TTK4135
OPTIMIZATION AND CONTROL

Helicopter Lab Report

Group 36:

528532 Khuong Huynh
528515 Tomas Nils Tellier

April 18, 2023

Department of Engineering Cybernetics

Contents

1 10.2 - Optimal Control of Pitch/Travel without Feedback

1.1 The continuous model

The system's state space form was derived from the model equations in the lab manual:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1K_{pp} \end{bmatrix} u \tag{1}$$

where

$$\boldsymbol{x} = \begin{bmatrix} \lambda & r & p & \dot{p} \end{bmatrix}^\top \quad , \quad u = p_c \tag{2}$$

The model represents the helicopter's linearized dynamics with a control input on the helicopter's pitch.

1.2 The discretized model

One can dicretize the model with the Euler method by applying the following transformation to the system matrices:

$$\boldsymbol{x_{k+1}} = \boldsymbol{x_k} + TA_c\boldsymbol{x_k} + TB_cu_k \tag{3}$$

$$\Downarrow$$

$$A_d = I + TA_c \quad , \quad B_d = TB_c \tag{4}$$

where T is the discrete system's step length. Thus, the discrete model can be written as:

$$\boldsymbol{x_{k+1}} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & -K_2T & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & -K_1K_{pp}T & 1 - K_1K_{pd}T \end{bmatrix} \boldsymbol{x_k} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1K_{pp}T \end{bmatrix} u_k \tag{5}$$

1.3 The open loop optimization problem

We have that

$$\boldsymbol{z} = [\boldsymbol{x_1}, ..., \boldsymbol{x_N}, u_0, ..., u_{N-1}]^T \quad , \quad N = 100 \tag{6}$$

As we want to solve a *quadratic* optimization problem, the objective function can be formulated as

$$\begin{aligned} \min_{z} f(z) = \boldsymbol{z}^T G \boldsymbol{z} \qquad s.t. \qquad & A\boldsymbol{z} \leq \boldsymbol{b} \\ & A_{eq}\boldsymbol{z} = b_{eq} \\ & \boldsymbol{z_{lb}} \leq \boldsymbol{z} \leq \boldsymbol{z_{ub}} \end{aligned} \tag{7}$$

The matrix G contains the states and input weights for all the states and inputs in the horizon [1, p. 35]:

$$G = \begin{bmatrix} Q_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & Q_N & \ddots & & \vdots \\ \vdots & & \ddots & R_0 & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & R_{N-1} \end{bmatrix} \tag{8}$$

where the Q's and R's are simply the weight matrices repeated for the whole horizon. Since the system has four states, one input and the time horizon is of length $N = 100$, the dimension of G is 500×500.

The cost function that we wish to minimize only includes the helicopter's *travel* (the first state) and the input. Therefore, we have that

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

Since we only have one input, the weight matrix R is a scalar.

The problem has no inequality constraints - only equality constraints and an upper/lower bound on the pitch.

The equality constraints for the whole horizon can be written as [1, p. 36]:

$$A_{eq} = \left[\begin{array}{ccccc|ccccc} I & 0 & \cdots & \cdots & 0 & -B_0 & 0 & \cdots & \cdots & 0 \\ -A_1 & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & & 0 \\ 0 & \cdots & 0 & -A_N & I & 0 & \cdots & \cdots & 0 & -B_N \end{array}\right] \tag{10}$$

and

$$b_{eq} = \begin{bmatrix} A_0 x_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{11}$$

All values in z_{lb} and z_{ub} are set to $-\infty$ and ∞, respectively, except for the values concerning the third state, *pitch*. Here, we place the given constraint

$$|p_k| \leq \frac{\pi}{6}, \quad k \in \{1, ..., N\} \tag{12}$$

We want to include this constraint, as our model for the helicopter's dynamics is not accurate for larger pitch values. More precisely, the helicopter's elevation does not take into account the pitch. For instance, if the pitch is $\pm\frac{\pi}{2}\,rad$, it is physically impossible for the helicopter to generate an upwards force in order to augment or maintain its elevation. That is why we wish to avoid too large pitch values.

It could be wise to also implement the same constraint on the pitch reference, as this may affect the helicopter's optimal trajectory. Without this constraint, the quadprog solver could plan a pitch trajectory that is too steep going towards the constraint, which may result in the real system's pitch overshooting its constraint.

1.4 The weights of the optimization problem

The assignment asks to solve the optimization problems with different values of R:

