Incremental Nonlinear Dynamic Inversion applied to Quadrotor UAV Control

Ricardo Filipe Encarnação Almeida
ricardo.e.almeida@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

November 2017

Abstract

The research work aims to develop a feasible control law using Nonlinear Dynamic Inversion (NDI) methodology, both in its classical and incremental variants. In first place a review of relevant literature related with nonlinear multirotor control is presented, as well as a background theory in NDI, followed by a modulation of the quadrotor. This three components served as a basis for the implementation of an attitude and vertical velocity controller using both techniques, NDI and Incremental Nonlinear Dynamic Inversion (INDI). In INDI controller the state derivate estimate was performed with a second order bandpass filter, that not only estimates the derivative but also filters part of the high frequency noise from sensor measurements. A simulation platform was built in MATLAB/Simulink, including a model of the quadrotor and of its sensors. Both control laws were compared with a baseline controller from Ardupilot autopilot system that uses PID loops, proving the feasibility of this methods for quadrotor control. The INDI version of the controller was then implemented in Ardupilot code and compared with the original controller from this platform in the simulation environment of Ardupilot. The simulation results proved that the performance of this new control technique can match the original controller, having the advantages of: running at lower frequency, which requires a smaller computational power in real time applications; presenting a considerably smaller number of parameters to be adjusted; having robustness to model uncertainties.

Keywords: Incremental Nonlinear Dynamic Inversion, Quadrotor, Nonlinear control, Ardupilot, Attitude control

1. Introduction

In recent decades the interest in Unmanned Aerial Vehicles (UAVs) has considerably increased, due to the ability of executing their actions without an onboard pilot, being controlled by an airborne control and guidance system, commonly known as an autopilot, or even remotely. Initial applications were exclusively related with fixed wing configurations used for military operations, since they greatly reduce the risk of human loss and reach the objectives of the operations with higher efficiency. More recently multirotor UAVs, also known as multicopters, have become a relevant solution for performing a wide range of applications, thanks to their Vertical Takeoff and Landing and high manoeuvrability capabilities.

Among the numerous applications in which a multicopter can be used one can stand out two categories, namely, the applications where a multicopter is used with a camera, capturing images and video, and the applications where it is used as a transportation device. The first category includes search and rescue operations; agriculture; taking aerial footage for cinema, commercials, music concerts and festivals; monitoring the state of large constructions. In the second category one can include the transportation of donated organs and package delivering.

1.1 *Topic Overview*

All these applications can present several challenges in terms of controlling quadrotor UAVs, due to the high number of unpredicted disturbances that can be present in some flight situations and due to the required large flight envelope, including some more aggressive manoeuvres. The control problem was initially addressed with linear controllers, due to their simplicity in implementation. However, a quadrotor is an underactuated six degree of freedom nonlinear system, and linear control can only be applied near a specific stable flight condition, i.e. a trimming point. To extend the flight envelope one can design several linear controllers, which one operating at a specific trimming point, including also some functions that choose the best controller for the current flight condition. This technique is

known as gain scheduling ([1], [2]), and even though it has a simple implementation, its design represents a time consuming process without stability guarantees, so that alternatives have been proposed using inherent nonlinear control design methods in recent years, that would provide robust solutions to quadrotor control.

Some of the most common nonlinear control methods are Nonlinear Dynamic Inversion (NDI) ([3], [4]), Backstepping (BKS) ([5], [6]) and Sliding Mode Control (SMC) ([7], [8]). The first two also have incremental variants known as Incremental Nonlinear Dynamic Inversion (INDI) ([9], [10]) and Incremental Backstepping (IBKS) ([11]). With BKS one can prove the stability of the designed controller, which is not possible if NDI is used, but the greater complexity of the design process makes NDI a more interesting approach to obtain a simple and robust nonlinear controller for a quadrotor. The latter technique, SMC, produces a robust controller with good overall performance, but it obtains this result at the cost of a high control action, and therefore a large battery consumption. In conclusion, NDI and INDI seem promising control techniques to solve the problem of quadrotor UAV control.

1.2 *Objectives*

The main objectives of this document are related with the implementation of a nonlinear controller for the purpose of controlling a quadrotor. In order to compare the new method with existing and validated control techniques Ardupilot will be used, which is an autopilot platform that has been utilized by enthusiasts and professionals in several UAV applications. For the experimental validation tests one will use an UAVision UX-Spyro quadrotor. The following proposal was made to UAVision: the company provides the quadrotor for experimental validation tests and, in exchange, they receive a developed and implemented nonlinear control algorithm with robust properties and a simpler design (i.e., less tuning parameters). In an early stage of the development, the control solution was analyzed and validated using the MATLAB/Simulink environment. Considering the information of the previous sentences, it is possible to define the objectives as: develop a nonlinear control algorithm for UX-Spyro quadrotor; analyze and validate the control solution in a MATLAB/Simulink environment; adapt the control algorithm to Ardupilot and validate the implementation in its simulation platform; validate the control solution with experimental results. Unfortunately, the experimental tests were not done because UAVision was not available to perform them before the delivery of this document.

2. Nonlinear Dynamic Inversion

NDI is a control method developed in the late 1970's to overcome the limitations of conventional linear techniques. Its intuitive idea consists in the cancellation of the nonlinearities in a system, resulting in a linear structure for the closed loop dynamics. Considering the n^{th} order MIMO system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u} \tag{1a}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{1b}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the n^{th} order state vector; $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^m$ are the control input and output vectors, respectively; $\mathbf{f}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ are smooth field vectors in $\mathbb{R}^n$ and $\mathbf{G}(\mathbf{x})$ is a square matrix in $\mathbb{R}^n$.

