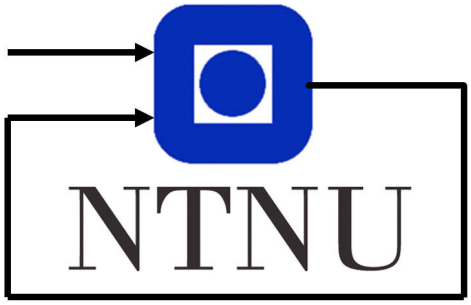
Helicopter Lab Report

TTK4135 Optimization and Control

495516
507755

April, 2021



Department of Engineering Cybernetics

Contents

1 10.2 - Optimal Control of Pitch/Travel without Feedback

1.1 The continuous model

The elevation is assumed to be zero in this part of the lab and we are left with modeling the pitch and travel of the helicopter. A model of the helicopter can be derived from the moment balances for the rotating axes pitch, travel and elevation. The equations of motion is derived in the lab exercise, and will be summarized below:

$$\ddot{e} + K_3 K_{ed} \dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c \tag{1a}$$
$$\ddot{p} + K_1 K_{pd} \dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c \tag{1b}$$
$$\dot{\lambda} = r \tag{1c}$$
$$\dot{r} = -K_2 p, \tag{1d}$$

where p_c is the desired pitch angle for the helicopter.

Utilizing the equations listed above we can write the model in continuous time state space form

$$\dot{x} = A_c x + B_c u, \tag{2}$$

where x and u equals

$$x = \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \end{bmatrix}, u = p_c. \tag{3}$$

The matrices A_c and B_c are defined as

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} \end{bmatrix} B_c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1 K_{pp} \end{bmatrix}. \tag{4}$$

The state space formulation is modeling respectively the travel, travel rate, pitch and pitch rate. The elevation is, as previously mentioned, currently being disregarded and will therefore not be included in the model. Hence, we do not model the PID controller and the associated feedback shown in figure 1.

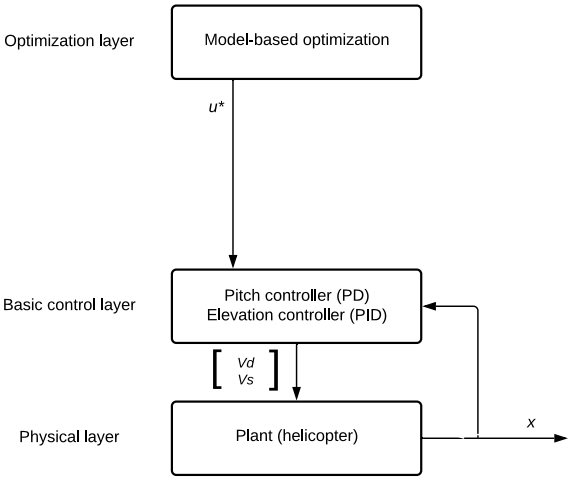


Figure 1: Control hierarchy of the system

The model includes the PD-controller of the basic control layer and the physical layer, as well as the feedback-loop from the plant to the PD-controller. The optimization layer is only included in the model in the sense that the output u^* will be fed directly into the basic control layer as the pitch reference p_c. The model-based optimization will be derived and calculated in section 1.3.

1.2 The discretized model

The explicit Euler method [3] uses the rule

$$x_{k+1} = x_k + \Delta t f(x_k, t_k), k = 0, ..., N \tag{5}$$

Inserting the continuous state space model derived above in equation (2) in the forward Euler method we get

$$x_{k+1} = x_k + \Delta t(A_c x_k + B_c u_k) \tag{6}$$

Hence, the resulting model on discrete time state space form becomes

$$x_{k+1} = \underbrace{(I + \Delta t A_c)}_{A_d} x_k + \underbrace{\Delta t B_c}_{B_d} u_k, \tag{7}$$

where the matrices A_d and B_d equals

$$A_d = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & -K_2\Delta t & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & -K_1K_{pp}\Delta t & 1 - K_1K_{pd}\Delta t \end{bmatrix}, B_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1K_{pp}\Delta t \end{bmatrix}. \quad (8)$$

1.3 The open loop optimization problem

In this lab we aim to calculate an optimal trajectory for moving the helicopter from $x_0 = [\pi \ 0 \ 0 \ 0]^T$ to $x_f = [0 \ 0 \ 0 \ 0]^T$. We derive this from minimizing the objective function given by

$$\Phi = \sum_{i=0}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + qp_{ci}^2, q \geq 0, \quad (9)$$

where we penalize deviations from the reference trajectory as well as the input and pitch reference point p_c by changing the value of the weight q.

We add a constraint to the pitch reference point p_k

$$|p_k| \leq \frac{30\pi}{180}. \quad (10)$$

In order to find the optimal path in MATLAB, we need to formulate the optimization problem as a standard form inequality constrained QP, where all inputs and states are optimization variables $z = [x_1^T, \ ... \ , \ x_N^T, u_0^T, \ ... \ , u_{N-1}^T]$. Using the discrete state space model derived in equation (6) and the objective function in equation (9) we obtain the following QP problem

$$\begin{aligned} \min_z f(z) &= \frac{1}{2} z^T G z \\ s.t & \\ A_{eq} z &= b_{eq} \\ |u_k| = |p_k| &\leq \frac{30\pi}{180}, \end{aligned} \quad (11)$$

where the matrix G is defined as

$$G = \begin{bmatrix} Q & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & Q & \ddots & & \vdots \\ \vdots & & \ddots & q & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & q \end{bmatrix}. \quad (12)$$

The Q-matrix is representing the weighting of the states in the cost function (9) and is defined as

$$Q = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{13}$$

The matrices A_{eq} and b_{eq} represents the discrete model of the system, where

$$A_{eq} = \begin{bmatrix} I & 0 & \cdots & 0 & -B_d & 0 & \cdots & 0 \\ -A_d & I & \ddots & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & -A_d & I & 0 & \cdots & 0 & -B_d \end{bmatrix}, b_{eq} = \begin{bmatrix} A_d x_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The MATLAB-code for formulating this QP-problem is shown in Section 1.7.