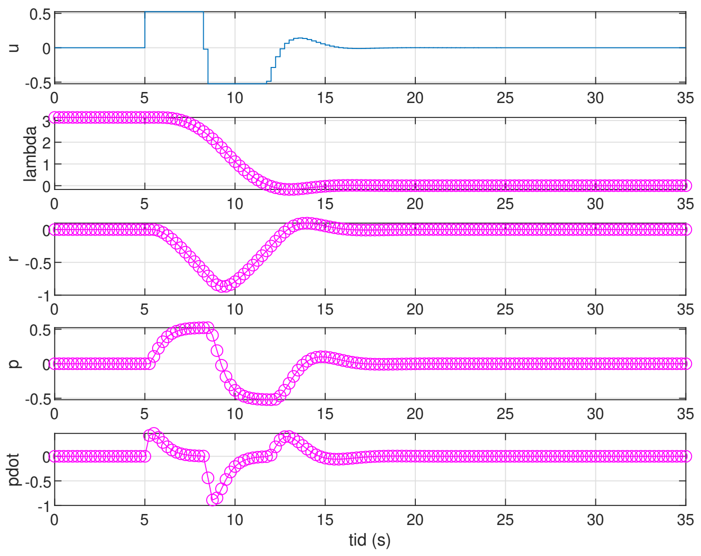

Figure 1: $R = 0.12$

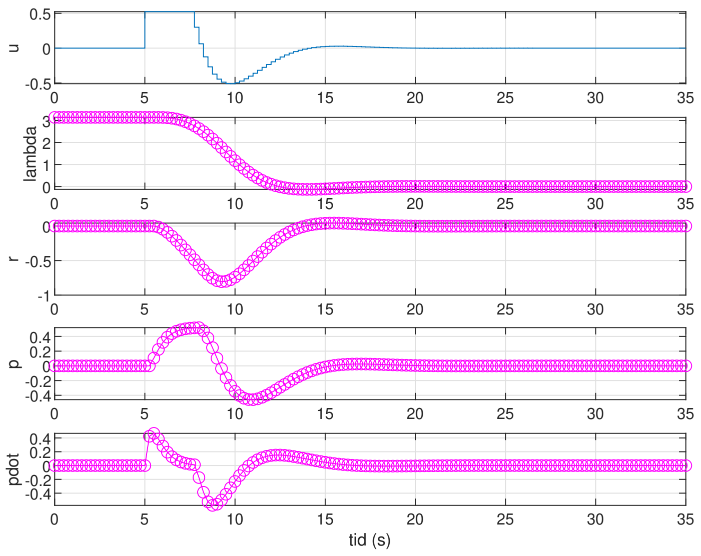

Figure 2: $R = 1.2$

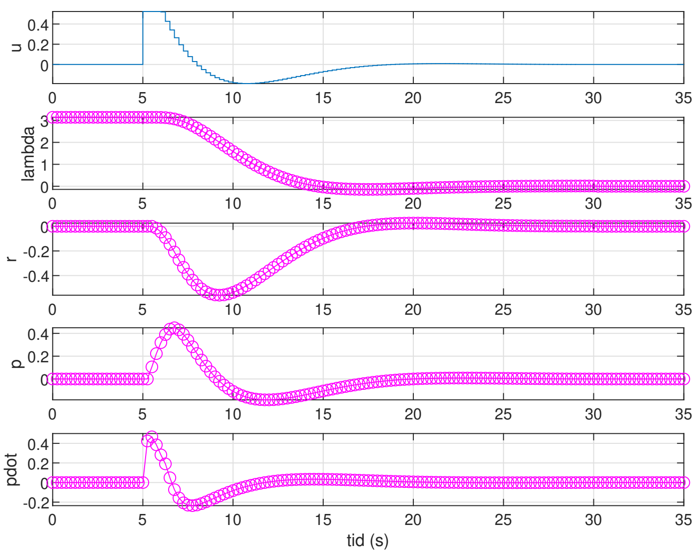

Figure 3: $R = 12$

The plots clearly show that when R is small, meaning that the use of input is *less* penalized and the state deviations are *more* penalized, the states reach their references more quickly. As R is increased and the use of input is more penalized, the state trajectories become smoother, but need more time in order to reach their references.

1.5 Experimental results

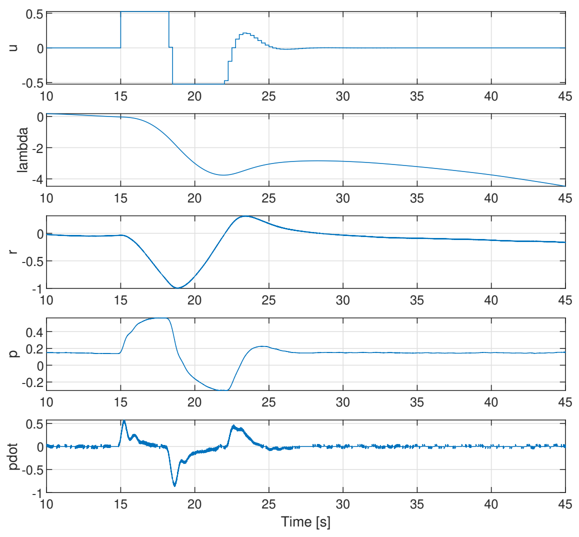

Figure 4: $R = 0.12$

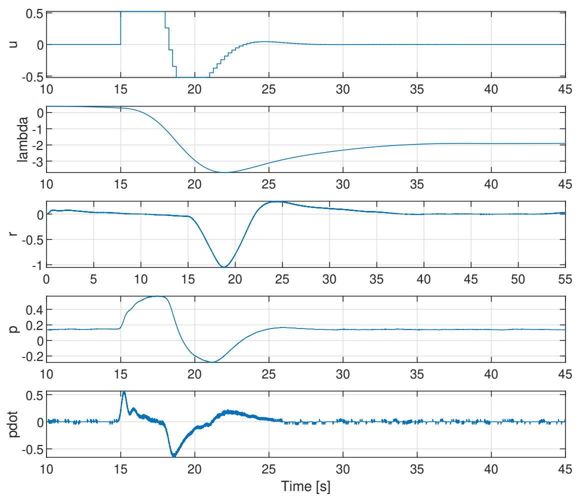

Figure 5: $R = 1.2$

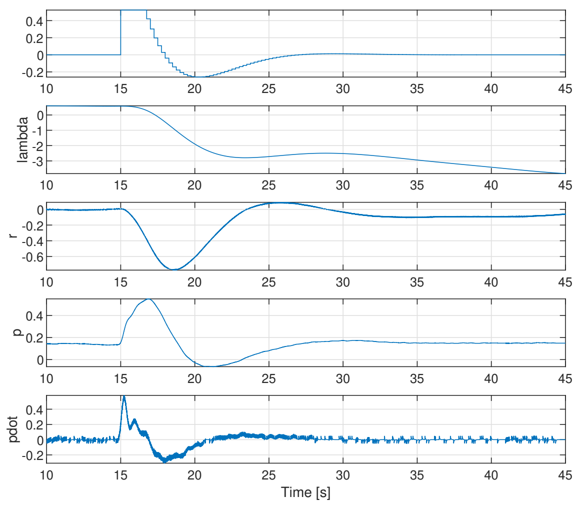

Figure 6: $R = 12$

When analyzing the plots one can clearly see that the real systems travel λ does not end in the desired point λ_f. By varying the value of R, it was observed that R = 1.2 (5) resulted in a stationary travel state. While for the other two cases, the deviation in travel continued to grow the longer the simulation went on. As there is no feedback implemented in the system yet, the travel λ would not be able to reach λ_f.

