TTK4135: Helicopter Lab Report

Group 44

Håkon Elfving
Kjetil S. Gjerden
Per Arne Kjelsvik

April 27, 2017

Department of Engineering Cybernetics

Abstract

Optimization algorithms and control theory is to be combined and used to control a stationary helicopter arm with 3 degrees of freedom (DOF) stabilised by a PID controller. An optimal trajectory is calculated using a linearised model, and applied to the plant in the open loop case subjected to linear constraints and the closed loop case subjected to linear and nonlinear constraints. Feedback control is realised by implementing a linear quadratic controller. This report shows how mathematical optimisation can be used in real world control, how the use of feedback control improves the use of a simplified model as well as how a linearised model can be controlled when modified with nonlinear constraints.

Contents

1 Introduction **1**

2 Problem Description & Mathematical model **2**
2.1 Equations of motion . . . 2

3 Optimal Control of Pitch/Travel without Feedback **6**
3.1 State Space Formulation . . . 6
3.2 Model Discretization . . . 7
3.3 Calculating the Optimal Path . . . 7
3.4 Results and Discussion . . . 9

4 Optimal Control of Pitch/Travel with Feedback (LQ) **11**
4.1 Implementing LQ Control . . . 11
4.2 Results . . . 12
4.3 MPC Discussion . . . 12

5 Optimal Control of Pitch/Travel and Elevation with and without Feedback **15**
5.1 Extended State Space Formulation . . . 15
5.2 Model Discretization . . . 15
5.3 Optimal Path with Nonlinear Inequality Constraints . . . 15
5.4 Results . . . 16
5.4.1 Open Loop . . . 16
5.4.2 Closed Loop . . . 19
5.5 State decoupling . . . 21
5.6 Alternative Constraints . . . 21

6 Conclusion/Discussion **23**

References **24**

A Appendix A - `MATLAB` code **25**
A.1 `MATLAB` - Initialisation . . . 25
A.2 `MATLAB` - Section 3 . . . 26
A.3 `MATLAB` - Section 4 . . . 28
A.4 `MATLAB` - Section 5 . . . 28

B Appendix B - `SIMULINK` models **31**
B.1 `SIMULINK` - Section 3 . . . 31
B.2 `SIMULINK` - Section 4 . . . 31
B.3 `SIMULINK` - Section 5.4.1 . . . 32
B.4 `SIMULINK` - Section 5.4.2 . . . 32

1 Introduction

This lab discusses how to, in practice, use optimization in the context of control. In the lab assignment, a 3 DOF helicopter system is to be studied. Given a seemingly accurate model of the system, the effects different controllers have on the system have been studied. First optimal trajectories and input, calculated through minimizing a quadratic optimization problem, was applied to the system with pitch as input, both with and without feedback. The feedback was implemented using a linear quadratic controller. Then a nonlinear constraint was put on the elevation, and once again, optimal input and trajectories was calculated and applied to the system with both pitch and elevation as input, with and without feedback. Lastly it was attempted to modify the given constraints of the problem to possibly increase the precision of the system. The system was implemented and run in `SIMULINK`.

Section 2 gives a closer description of the mathematical model used, as well as the lab setup. Section 3 presents how optimal control without feedback is used in control of the plant in question. In section 4, optimal control is augmented by implementing feedback using a linear quadratic controller and is compared to the open loop case in section 3. Section 5 presents how a nonlinear inequality elevation constraint is added to the optimization problem and how this nonlinear problem is solved.

2 Problem Description & Mathematical model

The laboratory setup consists of a movable arm with two rotors (the helicopter) in one end and a counterweight in the other, fixed to a stationary base (see fig. 1-2). The arm can be moved up and down around an elevation axis and rotate around a vertical axis (travel). The helicopter body itself can also rotate around an axis normal to the arm (pitch). These three angles are measured with optical position sensors. The helicopter can be controlled using two DC motors, whose applied voltage is assumed proportional to the force exerted by the helicopter propellers. The model parameters and values are shown in table 1.

The equations of motion for the pitch, travel- and elevation angle were derived on the basis of Newton's second law for rotation and moment balances:

$$\sum \vec{\tau}_z = \sum \vec{r}_i \times \vec{F}_i = \sum (\vec{m}_i \vec{r}_i^{\,2}) \vec{\alpha}_z = J\alpha_z \,. \tag{1}$$

Where $\vec{\tau}$ is the torque about the given axis (z), α is the angular acceleration and J is the moment of inertia around the given point, given a rigid body and that the cross-product is defined [3]. In our assignement we used the illustrations shown in fig. 2 as a basis for our mathematical models.

2.1 Equations of motion

$$J_e \ddot{e} = l_a K_f V_s - T_g \tag{2a}$$

$$\Downarrow$$

$$\ddot{e} = K_3 V_s - \frac{T_g}{J_e}, \quad K_3 = \frac{l_a K_f}{J_e} \tag{2b}$$

To control the elevation of the helicopter, the following PD controller is implemented

$$V_s = K_{ep}(e_c - e) - K_{ed}\dot{e} \quad , K_{ep}, K_{ed} > 0 \tag{3}$$

Combining eq. (2a) and eq. (3):

$$\ddot{e} = -K_3 K_{ed}\dot{e} - K_3 K_{ep} e - \frac{T_g}{J_e} + K_3 K_{ep} e_c \tag{4}$$

A further simplification that can be done is to include an integral term (resulting in a PID controller), and assume that the integral term counteracts the effect from the constant term $-\frac{T_g}{J_e}$, cancelling each other. This assumption gives rise to the following differential equation for the elevation angle:

$$\ddot{e} + K_3 K_{ed}\dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c \tag{5}$$

Following the same argumentation for the pitch and travel angle, gives

$$J_p \ddot{p} = K_f l_h V_d \tag{6a}$$

$$\Downarrow$$

$$\ddot{p} = K_1 V_d, \quad K_1 = \frac{K_f l_h}{J_p} \tag{6b}$$

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

giving the following closed loop system for the pitch

$$\ddot{p} + K_1 K_{pd} \dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c \,. \tag{6d}$$

And finally for the travel angle:

$$J_t \dot{r} = -K_p l_a \sin p \tag{7}$$

or

$$\dot{r} = -K_2 p, \quad K_2 = \frac{K_p l_a}{J_t} \tag{8}$$

$$\dot{\lambda} = r \,, \tag{9}$$

assuming p is small such that $\sin p \approx p$.

$$\tag{10}$$

Putting all these together, gives the following system of equations:

$$\ddot{e} + K_3 K_{ed} \dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c \tag{11a}$$

$$\ddot{p} + K_1 K_{pd} \dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c \tag{11b}$$

$$\dot{\lambda} = r \tag{11c}$$

$$\dot{r} = -K_2 p \,. \tag{11d}$$

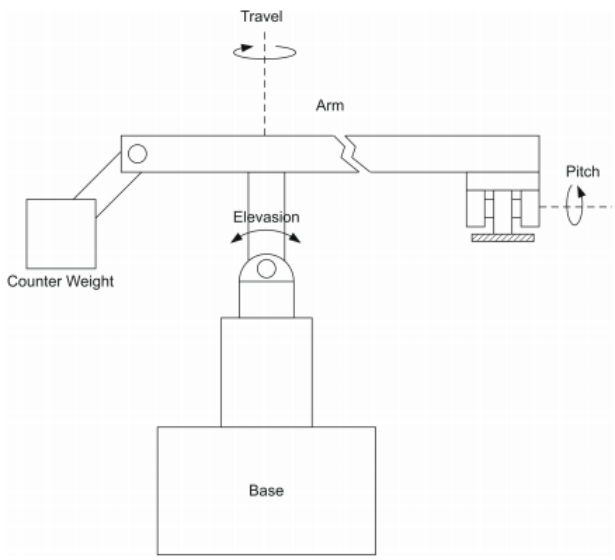

Figure 1: Cross section view of the helicopter system [image credit: NTNU, department of cybernetics]

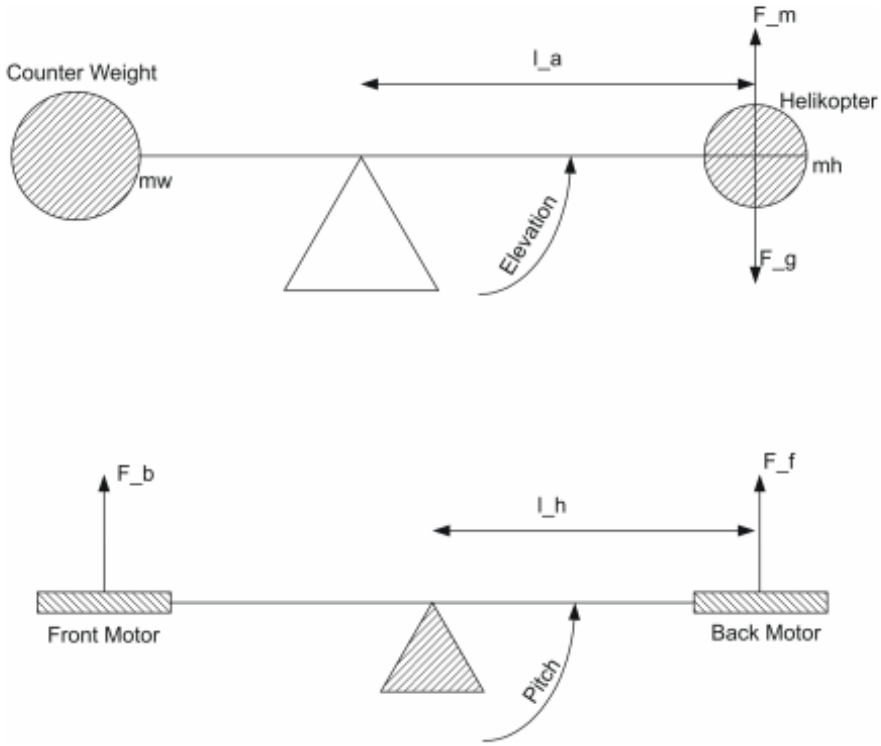

Figure 2: Free-body sketch of the helicopter system [image credit: NTNU, department of cybernetics]

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: Variable description.

Symbol	variable
p	Pitch
p_c	Setpoint for pitch
λ	Travel
r	Travel rate
e	Setpoint for elevation
e_c	Elevation reference
V_f	Voltage input, front motor
V_b	Voltage input, back motor
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	Torque exerted by gravity
x_0	Initial point
x_f	State reference

3 Optimal Control of Pitch/Travel without Feedback

In this part we calculate an optimal trajectory x^* and a corresponding optimal input sequence u^*, which is used to drive the helicopter from a defined initial point x_0 to a given reference point x_f. This is done by formulating the problem as a convex quadratic optimization problem with a given objective function. Furthermore we introduce a five second zero-padding on the calculated states and inputs for a more controlled test environment (in all subsequent parts). This is visible in the plots as a greyed-out zone marked by a dotted red line.

3.1 State Space Formulation

Starting off, it has now been assumed that elevation stays at zero. From the model derived in eq. (11) the continuous state-space system can be written on the form

$$\dot{\boldsymbol{x}} = \mathbf{A}_c \boldsymbol{x} + \mathbf{B}_c \boldsymbol{u}$$

with the tensors:

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K1K_{pp} & -K_aK_{pd} \end{bmatrix} \tag{12}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1K_{pp} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \end{bmatrix}. \tag{13}$$

Thus the model of the helicopter is solely based upon the plant's travel, travel rate, pitch and pitch rate. With elevation being disregarded we won't have the associated feedback and PID control shown in the illustration given in fig. 3. The model itself includes the two lower layers in fig. 3, i.e. the basic controller(s) and the plant itself. The objective of this assignment will largely be to calculate the optimal input which is to be fed to the basic control layer and plant, which happens in the optimization layer shown in fig. 3.

