TTK4115 - Helicopter Lab

Aleksander Asp - 767827
Marius Eskedal - 767880
Per Omvik - 768682

October 2017

Contents

Introduction

The purpose of the given assignment can be broken down into four main parts. The first part is to derive a controllable model for the helicopter system using mathematical modeling. Next we want to derive PD and multivariable controllers and optimize them using the feedback from our real-time system as well as our calculations. The third part entails development of a linear quadratic regulator (LQR) for optimal control. The fourth and final category consists of state estimation. More specifically demonstrating how states that are not directly measured, can still be estimated.
Essentially, this report describes how our regulation model was theorized and implemented for the helicopter system.

1 Part I - Mathematical modeling

1.1 Problem 1: Equations of motion

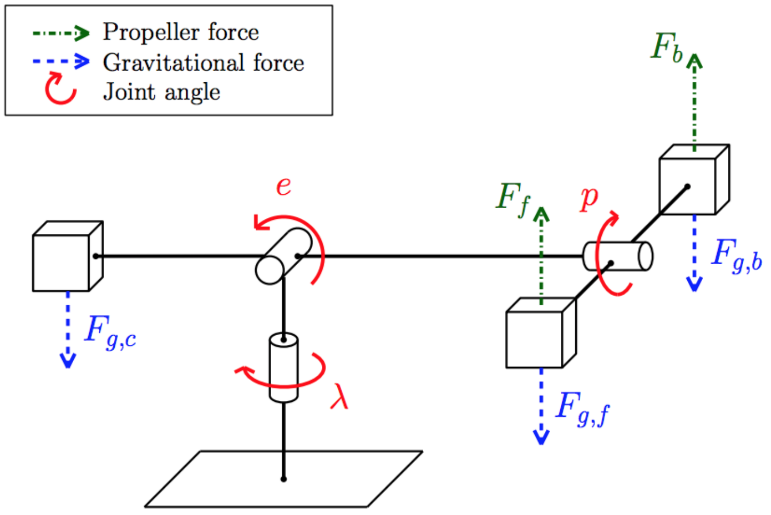

Figure 1: Helicopter model[2]

The value of the masses and lengths, denoted by m_x l_x, referenced in the report were given to us in the MATLAB file: `init_heli_all.m`. The moments of inertia about each rotation joint were calculated using

$$J_p = 2m_p l_p^2 \tag{1.1a}$$
$$J_e = m_c l_c^2 + 2m_p l_h^2 \tag{1.1b}$$
$$J_\lambda = m_c l_c^2 + 2m_p(l_h^2 + l_p^2) \tag{1.1c}$$

Symbol	Parameter	Value	Unit
g	Gravitational acceleration	9.81	$\frac{\text{m}}{\text{s}^2}$
l_p	Distance from pitch axis to motor	0.175	m
l_c	Distance from elevation axis to counterweight	0.46	m
l_h	Distance from elevation axis to helicopter head	0.66	m
J_p	Moment of inertia for pitch	0.044	kg m^2
J_e	Moment of inertia for elevation	1.03	kg m^2
J_λ	Moment of inertia for travel	1.08	kg m^2
m_p	Motor mass	0.72	kg
m_c	Counterweight mass	1.92	kg

Table 1: Constant values

To calculate the equations of motion for the helicopter, we need to calculate the sum of the torques about each rotation joint using Newton's second law of rotation.

$$J\ddot{\theta} = \sum \tau \tag{1.2}$$

We use (1.2) for the pitch angle p, elevation angle e and the travel angle λ:

The pitch is controlled by the difference between the forces created by the rotors. These forces are proportional to the voltage applied.

$$\begin{aligned}
J_p\ddot{p} &= l_pF_f - l_pFb + l_pF_{g,b}\cos(p) - l_pF_{g,f}\cos(p) \\
&= l_p(F_f - F_b) + l_p\cos(p)(F_{g,f} - F_{g,f}) \\
&= l_p(F_f - F_b) && , F_{g,b} = F_{g,f} \\
&= l_pK_f(V_f - V_b) && , F_f = K_fV_f \text{ and } F_b = K_fV_b \\
&= l_pK_fV_d && , V_d = V_f - V_b \\
&\Rightarrow J_p\ddot{p} = L_1V_d
\end{aligned} \tag{1.3}$$

Where $L_1 = l_pK_f$. K_f is the motor constant, which will be calculated in subsection 1.4.

The elevation of the helicopter is caused by the different forces acting perpendicularly on the rotation axis. In our case, gravity, depending on the elevation angle, and rotor forces, depending on pitch angle, are affecting the elevation.

$$\begin{aligned}
J_e\ddot{e} &= l_cF_{g,c}\cos(e) + l_h((F_f + F_b)\cos(p) - (F_{g,f} + F_{g,b})\cos(e)) \\
&= l_cm_cg\cos(e) + l_hK_fV_s\cos(p) - l_h(F_{g,f} + F_{g,b})\cos(e) && , V_s = V_f + V_d \\
&= (l_cm_cg - l_h2m_pg)\cos(e) + l_hK_fV_s\cos(p)\cos(e) && , F_{g,f} = F_{g,b} = m_pg \\
&\Rightarrow J_e\ddot{e} = L_2\cos(e) + L_3V_s\cos(p)
\end{aligned} \tag{1.4}$$

Where $L_2 = l_cm_cg - l_h2m_pg$, and $L_3 = l_hK_f$

The travel angle λ is determined by the torques applied from the rotors depending on both the elevation and pitch angles.

$$\begin{aligned}
J_\lambda\ddot{\lambda} &= -l_h(F_f + F_b)\sin(p)\cos(e) \\
&= -l_hK_f(V_f + V_b)\sin(p)\cos(e) \\
&= -l_hK_fV_s\sin(p)\cos(e) \\
&\Rightarrow J_\lambda\ddot{\lambda} = L_4V_s\sin(p)\cos(e)
\end{aligned} \tag{1.5}$$

Where $L_4 = -l_hK_f$

It is important to note that the equations are approximations, and do not take external factors such as friction from the air(drag) and in the rotation joints, into account.

1.2 Problem 2: Linearization

In reality, as the equations of motion suggest, the helicopter system is nonlinear. To implement a linear controller for the helicopter, we wish to linearize the system about the point $(p, e, \lambda)^T = (p^*, e^*, \lambda^*)^T$, where $p^* = e^* = \lambda^* = 0$. To find $(V_s^*, V_d^*)^T$ at this point, we use that $(\dot{p}, \dot{e}, \dot{\lambda})^T = (0, 0, 0)^T$ (and therefore $(\ddot{p}, \ddot{e}, \ddot{\lambda})^T = (0, 0, 0)^T$) for these values for V_s^* and V_d^*. Using (1.3) and (1.4) with $\ddot{p} = \ddot{e} = 0$ we get:

$$J_p \cdot 0 = L_1 V_d^* \Rightarrow V_d^* = 0$$

$$J_e \cdot 0 = L_2 \cos(0) + L_3 V_s^* \cos(0) \Rightarrow V_s^* = -\frac{L_2}{L_3}$$

$$\begin{bmatrix} V_s^* \\ V_d^* \end{bmatrix} = \begin{bmatrix} -\frac{L_2}{L_3} \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{l_c m_c g - l_h 2 m_p g}{l_h K_f} \\ 0 \end{bmatrix} \tag{1.6}$$

