Helicopter lab assignment

Marius Loraas, Marius Moum [253950], Even Wanvik

August 2018

Chapter 1

Introduction

This lab assignment is a part of the subject TTK4115 Linear System Theory at NTNU and is based around a Quanser 3 DOF Helicopter model.

Figure 1.1: The helicopter, with its two propellers and counterweight. [4]

The purpose of this assignment is to:

- Derive a model for a given system (the helicopter) that can be used for control purposes.
- Derive and apply PD and multivariable controllers in a real-time environment.
- Demonstrate how states can be estimated that are not directly measured.
- Give an introduction to optimal control by developing a linear quadratic regulator (LQR) for the helicopter.

The lab is completed using helicopter #10.

It is expected that the reader of this report is familiar with the assignment and the physical helicopter.

Chapter 2

Part 1 - Mathematical model

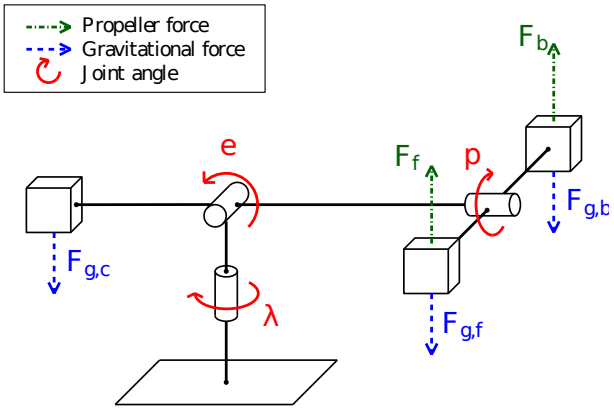

Figure 2.1: Forces acting on the helicopter.
[2]

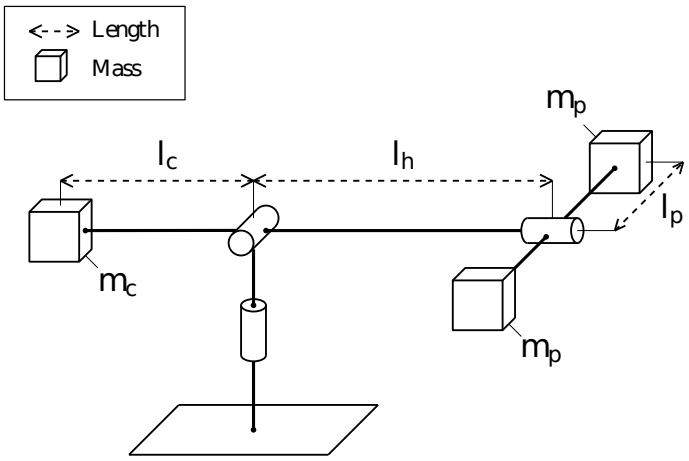

Figure 2.2: The dimensions of the helicopter.
[2]

2.1 Problem 1 - Mathematical model

In order to compute the equation of motion we have to understand what kind of physics affecting the helicopter. A good starting point is the Newton second law of rotation 2.1. We assume ideal conditions where friction forces and centripetal forces are neglected

$$\sum \tau = I\alpha \tag{2.1}$$

The helicopter can move both forwards and backwards. This is determined by the pitch angle, which is created by the difference in force activated on the front and back propeller motors. The propeller motor forces are given by F_f and F_b which is assumed to be linear related to the voltages V_f and V_b supplied to the propeller motors. Giving us the forces $F_f = K_f\ V_f$ and $F_b = K_f\ V_b$. We compute the rotation of moments around the pitch joint and arrive at the equation 2.2

$$\begin{aligned} \circlearrowleft^{+} \sum \tau &= F_f l_p - F_b l_p \\ K_f V_f l_p - K_f V_b l_p &= K_f l_p V_d \\ J_p \ddot{p} &= K_f l_p V_d \\ J_p \ddot{p} &= L_1 V_d \end{aligned} \tag{2.2}$$

To determine the elevation angle we need to bring along the gravitational forces. These forces are given by the masses of each motor, m_p, and the counterweight mass, m_c, multiplied my the gravitational constant g. We are now able to determine the equation 2.3 around the elevation joint.

$$
\begin{aligned}
\circlearrowleft^{+} \sum \tau &= (F_{g,c} l_p - (F_{g,f} + F_{g,b}) l_h) cos(e) + (F_f + F_b) l_h cos(p) \\
\circlearrowleft^{+} \sum \tau &= g(m_c l_c - 2 m_p l_h) cos(e) + K_f l_h V_s cos(p) \\
J_e \ddot{e} &= g(m_c l_c - 2 m_p l_h) cos(e) + K_f l_h V_s cos(p) \\
J_e \ddot{e} &= L_2 cos(e) + L_3 V_s cos(p)
\end{aligned}
\tag{2.3}
$$

The travel is the rotation around the z-axis and will be dependant on both the pitch and the elevation. This is due to the difference in arm length which occurs when the elevation is moving between zero or max/min. In the case were the pitch angle is zero, there will be no side forces, and accordingly no travel. As soon as the pitch angle is different from zero, the helicopter will start travelling due to the torque that is produced. The torque will then be dependant on the difference between arm lengths and the force of the motors. This gives us the equation 2.4

$$
\begin{aligned}
\circlearrowleft^{+} \sum \tau &= -(F_f + F_b) l_h sin(p) cos(e) \\
\circlearrowleft^{+} \sum \tau &= -K_f V_s l_h sin(p) cos(e) \\
J_\lambda \ddot{\lambda} &= L_4 V_s sin(p) cos(e)
\end{aligned}
\tag{2.4}
$$

2.2 Problem 2 - Linearization

Defining variables $(p^*,e^*,\lambda^*)^T$, around which we want to linearize the equations of motions, where $p^* = e^* = \lambda^* = 0$. First we'll determine the voltages V_s^* and V_d^* such that $(p^*,e^*,\lambda^*)^T$ is an equilibrium point of the system, i.e. $\dot{p}^* = \dot{e}^* = \dot{\lambda}^* = \ddot{p}^* = \ddot{e}^* = \ddot{\lambda}^* = 0$ (zero speed or acceleration implies the system is at an equilibrium) and $(V_s, V_d)^T = (V_s^*, V_d^*)^T$:

$$J_p\ddot{p} = L_1 V_d \overset{\ddot{p}=0}{\Rightarrow} V_d^* = 0 \tag{2.5a}$$

$$J_e\ddot{e} = L_2 cos(e) + L_3 V_s^* cos(p) \overset{\ddot{e}=0}{\Rightarrow} V_s^* = -\frac{L_2 cos(0)}{L_3 cos(0)} = \frac{L_2}{L_3} \tag{2.5b}$$

A coordinate transformation that simplifies the further analysis is introduced in the project manual [REF, section 5.1.2 equation 4], assuming that the system is at equilibrium the transform turns into:

$$\begin{bmatrix}\tilde{p}\\ \tilde{e}\\ \tilde{\lambda}\end{bmatrix} = \begin{bmatrix}p\\ e\\ \lambda\end{bmatrix} - \underbrace{\begin{bmatrix}p^*\\ e^*\\ \lambda^*\end{bmatrix}}_{=0} \Rightarrow \begin{bmatrix}p\\ e\\ \lambda\end{bmatrix} = \begin{bmatrix}\tilde{p}\\ \tilde{e}\\ \tilde{\lambda}\end{bmatrix} \tag{2.6a}$$

$$\begin{bmatrix}\tilde{V}_s\\ \tilde{V}_d\end{bmatrix} = \begin{bmatrix}V_s\\ V_d\end{bmatrix} - \begin{bmatrix}V_s^*\\ V_d^*\end{bmatrix} \Rightarrow \begin{bmatrix}V_s\\ V_d\end{bmatrix} = \begin{bmatrix}\tilde{V}_s\\ \tilde{V}_d\end{bmatrix} + \begin{bmatrix}\frac{L_2}{L_3}\\ 0\end{bmatrix} \tag{2.6b}$$

