TTK4135 Optimization and Control
Lab Report

Sigvart Hovland (732141)
Jens Kretschmer (771354)
Thomas Lürkens ()
Rafael Schwarzenegger (772012)
Group 4

May 6, 2016

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Abstract

The purpose of the lab in TTK4135 was to control a model of an helicopter that consisted of an arm, counter weight and two propellers attached to electrical DC motors. This was to gain real world experience in using optimization theory and algorithms with control theory. The helicopter were going to be controlled by a optimal sequence of inputs which were derived from a linearised model. Then test this input by running it in a open-loop system and in a closed-loop system with and without a linear quadratic regulator(LQR).

The whole lab were conducted using Matlab and an extension of Matlab called QuaRC which is a hardware-in-the-loop software generating machinecode running on external hardware which controlled the helicopter.

Contents

1 Introduction

In this lab we are going to use the techniques and theory taught in the course of optimization theory and control. To control a small helicopter model. We will be doing this by computing the optimal set of inputs for reaching a setpoint for the helicopter and give a corresponding input for the helicopter and see if it reaches its intended trajectory.

To compute this we need to derive a non-linear model for the dynamics of the system. Then linearize the model around an equilibrium. This method is widely used in control theory and have a good preformance if the system operates close to the linearization point. Due to errors could occure from linearization and unmodeled coupling and states we will try out a few different setup to measure the performance of the optimal control computed.

One where we use the optimal input directly without any feedback to see if there is any drift in the system indicating erros in either the linerization and/or the model. Also we will try to compensate for the inaccuracy of the model by using a feedback controller to eliminate drift. In this lab we will be using an linear-quadratic regulator to control the feedback loop. We will also discuss using an MPC instead of an LQR. Additional constraints will also be added to see if they have any inpact in limiting the nonlinear effects on the system.

2 Problem Description

2.1 Lab Setup

The helicopter, as shown on figure 1, is constructed from two main parts. The basis and the arm. The arm has got on one side two propellers and on the opposite side a counter weight. The arm has got 2 degrees of freedom and can therefore move up and down. The two propellers are attached to the arm by a rotary bound, as shows figure 2. The body of the helicopter can also rotate around its axis. Combined with the movement of the arm we observe the travel. The rotation of the propellers around the arm is denoted as the pitch.

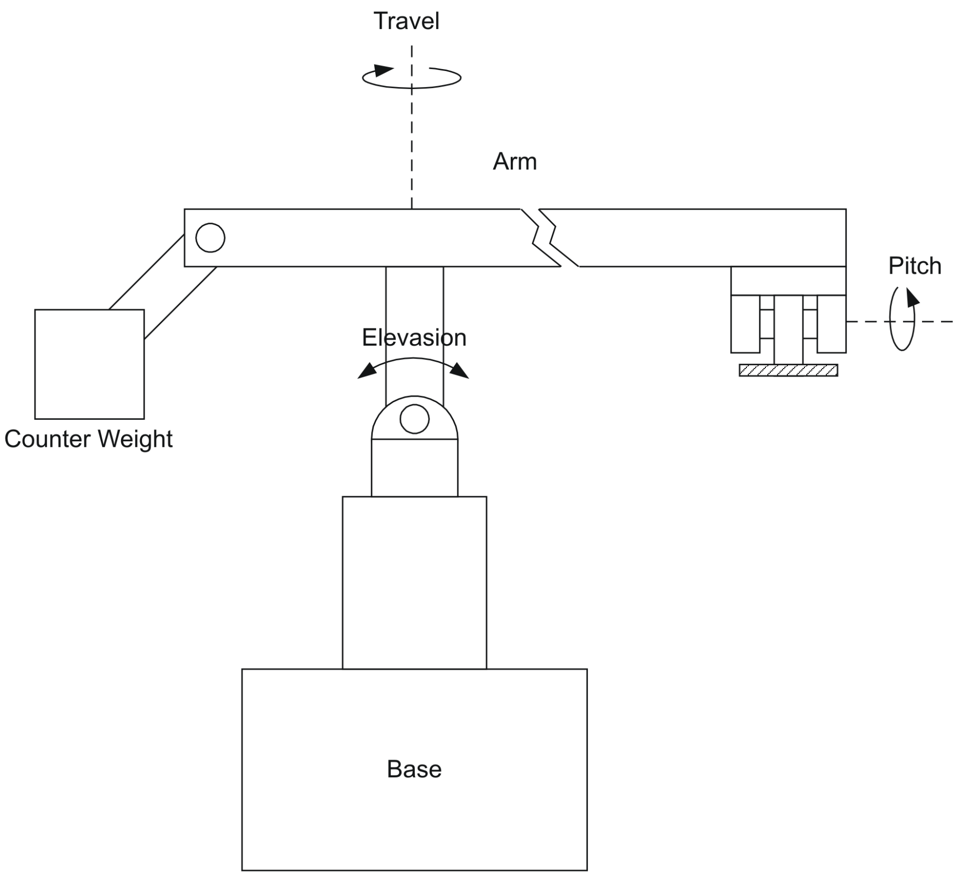

Figure 1: The elevation of the helicopter (Department of Engineering Cybernetics, 2014).

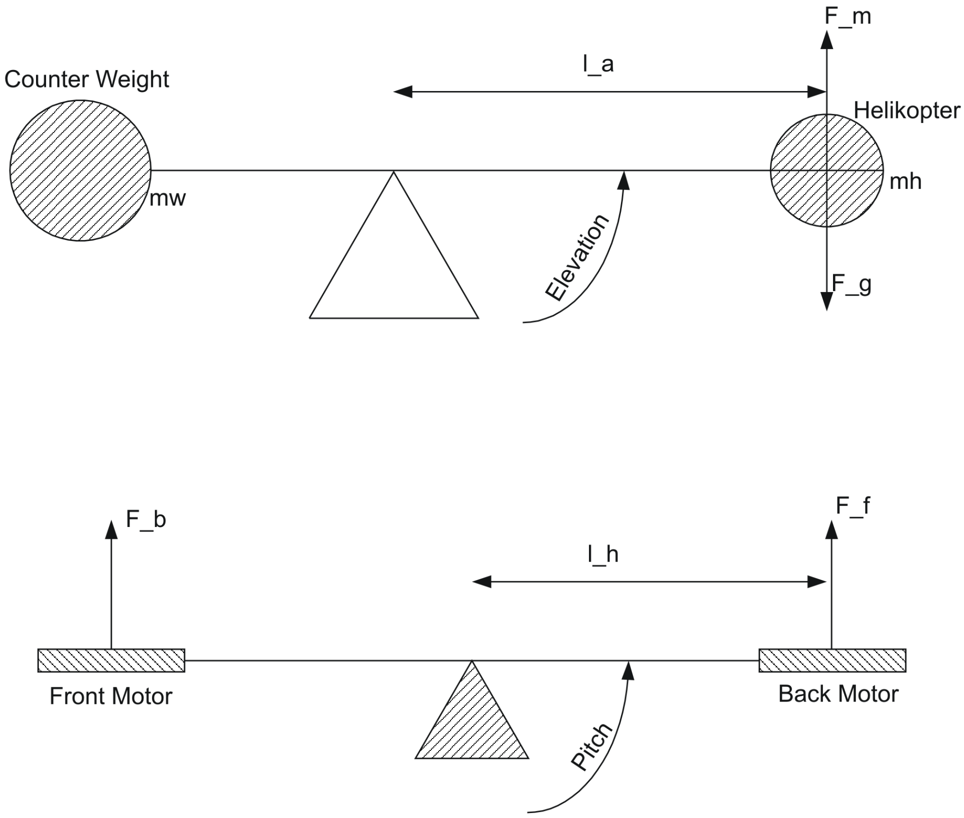

Figure 2: sketch of the helicopter (Department of Engineering Cybernetics, 2014).

2.2 Model derivation

The model is given by the quations (1). Equation (1a) describes the elevation, equation (1b) accesses the pitch angle. The speed is the derivation of the of the path as described in equation (1c). The travel acceleration is given by equation (1d).

$$\ddot{e} + K_3 K_{ed} \dot{e} + K_3 K_{ep} e = K_3 K_{ep} e_c \tag{1a}$$

$$\ddot{p} + K_1 K_{pd} \dot{p} + K_1 K_{pp} p = K_1 K_{pp} p_c \tag{1b}$$

$$\dot{\lambda} = r \tag{1c}$$

$$\dot{r} = -K_2 p \tag{1d}$$

The equation of elevation was derived from:

$$J_2 \ddot{e} = l_a K_f V_s - T_g \tag{2}$$

s.t.

$$\ddot{e} = K_3 V_s - \frac{T_g}{J_e}, \; K_3 = \frac{l_a K_f}{J_e}.$$

Similarly the equation for the pitch angle:

$$J_p \ddot{p} = K_f l_h V_d \tag{3}$$

s.t.

$$\ddot{p} = K_1 V_d, \; K_1 = \frac{K_f l_h}{J_p}.$$

Finally the travel acceleration:

$$J_t \ddot{r} = -K_p l_a \sin p$$

under the assumption $\sin p \approx p$. Then,

$$\dot{r} = K_2 p, \; K_2 = \frac{K_p l_a}{J_t}. \tag{4}$$

We remember that $\dot{\lambda} = r$.

3 Repetition/Introduction to Simulink/QuaRC

Simulink is a program for Model-Based Design. It overtakes the code from Matlab, compiles it into the C language and sends it to the right controllers. QuaRC[1] is a Real-Time control system, that is integrated into Simulink. To control the program, Matlab and Simulink is used. We use QuaRC for the build option as can be shown on figure 3. The work flow is that we first build the program. In Simulink is than the Matlab code compiled to the C language with Visual C++. Then the code is downloaded to QuaRC. We have also to set the following parameters, if we already didn't do so, like buffer size, sampling frequency. After this, the helicopter can be started. We assure that the power button at the helicopter is on and on the computer we can push Start. After the flight, we can compare the expected flight from the real flight in a Matlab figure. The realtime measurments will always be shown up as a piece wise constant function.

