Helicopter lab
TTK4135 Optimization and Control

Group 87
478451
768572
478456
478448

March 15, 2019

Department of Engineering Cybernetics

Abstract

In this experiment, the actuation of a helicopter arm with three degrees of freedom was optimized to reach a certain position, given equality and inequality constraints. The optimal actuation is calculated using MATLAB methods, and are then applied to the helicopters actuators using QuaRC[1]. For each task, the model of the helicopter is extended to include more states and/or add inequality and equality constraints. Feedback control is obtained by using a Linear-Quadratic Controller.

Contents

Introduction **1**

1 Problem Description **2**

2 Part II **4**
2.1 Continuous Time State Space Form 4
2.2 Discretization of Model 4
2.3 Optimal Trajectory and Deviation of Desired Point 5
2.4 Results and Discussion 7

3 Part III **10**
3.1 Optimal Controller with Feedback 10
3.2 MPC Discussion 12

4 Part IV **14**
4.1 Continuous State Space with Elevation 14
4.2 Discretization of the Model with Elevation 14
4.3 Optimal Control of Pitch, Travel and Elevation with and without Feedback 15
4.4 Results with and without Feedback 16
4.4.1 Without Feedback 17
4.4.2 With Feedback 17
4.5 Decoupling 22
4.6 Additional Constraints 22

5 Conclusion **25**

Appendix **26**

A MATLAB Code **26**
A.1 Part 2 - Feed-forward 26
A.2 Part 3 - LQR 29
A.3 Part 4 - Inequality constraint 29
A.3.1 Function for computing the nonlinear constraint . . . 34

B Simulink Diagrams **35**
B.1 Part 2 - Optimal input directly as reference for pitch 35
B.2 Part 3 - LQ controller with feedback 35
B.3 Part 4 - LQ controller with and without feedback 36

References **38**

Introduction

Formulating and implementing dynamic optimization problems are essential knowledge in a lot of industrial applications. The approach is similar for different applications, and it is in this report shown for a helicopter with three degrees of freedom, connected to a rigid body.
A formulation of the dynamic optimization problems is obtained using a discretized model of the helicopters equation of motion with a PID-controller for the elevation, and a PD-controller for the pitch.
The effects of different types of controllers have been studied using a mathematical simplified model of the helicopter. First, optimal control of pitch and travel was done, both with and without feedback. This was achieved by minimizing a quadratic optimization problem. A linear quadratic controller was used when implementing the feedback control. Later, a non-linear constraint was added to the elevation. Optimal control of pitch, travel and elevation with and without feedback was calculated and achieved.
In section 1 the lab setup and the problem description is presented. The mathematical model that is used in every part is also included. In section 2 a continuous and discretized time state space model is derived. The optimization problem is calculated in MATLAB and the helicopter model was implemented using Simulink. Section 3 introduces linear quadratic control feedback. In section 4 two new states are introduced, as elevation and elevation rate is now considered in the model. In addition, the inequality constraints on the elevation is non-linear.

1 Problem Description

The goal of the lab is to achieve optimal control of the helicopter modeled in fig. 1 with respect to given constraints. To do so, a mathematical model of the helicopter is necessary. The assignment [3] included the following model:

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

where $K_1, K_2, K_3, K_{ed}, K_{ep}, K_{pd}$ and K_{pp} are constants, p, e and λ represent the joints of the system, and e_c and p_c are setpoint variables.

To stabilize the helicopter both at the beginning and at the end of each input sequence, paddings of zero input were added. These paddings are illustrated in the plots by a faded green colour.

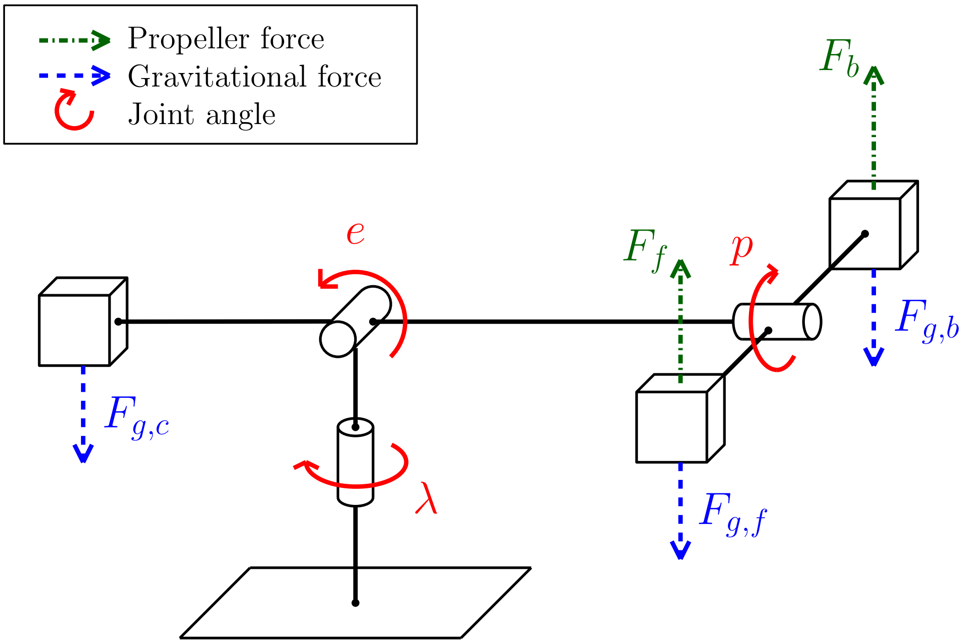


Figure 1: Lab setup. [Image credit: [2]].

Lab Setup

The helicopter consists of an arm attached to a rigid base. The helicopter is fixed to one end of the arm, and a counterweight is fixed to the other side.

The arm can be moved in three directions as seen in fig. 1. The arm can rotate around the vertical axis (travel), it can move up and down around an elevation axis (elevation) and around the axis normal to the frame (pitch).

There are two actuators on the helicopter, both connected on the same side of the arm. The forces from the actuator can be seen in fig. 1 as F_f and F_b. The forces are assumed to be proportional to the voltage applied, and the counter balance is calibrated such that a voltage of approximately 1.5 V to each motor will move the helicopter to an elevation such that the helicopter arm is perpendicular to the elevation axis.

2 Part II

In this part of the exercise the elevation has been disregarded, that is $e = 0$. Then the optimal trajectory x^* and a corresponding optimal input sequence u^* is calculated.

2.1 Continuous Time State Space Form

The continuous time state space form is derived from eq. (1) with $\mathbf{x} = [\lambda\ r\ p\ \dot{p}]^\top$ and $u = p_c$. The derived model is given by

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\lambda} \\ \dot{r} \\ \dot{p} \\ \ddot{p} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -k_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -k_1 k_{pp} & -k_1 k_{pd} \end{bmatrix}}_{A_c} \mathbf{x} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ k_1 k_{pp} \end{bmatrix}}_{B_c} u. \tag{2}$$

The model stated in eq. (1) is based on both the physics of the helicopter and the controllers associated with pitch and elevation, that is the PI-controller and the PID-controller. Further, the model is simplified by assuming that $e = 0$. This leads to eq. (2). The simplified model includes pitch, pitch rate, travel and travel rate. It models the two lower layers of fig. 2. The PID is not included in the simplified model, since it controls the elevation e, which is set to zero. The optimal input sequence will therefore only base itself on the plant and the PD controller.

2.2 Discretization of Model

The discretization is done using the forward Euler method, leading to eq. (3). Written out, this yields eq. (4).

$$\begin{aligned} \mathbf{x_{n+1}} &= \mathbf{x_n} + h\dot{\mathbf{x}}|_{\mathbf{x}=\mathbf{x_n}} \\ &= \mathbf{x_n} + h(A_c\mathbf{x_n} + B_c u_n) \\ &= (\mathbf{I} + hA_c)\mathbf{x_n} + hB_c u_n, \end{aligned} \tag{3}$$

$$\mathbf{x_{n+1}} = \underbrace{\begin{bmatrix} 1 & h & 0 & 0 \\ 0 & 1 & -k_2 h & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & -k_1 k_{pp} h & 1 - k_1 k_{pd} h \end{bmatrix}}_{A_d} \mathbf{x_n} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ h k_1 k_{pp} \end{bmatrix}}_{b_d} u. \tag{4}$$

where h a time step of 0.25 s.

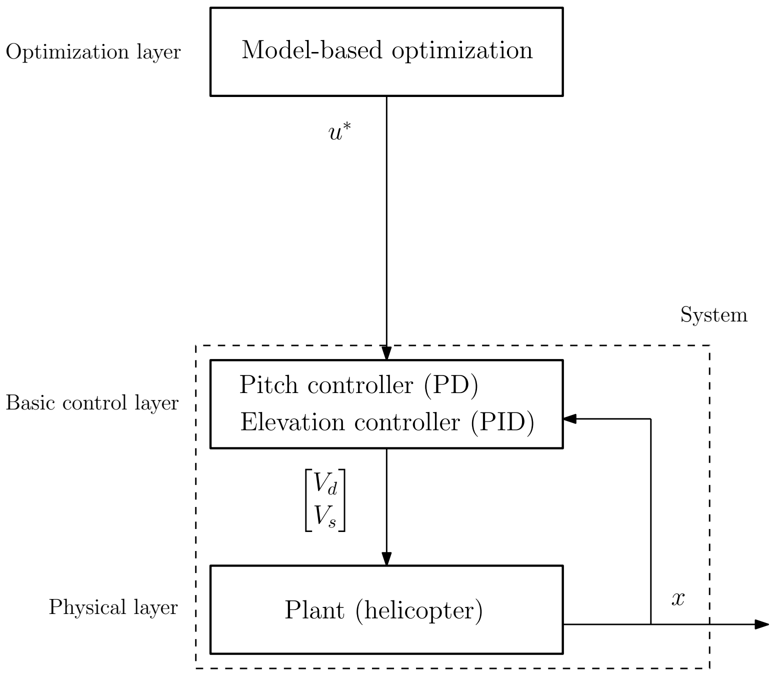


Figure 2: Layers in the control hierarchy with open loop. The two lower levels are modeled by eq. (2). The figure is based on [3].

