Helicopter lab report
TTK 4115

Group 53
Anders Granberg Drønnen 768686
Bendik Steinkjer Standal 757944
Filip Gornitzka Abelson 768687

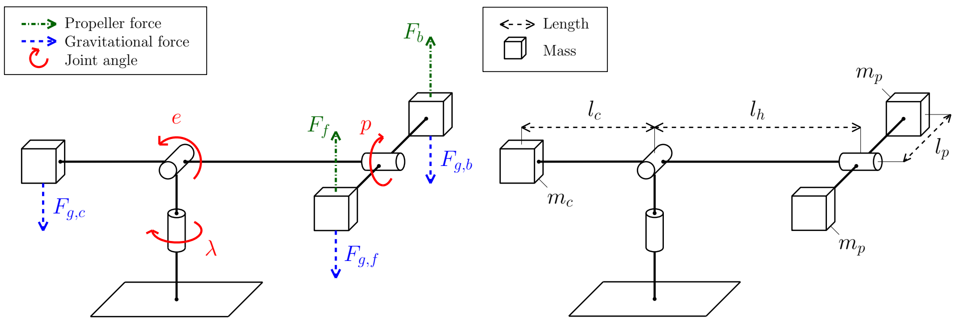
October 2017

Department of Engineering Cybernetics

Contents

1 Part I - Mathematical modeling

1.1 Problem 1 - Equations of motion



(a) Forces on the model (b) Masses and distances

Figure 1: Helicopter model, given in [1]

The goal of this problem is to derive the following equations:

$$J_p\ddot{p} = L_1 V_d \tag{1.1a}$$
$$J_e\ddot{e} = L_2\cos(e) + L_3 V_s\cos(p) \tag{1.1b}$$
$$J_\lambda\ddot{\lambda} = L_4 V_s\cos(e)\sin(p) \tag{1.1c}$$

These equations lay the foundation of what is the mathematical model of the helicopter. The force from the motors, front and back, is given by:

$$F_f = K_f V_f \tag{1.2a}$$
$$F_b = K_f V_b \tag{1.2b}$$

The motor voltages, sum and difference, are defined as:

$$V_s = V_f + V_b \tag{1.3a}$$
$$V_d = V_f - V_b \tag{1.3b}$$

The moments of inertia for pitch, elevation and travel are defined as the following:

$$J_p = 2m_p l_p^2 \tag{1.4a}$$
$$J_e = m_c l_c^2 + 2m_p l_h^2 \tag{1.4b}$$
$$J_\lambda = m_c l_c^2 + 2m_p(l_h^2 + l_p^2) \tag{1.4c}$$

By using Newton's second low for rotation, we can compute the equations of motion, in the form as stated above. We will insert equation (1.2a) and (1.2b).

$$J\alpha = \sum \tau \tag{1.5}$$

We will begin with the equation for pitch. The torques generated by gravity cancel each other out, so we end up with the torques generated by the motors.

$$\begin{aligned} J_p\ddot{p} &= l_pF_{g,b} - l_pF_{g,f} + l_pF_f - l_pF_b \\ &= l_pK_fV_f - l_pK_fV_b = l_pK_f(V_f - V_b) \\ \implies J_p\ddot{p} &= L_1V_d \end{aligned}$$

Now we will look at the equation for elevation. We must use basic trigonometry to find the correct components of the forces. As we can see, the forces are dependant on both pitch angle and elevation angle.

$$\begin{aligned} J_p\ddot{e} &= l_cF_{g,c}\cos(e) + l_h(F_f\cos(p) + F_b\cos(p) - l_hF_{g,f}\cos(e) - l_hF_{g,b}\cos(e) \\ &= l_cF_{g,f}\cos(e) + l_hK_f\cos(p)(V_f + V_b) - 2l_hm_pg\cos(e) \\ &= g(l_cm_c - 2l_hm_p)\cos(e) + l_hK_fV_s\cos(p) \\ \implies J_e\ddot{e} &= L_2\cos(e) + L_3V_s\cos(p) \end{aligned}$$

The last equation is the one for travel. Once again, we must use basic trigonometry to find the force and lever arm which are orthogonal.

$$\begin{aligned} J_\lambda\ddot{\lambda} &= \tau_f - \tau_b \\ &= -l_h\cos(e)F_f\sin(p) - l_h\cos(e)F_b\sin(p) \\ &= -l_hK_f(V_f + V_b)\cos(e)\sin(p) = -l_hK_fV_s\cos(e)\sin(p) \\ \implies J_\lambda\ddot{\lambda} &= L_4V_s\cos(e)\sin(p) \end{aligned}$$

So now we also have expressions for the constants L_{1-4}:

$$L_1 = l_pK_f \tag{1.6a}$$
$$L_2 = g(l_cm_c - 2l_hm_p) \tag{1.6b}$$
$$L_3 = l_hK_f \tag{1.6c}$$
$$L_4 = -L_3 = -l_hK_f \tag{1.6d}$$

1.2 Problem 2 - Linearization

To be able to control our helicopter, we must linearize the equation of motion. We will linearize the model around the point $(p, e, \lambda)^T = (p^*, e^*, \lambda^*)^T$ with $p^* = e^* = \lambda^* = 0$. Now we determine the voltages V_s^* and V_d^*, such that for all time $\dot{p} = \dot{e} = \dot{\lambda} = 0$. This implies that $\ddot{p} = \ddot{e} = \ddot{\lambda} = 0$ for all time.

Equation (1.1a) yields:

$$J_p \cdot 0 = L_1V_d \implies V_d^* = 0 \tag{1.7}$$

Equation (1.1b) yields:

$$\begin{aligned} J_e \cdot 0 &= L_2 \cos(0) + L_3 V_s^* \cos(0) \\ L_3 V_s^* &= -L_2 \implies V_s^* = -\frac{L_2}{L_3} \end{aligned} \tag{1.8}$$

We use the following coordinate transformation to linearize and rewrite the equations of motion.

$$\begin{bmatrix} \tilde{p} \\ \tilde{e} \\ \tilde{\lambda} \end{bmatrix} = \begin{bmatrix} p \\ e \\ \lambda \end{bmatrix} - \begin{bmatrix} p^* \\ e^* \\ \lambda^* \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \tilde{V_s} \\ \tilde{V_d} \end{bmatrix} = \begin{bmatrix} V_s \\ V_d \end{bmatrix} - \begin{bmatrix} V_s^* \\ V_d^* \end{bmatrix} \tag{1.9}$$

This gives us new expressions for the equations of motion.

$$\ddot{\tilde{p}} = \frac{L_1}{J_p} \tilde{V_d} \tag{1.10a}$$

$$\ddot{\tilde{e}} = \frac{L_2}{J_e} \cos(\tilde{e}) + \frac{L_3}{J_e} (\tilde{V_s} - \frac{L_2}{L_3}) \cos(\tilde{p}) \tag{1.10b}$$

$$\ddot{\tilde{\lambda}} = \frac{L_4}{J_\lambda} (\tilde{V_s} - \frac{L_2}{L_3}) \cos(\tilde{e}) \sin(\tilde{p}) \tag{1.10c}$$

We set up a state-space model on the following form and linearize the model around $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{u}_0 = (0, -\frac{L_2}{L_3})^T$.

