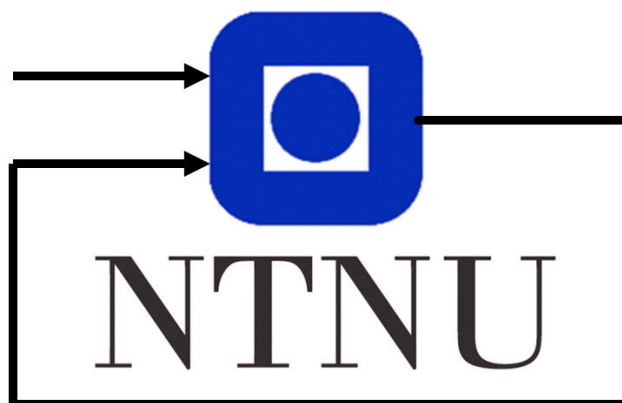


TTK 4135 Optimization and Control Helicopter Lab

Henrik Enger Reimers 552364
Erik Fougner Arnesen 598690

April, 2024



Department of Engineering Cybernetics

Contents

1	Introduction	1
1.1	Objective	1
1.2	System description	1
1.3	Derivation of mathematical model	2
1.3.1	Elevation model	2
1.3.2	Pitch model	2
1.3.3	Travel model	2
1.3.4	Summary of relevant equations	3
2	Optimal Control of Pitch/Travel without Feedback	4
2.1	The continuous model	4
2.2	The discretized model	4
2.3	Calculating the optimal path with quadratic programming	4
2.4	Results	6
3	Optimal Control of Pitch/Travel with Feedback (LQ)	8
3.1	LQ controller	8
3.2	Model Predictive Control	9
3.3	Results	9
3.4	MATLAB and Simulink	10
4	Optimal Control of Pitch/Travel and Elevation with Feedback	12
4.1	Expansion of the continuous model	12
4.2	The discretized model	12
4.3	Calculating the optimal path with a nonlinear constraint on elevation	12
4.4	Results	13
4.5	Decoupled model	15
4.6	Accounting for the connection between pitch and elevation	16
4.6.1	Rewriting the equation and state space representation	16
4.6.2	Adding additional terms to the objective function	16
4.6.3	Adjusting Q-weight in the LQ controller based on pitch angle	16
A	MATLAB	17
A.1	Initialization code	17
A.2	Optimal Control of Pitch/Travel without Feedback code	19
A.3	Optimal Control of Pitch/Travel with LQ control	22
A.4	Optimal Control of Pitch/Travel and Elevation with nonlinear constraint	25
B	SIMULINK diagrams	28
B.1	LAB 2	28
B.2	LAB 3	29
B.3	LAB 4	30
	References	31

1 Introduction

1.1 Objective

The objective of this lab is to utilize optimization techniques to control a system. The assignment specifies that a study on a 3- DOF helicopter is to be conducted. Given a mathematical model of the system, the goal is to study control behavior with different controllers. First we implement optimal control of pitch and travel, both with and without feedback. This was achieved by minimizing a quadratic optimization problem. The feedback was applied using a linear quadratic controller. In the last part of the assignment optimal control of pitch, travel and elevation with feedback was performed. A nonlinear inequality constraint was applied on the elevation during this task. *MATLAB* was utilized to calculate the optimal control parameters and *SIMULINK* was used to control the helicopter.

1.2 System description

The physical helicopter system consists of a base with an arm attached. The arm has a counterweight in one end and two rotors in the other end. The arm can rotate around the base (travel) and move up and down (elevation). A difference in voltage given to the rotors can be used to control rotation around an axis normal to the arm (pitch). These angles are measured using encoders. A cross- section of the helicopter can be seen in figure 1. The parameters and variables for the model can be seen in table 1 and 2.

Table 1: Parameters and values

Symbol	Parameter	Value	Unit
l_a	Distance from elevation axis to helicopter body	0.63	m
l_h	Distance from pitch axis to motor	0.18	m
K_f	Force constant motor	0.25	N/V
J_e	Moment of inertia for elevation	0.83	kg m ²
J_t	Moment of inertia for travel	0.83	kg m ²
J_p	Moment of inertia for pitch	0.034	kg m ²
mh	Mass of helicopter	1.05	kg
mw	Balance weight	1.87	kg
mg	Effective mass of the helicopter	0.05	kg
K_p	Force to lift the helicopter from the ground	0.49	N

Table 2: Variables

Symbol	Variable
p	Pitch
p_c	Setpoint for pitch
λ	Travel
r	Speed of travel
r_c	Setpoint for speed of travel
e	Elevation
e_c	Setpoint for elevation
V_f	Voltage, motor in front
V_b	Voltage, motor in back
V_d	Voltage difference, $V_f - V_b$
V_s	Voltage sum, $V_f + V_b$
$K_{pp}, K_{pd}, K_{ep}, K_{ed}$	Controller gains
T_g	Moment needed to keep the helicopter flying

1.3 Derivation of mathematical model

The equations for the mathematical model of the helicopter were derived using Newtons second law for rotation and moment balances. In combination with the sketch in figure 2 it is possible to derive a model of the elevation, pitch and travel.[1]

1.3.1 Elevation model

$$J\ddot{e} = l_a K_f V_s - T_g \quad (1a)$$

where

$$\ddot{e} = K_3 V_s - T_g J_e, \quad K_3 = \frac{l_a K_f}{J_e} \quad (1b)$$

The following PD- controller is used to control the elevation:

$$V_s = K_{ep}(e_c - e) - K_{ed}\dot{e}, \quad K_{ep}, K_{ed} > 0 \quad (1)$$

This results in the following system:

$$\ddot{e} = -K_3 K_{ed}\dot{e} + K_3 K_{ep}e - \frac{T_g}{J_e} + K_3 K_{ep}e_c \quad (2)$$

Finally an integral term is added resulting in an PID- controller. It is assumed that the integral term nullify the effect from the constant term $-\frac{T_g}{J_e}$. The model for the elevation is then:

$$\ddot{e} + K_3 K_{ed}\dot{e} + K_3 K_{ep}e = K_3 K_{ep}e_c \quad (3)$$

1.3.2 Pitch model

$$J_p \ddot{p} = K_f l_h V_d \quad (4)$$

where

$$\ddot{p} = K_1 V_d \quad K_1 = \frac{K_f l_h}{J_p} \quad (5)$$

The following PD - controller is used to control the pitch angle:

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p}, \quad K_{pp}, K_{pd} > 0 \quad (6)$$

The model for the pitch then results in:

$$\ddot{p} + K_1 K_{pd}\dot{p} + K_1 K_{pp}p = K_1 K_{pp}p_c \quad (7)$$

1.3.3 Travel model

$$J_t \ddot{r} = -K_p l_a \sin p \quad (8)$$

It is assumed that angle p is small, resulting in $\sin p = p$. Then,

$$\ddot{r} = K_2 p, \quad K_2 = \frac{K_p l_a}{J_t} \quad (9)$$

1.3.4 Summary of relevant equations

$$\ddot{e} = -K_3K_{ed}\dot{e} + K_3K_{ep}e - \frac{T_g}{J_e} + K_3K_{ep}e_c \quad (10)$$

$$\ddot{p} + K_1K_{pd}\dot{p} + K_1K_{pp}p = K_1K_{pp}p_c \quad (11)$$

$$\dot{\lambda} = r \quad (12)$$

$$\dot{r} = -K_2p \quad (13)$$

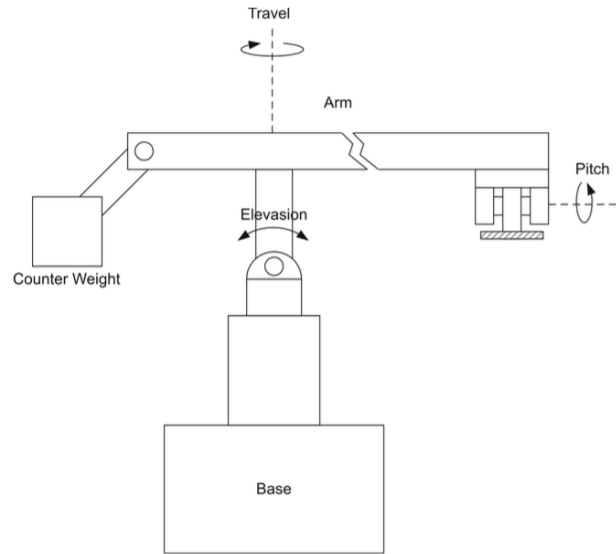


Figure 1: Cross-section of the helicopter. [1]

