Politecnico di Milano
School of Industrial and Information Engineering

Master of Science in

Automation and Control Engineering

Guidance and control for a fixed-wing UAV

Advisor: Prof. Marco LOVERA
Co-Advisors: Eng. Mattia GIURATO
Dr. Simone BALDI (TU Delft)

Thesis by:
Stefano FARI' Matr. 850272

Academic Year 2016–2017

Ai miei genitori.

Acknowledgments

Quando ripenso sentiero che ho percorso durante lo svolgimento di questa tesi, so che sono tante le persone che sento di dover ringraziare, che mi hanno aiutato e sostenuto, sempre. Il mio primo pensiero va a al Prof. Marco Lovera che durante la mia trasferta a Delft non mi ha mai fatto mancare il suo sostegno, sia professonale che morale, e mi ha consentito di portare a termine questa sfida. Qualunque parola possa spendere in questo spazio non basterebbe a dire grazie a Mattia. Senza il tuo aiuto, caparbietà, e generosità non ce l'avrei mai fatta, sei stato un faro nei momenti più bui. Un sentito grazie va al Dr. Simone Baldi per avermi dato la possibilità di affrontare una tematica a me tanto cara che mi ha dato la possibilità di crescere tanto in un nuovo contesto universitario. Chiaramente un abbraccio va a tutti quegli Amici su cui ho sempre potuto fare affidamento e con cui ho condiviso parte delle mie avventure: dalla mia copilota Basak, a Davide, al Rafo che mi ha sopportato per mesi, Giorgio, Ludo, Michele, Gianlu, Dani, che con la vostra presenza mi avete sempre spronato a dare il meglio. Un grazie al Marra (:€), che come sempre mi ha ricaricato con l'allegria ed entusiasmo giusto nei momenti più tortuosi. In questi mesi ho avuto la fortuna di avere accanto una persona unica, che mi ha sempre sopportato e supportato, con il suo infinto affetto, grinta, sorriso e pazienza... grazie Chiara.

Delle parole speciali vanno spese per i miei genitori e la mia famiglia. Avere avuto voi tutti alle spalle è un dono che porterò sempre con me.

Abstract

Nowadays, we often hear about autonomous Unmanned Aerial Vehicles because of their growing diffusion and ability to adapt to multiple application contexts. When speaking of UAVs, we refer to a category of small/medium-sized aerial vehicles that can fly autonomously. A category of great interest is that of fixed-wing UAVs, due to its inherent energy efficiency in performing medium/long range missions and the ability to carry payloads. For this reason, a fervent interest is emerging around it, bringing together communities from different contexts, from recreational to academic, from commercial to military. The common factor for any autonomous implementation is the UAV's ability to move in space with great precision toward given geometric references. For an aircraft facing an outdoor activity, the challenge is made even harder by the constant presence of wind. Making a UAV autonomous requires two main control layers: the autopilot, able to maintain attitude, height, airspeed, and course; the path following, able to maintain the aircraft on a given inertial reference. The research has therefore moved in the years towards the construction of increasingly robust path following laws. One of the most performing is known as Vector Field method. Recently, a research proposed an variant of the Vector Field approach using an adaptation strategy that showed promising results on simulations with a simplified dynamics models of fixed-wing UAVs.

This thesis wants to make possible an evaluation of the new Vector Field strategy using a complete aircraft model based on a real integrated platform in view of future experimentations. Therefore, the aircraft dynamics was deeply studied and its complete nonlinear dynamics equations were implemented on a Simulink simulator. Moreover, the hardware and software integration of a fixed-wing UAV was tackled. Every components, from the airframe to electronics were selected according to specific requirements, and the physical and aerodynamics parameters of the aircraft has been then estimated.

The brain of each autonomous aerial vehicle is the Flight Control Unit (FCU). The chosen FCU software is ArduPilot, an open-source flight software suite, which gives two fundamental benefits: it is possible to implement easily the guidance strategy slightly modifying an already complete software; it implements the autopilot layer, hence the low-level controllers.

In this way, after an appropriate analysis, the autopilot architecture was

reverse-engineered and included onto the Simulink simulator. This means having a simulator that runs the same core of the ArduPilot low-level control system.

The adaptive Vector Field strategy has been analysed and properly implemented in the simulator: finally the performance of the approach are presented and properly evaluated.

Sommario

Al giorno d'oggi, sentiamo spesso parlare di Aeromobili a Pilotaggio Remoto per via della loro crescente diffusione e capacità di adattarsi a molteplici contesti applicativi. Quando si parla di APR, in inglese UAV, in genere si fa riferimento ad una categoria di velivoli di piccola/media dimensione in grado di volare pilotati da remoto. Quando il velivolo può affrontare una missione autonoma, solitamente viene chiamato drone.

Una categoria di grande interesse è quella delle piattaforme ad ala fissa, per la loro innata efficienza in termini energetici nel poter svolgere missioni a medio-lungo raggio e per la capacità di poter trasportare del carico utile. Per questo motivo, intorno agli essa si sta sviluppando un fervente interesse che raccoglie comunità provenienti da diversi contesti, da quello ricreazionale a quello accademico, da quello commerciale a quello militare. Il minimo comune multiplo per qualunque implementazione autonoma, è la capacità dello UAV di potersi spostare nello spazio con grande precisione seguendo dei riferimenti geometrici assoluti. Per un sistema ad ala fissa, o aereo, la sfida è resa ancora più difficile dalla costante presenza del vento. La ricerca pertanto si è mossa negli anni verso la costruzione di controllori e leggi di navigazione sempre più robusti. Nella ricerca, gran parte delle leggi di guida per UAV vengono proposte inizialmente utilizzando modelli dinamici semplificati. Lo stesso vale per un approccio per l'inseguimento di riferimenti rettilinei ed orbite (path following) noto come Vector Field, una cui variante adattiva è stata recentemente sviluppata presso il DCSC di TU Delft.

Lo scopo della tesi è quello valutare le performance di tale controllo sulla dinamica completa di un aereo per prepararne l'implementazione su un vero UAV ad alla fissa. Per questo motivo, è stata affrontata l'integrazione di un UAV in tutti i suoi aspetti, dalla scelta dell'aereo a quella dell'elettronica, fino al software. Il software di volo scelto è ArduPilot,il cui codice open-source dà la possibilità di implementare facilmente nuovi controllori di path following, facendo uso di alcuni strumenti già integrati quali EKF, librerie per i sensori, o meccanismi di failsafe. Dopo un'apposita stima dei parametri fisici ed aerodinamici, il modello dinamico completo a 6-DOF dell'aereo è stato quindi implementato nell'ambiente di sviluppo Matlab/Simulink.

Solitamente le leggi di path following, come il Vector Field, rappresentano solamente lo strato di controllo più esterno. Sono necessari infatti dei controllori

di basso livello, dediti a regolare l'assetto dell'aereo, la sua quota, velocità e rotta. Per questo motivo, dopo aver descritto la struttura generale di controllo di basso livello, la tesi si occupa anche di analizzare l'effettiva struttura di controllo di ArduPilot ed implementarla all'interno del simulatore Simulink.

In questo modo si è ottenuto un simulatore completo con cui fosse possibile ottenere simulazioni accurate del sistema completo aereo-Ardupilot e su cui poter applicare e valutare l'approccio del Vector Field path following adattivo.

Contents

List of Figures

List of Tables

Introduction

Overview of fixed-wing UAVs

In the last few decades, a fervent interest has raised around the so-called Unmanned Aerial Vehicles. The UAVs are, as often happens for new aerospace technologies, a reality that firstly came up in the military field and then found civil applications. However, a huge evolution followed the miniaturization of the electronics and the falling of its costs, giving the possibility to group of people from companies, research institutes, or simply hobbyist, to find civil applications.

The UAVs definition includes a variety of different type of vehicles and sizes. Most common are fixed-wing vehicles and multi-rotors vehicles; this thesis is focused on the former category. In particular, this work puts the attention on the miniature UAVs or small UAVs (SUAVs), which are aerial vehicle small enough to be man-portable, like the one in Figure 2.

Figure 1: Fixed-wing UAV used for military surveillance

More in detail, a fixed-wing UAV is an aerial vehicle which is capable of flight using wings. Fixed-wing aircraft are distinct from rotary-wing aircraft, in which the wings form a rotor mounted on a spinning shaft, and from ornithopters, in which the wings flap in similar manner to a bird. The wings are able to provide

Figure 2: The HobbyKing Bix3 during the landing stage

the lift force to win the gravitational force thanks to their shape and the forward vehicle speed. The manoeuvrability of an aircraft is ensured by some control surfaces, that are able to locally modify the air-flux and to provide the necessary moments to change the body attitude. Normally they have three control surfaces, the ailerons (at the wings edges), the elevator and the rudder (at the horizontal and vertical tails edges respectively) that influence the attitude, thus the aircraft motion, jointly with a device able to generate a thrust force, such as a propeller.

A fixed-wing vehicle becomes a drone, if it is able to complete a totally autonomous flight. As such, it is necessary to equip it with a brain, the Flight Control Unit. Its architecture follows the one in Figure 3 and relies on multiple layers:

- The block called **Autopilot** refers to the low-level control algorithms that is able to maintain roll and pitch angles, airspeed, altitude, and course heading;
- All levels rely on accurate UAV state estimates obtained by dynamically filtering the onboard sensors, which includes accelerometers, gyroscopes, barometers, magnetometers, GPS receivers and airspeed sensors. This task is performed by the **State estimator**;
- UAVs must be able to maneuver effectively in wind; the **Path following** layer is meant to maintain the vehicle on the desired path;
- The **Path manager** is what supervises the navigation of the UAV with a finite-state machine which has to converts a sequence of waypoints into a sequence of path primitives that the Path following can track;
- Finally, the **Path planner** is in charge of producing the path definitions for the Path following layer using point-to-point algorithms, where the object

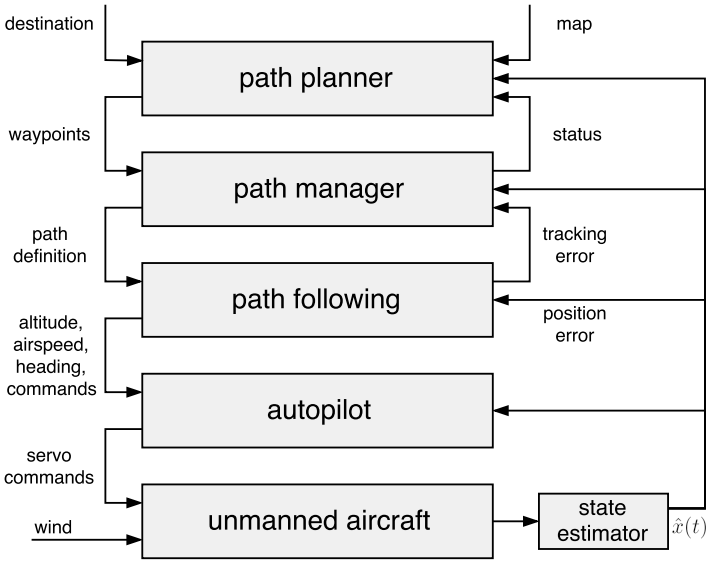


Figure 3: Autonomous UAV overall control system architecture

is to navigate from a start position to an end position, possibly avoiding obstacles, or to cover a specific area as well.

These platforms are gaining more and more interest in the academic and research fields because they allow the development and testing of new control algorithms, navigation systems and sensor-fusion algorithms. Civil and commercial applications are not as developed as the military counterpart, although potential applications are extremely broad in scope, including:

- environmental monitoring;
- inspection or surveillance;
- homeland security;
- aerial mapping;
- search and rescue operations;
- cargo deliveries;
- first aid and remote medical assistance.

To extend the usefulness of fixed-wing UAVs beyond their current applications, the capability to plan paths and to follow them accurately is of great importance. Unlike piloted vehicles, which rely on the pilot to navigate over demanding terrain or to avoid obstructions, UAVs rely on automation to provide this functionality.

As applications such as urban surveillance and rural search and rescue require UAVs to fly down city streets surrounded by buildings or near the surface of abruptly changing mountainous terrain, the ability to follow preplanned paths with precision is essential. For missions involving cooperation among a team of UAVs, precise path tracking is crucial to achieving the cooperation objective.

For small aerial vehicles (SUAVs), such as those of primary interest in this study, wind disturbances, dynamic characteristics, and the quality of sensing and control all limit the achievable tracking precision. For UAVs, wind speeds are commonly 20%–50% of the desired airspeed. Effective path tracking strategies must overcome the effect of this ever present disturbance. For most fixed-wing UAVs, the minimum turn radius is in the range of 10–50 m. This places a fundamental limit on the spatial frequency of paths that can be tracked. Thus, it is important that the path-tracking algorithms utilize the full capability of the UAV.

Thesis description

Recently, many path following strategies have been proposed. One of these is the so-called Vector Field approach, that in many research studies showed very effective tracking performance. An improvement of the method is proposed by Dr. S. Baldi (DCSC, TU Delft) using an adaptive term able to estimate the aircraft ground speed which is often unknown in presence of wind [1]. The guidance laws are usually designed assuming the simplest dynamics model, as in the case of the just mentioned Vector Field variant. This is favourable to put in evidence their capabilities, but does not provide a suitable insight of the path tracking performances of a real aircraft in advance.

Given the above considerations, the purpose of this thesis is to constitute a starting point for the new path following evaluation and implementation on a real platform. This is done tackling many aspects, which are:

- **Fixed-wind UAV modelling**. Many flight mechanics aspects are presented to obtain the equations of motion and forces and moments expressions, explaining how the dynamics can be influenced from the environmental conditions. The nonlinear model is implemented on a complete simulator in the Simulink environment. It also includes useful visual instruments for a run time analysis.

- **Hardware and software integration** of a real UAV. In order to provide the capability to practically test the guidance algorithm in the future, a small fixed-wing aircraft has been built from scratch. This required the selection of suitable electronic components which will constitute the aircraft payload. The UAV is then assembled and integrated with a Flight Control Unit.

Among the possible options, the flight software choice fell on ArduPilot, a professional grade open-source autopilot software suite.

- As seen, the path following algorithms lay on the fundamental autopilot layer. For this reason, the study ArduPilot **autopilot control architecture** has been done via reverse-engineering of its code and implemented in the Simulink simulator. To better understand its logic, it is necessary to present how an autopilot control structure can be designed and what assumptions are made behind some control choices.

- The **Vector Field path following** principles are presented, analysing both standard and adaptive variants. The adaptive Vector Field strategy, whose performance has been only verified using a simplified point mass dynamics, is implemented in the full simulator. The simulation results of the technique applied to the UAV dynamics are shown and evaluated in comparison with the standard VF approach.

ArduPilot open-source feature is fundamental, because gives two main benefits. Firstly, it allows the future integration of the new guidance law, using a reliable architecture which results from years of developments among independent developers, and, at the same moment, taking advantage of several features such as built-in sensors drivers, advanced failsafe algorithms, or multiple flight modes operations. Moreover, accessing the code makes possible the replication of the autopilot layer on the Simulink simulator, key aspect to provide the Simulink simulator with closer results to the real implementation. The thesis work has been carried out both at TU Delft and Politecnico di Milano.

Thesis structure

Chapter 1 starts with the formal presentation of the multiple coordinate frames necessary to understand how quantities are referenced. The 6-DOF kinematics and and rigid body dynamics equations are obtained. Afterwards, the equations of the forces and moments are described, together with the wind model.

Chapter 2 faces the main requirements for the airframe, and choice of the electronic equipment and Flight Control Unit. A description of every components used in the UAV follows. Then, the unknown model parameters of the aforementioned system are estimated, and the software Digital DATCOM is used to obtain an estimate of the aerodynamic coefficients.

Chapter 3 firstly presents the general autopilot architecture. This is necessary to introduce the linearized model transfer function and give the basis to understand how the problem of controlling the aircraft MIMO system can be faced. Next to this, the ArduPilot control scheme is presented, *i.e.*, the roll, pitch and side-slip control schemes, together with the Total Energy Control System, devoted to the

height and airspeed control. Each loop makes use of a PID controller, therefore, before proceeding, they have to be tuned. The description of the experimental tuning procedure is then presented.

Chapter 4 describes the two Vector Field path following variants. Both are formulated using a reduced order model with a simplified course dynamics; the course control loop is then synthesized. The Vector Field guidance is shown for two path primitives, the straight-line and the orbit. With their composition one can build up more involved paths. After that, the adaptive modification proposed in [1] is described and the Simulink implementation is shown.

Chapter 5 presents and evaluates the Vector Field simulation results under four specific environmental condition, both for the standard and adaptive VF variants. In this way it is possible to have a direct feedback about the improvements of the new method.

Chapter 1

Fixed-wing UAV modelling

In this chapter the problem of modelling the dynamics of a fixed-wing aircraft is presented and evaluated. When studying aerial vehicles, several coordinate frames are needed for a correct modelling and phenomena understanding. Therefore, the rotation formalism and the coordinate systems used to describe the position and orientation of the aircraft with its sensors are presented (Sections 1.1 and 1.2). A key aspect is then tackled, which is the wind triangle description, since it determines how the aerodynamics and wind disturbances influence the rigid-body motion (Section 1.3). With this background, the equations of motion for the UAV can be derived (Section 1.4) and the external forces and moments expressions are shown in Section 1.5. Finally, the atmospheric model is briefly introduced in Section 1.6 and the Simulink simulator is shown at end of this chapter in Section 1.7.

1.1 Rotation formulas

We begin considering the coordinate frames depicted in Figure 1.1. Vector $\mathbf{p}$ can be expressed in both the frame $\mathcal{F}^0$ (specified by $[\mathbf{i}^0, \mathbf{j}^0, \mathbf{k}^0]$) and in the $\mathcal{F}^1$ frame (specified by $[\mathbf{i}^1, \mathbf{j}^1, \mathbf{k}^1]$). The vector sets $[\mathbf{i}^0, \mathbf{j}^0, \mathbf{k}^0]$ and $[\mathbf{i}^1, \mathbf{j}^1, \mathbf{k}^1]$ are each mutually orthogonal sets of unit basis vectors.

Adopting the to-from notation, the rotation matrix from $\mathcal{F}^0$ to the coordinate system $\mathcal{F}^1$ is named as $\mathcal{R}_0^1$. A generic vector $\mathbf{p}^0$, can be resolved in system $\mathcal{F}^1$ through the matrix operation:

$$\mathbf{p}^1 = \mathcal{R}_0^1 \mathbf{p}^0. \tag{1.1}$$

As an example, from the geometry in Figure 1.1, the right-handed rotation about the $\mathbf{k}^0$ axis of an angle θ is defined as:

$$\mathcal{R}_0^1 \triangleq \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.2}$$

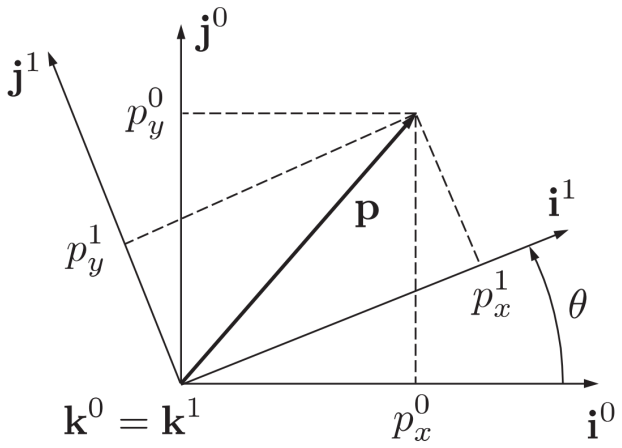


Figure 1.1: Rotation in a 2D view (**k** axes point outwards)

Proceeding in a similar way, the right-handed rotation about the $\mathbf{j}^0$ axis would have given

$$\mathcal{R}_0^1 \triangleq \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{1.3}$$

while a rotation about the axis $\mathbf{i}^0$ would be

$$\mathcal{R}_0^1 \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}. \tag{1.4}$$

The matrix $\mathcal{R}_0^1$ is an example of the class of orthonormal rotation matrices which have the following properties:

1. $(\mathcal{R}_0^1)^{-1} = (\mathcal{R}_0^1)^T = \mathcal{R}_1^0$;
2. $\mathcal{R}_1^2\mathcal{R}_0^1 = \mathcal{R}_0^2$;
3. $\det(\mathcal{R}_0^1) = 1$.

1.2 Coordinate frames

To derive and understand the dynamics of UAVs, several coordinate systems are of interest. The reasons for using different coordinate systems are:

- Newton's equations of motion are derived relatively to an inertial reference frame. However, motion is most easily described in a body-fixed frame.

- Aerodynamic forces and moments act on the aircraft body and are most easily described in a body/fixed reference frame.
- On-board sensors, like accelerometers and rate gyros measure information with respect to the body frame. Alternatively, GPS measures position, ground speed and course relatively to the inertial frame.
- Mission requirements, like loiter points and flight trajectories, are specified in the inertial frame.

1.2.1 The inertial frame $\mathcal{F}^i$

The inertial coordinate system is an earth-fixed frame with its origin at the defined home location. As shown in the bottom left of Figure 1.2, the unit vector $\mathbf{i}^i$ is directed north, $\mathbf{j}^i$ is directed east, and $\mathbf{k}^i$ is directed towards the center of Earth, or down. This is commonly referred as the NED (north-east-down) reference frame.

1.2.2 The vehicle frame $\mathcal{F}^v$

The vehicle frame simply represent the inertial frame translated onto the center of mass of the vehicle. The axes of $\mathcal{F}^v$ are pointed in the same directions of $\mathcal{F}^i$ (Figure 1.2).

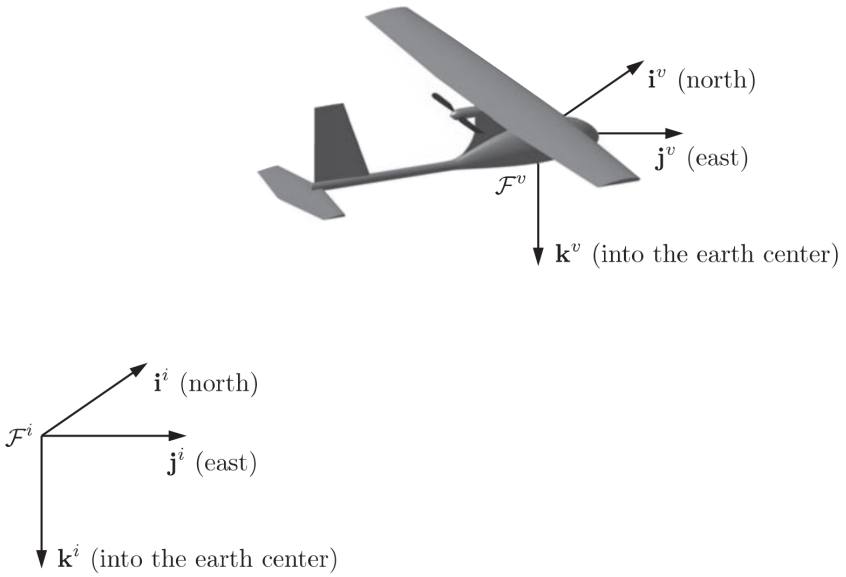
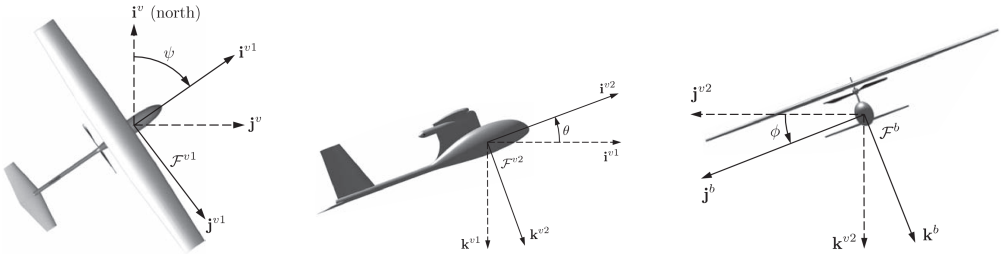


Figure 1.2: The inertial frame $\mathcal{F}^{\mathbf{i}}$ at the bottom left and the vehicle frame $\mathcal{F}^v$ placed at the UAV center of mass.

As will be shown, to define the body frame one encounters specific rotations using angles named ϕ, θ, and ψ, about the $\mathbf{k}$ axis, the new $\mathbf{j}$ axis and the newer

i axis respectively. These angles are the so-called Euler angles, and their naming convention follows NASA standard notation.

1.2.3 The body frame $\mathcal{F}^b$



(a) From $\mathcal{F}^v$ to $\mathcal{F}^{v1}$ (b) From $\mathcal{F}^{v1}$ to $\mathcal{F}^{v2}$ (c) From $\mathcal{F}^{v2}$ to $\mathcal{F}^b$

Figure 1.3: Frames definition by three rotations of ψ, θ and ϕ

To go from the vehicle frame to the body frame, firstly define the $\mathcal{F}^{v1}$ frame as:

$$\mathbf{p}^{v1} = \mathcal{R}_v^{v1}(\psi)\mathbf{p}^v \tag{1.5}$$

where $\mathcal{R}_v^{v1}(\psi)$ is the right-handed rotation matrix about the $\mathbf{k}^v$ axis by the heading, or yaw angle ψ. $\mathcal{F}^{v1}$ has its $\mathbf{i}^{v1}$ axis pointing out the nose of the airframe (Figure 1.3a).

A new rotation about $\mathbf{j}^{v1}$ by the pitch angle θ is then conducted. Define therefore the new intermediate coordinate system $\mathcal{F}^{v2}$ where $\mathbf{p}^{v1}$ can be expressed relatively as:

$$\mathbf{p}^{v2} = \mathcal{R}_{v1}^{v2}(\theta)\mathbf{p}^{v1}. \tag{1.6}$$

As before, $\mathbf{i}^{v2}$ points out the nose of the UAV, but $\mathbf{k}^{v2}$ now points out the belly (Figure 1.3b).

Finally, the body frame $\mathcal{F}^b$ is obtained with the last rotation about the $\mathbf{i}^{v2}$ axis by the roll angle ϕ. In this situation, $\mathbf{i}^b$ points out the nose of the airframe, $\mathbf{j}^b$ points out the right wing and $\mathbf{k}^b$ points out the belly (Figure 1.3c). The transformation from $\mathcal{F}^{v2}$ to $\mathcal{F}^b$ is:

$$\mathbf{p}^b = \mathcal{R}_{v2}^b(\phi)\mathbf{p}^{v2} \tag{1.7}$$

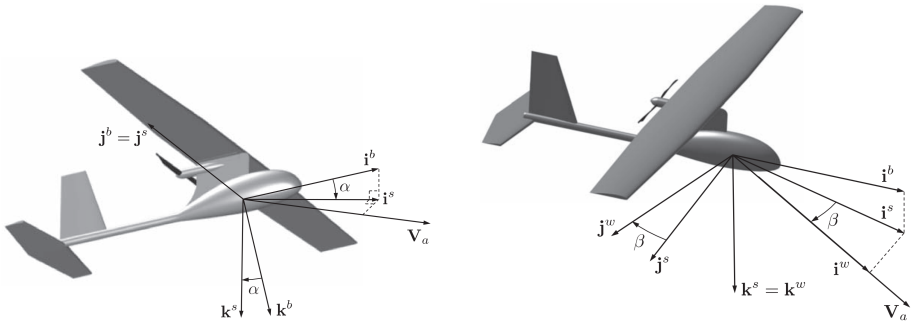
The final rotation matrix from the vehicle frame to the body frame is therefore given by:

$$\mathcal{R}_v^b(\phi,\theta,\psi) = \mathcal{R}_{v2}^b(\phi)\mathcal{R}_{v1}^{v2}(\theta)\mathcal{R}_v^{v1}(\psi)$$
$$= \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\psi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\psi s_\theta s_\psi - s_\psi c_\psi & c_\phi c_\theta \end{bmatrix} \tag{1.8}$$

where $c_\phi = \cos\phi$ and $s_\phi = \sin\phi$.

