Helicopter Lab Report

Student 528076
Student 528113

April 20, 2022

Department of Engineering Cybernetics

Contents

1 10.2 - Optimal Control of Pitch/Travel without Feedback

The goal of the first part of the lab is to formulate a convex, quadratic optimization problem, and calculate the optimal input sequence that pivots the helicopter 180 degrees. This is done with a QP solver in MATLAB.

1.1 The continuous model

The continuous closed loop model for the helicopter is given by

$$\begin{aligned}
&\ddot{e} + K_3 K_{ed}\dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c && \text{(1a)}\\
&\dot{\lambda} = r && \text{(1b)}\\
&\dot{r} = -K_2 p && \text{(1c)}\\
&\ddot{p} + K_1 K_{pd}\dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c && \text{(1d)}
\end{aligned}$$

where e is elevation, λ is travel, r is travel rate, and p is pitch. The model contains the dynamics of the helicopter in a feedback loop with a PD-controller, where the input is the pitch reference p_c. In this part the elevation e of the helicopter is not taken into account, and we assume $e = 0$. The system of equations can be written on the form

$$\dot{\mathbf{x}} = \mathbf{A}_c\mathbf{x} + \mathbf{B}_c u, \tag{2}$$

where $\mathbf{x} = \begin{bmatrix}\lambda & r & p & \dot{p}\end{bmatrix}^\top$ and $u = p_c$. $\mathbf{A}_c$ and $\mathbf{B}_c$ are given by

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} \end{bmatrix} \tag{3a}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1 K_{pp} \end{bmatrix} \tag{3b}$$

The model is used in the optimization layer to calculate the optimal trajectory $\mathbf{x}^*$, and a corresponding optimal input sequence u^*. The optimal input is fed into the control layer, which in turn outputs two voltages that control the pitch and elevation of the physical helicopter. Altogether, the three layers comprise an open loop optimal control system. The structure of the control system can be seen in Figure 1.

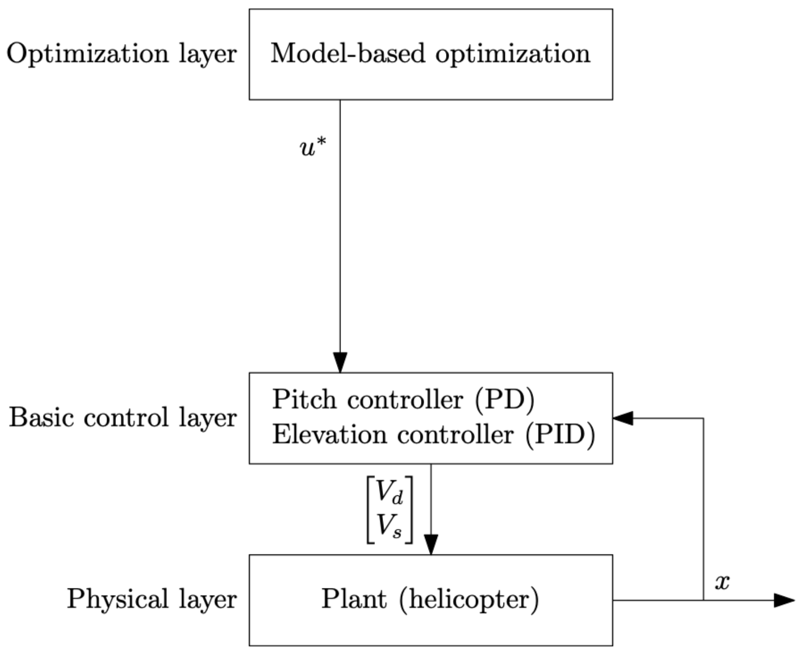

Figure 1: Diagram of the layers in the control hierarchy [1]

1.2 The discretized model

To discretize the state space model we utilize the forward Euler method. Numerical methods such as forward Euler, introduces some numerical error, but this can be reduced using a short timestep. The forward Euler method is

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \dot{\mathbf{x}}_k \tag{4}$$

Inserting the continuous model for $\dot{\mathbf{x}}_k$ yields,

$$\mathbf{x}_{k+1} = \underbrace{(\mathbf{I} + \Delta t \mathbf{A}_c)}_{\mathbf{A}_d} \mathbf{x}_k + \underbrace{\Delta t \mathbf{B}_c}_{\mathbf{B}_d} u_k \tag{5}$$

where

$$\mathbf{A}_d = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & -\Delta t K_2 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & -\Delta t K_1 K_{pp} & 1 - \Delta t K_1 K_{pd} \end{bmatrix} \tag{6a}$$

$$\mathbf{B}_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\Delta t K_1 K_{pp} \end{bmatrix} \tag{6b}$$

For this task and the rest we use $\Delta t = 0.25s$.

1.3 The open loop optimization problem

The goal is to move the helicopter from $\mathbf{x}_0 = \begin{bmatrix}\lambda_0 & 0 & 0 & 0\end{bmatrix}^\top$ to $\mathbf{x}_f = \begin{bmatrix}\lambda_f & 0 & 0 & 0\end{bmatrix}^\top$, with $\lambda_0 = \pi$ and $\lambda_f = 0$. To calculate the optimal trajectory we minimize the following quadratic objective function

$$\phi = \sum_{i=0}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + q p_{ci}^2, \quad q \geq 0 \tag{7}$$

with the constraint

$$|u_k| = |p_k| \leq \frac{30\pi}{180}, \quad k \in \{1, ..., N\}, \quad N = 100 \tag{8}$$

As we will use the MATLAB function `quadprog` to solve the minimization problem, we need to reformulate the problem into standard form

$$\begin{aligned} &\min_z \ \frac{1}{2}\mathbf{z}^\top \mathbf{G}\mathbf{z} \\ &\quad s.t \\ &\begin{cases} \mathbf{A}_{eq}\mathbf{z} = \mathbf{b}_{eq} \\ u_l \leq u \leq u_u \end{cases} \end{aligned} \tag{9}$$

where the optimization variable $\mathbf{z} = \begin{bmatrix}\mathbf{x}_1^\top, \dots, \mathbf{x}_N^\top, u_0^\top, \dots, u_{N-1}^\top\end{bmatrix}$ contains the state trajectory and inputs in the event horizon. $u_l = -\frac{30\pi}{180}$ and $u_u = \frac{30\pi}{180}$ are the input constraints. The matrix $\mathbf{G}$ is a diagonal matrix containing the weights on the states and input.

$$\mathbf{G} = \begin{bmatrix} \mathbf{Q_1} & & & & & \\ & \ddots & & & & \\ & & \mathbf{Q_1} & & & \\ & & & \mathbf{R_1} & & \\ & & & & \ddots & \\ & & & & & \mathbf{R_1} \end{bmatrix} \tag{10}$$

As travel λ is the only state present in the objective function, the corresponding diagonal element in $\mathbf{Q_1}$ will be the only non-zero value in the matrix.

