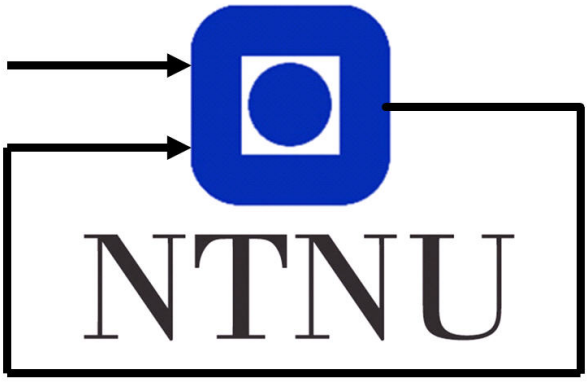
OptReg lab report

Group ??
Vemund Rogne
Jørgen Haaland

April 5, 2021



Department of Engineering Cybernetics

Contents

1 Optimal Control of Pitch/Travel without Feedback

1.1 Derivation of a continous time state space model

In this part of the exercise we will disregard elevation, therefore we assume $e = 0$ and do include it in the model.

The state-vector, $\boldsymbol{x}$ is defined as:

$$\boldsymbol{x} = \begin{bmatrix} \lambda & r & p & \dot{p} \end{bmatrix}^T, \tag{1.1}$$

where λ is travel, r is the travel rate, p is pitch and $\dot{p}$ is pitch rate.

The dynamic equations for the system was given in the problem description . These following equations were given:

Add ref to page and paper

$$\dot{\lambda} = r \tag{1.2a}$$

$$\dot{r} = -K_2 p, \quad K_2 = \frac{K_p l_a}{J_t} \tag{1.2b}$$

$$\dot{p} = \dot{p} \tag{1.2c}$$

$$\ddot{p} = -K_1 V_d = K_1 K_{pd}\dot{p} - K_1 K_{pp} p + K_1 K_{pp} pc, \quad K_1 = \frac{K_f l_h}{J_p} \tag{1.2d}$$

The state-space form of the system therefore becomes:

$$\underbrace{\begin{bmatrix} \dot{\lambda} \\ \dot{r} \\ \dot{p} \\ \ddot{p} \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} \end{bmatrix}}_{\boldsymbol{A}_c} \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1 K_{pp} \end{bmatrix}}_{\boldsymbol{B}_c} \underbrace{p_c}_{u} \tag{1.3}$$

1.1.1 A deeper dive into the state-space model

The state-space form of the system models two part of the whole system, namely:

1. The physics of the helicopter.
2. The proportional-derivative controller for the pitch.

This is shown in fig. 1 where the red dotted box shows what we model with the state space model.

Add ref to the figure in the problem description

This becomes clear studying the equations in eq. (1.2). They describe the helicopters physics for all states except elevation and elevation rate. λ and r is dependent on the helicopters pitch, p. p and $\dot{p}$ is however dependent on the voltage difference, V_d. The voltage difference is the output of the PD-controller for controlling the pitch.

Should we add acronym list?

To summarize, this means that our state space model describes the helicopter's physics through the dynamic equations for λ, r, p and $\dot{p}$, while the equation of V_d describes the PD-controller used to control the pitch angle. In total our state-space model is modelling both the helicopter and the PD controller for pitch.

1.1.2 Stability and eigenvalues

The properties of this system is dependent on physical constants $(l_a, J_t, ...)$ and control parameters (K_{pp}, K_{pd}).

Symbolic expressions in Matlab shows that the eigenvalues of A are:

$$\lambda = \pm\frac{1}{2}\left(\sqrt{-K_1(-K_1 K_{pd}^2 + 4K_{pp})} - K_1 K_{pd}\right) \tag{1.4}$$