Differentiating a component y_i of the output vector r_i times, until an explicit dependence on the input appears, and using Lie derivative notation [12], one gets:

$$y_i^{(r_i)} = L_f^{r_i} h_i(\mathbf{x}) + \sum_{j=1}^{n} L_{g_j} L_f^{r_i - 1} h_i(\mathbf{x}) u_j \tag{2}$$

r_i is called the relative degree of the i^{th} output component. The total relative degree r is:

$$r = \sum_{j=1}^{n} r_j \tag{3}$$

If the total relative degree is smaller than the total system order, there are internal dynamics that must be bounded to assure an effective control. Collecting all the differentiated outputs y_i one can obtain a similar expression depending on the input $\mathbf{u}$:

$$\begin{bmatrix} y_1^{(r_1)} \\ \vdots \\ y_n^{(r_n)} \end{bmatrix} = \mathbf{a}(\mathbf{x}) + \mathbf{B}(\mathbf{x})\mathbf{u} \tag{4}$$

where $\mathbf{a}(\mathbf{x})$ and $\mathbf{B}(\mathbf{x})$ are constituted by the Lie derivatives, represented in Equations (6) and (7). Replacing the differentiated outputs vector by a virtual control vector $\nu = [\nu_1, \ \ldots, \ \nu_n]^T$ one can obtain an expression for the input $\mathbf{u}$:

$$\mathbf{u} = \mathbf{B}(\mathbf{x})^{-\mathbf{1}} \left[\nu - \mathbf{a}(\mathbf{x})\right] \tag{5}$$

$$\mathbf{a}(\mathbf{x}) = \begin{bmatrix} L_f^{r_1} h_1(\mathbf{x}) \\ \vdots \\ L_f^{r_n} h_n(\mathbf{x}) \end{bmatrix} \tag{6}$$

$$\mathbf{B}(\mathbf{x}) = \begin{bmatrix} L_{g_1} L_f^{r_1 - 1} h_1(\mathbf{x}) & \ldots & L_{g_n} L_f^{r_1 - 1} h_1(\mathbf{x}) \\ \vdots & \vdots & \vdots \\ L_{g_1} L_f^{r_n - 1} h_n(\mathbf{x}) & \ldots & L_{g_n} L_f^{r_n - 1} h_n(\mathbf{x}) \end{bmatrix} \tag{7}$$

If the number of inputs is different from the number of outputs, $\mathbf{B}(\mathbf{x})$ is not square and therefore one needs to use a MoorePenrose pseudoinverse. The closed loop form $y_i^{(r_i)} = \nu_i$ for $i = 1, \ldots, n$ shows that each component is independent and decoupled from the remaining components. Thus, the controllers for each output channel can be designed separately, in order to meet the required tracking performance.

3. Incremental Nonlinear Dynamic Inversion

The concept of INDI was developed to reduce the dependence from an exact model of the system, and consequently improve the robustness to model uncertainties. For this derivation one can consider a general nonlinear MIMO system with the following form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{8}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the n^{th} order state vector; $\mathbf{u}$ is the control input vector and $\mathbf{f}(\mathbf{x}, \mathbf{u})$ is a nonlinear function. Applying a first order Taylor series expansion, the system can be linearized around the point $(\mathbf{x} = \mathbf{x_0}, \mathbf{u} = \mathbf{u_0})$:

$$\dot{\mathbf{x}} \approx \dot{\mathbf{x}}_\mathbf{0} + \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x_0}, \mathbf{u}=\mathbf{u_0}} (\mathbf{x} - \mathbf{x_0}) + \left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x}=\mathbf{x_0}, \mathbf{u}=\mathbf{u_0}} (\mathbf{u} - \mathbf{u_0}) \tag{9}$$

This expression can be further simplified assuming a high sample rate for the system and using the principle that input dynamics are considerably faster than state dynamics, hence $\mathbf{x} - \mathbf{x_0} \approx 0$, obtaining Equation (10). Replacing $\left.\frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\right|_{\mathbf{x}=\mathbf{x_0}, \mathbf{u}=\mathbf{u_0}}$ by $\mathbf{G}(\mathbf{x_0}, \mathbf{u_0})$, and introducing the incremental input $\Delta\mathbf{u} = \mathbf{u} - \mathbf{u_0}$ in Equation (9) one gets:

$$\dot{\mathbf{x}} \approx \dot{\mathbf{x}}_0 + \mathbf{G}(\mathbf{x_0}, \mathbf{u_0})\Delta\mathbf{u} \tag{10}$$

which can be solved for $\Delta\mathbf{u}$ if matrix $\mathbf{G}(\mathbf{x_0}, \mathbf{u_0})$ is a square nonsingular matrix (i.e., it can be inverted, and consequently the system has the same number of inputs and states). Again if one introduces the virtual control input $\nu = \dot{\mathbf{x}}$, the incremental form is given then by:

$$\Delta\mathbf{u} = \mathbf{G^{-1}}(\mathbf{x_0}, \mathbf{u_0})\,(\nu - \dot{\mathbf{x}}_\mathbf{0}) \tag{11}$$

In INDI control theory, $\dot{\mathbf{x}}_\mathbf{0}$ is assumed to be measurable and therefore this method is also denominated sensor based control, in contrast with $\mathbf{f}(\mathbf{x}, \mathbf{u})$ that is an example of model based control. The total control inputs are obtained by adding the previous control input $\mathbf{u_0}$ to $\Delta\mathbf{u}$ (Equation (12)) then, it presents integrative characteristics.

Similarly to NDI the system is now linearized and a linear control law can be designed to meet the tracking or stabilization requirements. A diagram of this approach can be depicted in Figure 1.

$$\mathbf{u} = \mathbf{u_0} + \Delta\mathbf{u} \tag{12}$$

The following conclusions can be drawn from the comparison between NDI and INDI:

- NDI requires the complete knowledge of the system and it is applicable only to systems affine in the control input. Alternatively, INDI only uses the control effectiveness matrix, not requiring a model for $f(x)$. For this reason INDI is expected to be more robust to model uncertainties. Moreover, INDI approach can be used with a general nonlinear system.
- INDI controller needs a high sampling frequency, an actuation dynamics much faster than a system dynamics and the observations of both the state derivatives and the control input signal. This represents a disadvantage of this controller, since it will be more dependent from sensor noise, biases and delays.

4. Quadrotor Modelling and Controller Implementation

4.1 *Quadrotor Model*

An important aspect in the modelling of the quadrotor is the choice of the body reference frame, since it will influence both the inertia and actuation matrices. In this case a body reference frame equal to the one used in Ardupilot was chosen (x axis pointing forward, y axis pointing to the right and z axis pointing down), avoiding compatibility issues during the implementation of the control algorithm. For the inertial reference frame a regular North-East-Down (NED) frame was used.

