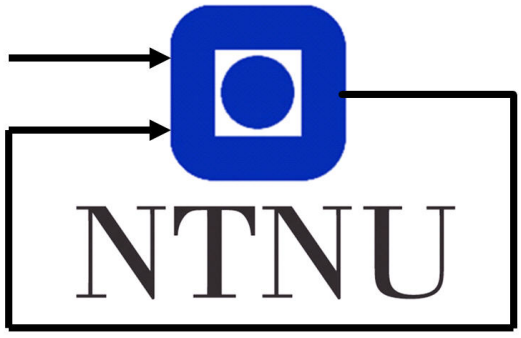
# TTK4115 - Helikopterlab

Gruppe 12
#490138 - Lars Musæus
#478394 - Niels Sole Semb
#490152 - Jakob Eide Grepperud

November 18, 2019




Department of Engineering Cybernetics

# Contents

# Introduction

This is the notorious helicopter lab in the subject of Linear Systems Theory and needs no further introduction.

Table 1: Parameters and values.

| Symbol | Parameter | Value | Unit |
|---|---|---|---|
| $l_c$ | Distance from elevation axis to the counter weight | 0.46 | m |
| $l_h$ | Distance from elevation axis to helicopter body | 0.66 | m |
| $l_p$ | Distance from pitch axis to motor | 0.175 | m |
| $K_f$ | Motor force constant | – | N/V |
| $J_e$ | Moment of inertia for elevation | – | kg m$^2$ |
| $J_t$ | Moment of inertia for travel | – | kg m$^2$ |
| $J_p$ | Moment of inertia for pitch | – | kg m$^2$ |
| $m_p$ | Mass of motor | 0.72 | kg |
| $m_c$ | Counter weight | 1.92 | kg |
| $F_f$ | Force from front motor | – | N |
| $F_b$ | Force from back motor | – | N |
| $F_{g,c}$ | Gravitational force on the counterweight | – | N |
| $F_{g,f}$ | Gravitational force on the front motor | – | N |
| $F_{g,b}$ | Gravitational force on the back motor | – | N |
| $V_f$ | Voltage supplied to front motor | – | V |
| $V_b$ | Voltage supplied to back motor | – | V |
| $V_{s,0}$ | Motor force need to keep the helicopter horizontal | – | V |
| $K_i, L_i$ | Constants | – | |

# 1 Part I – Mathematical modeling

## 1.1 Helicopter model

To be able to control the helicopter and analyze it's behaviour, it first has to be modelled mathematically; The physical forces, variables and constants are labelled and illustrated as show in Figure 1. In addition are the masses and distances shown in Figure 2.

The following is given by the assignment text: The propeller forces for the front and back propeller are given by $F_f$ and $F_b$, respectively. It is assumed that there is a linear relation between the voltages $V_f$ and $V_b$ supplied to the motors and the forces generated by the propeller

$$F_f = K_f V_f \tag{1}$$

$$F_b = K_f V_b \tag{2}$$

where $K_f$ is the motor force constant.

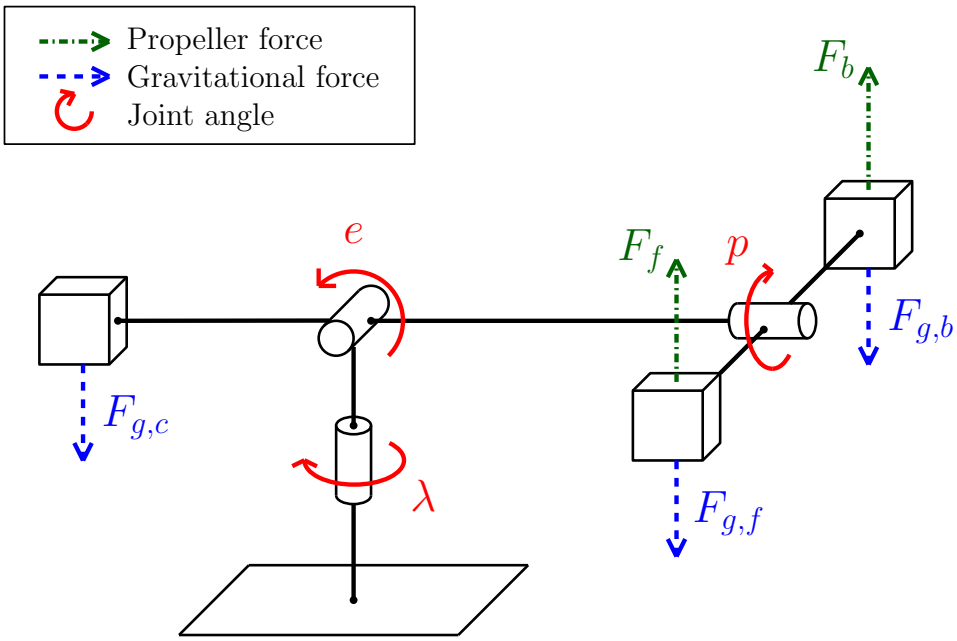



Figure 1: Helicopter model

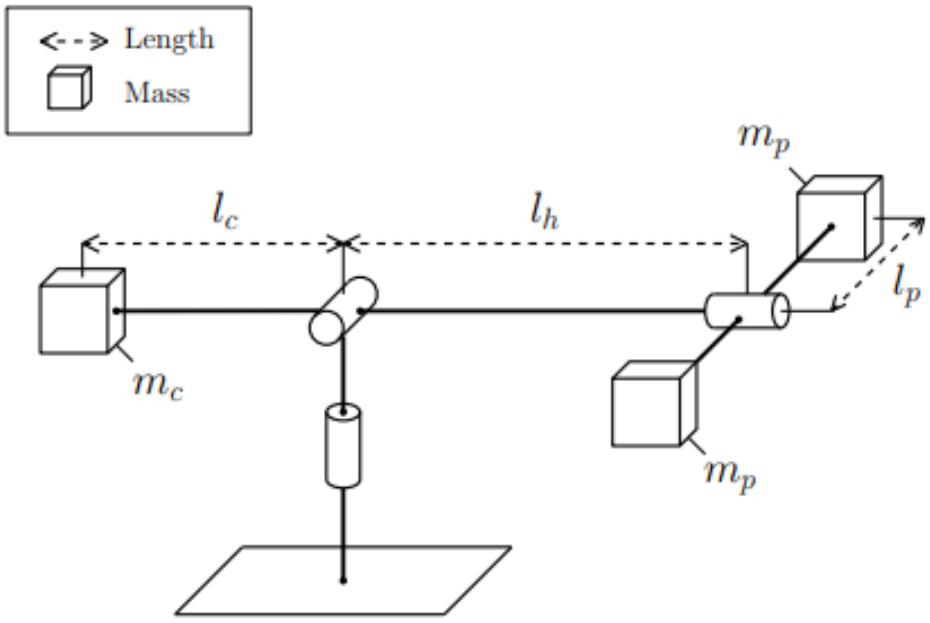



Figure 2: Masses and distances

$V_d$ is the difference between the motor voltages

$$V_d = V_b - V_f \quad . \tag{3}$$

$V_s$ is the sum of the motor voltages:

$$V_s = V_f + V_b \quad . \tag{4}$$

$\tilde{V_s}$ is the variable that serves as the system control input, and is a result of the sum of the motor voltages and the voltage needed to keep the helicopter horizontal:

$$\tilde{V_s} = V_s - V_{s,0} \tag{5}$$

## 1.2 Physical prerequisites

Applying torque to an object will cause the object to rotate. Torque is given by

$$\tau = \boldsymbol{F} \times \boldsymbol{r} \tag{6}$$

where $\boldsymbol{F}$ is the force on the moment arm $\boldsymbol{r}$, when $\boldsymbol{r}$ is at a right angle to the rotating axis.

The angular acceleration $\alpha$ is related to the applied torque $\boldsymbol{\tau}$ by Newton's Second Law of Rotation

$$\sum \tau = J\alpha \tag{7}$$

where $J$ is the moment of inertia about the rotating object.

## 1.3 Problem 1 - Equation of motion

In the equations of motion for the pitch angle $p$, the elevation angle $e$, and the travel angle $\lambda$, it can be assumed small departures from the linearization point. Consequently small angle approxiamations can be used:

$$\cos(x) \approx 1, \sin(x) \approx x \tag{8}$$

### Moment of inertia about the pitch axis

The difference between the forces from the propellers, $F_b - F_f$ is proportional to the torque about the pitch axis. Applying Equation (6), Equation (7) and Equation (8):

$$\begin{aligned} J_p\ddot{p} &= l_p(F_f - F_b - F_{g,f} + F_{g,b}) \\ &= l_p(K_f V_f - K_f V_b - m_p g + m_p g) \\ &= K_f l_p V_d \\ &= L_1 V_d \end{aligned} \tag{9}$$

### Moment of interia about the elevation axis

For the elevation angle $e$ Equation (6), Equation (7) and Equation (8) gives

$$\begin{aligned} J_e\ddot{e} &= F_{g,c}l_c\cos(e) - 2F_{g,f}l_h\cos(e) + (F_f + F_b)l_h\cos(p) \\ &= F_{g,c}l_c + l_h(F_f + F_b - 2F_{g,f}) \\ &= F_{g,c}l_c + l_h(K_f V_s - 2F_{g,f}) \\ &= K_f l_h(V_s - \frac{2F_{g,f}}{K_f} + \frac{F_{g,c}l_c}{K_f l_h}) \\ &= L_2(V_s - V_{s,0}) \end{aligned} \tag{10}$$

where $V_{s,0}$ is the voltage needed for the helicopter to stay at the linearization point

### Moment of interia about the travel axis

For the travel angle $\lambda$ Equation (6), Equation (7) and Equation (8) gives

$$\begin{aligned} J_\lambda\ddot{\lambda} &= (F_f + F_b)l_h\sin(p)\cos(e) \\ &= K_f l_h V_s p \\ &= L_3 V_s p \end{aligned} \tag{11}$$

Notice how the torque varies with the elevation angle of the helicopter.

### Constants and the result of linearization

From Equation (9), Equation (10) and Equation (11), the constants can be determined:

$$L_1 = K_f l_p \tag{12a}$$
$$L_2 = K_f l_h \tag{12b}$$
$$L_3 = K_f l_h \tag{12c}$$

As a result of modifying equation eq. (9), eq. (10) and eq. (11) to use the definition $\tilde{V_s}$, and removing all higher order terms in light of out linearized system, we get the equations:

$$\ddot{p} = K_1 V_d \tag{13a}$$
$$\ddot{e} = \frac{L_3}{J_e}\tilde{V_s} = K_2\tilde{V_s} \tag{13b}$$
$$\ddot{\lambda} = K_3 p \tag{13c}$$

where the constants are defined as follows:

$$K_1 = \frac{L_1}{J_p} \tag{14a}$$

$$K_2 = \frac{L_2}{J_e} \tag{14b}$$

$$K_3 = \frac{L_3 V_s}{J_\lambda} \tag{14c}$$

### 1.4 Problem 2 - Parameter identification

Resting the helicopter at the table and scoping the measured output from the encoder gives the pitch offset, while the elevation offset is found by holding the helicopter horizontal. The offsets found are given below in rad and were placed in the Matlab Block "Heli 3D":

$$E_{off} = -0.53$$
$$P_{off} = 0.088$$

Attempting to estimate $V_{s,0}$ is tough given the difficulty of controlling the helicopter without a controller. This is made significantly easier by manually balancing the helicopter while finding the elevation equilibrium. The measured value was

$$V_{s,0} = 7.2\text{V} \tag{15}$$

Using eq. (6) and eq. (7) we can derive the sum of all the torques in our system that keeps it horizontal:

$$\sum \tau = J\alpha = 0 \tag{16}$$

$$(F_f + F_b)l_h - 2m_p g l_h + m_c g l_c = 0 \tag{17}$$

Inserting eq. (1) and making the assumption that $V_s = V_{s,0}$ in our equilibrium:

$$K_f V_{s0} l_h - 2m_p g l_h + m_c g l_c = 0 \tag{18}$$

$$K_f = \frac{g(2m_p l_h - m_c l_c)}{V_{s,0} l_h} = 0.13872\text{N}V^{-1} \tag{19}$$

### 1.5 Problem 3 - Verification and manual control

The linearized model was implemented as shown in fig. 3. The system was tried out in two scenarios; one from the linearization point and one where our linearization assumptions in elevation, pitch and travel are not theoretically valid anymore.

As expected, when the helicopter is kept around the linearization point, our linearized model performs acceptable. However, when we begin to get a bit of deviation,

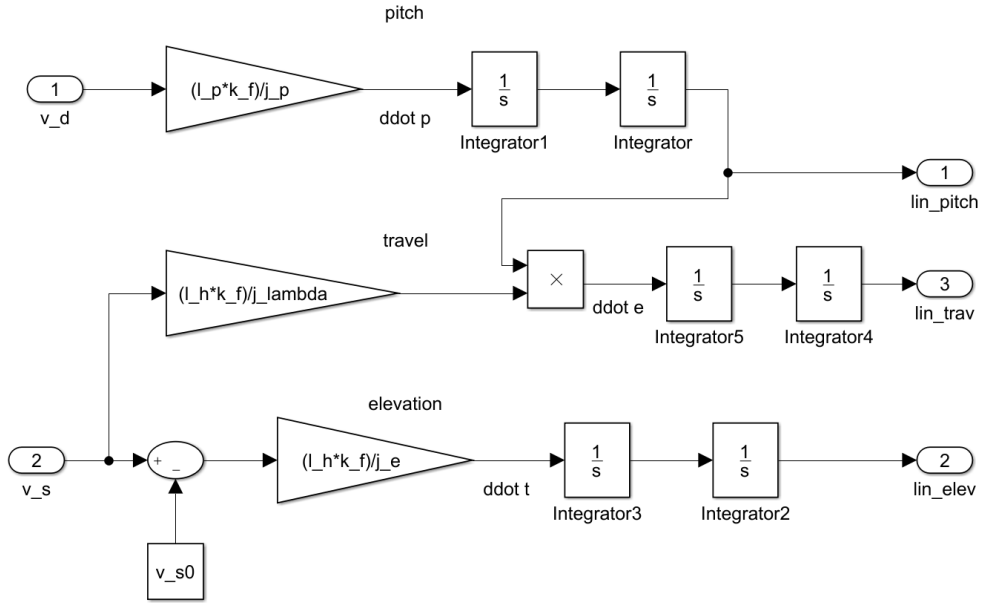
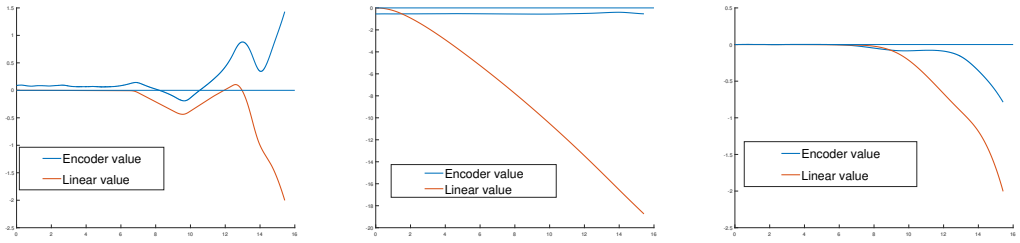



Figure 3: lineriazed model in Matlab



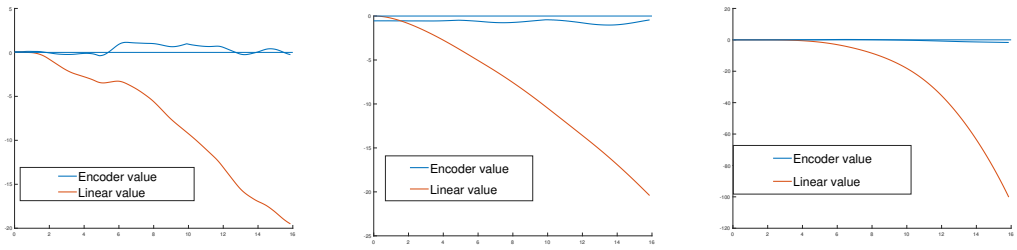

(a) Pitch (b) Elevation (c) Travel

Figure 4: Plots for pitch, elevation and travel when the helicopter is launched from the linearization point

the linearized model is not good enough. This can be seen in fig. 4a and fig. 4c. It's worth noticing that when the travel deviate from the linearizaton point it diverges much faster than pitch and elevation. This is because of the 4-part integration that is used to calculate travel. In addition does fig. 4b show that when $\tilde{V}_s$ is not equal zero, which will happens from time to time, the elevation in our model will diverge at a high speed. The reason why it is the elevation that is affected the most, is because that's the only variable that uses $\tilde{V}_s$ as reference.