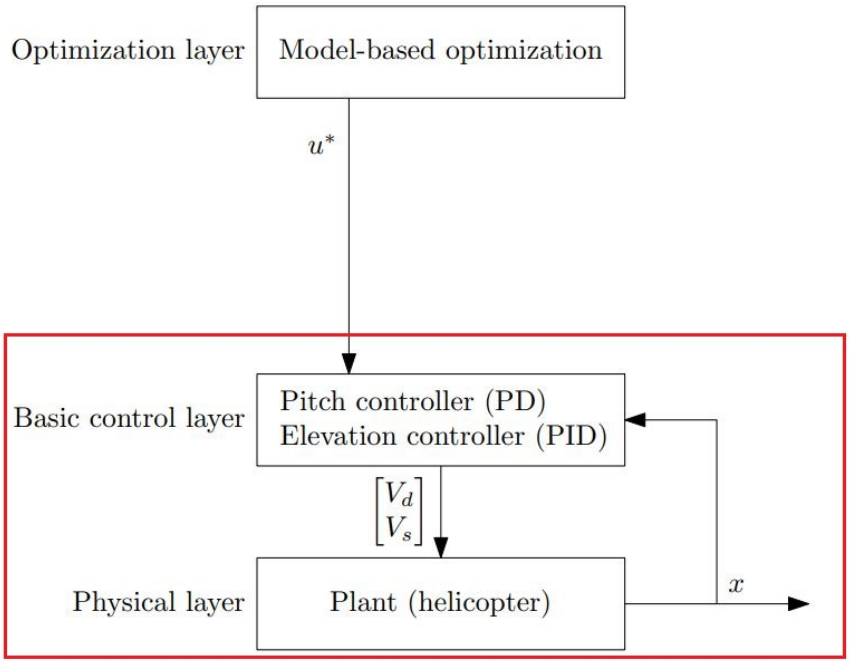


Figure 1: The red box encapsulate what is modelled in the state space model described by eq. (1.3).

The eigenvalues of the continous model, with $K_{pp} = 0,1, K_{pd} = 0,4$ are:

$$\begin{bmatrix} 0 \\ 0 \\ -0.26 + 0.24i \\ -0.26 - 0.24 \end{bmatrix} \tag{1.5}$$

1.2 Discretizing the continous time model

A discretized model is required for generating an optimal trajectory. *[...] continous time models require quite different solution methods* [1]

We will discretize the model using the forward Euler method, which is given by:

Do we need to derive forward euler?

$$\boldsymbol{x}[k+1] = \boldsymbol{I}\boldsymbol{x}[k] + T\boldsymbol{A_c}\boldsymbol{x}[k] + T\boldsymbol{B}_c, \tag{1.6}$$

where T is the sample-time in the discrete model.

Add reference to linsys slides

Reformulating this, we can write:

$$\boldsymbol{x}_{k+1} = \underbrace{(\boldsymbol{I} + T\boldsymbol{A}_c)}_{\boldsymbol{A}_d}\boldsymbol{x}_k + \underbrace{T\boldsymbol{B}_c}_{\boldsymbol{B}_d}u_k \tag{1.7}$$

On matrix form $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ becomes:

$$\boldsymbol{A}_d = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & -TK_2 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & -TK_1K_{pp} & 1 - TK_1K_{pd} \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ TK_1K_{pp} \end{bmatrix} \tag{1.8}$$

1.2.1 Checking stability

The stability condition for eq. (1.7) is that all eigenvalues of $\boldsymbol{A}_d$ is less than one in absolute values, i.e.:

$$|\lambda_i| \leq 1, \quad \text{for } i = 1, 2, 3, 4 \tag{1.9}$$

, where λ_i is the i'th eigenvalue of $\boldsymbol{A}_d$.

Where is this equation from? I believe that it works, but we should either derive it or reference where we found it. ANSWER: From linsys slides, see above.

Using the constant values given in the MATLAB we found the eigenvalues of $\boldsymbol{A}$ to be ...

Add Matlab appendix

find eigenvalues

1.3 The open loop optimization problem

How is it formulated?

The open loop optimization problem finds an optimal trajectory of the helicopter with the cost function

$$\phi = \sum_{i=0}^{N-1} \left(\lambda_{i+1} - \lambda_f\right)^2 + q p_{ci}^2, \quad q \geq 0 \tag{1.10}$$

where q is the weight of input-usage. Subject to constraints.

1.3.1 Formulating the cost function

How to you formulate a difference? Im trying to understand that before I write this section

1.3.2 The constraints of the optimization problem

There are two separate types of constrains in this problem, the system itself and imposed constraints. The system constrains is the physics of the helicopter, while the imposed constraints are for instance a constraint on the pith-reference:

$$|p_k| \leq \frac{30}{180}\pi, k \in \{1, ..., N\} \tag{1.11}$$

The physics of the helicopter is added in A_{eq} and b_{eq}.

$$A_{eq} = \begin{bmatrix} I & 0 & \cdots & \cdots & 0 & -B & 0 & \cdots & \cdots & 0 \\ -A & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -A & I & 0 & \cdots & \cdots & 0 & -B \end{bmatrix}, \; b_{eq} = \begin{bmatrix} Ax_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{1.12}$$