1.4 The weights of the optimization problem

As mentioned, we can penalize the usage of the input, the pitch reference point p_c, by changing the value of the weight q. Increasing q will produce a subtle and smaller-scaled input sequence, however, allowing larger deviations of the states of the system. Decreasing q, allows for a more aggressive control, where the cost function emphasise minimizing the deviations of the travel from the reference trajectory, rather than emphasising a conservative pitch-rate. After minimizing the objective function in equation (9) in MATLAB with different weights q, we obtain the values for u^*, the optimal input sequences, and their respective optimal travel trajectories, λ^*. The input sequences with respect to the weights $q = 0.1, 1$ and 10 is shown in figure 2, while the corresponding travel trajectories is shown in figure 3.

As expected, in figure 3 we observe that when using a smaller value of q and allowing for a more aggressive control of the pitch, the travel converges faster towards the final reference value $\lambda_f = 0$. This correlate with figure 2, where the input and pitch reference point p_c for the smallest weights tangents both the constraint values that was labelled in equation 10, which trigger a steeper travel trajectory compared to the trajectory related to the weight $q = 10$.

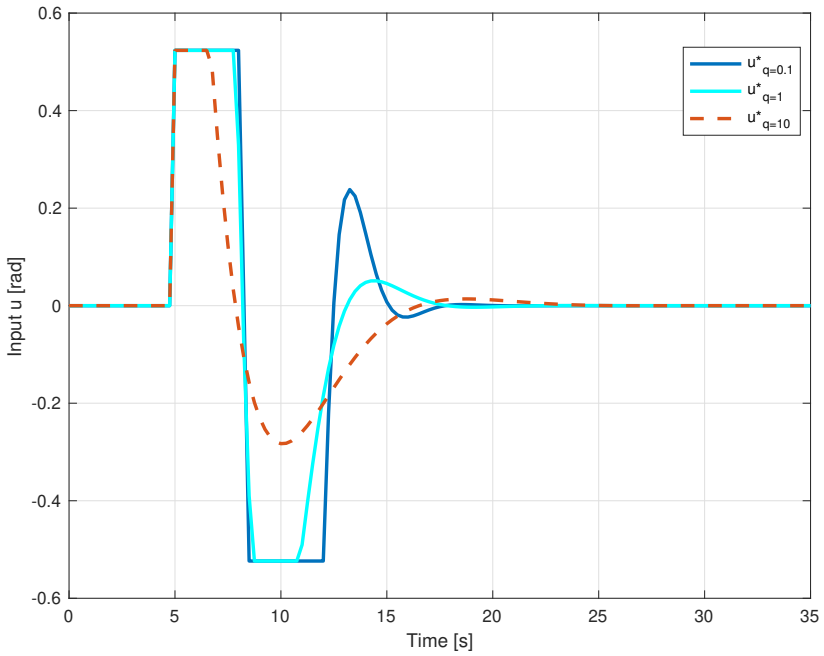


Figure 2: Calculated optimal input sequence u^* for different weights

1.5 The objective function

The objective function that is defined in equation (9), includes the terms $(\lambda_{i+1} - \lambda_f)^2$ and qp_{ci}^2. As the travel λ becomes smaller, the difference between the two terms grows quadratically, due to the quadratic terms in the objective function. Hence, as the travel λ approaches zero, the pitch input p_c will be largely emphasised in the objective function.

Another potential problem that can arise from this cost function, is the possibility of choosing the control horizon N too small. Since we have a constraint on the pitch reference point, as well as having the possibility to penalize the control input in the cost function, the input sequence will be limited for the helicopter travel angle. This means that if we define a too small control horizon N, we do not have enough input to steer the helicopter to its desired final state before the simulation time has ended. This will cause the travel rate to be nonzero at end-time and the helicopter will drift away from the final reference point λ_f.

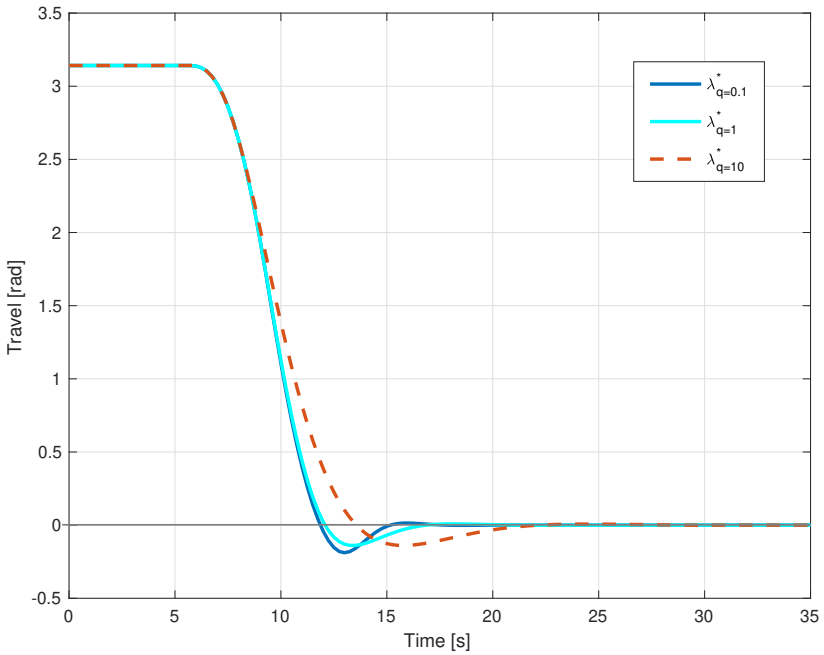


Figure 3: Calculated optimal travel trajectory λ^* for different weights

1.6 Experimental results

Figure 4 shows the helicopter travel trajectory for the different weights $q = 0.1, 1$ and 10 along with their respective calculated optimal trajectories. We observe the same as in the simulation of the optimal travel trajectory; a smaller value of q leads to a faster convergence towards the reference angle of 0 degrees. However, independent of weight-value, we observe that the travel of the helicopter never reaches its intended destination $\lambda = \lambda_f$. Furthermore, the helicopter initiate with drifting the opposite way of the calculated trajectory, before recovering and turning towards the right direction.

