Helicopter Lab

Group 38
Student 730271
Student 478466
Student 478376

March 15, 2019

Department of Engineering Cybernetics

Abstract

This report is part of the NTNU course TTK4135 Optimization and Control. A helicopter plant was controlled using optimal control theory. An optimal input sequence that moves the helicopter 180 degrees with two degrees of freedom, with linear constraints, was calculated and implemented in open loop. Then feedback was introduced through a linear quadratic controller. And finally, an optimal trajectory for three degrees of freedom was calculated with and without feedback with additional nonlinear constraints. The results of the report demonstrate the use of optimal control and the advantages of feedback.

Contents

1 Introduction

This report tackles three tasks all relating to optimization and control as a part of NTNU's course TTK4135. More specifically finding optimal control sequences for a helicopter arm with three degrees of freedom; pitch, travel and elevation. The basic outlines of the problems are as follows:

- Optimal control of pitch/travel without feedback:

  Find an optimal trajectory and input sequence that moves the helicopter 180° with a linear constraint on the pitch.

- Optimal control of pitch/tavel with feedback:

  Introduce feedback to the previous controller by implementing a linear quadratic controller

- Optimal control of pitch/travel and elevation with and without feedback:

  Find an optimal trajectory and input sequence that moves the helicopter 180° with a linear constraint on pitch and a nonlinear constraint on elevation. With and without feedback.

Some of the problems also have theory related questions. In all the tasks the assumption is made that the pitch and elevation controllers are turned on and well tuned. Their references p_c and e_c will work as the variables of $\boldsymbol{u}(t)$.

The report is organized as follows: Section 2 contains a quick description of the lab setup. Section 3 contains the relevant theory and model derivations, split into different themes. Section 4 shows the results and the discussion of them, split into tasks. Section 5 contains a short conclusion and some closing remarks. Simulink models will be shown where they are relevant. Appendix A contains our MATLAB code. The Bibliography can be found at the end, on page 28.

2 Lab setup

The helicopter lab setup consists of a helicopter station, power supply, and Quanser Q4 cards, as well as hardware for signal conditioning and measuring. The helicopter consists of a vertical mounted base, on top of which a horizontal arm is attached. Two motors with propellers, the helicopter head, are attached to one side of the arm, a counterweight to the other.

The setup has three rotational joints and thus three degrees of freedom. The attached arm rotates around its axis, the helicopter moves up and down with respect to the joint between the base and the arm, and the helicopter head may tilt with respect to the arm. These will be referred to as travel, elevation and pitch respectively. The helicopter is actuated by the two DC motors connected to the propellers on the helicopter head.

3 Theory

3.1 Model description

The helicopter plant is modelled as three point masses. Two point masses represent the pair of propellers and motors on the helicopter head, the third point mass represent the counterweight. Consequently, the mass of the base and arms are considered to be zero. The model is depicted in Figure 1. The point masses are depicted as cubes, the rotational joints as cylinders, and the arms and base as straight lines connecting the masses and joints.

The angular displacement of the joints are defined as p, e, and λ for pitch, elevation and travel, respectively. The pitch is defined as zero when the helicopter head is balanced horizontally. Similarly, elevation is defined as zero when the arm is balanced horizontally. The travel is zero wherever the helicopter starts.

The inputs to the plant from the control layer are V_s and V_d representing the voltage sum and difference applied to the DC motors. The propeller forces F_f and F_b act perpendicular to the head and are assumed to have a proportional relation with the voltage from the DC motors $F_{f,b} = K_f V_{f,b}$. Where V_f and V_b are the inputs for the front and back motors respectively. The voltage sum and difference are then defined as $V_s = V_f + V_b$ and $V_d = V_f - V_b$.

3.2 Model derivation

The helicopter model is derived by applying the torque balances of the system. For a list parameters see appendix B. The torque balance of the elevation is given by

$$J_e \ddot{e} = l_a K_f V_s - T_g, \tag{1}$$

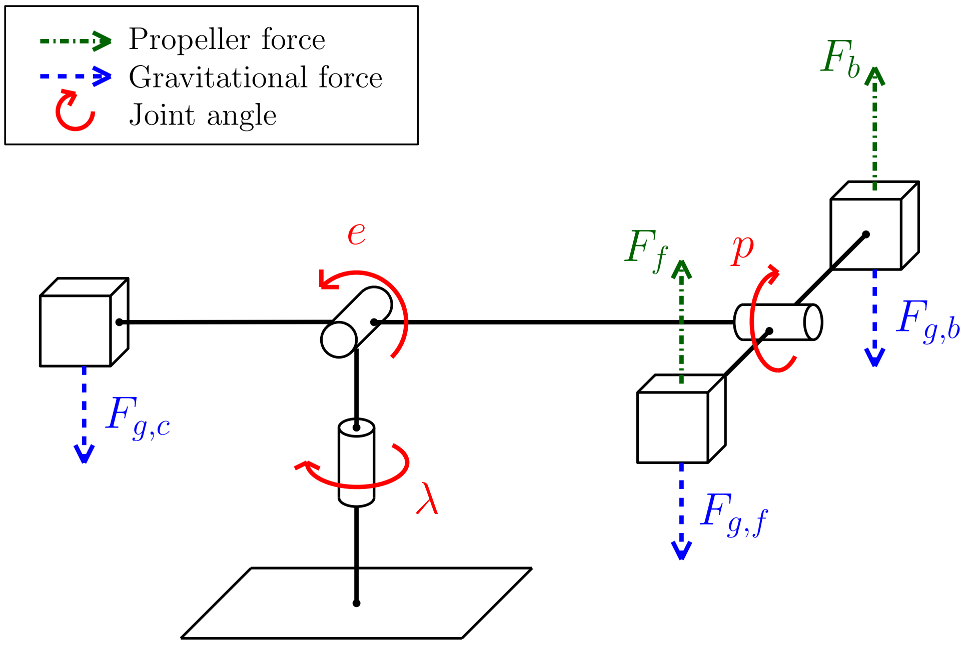

Figure 1: Helicopter model, with the states p, e and λ indicated [**assignment**].

which is controlled by the PD controller

$$V_s = K_{ep}(e_c - e) - K_{ed}\dot{e}, \quad K_{ep}, K_{ed} > 0. \tag{2}$$

Furthermore, by adding an integral term that cancels the constant term in eq. (1), the following closed loop elevation model is found:

$$\ddot{e} + K_3 K_{ed}\dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c, \quad K_3 = \frac{l_a K_f}{J_e}. \tag{3}$$