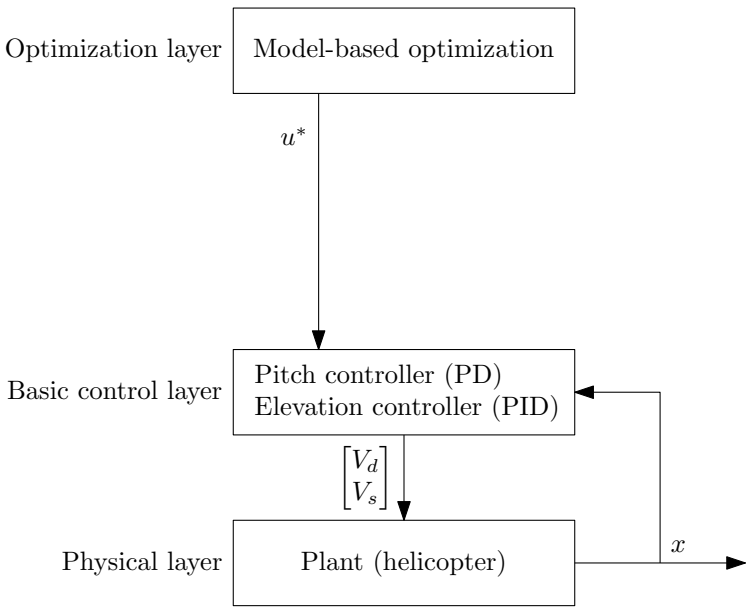

Figure 3: Control hierarchy of the system [image credit: NTNU, department of cybernetics]

3.2 Model Discretization

Forward Euler [2, p. 521] is defined (on state-space form) as

$$\frac{\boldsymbol{x_{k+1}} - \boldsymbol{x_k}}{\Delta t} = \mathbf{A}_c \boldsymbol{x}_k + \mathbf{B}_c \boldsymbol{u}_k \,.$$

Solving for the next timestep we get

$$\boldsymbol{x_{k+1}} = \underbrace{(\mathbf{I} - \Delta t \mathbf{A}_c)}_{\mathbf{A}_d} \boldsymbol{x}_k + \underbrace{\Delta t \mathbf{B}_c}_{\mathbf{B}_d} \boldsymbol{u}_k \,, \tag{14}$$

where

$$\mathbf{A}_d = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & -\Delta t K_2 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & -\Delta t K_1 K_{pp} & 1 - \Delta t K_1 K_{pd} \end{bmatrix}, \quad \mathbf{B}_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta t K_1 K_{pp} \end{bmatrix}, \quad \Delta t = 0.25\text{s} \,.$$

3.3 Calculating the Optimal Path

An optimal trajectory is to be calculated using the objective function given by

$$\Phi = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q p_{ci}^2, \;\; q \geq 0 \,. \tag{15}$$

The stated objective function (15) results in the following Hessian matrix:

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

The optimal path from the initial point $\boldsymbol{x}_0 = [\lambda_0 \;\; 0 \;\; 0 \;\; 0]^\top$ to the reference point $\boldsymbol{x}_f = [\lambda_f \;\; 0 \;\; 0 \;\; 0]^\top$, where $\lambda_0 = \pi$ and $\lambda_f = 0$, is found by solving the following inequality constrained QP:

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \;\; & f(z) = \frac{1}{2} z^\top \boldsymbol{G} z \\ & \text{s.t.} \\ & A_{eq} z = b_{eq} \\ & |u_k| = |p_k| \leq \frac{30\pi}{180}, \quad i \in \{1, ..., N\} \,, \end{aligned} \tag{16}$$

where $N = 100$, $n = N(n_x + n_u)^\dagger$, $\mathbf{z} = \begin{bmatrix} x_1^\top & \dots & x_N^\top, & u_0 & \dots & u_{N-1}^\top \end{bmatrix}^\top$, $\mathbf{G}$ is defined as

$$\mathbf{G} = \begin{bmatrix} \mathbf{Q} & 0 & \dots & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \mathbf{Q} & \ddots & & \vdots \\ \vdots & & \ddots & q & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 & q \end{bmatrix}$$

$^\dagger n_x$ = number of states, n_u = number of inputs

and

$$\mathbf{A}_{eq} = \begin{bmatrix} \mathbf{I} & 0 & \dots & \dots & 0 & -\mathbf{B} & 0 & \dots & \dots & 0 \\ -\mathbf{A} & \mathbf{I} & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -\mathbf{A} & \mathbf{I} & 0 & \dots & \dots & 0 & -\mathbf{B} \end{bmatrix}, \quad \mathbf{B}_{eq} = \begin{bmatrix} \mathbf{A}\mathbf{x}_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

According to eq. (11) in the lab assignment [1], travel is controlled by integrating the pitch twice. This results in suboptimal control, since there are possibilities for large error propagations, and might be partly to blame for some of the result discrepancies.

`MATLAB` was used to calculate the optimal trajectory, using the function `quadprog`. To begin with, `init08.m`, handed out along with the lab assignment text, is run (to initialize necessary components for the problem and `SIMULINK`, such as K_p and $V_{d_{max}}$). Thereafter, we declare $\mathbf{A}$ and $\mathbf{B}$ given in eq. (12) and eq. (13), and then discretize the system using forward Euler where Δt is defined as 0.25s. Next, we declare $\mathbf{x}_0$, $\mathbf{x}_f$, $\mathbf{Q}$, $q(= 0.1)$, n_x, n_u, and N as stated above. To fill in $\mathbf{A}_{eq}$ and the $\mathbf{G}$ matrix, the function `gena2` and `genq2`, which was handed out along with the project, was used. Next, $\mathbf{z}$ is initialized. Now we're ready to implement the constraints:

Listing 1: `MATLAB` code for input and state constraints of the QP-problem.

```
% Bounds
pk    = 30*pi/180;                    % Absolute boundary on pitch
ul    = -pk;                          % Lower bound on control -- u1
uu    = pk;                           % Upper bound on control -- u1

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] = genbegr2(N,M,xl,xu,ul,uu);
vlb(n)    = 0;                        % We want the last input to be zero
vub(n)    = 0;                        % We want the last input to be zero
```

Here another given help function, `genbegr2`, was used to generate boundaries on the input and states of our system. Everything is now set to solve the problem and find optimal trajectories using the `MATLAB` function `quadprog`:

Listing 2: `MATLAB quadprog` implementation to find $\mathbf{z}$ and λ.

```
[z, lambda] = quadprog(Q,c,[],[],A_eq,B_eq,vlb,vub,x0);
```

`quadprog` solves the QP-problem using an interior-point-algorithm. The optimal trajectory has now been found, all that remains then is to extract the input and states from $\mathbf{z}$, add 5s padding before and after the input, and import the data to `SIMULINK`. The full code for this exercise can be found in appendix A.2, as well as a full `SIMULINK` model in appendix B.1.

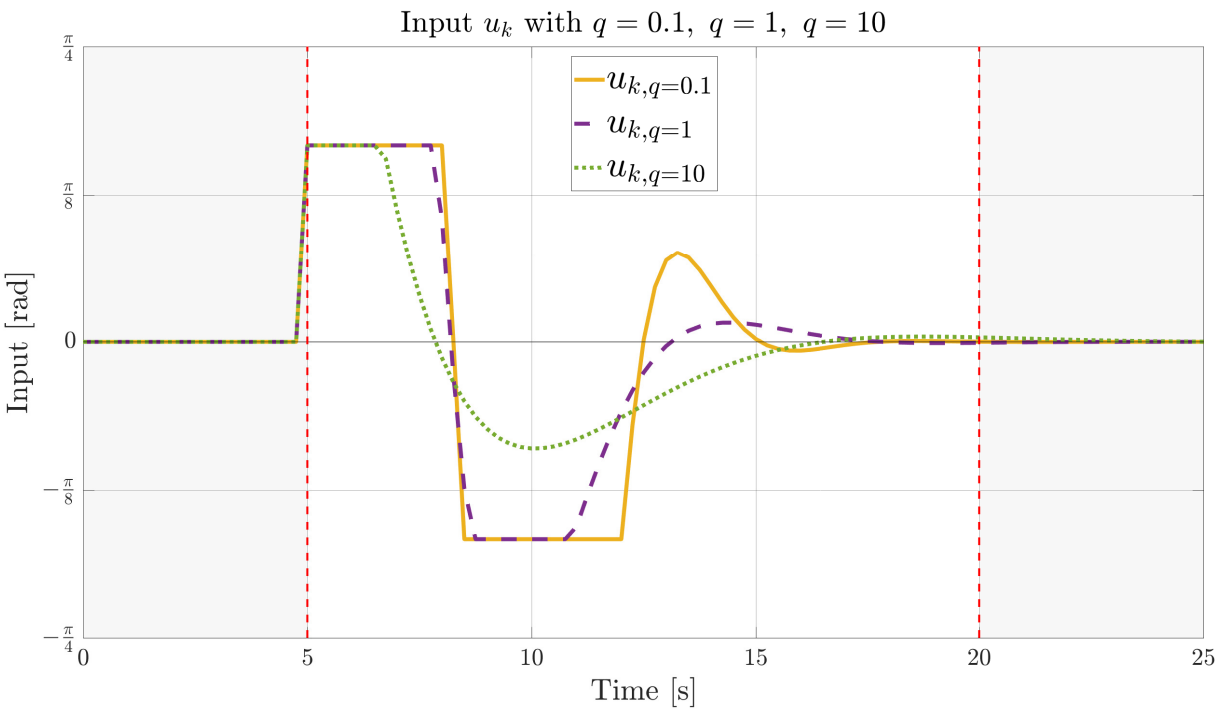

Figure 4: Calculated optimal input for different weights.

3.4 Results and Discussion

When running the system with different weights, we got the results shown in fig. 4 and fig. 5. Increasing q gives a more conservative input usage which results in a slower more gentle control, however there is potential for large deviations. Decreasing q means we allow for 'more input', which results in more aggressive control and a faster reference convergence. Fig. 5 shows that increasing q results in a trajectory which never completely reaches the intended end point. From this we see that keeping $q = 0.1$ results in fast control as well as letting the helicopter reach the intended reference angle of 180 degrees, albeit with some overshoot. The overshoot is in part because of a simplified, imperfect linearised model and the fact that we approximate the travel by integrating the pitch twice, eq. (11d), to control the helicopter. Fig. 6 shows that the helicopter fails to stay at x_f. This is due to the fact that we rely on open loop control, thus the controller observes the helicopter reaching the intended point and assumes stationarity, not taking the plant momentum or other factors into account. The use of feedback control can prevent the helicopter from overshooting by including these factors.

Looking at the objective function in eq. (15), we observe that as $\lambda = \lambda_f$ the only term remaining is qp_{ci}^2, which means that the input minimizing the objective function is $u = 0$. In practice this means that when $\lambda = \lambda_f$ is reached, u is set to zero, halting our controller, resulting in an overshoot/drift-off due to the helicopter momentum. This comes from not taking travel rate into account, so the helicopter might just pass through λ_f without stopping. Also, there's no inner controller for travel, so deviations is to be expected.

