Katholieke
Universiteit
Leuven

**Department of
Electrical Engineering**

# COMPUTERGESTUURDE REGELTECHNIEK: QUADCOPTER

**De Weer Sofie - r0802671**

**Pollet Max - r0757781**

Academic Year 2022–2023

# Contents

# List of Figures

# List of Tables

# Introduction

This reports will analyse different control strategies to control a quadcopter that has to fly through a certain track. Due to the controller needing to follow a course will two strategies be used to eliminate the steady state error, they are full state feedback and integral control. Both have their advantages and disadvantages.

The system is considered when it returns its state vector and also the case when it doesn't. This will result in different control strategies. Such as, when the state vector is available will LQR be used. While when there is no state vector available will LQG and pole placement be used to make the controller.

To simulate the system with the different control strategies are Simulink and MATLAB used. Appendix A describes the use of each file that was provided with the report.

# Chapter 1

# Linearization

## 1.1 Finding the equilibrium point

The following nonlinear system is given:

$$
\begin{aligned}
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{z} &= v_z \\
\dot{v}_x &= -\frac{k_d}{m} v_x + \frac{k c_m}{m} \left(\sin\psi \sin\phi + \cos\psi \cos\phi \sin\theta\right) \left(v_1^2 + v_2^2 + v_3^2 + v_4^2\right) \\
\dot{v}_y &= -\frac{k_d}{m} v_y + \frac{k c_m}{m} \left(\cos\phi \sin\psi \sin\theta - \cos\psi \sin\phi\right) \left(v_1^2 + v_2^2 + v_3^2 + v_4^2\right) \\
\dot{v}_z &= -\frac{k_d}{m} v_z - g + \frac{k c_m}{m} \left(\cos\theta \cos\phi\right) \left(v_1^2 + v_2^2 + v_3^2 + v_4^2\right) \\
\dot{\phi} &= \omega_x + \omega_y(\sin\phi \tan\theta) + \omega_z(\cos\phi \tan\theta) \\
\dot{\theta} &= \omega_y \cos\phi - \omega_z \sin\phi \\
\dot{\psi} &= \frac{\sin\phi}{\cos\theta}\omega_y + \frac{\cos\phi}{\cos\theta}\omega_z \\
\dot{\omega}_x &= \frac{L k c_m}{I_{xx}}(v_1^2 - v_2^2) - \left(\frac{I_{yy} - I_{zz}}{I_{xx}}\right) \omega_y \omega_z \\
\dot{\omega}_y &= \frac{L k c_m}{I_{yy}}(v_2^2 - v_4^2) - \left(\frac{I_{zz} - I_{xx}}{I_{yy}}\right) \omega_x \omega_y \\
\dot{\omega}_z &= \frac{L k c_m}{I_{zz}}(v_1^2 - v_2^2 + v_3^2 - v_4^2) - \left(\frac{I_{xx} - I_{yy}}{I_{zz}}\right) \omega_x \omega_y
\end{aligned}
\tag{1.1}
$$

This can also be written in the following state space description as seen in the exercise description:

$$
\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{\xi}(\boldsymbol{x}, \boldsymbol{u}) \\
\boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}
\end{aligned}
\tag{1.2}
$$

The parameters can be found in table 1.1.

| Parameter | Symbol | Value | Unit |
|---|---|---|---|
| Mass of the quadcopter | $m$ | 0.5 | $kg$ |
| Radius of the quadcopter | $L$ | 0.25 | $m$ |
| Propeller lift coefficient | $k$ | $3 \times 10^{-6}$ | $N\ s^2$ |
| Propeller drag coefficient | $b$ | $1 \times 10^{-7}$ | $N\ m\ s^2$ |
| Acceleration of gravity | $g$ | 9.81 | $^m/_{s^2}$ |
| Air friction coefficient | $k_d$ | 0.25 | $^{kg}/_s$ |
| Quadcopter inertia about the $x^b$-axis | $I_{xx}$ | $5 \times 10^{-3}$ | $kg\ m^2$ |
| Quadcopter inertia about the $y^b$-axis | $I_{yy}$ | $5 \times 10^{-3}$ | $kg\ m^2$ |
| Quadcopter inertia about the $z^b$-axis | $I_{zz}$ | $1 \times 10^{-2}$ | $kg\ m^2$ |
| Motor constant | $c_m$ | $1 \times 10^4$ | $v^{-2}\ s^{-2}$ |

Table 1.1: Parameters of the quadcopter model

This model must be linearized around a certain equilibrium point such that the following state space description holds:

$$\begin{aligned} \Delta\dot{\boldsymbol{x}} &= \boldsymbol{A}\Delta\boldsymbol{x} + \boldsymbol{B}\Delta\boldsymbol{u} \\ \Delta\boldsymbol{y} &= \boldsymbol{C}\Delta\boldsymbol{x} \end{aligned} \tag{1.3}$$

With $\boldsymbol{A} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}^*,\boldsymbol{u}^*}$ and $\boldsymbol{B} = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{u}}\Big|_{\boldsymbol{x}^*,\boldsymbol{u}^*}$ the linearized state matrices,
$\Delta\boldsymbol{x} = \begin{bmatrix}\Delta x & \Delta y & \Delta z & \Delta v_x & \Delta v_y & \Delta v_z & \Delta\phi & \Delta\theta & \Delta\psi & \Delta\omega_x & \Delta\omega_y & \Delta\omega_z\end{bmatrix}^T = \boldsymbol{x} - \boldsymbol{x}^*$ the state,
$\Delta\boldsymbol{u} = \begin{bmatrix}\Delta v_1^2 & \Delta v_2^2 & \Delta v_3^2 & \Delta v_4^2\end{bmatrix}^T = \boldsymbol{u} - \boldsymbol{u}^*$ the input and $\Delta\boldsymbol{y} = \begin{bmatrix}\Delta x & \Delta y & \Delta z & \Delta\phi & \Delta\theta & \Delta\phi\end{bmatrix}^T = \boldsymbol{y} - \boldsymbol{y}^*$ the output.

Before the linearization can take place must the equilibrium point where the output $\boldsymbol{y}^* = \boldsymbol{0}$ be found. Thus solving the steady state equations with this output will result in the equilibrium point.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{\xi}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{0} \\ \boldsymbol{y}^* &= \boldsymbol{C}\boldsymbol{x} = \boldsymbol{0} \\ &\Downarrow \\ \boldsymbol{x}^* &= \boldsymbol{0} \\ \boldsymbol{u}^* &= \frac{gm}{4kc_m}\boldsymbol{1}_{4\times 1} = 40.875 \cdot \boldsymbol{1}_{4\times 1} \end{aligned} \tag{1.4}$$

Linearizing around this equilibrium point results in the following linear system:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{k_d}{m} & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{k_d}{m} & 0 & -g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{k_d}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{c_m k}{m} & \frac{c_m k}{m} & \frac{c_m k}{m} & \frac{c_m k}{m} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{L c_m k}{I_{xx}} & 0 & -\frac{L c_m k}{I_{xx}} & 0 \\ 0 & \frac{L c_m k}{I_{yy}} & 0 & -\frac{L c_m k}{I_{yy}} \\ \frac{b c_m}{I_{zz}} & -\frac{b c_m}{I_{zz}} & \frac{b c_m}{I_{zz}} & -\frac{b c_m}{I_{zz}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.06 & 0.06 & 0.06 & 0.06 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.5 & 0 & -1.5 & 0 \\ 0 & 1.5 & 0 & -1.5 \\ 0.1 & -0.1 & 0.1 & -0.1 \end{bmatrix}$$

$$C = \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{I}_3 & \boldsymbol{0}_3 \end{bmatrix}$$

$$\boldsymbol{D} = \boldsymbol{0}_{6\text{x}4}$$

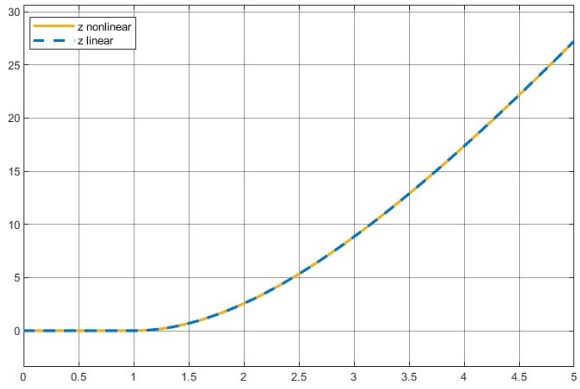
The step response of the following input

$$\Delta \boldsymbol{u} = \begin{cases} 0 & t \leq 1 \\ 25 & t > 1 \end{cases}$$

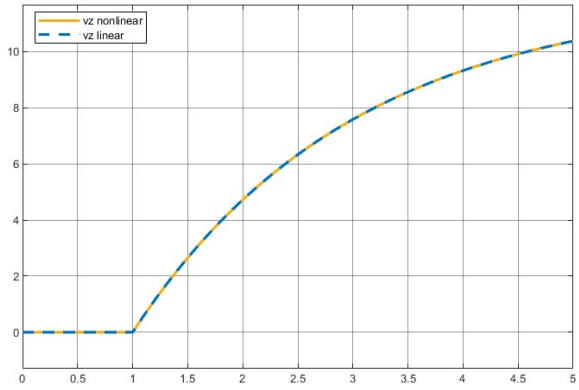
will only cause the states $z$ and $v_z$ to change over time. The linear and nonlinear model are plotted against each other in figure 1.1a for $z$ and 1.1b for $v_z$. Both figures show that the linear model follows the nonlinear model very accurately. No steady state error is visible. The linearized system has 12 poles which are displayed in table 1.2.

| pole | value | multiplicity |
|---|---|---|
| $\lambda_1$ | 0 | 9 |
| $\lambda_2$ | $-0.5$ | 3 |

Table 1.2: Poles of the linearized system




(a) Step response of the $z$-state of the nonlinear and linear model




(b) Step response of the $v_z$-state of the nonlinear and linear model

Figure 1.1: Step response of the nonlinear and linear system

# Chapter 2

# Discretisation

The controller of the quadcopter is in reality not a continuous system but a discrete system with a sampling time of $T_s = 0.05$ seconds. The existing continuous system has to be transformed in to a discretized system. There are multiple discretisation methods and each method influences the poles, controllability and observability differently. The different methods that are compared are Zero-hold, Forward Euler and Tustin (or bilinear rule). Each discretisation resulted in unstable poles and a controllable, observable system. The Euler Discretization has 4 transmission zeros, zero order hold has 2 transmission zeros ans Tustin has only 1 zero transmission zeros as shown in table 2.1.

| Euler | ZOH | Tustin |
|---|---|---|
| -3.7321 | -0.9917 | -1 |
| -3.7088 | -1.0000 | |
| -0.2679 | | |
| -0.2663 | | |

Table 2.1: Transmission zeros of the different kind of discretization methods

The method that is used is the Tustin transformation due to the transformation preserving stability of the system by mapping the stable left-half s-plane to the stable unit circle in the z-plane. This method uses the following transformation:

$$s \leftarrow \frac{2}{T_s}\frac{z-1}{z+1}$$

Using the Tustin discretization provides full controlability and observability with 9 poles at 1 and 3 poles at 0.9753, this concludes that the discrete system is unstable (marginally stable) and minimal. Furthermore will the system be fully stabilizable and detectable, which is needed for designing a controller and observer. Figure 2.1 shows the poles (cross) and zeros (circle) of the discretized system.

