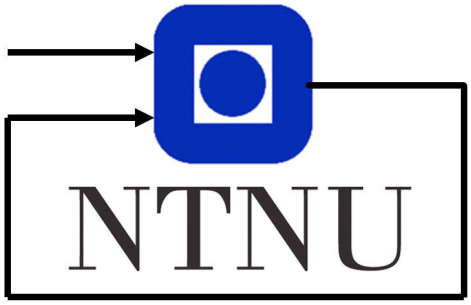
Helicopterlab Report

Group 29
Johan Brun 768630
Theodor Tveit Husefest 478428
Marius Bæver Larsen 478447

Oktober 21, 2018



Department of Engineering Cybernetics

Abstract

In this lab exercise we have derived a model for a mounted helicopter and controlled its flight in real time using feedforward, monovariable control, LQR and state observers. The exercise seek to better understand linear control and how the different parts of a linear system affect each other as well as highlight challenges in controlling multivariable systems and the limitations of estimated states.

Contents

1 Part 1 - Mathematical modeling

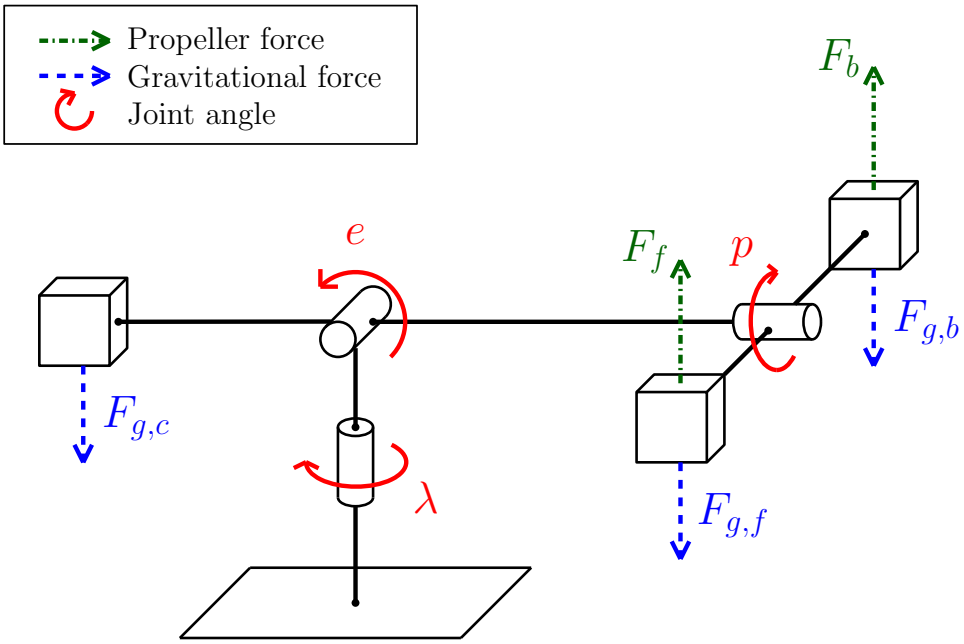


Figure 1: Forces working on the systems joint angles.

1.1 Problem 1 - Equations of Motion

Since our system (Figure 1) consists of masses rotating around an axis we use Newtons second law of rotation, $\Sigma\tau = I\alpha$, to compute our equations. In the following computations we use that $V_s = V_f + V_b$ and $V_d = V_f - V_b$, as well as $F_f = K_f V_f$ and $F_b = K_f V_b$.

The only forces that affect the pitch angle is the forces from the front and back rotor. Therefore we can say that the sum of torque is the product of arm and the difference of forces. We assume $F_{f,g} = F_{b,g}$.

$$
\begin{aligned}
\Sigma\tau &= J_p\ddot{p} \\
J_p\ddot{p} &= l_p((F_f - F_{g,f}) - (F_b - F_{g,b})) \\
J_p\ddot{p} &= l_p(F_f - F_b) = l_p(K_f V_f - K_f V_b) \\
J_p\ddot{p} &= l_p K_f V_d \\
J_p\ddot{p} &= L_1 V_d
\end{aligned} \tag{1.1}
$$

The elevation of the helicopter dependent of both the pitch and its own elevation, as both the gravitational force and the upward directed force from the motors are at their maximum when $e = 0$ and $p = 0$ respectively. As such the gravitational and motor-force components will include a cosine of e and p.

$$\begin{aligned}
\Sigma\tau &= J_e\ddot{e} \\
J_e\ddot{e} &= (l_cF_{g,k} - l_h(F_{g,f} + F_{g,b}))\cos e \\
&\quad + l_h((F_f - F_{g,f}) + (F_b - F_{g,b}))\cos p \\
J_e\ddot{e} &= (gl_cm_c - 2gl_hm_p))\cos e + l_hK_fV_s\cos p \\
J_e\ddot{e} &= L_2\cos e + L_3V_s\cos p
\end{aligned} \tag{1.2}$$

Since the arm working on the travel angle varies with the angle to both e and p, we have to include them in the equations.

$$\begin{aligned}
\Sigma\tau &= J_\lambda\ddot{\lambda} \\
J_\lambda\ddot{\lambda} &= (F_f + F_b)\cos e\sin p \\
J_\lambda\ddot{\lambda} &= K_fV_s\cos e\sin p \\
J_\lambda\ddot{\lambda} &= L_4V_s\cos e\sin p
\end{aligned} \tag{1.3}$$

The constants L_i, $i = \{1, 2, 3, 4\}$, are:

$$L_1 = l_pK_f \tag{1.4a}$$
$$L_2 = gl_cm_c - 2gl_hm_p \tag{1.4b}$$
$$L_3 = l_hK_f \tag{1.4c}$$
$$L_4 = -l_hK_f \tag{1.4d}$$

1.2 Problem 2 - Linearization

To use our model we linearize it around the point $(p, e, \lambda)^T = (p^*, e^*, \lambda^*)^T$ with $p^* = e^* = \lambda^* = 0$. Therefore $(\dot{p}, \dot{e}, \dot{\lambda})^T = \vec{0}$, and $(\ddot{p}, \ddot{e}, \ddot{\lambda})^T = \vec{0}$. From equations (1.1) and (1.2) we can calculate (V_s^*, V_d^*).