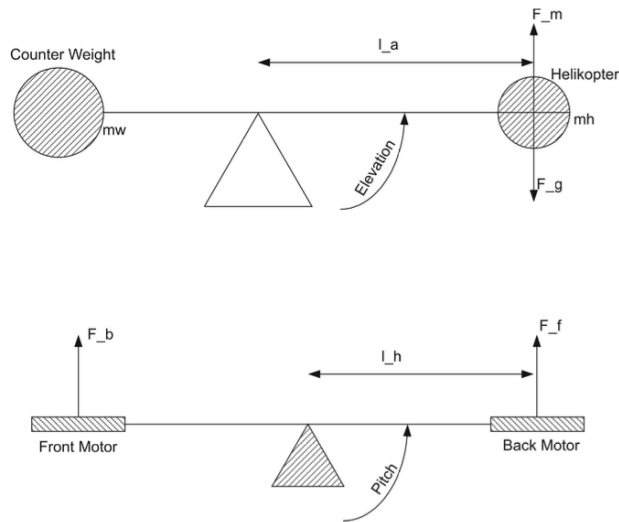


Figure 2: Sketch of the helicopter. [1]

2 Optimal Control of Pitch/Travel without Feedback

In the first part of the lab assignment the task is to calculate an optimal trajectory x^* and input sequence u^* . This is achieved by minimizing the given quadratic programming problem. The optimal input sequence was used to drive the helicopter from an initial starting position x_0 to a predefined reference point x_f . It is assumed that elevation is equal to zero.

2.1 The continuous model

The mathematical model derived in section 1.3 can be expressed in continuous state-space form:

$$\begin{aligned}\dot{x} &= A_c x + B_c u \\ x &= [\lambda \quad r \quad p \quad \dot{p}]^T \\ u &= p_c\end{aligned}\tag{14}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1 K_{pp} \end{bmatrix} p_c\tag{15}$$

2.2 The discretized model

In order to discretize the model, the forward Euler method is used. The method in state-space form is defined as the following:

$$x_{k+1} = \underbrace{(I - \Delta t A_c)}_{A_d} x_k + \underbrace{\Delta t B_c u_k}_{B_d}, \quad \Delta t = 0.25s\tag{16}$$

where

$$A_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \Delta t & 1 & 0 & 0 \\ 0 & -\Delta t K_2 & 1 & -\Delta t K_1 K_{pp} \\ 0 & 0 & \Delta t & 1 - \Delta t K_1 K_{pd} \end{bmatrix}, \quad B_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta t K_1 K_{pp} \end{bmatrix}\tag{17}$$

2.3 Calculating the optimal path with quadratic programming

To obtain the optimal trajectory the following objective function needs to be minimized:

$$\Phi = \sum_{i=1}^N (\lambda_i - \lambda_f)^2 + q p_{ci}^2, \quad q \geq 0\tag{18}$$

Using equation 33 it is possible to derive the Hessian matrix (Q):

$$Q = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\tag{19}$$

The optimal path is to be calculated from the initial starting point of $x_o = [\pi \quad 0 \quad 0 \quad 0]^T$ to the reference point $x_f = [0 \quad 0 \quad 0 \quad 0]^T$. To achieve this the following inequality constrained quadratic problem (QP) needs to be solved:

$$\min_{z \in R^n} f(z) = \frac{1}{2} z^T G z, \quad s.t. \quad A_{eq} z = b_{eq},\tag{20}$$

G , A_{eq} and b_{eq} is defined as:

$$G = \begin{bmatrix} Q & 0 & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & Q & \ddots & & \vdots \\ \vdots & & \ddots & q & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & q \end{bmatrix} \quad (21)$$

$$A_{eq} = \begin{bmatrix} I & 0 & \dots & \dots & 0 & -B & 0 & \dots & \dots & 0 \\ -A & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -A & I & 0 & \dots & \dots & 0 & -B \end{bmatrix}, \quad b_{eq} = \begin{bmatrix} Ax_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (22)$$

and inequality constraint on the pitch angle:

$$|p_k| \leq \frac{\pi}{6}, \quad k \in \{1, \dots, N\} \quad (23)$$

The optimization problem was solved using the quadprog function in matlab. First we had to load a initialization file that were handed out to us. This file contained all the necessary variables that were required for the model definition, like K_{pp} and K_1 . Next, we discretize the model as described in 2.2.

The next step is then to declare x_0 , N and a z vector containing all the variables that we want to optimize:

$$z = (x_1^T \dots, x_N^T, u_1^T \dots, u_{N-1}^T), \quad N = 100 \quad (24)$$

Several helper function was given to us to help with generation of various matrices. Firstly we utilized `gen_constraints` to produce constraints for the measurements and inputs. To fill in the Q and A_{eq} matrices the helper functions `gen_q` and `gen_aeq` was used.

Lastly we called the quadprog function wich solved the QP- problem and gave us the optimal input for the helicopter. The input values was then extracted and imported into our *SIMULINK* model, so we could compare the simulation in *MATLAB* with the physical system. Both the *MATLAB* code and the *SIMULINK* model can be found in the appendix

2.4 Results

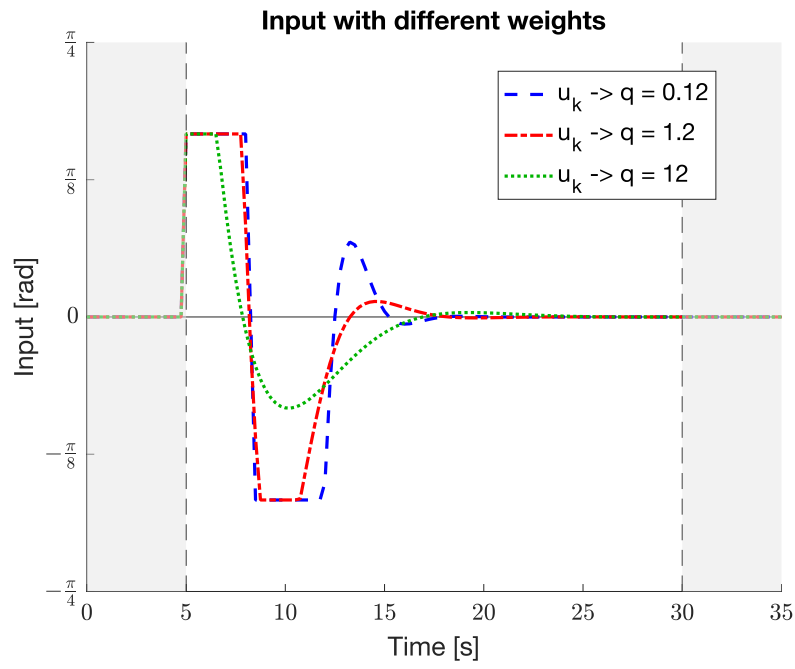


Figure 3: Optimized input for different weights.

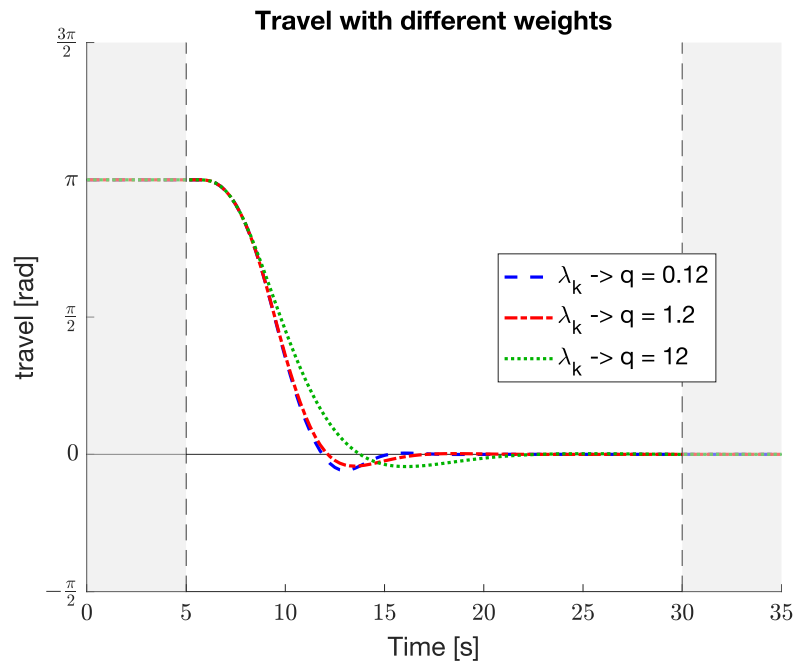


Figure 4: Plot of travel for different weights.