Similarly the pitch torque balance is given by

$$J_p \ddot{p} = l_h K_f V_d, \tag{4}$$

which is controlled by the PD controller

$$V_d = K_{pp}(p_c - p) - K_{pd}\dot{p}, \quad K_{pp}, K_{pd} > 0. \tag{5}$$

This results in the following model of the closed loop helicopter pitch:

$$\ddot{p} + K_1 K_{pd}\dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c, \quad K_1 = \frac{l_h K_f}{J_p}. \tag{6}$$

Finally the travel rate is given by

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

By assuming a small pitch angle and linearizing around 0 we get the following model for pitch:

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

Finally the closed loop helicopter system is expressed as:

$$\ddot{e} + K_3 K_{ed} \dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c, \tag{9a}$$

$$\ddot{p} + K_1 K_{pd} \dot{p} + K_1 K_{pp} e = K_1 K_{pp} p_c, \tag{9b}$$

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

$$\dot{r} = -K_2 p. \tag{9d}$$

Consider the continuous time state space system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}_c \boldsymbol{x} + \boldsymbol{B}_c \boldsymbol{u}. \tag{10}$$

By letting

$$\boldsymbol{x} = \begin{bmatrix} \lambda & r & p & \dot{p} & e & \dot{e} \end{bmatrix}^T, \quad \boldsymbol{u} = \begin{bmatrix} p_c & e_c \end{bmatrix}^T \tag{11}$$

we can express the system matrix as

$$\boldsymbol{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3 K_{ep} & -K_3 K_{ep} \end{bmatrix} \tag{12}$$

and the input matrix as

$$\boldsymbol{B}_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1 K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3 K_{ep} \end{bmatrix}. \tag{13}$$

The system is then discretized using the forward Euler method given by

$$\boldsymbol{x_{k+1}} = (\boldsymbol{I} - T\boldsymbol{A}_c)\boldsymbol{x}_k + T\boldsymbol{B}_c \boldsymbol{u}_k = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{14}$$

with the state transition matrix

$$\boldsymbol{A} = \begin{bmatrix} 1 & T & 0 & 0 & 0 & 0 \\ 0 & 1 & -TK_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 & 0 \\ 0 & 0 & -TK_1 K_{pp} & 1 - TK_1 K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & -TK_3 K_{ep} & 1 - TK_3 K_{ep} \end{bmatrix}, \tag{15}$$

and input matrix

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ TK_1K_{pp} & 0 \\ 0 & 0 \\ 0 & TK_3K_{ep} \end{bmatrix}. \tag{16}$$

Notice that the states e and $\dot{e}$ are decoupled from the rest of the states of the system. This allows us to simplify the state space system by disregarding the elevation dynamics and only use the model for pitch and travel dynamics. It follows that the simplified state space system has the states and input

$$\boldsymbol{x} = \begin{bmatrix} \lambda & r & p & \dot{p} \end{bmatrix}^T, \quad u = p_c \tag{17}$$

with state transition matrix and input matrix given by

$$\boldsymbol{A}_d = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & -TK_2 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & -TK_1K_{pp} & 1 - TK_1K_{pd} \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ TK_1K_{pp} \end{bmatrix}. \tag{18}$$

3.3 Dynamic optimization of helicopter system

The general constrained optimization problem can be formulated as

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & f(z) \\ \text{subject to} \quad & c_i(z) = 0, \ i \in \mathcal{E} \\ & c_i(z) \geq 0, \ i \in \mathcal{I} \end{aligned} \tag{19}$$

where $c_i(z)$ are the constraints. In dynamic optimization one finds the optimal trajectory given a state space system over some finite prediction horizon in time. The objective function for a discrete time dynamic optimization problem can then be stated as

$$f(x_1, ..., x_N, u_0, ..., u_{N-1}) = \sum_{t=0}^{N-1} f_t(x_{t+1}, u_t) \tag{20}$$

In optimal control the most important dynamic optimization problem is the QP, which has a quadratic objective function and linear constraints:

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & f(z) = \sum_{t=0}^{N-1} \frac{1}{2}\boldsymbol{x}_{t+1}^T Q \boldsymbol{x}_{t+1} + \frac{1}{2}\boldsymbol{u}_t^T \boldsymbol{R} \boldsymbol{u}_t \\ \text{subject to} \quad & \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t, t = 0, ..., N-1 \\ & \boldsymbol{x}_l \leq \boldsymbol{x}_t \leq \boldsymbol{x}_h, t = 1, ..., N \\ & \boldsymbol{u}_l \leq \boldsymbol{u}_t \leq \boldsymbol{u}_h, t = 0, ..., N-1 \end{aligned} \tag{21}$$

For the helicopter system the problem is to calculate the optimal trajectory from $\lambda_0 = \pi$ to $\lambda_f = 0$, with different constraints. The optimization algorithm will calculate a complete input sequence of the manipulated variable $\boldsymbol{u}$ based on the model, the objective function and the constraints. The objective function for the simplified state space model is given by:

$$\Phi = \sum_{i=1}^{N-1} (\lambda_i - \lambda_f)^2 + q p_{ci}^2, \quad q \geq 0 \tag{22}$$

which when written on the general QP form in eq. (21) results in the matrices:

$$\boldsymbol{Q} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad R = 2q. \tag{23}$$

And considering the constraint on pitch:

$$|p_k| \leq \frac{30\pi}{180}, \quad k \in \{1, ..., N\}, \tag{24}$$

the trajectory can be found by solving:

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & f(z) = \frac{1}{2} \boldsymbol{z}^T \boldsymbol{G} \boldsymbol{z} \\ \text{subject to} \quad & \boldsymbol{A}_{eq} \boldsymbol{z} = \boldsymbol{b}_{eq} \\ & \boldsymbol{z}_l \leq \boldsymbol{z} \leq \boldsymbol{z}_h \end{aligned} \tag{25}$$

where $\boldsymbol{z} = [x_1^T, ..., x_n^T, u_0^T, ..., u_{n-1}^T]^T$ and

$$\boldsymbol{A}_{eq} = \begin{bmatrix} \boldsymbol{I} & 0 & \cdots & \cdots & 0 & -\boldsymbol{B} & 0 & \cdots & \cdots & 0 \\ -\boldsymbol{A} & \boldsymbol{I} & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\boldsymbol{A} & \boldsymbol{I} & 0 & \cdots & \cdots & 0 & -\boldsymbol{B} \end{bmatrix}, \tag{26}$$

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{Q} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \boldsymbol{Q} & \ddots & & \vdots \\ \vdots & & \ddots & \boldsymbol{R} & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & \boldsymbol{R} \end{bmatrix} \boldsymbol{B}_{eq} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{x_0} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{27}$$

The objective function for the full state space model is given by:

$$\Phi = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2 \tag{28}$$

resulting in

$$\boldsymbol{Q} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{R} = \begin{bmatrix} 2q_1 & 0 \\ 0 & 2q_2 \end{bmatrix}. \tag{29}$$