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

Where,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_6 \end{bmatrix} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix} \quad \text{and } \mathbf{u} = \begin{bmatrix} \tilde{u_1} \\ \tilde{u_2} \end{bmatrix} = \begin{bmatrix} \tilde{V_s} \\ \tilde{V_d} \end{bmatrix}$$

Using equation (1.10a), (1.10b) and (1.10c) we get the following expression for $\dot{\mathbf{x}}$.

$$\dot{\mathbf{x}} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{L_1}{J_p} u_2 \\ \frac{L_2}{J_e} \cos(x_3) + \frac{L_3}{J_e} u_1 \cos(x_1) \\ x_6 \\ \frac{L_4}{J_\lambda} u_1 \cos(x_3) \sin(x_1) \end{bmatrix}$$

Now we can find the matrices $\mathbf{A}$ and $\mathbf{B}$.

$$\mathbf{A} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial x_1} & \cdots & \frac{\partial f_6}{\partial x_6} \end{bmatrix} \Bigg|_{\mathbf{x_0}, \mathbf{u_0}}, \quad \mathbf{B} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_6}{\partial u_1} & \frac{\partial f_6}{\partial u_2} \end{bmatrix} \Bigg|_{\mathbf{x_0}, \mathbf{u_0}}$$

Calculating the partial derivatives and inserting for the points we are linearizing about, we get

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{L_2 L_4}{J_\lambda L_3} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & \frac{L_1}{J_p} \\ 0 & 0 \\ \frac{L_3}{J_e} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{1.11}$$

And this gives us the linearized equations:

$$\ddot{\tilde{p}} = K_1 \tilde{V}_d \tag{1.12a}$$

$$\ddot{\tilde{e}} = K_2 \tilde{V}_s \tag{1.12b}$$

$$\ddot{\tilde{\lambda}} = K_3 \tilde{p} \tag{1.12c}$$

where

$$K_1 = \frac{L_1}{J_p}, K_2 = \frac{L_3}{J_e} \text{ and } K_3 = -\frac{L_2 L_4}{L_3 J_\lambda} \tag{1.13}$$

1.3 Problem 3 - Feed forward control

Attempting to control the helicopter using feed forward proved to be a challenge, just as expected. We added gains to the joystick output in order to make the response easier to handle. We observed that the physical behaviour of the helicopter resembled the mathematical modelling quite well. However, they will never be perfect, as there exists physical parameters that was not included in the model, e.g. drag and friction. This can explain possible discrepancies in the model and real world.

1.4 Problem 4 - Helicopter at equilibrium

Now we want to determine the motor voltage constant, K_f, by measuring the value of $V_s = V_s^*$ which makes the helicopter maintain the equilibrium value $e = e^* = 0$. We measured the voltage to be $V_s = V_s^* = 6.75$ V.
From equation (1.8), we have that $V_s^* = -\frac{L_2}{L_3}$. Furthermore, we can expand this expression using equation (1.6b) and equation (1.6c), then solve for K_f and finally insert the constant values and the measured value for V_s^*.

$$\begin{aligned} V_s^* &= -\frac{L_2}{L_3} = \frac{g(2l_h m_p - l_c m_c)}{l_h K_f} \\ \implies K_f &= \frac{g(2l_h m_p - l_c m_c)}{l_h V_s^*} = 0.148 \end{aligned} \tag{1.14}$$

2 Part II - Monovariable control

2.1 Problem 1 - Pitch angle PD controller

A PD controller is to be added to control the pitch angle p. This controller is given as:

$$\tilde{V}_d = K_{pp}(\tilde{p}_c - \tilde{p}) - K_{pd}\dot{\tilde{p}} \tag{2.1}$$

From equation (1.12a) we see that $\tilde{V}_d = \frac{\ddot{\tilde{p}}}{K_1}$. Substituting this into the equation for the PD controller (2.1) gives:

$$\begin{aligned}\ddot{\tilde{p}} &= K_1(K_{pp}(\tilde{p}_c - \tilde{p}) - K_{pd}\dot{\tilde{p}} \\ \ddot{\tilde{p}} + K_1K_{pd}\dot{\tilde{p}} + K_1K_{pp}\tilde{p} &= K_1K_{pp}\tilde{p}_c\end{aligned}$$

Using the Laplace transform and assuming $\tilde{p}(0) = \dot{\tilde{p}}(0) = 0$, gives us the following transfer function:

$$\begin{aligned}s^2\tilde{p}(s) + sK_1K_{pd}\tilde{p}(s) + K_1K_{pp}\tilde{p}(s) &= K_1K_{pp}\tilde{p}_c(s) \\ \tilde{p}(s)(s^2 + sK_1K_{pd} + K_1K_{pp}) &= K_1K_{pp}\tilde{p}_c(s) \\ \frac{\tilde{p}(s)}{\tilde{p}_c(s)} &= \frac{K_1K_{pp}}{s^2 + K_1K_{pd}s + K_1K_{pp}} \qquad (2.2)\end{aligned}$$

The linearized pitch dynamics can be regarded as a second-order linear system, generally given by the transfer function:

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

We wish to obtain critical damping, in order to get the system to return to its equilibrium in the minimum amount of time without overshooting. Hence, we set $\zeta = 1$. The general transfer function for this type of system is given by:

$$h(s) = \frac{K\omega_0^2}{s^2 + 2\omega_0 s + \omega_0^2} \tag{2.3}$$

Comparing the transfer function of our system, equation (2.2), to the general one, equation (2.3), gives the following equations for the regulator parameters (with K = 1):

$$K_{pp} = \frac{\omega_0^2}{K_1} \text{ and } K_{pd} = 2\frac{\omega_0}{K_1} \tag{2.4}$$

Now, we need to tune the controller gains K_{pp} and K_{pd}. Our goal is to make the controller regulate the pitch angle rapidly to its desired value, without excessive oscillations. The natural frequency ω_o decides the rapidness of the pitch regulation. Increasing ω_o will make the controller react to pitch changes faster. This is because we move the poles of the system further into the left plane, seen by examining the the denominator of equation (2.3):

$$(s + \omega_0)^2$$

Looking at the expressions for K_{pp} and K_{pd}, given in equation (2.4), we see that they're both dependant of ω_0. Unsurprisingly, this means that tuning of the controller gains influences the closed-loop eigenvalues and the pitch response. A higher value for K_{pp} for instance, will move the eigenvalues to the left in the complex plane, and give faster pitch response. However, there is a limit to how much the controller gains, and thus the natural frequency, may be increased. Increasing it too much may damage the helicopter, or make it unstable. Through a process starting with initial guesstimates, followed by tuning by trial and error, we found values that gave us a satisfying response:

$$K_{pp} = 14$$
$$K_{pd} = 9.765$$

After including the PD controller for the pitch angle, we experienced that the helicopter was easier to control than it was when only feed forward was used. This was expected. The Matlab script for this problem can be found in appendix A.1. The Simulink model can be found in appendix B fig. 8 and fig. 9.

2.2 Problem 2 - Travel rate P controller

We now aim to control the travel rate $\dot{\tilde{\lambda}}$ using a simple P controller:

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

where $K_{rp} < 0$. We now assume that the pitch angle is controlled perfectly, that is $\tilde{p} = \tilde{p}_c$. Substituting equation (1.12c) into equation (2.2) for the P controller gives:

$$\ddot{\tilde{\lambda}} = K_3(K_{rp}(\dot{\tilde{\lambda}}_c - \dot{\tilde{\lambda}}))$$
$$\ddot{\tilde{\lambda}} + K_3K_{rp}\dot{\tilde{\lambda}} = K_3K_{rp}\dot{\tilde{\lambda}}_c$$

Using the Laplace transform and assuming $\tilde{\lambda}(0) = \dot{\tilde{\lambda}}(0) = 0$, gives us the transfer function from the reference $\dot{\tilde{\lambda}}_c$ to the travel rate $\dot{\tilde{\lambda}}$:

$$s\dot{\tilde{\lambda}}(s) + K_3K_{rp}\dot{\tilde{\lambda}}(s) = K_3K_{rp}\dot{\tilde{\lambda}}_c(s)$$
$$\dot{\tilde{\lambda}}(s)(s + K_3K_{rp}) = K_3K_{rp}\dot{\tilde{\lambda}}_c(s)$$
$$\frac{\dot{\tilde{\lambda}}(s)}{\dot{\tilde{\lambda}}_c(s)} = \frac{K_3K_{rp}}{s + K_3K_{rp}}$$

The transfer function can be written as:

$$\frac{\dot{\tilde{\lambda}}(s)}{\dot{\tilde{\lambda}}_c(s)} = \frac{\rho}{s+\rho}$$

where $\rho = K_3 K_{rp}$. We tested the helicopters response for different values for K_{rp}, and found that $K_{rp} = -1.2$ gave a fast and accurate response. The Matlab script for this problem can be found in appendix A.1. The Simulink model can be found in appendix B fig. 10 and fig. 11.

