TTK4135 Optimization and Control
Lab Report

736698, 742388, 742367, 755875 and 731918
Group 17

May 1, 2015

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Abstract

The purpose of the helicopter project in TTK4135 is to get experience combining optimization algorithm and control theory on a real world platform. The helicopter will be controlled by the optimal trajectory / input according to a linearized model, both in open-loop and closed-loop with a linear quadratic regulator. The project also gives an introduction to hardware-in-the-loop (HIL) and automatic generation of code using MatLab with QuaRC.

The main topics of this project is:

- Brief introduction to the model of the helicopter and the assumptions done when linearizing it.
- Discretization of the model
- Optimal control with cost function and linear inequality constraints
- Applying linear quadratic regulator (LQR) to the optimal trajectory
- Optimal control in two dimensions with non-linear inequality constraints

Contents

1 Intro

1.1 Modeling, linearization

On this project we will be using optimization theory to control a model of a helicopter. This is done by: making a non-linear model of the helicopter, and the linearize it at an equilibrium. This method is widely used in control theory and works well if the system operates near or at the equilibrium or have quite small non-linearities. However when this is not the case, a linearized model may not be adequate for stabilizing and controlling the system.

The reason most control engineers prefer linear model is the ability to analyze them and tune regulators. In this project we linearize one time manually around an equilibrium, but there is no limitations on the number of times the model can be linearized and the process can be done automatically by a computer. More about this in the last section.

1.2 Optimal control, limitations without feedback

Optimization theory is a knowledge used in many applications, from economy and production planing to control theory. In this project we will be looking at the last application, and how it can be used to solve control theory problems. The main difference between a conventional regulator who only uses the current state or output and a reference and a trajectory from an optimization over a horizon is the ability to "look into the future".

While a conventional regulator will calculate an error and correct the input thereafter, a planned trajectory from an optimization problem will contain inputs for the complete horizon. However, there can be quite drastic drawbacks with implementing this optimal sequence of inputs directly to the system since there are no feedback, any error caused by modeling errors and disturbances will accumulate and often cause drift and maybe instability.

1.3 Optimal control, trust the trajectory not the input

However, while the optimal sequence of inputs may not be perfect in open-loop, the planned trajectory for the states are often quite useful, even if the model have errors. Here it is possible to implement a traditional PID regulator, or as we will be doing in this project, implement a linear quadratic regulator (LQR).

The LQR can be both time variant and time invariant, dictated by the nature of the system. As mentioned in the linearizion part, the calculation of the LQR gain can and will be done by a computer, and will often be recalculated when re-linearizing. By tuning the LQR so that it prioritize to follow the trajectory of the states rather than following the optimal input, it is possible to compensate for modeling errors.

1.4 Optimal control with non-linear constraints, and thoughts on MPC

When controlling two degrees of freedom with non-linear inequality constraints with optimization theory, the time it takes to solve the optimization problem is quite long. When calculating the trajectory once and then executing the whole horizon this does not become a problem.

However when using MPC, which is to recalculate the horizon every time step and only use the first calculated input, this time becomes a problem. This calculation time can be improved by selecting a better solver, increase hardware specifications and do simplifications / linearizion. When achieving this on a low-power unit like a micro controller, FPGA, DSP or equivalent, a new world in control theory will open.

2 Problem Description

2.1 Hardware setup

The helicopter is fixed to a base and has an extended arm with motors attached at one end, and a balance weight at the other. We parametrize the helicopter state by three rotations:

- Travel (λ): At the base about the vertical axis
- Elevation (e): At the arm joint about a horizontal axis
- Pitch (p): At the head joint about the arm axis

These angles are measured by the encoders at each rotational joint. We can affect the motion by adjusting the voltage input to the DC motors. Propellers are attached to the motors, and the thrust generated is assumed to be proportional to the voltage applied, with identical motor constant for both motors. Figure (1) shows the helicopter and the relevant angles. Table (1) shows the parameters for the helicopter used in the report.

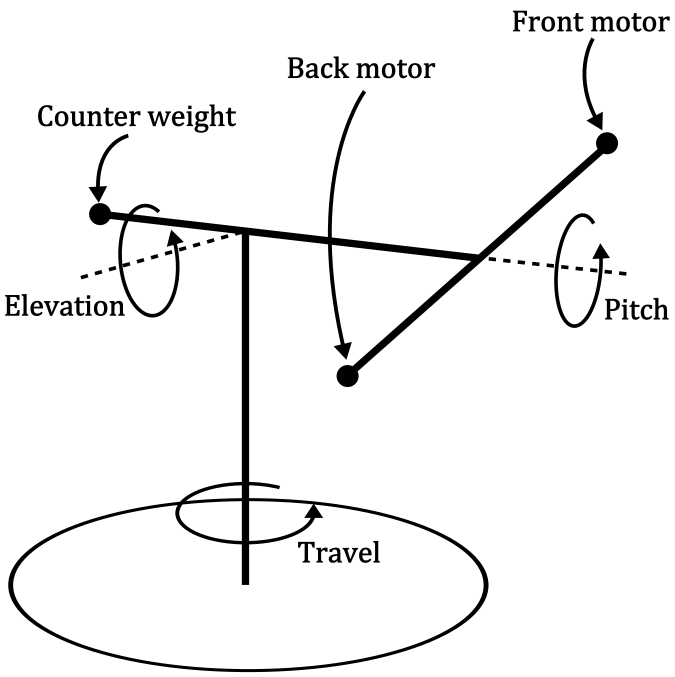

Figure 1: The helicopter hardware setup

2.2 Software setup

We use Simulink and Matlab to implement the control program, and interface with the helicopter through QuaRC and Realtime Workshop. See [Jesmani (2015)] for more information.

2.3 Model discussion

In order to compute an optimal control input, we need some measure of optimality. To this extent, we use a discrete-time state-space model, that describes the motion of our helicopter given an input sequence. The first step is to derive the equations of motions in continuous-time. We do this by considering torque balances about each rotation axis, and applying Newton's law. A full derivation is not interesting, so we refer to the exercise text [Jesmani (2015)] and list the resulting set of differential equations below.

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

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

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

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

The associated nomenclature is listed in table (2). This model deserves some discussion. First, a complete model would have (non-linear) trigonometric terms. But such a model is impractical for use in dynamic optimization schemes. Instead, we linearize the model by assuming that the pitch and elevation angles are both close to zero.

The resulting model has the advantage of being compatible with highly efficient techniques that have been developed specifically for linear optimization problems. It is however a very simple model, and omits any interaction between elevation and pitch angle/travel. While there will always be modelling errors due to process noise, or inaccurate measurements/parameters, this simplification will affect the computed optimal control. We can dampen the effect from the error by keeping the system close to the linearization point, for instance by constraining the angles to be within a margin around equilibrium.

