CACSD Exercise session 3

Case study: Quadcopter

Delphine Vandamme

January 17, 2014

Contents

1 Introduction **2**

2 Linearization **2**

3 Discretization **4**

4 LQR control **6**

4.1 Experiments without payload 6

4.1.1 Simulation results 8

4.2 Experiments with payload 11

4.2.1 Simulation results 11

5 LQG control **15**

5.1 Experiments with and without payload 16

6 Conclusion **19**

1 Introduction

The aim of this report is the design of an LQR and LQG controller that manages the position and attitude of a quadcopter. Specifically, the quadcopter should reach several checkpoints as fast as possible, within a room that is 6 meters wide, 6 meters long and 3 meters high. Furthermore, the quadcopter should be able to do this when carrying a payload of 0.1kg.

The motion of the quadcopter can be described by a set of mechanical and electrical equations. These can be written in the following nonlinear state space form:

$$\begin{aligned} \dot{\mathbf{x}} &= \boldsymbol{\xi}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{C}\mathbf{x}, \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{12} = [x\ y\ z\ v_x\ v_y\ v_z\ \phi\ \theta\ \psi\ \omega_x\ \omega_y\ \omega_z]^T$ is the state vector of the system, $\mathbf{u} \in \mathbb{R}^4 = [v_1^2\ v_2^2\ v_3^2\ v_4^2]^T$ is the input vector in terms of the squared voltages of the rotors, $\mathbf{y} \in \mathbb{R}^6 = [x\, y\, z\, \phi\, \theta\, \psi]^T$ is the output vector and $\boldsymbol{\xi}$ is a nonlinear vector function describing the equations.

2 Linearization

First the nonlinear model 1 is linearized around an operating point as follows:

$$\begin{aligned} \boldsymbol{\Delta}\dot{\mathbf{x}} &= \mathbf{A}\boldsymbol{\Delta}\mathbf{x} + \mathbf{B}\boldsymbol{\Delta}\mathbf{u} \\ \boldsymbol{\Delta}\mathbf{y} &= \mathbf{C}\boldsymbol{\Delta}\mathbf{x}, \end{aligned} \tag{2}$$

where $\boldsymbol{\Delta}\mathbf{x} = \mathbf{x} - \mathbf{x}^*$, $\boldsymbol{\Delta}\mathbf{u} = \mathbf{u} - \mathbf{u}^*$ and $\boldsymbol{\Delta}\mathbf{y} = \mathbf{y} - \mathbf{y}^*$ are the deviation variables and $\mathbf{x}^*$, $\mathbf{u}^*$ and $\mathbf{y}^*$ are the values of the operating point. The chosen operating point is the point of stationary flight, where the quadcopter is in hover. This is also an equilibrium point.

$$\begin{aligned} \mathbf{x}^* &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{u}^* &= \begin{bmatrix} v^2 & v^2 & v^2 & v^2 \end{bmatrix} \\ \mathbf{y}^* &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

where v^2 is the squared voltage of each rotor when the quadcopter is in hover. We have that

$$v^2 = \frac{\omega^2}{c_m},$$

where ω is the angular velocity of each motor and $c_m = 10^4$ is a constant. Note that the voltages of the rotors are the same because when the quadcopter is hovering we have that the angular velocities all equal ω_{hover}. We find following expression for the angular velocity:

$$\omega^2 = \frac{\mathbf{T^b}}{4k} = \frac{m \cdot g}{4k}$$

where m is the mass of the system, g is the acceleration due to gravity, k is the propeller lift coefficient and $\mathbf{T^b}$ is the total thrust generated by the 4 rotors. We then find an expression for the squared voltage:

$$v^2 = \frac{mg}{4kc_m} \approx 40.88\, V^2,$$

Now that the operation point is found, the system can be linearized. The A-matrix can be obtained by calculating the Jacobian of the vector function $\boldsymbol{\xi(x, u)}$ with respect to the state vector $\mathbf{x}$ followed by evaluating it for $\mathbf{x} = \mathbf{x}^*$, $\mathbf{u} = \mathbf{u}^*$. Likewise, the B-matrix is obtained by calculating the Jacobian of $\boldsymbol{\xi(x, u)}$ with respect to the input vector $\mathbf{u}$ followed by evaluating it for the operating point. The following numerical values for the system matrices are found:

$$A = \frac{\partial \xi}{\partial x}|_{x^*,u^*} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \frac{\partial \xi}{\partial u}|_{x^*,u^*} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.06 & 0.06 & 0.06 & 0.06 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.5 & 0 & -1.5 & 0 \\ 0 & 1.5 & 0 & -1.5 \\ 0.1 & -0.1 & 0.1 & -0.1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

3 Discretization

Now we obtain a discrete-time state space model using the bilinear transformation and a sampling time of $T_s = 0.05$ s. The following system matrices are found:

$$A_d = \begin{bmatrix} 1 & 0 & 0 & 0.0494 & 0 & 0 & 0 & 0.0121 & 0 & 0 & 3.03e-04 & 0 \\ 0 & 1 & 0 & 0 & 0.0494 & 0 & -0.0121 & 0 & 0 & -3.03e-04 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.0494 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9753 & 0 & 0 & 0 & -0.4844 & 0 & 0 & 0.0121 & 0 \\ 0 & 0 & 0 & 0 & 0.9753 & 0 & -0.4844 & 0 & 0 & -0.0121 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9753 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.05 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0 & 1.14e-05 & 0 & -1.14e-05 \\ -1.14e-05 & 0 & 1.14e-05 & 0 \\ 7.41e-05 & 7.41e-05 & 7.41e-05 & 7.41e-05 \\ 0 & 4.54e-04 & 0 & -4.54e-04 \\ -4.54e-04 & 0 & 4.54e-04 & 0 \\ 0.003 & 0.003 & 0.003 & 0.003 \\ 0.0019 & 0 & -0.0019 & 0 \\ 0 & 0.0019 & 0 & -0.0019 \\ 1.25e-04 & -1.25e-04 & 1.25e-04 & -1.25e-04 \\ 0.075 & 0 & -0.075 & 0 \\ 0 & 0.075 & 0 & -0.075 \\ 0.005 & -0.005 & 0.005 & -0.005 \end{bmatrix}$$

