8$, which yields $K_f = 0.1281$.

2.3 Task 3 - PD-Control

The PD-controller to be implemented is given by

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p}. \tag{5a}$$

Implementation of the PD-controller in Simulink is depicted in figure 2.

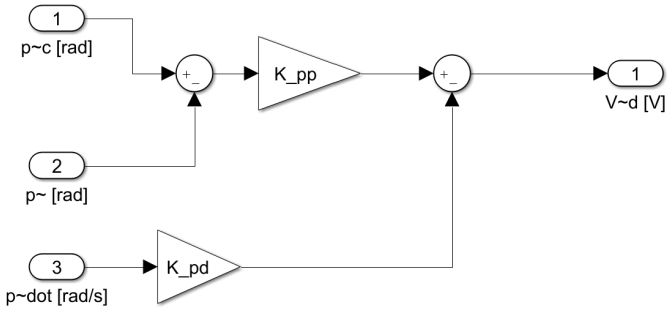


Figure 2: Implementation of the PD-controller in simulink.

The transfer function of the controller is obtained by using Laplace transform. From the transfer function the constants K_{pd} and K_{pp} are determined as

$$K_{pd} = -\frac{\lambda_1 + \lambda_2}{K_1} \tag{6a}$$

$$K_{pp} = \frac{\lambda_1 \lambda_2}{K_1} \tag{6b}$$

We experimented with different pole placements to examine how placement of the poles affected the behavior of the system. Theoretically, poles placed in the right half-plane provide an unstable system, while poles placed on the imaginary axis yield a marginally stable system [2, p. 266]. Poles placed exclusively in the left half-plane give a stable system. As we want a stable system, the poles in the experiment were placed solely in the left half-plane. This results in three possible options for pole placements on the negative real axis: coinciding, distinct or complex conjugated poles.

In addition, the magnitude of the poles affects how the system behaves. Concerning the transient response of the system, this can be described by analyzing the solution of the state equation, $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$, which is given by $\mathbf{x}(t) = \mathbf{e}^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t \mathbf{e}^{(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau$ [1, p. 87]. This means that a high magnitude of the poles yields a fast dynamic and the transient response subsides quickly, while a small magnitude of the poles yields a slow dynamic. Furthermore, placing the poles further into the left half-plane, makes the response more oscillatory, due to time lags in the system [6, p. 115]. This is confirmed in our testing, where poles of higher values tend to give a faster and oscillating response, independent of pole type. We will see this in the following experimentation.

Firstly, we tested how the system behaved with coinciding poles. From theory we know that coinciding poles yield a critically damped system with

$\zeta = 1$ [2, p. 143]. With coinciding poles $p_1 = p_2 = -1$ the response was fairly slow, as shown in figure 3. To get a faster response we placed the poles further into the left half-plane, at $p_1 = p_2 = -5$. Although the response was faster, undesired oscillations were introduced. Consequently, we compromised between a fast response and acceptable amount of oscillations, which we obtained by placing the poles at $p_1 = p_2 = -3$.

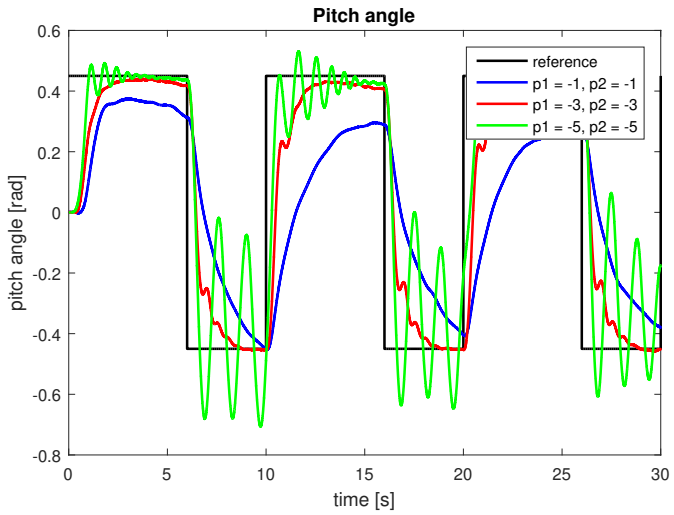


Figure 3: Plot of the system response with different placements of coinciding poles.

Furthermore, we experimented with distinct real poles on the negative real axis. From theory we know that this corresponds to an overdamped system with $\zeta > 1$ [2, p. 141]. The poles were first placed at $p_1 = -1$ and $p_2 = -3$. As we see in figure 4, the signal is overdamped and the stationary error is significant, as expected. The response is also slightly slower than the other responses. By setting $p_1 = -9$ and $p_2 = -1$, we achieved a faster response. Nevertheless, there is still a noteworthy stationary error. To decrease this deviation, we placed the poles further into the left half-plane. With $p_1 = -20$ and $p_2 = -1$, the response became oscillatory. In addition, the stationary error did not decrease, as we aimed for.

Lastly, we experimented with complex conjugated poles. Theoretically this provides an underdamped system with $\zeta < 1$ [2, p. 143]. Initially, we tested the poles placed at $p_1 = -1 + 1i$ and $p_2 = -1 - 1i$. As we see in figure 5, this yields a quite slow response. As we increase the magnitude of the real parts of the poles to $p_1 = -3 + 1i$ and $p_2 = -3 - 1i$, we get a faster response, although the response is more oscillating. Moreover, as we increase the magnitude of the imaginary parts to $p_1 = -1 + 3i$ and $p_2 = -1 - 3i$, the oscillating behaviour increases. This is expected from theory, as the imaginary parts of the poles determines the oscillating frequency. Higher

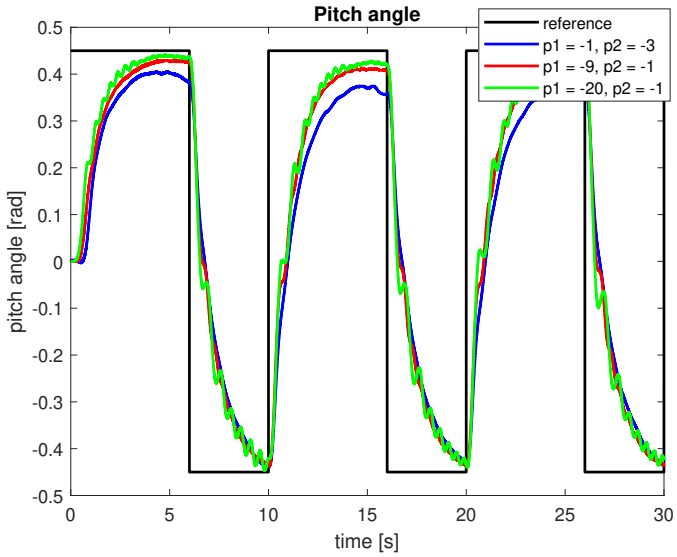


Figure 4: Plot of the system response with different placements of distinct real poles.

imaginary parts in the poles introduces a higher frequency in the response [6, p. 115].

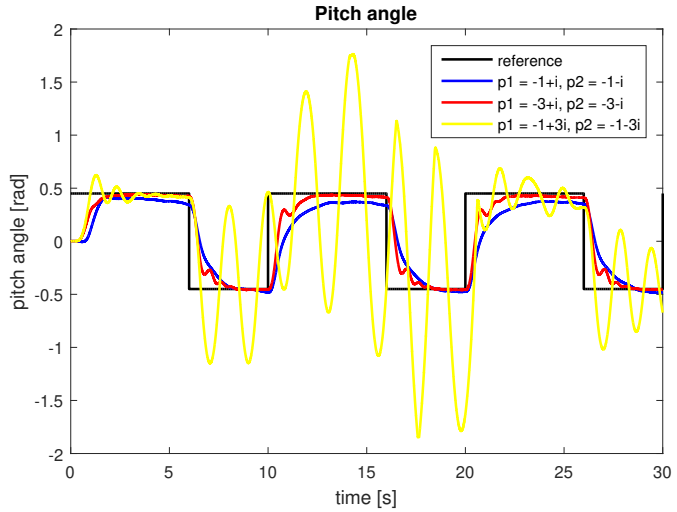


Figure 5: Plot of the system response with different placements of complex conjugated poles.

From the plots of coinciding, distinct and complex conjugated poles in figures 3, 4 and 5, respectively, notice that the distinct real poles generally have a slower response than the others. This reflects the theory well, as

the real distinct poles gives an overdamped system. Furthermore, observe that oscillations are introduced at lower pole values when using complex conjugated poles compared to coinciding and distinct real poles. As complex conjugated poles yield an underdamped system, this also reflects the theory well.

From theory we know that coinciding poles gives the best result, as the response is critically damped. This applies for our experimentation as well. However, it is important to keep in mind that this is not always the case for physical systems. Imperfect modeling and linearization gives an inaccurate representation of the real system, which will affect the response with different poles.

In the tuning we chose coinciding poles at $p_1 = p_2 = -3$, as this yield a fast response with small oscillating behaviour.

3 Part 2 - Multivariable control

In this section we are expanding our monovariable control system into a multivariable system, making it possible to control multiple states simultaneously. The multivariable system is given by

$$\underbrace{\begin{bmatrix}\dot{p}\\ \ddot{p}\\ \ddot{e}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}}_{\mathbf{A}}\mathbf{x} + \underbrace{\begin{bmatrix}0 & 0\\ 0 & K_1\\ K_2 & 0\end{bmatrix}}_{\mathbf{B}}\mathbf{u}. \tag{7}$$

3.1 Task 1 - Implementation

The implementation shown in figure 6 aims to track the reference r $= [p_c, \dot{e_c}]^T$, which is given by the joystick input. The x-axis is connected to the pitch reference p_c, and the y-axis is connected to the reference for elevation rate $\dot{e_c}$.

The implementation includes a state-feedback controller with reference-feed-forward on the form $\mathbf{u} = \mathbf{Fr} - \mathbf{Kx}$.

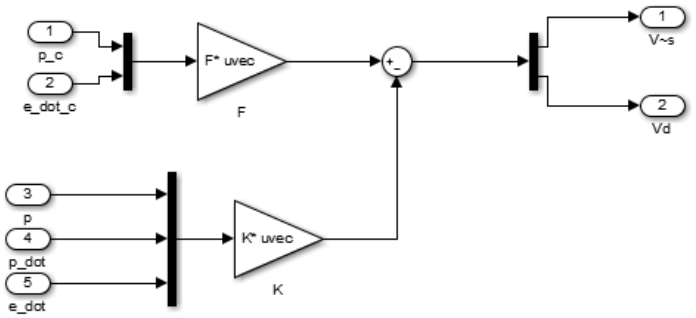


Figure 6: Implementation of the controller $\mathbf{u} = \mathbf{Fr} - \mathbf{Kx}$ in Simulink.

3.2 Task 2 - Pole Placement

To design the **K** matrix in the controller, pole placement is used. The poles are placed by using the Matlab function `K = place(A,B,p)` where p is a vector of poles. In the experimentation, we will examine how different pole placements affect the behavior of the system.