1.6 MATLAB and Simulink

```
%% Initialization and model definition
init03; % Run init

% Discrete time system model. x = [lambda r p p_dot
   ]'
delta_t = 0.25; % sampling time
A1 = [1 delta_t 0 0;
      0 1 -K_2*delta_t 0;
      0 0 1 delta_t;
      0 0 -K_1*K_pp*delta_t 1-K_1*K_pd*delta_t];
B1 = [0; 0; 0; K_1*K_pp*delta_t];
...
% Initial values
x1_0 = pi;                                      % Lambda
x2_0 = 0;                                       % r
x3_0 = 0;                                       % p
```

```
x4_0 = 0;                                      % p_dot
x0 = [x1_0 x2_0 x3_0 x4_0]';                   % Initial
   values

% Time horizon and initialization
N  = 100;                                       % Time
   horizon for states
M  = N;                                         % Time
   horizon for inputs
T = (N+40)*delta_t;                             % Simulation
    time
time_steps = [0:delta_t:T]';                    % Array for
   time steps;
z  = zeros(N*mx+M*mu,1);                        % Initialize
    z for the whole horizon
z0 = z;                                         % Initial
   value for optimization

% Bounds
ul          = -pi/6;                           % Lower bound
    on control
uu          = pi/6;                           % 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]       = gen_constraints(N,M,xl,xu,ul,uu);
   % hint: gen_constraints
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)
```

```
Q = zeros(mx,mx);
Q(1,1) = 1;                                 % Weight
   on state x1
Q(2,2) = 0;                              % Weight on
   state x2
Q(3,3) = 0;                              % Weight on
   state x3
Q(4,4) = 0;                               % Weight on
    state x4
R = 12;                                     % Weight
   on input
G = gen_q(Q,R,N,M);                       % Generate
   G, hint: gen_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 = gen_aeq(A1,B1,N,mx,mu);              % Generate
    A, hint: gen_aeq
beq = zeros(N*mx, 1);                     % Generate
   b
beq(1:4, 1) = A1*x0;

%% Solve QP problem with linear model
tic
[z,lambda] = quadprog(2*G, c, [], [], Aeq, beq, vlb,
    vub, x0); % hint: quadprog. Type 'doc quadprog'
   for more info
t1=toc;

%% 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

num_variables = 5/delta_t;
```

```
zero_padding = zeros(num_variables,1);
unit_padding  = ones(num_variables,1);

% Add zero-padding (5 sec) before and after
   trajectory
u    = [zero_padding; u; zero_padding];
x1   = [pi*unit_padding; x1; zero_padding];
x2   = [zero_padding; x2; zero_padding];
x3   = [zero_padding; x3; zero_padding];
x4   = [zero_padding; x4; zero_padding];

u_ts = timeseries(u, time_steps);
```