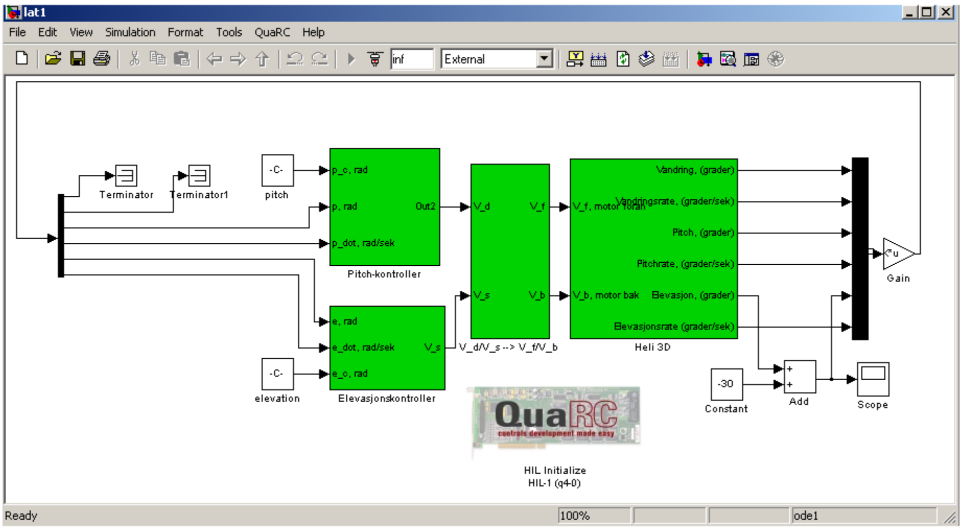

Figure 3: A diagram of a Simulink work flow (Department of Engineering Cybernetics, 2014)

[1] http://www.quarcservice.com/ReleaseNotes/files/quarc_user_guide.html

4 Optimal Control of Pitch/Travel without Feedback

4.1 State space form

For Problem 2 an optimal control sequence had to be calculated disregarding the elevation e and relinquishing feedback. The states were given by the problem description with $\mathbf{x} = \begin{bmatrix}\lambda & r & p & \dot{p}\end{bmatrix}^\top$ and the only input was given by $u = p_c$. Therefore the model can be written in time state space form in the following way:

$$\underbrace{\begin{bmatrix}\dot{\lambda}\\ \dot{r}\\ \dot{p}\\ \ddot{p}\end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & -K_2 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd}\end{bmatrix}}_{\mathbf{A_c}} \underbrace{\begin{bmatrix}\lambda\\ r\\ p\\ \dot{p}\end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix}0\\ 0\\ 0\\ K_1K_{pp}\end{bmatrix}}_{\mathbf{B_c}} \underbrace{p_c}_{\mathbf{u}} \tag{5}$$

4.2 Discussion of the model

The control hierarchy of the model is displayed in following figure:

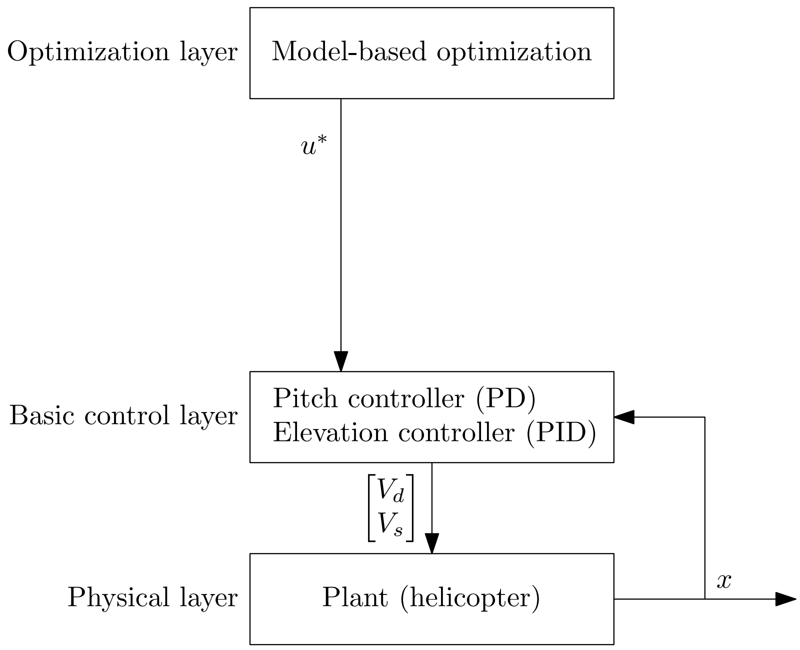

Figure 4: Control hierarchy[2]as used in Problem 2

Since the model as presented above also contains the pitch and elevation setpoints as inputs it does not only depict the phyiscal behaviour of the

[2]as depicted in the problem description

helicopter. Instead also the inner control loops of pitch and elevation are modeled.

4.3 Discretisation

In order to use the time state space form to calculate the optimal trajectory from the optimal control sequence it had to be discretised in time. For the discretisation the forward Euler method was used:

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \Delta t \dot{\mathbf{x}}_\mathbf{k} \tag{6}$$

using the time state space form (5)

$$\begin{aligned}\mathbf{x_{k+1}} &= \mathbf{x_k} + (\mathbf{A_c}\mathbf{x_k} + \mathbf{B_c}u_k)\Delta t \\ &= (\mathbf{I} + \Delta t\mathbf{A_c})\mathbf{x_k} + \Delta t\mathbf{B_c}u_k \\ &= \mathbf{A}\mathbf{x_k} + \mathbf{B}u_k \end{aligned} \tag{7}$$

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

For the discretisation a time step $\Delta t = 0.25$s was used.

4.4 Optimal trajectory

For the starting point $\mathbf{x_0} = \begin{bmatrix}\lambda_0 & 0 & 0 & 0\end{bmatrix}^\top$ and the terminal point $\mathbf{x_f} = \begin{bmatrix}\lambda_f & 0 & 0 & 0\end{bmatrix}^\top$ the optimal trajectory had to be calculated. λ_0 was set to π and λ_f was set to 0.So the helicopter should make a curve of 180°and then remain at his position. In order to find the optimal trajectory the following cost function was minimised with different weightings q for the pitch input p_{ci}:

$$\phi = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + qp_{ci}^2, \quad q \geq 0 \tag{9}$$

Additionally the following constraint on the input value was implemented:

$$|p_k| \leq \frac{30\pi}{180}, \quad k \in \{1, \ldots, N\} \tag{10}$$

The constraint prevents the helicopter from flying rather extreme manoeuvres where the pitch angle exceeds 30°and thus contributes to the overall safety of the equipment and the laboratory staff. Since the cost function is quadratic in both variables, the pitch input and the deviation of the position from the terminal point, it can be solved by quadratic programming algorithms. Though the helicopters position will always be compared to the terminal point. This may result in fast helicopter movement and aggressive steering, which can be hard to handle due to the helicopter's momentum. High pitch angles may also exceed the area in which the system equation are sufficiently accurate since they rely on small angle approximations. Also aggressive inputs on the pitch control may cause the pitch angle to exceed the constraints due to the momentum of the helicopter.

The MATLAB code can be seen in appendix A.1 and the Simulink diagram can be seen in fig. 13.

4.5 Results of the optimization

To solve the optimization problem the MATLAB function `quadprog` was used. Values of 0.1, 1, and 10 were chosen for the weighting factor q. 5 seconds of zeroes were added before and after the control sequence in order to give the helicopter some time to stabilise. The results were exported from MATLAB and are depicted in the Figures (5), (6) and (7).

As expected a lower value for q allows much stronger inputs than higher values. This is logical since a higher value for q emphasizes the influence of the inputs in the cost function. Therefore the higher q is the less inputs are given in order to keep the value of the cost function low. Still, regardless of the value of q the helicopter does not remain at his terminal point but continues moving as we can see in the plot of λ. This behaviour can be ascribed to the missing feedback. The helicopter does only follow the control sequence that has been calculated in advance and does not compare the setpoints with his current state.