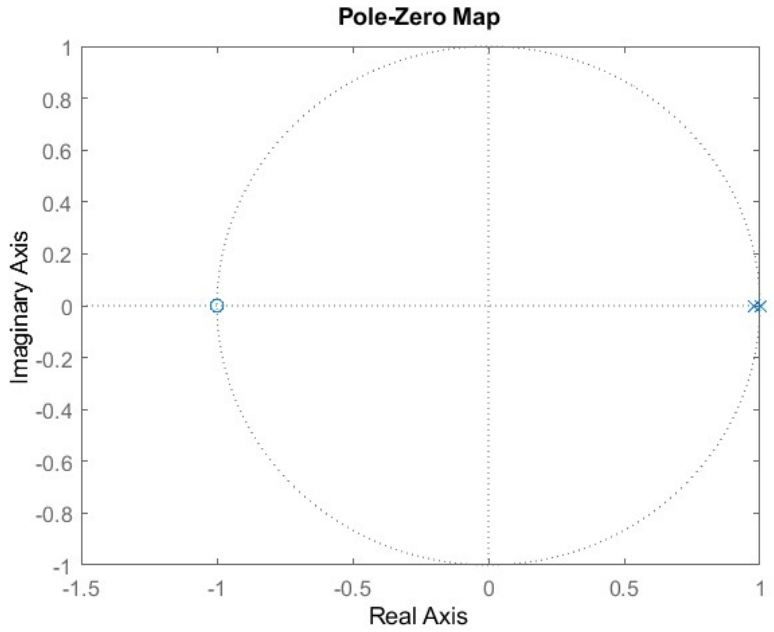



Figure 2.1: Poles and zeros of the discretized system

The obtained system matrices are

$$A_d = \begin{bmatrix} 1 & 0 & 0 & 0.0494 & 0 & 0 & 0 & 0.0121 & 0 & 0 & 3.0278 \times 10^{-4} & 0 \\ 0 & 1 & 0 & 0 & 0.0494 & 0 & -0.0121 & 0 & 0 & -3.0278 \times 10^{-4} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.0494 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9753 & 0 & 0 & 0 & 0.4844 & 0 & 0 & 0.0121 & 0 \\ 0 & 0 & 0 & 0 & 0.9753 & 0 & -0.4844 & 0 & 0 & -0.0121 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9753 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.05 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0 & 1.1354 \times 10^{-5} & 0 & -1.1354 \times 10^{-5} \\ -1.1354 \times 10^{-5} & 0 & 1.1354 \times 10^{-5} & 0 \\ 7.4074 \times 10^{-5} & 7.4074 \times 10^{-5} & 7.4074 \times 10^{-5} & 7.4074 \times 10^{-5} \\ 0 & 4.5417 \times 10^{-4} & 0 & -4.5417 \times 10^{-4} \\ -4.5417 \times 10^{-4} & 0 & 4.5417 \times 10^{-4} & 0 \\ 0.003 & 0.003 & 0.003 & 0.003 \\ 0.0019 & 0 & -0.0019 & 0 \\ 1.25 \times 10^{-4} & -1.25 \times 10^{-4} & 1.25 \times 10^{-4} & -1.25 \times 10^{-4} \\ 0.075 & 0 & -0.075 & 0 \\ 0 & 0.075 & 0 & -0.075 \\ 0.005 & -0.005 & 0.005 & -0.005 \end{bmatrix}$$

$$C_d = \begin{bmatrix} 1 & 0 & 0 & 0.0247 & 0 & 0 & 0 & 0.0061 & 0 & 0 & 1.5139 \times 10^{-4} & 0 \\ 0 & 1 & 0 & 0 & 0.0247 & 0 & -0.0061 & 0 & 0 & -1.5139 \times 10^{-4} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.0247 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.025 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.025 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0.025 \end{bmatrix}$$

$$D_d = \begin{bmatrix} 0 & 5.6771 \times 10^{-6} & 0 & -5.6771 \times 10^{-6} \\ -5.6771 \times 10^{-6} & 0 & 5.6771 \times 10^{-6} & 0 \\ 3.7037 \times 10^{-5} & 3.7037 \times 10^{-5} & 3.7037 \times 10^{-5} & 3.7037 \times 10^{-5} \\ 9.3750 \times 10^{-4} & 0 & -9.3750 \times 10^{-4} & 0 \\ 0 & 9.3750 \times 10^{-4} & 0 & -9.3750 \times 10^{-4} \\ 6.25 \times 10^{-5} & -6.25 \times 10^{-5} & 6.25 \times 10^{-5} & -6.25 \times 10^{-5} \end{bmatrix}$$

# Chapter 3

# LQR

The first controller that will be designed is an LQR controller. Since the quadcopter must follow a certain path in space will the controller first be implemented with a full state feedback scheme and afterwards an integral controller scheme. The assumption in this part is that the full state vector is fully available.

## 3.1 Full state feedback

### 3.1.1 Design process

The full state feedback scheme is able to track the reference $r(k)$ by modifying the feedback law as follows

$$\boldsymbol{u}(k) = -K(\boldsymbol{x}(k) + N_x\boldsymbol{r}(k)) + N_u\boldsymbol{r}(k)$$

This kind of a control law will ensure there is no steady state error to a step input $\boldsymbol{r}(k)$. Inserting this control law into the state space description and ensuring that the system is in steady state will result in the following equation to retrieve the matrices $N_x$ and $N_u$.

$$\underbrace{\begin{bmatrix} A_d - I & B_d \\ C_d & D_d \end{bmatrix}}_{\mathbb{R}^{18\times 16}} \underbrace{\begin{bmatrix} N_x \\ N_u \end{bmatrix}}_{\mathbb{R}^{16\times 6}} = \begin{bmatrix} \mathbf{0}_{12\times 6} \\ \boldsymbol{I}_6 \end{bmatrix} \tag{3.1}$$

This results in the matrices $N_u$ and $N_x$ being

$$N_u = 0_{4\times 6}$$

$$N_x = \begin{bmatrix} 1 & 0 & 0 & 0 & -0.0823 & 0 \\ 0 & 1 & 0 & 0.0823 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.1261 & 0 \\ 0 & 0 & 0 & -3.1261 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8386 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.8386 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since there are 4 inputs and 6 outputs is the system in equation 3.1 overdetermined and will this system only have the least square solution as its solution. An important result from this is that the zero steady state error is not longer guaranteed since there is a lack of degrees of freedom to control all outputs. The new system with controller looks now as shown in figure 3.1 .

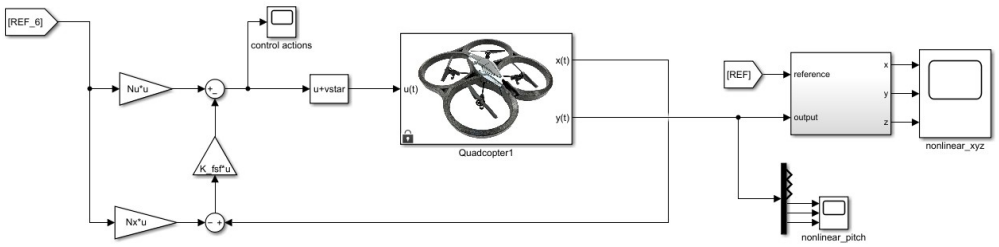



Figure 3.1: Diagram of the full state feedback system

The LQR controller reduces the quadratic cost function depicted in equation 3.2. The Q and R matrices are self determined. $Q$ and $R$ are iteratively chosen such that the quadcopter follows the reference as close as possible. Thus changing the values in $Q$ will have an effect in the states reaching the reference and changing the values in $R$ will have an influence on the control actions. Generally will the values in $Q$ be high to make sure that the steady state error stays small. For $R$ can't the values be too high or too small since both will lead to unsatisfactory results. Too small will lead to saturation of the actuators while too high will lead to controller taking no action.
Due to the system having to follow a certain path in space will the states $x$, $y$ and $z$ be the most important and consequently have the highest weight in $Q$. The other components don't matter too much as long as they don't make the system unstable. The chosen matrices $Q$ and $R$ are depicted in equation 3.3.

$$\sum_{n=0}^{\infty} \boldsymbol{x}_k^T Q \boldsymbol{x}_k + \boldsymbol{u}_k^T R \boldsymbol{u}_k \tag{3.2}$$

$$\begin{aligned} Q_{\text{state feedback}} &= \begin{bmatrix} 50 & 0 & 0 & \mathbf{0}_{1\times 9} \\ 0 & 50 & 0 & \mathbf{0}_{1\times 9} \\ 0 & 0 & 400 & \mathbf{0}_{1\times 9} \\ \mathbf{0}_{9\times 1} & \mathbf{0}_{9\times 1} & \mathbf{0}_{9\times 1} & 10\boldsymbol{I}_9 \end{bmatrix} \in \mathbb{R}^{12\times 12} \\ R_{\text{state feedback}} &= 0.1\boldsymbol{I}_4 \in \mathbb{R}^{4\times 4} \end{aligned} \tag{3.3}$$

From this will the control gain $K$ be computed with the command *dlqr()* to ensure that the discrete time form is used. The resulting gain is stated in equation 3.4.

