OptReg lab report

Group ??
Vemund Rogne
Jørgen Haaland

March 30, 2021

NTNU
Department of Engineering Cybernetics

Contents

1 Optimal Control of Pitch/Travel without Feedback **1**
1.1 Derivation of a continous time state space model . . . 1
1.1.1 A deeper dive into the state-space model . . . 1
1.1.2 Stability and eigenvalues . . . 1
1.2 Discretizing the continous time model . . . 2
1.2.1 Checking stability . . . 3
1.3 The open loop optimization problem . . . 3
1.3.1 Formulating the cost function . . . 3
1.3.2 The constraints of the optimization problem . . . 3
1.4 The weights of the optimization problem . . . 4
1.5 The objective function . . . 4
1.6 Experimental results . . . 5
1.7 MATLAB and Simulink . . . 5
1.7.1 MATLAB . . . 5
1.7.2 Simulink . . . 8

2 Optimal Control of Pitch/Travel with Feedback (LQ) **9**
2.1 Introducing feedback . . . 9
2.2 LQ controller . . . 9
2.3 Model Predictive Control . . . 9
2.4 Experimental results . . . 9
2.5 MATLAB and Simulink . . . 9

3 10.4 - Optimal Control of Pitch/Travel and Elevation with Feedback **10**
3.1 The continuous model . . . 10
3.2 The discretized model . . . 10
3.3 Experimental results . . . 10
3.4 Decoupled model . . . 10
3.5 MATLAB and Simulink . . . 10
3.6 Optional exercise . . . 10

References **12**

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:

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

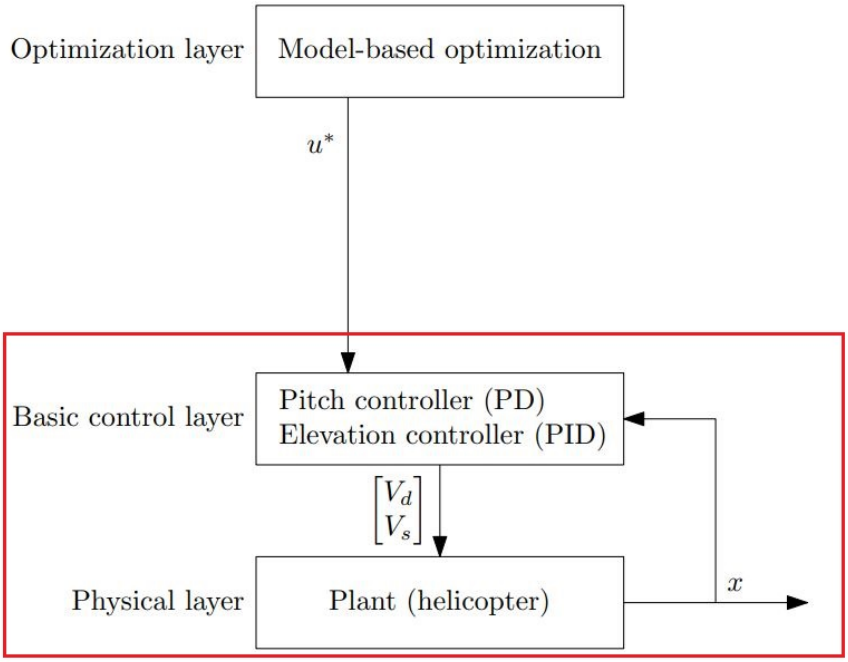


Figure 1: The red box encapsulate what is modelled in the state space model described by eq. (1.3).

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

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

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.