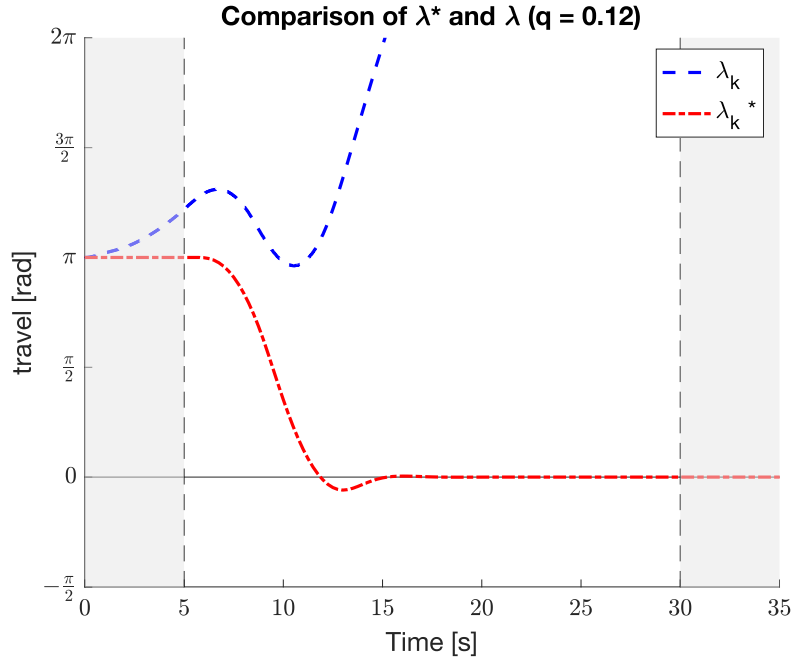


Figure 5: Optimal travel vs. recorded travel.

Increasing the weight q gave a smoother input sequence, and this lead to a more gentle control of the helicopter. It is possible to observe in figure 3 that $q = 12$ converges to 0 slowly. Decreasing q results in aggressive control, with sudden changes in the input and faster convergence to 0.

Looking at figure 5 it is possible to see that the helicopter never reaches the intended reference point x_f . This is because the controller in use doesn't rely on feedback, therefore it assumes stationary when it reaches the reference point. In practice the helicopter still has momentum and continues past x_f . Implementing a closed- loop solution can help remedy this overshoot.

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

3.1 LQ controller

An LQ controller is short for Linear-Quadratic controller, this is a type of optimal controller used to design control systems. It's goal is to minimize the quadratic criteria for a linear model [1]. Using the previously calculated optimal trajectory from exercise 2, we have the optimal input sequence u^* and the optimal trajectory x^* . Utilizing these variables we can integrate feedback into the control system

$$u_k = u_k^* - K(x_k - x_k^*) \quad (25)$$

Here the input sequence u^* is calculated using the systems ability to follow then calculated optimal trajectory. Whenever a deviation is observed from the optimal trajectory x^* , this will then effect the variable feedback back to the system. The final unknown variable K is the optimal gain matrix, which can be calculated using the MATLAB function *dlqr* that minimize the quadratic objective function.

$$J = \sum_{i=0}^{\infty} \Delta x_{i+1}^T Q \Delta x_{i+1} + \Delta u_i^T R \Delta u_i, \quad Q \geq 0, R \geq 0 \quad (26)$$

for the linear model

$$\Delta x_{i+1} = A \Delta x_i + B \Delta u_i \quad (27)$$

not including the inequality constraints. Here, Δx and Δu are deviations from the optimal trajectory

$$\Delta x = x - x^* \quad (28)$$

$$\Delta u = u - u^* \quad (29)$$

Here the **Q** and **R** are the weighted matrices. The **Q** matrix weighs the importance to minimize the deviations of the system's states from the optimal trajectory, while the **R** matrix weighs the importance to minimize the control effort.

$$Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix}, \quad R = r_1$$

When placing a large value in a specific position in the **Q** matrix such as q_1 , we place a larger importance on the state travel. This will then penalize any deviation from the corresponding optimal state more heavily, making the controller working harder to keep that state closer to the desired value. lower values on the other hand will lead to more freedom to deviation to the state. But in the **R** matrix case, we place importance to minimize the control effort. If we place a high value on r_1 we penalize more for using more control, making the controller more conservative on using control input.

In our case we want the controller to penalize deviation of travel and travel rate, while allowing states such as pitch and pitch rate to be more free. Our goal is to change the travel state to a specific value, therefore it is beneficial to be strict, ensuring a successful trajectory and stable final position. For **R** on the other hand we may want to have a lower value on the state to making the controller less conservative. After multiple tests we ended up with the **Q** and **R** matrix.

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \quad R = 0.1$$

The resulting optimal gain matrix K becoming

$$K = [-3.9054, -8.5146, 3.4289, 0.8503]$$

3.2 Model Predictive Control

Implementing a MPC, where the idea is to constantly updating the model predictions using every new measurment. To implement the MPC it would entail having to sole the QP problem at every timestep. Some of the advantages to MPC is that it takes into consideration constraint, such as physical limitation due to limits to actuator. As well as being intuitive and simple control policy as noted in [2]. Some Advantages that are also notable is MPC outperforming LQ in trajectory tracking, while comparably having a reduced actuator wear. This is due to MPC being characterized having smoother changes in control signals, noted in [3]. But there are some disadvantages to Model predictive control(MPC) as well. It will require having to solve the optimization problem at every time step. leading to needing larger computational resources by having a higher computational load, as noted in[2]. This is leading to MPC for fast real-time applications being limited, when compared to LQ due to only having solve the optimization problem once. Taking into account how fast the helicopter moves, and how fast the sensor readings that are being sent to the controller. In summation, MPC offers greater flexibility particularly in constrained, and multivariable systems, at the cost of higher computational requirements. LQR provides a simpler, computationally efficient for without taking into account constraints, making the choice between MPC and LQR dependent on the specific requirements of the control problem. implementing the MPC controller would potentially look something like in figure 6

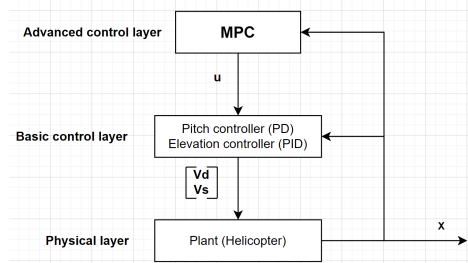


Figure 6: Diagram of MPC implementation

3.3 Results

When comparing the results from different experiments, we see that our LQR controller preforms very well in following the optimal trajectory in travel. When comparing to pitch it travels well in the start but suffers from steady state error at the end. The fact that the LQ controller with feedback tracks better then the controller without feedback should also come as a surprise.

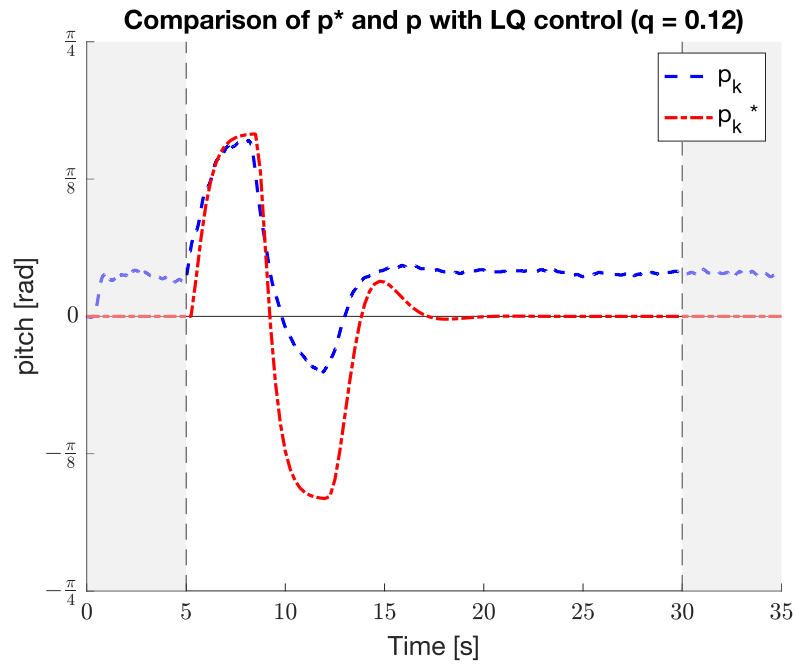


Figure 7: controller following optimal trajectory in pitch.