For this case a nonlinear constraint on the elevation will be added, meaning a QP algorithm can no longer solve it. The elevation constraint is given by

$$e_k \geq \alpha e^{(-\beta(\lambda_k - \lambda_t)^2)} \quad \forall k \in \{1, ..., N\}, \tag{30}$$

which can be restated as a general constraint on the form in eq. (19) as

$$c(x_k) = \alpha e^{(-\beta(\lambda_k - \lambda_t)^2)} - e_k \leq 0 \quad \forall k \in \{1, ..., N\}. \tag{31}$$

3.4 Linear quadratic control

With the precalculated solution any plant deviations from the predicted path will not be corrected. One could therefore introduce state feedback to make the control system more robust. This can be done by implementing a LQ controller with the manipulated variable

$$\boldsymbol{u}_k = \boldsymbol{u}_k^* - \boldsymbol{K}^T (\boldsymbol{x}_k - \boldsymbol{x}_k^*). \tag{32}$$

The new input $\boldsymbol{u}_k$ now has an added correction term proportional to the deviation from the optimal trajectory x_k^*. The LQR finds the optimal gain matrix K by minimizing the cost function:

$$J = \sum_{i=0}^{\infty} \Delta x_{i+1}^T \boldsymbol{Q} \Delta x_{i+1} + \Delta u_i^T \boldsymbol{R} \Delta u_i, \quad \boldsymbol{Q} \geq 0, \;\; \boldsymbol{R} \geq 0 \tag{33}$$

for the linear model

$$\Delta \boldsymbol{x}_{i+1} = \boldsymbol{A} \Delta \boldsymbol{x}_i + \boldsymbol{B} \Delta \boldsymbol{u}_i, \tag{34}$$

where $\Delta \boldsymbol{x}$ and $\Delta \boldsymbol{u}$ are deviations from the optimal solution. For simplicity let $\boldsymbol{Q}$ and $\boldsymbol{R}$ be diagonal matrices, to easily penalize the different states and inputs independently.

It can be shown that the solution to the infinite horizon LQ control problem is given by

$$\boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P})^{-1}\boldsymbol{A}, \tag{35a}$$

$$\boldsymbol{P} = \boldsymbol{Q} + \boldsymbol{A}^T\boldsymbol{P}(\boldsymbol{I} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P})^{-1}\boldsymbol{A}, \tag{35b}$$

$$\boldsymbol{P} = \boldsymbol{P}^T \geq 0, \tag{35c}$$

where eq. (35b) is the discrete time algebraic Ricatti equation.

Another solution to handling devations from the planned trajectory would be to implement a MPC controller. Model Predictive Control continuously reevaluates the dynamic optimization problem, using the current state measurement as $\boldsymbol{x}_0$. It follows that the main difference between LQR and MPC is that the MPC solves the optimization problem at every time step with state feedback. Meaning that if the helicopter for some reason falls away from the optimal path it would find a new optimal path from the current state. This will however require more computation compared to the LQR which only solves the infinite horizon QP problem once. In theory a MPC controlled system would be more robust, as it always adapts to the new situation, while the LQR only corrects the input to stay on the precalculated trajectory. Figure 2 shows how the control structure with MPC would look.

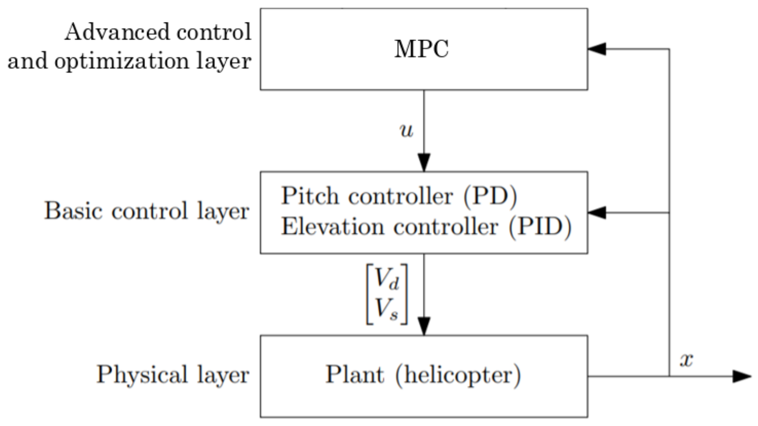
Advanced control and optimization layer
MPC
u
Basic control layer
Pitch controller (PD)
Elevation controller (PID)
$\begin{bmatrix} V_d \\ V_s \end{bmatrix}$
Physical layer
Plant (helicopter)
x

Figure 2: Control layers using MPC

4 Results and discussions

4.1 Optimal control of pitch and travel

4.1.1 Results

Using the values of table 1, the state transition matrix and input matrix were determined to be

$$\boldsymbol{A}_d = \begin{bmatrix} 1 & 0.250 & 0 & 0 \\ 0 & 1 & -0.142 & 0 \\ 0 & 0 & 1 & 0.250 \\ 0 & 0 & -0.810 & 0.100 \end{bmatrix}, \quad \boldsymbol{B}_d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.810 \end{bmatrix} \tag{36}$$

for the simplified system with elevation reference $e_c = 0$.

The optimization problem in eq. (25) was solved using the MATLAB function `problem2.m` found in appendix A.1. The optimal state trajectories for weights $q = 0.1$, $q = 1$ and $q = 10$ is plotted in fig. 3. The optimal input sequence that generate the corresponding trajectories is in fig. 4. Note that the helicopter is given 5 seconds to stabilize around the origin before the optimal trajectory is applied as a reference to the low level controllers. Furthermore travel starts at $\lambda_0 = \pi$, such that the objective function eq. (22) simplifies, since $\lambda_f = 0$