The dynamic model is composed by the Newton-Euler equations of motion. Considering the two main forces applied to the quadrotor, namely, the gravity (F_g) and thrust (F_t) forces, the linear equation of motion is given by Equation (13), where $\mathbf{V}$ is the velocity vector expressed in the inertial frame and $\mathbf{R_b^i}$ represents the transformation from the body frame to inertial frame [13].

$$m\dot{\mathbf{V}} = \mathbf{R_b^i} \begin{bmatrix} 0 \\ 0 \\ -F_t \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ F_g \end{bmatrix} \tag{13}$$

In the case of quadrotor control it is more useful to write the moments equation with respect to the body frame (Equation (14)), where subscript $\mathbf{b}$ is included in the variables that are now expressed in the body frame, for the sake of clarity, and with math symbol $\times$ representing the cross product. $\mathbf{M_b}$ is the moment vector, $\mathbf{I_b}$ represents the inertia matrix and $\omega_\mathbf{b} = (p, q, r)$ is the angular velocity vector.

$$\mathbf{M_b} = \mathbf{I_b}\ \dot{\omega}_\mathbf{b} + \omega_\mathbf{b} \times \mathbf{I_b}\ \omega_\mathbf{b} \tag{14}$$

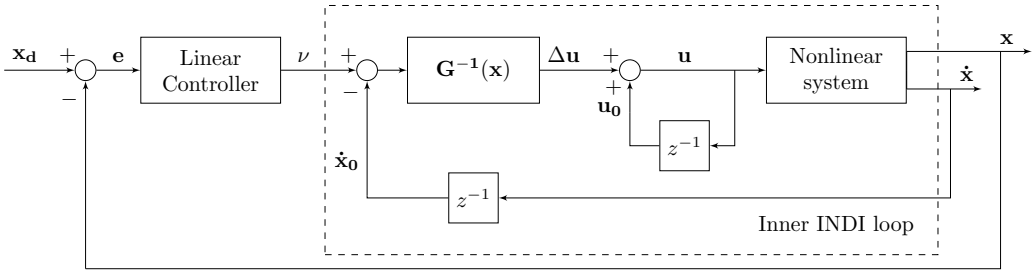

Figure 1: Tracking problem using INDI in the inner loop and linear control in the outer loop.

The moment components are given by expression (15). The x axis component originates a roll moment and the y axis a pitch moment. These two moments are proportional to the thrust forces applied in each propeller, with l being the arm for the moment. For the yaw moment, on the z axis, a relation between the force and the moment must be estimated. For now the proportionality constant will be named K_m. The forces given by the propellers are represented from F_1 until F_4.

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} l\left(-F_1 + F_2 + F_3 - F_4\right) \\ l\left(F_1 - F_2 + F_3 - F_4\right) \\ K_m\left(F_1 + F_2 - F_3 - F_4\right) \end{bmatrix} \tag{15}$$

The state vector $\mathbf{x}$ includes the body frame angular rates and the inertial vertical velocity ($\mathbf{x} = \begin{bmatrix} p & q & r & v_z \end{bmatrix}^T$). The derivative of this vector, using the equations of motion, can be expressed in the form of Equation (1a), with $\mathbf{f(x)}$ and $\mathbf{G(x)}$ given by expressions (16) and (17), respectively. The relation between the force on each propeller (F_i) and the correspondent motor input (T_i) is given by $F_i = K_T T_i$. The input vector $\mathbf{u}$ is composed by the motor inputs T_i.

$$\mathbf{f(x)} = \begin{bmatrix} -qr\dfrac{I_z - I_y}{I_x} \\ -pr\dfrac{I_x - I_z}{I_y} \\ -pq\dfrac{I_y - I_x}{I_z} \\ g \end{bmatrix} \tag{16}$$

4.2 *Controller Implementation*

The outer loop linear controller computes the virtual inputs using the control laws of Equation (18). If one considers small Euler angles approximation the roll and pitch laws correspond to second order low-pass filters and the yaw angle law corresponds to a first order low-pass filter. Good performance was achieved with $\alpha = 19.75\ [s^{-2}]$, $\beta = -8\ [s^{-1}]$ and $K_{\dot{\psi}} = K_{v_z} = 2\ [s^{-1}]$. The block diagram for the NDI controller is depicted in Figure 2.

$$\begin{bmatrix} \dot{p}_d \\ \dot{q}_d \\ \dot{r}_d \\ \dot{v}_{z_d} \end{bmatrix} = \underbrace{\begin{bmatrix} \alpha & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 \\ 0 & 0 & K_{\dot{\psi}} & 0 \\ 0 & 0 & 0 & K_{v_z} \end{bmatrix}}_{\mathbf{K_1}} \begin{bmatrix} \phi_d - \phi \\ \theta_d - \theta \\ \dot{\psi}_d - \dot{\psi} \\ v_{z_d} - v_z \end{bmatrix} + \underbrace{\begin{bmatrix} \beta & 0 \\ 0 & \beta \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\mathbf{K_2}} \begin{bmatrix} p \\ q \end{bmatrix} \tag{18}$$

One difference between INDI and NDI controllers in the inner loop consists in a replacement of $\mathbf{f(x)}$ for an estimate $\dot{\mathbf{x}}_\mathbf{0}$ of the NDI state derivative, estimated using a second order bandpass filter (BP from Equation (19)), that not only estimates the derivative, but also reduces the high frequency noise from the sensor measurements. This bandpass filter can be seen as a differentiation (s variable from Laplace domain in the numerator), followed by a low-pass filtering. Good estimates were achieved with $\omega_n = 200\pi\ [rad/s]$ and $\xi = 0.8$. To maintain the synchronization between all the variables used in the computation of the control laws a second order low-pass filter, with the same characteristics, was used for the remaining variables. The other difference is related with the control input ($\mathbf{u}$), now being an increment ($\mathbf{\Delta u}$) that needs to be added to the previous control input ($\mathbf{u_0}$).

$$\text{BP} = \frac{\omega_n^2 s}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{19}$$

The position control that provides the desired values for the roll angle, the pitch angle and the vertical velocity is adapted from Ardupilot code [14]. The yaw rate is provided directly by the pilot, being zero in autonomous flights. This position controller receives a desired 3D position, expressed in the inertial NED reference frame.