3 Part III - Multivariable control

3.1 Problem 1 - State-Space Representation

We put the system in a state-space formulation on the form:

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

where A and B are matrices, and our state and input vector are:

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

This gives us the following state-space model:

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

3.2 Problem 2 - Linear Quadratic Regulator

We now aim to track the reference $\mathbf{r} = \begin{bmatrix} \tilde{p_c}, & \dot{\tilde{e_c}} \end{bmatrix}^T$ for the pitch angle $\tilde{p}$ and elevation rate $\dot{\tilde{e}}$. The reference values $\tilde{p}_c$ and $\dot{\tilde{e}}_c$ are given by the joystick, the x-axis and y-axis respectively. Firstly, we examine if the system is controllable. The controllability matrix is given by:

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & K_1 & 0 & 0 \\ 0 & K_1 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We see that $\mathcal{C}$ has full rank, which means that the system is controllable. In order to actually control the system, we implement a controller of the form:

$$\mathbf{u} = \mathbf{Pr} - \mathbf{Kx}$$

In the controller, $\mathbf{r}$ is the reference given by the joystick output. The matrix $\mathbf{K}$ corresponds to the linear quadratic regulator (LQR) for which the control input $\mathbf{u} = -\mathbf{Kx}$ optimizes the cost function

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

The matrix $\mathbf{K}$ is obtained by using the MATLAB command K = lqr(A,B,Q,R). $\mathbf{Q}$ and $\mathbf{R}$ are weighting matrices, used as design parameters.

For simplicity, $\mathbf{Q}$ and $\mathbf{R}$ are diagonal. Obtaining a fast and accurate response was our main focus when deciding their values. In order to achieve this, we needed to penalize the deviation of the output more. This is done by increasing the elements on the diagonal of **Q**, which correspond to the states. Choosing large values for **Q** means trying to stabilize the system with the least possible changes in the states, which gives a fast response.

When deciding the elements of **Q** and **R**, we started with an initial guess. With our goal of obtaining a fast convergence of the output in mind, we tuned from there. This design is evident with the elements of **Q** being high compared to the elements of **R**:

$$\mathbf{Q} = \begin{bmatrix} 91.2 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 100 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Our state-space model with the controller is

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{B}(\mathbf{Pr} - \mathbf{Kx})$$

We want to obtain a solution such that $\mathbf{y}(t) \rightarrow \mathbf{r}$ as t $\rightarrow \infty$. When the output gets stable, $\dot{\mathbf{x}} \rightarrow 0$. We get:

$$\begin{aligned} 0 &= \mathbf{Ax}_\infty + \mathbf{B}(\mathbf{Pr} - \mathbf{Kx}_\infty) \\ (\mathbf{A} - \mathbf{BK})\mathbf{x}_\infty &= -\mathbf{BPr} \\ \mathbf{x}_\infty &= (\mathbf{BK} - \mathbf{A})^{-1}\mathbf{BPr} \end{aligned}$$

Substituting this into $\mathbf{y}_\infty = \mathbf{Cx}_\infty$ yields:

$$\mathbf{y}_\infty = \left[\mathbf{C}(\mathbf{BK} - \mathbf{A})^{-1}\mathbf{B}\right]\mathbf{Pr}$$

which gives:

$$\mathbf{P} = \left[\mathbf{C}(\mathbf{BK} - \mathbf{A})^{-1}\mathbf{B}\right]^{-1}$$

Our **K** and **P** matrices:

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 10 \\ 9.5499 & 9.0841 & 0 \end{bmatrix}, \mathbf{P} = \begin{bmatrix} 0 & 10 \\ 9.5499 & 0 \end{bmatrix}$$

The Matlab script for this problem can be found in appendix A.2. The Simulink model can be found in appendix B fig. 12 and fig. 13.

3.3 Problem 3 - PI controller

We now modify the controller to include an integral effect for the elevation rate and the pitch angle. Including an integral effect makes our controller a

PI controller, and results in two additional states. We call the new states γ and ζ, and their differential equations are given by:

$$\dot{\gamma} = \tilde{p} - \tilde{p}_c$$
$$\dot{\zeta} = \dot{\tilde{e}} - \dot{\tilde{e}}_c$$

The state vector and the input vector are now given by:

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

Adding an integral effect changes the dimensions of matrices **A**, **B**, **Q** and **K**. Using **Q** and **R** from the previous problem as an initial guess, we tuned the new weighting matrices with the same objective as before. **K** were then found from K = lqr(A,B,Q,R):

$$\mathbf{Q} = \begin{bmatrix} 91.2 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 50 \end{bmatrix}, \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 15.7999 & 0 & 7.0711 \\ 15.3004 & 10.1047 & 0 & 7.0711 & 0 \end{bmatrix}$$

We used the same matrix **P** as in problem 2.

The step response of the helicopter without and with integral effect can be seen in fig. 2 and fig. 3 respectively. We see that the difference in elevation is clear. With the integral effect, the helicopter is able to maintain the reference position. This is a result of the added integrating states.

The Matlab script for this problem can be found in appendix A.2. The Simulink model can be found in appendix B fig. 14 and fig. 15.