2.3 Optimal Trajectory and Deviation of Desired Point

The task is to calculate an optimal trajectory for moving the helicopter from $\mathbf{x_0} = [\lambda_0 \ 0 \ 0 \ 0]^\top$ to $\mathbf{x_f} = [\lambda_f \ 0 \ 0 \ 0]^\top$, where $\lambda_0 = \pi$ and $\lambda_f = 0$. Since the position sensors on the helicopter are relative, a value of π is added to the measurement of the travel. This is done in order to make the helicopter start in $\mathbf{x_0}$. Further, the optimization is done with the use of the following objective function,

$$\phi = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q p_{ci}^2, \quad q \geq 0. \tag{5}$$

To solve this optimization problem in MATLAB, it is needed to present the problem on standard form,

$$\phi = \frac{1}{2} \sum_{i=1}^{N} x^\top Q x + u^\top R u. \tag{6}$$

Comparing eq. (5) and eq. (6), it can be seen that the Q matrix can be found as,

$$Q = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

while the input weight R is defined as q. Next, construction of matrices such that the state and actuation for each time step is contained within these matrices is done. It is desired to find Z, G, A_{eq} and b_{eq} such that

$$\min_{z} f(z) = \frac{1}{2} z^T G z \quad \text{s.t.} \tag{8a}$$

$$A_{eq} z = b_{eq}, \tag{8b}$$

is equivalent to eq. (6). Further, we want to add a constraint on the actuation, the pitch, of the form

$$|u_k| = |p_k| \leq \frac{30\pi}{180}, \quad k \in \{1, ..., N\}, \tag{9}$$

where $N = 100$. We assign $z = \left[x_1^T, \cdots, x_N^T, u_0^T, \cdots, u_{N-1}^T \right]^T$. We can then see that the matrix G is defined as

$$G = \begin{bmatrix} Q & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & Q & \ddots & & \vdots \\ \vdots & & \ddots & q & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & q \end{bmatrix}. \tag{10}$$

In MATLAB, the G matrix was created using the handed-out function *qen_q*. Further, the model of the helicopter must be implemented as constraints, which can be expressed as the matrices A_{eq} and b_{eq}. The matrices have the following form

$$A_{eq} = \begin{bmatrix} I & 0 & \cdots & \cdots & 0 & -B_d & 0 & \cdots & \cdots & 0 \\ -A_d & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -A_d & I & 0 & \cdots & \cdots & 0 & -B_d \end{bmatrix}, \tag{11a}$$

$$B_{eq} = \begin{bmatrix} A_d x_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{11b}$$

It can easily be verified that each row of $A_{eq}z = b_{eq}$ contains the constraints for each time step. The matrix A_{eq} was implemented in MATLAB by using another handed-out function, *gen_aeq*.

The solution of the stated QP in eq. (8) was found using the MATLAB function *quadprog*, which is a solver for finding the minimum of a quadratic objective function with linear constraints. The whole MATLAB script is shown in A.1 and the SIMULINK implementation is shown in fig. 16.

2.4 Results and Discussion

A problem using eq. (5) could arise if we were to steer the helicopter to $\lambda = \lambda_f$ with an insufficiently short time horizon. Minimizing the cost function does not explicitly ensure that $\lambda = \lambda_f$, it only ensures that the expression is as small as possible given the manipulated variables. If N is not selected sufficiently large, then the optimal path might not be to stabilize the helicopter at $\lambda = 0$. For instance, the optimal path could be to get to $\lambda = 0$ as quickly as possible, and then just continue past it. This unwanted phenomenon can be removed by selecting N large enough, e.g. above 100, where the optimal trajectory then will include steering λ to λ_f.

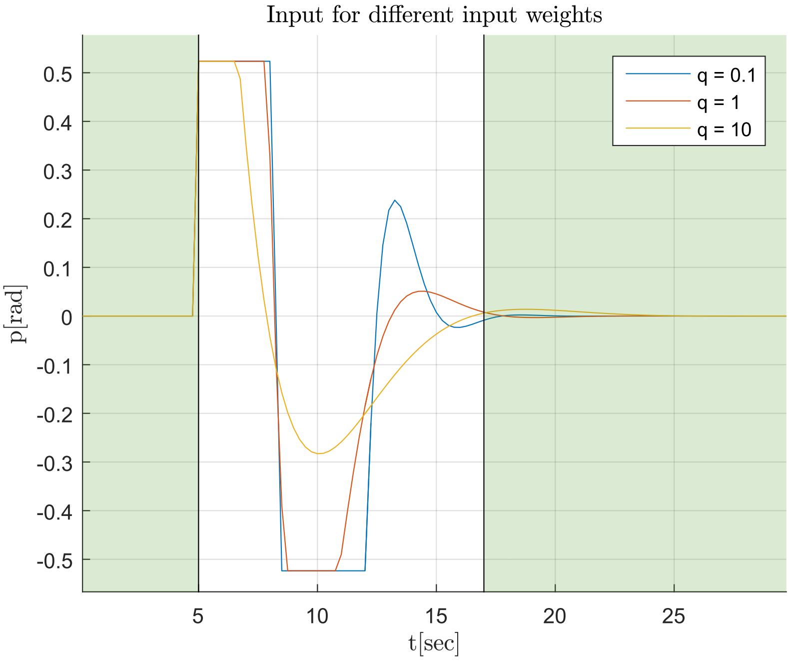


Figure 3: Input sequences for different values of q.

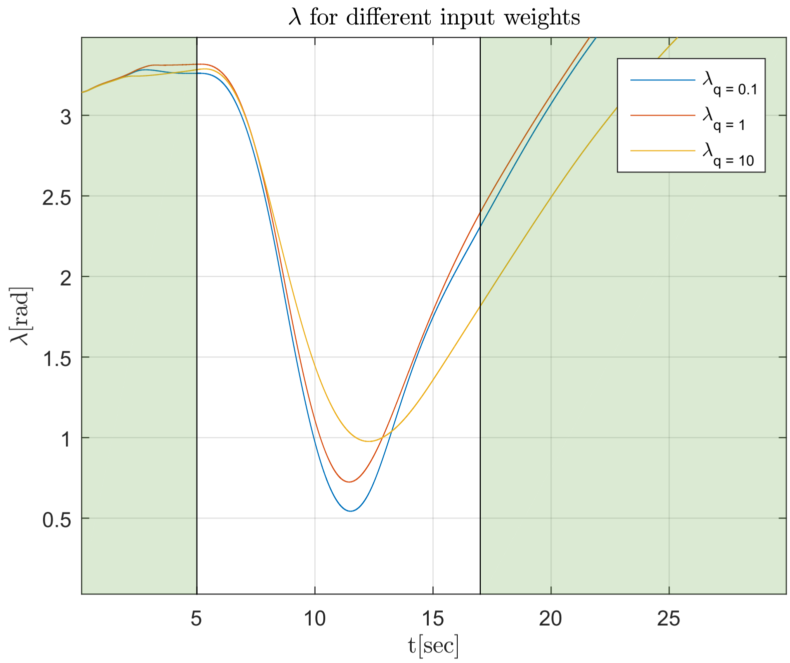


Figure 4: The measured travel plotted for different values of q.

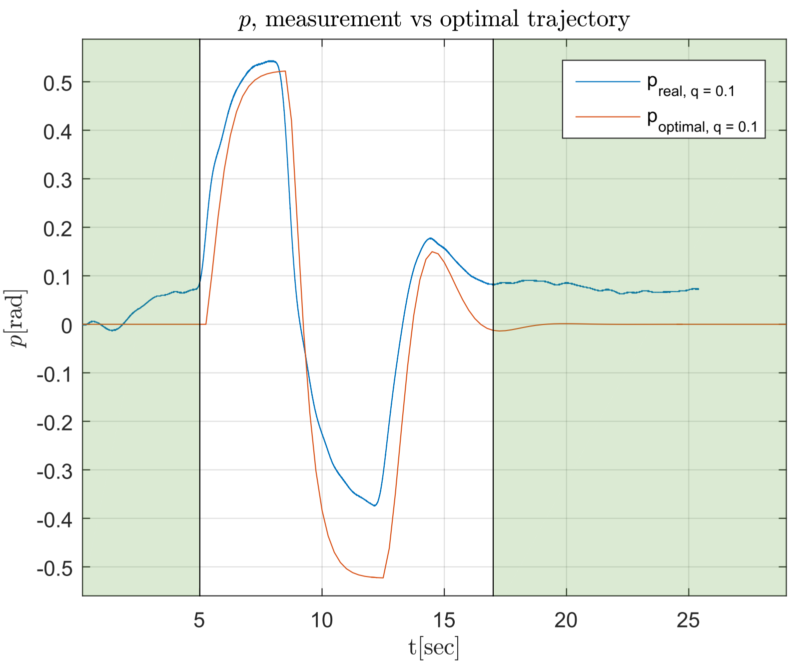


Figure 5: The optimal and measured pitch plotted for $q = 0.1$.

As shown in fig. 4, the helicopter does not reach the desired point x_f regardless of how the parameter q is chosen. This is mainly due to the dynamics of the PD-controller. By looking at fig. 3, the optimal input sequences consists of parts where the input changes instantly. From a physical perspective, this is not achievable. Based on the controller's dynamics, the optimization yields an optimal trajectory for the pitch, and the result is shown in fig. 5. However, it can be seen that the pitch does not manage to follow this trajectory perfectly, ultimately hindering the helicopter from reaching x_f. An offset error can also be seen in fig. 5, which leads to further deviations between the pitch and the optimal trajectory.