$$C_d = \begin{bmatrix} 1 & 0 & 0 & 0.0247 & 0 & 0 & 0 & 0.0061 & 0 & 0 & 1.51e-04 & 0 \\ 0 & 1 & 0 & 0 & 0.0247 & 0 & -0.0061 & 0 & 0 & -1.51e-04 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.0247 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.025 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.025 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.025 \end{bmatrix}$$

$$D_d = \begin{bmatrix} 0 & 5.67e-06 & 0 & -5.67e-06 \\ -5.67e-06 & 0 & 5.67e-06 & 0 \\ 3.70e-05 & 3.70e-05 & 3.70e-05 & 3.70e-05 \\ 9.38e-04 & 0 & -9.38e-04 & 0 \\ 0 & 9.38e-04 & 0 & 9.38e-04 \\ 6.25e-05 & -6.25e-05 & 6.25e-05 & -6.25e-05 \end{bmatrix}$$

Before designing a controller we perform a short open-loop analysis on the discrete-time model. The system has nine unstable poles at 1 and three stable poles at 0.9753. The system is controllable and observable because the rank of the controllability matrix and the observability matrix equals 12, the rank of A_d. It follows that the system is also stabilisable and detectable. Since all the modes of A_d are observable and controllable, the system is minimal. The model has eight transmission zeros at -1.

In order to verify how good the discrete-time linear model is approximating the nonlinear model, a small step is applied to the inputs of the models and their respective outputs

compared. Figure 1 shows the Simulink diagram used to test the model. As can be seen from figure 2 the model output and states follow the nonlinear output well.

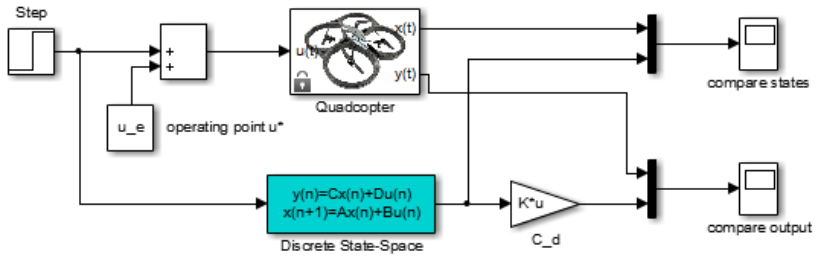

Figure 1: Simulink diagram used to test the discrete linear model.

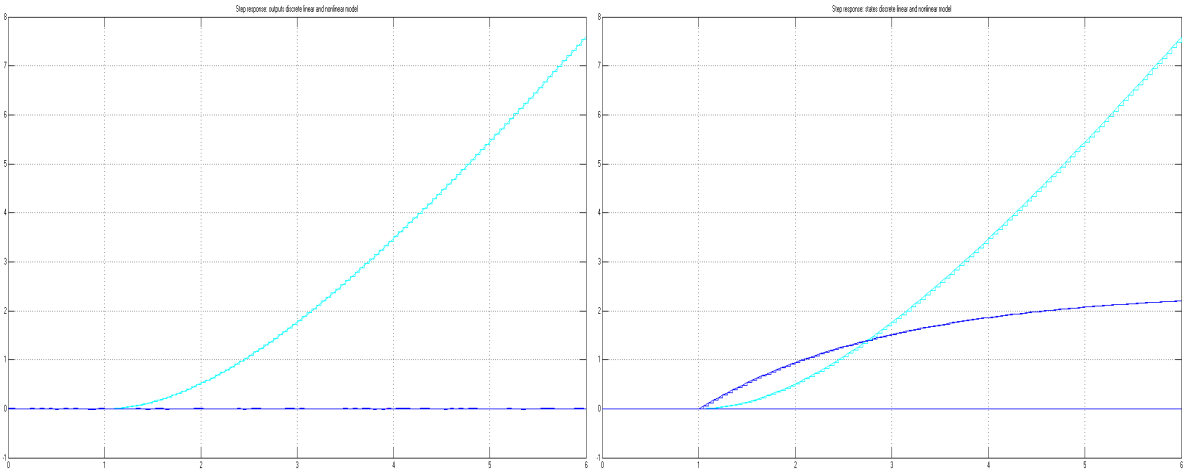

Figure 2: Left: comparison of the outputs of the nonlinear and discrete linear model, right: comparison of the states of the nonlinear and discrete linear model, both in function of time.

4 LQR control

The discrete-time model derived earlier is used to design an LQR controller. The control goal is a tracking problem: the quadcopter should reach six checkpoints in a room as fast as possible. A checkpoint is reached when the quadcopter enters a sphere of radius 0.08 m around the checkpoint. It is assumed the entire state-vector is available.

4.1 Experiments without payload

A first full-state feedback LQR controller is designed using reference input (N_x, N_u). The state feedback gain K was obtained by minimizing the LQR cost function. The weighting matrices were found by trial and error and plotting the step response. Lowering the R-value results in a smaller settling and rise time which is wanted but it also increases the control

effort. A large weight for the z-state is needed to ensure a fast and small transient for the z-output, but again a too large weight increases the control output. Balancing these elements, we settled on the following values for the weighting matrices:

$$Q = I_{12}, \quad Q(3,3) = 500$$
$$R = I_6.$$

To ensure zero steady-state error to a step input, the feedback control law is modified:

$$u_k = -Kx_k + (N_u + KN_x)r_k$$

where r_k is the reference input. The matrices $N_u \in \mathbb{R}^{4x6}$ and $N_x \in \mathbb{R}^{12x6}$ are calculated by solving the following system:

$$\begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} A_d - I_{12} & B_d \\ C_d & D_d \end{bmatrix}^{-1} \begin{bmatrix} zeros(12,6) \\ I_6 \end{bmatrix}$$

The Simulink diagram of the implemented LQR controller is shown in figure 3.