3.4 MATLAB and Simulink

Listing 1: MATLAB code for calculating the optimal gain matrix K

```

1  %% Q, R and K matrix
2  q_1 = 5;
3  q_2 = 1;
4  q_3 = 0.1;
5  q_4 = 0.1;
6  Q_matrix = diag([q_1,q_2,q_3,q_4]);
7
8  R = 0.1;
9  K = dlqr(A1,B1,Q_matrix,R);

```

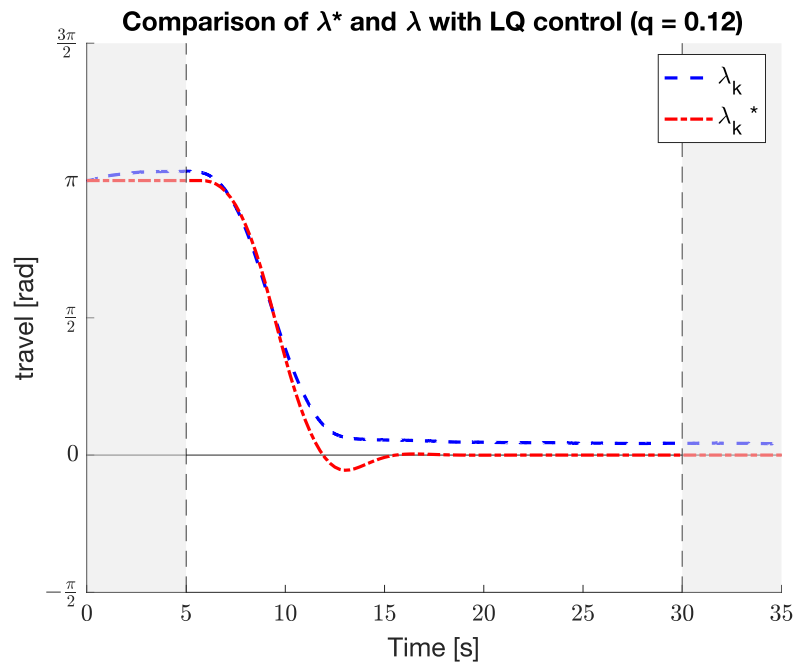


Figure 8: controller following optimal trajectory in travel.

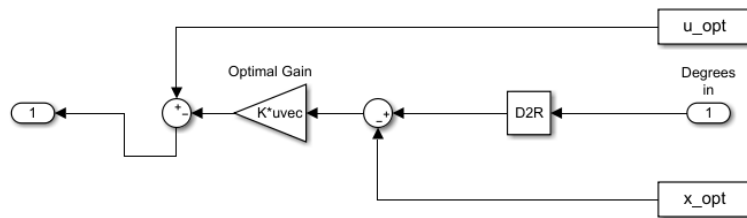


Figure 9: Implementation of LQ controller.

4 Optimal Control of Pitch/Travel and Elevation with Feedback

In the last section of this report the dynamics of elevation e is included. The goal is now to calculate an optimal path from x_0 to x_f past an obstacle which require a change in the elevation angle. This obstacle is implemented in the problem via a nonlinear constraint. This requires the use of a nonlinear solver to find the optimal path of the two dimensions.

4.1 Expansion of the continuous model

In order to accommodate the new conditions, two new states e and \dot{e} needs to be incorporated. A new input controlling the elevation is also included. The continuous state-space form is now expressed as the following:

$$\begin{aligned}\dot{x} &= A_c x + B_c u \\ x &= [\lambda \quad r \quad p \quad \dot{p} \quad e \quad \dot{e}]^T \\ u &= [p_c \quad e_c]^T\end{aligned}\tag{30}$$

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3 K_{ep} & -K_3 K_{ed} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3 K_{ep} \end{bmatrix} \begin{bmatrix} p_c \\ e_c \end{bmatrix}\tag{31}$$

4.2 The discretized model

In order to discretize the model, the forward Euler method described in equation 16 is used. where

$$A_d = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & -\Delta t K_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & -\Delta t K_1 K_{pp} & 1 - \Delta t K_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & -\Delta t K_3 K_{ep} & 1 - \Delta t K_3 K_{ed} \end{bmatrix}, \quad B_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & \Delta t K_3 K_{ep} \end{bmatrix}\tag{32}$$

4.3 Calculating the optimal path with a nonlinear constraint on elevation

Estimating the optimal path from $x_o = [\pi \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$ to $x_f = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$ now requires a minimization of the following objective function:

$$\phi = \sum_{i=1}^{N-1} (\lambda_{i+1} - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2, \quad q_1, q_2 \geq 0\tag{33}$$

$$Q = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\tag{34}$$

$$R = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}\tag{35}$$

a nonlinear constraint is also imposed on the elevation for every point in time:

$$c(x_k) = \alpha \exp(-(\lambda_k - \lambda_t)^2) - e_k \leq 0 \quad (36)$$

where $\alpha = 0.2$, $\beta = 20$, $\lambda_t = \frac{2\pi}{3}$

Listing 2: MATLAB code for calculating the nonlinear constraint for every timestep

```

1 function [c, ceq] = nl_constraints(z)
2     alpha = 0.2;
3     beta = 20;
4     lambda_t = (2*pi)/3;
5     nx = 6;
6     N = 40;
7     c = zeros(N,1);
8     for k = 1:N
9         c(k) = alpha*exp(-beta*(z(1+(k-1)*nx)-lambda_t)^2)
10             - z(5+(k-1)*nx);
11     end
12     ceq = [];
13 end

```

In the previous tasks the optimization problem was solved using quadratic programming, but in this case the problem includes a nonlinear constraint. Therefore we need to use the `fmincon` function to find the optimal path.

Listing 3: MATLAB code for solving the optimization problem using `fmincon`

```

1 %% Solve QP problem with fmincon
2
3 f = @(z) 1/2*z'*Q*z;
4 opt = optimoptions('fmincon','Algorithm','sqp','MaxFunEvals',
5     4000);
6 tic
7 [z, zval, exitflag] = fmincon(f,z0,[], [], Aeq, beq, vlb, vub
8     , @nl_constraints, opt);
9 toc

```

4.4 Results

Our weighting matrix focuses on penalizing travel and elevation. There is a higher emphasis on penalizing elevation as the main objective is to clear the obstacle. We also implemented a penalizing term on pitch rate as we observed that the helicopter changed the pitch to fast, causing the helicopter to crash in the ground. After a lot tests we ended up with the following Q and R matrix:

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}, \quad R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \quad (37)$$

The resulting optimal gain matrix is defined as:

$$K = \begin{bmatrix} -1.1643 & -3.4772 & 1.6596 & 0.6706 & \approx 0 & \approx 0 \\ \approx 0 & \approx 0 & \approx 0 & \approx 0 & 9.3768 & 8.0317 \end{bmatrix} \quad (38)$$

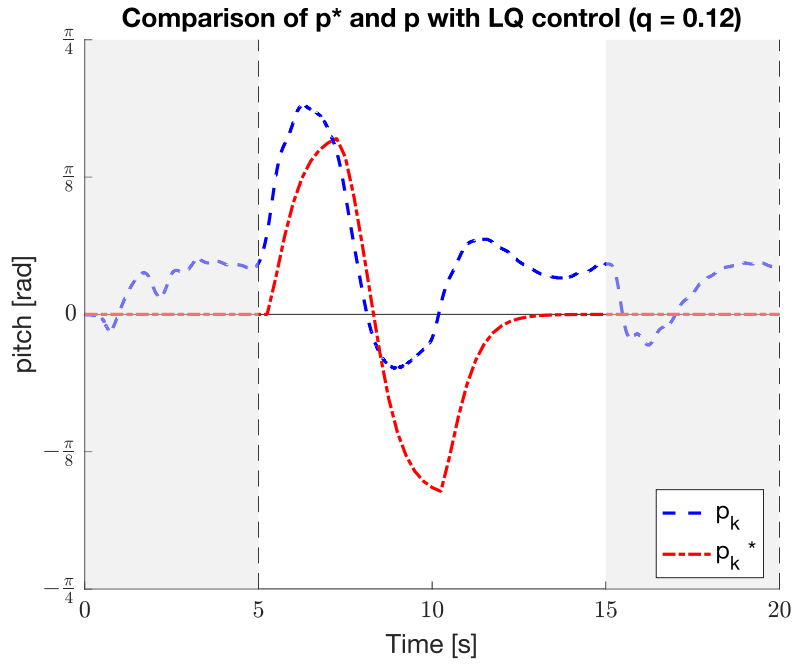


Figure 10: Optimal pitch vs. recorded pitch.