Inspecting the plots from fig. 5, we can immediately tell that the linearized model is preforming poorly. It flies from the assumptions that the three angles from pitch, elevation and travel is close to equal zero ($p = e = \lambda = 0$), which is not the situation

s




(a) Pitch (b) Elevation (c) Travel

Figure 5: Plots for pitch, elevation and travel when the helicopter is launched outside the linearization point

here. The most optimal scenario is when that is the case and when $V_{s,0}$ is applied. Open loop systems, in general, is not a good procedure to regulate a system, as we have seen from our plots. The physical model will always be non-linear and have differences, such as weight differences between the the front and the back motor. This can be solved however when a regulator is implemented, as we can see in the next parts.

# 2 Part II – Monovariable control

## 2.1 Problem 1 - PD controller

The PD contoller to implement on the system from section 1 is given by

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p} \tag{20}$$

with $K_{pp},\ K_{pd} > 0$. Substituting into eq. (13) gives:

$$\ddot{p} = K_1 K_{pp}(p_c - p) - K_{pd}\dot{p}$$

With initial conditions being zero, $p(0) = \dot{p}(0) = 0$, we can express transform the differential equation into the Laplace domain:

$$\begin{aligned} s^2 p(s) &= K_1 K_{pp}(p_c(s) - p(s)) - sK_{pd}p(s) \\ p(s)(s^2 + K_{pd}s + K_1 K_{pp}) &= K_1 K_{pp} p_c(s) \end{aligned}$$

Simple algebraic manipulation then leads us to the transfer function between pitch angle reference to pitch angle

$$\frac{p}{p_c}(s) = \frac{K_1 K_{pp}}{s^2 + K_1 K_{pd} s + K_1 K_{pp}} \quad . \tag{21}$$

## 2.2 Problem 2 - Pole Placement

We can now use this transfer function to place the poles in order to give us an optimal response. We may rewrite eq. (21) as

$$\frac{p}{p_c}(s) = \frac{K_1 K_{pp}}{(s - p_1)(s - p_2)} \quad , \tag{22}$$

where $p_1$ and $p_2$ are the poles of the system.

By comparing the characteristic polynomials of eq. (21) and eq. (22), we get the following:

$$\begin{aligned} s^2 - (p_1 + p_2)s + p_1 p_2 &= s^2 + K_1 K_{pd} s + K_1 K_{pp} \\ p_1 + p_2 &= -K_1 K_{pd} \\ p_1 p_2 &= K_1 K_{pp} \end{aligned}$$

With $K_1 = \frac{L_1}{J_p}$ (see eq. (14)), we can now decide the controller parameters $K_{pd}$ and $K_{pp}$ from the position of poles $p_1$ and $p_2$:

$$K_{pd} = \frac{p_1 + p_2}{-K_1} \tag{23a}$$

$$K_{pp} = \frac{p_1 p_2}{K_1} \tag{23b}$$

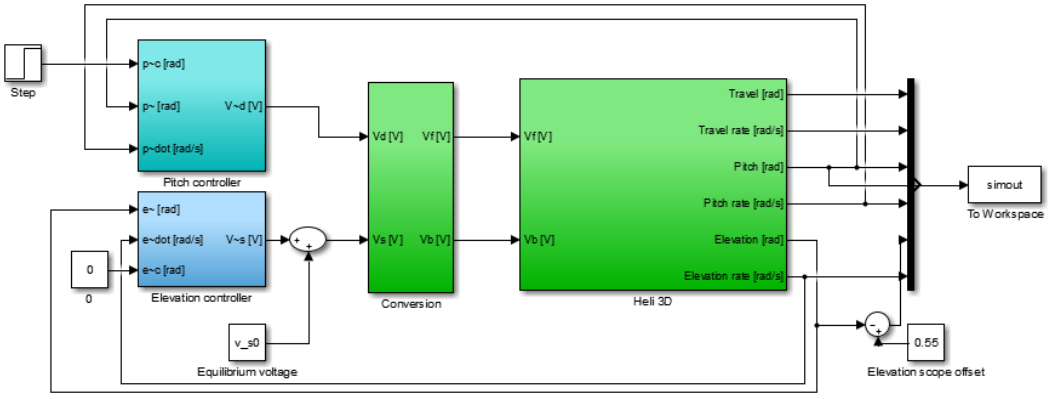



Figure 6: Pitch controller setup

Now, we experiment with the poles in different positions. From controlling theory, we know that poles in the right half plane will generate an unstable system. If the poles are on the imaginary axis, the system will be marginally stable. We therefore want negative real poles or a pair of complex conjugated poles with negative real part.

The controller is tested with different parameter values in order to identify the best response. Our setup was to let the helicopter start at ground level, with $e(0), p(0), \dot{e}(0), \dot{p}(0) = 0$. Engines are started in order to let the helicopter reach the linearization point. After 4 seconds, a step signal changes the pitch angle reference from 0 to 0.3rad. The implementation is seen in fig. 6.

We first started setting both poles equal to a negative real value. As we know from theory, this will give a critically damped system with $\zeta = 1$. Through experimenting, we found that the system responded fairly slow with values such as $p_1 = p_2 = -1$. We found that poles with a larger negative value than $p_1 = p_2 = -5$ resulted in large oscillations, and less than this resulted in a slow system. See fig. 7.

Though in theory, a double pole will yield good results, this is not always the case for physical systems, due to imperfect modelling and different non-linear factors. We therefore experimented with the position of the poles, including two real poles with different negative values. This leads to an overdamped system.

We also experimented with complex conjugated poles, meaning the system is under-damped. Examples of both underdamped and overdamped responses are shown in fig. 8. As expected, we found that by making the system slightly underdamped, we got a faster but oscillating response. If we increase the magnitude of the imaginary parts of $p_1$ and $p_2$, the oscillations increased as the system gets more unstable.

Different placement of the poles and the resulting $K_{pd}$ and $K_{pp}$ are shown in table 2. Note that the table is divided into 3 sections, separating the double poles, different real poles and complex conjugated poles, respectively. Though double poles

Table 2: Parameters deduced from pole placement.

| $p_1$ | $p_2$ | $K_{pd}$ | $K_{pp}$ |
|---|---|---|---|
| $-2$ | $-2$ | 1.54 | 1.54 |
| $-5$ | $-5$ | 3.85 | 9.63 |
| $-8$ | $-8$ | 6.16 | 24.66 |
| $-6$ | $-4$ | 3.85 | 9.25 |
| $-25$ | $-2$ | 10.40 | 19.26 |
| $-3-0.3i$ | $-3+0.3i$ | 2.31 | 3.50 |
| $-4-2.5i$ | $-4+2.5i$ | 3.08 | 8.57 |
| $-5-5i$ | $-5+5i$ | 3.85 | 19.26 |

in theory yields the best result, we found that a slightly overdamped system gave us a smooth and quick response. The poles we used were $p_1 = -6, p_2 = -4$.

The pitch controller is a PD-controller, so a stationary deviation from the reference is to be expected, as seen in fig. 7 and fig. 8. In general, it was hard to notice the subtle differences in the helicopter movement when choosing the controller parameters from pole positioning. An alternative method of choosing $K_{pd}$ and $K_{pp}$ was found to be better, and is described in the next subsection.

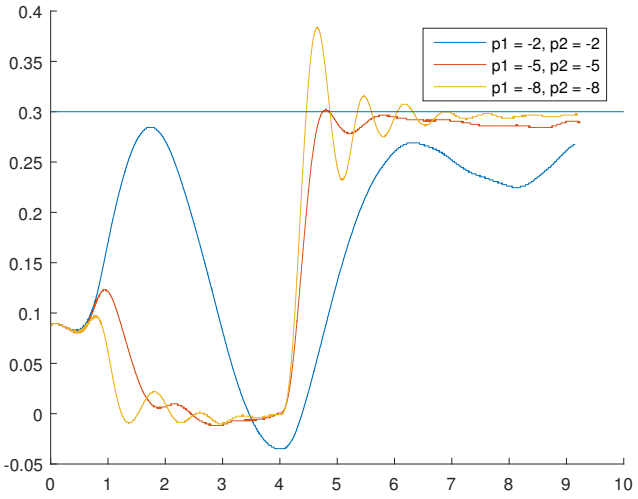



Figure 7: Pitch response with $p_c = 0 \rightarrow 0.3$rad, using double poles

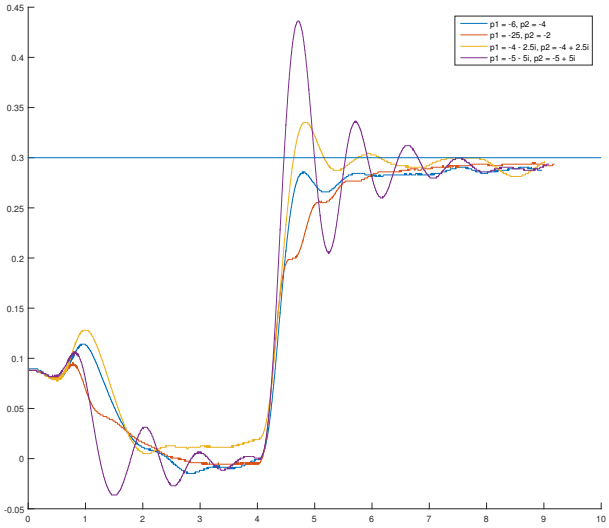



Figure 8: Pitch response with $p_c = 0 \rightarrow 0.3$rad, using different poles.

### 2.3 Problem 3 - Harmonic Oscillator

Instead of using pole positioning, we will regard the transfer function from eq. (21) as an harmonic oscillator. We then have a second-order linear system on the general form

$$h(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + {\omega_0}^2} \tag{24}$$

In general, we would want a critically damped system with $\zeta = 1$, as this theoretically leads to the most optimal response. Comparing the two functions eq. (24) and eq. (21) gives us expressions for $\zeta$ and $\omega_0$ which we use in order to give us the relation between $K_{pp}$ and $K_{pd}$:

$$\omega_0^2 = K_1 K_{pp} \tag{25a}$$

$$\omega_0 = \sqrt{K_1 K_{pp}} \tag{25b}$$

$$\zeta = \frac{K_1 K_{pd}}{2\omega_0} = \frac{K_1 K_{pd}}{2\sqrt{K_1 K_{pp}}} = \frac{K_{pd}}{2}\sqrt{\frac{K_1}{K_{pp}}} \tag{26}$$

$$\sqrt{K_{pp}} = \frac{K_{pd}\sqrt{K_1}}{2\zeta} \tag{27a}$$

$$K_{pp} = \frac{K_{pd}^2 K_1}{4\zeta^2} \tag{27b}$$

Table 3: Parameter values deduced from $\omega_0$ and $\zeta$.

| $K_{pd}$ | $K_{pp}$ | $\zeta$ |
|---|---|---|
| 2.8 | 5.09 | 1 |
| 4 | 10.38 | 1 |
| 6 | 23.35 | 1 |
| 6 | 16.22 | 1.2 |
| 4 | 16.22 | 0.8 |
| 4.5 | 36.5 | 0.6 |

By choosing $\zeta = 1$, it is now possible to tune the aggressiveness of the controller while in theory maintain critical damping by tuning the gain $K_{pp}$. By experimenting with different values for $K_{pd}$, we found the corresponding values for $K_{pp}$ through eq. (27), as shown in table 3. The setup was similar to the one in section 2.2. We found the response to be too slow with $K_{pp} < 10$, and increasing $K_{pp}$ more than this resulted in an oscillating system. This is to be expected, as it is equal to placing the poles of the system further away from the origin. As fig. 9 shows, a $K_{pp} \approx 10$ gave a fairly good response.

By changing the damping coefficient $\zeta$ to a value different than 1 will change the system dynamic. As seen in fig. 10, increasing $\zeta$ will lead to an overdamped system. This seemed to give the smoothest and fastest response for the pitch controller, even if it in theory is not. Decreasing $\zeta$ to a value $< 1$ resulted in a heavily oscillating system. As fig. 10 shows, decreasing the value too much results in an unstable system. The values of the plotted responses are found in table 3.

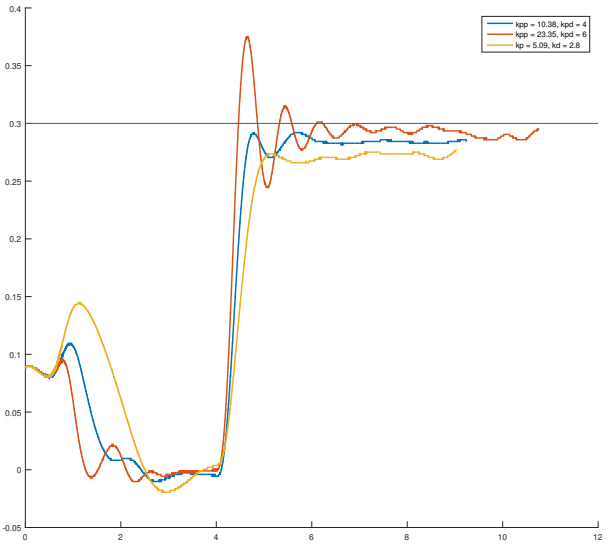



Figure 9: Responses with $\zeta = 1$.

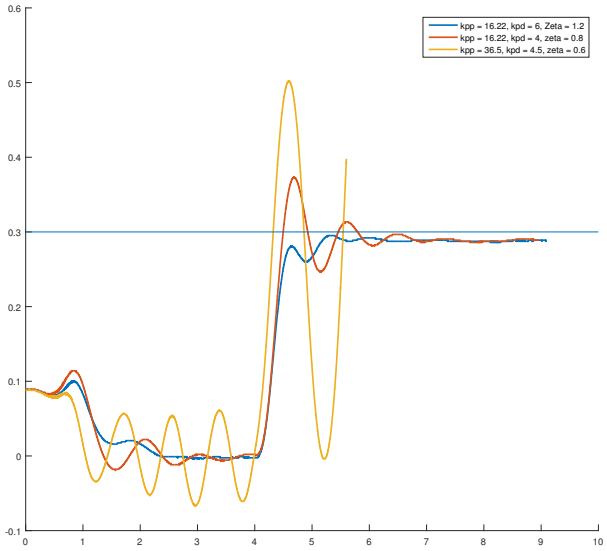



Figure 10: Responses with $\zeta \neq 1$.