As said, the ϕ, θ and ψ angle are commonly referred as Euler angles, which provide an intuitive means for representing the orientation of the body with respect to the inertial frame.

1.2.4 The stability and wind frames $\mathcal{F}^s$, $\mathcal{F}^w$



(a) From $\mathcal{F}^b$ to $\mathcal{F}^s$ (b) From $\mathcal{F}^s$ to $\mathcal{F}^w$

Figure 1.4: Representation of stability and wind frames

In order to complete the frame definitions and build up a suitable background when treating of aerodynamic forces, the stability and wind coordinate systems need to be introduced.

The velocity of the aircraft with respect to the surrounding air is called *airspeed* and denoted as $\mathbf{V}_a$. To generate the lift necessary to flight, the wings must fly at a positive angle with respect to the airspeed vector. This angle is called angle of attack and denoted with α. For this reason, the stability frame must be introduced as the left-handed rotation about $\mathbf{j}^b$ by the angle α from $\mathcal{F}^b$. The need for a left-handed rotation comes from the definition of the angle of attack, which must be positive when the right-handed rotation from the stability frame to the body frame occurs. The situation is represented in Figure 1.4a. The transformation from $\mathcal{F}^b$ to $\mathcal{F}^s$ is:

$$\mathbf{p}^s = \mathcal{R}_b^s(\alpha)\mathbf{p}^b \tag{1.9}$$

where the left-handed rotation matrix $\mathcal{R}_b^s(\alpha)$ is

$$\mathcal{R}_b^s(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}. \tag{1.10}$$

Whenever the airspeed vector does not lie on the $(\mathbf{i}^b, \mathbf{k}^b)$ plane, it is necessary to define another angle, namely the side-slip angle β. By rotating the stability frame by a right-handed rotation of β about $\mathbf{k}^s$, a new coordinate system is defined and called wind frame $\mathcal{F}^w$ (Figure 1.4b). Here, the unit vector $\mathbf{i}^w$ is aligned with the airspeed direction V_a.

The transformation is:

$$\mathbf{p}^w = \mathcal{R}_s^w(\beta)\mathbf{p}^s \tag{1.11}$$

where

$$\mathcal{R}_s^w(\beta) = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.12}$$

In conclusion, the total transformation from the body frame to the wind frame is defined as:

$$\begin{aligned} \mathcal{R}_b^w(\alpha,\beta) &= \mathcal{R}_s^w(\beta)\mathcal{R}_b^s(\alpha) \\ &= \begin{bmatrix} \cos\beta\cos\alpha & \sin\beta & \cos\beta\sin\alpha \\ -\sin\beta\cos\alpha & \cos\beta & -\sin\beta\sin\alpha \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}. \end{aligned}$$

1.3 The wind triangle

When dealing with UAVs, wind has a strong impact on the flight mechanics since it may even represent 20%-50% of the airframe airspeed; moreover, the aerodynamic forces depends on the relative speed with respect to surrounding air. Therefore wind must be properly taken into account during modelling. What follows is aimed to derive the essential expressions in formulating the equations of motion for an UAV.

The wind velocity relative to the inertial frame is hereafter named $\mathbf{V}_w$. In the same way, the airspeed with respect to the same frame is denoted with $\mathbf{V}_a$. To figure out how wind comes into play, one defines the ground speed as $\mathbf{V}_g$, hence the UAV velocity relative to the inertial frame. Airspeed, ground speed and wind speed vectors are related by equation (1.13):

$$\mathbf{V}_a = \mathbf{V}_g - \mathbf{V}_w. \tag{1.13}$$

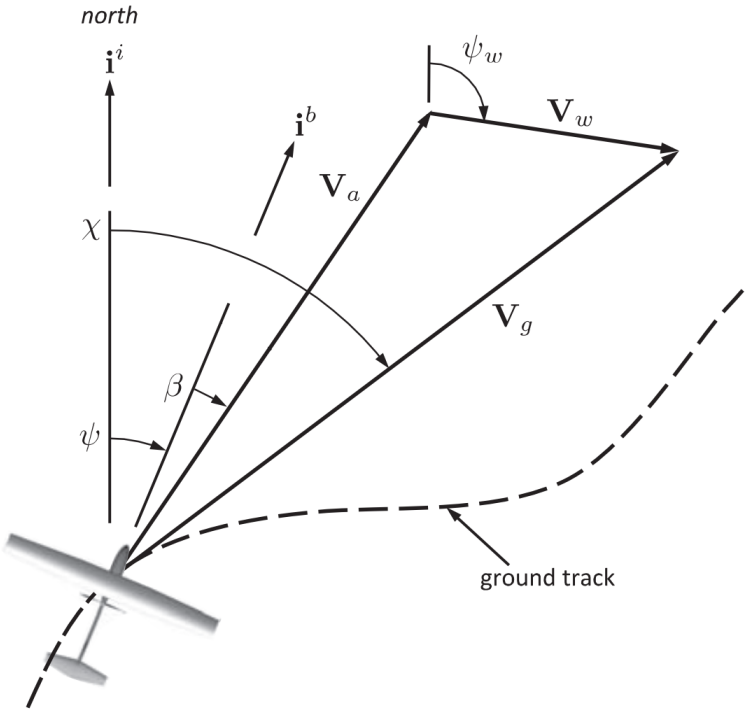


Figure 1.5: The wind triangle

The ground speed can be expressed in the body frame in terms of components along $(\mathbf{i}^b, \mathbf{j}^b, \mathbf{k}^b)$ axes as:

$$\mathbf{V}_g^b = \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{1.14}$$

Similarly, the wind components as expressed in the body frame are

$$\mathbf{V}_w^b = \begin{bmatrix} u_w \\ v_w \\ w_w \end{bmatrix} = \mathcal{R}_v^b(\phi, \theta, \psi) \begin{bmatrix} w_n \\ w_e \\ w_d \end{bmatrix}, \tag{1.15}$$

where the last column vector is $\mathbf{V}_w$.

Recalling that the airspeed $\mathbf{V}_a$ is the velocity of the UAV in the wind frame, it holds true that:

$$\mathbf{V}_a^w = \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix}. \tag{1.16}$$

To obtain the airspeed in the body components, applying equation (1.13), a rotation from the wind to the body frame $\mathcal{R}_w^b$ is performed:

$$\mathbf{V}_a^b = \begin{bmatrix} u - u_w \\ v - v_w \\ w - w_w \end{bmatrix} = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix} = \mathcal{R}_w^b \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix} = V_a \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}. \tag{1.17}$$

$$\tag{1.18}$$

Solving (1.17) for V_a, α, and β:

$$V_a = \sqrt{u_r^2 + v_r^2 + w_r^2} \tag{1.19}$$

$$\alpha = \tan^{-1}\left(\frac{w_r}{u_r}\right) \tag{1.20}$$

$$\beta = \sin^{-1}\left(\frac{v_r}{\sqrt{u_r^2 + v_r^2 + w_r^2}}\right). \tag{1.21}$$

Equations (1.19), (1.20) and (1.21) will be essential when formulating the equations of motion for the UAV.

To complete the picture, the wind triangle is presented in Figure 1.5. The angle between the wind vector and $\mathbf{i}^i$ is denoted with ψ_w. A new angle χ is introduced, which represents the angle between the true North and the projection of the $\mathbf{V}_g$ on the horizontal plane $(\mathbf{i}^b, \mathbf{j}^b)$. This absolute quantity will be recalled in Chapter 4, since it constitutes the control variable for the the guidance logic.

1.4 Kinematics and flight dynamics

In developing the equations of motion for the UAV, twelve state variables will be introduced. They are summarized in Table 1.1, while in Figure 1.6 the axes of motion are depicted.

1.4.1 Kinematics

Given the states definition given in Table 1.1, u, v and w are the inertial velocity components projected onto the body frame. Therefore the relationship between translational velocity and position requires a differentiation and rotation:

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \mathcal{R}_b^v(\phi, \theta, \psi) \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{1.22}$$

As for the angular quantities, the derivation is complicated by the fact that roll, pitch, and yaw angles are defined with respect to their intermediate frames respectively. The body-frame angular rates can be expressed in terms of the

Name	Description
p_n	Inertial north position along $\mathbf{i}^i$ in $\mathcal{F}^i$
p_e	Inertial east position along $\mathbf{j}^i$ in $\mathcal{F}^i$
p_d	Inertial down position along $\mathbf{k}^i$ in $\mathcal{F}^i$
u	Body frame velocity along $\mathbf{i}^b$ in $\mathcal{F}^b$
v	Body frame velocity along $\mathbf{j}^b$ in $\mathcal{F}^b$
w	Body frame velocity along $\mathbf{k}^b$ in $\mathcal{F}^b$
ϕ	Roll angle defined with respect to $\mathcal{F}^{v2}$
θ	Pitch angle defined with respect to $\mathcal{F}^{v1}$
ψ	Yaw angle defined with respect to $\mathcal{F}^v$
p	Roll rate measured along $\mathbf{i}^b$ in $\mathcal{F}^b$
q	Pitch rate measured along $\mathbf{j}^b$ in $\mathcal{F}^b$
r	Yaw rate measured along $\mathbf{k}^b$ in $\mathcal{F}^b$

Table 1.1: State variables of the equations of motion

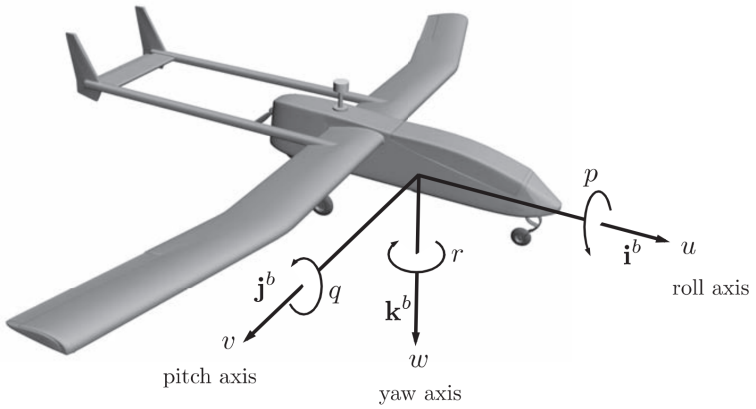


Figure 1.6: Definition of axes of motion

derivatives of the Euler angles, provided that the proper angular rotations are carried out as:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \mathcal{R}^b_{v2}(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \mathcal{R}^b_{v2}(\phi)\mathcal{R}^{v2}_{v1}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{1.23}$$

which gives:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. \tag{1.24}$$

Inverting this expression yields

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\psi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{1.25}$$

Equation (1.25) shows that Euler angles representation of the attitude has a mathematical singularity when $\theta = \pm 90\,\text{deg}$, in which case the yaw angle ψ is not defined. This situation is commonly known as gimbal lock. Fortunately, for fixed-wing UAVs, this is unlikely to happen since in normal flight conditions the pitch angle is usually small.

1.4.2 Rigid-body dynamics

To derive the dynamic equations of motion, the second Newton's law is applied. The forces and moments have to be expressed in the inertial frame; however, this quantities may be also expressed using components associated with other frames, such as the body frame. The flat earth model is used, which is appropriate for small UAVs.

Translational motion

Regarding translational motion, it holds that:

$$m\left(\frac{d\mathbf{V}_g}{dt}\right)^i = \sum_k \mathbf{F}_k \tag{1.26}$$

where m is the time-invariant mass of the UAV and $\mathbf{F}_k$ is the k-th external force acting on the airframe. External forces are represented by gravity, and aerodynamic and propulsive forces. The superscript i indicates that the time derivative is operated in the inertial frame.

From Appendix A, it is known that the derivative $\left(\frac{d\mathbf{V}_g}{dt}\right)^i$ can be written in terms of the derivative in the body frame. Equation (1.26) then becomes:

$$m\left[\left(\frac{d\mathbf{V}_g}{dt}\right)^b + \boldsymbol{\omega}_{b/i} \times \mathbf{V}_g\right] = \sum_k \mathbf{F}_k \tag{1.27}$$

where $\boldsymbol{\omega}_{b/i}$ is the angular velocity of frame $\mathcal{F}^b$ in the frame $\mathcal{F}^i$. Expressing (1.27) in terms of body frame components one get:

$$m\left[\left(\frac{d\mathbf{V}_g^b}{dt}\right)^b + \boldsymbol{\omega}_{b/i}^b \times \mathbf{V}_g^b\right] = \sum_k \mathbf{F}_k. \tag{1.28}$$

Equation (1.28) simplifies the application of Newton's law since the forces are *expressed* in the body frame. Recalling that $\mathbf{V}_g^b = [u \quad v \quad w]^T$ and $\boldsymbol{\omega}_{b/i}^b = [p \quad q \quad r]^T$, reorganising the terms of (1.28), and letting $\sum_k \mathbf{F}_k = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$, one finally has:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}. \tag{1.29}$$

Rotational motion

Newton's law for rotational motion states that

$$\left(\frac{d\mathbf{h}}{dt}\right)^i = \sum_k \mathbf{M}_k \tag{1.30}$$

where the left-hand side term is the time derivative of the body angular momentum vector with respect to the inertial frame, and $\mathbf{M}_k$ is the k-th externally applied moment. Following the same procedure of the translational dynamic equations, equation (1.30) becomes:

$$\begin{aligned} \left(\frac{d\mathbf{h}}{dt}\right)^i &= \left(\frac{d\mathbf{h}}{dt}\right)^b + \boldsymbol{\omega}_{b/i} \times \mathbf{h} \\ &= \left(\frac{d\mathbf{h}^b}{dt}\right)^b + \boldsymbol{\omega}_{b/i}^b \times \mathbf{h}^b = \sum_k \mathbf{M}_k. \end{aligned} \tag{1.31}$$

For a rigid body, the angular momentum is defined as $\mathbf{h}^b = \mathbf{J}\boldsymbol{\omega}_{b/i}^b$, where $\mathbf{J}$ is the inertia tensor, defined as:

$$\mathbf{J} \triangleq \begin{bmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{bmatrix} \tag{1.32}$$

where

$$\begin{array}{llllll} J_x &= \int (y^2 + z^2) dm & J_y &= \int (x^2 + z^2) dm & J_z &= \int (x^2 + z^2) dm \\ J_{xy} &= \int (xy) dm & J_{xz} &= \int (xz) dm & J_{yz} &= \int (yz) dm. \end{array}$$

Aircraft are commonly symmetric about the plane spanned by $\mathbf{i}^b$ and $\mathbf{k}^b$, and for this reason we let $J_{xy} = J_{yz} = 0$. Since $\mathbf{J}$ is defined in the body frame, its derivative in the same frame is obviously $\left(\frac{d\mathbf{J}}{dt}\right)^b = 0$.

Equation (1.31) then becomes:

$$\mathbf{J}\left(\frac{\boldsymbol{\omega}_{b/i}^{b}}{dt}\right)^{b} + \boldsymbol{\omega}_{b/i}^{b} \times \left(\mathbf{J}\boldsymbol{\omega}_{b/i}^{b}\right) = \sum_{k} \mathbf{M}_{k} \tag{1.33}$$

and left-multiplying by $\mathbf{J}$ we get

$$\left(\frac{\boldsymbol{\omega}_{b/i}^{b}}{dt}\right)^{b} = \dot{\boldsymbol{\omega}}_{b/i}^{b} = \mathbf{J}^{-1}\left[-\boldsymbol{\omega}_{b/i}^{b} \times \left(\mathbf{J}\boldsymbol{\omega}_{b/i}^{b}\right) + \sum_{k} \mathbf{M}_{k}\right]. \tag{1.34}$$

Recalling that p, q, and r are the angular rates in the body frame,

$$\boldsymbol{\omega}_{b/i}^{b} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \Rightarrow \dot{\boldsymbol{\omega}}_{b/i}^{b} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}. \tag{1.35}$$

Letting $\sum_k \mathbf{M}_k = \begin{bmatrix} \mathcal{L} & \mathcal{M} & \mathcal{N} \end{bmatrix}^T$, the rotational dynamics equations are finally derived:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6 (p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{bmatrix} + \begin{bmatrix} \Gamma_3 \mathcal{L} + \Gamma_4 \mathcal{N} \\ \frac{1}{J_y}\mathcal{M} \\ \Gamma_4 \mathcal{L} + \Gamma_8 \mathcal{N} \end{bmatrix} \tag{1.36}$$

where

$$\Gamma_1 = \frac{J_{xz}(J_x - J_y + J_z)}{\Gamma} \qquad \Gamma_2 = \frac{J_z(J_z - J_y) + J_{xz}^2}{\Gamma}$$
$$\Gamma_3 = \frac{J_z}{\Gamma} \qquad \Gamma_4 = \frac{J_{xz}}{\Gamma}$$
$$\Gamma_5 = \frac{J_z - J_x}{J_y} \qquad \Gamma_6 = \frac{J_{xy}}{J_y}$$
$$\Gamma_7 = \frac{(J_x - J_y)J_x + J_{xy}^2}{\Gamma} \qquad \Gamma_8 = \frac{J_x}{\Gamma}$$

and $\Gamma = J_x J_z - J_{xz}^2$. Concluding, the 6 degrees-of-freedom, 12-state non-linear model for the UAV kinematics and dynamics is summarized by equations (1.22),

(1.29), (1.25), (1.36), and reported below:

$$\begin{cases} \begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{bmatrix} = \begin{bmatrix} c_\theta c_\psi & s_\psi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s\psi + c_\phi c_\psi & c_\psi s_\theta s_\psi - s_\psi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \\ \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & \sin\phi \\ 0 & \sin\phi\sec\theta & \cos\psi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \\ \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \Gamma_1 pq - \Gamma_2 qr \\ \Gamma_5 pr - \Gamma_6(p^2 - r^2) \\ \Gamma_7 pq - \Gamma_1 qr \end{bmatrix} + \begin{bmatrix} \Gamma_3\mathcal{L} + \Gamma_4\mathcal{N} \\ \frac{1}{J_y}\mathcal{M} \\ \Gamma_4\mathcal{L} + \Gamma_8\mathcal{N} \end{bmatrix} \end{cases} . \tag{1.37}$$

1.5 External forces and moments

In order to complete the aircraft modelling, the external forces and moments are described. The sum of all external forces and moments is:

$$\begin{aligned} \mathbf{F} &= \mathbf{F}_g + \mathbf{F}_a + \mathbf{F}_p \\ \mathbf{M} &= \mathbf{M}_a + \mathbf{M}_p, \end{aligned}$$

where subscript g stay for the gravity contribution, a for the aerodynamic, and p for the propulsion.

1.5.1 Gravitational force

The effect of the gravitational field is described as a force acting in the $\mathbf{k}^v$ direction at the aircraft center of mass. Therefore it does not produce any moments.

$$\mathbf{F}_g^v = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \Rightarrow \mathbf{F}_g^b = \mathcal{R}_v^b(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{1.38}$$

1.5.2 Aerodynamic forces and moments

Before giving the expressions of the forces and moments acting onto the body, the control surfaces must be introduced first. Figure 1.7 shows the standard configuration, where the aileron deflection is denoted with δ_a, the elevator deflection is

denoted with δ_e, the rudder deflection with δ_r, and the throttle with δ_t. They are all expressed in radians, except the throttle that is in [%]. The positive deflection of the surfaces is determined by applying the right-hand rule to the hinge axis of the control surface. For the ailerons, the deflection can be expressed as:

$$\delta_a = \frac{1}{2}(\delta_{a_{\text{left}}} - \delta_{a_{\text{right}}}) \tag{1.39}$$

where $\delta_{a_{\text{left}}}$ and $\delta_{a_{\text{right}}}$ are the left and right ailerons deflection respectively. In this way, a positive δ_a will produce, as will be seen, a positive rolling moment about $\mathbf{i}^b$, thus respecting the convention.

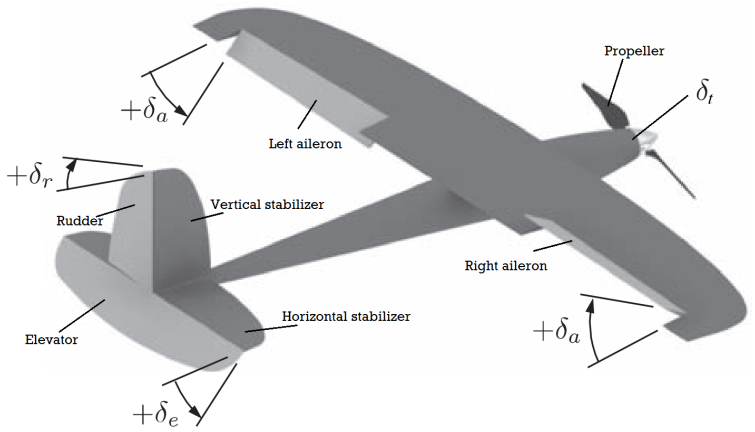


Figure 1.7: Aircraft control surfaces representation, plus the propeller

Despite the fact that there are three forces and three moments to define, they will be grouped in order to analyse the longitudinal aerodynamics first, and then the lateral aerodynamics.

Longitudinal aerodynamics

The forces and the moment that cause a body motion in the $(\mathbf{i}^b, \mathbf{k}^b)$ plane are the lift force, the drag force and the pitch moment. Their definitions are respectively:

$$F_{\text{lift}} = \frac{1}{2}\rho V_a^2 S C_L(\alpha, q, \delta_e) \tag{1.40}$$

$$F_{\text{drag}} = \frac{1}{2}\rho V_a^2 S C_D(\alpha, q, \delta_e) \tag{1.41}$$

$$\mathcal{M} = \frac{1}{2}\rho V_a^2 S c C_m(\alpha, q, \delta_e) \tag{1.42}$$

where ρ is the air density, S the planform area of the single wing, c is the main chord of the wing. V_a is the airspeed as already described in (1.13). C_L, C_D, and

C_m are non-dimensional coefficients which depend on the angle of attack α, the pitch rate q, and the elevator deflection δ_e. This relationship can be approximated with a first order Taylor series expansion:

$$C_L(\alpha, q, \delta_e) = \left[C_{L_0} + \frac{\partial C_L}{\partial \alpha}\alpha + \frac{\partial C_L}{\partial q}q + \frac{\partial C_L}{\partial \delta_e}\delta_e \right] \tag{1.43}$$

which can be rewritten more compactly and normalizing the third term (which was expressed in [rad/s] because of q) as:

$$C_L(\alpha, q, \delta_e) = \left[C_{L_0} + C_{L_\alpha}\alpha + C_{L_q}\frac{c}{2V_a}q + C_{L_{\delta_e}}\delta_e \right]. \tag{1.44}$$

In the previous expression, $C_{L_\alpha} \triangleq \frac{\partial C_L}{\partial \alpha}$, $C_{L_q} \triangleq \frac{\partial C_L}{\partial \frac{qc}{2V_a}}$, and $C_{L_{\delta_e}} \triangleq \frac{\partial C_L}{\partial \delta_e}$.

Proceeding similarly for the drag and the pitching moment, one has:

$$C_D(\alpha, q, \delta_e) = \left[C_{D_0} + C_{D_\alpha}\alpha + C_{D_q}\frac{c}{2V_a}q + C_{D_{\delta_e}}\delta_e \right] \tag{1.45}$$

$$C_m(\alpha, q, \delta_e) = \left[C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_a}q + C_{m_{\delta_e}}\delta_e \right]. \tag{1.46}$$

F_{lift} and F_{drag} are expressed in the stability frame (Figure 1.8). To bring them onto the body frame, the following expression holds:

$$\begin{bmatrix} f_x \\ 0 \\ f_z \end{bmatrix} = \mathcal{R}_w^b(\alpha, \beta) \begin{bmatrix} -F_{\text{drag}} \\ 0 \\ -F_{\text{lift}} \end{bmatrix} \tag{1.47}$$

The negative sign came to respect the NED convention of the body frame.

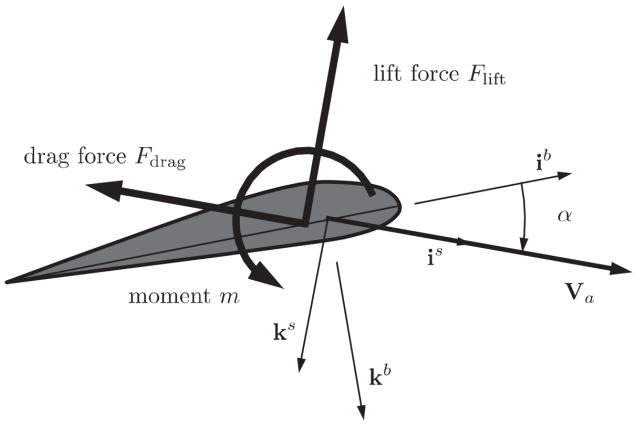


Figure 1.8: Lift and drag forces directions for a positive angle of attack

Lateral aerodynamics

The lateral aerodynamics directly influences the lateral direction along $\mathbf{j}^b$, as well as the rolling and yawing moments. They are heavily influenced by β as depicted in the following equations:

$$f_y = \frac{1}{2}\rho V_a^2 S C_Y(\beta, p, r, \delta_a, \delta_r) \tag{1.48}$$

$$\mathcal{L} = \frac{1}{2}\rho V_a^2 S b C_l(\beta, p, r, \delta_a, \delta_r) \tag{1.49}$$

$$\mathcal{N} = \frac{1}{2}\rho V_a^2 S b C_n(\beta, p, r, \delta_a, \delta_r). \tag{1.50}$$

The new coefficient b represents the wingspan. The first-order Taylor series of the coefficients C_Y, C_l, and C_n is again:

$$C_Y(\beta, p, r, \delta_a, \delta_r) = \left[C_{Y_0} + C_{Y_\beta}\beta + C_{Y_p}\frac{b}{2V_a}p + C_{Y_r}\frac{b}{2V_a}r + C_{Y_{\delta_a}}\delta_a + C_{Y_{\delta_r}}\delta_r\right] \tag{1.51}$$

$$C_l(\beta, p, r, \delta_a, \delta_r) = \left[C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r\right] \tag{1.52}$$

$$C_n(\beta, p, r, \delta_a, \delta_r) = \left[C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r\right]. \tag{1.53}$$

This section is concluded with two remarks: firstly C_{Y_0}, C_{l_0}, and C_{n_0} are zero for aircraft symmetric about $(\mathbf{i}^b, \mathbf{k}^b)$. Secondly, the coefficients related with α, β, p, q, and r are commonly known as *stability derivatives*, while those related to δ_a, δ_e, and δ_r as *control derivatives*.

1.5.3 Propulsion force

In literature there are many models of propellers. A simple model, which is suitable for UAV modelling, can be developed applying the Bernoulli principle to compute the thrust of the propeller. It is known that:

$$F_p = S_p \Delta P \tag{1.54}$$

where S_p is the area swept out by the propeller and $\Delta P = P_{\text{out}} - P_{\text{in}}$, being P_{in} and P_{out} the pressures before and after the propeller respectively.

Using Bernoulli's principle, the inward pressure and outward pressure are expressed by:

$$P_{\text{in}} = P_0 + \frac{1}{2}\rho V_a^2 \tag{1.55}$$

$$P_{\text{out}} = P_0 + \frac{1}{2}\rho V_{\text{out}}^2 \tag{1.56}$$

where V_{out} is the air velocity at the exit of the propeller.

References [2], [3] shows that V_{out} can be approximated as $R\Omega$, where R is the radius of the propeller. Giurato thesis [2] points out that the relationship between Ω and the commanded throttle δ_t exists and is linear:

$$\Omega = k_{\text{motor}}\delta_t + q_{\text{motor}}. \tag{1.57}$$

Finally the following equation holds:

$$F_p = S_p C_{\text{prop}}(P_{\text{out}} - P_{\text{in}}) \tag{1.58}$$

$$= \frac{1}{2}\rho S_p C_{\text{prop}}[(R\Omega)^2 - V_a^2] \tag{1.59}$$

where C_{prop} is a non-dimensional coefficient, called rotor thrust coefficient.