Second, the model incorporates a PID controller for the elevation, which is assumed to counteract the constant torque due to gravity, as well as a PD controller for the pitch. The result of this is that our goal in the optimization problem is to compute the setpoints, e_c and p_c. Computing the appropriate voltages is left to the internal controllers.

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	0.20	N/V
J_e	Moment of inertia for elevation	1.11	kg m^2
J_t	Moment of inertia for travel	1.11	kg m^2
J_p	Moment of inertia for pitch	0.045	kg m^2
m_h	Mass of helicopter	1.42	kg
m_w	Balance weight	1.80	kg
m_g	Effective mass of the helicopter	0.025	kg
K_p	Force to lift the helicopter from the ground	0.25	N

Table 2: Variables

Symbol	Variable
p	Pitch
p_c	Setpoint for pitch
λ	Travel
r	Speed of travel
r_c	Setpoint for speed of travel
e	Elevation
e_c	Setpoint for elevation
V_f	Voltage, motor in front
V_b	Voltage, motor in back
V_d	Voltage difference, $V_f - V_b$
V_s	Voltage sum, $V_f + V_b$
$K_{pp}, K_{pd}, K_{ep}, K_{ei}, K_{ed}$	Controller gains
T_g	Moment needed to keep the helicopter flying

TODO: Variable notation for plots and stuff.

3 Repetition/Introduction to Simulink/QuaRC

This part is just a repetition, and therefore does not contain any important results. However, the part is important because it is used to test the hardware setup at the lab. This was done using a handed out script that made the helicopter hover at zero elevation and zero pitch. The helicopter suffered from minor drifting as a result of not perfectly tuned motor constants in the handed out script. This was expected and did not introduce a problem since the drifting would be eliminated as soon as some form of feedback was implemented.

4 Optimal Control of Pitch/Travel without Feedback

In this section an optimal trajectory x^* and a corresponding input sequence u^* was calculated. The trajectory should bring the helicopter from an initial state to another predefined state. No feedback was used to correct for deviations from the calculated trajectory. Also, elevation was disregarded, that is, $e = 0$ was assumed for this part. The computation of the optimal trajectory was formulated as a discrete convex optimization problem.

4.1 The helicopter model on continuous-time state-space form

The system (1) describes the helicopter plant, with a basic control layer consisting of PID and PD controllers for elevation and pitch. The optimization layer gives the inputs to these regulators, as shown in figure (2).

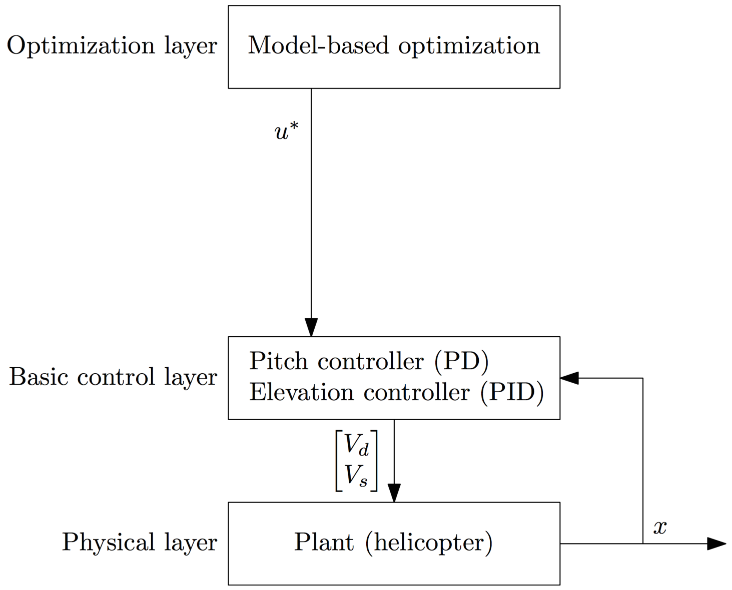

Figure 2: Control hierarchy

The system can be written on continuous-time state-space form:

$$\dot{x} = A_c x + B_c u \tag{2}$$

with $x = \begin{bmatrix} \lambda & r & p & \dot{p} \end{bmatrix}^T$ and $u = p_c$. The continuous-time system matrices for this model are:

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -K_2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -K_1 K_{pp} & -K_1 K_{pd} \end{bmatrix} \quad \text{and} \quad B_c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ K_1 K_{pp} \end{bmatrix} \tag{3}$$

4.2 Discretization

The system dynamics was implemented as a sequence of constraints in the optimization problem, and the system model therefore had to be written in discrete-time state-space form,

$$x_{k+1} = A x_k + B u_k. \tag{4}$$

The model was discretized using forward-Euler with timestep h.

$$\dot{x}_k \approx \frac{x_{k+1} - x_k}{h} \tag{5}$$

Inserting this into (2),

$$\dot{x}_{k+1} \approx (I + hA_c)x_k + hB_c u_k \tag{6}$$

is obtained. A suitable approximation for the discrete-time matrices is then

$$A \approx I + hA_c \qquad \text{and} \qquad B \approx hB_c \tag{7}$$

These matrices were computed in Matlab, and are therefore not shown explicitly here.

4.3 Computation of optimal trajectory

An optimal trajectory can be generated by minimizing the cost function

$$\phi = \sum_{i=1}^{N} (\lambda_i - \lambda_f)^2 + q p_{ci}^2 \tag{8}$$

for some scalar weight $q \geq 0$ over the finite horizon of states and inputs

$$z = (x_1 \ x_2 \ \dots \ x_N \ u_1 \ u_2 \ \dots \ u_N)^T \tag{9}$$

This was done in Matlab using the function quadprog. The discrete system dynamics was implemented as equality constraints of the form $A_{\text{eq}} z = B_{\text{eq}}$, where A_{eq} and B_{eq} are given by the left- and right-hand side of the N constraints

$$\begin{aligned} x_1 - Bu_0 &= Ax_0 \\ x_2 - Ax_1 - Bu_1 &= 0 \\ &\vdots \\ x_N - Ax_{N-1} - Bu_{N-1} &= 0 \end{aligned}$$

We would also like to constrain the system state and input to be within a range

$$x^{\min} \leq x_{t+1} \leq x^{\max} \tag{10}$$

$$u^{\min} \leq u_t \leq u^{\max} \tag{11}$$

for $t = 0 \dots N-1$. Applying these constraints to all states and inputs in the solution horizon, we have

$$\begin{bmatrix} I \\ -I \end{bmatrix} z \leq \begin{bmatrix} \{x_{t+1}^{max}\} \\ \{u_t^{max}\} \\ \{x_{t+1}^{min}\} \\ \{u_t^{min}\} \end{bmatrix}_{t=0..N-1} \tag{12}$$

,

which can be implemented as an inequality constraint of the form $A_{\text{iq}} z \leq B_{\text{iq}}$. Solving the optimization problem with different weights q did not lead to any significant differences in the trajectory because the model inaccuracies made the helicopter drift away from the desired state anyway.