$$J_p0 = L_1V_d^* \implies V_d^* = 0$$

$$J_e 0 = L_2 \cos 0 + L_3 V_s^* \cos 0 \implies V_s^* = -\frac{L_2}{L_3}$$

$$\begin{bmatrix} V_s^* \\ V_d^* \end{bmatrix} = \begin{bmatrix} -\frac{L_2}{L_3} \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \tilde{p} \\ \tilde{e} \\ \tilde{\lambda} \end{bmatrix} = \begin{bmatrix} p \\ e \\ \lambda \end{bmatrix} - \begin{bmatrix} p^* \\ e^* \\ \lambda^* \end{bmatrix} \text{ and } \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} = \begin{bmatrix} V_s \\ V_d \end{bmatrix} - \begin{bmatrix} V_s^* \\ V_d^* \end{bmatrix} \tag{1.5}$$

Using our new coordinates from (1.5), we can rewrite our model equations from (1.1), (1.2) and (1.3).

$$\ddot{\tilde{p}} = \frac{L_1}{J_p} \tilde{V}_d \tag{1.6a}$$

$$\ddot{\tilde{e}} = \frac{L_2}{J_e} \cos \tilde{e} + (\tilde{V}_s - \frac{L_2}{L_3}) \frac{L_3}{J_e} \cos \tilde{p} \tag{1.6b}$$

$$\ddot{\tilde{\lambda}} = L_4 (\tilde{V}_s - \frac{L_2}{L_3}) \cos \tilde{e} \cos \tilde{p} \tag{1.6c}$$

We define $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix}$, and from (1.6) we get

$$\dot{\mathbf{x}} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{L_1}{J_p} \tilde{V}_d \\ x_4 \\ \frac{L_2}{J_e} \cos \tilde{e} + (\tilde{V}_s - \frac{L_2}{L_3}) \frac{L_3}{J_e} \cos \tilde{p} \\ x_6 \\ L_4 (\tilde{V}_s - \frac{L_2}{L_3}) \cos \tilde{e} \cos \tilde{p} \end{bmatrix} \tag{1.7}$$

We wish to write our system on the form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ where $\mathbf{A}$ and $\mathbf{B}$ are linearized around $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{u}_0 = (0, -\frac{L_2}{L_3})^T$.

$$\mathbf{A} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_6}{\partial x_1} & \cdots & \frac{\partial h_6}{\partial x_6} \end{bmatrix}\Bigg|_{\mathbf{x}_0,\mathbf{u}_0}, \mathbf{B} = \begin{bmatrix} \frac{\partial h_1}{\partial u_1} & \frac{\partial h_1}{\partial u_2} \\ \vdots & \vdots \\ \frac{\partial h_6}{\partial u_1} & \frac{\partial h_6}{\partial u_2} \end{bmatrix}\Bigg|_{\mathbf{x}_0,\mathbf{u}_0} \tag{1.8}$$

Using the equations from (1.7) and (1.8) we find our matrices.

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

Finally we end up with:

$$\ddot{\tilde{p}} = K_1 \tilde{V}_d \tag{1.9a}$$

$$\ddot{\tilde{e}} = K_2 \tilde{V}_s \tag{1.9b}$$

$$\ddot{\tilde{\lambda}} = K_3 \tilde{p} \tag{1.9c}$$

where $K_1 = \frac{L_1}{J_p}$, $K_2 = \frac{L_3}{J_e}$ and $K_3 = \frac{L_4}{J_\lambda} V_s^*$.

1.3 Problem 3 - Feed Forward

In this problem we connected the output of the joystick directly to the input of the helicopter in an attempt to control the helicopter. The pitch was constantly increasing in one direction, so we disconnected V_d in order to ensure that each motor got the same input, and observed that V_b had a slightly higher output than V_f. We reduced the input of V_b with a gain block in order to compensate for the error, and although the helicopter still was difficult to control it did a better job of balancing. After measuring V_s^* in section 1.4, we removed the gain block.

This change in pitch of the physical system is a clear difference from the theoretical models, since the pitch should not change while V_d and $\tilde{V}_d$ are zero. When we disconnected V_d (i.e. it was always zero) the helicopter was still tilting. This error propagates into the theoretical models for travel and elevation causing discrepancies in them as well.

We believe the error was caused by a difference in the motors resulting in one giving a higher output than the other as well as inaccurate physical constants, as well as drifting in sensors.

1.4 Problem 4 - Equilibrium

By measurement we found that we needed to adjust the elevation by $-32°$ so that the helicopter was horizontal when $e = 0$. We found it challenging to measure V_s^* as it was difficult to keep the helicopter at a stable elevation. By supporting the helicopter at horizontal elevation and gradually increasing V_s until the helicopter started supporting itself we found:

$$V_s^* = 8.2V$$

and by inserting it into (1.2) we get:

$$K_f = 0.1218$$

2 Part 2 - Monovariable control

2.1 Problem 1 - Pitch controller

The PD controller for the system is given as

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

By inserting (1.9a) into (2.1) we can apply the Laplace transform to find the transfer function $\frac{\tilde{p}(s)}{\tilde{p}_c(s)}$. Since $\tilde{p}(0) = \dot{\tilde{p}}(0) = 0$, due to the property of linearizing around the equilibrium point, the equation can simply be reduced to

$$\tilde{p}(s)(\frac{s^2}{K_1} + sK_{pd} + K_{pp}) = \tilde{p}_c(s)K_{pp}$$

$$\implies \frac{\tilde{p}(s)}{\tilde{p}_c(s)} = \frac{K_1 K_{pp}}{(s^2 + K_1 K_{pd} s + K_1 K_{pp})} \tag{2.2}$$

Next we need to find reasonable values for K_{pp} and K_{pd}. To achieve a critical damped system we are ensuring repeated poles on the real negative axis. This is done by comparing our transfer function with a general second-order linear system.

$$\frac{K\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{2.3}$$

with damping ratio $\zeta = 1$. By comparing (2.2) with (2.3), we can express the regulator parameters as functions of ω_0

$$K_{pp} = \frac{\omega_0^2}{K_1} \tag{2.4a}$$

$$K_{pd} = \frac{2\omega_0}{K_1} \tag{2.4b}$$

K_{pp} and K_{pd} are directly related to ω_0 and in turn to the poles of the closed-loop system, as seen by looking at the denominator of the system $(s + \omega_0)^2$. This means that when we are increasing K_{pp} and K_{pd} we are moving the eigenvalues of the system further into the left plane, giving the system a faster response. There are however physical limitations to how fast the response can be before the system starts oscillating or even becomes unstable.