The variables around which we decided to linearize is equal to zero, as declared earlier in this section, making the original state vectors equal to the new ones. We can now substitute the equations of motion (eq. [ref to eq.ofmotion]) into the coordinate transformation introduced in eq 2.6 to linearize the system around the state vector $(\tilde{p}, \tilde{e}, \tilde{\lambda})^T$ and input vector $(\tilde{V}_s, \tilde{V}_d)^T$. Assuming that the moments of inertia are constant and given by (dropp dette?)

$$J_p = 2m_p l_p^2 \tag{2.7a}$$

$$J_e = m_c l_c^2 + 2m_p l_h^2 \tag{2.7b}$$

$$J_\lambda = m_c l_c^2 + 2m_p(l_h^2 + l_p^2) \tag{2.7c}$$

the equations of motion can be written as:

$$\ddot{\tilde{p}}(\tilde{p} - 0) = \frac{L_1 \tilde{V}_d}{J_p} \tag{2.8a}$$

$$\ddot{\tilde{e}}(\tilde{e} - 0) = \frac{L_2 cos(\tilde{e}) + L_3(\tilde{V}_s - \frac{L_2}{L_3})cos(\tilde{p})}{J_e} \tag{2.8b}$$

$$\ddot{\tilde{\lambda}}(\tilde{\lambda} - 0) = \frac{L_4(\tilde{V}_s - \frac{L_2}{L_3})cos(\tilde{e})sin(\tilde{p})}{J_\lambda} \tag{2.8c}$$

Linearizing the resulting system around a given point $(\tilde{p}, \tilde{e}, \tilde{\lambda})^T = (0,0,0)^T$ with input $(\tilde{V}_s, \tilde{V}_d)^T = (0,0)^T$ gives us the linearized state/input Jacobian matrices (the angular velocity vectors are omitted, as they do not alter the states)

$$\tilde{A} = \begin{bmatrix} \frac{\partial \ddot{\tilde{p}_1}}{\partial \tilde{p}_1} & \frac{\partial \ddot{\tilde{p}_1}}{\partial \tilde{e}_1} & \frac{\partial \ddot{\tilde{p}_1}}{\partial \tilde{\lambda}} \\ \frac{\partial \ddot{\tilde{e}_1}}{\partial \tilde{p}_1} & \frac{\partial \ddot{\tilde{e}_1}}{\partial \tilde{e}_1} & \frac{\partial \ddot{\tilde{e}_1}}{\partial \tilde{\lambda}} \\ \frac{\partial \ddot{\tilde{\lambda}}}{\partial \tilde{p}_1} & \frac{\partial \ddot{\tilde{\lambda}}}{\partial \tilde{e}_1} & \frac{\partial \ddot{\tilde{\lambda}}}{\partial \tilde{\lambda}} \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} \frac{\partial \ddot{\tilde{p}_1}}{\partial \tilde{V}_d} & \frac{\partial \ddot{\tilde{p}_1}}{\partial \tilde{V}_s} \\ \frac{\partial \ddot{\tilde{e}_1}}{\partial \tilde{V}_d} & \frac{\partial \ddot{\tilde{e}_1}}{\partial \tilde{V}_s} \\ \frac{\partial \ddot{\tilde{\lambda}}}{\partial \tilde{V}_d} & \frac{\partial \ddot{\tilde{\lambda}}}{\partial \tilde{V}_s} \end{bmatrix}. \tag{2.9}$$

Applying the derivations using the variables given around equilibrium $(\tilde{p}, \tilde{e}, \tilde{\lambda}$ and $(\tilde{V}_s, \tilde{V}_d)^T$, where $(\tilde{p}, \tilde{e}, \tilde{\lambda})^T$ will be referred to as $\tilde{x}$ and $(\tilde{V}_s, \tilde{V}_d)^T$ as $\tilde{u}$ for simplification, the state/input matrices can be used to produce the linearized state-space model

$$\dot{\tilde{x}} = \tilde{A}\tilde{x} + \tilde{B}\tilde{u} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{L_4}{J_\lambda} & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & \frac{L_1}{J_p} \\ \frac{L_3}{J_e} & 0 \\ 0 & 0 \end{bmatrix} u. \tag{2.10}$$

A simple matrix multiplication provides each equation with a coefficient:

$$\ddot{\tilde{p}} = \frac{L_1 \tilde{V}_d}{J_p} = K_1 \tilde{V}_d \Rightarrow K_1 = \frac{L_1}{J_p} = \frac{K_f}{2 m_p l_p} \tag{2.11a}$$

$$\ddot{\tilde{e}} = \frac{L_3 \tilde{V}_s}{J_e} = K_2 \tilde{V}_s \Rightarrow K_2 = \frac{L_3}{J_e} = \frac{K_f l_h}{m_c l_c^2 + 2 m_p l_h^2} \tag{2.11b}$$

$$\ddot{\tilde{\lambda}} = K_3 \frac{L_4 \tilde{p}}{J_\lambda} \Rightarrow K_3 = \frac{L_3}{J_e} = \frac{K_f l_h}{m_c l_c^2 + 2 m_p l_h^2} \tag{2.11c}$$

calculate K's K1 = 0.582, K2 = 0.0935, K3 = -0.612

2.3 Problem 3 - Feed forward control

When implementing feed forward control, the operator controls the actuators directly.

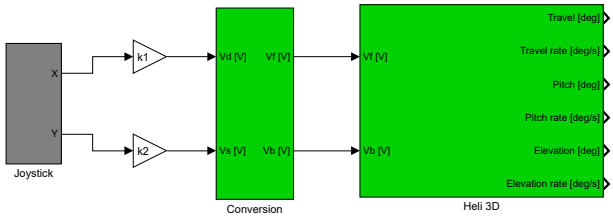

Figure 2.3: Simulink diagram showing the implementation of feed forward control of the helicopter.

The joystick returns a value between -1 and 1. This must be scaled in order to match the magnitude of the values in the rest of the system. This scaling will also affect the sensitivity of the joystick. By trial and error, we found that the values $K_1 = 5$ and $K_2 = 6$ gave a reasonable compromise between sensitivity and maximum thrust.

Looking at the mathematical model of the helicopter, one would assume that increasing only the voltage sum V_s would not affect the pitch angle of the helicopter. However, in real life, we observe that this is not the case. The pitch clearly changes, and the helicopter starts travelling.

The reason for these discrepancies between the helicopter and the theoretical model, is that when deducing the model, a series of assumptions were made.

The assumption of that both motors are equally heavy and powerful, are very likely to be incorrect. This may be one of the factors affecting the pitch angle.

Another example which lead to discrepancies was the turbulence in the environment around the helicopter. We could easily see that the helicopter struggled to maintain the balance when it operated close to the corner of the room. A similar behaviour occurred when the door opened and a draught of air went through the room. At take-off, the increased lift from the "ground effect" may also affect the model.

2.4 Problem 4 - Encoder offset/gain and motor force constant

we first needed to bring the helicopter into a stable and level state. One way of doing this is manually by using the joystick, but as we experienced in the previous task, this was quite difficult. Therefore, we implemented two Simulink PI regulators with very slow response, and let the system stabilise itself.