The objective function (8) weights the input relative to the state deviations using the weight parameter q. The term $(\lambda_i - \lambda_f)^2$ is the state deviations. They are squared to prioritize the largest deviation. Note that the cost function (8) does not take into consideration that λ_i plus some multiple of 2π describes the same physical orientation of the helicopter. For example, if the reference is 0 and $\lambda_i = 2\pi$, it will be regarded as a large error, even though the helicopter is in fact in the desired orientation. A more optimal scheme would take this into consideration.

4.4 Implementation

Figure (3) shows the implementation with $q = 1$. Zeroes was added on both sides of the input sequence to give time to initialize and stabilize the helicopter. As can be seen from the figure, the helicopter does not reach its desired final state.

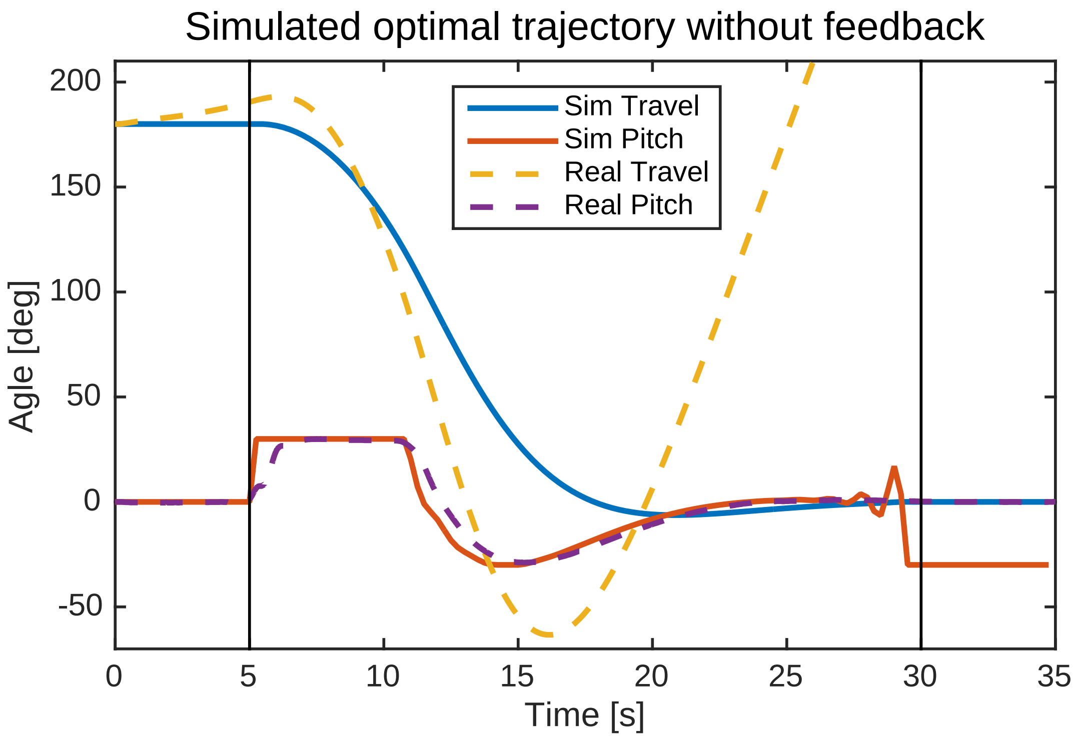

Figure 3: Plot of day 2

The deviation was caused by an imperfect model. A perfect model is unrealistic to construct, and without feedback, the model inaccuracies will lead to nonoptimal results in the real world. For example, the pitch-regulator in our model is fast enough to follow its input perfectly, and we are using a linear model that clearly does not correspond perfectly to the real helicopter.

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

This problem involves implementing an LQ controller for optimal control with feedback. We will calculate a gain matrix K using the LQ controller, implement feedback on the helicopter, and look into if MPC is a good alternative to LQ.

5.1 Calculating the gain matrix K

For calculating the gain matrix K we needed to use the matlab function dlqr, which is a linear-quadratic regulator design that minimiizes the cost function:

$$J = \sum x'Qx + u'Ru + 2 * x'Nu \tag{13}$$

. This function depend on the system matrices A and B aswell as Q and R, which we will need to choose an appropriate weight on. At first we choose Q and R as:

$$\mathrm{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 1 \end{bmatrix} \tag{14}$$

By trial and error we got the best result by penalizing the state for travel a lot more than the pitch. This gave us a new Q matrix on the form:

$$\mathrm{Q} = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

.

5.2 Implementing feedback on the helicopter

An implementation of feedback can be seen on the simulink diagram.

