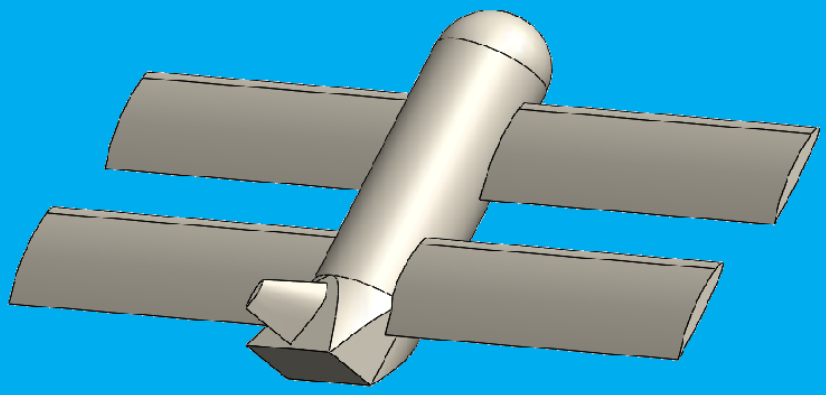
Hybrid UAV Attitude Control using INDI and Dynamic Tilt-Twist

Thesis report

L.F.A. Dellemann - 4616057



TU Delft

Hybrid UAV Attitude Control using INDI and Dynamic Tilt-Twist

Thesis report

by

L.F.A. Dellemann - 4616057

This thesis was written, for the MSc degree in Aerospace Engineering at the TU Delft. This thesis was conducted within the the Control & Simulation department from October 2019 until December 2020.

Student name:	Lars Dellemann
Student number:	4616057
Date:	03-12-2020

Supervisor:	Ir. C. de Wagter
Thesis committee:	Prof. dr. G.C.H.F. de Croon
	Dr. J. Guo
	Dr. E.J.J. Smeur

TU Delft

Preface

This thesis was written, for the MSc degree in Aerospace Engineering at the TU Delft. It was conducted within the the Control & Simulation department from October 2019 until November 2020.

The sources for this thesis thoughts and concepts are based on the performed literature study, meetings with the daily supervisors, earlier research and meetings with employees within the faculty.

This thesis is written for engineers, aerospace students and researchers that have a sustained understanding of control theory principles and/or have an interest for getting a better understanding of tailsitter control.

Acknowledgements

I want to express my gratitude to Ir. C. de Wagter, my daily supervisor. His suggestions and support helped me conduct this research.

In addition, I want to thank Dr. E.J.J. Smeur for helping me with the paparazzi software and understanding the controller.

Furthermore I want to express a word of gratitude to the Nederdrone team which shared their knowledge about the tailsitter and gave me the possibility to work with the Nederdrone.

Contents

I

Preliminary Thesis Report

1 Introduction

1.1. General

The market for applications of Unmanned Aerial Vehicles (UAV) is increasing [29]. The first UAV was made in 1849 (a hot air balloon) [3]. Until 1984, the use of this type of vehicle was small. From 1984 onward, the military started investing in remotely controlled UAVs [11] and first used them during the 1990's. Due to the development of small processors the UAV became available for the public since the last couple of decades [33]. Many companies are currently developing UAVs for various purposes [2] [25]. The reasons to use these type of aircraft are their low cost, high maneuverability and ease of use. A quadcopter is a type of UAV which has good vertical take-off and landing (VTOL) and maneuverability performance. One of the major drawbacks of quadcopters is their flight endurance. The fixed wing UAV is comparable to the conventional shape of an aircraft. A fixed wing is able to cover larger distances than quadcopters, but it has poor hover performance. A solution to achieve both the benefits of a quadcopter (high maneuverability) and a fixed wing plane (able to cover long distances) is to use a hybrid UAV (Figure 1.3). This type of aircraft is able to take off vertically. A tailsitter is a special type of a hybrid UAV. A tailsitter uses the same motors for hover and forward flight. Once it is in the air, it is able to pitch down and fly like a fixed wing. One of the challenges with this type of UAV is during the landing phase, especially during tough conditions (hard and unpredictable wind), which regularly occurs on vessels [6]. The vessel used during this project is the Guardian, see Figure 1.1. The challenge lies in the nonlinear behaviour of the UAV. During hover some actuators become less efficient and saturation occurs. The controller can demand more than the actuator can handle. Therefore, it needs to decide which manoeuvre has the highest priority: pitch, roll or yaw. Furthermore, the actuator effectiveness changes orders of magnitude depending on the amount of airflow that passes over them. For example, when hovering, the airflow over the wings is almost negligible which reduces the effectiveness of the elevons.

Figure 1.1: Vessel example, the Guardian

1.2. Types of tailsitters

There are multiple types of UAVs, within this research the tailsitter is used. In order to get a better understanding of how these type of UAVs operate, the different variants will be presented. This also includes looking into the different components and their functions. In the figures on the next page examples of tailsitters are shown. The tailsitter in figure 1.2 is one of the first tailsitters[9], which was an operational tailsitter that

was not developed further due to required pilot experiences. The aircraft was hard to control as there were no advanced controllers to help the pilot. With two props, yaw control could be possible by varying torque. It had two elevons and a rudder that were used to control the attitude. Another disadvantage of this model is the handling capability in hover. During hover it must be stabilised by the aerodynamic control surfaces, but the airflow over the control surfaces is very small which makes them less effective. The Cyclone (Figure 1.3) has only two aerodynamic control surfaces (the elevons). These are used to control the pitch and the yaw rotations. The props are used to generate thrust and roll (in hover), rolling with the props has a side effect in yaw due to the torque difference in both propellers. It needs to be compensated by the elevons. The pitch of the props can not be changed which leads to low performance, as efficient hovering requires other pitch angles than for cruise flight [32]. The UAVs from Figures 1.4 and 1.5 were made by the TU Delft. The Delftacopter has seven actuators [6]. The rotor of the Delftacopter is very large compared to the wings (fuselage), which means that the rotor contributes more to the rotational inertia than is typical for quadcopters and less for helicopters. This interferes significantly with the dynamics of the Delftacopter and complicates the control. The NederDrone has 20 actuators, 12 of them generate the lift in hover. During this project the NederDrone is the used tailsitter.

Figure 1.2: The Convair Pogo

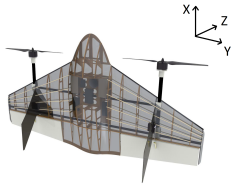


Figure 1.3: The Cyclone

Figure 1.4: Delftacopter

Figure 1.5: NederDrone

1.3. NederDrone

The tailsitter used for this project is the NederDrone, see Figure 1.5. The NederDrone is being developed in cooperation with the Dutch defense department. Eventually the tailsitter must be able to perform long endurance recon flights autonomously. This will be accomplished by using a hydrogen tank, with a target flight time of four hours. The NederDrone has two wings which carry twelve engines and hold eight elevons. The twelve engines are used in hover, only four are used during forward flight. The wingspan is approximately two meters. At the beginning of this thesis it was controlled by a PID controller and uses Paparazzi software.

1.4. Problem definition

For the autonomous landing of the tailsitter several problems need to be considered and solved, which are:

1. Wind conditions affected by the ship
2. Ground effect (clarified in Appendix A)
3. How to keep track of the vessel's position
4. Actuator saturation during hover
5. Nonlinear flight dynamics

This thesis focuses on problems 1, 4 and 5. The objective of this study is:

Improve the landing capabilities of the NederDrone when landing on a vessel with high distortions (wind) by designing a suitable controller and demonstrating its performance experimentally.

The project can be divided into three pillars: the literature study in the already existing solutions to the problems, designing a suitable controller and test/validate the controller. Part I includes the literature study. In chapter 2 the type of controllers are explained. The control allocation methods are described in chapter 3.

After the literature study the paper is presented, part II. It presents the dynamic tilt-twist method in combination with an INDI controller, as well as the performed tests. In part III general conclusions are drawn and recommendations are given. The appendix starts after part III. The ground effect acting on the Nederdrone is described in appendix A. In Appendix B two type of controllers are compared, IBS and INDI. The quaternion feedback error method is compared to the tilt-twist method in appendix C. The data analyse done for the flight test for the paper is shown in appendix D.

2

Controllers

To determine the right controller for this project the different types need to be investigated and compared. There are a lot of different controllers available, each with their own advantages and disadvantages. Some were designed specially for nonlinear systems, which makes them more suitable for this project. This literature study will provide a broad view of the available controllers. First a close look into the widely used PID controller is given. As this controller is not well suited for nonlinear systems, controllers with better characteristics will be examined. These controllers are the model predictive control, incremental backstepping, nonlinear dynamic inversion and its variants. Before looking into the different types of controllers the general outline will be mentioned.

2.1. General controller overview

2.1.1. Inner and Outer Loop

Separating the control in two cascaded loops is generally seen in the UAV industry [16, 20]. The advantages of using this method is the ease of tuning and simplicity of implementation. Furthermore, controlling a 4th order system with a PID is impossible. When using two PID controllers for two 2th order systems it becomes controllable. The two used loops are the inner loop and the outer loop, as shown in Figure 2.1. Within the inner loop the moments are controlled, which are linked to the attitude. The outer loop controls the forces, which are linked to the position.

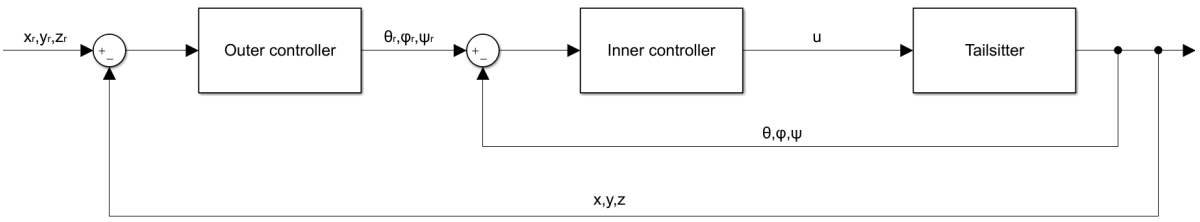


Figure 2.1: General tailsitter control

2.1.2. Resolved tilt-twist angle control

Within the inner loop both the tilt and the twist are controlled. The most important factor is the tilt, as it needs to overcome the gravity. Matsumoto et al. described a solution to prioritise the tilt over the twist, by using resolved tilt-twist (RTT) [23, 37].

The quaternion feedback works when the attitude errors are small but will have problems as the errors become larger. An example: the required attitude is 0, 90, 0 degrees for α, β, γ and the current attitude is 180, 80, 0 degrees. Calculating the quaternion error (0, -0.57, 0, -0.34) will have no initial error around the y-axis. The tilt-twist method is further elaborated and simulated in appendix C.

2.1.3. Resolved Euler-angles

As the resolved tilt twist angle uses more computational power compared to the normal quaternion or Euler feedback an error determination method is proposed by Argely et al [30]. The Resolved Euler Angle (REA) has almost the same performance as the resolved tilt-twist but uses less computational effort. The REA method describes the attitude error as three subsequent rotations. First the heading error is computed by pitching around $\widetilde{\phi}$,$\widetilde{\theta}$ and $\widetilde{\psi}$ or roll, pitch and yaw. The needed formulas for the errors can be seen in equation 2.1, 2.2 and 2.3. The equations make use of the quaternions. Singularity occurs when $\widetilde{\theta}$ equals $\pm 90°$. This can be solved by saturating $\widetilde{\eta}$.

$$\widetilde{\phi} = \tan^{-1}\left(\frac{2(\widetilde{\eta}_0\widetilde{\eta}_1 - \widetilde{\eta}_2\widetilde{\eta}_3)}{2\widetilde{\eta}_0^2 + 2\widetilde{\eta}_3^2 - 1}\right) \tag{2.1}$$

$$\widetilde{\theta} = \sin^{-1}\left(2(\widetilde{\eta}_0\widetilde{\eta}_2 - \widetilde{\eta}_1\widetilde{\eta}_3)\right) \tag{2.2}$$

$$\widetilde{\psi} = \tan^{-1}\left(\frac{2(\widetilde{\eta}_0\widetilde{\eta}_3 - \widetilde{\eta}_1\widetilde{\eta}_2)}{2\widetilde{\eta}_0^2 + 2\widetilde{\eta}_1^2 - 1}\right) \tag{2.3}$$

2.2. Attitude Error

In order to control the tailsitter a feedback error is needed. There are several ways to generate the feedback error. Two coordinates are required to calculate it, namely the inertial frame and the body frame. A way to describe the body frame is: it is fixed to the UAV and the origin lies in the centre of gravity of the body. The x-axis points to the nose of the aircraft, see Figure 2.2. The y-axis points along the right wing and the z-axis to the bottom of the UAV, following the right hand rule. In the inertial frame the x-axis points to the north, the y-axis to the east and the z-axis is pointing down. The origin is at an arbitrary point.

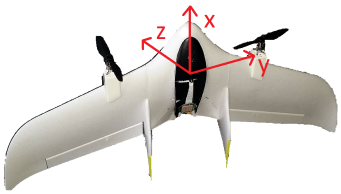


Figure 2.2: Tailsitter with reference frame

2.2.1. Euler Angles

One of the simplest attitude error descriptions is the Euler angles error feedback. Where the attitude error is described by the angles ϕ, θ and ψ [4]. Applying this within a controller can cause gimbal locking. When gimbal lock occurs when two planes are aligned, as in figure 2.3. Moving towards a new position from a the gimbal lock will cause an inefficient manoeuvre.

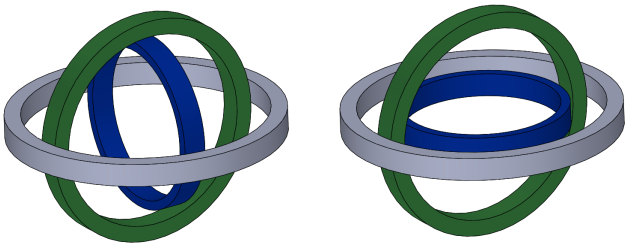


Figure 2.3: Gimbal lock

2.2.2. Quaternion feedback control

Quaternion feedback works with the quaternion system which is widely used in the UAV industry as it is computationally efficient. The gimbal lock issue that can occur when Euler angles are used is also eliminated by using quaternions. The basic formula to use for quaternion feedback can be seen in equation 2.4.

$$\eta_{err} = \eta_{des} \otimes \eta_{cur}^{-1} \tag{2.4}$$

Where η_{err} corresponds to the attitude error around the body axis. $\eta_{desired}$ is the desired attitude and the current attitude is η_{cur}.

2.3. PID

One of the most commonly used controllers is the Proportional Integral Derivative (PID) controller. The general model of the PID controller can be seen in Figure 2.4. The PID controller can easily be applied to linear systems and can be adapted as the controller can be split up into three different parts. The first part is the Proportional, which is always in the controller. It determines how fast the error will be decreased. The Integral can be used to remove the steady state error. The derivative uses the rate of change of the error and is able to influence the damping term. As the controller is used in several areas, for example in UAVs and aircraft a number of papers about the topic have been published [27, 28, 46]. The type of UAV used for this project is highly nonlinear. To be able to control the tailsitter gain scheduling must be done for every flight condition and a gain table must be generated. This will take a large amount of flight tests and it is not guaranteed that all possible flight conditions are included. Therefore this controller will not be used for this project.

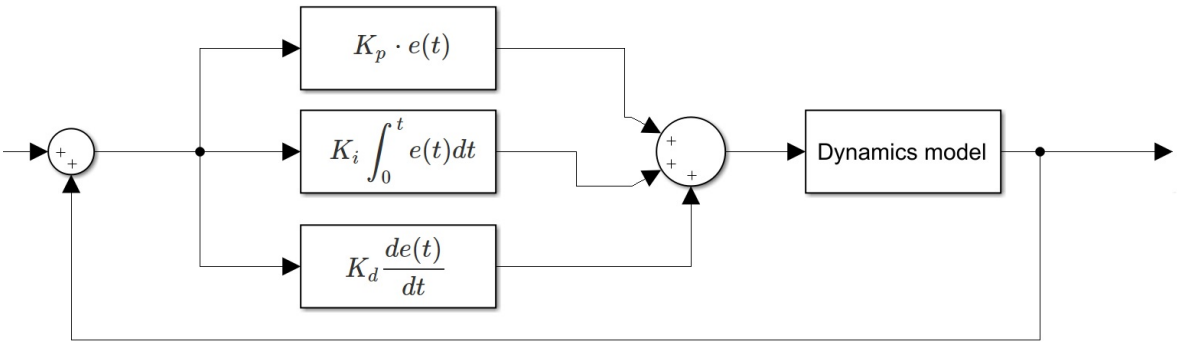


Figure 2.4: Standard PID scheme

2.4. MPC

The Model Predictive Controller (MPC) determines the control behaviour by predicting the upcoming path. A simple example is driving into a turn. Humans adapt to the upcoming turn by reducing speed, a standard PID controller will start to react at the beginning of the turn instead of anticipating it earlier. As it requires an accurate model of the system and adapts to the future it is one of the most promising controllers [7]. The controller can also be used for nonlinear systems by using the Taylor series expansion [10]. H. Khan et al. performed simulation tests on quadcopters with an MPC controller [27]. Nevertheless, there are some major drawbacks to the controller which makes it unsuitable for this project. The first one is the accurate model that is needed. A part of getting an accurate model (especially for a highly nonlinear system) is to gather a large amount of data which goes beyond the scope of the project. The second disadvantage is the computational effort required. As the goal of the tailsitter is to fly as long as possible a light processor is used. To be able to use the MPC controller a more powerful processor is needed which consumes more power. If weight and development time were not limited, the adaptive MPC would be the most suitable controller.

2.5. Sliding mode control

The sliding mode controller is robust and designed for nonlinear systems. E. Zheng et al. performed real flight tests with a UAV [17]. It requires to follow a path. Sliding mode is a trajectory based controller, where it switches between two 'positions' which will results in chattering, which is one of the main disadvantages of the controller. Furthermore, R. Ghosh Dastidar [13] found that the fuel efficiency is low as it requires a lot of control.

2.6. Backstepping

Backstepping control is based on using control Lyapunov functions. These functions give information about the stability and determine the tracking error of nonlinear systems. This tracking error is returned to the controller. An overview of Lyapunov candidates functions can be seen in table 2.1. Several studies are performed regarding this subject. Van Ekeren et al. performed real flight tests on a fixed wing aircraft [44] and Guerreiro et al. simulated an IBS controller for a Boeing 747 [19]. A real-life test with a quadcopter was performed by D. Lee et al. [12].

Table 2.1: Lyapunov candidates

$V(x)$	$\dot{V}(x)$
$\frac{1}{2}x^2$	$x\dot{x}$
$\lvert x\rvert$	$\operatorname{sgn}(x)$
$\sqrt{\lvert x\rvert}$	$\operatorname{sgn}(x)\frac{1}{2}\lvert x\rvert^{-\frac{1}{2}}\dot{x}$
$\ln(x^2+1)$	$2x\dot{x}(x^2+1)^{-1}$
$\ln^2(x+1)$	$x\dot{x}\cdot\log(x+1)(x+1)^{-1}$

In order to determine if the systems are asymptotically stable a scalar function $V(x)$ is used with the following conditions: the function is positive for every x value greater then zero and its time derivative must be negative. To create a backstepping controller the following steps are carried out:

System dynamics
To define the backstepping controller the system dynamics need to be known. These will be defined as:

$$\dot{x}_i = f_i(\boldsymbol{x}_i) + g_i(\boldsymbol{x}_i)x_{i+1}, \quad i = 1,...,n-1 \tag{2.5a}$$

$$\dot{x}_n = f_n(\boldsymbol{x}) + g_n(\boldsymbol{x})u \tag{2.5b}$$

Where the states are defined by $\boldsymbol{x}_i$ and the control inputs are x_{i+1} and u. The controller will follow a reference signal (x_{1r}) by comparing it with x_1. The next steps will be continued with $n = 2$.

Subsystem one
As the defined system has two state variables, x_1 and x_2, the first equation to look at is for the outer loop (x_1), see equation 2.6.

$$\dot{x}_1 = f_1(\boldsymbol{x}_1) + g_1(\overline{\boldsymbol{x}}_1)x_2 \tag{2.6}$$

As can be seen, the equation depends on x_2. This is known as the virtual input as it is not the real control input but a state variable. Next, the errors need to determined. As the system has two state variables there will be two tracking errors z_1 and z_2:

$$z_1 = x_1 - x_{1r} \tag{2.7a}$$

$$z_2 = x_2 - \alpha_1 \tag{2.7b}$$

Where α is the stabilizing function. Now the Lyapunov candidate needs to be selected. The first candidate of table 2.1 is chosen. The Lyapunov candidate and its derivative can be seen in equation 2.8.

$$V_1(z_1) = \frac{1}{2}z_1^2 \tag{2.8a}$$

$$\dot{V}_1(z_1) = z_1\dot{z}_1 = z_1(z_2 + \alpha_1 - \dot{x}_{1r}) \tag{2.8b}$$

The stabilizing control law needs to be chosen. For this example the control law is defined as:

$$\alpha_1 = -c_1 z_1 + \dot{x}_{1r} \tag{2.9}$$

Inserting equation 2.9 into 2.8b gives:

$$\dot{V}_1(z_1) = -c_1 z_1^2 + z_1 z_2 \tag{2.10}$$

Where c_1 is the control gain. The second component of the equation will be excluded by using the upcoming steps. This is done to always get a negative derivative.

Subsystem two
The second subsystem is designed to eliminate the last term of equation 2.10. By using the earlier mentioned Lyapunov control function this will result in:

$$V_2(z) = V_1 + \frac{1}{2}z_2^2 \tag{2.11}$$

The derivative is:

$$\dot{V}_2(z) = \dot{V}_1 + z_2\dot{z}_2 = -c_1 z_1^2 + z_1 z_2 + z_2[\dot{x}_2 - \dot{\alpha}_1] \tag{2.12}$$

Where the $\dot{x}_2$ component depends on u (equations 2.5b and 2.7b). The last step is to determine the formula for u. The formula must be selected so $\dot{V}_2$ is only dependent of $-c_1 z_1^2 - c_2 z_2^2$, so the outcome is always negative (stable). For this method a detailed model description is needed, as can be seen in equation 2.5. An addition to the standard backstepping method is the incremental one which requires a less defined model.

2.7. Incremental Backstepping

Incremental Backstepping (IBS) has more potential to be used for this project. The method increases the robustness of the closed loop system, as it does not rely completely on having a well-defined model [41]. To get to the incremental form, a first order Taylor expansion needs to be carried out. The same model is used as written in the previous part, see equation 2.5 where n is equal to two. When defining the control Lyapunov function for z_2 it becomes:

$$V(z) = \frac{1}{2}z_2^2 \tag{2.13a}$$

$$\dot{V}(z) = z_2\dot{z}_2 = z_2(\dot{x}_2 - v) \tag{2.13b}$$

$$\dot{x}_2 = \dot{x}_{2,0} + A_0\Delta x + G_0\Delta u \tag{2.13c}$$

The assumption is made that the dynamic system increments are small compared to the actuator dynamics ($A_0\Delta x << B_0\Delta u$). Therefore, the component dependent on Δx is eliminated. B_0 represents the control effectiveness matrix. By taking the correct stabilizing control law the system is defined:

$$\Delta u = G_0^{-1}(-cz_2 - \dot{x}_0 - v) \tag{2.14}$$

Previous research showed that this method performs almost identical as INDI [26]. A simulation was performed where an IBS and an INDI controller are designed and compared, see Appendix B. As will be shown later both IBS and INDI are both suited to be used for the tailsitter. Where each controller has its own method. Several tests were done with tailsitters (with less actuators) in normal flight conditions, which will be the bases of this research. In the next two sections the INDI controller will be explained further.

2.8. NDI

A controller which was designed especially for nonlinear systems is the Nonlinear Dynamic Inversion controller (NDI) [35, 47]. NDI has some drawbacks, which makes it hard to implement for a real life tailsitter. The model of the tailsitter must be accurate. In order to get an accurate model a lot of analysis and simulations must be performed which goes beyond the scope of the project. Controllers based on NDI, which are explained later are suitable to be used. Before going into these controllers a look at the fundamentals of these controllers are given, the NDI controller. The NDI controller uses algebraic transformation to convert a nonlinear dynamic system into a linear system. A restriction on the use of a NDI controller is that the system must be feedback linearisable. First the system will be explained as a single input single output (SISO) system. The system is described as equation 2.15.

$$\dot{\mathbf{x}}_\mathbf{c} = f(\mathbf{x}) + G(\mathbf{x})\mathbf{u} \tag{2.15}$$

$$y = h(\mathbf{x}) \tag{2.16}$$

Where $\dot{\mathbf{x}}_\mathbf{c}$ is the output of the system, which consists of the control variables and is a part of the state variables. The state variables are given as $\mathbf{x} = [x_1, x_2, ..., x_n]$ and $\mathbf{u}$ is the control input. To get to the desired input several

steps are required. First the Lie derivative of $h(\mathbf{x})$ with respect to $f(x)$ and $g(x)$ and the gradient of $h(\mathbf{x})$ needs to be determined, as shown in equation 2.17.

$$L_{\mathbf{f}}h(\mathbf{x}) = \nabla h(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}) \qquad L_{\mathbf{g}}h(\mathbf{x}) = \nabla h(\mathbf{x}) \cdot \mathbf{g}(\mathbf{x}) \qquad \nabla h(\mathbf{x}) = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \tag{2.17}$$

By performing these steps the input can be determined, as shown in equation 2.18. Here, r represents the number of differentiations until $L_{\mathbf{g}}L_{\mathbf{f}}^{r-1} \neq 0$.

$$u = \frac{1}{L_{\mathbf{g}}L_{\mathbf{f}}^{r-1}h(\mathbf{x})}(-L_{\mathbf{f}}^{r}h(x) + v) \tag{2.18}$$

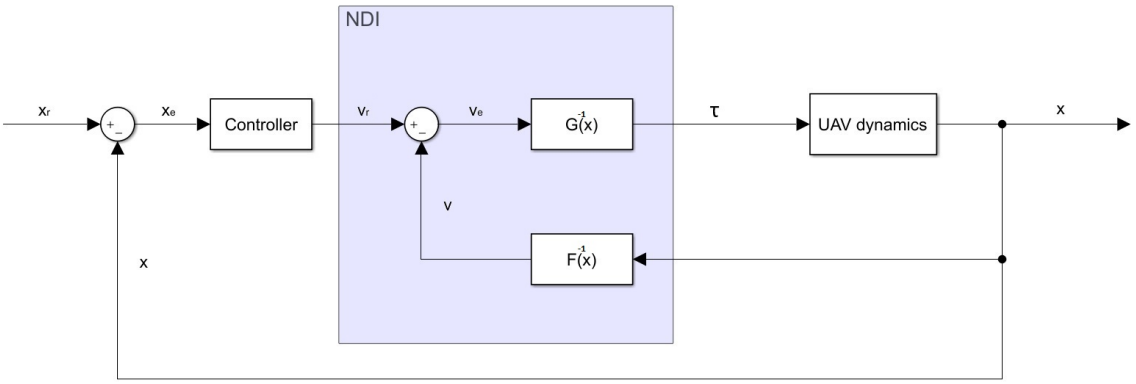


Figure 2.5: Standard NDI scheme

An NDI controller can be controlled in two ways, by a multi-loop or by a single-loop [40]. In the multi-loop case contains a cascaded structure where the outer loop is used to convert the angles command to angular rate commands. The inner loop is used as the controller for the angular rate.

Further research has been performed in order to solve the problems with NDI. Examples of controllers that are based on NDI are INDI and adaptive INDI. The potential of these controllers will be explained.

2.9. INDI

The Incremental Nonlinear Dynamic Inversion (INDI) works on the principle of actuator effectiveness and gives the increment of the controller input [16]. The INDI controller was first presented by S. Sieberling et al. in 2010 [35]. It is based on NDI, where the feedback is coming from the acceleration sensors. Instead of having an accurate model the INDI uses the accelerometer (IMU) to determine the state of the UAV, which is used for the feedback of the INDI. The drawback of INDI is that the noise coming from the measurements propagates and it relies on inverting. Normal control systems also have trouble with noise but for INDI, the situation is worse, as it contains more components which rely on the measurements coming from the sensors. For INDI the G-matrix (control effectiveness matrix) is fixed. Whereas for NDI both the F- and the G-matrix need to be known, for INDI this is only the G matrix needs to be known. This makes the controller less model dependent. During the design of the controller it is not possible to determine the stability as can be done with IBS. This can become a problem for certification [41]. Furthermore it cannot be used directly for non-minimum phase systems, where the systems first respond in the opposite direction to the new input. This cannot be done because the virtual controller directly relies on the output but there are integrators in between [40]. INDI has been tested on both UAVs and general aviation aircraft. E. Smeur et al. [15] performed flight tests inside the wind tunnel, where the UAV (quadcopter) needed to fly in and out of the wind tunnel's wind stream. The UAV used for the test had an INDI controller for the inner (attitude control) and the outer loop (position control). The way it recovered from the sudden change of wind shows that the INDI controller works better then the compared PID controller. E. van Ekeren et al. [40] shows that the INDI controller can be used for fixed wing aircraft. The INDI controller was used for the angular rate and the attitude control. For this research, real flight tests were also performed. R. van 't Veld showed that the controller can also be used in general aviation

aircraft (Cessna Citation) [42]. Unfortunately there were no real flight tests performed, nevertheless some interesting results of the research can be used. When setting the sampling time of a discrete INDI smaller then 0.02s, the stability margins become large.

If this controller is used for this project the G-matrix needs to be determined, which can be done by performing flight tests where several flight conditions are checked. The general system of INDI can be seen in equation 2.19.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{2.19}$$

By assuming that the sampling time is very short and that the control effectors are instant the system can be linearised around the current point (subscript 0). By taking the derivative with respect to x the F matrix is generated, and the G matrix is generated by taking the derivative with respect to u, as shown in equation 2.20. So the complete linearistation around the current point can be seen in equation 2.21.

$$\frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} = F(\mathbf{x_0}, \mathbf{u_0}) \qquad \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} = G(\mathbf{x_0}, \mathbf{u_0}) \tag{2.20}$$

$$\dot{\mathbf{x}} \approx \dot{\mathbf{x}}_0 + F(\mathbf{x_0}, \mathbf{u_0})\Delta\mathbf{x} + G(\mathbf{x_0}, \mathbf{u_0})\Delta\mathbf{u} \tag{2.21}$$

The derivation can be simplified by using the time-scale separation (equation 2.22). This assumption reduces the formula to calculate the state derivative [42]. In order to use it for INDI, the incremental input needs to be known, Δu. This can be done by taking the inverse of the control effectiveness matrix G. Furthermore the virtual input is introduced as ν, which will come from the IMU data. This will result in equation 2.24.

$$F(\mathbf{x_0}, \mathbf{u_0})\Delta\mathbf{x} \ll G(\mathbf{x_0}, \mathbf{u_0})\Delta\mathbf{u} \tag{2.22}$$

$$\dot{\mathbf{x}} \approx \dot{\mathbf{x}}_0 + G(\mathbf{x_0}, \mathbf{u_0})\Delta\mathbf{u} \tag{2.23}$$

$$\Delta\mathbf{u} \approx (\mathbf{v} - \dot{\mathbf{x}}_0)G^{-1}(\mathbf{x_0}, \mathbf{u_0}) \tag{2.24}$$

The input from the controller must not be incremental but must be the real new input. This is accomplished by taking the input of the previous time step and adding the incremental input. An example code is written for a pendulum, which can be seen in Appendix B.

