CONTROLLING OF AN SINGLE DRONE

Hovering the drone during flight modes

AUGUST 28, 2015

TU/E EINDHOVEN
DEPARTMENT OF MECHANICAL ENGINEERING
DYNAMICS&CONTROL
DR. IR. A.A.J. LEFEBER
Emile Biever 0888839

Abstract

Unmanned aerial vehicles are 'small' flying machines without the need of any pilot and is controlled from the ground with the use of a controller with WIFI-connection and is mostly called, drone. Lately the drone become popular due to the growth of the sensor and microprocessor technology, therefore the drone is interesting for surveillance e.g. the army, safety for companies or for private purposes to guarding houses and many other cases. In the near future, drones will have to cooperate to accomplish certain tasks together. This asks for synchronization of drones. In order to start a project width for synchronization of multiple drones, there is first the understanding and controlling of a single drone necessary to extend this project any further. Therefore this project has the main focus of controlling a single drone to maintain hovering. Within this task the mathematical Program MatLab is used.

In this project the AR. Drone 2.0 from the company Parrot is used and falls under the category Quadrotor systems. The Quadrotor is an highly non-linear dynamical system and the derivation of such systems are complex due to the four rotors (inputs) and six degrees of freedom. Therefore the Drone is known as an underactuated system. After the derivation of the dynamical model there is a linearized model developed for the Quadrotor to maintain hovering. To be able to let the Quadrotor hover, there is a controller needed that stabilizes the system for the Quadrotor to stay hovering at the given position. This controller is implemented into the non-linear model and the simulation results are presented. This model is ready for use to implement in the so called software development kit(SDK) what is the next stage of controlling a single drone.

This report gives the non-linear dynamical model of the Quadrotor and a linearized model for the Quadrotor for maintaining hovering. The model is set in such a way that it is ready to develop the controller for the non-linear dynamical model. The future work of this project is to develop the controller and implement the controlled model in the non-linear model before putting the dynamical model into the SDK of the Quadrotor.

Table of Contents

List of figures

1 Introduction

Unmanned aerial vehicles are controlled from the ground with the use of a controller with WIFI-connection and are 'small' flying machines without the need of any pilot and is mostly called, drone. Lately the drone become popular due to the growth of the sensor and microprocessor technology, therefore the drone is interesting for surveillance e.g. the army, safety for companies or for private purposes to guarding houses and many other cases. In the near future, drones will have to cooperate to accomplish certain tasks together. This asks for synchronization of drones.

For this project AR. Drone 2.0 is used and is called from now on Quadrotor. The Quadrotor is a drone that exists with four rotors which can differ in speed from each other. This change gives the drone the possibility to fly in any direction. The drone has six degrees of freedom (DOF) for translation and rotation.

The goal of this project is to develop a mathematical model that gives the Quadrotor the possibility to hover and to prepare the mathematical model to be ready for implementing in the so called software development kit(SDK) to control the Quadrotor. Therefore the need to develop a mathematical model to describe the motion of the Quadrotor is essential. The model itself is not enough. There is the need to develop a controller that stabilizes the Quadrotor at a given position or movement for the purposes not to drift away from given manoeuvres.

In chapter 2 the principles of a Quadrotor drone is discussed with its movements along and around its principle axes. In the third chapter the dynamical model for the Quadrotor is derived by the Euler-Lagrangian method and is compared to a different model that is derived by the Newton-Euler method. After the derivation of the dynamical model is known in chapter 4 the linearized model is discussed to maintain hovering. In chapter 5 the controller is discussed that linearize the Quadrotor to hover. The conclusions are discussed in chapter 6.

2 The Quadrotor system

This chapter gives the general principles of the Quadrotor. This section discusses the possible movements and rotation of the Quadrotor around the principle axes. By varying the rotor-speeds, the Quadrotor can control the movements and rotations along and around the principle axes in order to fly and maintain hovering.

2.1 What is an Quadrotor and how does it work

The AR drone 2.0 is a Quadrotor drone and falls under the category of helicopter vehicles. This Quadrotor is shown in Figure 1 and a schematic view in Figure 2. A Quadcopter possesses four rotors that can deliver a certain thrust force with a certain distance to the center of gravity (cog.) of the drone. By varying the speed of these rotors the delivered thrust-forces can be changed. The drone is now able to maintain stationary hovering or it is able to fly in a certain direction. By varying the rotor speeds the quadrotor is able to pitch(ϕ), roll(θ) and yaw(ψ) around the x-, y- and z-axes respectively. See Figure 2 with the denoted frames of reference and Table 1 for the summary of possible rotations.

Table 1 Rotational movements around the principal axes in the body frame of reference (R_b)

Rotation	Axes rotation in frame R_b
pitch(ϕ)	Rotation around the principal x-axes
roll($\boldsymbol{\theta}$)	Rotation around the principal y-axes
yaw(ψ)	Rotation around the principal z-axes

Due to these four rotors (actuators) the Quadrotor is known as an underactuated system because the Quadrotor has six degrees of freedom (6-DOF). Therefore, due to four outputs (actuators) and six inputs (the 6-DOF), the Quadrotor cannot give an acceleration at any time in a certain direction without changing the current dynamic behavior in another degree of freedom, as explained in [1]. Conventional helicopters [3] fly due to modifying the lift-force vector in both magnitude and direction. This is done by varying rotation speed, angle of blade attack (pitch angle) and cyclic pitch angle (single rotor helicopter with tail-rotor to oppose the induced moment) [2]. For the quadrotor, the pitch angle is fixed which means the only way to introduce lift-force besides varying rotor speed is by the possible deformation of the blades due to the air drag. Figure 2 shows the quadrotor with thrust forces(F_1-F_4), possible rotary motions(ω_1-ω_4) and the axis-system with orientations for the body-frame of the quadrotor drone and inertial frame with respect to the earth.

Figure 1 AR drone 2.0

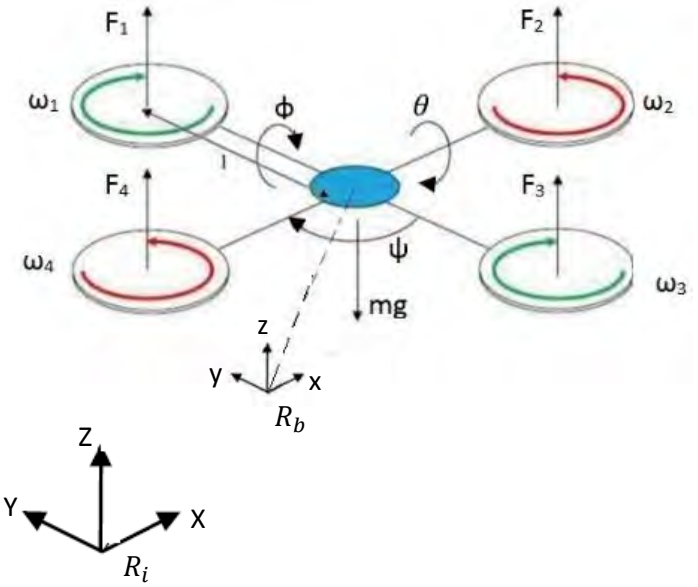

Figure 2 Schematic of the Quadrotor drone with inertial frame R_i with respect to the earth and inertial frame R_b with respect to the body centre of gravity.

The following example shows how the drone moves in forward direction (the explanation is also applicable for opposite direction):

In order to hover with a drone of the quadrotor type the rotor speeds should increase to move up, or keep the rotor speed levelled, to maintain hovering. For flight modes of the quadrotor drone, it is necessary to change the rotor speeds of the four rotors. When the rotor speed of the third and fourth rotor increases the quadrotor gives a pitch movement towards the ground (diving). Therefore, the rotor speeds of the first and second rotor must increase in speed to maintain stable flight (producing counter moment). The quadrotor is able to roll if the rotor speed of the second and third rotor increases. This causes the quadrotor to flip, for maintaining stable flight it is necessary that the first and fourth rotor increases in speed to maintain stable flight. For the change in direction of the quadrotor drone (yaw), the rotor speeds of the second and fourth rotors must increase to move the fuselage counter clockwise. Therefore, if straight flying is needed, the rotor speed of the first and third rotor has to be increased (counter torque).

In this section the introduction is given about the possible movements of the Quadrotor. The given discussion may seem simple in theory but practically there are many factors that has to be taken into count to maintain stable flight (e.g. flying in outdoor environment with many obstacles and harsh winds)

3 Dynamic model of the Quadrotor drone derived with the Euler-Lagrangian method

This chapter explains the mathematical model of the Quadrotor in its dynamical behavior. In the first section the translational and rotational movement of the quadrotor is discussed, section two gives the assumptions made for deriving the mathematical model, followed by the derivation of the non-linear equations from the Quadrotor. The end of this chapter gives the summary of the derived equations from the quadrotor that represents its dynamical behavior.

3.1 Inertial and body frame of reference and its relation in 3D

As mentioned earlier there are two reference frames for the quadrotor, one inertial frame that is fixed with respect of the earth denoted as $\vec{\mathrm{R}}_i$ and one frame of reference attached to the body in the cog. denoted as $\vec{\mathrm{R}}_b$. The body of the quadrotor is able to rotate around its axes of orientation. In order to describe its movement/behavior, a translation is required from the inertial frame to the body frame. Euler-angles and quaternions (Euler-parameters) are two methods that makes this translation possible. Euler-angles uses three different fairly intuitive rotations around its principal axes for pitching(ϕ), rolling(θ) and yawing(ψ). The use of quaternions is more complex because it involves using four parameters with extension to complex numbers to describe the orientation [4,5].