It is obvious that the propeller is placed in order to not generate any lateral force. This means that the thrust generated can be decomposed in one vertical and one horizontal component to get the final expression of $\mathbf{F}_p$:

$$\mathbf{F}_p = \begin{bmatrix} F_{p,x} \\ 0 \\ F_{p,z} \end{bmatrix}. \tag{1.60}$$

$F_{p,x}$ and $F_{p,z}$ are the projected components of F_p onto $\mathbf{i}^b$ and $\mathbf{k}^b$ respectively. The motor rotation always causes a slow rolling motion of the aircraft which is treated as a disturbance and counteracted with a small occasional deflection of the ailerons.

1.6 Atmospheric disturbances

It has been seen that wind plays a significant role in defining the aircraft dynamics. Wind is modelled as the composition of a constant part, the steady-state ambient wind $\mathbf{V}_{w,s}$, and a dynamic part $\mathbf{V}_{w,d}$.

$$\mathbf{V}_w = \mathbf{V}_{w,s} + \mathbf{V}_{w,d}. \tag{1.61}$$

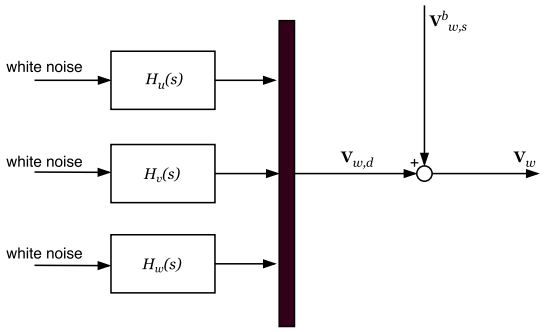


Figure 1.9: Block representation of the wind velocity vector

The constant vector $\mathbf{V}_{w,s}$ is better described in the inertial frame, however, remembering equation (1.17), it has to be expressed onto the body frame:

$$\mathbf{V}_{w,s} = \mathcal{R}^b_i(\phi, \theta, \psi)\mathbf{V}^i_{w,s}. \tag{1.62}$$

The dynamic part $\mathbf{V}_{w,d}$ represents wind gusts and other atmospheric disturbances. Experimental results indicate that a good model for the non-steady gust portion of the wind model is obtained by passing white noise through appropriate forming filters [4]. One of the most commonly used models of continuous gusts is the Dryden wind model [5].

The Dryden wind turbulence model uses the Dryden spectral representation to add turbulence to the aerospace model by passing band-limited white noise through appropriate forming filters that can be written in a simplified form as:

$$H_u(s) = \sigma_u \sqrt{\frac{2V_a}{L_u}} \frac{1}{s + \frac{V_a}{L_u}} \tag{1.63}$$

$$H_v(s) = \sigma_v \sqrt{\frac{3V_a}{L_v}} \frac{s + \frac{V_a}{\sqrt{3}L_v}}{\left(s + \frac{V_a}{L_v}\right)^2} \tag{1.64}$$

$$H_w(s) = \sigma_w \sqrt{\frac{3V_a}{L_w}} \frac{s + \frac{V_a}{\sqrt{3}L_w}}{\left(s + \frac{V_a}{L_w}\right)^2} \tag{1.65}$$

where σ_u, σ_v, and σ_u are the intensities of the turbulence along the vehicle frame axes; L_u, L_v, and L_w are the spacial wavelengths, and V_a is the airspeed of the vehicle.

In Figure 1.9, the overall wind model is presented. It is through $[u_w, v_w, w_w]^T$ that the airspeed vector in body frame components can be computed and through these variables the atmospheric effects enter in the computation of the angle of attack and side-slip angle, thus influencing the motion of the aircraft.

1.7 Conclusions

The equations of motion in (1.37) are implemented in Simulink. The Aerospace Blockset is a Simulink plug-in that extends Simulink capabilities with blocks for modelling and simulating aircraft, spacecraft, rockets, and propulsion systems, as well as unmanned airborne vehicles. It also includes blocks that implement mathematical representations from aerospace standards, common references, and environmental models. These auxiliary blocks are used throughout the simulator when they enhance its overall clarity and functionality.

In Figure 1.10 the forces and moments contributions are shown on the left. At the bottom, the block 'Derived conditions' contains the implementation of the wind triangle, hence it computes the airspeed, angle of attack, side-slip angle, course angle, and other useful quantities taking as input the wind vector (*EnvData*) and the vector $\mathbf{V}_g$ from *PlantData*. The simulator uses many variables scattered in all its layers. As noticeable, there is an extensive use of Bus objects. This helps to maintain the model clear, bringing all the signals across the Simulink model.

To have an immediate comprehension of the UAV behaviour, a specific block was built as a visual interface. In there, flight instruments can help to analyse the flight and reveal potential errors (Figure 1.11). Moreover, a 3D interface helps analysing the angular rates progression, a real-time parser slows down the simulation when needed, and a box shows the model initial conditions each time the simulation is run. There is also the possibility to show the running simulation on FlightGear (see Reference [6], Figure 1.12).

Lastly, in Figure 1.13 the environment modelling is shown. The Aerospace Blockset already gives an implementation of the Dryden turbulence model.

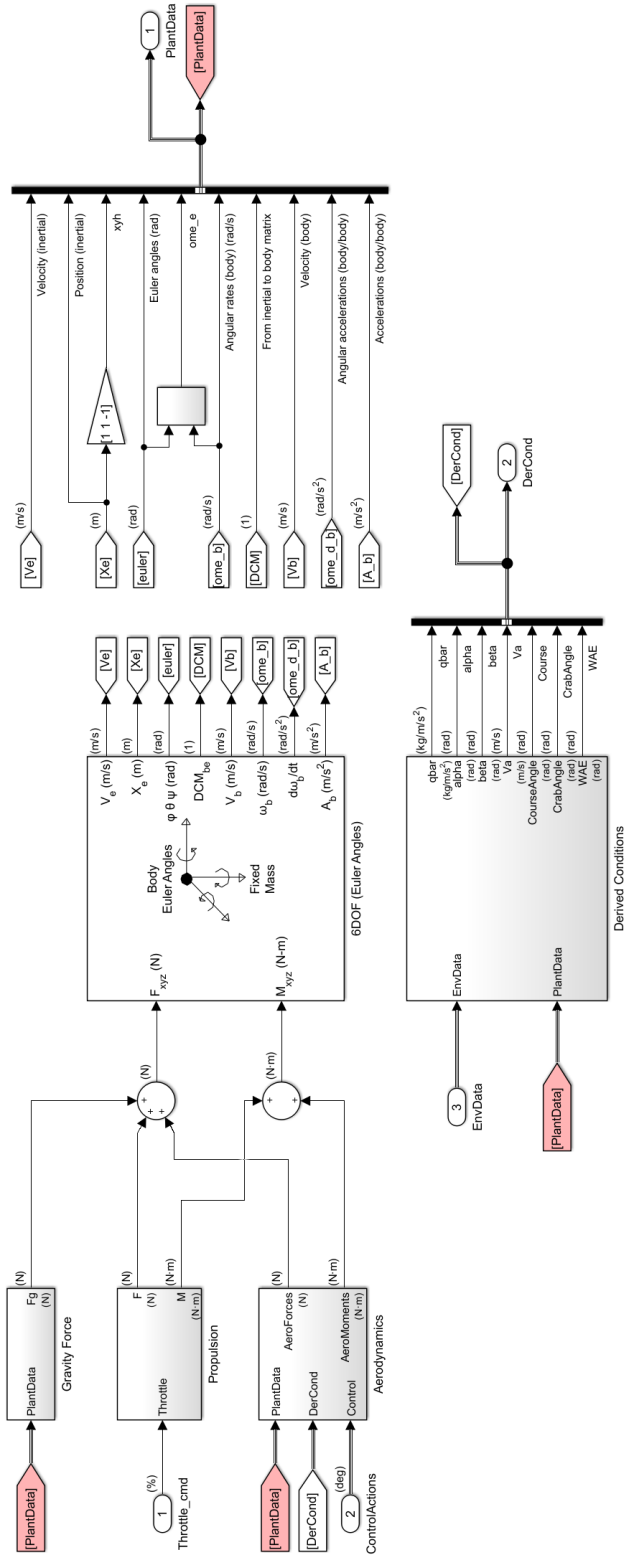


Figure 1.10: Simulink block diagram of the UAV dynamics

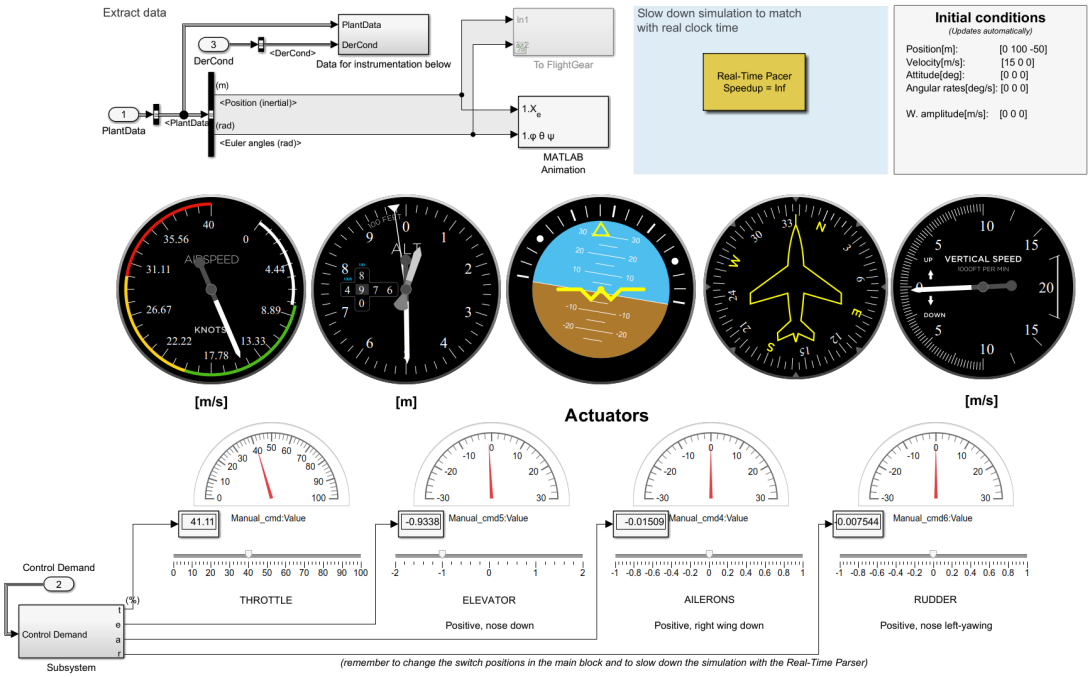


Figure 1.11: Simulink visual interface sub-block

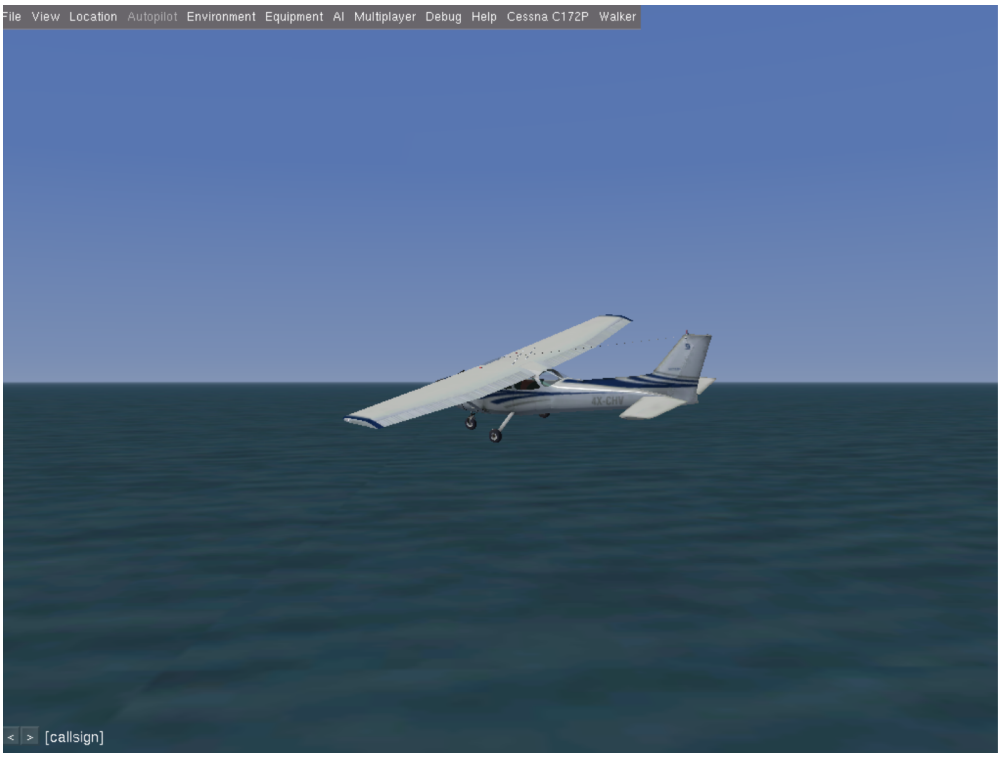


Figure 1.12: A Simulink simulation running with the FlightGear interface

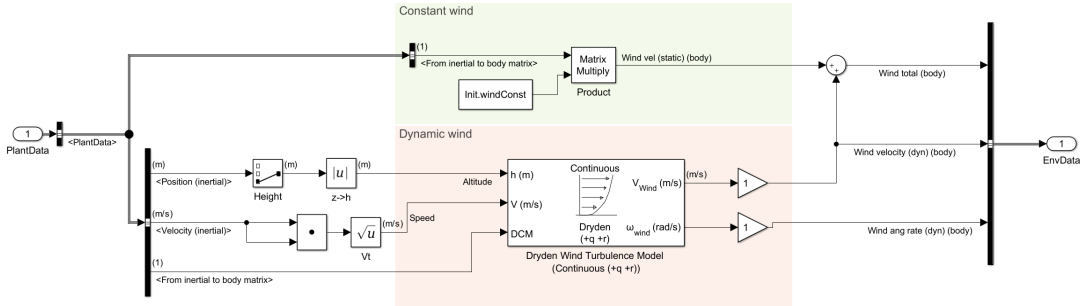


Figure 1.13: The simulator environment sub-block

Chapter 2

Hardware and software integration

This chapter presents the integration of the actual fixed-wing UAV which is studied in this thesis. In Section 2.1 the principal considerations behind the components selection are discussed and the requirements then illustrated. Successively the Flight Control Unit software and hardware are described pointing out motivations behind its choice. Successively, the UAV components are analysed one by one in Section 2.3, while the integration is treated in Section 2.4. Afterwards the unknown physical and aerodynamic model parameters will be estimated for the chosen platform in Section 2.5. Finally the Simulink implementation details are shown.

2.1 Preliminary requirements

Fixed-wing aircraft have plenty of possible physical configurations: not only the geometrical properties affect the flight performance, but also some factors like the tail configuration or the motor mount can result decisive when having to perform multiple flight tests. For the thesis purposes, at least at this stage, the focus is not to have an high performance vehicle to obtain impressive results, rather to build up a standard framework where it is possible to apply high-level controllers for the guidance with the maximum flexibility. The hardware project requirements relative to the airframe are focused on lightness (for handling and safety reasons), on the inner longitudinal and lateral stability (given by the geometric design), durability, and, obviously, on its ease of carrying a payload which contains the electronic equipment.

Lightness is mainly given by materials; for small aircraft it is common to use foam airframes. Stability is provided by the body and wings geometry. One important factor to this purpose is the position of the wings. The so-called "high-wing" makes the plane more stable because the plane is "hung" to the wings: its

center of gravity is lower than the bearing point of the lift force, so the aircraft tends to return alone in a stable position; the "low-wing", instead, with the center of gravity placed above the point of application of the lift, makes the plane more unstable but at the same time gives it more manoeuvrability. Some considerations should be also made for the propeller position: in general the propeller can employed in a 'pull' or 'push' configuration. In the first case, it is placed at the nose of the aircraft, while in the second case it is placed behind the center of mass. In general, in UAVs is is favourable to have the push configuration, since the propeller, one of the most fragile component, can be better protected from eventual impacts and therefore last longer.

Above considerations are summarised into the following requirements:

1. Frame configuration: standard airframe with ailerons, elevator, rudder, and propeller, and "high-wing" mount;
2. Frame material: foam;
3. Frame dimension: standard plane dimension for the RC aeromodels category (100÷150cm of wing span and a maximum of 130cm of length);
4. Overall weight, including the payload: up to 2kg;
5. Motor mount: push configuration.

2.2 Flight Control Unit

The Flight Control Unit, or FCU, is the core of each autonomous UAV. Choosing a flight control unit among the existing wide range of options is not trivial. Before choosing the micro-controller hardware, the autopilot software that must be defined.

2.2.1 The software

Recalling that the final goal is to integrate a new guidance logic, it is essential to be able to replicate the low level controllers structure into the Simulink model to provide accurate simulations. For this reason, the first condition is dealing with an open-source firmware. The choice fell on ArduPilot, a professional grade open source unmanned vehicle autopilot software suite, capable of controlling multiple kind of autonomous vehicles, like multirotor drones, fixed-wing and VTOL model aircraft, model helicopters, boats, submarines and ground rovers. ArduPilot could also guarantee the most versatility and support from the point of view of operational documentation and users online feedback.

One of the main feature of ArduPilot is to let the user operate under different flight modes, which are interchangeable using a switch command on his radio controller. The main flight modes are:

Figure 2.1: The HobbyKing HKPilot32 micro-controller

- MANUAL: the radio controller stick commands of δ_a, δ_e, δ_r, and δ_t are replicated to the control actuators as they are. The controller do not play any role, it is the pilot that closes the loop.
- Fly-by-wire A (FBWA): the FCU enables the control of the roll and pitch angles, whose reference is given by the user with the radio controller stick commands.
- Fly-by-wire B (FBWB): similar to FBWA, the FPU controls altitude and airspeed too, taking as inputs the airspeed and rate of climb from the user radio commands.
- AUTOTUNE: it is the same of FBWA mode, but meanwhile uses the aircraft response to tune the pitch and roll controllers and adjusts the control parameters online.
- AUTO: the FPU enables also the guidance logic. The UAV will follow a mission (a set of GPS waypoints and other commands) set by the user.

2.2.2 The hardware

ArduPilot can run on many different micro-controllers and platforms. The HobbyKing HKPilot32 (Figure 2.1) was chosen. It is a Pixhawk clone, an open-hardware flight controller specifically meant for UAV applications. It has two redundant IMUs, which are necessary for the estimation of the plane attitude. An IMU (inertial measurement unit) is a very small electronic device which integrates an accelerometer, a gyroscope, and usually a magnetometer. In HKPilot32,

one IMU contains a 3-axis accelerometer, a 3-axis gyro, and one magnetometer, while the second one only contains a accelerometer and gyro, both different from the first IMU. Any accelerometer can output three acceleration measurements, one per axes, while gyroscopes measures the body angular rates on the three orthogonal axes. There is also a barometer for an indirect altitude measurement.

The micro-controller has plenty of computational capabilities, with a dedicated unit for the floating point computations (FPU). In addition, a real-time operating system runs on it, called NuttX. A Real Time Operating System separates the program functions into self-contained tasks and implements an on-demand scheduling of their execution. The main two benefits are that controller freezing is extremely rare and that the tasks can be executed in parallel. The HKPilot32 has also a redundant power supply configuration: if the main power source fails, the current is drawn from the main output servo-rail (powered separately). Finally, the system can save data logs on an on-board MicroSD card.

2.3 Components description

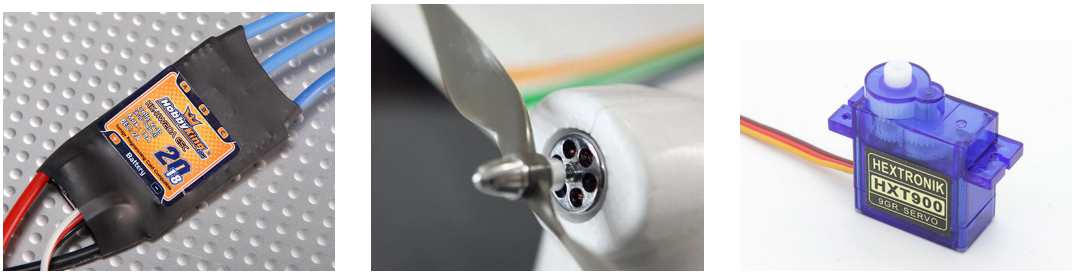
2.3.1 The airframe

Figure 2.2: The HobbyKing Bixler V1.1

The chosen frame is a Bixler v1.1 (HobbyKing). It is made of EPO (Expanded Poly-Olefin) foam, which guarantees the required lightness. With this material, the weight of the empty plane is of 650 g. Foam also provides some elasticity, which is favourable in case of hard landings, and make fractures easier to repair in case of damage. The wings and tail are strengthened with internal plastic rods.

The wingtips have the so-called winglets, which helps in reducing the aircraft's drag by partial recovery of the vortex energy generated by the wings tips, thus enabling a stabler flight. The Bixler comes with integrated servo mechanisms for the control surfaces deflections, and a brushless DC (BLDC) motor. The propeller is placed in the more convenient 'push' configuration.

2.3.2 Motor, ESC, and servos



(a) (b) (c)

Figure 2.3: An example of HobbyKing ESC (left), the motor mount on the Bixler (center) and the HXT900 servo motor (right)

Regarding the motor, the main parameter which describes its behaviour is the Kv value, which stands for the number of revolutions per minute for each volt applied to the motor (with no load connected). As for the Bixler, the out-of-the-box motor has a rating of 1900÷2000 Kv, which for a voltage in the range of 11.1÷12.6V equals a maximum speed of 22200÷25200 rpm. BLDC motors need to be supplied by an inverter with a DC switching electric signal. This device is known in the RC-models world as Electronic Speed Controller, or ESC.

The selected ESC is a 20A ESC (HobbyKing), which means that can it take a DC voltage input from an electric source for a maximum drawn current of 20A, and outputs the required three-phase electric signal to the motor. The power is regulated according to the PWM throttle signal coming from the micro-controller by means of one 3-pins additional cable, called UBEC. This cable is also meant to bring a DC 5V power supply to the servo-rail without an additional battery.

The four servos are light (9 g each), a rise time from 0 to 60 degrees of 0.12 seconds, a maximum torque of 1.6 kg·cm, and commanded through a PWM signal too.

2.3.3 Battery

The battery is a Turnigy nano-tech with a capacity of 2200mAh and 3 Lithium-Polymer cells in series. This gives the battery an operating voltage of 11.1V and an approximated flight time of 15 minutes.

2.3.4 Remote controller and receiver

The remote controller and receiver must be chosen according to the application. The radio set operates on the aircraft using a wireless signal at 2.4GHz; it is capable of driving the main four actuators with the same number of channels. The radio receiver outputs the different channels as PWM signals. In this case, apart from these basic functionalities, radio system has to be versatile and to give the possibility to switch between the different FCU flight modes. The radio chosen is the Turnigy 9X, which also enables using advanced features that facilitate the flight experience.

Figure 2.4: Turnigy 9X radio system

2.3.5 Telemetry module

The Telemetry radio set allows to link 'wirelessly' a flight controller to a USB-equipped device such as a computer, tablet or smartphone supporting that supports a powered USB connection (*i.e.*, the so-called Ground Station). The telemetry radio set not only lets the operator at the Ground Station see live data, such as live GPS position overlaid on a map, but also, for example, system voltage, heading, waypoint navigation, and artificial horizon. It is crucial to know the state of the UAV before, during and after operating a flight, and to have a redundant log file, in case the on-board logging system fails.

The operating frequency is 433Mhz, legally allowed by European rules.

2.3.6 GPS and compass

The GPS and compass are essential for the UAV navigation and course estimation. The GPS is a Ublox NEO M8N, packed together with an extra 3-axis magnetometer. This magnetometer will be used instead of the Pixhawk integrated magnetometer because, being external, it can be placed far from the electronic devices and power cables. In fact, magnetometers are sensors known to be particularly affected by perturbations of the magnetic field in their proximity.

Figure 2.5: GPS receiver and compass module

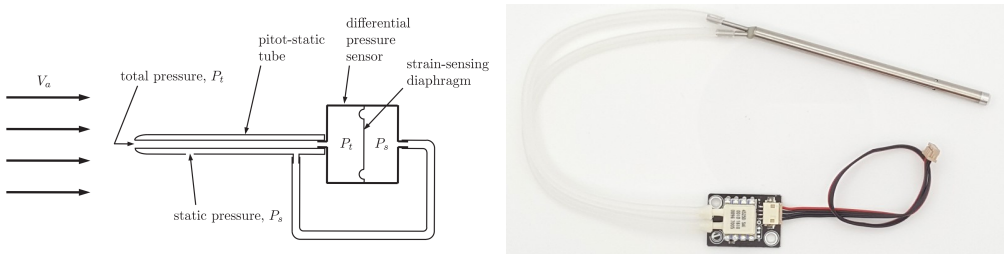
2.3.7 Airspeed sensor

Airspeed can be measured using a pitot probe in conjunction with a differential pressure transducer as depicted schematically in Figure 2.6a. The pitot tube has two ports: one that is exposed to the total pressure and another exposed to the static pressure. The total pressure is the pressure at the tip of the probe opened to the oncoming flow. The static pressure is simply the ambient pressure of the surrounding air. The flow is stagnant or stopped at the tip. As a result, pressure is built up so that the pressure at the tip is higher than that of the surrounding fluid. The difference of pressure is then measured by a digital sensor shown in Figure 2.6b and than converted in airspeed using the Bernoulli principle:

$$P_{\text{total}} - P_{\text{static}} = \frac{\rho V_a^2}{2}. \tag{2.1}$$

2.4 Integration

The integration of all the electronics submodules inside the airframe required multiple iterations up the the final arrangement shown in Figure 2.8.



(a) (b)

Figure 2.6: Description of the airspeed measurement principle (left) and the pitot tube with airspeed sensor (right)

Figure 2.7: HKPilot32 mount inside the Bixler (bottom view)

The micro-controller contains two IMUs, which are necessary for the estimation of the plane attitude. An inertial measurement unit (IMU) is an electronic device that integrates an accelerometer, a gyroscope, and usually a magnetometer too.

For this reason, it had to be placed as close as possible to the center of gravity. In general, aircraft should have the center of mass placed placed at 1/4 of the wing chord starting from the front, as shown in Figure 2.8 with the green dot. Then, the components mass distribution had to be studied accordingly; the empty plane was initially tail-heavier, so the battery, the heavier component, had to be inevitably placed at the front.