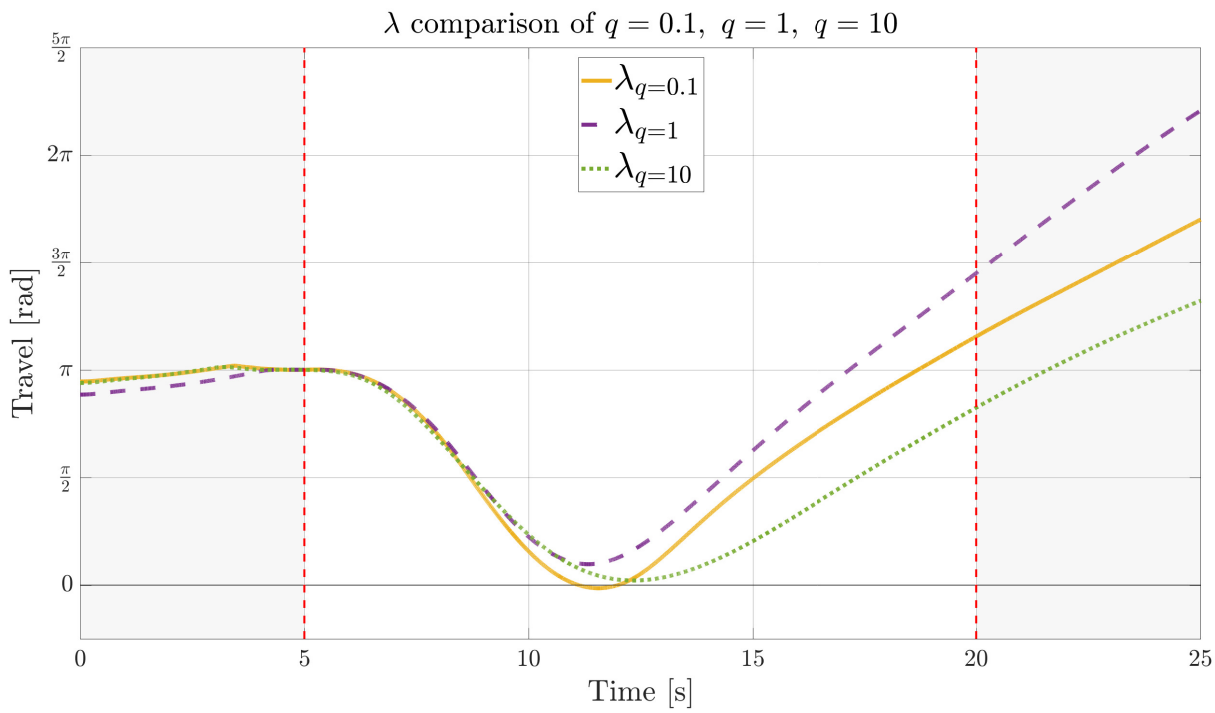

Figure 5: Measured helicopter travel for different weights.

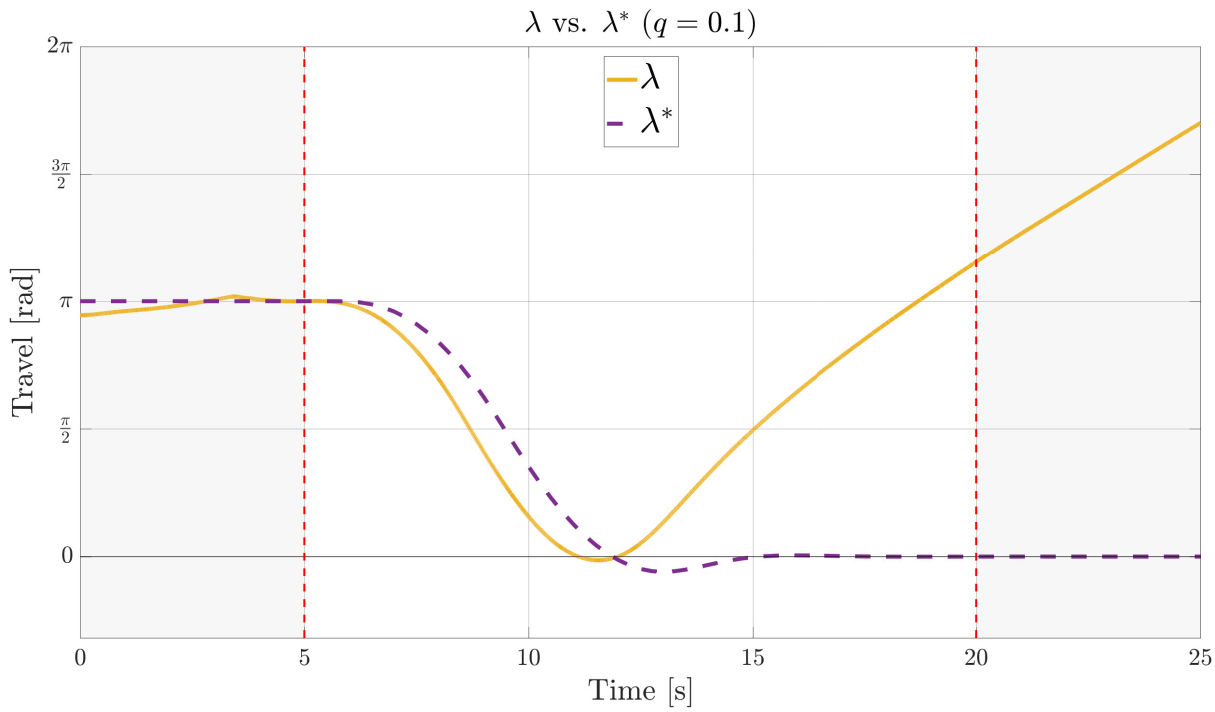

Figure 6: Calculated optimal travel vs the measured travel, no feedback.

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

In this part we introduce feedback in the optimal controller by implementing a linear quadratic (LQ) controller, which minimizes a quadratic criteria for a linear model as shown in eq. (18).

4.1 Implementing LQ Control

State feedback is now introduced to the system to correct deviations caused by errors in the model. This is done by implementing an LQ controller, by using the manipulated variable

$$\boldsymbol{u}_k = \boldsymbol{u}_k^* - \mathbf{K}^T(\boldsymbol{x}_k - \boldsymbol{x}_k^*) \tag{17}$$

where $\boldsymbol{u}_k^*$ and $\boldsymbol{x}_k^*$ is the optimal input sequence and trajectory, calculated the same way as in section 3.3. Every time step where a deviation from $\boldsymbol{x}^*$ is observed, the gain matrix $\mathbf{K}$ will modify the manipulated variable. In this exercise, the gain matrix $\mathbf{K}$ was found by minimizing the quadratic objective function

$$J = \sum_{i=0}^{\infty} \Delta x_{i+1}^T \boldsymbol{Q} \Delta x_{i+1} + \Delta u_i^T \boldsymbol{R} \Delta u_i, \quad Q \succeq 0, \;\; R \succ 0 \tag{18}$$

for a linear model

$$\Delta x_{i+1} = A\Delta x_i + B\Delta u_i \tag{19}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are weight matrices for the states and inputs respectively. The weighting tensors were set to:

$$\mathbf{Q} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \quad \mathbf{R} = 0.1 \;\; .$$

These were chosen to penalise deviations in travel and travel rate, while giving more freedom to pitch and pitch rate, as well as allowing faster control with $\mathbf{R} = 0.1$.
The gain matrix $\mathbf{K}$ is calculated using the `MATLAB` command `dlqr`, solving the stationary algebraic Riccatti equation. The resulting $\mathbf{K}$ matrix became:

$$K = \begin{bmatrix} -3.1247 & -7.3149 & 3.0336 & 0.7792 \end{bmatrix}$$

Listing 3: `MATLAB` code for finding optimal gain matrix $\boldsymbol{K}$

```
%% [——— Task 10.3.1 Q,R and K (LQR) ———]
Q_lqr = diag([3 1 .05 .1]);
R_lqr = diag([.1]);
K_lqr = dlqr(A,B,Q_lqr,R_lqr);
```

In `SIMULINK` the feedback was implemented with several blocks, for example `from workspace` to import x^*, t and u^*. Our implementation can bee seen in fig. 7, where we chose to build the equations in a cascading style. The full `SIMULINK` implemtenation can bee seen in appendix B.2.

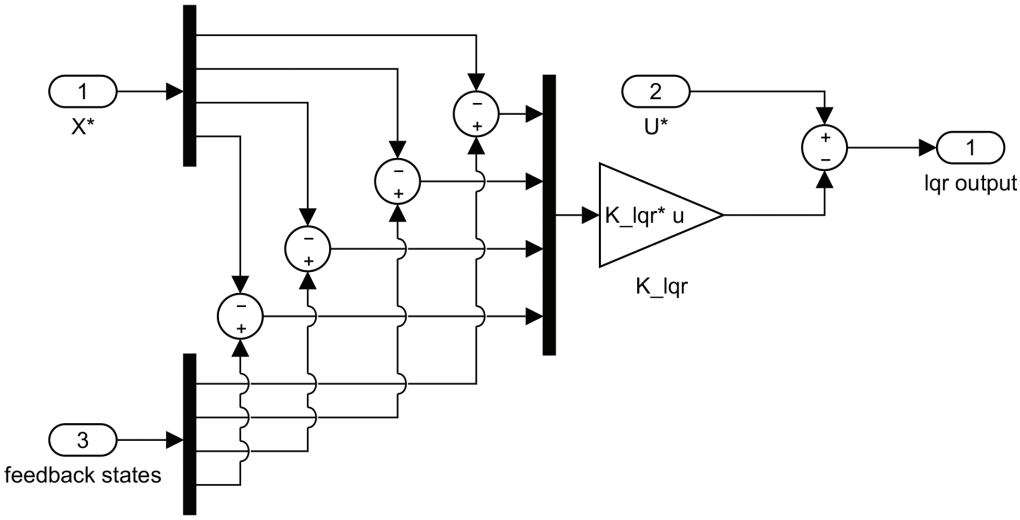

Figure 7: Cascading LQR implementation in `SIMULINK`

4.2 Results

Comparing the open loop response in fig. 6 with the closed loop response in fig. 8, it is clearly seen that the inclusion of feedback has a significant effect on the plant, effectively removing the drift-off. The helicopter still doesn't quite reach x_f due to our comparatively free pitch weighting, causing a less aggressive system overall, as well as model simplifications discussed in section 2.1. This is also evident in fig. 9, as the measured pitch is far from following the optimal pitch route exactly.

4.3 MPC Discussion

Realizing MPC would mean solving the QP problem for each time step, utilizing the first calculated input on the next iteration. One advantage of doing this is that it would likely result in more precise control (due to feedback being incorporated in the opt. problem), but it has the obvious disadvantage of being more time, and computationally, expensive in action. The LQ controller on the other hand, only solves the QP problem once, and sticks to that result for the entire horizon, meaning that the optimal gain can also be calculated and stored in advance. The MPC can be said to be resistant to disturbances, being able to simply calculate a new trajectory if it loses its previous one. The LQ controller however would try to get to the previously optimal path, and thus it might oscillate around the trajectory. Since our system has fast dynamics, the use of MPC would most likely not yield much better performance compared to a faster, simpler form of feedback control. Fig. 10 explains the role MPC would play in the control hierarchy if we were to replace the LQ controller with it.

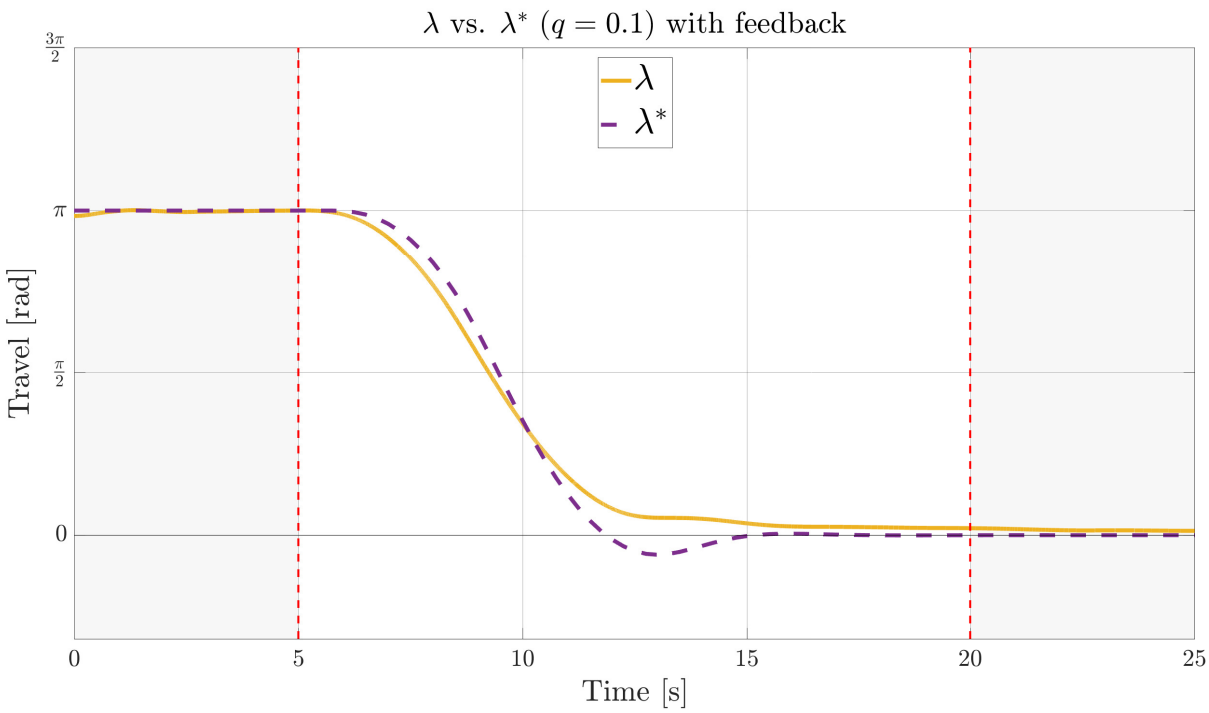

Figure 8: Measured helicopter travel vs optimal travel with feedback.