Performing the multiplication (and abusing notation) shows that:

$$A_{eq} z = b_{eq} \implies \begin{bmatrix} x_1 - Bu_0 = Ax_0 \\ -Ax_1 + x_2 - Bu_1 = 0 \\ \vdots \end{bmatrix} \implies \begin{bmatrix} x_1 = Ax_0 + Bu_0 \\ x_2 = Ax_1 + Bu_1 \\ \vdots \end{bmatrix} \tag{1.13}$$

Add the steps in our formulation. How we get to the quadprog-formulation:

- How the model is formulated
- How we formulate it as a QP problem with z
- What out constraints are (equality $A_{eq} z = B_{eq}$, inequality: $u_{low} < ...$)

1.4 The weights of the optimization problem

Try using the values 0.1, 1 and 10 as weights q. Plot the manipulated variable and the output. Comment the results with respect to the different weights chosen.

Weighing the input higher by increasing the value of q means that we are placing a higher cost of input - reducing the input usage. This will in turn mean that the cost of deviation in λ is in relation to the input, cheaper. The result is lower input usage and a slower response. This is exactly what is seen in fig. 2.

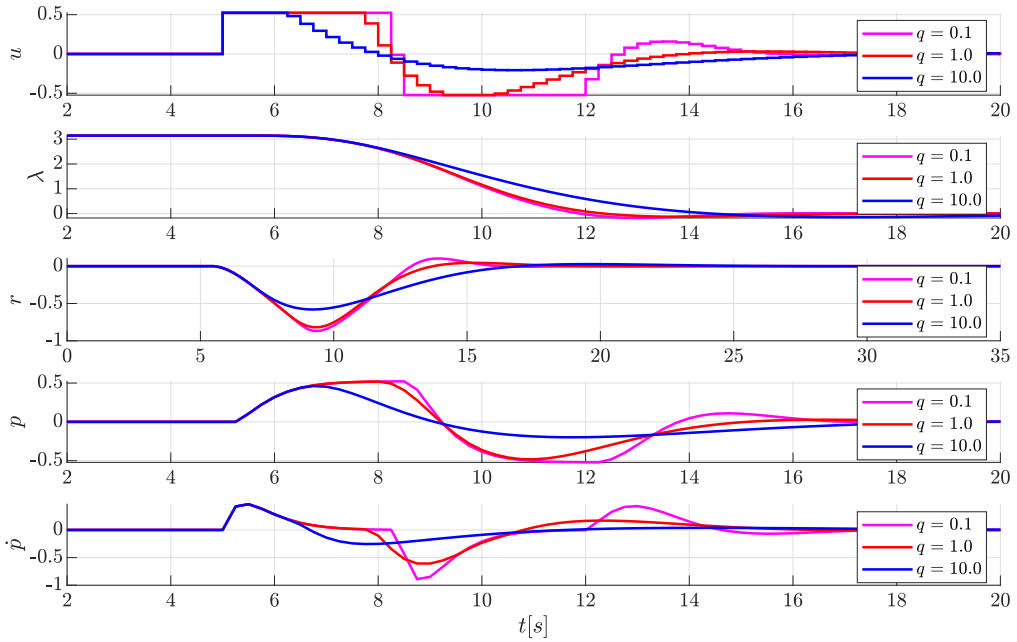


Figure 2: Manipulated variable and outputs with different values of q.

1.5 The objective function

Furthermore, discuss the objective function (15) (in the lab assignment text) in particular the term $(\lambda_i - \lambda_f)^2$. For instance, could any unwanted effects arise from steering the helicopter to $\lambda = \lambda_f$ with this objective function?

From the forum: Here are some questions that will help you figure out what is asked for: What will happen when you have some elements that are very large compared to others in a quadratic objective function? Which consequence(s) does this have for the setup in the lab? What is the impact of the length of the control horizon (imagine that the step length, h, is fixed)?

1.6 Experimental results

Printouts of data from relevant experiments (plots). Discussion and analysis of the results. Answer 10.2.2.7 here.