By choosing reference values for the regulators using trial and error, we were able to bring the helicopter to a stable, steady and level state. We found the following constants:

$$p_{offset} = 6,0^\circ = 6,0^\circ \cdot \frac{2\pi}{360^\circ}\ [rad] \approx 0,105\ [rad] \tag{2.12a}$$

$$e_{offset} = 29,8^\circ = 29,8^\circ \cdot \frac{2\pi}{360^\circ}\ [rad] \approx 0,520\ [rad] \tag{2.12b}$$

$$V_s = 6,819\ [V] \tag{2.12c}$$

By using the measured value of V_S^* we are now able to determine the value of K_f.

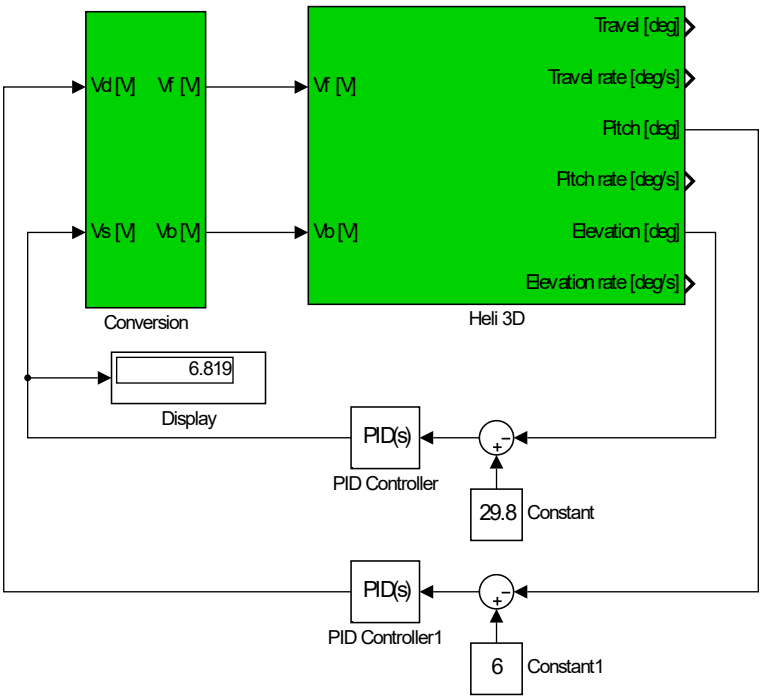

Figure 2.4: Simulink diagram showing how we stabilised the system to find values for the angle offset constants and voltage Vs. We adjusted the two constants until the helicopter was stable. The screen shot were made when the system was in equilibrium.

$$V_S^* = -\frac{L_2}{L_3} = -\frac{g(m_c l_c - 2m_p l_h)}{K_f l_h} \tag{2.13a}$$

$$K_f = -\frac{g(m_c l_c - 2m_p l_h)}{V_S^* l_h} \tag{2.13b}$$

$$K_f = -\frac{9.81(1.92 \cdot 0.46 - 2 \cdot 0.72 \cdot 0.66)}{6.819 \cdot 0.66} \tag{2.13c}$$

$$K_f = 0.1465 \tag{2.13d}$$

Chapter 3

Monovariable control

3.1 Problem 1 - PD controller for pitch angle

In this part of the assignment the goal is to design a controller for the pitch angle $\tilde{p}$ and travel rate $\dot{\tilde{\lambda}}$.
We have been given a PD controller in the first problem, and its given as

$$\tilde{V_d} = K_{pp}(\tilde{p_c} - \tilde{p}) - K_{pd}\dot{\tilde{P}} \tag{3.1}$$

Moving on we are asked to Laplace transform the resulting differential equation to find the transfer function. The deriving is done like shown below

$$\begin{aligned}
\ddot{\tilde{p}} &= K_1 V_d \\
\ddot{\tilde{p}} &= K_1(K_{pp}(\tilde{p_c} - \tilde{p}) - K_{pd}\dot{\tilde{p}}) \\
\tilde{p}S^2 &= K_1 K_{pp}\tilde{p_c} - K_1 K_{pp}\tilde{p} - K_1 K_{pd}\dot{\tilde{p}}S \\
\tilde{p}(S^2 + K_1 K_{pp} + K_1 K_{pd}S) &= K_1 K_{pp}\dot{\tilde{p}} \\
\frac{\tilde{p}}{\tilde{p_c}} &= \frac{K_1 K_{pp}}{S^2 + K_1 K_{pd}S + K_1 K_{pp}}
\end{aligned} \tag{3.2}$$

In order to find values for K_{pp} and K_{pd}, we need to say something about the desired behaviour of the system. The assignment states that the controller should control the pitch angle rapidly to its desired value. However, it should not give rise to excessive oscillations. A possible approach to achieve this behaviour, is to use pole placement according to the Butterworth polynomials. This will give an underdamped system without oscillations, that can be somewhat slow. This can be a good basis for further manual tuning.

The normalized butterworth polynomial will tune the system to achieve the following natural resonant frequency ω_0 and damping ratio ζ:

$$\omega_0 = 1 \ [rad/sec] \tag{3.3a}$$

$$\zeta = \frac{1}{\sqrt{(2)}} \approx 0.707 \tag{3.3b}$$

By comparing the denominator of the transfer function with the second order normalized Butterworth polynomial, we can find values for K_{pp} and K_{pd}.

$$S^2 + K_1 K_{pd} S + K_1 K_{pp} = S^2 + 1.4142S + 1 \tag{3.4a}$$

$$\begin{bmatrix} K_1 K_{pd} \\ K_1 K_{pp} \end{bmatrix} = \begin{bmatrix} 1.4142 \\ 1 \end{bmatrix} \tag{3.4b}$$

$$K_1 = 0.581 \implies \begin{bmatrix} K_{pd} \\ K_{pp} \end{bmatrix} = \begin{bmatrix} 2.434 \\ 1.721 \end{bmatrix} \tag{3.4c}$$

As expected, When testing these values for K_{pp} and K_{pd}, we experienced somewhat slow response. After some more manual tuning, we found these values to give decent performance and stability:

$$\begin{bmatrix} K_{pd} \\ K_{pp} \end{bmatrix} = \begin{bmatrix} 2.4 \\ 6.1 \end{bmatrix} \tag{3.5a}$$

Using Matlab to generate a bode plot of the transfer function based on these values, we check the stability margins.

First, we notice that the gain margin is infinity. This is due to the system being modelled as a second order system, whose phase will never cross -180 degrees and go into positive feedback. However, in real world systems, non-linearities like time delays and saturations will prevent the system from having an 'infinite' gain margin.

The phase margin can be considered a "safety margin" to ensure non-oscillatory behaviour, and is read to be $63,2^\circ$. This is a good compromise, that provides decently fast step response and good stability, without too much overshoot and ringing.

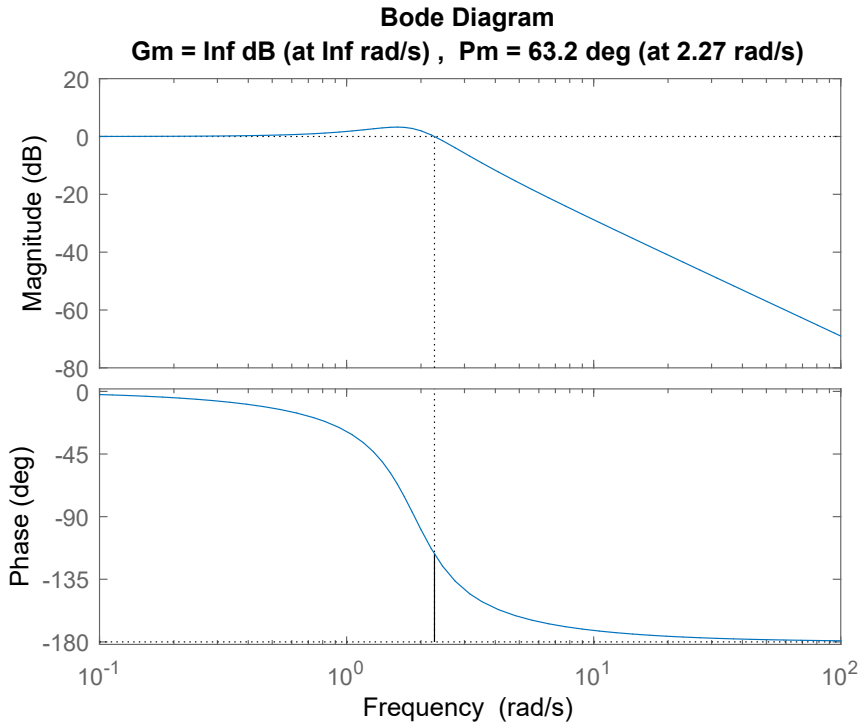

Figure 3.1: Bode plot showing the stability margins

3.1.1 Problem 2 - P controller for travel rate

The travel rate $\dot{\tilde{\lambda}}$ is now to be controlled using a simple P controller