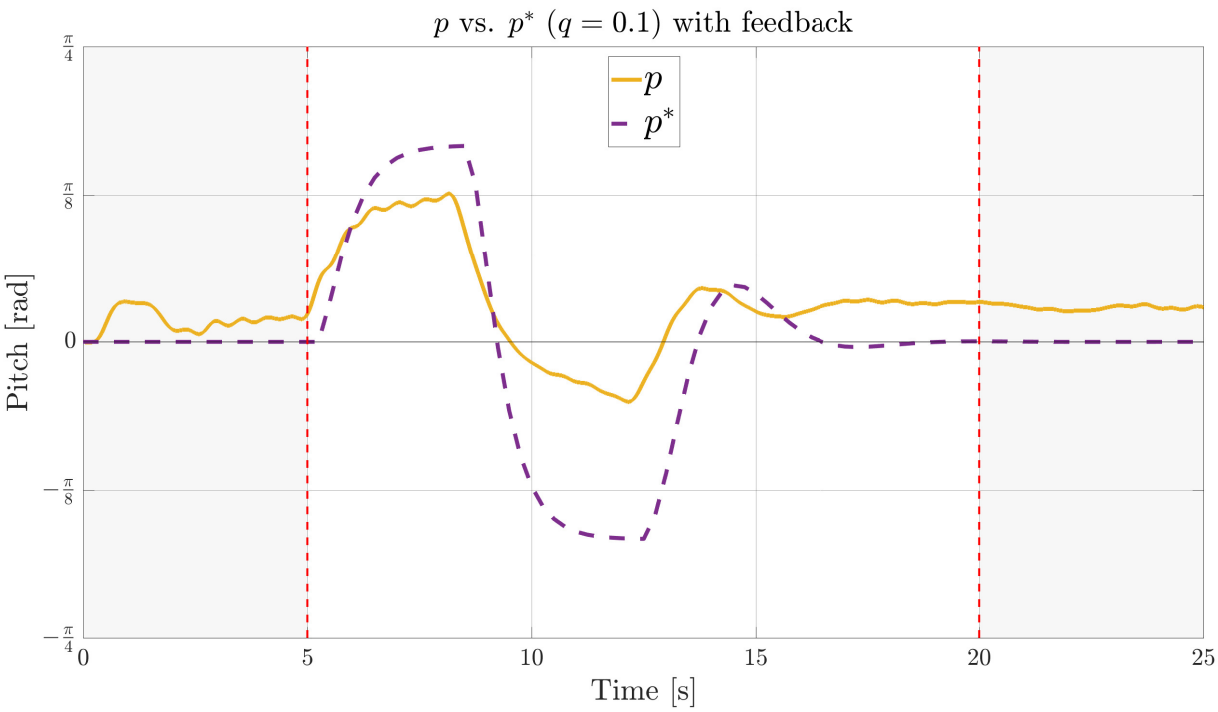

Figure 9: Measured helicopter pitch vs optimal pitch.

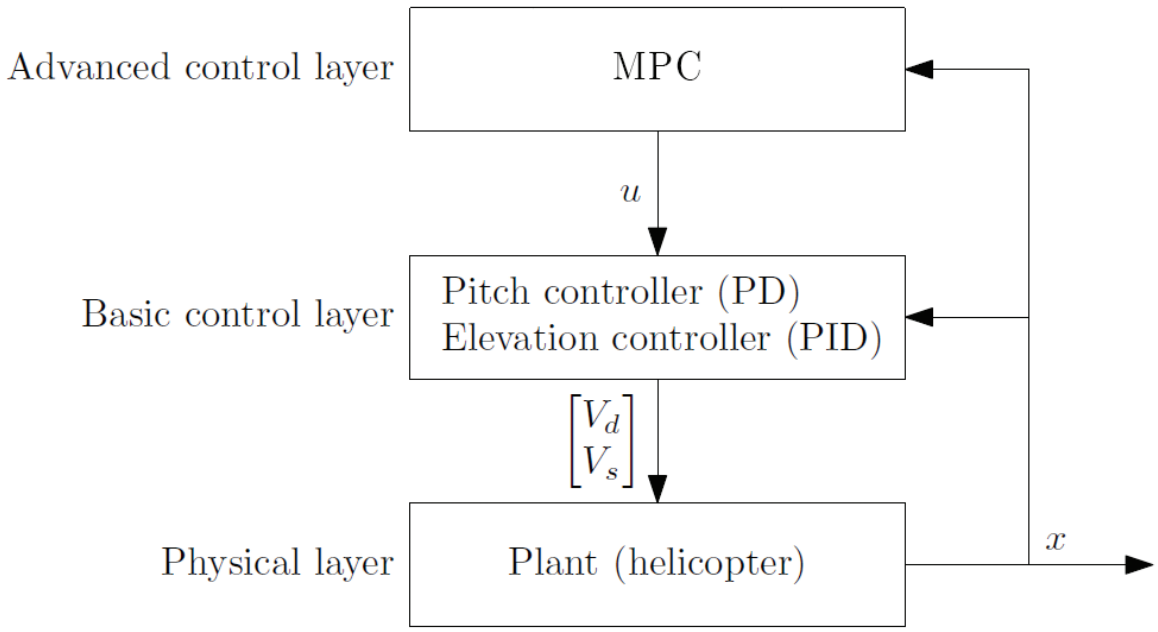

Figure 10: The structure of the control layers using an MPC instead of LQR

5 Optimal Control of Pitch/Travel and Elevation with and without Feedback

In this part we will calculate an optimal trajectory for the helicopter in two dimensions, travel and elevation, by extending our model to include e and $\dot{e}$. The helicopter will now be moved from the initial point to the reference point while avoiding a set obstacle causing the elevation angle to change mid-flight. The obstacle is implemented in the optimization problem as a nonlinear elevation constraint, thus the problem must now be solved using a nonlinear solver.

5.1 Extended State Space Formulation

$$x = [\lambda \;\; r \;\; p \;\; \dot{p} \;\; e \;\; \dot{e}]^T, \qquad u = [p_c \;\; e_c]^T$$

Including elevation and elevation rate in eq. (11), the new state space model becomes

$$\begin{bmatrix} r \\ \dot{r} \\ \dot{p} \\ \ddot{p} \\ \dot{e} \\ \ddot{e} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3K_{ep} & -K_3K_{ed} \end{bmatrix}}_{A_c} \begin{bmatrix} \lambda \\ r \\ p \\ \dot{p} \\ e \\ \dot{e} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3K_{ep} \end{bmatrix}}_{B_c} \begin{bmatrix} p_c \\ e_c \end{bmatrix}.$$

5.2 Model Discretization

Using the same method as in section 3.2, we get the following discretized model approximation

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & -\Delta t K_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & -\Delta t K_1 K_{pp} & 1-\Delta t K_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & -\Delta t K_e K_{ep} & 1-\Delta t K_3 K_{ed} \end{bmatrix}}_{A_d} x_k + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \Delta t K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & \Delta t K_3 K_{ep} \end{bmatrix}}_{B_d} u_k,$$

where $x = [\lambda \;\; r \;\; p \;\; \dot{p} \;\; e \;\; \dot{e}]^\top$ and $u = [p_c \;\; e_c]^\top$.

5.3 Optimal Path with Nonlinear Inequality Constraints

To calculate the optimal path from $x_0 = [\lambda_0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^\top$ to $x_f = [\lambda_f \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^\top$, we minimize the objective function Φ given as

$$\begin{aligned} \Phi &= \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2, \quad q_1, q_2 > 0, \;\; N = 40 \\ &= \sum_{i=0}^{N-1} x_{i+1}^\top Q x_{i+1} + u_i^\top R u_i, \end{aligned}$$

where

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

The following nonlinear constraint was imposed on the elevation:

$$c(x_k): \ \ e_k \geq \alpha \exp[-\beta(\lambda_k - \lambda_t)^2], \quad \forall \ k \in \{1, ..., N\},$$

where $\alpha = 0.2$, $\beta = 20$ and $\lambda_t = \frac{2\pi}{3}$, while still keeping the equality constraints similar to those stated in eq. (16). Since the optimization problem now includes nonlinear constraints, the standard QP solver is insufficient. The `MATLAB` code for this exercise is very similar to the code from section 3.3 (declaring constants and matrices, forward Euler, generating $\mathbf{A}_{eq}$ etc.) to begin with. The major difference initially, is that we also declare `N`, `alpha`, `lambda_t`, `beta`, `nx` (as `global` variables). This is because we need to pass the constraint $c(x_k)$ to our solver at every point in time, which we do through our self-written function `constraints`:

Listing 4: `constraints.m`, which calculates the elevation constraint for every time step.

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

Instead of `quadprog`, which can't handle nonlinear constraints, the function `fmincon` is used, which uses a SQP (Sequential Quadratic Programming)-based algorithm. After setting up the system, generating constraints etc. the problem is solved with `fmincon`:

Listing 5: Solving the nonlinear QP-problem using `fmincon` and `constraints.m`.

```
% Find Z and lambda by using SQP-based fmincon
f   = @(z) 1/2*z'*G*z;
opt = optimoptions('fmincon','Algorithm','sqp','MaxFunEvals',400000);

tic
[Z, ZVAL, EXITFLAG] = ...
     fmincon(f, z0, [], [], A_eq, B_eq, vlb, vub, @constraints, opt);
toc
```

Now we have the optimal input and states of the system. All that remains is to extract them, add 5s padding before and after the input, before importing them in `SIMULINK`. The full code for this exercise can be found in the appendix (appendix A.4), as well as the `SIMULINK` model in appendix B.3 for the open-loop implementation.

5.4 Results

5.4.1 Open Loop

The results from controlling the helicopter using optimal states and input and no feedback is shown in fig. 11 - fig. 13. Fig. 11 shows that the helicopter follows the optimal path fairly well,

but still suffers from the drift-off caused by open loop control. It is also shown that the optimal trajectory shows a big spike in travel rate at $t \approx 15$s which is not ideal for the system, especially not at the end of the input sequence, since the input is turned off right after. In this open loop case, the helicopter fails miserably at keeping up with the desired/necessary elevation increase, even with strict elevation weights.

