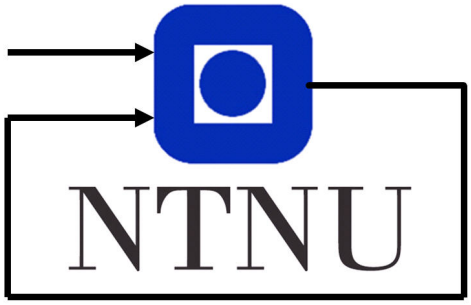
TTK4135
Optimization and Control
Helicopter Lab

Group 20
478419
478447
478428
478404

March 15, 2019



Department of Engineering Cybernetics

Abstract

In this experiment the objective is formulating and solving optimization problems for the trajectory of an attached helicopter. The state space model is discretized, and solutions are found for control of pitch, travel and elevation both with and without feedback. The solutions are implemented and tested on the helicopter using Matlab and Simulink.

Contents

1 Introduction

This lab involves formulating a dynamic optimization problem. The task is to implement optimal control on a stationed helicopter both with and without feedback. Given certain constraints, an optimal trajectory and it's corresponding optimal control input sequence should be computed for the helicopter. The problem will be discretized and solved using Matlab and Simulink.

The report starts out with a description of the problem, including the lab setup and a model of the helicopter. In this section the system equations that will be used throughout the report are stated.

Next, a discretization of the system will be used together with different types of constraints to calculate optimal trajectory and inputs. This is implemented on the helicopter both with and without feedback.

After each exercise the achieved results will be presented, explained and discussed before ending the report with a final conclusion on the project.

2 Problem Description

The helicopter consists of a base with an arm attached. To balance the helicopter a weight is added on the opposite end of the arm. Figure 1 shows an illustration of the helicopter and its components. The arm can be moved up and down around the elevation (e) axis depending on the sum of the force generated from both helicopter blades. The difference in force generated from the blades controls movement around the pitch (p) axis, which controls movement around the travel (λ) axis. The force from the blades is assumed to be proportional to the voltage applied to the motors. Parameter values and variables are listed in tables 1 and 2. Equation (1) describes the differential equations for elevation (e), pitch (p) and travel (λ).

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

Discretization of the models during the exercise will be done with the forward Euler method shown in eq. (2)

$$\frac{x_{k+1} - x_k}{\Delta t} = A_c x_k + B_c u_k \tag{2}$$

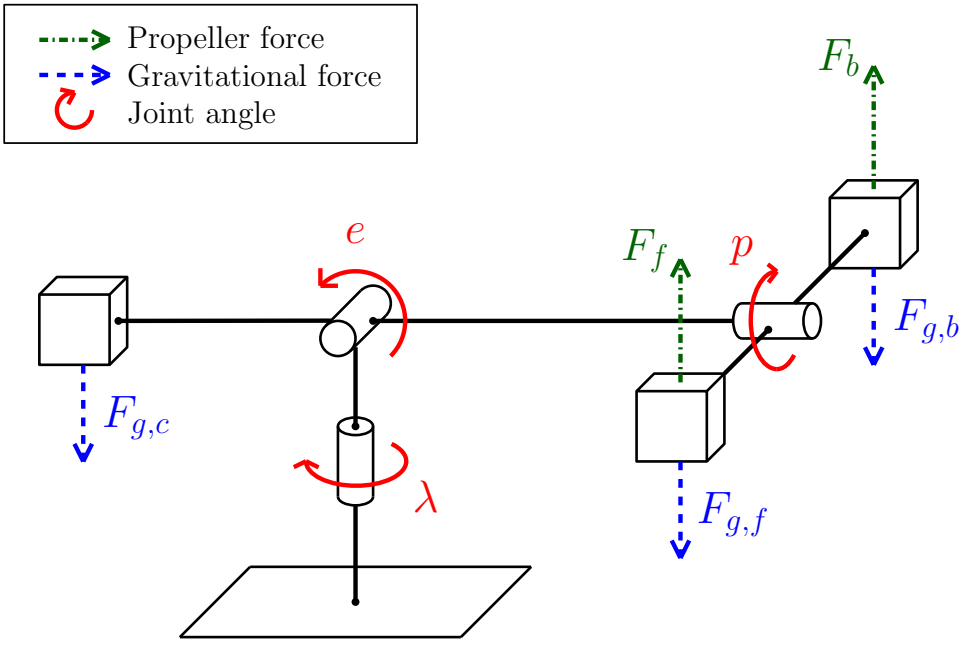


Figure 1: Illustration of the helicopter

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
m_h	Mass of helicopter	1.05	kg
m_w	Balance weight	1.87	kg
m_g	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, $V_f + V_b$
$K_{pp}, K_{pd}, K_{ep}, K_{ei}, K_{ed}$	Controller gains
T_g	Moment needed to keep the helicopter flying

3 Problem 10.2 Optimal Control of Pitch and Travel without Feedback

Firstly, the elevation is disregarded when calculating an optimal trajectory $\mathbf{x}^*$ and an optimal input sequence $\mathbf{u}^*$. The input sequence will be used as control input and implemented as the setpoint for the inner controllers of the helicopter. There will however be no feedback to correct for deviations, and therefore a deviation from the setpoint is expected. Figure 2 illustrates the implementation of optimal control of the pitch and travel without feedback.

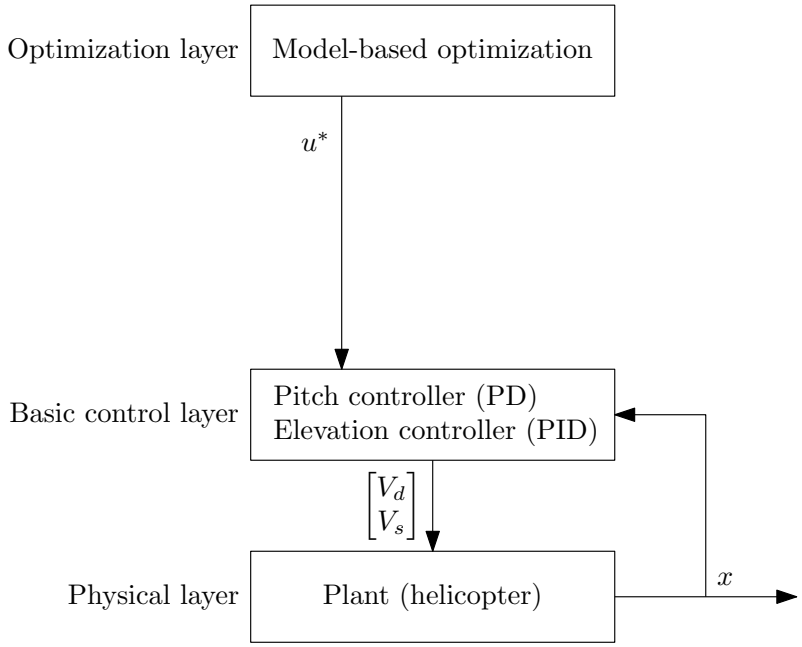


Figure 2: Illustration of control hierarchy in open loop.

The model is written on continuous time state space form as

$$\dot{\mathbf{x}} = \mathbf{A_c}\mathbf{x} + \mathbf{B_c}\mathbf{u} \tag{3}$$

with

$$\mathbf{x} = \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \end{bmatrix}, \quad u = p_c \tag{4}$$

and

$$\mathbf{A_c} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} \end{bmatrix}, \quad \mathbf{B_c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1K_{pp} \end{bmatrix} \tag{5}$$

The model described in eqs. (3) to (5) includes the helicopter dynamics, as well as the controller for pitch and works as the lower layers of the system. In this exercise, elevation (e) is not included in the states, and the model only involves travel (λ), pitch (p) and their derivatives.

3.1 Discretizing Using Forward Euler

Discretization of the model described in eq. (3) by using forward Euler (2) gives

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} + \mathbf{B}\mathbf{u_k} \tag{6}$$

with

$$\mathbf{x_k} = \begin{bmatrix} \lambda_k \\ r_k \\ p_k \\ \dot{p_k} \end{bmatrix}, \quad u_k = p_c \tag{7}$$

and

$$\mathbf{A} = \begin{bmatrix} 1 & 1\Delta t & 0 & 0 \\ 0 & 1 & -K_2\Delta t & 0 \\ 0 & 0 & 1 & 1\Delta t \\ 0 & 0 & -K_1K_{pp}\Delta t & 1 - K_1K_{pd}\Delta t \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1K_{pp}\Delta t \end{bmatrix} \tag{8}$$

3.2 Calculating the Optimal Trajectory

The optimal trajectory, $\mathbf{x}^*$, will move the helicopter from position $\mathbf{x}_0 = [\lambda_0 \ 0 \ 0 \ 0]^\top$ to $\mathbf{x}_f = [\lambda_f \ 0 \ 0 \ 0]^\top$ with $\lambda_0 = \pi$ and $\lambda_f = 0$.

$\mathbf{x}^*$ will be found by formulating the problem as an optimization problem, which then will be solved as a quadratic program of the following form:

$$\min_{\mathbf{z}} \frac{1}{2}\mathbf{z}^\top \mathbf{H}\mathbf{z} \quad \text{subject to} \quad \begin{cases} \mathbf{A}_{eq}\mathbf{z} = \mathbf{b}_{eq} \\ lb \leq \mathbf{z} \leq ub \end{cases} \tag{9}$$

with $\mathbf{z}$ acting as a vector containing all the state values $\mathbf{x}_k$ and input values $\mathbf{u_k}$ at every time step k, given by:

$$\mathbf{z} = [x_0, ..., x_{n-1}, u_0, ..., u_{n-1}]^\top \tag{10}$$

where n is the number of time steps.