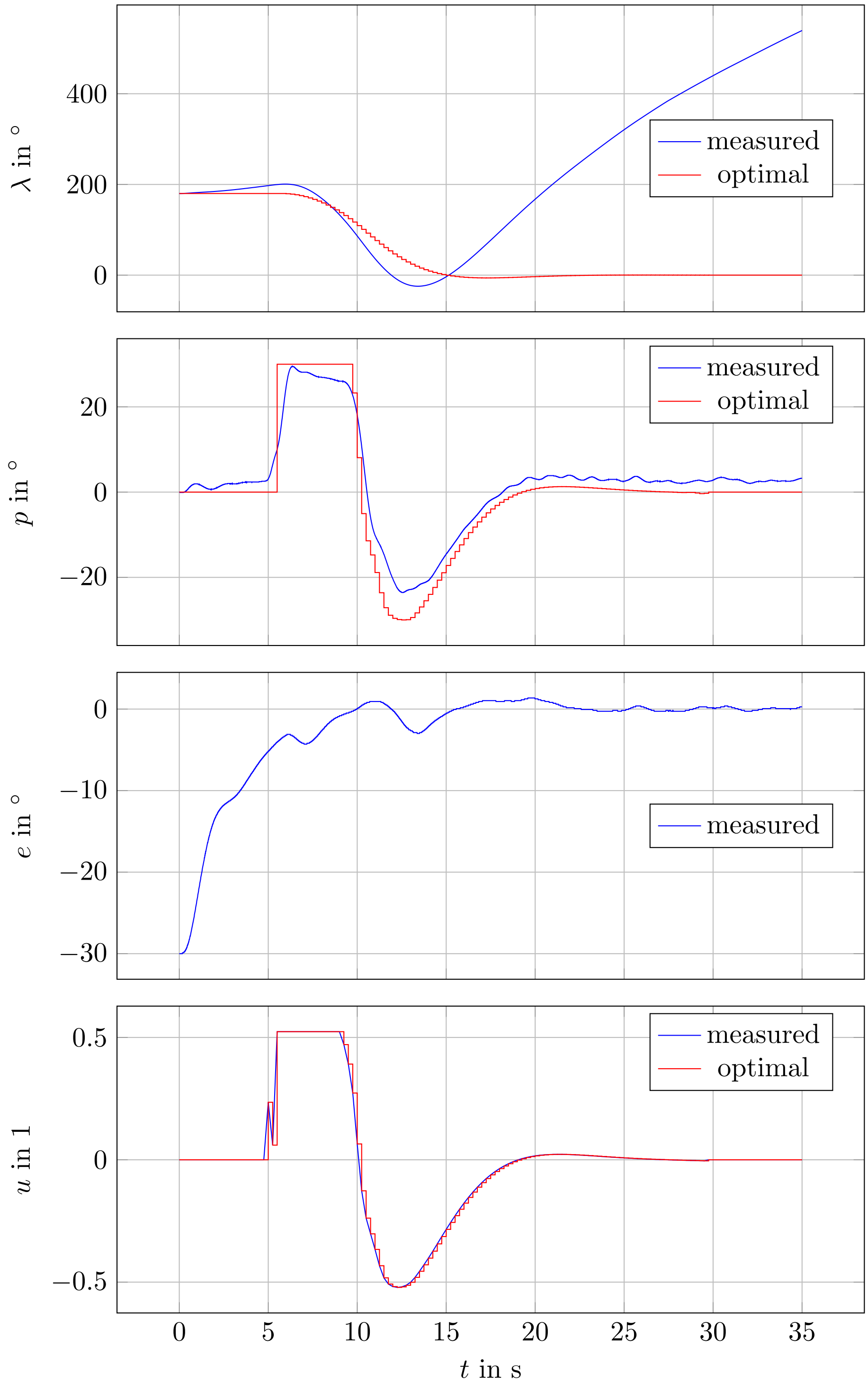

Figure 5: Results for $R = 1.0$

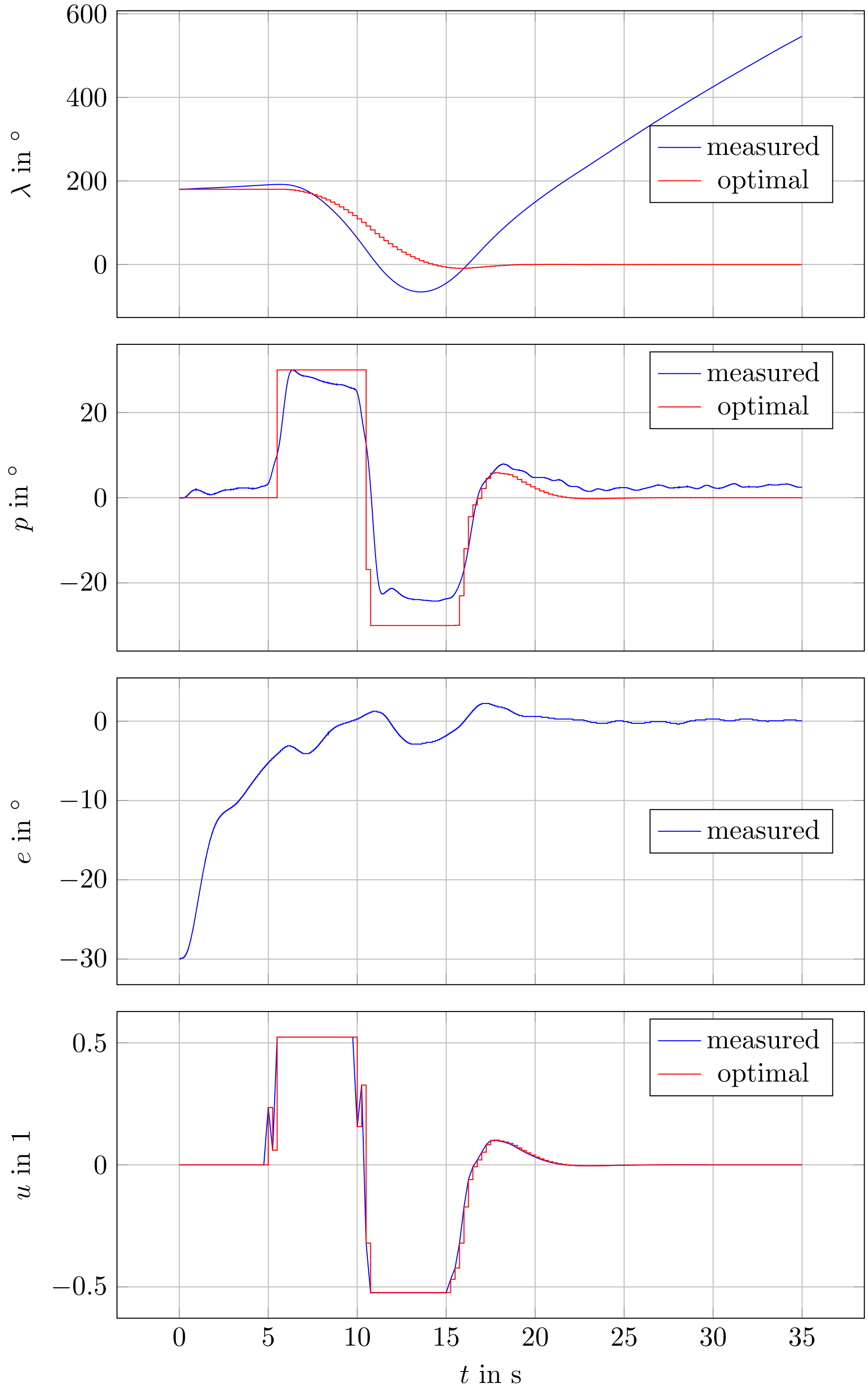

Figure 6: Results for $R = 0.1$

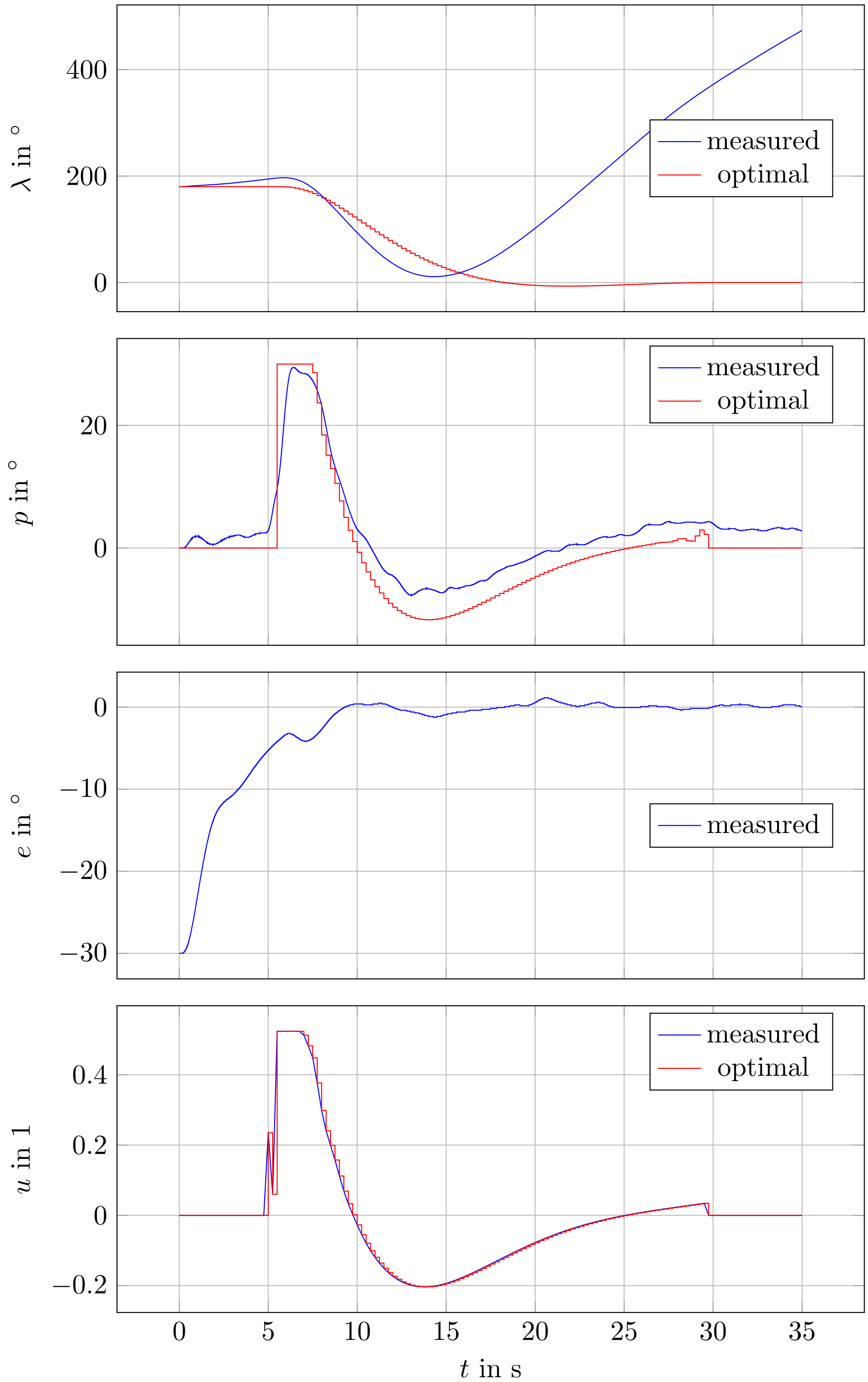

Figure 7: Results for $R = 10$

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

5.1 Introducing feedback

In Problem 3 the open control loop from Problem 2 had to be closed by adding a feedback loop to the optimal controller. In order to do so the input had to manipulated in the following way:

$$u_k = u_k^* - K^\top (x_k - x_k^*) \tag{11}$$

As long as the helicopter is following the optimal trajectory x_k^* only the optimal input will be applied. If the helicopter deviates from the optimal trajectory countermeasures depending on the matrix K will be taken. The matrix K was calculated by a LQ controller which minimises the quadratic objective funktion J for the linear model:

$$J = \sum_{i=0}^{\infty} \Delta x_{i+1}^\top Q \Delta x_{i+1} + \Delta u_i^\top R \Delta u_i, \quad Q \geq 0, R > 0 \tag{12}$$

The weighting matrices Q and R will influence the behaviour of the feedback control. Higher values in the Q-matrix will punish deviations from the optimal trajectory harder while higher values in the R-matrix will punish the usage of the manipulated variable. The optimal K-matrix was found by using the `dqlr`-function from MATLAB. The MATLAB code can be seen in appendix A.2 and the Simulink diagram can be seen in fig. 14 and fig. 15.