$$K = \begin{bmatrix} 0 & -8.2355 & 28.8971 & 0 & -6.5474 & 14.2971 & 26.1472 & 0 & 4.6588 & 5.4723 & 0 & 6.7081 \\ 8.2355 & 0 & 28.8971 & 6.5474 & 0 & 14.2971 & 0 & 26.1472 & -4.6588 & 0 & 5.4723 & -6.7081 \\ 0 & 8.2355 & 28.8971 & 0 & 5.5474 & 14.2971 & -26.1472 & 0 & 4.6588 & -5.4723 & 0 & 6.7081 \\ -8.2355 & 0 & 28.8971 & -6.5474 & 0 & 14.2971 & 0 & -26.1472 & -4.6588 & 0 & -5.4723 & -6.7081 \end{bmatrix} \tag{3.4}$$

The closed-loop poles of the new system are given in table 3.1. All poles in the system are in the unit circle and thus is the closed-loop system stable.

| pole | value | multiplicity |
|---|---|---|
| $\lambda_1$ | 0.9506 | 1 |
| $\lambda_2$ | 0.9007 | 2 |
| $\lambda_{3,4}$ | $0.8932 \pm 0.1104i$ | 2 |
| $\lambda_{5,6}$ | $0.7737 \pm 0.1536i$ | 1 |
| $\lambda_7$ | 0.7329 | 1 |
| $\lambda_8$ | 0.076 | 2 |

Table 3.1: Poles of the closed loop system with full state feedback

### 3.1.2 Simulation without a load

When the controller is tested for the given reference will this result in figures 3.2-3.4, which show the position, the control actions and the pitch angles during simulation. Figure 3.3 shows that the input limits are reached for

a split second so saturation is not a problem. Figures 3.2 and 3.4 show the positions and pitch of the quadcopter respectively. As seen in the figures will the quadcopter reach the seven checkpoints in time. The average time to reach a checkpoint is 1.643 seconds.

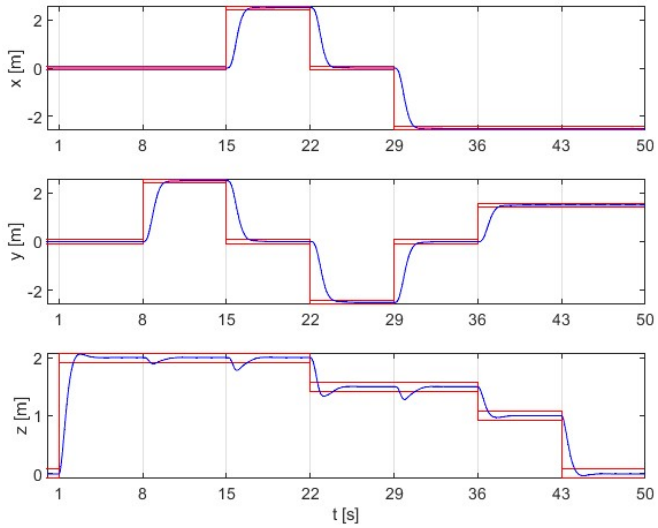



Figure 3.2: $x$, $y$ and $z$ position of the full state feedback controller

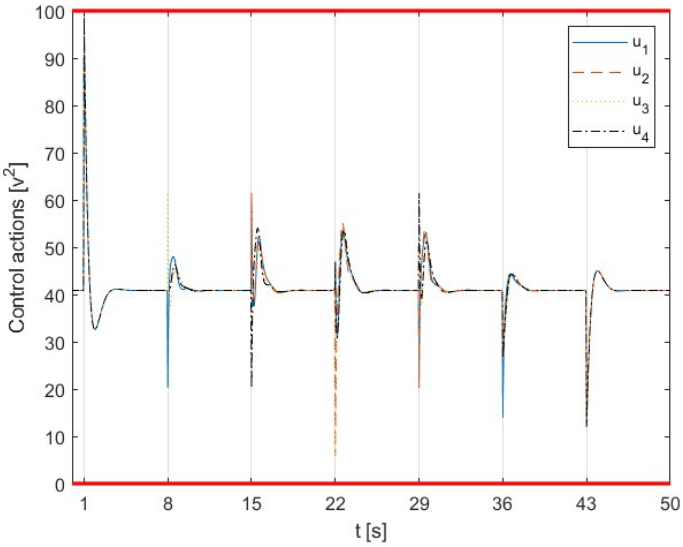



Figure 3.3: Control actions of the full state feedback controller

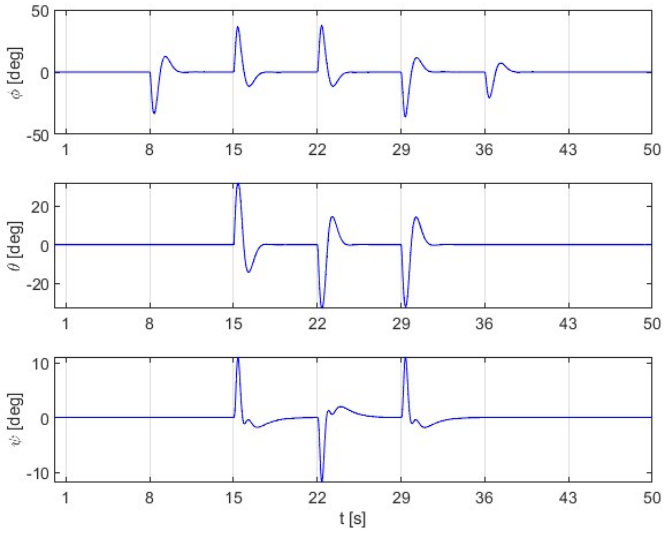



Figure 3.4: $\phi$, $\theta$ and $\psi$ angles of the full state feedback controller

### 3.1.3 Simulation with load

When adding a load to the initial controller will the system now fail to reach the checkpoints as seen in figure 3.5. The reason for this steady state error comes from the fact that the system has more outputs than inputs. This resulted in the system in equation 3.1 to be overdetermined and thus was a zero steady state error not guaranteed.

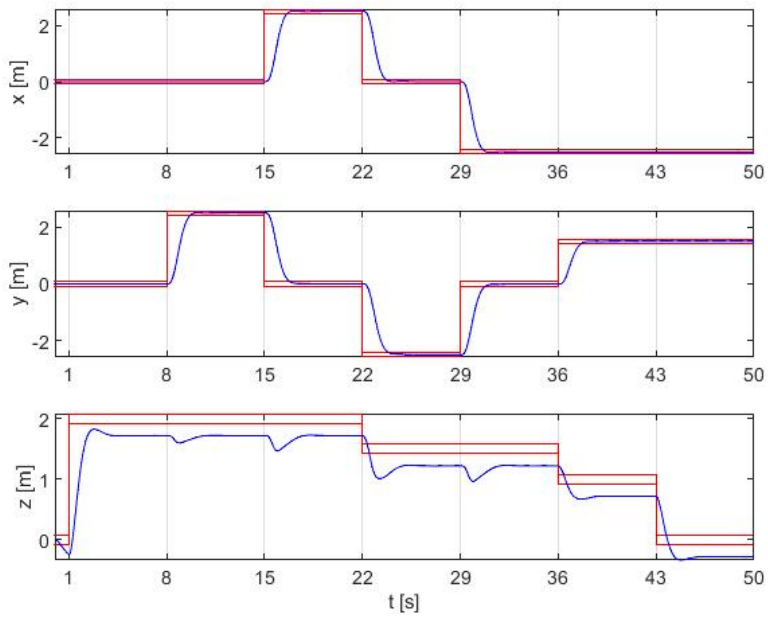



Figure 3.5: Simulation with previous full state feedback controller

When adjusting the matrices $Q$ and $R$ to the ones in equation 3.5, will the system now be able to reach the checkpoints when there is a load of 0.1 kg. Figures 3.6 and 3.8 show the position and angles of simulation of the augmented controller with load. The system now needs an average time of $1.521s$ to reach each checkpoint.

Since the values in the $R$ matrix are smaller than before will this mean that the control actions are higher than before. Figure 3.7 shows that the control actions indeed saturate the actuators more than before.

$$Q_{load} = \begin{bmatrix} 50 & 0 & 0 & \mathbf{0}_{1\times 9} \\ 0 & 50 & 0 & \mathbf{0}_{1\times 9} \\ 0 & 0 & 1000 & \mathbf{0}_{1\times 9} \\ \mathbf{0}_{9\times 1} & \mathbf{0}_{9\times 1} & \mathbf{0}_{9\times 1} & 10\boldsymbol{I}_9 \end{bmatrix} \tag{3.5}$$
$$R_{load} = 0.01\boldsymbol{I}_4$$

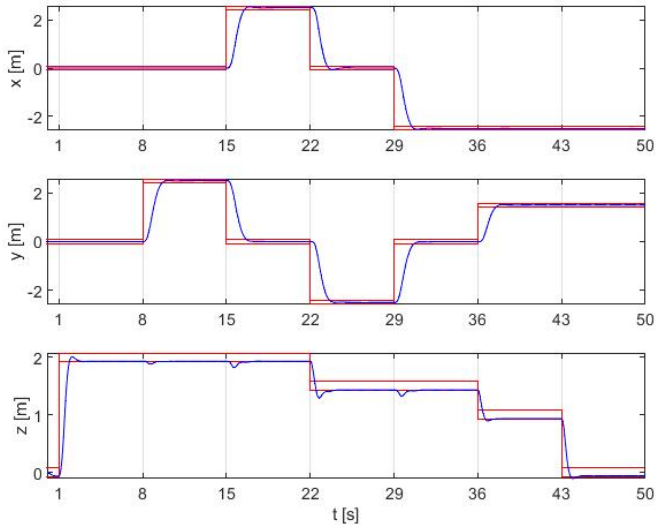



Figure 3.6: $x$, $y$ and $z$ position of the full state feedback controller with load

