BACHELOR

Measuring turbulence using a quadcopter with LQR control

Ferede, R.

Award date:
2019

Measuring Turbulence using a Quadcopter with LQR Control

Bachelor final project

July 27, 2019

Supervisor
Federico Toschi

Robin Ferede 1004572

Abstract

In this thesis I will derive equations of motion of a quadcopter with external turbulent forces, design an LQR controller for stabilisation, and investigate a methods to measure turbulence based on the state-space trajectories. The LQR controller together with the method of integral feedback is implemented in a simulation using Python. The quadcopter can be observed to be stable even with the introduction of aerodynamic forces. Using a dataset of trajectories from tracer particles in turbulent flows, the response of the controlled quadcopter to turbulence is simulated and analyzed. From these simulated state-space trajectories of the quadcopter it will be shown that the velocity and vorticity of the turbulent flow around the quadcopter can be reconstructed.

Contents

Introduction

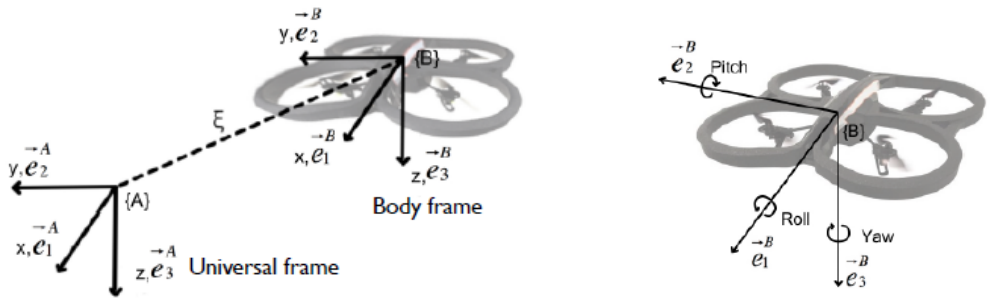
In recent years, there has been an increase in popularity of the use of small unmanned aerial vehicles (UAVs) for a wide variety of applications ranging from surveillance and photography to more advanced military purposes. These UAVs often come in the form of relatively small, lightweight quadcopters. For most applications, these quadcopters are exposed to outside environments with strong and often turbulent winds. For this reason, most quadcopters come equipped with an on-board flight controller that compensate for these winds by changing the thrusts from the propellers using on-board sensor data.

This project investigates the possibility of using a quadcopter as a flexible instrument to measure atmospheric turbulence from inside the flow. Whereas normally measurements of atmospheric turbulence involve either using fixed probes on towers or flying probes mounted on airplanes, this project proposes a more dynamic, smaller scale solution. Especially for measurements of the smaller scale turbulent fluctuations, the response of a small quadcopter could provide useful information. A main question of this project is how to properly design a controller that will make sure that the quadcopter can maintain its position in such a turbulent flow. Another question is whether it is possible to obtain relevant information from a turbulent flow by using the sensor data from the quadcopter's flight controller. These two question will be investigated in the following sections starting with a derivation of a physical model for the quadcopter based on which the controller and the measurement method will be constructed.

1 Quadcopter model

The dynamics of a quadcopter can be modeled using rigid body mechanics. The quadcopter can be described as a rigid body with mass m and moment of inertia I described by a 3 by 3 matrix assumed to be diagonal.

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{1.1}$$



(a) The axis convention used to describe the state of the drone. Note that the z-axis points downwards and the x-axis corresponds to the front of the drone.

(b) The orientation of the drone can be described by the three angles: yaw ψ, pitch θ and roll ϕ

Figure 1.1: Conventions for position and orientation of the body frame

The state of the quadcopter will be fully described by 12 variables. First of all, the position will be described in the universal frame using the axis convention seen in figure 1.1. For the rest of this thesis, this will be denoted by

$$\boldsymbol{\xi} = (x, y, z) \tag{1.2}$$

The orientation of the quadcopter will be described by the euler angles shown in figure 1.1. These angles will be described by

$$\boldsymbol{\lambda} = (\phi, \theta, \psi) \tag{1.3}$$

Each of these angles corresponds to a rotation around one of the axis in the world frame.

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

The orientation of the body frame with respect to the world frame in figure 1.1 can be written in terms of these rotation matrices as follows:

$$R(\boldsymbol{\lambda}) = R(\phi, \theta, \psi) = R_z(\psi) R_y(\theta) R_x(\phi)$$

The velocity of the quadcopter expressed in the world frame will be given by

$$\boldsymbol{v} = (v_x, v_y, v_z) \tag{1.5}$$

Now finally, the angular velocity vector of the quadcopter will be described in the body frame. The direction of this vector is perpendicular to the plane of rotation, according to the right hand rule while the magnitude corresponds to the absolute angular velocity in radians per second. It will be denoted by

$$\boldsymbol{\Omega} = (p, q, r) \tag{1.6}$$

Together, these 12 state variables completely describe the state of the quadcopter. These variables will be described by the 12-dimensional state vector

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\Omega} \end{bmatrix} \tag{1.7}$$

The evolution of this state is now completely determined by the state itself together with the external forces and torques that are provided. For simplicity we will now assume that the only forces acting on the drone are gravity and the thursts generated by the propellers (figure 1.2).

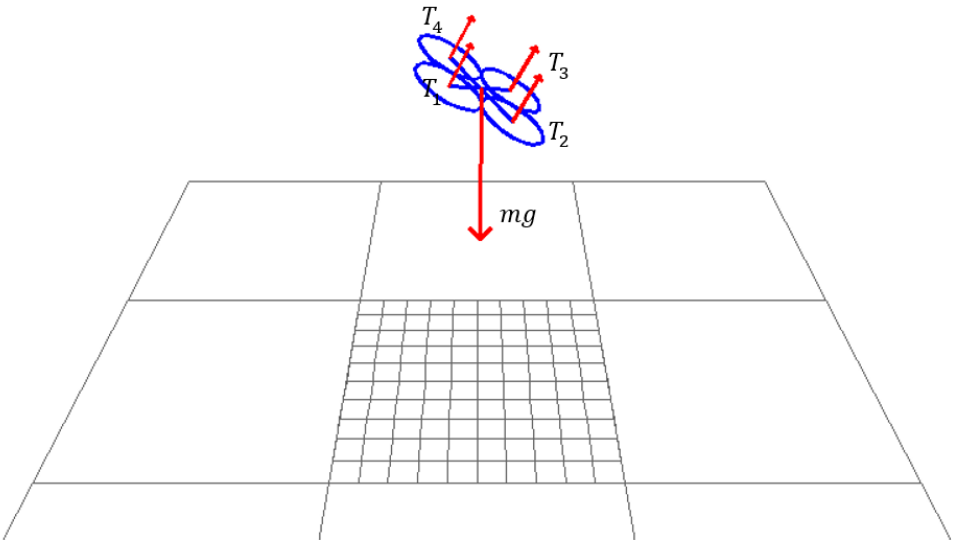


Figure 1.2: The forces acting on the quadcopter where $T_1, \ldots, T_4$ are the thrusts generated by the propellers

Since all propellers point in the same direction, the total thrust generated by the propellers is equal to

$$T = T_1 + T_2 + T_3 + T_4 \tag{1.8}$$

The torques generated by the propellers expressed in the body frame can also be derived using the propeller configuration seen in figure 1.3. In this configuration the propellers on opposite diagonal rotate in the same direction. For this model we will assume each propeller produces a reaction torque proportional to the thurst

$$\tau_i = bT_i \tag{1.9}$$

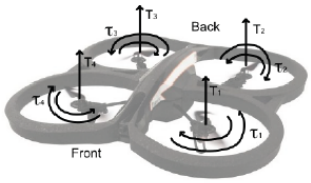


Figure 1.3: Configuration of the propellers. Note that the inside arrow denotes the rotation direction of the propeller, while the outside arrow denotes the reaction torque in the opposite direction.

The torques generated in each direction (expressed in the body frame) can thus be described by

$$\tau_\phi = d(T_1 + T_2) - d(T_3 + T_4) \tag{1.10}$$

$$\tau_\theta = d(T_1 + T_4) - d(T_2 + T_3) \tag{1.11}$$

$$\tau_\psi = b(-T_1 + T_2 - T_3 + T_4) \tag{1.12}$$

Where d is the distance from the center of the propeller to the axis of rotation. So to summarize, the total force and torques generated by $T_1, \ldots, T_4$ can be calculated by

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ d & d & -d & -d \\ d & -d & -d & d \\ -b & b & -b & b \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{1.13}$$

It is important to note that this square matrix is invertible. This namely means that any combination of total thrust T and torques $\tau_\phi, \tau_\theta, \tau_\psi$ can be generated by choosing $T_1, \ldots, T_4$. For this reason, the equations in the next sections will only be stated in terms of T and $\tau_\phi, \tau_\theta, \tau_\psi$. It is however always possible to restate the equations in terms of $T_1, \ldots, T_4$. With this in mind we can now construct the equations of motion that describe how the state (1.7) evolves given some fixed forces and torques.

1.1 Equations of motion

Since the thrust always points in the negative z-direction in the body frame of the drone, the thrust vector in the world frame can be described by

$$-R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) \tag{1.14}$$

Accounting for gravity and using Newton's second law of motion, we get an expression for the total force acting on the drone:

$$m\dot{\boldsymbol{v}} = mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) \tag{1.15}$$

Similarly, since both $\boldsymbol{\Omega}$ and $\boldsymbol{\tau} = (\tau_\phi, \tau_\theta, \tau_\psi)$ are expressed with respect to the body frame, we can describe the evolution of the angular velocity using Euler's equation [1] of rigid body dynamics.

$$I\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} \tag{1.16}$$

All together this results in the following system of differential equations which describe the motion of the quadcopter given a thrust T and a torque $\boldsymbol{\tau}$. The fully expanded system of equation can be found in the appendix (A.1).

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} \end{aligned} \tag{1.17}$$

Here $Q(\boldsymbol{\lambda})$ denotes a transformation between angular velocities and Euler angles given by

$$Q(\boldsymbol{\lambda}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \tag{1.18}$$

A detailed derivation of this matrix can be found in appendix B.