Choosing

$$\mathbf{H} = \begin{bmatrix} \mathbf{Q} & 0 & \dots & & \dots & 0 \\ 0 & \ddots & & & & \vdots \\ \vdots & & \mathbf{Q} & & & \\ & & & \mathbf{P} & & \vdots \\ \vdots & & & & \ddots & 0 \\ 0 & \dots & & \dots & 0 & \mathbf{P} \end{bmatrix} \tag{11}$$

$$\text{with} \quad \mathbf{Q} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{P} = P = q \tag{12}$$

gives the following cost function:

$$\phi = \frac{1}{2}\mathbf{z}^\top \mathbf{H}\mathbf{z} = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q p_{ci}^2, \quad q \geq 0 \tag{13}$$

where N is equal to the number of time steps. Note that $\lambda_f = 0$ gives no linear term, and hence this has been omitted from the equation. Here, q is used as a weight on the pitch setpoint p_c. By increasing the weight, the usage cost of the manipulated variable is increased. In other words, by increasing the weight of the manipulated variable, one would expect to see less use of this manipulated variable in the calculated optimal solution.

As the helicopter will get unstable if rotated too far around the pitch axis, a pitch constraint is needed. This is modeled as the upper and lower bounds, ub and lb respectively, (denoted uu and ul in appendix A.2). The pitch constraint which is implemented is given by:

$$|p_k| \leq \frac{30\pi}{180}, \quad k \in \{1, ..., N\} \tag{14}$$

The equality constraints for the quadratic program will be the discrete state space system given in (6) at every time step k. This gives the following $\mathbf{A}_{eq}$ for the quadratic program:

$$\mathbf{A}_{eq} = \left[\begin{array}{ccccc|ccccc} \mathbf{I} & 0 & & \dots & 0 & -\mathbf{B} & 0 & & \dots & 0 \\ -\mathbf{A} & \mathbf{I} & 0 & \dots & \vdots & 0 & -\mathbf{B} & 0 & \dots & 0 \\ 0 & -\mathbf{A} & \mathbf{I} & \dots & \vdots & 0 & 0 & -\mathbf{B} & \dots & \vdots \\ \vdots & \vdots & & \ddots & 0 & \vdots & \vdots & & \ddots & 0 \\ 0 & \dots & 0 & -\mathbf{A} & \mathbf{I} & 0 & 0 & \dots & 0 & -\mathbf{B} \end{array}\right] \tag{15}$$

and $\mathbf{b}_{eq}$ is given by:

$$\mathbf{b}_{eq} = [\mathbf{A}\mathbf{x}_0 \;\; 0 \;\; \dots \;\; 0]^\top \tag{16}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the system matrices given by (6).

Finally, the optimization problem is solved using the Matlab function `quadprog` with $N = 100$. Figure 3 shows how different values of q affect the optimal input-sequence and estimated pitch. As expected we observe that higher values of q penalizes use of input. $q = 1$ seems like a reasonable value to use, as we get a quick response without overshooting. Also, it is clear that the pitch constraint given by eq. (14) is a reasonable constraint to implement as it is active for longer periods.

By studying the cost-function (13) it is observed that when $\lambda = \lambda_f$, the cost-function reduces to $\phi = \sum_{i=1}^{N} q p_{ci}^2$. The input which now minimize the cost-function is zero, resulting in overshoot or drifting. This is a result of not including travel-rate in the cost-function, meaning that the input is set to zero regardless of the speed the helicopter is traveling. In addition, as it is pitch and not travel being controlled, the system is affected by errors in the system equations.

3.3 Implementing the Optimal Trajectory

The calculated optimal trajectory is then implemented on the helicopter with $q = 1$. This is done in Simulink, and the model used to control the helicopter with the optimal trajectory can be seen in fig. 14. The two controllers used to control pitch and elevation are simple proportional controllers. Since the sensors of the helicopter are reset at initialization, an offset of π rad is added to the measurement in Simulink.

The optimal input sequence is padded with zeros in both ends to allow the helicopter to stabilize before starting on the optimal trajectory. The plotted results can be seen in figs. 4 and 5.

It can be seen from the measured travel λ that there is a deviation between the measured travel trajectory and the optimal travel trajectory. This is due to the lack of feedback control of the travel. The helicopter starts drifting, and the does not end in the desired point $\mathbf{x}_f = [0 \;\; 0 \;\; 0 \;\; 0]^\top$. As stated in the opening of this section, this is to be expected. When the optimal trajectory is implemented without feedback control, there will be no way of compensating for physical disturbances and weaknesses in the used mathematical model, as these results clearly illustrate.

The optimal trajectory for the pitch p is followed rather closely due to the implemented pitch controller. However, a constant deviation is present, which will be discussed further in section 4.