We introduce the coordinate transform

$$\begin{bmatrix} \tilde{p} \\ \tilde{e} \\ \tilde{\lambda} \end{bmatrix} = \begin{bmatrix} p \\ e \\ \lambda \end{bmatrix} - \begin{bmatrix} p^* \\ e^* \\ \lambda^* \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} = \begin{bmatrix} V_s \\ V_d \end{bmatrix} - \begin{bmatrix} V_s^* \\ V_d^* \end{bmatrix} \tag{1.7}$$

The transformed states $\tilde{\mathbf{x}}$ become:

$$\begin{aligned}
\tilde{x}_1 &= \tilde{p} \Rightarrow \dot{\tilde{x}}_1 = f_1 = \tilde{x}_2 \\
\tilde{x}_2 &= \dot{\tilde{p}} \Rightarrow \dot{\tilde{x}}_2 = f_2 = \frac{L_1 \tilde{V}_d}{J_p} \\
\tilde{x}_3 &= \tilde{e} \Rightarrow \dot{\tilde{x}}_3 = f_3 = \tilde{x}_4 \\
\tilde{x}_4 &= \dot{\tilde{e}} \Rightarrow \dot{\tilde{x}}_4 = f_4 = \frac{L_2}{J_e} \cos \tilde{x}_3 + \frac{L_3}{J_e} \cos \tilde{x}_1 \left(\tilde{V}_s - \frac{L_2}{L_3} \right) \\
\tilde{x}_5 &= \tilde{\lambda} \Rightarrow \dot{\tilde{x}}_5 = f_5 = \tilde{x}_6 \\
\tilde{x}_6 &= \dot{\tilde{\lambda}} \Rightarrow \dot{\tilde{x}}_6 = f_6 = \frac{L_4 \cos \tilde{x}_3 \sin \tilde{x}_1}{J_\lambda} \left(\tilde{V}_s - \frac{L_2}{L_3} \right)
\end{aligned}$$

We want to linearize the state-space representated model:

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

about the equilibrium points $\mathbf{x}_0$ and $\mathbf{u}_0$. This is done by the following formula:

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_6} \\ \vdots & \ddots & \\ \frac{\partial f_6}{\partial x_1} & & \frac{\partial f_6}{\partial x_6} \end{bmatrix} \Bigg|_{x=\mathbf{x}_0, u=\mathbf{u}_0} \tilde{\mathbf{x}} + \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \vdots & \vdots \\ \frac{\partial f_6}{\partial u_1} & \frac{\partial f_6}{\partial u_2} \end{bmatrix} \Bigg|_{x=\mathbf{x}_0, u=\mathbf{u}_0} \tilde{\mathbf{u}}$$

For our system, we get

$$\dot{\tilde{\mathbf{x}}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{L_4 V_s^*}{J_\lambda} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{L_1}{J_p} \\ 0 & 0 \\ \frac{L_3}{J_e} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tilde{\mathbf{u}} \tag{1.8}$$

Inserting for $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ result in

$$\dot{\tilde{x}}_2 = \ddot{\tilde{p}} = \frac{L_1}{J_p}\tilde{V}_d = K_1\tilde{V}_d \tag{1.9a}$$

$$\dot{\tilde{x}}_4 = \ddot{\tilde{e}} = \frac{L_3}{J_e}\tilde{V}_s = K_2\tilde{V}_s \tag{1.9b}$$

$$\dot{\tilde{x}}_6 = \ddot{\tilde{\lambda}} = \frac{L_4}{J_\lambda}V_s^*\tilde{p} = K_3\tilde{p} \tag{1.9c}$$

1.3 Problem 3: Feed forward

Using a direct feed forward from the joystick outputs to the voltages resulted in a system that was difficult to control. Before we started, we predicted that the input needed to control the helicopter would need to be small and precise. At the start, the joystick gains for both x and y were set at 1. These were too small to give the helicopter enough lift to leave the surface. We adjusted the y-gain to 8 and the x-gain to 3, making the control of the helicopter easier and more responsive. Regardless, the helicopter was not easy to control over longer periods of time. The helicopter would quickly make sporadic movements which were difficult to correct, and often overcompensated for, resulting in a crash. Due to this, we found that linking the theoretical models in (1.9) to the physical behavior of the helicopter was difficult. The plots from the simulink scopes for both elevation and pitch were messy and difficult to follow. The values from the plots were consequently not evidently linked to either V_s or V_d. We imagine that this was mostly because the pitch and elevation angles were not sufficiently stable around the equilibrium points used in the linearization.

1.4 Problem 4: Motor constant

We found V_s^* by holding the elevation of the helicopter at the equilibrium point $e^* = e = 0$, and reading the sum of voltages V_s caused by the joystick input. We determined the elevation angle when the helicopter was still on the table to be $-29°$ while defining zero degrees as when the helicopter is in a horizontal position. We found V_s^* to be 6.2V. Furthermore, the motor constant can be determined using (1.6):

$$K_f = -\frac{l_c m_c g - l_h 2 m_p g}{l_h V_s^*} = 0.1611 \tag{1.10}$$

2 Part II - Monovariable control

2.1 Problem 1: Pitch PD controller

We want to control the pitch angle using a PD controller. The controller is given by

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

The implemented Simulink model for the pitch controller is shown in figure 9 in Appendix B. This is a sub-model of the entire system shown in figure 8 in Appendix B.

Substituting $\tilde{V}_d$ in (2.1) for $\tilde{V}_d$ in (1.9a) and applying the Laplace transform to the resulting second order differential equation yields the following transfer function

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

$$\Rightarrow \tilde{p}(s^2 + K_1K_{ds} + K_1K_p) = K_1K_p\tilde{p}_c$$

$$\Rightarrow \frac{\tilde{p}(s)}{\tilde{p}_c(s)} = \frac{K_1K_{pp}}{s^2 + K_1K_{pd}s + K_1K_{pp}} \tag{2.2}$$

When deciding the values for K_{pd} and K_{pp} we utilize the form of the transfer function of a critically damped system, $\frac{K\omega_0^2}{s^2+2\zeta\omega_0 s+\omega_0^2}$ with $\zeta = 1$. We choose $\zeta = 1$ to achieve critical damping, which results in an optimal system for speed and stability. This also gives us eigenvalues located on the real axis in the complex plane. Comparing this transfer function with the one obtained in (2.2) we arrive at a ratio between K_{pd} and K_{pp} given by $K_{pd}^2 = \frac{4K_{pp}}{K_1}$. After testing and tuning the response of the helicopter with varying values of K_{pp} and K_{pd} we arrive at

$$K_{pd} = 6 \tag{2.3}$$

$$K_{pp} = 10 \tag{2.4}$$

Choosing too large values for K_{pp} and K_{pd} would result in a damping ratio $\zeta > 1$, which in turn results in two distinct, real eigenvalues and our system becomes over-critically damped. On the opposite side, too small values would give two complex conjugated eigenvalues, $\zeta < 1$ and an under-critically damped system.

The helicopter is easier to control after the pitch PD controller has been implemented. The system is no longer oversensitive to joystick input and we do not need to manually correct for a non-horizontal pitch angle. However, when looking at the step response of the pitch, shown in figure 2 in Appendix A, we see that it is unable to hold the reference for longer periods, decaying over time. This is likely due to the fact that we are far from the linearization point of $p = 0$.

