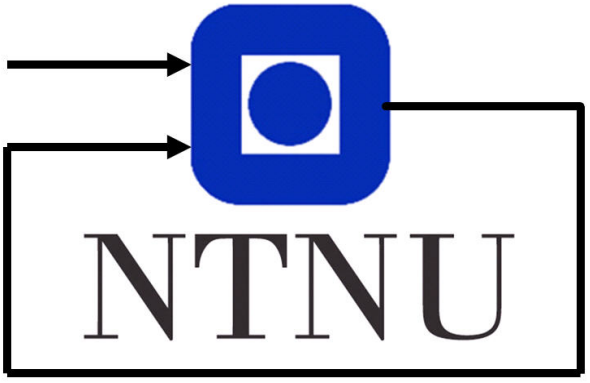
Helicopter lab assignment

Group 01
Olav Austrheim Stenså, 478021
Simen Stensrød Allum, 507743

Fall, 2020



Department of Engineering Cybernetics

Contents

1 Introduction

This is a report describing our work with the helicopter lab in TTK 4115 - "Linear system theory" at NTNU in the fall 2020. The lab consist of four lab days and has to do with control of a helicopter model. The lab covers a lot of important topics in TTK 4115, mainly:

- Basic modelling, linearization and state space models
- LQR and LQR with integral effect (augmented system)
- Observability
- Luenberger observer
- Discrete Kalman filter

The report is organized chronologically and structured in chapters corresponding to each lab day. The modelling of the system is given its own chapter. Each chapter is structured with any work done prior to the lab first, and then the lab assignments following in the order they were done. They are followed by a separate chapter containing solutions to problems we encountered in the lab. There is no appendix containing all our Matlab code and Simulink diagrams. Instead, parts of diagrams or pieces of code are included as figures in the report itself.

Our purpose at the lab was never to achieve perfect tuning or optimizing everything. Instead our focus has been to experiment with different tuning methods and to find pros and cons with these methods. In situations where we did not achieve the results we wanted/expected, the report focuses on why and most importantly how we would have done it differently the next time. In some cases we also propose changes to the system that may solve the problem.

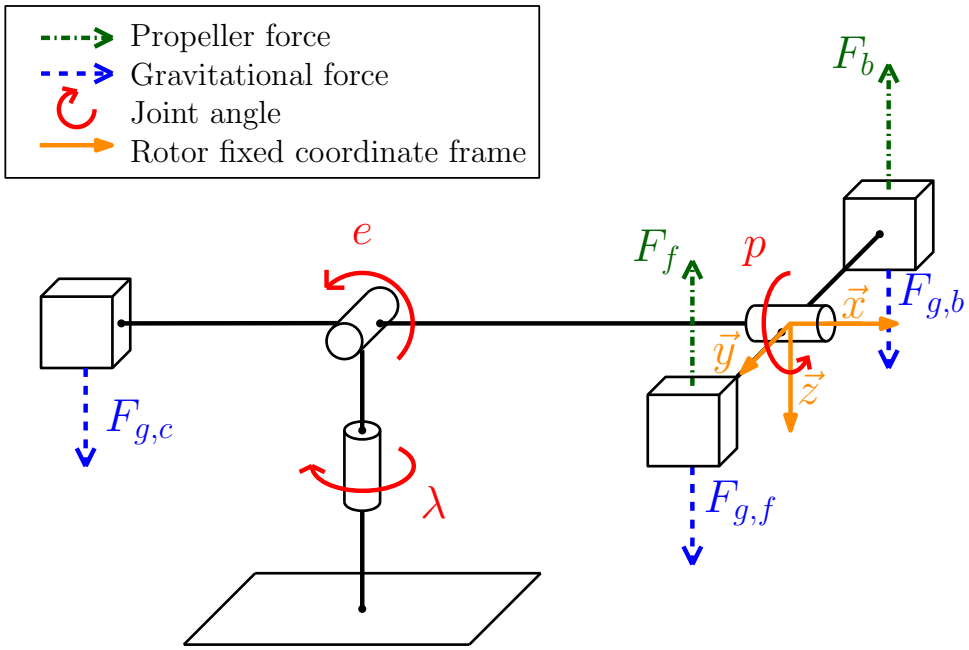


Figure 1: The helicopter setup with defined forces and angles [2]

2 System modeling

2.1 System mechanics

The helicopter systems has to be modelled mathematically. The helicopter is modelled as three point-masses; two point masses that represents the helicopter motors with the rotors, m_f and m_b, and one point mass that represents the counter weight, m_c. The two motor masses are connected to the elevation axis with a distance l_h and the counterweight is connected to the same axis with a distance l_c. The positive directions for the rotation around the axis as well as the direction of the coordinate system being used is as in figure 1. Furthermore it is assumed that there is a linear relationship between supplied voltage to the motors, V_f and V_b, and applied forces from the rotors, F_f and F_b (2).

$$F_f = K_f V_f \tag{1}$$
$$F_b = K_f V_b \tag{2}$$

We also define the difference and sum of the supplied voltages as in equation 4.

$$V_s = V_b + V_f \tag{3}$$
$$V_d = V_f - V_b \tag{4}$$

The helicopter can rotate in three different directions. The rotation around the vertical axis is the travel of the helicopter and is measured by the angle λ. p denotes the pitch angle of the helicopter head, and e denotes the elevation of the helicopter. If we ignore any friction in the joints and apply newtons second law of rotational motion, we get the set of three equations as given in equation 5 with constants given in equation 14.

$$J_p\ddot{p} = L_1 V_d \tag{5a}$$
$$J_e\ddot{e} = L_2\cos(e) + L_3 V_s\cos(p) \tag{5b}$$
$$J_\lambda\ddot{\lambda} = L_4 V_s\cos(e)\sin(p) \tag{5c}$$

$$\begin{aligned} L_1 &= l_p K_f \\ L_2 &= g(l_c m_c - 2l_h m_p) \\ L_3 &= l_h K_f \\ L_4 &= -L_h K_f \end{aligned} \tag{6}$$

2.2 Linearization

We want to linearize the system around the point where all the state variables are equal to zero $(p, e, \lambda) = (\dot{p}, \dot{e}, \dot{\lambda}) = (0,0,0)$. We must first choose appropriate values $(V_{s,0},\ V_{d,0}) = ({V_s}^*, {V_d}^*)$. At the linearization-point equation (5a) reduces to (7) and (5b) reduces to (8).

$$0 = L_1 V_{d,0} \Rightarrow V_{d,0} = 0 \tag{7}$$

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

A change of variables shown in (9) is used to clearify the linearization process.

$$\begin{bmatrix}\tilde{p}\\ \tilde{e}\\ \tilde{\lambda}\end{bmatrix} = \begin{bmatrix}p\\ e\\ \lambda\end{bmatrix} - \begin{bmatrix}p^*\\ e^*\\ \lambda^*\end{bmatrix} \; and \; \begin{bmatrix}\tilde{V_s}\\ \tilde{V_d}\end{bmatrix} = \begin{bmatrix}V_s\\ V_d\end{bmatrix} - \begin{bmatrix}V_s^*\\ V_d^*\end{bmatrix} \tag{9}$$

If we reformulate the equations in (5) and insert our values for $V_{s,0}$ and $V_{d,0}$ we get the system in equation 10.

$$\ddot{\tilde{p}} = \frac{L_1\tilde{V_d}}{J_p} \tag{10a}$$
$$\ddot{\tilde{e}} = \frac{L_2\cos(\tilde{e}) + L_3(\tilde{V_s} - \frac{L_2}{L_3})\cos(\tilde{p})}{J_e} \tag{10b}$$
$$\ddot{\tilde{\lambda}} = \frac{L_4(\tilde{V_s} - \frac{L_2}{L_3})\cos(\tilde{e})\sin(\tilde{p})}{J_\lambda} \tag{10c}$$

Linearization around the point $(\tilde{p}, \tilde{e}, \tilde{\lambda})^T = \vec{0}$ and $(\tilde{V_s}, \tilde{V_d})^T = \vec{0}$ gives us the linearized system in equation 11.

$$\begin{bmatrix}\dot{\tilde{p}}\\ \ddot{\tilde{p}}\\ \dot{\tilde{e}}\\ \ddot{\tilde{e}}\\ \dot{\tilde{\lambda}}\\ \ddot{\tilde{\lambda}}\end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ -\frac{L_4L_2}{J_\lambda L_3} & 0 & 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\tilde{p}\\ \dot{\tilde{p}}\\ \tilde{e}\\ \dot{\tilde{e}}\\ \tilde{\lambda}\\ \dot{\tilde{\lambda}}\end{bmatrix} + \begin{bmatrix} 0 & 0\\ 0 & \frac{L_1}{J_p}\\ 0 & 0\\ \frac{L_3}{J_e} & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\tilde{V_s}\\ \tilde{V_d}\end{bmatrix} \tag{11}$$

We can rewrite equation (11), as in equation 12.

$$\ddot{\tilde{p}} = \frac{L_1}{J_p}\tilde{V_d} \tag{12a}$$

$$\ddot{\tilde{e}} = \frac{L_3}{J_e}\tilde{V_s} \tag{12b}$$

$$\ddot{\tilde{\lambda}} = -\frac{L_4 L_2}{L_3 J_\lambda}\tilde{p} = \frac{L_2}{J_\lambda}\tilde{p} \tag{12c}$$

From equation (9) we know that $\tilde{p} = p$, $\tilde{e} = e$, $\tilde{\lambda} = \lambda$ and $\tilde{V_d} = V_d$. We therefore return to use the original variable names in the linearized system equations (13) and for the rest of the report. The coefficiants of the linearized system are renamed and the physical constant are substituded in to get equation (14).

$$\ddot{p} = K_1 V_d \tag{13a}$$

$$\ddot{e} = K_2 \tilde{V_s} \tag{13b}$$

$$\ddot{\lambda} = K_3 p \tag{13c}$$

$$\begin{aligned} K_1 &= \frac{K_f}{2m_p l_p} \\ K_2 &= \frac{K_f l_h}{m_c l_c^2 + 2m_p l_h^2} \\ K_3 &= \frac{g(m_c l_c - 2m_p l_h)}{m_c l_c^2 + 2m_p(l_h^2 + l_p^2)} \end{aligned} \tag{14}$$

The motor force constant K_f (15), can be found by doing a moment balance around the elevation joint.

$$K_f = \frac{g(2l_h m_p - l_c m_c)}{V_{s,0} l_h} \tag{15}$$

3 Lab day 1

3.1 Preparations

A PD controller (16a), is chosen to control the pitch of the system. This is combined with the system equations (13) to make a transfer function (16b) from the reference p_c to the pitch angle p. As we intend to use pole placement for tuning, the controller coefficients (17) can be found from the transfer function.

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p} \tag{16a}$$

$$\frac{\tilde{p}(s)}{\tilde{p_c}(s)} = \frac{K_1 K_{pp}}{s^2 + sK_1 K_{pd} + K_1 K_{pp}} \tag{16b}$$

$$K_{pd} = \frac{-(\lambda_1 + \lambda_2)}{K_1} \tag{17a}$$

$$K_{pp} = \frac{\lambda_1 \lambda_2}{K_1} \tag{17b}$$

3.2 Manual control

Our first attempt at controlling the helicopter was done using feedforward control with the joystick. The Y-axis on the joystick was connected to the voltage sum, V_s and the X-axis controlled the voltage difference, V_d. We experienced that this was difficult to do. The following issues where observed:

- It was difficult to keep the helicopter pitch steady and stable.
- The helicopter tends to loose elevation while pitching and this is difficult to compensate for.
- The helicopter was easier to control when close to the ground than with high elevation.
- Having a high angular speed around the λ-axis helps to stabilize the elevation.

3.3 Parameter identification

Before attempting to control the helicopter we verified that the encoders gave feedback in the direction stated by figure 1. Then we added an offset to the elevation encoder so that the feedback where zero when the helicopter arm was parallel to the floor. In order to implement the linearization from the preparations we needed to determine the motor constant K_f. This was done experimentally by setting a constant value to V_s and keeping the pitch at zero by hand. The value of V_s was increased gradually until the helicopter arm was parallel to the floor (elevation = 0). At this point the constant value applied to V_s equaled to $V_{s,0}$. Then the motor constant K_f was calculated as shown in the preparations (15).

3.4 PD control

We were now ready to control the system using separate elevation and pitch controllers. The elevation controller was given, but the pitch controller had to be implemented by us as in figure 2.

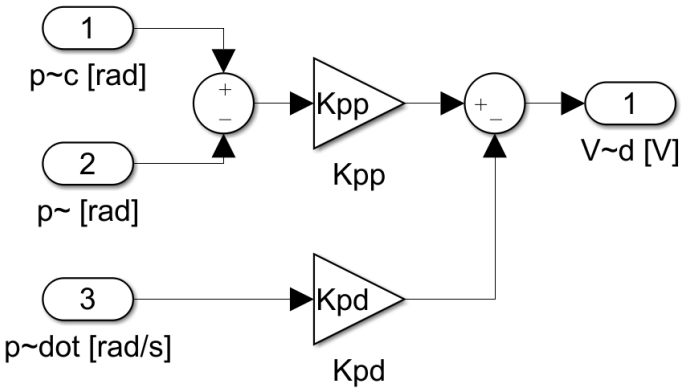


Figure 2: The PD-pitch-controller implemented in Simulink

3.4.1 Tuning criteria

The following properties was prioritized for tuning the system:

- Stability
- No/low chance of crashing
- Ideally a somewhat quick response

In addition, the following properties was desired to make the helicopter controllable from the joystick.

