TTK4135 Helicopter lab report

Student: 506889
Student: 507766

April 10, 2021

Department of Engineering Cybernetics

Abstract

This report documents Helicopter lab in the course Optimization and Control at NTNU in the spring of 2021. The purpose of the lab is to solve the optimization problem for the helicopter. The project can be broken down into two main parts. The first part is optimization of the helicopters trajectory and input sequence. The second part consist of implementing feedback with a linear quadratic (LQ) controller. The project was divided into three days, the first two to implement optimisation and LQcontroller for four states and the third for all six states. The report is therefore divided into each day. Throughout the report we will associate the theory given in the course with our observations at the lab.

Contents

1 Model description

The helicopter consists of a base with an arm attached, as shown in figure 41.

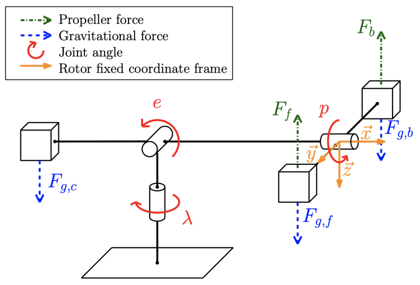


Figure 1: Helicopter lab set up

The position and trajectory of the helicopter can be represented by the angle of the elevation, pitch and travel and their corresponding rates. Which can be represented by x given by

$$\boldsymbol{x} = \begin{bmatrix} \lambda & r & p & \dot{p} & e & \dot{e} \end{bmatrix}^\top \tag{1}$$

A combination of equation of moment balance and PID controllers for the pitch and elevations result in the following equations of the derivative of the state given in equation (1).

$$\dot{\lambda} = r \tag{2a}$$
$$\dot{r} = -K_2 p \tag{2b}$$
$$\dot{p} = \dot{p} \tag{2c}$$
$$\ddot{p} = -K_1 K_{pp} p - K_1 K_{pd} \dot{p} + K_1 K_{pp} p_c \tag{2d}$$
$$\dot{e} = \dot{e} \tag{2e}$$
$$\ddot{e} = -K_3 K_{ep} e - K_3 K_{ed} \dot{e} + -K_3 K_{ep} e_c \tag{2f}$$

Where the constants in the equations are respectively

Table 1: Parameters and values

Symbol	Parameter	Value	Unit
L_a	Distance from elevation axis to helicopter body	0.63	m
l_h	Distance from pitch axis to motor	0.18	m
K_f	Force constant motor	0.25	$\frac{N}{V}$
J_e	Moment of inertia for elevation	0.83	kg m^2
J_t	Moment of inertia for travel	0.83	kg m^2
J_p	Moment of inertia for pitch	0.034	kg m^2
m_h	Mass of helicopter	1.05	kg
m_w	Balance weight	1.87	kg
m_g	Effective mass of the helicopter	0.05	kg
K_p	Force to lift the helicopter from the ground	0.49	N

Table 2: Variables

Symbol	Variable
p	Pitch
p_c	Setpoint for pitch
λ	Travel
r	Speed of travel
r_c	Setpoint for speed of travel
e	Elevation
e_c	Setpoint for elevation
V_f	Voltage, motor in front
V_b	Voltage, motor in back
V_d	Voltage difference, $V_f - V_b$
V_s	Voltage sum, $V_f + V_b$
$K_{pp}, K_{pd}, K_{ep}, K_{ei}, K_{ed}$	Controller gains
T_g	Moment needed to keep the helicopter flying

2 10.2 - Optimal Control of Pitch/Travel without Feedback

In this section we will implement the optimal control of the trajectory $\boldsymbol{x}^*$ and the input sequence $\boldsymbol{u}^*$. This will drive the helicopter from an initial state $\boldsymbol{x_0}$ to a preferred state $\boldsymbol{x_f}$. This can be realised by solving the cost function given by equation (11). To attain this we disregard the elevation and set both the elevation angle and elevation rate to zero. Which gives us the trajectory $\boldsymbol{x}$ and input $\boldsymbol{u}$ given by respectively equation (3) and (4).

$$\boldsymbol{x} = \begin{bmatrix} \lambda & r & p & \dot{p} \end{bmatrix}^\top \tag{3}$$

$$\boldsymbol{u} = p_c \tag{4}$$

2.1 The continuous model

From the time derivatives of the states given by equations (2a)-(2d), we derive the continuous time state space model

$$\dot{\boldsymbol{x}} = \boldsymbol{A_c}\boldsymbol{x} + \boldsymbol{B_c}\boldsymbol{u} \tag{5}$$

with corresponding matrices

$$\boldsymbol{A_c} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} \end{bmatrix}, \quad \boldsymbol{B_c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1K_{pp} \end{bmatrix} \tag{6}$$

Thus this model is solely based on the helicopters travel, travel rate, pitch and pitch rate. It consists of a physical layer and a basic control layer, as illustrated in figure 2. Therefore, it only delivers feedback to the basic control layer, containing the PID- and PD-controller, and not to the optimization layer. This also corresponds to our simulink diagram shown in figure 9, which is based on the hierarchy, where the feedback is only sent to the controllers, and not back to the model-based optimization. However, in this section we focus on the optimization layer that delivers the optimal input to these regulators.

The model is based on linearization of simplified equation and several assumptions, like disregarding elevation and considering the angles and rates as decoupled. Therefore, we have to keep in mind that in the regulation of a real helicopter, this model might not work as well.

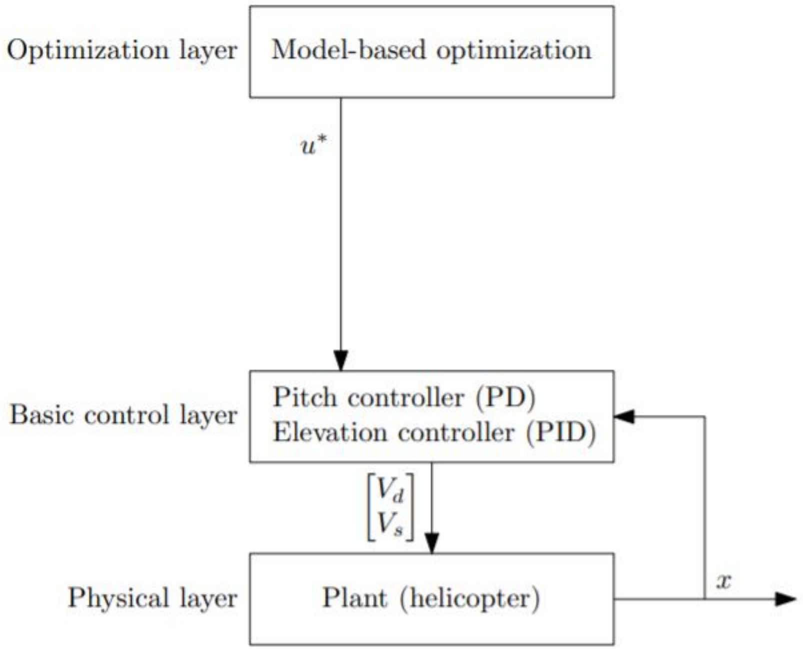


Figure 2: Layers in the control hierarchy without feedback. Copied from assignment text.

2.2 The discretized model

The discretized model was found by using the forward Euler method [1]. $\dot{\boldsymbol{x}}$ can be rewritten as $\frac{\boldsymbol{x_{k+1}}-\boldsymbol{x_k}}{\Delta t}$. By substituting this into equation (5), we get

$$\frac{\boldsymbol{x_{k-1}} - \boldsymbol{x_k}}{\Delta t} = \boldsymbol{A_c x_k} + \boldsymbol{B_c u_k} \tag{7}$$

From the definition of discrete time state space form we have

$$\boldsymbol{x_{k+1}} = (\boldsymbol{I} + \Delta t \boldsymbol{A_d})\boldsymbol{x_k} + \Delta t \boldsymbol{B_c u_k} = \boldsymbol{A_d x_k} + \boldsymbol{B_d u_k} \tag{8}$$

with matrices given at $\Delta t = 0.25$

$$\boldsymbol{A_d} = (\boldsymbol{I}+\Delta t \boldsymbol{A_d}) = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & -\Delta t K_2 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & -\Delta t K_1 K_{pp} & 1 - \Delta t K_1 K_{pd} \end{bmatrix} = \begin{bmatrix} 1.0000 & 0.2500 & 0 & 0 \\ 0 & 1.0000 & -0.1415 & 0 \\ 0 & 0 & 1.0000 & 0.2500 \\ 0 & 0 & -0.8100 & 0.1900 \end{bmatrix} \tag{9}$$

$$\boldsymbol{B_d} = (\Delta t \boldsymbol{B_d}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta t K_1 K_{pp} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.8100 \end{bmatrix}. \tag{10}$$

2.3 The open loop optimization problem

An optimal trajectory for moving the helicopter from initial state $\boldsymbol{x_0} = \begin{bmatrix} \lambda_0 & 0 & 0 & 0 \end{bmatrix}^T$ to final state $\boldsymbol{x_f} = \begin{bmatrix} \lambda_f & 0 & 0 & 0 \end{bmatrix}^T$ is to be calculated by minimizing the objective cost function given by

$$\phi = \sum_{i=0}^{N-1} = (\lambda_{i+1} - \lambda_f)^2 + q p_{ci}^2, \qquad q \geq 0 \tag{11}$$

where $\lambda_0 = \pi$, $\lambda_f = 0$ and the elevation angle is assumed constant. The manipulated variable u has the constraint

$$|\boldsymbol{u_k}| = |p_k| \leq \frac{30\pi}{180}, \qquad k \in \{1, ..., N\} \tag{12}$$

In order to implement and solve this in Matlab, the optimization problem needs to be formulated as a standard QP problem. With lower and upper bounds imposed on the pitch state and controller set-point, and the lack of inequality constraints, the QP-problem can be forumlated as

$$\min_z f(z) = \frac{1}{2} z^T \boldsymbol{G} z \qquad s.t. \qquad \begin{cases} \boldsymbol{A_{eq}} z^* = \boldsymbol{B_{eq}} \\ z_{\boldsymbol{low}} \leq z^* \leq z_{\boldsymbol{high}} \end{cases} \tag{13}$$

where $z = \begin{bmatrix} \boldsymbol{x_1} & \ldots & \boldsymbol{x_N} & , u_0 & \ldots & u_{N-1} \end{bmatrix}^T$ contains all the variables that needs to be optimized. It is a combination of states and inputs over a time horizon N, and has dimension N*mx+N*mu, where mx is the number of states and mu is the number of inputs.

By reformulating the cost function we get

$$\phi = \sum_{i=0}^{N-1} \frac{1}{2} (\boldsymbol{x_{i+1}} - \boldsymbol{x_0})^T \boldsymbol{Q} (\boldsymbol{x_{i+1}} - \boldsymbol{x_0}) + \frac{1}{2} u_i^T R u_i \tag{14}$$

Since the goal of this task is based on λ, it needs to be delivered. In order to do this we have to use the following gains in the cost function (14)

$$\boldsymbol{Q} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{R} = q \tag{15}$$

Based on this, we get the following block diagonal G-matrix in equation (13)

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{Q}_1 & 0 & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \boldsymbol{Q}_{N-1} & \ddots & & \vdots \\ \vdots & & \ddots & R_0 & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & R_{N-1} \end{bmatrix} \tag{16}$$

with dimension (N*mx+N*mu)x(N*mx+N*mu).

To obtain the $\boldsymbol{A_{eq}}$ and $\boldsymbol{B_{eq}}$ matrices, the discrete system must be reformulated to

$$-\boldsymbol{A_d x_k} + \boldsymbol{x_{k+1}} - \boldsymbol{B_d} u_k = 0 \tag{17}$$

By looking at each time step we obtain the following matrices

$$\boldsymbol{A_{eq}} = \left[\begin{array}{ccccc|ccccc} \boldsymbol{I} & 0 & \dots & \dots & 0 & -\boldsymbol{B_d} & 0 & \dots & \dots & 0 \\ -\boldsymbol{A_d} & \boldsymbol{I} & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -\boldsymbol{A_d} & \boldsymbol{I} & 0 & \dots & \dots & 0 & -\boldsymbol{B_d} \end{array}\right] \tag{18}$$

$$\boldsymbol{B_{eq}} = \begin{bmatrix} \boldsymbol{A_d x_0} \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{19}$$

This QP was solved using the Matlab code shown in section 2.7.

2.4 The weights of the optimization problem

We tried using the values 0.1, 1 and 10 as weights q, to check how this would impact the response. Changing q corresponds to changing $\boldsymbol{R}$ in equation (14) which affects the penalizing of $\boldsymbol{u}$. Since $\boldsymbol{u}$ is the reference of pitch, and travel is dependant on pitch (2b), penalizing $\boldsymbol{u}$ will penalize pitch and then affect travel.

The result of the experimentation is plotted in figures 3 - 5. From theory we know that the larger value of q, the more you penalize the input. The larger q got the smoother the response became, and one can observe from the plots, that there then will be a smaller overshoot from the steady sate, especially for q=10 in figure 5. Decreasing q allows for more input and larger overshoot from desired value before it converges against it, as we can see for q=0.1 in figure 3. However, the response became faster and the reference was reached in a shorter amount of time. So, with greater values of q we get a slower, but more accurate response, and with decreasing q values the system gets faster, but also gets more overshoot.