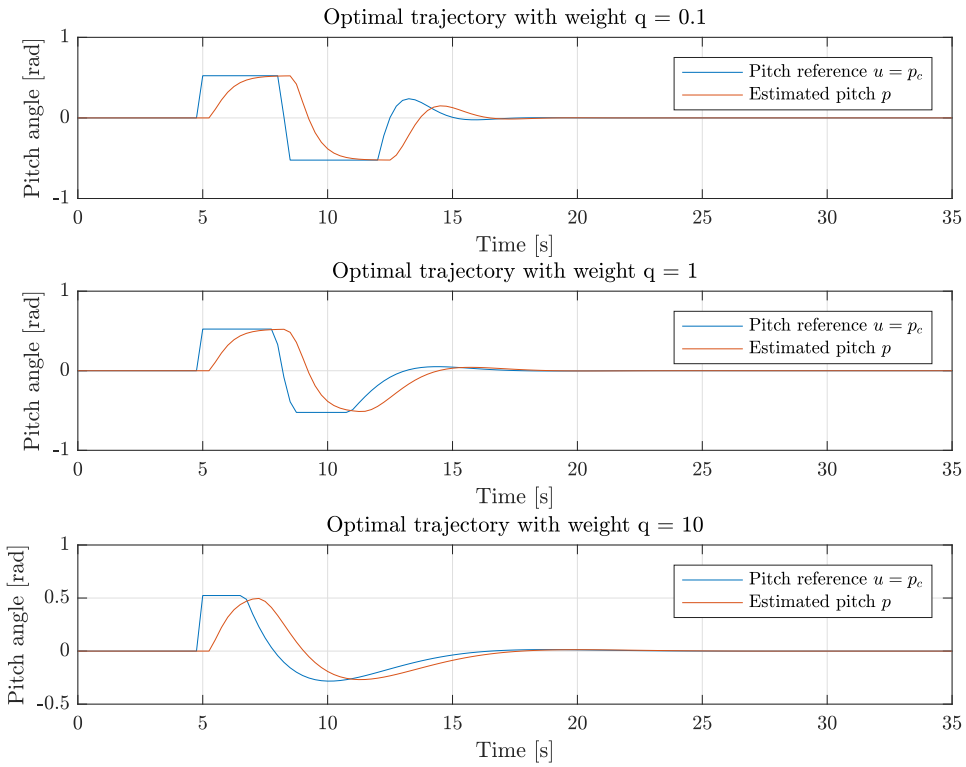


Figure 3: Manipulated variable p_c vs. estimated output p

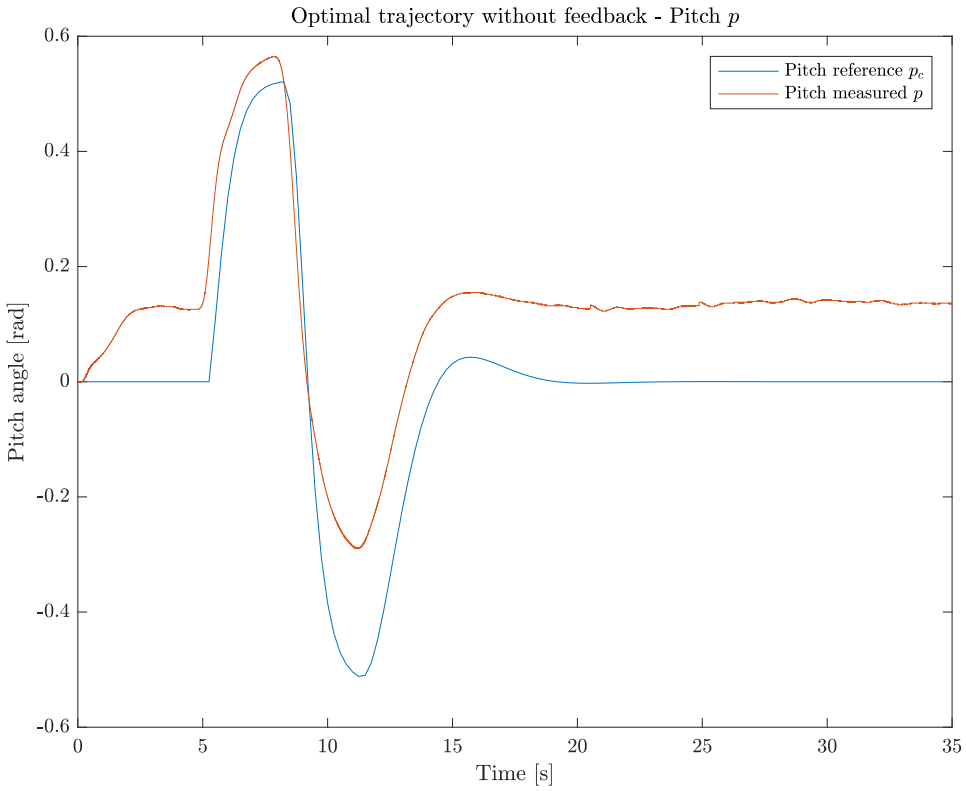


Figure 4: Optimal control without feedback – Plot of the pitch p

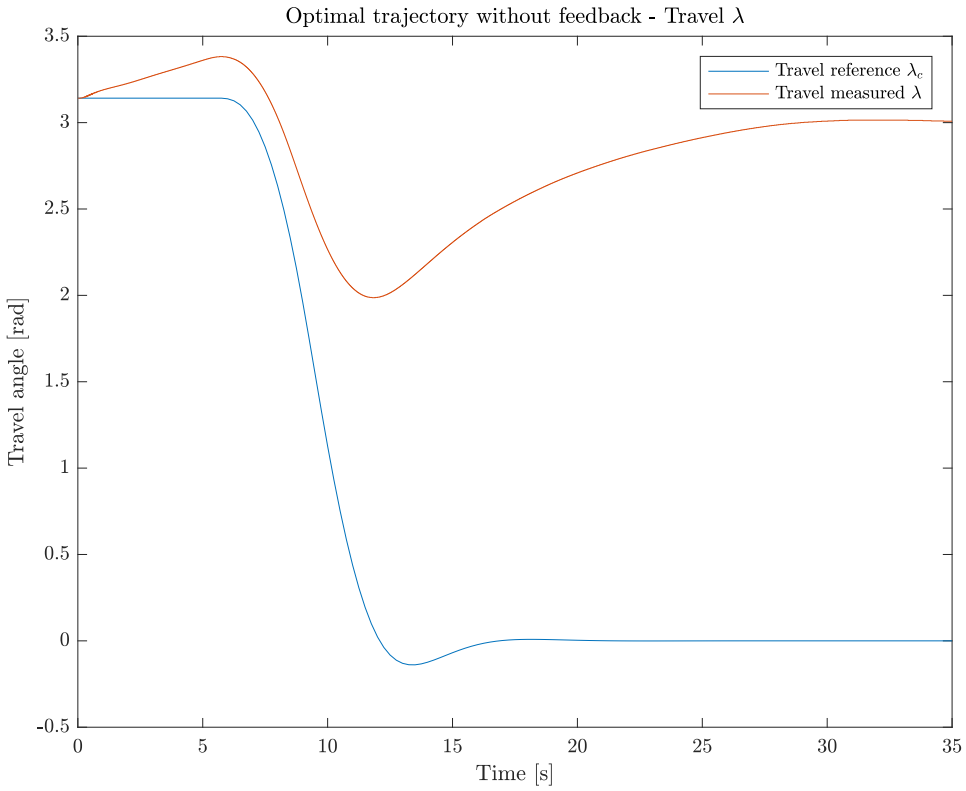


Figure 5: Optimal control without feedback – Plot of the travel λ

4 Problem 10.3 Optimal Control of Pitch and Travel with Feedback (LQ)

In this part, feedback is to be introduced in the optimal controller. This will allow the controller to adjust the input such that the optimal trajectory is followed more closely. Figure 6 illustrates the control hierarchy with feedback. To avoid misunderstandings, all matrices involved in the LQ controller are denoted with **lq**

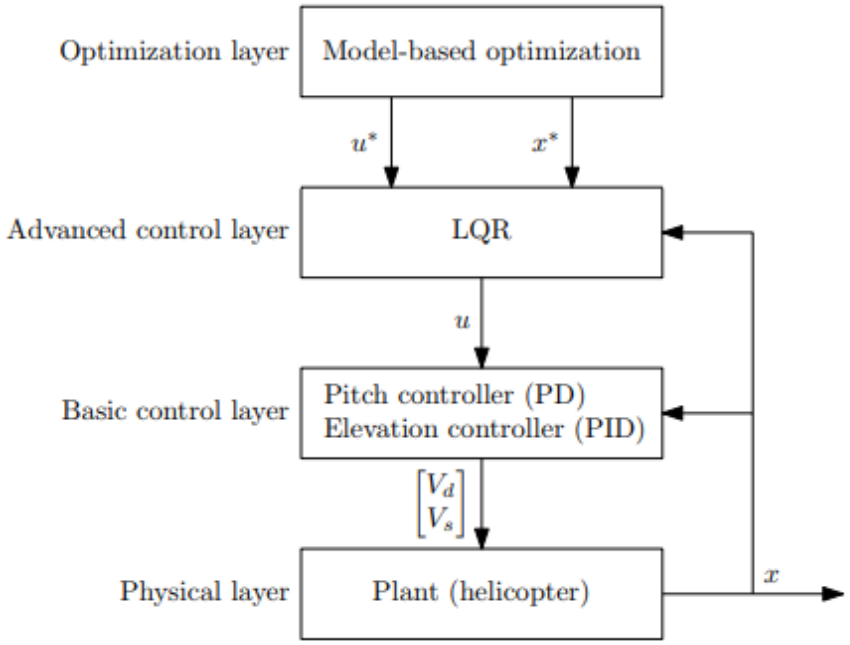


Figure 6: Illustration of control hierarchy with feedback

4.1 Feedback Controller (LQ)

In order to implement feedback control, a LQ controller is used. This is a linear quadratic controller, in the sense that it will minimize a quadratic criteria for a linear model. In essence, deviations in the state $\mathbf{x}$ will be penalized from the optimal trajectory $\mathbf{x}^*$ according to a gain matrix $\mathbf{K_{lq}}$. Equation (17) shows the implemented feedback control:

$$\mathbf{u}_k = \mathbf{u}_k^* - \mathbf{K_{lq}}^\top(\mathbf{x}_k - \mathbf{x}_k^*) \tag{17}$$

The gain matrix $\mathbf{K_{lq}}$ is calculated as an LQ controller, which minimizes the quadratic objective function:

$$J = \sum_{i=0}^{\infty} \Delta\mathbf{x}_{i+1}^{\top}\mathbf{Q_{lq}}\Delta\mathbf{x}_{i+1} + \Delta\mathbf{u}_i^{\top}\mathbf{R_{lq}}\Delta\mathbf{u}_i, \quad \mathbf{Q_{lq}} \geq 0, \mathbf{R_{lq}} > 0 \tag{18}$$

for a linear model

$$\Delta\mathbf{x}_{i+1} = \mathbf{A}\Delta\mathbf{x}_i + \mathbf{B}\Delta\mathbf{u}_i \tag{19}$$

where $\Delta\mathbf{x}$ and $\Delta\mathbf{u}$ are deviations from the optimal trajectory, and $\mathbf{Q_{lq}}$ and $\mathbf{R_{lq}}$ indicates how much one wishes to penalize deviations from the optimal trajectory $\mathbf{x}^*$ and use of the manipulated variable $\mathbf{u}$, respectively.

$$\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^* \tag{20}$$

$$\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}^* \tag{21}$$

$\mathbf{K_{lq}}$ is calculated using the MATLAB function `dlqr()`. $\mathbf{Q_{lq}}$ and $\mathbf{R_{lq}}$ are chosen as diagonal matrices for simplicity. By eq. (7) $\mathbf{R_{lq}}$ is reduced to a scalar, R_{lq}. By trial and error, the following values for $\mathbf{Q_{lq}}$ and R_{lq} are found:

$$\mathbf{Q_{lq}} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}, \quad R_{lq} = 1 \tag{22}$$

This $\mathbf{Q_{lq}}$ is mostly focused on penalizing travel error, which is the main objective. Slightly different values are used in section 5. The values found here results in the following $\mathbf{K_{lq}}$:

$$\mathbf{K_{lq}} = \begin{bmatrix} -2.627 & -6.405 & 2.891 & 0.867 \end{bmatrix} \tag{23}$$

4.2 Implementing the Feedback Controller

The feedback is implemented on the controller by including it in the Simulink model used in section 3, as can be seen in fig. 15. A `From Workspace` block is used to import the input sequence $\mathbf{u}^*$ into the model. Figure 16 shows the LQ controller being used.

The results obtained after implementing feedback are plotted in figs. 4 and 7. Comparing figs. 5 and 8 with and without feedback, respectively, shows a notable difference in travel λ. The helicopter no longer drifts, and

seems to follow the optimal trajectory closely. This is due to the effect from the feedback, as the LQ controller will ensure that deviations from the optimal trajectory $\mathbf{x}^*$ will be compensated for.

One thing to note is that while the optimal trajectory is now followed more closely, there is still a constant deviation from the optimal trajectory, both in the pitch p and the travel λ. This is due to the fact that neither the pitch controller nor the travel controller have any integral effect, resulting in a constant deviation. Implementing the P controller and the LQ controller with integral effect would eliminate this deviation. This will not be attempted in this experiment.