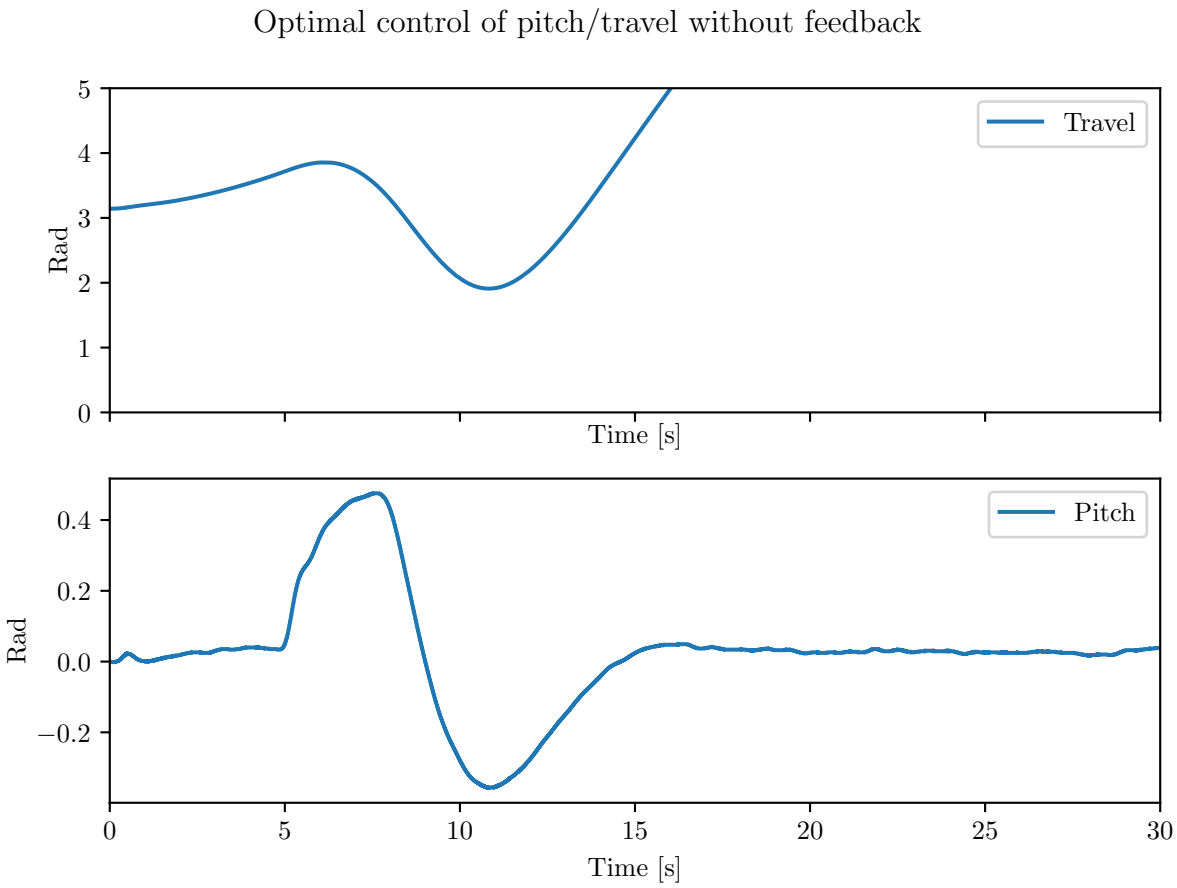


Figure 3: Results of LAB2

As fig. 3 clearly shows the control sequence did not yield the desired travel-response. This is expected, as there is no feedback - any small variations will move the response from the planned one. In this case the variations are quite large, particularly the offset from what the encoder measures as 0 pitch as opposed to what the real-world 0 pitch is. This difference makes the helicopter travel while it keeps 0-pitch.

The response shows a major travel-offset from the generated sequence, this is primarily because of the offset described above. The pitch-response is however quite close to the optimal trajectory - this is expected as there is not the same level of offset as in travel.