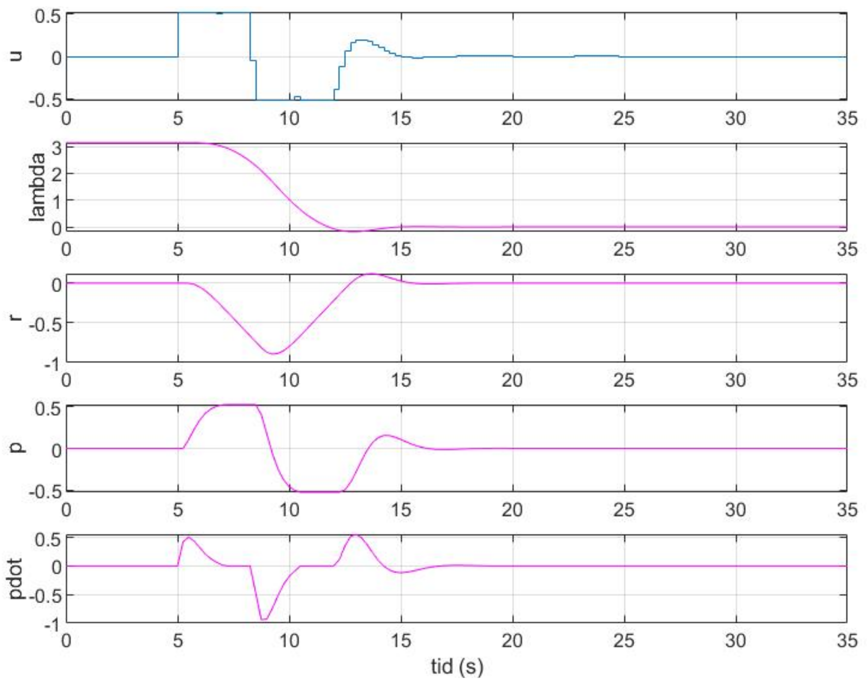


Figure 3: Simulated states with q=0.1

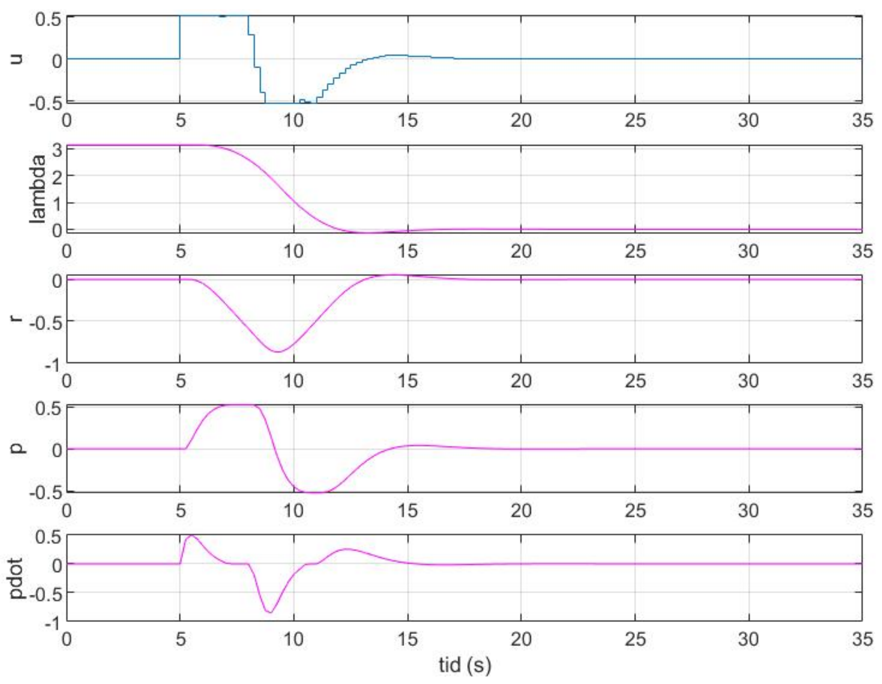


Figure 4: Simulated states with q=1

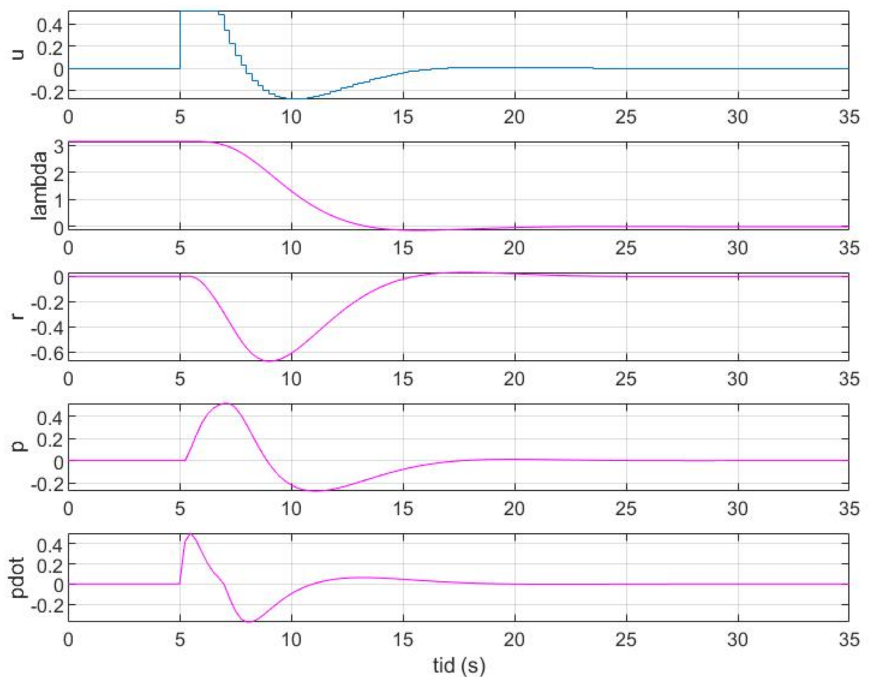


Figure 5: Simulated states with q=10

2.5 The objective function

In this system it is important to understand the objective function (11). Of particular importance and interest is the term $(\lambda_i - \lambda_f)^2$.

If we set $\lambda = \lambda_f$ then the only remaining part of the cost function would be qp_{ci}^2. This implies that travel is not taken into account, and we could reach λ_f without realizing it, and the helicopter would keep going. There is no inner controller of travel, so there is reason to expect large deviations from desired value. In addition, the remaining part of the cost function is dependant on $\boldsymbol{u}$. Then, to minimize it, we only need to set $\boldsymbol{u} = 0$. When there is no input, the controller stops, resulting in the helicopter drifting away from the desired value, due to the helicopter momentum. This means that when $\phi = 0$, there will no longer be given any inputs, and the helicopter will fall out of the desired position, and continue to move. Then, the system will have to try to regulate back into the stationary value.

Since the helicopter lack a state feedback for travel, there will be no compensation for any deviation from travel angle or rate. This causes the helicopter to drift away from desired value, and end up at a non-optimal position. This problem will be fixed using a state feedback controller in the next section, 3.

2.6 Experimental results

The full state is illustrated in figure 6, but to get a closer look at the most interesting values we look at plots 7 and 8.

The most obvious difference from the optimal trajectory and the measured trajectory is that the travel of the helicopter goes to infinity in the physical case, as seen in figure 6 and 7. A reason for this is that our system is open-loop, and the deviations will not be compensated for. This may also be due to the fact that the model is based on simplifications, linearizations and assumptions of the real world. The real world has pressure differences, wind and the elevation angle and rate is non zero, and must be taken into account. Because of this there is model errors that will interfere with the helicopter stopping at desired value, it overcorrects and the helicopter continues to spin, as seen in figure 7.

However, we observe in figure 8 that the helicopter's pitch manages to somewhat follow the optimal input trajectories given in figures 3-5, and there is only a small deviation from desired value. This goes for all the experimented q values. This illustrates a direct effect of adding feedback to an inner controller, and we observe that the deviations are corrected.

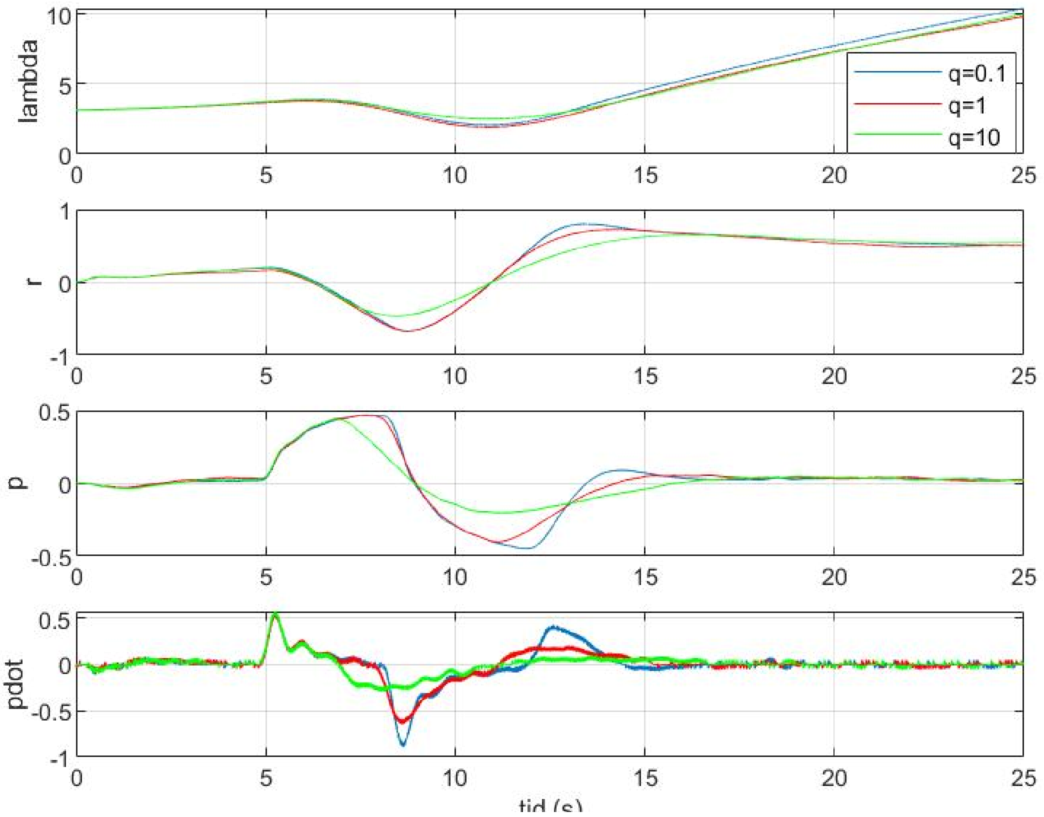
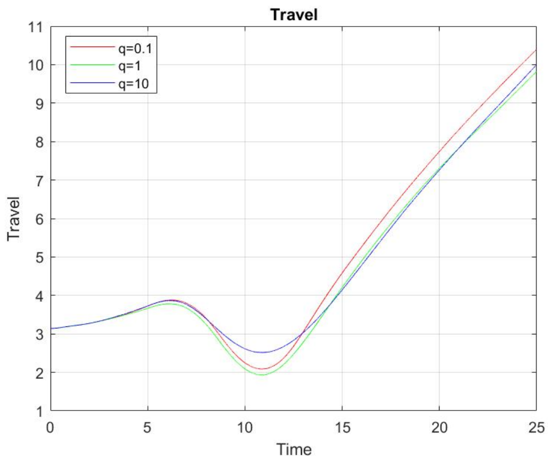


Figure 6: Output from day2 vs simulated values with x^* for q=0.1



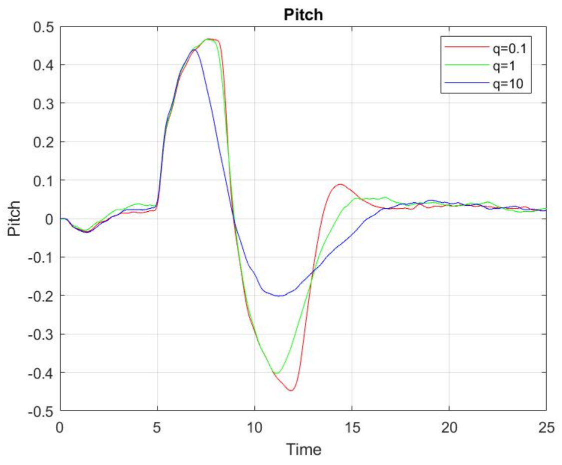


Figure 7: Travel with x^* for q=0.1

Figure 8: Pitch with x^* for q=0.1

2.7 MATLAB and Simulink

The model for day 2 is implemented as the simulink shown in figure 9. The travel angle has an initial value of $\pi = 180$ degrees, which is added in the diagram.