# 3 Part III - Multivariable Control

## 3.1 Problem 1 - State-space formulation

We want to put the system on a state-space formulation of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{28}$$

with the new state and input of the system given by

$$\boldsymbol{x} = \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} \quad \text{and} \quad \boldsymbol{u} = \begin{bmatrix} \tilde{V}_s \\ V_d \end{bmatrix} \quad . \tag{29}$$

Using eqs. (13a) and (13b)gives us the equations

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{p} \\ \ddot{p} \\ \ddot{e} \end{bmatrix} = \begin{bmatrix} x_2 \\ K_1 u_2 \\ K_2 u_1 \end{bmatrix} \quad , \tag{30}$$

which yields through eq. (28) the following system matrices

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad , \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix} \quad . \tag{31}$$

We want to examine the controllability of the system. We find this by using the properties of the controllability matrix $\boldsymbol{\mathcal{C}}$. As our $\boldsymbol{A}$-matrix is an $n$x$n$-matrix with $n = 3$, the controllability matrix is computed to be

$$\boldsymbol{\mathcal{C}} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{A}^2\boldsymbol{B} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & K_1 & 0 & 0 \\ 0 & K_1 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad . \tag{32}$$

It is clear that $rank(\boldsymbol{\mathcal{C}}) = 3$, which means $\boldsymbol{\mathcal{C}}$ has full rank, and thus, according to [2, Theorem 6.1] the system is controllable. In simple terms, this means that we are able to directly alter the states from changing the input signal, within a finite time interval. No information about the history of the system is needed in order to predict the states in the future, only the current system state. In the next problem, we will use this ability as we implement an LQR.

## 3.2 Problem 2 - Linear Quadratic Regulator

The reference for the pitch angle $p$ and the elevation rate $\dot{e}$ is now set to be $\boldsymbol{r} = [p_c, \dot{e_c}]^T$, where $p_c$ and $\dot{e_c}$ are given by the joystick output on the x-axis and y-axis, respectively.

Next, it is desired to implement a controller of the form

$$\boldsymbol{u} = \boldsymbol{F}\boldsymbol{r} - \boldsymbol{K}\boldsymbol{x} \tag{33}$$

where $\boldsymbol{F}\boldsymbol{r}$ represents the reference feed forward, and $\boldsymbol{K}\boldsymbol{x}$ represents the state feedback.

$\boldsymbol{K}$ is found such that $\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x}$ optimizes the cost function

$$J = \int_0^\infty (\boldsymbol{x}^T \boldsymbol{Q}_{\text{LQR}} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R}_{\text{LQR}} \boldsymbol{u}) dt \quad . \tag{34}$$

The weighting matrices $\boldsymbol{Q}_{\text{LQR}}$ and $\boldsymbol{R}_{\text{LQR}}$ are given as

$$\boldsymbol{Q}_{\text{LQR}} = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \quad \text{and} \quad \boldsymbol{R}_{\text{LQR}} = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix} \quad , \tag{35}$$

representing the costs for state error and the costs for input, respectively. By increasing the magnitude of the entries on the diagonal of the $\boldsymbol{Q}_{\text{LQR}}$ matrix, we penalize the error of the corresponding state, e.g. $q_1$ will penalize the pitch angle error $p - p_c$. Similarly, increasing the value of $r_1$ in the $\boldsymbol{R}_{\text{LQR}}$ matrix, we penalize the system for using a high input voltage $\tilde{V}_s$. An increase of weight corresponds to the importance of the LQR minimizing said factor.

$\boldsymbol{Q}_{\text{LQR}}$ and $\boldsymbol{R}_{\text{LQR}}$ are chosen to be diagonal matrices and the weights are chosen experimentally in order to get a fast and accurate response. The optimal gain matrix $\boldsymbol{K}$ is calculated numerically using the built-in Matlab function `lqr(A,B,Q,R)`, and the structure is seen in the following equation:

$$\boldsymbol{u} = \begin{bmatrix} \tilde{V}_s \\ V_d \end{bmatrix} = \begin{bmatrix} F_{11} & F_{21} \\ F_{12} & F_{22} \end{bmatrix} \begin{bmatrix} p_c \\ \dot{e}_c \end{bmatrix} - \begin{bmatrix} 0 & 0 & k_1 \\ k_2 & k_3 & 0 \end{bmatrix} \begin{bmatrix} p \\ \dot{p} \\ \dot{e} \end{bmatrix} \quad . \tag{36}$$

By observing that in our linearized model, the pitch of the helicopter does not depend on the elevation of the helicopter and vice versa, we can conclude that $F_{11}, F_{22} = 0$.

For fixed values of references $p_c$ and $\dot{e_c}$, we want to have a system where $\lim_{t\to\infty} \boldsymbol{y} = \boldsymbol{r}$. As $t \to \infty$ implies that there is no more transients in the system, we use eq. (33) to find an expression for $\boldsymbol{F}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{F}\boldsymbol{r} - \boldsymbol{K}\boldsymbol{x}) \tag{37}$$

$$\boldsymbol{0} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x} + \boldsymbol{B}\boldsymbol{F}\boldsymbol{r}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} = \boldsymbol{r} = \boldsymbol{C}(\boldsymbol{B}\boldsymbol{K} - \boldsymbol{A})^{-1}\boldsymbol{B}\boldsymbol{F}\boldsymbol{r} \quad , \tag{38}$$

such that

$$\boldsymbol{F} = [\boldsymbol{C}(\boldsymbol{B}\boldsymbol{K} - \boldsymbol{A})^{-1}\boldsymbol{B}]^{-1} \quad . \tag{39}$$

```
A = [0   1   0   ;
     0   0   0   ;
     0   0   0] ;
C = [1   0   0   ;
     0   0   1] ;
B = [0       0   ;
     0       k_1;
     k_2     0 ];
Q = diag([50,   10, 55]);
R = diag([1.5, 0.5]);
K = lqr(A, B, Q, R);
F = inv(C*(inv(B*K-A))*B);
```

The entries of the $\boldsymbol{F}$ matrix can be found directly through calculating the inverse in Matlab as shown above, as the matrix $\boldsymbol{C}(\boldsymbol{BK}-\boldsymbol{A})^{-1}\boldsymbol{B}$ is invertible. $\boldsymbol{F}$ may also be found through algebraic calculations. Inserting eq. (31) into eq. (36) with $\boldsymbol{u}=\boldsymbol{0}$ and $\boldsymbol{x}=[p_c, 0, \dot{e_c}]^T$ gives the equations

$$0 = p_c(F_{12} - k_1) \tag{40a}$$
$$0 = \dot{e_c}(F_{21} - k_3) \quad , \tag{40b}$$

which then gives us the matrix

$$\boldsymbol{F} = \begin{bmatrix} 0 & k_1 \\ k_2 & 0 \end{bmatrix} \quad . \tag{41}$$

The elements of $\boldsymbol{F}$ depend on the weights in matrices $\boldsymbol{Q}_{\text{LQR}}$ and $\boldsymbol{R}_{\text{LQR}}$. The entries of the $\boldsymbol{F}$ matrix optimizes the reference feed forward, so that the LQR ensures that $\boldsymbol{y} \to \boldsymbol{r}$ as $t \to \infty$.

The experimenting is done by separating the parameters which affect the responses in elevation rate $\dot{e}$ and pitch $p$. Some different values are shown in tables 4 and 5, respectfully. Note that the column "Tag" in table 5 refers to the legend of fig. 13.

We first focused on optimizing the response in elevation rate. By using the Joystick as a reference for the elevation rate $\dot{e}$, we observed different responses according to the values of $q_3$ and $r_1$, some of which are shown in table 4. The measured $\dot{e}$ from the encoder versus the joystick reference is seen in fig. 11, with the LQR parameters from $\text{Q}_{\text{LQR1}}$. Note that the helicopter is started from ground level with a offset voltage of $V_{s,0}$, leading to a spike in the elevation rate error before $t = 4$. We found that $q_3 = 55$ was the optimal value for elevation rate tuning, as a lower value made the helicopter track the reference badly, and a higher value made the helicopter oscillate. With $r_1 = 1.4$, the response was quick, while at the same time slightly penalizing higher values of $\tilde{V_s}$, resulting in smooth and controlled movement.

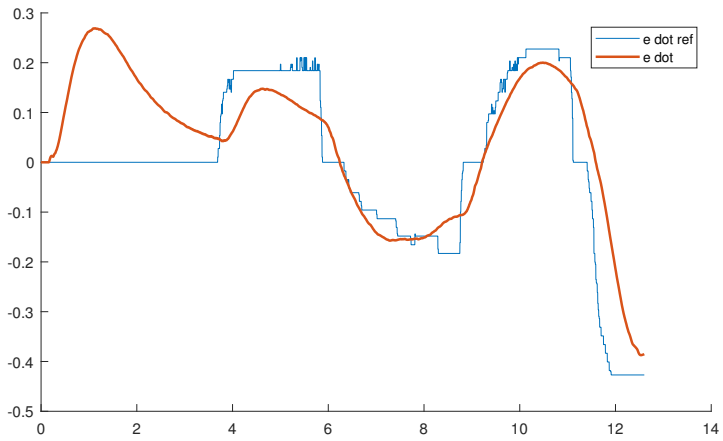



Figure 11: Elevation rate response with Joystick input, problem 2.

After the controller was tuned for elevation response, we focused on the pitch angle. By changing the weights of $q_1$, $q_2$ and $r_2$, we penalize the errors in both pitch angle $p$ and pitch rate $\dot{p}$, as well as the voltage reference between the two motors. For tracking the pitch response, we used a setup in Simulink containing 2 step functions. At $t = 0$ the helicopter starts at ground level and is then set at the linearization point, where $e = 0$. At $t = 5$s, a reference angle of $p = 0.5$rad is set. At $t = 9$s the reference angle is reset to $p = 0$. With this setup we were easily able to identify the behaviour of the helicopter. The Simulink implementation is shown in fig. 12.

As shown in fig. 13, a slow response is acquired in both cases $\mathrm{Q}_{\mathrm{LQR2}}$ and $\mathrm{Q}_{\mathrm{LQR3}}$: Penalizing $V_s$ too much and penalizing errors $p - p_c$ and $\dot{p} - \dot{p}_c$ too little. An oscillating response due to very large values of $\boldsymbol{u}$ is also shown in the figure, a result of penalizing $V_s$ too little. The most optimal response found had a balanced input and a fast, smooth response, and is given by $\mathrm{Q}_{\mathrm{LQR1}}$. Albeit the $\boldsymbol{K}$ matrix ensures that the error will be minimized over time, an error is still expected to be somewhat prevalent in a controller with no integral effect.

### 3.3 Problem 3 - LQR with integral effect

Now the controller from Section 3.2 are modified to include an integral effect to quickly remove any error between the reference and $\boldsymbol{y}$. We introduce two new states,

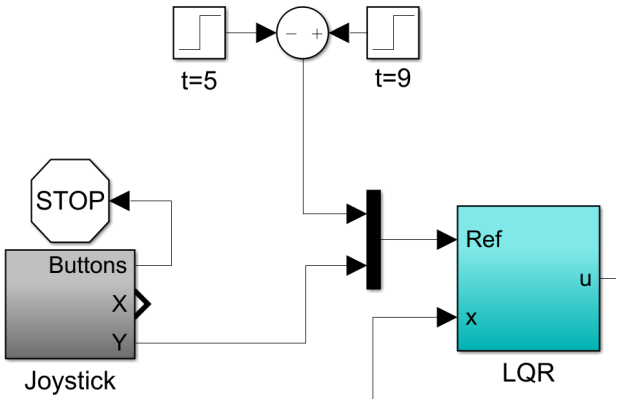



Figure 12: Simulink setup for pitch tuning.

Table 4: LQR tuning by elevation rate, Problem 2.

| $q_1$ | $q_2$ | $q_3$ | $r_1$ | $r_2$ | Tag |
|---|---|---|---|---|---|
| 50 | 10 | 55 | 1.5 | 0.5 | $\mathrm{Q}_{\mathrm{LQR1}}$ |
| 50 | 10 | 100 | 10 | 0.5 | $\mathrm{Q}_{\mathrm{LQR2}}$ |
| 50 | 10 | 200 | 0.5 | 0.5 | $\mathrm{Q}_{\mathrm{LQR3}}$ |

Table 5: LQR tuning by pitch angle and rate, Problem 2.

| $q_1$ | $q_2$ | $q_3$ | $r_1$ | $r_2$ | Tag |
|---|---|---|---|---|---|
| 50 | 10 | 55 | 1.5 | 0.5 | $\mathrm{Q}_{\mathrm{LQR1}}$ |
| 50 | 20 | 55 | 1.4 | 5 | $\mathrm{Q}_{\mathrm{LQR2}}$ |
| 50 | 10 | 55 | 1.4 | 0.1 | $\mathrm{Q}_{\mathrm{LQR3}}$ |
| 5 | 5 | 55 | 1.4 | 0.5 | $\mathrm{Q}_{\mathrm{LQR4}}$ |

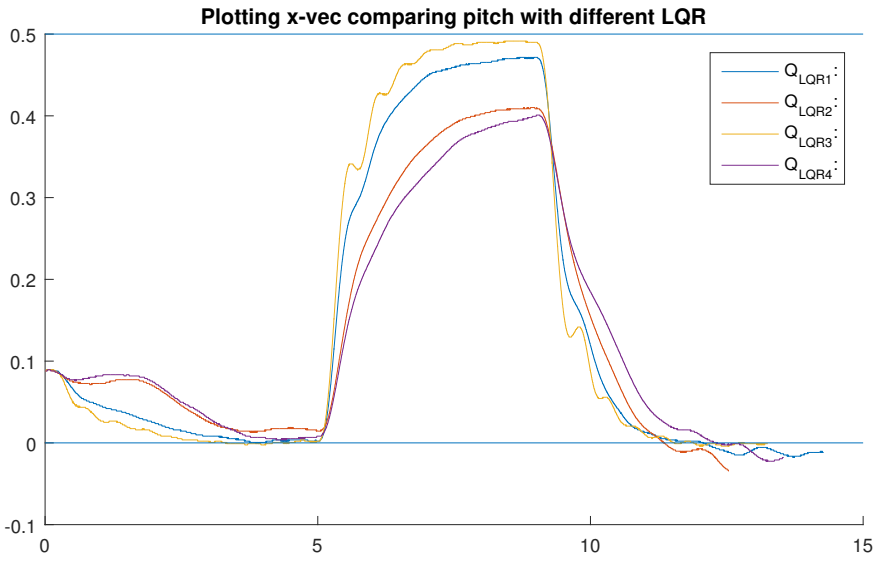



Figure 13: Pitch response with LQR from Problem 2

$\gamma$ and $\zeta$, with differential equations given by

$$\dot{\boldsymbol{x}}_a = \begin{bmatrix} \dot{\gamma} \\ \dot{\zeta} \end{bmatrix} = \begin{bmatrix} p - p_c \\ \dot{e} - \dot{e}_c \end{bmatrix} \quad . \tag{42}$$

These states represent the errors of pitch angle and elevation rate from the reference input. By penalizing these states in an LQR model, we can choose the rate of which the errors reduce. The helicopter will then quickly approach the equilibrium point where our linearized model holds.

We will now use an augmented state space model which includes the new states $\gamma$ and $\zeta$. Using eqs. (31) and (42), our augmented state space model is now given by

$$\dot{\bar{\boldsymbol{x}}} = \bar{\boldsymbol{A}}\bar{\boldsymbol{x}} + \bar{\boldsymbol{B}}\boldsymbol{u} + \boldsymbol{P}\boldsymbol{r} \tag{43a}$$

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{x_a}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_a} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{bmatrix} \boldsymbol{r} \quad , \tag{43b}$$

where $\bar{\boldsymbol{x}}$ is the new state vector and $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are the new system matrices:

$$\bar{\boldsymbol{A}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad , \quad \bar{\boldsymbol{B}} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad . \tag{44}$$

Following the introduction of the two new states, the cost matrix $\boldsymbol{Q}$ is expanded to include the cost of the errors:

$$\bar{\boldsymbol{Q}} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q_a} \end{bmatrix} \quad \text{, where} \quad \boldsymbol{Q_a} = \begin{bmatrix} q_4 & 0 \\ 0 & q_5 \end{bmatrix} \tag{45}$$

and $q_4$ and $q_5$ will tune the corresponding states, that is $\gamma$ and $\zeta$, respectively. The gain matrix $\bar{\boldsymbol{K}}$ can again be calculated using the `lqr` command from section 3.2 in Matlab, yielding

$$\bar{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K_a} \end{bmatrix} = \begin{bmatrix} 0 & 0 & k_1 & 0 & k_4 \\ k_2 & k_3 & 0 & k_5 & 0 \end{bmatrix} \quad . \tag{46}$$

The controller output is now described as

$$\bar{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K_a} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{x_a} \end{bmatrix} + \bar{\boldsymbol{F}}\boldsymbol{r} \quad . \tag{47}$$

As the new values for $q_4$ and $q_5$ will also alter the response in pitch and elevation rate, we needed to tune both $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$. The new parameters represents to what extent the controller will penalize the accumulated errors $\int p - p_c \; dt$ and $\int \dot{e} - \dot{e}_c \; dt$.