2.10. Adaptive INDI

The normal INDI controller has some limitations for the correctness of the control effectiveness matrix as it can change during flight conditions. E. Smeur et al. described a method where the control effectiveness matrix is updated during flight, known as adaptive INDI [8, 16]. When the model does not respond as it should, due to an inaccurate G-matrix, it is adapted. The delay of the model needs to be considered. When an actuator is exited it needs some time to adjust. There are different models presented with different feedback. Previous research has shown that it is possible to use adaptive INDI to control a tailsitter [36]. The research focused mainly on the transition between normal flight and hover flight. The theory was proven to be successful by performing flight tests with a quadcopter. There are some drawbacks when using adaptive INDI, as it can be calibrated wrong which can result in an unstable flight. For example, if the model is calibrated while still on the ground it will generate a G-matrix which does not correspond with the conditions once airborne. Furthermore, the conditions for this project describe unexpected wind gusts which can influence the performance and therefore possibly cause inaccurate estimates of the parameters.

3

Control allocation

To achieve a new state the Nederdrone has several options to chose from. It needs to be considered how control will be allocated to the actuators when there are more actuators available than are necessary for reaching the desired state. For example, to change the pitch angle the propellers or the elevons can be used. The allocation of the actuators is mentioned within this part of the report. Also, the saturation is mentioned, which can be dealt with by control allocation. Several methods can be used to solve these problems, including the weighted least squares and the sequential least squares methods, which are optimization techniques [14].

The control allocation problem is shown graphically in Figure 3.1. It will translate the virtual input ($v(t)$) coming from the controller into the required input for the UAV. Here, the required input u is bounded by:

$$\underline{u} \leq u \leq \overline{u}$$

Where $\underline{u}$ and $\overline{u}$ are the upper and lower bounds of the required input. The optimization will give the optimum path by setting the correct inputs. The optimum is defined by defining the weight factors, where the goal can be to achieve the new state as fast as possible. When the UAV is not able to achieve the given virtual input the optimization should approach the required attitude as closely as possible.

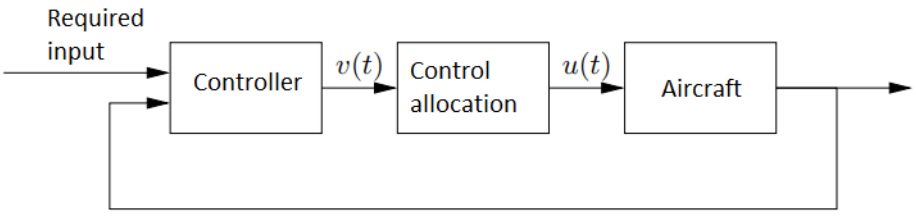


Figure 3.1: Flowchart of the control loop, including the control allocation phase.

3.1. Saturation

Saturation of the actuators is a problem when flying with the NederDrone. During hover saturation of the elevons and motors can occur. During this flight condition the effectiveness of the elevons is small as the airflow over the wings is only coming from the propellers. The saturation is occurring and must be dealt with in a smart manner. As a control allocation method needs to be selected a logical decision is to include the saturation. Other ways of dealing with the saturation is to apply Pseudo Control Hedging [24]. With this method the aircrafts response to the given inputs is analysed. Thereafter, the required input is changed so states with less priority will get a lower input difference, for example the thrust vector to overcome the gravity is more important than yaw control.

3.2. Least squares

There are several types of least squares methods. All of them will try to solve equation 3.1 for u.

$$Bu = v \tag{3.1}$$

Where B is the control effectiveness matrix, u the controller input, v represent the virtual input. Sequential least squares is one of these techniques. It is formulated in equation 3.2.

$$u_{seq} = \arg\min_{u}[||W_a(u-u_p||) : u \in \arg\min_{\underline{u}\leq u\leq \overline{u}} ||W_b(Bu-v)||] \tag{3.2}$$

In this equation u_p represents the preferred control vector and W_a and W_b are weight factors. Where the W_a matrix sets the weight factors for the inputs and W_b for the objective. To find the solution a redistribution scheme is used [43]. Every actuator that exceeds its maximum deflection is saturated and removed from the optimization. The optimization is then repeated with the remaining free variables. This process is repeated until a feasible solution is found or until all the actuators are saturated.

Equation 3.2 can also be reformulated as one optimization known as the weighted least square method (WLS):

$$u_{wls} = \arg\min_{\underline{u}\leq u\leq \overline{u}} ||(W_a(u-u_p)||^2 + \gamma||W_b(Bu-v)||^2 \tag{3.3}$$

The WLS method is known to be faster than the standard least squares method. It minimizes a cost function where multiple actuators are included. The cost function is defined by introducing a parameter γ [34], that represents the relation between the first and second objective[14].

3.3. Active set method

The active set method was introduced by Härkegård [21]. Whereas the least squares methods only gives an approximation of the optimal solution, the active set method gives the optimum solution within a finite number of iterations. The method is efficient when using the weighted least squares, especially when an good estimate of the optimal set is available. The active set method uses several constrains consisting of equality and bounded constraints. All active set methods work in the same manner: every iteration the inequality constrains are treated as equality constraints, while the other constrains are neglected. Then, the Karush-Kuhn-Tucker (KKT) optimality conditions will be checked [34]. These are first derivative tests where the optimality is determined. The optimization finishes when this condition is met, otherwise the active constraint that broke the condition is removed. In general, the difference between the least squares method and the active set method is that saturated variables can become 'free' variables again. Furthermore, it determines more precisely which variables should be saturated. A variant of the active set method is provided by Schofield [34] which is especially suited for time varying problems. The method saturates all free variables where the values of the unconstrained maximum are outside the feasible set, where the normal case moves along a line from the current iteration towards unconstrained minimum until it touched the border of the feasible set.

3.4. Multi-parametric quadratic programming

The multi parametric quadratic programming (MPQP) method is an offline method where the computation power required during flight is very low [1, 39]. This comes with a cost of, pre-calculated parameters need to be stored on-board. Another disadvantage occurs when the flight conditions change, which makes it necessary to re-calculate and update the parameters that are stored on-board before flight. This makes the method not suitable for this project.

3.5. Sequential quadratic programming

Another approach to control allocation is dynamic control allocation (DCA) [22]. DCA allocates control while also considering frequency by using information from the previous time steps. The DCA can be considered as a sequential quadratic programming problem (SQP) as shown in equation 3.4.

$$u(k) = \arg\min_{u\in\Omega} \quad (||(W_a(u(k)-u_s(k))||^2 + ||W_b(u(k)-u(k-T))||^2) \tag{3.4a}$$

$$\Omega = \arg\min_{\underline{u}(t)\leq u(t)\leq \overline{u}(t)} ||W_c(Bu(t)-v(t))|| \tag{3.4b}$$

Where the true control input is represented by u, u_s is the desired control input, v is the virtual input, k is the time step and T the sampling time. The weight factors are given by matrices W_a, W_b and W_c. The control effectiveness is given by the matrix B. First, equation 3.4b minimizes the virtual control error (weighted by W_c) by giving the feasible control inputs. Next, it picks the control input that minimizes equation 3.4a. The most important goal is to fulfill the virtual control demand. When this is not possible due to saturation of the actuators, the weight matrix W_c can be changed. It is possible that different control inputs give the same virtual control error. Which is the same as Ω does not contain a single point. This can be solved by minimizing equation 3.4a and tuning it with weight factor matrices W_a and W_b. By increasing W_a, the time it takes to get to the right value will decrease. Increasing W_b will decrease the acceleration of the actuators, reducing the wear and tear. SQP is known for its numerical simplicity [5]. The method tries to find an optimum on an ellipsoid that surrounds the feasible control input space of the actuators. It is not guaranteed that it will find the optimal solution [18]. This is one of the drawbacks of the method as it can give poor results.

3.6. Interior point method

The interior point method is suited for systems with a large number of actuators but can be computationally expensive as a large cost function needs to be minimized [31]. The method is a path following method where it attempts to travel around the central path to find the optimal solution. The interior point method relies on a Lagrangian (equation 3.5), the cost function J needs to be minimised.

$$\min_x \quad J = \frac{1}{2}x^T Hx + c^T x \qquad \text{subject to} \quad x + w = x_{max}, \quad x \geq 0 \quad w \geq 0 \tag{3.5}$$

$$H = 2(G^T G + hI) \qquad c^T = -2(a_0^T G + hx_0^T)$$

$$x = u - u_{min} \qquad x_0 = u_0 - u_{min} \qquad a_0 = a_d - Gu_{min}$$

Here G is the control effectiveness matrix, h is the weight factor of the actuators, w is the slack variable, a_d is the acceleration vector and u_0 is the preferred control value. According to J. Petersen and M. Bodson [31] the method is efficient for a small number of actuators. For this project there are 16 actuators.

3.7. Fixed point method

The fixed point method is able to find an optimal point but needs many iterations [5]. It does this by solving the recursive fixed point contraction algorithm. The cost function must be written in terms of x_n and x_{n+1}. For the first iteration, a starting value is assumed for x_n. For each subsequent iteration, the value of x_{n+1} from the previous is assigned to x_n. The method should converge on a single value for x_n. It converges if x_{n+1} has a derivative with respect to x_n that has an absolute value smaller than 1. An example how to use the fixed point method for a general function:

$$f(x) = x^2 - x - 1 = 0 \tag{3.6a}$$

$$x^2 = x + 1 \tag{3.6b}$$

$$x = 1 + \frac{1}{x} \tag{3.6c}$$

$$x_{n+1} = 1 + \frac{1}{x_n} \tag{3.6d}$$

In order to determine whether this method converges, the derivative needs to be determined, as shown in equation 3.7. If the initial value of x_0 is 2 the absolute value of equation 3.7b will be 0.25 which is smaller than one, so the method converges in this case.

$$g(x) = 1 + \frac{1}{x} \tag{3.7a}$$

$$g'(x) = -\frac{1}{x^2} \tag{3.7b}$$

The method is easy to implement, but converges slowly. This disadvantage is usually ameliorated by choosing a fixed maximum number of iterations.

Introduction part II, III and IV

Part II, III and IV continue with the information gathered in part I. Part II includes the paper. The paper describes the dynamic tilt-twist method in combination with the INDI. The theory, simulations and flight test are described in the paper.

Within Part III conclusions are drawn and recommendations are given about the complete thesis report.

The appendixes are included in part IV. The ground effect acting on the Nederdrone is one of the problems the tailsitter has during the landing phase. A closer look is given towards this problem in appendix A.

This thesis focuses on the controller of the Nederdrone. Both the INDI and IBS controller were promising controllers. To compare the two controllers a pendulum simulation is performed in Matlab, see appendix B.

The feedback error used in the controller can improve the hover capabilities of the Nederdrone. The tilt-twist method and the quaternion method are compared by performing a simulation in Matlab. This is mentioned in appendix C.

The INDI controller with dynamic feedback error method, mentioned in the paper, is used during test flights. The results of these test flights are shown in appendix D. During these test flights the Nederdrone used the INDI controller with the quaternion feedback method, tilt-twist method and the dynamic tilt-twist method subsequently.

II

Paper

Hybrid UAV Attitude Control using INDI and Dynamic Tilt-Twist

L.F.A. Dellemann - 4616057
Delft University of Technology

ABSTRACT

The application of Unmanned Aerial Vehicles (UAVs) is increasing, much like the performance of these aircraft. A tailsitter is a type of UAV which is capable of performing vertical take-offs and landings (VTOL) and long endurance flights. During hover, the yaw control is limited due to the dynamics of these tailsitters. The generally used quaternion feedback for the attitude does not compensate for this as it describes a singular rotation. Tilt-twist is a solution to the problem as it splits the tilt (pitch and roll) from the twist (yaw). The axis of the yaw rotation is body fixed. When hovering with a pitch and/or roll angle the twist axis will be aligned with the body z-axis, instead of the desired gravitational force vector (for position control). Previous tilt-twist methods used a PID controller. This paper describes an improvement over previous tilt-twist approaches, the dynamic tilt-twist in combination with INDI. The INDI controller is designed for nonlinear systems. The dynamic tilt-twist compensates for the problem with the normal tilt-twist as test results will demonstrate. Tests are performed in a simulation and a real life test with the NederDrone hybrid tailsitter is done.

1 INTRODUCTION

The market for applications of unmanned aerial vehicles is increasing [1]. The first UAV was made in 1849 (a hot air balloon) [2]. Until 1984, the use of this type of vehicles was small. From 1984 onward, the military started investing in remotely controlled UAVs [3] and first used them during the 1990's. Due to the development of small processors the UAV became available for the public since the last couple of decades [4]. Many companies are currently developing UAVs for various purposes [5] [6]. The reasons to use this type of aircraft are their low cost, high maneuverability and ease of use. A quadcopter is a type of UAV which can perform VTOL and the maneuverability is agile. One of the major drawbacks of quadcopters is their flight endurance. The fixed wing UAV has a shape comparable to the shape of an conventional aircraft. A fixed wing aircraft is typically able to cover larger distances than quadcopters, but is not able to hover. A solution to achieve both the benefits of a quadcopter (high maneuverability) and a fixed wing aircraft (able to cover long distances) is to use a hybrid UAV (Figure 1). This type of aircraft is able to take-off vertically.

Figure 1: The Nederdrone, capable of performing VTOL. The tailsitter has 20 actuators (12 motors and 8 elevons) and the energy is stored in a hydrogen tank.

The most common hybrid UAV types are tilt-rotors [7], tilt-wings [8], tailsitters [9] and quadplanes [10]. Both the tilt-rotors and tilt-wings have components which can rotate during the transition between hover and forward flight. A quadplane uses different actuators for hover and forward flight. A tailsitter uses the same actuators in hover and forward flight and rotates the entire body of the aircraft. Once it is in the air, it is able to pitch down and fly like a fixed wing aircraft. One of the challenges with this type of UAV is its controllability during the landing phase, especially during hard and unpredictable wind conditions, which regularly occur on vessels [11]. The challenge originates in the nonlinear behaviour of the UAV and the high inertia of the wings. During hover, some of the actuators become less efficient, the wings experience turbulent and hard wind while the airflow over the elevons is minimized. Furthermore, the motors need to handle yaw control while the torque is reduced by efficient props, which results in saturated actuators. Better control behaviour is achieved during low wind speeds.

1.1 Types of tailsitters

This research is conducted on a tailsitter. In order to get a better understanding of how this type of UAV operates, different variants will be presented. This includes an examination of the different components and their functions. The tailsitter

in Figure 2 was one of the first tailsitters [12], which was an operational tailsitter that was not developed further due to the required level of pilot experience. The aircraft was hard to control as there were no advanced controllers to help the pilot. With two props, yaw control was possible by varying torque. It had two elevons and a rudder that were used to control the attitude. Another disadvantage of this model was the handling capability during the hover phase. During hover it had to be stabilised by the aerodynamic control surfaces, but the airflow over the control surfaces was very small which made them less effective.

The UAVs from Figures 1, 3 and 4 were developed by the TU Delft. The Cyclone [13] (Figure 3) has only two aerodynamic control surfaces (i.e. elevons) and two motors. The control surfaces are used to control the pitch and the yaw rotations. The props are used to generate thrust and control roll in hover. Rolling with the props has a side effect in yaw due to the torque difference between the propellers. It needs to be compensated by the elevons during the hover phase. The pitch of the props cannot be changed which leads to low performance, as the pitch angle for optimal efficiency differs between hover and cruise flight [14]. The Delftacopter has seven actuators [11]. The rotor of the Delftacopter is very large compared to the wings (fuselage), which means that the rotor contributes more to the rotational inertia than is typical for quadcopters and less for helicopters. This interferes significantly with the dynamics of the Delftacopter and complicates the control system. The NederDrone has 20 actuators, 12 of which generate lift during hover.

Figure 2: Convair XYF-1

Figure 3: The Cyclone

Figure 4: Delftacopter

1.2 NederDrone

The tailsitter used for this project is the NederDrone [9], shown in Figure 1. The NederDrone is being developed in cooperation with the Dutch defense department. Eventually the tailsitter must be able to perform long endurance recon flights autonomously. This will be accomplished by using a hydrogen tank, with a target flight time of four hours. The NederDrone has two wings which carry twelve engines and hold eight elevons. The twelve engines are all used during hover, but only four are used during forward flight. The wingspan is approximately two meters. Several problems need to be solved for the autonomous landing of the tailsitter on a moving vessel during high wind distortions:

1. Wind conditions affected by the ship
2. Ground effect
3. Tracking the vessel's position
4. Actuator saturation during hover
5. Nonlinear flight dynamics

The objective of this study is to improve the autonomous landing capabilities of the NederDrone when landing on a moving vessel, where high wind distortions are present, by designing or improving the autopilot and demonstrating its performance experimentally. To solve the problems listed above, several aspects of the tailsitter can be changed, such as the type of controller [15]. This paper focuses on handling problems 1, 4 and 5, by handling the feedback error.

The feedback error is generally calculated by using the quaternions. Euler angles are also an option but have certain limitations, such as gimbal lock, and are less efficient as more computational power is needed [16]. The quaternion method does have limitations when used in tailsitters. It calculates the shortest path to the new position without considering the effectiveness of the actuators. During hover, yaw is harder to control for tailsitters than pitch and roll. The newly proposed method was based on the research done by Matsumoto et al. [16]. This research used a different feedback error method, called the tilt-twist method. Better hover position control was achieved by splitting the tilt error (pitch en roll) and the twist error (yaw). When larger pitch angles are present (due to wind) the twist will be around the body fixed z-axis instead of the desired gravitation force vector. Previous work described the method in combination with a PID controller. This paper proposes a modification of the already existing tilt-twist method called the dynamic tilt-twist method. This modification will allow it to be used with larger pitch and roll angles during hover. Furthermore, the (dynamic) tilt-twist method is tested in combination with a incremental nonlinear dynamic inversion (INDI) controller, which is designed for nonlinear systems [17].

In Section 2 the theory behind the feedback error will be given. This includes the attitude representation. The three methods (quaternion, tilt-twist and dynamic tilt-twist) that will be compared are also included in Section 2. Section 3 describes the methodology for both the simulation and the real life test. The results of the simulations and the real-life test are also presented in Section 3. Conclusions are drawn in the last Section.

2 METHOD

2.1 Axis definition

A axis system needs to be defined in order to describe the feedback error. Both the required attitude and the current attitude must be defined in the same coordinate frame. The axes are body fixed, where the z-axis is parallel to the gravitational force during hover, when the pitch and roll angles are zero. The x-axis goes through the belly of the UAV and the y-axis through the right wing (see Figure 5). In forward flight, the axes are kept body fixed, which implies that the z-axis is perpendicular to the gravitational force vector when the pitch is -90 degrees and the roll angle is zero.

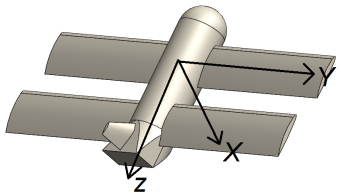


Figure 5: Axis definition, body fixed. If the pitch and roll angles are 0 degrees, the z-axis is parallel to the gravitational force vector.

2.2 Quaternion

A commonly used feedback method in the industry is the quaternion method [18]. The quaternions describe the transition of the attitude in one single rotation, when using the Euler angles the rotation is calculated in three subsequent rotations. Another difference between the quaternions and the Euler angles is the computational efficiency [19]. Quaternions can be calculated faster as no trigonometric functions are required. In addition, the problem of gimbal lock does not occur. A quaternion describes the attitude with a rotation and a three dimensional unit vector component, see Equation 1.

$$\boldsymbol{q} = \begin{bmatrix} \cos(\frac{\eta}{2}) \\ \boldsymbol{r}\sin(\frac{\eta}{2}) \end{bmatrix} = \begin{bmatrix} q_i & q_x & q_y & q_z \end{bmatrix}^\top \tag{1}$$

To calculate the errors by using the quaternions, the current and the desired quaternions need to be known. The current attitude is given by

$$\boldsymbol{q}_c = \begin{bmatrix} q_{ci} \\ q_{cx} \\ q_{cy} \\ q_{cz} \end{bmatrix} \tag{2}$$

The desired attitude is given by

$$\boldsymbol{q}_d = \begin{bmatrix} q_{di} \\ q_{dx} \\ q_{dy} \\ q_{dz} \end{bmatrix} \tag{3}$$

The attitudes $\boldsymbol{q}_c$ and $\boldsymbol{q}_d$ are used to calculate the errors for the controller, as shown in Equations 4 and 5.

$$\boldsymbol{q}_{err} = \boldsymbol{q}_d \otimes \boldsymbol{q}_c^{-1} \tag{4}$$

$$\boldsymbol{q}_{err} = \begin{bmatrix} q_{err1} \\ q_{err2} \\ q_{err3} \\ q_{err4} \end{bmatrix} = \begin{bmatrix} q_{d0} & q_{d1} & q_{d2} & q_{d3} \\ -q_{d1} & q_{d0} & q_{d3} & -q_{d2} \\ -q_{d2} & -q_{d3} & q_{d0} & q_{d1} \\ -q_{d3} & q_{d2} & q_{d1} & q_{d0} \end{bmatrix} \boldsymbol{q}_c \tag{5}$$

The controller needs to separate errors for the x-, y- and z-axis. These can be found with Equations 6, 7 and 8, these Equations are used for a PD controller. δ_a, δ_e and δ_r represent the aileron deflection, elevator deflection and rudder deflection, respectively.

$$\delta_a = -2(k_p q_{err_2} + k_d \dot{q}_{err_2}) \tag{6}$$

$$\delta_e = -2(k_p q_{err_3} + k_d \dot{q}_{err_3}) \tag{7}$$

$$\delta_r = -2(k_p q_{err_4} + k_d \dot{q}_{err_4}) \tag{8}$$

The quaternions are limited when saturation of the actuators occur or the response time of the actuators is insufficient. If a large yaw error occurs while the pitch angle is small, the quaternion feedback controller will find the shortest path to the required attitude, for which the yaw error will be the most significant factor. To align the thrust vector with the gravitational vector the most relevant part is the pitch error.

2.3 Tilt-Twist

When a tailsitter hovers, complications occur because the twist (yaw) is limited, due to saturation of the actuators. The thrust/tilt (combination of pitch and roll) component is reliable for the xyz position. In the situation where a hovering tailsitter needs to follow a position the tilt-twist is a solution. During perfect hover, the tilt is zero degrees when the body z-axis is opposite to the gravitational vector. As was mentioned before, this method splits the motion into a tilt and a twist component. This method is described by Matsumoto et al [16]. Beach et al. improved the method to make it more computationally efficient [20]. The procedure of the tilt-twist method will be explained next.

2.4 Rotation matrix

A rotation matrix is used to rotate from one frame to another and is calculated based on the current and the desired attitudes, as shown in 9. The matrix will be used for both the tilt-twist method and the dynamic tilt-twist method.

$$\boldsymbol{R}(\boldsymbol{q}) = \begin{bmatrix} q_i^2+q_x^2-q_y^2-q_z^2 & 2(q_x q_y+q_z q_i) & 2(q_x q_z-q_y q_i) \\ 2(q_x q_y-q_z q_i) & q_i^2-q_x^2+q_y^2-q_z^2 & 2(q_y q_z+q_x q_i) \\ 2(q_x q_z+q_y q_i) & 2(q_y q_z-q_x q_i) & q_i^2-q_x^2-q_y^2+q_z^2 \end{bmatrix} \tag{9}$$

Rotation matrices can be calculated for both the current ($\boldsymbol{R}_c$) and the desired ($\boldsymbol{R}_d$) attitudes, as shown in Equation 10.

$$\boldsymbol{R}_d = \boldsymbol{R}(\boldsymbol{q}_d) \quad \boldsymbol{R}_c = \boldsymbol{R}(\boldsymbol{q}_c) \tag{10}$$

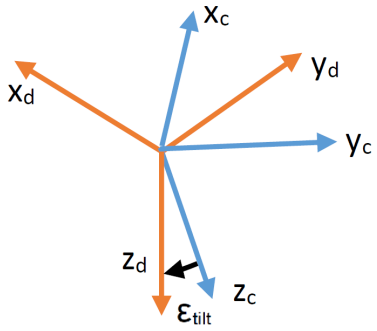


Figure 6: Total tilt error definition. In this case the pitch error is 10°, the roll error 5° and the yaw error is 60°.

2.4.1 *Tilt error*

The first part of the tilt-twist method consists of calculating the tilt error. As was mentioned before, this error is a combination of the pitch- and the roll error. The error is calculated using the rotation matrices to align the current frame with the desired frame, as shown in Equation 11. If the pitch and roll error are small compared to the yaw error, the total tilt error can be defined according to Figure 6.

$$\boldsymbol{R}_d = \boldsymbol{R}_{err}\boldsymbol{R}_c \tag{11}$$

Equation 11 can be rewritten as,

$$\boldsymbol{R}_{err} = \boldsymbol{R}_d\boldsymbol{R}_c^\top = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \tag{12}$$

The axis definition from Matsumoto et al. is different from the one used in this paper. Here, the tilt error is about the x- and y-axis instead of the y- and z-axis. The third row component of matrix $\boldsymbol{R}_{err}$ will provide the tilt error. The x component of the tilt error is given by

$$Tilt\ error_1 \triangleq \varepsilon_x = -atan2(r_{3,2}, r_{3,3}) \tag{13}$$

Where $atan2$ is the inverse tangent. The negative sign ensures that the rotation is in the desired direction. The y component is given by Equation 14.

$$Tilt\ error_2 \triangleq \varepsilon_y = atan2(r_{3,1}, r_{3,3}) \tag{14}$$

2.4.2 *Twist error*

In the second part of the tilt-twist method the twist error is calculated, which is the given error around the body fixed z-axis. To calculate the error, an intermediate coordinate frame is formulated to align the current z-axis with the desired z-axis. This is done by using the rotation matrices,

$$\boldsymbol{R}_d = \begin{bmatrix} \boldsymbol{r}_{d1} \\ \boldsymbol{r}_{d2} \\ \boldsymbol{r}_{d3} \end{bmatrix} \quad \boldsymbol{R}_c = \begin{bmatrix} \boldsymbol{r}_{c1} \\ \boldsymbol{r}_{c2} \\ \boldsymbol{r}_{c3} \end{bmatrix} \tag{15}$$

The total tilt error will be used to align both the z-axes. The complete tilt error is calculated by using both $\boldsymbol{R}_c$ and $\boldsymbol{R}_d$, as shown in Equation 16.

$$Tilt\ error \triangleq \varepsilon_{tilt} = \cos^{-1}\left(\boldsymbol{r}_{d3}^\top \cdot \boldsymbol{r}_{c3}^\top\right) \tag{16}$$

In Equation 17 the unit length axis is defined.

$$k = \frac{\boldsymbol{r}_{c3}^\top \times \boldsymbol{r}_{d3}^\top}{|\boldsymbol{r}_{c3}^\top \times \boldsymbol{r}_{d3}^\top|} \tag{17}$$

As the rotation needs to happen in the vehicle frame the unit vector k needs to be in the vehicle frame, which is done by

$$\boldsymbol{v}^b = \boldsymbol{R}_c k = \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \end{bmatrix} \tag{18}$$

and

$$\boldsymbol{v} = \begin{bmatrix} 0 & -v_z^b & v_y^b \\ v_z^b & 0 & -v_x^b \\ -v_y^b & v_x^b & 0 \end{bmatrix} \tag{19}$$

Next a rotation matrix needs to be defined which rotates a vector around a vector. The angle of this vector is also included. This is accomplished by using the Rodrigues' rotation formula [21], where the angle is ε_{tilt} and the vector is $\boldsymbol{v}^b$. The rotation matrix is given by Equation 20.

$$\boldsymbol{R}_v = \begin{cases} \boldsymbol{I}, & \varepsilon_{tilt} = 0 \\ \boldsymbol{I} - \boldsymbol{v}\sin(\varepsilon_{tilt}) + \boldsymbol{v}^2[1 - \cos(\varepsilon_{tilt})], & \varepsilon_{tilt} \neq 0 \end{cases} \tag{20}$$

The attitude of the UAV can be compensated with the tilt error, which can be found by using $\boldsymbol{R}_v$. By multiplying $\boldsymbol{R}_v$ with the rotation matrix of the current attitude of the UAV ($\boldsymbol{R}_c$) a new rotation matrix $\boldsymbol{R}_p$ is found, as shown in Equation 21.

$$\boldsymbol{R}_p = \boldsymbol{R}_v^\top \boldsymbol{R}_c \tag{21}$$

Where

$$\boldsymbol{R}_p = \begin{bmatrix} \boldsymbol{r}_{p1} \\ \boldsymbol{r}_{p2} \\ \boldsymbol{r}_{p3} \end{bmatrix} \tag{22}$$

The absolute twist error can be found using the x components of $\boldsymbol{R}_p$ and $\boldsymbol{R}_d$, see Equation 23. In Figure 7 the frame has

already been compensated for the tilt error and the twist error is shown.

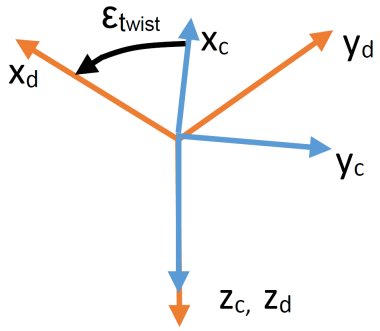


Figure 7: Twist error definition. In this case the pitch error is 0° and the roll error 0° due to the compensation for the tilt error.

$$\varepsilon_{twist} = \cos^{-1}(\boldsymbol{r}_{p1}^{\top} \cdot \boldsymbol{r}_{d1}^{\top}) \tag{23}$$

The sign of the twist error needs to be determined, as Equation 23 gives the absolute value of the twist error. By using the y component of $\boldsymbol{R}_p$ this can be achieved, see Equation 24.

$$\varepsilon_{sign} = \cos^{-1}(\boldsymbol{r}_{p2}^{\top} \cdot \boldsymbol{r}_{d1}^{\top}) \tag{24}$$

When this value is above 90°, or $\frac{\pi}{2}$ radials, the sign of the twist error becomes negative. The final twist error can be seen in Equation 25.

$$Twist\ error \triangleq \varepsilon_z = \begin{cases} \varepsilon_{twist}, & \varepsilon_{sign} \leq \frac{\pi}{2} \\ -\varepsilon_{twist}, & \varepsilon_{sign} > \frac{\pi}{2} \end{cases} \tag{25}$$

Together, the calculated errors form the total feedback error, given by:

$$Feedback\ error \triangleq \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} \tag{26}$$

For a PD controller, the actuator deflections of the ailerons, elevator and rudder are calculated with Equation 27, 28 and 29.

$$\delta_a = k_p \varepsilon_x - k_d \dot{\varepsilon}_x \tag{27}$$

$$\delta_e = k_p \varepsilon_y - k_d \dot{\varepsilon}_y \tag{28}$$

$$\delta_r = k_p \varepsilon_z - k_d \dot{\varepsilon}_z \tag{29}$$

In Figure 8 and 9 the difference is depicted of a quaternion feedback error method and the tilt-twist method. A simulation is performed where there is a small pitch and roll error and a large yaw error. The tilt-twist method solves the pitch and roll in a shorter time period then the quaternion feedback method. Which will result in larger position errors for the quaternion method. From Figure 9 it can be seen that the tilt twist method uses both pitch and roll to solve the problem.