The Euler angles method has the disadvantage that due to describing the system by angles, singularities occur by changing the coordinates when rotating the body frame of reference [5,6]. Where the Euler-angles make use of matrices, quaternions make use of four parameters for representing the orientation which involves high computational effort. Due to the complexity of quaternions this project makes use of the Euler-angle method.

In order to overcome the mentioned singularities the possible rotations of the Quadrotor have to be evaluated. In paragraph 3.4 it becomes clear that the only singularity that could occur is the rolling motion of the Quadrotor with angle (θ). Therefore, the domain of (θ) is $[-\frac{\pi}{2} < \theta < \frac{\pi}{2}]$ is suitable to prevent singularities.

3.1.1 Translation of the Quadrotor system from inertial frame into body frame

Generally the origin of the inertial frame differs from the origin of the body frame of the quadrotor as already discussed above. In order to translate the inertial frame into body frame a derivation is set in order to obtain a rotation matrix [4]. This is done by the xyz-convention (that is one of the twelve possible types of conventions). This means that there is a rotation around the z-axes followed by a rotation around the new obtained y-axes and ending with a rotation around the new obtained x-axes. This procedure is illustrated in Figure 3 on the following page.

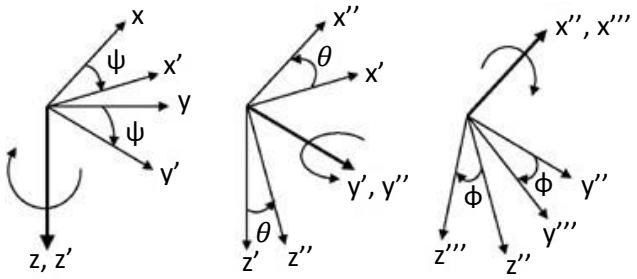

Figure 3 Rotations of the body frame of the quadrotor. In the first stage a rotation around the z-axes with angle ψ result in a coordinate system with z'y'x'-axes. In the second stage a rotation around the new y'-axes with an angle Θ result in a coordinate system z''y''x''-axes. The third rotation around the new x''-axes with an angle ϕ result in a coordinate system with z'''y'''x'''-axes.

It is assumed that, before the quadrotor rotates, the body frame coincides with the inertial frame such that $\vec{\mathrm{R}}_b = \vec{\mathrm{R}}_i$. The first rotation around the z-axes with the yaw(ψ) angle results in a change in the xy-plane and therefore the location of the z-axes does not change. This is shown in the matrix of equation[3.3] by the 'one' placed at the zz-location (third row and third column). This is a 3x3 matrix. In this derivation, S=sin ,and C=cos. The derivation is given by equations [3.3] till [3.6].

Rotation around the z-axes: $$R_1(\psi) = \begin{bmatrix} C_\psi & -S_\psi & 0 \\ S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \qquad [3.3]$$

The second rotation is around the new y'-axes, the roll (θ)-angle. In this situation the location y'-axis does not change and there is only a change in coordinates possible in the new x''z''-plane.

Rotation around the y'-axes: $$R_2(\theta) = \begin{bmatrix} C_\theta & 0 & S_\theta \\ 0 & 1 & 0 \\ -S_\theta & 0 & C_\theta \end{bmatrix}. \qquad [3.4]$$

The last translation is around the new x''-axis by the pitch(ϕ)-angle. In this situation the coordinate of the x''-axis does not change and there is only a change in coordinates possible in the new y'''z'''-plane.

Rotation around the x''-axes: $$R_3(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi & C_\phi \end{bmatrix}. \qquad [3.5]$$

After multiplying the matrixes in the listed order (order matters!) the transformed rotation matrix is obtained which gives the relative position of a point in the body frame of the quadrotor:

$$R = R_1 R_2 R_3 = \begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\phi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi S_\theta \\ C_\theta S_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\phi S_\theta S_\psi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}. \qquad [3.6]$$

With the rotation-matrix in equation [3.6] the translation from the body frame according to the inertial frame is set. This rotation-matrix is derived by use of Euler-angles. These Euler-angles can give singularities, but due to the given domain for theta, the singularity is prevented.

3.2 The assumptions made for the mathematical model

In order to understand the derivation of the mathematical model it is important that the variables used in this project for deriving the mathematical model are known. These variables are placed in four vectors that represent position (ξ), velocity (V), rotation (η), and angular velocity (ω) in the x-, y-, and z-direction. The used vectors are given below.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}; \qquad V = \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \qquad \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}; \qquad \omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix};$$

The variables and its definitions are listed in Table 2.

Table 2 Definitions of the twelve variables used to derive the dynamical model of the quadrotor system

Variable	Definition
x	Position in the x-direction of the inertial frame R_i
y	Position in the y-direction of the inertial frame R_i
z	Position in the z-direction of the inertial frame R_i
u	Velocity in the x-direction of the body frame R_b
v	Velocity in the y-direction of the body frame R_b
w	Velocity in the z-direction of the body frame R_b
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle
p	Body angular velocity (roll)
q	Body angular velocity (pitch)
r	Body angular velocity (yaw)

There are a few assumptions made to simplify the model that describes the Quadrotor motion and to make the model less complex, the following

- The Quadrotor has a rigid body
- The Quadrotor has an infinite stiffness (the drone does not deform after applying forces on the body)
- The inertia tensor of the quadrotor is diagonal
- The thrust forces are linear (The forces vs. the rotation speed of the rotors)
- The internal dynamics in the body frame is not changing

The following points gives further explanation for the made assumptions

The inertia tensor of the quadrotor is diagonal

There exists an alignment frame in the body of the Quadrotor that the inertia tensor is diagonal. This alignment is, when the inertial frame coincides with the body frame principle axes of inertia [5]. For this project it is assumed that the body frame coincides with its principle axes.

The internal dynamics in the body frame is not changing

The body of the Quadrotor has an infinite stiffness which means that the body does not deform after applying forces at the body. Therefore, due to no internal deformation of the material (structure of material stays the same) there is no internal tension of the material resulting in no internal dynamics.

3.3 Dynamics and kinematics of the Quadrotor system

In this section the derivation of a nonlinear dynamical model is discussed. The derivation in this section makes use of the method by Lagrange. The Lagrangian method describes the system (Quadrotor) as whole instead of a system where the individual components are described separately (when describing it via the Newton approach). The dynamics of the Quadrotor is described by two scalars for the kinetic and the potential energy, and the generalized non-conservative forces. These formalism is based on the generalized coordinates (q), with q=[ξ' η']'[1] ϵ $\mathbb{R}^6$. Because in this project two frames of reference are presented that differ from each other due to the translation and rotational movements of the Quadrotor, the system gets a matrix with twelve nonlinear dynamical equations what represent the quadrotor with six DOF's. In order to understand the derivation of the model, the derivation of moving/rotating frames in a 3D-plane is represented in Appendix A [7,8]

As mentioned in paragraph 3.2 the quadrotor has an infinite stiffness and we assume that the quadrotor has a rigid body. This makes the derivation of the dynamical behavior of the quadrotor easier due to no internal dynamics in the body.

There are two equations used that give the dynamical behavior of the quadrotor: the Lagrangian form equation [3.7] and the Lagrangian equation of motion [3.8]. The Lagrangian (L) in [3.7] states that the energy produced by the Quadrotor is equal to the translational and rotational (kinetic)energy (E_k) subtracted with the potential energy (E_p) of the system.

The Lagrangian equation given in [3.8] is used to obtain the generalized non-conservative forces (f_ξ) and moments(τ_η) (discussed at the end of this paragraph). In order to obtain the non-conservative forces and moments, the time derivative is taken of the derivation of the Lagrangian form with respect to the generalized coordinates($\dot{q}$) subtracted by the derivation of the Lagrangian with respect to the generalized coordinates(q)

$$L(q, \dot{q}) = E_{ktrans} + E_{krot} - E_p \qquad [3.7]$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) = \begin{bmatrix} f_\xi \\ \tau_\eta \end{bmatrix}. \qquad [3.8]$$

These two equations are the basis to derive the entire nonlinear dynamical behavior of the quadrotor. The following subsections show the derivation of the model

- Translational kinematics
- Rotational kinematics
- Translational dynamics
- Rotational dynamics
- Forces acting on the quadrotor
- Moments acting on the quadrotor

3.3.1 Translational kinematics

In order to give the velocities for the translational direction of the quadrotor drone, the rotation matrix is multiplied by the velocity vector(V) of $\vec{R}_b$ with respect to $\vec{R}_i$. Therefore multiplying the velocity vector of $\vec{R}_b$ with the rotation matrix (as discussed in paragraph 3.1) gives the translational movement of the quadrotor in its body frame with respect to the inertial frame of reference. This is shown in equation [3.9] with the vector u_b, v_b, and w_b representing the velocities of the body from the

[1] The notation prime ' denotes the transposed matrix

Quadrotor and the vector u_i, v_i, and w_i representing the velocities at the inertial frame with result of the quadrotor velocity with its rotation in equation [3.10]

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = R \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}, \quad [3.9]$$

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \begin{bmatrix} C_\theta C_\psi & C_\psi S_\theta S_\phi - C_\phi S_\psi & S_\phi S_\psi + C_\phi C_\psi S_\theta \\ C_\theta S_\psi & C_\phi C_\psi + S_\theta S_\phi S_\psi & C_\phi S_\theta S_\psi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix} \begin{bmatrix} u_b \\ v_b \\ w_b \end{bmatrix}. \quad [3.10]$$

3.3.2 Rotational kinematics

The rotational kinematics describes the rotation of the Quadrotor body frame with respect to the inertial frame of reference. In order to do so, the angular velocities have to be equal with the time derivative of the vector (η) with the respective transposed rotation($(R_1, R_2, \text{and } R_3)$ to transform the rotation from the inertial frame into the body frame, applied [10]. Because the angular velocity (ψ) has a rotation in the second and third rotation (R_3R_2) it has to be multiplied with this rotation. The angular velocity (θ) has only a rotation in the third rotating frame and therefore needs only to be applied on the third rotating frame(R_3). The angular velocity (ϕ) exists as a coordinate in the body frame and inertial frame of reference and therefore needs no applying of any rotating frame. This is given by equation [3.11].