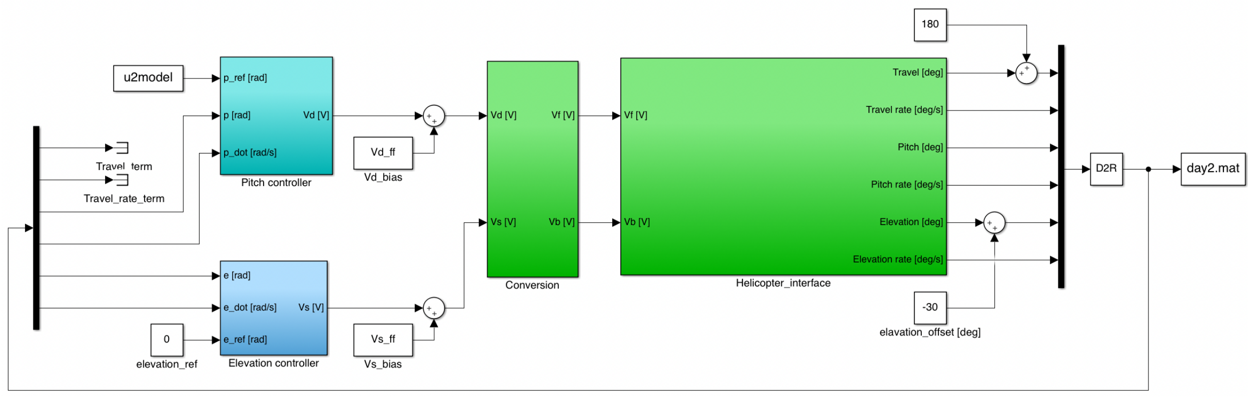


Figure 9: Simulink from day 2

The code used to define and solve the quadratic problem is

```
%% Initialization and model definition
init;

% Discrete time system model. x = [lambda r p pdot]'
deltat = 0.25; % sampling time
A1 = [1 deltat 0                0;
      0 1      -deltat*K2       0;
      0 0      1                deltat;
      0 0      -deltat*K1*Kpp 1-deltat*K1*Kpd];
B1 = [0;
     0;
     0;
     deltat*K1*Kpp];

% Number of states and inputs
mx = size(A1,2); % Number of states (number of columns in A)
mu = size(B1,2); % Number of inputs(number of columns in B)

% Initial values
x10 = pi;                               % Lambda
x20 = 0;                                % r
x30 = 0;                                % p
```

```
x40 = 0;                              % pdot
x0 = [x10 x20 x30 x40]';              % Initial values

% Time horizon and initialization
N  = 100;                      % Time horizon for states
M  = N;                        % Time horizon for inputs
z  = zeros(N*mx+M*mu,1);       % Initialize z for the whole horizon
z0 = z;                        % Initial value for optimization

% Bounds
pk = 30*pi/180;
ul = -pk;                 % Lower bound on control
uu = pk;                  % Upper bound on control

xl = -Inf*ones(mx,1);     % Lower bound on states (no bound)
xu = Inf*ones(mx,1);      % Upper bound on states (no bound)
xl(3) = ul;               % Lower bound on state x3
xu(3) = uu;               % Upper bound on state x3

% Generate constraints on measurements and inputs
[vlb,vub] = genconstraints(N,M,xl,xu,ul,uu);
vlb(N*mx+M*mu) = 0;     % We want the last input to be zero
vub(N*mx+M*mu) = 0;     % We want the last input to be zero

% Generate the matrix Q and the vector c (objecitve function weights in the QP problem)
Q1 = zeros(mx,mx);
Q1(1,1) = 2;          % Weight on state x1
Q1(2,2) = 0;          % Weight on state x2
Q1(3,3) = 0;          % Weight on state x3
Q1(4,4) = 0;          % Weight on state x4
P1 = 1;               % Weight on input
Q = genq(Q1,P1,N                 % Generate Q
c = zeros(N*mx+M*mu,1);          % Generate c, this is the linear constant term in the QP

%% Generate system matrixes for linear model
Aeq = genaeq(A1,B1,N,mx,mu);  % Generate A
beq = zeros(size(Aeq,1),1);   % Generate b
beq(1:mx) =A1*x0;
```

```
%% Solve QP problem with linear model
tic
[z,lambda] = quadprog(Q, c, [], [], Aeq, beq, vlb, vub, x0);
t1=toc;

% Calculate objective value
phi1 = 0.0;
PhiOut = zeros(N*mx+M*mu,1);
for i=1:N*mx+M*mu
  phi1=phi1+Q(i,i)*z(i)*z(i);
  PhiOut(i) = phi1;
end

%% Extract control inputs and states
u  = [z(N*mx+1:N*mx+M*mu);z(N*mx+M*mu)]; % Control input from
   solution

x1 = [x0(1);z(1:mx:N*mx)];      % State x1 from solution
x2 = [x0(2);z(2:mx:N*mx)];      % State x2 from solution
x3 = [x0(3);z(3:mx:N*mx)];      % State x3 from solution
x4 = [x0(4);z(4:mx:N*mx)];      % State x4 from solution

numvariables = 5/deltat;
zeropadding = zeros(numvariables,1);
unitpadding  = ones(numvariables,1);

u   = [zeropadding; u; zeropadding];
x1  = [pi*unitpadding; x1; zeropadding];
x2  = [zeropadding; x2; zeropadding];
x3  = [zeropadding; x3; zeropadding];
x4  = [zeropadding; x4; zeropadding];
```

3 10.3 - Optimal Control of Pitch/Travel with Feedback (LQ)

In this part feedback will be introduced in the optimal controller by using a Linear Quadratic, LQ, controller. The same optimal trajectory $\boldsymbol{x}^*$ and optimal input sequence $\boldsymbol{u}^*$ as calculated in section 2 will be used in this section.

Our system will now have the hierarchy given in figure 10.

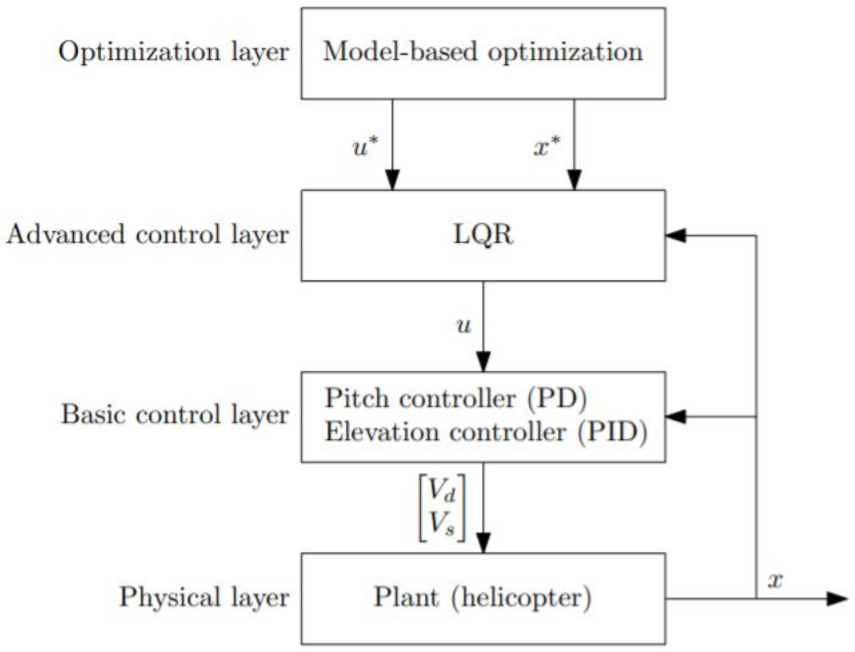


Figure 10: Layers in control hierarchy with LQR. Copied from assignment text.

3.1 LQ controller

The LQ controller minimizes a quadratic criteria for a linear model. The feedback data is obtained by subtracting the calculated optimal states from the measured states. In the case of a deviation, the controller compensates for it by altering control input $\boldsymbol{u_k}$ at time step k, which is given by

$$\boldsymbol{u_k} = \boldsymbol{u_k}^* - \boldsymbol{K}^\top(\boldsymbol{x_k} - \boldsymbol{x_k}^*), \tag{20}$$

where K is the gain matrix that will modify the manipulated variable to compensate for deviations that occurs. This matrix is found by the controller minimizing the following discrete time quadratic function

$$J = \sum_{i=0}^{\infty} \Delta \boldsymbol{x_{i+1}}^{\top} \boldsymbol{Q} \Delta \boldsymbol{x_{i+1}} + \Delta \boldsymbol{u_i}^{\top} \boldsymbol{R} \Delta \boldsymbol{u_i}, \qquad \boldsymbol{Q} \geq 0, \quad \boldsymbol{R} \geq 0 \tag{21}$$

for a linear model

$$\Delta \boldsymbol{x_{i+1}} = \boldsymbol{A} \Delta \boldsymbol{x_i} + \boldsymbol{B} \Delta \boldsymbol{u_i} \tag{22}$$

where $\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^*$ and $\Delta \boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}^*$.

$\boldsymbol{Q}$ and $\boldsymbol{R}$ are weight matrices for the states and inputs, respectively. We will further refer to them as ***LQ_Q*** and ***LQ_R*** to separate them from the **Q** and **R** matrices from section 2. They symbolize how much you want to penalize the variables, thus deciding how much of a deviation from desired state you will allow. The greater the values of ***LQ_Q*** and ***LQ_R***, the smaller the deviation will become. In addition, the relation between ***LQ_Q*** and ***LQ_R*** also contributes to the amount of penalizing. We will experiment on how different values of ***LQ_Q*** and ***LQ_R*** affects our system. These matrices are also important factors in calculating the gain matrix, **K**.

K was calculated using the Matlab command **dlqr** as shown in subsection 3.4. This command solves the algebraic Riccati equation (23).

$$\boldsymbol{S} = \boldsymbol{Q} + \boldsymbol{A}^T \boldsymbol{S} (\boldsymbol{I} + \boldsymbol{B} \boldsymbol{R}^{-1} \boldsymbol{B}^T \boldsymbol{S})^{-1} \boldsymbol{A} \tag{23}$$

This solution gives us a matrix **S**, which Matlab further uses to solve for **K** using the following equation [2]

$$\boldsymbol{K} = (\boldsymbol{B}^T \boldsymbol{S} \boldsymbol{B} + \boldsymbol{R})^{-1} (\boldsymbol{B}^T \boldsymbol{S} \boldsymbol{A} + N^T) \tag{24}$$

with N=0.

After experimenting with many different values for ***LQ_Q*** and ***LQ_R***, the final tuning was

$$\boldsymbol{LQ\_Q} = \begin{bmatrix} 7 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{LQ\_R} = 0.1 \tag{25}$$

When tuning the helicopter, travel accuracy was the main focus. For that reason, we chose these values as they penalize travel and travel rate deviations quite hard. This allows travel to get close to the desired value, without sacrifising pitch too much, as shown in figures 14 and 15. Also, a small penalizing of the input, $\boldsymbol{LQ\_R} = 0.1$, allows faster control compared to larger values, as seen in figure 12. So, these chosen matrices penalizes deviations of travel and travel rate, while giving more freedom to pitch and pitch rate. This is desired, as the task is mostly based on travel. Some penalizing of pitch is necessary as travel is dependent on it, as observed from equation (2b).

3.2 Model Predictive Control

In this exercise MPC controller could have been used as an alternative strategy to get an optimal trajectory with feedback. This is a form of control in which a finite horizon open-loop problem is solved at each sampling time to obtain the current control action [3]. By using this the layer hierarchy would be as illustrated in figure 11, and the system would be entirely closed loop since all of the different layers would be given feedback. It does not need a separate optimization layer, because the MPC controller serves as both an optimization layer and an advanced control layer.

This MPC controller is realized by solving the QP problem N steps ahead for each time step, based on the new feedback value of $\boldsymbol{x_i}$. Only the first element of the resulting optimal sequence is used, and for the next iteration you have to calculate it again. The LQ controller on the other hand, only solves the QP problem once, and this result is used throughout the time horizon. Therefore, the MPC is more computationally expensive than the LQ. Furthermore, MPC is dependant on a more accurate physical model than the LQ controller is. In our case, the model is based on simplifications, so MPC might not work as well as LQ, and may have faults.

However, compared to the LQ controller, the MPC controller provide more precise control because the optimal input is constantly modified, and possible deviations will then be handled faster. Another advantage of the MPC is that it is more robust and is more resistant to disturbances. This is because it can calculate a new trajectory if the new one is lost, opposed to the LQ controller that might oscillate around the trajectory as it tries to follow the previously optimal path.

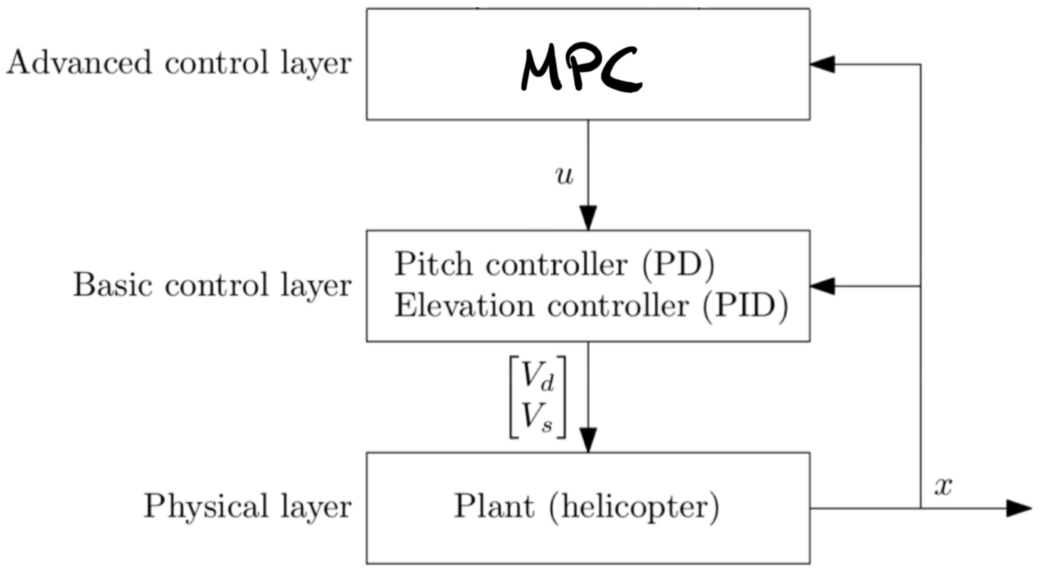


Figure 11: Layers of the hierarchy with MPC

3.3 Experimental results

To find the optimal values of **LQ_Q** and **LQ_R** we had to experiment with different values. We also experimented with some extreme values, to see how this would affect our helicopter. The experimented **LQ_Q**'s and **LQ_R**'s are given in table 3, and the system responses are plotted in figures 12 and 13.

Diagonal of LQ_Q	Diagonal of LQ_R	Corresponding K matrix
$\boldsymbol{LQ\_Q_1} = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$	$\boldsymbol{LQ\_R_1} = \begin{bmatrix}1\end{bmatrix}$	$\boldsymbol{K_1} = \begin{bmatrix}-0.6426 & -2.4359 & 1.2587 & 0.5186\end{bmatrix}$
$\boldsymbol{LQ\_Q_2} = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$	$\boldsymbol{LQ\_R_2} = \begin{bmatrix}0.1\end{bmatrix}$	$\boldsymbol{K_2} = \begin{bmatrix}-0.8966 & -3.1869 & 1.7264 & 0.7724\end{bmatrix}$
$\boldsymbol{LQ\_Q_3} = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$	$\boldsymbol{LQ\_R_3} = \begin{bmatrix}10\end{bmatrix}$	$\boldsymbol{K_3} = \begin{bmatrix}-0.2798 & -1.3519 & 0.6735 & 0.2282\end{bmatrix}$
$\boldsymbol{LQ\_Q_4} = \begin{bmatrix}7 & 3 & 0.5 & 1\end{bmatrix}$	$\boldsymbol{LQ\_R_4} = \begin{bmatrix}0.1\end{bmatrix}$	$\boldsymbol{K_4} = \begin{bmatrix}-2.2402 & -5.8583 & 2.6491 & 0.9247\end{bmatrix}$
$\boldsymbol{LQ\_Q_5} = \begin{bmatrix}50 & 1 & 1 & 1\end{bmatrix}$	$\boldsymbol{LQ\_R_5} = \begin{bmatrix}0.1\end{bmatrix}$	$\boldsymbol{K_5} = \begin{bmatrix}-5.3769 & -10.6351 & 4.1699 & 1.1704\end{bmatrix}$
$\boldsymbol{LQ\_Q_6} = \begin{bmatrix}1 & 1 & 20 & 1\end{bmatrix}$	$\boldsymbol{LQ\_R_6} = \begin{bmatrix}0.1\end{bmatrix}$	$\boldsymbol{K_6} = \begin{bmatrix}-0.6508 & -3.2479 & 3.1728 & 1.1481\end{bmatrix}$

Table 3: ***LQ_Q***, ***R*** and ***K*** matrix

Since the relation between ***LQ_Q*** and ***LQ_R*** has an important part in their effect on the system, we first experimented with what value of ***LQ_R*** should be used in further experimentation and tuning. This was done by keeping ***LQ_Q*** constant as the identity matrix, and changing the values of ***LQ_R***. The result is shown in figure 12.

