Quadcopter assignment

Methods and Algorithms for Advanced Process Control

Zander Op de Beeck & Pieter Vanslambrouck

June 3, 2022

This report was submitted in combination with the corresponding Simulink and Matlab files. A description of these files is given in the file `description.txt`.

1 Linearization

In steady state, the entire 12D state vector is filled with zero entries. Additionally, we can determine the control inputs in the steady state using Equation (10).

$$v_1^2 + v_2^2 + v_3^2 + v_4^2 = \frac{gm}{kc_m} \Rightarrow v_i^2 = \frac{gm}{4kc_m} \quad i = 1, 2, 3, 4$$

Hence, we have

$$\begin{cases} \mathbf{x}^* = \texttt{zeros(12,1)} \\ \mathbf{u}^* = \begin{pmatrix} \frac{gm}{4kc_m} & \frac{gm}{4kc_m} & \frac{gm}{4kc_m} & \frac{gm}{4kc_m} \end{pmatrix}^T \\ \mathbf{y}^* = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T. \end{cases}$$

The nonlinear equations are $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. The corresponding Jacobian is

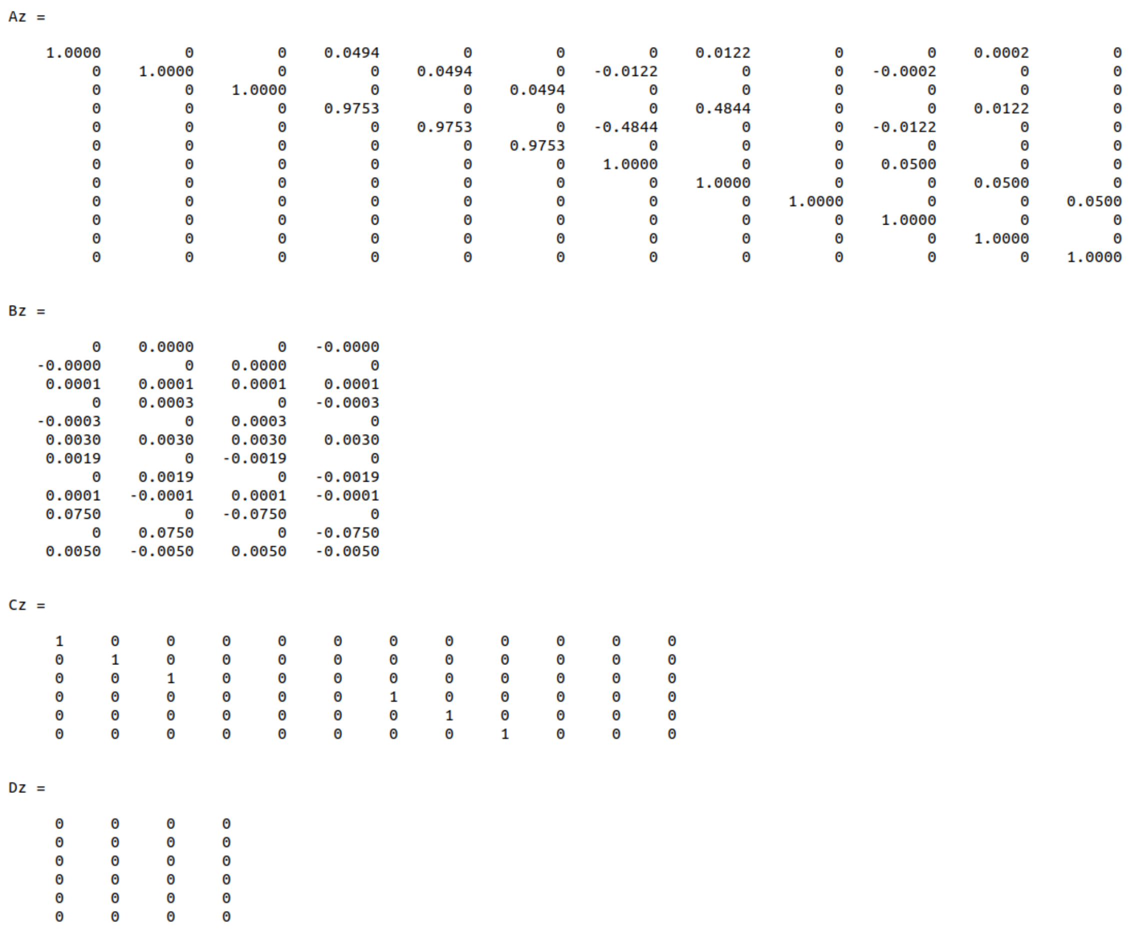
$$\begin{aligned} A &= \begin{pmatrix} \frac{\partial f_1}{\partial x} & \cdots & \frac{\partial f_1}{\partial \omega_z} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{12}}{\partial x} & \cdots & \frac{\partial f_{12}}{\partial \omega_z} \end{pmatrix} \\ &= \begin{pmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & A_{22} & A_{23} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{pmatrix} \end{aligned}$$

with

$$A_{22} = \frac{-k_d}{m} I_{3\times3} = -0.5 I_{3\times3} \qquad A_{23} = \begin{pmatrix} 0 & g & 0 \\ -g & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 9.81 & 0 \\ -9.81 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Similarly, the matrix B is found.