Furthermore, the QP standard form requires that we multiply by a factor of $\frac{1}{2}$, resulting in $\mathbf{Q_1}$ being defined as

$$\mathbf{Q_1} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

Due to p_c being the only input to the system, the weight $\mathbf{R_1} = 2q$. The model constraint have to be formulated in such a way that it includes every time step in the event horizon. We define

$$\mathbf{A}_{eq} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & -\mathbf{B} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ -\mathbf{A} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & -\mathbf{B} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & -\mathbf{A} & \mathbf{I} & \ddots & \vdots & \mathbf{0} & \mathbf{0} & -\mathbf{B} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathbf{0} & \vdots & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & -\mathbf{A} & \mathbf{I} & \mathbf{0} & \dots & \dots & \mathbf{0} & -\mathbf{B} \end{bmatrix} \tag{12a}$$

$$\mathbf{b_{eq}} = \begin{bmatrix} \mathbf{Ax}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \tag{12b}$$

This optimization problem is a QP-problem with linear constraints. As $\mathbf{G} \geq 0$ the problem is convex, meaning it can easily be solved with a QP algorithm. `quadprog` offers three different algorithms for optimization, but defaults to the interior point method, which only handles convex problems.

1.4 The objective function

When we steer the helicopter from λ_0 to λ_f, the first term in the objective function goes to zero and we are left with

$$\phi = \sum_{i=0}^{N-1} q p_{ci}^2 \tag{13}$$

The objective function is now minimized by $p_c = 0$, meaning there will be input of zero to the controller when the trajectory reaches λ_f. Due to there only being feedback on pitch, the system does not know if it follows the optimal path in regards to travel. Slight deviations from the optimal trajectory in pitch can create unwanted momentum in travel direction, causing unwanted drift. We need the model to perfectly describe the dynamics of the system for open loop optimization to be successful.

The objective function also runs the risk of providing a unnecessary cumbersome trajectory. If for example $\lambda_0 = 2\pi$ and $\lambda_f = 0$, the optimal trajectory calculated would pivot the helicopter from 2π to 0, while in fact the helicopter initially was positioned in the desired endpoint. This problem could be solved by preprocessing $(\lambda_0 - \lambda_f)$ using some simple modular arithmetic.

1.5 The weights of the optimization problem

Our experimentation with different values of q showed that a higher q limits the input capacity. In Figure 2 we see that $q = 10$ yielded a very restrained input sequence. This resulted in a trajectory far off the target, as showcased in Figure 3. Decreasing the weight to $q = 1$ gave a less conservative input usage. This allowed for faster control, and resulted in the helicopter maneuvering slightly closer to the intended trajectory. When further lowering to $q = 0.1$, the pitching was even more aggressive, perhaps turning the helicopter too quickly. It is clear from Figure 3 that all trajectories were heavily affected by drift in the positive travel direction, making them somewhat difficult to compare.

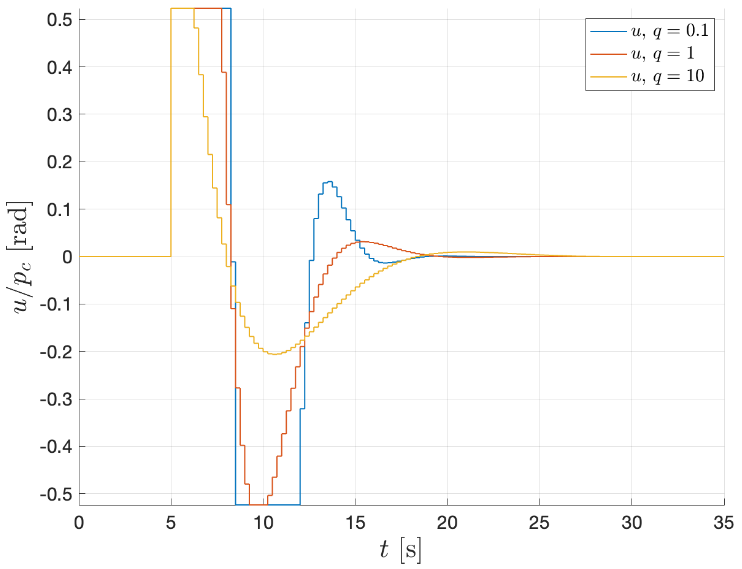

Figure 2: Optimal input sequence with different q

1.6 Experimental results

As is evident from Figure 4 the helicopter does not reach the target λ_f. Model imperfections and asymmetry in the hardware, causes the helicopter to drift off in the positive travel direction. As pitch is the only state being fed back, the controller can only assume that the helicopter has reached λ_f, after having tracked the input sequence. The lack of feedback on travel and travel rate makes it impossible to do corrections underway, and hence the performance primarily depends on the quality of the model. Judging by the results of the experiments, it is safe to say that the model is inadequate for open loop control. In the following tasks we will introduce feedback by implementing a LQ (*Linear Quadratic*) controller. This addition should drastically improve the performance of the helicopter.