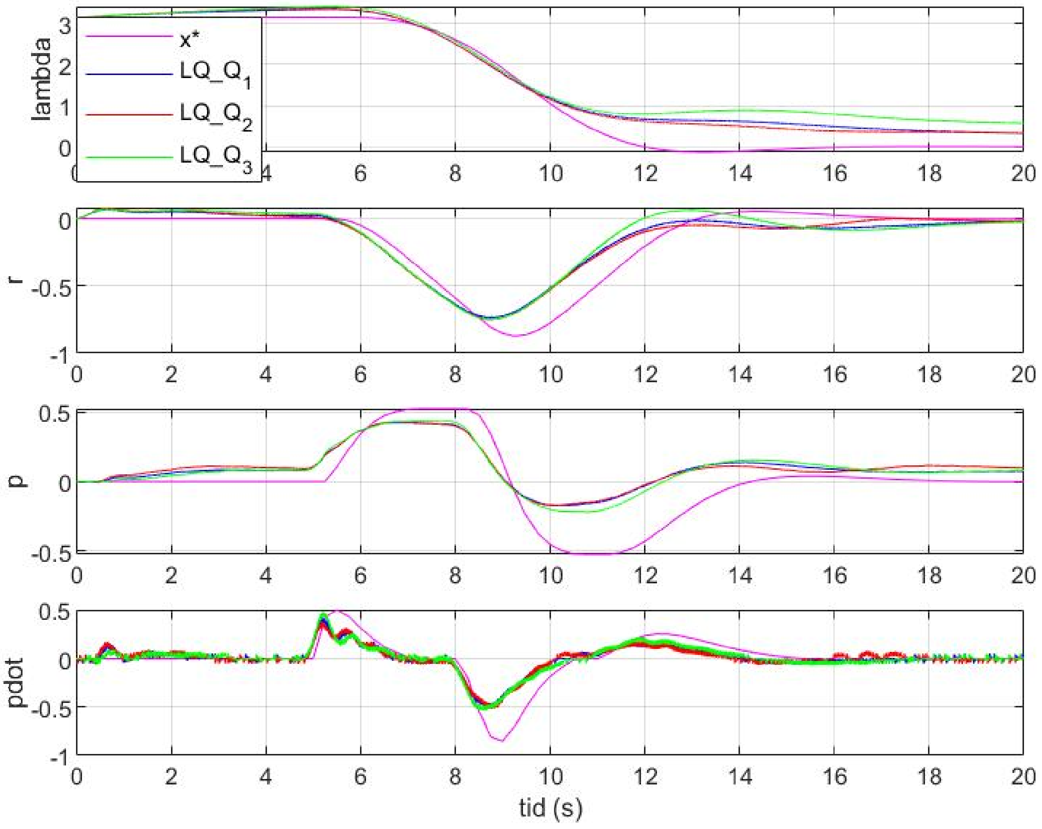


Figure 12: Different trajectories for $\boldsymbol{LQ\_Q}$ = diag([1 1 1 1]) and $\boldsymbol{LQ\_R}_1$, $\boldsymbol{LQ\_R}_2$, $\boldsymbol{LQ\_R}_3$

After observing the plots in figure 12, we decided to go with $\boldsymbol{LQ\_R}$=0.1 and tune our $\boldsymbol{LQ\_Q}$-matrix based on this. In this task we decided to prioritize travel, as this is the only variable that has a different initial and stationary value. The reason why we decided not to use $\boldsymbol{LQ\_R}$=10, the green plot, is because this had large deviations from desired value in several of our states. This is mainly because it penalizes the input too much, and the system is not allowed enough input to quickly regain the optimal value, resulting in a slow system. It may also be related to model errors from simplifications. Both $\boldsymbol{LQ\_R}$=1 and $\boldsymbol{LQ\_R}$=0.1 could have been used as they both had good approximations to desired values, but since $\boldsymbol{LQ\_R} = 0.1$ got the system closer to travel's optimal value, $\lambda_f = 0$, and also made the system faster allowing more input, we decided to further tune with this value.

We also found it interesting to experiment with a $\boldsymbol{LQ\_Q}$ matrix that had a large penalizing of travel, $\boldsymbol{LQ\_Q}_5$, and one that had a large penalizing of pitch, $\boldsymbol{LQ\_Q}_6$. These are illustrated with our tuned system in figure 13.

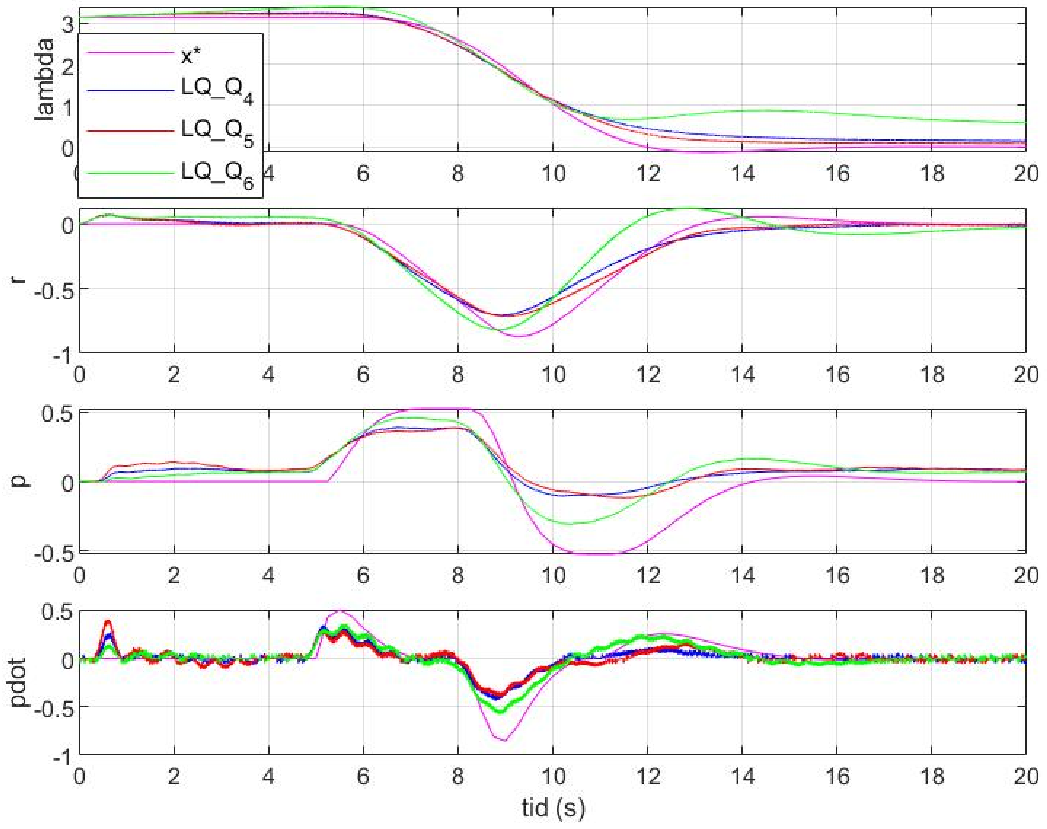


Figure 13: Experimentation for LQ Controller

The initial jump in pitch, figure 6, is caused by the rise of the helicopter as elevation tries to reach zero. Due to the model not taking elevation into consideration and expects it to be zero, it is nearly impossible to tune away this initial jump.

To get a closer look at travel and pitch for the experimented values we plotted these alone in figures 14 - 19.

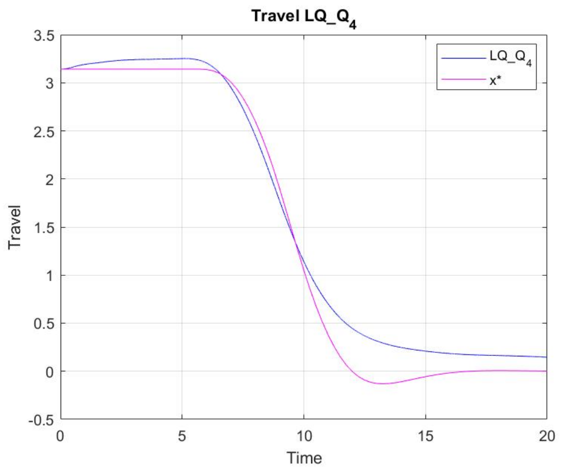


Figure 14: $\boldsymbol{LQ\_Q_4}$: Measured vs Optimal Travel

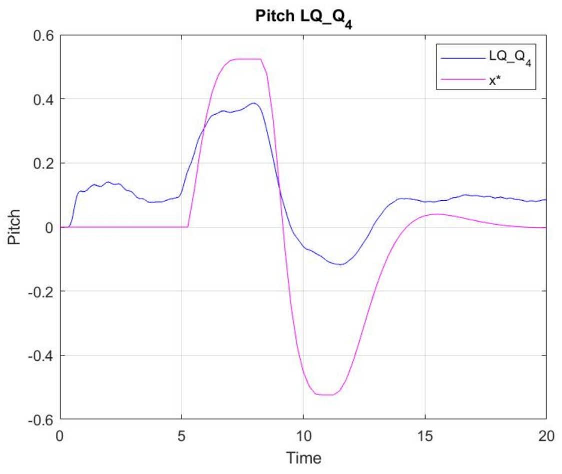


Figure 15: $\boldsymbol{LQ\_Q_4}$: Measured vs Optimal Pitch

As observed in figure 14, the tuned system gets quite close to the final state of travel, it only has a deviation of 0.2. Ideally it should reach zero, but we could not penalize the system enough without getting lots of deviation from pitch. We could have penalized pitch rate less, as this would allow for more rapid changes in pitch and better following, but this would affect travel in an unwanted way. Further discussion for these matrices was given in section 3.1.

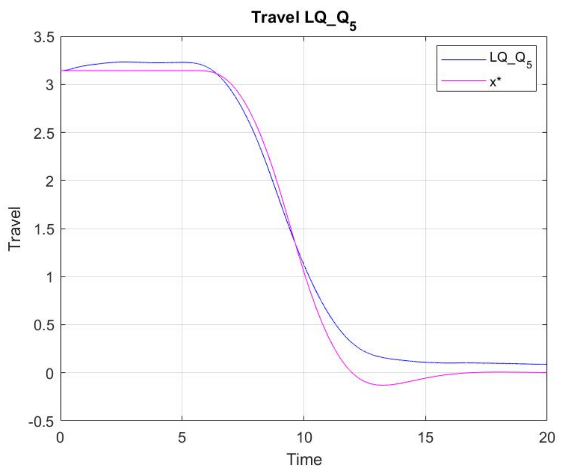


Figure 16: $\boldsymbol{LQ\_Q_5}$: Measured vs Optimal Travel

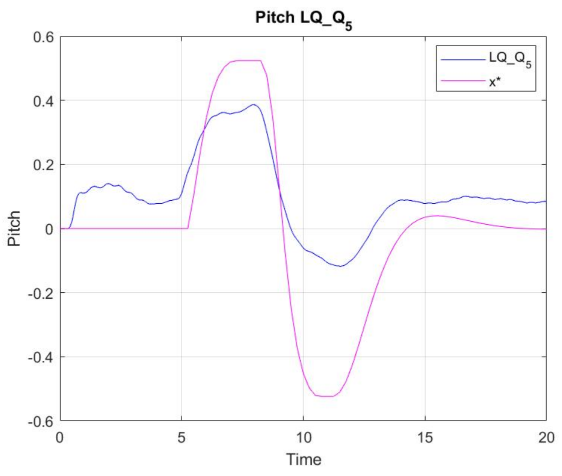


Figure 17: $\boldsymbol{LQ\_Q_5}$: Measured vs Optimal Pitch

As expected based on theory, when using a large value for travel, matrix $\boldsymbol{LQ\_Q_5}$, travel got close to desired value without much deviation, figure 16. However, pitch got large deviations from the optimal value and did not follow the trajectory well, observed in figure 17. Even though this system solved the task of getting travel to zero, the other states need to be taken into account to keep the helicopter steady. So, even though this system has similar response to $\boldsymbol{LQ\_Q}_4$, it is not as good of a solution.