1.7 MATLAB and Simulink

Code and diagrams go here

1.7.1 MATLAB

Listing 1: MATALB code for lab 2

```
% TTK4135 - Helicopter lab
% Hints/template for problem 2.
% Updated spring 2018, Andreas L. Flten

%% Initialization and model definition
init05; % Change this to the init file corresponding to your helicopter
```

```
7
8  % Continous time model
9  Ac = [0, 1,        0,        0;
10       0, 0,     -K_2,        0;
11       0, 0,        0,        1;
12       0, 0, -K_1*K_pp, -K_1*K_pd];
13
14 Bc = [     0;
15            0;
16            0;
17       K_1*K_pp];
18
19 % Discrete time system model. x = [lambda r p p_dot]'
20 delta_t = 0.25; % sampling time
21 A1 = eye(4) + (delta_t.*Ac);
22 B1 = (delta_t.*Bc);
23
24 % Number of states and inputs
25 mx = size(A1,2); % Number of states (number of columns in A)
26 mu = size(B1,2); % Number of inputs(number of columns in B)
27
28 % Trajectory start and end values
29 lambda_0 = pi;
30 lambda_f = 0;
31 % Initial values
32 x1_0 = lambda_0;                          % Lambda
33 x2_0 = 0;                                 % r
34 x3_0 = 0;                                 % p
35 x4_0 = 0;                                 % p_dot
36 x0 = [x1_0 x2_0 x3_0 x4_0]';              % Initial values
37
38 % Time horizon and initialization
39 N  = 100;                                 % Time horizon for states
40 M  = N;                                   % Time horizon for inputs
41 z  = zeros(N*mx+M*mu,1);                  % Initialize z for the whole
       horizon
42 z0 = z;                                  % Initial value for
       optimization
43
44 % Bounds
45 ul          = -30*pi/180;                    % Lower bound on control
46 uu          = -ul;                           % Upper bound on control
47
48 xl      = -Inf*ones(mx,1);                % Lower bound on states (no
       bound)
49 xu      = Inf*ones(mx,1);                 % Upper bound on states (no
       bound)
50 xl(3)   = ul;                             % Lower bound on state x3
51 xu(3)   = uu;                             % Upper bound on state x3
52
53 % Generate constraints on measurements and inputs
54 [vlb,vub]       = gen_constraints(N,M,xl,xu,ul,uu); % hint:
       gen_constraints
55 vlb(N*mx+M*mu)  = 0;                      % We want the last input to be
       zero
56 vub(N*mx+M*mu)  = 0;                      % We want the last input to be
       zero
57
```

```
% Generate the matrix Q and the vector c (objecitve function weights in
    the QP problem)
Q1 = zeros(mx,mx);
Q1(1,1) = 1;                                % 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 = gen_q(Q1,P1,N,M);                      % Generate Q, hint: gen_q
% The constant linear term is zero in our case
c = zeros(size(Q, 2), 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(size(Aeq, 1), mu);                 % Generate b
A0x0 = A1*x0;
beq(1:size(A0x0,1), :) = A0x0;

%% Solve QP problem with linear model
tic
% x = quadprog(H,f,A,b,Aeq,beq,lb,ub);
[z,lambda] = quadprog(Q, c,[],[], Aeq, beq, vlb, vub);% hint: quadprog.
    Type 'doc quadprog' for more info
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

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

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];

%% Plotting
t = 0:delta_t:delta_t*(length(u)-1);

figure(2)
subplot(511)
stairs(t,u),grid
```

```
ylabel('u')
subplot(512)
plot(t,x1,'m',t,x1,'mo'),grid
ylabel('lambda')
subplot(513)
plot(t,x2,'m',t,x2','mo'),grid
ylabel('r')
subplot(514)
plot(t,x3,'m',t,x3,'mo'),grid
ylabel('p')
subplot(515)
plot(t,x4,'m',t,x4','mo'),grid
xlabel('tid (s)'),ylabel('pdot')
```

1.7.2 Simulink

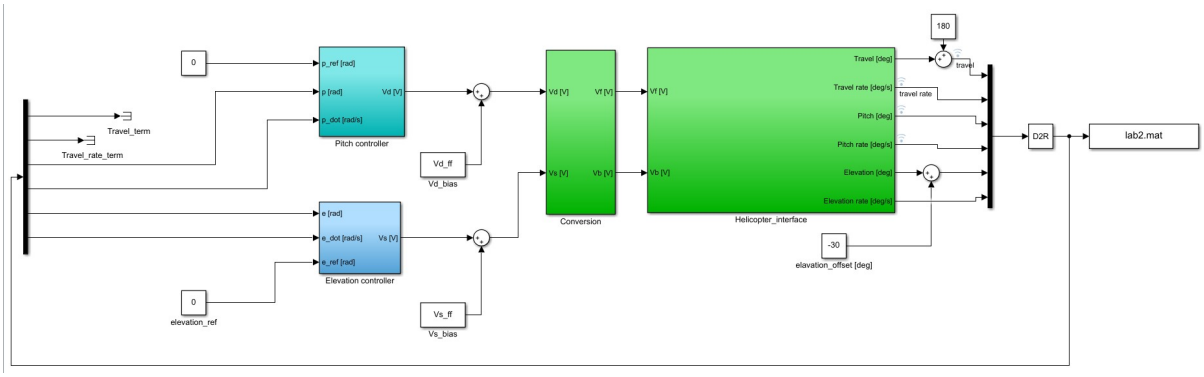


Figure 4: Simulink diagram used in lab 2.