An adjustable input scaling gain η was used to reduce the disturbances of INDI controller. This gain varies between 0 and 1, and it is applied in the incremental control action $\mathbf{\Delta u}$. The input scaling gain scales the new incremental control action in the overall input $\mathbf{u}$, with $\eta = 1$ corresponding to a complete action and $\eta = 0$ no action at all. A low value for this parameter reduces the effect of disturbances because it minimizes control action to sudden changes in the derivative estimates. At the

$$\mathbf{G}(\mathbf{x}) = \begin{bmatrix} \frac{-lK_T}{I_x} & \frac{lK_T}{I_x} & \frac{lK_T}{I_x} & \frac{-lK_T}{I_x} \\ \frac{lK_T}{I_y} & -\frac{lK_T}{I_y} & \frac{lK_T}{I_y} & \frac{-lK_T}{I_y} \\ \frac{K_m K_T}{I_z} & \frac{K_m K_T}{I_z} & \frac{-K_m K_T}{I_z} & \frac{-K_m K_T}{I_z} \\ \frac{-\cos\phi\cos\theta}{m} K_T & \frac{-\cos\phi\cos\theta}{m} K_T & \frac{-\cos\phi\cos\theta}{m} K_T & \frac{-\cos\phi\cos\theta}{m} K_T \end{bmatrix} \tag{17}$$

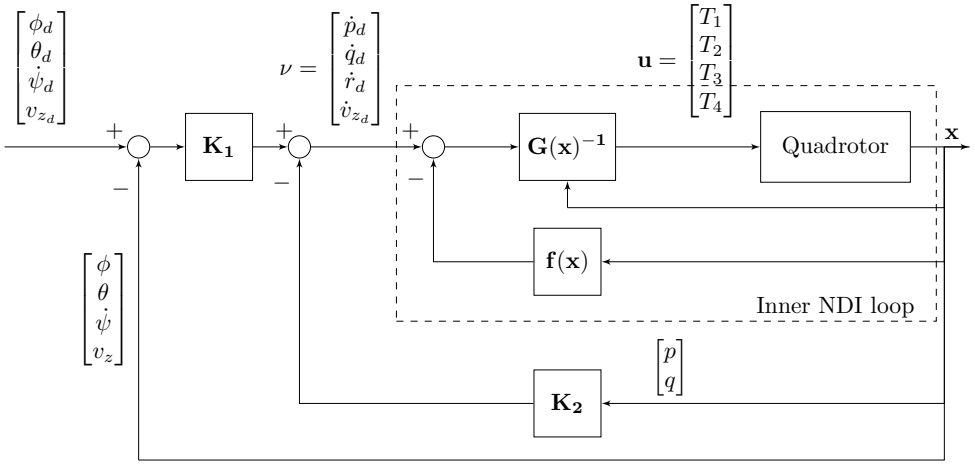

Figure 2: NDI controller block diagram.

same time control action saturations are also minimized. However, the tracking performance degrades for lower input scaling gain values, given that a perfect dynamic inversion is no longer achieved. The value $\eta = 0.3$ was selected for the simulation results, after performing a parametric analysis with several values for this parameter. For NDI and Ardupilot controllers a sampling time $T_s = 0.0025$ $[s]$ ($f_s = 400$ $[Hz]$) was selected. With INDI controller the sampling time was modified to $T_s = 0.02$ $[s]$ ($f_s = 50$ $[Hz]$), due to the high frequency noise that appears in the state vector derivative estimates. The sampling times were also selected using a parametric analysis.

5. Implementation and Results in MATLAB/Simulink

5.1 *MATLAB/Simulink Model*

The model of the Quadrotor that was used for all the simulations performed in MATLAB/Simulink simulation environment was developed by the supervisor of this thesis, and includes the following additional details, in order to provide a better approximation to the real Spyro quadrotor: a battery model; aerodynamic drag force; wind velocity input; motor and propeller models; magnetic field information.

The values for matrix $\mathbf{G}(\mathbf{x})$ were obtained using numerical differentiation, performed in MATLAB/Simulink. These values are kept constant during simulations, if small Euler angles are considered (neglecting $\cos\phi$ and $\cos\theta$ from expression (17)), and therefore they do not need to be estimated in real-time. For $\mathbf{f}(\mathbf{x})$ one has to compute its value in real-time, given that it is dependent from the body angular velocity components.

Since INDI controller is a sensor based solution, then it is necessary to provide a model of the sensors. In this case one included Band-Limited White Noise (BLWN) around the ideal values to estimate the measurements of the following sensors: gyroscope; accelerometer; barometer; magnetometer; GPS sensor; attitude (Euler angles) sensor.

5.2 *Simulation Results*

The simulations performed in MATLAB/Simulink for the attitude and vertical velocity control have a duration of 10 $[s]$, with different reference values for each control channel ($\phi = 10°$, $\theta = 20°$, $\dot{\psi} = 3°/s$ and $v_z = -4m/s$) in the interval $[2\ [s], 6\ [s]]$. In order to reduce the static error in the vertical velocity channel, an integral gain was introduced in the linear control law for this channel, with a value $K_i = 0.2$ $[s^{-2}]$.

The results for the simulation with the NDI controller can be seen in Figures 3 and 4.

Analyzing the simulation results one can conclude that the controller presents good performance for the attitude channels, given that all this desired references are achieved. The roll and pitch angle tracking is directly seen from the Euler angles graph, but the yaw rate tracking can also be seen in the same graph, even though it is in an indirect manner. A final yaw angle around 12.5 $[°]$ is reached in a four second reference duration, thus, the yaw rate is approximately the 3 $[°/s]$. The vertical velocity channel this time does not reach the asked -4 $[m/s]$ and when the reference returns to zero