$$\begin{aligned} B &= \begin{pmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_4} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{12}}{\partial u_1} & \cdots & \frac{\partial f_{12}}{\partial u_4} \end{pmatrix} \\ &= \begin{pmatrix} 0_{5\times4} \\ \frac{kc_m}{m} \texttt{ones(1,4)} \\ 0_{3\times4} \\ B_{\text{lower}} \end{pmatrix} \end{aligned}$$



```
Az =

    1.0000         0         0    0.0494         0         0         0    0.0122         0         0    0.0002         0
         0    1.0000         0         0    0.0494         0   -0.0122         0         0   -0.0002         0         0
         0         0    1.0000         0         0    0.0494         0         0         0         0         0         0
         0         0         0    0.9753         0         0         0    0.4844         0         0    0.0122         0
         0         0         0         0    0.9753         0   -0.4844         0         0   -0.0122         0         0
         0         0         0         0         0    0.9753         0         0         0         0         0         0
         0         0         0         0         0         0    1.0000         0         0    0.0500         0         0
         0         0         0         0         0         0         0    1.0000         0         0    0.0500         0
         0         0         0         0         0         0         0         0    1.0000         0         0    0.0500
         0         0         0         0         0         0         0         0         0    1.0000         0         0
         0         0         0         0         0         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0         0         0         0         0         0    1.0000


Bz =

         0    0.0000         0   -0.0000
   -0.0000         0    0.0000         0
    0.0001    0.0001    0.0001    0.0001
         0    0.0003         0   -0.0003
   -0.0003         0    0.0003         0
    0.0030    0.0030    0.0030    0.0030
    0.0019         0   -0.0019         0
         0    0.0019         0   -0.0019
    0.0001   -0.0001    0.0001   -0.0001
    0.0750         0   -0.0750         0
         0    0.0750         0   -0.0750
    0.0050   -0.0050    0.0050   -0.0050


Cz =

     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


Dz =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Figure 1: The matrices of the discretized linearized system. The zero-order hold method was used to obtain these matrices.

with $\frac{kc_m}{m} = 0.06$ and