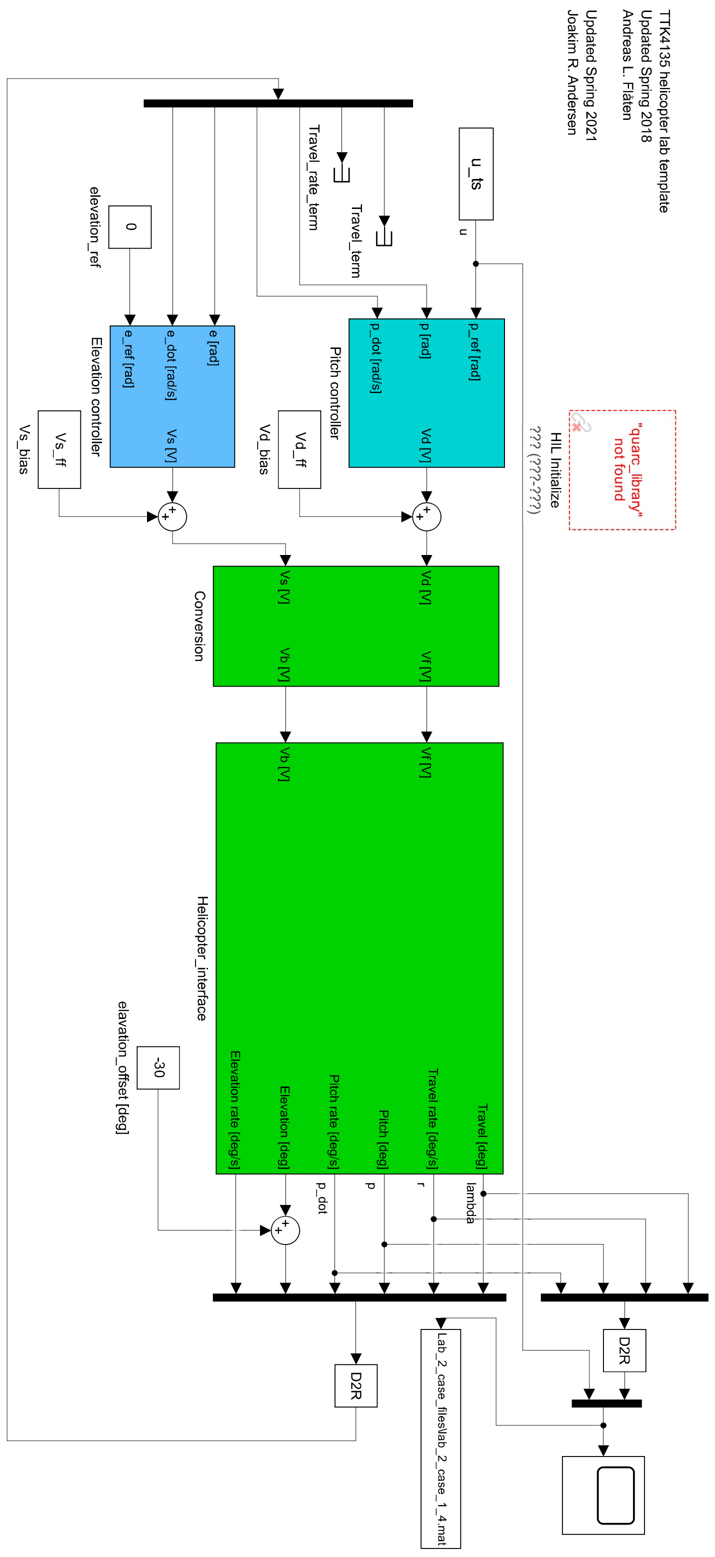

Figure 7: Simulink diagram from lab 2

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

2.1 LQ controller

The LQR is an optimal controller that minimizes a quadratic cost function over an infinitely long time horizon, consisting of a weighted sum of the states' errors and control inputs. The objective function is minimized subject to a linear model and constraints. The controller's behaviour is mainly determined by the weight matrices Q and R, whose values penalize the state deviations and the use of control inputs, respectively.

A larger weight in the Q matrix means that the corresponding state has a greater impact on the overall system's performance, and deviations in this state will be more penalized. Similarly, a larger weight in the R matrix indicates that the corresponding input has a greater impact on the system, or more precisely, on the cost function, meaning that the controller will tolerate less use of said input.

Similarly to the parameters in a PID controller, choosing these weights is a matter of tuning. However, there are some rules of thumb to help us make rational guesses.

Firstly, it is common to use *diagonal* matrices for Q and R. This makes the tuning easier, as there is only one weight associated to a given state or input. Secondly, one can apply *Bryson's rule* as a good starting point for the weights:

$$Q_{ii} = \frac{1}{maximum\ acceptable\ value\ of\ x_i^2} \tag{13}$$

$$R_{jj} = \frac{1}{maximum\ acceptable\ value\ of\ u_j^2} \tag{14}$$

2.2 Model Predictive Control

To implement the MPC scheme (A) instead of an LQ controller, the optimal output trajectory u for the whole horizon is calculated at each timestep, and only its first element is used. This can be implemented by adding a script in Simulink that takes in the measurements to predict x^*_{new} and u^*_{new} with the use of quadprog.

The advantages of using such an MPC controller are as follows:

- Handles nonlinear systems: MPC can handle nonlinear systems, while an LQ controller is limited to linear systems
- Handles constraints: An LQ controller will assume that *unlimited* output is available to reach the desired state, while an MPC can handle constraints on input and states.
- Able to adapt: MPC can adapt to changing conditions by recalculating the optimized trajectory at each timestep, while an LQ controller will assume that the system is time invariant.

The disadvantages are:

- Computationally complexity: MPC requires that the optimization problem to be solved at each time step, which can be computanionally expensive for large systems.
- Model accuracy: MPC depends on an accurate model of the system, which can be difficult and time consuming. If the prediction model is inaccurate, the MPC may perform poorly.

Figure 8 shows how the system's block diagram would look like with the implementation of an MPC.

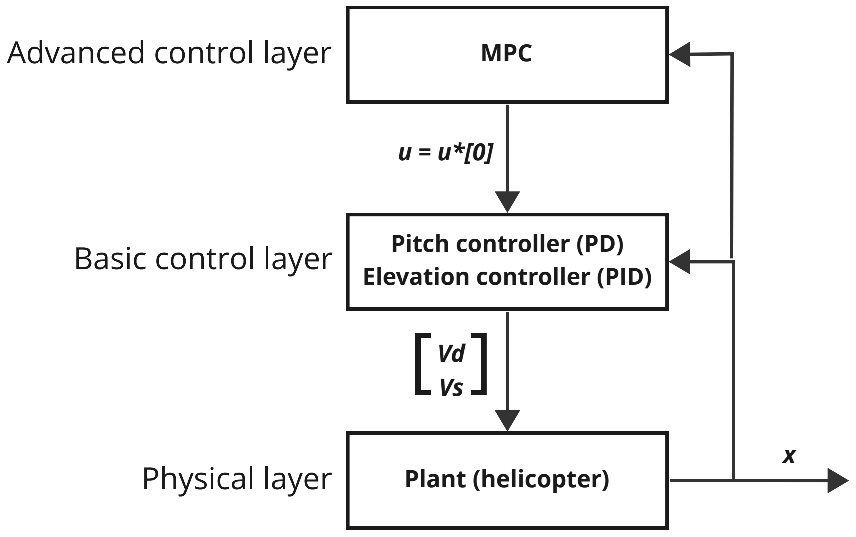

Figure 8: Block diagram with MPC

2.3 Experimental results

In this lab these values were tested:

Case 1:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad , \qquad R = 1 \tag{15}$$

Case 2:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad , \qquad R = 4 \tag{16}$$

Case 3:

$$Q = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \quad , \qquad R = 1 \tag{17}$$