- A response quick enough that the need of over-compensating is unnecessary
- No major oscillations (this would make the helicopter uncomfortable to control)

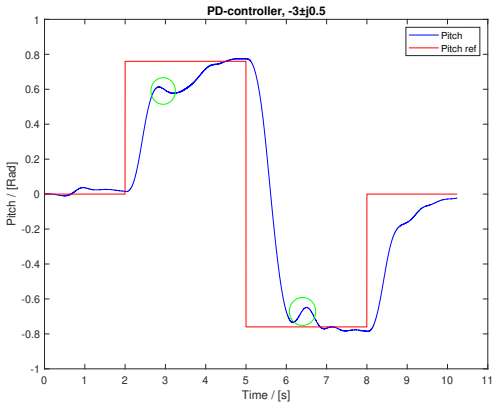
This resulted in a fairly non-aggressive approach to tuning.

3.4.2 System limitations and tuning procedure

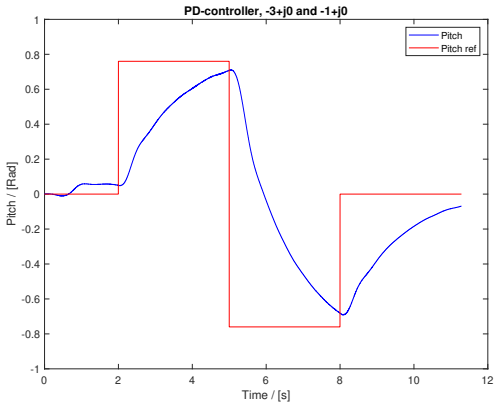
Before tuning the controller we tested the system for its limitations. We wanted to see which pitch angle was the maximum where the helicopter still had the power to maintain elevation. Through testing we concluded that this was about 45-50 degrees. To be safe we decided on 45 degrees ($\pi/4$). Pitch angles that exceeds this is our opinion not applicable for control via joystick and therefore the system was tuned to handle this sequence of inputs:

1. Initialize to zero elevation and zero pitch
2. Step to 45 degrees in positive direction
3. Step to 45 degrees in negative direction for a total of 90 degrees - the "Worst case scenario"
4. Step back to zero pitch.

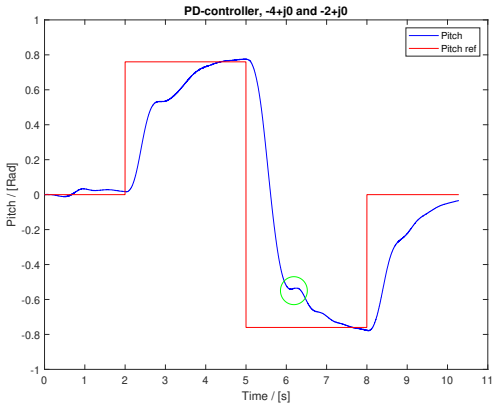
Choosing complex poles gave a fairly quick response, but the oscillatory behaviour meant that the helicopter became uncomfortable to control with a joystick because the system would sometimes pitch opposite of the joystick in order to correct. We also experienced that a response this quick would seemingly make the elevation controller struggle to maintain elevation in some sequences. This could cause a crash and we abandoned complex poles.



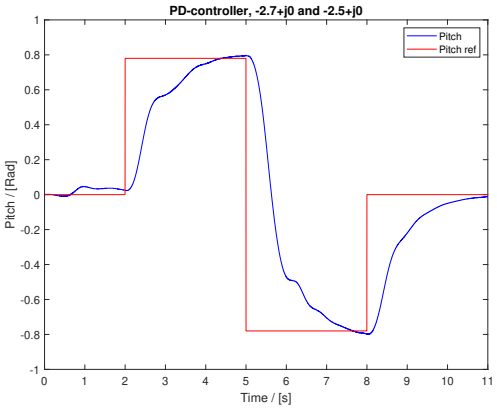
(a) Response with complex poles



(b) Response with real poles



(c) Response with real poles



(d) Final response

Figure 3: Different responses, green ellipse highlights unwanted behaviour.

We continued with real-value poles. Figure 3b is included to show why choosing poles which are too close to zero is a bad move. The response is way too slow. 3c tells us that poles too far apart gives oscillatory tendencies, which theoretically should not happen. However the graph does not show that the elevation controller is struggling and probably interfering with the pitch controller (saturation in the total available propeller-power). In the end we settled upon a marginally over-damped system with poles at -2.7+j0 and -2.5+j0. The response still not perfect, but moving the poles further towards zero only meant a slower response without being sufficiently "cleaner". We considered the response to be clean enough as long as the pitch stayed in the desired direction and not turning the opposite way.

4 Lab day 2

4.1 Preparations

4.1.1 State space model

We want the system on a state-space form as in equation 18.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{18}$$

Where $\mathbf{A}$ and $\mathbf{B}$ are matrices and the state-vector $\mathbf{x}$ and input-vector $\mathbf{u}$ are as stated in equation 19. This gives us the system in equation 20.

$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} \text{ and } \mathbf{u} = \begin{bmatrix} \tilde{V_s} \\ V_d \end{bmatrix} \tag{19}$$

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \ddot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V_s} \\ V_d \end{bmatrix} \tag{20}$$

4.1.2 Controllability

We want to examine the controllability of the system. We calculate the controllability-matrix, $\mathcal{C}$, of the system. The controllability-matrix of a system is defined in [1] as:

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} & \dots & \mathbf{A^{n-1}B} \end{bmatrix}$$

Inserted values for $\mathbf{A}$ and $\mathbf{B}$ we get the controllability-matrix in 21.

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & 0 & K_1 & 0 & 0 \\ 0 & K_1 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{21}$$

We observe that our controllability-matrix, $\mathcal{C}$, in equation 21 has full row-rank. Thus the system is controllable.

4.1.3 3-state controller

We want to track the reference $\mathbf{r} = \begin{bmatrix} p_c & \dot{e}_c \end{bmatrix}^T$ for the pitch-angle p and the elevation rate $\dot{e}$. We consider a state-feedback controller with reference-feed-forward (22) with matrices as defined in equation 23 and 24.

$$\mathbf{u} = \mathbf{F}\mathbf{r} - \mathbf{K}\mathbf{x} \tag{22}$$

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \end{bmatrix} \text{ and } \mathbf{F} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \tag{23}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{24}$$

By assuming stability, $\dot{\mathbf{x}}(\mathrm{t} = \infty) = \mathbf{0}$ and defining $\mathbf{x}_\infty \triangleq \mathbf{x}(\infty)$ we can combine equation 18 and 22 in 25.

$$\begin{aligned}
\dot{\mathbf{x}} = 0 &= \mathbf{A}\mathbf{x}_\infty + \mathbf{B}(\mathbf{F}\mathbf{r} - \mathbf{K}\mathbf{x}_\infty) \\
\Rightarrow \mathbf{x}_\infty &= (\mathbf{B}\mathbf{K} - \mathbf{A})^{-1}\mathbf{B}\mathbf{F}\mathbf{r} \\
\Rightarrow \mathbf{y}_\infty &= \mathbf{C}(\mathbf{B}\mathbf{K} - \mathbf{A})^{-1}\mathbf{B}\mathbf{F}\mathbf{r}
\end{aligned} \tag{25}$$

We want $\mathbf{y}_\infty = \mathbf{r}$, therefore we must choose $\mathbf{F} = (\mathbf{C}(\mathbf{B}\mathbf{K} - \mathbf{A})^{-1}\mathbf{B})^{-1}$).

4.1.4 Augmented system and controller

In order for the controller to have integral effect, we want to create an augmented system based on (18) with two additional states representing the integral of our control variables (equation 26a). An augmented state vector (26b), a modified controller (26c) based on (22). The resulting system is shown in (27).

$$\dot{\mathbf{x}}_\mathbf{a} = \begin{bmatrix} \dot{\gamma} \\ \dot{\zeta} \end{bmatrix} = \begin{bmatrix} p \\ \dot{e} \end{bmatrix} - \begin{bmatrix} p_c \\ \dot{e}_c \end{bmatrix} = \mathbf{C}\mathbf{x} - \mathbf{r} \tag{26a}$$

$$\bar{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ \mathbf{x_a} \end{bmatrix}, \text{ where } \mathbf{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} \text{ and } \mathbf{x_a} = \begin{bmatrix} \gamma \\ \zeta \end{bmatrix} \tag{26b}$$

$$\bar{\mathbf{u}} = \bar{\mathbf{F}}\mathbf{r} - \bar{\mathbf{K}}\bar{\mathbf{x}}, \text{ where } \bar{\mathbf{K}} = \begin{bmatrix} \mathbf{K} & \mathbf{K_a} \end{bmatrix} \tag{26c}$$

$$\dot{\bar{\mathbf{x}}} = \bar{\mathbf{A}}\bar{\mathbf{x}} + \bar{\mathbf{B}}\bar{\mathbf{u}} + \mathbf{G}\mathbf{r} \tag{27a}$$

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_\mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{3x3} & \mathbf{0}_{3x2} \\ \mathbf{C}_{2x3} & \mathbf{0}_{2x2} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x_a} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{3x2} \\ \mathbf{0}_{2x2} \end{bmatrix} (\bar{\mathbf{F}}\mathbf{r} - \begin{bmatrix} \mathbf{K} & \mathbf{K_a} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x_a} \end{bmatrix}) + \begin{bmatrix} \mathbf{0}_{3x2} \\ -\mathbf{I}_{2x2} \end{bmatrix} \mathbf{r} \tag{27b}$$

$\bar{\mathbf{K}}$ for our new augmented system can be found by using the Matlab-command *lqr*() with the new augmented system matrices as input ($\bar{\mathbf{K}} = lqr(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{Q_{5x5}}, \mathbf{R_{2x2}})$). We want a stable response meaning $\lim_{t\to\infty} \begin{bmatrix} \dot{\mathbf{x}} & \dot{\mathbf{x}}_\mathbf{a} \end{bmatrix}^T = 0$. This in combination with the bottom equation in (27b) gives us $0 = \mathbf{C}\mathbf{x} - \mathbf{r} \Rightarrow \mathbf{y} = \mathbf{r}$. By adding integral action, we have thus made the output $\mathbf{y} = \mathbf{r}$ as t $\to \infty$. We no longer need to design $\bar{\mathbf{F}}$ so that the output $\mathbf{y}$ goes towards the reference $\mathbf{r}$ when time goes towards infinity, but it can be chosen arbitrarily within reason. From the top equation in (27b) it can be shown that the optimal equation for $\bar{\mathbf{F}}$ is independent of whether we add integral action and it can be chosen in the same way as we did in the system without integral action (25).

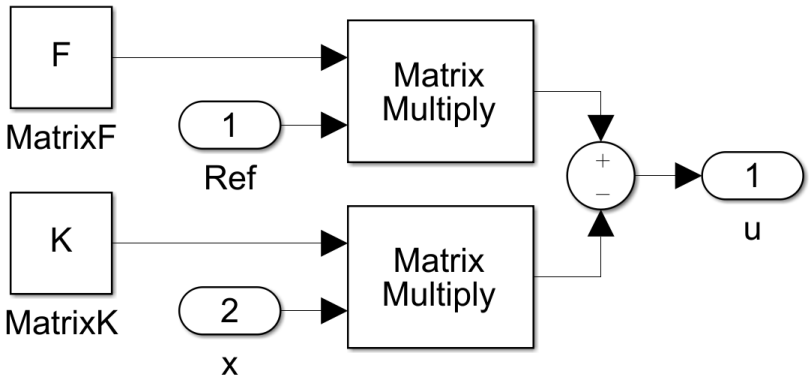


Figure 4: 3-state LQR-controller

4.2 3-state feedback controller

4.2.1 Controller implementation

The controller was implemented in simulink according to equation (22), as shown in figure 4.

4.2.2 Testing procedure

The system was tuned experimentally by educated guesses at pole placement. The system outputs was tested individually for simplicity. Pitch was tested with the same square wave as used in day 1. Elevation rate is not as easy to test by a similar procedure. Therefore we just tested the elevation rate with the joystick and settled on what "felt right"; a bit less scientific of an approach. The following criteria was used:

- Stability
- Quick response in elevation
- No crashing
- No major oscillations in pitch

4.2.3 Pole placement

```
1       K = place(A,B,p);
```

Figure 5: Matlab code to find the K-matrix with pole placement.

Pole placement was used to tune the system, this was easily implemented by including the system matrices in a MATLAB-script. Then using the command in figure 5 to place the poles at the vector p, which contains the desired poles and find the right **K**-matrix. In order for the system to allow a sufficient elevation rate, we observed that one of the poles had to be chosen on the real axis close to 0. The two other poles could be chosen complex and changing those does not affect the elevation rate significantly. Our final tuning is shown in figure 6.