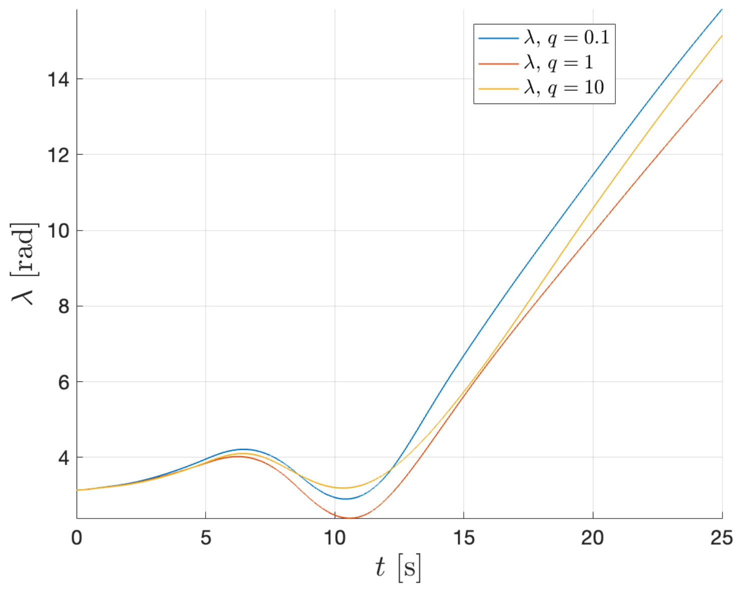

Figure 3: Recorded travel trajectory with different q

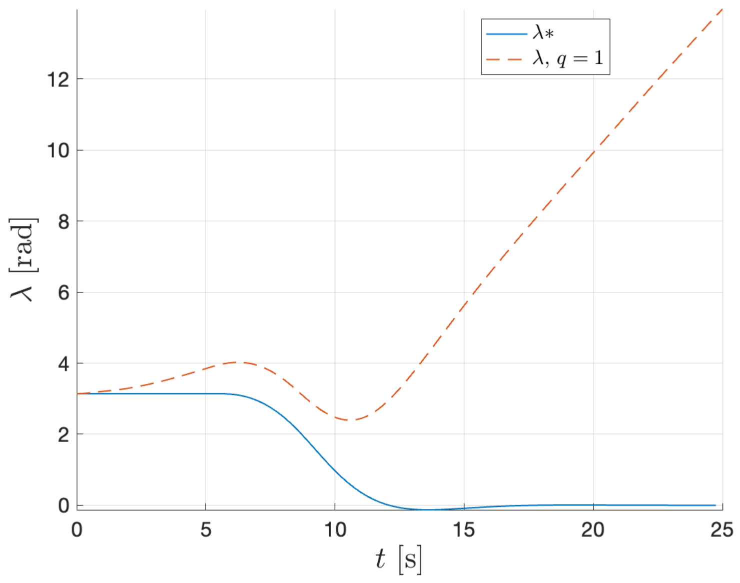

Figure 4: Optimal vs. recorded path

1.7 MATLAB and Simulink

Listing 1: MATLAB code - Open Loop Optimization

```
init08;
A_c = [0 1 0 0;
       0 0 -K_2 0;
       0 0 0 1;
       0 0 -K_1*K_pp -K_1*K_pd];
B_c = [0 0 0 K_1*K_pp]';

mx = size(A_c,2); mu = size(B_c,2);

dt = 0.25;
A_d = eye(mx) + dt*A_c;
B_d = dt*B_c;

x0 = [pi 0 0 0]';
N  = 100; M  = N;
z  = zeros(N*mx+M*mu,1); z0 = z;

ul = -pi/6; uu = pi/6;
xl = -Inf*ones(mx,1); xu = Inf*ones(mx,1);
xl(3) = ul; xu(3) = uu;

[vlb,vub] =  gen_constraints(N,M,xl,xu,ul,uu);
vlb(N*mx+M*mu)  = 0; vub(N*mx+M*mu)  = 0;
```

```

Q = diag([1 0 0 0]);
R = 1;
I_N = eye(N); I_M = eye(M);
G = 2*blkdiag(kron(I_N, Q), kron(I_M, R));

Aeq = gen_aeq(A_d,B_d,N,mx,mu);
beq = [A_d*x0; zeros((N-1)*mx,1)];

tic
[z,lambda] = quadprog(G,[],[],[],Aeq,beq,vlb,vub,x0);
t1=toc;

padding_time = 5;
num_variables = padding_time/dt;
zero_padding = zeros(num_variables,1);
unit_padding  = ones(num_variables,1);

u  = [z(N*mx+1:N*mx+M*mu);z(N*mx+M*mu)];
u   = [zero_padding; u; zero_padding];
x1  = [pi*unit_padding; x0(1); z(1:mx:N*mx); zero_padding];
x2  = [zero_padding; x0(2); z(2:mx:N*mx); zero_padding];
x3  = [zero_padding; x0(3); z(3:mx:N*mx); zero_padding];
x4  = [zero_padding; x0(4); z(4:mx:N*mx); zero_padding];

t = 0:dt:dt*(length(u)-1);
u_t = timeseries(u,t);
```

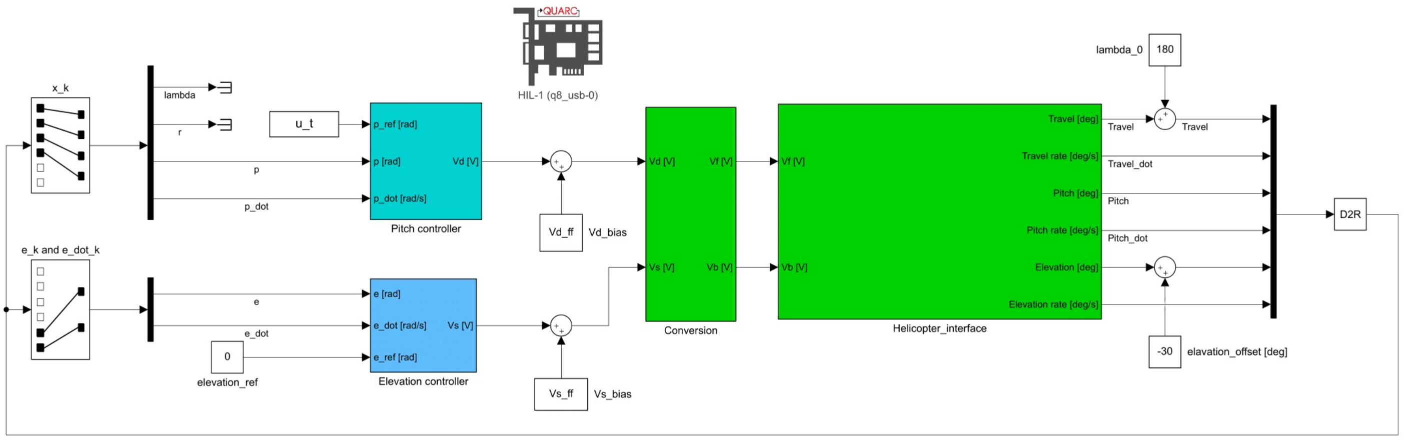

Figure 5: Simulink Diagram

2 10.3 - Optimal Control of Pitch/Travel with Feedback (LQ)

In this part we will introduce feedback by adding a LQ controller, in order to correct for disturbances and model errors.

2.1 LQ controller

To implement a LQ controller the manipulated variable is updated to

$$\mathbf{u}_k = \mathbf{u}_k^* - \mathbf{K}^\top(\mathbf{x}_k - \mathbf{x}_k^*) \tag{14}$$

where $\mathbf{u}_k^*$ and $\mathbf{x}_k^*$ are the optimal trajectories calculated in section 1. If the helicopter follows the optimal trajectory, then $\mathbf{u}_k = \mathbf{u}_k^*$. If there is a deviation between the trajectories, $\mathbf{u}_k$ is adjusted by the feedback term. The performance of the state feedback controller is determined by the choice of the feedback gain matrix $\mathbf{K}$. In order to find a good choice for $\mathbf{K}$ we solve the infinite-horizon LQ problem, which is also referred to as LQR. The problem is formulated as

$$\min_{\Delta\mathbf{x},\Delta\mathbf{u}} J = \sum_{i=0}^{\infty} \Delta\mathbf{x}_{i+1}^\top \mathbf{Q} \Delta\mathbf{x}_{i+1} + \Delta\mathbf{u}_i^\top \mathbf{R} \Delta\mathbf{u}_i,\ \mathbf{Q} \geq 0, \mathbf{R} > 0 \tag{15}$$

for a linear model

$$\Delta\mathbf{x}_{i+1} = \mathbf{A_d}\Delta\mathbf{x}_i + \mathbf{B_d}\Delta\mathbf{u}_i \tag{16}$$

where

$$\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^* \tag{17a}$$
$$\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}^* \tag{17b}$$