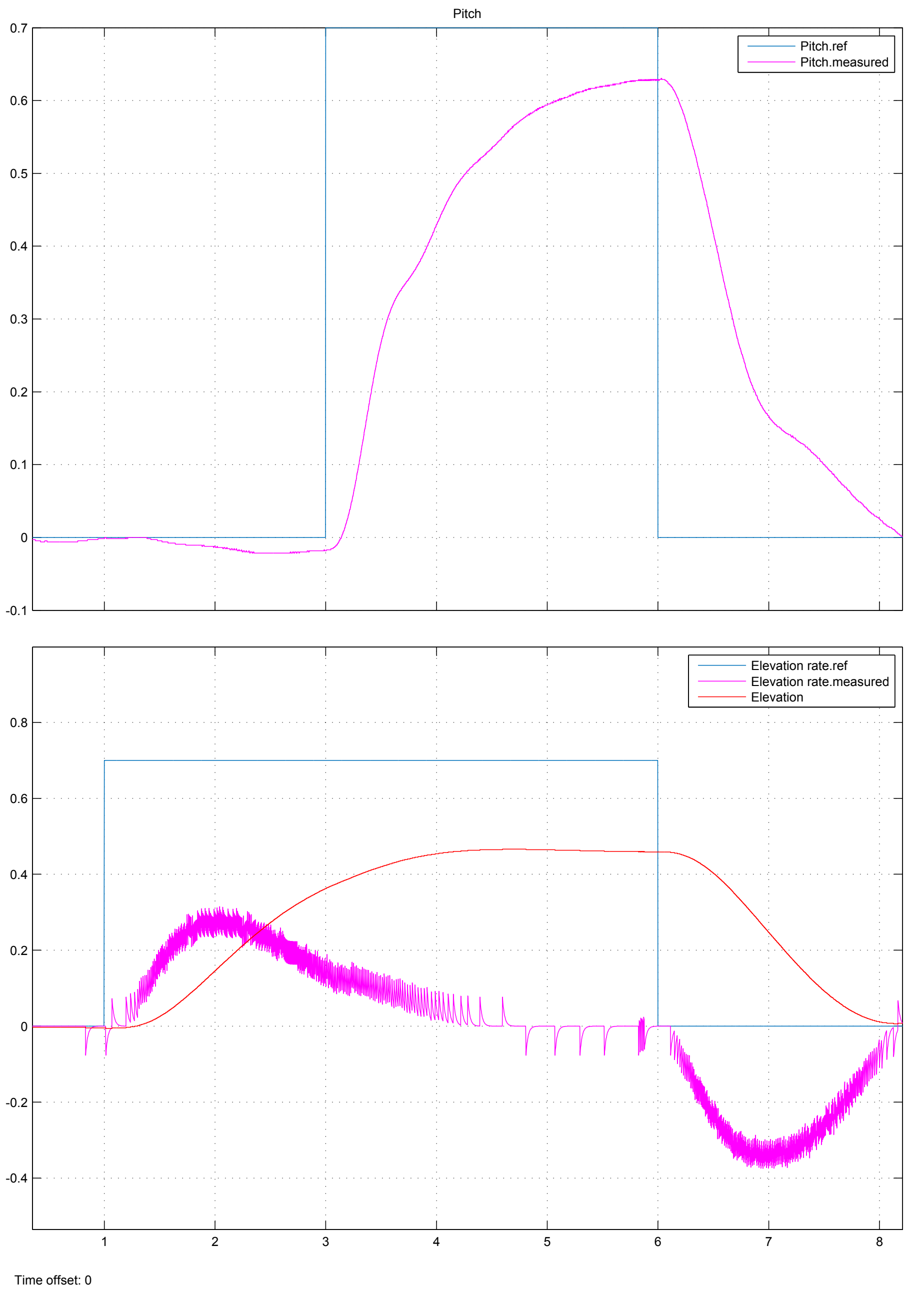


Figure 2: Helicopter response without integral effect, pitch on top and elevation below

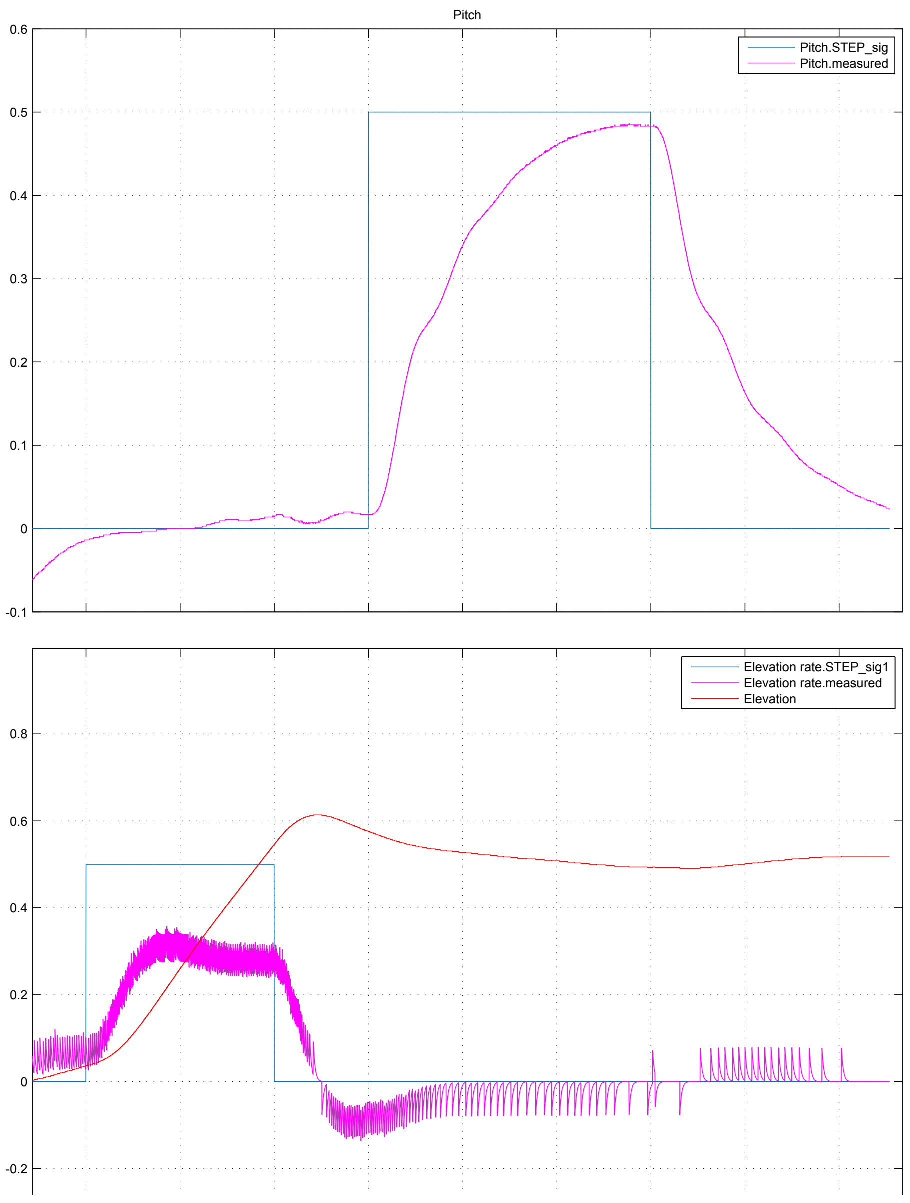


Figure 3: Helicopter response with integral effect, pitch on top and elevation below

4 Part IV - State Estimation

4.1 Problem 1 - State-space formulation

We wish to derive a state-space formulation from the system of linearized equations derived in 1.2 on the form

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu}\\ \mathbf{y} &= \mathbf{Cx}\end{aligned} \tag{4.1}$$

where **A, B** and **C** are matrices defined below. The state vector, the input vector and the output vector are now given by:

$$\mathbf{x} = \begin{bmatrix}\tilde{p}\\ \dot{\tilde{p}}\\ \tilde{e}\\ \dot{\tilde{e}}\\ \tilde{\lambda}\\ \dot{\tilde{\lambda}}\end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix}\tilde{V}_s\\ \tilde{V}_d\end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix}\tilde{p}\\ \tilde{e}\\ \tilde{\lambda}\end{bmatrix} \tag{4.2}$$

From the output equation and the state vector, we can directly see that the matrix **C** is given by

$$\mathbf{C} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix} \tag{4.3}$$

Furthermore, using the equations (1.12a)-(1.12c), the state equation can be written as

$$\dot{\mathbf{x}} = \begin{bmatrix}\dot{\tilde{p}}\\ \ddot{\tilde{p}}\\ \dot{\tilde{e}}\\ \ddot{\tilde{e}}\\ \dot{\tilde{\lambda}}\\ \ddot{\tilde{\lambda}}\end{bmatrix} = \begin{bmatrix}x_2\\ K_1\tilde{V}_d\\ x_4\\ K_2\tilde{V}_s\\ x_6\\ K_3x_1\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ K_3 & 0 & 0 & 0 & 0 & 0\end{bmatrix}}_{\mathbf{A}}\mathbf{x} + \underbrace{\begin{bmatrix}0 & 0\\ 0 & K_1\\ 0 & 0\\ K_2 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}}_{\mathbf{B}}\mathbf{u} \tag{4.4}$$

4.2 Problem 2 - Linear observer

In order to examine the observability of the system, we calculate the observability matrix using

$$\mathcal{O} = \begin{bmatrix}\mathbf{C}\\ \mathbf{CA}\\ \mathbf{CA^2}\\ \vdots\\ \mathbf{CA^{n-1}}\end{bmatrix} \tag{4.5}$$

We have learned from [2] that the criteria for a system to be observable is that equation (4.5) has full rank. In our case, this is achieved after the two first computations in the observability matrix:

$$\mathcal{O}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

Hence, our system is indeed observable.