5.2 Results

Different values for Q and R were implemented. Some of the results are shown in the Figures (8) and (9)
The best fit to the optimal trajectory could be attained by the weighting displayed in Figure 9. Due to the closed feedback loop most variations of Q and R remain at the terminal point. Though in some cases there is still some movement of the helicopter visible or the helicopter does not reach the terminal point at all (compare Figure 8). For most tested variations there appear oscillations in the pitch. An exceptional case can be seen in Figure 8. Although the movement in pitch seems to be more controlled in these cases, they are amongst the worst options considering the deviation of the measured traveling from the optimal trajectory.

5.3 Comparison to Model Predictive Control

An alternative strategy would be to use an MPC controller. This could be implemented calculating by a solution of a numerical optimization problem corresponding to a finit-horizon optimal control problem at each sampling instant. Using the current state computing the optimal trajectory from the current state to the targeted state given a finite window of time. This would mean that only the first window/initial state of the optimal control would be used and the rest would be calcualted on the fly using the same method.(Johansen, 2011)

The concept of optimal control, and in particular its practical implementation in terms of Nonlinear Model Predictive Control (NMPC) is an attractive alternative since the complexity of the control design and specification increases moderately with the size and complexity of the system. In particular for systems that can be adequately modeled with linear models, MPC has become the de-facto standard advanced control method in the process industries. (Johansen, 2011)

However this is still an emerging field in UAV and Aerial application this is due to the computational expensiveness of the MPC as a QP problem would use a lot more time pr. iteration than just an LQR. The nonlinear programming problem may have multiple local minima and will demand a much larger number of computations at each sample, even without providing any hard guarantees on the solution. Hence, NMPC is currently not a panacea that can be plugged in to solved any control problem. Also to have good control in Aerial applications you would have to have a very short window/time-step which increases computation time again. An MPC with longer time steps would yield worse preformance than the LQR due to the fast dynamics of the system. (Johansen, 2011)

Also any deviation from the optimal trajectory is likely caused by an incomplete model of the system due to simplification. Or due to unmodeled external distrubances such as wind or drag. An MPC controller would have a hard time handeling it if not modelled.

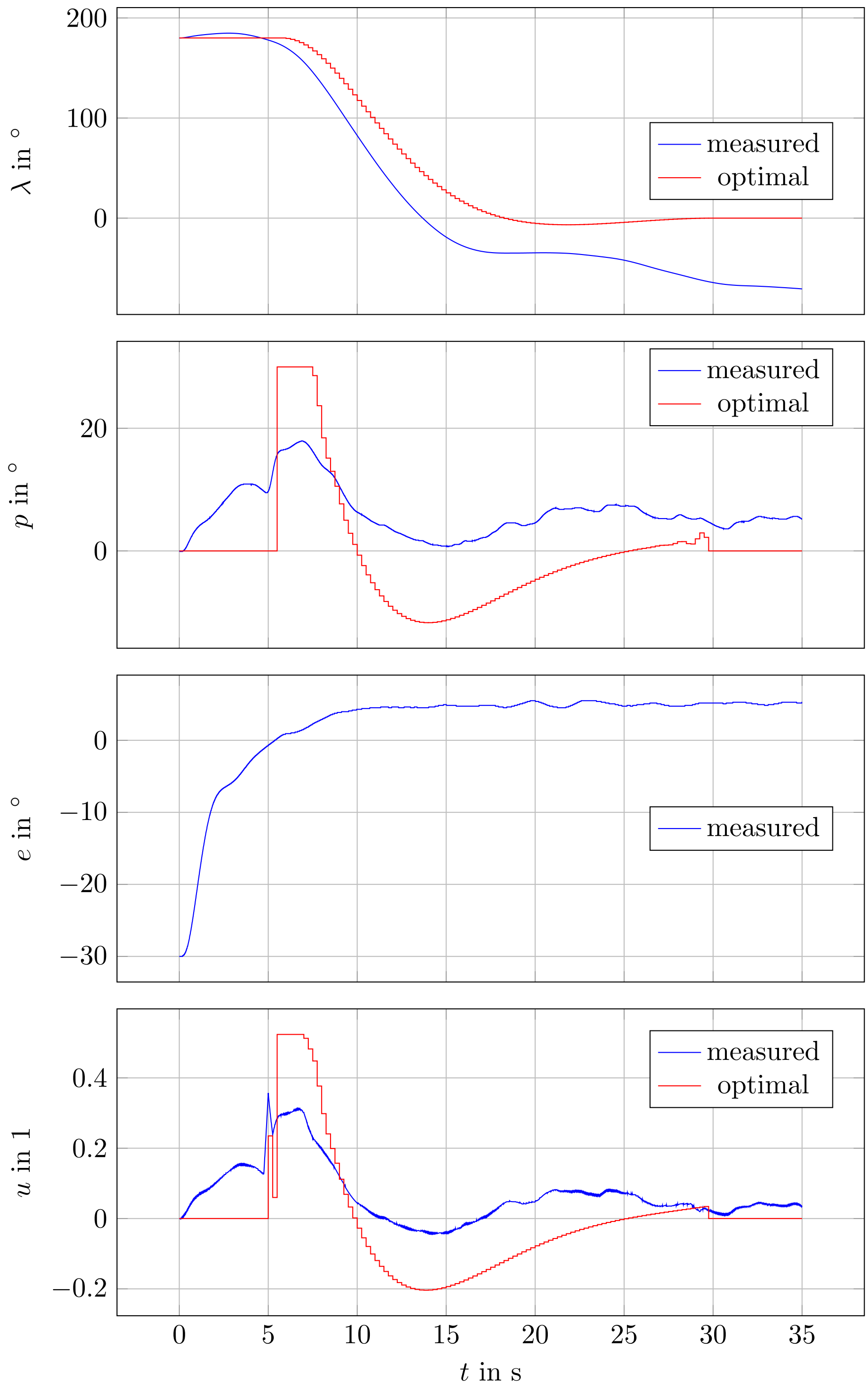

Figure 8: Results for $Q = [1, 0, 10, 10]$ and $R = 0.5$

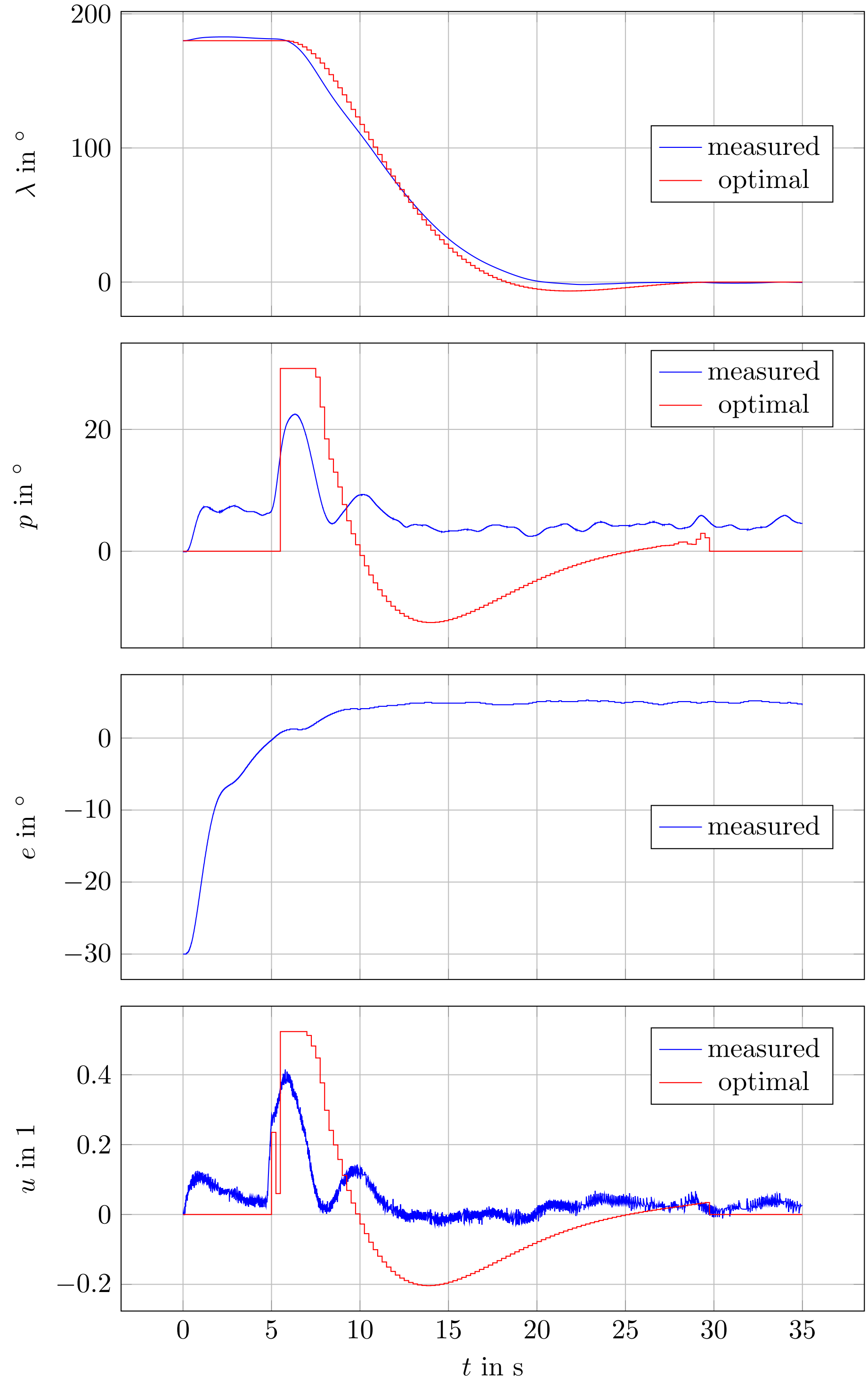

Figure 9: Results for $Q = [10, 1, 5, 0]$ and $R = 0.5$

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

In this part of the excersice a constraint on the elevation is added. Therefore the equation describing the dynamics of the elevation e from (1) must be added to the state space representation of the model of the helicopter