The PD-controller where implemented in Simulink, as shown in Figure 8 Appendix B. Next, the system where tuned by starting with a low value for ω_0 and increasing it gradually. As expected, a higher ω_0 gave a faster response, which was desired. After testing both low values, that gave a too slow system, and high values, that gave oscillating behavior, we ended up with a value of $\omega_0 = 2.60$ which gave a satisfactory response, rapid and no rise for oscillating behaviour. By inserting ω_0 in (2.4) we got the parameter values

$$K_{pp} = 13.99$$
$$K_{pd} = 10.76$$

Since the system only has poles in the left half plane, in theory it should be stable for any value of ω_0. However, there is an unavoidable time-delay in the system which makes the system non-linear and unstable if K_{pp} (ω_0) is too big. Additionally, factors like backlash in the mechanics and saturation of the motors can make the system non-linear and give rise to the oscillating behaviour we experienced with high ω_0.

2.2 Problem 2 - Travel controller

The travel rate $\dot{\tilde{\lambda}}$ was to be controlled by a simple P controller:

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

By assuming $\tilde{p} = \tilde{p}_c$ and inserting $\ddot{\lambda}$ from (1.9) we get

$$\begin{aligned} \frac{\ddot{\lambda}}{K_3} &= K_{rp}(\dot{\lambda}_c - \dot{\lambda}) \\ \ddot{\lambda}(t) &= K_3 K_{rp}(\dot{\lambda}_c(t) - \dot{\lambda}(t)) \\ \overset{\mathcal{L}}{\Rightarrow} \frac{\dot{\tilde{\lambda}}(s)}{\dot{\tilde{\lambda}}_c(s)} &= \frac{K_3 K_{rp}}{s + K_3 K_{rp}} \\ \frac{\dot{\tilde{\lambda}}(s)}{\dot{\tilde{\lambda}}_c(s)} &= \frac{\rho}{s + \rho} \end{aligned} \tag{2.6}$$

where $\rho = K_3 K_{rp}$. The laplace transformation was done with respect to $\dot{\tilde{\lambda}}$. By adding the P controller (2.5) to the Simulink diagram, as shown in Figure 9 Appendix B, we where able to test different values of K_{rp}. We

observed that values higher than -0.50 gave a too slow response and the system got impossible to control with values below -2.00. We where satisfied with the value $K_{rp} = -0.75$ as it was quick and safe to control.

3 Part 3 - Multivariable control

3.1 Problem 1 - State space equation

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.1}$$

$$\mathbf{x} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \end{bmatrix} \text{ and } \mathbf{u} = \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix}$$

By using equations for the linearized system from (1.9) we got the state-space equation in (3.2)

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\tilde{p}} \\ \ddot{\tilde{p}} \\ \ddot{\tilde{e}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} \tag{3.2}$$

3.2 Problem 2 - LQR P-controller

We will now track the reference $\mathbf{r} = [\tilde{p}_c, \dot{\tilde{e}}_c]^T$ which will be given by the joystick. The x-axis input will give the pitch reference and the y-axis will give the elevation rate reference, implying that the helicopter should stay at the current elevation when the elevation-rate reference is zero.

To start we calculate the controllability matrix.

$$\mathcal{C} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & K_1 & 0 & 0 \\ 0 & K_1 & 0 & 0 & 0 & 0 \\ K_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

$\mathcal{C}$ clearly has full rank, meaning that the set ($\mathbf{A}$, $\mathbf{B}$) is controllable and that we can implement a controller in Simulink on the form

$$\mathbf{u} = \mathbf{Pr} - \mathbf{Kx} \tag{3.4}$$

as shown in Figure 11 Appendix B.

The matrix $\mathbf{K}$ corresponds to the linear quadratic regulator (LQR) for which the control input $\mathbf{u} = -\mathbf{Kx}$ optimizes the cost function.

$$J = \int_0^\infty (\mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t))dt \tag{3.5}$$

We calculated $\mathbf{K}$ using the MATLAB command `K = lqr(A, B, Q, R)`. $\mathbf{Q}$ and $\mathbf{R}$ are weighting matrices for the state and input respectively, which for simplicity's sake will be diagonal. At first both $\mathbf{Q}$ and $\mathbf{R}$ where set to identity matrices, and as we want a fast and accurate response we increased the weighting of the states in $\mathbf{Q}$, while $\mathbf{R}$ remained the same. By trial and error we concluded with 80, 10 and 100 down the diagonal of $\mathbf{Q}$ (A.3).

$\mathbf{P}$ is chosen such that in theory $\lim_{t\to\infty} \tilde{p}(t) = \tilde{p}_c$, $\lim_{t\to\infty} \dot{\tilde{e}}(t) = \dot{\tilde{e}}_c$ and since $\dot{\tilde{e}}_c$ and $\tilde{p}_c$ are fixed values, $\dot{\tilde{p}}(t) = 0$ for $\lim_{t\to\infty}$. This implies that

$$\dot{\mathbf{x}}_\infty = 0 \tag{3.6a}$$

$$\mathbf{x}_\infty = \begin{bmatrix} \tilde{p}_c \\ 0 \\ \dot{\tilde{e}}_c \end{bmatrix} \Rightarrow \mathbf{r} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{C}} \mathbf{x}_\infty \tag{3.6b}$$

By inserting (3.4) in (3.1), the system can be presented as

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}(\mathbf{P}\mathbf{r} - \mathbf{K}\mathbf{x}) = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} + \mathbf{B}\mathbf{P}\mathbf{r}$$

Furthermore, (3.6a) gives

$$\mathbf{B}\mathbf{P}\mathbf{r} = (\mathbf{B}\mathbf{K} - \mathbf{A})\mathbf{x}_\infty \tag{3.7}$$

And by inserting (3.6b), an equation for $\mathbf{P}$ is given by

$$\mathbf{P} = [\mathbf{C}(\mathbf{B}\mathbf{K} - \mathbf{A})^{-1}\mathbf{B}]^{-1} \tag{3.8}$$

Next, let $\mathbf{K}$ be on the form

$$\mathbf{K} = \begin{bmatrix} k_1 & \dots & k_m \\ \vdots & \ddots & \vdots \\ k_n & \dots & k_{n*m} \end{bmatrix}$$

and is found by MATLAB, (A.3), using Equation 3.8. Then $\mathbf{P}$ is the opposite diagonal matrix

$$\mathbf{P} = \begin{bmatrix} 0 & k_3 \\ k_4 & 0 \end{bmatrix} \tag{3.9}$$

With k_3 and k_4 being elements of $\mathbf{K}$.