Next, we wish to create a linear observer for the system of the form:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y} - \mathbf{C}\hat{\mathbf{x}}) \tag{4.7}$$

where $\mathbf{L}$ is the observer gain matrix. Because the system is observable, we have the opportunity to place the poles of the system where we want in order to obtain an optimal response. We can do this by choosing an appropriate gain matrix $\mathbf{L}$. The estimator $\hat{\mathbf{x}}$ has the same poles as $\mathbf{A} - \mathbf{LC}$. Generally, we want the error dynamics to be faster than the system itself. In practice, this means that the poles of $\mathbf{A} - \mathbf{LC}$ should be placed further into the left half plane than the poles of $\mathbf{A} - \mathbf{BK}$, which represents the system dynamics. We place the estimator poles on a circular arc in the left half plane, shown in fig. 4. The radius of this arc is determined by a multiple of the distance to the most negative pole of the original system. MATLAB provides a simple function place() to compute the observer gain matrix $\mathbf{L}$. MATLAB code for implementing this is shown in appendix A.3.
Choosing a ideal gain matrix will be very important for this system, because although we want a fast estimator, the further into the left half plane the estimator poles are located, the more we will amplify noise in the measurements. In our project, we ended up placing the estimator poles on an arc with radius 13 times the distance of the most negative pole from the original system.

As in part 3, we tried tuning with different values for the matrices $\mathbf{Q}$ and $\mathbf{R}$ until the desired behaviour was reached. We seemed however to get the best results with similar values as in part III, see eq. (3.1). With the computed gain matrix

$$\mathbf{L} = \begin{bmatrix} 97,94 & 6,20 & -13,90 \\ 2476,40 & 318,65 & -725,98 \\ -2,47 & 98,99 & 1,78 \\ -114,14 & 2600,04 & 80,90 \\ 15,36 & 2,71 & 96,90 \\ 786,98 & -148,13 & 2412,86 \end{bmatrix} \tag{4.8}$$

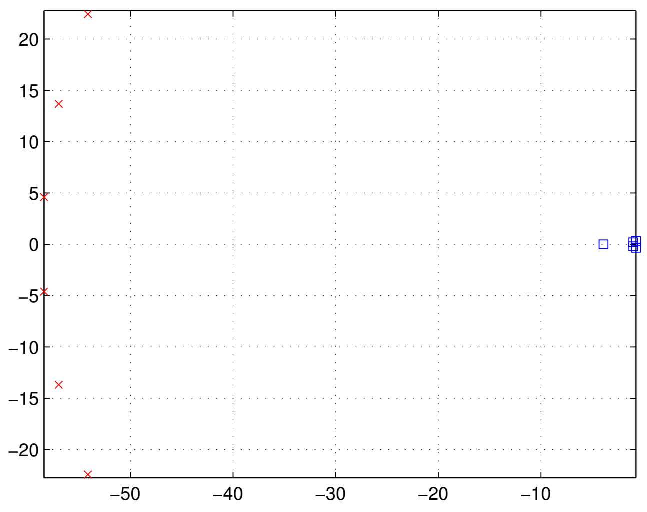


Figure 4: Pole placement for the system with integral controller from part III. The system poles are located in blue on the right, while the observer poles are seen in red.

we were able to reach satisfying results with both controllers from part III. Plots of the estimated values versus the real values for both controllers are shown below, first for the regular P controller in fig. 5 and then the PI controller in fig. 6. The Simulink-diagram to show the implementation of the observer is included in fig. 16, appendix B

4.3 Problem 3 - Observer without pitch

We aim to control our system by only measuring $\tilde{e}$ and $\tilde{\lambda}$. The output $\mathbf{y}$ and output matrix $\mathbf{C}$ becomes:

$$\mathbf{y} = \begin{bmatrix} \tilde{e} \\ \tilde{\lambda} \end{bmatrix} \tag{4.9}$$

and

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.10}$$

Again, as in 4.2, we use equation (4.5) to examine if the system is observable:

$$\mathcal{O} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -K_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -K_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.11}$$

We can clearly see that equation (4.11) has full rank, so the system is observable when measuring only $\tilde{e}$ and $\tilde{\lambda}$. However, if one only measures $\tilde{p}$ and $\tilde{e}$, this is not the case! The output matrix becomes:

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

which leads to the observability matrix

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Now, the observability matrix has rank 4, which is not full rank. Hence the system is not observable and it will be impossible to control with a state estimator. This is due to the fact that the pitch, $\tilde{p}$, can be found by differentiating $\tilde{\lambda}$ twice and multiplying with a constant K_3, (1.12c). However, this is not the case when we try to replace $\tilde{\lambda}$ with $\tilde{p}$ as a measured state. If we try to integrate $\tilde{\lambda}$ twice in order to find the pitch, we will end up with unknown constants, and thus information will be lost.

When testing the helicopter with equation (4.9) as the measured states, it turned out to be difficult to acquire a good response. Even though it is possible in theory to control the system with only these two measured states, it did not work well in practice, as we can see from fig. 7. This can be explained by the fact that we differentiate $\tilde{\lambda}$ three times to get the pitch rate $\dot{\tilde{p}}$. During this action we also differentiate measurement noise, which results in a significant amplification of this noise. After some testing, we chose the estimator poles to be on the real axis, and much lower values of L compared to equation (4.8):

$$\mathbf{L} = \begin{bmatrix} 1,95 & -90,72 \\ 1,19 & -44,14 \\ 6,50 & -0,01 \\ 10,00 & -0,012 \\ -0,07 & 10,50 \\ -0,54 & 38,00 \end{bmatrix}$$

This will lead to a slower observer, but less amplification of noise, and by placing poles on the real axis, we achieve as much damping as possible. After tuning of the weighting matrices, we found a somewhat acceptable behaviour when reducing the integral effect by giving the last two diagonal elements of the weighting matrix **Q** small values, and making power less available by increasing values of **R**. We ended up with the following matrices:

$$\mathbf{Q} = \begin{bmatrix} 8 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix},$$

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 0.3149 & 0 & 0.0032 \\ 0.1090 & 0.6124 & 0 & 0.0032 & 0 \end{bmatrix}.$$

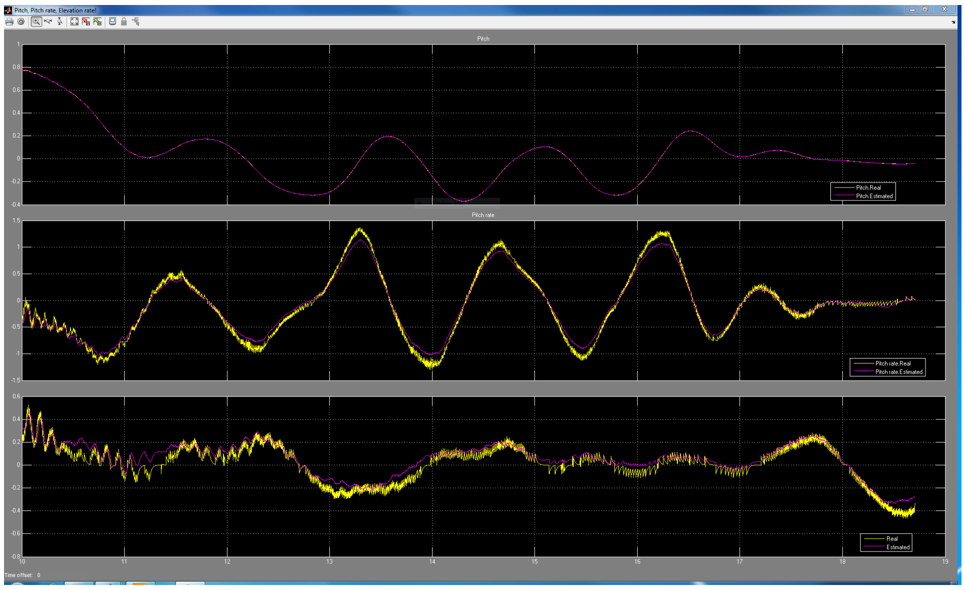


Figure 5: Part IV, problem 2 - P controller. Shows the real (yellow) and estimated (purple) states for pitch, pitch rate and elevation rate respectively. As we can see, the estimated states follows the real states quite well.