Accelerometers are also very sensitive to vibrations. With excessive vibrations, the state estimates can lead to very bad performance in modes that rely on accurate positioning (*e.g.*, Return to Launch failsafe mode, or AUTO flight mode). So four foam dampers were placed between the FPU and the fixing surface, at the corners. These dampers are required to master three effects:

- reduce sensor errors due to mechanical environment solicitations;
- protect sensors as they can be damaged by shocks or vibrations;

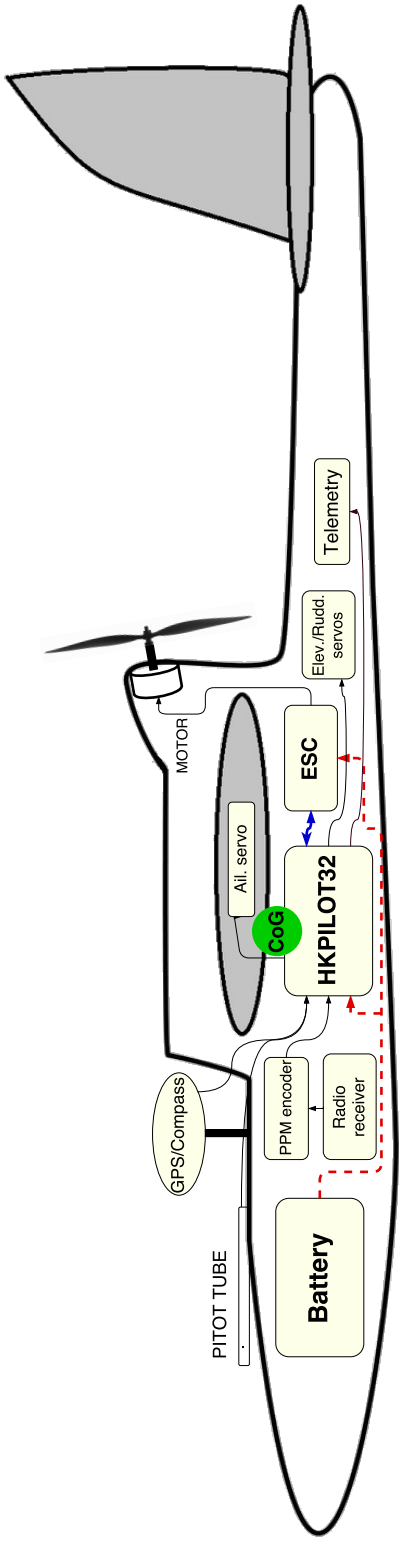


Figure 2.8: Side plane cross-section which shows how the electronics is distributed inside the UAV

Figure 2.9: Side view of the full Bixler after some flights

- contain parasitic IMUs movements within a limited bandwidth, where processing will be able to compensate for them.

The external magnetometer was placed, together with the GPS, 20 cm away from the ESC and 10 cm away from the main power distribution line (depicted as a red dashed line).

The HKPilot32 controller outputs the control commands as PWM signals to the actuators, but requires the inputs from the radio receiver with multiplexed signal on a single line; the PPM encoder module is in charge of multiplexing the PWM signals from the radio receiver to a single PPM signal.

In order to prepare the UAV for flight, the ESC was calibrated to match with the full range of the transmitter throttle stick, as well as the 3-axes accelerometer. The magnetometer was calibrated on the field.

2.5 Parameters estimation

In Chapter 1, it was seen that the model presents several parameters. They have to be estimated and are summarized in Table 2.1.

Physical	Aerodynamic	Other
m	$C_{D_0}, C_{D_\alpha}, C_{D_q}, C_{D_{\delta_e}}$	k_{motor}
$\mathbf{J}$	$C_{Y_\beta}, C_{Y_p}, C_{Y_r}, C_{Y_{\delta_a}}, C_{Y_{\delta_r}}$	q_{motor}
b	$C_{L_0}, C_{L_\alpha}, C_{L_q}, C_{L_{\delta_e}}$	C_{prop}
c	$C_{l_\beta}, C_{l_p}, C_{l_r}, C_{l_{\delta_a}}, C_{l_{\delta_r}}$	
S	$C_{m_0}, C_{m_\alpha}, C_{m_q}, C_{m_{\delta_e}}$	
S_p	$C_{n_\beta}, C_{n_p}, C_{n_r}, C_{n_{\delta_a}}, C_{n_{\delta_r}}$	

Table 2.1: List of the model parameters

The aircraft physical parameters are easily determined with common measurement instruments and resumed in Table C.1. The tensor of inertia **J** is usually determined experimentally using particular equipments, such as the bifilar pendulum, or through a CAD model. However, in [7] of the Bixler it was found an inertia estimate obtained using XFLR5, an analysis tool for airfoils, wings and airplanes.

The most difficult task comes when having to estimate the aerodynamic coefficients. Disregarding the experimental determination in the wind tunnel and given the impossibility to perform repeatable experiments for the identification, the only affordable way is using a predisposed aeronautical software, such as Digital DATCOM.

2.5.1 Digital DATCOM

The United States Air Force Stability and Control Digital DATCOM is a software that implements the methods contained in the USAF Stability and Control DATCOM to calculate the static stability, control and dynamic derivative characteristics of fixed-wing aircraft [8], [9]. Born in 1977, it was made publicly available after some decades.

Digital DATCOM requires an input file containing a geometric description of an aircraft in order to output its corresponding dimensionless coefficients according to the specified flight conditions. Compared to the modern computational fluid dynamics tools, DATCOM provides naturally less accurate results, however, it is shown in [10] that the software still produces comparable results with experimental data and therefore it is chosen for its ease of use.

The software outputs a set of tables containing the aerodynamic parameters for each flight condition, determined by:

1. a set of altitudes (in meters);
2. a set of airspeeds (in meters per second);

as function of

- a set of angles of attack (in degrees);

- a set of elevator and ailerons deflection angles (in degrees).

From DATCOM outputs, one can interpolate this data to obtain the coefficients in each flight condition. Because of the simplified context of application (extremely low altitude and velocities well below the transonic regime), the reference altitude just one value set at 50 m, and the reference airspeed at Mach 0.03 (10 m/s). The angle of attack input is a vector of 20 elements in the interval $[-2, 25]$ deg, while the elevator and ailerons deflections are in the interval $[-20, 20]$ deg and $[-25, 25]$ deg respectively.

One of the geometrical inputs to be defined are the airfoils. An airfoil is the shape of a wing (as seen in cross-section); as the lift force is generated by the difference of pressure between the underlying air pressure and the above pressure, the aerodynamic characteristics of an aerial vehicle depend particularly on this. The airfoils for the wings, the elevator and rudder are determined using an online tool, called NACA calculator [11], which respects the NACA convention. It was possible to determine the best shape which fits with the Bixler wings shapes by inspection. The shape of the NACA airfoils is described using a series of digits following the word "NACA", which uniquely determines the airfoil characteristics, like the camber slope and position. For this case, the profile NACA-1410 (slightly cambered at the front) was chosen for the wing, and NACA-0010 for elevator and rudder (symmetrical 'drop' shape).

The input code can be found in Appendix B. The aircraft geometry input as defined dor DATCOM is shown in Figures 2.10, 2.11 and 2.12. DATCOM cannot reproduce the winglet effects, so the wing is modelled as flat.

The complete set of output parameters is reported in Appendix C. The indicators of a correct geometrical modelling and aircraft stability are the drag, pitching and lifting curves as functions of the angle of attack α. They are shown in Figures 2.14, 2.15 and 2.16. In the lift coefficient plot, the relationship with α is linear [12] for small α values; the stall is taken into account for a very high angle of attack, but after the stall the software gives inaccurate estimates. Regarding the drag curve, it is correctly quadratic for small α. Lastly, the pitch curve has to be almost linearly decreasing for the stability, with a positive value at $\alpha = 0$: this means that for $\alpha = 0$ there is a pitching moment which gives nose up. This causes α to increase and afterwards a negative pitching moment is induced when the curve crosses $C_m = 0$. This intuitively means that, given the appropriate airspeed, the aircraft tries to stabilise the pitching motion.

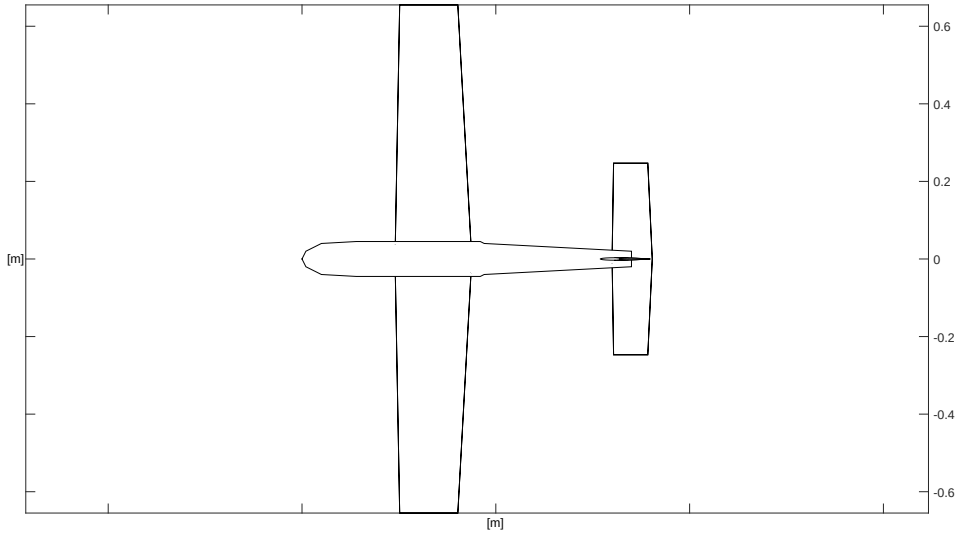


Figure 2.10: Top view of the DATCOM geometry input file

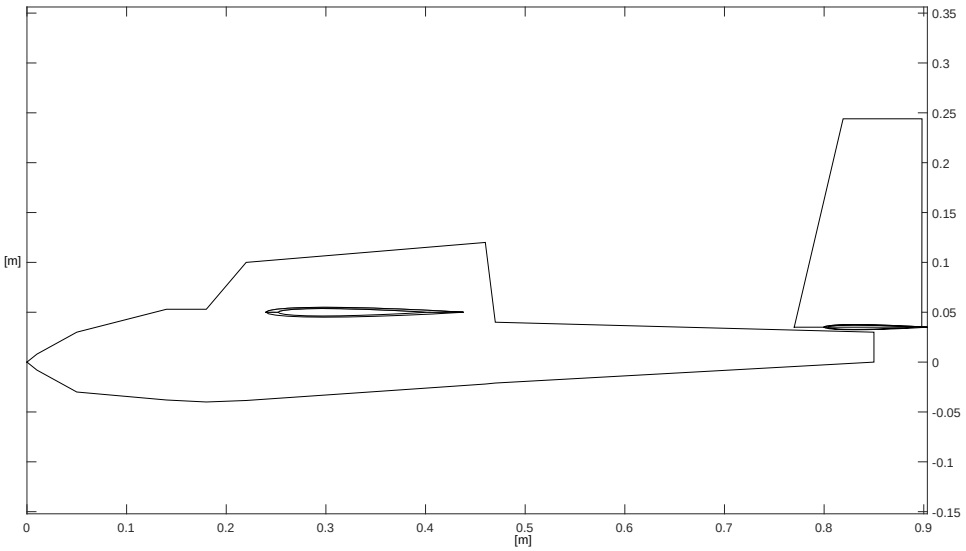


Figure 2.11: Side view of the DATCOM geometry input file

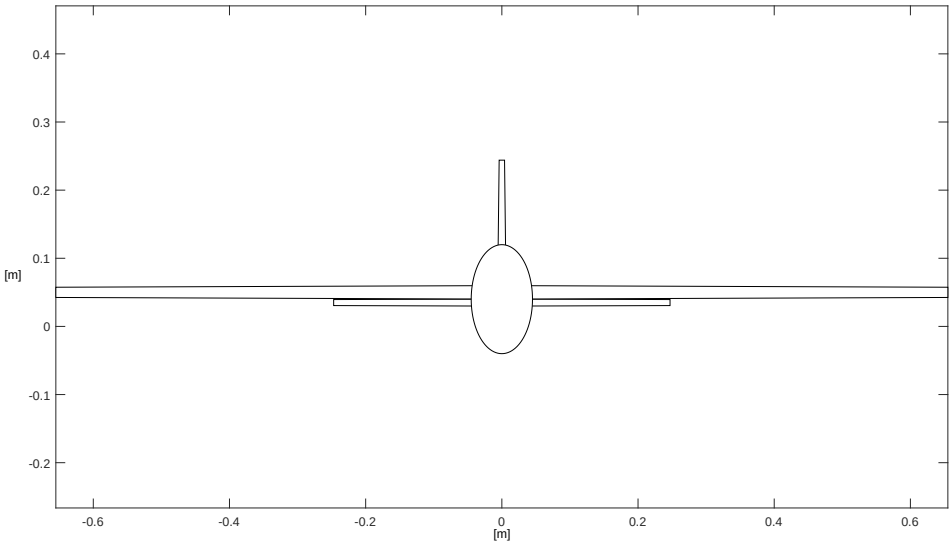


Figure 2.12: Front view of the DATCOM geometry input file

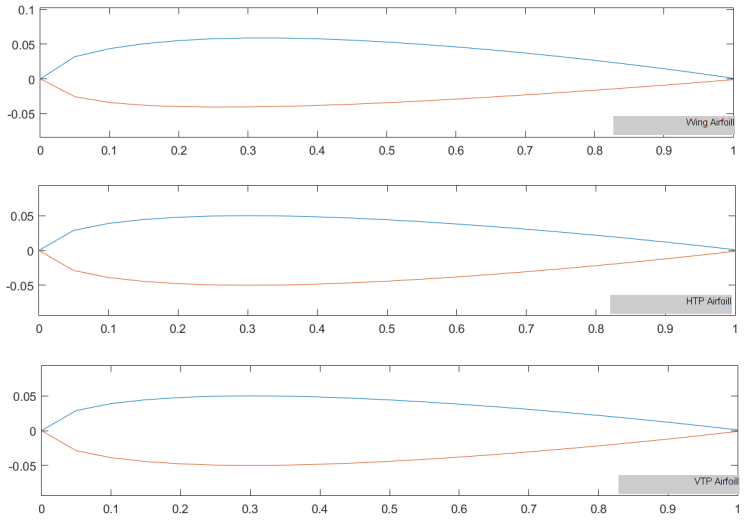


Figure 2.13: Airfoils of the DATCOM geometry input file

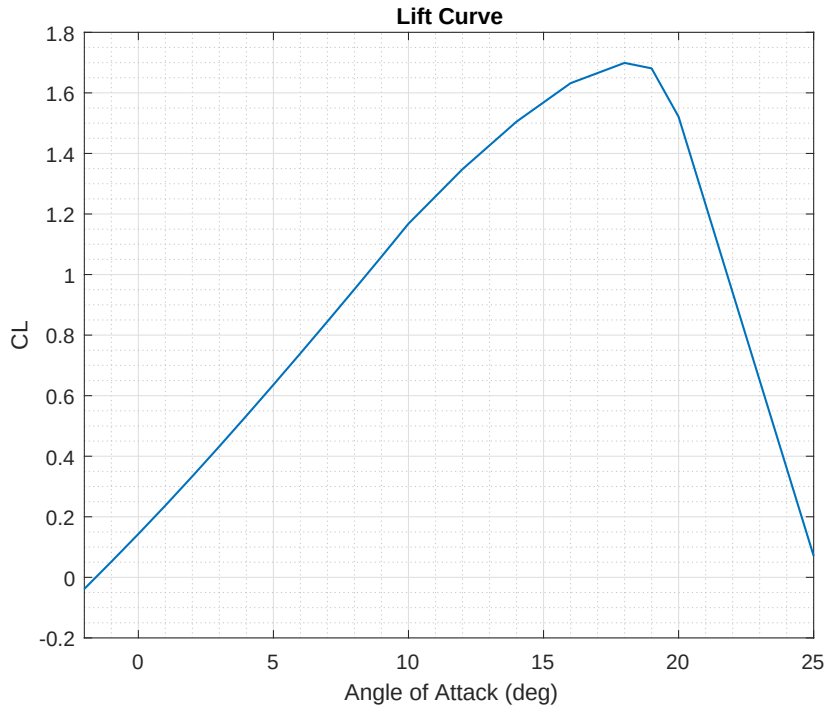


Figure 2.14: Lift coefficient curve as function of the angle of attack α

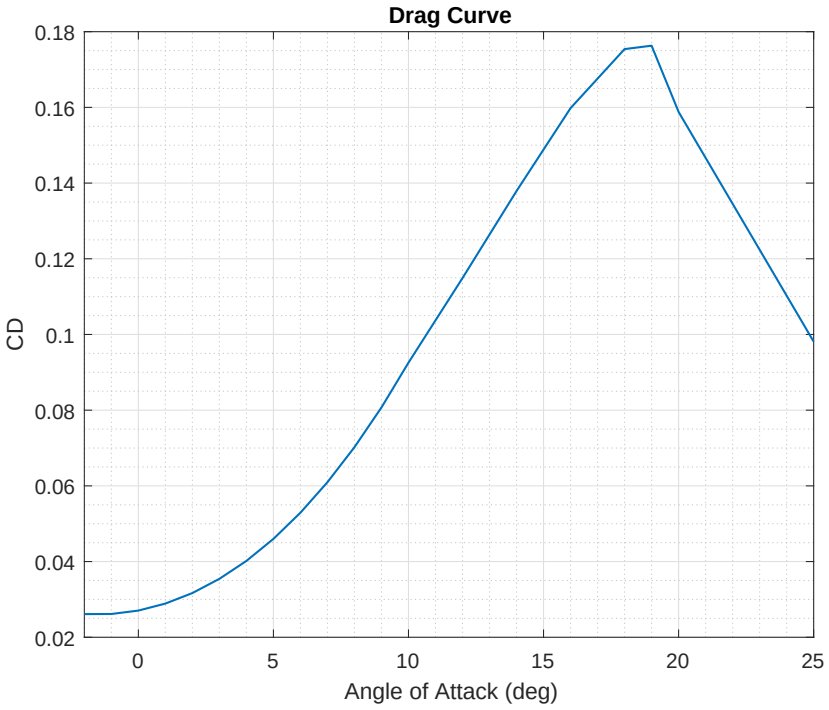


Figure 2.15: Drag coefficient curve as function of the angle of attack α

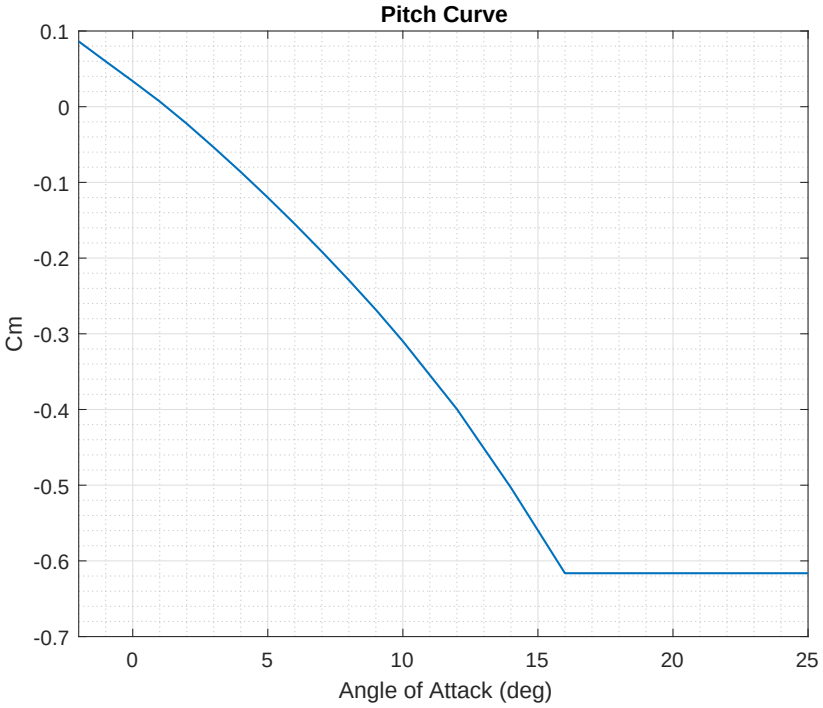
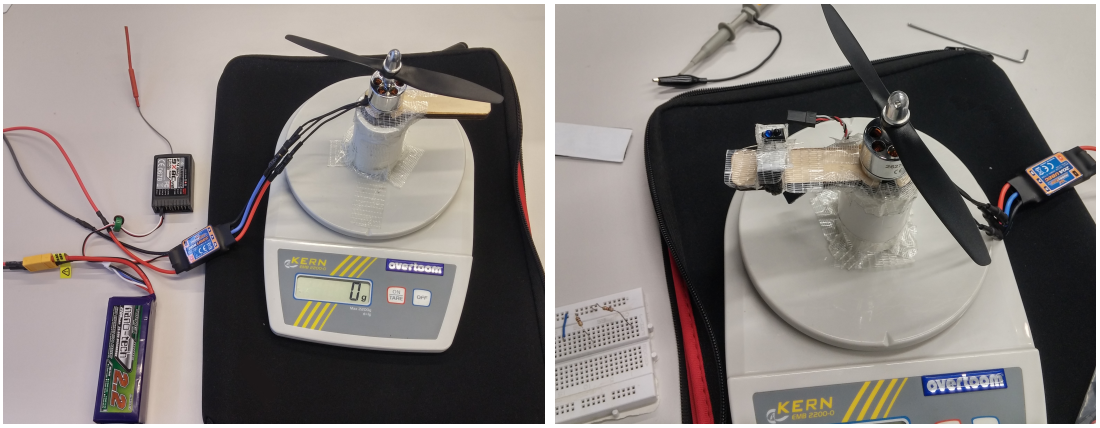


Figure 2.16: Pitch coefficient curve as function of the angle of attack α



(a) (b)

Figure 2.17: Testing platform for the motor and propeller coefficients estimation

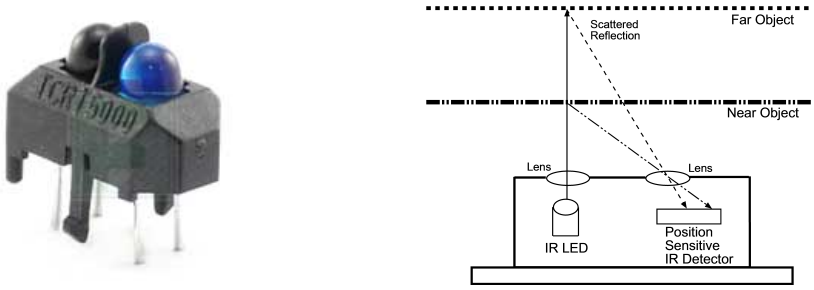
2.5.2 Propeller response identification

To obtain an estimate of C_{prop}, k_{motor}, and q_{motor} one has to obtain the thrust curve as function of propeller's speed rotation and the propeller speed as function of the commanded throttle. To do so an experiment was set up.

Testing platform

To measure the thrust, the motor with the propeller was placed, with a specific cylindrical support, on a digital balance (Figure 2.17a). The ESC which drove the motor, previously calibrated, was connected to the radio receiver and the battery.

Afterwards, a distance infrared sensor was placed under the propeller (Figure 2.17b). It is composed by a common infrared LED and a "receiver" photodiode (Figure 2.18a). Since the photodiode detector and the IR LED have a fixed distance and orientation relative to each other, the distance of an object will affect the angle at which the light from the IR LED hits the receiver (Figure 2.18b). By looking at where the light hits the detector, it is possible to calculate the angle of the light and from that angle derive the distance to the object. Upon connecting it to its specific signal conditioning circuit (Figure 2.19a), when the rotating propeller pass over the sensor, the output voltage of the photodiode is low, otherwise it is high, and in this way it produces as output a periodic signal. This output signal is then revealed by an oscilloscope: modern oscilloscopes have the capability of display the frequency of a periodic signal. At the bottom left of Figure 2.19b, it is possible to read the frequency of the signal, which is proportional to the rotational speed (e.g. if the propeller has two blades, the real frequency is half of the measured one).



(a) (b)

Figure 2.18: The Vishay TCRT5000 infrared sensor used for the test (left) and its functioning principle (right)

Static Response

In order to identify the coefficients described in section 1.5.3, for the test assume $V_a = 0$. The thrust equation can be written as:

$$F_P = K_T \Omega^2 \tag{2.2}$$

where $K_T = \frac{1}{2}\rho S_p C_{\text{prop}} R^2$. By fitting the test data with a parabolic curve, results in Figure 2.20 are obtained.

This implies that the estimated $\hat{K_T}$ and $\hat{C}_{\text{prop}}$ are:

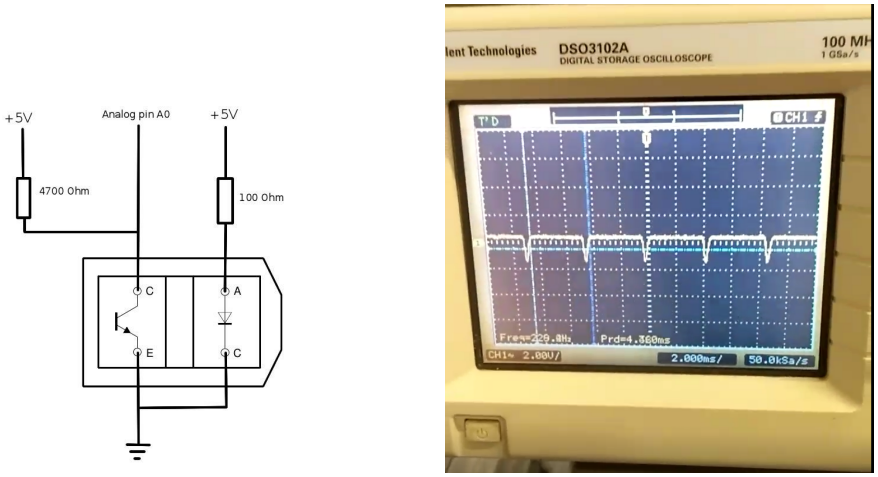
$$\hat{K_T} = 2.45 \cdot 10^{-5} \rightarrow \hat{C}_{\text{prop}} = 0.12. \tag{2.3}$$

By fitting a line with the Ω-δ_t data (Figure 2.21), the estimates $\hat{k}_{\text{motor}}$ and then $\hat{q}_{\text{motor}}$ can be obtained:

$$\hat{k}_{\text{motor}} = 11.39\,[\text{rad/s}], \qquad \hat{q}_{\text{motor}} = 239\,[\text{rad/s}]. \tag{2.4}$$

As one can notice, the 'goodness of fit' is not very high. This is caused mainly by the fact that the measurement instruments were not meant to be used this way: the balance during measurements suffered a lot for the vibrations caused by the motor rotation. A minor effect is also given by the the so-called 'ground effect' [13].

This model parameters uncertainties can be lowered by using a different test-bed (as an example using a load cell instead of the digital balance) or using ready-to-use professional system like the one in Figure 2.22. This equipment was unfortunately unavailable.



(a) (b)

Figure 2.19: The signal conditioning circuit(left) and the sensor fixing (right)

Dynamic response

The system dynamics of a fixed-wing UAV are extremely slower than the actuators. From [2] one can get an idea of the nature of the response and the involved time constants. The dynamics of the motor is modelled as a first order system:

$$G_{motor}(s) = \frac{\Omega(s)}{\delta_t(s)} = \frac{a}{s+a} \tag{2.5}$$

where $a = 45$.

2.5.3 Servos response

The servos datasheet reveal a settling time of 0.12s from 0 to 60deg. Servos have a time constant which is very close to the motor one, so, considering what said above, it can be approximated again with $G_{\text{servo}}(s) = G_{\text{motor}}(s)$.

2.6 Conclusions

As a conclusion of this chapter, the Simulink implementation principle of the aerodynamic forces is shown. In Figure 2.23 one can see an example of the DATCOM output tables with respect to α are interpolated to obtain the discussed coefficients.