$$B_{\text{lower}} = \begin{pmatrix} \frac{Lkc_m}{I_{xx}} & 0 & -\frac{Lkc_m}{I_{xx}} & 0 \\ 0 & \frac{Lkc_m}{I_{yy}} & 0 & -\frac{Lkc_m}{I_{yy}} \\ \frac{bc_m}{I_{zz}} & -\frac{bc_m}{I_{zz}} & \frac{bc_m}{I_{zz}} & -\frac{bc_m}{I_{zz}} \end{pmatrix} = \begin{pmatrix} 1.5 & 0 & -1.5 & 0 \\ 0 & 1.5 & 0 & -1.5 \\ 0.1 & -0.1 & 0.1 & -0.1 \end{pmatrix}.$$

Finally, the entries of C are quite straightforward.

$$C = \begin{pmatrix} I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \end{pmatrix}$$

We have verified that the linear approximation behaves as expected by implementing the suggested testcase (for reference, this was implemented in `test_linearization.slx`). The outputs after applying the described step change match closely. Hence, we are confident the matrices above are correct.

2 Discretization

We choose the zero-order hold method to discretize the system. This method is chosen because it has good properties. In particular, the physical meaning of the state vector is preserved. The numerical values of the resulting matrices are given in Figure 1.

The discretized system is marginally stable. The poles are located either on or within the unit circle. Since the controllability and observability have rank $n = 12$, the system is controllable and observable. This means the system is also stabilizable and detectable, because unstable modes are controllable and observable. Finally, it is minimal because the system is both controllable and observable.

Using the command `tzero`, we find 2 transmission zeros that are approximately equal to -1. This means we can find inputs of the form $u[k] = u_0(-1)^k$ that result in a zero output of the linearized system[1].

For the remainder of the report, we will refer to the matrices of the discretized system with A, B, C and D for notational convenience.

3 LQR control

The control goals are as following. The drone has to reach each checkpoint within 7 seconds with a maximal error of 0.08 m. Additionally, the entire trajectory should respect the physical limitations of the room, which has dimensions 6 m × 6 m × 3 m. Finally, the control system should also be robust enough to attain these goals with a payload of 0.1 kg.

3.1 Full-state feedback controller

First, we compute the matrices N_x and N_u. For this purpose, we construct a matrix

$$M = \begin{pmatrix} A - I_{12\times12} & B \\ C & D \end{pmatrix}.$$

Since the system is underactuated[2], the matrix equation $M \begin{pmatrix} N_x \\ N_u \end{pmatrix} = \begin{pmatrix} 0_{12\times6} \\ I_{6\times6} \end{pmatrix}$ is over-determined. In MATLAB, we can compute the least-squares solution conveniently by using the backslash-operator: $\begin{pmatrix} N_x \\ N_u \end{pmatrix} = M \backslash \begin{pmatrix} 0_{12\times6} \\ I_{6\times6} \end{pmatrix}$.