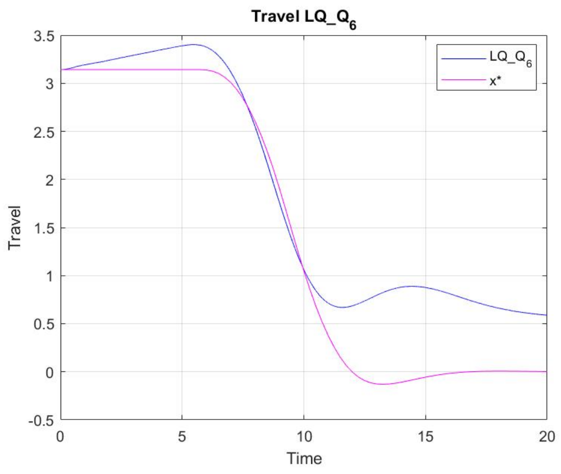


Figure 18: $\boldsymbol{LQ\_Q_6}$: Measured vs Optimal Travel

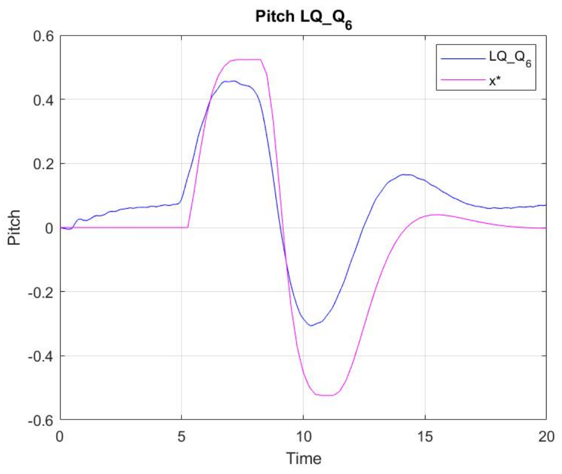


Figure 19: $\boldsymbol{LQ\_Q_6}$: Measured vs Optimal Pitch

To no surprise, using a large value for pitch penalized pitch a lot, and only small deviations occured from optimal trajectory, figure 19. There was only a small direct penalizing of travel, but it was still quite penalized. The reason for this is that travel is dependant on pitch, which means that penalizing pitch affects travel aswell. However, one can observe from figure 13 that travel rate is lagging behind optimal value and gets large overshoots, resulting in travel still having a large deviation from desired value. This is a result of travel rate not being penalized enough, and it has too much freedom which allows rapid changes. Perhaps, letting the simulation run for a longer time period would let the value get closer to λ^*, but this would not change the fact that the system is too slow. Overall these values gave a bad result because the optimal pitch setpoint was computed for an unrealistic model.

By comparing these results with the ones obtained in section 2.6, we can see that adding feedback has significant effect on the helicopter. It effectively removes the drift off that occur in figure 7, and helps the helicopter reach its desired value, as shown in figure 14. This means that deviations from desired trajectory is successfully compensated for and the tracking is highly improved, when implementing a feedback controller.

3.4 MATLAB and Simulink

The full simulink diagram used to solve the problem of section 2, is shown in figure 20.

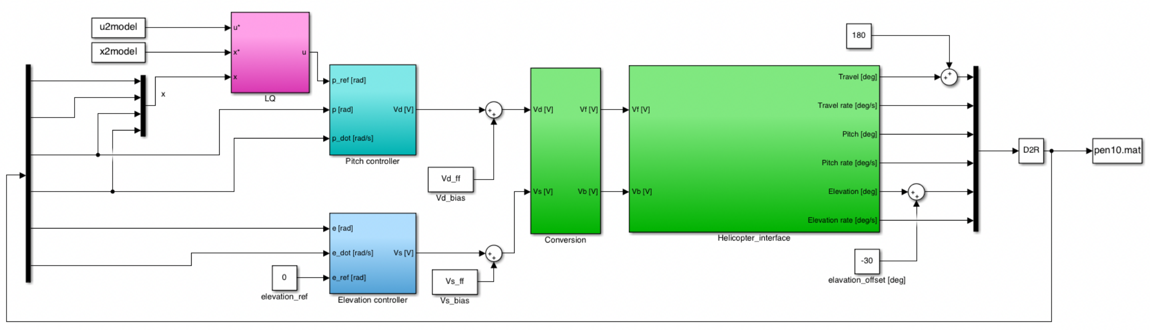


Figure 20: Simulink for day 3

The implementation of the LQ controller is shown in the simulink diagram in figure 36.

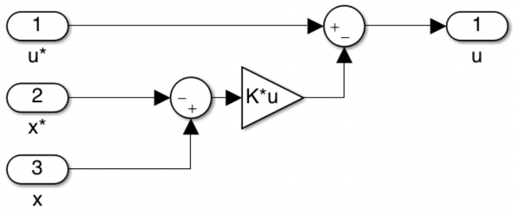


Figure 21: Simulink implementation of the LQcontroller

The Matlab code used for implementation and solving the problem:

```
%Labday 3
%load day 2
run(”templateproblem2.m”)

%Define LQ matrices
Q = diag([7 3 0.5 1]);
R = 0.1;

%Get gain matrix
```

```
[K, S, e] = dlqr(A1, B1, Q, R);

%Sending data
xopt = [x1 x2 x3 x4];
x2model = [t', xopt];
```

4 10.4 - Optimal Control of Pitch/Travel and Elevation with Feedback

In this part we will calculate the optimal trajectory in two dimensions. The goal is to move the helicopter from an initial state $\boldsymbol{x_0}$ to a final state $\boldsymbol{x_f}$ past a restriction causing the elevation angle to change during the flight, which means that we have to include the elevation angle and rate in our model.

4.1 The continuous model

By including elevation angle and rate in our model we get the state

$$\boldsymbol{x} = \begin{bmatrix} \lambda & r & p & \dot{p} & e & \dot{e} \end{bmatrix}^\top \tag{26}$$

The setpoint e_c is included in the system, which gives the input u

$$\boldsymbol{u} = \begin{bmatrix} p_c \\ e_c \end{bmatrix} \tag{27}$$

From the description of the model we have six equations for the time derivatives states, given in equation (2a)-(2f). Which gives us the following continuous system with matrices

$$\boldsymbol{A_c} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3K_{ep} & -K_3K_{ed} \end{bmatrix} \tag{28}$$

$$\boldsymbol{B_c} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ K_1K_{pp} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & K_3K_{ep} \end{bmatrix} \tag{29}$$

Which can be summed up in the continuous model

$$\dot{\boldsymbol{x}} = \boldsymbol{A_c x} + \boldsymbol{B_c u} \tag{30}$$

4.2 The discretized model

By using the forward Euler method introduced in 2.2 we can get the discretized model. With $\boldsymbol{A_d}$ and $\boldsymbol{B_d}$ calculated as in equation (7). The results is given in equations (31)-(33).

$$\boldsymbol{x_{k+1}} = \boldsymbol{A_d}\boldsymbol{x_k} + \boldsymbol{B_d}\boldsymbol{u_k} \tag{31}$$

$$\boldsymbol{A_d} = \mathbb{I} + \Delta t \boldsymbol{A_c} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & -\Delta t K_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 1-\Delta t K_1 K_{pd} & -\Delta t K_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & & 1 & \Delta t \\ 0 & 0 & 0 & 0 & -K_3 K_{ep} & 1-\Delta t K_3 K_{ep} \end{bmatrix} \tag{32a}$$

$$= \begin{bmatrix} 1 & 0.25 & 0 & 0 & 0 & 0 \\ 0 & 1 & -0.1415 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.25 & 0 & 0 \\ 0 & 0 & -0.81 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.25 \\ 0 & 0 & 0 & 0 & -0.0625 & 0.75 \end{bmatrix} \tag{32b}$$

$$\boldsymbol{B_d} = \Delta t \boldsymbol{B_c} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \Delta t K_3 K_{ep} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.8100 & 0 \\ 0 & 0 \\ 0 & 0.0625 \end{bmatrix} \tag{33}$$

Where Δt is 0.25 seconds.

4.3 Experimental results

Since the state and input has increased by respectively two and one variable, we have to redefine our objective function ϕ accordingly, which gives us equation (34)

$$\phi = \sum_{i=0}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2, \qquad q_1, q_2 \geq 0 \tag{34}$$

Which can be rewritten as

$$\phi = \sum_{i=0}^{N-1} \frac{1}{2}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_0)^T \boldsymbol{Q} (\boldsymbol{x}_{i+1} - \boldsymbol{x}_0) + \frac{1}{2}\boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i \tag{35}$$

where **Q** and **R** is given by

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix} \tag{36}$$

The variables of the input $\boldsymbol{u}$ is constrained by p_k, given in equation (12), and e_k, for each timestep k. The manipulated variable e_c is constrained by a nonlinear constraint given in equation (37)

$$c(x_k) = \alpha exp(-\beta(\lambda_k - \lambda_t)^2) - e_k \geq 0 \qquad \forall k \in \{1, ..., N\} \tag{37}$$

With $\alpha = 0.2, \beta = 20$ and $\lambda_t = 2\pi/3$.

Since the constraint (37) is nonlinear we have to use fmincon as an Sequential Quadratic Programming (SQP) type algorithm. Our implementation of the constraint for each timestep is given in section 4.5

We also have to update and tune our LQ controller to include our two new states and elevation input. Which results in a total of three matrices, $\boldsymbol{R}$, $\boldsymbol{LQ\_Q}$, $\boldsymbol{LQ\_R}$, that we have to decide. We started with finding out how large q_1 and q_2 have to be, i.e how much we need to penalize the input u. We keep $\boldsymbol{LQ\_Q}$ and $\boldsymbol{LQ\_R}$ constant at respectively

$$\boldsymbol{LQ\_Q} = \begin{bmatrix} 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{38a}$$

$$\boldsymbol{LQ\_R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{38b}$$

Since $\boldsymbol{LQ\_Q}$ and $\boldsymbol{LQ\_R}$ are constant the gain matrix **K** is also constant at

$$\mathbf{K} = \begin{bmatrix} -2.2482 & -5.8839 & 2.6688 & 0.8254 & 0 & 0 \\ 00 & 0 & 0 & 0 & 2.0230 & 2.5787 \end{bmatrix} \tag{39}$$

which is an extension of the LQ controller from section 3.

We tried 3 different values for $\boldsymbol{R}$ matrix. One where $\boldsymbol{R}$ is somewhat equal to $\boldsymbol{Q}$, one where $\boldsymbol{R}$ is approximately ten times larger than $\boldsymbol{Q}$ and one where $\boldsymbol{R}$ is approximately ten times smaller than $\boldsymbol{Q}$. This is summed up in table 4. The reason for comparing the ratio between $\boldsymbol{Q}$ and $\boldsymbol{R}$ is because $\boldsymbol{Q}$ and $\boldsymbol{R}$ are weight matrices and the ratio between them affects how much we punish the state $\boldsymbol{x}$ and $\boldsymbol{u}$ respectively.

Diagonal of $\mathbf{R} = \begin{bmatrix} q_1 & q_2 \end{bmatrix}$
$R_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$
$R_2 = \begin{bmatrix} 10 & 10 \end{bmatrix}$
$R_3 = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$

Table 4: Caption

Looking back, we probably should have tried having q_1 and q_2 different from each other. However this only affects how much we penalize the pitch versus the elevation in the PID-regulator.

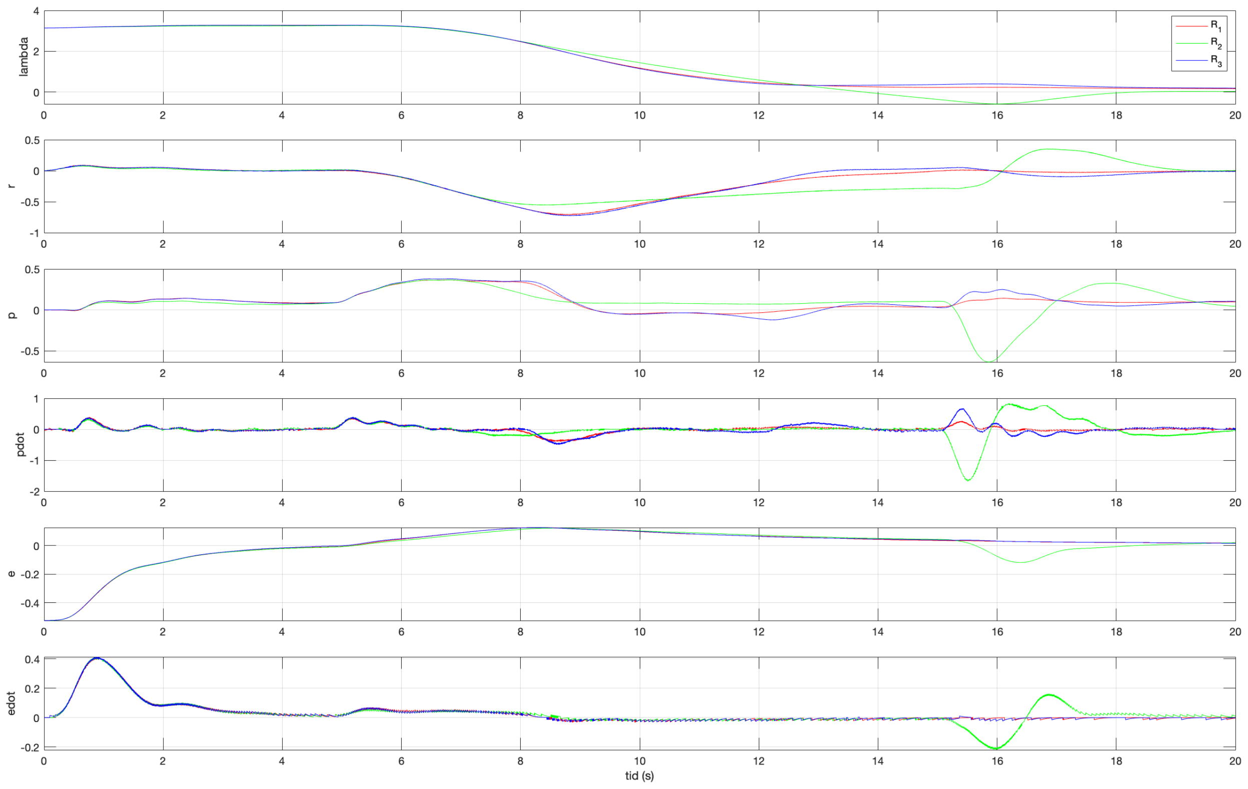


Figure 22: Plots of the state and input trajectory for $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$