2.2 Problem 2: Travel rate P controller

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

$$\tilde{p}_c = K_{rp}(\dot{\tilde{\lambda}}_c - \dot{\tilde{\lambda}}), \tag{2.5}$$

where the constant $K_{rp} > 0$.

From equation (1.9c) we have that $\ddot{\tilde{\lambda}} = K_3\tilde{p}$. Using this, and assuming that we have perfect pitch control, i.e $\tilde{p} = \tilde{p}_c$ we get the following equality

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

Using the Laplace transform we obtain

$$s\dot{\tilde{\lambda}} = K_3 K_{rp}(\dot{\tilde{\lambda}}_c - \dot{\tilde{\lambda}}) \Rightarrow \frac{\dot{\tilde{\lambda}}}{\dot{\tilde{\lambda}}_c} = \frac{K_3 K_{rp}}{s + K_3 K_{rp}},$$

which can be written as the following transfer function

$$\frac{\dot{\tilde{\lambda}}(s)}{\dot{\tilde{\lambda}}_c(s)} = \frac{p}{s+p}, \tag{2.6}$$

where $p = K_3 K_{rp}$.

After the implementation of the P controller the travel rate reaches a constant value based on the output of the joystick. By tuning and testing the system in real-time we saw that a joystick gain of 0.9 and $K_{rp} = -1.5$ gave optimal response.

The P controller is shown in the Simulink model in figure 10 in Appendix B

3 Part III

3.1 Problem 1: Multivariable state-space equation

Introducing multivariable Controller for both pitch angle $\tilde{p}$ and elevation rate $\dot{\tilde{e}}$, we now have a new state and input vector.

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

$$\mathbf{x} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} \tilde{V_s} \\ \tilde{V_d} \end{bmatrix} \tag{3.1}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix} \tag{3.2}$$

Using (3.1) and (3.2) gives us the following state-space equation for our multivariable system

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V_s} \\ \tilde{V_d} \end{bmatrix} \tag{3.3}$$

3.2 Problem 2: Controllability

To compute if our system is controllable, we calculate the rank of the controllability Matrix $\mathcal{C}$, which is given by

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

Using values from $\mathbf{A}$ and $\mathbf{B}$ in (3.2) result in

$$\mathcal{C} = \begin{bmatrix} 0 & 0 & 0 & K_1 & 0 & 0 \\ 0 & K_1 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

Observing that $\mathcal{C}$ has full rank, we conclude that the system is controllable. This means we can use state feedback to place the poles of our system anywhere we want.
We aim to track the reference $\mathbf{r} = \begin{bmatrix} \tilde{p_c} & \dot{\tilde{e_c}} \end{bmatrix}^T$ where the respective elements are references of the joystick's X and Y position by a controller with forward reference and state feedback.

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$\mathbf{u} = \mathbf{P}\mathbf{r} - \mathbf{K}\mathbf{x} \tag{3.6}$$

Here $\mathbf{K}$ corresponds to the linear quadric regulator (LQR) for which the control input $\mathbf{u} = -\mathbf{K}\mathbf{x}$ optimizes the cost function, giving us the optimal pole placement.

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

$\mathbf{Q}$ and $\mathbf{R}$ are diagonal weighted matrices which have to be tuned by trial and error to optimize the response of the system. The diagonal values of $\mathbf{Q}$ correspond to how high the cost of mistakes for

each state are and the diagonal values of **R** corresponds to how much power is readily for the inputs. The K matrix is given by the MATLAB command LQR(A,B,Q,R) for the chosen **Q** and **R** values. Determining **P** involves finding a solution to which feed forward gain, **P**, results in $y(t) = \mathbf{r}$ as $t \to \infty$. As $t \to \infty$ we have the conditions $\dot{x} = 0$ and $\mathbf{y}_\infty = \mathbf{r}_0$
Inserting u from (3.6) into the general state-space equation and evaluating at steady state

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \qquad \mathbf{y} = \mathbf{C}\mathbf{x} \tag{3.7}$$

$$0 = \mathbf{A}\mathbf{x}_\infty + \mathbf{B}(\mathbf{P}\mathbf{r}_0 - \mathbf{K}\mathbf{x}_\infty) \Longrightarrow \mathbf{x}_\infty = (\mathbf{B}\mathbf{K} - \mathbf{A})^{-1}\mathbf{B}\mathbf{P}\mathbf{r}_0$$

$$\mathbf{y}_\infty = \mathbf{C}(\mathbf{B}\mathbf{K} - \mathbf{A})^{-1}\mathbf{B}\mathbf{P}\mathbf{r}_0 \Longrightarrow \mathbf{P} = (\mathbf{C}[\mathbf{B}\mathbf{K} - \mathbf{A}]^{-1}\mathbf{B})^{-1} \tag{3.8}$$

For our tuning process we initially used low values for **Q** and high values for **R**, protecting the system of mechanical saturation.

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{3.9}$$

As the process of tuning is an iterative process, the method we used was for our chosen **R** values, optimize **Q** with regards to which states need optimizing until the system behaves best for the chosen **R** values. Then we lowered the **R** values to make the system more aggressive and repeat. This was done until the system behaved at desirable functionality. The following values resulted in the best response

$$\mathbf{Q} = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 500 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.10}$$

This gives us the following **K** and **P** values

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 31.622 \\ 14.142 & 9.708 & 0 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 0 & 31.622 \\ 14.142 & 0 \end{bmatrix} \tag{3.11}$$

This tuning lead to fast responsiveness in small changes for both the pitch and the elevation. As the system was tuned aggressively, it responds fast to small changes, but with some minor oscillations at times. Though we were not able to tune the system to maintain a constant level after we set the reference. As displayed in figure 3, the elevation gradually deviates after being applied a step function. This is likely due to noise from the elevation rate measurement causing the system to not be in its linearized area which causes a deviation our currently designed system can not compensate for. The best we could do was lowering the rate of decline by aggressively tuning the system towards maintaining this state.
The controller with integral effect is shown in the Simulink model in figure 11 in Appendix B

3.3 Problem 3: Integral effect

To include integral effect in our controller for elevation rate $\dot{\tilde{e}}$ and pitch angle $\tilde{p}$ we need to include two additional states.

$$\begin{aligned} \dot{\gamma} &= \tilde{p} - \tilde{p}_c \\ \dot{\zeta} &= \dot{\tilde{e}} - \dot{\tilde{e}}_c \end{aligned} \tag{3.12}$$

$$\overline{\mathbf{x}} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \\ \gamma \\ \zeta \end{bmatrix} \quad \overline{\mathbf{u}} = -\overline{\mathbf{K}}\overline{\mathbf{x}} + \overline{\mathbf{P}}\mathbf{r} \tag{3.13}$$

The states $\overline{x}$ and state feedback matrix $\overline{\mathbf{K}}$ can be separated into

$$\mathbf{x} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \end{bmatrix} \quad \mathbf{x}_i = \begin{bmatrix} \gamma \\ \zeta \end{bmatrix} \quad \overline{\mathbf{K}} = \begin{bmatrix} \mathbf{K} & \mathbf{K}_i \end{bmatrix} \tag{3.14}$$