Moreover, the system model is linearized and simplified, resulting in an imperfect model. This will contribute to deviations between the actual and the optimal trajectory. It should be stated that the constraint on the pitch and the fact that e is set to 0 will minimize the deviations resulting from the linearization.

Lastly, according to eq. (1), the travel is controlled by integrating the pitch twice. Disturbances and deviations will get integrated, thus leading to further deviations.

As shown in fig. 4 the helicopter will not manage to station itself at a constant angle. A reason for this is the correlation between the helicopter's dynamics and the cost function. By setting $\lambda_i = \lambda_f$ in eq. (5), it is observed that the cost function is reduced to qp_{ci}^2. The cost function is then minimized by setting $p_{ci} = 0$. As a result of this, the input is set to zero after the reference is met, as shown in fig. 3 after $t \approx 20sec$, ultimately halting the controller. Due to the helicopters motion deviating from the optimal motion, this will in practice lead to the helicopter drifting away from the reference, as the inertia is not zero when the controller thinks λ_f is reached. An option to prevent this drifting would be to implement feedback, as this would drastically increase the system's robustness.

Ultimately, the best response was achieved using $q = 0.1$, as seen in fig. 4. This is because the use of input is penalized less, and the controller will thus follow the optimal trajectory better as compared to $q = 1$ or $q = 10$.

3 Part III

In this part, a feedback in the optimal controller is introduced. The feedback is implemented using a LQ controller, which finds the optimal gain by minimizing a quadratic cost function.

3.1 Optimal Controller with Feedback

Feedback will now be implemented such that if a deviation from the optimal trajectory x^* occurs, the input will be modified by the feedback control. A new input variable, which allows feedback control, is therefore introduced as

$$u_k = u_k^* - K^\top (x_k - x_k^*). \tag{12}$$

Observe from eq. (12) that if the system is following the optimal trajectory ($x_k = x_k^*$), the newly introduced input variable u_k is unchanged from before.

However, a good gain matrix K is needed in order to weight the trajectory deviation well. In this task, the gain matrix is calculated as a LQ controller, which minimizes the following infinite horizon quadratic objective function

$$J = \sum_{i=0}^{\infty} \Delta x_{i+1}^\top Q \Delta x_{i+1} + \Delta u_i^\top R \Delta u_i, \quad Q \geq 0, R > 0, \tag{13}$$

where

$$\Delta x_{i+1} = A \Delta x_i + B \Delta u_i, \tag{14}$$

and

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

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

It is worth noticing that the minimization problem is stated without inequality constraints. This is a pre-condition for the derivation of a LQ controller. Further, a general property of the Ricatti equation is that its solution will settle on a stationary value, given enough iterations. This property is used when deriving the optimal gain matrix K. The input that minimizes eq. (13) is given as

$$\Delta u_k = -K \Delta x_t, \tag{16}$$

where the optimal gain matrix K is given by

$$K = R^{-1} B_d^T P (I + B_d R^{-1} B_d^T P)^{-1} A_d. \tag{17}$$

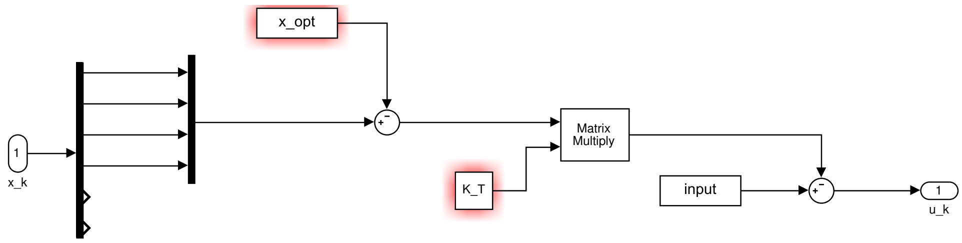


Figure 6: SIMULINK implementation of feedback.

The P matrix is defined as the positive semi-definite solution of the stationary Ricatti equation

$$P = Q + A_d^T P(I + B_d R^{-1} B_d^T P)^{-1} A_d \tag{18a}$$

$$P = P^T \succeq 0 \tag{18b}$$

In practice, the optimal gain matrix K was calculated using the *dlqr* function in MATLAB. The Q matrix and R matrix were chosen to be

$$Q = \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}, \quad R = 1 \tag{19}$$

which represents a fairly aggressive control. Deviations in travel were penalized significantly, while more freedom was given to pitch and pitch rate. The input weight was set to the default value of 1. The values were chosen with the purpose of ensuring that the travel would follow the optimal travel trajectory fairly well, but still ignoring the oscillatory behaviour that occurs around 12 seconds in fig. 7. The chosen matrices resulted in the following K matrix

$$K = \begin{bmatrix} -2.9353 & -6.6523 & 2.7542 & 0.6918 \end{bmatrix}. \tag{20}$$

In SIMULINK, the feedback was implemented using *from workspace* blocks in order to gain access to x^* and u^*. The SIMULINK implementation of the feedback was further based on matrix multiplication, and is shown in fig. 6. The code for the LQ controller shown in A.2 and the SIMULINK implementation for this task is shown in fig. 17.

By comparing the responses in fig. 4 with the response in fig. 7, it can be stated that the feedback offers a tremendous improvement. In fig. 7 the travel reaches the reference and the drift-off is completely removed. A stationary deviation is present due to an offset error, where the same error occurs both at the start and at the end of the flight.

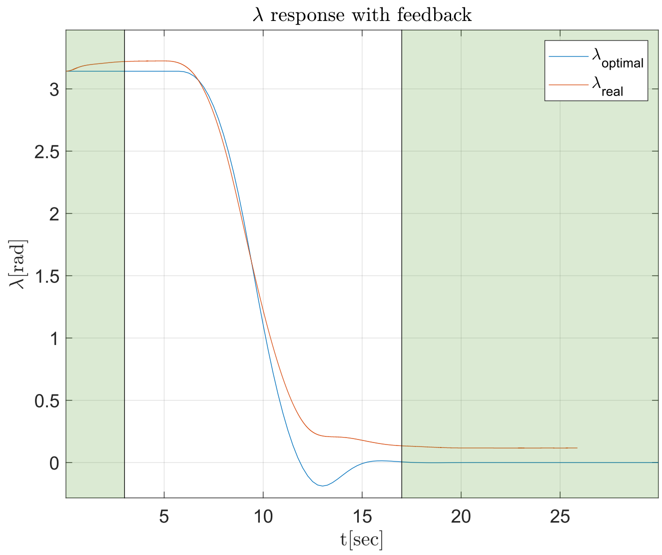


Figure 7: Comparison between optimal and measured λ with feedback.

3.2 MPC Discussion

An alternative strategy to a LQ controller is to use an MPC controller. In order to realize the MPC controller, the QP would be recomputed at every time step. The current state of the plant is used as the initial value, thus including a feedback control. At every time step the optimization yields an optimal control sequence, and the first input in this sequence is applied to the plant. How the MPC controller would fit in the control hierarchy is shown in fig. 8.

An advantage of using MPC would be reduction of deviations from disturbances and model errors. Since no model is perfectly accurate, and disturbances occur, the actual trajectory will differ from the optimal trajectory. This unwanted effect is reduced by the feedback control in the MPC, where the controller will calculate a new optimal trajectory at every time step based on the current state.

The LQ controller, on the other hand, will use only one optimal trajectory and then try to continuously correct deviations from that specific trajectory. As a result of this, disturbances and model errors could lead to a somewhat oscillatory behaviour. While the LQ controller tries to correct itself back to the optimal trajectory it may under- or overshoot, and thus behave less rigid than the MPC.

A disadvantage of the MPC controller is that it is computational heavy. This is especially the case if the time horizon in the QP is long. The QP is recomputed at every time step, something which requires considerable resources. The LQ controller however, only solves the QP once. The optimal

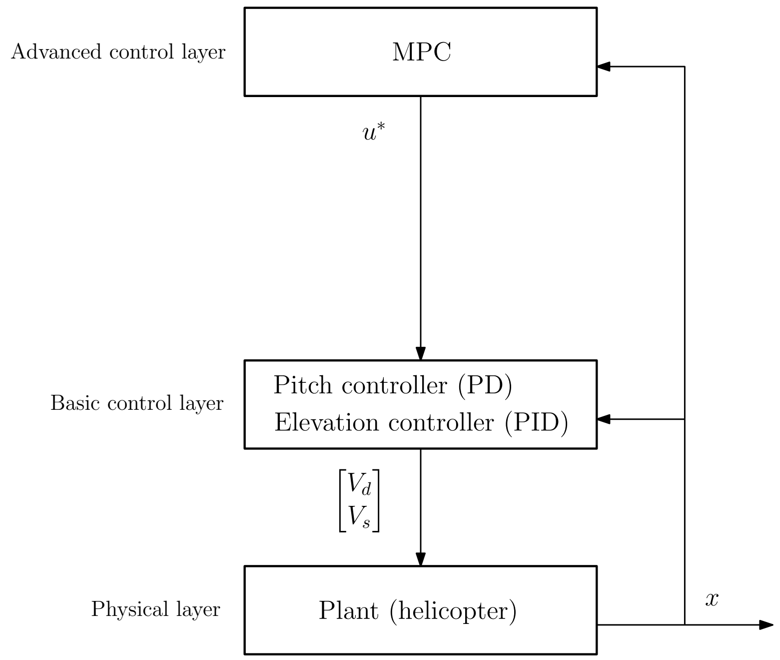


Figure 8: Layers in the control hierarchy with MPC. The figure is based on [3].

gain is then stored and used throughout the whole sequence. This gives the advantage that the optimal gain could be calculated in advance of application and independently of the states.