In order for the helicopter to stabilize before beginning to calculate the optimal travel trajectory, we added a 5 seconds zero-padding in front of the input sequence, as seen in figure 2. This explains the initiate drift of the helicopter, where no input will cause minor weight and motor differences to create a nonzero pitch angle and helicopter-momentum in one of the directions. After five seconds, the helicopter-pitch starts following the calculated pitch reference sequence p_c, leading the helicopter to change direction and chase the calculated travel trajectory. Since the helicopter already had a mo-

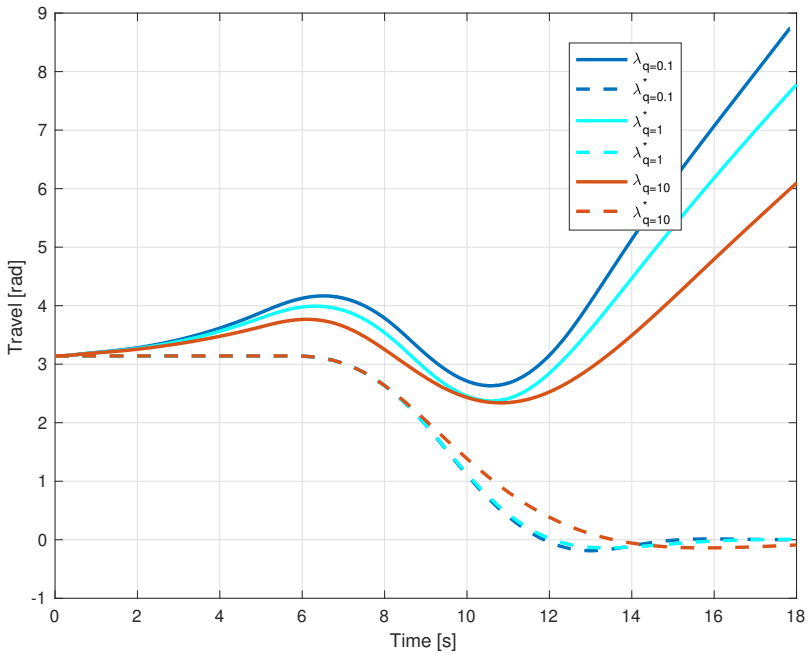


Figure 4: Measured travel trajectory with different q

mentum towards the opposite direction, the pre-calculated input sequence will not be sufficient to abruptly turn the helicopter. This causes the travel trajectory to always lag behind the intended calculated input sequence, and the final pitch-turn that was intended to correct a overshoot of the travel-angle, will now turn the helicopter's direction after moving approximately 90 degrees. In other words, the helicopter will never reach its desired state x_f.

The fact that the travel trajectory is derived from integrating the pitch twice will also lead to additional errors. Implementing a feedback control, would make it possible to correct u^* when the trajectory of the helicopter did not follow the optimal trajectory, and would have eliminated the deviations we experience with open-loop control.

An interesting observation is that the travel trajectory with weight $q = 10$, is in the open-loop control case closest to track the optimal travel trajectory. When the travel trajectory lags behind the calculated input sequence and trajectory, smaller weights and aggressive control will create more momentum and possibly add larger deviations from the trajectory.

1.7 MATLAB and Simulink

Listing 1: MATLAB code 10.2

```
%% Initialization and model definition
init05;

% 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];

% Number of states and inputs
mx = size(A1,2); % Number of states
mu = size(B1,2); % Number of inputs

% 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
z  = zeros(N*mx+M*mu,1);  % Initialize z
% for the whole horizon
z0 = z;
n = N*mx+M*mu;

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

xl      = -Inf*ones(mx,1);   % Lower bound on states
xu      = Inf*ones(mx,1);    % Upper bound on states
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);
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) = 2;    % Weight on state x1
Q1(2,2) = 0;   % Weight on state x2
Q1(3,3) = 0;   % Weight on state x3
Q1(4,4) = 0;   % Weight on state x4
P1 = 10;       % Weight on input, q
Q = gen_q(Q1,P1,N,M);
c = zeros(n, 1);

%% Generate system matrixes for linear model
Aeq = gen_aeq(A1,B1,N,mx,mu);
beq = Aeq*z;
beq(1) = x1_0;


%% Solve QP problem with linear model

tic
[z,lambda] = quadprog(Q,c,[],[],Aeq,beq,vlb,vub);
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];
```

```
%% Input imported to the helicopter
t = 0:delta_t:delta_t*(length(u)-1);

input = timeseries(u, t);
```

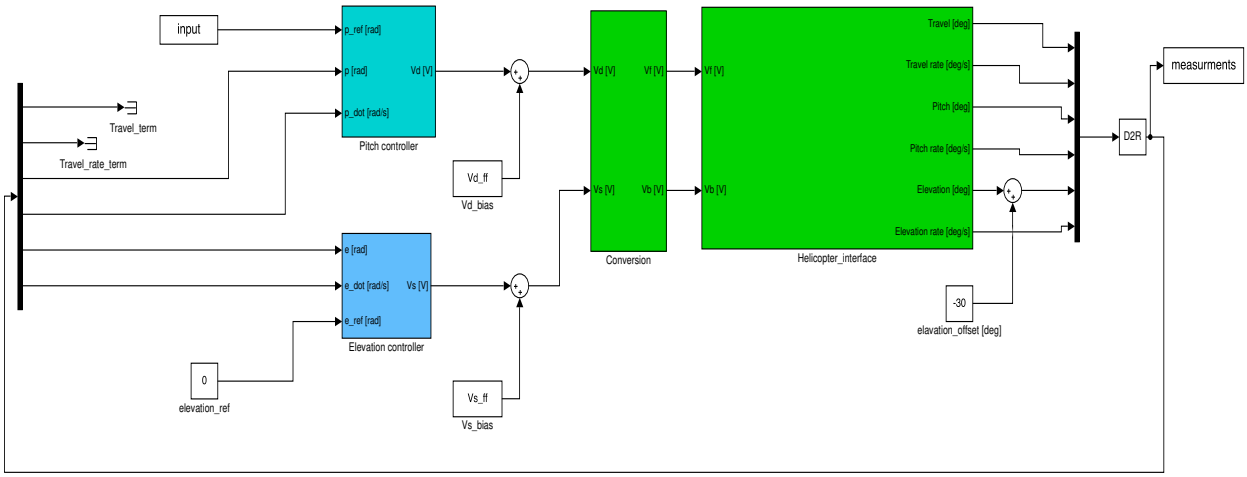


Figure 5: SIMULINK model 10.2

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

2.1 LQ controller

As suggested in the section 10.2, we can implement feedback in the controller in order to correct deviations caused by errors in the model. This will by done by introducing a LQ controller that modifies the input if the state variables deviates from the calculated optimal trajectories. The LQ controller calculates the optimal feedback gain K such that the state-feedback law