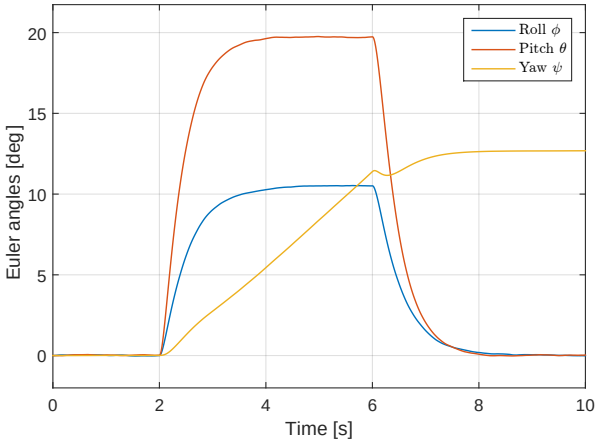

Figure 3: Euler angles results for simulation with NDI controller

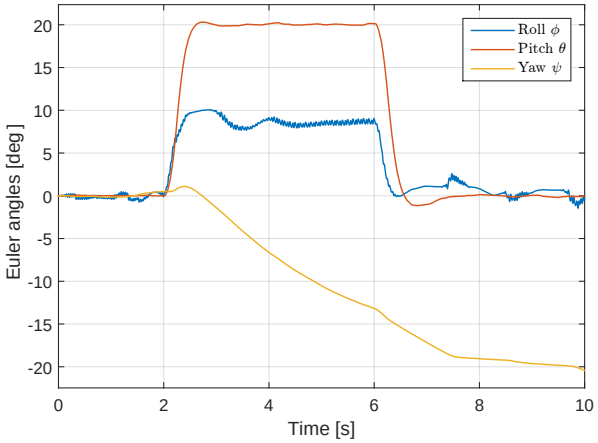

Figure 5: Euler angles results for simulation with Ardupilot controller

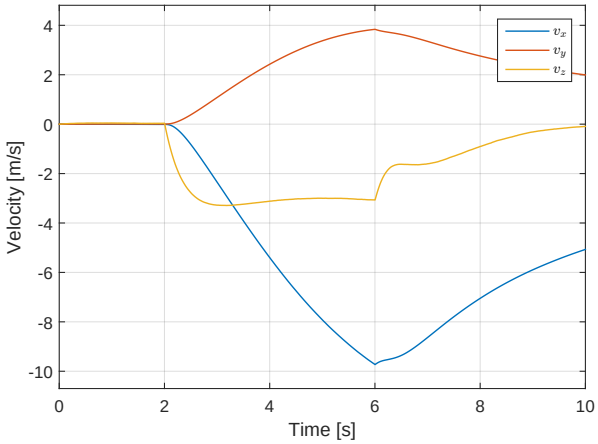

Figure 4: Inertial frame velocity results for simulation with NDI controller

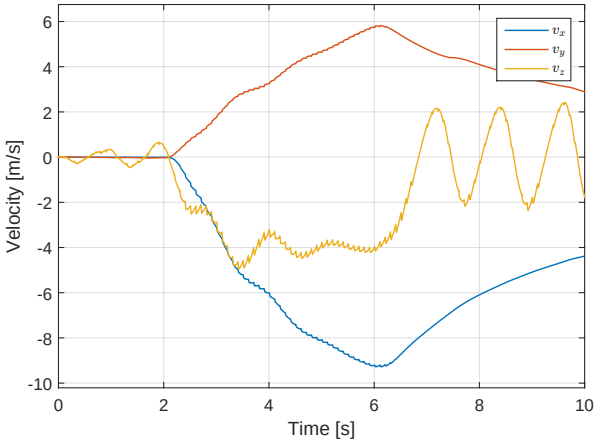

Figure 6: Inertial frame velocity results for simulation with Ardupilot controller

the vertical velocity still takes almost four seconds to achieve the same value. The previous behaviour occurs due to the integrative term because the integral must also converge. The reference tracking for this channel presents some degradation due to an imperfect dynamic inversion of the quadrotor model, which is a consequence of the disturbances and the noise modeled in the sensors, as well as the simplifications made to the model.

The results for the Ardupilot original controller are depicted in Figures 5 and 6. This control law is formed by several PID loops, thus, it presents a larger number of parameters that need to be adjusted during the design process.

For this controller, the tracking was no longer achieved for all the references, given that the yaw rate is no longer controlled. The roll angle reached the desired reference of 10 [°], but the effect of noise is clear, given that the curve has high frequency oscillations throughout the simulation. It also possesses small disturbances that deviate the curve from its reference value several times during the simulation. The pitch angle is the channel with better tracking, presenting only minor overshoots during the reference steps. Finally, the oscillations in the vertical velocity curve are obvious, being more pronounced after the reference steps (after the 6 [s]).

Comparing the simulations of both the original and NDI controllers one can observe that the latter is more robust to noise, given that it produces simulation results with less oscillations. Only the NDI controller guarantees tracking, with the original controller failing to reach the desired yaw rate.

The results with INDI controller are shown in Figures 7 and 8, where one can see that all the references are followed with good performance, in contrast to the NDI controller, where the desired vertical velocity reference was not followed. Therefore, INDI seems the most interesting technique to solve this quadrotor control problem.

To evaluate the robustness of the INDI controller one performed simulation tests including the most common disturbances in a quadrotor flight, namely, model uncertainties, wind velocity and loss of performance in a motor. The results to these tests

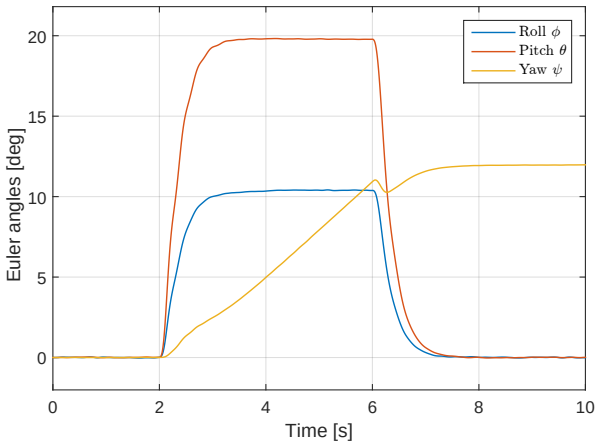

Figure 7: Euler angles results for simulation with INDI controller

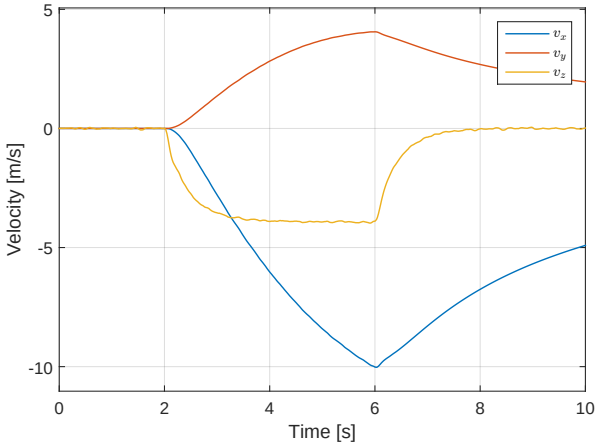

Figure 8: Inertial frame velocity results for simulation with INDI controller

show that the controller still achieves good performance with some degree of uncertainties in the values of $\mathbf{G}(\mathbf{x})$ matrix, which can be relevant to keep a stable and efficient flight when there are variations in mass, moments of inertia or other geometric parameters, allowing one to use the same controller with different quadrotor variants, without adjusting any parameter. Wind is usually present in outdoor flights, being a common source of disturbances, thus it is interesting to test the robustness of the control system to this type of disturbances. The wind is simulated using a specific input of the MATLAB/Simulink model. The results show that the performance only starts to degrade when the wind has a magnitude of 5 $[m/s]$ in each direction (along x, y and z axes directions). Besides model uncertainties and wind velocity, disturbances can also appear due to the loss of of effectiveness in the actuators. To simulate this effect the power provided to a motor is decreased in a varying percentage, from 0% until 40%, relative to the nominal value provided by the INDI control algorithm. The performance of the controller starts to degrade when the reduction in power is higher than 20% (i.e., 80% of the original power).