Although the helicopter was easier to control with the LQR P-controller, we did not get quite the desired response. When giving a elevation rate reference, we want the helicopter to keep its elevation. It did not manage this, due to steady-state error.

3.3 Problem 3 - Adding integral effect

As the helicopter where not able to keep its height due to steady-state error, an integral effect to the controller where to be added for both pitch angle and elevation rate (achieving a PI-controller). This means two more states had to be added to our state vector $\mathbf{x}$

$$\begin{aligned} \dot{\gamma} &= \tilde{p} - \tilde{p}_c \\ \dot{\zeta} &= \dot{\tilde{e}} - \dot{\tilde{e}}_c \end{aligned} \tag{3.10}$$

which leaves us the new state vector and same input vector

$$\mathbf{x} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \\ \gamma \\ \zeta \end{bmatrix} \text{ and } \mathbf{u} = \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} \tag{3.11}$$

New matrices $\mathbf{A}$ and $\mathbf{B}$ where calculated using our linearized equations (1.9)

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \dot{\tilde{e}} \\ \gamma \\ \zeta \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} \tag{3.12}$$

As the new $\mathbf{K}$ was given by the LQR command in MATLAB, we had to tune the two new states with our $\mathbf{Q}$ matrix. During our trial and error tuning process, we ended up penalizing the states $\tilde{p}$, $\dot{\tilde{e}}$ and γ and got a tuning we thought was accurate and robust. We used the same $\mathbf{P}$ as in section 3.2.

$$\mathbf{Q} = \begin{bmatrix} 90 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 60 \end{bmatrix} \tag{3.13}$$

By implementing the PI-controller to the system, the steady-state error was removed and the helicopter where able to keep its elevation a lot better then with only a P-controller, which is shown in Figure 2.

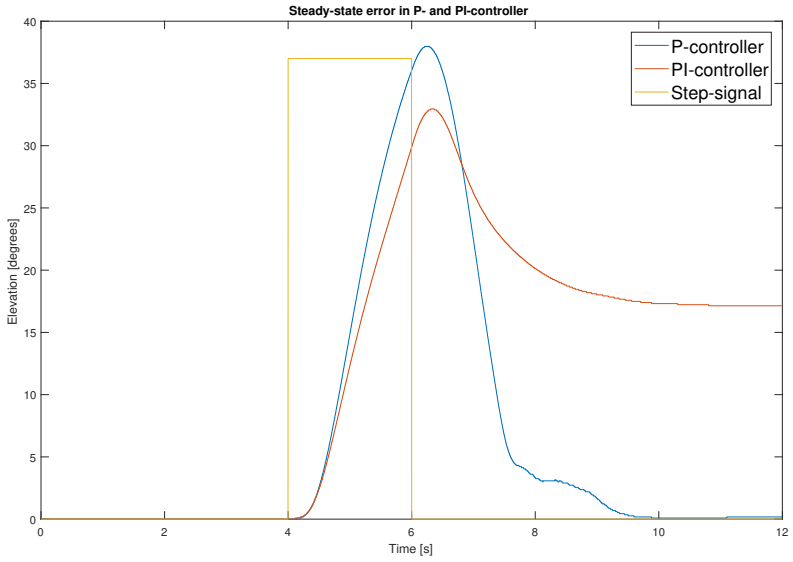


Figure 2: Plot of step response when using P- and PI-controller. We can see that the PI-controller does a much better job at keeping elevation.

4 State-estimation

4.1 Problem 1 - State-space equation

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

$$\mathbf{x} = \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} \text{ and } \mathbf{y} = \begin{bmatrix} \tilde{p} \\ \tilde{e} \\ \tilde{\lambda} \end{bmatrix} \tag{4.1}$$

By using our linearized system from (1.9) we find our matrices **A**, **B** and **C** and get our state-space model.

$$\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ K_3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \tilde{p} \\ \dot{\tilde{p}} \\ \tilde{e} \\ \dot{\tilde{e}} \\ \tilde{\lambda} \\ \dot{\tilde{\lambda}} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & K_1 \\ 0 & 0 \\ K_2 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \tilde{V}_s \\ \tilde{V}_d \end{bmatrix} \tag{4.2}$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{C}} \mathbf{x} \tag{4.3}$$

4.2 Problem 2 - Linear Observer

To check the observability of the system we used Matlab's function `obsv(A,C)` (see appendix A.5), which gives the observability matrix (4.4). We found that it has full rank and every state is observable. We can therefore implement our linear observer (equation (4.5)) in Simulink, see Figure 13 Appendix B.

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA^{n-1}} \end{bmatrix} \tag{4.4}$$

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

In order to find our observer gain matrix $\mathbf{L}$ we needed to know more about the plant, since the error correction has to be faster than the plant dynamics. Since $\mathbf{u} = \mathbf{Pr} - \mathbf{Kx}$, the system can be written as $\dot{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})\mathbf{x} + \mathbf{Pr}$ and therefore we examine the eigenvalues of $\mathbf{A} - \mathbf{BK}$.
The dynamics of our linear observer (equation (4.5)) is determined by $\mathbf{A} - \mathbf{LC}$, and we chose to have the poles of $\mathbf{L}$ placed in a fan with radius 20 times larger than the radius of the largest pole of $\mathbf{A} - \mathbf{BK}$. We experimented with different values, but 20 times larger gave a fast observer and acceptable levels of noise.
We did this for both the P-controller and the PI-controller, and as we were satisfied by our controller in part 3, we chose not to alter our LQR-controllers. Using our estimated states, we were able to control the helicopter without noticeable difference compared to using measured states.