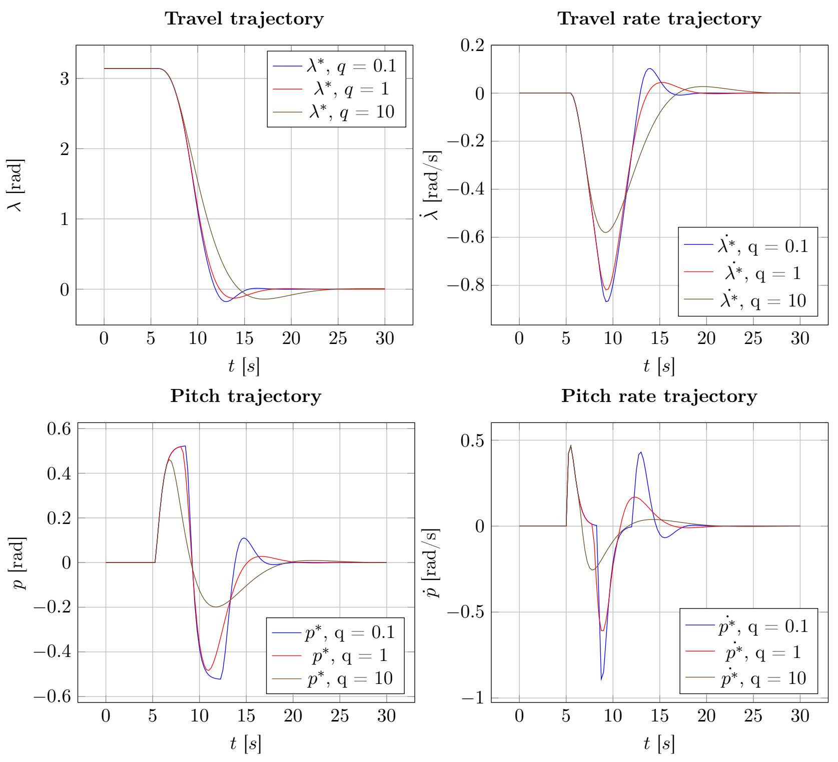

Figure 3: Calculated optimal trajectory for different weights q

The closed loop system was implemented in Simulink, and the LQR gain was calculated in the script appendix A.2. The Simulink block diagram is shown in **??**. The script utilizes the MATLAB function `dlqr` to calculate the gain matrix $\boldsymbol{K}$ given the weight matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. The following weights where found from tuning the system:

$$\boldsymbol{Q} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R = 1. \tag{37}$$

This resulted in the gain matrix

$$\boldsymbol{K} = \begin{bmatrix} -4.884 & -10.202 & 4.294 & 1.040 \end{bmatrix}. \tag{38}$$

The measured vs. calculated closed loop trajectories are found in fig. 6, and a comparison between the optimal input sequence fed into the LQR and the resulting input sequence from the LQR is in fig. 4

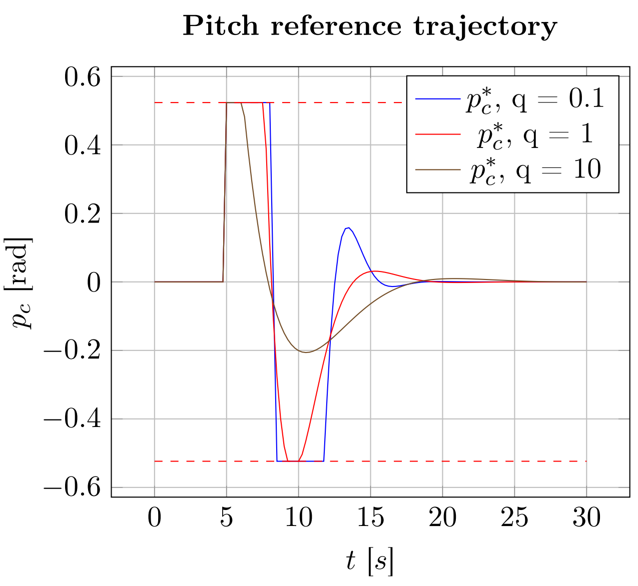

Figure 4: Optimal input sequence for different weights q

4.1.2 Discussion

When increasing the input weight q, we observe that the calculated optimal input sequence in fig. 4 gets slower and less aggressive. When the cost of the input for each time step in the objective function eq. (22) gets larger relative to the travel cost, the optimal trajectory will naturally be slower. This is seen in the state trajectories in fig. 3, particularly we observe that the helicopter will use more time to reach the origin when the input cost is increased.

When evaluating the potency of the formulated optimization problem, one thing to note is that the objective function only uses travel out of all the states. This makes the problem simple and easier to solve, but it is definitely worth considering adding more states to the objective function. For instance one might want to penalize large pitch and elevation, since the model is only accurate around the origin. For large pitch or elevation angles the nonlinearities of the system are more dominant and thus the calculated trajectory will be more inaccurate. Furthermore, it would also be possible to penalize the derivative states, in order to avoid fast changes. Adding more constraints on the states in the system would also help dealing with these problems, which is discussed more later. Another issue with the current objective function is that when zero travel is reached, the input will be zero. This is not necessarily wanted behaviour, as the helicopter will still have a certain travel rate and might pass the zero point and oscillate. Penalizing travel rate might help here.

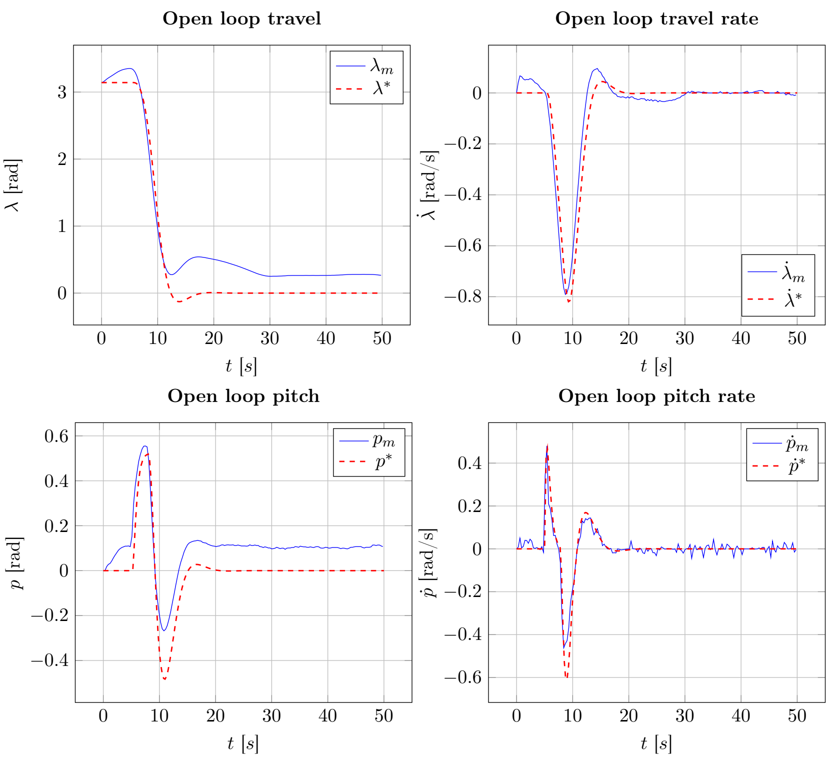

Figure 5: Measured and calculated open loop state trajectories.