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{A}_c\mathbf{x} + \mathbf{B}_c\mathbf{u} \tag{13a}$$

with

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1K_{pp} & -K_1 * K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3 * K_{ep} & -K_3 * K_{ed} \end{bmatrix} \tag{13b}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3K_{ep} \end{bmatrix} \tag{13c}$$

$$\mathbf{x} = \begin{bmatrix} \lambda & r & p & \dot{p} & e & \dot{e} \end{bmatrix}^T \tag{13d}$$

$$\mathbf{u} = \begin{bmatrix} p_c & e_c \end{bmatrix}^T . \tag{13e}$$

The new input e_c is the stepoint of the elevation. The continuous model is then converted to a time discrete model

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u} \tag{14}$$

with the forward Euler method

$$\mathbf{A} = \mathbf{I}_{6\times 6} + \mathbf{A}_c\Delta t \tag{15a}$$

$$\mathbf{B} = \mathbf{B}_c\Delta t \tag{15b}$$

with Δt being the sampling time.

The cost function

$$\phi = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q_1 p_{ci}^2 + q_2 e_{ci}^2 \tag{16}$$

is used as a minimization criteria, with the final value for the travel $\lambda_f = 0$ and $q_1 = 1$ and $q_2 = 2$. The values for q_1 and q_2 are chosen this way to reduce the oscillations in the opimal trajectory of p and $\dot{p}$ which occur if $q_1 = q_2 = 1$ is used.

The initial value $\mathbf{x}_0 = \begin{bmatrix}\pi & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T$ is used to ensure a travel distance of π.

As in section 4 a constraint of $\pm 30°$ is used for the pitch p and the setpoint of the pitch p_c. Input constraints of $\pm 60°$ for e_c are added to avoid a collision between the helicopter and the table on which the helicopter is mounted. Since (15) needs to be valid at each time step the equaions are added as equality constraints

$$\mathbf{A}_{eq} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & -\mathbf{B} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ -\mathbf{A} & \mathbf{I} & \ddots & & \vdots & \mathbf{0} & \ddots & \ddots & & \vdots \\ \mathbf{0} & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathbf{0} & \vdots & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{A} & \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0} & -\mathbf{B} \\ \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{I} & \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{0} \end{bmatrix} \tag{17a}$$

$$\mathbf{B}_{eq} = \begin{bmatrix} \mathbf{A}\mathbf{x}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{x}_f \end{bmatrix}, \tag{17b}$$

with $\mathbf{x}_f = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T$ being the final state at $t = N$. A nonlinear constraint

$$c(\mathbf{x}_k) = \alpha \exp\left(-\beta (\lambda_k - \lambda_t)^2\right) - e_k \leq 0 \quad \forall k \in \{1, \ldots, N\} \tag{18}$$

with $\alpha = 0.2$, $\beta = 20$ and $\lambda_t = \frac{2\pi}{3}$, is added. Since (18) is nonlinear the optimization problem is nonlinear and therefore a nonlinear solver is used. The MATLAB command `fmincon` with three different algorithms is used. The SQP algorithm converges to a solution where the tarjectory of the travel λ consists of a single step from π to 0, which is unphysical and therefore cannot be used as a reference trajoctory for the helicopter. The active-set method converges to a solution where the input u_1 is 83 % of the time at saturation limit. Although this is only the open loop trajectory it means that when the loop is closed using the LQR the input is still at the saturation limit (assuming that the model is perfect and that there are no disturbances) which means that the control loop is actually open. Because of that the interior-point method is used which results in an trajectory where the input is only 8 % of the time at the saturation limit. The computation time for calculating the trajectory are 0.33 s for the SQP method, 8.06 s for the active-set method and 53.51 s for the interior-point method, but this is of little concern since the optimization problem doesn't need to be solved online.

The code for calculating the optimal trajectory is shown in appendix A.3, the nonlinear constraint function can be seen in appendix A.4 and the calculation of the LQR gain matrix is done in appendix A.5. The simulink diagram is shown in fig. 16 and fig. 17.

The time curve of the helicopter without feedback can be seen in fig. 10. As in section 4 the trajectory of the pitch p is followed, which is due to the pitch control loop which helps to counteract for modeling errors and that a linear model of the nonlinear system is used. The same applies for the elevation control loop which had a reference point of zero in the last two sections and has a trajectory unequal to zero due to the nonlinear constraint (18) in this section. The reference trajectory of the travel λ is not followed satisfactorily. This is the case because the travel λ is controlled in open loop and due to modeling errors the input is not calculated correctly which causes the severe deviations.

As in section 5 an LQR is used as feedback controller. The input is then calculated by

$$\mathbf{u}_t = \mathbf{u}_t^* - \mathbf{K}\left(\mathbf{x}_t - \mathbf{x}_t^*\right) \tag{19}$$

with $\mathbf{u}_t^*$ being the optimal input sequence and $\mathbf{x}_t^*$ being the optimal trajectory of the states. This new input sequence $\mathbf{u}$ is then used instead of the optimal input trajectory $\mathbf{u}_*$ to ensure that the deviations of the desired travel trajectory λ is reduced. The weighting matrices

$$\mathbf{Q} = \operatorname{diag}\left(5, 1, 1, 1, 1, 1\right) \tag{20a}$$

$$\mathbf{R} = \operatorname{diag}\left(1, 1\right) \tag{20b}$$

are used, which results in a feedback matrix

$$\mathbf{K} = \begin{bmatrix} -0.1899 & -0.6725 & -0.7316 & 0.0272 & -0.0000 & -0.0000 \\ 0.0000 & 0.0000 & -0.0000 & -0.0000 & 0.0892 & 0.4678 \end{bmatrix}. \tag{21}$$

The values for $\mathbf{Q}$ and $\mathbf{R}$ are chosen such that the trajectory of the travel λ is followed with small deviations and that there are no oscillations. Higher weights on the travel λ cause oscillations. Higher weights on the pitch p cause large deviations in the travel, since the controller tries to decrease deviations between the pitch p and the optimal pitch trajectory p^*. Higher weights on the input u_1 has approximately the same effect as higher weights on the pitch p. The weight on the input u_2 influences the behaviour only minimally.

The time curve of the helicopter with an LQR as feedback controller is shown in fig. 11. The deviations from the trajectory of the travel λ are much smaller than compared to the ones in fig. 10. Apart from that there are less oscillations in the time curve of the elevation e.

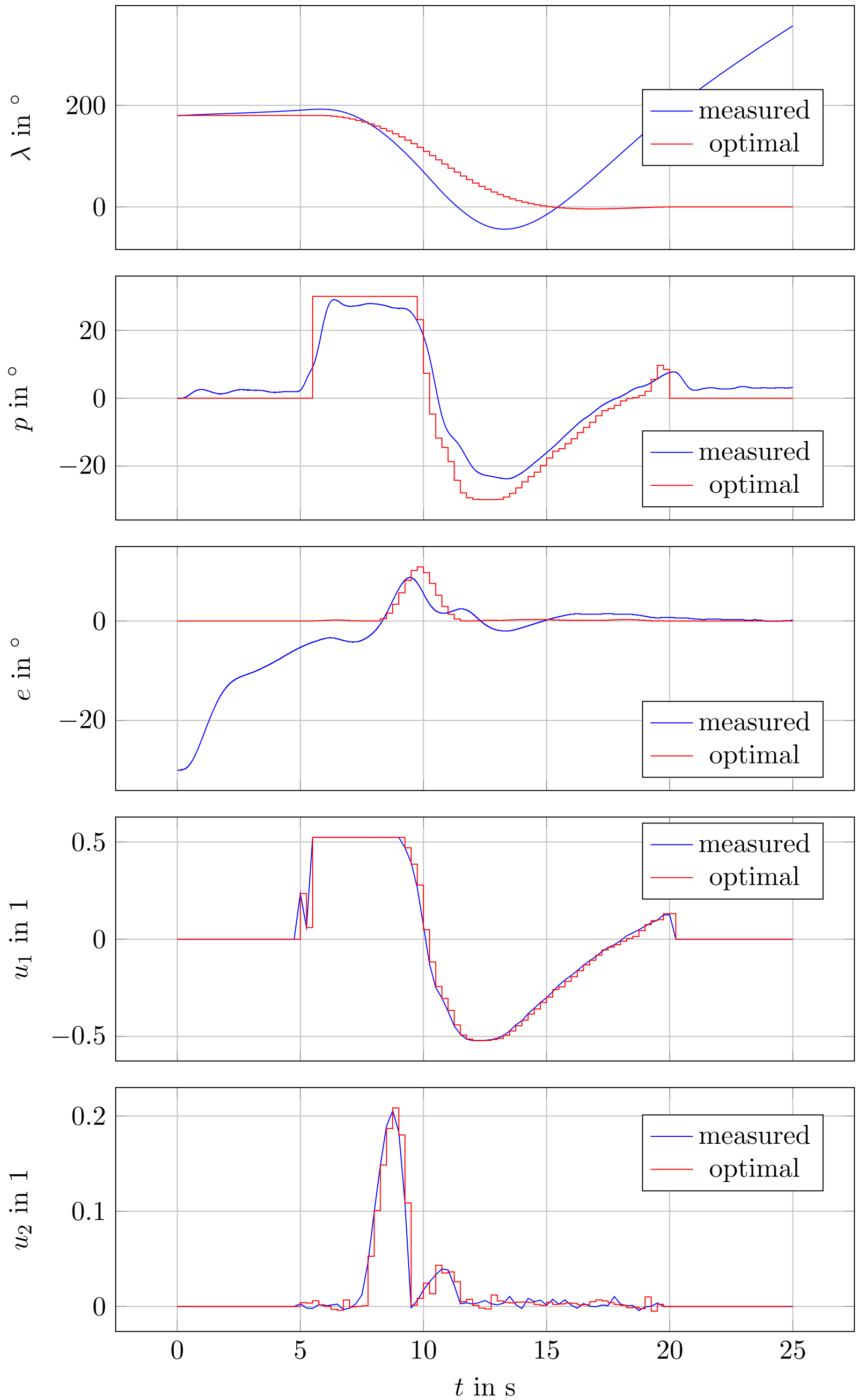

Figure 10: Time curve of inputs and states without using feedback.