Using (3.12), $\dot{\mathbf{x}}_i$ can be written as

$$\dot{\mathbf{x}}_i = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{e}} \end{bmatrix} - \begin{bmatrix} \tilde{p}_c \\ \dot{\tilde{e}}_c \end{bmatrix} = \mathbf{C}\mathbf{x} - \mathbf{r} \tag{3.15}$$

$$\Downarrow$$

$$\overline{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & 0 \\ \mathbf{C} & 0 \end{bmatrix} \quad \overline{\mathbf{B}} = \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} \tag{3.16}$$

Which achieves the following state-space model

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_i \end{bmatrix} = \begin{bmatrix} \mathbf{A} & 0 \\ \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_i \end{bmatrix} - \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} \begin{bmatrix} \mathbf{K} & \mathbf{K}_i \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_i \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ 0 \end{bmatrix} \overline{\mathbf{P}}\mathbf{r} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} \mathbf{r}$$

$$\Downarrow$$

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_i \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{B}\mathbf{K} & \mathbf{B}\mathbf{K}_i \\ \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_i \end{bmatrix} + \begin{bmatrix} \mathbf{B}\overline{\mathbf{P}} \\ -1 \end{bmatrix} \mathbf{r} \tag{3.17}$$

As $\overline{\mathbf{K}}$ for our system is defined as $\overline{\mathbf{K}} = lqr(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\mathbf{Q}}, \overline{\mathbf{R}})$, we now need to evaluate $\overline{\mathbf{P}}$.
Evaluating (3.17) as $t \to \infty \Rightarrow 0 = \mathbf{C}\mathbf{x} - \mathbf{r} \Rightarrow \mathbf{y} = \mathbf{r}$ meaning that the integral part of the system will make the output equal to our reference. As this is independent of our $\mathbf{x}$ part of the system, we no longer need to design $\overline{\mathbf{P}}$ as a proportional gain for our reference to achieve $\mathbf{y} = \mathbf{r}$ as $t \to \infty$. Instead we can design $\overline{\mathbf{P}}$ to improve tracking and bandwidth by designing $\overline{\mathbf{P}}$ as if we have no integral action for our system, which is what we did in the last problem.

$$\overline{\mathbf{P}} = (\mathbf{C}(\mathbf{B}\mathbf{K} - \mathbf{A})^{-1}\mathbf{B})^{-1} \tag{3.18}$$

Using the same process as last time for our tuning, and focusing on minimizing errors in $\zeta = \dot{\tilde{e}} - \dot{\tilde{e}}_c$ as this will result in maintaining a steady elevation level better

$$\mathbf{Q} = \begin{bmatrix} 50 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.19}$$

This gives us the following $\mathbf{K}$ and $\mathbf{P}$ values

$$\overline{\mathbf{K}} = \begin{bmatrix} 0 & 0 & 25.979 & 0 & 14.142 \\ 9.477 & 6.296 & 0 & 3.162 & 0 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} 0 & 25.979 \\ 9.477 & 0 \end{bmatrix} \tag{3.20}$$

We observe that adding integral effect to our controller makes it easier to tune the system as it is generally stabler. This is because the integral effect removes stationary errors from pitch and elevation rate, making the system more robust and easier to track. As shown in figure 3, after applying the same step function to our system as we did for our system in section 3.2, the elevation does not decline and maintains a steady elevation level. This controller design is very functional as the system behaves desirably in tracking the pitch and elevation rate.
The controller with integral effect is shown in the Simulink model in figure 12 in Appendix B

4 Part IV - State estimation

4.1 Problem 1: State space representation

We can represent the system of linearized equations (1.9), as a state space model:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x}\end{aligned} \tag{4.1}$$

We use the following for $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{y}$:

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

For the $\mathbf{A}$ and $\mathbf{B}$ matrices we can use the results from (1.8). $\mathbf{C}$ is the relation between $\mathbf{y}$ and $\mathbf{x}$. We end up with the following state space representation:

$$\begin{aligned}\dot{\mathbf{x}} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ K_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} \\ \mathbf{y} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix}\end{aligned} \tag{4.3}$$

4.2 Problem 2: Observer

The system is considered observable if the rank of the observability matrix

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \vdots \\ \mathbf{CA}^5 \end{bmatrix} \tag{4.4}$$

is equal to the degree of our system, which is 6. Using `O = obsv(A,C)` in MATLAB, we can confirm that the rank of $\mathcal{O}$ is 6, and the system is considered observable. We can then construct the observer

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{L}(\mathbf{y}_m - \mathbf{C}\hat{\mathbf{x}}) \tag{4.5}$$

The error is given as

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$$
$$\Rightarrow \dot{\mathbf{e}} = \dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}}$$

Our measured output, $\mathbf{y}_m$, is considered to include noise $\mathbf{n}$. Using (4.5), we arrive at

$$\dot{\mathbf{e}} = (\mathbf{A} - \mathbf{LC})\mathbf{e} + \mathbf{Ln} \tag{4.6}$$

The matrtix $\mathbf{L}$ can be chosen such that

$$\lim_{t\to\infty} \mathbf{e} = 0$$

Because the system is observable, we can place the poles of the closed loop observer $(\mathbf{A} - \mathbf{LC})$ arbitrarily. Generally, the dynamics of the error should be between 2 and 20 times faster than the plant itself, which is $(\mathbf{A} - \mathbf{BK})$[3]. To achieve this, the poles of the closed loop observer needs to have a larger negative real part compared to the poles of the closed loop system. This was done by fanning out the poles in the left half-plane with a radius multiple times larger than the distance from the origin to fastest converging pole of the plant. Furthermore, the $\mathbf{L}$ matrix was set using `place` in MATLAB. The full implementation is shown in the MATLAB script in appendix C. The simulink implementation for the complete model with integral effect and observer are figure 15 and 16, in appendix B.

The values of the $\mathbf{L}$ matrix increase when the poles are placed with large negative real parts, and decrease when they are placed closer to the origin. If we study equation (4.6), we can see that a faster observer will also lead to amplified measuring noise. In our testing, we observed that the plots for the estimated states became smoother as the poles approached the origin, as expected. This however, led to more inaccurate estimation with larger offsets and ultimately: worse control. Despite this, we discovered that for us to notice any clear changes in the quality of control, the radius of the poles would need to change tenfolds. During our use of the estimated feedback, we consistently got a positive offset on the estimated elevation rate compared to the measured one, causing the helicopter to descend slowly. This problem was improved upon when we increased the radius of the closed loop observer poles, but even increasing it to 60 times the value of the most negative plant pole, did not completely fix our issue. The plots comparing estimated and measured states are in figure 5 and 6 in appendix A.

4.3 Problem 3: Observer without pitch

If we only measure $\tilde{e}$ and $\tilde{\lambda}$, we get

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

This changes our observability matrix $\mathcal{O}$, and we need to reassure that the rank of $\mathcal{O}$ remains at 6. Again, using `O = obsv(A,C)` in MATLAB, we find this to be the case.
If we, however, only measure $\tilde{p}$ and $\tilde{e}$, and the $\mathbf{C}$ matrix changes accordingly, the same MATLAB command gives us an observability matrix $\mathcal{O}$ where $Rank(\mathcal{O}) = 4$. We can subsequently determine that the system is not observable.