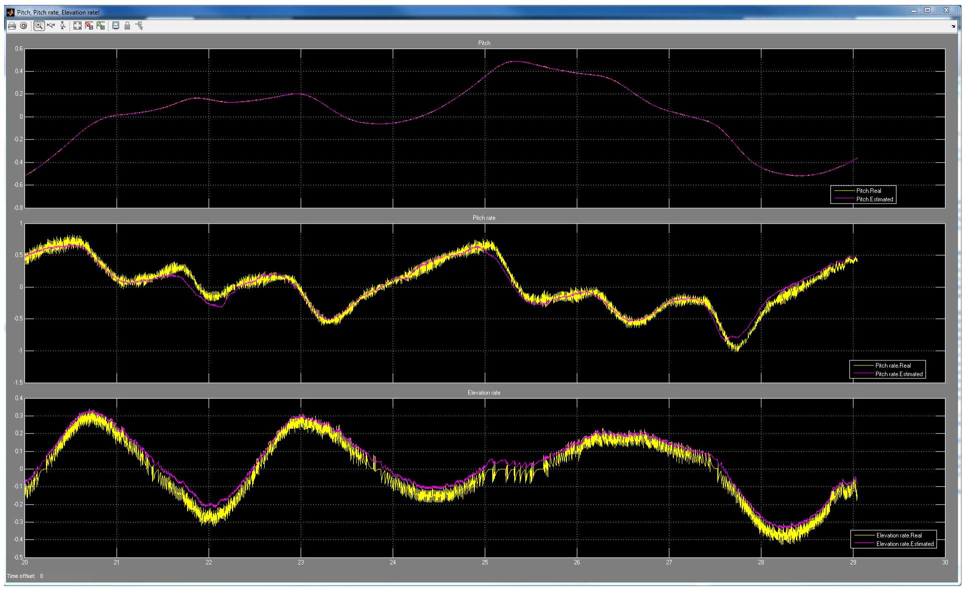


Figure 6: Part IV, problem 2 - PI controller. Shows the real (yellow) and estimated (purple) states for pitch, pitch rate and elevation rate respectively. Again, as we can see, the estimated states follows the real states quite well.

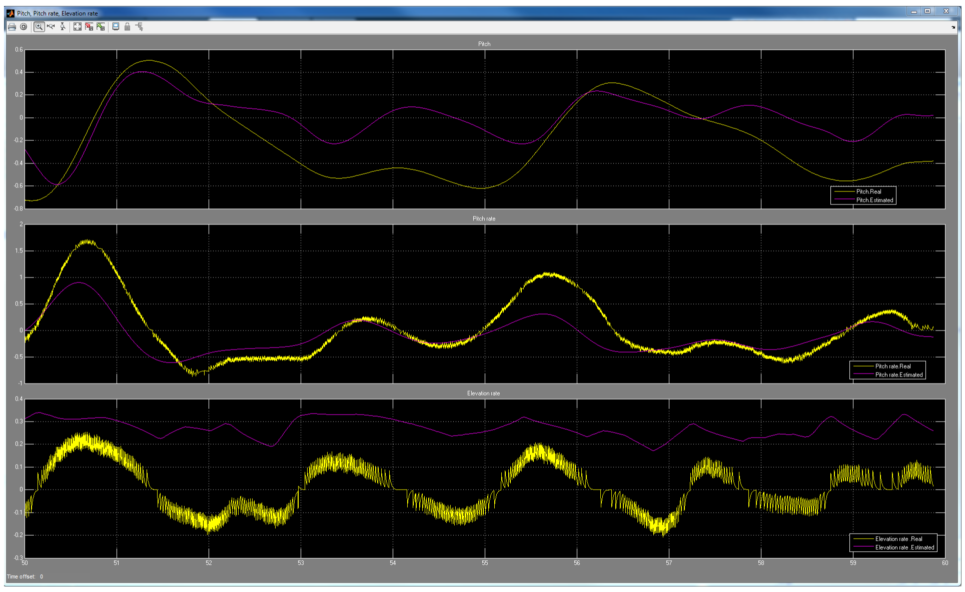


Figure 7: Part IV, problem 3. Shows the real (yellow) and estimated (purple) states for pitch, pitch rate and elevation rate respectively. Clearly, this is not an optimal result as the real and estimated states differ quite a bit.

A MATLAB Code

This section contains our MATLAB code, but only the relevant and changed parts for each problem.

A.1 Part II

```
%% CALCULATED VALUES
%Part II

K_f = 0.148;
L_1 = l_p * K_f;
L_2 = g*(l_c * m_c - 2 * l_h * m_p);
L_3 = l_h * K_f;
L_4 = -L_3;

K_1 = L_1/J_p;
K_2 = L_3/J_e;
K_3 = -L_2*L_4/(L_3 * J_lambda);

K_pp = 14;
K_pd = 2*sqrt(K_pp/K_1);
K_rp = -1.2;

rho = K_rp * K_3;
```

A.2 Part III

```
%% Matrices
%Part III - problem 2
A_L = [0 1 0; 0 0 0; 0 0 0];
B_L = [0 0; 0 K_1; K_2 0];
C_L = [1 0 0; 0 0 1];
Q_L = diag([91.2 50 100]);
R_L = diag([1 1]);

%Part III - problem 3 - integrator
A_I = [0 1 0 0 0; 0 0 0 0 0; 0 0 0 0 0; 1 0 0 0 0; 0 0 1 0 0];
B_I = [0 0; 0 K_1; K_2 0; 0 0; 0 0];
C_I = [1 0 0 0 0; 0 0 1 0 0];
Q_I = diag([91.2 50 100 50 50]);
R_I = diag([1, 1]);

%% Linear Quadratic Regulator
```

```
%Part III - problem 2

K_L = lqr(A_L, B_L, Q_L, R_L);
P_L = inv(C_L*inv(B_L*K_L-A_L)*B_L);

%Part III - problem 3
K_I = lqr(A_I, B_I, Q_I, R_I);
P_I = [0 9.999999999999998;9.549869109050652 0];
```