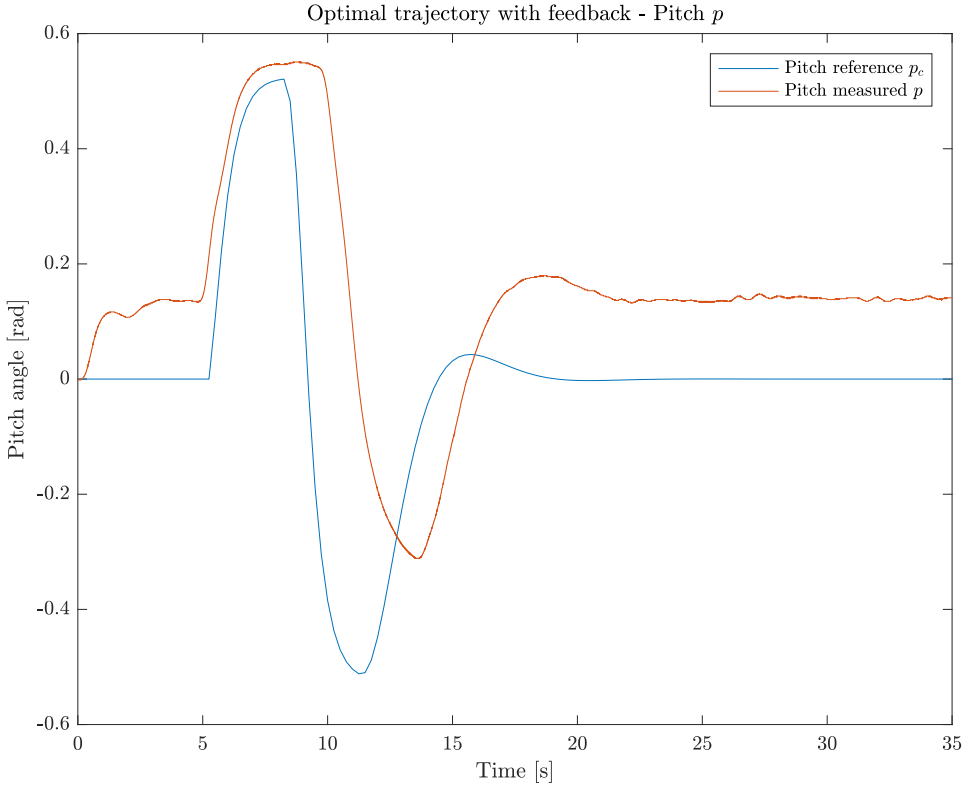


Figure 7: Optimal control with feedback – Plot of the pitch p

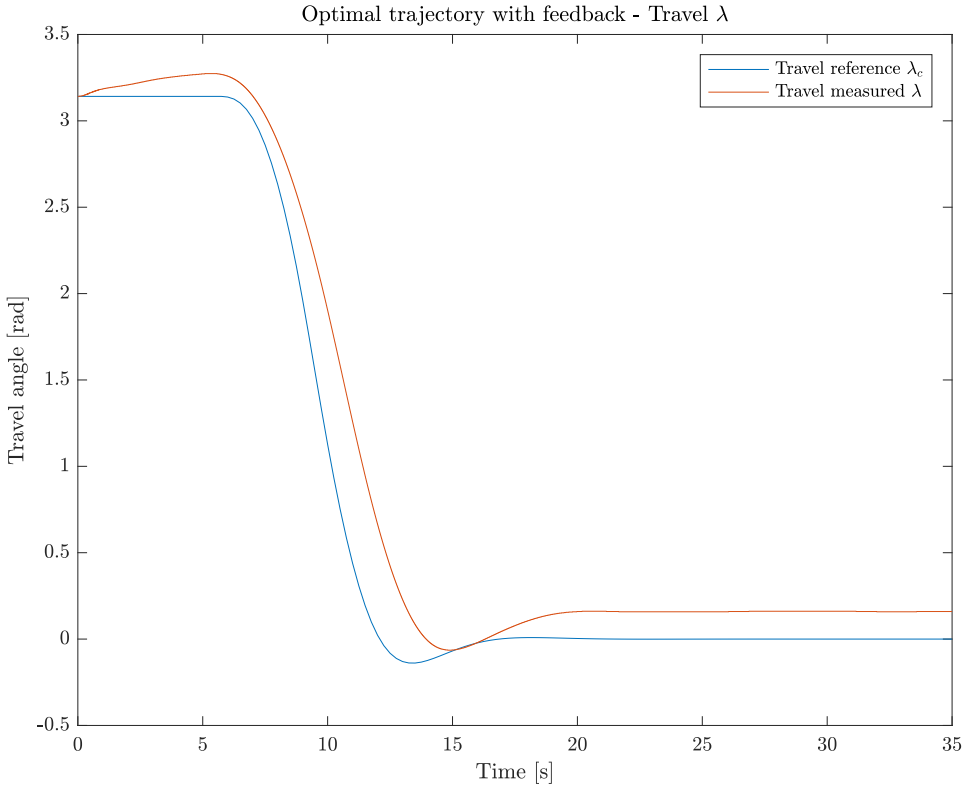


Figure 8: Optimal control with feedback – Plot of the travel λ

4.3 An Alternative Approach: MPC Controller

An alternative strategy would be to use an MPC controller. With the LQ controller, the optimal trajectory and input sequence is only computed once, and the feedback compensates for any deviations. An MPC controller computes a new optimal trajectory $\mathbf{x}^*$ and input sequence $\mathbf{u}^*$ for every iteration based on feedback from state $\mathbf{x}$, which is used as the new initial state for an optimization problem with one less iteration. Then the first element in the newly computed optimal input sequence $\mathbf{u}_\mathbf{k}^*$ is applied, before the $\mathbf{x}_{k+1}$ is again fed into the optimization layer.

Input blocking can also be used with an MPC controller to reduce the amount of calculations, reducing total computational costs. Input blocking is a technique where the amount of optimization variables are reduced by grouping input variables $\mathbf{u}_k$ into groups with identical values. This will result in a less optimal trajectory, but if the right balance is found between the length and timing of the input blocks, this can lead to a good result with much less computational costs.

A disadvantage with implementing the MPC controller instead of using the LQ controller with feedback is that computing a new optimal input

sequence $\mathbf{u}^*$ requires some computation time, and to do this for each iteration, the sampling time would probably need to be longer, resulting in a less responsive system and a worse optimal trajectory.

However, a MPC controller would be able to compute the most optimal input at each iteration point based on feedback from the previous iteration, in contrast to just adjusting the input based on the current deviation from the optimal trajectory, as the LQ controller does. If it had been possible to compute the optimal input sequence quicker, using a MPC controller would have resulted in a more optimal sequence than what is achieved with the LQ controller.

5 Problem 10.4 Optimal Control of Pitch/Travel and Elevation with and without feedback

In this part, an optimal trajectory is to be calculated in two dimensions. The optimal trajectory will now be calculated with respect to both travel λ and elevation e.

5.1 Augmenting the State Space Model

To calculate an optimal trajectory in two dimensions it is necessary to include e in the state vector $\mathbf{x}$, and to include a new elevation setpoint e_c for the system in the manipulated variable $\mathbf{u}$. This gives the following state space vectors:

$$\mathbf{x} = \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} p_c \\ e_c \end{bmatrix} \tag{24}$$

The matrices $\mathbf{A_c}$ and $\mathbf{B_c}$ are found using eq. (1). By discretizing using the Forward Euler method described in eq. (2), the following discrete state space model is found:

$$\mathbf{x}_{k+1} = (\Delta t \mathbf{A_c} + \mathbf{I})\mathbf{x}_k + \Delta t \mathbf{B_c} \mathbf{u}_k \tag{25}$$

with:

$$\begin{aligned} \mathbf{A_c} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3 K_{ep} & -K_3 K_{ed} \end{bmatrix} \\ \mathbf{B_c} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3 K_{ep} \end{bmatrix} \end{aligned} \tag{26}$$

5.2 Calculating the Optimal Trajectory

The optimal trajectory $\mathbf{x}^*$ will move the helicopter from the same initial point $\mathbf{x}_0$ to $\mathbf{x}_f$ as in section 3, but this time with the state variable $\mathbf{x}$ modified

to include the elevation. That is, from $\mathbf{x}_0 = [\lambda_0 \ 0 \ 0 \ 0 \ 0 \ 0]^\top$ to $\mathbf{x}_f = [\lambda_f \ 0 \ 0 \ 0 \ 0 \ 0]^\top$ where $\lambda_0 = \pi$ and $\lambda_f = 0$.

As in section 3, the optimal trajectory will be calculated by solving an optimization problem formulated as the quadratic program given by eq. (9), where a nonlinear constraint will also be implemented on the elevation e.

The criteria to be minimized is now

$$\phi = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2 \tag{27}$$

giving $\mathbf{H}$ as defined in eq. (11) with

$$\mathbf{Q} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{P} = \begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix} \tag{28}$$

where q_1 and q_2 are used as weights on the pitch setpoint p_{ci} and the elevation setpoint e_{ci}, respectively. As in section 3 larger weights penalize the output. A goal for tuning these cost variables is to ensure that the inputs are balanced in such a way that the helicopter can follow the path freely. In this experiment, setting $q_1 = q_2 = 1$ proved to be give good results, giving a rather fast response without overshooting.

The quadratic program will still have equality constraints determined by the system. That is, the equality constraints will be on the same form as those used in section 3, given in eq. (15) and eq. (16), but now with the system matrices $\mathbf{A}$ and $\mathbf{B}$ given by eq. (25) and eq. (26).

The same pitch constraint used in section 3 given by eq. (14) will be used in this part. The nonlinear inequality constraint to be implemented on the elevation e is described by the following expression:

$$e_k \geq \alpha \exp(-\beta(\lambda_k - \lambda_t)^2) \ \forall k \in \{1, \ldots, N\} \tag{29}$$

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

Finally, the optimization problem is solved. As the elevation constraint given by (29) is a nonlinear constraint (appendix A.5.2), an SQP-type algorithm is used to solve the optimization problem. Therefore, `fmincon` is used to solve the quadratic program. In order to simulate the whole trajectory from start to finish, $N = 60$ is used instead of the $N = 40$ from the problem description for the experiment. This gives some extra computational time, but this is not a problem as `fmincon` only runs once when calculating the optimal trajectory before actually starting the helicopter.