We found that the helicopter responded more precisely as we increased the weights of $q_4$ and $q_5$, in particular the latter. Increasing this made the helicopter follow the joystick reference very accurately, so that if the joystick was released to neutral position, the helicopter would stop almost immediately. Setting this value to $< 1$ resulted in the helicopter going straight to the linearization point, despite the reference elevation rate being zero. We also increased the $q_3$ parameter slightly as to oppose the overshoot which resulted from penalizing the elevation rate error too little.

The tuning variable corresponding to the state $\gamma$ was decided through experimentation with the same set up as before, shown in fig. 12. Using values of $q_4 < 50$ gave little to no contribution to the pitch response, both from observing the helicopter visually as well as scoping the pitch output. Increasing it further than this did however result in a faster correction in pitch angle. We found that using a magnitude of 100 gave a satisfying response, in combination with using $r_2 = 1$, penalizing the voltage difference between the motors more than before.

As before, the matrix $\bar{\boldsymbol{F}}$ from eq. (47) is optimizing the feed forward gain as described in section 3.2. The matrix ensures that the stationary error is minimized, however, with the introduction of integral effect, this property becomes obsolete. Our LQR is designed such that we can choose the rate of which our errors decline, so that the $\bar{\boldsymbol{F}}$ works as a simple gain matrix. Although different matrices results in different behaviour, we can choose this matrix freely (within sensible boundaries) without it affecting the stationary error. As the new matrix $[\bar{\boldsymbol{C}}(\bar{\boldsymbol{B}}\bar{\boldsymbol{K}} - \bar{\boldsymbol{A}})^{-1}\bar{\boldsymbol{B}}]$ is no longer invertible with the new matrix dimensions, we may compute this algebraically

through eq. (47). However, due to the integral effect, we simply choose

$$\bar{\boldsymbol{F}} = \boldsymbol{F} = \begin{bmatrix} 0 & k_1 \\ k_2 & 0 \end{bmatrix} \quad . \tag{48}$$

The final matrices used for our 5-state LQR are

$$\bar{\boldsymbol{Q}} = \begin{bmatrix} 25 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 \\ 0 & 0 & 75 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix} \quad \text{and } \bar{\boldsymbol{R}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad . \tag{49}$$

Using the values for $\bar{\boldsymbol{Q}}$ and $\bar{\boldsymbol{R}}$, we end up with the gain matrix

$$\bar{\boldsymbol{K}} = \begin{bmatrix} 0 & 0 & 17.34 & 0 & 10.00 \\ 14.36 & 9.07 & 0 & 10.0 & 0 \end{bmatrix} \quad . \tag{50}$$

The fig. 14 shows the response of our system using different $\boldsymbol{F}$ matrices with the pitch-testing setup used in section 3.2. $F_0$ represents eq. (48), and

$$F_1 = \begin{bmatrix} 0 & 70 \\ 60 & 0 \end{bmatrix} \quad , \quad F_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad .$$

The first resulted, as predicted, in an unstable system. The last one is very slow, but the stationary error tends to 0, due to integral action.

The Simulink implementation of our plant with a 5-state LQR is shown in fig. 16. The LQR itself is seen in fig. 15.

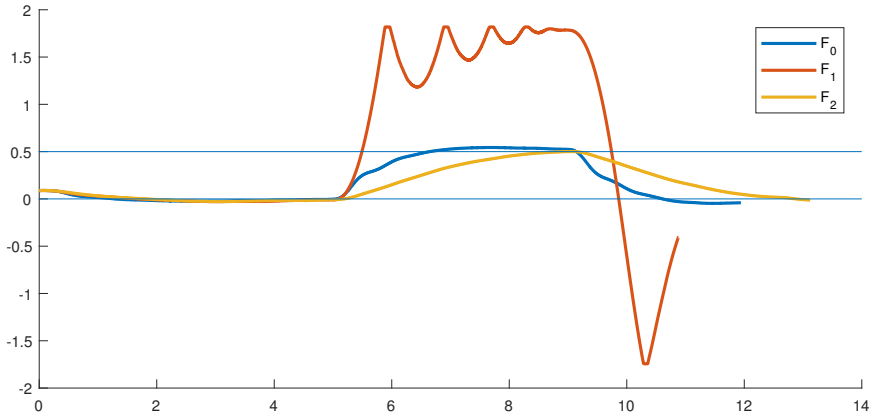



Figure 14: Pitch response with references 0.5 and 0 with different F matrices.

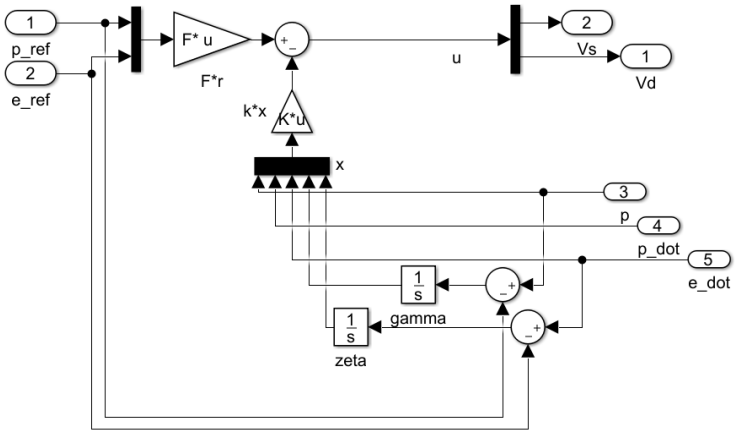



Figure 15: 5-state LQR implementation in Simulink.

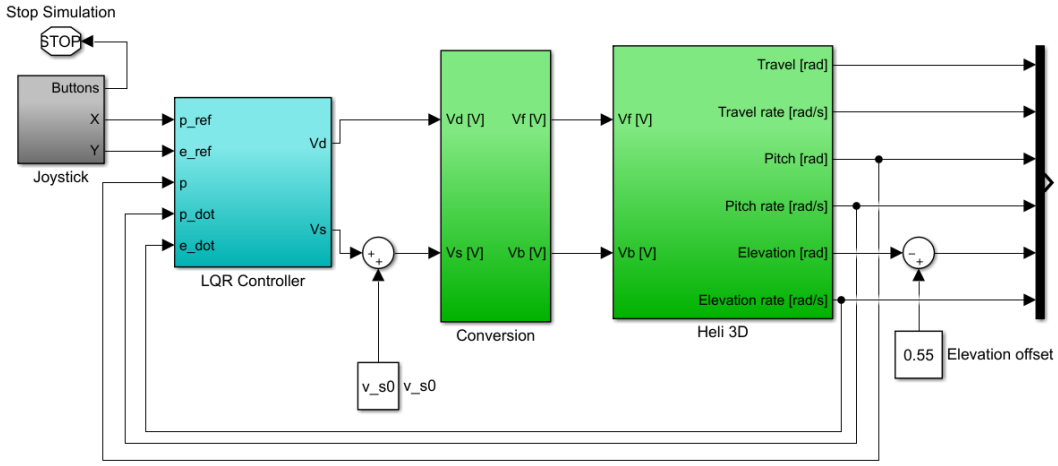



Figure 16: Simulink implementation with LQR.

# 4 Part IV - Inertial Measurement Unit

An Inertial Measurement Unit, IMU for short, is an extremely practical device when it comes to measuring the internal states of a dynamic systems such as a helicopter. In parts IV, V and VI, we will be using an IMU equipped with a gyroscope, an accelerometer and a compass.

## 4.1 Problem 1 - IMU Characteristics

When holding the helicopter at the linearization point and rotating it about one axis at the time, the gyroscope output is almost the same as the encoder-rates. By first looking at fig. 23 and fig. 24 which is the comparison of the elevation rate outputs and pitch rate outputs respectively, you can see that the gyro and the encoder is more or less the same, but the gyro has an noticeably amount of noise. Moving over to travel, fig. 25, including the noise in the gyro output we also have a slight offset.

Now if the helicopter is held horizontal and we continuously move the helicopter up and down i.e changing the elevation constantly, we can see in fig. 26 that the travel rate from the encoder is stationary, while the elevation rate from the IMU goes up and down. Now if we rotate the helicopter about 90° around the x-axis thus changing the pitch, and move the helicopter left and right we can see in fig. 26 that after about 6 seconds(when we alter the pitch) the gyro elevation rate output follows the encoder travel rate instead of the elevation rate. The reason for this is that the IMU is fixed to the pitch arm of the helicopter which leads to the x-, and y-axes are changed by the pitch angle of the helicopter.

If we do the same movement with the helicopter and look at the accelerometer output fig. 27, we can first off see that when we start with the helicopter on the ground(from t=0 to t=2), that $a_z$ is slightly less than $g = -9.81 m/s^2$ and that $a_x \neq 0$. This is because the helicopter's x-direction is not horizontal, therefore it gets affected by the gravitational pull. From $t$=2 to $t$=6 we keep the pitch flat, and move the helicopter up and down around the linearization point. Here $a_x$ oscillates around 0 for the same reason as above - it increases with the elevation angle. The $a_z$ output has bigger oscillations as we would expect, and has its peaks when we change direction from up to down or vice versa. $a_y$ has next to none changes in this particular part of the sequence, the changes you see is due to noise and that we are not able to keep the pitch perfectly at zero. The next step in our sequence is from about $t$=6 to $t$=11 (see the stapled lines). Here we rotate the pitch 90° and then move it back and forth in the travel direction, and try ti keep the elevation at zero. Now we can see that as $a_y$ is at this moment pointing downwards it has the acceleration equal to $g$, and that the oscillations of $a_z$ corresponds to the changes in travel, as it is pointing in the horizontal direction. The last part of the sequence, as $t$>11, we flip the pitch back to zero while still changing the travel, and we now see that $a_y$ and $a_z$ switch to the others behaviour from the last sequence. This makes sense now that $a_z$ now points down and $a_y$ points horizontal.

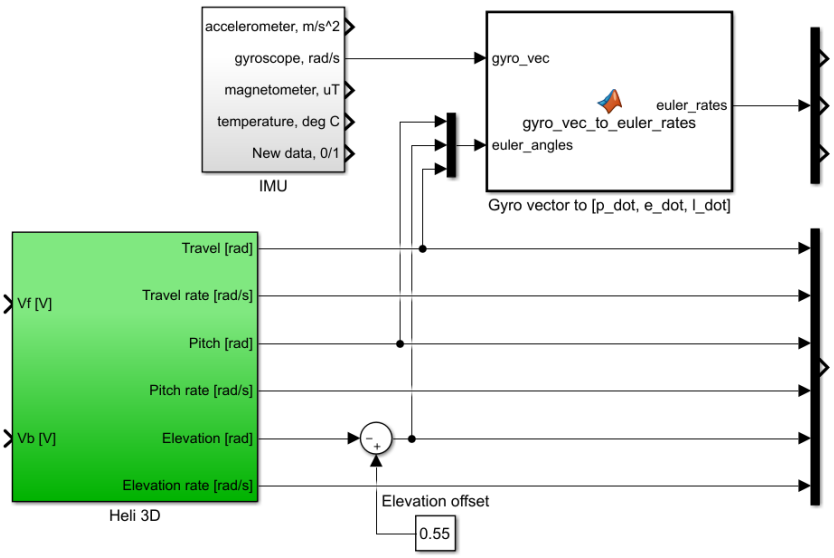



Figure 17: A screengrab of the setup with the transformation block "gyro_vec_to_euler_rates".

## 4.2 Problem 2 - Gyroscope transfrom

As discussed in section 4.1, we cannot use the gyro measurements directly due to the physical placement of the IMU in our setup. However we can fix this by using the pitch and elevation angles from the encoder to calculate the correct pitch-, elevation-, and travel-rate. This conversion is done in a Simulink block provided to us by the course staff. Our setup with the new Simulink block is shown in fig. 17. The block simply takes the pitch, elevation and travel from the encoder as well as the pitch-, elevation- and travel-rate from the gyroscope as input. The function compares the values and performs trigonometric operations and outputs the correct pitch-, elevation- and travel-rate.

## 4.3 Problem 3 - Observability

With our transformed gyro measurements we now have the pitch-, elevation- and travel-rate vector

$$\mathbf{y}_{gyro} = \begin{bmatrix} \dot{p} \\ \dot{e} \\ \dot{\lambda} \end{bmatrix} \tag{51}$$

We define a state- and input vector:

$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \\ e \\ \dot{e} \\ \lambda \\ \dot{\lambda} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} V_s \\ V_d \end{bmatrix} \tag{52}$$

As shown in eq. (9) - eq. (14) we have

$$\ddot{p} = K_1 V_d, \ddot{e} = K_2 \tilde{V_s} \tag{53}$$

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{54}$$

$$\mathbf{y}_{gyro} = \mathbf{C}_{gyro}\mathbf{x} \tag{55}$$

with eq. (53), we get our state matrix $\mathbf{A}$, input matrix $\mathbf{B}$ and output matrix $\mathbf{C}$:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \\ \dot{\lambda} \\ \ddot{\lambda} \end{bmatrix} = \begin{bmatrix} x_2 \\ K_1 u_2 \\ x_4 \\ K_2 u_1 \\ x_6 \\ K_3 x_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ K_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \mathbf{x} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{B}} \mathbf{u} \tag{56}$$

$$\mathbf{y}_{gyro} = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}_{gyro}} \mathbf{x} \tag{57}$$

Now if we run the matlab script:

```
A = [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];

C_gyro = [0   1   0   0   0   0;
          0   0   0   1   0   0;
          0   0   0   0   0   1];

O_gyro = obsv(A,C_gyro);
```

and then take a look at our new variable $O_{gyro}$, generated in matlab:

$$O_{gyro} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.6117 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.6117 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{58}$$

we can quickly see that the full state is not observable, because both $e$ and $\lambda$ are not linearly independent. The reason for $p$ being observable and not $e$ and $\lambda$, even though p is not an output from the gyro, is because by looking at eq. (13c) we can see that by differentiating $\dot{\lambda}$, which is observable from the gyro output, we can derive $p$.

### 4.4 Problem 4 - Accelerometer

When the helicopter is stationary, the accelerometer on the IMU measures the force counteracting gravity. This force will always point straight upwards with a magnitude equal to the gravitational constant $g$.

We want to derive equations for the measured acceleration $[a_x, a_y, a_z]$ from the IMU, given the elevation and pitch angle when the helicopter is stationary. First off we derive them only by looking at changes in elevation, and having the pitch set to zero.