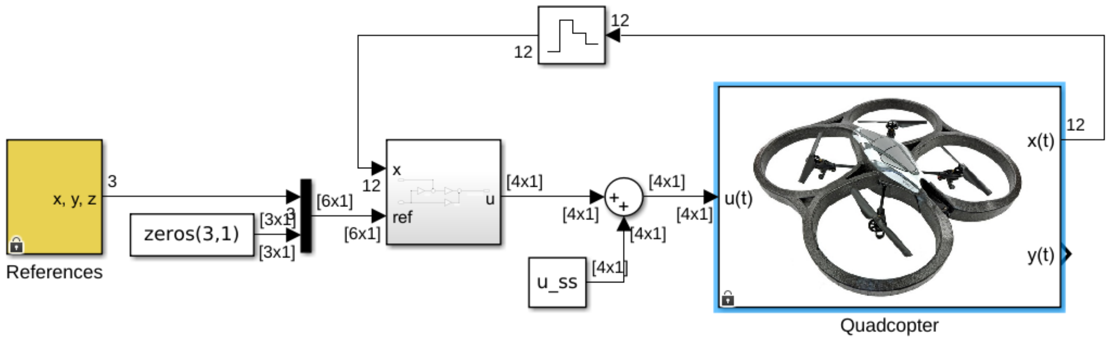
Next, the gain K can be easily determined by using `dlqr(A,B,Q,R)` in MATLAB.

We set $R = I_{4\times4}$ and tune Q relatively to R. The entries of Q corresponding to velocities are set to zero because limiting the speed of the movement of the quadcopter is not our primary goal. After iterative tuning, we choose `Q = diag([1e3 1e3 1e3 0 0 0 1e4 1e4 1e2 0 0 0])`.

The Simulink diagram of this controller can be found in Figure 2. Simulation results with and without payload are given in Figures 3 and 4. In the simulation without payload, we made sure to achieve the control goals mentioned at the beginning of this section (Section 3). However, the z-coordinate cannot be tracked accurately in the presence of a payload.

[1]The vector u_0 corresponding to a transmission zero can be found by determining the null space of a matrix, see course notes p. 82.

[2]There are 6 outputs, but only 4 inputs.



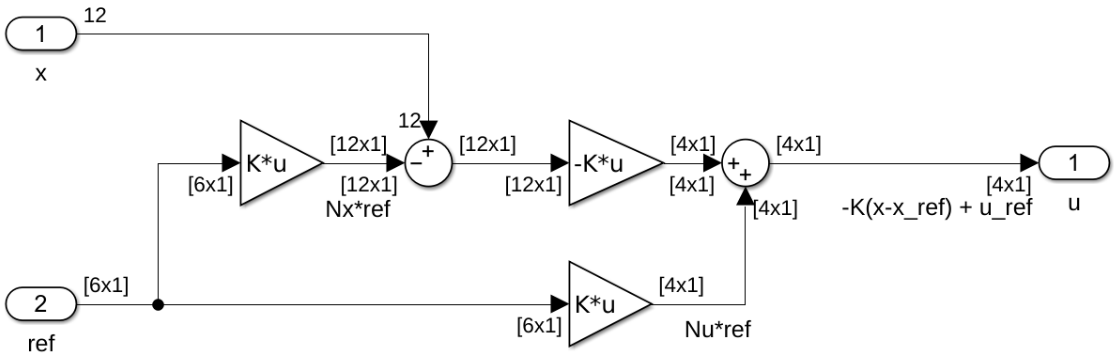


Figure 2: Simulink diagram of the standard **LQR** controller. The lower figure is the subsystem from the upper figure. As the controller is designed for the discretized system, we have added a zero-order hold block with a sampling time of $T_s = 0.05$ s.

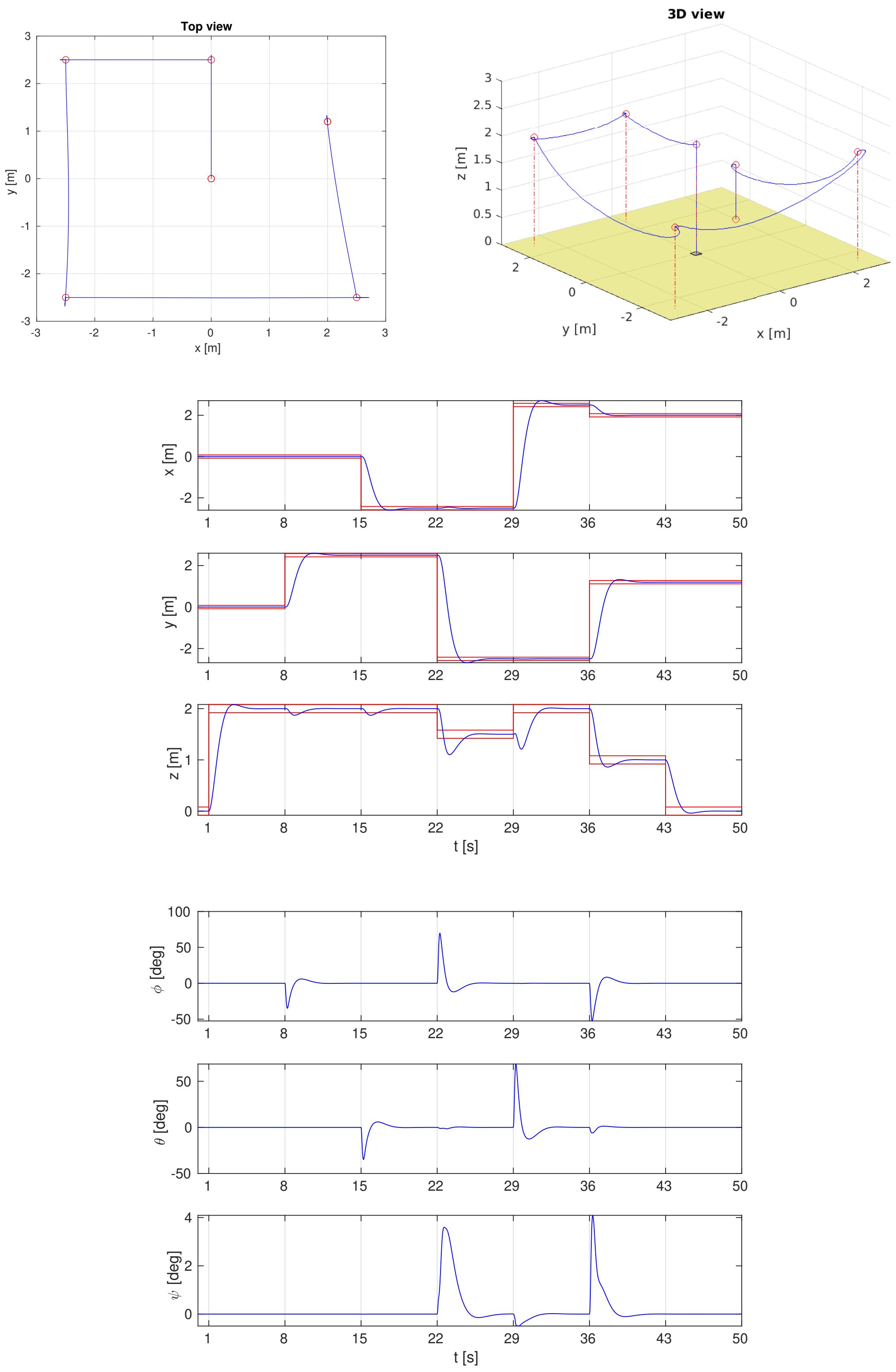


Figure 3: Simulation results of the standard **LQR** controller without payload.

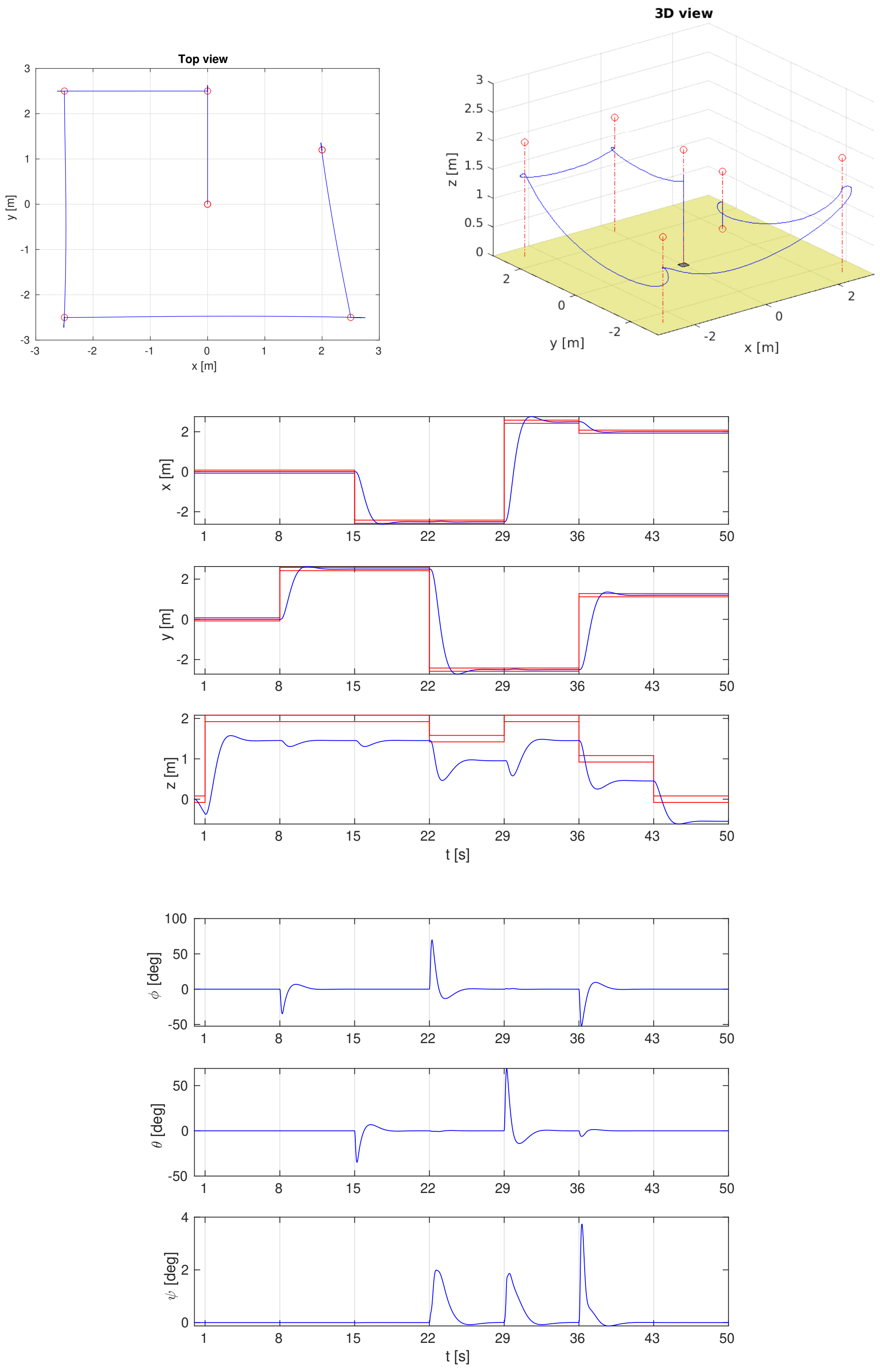


Figure 4: Simulation results of the standard **LQR** controller with a payload of 0.1 kg. The z-coordinate exhibits a clear steady-state error. Moreover, we sometimes have $z < 0$, which is impossible but not modelled by the nonlinear quadcopter block in Simulink.

3.2 LQR controller with integral action

The state vector is augmented by concatenating it with the vector x_I that satisfies

$$x_I(k+1) = x_I(k) + y(k)[1:3] - r(k) = x_I(k) + C[1:3,:]x(k) + D[1:3,:]u(k) - r(k)$$

Where $r(k)$ are the reference outputs. The evolution of the state of the augmented system is defined by

$$\begin{aligned}\begin{pmatrix} x_I(k+1) \\ x(k+1) \end{pmatrix} &= \begin{pmatrix} I_{3\times3} & C[1:3,:] \\ 0_{12\times3} & A \end{pmatrix} \begin{pmatrix} x_I(k) \\ x(k) \end{pmatrix} + \begin{pmatrix} D[1:3,:] \\ B \end{pmatrix} u(k) + \begin{pmatrix} -I_{3\times3} \\ 0_{12\times3} \end{pmatrix} r(k) \\ &:= A_{\text{int}} \begin{pmatrix} x_I(k) \\ x(k) \end{pmatrix} + B_{\text{int}} u(k) + \begin{pmatrix} -I_{3\times3} \\ 0_{12\times3} \end{pmatrix} r(k)\end{aligned}$$