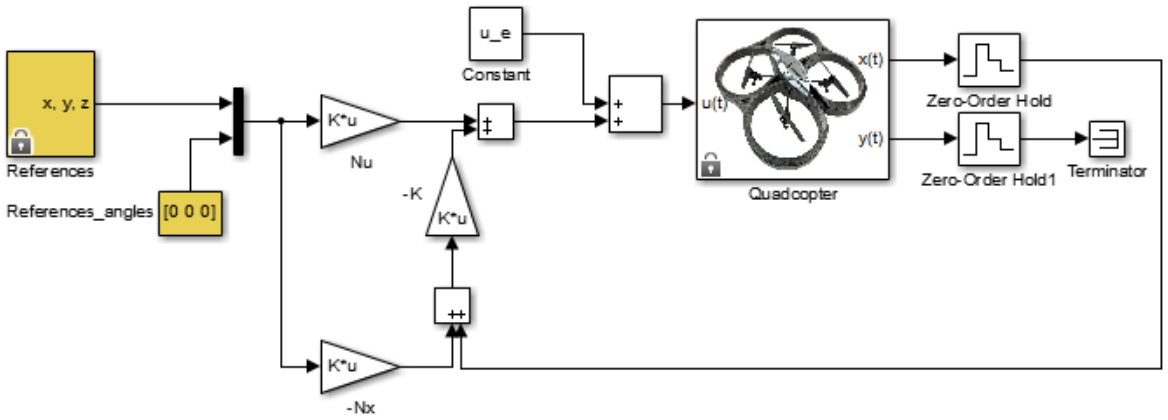

Figure 3: Simulink diagram of the LQR controller using reference input.

A second LQR controller is designed, now with integral action. Since we are interested in tracking the x, y and z coordinates we augment the plant with three extra states integrating the output error. In discrete time the integrator output x_{Ik+1} is computed using the forward Euler formula

$$x_{Ik+1} = x_{Ik} + T_s(y_k - r_k).$$

The augmented state equations become

$$\begin{bmatrix} x_{Ik+1} \\ x_{k+1} \end{bmatrix} = A_{Int} \begin{bmatrix} x_{Ik} \\ x_k \end{bmatrix} + B_{Int}u_k - \begin{bmatrix} I \\ 0 \end{bmatrix} r_k,$$

where

$$A_{Int} = \begin{bmatrix} I_3 & C_d(1:3,:) * T_s \\ zeros(12,3) & A_d \end{bmatrix}$$
$$B_{Int} = \begin{bmatrix} D_d(1:3,:) * T_s \\ B_d \end{bmatrix}.$$

The rank of the new controllability matrix equals 15 which is the rank of A_{Int}. This means the augmented system is controllable. Given these system matrices, the new feedback gain K_{Int} can be calculated. Since we have three extra states the weighting matrix Q is changed. Again we need a weight on the state that is the z coordinate. In addition we add a large weight for the extra state that integrates the error on the z coordinate. We use the following new weighting matrices:

$$Q_{Int} = I_{15}, \quad Q_{Int}(3,3) = 800,\ Q_{Int}(6,6) = 150$$
$$R_{Int} = 0.1 \cdot I_6$$

We changed the R as well to increase the speed. The Simulink diagram of the LQR controller with integral action is shown in figure 4.

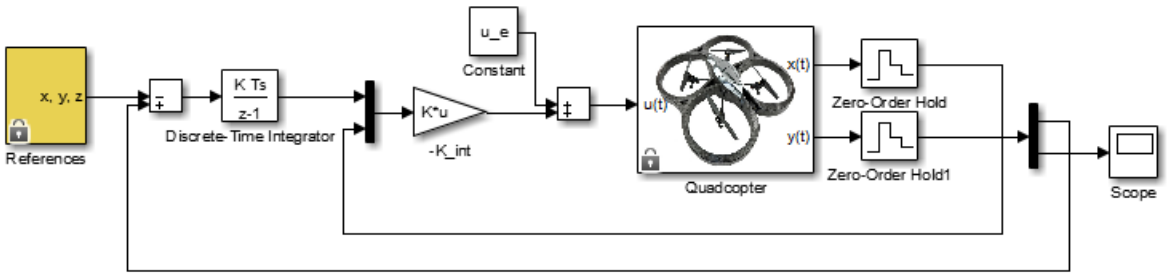

Figure 4: Simulink diagram of the LQR controller with integral.

4.1.1 Simulation results

The simulation reports (no payload) are given in table 1. Figure 5 and 6 shows the results obtained with the first LQR controller. Figure 7 and 8 shows the results obtained with the LQR controller with integral action. From these results both control strategies seem to perform equally well. The LQR controller with integral action is a second slower and the control actions are a bit higher (see figure 7 right), but the difference is minimal. There's only a slight difference in timing between the checkpoints. These differences could be due to the different weighting matrices that were used when calculating the feedback gain.
Note that the average time for both simulations could be made smaller by reducing the weight matrix R even more. But this would also lead to increasing voltages.

LQR - Simulation report
Full state vector: available
Number of checkpoints reached: 7/7
Payload: 0 kg
From checkpoint 0 to 1: 1.85 s
From checkpoint 1 to 2: 4.20 s
From checkpoint 2 to 3: 4.20 s
From checkpoint 3 to 4: 4.85 s
From checkpoint 4 to 5: 4.80 s
From checkpoint 5 to 6: 4.60 s
From checkpoint 6 to 7: 1.70 s
Average time: 3.743 s

LQR integral control - Simulation report
Full state vector: available
Number of checkpoints reached: 7/7
Payload: 0 kg
From checkpoint 0 to 1: 1.85 s
From checkpoint 1 to 2: 4.50 s
From checkpoint 2 to 3: 4.50 s
From checkpoint 3 to 4: 4.85 s
From checkpoint 4 to 5: 4.85 s
From checkpoint 5 to 6: 4.70 s
From checkpoint 6 to 7: 1.70 s
Average time: 3.850 s

Table 1: Simulation reports of the two LQR controllers; quadcopter without payload.