Controlling the helicopter using the implemented estimated state feedback while only measuring $\tilde{e}$ and $\tilde{\lambda}$ proved to be very challenging. If we examine the plots of the estimated states in figure 7 in appendix A, we can see large disparities between estimated and measured states. The estimated pitch and pitch rate are calculated using the measured $\tilde{\lambda}$. The relation between $\tilde{\lambda}$ and $\tilde{p}$ is described in equation (1.9c) which indicates that $\tilde{p}$ is retrieved by differentiating $\tilde{\lambda}$ twice, and dividing by K_3. This means that if measurements for $\tilde{\lambda}$ are noisy, we get amplified errors and highly incorrect results. We tried to reduce the negative real value of the poles to dampen this noise, but we saw no improvement in control. The plots comparing estimated and measured states are in figure 7 in appendix A

Appendices

A Plots

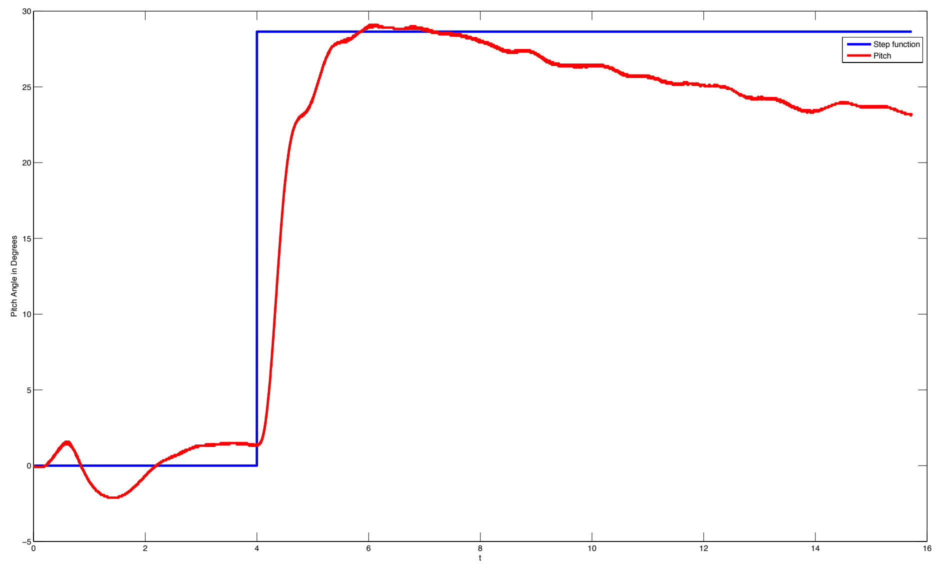

Figure 2: Measured pitch angle after a step function is applied. Part 2 - Problem 1

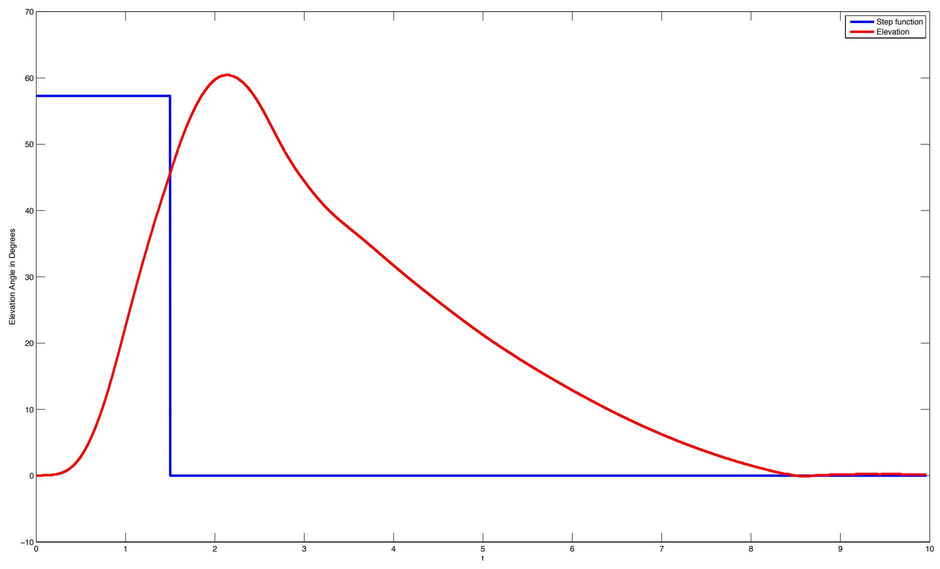

Figure 3: Measured elevation angle after a step function is applied. Part 3 - Problem 2

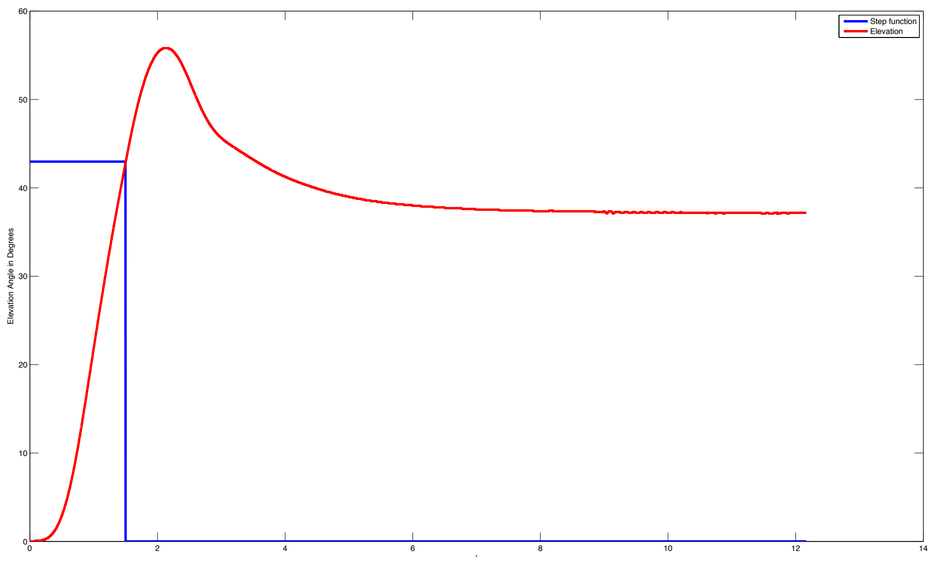

Figure 4: Measured elevation angle after a step function is applied. Part 3 - Problem 3