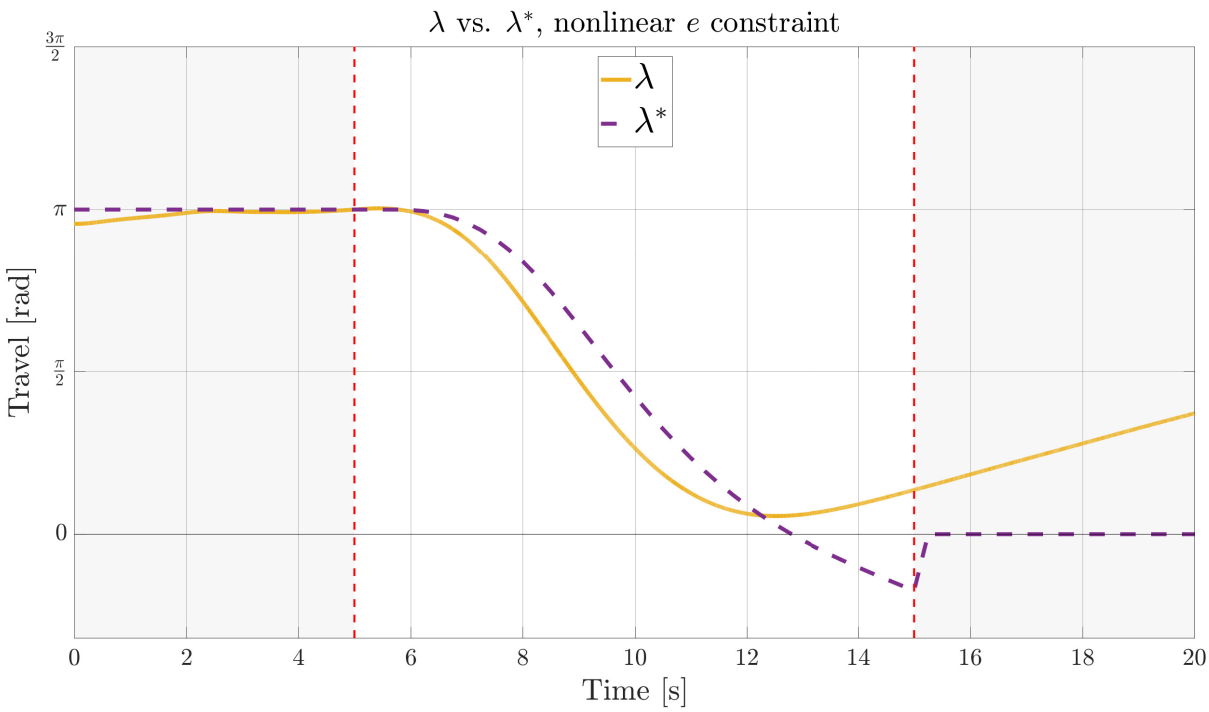

Figure 11: Optimal travel (SQP) vs measured travel angle, without feedback.

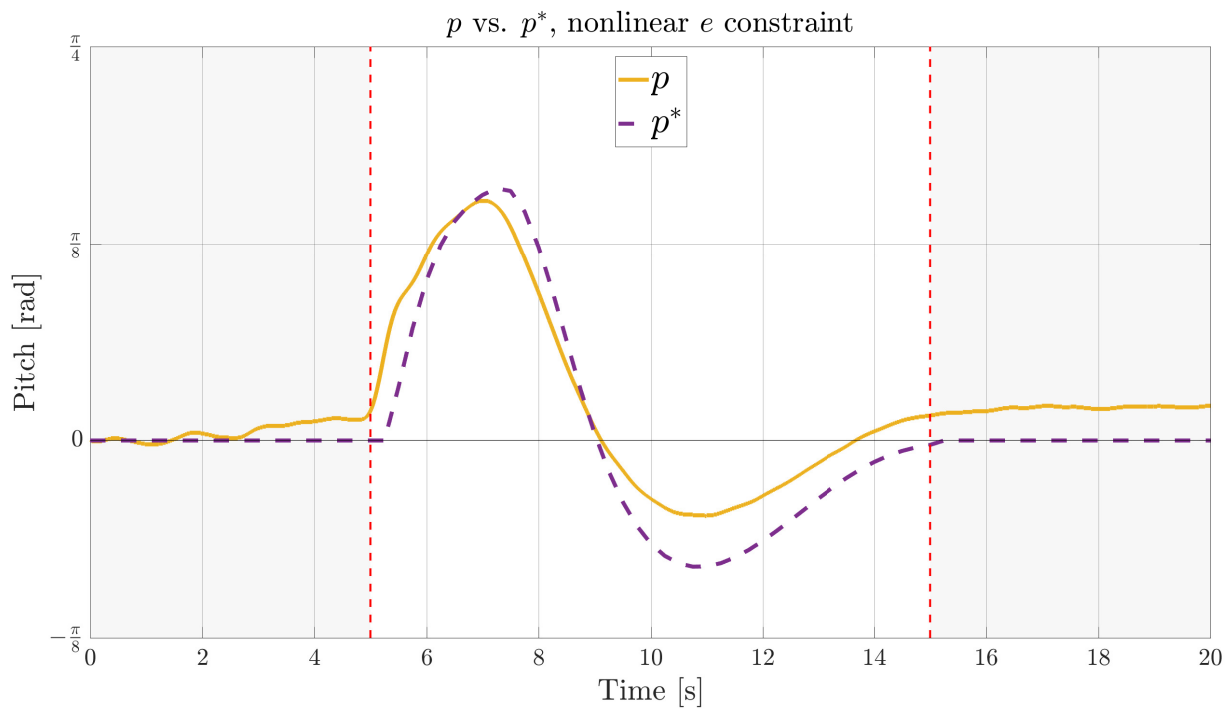

Figure 12: Optimal pitch (SQP) vs the measured pitch angle, without feedback.

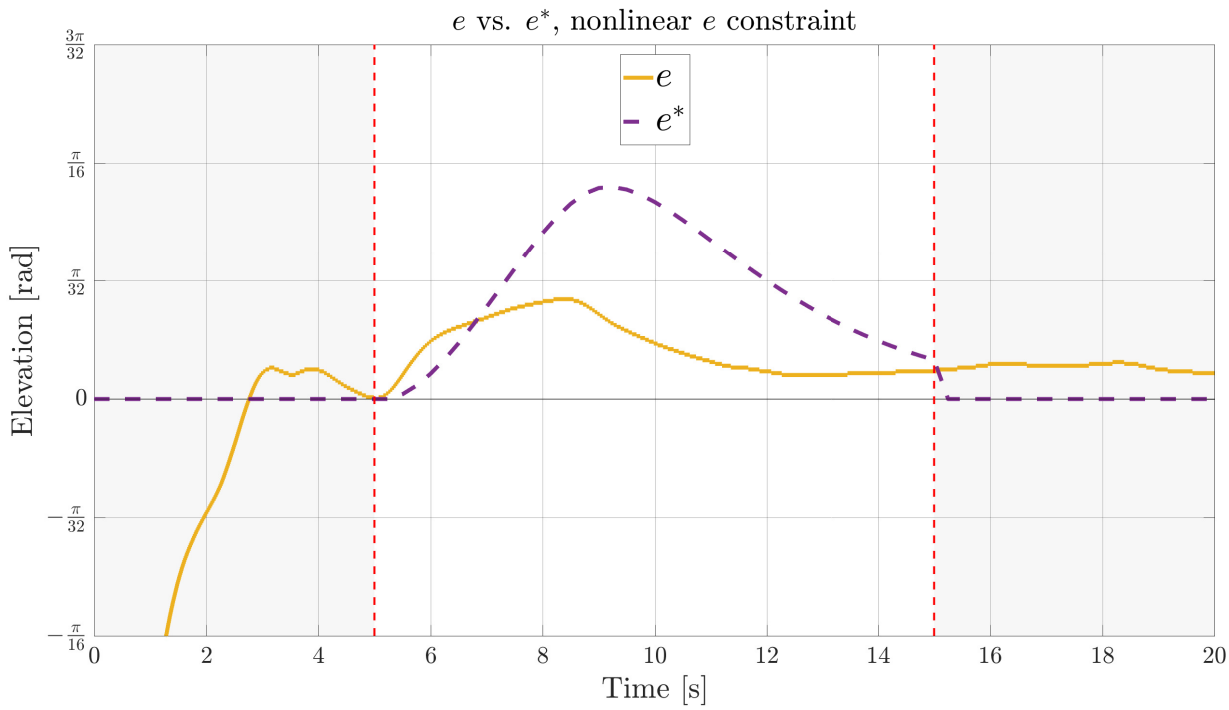

Figure 13: Optimal elevation (SQP) vs the measured elevation angle, without feedback.

5.4.2 Closed Loop

The following weights were placed on the closed loop system:

$$Q = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}, \quad R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

The resulting **K**-matrix became:

$$K = \begin{bmatrix} -3.6396 & -7.9729 & 3.32268 & 0.9306 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12.1113 & 10.263 \end{bmatrix}$$

The weights were chosen to penalize deviations in elevation and travel, while also keeping the helicopter from rising too quickly. This is a balancing act, since increasing travel accuracy (which is controlled by pitch) means removing some the downwards thrust the system thinks the elevation input can provide. The weighting were chosen larger for elevation than travel, as clearing the elevation constraint was prioritized over reaching x_f precisely. The pitch weighting was comparatively relaxed to allow for a more smooth travel trajectory and better elevation tracking. Figures 15 - 17 shows the result of implementing a feedback controller into our nonlinear problem.

Again we see that the open loop drift-off, as well as the spike in travel towards the end, is eliminated and we reach the reference without significant steady state error. After implementing the feedback controller we also experienced a big improvement regarding the elevation, as we are now much closer to the optimal elevation trajectory shown in fig. 15. In the original exercise, we were meant to solve the problem for $N = 40$ (10s), but we found this to be insufficient. The helicopter didn't reach λ_f fast enough, and zero-padding meant that a sudden burst in pitch rate, causing bad performance near the end. Therefore the horizon was increased to 50 (12.5s). This didn't increase the computational time significantly, and solved the problem of adjusting suddenly near λ_f. Our solution without feedback was implemented and run with a horizon of 40 since increasing the horizon without feedback didn't improve the performance significantly.

In section 4.1 the LQR-controller was implemented in a cascade in `SIMULINK`. With the system expanded, a simpler implementation was chosen, as can be seen in fig. 14. A view of the full `SIMULINK`-system with feedback can be found in appendix B.4.

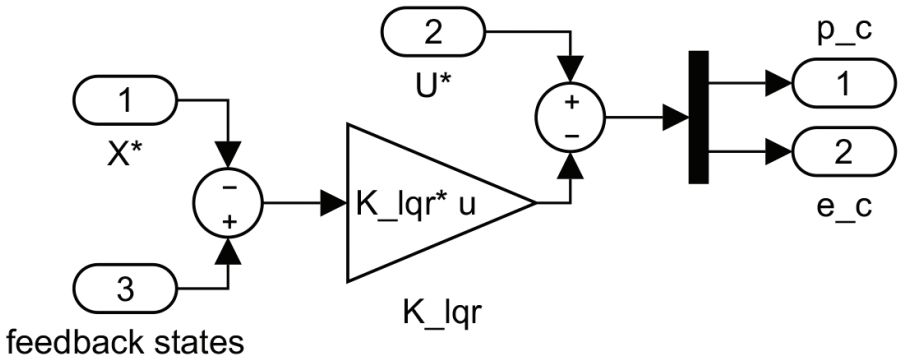

Figure 14: LQR `SIMULINK` implementation for full system

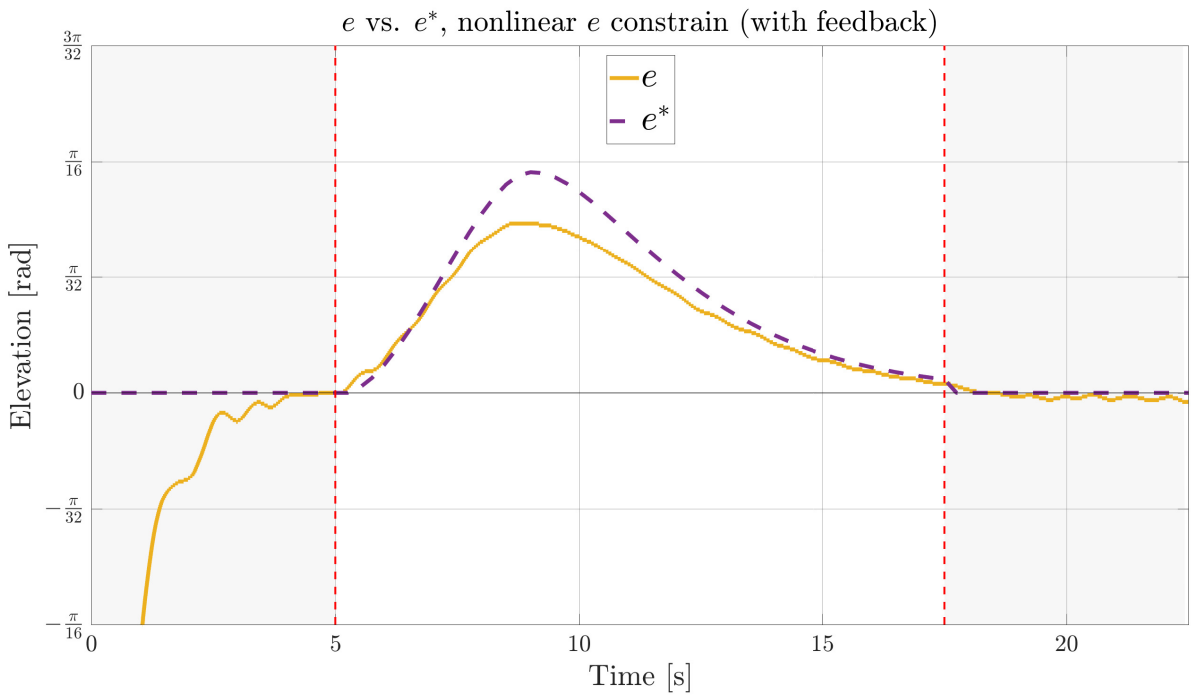

Figure 15: Optimal elevation (SQP) vs the measured, with feedback.