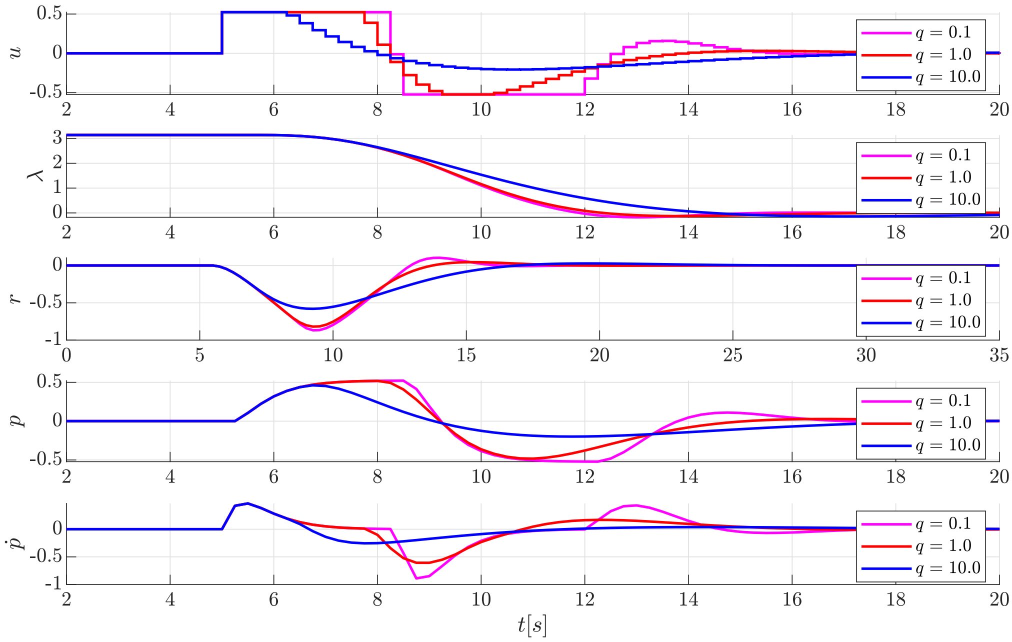


Figure 2: Manipulated variable and outputs with different values of q.

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.

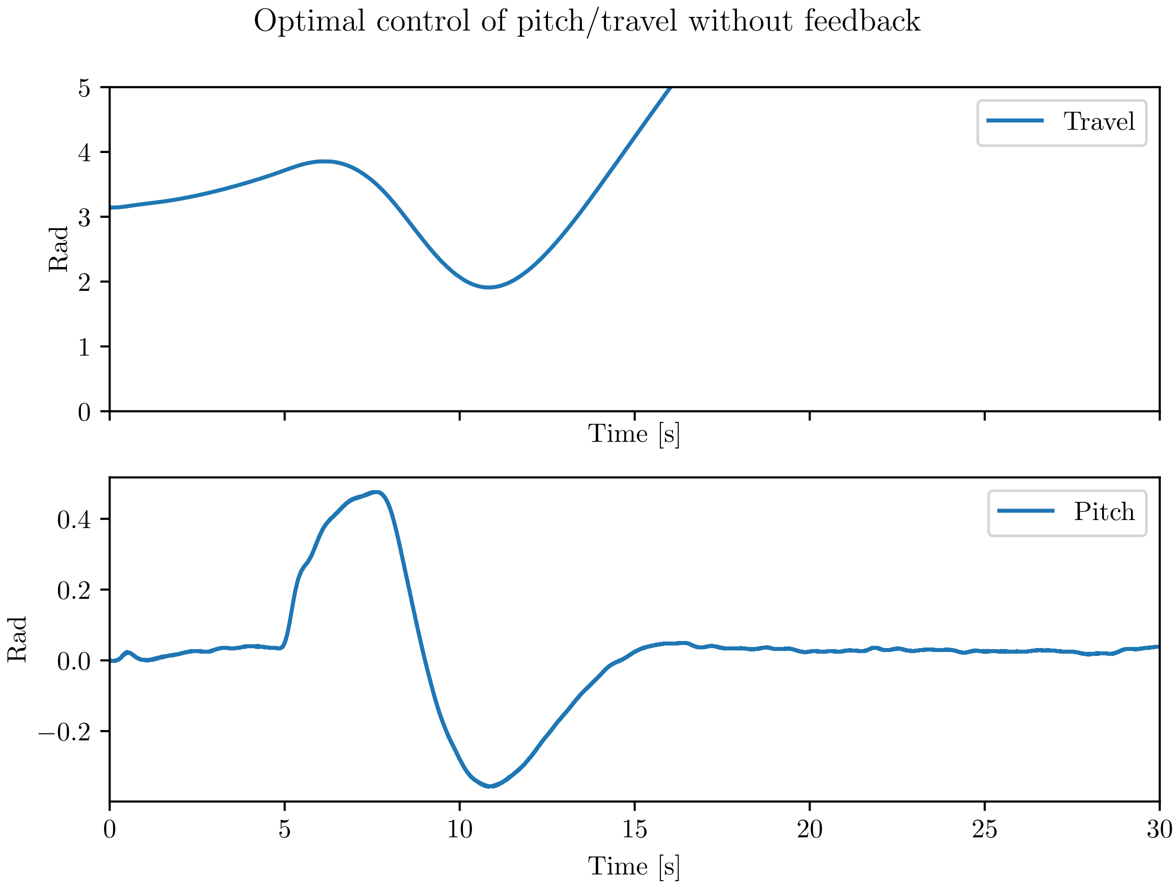


Figure 3: Results of LAB2

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
```

```
% Continous time model
Ac = [0, 1,       0,        0;
      0, 0,     -K_2,        0;
      0, 0,       0,        1;
      0, 0, -K_1*K_pp, -K_1*K_pd];