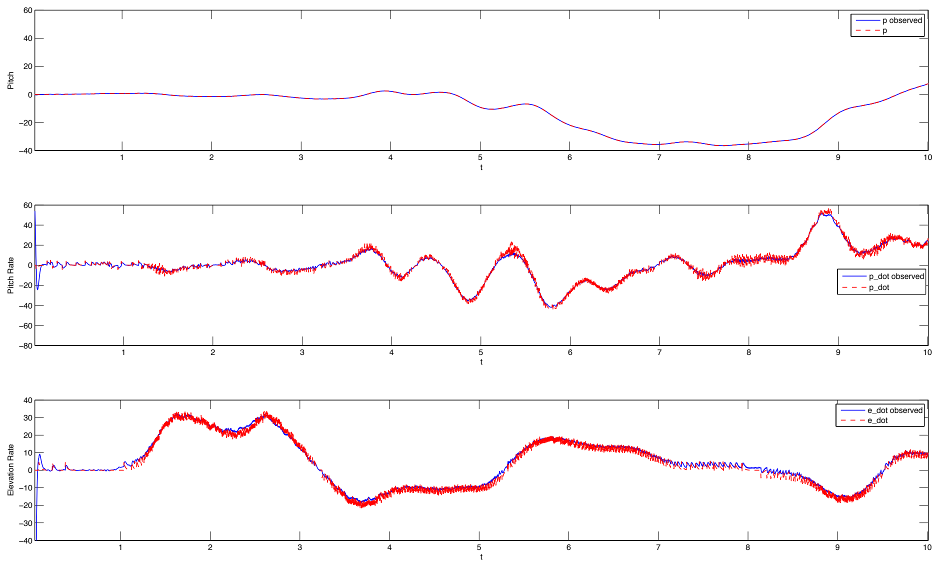

Figure 5: Part 4 - Problem 2

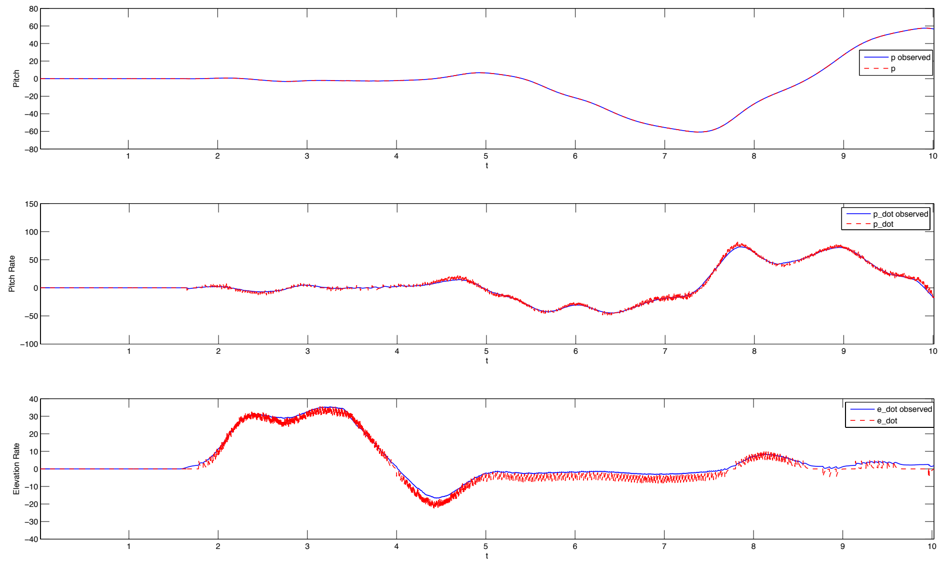

Figure 6: Part 4 - Problem 2, with integral effect

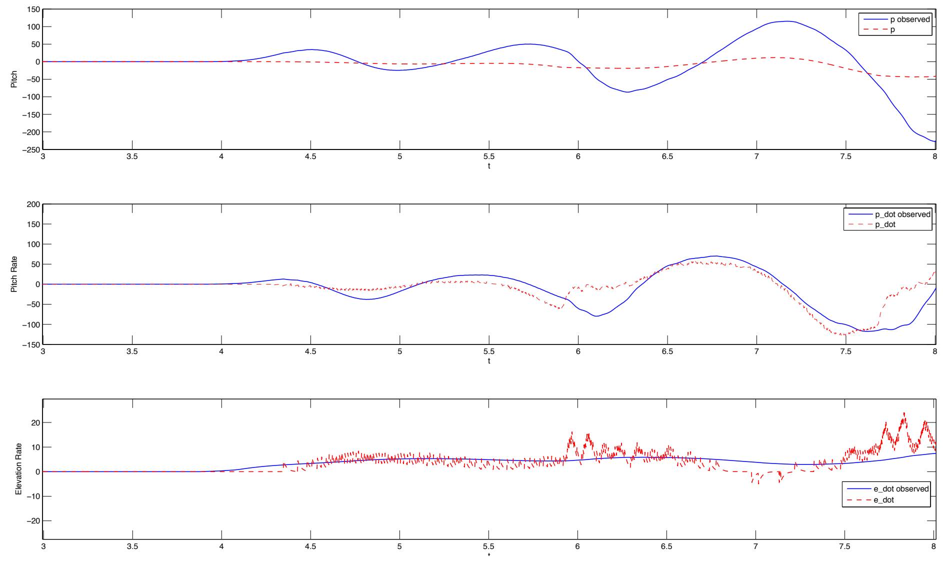

Figure 7: Part 4 - Problem 3

B Simulink models

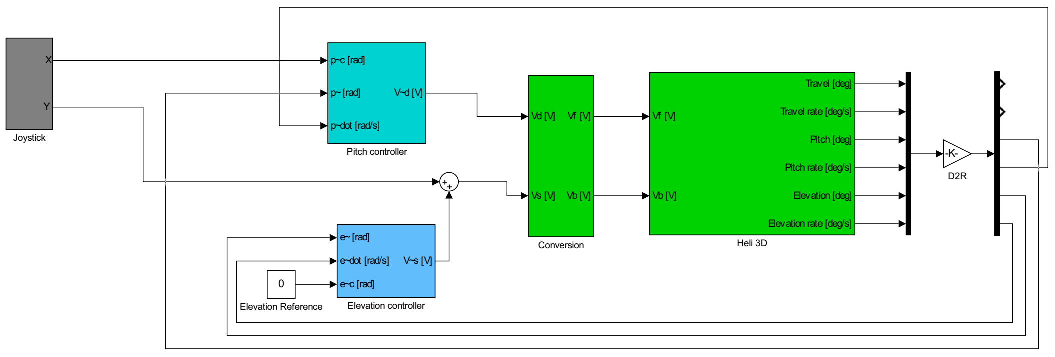

Figure 8: Part 2 - Problem 1

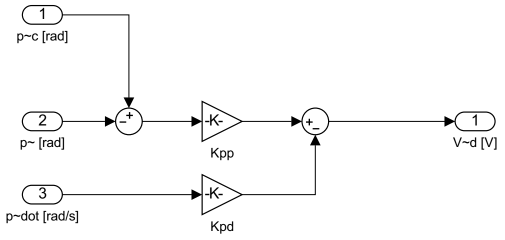

Figure 9: The pitch controller in Part 2 - Problem 1

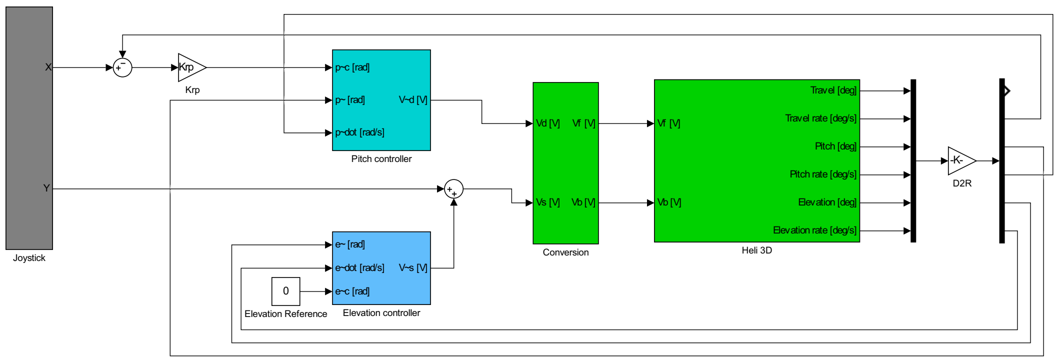

Figure 10: The pitch controller in Part 2 - Problem 2

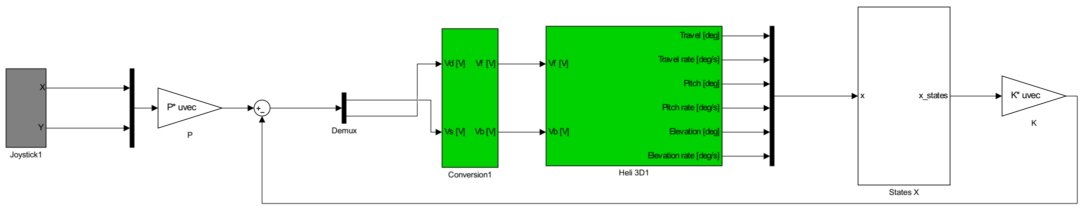

Figure 11: Part 3 - Problem 2

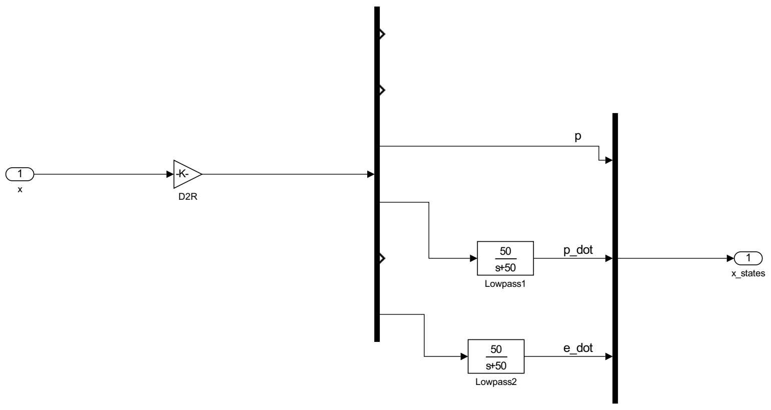

Figure 12: "States X" in Part 3 - Problem 2

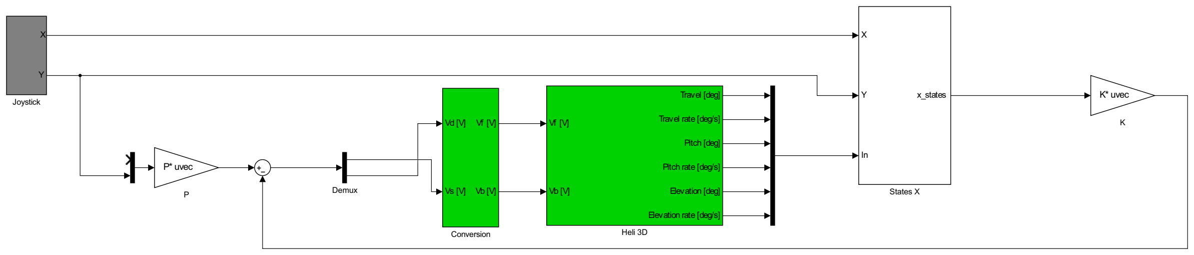

Figure 13: Part 3 - Problem 3

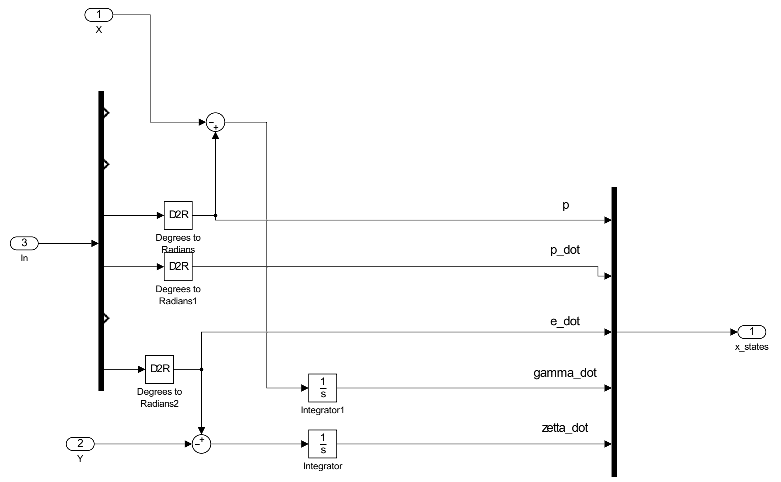

Figure 14: "States X" in Part 3 - Problem 3

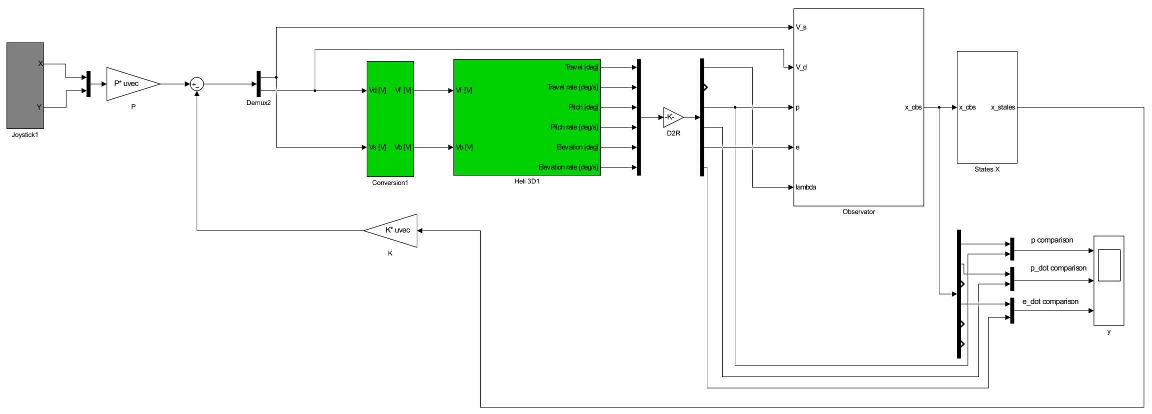

Figure 15: Part 4 - Problem 2

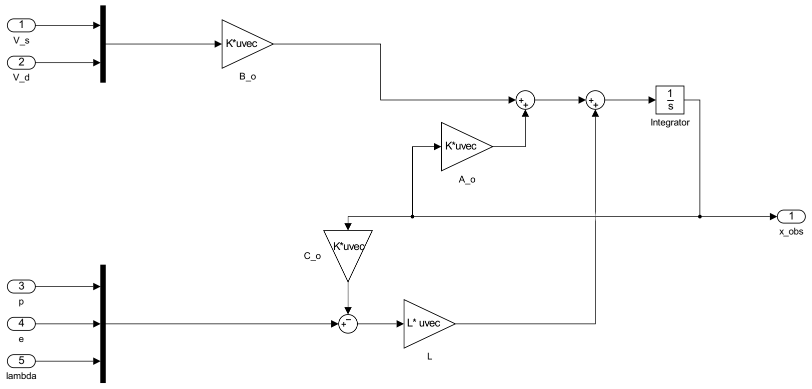

Figure 16: "Observator" in Part 4 - Problem 3

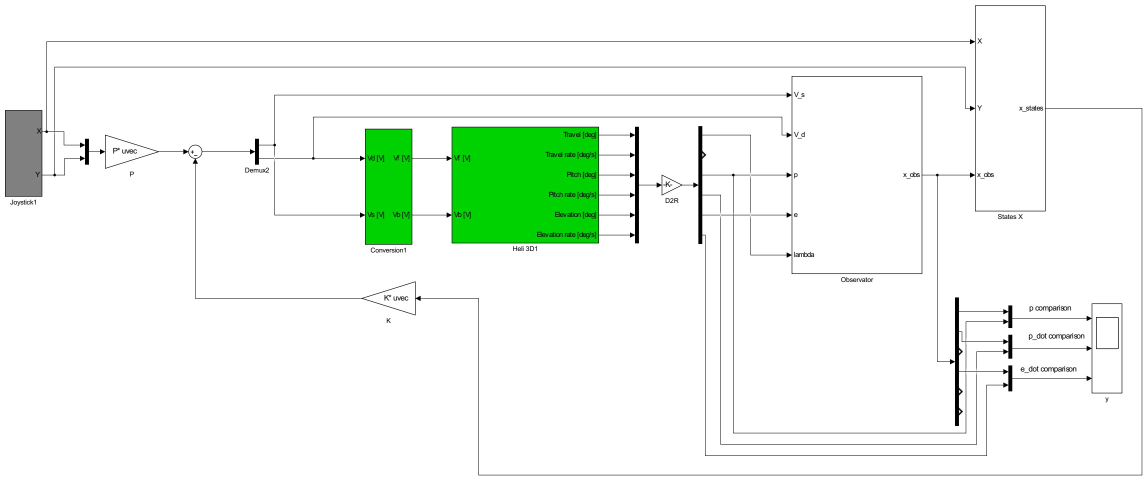

Figure 17: Part 4 - Problem 2, with integral effect

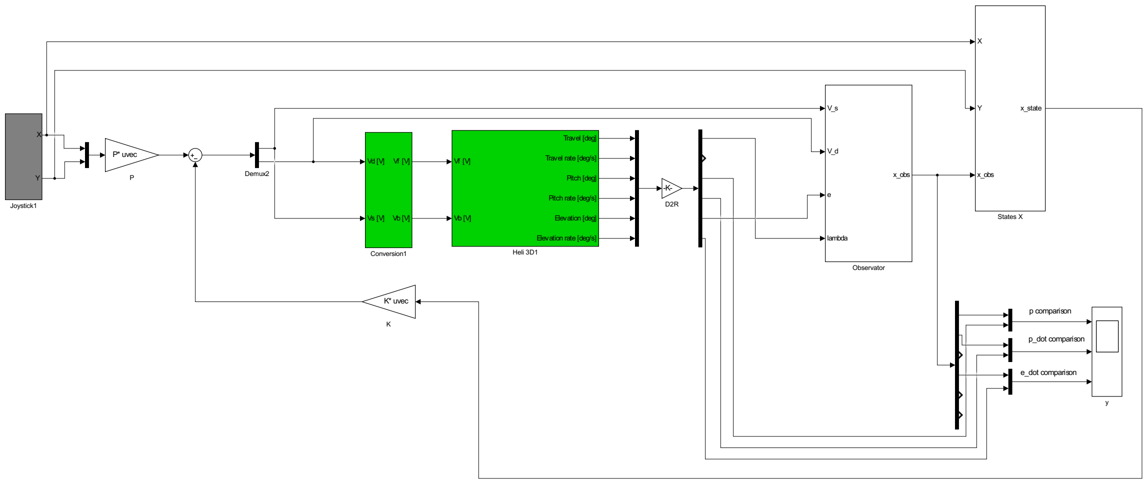

Figure 18: Part 4 - Problem 3

C MATLAB script

```