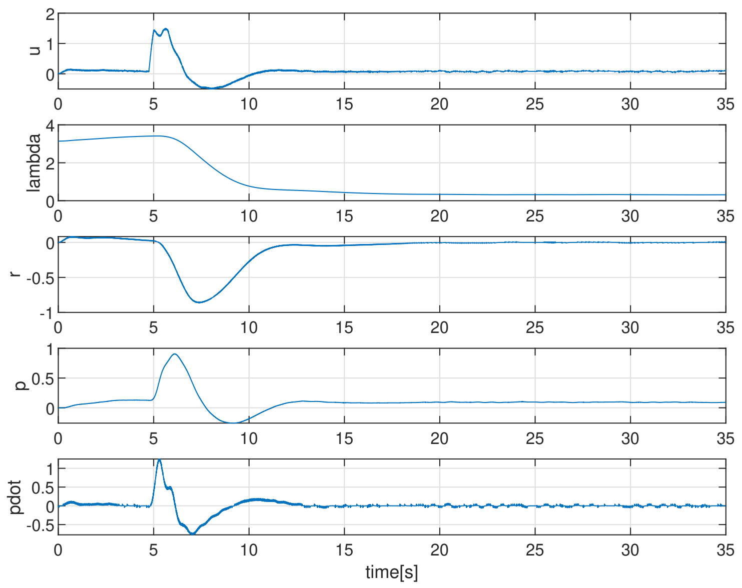

Figure 9: Case 1: $Q = I, R = 1$

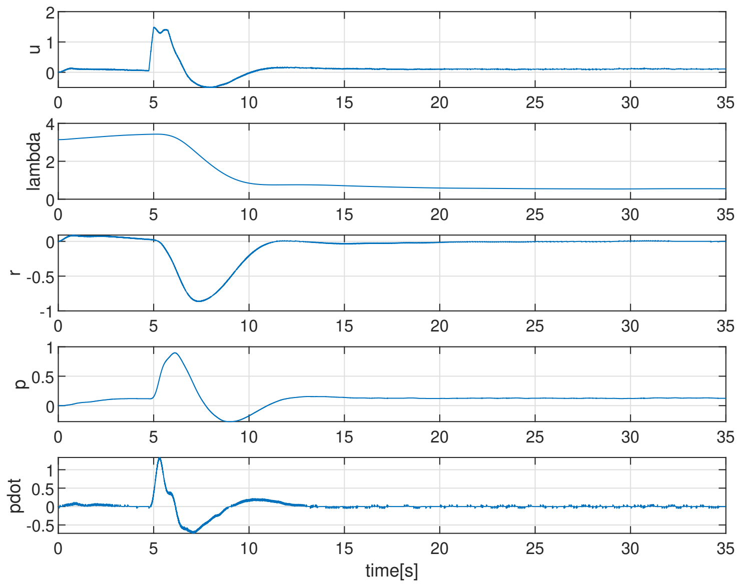

Figure 10: Case 2: $Q = I, R = 4$

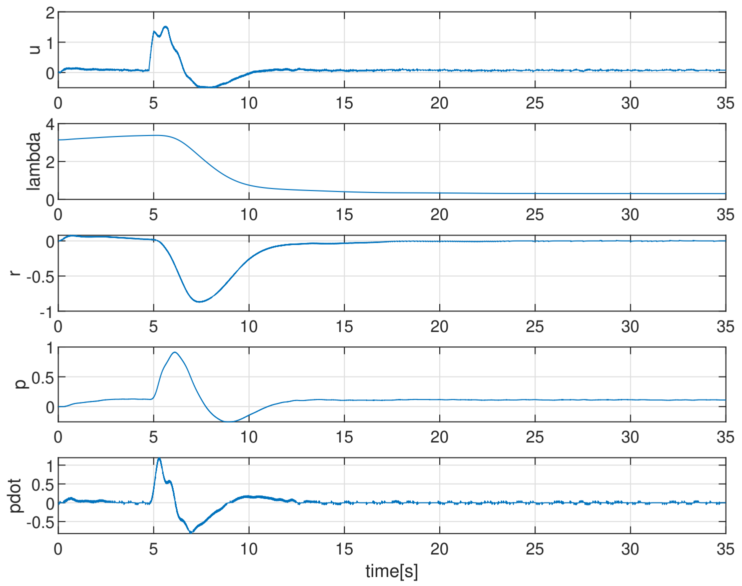

Figure 11: Case 3: $Q = 4I, R = 1$

When analyzing the plots one can see that the travel (λ) stabilizes near the desired value (λ_c) compared to the previous task. This is the result of implementing a feedback controller. As a deviation occurs in relation to the calculated state, the feedback controller will take this into consideration and adjust its output accordingly.

During the testing of the helicopter with different values for Q and R in the LQ controller, small differences in the stationary deviation of λ were observed. Specifically, in 10, the stationary deviation in λ was found to be larger than in 11. This result was expected since the states are more penalized in the latter test. The choices of Q and R used in the LQ controller were deliberately selected to emphasize the observed difference.