As long as (16) is stabilizable there will always exists a well defined solution of the optimization problem. The solution provides the discrete-time Riccati equation

$$\mathbf{A_d}\mathbf{P}\mathbf{A_d} - \mathbf{P} - (\mathbf{A_d}^\top\mathbf{P}\mathbf{B_d})(\mathbf{B_d}^\top\mathbf{P}\mathbf{B_d} + \mathbf{R})^{-1}(\mathbf{B}^\top\mathbf{P}\mathbf{A_d}) + \mathbf{Q} = \mathbf{0} \tag{18}$$

which we can use to derive the state feedback gain matrix

$$\mathbf{K} = (\mathbf{B_d}^\top\mathbf{P}\mathbf{B_d} + \mathbf{R})^{-1}(\mathbf{B_d}^\top\mathbf{P}\mathbf{A}). \tag{19}$$

The implementation of LQR control adds an additional layer to the control hierarchy, illustrated in Figure 6.
In MATLAB the feedback gain matrix is easily found using the built-in function `dlqr`. The code is presented in listing 2

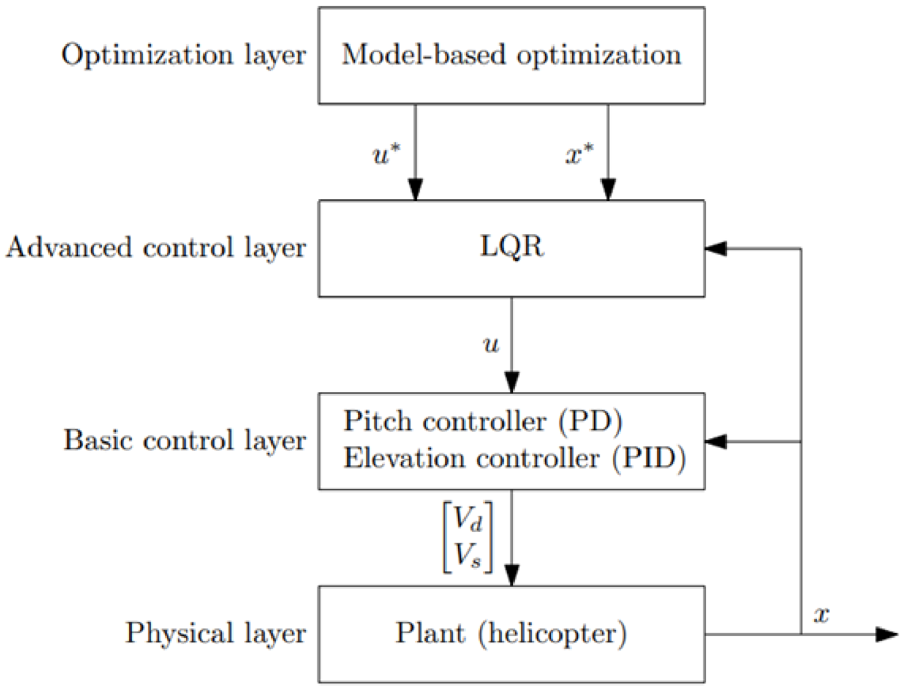

Figure 6: Diagram of the layers in the control hierarchy with LQR [1]

The **Q** and **R** matrices in the objective function are both diagonal matrices. In this case, R is a scalar as we only have a single input to the system. Each value along the diagonal of **Q** correspond to one of the states in **x**, and these values decide to what degree we wish to penalize deviation between state and the optimal trajectory. For example, if we wish to penalize deviation in λ we set Q_{11} high in relation to the other elements of **Q**. Similarly, each diagonal element of R correspond to one input in u. By adjusting values of R we decide how much we want to penalize deviation from the optimal input trajectory. In our testing of the LQ controller, we experimented with different values for the weights in **Q**. As travel is the only state present in the optimization problem, we wanted to mainly penalize deviation in λ. We found that this gave good performances. We also examined how the LQR manipulated the input, u. The results are discussed in section 2.3.

2.2 Model Predictive Control

An alternative strategy of implementing a closed loop optimal controller is MPC. Instead of adding an additional control layer, we introduce state feedback to the optimization layer. To realize a MPC controller we have to solve the QP problem from section 1 for a given horizon, at every time step. The first step of the optimal input sequence is implemented, while

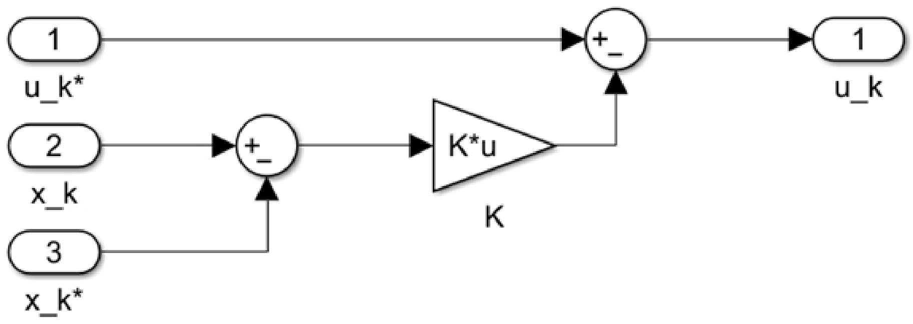

Figure 7: Simulink - State Feedback Implementation

the rest of the steps are disregarded. The feedback from measurements acts as the initial conditions in the optimization problems. While LQR is only subjected to the system dynamics, MPC also allows for arbitrary constraints on states and input. In our case, a MPC controller would always enforce the constraint $|p| \leq \frac{\pi}{6}$, whereas the LQR can potentially violate this. The clear advantage of MPC is that it redetermines the optimal path and input, at every sample point making it more robust against model error and disturbances. LQR conversely has to trust the initial solution, as it is unable to make any online corrections . Furthermore, since MPC generally does not make any assumptions about the linearity of the system, a nonlinear model could be utilized. This could make the system more robust when wandering away from the linearization point, but the convergence of a optimum is never a guarantee. LQR does however have a significant edge when it comes to processing power. As MPC calculates a new optimum at every time step, it can become quite computationally expensive.