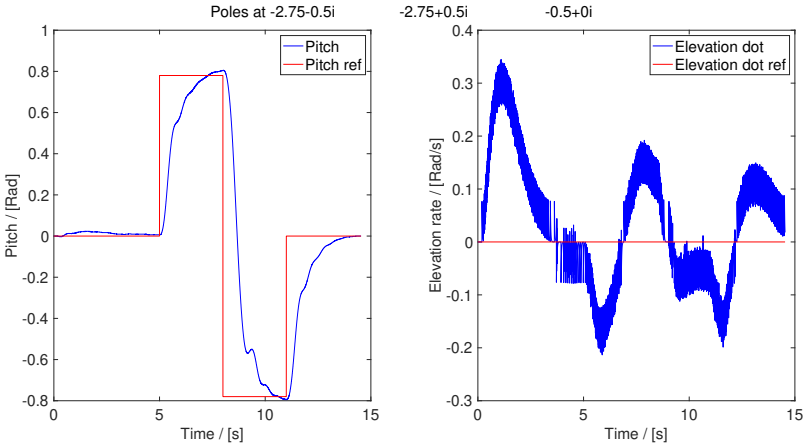


Figure 6: Final response, pole placement

4.2.4 LQR

```
K = lqr(A,B,Q,R);
```

Figure 7: Matlab code to find the K-matrix with LQR.

As an alternative approach to pole placement we tried choosing the K matrix with LQR using the MATLAB command in figure 7. The **Q** (28a) and **R** (28b) matrices were chosen diagonal and according to Bryson's rule.

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{(p_{max})^2} & 0 & 0 \\ 0 & \frac{1}{(\dot{p}_{max})^2} & 0 \\ 0 & 0 & \frac{1}{(\dot{e}_{max})^2} \end{bmatrix} \tag{28a}$$

$$\mathbf{R} = \begin{bmatrix} \frac{1}{(\tilde{V}_{s,max})^2} & 0 \\ 0 & \frac{1}{(V_{d,max})^2} \end{bmatrix} \tag{28b}$$

In this case this tuning approach is useful, since we have knowledge of some of the elements in the matrices:

- $\tilde{V}_{s,max} = 24$ (Two motors @ 12 v)
- $V_{d,max} = 12$ (One motor @ 12 v)
- $p_{max} = \frac{\pi}{4}$ (Our experienced pitch limit from lab day 1)

This reduces the problem to only choosing how much pitch rate and elevation rate we want to allow for. We experienced that increasing the $\dot{e}_{max}$ parameter actually did increase the allowed elevation rate, therefore this had to be chosen reasonably large so that changing the elevation of the helicopter would not feel "laggy" when controlling with the joystick. The $\tilde{V}_{s,max}$ parameter clearly had an impact on the pitch rate, in general choosing this too high gave oscillations in the pitch and choosing it too low meant a slow response. Figure 8 shows our final tuning.

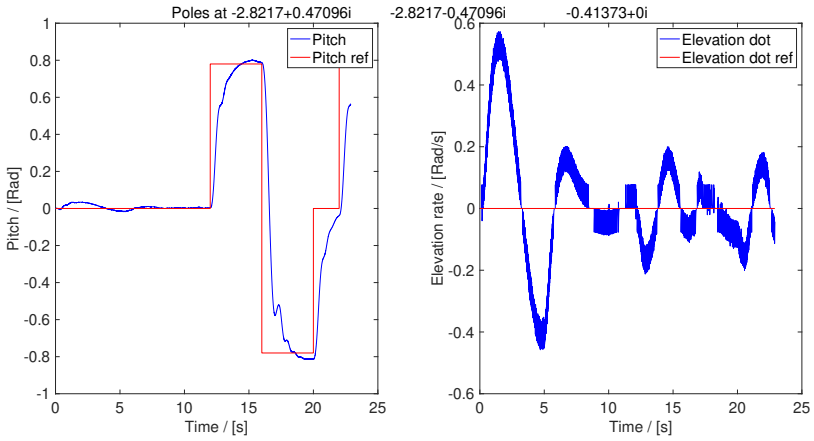


Figure 8: Final response, 3-state LQR

4.2.5 The relation between LQR parameters and poles

The LQR tuning resulted in a set of 3 poles: 1 set of complex poles and a single real pole. A set of 3 real poles is also possible, but did not meet our tuning criteria. We experimented with how a change of values in the Q matrix affected the poles. We found the following relations:

- Larger $\dot{e}_{max}$ -> Real pole moving closer to 0.
- Larger $\dot{p}_{max}$ -> Complex poles moving away from the real axis.
  (imaginary part becoming more significant)

4.3 5-state feedback controller

4.3.1 Controller implementation

The PI-controller in figure 9 was implemented according to equations (26a) and (26c) in the preparations.

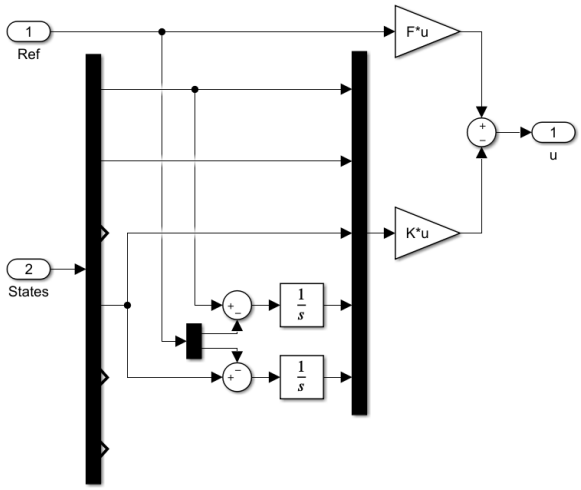


Figure 9: PI-controller, $States = [p, \dot{p}, e, \dot{e}, \lambda, \dot{\lambda}]^T$.

4.3.2 Pole placement

As an experiment we tried using pole placement with MATLAB and $K = place(\bar{A},\bar{B},p)$ as an alternative to the LQR-tuning method. We tried various poles, where we kept one pole on the real axis close to 0 as we did with the 3-state system. For the other four poles we tried with complex-conjugated poles and poles on the real axis. In our experience it is hard to tune a MIMO-system with this many poles by the method of pole-placement. There seems to be no connection between single poles and single states in the 5-state system as we saw tendencies to in the 3-state system. We therefore went back to tuning the system with the help of LQR.

4.3.3 Approach to LQR-tuning

LQR-tuning of the 5-state system was done by changing the matrices **Q** and **R**. Since the controller has integral effect on the states we want to control, we also have the option to change the matrix **F**. Our tuning criteria is shown underneath:

1. To have a stable/working controller for lab day 3 and 4
2. Stability in pitch.
3. The helicopter should be comfortable to control from a joystick.
4. Ability to follow pitch reference.
5. The helicopter should be comfortable to control from a joystick.
6. Stability in elevation rate.
7. Allow for a high pitch rate.
8. Allow for a high elevation rate.
9. Ability to follow elevation rate reference.

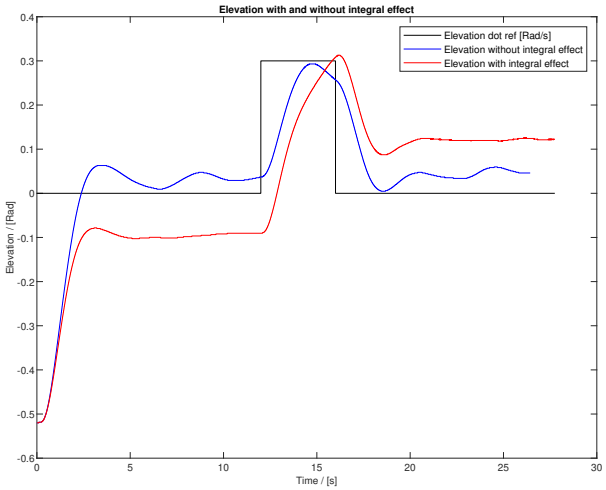


Figure 10: Ability to maintain elevation different from zero.

Ideally, all of these criteria would have been fulfilled. This, however turned out to be extremely difficult and some unfortunate compromises had to be made. The list in numbered in the order we ended up prioritizing. A few of them need justifying; Stability in elevation rate is not prioritized as high as stability in pitch. This means that there exists certain situations (at high elevation angles) where the elevation rate can end up oscillating. We found this acceptable since it did not crash the helicopter and it was possible to regain control by lowering the elevation angle. Deprioritizing the ability to follow elevation reference can seem a bit strange, as the purpose of implementing integral effect usually is to obtain this ability. However, tuning the system more aggressively for it to reach the reference proved to be difficult, since we always had to sacrifice either stability, or make the helicopter slow and/or difficult to control from a joystick. It should be mentioned that the integral effect helps the helicopter maintain altitude instead of sinking back to the equilibrium, see figure 10. Note that the controller without integral effect does not stabilize exactly at zero due to choice of wrong $V_{s,0}$.

4.3.4 Attempt at tuning using Bryson's rule

The K matrix where calculated using LQR, this time with the new system matrices $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$. The $\mathbf{Q}$ and $\mathbf{R}$ matrices where again chosen diagonal and according to Bryson's rule to start with. From there we extracted the poles from the LQR-function and plotted a pulse-response for the pitch. We tried to understand which poles corresponded to which parameter in the diagonal matrices $\mathbf{Q}$ and $\mathbf{R}$ but understood quickly that there not necessarily is any connection between these parameters and the poles as we saw in the three-state system. Therefore the diagonal in $\mathbf{Q}$ and $\mathbf{R}$ where inserted with 1's and we changed values on the diagonal to see the response of the system. After a lot of experimenting we ended up with the matrices described in the following sections.

4.3.5 Choice of the matrix R

$$\mathbf{R} = \begin{bmatrix} 2(\tilde{V_s}) & 0 \\ 0 & 1(V_d) \end{bmatrix} \tag{29}$$

The $\mathbf{R}$ matrix (29) was chosen with values way smaller than the Q matrix. The reason for this is that we want to allow for large motor speeds and to allow for our system to be quick. The first element, corresponding to $\tilde{V_s}$ is chosen to be twice the size of the last element, corresponding to V_d. This somewhat prioritizes V_d according to our tuning criteria in section 4.3.3.

4.3.6 Choice of the matrix Q

$$\mathbf{Q} = \begin{bmatrix} 75(p) & 0 & 0 & 0 & 0 \\ 0 & 30(\dot{p}) & 0 & 0 & 0 \\ 0 & 0 & 1(\dot{e}) & 0 & 0 \\ 0 & 0 & 0 & 100(\gamma) & 0 \\ 0 & 0 & 0 & 0 & 5(\zeta) \end{bmatrix} \tag{30}$$

The main reason we settled for this $\mathbf{Q}$-matrix, is its ability to keep the pitch stable for high elevation angles. Secondarily we set the fourth element on the diagonal (corresponding to γ) large in order to "punish" the pitch-integral from growing to large and causing stability issues. A potential solution to this is proposed in section 7.1. The third element (corresponding to $\dot{e}$) was chosen very low to allow for large elevation rates, this also helps keeping the helicopter easy to control with a joystick.

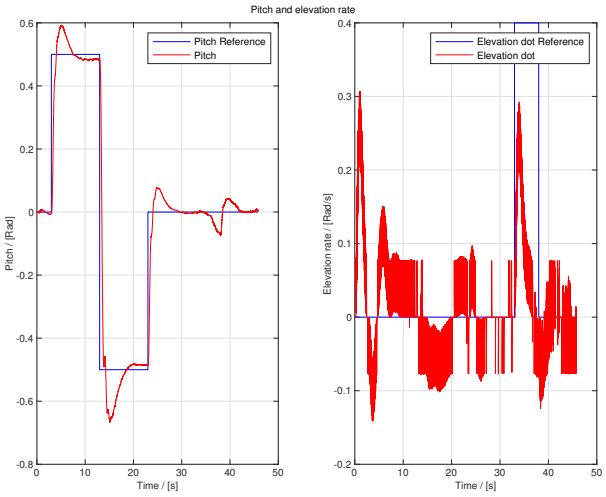


Figure 11: Overshoot in pitch

4.3.7 Changing the feed-forward matrix F

In section 4.1.4 we concluded that the matrix **F** can be chosen arbitrarily when integral effect is added. The ideal matrix should still be the same as in the 3-state controller in theory, since it will let the integrals be close to zero at stationary conditions. With the **K** matrix chosen from **R** and **Q** (29 and 30), **F** should be chosen as in (31).

$$\mathbf{F} = \begin{bmatrix} 0 & 6 \\ 16 & 0 \end{bmatrix} \tag{31}$$

Unfortunately, our choice for the Q and R matrices led to an unacceptable overshoot in pitch when applying a large pitch angle step (see figure 11). Since we could not find a tuning that both was sufficiently stable and did not have the overshoot in pitch, we decided to try something unorthodox. We discovered that reducing the feed-forward gain in the **F**-matrix could remove the overshoot. This resulted in the adjusted **F**-matrix in (32) and the response shown in figure 12. Why this is unorthodox is discussed in section 4.3.8.

$$\mathbf{F} = \begin{bmatrix} 0(p_c \Rightarrow \tilde{V}_s) & 6(\dot{e}_c \Rightarrow \tilde{V}_s) \\ 8(p_c \Rightarrow V_d) & 0(\dot{e}_c \Rightarrow V_d) \end{bmatrix} \tag{32}$$

A matrix on the simplest form (33), was also tested, but this lead to a system that was unacceptably slow. The system was still fully functional though, and it seems like the **F**-matrix can be chosen arbitrarily within reason[1]. This is due to the integral effect of the controller.