By using the command `K = place(A,B,p)` we got a **K** matrix on the form

$$\mathbf{K} = \begin{bmatrix}0 & 0 & k_{13}\\ k_{21} & k_{22} & 0\end{bmatrix}$$

Pole 1	Pole 2	Pole 3	k_{13}	Label
-1+1i	-1	-1-1i	12.23	$p1$
-1	-1	-2	12.23	$p2$
-1	-2	-3	12.23	$p3$

Table 1: Values for the poles used in simulating elevation rate. The labels correspond to figure 7.

Thus, the controller can be expressed as

$$\underbrace{\begin{bmatrix}\tilde{V}_s\\ V_d\end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix}F_{11} & F_{12}\\ F_{21} & F_{22}\end{bmatrix}}_{\mathbf{F}}\underbrace{\begin{bmatrix}p_c\\ \dot{e}_c\end{bmatrix}}_{\mathbf{r}} - \underbrace{\begin{bmatrix}0 & 0 & k_{13}\\ k_{21} & k_{22} & 0\end{bmatrix}}_{\mathbf{K}}\underbrace{\begin{bmatrix}p\\ \dot{p}\\ \dot{e}\end{bmatrix}}_{\mathbf{x}} \tag{8}$$

From the expression for the controller in equation 8, we can see that the pitch, pitch rate and elevation rate is related to k_{21}, k_{22} and k_{13}, respectively.

Firstly, we examined the elevation rate. We tested the poles given in table 1, which all yield the same value for k_{13}. Figure 7 shows the response for these values. As we can see, the responses are approximately equal for the different poles. This indicates that the type of pole is not crucial for the behavior of the system. As long as the value k_{13} is the same for each type of pole, the response in elevation rate is the same.

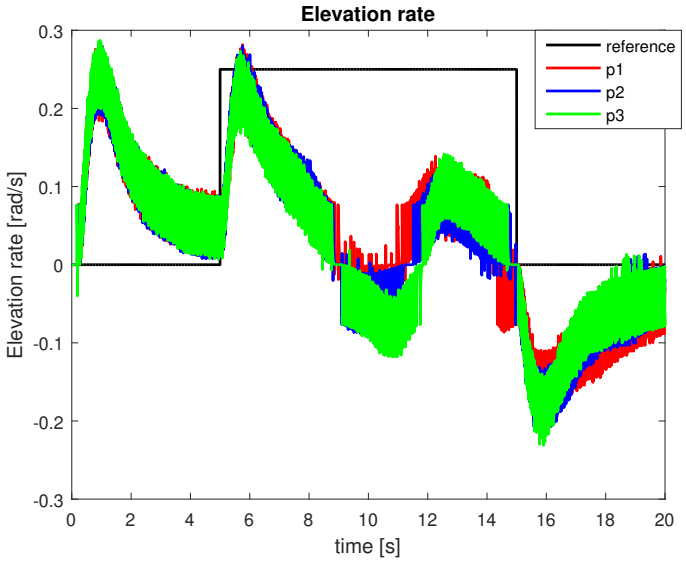


Figure 7: Plot of the responses in elevation rate with different poles

Thereafter, we examined the pitch angle. We experimented with the poles

Pole 1	Pole 2	Pole 3	k_{21}	k_{22}	Label
-1	-1+1i	-1-1i	3.9358	3.9358	$p1$
-1	-2	-3	11.8073	9.8394	$p2$
-1	-2	-2	3.9358	5.9037	$p3$

Table 2: Values for the poles used in simulating pitch angle. The labels correspond to figure 8.

given in table 2. As the magnitude of the pole p_2 is higher than the others, we expect from theory that this results in the quickest response. From figure 8 we observe that the response reflects the expectation well. Furthermore, observe that the slowest response is given by three poles placed the closest to each other, p_3. This fits well with theory, as placing the poles in a small region usually results in a slow response [1, p. 238].

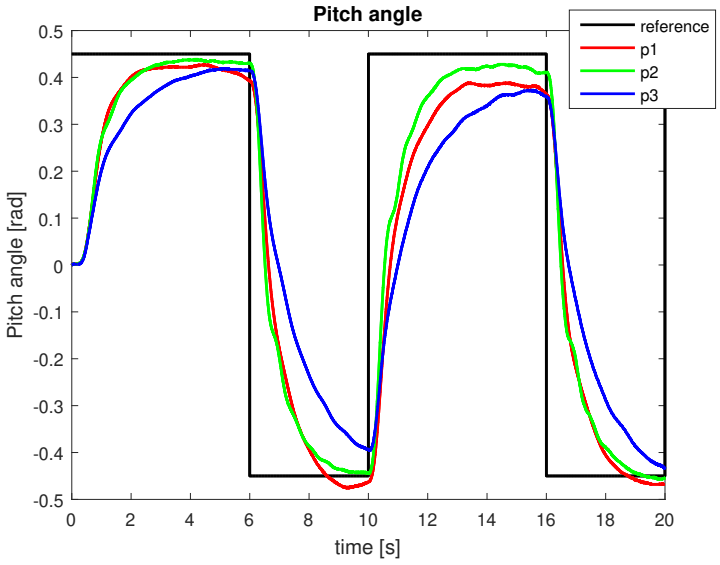


Figure 8: Plot of the responses in pitch with different poles

If we examine the table 8 more closely, we can evaluate the behavior of the system based on k_{21} and k_{22}. To begin with, we analyze equation 8, which results in the following equation for V_d

$$V_d = k_{21}(p_c - p) - k_{22}\dot{p}. \tag{9}$$

Firstly, we observe that V_d is determined by both pitch and pitch rate. We see that higher values for k_{22} yields lower values for V_d, while higher values for k_{21} yields higher values for V_d. This means that the relation between k_{21} and k_{22} determines the transient response of the pitch. As V_d increases, the response

p_1	p_2	p_3	k_{13}	k_{21}	k_{22}
-3	-7	-5	36.6862	68.8761	23.6147
-1	-10	-5	12.2287	98.3945	29.5183
-1	-3	-3	36.6862	5.9037	7.8716
-1	-10	-10	122.2875	19.6789	21.6468
-1	-2+1i	-2-1i	12.2287	9.8394	7.8716
-1	-1-3i	-1+3i	12.2287	19.6789	3.9358

Table 3: Different pole placement experimentation.

gets quicker. With the aim of getting a fast response, the desired relation between k_{21} and k_{22} is $k_{21} \geq k_{22}$, resulting in a high difference voltage V_d. This reflects the previous discussion of figure 8, where p_1 and p_2 fulfill $k_{21} \geq k_{22}$ and yields the fastest responses. As $k_{21} < k_{22}$ in p_3, the response slower than the others. To extend this discussion in conjunction with pole types, we observe some repeating tendencies in our experimentation. As shown in table 3, real distinct poles results in $k_{21} > k_{22}$, coinciding poles yields $k_{21} < k_{22}$ and complex conjugated poles gives $k_{21} \geq k_{22}$.

During the experimentation, we had an additional interesting observation. As we controlled the elevation rate of the helicopter with the joystick as reference, we expected the helicopter to reach maximum height when the joystick was directed upwards. In despite of our expectations, the helicopter did not reach the maximum height. This is caused by movement of the center of mass. When the elevation angle increases, the center of mass is moved. This results in the helicopter head to be perceived as heavier and requires more force to continue the upwards movement.

3.3 Task 3 - Linear Quadratic Regulator

Instead of using pole placement to design the **K** matrix, a linear quadratic regulator algorithm (hereafter called LQR) will be used. The LQR algorithm uses the same regulator as in section 3.1, $\mathbf{u} = \mathbf{Fr} - \mathbf{Kx}$, but optimizes the **K** matrix such that the cost function

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

is minimized [4]. For simplicity, the weighting matrices $\mathbf{Q}_{LQR}$ and $\mathbf{R}_{LQR}$ are diagonal matrices. The **K** matrix is computed by using the Matlab command `lqr(A,B,Q,R)`.

The matrices $\mathbf{Q}_{LQR}$ and $\mathbf{R}_{LQR}$ are on the form

$$\mathbf{Q_{LQR}} = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \tag{10}$$

and

$$\mathbf{R_{LQR}} = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix} \tag{11}$$

where the dimensions are obtained from the cost function. The matrix $\mathbf{Q_{LQR}}$ represents the costs for state error and the matrix $\mathbf{R_{LQR}}$ represents the costs for input. By increasing the entries in $\mathbf{Q_{LQR}}$, we penalize the error to the corresponding state. By increasing the entries q_1, q_2 and q_3 in $\mathbf{Q_{LQR}}$, we penalize the state error $p - p_c$, $\dot{p} - \dot{p_c}$ and $\dot{e} - \dot{e_c}$, respectively. By increasing the entries in $\mathbf{R_{LQR}}$, r_1 and r_2, we penalize the system for using a high input voltage $\tilde{V_s}$ and V_d, respectively.

In this task, we experimented with different values for $\mathbf{Q_{LQR}}$ and $\mathbf{R_{LQR}}$ to see how the matrices affect the behavior of the helicopter. The experimentation was done by identifying which parameters in $\mathbf{Q_{LQR}}$ and $\mathbf{R_{LQR}}$ that corresponds with the different states and inputs, respectively. Firstly, we focused on optimizing the response in elevation rate by altering the values q_3 and r_1. After testing different values of r_1, we concluded that $r_1 = 1.7$ gave a quick response, while slightly penalizing higher values of $\tilde{V_s}$ as well. This gave a smooth and controlled response. Furthermore, by experimenting with

q_1	q_2	q_3	r_1	r_2	label
100	80	150	1.7	1	LQR_1
100	10	150	1.7	1	LQR_2
100	200	150	1.7	1	LQR_3
100	80	150	1.7	0.5	LQR_4

Table 4: Different values for q_2 and r_2. The labels refer to figure 4.

various values of q_3, we concluded that $q_3 = 150$ provided the best result for elevation rate, as a higher value made the helicopter oscillate, and a lower value led to a poor reference tracking.

To get an optimal pitch response, we altered q_2 and r_2, as shown in table 4, with associated responses in figure 9. Our aim was to get a rapid response, while also avoiding overshoots. As we desire a faster response than in *LQR1*, we decreased q_2, making the system penalize the state error in pitch rate less. As we see from *LQR2*, this leads to a quicker response. However, decreasing q_2 also introduces overshoot at approximately 17 seconds in the plot. As we do not want any overshoot, we would rather have a slightly slower response in *LQR1*. The plot also shows some oscillations in the response *LQR1*. To reduce the oscillations, we increased the value of q_2, making the system penalize the state error in pitch rate more, which gave the response *LQR3*. Although we discarded the oscillations, the response was quite slow. As we aimed for a quick response, we rather accepted small oscillations, and concluded that $q_2 = 80$ gave the best response. Moreover, to get an even faster response, we decreased r_2, which results in penalizing the input V_d less. As shown in *LQR4*, this leads to a slightly faster response and an acceptable amount of oscillations, making *LQR4* the best response overall.