The constraint on the elevation is not satsfied perfectly due to the coupling of the pitch p and travel λ with the elevation e which is not considered in the simplified model which is used in this laboratory. The optimal input sequences don't incorporate the coupling because of the simplified model and thereby cause the deviations. The decoupling of the simplified model can also be observed in the matrix $\mathbf{K}$ which has a block diagonal structure. The coupling of the two subsystems can be observed when e.g. the pitch is 90°, both turbines have no effect on the elevation since the direction of force and the direction of the elevation angle are orthogonal. A way to improve the performance would be to use a better model for the optimization problem which incorporates the coupling.

The deviations from time curve of the pitch p are larger but they are negligible since the goal is to control the travel λ and to satisfy the constraints. Due to the feedback the noise of the measurement can be seen in in the input trajectory. The impulses that can be observed in the input u_2 occur because the rates r, $\dot{p}$ and $\dot{e}$ are estimated using filters of the type

$$G(s) = \frac{sT}{s+T} \tag{22}$$

which have a differentiating behaviour for frequencies lower than $\omega = \frac{1}{T}$. Since the measurements of λ, p and e have discrete values a step occurs whenever the value changes which results in an impulse in the velocity estimation.

To improve the optimal trajectory constraints on r and $\dot{e}$ are introduced. The constraints are

$$-0.15 \leq r \leq 0.15 \tag{23a}$$

$$-0.12 \leq e \leq 0.12 \ . \tag{23b}$$

The time curve with additional constraints on r and $\dot{e}$ is shown in fig. 12. The number of steps N is increased to 100 to get a feasible solution. The reason behind this is, that it takes a certain amount of time respectively time steps N to to move from π to 0 with a certain maximum value of the travel rate r. The increased number of steps cause the computation time to rise to 2100 s compared to 50 s which are needed for computation of the trajectory in fig. 10 and fig. 11. The deviations of the optimal travel λ^* trajectory are smaller than in fig. 11, although some oscillation occur which are caused by a too large gain in the $\mathbf{K}$ matrix, a smaller weight on the deviations of the travel λ would reduce these. The oscillations can also be observed in the input u_1 and the pitch p.

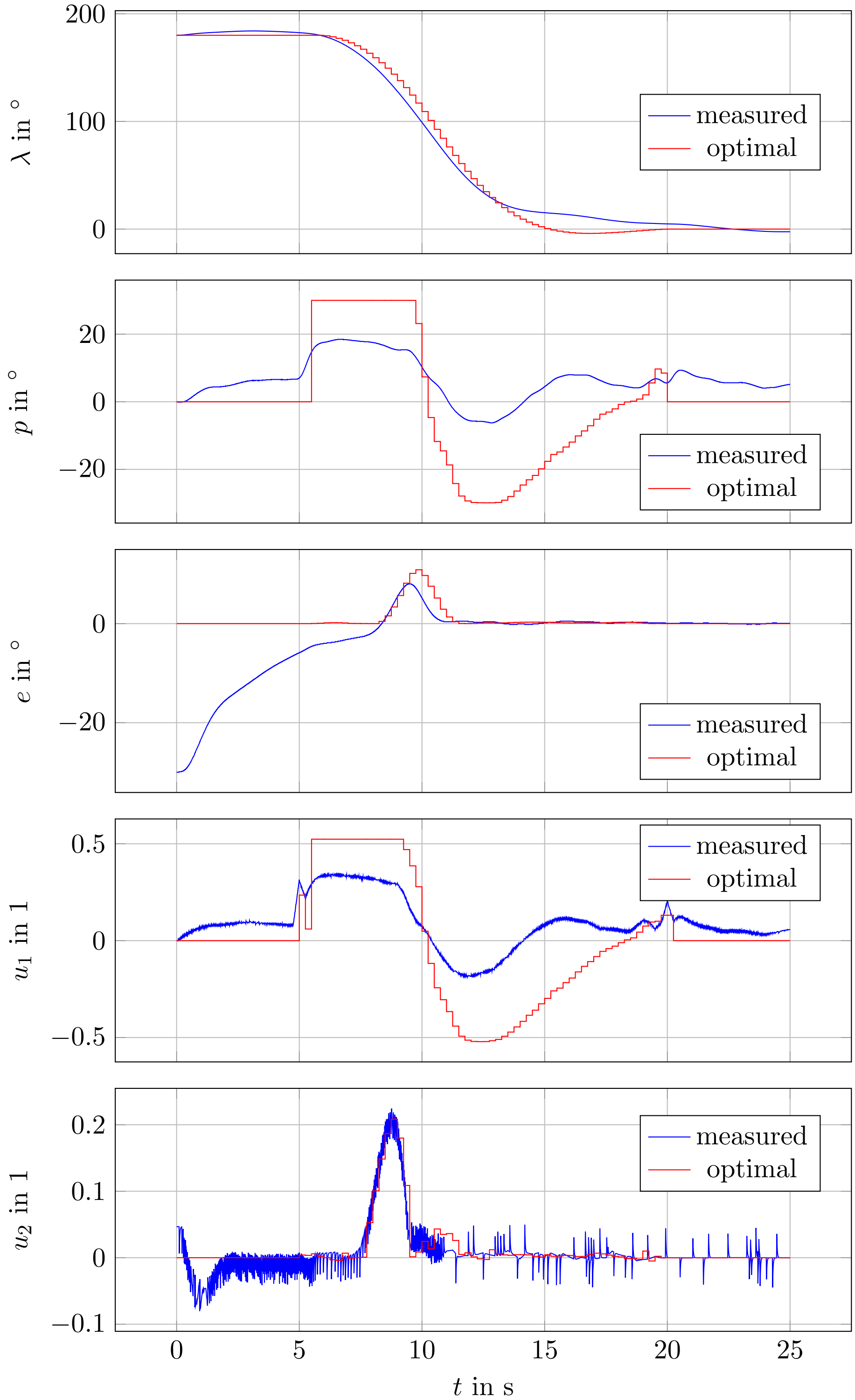

Figure 11: Time curve of inputs and states while using (21) as feedback controller.

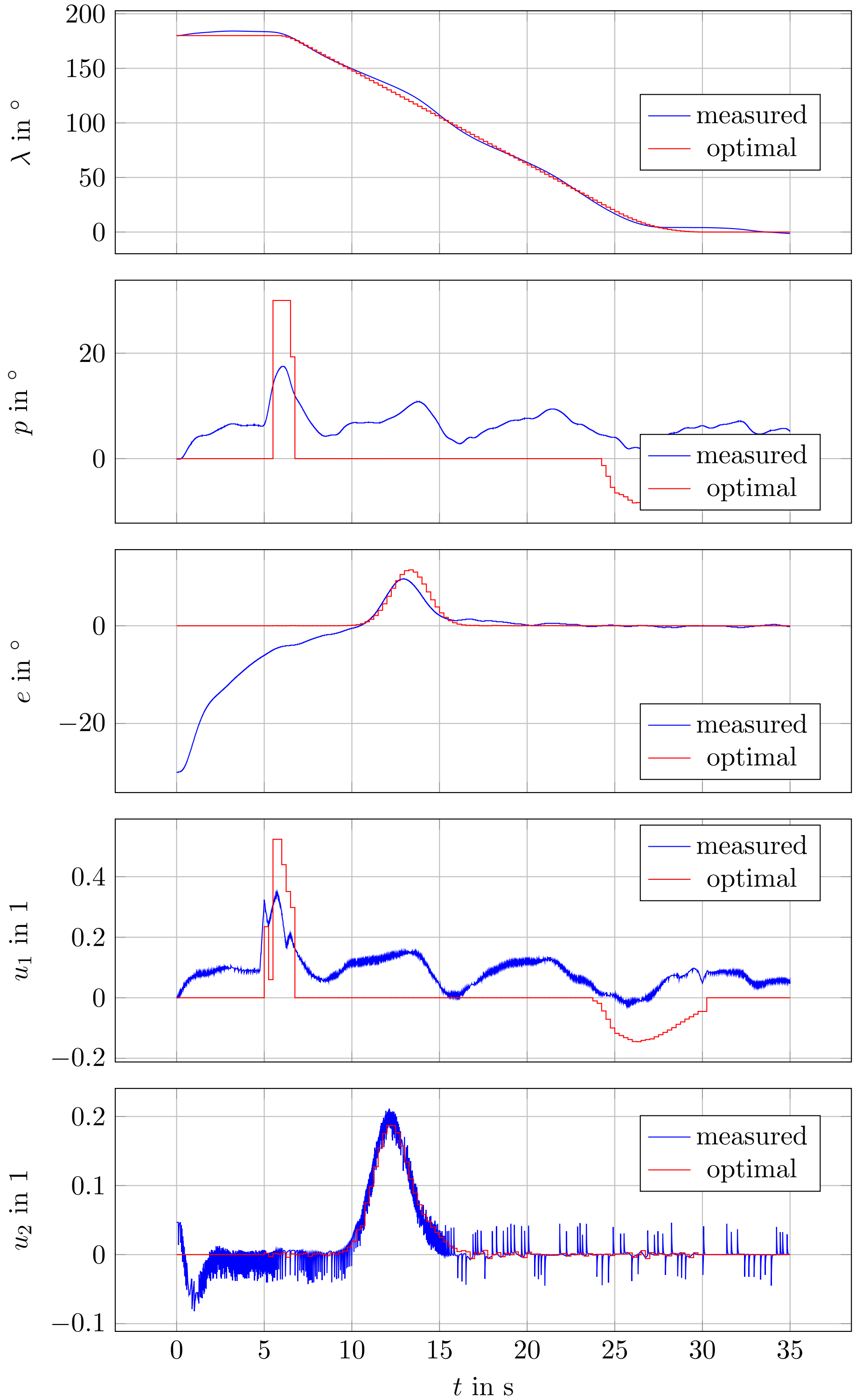

Figure 12: Time curve of inputs and states while using the optimal trajectory with additional constraints (23).