Need a much better image of this.

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

In this task we add feedback to the optimal controller that we developed in section 1. Feedback is clearly needed as the experimental results in section 1.6 show that the helicopter response quickly deviates from the optimal trajectory. The results of this laboratory exercise shows that adding feedback greatly improves the perfomance of the helicopter.

2.1 Introducing feedback

There are many ways of adding feedback to a system. In this case the assignment states that we are to add feedback between the calculated optimal trajectory and the actual trajectory. In other words the feedback introduced here modifies the input u to get closer to the optimal trajectory x^*.

The pitch-control input is now governed by the equation:

$$u_k = u_k^* - \boldsymbol{K}^T(\boldsymbol{x}_k - \boldsymbol{x}_k^*) \tag{2.1}$$

where u_k^* and $\boldsymbol{x}_k^*$ are the optimal input and state trajectories predicted in the optimization layer.

This feedback equation modifies the input u_k to the pitch-controller if the state x_k deviates from the optimal value x_k^*. The feedback gain K is calculated using an LQ controller.

2.2 LQ controller

As explained in the problem description, an LQ (or linear-quadratic) controller, solves the quadratic objective function given by cref

$$J = \sum_{i=0}^{\infty} \Delta x_{i+1}^\top Q \Delta x_{i+1} + \Delta u_i^\top R \Delta u_i, \quad Q \geq 0, \quad R > 0 \tag{2.2}$$

for a linear model

$$\Delta x = A\Delta x_i + B\Delta u_i \tag{2.3}$$

without including inequality constraints. Here $\Delta x = x - x^*$ and $\Delta u = u - u^*$ are deviations from the optimal trajectory.

The matrix Q and the scalar R are the weights of the optimalization problem. Q determines how much state-deviations should be penalized, while R determines how much input-deviation should be punished. This allows the designer to optimize the regulator to the specific implementation: Is it more important to have low input-deviation than state-deviation?

In this system there is only one constraint on the input; that it shall not exceed 30 degrees. Therefore a small R-value compared to Q is warranted. This will produce a regulator that tries harder to minimize state-deviation than input-deviation.

2.3 Model Predictive Control

Model Predictive Control is another way of introducing feedback to an optimal control system. In an MPC controlled system the optimal response and input is recalculated at every timestep, the input used is simply the first of the optimal input values calculated at every step.

This is a drastically different approach to the LQ-method impemented in this laboratory exercise.

2.3.1 Modified Control Hierarchy with MPC

Rather than introducing the Advanced Control Layer with the LQ-controller; introducing MPC would introduce the feedback to the Optimization Layer instead. The optimization layer would use the current state value and generate an optimal trajectory to the target, outputting pitch-setpoints to the Basic control layer at every timestep.

2.4 Experimental results

It is very clear that introducing feedback produced results much better than the ones achieved without feedback, almost regardless of the tuning of the LQR regulator. This is of course what was expected.

The group performed experiments with different values of Q and R to analyze the performance of the helicopter with different tunings.

Figure 5 shows that changing the R-value did not drastically change the response, nevertheless it was observed that a larger R-value produces greater state-offset and smaller input-offset.

Figure 6 shows that increasing the Q-value greatly reduces the offset from the optimal and actual travel response. It was also observed that this caused oscillations in the input-values.

The experimental results are consistent with the theory of the LQ controller laid out in section 2.2.

The group observed that the state never reached the optimal trajectory, even with very high Q-values. The group believes that adding integration to the LQ-regulator would eliminate the stationary- and reduce the dynamic offset. This was however not explored further.