Bc = [      0;
            0;
            0;
      K_1*K_pp];

% Discrete time system model. x = [lambda r p p_dot]'
delta_t = 0.25; % sampling time
A1 = eye(4) + (delta_t.*Ac);
B1 = (delta_t.*Bc);

% 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)

% Trajectory start and end values
lambda_0 = pi;
lambda_f = 0;
% Initial values
x1_0 = lambda_0;                              % 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
z  = zeros(N*mx+M*mu,1);                      % Initialize z for the whole horizon
z0 = z;                                       % Initial value for optimization

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

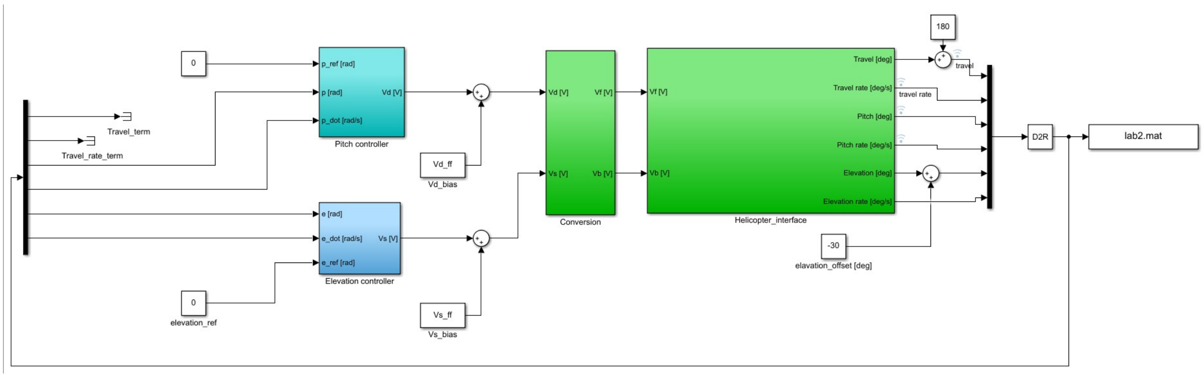


Figure 4: Simulink diagram used in lab 2.

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

2.1 Introducing feedback

Feedback is introduced to eq. (1.7) by using the following term for the input:

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

, where u_k^* and $\boldsymbol{x}_k^*$ is, respectively, the optimal input- and state-trajectory calculated with, $\boldsymbol{x}_k$ is the state and $\boldsymbol{K}$ is the feedback gain matrix.

add cref to the equation doing this

Do we need to say something about k:
x_k is the state at timestep k (or something like that)

.

In eq. (2.1), the last term $-\boldsymbol{K}^T(\boldsymbol{x}_k - \boldsymbol{x}_k^*)$ is the correction term. If there is a deviation in the actual state $\boldsymbol{x}_k$ and the optimal state $\boldsymbol{x}_k^*$, this term will correct the input u_k so the next state $\boldsymbol{x}_{k+1}$ comes closer to the optimal state.

correct is not a good word here

Dette er en svært kronglete setning. Formuler denne bedre

The next challenge is to find a good feedback gain matrix $\boldsymbol{K}$, that gives us a good behavior for the system. We will use a LQ controller for this task.

2.2 LQ controller

Briefly explain LQ controller. Especially, but not limited to, what is the role of the matrices Q and R? Justify your choice of weights.

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 weights in the optimalization problem. The value of Q determines how much deviation in the state value should be penalized, while the value of R determines how much input-usage should be punished. This allows the designer to optimize the regulator to the specific implementation: A system where the input is cheap (like the helicopter used in this assignment) would have a relatively small value of R compared to Q.

2.3 Model Predictive Control

Answer 10.3.1.3 here.

2.4 Experimental results

Printouts of data from relevant experiments (plots). Discussion and analysis of the results. Answer 10.3.2.5 here.