We observe from the open loop response that the helicopter model is relatively accurate, and the helicopter almost manages to hit the travel target. But the predicted state trajectories are obviously not exactly correct. This is because the nonlinear dynamics of the helicopter are neglected, as well as factors such as air resistance, friction in all the joints and motors, neglection of motor dynamics, the ground effect and other nonlinearities in the system. There is a substantial initial drift in pitch, likely originating from the stabilization of the elevation in the first few seconds of flight. As a result, when the input sequence is applied the helicopter is actually ahead of the optimal trajectory, as seen in fig. 5. When the LQR is used to correct for such deviations from the calculated trajectory, we observe a more accurate tracking of the trajectory in fig. 6. But we still observe a offset in pitch and travel, although smaller. Introducing integral action by adding the integrals of the states as states in the LQR is a possible solution here. But an offset in pitch is likely to be expected, as the helicopters usually need some pitch in order to get a constant travel angle, because of discrepancies in the physical system.

An alternative to the LQR which has introduced in section 3.4 is a MPC controller. As discussed, the MPC controller continuously reevaluates what

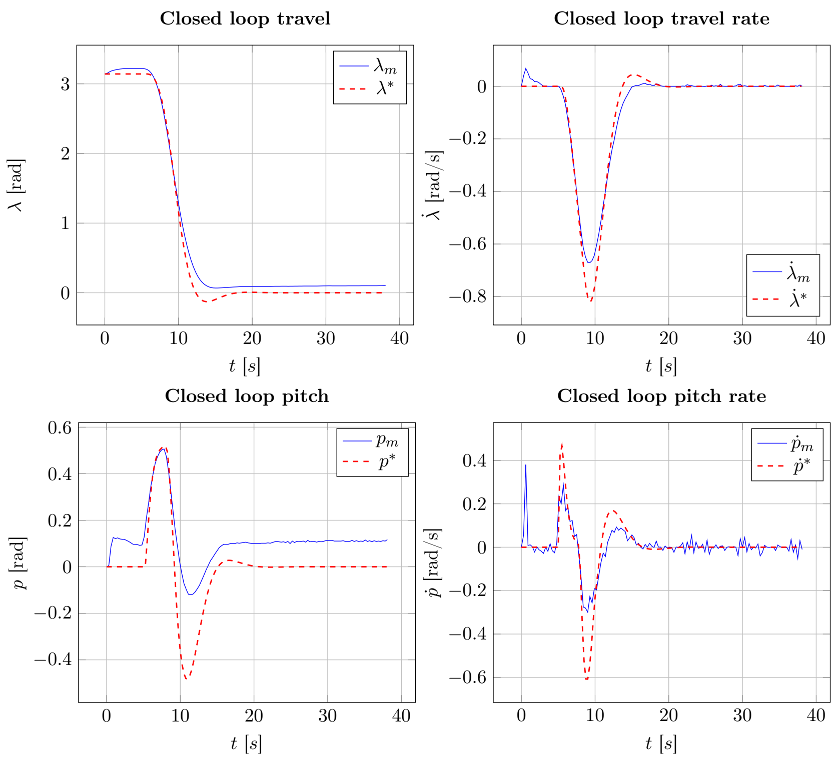

Figure 6: Measured and calculated closed loop state trajectories.

the current optimal input trajectory is. This means the system would handle large and sudden deviations much better, as it would simply recalculate the trajectory. The LQR on the other hand would only try to go back to the original trajectory, which is a lot less robust. For instance if the helicopter was ahead of its trajectory the LQR would try to slow the helicopter down in order to track the reference, while a MPC solution would calculate a better reference from the current position, which evidently is desirable. The downside is that an MPC controller controller is very computationally expensive compared to a LQR solution. When the objective function has hundreds of optimization variables and the problem needs to be solved several times per second, it demands high performance hardware. This is especially the case when we have nonlinear constraints and need to use a nonlinear solver. Nevertheless, it would be possible to implement a MPC algorithm in this system, which would make it more robust. When using MPC, strategies like input blocking could be applied to improve the computation time.

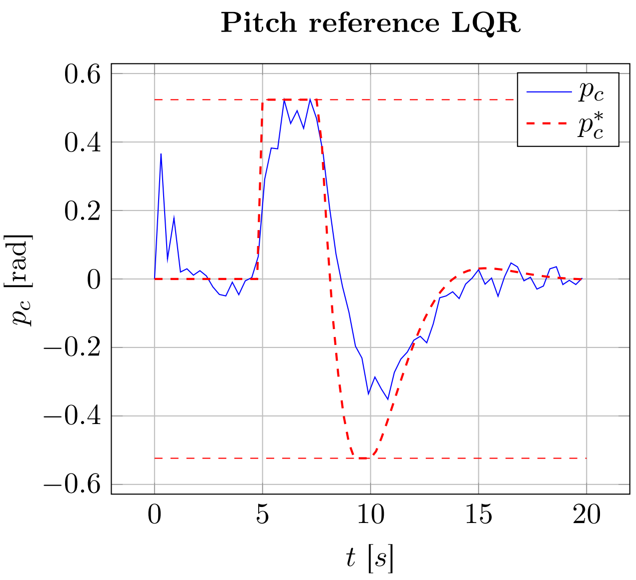

Figure 7: Optimal input sequence and input sequence from LQR

4.2 Optimal control of elevation, pitch and travel

4.2.1 Results

The state transition matrix and input matrix for the system with states and inputs as specified in eq. (11) were determined to be

$$\boldsymbol{A_d} = \begin{bmatrix} 1 & 0.250 & 0 & 0 & 0 & 0 \\ 0 & 1 & -0.142 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.250 & 0 & 0 \\ 0 & 0 & -0.810 & 0.100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.250 \\ 0 & 0 & 0 & 0 & -0.063 & 0.750 \end{bmatrix}, \boldsymbol{B}_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.810 & 0 \\ 0 & 0 \\ 0 & 0.063 \end{bmatrix}. \tag{39a}$$

The solution to the optimization problem with the nonlinear constraint eq. (31) was implemented in the MATLAB scripts appendix A.3 and appendix A.4, which utilizes the SQP solver `fmincon`. The open and closed loop systems where then implemented in Simulink and tested. Note that the zero padding was increased from 5 to 8 seconds, and that an initialization phase was implemented where the LQR is turned off, letting the system stabilize at zero elevation. The weight matrices where determined from tuning the closed loop system, given by

$$\boldsymbol{Q} = \begin{bmatrix} 200 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}, \quad \boldsymbol{R} = \begin{bmatrix} 20 & 0 \\ 0 & 1 \end{bmatrix}. \tag{40}$$

The gain matrix was then found to be

$$\boldsymbol{K} = \begin{bmatrix} -2.292 & -5.593 & 2.402 & 0.600 & 0 & 0 \\ 0 & 0 & 0 & 0 & 14.886 & 9.801 \end{bmatrix}. \tag{41}$$

The plots of the measured and calculated trajectories for the open and closed loop systems are in fig. 8 and fig. 9 respectively. The optimal input sequence and the calculated input sequence from the LQR are in fig. 10. Finally the nonlinear elevation constraint is plotted versus the elevation in the travel-elevation phase space in fig. 11.