1.2 Model with turbulence

The model with added external forces and torques can be described by

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) + \boldsymbol{F_{ext}} \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} + R(\boldsymbol{\lambda})^T\boldsymbol{\tau_{ext}} \end{aligned} \tag{1.19}$$

Note that since $\boldsymbol{\Omega}$ is expressed in the body frame, $\boldsymbol{\tau_{ext}}$ needs to be transformed by $R(\boldsymbol{\lambda})^T$. These external forces and torques will be derived using a simple drag model. Let $\boldsymbol{v}_{wind}$ and $\nabla \times \boldsymbol{v}_{wind}$ describe the velocity and vorticity of a turbulent fluid at the position of the quadcopter. We will calculate the external forces and torques on the quadcopter as follows:

$$\boldsymbol{F_{ext}} = \alpha(\boldsymbol{v}_{wind} - \boldsymbol{v}) \tag{1.20}$$

$$\boldsymbol{\tau_{ext}} = \beta(\nabla \times \boldsymbol{v}_{wind} - R(\boldsymbol{\lambda})\boldsymbol{\Omega}) \tag{1.21}$$

Here α and β correspond to coefficients of friction that are related to the shape of the quadcopter. Substitution into (1.19) results in the following model

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) + \alpha(\boldsymbol{v}_{wind} - \boldsymbol{v}) \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} + \beta(R(\boldsymbol{\lambda})^T(\nabla \times \boldsymbol{v}_{wind}) - \boldsymbol{\Omega}) \end{aligned} \tag{1.22}$$

This will be the model used in the simulation and statistical analysis in the later chapters.

2 Control

Consider the quadcopter model without external forces (1.17). In this model the only forces are gravity and the thrusts of the propellers. These forces are assumed to be fixed, since no form of feedback has yet been discussed. From equation (1.17) it can quickly be observed that the system only has stationary points for $T = mg$ and $\boldsymbol{\tau} = 0$. The stationary point will then be of the form

$$\boldsymbol{y}_0 = \begin{bmatrix} \boldsymbol{\xi}_0 & = (x_0, y_0, z_0) \\ \boldsymbol{v}_0 & = (0,0,0) \\ \boldsymbol{\lambda}_0 & = (0,0,0) \\ \boldsymbol{\Omega}_0 & = (0,0,0) \end{bmatrix} \tag{2.1}$$

We will refer to this state, together with the control inputs $T = mg$ and $\boldsymbol{\tau} = 0$ as the state of hover. This stationary state namely corresponds to the drone hovering on a fixed position (x_0, y_0, z_0). Substituting these fixed control inputs $T, \boldsymbol{\tau}$ into equation (1.17) results in

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(mg\boldsymbol{e_3}) \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} \end{aligned} \tag{2.2}$$

Substituting (2.1) indeed results in $\dot{\boldsymbol{\xi}} = \dot{\boldsymbol{v}} = \dot{\boldsymbol{\lambda}} = \dot{\boldsymbol{\Omega}} = 0$. However, this state of hover is not stable. Small deviations in the state, will cause the state to deviate even further. For example, by slightly changing the orientation of the drone $\boldsymbol{\lambda}$, the acceleration $\dot{\boldsymbol{v}}$ will be non zero causing the drone to drift in one direction. For this reason a controller is needed. The forces and torques $T, \boldsymbol{\tau}$ provided by the propellers need to be adjusted based on the state in such a way that the system will stabilize around this state of hover. The goal for the next subsections will be to design a feedback function $F : \mathbb{R}^{12} \to \mathbb{R}^4$ that maps the state vector to the control inputs

$$F : (\boldsymbol{\xi}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\Omega}) \mapsto (T, \tau_x, , \tau_y, \tau_z) \tag{2.3}$$

such that the system remains stable around the hover state.

2.1 Linear state feedback

As the name suggest, the method of linear state feedback uses a linear function for equation (2.3) to determine the control inputs. More specifically, for our case we will need some 4 by 12 matrix K that maps the 12 dimensional state to the 4 dimensional control input. To keep the notation simple, we will describe the state and control input by the letters $\boldsymbol{y}$ and $\boldsymbol{u}$ respectively

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\Omega} \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \tag{2.4}$$

Equation (1.17) will be written as

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) \tag{2.5}$$

A stationary point of this system will be described by $(\boldsymbol{y}_0, \boldsymbol{u}_0)$ where $\boldsymbol{y}_0$ is a state of hover given by (2.1) and where $\boldsymbol{u}_0$ is the corresponding control input

$$\boldsymbol{u}_0 = \begin{bmatrix} T = mg \\ \tau_x = 0 \\ \tau_y = 0 \\ \tau_z = 0 \end{bmatrix} \tag{2.6}$$

$T = mg, \boldsymbol{\tau} = 0$. We will now construct a linear feedback function that will stabilize the system (2.5) around $(\boldsymbol{y}_0, \boldsymbol{u}_0)$ by using the feedback function

$$(\boldsymbol{u} - \boldsymbol{u}_0) = -K(\boldsymbol{y} - \boldsymbol{y}_0) \tag{2.7}$$

Where K is a 4 by 12 matrix. Or in other words

$$\begin{bmatrix} T - mg \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = - \begin{bmatrix} k_{1,1} & \cdots & k_{1,12} \\ \vdots & \ddots & \vdots \\ k_{4,1} & \cdots & k_{4,12} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi} - \boldsymbol{\xi}_0 \\ \boldsymbol{v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\Omega} \end{bmatrix}$$

Non linear	**Linearized around hover** $(T = mg + \delta T)$
$\dot{x} = v_x$	$\dot{x} = v_x$
$\dot{y} = v_y$	$\dot{y} = v_y$
$\dot{z} = v_z$	$\dot{z} = v_z$
$\dot{v_x} = -(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)\frac{T}{m}$	$\dot{v_x} = -g\theta$
$\dot{v_y} = -(\cos\phi\sin\psi\sin\theta + \cos\psi\sin\phi)\frac{T}{m}$	$\dot{v_y} = g\phi$
$\dot{v_z} = -(\cos\phi\cos\theta)\frac{T}{m} + g$	$\dot{v_z} = -\delta T/m$
$\dot{\phi} = p + \sin\phi\tan\theta q + \cos\phi\tan\theta r$	$\dot{\phi} = p$
$\dot{\theta} = \cos\phi q - \sin\phi r$	$\dot{\theta} = q$
$\dot{\psi} = (\sin\phi/\cos\theta)q + (\cos\phi/\cos\theta)r$	$\dot{\psi} = r$
$\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{\tau_x}{I_x}$	$\dot{p} = \tau_x$
$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{\tau_y}{I_y}$	$\dot{q} = \tau_y$
$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{\tau_z}{I_z}$	$\dot{r} = \tau_z$

Table 2.1: Non linear and linearized equations of motion

The feedback matrix will be determined by analysing the stability of equation (2.5) around $\boldsymbol{y}_0$. The stability can be determined using the linearization of this equation around $\boldsymbol{y}_0, \boldsymbol{u}_0$.

$$\dot{\boldsymbol{y}} = A(\boldsymbol{y} - \boldsymbol{y}_0) + B(\boldsymbol{u} - \boldsymbol{u}_0) \tag{2.8}$$

Where A, B are real constant matrices with dimensions 12×12 and 4×12 respectively. A fully expanded form of these equations can be found in Table 2.1. The matrices A and B can be found by calculating the following partial derivatives of $\boldsymbol{f}$ from equation (2.5):

$$A_{ij} = \frac{\partial f_i}{\partial y_j} \quad B_{ij} = \frac{\partial f_i}{\partial u_j} \tag{2.9}$$

This gives the following matrices

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1/m & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1/I_x & 0 & 0 \\ 0 & 0 & 1/I_y & 0 \\ 0 & 0 & 0 & 1/I_z \end{bmatrix} \tag{2.10}$$

The goal is now to find a matrix K such that

$$\dot{\boldsymbol{y}} = A(\boldsymbol{y} - \boldsymbol{y}_0) + B(\boldsymbol{u} - \boldsymbol{u}_0) \tag{2.11}$$

$$= A(\boldsymbol{y} - \boldsymbol{y}_0) - BK(\boldsymbol{y} - \boldsymbol{y}_0) \tag{2.12}$$

$$= [A - BK](\boldsymbol{y} - \boldsymbol{y}_0) \tag{2.13}$$

is stable. From ODE theory, it follows that this is only the case when all the eigenvalues of $[A - BK]$ have a negative real part. Under some weak condition on the matrices A, B it has been proven [2] that there exists a matrix K that achieves this[1]. There are even many different choices of K that all stabilize the system. In the next section we will investigate one method to choose such a stabilizing matrix in a certain optimal way.

2.2 Linear–quadratic regulator (LQR)

In the theory of optimal control, the objective is to design a feedback that minimizes a certain cost function. The linear quadratic regulator provides an optimal linear feedback for some quadratic cost function. The problem can be stated as follows. Consider a general linear system given by

$$\dot{\boldsymbol{y}} = A\boldsymbol{y} + B\boldsymbol{u} \tag{2.14}$$

Where A has dimensions $n \times n$ and B has dimensions $n \times m$. The linear quadratic regulator then provides a unique stabilizing feedback control law [3].

$$\boldsymbol{y} = -K\boldsymbol{y} \tag{2.15}$$

That minimizes the cost function

$$J = \int_0^\infty \boldsymbol{y}Q\boldsymbol{y}^T + \boldsymbol{u}R\boldsymbol{u}^T dt \tag{2.16}$$

Where Q and R are real symmetric positive definite matrices that can be chosen beforehand. Note that since the integrand is always positive, the cost function is a measure of the accu-

[1]The condition on the matrices A and B is that the matrix $[B, AB, A^2B, \ldots, A^{11}B] \in \mathbb{R}^{12\times 48}$ has full rank. In this case the system is called 'controllable'. Using Mathematica it could be confirmed that our system (A, B) is indeed controllable.

mulated error of both the state and of the control input.

Using methods from calculus of variations it can be shown that the unique solution to this problem is given by

$$K = R^{-1}B^T P \tag{2.17}$$

where P is the unique positive definite solution to the algebraic Riccati equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{2.18}$$

The derivation of these equations is outside the scope of this thesis but can be found in the lecture notes [3] referenced in the bibliography. Now given some matrices Q and R we can determine the optimal feedback K for any controllable linear system (A, B) by solving equation (2.18).