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_3^T \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_3^T * R_2^T \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}, \quad [3.11]$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix},$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}.$$

Therefore, the result of the time derivative of the rotation matrix(ξ) in the body frame is given in equation[3.12]

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad [3.12]$$

3.3.3 Translational dynamics

The translational dynamics describes the movement of the quadrotor with respect to its acceleration. To describe the translational dynamics equation[3.8] is rearranged for the non-conservative forces only, see equation [3.13]. In this equation m is the mass of the Quadrotor, g is the gravitational force acting on the Quadrotor, e_3 is the third column of the rotation matrix, and $\ddot{\xi} = [\ddot{x} \quad \ddot{y} \quad \ddot{z}]'$.

$$f_\xi = m\ddot{\xi} + mge_3. \quad [3.13]$$

Solving equation[3.13] for acceleration vector $\ddot{\xi}$ with the total thrust force U_1, and adding the aerodynamic disturbance vector $[A_x\ A_y\ A_z]'$ result in equation[3.15] (taken from[16]) on the next page for the acceleration of the Quadrotor in the body frame.

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}\frac{1}{m}(\cos(\psi)\sin(\theta)\cos(\phi)+\sin(\psi)\sin(\phi))U_1 + \frac{A_x}{m}\\ \frac{1}{m}(\sin(\psi)\sin(\theta)\cos(\phi)-\cos(\psi)\sin(\phi))U_1 + \frac{A_y}{m}\\ -g+\frac{1}{m}(\cos(\theta)\cos(\phi))U_1 + \frac{A_z}{m}\end{bmatrix}. \quad [3.15]$$

3.3.4 Rotational dynamics

The rotational dynamics describes the rotation of the quadrotor with respect to its angular acceleration. In order to do so, equation [3.8] is rearranged for the non-conservative moments only. See equation[3.16], where the solution is taken from[16].

$$\ddot{\eta} = M(\eta)^{-1}(\tau_\eta - C(\eta,\dot{\eta})\dot{\eta}). \quad [3.16]$$

In equation [3.16] $M(\eta)$ is the transposed matrix of the first partial derivative of the rotational vector ω (Jacobian matrix from equation[3.12], full explanation given in chapter 4) multiplied with the inertia tensor(I) and the rotational vector ω (equation[3.12]) in normal state. Equation[3.17] gives the formulation.

$$M(\eta) = \omega' I \omega. \quad [3.17]$$

In order to describe the inertia tensor, the quadrotor is modelled as one body. Actually the inertia tensor of the quadrotor is a 3x3 matrix without any zero elements as can be seen in the first part of equation [3.18]. the main diagonal represents the moment of inertia and the other elements outside of the main diagonal represents the products of inertia. As mentioned earlier we assume that the quadrotor is symmetric about every axes what results in a diagonal matrix as can be seen in the last part of equation [3.18] for the Quadrotor. The result of $M(\eta)$ is given in equation[3.19]

$$I = \begin{bmatrix}I_{xx} & I_{yx} & I_{zx}\\ I_{xy} & I_{yy} & I_{zy}\\ I_{xz} & I_{yz} & I_{zz}\end{bmatrix} \rightarrow I_b = \begin{bmatrix}I_{xx} & 0 & 0\\ 0 & I_{yy} & 0\\ 0 & 0 & I_{zz}\end{bmatrix}. \quad [3.18]$$

$$M(\eta) = \begin{bmatrix}I_{xx} & 0 & -I_{xx}S_\theta\\ 0 & I_{yy}C_\phi^2 + I_{zz}S_\phi^2 & (I_{yy}-I_{zz})C_\phi S_\phi C_\theta\\ -I_{xx}S_\theta & (I_{yy}-I_{zz})C_\phi S_\phi C_\theta & I_{xx}S_\theta^2 + I_{yy}S_\phi^2 C_\theta^2 + I_{zz}C_\phi^2 C_\theta^2\end{bmatrix}. \quad [3.19]$$

The last term in equation[3.16] describes the Coriolis(C). The quadrotor makes use of two frames of reference, the body frame and an inertial frame that is fixed with the earth. Normally the two frames can rotate separately from each other which causes what is called the Coriolis effect. Due to the little difference in rotation from the earth on the Quadrotor it is trivial to neglect the rotation of the earth, therefore the Coriolis term is in the body of the Quadrotor and is given in an 3x3 matrix in equation[3.20] on the next page, taken from[16].

$$C(\eta,\dot{\eta}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}. \quad [3.20]$$

$c_{11} = 0,$
$c_{12} = (I_{yy} - I_{zz})(\dot{\theta}C_{\phi}S_{\phi} + \dot{\psi}S_{\phi}^2C_{\theta}) + (I_{zz} - I_{yy})\dot{\psi}C_{\phi}^2C_{\theta} - I_{xx}\dot{\psi}C_{\theta},$
$c_{13} = (I_{zz} - I_{yy})\dot{\psi}C_{\phi}S_{\phi}C_{\theta}^2,$
$c_{21} = (I_{zz} - I_{yy})(\dot{\theta}C_{\phi}S_{\phi} + \dot{\psi}S_{\phi}^2C_{\theta}) + (I_{yy} - I_{zz})\dot{\psi}C_{\phi}^2C_{\theta} + I_{xx}\dot{\psi}C_{\theta},$
$c_{22} = (I_{zz} - I_{yy})\dot{\phi}C_{\phi}S_{\phi},$
$c_{23} = -I_{xx}\dot{\psi}S_{\theta}C_{\theta} + I_{yy}\dot{\psi}S_{\phi}^2C_{\theta}S_{\theta} + I_{zz}\dot{\psi}C_{\phi}^2S_{\theta}C_{\theta},$
$c_{31} = (I_{yy} - I_{zz})\dot{\psi}C_{\theta}^2S_{\phi}C_{\phi} - I_{xx}\dot{\theta}C_{\theta},$
$c_{32} = (I_{zz} - I_{yy})(\dot{\theta}C_{\phi}S_{\phi}S_{\theta} + \dot{\phi}S_{\phi}^2C_{\theta}) + (I_{yy} - I_{zz})\dot{\phi}C_{\phi}^2C_{\theta} + I_{xx}\dot{\psi}S_{\theta}C_{\theta} - I_{yy}\dot{\psi}S_{\phi}^2C_{\theta}S_{\theta} - I_{zz}\dot{\psi}C_{\phi}^2S_{\theta}C_{\theta},$
$c_{33} = (I_{yy} - I_{zz})\dot{\phi}C_{\theta}^2S_{\phi}C_{\phi} - I_{yy}\dot{\theta}S_{\phi}^2C_{\theta}S_{\theta} - I_{zz}\dot{\theta}C_{\phi}^2C_{\theta}S_{\theta} + I_{xx}\dot{\theta}C_{\theta}S_{\theta}.$

The dynamical model is at this stage almost derived except for the forces and moments that is discussed in the following two subsections. The results from the given equations [3.13] and [3.16] are of most importance for describing the quadrotor as a non-linear dynamical system.

3.3.5 Forces acting on the Quadrotor

In equation[3.21] the forces are given that act on the body-frame of the Quadrotor. In this equation the force(F) is the net result of all the forces acting on the body and the produced thrust force from the rotors. Equation [3.21] can be separated in the forces that has to be surmounted in order to flight and maintain hovering, denoted by the thrust-forces(F_t) and the external forces, gravitational force(F_g) and the drag force(F_d), are the forces that has to be surmounted.

$$F = F_g + F_t + F_d. \quad [3.21]$$

The gravitational force in equation[3.21] acts in the negative z-direction of the body-frame of the Quadrotor as shown in Figure 2. The gravitational force is given in equation[3.22]

$$F_g = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}. \quad [3.22]$$

The thrust forces in figure 2 on page 1 are acting in the positive z-direction (upwards). Therefore column(e_3) of the rotation matrix as used in equation[3.13] is multiplied with the thrust forces of the four rotors, gives the thrust forces acting along the principle axes. This is shown in equation[3.23] with the result in equation[3.24].

$$F_t = R(e_3)\begin{bmatrix} F_{t,x} \\ F_{t,y} \\ F_{t,z} \end{bmatrix}, \quad [3.23]$$

$$F_t = \begin{bmatrix} S_{\phi}S_{\psi} + C_{\phi}C_{\psi}S_{\theta} \\ C_{\phi}S_{\theta}S_{\psi} - C_{\psi}S_{\phi} \\ C_{\theta}C_{\phi} \end{bmatrix}\begin{bmatrix} F_{t,x} \\ F_{t,y} \\ F_{t,z} \end{bmatrix},$$

$$F_t = \begin{bmatrix} F_{t,x}(\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi)) \\ F_{t,y}(\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi)) \\ F_{t,z}(\cos(\theta)\cos(\phi)) \end{bmatrix}. \quad [3.24]$$

From now on the thrust force vector $[\mathrm{F_{t,x}}\ \mathrm{F_{t,y}}\ \mathrm{F_{t,z}}]$ is given as U_1, this is one of the four main control inputs of the system. Further explanation is discussed in chapter 4.