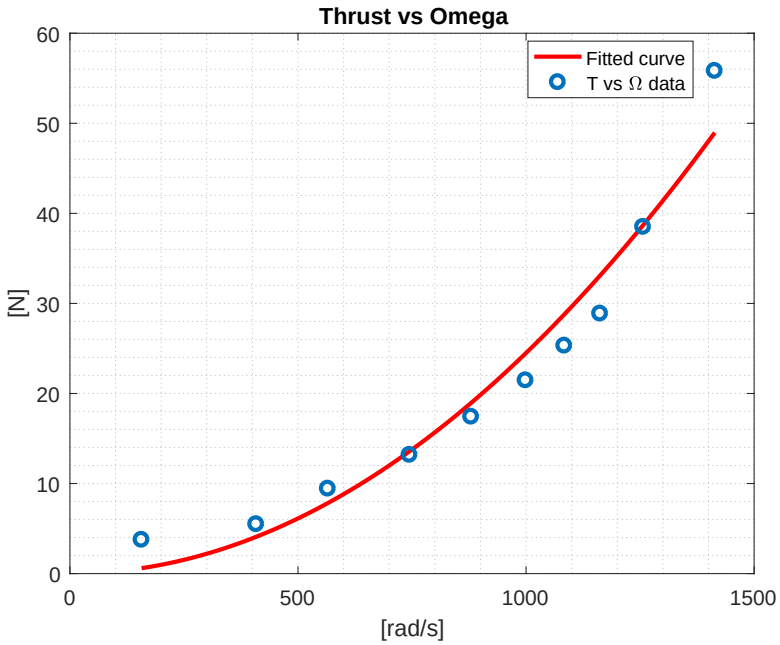


Figure 2.20: Thrust curve as function of the propeller angular speed

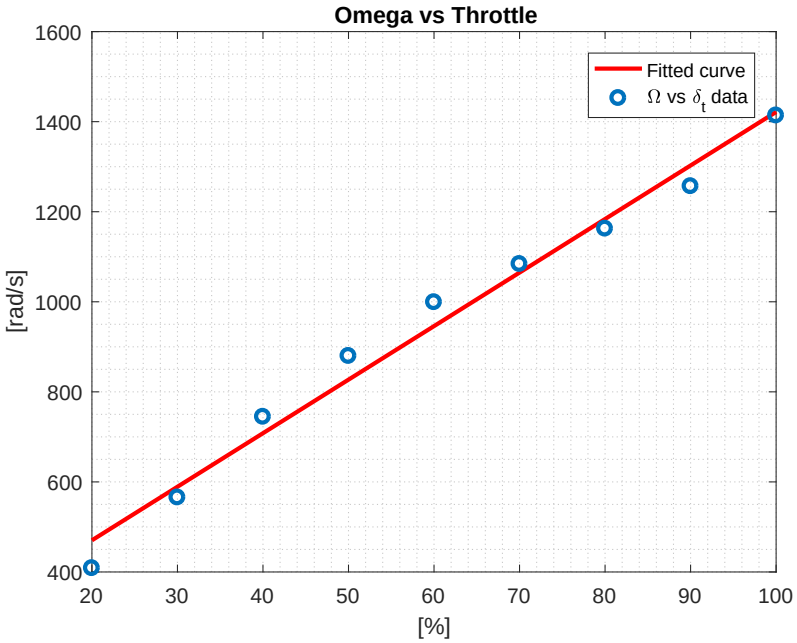


Figure 2.21: Propeller angular speed as function of the commanded throttle

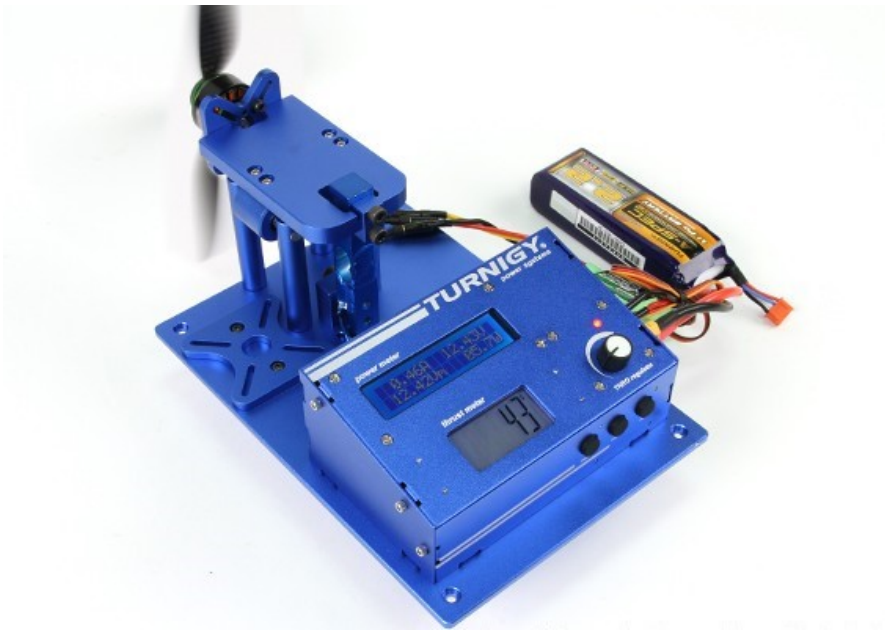


Figure 2.22: Turnigy thrust stand and power analyser

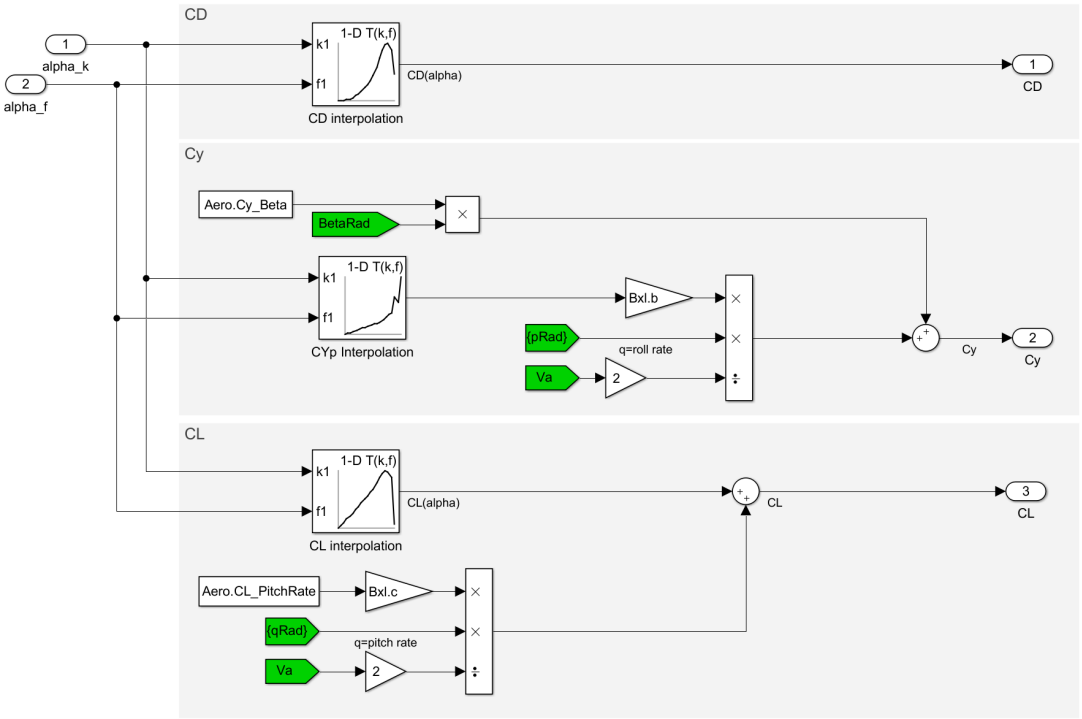


Figure 2.23: Implementation of the C_D, C_y, and C_L coefficients interpolations

Chapter 3

The autopilot

In this chapter the general autopilot control structure for a fixed-wing UAV, based on linear controllers, is firstly presented in Section 3.1. Getting acquainted with that is necessary to better understand the ArduPilot controllers structure. The latter will be discussed in Section 3.2. Lastly, some conclusions are drawn and the Simulink implementation is shown.

3.1 General structure of the autopilot

The primary goal in autopilot design is to control the inertial position (p_n, p_e, p_d) and attitude (ϕ, θ, ψ) of an aircraft. Let us express the nonlinear model of equation (1.37) in the form

$$\dot{x} = f(x, u) \tag{3.1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$,

$$u = \begin{bmatrix} \delta_a & \delta_e & \delta_r & \delta_t \end{bmatrix}^T, \tag{3.2}$$

$$x = \begin{bmatrix} p_n & p_e & h & u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}^T, \tag{3.3}$$

and $h = -p_d$. The system is in equilibrium if:

$$f(\bar{x}, \bar{u}) = 0 \tag{3.4}$$

The equilibrium condition $(\bar{x}, \bar{u})$ has to be determined in order to linearize the system about that point.

3.1.1 Trim condition

When a UAV is in constant-altitude wings-level steady flight, a subset of its states are in equilibrium. In particular:

$$\begin{bmatrix} \dot{h} \\ \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = 0 \tag{3.5}$$

In the flight dynamics literature, the aircraft is said to be in trim. In the process of performing trim calculations for the aircraft, the wind will be treated as an unknown disturbance, which makes $\mathbf{V}_a = \mathbf{V}_g$ and $\psi = \chi$.

Mathematically speaking, the objective is to compute trim states and inputs (*i.e.*, the couple $(\bar{x}, \bar{u})$) that makes the aircraft simultaneously satisfy the following three conditions:

- it is travelling at constant speed $\bar{V_a}$;
- it is travelling at a constant $\gamma_{fp} = (\theta - \alpha)$ angle;
- it is in a constant orbit of radius $\bar{R} \in [R_{min}, +\infty]$, with $R_{min} > 0$.

Angle γ_{fp} is called flight path angle. From these conditions, it is possible to see that there exist a countable subset of equilibrium points. From this subset are obviously excluded all the conditions which do not allow for a regular flight (*e.g.*, an airspeed less than the minimum stall airspeed, a too high α, etc.). Therefore linearization should be conducted about the right operating point time by time. It is remarkable that looking at equations (1.37), the trimmed flight is independent of p_n, p_e, p_d, and ψ (since p_n and p_e depend on ψ).

3.1.2 Linearized model

For most flight manoeuvres of interest, autopilots are designed with the assumption of decoupled lateral and longitudinal dynamics. In this way, the structure and the development of an autopilot significantly simplifies and allows the use of successive loop closures, yielding good overall performance.

For the lateral dynamics, the variables of interest are the roll angle ϕ, the roll rate p, the heading angle ψ, and the yaw rate r. The control surfaces used to

influence the lateral dynamics are the ailerons and the rudder. Ailerons primarily influence the roll rate p; additionally, both ailerons and rudder influence the yaw angle ψ.

Similarly, the variables of interest for the longitudinal dynamics are the pitch angle θ, the pitch rate q, the altitude h and the airspeed V_a. The control signals used to influence the longitudinal dynamics are the elevator δ_e, and the throttle δ_t. The elevator is used to directly influence the pitch angle θ. In turn, the pitch angle can be used to manipulate both the altitude and the airspeed. Vice versa, the throttle influences the airspeed and the altitude. Therefore there are some cross effects, which will be discussed in greater detail later.

Beard [14], shows that, with the above assumption of decoupled dynamics, the transfer functions of the linearized lateral dynamics about the equilibrium $(\bar{x}, \bar{u})$, defined as in Section 3.1.1, are:

$$\text{Roll angle} \qquad \phi(s) = \frac{a_{\phi 2}}{s(s + a_{\phi_1})} \left(\delta_a(s) + \frac{1}{a_{\phi 2}} d_{\phi 2}(s) \right) \tag{3.6}$$

$$\text{Course angle} \qquad \chi(s) = \frac{g/V_g}{s} \left(\phi(s) + d_\chi(s) \right) \tag{3.7}$$

where $d_{\phi_2}(s)$ and $d_\chi(s)$ are intended as disturbances whereby the cross-effects of neglected dynamics and wind action come into play. Coefficients a_{ϕ_1} and a_{ϕ_2} are scalars coming from the linearization.

The transfer functions for the linearized longitudinal dynamics, instead, are:

$$\text{Pitch angle} \qquad \theta(s) = \frac{a_{\theta_3}}{s^2 + a_{\theta_1} s + a_{\theta_2}} \left(\delta_e(s) + \frac{1}{a_{\theta 3}} d_{\theta_2}(s) \right) \tag{3.8}$$

$$\text{Height(1)} \qquad h(s) = \frac{V_a}{s} \left(\theta(s) + \frac{1}{V_a} d_h \right) \text{ if } V_a \text{ constant} \tag{3.9}$$

$$\text{Height(2)} \qquad h(s) = \frac{\theta}{s} \left(V_a(s) + \frac{1}{\theta} d_h \right) \quad \text{if } \theta \text{ constant} \tag{3.10}$$

$$\text{Airspeed} \qquad V_a^*(s) = \frac{1}{s + a_{V_1}} \left(a_{V_2} \delta_t^*(s) - a_{V_3} \theta^*(s) + d_V(s) \right) \tag{3.11}$$

where $V_a^* = V_a - \bar{V}_a$ is the deviation of V_a from the trim $\bar{V}_a$, $\theta^* = \theta - \bar{\theta}$ is the deviation of θ from the trim $\bar{\theta}$, and $\delta_t^* = \delta_t^* - \bar{\delta}_t$ is the deviation of δ_t from the trim $\bar{\delta}_t$. Again, parameters with letter a are scalar coefficients and inputs $d_{\theta_2}(s)$, $d_h(s)$ and $d_V(s)$ are the disturbances from the neglected dynamics and the atmospheric disturbances.

The linearized dynamics equations put in evidence how all the transfer functions are of the first or second order. Therefore, conventional PID controllers or lead-lag compensators can be employed effectively.

3.1.3 General scheme

From equations (3.9), (3.10) and (3.11), one can see a cross-coupling between inputs δ_e and δ_t, with θ and V_a outputs. It is therefore necessary to decide which is the most effective control input to regulate h and V_a according to the current flight regime. A reasonable regimes division can be: climb zone, altitude hold zone, descent zone. One could use full throttle and zero throttle in the climb zone and descent zone respectively, and regulate the airspeed with the pitch angle. In the altitude hold zone, the simplest and most effective solution is to regulate altitude by commanding pitch, and the airspeed by commanding the throttle. For simplicity the latter situation will be taken into account now. In the second part of the chapter, a more effective way, implemented in the ArduPilot, will be presented.

To be thorough, concerning the lateral control, there exists another (even if not essential) control loop which employs the rudder deflection δ_r to bring the side-slip angle β to zero.This is a valuable flight condition for manned flights because gives a null lateral acceleration and then more passengers comfort. The transfer function from the rudder deflection to the side-slip angle is given by:

$$\text{Side-slip} \qquad \beta(s) = \frac{a_{\beta_2}}{s + a_{\beta_1}} \left(\delta_r(s) + d_\beta(s)\right). \tag{3.12}$$

Define now:

$$P_\phi(s) = \frac{a_{\phi_2}}{s(s + a_{\phi_1})}\delta_a(s) \qquad P_\chi(s) = \frac{g/V_g}{s}\phi(s) \tag{3.13}$$

$$P_\theta(s) = \frac{a_{\theta_3}}{s^2 + a_{\theta_1}s + a_{\theta_2}}\delta_e(s) \qquad P_h(s) = \frac{V_a}{s}\theta(s) \tag{3.14}$$

$$P_{V_a}(s) = \frac{1}{s + a_{V_1}}a_{V_2}\delta_t^*(s) \qquad P_\beta(s) = \frac{a_{\beta_2}}{s + a_{\beta_1}} \tag{3.15}$$

and $C_\phi(s)$, $C_\chi(s)$, $C_\theta(s)$, $C_h(s)$, $C_{V_a}(s)$ and $C_{\beta(s)}$ six PID controllers. Input disturbances are summarized compactly as $\tilde{d}_\phi$, $\tilde{d}_\chi$, $\tilde{d}_\theta$, $\tilde{d}_h$, $\tilde{d}_{V_a}$ and $\tilde{d}_\beta$. It is possible to see the overall control architecture in Figure 3.1.

As known, successive loop closures work better if the dynamics of the inner and outer loops are frequency decoupled [15], [16]. This can be obtained with a wise tuning of the PID controllers.

3.2 ArduPilot control scheme

The inner loops for the roll, pitch and side-slip are introduced first. The airspeed and height control make use of an energy-based method and will be subsequently

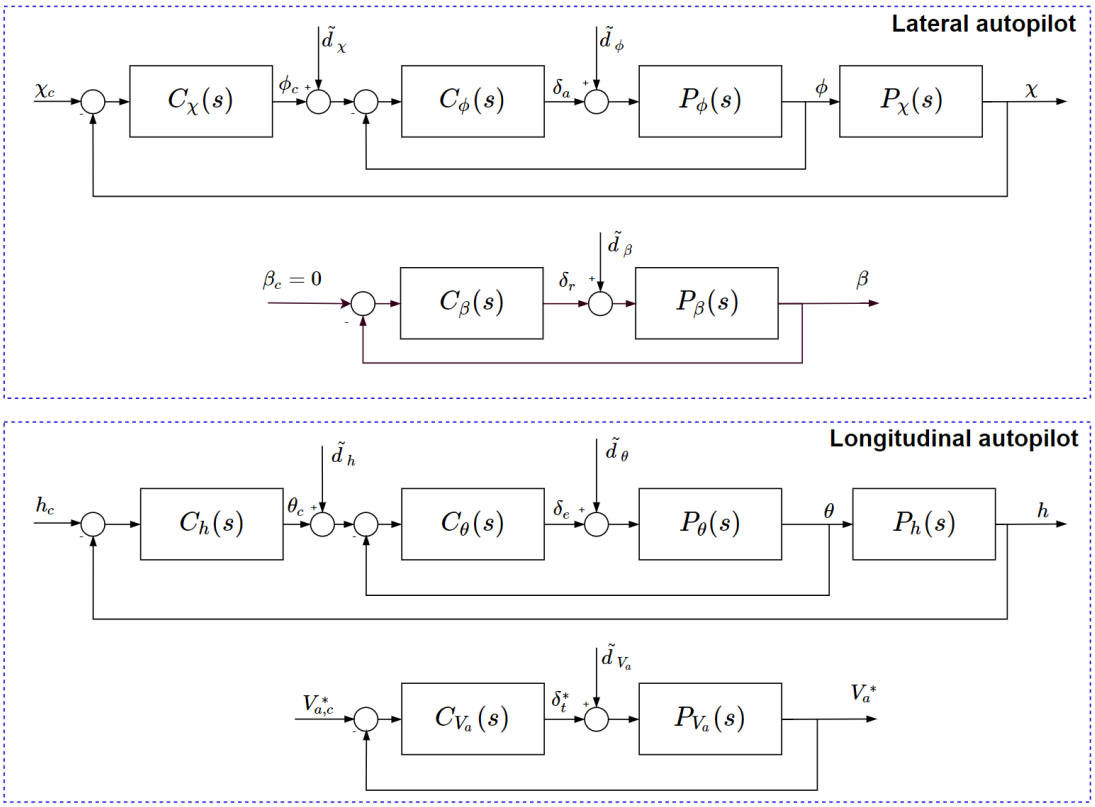


Figure 3.1: General autopilot control scheme

described. Being implemented on a micro-controller obviously implies that controllers are discrete time.

Before moving on, a brief introduction about the ArduPilot code structure is given. By looking at Figure 3.2, one can notice different layers:

- At the bottom of the figure, there are all the supported boards, including both open-hardware and closed-hardware, and the sensors. They are provided with a real time operating system, where it is possible to run the ArduPilot software.

- It is then clear that ArduPilot code must be capable of being compiled according to the user hardware. This task is executed by the hardware abstraction layer (or HAL). The choice of a hardware is usually independent from the type of vehicle. Since ArduPilot can be used on different aerial vehicles, it has a set of shared libraries that can be accessed from the vehicle specific flight code. This provides a useful generalization, for example, to the sensor libraries, the Extended Kalman Filter, or the PID controller, aimed to reduce the coding effort and time. The vehicle-specific code for aircraft is named as ArduPlane.

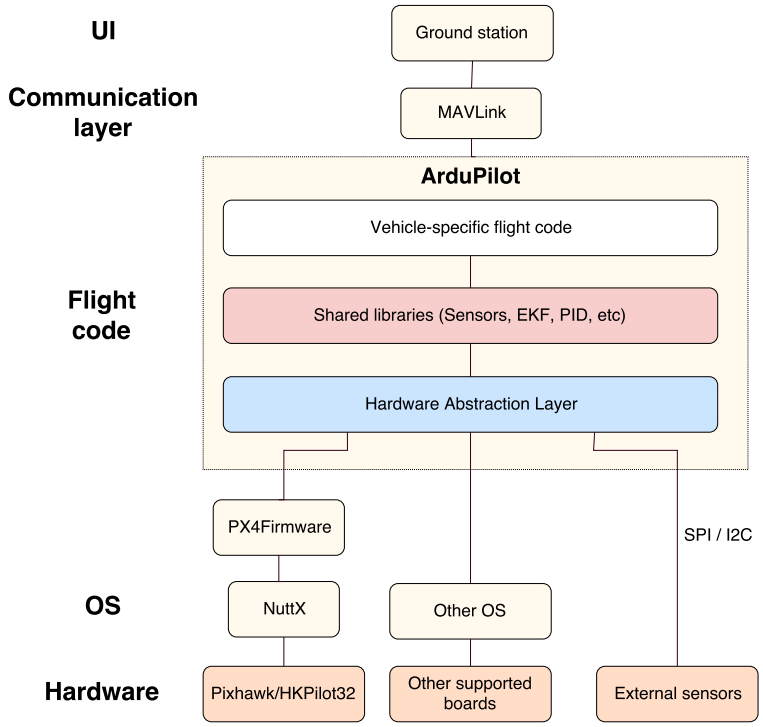


Figure 3.2: ArduPilot code structure

- Finally, the MAVLink protocol guarantees the communication between the vehicle and the ground station.

This structure has been introduced to face the rapid increase of popularity of the software, both among users and developers. Being developed from a large number of independent developers, the code is meant to be as general as possible to easily welcome improvements in its performance and features.

Of course, this uniforms some aspects, like the control strategy. The attitude controllers for both aircraft and molti-rotor vehicles are the PIDs, whose library is then shared.

Taking as example the roll control, ArduPilot developers originally implemented, in the older versions (till about 2015), the classical control scheme of Figure 3.3, where $C_{1_\phi}(z)$ is a discrete PID controller in the form:

$$C_{1_\phi}(z) = \left(K_{P_\phi} + K_{I_\phi} T_s \frac{1}{z-1} + K_{D_\phi} \frac{N}{1 + N T_s \frac{1}{z-1}} \right) e_\phi(z) \tag{3.16}$$

provided with a saturation and anti wind-up action. This follows the general structure given in the previous section. During ArduPilot evolution, this simple control loop (based on a ϕ estimate) had been afterwards changed. This can be caused by unsatisfactory performances, but also by the will to introduce nested loops to directly exploit the angular rate readings from the gyros. Dealing with

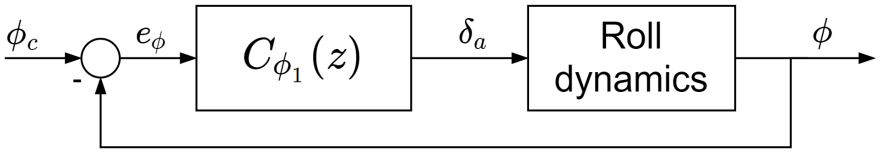


Figure 3.3: ArduPilot old roll controller

the angular rates in the inner loop gives also a significant benefit that will be analysed when introducing the pitch control loop, because it allows for an easy implementation of the coordinated turn.

The new control structure is depicted in Figure 3.4. For what discussed above, the structure is naturally the same for the pitch control loop, with the appropriate modifications.

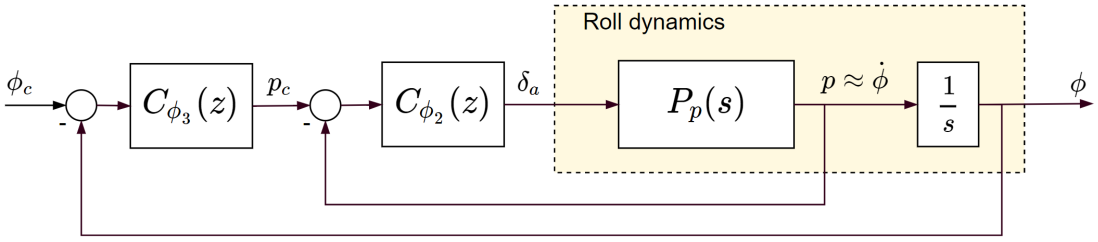


Figure 3.4: New ArduPlane control scheme

The assumption of $p \approx \dot{\phi}$ was made. It is worth remarking that from the expression of (ϕ, θ, ψ) as function of (p, q, r) (equation (1.24)), we have:

$$p = \dot{\phi} - \dot{\psi} \sin \theta. \tag{3.17}$$

Since θ is usually small, the assumption is reasonable.

3.2.1 Roll control loop

When the control structure was modified, the new update was meant to be rolled out inside a firmware upgrade. Nevertheless, the developers wanted that the users could maintain the same values of K_{P_ϕ}, K_{I_ϕ}, and K_{D_ϕ}, while moving to a cascade control of p in the outer loop and ϕ in the inner loop.

To operate the conversion, new parameters need to be introduced:

$$\Omega_\phi = \frac{1}{\tau_\phi} \tag{3.18}$$

$$\tilde{K}_{P_\phi} = (K_{P_\phi} - K_{I_\phi}\tau)\tau - K_{D_\phi} \tag{3.19}$$

$$\tilde{K}_{I_\phi} = K_{I_\phi}\tau \tag{3.20}$$

where $\tau_\phi \in [0.4, 1]$ is a new tuning parameter (which can be seen as the time constant in seconds from demanded to achieved bank angle, see [17]). In its present configuration the scheme is the one in Figure 3.5.

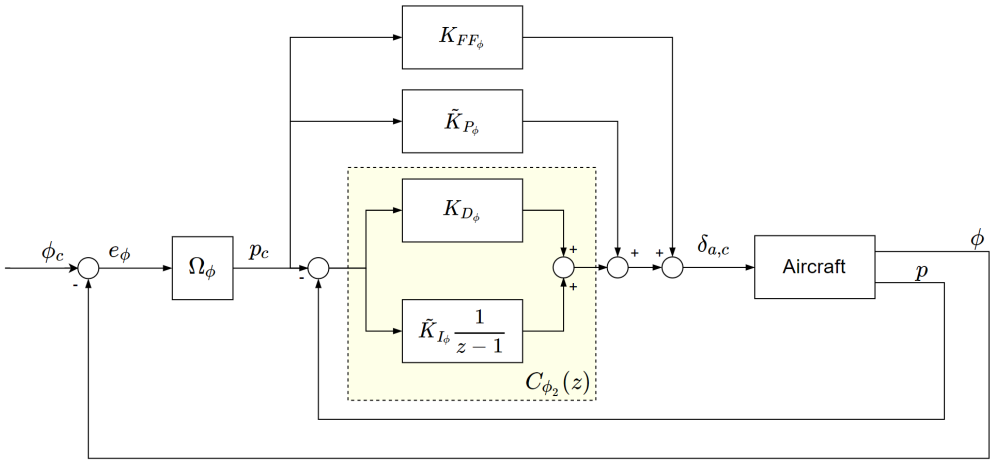


Figure 3.5: ArduPilot general scheme for the roll control loop

Moreover, the pure derivative action of the old PID disappear. There is also an optional feed-forward action with gain K_{FF_ϕ} that is not going to be used here and therefore set to zero.

Lastly a scaling action is introduced to moderate the ailerons control action according to the airspeed: at high speed the surfaces are moved less, and at low speed are moved more. The scaling equation is:

$$\text{scaler} = \frac{V_{a,\text{scaling}}}{V_a} \tag{3.21}$$

where $V_{a,\text{scaling}}$ is usually set at the wanted cruise speed (used value is 15 m/s). The final control scheme represented in Figure 3.6.