2.3 Integral Control

Up until this point we have only considered designing a controller for the equations of motion without external forces. This will actually be a problem since the introduction of external forces can cause the system controlled by the LQR feedback to no longer stabilize around the desired hover state. In this section we will discuss the method of integral control to overcome this problem

Consider the model with external forces (1.19). For this section we will assume that these external forces do not depend on time. Suppose T and $\boldsymbol{\tau}$ were computed from some linear feedback derived from the LQR algorithm.

$$(\boldsymbol{u} - \boldsymbol{u}_0) = -K(\boldsymbol{y} - \boldsymbol{y}_0) \tag{2.19}$$

Where $\boldsymbol{u}_0$ and $\boldsymbol{y}_0$ are again given by the hover state (equation (2.1) and (2.6)). Then for $\boldsymbol{y} = \boldsymbol{y}_0$ the control input would be $\boldsymbol{u} = \boldsymbol{u}_0$. Substitution into (1.19) now gives

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= 0 \\ m\dot{\boldsymbol{v}} &= \boldsymbol{F_{ext}} \\ \dot{\boldsymbol{\lambda}} &= 0 \\ I\dot{\boldsymbol{\Omega}} &= \boldsymbol{\tau_{ext}} \end{aligned} \tag{2.20}$$

Which is now no longer stationary. This means that the quadcopter will no longer stabilize around this hover state. The result of this will be that the drone would drift away and stabilize with an offset from the desired position $\boldsymbol{\xi}_0$. This drift can be eliminated by using integral control. Instead of calculating the feedback based on the state variables only, in integral control extra information is used. Namely the state vector will be augmented with extra variables corresponding to some integrated error. This extra variable will ensure that the only stationary state will be the state of zero 'error'. In the case of our quadcopter model this could mean the following. Suppose we want to ensure that there is no drift in position

anymore. Then we can augment our state as follows

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) + \boldsymbol{F_{ext}} \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} + R(\boldsymbol{\lambda})^T\boldsymbol{\tau_{ext}} \\ \dot{\boldsymbol{\sigma}} &= \boldsymbol{\xi} - \boldsymbol{\xi}_0 \end{aligned} \tag{2.21}$$

Similar to the previous section, we can now derive a new LQR feedback that takes into account this extra state variable based on the linearization of the new equations[2]. Since the new variable corresponds to the integrated error

$$\boldsymbol{\sigma}(t) = \int_0^t \boldsymbol{\xi}(t') - \boldsymbol{\xi}_0 dt' \tag{2.22}$$

we can now expect that the drone will react with stronger forces and torques the longer it remains offset from the target $\boldsymbol{\xi}$. Also the new equation in (2.21) ensures that the state can only be stationary if $\dot{\boldsymbol{\sigma}} = 0$ and thus $\boldsymbol{\xi} = \boldsymbol{\xi}_0$ which is exactly what is desired.

[2]Note that in the linearization used for the LQR feedback, the external forces can not be incorporated since they are treated as unknowns.

3 Measuring turbulence

Given that there is some stabilizing LQR control with integral feedback implemented on the quadcopter, the next step in this project is to determine the turbulence based on the evolution of the quadcopter's state.

Since integral feedback is used the system with external forces will be given by

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) + \boldsymbol{F}_{ext} \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} + R(\boldsymbol{\lambda})^T\boldsymbol{\tau}_{ext} \\ \dot{\boldsymbol{\sigma}} &= \boldsymbol{\xi} - \boldsymbol{\xi}_0 \end{aligned} \tag{3.1}$$

Note that here $\boldsymbol{u} = (T, \tau_x, \tau_y, \tau_z)$ is implicitly given by

$$(\boldsymbol{u} - \boldsymbol{u}_0) = -K(\boldsymbol{y} - \boldsymbol{y}_0) \tag{3.2}$$

The equations (3.1) can be rearranged to obtain $\boldsymbol{F}_{ext}$ and $\boldsymbol{\tau}_{ext}$

$$\begin{aligned} \boldsymbol{F}_{ext} &= m\dot{\boldsymbol{v}} - mg\boldsymbol{e_3} + R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) \\ \boldsymbol{\tau}_{ext} &= R(\boldsymbol{\lambda})(I\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega} - \boldsymbol{\tau}) \end{aligned}$$

Using the simple drag model from (1.2) we get the following expressions for the external force and torque:

$$\begin{aligned} \boldsymbol{F_{ext}} &= \alpha(\boldsymbol{v}_{wind} - \boldsymbol{v}) \\ \boldsymbol{\tau_{ext}} &= \beta(\nabla \times \boldsymbol{v}_{wind} - R(\boldsymbol{\lambda})\boldsymbol{\Omega}) \end{aligned}$$

In these equations $\boldsymbol{v}_{wind}$ and $\nabla \times \boldsymbol{v}$ are the wind velocity and vorticity at the current position of the quadcopter. Now assuming the values α, β and m, I, g are known we can actually reconstruct the wind velocity and vorticity at each point along the quadcopter's trajectory.

Let $\boldsymbol{y}(t)$ be a state-space trajectory of the quadcopter (i.e a solution to the equations (3.1)). The control inputs $\boldsymbol{u}(t)$ at each point in time are given by the LQR feedback (3.2) which depends only on the current state $\boldsymbol{y}(t)$. This means that given the state at time t

$$\boldsymbol{y}(t) = (\boldsymbol{\xi}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\Omega}, \boldsymbol{\sigma}) \tag{3.3}$$

$$\boldsymbol{u}(t) = (T, \boldsymbol{\tau}) \tag{3.4}$$

$\boldsymbol{y}(t) = (\boldsymbol{\xi}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\Omega}, \boldsymbol{\sigma})$ all the control inputs $\boldsymbol{u}(t) = (T, \boldsymbol{\tau})$ are also known. This makes it possible to solve for $\boldsymbol{v}_{wind}, \nabla \times \boldsymbol{v}_{wind}$ in the following equations

$$\begin{aligned} \alpha(\boldsymbol{v}_{wind} - \boldsymbol{v}) &= m\dot{\boldsymbol{v}} - mg\boldsymbol{e_3} + R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) \\ \beta(\nabla \times \boldsymbol{v}_{wind} - R(\boldsymbol{\lambda})\boldsymbol{\Omega}) &= R(\boldsymbol{\lambda})(I\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega} - \boldsymbol{\tau}) \end{aligned}$$

resulting in

$$\boldsymbol{v}_{wind} = \boldsymbol{v} + \frac{1}{\alpha}(m\dot{\boldsymbol{v}} - mg\boldsymbol{e_3} + R(\boldsymbol{\lambda})(T\boldsymbol{e_3})) \tag{3.5}$$

$$\nabla \times \boldsymbol{v}_{wind} = R(\boldsymbol{\lambda})\boldsymbol{\Omega} + \frac{1}{\beta}(R(\boldsymbol{\lambda})(I\dot{\boldsymbol{\Omega}} + \boldsymbol{\Omega} \times I\boldsymbol{\Omega} - \boldsymbol{\tau}))$$

Performing these calculation using the sensor data of the quadcopter makes it possible to do a time series measurement of the wind velocity and vorticity around the quadcopter!

4 Simulation and statistical analysis

4.1 Simulation

In this section we will simulate the dynamics of a quadcopter with LQR control subject to external forces using the programming language Python together with the libraries NumPy and SciPy. The control feedback discussed in section 2 will be numerically calculated for a quadcopter with given intrinsic parameters. The response of this closed-loop system will then be investigated both with and without external forces.

For the simulation the following parameters were chosen:

Parameter	Value	Unit
m	1	kg
I_x	0.1	kg m^2
I_y	0.1	kg m^2
I_z	0.05	kg m^2
g	10.0	m s^{-2}
α	0.1	t^{-1}
β	0.1	t^{-1}

Table 4.1: Parameters used for the simulation. Where m is the mass of the quadcopter, I_x, I_y, I_z are the moments of inertia, g the gravitational constant and α and β the drag coefficients from equations (1.20) and (1.21).

4.1.1 Model without external forces

For simplicity, we will start with an analysis of an LQR controller in the model without external forces (1.17). The objective will be to stabilize the quadcopter around the state of hover described in equation (2.1). This state will be chosen to be at one meter above the ground. In other words

$$\boldsymbol{y}_0 = \begin{bmatrix} \boldsymbol{\xi}_0 & = (0,0,-1) \\ \boldsymbol{v}_0 & = (0,0,0) \\ \boldsymbol{\lambda}_0 & = (0,0,0) \\ \boldsymbol{\Omega}_0 & = (0,0,0) \end{bmatrix} \quad \boldsymbol{u}_0 = \begin{bmatrix} T = 10.0 \\ \tau_x = 0 \\ \tau_y = 0 \\ \tau_z = 0 \end{bmatrix} \tag{4.1}$$

Note that $T = 10.0$ is found by substituting m and g in equation (2.6). Also note that the z-component of $\boldsymbol{\xi}_0$ is negative due to the axis convention chosen in section 1. To derive now a stable linear feedback will use the method of LQR control to calculate matrix K corresponding to the feedback

$$(\boldsymbol{u} - \boldsymbol{u}_0) = -K(\boldsymbol{y} - \boldsymbol{y}_0) \tag{4.2}$$

This matrix can be found by numerically solving the Riccati equation (2.18) for P given the matrices A, B, Q and R. Here the matrices Q and R determine the cost function (2.16) and

can be chosen[3]. The matrices A and B can be found by substituting the parameters from Table 4.1 into (2.10) resulting in

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 10.0 & 0 & 0 \\ 0 & 0 & 10.0 & 0 \\ 0 & 0 & 0 & 20.0 \end{bmatrix} \tag{4.3}$$

For the simulation we will choose the following diagonal matrices for Q and R.

$$Q = \mathbf{diag}(\begin{bmatrix} 100 & 100 & 100 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}) \quad R = \mathbf{diag}(\begin{bmatrix} 10 & 10 & 10 & 10 \end{bmatrix}) \tag{4.4}$$

These diagonal elements determine the weight of each of the state variables in the cost function (2.16). Since this cost function will be minimized by the LQR feedback, we can expect that the position will be controlled 'best' and thus remain closest to $\boldsymbol{\xi}_0$. The diagonal elements of R make sure that the control outputs given by the feedback do not grow to big. This is especially important considering that in a real quadcopter all the thrusts are limited to a certain range. Since the linear system is given by (2.8) the cost function can be written as follows

$$\begin{aligned} J = \int_0^\infty & 100(x+1)^2 + 100y^2 + 100z^2 + v_x^2 + v_y^2 + v_z^2 + \phi^2 + \theta^2 + \psi^2 + p^2 + q^2 + r^2 \\ & + 10(T - mg)^2 + 10\tau_x + 10\tau_y + 10\tau_z dt \end{aligned}$$

Note that for this function to be minimized, the state would have to converge towards $x = -1, T = mg$ and $y, z, v_x, v_y, v_z, \phi, \theta, \psi, p, q, r = 0$ i.e $\boldsymbol{y} = \boldsymbol{y}_0, \boldsymbol{u} = \boldsymbol{u}_0$.