2.4 MATLAB and Simulink

The following code only includes changes made from lab 2.

```
%% Calculate LQR gain matrix
Q_lqr = [1000 0 0 0;
          0 1 0 0;
          0 0 1 0;
          0 0 0 1;];
R_lqr = 1;

K = dlqr(A1,B1, Q_lqr, R_lqr) % Feedback gain

...

x_ts = timeseries([x1 x2 x3 x4], time_steps); % Send
    optimal trajectory to Simulink
```

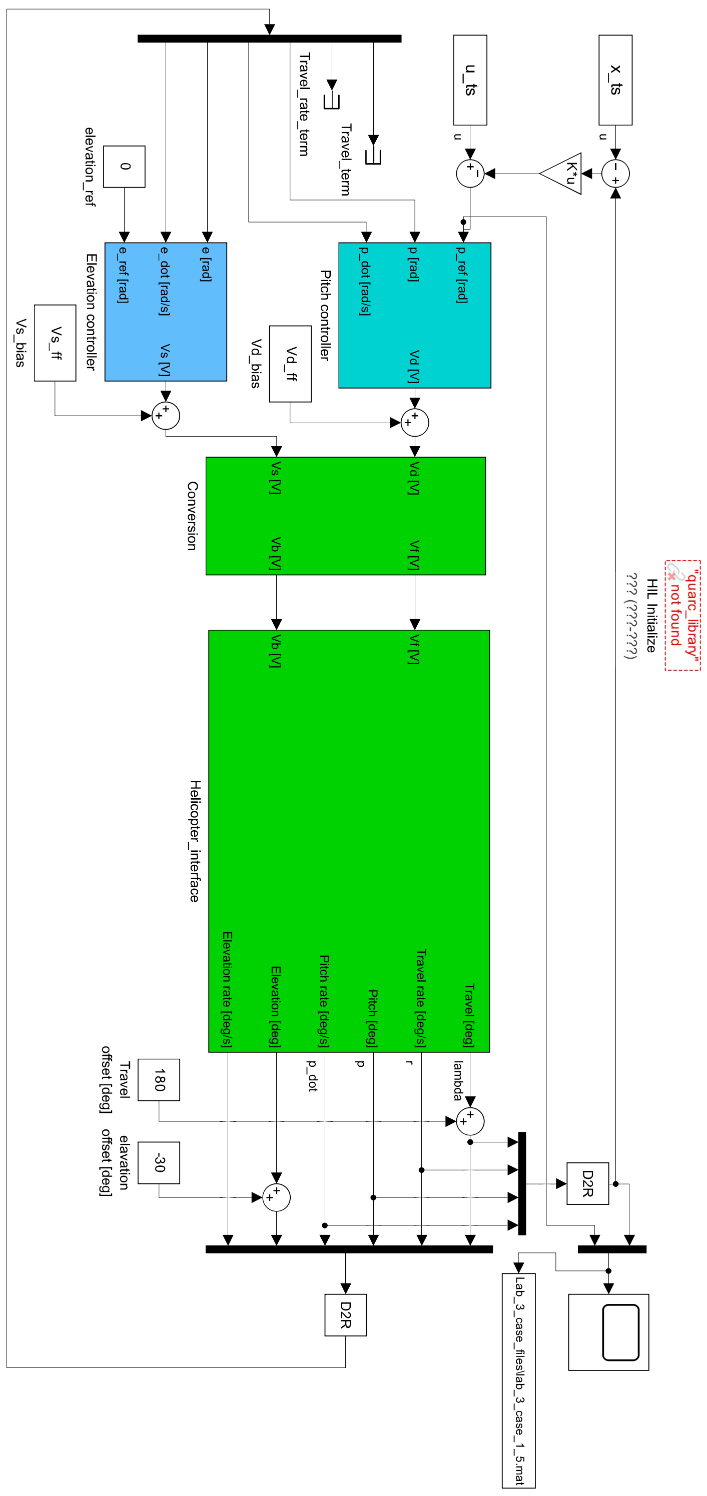

Figure 12: Simulink diagram from lab 3

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

3.1 The continuous model

Given

$$\boldsymbol{x} = \begin{bmatrix}\lambda & r & p & \dot{p} & e & \dot{e}\end{bmatrix}^\top \quad , \quad \boldsymbol{u} = \begin{bmatrix}p_c & e_c\end{bmatrix}^\top \quad , \tag{18}$$

the system can be written on continuous state space form as:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3K_{ep} & -K_3K_{ed} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3K_{ep} \end{bmatrix} \boldsymbol{u} \tag{19}$$

3.2 The discretized model

Again, one can apply the transformations given by equations 3 and 4 to discretize the system. This yields the following system:

$$\boldsymbol{x_{k+1}} = A_d \boldsymbol{x_k} + B_d \boldsymbol{u_k} \tag{20}$$

where

$$A_d = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & -K_2T & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & -K_1K_{pp}T & 1-K_1K_{pd}T & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & -K_3K_{ep}T & 1-K_3K_{ed}T \end{bmatrix} \tag{21}$$

and

$$B_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1 K_{pp} T & 0 \\ 0 & 0 \\ 0 & K_3 K_{ep} T \end{bmatrix} \tag{22}$$