5.3 Implementing the Optimal Trajectory

The optimal input sequence is then implemented, both with and without feedback. The plotted results with and without feedback can be seen in fig. 10, fig. 11 and fig. 12. By trial and error we used the following $\mathbf{Q_{lq}}$ and $\mathbf{R_{lq}}$ matrices in the LQ controller, resulting in the gain matrix $\mathbf{K_{lq}}$ given in eq. (31).

$$\mathbf{Q_{lq}} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 30 \end{bmatrix} \quad \text{and} \quad \mathbf{R_{lq}} = \begin{bmatrix} 0.1 & 0 \\ 0 & 1 \end{bmatrix} \tag{30}$$

$$\mathbf{K_{lq}} = \begin{bmatrix} -1.5409 & -4.3189 & 2.0328 & 0.5652 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.1226 & 4.1445 \end{bmatrix} \tag{31}$$

Figure 13 shows the pitch reference fed into the pitch controller. This is linear combination of λ, $\dot{\lambda}$, p, $\dot{p}$ which can be seen from the gain matrix eq. (31). As we use numerical derivation, the contribution from $\dot{\lambda}$ and $\dot{e}$ results in a noisy reference, which is clearly seen in the plot. A solution to this is to use estimated states, which would give a smoother reference for p_c and e_c.

The oscillations of p_c explains the slightly oscillating behaviour of the measured p. It is also evident that there is a time delay in in the system (mainly in changing the speed of the rotors), and the reference is moving too fast for the helicopter to follow. A suggestion to fix this would be to add a constraint on how fast the input can change, but this is not attempted in the experiment.

Comparing the plots with and without feedback, it is clear that using feedback gives a better performance, as both travel and elevation follows the reference much closer. Using feedback results in stationary travel when the reference is met, with the exception of the stationary deviation described in section 4. Overall, this is an improved behavior compared to the drift that can be see in the plot of the travel without feedback in fig. 11.

5.4 Satisfying the elevation constraint

In fig. 12 the measured elevation and the optimal elevation trajectory are plotted together with the elevation constraint as defined with regards to the actual measured travel. According to the plot, it is clear that the elevation criteria is not satisfied by the measured elevation trajectory as the helicopter is not able to fly above the defined elevation constraint. By looking at the

elevation level required to satisfy the elevation constraint, it is also clear that neither does the optimal trajectory. This is due two causes: restrictions in the used mathematical model, and a step size that is too short.

The first cause, restrictions in the mathematical model, can be seen in fig. 12 as the deviation between the measured trajectory and the optimal trajectory. This deviation is because of the fact that the optimal trajectory is calculated based on a linearized and thus inaccurate model when moving away from the linearization point. This makes the calculated optimal trajectory harder to follow for the helicopter in reality, no matter how well the controllers giving the feedback effect are tuned. The restrictions in the mathematical model are discussed further in the next subsection.

The second, and perhaps most notable problem, is that the optimal trajectory does not seem to satisfy the elevation constraint by itself, even before actually testing the helicopter in reality. This us due to the step size used in the optimization problem. A step time that is too short will result in an underestimate of the elevation constraint, as the solver will not be able to calculate the peak value correctly. This is illustrated in fig. 9, where the optimal trajectory elevation is plotted with regards to the optimal travel. The first subplot shows that the optimal trajectory is actually correct in the sense that it satisfies the constraint given the step size, as the optimal trajectory is above the constraint value for any given step.

The step size used in the optimization problem is $\Delta t = 0.25$ with $N = 60$. A halving of the step size and a doubling of the number of steps clearly shows that the optimal trajectory will come closer to satisfying the actual elevation constraint, as shown in the second subplot in fig. 9.

5.5 State Decoupling and Weaknesses in the Mathematical Model

In the used mathematical model eq. (1), the sum of inertia for the elevation is calculated neglecting the pitch of the rotors. In reality the moment generated by the rotors would depend on the pitch, as a high pitch angle would make the rotors contribute less to the elevation. As this is not compensated for in the model, the pitch setpoint needed to achieve the optimal trajectory will be underestimated. This decoupling is also seen from $\mathbf{K_{lq}}$ eq. (31), where the pitch setpoint p_c is calculated as a linear combination of λ, $\dot{\lambda}$, p and $\dot{p}$ and elevation setpoint e_c is calculated as a linar combination of e and $\dot{e}$.

It is also seen in the results, as the helicopter moves far less than 180° when tested in reality (without the optimal trajectory feedback), when it, according to calculations based on the model, should have travelled a full 180°. It should still be noted that some error is of course to be expected, as there are other factors (like friction) which have also been neglected.

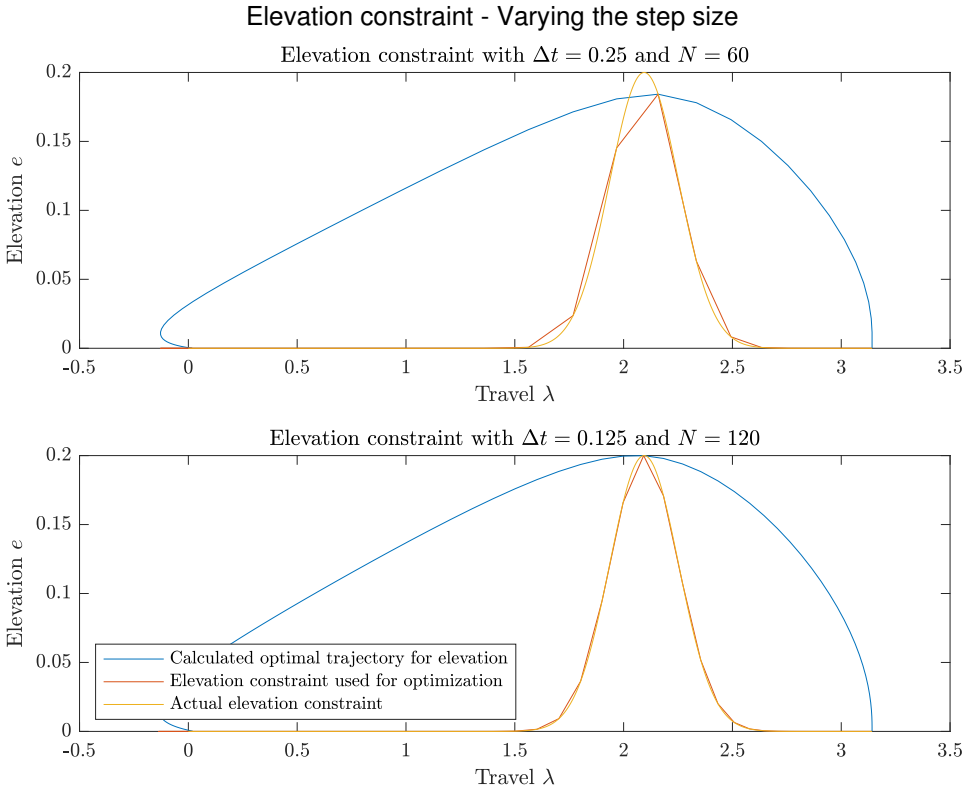


Figure 9: Illustrating how varying the step size changes the elevation constraint and thus the optimal trajectory.

The mathematical model could have been improved by compensating for the effect the pitch has on the sum of inertia for the elevation. This can be done by introducing a cosine decomposition, but as this results in a nonlinear system, it also results in a more complicated optimization problem, as well as a more complicated control system.