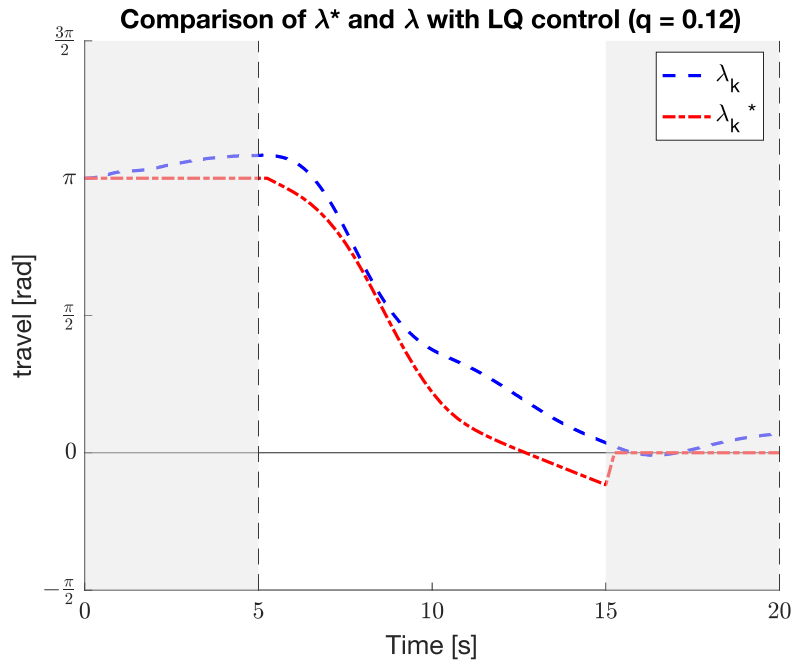


Figure 11: Optimal travel vs. recorded travel.

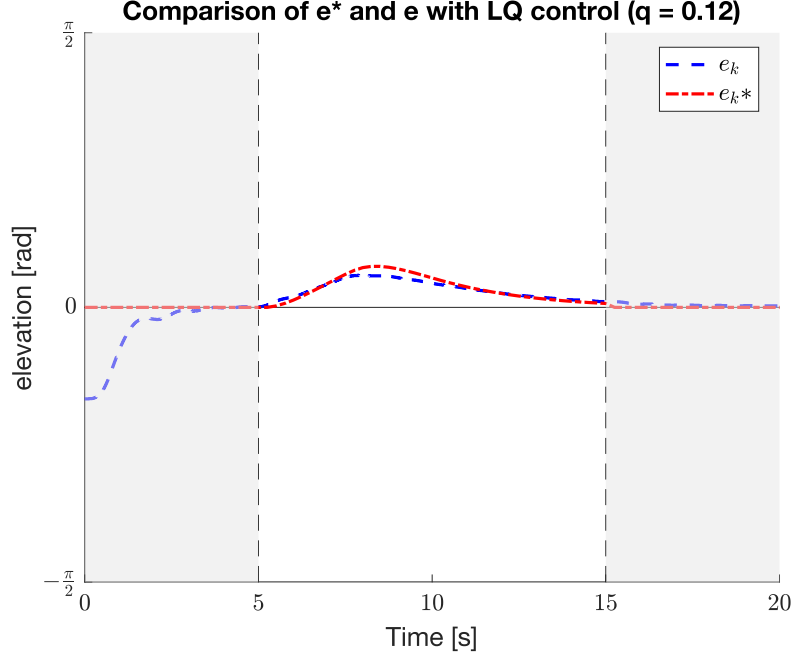


Figure 12: Optimal elevation vs. recorded elevation.

The weighting matrix resulted in an almost optimal elevation trajectory while having a less accurate trajectory for pitch and travel. This was expected as our aim was primarily to clear the simulated obstacle.

The expanded LQ- controller can be seen in the figure below:

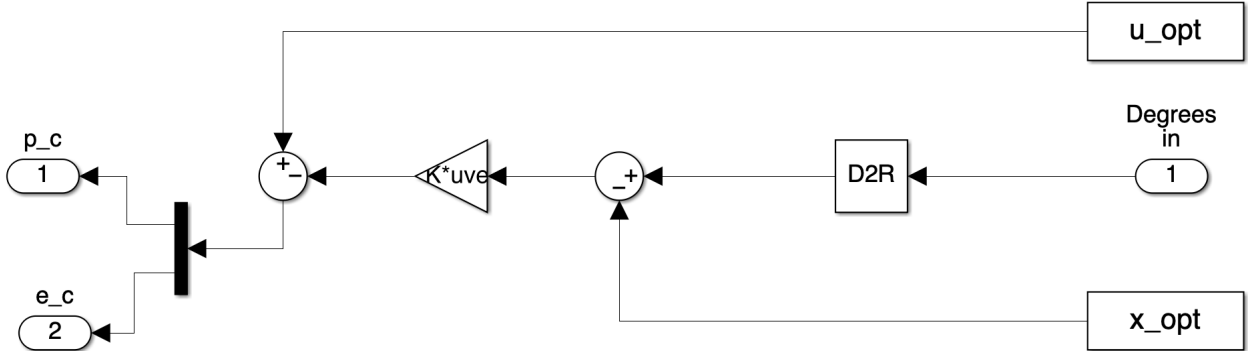


Figure 13: Final implementation of LQ controller in *SIMULINK*.

4.5 Decoupled model

In section 1.3 equation 1a states that the elevation is independent of pitch. During previous tests with the physical helicopter it is obvious that there exists some connection between these two states. This can be observed when p is set to $\frac{\pi}{2}$. Even though the pitch is changed significantly, the elevation remains fairly stable. The model doesn't declare a relationship between pitch and elevation dynamics, but there are two good reasons why it still works.

The first reason describes the robustness of LQ- controllers. Many control algorithms, including LQ- controllers, are inherently robust to small modeling inaccuracies or simplifications. They can still perform adequately even if the underlying model does not fully capture all the dynamics or interactions in the system. [4] In this case, the controller is able to stabilize the helicopter despite not specifically accounting for the relationship between pitch and elevation.

The second reason describes how feedback is able to contribute to stable LQ- control. The controller relies on feedback from the actual system to adjust and stabilize the helicopter's dynamics.

Even if the model does not explicitly include the connection between pitch and elevation, the controller can learn and adapt to these interactions through feedback during operation.

4.6 Accounting for the connection between pitch and elevation

4.6.1 Rewriting the equation and state space representation

This could be a good suggestion. By explicitly incorporating the sinusoidal relationship between pitch and elevation dynamics into the model, the controller can better capture the system's behavior and potentially improve performance.

4.6.2 Adding additional terms to the objective function

This could be a bad suggestion. While incorporating more terms into the objective function might seem like a way to account for the pitch-elevation connection, it could also introduce unnecessary complexity and potentially destabilize the controller.

4.6.3 Adjusting Q-weight in the LQ controller based on pitch angle

This could be a viable suggestion. Adjusting the weighting matrix Q in the LQR controller based on the pitch angle allows the controller to prioritize certain states or dynamics during different phases of flight. It could potentially improve performance by providing better control over critical states.