4.2.2 Discussion

When comparing the open loop with the closed loop performance, there are several observations to be made. In both cases, the deviation between the measured and calculated travel values are relatively small, with the exception of a slight drop in the calculated value at around $t = 15s$, where the open loop system actually performs better than the closed loop system, in terms of largest deviation.

This is in fact also the case with the pitch performance, that is, the closed loop measured pitch trajectory deviates significantly from the optimal trajectory, compared to its open loop counterpart.

When comparing the elevation, however, the open loop system performs poorly. The closed loop system manages to follow this trajectory closely, although there is a notable deviation around the maximum elevation value between $t = 10s$ and $t = 15s$.

In order to understand why the open loop system arguably performs better with regard to some of the states, yet poorly with regard to elevation, a discussion of the weight matrices is necessary. As described above, the diagonal entries of the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ govern the extent to which the controller will penalise deviations of the states or use of inputs corresponding to that entry. The considerations taken when these matrices are determined are governed by the system specifications. When tuning the controller, the systems ability to not violate the elevation constraints representing an obstacle in the helicopter path were kept in particularly close consideration. Therefore the weights on elevation and elevation rate were tuned quite large. Especially pitch was sacrificed to better track elevation and travel. This partly explains the closed loop behaviour. As the deviation from the calculated elevation trajectory increases, the closed loop controller will prioritise following this trajectory at the expense, so to speak, of the other states.

As indicated above, this is only part of the explanation. As the model is described above, there is no reason the helicopter should be unable to successfully follow all of the calculated states simultaneously. Yet there is clearly a connection between increasing elevation and deviation between measurement and calculation in the other states.

This is due to the linearization of the model. One of the consequences of the linearization is the decoupling of the elevation and elevation rate from the rest of the system, that is, the model assumes that the elevation and elevation rate does not affect the dynamics of the other states and vice versa.

This is in reality not the case. As the value of the pitch increases, which is how the travel is actuated, the vertical component of the propeller force is reduced, thus reducing the elevation acceleration. When this effect is ignored, the optimal trajectory will in a sense "overestimate" the plants ability to follow an increasing elevation and increasing travel trajectory simultaneously, resulting in an input sequence that fails to deliver sufficient thrust to successfully follow multiple state trajectories.

Note how the deviations between measured and calculated trajectory increases as the calculated trajectory of both elevation and travel increase simultaneously. This is perhaps most clear from the elevation of the open loop system as seen in fig. 10: The curve of the calculated and measured trajectory are initially relatively similar, up until around $t = 9s$, at which point the measured trajectory changes to a slower response. This coincides with an increase in the pitch. The helicopter head is tilted in order to actuate the desired travel trajectory, at the expense of the elevation acceleration, ultimately leading to a gradually increasing deviation in measured and calculated elevation.

Another important point to make when discussing the open and closed loop responses is that the open loop system has a large drift in travel after the input sequence is applied, as seen in fig. 8. Without feedback, the offset in pitch from the discrepancies in the system results in a drift in travel, which is not corrected. By closing the loop, we observe in fig. 9 that none of the states drift, but we still have offsets. These offsets could be minimized by adding integral action to the LQR and the optimization algorithm, but an virtually inherit bias from the fact that we have to balance several states at the same time is hard to get rid of. This is observed in the large offset in pitch since the weight on pitch is small relative to elevation and travel.

As discussed above, it is evident that feedback is absolutely necessary to satisfy the elevation constraint. This can be further analyzed in the travel-elevation space in plot fig. 11. While both the open and closed loop measured trajectories does not satisfy the constraint, the closed loop system performs vastly better.

An interesting observation can be made regarding the optimal trajectory, namely that the optimal trajectory is also below the Gaussian constraint. This is due to the fact that the optimization algorithm works in discrete time, and therefore only evaluates the constraint at each time step. It will therefore miss the constraint maximum. Reducing the size of the time steps or adding additional constraints to satisfy the Gaussian constraint at the maximum could help with this issue.

Considering the problem with assuming decoupled states, making the pitch constraint narrower would minimize the nonlinear dynamics and make the model and thus the calculated trajectory more accurate. Testing with adding additional constraints is also definitely worth considering. Particularly, adding constraints on the derivatives in the system could improve performance. For example, limiting the travel rate would ensure smaller pitch angles, leading to the same benefits with regards to the issues of linearization.

Another option to manage the issue with decoupled states is to use an improved model. It would be possible to modify the optimization problem to include the nonlinear dynamics of the system, and thus get a more realistic trajectory. The downside of this approach is of course the added complexity of no longer having a linear system, which is harder to analyze and more cumbersome for the optimization algorithm. One could also consider using a more advanced method of discretization, like exact discretization if the linear system model is used. As previously discussed, using MPC instead of linear quadratic control might also prove more robust.