Is this augmented system controllable? To figure this out, we compute $M_c = \texttt{ctrb}(A_{\text{int}}, B_{\text{int}})$. Since the matrix M_c is of full rank, the system is controllable and we can proceed without any issues.

Next, the gain K_{15} is computed by using `K_15 = dlqr(A_int, B_int, Q_int, R_int)`. We again set $R_{\text{int}} = I_{4\times4}$ and tune Q_{int} relatively to R_{int}. Only the entries of Q_{int} corresponding to the integrated position errors and the 3 orientation angles are non-zero. After tuning, we choose `Q_int = diag([1e3 1e3 1e3, 0 0 0, 0 0 0, 1e6 1e6 1e2, 0 0 0])`.

The Simulink diagram of this controller can be found in Figure 5. Simulation results with and without payload are given in Figures 6 and 7. We made sure to achieve the control goals mentioned at the beginning of this section (Section 3). During landing the z-coordinate becomes negative for a short timespan, but in reality this would most likely not result in any issues. The controller is robust enough to survive a little bump on the ground.

3.3 Comparison of LQR controllers

The standard LQR controller relies on an exact model of the system. On the other hand, the LQR controller with integral action is more robust since it actively incorporates the tracking error and adjusts the control inputs accordingly. This is akin to the integral action in a PID controller, which eliminates the steady-state error.

We can verify these claims by looking at the simulation results. The results without payload are given in Figures 3 and 6. The results with payload are given in Figures 6 and 7. When the model accurately describes the quadcopter (i.e. no payload), both models perform well. However, if we add a payload the standard LQR controller doesn't manage to track the reference height. There's a huge steady-state error. The more robust LQR controller with integral action doesn't have this problem. Moreover, the simulation results with payload are almost indistinguishable from the ones without payload!

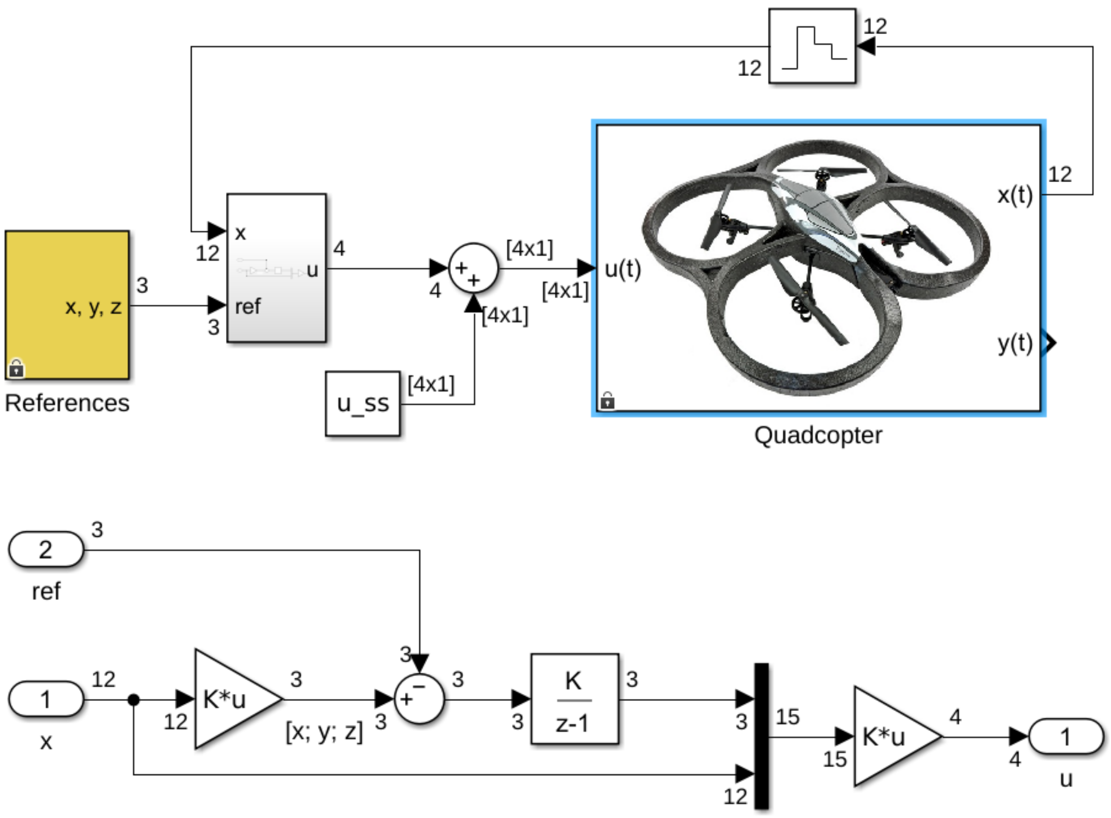


Figure 5: Simulink diagram of the **LQR** controller with **integral action**. The lower figure is the subsystem from the upper figure.

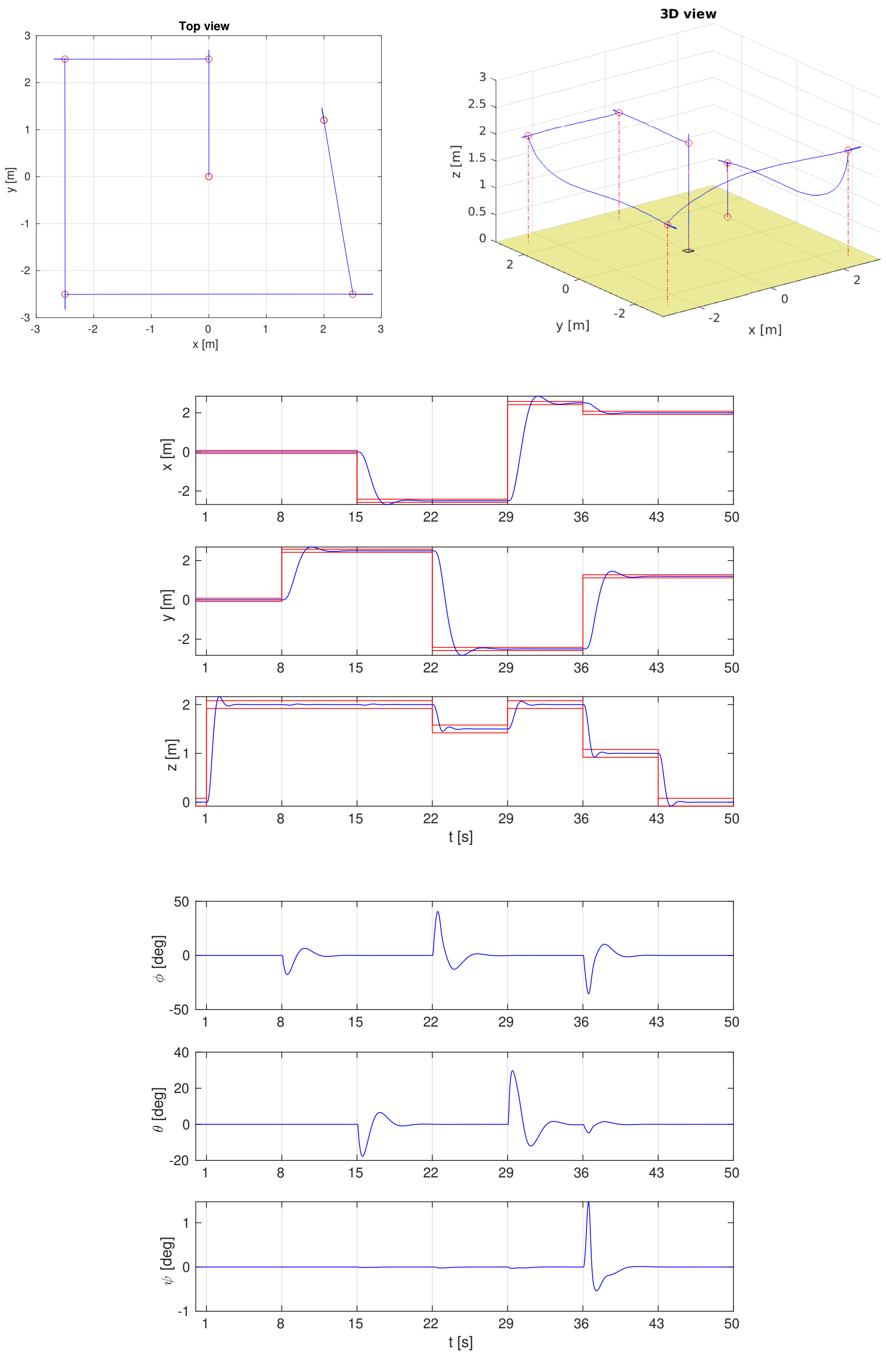


Figure 6: Simulation results of the **LQR** controller with **integral action** without payload.

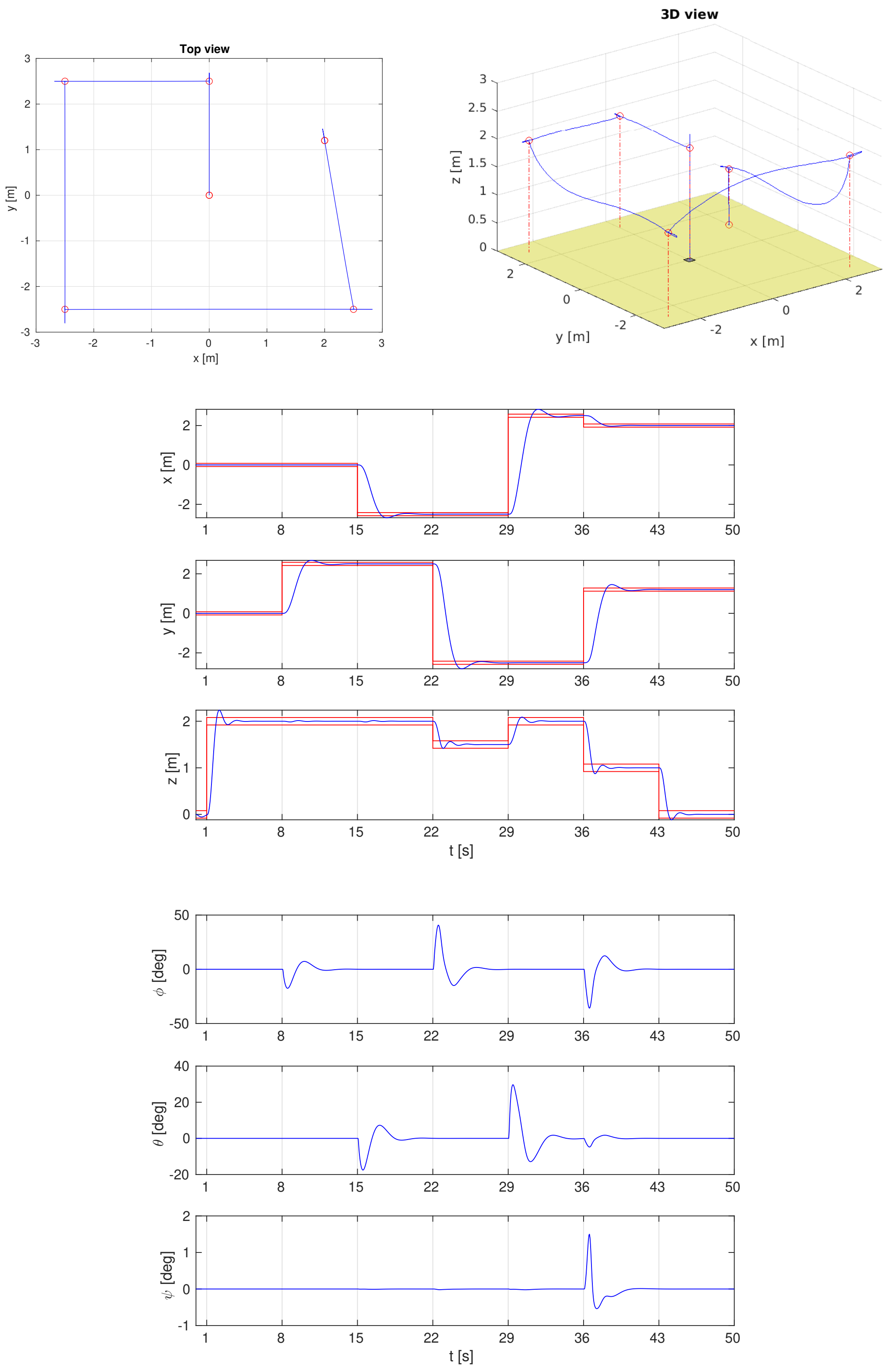


Figure 7: Simulation results of the **LQR** controller with **integral action** with a payload of 0.1 kg. Compared to the simulation of the standard LQR controller (Figure 4), the steady-state error of the z-coordinate has disappeared.

4 LQG control

We assume the measurement errors of the output to be uncorrelated. Hence, we set