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{33}$$

The final response of the LQR-tuned system with adjusted feed-forward matrix **F** is shown in figure 12. Note that the helicopter does not like to maintain the pitch reference as the travel rate increases. Ideally we wanted to compensate for this with more integral effect, but that led to stability issues. When kept at low travel rates the integral effect helps the pitch reach the reference. A solution to this is proposed in section 7.1.

[1]Extremely high- or negative gain is not within reason, neither is feed-forward affecting a non related state.

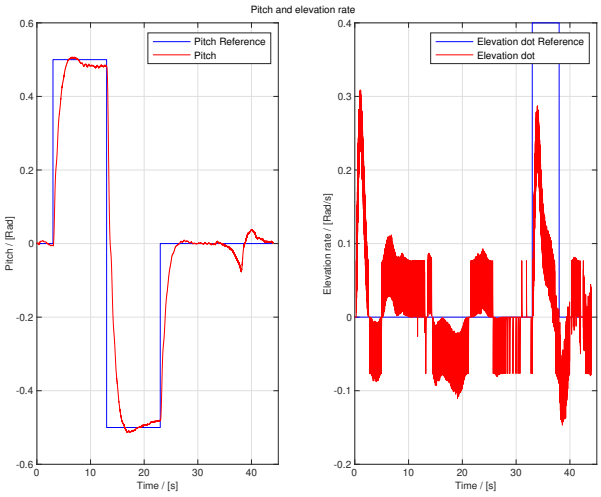


Figure 12: Response of the final tuning of the five-state LQR-system

4.3.8 Consequences of changing the F-matrix

We shall now take a close look at the controller to see why changing the **F**-matrix is an odd thing to do. When choosing the **K** and **F** matrices with LQR, we always end up with the two matrices in (34).

$$K = \begin{bmatrix} 0 & 0 & K_{13} & 0 & K_{15} \\ K_{21} & K_{22} & 0 & K_{24} & 0 \end{bmatrix} \tag{34a}$$

$$F = \begin{bmatrix} 0 & F_{12} \\ F_{21} & 0 \end{bmatrix} \tag{34b}$$

$$\begin{bmatrix} \tilde{V_s} \\ V_d \end{bmatrix} = \begin{bmatrix} 0 & F_{12} \\ F_{21} & 0 \end{bmatrix} \begin{bmatrix} p_c \\ \dot{e_c} \end{bmatrix} - \begin{bmatrix} 0 & 0 & K_{13} & 0 & K_{15} \\ K_{21} & K_{22} & 0 & K_{24} & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \\ \int_0^t (p - p_c)\, d\tau \\ \int_0^t (\dot{e} - \dot{e_c})\, d\tau \end{bmatrix} \tag{35}$$

$$\tilde{V_s} = F_{12}\dot{e_c} - K_{13}\dot{e} - K_{15} \int_0^t (\dot{e} - \dot{e_c})\, d\tau \tag{36a}$$

$$V_d = F_{21}p_c - K_{21}p - K_{22}\dot{p} - K_{24} \int_0^t (p - p_c)\, d\tau \tag{36b}$$

As long as $F_{12} = K_{13}$ and $F_{21} = K_{21}$ (The ideal F-matrix from (25)), this simplifies nicely into (37):

$$\tilde{V_s} = K_{13}(\dot{e_c} - \dot{e}) + K_{15} \int_0^t (\dot{e_c} - \dot{e})\, d\tau \tag{37a}$$

$$V_d = K_{21}(p_c - p) - K_{22}\dot{p} + K_{24} \int_0^t (p_c - p)\, d\tau \tag{37b}$$

Which are respectively a classic PI-controller and a PID-controller. But once we decide to let $F_{21} = \frac{1}{2}K_{21}$ as we did in (32), equations (38) gets an extra term marked in red:

$$\tilde{V}_s = K_{13}(\dot{e_c} - \dot{e}) + K_{15}\int_0^t (\dot{e_c} - \dot{e})\, d\tau \tag{38a}$$

$$V_d = \frac{1}{2}K_{21}(p_c - p) - \frac{1}{2}K_{21}p - K_{22}\dot{p} + K_{24}\int_0^t (p_c - p)\, d\tau \tag{38b}$$

This term will help to reduce the overshoot, which we proved it did in figure 12. The term will in fact reduce the controller output at high pitch angles. The problem with this is that it will also try to force the pitch back to zero, and the integral will have to compensate for this. This controller is fairly unorthodox in control theory, but worked for us. Most importantly it provided a stable controller that we could use for lab day 3 and 4.

4.3.9 Approach if we were to do this again

If we knew what we know now when approaching the LQR tuning, we would do a couple of things differently. Our tuning approach would be something like:

1. Tune the pitch of the system by balancing p, $\dot{p}$ and γ in the Q-matrix to remove the overshoot.
2. Tune the elevation of the system by balancing $\dot{e}$ and ζ in the Q-matrix.
3. Tune the ratio between pitch and elevation in the Q-matrix.
4. Tune the speed of the system by adjusting the R-matrix.

This approach is a bit more organized and reduces the problem into manageable sub-problems.

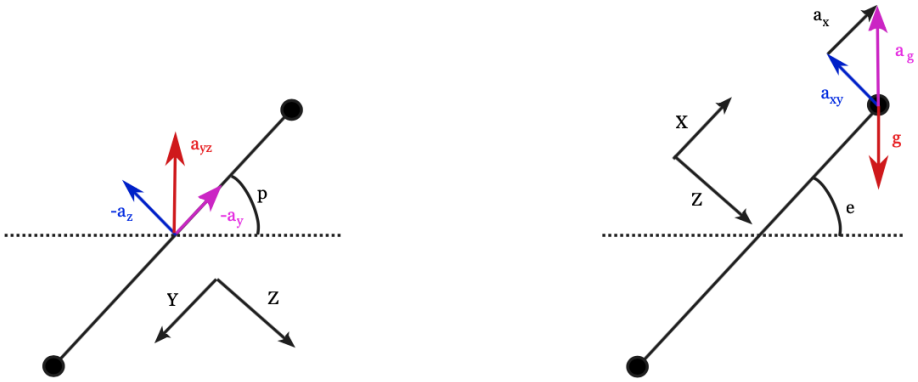


Figure 13: Free body diagram

5 Lab day 3

5.1 Preparations

We want the system on the state-space formulation in equation 50.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \tag{39}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are matrices and the state and input vectors are given in equation 40.

$$\mathbf{x} = \begin{bmatrix} p & \dot{p} & e & \dot{e} & \dot{\lambda} \end{bmatrix}^T \text{ and } \mathbf{u} = \begin{bmatrix} \tilde{V}_s & V_d \end{bmatrix}^T \tag{40}$$

This gives us the system in equation 41. The $\mathbf{C}$-matrix depends on which states are measured.

$$\begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \\ \ddot{\lambda} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ K_3 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_3 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_s \\ V_d \end{bmatrix} \tag{41}$$

From figure 13 we get the equations stated in 42 and from the measured acceleration we can derive the equations for pitch and elevation as stated in 43.

$$\begin{aligned} a_z &= -a_g cos(e) cos(p) \\ a_y &= -a_g cos(e) sin(p) \\ a_x &= a_g sin(e) \end{aligned} \tag{42}$$

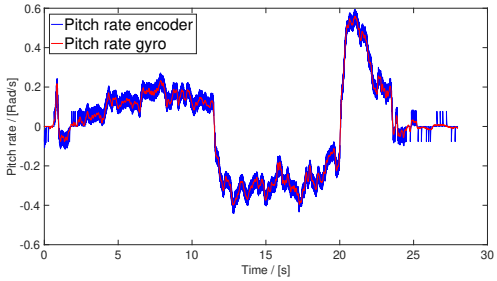
$$p = \arctan\left(\frac{a_y}{a_z}\right) \tag{43a}$$

$$e = \arctan\left(\frac{a_x}{\sqrt{{a_y}^2 + {a_z}^2}}\right) \tag{43b}$$

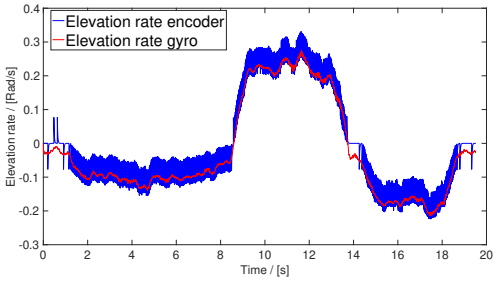
A linear observer given in equation 44 should be used instead of the encoder-values to estimate the states $\mathbf{x}$. In this case we need to calculate the observability-matrix $\mathcal{O}$ [1], given in equation 45, in order to see if the system is observable. By checking if the observability-matrix is full rank we can determine whether the system is observable.

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \tag{44}$$

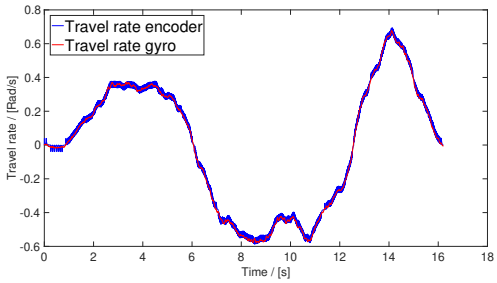
$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix} \tag{45}$$



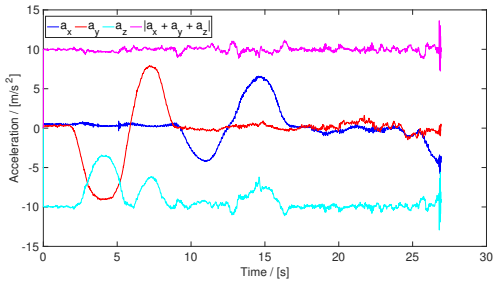
(a) Pitch rate comparison with movement only around the pitch axis.



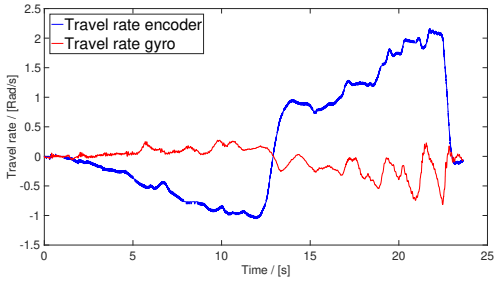
(b) Elevation rate comparison with movement only around the elevation axis.



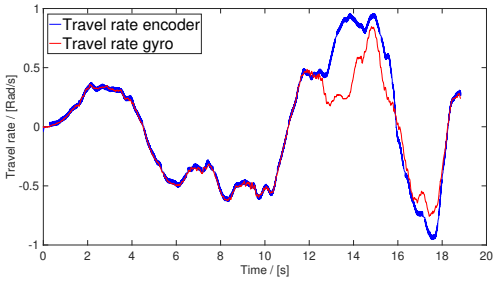
(c) Travel rate comparison with movement only around the travel axis.



(d) Accelerometer output with movement around one axos at the time: p∈[2,9], e∈[9,17], λ ∈[17,25]



(e) Travel rate comparison with movement around a second axis.



(f) Two axis rotation after 12 seconds.

Figure 14: Raw measurements

5.2 IMU feedback

5.2.1 Gyro

To understand the output from the IMU, we experimented with moving the helicopter by hand. While turning only one axis at the time, the gyro output followed the encoder nicely (figure 14a, 14b and 14c). Then a rotation around a second axis was introduced. This resulted in gyro readings that were not even close to the encoder (figure 14e and 14f). This is because the gyro has a coordinate system relative to the helicopter. It is different to the one we defined when modeling our system which is relative to the table. As an example we can imagine that the helicopter is at $e = \lambda = 0$ and $p = \frac{\pi}{2}$. If this is the case, the gyro will see travel rate as elevation rate and vice versa. A rotation matrix (46a) is necessary to correct for this. The rotation matrix depends on values for pitch and elevation and could potentially introduce additional noise and error to the rates. We have to make sure our state estimate/feedback is fairly accurate and not too noisy.

$$\mathbf{T} = \begin{bmatrix} 1 & sin(p)tan(e) & cos(p)tan(e) \\ 0 & cos(p) & -sin(p) \\ 0 & sin(p)/cos(e) & cos(p)/cos(e) \end{bmatrix} \tag{46a}$$

$$\begin{bmatrix} \dot{p} \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} = \mathbf{T} \begin{bmatrix} \dot{p}_{gyro} \\ \dot{e}_{gyro} \\ \dot{\lambda}_{gyro} \end{bmatrix} \tag{46b}$$

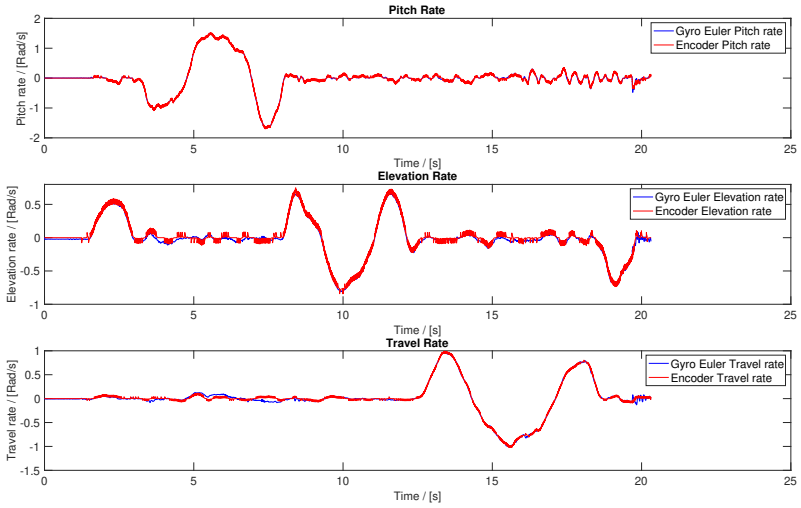


Figure 15: Corrected gyro euler rates compared with encoder values

5.2.2 Accelerometer

To calculate the pitch- and elevation-angle that will be used as a measurements for the two states respectively, we will use measurements from the accelerometer. These two angles will also be used to correct the gyro measurements. The accelerometer measures proper acceleration meaning it measures the acceleration in relation to free-fall. The three-state accelerometer vector $\mathbf{a_{xyz}} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}$ from the IMU is plotted in figure 14d. From the figure we see that a change in pitch and elevation will alter the decomposed accelerometer measurement, while a change in travel will keep them close to constant (close due to imperfect turning of the helicopter) as λ does not alter the measured acceleration (42). It can also be noted that the absolute value of the sum of the decomposed accelerometer measurement is close to free-fall acceleration as expected from a proper accelerometer. The decomposed accelerometer measurement is used to indirectly calculate the helicopters orientation. Equation 43 is implemented as a Matlab-function in Simulink as in figure 17.