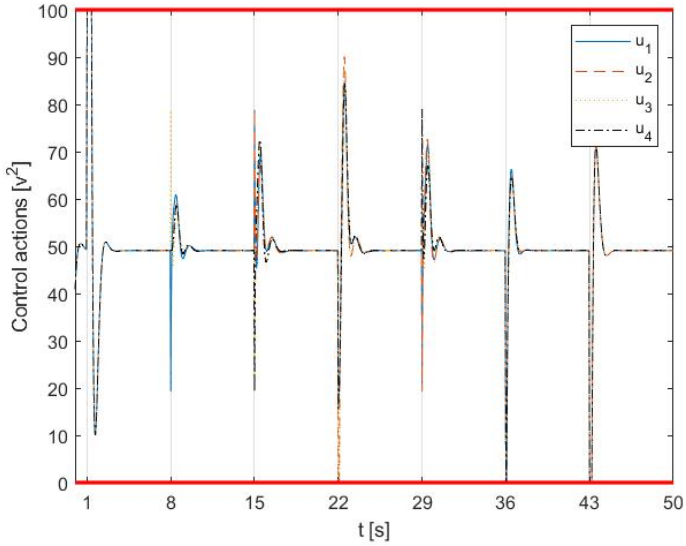



Figure 3.7: Control actions of the full state feedback controller with load

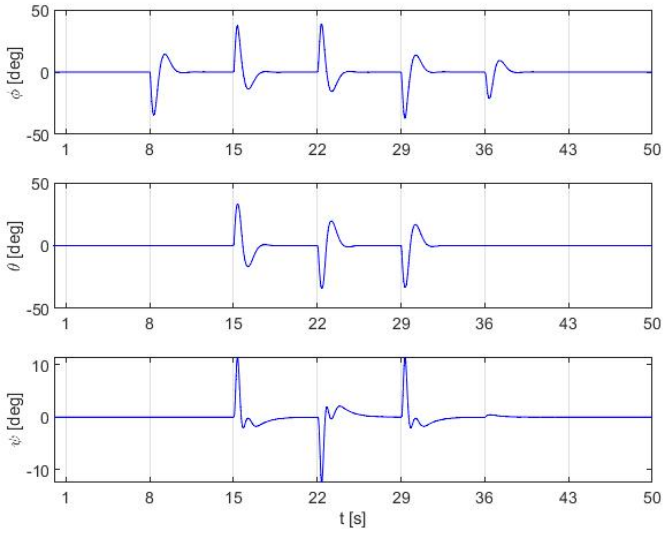



Figure 3.8: $\phi$, $\theta$ and $\psi$ angles of the full state feedback controller with load

## 3.2 Integral control

### 3.2.1 Design process

Since only $x$, $y$ and $z$ need to be tracked must the controller only add three integrators. In fact, since there are only four inputs can the controller only have a maximum of four integrators. Thus the system is augmented to add these new integrators. This will lead to the new system matrices which are stated in equation 3.6. The matrices there are obviously the system matrices of the discretized system.

$$\begin{aligned} A_{\text{integral}} &= \begin{bmatrix} \boldsymbol{I}_3 & C_d(1:3,:) \\ \boldsymbol{0}_{12\times 3} & A_d \end{bmatrix} \\ B_{\text{integral}} &= \begin{bmatrix} D_d(1:3,:) \\ B_d \end{bmatrix} \end{aligned} \tag{3.6}$$

This new system has 15 states due to the three new states from the integrator. This means that the controllability matrix should have rank 15. When performing the check in MATLAB will this be the case. This will also lead to the control law needing to be updated to the following

$$\boldsymbol{u}(k) = -\begin{bmatrix} K_I & K_s \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_I(k) \\ \boldsymbol{x}(k) \end{bmatrix}$$

With $K_I$ and $\boldsymbol{x}_I$ the gain and states of the three integrators and $K_s$ and $\boldsymbol{x}$ the gain and states of the original system.

The $Q$ and $R$ matrices for the LQR controller are chosen in such a way that the controller will act fast on changes in the reference and will also penalize big errors between the desired state and the reference. A fast controller can be achieved by making the values in $R$ small, and having a small steady state error can be obtained by making the values of $Q$ at the reference states big. The result is summarized in equation 3.7.

$$\begin{aligned} Q_{integral} &= \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_{3\times 1} & \boldsymbol{0}_{3\times 1} & \boldsymbol{0}_{3\times 1} & \boldsymbol{0}_{3x9} \\ \boldsymbol{0}_{1x3} & 50 & 0 & 0 & \boldsymbol{0}_{1x9} \\ \boldsymbol{0}_{1x3} & 0 & 50 & 0 & \boldsymbol{0}_{1x9} \\ \boldsymbol{0}_{1x3} & 0 & 0 & 350 & \boldsymbol{0}_{1x9} \\ \boldsymbol{0}_{9x3} & \boldsymbol{0}_{9\times 1} & \boldsymbol{0}_{9\times 1} & \boldsymbol{0}_{9\times 1} & \boldsymbol{I}_9 \end{bmatrix} \in \mathbb{R}^{15x15} \\ R_{integral} &= (1\times 10^{-3})\boldsymbol{I}_4 \in \mathbb{R}^{4x4} \end{aligned} \tag{3.7}$$

Finding the correct values for both matrices is a case of trial and error. Since these matrices were already obtained in the previous controller which had a full state feedback controller, was the best choice to start with the same values for the new $Q$ and $R$. The initial values already gave a result that was satisfactory, but the outcome could be better. The augmented system with the controller is shown in figure 3.9. As can be seen will this system integrate the difference between first three outputs and its references of the system which will then be multiplied with the control gain $-K_I$. This results in the controller being more robust for steady state errors, which was not the case for the full state feedback controller.

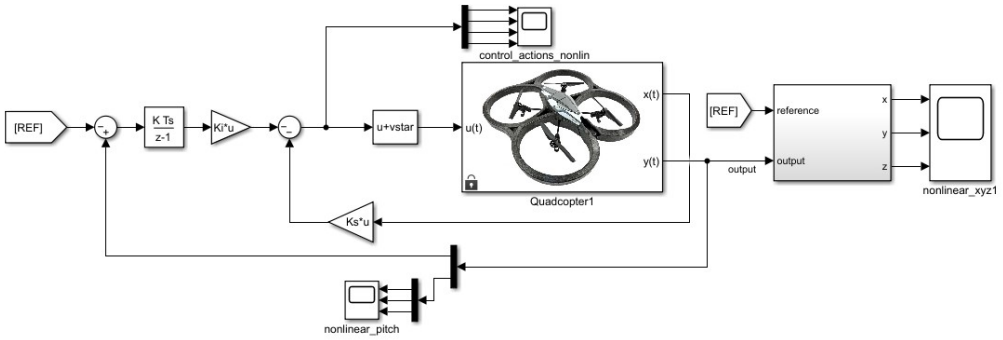



Figure 3.9: The augmented system with the controller

The gain matrices $K_I$ and $K_s$ are obtained via the command *dlqr()* which will return

$$K_I = \begin{bmatrix} 0 & -4.7753 & 11.4302 \\ 1.7753 & 0 & 11.4302 \\ 0 & 4.7753 & 11.4302 \\ -4.7753 & 0 & 11.4302 \end{bmatrix}$$

$$K_s = \begin{bmatrix} 0 & -85.3323 & 259.8984 & 0 & -28.9353 & 45.6408 & 63.1062 & 0 & 13.1972 & 7.7735 & 0 & 15.4968 \\ 85.3323 & 0 & 259.8984 & 28.9353 & 0 & 45.6408 & 0 & 63.1062 & -13.1972 & 0 & 7.7735 & -15.4968 \\ 0 & 85.3323 & 259.8984 & 0 & 28.9353 & 45.6408 & -63.1062 & 0 & 13.1972 & -7.7735 & 0 & 15.4968 \\ -85.3323 & 0 & 259.8984 & -28.9353 & 0 & 45.6408 & 0 & -63.1062 & -13.1972 & 0 & -7.7735 & -15.4968 \end{bmatrix}$$

The closed-loop system will now also have 15 poles which are shown in table 3.2. Since all poles are inside the unit circle will the closed-loop system be stable.

| pole | | multiplicity |
|---|---|---|
| $\lambda_1$ | 0.9506 | 1 |
| $\lambda_2$ | 0.9479 | 1 |
| $\lambda_{3,4}$ | $0.8823 \pm 0.1666i$ | 2 |
| $\lambda_{5,6}$ | $0.8519 \pm 0.025i$ | 2 |
| $\lambda_{7,8}$ | $0.7039 \pm 0.2055i$ | 1 |
| $\lambda_9$ | 0.7329 | 1 |
| $\lambda_{10}$ | 0.076 | 2 |

Table 3.2: Closed loop poles of the system with integral control

### 3.2.2 Simulation without load

Figure 3.10 shows the $x$, $y$ and $z$ position of the nonlinear system when applying a reference. As seen in this figure will the quadcopter reach the desired positions in the correct time. Another important aspect of designing the controller was making sure that the control actions don't saturate the inputs. These actions are influenced by the values in the $R$ matrix seen in equation 3.7. The control actions are shown in figure 3.11 and show that they stay inside the interval $[0, 100]V$. Since the output also consists of the pitch can they also be plotted for the duration of the simulation, they are shown in figure 3.12.
The quadcopter reaches every checkpoint in time with an average time of $t = 2.007s$.