Our system has relatively fast dynamics, and a MPC may therefore not yield a considerably better performance. This is due to the MPC being a rather slow form of controller, where a substantially amount of decision variables must be computed at every time step. To match the dynamics of the system, a faster and less computational heavy form of controller may be preferred.

4 Part IV

In this part, an optimal trajectory for both elevation and travel is desired. The model is therefore extended to include the two new states e and $\dot{e}$. Further, the helicopter will be moved from an initial point to a reference point while avoiding an obstacle. The obstacle is implemented as a nonlinear constraint for the elevation, more on this in section 4.3. Because of this constraint, it is now needed to use a nonlinear solver to solve the optimization problem.

4.1 Continuous State Space with Elevation

Firstly, a new continuous state space model is required. With elevation implemented, the new states and input are $\mathbf{x} = [\lambda \ r \ p \ \dot{p} \ e \ \dot{e}]^\top$ and $\mathbf{u} = [p_c \ e_c]^\top$, respectively. The complete continuous state space model is

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\lambda} \\ \dot{r} \\ \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -k_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -k_1 k_{pp} & -k_1 k_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -k_3 k_{ep} & -k_3 k_{ed} \end{bmatrix}}_{A_c} \mathbf{x} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ k_1 k_{pp} & 0 \\ 0 & 0 \\ 0 & k_3 k_{ep} \end{bmatrix}}_{B_c} \mathbf{u}. \quad (21)$$

4.2 Discretization of the Model with Elevation

The discretization of the continuous state space model eq. (21) is done by using the forward Euler method, as was done in section 2.2. The expression for $\mathbf{x}_{t+1}$ is already derived in eq. (3). With the additional states for elevation control, this results in the following discrete state space system

$$
\mathbf{x}_{n+1} = \begin{bmatrix} \dot{\lambda} \\ \dot{r} \\ \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & h & 0 & 0 & 0 & 0 \\ 0 & 1 & -hk_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & h & 0 & 0 \\ 0 & 0 & -hk_1k_{pp} & 1-hk_1k_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & h \\ 0 & 0 & 0 & 0 & -hk_3k_{ep} & 1-hk_3k_{ed} \end{bmatrix}}_{A_d} \mathbf{x}_n
$$

$$
+ \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ hk_1k_{pp} & 0 \\ 0 & 0 \\ 0 & hk_3k_{ep} \end{bmatrix}}_{B_d} \mathbf{u}. \tag{22}
$$

4.3 Optimal Control of Pitch, Travel and Elevation with and without Feedback

It is desired to move the helicopter from $\mathbf{x_0} = [\lambda_0 \ 0 \ 0 \ 0 \ 0 \ 0]^\top$ to $\mathbf{x_f} = [\lambda_f \ 0 \ 0 \ 0 \ 0 \ 0]^\top$, similarly as presented in section 2.3 with $\lambda_0 = \pi$ and $\lambda_f = 0$. To find the optimal path for this, the following objection function is minimized

$$
\begin{aligned}
\phi &= \frac{1}{2}\sum_{i=1}^{N}(\lambda_i - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2, \quad N = 40 \\
&= \sum_{i=1}^{N} \mathbf{x}_{i+1}^\top Q \mathbf{x}_{i+1} + \mathbf{u}_i^\top R \mathbf{u}_i,
\end{aligned} \tag{23}
$$

where the input weights are

$$
Q = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}, \quad q_1, q_2 > 0.
$$

The constraint for elevation is given as

$$
e_k \geq \alpha e^{-\beta(\lambda_k - \lambda_t)^2}, \quad \forall k \in \{1, ..., N\}, \tag{24}
$$

with $\alpha = 0.2$, $\beta = 20$ and $\lambda_k = \frac{2\pi}{3}$.

Observe that this constraint is nonlinear, hence a QP solver cannot solve this. Therefore the function *fmincon* in MATLAB is used. The implementation is similar to *quadprog*, that was used earlier, but it takes nonlinear constraints in addition to the linear constraints. The following code was used to calculate z

```
object_fun = @(z) (1/2*z'*Q*z);
z0(1:mx) = x0;
opt = optimoptions('fmincon', 'Algorithm', ...
    'sqp', 'MaxFunEvals', 40000);

tic
[z, ZVAL, EXITFLAG] = fmincon(object_fun, z0, [], [], ...
    Aeq, beq, vlb, vub, @NONLNCON, opt);
t1=toc;
```

Listing 1: Calculating z with nonlinear constraint.

The function's nonlinear input in listing 1 is a function reference. It was therefore necessary to create a function which calculates the nonlinear elevation constraint. Global variables was used to pass the constraint to our solver at every iteration. The nonlinear constraint function is defined as

```
function [C, Ceq] = NONLNCON(z)
    global alpha beta lambda_t mx N
    C = zeros(N,1);
    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
```

Listing 2: Nonlinear elevation constraint

After calculating the optimal states and inputs, they were extracted and imported to SIMULINK. The complete code is shown in A.3 and the two SIMULINK implementations, with and without feedback, are shown in fig. 19 and fig. 20.

4.4 Results with and without Feedback

The responses of elevation, pitch and travel are shown in fig. 9, fig. 10 and fig. 11. Every plot is divided into two subplots, with and without feedback.

This is done for the reader to more easily notice the difference between the responses with and without feedback.

4.4.1 Without Feedback

The results can be seen in fig. 9a, fig. 10a and fig. 11a. Notice that the helicopter follows the optimal path somewhat well, especially for pitch and elevation. However, it is visible that the system is affected by drift, which especially applies for travel. This is due to the nature of open loop control. There is no feedback to control the drifting. The system fails to achieve its goal to follow the optimal path.

Observe that the pitch angle follows the optimal pitch path closely. This is because pitch is the input. It is also the reason why the elevation doesn't fulfill the constraint. The helicopter tilts too much, without regarding how this affects elevation, even though a strict elevation constraint is applied.

4.4.2 With Feedback

The feedback is implemented similarly as in section 3. The results can be seen in fig. 9b, fig. 10b and fig. 11b. The following weights were applied in the closed loop system

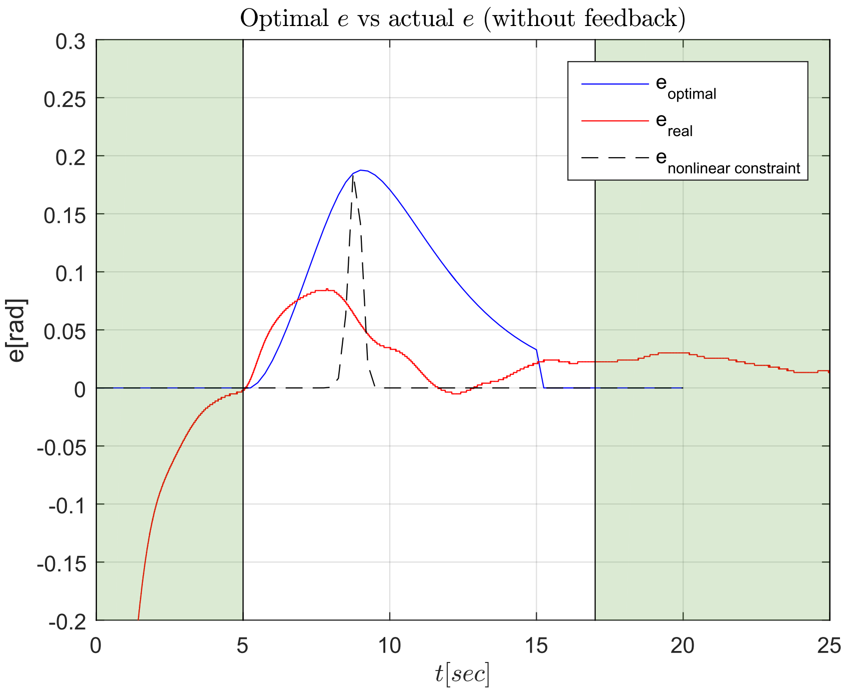
$$Q = \begin{bmatrix} 30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{25}$$

and the gain matrix is

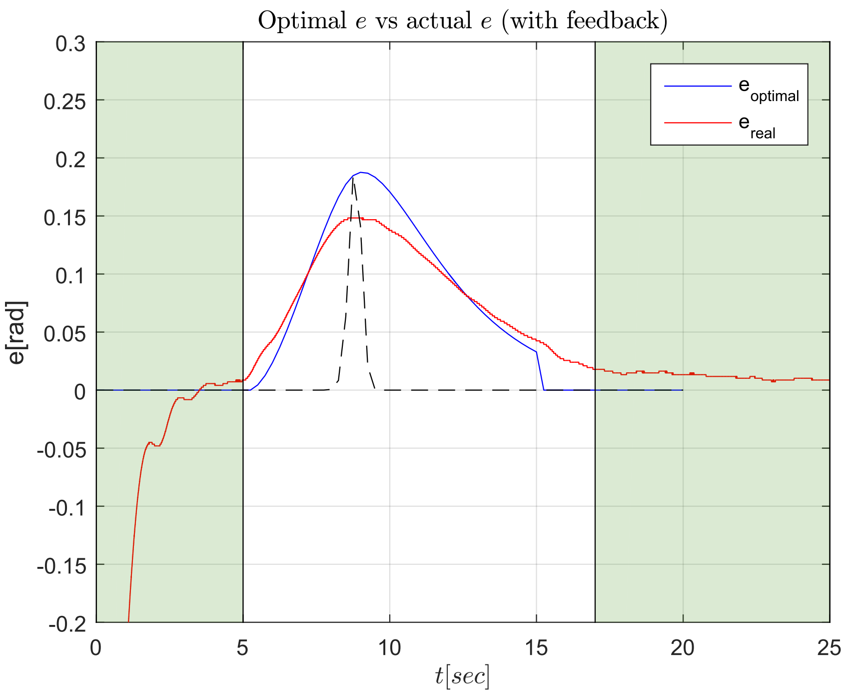
$$K = \begin{bmatrix} -2.032 & -5.853 & 3.338 & 0.967 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7.358 & 5.617 \end{bmatrix}. \tag{26}$$

The weights are chosen to penalize deviation in pitch and travel and particularly elevation. Tuning this is a fine balance, because valuing elevation as much as we have done, affects travel and pitch. This is especially visible in the pitch plot, fig. 10b, where the pitch angle doesn't keep up with the optimal pitch angle around $t \approx 10s$. This is because it doesn't want to loose elevation since the elevation weight is so high. It is necessary to have higher weight on the elevation than pitch and travel, because travel is controlled by pitch. Having a high weight on travel would result in aggressive pitch change, which would again loose altitude for the helicopter.