In figure 16 we see that we have a nearly perfect match of the corrected rates and the encoder rates. When it comes to the corrected values for pitch and elevation (figure 16) we see that the corrected accelerometer values are more noisy than the encoder values. Note that in figure 16 and 15 the helicopter is turned by hand, meaning less introduction of noise caused by the rotors etc. This noise must be taken into account when controlling the helicopter from the joystick with accelerometer feedback.

There is an offset between the encoder elevation angle and the corrected accelerometer angle in figure 16. The encoder value is lower in comparison with the corrected accelerometer value. This may be a result of the offset-value added to the encoder-output to make it zero at the linearization-point as discussed in section 3.3. Additionally, if the table under the helicopter is not perfectly leveled, the same error may occur. The encoder measures the elevation angle in relation to the helicopter-arm attached to the table. An offset-value on the accelerometer output (and the gyro) was found later through testing. It was corrected for by adding / subtracting a constant from the respective output on the accelerometer until the values at the linearization point were equal to the encoder values. See section 7.2. We have thus assumed that the encoder gives a more correct value than the accelerometer.

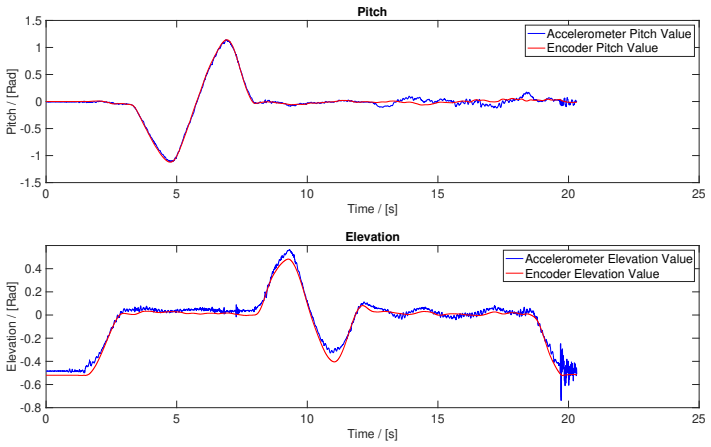
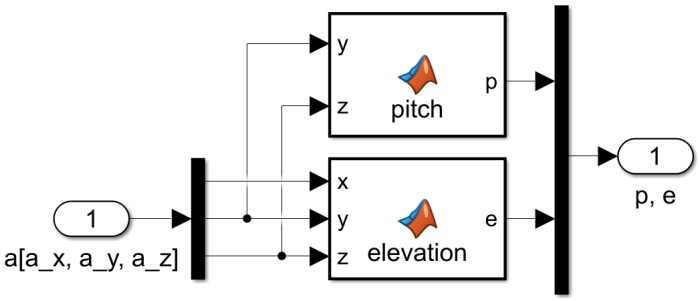


Figure 16: Calculated Pitch- and elevation-angle compared with encoder values



(a) Subsystem pitch and elevation.

```
function p = pitch(y,z)
    if (z == 0)
        p = 0;
    else
        p = atan(y/z);
    end
```

(b) Code content of the pitch function block.

```
function e = elevation(x,y,z)
    if (y == 0 && z == 0)
        e = 0;
    else
        e = atan(x/sqrt(y^2+z^2));
    end
```

(c) Code content of the elevation function block.

Figure 17: Simulink subsystem to determine pitch and elevation from accelerometer outputs.

5.3 Theoretical discussion

5.3.1 The relation between the A-matrix and observability

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ K_3 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{47}$$

For a system to be observable, all states must either be measured directly or be possible to calculate from the states that are measured, $\mathbf{y}$. A state cannot be calculated through integration of another state. This is due to the missing initial condition needed in the integral. Calculation through the derivative is fine though. Through calculating the observability matrix in the preparations we learned that two of the states always had to be measured; e and $\dot{\lambda}$. If we take a look at the A-matrix (47), we see that columns 3 and 5 (corresponding to e and $\dot{\lambda}$) have only zeroes, while the other columns have one element different from zero. A somewhat intuitive explanation to this is:

1. States 3 and 5 cannot be calculated and have to be measured.
2. State 1 can be calculated from state 5.
3. State 2 can be calculated from state 1.
4. State 4 can be calculated from state 3.

With our states:

1. e and $\dot{\lambda}$ cannot be calculated and have to be measured.
2. p can be calculated from $\dot{\lambda}$.

3. $\dot{p}$ can be calculated p.

4. $\dot{e}$ can be calculated from e.

This can be related to the linearized system equations 13, which tells us that p is just a derivative of $\dot{\lambda}$ (see equation 48a). Additionally, $\dot{p}$ and $\dot{e}$ are just derivatives too (equations 48b and 48c). It is clear that the states that do not have to be measured can be calculated from the measured states. If we wanted to go the other way around an integration would be necessary and the initial condition would be missing. This is the reason why $\dot{\lambda}$ and e have to be measured. It should be noted that the equations used to calculated these states are in some cases (equations 48b and 48b) heavily linearized and that must be kept in mind if you want to estimate a state.

$$p = \frac{d}{dt}\frac{\dot{\lambda}}{K3} \tag{48a}$$

$$\dot{p} = \frac{d}{dt}p = \frac{d^2}{dt^2}\frac{\dot{\lambda}}{K3} \tag{48b}$$

$$\dot{e} = \frac{d}{dt}e \tag{48c}$$

5.3.2 Which measurements are necessary

In part 5.3.1 we concluded that measuring two states should be enough. However this does not mean that measuring only two states is a good idea. The reason we do not think so is that the system is non linear and the calculation of states is based on a linearized state space model. Our main concern is the pitch, which is quite heavily linearized. The relation between $\dot{\lambda}$ and p is only correct for $p = 0$, and is far from correct for large pitch angles. In addition to this, the $\dot{p}$ state then has to be calculated by derivation of a calculated state. This seems like a bad idea and we suspect that it could result in poor stability at large pitch angles. This means that if we wanted to reduce our number of states we would go for at least three; e and $\dot{\lambda}$ and either $\dot{p}$ or p. Probably p, since that is a state we want to control, and to avoid estimating p through a linearized relation in equation (48a). The other state which is controlled in our system is $\dot{e}$, estimating this through taking the derivative of e can be done, but might result in amplified noise. None of this really matters since the IMU returns all five states anyways, but would be an interesting discussion and necessary if some of the states were more difficult to measure.

5.3.3 Assumptions regarding the accelerometer

In our derivation of the equations in figure 13, we assumed that the acceleration due to gravity was the only acceleration affecting our system and that the combined acceleration was completely vertical. This is a bold assumption. The whole idea of controlling the helicopter is to accelerate it. Also, fast travel rates creates a centripetal acceleration. Thankfully the force of gravity on earth is fairly significant, so the acceleration of the helicopter does not affect the vertical reference too much. However we experienced a significant error from the centripetal acceleration. Theoretically this could be corrected for by taking in to account the readings from the gyro in a similar way as when correcting the gyro output (46a). Ideally we would make a complete system that corrects for acceleration along all the axis, but that would introduce non linear mutual dependencies and algebraic loops between all the states and make stability analysis / intuition of stability difficult. A fairly simple solution is proposed in section 7.3.

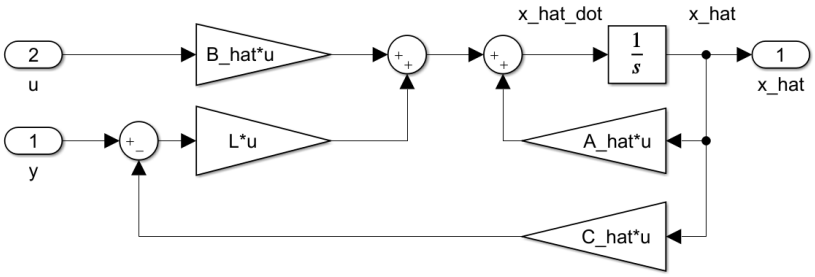


Figure 18: State estimator implemented in simulink.

5.4 State estimator

By defining the error of the estimated state as in equation 49a and by assuming that our measured signal is $\mathbf{y_m} = \mathbf{Cx} + \mathbf{w}$, where $\mathbf{w}$ is the noise of the measurement we get the relationship stated in equation 49b.

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} \tag{49a}$$

$$\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{LC})\mathbf{e} - \mathbf{Lw} \tag{49b}$$

By choosing a matrix $\mathbf{C}$ such that the system is observable it is possible to place the poles of the estimator arbitrary by choosing the estimator-gain matrix $\mathbf{L}$. We should choose the poles of the estimator (system $\mathbf{A} - \mathbf{LC}$) such that the dynamics of the estimator is faster than the system it self. The dynamics of the system is given by the poles of $\mathbf{A} - \mathbf{BK}$. By using the Matlab-function *place*, the poles of the estimator was distributed along an arc in the left half plane, with radius r from the origin and and opening angle of ϕ from the positive imaginary axis. The radius was calculated by finding the largest eigenvalue of the system, and then multiply this value with a factor. By doing it this way we ensure that we get faster dynamics in the observer than in the system. See figure 19 for the code used to do this.

By choosing large values for $\mathbf{L}$ we see from equation 49b that the noise in the measurement will be amplified. By choosing the values too low we will end up with an estimator that is too slow. We need to choose the values such that we get an acceptable estimator-speed, with minimal disturbance from the measurement.

```
1 sys_max = max(abs(eig(A_bar-B_bar*K))); %Fastest system eigenvalue
2 r = 20*sys_max; %Radius from origin
3 phi = pi/8; %Biggest angle from negative real axis
4 %Estimator poles
5 Lpoles = -r.*[exp(-phi*1i) exp(-(phi/2)*1i) 1 exp(+(phi/2)*1i) exp(+phi*1i)];
6 % L-matrix
7 L = (place((A_hat)',(C_hat)',Lpoles))';
```

Figure 19: Matlab code to find the L-matrix with pole placement

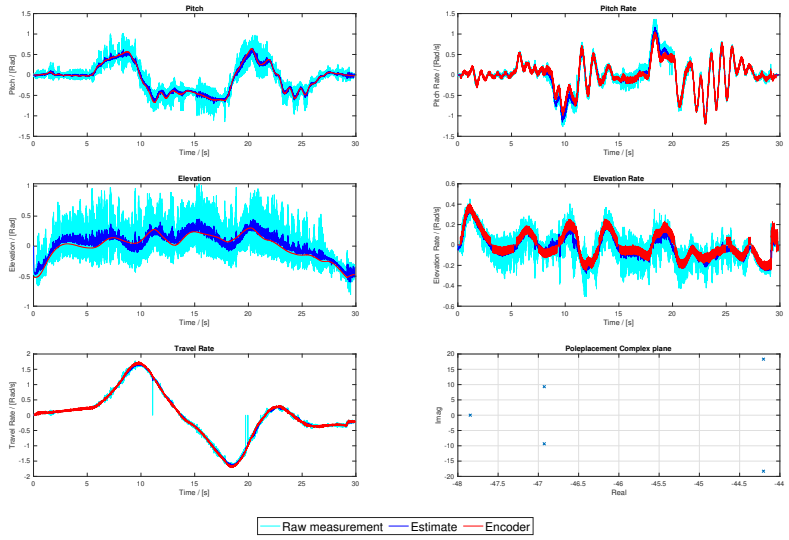


Figure 20: Final tuning of the observer with IMU feedback

The estimator given in equation 44 was implemented as shown in figure 18 with matrices $\mathbf{A}$ and $\mathbf{B}$ as given in equation 45. The LQR with integral action were used as controller. The encoder was used for feedback to the controller and to the "correction-input" were we use the angles e and p to correct for rotations in the gyros coordinate system. The encoder values were used in order to have a stable system it was possible to make a rough initial tuning on. The state estimates were compared to the measured encoder-values and the raw-measurements. By using the encoder for feedback we experienced that we could have a pretty large gain factor on the largest eigenvalue of the system an hence a large radius of the arc. We also observed that we could have a close spread of the poles. We ended up with a gain-factor of 20.