The aerodynamic drag forces acts on the body frame in the opposite direction of the unit velocity vector($\frac{\mathrm{V}}{|\mathrm{V}|}$) in the quadrotor body frame. This is given in the following equation[3.24] with (V) the velocity vector in the body-frame given in paragraph 3.2.

$$\mathrm{F}_d = -\frac{1}{2} c_f \rho A_f V^2 \frac{\mathrm{V}}{|\mathrm{V}|}. \qquad [3.24]$$

It is assumed that the drag coefficient is constant in Equation[3.24] and therefore proportional to the squared fluid angular velocity given in equation [3.25] for any of the four rotors with k_f being an arbitrary constant [3,11,12]

$$\mathrm{F}_d = \mathrm{k}_f \omega_i^2. \qquad [3.25]$$

From now on the force F_d is presented in the vector $[A_x\ A_y\ A_z]'$ in the direction along the axes of the body frame.

3.3.6 Moments acting on the Quadrotor

The moments acting on the quadrotor can be separated into two parts, namely: the moments produced by the four rotors that react on the body of the quadrotor system and the aerodynamic torque that is produced by the air fluid. These aerodynamic moments can be separated into three parts for torque caused by the drag, gyroscopic torque and torque caused by velocity differences between the rotors and the wind-velocity. The result in the end of this subsection shows that the aerodynamic moments acting around the z-axes is sufficient enough for this nonlinear dynamical model [9].

The moments produced by the four rotors can easily be combined with an arbitrary constant (k_f) for drag [3]. This is shown in equation [3.26]-[3.28] here are the forces F similar to the forces F_1-F_4 shown in Figure 2 and multiplied with the distance(l) from the Cog gives the produced torque by the rotors around the x-,y-, and z-axes. The other parameters are discussed in the end of this subsection. The summation of M_{Fi} is produced by the four rotors that produce an torque around the z-axes.

$$\tau_\phi = k_f(\mathrm{F}_3 - \mathrm{F}_1)\mathrm{l} \qquad [3.26]$$

$$\tau_\theta = k_f(\mathrm{F}_4 - \mathrm{F}_2)\mathrm{l} \qquad [3.27]$$

$$\tau_\psi = -\gamma r + k_t k_f \textstyle\sum_{i=1}^{4} \mathrm{M_{Fi}} \qquad [3.28]$$

The amount of drag force is dependent on the shape of the rotor blades and is an second order effect [9]. In order to find this second order effect the torque has to be described as a function of the rotor angular velocity. This is shown in the end result of the matrix for the whole quadrotor system (equation [3.32]).

Gyroscopic torque is produced by the spinning rotors where this torque moves towards the center of the quadrotor that is moving with an angular velocity (ω). The torque acting on the centre of mass is given in the following equation [3.29]

$$\mathrm{T_g} = -I_{zz}[\omega_b \times (-e_3\omega_1) + \omega_b \times (-e_3\omega_2) + \omega_b(-e_3\omega_3) + \omega_b(-e_3\omega_4)]. \qquad [3.29]$$

This is the equation given for the gyroscopic torque where $e_3$is the z-direction of the body reference frame. This formula is simplified in a factor γr that reacts around the z-direction of the quadrotor. This is given in equation [3.30] (same factor is used in equation [3.28]).

$$\tau_d = (0,0,-\gamma r). \qquad [3.30]$$

The last torque that is discussed is the torque by velocity differences. This torque is explained by the following [9]: the Quadrotor is moving with a velocity not equal to zero. This means that the air velocity at the advancing rotor blades is higher than the receding rotor blades. This effect causes the thrust vectors point of attack on the rotor disk to move from the center and causes the blades to flap. This flapping is dependent on angular velocity of the rotor blades and the velocity of the Quadrotor. This torque by velocity differences is given in equation [3.31]:

$$\mathrm{T_v} = -\alpha[V' \times (e_3(\omega_{\mathrm{M4}} - \omega_{\mathrm{M2}} + \omega_{\mathrm{M3}} - \omega_{\mathrm{M1}}))], \qquad [3.31]$$

where alpha is a factor depending on the design of the rotor blades and ω_{Mi} the respective rotor blade. This torque exists in the body of the quadrotor but due to the fact that the torques cancel each other out (clockwise rotor blades and counter clockwise rotor blades) due to the symmetry of the frame about the velocity factor in the x-, y, and z-plane, equation [3.31] can be neglected and the former equations [3.26]-[3.28] and equation [3.30] is used.

This paragraph showed the derivation of the mathematical model of the Quadrotor by the Euler-Lagrangian method. This derivation is done in a straight forward way and is extended with the forces and moments due to the aerodynamic effects.

3.4 The dynamical model with thrust forces and aerodynamic forces/moments

In the former paragraphs the non-linear dynamical model of the Quadrotor is discussed. This model uses a inertial frame fixed with the earth and a frame in the body of the Quadrotor. The derivation of the model is derived in a straight forward way by using the Euler-Lagrangian method and represents the movements of the body frame according to the inertial frame. As mentioned in paragraph 3.1 the domain for (θ) to prevent singularities is set by $[-\frac{\pi}{2} < \theta < \frac{\pi}{2}]$.

The dynamical system is given in equation [3.32] on the next page and the definitions of the parameters are listed in Table 3. This equation consists of ten rows representing the translational and rotational motion of the Quadrotor with 6-DOF. This matrix is set by collecting the results of the former subsections in paragraph 3.3. Hereby the non-linear dynamical model of the Quadrotor is derived.

$$\begin{bmatrix} \dot{\mathrm{x}} \\ \dot{\mathrm{y}} \\ \dot{\mathrm{z}} \\ \ddot{\mathrm{x}} \\ \ddot{\mathrm{y}} \\ \ddot{\mathrm{z}} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{\eta} \end{bmatrix} = \begin{bmatrix} \dot{\mathrm{x}} \\ \dot{\mathrm{y}} \\ \dot{\mathrm{z}} \\ \frac{1}{\mathrm{m}}(\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi))\mathrm{U}_1 + \frac{\mathrm{A_x}}{\mathrm{m}} \\ \frac{1}{\mathrm{m}}(\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi))\mathrm{U}_1 + \frac{\mathrm{A_y}}{\mathrm{m}} \\ -\mathrm{g} + \frac{1}{\mathrm{m}}(\cos(\theta)\cos(\phi))\mathrm{U}_1 + \frac{\mathrm{A_z}}{\mathrm{m}} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \mathrm{M}(\eta)^{-1}(\tau_\eta - \mathrm{C}(\eta,\dot{\eta})\dot{\eta}) \end{bmatrix} \qquad [3.32]$$

Table 3 Variables with the given definitions of the mathematical model of the Quadrotor

Variable	Definition	Variable	Definition
$\dot{x}, \dot{y}, \dot{z}$	Velocities in the body frame	U_1,U_2,U_3,U_4	Control inputs
$\ddot{x}, \ddot{y}, \ddot{z}$	Acceleration in the body frame	A_x, A_y, A_z	Aerodynamic disturbances
ϕ, θ, ψ	pitch, rolling and yaw- angle around the principle axes of the body frame	m	Mass of the quadrotor
$\dot{\phi}, \dot{\theta}, \dot{\psi}$	Angular velocities of the Quadrotor around the principle axes of the body frame	g	Gravitational constant
$\ddot{\eta}$	Angular accelerations of the Quadrotor around the principle axes of the body frame		

The mathematical model in this chapter is derived by the Euler-Lagrangian method as mentioned above. This model could also be derived in the Newton-Euler way, this gives the same result as given in equation[3.32]. In this report the method by Newton-Euler is not discussed. As can be seen in equation[3.33] taken from [12,17] the result is similar to the obtained mathematical model by the Euler-Lagrangian method.

$$\dot{\xi} = v$$
$$m\dot{v} = R_{E_z} T_f - mgE_z \qquad [3.33]$$
$$\dot{R} = R\hat{\Omega}$$
$$I\dot{\Omega} = -\Omega \times I\Omega + \tau$$

The model in equation[3.33] is not extended with the Coriolis-effect and the aerodynamic forces used as in equation [3.32]. with $\xi = [x\ y\ z]'$ the position of the Cog, v the velocity vector of the Quadrotor along the principle axes, R_{E_z} the third column of the rotation matrix, T_f the thrust force, Ω the angular velocities, I the inertia tensor, and τ being the torques produced by the propellers aerodynamic drag and the distance of the arm length of the Quadrotor. After the derivation of equation[3.33] the result lead to the same expression as given in equation[3.32]

In this chapter the non-linear dynamical model of the Quadrotor is derived by the Euler-Lagrangian method. The model given in equation[3.32] with 6-DOF and twelve non-linear dynamical equations describes the behavior of the Quadrotor. The fact that the same result obtained by the Newton-Euler method as in the Euler-Lagrangian method, gives reliance in the proceeding of this project.

4 Linearization of the Quadrotor for maintaining hovering

This chapter explains the linearization of the hovering Quadrotor. LTV-systems are systems where the output of system differs explicitly on time and is used for e.g. non-linear systems to linearize around a certain operating point, or to follow a certain (optimal) trajectory. An system can be linearized when there is a "local", linear model around an operating point. By using this structured method, the next step to control the non-linear model of the Quadrotor can be done comparatively easy. The controlling of the linearized model is discussed in chapter 5.