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)

$$\underbrace{\begin{bmatrix} \dot{\lambda} \\ \dot{r} \\ \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3K_{ep} & -K_3K_{ed} \end{bmatrix}}_{\boldsymbol{A}_c} \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3K_{ep} \end{bmatrix}}_{\boldsymbol{B}_c} \underbrace{\begin{bmatrix} p_c \\ e_c \end{bmatrix}}_{u} \tag{3.1}$$

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

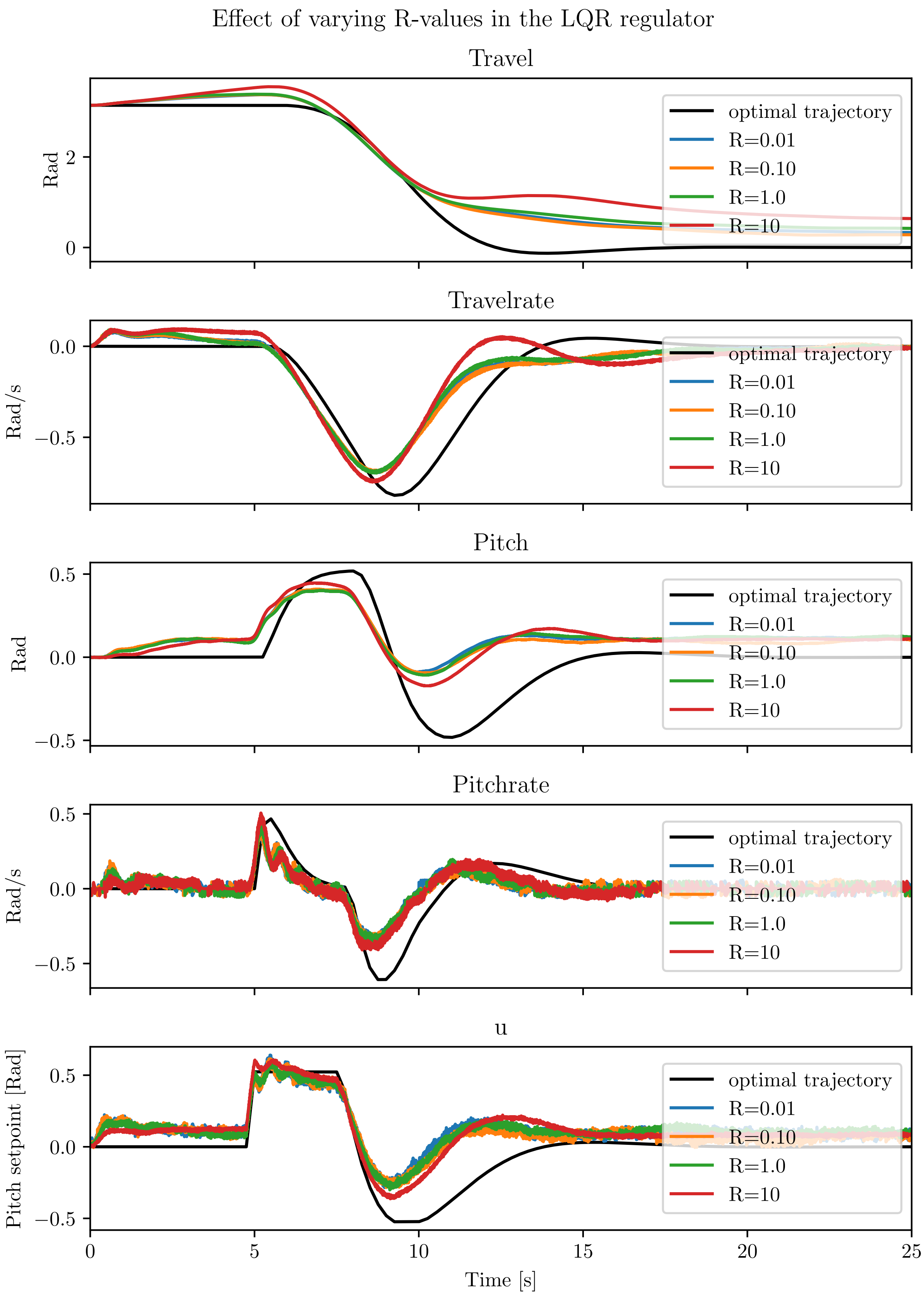
Answer 10.4.2.7

3.5 MATLAB and Simulink

Code and diagrams go here

3.6 Optional exercise

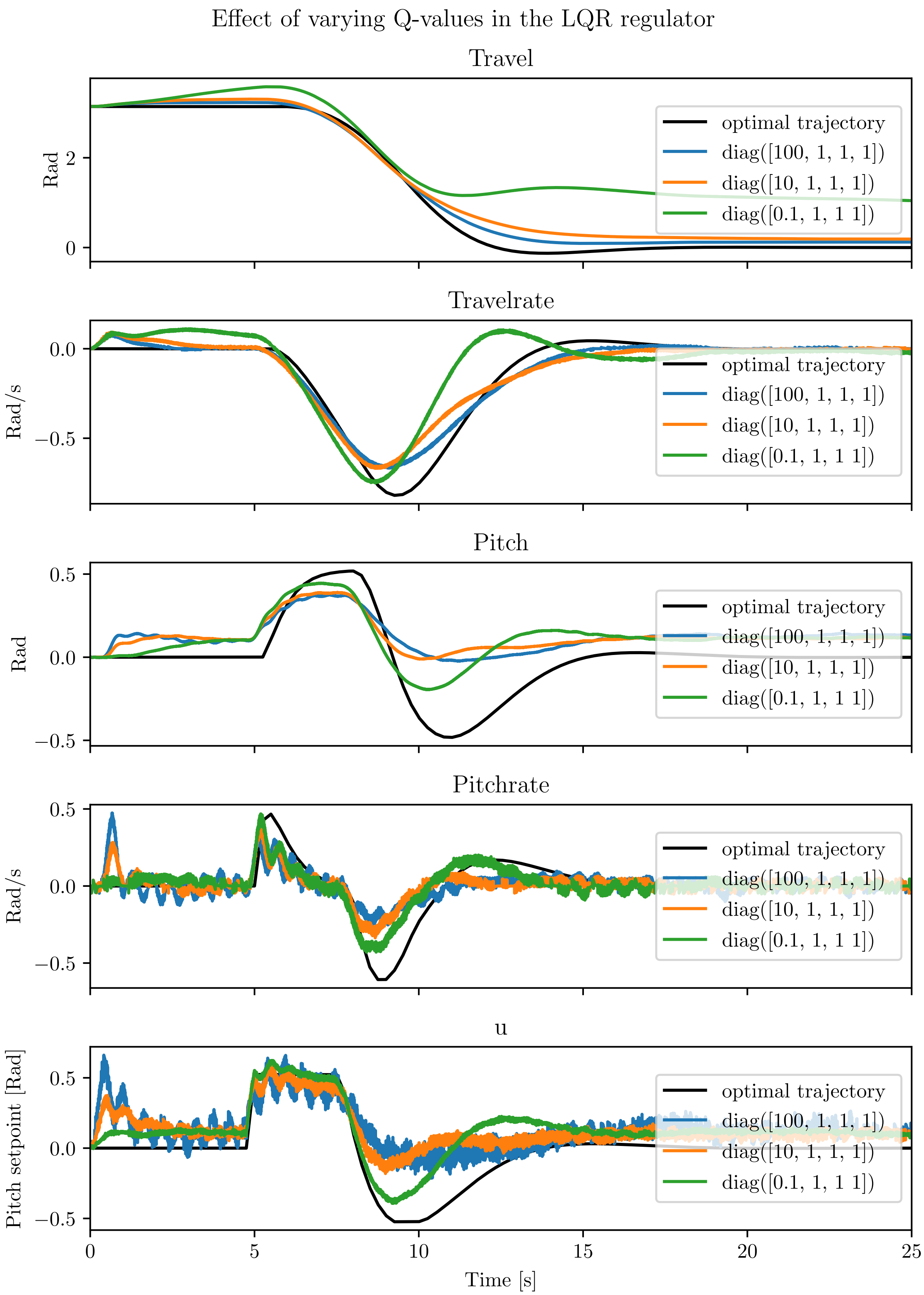
Which constraints did you add? What was the results? Plots? Discussion?



Effect of varying R-values in the LQR regulator
Travel
Rad
Travelrate
Rad/s
Pitch
Rad
Pitchrate
Rad/s
u
Pitch setpoint [Rad]
Time [s]
optimal trajectory
R=0.01
R=0.10
R=1.0
R=10

References

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


Effect of varying Q-values in the LQR regulator
Travel
Rad
2
0
optimal trajectory
diag([100, 1, 1, 1])
diag([10, 1, 1, 1])
diag([0.1, 1, 1 1])
Travelrate
Rad/s
0.0
−0.5
Pitch
Rad
0.5
0.0
−0.5
Pitchrate
Rad/s
0.5
0.0
−0.5
u
Pitch setpoint [Rad]
0.5
0.0
−0.5
0
5
10
15
20
25
Time [s]