The estimated states were used as feedback to the controller and to the "correction-input" instead of the encoder-measurements. The previously used gain-factor from the encoder-feedback version were used as an initial guess, and the voltage to the helicopter-block were disconnected in order to reduce the noise from the motors. With the voltage disconnected we experienced as expected that we could increase the gain-factor and have a more aggressive tuning of the observer. We increased the gain-factor to 30 and connected the voltage so we could control the helicopter by the joystick. The noise present at the estimated states was now much higher. This corresponds with the theory as discussed; a less noisy measurement $\mathbf{y_m}$ can tolerate a more aggressive tuning of the observer (and vice versa). We ended up lowering the gain-factor to 20. The final tuning is in figure 20.

In figure 20 wee see that there are some noise, especially on elevation and pitch and their rates. This noise could be reduces by lowering the gain-factor even more, but lowering the gain-factor and $\mathbf{L}$ would result in a slower error dynamics. From equation 49b we can see that lowering $\mathbf{L}$ would result in the model of the state estimator having a larger impact, and therefore leave the observer more vulnerable to modelling inaccuracies. As this is the a pretty heavy linearized system this would have larger effect if the system is far away from the linearization point. We tried to reduce the gain-factor even more, but the dynamics became to slow, and the state estimates were not good enough to be able to control the helicopter by the joystick.

6 Lab day 4

6.1 Full state-space model

To model the system for use in a Kalman filter, we use the complete, linearized space-state model equal to the one we found in the preparations to day 1 (section 2.2), equation (11). The $\hat{\mathbf{C}}$-matrix in (52) is chosen to correspond with the five measured states from the IMU, $\mathbf{y}$ in equation (51) and will allow the Kalman filter to estimate the travel (λ).

$$\begin{aligned} \dot{\mathbf{x}} &= \hat{\mathbf{A}}\mathbf{x} + \hat{\mathbf{B}}\mathbf{u} \\ \mathbf{y} &= \hat{\mathbf{C}}\mathbf{x} \end{aligned} \tag{50}$$

$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \\ \dot{\lambda} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} \tilde{V}_s \\ V_d \end{bmatrix}, \mathbf{y} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} \tag{51}$$

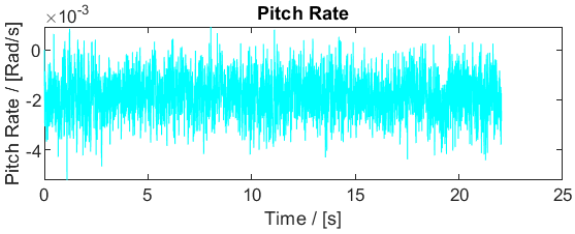
$$\hat{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ K_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \hat{\mathbf{B}} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_3 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \hat{\mathbf{C}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{52}$$

6.2 Observability

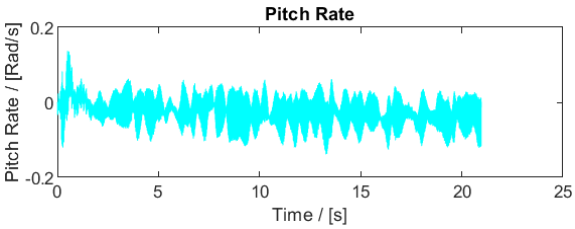
By introducing the λ state in to the state space model in section 6.1 and not measuring it, we end up with a system that is not observable. This is because the only way of obtaining λ is through integration of $\dot{\lambda}$. As discussed in 5.3.1, the system has no way of determining the initial condition. In our plots you will see a fairly good estimate of λ nevertheless. This is because we have manually set the initial condition to zero in the Kalman filter and λ is defined to be zero wherever the helicopter starts from. If we had defined a permanent zero for the travel angle, this would not work. This is the reason why the estimated state is not observable, but show that an estimate does not have to be useless only because we do not have observability.

6.3 The covariance matrix $\mathbf{R_d}$

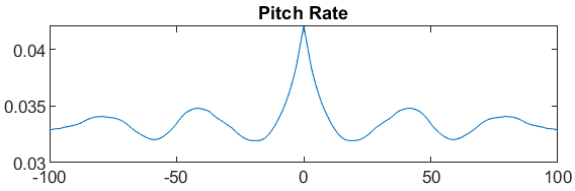
To implement the Kalman filter in our system, we first needed to find the covariance matrix R_d experimentally. At first we first did a reading of the IMU feedback signals with the helicopter laying still on the ground. The readings were saved as a matrix *y.signals.values* with about 20s of data. The readings was then processed in the Matlab function *cov* (22). The resulting matrix is shown in (53a). This would represent the covariances from just the measurement noise. Next we repeated the same procedures for data recorded while the helicopter was flying at the linearization point. This resulted in a matrix (53b). This matrix also includes noise and disturbances such as motor noise, vibrations, imbalances in the rotors and other effects of actually running the system. Non of the recorded noise sequences can be classified as white noise, there are clearly some periodicity present in our signals. This is easily verified by taking the autocorrelation of the signals, which should have been a tall spike at lag zero if the noise



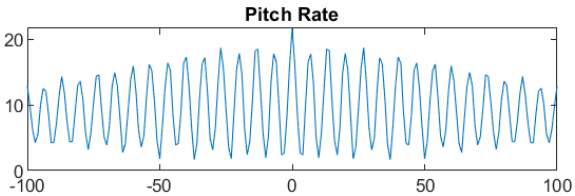
(a) Raw data with the helicopter laying still on the ground.



(b) Raw data with the helicopter flying at equilibrium.



(c) Autocorrelation of the data in 21a.



(d) Autocorrelation of the data in 21b.

Figure 21: Pitch rate as an example on noise in the measurements.

was perfectly white. In the case where the helicopter is laying still on the ground (figure 21a), the noise looks fairly white. Its autocorrelation sequence in figure 21c confirms that this is the case. It has a maximum at lag zero, but also some spikes at other values of lag. In this case the white noise assumption seems reasonable. In the case when the helicopter is flying in figure 21b, the noise is definitely not white and the autocorrelation in figure 21d confirms that. Considering that disturbances such as motor noise is periodic by nature, it is not very surprising but to call it white noise is a fairly bold assumption in this case. Also we should note that the noise is slightly biased in both cases. This is probably due to some drift in the gyro.

It is debatable how scientific this process is. To keep the helicopter flying at equilibrium we had to use a controller. The LQR-controller from section 4.3 with encoder feedback was chosen. One problem with this approach is that the noise from the encoders are fed into the loop. The motors dynamics are capable of filtering away some of the higher frequencies, but we cannot guarantee that nothing is fed through and into the measurements. This would pollute our measurement noise with encoder noise and the R_d matrix would be less accurate. The data we got with the helicopter laying still is probably more correct in that manner, but obviously lacks the some of the properties from a flying helicopter.

$$\mathbf{R_{d,still}} = \begin{bmatrix} 138,69 & 6,78 & 13,66 & -4,38 & 6,67 \\ 6,78 & 76,87 & 20,17 & -17,30 & -19,90 \\ 13,66 & 20,17 & 4975,02 & 274,54 & 34,26 \\ -4,38 & -17,30 & 274,54 & 204,73 & 31,44 \\ 6,67 & -19,90 & 34,26 & 31,44 & 106,06 \end{bmatrix} 10^{-8} \tag{53a}$$

$$\mathbf{R_{d,flying}} = \begin{bmatrix} 29,75 & 8,58 & 55,86 & -25,85 & -1,64 \\ 8,58 & 14,04 & -29,30 & -13,62 & -20,96 \\ 55,86 & -29,30 & 507,16 & -52,45 & 97,44 \\ -25,85 & -13,62 & -52,45 & 123,08 & -77,47 \\ -1,64 & -20,96 & 97,44 & -77,47 & 346,53 \end{bmatrix} 10^{-4} \tag{53b}$$

```
R_d = cov(y.signals.values);
```

Figure 22: Matlab code to find the covariance matrix R_d

```
step_time = 0.002;
sys = ss(A_hat, B_hat, C_hat, 0);

d_sys = c2d(sys, step_time);

A_d = d_sys.a;
B_d = d_sys.b;
C_d = d_sys.c;
D_d = d_sys.d;
```

Figure 23: Code to discretize the system

6.4 Discretization and Kalman filter implementation

The state space model in 6.1 was discretized using the code in figure 23 to be used in the discrete Kalman filter. The Kalman filter was implemented in simulink as in figure 24. A split design was chosen to separate the correction and prediction step. A design with just one function block was also tested and worked just as well, but the split design can be a bit more intuitive/easy to work with/easy to troubleshoot. Initial values for the states was set according to the initial position of the helicopter at startup.

6.5 Kalman filter tuning

6.5.1 The impact of the $\mathbf{Q_d}$ matrix

The $\mathbf{Q_d}$ matrix represents the uncertainty model. A $\mathbf{Q_d}$ matrix with all elements equal to zero would mean that there are no uncertainty in our model, the system is perfectly modeled. In this case the Kalman filter will only rely on the model and completely ignore the measurements. If all the elements (on the diagonal) are infinity, we have absolutely no faith in our model and the Kalman filter will rely on the measurements only. Non of these cases should ever occur, since perfect modelling is close to impossible to achieve and having zero faith in your own model makes the Kalman filter a bad choice. A larger $\mathbf{Q_d}$ tells the Kalman filter that variations in the states is assumed to be large. And by looking at the implemented Kalman filter i figure 24, we see that by larger $\mathbf{Q_d}$, the larger the Kalman gain $\mathbf{K}$ something that mean that we 'boost' the measurement noise in the correction step: $\hat{\mathbf{x}}[\mathbf{k}] = \bar{\mathbf{x}}[\mathbf{k}] + \mathbf{K}(\mathbf{C_d}\mathbf{x}[\mathbf{k}] + \mathbf{v_d} - \mathbf{C_d}\hat{\mathbf{x}}[\mathbf{k}]$. Knowing that the $\mathbf{Q_d}$ matrix is really a covariance matrix, we can say that the elements on the diagonal must be positive. In other words, the uncertainty is somewhere between 0 and infinity. Because we are fairly confident about our model, it should hopefully be closer to 0. A diagonal matrix with only 1's seems like a nice starting point.

6.5.2 Tuning the $\mathbf{Q_d}$ matrix

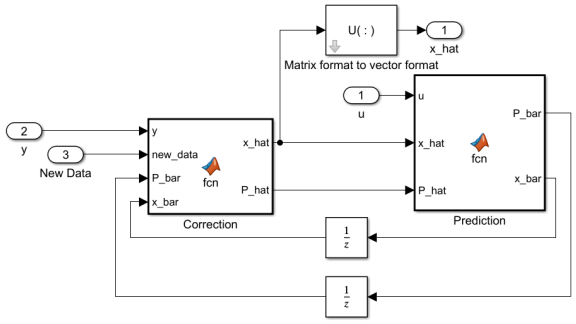
The encoder values was used for feedback to the LQR five-state controller. From the response of the Kalman filter with $\mathbf{Q_d}$ as the identity-matrix we observed that the estimated states were

```
function [x_hat, P_hat] = fcn(y, new_data, P_bar, x_bar, A_d, B_d, C_d, D_d, ...
    Q_d, R_d, Elev_offs)
%#codegen
persistent init_flag
if(isempty(init_flag)) %Initialial values
    P_bar = zeros(6);
    x_bar = [0;0;-Elev_offs;0;0;0];
    init_flag = 1;
end
K = P_bar*C_d'*inv(C_d*P_bar*C_d'+R_d);
if(new_data == 0)
    x_hat = x_bar;
    P_hat = P_bar;
else
    x_hat = x_bar+K*(y-C_d*x_bar);
    P_hat = (eye(6)-K*C_d)*P_bar*(eye(6)-K*C_d)' + K*R_d*K';
end
```

(a) Correction step

```
function [P_bar,x_bar] = fcn(u, ...
    x_hat, P_hat, A_d, B_d, C_d, ...
    D_d, Q_d, R_d, Elev_offs)
%#codegen
persistent init_flag
if(isempty(init_flag)) ...
    %Initialial values
    P_bar = zeros(6);
    x_bar = [0;0;-Elev_offs;0;0;0];
    init_flag = 1;
end
P_bar = A_d*P_hat*(A_d)'+Q_d;
x_bar = A_d*x_hat+B_d*u;
```

(b) Prediction step



(c) Kalman filter

Figure 24: The Kalman filter implemented in Matlab.

way to noisy, and the elements of the diagonal had to be reduced in order to rely more on the model and less on the noisy measurements. To make the tuning a bit easier and because we never had tuned a Kalman filter before we decided to consequently reduce the element in $\mathbf{Q_d}$ that corresponded to the noisiest state (reduce element two on the diagonal in order to reduce the noise in the second state). This turned out to be a good way to do it. It seemed that a change in the diagonal of $\mathbf{Q_d}$ resulted in a change in the corresponding state. This method was used systematically until we had a desired response. The Kalman filter state estimates were then used as feedback to the controller, and the method was used until we got a satisfactory response. The final tuning is as in figure 25.