$$u_k = u_k^* - K^\top (x_k - x_k^*), \tag{14}$$

minimizes the quadratic objective function

$$\min_{z \in R^n} f(z) = \sum_{i=0}^{\infty} \frac{1}{2} \Delta x_{i+1}^\top Q \Delta x_{i+1} + \frac{1}{2} \Delta u_t^\top R \Delta u_t, \tag{15a}$$

subject to the discrete-time state-space model

$$\Delta x_{i+1} = A \Delta x_i + B \Delta u_i, \tag{15b}$$

where

$$z^\top = (\Delta x_1^\top, ..., \Delta x_N^\top, \Delta u_0^\top, ..., \Delta u_{N-1}^\top) \tag{15c}$$

$$\Delta x = x - x^* \tag{15d}$$

$$\Delta u = u - u^*. \tag{15e}$$

The K-matrix is derived by solving the stationary Riccati equation [2]

$$K = R^{-1} B^\top P (I + B R^{-1} B^\top P)^{-1} A \tag{16a}$$

$$P = Q + A^\top P (I + B R^{-1} B^\top P)^{-1} A \tag{16b}$$

By using the dlqr-function, found in the control system toolbox for MATLAB, we acquire the gain matrix K.

The weighting matrices for the states and the manipulated variable are respectively Q and R as defined in the objective equation (15a). If we assume that the R-matrix is constant, we only need to tune the Q-matrix. By looking at the objective function, we see that higher values of Q will lead to more emphasis on minimizing Δx and allowing higher values of Δu. On the diagonal of the Q-matrix we can weight the different states in our system,

and hence prioritize the states we want to follow their respective trajectory. Initially, we set the weighting tensors to

$$Q_0 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, R_0 = 1. \tag{17}$$

We chose these values to penalize deviations in travel and travel rate, since the travel trajectory has the largest range of angles and movement rate, leading the travel to be more prone to errors than the pitch state and its respective derivative.

After running and tuning the helicopter for a while it became evident that penalizing the travel and travel rate compared to the pitch and pitch rate was of greater importance in order to obtain an optimal result. In order to maximize the input given to the helicopter, we added an extra zero to the weighting-values of the travel and travel rate, and ended up with the tensors

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, R = 1. \tag{18}$$

The K-matrix with these weights were calculated to be

$$K = [-3.408 \quad -5.819 \quad 1.879 \quad 0.371]. \tag{19}$$

As expected, when using the K-matrix above in equation (14), we see that deviations in travel and travel rate will cause a greater increase of u_k to help bringing the helicopter travel back on the optimal trajectory. The experimental results will be discussed further in section 2.3.

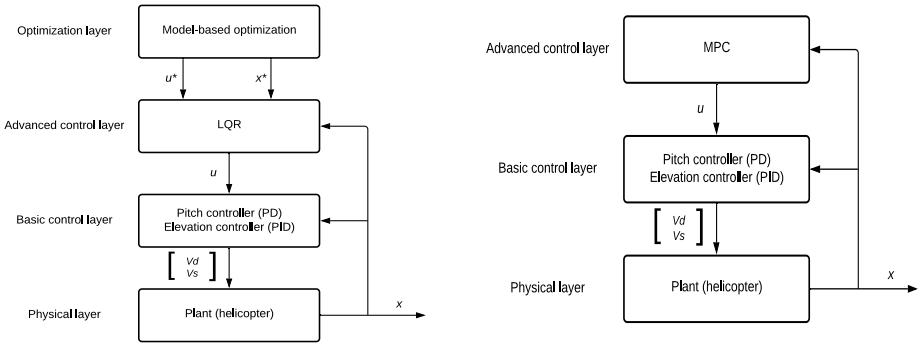
2.2 Model Predictive Control

An alternative strategy would be to use MPC, model predictive control, instead of an LQ-controller. An MPC, compared to an LQ controller, solves the QP-problem at each sampling instant, using the current state of the system as the initial state [2]. Hence, the MPC yields an optimal control sequence at each time step, where the first control in this sequence is applied to the plant.

There are advantages of choosing an MPC over our implemented LQ-controller. Our model will not be a completely correct representation of the real system and disturbances that are not accounted for in our model can effect the helicopter trajectory. Hence, the states will to a certain extent deviate from our prediction x^*. As a result, the LQ-controller will try to follow the pre-calculated values x^* and u^*, even though a new optimal trajectory from the current state will be different compared to the initial sequence. With an MPC, the trajectory will be modified at every time step, leading the controller to no longer being bound by the first pre-calculated sequence.

A disadvantage of implementing MPC is the computational time and cost. While our original LQ-controller computes an input sequence one time, our MP-controller solves a QP-problem at every time step. This can be computationally demanding, and cause instability if the system has very fast dynamics.

The difference between the control hierarchy layers with the respective LQ controller and MPC is shown in figure 6. The main difference in figure 6b is the lack of the optimization layer and LQR-layer, which is accomplished by renewing the optimal input sequence for every time step.



(a) Control hierarchy with LQ-controller (b) Control hierarchy with MPC

Figure 6: Illustration of control hiearchy layers

2.3 Experimental results

The data for the travel and pitch angle with the weighting tensors stated in equation (17) and (18), have been plotted in figure 7 and 8.

As mentioned in section 2.1, by comparing figure 7 and 8, we observe that a

larger ratio between the travel states and input weightings allows for a more aggressive control use and a optimal travel trajectory.

The main difference between these results with added feedback compared to the open-loop control in figure 4, is how well the travel λ is following its calculated trajectory. While optimal control without feedback has no way of adjusting the input u, we are now able to dynamically change the input at every time step, so that the helicopter no longer drifts to the same extent. However, we observe in figure 7 and 8, that the helicopter does not quite reach x_f. This is due to the fact that we have not emphasised penalizing the deviations of pitch and pitch rate in our cost function. This is evident in figure 7 and 8, where we observe that the pitch response is far from following the calculated optimal trajectory u. Another reason is the model imperfections that was discussed in section 1.6.