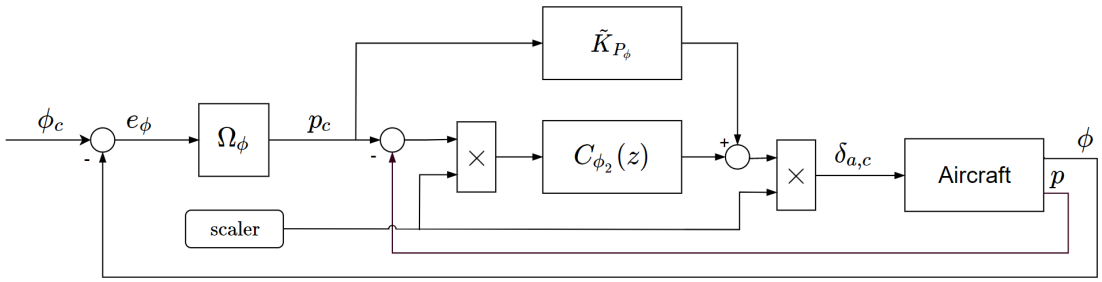


Figure 3.6: Complete ArduPilot scheme for the roll control loop

3.2.2 Pitch control loop

The pitch control loop follows the principles explained previously for the roll. There is only one addition given by a pitch rate correction when the roll angle is not zero. This is called turn compensation [18].

Turn compensation

In straight and level flight, the lift force acts upwards to counteract the weight force of the aircraft. During a turn, the lift acts at an angle ϕ away from the plane $(\mathbf{i}^v, \mathbf{j}^v)$, as in Figure 3.7. It is useful to resolve the lift into a vertical component and a horizontal component. If one wants the aircraft to continue at constant altitude, the lift vertical component must continue to equal the weight of the aircraft.

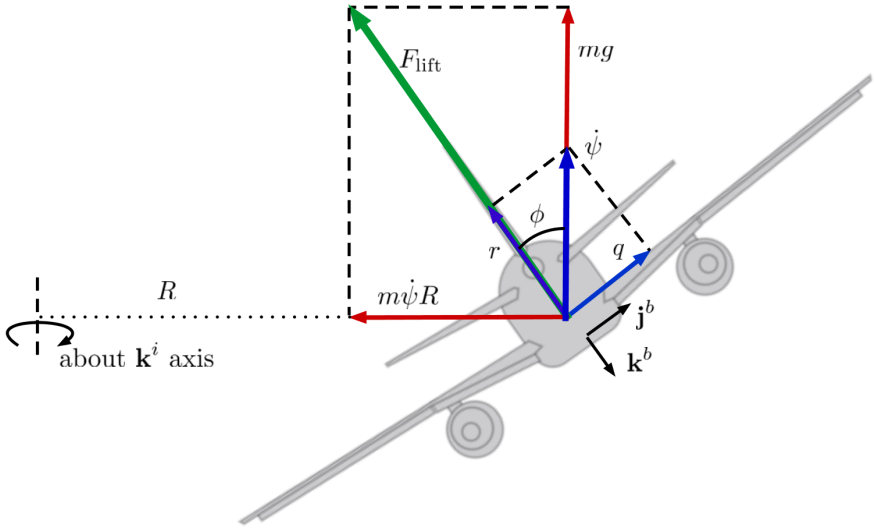


Figure 3.7: Gravity and lift force during a turn

To correct for this, and avoid losing altitude, it is necessary to increase the angle of attack, thus increasing the total lift force produced by the wings of the aircraft. The lift vector has to be the vector sum of the weight vector mg and the centripetal force $m\dot{\psi}^2 R = mV_a\dot{\psi}$, where R is the radius of curvature of the circular path of the aircraft.

In fact, in absence of wind, summing forces in the horizontal direction gives:

$$F_{\text{lift}} \sin\phi = mV_a\dot{\psi}. \tag{3.22}$$

Similarly, the vertical component must be equal to the gravity force:

$$F_{\text{lift}} \cos\phi = mg. \tag{3.23}$$

Dividing (3.22) by (3.23) yields

$$\dot{\psi} = \frac{g}{V_a} \tan \phi. \tag{3.24}$$

that is the only value of yaw rate for which the coordination can be achieved. In [14], it is shown that equation (3.24) holds true also in the presence of wind (without down component).

Assuming $\theta = 0$, we know from the equation (1.25) that $\dot{\psi}$ is expressed as function of both q and r (Figure 3.7). Therefore the pitch rate due to $\dot{\psi}$ is:

$$q = \dot{\psi} \sin \phi \tag{3.25}$$

that for a given value of $\dot{\psi}$ is just the pitch rate q necessary for turn compensation. Substituting (3.24) into (3.25), the following equation is obtained:

$$q = \frac{g}{V_a} \tan \phi \sin \phi. \tag{3.26}$$

Finally, for a non null pitch angle, the implemented pitch correction is given by:

$$q_{coord} = \cos \theta \left| \frac{g}{V_a} \tan \phi \sin \phi \right| k_{q_{\text{coord}}} \tag{3.27}$$

where $k_{q_{\text{coord}}}$ is a tunable gain parameter normally set to 1.

The pitch control scheme

The pitch control scheme is given in Figure 3.8.

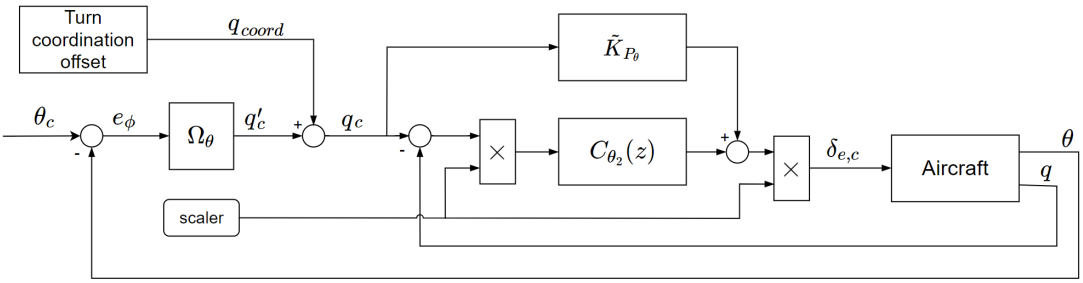


Figure 3.8: Complete ArduPilot scheme for the pitch control

It is remarkable that, with the same principle of the roll control loop, the scheme works with the assumption that $\dot{\theta} \approx q$. However, q can be directly obtained from the gyros measurements.

3.2.3 Side-slip control loop

The rudder management is usually focused on controlling the side-slip and bringing it to zero by means of rudder deflections. However, in ArduPilot the rudder control implements also other two functions:

- a yaw damping;
- a mixing action with the commanded roll angle.

The yaw damper is intended for aircraft with an inadequate vertical stabiliser surface, whose tail tend to 'skid' downward in the $(\mathbf{i}^b, \mathbf{j}^b)$ plane during a rolling motion. The mixing action is called *Aileron-Rudder Interconnect*, or ARI. The objective of the ARI is to cancel out the adverse yaw caused by aileron deflections (it is called 'adverse' yaw because positive aileron yields negative roll acceleration, but positive yaw acceleration). It is defined as $\delta_{r_{mix}} = k_{r_{mix}} \delta_{a,c}$, with $k_{r_{mix}}$ a tuning parameter usually set to 0.5.

Side-slip control is not taken into account at this stage of the project. Since the side-slip angle control is not essential, and the airframe is well designed, only the ARI contribution remains.

For the sake of completeness, as for the side-slip angle control, recalling the coordinated turn, from Figure 3.7 and equation (3.24), the yaw rate that brings to zero the lateral force is:

$$r = \frac{g}{V_a} \sin\theta. \tag{3.28}$$

The ArduPilot implemented version is:

$$r_{coord} = \frac{g}{V_a} k_{r_{coord}} \sin\theta \tag{3.29}$$

where $k_{r_{coord}}$ is a tunable gain parameter normally set to 1. The controller is an integrator with gain k_{β_I}, which takes as input $e_r = r - r_{coord}$ filtered with a high-pass filter. A corrective additive term sums up to e_r if the performance are not satisfactory with respect to the lateral acceleration (measured by the accelerometer), using the tunable the parameter $k_{r_{slip}}$ (normally set to zero).

The overall control scheme is shown in Figure 3.9.

3.2.4 The Total Energy Control System

In the general scheme of Figure 3.1, the airspeed was controlled by means of δ_t and the height with δ_e. Anyway, from equations (3.9), (3.10) and (3.11), it is evident that the altitude dynamics and the airspeed dynamics are not decoupled. Consider the case when an aircraft pitches up while the thrust does not change. If the dynamics were truly decoupled, the aircraft's altitude would increase while the airspeed would remain unchanged. However, the airspeed will obviously decrease

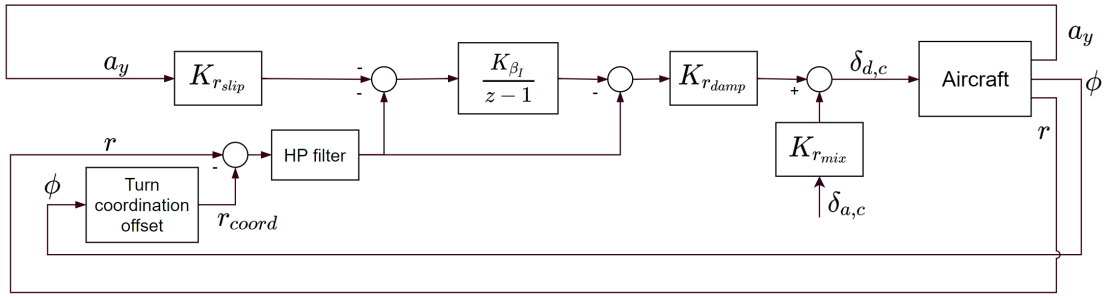


Figure 3.9: Complete ArduPilot scheme for the rudder surface deflection

while the altitude increases. In other words, some of the aircraft's kinetic energy is converted to potential energy. Decoupled dynamics lead to poor performance with respect to other methods based on energy considerations. One of these is called Total Energy Control System, or TECS, and throughout the years has been deeply analysed and improved, for example with nonlinear or adaptive design technique (the interested readers can start with [19]).

In the following, the ArduPilot implementation of TECS will be described.

Energy definitions

By definition, the specific kinetic and potential energies for a point mass object are given by:

$$e_k \triangleq \frac{1}{2}V_a^2 \tag{3.30}$$

$$e_p \triangleq gh. \tag{3.31}$$

Similarly, specific kinetic and potential energy rates are:

$$\dot{e}_k \triangleq V_a\dot{V}_a \tag{3.32}$$

$$\dot{e}_p \triangleq g\dot{h}. \tag{3.33}$$

The demanded height and airspeed define the energy quantities as follows

$$e_{k,c} = \frac{1}{2}V_{a,c}^2 \tag{3.34}$$

$$e_{p,c} = gh_c \tag{3.35}$$

$$\dot{e}_{k,c} = V_{a,c}\dot{V}_{a,c} \tag{3.36}$$

$$\dot{e}_{p,c} = g\dot{h}_c. \tag{3.37}$$

It is now possible to introduce the specific total energy as:

$$e_T = e_k + e_p \tag{3.38}$$

and the specific total energy rate as:

$$\dot{e}_T = \dot{e}_k + \dot{e}_p = g\dot{h} + V_a\dot{V}_a. \tag{3.39}$$

Usually TECS controllers make use of a scaled version of $\dot{e}_T$ defined as:

$$\dot{e}'_T = \frac{\dot{e}_T}{gV_a} = \frac{\dot{V}_a}{g} + \frac{\dot{h}}{V_a} \tag{3.40}$$

In absence of wind, the angle between $\mathbf{V}_g$ and the $(\mathbf{i}^v, \mathbf{j}^v)$ plane (*i.e.*, the horizontal plane), is the flight path angle γ_{fp}. So (recalling that $V_g = ||\mathbf{V}_g||_2$),

$$\dot{h} = V_g \sin\gamma = V_a \sin\gamma. \tag{3.41}$$

Therefore, for small γ angles, equation (3.40) becomes:

$$\dot{e}'_T = \frac{\dot{V}_a}{g} + \gamma. \tag{3.42}$$

Now, considering the aircraft as a point mass, assuming that γ and α are small and the directions of the thrust and drag are aligned, the forces balance is given by the expression

$$F_p - D = g\dot{e}'_T \tag{3.43}$$

where F_p is the thrust force and D the drag. Therefore the correct thrust control strategy is to develop incremental thrust command as follows:

$$\Delta F_p = g\Delta\dot{e}'_T. \tag{3.44}$$

This demonstrates how $\Delta F_p \propto \Delta\dot{e}'_T$ and how altering the thrust will proportionally alter the specific rate of energy into the airplane, increasing the sum of the flight path angle and the acceleration along the flight path. Therefore thrust is used the total energy control.

In parallel, it is known that the elevator deflection is energy conservative with a good approximation. This allows to exchange potential energy for the kinetic energy, and vice versa, using the elevator, that, in conclusion, is can be interpreted as an energy distribution control.

Pitch and thrust control

ArduPilot control scheme uses two PID controllers to generate the throttle and pitch commands.

The thrust controller combines a PID with a feedforward term. The feedforward is given by:

$$T_{ff} = T_D + k_{T,ff}\dot{e}_{T,c} + k_{T_\phi}\left(\frac{1}{\cos^2\phi} - 1\right) \tag{3.45}$$

where T_D is the trim thrust needed to counteract the drag force, $k_{T,ff}$ is a parameter that control the feedforward amount and k_{T_ϕ} is a parameter that accounts for the increased drag during the aircraft banking.

The PID, based on the total energy rate, is written in the time domain as:

$$\delta_t = T_{ff} + K_{p_{thr}}\tilde{e}_T + K_{d_{thr}}\dot{\tilde{e}}_T + K_{I_{thr}}\int_{t_0}^{t}\tilde{e}_T(\tau)d\tau \tag{3.46}$$

where $\tilde{e_T} = e_{T,c} - e_T$ and $e_{T,c} = e_{k,c} + e_{p,c}$.

Concerning the pitch controller, the difference between the potential energy and the kinetic energy (or energy balance) will be hereby denoted as $e_b = e_p - e_k$. In ArduPilot the pitch controller uses a scaled version of the energy difference:

$$e'_b = (2 - \epsilon_{ke})e_p - \epsilon_{ke}e_t = (2 - \epsilon_{ke})gh - \epsilon_{ke}\frac{V_a^2}{2} \tag{3.47}$$

where $\epsilon_{ke} \in [0, 2]$ is a weighting factor for the potential and kinetic energies that determines the altitude and airspeed control priority. A $\epsilon_{ke} = 0$ means that only the altitude error is used and full priority is given to the height control (for example when no airspeed measurement is available) and $\epsilon_{ke} = 2$ means that only the airspeed error is used, thus prioritizing the airspeed control (for example when an underspeed condition verifies). Normally, $\epsilon_{ke} = 1$. The commanded pitch is generated by a PID with a feedforward term and is given by:

$$\theta_c = \frac{1}{V_a}\left(K_{P_\Theta}\tilde{e}_b + \frac{\dot{e}'_{b,c}}{g} + K_{D_\Theta}\dot{\tilde{e}}_b + K_{I_\Theta}\int_{t_0}^{t}\tilde{e}_b(\tau)d\tau\right) \tag{3.48}$$

where $\tilde{e}_b = e'_{b,c} - e'_b$ and $e'_{b,c}$ is the weighted commanded energy balance in the fashion of equation (3.47).

3.3 ArduPilot state estimation

In most flight controllers not all the quantities of interest can be directly measured and therefore a state estimator is required. In Ardupilot, this is done using an Extended Kalman Filter, which estimates vehicle position (p_n, p_e, p_d), velocity

(u, v, w) and angular orientation (ϕ, θ, ψ) based on rate gyroscopes, accelerometer, compass, GPS, airspeed and barometric pressure measurements.

The flight controller has two IMUs available, so ArduPilot allows to use two EKF "cores" (*i.e.*, two instances of the EKF) that run in parallel, each using a different IMU. At every update, only the output from a single EKF core is used, that core being the one that reports the best health which is determined by the consistency of its sensor data. The only quantities directly exploited are the gyros angular rates measurements (and a_y in the discussed adjustment to achieve the coordinated turn).

It is worth remarking that the expression of $\dot{e}_k$ contains $\dot{V}_a$. In ArduPilot it was preferred to avoid to compute the derivative the airspeed sensor measurements, but to take the accelerometer measurement on the $\mathbf{i}^b$ axis and subtract the contribution given by the gravity, hence formally:

$$\dot{V}_a = a_x - g \sin\theta. \tag{3.49}$$

The EKF and the sensor dynamics are not implemented in the simulator at this project stage, but can be taken into account in future developments.

3.4 Implementation

The ArduPilot controllers are implemented on the Simulink simulator. Being discrete, it was necessary to obtain the time sampling for the low-level layer, which is of 400Hz.

One may notice that in the general lateral autopilot scheme, the course angle χ controller was introduced with an external loop too. Anyway, the choice of this outer control loop strongly depends on which guidance low is implemented in the upper layer. As will be analysed, ArduPilot do not implements a similar guidance law, because it makes use of a lateral acceleration command in place of χ_c to generate the roll angle reference [20]. In the Vector Field strategy, a course angle command χ_c is given to low-level controllers, so the outer lateral control loop has to be synthesised from scratch from the input χ_c to the output ϕ_c.

The PID gains and the other parameters are obtained after the controllers in-flight tuning of the real UAV and reported in Appendix D.

In Figures 3.10, 3.11, 3.12, and 3.13, the Simulink implementation is shown. Proceeding bottom-up, as an example, in Figure 3.10 one can see the PID implementation for the pitch control, which is embedded inside the pitch controller in Figure 3.11. In Figure 3.12 the structure of the roll, pitch and rudder deflection controller is then shown, while the final overview of the autopilot layer is given in Figure 3.13.

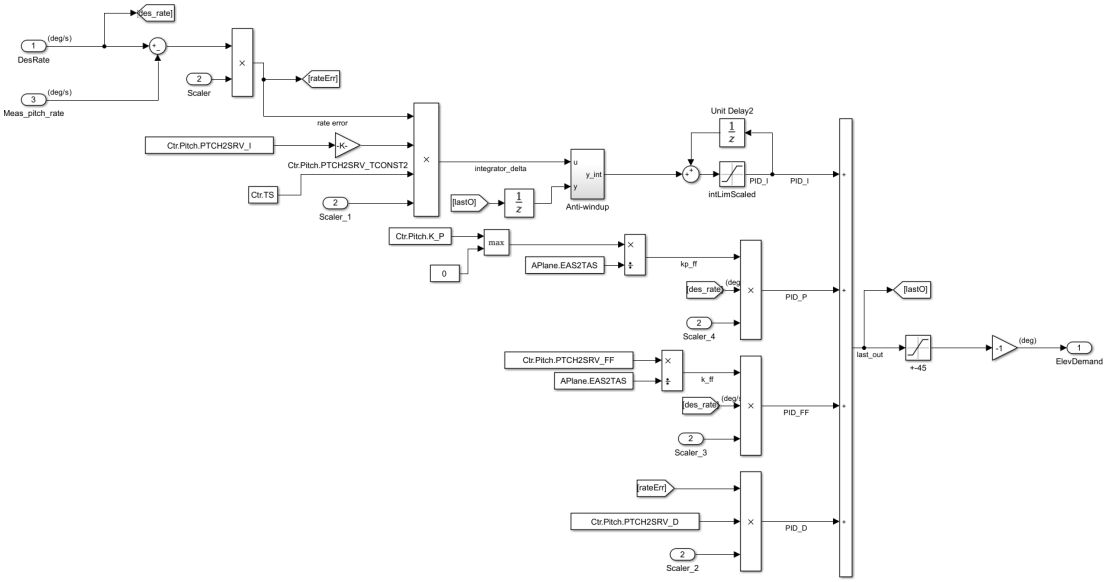


Figure 3.10: ArduPilot PID Simulink implementation

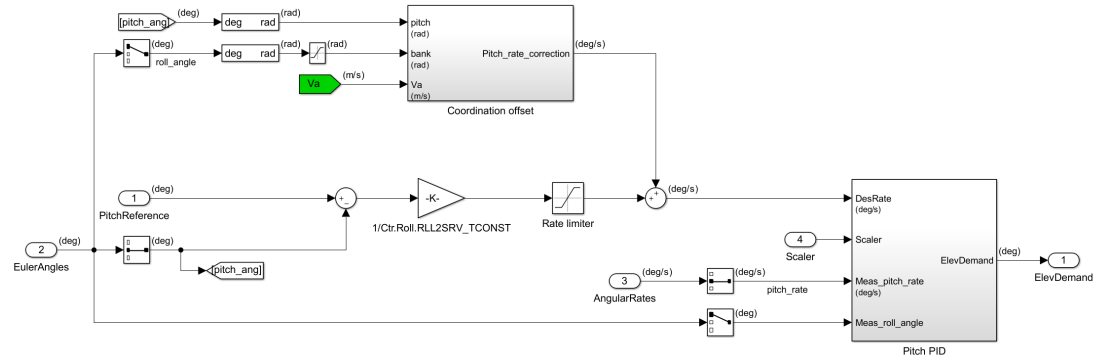


Figure 3.11: Exemplifying pitch controller Simulink implementation

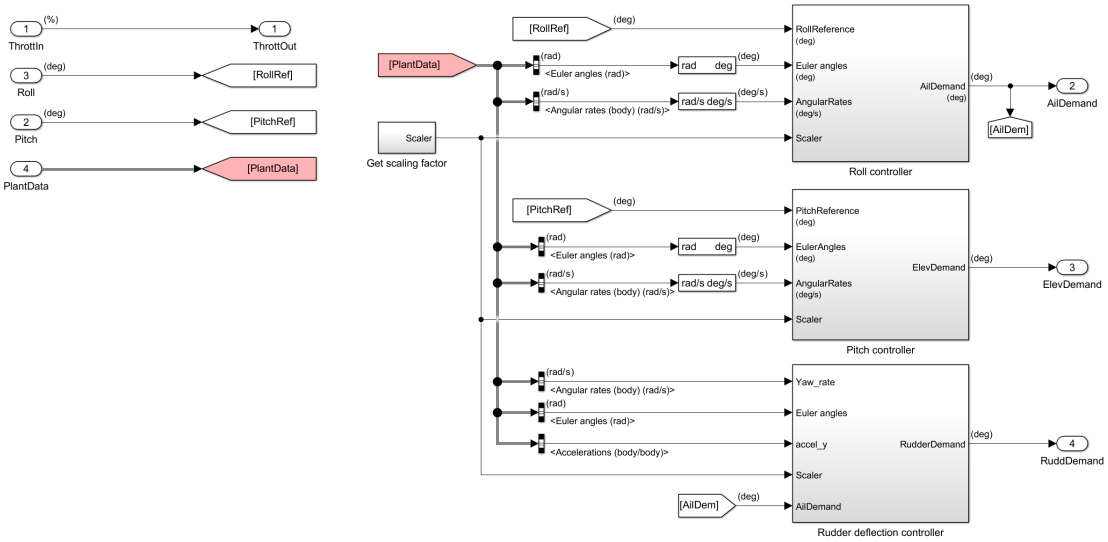


Figure 3.12: Roll, pitch and rudder deflection controllers implemented on the Simulink simulator

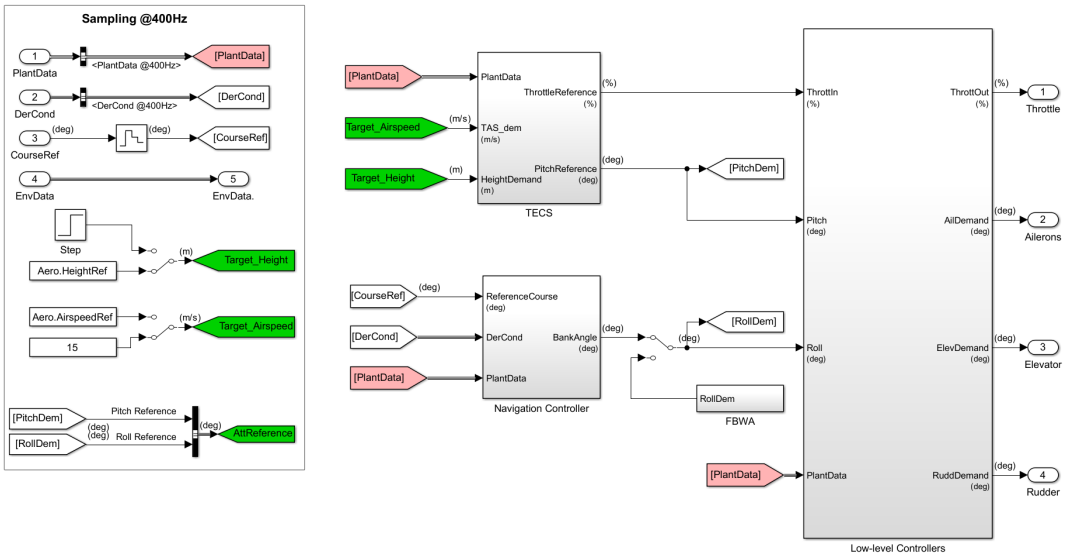


Figure 3.13: ArduPilot autopilot layer implemented on the Simulink simulator

3.5 Experimental results

After the software implementation, a series of experimental tests followed in order to verify the simulator performance. Before presenting the results, a description of the test is firstly presented.

3.5.1 Test description

The experimental tests were conducted in two different locations: some of them in the countryside of Delft (NL), and some others in a specific area in the province of Milan (IT) licensed to RC-models flight. Before dealing with the micro-controller, it was necessary to acquire some piloting skills for the RC-models. After that period, the aircraft was assembled and prepared for the tuning of the PID controllers with some pre-flight operations:

- Center of gravity check.
- Accelerometer calibration: it is necessary to have a correct attitude estimate. The plane is placed on a flat surface on all its sides as accurately as possible. The software records the accelerometers readings of the gravity acceleration and applies its calibration routine.
- Magnetometer calibration: it is necessary to rotate the aircraft different times of 360 degrees, once per each side of the aircraft. In this way the software can learn the magnetometer offsets to provide accurate yaw and course estimates.
- Airspeed sensor calibration: it is necessary to improve the airspeed readings. It consists in taking-off and flying a repeated circuit or circular loiter for 5 minutes. This was done in MANUAL mode.

3.5.2 PIDs tuning

The PID tuning can be operated in two different ways: one is to follow the autotune routine of the ArduPilot, the other is to tune manually the gains.

In practice, the tuning of the PIDs was conducted with both methods. The autotune is activated switching to a specific flight mode, called, indeed, AUTOTUNE.

The AUTOTUNE mode is a flight mode that flies in the same way as FBWA, but uses changes in flight attitude input by the pilot to learn the key values for roll and pitch tuning. So the pilot uses their transmitter mode switch to switch to AUTOTUNE mode and then flies the plane for a few minutes. While flying the pilot needs to input as many sharp attitude changes as possible so that the autotune code can learn how the aircraft responds. Autotune mode tunes the

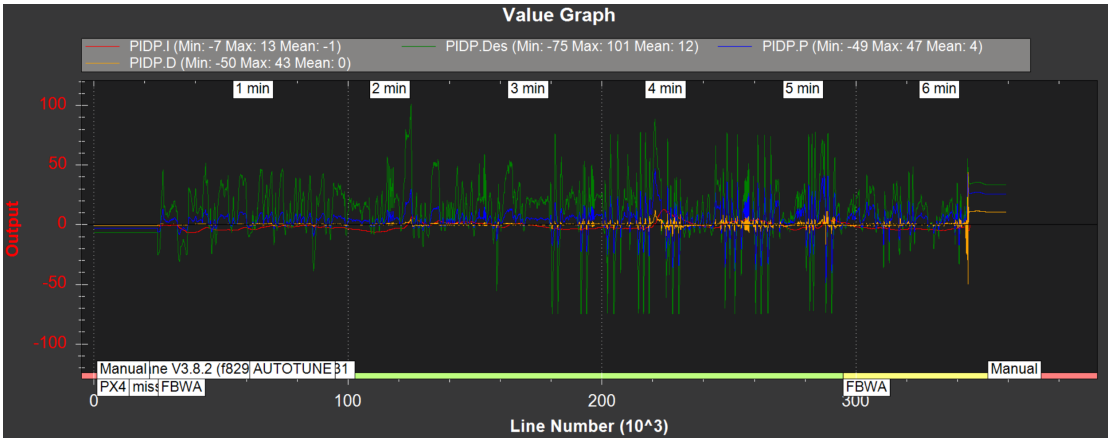


Figure 3.14: Example of the pitch PID autotune routine

P gain directly, but sets the D and I gain conservatively based on the selected auto-tune aggressiveness. Therefore it is chosen as starting point.