$$\tilde{p}_c = K_{rp}(\dot{\tilde{\lambda}}_c - \dot{\tilde{\lambda}}) \tag{3.6}$$

with constant negative values of K_{rp} ($K_{rp} < 0$). Assuming that the pitch angle is controlled perfectly ($\tilde{p} = \tilde{p_c}$), the travel rate controller can be substituted into the linearized equation of motion for the travel (eq. 2.11c):

$$\ddot{\tilde{\lambda}} = K_3\tilde{p} = K_3\tilde{p}_c = K_3K_{rp}(\dot{\tilde{\lambda}}_c - \dot{\tilde{\lambda}}). \tag{3.7}$$

The Laplace Transform of the differential equation (assuming initial conditions $\dot{\tilde{\lambda}}(0) = 0$) now gives us the transfer function of the travel rate controller

$$\begin{aligned} s\dot{\tilde{\lambda}}(s) &= K_3K_{rp}(\dot{\tilde{\lambda}}_c - \dot{\tilde{\lambda}}) \\ \dot{\tilde{\lambda}}(s)(s + K_3K_{rp}) &= K_3K_{rp\dot{\tilde{\lambda}}_c} \\ \frac{\dot{\tilde{\lambda}}(s)}{\dot{\tilde{\lambda}}_c(s)} &= \frac{K_3K_{rp}}{(s + K_3K_{rp})}, \end{aligned} \tag{3.8}$$

where the constants K_3 and K_{rp} can be represented as a new constant $\rho = K_3 K_{rp}$.

The complex variable s is the same as the natural angular frequency of the system. Substituting the natural angular frequency in the denominator of the transfer function in eq. 3.8 then gives us that

$$w_{0\dot{\lambda}} = -K_3 K_{rp}. \tag{3.9}$$

The pitch controller from the previous task will form an "inner loop", while the travel rate controller will be the "outer loop" providing a reference to the pitch controller. As a fundamental rule when working with cascade control systems, the secondary process must react to the secondary controller's efforts at least three or four times faster than the primary process reacts to the primary controller. This allows the secondary controller enough time to compensate for inner loop disturbances before they can affect the primary process [https://www.controleng.com/single-article/fundamentals-of-cascade-control.html]. The relationship between the natural frequencies of the primary and secondary system (eq. 3.4b) can then be applied to find the gain of the outer loop from equation 3.9:

$$K_{rp} = -\frac{w_0}{5}\frac{1}{K_3} = -\frac{K_1 K_{pd}}{5K_3} = \frac{0.581 \cdot 6.1}{5 \cdot 0.612} \approx 1.16 \tag{3.10}$$

Chapter 4

Part 3 - Multivariable control

4.1 Problem 1 - State-space formulation

First we need to put the system of equations given by the relations for pitch and elevation in a state.space formulation.

$$\dot{x} = Ax + Bu \tag{4.1a}$$
$$y = Dx + Eu \tag{4.1b}$$

With the state and input vector

$$x = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \end{bmatrix} \text{ and } u = \begin{bmatrix} \tilde{V_s} \\ \tilde{V_d} \end{bmatrix} \tag{4.2}$$

Given these, we find that the equations 2.11a and 2.11b will be the relevant linearized system of equations for this state-space representation. To make everything a bit easier we now rename the state vector x_1, x_2 and x_3. We now are presented these equations

$$\dot{x_1} = \dot{\tilde{p}} \tag{4.3a}$$
$$\dot{x_2} = \ddot{\tilde{p}} = K_1 \tilde{V_d} \tag{4.3b}$$
$$\dot{x_3} = \ddot{\tilde{e}} = K_2 \tilde{V_s} \tag{4.3c}$$

The ambition is now to control the pitch angle $\tilde{p}$ and elevation rate $\dot{\tilde{e}}$. For this to be possible we have to measure these.

$$y_1 = \tilde{p} \tag{4.4a}$$

$$y_2 = \dot{\tilde{e}} \tag{4.4b}$$

We are now able to write the system on the state space form.

$$\dot{x} = \begin{bmatrix} \dot{\tilde{p}} \\ \ddot{\tilde{p}} \\ \ddot{\tilde{e}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & k_1 \\ k_2 & 0 \end{bmatrix} \cdot \begin{bmatrix} \tilde{V_s} \\ \tilde{V_d} \end{bmatrix} \tag{4.5}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 & 0 \\ 0 & k_1 \\ k_2 & 0 \end{bmatrix} D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} E = 0 \tag{4.6}$$

4.1.1 Problem 2 - LQR

First, we will investigate if this system is controllable[3]. It's done by analyse the row rank of the controllability matrix, which is defined as

$$C = \begin{bmatrix} B & AB & ... & A^{n-1}B \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & k_1 & 0 & 0 \\ 0 & k_1 & 0 & 0 & 0 & 0 \\ k_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.7a}$$

We can surely see that the controllability matrix 4.7a has three rows that all are linearly independent, which means that the matrix has full row rank. We can conclude that the system is controllable. Next step is to implement the controller to the system. The assignment looking for a controller on the form

$$u = Pr - Kx \tag{4.8a}$$

In the equation 4.8b, r is the reference, which in this case is the input form the joystick. The K-matrix is corresponding to the linear quadratic regulator for which the input $u = -Kx$ optimizes the cost function. Next we insert the controller equation 4.8b in the state space equation 4.1a, which gives us

$$\dot{x} = Ax + Bu = Ax + B(Pr - Kx) = (A - BK)x + BPr \tag{4.9}$$

We know that E = 0 from 4.1b, and will accordingly get the state space model for the closed-loop system.

$$\dot{x} = (A - BK)x + BPr \tag{4.10a}$$
$$y = Dx \tag{4.10b}$$

To find the K-matrix in the 4.10a we can use the $MATLAB$ - command $lqr(A, B, Q, R)$. Before using this command we need to define the weighting matrices Q and R. The parameter Q and R is used as design parameter to penalize the state variables and the control signals. If the values becomes large, you penalize the signals even more. For instance, if you choose a large R value you stabalizing the system with less energy. This type of stabalizing is called *expensive control strategy*. Equivalently, if you choose a large value for Q, then you stabalizing the system with the minimum available differences in the states. Further, a large Q will make the system less worried about big changes in the states. Since the Q and R have such a close relation, it's often a good advise to keep start with Q and R as a identity-matrices and from there on adjust them. We started out with these matrices

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \; and \tag{4.11a}$$
$$\tag{4.11b}$$
$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.11c}$$

And advanced by tuning them until we got some good results. We ended up with Q and R matrices looking like this.

$$Q = \begin{bmatrix} 120 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 70 \end{bmatrix} \; and \tag{4.12a}$$
$$\tag{4.12b}$$
$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{4.12c}$$