Observe that the drift off in travel from the open loop system is eliminated, or is with a small steady state error. This can be viewed in fig. 11b. With the feedback control implemented, elevation was much better. The elevation angle is much closer the optimal elevation path with feedback than without.

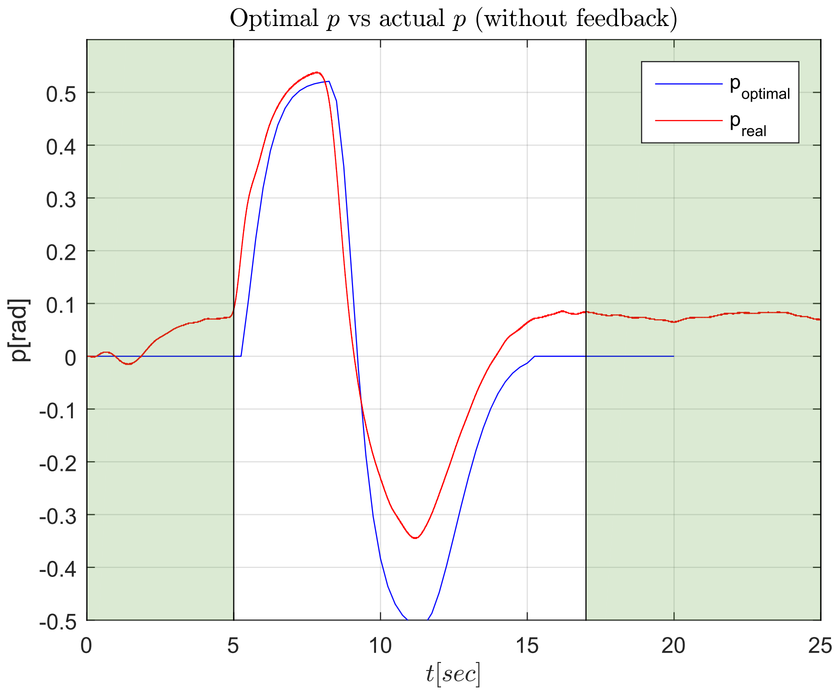


(a) Without feedback

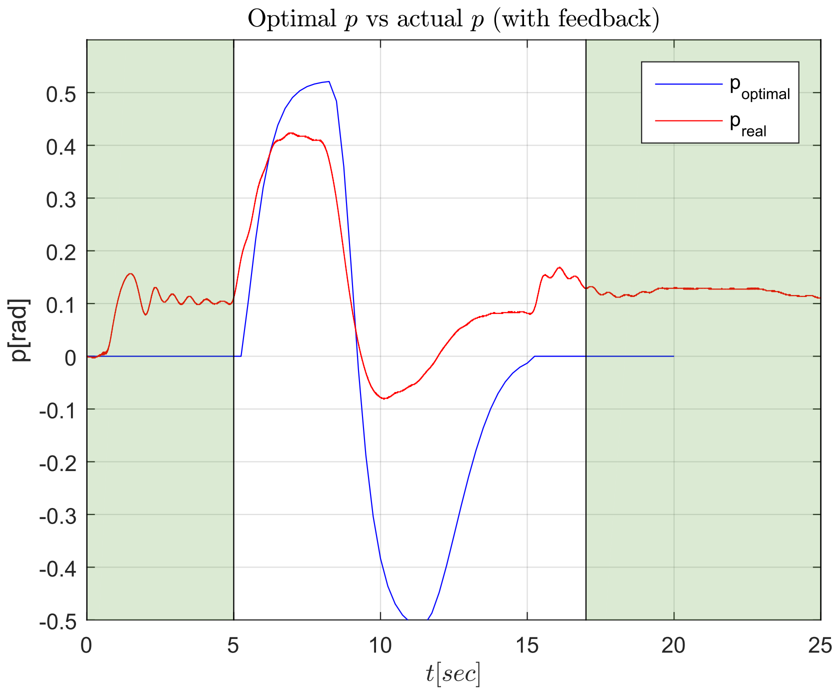


(b) With feedback

Figure 9: Actual elevation response and optimal elevation with and without feedback.

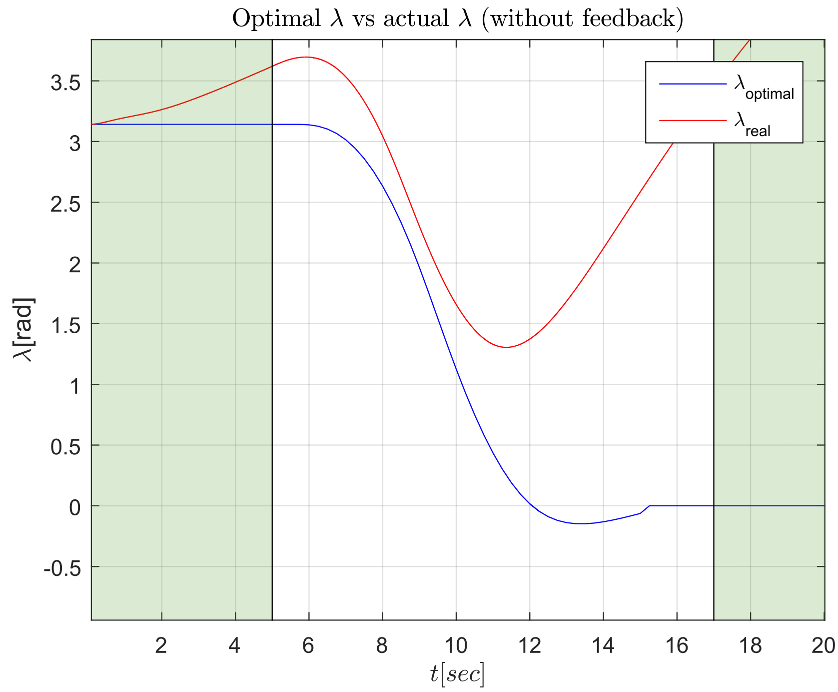


(a) Without feedback.

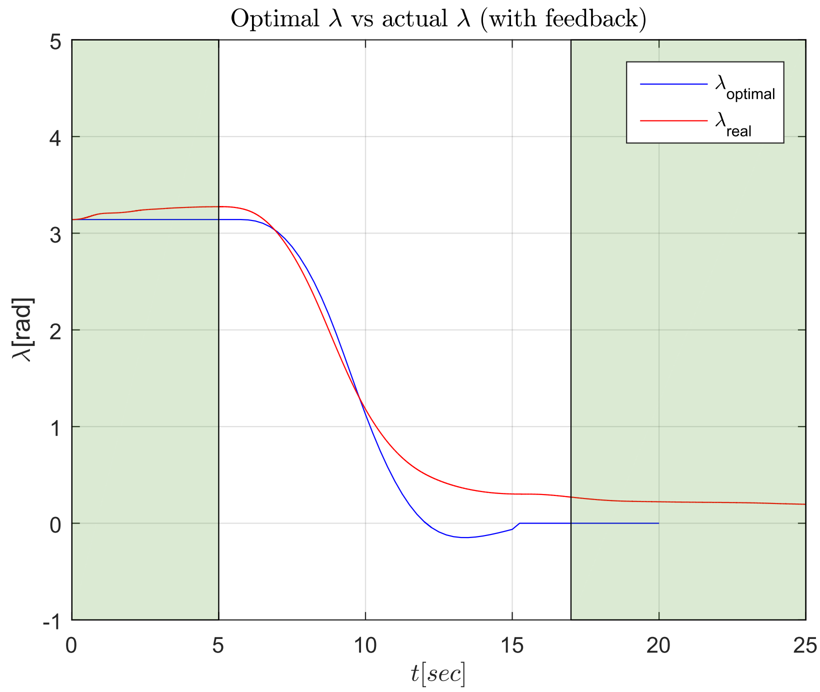


(b) With feedback

Figure 10: Actual pitch response and optimal pitch with and without feedback.



(a) Without feedback



(b) With feedback

Figure 11: Actual travel response and optimal travel with and without feedback.

4.5 Decoupling

In the model, pitch and elevation are not coupled, although this is not true in reality. The upwards working forces is naturally connected to the pitch angle. When the helicopter is turning, as in fig. 12, the plant will experience less upwards lifting thrust, i.e. $||F_f + F_b|| > ||F_{fy} + F_{by}||$ if the pitch is not zero.