7 Discussion

7.1 Optimal control without feedback

When controlling the helicopter using optimal control without feedback. The Estimated optimal input sequence would yield quite a poor preformance as the actual travel deviates a lot from the optimal travel. It overshoots and doesn't enter a steady state such as the optimal input predicts it should. This is caused by the linearization and simplicity of the model causing modelling and linearization errors to become to large. Optimal inputs such as Pitch and elevation were followed better than with feedback this is of course due to not being altered by the LQR.

The inputs were however limited correctly and does not exceed their constraints. So if you need strict limitations on inputs you would have to go with either open-loop or an MPC.

7.2 Optimal control with feedback

Yielded better preformance than the open-loop. This is due to modelling and lineraization errors being corrected by the LQR layer. However the LQR makes it so that the helicopter isn't following it's optimal input as closly as in the open-loop. However it follows the intended travel path much closer than with the open-loop and it's able to enter a steady state not causing the helicopter to drift away from the set point. We ended up peanlizing deviations from travel/trajectory more than pitch and elivation to ensure it followed the intended path. Adding elvation and not just the pitch and travel to the optimal control input we were able to improve the following of the optimal travel and elevation. However the pitch would then follow its optimal input even less. Also another problem were that we could not guarantee that the final input no longer were guaranteed to be within the constraints set by the optimization problem. This is due to the LQR not being able to limit the inputs used. Therfor adding more constraints did improve a bit on the open-loop preformance but had neglectible impact on the preformance of the closed-loop system. We also discovered using different types of SQP algorithms for calculating the QP had a significant impact on the calculation time and precision of the solution and the active-set algorithm yielded the best preformance with decent accuracy for this problem. While the inter-point algorithm that is default had better accuracy but were painfully slow.

7.3 MPC vs LQR

As more powerful hardware is getting cheaper an NMPC would be more feasible in the future and could potentially yield better preformance than calculating the optimal input trajectory before hand and just sending it in as input using an LQR. Also an NMPC would be able to enforce constraints on the states/inputs of the system better than an LQR with calculated optimal trajectory.

Also an MPC would yield a better preformance than the open-loop optimal control. As you would have implicite feedback by calculating the trajectory from the previous state within the time window of the MPC.

8 Conclusion

Using the modeled helicopter at the lab. We were able to try to control the helicopter with the different approaches stated from before.

When using only an optimal input sequence with no feedback we would get quite an unstable system as seen in the result above. The reason why this did not work so well is because the model we use for the helicopter is quite simplified and does not reflect all the coupling and dynamics of the system being controlled. Resulting in drift from the set point. However all the constraints set were not violated as in the LQR.

However when we used an optimal input sequence combined with control theory using an LQR in a feedback loop our results drastically improved our results. Also this solution did not require a long computation time as, but sacrifices the ability to impose hard constraints on input and states as the LQR does not limit its output.

Therefor if we need constraints on the feedback a solution would be to use an MPC. However it would be a lot more computational intensive and would require better/faster or specialized hardware to work seamlessly. This could be possible in the future and is a part of an emergin field in UAVs as high-preformance microcontrollers are getting cheaper and more energy efficiant. Also external computational systems for UAVs using positioning tools such as GPS and cameras enables computational heavy solutions such as NMPCs to be viable for UAVs. They are of course this solution is less viable for regular aerial vehicles as the hardware has to be onboard.

A MATLAB Code

A.1 problem2.m

```
% ****************************************************
% *                                                  *
% *       Optimization and Control                   *
% *                                                  *
% *       Helikopterlab                              *
% *                                                  *
% * problem2.m                                       *
% *                                                  *
% *                                                  *
% * Updated on 04/2009 by Agus Ismail Hasan          *
% *                                                  *
% ****************************************************

init01;
delta_t   = 0.25;                           % sampling time
sek_forst = 5;

% System model. x=[lambda r p p_dot]'

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

A1 = eye(4) + Ac*delta_t;
B1 = Bc*delta_t;

% Number of states and inputs

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

% Initial values

x1_0 = pi;                                 % Lambda
x2_0 = 0;                                  % r
x3_0 = 0;                                  % p
x4_0 = 0;                                  % p_dot
x0   = [x1_0 x2_0 x3_0 x4_0]';             % Initial values
```

```
xf   = [0 0 0 0]';

% Time horizon and initialization

N  = 100;                      % Time horizon for states
M  = N;                        % Time horizon for inputs
z  = zeros(N*mx+M*mu,1);% Initialize z for the whole horizon
z0 = z;                  % Initial value for optimization

% Bounds

ul      = -30*pi/180;    % Lower bound on control -- u1
uu      = 30*pi/180;     % Upper bound on control -- u1

xl      = -Inf*ones(mx,1); % Lower bound on states
xu      = Inf*ones(mx,1);  % Upper bound on states
xl(3)   = ul;              % Lower bound on state x3
xu(3)   = uu;              % Upper bound on state x3

% Generate constraints on measurements and inputs

vlb      = [kron(ones(N,1),xl);kron(ones(N*mu,1),ul)];
vub      = [kron(ones(N,1),xu);kron(ones(N*mu,1),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;                   % Weight on state x1
%Q1(2,2) = ;                   % Weight on state x2
Q1(3,3) = 0;                   % Weight on state x3
%Q1(4,4) = ;                   % Weight on state x4
P1 = 10;                       % Weight on input
Q = 2*genq2(Q1,P1,N,M,mu);     % Generate Q
c = zeros(N*mx+M*mu,1);        % Generate c

% Generate system matrixes for linear model
Aeq = gena2(A1,B1,N,mx,mu);
Aeq = [ Aeq; [zeros(mx,(N-1)*mx), eye(4), zeros(mx,N*mu)]];

beq = [A1*x0; zeros((N-1)*mx,1);xf];       % Generate b

% Solve Qp problem with linear model
```

```
tic
[z,lambda] = quadprog(Q,[],[],[],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)];     % 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

Antall = 5/delta_t;
Nuller = zeros(Antall,1);
Enere  = ones(Antall,1);

u   = [Nuller; u; Nuller];
x1  = [pi*Enere; x1; Nuller];
x2  = [Nuller; x2; Nuller];
x3  = [Nuller; x3; Nuller];
x4  = [Nuller; x4; Nuller];

%save trajektor1ny

% figure
t = 0:delta_t:delta_t*(length(u)-1);     % real time

figure(2)
subplot(511)
stairs(t,u),grid
ylabel('u')
subplot(512)
plot(t,x1,'m',t,x1,'mo'),grid
ylabel('lambda')
```

```
subplot(513)
plot(t,x2,'m',t,x2','mo'),grid
ylabel('r')
subplot(514)
plot(t,x3,'m',t,x3,'mo'),grid
ylabel('p')
subplot(515)
plot(t,x4,'m',t,x4','mo'),grid
xlabel('tid (s)'),ylabel('pdot')
```

A.2 problem3.m

```
%% Optimal control of Pitch/Travel with Feedback(LQ)
problem2;

% [lambda r p p_dot]
LQR.Q = diag([1, 0, 10, 0]);
LQR.R = 0.5;

K = dlqr(A1, B1, LQR.Q, LQR.R);
```