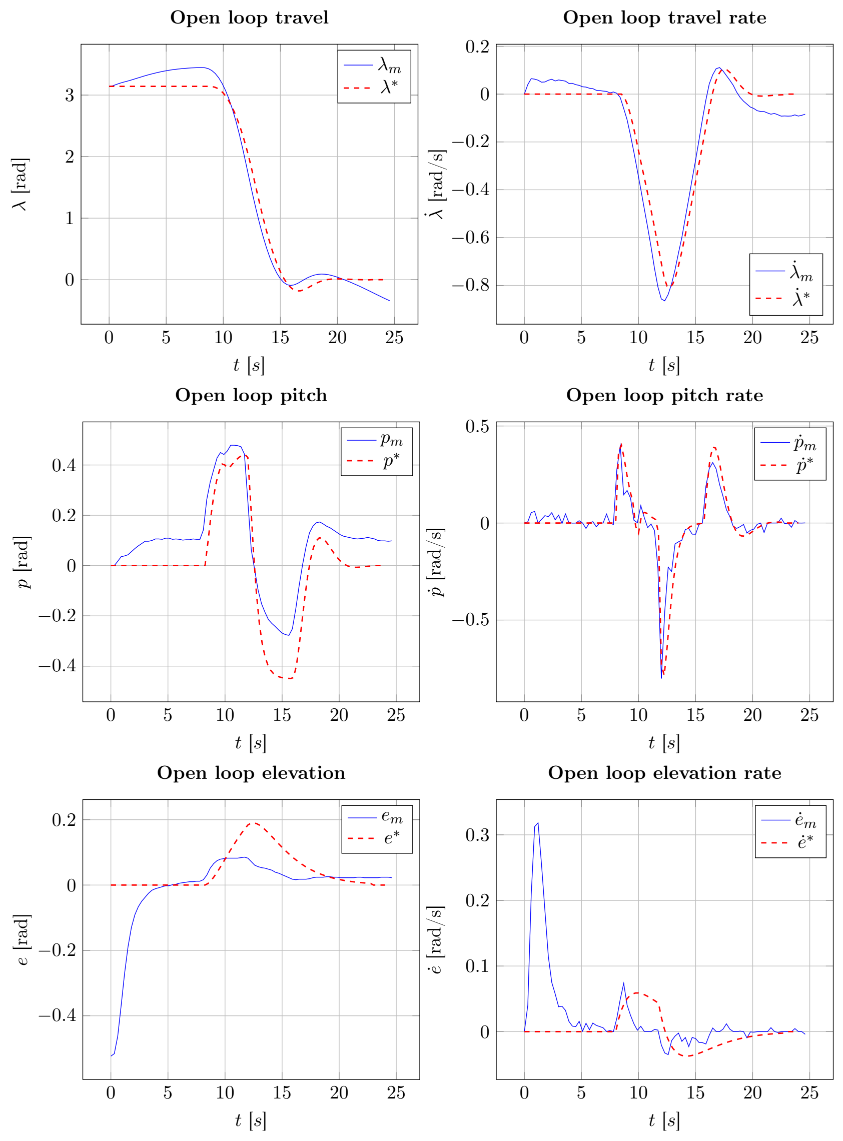

Figure 8: Measured and calculated open loop state trajectories.

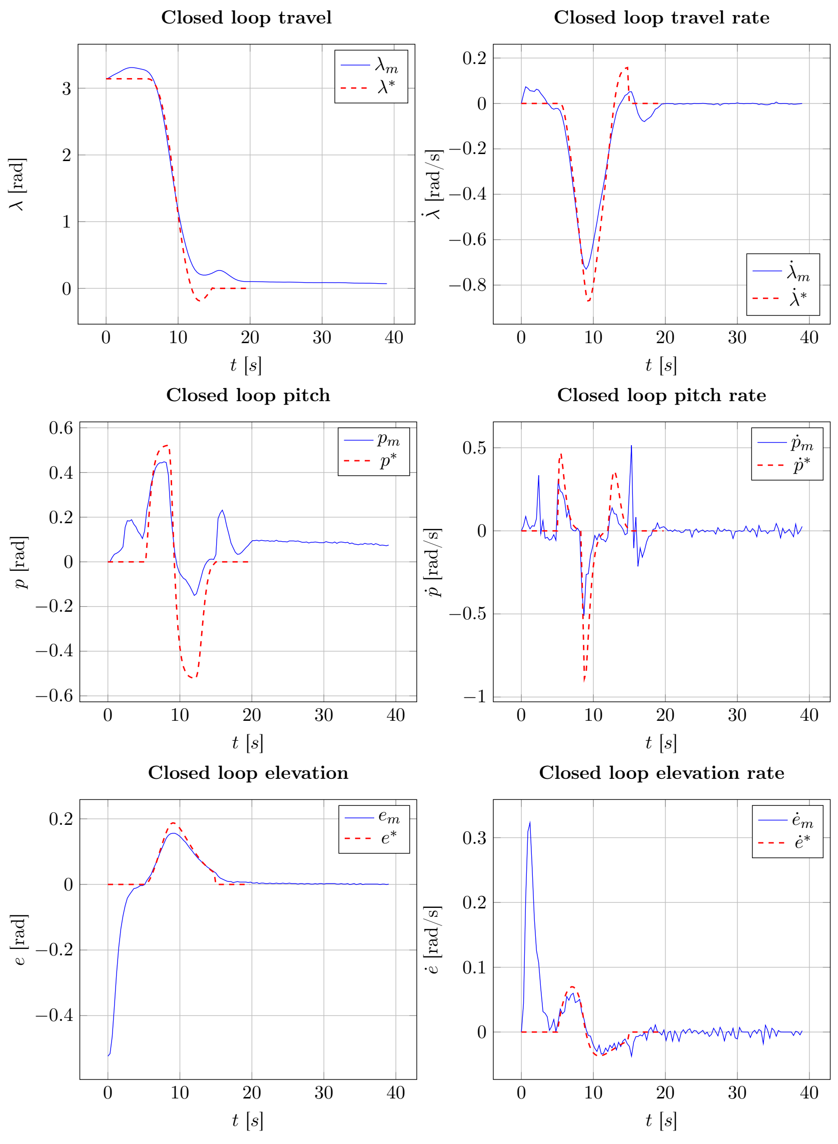

Figure 9: Measured and calculated closed loop state trajectories.

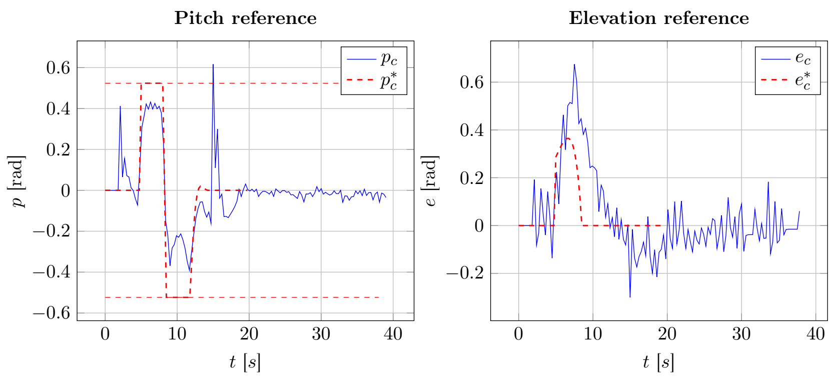

Figure 10: Optimal input sequence and input sequence from LQR.

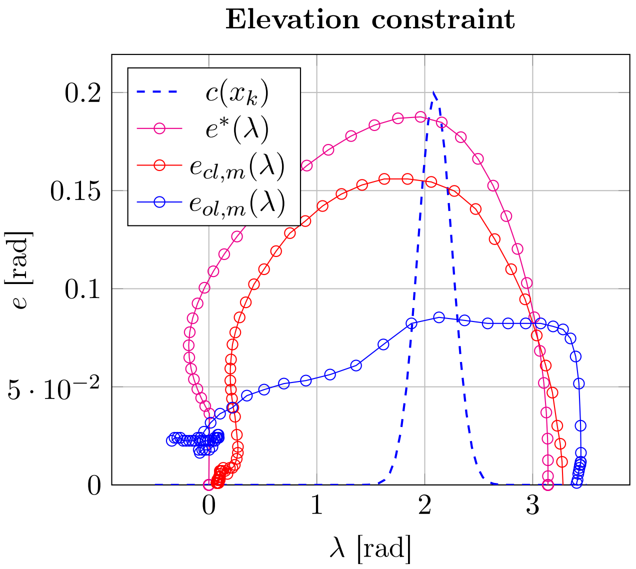

Figure 11: The elevation constraint, the optimal elevation trajectory and the measured elevation.