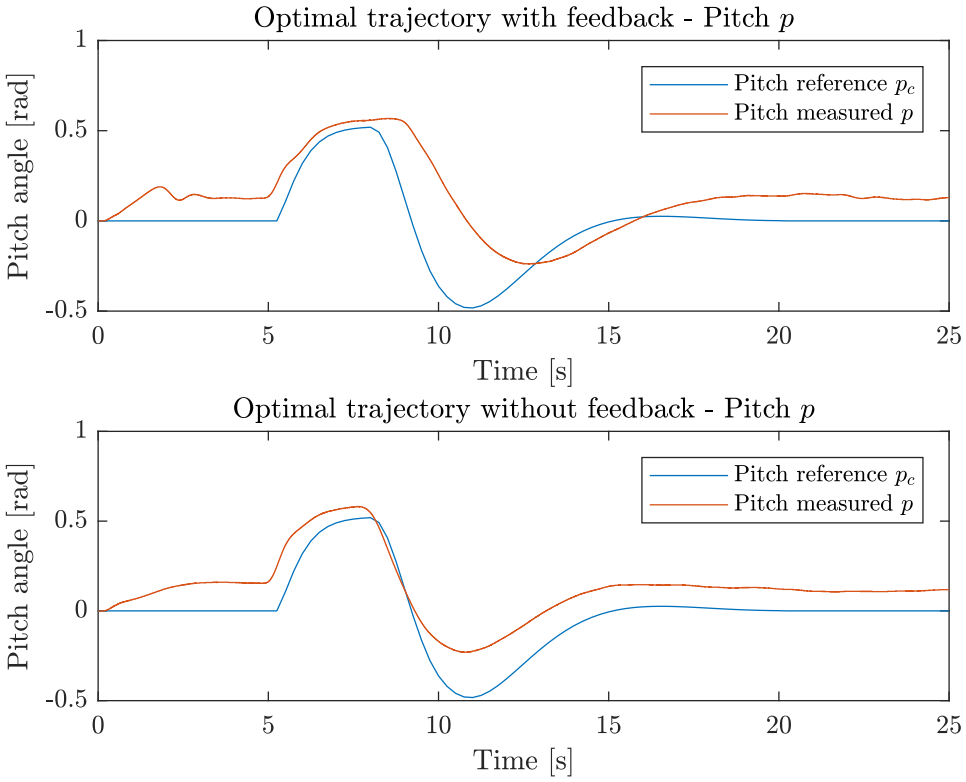


Figure 10: Optimal control with and without feedback – Plot of the pitch p

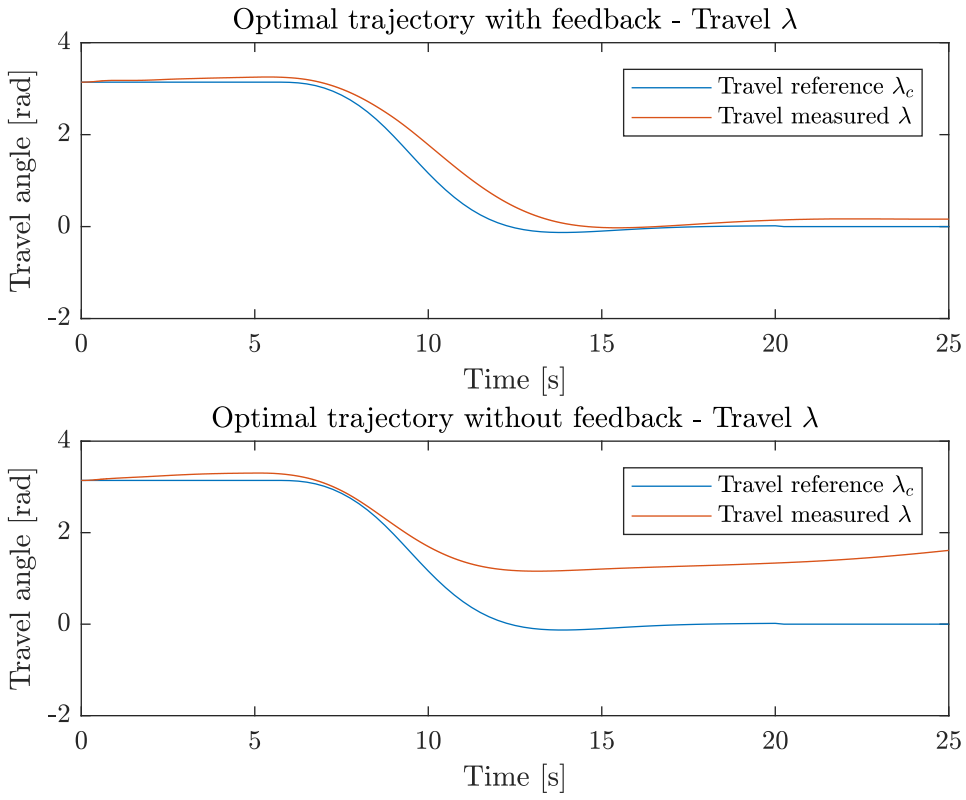


Figure 11: Optimal control with and without feedback – Plot of the travel λ

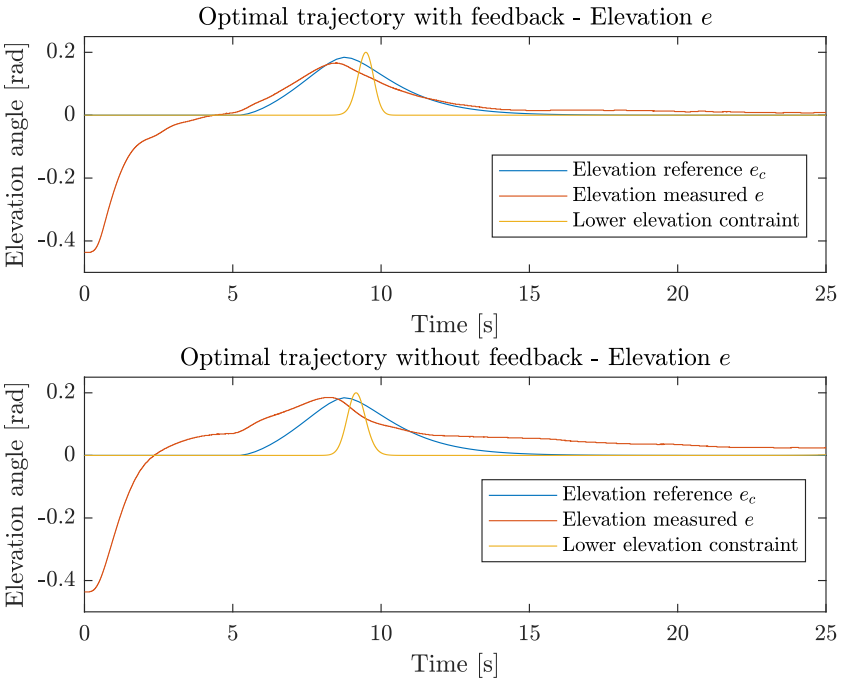


Figure 12: Optimal control with and without feedback – Plot of the elevation e

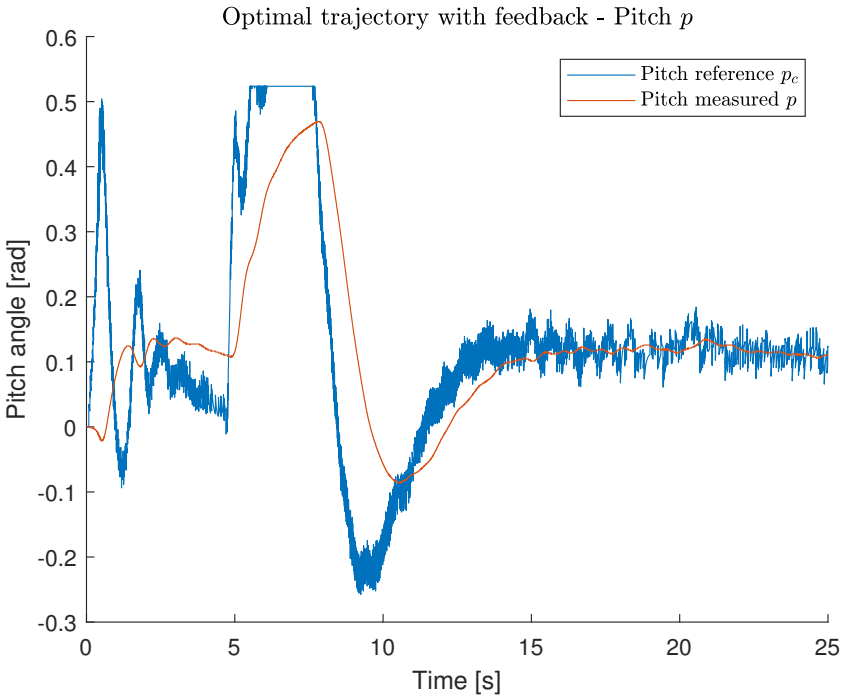


Figure 13: Optimal control with and without feedback – Plot of the pitch p

6 Conclusion

In this lab experiment, optimal trajectories are calculated and implemented on a QUARC helicopter using MatLab and Simulink.

Early in the experiment, the objective is to rotate the helicopter 180° around the travel axis by controlling the pitch without introducing any restrictions in the control of the elevation angle. A quadratic cost function is minimized to find the optimal input sequence, taking constraints on the pitch angle into consideration. At first this input sequence is fed directly into the helicopter, but this does not result in the desired behaviour as the helicopter is not able to follow the optimal trajectory. To prevent this, a LQ controller is implemented to achieve feedback control. As a result the helicopter follows its optimal path much more accurately, giving a good 180° rotation about the travel axis.

Later in the experiment, the objective is to control the helicopter with restrictions added on the elevation in addition to the previous constraints on the pitch. A nonlinear constraint on the elevation angle is introduced and a SQP-type algorithm is used to solve the new optimization problem. Again this is attempted both with and without feedback. As before, feedback resulted in a more accurate helicopter which followed the optimal trajectory more closely than it was able to without feedback.

This experiment has clearly shown the weaknesses of using optimal control by itself without feedback. Due to weaknesses in the used mathematical models and computational inaccuracies, an optimal trajectory with a corresponding optimal input sequence, calculated purely based on theory will never be sufficient in reality. In order to achieve a desirable result, feedback control has to be used.

As a conclusion, this experiment has shown that optimal control is no doubt a powerful tool when used together with feedback control.