2.3 Experimental results

In our experimentation we compared the difference in performance when penalizing λ and p. We adjusted only the values of $\mathbf{Q}$ while the parameters $q = 1$ and $R = 1$ were kept constant. From Figure 9 we notice a significant difference in the performance of the two configurations. By tuning Q_{11} high, the response of λ in the physical plant is closer to the optimal trajectory λ^* than by tuning Q_{33} high. This is expected as the system focuses on minimizing $\lambda - \lambda^*$ when Q_{11} highly weighted. Similarly, we see an improvement in pitch when tuning Q_{33} high in Figure 10. It is evident that adding feedback on travel and travel rate, mitigated the problem of drift in the positive travel direction.
By observing the new manipulated variable u_k in Figure 11, we note that it differs from the optimal input trajectory u^*. Especially from $t = 9$ s,

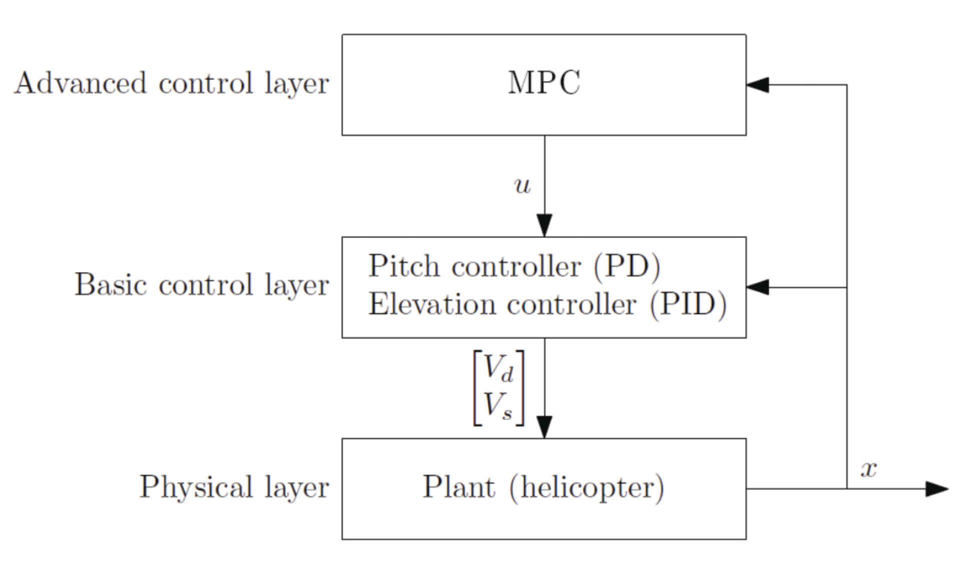

Figure 8: Diagram of the layers in the control hierarchy with MPC

the LQR pulls the input in the negative direction to achieve stationarity in travel. By observing Figure 10 and Figure 11 one can argue that it is necessary to keep $p \approx 0.10$ to avoid drift in λ.
In the approximate interval of $t \in [9, 11]$ s, u_k violates the input constraint of $|p| \leq \frac{\pi}{6}$, as mentioned was possible with LQR control in section 2.2. This shows that if the deviations and weights were large enough, the LQR could potentially destabilize the plant.

2.4 MATLAB and Simulink

Listing 2: MATLAB code - LQR

```
Q_lqr = diag([50 1 1 1]);
R_lqr = 1;
K = dlqr(A_d,B_d,Q_lqr,R_lqr);
```