As an example, Figure 3.14 shows the pitch PID autotuning process: the blue line, indicating the achieved pitch rate, progressively tracks closer and closer the green line, *i.e.*, the demanded pitch rate by the pilot.

PID gains were successively refined manually until satisfactory performances were achieved. Final gains values are reported in Appendix D.

The side-slip controller was not tuned since it is not necessary to have the side-slip angle control.

After the roll and pitch tuning, the refinement of the values for the TECS PIDs was done. Fortunately the performances were initially good and small changes were made to the default values.

All the parameters were then reported in Simulink.

3.5.3 Closed-loop verification

Besides the tuning of the low-level controllers, another important step should be conducted, which is to validate the the dynamic model of the aircraft after any parameters estimation. Being conscious of the importance of an open-loop validation, this step was momentarily skipped for the following reasons:

- The guidance law to be developed lays onto the low-level controller layer, therefore it is sufficient that the overall system dynamics as seen from the upper guidance layer respects the real closed-loop dynamics of the aircraft.
- This process requires particularly good weather conditions. Winds can compromise the experiments and the eventual parameters identification, also because of some heavily coupled dynamics. As an example, a small cross wind after the open-loop pilot input can results on a higher roll displacement, which falsifies both the roll and yaw dynamics.

- In the fashion they are computed from Datcom, the parameters are too many. Before going on, in the future it is necessary to reduce the aerodynamic coefficients complexity, using, for example, a linear approximation as function of the angle of attack.

- There was not the sufficient time during the thesis course combined with the weather conditions.

This aspect will be doubtless deepened after the end of the project. Nevertheless, the closed-loop verification, offers very good results.

For 'closed-loop' we intend the use of FBWA mode, hence the use of roll and pitch controller. The pilot gives as inputs the roll and pitch angle references in place of ailerons and elevator deflections. Practically, the test was conduced by firstly putting manually the aircraft levelled; afterwards the FBWA mode is activated and several sharp inputs (full stick) of roll angles were given. Inputs and outputs were logged. With the same inputs and setting the simulator with the same initial conditions for the attitude and angular rates, a comparison can for the lateral dynamics (the one of interest) for the roll and yaw angles can be presented.

In Figures 3.15 and 3.16, one can see that the simulator and real dynamics are comparable. Therefore, the lateral model of the aircraft sufficiently approximates the real dynamics.

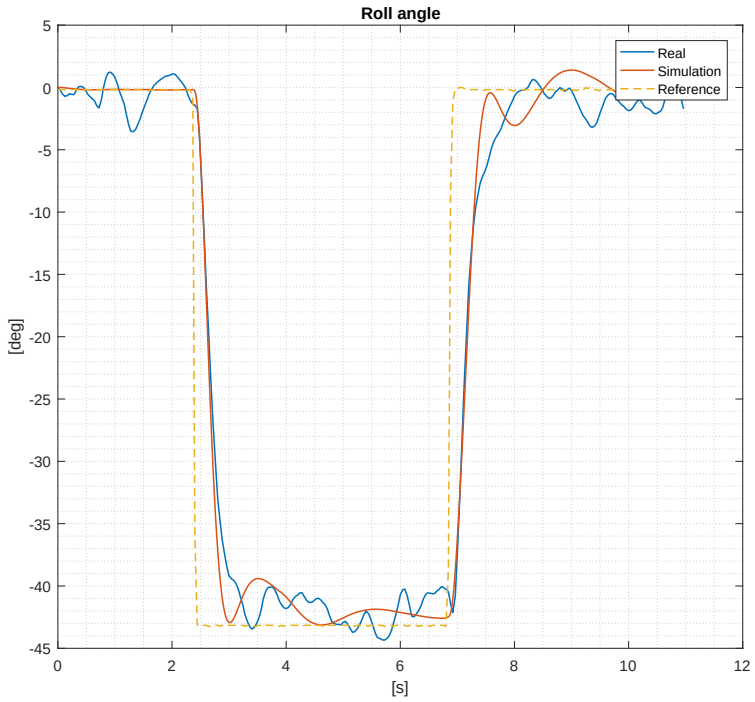


Figure 3.15: Roll dynamics comparison between the simulator and the test

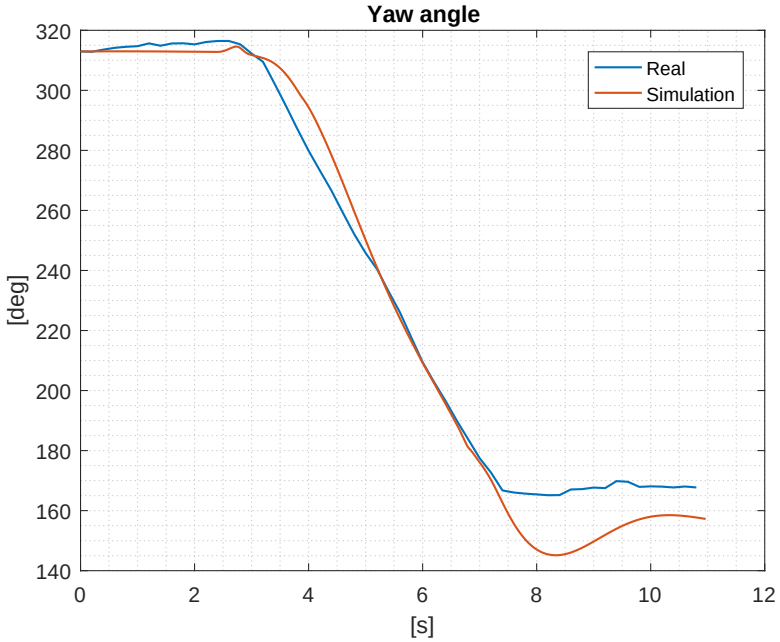


Figure 3.16: Yaw dynamics comparison between the simulator and the test

Chapter 4

Vector Field path following

The primary challenge in tracking straight-line segments or circular orbits is wind. The strategy of tracking a moving point can result in significant problems if disturbances are not properly accounted for, and, moreover, predicting these disturbances is not easy. For this reason, rather than using a trajectory tracking approach, a path tracking approach is generally preferred, where the goal is being on the path instead of being at a certain point at a particular time. The reference quantity in the path following strategies is called cross-track error, which is defined as the spacial deviation form the desired path.

Given a path, the initial location of the aerial vehicle, and its course angle χ, the path-following problem is to determine the commanded course angle that accurately tracks a path. The paths considered here only develop on the horizontal plane (formally the plane $(\mathbf{i}^i, \mathbf{j}^i)$), thus implying that the UAV is flying at a constant altitude.

In the following, an introduction to the Vector Field path following approach is firstly given in Section 4.1, and then a reduced order model for the guidance purpose is shown in Section 4.2. It will be seen that the proposed method works under a particular assumption for the course-hold loop. Therefore, the course loop is appropriately synthesized in Section 4.3. Afterwards, the Vector Field is described and analysed for the straight-line and orbit path primitives, and, among all its different variants, the method presented in [21] will be especially accounted for and implemented in Sections 4.4 and 4.5. Lastly, following [1], the adaptive modification to the guidance law is introduced to improve its wind disturbance rejection.

4.1 Introduction

In a survey and analysis of the most common fixed-wing UAV path following techniques [22], many of these are applied to a point mass UAV kinematic model. Some of them, such as the 'carrot-chasing' algorithm, uses a VTP (Virtual Target

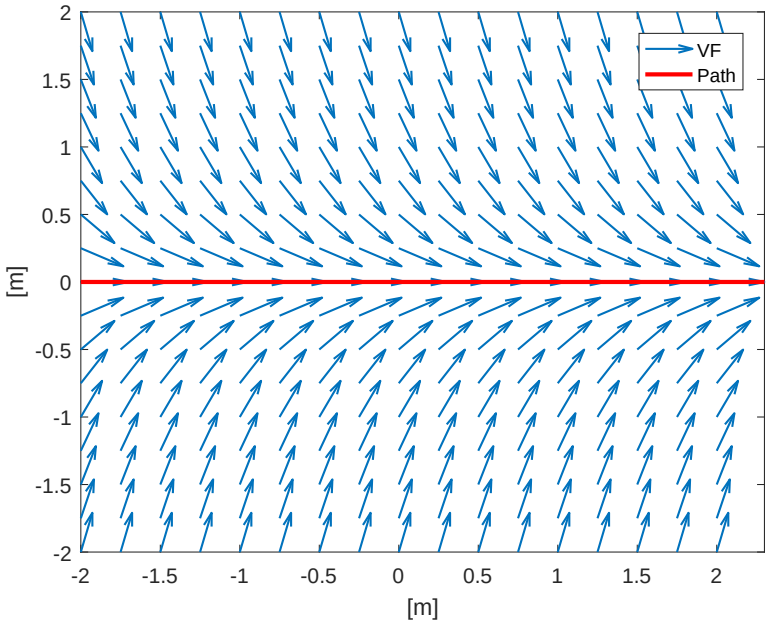


Figure 4.1: Vector field straight line path following example

Point) on the path and direct the UAV to chase the VTP. Other analysed techniques are the Vector Field method, the pure pursuit with line of sight method, and a LQR geometrical control. From those results, it has been shown that Vector Field path following approach achieves the lowest steady-state cross-track error and requires the least control effort with respect to the other approaches. Conversely, it requires the tuning of different key parameters.

The goal of the Vector Field approach is clearly to drive the cross-track error to zero asymptotically using the course angle χ as the control variable. For this reason, wherever the UAV is with respect to the required path, it is necessary that the the commanded angle χ_c results in the UAV moving towards the path. The angle χ is the most convenient control variable for this objective, since it is inertial-referenced. Considering each point around the desired path, the set of desired course angles is called vector field because it constitutes a set of vectors (relative to the path) of course unit vectors. It is very similar to the artificial potential fields widely used in the robotics community for obstacle avoidance, with the difference the the Vector Field method does not necessarily represent the gradient of a potential; rather, the vector field simply indicates a desired direction of travel.

The fundamental paths are the line and the orbit; combining these two primitives it is possible to build up more involved paths.

4.2 Dynamic model for guidance

In Chapter 1 the full equations of motion show a high mathematical complexity. The development of the guidance laws is usually done on a simplified dynamic model, that anyhow still captures the essential behaviour of the system. The main simplification is to eliminate the forces and moments balance equations, thus eliminating the need to calculate aerodynamic forces and moments acting on the airframe. Therefore, for the guidance purpose, the UAV is assumed to be flying:

- at a constant altitude h;
- at a constant airspeed V_a.

In this way, the simplified UAV model is formulated to include only its kinematics, and its positional equations are:

$$\begin{cases} \dot{p}_n = V_a \cos\psi + w_n \\ \dot{p}_e = V_a \sin\psi + w_e \end{cases} . \tag{4.1}$$

The heading angle will be controlled by the vector-field path-following approach. However, it is more convenient to avoid the dependency from the wind and express the kinematics equations in terms of ground speed and course angle instead of airspeed and heading angle. Recalling the wind triangle of Figure 1.5, one has:

$$\begin{cases} \dot{p}_n = V_g \cos\chi \\ \dot{p}_e = V_g \sin\chi \end{cases} . \tag{4.2}$$

Finally the course angle dynamics has to be defined. The angle χ is hereafter assumed to be measurable (this holds true with our configuration). The guidance law stability proof in [21] and [1], is simplified a lot if the course-hold dynamics is assumed to be a first order written as:

$$\dot{\chi} = \alpha_\chi(\chi_c - \chi), \tag{4.3}$$

where α_χ determines dynamics of the system. To make this assumption valid it is therefore necessary to design a course hold loop controller which is able to let the overall UAV course dynamics being as close as possible to a first order system.

4.3 Course-hold loop design

For the course angle control a PID controller is chosen. To tune the PID for the outer course loop, one should linearize the model about a trim point.

Figure 4.2: Simulink model ready to be trimmed

4.3.1 Trim and linearization

The trim point is computed with the Simulink *Linear analysis* toolbox in the condition of straight and level flight. The model is set up in such a way that the inputs and outputs are respectively:

$$u = \begin{bmatrix} \delta_a \\ \delta_e \\ \delta_r \\ \delta_t \end{bmatrix} \qquad y = \begin{bmatrix} V_a \\ \alpha \\ \beta \end{bmatrix} \tag{4.4}$$

as shown in Figure 4.2. The purpose in specifying V_a, α and β as outputs is that the will is to let the Simulink 'trim' command maintain $V_a = \bar{V}_a$ and $\beta = \bar{\beta}$. The angle of attack is not imposed, but is a quantity of interest to visualize. In Figure 4.3, one can see the Simulink screen with the states specifications for trimming, including the states given by the actuators dynamics too. The trim airspeed is set at $\bar{V}_a = 15$ m/s, while the sideslip angle is forced to $\bar{\beta} = 0$.

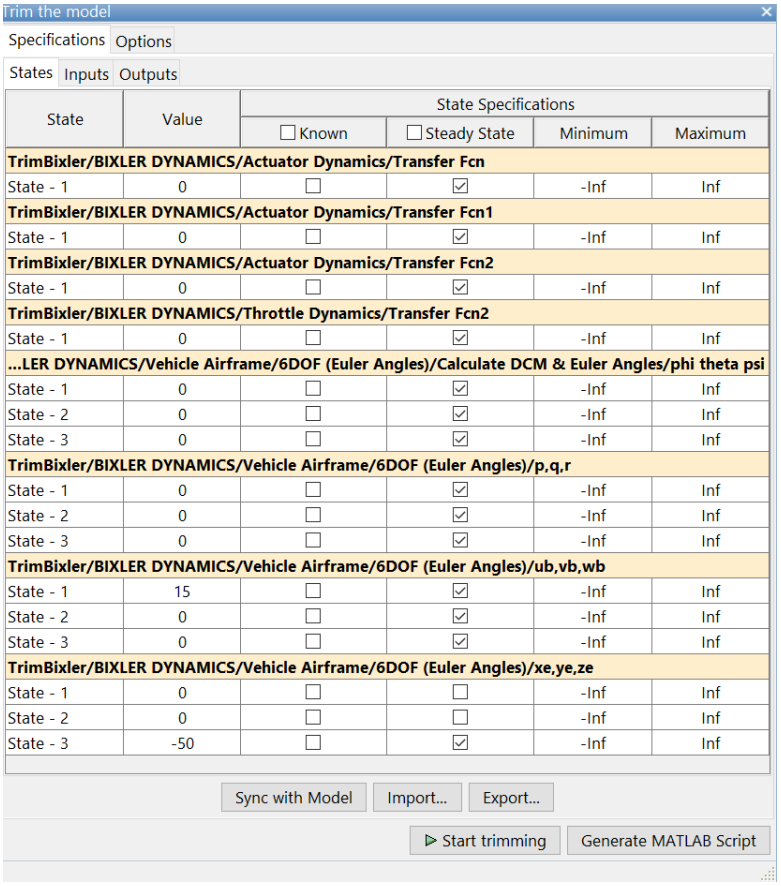
The equilibrium is reached when the input $\bar{u}$ is:

$$\bar{u} = \begin{bmatrix} 0 \\ -1.25 \\ 0 \\ 33.9 \end{bmatrix} \tag{4.5}$$

where the elevator deflection δ_e is reported in deg.

After stable zero-poles cancellations, the transfer function of linearized model about this trim point is:

$$\frac{\phi(s)}{\delta_a(s)} = P_\phi^*(s) = \frac{3080.7}{s(s+45)(s+9.344)}. \tag{4.6}$$



State	Value	State Specifications			
		Known	Steady State	Minimum	Maximum
TrimBixler/BIXLER DYNAMICS/Actuator Dynamics/Transfer Fcn					
State - 1	0	☐	☑	-Inf	Inf
TrimBixler/BIXLER DYNAMICS/Actuator Dynamics/Transfer Fcn1					
State - 1	0	☐	☑	-Inf	Inf
TrimBixler/BIXLER DYNAMICS/Actuator Dynamics/Transfer Fcn2					
State - 1	0	☐	☑	-Inf	Inf
TrimBixler/BIXLER DYNAMICS/Throttle Dynamics/Transfer Fcn2					
State - 1	0	☐	☑	-Inf	Inf
...LER DYNAMICS/Vehicle Airframe/6DOF (Euler Angles)/Calculate DCM & Euler Angles/phi theta psi					
State - 1	0	☐	☑	-Inf	Inf
State - 2	0	☐	☑	-Inf	Inf
State - 3	0	☐	☑	-Inf	Inf
TrimBixler/BIXLER DYNAMICS/Vehicle Airframe/6DOF (Euler Angles)/p,q,r					
State - 1	0	☐	☑	-Inf	Inf
State - 2	0	☐	☑	-Inf	Inf
State - 3	0	☐	☑	-Inf	Inf
TrimBixler/BIXLER DYNAMICS/Vehicle Airframe/6DOF (Euler Angles)/ub,vb,wb					
State - 1	15	☐	☑	-Inf	Inf
State - 2	0	☐	☑	-Inf	Inf
State - 3	0	☐	☑	-Inf	Inf
TrimBixler/BIXLER DYNAMICS/Vehicle Airframe/6DOF (Euler Angles)/xe,ye,ze					
State - 1	0	☐	☐	-Inf	Inf
State - 2	0	☐	☐	-Inf	Inf
State - 3	-50	☐	☑	-Inf	Inf

Figure 4.3: Simulink trim screen

The transfer function is in the same form of $P_\phi(s)$; Simulink linearization procedure also consider every cross-coupling coming from all the model dynamics. The pole given by $(s + 45)$ comes from the actuator dynamics.

4.3.2 Controller synthesis

Using the roll loop structure discussed in Section 3.2.2, the closed-loop transfer function from the commanded roll angle to the achieved roll angle is described from:

$$\frac{\phi(s)}{\phi_c(s)} = \frac{2017.8}{(s+45)(s^2+8.467s+44.88)}. \tag{4.7}$$

Its Bode diagram in Figure 4.4 shows that its critical frequency is 5 rad/s.

From equation (3.7), neglecting the disturbances, we saw that the transfer function from ϕ to the course angle χ is just $\frac{g/V_g}{s}$. Therefore the system to be

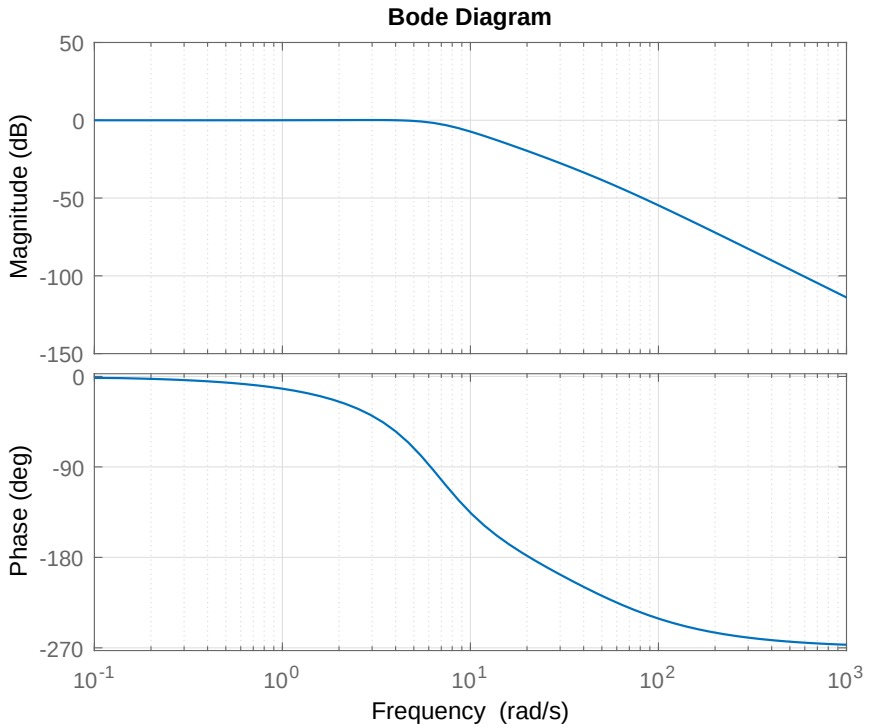


Figure 4.4: Bode diagram of $\phi(s)/\phi_c(s)$

controlled is:

$$P_\chi^*(s) = \frac{\chi(s)}{\phi_c(s)} = \frac{1319.6}{s(s+45.46)(s^2+8.467s+44.88)} \tag{4.8}$$

The goal is to make the closed-loop transfer function $F_\chi(s) = \chi(s)/\chi_c(s)$ as close as possible to a first order system. To this end, it is sufficient to use a P controller, denoted as $C_\chi(s)$; tuning the proportional gain one modifies the crossover frequency of the closed-loop system. A lower value corresponds to a slower response, but a larger bandwidth where equation(4.3) holds true.

In Figure 4.5, there are some Bode diagrams of the closed-loop transfer function $F_\chi(s)$, using different proportional gains. The final choice was $K_{P_\chi} = 0.7$: this choice assures that the course dynamics behaves as a first order system in the frequency range $[0, 6]$ rad/s, and also guarantees a sufficient frequency separation between the inner and outer loop since its crossover frequency is about 0.1 rad/s, more than one decade before the crossover frequency of the inner roll control loop.

For this reason, the closed-loop transfer function can be written as:

$$F_\chi(s) = \frac{K_{P_\chi} g/V_g}{s + K_{P_\chi} g/V_g} \tag{4.9}$$

and the wanted α_χ value is:

$$\alpha_\chi = \frac{K_{P_\chi} g}{V_g} = 0.4578. \tag{4.10}$$

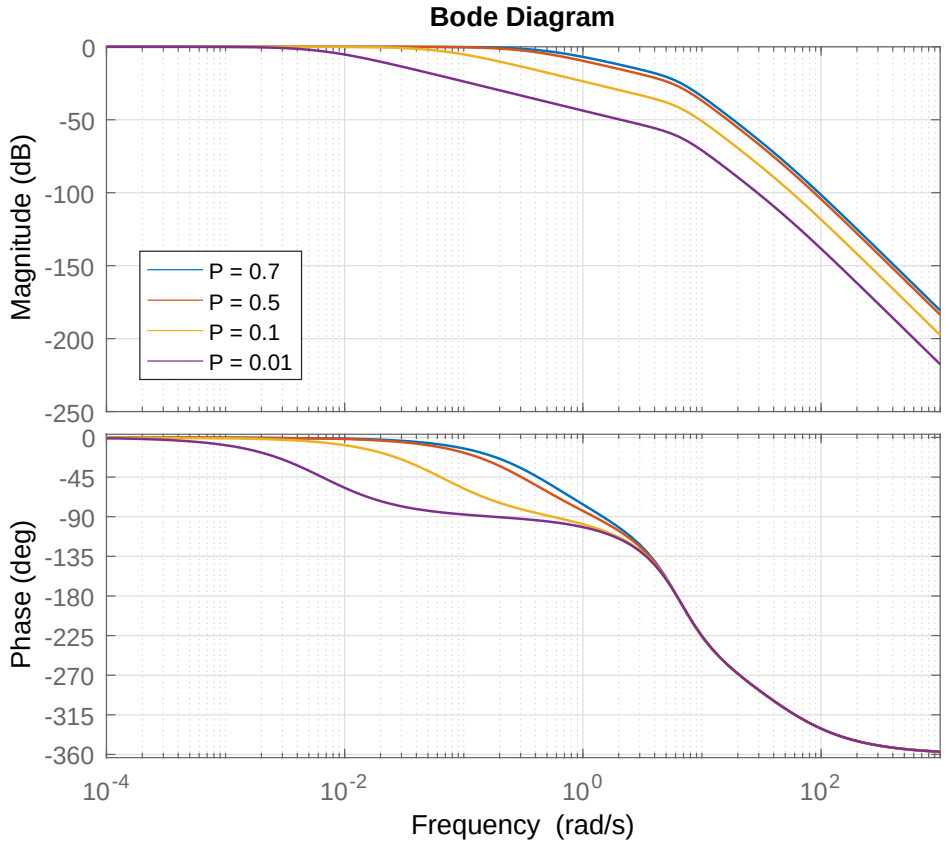


Figure 4.5: Bode diagrams of different closed-loop transfer functions $F_\chi(s)$ for different P and D values of K_{P_χ}

4.4 Straight-line path following

In the following, the subscript '*sl*' will be used to indicate variables referred to the straight-line case, while '*o*' is used with respect to the orbit case.

4.4.1 Definition

A straight-line path is described by two vectors $\mathbf{r} \in \mathbb{R}^2$ and $\mathbf{q} \in \mathbb{R}^2$ as

$$\mathcal{P}_{\text{line}}(\mathbf{r}, \mathbf{q}) = \left\{ \mathbf{x} \in \mathbb{R}^2 : \mathbf{x} = \mathbf{r} + \lambda \mathbf{q}, \lambda \in \mathbb{R} \right\} \tag{4.11}$$

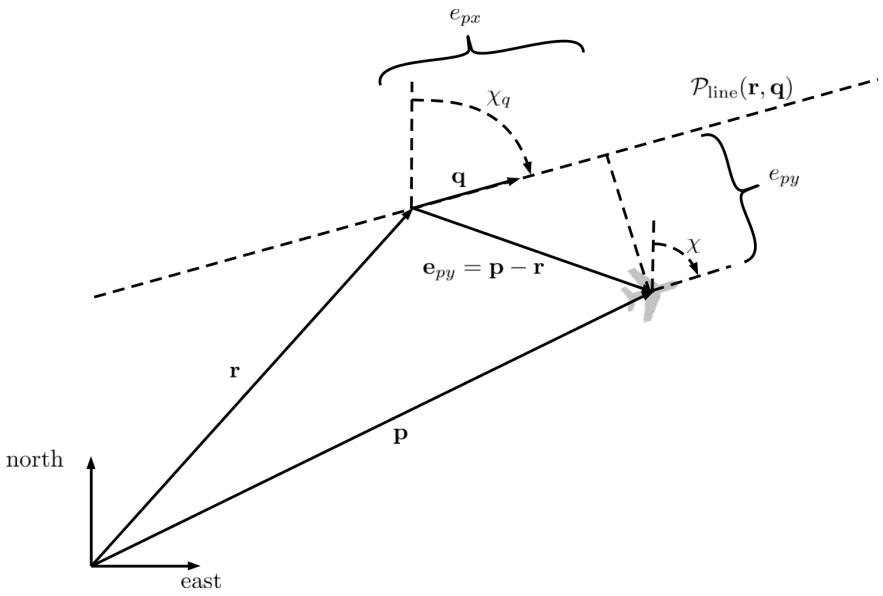


Figure 4.6: Representation of $\mathcal{P}_{\text{line}}$ and the quantities of interest for the UAV straight-line path following

where $\mathbf{r}$ is the origin of the path, and $\mathbf{q} = [q_n, q_e]$ is the unit direction vector describing the line direction in north and east components. The course angle of the line is given by:

$$\chi_q = \text{atan2}\left(\frac{q_e}{q_n}\right). \tag{4.12}$$

Since the line can have whatever orientation, it is worth to introduce a new coordinates frame, hereby called path frame. An observer on the path frame sees the reference line always pointing to the north. The path frame has its center in $\mathbf{r}$; the rotation matrix from the inertial frame to the path frame has the usual fashion:

$$\mathcal{R}_i^p = \begin{bmatrix} \cos\chi_q & \sin\chi_q \\ -\sin\chi_q & \cos\chi_q \end{bmatrix}. \tag{4.13}$$

Define the cross-track error as the vector $\mathbf{e}_p = \mathbf{p} - \mathbf{r}$, where $\mathbf{p} = [p_n, p_e]^T$. For the above considerations it holds true that:

$$\mathbf{e}_p = \begin{bmatrix} e_{px} \\ e_{py} \end{bmatrix} = \mathcal{R}_i^p(\chi_q)(\mathbf{p}^i - \mathbf{r}^i). \tag{4.14}$$

The quantity of interest is e_{py} which represents the lateral deviation from the path, *i.e.*, the lateral cross-track error.