Before the discussion of the linearization of the hovering Quadrotor, the principles of the method for linearization is discussed. To find the linearized system the following equation[4.1] should be derived. In this equation the term Ax is the state transition, whit A being the matrix A ϵ $\mathbb{R}^{nxn}$, derived of the non-linear model with respect to variables, and x the difference between the initial values and the reference point(s). The Term Bu is the input state, with B being the matrix B ϵ $\mathbb{R}^{nxm}$, the derived input matrix that has to be controlled and u is the difference between the initial input value and the reference value.

$$\dot{x} = A\Delta x + B\Delta u. \qquad [4.1]$$

Equation[4.1] is the result of deriving the non-linear mathematical model of the Quadrotor into a system $\dot{x} = f(x, u)$ into $\dot{x} = f(0,0) = 0$. In order to describe this system to be zero, the following is true[18,19]

$$\dot{\tilde{x}} = \dot{x} = f(x, u) = f(\tilde{x} + x^*, \tilde{u} + u^*) = \tilde{f}(\tilde{x}, \tilde{u}). \qquad [4.2]$$

In equation[4.2] the symbols $\tilde{x}$ and $\tilde{u}$ are the initial conditions and the symbols x^* and u^* are the reference conditions. This equation is not in the proper state to work with, in order to be useful the equation has to be derived into a Taylor expansion as shown in equation[4.3]

$$\dot{\tilde{x}} = \dot{x} = \tilde{f}(\tilde{x}, \tilde{u}) + \frac{\partial f}{\partial x}(\tilde{x}, \tilde{u})\Delta x + \frac{\partial f}{\partial u}(\tilde{x}, \tilde{u})\Delta u. \qquad [4.3]$$

As described the first term in equation [4.2] and [4.3] $(f(0,0) = 0)$ is zero. The second term is called the state transition form and becomes the A-matrix, and the third term is the input state that becomes the B-matrix as shown in equation[4.1]. In reality the form that is given in equation[4.3], has some higher order terms but for this matter there is only interest in first order terms, in other words, this equation is called the Jacobian of $\dot{x}$. The Jacobian is given in matrix form (matrix A and B) and shows only the derivation of the first order. The Jacobian matrix used in equation [4.1] is given in equation[4.4]

$$\dot{x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \Delta x + \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_n} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial u_1} & \frac{\partial f_m}{\partial u_2} & \cdots & \frac{\partial f_m}{\partial u_n} \end{bmatrix} \Delta u. \qquad [4.4]$$

4.1 Linearized models of the Quadrotor

In this paragraph the linearization of the hovering Quadrotor is discussed. As first the non-linear model in matrix form given in equation [3.32] is set in the form given in equation [4.2]. In the second part the fixed points are determined to be able to let the Quadrotor hover. In the third and last part derivation is applied for the state matrix A and the input matrix B in the following sections

- Dynamics of the system
- Fixed point for the hovering Quadrotor
- Linearization of the state matrix A and the input matrix B

Dynamics of the system

In chapter 3 the non-linear dynamical model of the Quadrotor is derived, with result given in equation[3.32]. In order to translate this model of the body of the Quadrotor in the proper state, the time derivatives has to be set equal to the correct entries of the vector x (observe that the x used here is different than the x used for the position of the Quadrotor in x) and the vector U, namely: the state vector and the input vector. This is given in equation[4.5] for the state matrix and the input matrix is given in equation[4.6]

$$\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix}, \rightarrow \dot{x} = f(x,u) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{\eta} \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ \frac{1}{m}(\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi))U_1 + \frac{A_x}{m} \\ \frac{1}{m}(\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi))U_1 + \frac{A_y}{m} \\ -g + \frac{1}{m}(\cos(\theta)\cos(\phi))U_1 + \frac{A_z}{m} \\ x_{10} \\ x_{11} \\ x_{12} \\ M(\eta)^{-1}(\tau_\eta - C(\eta,\dot{\eta})\dot{\eta}) \end{bmatrix}. \quad [4.5]$$

$$U = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}. \quad [4.6]$$

From Equation[4.5] and [4.6], the matrices A and B from equation[4.1] can be derived. For the Quadrotor to hover is the assumption made that the velocity of the Quadrotor is zero and therefore no acceleration in the translational and rotational modes. In general the Δx and ΔU vectors of the Quadrotor is as follows, see equation[4.7]

$$\Delta x = \begin{bmatrix} x - x^* \\ y - y^* \\ z - z^* \\ \dot{x} - 0 \\ \dot{y} - 0 \\ \dot{z} - 0 \\ \phi - 0 \\ \theta - 0 \\ \psi - 0 \\ \dot{\phi} - 0 \\ \dot{\theta} - 0 \\ \dot{\psi} - 0 \end{bmatrix}, \qquad \Delta U = \begin{bmatrix} U_1 - U_1^* \\ U_2 - U_2^* \\ U_3 - U_3^* \\ U_4 - U_4^* \end{bmatrix}. \qquad [4.7]$$

Fixed point for the hovering Quadrotor

Equation[4.1] is found with the proper states and can be solved for the situation $f(0,0) = 0$, solving for this solution (x_1 up to x_{12} equal to zero) results in the system with the fixed points (x^*, u^*) for the Quadrotor to hover. See equation[4.8]

$$f(0,0) = \begin{bmatrix} -g + \frac{U_1 + A_z}{m} \\ \frac{U_2}{I_{xx}} \\ \frac{U_3}{I_{yy}} \\ \frac{U_4}{I_{zz}} \end{bmatrix} = 0, \quad \rightarrow \quad \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} mg - A_z \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0. \qquad [4.8]$$

Linearization of the state matrix A and the input matrix B

The method used to derive the matrix A and B is done with the symbolic toolbox and the Jacobian statement in the MatLab environment. As said in the beginning of this chapter the linearization is done with the Jacobian matrix. Therefore the first order derivative is taken from the state and input matrix given in equation [4.5]. The state matrix A $\in \mathbb{R}^{12x12}$ and B $\in \mathbb{R}^{4x12}$. The higher order terms of the derivatives are not taken into account. For the state matrix A, the equations are derived with respect to x_1 up to x_{12} and for the input matrix B the equations are derived with respect to U_1 up to U_4. The matrix A and B is given in equation[4.9] and [4.10] on the next page respectively. The MatLab code is given in Appendix B.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{A_x - (A_z - mg)}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{A_y + (A_z - mg)}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \qquad [4.9]$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix}. \quad [4.10]$$

4.2 Concluding remarks to the linearized model

In this chapter the linearized model for the Quadrotor is derived to be able to hover. First the principles of the linearization is discussed followed up with determining the matrix that has to be derived to obtain the linearized model. To do so, the fixed points are determined to be able for the Quadrotor to hover. After the fixed points are known the linearization is conducted with results of the state matrix A and the input matrix B. The derivation of the linearized model is done within the MatLab environment. The MatLab code is given in Appendix B. In [20] a slightly different linearization is given as derived in this chapter because of the numerical solution. This gives reliance to start with controlling the linearized model for the non-linear dynamical system of the Quadrotor in chapter 5.

5 Controlling the Quadrotor for hovering

In this chapter the controlling of the linearized model as discussed in chapter 4 is discussed. It is important to know as a first step if the linearized model as derived for the hovering case is controllable or not. This is discussed in paragraph 5.1. After it is shown that the given system is completely controllable the controller can be designed for the proper gains to use for stabilizing the linearized model. This is discussed in paragraph 5.2 with determining the eigenvalues of the system and the pole-placements.

5.1 Controllability of the linearized model

In order to develop a controller for strongly non-linear dynamical systems as a Quadrotor the non-linear dynamical system need to be controllable. In the case the system is not controllable the linearized model has to be changed in such a way, the system could be controllable. To check if the linearized model is controllable whether or not, the controllability matrix E is used with the state matrix A and input matrix B of the linearized model , see equation[5.1]

$$\mathrm{E} = [\mathrm{B}\ \mathrm{AB}\ \mathrm{A}^2\mathrm{B}\ \cdots\ \mathrm{A}^{\mathrm{n}-1}\mathrm{B}]. \qquad [5.1]$$

The two matrices A and B with twelve rows is derived in chapter 4 and can be placed in equation[5.1] to check if the system is controllable or not. The system is said to be controllable if the controllability matrix E is full rank.

The linearized model in chapter 4 is placed in the controllability matrix E with result that the matrix E is full rank, with exceptions $\mathrm{A_x} - (\mathrm{A_z} - \mathrm{mg}) = 0$ and $\mathrm{A}_y + (\mathrm{A_z} - \mathrm{mg}) = 0$. This means that the derived system is completely controllable and therefore a controller can be developed. The controllability is checked with the controllability statement within the MatLab environment, see Appendix B for the code used in MatLab.

5.2 Linear controller for stabilizing the linearized model

In the preceding subparagraph the controllability matrix E shows the linearized model of the Quadrotor that is controllable due to the full rank of Matrix E. Therefore it is possible to develop a controller to stabilize the Quadrotor system. In order to stabilize the linearized system the x-vector needs to go to zero which means the Quadrotor becomes stable and no oscillatory motion is visible, therefore the initial value for x becomes equal to the reference value of x^*.

To determine if the signal need to be enhanced or weakened, the input of the Quadrotor needs to be multiplied by the scalar K that represents the gains for the Quadrotor system. The form is given in equation[5.2].

$$\mathrm{U} = -\mathrm{kx}, \quad \rightarrow \quad \tilde{\mathrm{U}} = -\mathrm{k}\tilde{\mathrm{x}}. \qquad [5.2]$$

It is shown in chapter 3 that the Quadrotor has multiple inputs and multiple outputs (MIMO) therefore the following steps must be taken to control the system[22]

Given (A,B) controllable $A \in R^{nxn}$, $B \in R^{nxm}$ with $m > 1$

1) Find $F \in R^{mx1}$ and $N' \in R^{mxn}$ such that (A+BN' , BF) is controllable

2) Apply single input pole placement on the system (A+BN' , BF). This gives N" ϵ R^{1xn} such that (A+BN')+(BF)N" the appropriate characteristic polynomial.