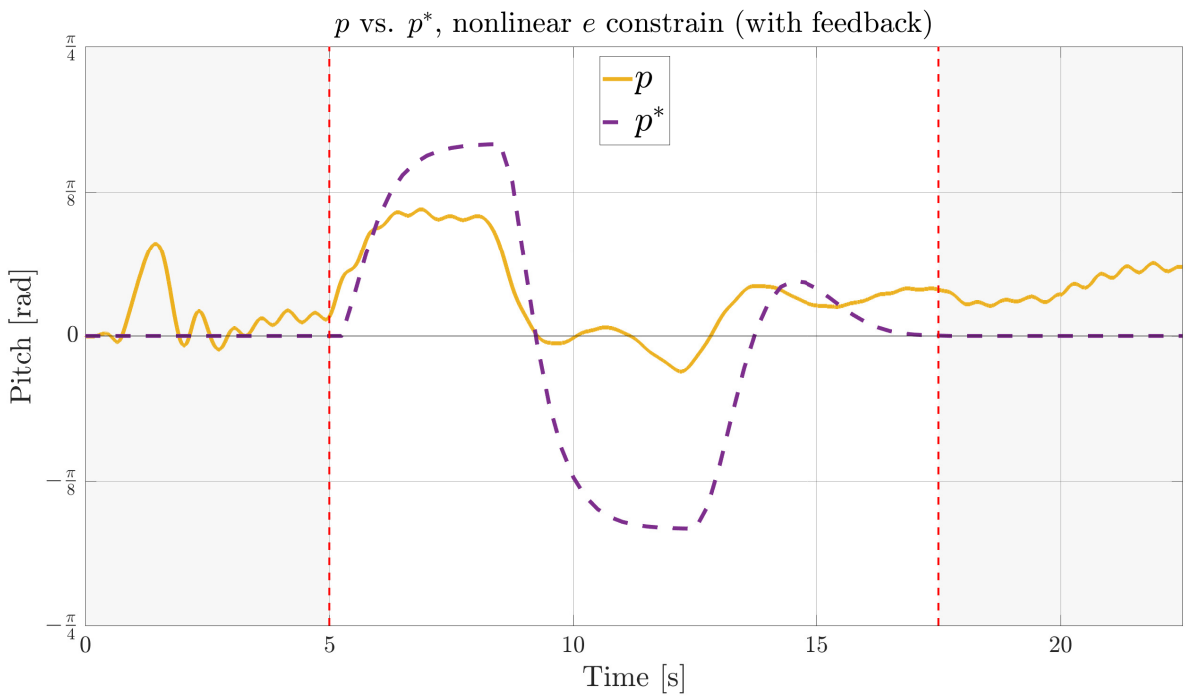

Figure 16: Optimal pitch (SQP) vs the measured, with feedback.

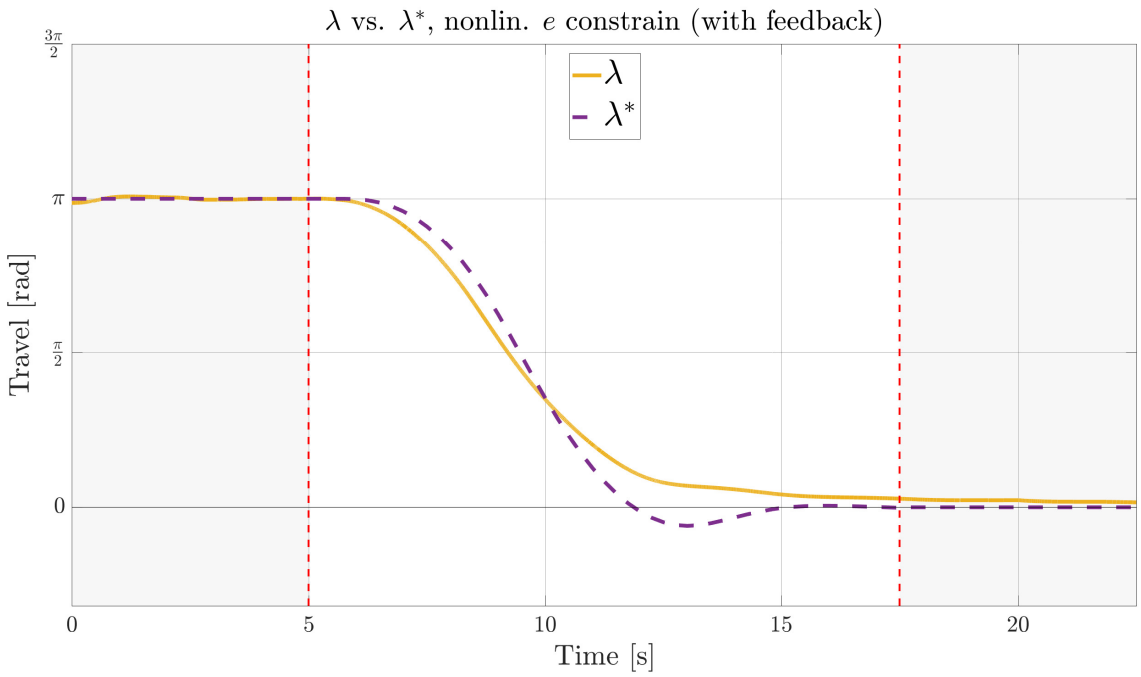

Figure 17: Optimal travel (SQP) vs the measured, with feedback.

5.5 State decoupling

In the model, pitch and elevation are not coupled. In reality, however, the elevation is coupled with the pitch: the plant gets more downward thrust from the same amount of force if the helicopter is horizontal compared to tilted. This is easy to verify if you hold the helicopter completely tilted; there won't be any upwards thrust. This means that the plant will experience less thrust if the pitch is high, compared to when the helicopter is turning. However, this is not reflected in our simplified model. This is evident when comparing the optimal trajectories in fig. 15 and fig. 16, where the pitch is at its maximum when elevation is increased the most. If the model was correct that would be the case, but it is simply not possible to achieve both optimal paths with this model, since the states are coupled in reality. Since the elevation constraint then is placed in the middle of the turn, the actual elevation e will necessarily be lower than e^*. So there must be a trade-off, which is why we chose to prioritise elevation in the controller tuning. The decoupling is something to consider when it comes to improving the model, although including the link between e and p would result in a nonlinear system, which is more computationally cumbersome.

5.6 Alternative Constraints

Some of the unwanted effects caused by model simplifications discussed in section 5.5 was attempted minimized by imposing stricter pitch constraints. Lowering the maximum allowed pitch from $\frac{30\pi}{180}$ to $\frac{20\pi}{180}$ caused the helicopter to achieve higher maximum altitude (the horizon, N, was increased to 50 in this exercise as well), although it still failed to reach the optimal trajectory completely. This is again due to the fact that the pitch and elevation are decoupled in the model used. A comparison between the open loop, closed loop and alternative closed loop responses as well as the optimal trajectory and the actual elevation constraint, is shown in shown in fig. 18. Here we clearly see an increase in the elevation from a stricter pitch constraint, which makes sense considering the model.

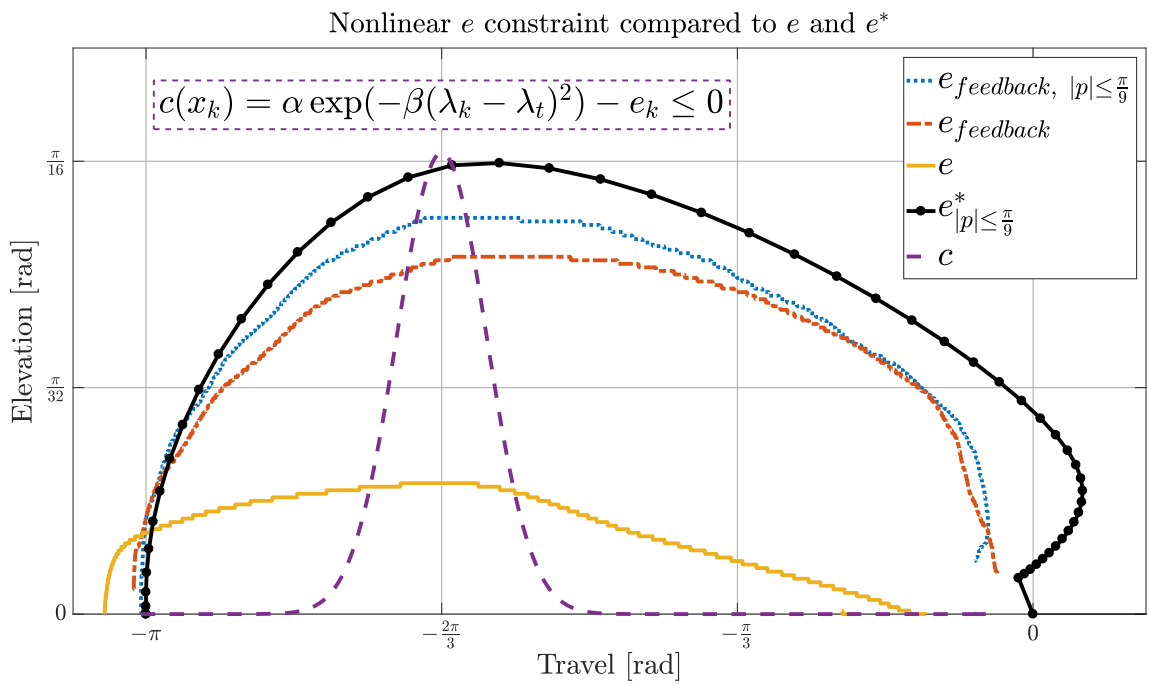

Figure 18: Optimal trajectory and the trajectories measured for the open loop, closed loop and alternative closed loop case plotted against the actual elevation constraint.