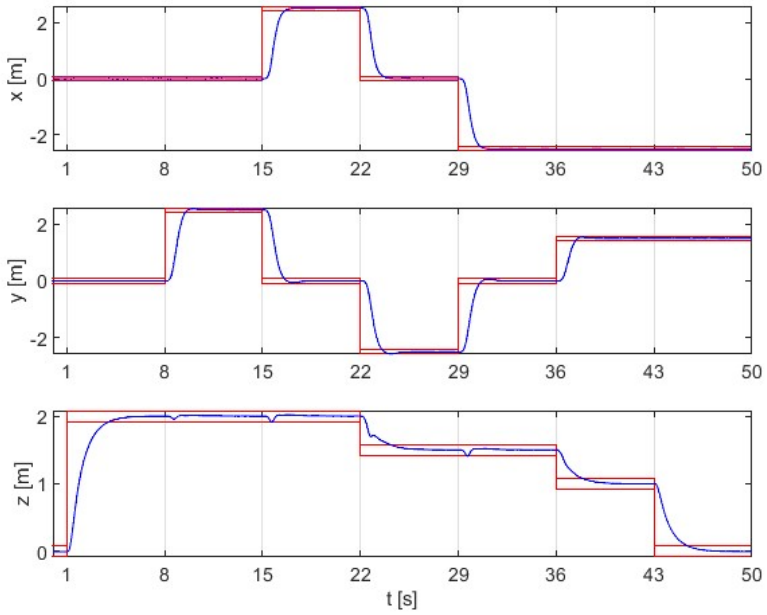



Figure 3.10: $x$, $y$ and $z$ position of the integral controller

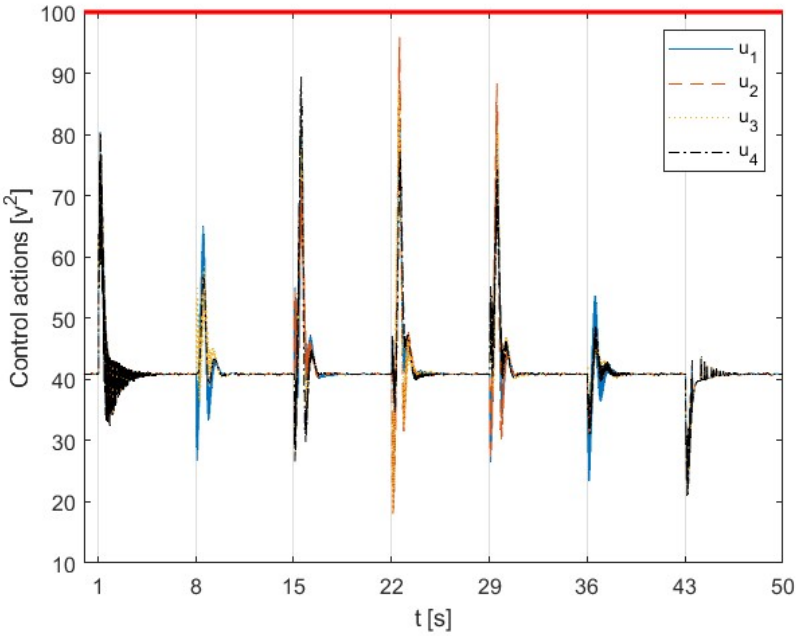



Figure 3.11: Control actions of the integral controller

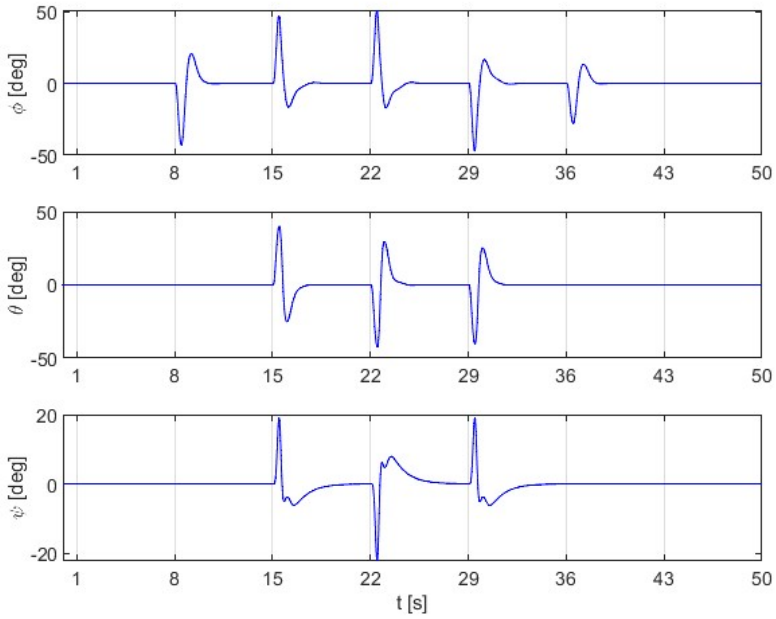



Figure 3.12: $\phi$, $\theta$ and $\psi$ angles of the integral controller

### 3.2.3 Simulation with load

When simulating the system with the integral controller to have a load will the controller be able to erase the steady state error. This is mainly due to the added integrators in the controller. Figure 3.13 shows that the system does reach every checkpoint in the given time. The average time to reach such a checkpoint in this case is $2.02s$. Since the controller was not designed for this load will the control actions also have an influence, namely they will be higher. This is proven in figure 3.14 which shows that the actuators are saturated more often. The only part that doesn't change much during the simulation is the pitch, as seen in figure 3.15.

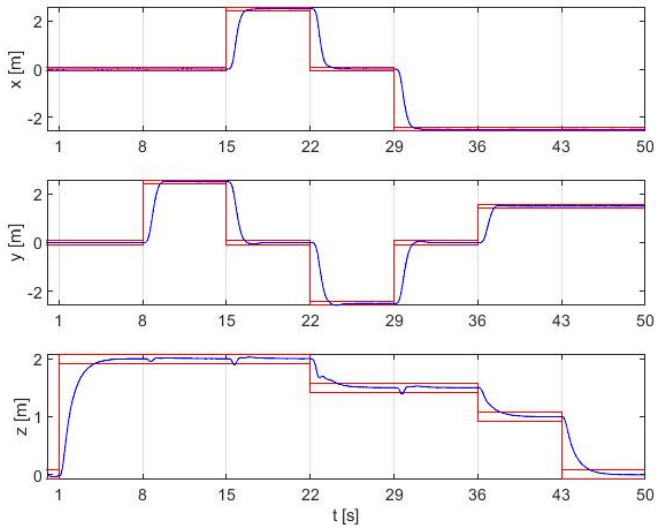



Figure 3.13: $x$, $y$, and $z$ position of the integral controller with load

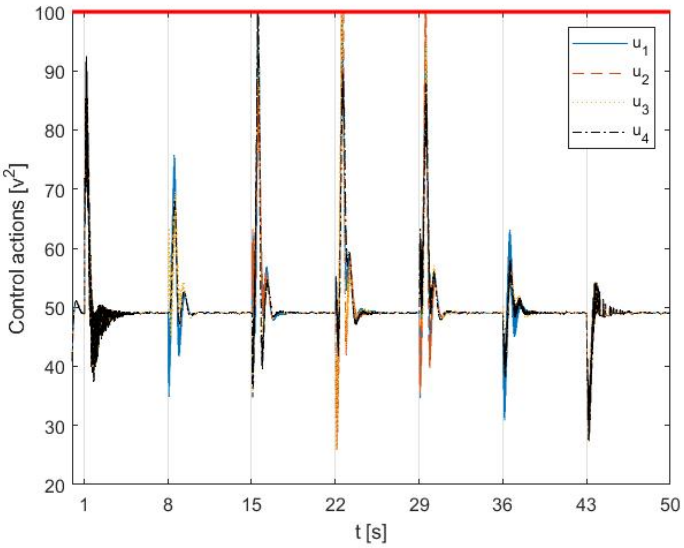



Figure 3.14: Control actions of the integral controller with load

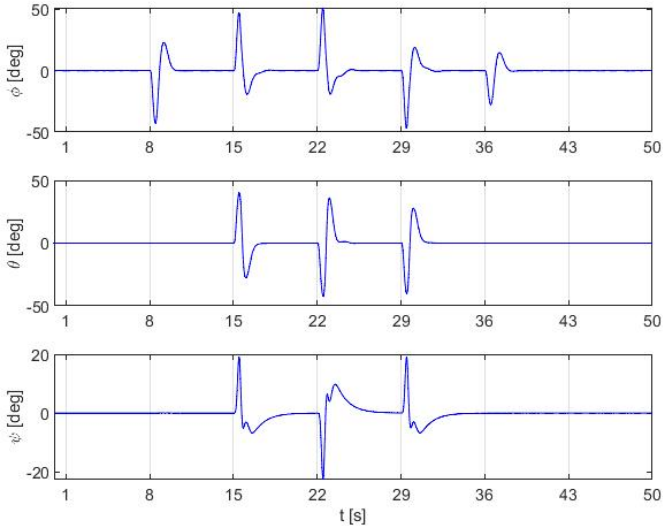



Figure 3.15: $\phi$, $\theta$ and $\psi$ angles of the integral controller with load

## 3.3 Discussion

Both control strategies successfully complete the course under the desired time limit when no load is added. The full state feedback controller takes an average of 1.643 seconds to reach the next point while the integral controller needs an average of 2.007 seconds.

The biggest difference between the two schemes is the control actions that are undertaken. Each time that the reference changes will the control actions of the integral control become very high while those of the full state feedback controller stay small. The only time that the full state feedback control actions become high and saturate the actuators is at the start of the simulation.

Since both controllers complete the course with success is the best controller the one with the lowest average time to reach each checkpoint, which is the full state feedback controller.

When a load is added will both controllers be able to finish the course, nevertheless is the integral controller better than the full state feedback controller for the sole fact that no adjustments had to be made. Since in real life making adjustments to a controller is not so easy.

# Chapter 4

# LQG

In this part will for the first time the states $\boldsymbol{x}$ not be available and thus only the output $\boldsymbol{y}$ and input $\boldsymbol{u}$ can be used to control the system.

## 4.1 Design choices

Since the controller can't use the state-vector anymore will it have to use an estimator to determine the states of the quadcopter by using measurements from the input and output. These sensors add measurement noise $v$ to the system and due to the system having some uncertainties will this add process noise $w$. A stochastic state space description of the system is given in equation 4.1.

$$\begin{aligned} \boldsymbol{x}(k+1) &= A_d\boldsymbol{x}(k) + B_d\boldsymbol{u}(k) + B_1 w(k) \\ \boldsymbol{y}(k) &= C_d\boldsymbol{x}(k) + D_d\boldsymbol{u}(k) + v(k) \end{aligned} \tag{4.1}$$

With $B_1 = \boldsymbol{I}_{12}$.