5 Conclusion

Throughout the exercise the implementation and practical use of optimal control, with and without feedback has been explored. First, the solution to an optimization problem, with a linear constraint on pitch, was passed to the plant. Without feedback it was evident that it struggled to follow the calculated trajectory, but for some states, it was surprisingly accurate. The model simplifications and lack of knowledge of its current whereabouts, makes this an unreliable control method. The second task demonstrated how the introduction of feedback, through the implementation of a linear quadratic controller, can improve the result. The closed loop system was more able to track the optimal trajectory, as deviations were penalized in the input. The last task introduced a nonlinear constraint to the elevation angle. The solution to the new optimization problem was given to the plant with and without feedback. Once again, the use of feedback provided better results.

A MATLAB Code

A.1 problem_2.m

```
%% Initialization and model definition
init03;

% Discrete time system model. x = [lambda r p p_dot]'
delta_t = 0.25; % sampling time
A1 = [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)];

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

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

% Initial values
x1_0 = pi;
x2_0 = 0;
x3_0 = 0;
x4_0 = 0;
x0 = [x1_0 x2_0 x3_0 x4_0]';

% Time horizon and initialization
N  = 100;
M  = N;
z  = zeros(N*mx+M*mu,1);
z0 = z;

% Bounds
ul      = -30*pi/180;
uu      = 30*pi/180;

xl      = -Inf*ones(mx,1);
xu      = Inf*ones(mx,1);
xl(3)   = ul;
xu(3)   = uu;

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

```
vlb(N*mx+M*mu)  = 0;
vub(N*mx+M*mu)  = 0;

% Generate the matrix Q and the vector c
Q1 = zeros(mx,mx);
Q1(1,1) = 1;
Q1(2,2) = 0;
Q1(3,3) = 0;
Q1(4,4) = 0;
P1 = 10;
Q = gen_q(Q1,P1,N,M);
c = zeros(500, 1);

%% Generate system matrixes for linear model
Aeq = gen_aeq(A1,B1,N,mx,mu);
beq = [A1*x0; zeros(396,1)];

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

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

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

x1 = [x0(1);z(1:mx:N*mx)];
x2 = [x0(2);z(2:mx:N*mx)];
x3 = [x0(3);z(3:mx:N*mx)];
x4 = [x0(4);z(4:mx:N*mx)];

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.2 LQR.m

```
%Weight and gain matrices for LQR
R_lqr = 1;
Q_lqr = diag([100; 1; 10; 1;]);
[K_lqr,S,e] = dlqr(A1,B1,Q_lqr,R_lqr);
```

A.3 SQP.m

```
%% Initialization and model definition
init03;

% Discrete time system model. x = [lambda r p p_dot e e_dot]'
T   = 0.25; % sampling time
A_c = [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];

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

A_d = eye(6) + T*A_c;
B_d = T*B_c;

% Number of states and inputs
mx = size(A_d,2);
mu = size(B_d,2);

% Initial values
x1_0 = pi;
x2_0 = 0;
x3_0 = 0;
x4_0 = 0;
x5_0 = 0;
x6_0 = 0;
x0 = [x1_0 x2_0 x3_0 x4_0 x5_0 x6_0]';

% Time horizon and initialization
```

```
N   = 60;
M   = N;
z   = zeros(N*mx+M*mu,1);
z0  = z;

% Bounds
ul        = -Inf*ones(mu,1);
uu        = Inf*ones(mu,1);
ul(1)     = -0.45;
uu(1)     = 0.45;
ul(2)     = -0.3;
uu(2)     = 0.3;

xl        = -Inf*ones(mx,1);
xu        = Inf*ones(mx,1);
xl(3)     = -0.45;
xu(3)     = 0.45;
xl(5)     = -0.3;
xu(5)     = 0.3;
xl(6)     = -0.2;
xu(6)     = 0.2;

% Generate constraints on measurements and inputs
[vlb,vub]        = gen_constraints(N,M,xl,xu,ul,uu);
vlb(N*mx+M*mu)   = 0;
vub(N*mx+M*mu)   = 0;

% Generate the matrix Q and the vector c
Q1 = zeros(mx);
Q1(1,1) = 1;
Q1(2,2) = 0;
Q1(3,3) = 0;
Q1(4,4) = 0;
Q1(5,5) = 0;
Q2(6,6) = 0;

P1 = zeros(mu);
P1(1,1) = 0.1;
P1(2,2) = 0.1;

Q = gen_q(Q1,P1,N,M);
c = zeros(N*(mu+mx), 1);

%% Generate system matrixes for linear model
```

```
Aeq = gen_aeq(A_d,B_d,N,mx,mu);
beq = [A_d*x0; zeros((N-1)*mx,1)];

%% Solve nonlinear problem
opts = optimset('Display','iter','Algorithm','sqp',...
'MaxFunEval',inf,'MaxIter',Inf);

f = @(z) 0.5*z'*Q*z;

[z, lambda] = fmincon(f, z0, [], [], Aeq, beq, ...
vlb, vub, @nonlcon, opts);

%% Extract control inputs and states
u1 = z(N*mx+1:mu:N*mx+M*mu);
u2 = z(N*mx+2:mu:N*mx+M*mu);

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

padding = 8/T;  %used 5 here
zero_padding = zeros(padding,1);
unit_padding  = ones(padding,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];
```

A.4 nonlcon.m

```
function [c, ceq] = nonlcon(z)

alpha = 0.2;
beta = 20;
lamda_t = 2*pi/3;
```

```
N = 40;

c = zeros(N, 1);
ceq = [];

for i=1:N
    c(i) = alpha*exp(-beta*(z((i-1)*6 + 1) - lamda_t)^2)...
    - z((i-1)*6 + 5);
end
```

A.5 LQR2.m

```
%Weight and gain matrices for LQR
R_lqr = diag([20; 1]);
Q_lqr = diag([200; 20; 10; 5; 500; 10]);
[K_lqr,S,e] = dlqr(A_d,B_d,Q_lqr,R_lqr);
```

B Parameters and values

Table 1: Parameters and values.

Symbol	Parameter	Value	Unit
l_c	Distance from elevation axis to counterweight	0.5	m
l_a	Distance from elevation axis to helicopter head	0.65	m
l_h	Distance from pitch axis to motor	0.17	m
K_f	Force constant motor	0.0463	$\frac{N}{V}$
J_e	Moment of inertia for elevation	0.338	$kg\,m^2$
J_λ	Moment of inertia for travel	0.338	$kg\,m^2$
J_p	Moment of inertia for pitch	0.0116	$kg\,m^2$
m_h	Mass of motors	0.4	kg
m_g	Effective mass of the helicopter	0.03	kg

bibliography