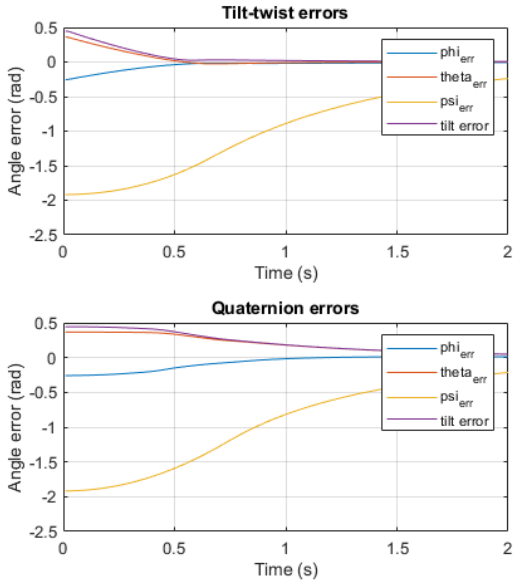


Figure 8: Error handling comparison between tilt-twist and quaternion feedback, in Euler angles. The tilt error is also shown, which is linked to the trust vector.

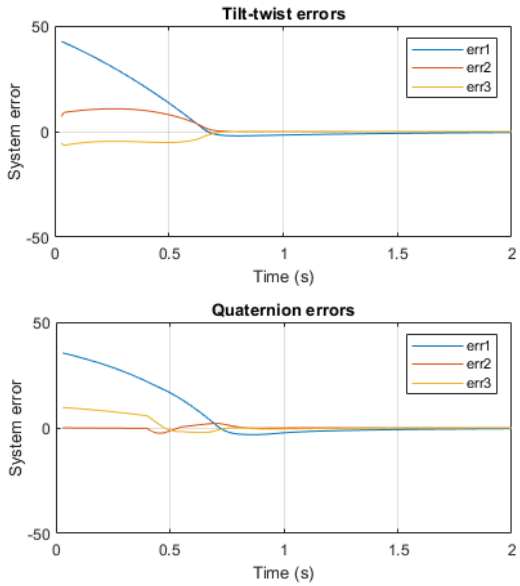


Figure 9: Error handling comparison between tilt-twist and quaternion feedback, inputs for the system.

2.5 *Dynamic Tilt-Twist*

In the tilt-twist method the twist component is always measured around the body fixed z-axis. However, the Nederdrone

can still have a very large pitch angle (e.g. 60 degrees) when hovering with significant wind. When landing on a vessel the twist vector should be parallel with the gravitational force factor, to ensure position control. The body tilt axis does not correspond to the lift vector anymore. The idea of having the lift vector making the shortest path becomes invalid. Instead, as the wings of the hybrid UAV still participate heavily in the lift production, the tilt axis is redefined to be parallel with the gravitational vector. To align the twist vector with the gravitational vector the rotational matrix $\boldsymbol{R}_a$ is used, see Equation 30. It requires the pitch (θ) and roll (ϕ) angle. When the pitch angle is -90° the rotation matrix contains the values shown in Equation 31. Effectively, the first and third row are swapped. The twist will no longer be around the x-axis but around the z-axis.

$$\boldsymbol{R}_a(\theta,\phi) = \begin{bmatrix} \cos -\theta & \sin -\theta \sin -\phi & \sin -\theta \cos -\phi \\ 0 & \cos -\phi & -\sin -\phi \\ -\sin -\theta & \cos -\theta \sin -\phi & \cos -\theta \cos -\phi \end{bmatrix} \tag{30}$$

$$\boldsymbol{R}_a(-\frac{\pi}{2},0) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{31}$$

The procedure to calculate the actuator deflections is similar to the one described in Section 2.3 except for the changes described below. First, the rotation matrices from Equation 10 need to be adjusted by using the matrix from Equation 30. The new rotation matrices are given by Equations 32 and 33.

$$\boldsymbol{R}_d = \boldsymbol{R}_a \boldsymbol{R}(\boldsymbol{q}_d) \tag{32}$$

$$\boldsymbol{R}_c = \boldsymbol{R}_a \boldsymbol{R}(\boldsymbol{q}_c) \tag{33}$$

The last step is to determine the actuator deflections to compensate again for the dynamic tilt angle. $\boldsymbol{R}_a$ is modified by multiplying both θ and ϕ by -1 to produce $\boldsymbol{R}_{-a}$, which is used to adjust the feedback error, as shown in Equation 30:

$$Feedback\ error \triangleq \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} \boldsymbol{R}_{-a} \tag{34}$$

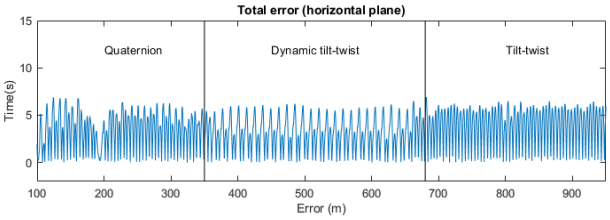
3 Simulation and Flight Test

The Nederdrone is used to perform the simulations and the flight test. The controller used during the test is an INDI controller [22]. Three different feedback errors are compared, quaternion feedback, tilt-twist and dynamic tilt-twist. The setup is identical for the simulations and the flight test. The tests were conducted by flying the Nederdrone from point 1 to point 2 and back. When it hovers at a point for three seconds the setpoint is instantly changed with a heading offset ψ_{change}. An artificial twist error is introduced (bypassing the reference model) to simulate the effect of turbulence acting on a wing and inducing saturation. Thereafter it flies to the other point. The goal was to simulate the conditions the Nederdrone would experience during a landing procedure on a moving vessel. The sudden change in heading represents a sudden gust of wind. When landing on a moving vessel the x- and y location also change over time, which is simulated by instructing the drone to fly from point 1 to point 2 and back again, repeatedly.

3.1 Results

3.1.1 Simulation

Before the flight tests, the different types of feedback errors were tested in a simulation. The same controller settings were used for the flight test. The heading offset (ψ_{change}) during the simulation was 160°. The simulation results can be seen in Figure 11 and 10. The simulations were performed with a low fidelity model of the Nederdrone. Detailed PD tuning was not performed. The maximum accelerations were high, which is undesirable, as it causes extra stress on the drone. PD tuning would result in lower accelerations. The simulation was performed to get an impression of the relative improvement of the tilt twist and dynamic tilt twist methods over the quaternion feedback method. During the simulation, most wind effects were ignored, except for the wind gust described earlier. As can be seen from the Figures below, the quaternion feedback resulted in irregular behaviour. The tilt-twist and dynamic tilt-twist methods were much more consistent.



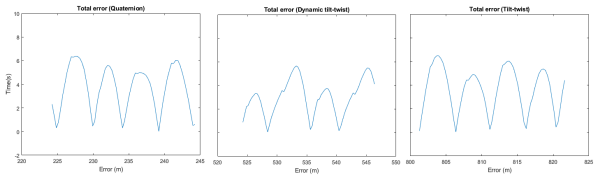
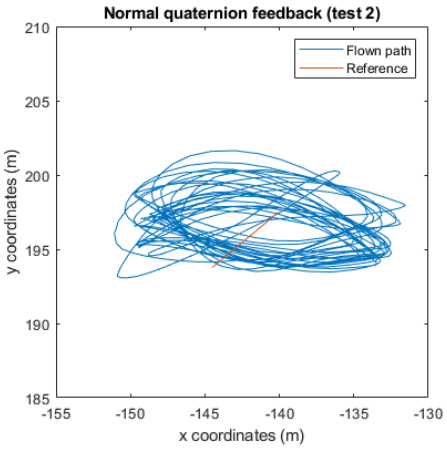
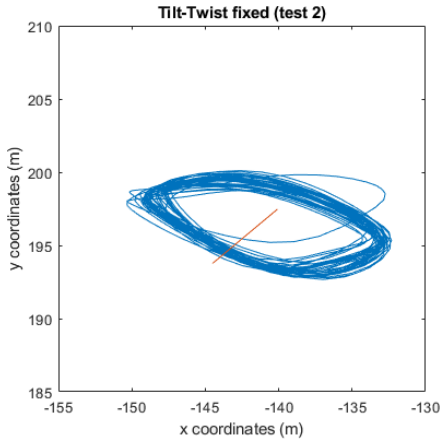


Figure 10: Total error in the horizontal plane (reflection on earth). Normal quaternion 0-350 seconds, dynamic tilt-twist 350-680 seconds, tilt-twist 680-980 seconds





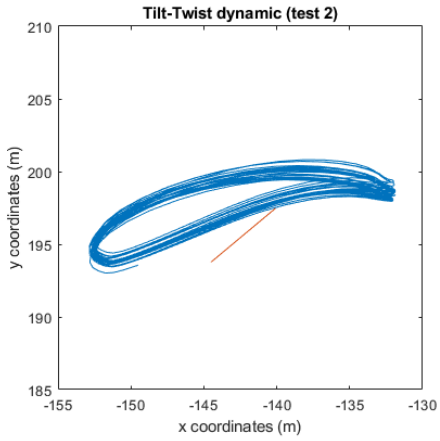


Figure 11: Flight paths during simulation (quaternion, tilt-twist, dynamic tilt-twist). The quaternion feedback method had problems with handling the yaw angle. Both the tilt-twist and the dynamic tilt-twist method showed more stable behaviour.

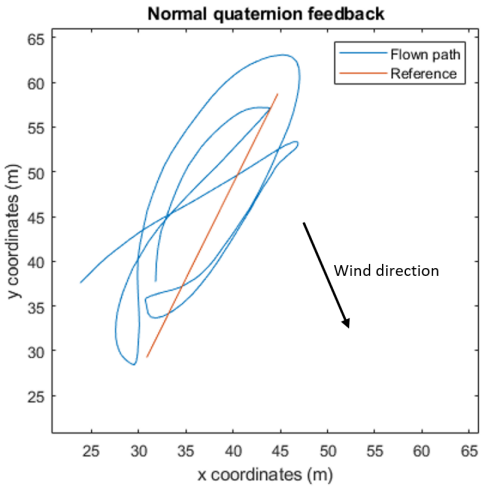
3.1.2 Flight test

During the flight test the wind conditions were challenging for the Nederdrone. The smallest angle between the wind and the path (orange) was 50°. Yaw control of the Nederdrone becomes easier when flying parallel to the wind, as the wind helps with rotating the tailsitter. The heading offset, ψ_{change}, was 45°. Higher heading offsets could result in an unsafe flight, especially when the quaternion feedback controller is used. Due to the large instantaneous error, the tilt component can become unstable.

In Figure 12.A the flight path of the Nederdrone can be seen when the quaternion feedback is used. It needed to follow the orange line, but instead it behaved irregularly, with a larger average distance from the line, see Table 1. When the tilt-twist method was used the behaviour improved (Figure 12.B), with a consistent flight path. The dynamic tilt-twist method improved the behavior even further (Figure 12.C), demonstrating an even smaller average distance from the reference line. The average distances for the different methods are shown in Table 1. A straight line was chosen for the reference path to simplify the calculation of the distance between the drone and the path, which made it possible to calculate this distance for every time step.

Table 1: Test results, distance from reference line.

	Average distance (m)
Quaternion	3.27
Tilt-twist	2.74
Dynamic tilt-twist	2.08



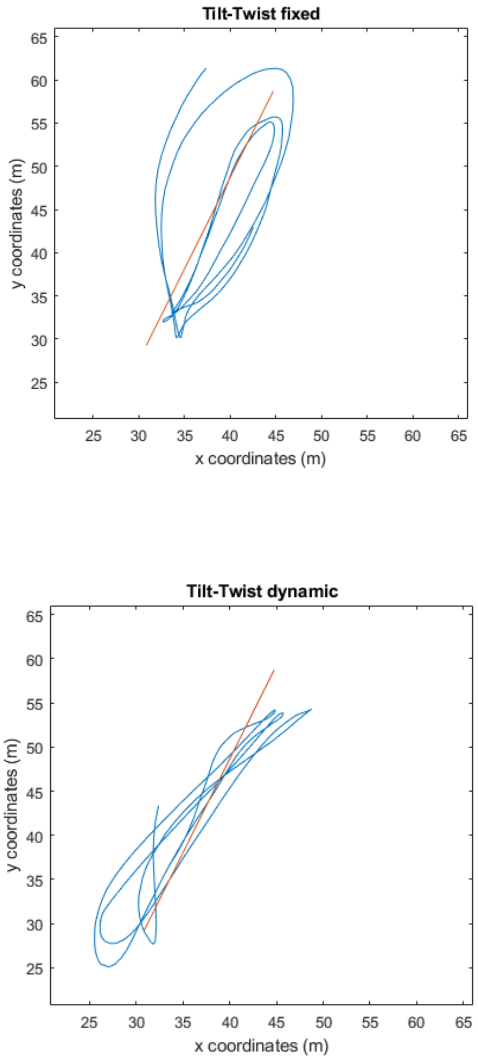


Figure 12: Flown path during flight test, the wind conditions were equal for all flights

4 Conclusion

This paper presented an improvement to the existing tilt-twist method, namely the dynamic tilt-twist method. The method was explained and compared to the quaternion feedback method and the tilt-twist method. The simulation and the flight test demonstrated that the quaternion feedback method had problems following the required path, whereas the tilt-twist method showed some improvements and the dynamic tilt-twist showed the best results. The dynamic tilt-twist method is suitable for tailsitters that vary their pitch and roll angles during hover and experience yaw/position control problems. In forward flight, the twist is rotated -90 degrees and can be used to align the twist vector along the axis where the rudder acts, which proved to be an effective addition to the tilt-twist method.

References

[1] G. Mattei L. Canetta and A. Guanziroli. Exploring commercial UAV market evolution from customer requirements elicitation to collaborative supply network management. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1016–1022, June 2017.

[2] V. Ambrosia A. Watts and E. Hinkley. Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use. 2012.

[3] J. H. Christner. Pioneer unmanned air vehicle accomplishments during Operation Desert Storm. In Thomas W. Augustyn and Paul A. Henkel, editors, *Airborne Reconnaissance XV*, volume 1538, pages 201 – 207. International Society for Optics and Photonics, SPIE, 1991.

[4] P. Murrieri S. Bouabdallah and R. Siegwart. Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pages 4393–4398 Vol.5, 2004.

[5] A. Kumar A. Saha and A. K. Sahu. FPV drone with GPS used for surveillance in remote areas. In *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 62–67, 2017.

[6] S. Ge K. Feng, W. Li and F. Pan. Packages delivery based on marker detection for UAVs. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 2094–2099, Aug 2020.

[7] G. Flores and R. Lozano. Transition flight control of the quad-tilting rotor convertible MAV. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 789–794, 2013.

[8] N. Okada K. Muraoka and D. Kubo. *Quad Tilt Wing VTOL UAV: Aerodynamic Characteristics and Prototype Flight*.

[9] R. Ruijsink F. van Tienen E. van der Horst C. De Wagter, B. Remes. Design and Testing of a Vertical Take-Off and Landing UAV Optimized for Carrying a Hydrogen Fuel Cell with a Pressure Tank. *Unmanned Systems*, 08(04):279–285, 2020.

[10] A. Wang and T. Chan. Estimation of Drag for a Quad-Plane Hybrid Unmanned Aerial Vehicle. 11 2017.

[11] E. Smeur K. van Hecke F. van Tienen E. van der Horst C. De Wagter, R. Ruijsink and B. Remes. Design, control, and visual navigation of the DelftaCopter VTOL

tail-sitter UAV. *Journal of Field Robotics*, pages 937–960, 2018.

[12] W. F. Chana and J. F. Coleman. World's First VTOL Airplane Convair/Navy XFY-1 Pogo. *SAE Transactions*, pages 1261–1266, 1996.

[13] H. Garcia de Marina M. Bronz, E. J. Smeur and G. Hattenberger. Development of A Fixed-Wing mini UAV with Transitioning Flight Capability. In *35th AIAA Applied Aerodynamics Conference*.

[14] L. Meier R. Bapst, R. Ritz and M. Pollefeys. Design and implementation. 2015.

[15] E. Smeur. Incremental Control of Hybrid Micro Air Vehicles. 2018.

[16] R. Suzuki A. Oosedo K. Go Y. Hoshino A. Konno T. Matsumoto, K. Kita and M. Uchiyam. A Hovering Control Strategy for a Tail-Sitter VTOL UAV that Increases Stability Against Large Disturbance. 2010.

[17] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. *Journal of Guidance, Control, and Dynamics*, 33(6):1732–1742, 2010.

[18] H. Abaunza J. Cariño and P. Castillo. Quadrotor quaternion control. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 825–831, June 2015.

[19] C. Milazzo A. Ricciardello A. Alaimo, V. Artale. Comparison between Euler and quaternion parametrization in UAV dynamics. *AIP Conference Proceedings*, 1558(1):1228–1231, 2013.

[20] T. McLain R. Beard J. Beach, M. Argyle and S. Morris. Tailsitter attitude control using resolved tilt-twist. 2014.

[21] E. W. Weisstein. *Rodrigues' Rotation Formula.*, 2020 (accessed October 9, 2020) https://mathworld.wolfram.com/RodriguesRotationFormula.html.

[22] R. C. van 't Veld. Incremental Nonlinear Dynamic Inversion Flight Control. 2019.

III

Conclusions and recommendations

Conclusions and recommendations

Within this thesis is investigated on how to improve the landing performance of the Nederdrone. The goal of this thesis was defined by:

Improve the landing capabilities of the NederDrone when landing on a vessel with high distortions (wind) by designing a suitable controller and demonstrating its performance experimentally.

By looking into the controller an improvement has been realised. Using an INDI controller in combination with dynamic tilt-twist showed sufficient improvements from both the simulations as the test flight. Within the literature study the current problems with the Nederdrone were mentioned during the landing phase. The wind conditions, non-linearity and the saturation of the actuators were the problems. By using INDI and dynamic tilt-twist the problems were reduced.

Recommendations

During this thesis a flight test was conducted. Flight tests can be done on a real vessel to further validate the proposed method. Furthermore, the dynamic tilt-twist is tested during hover. More tests can be done to receive the performance during forward flight. Control allocation is not further elaborated after the preliminary thesis report. Research into the control allocation can be conducted to further improve the hover performance of the Nederdrone.

IV

Appendix

A

Appendix A Ground Effect

During the last phase of the landing of the UAV the tailsitter experiences a lot of problems due to the ground effect. In order to be able to counteract this effect it will be further elaborated in the chapter.

A.0.1. Problem description

Several flight tests with the NederDrone are already performed for this project. During these flight tests the UAV was mainly tested in order to check what the problems are with the tailsitter. In the final phase of the landing it could be noticed that the ground effect was kicking in and the UAV started tipping forward, see Figure A.1. This is a desired effect as it will land horizontally. The only problem is that it is a uncontrolled movement. To be able to get a controlled pitching forward movement the ground effect will be tested during flight test. Before the flight tests the written theory will be investigated.

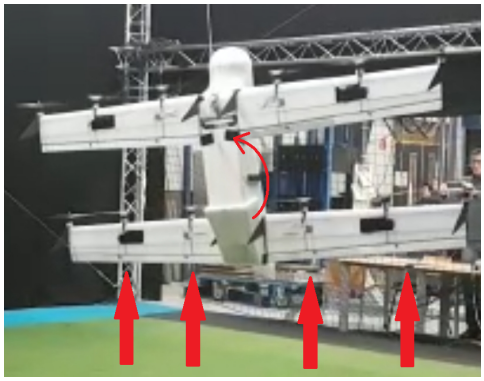


Figure A.1: The ground effect forces on the tailsitter

A.0.2. Ground effect on tailsitters

Not a lot of research is performed on the ground effect on tailsitters. Nevertheless, C. Thipyopas et al. [38] did some research and performed test with a tailsitter. During their research several fight tests were performed with a tailsitter with counter rotating propellers. During the tests the forces and moments due to the ground effect were measured. As the design of the tailsitter is comparable with the design of the NederDrone this research can be used as a reference.

The wing height will influence the ground effect behaviour, as the motors of the tailsitter are above the wings in the situation of hovering (see Figure A.1). According to C. Thipyopas the ground effect kicks in at a height of approximately two times the prop width, when also considering the wing under the props. Which influences the ground effect.

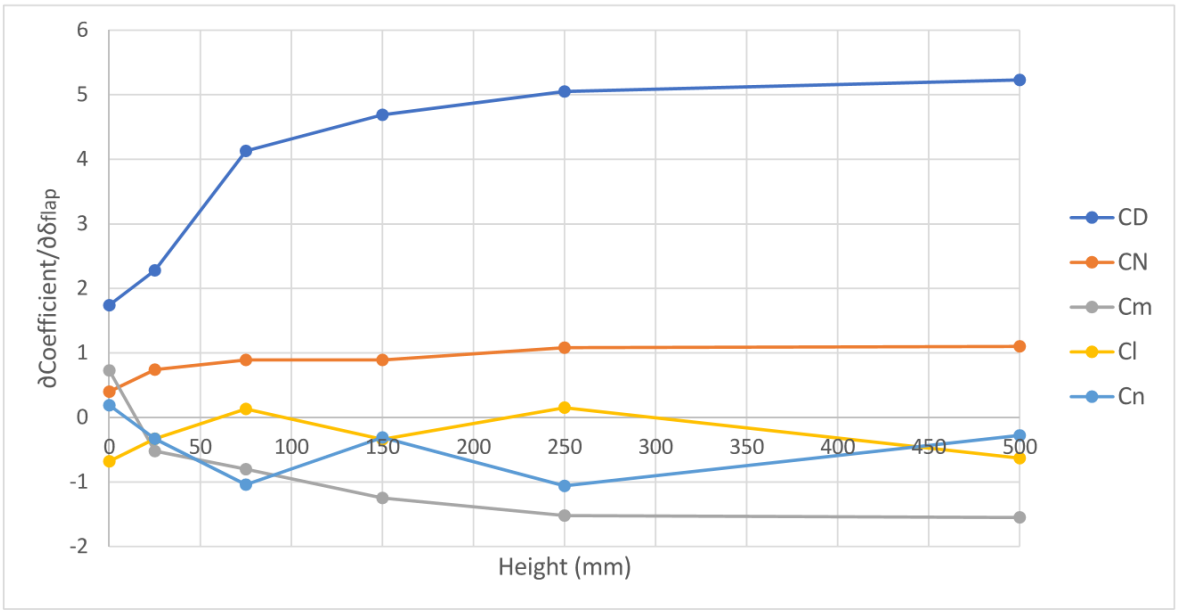


Figure A.2: Coefficients and actuator deflection relations against the altitude

The ground effect reduces the required thrust to hover. The thrust delivered by the motors is nearly constant. However, the down load force coefficient (C_{D_p}) changes, it reduces around 52%. The difference is from out of ground effect region up until just above the ground. The down load force consists of the pressure drag and skin friction in the x direction. The moments and the normal force (z-direction) caused by the ground effect can be neglected as they are close to zero[38]. Nevertheless the efficiency of the actuators (flaps) does decrease. The coefficients change when varying the angle of the flaps at different altitudes and it is quasi linear. In Figure A.2 the results of deflecting the flaps can be seen. It represents the change of the coefficients per degree of deflection against the altitude. It can be noticed it does affect the C_{D_p} more then it does to the other coefficients.

B

Appendix B Pendulum Simulation

Within this section a simulation is performed to compare the INDI controller with the IBS controller. The used model is a pendulum (Figure B.1). It is described by formulas B.1a and B.1b.

$$\dot{x}_1 = x_2 \tag{B.1a}$$

$$\dot{x}_2 = -\frac{g}{l}\sin x_1 - \frac{k}{m}x_2 + \frac{1}{ml^2}u \tag{B.1b}$$

Where x_1 represents the angle around the pivot point, g the gravitational acceleration, l the length of the chord of the pendulum, k the friction coefficient, m the mass of the pendulum (the chord is not included) and u is the input which is the applied torque.

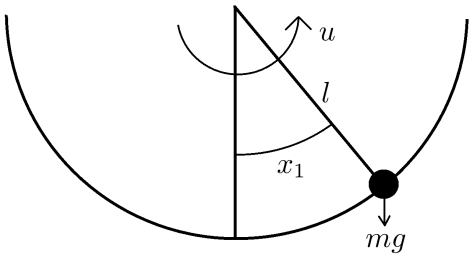


Figure B.1: Pendulum

For convenience equation B.1b will be rewritten into the form:

$$\dot{x}_2 = \beta_1 \sin x_1 + \beta_2 x_2 + \beta_3 u \tag{B.2}$$

Where:

$$\beta_1 = -\frac{g}{l} \quad \beta_2 = -\frac{k}{m} \quad \beta_3 = \frac{1}{ml^2}$$

A reference signal will be generated, which the controller needs to follow. The first reference signal is a step input and the second input is a sinusoidal.
For the simulations the value used are $g = -9.81 m/s^2$, $l = 1m$, $k = 5$ and $m = 10g$.

B.0.1. INDI controller

The INDI controller must be able to reduce the error from equation B.3.

$$z_1 = x_{1r} - x_1 \tag{B.3}$$

x_1 needs to be differentiated until the control input u occurs:

$$\dot{x}_1 = x_2 \tag{B.4a}$$

$$\ddot{x}_1 = \dot{x}_2 = \beta_1 \sin x_1 + \beta_2 x_2 + \beta_3 u \tag{B.4b}$$

From equation B.4b only the last component is dependent of u. Therefore, the control effectiveness consists of β_3. Writing equation B.4b into the incremental form, by using a first order Taylor expansion gives:

$$\dot{x}_2 \cong \dot{x}_{2,0} + \theta_1 \cos x_{1,0} \Delta x_1 + \theta_2 \Delta x_2 + \theta_3 \Delta u \tag{B.5}$$

When using NDI the first and second component are used in the feedback, for an INDI controller the information of the sensor is used to determine $\dot{x}_2$ (feedback). So the δx components are neglected. $\dot{x}_2$ will be written as the virtual input v. The incremental input will become:

$$\Delta u \cong \frac{1}{\theta_3}[v - \dot{x}_{2,0}] \tag{B.6}$$

Where $\frac{1}{\theta_3}$ is the inverse control effectiveness matrix (G^{-1}). To get from the angular state error to the virtual input a PD controller is used. As the incremental feedback will reduce the integration, the I of the PID controller is not required. The complete controller can be seen in Figure B.2

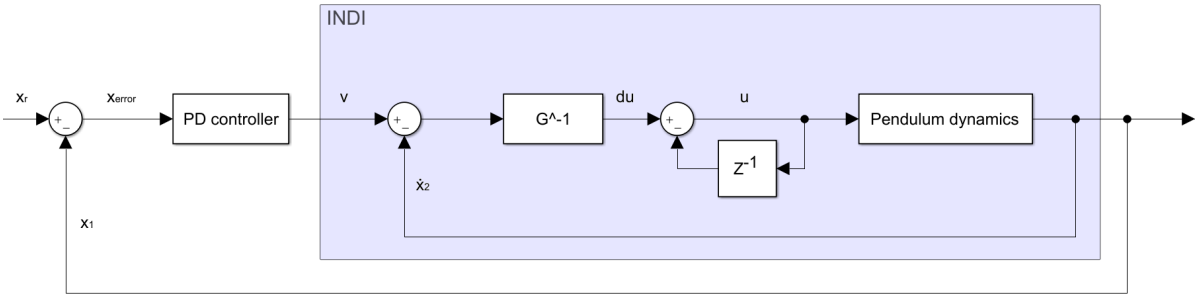


Figure B.2: INDI controller

B.0.2. IBS controller

For the IBS controller several steps are required. The two tracking errors can be defined as:

$$z_1 = x_1 - x_{1r} \tag{B.7a}$$

$$z_2 = x_2 - \alpha_1 \tag{B.7b}$$

Where x_{1r} is the reference state and α_1 is the stabilizing control law. For the incremental part the equation for $\dot{x}_2$ will be rewritten by using the first order Taylor series expansion.

$$\dot{x}_2 \cong \dot{x}_{2,0} + \theta_1 \cos x_{1,0} \Delta x_1 + \theta_2 \Delta x_2 + \theta_3 \Delta u \tag{B.8}$$

As the incremental states are small in comparison to the incremental control ($\theta_1 \cos x_{1,0} \Delta x_1 + \theta_2 \Delta x_2 << \theta_3 \Delta u$) [41] $\dot{x}_2$ will become:

$$\dot{x}_2 \cong \dot{x}_{2,0} + \theta_3 \Delta u \tag{B.9}$$

The differentiation of the set of equations B.7 is:

$$\dot{z}_1 = z_2 + \alpha_1 - \dot{x}_{1r} \tag{B.10a}$$

$$\dot{z}_2 = \dot{x}_{2,0} + \theta_3 \Delta u - \dot{\alpha}_1 \tag{B.10b}$$

For the first subsystem the control Lyapunov function is formulated as:

$$V_1(z_1) = \frac{1}{2} z_1^2 \tag{B.11}$$

Where the derivative is:

$$\dot{V}_1(z_1) = z_1 \dot{z}_1 = z_1[z_2 + \alpha_1 - \dot{x}_{1r}] \tag{B.12}$$

The control law (α_1) will be defined as:

$$\alpha_1 = -c_1 z_1 + \dot{x}_{1r} \tag{B.13}$$

Inserting equation B.13 into B.12 gives:

$$\dot{V}_1(z_1) = z_1 z_2 - c_1 z_1^2 \tag{B.14}$$

Next the second subsystem will be defined. The control Lyapunov function will be used for the second one:

$$V(z) = V_1(z_1) + \frac{1}{2} z_2^2 \tag{B.15}$$

And its derivative:

$$\dot{V}(z) = \dot{V}_1(z_1) + z_2 \dot{z}_2 \cong z_1 z_2 - c_1 z_1^2 + z_2 [\dot{x}_{2,0} + \theta_3 \Delta u - \dot{\alpha}_1] \tag{B.16}$$

To assure stability the incremental input Δu can be selected so $\dot{V}$ becomes like equation B.17. The associated formula for u can be seen in equation B.18.

$$\dot{V} = -c_1 z_1^2 - c_2 z_2^2 \tag{B.17}$$

$$\Delta u = \frac{1}{\theta_3} [-\dot{x}_{2,0} + \dot{\alpha}_1 - z_1 - c_2 z_2] \tag{B.18}$$

B.0.3. Results

The simulation time is set to be 100 seconds with both the controllers. The P and D value for the INDI and IBS are different. It is tuned to get a stable performance. The values used for the simulation can be seen in table B.1. The input u is saturated at 250.