Although the **K** matrix ensures that the state error will be minimized over time, we will have a stationary error due to no integral effect in this task. The next section introduces integral effect to the system.

3.4 Task 4 - LQR with integral action

In this task, the controller was modified to include an integral effect for elevation rate and pitch angle. From theory we know that introducing integral effect yields a response with no stationary error [3, p. 59]. Due to the integral effect, two new states, γ and ζ, are introduced to the system. The system is therefore described as

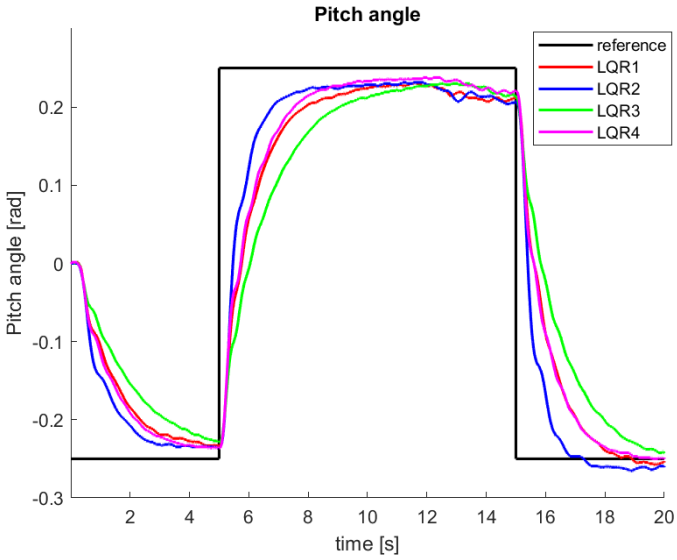


Figure 9: Plot of the response with different values of q_2 and r_2.

$$\underbrace{\begin{bmatrix}\dot{p}\\ \ddot{p}\\ \ddot{e}\\ \dot{\gamma}\\ \dot{\zeta}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0\end{bmatrix}}_{\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix}0 & 0\\ 0 & K_1\\ K_2 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}}_{\mathbf{B}} \mathbf{u} + \underbrace{\begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix}}_{\mathbf{G}} \mathbf{r} \tag{12}$$

where

$$\begin{bmatrix}\dot{\gamma}\\ \dot{\zeta}\end{bmatrix} = \begin{bmatrix}p - p_c\\ \dot{e} - \dot{e}_c\end{bmatrix}. \tag{13}$$

Figure 10 shows the implementation of the controller. As in section 3.3, we design the $\mathbf{K}$ matrix by using the Matlab command `lqr(A,B,Q,R)`.

The $\mathbf{Q_{LQR}}$ matrix is now on the form

$$\mathbf{Q_{LQR}} = \begin{bmatrix}q_1 & 0 & 0 & 0 & 0\\ 0 & q_2 & 0 & 0 & 0\\ 0 & 0 & q_3 & 0 & 0\\ 0 & 0 & 0 & q_4 & 0\\ 0 & 0 & 0 & 0 & q_5\end{bmatrix} \tag{14}$$

because we have an augmented state vector with two additional states. The $\mathbf{R_{LQR}}$ matrix is not affected by the augmented state vector and remains as in section 3.3. Due to this, we have only altered $\mathbf{Q_{LQR}}$ in the experimentation.

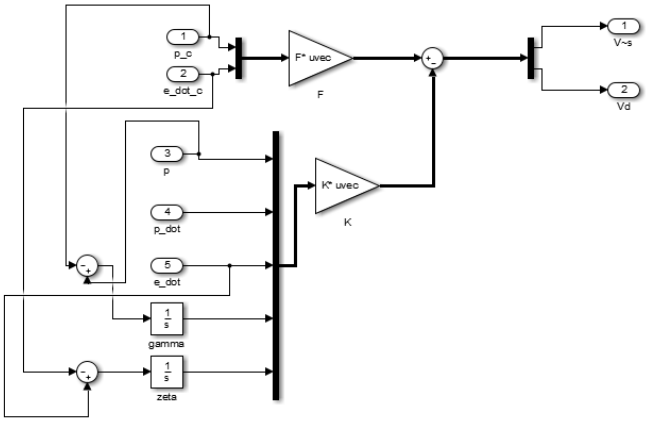
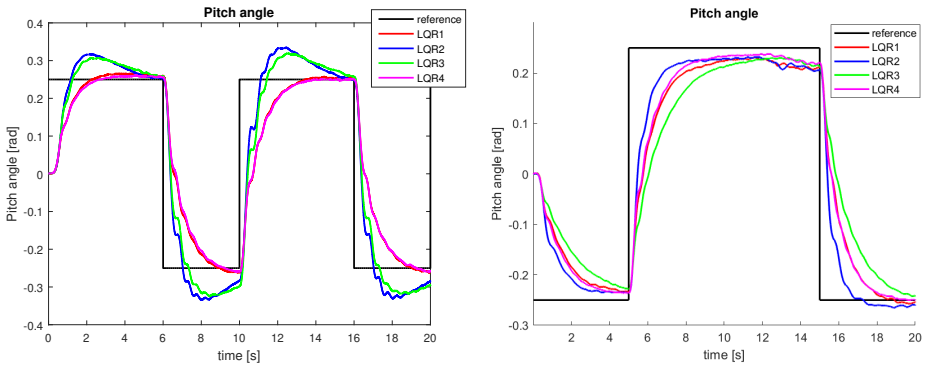


Figure 10: The implemented controller with integral effect.

To begin with, we experimented with the values q_1, q_2 and q_3 as obtained from section 3.3, and altering the additional entries q_4 and q_5 exclusively. By comparing the figures in 11, we observe that introducing integral effect does not drastically change the transient response, but the stationary error is removed, as expected from theory. However, introducing integral effect provides overshoot in the response. As we still aim towards a quick response without overshoot, we alter q_4 as shown in table 5, in order to reduce the overshoot. As a consequence, we gain a slightly slower response. As shown in the figure, the best result is given by $LQR1$, which illustrates a smooth and relatively fast response with approximately no overshoot. q_5 affects the elevation rate in a similar manner, and $q_5 = 1$ yields the best response for our system.



(a) With integral effect (b) Without integral effect

Figure 11: Plots of the response with and without integral effect.

q_1	q_2	q_3	q_4	q_5	label
125	80	150	1	1	LQR_1
125	80	150	100	1	LQR_2
125	80	150	50	1	LQR_3
125	80	150	0.5	1	LQR_4

Table 5: Different values for q_4 and q_5. The labels refer to figure 11.

Furthermore, we varied q_1 together with q_4 with the purpose of gaining a more rapid response. Table 6 shows the variation of values in the experimentation. From figure 12 we observe greater overshoots when using these values, as well as oscillations. Thus, we conclude that the values for q_1, q_2 and q_3 from section 3.3 are preferred also when using integral action, in addition to $q_4 = q_5 = 1$.

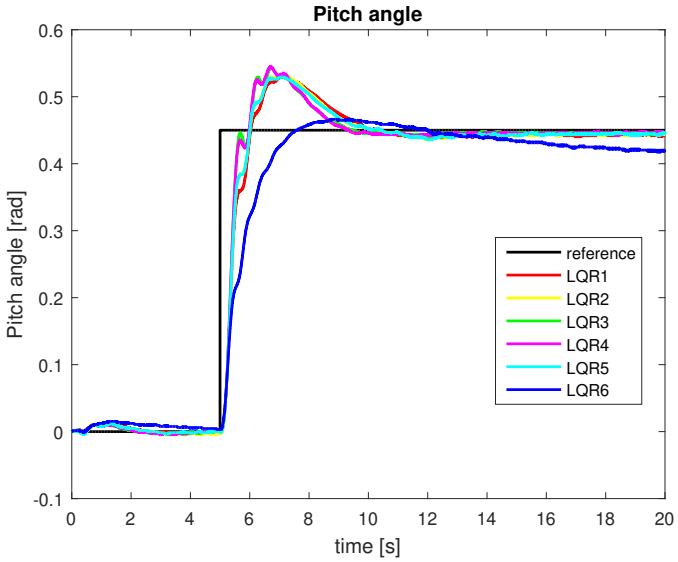


Figure 12: Different responses while altering q_1 and q_4.

In the lab preparation we calculated the matrix $\mathbf{F}$ in the case concerning LQR with integral effect by solving the equation

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} + \mathbf{Gr} = (\mathbf{A} - \mathbf{BK})\mathbf{x} + (\mathbf{BF} + \mathbf{G})\mathbf{r} = \mathbf{0}. \tag{15}$$

This yields the equations

$$-\dot{p} = 0 \tag{16}$$

$$K_1k_{21}p + K_1k_{22}\dot{p} + K_1k_{23}\dot{e} + K_1k_{24}\gamma + K_1k_{25}\zeta = K_1F_{21}p_c + K_1F22\dot{e_c} \tag{17}$$

$$K_2k_{11}p + K_2k_{12}\dot{p} + K_2k_{13}\dot{e} + K_2k14\gamma + K_2k_{15}\zeta = K_2F11p_c + K_2F12\dot{e_c} \tag{18}$$

q_1	q_2	q_3	q_4	q_5	label
60	40	75	80	50	LQR_1
60	40	75	100	50	LQR_2
60	40	75	200	50	LQR_3
50	40	75	200	50	LQR_4
70	40	75	200	50	LQR_5
70	40	75	100	50	LQR_6

Table 6: Different values for q_1 and q_4. The labels refer to figure 12.

F_{11}	F_{12}	F_{21}	F_{22}	Label
0	1	1	0	F1
0	30	50	0	F2
0	10.34	17.11	0	F3

Table 7: Different values for the matrix F tested. The label refers to label in the figure.

$$p = p_c \tag{19}$$

$$\dot{e} = \dot{e_c} \tag{20}$$

As equations 19 and 20 are independent of the matrix **F**, the issue of stationary error is removed. As the matrix **F** ensures that the stationary error is minimized, this property is no longer needed when introducing integral effect, meaning that we can choose the matrix F freely. To test this hypothesis we examined the different values of the **F** matrix shown in table 7, with corresponding responses in figure 13. From the plot we can see that choosing **F** freely, in fact does not affect the standard deviation, although it affects the transient response. Due to this, how we design **F** must be taken into account in order to gain an optimal response. Using smaller values for **F** yields a slow response, while using higher values for **F** gives a fast response with a significant overshoot. Therefore, we chose to keep the **F** matrix as we derived for the LQR without integral effect in the preparations.