```
R_kalman = diag([2.5e-5 2.5e-5 2.5e-5 7.57e-5 7.57e-5 7.57e-5]).
```

Initially, the covariance matrix of the process noise set to $\sigma^2 I_{12\times 12}$, where the parameter σ is tuned to achieve the desired results. There is a trade-off between speed of the state estimation and sensitivity to noise. We chose $\sigma = 10^{-3}$. This choice of Q works fine in the simulation without payload. However, in the presence of a payload a large steady-state error in the z-coordinate is visible. To overcome this, we overwrite the element `Q(6,6)` to 10^{-3}. This models the fact that the velocity of the z-coordinate is more uncertain due to the additional payload. We choose this adapted matrix Q for the rest of this section.

The matrix L of the estimator is computed in MATLAB as following.

```
M = dlqe(A, eye(12), C, Q_kalman, R_kalman);
L = AM;
```

The estimator is defined as

$$\begin{aligned}\hat{x}(k+1) &= A\hat{x}(k) + Bu(k) + L(y(k) - C\hat{x}(k) - Du(k)) \\ &= (A - LC)\hat{x}(k) + \begin{pmatrix} B - LD & L \end{pmatrix} \begin{pmatrix} u(k) \\ y(k) \end{pmatrix}.\end{aligned}$$

By writing the state equation as in the last line of the equation above, we can conveniently implement the state estimator in Simulink. A block has been made with input $\begin{pmatrix} u(k) \\ y(k) \end{pmatrix}$ and output $\hat{x}(k)$ (Figure 8).

The Simulink diagram of this controller can be found in Figure 8. Simulation results with and without payload are given in Figures 9 and 10. We compare the simulation results with the ones of the LQR controller with integral action from the previous section (Figures 4 and 7). The plots are very similar. The added measurement noise is visible in the evolution of the angles and the 3D trajectory. A small steady-state error on the z-coordinate is also visible for the LQG controller. Additionally, the oscillations of the z-coordinate have been amplified a bit. We conclude that the addition of noise and a Kalman filter has not made a big impact on the simulation results.

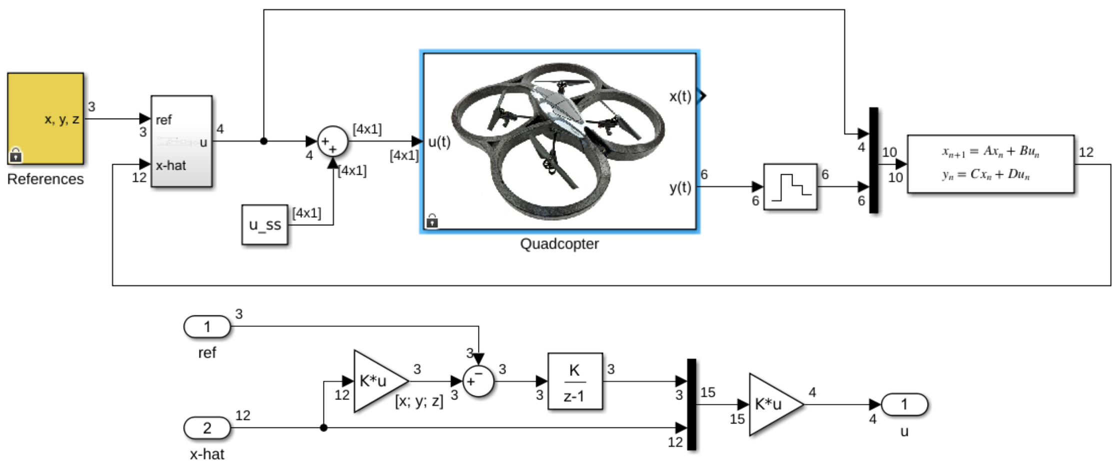