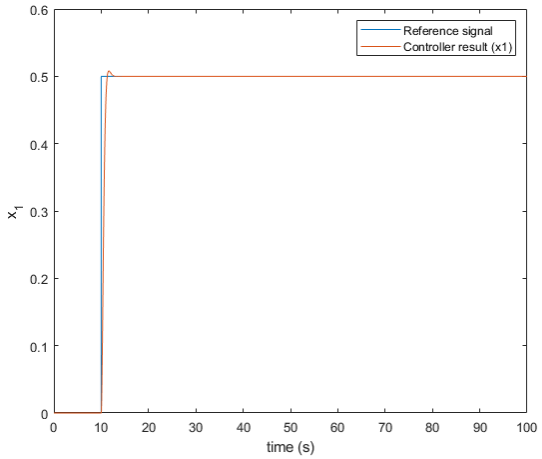
Table B.1: Values used for simulation

Symbol	Value
g	-9.81
l	1
k	5
m	10
c1	10
c2	10

For the simulation three different reference signals are generated: step, sinusoidal and a polynomial. The results of these simulations can be seen in Figure B.3 to B.8. The computational time of the simulation can be seen in Table B.2. From the table it can be seen the computational power required is less for INDI.

Table B.2: Average computational time simulation

Input	Average time IBS	Average time INDI
Step	0.133	0.123
Sinus	0.132	0.127
Polynomial	0.121	0.118



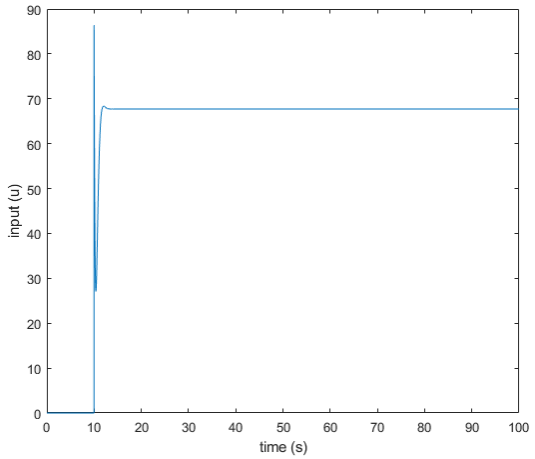
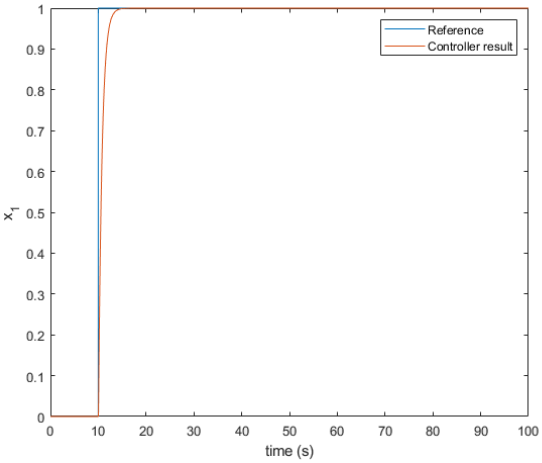


Figure B.3: INDI simulation results (step input)



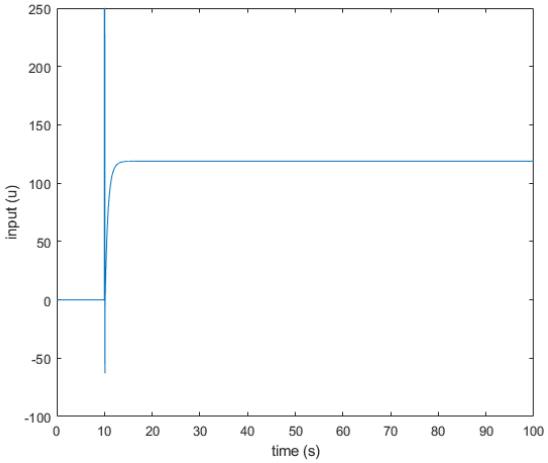
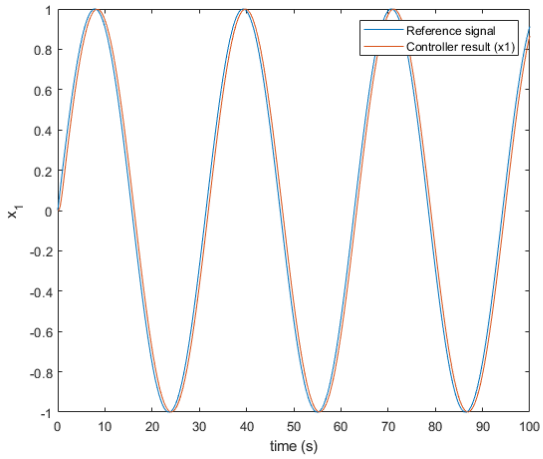


Figure B.4: IBS simulation results (step input)



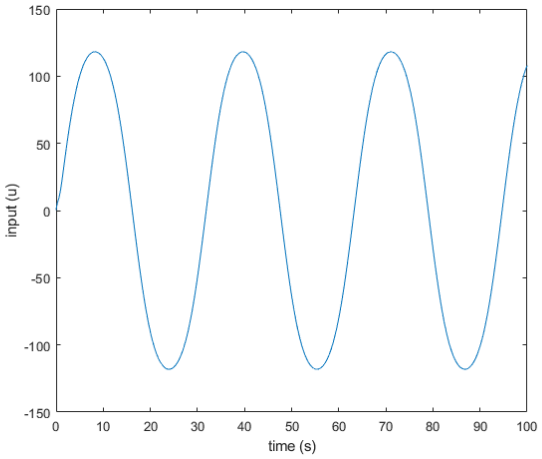


Figure B.5: INDI simulation results (sinus input)

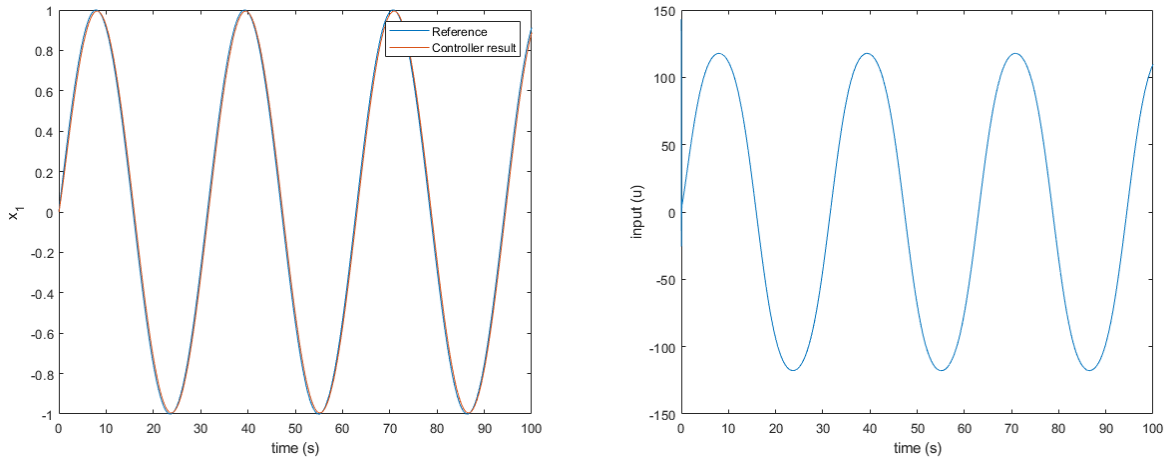


Figure B.6: IBS simulation results (sinus input)

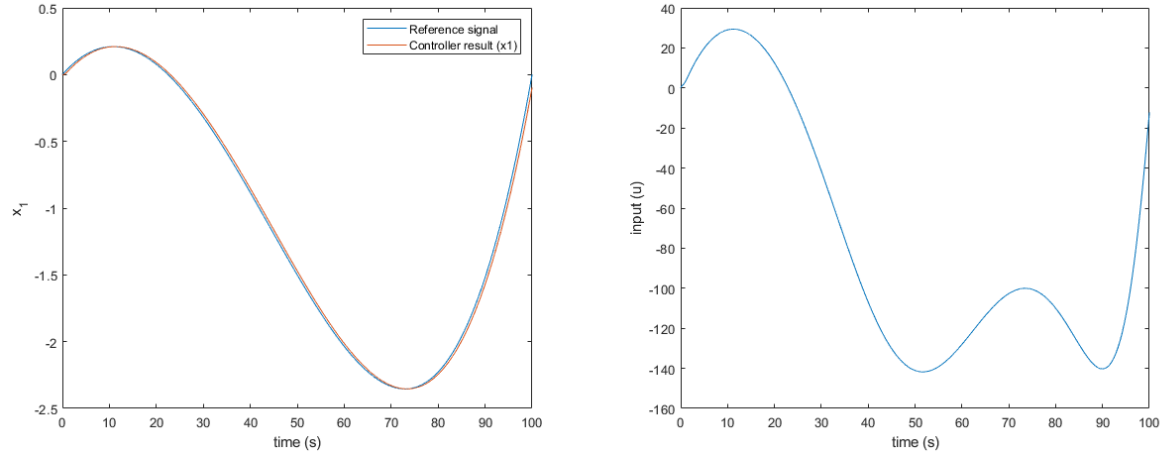


Figure B.7: INDI simulation results (polynomial input)

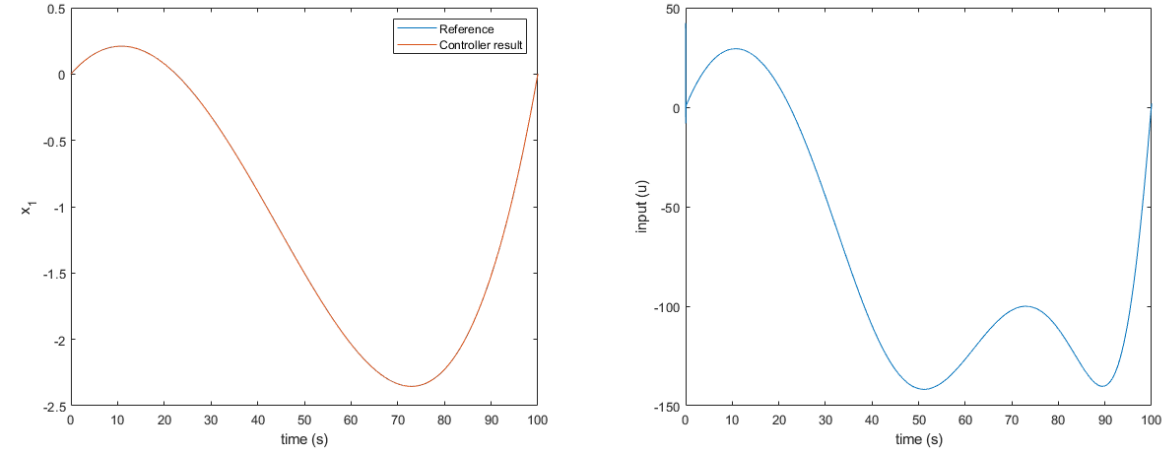


Figure B.8: IBS simulation results (polynomial input)

C

Appendix C Feedback Error Methods Comparison

For the inner loop the attitude error can be determined in several ways. The most straight forward way is to use Euler angles. When dealing with Euler angles singularities occur, especially for large errors. Within the UAV industry often is chosen to use quaternions. With this method no singularities occur and it is computational efficient. Quadcopters and tailsitters have problems with yaw movements as the response time is long. Quaternion feedback calculates the errors by determining the shortest angle towards the new attitude. The alternative is to use the tilt-twist method, where quaternions are used and the error is split up into tilt (pitch and roll) and twist (yaw). The thrust vector is the most important which must be achieved in order to keep the UAV flying. Within this chapter the quaternion method is compared to the tilt-twist by using a Matlab simulation and a flight test. For the Matlab simulation the used model is based on the Nederdrone. It is a simplified version which contains six actuators, four motors (one for each wing) and two elevons (one for each side). The used state space is shown in equation C.1 and C.2.

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{C.1}$$

Where x is the state vector and the input is given by u. The state x, input u and the A and B matrix are defined by:

$$\mathbf{x} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ p \\ q \\ r \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ \delta_{e1} \\ \delta_{e2} \end{bmatrix},$$

$$A = 0.5 \begin{bmatrix} 0 & 0 & 0 & -q_1 & -q_2 & -q_3 \\ 0 & 0 & 0 & q_0 & -q_3 & q_2 \\ 0 & 0 & 0 & q_3 & q_0 & -q_2 \\ 0 & 0 & 0 & -q_3 & q_2 & q_1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{I_{xx\delta_{e1}}} & -\frac{1}{I_{xx\delta_{e2}}} \\ -\frac{1}{I_{yyT_1}} & -\frac{1}{I_{yyT_2}} & \frac{1}{I_{yyT_3}} & \frac{1}{I_{yyT_4}} & \frac{1}{I_{yy\delta_{e1}}} & \frac{1}{I_{yy\delta_{e2}}} \\ -\frac{1}{I_{zzT_1}} & \frac{1}{I_{zzT_2}} & -\frac{1}{I_{zzT_3}} & \frac{1}{I_{zzT_4}} & 0 & 0 \end{bmatrix} \tag{C.2}$$

For this simulation a PD controller is used to control the system. The axis are defined according to Figure C.1.

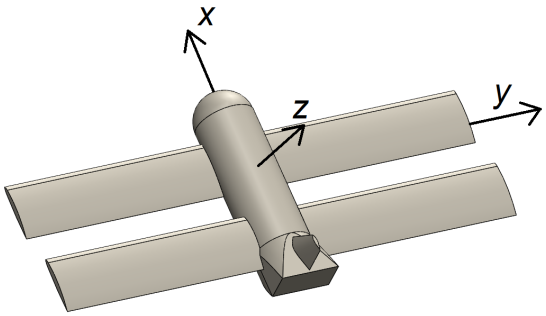


Figure C.1: Axis directions

C.0.1. Quaternion

The relationship between different quaternions can be described by:

$$\eta_{des} = \eta_{err} \otimes \eta_{cur} \tag{C.3}$$

To get the error the following equation can be used:

$$\eta_{err} = \eta_{des} \otimes \eta_{cur}^{-1} \tag{C.4}$$

$$\eta_{err} = \begin{bmatrix} q_{err0} \\ q_{err1} \\ q_{err2} \\ q_{err3} \end{bmatrix} \quad \eta_k = \begin{bmatrix} q_{r0} & q_{r1} & q_{r2} & q_{r3} \\ -q_{r1} & q_{r0} & q_{r3} & -q_{r2} \\ -q_{r2} & -q_{r3} & q_{r0} & q_{r1} \\ -q_{r3} & q_{r2} & -q_{r1} & q_{r0} \end{bmatrix} \quad \eta_{cur} = \begin{bmatrix} q_{c0} \\ q_{c1} \\ q_{c2} \\ q_{c3} \end{bmatrix} \tag{C.5}$$

The input u is calculated by using q_{err1}, q_{err2} and q_{err3} and applying a PID controller to control the angular accelerations.

C.0.2. Tilt-twist

For tilt twist the tilt error needs to be calculated first. The tilt error consists of two components. Around the y-axis and the z-axis. The rotation matrices of the current attitude and the desired attitude compared to the earth reference frame need to be determined. Rotation matrices can be generated by using equation C.6.

$$\widetilde{\mathbf{R}}_\eta = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_3 q_0) & 2(q_1 q_3 - q_2 q_0) \\ 2(q_1 q_2 - q_3 q_0) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_1 q_0) \\ 2(q_1 q_3 + q_2 q_0) & 2(q_2 q_3 - q_1 q_0) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{C.6}$$

Rotation matrices can be calculated for both the current ($\boldsymbol{R}_c$) and the desired ($\boldsymbol{R}_d$) attitudes, as shown in equation C.7.

$$\boldsymbol{R}_d = \widetilde{\mathbf{R}}_\eta(\eta_{des}) \quad \boldsymbol{R}_c = \widetilde{\mathbf{R}}_\eta(\eta_{cur}) \tag{C.7}$$

Tilt error

The first part of the tilt-twist method consists of calculating the tilt error. As was mentioned before, this error is a combination of the pitch and the roll error. The error is calculated by using the rotation matrices to align the current frame to the desired frame, as shown in equation C.8.

$$\boldsymbol{R}_d = \boldsymbol{R}_{err}\boldsymbol{R}_c \tag{C.8}$$

Equation C.8 can be rewritten as,

$$\boldsymbol{R}_{err} = \boldsymbol{R}_d\boldsymbol{R}_c^\top = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \tag{C.9}$$

The tilt error is about the y- and z-axis. The first row component of matrix $\boldsymbol{R}_{err}$ will provide the tilt error. As both the x-axis need to be aligned. In equation C.10 an example is given of the rotation matrices when there

is a roll angle of 10 degrees, in case of a pitch error of 10 degrees the matrices will look like equation C.11. A visualisation is given in figure C.2.

$$\boldsymbol{R}_d = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad \boldsymbol{R}_c = \begin{bmatrix} -0.17 & 0 & 0.98 \\ 0.98 & 0 & 0.17 \\ 0 & -1 & 0 \end{bmatrix} \quad \boldsymbol{R}_{err} = \begin{bmatrix} 0.98 & -0.17 & 0 \\ 0.17 & 0.98 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{C.10}$$

$$\boldsymbol{R}_d = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad \boldsymbol{R}_c = \begin{bmatrix} 0 & -0.17 & 0.98 \\ 1 & 0 & 0 \\ 0 & -0.98 & -0.17 \end{bmatrix} \quad \boldsymbol{R}_{err} = \begin{bmatrix} 0.98 & 0 & -0.17 \\ 0 & 1 & 0 \\ 0.17 & 0 & 0.98 \end{bmatrix} \tag{C.11}$$

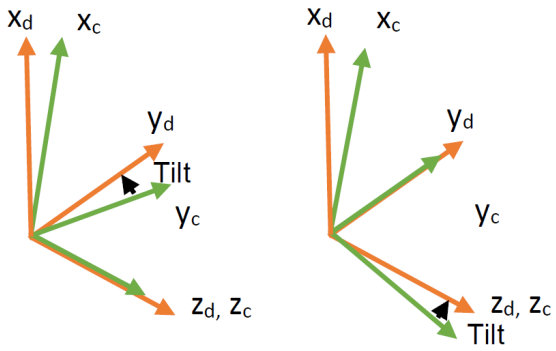


Figure C.2: Axis definition in case there is pitch or roll error of 10 degrees.

The y component of the tilt error is given by

$$Tilt\ error_1 \triangleq \varepsilon_y = -atan2(r_{1,3}, r_{1,1}) \tag{C.12}$$

Where $atan2$ is the inverse tangent. The negative sign ensures that the rotation is in the desired direction. The z component is given by equation C.13.

$$Tilt\ error_2 \triangleq \varepsilon_z = atan2(r_{1,2}, r_{1,1}) \tag{C.13}$$

Twist error
In the second part of the tilt-twist method the twist error is calculated, which is the given error around the body fixed x-axis. To calculate the error, an intermediate coordinate frame needs to be found which aligns the current x-axis with the desired x-axis, which will still include the twist error. This is done by using the rotation matrices,

$$\boldsymbol{R}_d = \begin{bmatrix} \boldsymbol{r}_{d1} \\ \boldsymbol{r}_{d2} \\ \boldsymbol{r}_{d3} \end{bmatrix} \quad \boldsymbol{R}_c = \begin{bmatrix} \boldsymbol{r}_{c1} \\ \boldsymbol{r}_{c2} \\ \boldsymbol{r}_{c3} \end{bmatrix} \tag{C.14}$$

The total tilt error will be used to align the x-axes. The complete tilt error is calculated by using both $\boldsymbol{R}_c$ and $\boldsymbol{R}_d$, as shown in equation C.15.

$$Tilt\ error \triangleq \varepsilon_{tilt} = \cos^{-1}(\boldsymbol{r}_{d1}^\top \cdot \boldsymbol{r}_{c1}^\top) \tag{C.15}$$

In equation C.16 the unit length axis is defined.

$$k = \frac{\boldsymbol{r}_{c1}^\top \times \boldsymbol{r}_{d1}^\top}{|\boldsymbol{r}_{c1}^\top \times \boldsymbol{r}_{d1}^\top|} \tag{C.16}$$

As the rotation needs to happen in the vehicle frame the unit vector k needs to be in the vehicle frame, which is done by

$$\boldsymbol{v}^b = \boldsymbol{R}_c k = \begin{bmatrix} v_x^b \\ v_y^b \\ v_z^b \end{bmatrix} \tag{C.17}$$

and

$$\boldsymbol{v} = \begin{bmatrix} 0 & -v_z^b & v_y^b \\ v_z^b & 0 & -v_x^b \\ -v_y^b & v_x^b & 0 \end{bmatrix} \tag{C.18}$$

Next a rotation matrix needs to be defined which rotate a vector around a vector. The angle of this vector is also included. This is accomplished by using the Rodrigues' rotation formula [45], where the angle is ε_{tilt} and the vector is $\boldsymbol{v}^b$. The rotation matrix is given by equation C.19.

$$\boldsymbol{R}_v = \begin{cases} \boldsymbol{I}, & \varepsilon_{tilt} = 0 \\ \boldsymbol{I} - \boldsymbol{v}\sin(\varepsilon_{tilt}) + \boldsymbol{v}^2[1-\cos(\varepsilon_{tilt})], & \varepsilon_{tilt} \neq 0 \end{cases} \tag{C.19}$$

The attitude of the UAV can be compensated with the tilt error, which can be found by using $\boldsymbol{R}_v$. By multiplying $\boldsymbol{R}_v$ with the rotation matrix of the current attitude of the UAV ($\boldsymbol{R}_c$) a new rotation matrix $\boldsymbol{R}_p$ is found, as shown in equation C.20.

$$\boldsymbol{R}_p = \boldsymbol{R}_v^\top \boldsymbol{R}_c \tag{C.20}$$

Where

$$\boldsymbol{R}_p = \begin{bmatrix} \boldsymbol{r}_{p1} \\ \boldsymbol{r}_{p2} \\ \boldsymbol{r}_{p3} \end{bmatrix} \tag{C.21}$$

The absolute twist error can be found using the x components of $\boldsymbol{R}_p$ and $\boldsymbol{R}_d$, see equation C.22.

$$\varepsilon_{twist} = \cos^{-1}(\boldsymbol{r}_{p3}^\top \cdot \boldsymbol{r}_{d3}^\top) \tag{C.22}$$

The sign of the twist error needs to be determined, as equation C.22 gives the absolute value of the twist error. By using the y component of $\boldsymbol{R}_p$ this can be achieved, see equation C.23.

$$\varepsilon_{sign} = \cos^{-1}(\boldsymbol{r}_{p2}^\top \cdot \boldsymbol{r}_{d3}^\top) \tag{C.23}$$

When this value is above 90°, or $\frac{\pi}{2}$ radials, the sign of the twist error becomes negative. The final twist error can be seen in equation C.24.

$$Twist\ error \triangleq \varepsilon_z = \begin{cases} -\varepsilon_{twist}, & \varepsilon_{sign} \leq \frac{\pi}{2} \\ \varepsilon_{twist}, & \varepsilon_{sign} > \frac{\pi}{2} \end{cases} \tag{C.24}$$

Together, the calculated errors form the total feedback error, given by:

$$Feedback\ error \triangleq \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{bmatrix} \tag{C.25}$$

For a PD controller, the actuator deflections of the ailerons, elevator and rudder are calculated with equation C.26, C.27 and C.28.

$$\delta_a = k_p\varepsilon_x - k_d\dot{\varepsilon}_x \tag{C.26}$$

$$\delta_e = k_p\varepsilon_y - k_d\dot{\varepsilon}_y \tag{C.27}$$

$$\delta_r = k_p\varepsilon_z - k_d\dot{\varepsilon}_z \tag{C.28}$$

C.0.3. Simulation

Both the feedback error methods are simulated in Matlab by using the above mentioned state space. The used values of the system is shown in table C.1. The high inertia values for the x-axis represent the saturation of the actuators. As both methods give different errors a PD tuning is performed for both the situations (quaternion and tilt-twist).

Table C.1: Used values for the Matlab simulations

	Value
$I_{xx\delta_{e1}}$	6
$I_{xx\delta_{e2}}$	6

	Value
I_{yyT_1}	0.06
I_{yyT_2}	0.06
I_{yyT_3}	0.06
I_{yyT_4}	0.06
$I_{yy\delta_{e1}}$	0.06
$I_{yy\delta_{e1}}$	0.06

	Value
I_{zzT_1}	0.06
I_{zzT_2}	0.06
I_{zzT_3}	0.06
I_{zzT_4}	0.06

Results

Figure C.3 until C.9 show the behaviour of the system. The quaternion feedback decreases the tilt error slowly (phi and theta). Tilt-twist is able to remove the tilt error within 0.6 seconds, where the quaternion feedback needs more then a second, see figure C.3 and C.5. Furthermore, tilt-twist uses all the axis to define the error, the quaternion feedback uses only two (figure C.6 and C.7).

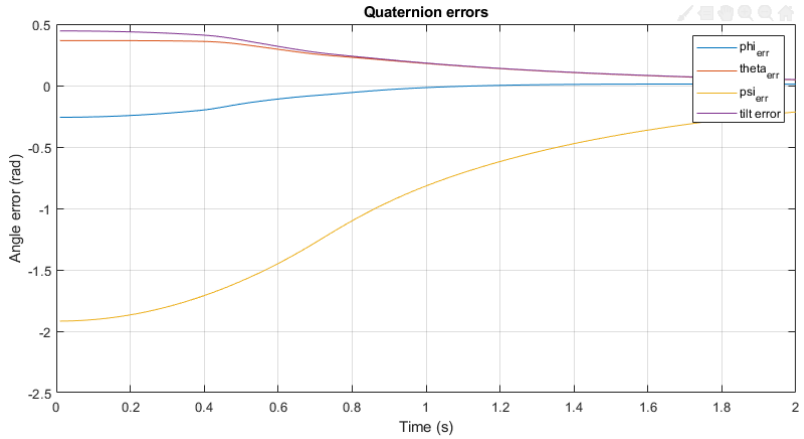


Figure C.3: Simulation results for the quaternion feedback. Attitude representation in Euler angles.

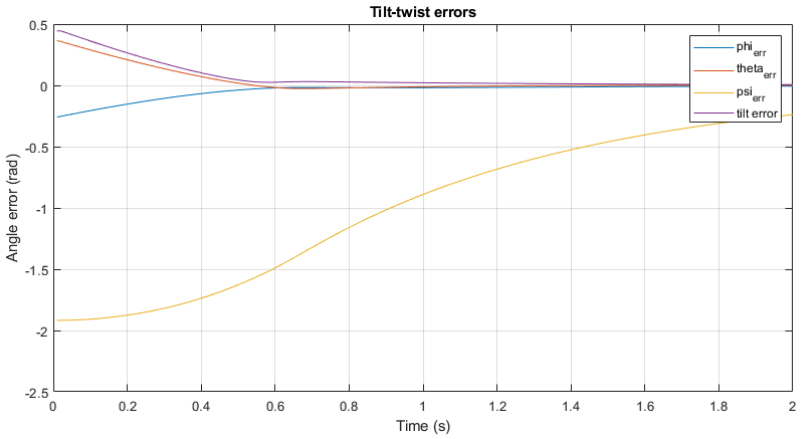


Figure C.4: Simulation results for the tilt-twist feedback. Attitude representation in Euler angles.

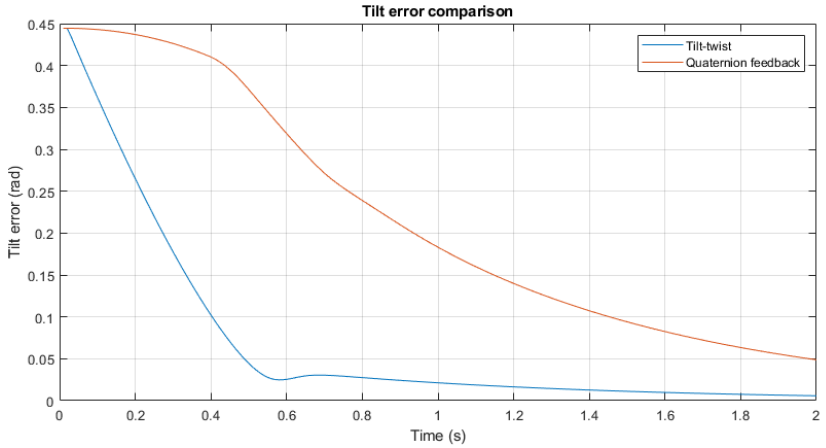


Figure C.5: Tilt error comparison between quaternion feedback and tilt-twist.

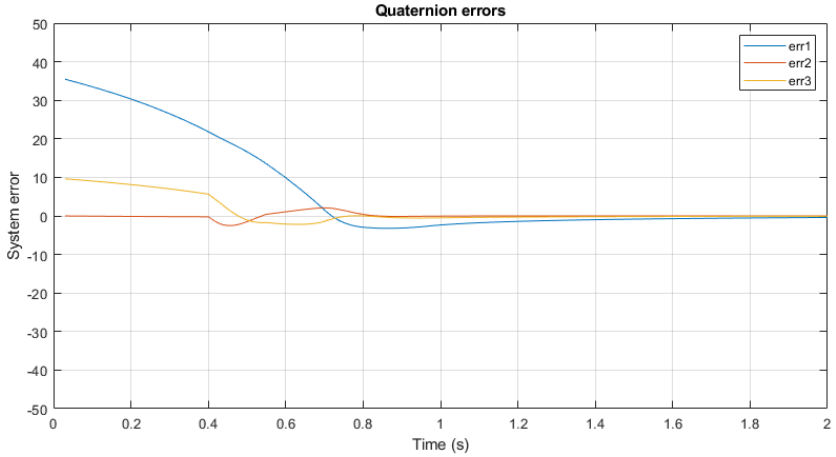


Figure C.6: Simulation results for the quaternion feedback. The errors from the calculation for the quaternion feedback.

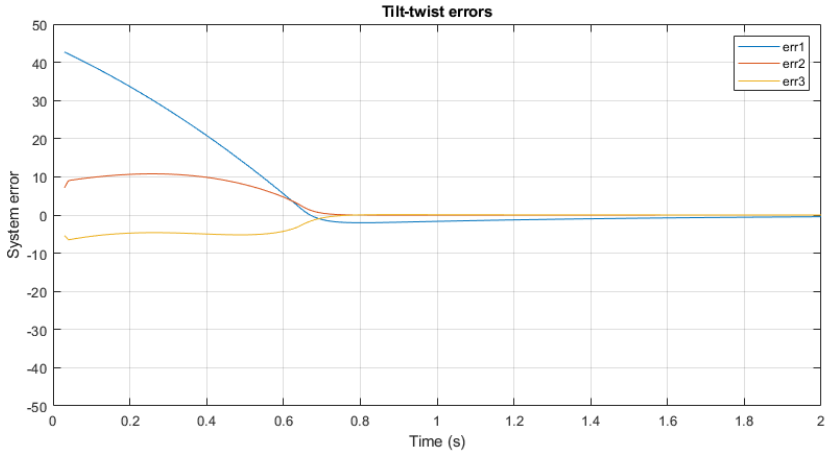


Figure C.7: Simulation results for the tilt-twist feedback. The errors from the calculation for the tilt-twist feedback.