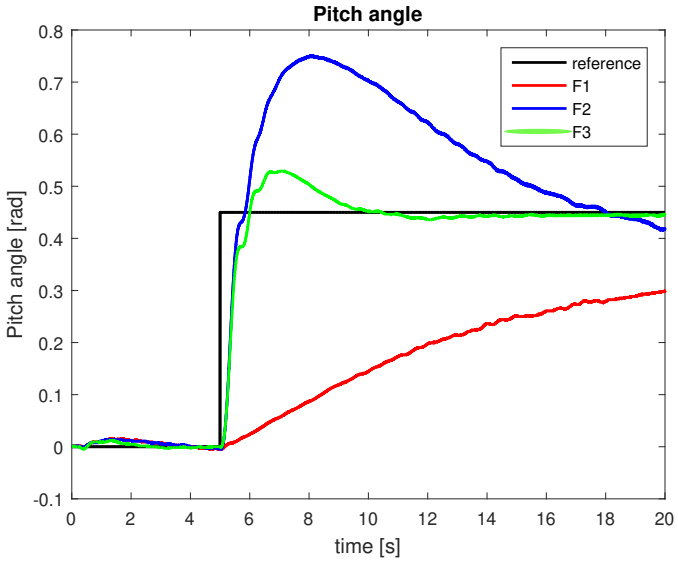


Figure 13: Plot of the response with different F-matrices.

4 Part 3 - Luenberger Observer

As we want to be able to control a free-flying helicopter, the encoder values cannot be used as a direct measurement because it requires the helicopter to be stuck to ground. We are therefore going to use an inertial measurement unit (IMU) instead of the encoder for measurements. However, the IMU measurement cannot be directly used as feedback to the controller, as it yields noisy signals. To prevent noisy signals going directly into the feedback, we utilize a Luenberger observer to estimate states, which is used as feedback to the controller.

4.1 Task 1 - IMU characteristics

In this task the IMU is used to measure the states of the helicopter. The IMU contains an accelerometer and a gyroscope. We will look at the data from the IMU by manually moving the helicopter and scope the responses in Matlab.

To begin with, we held the helicopter at the linearization point and rotated it only around one axis at the time. To see how the gyroscope output compares to the encoder rates, we plotted the gyroscope values against the encoder rates for the elevation rate, pitch rate and travel rate. In the figures of the elevation rate, pitch rate and travel rate, figure 14, 15 and 16 respectively, we can see that the gyroscope output and encoder rates are almost identical. Although it appears that the encoder values are more noisy than the gyro values in these plots, normally the IMU signals are more noisy. This is caused by the lack of voltage supply as we move the helicopter manually. In this case, the IMU measurements are not affected by motor vibrations as they would be when connecting the voltage. As we will see later, in section 4.4, the IMU signals has in fact significant noise compared to the encoder measurements.

Moreover, we started at the linearization point holding the helicopter horizontally. Thereafter, we moved the helicopter up and down, thus changing the rotation of the elevation axis. After roughly five seconds, we rotated the pitch with 90 degrees, while still moving the helicopter up and down. In figure 17 we have plotted the gyroscope output of the elevation rate against the encoder output for travel rate. Observe that during the five first seconds, while only rotating the helicopter around the elevation axis, the gyroscope measurement follows the elevation rate. When adding the 90 degree pitch rotation, the gyroscope output does no longer follow the change in elevation rate. Notice that the gyroscope output is now more similar to the change in travel rate. This happens as a result of the gyroscope being fixed to the pitch arm, causing the z- and y-axis to change when altering the pitch angle. As a consequence of this, the gyroscope tracks the incorrect axes when the pitch is rotated.

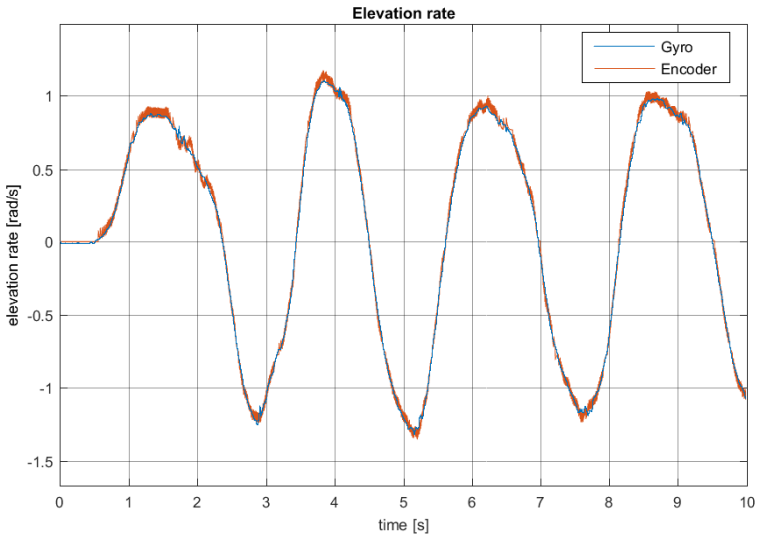


Figure 14: Gyroscope and encoder comparisons of elevation rate when manually moving the helicopter.

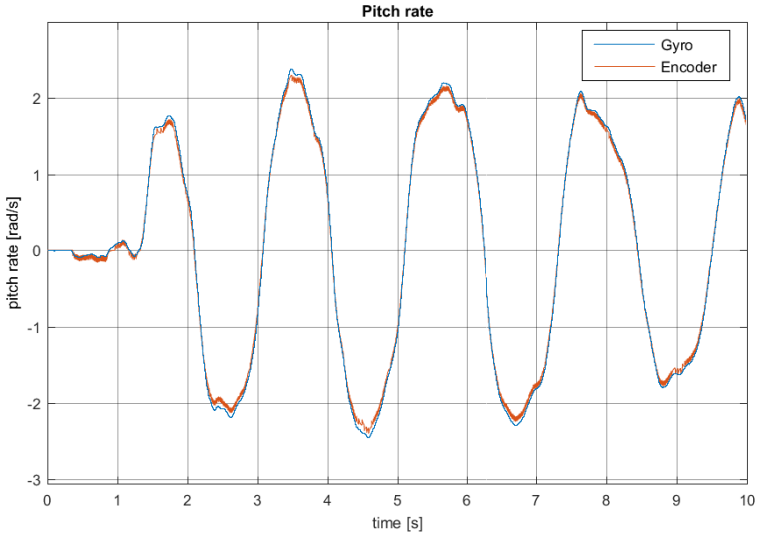


Figure 15: Gyroscope and encoder comparisons of pitch rate when manually moving the helicopter.

By examining the acceleration output while doing the same movement, we observe in figure 18 that when starting the helicopter from the table, $a_z = -9.81 m/s^2$ and $a_x \neq 0$ at $t = 0$. a_x is not equal to zero, because the x-axis of the helicopter is not horizontal while laying on the table, and is therefore affected by the gravitational pull. During the five first seconds of the simulation, while only moving the helicopter up and down, we see that

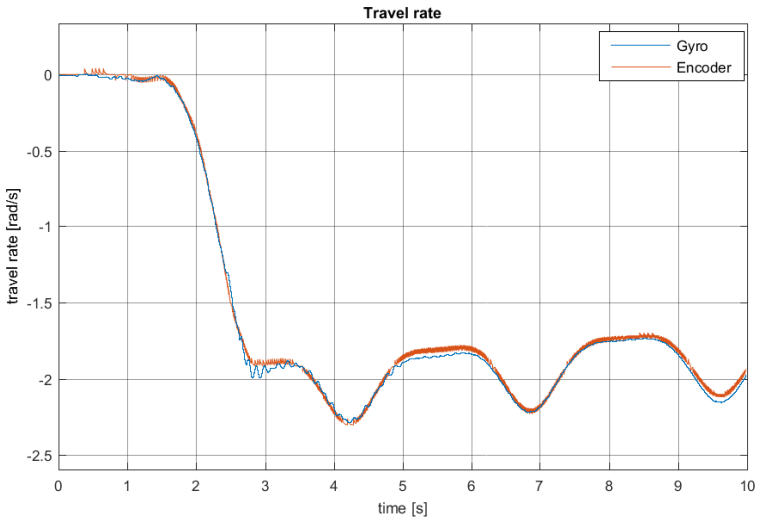


Figure 16: Gyroscope and encoder comparisons of travel rate when manually moving the helicopter.

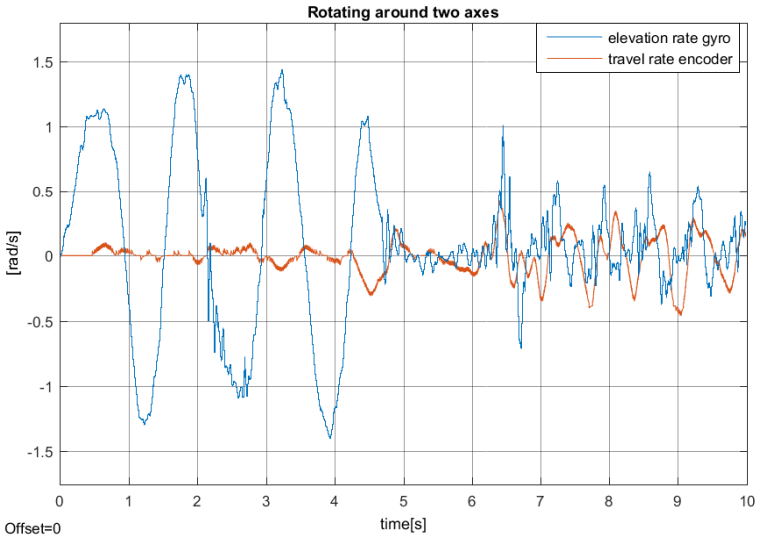


Figure 17: Gyroscope measurement of the elevation rate compared to the encoder measurement of travel rate while rotating around the elevation and pitch axes.

a_z follows the change in elevation rate, as expected. However, when rotating the pitch, we observe that a_z and a_y is swapped, meaning that a_z tracks the measurement in travel rate instead of elevation rate, and a_y tracks the elevation rate instead of travel rate. This reflects that our findings in the last paragraph also applies to the accelerometer output, as the accelerometer is

also fixed at the pitch head.

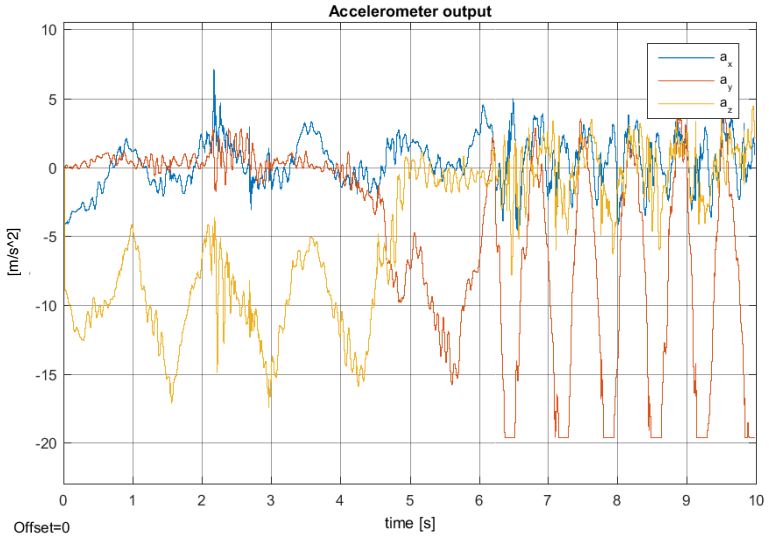


Figure 18: Accelerometer output while changing elevation angle and pitch angle.