3) N=N'+N"

To determine the values for the gains, the eigenvalues of the linearized model should be determined and a given polynomial where the poles has to lie in the plane. In order to be a 'stable' system the eigenvalues(poles) should be lying in the left-half complex plane. The eigenvalues are calculated by the determinant given in equation[5.3]. with the two matrices as shown in step 2 and K being the gain scalar and a variable and solving for λ with the determinant (det) equal to zero gives the proper values for the gains one up to twelve (the gain matrix need to be equal to the same amount of entries of matrix A).

$$\det(\lambda \mathrm{I} - (\mathrm{A} + \mathrm{BN'} - \mathrm{BFK})). \qquad [5.3]$$

The poles are chosen to lie at the following place

$$(\lambda + 1)^{12} = 0 \qquad [5.4]$$

Solving equation [5.3] and [5.4] together result in the solution for the pole placement with the determined gains and the stability of the Quadrotor. See the MatLab code given in Appendix B.

In order to be a stable system the rule of Lyapunov states[23], if the linearized system is strictly stable (i.e., if all eigenvalues of A are strictly in the left-half complex plane), then the equilibrium point is asymptotically stable (for the actual non-linear system). If it shown that the eigenvalues are present in the left half plane then the gain matrix in combination with the linearized model can be put in to the non-linear dynamical model by using equation[5.5]

$$\mathrm{U} = \mathrm{U}^{*} - \mathrm{k}(\mathrm{x} - \mathrm{x}^{*}). \qquad [5.5]$$

6 Conclusions and recommendations

Unmanned aerial vehicles are ‘small’ flying machines without the need of any pilot and is controlled from the ground with the use of a controller with WIFI-connection and is mostly called, drone. In the near future, drones will have to cooperate to accomplish certain tasks together. This asks for synchronization of drones. Within this task the mathematical Program MatLab is used.

This report gives the non-linear dynamical model of the Quadrotor and a linearized model for the Quadrotor for maintaining hovering. The model is set in such a way that it is ready to develop the controller for the non-linear dynamical model. There are no simulation results, therefore the result of the similarity between the non-linear-dynamical model derived in this report and the models used in papers with similar research are reliable to proceed in future work.

The future work of this project is to determine the gains and the poles of the controller and check if the system is stable whether or not. After the controller is developed, the implementation of the controller into the dynamical model of the Quadrotor can be done and check with simulations if the Quadrotor is not drifting away from its given position. If the simulation results proved to be correct, the implementation of the Dynamical model into the Software development kit can be done. After the implementation, the Quadrotor can fly with the given hovering maneuver.

Bibliography

[1] Brandão A.S., Filho M.S., Carelli R., High-level underactuated non-linear Control for rotorcraft machines, Mechatronics (ICM), Feb. 27 2013-March 1 2013 IEEE international conference

[2] McKerrow P.J. Modelling the Draganflyer four helicopter 2004, University of Wollongong

[3] Mueller M.W., D`Andrea R., Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers., Robotics & Automation (ICRA), May 31-June 7, 2014 Hong Kong, China IEEE international conference

[4] MIT OpenCourseWare., 2.017J Design of electromechanical robotic systems., Kinematics of moving frames., Fall 2009., http://ocw.mit.edu

[5] Rich M., Model development, system identification, and control of a quadrotor helicopter., Master thesis Iowa State University., 2012

[6] Xiong J., Zheng E., Position and attitude tracking control for a quadrotor UAV., Paper Elsevier., College of mechanical and electrical engineering, China Jiliang university, Hangzhou 310018, PR China., 01-16-2014.

[7] Brizard A.J., Motion in a non-inertial frame-Dartmouth college., Saint michael's college., July-7, 2007

[8] Meriam J.L., Kraige L.G., Engineering mechanics dynamics 7th edition., section 5-7., 2013 John Wiley&Sons Singapore Pte. Ltd

[9] Buchholz T.T., Greatarsson D., Construction of a four rotor helicopter control system., Master thesis Technical university Denmark., Feb-2009

[10] Beard&McLain., Small unmanned aircraft., Chapter 3, Kinematics and dynamics., Princeton university press 2012

[11] Pounds P, & Mahony P.P., Hynes P, & Roberts., Australian National University Design of a four-rotor aerial robot., Proc. 2002 Australasian conference on robotics and automation., Auckland, 27-29 November 2002

[12] Carrillo G., López D., Lozano A.E., Pégard R., Quad rotorcraft control chapter 2., Vision-based hovering and navigation., Springer., 2013.

[13] Naidoo Y., Stopforth R., Bright G., Quad-rotor unmanned aerial vehicle helicopter modelling & control., Intech open access publisher., 08-25-2011

[14] Castillo P., Dzul A., Lozano R., Real-time stabilization and tracking of a four rotor mini-rotorcraft., Proceedings of 2004 IEEE/RSJ international conference on intelligent robots and systems., September 28 – October 2, 2004, Sendai, Japan

[15] Bramwell A. R. S., Done G., Balmford D., Bramwell's helicopter dynamics., A. R. S. Bramwell, George Done and David Balmford 2001

[16] Raffo G.V., Ortega M.G., Rubio F.R., An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter, Automatica 46(2010)29-39, 29-09-2009

[17] Zemalache K.M., Maaref H., Controlling a drone: Comparison between a based model method and a fuzzy inference system, Appleid soft computing, IBISC Laboratory, CNRS-FRE 3190, 08-10-2008

[18] Egerstedt M., Course for the control of mobile robots, George tech university, School of electrical and computing engineering, Lecture 3.3 Linearization, 02-04-2013

[19] Lygeros J., Ramponi F. A., Lecture notes on linear system theory, Automatic control laboratory ETH Zurich, Department of information engineering university of Brescia, 01-03-2015

[20] Balas C., MSc thesis, Modelling and linear control of a Quadrotor, Cranfield university, School of engineering, September 2007.

[21] Cao Y., ENGR487 Lecture10 Controllability, Observability and Pole Placement, University of British Colombia, 02-07-2014.

[22] Steinbuch M., Advanced motion control, Technical university Eindhoven, 2015

[23] Slotine J.J.E, Li W., Appleid Nonlinear control, Section 3.3: Linearization and local stability, Englewood Cliffs New Jersey 07632, Prentice Hall

Appendix A

the derivation of moving/rotating frames in a 3D-plane

In chapter 3 of this report the derivation of the mathematical model is discussed of the AR Drone 2.0 what is called a Quadrotor drone. In order to understand the derivation of the model, in this appendix the derivation is given for deriving the motion of a mass particle(P) that is shown in Figure A1 [7,8]. In Figure A1 R is the position of the origin of the rotating frame according to the origin of the fixed frame, and r' and r are the position of the mass particle(p) according to the fixed frame and rotating frame respectively. Omega(ω) denotes the rotation of the system.

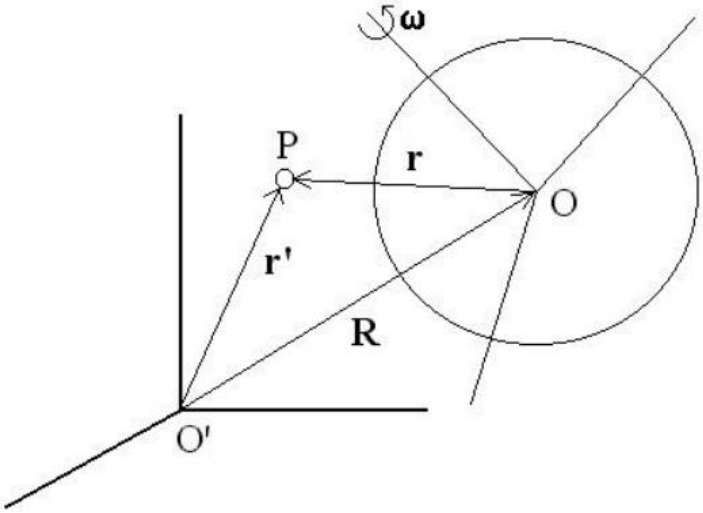

Figure A1 Motion of a particle P in a fixed and rotated frame that are related to each other

The distance from the origin to the mass particle(p) in vector notation is given in equation[A.1]

$$r' = R + r \qquad \text{[A.1]}$$

In order to represent the position of the mass particle from the fixed frame into the rotating frame is the time derivative of the vector r' in the fixed frame taken what is equal to the time derivative of the vector r in the rotating frame and the cross-product of the rotation of the system with vector r, see equation [A.2]

$$(\frac{dr'}{dt})_f = (\frac{dr}{dt})_r + \omega \times r \qquad \text{[A.2]}$$

The mass particle is moving from its initial position and has therefore a rate of change with a certain velocity according to the fixed frame. Due to the rate of change of the mass particle, there is a derivation of the fixed- and rotating frame given in the following equation [A.3]

$$v_f = (\frac{dr'}{dt})_f \quad \text{and} \quad v_r = (\frac{dr}{dt})_r \qquad \text{[A.3]}$$

Using equation[A.2] and taking the time derivative of equation[A.1] gives equation [A.4]

$$v_f = (\frac{dR}{dt})_f + (\frac{dr}{dt})_f = V + v_r + \omega \times r \qquad \text{[A.4]}$$

With V as the translational velocity of the rotating frame origin $(\frac{dR}{dt})_f$. Now the velocity of the mass particle is known, the same result can be achieved from the derivation of the velocity vector in order to obtain the acceleration of the mass particle. This is given in equation[A.5]

$$a_f = (\frac{dv_f}{dt})_f \quad \text{and} \quad a_r = (\frac{dv_r}{dt})_r \qquad \text{[A.5]}$$

Using equation[A.4] and [A.5] results in the acceleration of mass particle from the fixed frame origin

$$a_f = (\frac{dV}{dt})_f + (\frac{dv_r}{dt})_f + (\frac{d\omega}{dt})_f \times r + \omega \times (\frac{dr}{dt})_f \qquad \text{[A.6]}$$

$$a_f = A + a_r + \dot{\omega} \times r + \omega \times (\omega \times r) + 2\omega \times v_r + a_r$$

With A the translational acceleration of the rotating frame origin $(\frac{dV}{dt})_f$, and $2\omega \times v_r$ is the Coriolis term of the equation. It represents the difference between the acceleration of the path the system describes and point p [7,8].