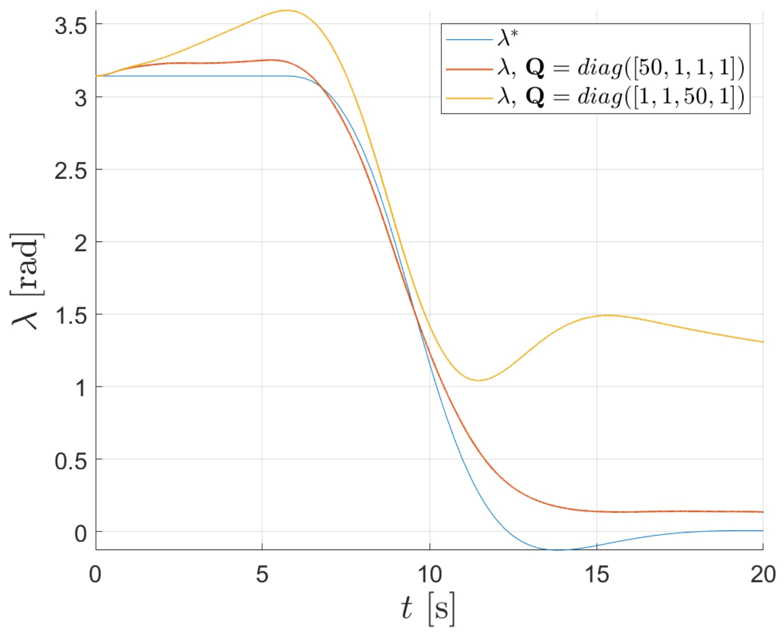

Figure 9: Travel: Open Loop Optimal Controller w/ LQR feedback

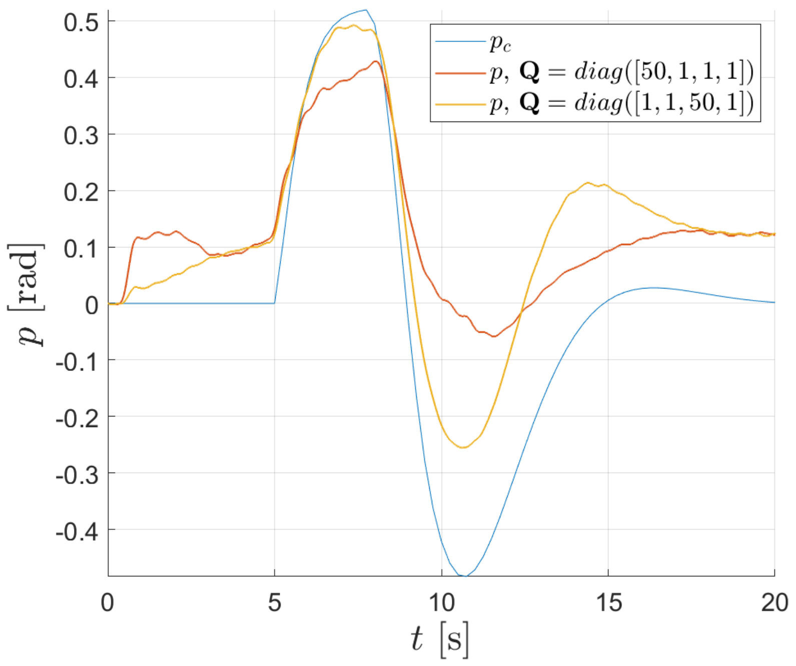

Figure 10: Pitch: Open Loop Optimal Controller w/ LQR feedback

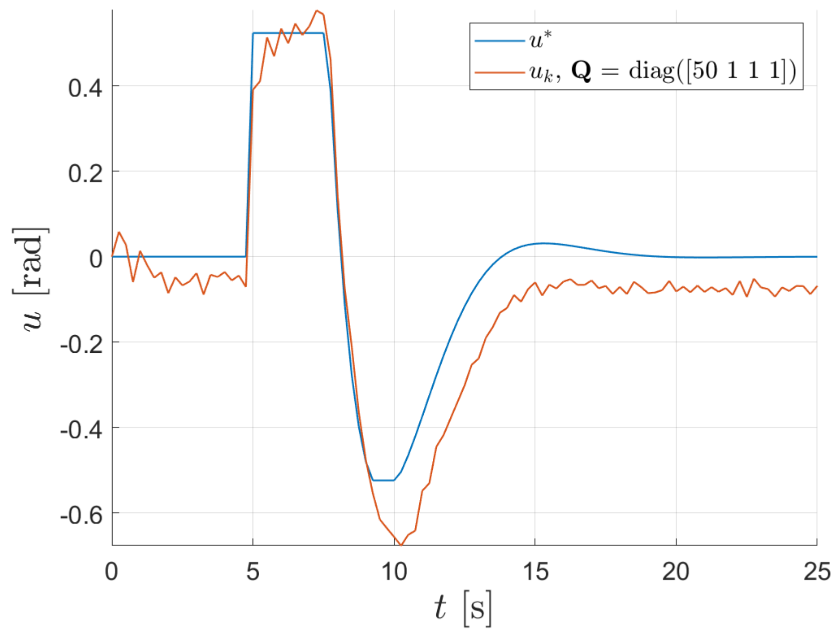

Figure 11: Comparison of u^* and u_k

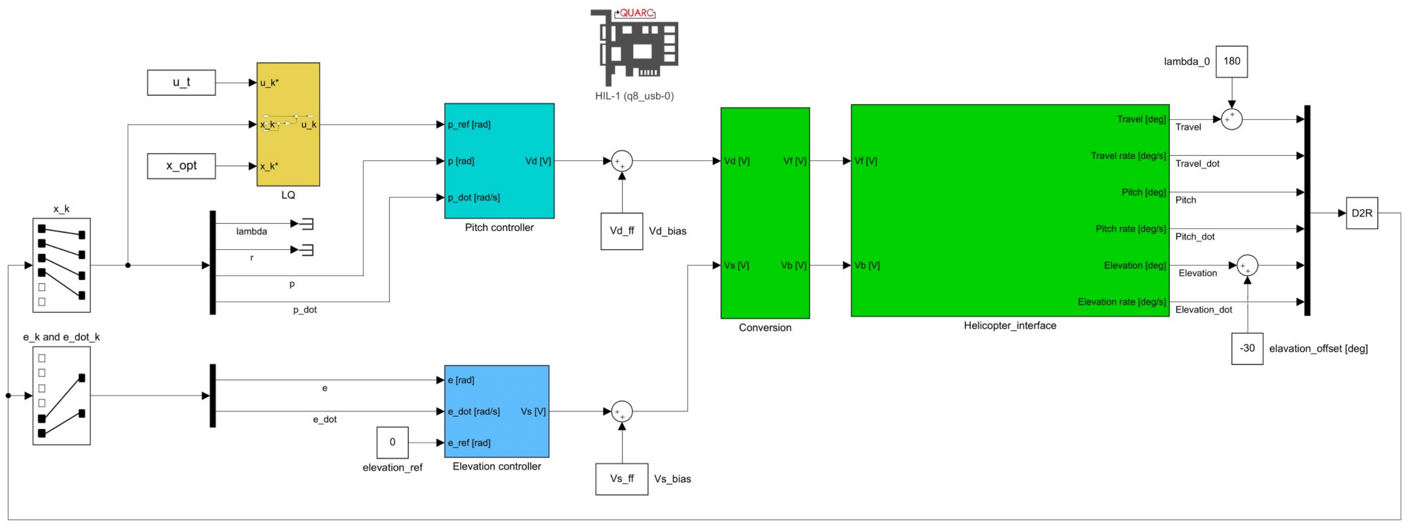

Figure 12: Day 3: Simulink Diagram

3 10.4 - Optimal Control of Pitch/Travel and Elevation with Feedback

In this section we will extend our model to include the states e and $\dot{e}$. The manipulated variable $\mathbf{u}$ will consist of two setpoints, p_c and e_c. Furthermore, a nonlinear constraint on elevation is added to the optimization problem, meaning that we need a nonlinear solver for our problem.

3.1 The continuous model

We update the state vector and the manipulated variable by including the equations for elevation and elevation rate from (1) such that

$$\mathbf{x} = \begin{bmatrix}\lambda & r & p & \dot{p} & e & \dot{e}\end{bmatrix}^\top, \mathbf{u} = \begin{bmatrix}p_c & e_c\end{bmatrix}^\top$$

which results in the following continous state space model

$$\begin{bmatrix}\dot{\lambda}\\ \dot{r}\\ \dot{p}\\ \ddot{p}\\ \dot{e}\\ \ddot{e}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & -K_2 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & -K_3K_{ep} & -K_3K_{ed}\end{bmatrix}\begin{bmatrix}\lambda\\ r\\ p\\ \dot{p}\\ e\\ \dot{e}\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ K_1K_{pp} & 0\\ 0 & 0\\ 0 & K_3K_{ep}\end{bmatrix}\begin{bmatrix}p_c\\ e_c\end{bmatrix}. \tag{20}$$

We include a term for e_c in the objective function

$$\phi = \sum_{i=0}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2. \tag{21}$$

3.2 The discretized model

To discretize the system we again make use of the forward Euler method as stated in (5). This provides a discrete state space model of the system with the following system and input matrices

$$\mathbf{A}_d = \begin{bmatrix}1 & \Delta t & 0 & 0 & 0 & 0\\ 0 & 1 & -\Delta t K_2 & 0 & 0 & 0\\ 0 & 0 & 1 & \Delta t & 0 & 0\\ 0 & 0 & -\Delta t K_1 K_{pp} & 1-\Delta t K_1 K_{pd} & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & \Delta t\\ 0 & 0 & 0 & 0 & -\Delta t K_3 K_{ep} & 1-\Delta t K_3 K_{ed}\end{bmatrix} \tag{22a}$$

$$\mathbf{B}_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \Delta t K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & \Delta t K_3 K_{ep} \end{bmatrix} \tag{22b}$$

3.3 The nonlinear constraint

The nonlinear constraint implemented in the optimization is stated as follows

$$e_k \geq \alpha \exp(-\beta(\lambda_k - \lambda_t)^2), \forall k \in \{1, ..., N\} \tag{23}$$

As λ_k approaches λ_t the expression grows and the maximum value of the constraint is reached when $\lambda_k = \lambda_t$. The constraint can be viewed as an imaginary wall that the helicopter needs to clear, as seen in Figure 13. The size of the wall is determined by the constants α and β, and the placement determined by λ_t.
The introduction of a nonlinear constraint in our optimization problem requires the use of a nonlinear solver. For this purpose we used the MATLAB function `fmincon`. The implementation of the nonlinear constraint in MATLAB can be seen in Listing 3.