By using fig. 18 and basic trigonometry we get our three equations (59) for measured acceleration(when pitch is zero). We assume that all accelerations are linear, $a_y$ is zero when the pitch is zero, and that $a_y$ is unaffected by changes in elevation.

$$a_{x,p=0} = g \sin e \tag{59a}$$

$$a_{y,p=0} = 0 \tag{59b}$$

$$a_{z,p=0} = -g \cos e \tag{59c}$$

Now we can derive the measured acceleration when $p \neq 0$. We now assume that the elevation is zero which implies that the helicopter is in the linearization point, moreover we assume that $a_x$ is unaffected by changes in pitch and is zero when the elevation is zero, hence

$$a_{x,p\neq 0} = 0 \tag{60a}$$
$$a_x = a_{x,p=0} = g \sin e \tag{60b}$$

In fig. 19 we can use the fact that $g* = a_{z,p=0}$ which we have in (59c), to derive both $a_{z,p\neq 0}$ and $a_{y,p\neq 0}$

$$a_{y,p\neq 0} = -g \sin p \cos e \tag{61a}$$
$$a_{z,p\neq 0} = -g \cos e \cos p \tag{61b}$$

If we now use Pythagoras theorem with fig. 18, fig. 19, eq. (60b) and eq. (61) we can derive $e$:

$$\begin{aligned} a_{z,p=0}^2 &= a_{z,p\neq 0}^2 + a_{y,p\neq 0}^2 \\ a_{z,p=0} &= \sqrt{a_{z,p\neq 0}^2 + a_{y,p\neq 0}^2} \\ \cos e \cdot g &= \sqrt{a_{z,p\neq 0}^2 + a_{y,p\neq 0}^2} \\ \frac{\cos e}{\sin e} a_x &= \sqrt{a_{z,p\neq 0}^2 + a_{y,p\neq 0}^2} \\ e &= \arctan \frac{a_x}{\sqrt{a_z^2 + a_y^2}} \end{aligned} \tag{62}$$

By solving eq. (61a) and eq. (61b) for $g$ we can derive the pitch:

$$\begin{aligned} a_{y,p\neq 0} &= -g \sin p \cos e \\ -g &= \frac{a_{y,p\neq 0}}{\sin p \cos e} \end{aligned} \tag{63a}$$

$$\begin{aligned} a_{z,p\neq 0} &= -g \cos e \cos p \\ -g &= \frac{a_{z,p\neq 0}}{\cos e \cos p} \end{aligned} \tag{63b}$$

$$\begin{aligned} g &= g \\ \frac{a_{y,p\neq 0}}{\sin p \cancel{\cos e}} &= \frac{a_{z,p\neq 0}}{\cancel{\cos e} \cos p} \\ \frac{\sin p}{\cos p} &= \frac{a_{y,p\neq 0}}{a_{z,p\neq 0}} \\ p &= \arctan \frac{a_y}{a_z} \end{aligned} \tag{63c}$$

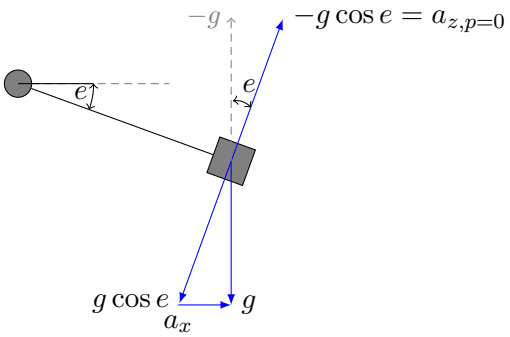



Figure 18: Side view of helicopter

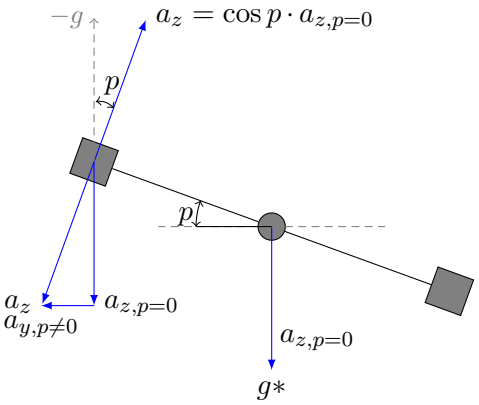



Figure 19: Front view of system

To sum up our assumptions we used to get our $e$ and $p$; firstly we assumed that all accelerations are linear. We then started of by looking at the system from the side with the pitch set to zero fig. 18, and then looked at the system from the front with elevation set to zero fig. 19. By this procedure we can easily decompose all necessary angles and forces that was initially in three dimensions.

### 4.5 Problem 5 - Observability Part II

Now that we have our two additional measurements $p$ and $e$, our output vector looks like this:

$$\mathbf{y}_{IMU} = \begin{bmatrix} \dot{p} \\ \dot{e} \\ \dot{\lambda} \\ p \\ e \end{bmatrix} \tag{64}$$

Our output vector has changed, which gives us a new output matrix $\mathbf{C}$:

$$\mathbf{y}_{IMU} = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \\ x_1 \\ x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{C}_{IMU}} \mathbf{x} \tag{65}$$

We now run the matlab function *obsv* with our new output matrix:

```
A = [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];

C_IMU = [0   1   0   0   0   0;
         0   0   0   1   0   0;
         0   0   0   0   0   1;
         1   0   0   0   0   0;
         0   0   1   0   0   0];
O_IMU = obsv(A,C_IMU);
```

The new observability matrix $O_{IMU}$ now has one more observable state, $e$, because $e$ is now a part of measurement and output vector $y_{IMU}$.

### 4.6 Problem 6 - Noise

We have produced an experimental time-series when the helicopter is laying still on the ground as well as a time-series when the helicopter is kept stationary at the linearization point while flying in fig. 20 and fig. 21, respectively. It is clear that there is noise in both cases, but if we compare the plots, the noise while the helicopter is flying is almost 100 times higher. There are many reasons for the noise, firstly when the helicopter is standing still on the ground there are no vibrations from the motors and no small changes in any of the rotations. When the helicopter is flying there will obviously appear vibrations, as well as the system is not perfectly still in the air, and this will affect the measurements. If we want to classify the type of noise we get, we can note that the noise is bandlimited and has nearly zero mean. We consider the noise in both cases as white noise, more specific a white sequence which is defined simply as a sequence of zero-mean, uncorrelated random variables (p.77, [1]). One can see that there is a slight bias, but when we take this bias to consideration we get zero mean noise value i.e. a white sequence. The covariance

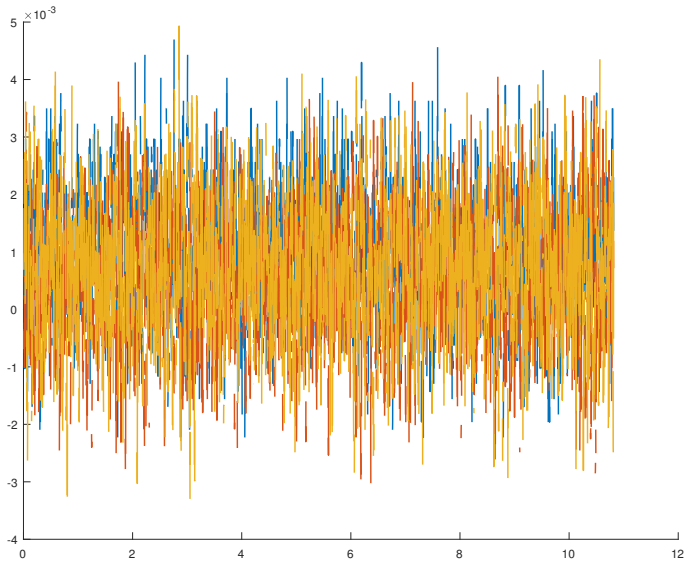



Figure 20: $y_{IMU}$ when stationary on the ground.

in both cases are - as expected different; where the covariance matrix when flying at the linearization point has much larger diagonal values than the covariance matrix when the helicopter is laying on the ground eq. (66). The covariance tells us the variance in our measurements, which fits well with both our plots and assumptions.

$$\underbrace{\begin{bmatrix} 0.0179 & - & - & - & - \\ - & 0.0090 & - & - & - \\ - & - & 0.0170 & - & - \\ - & - & - & 0.0134 & - \\ - & - & - & - & 0.0466 \end{bmatrix}}_{\mathbf{Cov}_{air}} \underbrace{\begin{bmatrix} 4.5e^{-6} & - & - & - & - \\ - & 6.8e^{-7} & - & - & - \\ - & - & 4.5e^{-6} & - & - \\ - & - & - & 2e^{-6} & - \\ - & - & - & - & 2.9e^{-6} \end{bmatrix}}_{\mathbf{Cov}_{Ground}} \tag{66}$$

Now that we measure all relevant states for our controllers and try to fly it only by using the measurements as feedback it flies very poorly as you can see in fig. 22. This does not come as a surprise due to all the noise in the measurements that goes unfiltered into our controller.

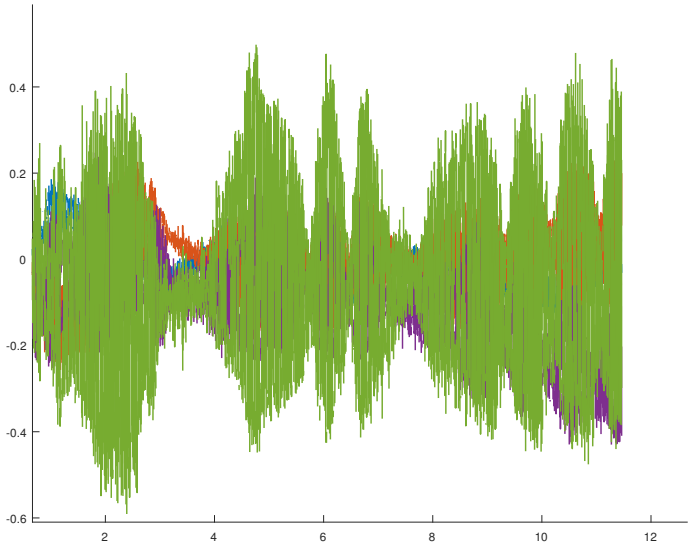



Figure 21: $y_{IMU}$ when flying in linearization point.

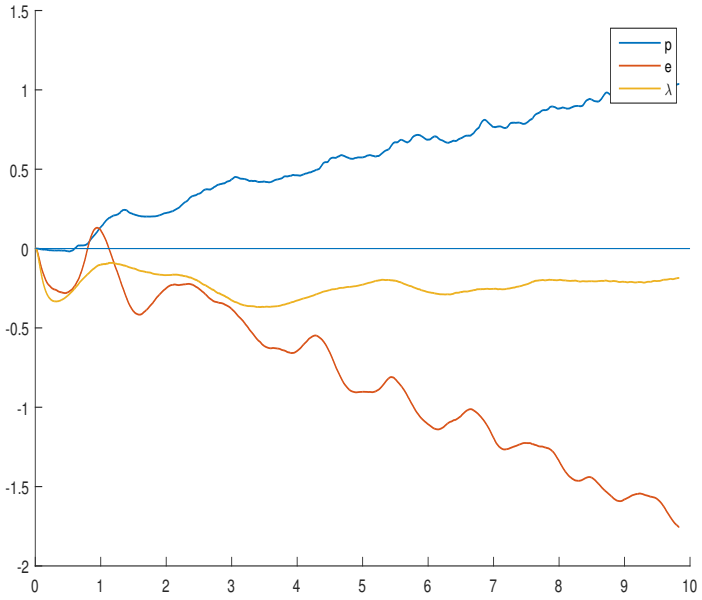



Figure 22: Flying the helicopter using only the measurements as feedback.

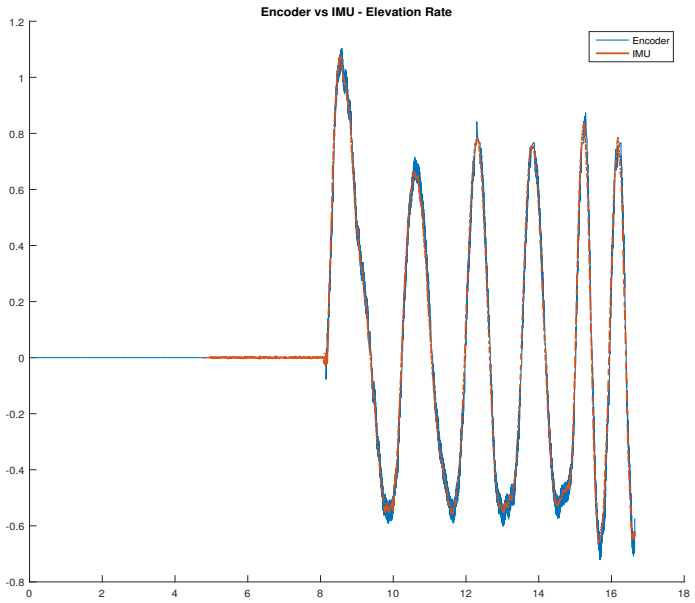



Figure 23: Gyroscope and encoder elevation rate comparison

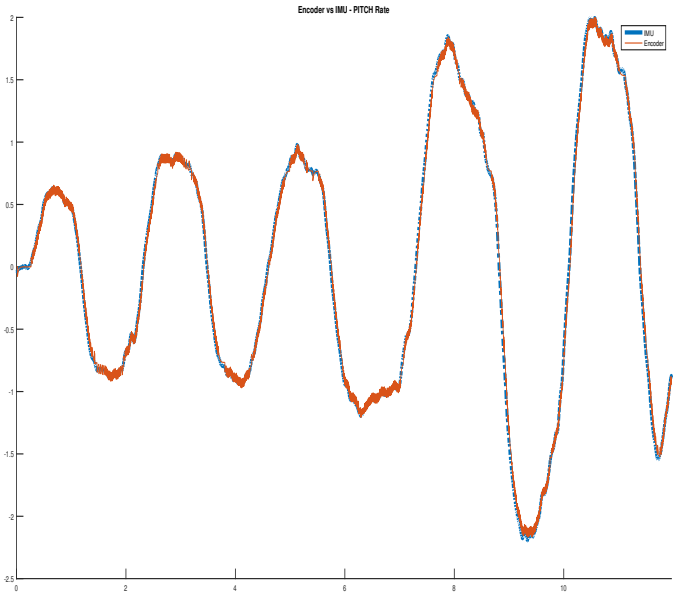



Figure 24: Gyroscope and encoder pitch rate comparison

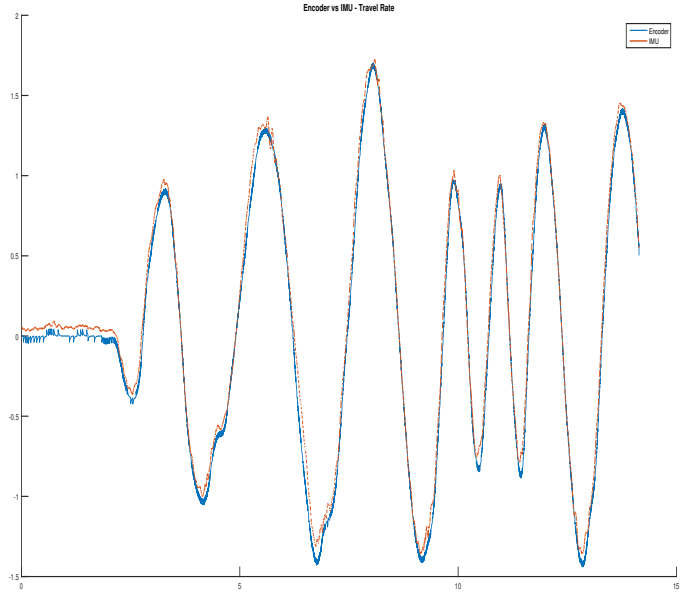



Figure 25: Gyroscope and encoder travel rate comparison

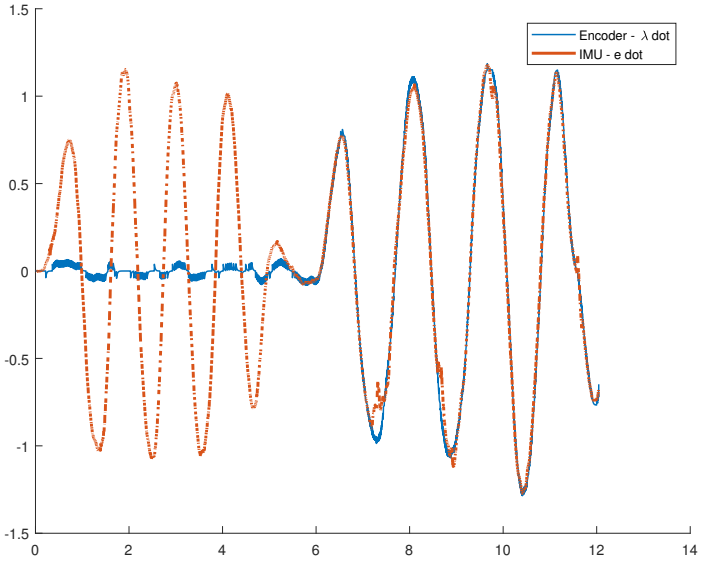



Figure 26: Gyroscope $\dot{e}$ and encoder $\dot{\lambda}$ comparison when rotated about two axes.