After finding the values of Q and R, we are now able to find P matrix such that

$$\lim_{t \to \infty} \tilde{p}(t) = \tilde{p_c} \tag{4.13a}$$
$$\lim_{t \to \infty} \dot{\tilde{e}}(t) = \dot{\tilde{e}_c} \tag{4.13b}$$

These conditions gives us

$$\dot{x}_\infty = 0 \tag{4.14a}$$
$$y_\infty = r_\infty \tag{4.14b}$$

We will then find the P-matrix

$$u = Pr - Kx \tag{4.15}$$

At this moment, we want x_∞ on the left hand side, and we carefully taking care of the conditions

$$\dot{x}_\infty = (A - BK)x_\infty + BPr = 0 \tag{4.16a}$$
$$x_\infty = (BK - A)^{-1} + BPr \tag{4.16b}$$

Afterwards we look at y_∞, and the conditions related for that equation as well.

$$y_\infty = Cx_\infty = r \tag{4.17}$$

At last we implement the r to the 4.16b, and get the final solution for the $P - matrix$

$$r = [C(BK - A)^{-1}B]Pr \tag{4.18a}$$
$$P = [C(BK - A)^{-1}B]^{-1} \tag{4.18b}$$

After finding the $P - matrix$ we found the $K - matrix$ with the $MATLAB$ - command $lqr(A, B, Q, R)$ as well. In figure 4.1.1 you can see how we implemented the LQR controller, with both P and K.

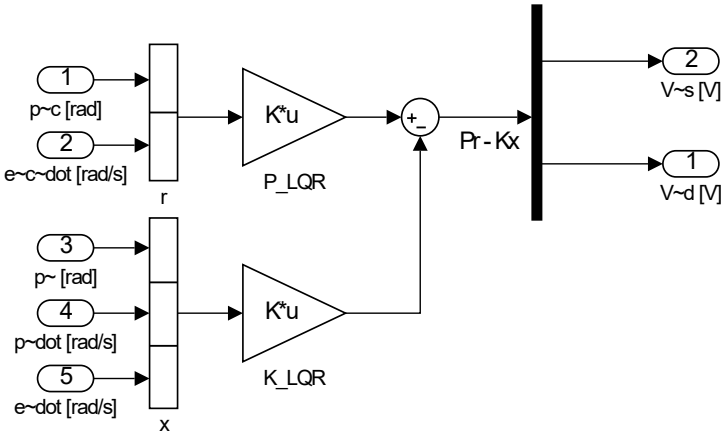

Figure 4.1: LQR controller implemented in Simulink.

We managed to find Q and R values that gave us a fast helicopter as well as an accurate one. In figure 4.2 we can a see plot of how the helicopter's pitch and elevation reacted to the joystick reference. We drove the helicopter in a certain way, such that we could analyse the graphs afterwards. First we got it in the air with 0 elevation-angle, then we drove it anti-clockwise followed by clock-wise, then we increased the elevation even more until we landed the helicopter again. Later on in the report we will be discussing the plots, and then compare it to the LQR with integral effect.

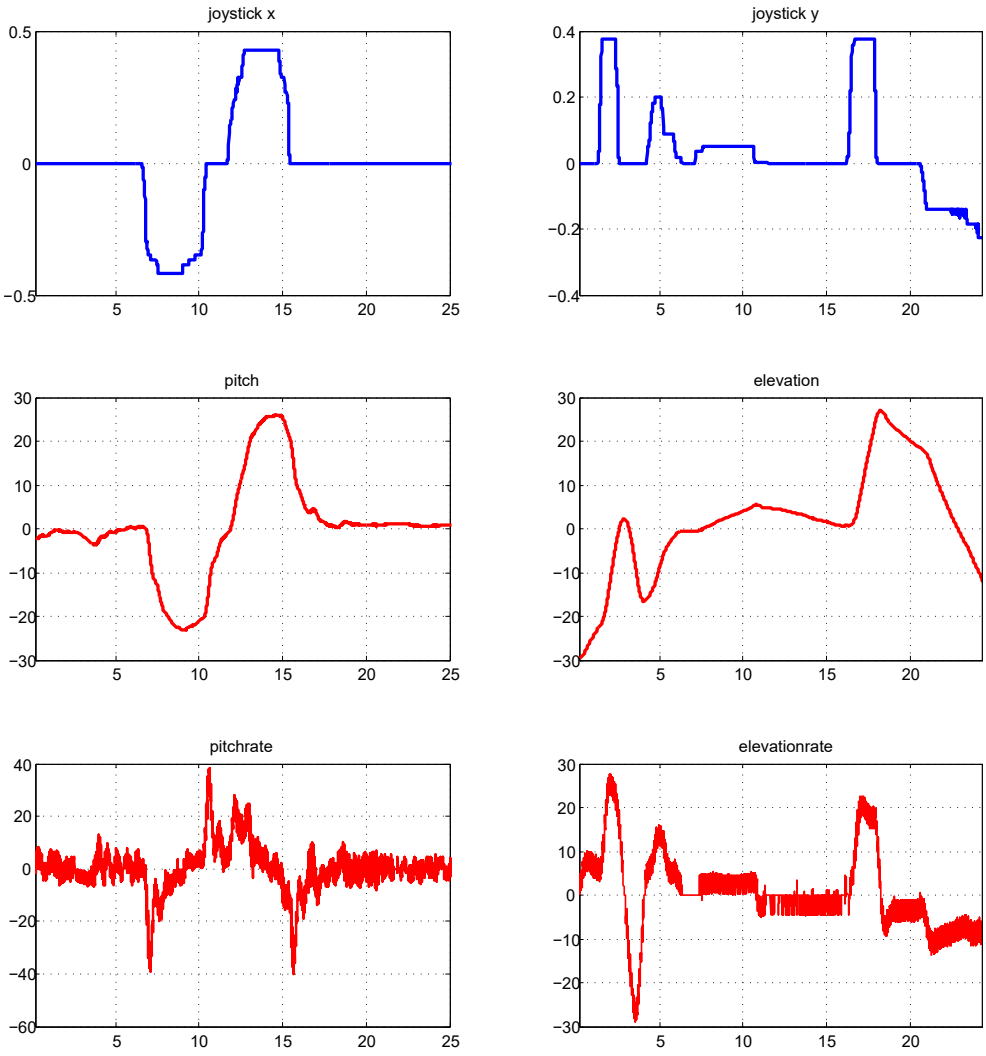

Figure 4.2: Plots showing the response of the system using the LQR controller without integral effect.

4.1.2 Problem 3 - Integral feedback effect

We now want to ensure that there are no stationary error by modifying the LQR controller, adding integral effect. This will result in two additional states γ and ζ with the differential equation for each of them given as

$$\dot{\gamma} = \tilde{p} - \tilde{p_c} \tag{4.19a}$$
$$\dot{\zeta} = \dot{\tilde{e}} - \dot{e_c} \tag{4.19b}$$

The state and input vectors are given by

$$x = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \\ \gamma \\ \zeta \end{bmatrix} \text{ and } u = \begin{bmatrix} \tilde{V_s} \\ \tilde{V_d} \end{bmatrix} \tag{4.20}$$

Since we already have been renaming the states, we did the same with these two.

$$\dot{x_4} = y_1 - r_1 = \tilde{p} - \tilde{p_c} = \dot{\gamma} \tag{4.21a}$$
$$\dot{x_5} = y_2 - r_2 = \dot{\tilde{e}} - \dot{e_c} = \dot{\zeta} \tag{4.21b}$$

With integral effect and the new states we get new matrices. The new state space matrices will look like this:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.22}$$

Yet again we had to find the weighing matrices Q and R, but now with new dimensions. We started from with Q and R as a identity-matrices and again, and found some good values at last and ended up with matrices looking like this