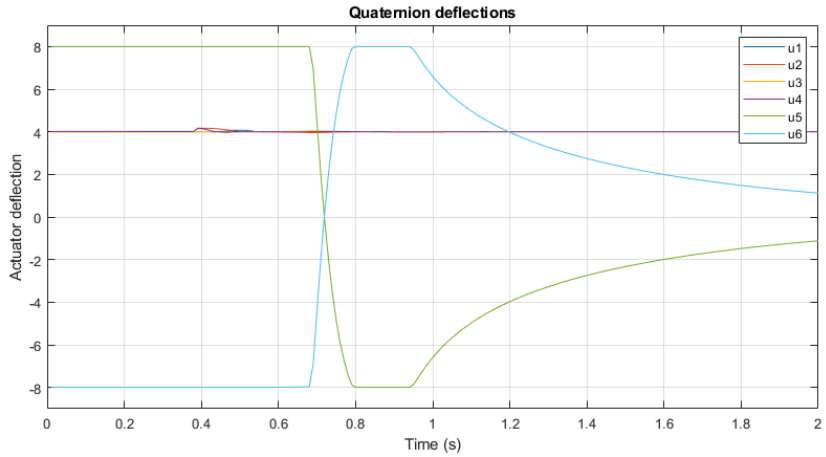


Figure C.8: Simulation results for the quaternion feedback. The actuator deflections.

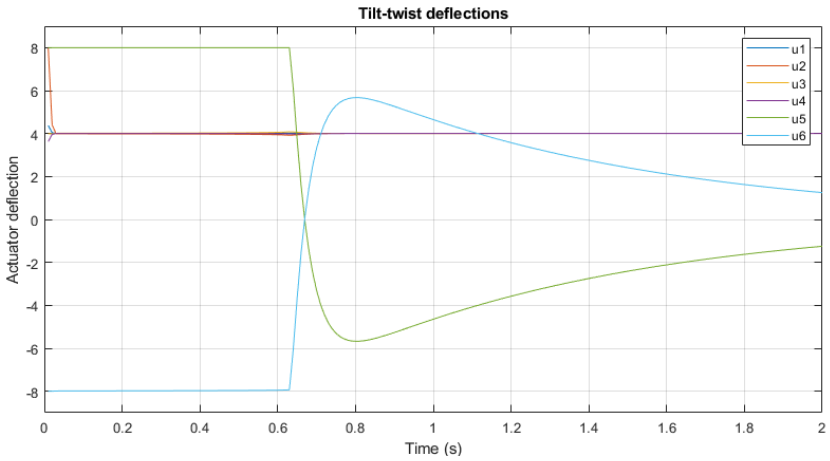


Figure C.9: Simulation results for the tilt-twist feedback. The actuator deflections.

C.0.4. Quadcopter flight tests

Both the feedback methods were applied into a Bebop, see figure C.10. The flight tests were performed at the Cyberzoo (indoor location) at the TU Delft. The yaw was saturated. During the test flight the drone was controlled manually. The pilot controlled the position of the quadcopter. The pilot controlled the Bebop aggressively, see figure C.11. In figure C.12 the height of the Bebop is shown. When using the Quaternion feedback the drone crashed two times. Furthermore it had unpredictable height drops. The tilt-twist method did not crash and was able to perform an safe flight. In figure C.13 the tilt error is shown. The tilt error was easily reduced when using tilt-twist, during quaternion the tilt error was not constantly removed and kept present. From this test a clear difference is seen in the behaviour of the quadcopter, where the tilt-twist performed much better and there were no crashes.

Figure C.10: The Bebop used for the test flight.

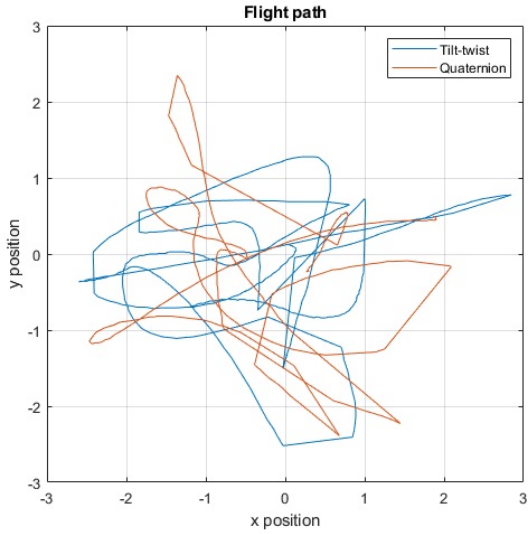


Figure C.11: Flight path during the tests

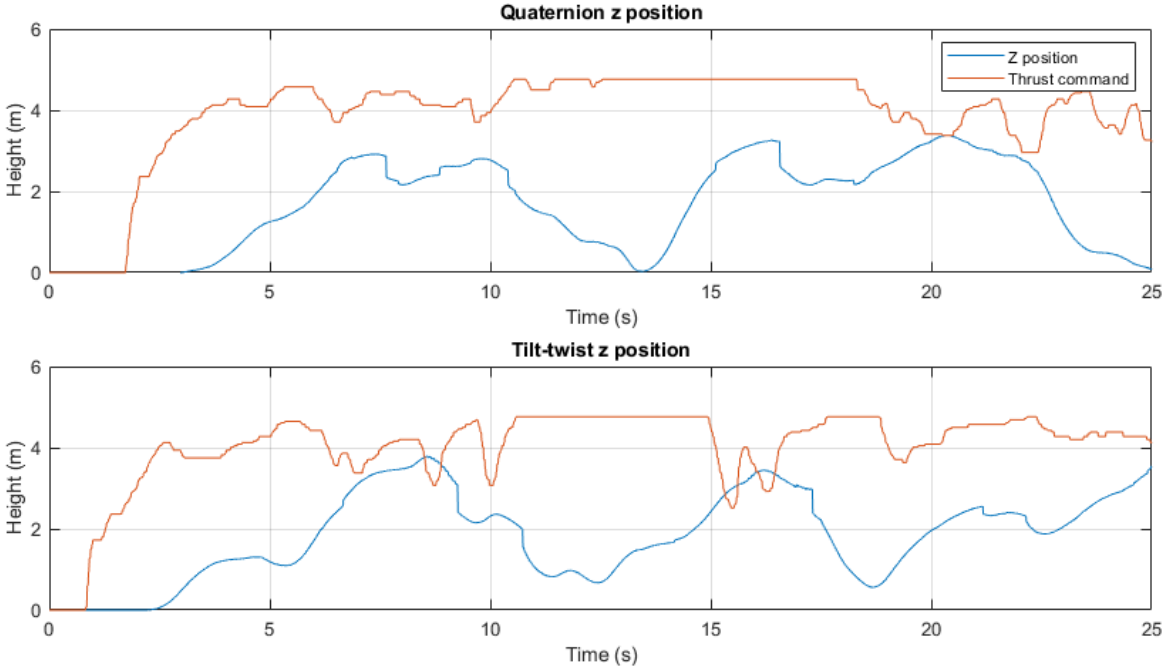


Figure C.12: The z position of the Bebop during the test flight.

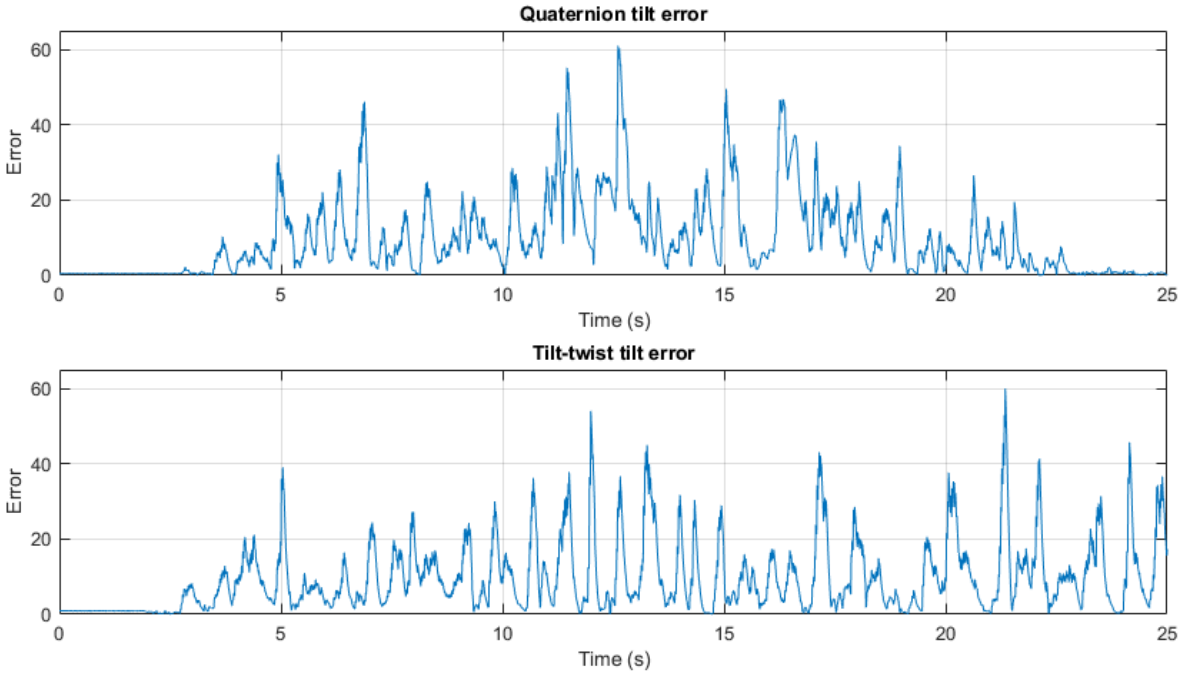


Figure C.13: The tilt error during the test flight.

D

Appendix D Flight tests results

In this section all the plots of the performed flight tests are presented. The first test was conducted at airport Valkenburg on the 22^{th} of July 2020. The second test was conducted at airport Valkenburg on the 19^{th} of August 2020.

D.1. Test 1 Results

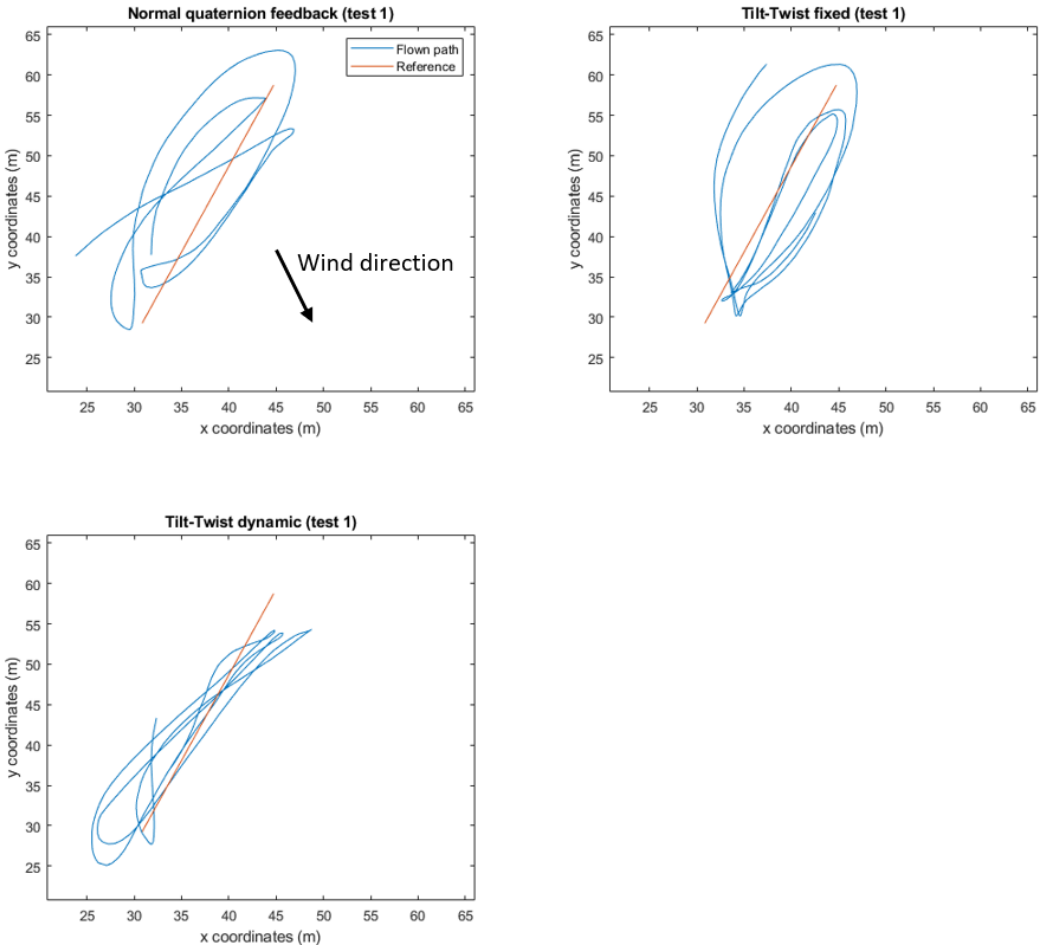


Figure D.1: Flight paths during test 1

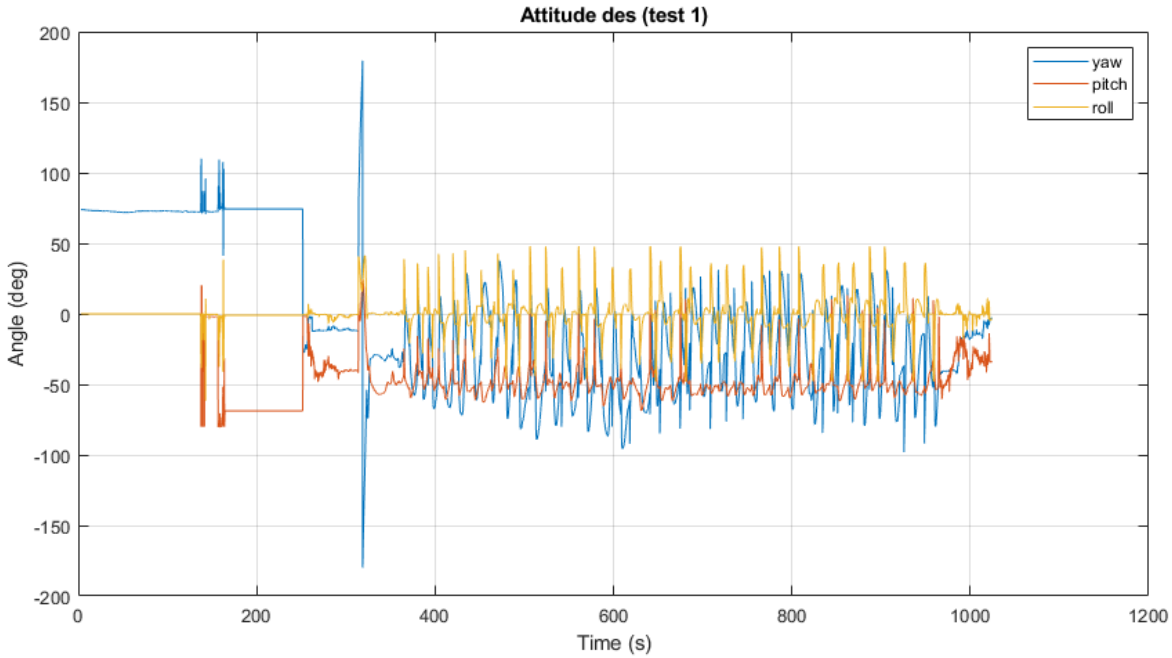


Figure D.2: The desired attitude during the test.

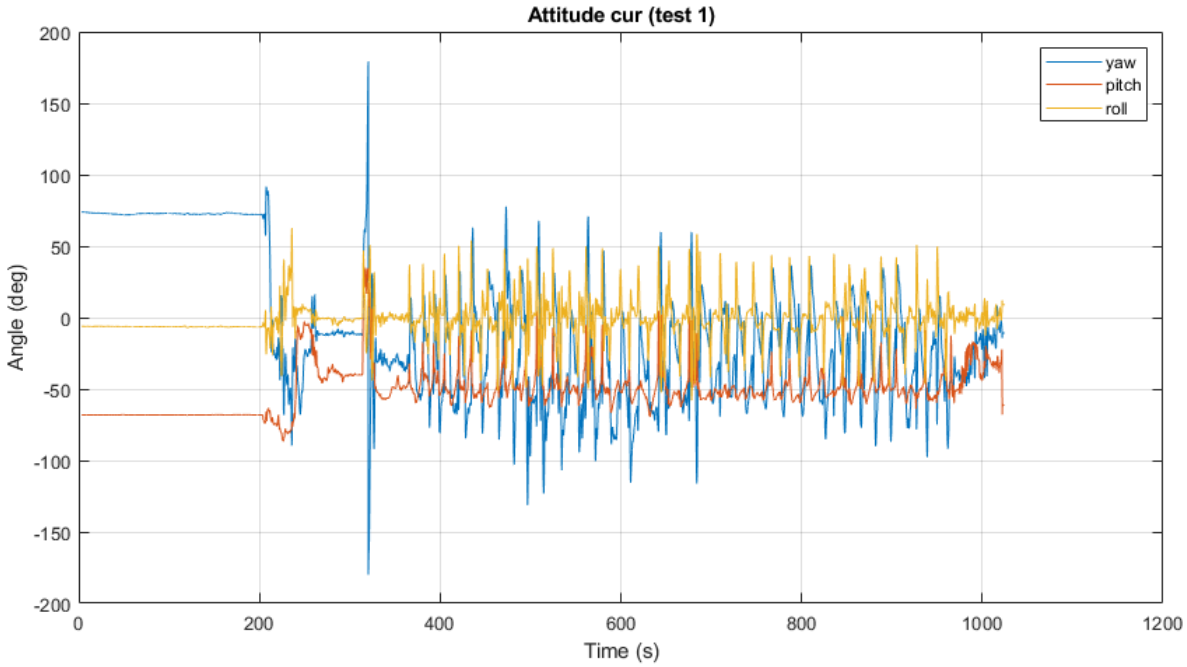


Figure D.3: The attitude of the Nederdrone during the test.

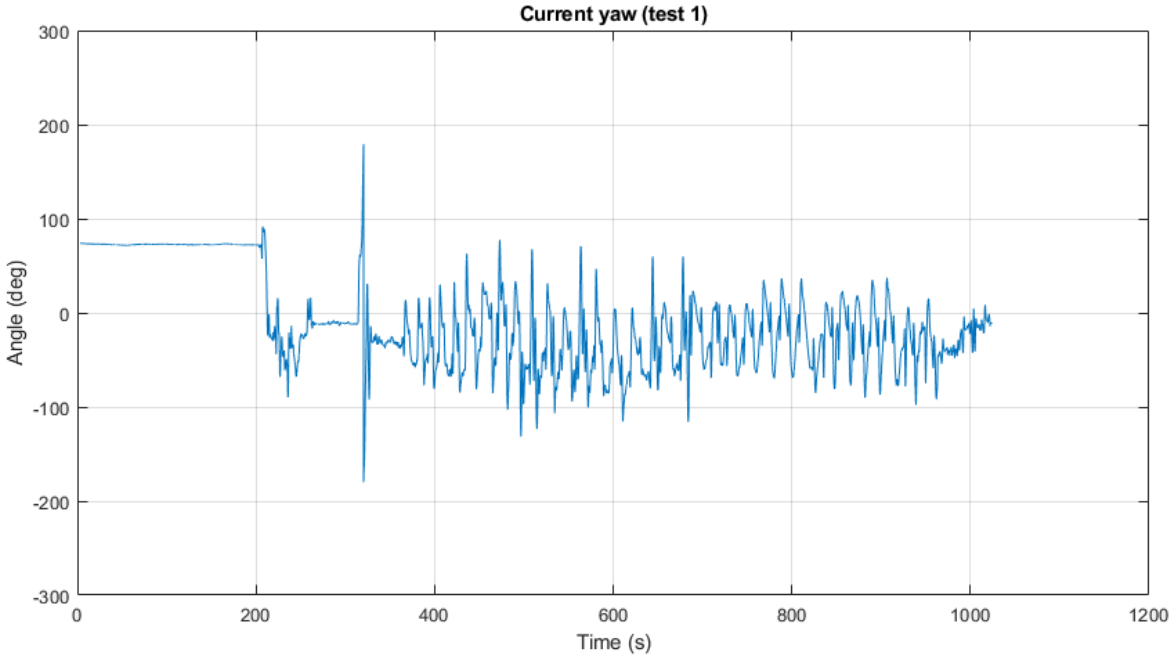


Figure D.4: Yaw angle of the Nederdrone during the test flight.

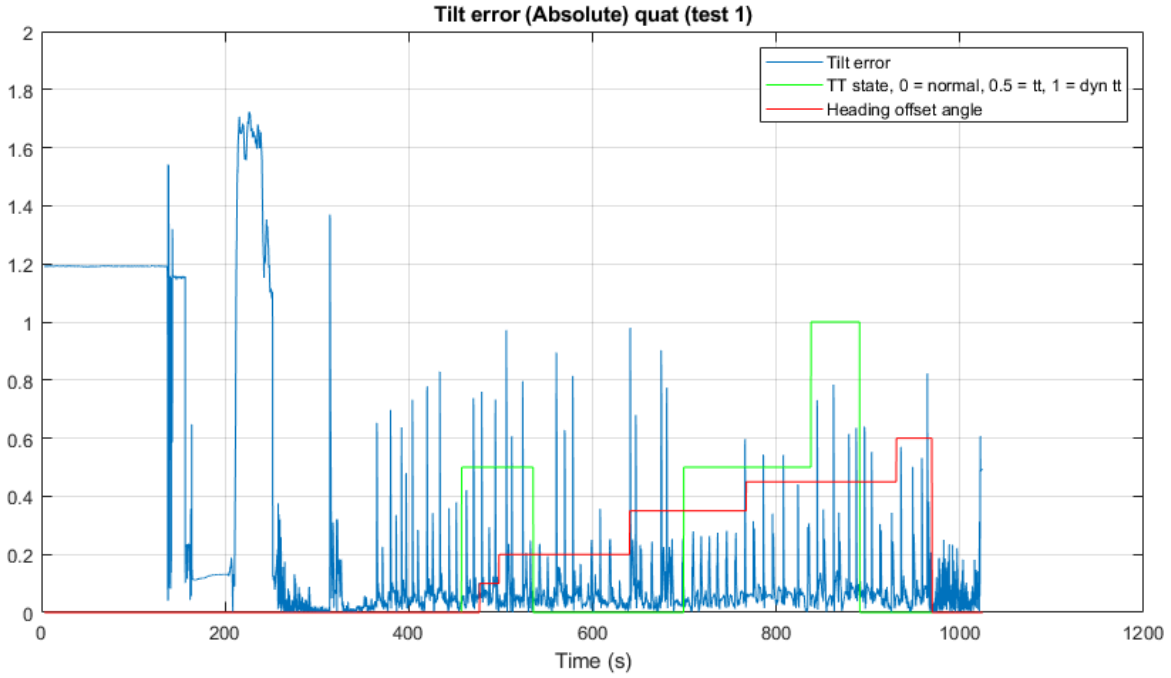


Figure D.5: Plot of the tilt error, included is the heading offset angle and which error feedback method is used, 0 is the quaternion feedback, 0.5 is the tilt-twist and 1 is the dynamic tilt-twist.

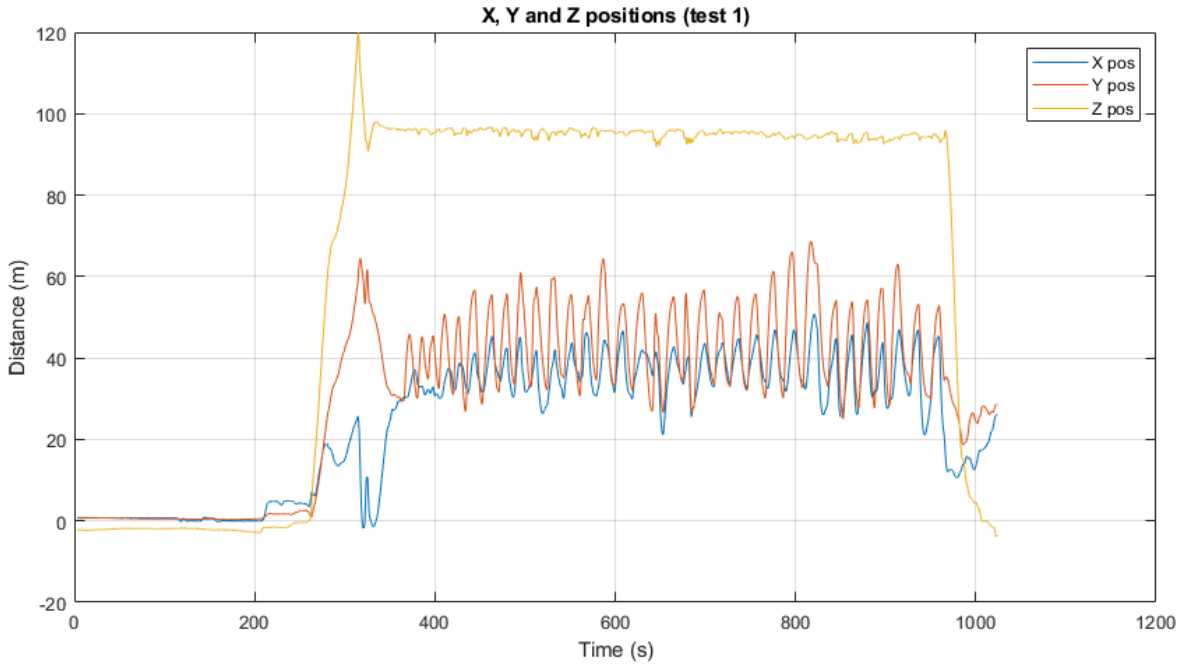


Figure D.6: Xyz position of the Nederdrone

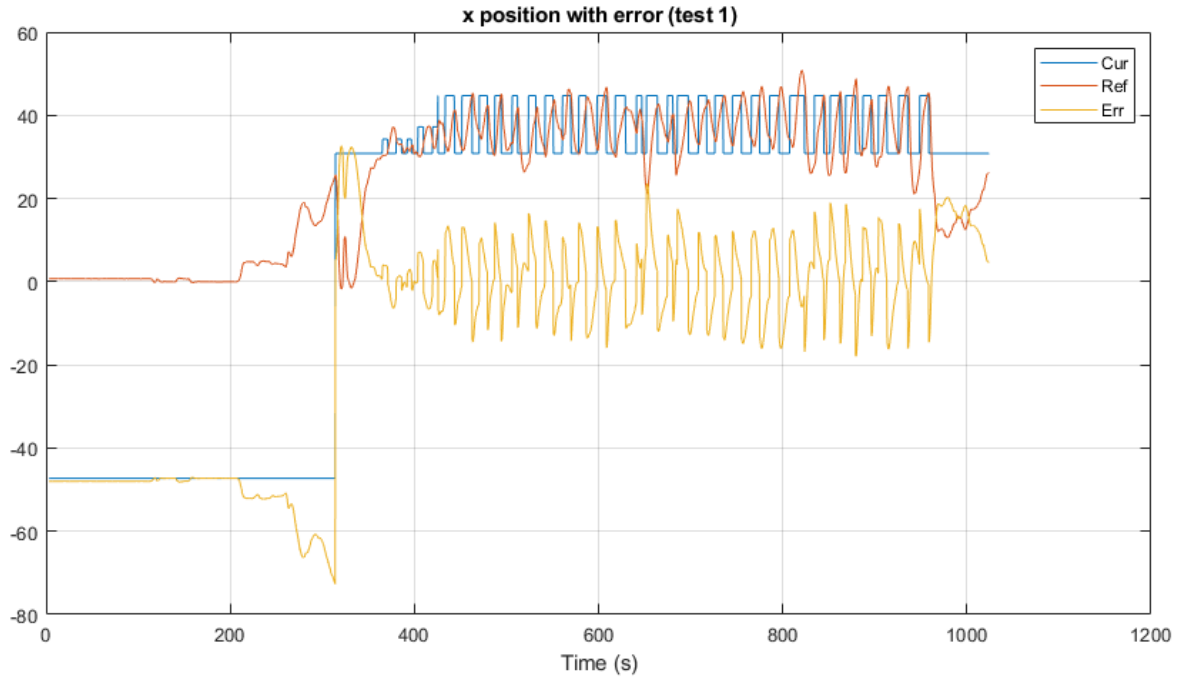


Figure D.7: The x position of the Nederdrone against the desired x position, the error is also included.

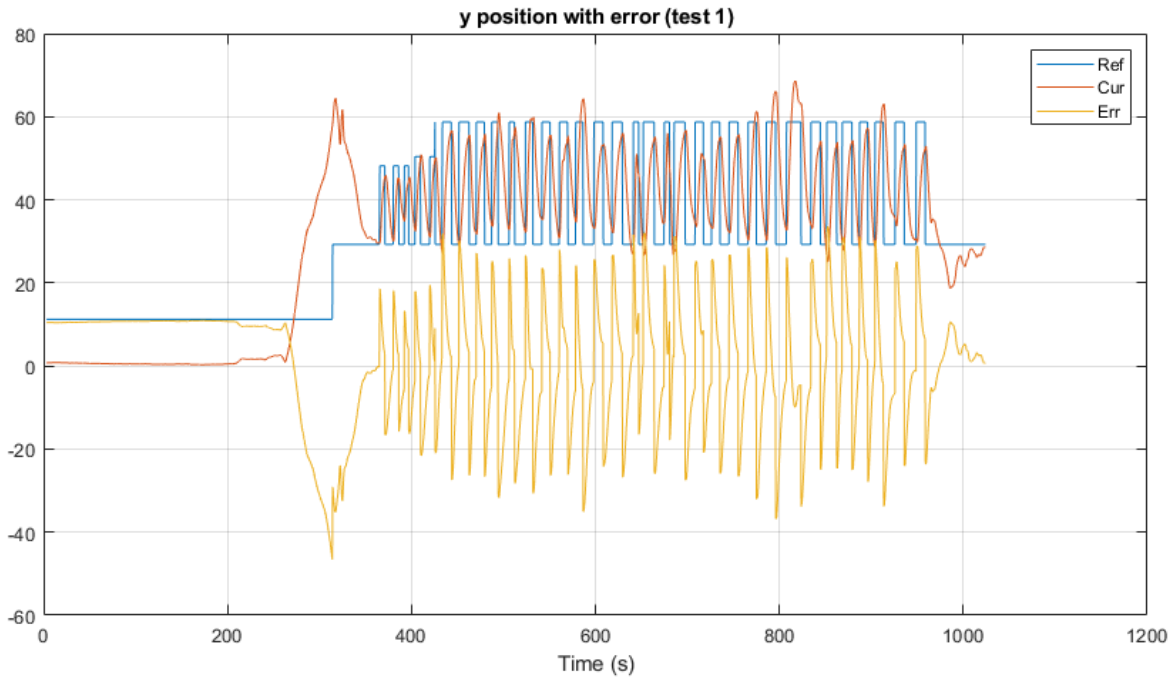


Figure D.8: The y position of the Nederdrone against the desired x position, the error is also included.