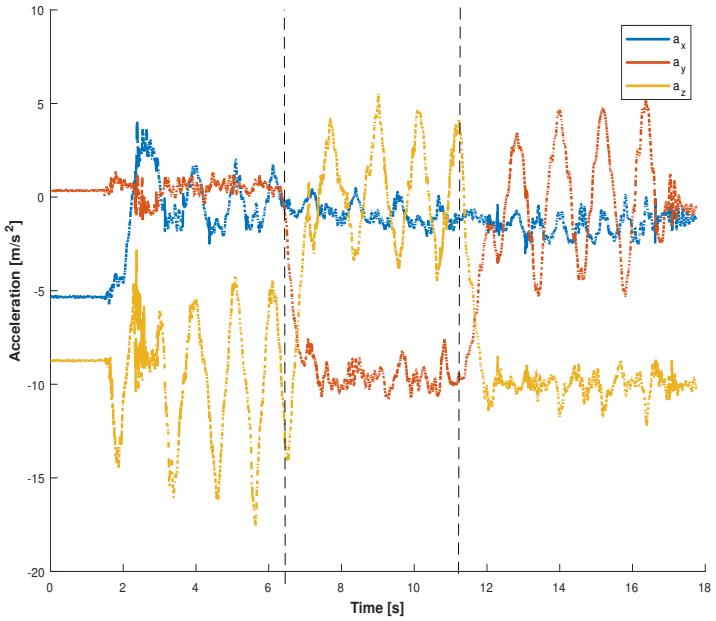



Figure 27: Accelerometer output when first moving in elevation, then rotating the pitch 90°, then move back and forth in travel direction.

## Part V & VI: The Kalman Filter

We are now going to replace the unit from were we get our output feedback from the encoders to the IMU. However, it is not sufficient enough to just feed the measurement from the IMU directly back to the controller, as in the case of the encoders. The solution is to develop a estimator and using that in combination with our model of the system to get a better estimate for the state. This again will allow us to integrate a Kalman Filter (KF). The KF is described by following equations:

Correction with new data:

$$\boldsymbol{K}[k] = \bar{\boldsymbol{P}}\boldsymbol{C}_d^\top(\boldsymbol{C}_d\bar{\boldsymbol{P}}\boldsymbol{C}_d^\top + \boldsymbol{R}_d)^{-1} \tag{67a}$$

$$\hat{\boldsymbol{x}}[k] = \bar{\boldsymbol{x}} + \boldsymbol{K}(\boldsymbol{y} - \boldsymbol{C}_d\bar{\boldsymbol{x}}) \tag{67b}$$

$$\hat{\boldsymbol{P}}[k] = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)\bar{\boldsymbol{P}}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)^\top + \boldsymbol{K}\boldsymbol{R}_d\boldsymbol{K}^\top \quad . \tag{67c}$$

Predicting ahead:

$$\bar{\boldsymbol{x}}k+1] = \boldsymbol{A}_d\hat{\boldsymbol{x}}[k] + \boldsymbol{B}_d\boldsymbol{u}[k] \tag{68a}$$

$$\bar{\boldsymbol{P}}[k+1] = \boldsymbol{A}_d\hat{\boldsymbol{P}}[k]\boldsymbol{A}_d^\top + \boldsymbol{Q}_d \tag{68b}$$

# 5 Part V - Prediction Step

Since the IMU gives the data in discrete-time, a stochastic process model in discrete-time is necessary. Following model equations is therefore introduced:

$$\boldsymbol{x}[k+1] = \boldsymbol{A}_d\boldsymbol{x}[k] + \boldsymbol{B}_d\boldsymbol{u}[k] + \boldsymbol{w}_d[k] \tag{69a}$$

$$\boldsymbol{y}[k] = \boldsymbol{C}_d\boldsymbol{x}[k] + \boldsymbol{v}_d[k] \tag{69b}$$

$$\boldsymbol{w}_d \sim N(\boldsymbol{0}, \boldsymbol{Q}_d), \boldsymbol{v}_d \sim N(\boldsymbol{0}, \boldsymbol{R}_d) \tag{69c}$$

## 5.1 Problem 1 - Discretization

In order to dicretize the model we already have of our system, we can utilize the Matlab-function c2d. The sampling time that is given is 0.002s. As the observability of the system is unchanged i.e. $\boldsymbol{C} = \boldsymbol{C_d}.$, we only need to discretize $\boldsymbol{A}$ and $\boldsymbol{B}$. The discretized matrices was found to be

$$\boldsymbol{A_d} = \begin{bmatrix} 1 & 0.0020 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0020 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -0.0000 & -0.0000 & 0 & 0 & 1 & 0.0020 \\ -0.0012 & -0.0000 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{70}$$

$$\boldsymbol{B_d} = \begin{bmatrix} 0 & 0.000 \\ 0 & 0.0011 \\ 0.0000 & 0 \\ 0.0002 & 0 \\ 0 & -0.0000 \\ 0 & -0.0000 \end{bmatrix}, \tag{71}$$

$$\boldsymbol{C_d} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{72}$$

The Matlab code for this is as follows:

```
sys = ss(A, B, C, D);
sys_d = c2d(sys,0.002);
A_d = sys_d.A;
B_d = sys_d.B;
C_d = C
```

In our Kalman Filter we have two matrices that we are able to alter in order to get the performance from our system as we would like; $\boldsymbol{R_d}$ and $\boldsymbol{Q_d}$. $\boldsymbol{Q_d}$ is the covariance of the stochastic disturbance to the system, while $\boldsymbol{R_d}$ corresponds to the discrete-time covariance from the measurement. We use the values obtained by eq. (66) as $\boldsymbol{R_d}$.

## 5.2 Problem 2 - State Prediction

If we utilize the previous predicted state as the current state estimation, we are able implement a state prediction step. This can recursively be done in a loop, and result in an open loop estimation that could be expressed as:

$$\bar{\boldsymbol{x}}[k+1] = \boldsymbol{A}_d\bar{\boldsymbol{x}}[k] + \boldsymbol{B}_d\boldsymbol{u}[k] \tag{73}$$

This was implemented in Matlab (fig. 28) where we would get the predicted next state as output and the control signal ($u[k]$), current state prediction value ($\bar{x}[k]$)

and the matrices $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$. A unit delay block is added to separate time steps $k$ and $k+1$, while also supplying the open loop estimator with the initial condition [0,0,-0.49,0,0,0], compensating for the helicopter being started from ground level.

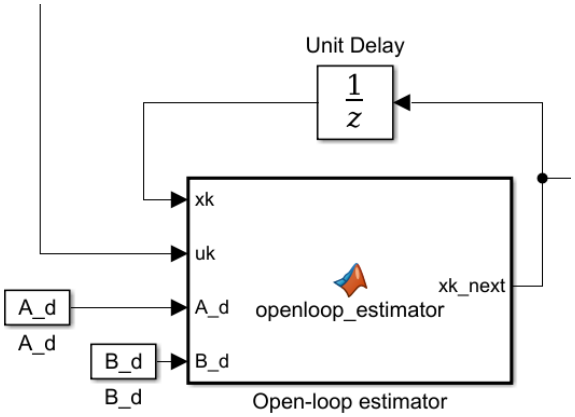



Figure 28: Matlab-function block for the open-loop estimator

```
function xk_next = openloop_estimator(xk, uk, A_d, B_d)
xk_next = A_d*xk + B_d*uk;
```

When comparing our open-loop estimator values against the experimental values obtained from the encoders, we can quickly see that there are huge deviations between the two. Since the open loop estimator only bases the estimates on the control signal and previous states, the variables in the system unsurprisingly diverge.

As done in section 1.4, we tested out two scenarios, where one was launched around the linearization point and one outside the linearization point of pitch, elevation and travel (fig. 30). fig. 29a shows that the helicopter is able to preform relatively good and stay in the equilibrium point for about 13 seconds before it "crashes". The scenario when we launched outside of the linearization is not as fortunate; Just using an open loop estimator that bases it's next state on the previous state and the control signal, will lead to the same faults as the linearized model in section 1.5. The mathematical linear model will always differ from the physical nonlinear model, meaning that corrections from measurements are needed.

### 5.3 Problem 3 - Prediction error covariance

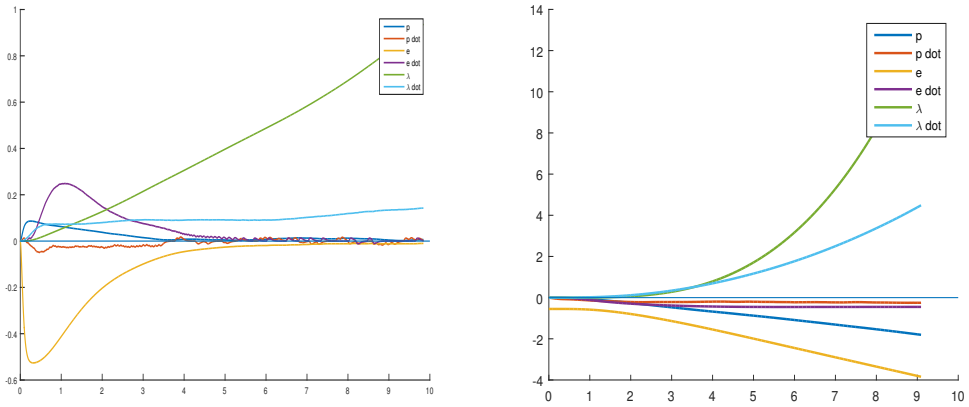
Now it is time to implement a data stream from the IMU to the open loop estimator so that it gets the correction the model requires to follow the physical model and in addition predict the next state. For that we need derive an expression for the uncertainty in the predicted state, $P[k+1]$. This is given by $\boldsymbol{A}_d$, $\bar{P}_d[k]$ and $\boldsymbol{Q}_d$ which equals $Var[\boldsymbol{w}_d]$ from eq. (69c). Using eq. (69a) and eq. (73):

$$
\begin{aligned}
\varepsilon[k+1] &= x[k+1] - \bar{x}[k+1] \\
&= A_d x[k] + B_d u[k] + w_d[k] - (A_d \bar{x}[k] + B_d u[k]) \\
&= A_d \varepsilon[k] + w_d[k]
\end{aligned}
$$

$$
\begin{aligned}
\bar{\boldsymbol{P}}[k+1] &= E[\bar{\varepsilon}[k+1]\bar{\varepsilon}^\top[k+1]] \\
&= E[\bar{\varepsilon}[k+1]\bar{\varepsilon}^\top[k+1]] \\
&= A_d \underbrace{E[\bar{\varepsilon}[k]\bar{\varepsilon}[k]^\top]}_{\textbf{P bar[k]}} A_d^\top + A_d \underbrace{E[\bar{\varepsilon}[k] w_d^\top[k]]}_{\textbf{0}} + \underbrace{E[w_d[k]\bar{\varepsilon}[k]]}_{\textbf{0}} A_d^\top + \underbrace{E[w_d[k] w_d^\top[k]]}_{\textbf{Var[w]} = \textbf{Q}} \\
&= A_d \bar{\boldsymbol{P}}[k] A_d^\top + Q_d
\end{aligned}
$$

Implemented in Matlab:

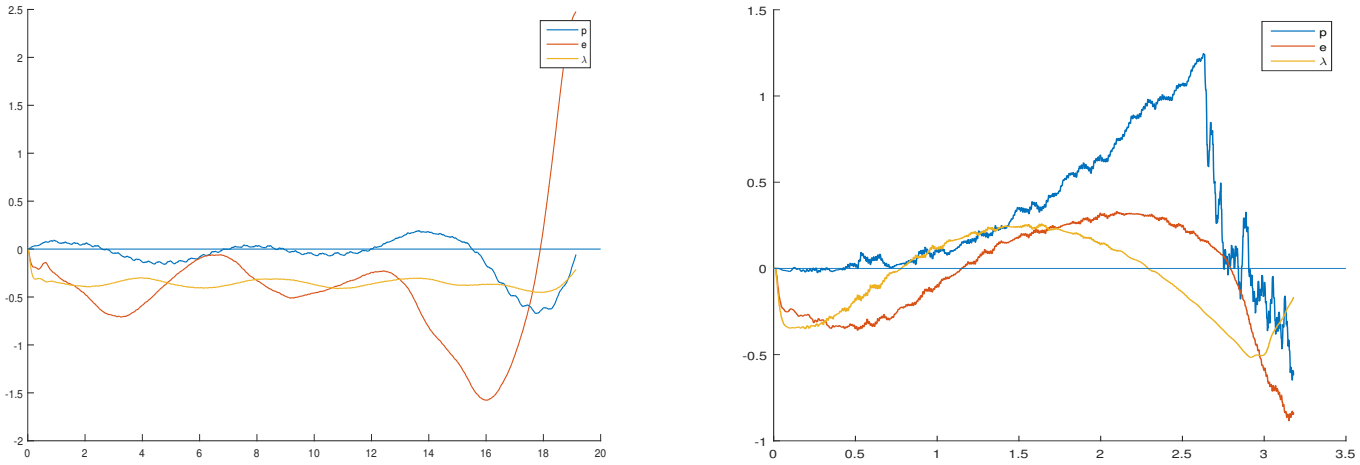
```
function [xk_next, P_pred_next] = openloop_estimator(xk,
    uk, A_d, B_d, Q_d, P_pred)

P_pred_next = A_d * P_pred * transpose(A_d) + Q_d;
xk_next = A_d*xk + B_d*uk;
```




(a) Low-pass filtered encoder output

(b) Predicted output

Figure 29: Plots for pitch, elevation and travel when the helicopter is launched from the linearization point




(a) Launched from the linearization point

(b) Launched from the non-linearization point

Figure 30: Two launching scenarios with an open loop estimator.

To test out our new model, we tried out with different $\boldsymbol{Q}_d$'s, but since we do not have a any correction, the error increased linearly with the rate of $\boldsymbol{Q}_d$. Consequently,

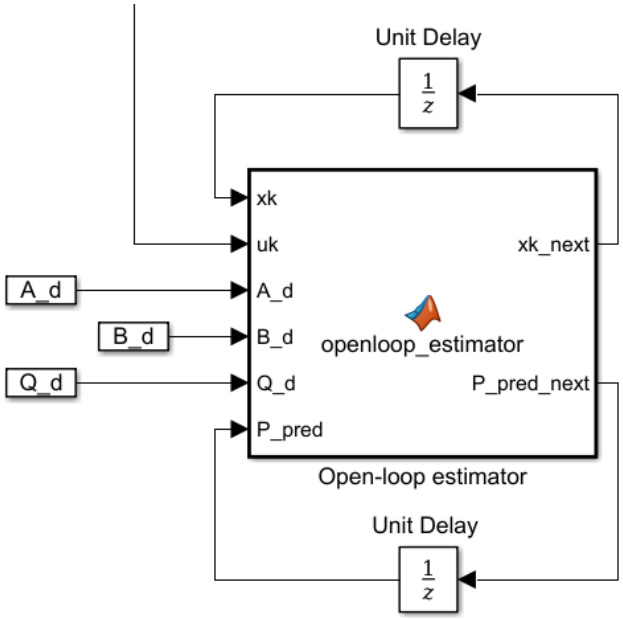



Figure 31: Open-loop estimator with P_Pred_next = $\hat{\boldsymbol{P}}[k+1]$

without correction our predicted states become more and more uncertain as time went by. Changing $\boldsymbol{Q}_d$ has therefore no effect on the open-loop estimates, as it only influences the correction step in the estimator.