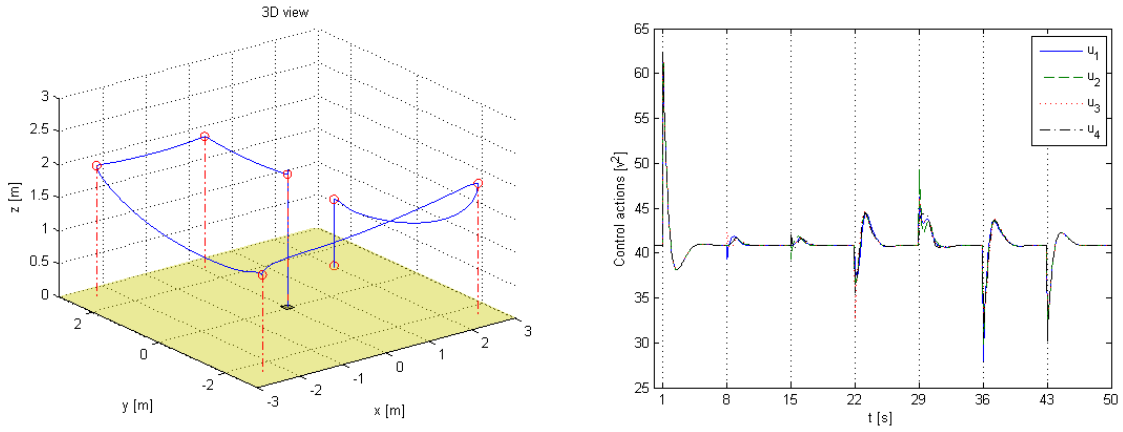

Figure 5: First LQR controller without payload - Left: 3D view of the trajectory followed by the quadcopter, right: control actions.

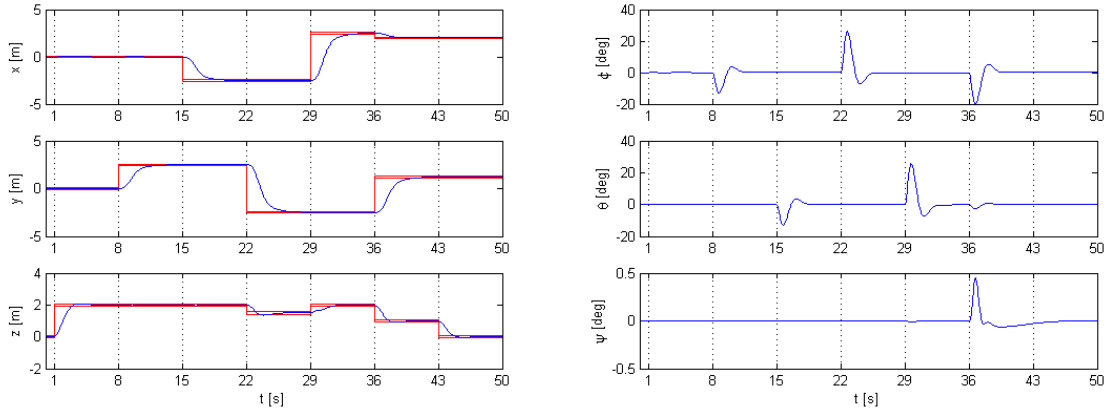

Figure 6: First LQR controller without payload - Left: evolution of the x, y and z coordinates, right: evolution of the angles.

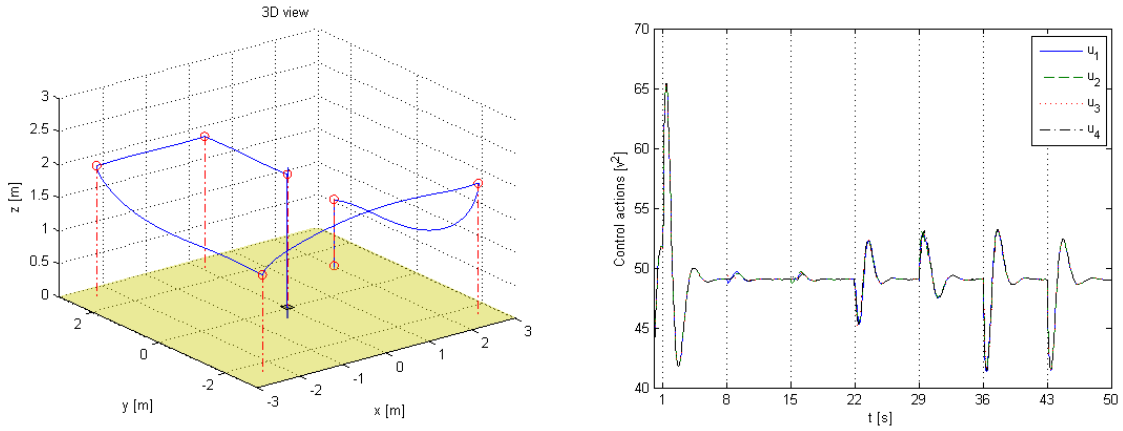

Figure 7: Integral action without payload - Left: 3D view of the trajectory followed by the quadcopter, right: control actions