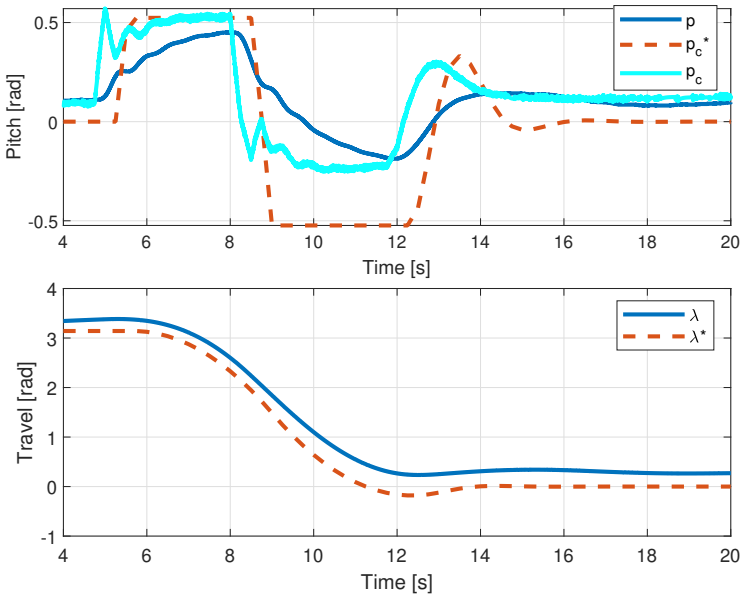


Figure 7: Pitch and travel with Q_0, R_0

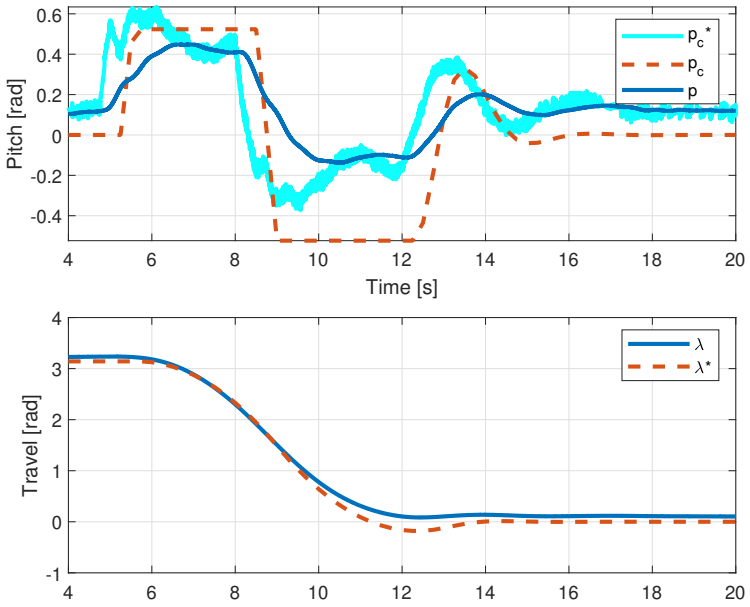


Figure 8: Pitch and travel with Q, R

2.4 MATLAB and Simulink

Listing 2: MATLAB code 10.3

```
%% Initialization and model definition
init05;

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 ; delta_t + (K_1 * K_pp)];

% Number of states and inputs
mx = size(A1,2);          % Number of states
mu = size(B1,2);          % Number of inputs

% 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
z  = zeros(N*mx+M*mu,1);   % Initialize z
z0 = z;                    % Initial value for optimization

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

xl      = -Inf*ones(mx,1);   % Lower bound on states
xu      = Inf*ones(mx,1);    % Upper bound on states
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);
vlb(N*mx+M*mu)  = 0;
vub(N*mx+M*mu)  = 0;

% Generate the matrix Q and the vector c
Q1 = zeros(mx,mx);
Q1(1,1) = 2 ;                % Weight on state x1
Q1(2,2) = 0;                 % Weight on state x2
Q1(3,3) = 0;                 % Weight on state x3
Q1(4,4) = 0;                 % Weight on state x4
P1 = 0.1;                    % Weight on input
Q = gen_q(Q1,P1,N,M) ;       % Generate Q
c = zeros(N*mx+M*mu,1);      % Generate c

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

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

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

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

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];
x = [x1 , x2 , x3 , x4];
%% LQ Control
Q = diag([100 30 0.1 0.1]);
R = 1;
K = dlqr(A1, B1, Q, R);
```