A Kalman estimator minimizes the error of the estimated state-vector when the variance of the process and measurement noise are known. The measurement noise variance matrix $R$ is known while the process noise matrix $Q$ ( see equation 4.4) is estimated by trail and error. A first estimation for the process variance is by calculating the variance of the difference between the discrete states and the nonlinear states. The values in $Q$ and $R$ also have an impact on the estimator taking either the model or sensors more into account. But setting both matrices to high values will lead to amplifying the noise and thus a trade-off between both is always present.
The Kalman filter gain $L$ is found by solving the ARE in equation 4.3 and plugging it in equation 4.2. The filter gain $L$ is the result of minimizing the variance of the estimation error.

$$L = A_d P C_d^T (R + C_d P C_d^T)^{-1} \tag{4.2}$$

$$P = A_d P_k A_d^T + B_1 Q B_1^T - A_d P C_d^T (R + C_d P C_d^T)^{-1} C_d P A_d^T \tag{4.3}$$

$$\begin{aligned} R &= \begin{bmatrix} 2.5 \times 10^{-5}\boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & 7.57 \times 10^{-5}\boldsymbol{I}_3 \end{bmatrix} \in \mathbb{R}^{6\times 6} \\ Q &= \begin{bmatrix} 1.5 \times 10^{-8}\boldsymbol{I}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & 6.5 \times 10^{-6}\boldsymbol{I}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & 6.5 \times 10^{-9}\boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 & 2 \times 10^{-12}\boldsymbol{I}_3 \end{bmatrix} \in \mathbb{R}^{12\times 12} \end{aligned} \tag{4.4}$$

In practice will this algebraic Riccati equation not be solved, but will a trick be used by finding the minimum of equation 3.2 of the dual system (seen in table 4.1). This would also lead to solving an ARE but from experiments result in a better observer gain $L$.

| LQR | LQG |
|---|---|
| $A_d$ | $A_d^T$ |
| $B_d$ | $C_d^T$ |
| $Q$ | $B_1 Q B_1^T$ |
| $K$ | $L^T$ |

Table 4.1: Dual system between LQG and LQR

The obtained gain $L$ is given in equation 4.5.

$$L = \begin{bmatrix} 0.2048 & 0 & 0 & 0 & 0.0046 & 0 \\ 0 & 0.2048 & 0 & -0.0046 & 0 & 0 \\ 0 & 0 & 0.1933 & 0 & 0 & 0 \\ 0.413 & 0 & 0 & 0 & 0.0204 & 0 \\ 0 & 0.413 & 0 & -0.0204 & 0 & 0 \\ 0 & 0 & 0.3641 & 0 & 0 & 0 \\ 0 & -0.0125 & 0 & 0.0055 & 0 & 0 \\ 0.0125 & 0 & 0 & 0 & 0.0055 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0101 \\ 0 & 0 & 0 & 9.2542 \times 10^{-5} & 0 & 0 \\ 2.0726 \times 10^{-4} & 0 & 0 & 0 & 9.2542 \times 10^{-5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.6172 \times 10^{-4} \end{bmatrix} \tag{4.5}$$

Figure 4.1 shows the diagram of the LQG controller. The diagram is entirely the same as the diagram and parameters of the LQR with integral control (see figure 3.9) except for the estimator which is represented as a state-space block with the following parameters (with $A_d, B_d, C_d, D_d$ the discrete state-space parameters):

$$\begin{cases} A = A_d - L_k C_d \\ B = [B_d - L_k D_d, L_k] \\ C = \boldsymbol{I}_{12\times 12} \\ D = \boldsymbol{0}_{12\times 10} \end{cases}$$

Due to the controller being implemented with integral control shall this lead to steady-state errors being able to go to zero as was the case in LQR with integral control.

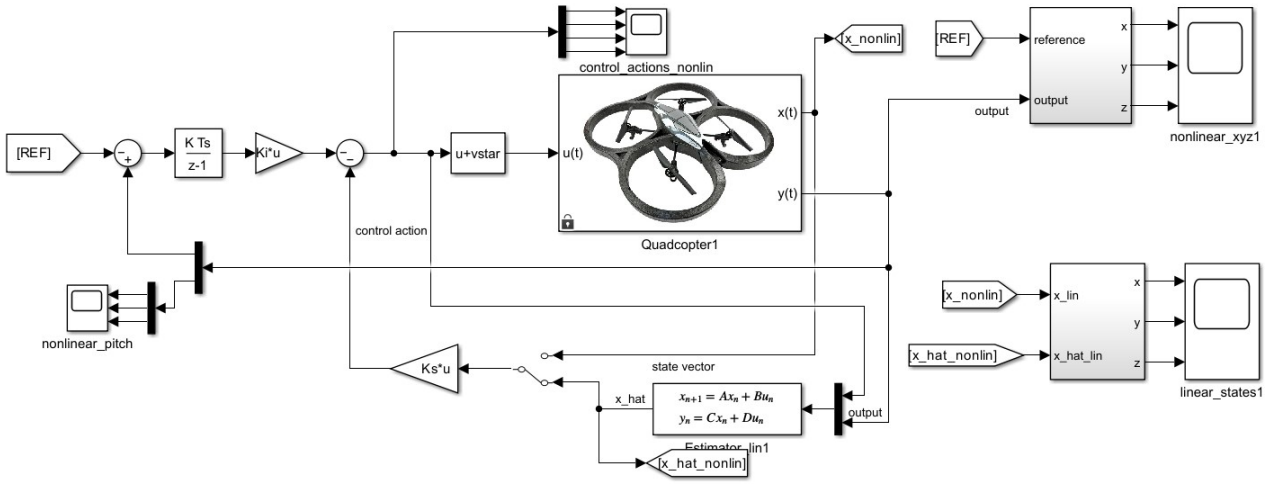
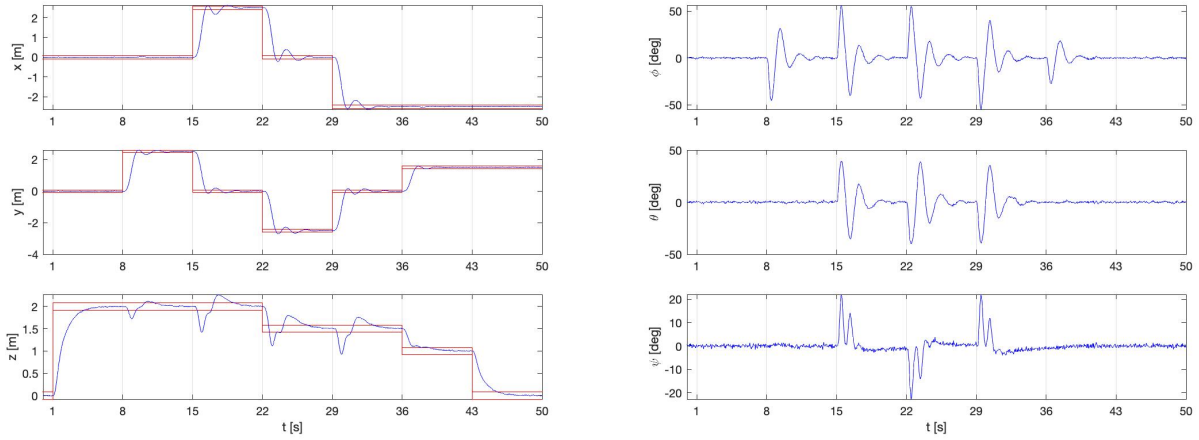



Figure 4.1: Simulink diagram of the LQG controller

## 4.2 Simulation without load

When the state-vector is not available and an estimator is needed, will the reaction time of the quadcopter increase from 2.007s to 3.450s. This is expected because the Kalman filter doesn't want to unnecessarily react on the noise of the system which will result in a slower average time to hit the checkpoint. Even though the system is more robust against noise it is not immune. Figures 4.3 and 4.2b show that there is still some control actions that happen due to noise. This can be seen by the oscillations when all control actions should be stagnant (when staying at the checkpoint). Figure 4.3 also shows that the control actions will never reach saturation, which is a good result. This also explains why the quadcopter takes longer to reach its destination, since the control actions are smaller than in LQR with integral control.




(a) $x$, $y$ and $z$ position

(b) $\phi$, $\theta$ and $\psi$ angles

Figure 4.2: Output values of the controller with LQG

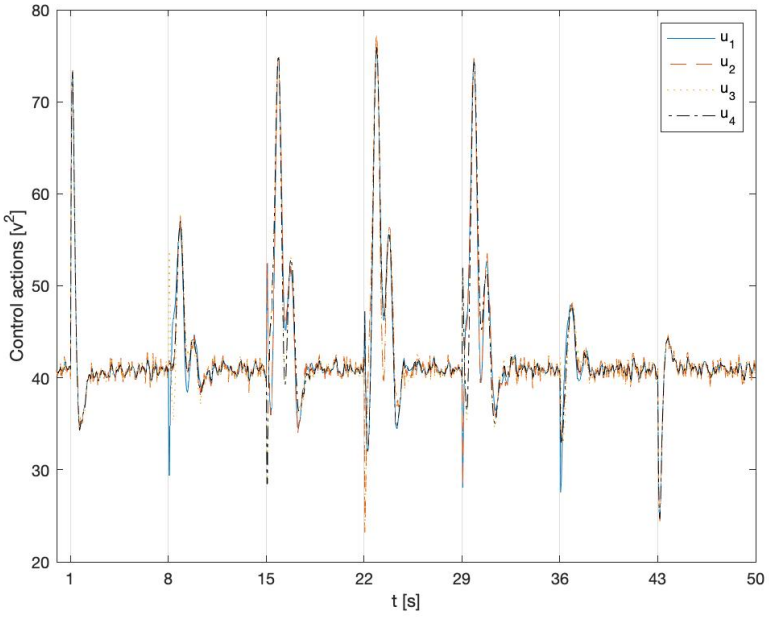
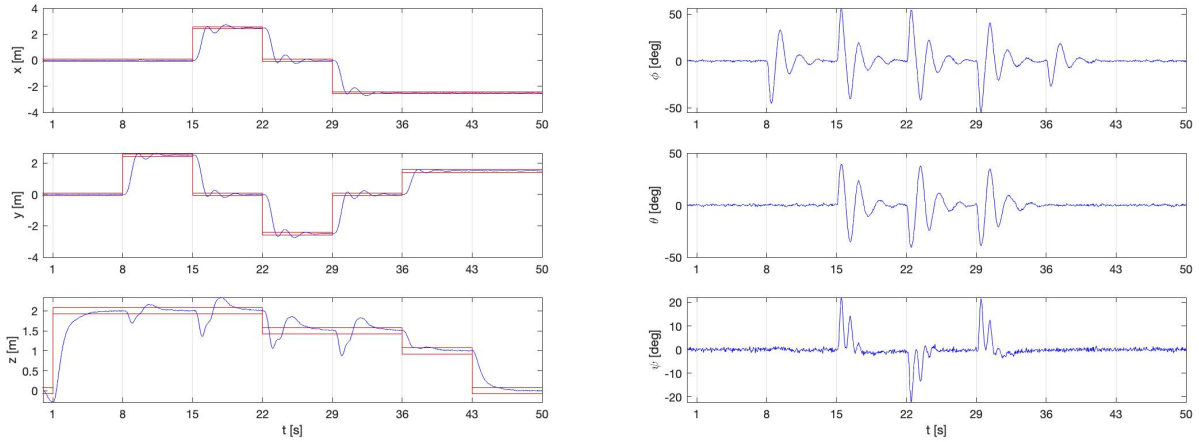