3.3 Experimental results

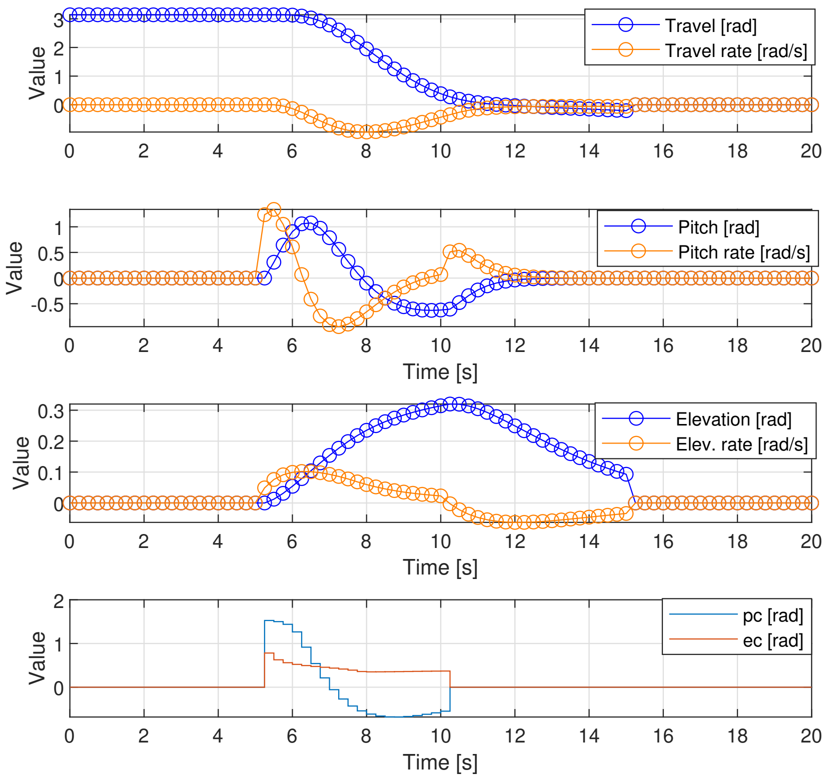

Figure 13: Optimal trajectory

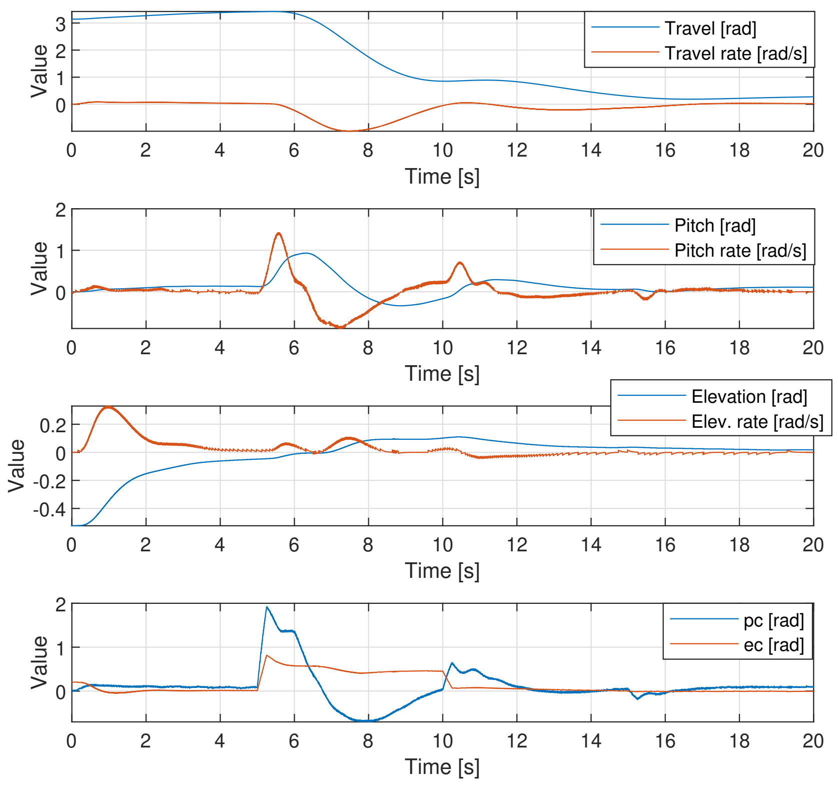

Figure 14: Real system's trajectory

When analyzing the plots for the calculated optimal trajectory and the real systems trajectory one can clearly see that the helicopter follows the trajectory. There are some deviatons in the elevation and the output for the pitch controller p_c. The deviation in elevation may be caused by the use of a linearized model of the system, while the output is most likely caused by the feedback controller. As there occurs any deviations from the optimal trajectory, the LQ controller will try to counteract that.

One can also notice that there is a stationary deviation in the travel (λ) due to the absence of an integrator in the system. The implementation of an integrator could address this deviation.

3.4 Decoupled model

The two reasons for why the helicopter still works are:

- The **feedback** loop stabilizes the helicopter
- The pitch **constraints** does not allow for the pitch to be greater than $\pm 30°$

Suggestion A:

The change of $\ddot{e}$ would allow the system to take the pitch into account when controlling the elevation. However, the system would become nonlinear, and thus would become more complex. Since the controller works fine by only applying constraints to the pitch, suggestion A is redundant and a bad suggestion.

Suggestion B:

Again, applying this change would result in a nonlinear objective function, which would require more sophisticated optimization techniques. Since this would require the controller to be unnecessarily complex, suggestion B is a bad suggestion.

Suggestion C:

Adjusting the weights in the Q-matrix every few time steps would be a considerable computational cost, but might not give a system as complex as introducing nonlinearities. Therefore, this is a good suggestion.

It is important to note that there are no right or wrong answers to these suggestions being good or bad, as it depends on one's priorities. When rating the suggestions, we decided to consider simplicity without being concerned with potential increase in computational requirements.