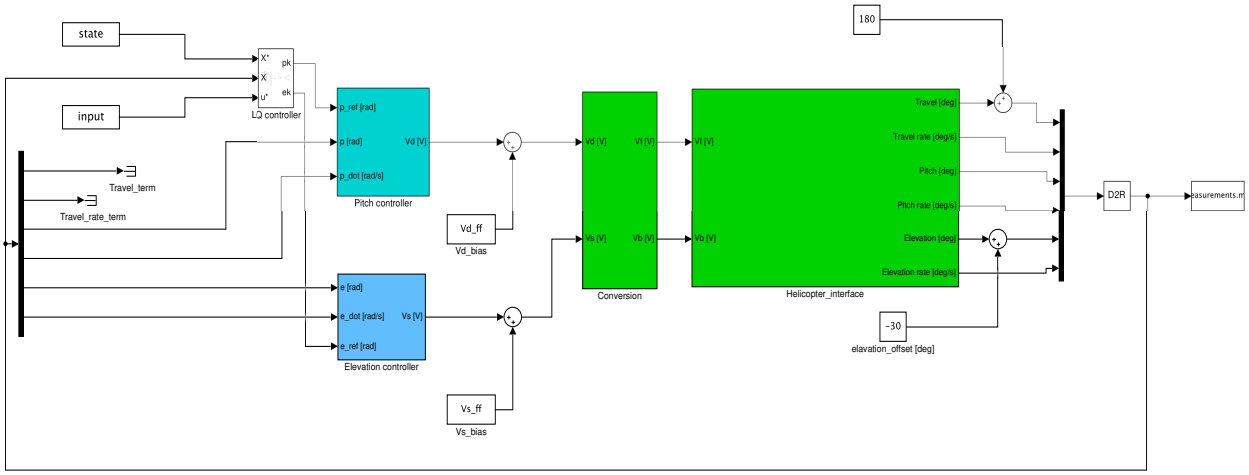


Figure 9: SIMULINK model 10.3

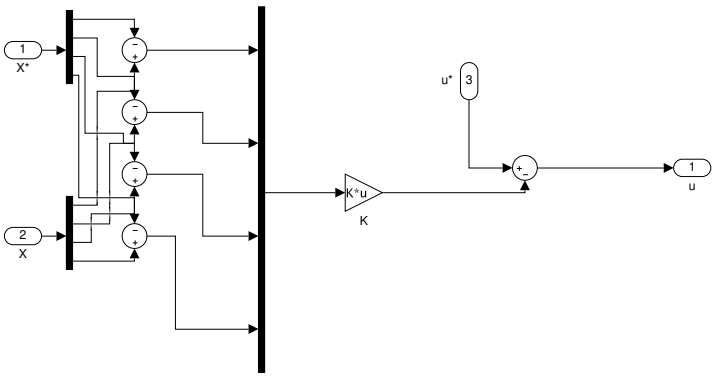


Figure 10: LQ control block in SIMULINK model 10.3

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

3.1 The continuous model

We introduce an extra dimension to the optimal trajectory by adding the states elevation e and elevation rate $\dot{e}$ to our model. We write the extended model in continuous state space form

$$\dot{x} = A_c x + B_c u, \tag{20}$$

where x and u now equals

$$x = \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix}, u = \begin{bmatrix} p_c \\ e_c \end{bmatrix}. \tag{21}$$

By utilizing the equations of motion that is listed in equation (1), we get the matrices

$$\mathrm{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} & 0 & \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3 K_{ep} & -K_3 K_{ed} \end{bmatrix}$$

$$\mathrm{B}_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3 K_{ep} \end{bmatrix}.$$

3.2 The discretized model

Discretizing the model using the forward Euler method as described in section 1.2 in equation (5), we obtain the same discretized state space model

$$x_{k+1} = \underbrace{(I + \Delta t A_c)}_{A_d} x_k + \underbrace{\Delta t B_c}_{B_d} u_k, \tag{23}$$

where the matrices A_d and B_d equals

$$\mathrm{A}_d = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & -K_2\Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & -K_1K_{pp}\Delta t & 1 - K_1K_{pd}\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & -K_3K_{ep}\Delta t & 1 - K_3K_{ed}\Delta t \end{bmatrix}$$

$$\mathrm{B}_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1K_{pp}\Delta t & 0 \\ 0 & 0 \\ 0 & K_3K_{ep}\Delta t \end{bmatrix}.$$

3.3 Experimental results

Introducing the new objective function

$$\Phi = \sum_{i=0}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2, q \geq 0, \tag{25}$$

with the added nonlinear elevation constraint

$$e_k \geq \alpha exp(-\beta(\lambda_k - \lambda_t)^2) \forall k \in \{1, ..., N\}, \tag{26}$$

we obtain new input and state sequences u^* and x^* that additionally calculates optimal trajectories for elevation and elevation rate. Since the constraint is nonlinear, we had to use the MATLAB function fmncon, that uses a SQP-algorithm designed for non-linear optimization problems.

We tried several values for q_1 and q_2. Initially, we used the values $q_1 = 1$ and $q_2 = 1$. However, we observed that in order for the elevation to satisfy the belonging inequality constraint as stated in equation (26) more accurately, we had to decrease q_2 compared to the other weightings in the cost function. We therefore landed on the values $q_1 = 1$ and $q_2 = 0.1$. If we decreased q_2 further we observed that the pre-calculated travel trajectory was to slow compared to the time horizon set to $N = 40$.

We continued using the LQ-controller with feedback as described in section 2.1, and ended up with the following weights for the calculation of K:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}, R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}, \tag{27}$$

resulting in the following K-matrix

$$K = \begin{bmatrix} -2.599 & -6.686 & 3.028 & 0.892 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9.377 & 8.032 \end{bmatrix}. \tag{28}$$

We started out with $Q = I$ to observe which elements in the matrices that should be modified. After some tuning, it became evident that travel and elevation were the two states that should be prioritized penalizing. By increasing the weighting element related to elevation, the helicopter trajectory would prioritize clearing the elevation constraint. However, this would also lead to less accurate pitch and travel. Therefore, it was necessary to balance upholding the constraint in elevation, as well as reaching the desired travel value λ_f fast enough, due to the limited amount of time steps.

Figure 11 shows the elevation trajectory from running the helicopter with the weighting matrices as stated in equation (27). Even though we prioritized elevation in our weighting matrix Q, we still observe that the elevation is not able to follow the optimal trajectory and hence not satisfy the inequality constraint. This is due to the the simplified decoupled model that will be discussed in the next section.

The travel and pitch trajectory is shown in figure 12. In this figure we observe larger deviations from the optimal trajectories. As mentioned, this is because of our choice of tuning, where the weighting elements of pitch and travel were decreased in order for the elevation to follow its optimal trajectory.

3.4 Decoupled model

In the equations of motion (1) we see that the four states travel λ, travel rate $\dot{\lambda}$, pitch p and pitch rate $\dot{p}$ are decoupled from the the states elevation e and elevation rate $\dot{e}$. In reality the first four states are not completely decoupled from the last two. This can be easily verified by tilting the helicopter 90

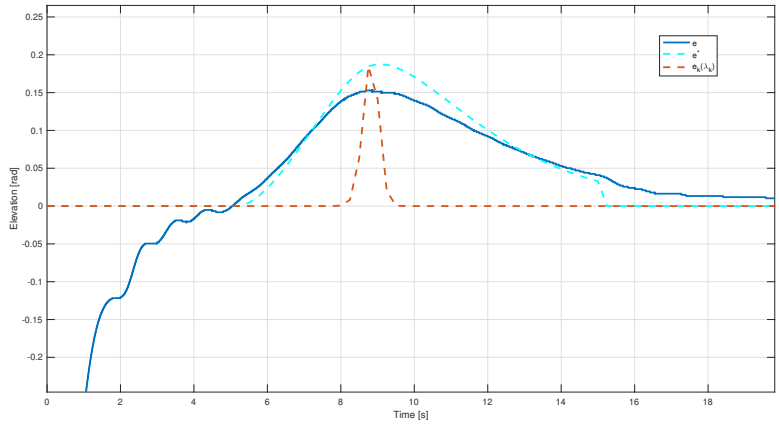


Figure 11: Elevation with Q and R compared to optimal trajectory and elevation constraint