# 6 Part VI - Correction Step

## 6.1 Problem 1 - Correction

The Kalman Filter will estimate the states of the system at current time step $k$ depending on the predicted state $\bar{\boldsymbol{x}}[k]$ and the measured states $\boldsymbol{y}[k]$. We let $\hat{\boldsymbol{x}}[k]$ be the measurement-corrected estimate at time-step $k$. The weighting matrix $\boldsymbol{K}[k]$ is scaled with the difference between prediction and measurement $\boldsymbol{y}[k] - \bar{\boldsymbol{y}}[k]$, as described in the following equations

$$\hat{\boldsymbol{x}}[k] = \bar{\boldsymbol{x}}[k] + \boldsymbol{K}[k](\boldsymbol{y}[k] - \bar{\boldsymbol{y}}[k]) \tag{74a}$$
$$\bar{\boldsymbol{y}}[k] = \boldsymbol{C}_d\bar{\boldsymbol{x}}[k] \tag{74b}$$

From this, we can derive the equation

$$\hat{\boldsymbol{x}}[k] = (\boldsymbol{I} - \boldsymbol{K}[k]\boldsymbol{C}_d)\bar{\boldsymbol{x}} + \boldsymbol{K}[k]\boldsymbol{y} \quad , \tag{75}$$

which shows the relation between the matrix $\boldsymbol{K}$ and to which extent the estimate will rely on either the prediction or the measurement. A large Kalman gain $\boldsymbol{K}[k]$ close to 1 emphasizes the measurement $\boldsymbol{y}$, and a low gain close to 0 relies on model predictions.

As the IMU does not send data at all times, the estimated state $\hat{\boldsymbol{x}}[k]$ is set to the predicted state $\bar{\boldsymbol{x}}[k]$ when there is no new data. We may now use eq. (73) with the estimated state:

$$\bar{\boldsymbol{x}}[k+1] = \boldsymbol{A}_d\hat{\boldsymbol{x}}[k] + \boldsymbol{B}_d\boldsymbol{u}[k] \tag{76}$$

The eq. (74) is implemented in the Matlab-function block from section 5.3:

```
if (new_data)
    xk_est = xk_pred + Kk*(y-C_d*xk_pred);
else
    xk_est = xk_pred;
end
xk_pred_next = A_d*xk_est + B_d*uk;
```

## 6.2 Problem 2 - Correction covariance

We want to find the covariance of the error in our estimate. This can be written as a matrix of the form

$$\hat{\boldsymbol{P}}[k] = \mathrm{E}[\hat{\epsilon}[k]\hat{\epsilon}^\top[k]] = \mathrm{E}[(\boldsymbol{x} - \hat{\boldsymbol{x}})(\boldsymbol{x} - \hat{\boldsymbol{x}})^\top] \quad . \tag{77}$$

Given the expression for $\bar{\boldsymbol{P}}[k]$, we can then find $\hat{\boldsymbol{P}}[k]$ by inserting eqs. (69b) and (75) into eq. (77), using $(\boldsymbol{x} - \bar{\boldsymbol{x}}) = \bar{\epsilon}[k]$ :

$$\hat{\boldsymbol{P}}[k] = \mathrm{E}[((\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)\bar{\epsilon} - \boldsymbol{K}\boldsymbol{v}_d)((\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)\bar{\epsilon} - \boldsymbol{K}\boldsymbol{v}_d)^\top] \tag{78a}$$

$$= \mathrm{E}[(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)\bar{\epsilon}\bar{\epsilon}^\top(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)^\top] + \mathrm{E}[\boldsymbol{K}\boldsymbol{v}_{\boldsymbol{d}}\boldsymbol{v}_{\boldsymbol{d}}^\top] \tag{78b}$$

$$= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)\mathrm{E}[\bar{\epsilon}\bar{\epsilon}^\top](\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)^\top + \boldsymbol{K}\mathrm{E}[\boldsymbol{R}_d]\boldsymbol{K} \tag{78c}$$

$$= (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)\bar{\boldsymbol{P}}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C}_d)^\top + \boldsymbol{K}\boldsymbol{R}_{\boldsymbol{d}}\boldsymbol{K}^\top \quad . \tag{78d}$$

This expression for the covariance error in our estimate is the a posteriori error covariance, which results from our predicted error covariance $\bar{\boldsymbol{P}}$. This is then used to calculate the new a priori covariance error, as described by eq. (68b).

As in section 6.1, we let the estimated error covariance $\hat{\boldsymbol{P}}[k]$ equal the predicted error covariance $\bar{\boldsymbol{P}}[k]$ when there is no new data from the IMU. We also let the predicted covariance $\bar{\boldsymbol{P}}[k+1]$ use the estimated covariance $\hat{\boldsymbol{P}}[k]$. This is added to our Matlab function block:

```
if (new_data)
    P_est = (eye(6) - Kk*C_d) * P_pred * transpose(eye(6)-
       Kk*C_d) + Kk*R_d*transpose(Kk);
else
    P_est = P_pred;
end
P_pred_next = A_d * P_est * transpose(A_d) + Q_d;
```

### 6.3 Problem 3 - Weighting matrix K

The error covariance matrix represents all of the 6 states of the helicopter. The elements on the diagonal therefore contains the error variance of each state, while the rest of the elements contains the covariances between the states. The Kalman filter considers only the variances, which should be minimized for optimal filtering. We therefore want to find an expression for our weighting matrix $\boldsymbol{K}[k]$ which minimizes the trace of the a posteriori error covariance, $\mathrm{tr}(\hat{\boldsymbol{P}}[k])$, corresponding to the variances only. This can be done by differentiating with respect to $\boldsymbol{K}$. Using eq. (78d) and

the properties of traces [3, Section 2.5], we calculate:

$$\frac{\partial \mathrm{tr}(\hat{\boldsymbol{P}}[k])}{\partial \boldsymbol{K}[k]} = \frac{\partial}{\partial \boldsymbol{K}}[\mathrm{tr}((\boldsymbol{I}-\boldsymbol{K}\boldsymbol{C}_d)\bar{\boldsymbol{P}}(\boldsymbol{I}-\boldsymbol{K}\boldsymbol{C}_d)^\top)+\mathrm{tr}(\boldsymbol{K}\boldsymbol{R}_{\boldsymbol{d}}\boldsymbol{K}^\top)] \tag{79a}$$

$$= \frac{\partial}{\partial \boldsymbol{K}}[tr(\bar{\boldsymbol{P}}-(\boldsymbol{K}\boldsymbol{C}_d)\bar{\boldsymbol{P}}-\bar{\boldsymbol{P}}(\boldsymbol{K}\boldsymbol{C}_{\boldsymbol{d}})^\top + (\boldsymbol{K}\boldsymbol{C}_{\boldsymbol{d}})\bar{\boldsymbol{P}}(\boldsymbol{K}\boldsymbol{C}_{\boldsymbol{d}})^\top)]+2\boldsymbol{K}\boldsymbol{R}_d \tag{79b}$$

$$= \boldsymbol{0}+\frac{\partial}{\partial \boldsymbol{K}}[\mathrm{tr}(\boldsymbol{K}\boldsymbol{C}_{\boldsymbol{d}}\bar{\boldsymbol{P}})-\mathrm{tr}(\bar{\boldsymbol{P}}\boldsymbol{C}_{\boldsymbol{d}}{}^\top\boldsymbol{K}^\top) + \mathrm{tr}(\boldsymbol{K}\boldsymbol{C}_{\boldsymbol{d}})\bar{\boldsymbol{P}}(\boldsymbol{K}\boldsymbol{C}_{\boldsymbol{d}})^\top)]+2\boldsymbol{K}\boldsymbol{R}_d \tag{79c}$$

$$= -(\boldsymbol{C}_{\boldsymbol{d}}\bar{\boldsymbol{P}})^\top-(\boldsymbol{C}_{\boldsymbol{d}}\bar{\boldsymbol{P}}^\top)^\top+2\boldsymbol{K}\boldsymbol{R}_d + \frac{\partial}{\partial \boldsymbol{u}}\mathrm{tr}(\boldsymbol{u}\bar{\boldsymbol{P}}\boldsymbol{u}^\top)\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{K}} \quad , \quad \boldsymbol{u}=\boldsymbol{K}\boldsymbol{C}_d \tag{79d}$$

$$= -2(\boldsymbol{C}_{\boldsymbol{d}}\bar{\boldsymbol{P}})^\top+2\boldsymbol{K}\boldsymbol{C}_{\boldsymbol{d}}\bar{\boldsymbol{P}}\boldsymbol{C}_{\boldsymbol{d}}{}^\top+2\boldsymbol{K}\boldsymbol{R}_d \quad . \tag{79e}$$

Here we have used the fact that $\bar{\boldsymbol{P}}$ is symmetric. Setting this expression to $\boldsymbol{0}$ then gives us the $\boldsymbol{K}[k]$ which minimizes the variances in the error covariance matrix:

$$\boldsymbol{K}[k] = \bar{\boldsymbol{P}}\boldsymbol{C}_{\boldsymbol{d}}{}^\top(\boldsymbol{C}_{\boldsymbol{d}}\bar{\boldsymbol{P}}\boldsymbol{C}_{\boldsymbol{d}}{}^\top+\boldsymbol{R}_{\boldsymbol{d}})^{-1} \quad . \tag{80}$$

This is the $\boldsymbol{K}$ that is implemented in our Kalman Filter. The block is updated to the final version with code shown in fig. 32 and simulink representation shown in fig. 33.

```
function [xk_pred_next, P_pred_next, xk_est, P_est] =
   kalman_estimator(xk_pred, uk, A_d, B_d, Q_d, C_d, y,
   R_d, new_data, P_pred)
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
P_pred_next = A_d * P_est * transpose(A_d) + Q_d;
xk_pred_next = A_d*xk_est + B_d*uk;
```

Figure 32: Kalman function block code

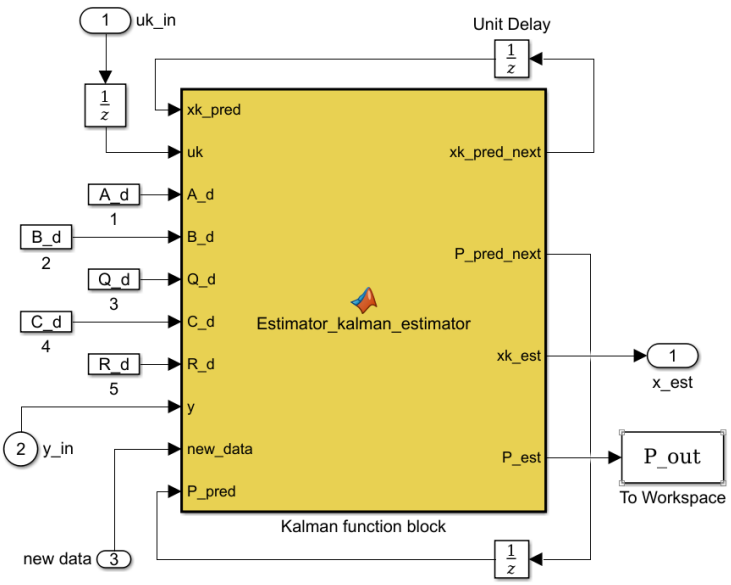



Figure 33: Simulink implementation of the Kalman filter.

### 6.4 Problem 4 - Tuning

As shown in eq. (69c), the disturbance $\boldsymbol{w}_d[k]$ is assumed to be white, with a Gaussian distribution around 0 and stochastic disturbance matrix $\boldsymbol{Q}_d$. The undetermined matrix $\boldsymbol{Q}_d$ is therefore left as a tuning variable, where we will adjust the weight of each individual diagonal element according to the response in each state.

From eq. (68b), it comes that the size of $\boldsymbol{Q}_d$ describes the relation between the measured states and the predicted states, as it affects the relative size of $\boldsymbol{K}[k]$, described in eq. (67a). As written in section 6.1, the size of $\boldsymbol{K}[k]$ will affect the relationship between prediction and measurement. Increasing the magnitude of $\boldsymbol{Q}_d$ will therefore make the filter place more weight on the measured states, whereas decreasing the magnitude makes the KF rely more on model predictions. Setting $\boldsymbol{Q_d}$ to $\mathbf{0}$ is the same as using an open-loop estimator and setting $\boldsymbol{Q_d}$ to $\boldsymbol{\infty}$ means pure measurement consideration. As we use the IMU to measure the states, we would therefore want fairly low values of $\boldsymbol{Q_d}$ with the IMU being noisy.

In section 6.5, we will use the helicopter with feedback directly from the estimated states, and the performance will depend on the KF's ability to filter out noise while maintaining accurate measurements. Thus, we use feedback from the encoder in this task, which allows us to compare our filter performance to the accurate encoder output.

The first step in tuning an optimal $\boldsymbol{Q}_d$ was to find approximately of what size the diagonal entries should have. We therefore started by setting the diagonal entries to 2, and then tracked the encoder measurement versus the estimated states. The

IMU produces a lot of noise as it is placed on a flying helicopter (see section 4.6), so that the estimated states were also full of noise. By multiplying the weights to different negative powers of 10, we found that the best dimension for the diagonal was $10^{-5}$. In general, values larger than this had too much noise, but lowering it further reduced the accuracy of the estimate. This is illustrated in fig. 34, and we can clearly see the noise disappear as we lower $\boldsymbol{Q}_d$.

After finding the approximate scale of the single entries, we can now tune each individually and track the result through the encoder measurement. We found that estimate for pitch was quite noisy, even if we lowered the values of $\boldsymbol{Q}_d(1,1)$ significantly. Doing so made the estimate track the measurements poorly, so we left it at $2 \cdot 10^{-5}$. The result is seen in fig. 35, where "Est" denotes values from our estimate $\hat{\boldsymbol{x}}$ and "Enc" refers to the encoder.

The pitch rate was also able to track fairly good with $\boldsymbol{Q}_d(2,2) = 2 \cdot 10^{-5}$. This is shown in fig. 36, with the $\hat{p}$ tracking the IMU with low noise magnitude. However, the IMU measurements versus encoder reference is slightly off, as we were encountering difficulties and inconsistencies with the IMU. Tuning this parameter to either low or high did not alter the estimated pitch rate significantly.

The elevation measurement from the IMU was very noisy, as seen in fig. 37. The KF removes most of the noise, while maintaining a decent accuracy. Leaving this equal to the pitch and pitch rate values seemed to give the best results.

The estimate for the elevation rate did track very accurately, even with low values of $\boldsymbol{Q}_d(4,4)$. We could therefore lower it significantly. fig. 38 shows the somewhat noisy response before lowering the value.