A MATLAB

A.1 Initialization code

```
1 % Initialization for the helicopter assignment in TTK4135.
2 % Run this file before you execute QuaRC -> Build.
3
4 % Updated spring 2018, Andreas L. Flaaten
5 % Updated Spring 2019, Joakim R. Andersen
6
7 clear all;
8 close all;
9 clc;
10
11 % The encoder for travel for Helicopter 2 is different from
    the rest.
12 travel_gain = 1; %
13 elevation_gain = 1; %
14
15 %% Physical constants
16 m_h = 0.4; % Total mass of the motors.
17 m_g = 0.03; % Effective mass of the helicopter.
18 l_a = 0.65; % Distance from elevation axis to helicopter body
19 l_h = 0.17; % Distance from pitch axis to motor
20
21 % Moments of inertia
22 J_e = 2 * m_h * l_a * l_a; % Moment of inertia for
    elevation
23 J_p = 2 * ( m_h/2 * l_h * l_h); % Moment of inertia for
    pitch
24 J_t = 2 * m_h * l_a * l_a; % Moment of inertia for
    travel
25
26 % Identified voltage sum and difference
27 V_s_eq = 7.45;%6.3; % Identified equilibrium voltage sum.
28 V_d_eq = 0.2;%0.35; % Identified equilibrium voltage
    difference.
29
30 % Model parameters
31 K_p = m_g*9.81; % Force to lift the helicopter from the
    ground.
32 K_f = K_p/V_s_eq; % Force motor constant.
33 K_1 = l_h*K_f/J_p;
34 K_2 = K_p*l_a/J_t;
35 K_3 = K_f*l_a/J_e;
36 K_4 = K_p*l_a/J_e;
37
38 %% Pitch closed loop syntesis
39 % Controller parameters
40 w_p = 1.8; % Pitch controller bandwidth.
41 d_p = 1.0; % Pitch controller rel. damping.
42 K_pp = w_p^2/K_1;
43 K_pd = 2*d_p*sqrt(K_pp/K_1);
44 Vd_ff = V_d_eq;
```

```

45
46 % Closed loop transfer functions
47 Vd_max = 10 - V_s_eq; % Maximum voltage difference
48 deg2rad = @(x) x*pi/180;
49 Rp_max = deg2rad(15); % Maximum reference step
50 s = tf('s');
51 G_p = K_1/(s^2);
52 C_p = K_pp + K_pd*s/(1+0.1*w_p*s);
53 L_p = G_p*C_p;
54 S_p = (1 + L_p)^(-1);
55
56 plot_pitch_response = 0;
57 if plot_pitch_response
58     figure()
59     step(S_p*Rp_max); hold on;
60     step(C_p*S_p*Rp_max/Vd_max);
61     legend('norm error', 'norm input')
62     title('Pitch closed loop response')
63 end
64
65 %% Elevation closed loop analysis
66 % Controller parameters
67 w_e = 0.5; % Elevation controller bandwidth.
68 d_e = 1.0; % Elevation controller rel. damping.
69 K_ep = w_e^2/K_3;
70 K_ed = 2*d_e*sqrt(K_ep/K_3);
71 K_ei = K_ep*0.1;
72 Vs_ff = V_s_eq;
73
74 % Closed loop transfer functions
75 Vs_max = 10 - V_s_eq; % Maximum voltage sum
76 Re_max = deg2rad(10); % Maximum elevation step
77 G_e = K_3/(s^2);
78 C_e = K_ep + K_ed*s/(1+0.1*w_e*s) + K_ei/s;
79 L_e = G_e*C_e;
80 S_e = (1 + L_e)^(-1);
81
82 plot_elev_response = 0;
83 if plot_elev_response
84     figure()
85     step(S_e*Re_max);
86     hold on;
87     step(C_e*S_e*Re_max/Vs_max);
88     legend('norm error', 'norm input')
89     title('Elevation closed loop response')
90 end

```

A.2 Optimal Control of Pitch/Travel without Feedback code

```

1 % TTK4135 - Helicopter lab
2 % Hints/template for problem 2.
3 % Updated spring 2018, Andreas L. Flaaten
4
5 %% Initialization and model definition
6 init04;
7
8 delta_t = 0.25;
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 0 0 K_1*K_pp]';
15
16 A_d = eye(4) + delta_t * Ac;
17 B_d = delta_t * Bc;
18
19 % Number of states and inputs
20 mx = size(A_d,2);
21 mu = size(B_d,2);
22
23 % Initial values
24 x1_0 = pi; % Lambda
25 x2_0 = 0; % r
26 x3_0 = 0; % p
27 x4_0 = 0; % p_dot
28 x0 = [x1_0 x2_0 x3_0 x4_0]'; % Initial values
29
30 % Time horizon and initialization
31 N = 100; % Time horizon for states
32 M = N; % Time horizon for inputs
33 z = zeros(N*mx+M*mu,1); % Initialize z vector
34 z0 = z; % Initial value for optimization
35
36 % Bounds
37 ul = -1/6*pi; % Lower bound on control
38 uu = 1/6*pi; % Upper bound on control
39
40 xl = -Inf*ones(mx,1); % Lower bound on states (no bound)
41 xu = Inf*ones(mx,1); % Upper bound on states (no bound)
42 xl(3) = ul; % Lower bound on state x3
43 xu(3) = uu; % Upper bound on state x3
44
45 % Generate constraints on measurements and inputs
46 [vlb,vub] = gen_constraints(N,M,xl,xu,ul,uu);
47 vlb(N*mx+M*mu) = 0; % Setting the last input to zero
48 vub(N*mx+M*mu) = 0; % Setting the last input to zero
49
50 % Generate the matrix Q and the vector c
51 Q1 = zeros(mx,mx);
52 Q1(1,1) = 2; % Weight on state x1

```

```

53 Q1(2,2) = 0; % Weight on state x2
54 Q1(3,3) = 0; % Weight on state x3
55 Q1(4,4) = 0; % Weight on state x4
56 P1 = 0.12; % Weight on input
57 Q = gen_q(Q1,P1,N,M);
58 c = zeros(1,5*N);
59
60 %% Generate system matrices for linear model
61 Aeq = gen_aeq(A_d, B_d, N, mx, mu);
62 beq = zeros(4*N,1);
63 beq(1:4,1) = A_d*x0;
64
65 %% Solve QP problem with linear model
66 tic
67
68 [z, lambda] = quadprog(Q, c, [], [], Aeq, beq, vlb, vub, x0);
69 t1=toc;
70
71 % Calculate objective value
72 phi1 = 0.0;
73 PhiOut = zeros(N*mx+M*mu,1);
74 for i=1:N*mx+M*mu
75     phi1=phi1+Q(i,i)*z(i)*z(i);
76     PhiOut(i) = phi1;
77 end
78
79 %% Extract control inputs and states
80 u = [z(N*mx+1:N*mx+M*mu);z(N*mx+M*mu)]; % Control input from solution
81
82 x1 = [x0(1);z(1:mx:N*mx)]; % State x1 from solution
83 x2 = [x0(2);z(2:mx:N*mx)]; % State x2 from solution
84 x3 = [x0(3);z(3:mx:N*mx)]; % State x3 from solution
85 x4 = [x0(4);z(4:mx:N*mx)]; % State x4 from solution
86
87 num_variables = 5/delta_t;
88 zero_padding = zeros(num_variables,1);
89 unit_padding = ones(num_variables,1);
90
91 u = [zero_padding; u; zero_padding];
92 x1 = [pi*unit_padding; x1; zero_padding];
93 x2 = [zero_padding; x2; zero_padding];
94 x3 = [zero_padding; x3; zero_padding];
95 x4 = [zero_padding; x4; zero_padding];
96
97 %% Plotting
98 t = 0:delta_t:delta_t*(length(u)-1);
99
100 figure(2)
101 subplot(511)
102 stairs(t,u),grid
103 ylabel('u')
104 subplot(512)
105 plot(t,x1,'m',t,x1,'mo'),grid

```

```
106 ylabel('lambda')
107 subplot(513)
108 plot(t,x2,'m',t,x2','mo'),grid
109 ylabel('r')
110 subplot(514)
111 plot(t,x3,'m',t,x3','mo'),grid
112 ylabel('p')
113 subplot(515)
114 plot(t,x4,'m',t,x4','mo'),grid
115 xlabel('tid (s)'),ylabel('pdot')
```

A.3 Optimal Control of Pitch/Travel with LQ control

```

1 % TTK4135 - Helicopter lab
2 % Hints/template for problem 2.
3 % Updated spring 2018, Andreas L. Flaaten
4
5 %% Initialization and model definition
6 init; % Change this to the init file corresponding to your helicopter
7
8 % Discrete time system model. x = [lambda r p p_dot]'
9 delta_t = 0.25; % sampling time
10 Ac = [0 1 0 0; 0 0 -K_2 0; 0 0 0 1; 0 0 -K_1*K_pp -K_1*K_pd];
11 Bc = [0; 0; 0; K_1*K_pp];
12
13 A1 = eye(4)+ delta_t * Ac;
14 B1 = delta_t * Bc;
15
16 % Number of states and inputs
17 mx = size(A1,2); % Number of states (number of columns in A)
18 mu = size(B1,2); % Number of inputs(number of columns in B)
19
20 % Initial values
21 x1_0 = pi; % Lambda
22 x2_0 = 0; % r
23 x3_0 = 0; % p
24 x4_0 = 0; % p_dot
25 x0 = [x1_0 x2_0 x3_0 x4_0]'; % Initial values
26
27 % Time horizon and initialization
28 N = 100; % Time horizon for states
29 M = N; % Time horizon for inputs
30 z = zeros(N*mx+M*mu,1); % Initialize z for the whole
    horizon
31 z0 = z; % Initial value for optimization
32
33 % Bounds
34 ul = -1/6*pi; % Lower bound on control
35 uu = 1/6*pi; % Upper bound on control
36
37 xl = -Inf*ones(mx,1); % Lower bound on states (no bound)
38 xu = Inf*ones(mx,1); % Upper bound on states (no bound)
39 xl(3) = ul; % Lower bound on state x3
40 xu(3) = uu; % Upper bound on state x3
41
42 % Generate constraints on measurements and inputs
43 [vlb,vub] = gen_constraints(N,M,xl,xu,ul,uu); % hint:
    gen_constraints
44 vlb(N*mx+M*mu) = 0; % We want the last input to be
    zero
45 vub(N*mx+M*mu) = 0; % We want the last input to be
    zero
46
47 % Generate the matrix Q and the vector c (objective function weights in
    the QP problem)