Now that all the matrices A, B, Q and R are numerically defined, the Riccati equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

can numerically be solved by using the SciPy function *solve_ continuous_ are*. The equation is solved directly using a Schur decomposition method described in [4]. The SciPy function uses the matrices A, B, Q, R as inputs and returns the matrix P as an output. The feedback matrix K can then be computed using (equation (2.17)).

$$K = R^{-1}B^T P$$

Performing these computations in Python and rounding to 3 decimal places gives our final

[3]Under the condition that both the matrices are real symmetric positive definite.

solution to the control problem.

$$K = \begin{bmatrix} 0 & 0 & -3.162 & 0 & 0 & -2.535 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.162 & 0 & 0 & 2.054 & 0 & 6.515 & 0 & 0 & 1.184 & 0 & 0 \\ -3.162 & 0 & 0 & -2.054 & 0 & 0 & 0 & 6.515 & 0 & 0 & 1.184 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.316 & 0 & 0 & 0.363 \end{bmatrix} \tag{4.5}$$

Substituting this matrix in equation (4.2) results in an explicit form for the control input.

$$\begin{aligned} T &= 10.0 + 3.16z + 2.535v_z \\ \tau_x &= -3.162y - 2.054v_y - 6.515\phi - 1.184p \\ \tau_y &= 3.162(x+1) + 2.054v_x - 6.515\theta - 1.184q \\ \tau_z &= -0.316\psi - 0.363r \end{aligned} \tag{4.6}$$

With this feedback law substituted in the equations of motion, we now have a closed the system

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} \end{aligned} \tag{4.7}$$

By calculating the eigenvalues of the linearization matrix in $\dot{\boldsymbol{y}} = [A - BK]\boldsymbol{y}$, we can check the stability of the system around the state of hover $\boldsymbol{y}_0$. Using the SciPy function *eig* we calculate the following eigenvalues

$$\begin{matrix} -1.781 + 3.667i & -1.781 - 3.667i & -1.781 + 3.667i \\ -1.781 - 3.667i & -6.243 & -4.142 + 1.369i \\ -4.142 - 1.369i & -4.142 + 1.369i & -4.142 - 1.369i \\ -1.013 & -1.267 + 1.247i & -1.267 - 1.247i \end{matrix}$$

Note that the real part of all these eigenvalues are in fact all negative. This means that the system is indeed stable. Using the SciPy function *odeint*, it is now possible to numerically integrate the differential equation (4.7). In figure 4.1 and 4.2 the solution to this closed loop system is plotted for different initial conditions. In both cases the state converges to the state of hover $\boldsymbol{y}_0$.

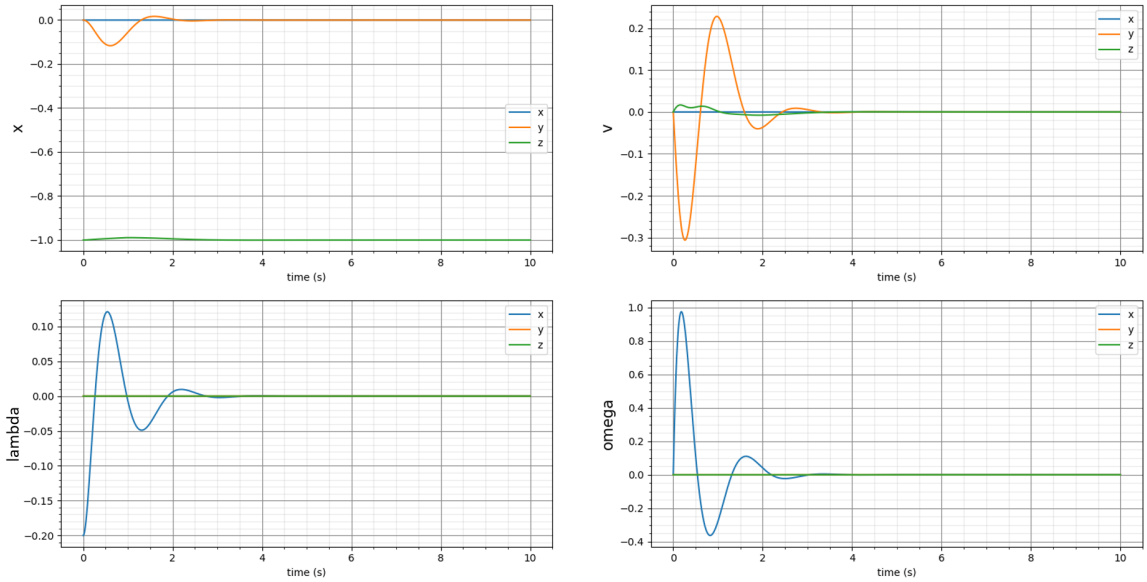


Figure 4.1: The state space trajectory of the system (4.7) with initial conditions $\boldsymbol{\xi}(0) = (0, 0, -1) \quad \boldsymbol{v}(0) = 0 \quad \boldsymbol{\lambda}(0) = (-0.2, 0, 0) \quad \boldsymbol{\Omega}(0) = 0$

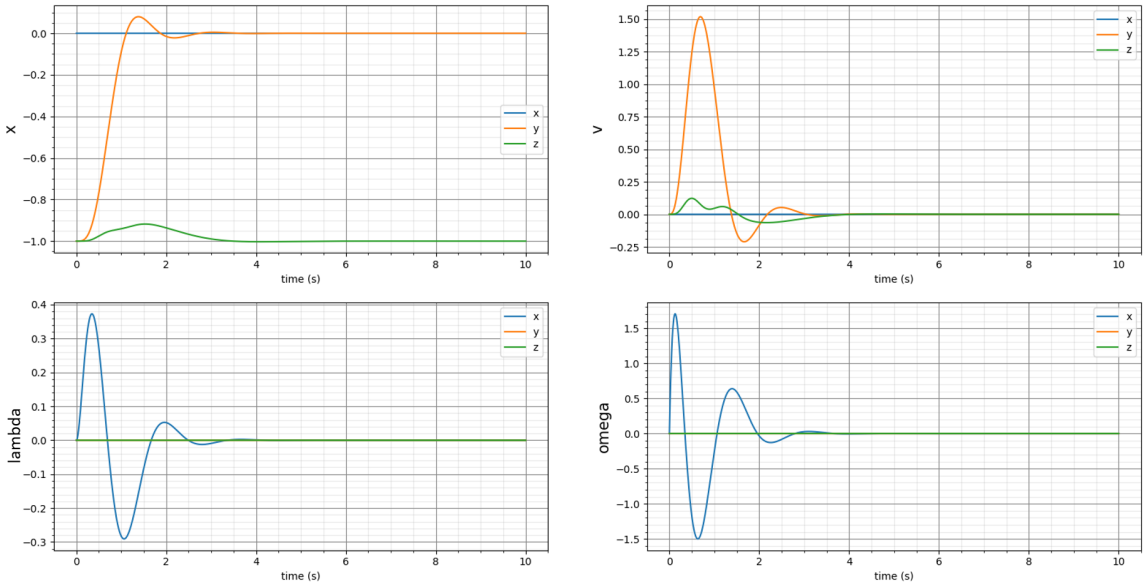


Figure 4.2: The state space trajectory of the system (4.7) with initial conditions $\boldsymbol{\xi}(0) = (0, -1, -1) \quad \boldsymbol{v}(0) = 0 \quad \boldsymbol{\lambda}(0) = (0, 0, 0) \quad \boldsymbol{\Omega}(0) = 0$

4.1.2 Model with external forces

Suppose that we now use the feedback derived in the previous section (4.6) for a model with external forces. As previously discussed in section 2.3 it can be expected that the state of hover is no longer stable. Using the simulation, the quadcopter's response to a constant external force can be analysed. Suppose a constant external force $\boldsymbol{F}_{ext} = (1, 0, 0)$ in the x-direction was added to the closed loop model (4.7) with the previously defined feedback (4.6).

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e_3} - R(\boldsymbol{\lambda})(T\boldsymbol{e_3}) + \boldsymbol{F}_{ext} \\ \dot{\boldsymbol{\lambda}} &= Q(\boldsymbol{\lambda})\boldsymbol{\Omega} \\ I\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times I\boldsymbol{\Omega} + \boldsymbol{\tau} \end{aligned} \tag{4.8}$$

If the quadcopter initially where to be in the state of hover $\boldsymbol{y}_0$, we can expect the position to drift away. This is exactly what we observe in Figure 4.3 where this system is numerically solved with initial condition $\boldsymbol{y}_0$. The quadcopter finally stabilizes in a slightly tilted position ($\theta \approx 0.1$ rad) and a slight offset in position ($x \approx 0.1$ m).