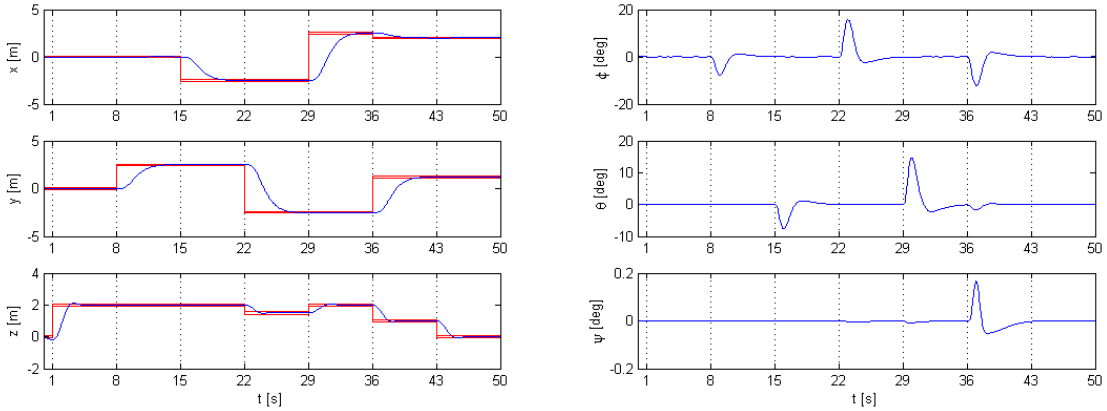

Figure 8: Integral action without payload - Left: evolution of the x, y and z coordinates, right: evolution of the angles.

4.2 Experiments with payload

We add a disturbance affecting the system dynamics by adding a payload of 0.1 kg to the quadcopter. We simulate both LQR control systems again, keeping the weighting matrices previously found.

4.2.1 Simulation results

When simulating the system with the first LQR controller with a payload of 0.1 kg, the performance has noticeably decreased. The tracking goals aren't met anymore because the controller is not able to drive the quadcopter high enough. This can be seen in figure 9, where the trajectory along the z-component is below the reference trajectory.

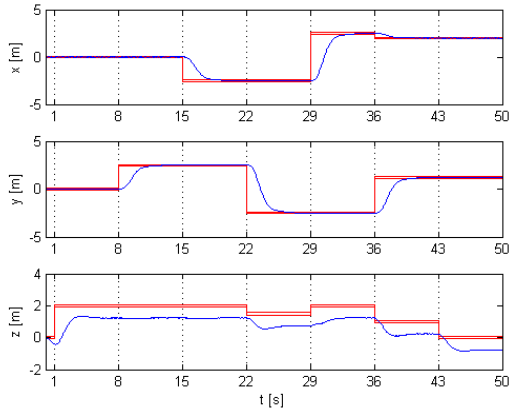

Figure 9: Evolution of the x, y and z coordinates for the first LQR controller with payload using same Q, R.

We try changing the weighting matrices to tune the first controller. When using the following values for Q and R, the new controller is able to drive the quadcopter to the checkpoints in time:

$$Q1 = I_{12}, \quad Q_{Int}(3,3) = 700$$
$$R1 = 0.01 \cdot I_6$$

By increasing the weight of the third state (z coordinate) , we can penalize the error of the z-trajectory more. Lowering the R value results in a faster response which is especially needed during the lift up in the first second. When we simulate the LQR controller with these new weighting matrices, the quadcopter is able to reach all the checkpoints. The simulation results can be found in table 2 left and figures 10 and 11. The average time is now lower (about 4 seconds lower), but the control actions are much higher, reaching the maximum voltage of 10 V (see figure 10 right).

The controller with integral action still performs well using the same weighting matrices as in the case without payload. This can be seen from the simulation report in table 2 right and figures 12 and 13. The effect of a payload results in a little higher average time (about 0.2 seconds higher), but the control actions are about the same (see figure 12 right).

The controller with integral control is more robust to the disturbance of the payload. By adding reference input using N_x and N_u in the first controller, the step respone has a zero steady-state error. But any disturbance can cause the error to be nonzero. In integral control the integral term magnifies the effect of long-term steady-state errors, reducing them to zero. This leads to more robust tracking.

LQR (new Q, R) - Simulation report
Full state vector: available
Number of checkpoints reached: 7/7
Payload: 0.1 kg
From checkpoint 0 to 1: 2.05 s
From checkpoint 1 to 2: 3.65 s
From checkpoint 2 to 3: 3.65 s
From checkpoint 3 to 4: 4.20 s
From checkpoint 4 to 5: 4.40 s
From checkpoint 5 to 6: 4.00 s
From checkpoint 6 to 7: 1.35 s
Average time: 3.329 s

LQR integral control - Simulation report
Full state vector: available
Number of checkpoints reached: 7/7
Payload: 0.1 kg
From checkpoint 0 to 1: 2.60 s
From checkpoint 1 to 2: 4.60 s
From checkpoint 2 to 3: 4.60 s
From checkpoint 3 to 4: 5.05 s
From checkpoint 4 to 5: 5.05 s
From checkpoint 5 to 6: 4.85 s
From checkpoint 6 to 7: 1.65 s
Average time: 4.057 s

Table 2: Simulation reports of the first LQR controller with new Q and R and of the LQR with integral action; quadcopter with payload.

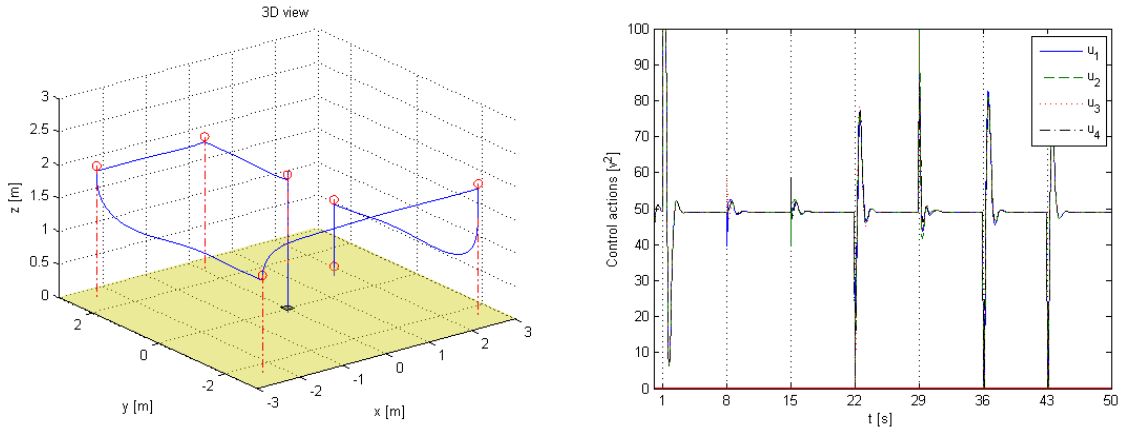

Figure 10: First LQR controller with payload and new Q and R - Left: 3D view of the trajectory followed by the quadcopter using the second LQR controller, right: control actions.