```



```

48 Q1 = zeros(mx,mx);
49 Q1(1,1) = 2; % Weight on state x1
50 Q1(2,2) = 0; % Weight on state x2
51 Q1(3,3) = 0; % Weight on state x3
52 Q1(4,4) = 0; % Weight on state x4
53 P1 = 0.12; % Weight on input
54 Q = gen_q(Q1,P1,N,M); % Generate Q, hint:
    gen_q
55 c = zeros(1,5*N); % Generate c, this is
    the linear constant term in the QP
56
57 %% Generate system matrixes for linear model
58 Aeq = gen_aeq(A1, B1, N, mx, mu); % Generate A, hint: gen_aeq
59 beq = zeros(4*N,1); % Generate b
60 beq(1:4,1) = A1*x0;
61
62 %% Solve QP problem with linear model
63 tic
64
65 [z, lambda] = quadprog(Q, c, [], [], Aeq, beq, vlb, vub, x0); % hint:
    quadprog. Type 'doc quadprog' for more info
66 t1=toc;
67
68 % Calculate objective value
69 phi1 = 0.0;
70 PhiOut = zeros(N*mx+M*mu,1);
71 for i=1:N*mx+M*mu
72     phi1=phi1+Q(i,i)*z(i)*z(i);
73     PhiOut(i) = phi1;
74 end
75
76 %% Extract control inputs and states
77 u = [z(N*mx+1:N*mx+M*mu);z(N*mx+M*mu)]; % Control input from solution
78
79 x1 = [x0(1);z(1:mx:N*mx)]; % State x1 from solution
80 x2 = [x0(2);z(2:mx:N*mx)]; % State x2 from solution
81 x3 = [x0(3);z(3:mx:N*mx)]; % State x3 from solution
82 x4 = [x0(4);z(4:mx:N*mx)]; % State x4 from solution
83
84
85 num_variables = 5/delta_t;
86 zero_padding = zeros(num_variables,1);
87 unit_padding = ones(num_variables,1);
88
89 u = [zero_padding; u; zero_padding];
90 x1 = [pi*unit_padding; x1; zero_padding];
91 x2 = [zero_padding; x2; zero_padding];
92 x3 = [zero_padding; x3; zero_padding];
93 x4 = [zero_padding; x4; zero_padding];
94
95 %% Q, R and K matrix
96 q_1 = 5;
97 q_2 = 1;

```

```

98 q_3 = 0.1;
99 q_4 = 0.1;
100 Q_matrix = diag([q_1,q_2,q_3,q_4]);
101
102 R = 0.1;
103 K = dlqr(A1,B1,Q_matrix,R);
104
105 %% Plotting
106 t = 0:delta_t:delta_t*(length(u)-1);
107 u_opt = timeseries(u,t);
108 x = [x1, x2, x3, x4];
109 x_opt = timeseries(x,t);
110
111 figure(2)
112 subplot(511)
113 stairs(t,u),grid
114 ylabel('u')
115
116 subplot(512)
117 plot(t,x1,'m',t,x1,'mo'),grid, hold on;
118 %plot(t_test,lambda_test), grid
119 ylabel('lambda')
120
121 subplot(513)
122 plot(t,x2,'m',t,x2,'mo'),grid, hold on;
123 %plot(t_test,r_test), grid
124 ylabel('r')
125
126 subplot(514)
127 plot(t,x3,'m',t,x3,'mo'),grid, hold on;
128 %plot(t_test, p_test), grid
129 ylabel('p')
130
131 subplot(515)
132 plot(t,x4,'m',t,x4,'mo'),grid, hold on;
133 %plot(t_test, pdot_test), grid
134 xlabel('tid (s)'),ylabel('pdot')

```

A.4 Optimal Control of Pitch/Travel and Elevation with nonlinear constraint

```

1 % TTK4135 - Helicopter lab
2 % Updated spring 2018, Andreas L. Flaaten
3
4 %% Initialization and model definition
5 init; % Change this to the init file corresponding to your helicopter
6
7 % Discrete time system model. x = [lambda r p p_dot]'
8 delta_t = 0.25; % sampling time
9
10 Ac = [0      1      0      0      0      0;
11       0      0     -K_2      0      0      0;
12       0      0      0      1      0      0;
13       0      0  -K_1*K_pp  -K_1*K_pd  0      0;
14       0      0      0      0      0      1;
15       0      0      0      0     -K_3*K_ep -K_3*K_ed];
16
17 Bc = [0 0; 0 0; 0 0; K_1*K_pp 0; 0 0; 0 K_3*K_ep];
18
19 A1 = eye(6)+ delta_t * Ac;
20 B1 = delta_t * Bc;
21
22 % Number of states and inputs
23 nx = size(A1,2);
24 mu = size(B1,2);
25 N = 40; % Time horizon for states
26 M = N; % Time horizon for inputs
27 n = N*nx+N*mu;
28 z = zeros(N*nx+M*mu,1); % Initialize z for the whole
    horizon
29 z0 = z; % Initial value for optimization
30
31 %% SQP parameters
32 q1 = 0.1;
33 q2 = q1;
34 alpha = 0.2;
35 beta = 20;
36 lambda_t = 2*pi/3;
37
38 %% Initial values
39 x1_0 = pi; % Lambda
40 x2_0 = 0; % r
41 x3_0 = 0; % p
42 x4_0 = 0; % p_dot
43 x5_0 = 0; % e
44 x6_0 = 0; % e_dot
45 x0 = [x1_0 x2_0 x3_0 x4_0 x5_0 x6_0]'; % Initial values
46 %% Bounds
47 ul = -1/6*pi; % Lower bound on control
48 uu = 1/6*pi; % Upper bound on control
49
50 x1 = -Inf*ones(nx,1); % Lower bound on states (no bound)
51 xu = Inf*ones(nx,1); % Upper bound on states (no bound)

```

```

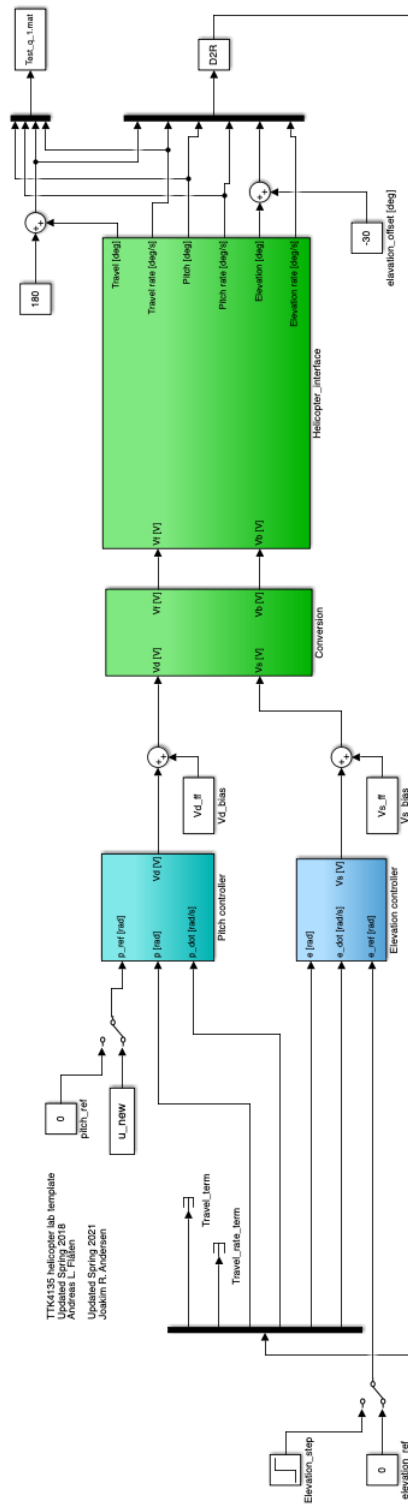
52 x1(3)      = ul;                % Lower bound on state x3
53 xu(3)      = uu;                % Upper bound on state x3
54
55 % Generate constraints on measurements and inputs
56 [vlb,vub]   = gen_constraints(N,M,xl,xu,ul,uu);
57 vlb(N*nx+M*mu) = 0;
58 vub(N*nx+M*mu) = 0;
59
60 %% Generate the matrix Q and the vector c (objective function weights in
    the QP problem)
61 Q1 = zeros(nx,nx);
62 Q1(1,1) = 2;                    % Weight on state x1
63 Q1(2,2) = 0;                    % Weight on state x2
64 Q1(3,3) = 0;                    % Weight on state x3
65 Q1(4,4) = 0;                    % Weight on state x4
66 Q1(5,5) = 0;
67 Q1(6,6) = 0;
68 P = zeros(mu,mu);
69 P(1,1) = 1;                    % Weight on input 1
70 P(2,2) = 1;                    % Weight on input 2
71
72 Q = gen_q(Q1,P,N,M);
73 c = zeros(1,5*N);
74
75 %% Generate system matrixes for linear model
76 Aeq = gen_aeq(A1, B1, N, nx, mu);
77 beq = zeros(size(Aeq,1),1);
78 beq(1:nx) = A1*x0;
79
80 %% Solve QP problem with fmincon
81
82 f = @(z) 1/2*z'*Q*z;
83 opt = optimoptions('fmincon','Algorithm','sqp','MaxFunEvals', 4000);
84 tic
85
86 [z, zval, exitflag] = fmincon(f,z0,[], [], Aeq, beq, vlb, vub, @
    nl_constraints, opt);
87 toc
88
89 %% Extract control inputs and states
90 u1 = [z(N*nx+1:mu:n); z(n-1)]; % Control input from solution
91 u2 = [z(N*nx+2:mu:n); z(n)];
92
93 x1 = [x0(1);z(1:nx:N*nx)];      % State x1 from solution
94 x2 = [x0(2);z(2:nx:N*nx)];      % State x2 from solution
95 x3 = [x0(3);z(3:nx:N*nx)];      % State x3 from solution
96 x4 = [x0(4);z(4:nx:N*nx)];      % State x4 from solution
97 x5 = [x0(5);z(5:nx:N*nx)];      % State x5 from solution
98 x6 = [x0(6);z(6:nx:N*nx)];      % State x6 from solution
99
100 num_variables = 5/delta_t;
101 zero_padding = zeros(num_variables,1);
102 unit_padding = ones(num_variables,1);