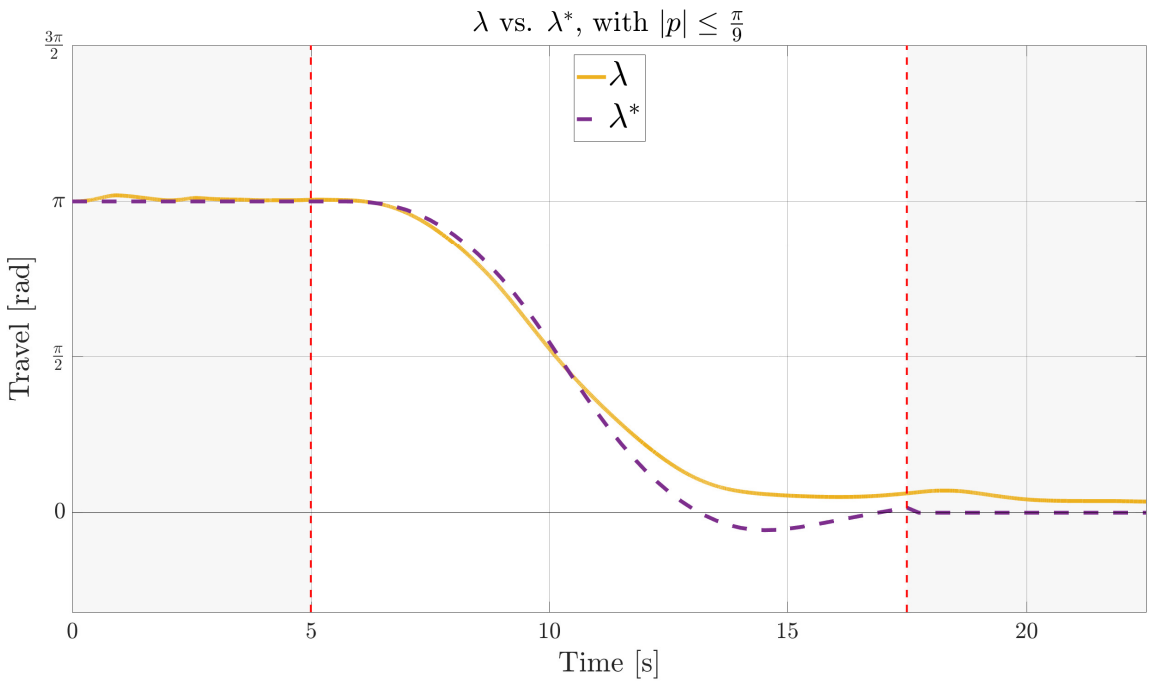

Figure 19: Optimal travel (SQP) vs the measured travel angle, with feedback and alternative constraints.

6 Conclusion/Discussion

Throughout the lab assignment, a few different control methods, based on optimal control and more classical control theory, have been tested and compared. Starting off, we had the system controlled by an optimal input calculated by `quadprog` in `MATLAB`, without any form of feedback. This caused large deviations from the optimal trajectory, due to open loop control and modeling simplifications, leading to trade-offs in control tuning; the optimal solution wants performance not achievable due to unrealistic decoupling between pitch and elevation. However with feedback introduced through a tuned LQ controller, the system performed a lot better while still being relatively time and computationally inexpensive.

Then a non-linear constraint was added to the elevation angle, effectively forcing the system out of its linearised working area. Without feedback it once again failed to stick to its optimal path, but with feedback managed to stay fairly close. That is, taking modelling errors into account. All in all, the results in fig. 8 and fig. 17 show that the use of feedback is essential, one relatively easy way of accomplishing this was to include a linear quadratic controller.

One model shortcoming, discussed in section 5.5, was attempted minimized by imposing stricter constraints on the pitch angle to try to increase the elevation angle. This gave the overall system a slightly more accurate response regarding elevation, as seen in fig. 18.

References

[1] Departement of Cybernetics Engineering. *TTK4135 Optimization and Control Helicopter Lab*. 2017.

[2] Gravdahl Jan T. Egeland Olav. *Modelling and simulation for automatic control*. Marine Cybernetics, 2003.

[3] Freedman Roger Young D. Hugh. *University Physics with Modern Physics*. Pearson Education, 2014.

A Appendix A - MATLAB code

A.1 MATLAB - Initialisation

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

% Updated spring 2017, Andreas L. Flaaten

clear all;
clc;
close all

%% 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 = 6.6; % Identified equilibrium voltage sum.
V_d_eq = 0.4; % 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;

%% Pitch closed loop syntesis
% Controller parameters
w_p = 1.8; % Pitch controller bandwidth.
d_p = 1.0; % Pitch controller rel. damping.
K_pp = w_p^2/K_1;
K_pd = 2*d_p*sqrt(K_pp/K_1);
Vd_ff = V_d_eq;

% Closed loop transfer functions
Vd_max = 10 - V_s_eq; % Maximum voltage difference
deg2rad = @(x) x*pi/180;
Rp_max = deg2rad(15); % Maximum reference step
s = tf('s');
G_p = K_1/(s^2);
C_p = K_pp + K_pd*s/(1+0.1*w_p*s);
L_p = G_p*C_p;
S_p = (1 + L_p)^(-1);

plot_pitch_response = 0;
```

```
if plot_pitch_response
    figure()
    step(S_p*Rp_max); hold on;
    step(C_p*S_p*Rp_max/Vd_max);
    legend('norm error', 'norm input')
    title('Pitch closed loop response')
end

%% Elevation closed loop analysis
% Controller parameters
w_e = 0.5; % Elevation controller bandwidth.
d_e = 1.0; % Elevation controller rel. damping.
K_ep = w_e^2/K_3;
K_ed = 2*d_e*sqrt(K_ep/K_3);
K_ei = K_ep*0.1;
Vs_ff = V_s_eq;

% Closed loop transfer functions
Vs_max = 10 - V_s_eq; % Maximum voltage sum
Re_max = deg2rad(10); % Maximum elevation step
G_e = K_3/(s^2);
C_e = K_ep + K_ed*s/(1+0.1*w_e*s) + K_ei/s;
L_e = G_e*C_e;
S_e = (1 + L_e)^(-1);

plot_elev_response = 0;
if plot_elev_response
    figure()
    step(S_e*Re_max);
    hold on;
    step(C_e*S_e*Re_max/Vs_max);
    legend('norm error', 'norm input')
    title('Elevation closed loop response')
end
```

A.2 MATLAB - Section 3

```
%% *------------------- PART 10.2 ----------------------------*
run('init.m')   % closes all figures, clears all and clc + init parameters
addpath('Help functions') % gena2, genbegr2 and other handed-out functions

%% [----- 10.2.1 - Continuous time state space -----]
A_c = [ 0       1       0       0     ;       % travel
        0       0     -K_2      0     ;       % travel dot
        0       0       0       1     ;       % pitch
        0       0 -K_1*K_pp -K_1*K_pd ;];     % pitch dot

B_c = [ 0
        0
        0
     K_1*K_pp  ];

%% [----- 10.2.2 - Forward Euler Method -----]
I       = eye(4);
```

```
delta_t = 0.25;
A       = I + delta_t*A_c;
B       = delta_t*B_c;

%% [——— 10.2.3 – Optimal trajectory ———]
% Inital values
lambda_0 = pi;      x0 = [lambda_0 ;0 ;0 ;0];
lambda_f = 0;       xf = [lambda_f ;0 ;0 ;0];
q0       = 0.1;     Q1 = 2*diag([1 0 0 0]);     % q0 = 0.1 OR 1 OR 10
time_pad = 5;       A1 = A;
sim_t    = 25;      B1 = B;

% 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)
N  = 100;                       % Time horizon for states
M  = N;                         % Time horizon for inputs
n  = N*mx+M*mu;

% Generating A_eq, B_eq and Q
A_eq       = gena2(A1, B1, N, mx, mu);
B_eq       = zeros(size(A_eq,1),1);
B_eq(1:mx) = A1*x0;
Q          = 2*genq2(Q1,q0,N,M,mu);

% Initialize z
c  = zeros(n,1);
z  = zeros(n,1);
z0 = z;

% Bounds
pk    = 30*pi/180;                    % Absolute boundary on pitch
ul    = -pk;                          % Lower bound on control -- u1
uu    = pk;                           % Upper bound on control -- u1

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] = genbegr2(N,M,xl,xu,ul,uu);
vlb(n)    = 0;                        % We want the last input to be zero
vub(n)    = 0;                        % We want the last input to be zero

% Using QP to find Z and lambda
tic
[z, lambda] = quadprog(Q,c,[],[],A_eq,B_eq,vlb,vub,x0);
toc

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

padded_zeros = time_pad/delta_t;
zero_padding = zeros(padded_zeros,1);
unit_padding = ones(padded_zeros,1);

t  = 0:delta_t:delta_t*(length(u)-1);
u  = [zero_padding; u; zero_padding];
x1 = [lambda_0*unit_padding; x1; zero_padding];
x2 = [zero_padding; x2; zero_padding];
x3 = [zero_padding; x3; zero_padding];
x4 = [zero_padding; x4; zero_padding];

% Input imported to helicopter
calculated_input.time               = t;
calculated_input.signals.values     = u;
calculated_input.signals.dimensions = 1;
```

A.3 MATLAB - Section 4

```
%% ------------------------- PART 10.3 -------------------*
run('Part_10_2.m')
addpath('Data logs')

%% [----- Task 10.3.1 Q,R and K (LQR) -----]
Q_lqr = diag([3 1 .05 .1]);
R_lqr = diag([.1]);
K_lqr = dlqr(A,B,Q_lqr,R_lqr);
```

A.4 MATLAB - Section 5

```
%% ------------------------- PART 10.4 -------------------*
run('init.m')   % closes all figures, clears all and clc + init parameters
addpath('Help functions') % gena2, genbegr2 and other handed-out functions
global N alpha lambda_t beta nx

%% [----- Task 10.4.1 System on continous state space -----]
A_c = [ 0     1     0       0        0       0      ;      % travel
        0     0   -K_2      0        0       0      ;      % travel dot
        0     0     0       1        0       0      ;      % pitch
        0     0 -K_1*K_pp -K_1*K_pd  0       0      ;      % pitch dot
        0     0     0       0        0       1      ;      % elevation
        0     0     0       0   -K_3*K_ep -K_3*K_ed ;];    % elevation dot

B_c = [ 0       0      ;
        0       0      ;
        0       0      ;
     K_1*K_pp   0      ;
        0       0      ;
        0    K_3*K_ep  ;];

%% [----- Task 10.4.2 Forward Euler Method -----]
I       = eye(6);
```

```
delta_t = 0.25;
A       = I + delta_t*A_c;
B       = delta_t*B_c;

%% [——— Task 10.4.3 SQP ———]
q1           = 0.1;    q2       = q1;
alpha        = 0.2;    beta     = 20;
lambda_0     = pi;     lambda_f = 0;        lambda_t = 2*pi/3;
time_padding = 5;      sim_t    = 22.5;

% Number of states and inputs
nx = size(A,2);                % Number of states (number of columns in A)
nu = size(B,2);                % Number of inputs(number of columns in B)
N  = 50;                       % Time horizon for states
M  = N;                        % Time horizon for inputs
n  = N*nx+M*nu;

x0     = [lambda_0 ;0 ;0 ;0; 0; 0];
xf     = [lambda_f; 0; 0; 0; 0; 0];
x      = zeros(N*6,1);
x(1:6) = x0;
Q_1    = 2*diag([1 0 0 0 0 0]);
P_1    = diag([q1 q2]);

% Generating A_eq, B_eq and Q
A_eq       = gena2(A, B, N, nx, nu);
B_eq       = zeros(size(A_eq,1),1);
B_eq(1:nx) = A*x0;
G          = 2*genq2(Q_1,P_1,N,M,nu);

% Initialize z
z  = zeros(n,1);
z0 = z;

% Bounds
pk         = 30*pi/180;                % Absolute boundary on pitch
ul         = [-pk; -inf];              % Lower bound on control —— u1
uu         = [pk; inf];                % Upper bound on control —— u1

xl(1:nx,1) = -Inf*ones(nx,1);          % Lower bound on states (no bound)
xu(1:nx,1) = Inf*ones(nx,1);           % Upper bound on states (no bound)
xl(3)      = ul(1);                    % Lower bound on state x3
xu(3)      = uu(1);                    % Upper bound on state x3

% Generate constraints on measurements and inputs
[vlb,vub]  = genbegr2(N,M,xl,xu,ul,uu);
vlb(n)     = 0;                        % We want the last input to be zero
vub(n)     = 0;                        % We want the last input to be zero