A MATLAB Code

A.1 init.m

```
% Initialization for the helicopter assignment in TTK4135.
% Run this file before you execute QuaRC -> Build.

% Updated spring 2018, Andreas L. Floten
% Updated Spring 2019, Joakim R. Andersen

close all;
clc;

% Set default interpreter as Latex
set(groot, 'defaultTextInterpreter', 'latex');
set(groot, 'defaultAxesTickLabelInterpreter', 'latex');
set(groot, 'defaultLegendInterpreter', 'latex');

% The encoder for travel for Helicopter 2 is different from the rest.
travel_gain = -2; %


%% Physical constants
m_h = 0.4; % Total mass of the motors.
m_g = 0.03; % Effective mass of the helicopter.
l_a = 0.65; % Distance from elevation axis to helicopter body
l_h = 0.17; % Distance from pitch axis to motor

% Moments of inertia
J_e = 2 * m_h * l_a *l_a;          % Moment of interia for elevation
J_p = 2 * ( m_h/2 * l_h * l_h);    % Moment of interia for pitch
J_t = 2 * m_h * l_a *l_a;          % Moment of interia for travel

% Identified voltage sum and difference
V_s_eq = 6.0; % Identified equilibrium voltage sum.
V_d_eq = 0.8; % Identified equilibrium voltage difference.

% Model parameters
K_p = m_g*9.81; % Force to lift the helicopter from the ground.
K_f = K_p/V_s_eq; % Force motor constant.
K_1 = l_h*K_f/J_p;
K_2 = K_p*l_a/J_t;
K_3 = K_f*l_a/J_e;
K_4 = K_p*l_a/J_e;
```

```
41
42 %% Pitch closed loop syntesis
43 % Controller parameters
44 w_p = 1.8; % Pitch controller bandwidth.
45 d_p = 1.0; % Pitch controller rel. damping.
46 K_pp = w_p^2/K_1;
47 K_pd = 2*d_p*sqrt(K_pp/K_1);
48 Vd_ff = V_d_eq;
49
50 % Closed loop transfer functions
51 Vd_max = 10 - V_s_eq; % Maximum voltage difference
52 deg2rad = @(x) x*pi/180;
53 Rp_max = deg2rad(15); % Maximum reference step
54 s = tf('s');
55 G_p = K_1/(s^2);
56 C_p = K_pp + K_pd*s/(1+0.1*w_p*s);
57 L_p = G_p*C_p;
58 S_p = (1 + L_p)^(-1);
59
60 plot_pitch_response = 0;
61 if plot_pitch_response
62     figure()
63     step(S_p*Rp_max); hold on;
64     step(C_p*S_p*Rp_max/Vd_max);
65     legend('norm error', 'norm input')
66     title('Pitch closed loop response')
67 end
68
69 %% Elevation closed loop analysis
70 % Controller parameters
71 w_e = 0.8; % Elevation controller bandwidth.
72 d_e = 1.0; % Elevation controller rel. damping.
73 K_ep = w_e^2/K_3;
74 K_ed = 2*d_e*sqrt(K_ep/K_3);
75 K_ei = K_ep*0.1;
76 Vs_ff = V_s_eq;
77
78 % Closed loop transfer functions
79 Vs_max = 10 - V_s_eq; % Maximum voltage sum
80 Re_max = deg2rad(10); % Maximum elevation step
81 G_e = K_3/(s^2);
82 C_e = K_ep + K_ed*s/(1+0.1*w_e*s) + K_ei/s;
83 L_e = G_e*C_e;
84 S_e = (1 + L_e)^(-1);
```

```
85
86 plot_elev_response = 0;
87 if plot_elev_response
88     figure()
89     step(S_e*Re_max);
90     hold on;
91     step(C_e*S_e*Re_max/Vs_max);
92     legend('norm error', 'norm input')
93     title('Elevation closed loop response')
94 end
```

A.2 Problem 10.2.3

A.2.1 problem_10_2_3.m

```
1  % TTK4135 - Helicopter lab
2  % Hints/template for problem 2.
3  % Updated spring 2018, Andreas L. Floten
4
5  %% System
6  % Discrete time system model. x = [lambda r p p_dot]'
7  delta_t = 0.25;                    % sampling time
8  A1 = [0 1 0 0;                     % A1 = (TA + I)
9        0 0 -K_2 0;
10       0 0 0 1;
11       0 0 -K_1*K_pp -K_1*K_pd] * delta_t + eye(4);
12
13 B1 = [0 0 0 K_1*K_pp]' * delta_t;  % B1 = TB;
14
15
16 %% Initial values
17 x1_0 = pi;                         % Lambda
18 x2_0 = 0;                          % r
19 x3_0 = 0;                          % p
20 x4_0 = 0;                          % p_dot
21 x0 = [x1_0 x2_0 x3_0 x4_0]';       % Initial values
22
23 % Time horizon and initialization
24 N  = 100;                          % Time horizon for states
25 M  = N;                            % Time horizon for inputs
26 % Number of states and inputs
27 mx = size(A1,2);                   % Number of states
28 mu = size(B1,2);                   % Number of inputs
29
30 z  = zeros(N*mx+M*mu,1);           % Initialize z for time horizon
```

```
31 z0 = z;                                  % Initial value for optimization
32
33 %% Bounds
34 ul      = -30*pi/180;                    % Lower bound on control
35 uu      = -ul;                           % 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 %% Constraints
43 % Generate constraints on measurements and inputs
44 [vlb,vub]       = gen_constraints(N,M,xl,xu,ul,uu);
45 vlb(N*mx+M*mu)  = 0;                     % Set last input to zero
46 vub(N*mx+M*mu)  = 0;                     % Set last input to zero
47
48 % Generate the matrix Q and the vector c
49 % (objective function weights in the QP problem)
50 Q1 = zeros(mx,mx);
51 Q1(1,1) = 2;                             % Weight on state x1
52 Q1(2,2) = 0;                             % Weight on state x2
53 Q1(3,3) = 0;                             % Weight on state x3
54 Q1(4,4) = 0;                             % Weight on state x4
55 P1 = q;                                  % Weight on input
56
57 % function Q = gen_q(Q1,P1,N,M)
58 Q = gen_q(Q1,P1,N,M);                    % Generate Q
59 c = zeros(size(Q,1),1);                  % Generate c
60
61 % Generate equality constraints
62 Aeq = gen_aeq(A1,B1,N,mx,mu);            % Generate Aeq
63 beq = zeros(size(Aeq,1),1);              % Generate beq
64 beq(1:mx) = A1 * x0;
65
66 %% Solve QP problem with linear model
67 [z,lambda] = quadprog(Q,c,[],[],Aeq,beq,vlb,vub,x0);
68
69 %% Extract control inputs and states
70 u  = [z(N*mx+1:N*mx+M*mu);z(N*mx+M*mu)];
71
72 x1 = [x0(1);z(1:mx:N*mx)];
73 x2 = [x0(2);z(2:mx:N*mx)];
74 x3 = [x0(3);z(3:mx:N*mx)];
```

```
x4 = [x0(4);z(4:mx:N*mx)];

%% Add zero-padding
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];
```

A.3 Problem 10.2.4

A.3.1 problem_10_2_4.m

```
%% Problem 2.4
%% Initialization and model definition
clear all; close all; clc;

addpath('../');
addpath('../help_functions');
addpath('../Problem_10_2_3');
init;

q = 1;
problem_10_2_3;

%% Create matrices to be used in Simulink

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

u_star = [t' u];
x_star = [t' x1 x2 x3 x4];
```

A.4 Problem 10.3

A.4.1 problem_10_3.m

```
clear all; close all; clc;

addpath('../');
addpath('../help_functions');
addpath('../Problem_10_2_4');

```

```
init;
problem_10_2_4;

%% LQ state-feedback
q_1 = 100;  % Travel
q_2 = 10;   % Travel rate
q_3 = 10;   % Pitch
q_4 = 10;   % Pitch rate

r_1 = 1 ;   % Pitch setpoint (input)

Q_lq = diag([q_1 q_2 q_3 q_4]);
R_lq = r_1;

[K_lq,S_lq,e_lq] = dlqr(A1,B1,Q_lq,R_lq);
```