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 120 & 0 & 0 \\ 1 & 0 & 0 & 50 & 0 \\ 0 & 0 & 1 & 0 & 60 \end{bmatrix}, \qquad R = \begin{bmatrix} 0.3 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.23}$$

With the new Q and R matrices we used the $MATLAB$ - command $lqr(A, B, Q, R)$ again to find the new $K - matrix$. We kept the $P - matrix$ from problem 2, even though it would not effect the system.. The implemented integrator effect with the new $Q - martix$ is shown in figure 4.3

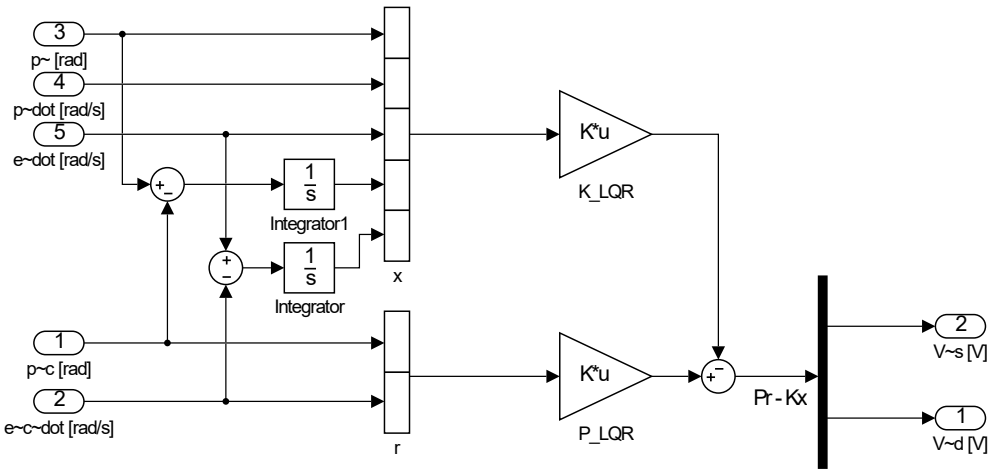

Figure 4.3: LQR controller with integral effect implemented in Simulink.

After implementing the integral effect to the LQR controller we experienced that the helicopter behaved even better. We could easily release the joystick and the helicopter would keep it's elevation and pitch angle with no deviation at all. The result is further described in the plot in figure 4.4. We drove the helicopter the same way as in the previous problem, to make the comparison easier to describe. If you look at the elevation for the LQR *without* integral effect, it's easy to see that we had to keep the engines provide lifting-force to keep it at 0 elevation angle. Over time the helicopter would have reached the 0 elevation angle, but that would take about a minute or so. Comparing it with the LQR w/integral effect, the elevation is easily held up after 3 seconds by the integral effect, not touching the joystick at all. The effect is also visible when we increase the elevation even more. For the LQR *without* integral effect, the helicopter looses elevation as soon as we release the joystick, reaching for the 0 elevation angle. As for the LQR w/integral effect it's just a small decrease of elevation, before it's stabilising again. This shows us that when we are setting the reference to zero elevationrate, the helicopter will keep the elevation at that point, as it supposed to do.
If we look at the pitch, we notice that the LQR w/integral effect oscillates when returning the pitch from clock-wise to zero, around 16 seconds. The cause of this is uncertain, but a theory is bad tuning, or integrator windup occuring at this point. Othervise the LQR *without* integral effect finds it difficult to maintain a

pitchrate around 0, while the LQR w/integral effect are way faster and stable around 0 pitch rate. We conclude that overall the integral effect is having a very positive impact on our system.

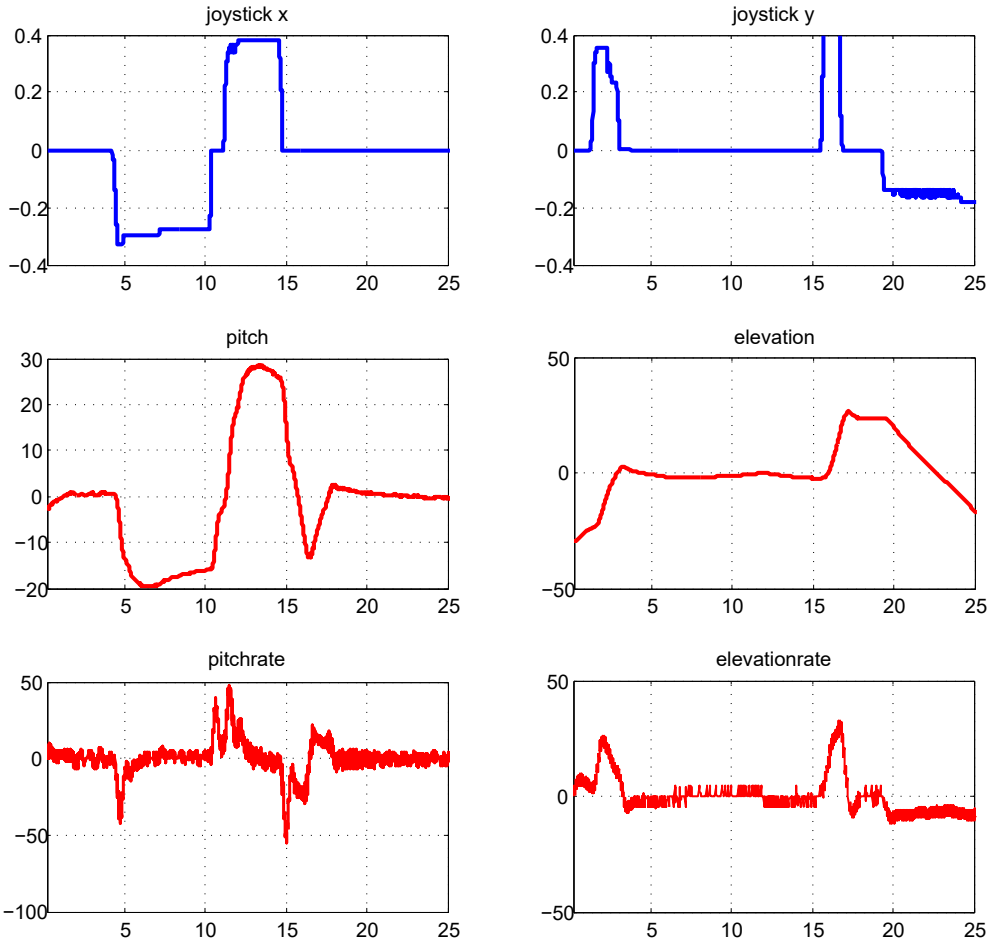

Figure 4.4:

Chapter 5

Part 4 - State estimator

The system has sensors for measuring the pitch angle, the elevation angle and the travel angle. The respective angular velocities however, have been computed using numerical differentiation. In this part, the system will be tested for observability[1] and an observer will be developed in order to estimate the omitted measurements. Circumflex over a variable denote an estimate of the variable, e.g. $\hat{x}$ is an estimate of x.

5.1 Problem 1 - State-space formulation

Given the state vector, input vector and output vector

$$x = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix}, \qquad u = \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} \qquad \text{and} \qquad y = \begin{bmatrix} \tilde{p} \\ \tilde{e} \\ \tilde{\lambda} \end{bmatrix} \tag{5.1}$$

[1] The system will be tested for observability using the *Kalman's rank condition*

From the equations of motion (eq. 2.11) we can derive a state-space formulation for the system that looks like

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ K_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} . \tag{5.2}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p} \\ \tilde{e} \\ \tilde{\lambda} \end{bmatrix}$$

5.2 Problem 2 - Examine observability

The concept of observability is said to be dual to that of controllability. Roughly speaking, while controllability studies the possibility of steering the state from the input; observability studies the possibility of estimating the state from the output. [1] For the system to be observable, its observability matrix $\mathcal{O}$ has to have full rank n, i.e. have n linearly independent rows. Using the function obsv(A,C) in MATLAB to generate the observability matrix:

$$\mathcal{O}(A,C) = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.3}$$

The generated matrix is reduced to only show the first two observability indices ($[C, CA]^T$), which already stands to show that we have full rank, which implies that the system is observable.