Appendix B

This appendix gives the MatLab script that is used to determine the linearized and controllability of the Quadrotor drone. This MatLab script is used for the chapters 4 and 5 of this report.

MatLab script for the linearized model

```
clc,clear, close all

syms x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_10 x_11 x_12 m g I_xx I_yy I_zz
A_x A_y A_z u_1 u_2 u_3 u_4 k_f k_t l omega_1 omega_2 omega_3 omega_4 k1 k2
k3 k4 k5 k6 k7 k8 k9 k10 k11 k12 lambda

f1=x_4;
f2=x_5;
f3=x_6;
f4=(1/m)*(cos(x_9)*sin(x_8)*cos(x_7)+sin(x_9)*sin(x_7))*u_1+((A_x/m)*sin(x_
8));
f5=(1/m)*(sin(x_9)*sin(x_8)*cos(x_7)-
cos(x_9)*sin(x_7))*u_1+((A_y/m)*sin(x_7)*cos(x_8));
f6=-g+(1/m)*(cos(x_8)*cos(x_7))*u_1+((A_z/m)*cos(x_7)*cos(x_8));
f7=x_10;
f8=x_11;
f9=x_12;
f10=((u_2       +       x_11*(I_xx*x_12*cos(x_8)       -       (I_yy       -
I_zz)*(x_12*cos(x_8)*sin(x_7)^2       +       x_11*cos(x_7)*sin(x_7))       +
x_12*cos(x_7)^2*cos(x_8)*(I_yy             -             I_zz))             +
x_12^2*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy                                    -
I_zz))*(I_yy*I_zz*cos(x_7)^4*cos(x_8)^2 + I_xx*I_yy*cos(x_7)^2*sin(x_8)^2 +
I_yy*I_zz*cos(x_8)^2*sin(x_7)^4    +    I_xx*I_zz*sin(x_7)^2*sin(x_8)^2    +
2*I_yy*I_zz*cos(x_7)^2*cos(x_8)^2*sin(x_7)^2))/(I_xx*I_yy*I_zz*cos(x_7)^4*c
os(x_8)^2          +          I_xx*I_yy*I_zz*cos(x_8)^2*sin(x_7)^4          +
2*I_xx*I_yy*I_zz*cos(x_7)^2*cos(x_8)^2*sin(x_7)^2)                           +
(sin(x_8)*(I_yy*cos(x_7)^2         +         I_zz*sin(x_7)^2)*(u_4         +
x_10*(I_xx*x_11*cos(x_8) - x_12*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy - I_zz))
-                    x_12*(I_xx*x_11*cos(x_8)*sin(x_8)                       -
I_zz*x_11*cos(x_7)^2*cos(x_8)*sin(x_8)                                       +
x_10*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy          -          I_zz)            -
I_yy*x_11*cos(x_8)*sin(x_7)^2*sin(x_8))        +        x_11*((I_yy        -
I_zz)*(x_10*cos(x_8)*sin(x_7)^2    +    x_11*cos(x_7)*sin(x_8)*sin(x_7))    -
x_10*cos(x_7)^2*cos(x_8)*(I_yy  -  I_zz)  -  I_xx*x_12*cos(x_8)*sin(x_8)  +
I_zz*x_12*cos(x_7)^2*cos(x_8)*sin(x_8)                                       +
I_yy*x_12*cos(x_8)*sin(x_7)^2*sin(x_8))))/(I_yy*I_zz*cos(x_7)^4*cos(x_8)^2
+                     I_yy*I_zz*cos(x_8)^2*sin(x_7)^4                        +
2*I_yy*I_zz*cos(x_7)^2*cos(x_8)^2*sin(x_7)^2)                                -
(cos(x_7)*sin(x_7)*sin(x_8)*(I_yy          -          I_zz)*(u_3          -
x_12*(I_zz*x_12*cos(x_7)^2*cos(x_8)*sin(x_8)  -  I_xx*x_12*cos(x_8)*sin(x_8)
+ I_yy*x_12*cos(x_8)*sin(x_7)^2*sin(x_8)) - x_10*(I_xx*x_12*cos(x_8) - (I_yy
-     I_zz)*(x_12*cos(x_8)*sin(x_7)^2     +     x_11*cos(x_7)*sin(x_7))     +
x_12*cos(x_7)^2*cos(x_8)*(I_yy - I_zz)) + x_10*x_11*cos(x_7)*sin(x_7)*(I_yy
- I_zz)))/(I_yy*I_zz*cos(x_8)*sin(x_7)^4 + I_yy*I_zz*cos(x_7)^4*cos(x_8) +
2*I_yy*I_zz*cos(x_7)^2*cos(x_8)*sin(x_7)^2);

f11=((I_zz*cos(x_7)^2          +          I_yy*sin(x_7)^2)*(u_3          -
x_12*(I_zz*x_12*cos(x_7)^2*cos(x_8)*sin(x_8)  -  I_xx*x_12*cos(x_8)*sin(x_8)
+ I_yy*x_12*cos(x_8)*sin(x_7)^2*sin(x_8)) - x_10*(I_xx*x_12*cos(x_8) - (I_yy
-     I_zz)*(x_12*cos(x_8)*sin(x_7)^2     +     x_11*cos(x_7)*sin(x_7))     +
x_12*cos(x_7)^2*cos(x_8)*(I_yy - I_zz)) + x_10*x_11*cos(x_7)*sin(x_7)*(I_yy
```

```
-      I_zz)))/(I_yy*I_zz*cos(x_7)^4      +      I_yy*I_zz*sin(x_7)^4      +
2*I_yy*I_zz*cos(x_7)^2*sin(x_7)^2) - (cos(x_7)*sin(x_7)*(I_yy - I_zz)*(u_4
+  x_10*(I_xx*x_11*cos(x_8)  -  x_12*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy  -
I_zz))           -           x_12*(I_xx*x_11*cos(x_8)*sin(x_8)           -
I_zz*x_11*cos(x_7)^2*cos(x_8)*sin(x_8)                                    +
x_10*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy          -          I_zz)          -
I_yy*x_11*cos(x_8)*sin(x_7)^2*sin(x_8))        +        x_11*((I_yy       -
I_zz)*(x_10*cos(x_8)*sin(x_7)^2   +   x_11*cos(x_7)*sin(x_8)*sin(x_7))    -
x_10*cos(x_7)^2*cos(x_8)*(I_yy  -  I_zz)  -  I_xx*x_12*cos(x_8)*sin(x_8)  +
I_zz*x_12*cos(x_7)^2*cos(x_8)*sin(x_8)                                    +
I_yy*x_12*cos(x_8)*sin(x_7)^2*sin(x_8))))/(I_yy*I_zz*cos(x_8)*sin(x_7)^4  +
I_yy*I_zz*cos(x_7)^4*cos(x_8) + 2*I_yy*I_zz*cos(x_7)^2*cos(x_8)*sin(x_7)^2)
- (cos(x_7)*sin(x_7)*sin(x_8)*(I_yy - I_zz)*(u_2 + x_11*(I_xx*x_12*cos(x_8)
-  (I_yy  -  I_zz)*(x_12*cos(x_8)*sin(x_7)^2  +  x_11*cos(x_7)*sin(x_7))  +
x_12*cos(x_7)^2*cos(x_8)*(I_yy           -           I_zz))           +
x_12^2*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy                                 -
I_zz)))/(I_yy*I_zz*cos(x_8)*sin(x_7)^4  +  I_yy*I_zz*cos(x_7)^4*cos(x_8)  +
2*I_yy*I_zz*cos(x_7)^2*cos(x_8)*sin(x_7)^2);

f12=((I_yy*cos(x_7)^2 + I_zz*sin(x_7)^2)*(u_4 + x_10*(I_xx*x_11*cos(x_8) -
x_12*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy          -          I_zz))         -
x_12*(I_xx*x_11*cos(x_8)*sin(x_8)  -  I_zz*x_11*cos(x_7)^2*cos(x_8)*sin(x_8)
+       x_10*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy       -       I_zz)       -
I_yy*x_11*cos(x_8)*sin(x_7)^2*sin(x_8))        +        x_11*((I_yy       -
I_zz)*(x_10*cos(x_8)*sin(x_7)^2   +   x_11*cos(x_7)*sin(x_8)*sin(x_7))    -
x_10*cos(x_7)^2*cos(x_8)*(I_yy  -  I_zz)  -  I_xx*x_12*cos(x_8)*sin(x_8)  +
I_zz*x_12*cos(x_7)^2*cos(x_8)*sin(x_8)                                    +
I_yy*x_12*cos(x_8)*sin(x_7)^2*sin(x_8))))/(I_yy*I_zz*cos(x_7)^4*cos(x_8)^2
+                   I_yy*I_zz*cos(x_8)^2*sin(x_7)^4                       +
2*I_yy*I_zz*cos(x_7)^2*cos(x_8)^2*sin(x_7)^2)  +  (sin(x_8)*(I_yy*cos(x_7)^2
+    I_zz*sin(x_7)^2)*(u_2    +    x_11*(I_xx*x_12*cos(x_8)    -    (I_yy    -
I_zz)*(x_12*cos(x_8)*sin(x_7)^2      +      x_11*cos(x_7)*sin(x_7))      +
x_12*cos(x_7)^2*cos(x_8)*(I_yy           -           I_zz))           +
x_12^2*cos(x_7)*cos(x_8)^2*sin(x_7)*(I_yy                                 -
I_zz)))/(I_yy*I_zz*cos(x_7)^4*cos(x_8)^2  +  I_yy*I_zz*cos(x_8)^2*sin(x_7)^4
+ 2*I_yy*I_zz*cos(x_7)^2*cos(x_8)^2*sin(x_7)^2) - (cos(x_7)*sin(x_7)*(I_yy -
I_zz)*(u_3      -      x_12*(I_zz*x_12*cos(x_7)^2*cos(x_8)*sin(x_8)       -
I_xx*x_12*cos(x_8)*sin(x_8)   +   I_yy*x_12*cos(x_8)*sin(x_7)^2*sin(x_8))   -
x_10*(I_xx*x_12*cos(x_8)   -   (I_yy   -   I_zz)*(x_12*cos(x_8)*sin(x_7)^2   +
x_11*cos(x_7)*sin(x_7))   +   x_12*cos(x_7)^2*cos(x_8)*(I_yy   -   I_zz))   +
x_10*x_11*cos(x_7)*sin(x_7)*(I_yy  -  I_zz)))/(I_yy*I_zz*cos(x_8)*sin(x_7)^4
+                    I_yy*I_zz*cos(x_7)^4*cos(x_8)                        +
2*I_yy*I_zz*cos(x_7)^2*cos(x_8)*sin(x_7)^2);

u_1_fixed_point=m*g-
(k_f*((omega_1)^2+(omega_2)^2+(omega_3)^2+(omega_4)^2));
u_2_fixed_point=k_f*((omega_3)^2-(omega_1)^2)*l;
u_3_fixed_point=k_f*((omega_4)^2-(omega_2)^2)*l;
u_4_fixed_point=k_f*k_t*((omega_1)^2-(omega_2)^2+(omega_3)^2-(omega_4)^2);

u=solve(u_1_fixed_point==0,     u_2_fixed_point==0,     u_3_fixed_point==0,
u_4_fixed_point==0, omega_1, omega_2, omega_3, omega_4);

u=[u.omega_1 u.omega_2 u.omega_3 u.omega_4];

A=jacobian([f1 ;f2 ;f3 ;f4 ;f5 ;f6 ;f7 ;f8 ;f9 ;f10 ;f11 ;f12],[x_1 x_2 x_3
x_4 x_5 x_6 x_7 x_8 x_9 x_10 x_11 x_12]);
```

```
B=jacobian([f1 ;f2 ;f3 ;f4 ;f5 ;f6 ;f7 ;f8 ;f9 ;f10 ;f11 ;f12],[u_1 u_2 u_3
u_4]);

A_x=1;
A_y=1;
A_z=1;
m=0.79;
g=9.81;
I_xx=0.004;
I_yy=0.004;
I_zz=0.0084;

u_1=m*g-A_z;
u_2=0;
u_3=0;
u_4=0;
x_1=0;
x_2=0;
x_3=0;
x_4=0;
x_5=0;
x_6=0;
x_7=0;
x_8=0;
x_9=0;
x_10=0;
x_11=0;
x_12=0;

A_x=1;
A_y=1;
A_z=1;
m=0.79;
g=9.81;
I_xx=0.004;
I_yy=0.004;
I_zz=0.0084;

A=eval(A)
B=eval(B)

E=rank(ctrb(A,B))

% load('Nprime');
% load('K');

Nprime=rand(4,12)*0.01
K=rand(4,1)*0.01

A_new=A+(B*Nprime)
B_new=B*K

Z=ctrb(A_new,B_new)
G=rank(Z)

K_gain=[k1 k2 k3 k4 k5 k6 k7 k8 k9 k10 k11 k12]

Lambda_matrix=lambda*eye(12,12)
System_matrix=A_new-(B_new*K_gain)
```

```
Q=det(Lambda_matrix-System_matrix)
P=expand((lambda+1)^12)

% maxiterations = 100000;
% maxvalue = 1000;
% ii = 0;
% while true
%     ii = ii + 1;
%     disp(['iteration ' num2str(ii)]);
%     K = maxvalue*rand(4,1) - (maxvalue/2)*ones(4,1);
%     N_ = maxvalue*rand(4,12) - (maxvalue/2)*ones(4,12);
%     E = A + B*N_;
%     G = B*K;
%     test1 = rank(ctrb(E,G));
%     ranks(ii) = test1;
%     if test1 == 12
%         disp('solution found :');
%         E
%         G
%         break;
%     elseif  ii >= maxiterations
%         disp('loop terminated, no solution found')
%         disp(['maximum rank found : ' num2str(max(ranks))]);
%         break;
%     end
% end
```