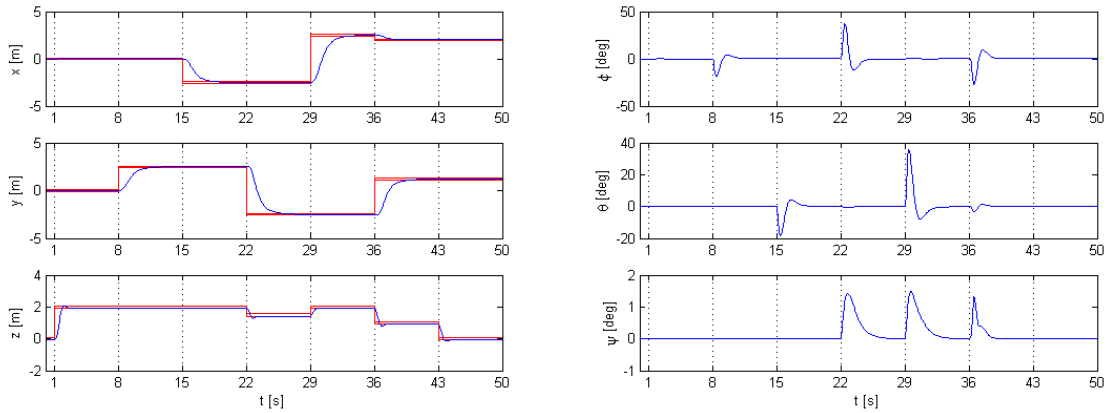

Figure 11: First LQR controller with payload and new Q and R - Left: evolution of the x, y and z coordinates, right: evolution of the angles.

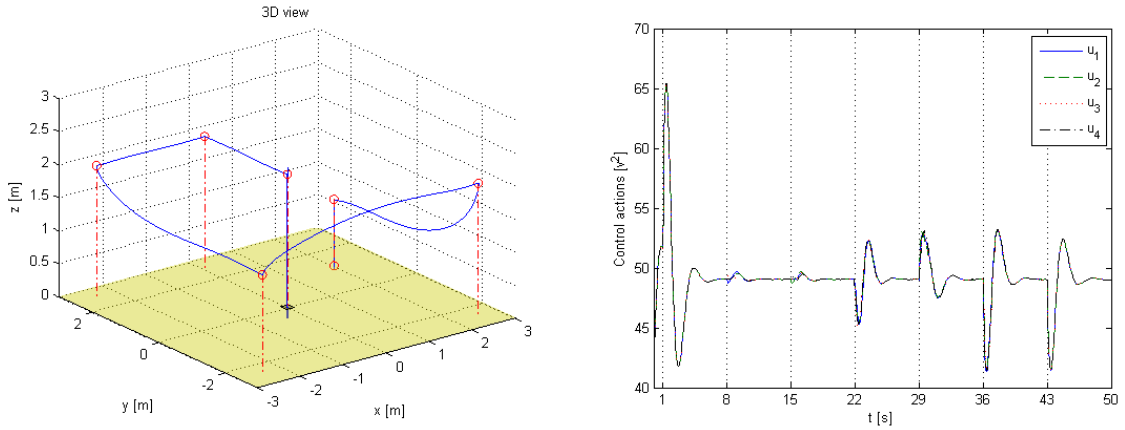

Figure 12: Integral action with payload - Left: 3D view of the trajectory followed by the quadcopter using the second LQR controller, right: control actions.

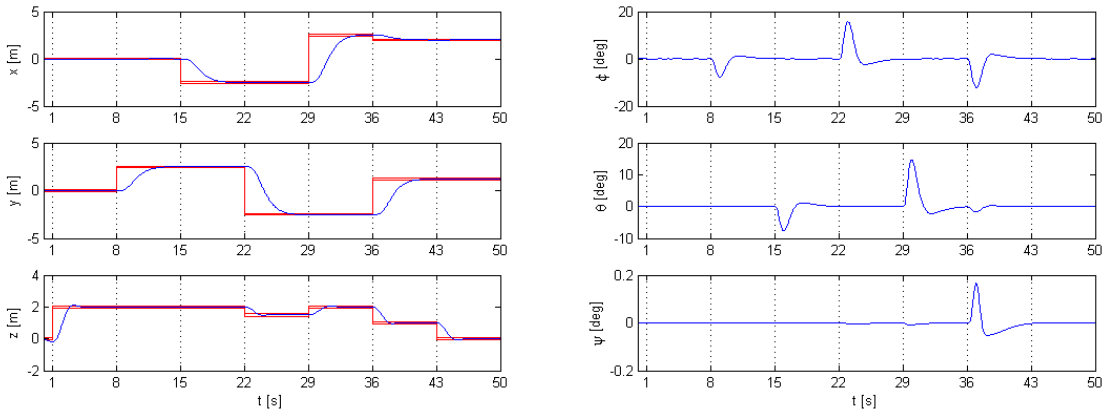

Figure 13: Integral action with payload - Left: evolution of the x, y and z coordinates, right: evolution of the angles.