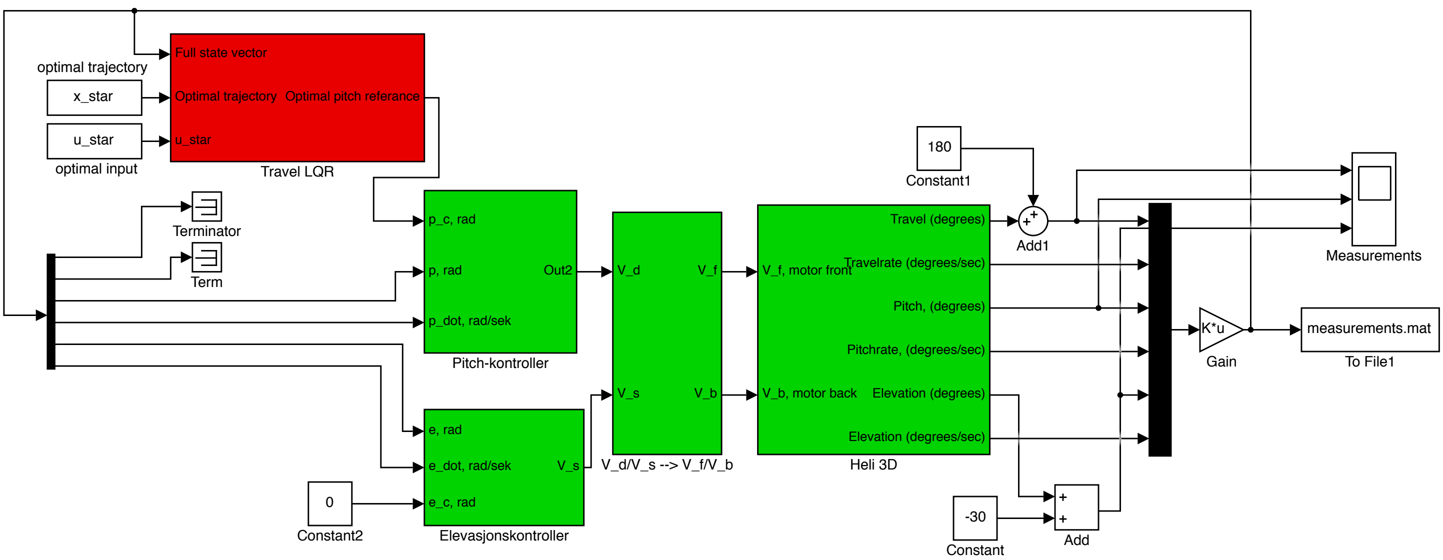

Figure 4: Simulink model with feedback

By using the gain matrix K calculated in last task, we see that by penalizing travel hard and pitch soft, the helicopter follows the travel trajectory closer than when weighting all states the same.

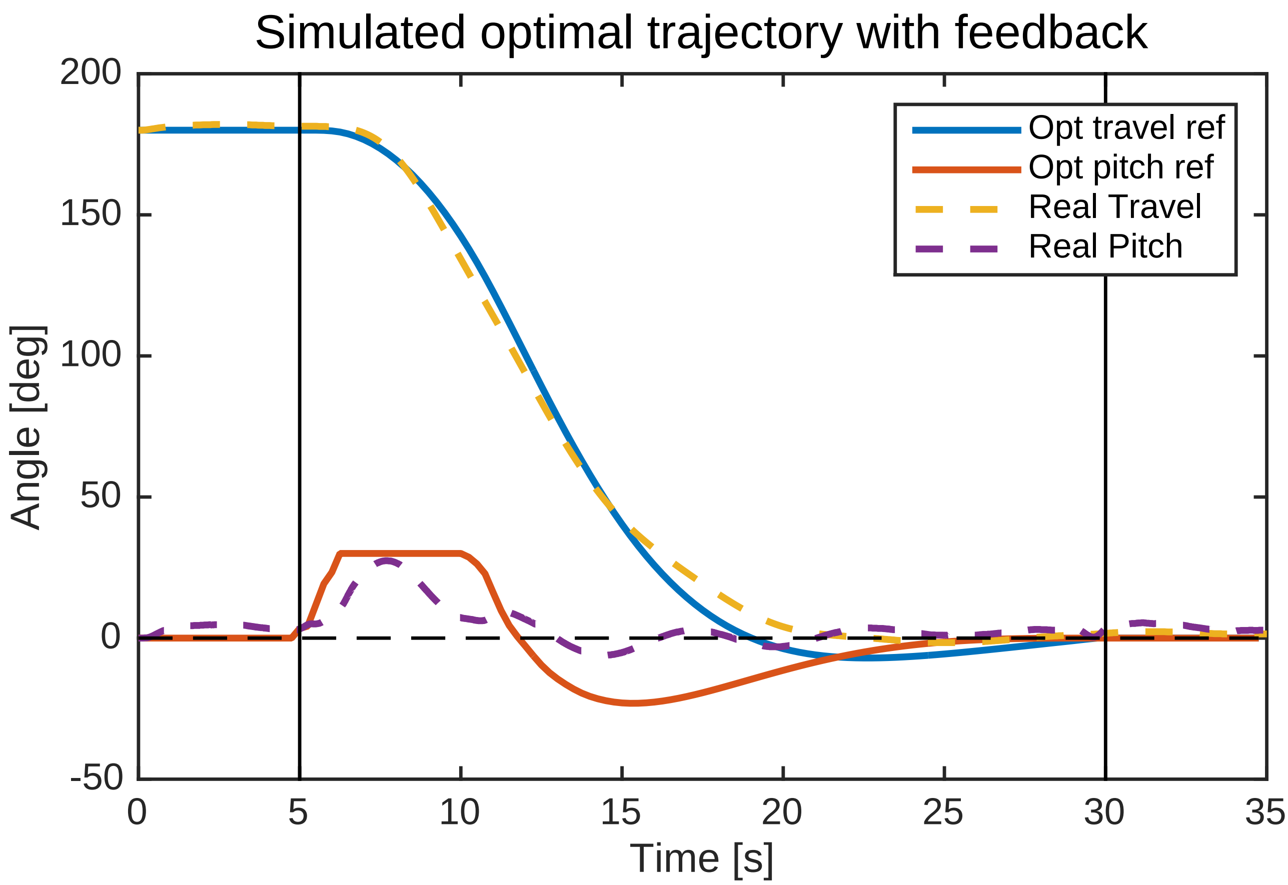

Figure 5: Plot of day 3 with Q = diag(50 1 1 1)

We also tried penalizing the pitch more than travel, which gave us a bad result. Figure 6 shows how the real travel trajectory did not follow the optimal. This is because the optimal pitch reference trajectory is based on a bad model.

5.3 Comparison between LQR and MPC

The way to implement an MPC controller would be to use the same procedure as for the LQR, but for every time step. Then using the first time step as the input u.

The advantages of using MPC instead of LQR; is that it gives us the possibility to have constraints in the regulator, can potentially produce a trajectory which performs its' task more cheaper and we get a implicit feedback with the use of MPC. The main disadvantage of using MPC is that the calculations are a lot heavier to perform.