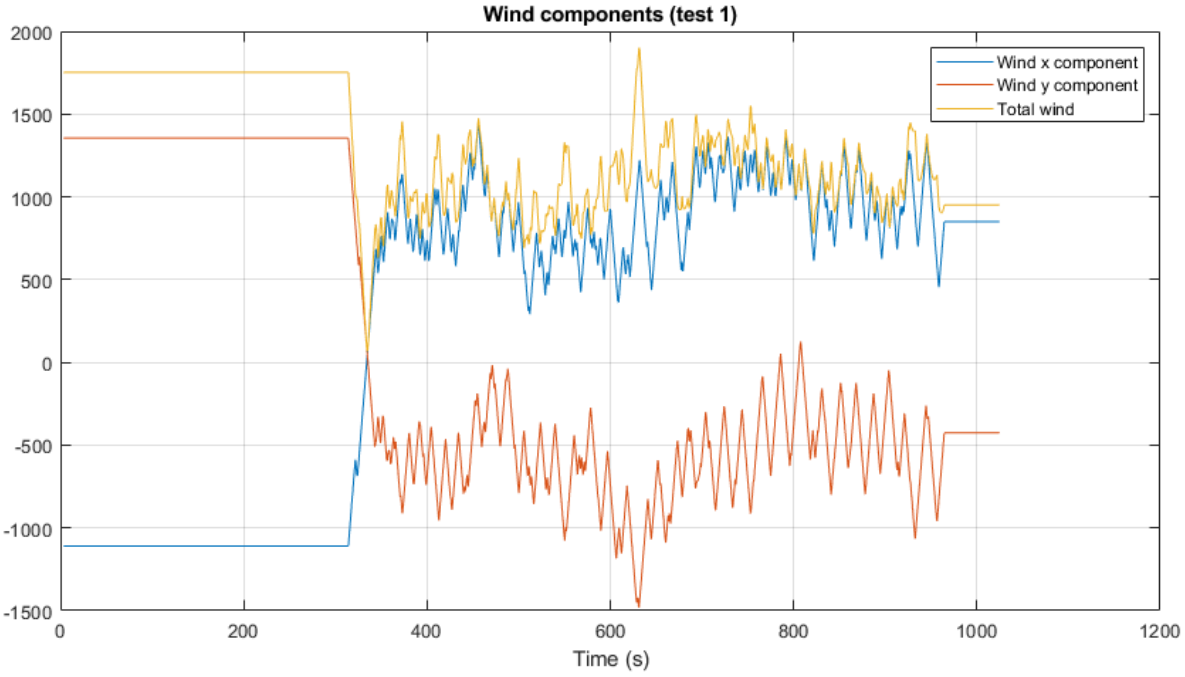


Figure D.9: The wind components, x is the east direction and y in the north direction.

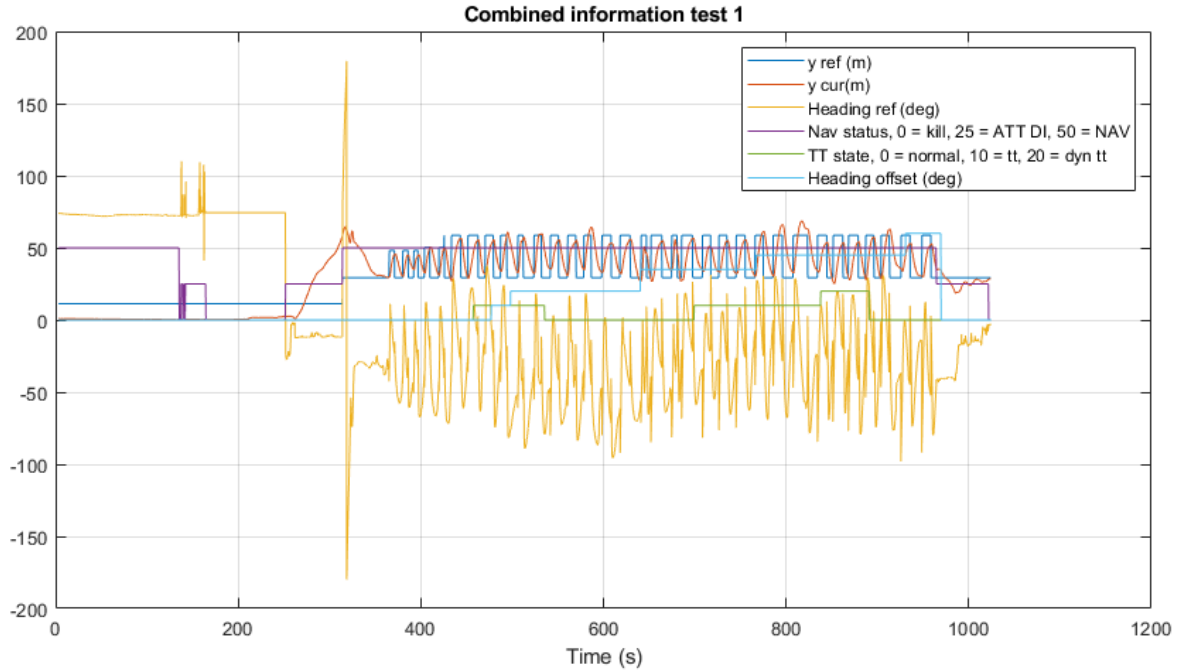


Figure D.10: Graph showing the y position of the Nederdrone and the desired y position, the desired heading, the navigation status (manual control or autopilot on), which feedback error is used and the heading offset.

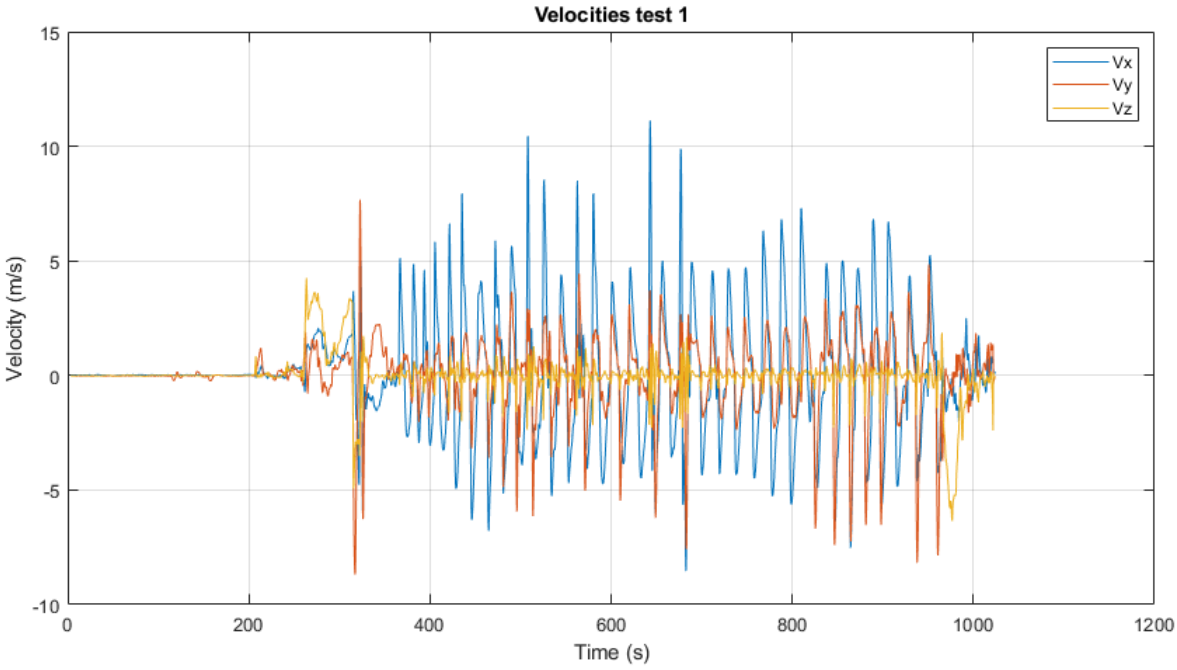


Figure D.11: The velocities of the Nederdrone

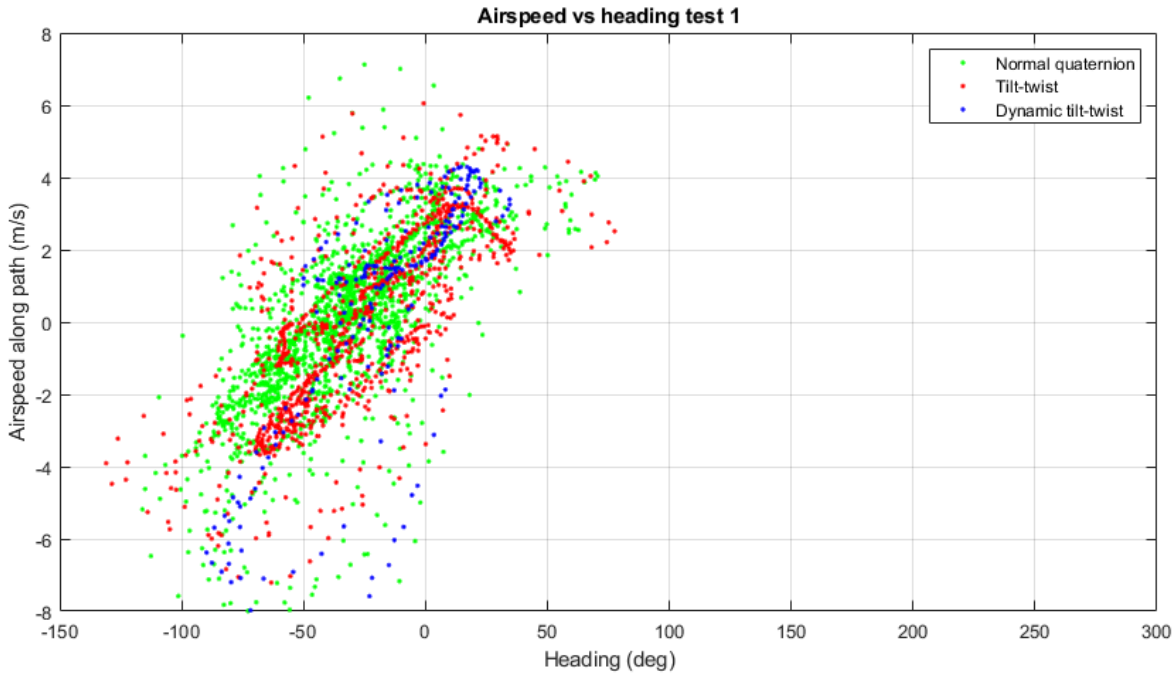


Figure D.12: The graphs show the heading of the Nederdrone against the airspeed along the desired flight path.

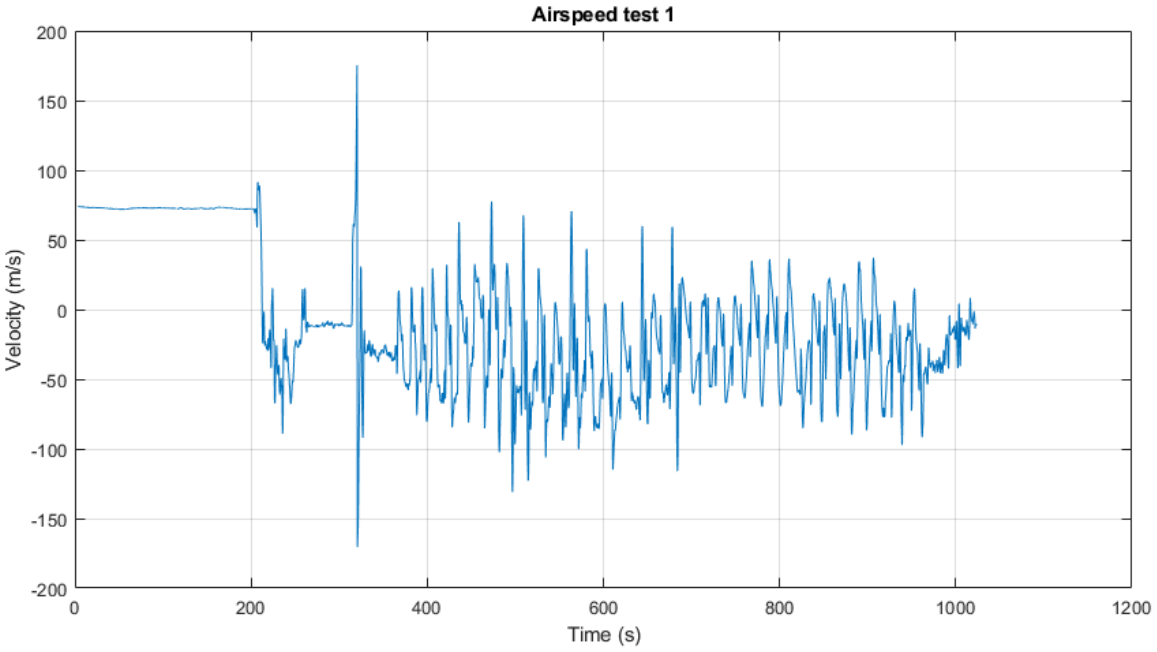


Figure D.13: The airspeed of the Nederdrone.

D.1.1. Test 1 zoomed

This section includes information of a zoomed section of the flight (30 seconds).

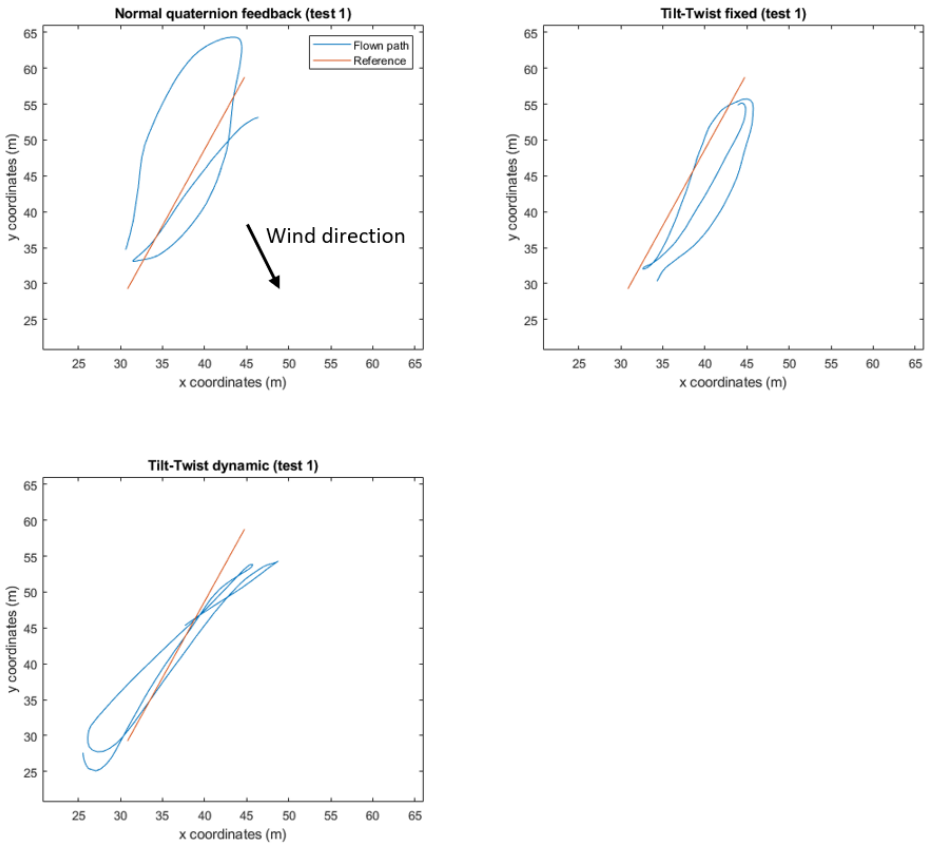


Figure D.14: Flight paths during test 1 (30 seconds)

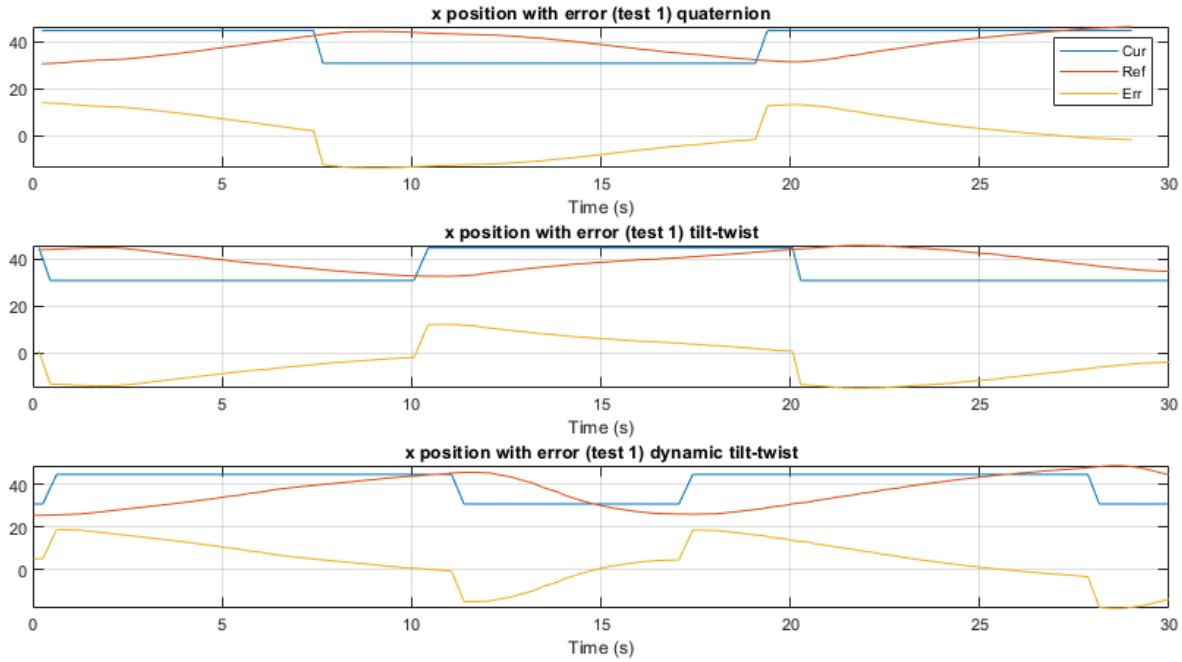


Figure D.15: The x required and the real x positions of the Nederdrone.

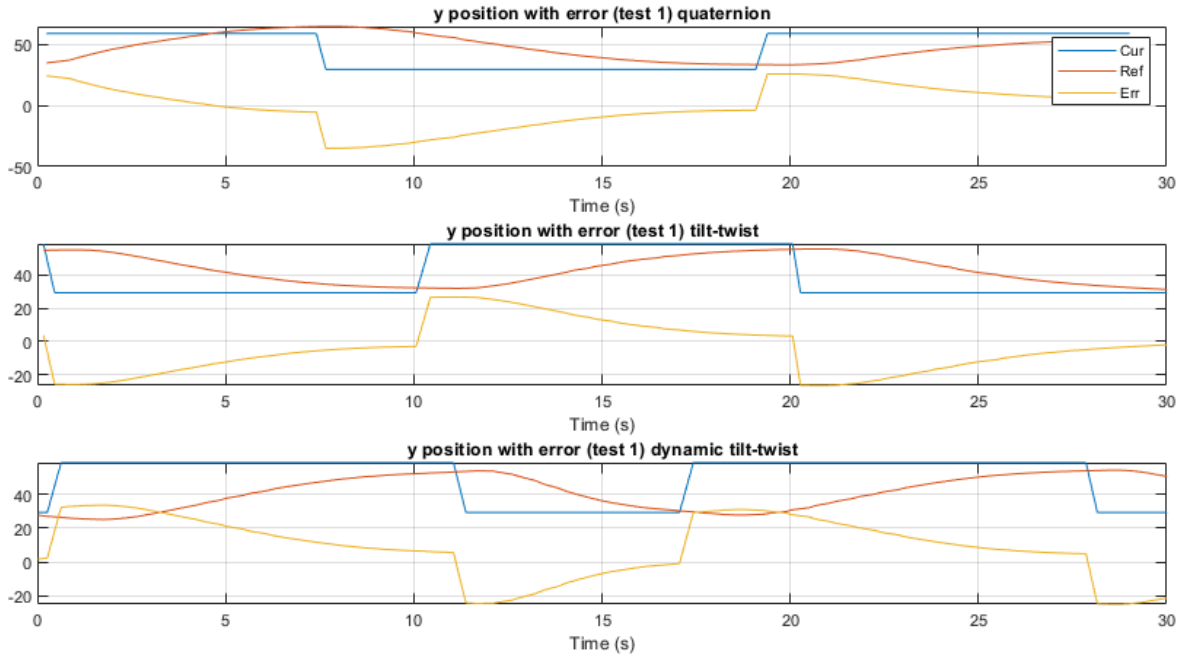


Figure D.16: The x required and the real y positions of the Nederdrone.

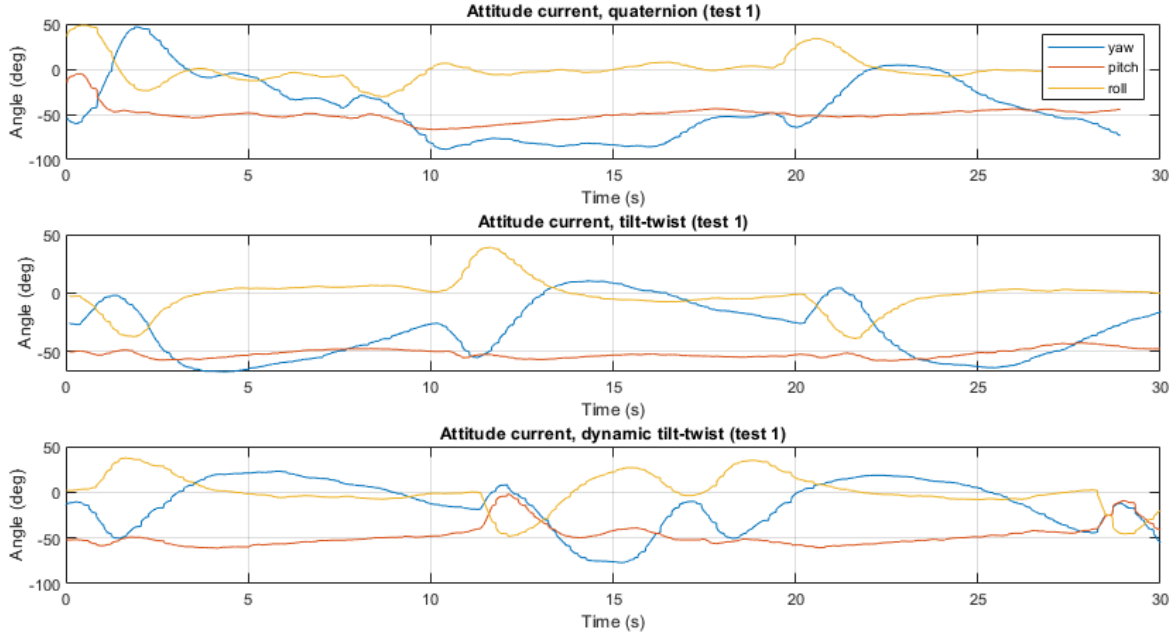


Figure D.17: The real attitudes of the Nederdrone.

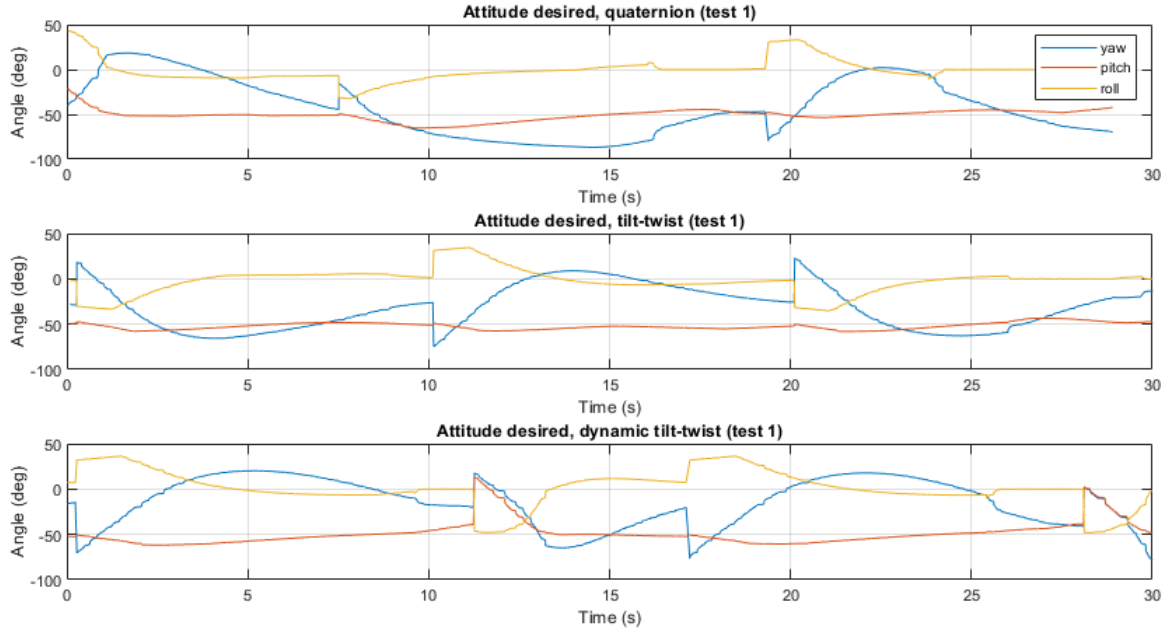


Figure D.18: The required attitudes of the Nederdrone.

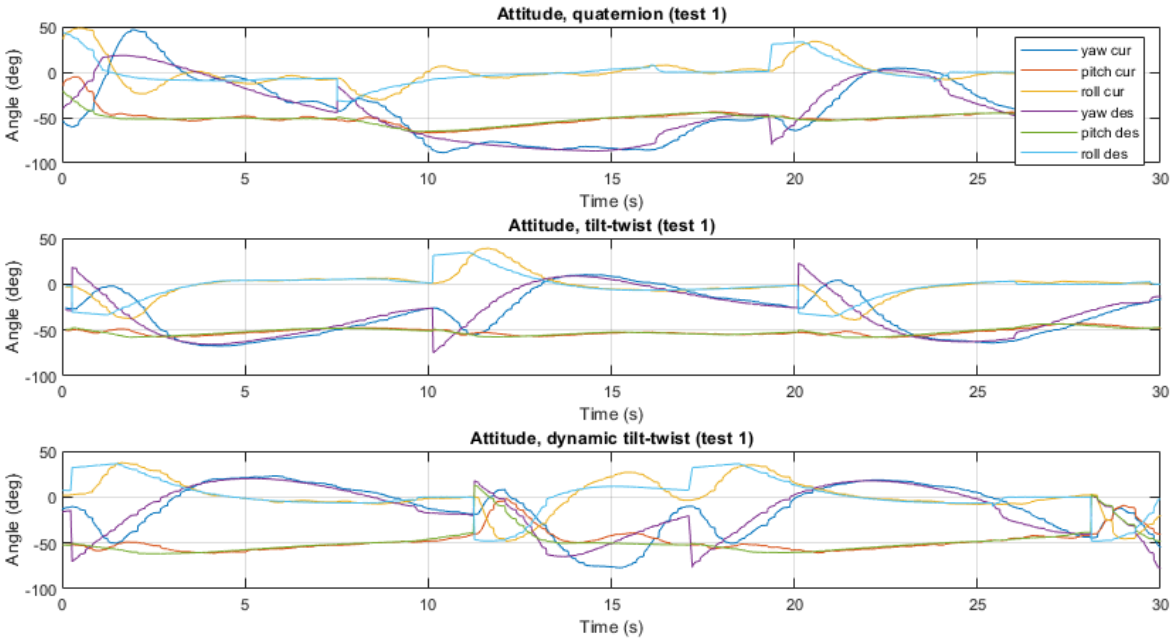


Figure D.19: The required and real attitudes of the Nederdrone.

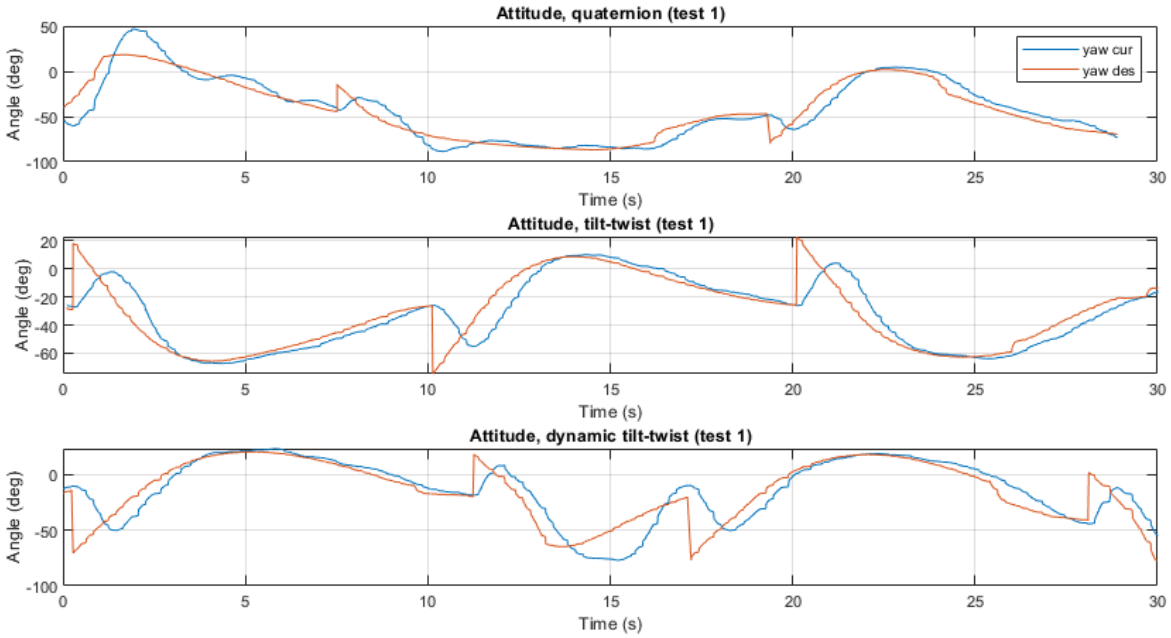


Figure D.20: The required yaw compared to the real yaw.