In Figure 3 and Figure 4 we have plotted the states we use in our controllers. We observe that the estimated stated follows the measured state accurately with P-controller and PI-controller respectively.

4.3 Problem 3 - Observer Without Pitch

$$\mathbf{y_1} = \begin{bmatrix} \tilde{e} \\ \tilde{\lambda} \end{bmatrix} \text{ and } \mathbf{y_2} = \begin{bmatrix} \tilde{p} \\ \tilde{e} \end{bmatrix}$$

Likewise to problem 2, we use Matlab to examine the observability of the system (see appendix A.7), given our given measurement vectors $\mathbf{y_1}$ and $\mathbf{y_2}$, and found our observability matrices $\mathcal{O}_1$ and $\mathcal{O}_2$. We found that $\mathcal{O}_2$ only has rank = 4 and we unable to estimate all the states in our system using $\mathbf{y_2}$. $\mathcal{O}_1$ however, has full rank, so we can use $\mathbf{y_1}$ as a measurement vector. To explain this we use equation (1.9) and observe that it is possible to differentiate the travel angle twice to obtain pitch. If we wish to find the travel angle from the pitch measurement, we have to integrate the pitch twice and we get unknown constants.

Even though it is possible to observe all states with $\mathbf{y_1}$, we did not get a very good result when doing it. We got a good estimate for $\dot{\tilde{e}}$, which is understandable as we measure $\tilde{e}$. However, for $\tilde{p}$ and $\dot{\tilde{p}}$ it is different. As your differentiate a measurement the high-frequency noise will be gained a lot more than the desired signal. And in our system, to find the pitch-rate using $\mathbf{y_1}$, we have to have to differentiate λ three times.
In Figure 5 we find the observer gain matrix $\mathbf{L}$ the same way as in problem 2

(appendix A.7). This means that we gain the already noisy signal, and our result is completely unusable in a controller. Note the units on the y-axis is [rad] and [rad/s] for $\tilde{p}$ and $\dot{\tilde{p}}$, and compared to the noise, the signal is lost.

To reduce the noise in our estimated states we reduced the radius of the poles of $\mathbf{L}$ considerably, and specially the poles which corresponds to pitch and pitch-rate. which gave us a slower, but usable observer. We also placed our poles on the real axis. We have done this in Figure 6, and we can see that we still have good estimation of $\dot{e}$, but now we can see that the estimated states are a lot closer to the measured states. However, it is still very hard to control the helicopter. Pole-placement of $\mathbf{L}$ can be seen in appendix A.7.

4.4 Plots

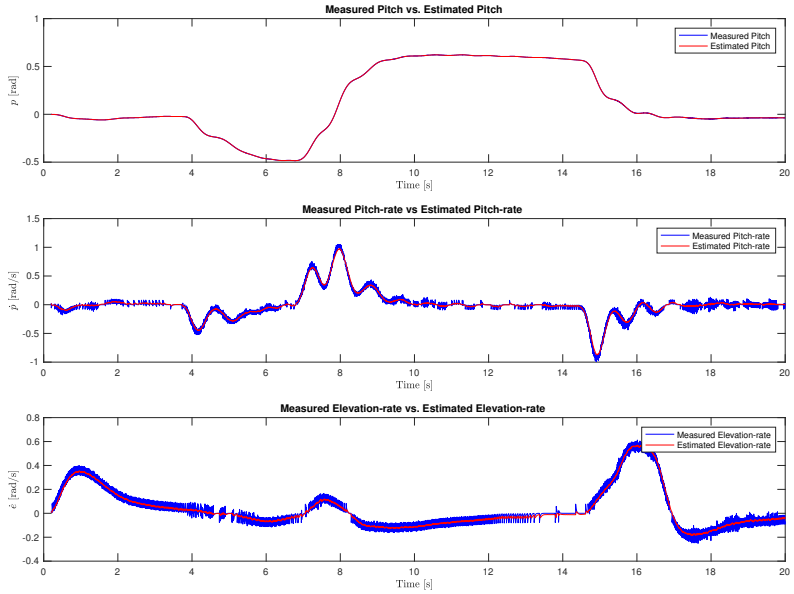


Figure 3: Plot from part 4, problem 2 with P controller. Shows estimated value (red) and measured value (blue) for $\tilde{p}$, $\dot{\tilde{p}}$ and $\dot{\tilde{e}}$ respectively.

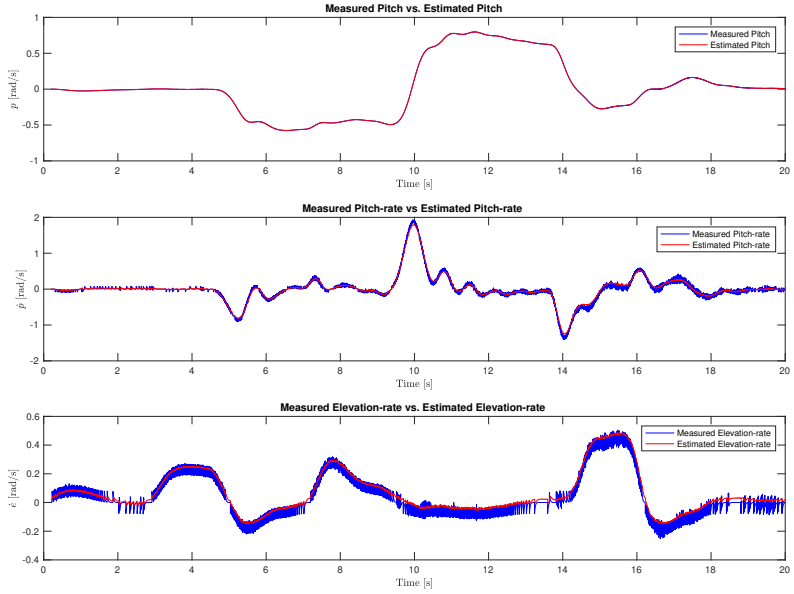


Figure 4: Plot from part 4, problem 2 with PI controller. Shows estimated value (red) and measured value (blue) for $\tilde{p}$, $\dot{\tilde{p}}$ and $\dot{\tilde{e}}$ respectively.