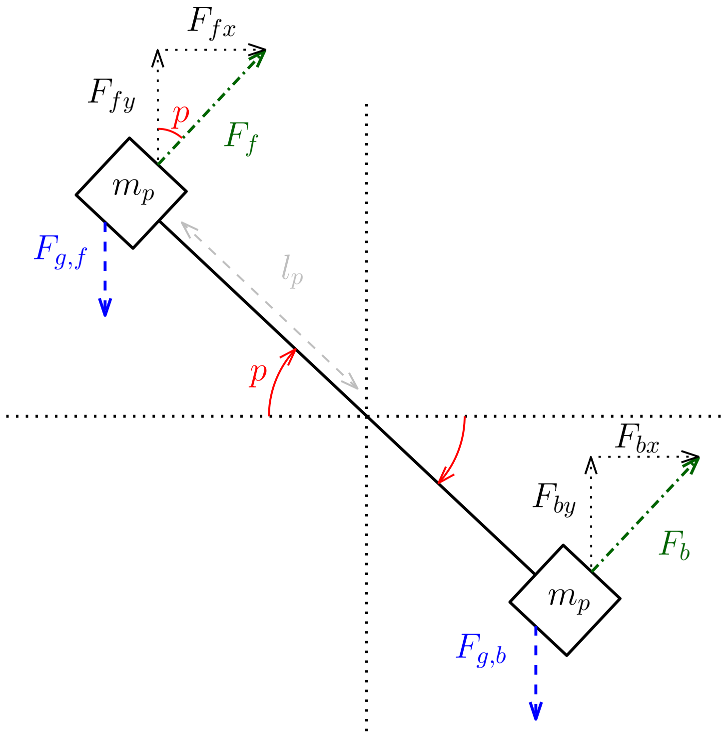


Figure 12: Forces around p joint angle

This decoupling affects the results, and it is therefore not possible to reach the optimal trajectory for both elevation and pitch. Hence, a tradeoff must be done between elevation and pitch. As a result, the actual elevation is lower than the optimal elevation, and the same yields for pitch. A possible improvement of the system could be to include this coupling of elevation and pitch. However, this would result in a nonlinear system that would much more computational heavy to solve.

4.6 Additional Constraints

Two new constraints were added. The travel rate and the elevation rate is now limited to $-0.4 < \dot{\lambda} < 0.4$ and $-0.1 < \dot{e} < 0.1$. The results can be seen in fig. 13, fig. 14 and fig. 15.

It still doesn't reach the optimal trajectory completely, but this is still partially due to the decoupling of states discussed in the previous section. Notice especially in fig. 14 the abrupt change in pitch angle at $t \approx 7s$. This is not physically possible to achieve, but it makes for a good reference for the helicopter's pitch.

However, one could conclude that even though the response with the extra added constraints are somewhat different from the one without the added constraints, it doesn't really make the system notably better. This is because by this part, the limitation of the control does not lie by the control itself, but by the model of the system.

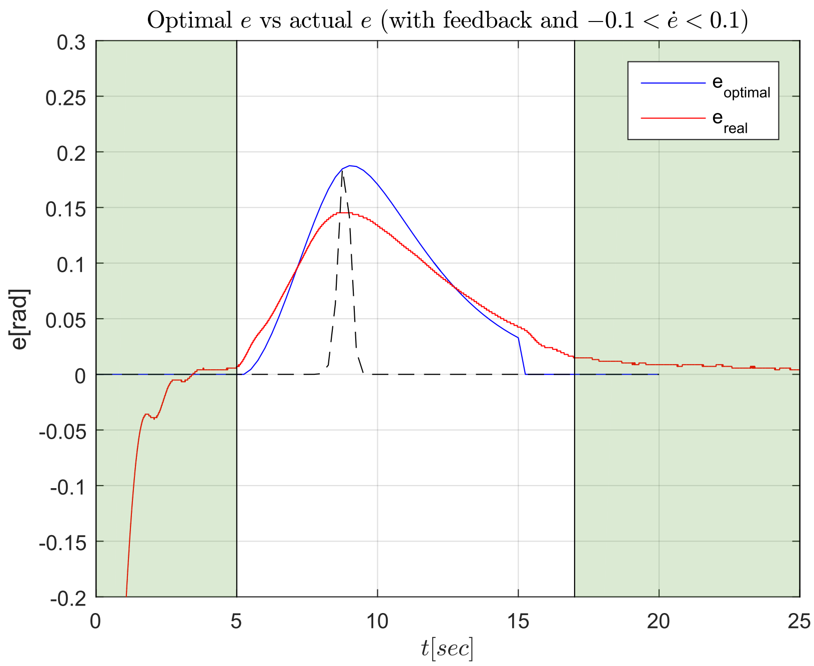


Figure 13: Optimal and actual elevation with additional constraints

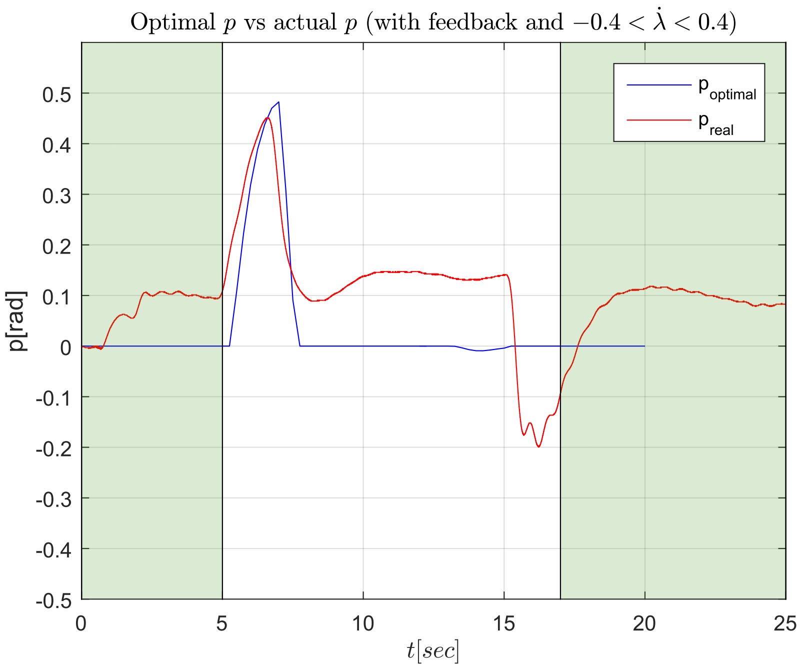


Figure 14: Optimal and actual pitch with additional constraints

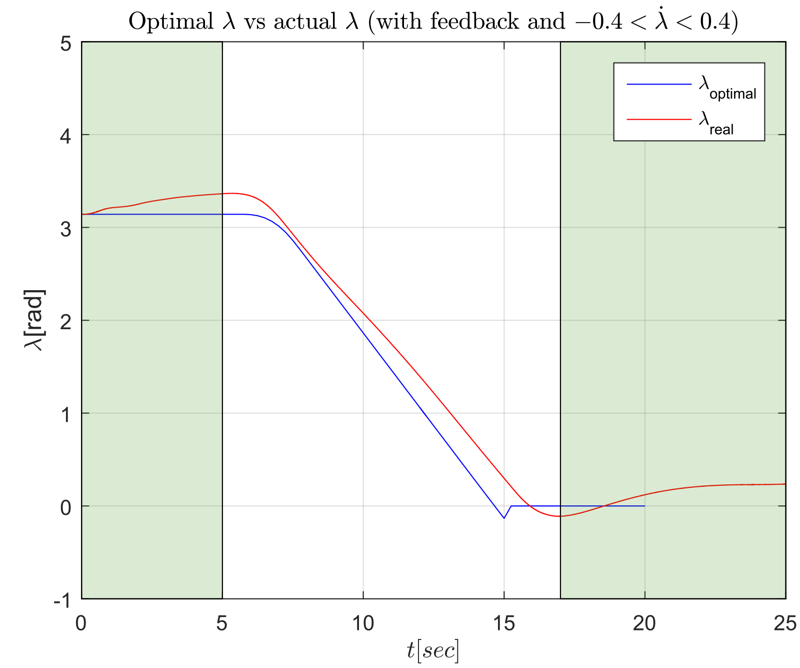


Figure 15: Optimal and actual travel with additional constraints

5 Conclusion

We have in this report shown and compared the responses of different optimization methods with different constraints. In the beginning, there was only linear constraints and no feedback control. As the project progressed, more advanced control was implemented. The nature of feedback control yields a much more robust system. No feedback resulted in drifting and unwanted behaviour. Inaccurate modeling also caused undesired response. However, a decoupling of pitch and elevation was a necessary sacrifice to obtain a linear system. The system performed much better when feedback with a LQ controller was introduced. Drifting was eliminated, but the linearization and simplification of the system still hindered the helicopter from achieving optimal trajectory.

The use of feedback control proved crucial regardless of the applied constraints. When the nonlinear elevation constraint was implemented, the helicopter performed poorly without feedback compared to when feedback was implemented. It still did not manage to fulfill its nonlinear constraint, but the performance was much better with feedback.

Better performance was attempted by introducing stricter constraints, with barely improved response compared to before. The newly implemented constraints of travel should in reality make the helicopter's elevation path better, because of how travel and pitch are physically related. However, the actual responses with and without stricter constraints, are almost identical. Again this traces back to the simplification of the actual nonlinear system that was made linear.