6. Implementation and Results in Ardupilot

The INDI control algorithm was integrated with the position control of Ardupilot in the automatic flight modes, except for takeoff and landing phases, since they have a considerably different control algorithm that cannot be merged with INDI. The control algorithm was implemented using C++ programming language, with the bandpass filter, to estimate the state derivatives, being digitally implemented using second order regressive finite differences [15].

6.1 *Simulation Results*

An eight shape tracking was performed for the INDI and original Ardupilot controllers, using the simulation platform (Software in the Loop) provided in Ardupilot. The quadrotor starts at a home position, climbs until an altitude $h = 50$ $[m]$, follows an eight shape, by tracking several consecutive waypoints, and finally returns to the home position. Complete simulation results, including 2D horizontal path, altitude, Euler angles and inertial frame vertical velocity are depicted in Figures 9, 11 13, 15 and 17 for the INDI controller. The same information is shown in Figures 10, 14, 16, 18 and 20 for the Ardupilot controller .

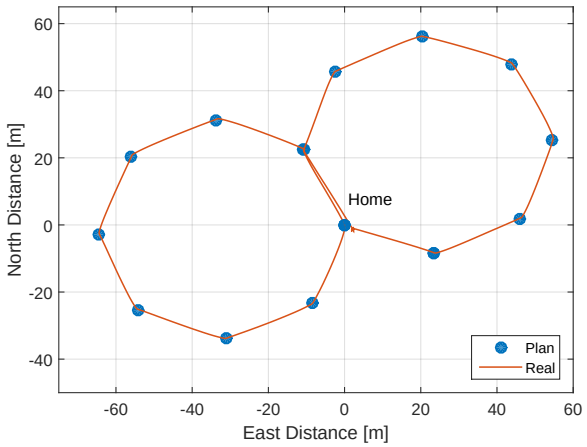

Figure 9: Position graph of eight shape tracking with INDI controller.

Analyzing the graphical representations of the 2D horizontal paths one can say that the tracking of the waypoints is successfully achieved with both controllers.

By the analysis of the altitude graphs around cruise altitude it is possible to say that both controllers present similar disturbances in terms of altitude, therefore they should be related with the vertical position controller, given that it is the same for both control designs. The INDI controller presents an advantage, namely, the altitude profile presents smoother transitions, i.e. the little peaks that are

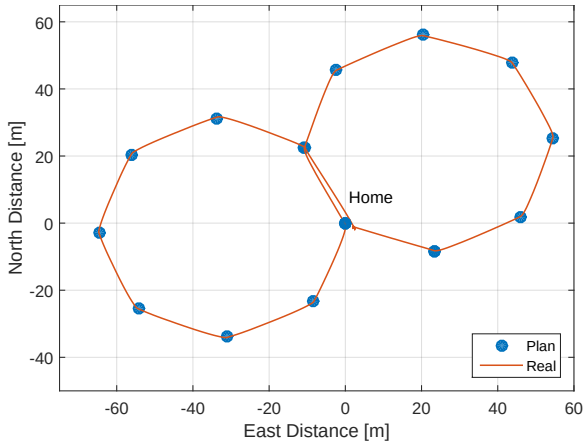

Figure 10: Position graph of eight shape tracking with Ardupilot controller.

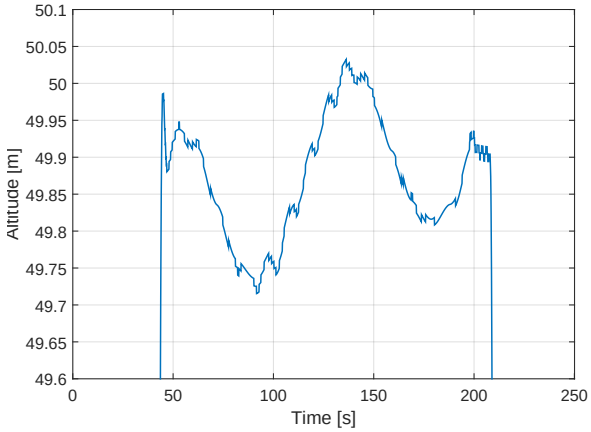

Figure 11: Altitude of eight shape tracking with INDI controller.

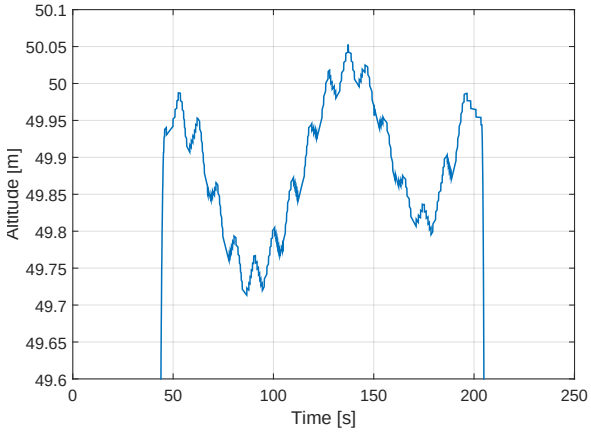

Figure 12: Altitude of eight shape tracking with Ardupilot controller.

observable in both graphs are smaller for the INDI controller. The more pronounced peaks in the altitude profile of the original controller will lead to a higher battery consumption.

Roll and pitch angles of both controllers do not match due to a different control approach in the tracking of a waypoint. The original controller steers the quadrotor front side to the waypoint, thus

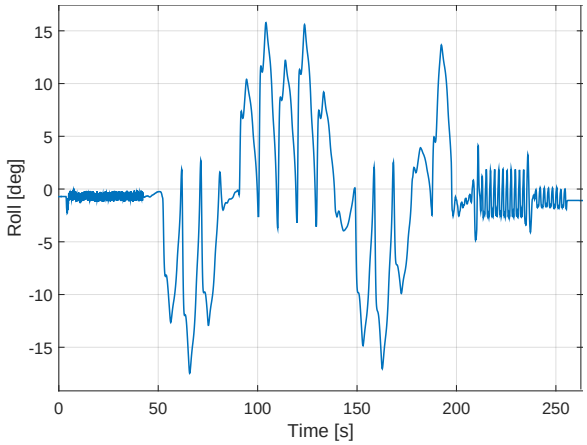

Figure 13: Roll angle of eight shape tracking with INDI controller.