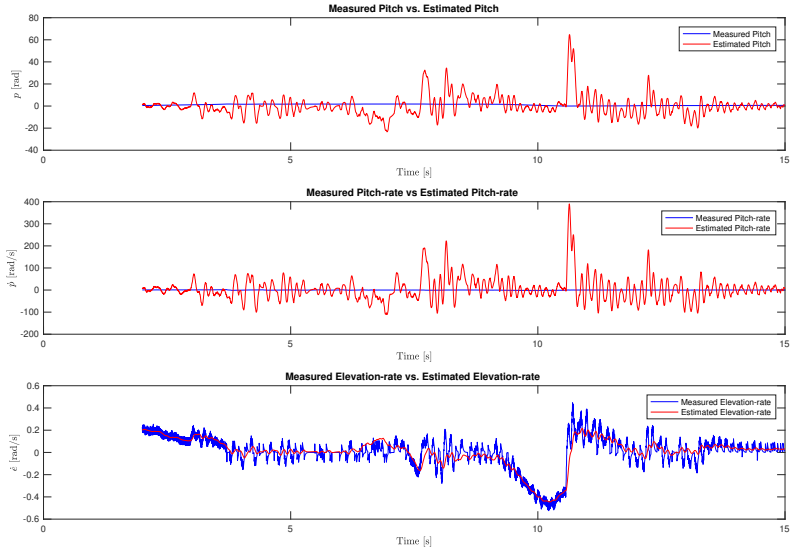


Figure 5: Plot from part 4, problem 3 with PI controller. Shows estimated value (red) and measured value (blue) for $\tilde{p}$, $\dot{\tilde{p}}$ and $\dot{\tilde{e}}$ respectively.

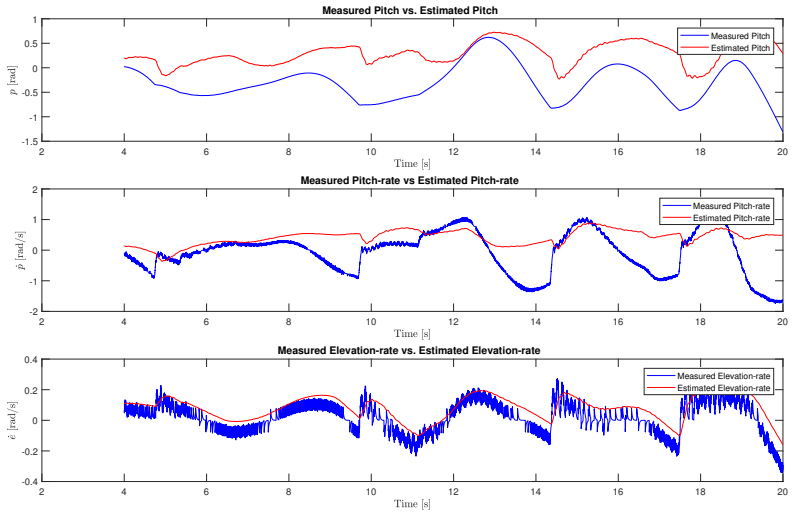


Figure 6: Plot from part 4, problem 3 with PI controller where the poles of $\mathbf{L}$ have been adjusted. Shows estimated value (red) and measured value (blue) for $\tilde{p}$, $\dot{\tilde{p}}$ and $\dot{\tilde{e}}$ respectively.

A MATLAB Code

A.1 Part 1

```
1
2  Vs_star = 8.2; % Measured sum voltage at equilibrium
3  K_f = g*(2*l_h*m_p-l_c*m_c)/(l_h*Vs_star); %Motor Force Constant
4
5  % Calculate constants for model equations
6  L_1 = l_p*K_f;
7  L_2 = g*l_c*m_c - 2*g*l_h*m_p;
8  L_3 = l_h*K_f;
9  L_4 = -l_h*K_f;
10
11 J_p = 2*m_p*l_p^2;
12 J_e = m_c*l_c^2 + 2*m_p*l_h^2;
13 J_lambda = m_c*l_c^2 + 2*m_p*(l_h^2 + l_p^2);
14
15 % Calculated constants for linearized model
16 K_1 = L_1/J_p;
17 K_2 = L_3/J_e;
18 K_3 = (L_4/J_lambda) * Vs_star;
```

A.2 Part 2

```
1
2  w0 = 2.6;
3  Kpp = w0^2/K_1;
4  Kpd = 2*w0/K_1;
5
6
7  Krp = -0.75;   % Travelrate controller constant
```

A.3 Part 3 - Problem 2

```
1
2  % State space matrices for given statevector
3  A_L = [0 1 0; 0 0 0; 0 0 0];
4  B_L = [0 0; 0 K_1; K_2 0];
5
6  % Chosen Q and R matrices
7  Q_L = [80 0 0; 0 10 0; 0 0 100];
8  R_L = [1 0 ; 0 1];
9
```

```
% Calculate K and P with MatLab's lqr-function
K_L = lqr(A_L, B_L, Q_L, R_L);
P_L = [0 K_L(1,3); K_L(2,1) 0];
```

A.4 Part 3 - Problem 3

```

% State space matrices for given state-vector
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];
% Chosen Q and R matrices
Q_I = [90 0 0 0 0;
       0 10 0 0 0;
       0 0 100 0 0;
       0 0 0 100 0;
       0 0 0 0 60];

R_I = [1 0;
       0 1];
% K and P calculated with MatLab's lqr-function
K_I = lqr(A_I, B_I, Q_I, R_I);
P_I = [0 K_I(1,3); K_I(2,1) 0];
```

A.5 Part 4 - Problem 2 with P-controller

```
% Matrices for state-estimation
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];

% Check observability
O = obsv(A_E,C_E);  % Calculate observabillity matrix
rank(O); % rank(O) = 6, so every state is observable

% Calculate the poles of the gain matrix L
% with state-space model from part 3 - problem 2
sys_poles_L = eig(A_L - B_L*K_L);
r0 = max(abs(sys_poles_L));  % Find largest radius

r = r0*20;
angle = pi/10;

spread = -angle:(angle/2.5):angle;
poles_L_L = -r*exp(spread*1i);

% Use Matlab's place-function to calculate L
L_L = place(transpose(A_E), transpose(C_E),poles_L_L).';
```