A.3 Part IV

```
%% Matrices
%Part IV

A_E = [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];
B_E = [0 0; 0 K_1; 0 0; K_2 0; 0 0; 0 0];
C_E = [1 0 0 0 0 0; 0 0 1 0 0 0; 0 0 0 0 1 0];
D_E = zeros(3, 2);
sys = ss(A_E, B_E, C_E, D_E, ...
    'StateName', {'p'; 'p_dot'; 'e'; 'e_dot'; 'lambda'; 'lambda_dot'},
    'InputName', {'V_s'; 'V_d'}, 'OutputName', {'p'; 'e'; 'lambda'});

%Checking observability
O = obsv(sys);
rank(O)

%% Pole placement
% Part IV - problem 2

system_poles_L = eig(A_L-B_L*K_L);
system_poles_I = eig(A_I-B_I*K_I);

r0_L = max(abs(system_poles_L));
r0_I = max(abs(system_poles_I));

freq = 13;
angle = pi/8;
spread = -angle:(angle/(2.5)):angle;

radius_L = r0_L*freq;
radius_I = r0_I*freq;

observer_poles_L = -radius_L * exp(1i*spread);
```

```
observer_poles_I = -radius_I * exp(1i*spread);

L_L = transpose(place(transpose(A_E),transpose(C_E),observer_poles_L));
L_I = transpose(place(transpose(A_E),transpose(C_E),observer_poles_I));

%% Additions for problem 4.3
%
%Only measuring e and lambda:

C_new = [0 0 1 0 0 0; 0 0 0 0 1 0];
D_new = 0;
sys_new = ss(A_E, B_E, C_new, D_new, ...
    'StateName', {'p'; 'p_dot'; 'e'; 'e_dot'; 'lambda'; 'lambda_dot'},
    'InputName', {'V_s'; 'V_d'}, 'OutputName', {'e'; 'lambda'});

new_poles = -[1.5 2 2.5 3 3.5 4];
L_new = transpose(place(transpose(A_E),transpose(C_new),new_poles));

O_new = obsv(sys_new);
rank(O_new)

%The rank of O_new is also 6.
```

B Simulink Diagrams

This section contains our Simulink models, given chronologically. Only the relevant and changed parts are shown. Lastly, we have included the top level final Simulink diagram.