5 LQG control

When designing the previous LQR controllers, it was assumed the entire state-vector is available. Since this is not the case anymore the state-vector $\hat{x}$ needs to be estimated from the measurements given by the sensors. We do this by implementing a Kalman filter. Given the system

$$x_{k+1} = A_d x_k + B_d u_k + B_d w_k \qquad (State\ equations)$$
$$y_k = C_d x_k + D_d u_k + v_k \qquad (Measurements)$$

where process noise w_k and measurement noise v_k are assumed to be zero mean white noises with the following covariances:

$$E(ww^T) = R_w$$
$$E(vv^T) = R_v$$

where $E(..)$ is the expected value. The covariance matrices of process and measurement noise are diagonal (independent noise inputs). The entries on the diagonal of the covariance matrix are the variances of each element of the noise vector. These are given for the measurement noise so the covariance matrix becomes

$$R_v = \begin{bmatrix} 2.5\cdot10^{-5} & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.5\cdot10^{-5} & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.5\cdot10^{-5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.57\cdot10^{-5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 7.57\cdot10^{-5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 7.57\cdot10^{-5} \end{bmatrix}$$

For the process noise we use the following covariance matrix:

$$R_w = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

Note that we assumed that the process noise is added to the input so that the process noise is in reality input noise.

Using the *lqe*-command in Matlab the observer gain matrix M is found such that the discrete Kalman filter

$$\hat{x}_{k+1} = A_d\hat{x}_k + B_d u_k + A_d M(y_k - C_d\hat{x}_k - D_d u_k)$$

$$\hat{x}_{k+1} = [A_d - A_d M C_d]\,\hat{x}_k + [B_d - A_d M D_d \quad A_d M] \begin{bmatrix} u_k \\ y_k \end{bmatrix}$$

gives an optimal estimate of the states. The LQG controller K then becomes

$$u_k = -K\hat{x}_k$$

which is the same as the LQR controller previously found. We use K_{Int} since we want to keep the integral action. The Simulink diagram implementing the Kalman filter and using the previously obtained LQR controller with integral action is shown in figure 14.

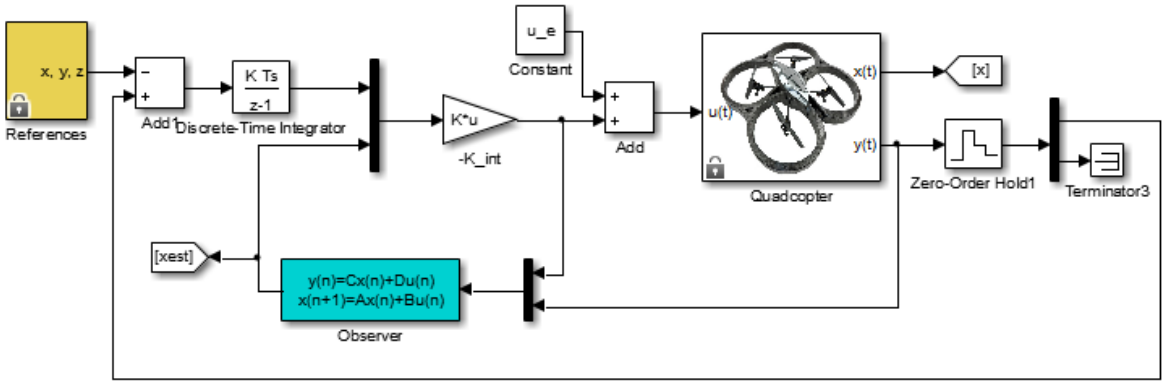

Figure 14: Simulink diagram of the LQR controller with integral action and Kalman filter.

5.1 Experiments with and without payload

The simulation results can be found in table 3 left and figures 15 and 16 for LQG without payload and figures 17 and 18 for LQG with payload. The average time for both cases (with and without load) are more or less the same. Adding the Kalman filter makes the quadcopter just a bit slower (about 0.1 s). However more control effort is used (figure 15 and 17 right). Also noticeable is the noise on the angles (figure 18 and 18 right) which results in trajectory.

LQG no load - Simulation report
Full state vector: estimated by an Observer
Number of checkpoints reached: 7/7
Payload: 0 kg
From checkpoint 0 to 1: 1.85 s
From checkpoint 1 to 2: 4.55 s
From checkpoint 2 to 3: 4.75 s
From checkpoint 3 to 4: 5.10 s
From checkpoint 4 to 5: 4.75 s
From checkpoint 5 to 6: 5.20 s
From checkpoint 6 to 7: 1.25 s
Average time: 3.921 s

LQG with load - Simulation report
Full state vector: available
Number of checkpoints reached: 7/7
Payload: 0.1 kg
From checkpoint 0 to 1: 2.40 s
From checkpoint 1 to 2: 4.65 s
From checkpoint 2 to 3: 4.85 s
From checkpoint 3 to 4: 5.55 s
From checkpoint 4 to 5: 5.15 s
From checkpoint 5 to 6: 5.50 s
From checkpoint 6 to 7: 1.35 s
Average time: 4.207 s

Table 3: Simulation reports of the LQG controller with integral action; quadcopter without and with payload.

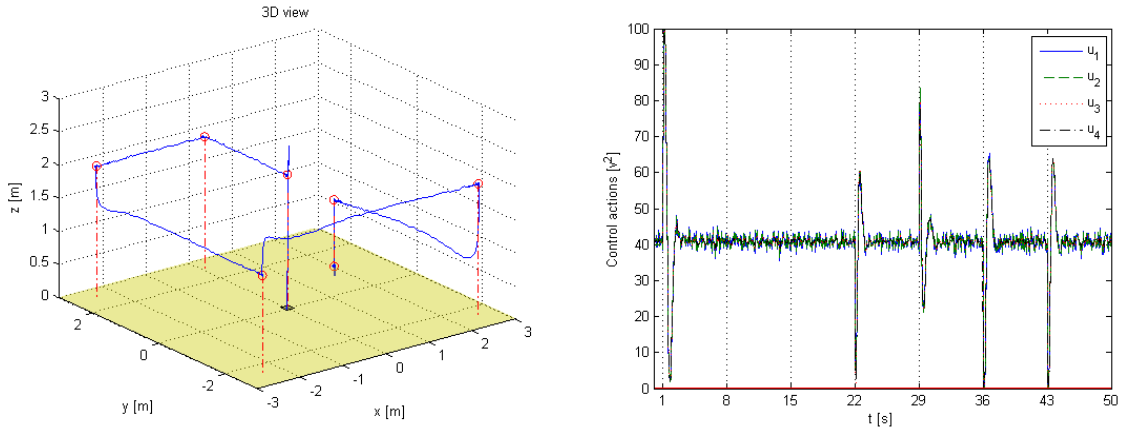

Figure 15: LQG without payload - Left: 3D view of the trajectory followed by the quadcopter, right: control actions.