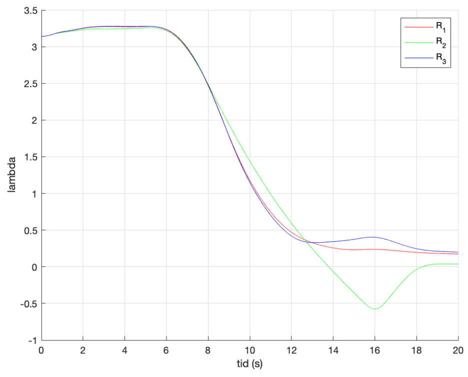


Figure 23: λ with $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$

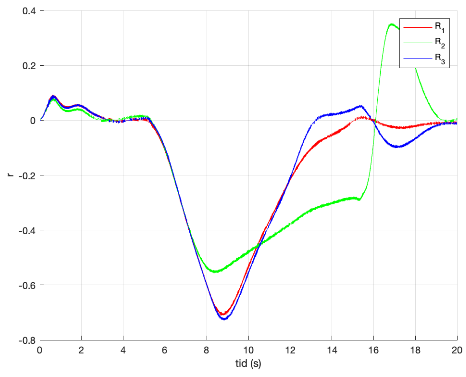


Figure 24: r with $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$

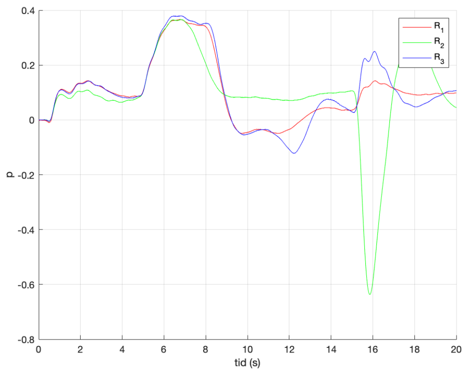


Figure 25: p with $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$

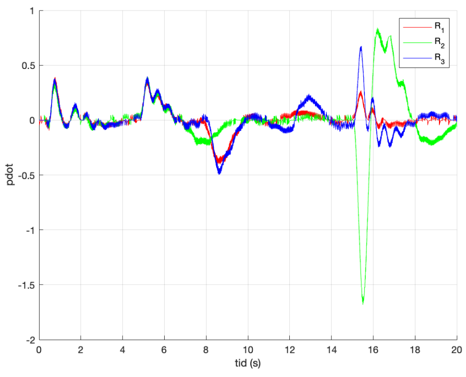


Figure 26: $\dot{p}$ with $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$

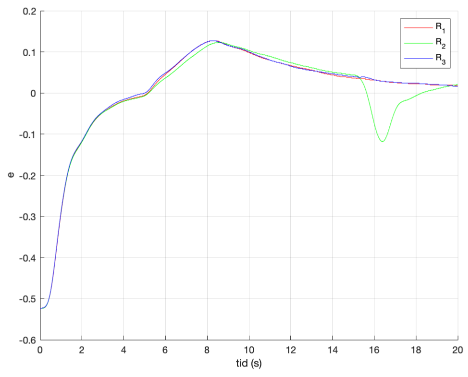


Figure 27: e with $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$

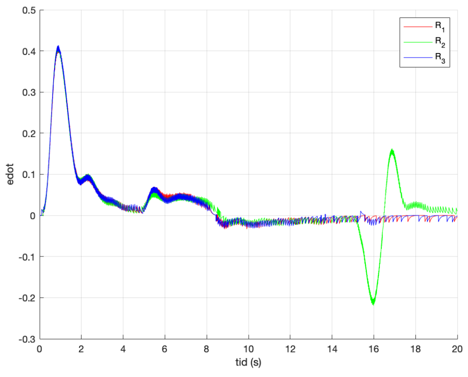


Figure 28: $\dot{e}$ with $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$

Due to our 5 second padding, the optimal trajectory and control don't have an effect before 5 seconds and after 15 seconds. From the plot in figure 22 we see that the measurements of the states differs in this time span. Anyways, since our goal is to get the helicopter to arrive at its final state, $\boldsymbol{x}_f$, our focus is to find the $\boldsymbol{R}$ matrix that gives the trajectory that ends closest to $\boldsymbol{x}_f$.

Each measurement follows somewhat the same route. From figure (23)-(28) we can see that $\boldsymbol{R}_2$ gets the closest to the final state $\boldsymbol{x}_f$, but shoots off from time to time. The trajectory of $\boldsymbol{R}_2$ is not as smooth and stable as $\boldsymbol{R}_1$ and $\boldsymbol{R}_3$. For the last 5 seconds, when the optimal trajectory $\boldsymbol{x}^*$ is 0 due to the padding, we can see that $\boldsymbol{R}_2$ struggles to keep the states and oscillates around 0 for each state in $\boldsymbol{x}$. Therefore, even thought $\boldsymbol{R}_2$ gets the helicopter closest to the final state, it cannot keep the helicopter stable and could be possibly dangerous to fly if this was a real helicopter with people in it. The reason for this is probably that we penalize the input $\boldsymbol{u}$ too much and the motor wont have enough power to stabilize and overreacts.

One thing that applies to $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$ is that they have basically the same trajectory for both the elevation and elevation rate given in figure (27) and (28). This means that varying q_2 doesn't have as much affect as varying q_1. $\boldsymbol{R}_2$ has some differences in the end, but this might be due to the states not being completely decoupled and that q_1 affects the elevation angle and rate at high values (i.e when the helicopter is unstable)

$\boldsymbol{R}_1$ and $\boldsymbol{R}_3$ have almost the same trajectory, and from figure 22 it is nearly impossible to see the difference between them. Figure (23)-(28) gives a clearer view of the plots, but loses the clarity of how the states depend on each other at each timesteps. Anyways, in figure 25 we can see that when the optimal trajectory $\boldsymbol{x}^*$ becomes zero at 15 seconds the trajectory of $\boldsymbol{R}_2$ undershoots while the trajectory of $\boldsymbol{R}_3$ overshoots. $\boldsymbol{R}_1$ is more stable and keeps the helicopter closest to x_f without shooting off. Therefore we chose to use $\boldsymbol{R}_1$ further in our experimentation.

Now we will try to tune the LQ controller. We keep the $\boldsymbol{LQ\_R}$ and $\boldsymbol{R}$ matrices constant given in equation (40). i.e we only tune w.r.t the state and not the input

$$\boldsymbol{LQ\_R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{40a}$$

$$\boldsymbol{R} = \boldsymbol{R}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{40b}$$

As mention before the $\boldsymbol{LQ\_Q}$ and $\boldsymbol{LQ\_R}$ matrix are used to calculate the gain matrix $\boldsymbol{K}$, as given in equation (24). From equation (20) we see that the gain matrix only decide how much of the difference between the trajectory $\boldsymbol{x}_k$ and $\boldsymbol{x}_k^*$ we are going to let influence the input at timestep k. Due to matrix multiplication, changing one value on the diagonal of $\boldsymbol{LQ\_Q}$ might change several different values in the $\boldsymbol{K}$-matrix, and therefore affecting several of the states in $\boldsymbol{x}$. Since $\boldsymbol{x}$ and $\boldsymbol{u}$ both increased in size, $\boldsymbol{K}$ also increased to dimension (dim($\boldsymbol{u}$)Xdim($\boldsymbol{x}$)) = 2x6 matrix. From table 5 we can see that the elements corresponding to the four first elements of $\boldsymbol{x}$ and second element is zero, and the two last elements of $\boldsymbol{x}$ and first elements of $\boldsymbol{u}$ is zero.

Which means that in theory λ, r, p and $\dot{p}$ only affects p_c and e and e_{dot} only affects e_c in the PID-controller. From table 5 we see that $\boldsymbol{K}(1{,}1{:}4)$ is the same for both $\boldsymbol{LQ\_Q_1}$ and $\boldsymbol{LQ\_Q_2}$, and different from $\boldsymbol{LQ\_Q_3}$. So in theory we should expect $\boldsymbol{LQ\_Q_1}$ and $\boldsymbol{LQ\_Q_2}$ to keep the same distance from $\boldsymbol{x}^*$, but since equation (20) takes into account the current state of the helicopter $\boldsymbol{x}_k$ and that the elevation has a physical affect they might be different. We tried more $\boldsymbol{LQ\_Q}$ than listed in table 5, but to condense the report we only listed the matrix that gave interesting pots for the discussion.

Diagonal of $\boldsymbol{LQ\_Q}$	Corresponding $\boldsymbol{K}$ matrix
$\boldsymbol{LQ\_Q_1} = \begin{bmatrix} 7 & 3 & 0.5 & 1 & 1 & 10 \end{bmatrix}$	$\boldsymbol{K}_1 = \begin{bmatrix} -2.2482 & -5.8839 & 2.6688 & 0.8254 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.5415 & 5.4424 \end{bmatrix}$
$\boldsymbol{LQ\_Q_2} = \begin{bmatrix} 7 & 3 & 0.5 & 1 & 1 & 0.1 \end{bmatrix}$	$\boldsymbol{K}_2 = \begin{bmatrix} -2.2482 & -5.8839 & 2.6688 & 0.8254 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.1043 & 2.0443 \end{bmatrix}$
$\boldsymbol{LQ\_Q_3} = \begin{bmatrix} 12 & 3 & 1 & 2 & 10 & 1 \end{bmatrix}$	$\boldsymbol{K}_3 = \begin{bmatrix} -2.1663 & -5.6170 & 2.5709 & 0.8220 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7.2159 & 5.8669 \end{bmatrix}$

Table 5: Caption

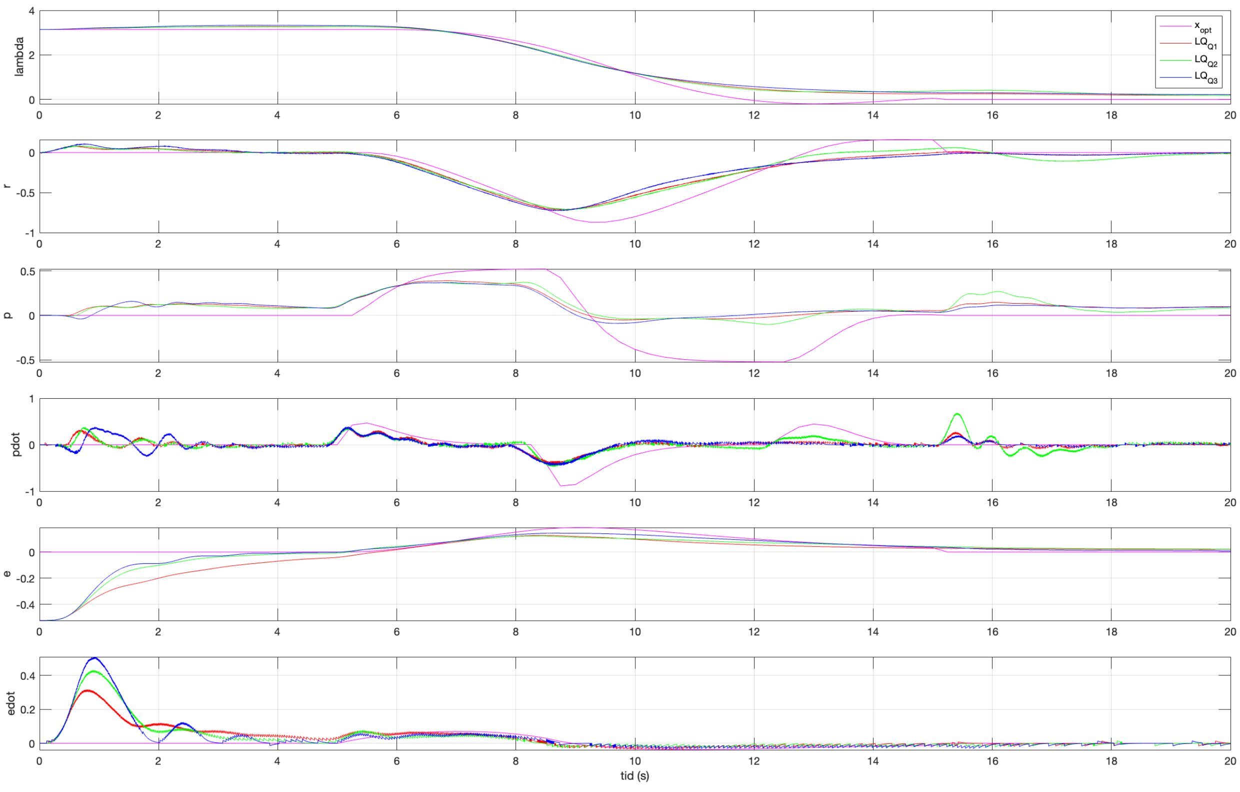


Figure 29: Plots of the state and input trajectory for $\boldsymbol{LQ\_Q_1}$, $\boldsymbol{LQ\_Q_2}$ and $\boldsymbol{LQ\_Q_3}$

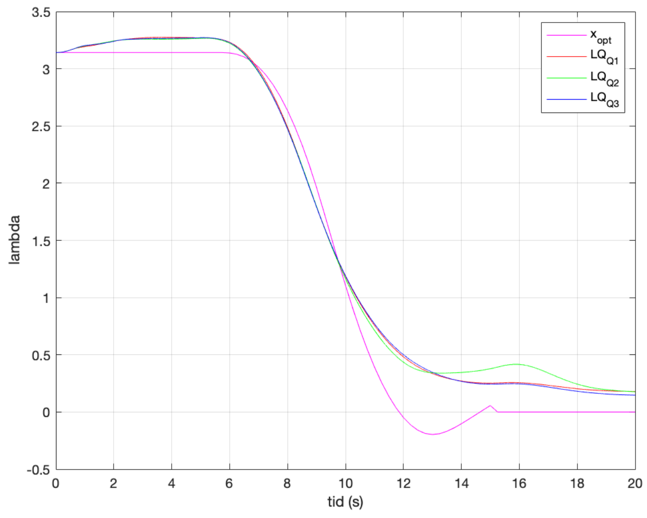


Figure 30: λ with $\boldsymbol{LQ\_Q_1}$, $\boldsymbol{LQ\_Q_2}$ and $\boldsymbol{LQ\_Q_3}$

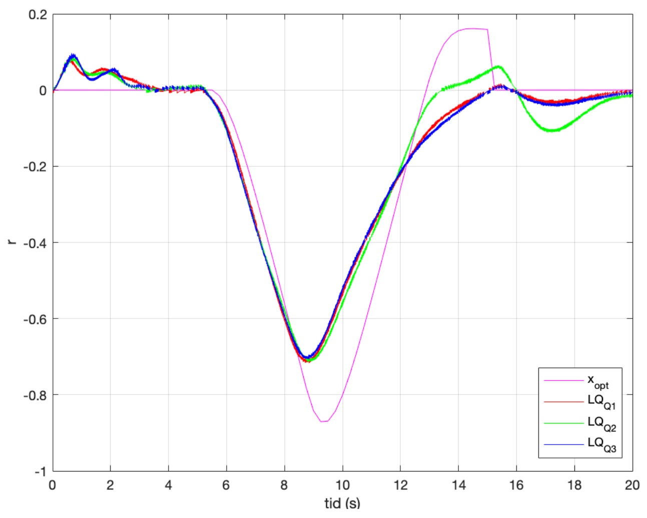


Figure 31: r with $\boldsymbol{LQ\_Q_1}$, $\boldsymbol{LQ\_Q_2}$ and $\boldsymbol{LQ\_Q_3}$