Listing 3: MATLAB code - Nonlinear constraint

```matlab
function [c,ceq] = gen_con(z, param)
% Generate nonlinear constraint in elevation
% param = [alpha, beta, lambda_t, mx, N]
alpha = param(1); beta = param(2); lambda_t = param(3);
mx = param(4); N = param(5);
c = zeros(N,1);
for k = 1:N
    c(k) = alpha*exp(-beta*(z((k-1)*m+1)-lambda_t)^2) - z((k-1)*mx+5);
end
ceq = [];
end
```

3.4 Experimental results

The primary goal of our experimentation was to test if the helicopter could generate enough thrust to clear the nonlinear constraint in elevation, while also reaching λ_f in a controlled manner. As a starting point for LQR tuning, we used $\mathbf{Q} = \mathbf{I}_6$ and $\mathbf{R} = \mathbf{I}_2$. The latter remained constant in all the experiments. The resulting trajectory cleared about half the constraint in elevation, as seen in Figure 13. To make the controller provide more vertical thrust, we gradually increased Q_{55} to penalize deviations in elevation more.

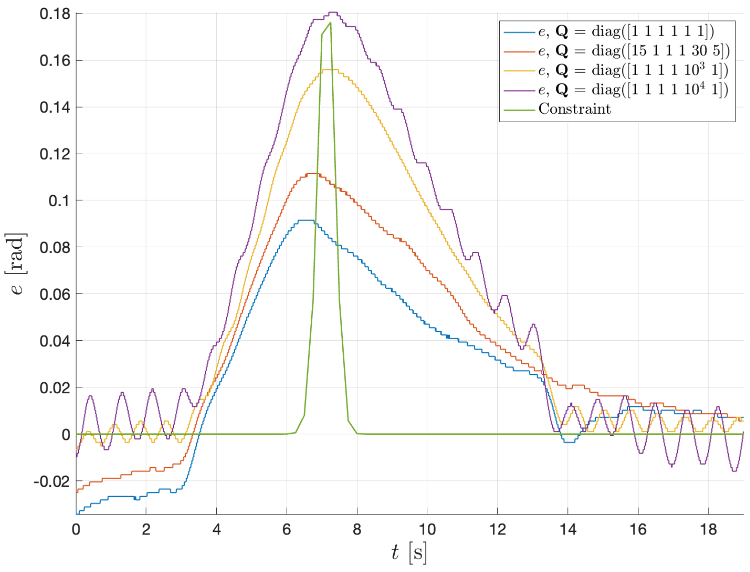

Figure 13: Recorded elevation trajectory with different LQR tuning

It quickly became clear that Q_{55} needed to be relatively large to be able to follow the intended trajectory, displayed in Figure 14. As Q_{55} increased, the control became increasingly more aggressive and the helicopter tended more towards instability. $Q_{55} = 10^4$ cleared the constraint by a slight margin, and just managed to remain stable. The corresponding feedback gain matrix $\mathbf{K}$ was

$$\mathbf{K} = \begin{bmatrix} -0.6488 & -2.4747 & 1.3122 & 0.4625 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40.9330 & 20.3295 \end{bmatrix} \tag{24}$$

It is clear that with this configuration, the LQ controller overcompensates when correcting the path in elevation. Even small deviations from the optimal trajectory will yield a high input due to the somewhat disproportional feedback gain. This results in the oscillatory behavior seen in Figure 13. The reason we need an unreasonably large weight to be able to clear the constraint, is the nonlinear relationship between e and p. This relation will be elaborated upon in subsection 3.5. As a consequence we had to compromise on the nonlinear constraint to get an overall satisfactory performance. The weights in Figure 14 were chosen to mainly penalize deviations in travel and elevation, while the pitch weight was kept relatively low to allow the helicopter to elevate efficiently. The helicopter did however not clear the nonlinear constraint, but the performance of the plant was satisfactory as a whole.

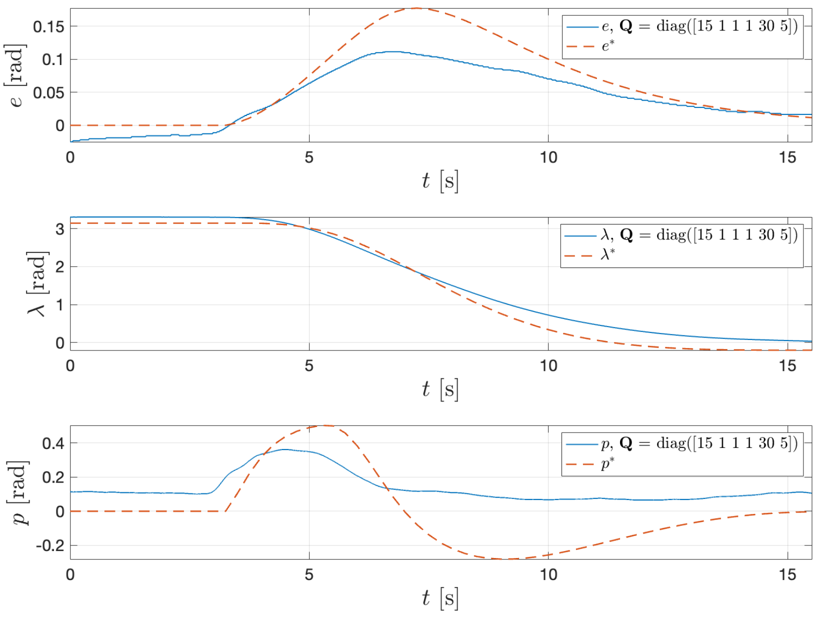

Figure 14: Optimal vs. recorded trajectory

3.5 Decoupled model

While inspecting the state space model we notice that e and $\dot{e}$ are decoupled from the rest of the states. This means that the controller can control the states separately, according to the state space model. In reality there is a nonlinear relation between the pitch and the elevation which is neglected in the model equations (1). As we tilt the pitch we will have a reduction in thrust upwards, meaning that a tilt in pitch will affect the movement in elevation. The assumption that pitch and elevation are decoupled is reasonable close to the operating point $p_p = 0$, $e_p = 0$, but by adding the nonlinear constraint in elevation we force the helicopter to migrate away from equilibrium, exposing the nonlinear relation between p and e. We see clearly from Figure 13 that the plant is struggling to satisfy the nonlinear constraint. This is, at least partly, a result of the optimal open loop controller calculating p^* and e^* as two independent trajectories. This is demonstrated in Figure 14 where pitch is at its maximum when elevation has its steepest ascent. As a result, the plant struggles to satisfy the constraint in elevation due to the reduction in upwards thrust as the pitch tilts. To improve this model one could implement a more complex, nonlinear model where elevation and pitch are coupled. However, this would require the use of a different controller as LQR assumes a linear model. Another option is to add stricter constraints on pitch, enabling the helicopter to provide more upwards thrust, thus allowing for better elevation tracking.