% Find Z and lambda by using SQP-based fmincon
f   = @(z) 1/2*z'*G*z;
opt = optimoptions('fmincon','Algorithm','sqp','MaxFunEvals',400000);

tic
[Z, ZVAL, EXITFLAG] = ...
```

```
78      fmincon(f, z0, [], [], A_eq, B_eq, vlb, vub, @constraints, opt);
79  toc
80
81  % Extract control inputs and states
82  u1 = [Z(N*nx+1:nu:n);Z(n-1)];           % Control input 1 from solution
83  u2 = [Z(N*nx+2:nu:n);Z(n)];             % Control input 2 from solution
84  x1 = [x0(1);Z(1:nx:N*nx)];              % State x1 from solution
85  x2 = [x0(2);Z(2:nx:N*nx)];              % State x2 from solution
86  x3 = [x0(3);Z(3:nx:N*nx)];              % State x3 from solution
87  x4 = [x0(4);Z(4:nx:N*nx)];              % State x4 from solution
88  x5 = [x0(5);Z(5:nx:N*nx)];              % State x5 from solution
89  x6 = [x0(6);Z(6:nx:N*nx)];              % State x6 from solution
90
91  num_pads     = time_padding/delta_t;
92  zero_padding = zeros(num_pads,1);
93  unit_padding = ones(num_pads,1);
94
95  u1 = [zero_padding; u1; zero_padding];
96  u2 = [zero_padding; u2; zero_padding];
97  x1 = [lambda_0*unit_padding; x1; zero_padding];
98  x2 = [zero_padding; x2; zero_padding];
99  x3 = [zero_padding; x3; zero_padding];
100 x4 = [zero_padding; x4; zero_padding];
101 x5 = [zero_padding; x5; zero_padding];
102 x6 = [zero_padding; x6; zero_padding];
103
104 t     = 0:delta_t:delta_t*(length(u1)-1);
105 x_opt = [x1 x2 x3 x4 x5 x6];
106 u     = [u1 u2];
107
108 % Input imported to helicopter
109 calculated_input.time               = t;
110 calculated_input.signals.values     = u;
111 calculated_input.signals.dimensions = 2;
112
113 %% [——— Task 10.4.4 - With feedack ———]
114 Q_lqr = diag([20 1 .5 1 40 10]);
115 R_lqr = diag([.1 .1]);
116 K_lqr = dlqr(A,B,Q_lqr,R_lqr);
117
118 %% [——— Task 10.4.6 - With optional constraint ———]
119 % To run with optional constraints, change the boundaries above, before
120 % solving the problem with fmincon. You can also change the tuning in the
121 % section right above, 10.4.4 - With feedback.
122 % In our application we changed the absolute boundary of pitch from
123 % 30*pi/180 to 20*pi/180 (pk = 20*pi/180, line 58)
```

B Appendix B - SIMULINK models

B.1 SIMULINK - Section 3

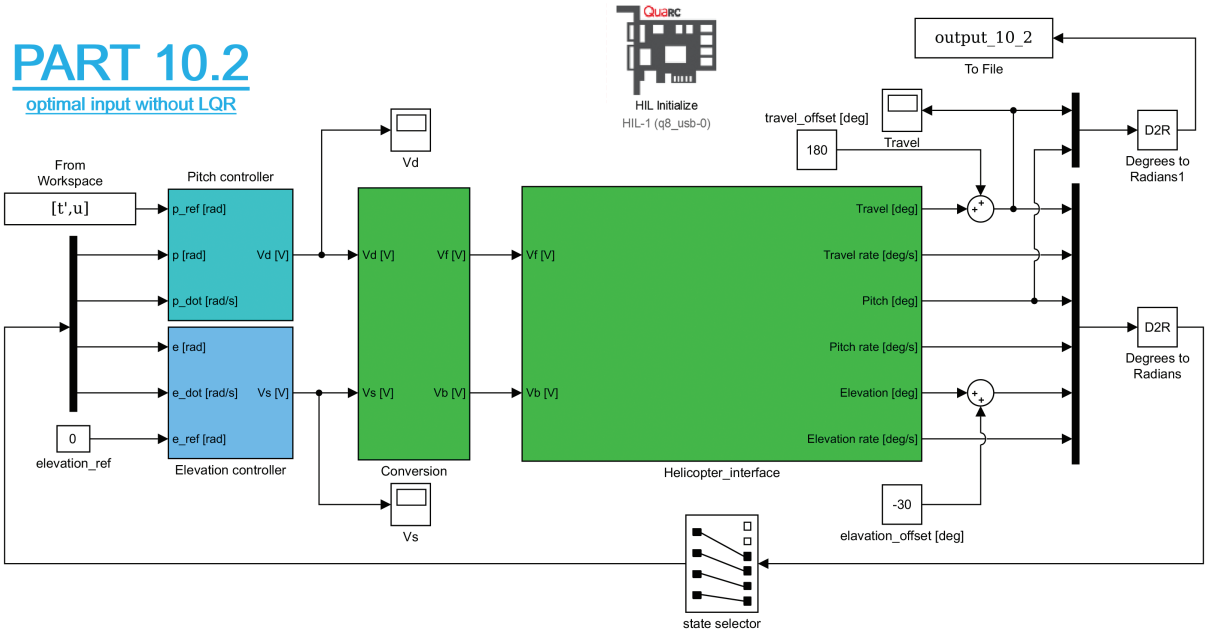

Figure 20: Full SIMULINK model from part 1 (section 3)

B.2 SIMULINK - Section 4

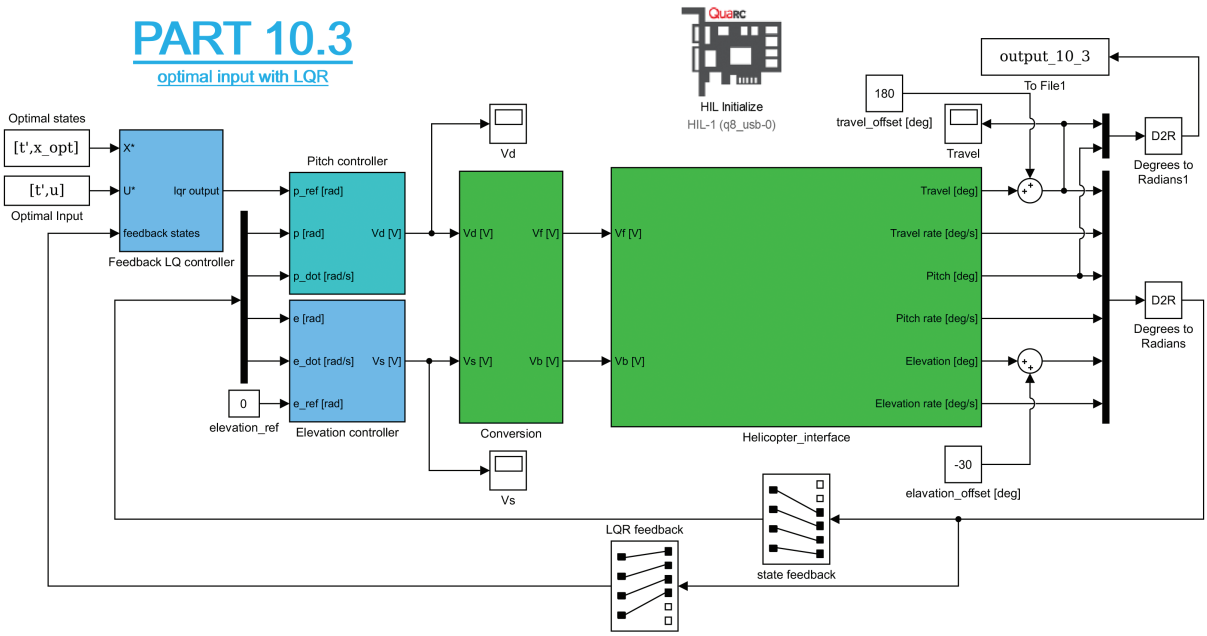

Figure 21: Full SIMULINK model from part 2 (section 4)

B.3 SIMULINK - Section 5.4.1

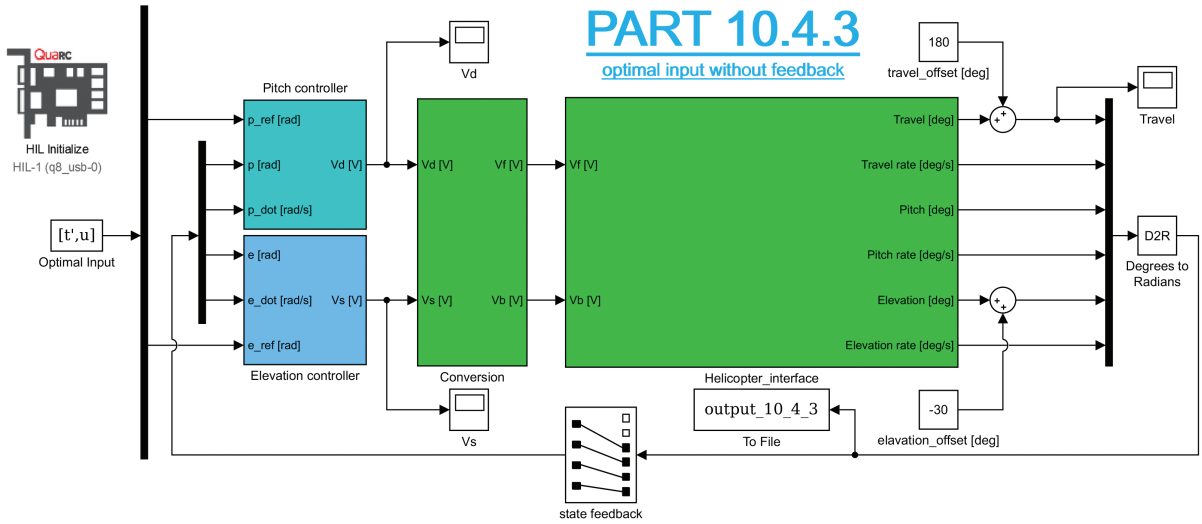

Figure 22: Full SIMULINK model from part 3 (section 5)

B.4 SIMULINK - Section 5.4.2

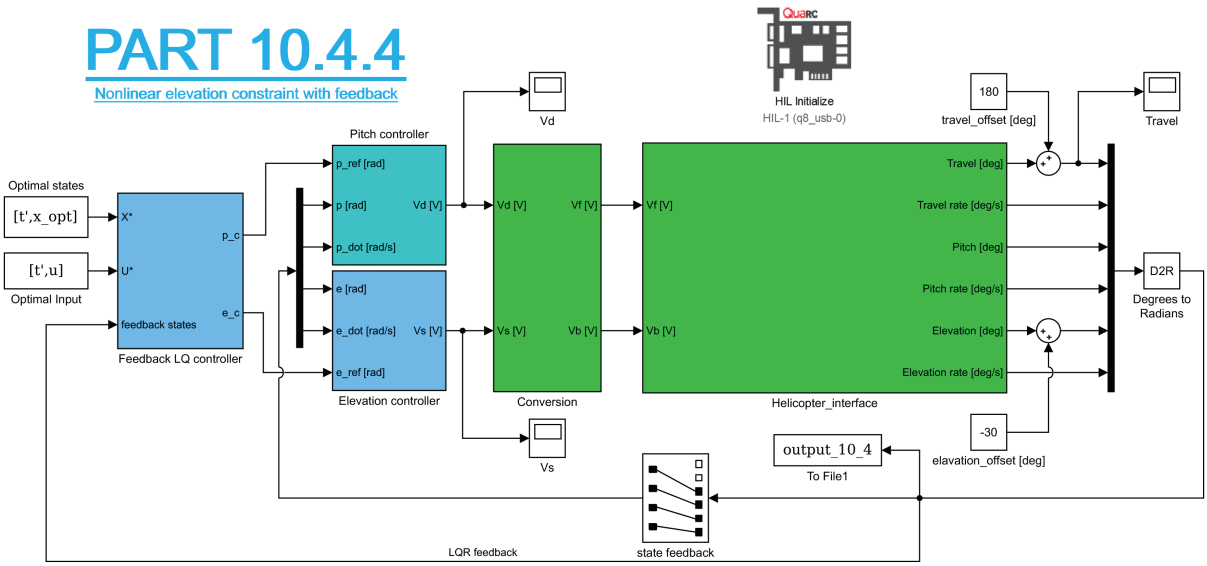

Figure 23: Full SIMULINK model from part 3 (section 5)