Figure 8: Simulink diagram of the **LQG** controller with integral action. The lower figure is the subsystem from the upper figure.

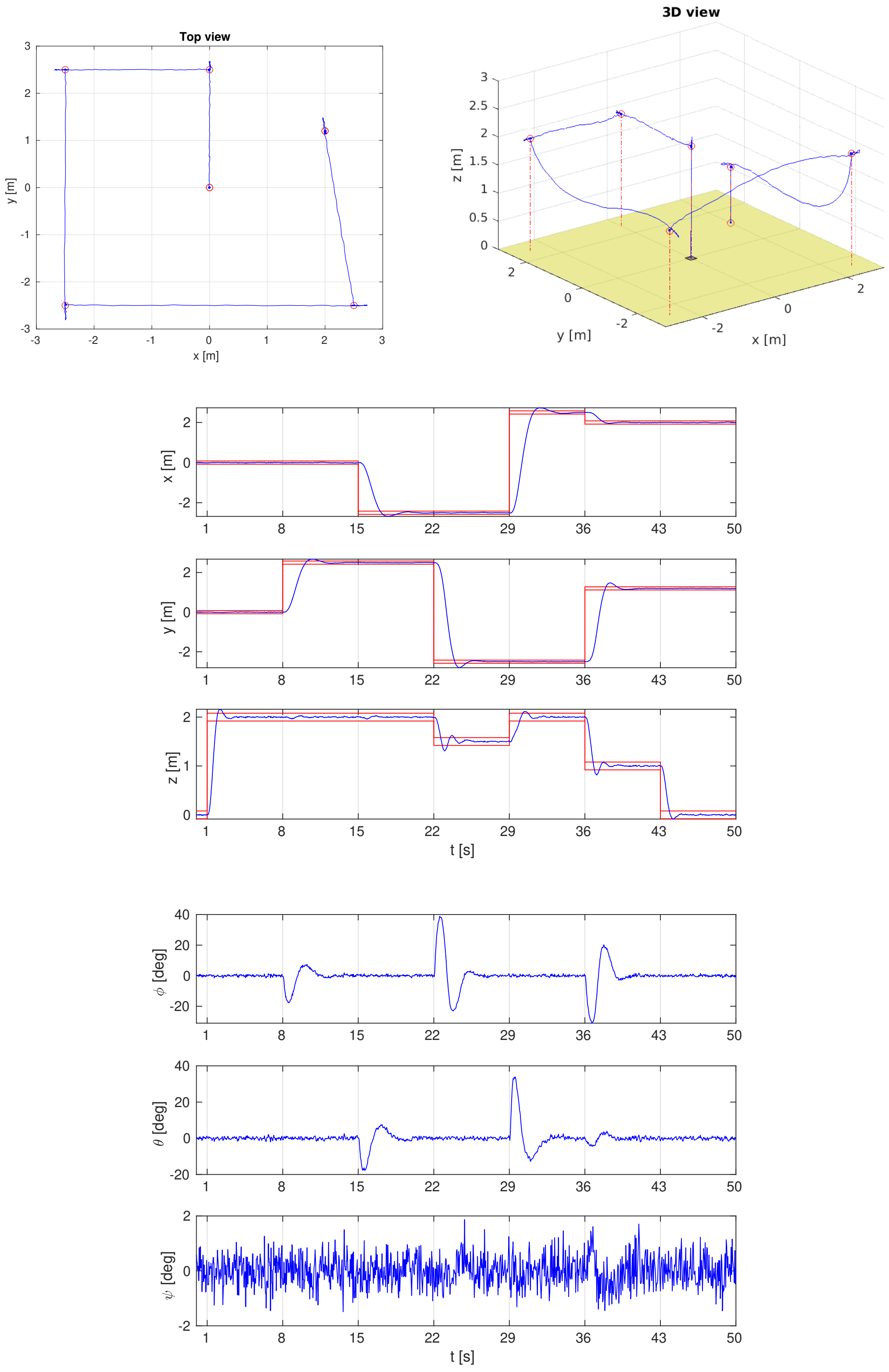


Figure 9: Simulation results of the **LQG** controller with integral action without payload.

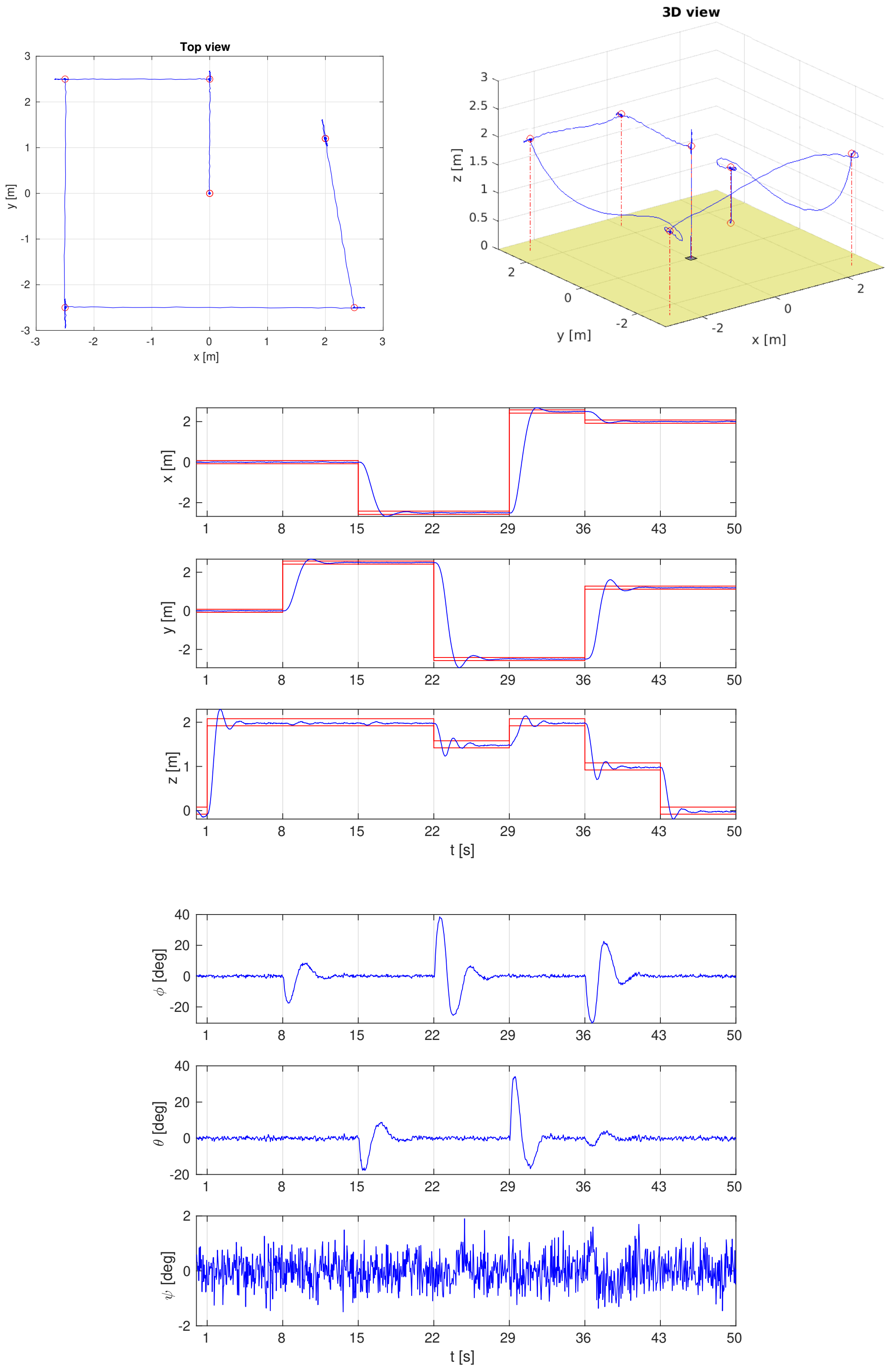


Figure 10: Simulation results of the **LQG** controller with integral action with a payload of 0.1 kg.

5 State feedback design via Pole Placement

To choose the poles of the controller, we use requirements on the second order behaviour of the system. In particular, we want the settling time to be $t_s = 5$ s. The damping ratio is chosen by considering we want little overshoot. This means the damping ratio should be close to 1[3]. By tuning and looking at simulation results, we choose $\zeta = 0.9$.

The 2 complex poles $a \pm bi$ can then be determined by using the identities $t_s = \frac{4.6}{\zeta\omega_n}$, $a = -\zeta\omega_n$ and $b = \omega_n\sqrt{1-\zeta^2}$. Initially, another 10 poles were added on the real axis to reach the required number of poles. We chose 10 different poles in the range $[10a, 8a]$, which matches the rule of thumb to choose them equal to $5\times$ to $10\times$ the real part of the complex poles. However, the simulation results were not satisfactory. By looking at the closed-loop poles of the LQR controller from Section 3.1, we figured additional complex poles could help. Hence we used the complex poles $a \pm bi$ and also added poles $1.25(a \pm bi)$ (the closed-loop poles from Section 3.1 matched this pattern). The remaining 8 poles are still chosen in the range $[10a, 8a]$.

Once we have determined the controller poles in the s-domain, we transform them to poles in the z-domain by using the identity $z = \exp(sT_s)$. The poles of the state estimator are chosen as $2\times$ the controller poles in the s-domain (the rule of thumb is $2\times$ to $5\times$ in this case).

As an additional check, we verify that the compensator is stable. This should be the case as we do not want the inputs to explode. We compute the eigenvalues (cf. p. 210 in course notes).