A MATLAB Code

A.1 Part 2 - Feed-forward

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

%% ---- PROBLEM 10.2.1 ---- %%
%%Initialization and continous model definition

init06; % Change this to the init file corresponding to your helicopter

A_c = [0 1 0 0;
       0 0 -K_2 0;
       0 0 0 1;
       0 0 -K_1*K_pp -K_1*K_pd];

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

%% ---- PROBLEM 10.2.2 ---- %%
%%Discretized model

delta_t = 0.25;                              % sampling time
I       = eye(4);
A1      = I + delta_t * A_c;
B1      = delta_t * B_c;
%% ---- PROBLEM 10.2.3 ---- %%
%%Calculate optimal trajectory over finite time horizon

% 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
```

```
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      = 30*pi/180;                    % 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 G and the vector c (objecitve function weights in the QP problem)
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
G       = gen_q(Q1,P1,N,M);                     % Generate G,
     hint: gen_q
c       = zeros(N*mx+M*mu, 1);                  % Generate c,
     this is the linear constant term in the QP

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

%%Solve QP problem with linear model
tic
[z,lambda] = quadprog(G,c,[],[],Aeq,beq,vlb,vub,x0);
    % 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+G(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]; % lambda
x2   = [zero_padding; x2; zero_padding];
x3   = [zero_padding; x3; zero_padding];    % Pitch
x4   = [zero_padding; x4; zero_padding];

%% ---- PROBLEM 10.2.4 ---- %%
%%Implement in simulink
% To workspace
t                           = 0:delta_t:delta_t*(length(u)-1);
input.signals.values      = u;
input.time                = t;
input.signals.dimensions = 1;
```

A.2 Part 3 - LQR

```
%% ---- PROBLEM 10.3.1 ---- %%
%%Introduce LQR controller

Q = diag([20 1 0.5 0.5]);
R = 1;

K      = dlqr(A1, B1, Q, R);     % dlqr - K-matrix for discrete lqr
K_T    = K';
```

A.3 Part 4 - Inequality constraint

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

%% ---- PROBLEM 10.4.2 ---- %%
%%Initialization and model definition
init06; % Change this to the init file corresponding to your helicopter

%Defined global for @NONLNCON
global lambda_t alpha beta mx N

```

```
%CONSTANTS
lambda_t = 2*pi/3; alpha = 0.2; beta = 20;


% Discrete time extended system model
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 -delta_t*K_3*K_ep 1-delta_t*K_3*K_ed
           ];
B1      = [0 0 0 delta_t*K_1*K_pp 0 0; 0 0 0 0 0 delta_t*K_3*K_ep]';

% 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
```

```
% Bounds
ul       = [-30*pi/180; -Inf];             % Lower bound on control
uu       = [30*pi/180; 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

%% Optional exercise 10.4.6 - bounds
%xl(6) = -0.1;                             %Lower bound on state x6
%xu(6) = 0.1;                              %Upper bound on state x6

xl(2) = -0.4;                              %Lower bound on state x2
xu(2) = 0.4;                               %Upper bound on state x2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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 G and the vector c (objecitve function weights in the QP problem)
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
Q1(5,5) = 0;                              % Weight on state x5
Q1(6,6) = 0;                              % Weight on state x6
P1      = 1;                              % Weight on input q1
P2      = 1;                              % Weight on input q2
P       = [P1 0 ; 0 P2];                  % Input matrix
G       = gen_q(Q1,P,N,M);                % Generate G, hint: gen_q
c       = zeros(N*mx+M*mu,1);             % Generate c, this is the linear constant term in the QP


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

%%Solve QP problem with nonlinear inequality constraint
object_fun           = @(z) (1/2*z'*G*z);
z0(1:mx)             = x0;                %A starting point for the solver
opt                  = optimoptions('fmincon', 'Algorithm', 'sqp', 'MaxFunEvals', 40000);
tic
[z, ZVAL, EXITFLAG] = fmincon(object_fun, z0, [], [], Aeq, beq, vlb, vub, @NONLNCON, opt);
t1=toc;

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

```

```matlab
%%Extract control inputs and states
u1 = [z(N*mx+1:mu:N*mx+M*mu);z(N*mx+M*mu-1)]; % Control input 1 from solution
u2 = [z(N*mx+2:mu:N*mx+M*mu);z(N*mx+M*mu)];   % Control input 2 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
x5 = [x0(5);z(5:mx:N*mx)];
x6 = [x0(6);z(6:mx:N*mx)];

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

u1  = [zero_padding; u1; zero_padding];
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];


% Solving LQR
Q      = diag([30 1 30 3 100 1]);
R      = [1 0; 0 1];
K      = dlqr(A1, B1, Q, R);           % dlqr - K-matrix for discrete lqr
K_T    = (K');

% Calculating constraint
nonlincon = zeros(length(x5), 1);
for i = 1: length(x1)
    nonlincon(i) = alpha*exp(-beta*(x1(i) - lambda_t)^2);
end
```

```
%% Creating struct for simulink input
t                           = 0:delta_t:delta_t*(length(u1)-1);
input.signals.values        = [u1 u2];
input.time                  = t;
input.signals.dimensions = 2;
x_opt.signals.values        = [x1 x2 x3 x4 x5 x6];
x_opt.time                  = t;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.3.1 Function for computing the nonlinear constraint