4.2 Task 2 - Transformations

From our observations in 4.1, we can not use the gyro-measurements directly as they did not provide the correct results when the helicopter was rotated. Therefore, we apply a rotation that uses the pitch and elevation angles to calculate the correct pitch-, elevation-, and travel-rate. The block that perform this cumbersome calculation has been provided by the course staff. To indirectly measure the orientation of the helicopter through the accelerometer, we implement equations for pitch and elevation, as derived in the preparations. This resulted in the following equations:

$$p = \arctan(\frac{a_y}{a_z}) \tag{21}$$

$$e = \arctan(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}) \tag{22}$$

The implementation of the equations for pitch angle and elevation angle is shown in figure 19. During the start-up of the program, measurements from the IMU will be zero, leading to a division by zero. Therefore, the implementation also include logic such that the output will be zero when a division by zero happens.

By comparing the transformed accelerometer output for pitch to the encoder pitch angle in figure 20, we observe that the measurements are quite

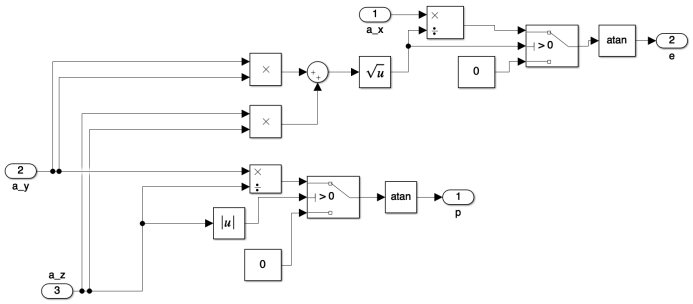


Figure 19: The transformation of pitch and elevation angle implemented in simulink.

similar, except for a small offset in the IMU measurements. This also the case for elevation. The same observations applies in figure 21, when comparing the pitch rate gyroscope to the encoder. Thus, we can confirm that the IMU conversions are correct.

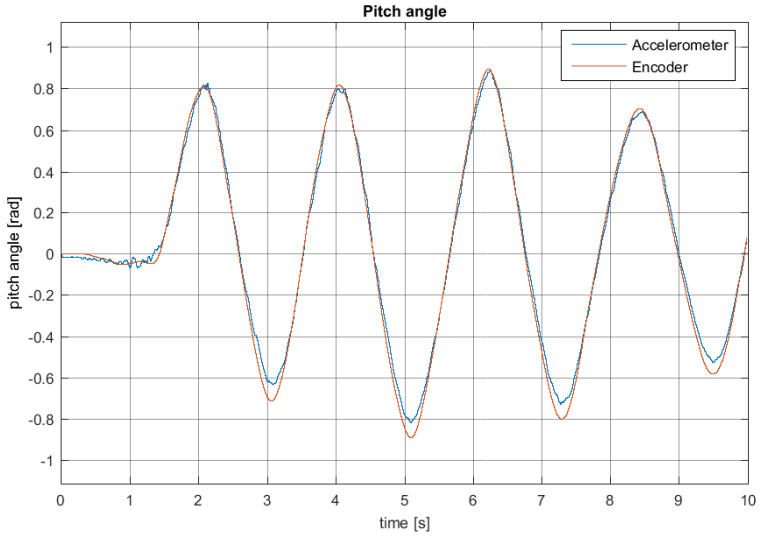


Figure 20: Plot of the transformed accelerometer output for pitch, and the encoder pitch angle.

Furthermore, when comparing the results from the transformed gyroscope outputs in figure 22 to the gyroscope measurements in figure 17, observe that by transforming the gyroscope measurements we have removed the issue concerning incorrect tracking of the axes when the pitch is rotated.

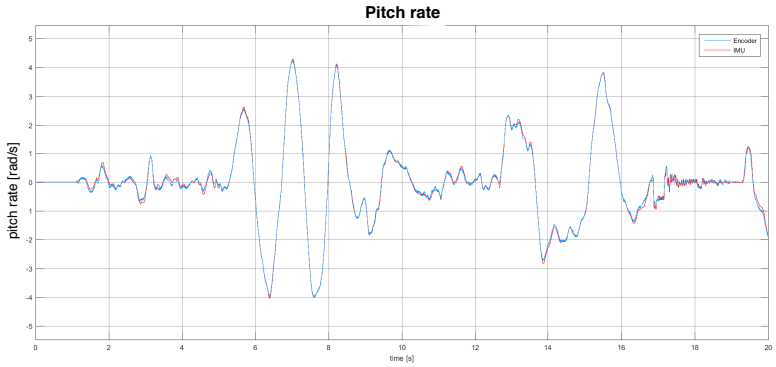


Figure 21: Plot of gyro output for pitch rate and the encoder pitch rate.

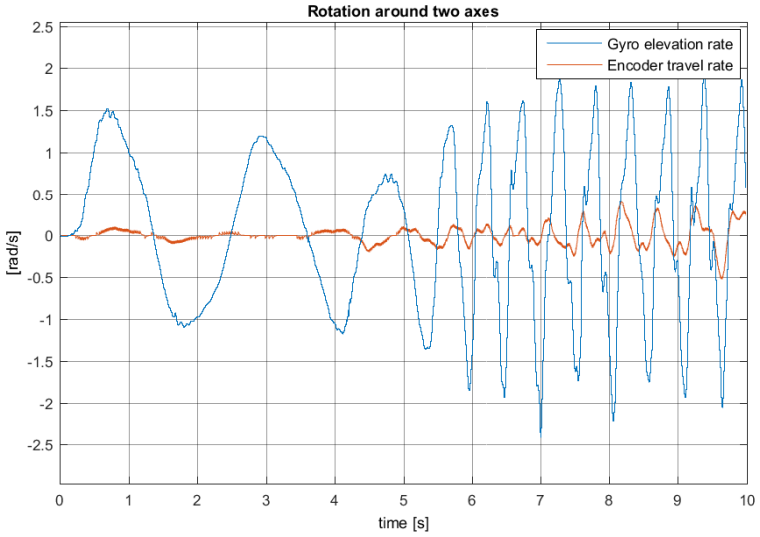


Figure 22: Transformed gyroscope measurement of the elevation rate compared to the encoder measurement of travel rate while rotating around the elevation and pitch axes.

4.3 Task 3 - Theoretical discussion

For a state to be observable, knowledge of the measurement $\mathbf{y}$ and the input $\mathbf{u}$ must uniquely determine the initial state $\mathbf{x}(0)$ [1, p. 153]. In the preparations, we derived that the minimum set of states required to be measured in order to make the system observable, is by measuring the states e and $\dot{\lambda}$. This can be explained by analyzing the equation of motion for the helicopter. By examining equations 3a and 3b, we observe that $\dot{\lambda}$ cannot be determined from the equations of elevation nor pitch. Hence, it is impossible to determine the travel rate without measuring it explicitly. Likewise, it is not possible to determine elevation and elevation rate without measuring the elevation

directly. However, by analyzing equation 3c, it is possible to derive the pitch and pitch rate from the measurement of $\dot{\lambda}$, meaning that the pitch is not required to be measured explicitly. Therefore, in theory, we only need to measure the states e and $\dot{\lambda}$ in order for the system to be observable.

Even though it theoretically is possible to extract information about the pitch from the travel rate, it was difficult to manage the helicopter by only measuring e and $\dot{\lambda}$ in practice. Simplifications in the modelling and linearization can cause these discrepancies, as some information is lost through these processes. In addition, to derive the pitch and pitch rate from the travel rate, we must differentiate travel rate not only once, but twice. The IMU measurement of travel rate includes high frequent noise. When differentiating highly frequent signals, we amplify the noise each time it is differentiated, making it difficult to get a good response when not measuring pitch and pitch rate directly. If it is possible to measure all the desired states, this is always preferred as this will make the response more accurate. We therefore measured all the states to get the desired response.

As we derive the equations for pitch and elevation angle, we utilize that the accelerometer measures the force counteracting gravity when the helicopter is stationary. This force will always point straight upwards with a magnitude equal to the gravitational constant g. This means that the accelerometer measurements will be correct when the helicopter is stationary, but not necessarily when the helicopter is moving. Furthermore, when using a high control signal, the system is more affected by noise from the motor vibrations. If the motor vibrations do not affect elevation, pitch and travel equally, this might lead to incorrect outputs from the accelerometer.

By examining the equations 21 and 22 for pitch and elevation, observe that one might divide by zero if either $a_z = 0$, or $a_z = 0$ and $a_y = 0$. In our implementation of the equations in Simulink, we are treating this case by setting the corresponding angle to zero if a division by zero occurs. By doing so, we might get pitch or elevation angles that does not apply to the actual physical behavior of the system. However, our experimentation shows that the model works well, and the assumptions we have used when deriving pitch and elevation angle does not critically affect the behavior of the response.

4.4 Task 4 - State estimator

In this task, a state estimator will be used to smooth the signal from the IMU. We will be using the Luenberger observer, given by the equation

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \tag{23}$$

where

$$\dot{\hat{\mathbf{x}}} = \begin{bmatrix} \dot{p} \\ \ddot{p} \\ \ddot{e} \\ \dddot{e} \\ \ddot{\lambda} \end{bmatrix} \tag{24}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ K_3 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{25}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{26}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{27}$$

Figure 23 shows the implementation of the observer in Simulink.

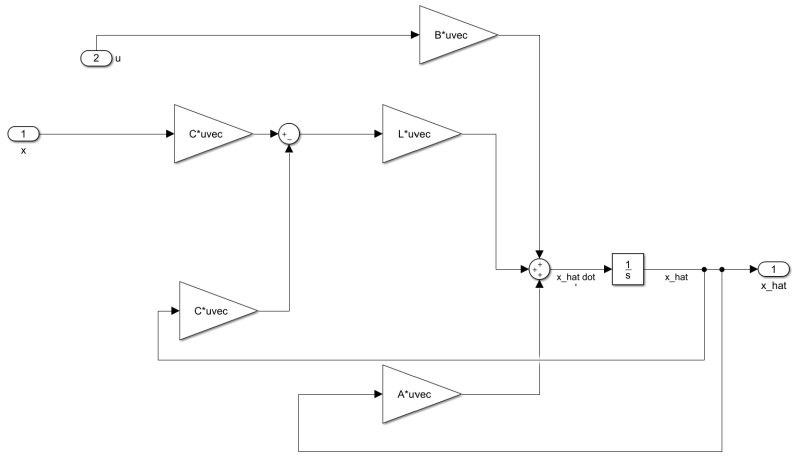


Figure 23: Implementation of the Luenberger observer in Simulink.