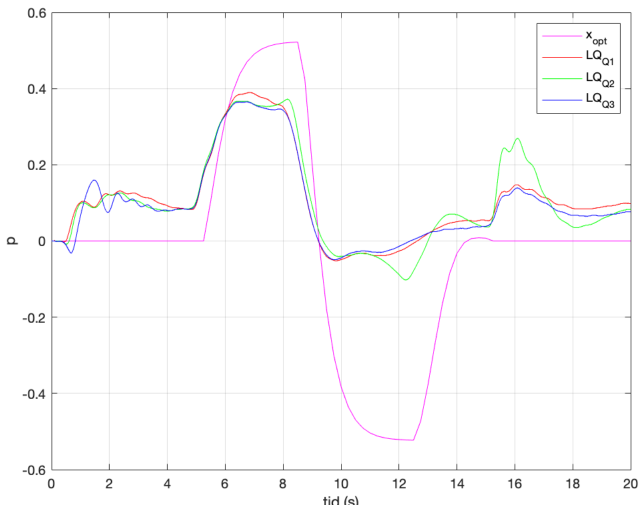


Figure 32: p with $\boldsymbol{LQ\_Q_1}$, $\boldsymbol{LQ\_Q_2}$ and $\boldsymbol{LQ\_Q_3}$

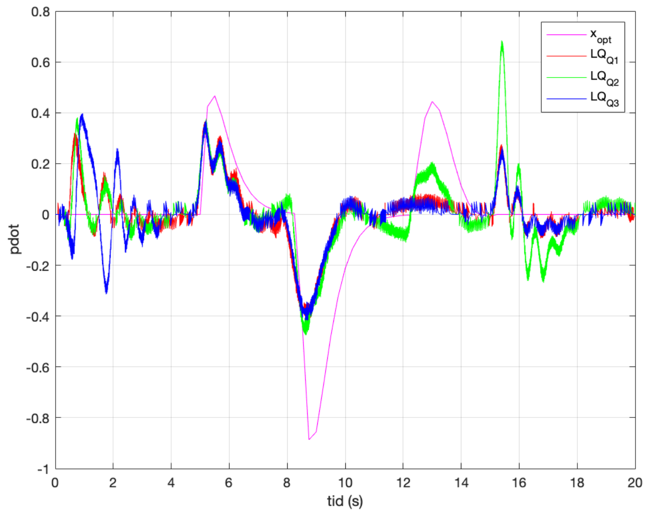


Figure 33: $\dot{p}$ with $\boldsymbol{LQ\_Q_1}$, $\boldsymbol{LQ\_Q_2}$ and $\boldsymbol{LQ\_Q_3}$

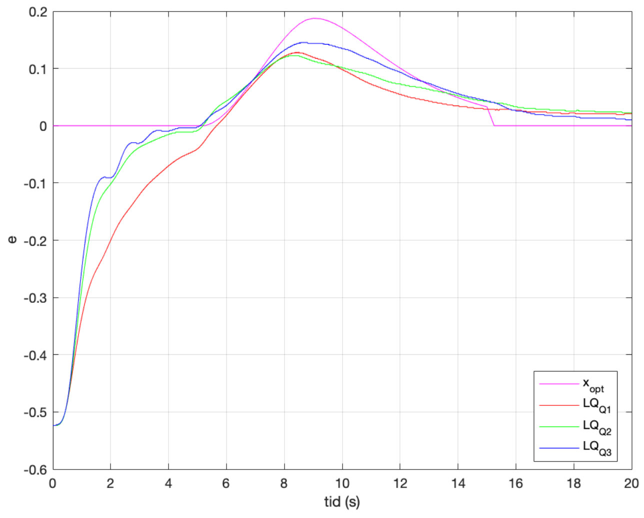


Figure 34: e with $\boldsymbol{LQ\_Q}_1$, $\boldsymbol{LQ\_Q}_2$ and $\boldsymbol{LQ\_Q}_3$

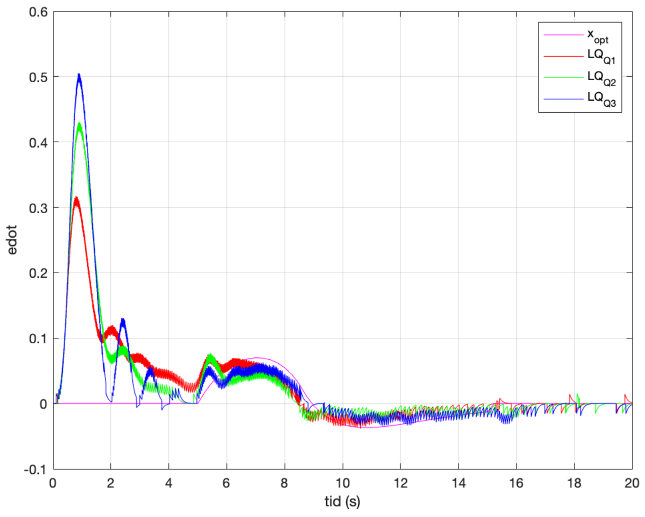


Figure 35: $\dot{e}$ with $\boldsymbol{LQ\_Q}_1$, $\boldsymbol{LQ\_Q}_2$ and $\boldsymbol{LQ\_Q}_3$

From figure 30-35 the green plot, $\boldsymbol{LQ\_Q}_2$, is not as stable and smooth as the others, even though $\boldsymbol{K}_2$ matrix looks similar to $\boldsymbol{K}_1$.

In figure 30 and 31 we can see that the plots follows the same trajectory. $\boldsymbol{LQ\_Q}_2$ almost has the same "bumps" as $\boldsymbol{x}^*$. For example at around 15 seconds mark in figure 30 and 31 , both $\boldsymbol{x}^*$ and $\boldsymbol{LQ\_Q}_2$ has overshoot. Metaphorically we can say that $\boldsymbol{LQ\_Q}_2$ trust $\boldsymbol{x}^*$ more, and overccorrects in the same way. $\boldsymbol{LQ\_Q}_1$ and $\boldsymbol{LQ\_Q}_3$ has a more smooth path, and don't have to "travel back" like $\boldsymbol{LQ\_Q}_2$ and $\boldsymbol{x}^*$ does. In the same methaporical way we can say that $\boldsymbol{LQ\_Q}_1$ and $\boldsymbol{LQ\_Q}_3$ trust the physical system more, and let it affect the input $\boldsymbol{u}$ more. An additional remark is that $\boldsymbol{LQ\_Q}_3$ gets closer to $\boldsymbol{x}_f$ by a small margin.

This can also be seen in the plots of the pitch angle and rate in figure 32 and 33. $\boldsymbol{LQ\_Q}_1$ and $\boldsymbol{LQ\_Q}_2$ starts with the same trajectory, and after some seconds $\boldsymbol{LQ\_Q}_2$ starts converging towards $\boldsymbol{x}^*$. At the end of the measurements the distance between the trajectory $\boldsymbol{x}$ and the optimal path $\boldsymbol{x}^*$ is so large that $\boldsymbol{LQ\_Q}_2$ almost becomes unstable. This is due to that the optimal path $\boldsymbol{x}^*$ has a small bump at around 14-15 seconds that throws $\boldsymbol{LQ\_Q}_2$ of path. $\boldsymbol{LQ\_Q}_1$ and $\boldsymbol{LQ\_Q}_3$ have throught the measurements followed $\boldsymbol{x}^*$ without letting the deviations of $\boldsymbol{x}^*$ affect their trajectory. This results in $\boldsymbol{LQ\_Q}_1$ and $\boldsymbol{LQ\_Q}_3$ having a smoother path and manages to keep the state $\boldsymbol{x}_f$ after 15 seconds.

Since the values of $\boldsymbol{K}(2{,}5{:}6)$ have more diversity in $\boldsymbol{K}_1$, $\boldsymbol{K}_2$ and $\boldsymbol{K}_3$. As seen in table 5, the gain of the elevation rate ($\boldsymbol{K}(2{,}6)$) is approximately the same for $\boldsymbol{LQ\_Q}_1$ and $\boldsymbol{LQ\_Q}_3$. Actually, even thought $\boldsymbol{LQ\_Q}_1$ and $\boldsymbol{LQ\_Q}_3$ have complety different values (i.e how much we want to penalize the states with respect to each other) we end up with nearly the same $\boldsymbol{K}$ matrix. The only difference between $\boldsymbol{K}_1$ and $\boldsymbol{K}_3$ is that elevation angle gain, $\boldsymbol{K}(2{,}5)$, almost has a ratio of 5. This can be seen in figure 34, where $\boldsymbol{LQ\_Q}_1$ moves slower than $\boldsymbol{LQ\_Q}_2$ and $\boldsymbol{LQ\_Q}_3$.

$\boldsymbol{LQ\_Q}_3$ is the $\boldsymbol{LQ\_Q}$ matrix that we ended up after tuning the helicopter. The trajectory is smooth and it manages to get closes to $\boldsymbol{x}_f$ without drifting off. Together with $\boldsymbol{LQ\_R}$, $\boldsymbol{Q}$ and $\boldsymbol{R}$ as described above we have defined all our variables for our system.

4.4 Decoupled model

As mentioned in our analysis of the gain matrix $\boldsymbol{K}$, we suggested that the 4 states are decoupled from each other. Although the plots in figure 30-35 informs of otherwise. Physically this makes sense. When the pitch angle is nonzero we will increase or decrease the elevation, e, and travel, λ, which will in turn affect their rates respectively. Therefore, will a penalising in one state affect the other five states. Therefore, it is not possible to calculate a optimal path for all states and inputs. A solution to improve this is to have a more complete model or we could introduce another state that indicates how the states covariance between them. This might result in a nonlinear system that could be difficult to compute.