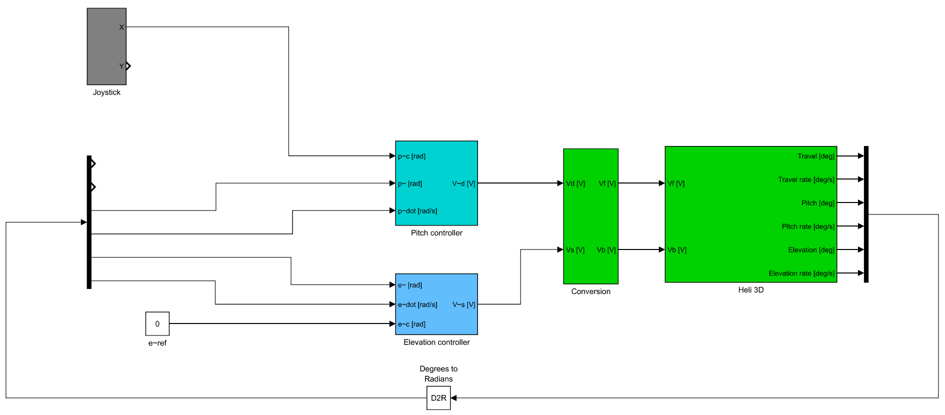


Figure 8: Part II - Problem 1, Simulink model

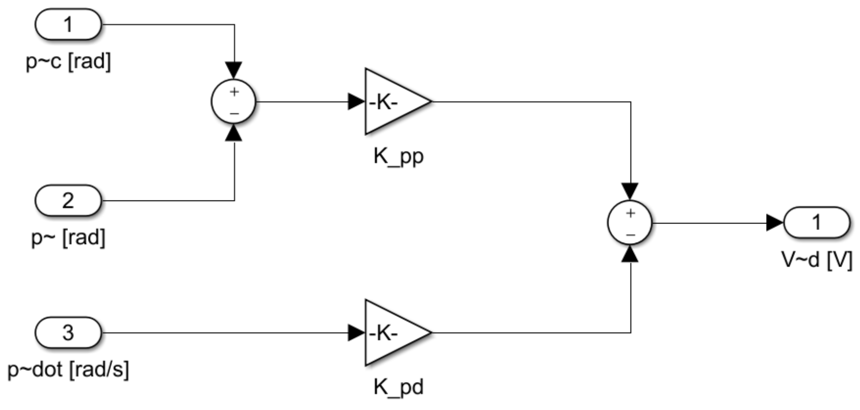


Figure 9: Part II - Problem 1, The PD controller for pitch angle

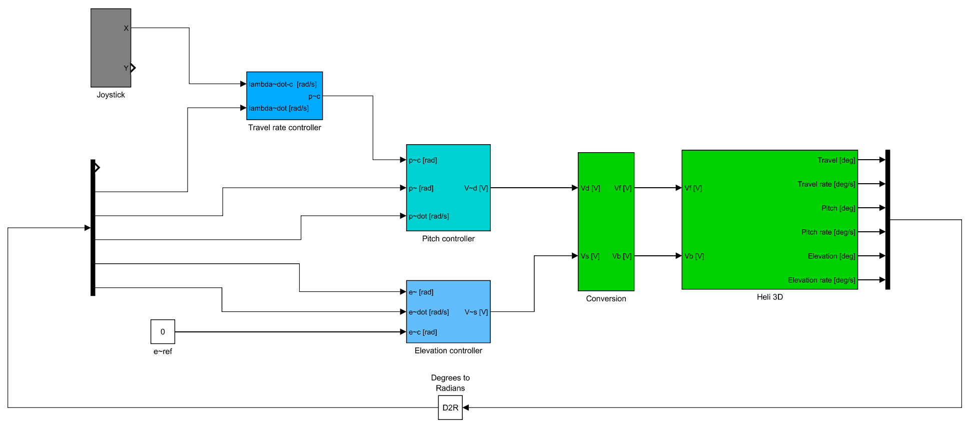


Figure 10: Part II - Problem 2, Simulink model

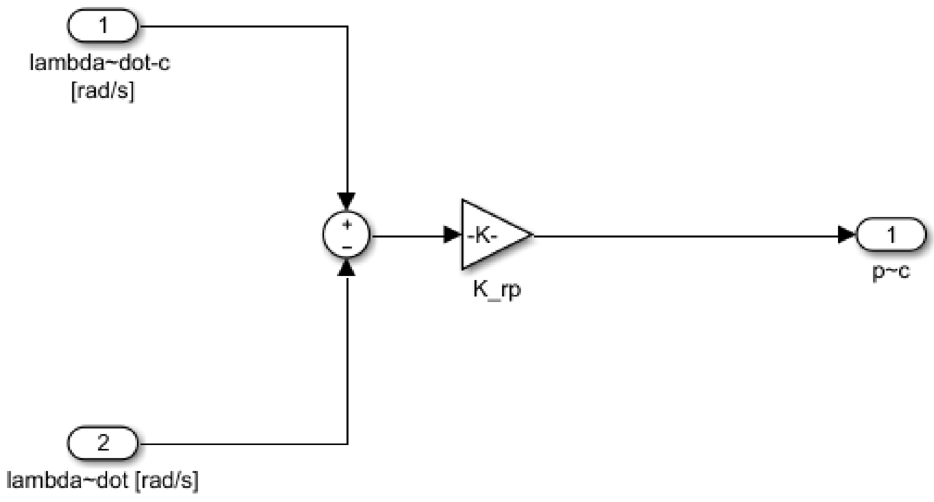


Figure 11: Part II - Problem 2, The P controller for travel rate

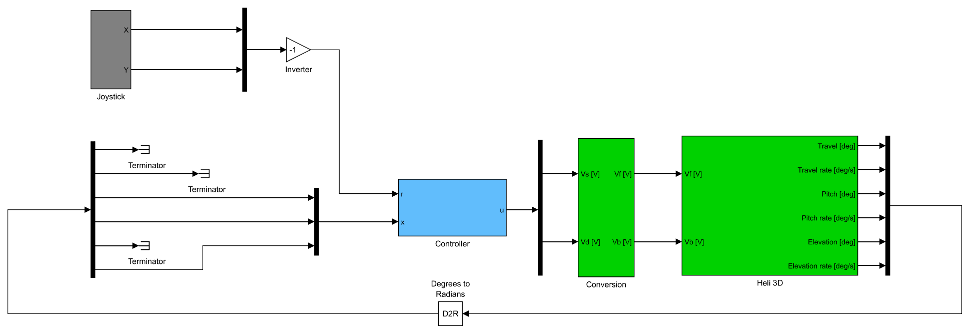


Figure 12: Part III - Problem 2, Simulink model

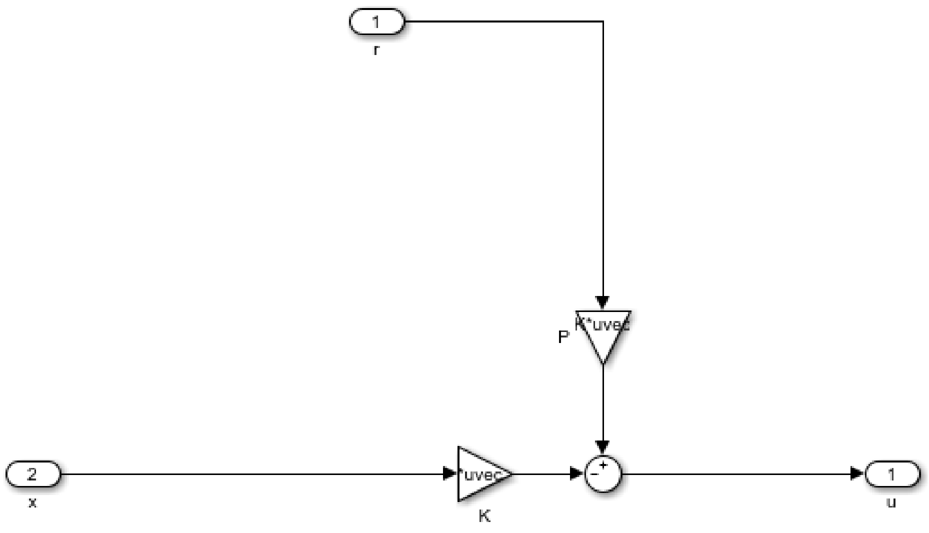


Figure 13: Part III - Problem 2, The LQR

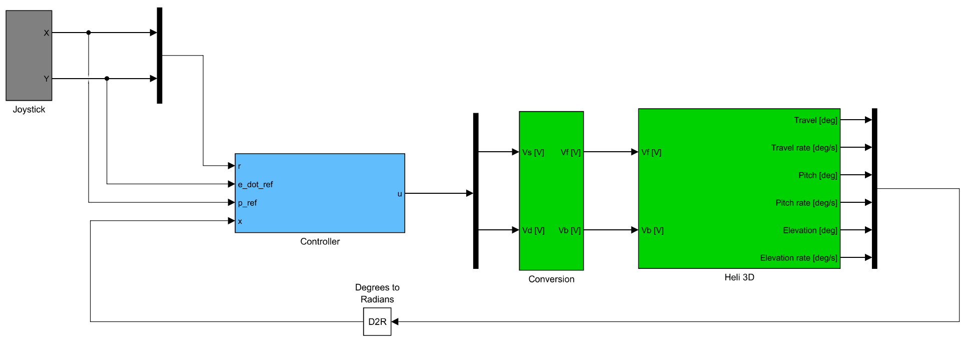


Figure 14: Part III - Problem 3, Simulink model

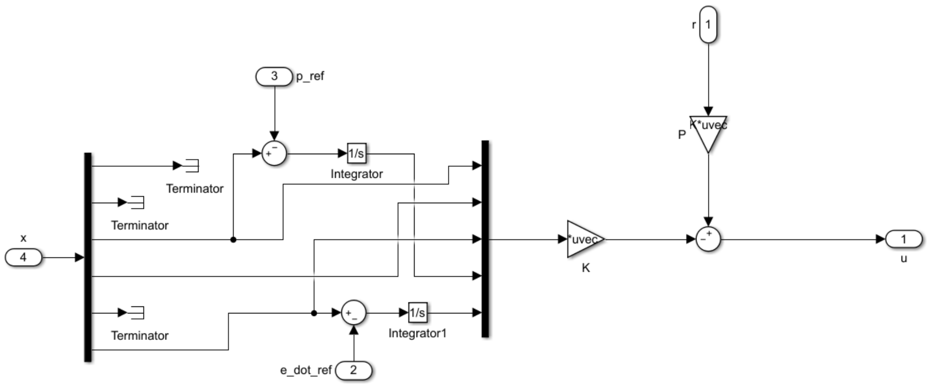


Figure 15: Part III - Problem 3, The PI controller

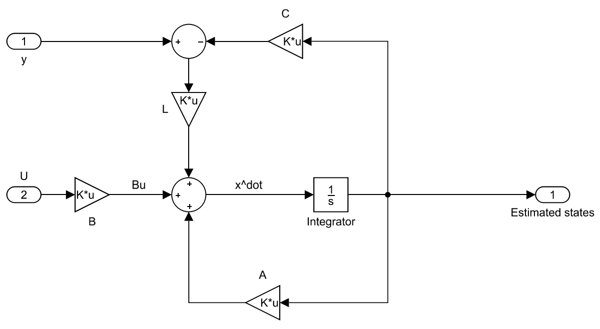


Figure 16: Part IV - Implementation of the estimator controller

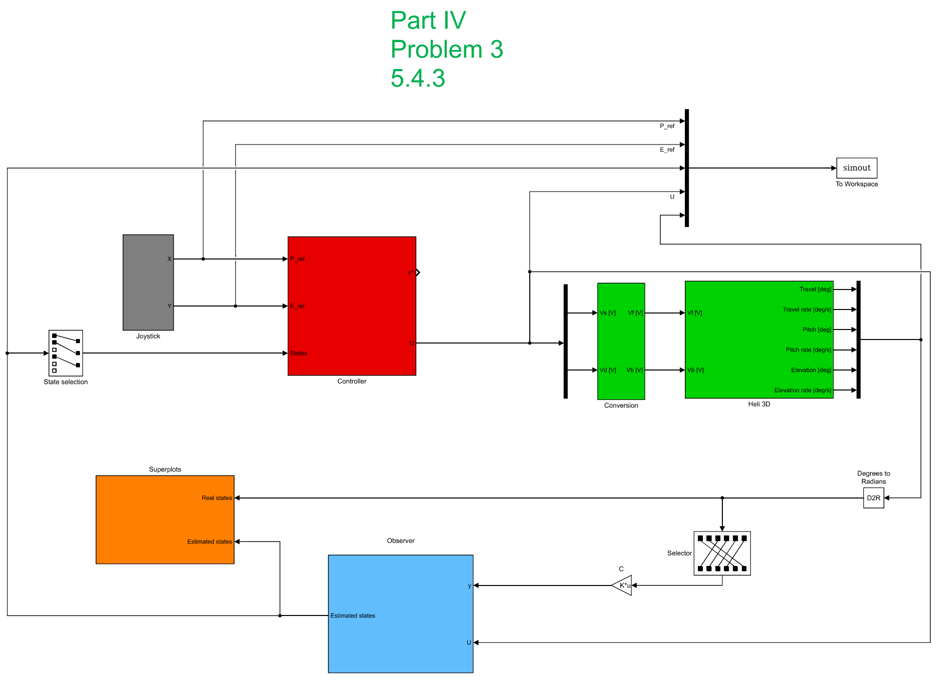


Figure 17: The final simulink diagram

References

[1] Kristoffer Gryte. *TTK 4115 Helicopter lab assignment.* Vol. 4.5. Department of Engineering Cybernetics NTNU, 2017.

[2] Morten D. Pedersen. *Lecture 5 - State feedback.* Department of Engineering Cybernetics NTNU, 2017.