Figure 4.3: Control actions of the controller with LQG

## 4.3 Simulation With load

Just as without load will the average time to reach a checkpoint increase. The time went form 2.020s to 3.750s. Due to the integral component of the controller is the quadcopter able to reach the z-position without fail just like in the LQR with integral control. The integral controller will manage to erase the steady-state error. Figures 4.5 and 4.4b again show that the noise of the system is not completely removed. The controller also manages to not saturate the actuators as seen in figure 4.5 where the control actions are well away from the limits




(a) $x$, $y$ and $z$ position

(b) $\phi$, $\theta$ and $\psi$ angles

Figure 4.4: Output values of the controller with LQG

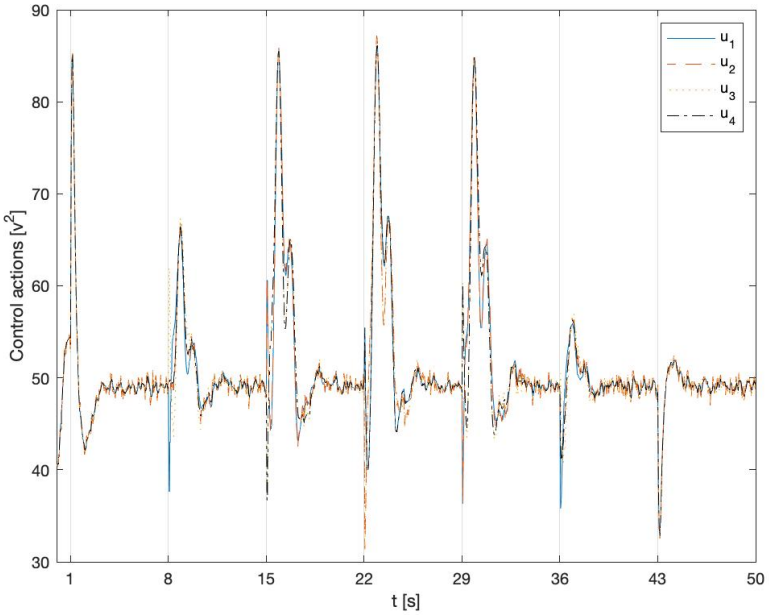



Figure 4.5: Control actions of the controller with LQG

# Chapter 5

# Pole placement

## 5.1 Design choices

Since the controller with the pole placement technique should have full state feedback will this not change much from the one with LQR and full state feedback as seen in figure 3.1. The only addition is the estimator for the states $\hat{x}$ and obviously the change of the controller gain $K$ to the one obtained from using the command *place()*. Figure 5.1 shows the simulink diagram of the controller with pole placement. The main parts of this diagram are the nonlinear system of the quadcopter, the full state feedback controller and the estimator. The nonlinear system is fed with the control actions which come from the linear discrete time controller. The control actions and output are then fed into the estimator, which produces an estimate of the states $\hat{\boldsymbol{x}}$. A close-up of the estimator is shown in figure 5.2. This system is the block representation of the following state space representation

$$\hat{\boldsymbol{x}}(k+1) = A_d\hat{\boldsymbol{x}}(k) + B_d\boldsymbol{u}(k) + L(\boldsymbol{y}(k) - C_d\hat{\boldsymbol{x}}(k) - D_d\boldsymbol{u}(k))$$

The estimator gain $L$ is obtained using the command *place()* on the dual system. The poles are chosen as two times the closed-poles of the system to ensure that the dynamics of the estimator are faster than that of the controller. Afterwards, the controller will then take the estimated states and calculate the new control actions via

$$\boldsymbol{u} = -K(\hat{\boldsymbol{x}}(k) - N_x\boldsymbol{r}(k)) + N_u\boldsymbol{r}(k)$$

Where $N_x$ and $N_u$ are the same matrices used in the LQR with full state feedback scheme. As said before is the control gain $K$ obtained by using the command *place()* in Matlab, which needs the closed-loop poles of the system. In total, 12 poles are needed, from which two are dominant and 10 are non-dominant.

The dominant poles are chosen based on the controller reaction speed. Since the system must reach its destination in seven seconds will the settling time $t_s$ be chosen as $3s$. The damping is chosen as $\zeta = 0.7$. This results in a natural frequency of $\omega_n = 2.1905\frac{rad}{s}$. The dominant poles will be

$$\begin{cases} \lambda_1 = -\zeta\omega_n + j\omega_n\sqrt{1-\zeta^2} = -1.533 + j1.564 \\ \lambda_2 = -\zeta\omega_n - j\omega_n\sqrt{1-\zeta^2} = -1.533 - j1.564 \end{cases}$$

The non-dominant poles can't be too far away from the origin to ensure that the control actions are not too high. But they also can't be too close to ensure that they don't dominate the dynamics of the system. For these reasons are they chosen as the following

$$\begin{cases} \lambda_{3,4,5,6} & = -3.1\zeta\omega_n = -4.753 \\ \lambda_{7,8,9,10} & = -3.2\zeta\omega_n = -4.9067 \\ \lambda_{11,12} & = -3.3\zeta\omega_n = -5.0600 \end{cases}$$

All the poles however are meant for a continuous time system and can thus not be applied to a discrete time system. For this reason must they be converted to discrete time poles with the transformation $z = e^{sT_s}$, where

$T_s = 0.05$ is the sampling time. These poles will result in the controller gain $K$ to be

$$K = \begin{bmatrix} 44.7558 & -1.8583 & 77.6599 & 24.7962 & -3.0513 & 32.1797 & 18.1024 & 53.4253 & 46.0207 & 3.9337 & 3.7766 & 20.304 \\ 6.8206 & 0.5128 & 77.6599 & 5.8006 & 0.2841 & 32.1797 & -0.6121 & 24.026 & -46.0207 & -0.0433 & 4.3524 & -20.304 \\ -44.7558 & 1.8583 & 77.6599 & -24.7962 & 3.0513 & 32.1797 & -18.1024 & -53.4253 & 46.0207 & -3.9337 & -3.7766 & 20.304 \\ -6.8206 & -0.5128 & 77.6599 & -5.8006 & -0.2841 & 32.1797 & 0.6121 & -24.026 & -46.0207 & 0.0433 & -4.3524 & -20.304 \end{bmatrix}$$

The same procedure will be done to obtain the closed-loop poles for the observer gain $L$. The resulting gain is

$$L = \begin{bmatrix} 0.521 & 0.0466 & 0.1025 & -0.0216 & 0.009 & 0 \\ -0.0081 & 0.6612 & -0.0406 & 0.0368 & 4.9154 \times 10^{-5} & 0 \\ -0.0372 & -0.0351 & 0.5817 & 0.1105 & 2.2506 \times 10^{-4} & 0 \\ 1.2344 & 0.4101 & 0.9 & -0.1741 & 0.477 & 0 \\ -0.0809 & 2.4975 & -0.3604 & -0.0664 & 4.8898 \times 10^{-4} & 0 \\ -0.3538 & -0.3113 & 1.7911 & 0.9924 & 0.0021 & 0 \\ 0.1284 & 0.0176 & 0.0556 & 0.59 & -7.7799 \times 10^{-4} & 0 \\ 0 & 0 & 0 & 0 & 0.7079 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6982 \\ 1.2384 & 0.1658 & 0.5271 & 1.9495 & -0.0075 & 0 \\ 0 & 0 & 0 & 0 & 3.0797 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.9341 \end{bmatrix}$$

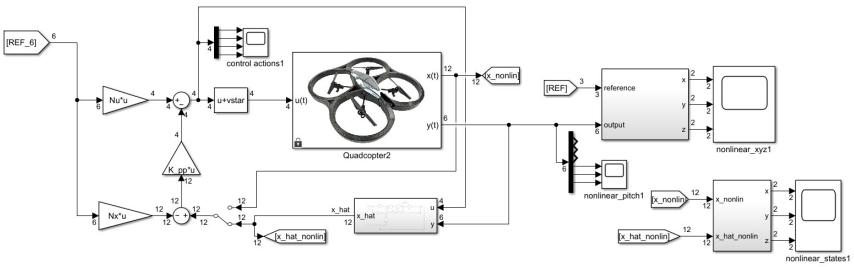



Figure 5.1: Simulink diagram of the controller with Pole placement

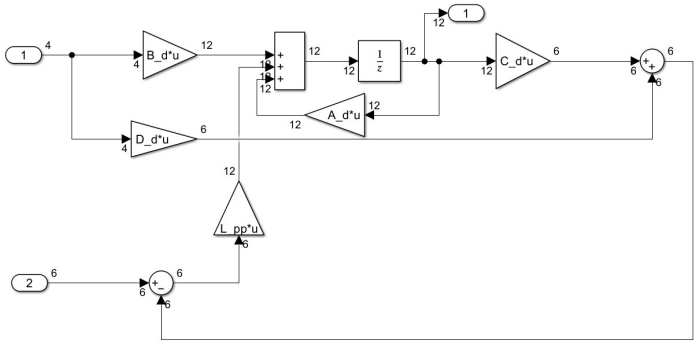
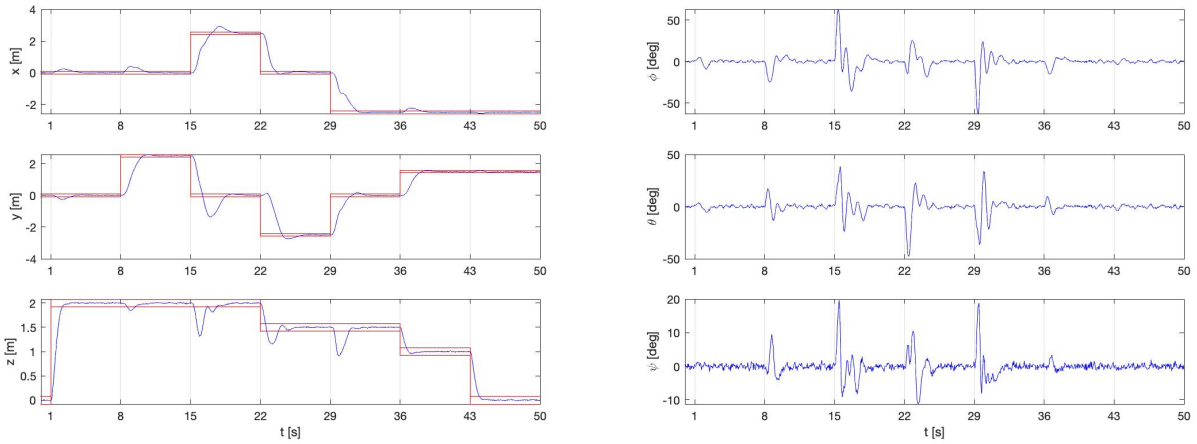



Figure 5.2: Estimator of the system via pole placement

## 5.2 Simulation without load

Figure 5.3a shows the position of the quadcopter when going through the course when no load is added to the system. As expected is the $z$-position the most difficult to achieve due to the force of gravity it has to conquer. The controller also seems to be less quick, since it takes a longer time to reach the set-points. This is also proven in the average time to reach the points, which is 2.721 seconds. Figure 5.3b also shows that the pitch angles are oscillating way more than the case where the states are known. Estimating the states has thus a great effect on the dynamics of the controller. Finally, the control actions must be in the range of operation which is $[0, 100]V$.