In the experimentation, the LQR controller with integral effect is used. We design the $\mathbf{L}$ matrix by using the Matlab command `L = place(A',C',p)'`, where p is a vector of five poles. Theoretically, pole placement yields the best result when placed on a half-circle in the left half-plane with equal distance between each pole, only altering the radius of the half-circle and the angle between the poles [1, p.238]. By magnifying the radius of the circle, we gain a faster response as the poles are further away from the imaginary axis. By altering the angle between the poles, we determine the amount of overshoot in the response. The larger the angle, the larger the overshoot. As this type of placement yields the best result, we use this strategy for pole placement in the experimentation.

As we wish to use the estimation of pitch, pitch rate and elevation rate as feedback to the controller, we chose to focus on these states in the experimentation and tuning. Furthermore, pitch and pitch rate had similar responses in the experimentation, for any pole placements, which also applies for elevation and elevation rate. As the rates have more noise in the measurements, our discussion is focused on pitch and elevation, while keeping in mind that the results are valid for the corresponding rates as well.

To determine the optimal values for the angle θ and the radius r, the principle of binary search was used. To begin with, we experimented with keeping the radius constant at $r = 1$ and varied θ. With $\theta = 30$, the estimation of the pitch was nearly perfect. However, the estimation of the elevation deviated significantly, as shown in figure 24. With $\theta = 5$, the estimation of the elevation was still unsatisfactory, while the pitch estimation was nearly unchanged. We therefore tested with $\theta = 15$. This led to less deviation in the estimation of elevation. Yet, as there is a delay in the estimation response compared to the encoder with $r = 1$, we wanted a faster estimator and therefore increased the value of r. We experimented with $r = 10$, which is displayed in figure 25. This resulted in a good estimation of the elevation, which followed the encoder value well. However, it also introduced some oscillations, as the poles are placed further into the left half-plane. In addition, this reduced the accuracy in the pitch estimation slightly. Due to these observations, we tested with $r = 5$, which yielded a poorly estimation of elevation. We therefore accepted some small oscillations and a slightly less accurate pitch estimation in order to maintain a good estimation in elevation. Thus, we chose $r = 10$ and $\theta = 15$ in order to maintain a fairly good estimation of all states.

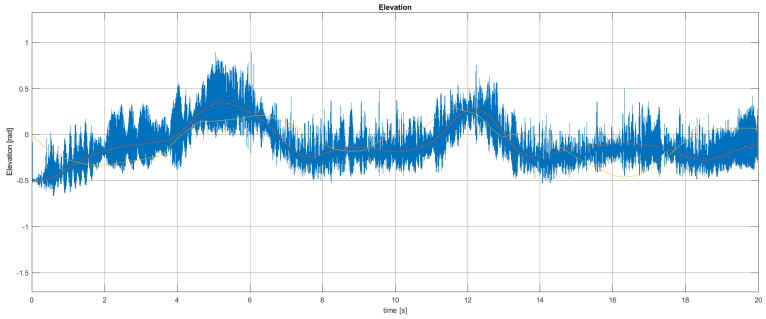


Figure 24: Plot of the estimated elevation angle with the IMU and encoder values when $r = 1$ and $\theta = 30$.

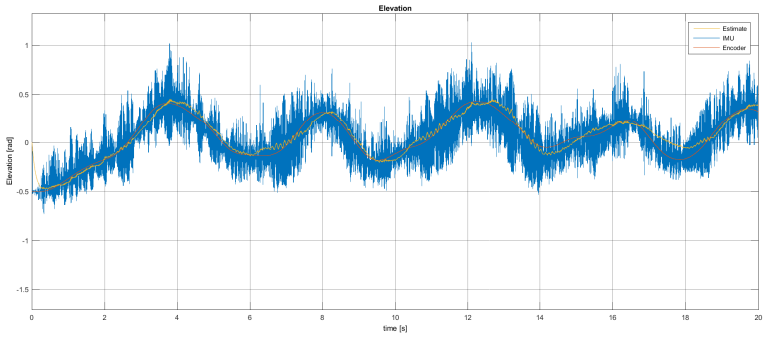


Figure 25: Plot of the estimated elevation angle with the IMU and encoder values when $r = 10$ and $\theta = 15$.

5 Part 4 - Kalman filter

Instead of using the Luenberger observer to estimate states to the feedback of the controller, we are going to implement the Kalman filter. The difference between the Kalman filter and Luenberger observer is that the Kalman filter models and considers the noise when choosing how much to trust the measurements and the prediction [5, p. 141].

The following discrete-time model will be used to describe the system.

$$\mathbf{x}[k+1] = \mathbf{A_d}\mathbf{x}[k] + \mathbf{B_d}\mathbf{u}[k] + \mathbf{w_d}[k] \tag{28}$$

$$\mathbf{y}[k] = \mathbf{C_d}\mathbf{x}[k] + \mathbf{v_d}[k] \tag{29}$$

$$\mathbf{w_d} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q_d}), \mathbf{v_d} \sim \mathcal{N}(\mathbf{0}, \mathbf{R_d}) \tag{30}$$

5.1 Task 1 - Noise estimate

We are going to find an estimate for the measurement noise, $\mathbf{R_d}$. To do this, we first produced an experimental time-series when the helicopter was laying on the ground. Then, we used the LQR controller with integral effect to keep the helicopter stationary at the linearization point, and produced an experimental time-series while the helicopter was flying. We plotted all the states in different plots to see the noise in each state clearly. The noise measurement in pitch while laying on the ground and while flying at the linearization point is shown in figure 26 and figure 27, respectively. These figures show that noise appears in both cases, but the noise while flying is almost 100 times larger than while laying on the ground. When laying on the ground, the helicopter is not affected by the motors or any other vibrations, which will affect the helicopter when it is flying. This also applied to the rest of the states, and is therefore not shown in the report.

If we consider the variance of the noise in the plots, we can characterize the type of noise. From theory, we know that white noise have mean equal to 0 [5, p. 77]. To find the exact means from the plots, we used the Matlab function `mean(A)`. For instance, the mean of pitch was -0.0121. The other states had values varying closely around zero as well. Since the mean value is nearly zero, we characterize the noise as white noise.

We then determined the covariance for the helicopter while laying on ground and while flying at the linearization point. The covariance describes the variance in our measurements. We used the Matlab function `cov(A)` to evaluate the signal covariances. We obtained the following covariance matrices:

$$\mathbf{Cov_{ground}} = \begin{bmatrix} 9.77 \cdot 10^{-7} & - & - & - & - \\ - & 6.53 \cdot 10^{-7} & - & - & - \\ - & - & 5.43 \cdot 10^{-5} & - & - \\ - & - & - & 4.89 \cdot 10^{-7} & - \\ - & - & - & - & 7.11 \cdot 10^{-7} \end{bmatrix} \tag{31}$$

$$\mathbf{Cov_{flying}} = \begin{bmatrix} 0.0056 & - & - & - & - \\ - & 0.0030 & - & - & - \\ - & - & 0.0290 & - & - \\ - & - & - & 0.0089 & - \\ - & - & - & - & 0.0781 \end{bmatrix} \tag{32}$$

As we can see from the covariance matrices 31 and 32, the diagonal values of the covariance matrix in the flying case are much larger than when laying on the ground. This is also reflected in the figures 26 and 27, as the noise varies significantly more when the helicopter is flying compared to when the helicopter lays on the ground. We will use the covariance matrix while flying as the estimate of the measurement noise, $\mathbf{R_d}$.

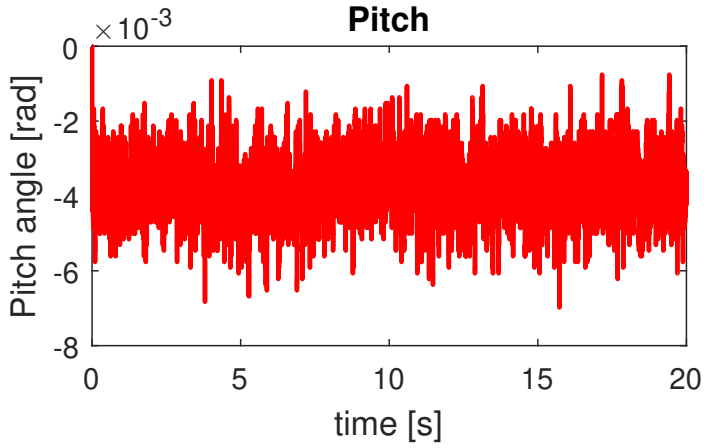


Figure 26: Plot of measured pitch covariance while helicopter laying on ground.

5.2 Task 2 - Discretization

In the remainder of the lab, we will be using the full state-space model with $\mathbf{x} = [p\ \dot{p}\ e\ \dot{e}\ \lambda\ \dot{\lambda}]$. By introducing a new state λ to the system, the new matrices for the system is given as

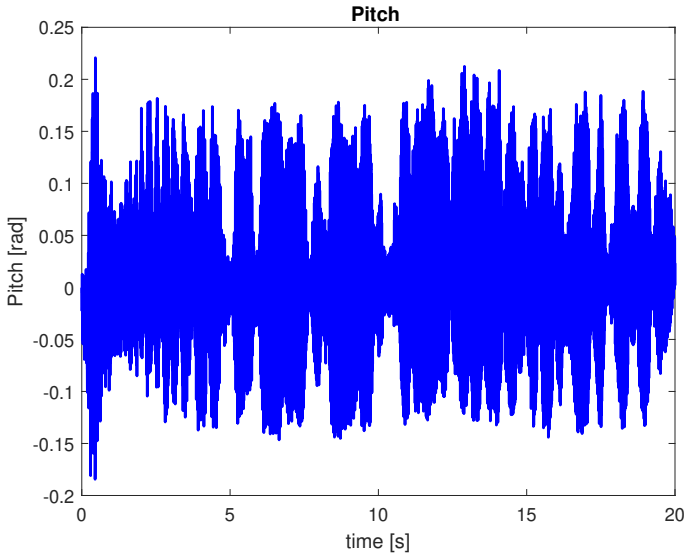


Figure 27: Plot of measured pitch covariance while helicopter flying at the linearization point.

$$\underbrace{\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \\ \dddot{\lambda} \\ \ddot{\lambda} \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ K_3 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \\ \dot{\lambda} \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} \tilde{V}_s \\ V_d \end{bmatrix}}_{\mathbf{u}} \tag{33}$$

To obtain the matrices $\mathbf{A_d}$ and $\mathbf{B_d}$, the continious-time system is discretized. As the observability of the system is unchanged, it is not necessary to discretize $\mathbf{C}$, resulting in $\mathbf{C_d} = \mathbf{C}$. In order to discretize the system, the Matlab function `c2d` is used. The Matlab script for discretizing the matrices is the following:

```
T = 0.002;
sys = ss(A_kalman,B_kalman,C_kalman,0);
sys_d = c2d(sys, T);
[A_d,B_d,C_d,~] = ssdata(sys_d);
disp(A_d);
disp(B_d);
disp(C_d);
```

This results in

$$\mathbf{A_d} = \begin{bmatrix} 1 & 0.0020 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0020 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1.23 \cdot 10^{-6} & 8.16 \cdot 10^{-10} & 0 & 0 & 1 & 0.0020 \\ 0.0012 & 1.22 \cdot 10^{-6} & 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

$$\mathbf{B_d} = \begin{bmatrix} 0 & 7.15 \cdot 10^{-8} \\ 0 & 7.15 \cdot 10^{-5} \\ 1.64 \cdot 10^{-7} & 0 \\ 1.64 \cdot 10^{-4} & 0 \\ 0 & 1.46 \cdot 10^{-14} \\ 0 & 2.91 \cdot 10^{-11} \end{bmatrix} \tag{35}$$

$$\mathbf{C_d} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{36}$$

5.3 Task 3 - Implementation

We wish to use a discrete-time Kalman filter to estimate the states of the system. The Kalman filter is described by the following equations.