We can now implement a state observer, which provides an estimate of the system's internal state. The linear observer for the system can be given as

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + bu + L(y_m - \hat{y}) \\ &= A\hat{x} + bu + L(Cx + n - C\hat{x}) \\ &= (A - LC)\hat{x} + Bu + LCx + Ln, \end{aligned} \tag{5.4}$$

which has two inputs u and y, some noise n, and it produces the estimated state $\hat{x}$. We will never be able to model the system exactly, but we can define

$$e := x - \hat{x}$$

as the error between the actual state and the estimation. Differentiating e and then substituting the standard LTI state-equation and equation 5.4 into it yield

$$\begin{aligned} \dot{e} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu - (A - LC)\hat{x} - Bu - LCx - Ln \\ &= (A - LC)x - (A - LC)\hat{x} - Ln \\ &= (A - LC)(x - \hat{x}) - Ln \\ &= (A - LC)e - Ln \end{aligned} \tag{5.5}$$

Because of the *separation principle*, the observer and controller can be designed separately, i.e. the controller gain and the observer gain are computed independently. The poles of the observer is given by its closed loop system $(A - LC)$. Intuitively the dynamics of the estimator should be faster than the control loop of the plant, hence the poles of the estimator should be further into the left

complex plane than the poles of the LQR. As shown in the code snippet given underneath, the poles are found using the *place* (line 30) function, which are uniformly distributed in an arc (line 27) in the left complex plane with a radius equal to ρ (line 25) multiples of the fastest LQR pole.

```
1  % ------ Code snippet for Part 4, prolem 2. ------
2  % weighting matrices Q and R w/ estimator
3  Q_est = [35  0  0  0  0;
4           0 50  0  0  0;
5           0  0 100  0  0;
6           0  0  0 69  0;
7           0  0  0  0 60];
8  R_est = [1  0;
9           0  1];
10
11 % Calculate the LQR gains
12 K_LQR = lqr(A_PI,B_PI,Q_est,R_est)          % State feedback gain
13 K_LQR_P = K_LQR(1:2,1:3);
14 P_LQR = inv(C_P*inv(B_P*K_LQR_P-A_P)*B_P) % Reference gain
15
16 % Get the poles for the LQR fedback system
17 A_cl = A_PI - B_PI*K_LQR
18 H = B_PI*P_LQR;
19 clSys = ss(A_cl, H, C, 0);       % LQR statespace model
20 clPoles = eig(clSys)             % LQR poles
21
22 % Calculate the observer poles
23 domPole = min(abs(clPoles));
24 fastPole = max(abs(clPoles));  % Get the fastest LQR pole
25 rho = 10;                      % Multiples of the fastest LQR pole
26 pole_mag = rho*fastPole;
27 pole_phi = ((180-20):40/5:(180+20))*pi/180; % Pole angle arc
28 L_poles = pole_mag*exp(i*pole_phi)          % Complex poles
29
30 L = abs(place(A_est', C_est', L_poles)')
```

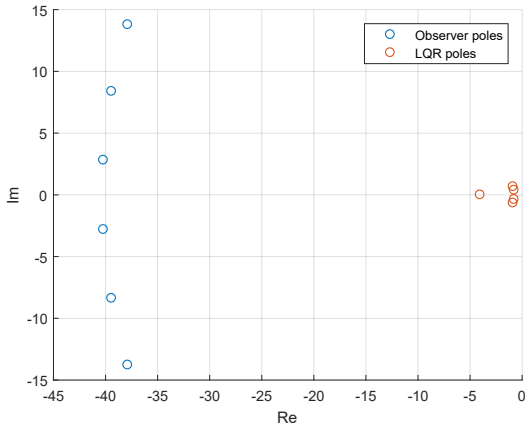

Figure 5.1: Pole-plot showing the rrelationship between the chosen estimator poles, compared to the closed-loop-system poles.

We chose to use poles ten times as fast as the fastest pole of the LQR, which gave us the poles shown in 5.2. When choosing the poles, one have to be aware of the

trade-off between a fast error correction and noise sensitivity of the observer. Higher observer (or state feedback) gain (L matrix) will make the error close quickly, but amplify disturbances and inaccuracies. As a rule of thumb, the outer loop, i.e. the observer, should be at least 2x faster than the inner loop to achieve the more likely actual state. Since the measured angles don't consist of any noise however, the speed of the estimator can be virtually as high as possible. The speed multiplier was chosen to be 10, which is more suitable for imitating a real world process. After some trial and error before achieving the given LQR weights (line 3 and 8 in the MATLAB code snippet), the generated L matrix was given by

$$L = \begin{bmatrix} 7.83 & 0.41 & 0 \\ 161.4 & 16.9 & 0 \\ 0.54 & 7.99 & 0 \\ 21.96 & 161.1 & 0 \\ 0 & 0 & 0 \\ 0.06 & 0 & 162.4 \end{bmatrix}. \tag{5.6}$$

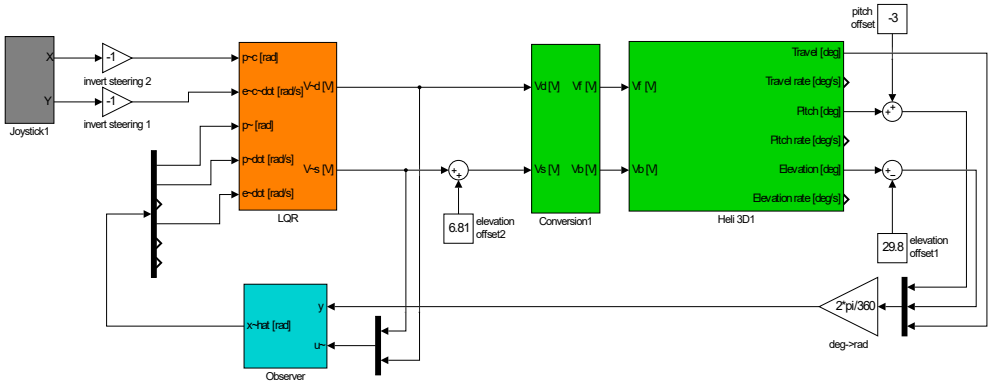

Figure 5.2: LQR controller implemented in Simulink.

The full Simulink implementation is shown in figure 5.2, which still utilises a LQR controller. The observer block itself is exposed in figure 5.2, where the transfer matrices determines the different states, controller outputs and measurements that will be included.

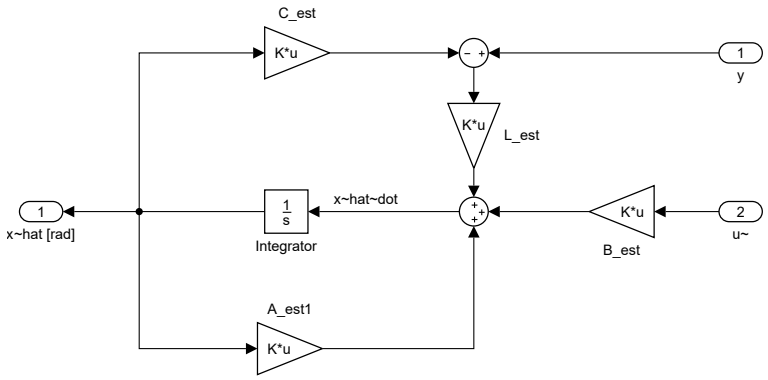

Figure 5.3: Observer block implemented in Simulink.