```
%% ---- PROBLEM 10.4.3 ---- %%
%%Implementation of the nonlinear constraint - to be fed into fmincon
function [C, Ceq] = NONLNCON(z)
global alpha beta lambda_t mx N
C = zeros(N,1);
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
```

B Simulink Diagrams

B.1 Part 2 - Optimal input directly as reference for pitch

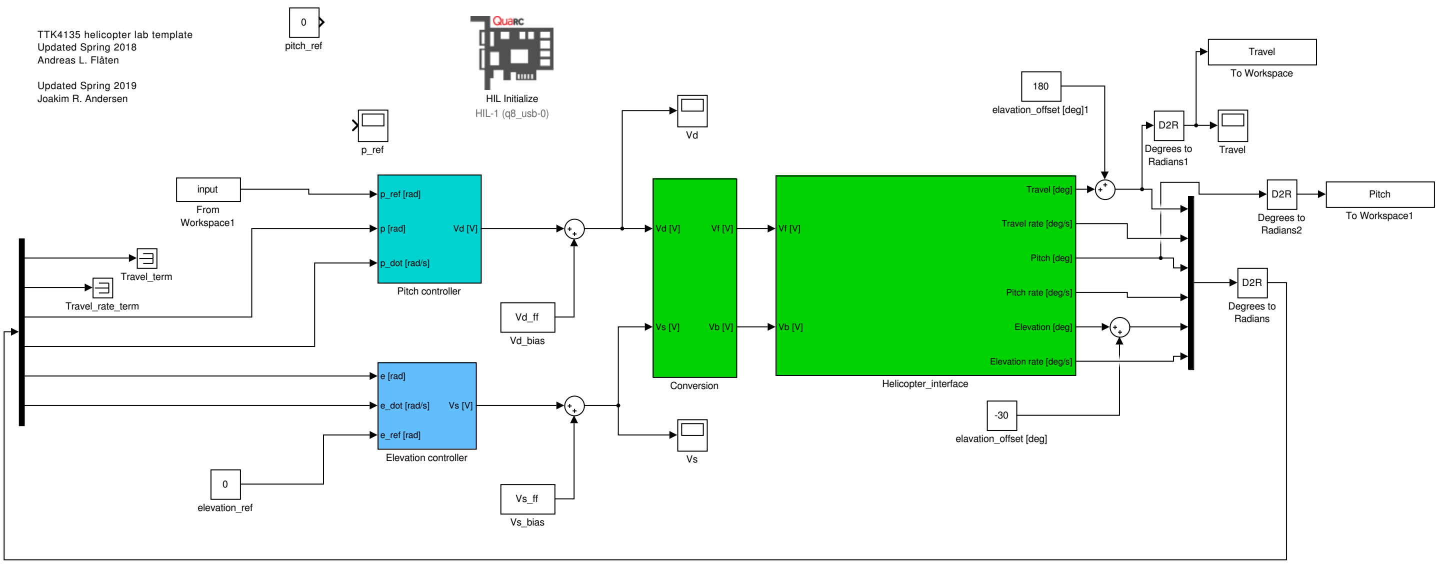


Figure 16: Simulink for part 2

B.2 Part 3 - LQ controller with feedback

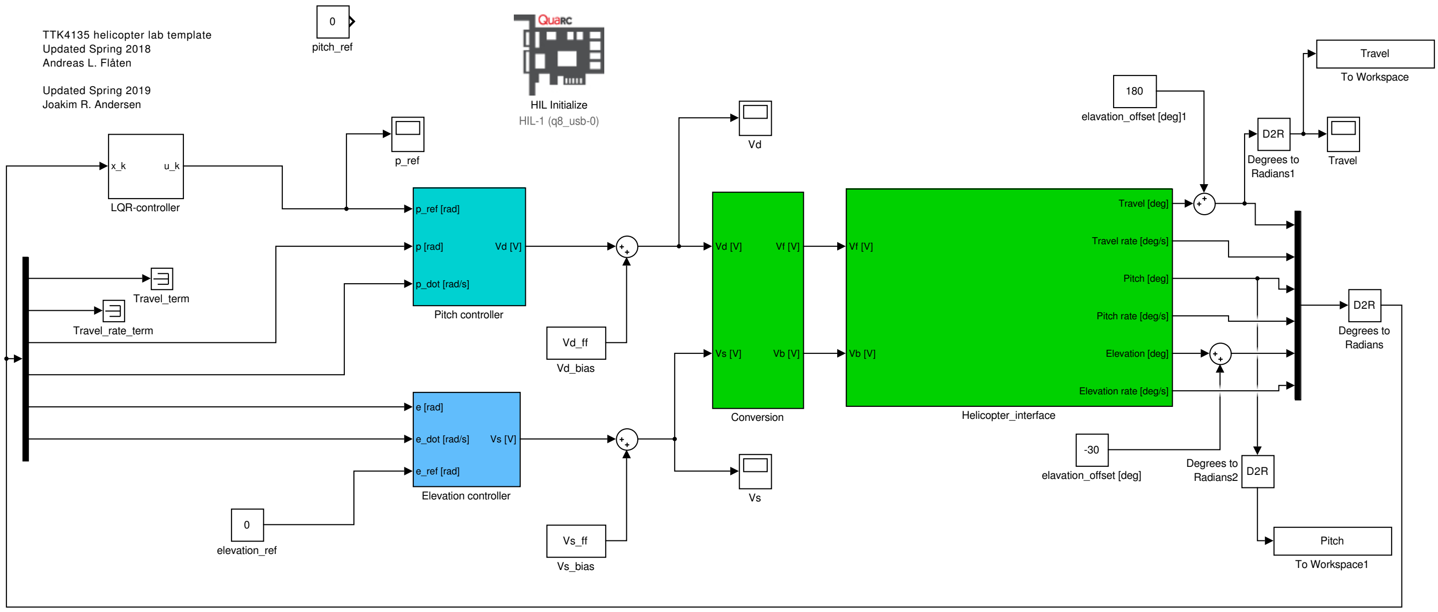


Figure 17: Simulink for part 3

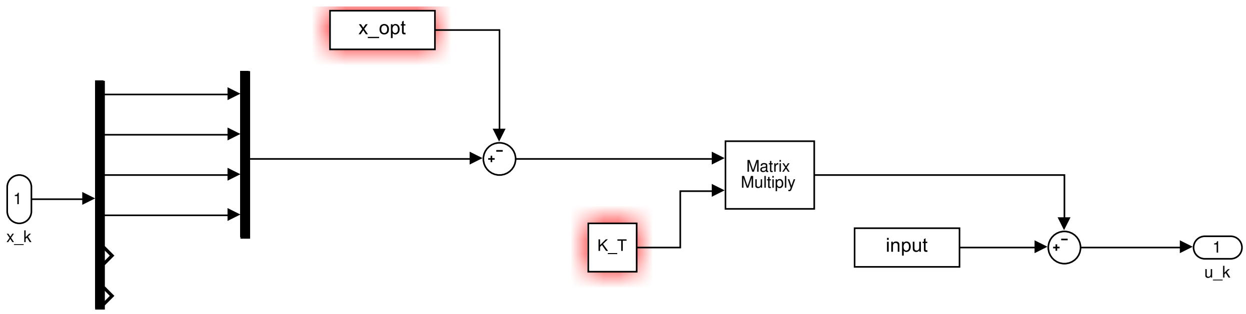


Figure 18: LQ controller submodule for part 3

B.3 Part 4 - LQ controller with and without feedback

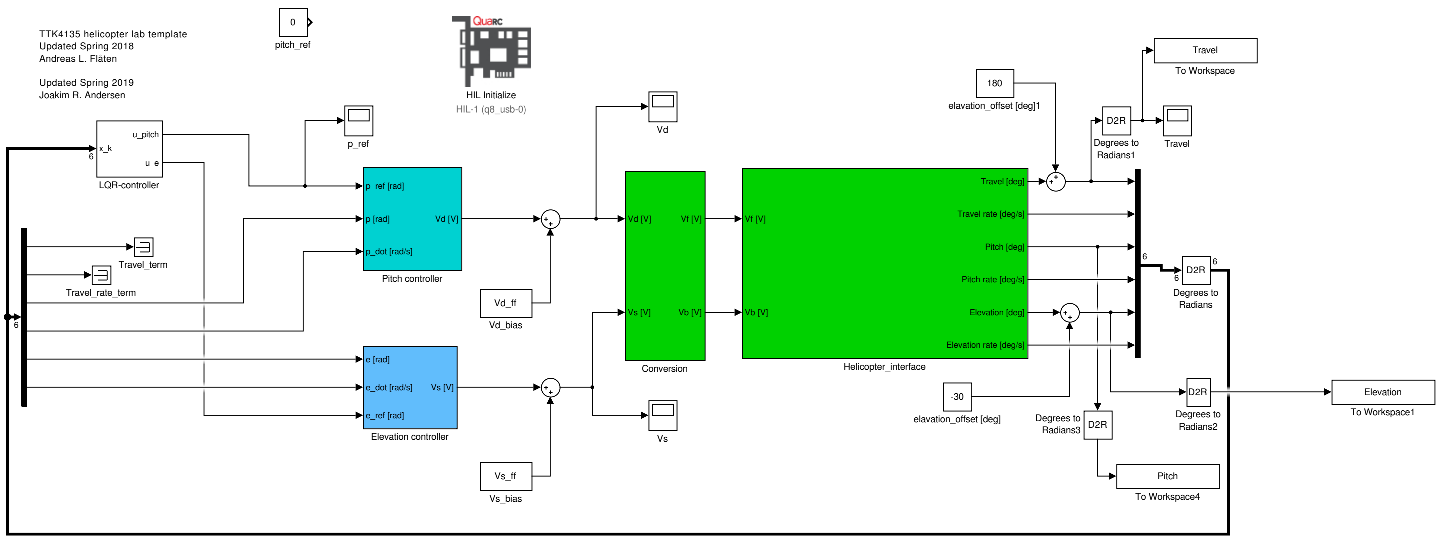


Figure 19: Simulink for part 4 with feedback

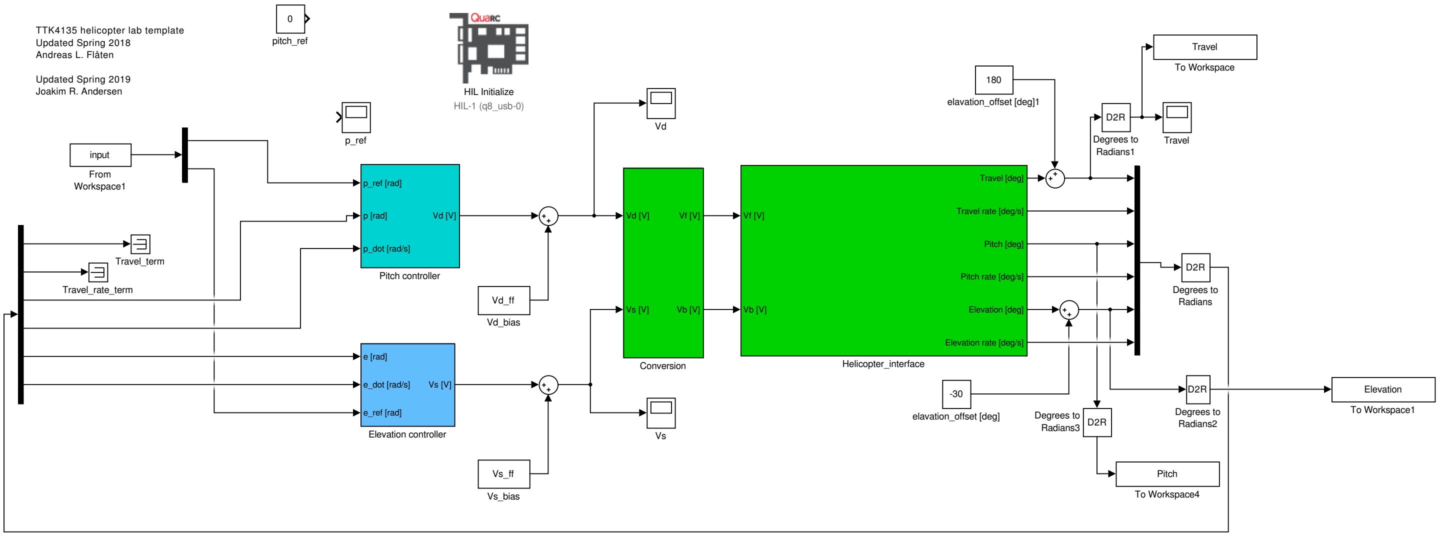


Figure 20: Simulink for part 4 without feedback

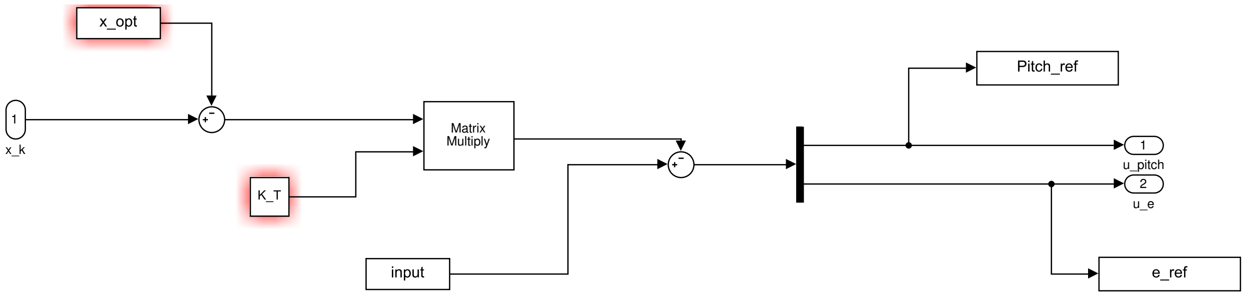


Figure 21: LQ controller submodule for part 4

References

[1] *QuaRC - QuaRC Real-Time Control*. https://www.quanser.com/products/quarc-real-time-control-software/. Accessed: 2019-03-08.

[2] *TTK4115 – Linear System Theory Helicopter lab assignment*. https://ntnu.blackboard.com/bbcswebdav/pid-420173-dt-content-rid-17161604_1/courses/194_TTK4115_1_2018_H_1/194_TTK4115_1_2018_H_1_ImportedContent_20180815022347/Helicopter_lab_assignment.pdf. Accessed: 2019-03-08.

[3] *TTK4135 - Optimization and Control Helicopter Lab*. https://ntnu.blackboard.com/bbcswebdav/pid-575741-dt-content-rid-19278776_1/courses/194_TTK4135_1_2019_V_1/Lab/LabExercise.pdf. Accessed: 2019-03-08.