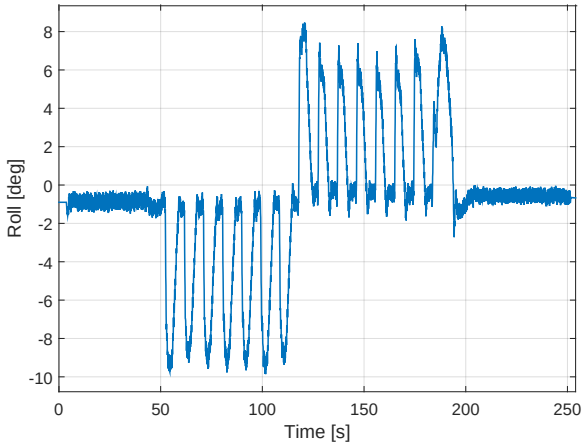

Figure 14: Roll angle of eight shape tracking with Ardupilot controller.

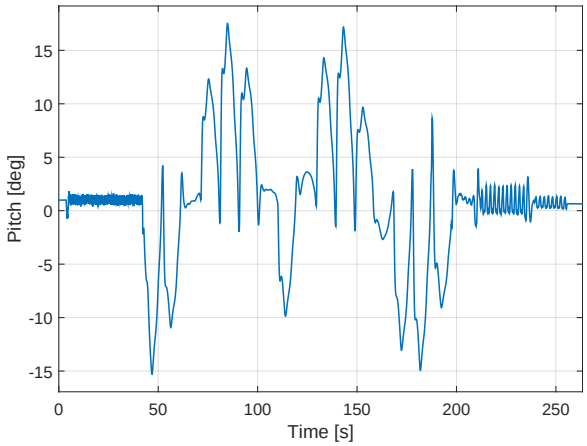

Figure 15: Pitch angle of eight shape tracking with INDI controller.

the initial situation in any waypoint is the same, which leads to the same peaks of roll and pitch angles for all waypoints, apart from an inversion in roll peaks at the middle of the simulation. This is caused by the inversion of turning side in the second octagon. INDI controller does not steer the quadrotor towards the waypoint, thus the roll and

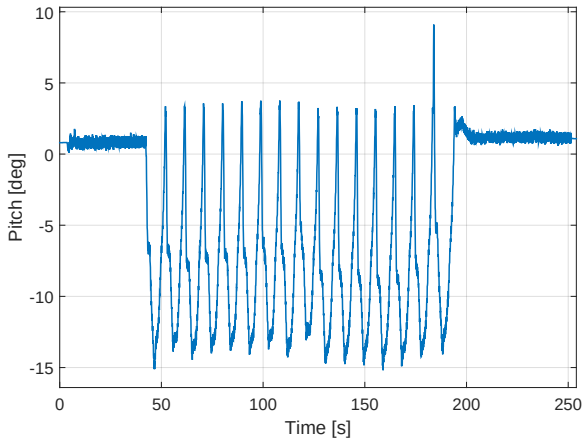

Figure 16: Pitch angle of eight shape tracking with Ardupilot controller.

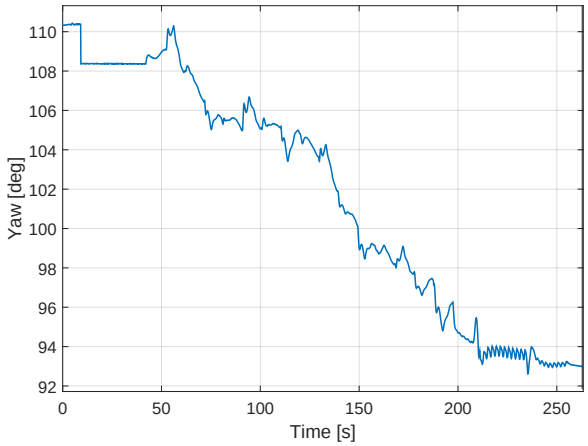

Figure 17: Yaw angle of eight shape tracking with INDI controller.

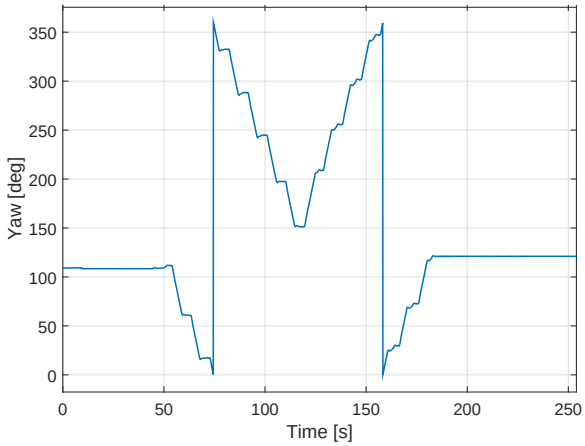

Figure 18: Yaw angle of eight shape tracking with Ardupilot controller.

pitch peaks are different for all waypoints. In terms of amplitude the roll peaks are higher with INDI controller (15 [°] against 10 [°]), being both around 15 [°] for the pitch angle. With regard to the existence of high frequency oscillations they are similar in the climb phase (i.e. in the first 45 [s] when both angles are close to zero) because the controller is

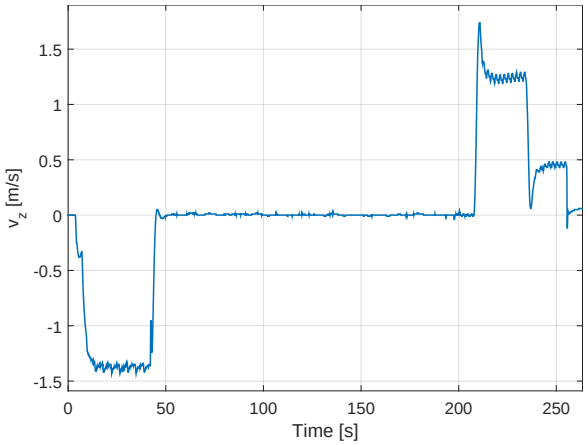

Figure 19: Vertical velocity of eight shape tracking with INDI controller.