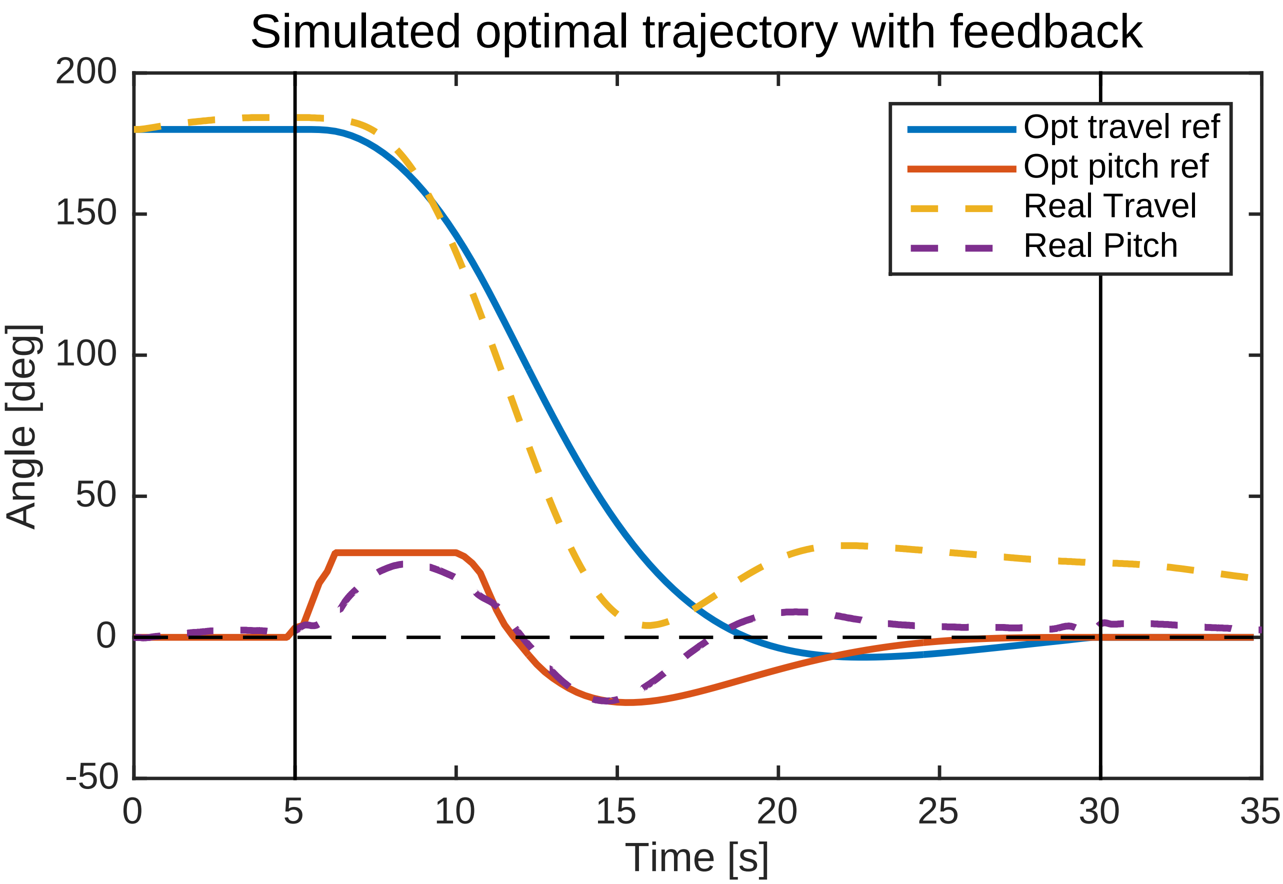

Figure 6: Plot of day 3 with Q = diag(1 1 10 1)

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

In this section we extended our model to include the remaining states: elevation, e, and elevation rate, $\dot{e}$. We used a non-linear solver to compute an optimal trajectory in two dimensions, and additionally constrained the elevation to avoid a restriction shaped as a bell-curve.

6.1 State-space formulation

The state-vector was extended with the remaining states,

$$x = \begin{bmatrix} \lambda & r & p & \dot{p} & e & \dot{e} \end{bmatrix}^T \tag{16}$$

and the elevation setpoint was added to the input-vector,

$$u = \begin{bmatrix} p_c & e_c \end{bmatrix}^T \tag{17}$$

The system is on the usual state-space form (2), with

$$\dot{x} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -K_1K_{pp} & -K_1K_{pd} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -K_3K_{ep} & -K_3K_{ed} \end{bmatrix}}_{A_c} x + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ K_1K_{pp} & 0 \\ 0 & 0 \\ 0 & K_3K_{ep} \end{bmatrix}}_{B_c} u \tag{18}$$

6.2 Discretization

We discretized (18) using the same method as in section (4). That is, an approximation of the discrete-time state-space matrices was

$$A \approx I + hA_c \qquad \text{and} \qquad B \approx hB_c \tag{19}$$

where I was the 6×6 identity matrix.

6.3 Modelling the restriction

A common application of optimal control is to implement restrictions, such as avoiding physical objects, as constraints in the optimization problem. Such restrictions can not be enforced when using only state-feedback controllers.

We wished to restrict the helicopter head to move above the bell-shaped curve

$$e_k \geq \alpha \exp(-\beta(\lambda_k - \lambda_t)^2) \tag{20}$$

for all timesteps k of the solution horizon. Since this is a non-linear constraint, we could no longer use a QP solver. Instead, a non-linear solver was needed, and in this case, the MATLAB function fmincon was used with a SQP-type algorithm.

6.4 Objective function

For this assignment, the cost function is the same as in equation (8), but with an extra term for penalizing elevation. We chose to set up the cost function on a more general form:

$$\phi = \sum (x^T Q x + u^T R u) \tag{21}$$

Here, Q and R are diagonal matrices, with unity weight on q_{travel}, r_{pitch} and $r_{elevation}$, so that it corresponds with the extended state and input vectors (16).

6.5 Results

For controlling the helicopter, we ended on a horizon of $N = 60$ timesteps, or 15 seconds. This gave an optimal trajectory that looked much the same as for assignment 3 for pitch and travel. In addition, an elevator trajectory just touching the top of the bell curve constraint was also calculated. The suggested horizon of $N = 40$ did not terminate in reasonable time, and was therefore replaced by a longer horizon giving reasonable results.

The solution without feedback acted in the same manner as it did in section (4). This can be seen in figure (7), the real trajectory of travel and elevation tried to follow the optimal solution, but the effect of not having a perfect model became more and more clear as time passed, and at the end of the horizon, the travel drifted away from the solution just like in section (4).