3.5 MATLAB and Simulink

```
% Discrete time system model. x = [lambda r p p_dot
   e e_dot]'
dt       = 0.25; % sampling time
A1 = [1 dt 0 0 0 0;
      0 1 (-K_2*dt) 0 0 0;
      0 0 1 dt 0 0;
      0 0 (-K_1*K_pp*dt) (1-K_1*K_pd*dt) 0 0;
      0 0 0 0 1 dt;
      0 0 0 0 (-K_3*K_ep*dt) (1-K_3*K_ed*dt)];

B1 = [0 0; 0 0; 0 0; K_1*K_pp*dt 0; 0 0; 0 K_3*K_ep*
   dt];

...

% Initial values
x1_0 = pi;                                          % Lambda
x2_0 = 0;                                           % r
x3_0 = 0;                                           % p
x4_0 = 0;                                           % p_dot
x5_0 = 0;                                       % e
x6_0 = 0;                                           %e_dot
x0 = [x1_0 x2_0 x3_0 x4_0 x5_0 x6_0]';        %
   Initial values

% Time horizon and initialization
N   = 40;                                      % Time
   horizon for states
M   = N;                                        % Time
   horizon for inputs
T = (N+40)*dt;                             % Simulation time
time_steps = [0:dt:T]';                    % Array for time
   steps;
z   = zeros(N*mx+M*mu,1);                       % Initialize
    z for the whole horizon
z0 = z;                                         % Initial
   value for optimization

...

% Nonlinear constraints
nonlcon = @elev_con;
```

```
% Generate constraints on measurements and inputs
[vlb,vub]       = gen_constraints(N,M,xl,xu,ul,uu);
   % hint: gen_constraints
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
Q1 = zeros(mx,mx);
Q1(1,1) = 1;                    % Weight on state x1
R = [1 0;
     0 1];                 % Weight on input
G = gen_q(Q1,R,N,M);       % Generate Q, hint: gen_q
c = zeros(N*mx+M*mu,1);    % Generate c, the linear
   constant term in the QP

%% Calculate LQR gain matrix
Q_lqr = [1 0 0 0 0 0;
         0 1 0 0 0 0;
         0 0 1 0 0 0;
         0 0 0 1 0 0;
         0 0 0 0 1 0;
         0 0 0 0 0 1];

R_lqr = [1 0;
         0 1];

K = dlqr(A1,B1, Q_lqr, R_lqr)

...

%% Solve QP problem with a nonlinear constraint
f = @(x) 1/2*x'*G*x;                    % Objective
    function?
tic
z = fmincon(f, z0, [], [], Aeq, beq, vlb, vub,
   nonlcon);
t1=toc;

...

ufake  = [0; 0; z(N*mx+1:N*mx+M*mu)]; % Control
```

```
   input from solution
u = zeros(N+1,2);
for i = 1:size(u,1)
    u(i,1) = ufake(2*i-1);
    u(i,2) = ufake(2*i);
end

...

u_ts = timeseries([u1 u2], time_steps);
x_ts = timeseries([x1 x2 x3 x4 x5 x6], time_steps);
```

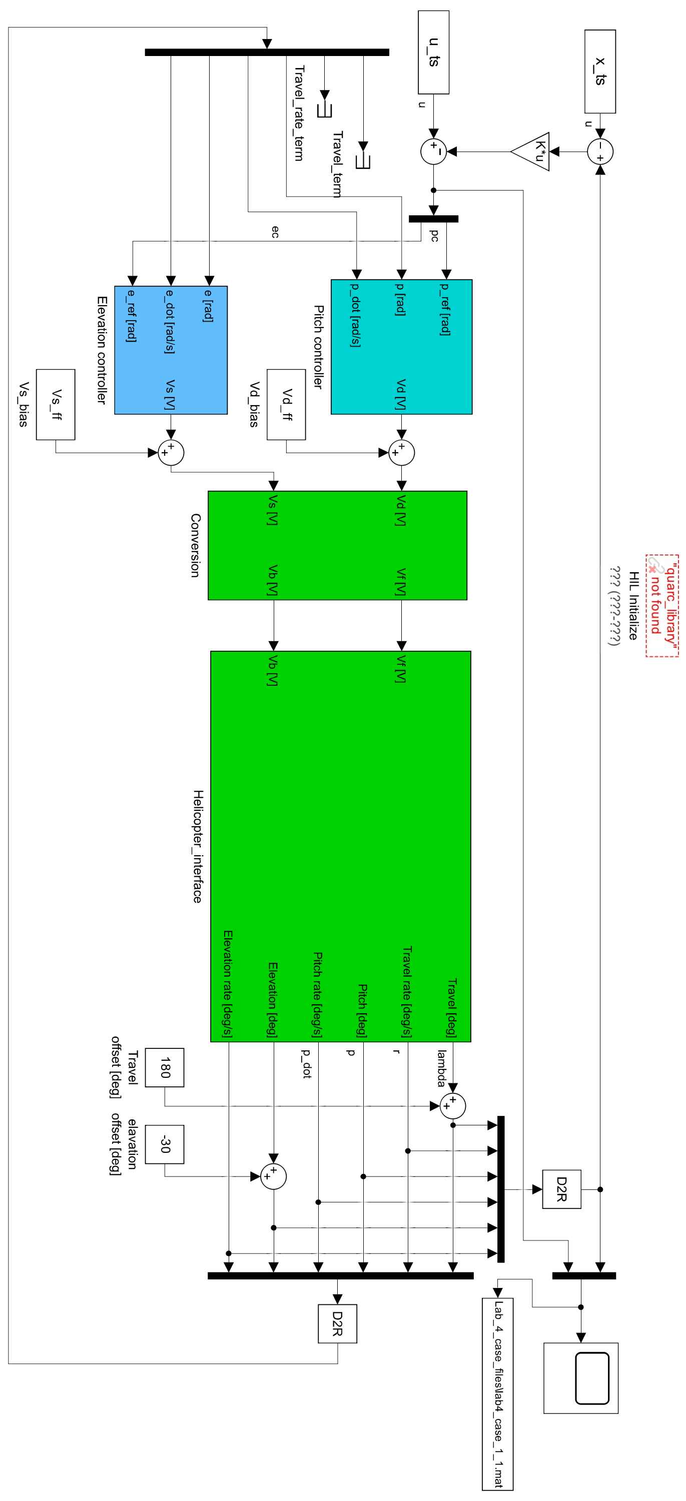

Figure 15: Simulink diagram from lab 4

References

[1] Bjarne Foss and Tor Aksel N. Heirung. *Merging optimization and control*. Department of Engineering Cybernetics - NTNU, 2016.