To ensure this were the non-dominant poles chosen in such a way that they weren't too far from the origin and thus didn't cause the control gain $K$ to have too high values. The result is pictured in figure 5.4 which shows that the control actions never reach their limits for a too long time.




(a) $x$, $y$ and $z$ position

(b) $\phi$, $\theta$ and $\psi$ angles

Figure 5.3: Output values of the controller with pole placement

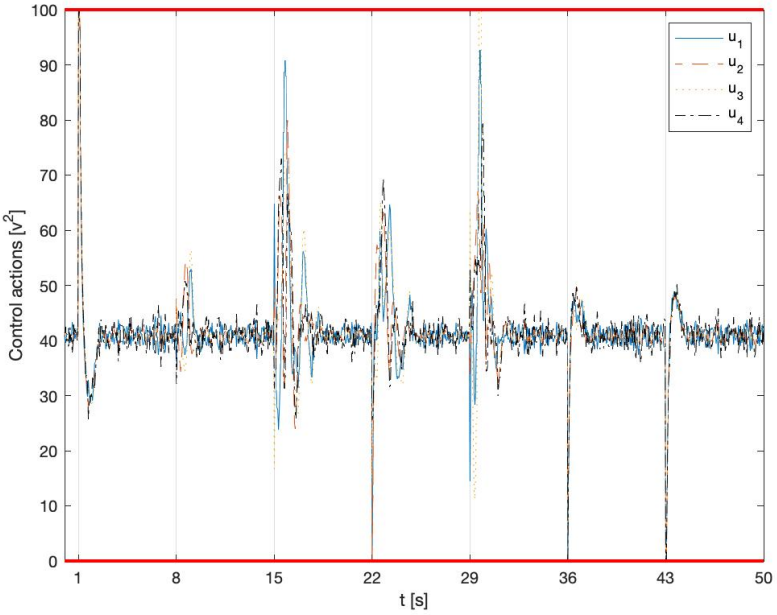
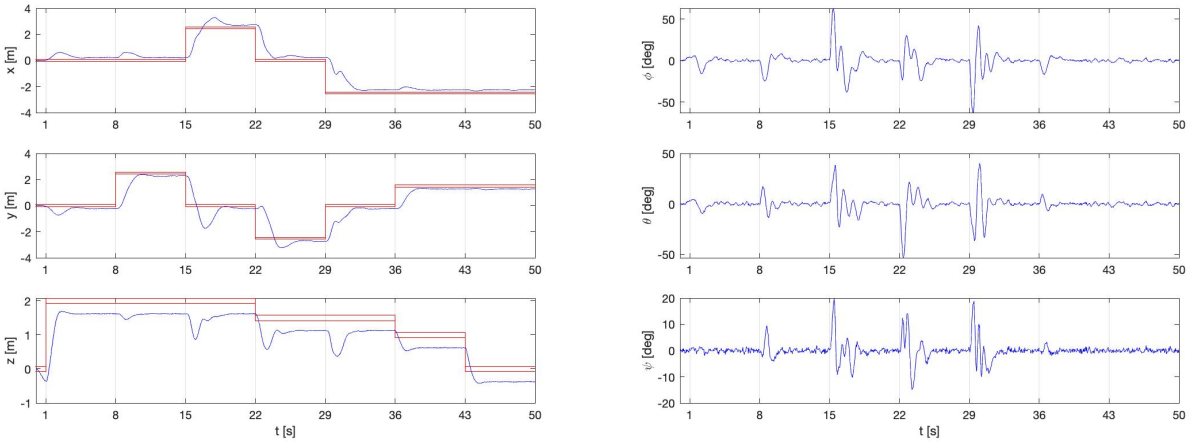



Figure 5.4: Control actions of the controller with pole placement

## 5.3 Simulation with load

Figure 5.5a shows the loaded quadcopter equipped with the pole placement controller. This time the quadcopter has even more trouble to reach the z-position, none of the checkpoints are even reached. This is due to the system having too few inputs for the amount of outputs, and thus are there not enough degrees of freedom. This will lead to the full state feedback controller not being able to erase the steady state error. If the non-dominant

poles are located further form the origin the controller becomes more aggressive and would fix the problem that the quadcopter can't reach the z-position. But then the roll angle will become so large that the quadcopter flips and becomes unstable.




(a) $x$, $y$ and $z$ position

(b) $\phi$, $\theta$ and $\psi$ angles

Figure 5.5: Output values of the controller with pole placement with 0.1 kg load

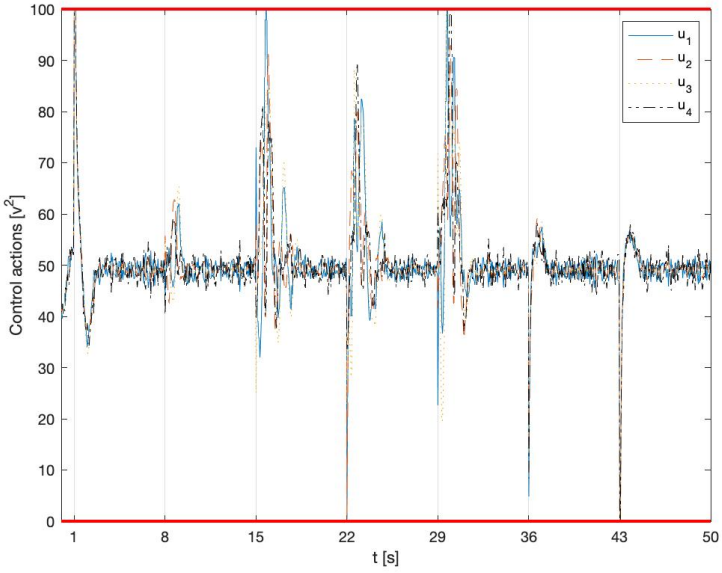



Figure 5.6: Control actions of the controller with pole placement with 0.1 kg load

## 5.4 Pole placement vs LQG

The most notable difference is that the LQG is able to reach all checkpoints while pole placement doesn't when the quadcopter is loaded. This is not surprising considering the LQG estimator has integral control while pole placement doesn't. Both pole placement and LQG are able to reach each checkpoint without load. The noise that the pole placement has is considerable more than with LQG. Which is normal since that LQG knows what the measurement and process noise are from the system and tries to reduce them.

# Conclusion

This report showed different control strategies for controlling a quadcopter with 12 states, six outputs and four inputs. The quadcopter had to reach certain set-points in time with or without a load, and thus reference tracking was introduced in two ways. The first one that was considered was full state feedback and the second one was integral control. The system was first considered to have its full state vector available which lead to the controller to be obtained by the LQR strategy. Afterwards, was the state vector not available anymore and were LQG and pole placement used to make the controller and observer.

The LQR controller was first implemented with full state feedback. This lead to the controller being able to reach the checkpoints in time when no load was available. This was not the case when the system was simulated with a load of $0.1kg$. The controller then failed to erase the steady state error due to the system being overdetermined. Then, a new controller gain $K$ was considered which succeeded in bringing the system to the checkpoints. Afterwards, was integral control added to the controller. This lead to the addition of three integrators which were possible in erasing the steady state error in the initial system and the system with a payload.

When the state vector was not available anymore were two strategies used. The first one was LQG with integral control. This controller was able to reach each checkpoint when the states were estimated. Due to the noise in the nonlinear system was the controller slower and calmer when reacting to the step changes. When adding the load was the controller still able to reach the checkpoints in time. This is due to the controller being implemented with integral control, which is able to eliminate steady state error.

The second observer was implemented with the pole placement technique using full state feedback. The poles were chosen in such a way that the controller was fast enough but won't make the control actions to saturate the actuators. The observer was designed in such a way that the dynamics of the observer were faster than the dynamics of the closed loop system. This lead to the observer being able to follow the changes in the output and input and thus making the estimated states more accurate. The obtained system was able to reach each checkpoint when no load was added. The system was slower than the LQG observer due to the observer and controller not taking the noise of the system into account. When a load was added did the system now fail due to the full state feedback.

# Appendix A

# Matlab code

All the simulations were made in MATLAB and are provided with the report. This appendix explains what the different files are used for:

- *init.m*: the initialization code. This code must be run before using any of the Simulink files due to the code initializing the variables used in the diagrams.
- *references_07.mat*: references used to simulate the course the quadcopter has to go through.
- *linear_model_check.slx*: a Simulink diagram to check the correctness of the linearization.
- *LQR_full_state_feedback.slx*: Simulink diagram that implements the LQR controller with full state feedback. In the diagram there is a choice to use the gain for the simulation without load $K\_fsf$ and with load $K\_fsf\_l$.
- *LQR_integral_control.slx*: Simulink diagram that implements the LQR controller with integral control.
- *LQG.slx*: Simulink diagram that implements the LQG controller with integral control.
- *Pole_placement_estimator.slx*: Simulink diagram that implements the pole placement controller and estimator with full state feedback.