D.2. Test 2 Results

The second flight test did not show much improvement. The quaternion feedback method already had good performance. There are several explanations for this, the wind direction was more parallel to the desired flight path. If the wind is perpendicular to the flight path the control effectiveness decreases. The heading change during the first test was much higher, the difference between the maximum and minimum values varied between 160 to 200 degrees for the first test and the second flight test varied between 75 to 100 degrees. If the difference in yaw is small the control is easier. The velocity was also different, during test one the distance between the points was bigger and resulted in larger velocities, the maximum velocities varied between 5 and 10 m/s. During test two the maximum velocities varied between 3 and 5 m/s. Flight test two was not included in the paper, as there were no major problems during flight test 2 for the quaternion. So no major improvements were achieved.

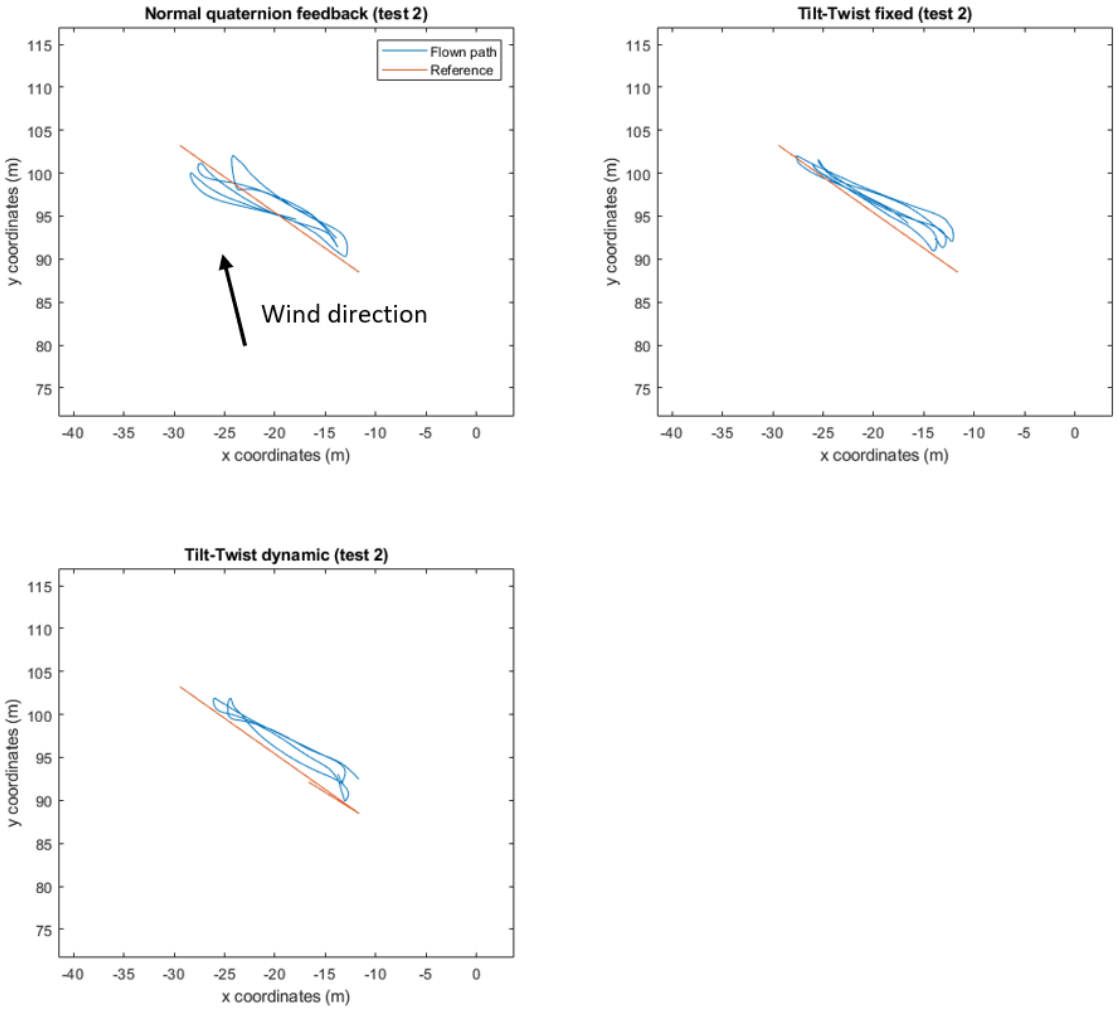


Figure D.21: Flight paths during test 2

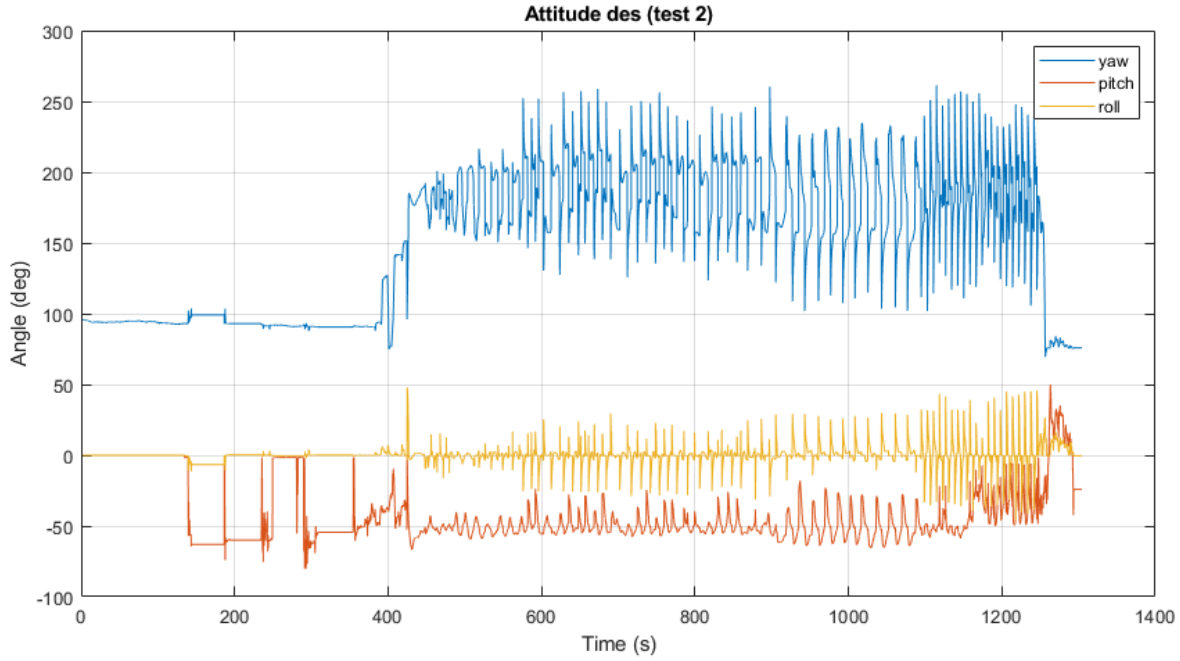


Figure D.22: The desired attitude during the test.

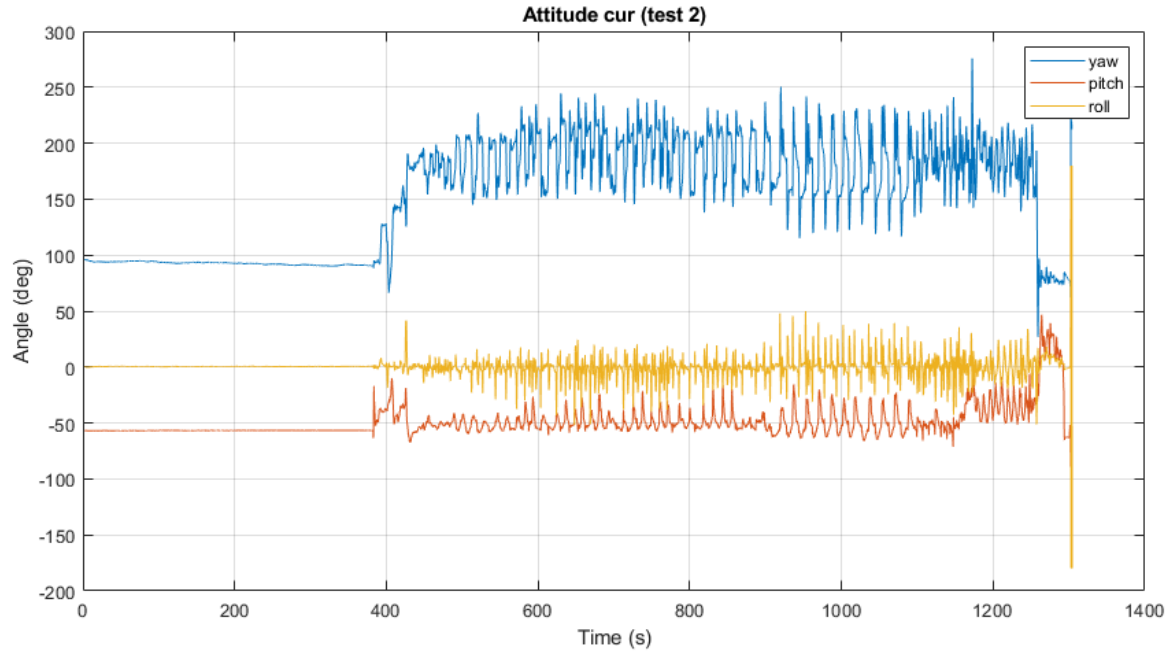


Figure D.23: The attitude of the Nederdrone during the test.

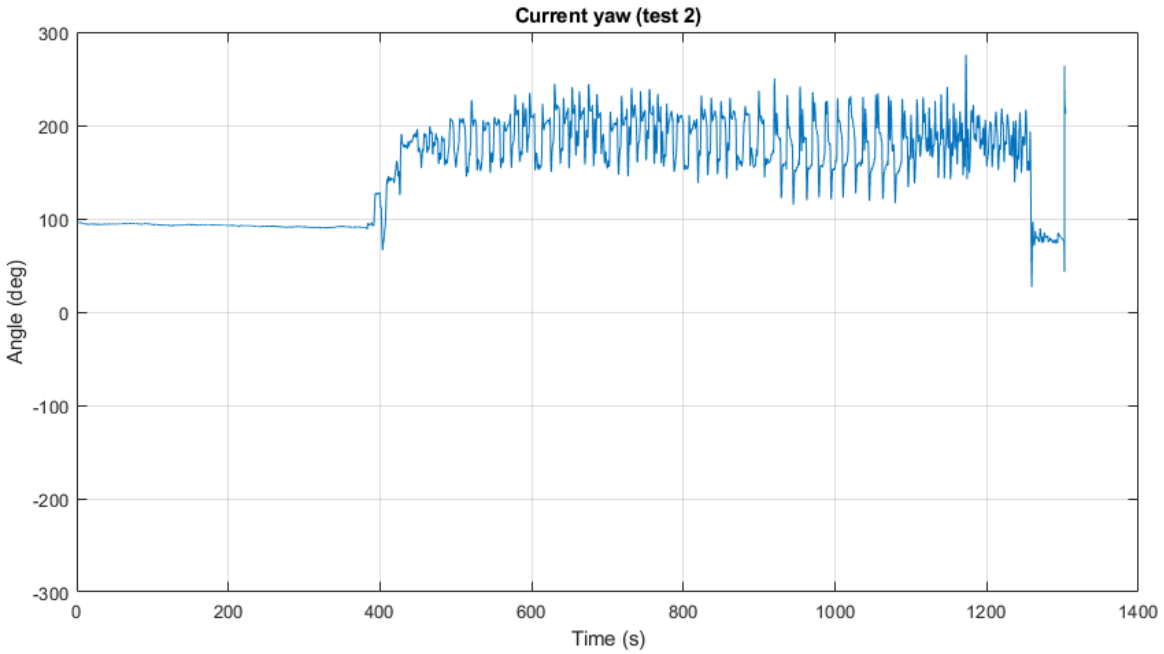


Figure D.24: Yaw angle of the Nederdrone during the test flight.

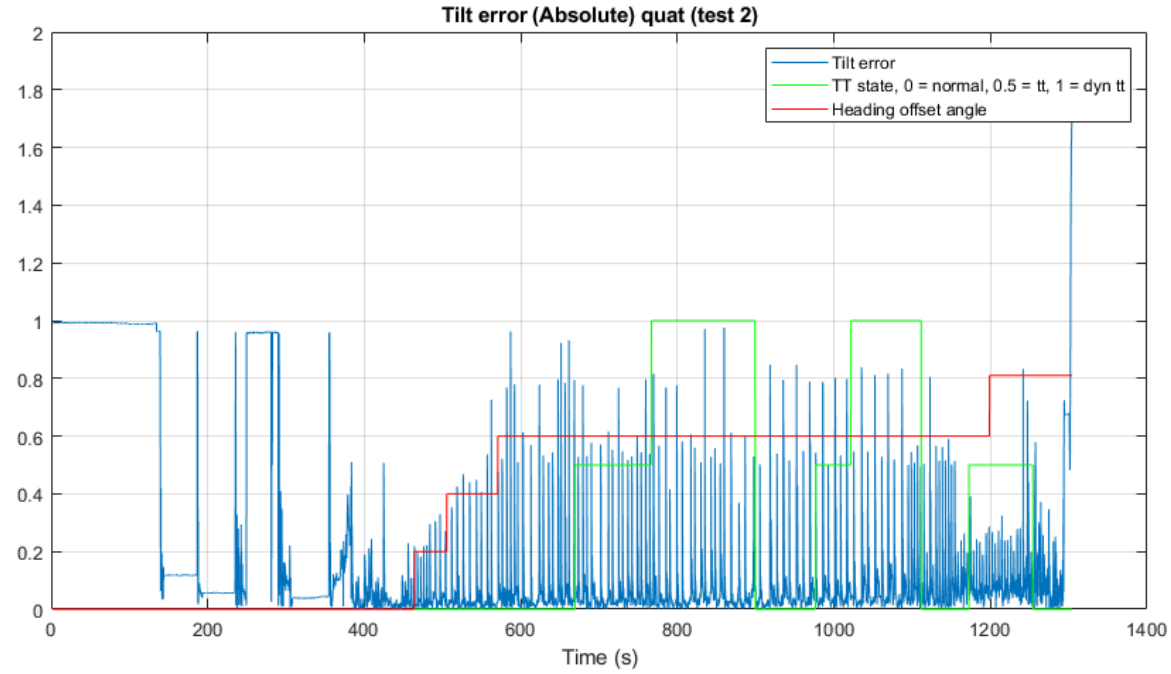


Figure D.25: Plot of the tilt error, included is the heading offset angle and which error feedback method is used, 0 is the quaternion feedback, 0.5 is the tilt-twist and 1 is the dynamic tilt-twist.

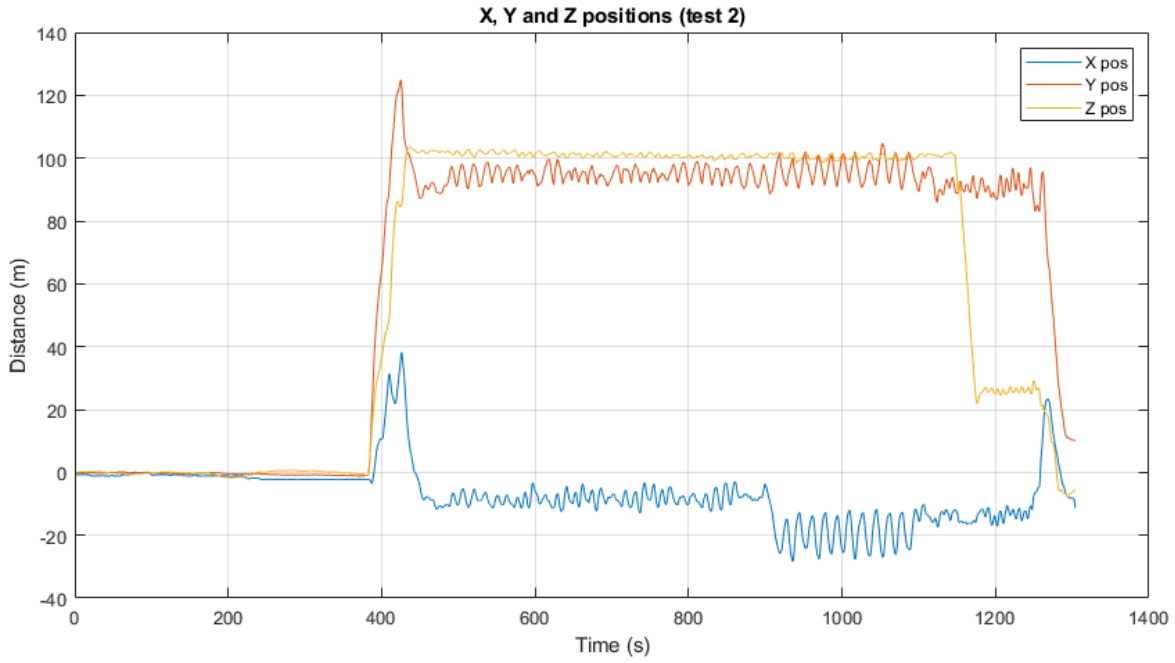


Figure D.26: Xyz position of the Nederdrone

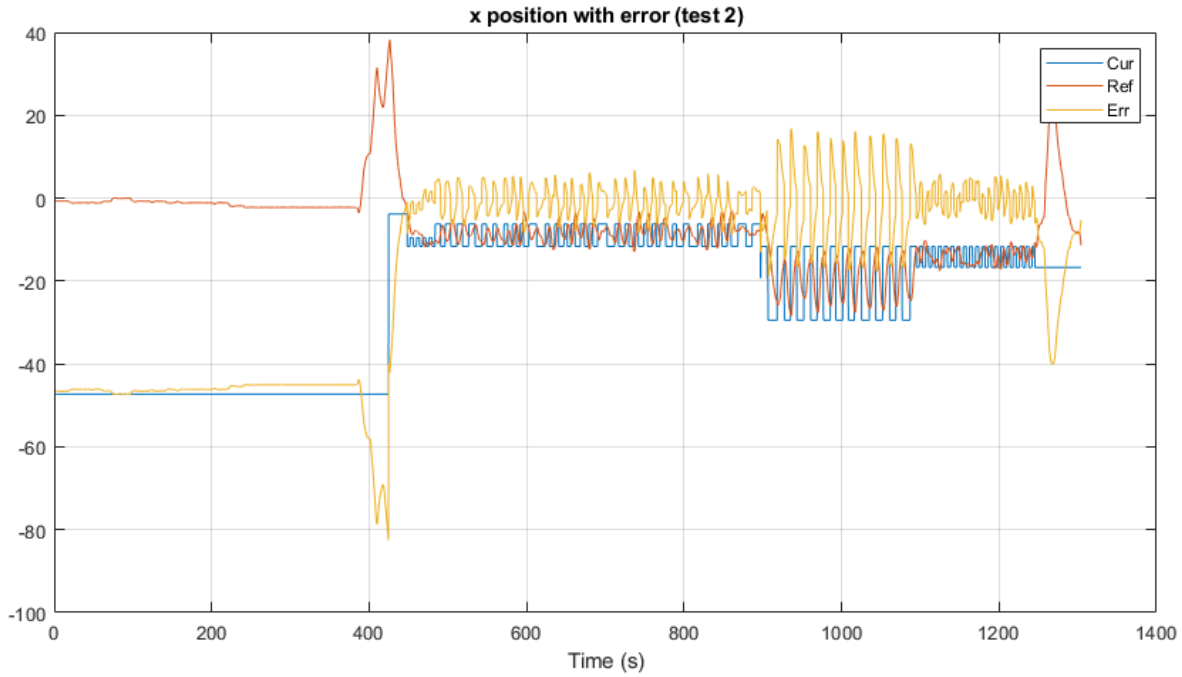


Figure D.27: The x position of the Nederdrone against the desired x position, the error is also included.

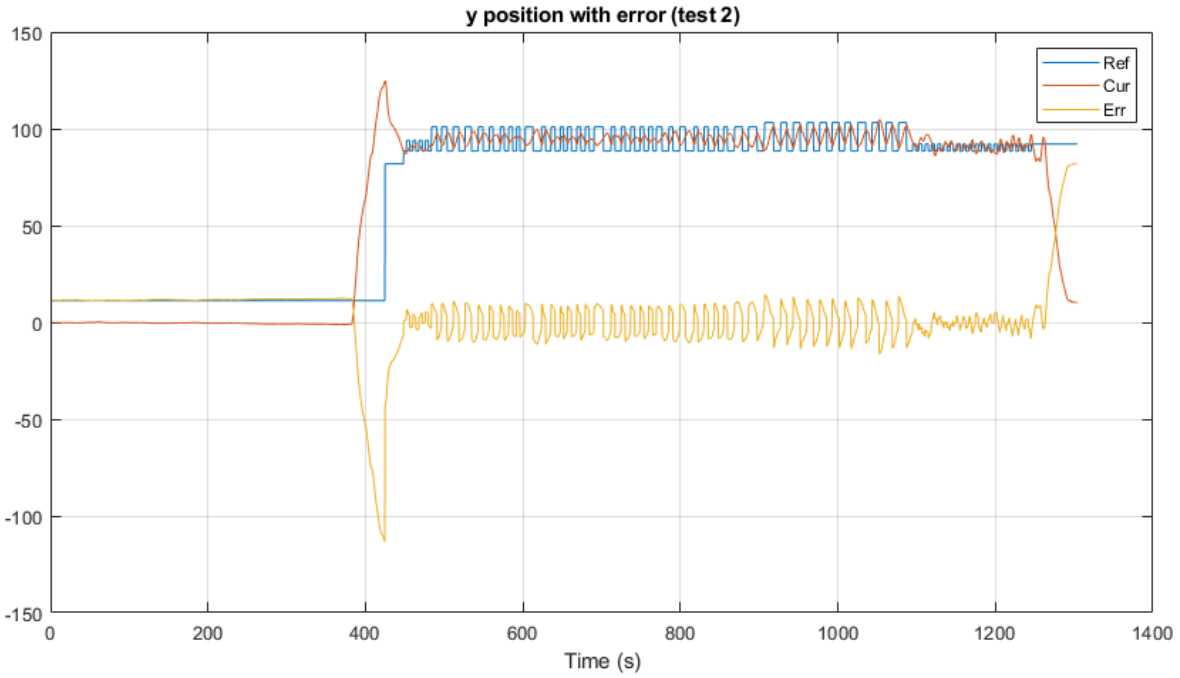


Figure D.28: The y position of the Nederdrone against the desired x position, the error is also included.

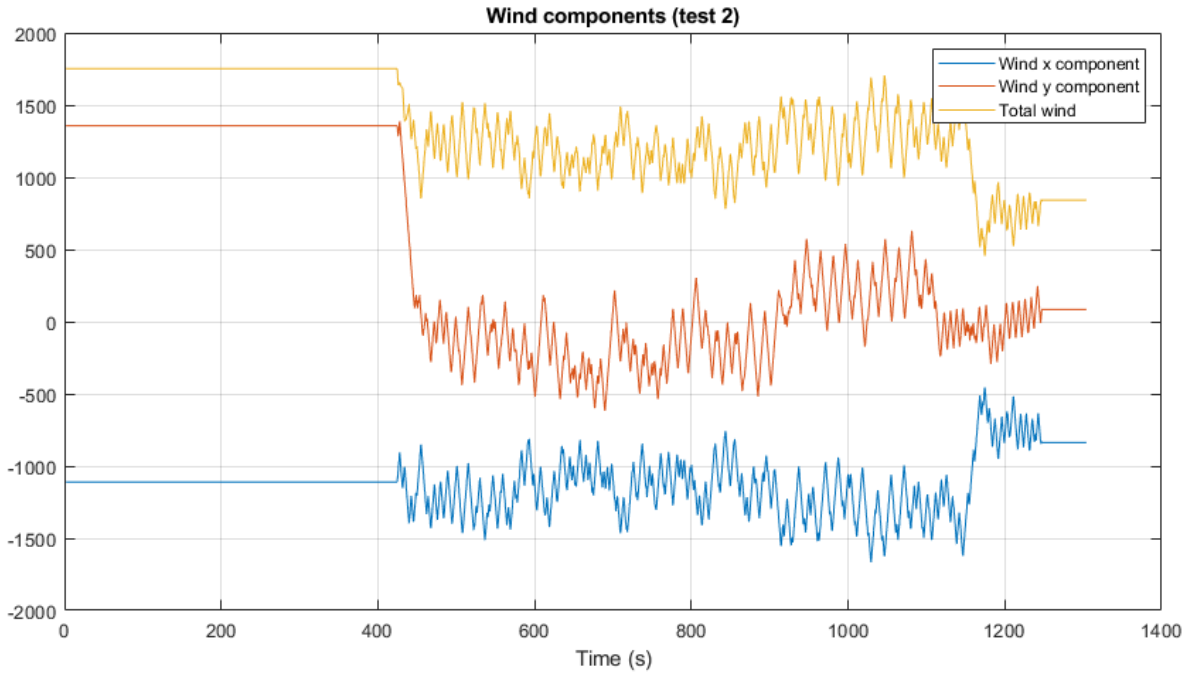


Figure D.29: The wind components, x is the east direction and y in the north direction.

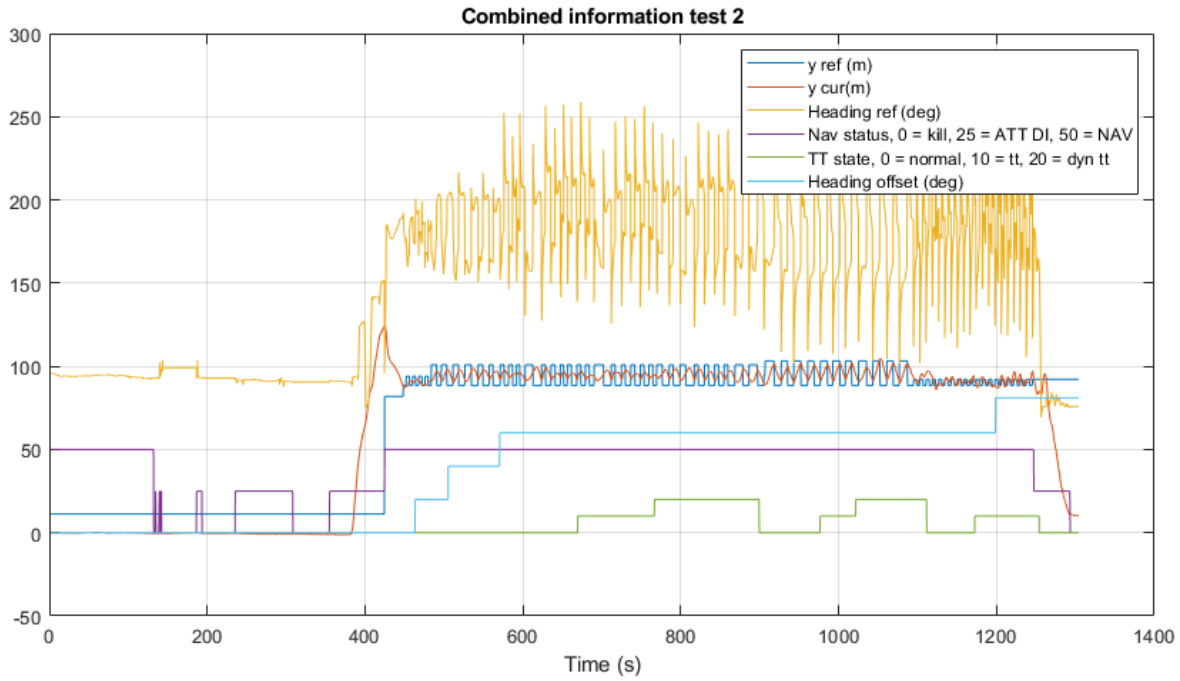


Figure D.30: Graph showing the y position of the Nederdrone and the desired y position, the desired heading, the navigation status (manual control or autopilot on), which feedback error is used and the heading offset.

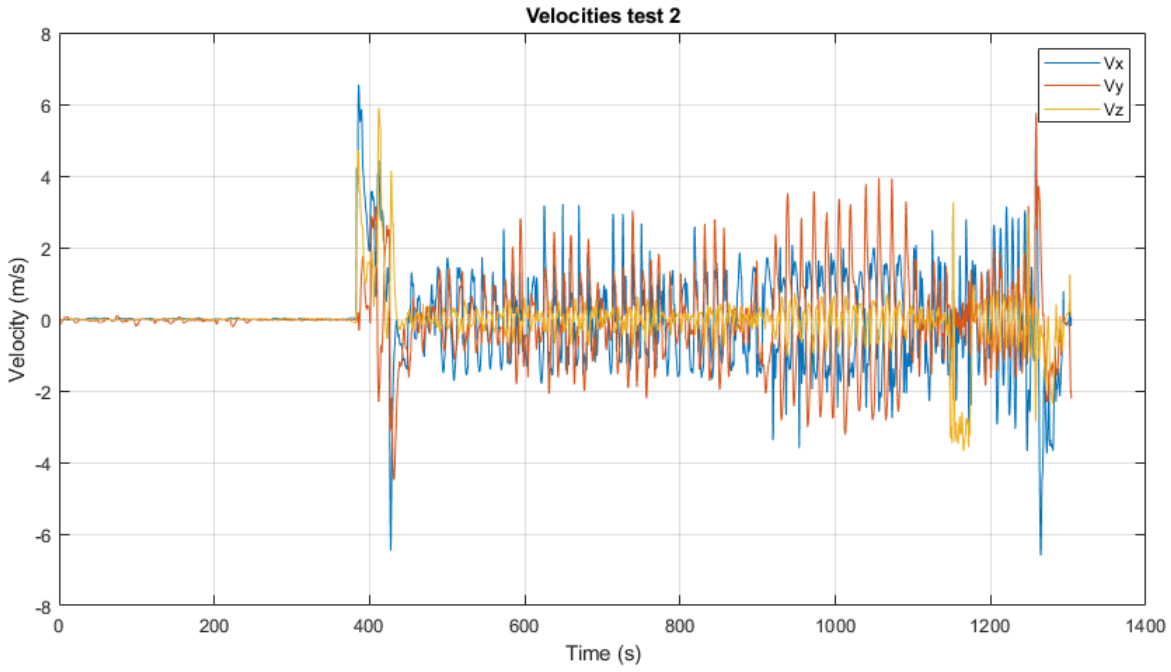


Figure D.31: The velocities of the Nederdrone

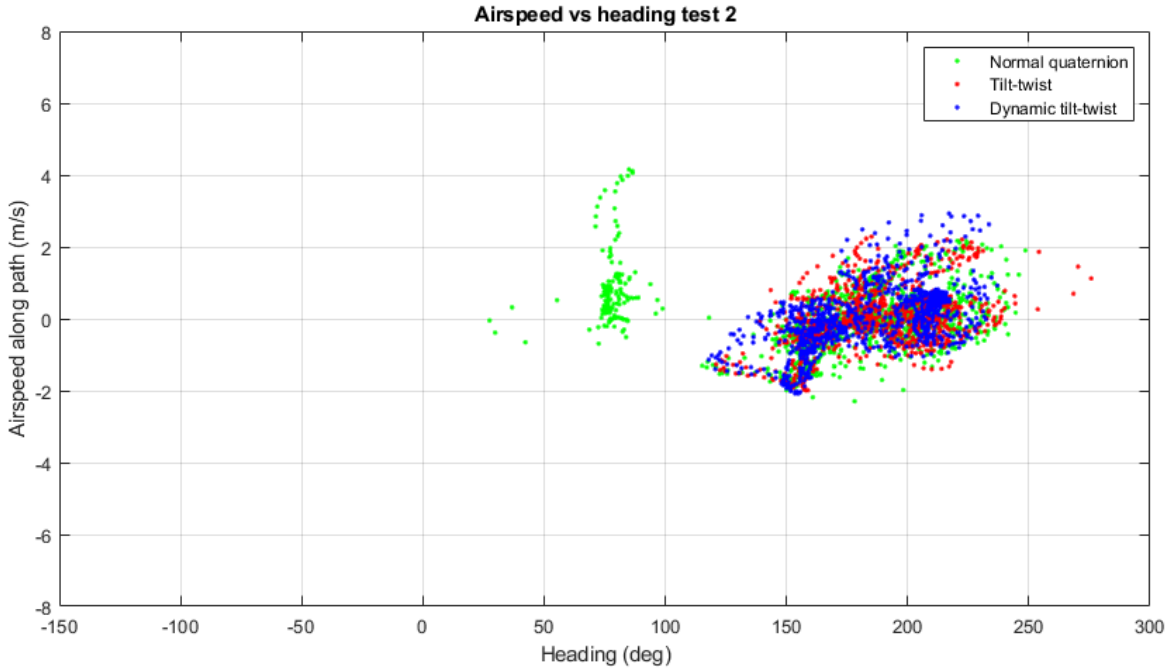


Figure D.32: The graphs show the heading of the Nederdrone against the airspeed along the desired flight path.