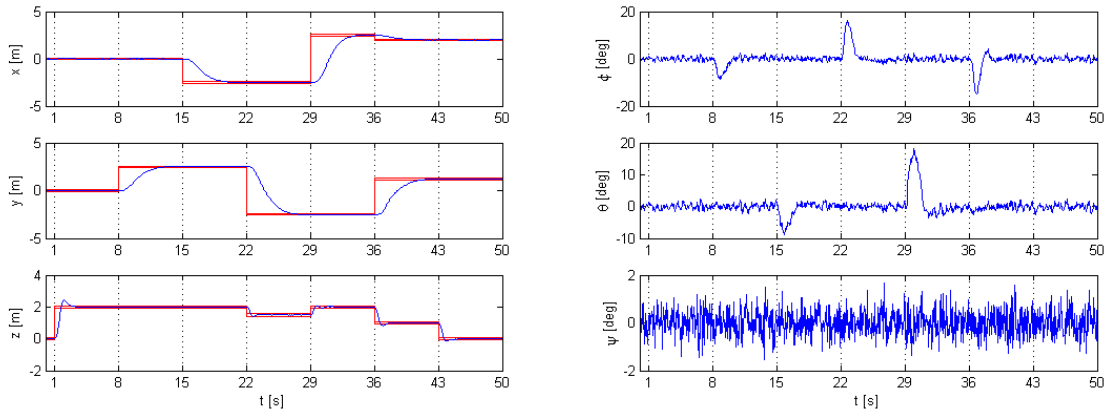

Figure 16: LQG without payload - Left: evolution of the x, y and z coordinates, right: evolution of the angles.

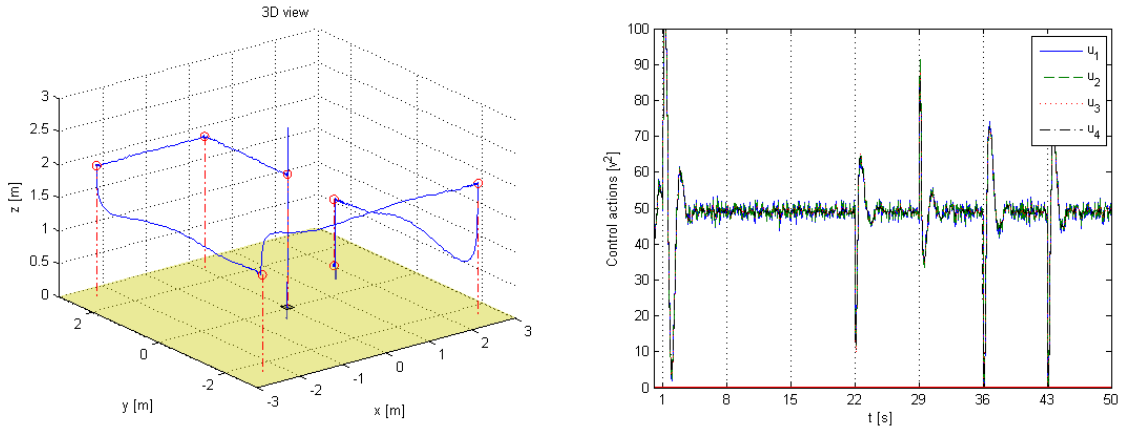

Figure 17: LQG with payload - Left: 3D view of the trajectory followed by the quadcopter, right: control actions.

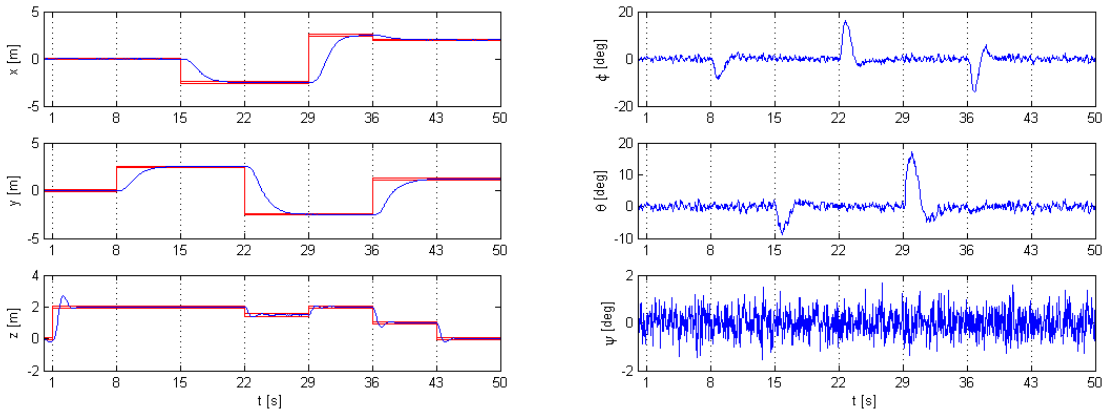

Figure 18: LQG with payload - Left: evolution of the x, y and z coordinates, right: evolution of the angles.

6 Conclusion

We've designed a controller that drives the quadcopter to six checkpoints as fast as possible. We started out by designing a simple full-state feedback LQR controller which we then improved by using integral action. This made the controller more robust and able to manage the payload better. Eventually, we designed an LQG control system: a Kalman filter that estimates the entire state-vector is used together with the LQR feedback. In order to do is we made some assumptions about the Gaussian character of the noise. The design goals were met in all cases.