A.5 Problem 10.4

A.5.1 problem_10_4.m

```
%clear all;
addpath('../');
addpath('../help_functions');
init;

%% System
% Discrete time system model. x = [lambda r p p_dot e e_dot]'
delta_t = 0.25;                 % Sampling time
A_d = [0 1 0 0 0 0;             % A_d = (TA + I)
       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] * delta_t + eye(6);

B_d = [0 0;                     % B_d = TB;
    0 0;
    0 0;
    K_1*K_pp 0;
    0 0;
    0 K_3*K_ep] * delta_t;

% Eulers forward method: x[k+1] = A_d*x[k] + B_d*u_k

%% Initial values
```

```
26 x1_0 = pi;                          % Lambda
27 x2_0 = 0;                           % r
28 x3_0 = 0;                           % p
29 x4_0 = 0;                           % p_dot
30 x5_0 = 0;                           % e evt -25
31 x6_0 = 0;                           % e_dot
32 x0 = [x1_0 x2_0 x3_0 ....
33     x4_0 x5_0 x6_0]';               % Initial values
34
35 global N;
36 N = 60;                             % Prediction horizon
37 M = N;                              % Number of input steps
38 mx = size(A_d,2);                   % Number of states
39 mu = size(B_d,2);                   % Number of inputs
40
41 %% Bounds
42 ul      = [-30*pi/180, -inf]';      % Lower bound on control
43 uu      = -ul;                      % Upper bound on control
44
45 xl      = -Inf*ones(mx,1);          % Lower bound on states (no bound)
46 xu      = Inf*ones(mx,1);           % Upper bound on states (no bound)
47 xl(3)   = ul(1);                    % Lower bound on state x3
48 xu(3)   = uu(1);                    % Upper bound on state x3
49
50 %% Constraints
51 % Generate constraints on measurements and inputs
52 [vlb,vub]       = gen_constraints(N,M,xl,xu,ul,uu);
53 vlb(N*mx+M*mu)  = 0;                % Set last input to zero
54 vub(N*mx+M*mu)  = 0;                % Set last input to zero
55
56 % Equality constraints
57 Aeq = gen_aeq(A_d,B_d,N,mx,mu); % Generate Aeq
58 beq = zeros(size(Aeq,1),1);     % Generate beq
59 beq(1:mx) = A_d * x0;
60
61 %% Generate costfunction matrices
62
63 % Initialize z
64 z = zeros(N*mx + M*mu,1);
65 z0 = z;                             % z0(1) = pi? Nei;
66 z0(1) = pi;
67
68 Q1 = zeros(mx,mx);
69 Q1(1,1) = 2;                        % Weight on state x1
```

```
70  Q1(2,2) = 0;                        % Weight on state x2
71  Q1(3,3) = 0;                        % Weight on state x3
72  Q1(4,4) = 0;                        % Weight on state x4
73  Q1(5,5) = 0;                        % Weight on state x5
74  Q1(6,6) = 0;                        % Weight on state x6
75
76  q_1 = 1;                            % Weight on pitch setpoint
77  q_2 = 1;                            % Weight on elevation setpoint
78  P1 = 2*diag([q_1 q_2]);
79
80  Q = gen_q(Q1,P1,N,M);  % Generate Q
81  nonlcon = @nonlincon;
82
83  %% Costfunction
84  fun = @(z) 0.5*z'*Q*z;
85
86  options = optimoptions('fmincon','Display','iter','Algorithm','sqp');
87  tic
88  z = fmincon(fun, z0, [],[],Aeq, beq, vlb, vub, nonlcon, options);
89  toc
90  %% LQ state-feedback
91  q_1 = 5;                            % Travel
92  q_2 = 1;                            % Travel rate
93  q_3 = 1;                            % Pitch
94  q_4 = 0.5;                          % Pitch rate
95  q_5 = 10;                           % Elevation
96  q_6 = 30;                           % Elevation rate
97
98  r_1 = 1;                            % Pitch setpoint (input)
99  r_2 = 0.1;                          % Elevation setpoint (input)
100 Q_lq = diag([q_1 q_2 q_3 q_4 q_5 q_6]);
101 R_lq = diag([r_1 r_2]);
102
103 [K_lq,S_lq,e_lq] = dlqr(A_d,B_d,Q_lq,R_lq);
104
105 %% Extract control inputs and states
106 u1 = [z(N*mx+1:2:N*mx+M*mu);z(N*mx+M*mu-1)];
107 u2 = [z(N*mx+2:2:N*mx+M*mu);z(M*mx+M*mu)];
108
109 x1 = [x0(1);z(1:mx:N*mx)];
110 x2 = [x0(2);z(2:mx:N*mx)];
111 x3 = [x0(3);z(3:mx:N*mx)];
112 x4 = [x0(4);z(4:mx:N*mx)];
113 x5 = [x0(5);z(5:mx:N*mx)];
```

```
x6 = [x0(6);z(6:mx:N*mx)];

%% Add zero-padding
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];

%% Create matrices to be used in Simulink
t = 0:delta_t:delta_t*(length(x1)-1);

u_star = [t' u1 u2];
x_star = [t' x1 x2 x3 x4 x5 x6];
```

A.5.2 nonlincon.m

```
function [c,ceq] = nonlincon(x)
    % Inequality constraint
    global N;

    alpha = 0.2;
    beta = 20;
    lambda_t = 2*pi/3;
    c = zeros(N,1);
    for k = 1:N
        c(k) = alpha*exp(-beta*(x((k-1)*6 + 1) ...
        -lambda_t)^2)-x((k-1)*6 + 5);
    end
    ceq = [];
end
```

B Simulink Diagrams

B.1 Problem 10.2 – Optimal Control of Pitch and Travel without Feedback

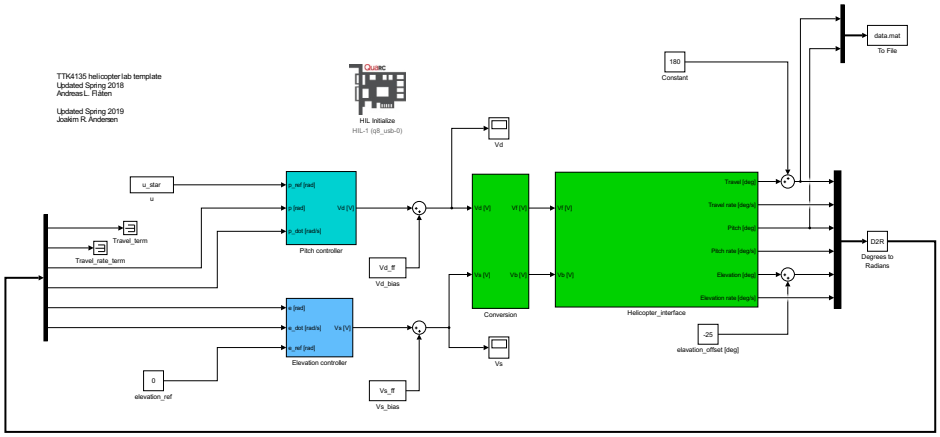


Figure 14: Simulink diagram for Problem 10.2

B.2 Problem 10.3 — Optimal Control of Pitch and Travel with Feedback (LQ)

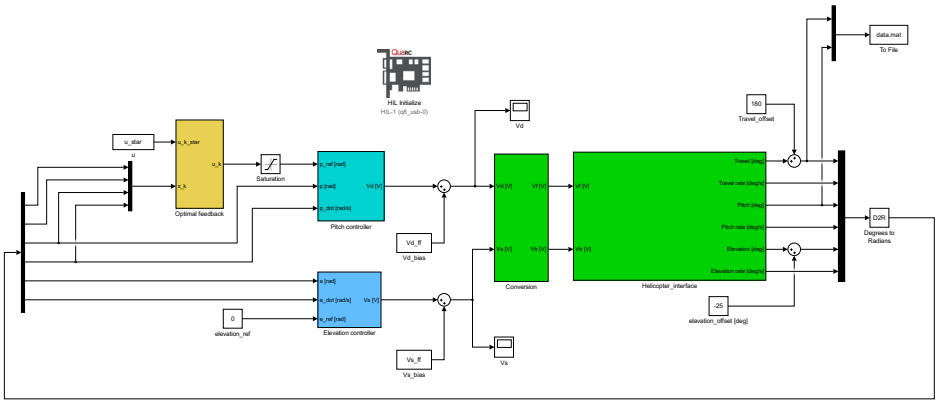


Figure 15: Simulink diagram for Problem 10.3

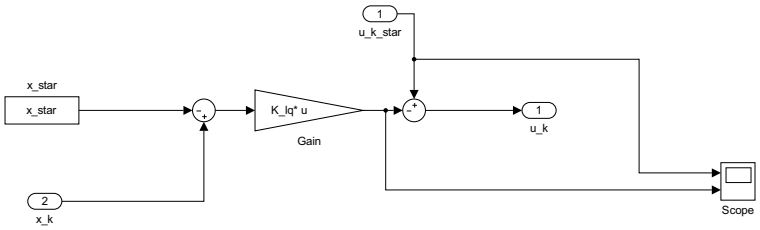


Figure 16: LQR Controller implemented in problem 10.3

B.3 Problem 10.4 - Optimal Control of Pitch/Travel and Elevation with and without feedback

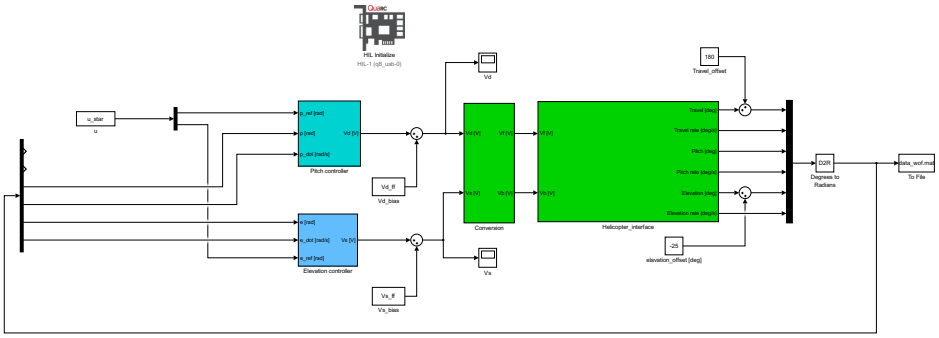


Figure 17: Simulink of problem 10.4 without feedback

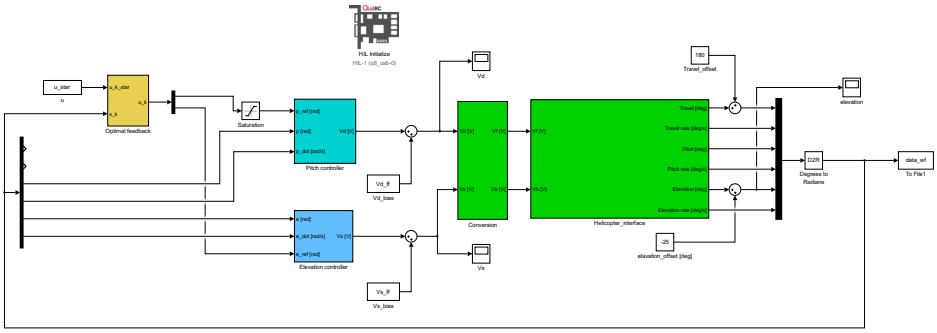


Figure 18: Feedback implemented in problem 10.4