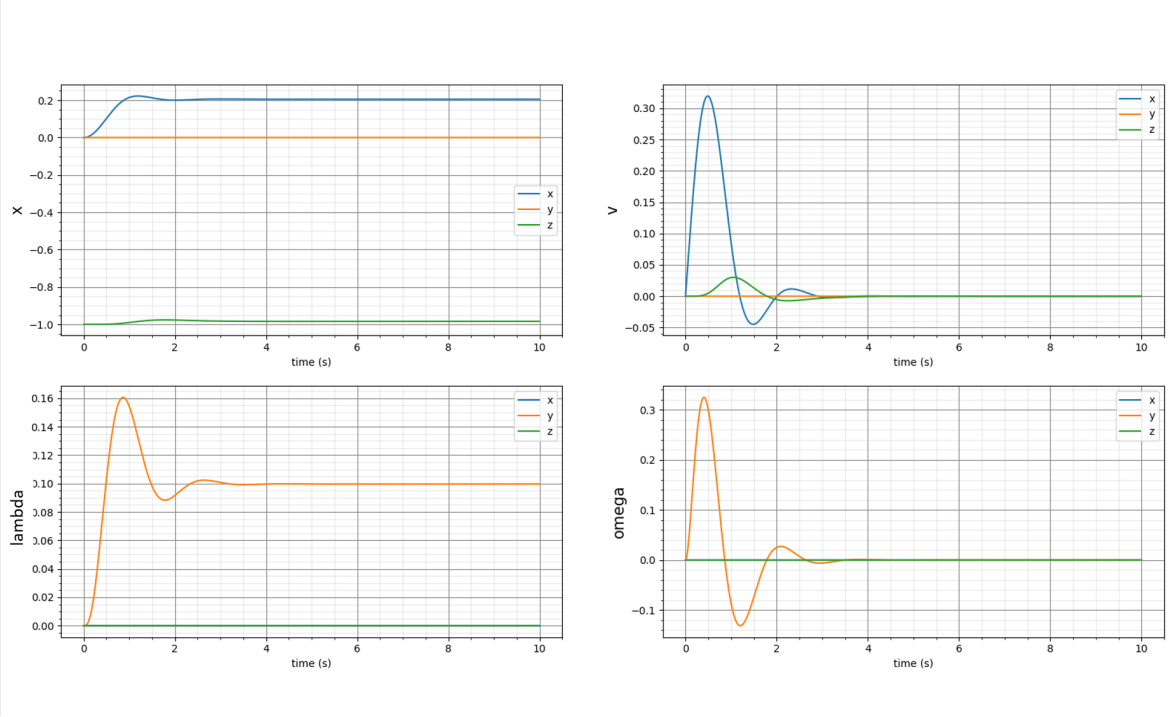


Figure 4.3: The state space trajectory of the system (4.7) with initial condition $\boldsymbol{y} = \boldsymbol{y}_0$. The quadcopter seems to drift away from the desired position.

For this reason we will add integral control to the simulation. Similar to the previous section, we will calculate a feedback matrix K by solving the Riccati equation with linearization matrices A and B substituted. Only this time these matrixes are derived from the augmented system (2.21)[4].

[4]In the linearization of this system, the external forces and torques are set to 0 since they are unkowns...

This results in the following matrices

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 10.0 & 0 & 0 \\ 0 & 0 & 10.0 & 0 \\ 0 & 0 & 0 & 20.0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.9}$$

The Q matrix will now also be a different size. The added variable $\boldsymbol{\sigma}$ can now also be included in the cost function. We will use

$$Q = \mathbf{diag}(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 100 & 1 & 1 & 1 & 100 & 100 & 100 \end{bmatrix}) \quad R = \mathbf{diag}(\begin{bmatrix} 10 & 10 & 10 & 10 \end{bmatrix}) \tag{4.10}$$

Note that now instead of adding extra weight to the variables (x, y, z), the variables $(\sigma_x, \sigma_y, \sigma_z)$ are now emphasised. This means that the controller will mostly minimize the integrated error of the position (2.22), which will prevent the quadcopter from drifting. The extra weight on the variable ψ was experimentally determined to perform better. By again numerically solving the Riccati equation using python we can find the new feedback matrix (rounded to 1 decimal place).

$$K = \begin{bmatrix} 0 & 0 & -4.3 & 0 & 0 & -3.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3.2 \\ 0 & 3.5 & 0 & 0 & 1.9 & 0 & 6.0 & 0 & 0 & 1.1 & 0 & 0 & 0 & -3.2 & 0 \\ -3.5 & 0 & 0 & -1.9 & 0 & 0 & 0 & 6.0 & 0 & 0 & 1.1 & 0 & -3.2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3.2 & 0 & 0 & 0.6 & 0 & 0 & 0 \end{bmatrix} \tag{4.11}$$

Using this feedback law, the system with a constant external force (4.8) will now be able to stabilize around the desired position as can be seen in figure 4.4

that cannot be used by the controller.

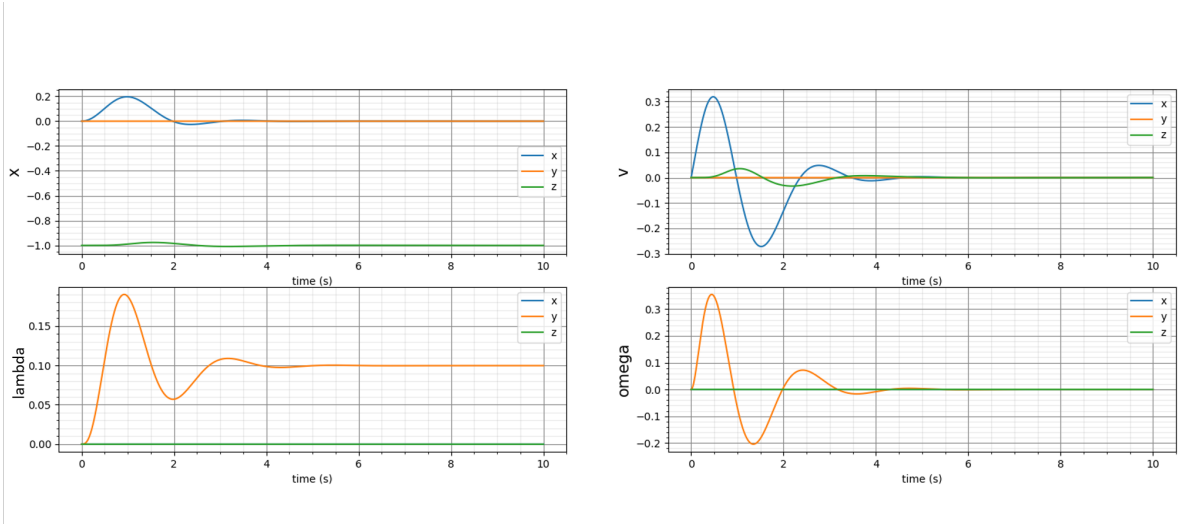


Figure 4.4: The state space trajectory of the model with a constant external force using integral feedback. The initial condition $\boldsymbol{y} = \boldsymbol{y}_0$ is used. Unlike in figure 4.3, the quadcopter now stabilizes at the desired position $\boldsymbol{\xi} = (0, 0, -1)$. The quadcopter is now slightly pitched $\theta \approx 0.1$ in order to compensate for the external force.

This is exactly the behavior that is desired for the quadcopter to be able to stabilize in turbulence. For this reason the feedback (4.11) will be used for the rest of this analysis.

4.1.3 Measuring turbulence

We will now simulate the quadcopter model where the external forces are due to drag in a turbulent fluid. In the simulation we will use the model from section 1.2 where

$$\boldsymbol{F_{ext}} = \alpha(\boldsymbol{v}_{wind} - \boldsymbol{v}) \tag{4.12}$$

$$\boldsymbol{\tau_{ext}} = \beta(\nabla \times \boldsymbol{v}_{wind} - R(\boldsymbol{\lambda})\boldsymbol{\Omega}) \tag{4.13}$$

Note that the values for the drag coefficients where set to $\alpha, \beta = 0.1$ (table 4.1). In the simulation, the velocity and vorticity of the wind $\boldsymbol{v}_{wind}, \nabla \times \boldsymbol{v}_{wind}$ at each moment in time t will be determined beforehand. A time series of turbulent velocities and velocities will be provided by a dataset. This dataset contains the trajectories of particles in a homogeneous and isotropic turbulent velocity field. Along these trajectories, the values of the wind velocities and velocity gradients are known[5]. The Figures 4.5, 4.6 and 4.7 show the simulated response of the controlled quadcopter to these turbulent velocities from one sample of the dataset. The effect of the velocity and vorticity are first simulated separately, and then also combined. In all these cases the quadcopter remains stable. It is important to note however that these velocities are scaled to be of a magnitude of around 2 m/s.

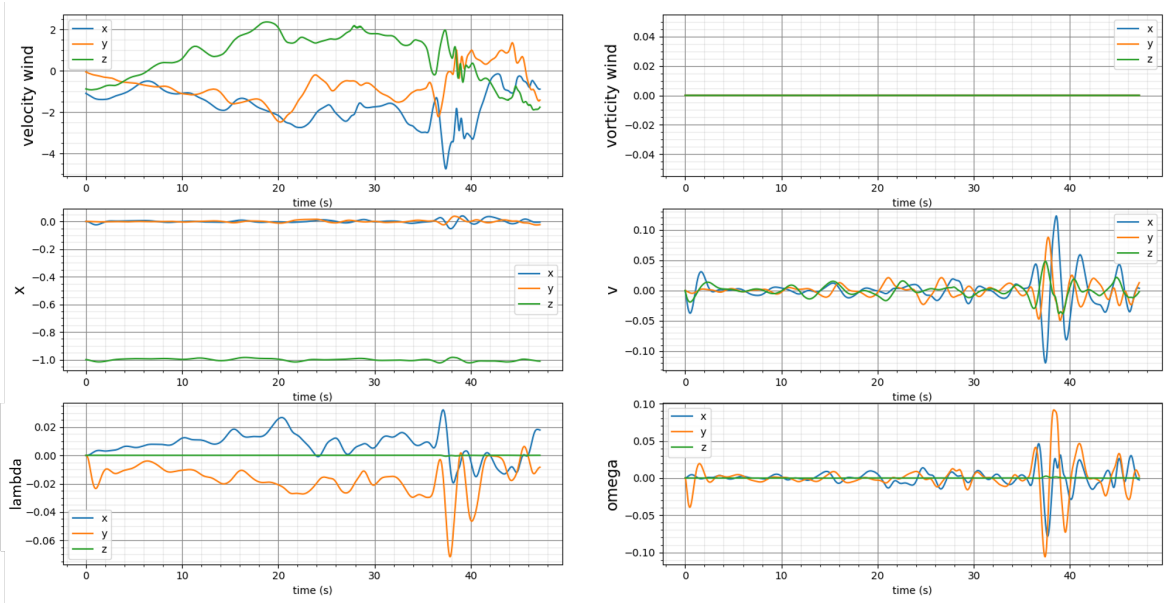


Figure 4.5: State space trajectory of the system subject to turbulent forces caused by the velocity of a turbulent fluid. The top two plots show the velocity and vorticity of the fluid, while the bottom plots describe the state $(\boldsymbol{\xi}, \boldsymbol{v}, \boldsymbol{\lambda}, \boldsymbol{\Omega})$. In this simulation the vorticity is assumed to be zero, so only linear forces are considered. It can be seen that the state oscillates around the state of hover while remaining relatively close.