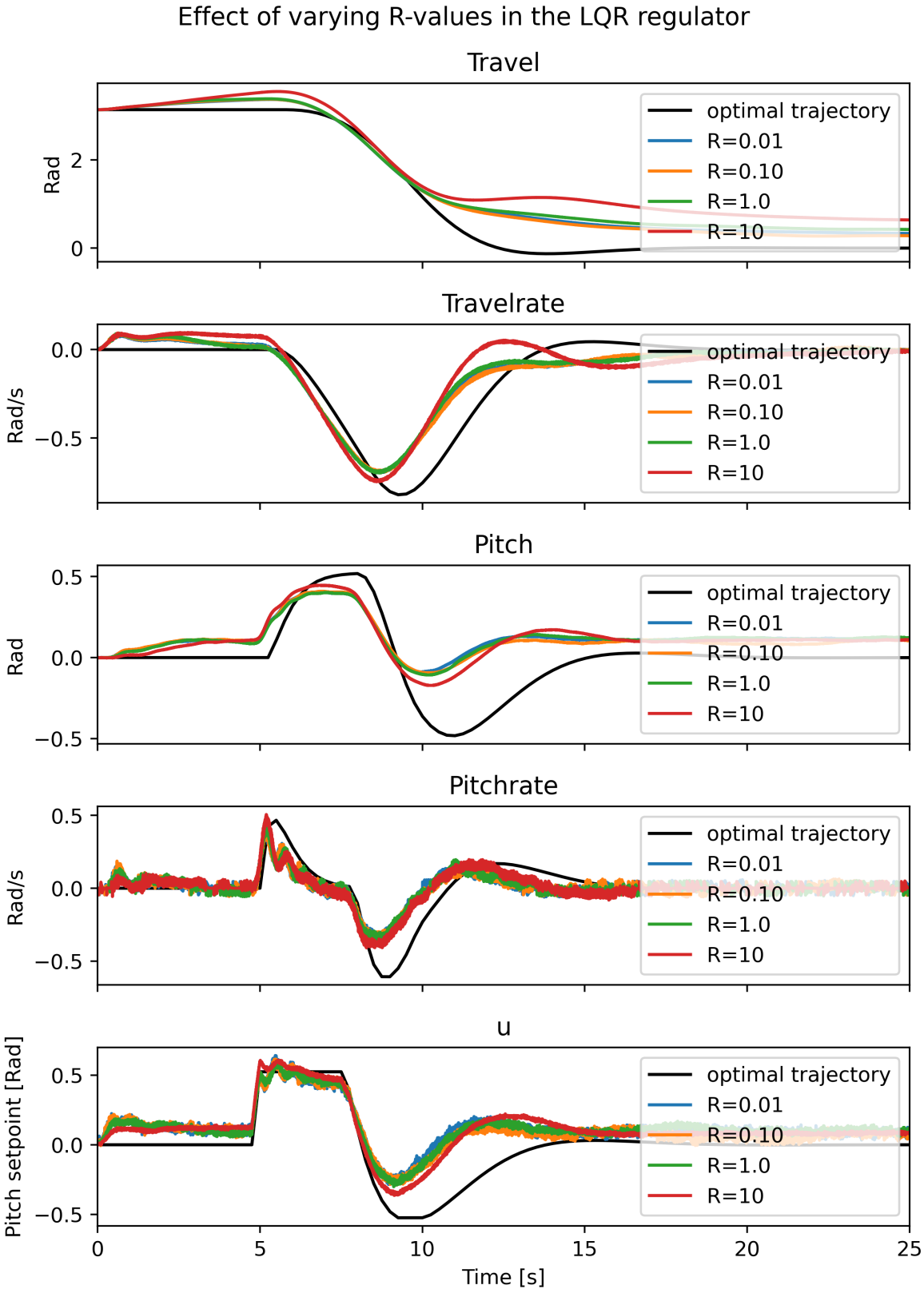


Figure 5: Results of varying R-values wihle keeping Q=diag([1,1,1,1])