A.6 Part 4 - Problem 2 with PI-controller

```

% Calculate the poles of the gain matrix L
% with state-space model from part 3 - problem 3
sys_poles_I = eig(A_I - B_I*K_I);
r0 = max(abs(sys_poles_I));

r = 20*r0;
angle = pi/10;

spread = -angle:(angle/2.5):angle;
poles_L_I = -r*exp(spread*1i);

% Use MatLab's place-function to calculate L
L_I = place(transpose(A_E), transpose(C_E),poles_L_I).';
```

A.7 Part 4 - Problem 3

```
1
2  C_E_1 = [0 0 1 0 0 0;
3      0 0 0 0 1 0];
4
5  C_E_2 = [1 0 0 0 0 0;
6      0 0 1 0 0 0];
7
8  % Check observability of C_E_1
9  O_1 = obsv(A_E,C_E_1);  % Calculate observabillity matrix
10 rank(O_1); % rank(O_1) = 6, so every state is observable
11
12 % Check observability of C_E_2
13 O_2 = obsv(A_E,C_E_2);  % Calculate observabillity matrix
14 rank(O_2); % rank(O_2) = 4, so two states are not observable
15
16 % Place the poles on the real axis
17 poles_L_I = -[1.5 2 10 12 14 16];
18
19 scatter(real(poles_L_I),imag(poles_L_I));
20
21 % Use MatLab's place-function to calculate L
22 L_I = place(transpose(A_E), transpose(C_E_1), poles_L_I).',
```

B Simulink Diagrams

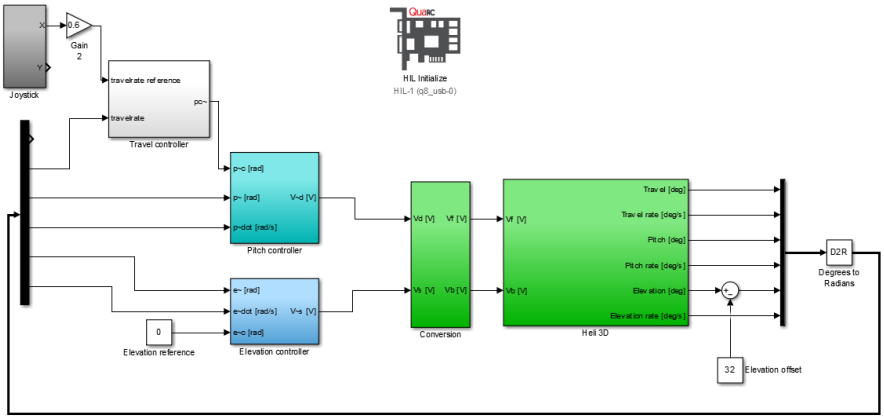


Figure 7: Overview Simulink diagram from Part 2. Note that in problem 1, the travel-controller is not connected.

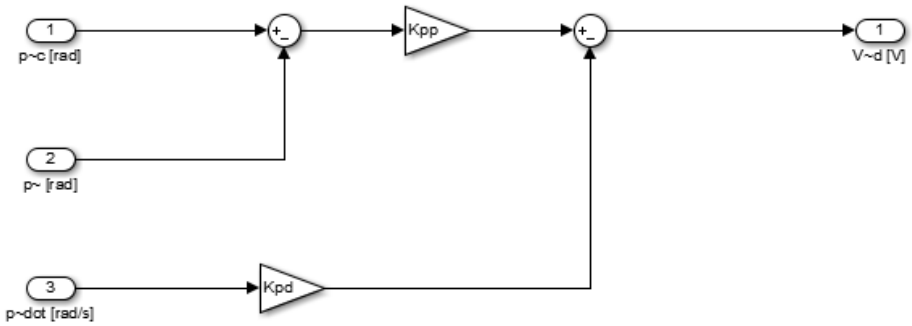


Figure 8: Pitch-controller Simulink diagram from Part 2.

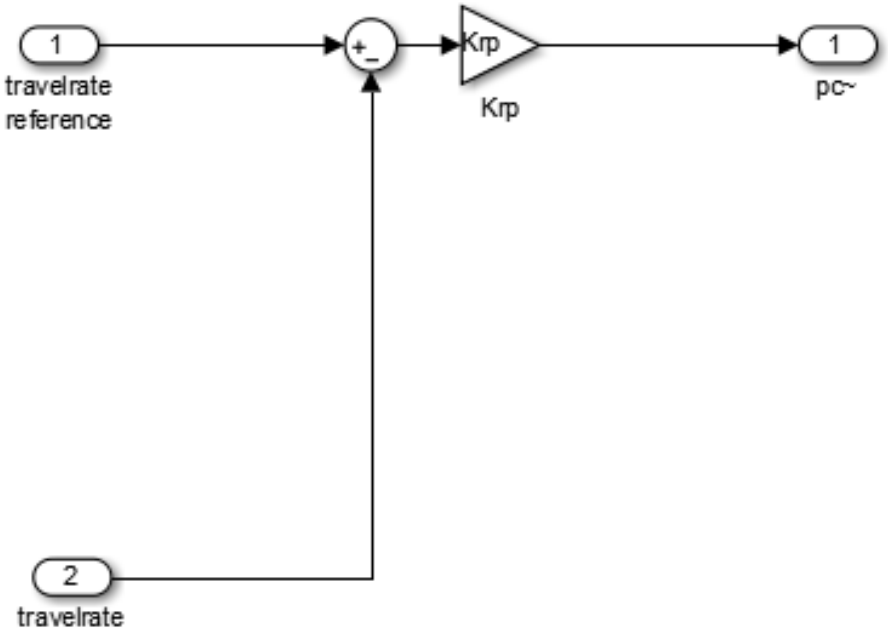


Figure 9: Travel-controller Simulink diagram from Part 2, problem 2.