It is tempting to just continue lowering the elements of $\mathbf{Q_d}$ until the response is a perfect curve with no noise. However, this is a bad idea and is illustrated in figure 26. The noise in the signal is incredibly low, but the price is oscillatory tendencies in the Kalman filter, which is completely useless as feedback. This tells us that there is a limit to how far we can go. It is

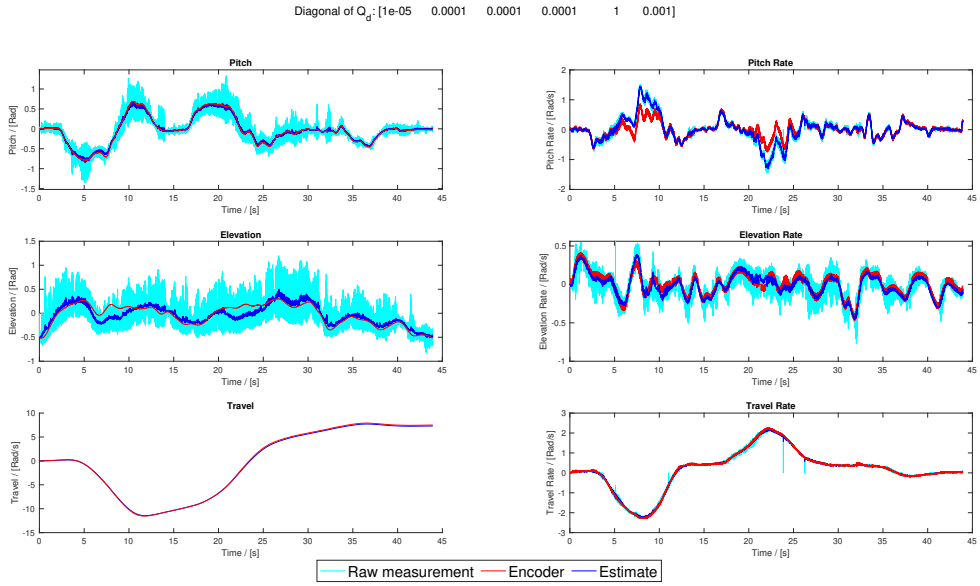


Figure 25: Final tuning of the Kalman filter

probably possible to go further than our final tuning in figure 25, but you wont get far before unwanted side effects start to arise and tuning of the filter will always have to be a compromise.

6.5.3 Effect of the loss of new measurements

As a part of the experimentation, a manual switch was added to the *new data* port of the Kalman filter. This way we could simulate that we had no new measurements. From the implementation in figure 24 we see that when we have no *new data* we do not correct the prediction step and rely completely on the prediction. The prediction step for $\hat{\mathbf{x}}$ is implemented as an discrete open loop observer, and we see that $\mathbf{K}$ and $\hat{\mathbf{P}}$ no longer affects the state vector $\hat{\mathbf{x}}$. From the theory behind an open loop observer, we know that the system model must be stable to be used in an open loop observer. In this case our system is not. The $\hat{\mathbf{A}}$ matrix (52) has 6 eigenvalues all equal to zero. Because of this we expect a very poor performance, since we are practically trying to simulate an unstable system. In terms of noise we expect the estimate to be smooth as there is no way the noise from the measurements will reach the estimated states. From figure 27 we see that the estimated states get bad very quickly as soon as the *new data* port is artificially disconnected. The estimates are also very smooth as expected as no noise propagate through the filter.

$$\hat{\mathbf{P}}_\infty = \begin{bmatrix} 1409 & 224 & -1 & 0 & -7710000 & -305800 \\ 224 & 5 & 0 & 0 & -101900 & -4311 \\ -1 & 0 & 14110 & 224 & 877 & 28 \\ 0 & 0 & 224 & 5 & 9 & 0 \\ -7710000 & -101900 & 877 & 9 & 5022000000 & 185900000 \\ -305800 & -4311 & 28 & 0 & 185900000 & 7075000 \end{bmatrix} \tag{54a}$$

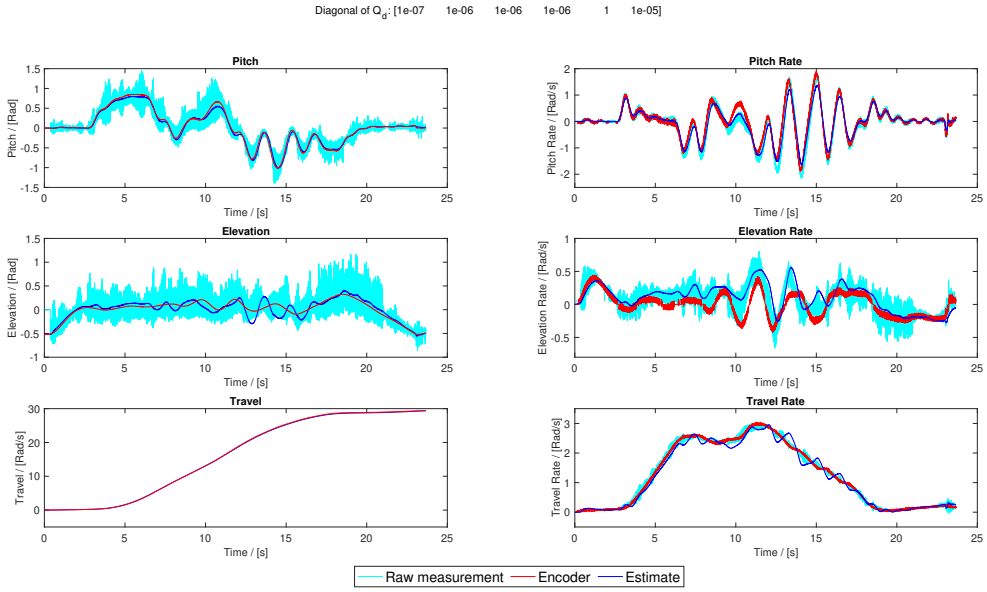


Figure 26: The effect of a Q_d matrix with too small elements

$$\mathbf{K}_\infty = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 547 & -4299 & 0.06 & -3 & 47 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{54b}$$

When the *new data* port is connected again the Kalman filter recovers very quickly (27). This is a mathematical property of the filter. As we disconnect the measurement artificially by setting *new data* to zero, $\hat{P}$ will grow very large over time (54a), which leads to K going towards K_∞ (54b). When we reconnected the measurement again we set the state-values we have measured equal to the measurement (56). The measurements are weighted very high as it has been long time since the Kalman filter has received any input and correction. This results in a step response. Equation (55) and (56) describes how the Kalman filter equations make this fast correction possible.

When *new data* is $= 0$:

$$\hat{\mathbf{P}}[k+1] = \mathbf{A_d}\hat{\mathbf{P}}[k]\mathbf{A_d^T} + \mathbf{Q_d} \tag{55a}$$
$$\hat{\mathbf{x}}[k+1] = \mathbf{A_d}\hat{\mathbf{x}}[k] + \mathbf{B_d}\mathbf{u}[k] \tag{55b}$$

When *new data* becomes 1 after being 0 for a long time:

$$\begin{aligned} \hat{\mathbf{x}}[k] &= \bar{\mathbf{x}}[k] + \mathbf{K}_\infty(\mathbf{y}[k] - \mathbf{C_d}\bar{\mathbf{x}}[k]) \\ &= (\mathbf{I} - \cancelto{0}{\mathbf{K}_\infty\mathbf{C_d}})\bar{\mathbf{x}}[k] + \mathbf{K}_\infty\mathbf{y}[k] \\ &= \mathbf{K}_\infty\mathbf{y}[\mathbf{k}] = \begin{bmatrix} y.p & y.\dot{p} & y.e & y.\dot{e} & \lambda & y.\dot{\lambda} \end{bmatrix}^T \end{aligned} \tag{56}$$

where:

$$\lambda = 547y.p - 4299y.\dot{p} + 0.06y.e - 3y.\dot{e} + 47y.\dot{\lambda}$$

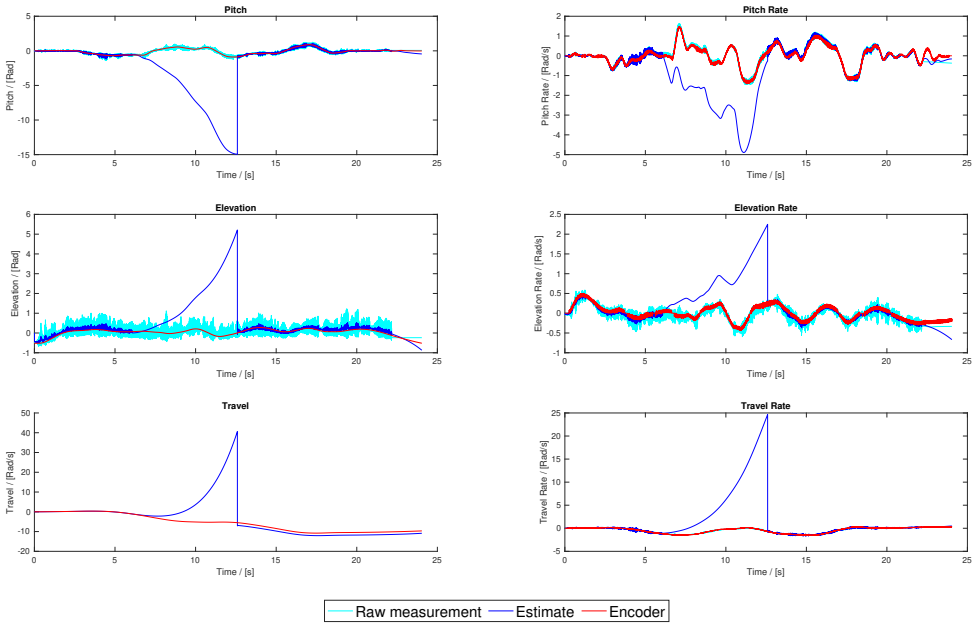


Figure 27: The effect of turning of the *new data* signal

In figure 27 we see the lambda estimate not being correct after we reconnect the measurement. Lambda is being estimated from the other measurements as we does not measure this state directly (56). It can be noted that lambda is essentially based on the measurement of p, $\dot{p}$ and $\dot{\lambda}$. As discussed in section 6.2 the system in not observable, but we get a pretty good estimate nevertheless. There is an error in lambda though, this is because the initial condition of the integral need to be estimated. How this is done is shown in (56). This results in an inaccurate estimate, due to linearization, noise and other imperfection. This shows us that a non observable state IS problematic to estimate.

The elements on row and column 5 in (54a) are very large. This may be a result of choosing $Q_d(5,5) = 1$. During tests it seemed liked $Q_d(5,5)$ could be chosen arbitrarily, however it may effect how λ is calculated after a long period of the measurement being disconnected.

6.5.4 Kalman vs Luenberger observer

Figure 28 shows a comparison of the performance of the Luenberger observer designed in lab day 3 (section 5) and the Kalman filter. Concluding which is better based on this figure alone would be dumb, as none of them are tuned to perfection (but fairly good) and they do not necessarily have the same area of use. Instead this section will focus a bit more on the pros and cons of both based on our experience at the lab.

The Kalman filter and the Luenberger observer are two estimators which are very similar in use. You can even argue that the Kalman filter is a more advanced version of the Luenberger observer. The basic idea is the same; if you have noisy measurements and a fairly good model of the system, you can simulate the system and correct the model inaccuracies with the measurements. In this way you can rely less on the noisy measurements and more on a noise-free

simulation. When tuned right, this will reduce the noise in the feedback signal. Additionally, both estimators are capable of estimating states. That is assuming the states are observable of course, see 5.3.1 for a discussion on that. In addition to being observable, we need a very accurate system model since estimated states cannot be corrected by dedicated measurements. A good model is necessary because the estimated states need to be corrected by other states/measurements and the mathematical relation should be as accurate as possible. Ideally, there should be a linear relation between the estimated state and the state it is estimated from. Our system model is heavily linearized and is therefore not suitable for such application. The main difference between the estimator is that the Kalman filter takes into account the statistical properties of the measurements. It also dynamically changes the estimator gain ($\mathbf{K}$) if the situation changes (for example the rate of incoming measurements or even loss of measurements).

Our experience with the estimators is that the Kalman filter was easier to work with. Once the $\mathbf{R_d}$ matrix was set, it was just a matter of choosing the $\mathbf{Q_d}$ matrix experimentally. This was an easy task since the elements in the $\mathbf{Q_d}$ matrix actually corresponded directly to the individual states (unlike a certain LQR method). The tuning was just a matter of adjusting the noise vs ability to follow the system for each state and we were done. The Luenberger observer was more difficult to tune, since changing the poles did not have the same distinct effect on single states, but instead all of the states. This meant we settled for a less good tuning compared to the Kalman filter.

In general it seems like the Kalman filter is the better choice if the measurement are really noisy **and** you have a way to find the properties of the noise. Tuning the Kalman filter does not seem like much fun if you have to guess both the $\mathbf{R_d}$ and the $\mathbf{Q_d}$ matrices. In that case the Luenberger observer is a better choice. It can be tuned via pole placement, which is still guesswork, but at least better than trying to guess covariances and such. If the measurements are not particularly noisy it seems like a bad move to include an estimator, but there are at least a couple of situations where they are useful anyways. If the rate of measurements is lower than what you want for your feedback-loop, an estimator can do the interpolation with a decent system model. This is implemented in our Kalman filter with the *new data* signal. Also, if you absolutely have to estimate a non measured state, that can be done. This requires a thorough modelling work though.