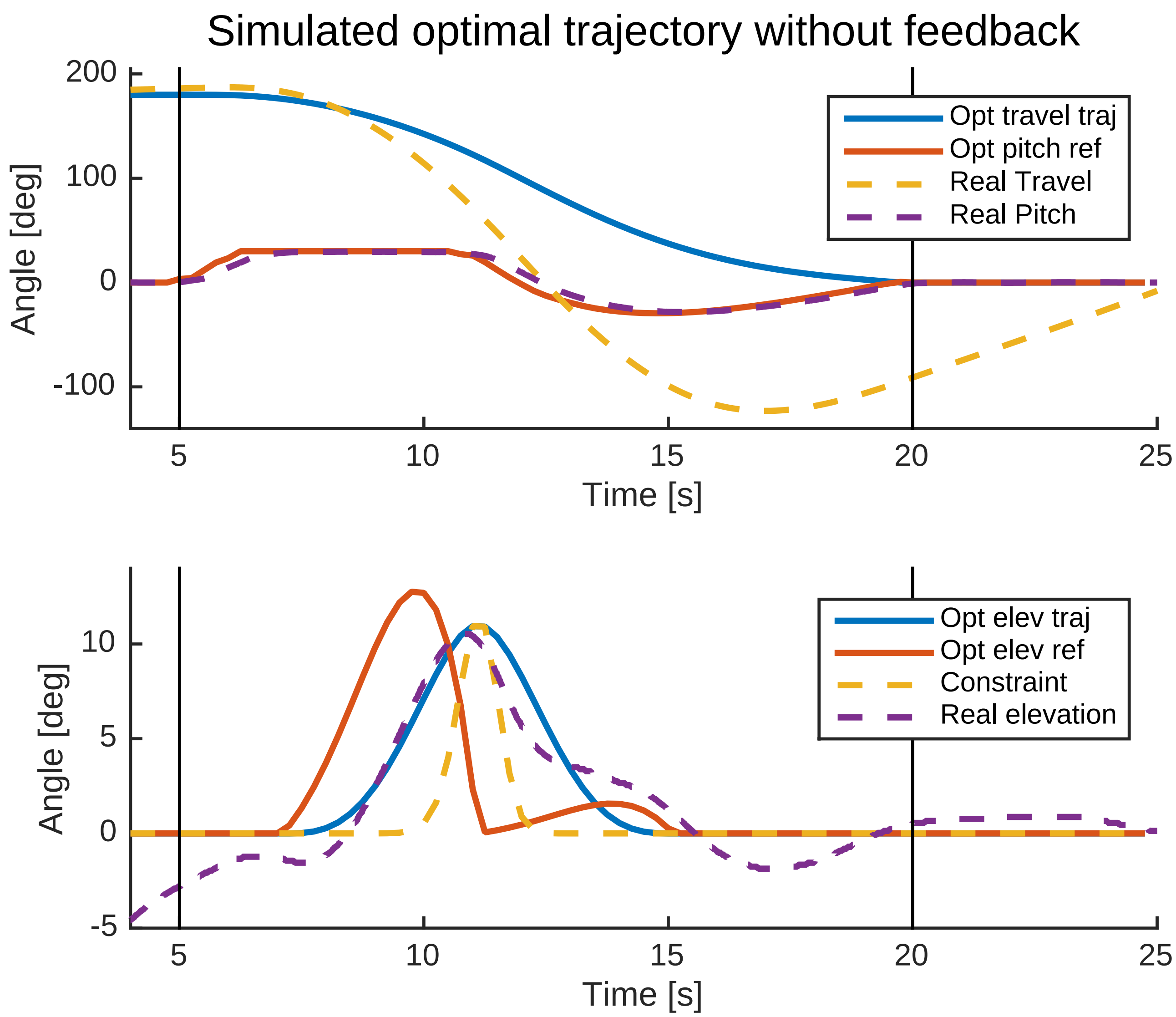

Figure 7: Problem 4 open loop response

Using state feedback LQR, the helicopter managed to follow the calculated trajectory quite good after some tuning. As in section (5), it was appropriate to weight deviation(s) in travel (and here also elevation) more than deviation in pitch. When the regulator is working on an inaccurate model, it is for the task given more constructive to try to follow the travel and elevation trajectories rather than the ideal, linearized input trajectory. This input will, given the unlinearities of the model, not lead to an appropriate response for the trajectories. The result, with different weights on the states of Q_{LQR} can be seen in the following figures.

One thing worth noticing is that the trajectory of elevation consequently fell below the constraint at the peak of the constraint. We believe this is due to the fact that our model suggests that there is no link between the model for elevation, and that of pitch. They are in fact linked, and because of this, the regulator doesn't manage to meet both demands perfectly at the same time, and we get a deviation. If the model would include these crossing terms, the result would probably be better. The problem with this approach is that it leads to a non linear objective function, which takes longer time to compute for the solver.

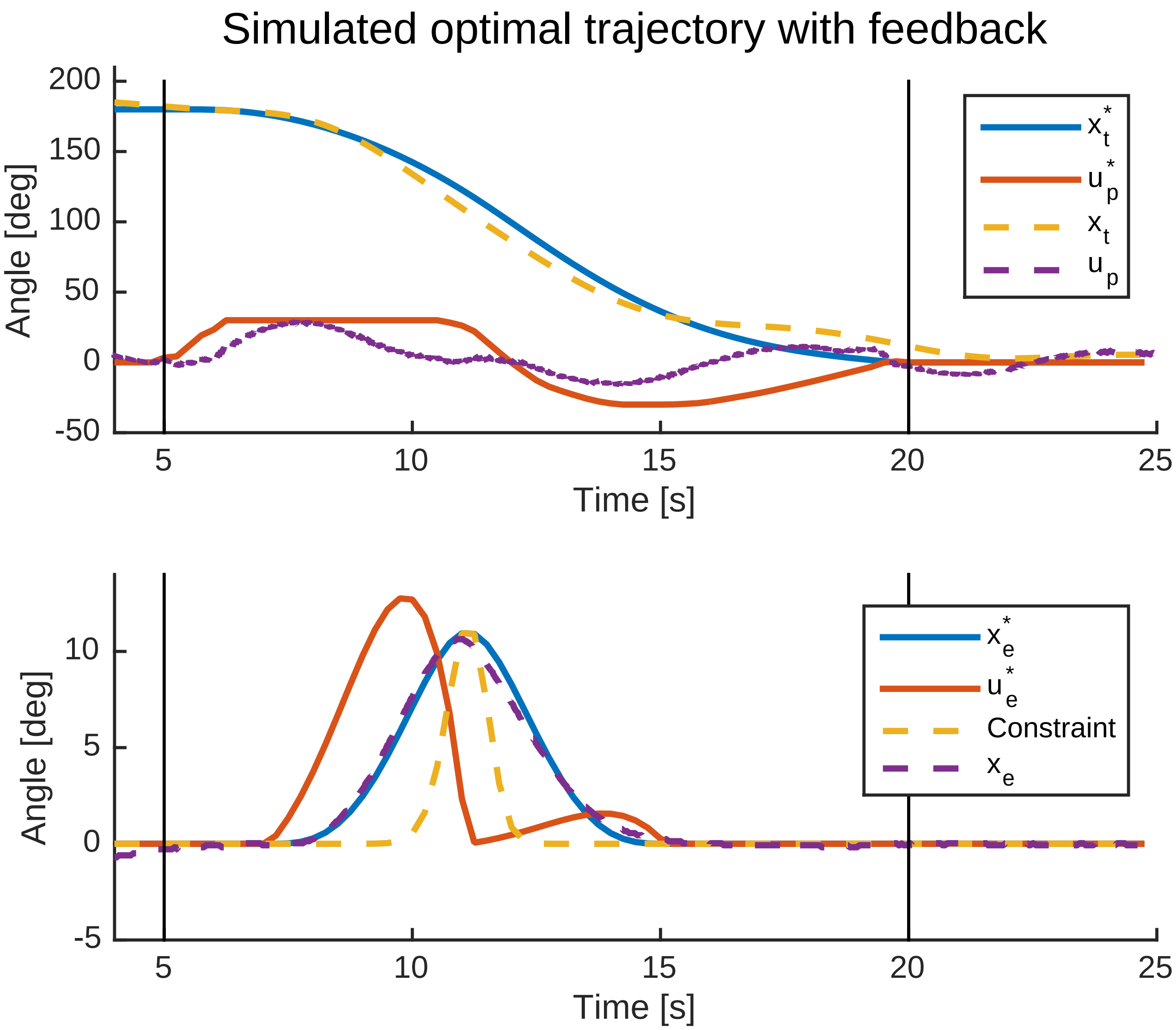

Figure 8: Plot day 4 closed loop Q=diag(20 1 1 1 30 10)