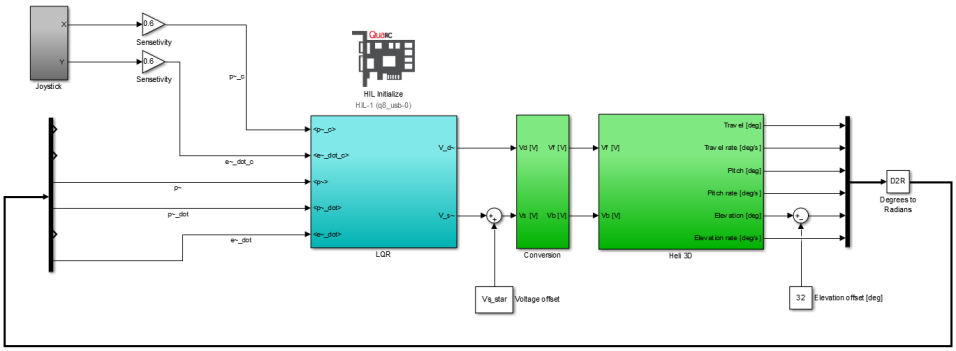


Figure 10: Overview Simulink diagram from Part 3.

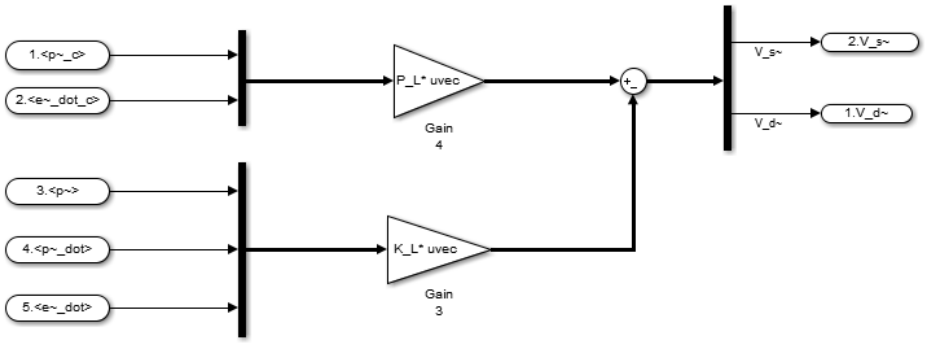


Figure 11: P-controller with LQR Simulink diagram from Part 3, problem 2.

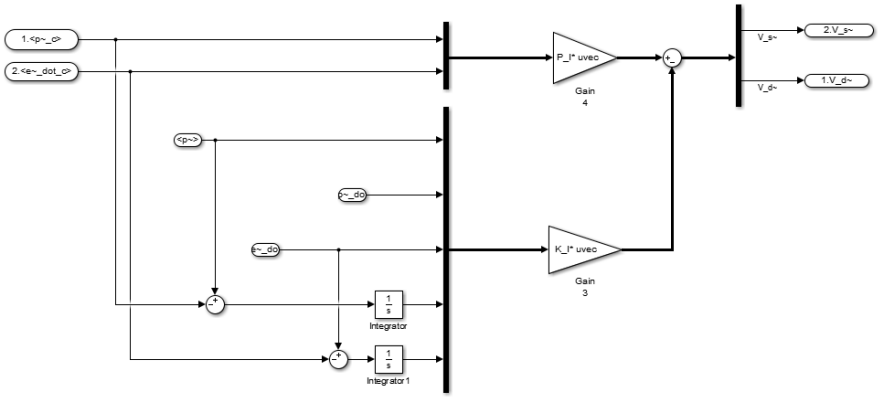


Figure 12: PI-controller with LQR Simulink diagram from Part 3, problem 3.

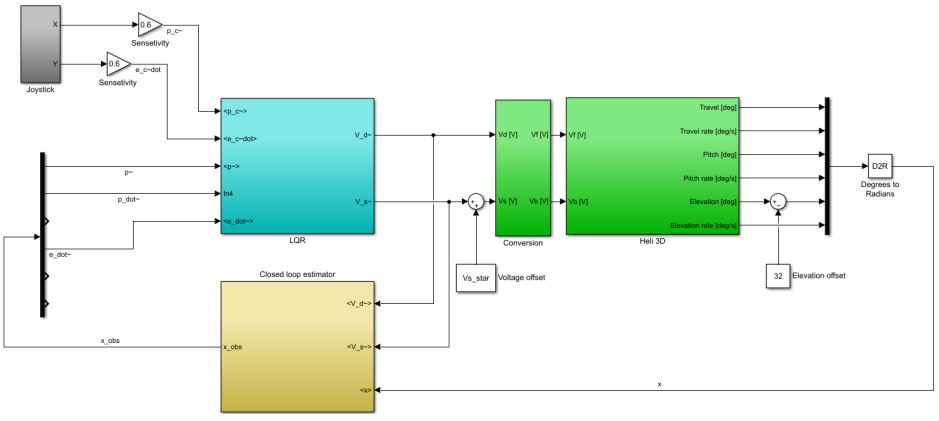


Figure 13: Overview Simulink diagram from Part 4.

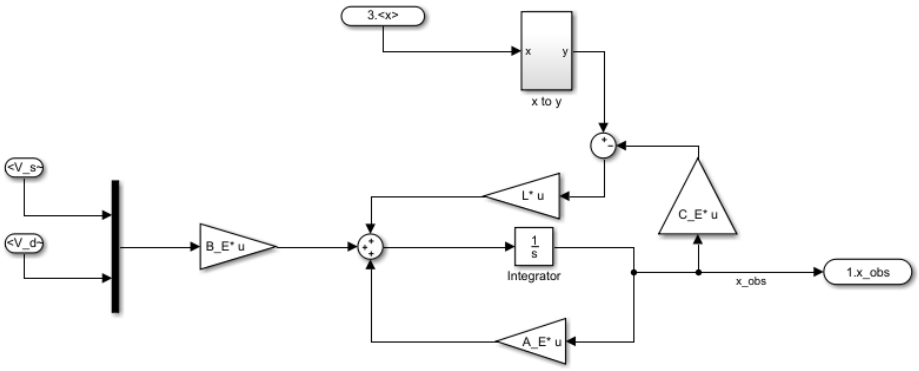


Figure 14: Simulink Diagram for closed loop estimator in part 4.