Appendix C

Software development kit of the AR. Drone 2.0

In this appendix the software development kit (SDK) for implementing the derived model as discussed in chapter 3 is discussed. In the first paragraph the description of the SDK is discussed and gives the meaning what can be done with the SDK. In the second paragraph the implementation of the dynamical of the Quadrotor system for hovering is discussed.

B.1 Description of the software development kit

The SDK is the environment to develop your own embedded code to control the Quadrotor drone. The Quadrotor drone is controlled by use of a controller with WIFI-connection and therefore a router on the ground. The Quadrotor system can measure around three degrees of freedom by use of a magnetometer on every axis and a pressure sensor to measure altitude on every height. In order to program your own embedded code you have to choose on what platform you want to program. The Quadrotor supports Linux, Apple, Android, and Windows. An important note for programing your own code is that the SDK not support rewriting your own embedded software and no direct access to the Quadrotor hardware (sensors/engines) is allowed.

The input of the Quadrotor that is given, should be in the following domain ϵ [-1,0 ; 1,0] and has to implemented as floating point numbers. The input that can be controlled by the user are the following

- The vertical speed
- Phi-angle
- Theta-angle
- Psi-angle

In order to allow the user to choose between smooth or dynamic moves, the arguments of this features are not directly the control parameters values, but a percentage of the maximum corresponding values as set in the drone parameters. In the follow table B.1 are the most important configuration parameters listed that can be configured before putting the Quadrotor into practice.

Table 4 Configuration parameters of the Quadrotor drone

Max_euler_angle	Outdor	Indoor/outdoor_control_$v_z$max
Max_altitude	Flight without shell	Indoor/outdoor_control_yaw
Min_altitude	Autonomous flight	Flying_mode
$V_z$max [mm/s]	Manual_trim	Hovering_range
Control_yaw [rad/s]	Indoor/outdoor_euler_angle_max	Flight_anim

On the following page is the Drone altitude viewer given that shows the Quadrotor during its operation in the sky. This viewer gives you the information for the rotational angles around its principle axes and the direction of heading. In this viewer you can give the Quadrotor reference values to allow the

Quadrotor to rotate to a given value. Next to insert the reference values is the possibility there to insert values for the gains to control the rate, cocarde, altitude and fixed point respectively.

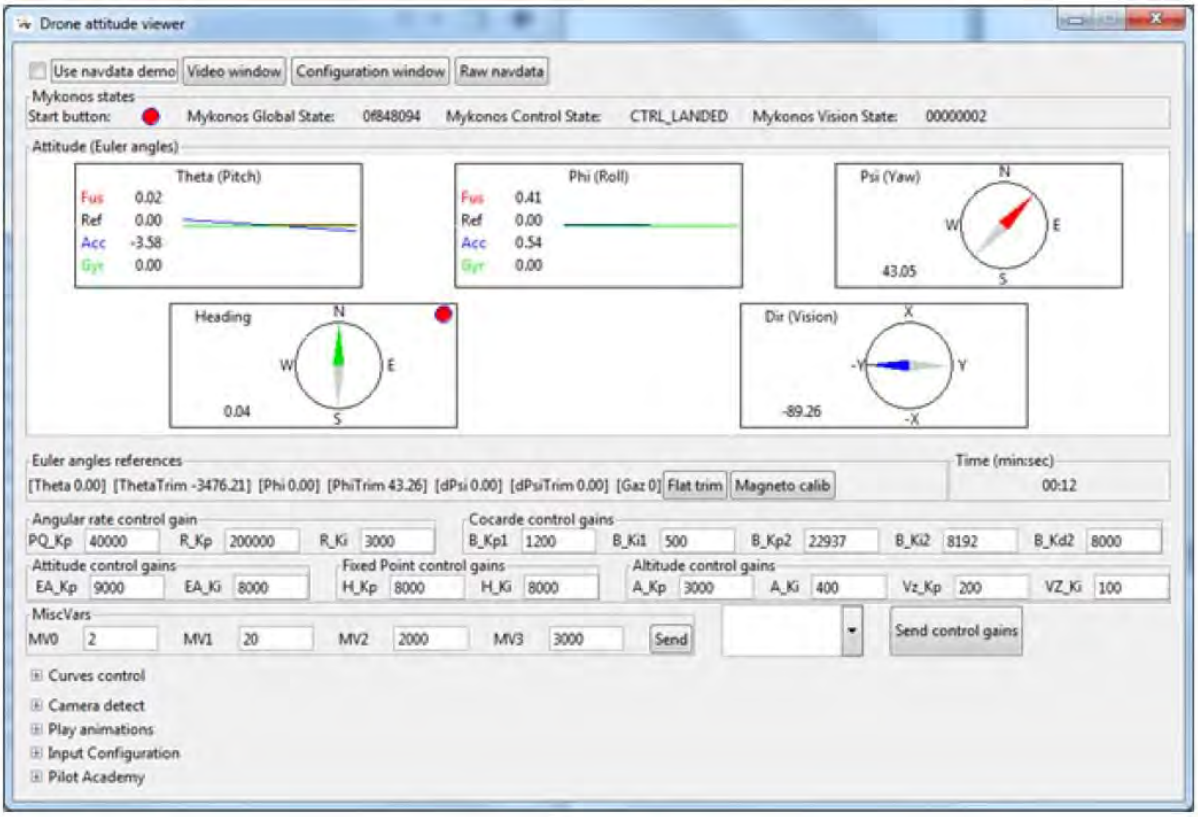

Figure B1 Drone altitude viewer for watching/changing parameters during flight modes