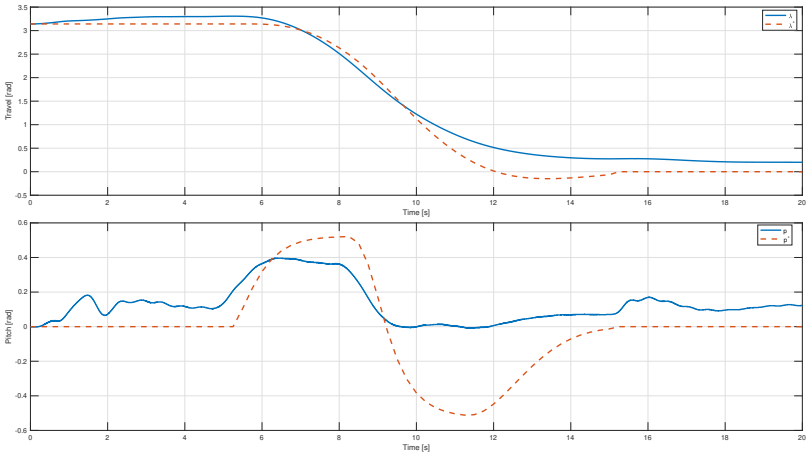


Figure 12: Pitch and travel with Q and R compared to optimal trajectory

degrees, such that the thrust direction points horizontally. In this position, it is not possible to adjust the elevation by increasing the thrust. Hence, the helicopter is not able to perfectly follow both optimal trajectories for the elevation e and the pitch p. As seen in figure 11 the helicopter is not able to respect the constraint when the pitch angle in figure 12 are nonzero. There must be a trade off in choosing between following the optimal trajectory for the pitch and elevation, where we in this case chose to prioritize elevation due to the added inequality constraint.

A solution to improve this would be to expand the simplified model. We would need to implement a model where the elevation e and the pitch p

influence each other, which can lead to nonlinear relationships. However, this model can be linearized around a desired point, for instance $e = 0$ and $p = 0$. By limiting the amount of tilt allowed, so that the pitch stays within the working area of the linearization, we would have a fairly simple model, as well as a relationship between the elevation e and the pitch p.

3.5 Optional exercise

We added a constraint to the travel rate $\dot{\lambda}$

$$\dot{\lambda} \leq 0.2 rad/s. \tag{29}$$

The results is shown in figure 13. We observe that the helicopter does not reach the end point λ_f by the end of the trajectory due to the small travel rate. We also see that the amount of pitch is limited, which is to be expected due to the correlation between the travel rate $\dot{\lambda}$ and pitch p.

We limited the travel rate as an attempt to reduce the coupling between pitch and elevation, where we expected the elevation to deviate less from its optimal trajectory. However, we see that the elevation trajectory continues to not comply with the constraint, because there still exists a coupling between the states, as well as other disturbances that causes deviations from the optimal sequence.