Figure 5.2 shows all the estimated states compared with the direct measurement data. The state estimations of the measured angles seems to be almost perfect, the angular velocity however, has a bit more error due to differentiation and other manipulations that needs to be applied. It is also worth mentioning that there is no noise from the real measurements because of low-pass filtering being performed before being made accessible. However, complications might occur as a result of the discretization, e.g. ZOH (Zero Order Hold), if the observer or LQR poles are too fast. In figure 5.2, you can see the inevitable, but minor, error produced by the observer. The small discretized steps from the tachometer, however, is smoothed out by the observer, which in return will produce a more robust LQR feedback.

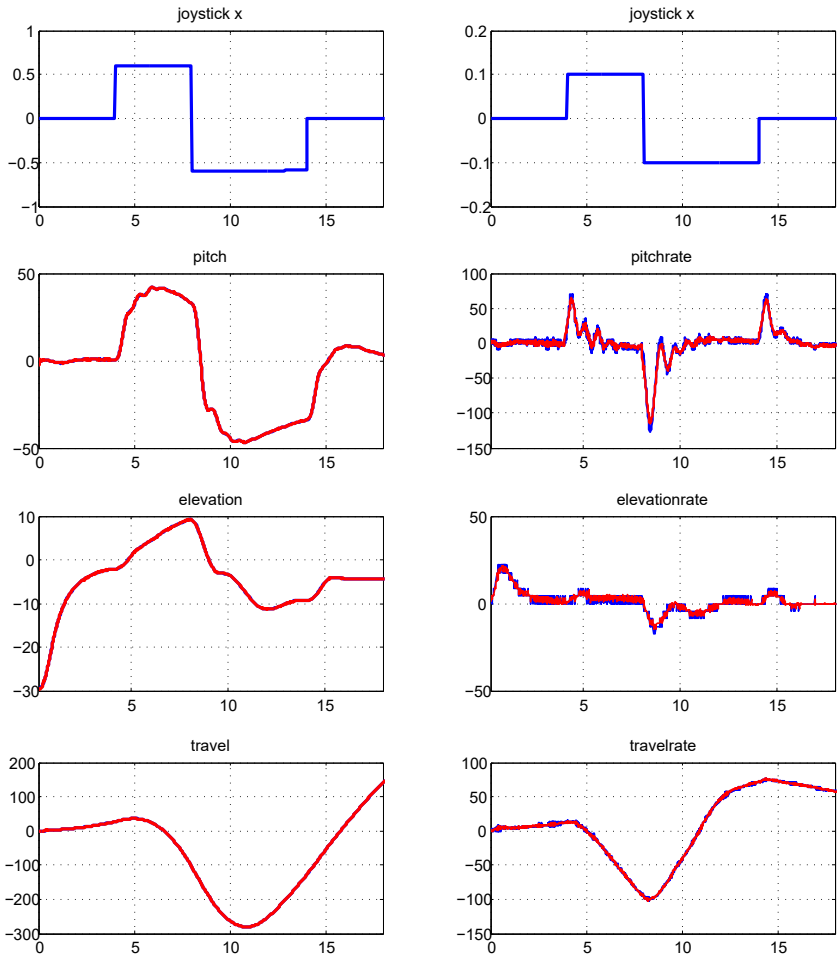

Figure 5.4: Plots showing the relation between estimated (red) and measured (blue) states.

5.3 Problem 3 - Elevation travel vs elevation pitch

This section will test if the system is observable if one only measures $\tilde{e}$ and $\tilde{\Lambda}$, but that it is not observable if one only measures $\tilde{p}$ and $\tilde{e}$.

To show that one or the other is controllable, the full observability matrices will be generated and tested for full rank. First out is the measurement of pitch and elevation, whose measurement vector y and matrix C is given by

$$y = \begin{bmatrix} \tilde{p} \\ \tilde{e} \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{5.7}$$

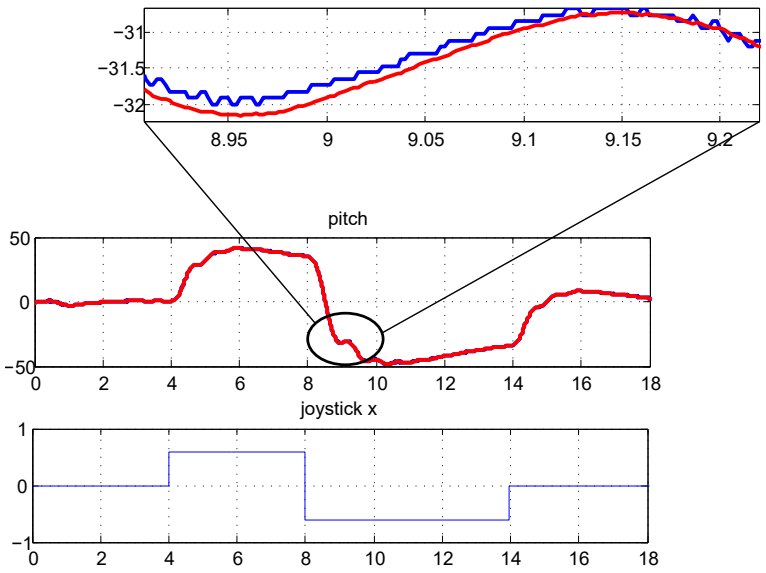

Figure 5.5: Plot showing a magnification of the pitch, indicating that the estimator (red) follows the real measurement (blue) closely.

which is used to generate the observability matrix

$$\mathcal{O}(A,C) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.8}$$

It is clear that the internal travel states cannot be deduced from the merely knowing pitch and elevation as external outputs.

Next up we will test if its possible to measure elevation and travel to be able to infer all of the internal states. Following the same procedure as for pitch and elevation, but this time with measurement vector y and matrix C given by

$$y = \begin{bmatrix} \tilde{e} \\ \tilde{\lambda} \end{bmatrix}, \qquad C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{5.9}$$

which is used to generate the observability matrix

$$\mathcal{O}(A,C) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -K_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -K_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.10}$$

Here all the internal states are linearly independent and deductable, i.e. the observability matrix has full rank. This is because the second derivative of the travel, $\ddot{\tilde{\lambda}} = -K_3\tilde{p}$, can be used to determine the pitch. However, get the pitch one would then have to differentiate the travel twice, and even worse, three times to extract the pitch velocity. All this differentiation, in conjunction with a given gain, will make this state highly susceptible to any noise or imperfections in the system.

We tried to set the L matrix to extremely low values, by making the observer poles slower compared to measuring all the angles, along with reducing LQR poles themselves. The poles were also made to be a set of real poles, instead of complex conjugated pairs, which might make the system slower, but less oscillatory. The biggest error originates from the integral effect of the reference vs state feedback error, where the observed state has a much larger value than the real measurement. In conclusion, the system is highly volatile, and there seems to be a fine line between having a stable and unstable system. Through extended trial and error, reducing the weights of the most inaccurate states, e.g. the integrated error and poles in general, we were not able to produce stable results, only slow down inflating controller output.

Try to apply fourier transform of measured signal, then maybe filter it accordingly to alleviate the noise implications. Only real-poles?

Bibliography

[1] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, Incorporated, 2014. ISBN: 9780199964543.

[2] NTNU Department of Engineering Cybernetics. *Helicopter lab assignment*. Blackboard. Accessed: 2018-09-04.

[3] Morten D. Pedersen. *TTK4115 Linear system theory, lecture notes*. Blackboard. Accessed: 2018-08-20.

[4] *Quanser*. `https://www.quanser.com/`. Accessed: 2018-09-04.