4.5 MATLAB and Simulink

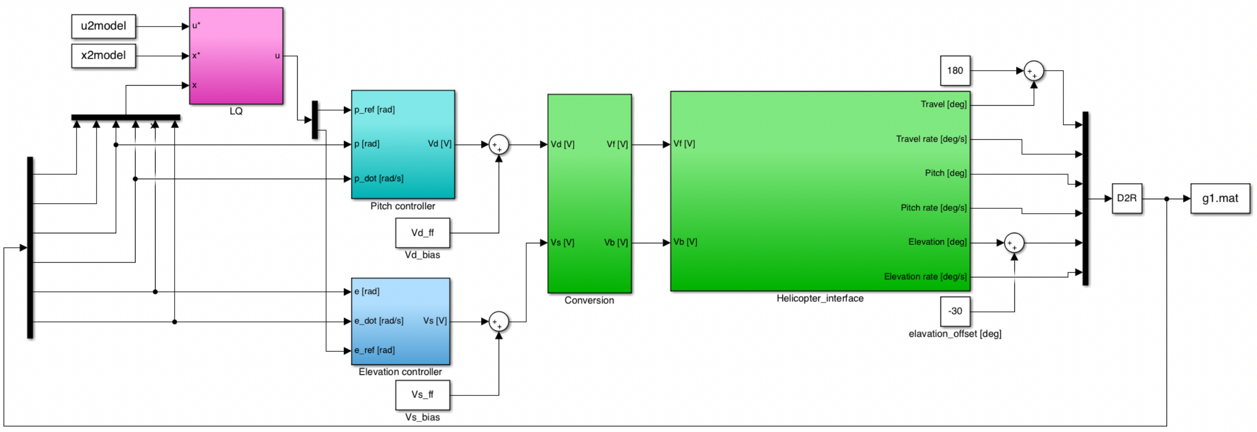


Figure 36: Simulink day4

```
%Elevation feedback
%% Initialization and model definition
init;

% Discrete time system model. x = [lambda r p pdot]'
Ac = [0 1      0          0         0         0;
      0 0    -K2          0         0         0;
      0 0      0          1         0         0;
```

```
        0 0    -K1*Kpp -K1*Kpd     0         0;
        0 0       0          0         0         1;
        0 0       0          0    -K3*Kep -K3*Ked];

Bc = [0        0;
        0        0;
        0        0;
        K1*Kpp 0;
        0        0;
        0        K3*Kep];

%discritize system
deltat = 0.25;
Ad = eye(mx) + deltat*Ac;
Bd = Bc*deltat;

 % Number of states and inputs
mx = size(Ad,2); % Number of states (number of columns in A)
mu = size(Bd,2); % Number of inputs(number of columns in B)

% Initial values
x10 = pi;                                % Lambda
x20 = 0;                                 % r
x30 = 0;                                 % p
x40 = 0;                                 % pdot
x50 = 0;                                 % e
x60 = 0;                                 % edot
x0 = [x10 x20 x30 x40 x50 x60]';  % Initial values

% Time horizon and initialization
N  = 40;                    % Time horizon for states
M  = N;                     % Time horizon for inputs
z  = zeros(N*mx+M*mu,1);    % Initialize z for the whole horizon
z0 = z;                     % Initial value for optimization
z0(1:mx) = x0;

%% LQController
LQQ = diag([12  3  2  5 10  1]);
LQR = diag([0.1 0.1]);
[K, S, e] = dlqr(Ad, Bd, LQQ, LQR);

%% Construction fmincon
%Values of fmincon
```

```
alpha = 0.2;
beta = 20;
lambdat = 2*pi/3;

q1 = 0.1;
q2 = 0.1;

% Generate the matrix Q and the vector c (objecitve function
   weights in the QP problem)
P1 = diag([q1 q2]);                % Weight on input
Q1 = diag([2 0 0 0 0 0]);
G = 2*genq(Q1,P1,N,M);             % gnerate Q,
c = zeros(N*mx+M*mu,1);

%% Generate system matrixes for linear model
Aeq = genaeq(Ad,Bd,N,mx,mu);  % Generate A,
beq = zeros(size(Aeq,1),1);        % Generate b
beq(1:mx) =Ad*x0;

%% Bounds
pk = 30*pi/180;
ul = [-pk; -inf];            % Lower bound on control
uu = [pk; inf];              % Upper bound on control

xl(1:mx,1) = -Inf*ones(mx,1); % Lower bound on states(no bound)
xu(1:mx,1) = Inf*ones(mx,1);  % Upper bound on states(no bound)
xl(3) = ul(1);                % Lower bound on state x3 dotp
xu(3) = uu(1);                % (Opt.) Upper bound on state x3
xl(2) = 2*ul(1);              % (Opt.) Lower bound on state x2
   r
xu(2) = 2*uu(1);              % (Opt.) Upper bound on state x2
xl(6) = ul(1);                % (Opt.) Lower bound on state x6
xu(6) = uu(1);                % (Opt.) Upper bound on state x6

% Generate constraints on measurements and inputs
[vlb,vub] = genconstraints(N,M,xl,xu,ul,uu);
vlb(N*mx+M*mu) = 0;      % We want the last input to be zero
vub(N*mx+M*mu) = 0;      % We want the last input to be zero

%nonlinear constraint
fun = @(z) 1/2*z'*G*z;
opt = optimoptions('fmincon', 'Algorithm', 'sqp');
tic;
```

```
z = fmincon(fun,z0,[],[],Aeq,beq,vlb,vub,@nonlinearconstraint,
    opt);
toc;

%% Extract control inputs and states
u1 = [z(N*mx+1:mu:N*mx+M*mu);z(N*mx+M*mu-1)]; % Control input
   from solution
u2 = [z(N*mx+2:mu:N*mx+M*mu);z(N*mx+M*mu)];
x1 = [x0(1);z(1:mx:N*mx)];       % State x1 from solution
x2 = [x0(2);z(2:mx:N*mx)];       % State x2 from solution
x3 = [x0(3);z(3:mx:N*mx)];       % State x3 from solution
x4 = [x0(4);z(4:mx:N*mx)];       % State x4 from solution
x5 = [x0(5);z(5:mx:N*mx)];       % State x5 from solution
x6 = [x0(6);z(6:mx:N*mx)];       % State x6 from solution

numvariables = 5/deltat;
zeropadding = zeros(numvariables,1);
unitpadding  = ones(numvariables,1);

u1   = [zeropadding; u1; zeropadding];
u2   = [zeropadding; u2; zeropadding];
x1  = [pi*unitpadding; x1; zeropadding];
x2  = [zeropadding; x2; zeropadding];
x3  = [zeropadding; x3; zeropadding];
x4  = [zeropadding; x4; zeropadding];
x5  = [zeropadding; x5; zeropadding];
x6  = [zeropadding; x6; zeropadding];

%% Plotting
t = 0:deltat:deltat*(length(u1)-1);

u = [u1 u2];
u2model = [t', u];

%Sending data
xopt = [x1 x2 x3 x4 x5 x6];
x2model = [t', xopt];
```

4.6 Optional exercise

We tried five different variations of constraints on the states. We changed the value of p_k and added constraints on the states $\dot{e}$ and r. The different constraints are listed in table 6, where the states with constraints are marked with red.

constraint number	p_k	$x_{low} \leq x \leq x_{up}$
1	$\frac{30\pi}{180}$	$\begin{bmatrix} -\infty \\ -p_k \\ -p_k \\ -\infty \\ -\infty \\ -p_k \end{bmatrix} \leq \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} \leq \begin{bmatrix} \infty \\ p_k \\ p_k \\ \infty \\ \infty \\ p_k \end{bmatrix}$
2	$\frac{30\pi}{180} = \frac{\pi}{6}$	$\begin{bmatrix} -\infty \\ -2p_k \\ -p_k \\ -\infty \\ -\infty \\ -p_k \end{bmatrix} \leq \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} \leq \begin{bmatrix} \infty \\ 2p_k \\ p_k \\ \infty \\ \infty \\ p_k \end{bmatrix}$
3	$\frac{\pi}{180}$	$\begin{bmatrix} -\infty \\ -2p_k \\ -p_k \\ -\infty \\ -\infty \\ -p_k \end{bmatrix} \leq \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} \leq \begin{bmatrix} \infty \\ 2p_k \\ p_k \\ \infty \\ \infty \\ p_k \end{bmatrix}$
4	$\frac{15\pi}{180} = \frac{\pi}{12}$	$\begin{bmatrix} -\infty \\ -2p_k \\ -p_k \\ -\infty \\ -\infty \\ -p_k \end{bmatrix} \leq \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} \leq \begin{bmatrix} \infty \\ 2p_k \\ p_k \\ \infty \\ \infty \\ p_k \end{bmatrix}$
5	$\frac{30\pi}{180} = \frac{\pi}{6}$	$\begin{bmatrix} -\infty \\ -2p_k \\ -p_k \\ -\infty \\ -\infty \\ -\infty \end{bmatrix} \leq \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} \leq \begin{bmatrix} \infty \\ 2p_k \\ p_k \\ \infty \\ \infty \\ \infty \end{bmatrix}$

Table 6: Contraints

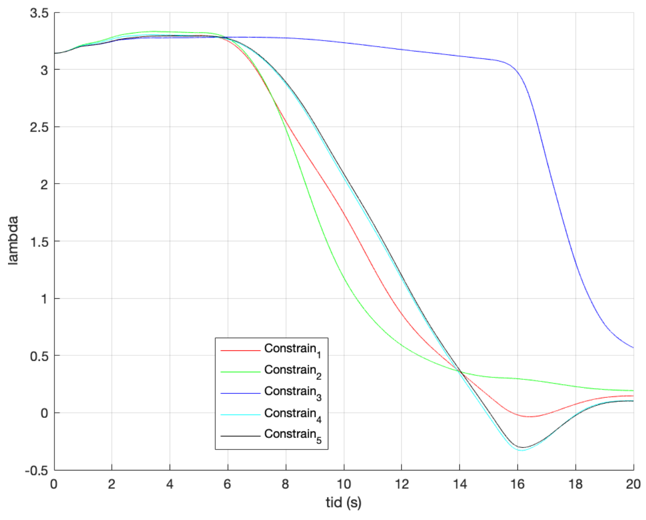


Figure 37: λ with constraints

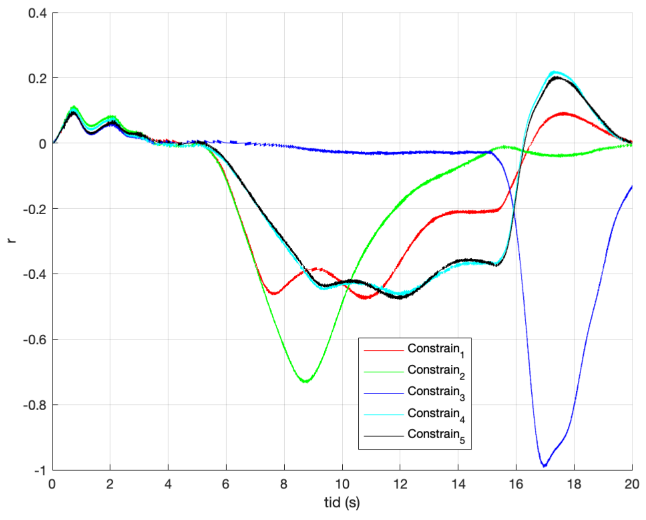


Figure 38: r with constraints

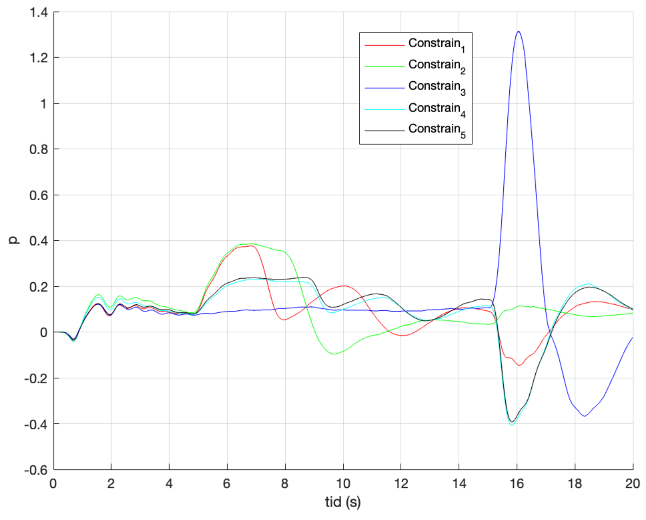


Figure 39: p with constraints

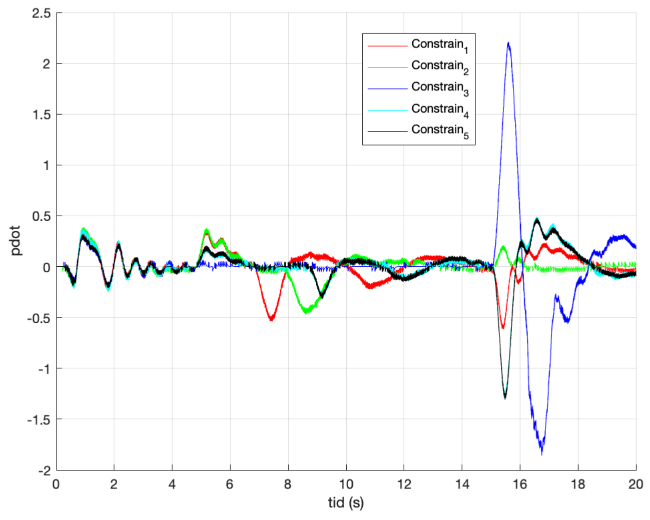


Figure 40: $\dot{p}$ with constraints

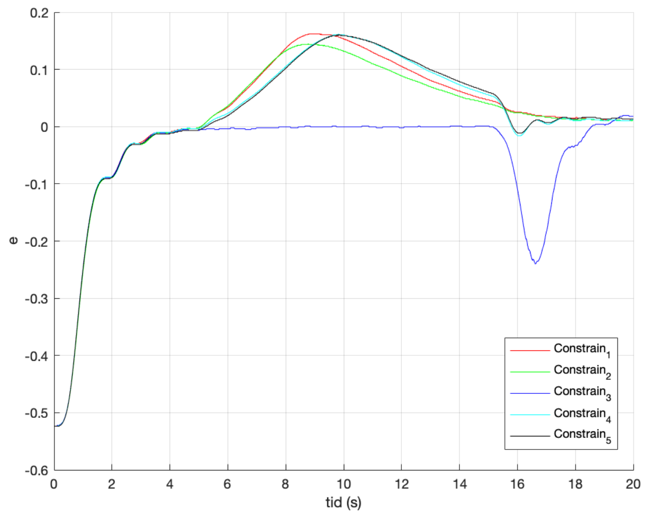


Figure 41: e with constraints

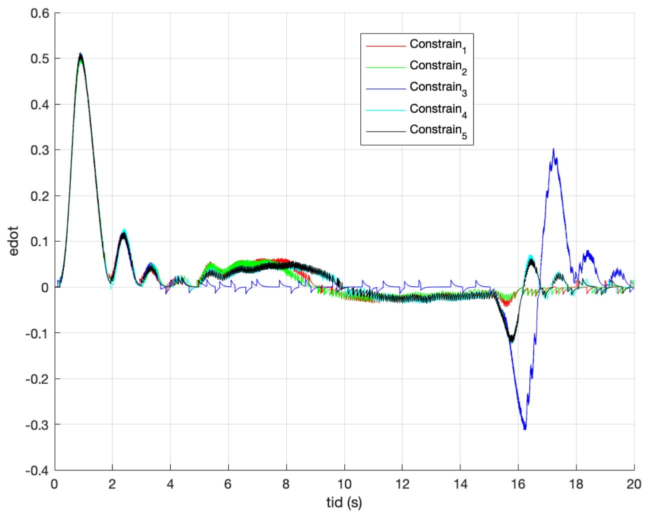


Figure 42: $\dot{e}$ with constraints

The difference between constraint-1 and constraint-2 is that the constraint on the travel rate, r, has been double to $2p_k$. This gives a larger feasible area and the travel rate has more "room" to move. The plots of the travel rate in figure 38 shows exactly this. The green line goes further down and becomes zero after 15 seconds. The helicopter reaches $\boldsymbol{x}_f$ faster, which gives it an advantage in the other states too.

The most noticeable anomaly in the plots in figure 37-42 is the blue plot, constraint 3. From table 6 we know that this is the constraint case with smallest p_k, i.e smallest feasible area. The helicopter lays still until 15 seconds has passed, and then acts abruptly. Therefore having $p_k = \frac{\pi}{180}$ makes a feasible area so small, that the only optimal trajectory reached is zero-valued.

The only difference between constraint-4 and constraint-5 is that constraint-5 don't have constraint on elevation. However the optimal trajectory is the same, where the cyan and black plot is completely the same. Therefore having the constraint on $\dot{e}$ at $\frac{30\pi}{180}$ is useless, because the optimal point exists inside of the restriction. The optimal trajectory for $\boldsymbol{x}$ and $\boldsymbol{u}$ that is affected by $\dot{e}$ exists in an area smaller than $\frac{30\pi}{180}$.

5 Conclusion

Throughout the lab assignment a few control methods based on optimal control theory have been tested and compared. The goal was to get a helicopter from a starting point $\boldsymbol{x_0}$ to a final point $\boldsymbol{x_f}$. In the first two controllers, only pitch and travel was considered. It started off with an open loop solution which simulated a system unable to reach its destination. Then, feedback was added as well as an LQ controller. This solution made the system more accurate, and it got closer to the desired destination. Finally, elevation was added to the model and got constrained by a non-linear constraint to prevent it from going too far over desired value. All in all, this lab has given a good introduction to optimization based control.

References

[1] Forward and backward euler methods. `https://web.mit.edu/10.001/Web/CourseNotes/DifferentialEquationsNotes/node3.html`. Accessed: 2021-02-25.

[2] dlqr. `https://se.mathworks.com/help/control/ref/dlqr.html`. Accessed: 2021-03-28.

[3] B. Foss and T. A. N. Heirung. *Merging Optimization and Control.* NTNU, 2016.