%%%%%%%%%%% Calibration of the encoder and the hardware for the specific
%%%%%%%%%%% helicopter
Joystick_gain_x = 1;
Joystick_gain_y = 1;


%%%%%%%%%%% Physical constants
g = 9.81; % gravitational constant [m/s^2]
l_c = 0.46; % distance elevation axis to counterweight [m]
l_h = 0.66; % distance elevation axis to helicopter head [m]
l_p = 0.175; % distance pitch axis to motor [m]
m_c = 1.92; % Counterweight mass [kg]
m_p = 0.72; % Motor mass [kg]


%---------------PART I---------------%

%%%%%%%%%%% Calibration of the encoder and the hardware for the specific
%%%%%%%%%%% helicopter
Joystick_gain_x = 3;
Joystick_gain_y = 8;

%%%%%%%%%%% Calculated values %%%%%%%%%%%
Elevation_offset = 29;
V_s = 6.2;
k_f = 0.1611

j_p = 2*m_p*(l_p)^2;
j_e = m_c*(l_c)^2+2*m_p*(l_h)^2;
j_lambda = m_c*(l_c)^2+2*m_p*((l_h)^2+(l_p)^2);

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_h*k_f;

K_1 = L_1/j_p;
K_2 = L_3/j_e;
K_3= L_4*V_s/j_lambda;

%---------------PART II---------------%

%---Problem 1---%

Kpp = 10;
Kpd = 8;

%---Problem 2---%

Kpp = 10;
Kpd = 6;
Krp = -1.5;

%---------------PART III---------------%

%---Problem 2---%

A = [0,1,0;0,0,0;0,0,0];
B = [0,0;0,K_1;K_2,0];
C = [1,0,0;0,0,1];
D = 0;