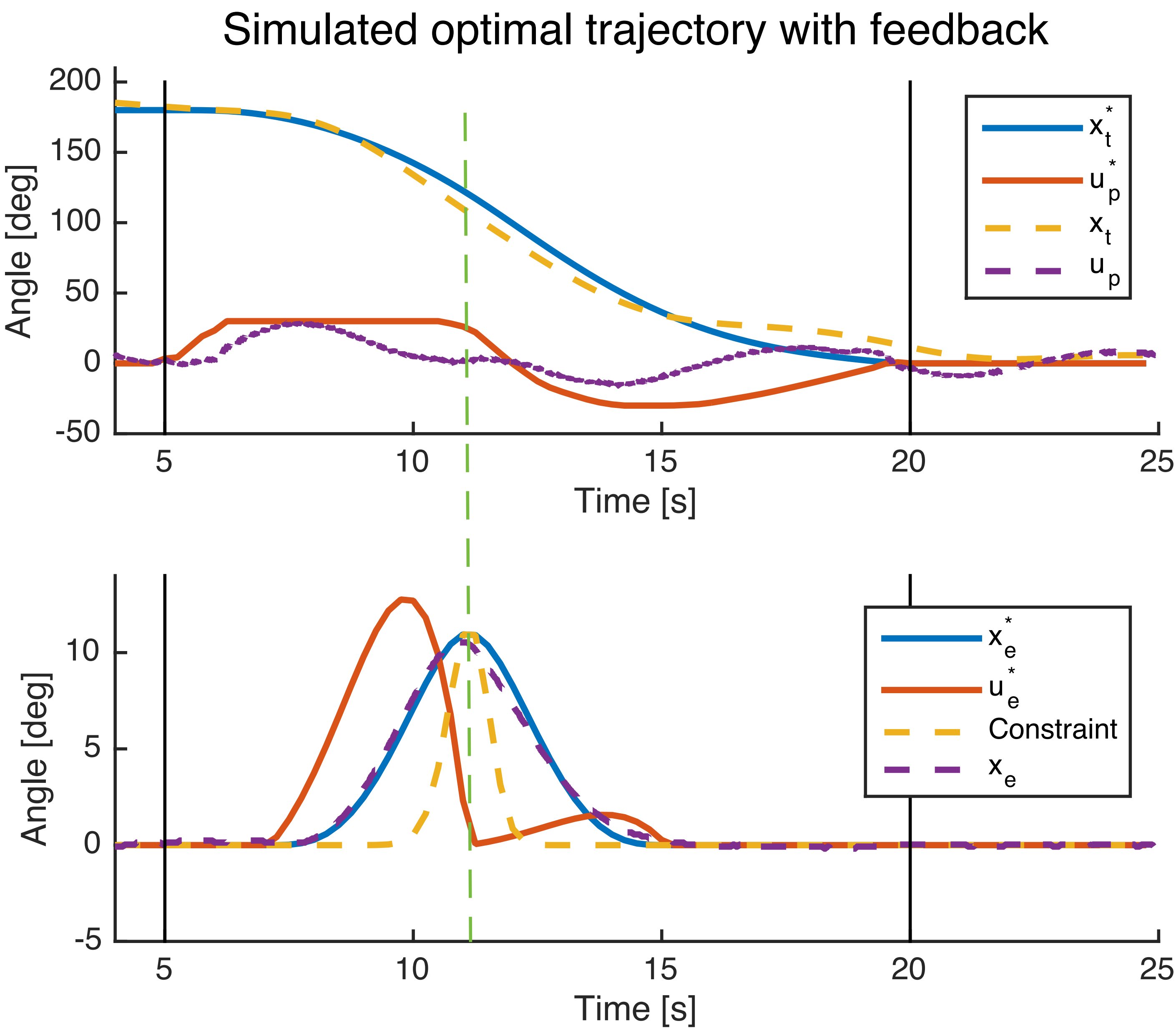

Figure 9: Coupling between pitch and elevation

7 Discussion

7.1 Optimal control without feedback

When controlling the helicopter with only the estimated optimal input sequence, the linearization and modeling errors becomes to large, and the helicopters behavior will deviate drastically from the estimated response. However, there is one benefit from using optimal control without feedback; the ability to limit inputs. So if the system need optimal control with limits on states and inputs, and also need feedback, MPC is the way to go. See section under.

7.2 Optimal control with feedback

While the open-loop configuration of optimal control was quite poor, the closed-loop configuration with a LQR was quite promising. Our testing showed that the optimal state trajectory was much more useful than the optimal input sequence. With a LQR this could be achieved by weighting the error in travel and pitch quite much. This way the regulator acts like a compensator on modeling errors since its additive with the optimal input sequence.

Another effect of including feedback on an optimal trajectory is that the final input is no longer guaranteed to be within the constraints of the optimization problem.

7.3 MPC with implicit feedback

As mentioned in the first section, optimal open-loop configuration performs quite poor caused by the lack of feedback. This is where MPC makes an entry, with its re-optimization every time step with the previous measured / estimated state as initial conditions. Since this is done every time step, only the first step of the input sequence is used. This way we get both an implicit feedback and the possibilities to set constraints on states and inputs.

8 Conclusion

TODO: Se template

A MATLAB Code

TODO: Se template

A.1 plot_constraint.m

```
% Plot a figure with some Latex in the labels
l = linspace(70,170)*pi/180;
a = 0.2;
b = 20;
l_b = 2*pi/3;

e = a*exp(-b*(l-l_b).^2);

l_deg = l*180/pi;
e_deg = e*180/pi;

figure(1)
plot(l_deg,e_deg, 'LineWidth', 2)

handles(1) = xlabel('$\lambda$/degrees');
handles(2) = ylabel('$e$/degrees');
set(handles, 'Interpreter', 'Latex');
```

B Simulink Diagrams

This section should contain your Simulink diagrams.