Correction with new data:

$$\mathbf{K}[k] = \overline{\mathbf{P}}[k]\mathbf{C_d}^T(\mathbf{C_d}\overline{\mathbf{P}}[k]\mathbf{C_d}^T + \mathbf{R_d})^{-1} \tag{37}$$

$$\hat{\mathbf{x}} = \overline{\mathbf{x}} + \mathbf{K}[k](\mathbf{y}[k] - \mathbf{C_d}\overline{\mathbf{x}}[k]) \tag{38}$$

$$\hat{\mathbf{P}}[k] = (\mathbf{I} - \mathbf{K}[k]\mathbf{C_d})\overline{\mathbf{P}}[k](\mathbf{I} - \mathbf{K}[k]\mathbf{C_d})^T + \mathbf{K}[k]\mathbf{R_d}\mathbf{K}^T[k] \tag{39}$$

Predicting ahead:

$$\overline{\mathbf{x}}[k+1] = \mathbf{A_d}\hat{\mathbf{x}}[k] + \mathbf{B_d}\mathbf{u}[k] \tag{40}$$

$$\overline{\mathbf{P}}[k+1] = \mathbf{A_d}\hat{\mathbf{P}}[k]\mathbf{A_d}^T + \mathbf{Q_d} \tag{41}$$

To implement the Kalman filter we implemented the prediction step and the correction step in two separate Matlab function blocks. As the data from the IMU arrives at a slower rate than the update frequency of the Simulink program, when there is no new data, the estimated state $\hat{\mathbf{x}}[k]$ is set to the predicted state $\overline{\mathbf{x}}[k]$ and the estimated error covariance $\hat{\mathbf{P}}[k]$ is set to the predicted error covariance $\overline{\mathbf{P}}[k]$.

This resulted in the following Matlab functions for step prediction and correction step:

```
function [xk_next, P_pred_next] = prediction_step(xk_est, uk,
A_d, B_d, Q_d, P_est)

xk_next = A_d*xk_est+B_d*uk;
P_pred_next = A_d * P_est * transpose(A_d) + Q_d;
```

```
function [xk_est, P_est] = correction_step(xk_pred,
C_d, R_d, P_pred,y,new_data)

if (new_data)
    Kk = P_pred*transpose(C_d)*inv(C_d*P_pred*transpose(C_d)
    + R_d);
    xk_est = xk_pred + Kk*(y-(C_d*xk_pred));
    P_est = (eye(6)-Kk*C_d)*P_pred*transpose((eye(6)-Kk*C_d))
    +Kk*R_d*transpose(Kk);
else
    xk_est = xk_pred;
    P_est = P_pred;
end
```

The implementation of these functions in Simulink is shown in figure 28.

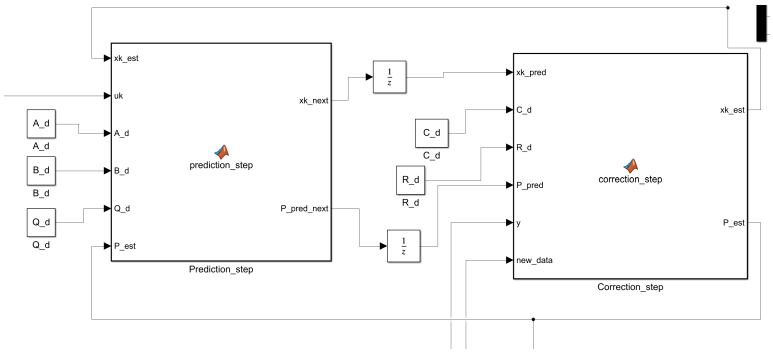


Figure 28: Implementation of the Kalman filter in Simulink.

5.4 Task 4 - Experimentation

The matrix $\mathbf{Q_d}$ describes the uncertainty in our model. In the experimentation we use the LQR regulator with integral effect with the encoder values as feedback. As shown in equation 30, the disturbance $w_d[k]$ is assumed to be white, with a Gaussian distribution around 0 and stochastic disturbance matrix $\mathbf{Q_d}$. The matrix $\mathbf{Q_d}$ is undetermined and left to be tuned by experimentation. The $\mathbf{Q_d}$ matrix adjusts the weight of the diagonal elements according to the response in each state.

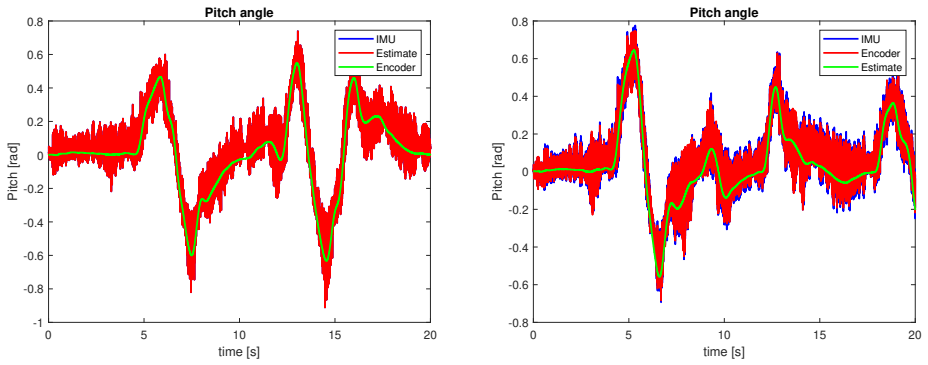
We experimented with different diagonal $\mathbf{Q_d}$ matrices and compared the estimated states $\hat{\mathbf{x}}$ with the encoders and the IMU measurements. The first

step in the experimentation was to choose the diagonal elements such that the weight of each state was determined by the values on the diagonal. As the rate signals are more noisy than the angles, we aimed to place less emphasis on the diagonal values that corresponds to the rates, making these states rely more on the prediction than the measurements. We prioritized to weight the pitch more than elevation, as we experienced that when relying the states on prediction, the pitch had tendencies to become oscillatory and unstable, to a greater extent that the other states. Moreover, as the IMU does not measure the travel, meaning that travel relies on the prediction regardless of the entry, we chose the corresponding entry in $\mathbf{Q_d}$ to zero. This resulted in the $\mathbf{Q_d}$ matrix given in 42.

$$\mathbf{Q_d} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix} \tag{42}$$

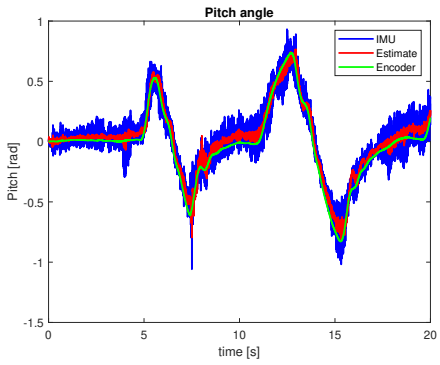
The next step of the experimentation was to determine to which extent the states should depend on the measurements versus the prediction. This was done by multiplying the $\mathbf{Q_d}$ matrix by powers of 10. From the subfigures in the plot 29, we observe that as we lower the power multiplied by the $\mathbf{Q_d}$ matrix, the estimation gets less noisy. However, when we lower the power of the matrix by a factor of 10^{-7}, the estimation does not follow the measurements as accurately as desired. Due to this, $\mathbf{Q_d} \cdot 10^{-5}$ seems to be the most appropriate magnitude for the disturbance matrix in this experiment. Nevertheless, from this experimentation, we can see that for higher values of $\mathbf{Q_d}$, the estimate resembles to the measurements, while for lower $\mathbf{Q_d}$ values, it resembles to the encoder value. This hypothesis fits well with theory, as we will discuss in the following paragraph.

From equation 41 we observe that $\mathbf{Q_d}$ affects the size of the weighting matrix $\mathbf{K}[k]$ in equation 37. This means that $\mathbf{Q_d}$ describes the relation between the measured states and the predicted states. Increasing $\mathbf{Q_d}$ will make the filter weight the measured states more, and decreasing $\mathbf{Q_d}$ makes the filter weight the predicted states more. Setting all the elements of $\mathbf{Q_d}$ to 0, the model would only rely on the predicted states. This means that setting $\mathbf{Q_d}$ to $\mathbf{0}$ gives an open loop estimator. Setting all the elements of $\mathbf{Q_d}$ to infinity, the model only relies on the measurements. We started testing this hypothesis by setting all the elements in $\mathbf{Q_d}$ to 0, resulting in the figure 30. We can see from the figure that the model is sensitive to disturbances and has a slow convergence rate, thus it operates as an open loop estimator. Figure 31 plots the response when the elements on the diagonal equals 100 000. As we can see, $\hat{\mathbf{x}}$ follows the IMU signal quite perfectly. This implies

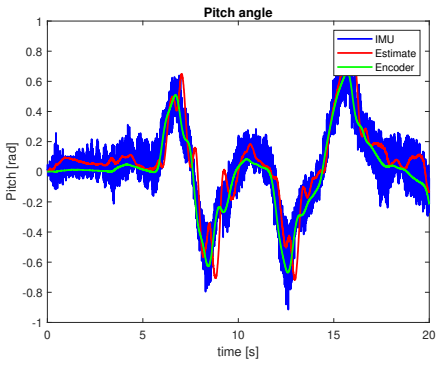


(a) $\mathbf{Q_d} \cdot 10^0$

(b) $\mathbf{Q_d} \cdot 10^{-2}$



(c) $\mathbf{Q_d} \cdot 10^{-5}$



(d) $\mathbf{Q_d} \cdot 10^{-7}$

Figure 29: Plots of the response with the $\mathbf{Q_d}$ matrix, defined in equation 42, in different orders of magnitude.

that large values on the diagonal of $\mathbf{Q_d}$ makes the estimated states rely only on the measurement, as expected.

Furthermore, we added a manual switch to the new-data signal, making it possible to artificially connect and disconnect it while the program is running. As we disconnect the new-data signal, only the prediction step will affect the response, as the the correction step of the Kalman filter will not be updated. This corresponds to an open loop estimator, similarly to setting $\mathbf{Q_d}$ to $\mathbf{0}$.

Figure 32 displays the plot of estimated state for pitch angle when connecting and disconnecting the new-data signal. As the new-data signal is connected, the noisy signals from the IMU is represented in the plot, as the correction step relies on the measurements in addition to the model. However, when disconnecting the new-data signal, the response only relies on the model, removing the noisy signals. As we disconnect the new-data signal the estimated state will drift. This can be due to imperfect modeling.