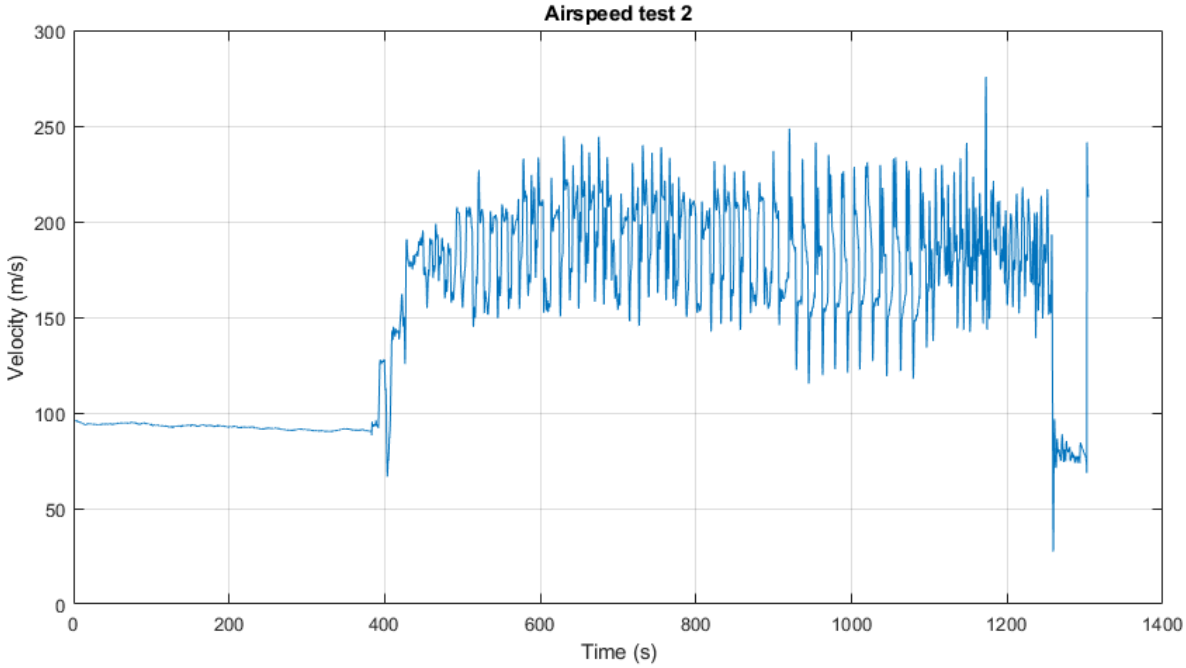


Figure D.33: The airspeed of the Nederdrone.

D.2.1. Test 1 zoomed

This section includes information of a zoomed section of the flight (30 seconds).

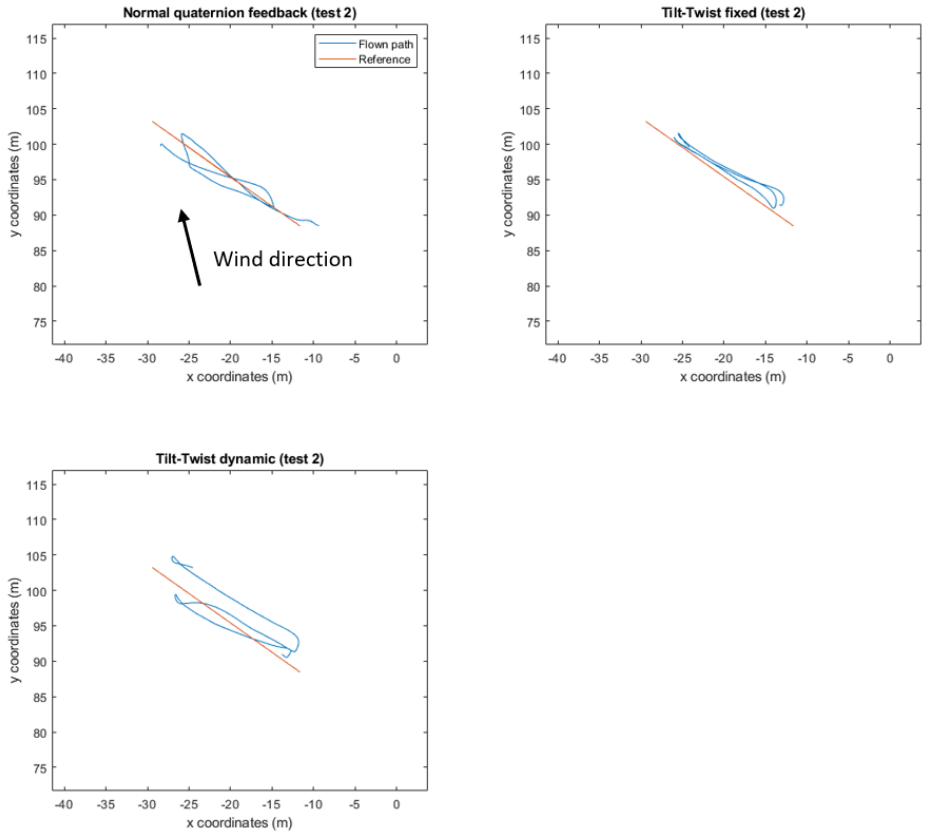


Figure D.34: Flight paths during test 2 (30 seconds)

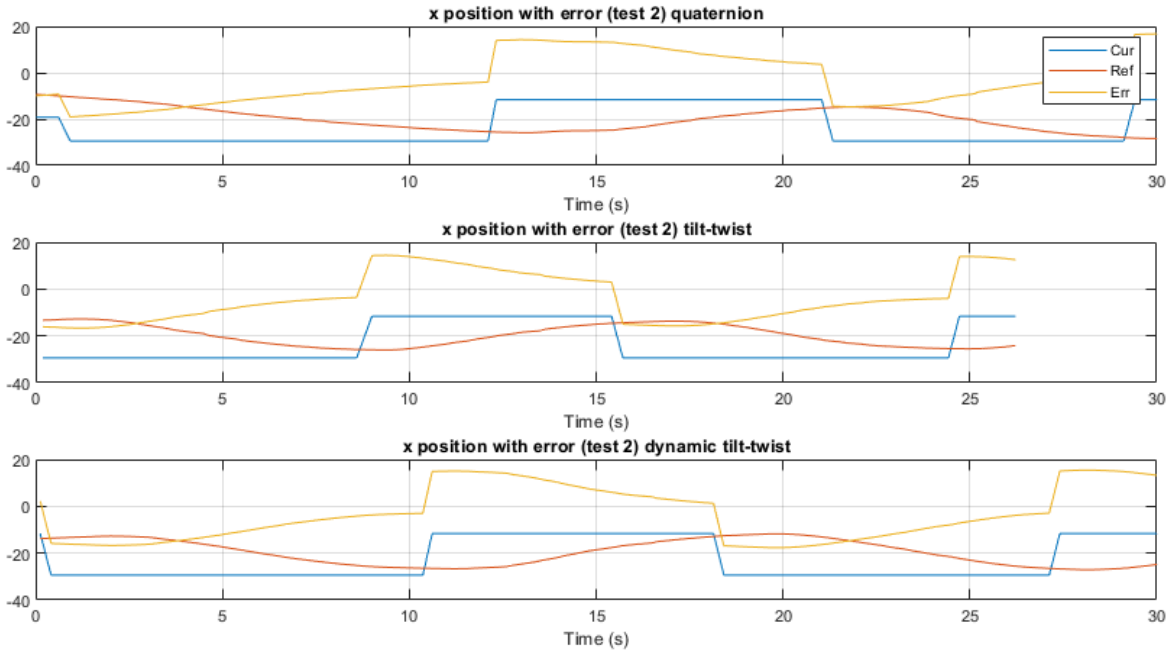


Figure D.35: The x required and the real x positions of the Nederdrone.

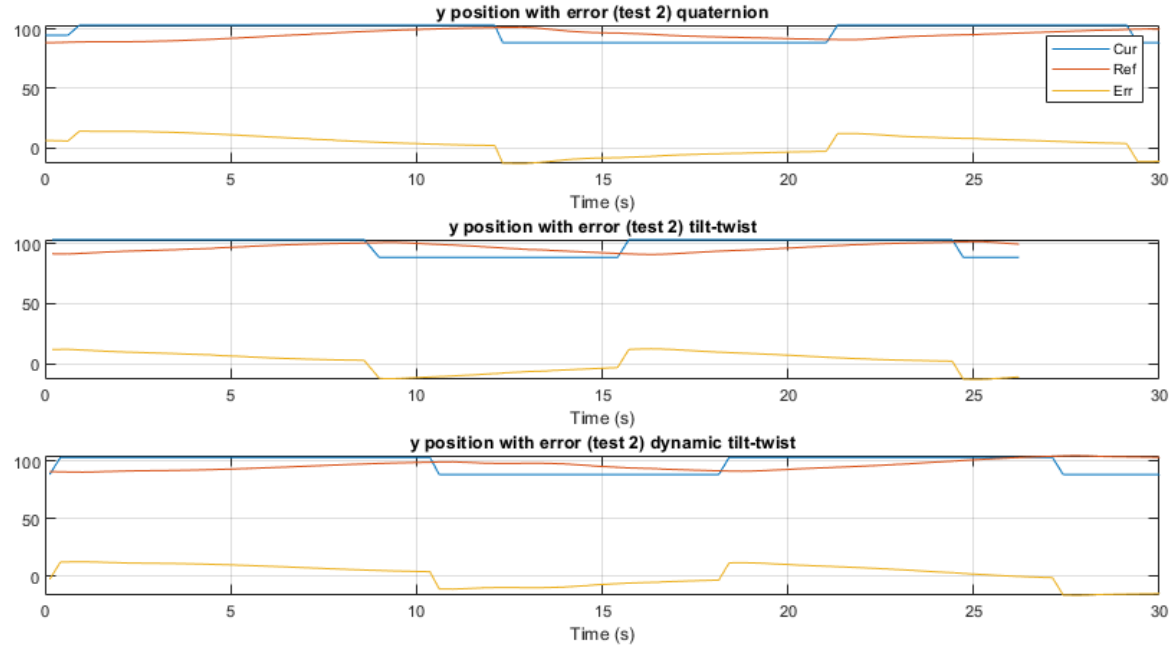


Figure D.36: The x required and the real y positions of the Nederdrone.

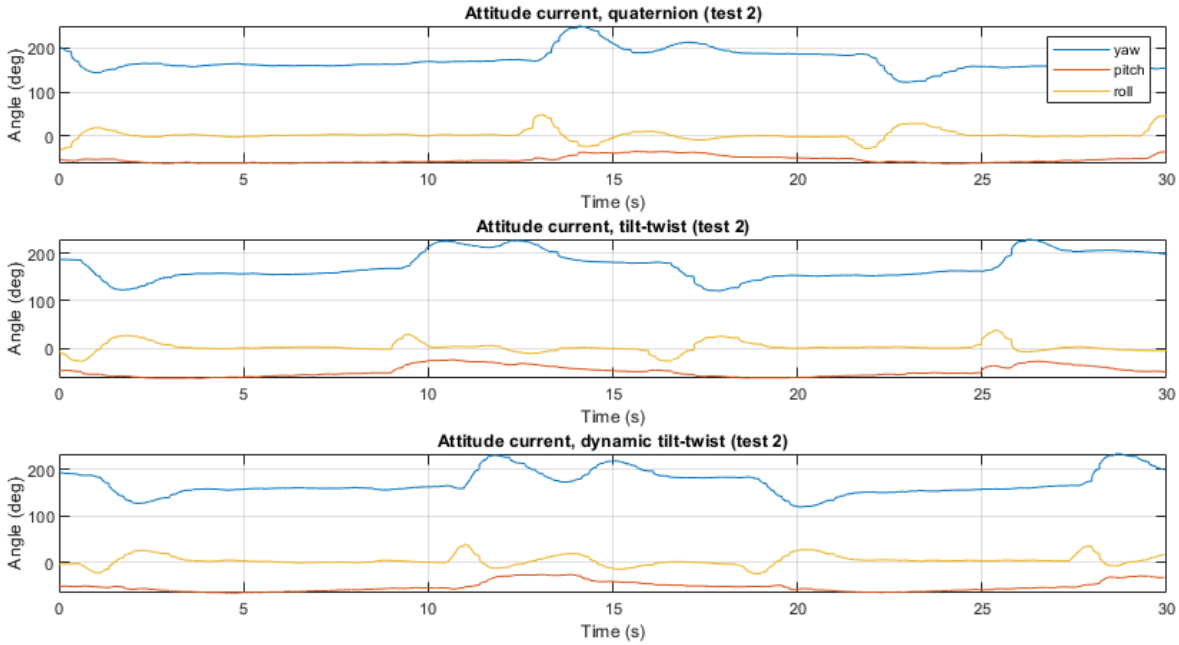


Figure D.37: The real attitudes of the Nederdrone.

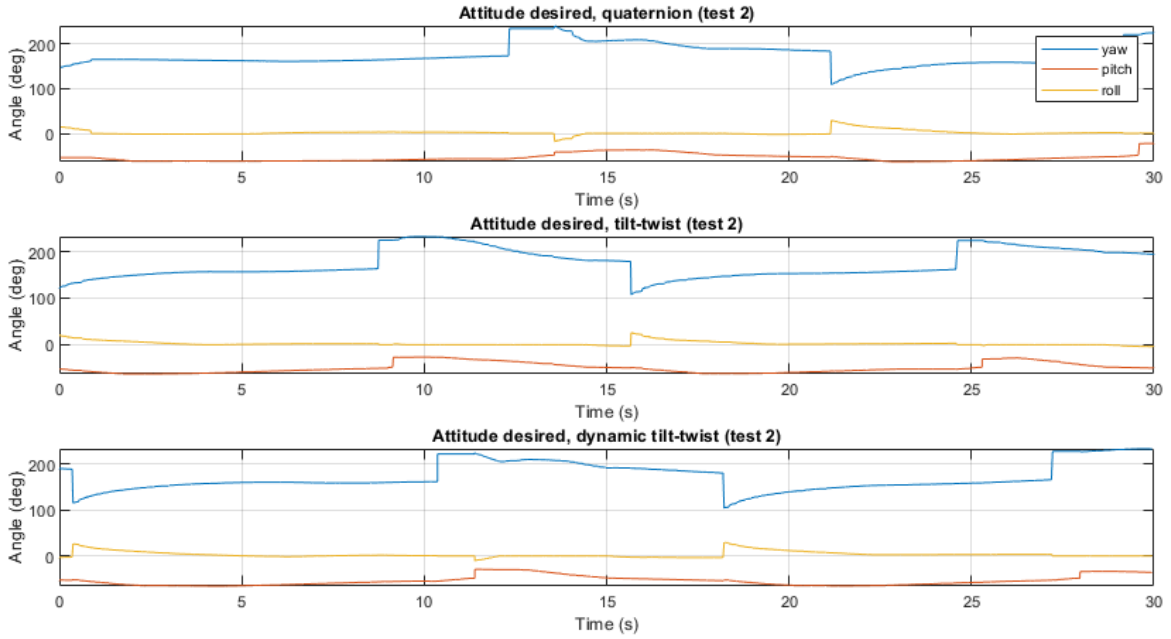


Figure D.38: The required attitudes of the Nederdrone.

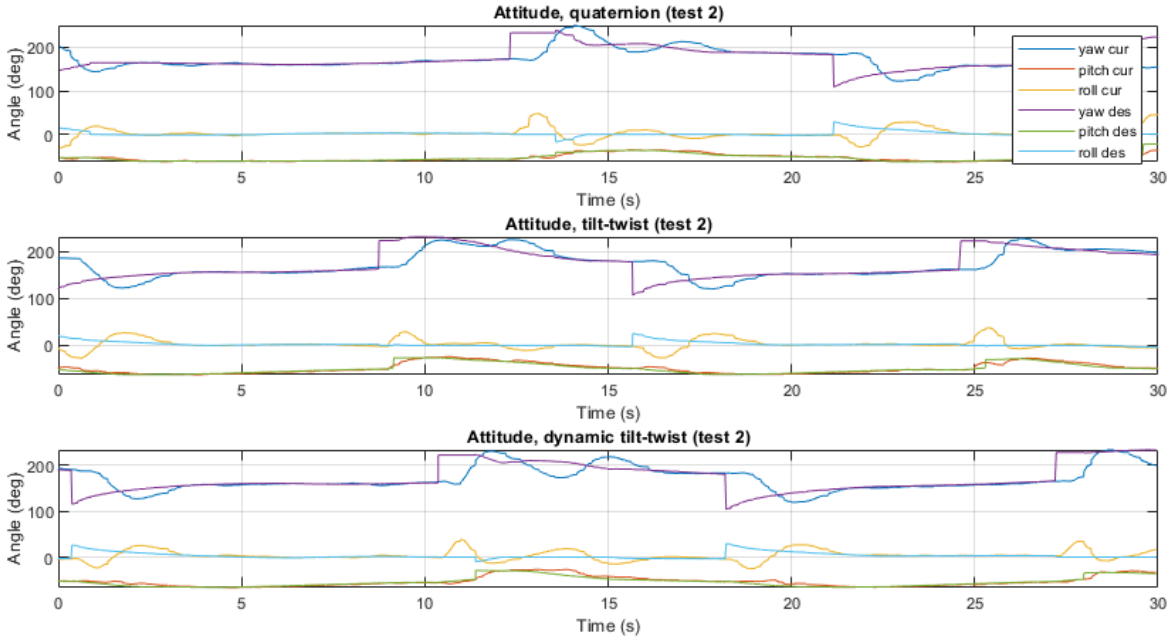


Figure D.39: The required and real attitudes of the Nederdrone.

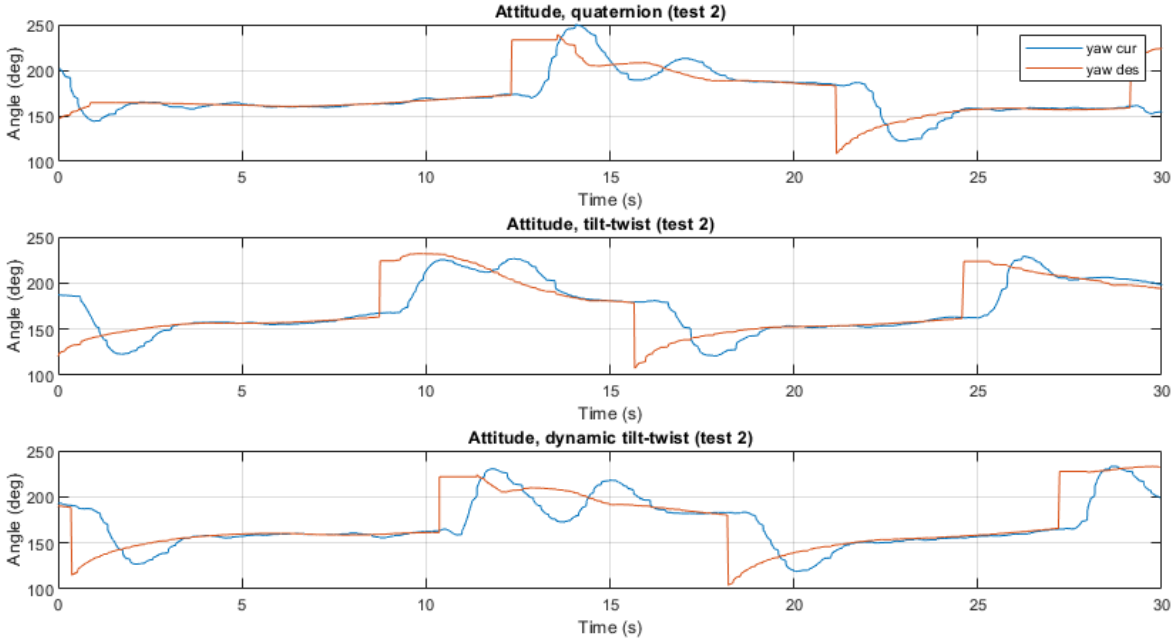


Figure D.40: The required yaw compared to the real yaw.

Bibliography

[1] S. Bhartiya A. Gupta and P. Nataraj. A novel approach to multiparametric quadratic programming. 2011. doi:10.1016/j.automatica.2011.06.019.

[2] A. Kumar A. Saha and A. K. Sahu. FPV drone with GPS used for surveillance in remote areas. In *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 62–67, 2017.

[3] V. Ambrosia A. Watts and E. Hinkley. Unmanned Aircraft Systems in Remote Sensing and Scientific Research: Classification and Considerations of Use. 2012. doi:10.3390/rs4061671.

[4] A. Subiantoro M. Djemai B. Pratama, A. Muis and R. Atitallah. Quadcopter Trajectory Tracking and Attitude Control Based on Euler Angle Limitation. 2018. doi:10.1109/CEIT.2018.8751819.

[5] M. Bodson. Evaluation of Optimization Methods for Control Allocation. 2002. doi:10.2514/2.4937.

[6] E. Smeur K. van Hecke F. van Tienen E. van der Horst C. De Wagter, R. Ruijsink and B. Remes. Design, control, and visual navigation of the DelftaCopter VTOL tail-sitter UAV. *Journal of Field Robotics*, pages 937–960, 2018. doi:10.1002/rob.21789.

[7] D. Prett C. Garcia and M. Morar. Model predictive control: Theory and practice. 1989. doi:10.1016/0005-1098(89)90002-2.

[8] S. Cao and L. Shen. Adaptive Incremental Nonlinear Dynamic Inversion Control Based on Neural Network for UAV Maneuver. 2019. doi:10.1109/AIM.2019.8868510.

[9] W. F. Chana and J. F. Coleman. World's First VTOL Airplane Convair/Navy XFY-1 Pogo. 1996. doi:10.2307/44725614.

[10] W. Chen. Closed-Form Nonlinear MPC for Multivariable Nonlinear Systems with Different Relative Degree. 2003. doi:10.1109/ACC.2003.1242497.

[11] J. H. Christner. Pioneer unmanned air vehicle accomplishments during Operation Desert Storm. In Thomas W. Augustyn and Paul A. Henkel, editors, *Airborne Reconnaissance XV*, volume 1538, pages 201 – 207. International Society for Optics and Photonics, SPIE, 1991. doi:10.1117/12.48706. URL `https://doi.org/10.1117/12.48706`.

[12] C. Ha D. Lee and Z. Zuo. Backstepping Control of Quadrotor-Type UAVs and Its Application to Teleoperation over the Internet. 2013. doi:doi.org/10.1007.

[13] R. Ghosh Dastidar. On the Advantages and Limitations of Sliding Mode Control for Spacecraft. 2010. doi:10.2514/6.2010-8777.

[14] C. de Wagter E. Smeur, D. Höppener. Prioritized Control Allocation for Quadrotors Subject to Saturation. 09 2017.

[15] G.de Croon E. Smeur and Q. Chu. Cascaded incremental nonlinear dynamic inversion for MAV disturbance rejection. 2018. doi:10.1016/j.conengprac.2018.01.003.

[16] Q. Chu E. Smeur and G. de Croon. Adaptive Incremental Nonlinear Dynamic Inversion for attitude control of Micro Air Vehicles. 2016. doi:10.2514/1.G001490.

[17] J. Xiong E. Zheng and J. Luo. Second order sliding mode control for a quadrotor UAV. 2014. doi:10.1016/j.isatra.2014.03.010.

[18] D. Enns. Control Allocation Approaches. 1998. doi:10.2514/6.1998-4109.

[19] N. Guerreiro and A. Moutinho. Robust Incremental Backstepping Controller for the Attitude and Airspeed Tracking of a Commercial Airplane. 2019. doi:10.1109/ICMAE.2019.8881034.

[20] D. Lee H. Lim, J. Park and H.J. Kim. Build Your Own Quadrotor: Open-Source Projects on Unmanned Aerial Vehicles. 2012. doi:10.1109/MRA.2012.2205629.

[21] O. Härkegård. Efficient Active Set Algorithms for Solving Constrained Least Squares Problems in Aircraft Control Allocation. 2002. doi:10.1109/CDC.2002.1184694.

[22] O. Härkegård. Dynamic Control Allocation Using Constrained Quadratic Programming. 2004. doi:10.2514/1.11607.

[23] T. McLain R. Beard J. Beach, M. Argyle and S. Morris. Tailsitter attitude control using resolved tilt-twist. 2014. doi:10.1109/ICUAS.2014.6842322.

[24] E. Johnson and A. Calise. Pseudo-control hedging: A new method for adaptive control. 2000.

[25] S. Ge K. Feng, W. Li and F. Pan. Packages delivery based on marker detection for UAVs. In *2020 Chinese Control And Decision Conference (CCDC)*, pages 2094–2099, Aug 2020. doi:10.1109/CCDC49329.2020.9164677.

[26] T. Keijzer. Flight testing of incremental backstepping based control laws with angular accelerometer feedback. *AIAA Scitech 2019 Forum*, 2019. doi:10.2514/6.2019-0129.

[27] H. Khan and M. Kadri. Attitude and Altitude Control of Quadrotor by Discrete PID control and Nonlinear Model Predictive Control. 2015. doi:10.1109/ICICT.2015.7469486.

[28] A. Kunickaite L. Balasevicius. Discrete-time PID Controller Design in Programmable Logical Controllers. *2007 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, pages 86–90, 2007. doi:10.1109/IDAACS.2007.4488380.

[29] G. Mattei L. Canetta and A. Guanziroli. Exploring commercial UAV market evolution from customer requirements elicitation to collaborative supply network management. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1016–1022, June 2017. doi:10.1109/ICE.2017.8279993.

[30] R. Beard T. McLain M. Argyle, J. Beach and S. Morris. Quaternion Based Attitude Error for a Tailsitter in Hover Flight. 2014. doi:10.1109/ACC.2014.6859324.

[31] J. Petersen and M. Bodson. Constrained Quadratic Programming Techniques for Control Allocation. 2006. doi:10.1109/TCST.2005.860516.

[32] L. Meier R. Bapst, R. Ritz and M. Pollefeys. Design and implementation. 2015. doi:10.1007/978-981-4451-66-6_3.

[33] P. Murrieri S. Bouabdallah and R. Siegwart. Design and control of an indoor micro quadrotor. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pages 4393–4398 Vol.5, 2004. doi:10.1109/ROBOT.2004.1302409.

[34] B. Schofield. On Active Set Algorithms for Solving Bound-Constrained Least Squares Control Allocation Problems. 2008. doi:10.1109/ACC.2008.4586883.

[35] S. Sieberling. Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. 2010. doi:10.2514/1.49978.

[36] E. Smeur. Incremental Control of Hybrid Micro Air Vehicles. 2018. doi:10.4233/uuid:23c338a1-8b34-40a6-89e9-997adbdafd75.

[37] R. Suzuki A. Oosedo K. Go Y. Hoshino A. Konno T. Matsumoto, K. Kita and M. Uchiyam. A Hovering Control Strategy for a Tail-Sitter VTOL UAV that Increases Stability Against Large Disturbance. 2010. doi:10.1109/ROBOT.2010.5509183.

[38] C. Thipyopas and J. Moschetta. Experimental Analysis of a Fixed-Wing VTOL MAV in Ground Effect. 2010. doi:10.1260/1756-8293.2.1.33.

[39] P. Tondel and T. Johnson. Control allocation for yaw stabilization in automotive vehicles using multiparametric nonlinear programming. 2005. doi:10.1109/ACC.2005.1469977.

[40] W. van Ekeren. Incremental Nonlinear Flight Control for Fixed-Wing Aircraft. 2016.

[41] P. van Gils. Adaptive Incremental Backstepping Flight Control. 2015. doi:10.2514/6.2016-1380.

[42] R. C. van 't Veld. Incremental Nonlinear Dynamic Inversion Flight Control. 2019.

[43] J. Virnig and D. Bodden. Multivariable control allocation and control law conditioning when control effectors limit. 2019. doi:10.2514/6.1994-3609.

[44] R. Kuchar Q. Chu W. van Ekeren, G. Looye and E. van Kampen. Design, Implementation and Flight-Test of Incremental Backstepping Flight Control Laws. 2018. doi:10.2514/6.2018-0384.

[45] E. W. Weisstein. *Rodrigues' Rotation Formula.*, 2020 (accessed October 9, 2020) https://mathworld.wolfram.com/RodriguesRotationFor mula.html. URL `https://mathworld.wolfram.com/RodriguesRotationFormula.html`.

[46] Y. Wang Z. Li S. Shen X. Lyu, H. Gu and F. Zhang. Design and Implementation of a Quadrotor Tail-sitter VTOL UAV. 2017. doi:10.1109/ICRA.2017.7989452.

[47] F. Zhanqi and L. Lin. The High Angle of Attack Aerodynamic modeling and Nonlinear Dynamic Inversion Flight Control Law Design. 2012. doi:10.1109/INDIN.2012.6300870.