Q = diag([200,50,500]);
R = diag([0.5,1]);

K = lqr(A,B,Q,R);
P = inv(C*(inv(B*K-A))*B);

%---Problem 3---%

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];
```

```
D_i = 0;

Q_i = diag([50,10,200,10,100]);
R_i = diag([0.5,1]);
K_i = lqr(A_i,B_i,Q_i,R_i);
P = inv(C*(inv(B*K_i(1:2,1:3)-A))*B);
%---------------PART IV---------------%

%---Problem 2---%

A_o = [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_o = [0,0;0,K_1;0,0;K_2,0;0,0;0,0];
C_o = [1,0,0,0,0,0;0,0,1,0,0,0;0,0,0,0,1,0];
O = obsv(A_o,C_o);
Orank = rank(O);

s_p = eig(A-B*K_i(1:2,1:3));
r0 = max(abs(s_p));
fr = 20;
phi = pi/8;
r = r0*fr;
spread = -phi:(phi/2.5):phi;
p = -r*exp(1i*spread);

L = transpose(place(transpose(A_o),transpose(C_o),p));

%---Problem 3---%

C_o = [0,0,1,0,0,0;0,0,0,0,1,0];
O = obsv(A_o,C_o);
Orank = rank(O);

s_p = eig(A-B*K_i(1:2,1:3));
r0 = max(abs(s_p));
fr = 2;
phi = pi/8;
r = r0*fr;
spread = -phi:(phi/2.5):phi;
p = -r*exp(1i*spread);

L = transpose(place(transpose(A_o),transpose(C_o),p));
```

References

[1] Kristoffer Gryte, *TTK4115 - Helicopter lab assignment*, NTNU, Version 4.5, August 2015.

[2] Figure retrieved from [1]

[3] Morten D. Pedersen, *TTK4115 - Lecture 6*, NTNU, 2017.