B.1 A Simulink Diagram

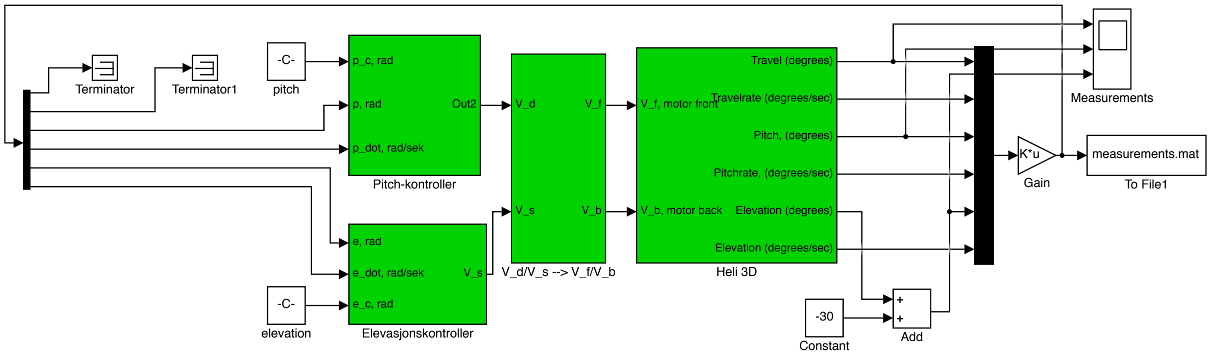

Figure 10: Simulink optimal open-loop model

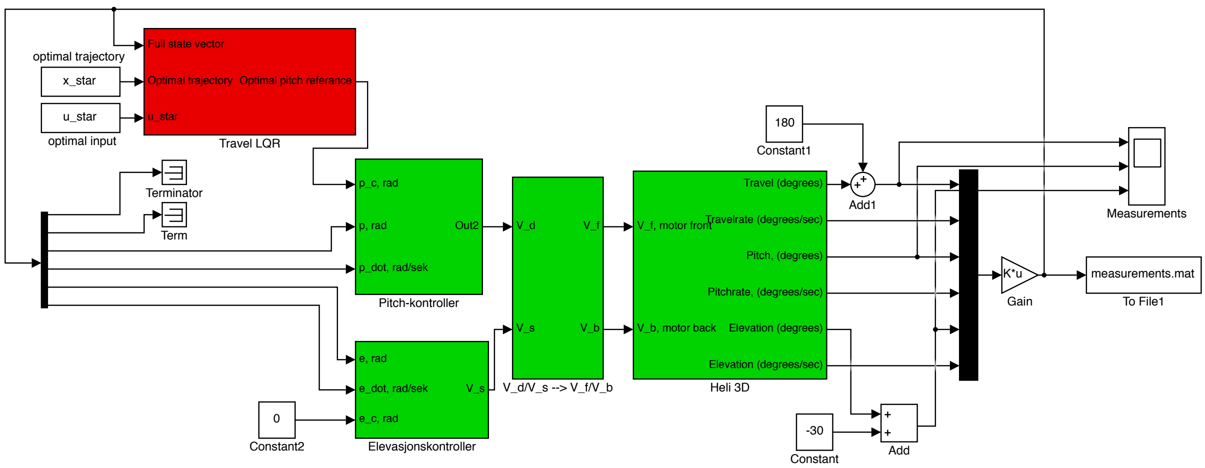

Figure 11: Simulink optimal closed-loop model

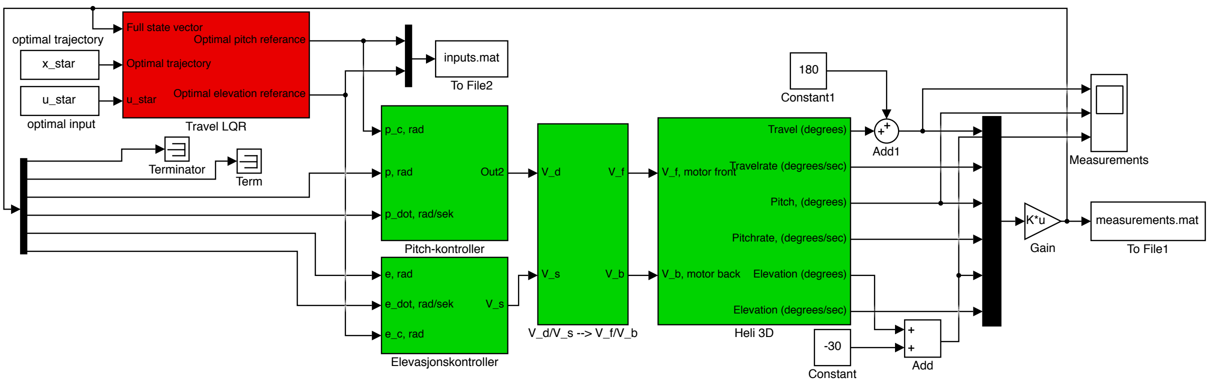

Figure 12: Simulink optimal closed-loop model in two dimensions

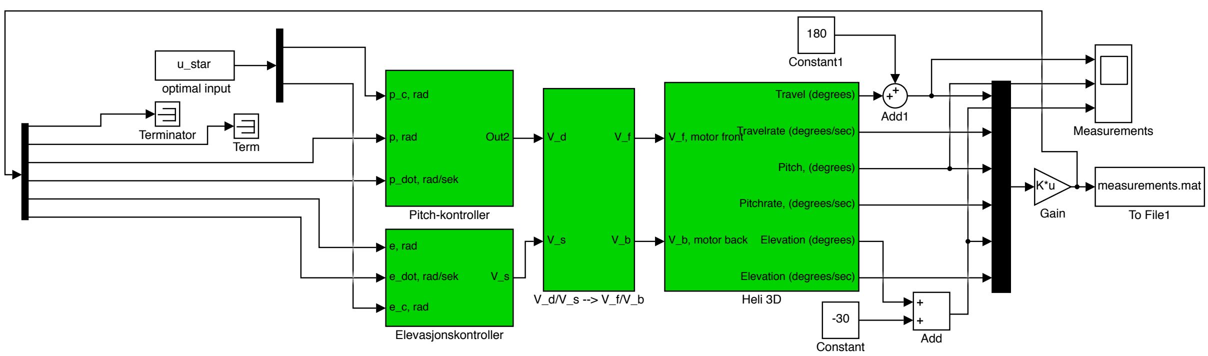

Figure 13: Simulink optimal open-loop model in two dimensions

Bibliography

Jesmani, M. (2015). Ttk4135 optimization and control, helicopter lab.