```
M = A - B*K - L*C + L*D*K;
eig(M)
```

We conclude the compensator is stable since the eigenvalues are all located within the unit circle.

The Simulink diagram of this controller can be found in Figure 11. Simulation results with and without payload are given in Figures 12 and 13. We compare the results with the LQR controller from Section 3.1 (Figures 3 and 4). The 3D trajectory of the controller using pole placement is less smooth. The quadcopter tends to make movements in the shape of an arch, especially when viewed from the top. This causes the quadcopter to come closer to the walls of the room. Finally, the simulations with a payload also show a steady-state error in the z-coordinate. We conclude the controller using pole placement is qualitatively similar to the LQR controller from Section 3.1.

In the assignment it is also asked to compare this controller with the LQG controller. This comparison is largely similar to the comparison between the standard LQR controller and the LQR controller with integral action from Section 3. The controller using pole placement relies heavily on an accurate model of the system. Since the payload is a disturbance of the model, the controller performs poorly in the presence of a payload. On the other hand, the LQG controller has an integral action. Hence, it acts to get rid of a steady-state error. We conclude the LQG controller is preferable since it is more robust.

[3]When applying a step input, the overshoot M_p (which is the maximal value of the response minus one) is equal to $\exp\left(\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}\right)$.

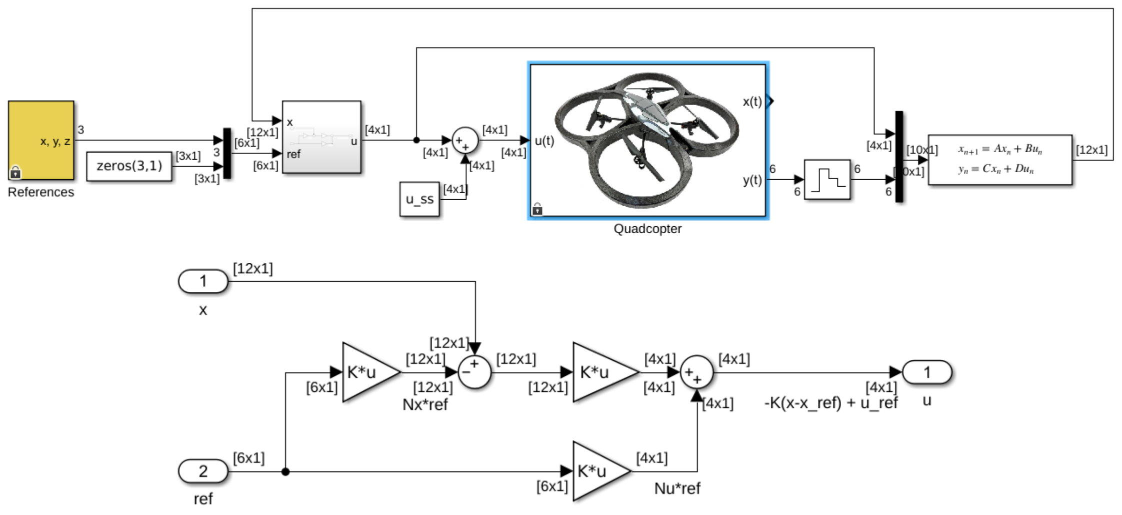


Figure 11: Simulink diagram of the full-state feedback controller using **pole placement**. The lower figure is the subsystem from the upper figure.

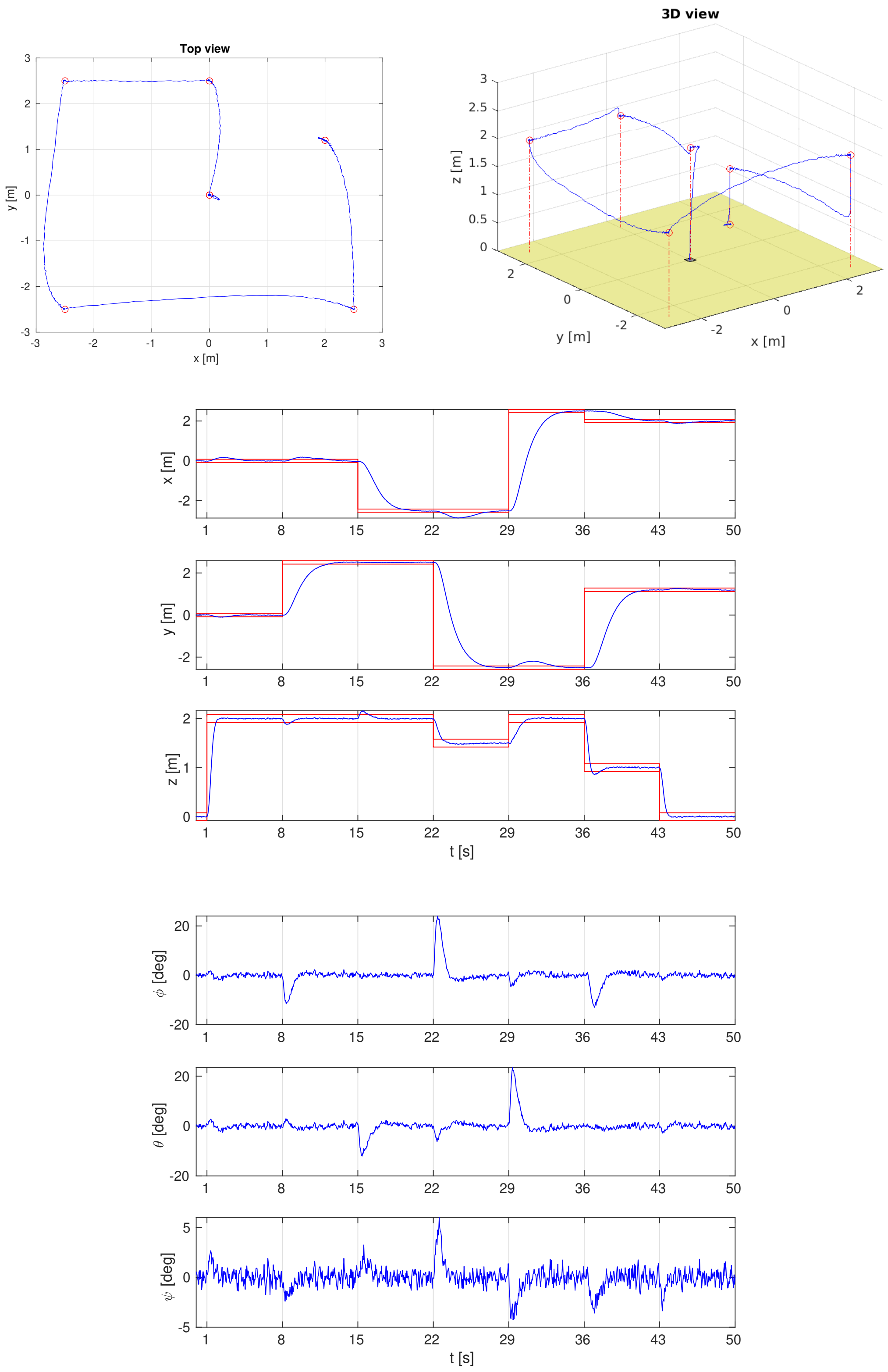


Figure 12: Simulation results of the full-state feedback controller using **pole placement** without payload.

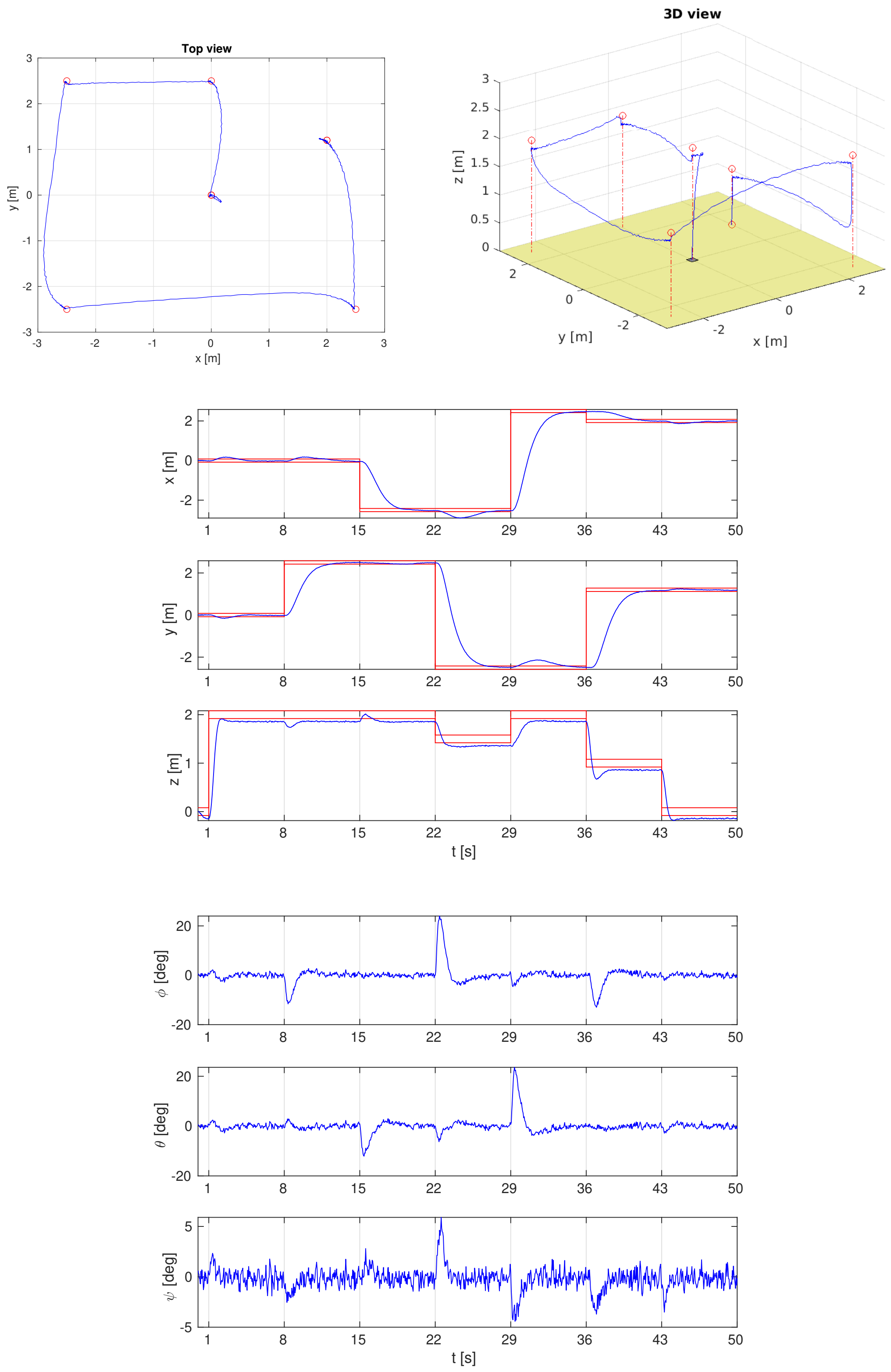


Figure 13: Simulation results of the full-state feedback controller using **pole placement** with a payload of 0.1 kg.