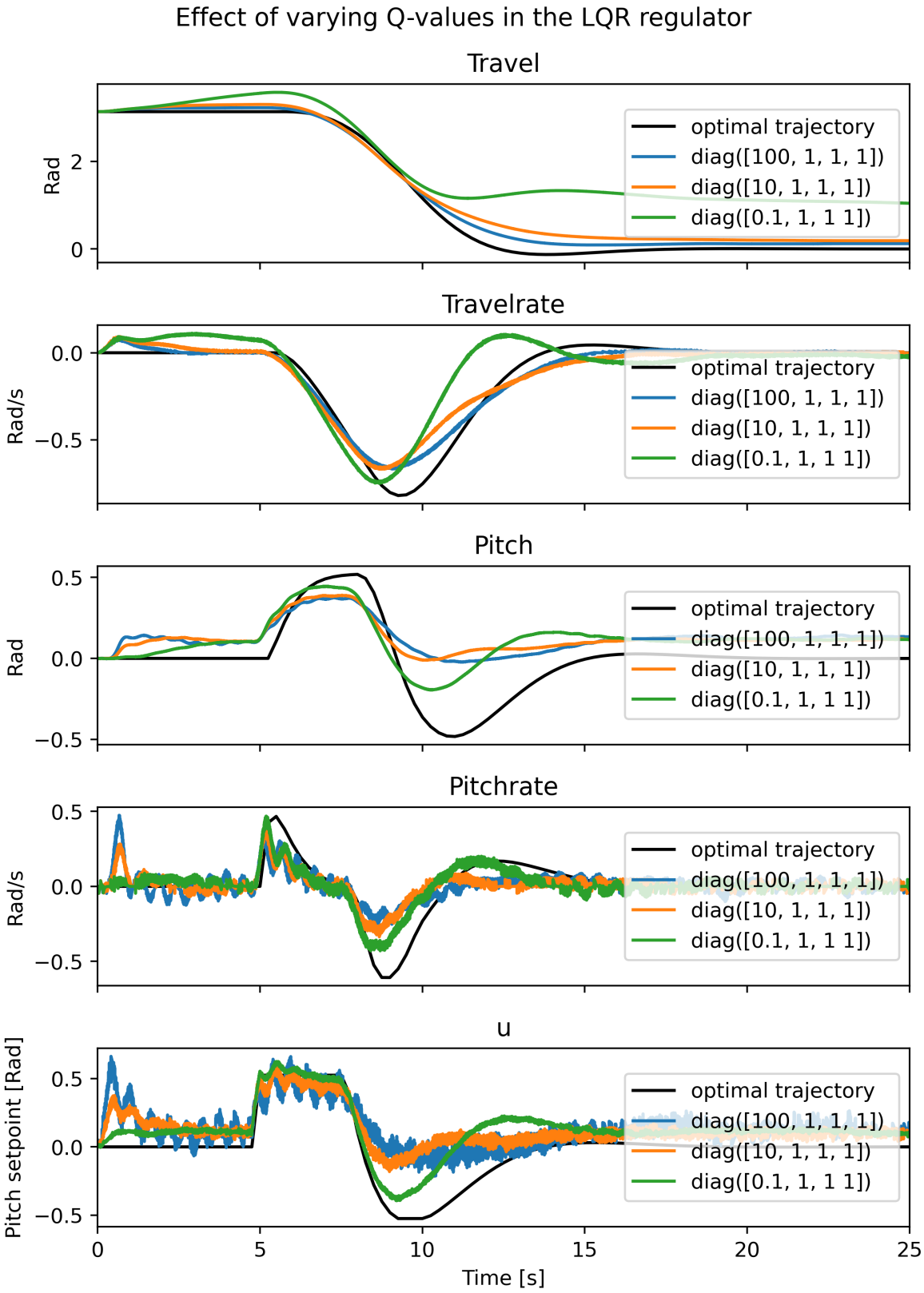


Figure 6: Results of varying Q-values while keeping $R = 1$

2.5 MATLAB and Simulink

Code and diagrams go here

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

We have to add the equation for elevation

$$\ddot{e} + K_3 K_{ed} \dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c$$

to the system defined in eq. (1.3)

$$\underbrace{\begin{bmatrix} \dot{\lambda} \\ \dot{r} \\ \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3 K_{ep} & -K_3 K_{ed} \end{bmatrix}}_{\boldsymbol{A}_c} \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3 K_{ep} \end{bmatrix}}_{\boldsymbol{B}_c} \underbrace{\begin{bmatrix} p_c \\ e_c \end{bmatrix}}_{u} \tag{3.1}$$

3.1 The continuous model

Answer 10.4.1.1

3.2 The discretized model

Answer 10.4.1.2

3.3 Experimental results

Printouts of data from relevant experiments (plots). Discussion and analysis of the results. Answer 10.4.2.6 here.

3.4 Decoupled model

Answer 10.4.2.7

3.5 MATLAB and Simulink

Code and diagrams go here

3.6 Optional exercise

Which constraints did you add? What was the results? Plots? Discussion?

References

[1] Bjarne Foss and Tor Aksel N. Heirung. *Merging optimization and control*. 2016.