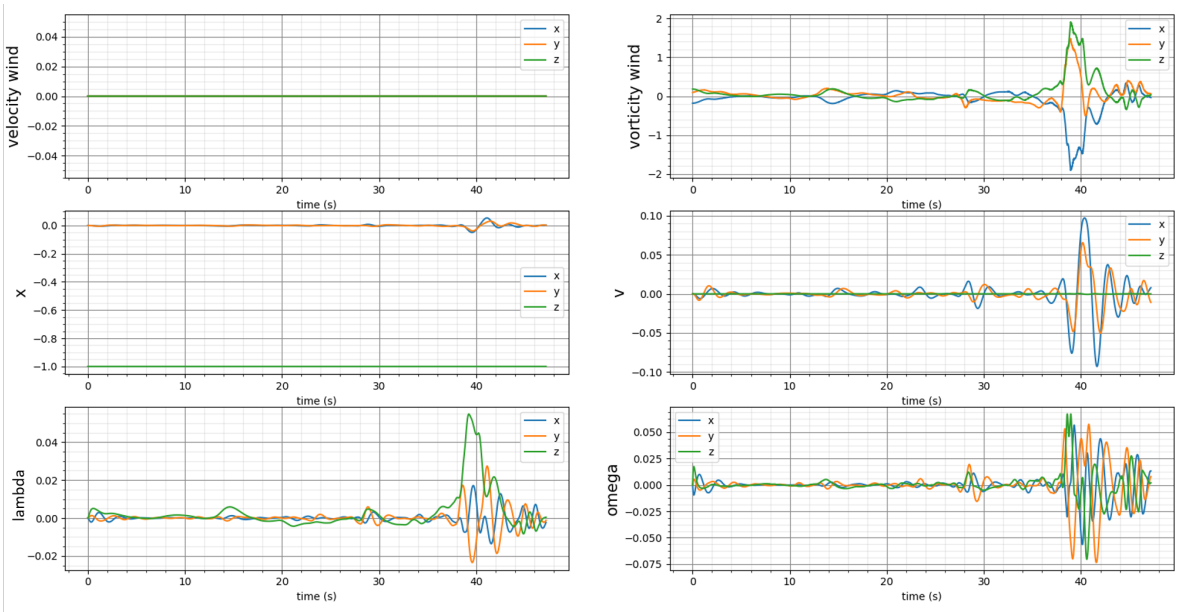


Figure 4.6: In this simulation velocity of the fluid is assumed to be zero, so now only the torques on the quadcopter are considered. It can be seen that the state still oscillates around the state of hover. Especially the oscillations of the angular velocity $(\boldsymbol{\Omega})$ seem to be strong.

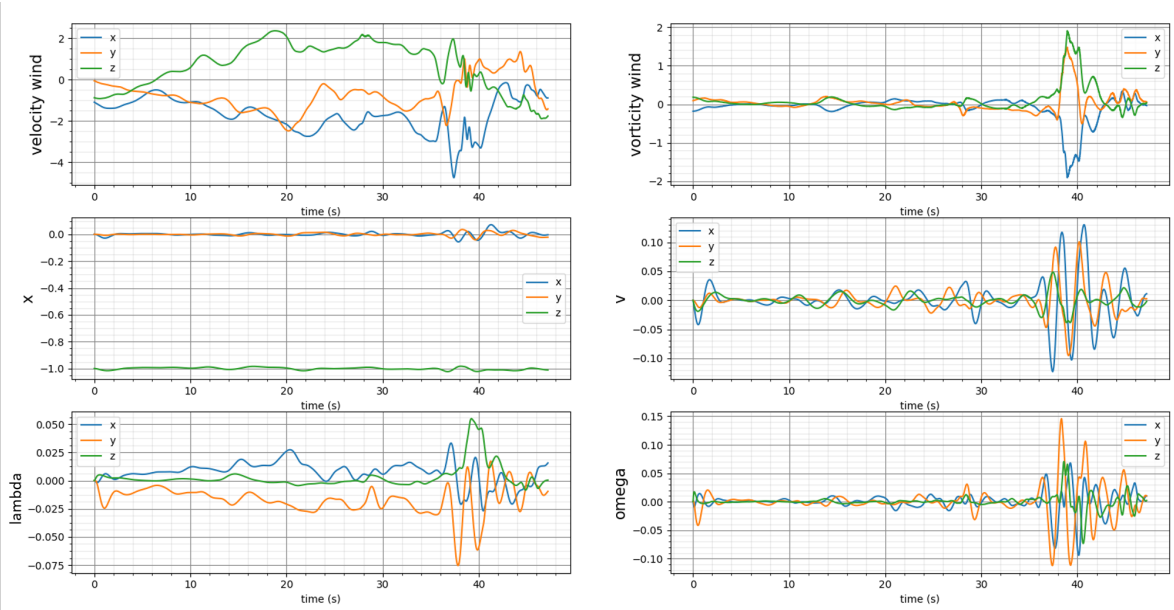


Figure 4.7: In this simulation both the vorticity and the velocity are accounted for. Also now the quadcopter is able to remain close to the desired position.

From these simulated state-space trajectories it is now possible to reconstruct the velocity and vorticity of the fluid using equation 3.5. The result of this can be seen in figure 4.8 where the trajectories from figure 4.7 are used.

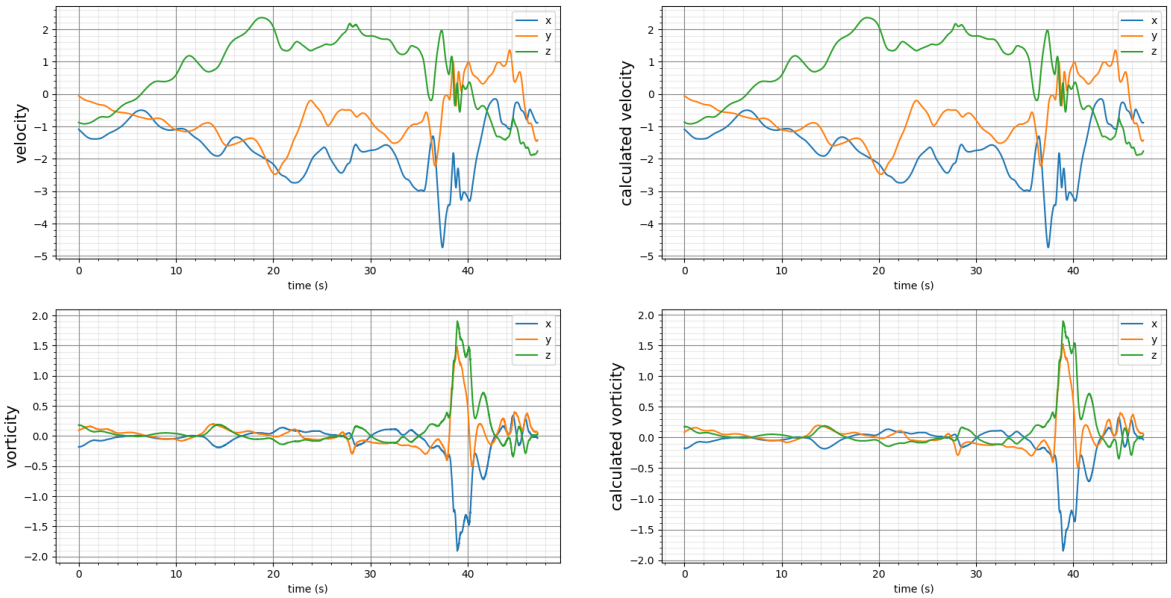


Figure 4.8: The velocity and vorticity calculated from the trajectories in figure 4.7

As expected, the velocities and vorticities can be completely reconstructed based on the state-

space trajectory. Together with the fact that the drone remains stable during the flight, this makes it possible to do a time series measurement of the fluid velocities at a desired point in space.

4.2 Statistical analysis

For this last section, we will statistically analyse the response of the quadcopter to the turbulence. This will be done by simulating state-space trajectories for many turbulence signals from the dataset. In this analysis, only the effect of the wind velocity will be taken into account. The torques caused by the vorticity will be ignored similar to figure 4.5. Using the same dataset as before, we will generate 1000 quadcopter trajectories at 3 different scales. First the fluid velocities will be of the same scale used in the previous section (around 2 m/s), then the velocities will be scaled up by a factor of 5 and 10. An sample of the fluid velocity at these 3 scales can be seen in figure.