3.6 MATLAB and Simulink

Listing 4: MATLAB code - Nonlinear optimization

```
init08;
A_c = [0 1 0 0 0 0;
       0 0 -K_2 0 0 0;
       0 0 0 1 0 0;
       0 0 -K_1*K_pp -K_1*K_pd 0 0;
       0 0 0 0 0 1;
       0 0 0 0 -K_3*K_ep -K_3*K_ed];
B_c = [0 0;
       0 0;
       0 0;
       K_1*K_pp 0;
       0 0;
       0 K_3*K_ep];

mx = size(A_c,2); mu = size(B_c,2);

dt = 0.25;
A_d = eye(mx) + dt*A_c;
B_d = dt*B_c;

x0 = [pi 0 0 0 0 0]';
N  = 50; M  = N;
z  = zeros(N*mx+M*mu,1); z0 = z;

ul = [-30*pi/180; -Inf]; uu = [30*pi/180; Inf];
xl = -Inf*ones(mx,1); xu = Inf*ones(mx,1);
xl(3) = ul(1); xu(3) = uu(1);

[vlb,vub] = gen_constraints(N,M,xl,xu,ul,uu);
vlb(N*mx+M*mu)  = 0; vub(N*mx+M*mu)  = 0;

Q = diag([1 0 0 0 0 0]);
R = diag([5 5]);
I_N = eye(N); I_M = eye(M);
G = 2*blkdiag(kron(I_N, Q), kron(I_M, R));

Aeq = gen_aeq(A_d,B_d,N,mx,mu);
beq = [A_d*x0; zeros((N-1)*mx,1)];

alpha = 0.2; beta = 20; lambda_t = 2*pi/3;
param = [alpha beta lambda_t mx N];
nonlincon = @(z)gen_con(z,param);
```

```matlab
objfun = @(z) 0.5*z'*G*z;

opt = optimoptions('fmincon','Algorithm','sqp','MaxFunEvals',Inf,'MaxIter',50000);
tic
[z, fval,exitflag,output] = fmincon(objfun,z0,[],[],Aeq,beq,vlb,vub,nonlincon,opt);
toc
c = gen_con(z,param) + z(5:mx:N*mx);
c = [c; 0;];

padding_time = 8;
num_variables = padding_time/dt;
zero_padding = zeros(num_variables,1);
unit_padding  = ones(num_variables,1);
u  = [z(N*mx+1:mu:end)'; z(N*mx+2:mu:end)'];
u = [u z(N*mx + M*mu -1:N*mx + M*mu)];
u1 = u(1,:); u2 = u(2,:);

x1 = [x0(1)*unit_padding; x0(1); z(1:mx:N*mx); zero_padding];
x2 = [zero_padding; x0(2); z(2:mx:N*mx); zero_padding];
x3 = [zero_padding; x0(3); z(3:mx:N*mx); zero_padding];
x4 = [zero_padding; x0(4); z(4:mx:N*mx); zero_padding];
x5 = [zero_padding; x0(5); z(5:mx:N*mx); zero_padding];
x6 = [zero_padding; x0(6); z(6:mx:N*mx); zero_padding];
c = [zero_padding; c; zero_padding];
u1   = [zero_padding; u1'; zero_padding];
u2   = [zero_padding; u2'; zero_padding];
u    = [u1'; u2'];

t = 0:dt:dt*(length(u1)-1);

Q_lqr = diag([15 1 0.5 1 30 5]);
R_lqr = diag([1 1]);

K = dlqr(A_d,B_d,Q_lqr,R_lqr);

u_t = timeseries(u,t);
x_opt = timeseries([x1 x2 x3 x4 x5 x6]',t);
```

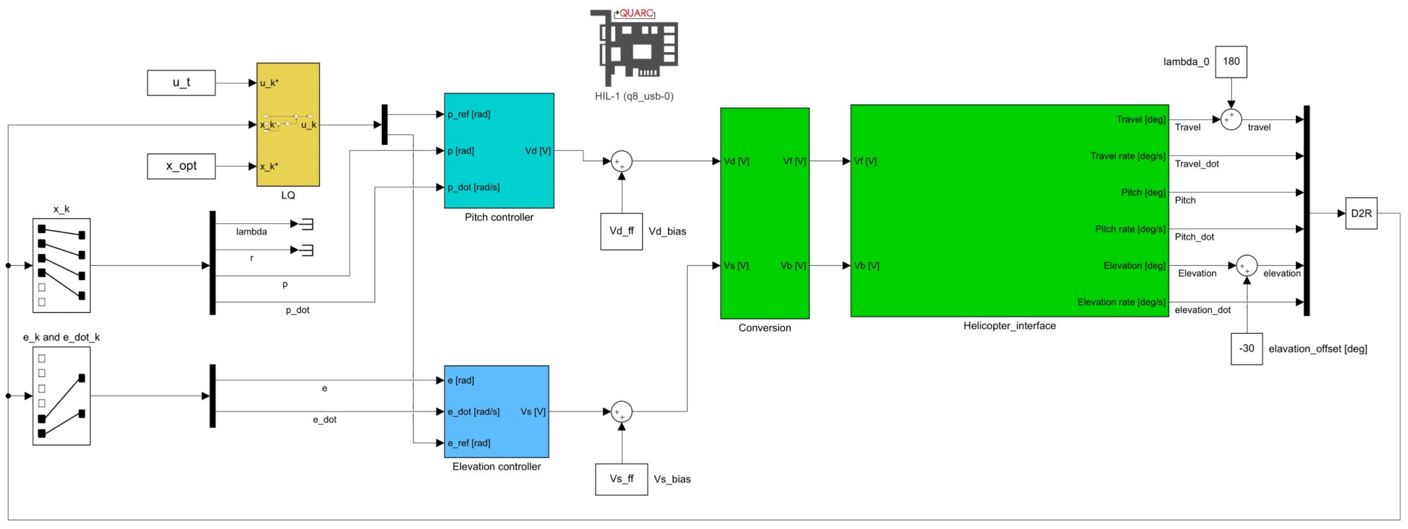

Figure 15: Day 4: Simulink Diagram

References

[1] *Helicopter Lab*. Department of Engineering Cybernetics, NTNU, Trondheim, January 2021.