The error covariance matrix quantifies the uncertainty in the estimate.

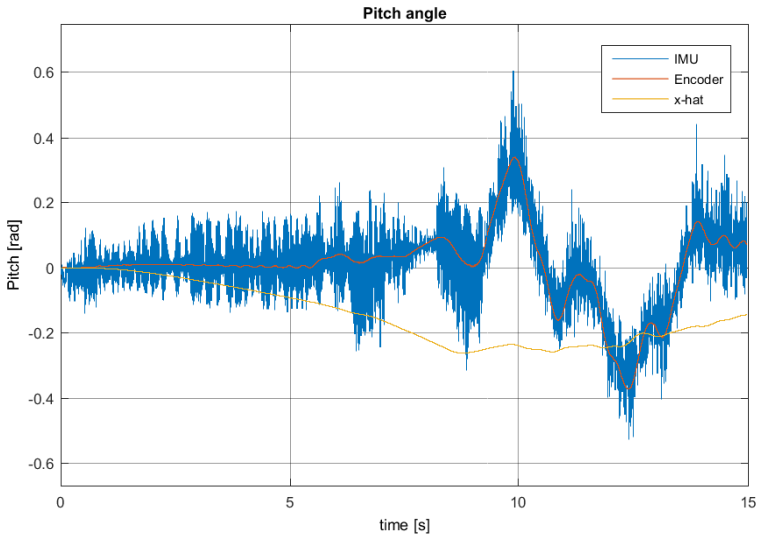


Figure 30: Plot of the estimated state $\hat{x}$ when setting all the elements in $\mathbf{Q_d}$ to 0, compared to IMU and encoder measurements.

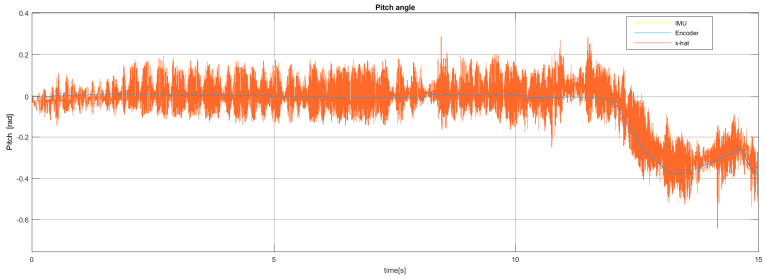


Figure 31: Plot of the estimated state $\hat{x}$ when setting all the elements in $\mathbf{Q_d}$ to 100 000, compared to IMU and encoder measurements.

As we see from figure 33, the error covariance is nearly zero when the new-data signal is connected, as we correct the estimate according to the measurements. As we disconnect the new-data signal, the error covariance increases, until we connect the new data signal. This can be explained by examining equation 41. For each time we predict the new state, the uncertainty $\mathbf{Q_d}$ is added to the measurements, meaning that the uncertainty increases for each time step.

When connecting the Kalman filter as feedback to the encoder by using the $\mathbf{Q_d}$ matrix given in 42 gained with a power of 10^{-5}, the pitch became oscillatory and unstable. As the estimation shall be 2-5 times faster than the controller, we tried to make the estimator more rapid by making the estimates rely more on the IMU measurements. After a lot of testing, we concluded that the matrix given in 43 gave the most satisfactory response.

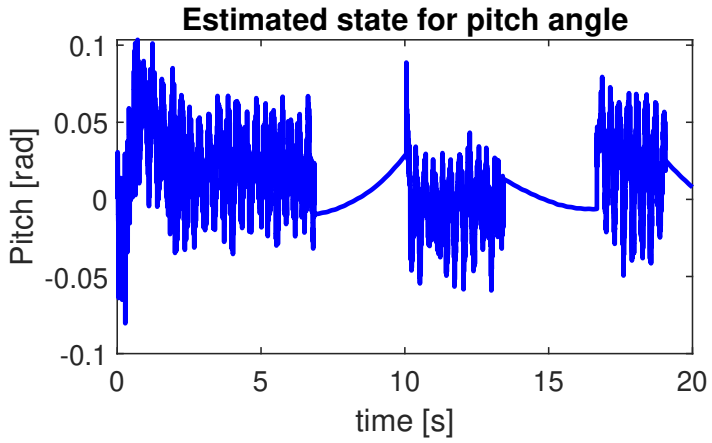


Figure 32: Plot of the estimated state for pitch angle when connecting and disconnecting the new-data signal.

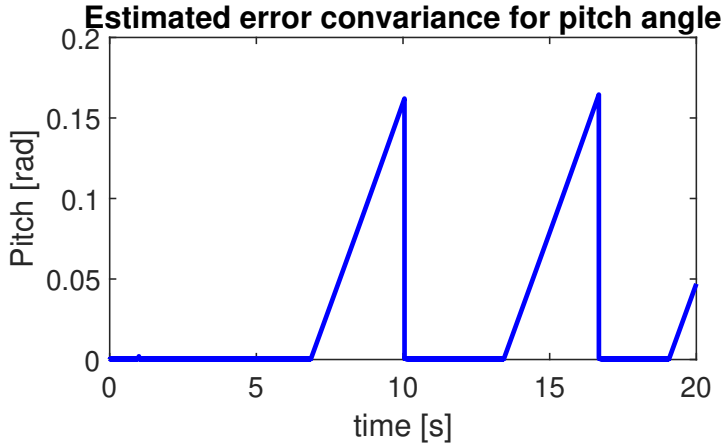


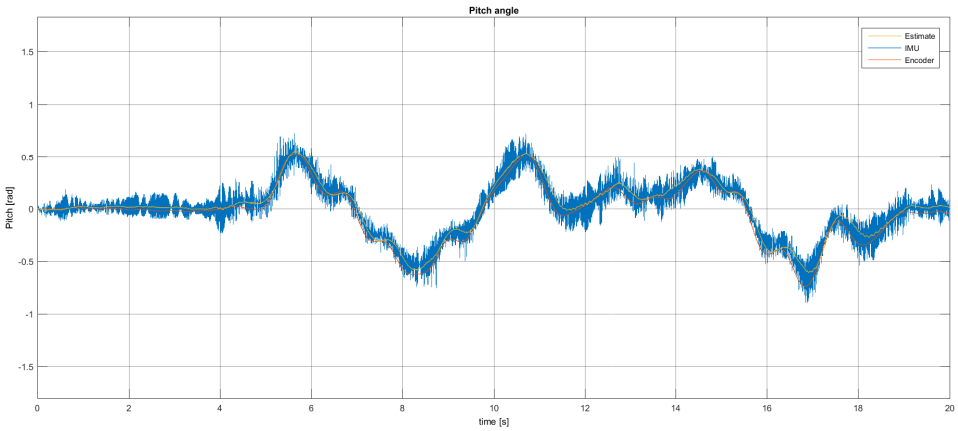
Figure 33: Plot of the estimated error covariance for pitch angle when connecting and disconnecting the new-data signal.

$$\mathbf{Q_d} = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0003 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0004 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.03 \end{bmatrix} \tag{43}$$

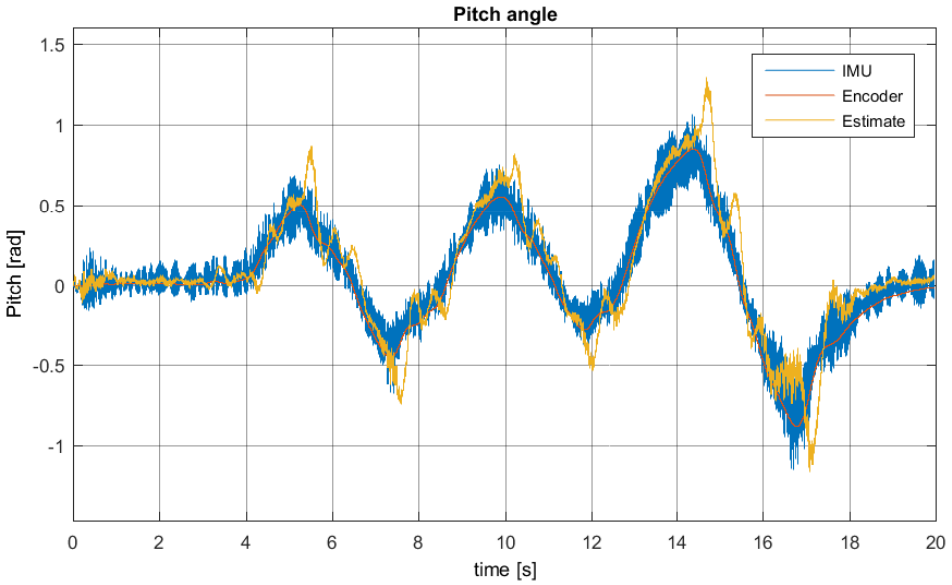
Throughout the lab, we have experimented with different estimators, namely the Luenberger observer and the Kalman filter. Both of the estimators use gain matrices to correct the error, $\mathbf{L}$ and $\mathbf{K}$, in order to get a correct estimation. The main difference between the Luenberger observer

and the Kalman filter is how the matrices **L** and **K** affect the estimation. The matrix **L** is designed by using pole placement, and determines the stability margin and the transient response of the estimator. Unlike this, the **K** matrix describes the weighting relation between the measurements and the predicted states. This is determined by the matrix $\mathbf{Q_d}$, which describes the uncertainty in our modeling. Having a higher uncertainty in the model means that we rely more on the measurements. Lower uncertainty means that we trust the predictions.

To conclude the report, we will compare the response when using the Luenberger observer as feedback to the controller to the response when using the Kalman filter. From the plot in figure 34, we can see that the estimation is remarkably more accurate when using the Luenberger observer than when using the Kalman filter. When designing the Kalman filter, we assumed that the noise was Gaussian distributed with mean value equal to zero. In section 5.1 we concluded that the mean value of the states was nearly zero, and we therefore classified the noise as white. However, as further experimentation yields poorer estimation when the filter relies more on the predicted states, the assumption of white noise may not hold, as we assumed. This may explain why the Kalman filter yields worse estimation than the Luenberger observer in our experiment.



(a) Result of the estimation when using the Luenberger observer.



(b) Result of the estimation when using the Kalman filter.

Figure 34: Plots of the responses when using the Luenberger and the Kalman filter as feedback to the controller.

References

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

[2] Balchen Jens G., Andresen Trond, and Foss Bjarne A. *Reguleringsteknikk*. Institutt for teknisk kybernetikk, NTNU, 2016.

[3] Sang-Hoon Kim. *Electric Motor Control*. Elsevier Inc., 2017.

[4] *LQR-note*. Sourced from Blackboard.

[5] Brown Robert Grover and Hwang Patrick Y.C. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley Sons, Inc., 2012.

[6] Roskilly Tony and Mikalsen Rikard. *Marine Systems Identification, Modeling and Control*. `https://doi.org/10.1016/C2013-0-18786-7`. Elsevier Ltd., 2015.