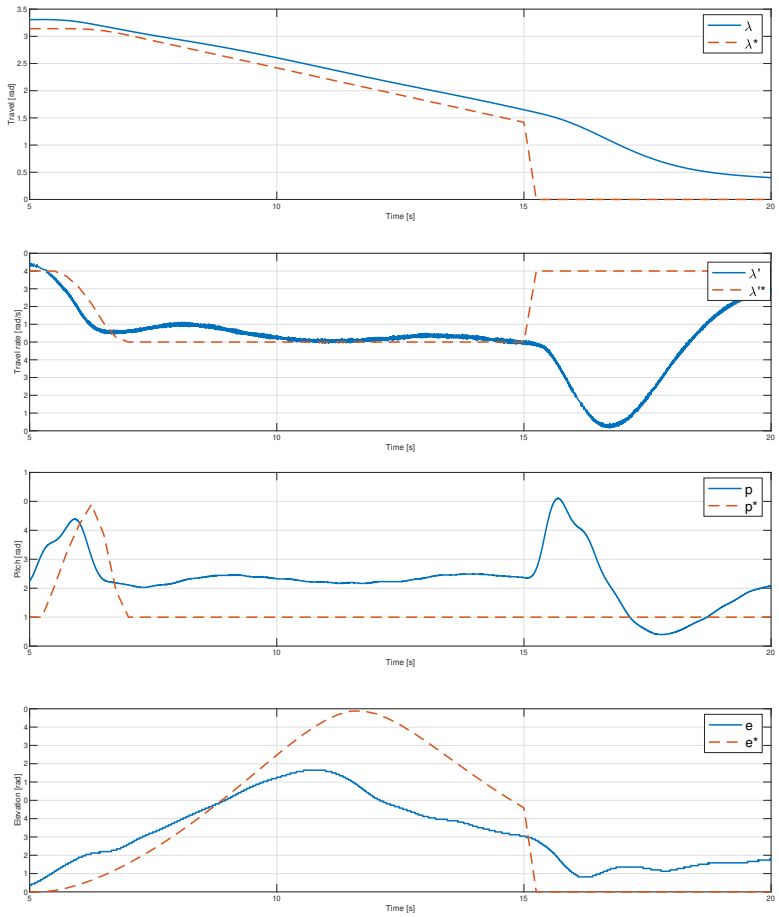


Figure 13: State trajectories with added travel rate constraint

3.6 MATLAB and Simulink

Listing 3: MATLAB code 10.4

```
init05; %

global N mx alpha beta lambda_t
% Global variables
q1 = 1;
q2 = 0.1;
alpha = 0.2;
beta = 20;
lambda_t = 2*pi/3;

% Discrete time system model. x = [lambda r p p_dot e e_dot]'
```

```
delta_t   = 0.25; % sampling time
A1 = [1  delta_t     0       0       0       0;
      0  1   -K_2*delta_t 0       0       0;
      0  0      1      delta_t  0       0;
      0  0  -K_1*K_pp*delta_t   1-(K_1*K_pd*delta_t)  0
0;
      0  0  0  0       1       delta_t;
      0  0  0  0   -K_3*K_ep*delta_t  1-(K_3*K_ed*delta_t)];

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

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

% Initial values
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
z  = zeros(N*mx+M*mu,1);    % Initialize z for the whole horizon
z0 = z; % Initial value for optimization
z0(1:6) = x0;

% Bounds
pk = 30*pi/180;
ul       = [-pk; -Inf];  % Lower bound on control
uu       = [pk; Inf];    % 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(1);         % Lower bound on state x3
xu(3)   = uu(1);% Upper bound on state x3

% Generate constraints on measurements and inputs
[vlb,vub]       =  gen_constraints(N,M,xl,xu,ul,uu);
```

```
vlb(N*mx+M*mu)  = 0;     % We want the last input to be zero
vub(N*mx+M*mu)  = 0;     % We want the last input to be zero

% Generate the matrix Q and the vector c
Q1 = zeros(mx,mx);
Q1(1,1) = 2 ;       % Weight on state x1
Q1(2,2) = 0;        % Weight on state x2
Q1(3,3) = 0;        % Weight on state x3
Q1(4,4) = 0;
Q1(5,5) = 0;
Q1(6,6) = 0;
R1 = zeros(mu,mu);
R1(1,1) = q1;
R1(2,2) = q2;
G = gen_q(Q1,R1,N,M) ;
c = zeros(N*mx+M*mu,1);

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


%% Solve QP problem with unlinear constraint
tic
options =
optimoptions('fmincon','Algorithm', 'sqp', 'MaxFunEvals', 40000);

f = @(z) (1/2)*z'*G*z;

[z,ZVAL, EXITFLAG] = fmincon(f, z0, [], [], ...
Aeq, beq, vlb, vub, @constraints, options);
% Calculate
t1=toc;

%% Extract control inputs and states
n = N*mx+N*mu;
u1  = [z(N*mx+1:mu:n);z(n-1)];  % Optimal input trajectory
u2 = [z(N*mx+2:mu:n);z(n)];

x1 = [x0(1);z(1:mx:N*mx)];    % State x1 from solution
x2 = [x0(2);z(2:mx:N*mx)];    % State x2 from solution
x3 = [x0(3);z(3:mx:N*mx)];    % State x3 from solution
x4 = [x0(4);z(4:mx:N*mx)];    % State x4 from solution
x5 = [x0(5);z(5:mx:N*mx)];    % State x5 from solution
x6 = [x0(6);z(6:mx:N*mx)];    % State x6 from solution


num_variables = 5/delta_t;
```

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

u1   = [zero_padding; u1; zero_padding]; % Optimal input
u2   = [zero_padding; u2; 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];
x5  = [zero_padding; x5; zero_padding];
x6  = [zero_padding; x6; zero_padding];

x = [x1 , x2 , x3 , x4, x5, x6];
u = [u1, u2];
t = 0:delta_t:delta_t*(length(u1)-1);

% Simulation of inequality constraint
con = zeros(length(t));
time_out = t;

for i = 1:length(t)
    con(i) = alpha*exp(-beta*(x1(i)-lambda_t)^2);
end

%% Input imported to the helicopter
input = timeseries(u,t);
state = timeseries(x,t);

%% With feedback
Q_lqr = diag([10  5  1  1  20  5]);
R_lqr = diag([0.1 0.1]);
K = dlqr(A1,B1,Q_lqr,R_lqr);
```

Listing 4: MATLAB function ineqConstraints

```
function [c, ceq] = ineqConstraints(z)
    global N mx alpha beta lambda_t
    for k= 1:N
        c(k) = alpha*exp(-beta*(z(1+(k-1)*mx)-lambda_t)^2) - z(5+(k-1)*mx);
    end
    ceq = [];
end
```

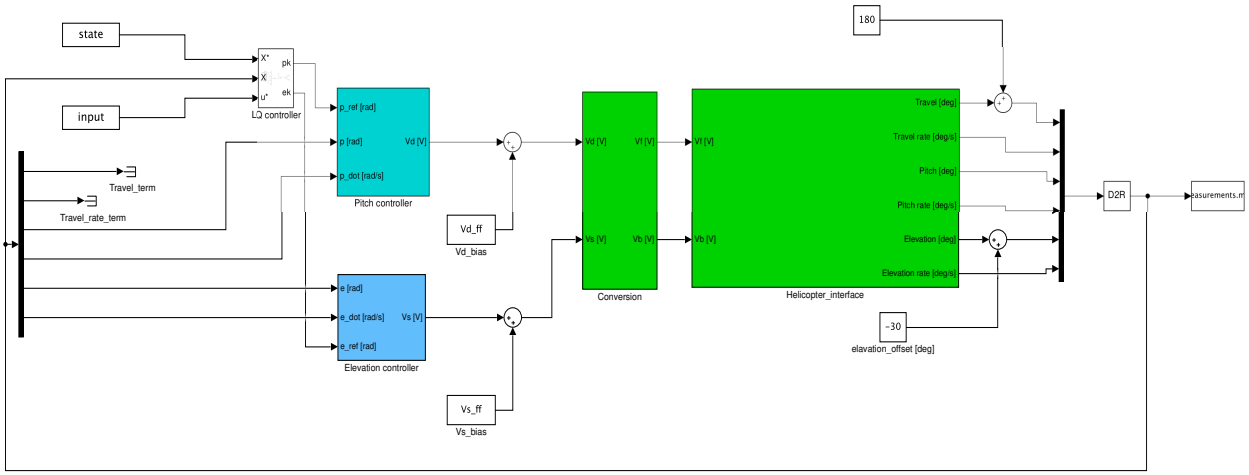


Figure 14: SIMULINK model 10.4

References

[1] Nocedal, J. and Wright, S. J.(2006) *Numerical Optimization(2.edition)*. New York: Springer

[2] Foss, B. and Heirung, T.A (2006) *Merging Optimization and Control*. Department of Engineering Cybernetics

[3] Egeland, O. and Gravdahl, T (2003) *Modeling and Simulation for Automatic Control*. Marine Cybernetics