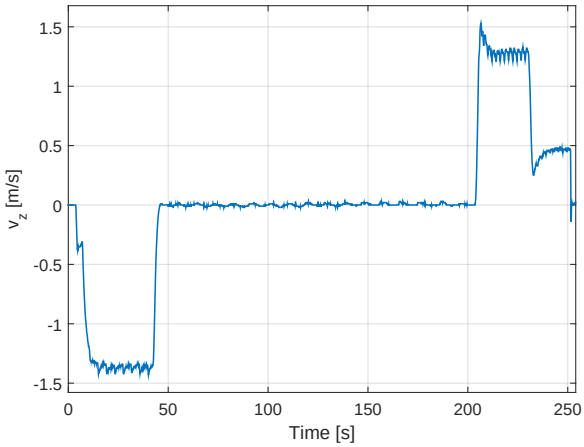

Figure 20: Vertical velocity of eight shape tracking with Ardupilot controller.

the same, only changing to INDI version when it reaches the cruise altitude of 50 [m]. During cruise altitude flight the original controller presents more oscillations than the new INDI controller. In the descent phase the INDI controller presents oscillations with more amplitude, but the Ardupilot controller has oscillations with higher frequency.

Due to the different approach on following a waypoint the yaw angles are also different. In the original controller its value is controlled, in order to steer towards the waypoint, decreasing when the quadrotor is turning left and increasing in the second octagon, when the quadrotor turns to the right side. The two leaps in the yaw angle happen so that yaw angle is always given in interval [0° , 360°]. With INDI the yaw angle is not controlled, being free to follow a yaw rate request by the pilot. The roll and pitch angles that are necessary to follow the waypoints originate small changes in yaw during the simulation. Finally, the vertical velocity curves are very similar, not presenting noticeable differences.

In conclusion, both controllers present very similar results, with INDI presenting smaller oscillations in some variables. The original controller also

has the disadvantage of requiring more time to tune the large quantity of gains that exist in the several PID loops. The model in Ardupilot simulation platform presents a different dynamics than the Spyro model used to design the INDI controller, and this controller still reaches good tracking performance, which proves the robustness of this control algorithm to model changes.

7. Conclusions

Both methodologies, i.e. NDI and INDI, were successfully implemented in MATLAB/Simulink, proving its validity as solutions for an attitude and vertical velocity controller for a quadrotor UAV. These new methods could be run at a smaller frequency than the original PID control design from Ardupilot, thus requiring a smaller computational power. An efficient estimation for the state derivative in INDI controller was implemented using a second order bandpass filter, which not only estimates the derivative but also filters some high frequency noise from sensor measurements. The latter was also implemented with success in Ardupilot platform, a professional grade autopilot system, showing that the INDI controller can match the performance of the original control design, which proves the feasibility of this new method in quadrotor control design, with the advantage of consuming less battery. The model of the quadrotor that is present in Ardupilot simulation platform is different from the Spyro model for which the controllers were developed, therefore a good performance of the controller proves the robustness to model uncertainties.

The developed code in Ardupilot is ready to be used in experimental tests, one only needs to upload the code to an autopilot board. An experimental validation using a real Spyro quadrotor was scheduled with UAVision, but unfortunately they were not available to perform the experimental tests before the delivery deadline of this document.

7.1 *Future Work*

As future work one can include an adaptive algorithm that would estimate the control effectiveness matrix values in real time, which could account for some stationary disturbances affecting the quadrotor. With this algorithm the values for $\mathbf{G}(\mathbf{x})$ are no longer necessary to be estimated, turning the INDI control version completely model free, so that it could be, in theory, applied to any quadrotor. The INDI controller can also be developed for takeoff and landing phases. Even though MATLAB/Simulink models a great part of the physical dynamics of the quadrotor, as well as the noise obtained from sensor measurements, a real time application with experimental tests would have provided results that could verify the level of robustness of the chosen control method to a variety of disturbances that were not modeled, e.g. loss of performance with battery consumption.

References

[1] Oosterom et al. Design of a gain-scheduling mechanism for flight control laws by fuzzy clustering. *Control Engineering Practice*, 14(7): 769–781, 2006.

[2] Alaeddin Bani Milhim. Modeling and fault tolerant pid control of a quad-rotor uav. *Master's Thesis, Concordia University, Montreal, Quebec, Canada*, 2010.

[3] Arthur J Krener. A decomposition theory for differentiable systems. *IFAC Proceedings Volumes*, 8(1):280–289, 1975.

[4] Lombaerts et al. Flight control reconfiguration based on online physical model identification and nonlinear dynamic inversion. In *Fault tolerant flight control*. Springer, 2010.

[5] Kanellakopoulos et al. Systematic design of adaptive controllers for feedback linearizable systems. In *American Control Conference, 1991*, pages 649–654. IEEE, 1991.

[6] Farrell et al. Command filtered backstepping. *IEEE Transactions on Automatic Control*, 54 (6):1391–1395, 2009.

[7] Mercado et al. Quadrotors flight formation control using a leader-follower approach. In *Control Conference (ECC), 2013 European*, pages 3858–3863. IEEE, 2013.

[8] R. Lopez et al. Altitude control of a quadrotor using adaptive sliding mode. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016.

[9] Pedro Valério Menino Simplício. *Helicopter Nonlinear Flight Control*. Master's thesis, Instituto Superior Técnico - Universidade de Lisboa, October 2011.

[10] Smeur et al. Adaptive incremental nonlinear dynamic inversion for attitude control of micro air vehicles. *Journal of Guidance, Control, and Dynamics*, 2015.

[11] Acquatella et al. Incremental backstepping for robust nonlinear flight control. *Proceedings of the EuroGNC 2013*, 2013.

[12] Jean Jacques E. Slotine and Weiping Li. *Applied nonlinear control*. Prentice hall, 1991.

[13] Eduardo Simoes da Silva. *Incremental Nonlinear Dynamic Inversion for Quadrotor Control*. Master's thesis, Instituto Superior Técnico - Universidade de Lisboa, November 2015.

[14] Ardupilot. Ardupilot development site. `http://ardupilot.org/dev/index.html`, 2017. Accessed: 2017-09-25.

[15] Joao Folgado. Computational mathematics lecture notes. Instituto Superior Técnico - Universidade de Lisboa, 2013.