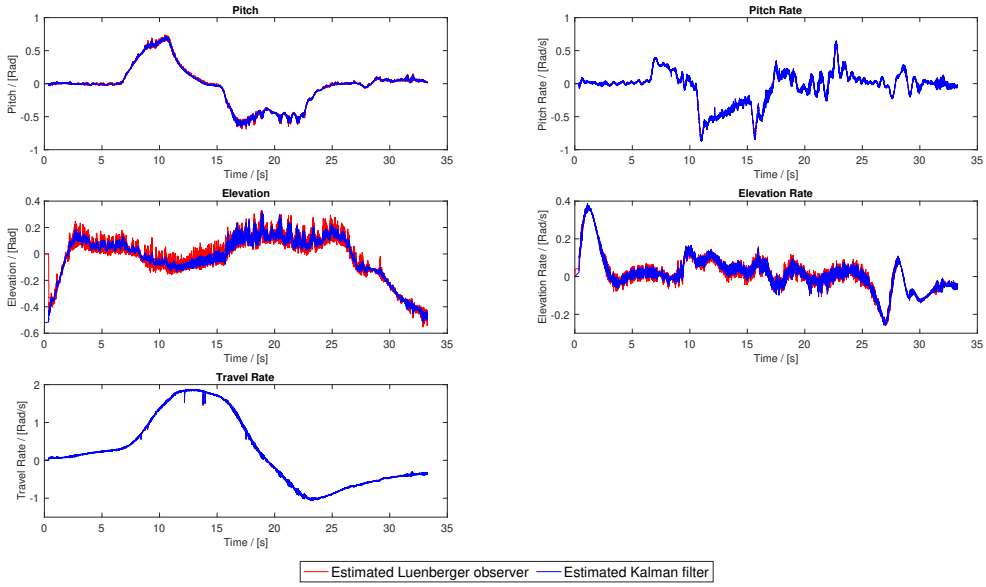


Figure 28: The Kalman filter vs the Luenberger observer

7 Solutions to encountered problems

This section contains our solutions to problems we encountered during the lab. They are placed here because they were implemented and tested after the other lab days using the Kalman filter system in 6. They are also to be presented together with the Kalman filter and not on their respective lab days.

7.1 Day 2: Integrator windup

During our experimentation with more aggressive tuning on the 5-state LQR-controller 4.3, we experienced standing oscillations in pitch and elevation occurring at high elevation angles. This should not be the case as all system poles are positioned in the left half-plane. Initially we decided to tune the system less aggressively to reduce this effect. During the next lab-sessions we hypothesized that this could be due to saturation effects which is likely to happen at high elevation angles when the motors do not have much headroom in terms of power. Since the problem only occurred when using a controller with integral effect, we figured that we were probably experiencing integrator wind-up, which is likely to introduce such oscillations. A simple check was done by measuring integral values and the result was somewhat disappointing. We measured high integrator values when provoking standing oscillations, but not nearly as much as we had hoped for.

Nevertheless we continued by capping the elevation rate-integral to a reasonable value. When testing this modified controller we were unable to provoke standing oscillations and the helicopter in general felt a bit "sharper" to control at high elevation angles. However this comes with a price; the idea of introducing the elevation integral was to make the helicopter capable of maintaining altitudes different from zero. Capping the integrator means that the helicopter will not be able to maintain altitude at very high elevation angles. The maximum angle it can still be able to maintain altitude is directly depending on the value used to cap the integrator, so this would have to be tuned to reduce the cons of the anti-windup. Since our testing was done using a less aggressive controller, we assume that the windup would be more significant in more aggressive controllers where the oscillations happen more often and are more significant. If we knew all of this before we started, we would probably have used anti-windup from the beginning and possibly achieved a faster system without sacrificing stability. It should be noted that anti-windup introduces another non-linearity and can complicate stability analysis.

7.2 Day 3 and 4: IMU drift

During the lab, we experienced an increasing drift in the IMU measurements. This was corrected for by adding constant values to the raw measurements until they were correct. Figure 29 shows how it was implemented in Simulink.

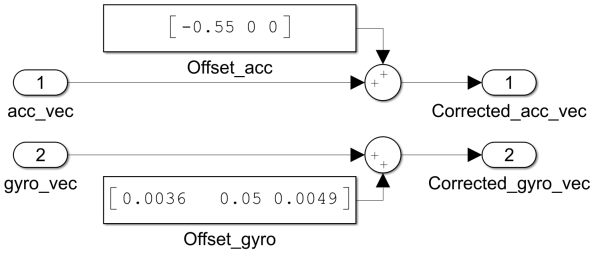


Figure 29: Correction system for IMU drift.

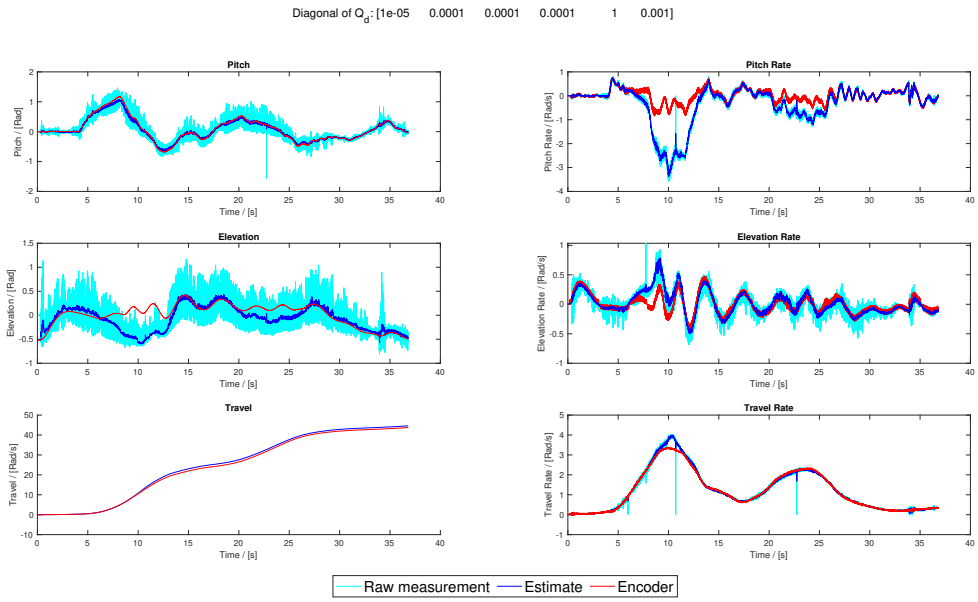


Figure 30: Effects of centripetal acceleration on the system

7.3 Day 3: Accelerometer compensation

In section 5.3.3 we concluded that a system for correcting the accelerometer output when the helicopter is exposed to acceleration would improve our feedback. We also stated that it should be fairly simple to avoid too many dependencies between the states. Our suggestion is therefore to use a simplified and trigonometry-free correction system based on the following observations from figure 30:

- Accelerations due to the helicopter accelerating is not a major issue. (Pitch would have been more affected if that was the case).
- Centripetal acceleration due to high travel rate is a major issue. (Elevation, which is calculated from acceleration along x-axis is heavily affected).
- The major issues with rates being way off at high travel rates are assumed to be because they are corrected with values from the elevation state.
- The helicopter tends to stabilize around elevation equilibrium at high travel rates. (Look at the encoder elevation during high travel rates, this is a physical property of a rotating pendulum).

The last observation is what makes the simplification possible, since high elevation angles and high travel rates can be assumed to not occur simultaneously. We choose to only correct for the centripetal acceleration resulting in equation (57a). (42) can be modified into (57c) Since we can assume $e \approx 0$, the equation can be simplified to equation (57b). This is done to get rid of a nasty feedback loop in the calculation of the elevation state. Then we can modify equation (42) to get (57d).

$$a_{c,x} = \dot{\lambda}^2 l_h \cos^2(e) \tag{57a}$$
$$\tilde{a}_{c,x} = \dot{\lambda}^2 l_h \tag{57b}$$
$$\hat{a_x} = a_x + a_{c,x} \tag{57c}$$
$$\tilde{a_x} = a_x + \tilde{a}_{c,x} \tag{57d}$$

We tested both the simplified version in (57d) and the nonlinear version (57c). The results are shown in figure 31 (simplified) and figure 32 (not simplified). When compared to figure 30 it is clear that both systems function according to expectations. Especially considering the extreme conditions the system is tested with. We should also note that states other than elevation are corrected. This confirms our hypothesis that the rates were off due to being corrected by the elevation state which is the only one directly affected by the centripetal acceleration.

The impact the correction system had on the helicopter is even more impressive. Before our system tended to become unstable at high pitch angles during high travel rates. We assumed that this was due to the linearization and was unfortunate since increase travel rate tends to be a result of large pitch angles (it happens quite often). After the correction, the pitch actually got a fairly accurate feedback and we were capable of pushing the system to pitch angles way larger than before and still keep the system stable.

When comparing figures 57d and 57c there is no real difference (which is reasonable since e is close to 0). In this situation it would be logical to choose the non simplified version because it is more accurate. However, we are still very sceptical about introducing another dependency in our system. With the simplified version we already have these dependencies:

- e calculated based on $\dot{\lambda}$. (31), (57d) and (43b)
- $\dot{\lambda}$ corrected by e (46a).

The non simplified version also introduces:

- e calculated based on e (32), (57c) and (43b).

We honestly do not know how such system would behave at all times. This scepticism and the fact that we loose basically no performance from choosing the simplified version, it seems like this is our safest choice.

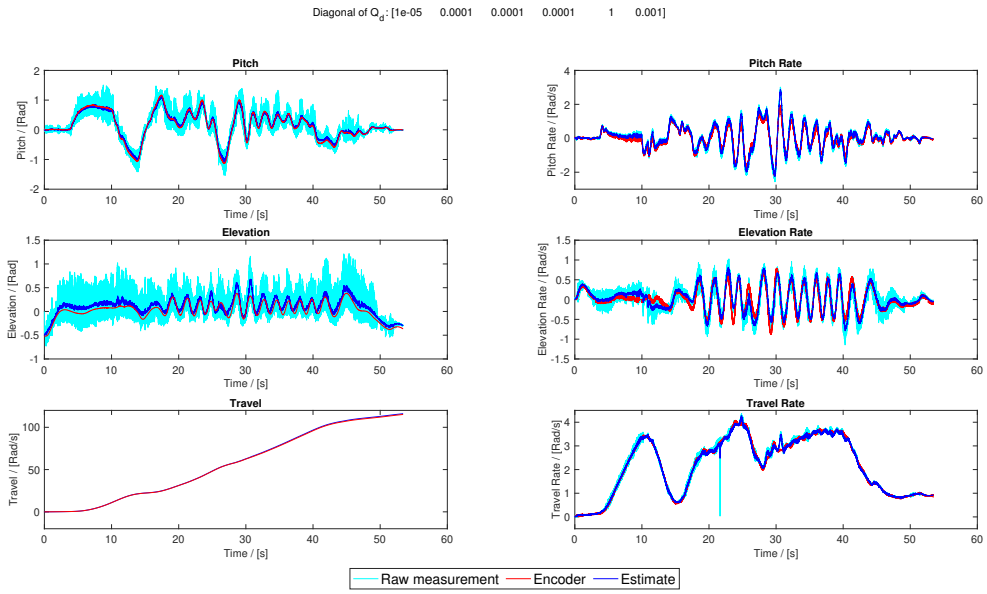


Figure 31: Centripental acceleration compensation, simplified version

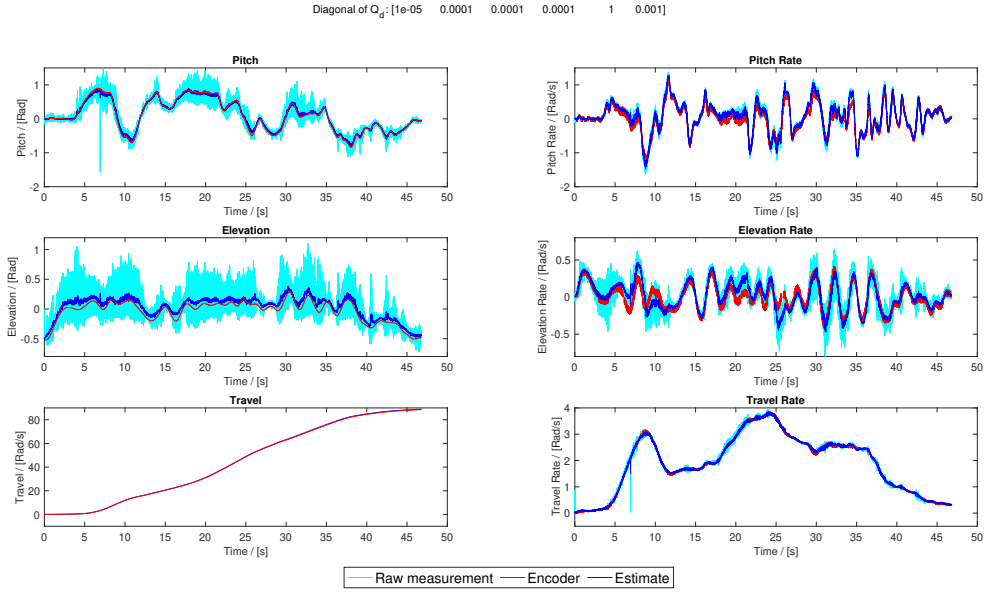


Figure 32: Centripental acceleration compensation, non simplified version

References

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

[2] Sverre Velten Rothmund. *Helicopter lab preparation*. Version 5.1 August 2020. Department of Engineering Cybernetics NTNU.