```

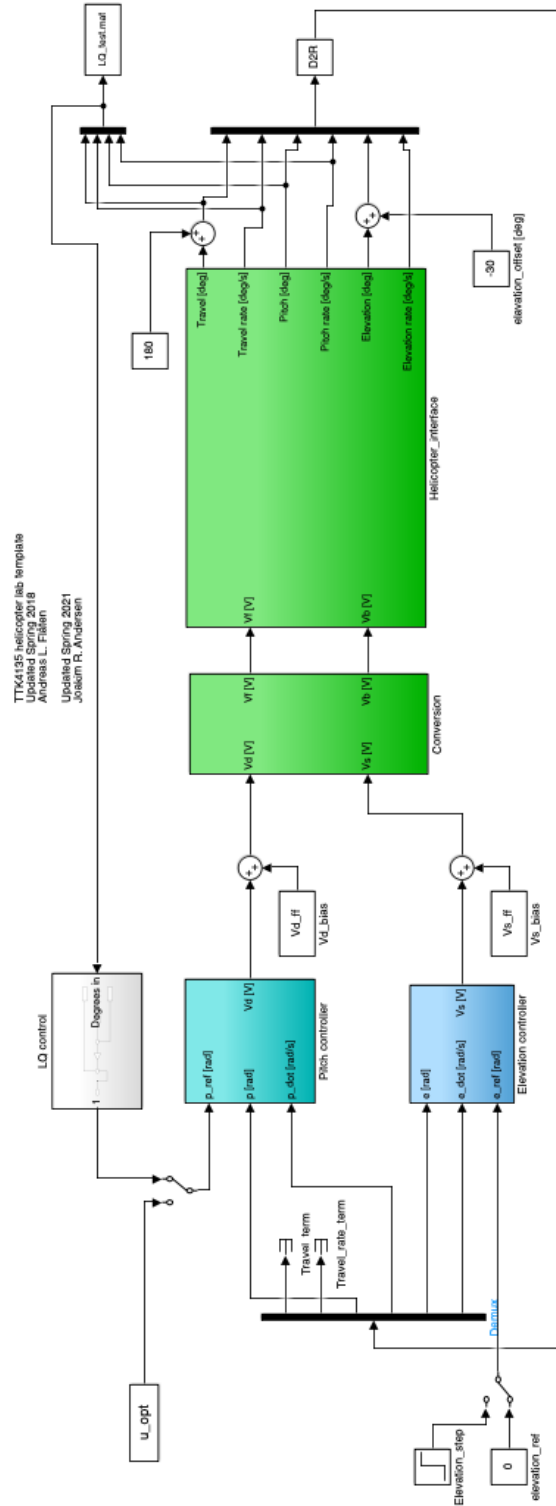
```
103 |  
104 | u1    = [zero_padding; u1; zero_padding];  
105 | u2    = [zero_padding; u2; zero_padding];  
106 | x1    = [pi*unit_padding; x1; zero_padding];  
107 | x2    = [zero_padding; x2; zero_padding];  
108 | x3    = [zero_padding; x3; zero_padding];  
109 | x4    = [zero_padding; x4; zero_padding];  
110 | x5    = [zero_padding; x5; zero_padding];  
111 | x6    = [zero_padding; x6; zero_padding];
```

B SIMULINK diagrams

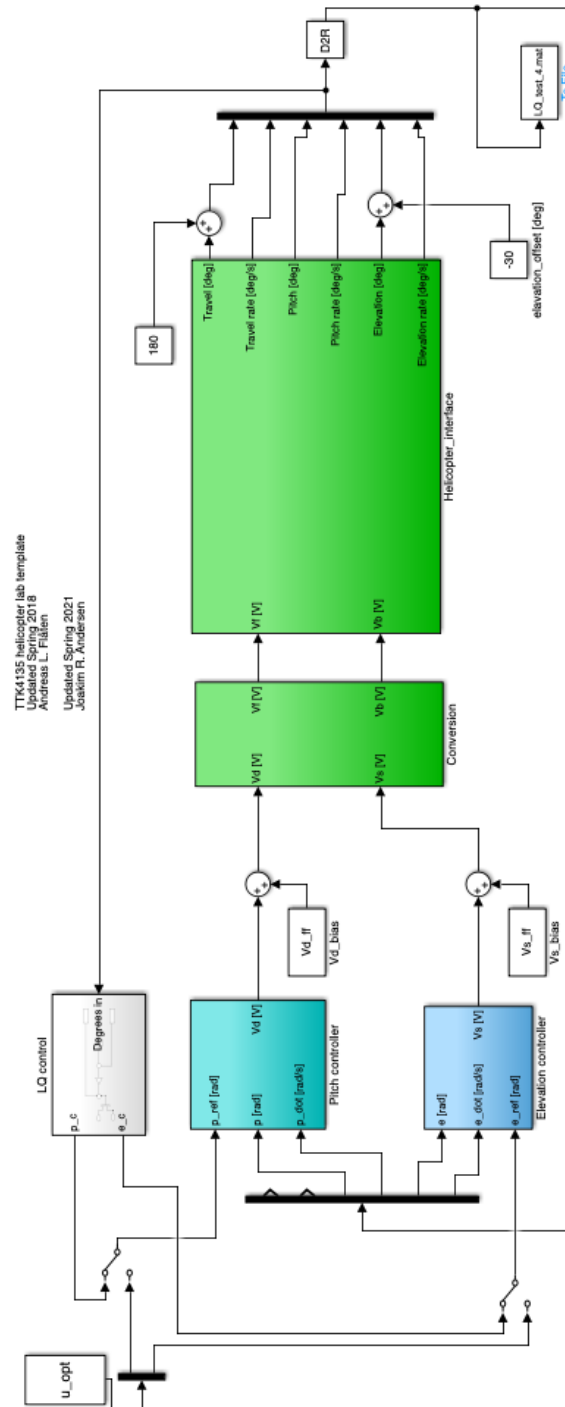
B.1 LAB 2



B.2 LAB 3



B.3 LAB 4



References

- [1] Norwegian University of Science and Technology Department of Engineering Cybernetics. TTK4135 Optimization and Control Helicopter Lab. 2023.
- [2] Jan Dentler. *Real-time Model Predictive Control of Cooperative Aerial Manipulation*. PhD thesis, 10 2018.
- [3] Andrzej Jezierski, Jakub Mozaryn, and Damian Suski. A comparison of lqr and mpc control algorithms of an inverted pendulum. 07 2017.
- [4] N. Lehtomaki, N. Sandell, and M. Athans. Robustness results in linear-quadratic gaussian based multivariable control designs. *IEEE Transactions on Automatic Control*, 26(1):75–93, 1981.