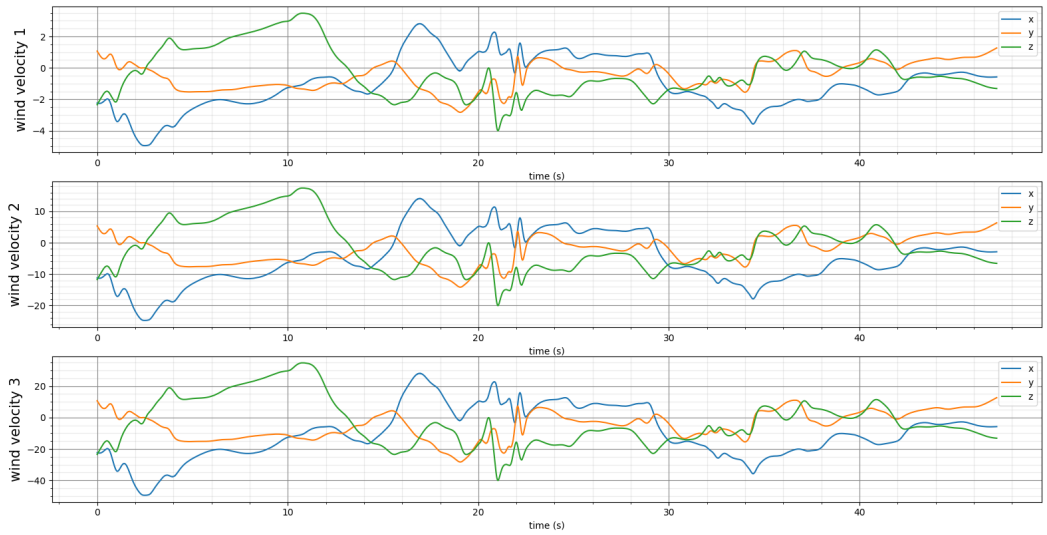


Figure 4.9: Sample of the wind velocity at the 3 different sacles

In total this results in 3 sets of 1000 quadcopter trajectories. These 3 sets are expected to have different statistical properties that can relate the stability of the controller to the intensity of the turbulence. The means and standard deviations of wind and the quadcopter's state is calculated for all these trajectories, and can be seen in table 4.2.

	trajectories set 1		trajectories set 2		trajectories set 3	
wind	mean	std	mean	std	mean	std
$v_{wind,x}$ (m/s)	0.001	1.614	0.004	8.069	0.008	16.138
$v_{wind,y}$ (m/s)	0.052	1.220	0.261	6.100	0.522	12.201
$v_{wind,z}$ (m/s)	-0.042	1.341	-0.209	6.703	-0.418	13.407
state variable	mean	std	mean	std	mean	std
x (m)	0.000	0.013	0.000	0.066	-0.001	0.137
y (m)	0.000	0.012	0.000	0.063	0.003	0.137
z (m)	-1.000	0.015	-1.000	0.074	-0.987	0.181
v_x (m/s)	0.000	0.023	0.000	0.119	-0.003	0.280
v_y (m/s)	0.000	0.022	0.000	0.115	-0.003	0.263
v_z (m/s)	0.000	0.016	0.000	0.083	0.014	0.238
ϕ (rad)	-0.001	0.014	-0.002	0.070	-0.004	0.146
θ (rad)	0.000	0.017	-0.001	0.087	-0.002	0.175
ψ (rad)	0.000	0.000	0.000	0.002	0.006	0.174
p (rad/s)	0.000	0.018	0.000	0.095	-0.001	0.254
q (rad/s)	0.000	0.019	0.000	0.100	0.000	0.256
r (rad/s)	0.000	0.000	0.000	0.009	0.003	0.341
Percentage crashed (%)	0		0		2.5	

Table 4.2: The mean and standard deviation of the state variables and the wind, simulated for the 3 trajectory sets. The values are rounded to 3 decimal places. Here the velocities in trajectory set 1 are scaled by a factor 5 in set 2, and by a factor of 10 in set 3. This scaling can also be seen in the standard deviations of the wind velocity. Note that in all cases the average state of the quadcopter corresponds closely with the desired hover state.

Even though, on average the state of the drone is stable around hover, there are still cases in the dataset where the turbulence is too strong, and the drone 'crashes' to the ground or, in other words the state surpasses $z > 0$. This happened in 25 trajectories from the third trajectory set where the turbulence is strongest. An example of such a trajectory can be seen in figure 4.10. The reason for these failures can be explained based on the design of the controller. Since the control feedback is derived from the equations of motion linearized around the hover state, there will be a region in state-space around the state of hover, where the system is stable. Outside this region however, the stability is no longer guaranteed. For these states the non-linear effects start to play a significant role which might cause the quadcopter to destabilize. It is therefor expected that for strong external forces, the quadcopter might no longer be stable.

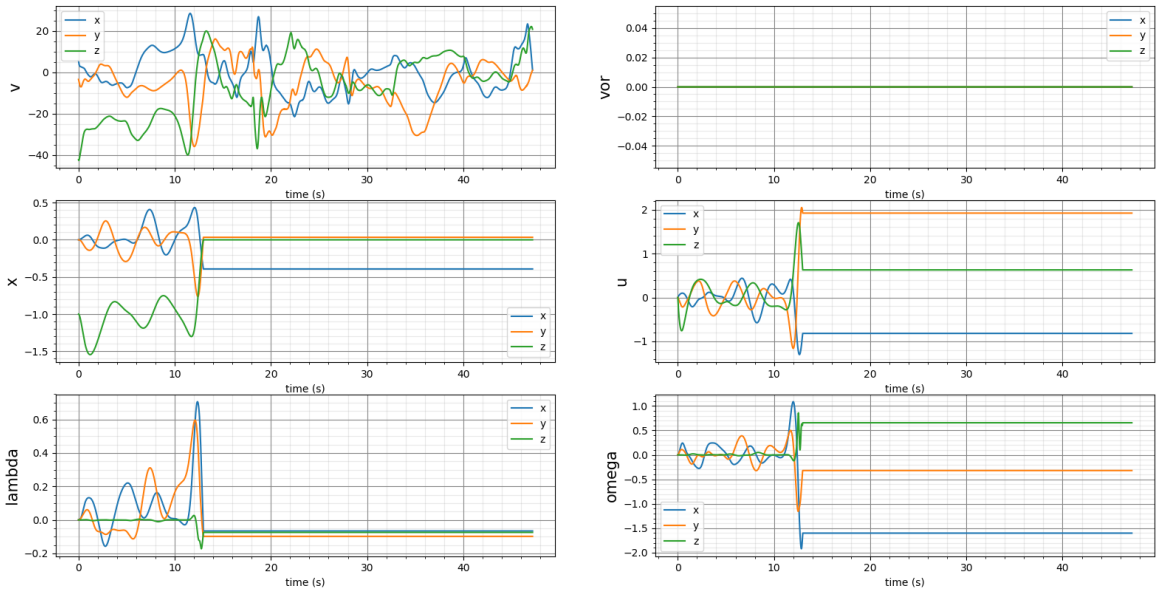


Figure 4.10: A trajectory from trajectory set 3 where the controller fails and the drone reaches the ground ($z = 0$). Once the drone crashes, the drone is simulated to stay on the ground. The instability seems to be caused by a rapid acceleration of the wind in the z- and y-direction.

4.2.1 Acceleration in turbulence

Another interesting statistic is the acceleration of the quadcopter in turbulence. The trajectory of the quadcopter could be analyzed similar to the paper [5] in which statistical properties of turbulence could be derived from the acceleration of particles transported by a turbulent flow. The paper showed among other things that the Reynolds number could be derived from the probability density function of the particles acceleration. Potentially, the trajectory of the controlled quadcopter could also be related to statistical properties of turbulence. The difference in this case is that the quadcopter, unlike the particles, actively responds to the turbulence.

	trajectories set 1		trajectories set 2		trajectories set 3	
variable	mean	standard deviation	mean	standard deviation	mean	standard deviation
$a_{wind,x}$	0.005	1.228	0.025	6.138	0.049	12.276
$a_{wind,y}$	-0.005	1.213	-0.027	6.064	-0.053	12.128
$a_{wind,z}$	0.009	1.1910	0.044	5.956	0.087	11.911
a_x	0.000	0.060	0.000	0.321	0.000	0.652
a_y	0.000	0.058	0.000	0.306	0.000	0.623
a_z	0.000	0.031	0.000	0.160	0.000	0.370

Table 4.3: The mean and standard deviation of the acceleration of the wind a_{wind} and the acceleration of the quadcopter a. The values are rounded to 3 decimal places.

For this analysis we will use the trajectories from set 1,2 and 3. Since the wind velocities used in the simulation come from a dataset of an isotropic turbulent flow [6], it will be expected that the acceleration statistics are similar in all direction. In table 4.3 and 4.11 it can be seen that this is indeed the case. For the accelerations of the drone however, it can be seen that the acceleration in the z-direction has a smaller standard deviation. Also in figure 4.12 it becomes clear that the acceleration in the z-direction has a different distribution then the x- and y-directions. Unlike in the paper [5], the quadcopter's response to turbulence seems to be orientation dependent. Reflecting on the quadcopter model, this behaviour is expected. In hover, the controller can namely directly compensate the turbulent forces in the z-direction by adapting the total thrust T. This will cause the z-acceleration to be quickly dampened. The forces in the x- and y-direction on the other hand, can only be compensated by both tilting the quadcopter and increasing the thrust. But since it will take some time before the quadcopter is in the tilted position, the will be accelerated more before being corrected.

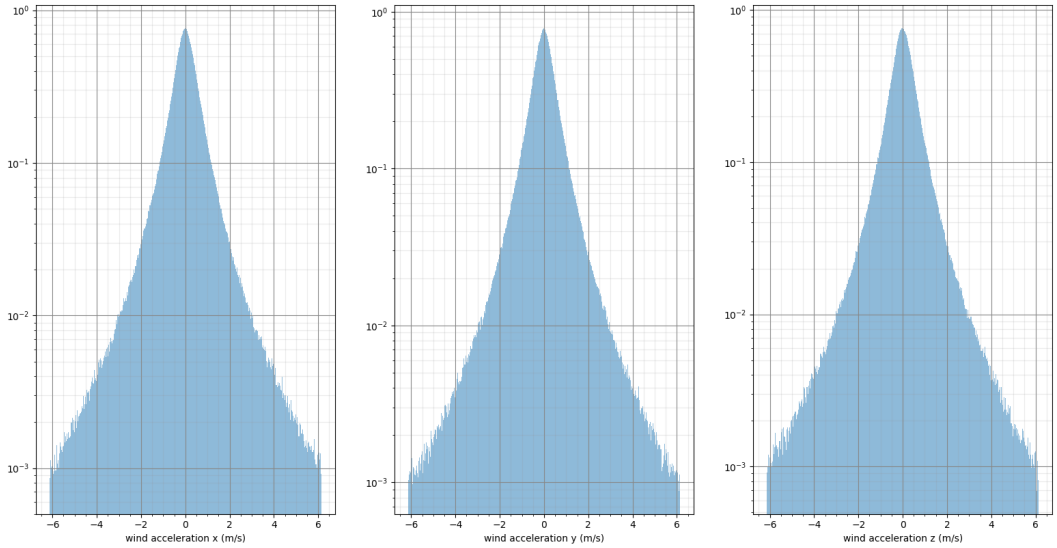


Figure 4.11: Probability density function wind acceleration from set 1. The y-axis is plotted on a log scale. As expected, x, y and z component of the acceleration all have a similar distribution.