A.3 problem4.m

```
% ***************************************************
% *                                                 *
% *       Optimization and Control                  *
% *                                                 *
% *       Helikopterlab                             *
% *                                                 *
% * problem4.m                                      *
% *                                                 *
% *                                                 *
% * Updated on 04/2009 by Agus Ismail Hasan         *
% *                                                 *
% ***************************************************

init01;
delta_t    = 0.25;                     % sampling time
sek_forst = 5;

% System model. x=[lambda r p p_dot e e_dot]'
% u = [p_c, e_c]

Ac = [ 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];
Bc = [0 0;
        0 0 ;
        0 0 ;
        K_1*K_pp 0;
        0 0;
        0 K_3*K_ep];

A1 = eye(6) + Ac*delta_t;
B1 = Bc*delta_t;

% Number of states and inputs

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

% Initial values

x1_0 = pi;                                  % Lambda
x2_0 = 0;                                   % r
x3_0 = 0;                                   % p
x4_0 = 0;                                   % p_dot
x5_0 = 0;                                   % e
x6_0 = 0;                                   % e_dot
x0 = [x1_0 x2_0 x3_0 x4_0 x5_0 x6_0]'; %Initial values
xf = [0 0 0 0 0 0]';% final values

% Time horizon and initialization

N  = 60;          % Time horizon for states
M  = N;           % Time horizon for inputs
z  = [x0; kron(ones(N-1,1), [0;0;0;0;0;0.1]); ...
    kron(ones(N,1), [0;0.1])];
z0 = z;           % Initial value for optimization

% Bounds

ul      = [-30*pi/180; -60*pi/180]; % Lower bound on u
uu      = [30*pi/180; 60*pi/180];   % Upper bound on u

xl      = -Inf*ones(mx,1); % Lower bound on states
```

```
xu      = Inf*ones(mx,1);   % Upper bound on states
xl(3)   = ul(1);            % Lower bound on state x3
xu(3)   = uu(1);            % Upper bound on state x3
%xl(2)    = -0.15;
%xu(2)    = 0.15;
%xl(6)    = -0.12;
%xu(6)    = 0.12;

% Generate constraints on measurements and inputs

vlb         = [kron(ones(N,1),xl);kron(ones(N,1),ul)];
vub         = [kron(ones(N,1),xu);kron(ones(N,1),uu)];
vlb(N*mx+M*mu)  = 0; %We want the last input to be zero
vub(N*mx+M*mu)  = 0; %We want the last input to be zero


% Generate the matrix Q and the vector c

Q1 = zeros(mx,mx);
Q1(1,1) = 1;                        % Weight on state x1
%Q1(2,2) = ;                        % Weight on state x2
Q1(3,3) = 0;                        % Weight on state x3
%Q1(4,4) = ;                        % Weight on state x4
P1 = diag([1,2]);                   % Weight on input
Q = genq2(Q1,P1,N,M,mu);            % Generate Q
c = zeros(N*mx+M*mu,1);             % Generate c

% Generate system matrixes for linear model
Aeq = gena2(A1,B1,N,mx,mu);
Aeq = [ Aeq; [zeros(mx,(N-1)*mx), eye(6), ...
    zeros(mx,N*mu)]];

beq = [A1*x0; zeros((N-1)*mx,1); xf];      % Generate b

% Solve nonlinear problem with linear model
phi = @ (x)  (x'*Q*x);
options = optimset('Display','notify', ...
    'Diagnostics','on',...
    'MaxFunEvals',Inf,'MaxIter',Inf);
tic
[z, lambda] = fmincon(phi, z0,[],[],Aeq,beq,vlb,...
    vub,@constr4,options);
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

u1 = [z(N*mx+1:mu:N*mx+M*mu);z(N*mx+M*mu-1)];
u2 = [z(N*mx+2:mu:N*mx+M*mu);z(N*mx+M*mu)];

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

Antall = 5/delta_t;
Nuller = zeros(Antall,1);
Enere  = ones(Antall,1);

u1   = [Nuller; u1; Nuller];
u2   = [Nuller; u2; Nuller];
x1  = [pi*Enere; x1; Nuller];
x2  = [Nuller; x2; Nuller];
x3  = [Nuller; x3; Nuller];
x4  = [Nuller; x4; Nuller];
x5  = [Nuller; x5; Nuller];
x6  = [Nuller; x6; Nuller];
u =  [u1 u2];
%save trajektor1ny
%%
 K = 0;
% figure
t = 0:delta_t:delta_t*(length(u1)-1);      % real time


figure(2)
subplot(511)
```

```
stairs(t,u1),grid
ylabel('u1')
subplot(512)
plot(t,x1,'m',t,x1,'mo'),grid
ylabel('lambda')
subplot(513)
plot(t,x2,'m',t,x2','mo'),grid
ylabel('r')
subplot(514)
plot(t,x3,'m',t,x3,'mo'),grid
ylabel('p')
subplot(515)
plot(t,x4,'m',t,x4','mo'),grid
xlabel('tid (s)'),ylabel('pdot')

figure(3)
subplot(311)
stairs(t,u2),grid
ylabel('u2')
subplot(312)
plot(t,x5,'m',t,x5,'mo'),grid
ylabel('e')
subplot(313)
plot(t,x6,'m',t,x6','mo'),grid
ylabel('e_dot')
```

A.4 constr4.m

```
function [ c, ceq ] = constr4( x )

    N = 60;
    mx = 6;
    alpha = 0.2;
    beta = 20;
    lambda_t=2*pi/3;

    lambda_k = x(1:6:N*mx);
    e_k = x(5:6:N*mx);

    c = alpha*exp(-beta*((lambda_k-lambda_t).^2))-e_k;
    ceq = [];
end
```

A.5 problem4_qr.m

```
% Optimal control of Pitch/Travel and Elevation
% with Feedback(LQ)

problem4;

%%

LQR.Q = diag([5, 1, 1, 1, 1, 1]);
LQR.R = diag([1 1]);

K = dlqr(A1, B1, LQR.Q, LQR.R);
```

B Simulink Diagrams

B.1 Problem 2

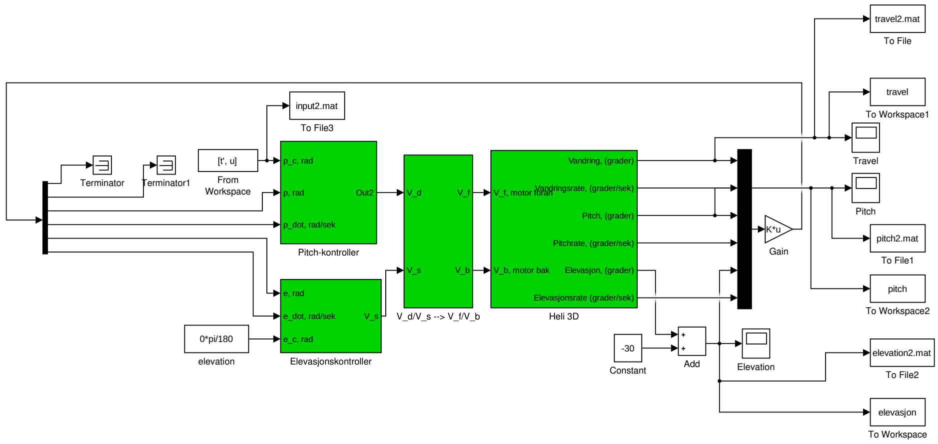

Figure 13: Simulink diagram of problem 2.

B.2 Problem 3

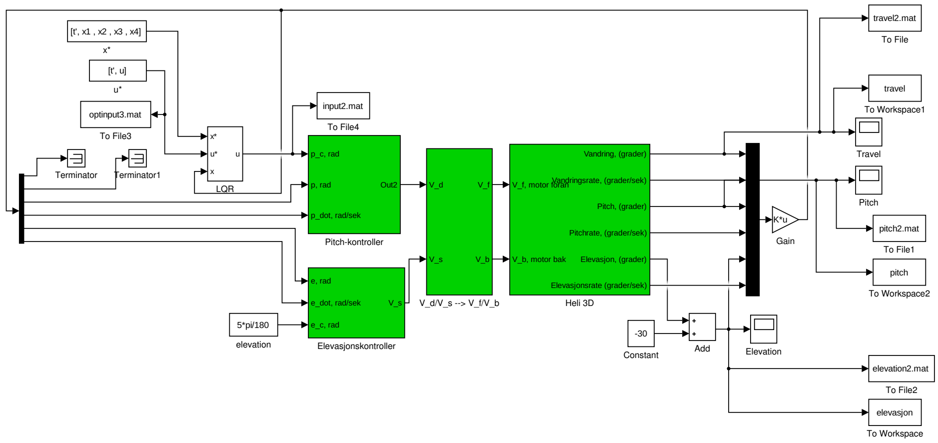

Figure 14: Simulink diagram of problem 3.

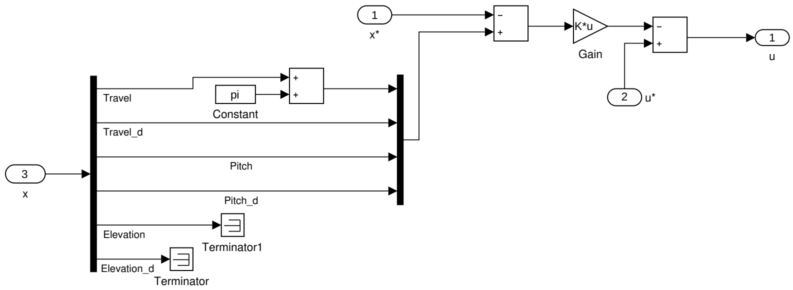

Figure 15: Simulink diagram of LQR subsystem of problem 3.

B.3 Problem 4

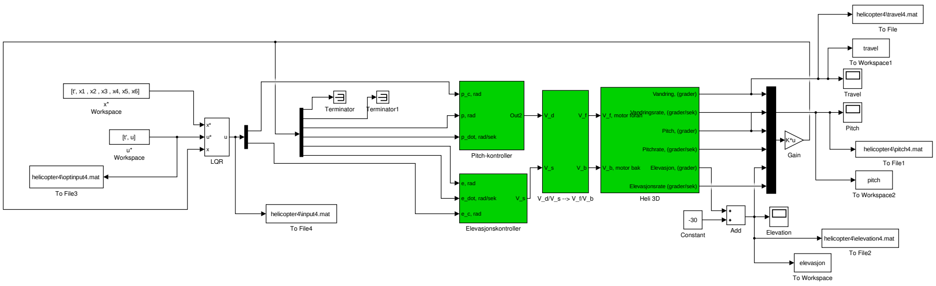

Figure 16: Simulink diagram of problem 4.

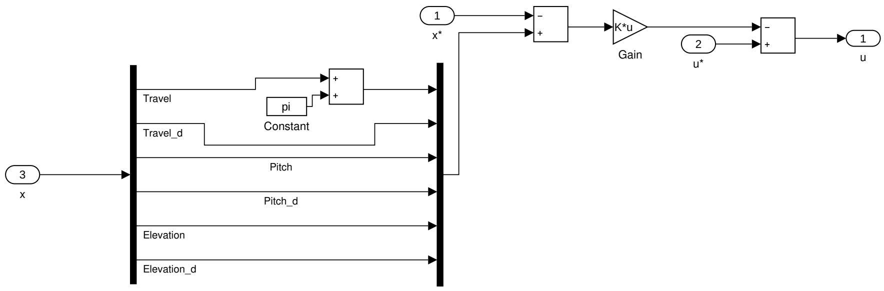

Figure 17: Simulink diagram of LQR subsystem of problem 4.

Bibliography

Department of Engineering Cybernetics, N. (2014). *TTK4135 Optimalization and Control Helicopter Lab.*

Johansen, T. A. (2011). *Introduction to Nonlinear Model Predictive Control and Moving Horizon Estimation.* STU/NTNU.