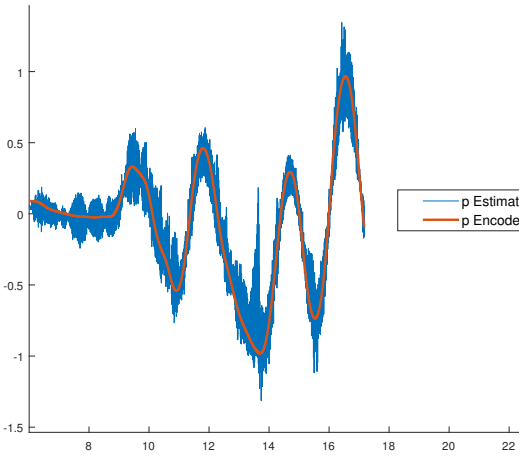
We chose to observe the Kalman Filter's effect first and foremost with pitch, pitch rate, elevation and elevation rate, as there is no reference input signal to either travel angle or travel rate. As the travel angle is not measurable without the encoders, $\boldsymbol{Q}_d(5,5)$ is set to 0. In order to filter out travel rate noise, $\boldsymbol{Q}_d(6,6)$ is left at $2 \cdot 10^{-5}$.

After tuning the Kalman Filter, we ended up with a matrix

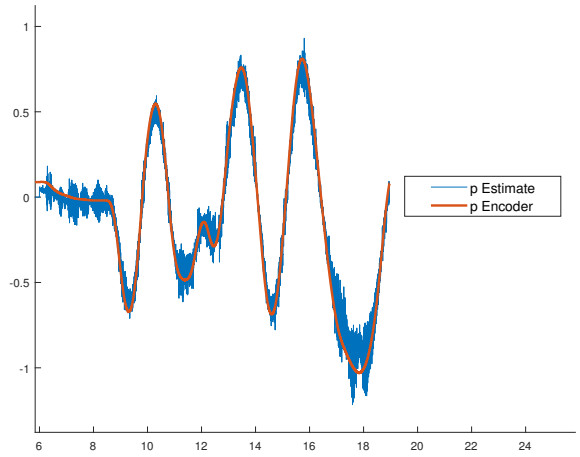
$$\boldsymbol{Q}_d = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.008 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \cdot 10^{-5} \quad . \tag{81}$$

## 6.5 Problem 5 - The Kalman filter

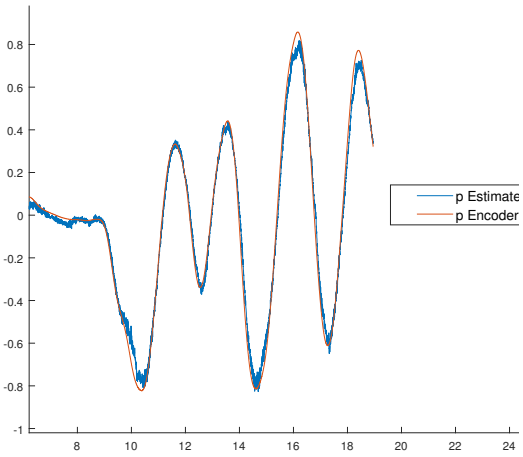
As the filter is now tuned, we will now attempt to fly the helicopter using the filter directly as feedback. As the IMU uses a few seconds to start up, we disable the filter



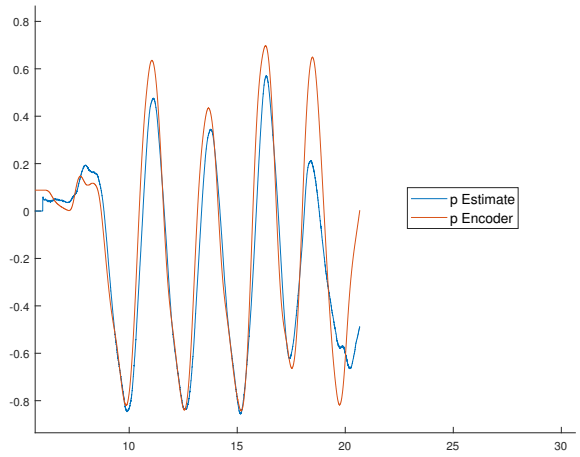

(a) $\text{diag}(\boldsymbol{Q}_d) = 2 \cdot 10^{0}$




(b) $\text{diag}(\boldsymbol{Q}_d) = 2 \cdot 10^{-3}$




(c) $\text{diag}(\boldsymbol{Q}_d) = 2 \cdot 10^{-5}$




(d) $\text{diag}(\boldsymbol{Q}_d) = 2 \cdot 10^{-7}$

Figure 34: Testing different values for $\boldsymbol{Q}_d$.

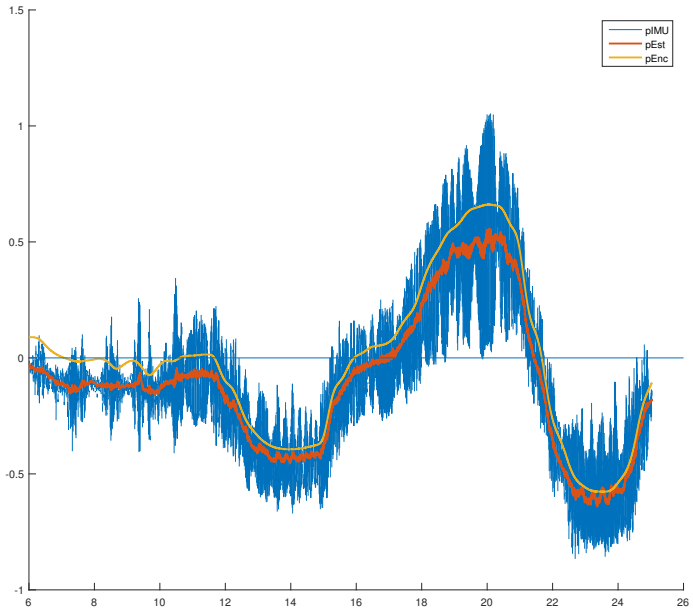



Figure 35: Pitch response from IMU, $\hat{\boldsymbol{x}}$ and encoder with joystick input.

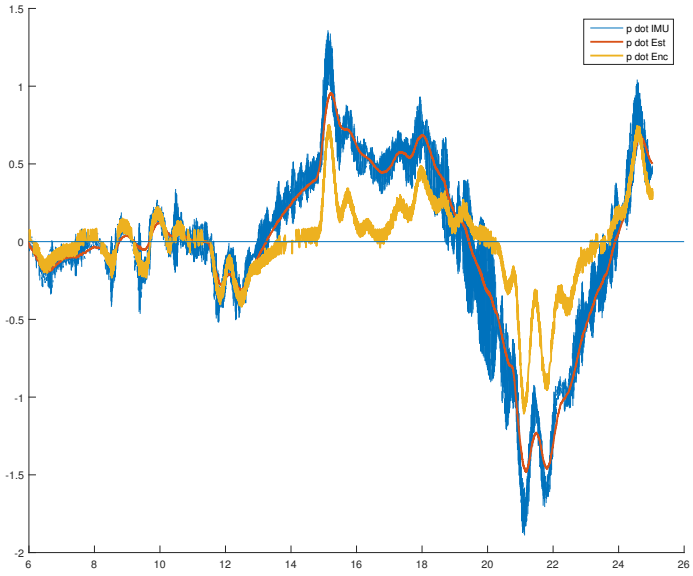



Figure 36: Pitch rate response from IMU, $\hat{\boldsymbol{x}}$ and encoder with joystick input.

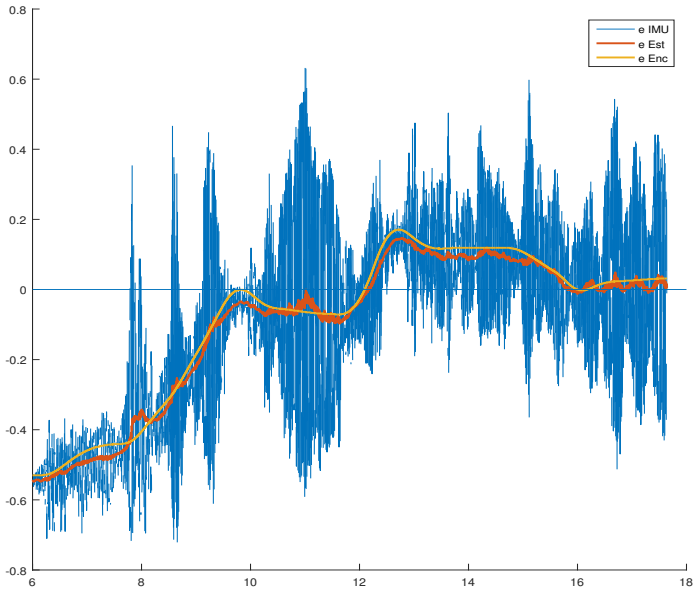



Figure 37: Elevation response from IMU, $\hat{\boldsymbol{x}}$ and encoder with joystick input.

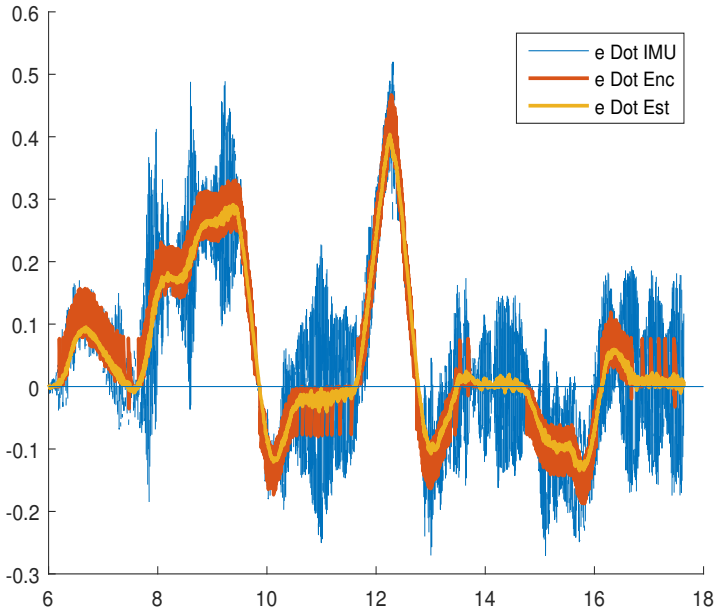



Figure 38: Elevation rate response from IMU, $\hat{\boldsymbol{x}}$ and encoder with joystick input.

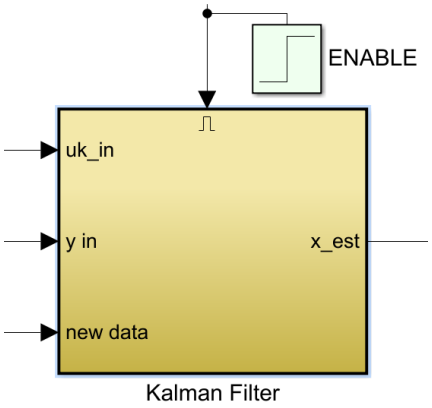



Figure 39: Using the "Enable" function at $t = 6$s in Simulink

and controller until the IMU sends the first data. This is done by using an "Enable" block for subsystems, as shown in fig. 39.

Using the state estimates as an input for the helicopter with the same tuning as in section 6.4 resulted at first in a shaky response. The elevation rate responded decent to the reference joystick input, but the helicopter pitch was oscillating badly. We were able to fly the helicopter, but it was clear we needed to tune $\boldsymbol{Q_d}$ more.

By decreasing the weights of $\boldsymbol{Q_d}(1,1)$, we found that the oscillating pitch response disappeared, meaning it was a result of noise of the measurements in the IMU. When flying the helicopter, we now saw a constant pitch offset of $\approx 0.2$ radians. This was solved by adding a constant offset to the IMU. Although we assume the noise to be purely white, this slight bias is a counter proof of that. We also found that increasing $\boldsymbol{Q_d}(2,2)$ reduced the visible shakiness of the helicopter. From trying different values here, we increased the value to 0.1, which made the helicopter movement smoother.

The same phenomenon later occurred while tuning the elevation and elevation rate with the KF as feedback. The elevation rate was constantly measured to be lower than it actually was, such that the helicopter was hard to steer. It is therefore clear that with such a noisy measurement method as the IMU on a vibrating helicopter, the noise tends to be biased to the point where it affects the system. A lot of tuning was done to $\boldsymbol{Q_d}(3,3)$ and $\boldsymbol{Q_d}(4,4)$ to remove the effects of the bias without having to add another offset, without any proper results. The Kalman Filter was not able to handle the bias in $\dot{e}$, which is sensible given that one of the reference signals is $\dot{e}$.

Correcting the offset and slightly adjusting a few of the variables in $\boldsymbol{Q}_d$, we were able to maneuver the helicopter with ease.

After examining the inputs of the Kalman Filter, we also tried to experiment with different values of the measured noise variance $\boldsymbol{R}_d$. The values for this stochastic matrix were obtained in section 4.6 with a flying helicopter, as shown in eq. (66). We wanted to test the significance of this constant matrix. Experimentation with increasing the diagonal entries led to a variety of responses from the helicopter, most

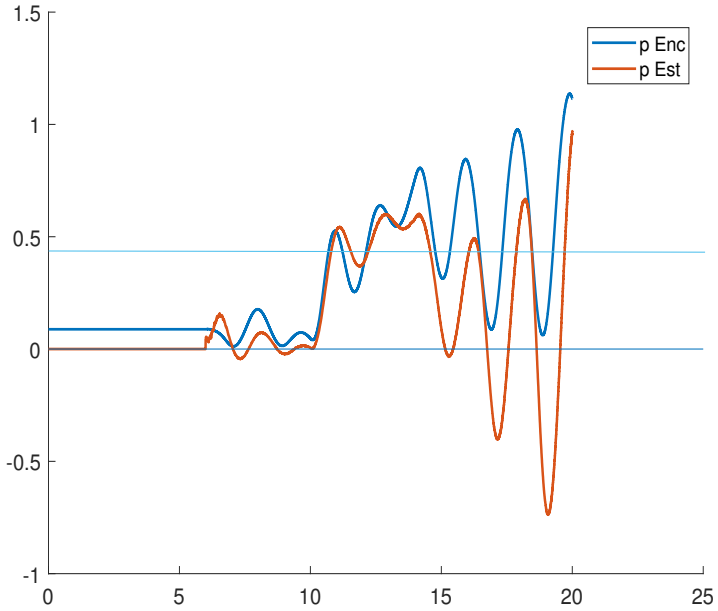



Figure 40: Pitch response with a custom $\boldsymbol{R}_d$ matrix.

noticeably after increasing element $\boldsymbol{R}_d(4,4)$. This corresponds to the measured pitch noise, and increasing this above 1 resulted in an oscillating helicopter. fig. 40 shows the pitch response with a set reference angle of $p = 0 \rightarrow 0.4\text{rad}$ at $t = 5$, with $\boldsymbol{R}_d(4,4) = 1$.

As described in section 4.6, using the IMU measurements directly as feedback to the helicopter made the helicopter practically non-controllable. The measurements are extremely noisy and the IMU does not send information at every time step.

Using an open loop estimator to control a system is seldom a good idea. This requires a *perfect* model, which is not the case. As the open loop system has no way of correcting itself, this was not the best.

Flying with the Kalman Filter greatly changed the performance, albeit after a lot of tuning. The responsiveness is not as fast as flying directly with the encoders as feedback, but that is to be expected.

The encoders make it quite easy to maneuver the helicopter as long as the helicopter controller is well tuned. Using a tuned LQR from Part III ensured this.

# 7 Conclusion

With a properly tuned Kalman Filter, we were able to fly the helicopter smooth and controlled, though the response was not as quick as feedback from the encoders. Encoders are nonetheless usually unavailable in a helicopter that freely moves through the air. The Kalman Filter enabled us to use an extremely noisy but easily available and affordable IMU to measure our control states through a gyroscope and accelerometers.

# References


[1] Hwang Patrick Y.C. Brown Robert Grover. *Introduction to Random Rignal and Kalman Filtering*. http://s1.nonlinear.ir/epublish/book/Introduction_to_Random_Signals_and_Applied_Kalman_Filtering_0470609699.pdff. 2012.

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

[3] M.S. Pedersen K.B. Petersen. *The Matrix Cookbook*. https://math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf. 2012.