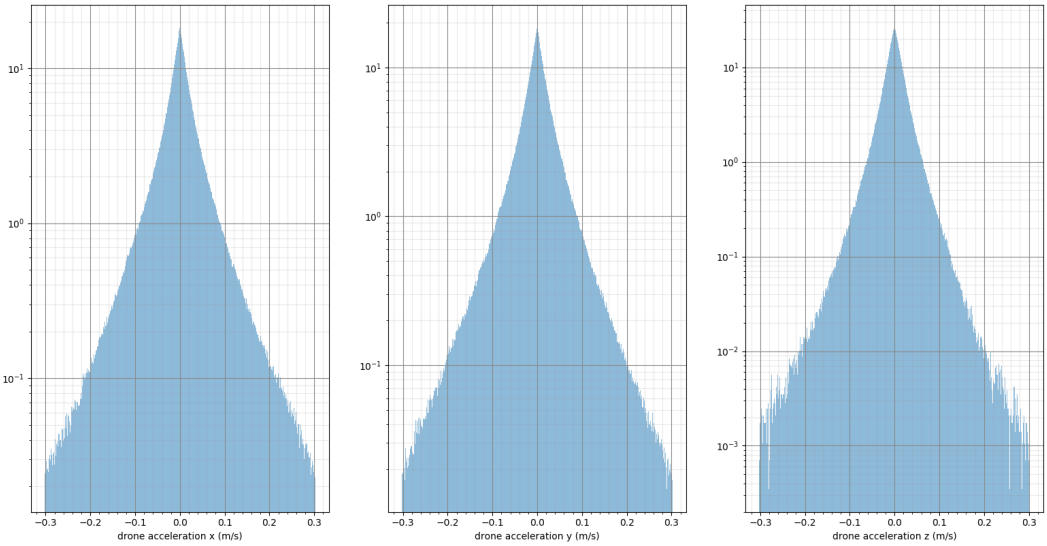


Figure 4.12: Probability density function quadcopter's acceleration from set 1. The y-axis is plotted on a log scale. Unlike the previous figure, the accelerations here are non-isotropic. The quadcopter seems to accelerate less in the z-direction.

From the probability density functions in figure 4.11 and 4.12 and the table 4.3 the following observations can be made

- The probability density function of the acceleration of the quadcopter has a very sharp peak around 0.
- The variance in acceleration of the quadcopter seems to be proportional to the variance

in acceleration of wind (around the quadcopter).

- The variance in acceleration of the quadcopter in the z-direction is roughly equal to half the variance in the x- and y-direction

However, since this analysis is only based on numerical simulations for a specific set of parameters, there is insufficient information to make qualitative predictions on the statistical properties of the turbulence. A subsequent project could shed more light on this by deriving an explicit model for the controlled quadcopter that could make predictions similar to the models used in [5].

5 Conclusions and Further work

The main results of this project can be summarized as follows. In the first chapter a physical model for the quadcopter is derived based on rigid body dynamics together with a simple model for aerodynamic drag. This model is then used to derive an optimal linear state feedback controller using the LQR algorithm. This controller however was not robust to external forces which could cause the quadcopter to drift from the desired position. Combining the LQR algorithm with the method of integral control, resulted in the final proposed algorithm that was both robust to external forces and also capable of keeping the quadcopter close to the desired position (4.4). Finally, using the derived model and controller, expressions for the velocity and vorticity of the wind around the quadcopter were derived (3.5). These expressions proved a way to measure these quantities using the state-space trajectory of the quadcopter. By obtaining this state information using sensor data, this project shows a potential way to use a quadcopter as a probe for atmospheric turbulence.

For a continuation of this project, the following aspects could be expanded upon. First of all, the statistical analysis in this project was not sufficient enough to make qualitative predictions about turbulence. Further research into the statistics of the quadcopter's response to turbulence could give a better insight into the properties of the turbulence. A possible method to do this would be making a simplified predictive model that relate the quadcopter's movement to the turbulence. From this model it might be possible to derive dimensionless numbers that describe the most important properties of the system. Also the drag model that is used in this project could possibly be improved. The current model namely treats the quadcopter as a point particle in a fluid, where the external torques are only due to the vorticity of the fluid at a single point. This model does not completely capture the effects of the smaller scale flows in and around the quadcopter. The quadcopter of course generates its own flows by its propellers causing complicated flows and vortexes. A more advanced model could shed some light on these dynamics. Lastly, another interesting topic for further investigation, would be the controller. While in the current controller only information from the quadcopter's state is used, an improved controller could possible also use information of the measured turbulent forces. Using these measurements it could be possible to respond more quickly, and stay in position more steadily.

References

[1] John R. Taylor. *Classical Mechanics*. University Science Books, 2005.

[2] W.Murray Wonham. *Linear multivariable control, a geometric approach*. Springer, 3 edition, 1979.

[3] George Halikias. Calculus-variations-optimal-control, 2007.

[4] Alan J Laub. A Schur method for solving algebraic Riccati equations, 1978.

[5] L. Biferale, G. Boffetta, A. Celani, B. J. Devenish, A. Lanotte, and F. Toschi. Multifractal statistics of lagrangian velocity and acceleration in turbulence. *Physical Review Letters*, 93(6):1–4, 2004.

[6] Roberto Benzi, Luca Biferale, Enrico Calzavarini, Detlef Lohse, and Federico Toschi. Velocity-gradient statistics along particle trajectories in turbulent flows: The refined similarity hypothesis in the Lagrangian frame. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 80(6):1–6, 2009.

A Full equations of motion

Full equation:

$$
\begin{aligned}
\dot{x} &= v_x \\
\dot{y} &= v_y \\
\dot{z} &= v_z \\
\dot{v_x} &= -(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta)\frac{T}{m} \\
\dot{v_y} &= -(\cos\phi\sin\psi\sin\theta + \cos\psi\sin\phi)\frac{T}{m} \\
\dot{v_z} &= -(\cos\phi\cos\theta)\frac{T}{m} + g \\
\dot{\phi} &= p + \sin\phi\tan\theta q + \cos\phi\tan\theta r \\
\dot{\theta} &= \cos\phi q - \sin\phi r \\
\dot{\psi} &= (\sin\phi/\cos\theta)q + (\cos\phi/\cos\theta)r \\
\dot{p} &= \frac{I_y - I_z}{I_x}qr + \frac{\tau_x}{I_x} \\
\dot{q} &= \frac{I_z - I_x}{I_y}pr + \frac{\tau_y}{I_y} \\
\dot{r} &= \frac{I_x - I_y}{I_z}pq + \frac{\tau_z}{I_z}
\end{aligned}
\tag{A.1}
$$

B Angular velocity to Euler angles

Here follows a derivation of the transformation from angular velocities $\boldsymbol{\Omega} = (p, q, r)$ to the derivatives of the Euler angles $\dot{\boldsymbol{\lambda}} = (\dot{\phi}, \dot{\theta}, \dot{\psi})$ (1.18).

Note that $\boldsymbol{\Omega}$ is described in the body reference frame that is rotated by $R(\boldsymbol{\lambda})$. We will now define a new reference frame that is aligned with the world frame, but with the origin at the center of the drone. In this frame, we can describe the angular velocity as follows

$$\boldsymbol{\omega} = R(\boldsymbol{\lambda})\boldsymbol{\Omega} \tag{B.1}$$

The motion of each point $\boldsymbol{x}$ in the rigid body can now be described in this reference frame as follows

$$\dot{\boldsymbol{x}} = \boldsymbol{\omega} \times \boldsymbol{x} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{B.2}$$

Also, each point can also be described by

$$\boldsymbol{x} = R(\boldsymbol{\lambda})\boldsymbol{x_o} \tag{B.3}$$

Where $\boldsymbol{x_o}$ is the corresponding point in the body frame. The motion can now be described in terms of the Euler angles as follows

$$\dot{\boldsymbol{x}} = \frac{d}{dt} R(\boldsymbol{\lambda})\boldsymbol{x_o} = \dot{R}\boldsymbol{x_o} \tag{B.4}$$

Now since $\boldsymbol{x_o} = R^T\boldsymbol{x}$ we get

$$\dot{\boldsymbol{x}} = \dot{R}R^T\boldsymbol{x} \tag{B.5}$$

Where R and $\dot{R}$ are shorthand for $R(\boldsymbol{\lambda})$ and $\frac{d}{dt}R(\boldsymbol{\lambda})$ respectively. We can now derive the transformation from angular velocities to Euler angles by solving

$$\dot{R}R^T = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{B.6}$$

Substitution of (1.4) gives

$$\dot{R}R^T = \begin{bmatrix} 0 & \sin\theta\dot{\phi} - \dot{\psi} & \cos\theta\sin\psi\dot{\phi} + \cos\psi\dot{\theta} \\ -\sin\theta\dot{\phi} + \dot{\psi} & 0 & -\cos\psi\cos\theta\dot{\phi} + \sin\psi\dot{\theta} \\ -\cos\theta\sin\psi\dot{\phi} - \cos\psi\dot{\theta} & \cos\psi\cos\theta\dot{\phi} - \sin\psi\dot{\theta} & 0 \end{bmatrix} \tag{B.7}$$

Solving equation (B.6) results in the angular velocities

$$\omega_x = \cos\psi\cos\theta\dot{\phi} - \sin\psi\dot{\theta} \tag{B.8}$$

$$\omega_y = \cos\theta\sin\psi\dot{\phi} + \cos\psi\dot{\theta} \tag{B.9}$$

$$\omega_z = -\sin\theta\dot{\phi} + \dot{\psi} \tag{B.10}$$

Or equivalently

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{B.11}$$

Using (B.1) we can now derive $Q(\boldsymbol{\lambda})$. We namely get

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{B.12}$$

$$= \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix}^{-1} R(\boldsymbol{\lambda}) \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} \tag{B.13}$$

$$= \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} \tag{B.14}$$