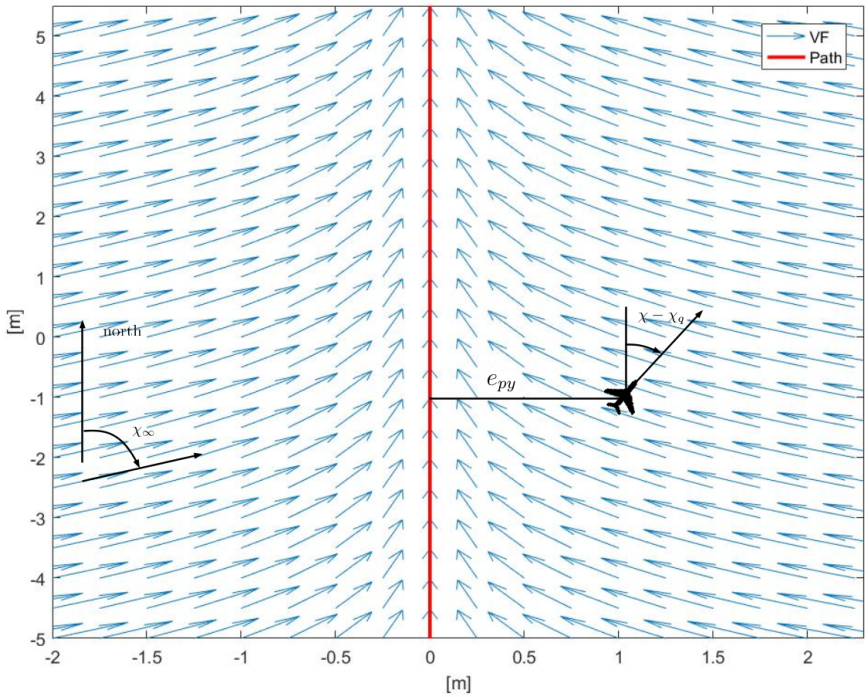


Figure 4.7: Vector field for straight-line path following

4.4.2 Strategy

The strategy is to construct a vector field such that, when e_{py} is large, lets the UAV course angle be $\chi_\infty \in \left(0, \frac{\pi}{2}\right]$, and when e_{py} approaches zero, ensures that the desired course also approaches zero. To this end, considering for the moment $\chi_q = 0$, a good candidate function for the desired course is:

$$\chi_d(e_{py}) = -\chi_\infty \frac{2}{\pi} \tan^{-1}(k_{sl} e_{py}) \tag{4.15}$$

where k_{sl} is a positive constant which regulates the rate of transition from χ_∞ to zero. Large values of k_{sl} result in short, abrupt transitions, while small values of k_{sl} cause long, smooth transitions in the desired course. Being $\chi_\infty \in \left(0, \frac{\pi}{2}\right]$, implies that $\chi_d \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$.

If $\chi_q \neq 0$, following Figure 4.7, it is necessary to reformulate $\chi_d(e_{ey})$ as:

$$\chi_d(e_{py}) = \chi_q - \chi_\infty \frac{2}{\pi} \tan^{-1}(k_{sl} e_{py}). \tag{4.16}$$

To prove that $e_{py} \to 0$ if $\chi \to \chi_d$, we can use the following positive-definite Lyapunov function:

$$\mathcal{V}_1 = \frac{1}{2} e_{py}^2. \tag{4.17}$$

From equation (4.14), it is known that

$$e_{py} = -\sin\chi_q(p_n - r_n) + \cos\chi_q(p_e - r_e), \tag{4.18}$$

and, considering the dynamical model introduced in equations (4.2), e_{py} derivative gives:

$$\begin{aligned}\dot{e}_{py} &= (-\sin\chi_q)V_g\cos\chi + (\cos\chi_q)V_g\sin\chi &(4.19)\\ &= V_g\sin(\chi - \chi_q). &(4.20)\end{aligned}$$

So, if $\chi \to \chi_d$, the derivative of $\mathcal{V}_1$ is given by:

$$\begin{aligned}\dot{\mathcal{V}}_1 &= e_{py}\dot{e}_{py} &(4.21)\\ &= e_{py}V_g\left(\sin(\chi_d - \chi_q)\right) &(4.22)\\ &= e_{py}V_g\left[\sin\left(-\chi_\infty\frac{2}{\pi}\tan^{-1}(k_{sl}e_{py})\right)\right] &(4.23)\end{aligned}$$

which is negative-definite for $e_{py} \neq 0$.
To have $\chi \to \chi_d$, define now a new positive-definite Lyapunov function as:

$$\mathcal{V}_2 = \frac{1}{2}\tilde{\chi}^2 \tag{4.24}$$

with $\tilde{\chi} = \chi - \chi_d$. Proceeding as before:

$$\begin{aligned}\tilde{\chi} &= \alpha_\chi(\chi_c - \chi) + \chi_\infty\frac{2}{\pi}\frac{k_{sl}\dot{e}_{py}}{1 + (k_{sl}e_{py})^2} &(4.25)\\ &= \alpha_\chi(\chi_c - \chi) + \chi_\infty\frac{2}{\pi}\frac{k_{sl}}{1 + (k_{sl}e_{py})^2}V_g\sin(\chi - \chi_q). &(4.26)\end{aligned}$$

So, the derivative of $\mathcal{V}_2$ yields

$$\begin{aligned}\dot{\mathcal{V}}_2 &= \tilde{\chi}\dot{\tilde{\chi}} &(4.27)\\ &= \tilde{\chi}\left(\alpha_\chi(\chi_c - \chi) + \chi_\infty\frac{2}{\pi}\frac{k_{sl}}{1 + (k_{sl}e_{py})^2}V_g\sin(\chi - \chi_q)\right). &(4.28)\end{aligned}$$

Finally, from the expression (4.28), it is possible to choose the controller function in order to make $\dot{\mathcal{V}}_2$ negative-definite, which is:

$$\chi_c = \chi - \frac{1}{\alpha_\chi}\chi_\infty\frac{2}{\pi}\frac{k_{sl}}{1 + (k_{sl}e_{py})^2}V_g\sin(\chi - \chi_q) - \xi_{sl}. \tag{4.29}$$

Selecting ξ as

$$\xi = \frac{\kappa_{sl}}{\alpha_\chi}\text{sign}(\tilde{\chi}) \tag{4.30}$$

with $\kappa_{sl} > 0$, it is possible to get

$$\dot{\mathcal{V}}_2 \leq -\kappa_{sl}|\tilde{\chi}|. \tag{4.31}$$

It is well known that the 'sign' function leads to chattering of the control action, therefore, to alleviate the problem, it is replaced by

$$\text{sat}(x) = \begin{cases} x, & \text{if } |x| < 1, \\ \text{sign}(x), & \text{otherwise.} \end{cases} \tag{4.32}$$

To control the boundary region of the 'sat' function, a term ε_{sl} is introduced as denominator of $\tilde{\chi}$ in the expression of ξ_{sl}. At the end, the controller expression is given by:

$$\chi_c = \chi - \frac{1}{\alpha_\chi}\chi_\infty \frac{2}{\pi}\frac{k}{1+(k_{sl}e_{py})^2}V_g \sin(\chi - \chi_q) - \frac{\kappa_{sl}}{\alpha_\chi}\text{sign}\left(\frac{\tilde{\chi}}{\varepsilon_{sl}}\right). \tag{4.33}$$

The presented control strategy is called in literature sliding mode control [23], where the properties of the sliding surface are moderated by the parameters χ_∞, k_{sl}, κ_{sl}, and ε_{sl} [21].

4.5 Orbit path following

4.5.1 Definition

An orbit with center $\mathbf{c} \in \mathbb{R}^2$, radius $\varrho \in \mathbb{R}$ and direction $\lambda \in \{-1, +1\}$, is defined as:

$$\mathcal{P}_{or}(\mathbf{c}, \varrho, \lambda) = \left\{\mathbf{r} \in \mathbb{R}^2 : \mathbf{r} = \mathbf{c} + \lambda\varrho[\cos\gamma, \sin\gamma]^T, \gamma \in [0, 2\pi)\right\}. \tag{4.34}$$

When $\lambda = 1$ the orbit path is in the clockwise direction, otherwise, when $\lambda = -1$ it is in the counter-clockwise direction (Figure 4.8).

For orbit path following it is better to express the UAV dynamics in polar coordinates. Therefore, given that $\mathbf{c} = [c_n, c_e]^T$, the new variable d is defined as the distance of the UAV from the orbit center, hence $d = ||\mathbf{p} - \mathbf{c}||_2$, and the phase angle as

$$\gamma = \text{atan2}\left(\frac{p_e - c_e}{p_n - c_n}\right). \tag{4.35}$$

Moreover, the dynamics equations (4.2) have to be transformed into the polar coordinates as well. For this reason, the dynamics in the normal and tangential directions to the orbit are

$$\begin{bmatrix} \dot{d} \\ d\dot{\gamma} \end{bmatrix} = \begin{bmatrix} \cos\gamma & \sin\gamma \\ -\sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} \dot{p}_n \\ \dot{p}_e \end{bmatrix} \tag{4.36}$$

$$= \begin{bmatrix} \cos\gamma & \sin\gamma \\ -\sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} V_g\cos\chi \\ V_g\sin\chi \end{bmatrix} \tag{4.37}$$

$$= \begin{bmatrix} V_g\cos(\chi-\gamma) \\ V_g\sin(\chi-\gamma) \end{bmatrix}. \tag{4.38}$$

Rearranging the terms,

$$\begin{cases} \dot{d} = V_g\cos(\chi-\gamma) \\ \dot{\gamma} = \frac{V_g}{d}\sin(\chi-\gamma) \end{cases}. \tag{4.39}$$

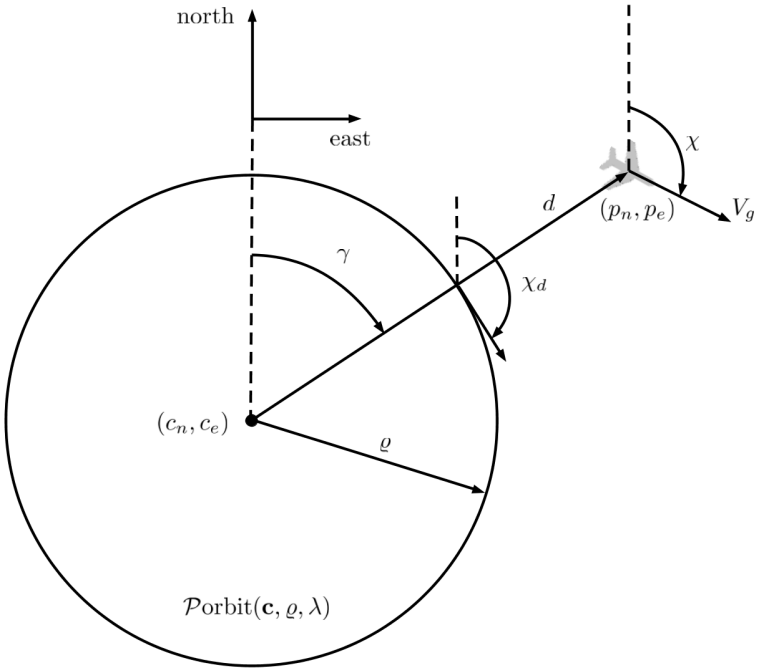


Figure 4.8: Representation of P_{orbit} and the quantities of interest for the orbit path following

4.5.2 Strategy

The strategy carried out to determine a controller expression for the path following of an orbit, follows almost identically what already introduced for the straight-line.

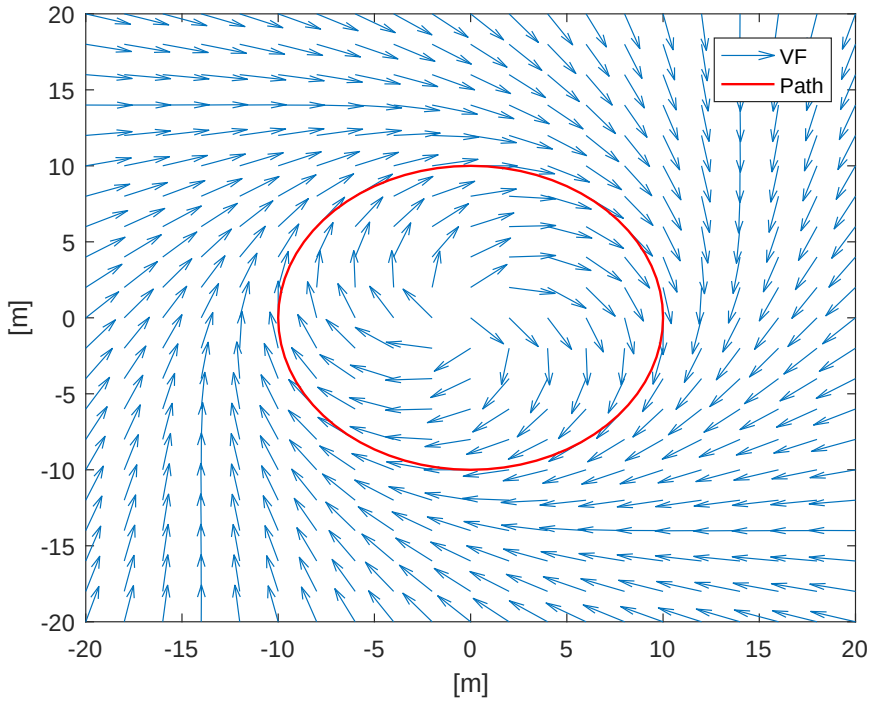


Figure 4.9: Vector field for orbit path following

This time, let us define the path error as

$$\tilde{d} = d - \varrho. \tag{4.40}$$

It is convenient that when $\tilde{d}$ is large, the desired course χ_d makes the UAV directed toward the orbit center, while, when $\tilde{d}$ is approaching zero, χ_d reduces to $\gamma + \lambda\frac{\pi}{2}$, *i.e.*, the tangential direction to the orbit perimeter according to the orbit direction (Figure 4.9).

Therefore, it is possible to define the desired course as:

$$\chi_d(\tilde{d}) = \gamma + \lambda\frac{\pi}{2} + \lambda\tan^{-1}(k_o\tilde{d}) \tag{4.41}$$

$$= \gamma + \lambda\left(\frac{\pi}{2} + \tan^{-1}(k_o\tilde{d})\right) \tag{4.42}$$

where k_o regulates the rate of transition to the orbit path.

To prove that $\tilde{d} \to 0$ asymptotically if $\chi \to \chi_d(\tilde{d})$, the considered Lyapunov function is $\mathcal{V}_1 = \frac{1}{2}\tilde{d}^2$, and its derivative is:

$$\dot{\mathcal{V}}_1 = V_g\tilde{d}\cos\left(\lambda\frac{\pi}{2} + \lambda\tan^{-1}(k_o\tilde{d})\right) \tag{4.43}$$

$$= -\lambda V_g\tilde{d}\sin(\tan^{-1}(k_o\tilde{d})) \tag{4.44}$$

which is less than zero for $\tilde{d} \neq 0$ since the argument of $\sin(\cdot)$ is in $(-\frac{\pi}{2}, +\frac{\pi}{2})$ for every $\tilde{d}$, implying $\tilde{d} \to 0$ asymptotically.

As before, define $\tilde{\chi} = \chi - \chi_d(\tilde{d})$ and differentiate to obtain

$$\dot{\tilde{\chi}} = \alpha_\chi(\chi_c - \chi) - \dot{\chi}_d(\tilde{d}) \tag{4.45}$$

$$= \alpha_\chi(\chi_c - \chi) - \frac{V_g}{d}\sin(\chi - \gamma) + \frac{k_o\lambda}{1 + (k_o\tilde{d})^2}V_g\cos(\chi - \gamma). \tag{4.46}$$

For brevity, call $\beta_o = k_o/\left(1 + (k_o\tilde{d})^2\right)$; the derivative of $\mathcal{V}_2$ is therefore:

$$\dot{\mathcal{V}}_2 = \tilde{\chi}\left(\alpha_\chi(\chi_c - \chi) - \frac{V_g}{d}\sin(\chi - \gamma) - \beta_o\lambda V_g\cos(\chi - \gamma)\right). \tag{4.47}$$

Choosing the control action as

$$\chi_c = \chi + \frac{V_g}{\alpha_\chi d}\sin(\chi - \gamma) + \frac{\beta_o\lambda}{\alpha_\chi}V_g\cos(\chi - \gamma) - \xi_o \tag{4.48}$$

brings $\dot{\mathcal{V}}_2$ to be negative-definite.

Equivalently to the straight-line case, ξ_o is

$$\xi_o = \frac{\kappa_o}{\alpha_\chi}\text{sat}\left(\frac{\tilde{\chi}}{\varepsilon_o}\right). \tag{4.49}$$

4.6 Adaptive modification of the control law

The performance of these two controllers is analysed in [21]. In general, this theory is founded on the assumption that the wind vector is known and constant over time. With this assumption, the optimality of the control laws is demonstrated in this research, where using χ_c is able to drive the lateral cross-track error to zero in both conditions. It is clear that the hypothesis of a constant wind is unrealistic.

In [1], the known-constant wind hypothesis is relaxed: a new wind vector is defined and constituted by a constant term and a time-varying term. The situation is better depicted in Figure 4.10, with a modification of the wind triangle previously presented, and where the wind vector is formally written as

$$\mathbf{V}_w(t) = \mathbf{W} + \mathbf{A}(t) \tag{4.50}$$

This slightly modifies the dynamics of equations (4.1) in the new

$$\begin{cases} \dot{p}_n = V_a\cos\psi + W\cos\psi_w + A(t)\cos\psi_a(t) \\ \dot{p}_e = V_a\sin\psi + W\sin\psi_w + A(t)\sin\psi_a(t) \end{cases} \tag{4.51}$$

where W and $A(t)$ are the amplitudes of the constant and time-varying parts of the wind vector, while ψ_w and ψ_a are the angles between the constant and

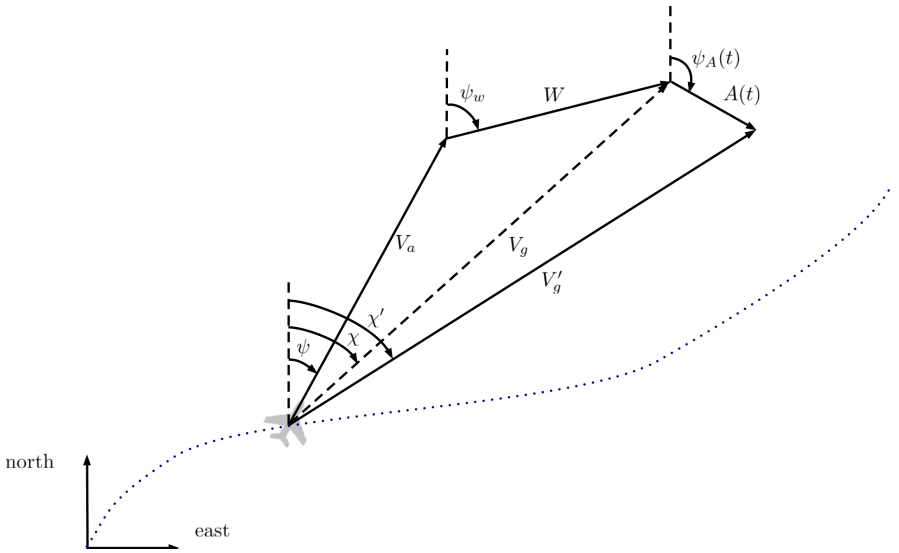


Figure 4.10: Wind triangle with the wind vector divided into one constant component and one time-varying component

time-varying vectors with respect to the north axis. In this case, it is more likely that the magnitude and orientation of the constant wind vector can be obtained as an estimate from local wind averaged measurements, while $A(t)$ and $\psi_a(t)$ are treated as slowly time-varying disturbances.

The new overall wind vector is therefore:

$$\begin{cases} W' \cos \psi'_w = W \cos \psi_w + A(t) \cos \psi_a(t) \\ W' \sin \psi'_w = W \sin \psi_w + A(t) \sin \psi_a(t) \end{cases} . \tag{4.52}$$

Also the inertial-referenced dynamics, of course, undergoes some small changes:

$$\begin{cases} \dot{p}_n = V'_g \cos \chi' \\ \dot{p}_e = V'_g \sin \chi'. \end{cases} \tag{4.53}$$

The control laws (4.33) and (4.48) depend on V_g and, with this modification, become:

$$\chi_c = \chi' - \frac{\chi_\infty}{\alpha_\chi} \frac{2}{\pi} \frac{k_{sl}}{1 + (k_{sl} e_{py})^2} V'_g \sin(\chi' - \chi_q) - \frac{\kappa_{sl}}{\alpha_\chi} \text{sign}\left(\frac{\tilde{\chi}}{\varepsilon_{sl}}\right) \tag{4.54}$$

$$\chi_c = \chi' + \frac{V'_g}{\alpha_\chi d} \sin(\chi' - \gamma) + \beta_o \frac{\lambda}{\alpha_\chi} V'_g \cos(\chi' - \gamma) - \frac{\kappa_o}{\alpha_\chi} \text{sat}\left(\frac{\tilde{\chi}}{\varepsilon_o}\right). \tag{4.55}$$

To account for disturbances, it is then necessary to know V'_g, that can be considered as a time-varying system parameter to be estimated. In [1], two nonlinear estimators for V'_g are presented, one for the straight-line case, and another

for the orbit case. The proof of the controllers capability to let the path tracking error and the course error $\tilde{\chi}$ go to zero asymptotically in both cases, is given in [1] and will not be discussed further. The V_g' estimate is denoted as $\hat{V}_g'$ and the controllers equations are rewritten in the form:

$$\chi_c = \chi' - \frac{\chi_\infty}{\alpha_\chi}\frac{2}{\pi}\frac{k_{sl}}{1+(k_{sl}e_{py})^2}\hat{V}_g'\sin(\chi'-\chi_q) - \frac{\kappa_{sl}}{\alpha_\chi}\text{sign}\left(\frac{\tilde{\chi}}{\varepsilon_{sl}}\right) \tag{4.56}$$

$$\chi_c = \chi' + \hat{V}_g'\left(\frac{1}{\alpha_\chi d}\sin(\chi'-\gamma) + \beta_o\frac{\lambda}{\alpha_\chi}\cos(\chi'-\gamma)\right) - \frac{\kappa_o}{\alpha_\chi}\text{sat}\left(\frac{\tilde{\chi}}{\varepsilon_o}\right). \tag{4.57}$$

4.6.1 Straight-line

Let $\Theta = \hat{V}_g' - V_g'$ be the estimation error, and consider a new Lyapunov function as

$$\mathcal{V} = \mathcal{V}_1 + \mu_{sl}\mathcal{V}_2 + \frac{1}{2}\Gamma_{sl}^{-1}\Theta^2, \tag{4.58}$$

where μ_{sl} is a positive scaling term to let $\mathcal{V}_1$ and $\mathcal{V}_2$ be of the same order of magnitude, and $\Gamma > 0$ the estimation gain. Its derivative is

$$\dot{\mathcal{V}} = \dot{\mathcal{V}}_1 + \mu_{sl}\dot{\mathcal{V}}_2 + \Gamma_{sl}^{-1}\Theta\dot{\Theta} \tag{4.59}$$

and, after some computations and neglecting $\dot{V}_g'$ since V_g' is slowly varying in time, one has

$$\dot{\mathcal{V}} = \dot{\mathcal{V}}_1 - \mu_{sl}\kappa_{sl}\tilde{\chi}'\text{sat}\left(\frac{\tilde{\chi}'}{\varepsilon_{sl}}\right) + \left[\dot{\hat{V}}_g'\Gamma_{sl}^{-1} - \mu_{sl}\tilde{\chi}'\chi_\infty\frac{2}{\pi}\frac{k_{sl}}{1+(k_{sl}e_{py})^2}\sin(\chi'-\chi_q)\right]\left(\hat{V}_g' - V_g'\right). \tag{4.60}$$

As seen before, the first two terms are negative. Finally, if

$$\dot{\hat{V}}_g' = \Gamma_{sl}\mu_{sl}\tilde{\chi}'\chi_\infty\frac{2}{\pi}\frac{k_{sl}}{1+(k_{sl}e_{py})^2}\sin(\chi'-\chi_q) \tag{4.61}$$

then $\dot{\mathcal{V}}$ will be negative semi-definite.

Two modifications to equation (4.61) are now introduced. At first, notice that the ground speed changes depending on the course. Therefore, a feedforward term is added to the estimator equation, which is

$$F_{sl} = \frac{\partial V_g'}{\partial \chi'}\dot{\chi}' = \frac{\partial V_g'}{\partial \chi'}\left[-\chi_\infty\frac{2}{\pi}\frac{k_{sl}}{1+(k_{sl}e_{py})^2}\sin(\chi'-\chi_q) - \kappa_{sl}\text{sat}\left(\frac{\tilde{\chi}'}{\varepsilon_{sl}}\right)\right] \tag{4.62}$$

The relationship between V_g' and χ' is obtained substituting equation (4.51) in equation (4.53). Taking the squared norm of both sides, one obtain:

$$V_g'^2 - 2V_g' \begin{bmatrix}\cos\chi' \\ \sin\chi'\end{bmatrix}^T \begin{bmatrix}w_n \\ w_e\end{bmatrix} + W'^2 + V_a^2 = 0 \tag{4.63}$$

where obviously $W' = \sqrt{w_n^2 + w_e^2}$. Solving (4.63) for V_g' and taking the positive solution gives:

$$V_g' = W'\cos(\psi_w' - \chi') + \sqrt{V_a^2 - W'^2\sin^2(\psi_w' - \chi')}. \tag{4.64}$$

The whole wind field is unknown, so in this last equation, W' and ψ_w' are approximated with the constant components W and ψ_w. This lets the derivative $\partial V_g'/\partial\chi'$ be in the following fashion:

$$\begin{aligned}\frac{\partial V_g'}{\partial\chi'} \simeq \frac{\partial V_g}{\partial\chi'} =& W\sin(\psi_w - \chi') \\ &+ \left(V_a^2 - W^2\sin^2(\psi_w - \chi')\right)^{\frac{1}{2}} W^2\sin(\psi_w)\cos(\psi_w - \chi')\end{aligned} . \tag{4.65}$$

A second modification is introduced. This time, it is made to prevent the possibility of an estimate drift in presence of a bounded disturbance (wind), and is called σ-modification [24], [25]. This modification adds damping to the estimator dynamics.

The estimator in its final form is given by

$$\dot{\hat{V}}_g' = \Gamma_{sl}\mu_{sl}\tilde{\chi}'\chi_\infty\frac{2}{\pi}\frac{k_{sl}}{1 + (k_{sl}e_{py})^2}\sin(\chi' - \chi_q) + F_{sl} - \sigma_{sl}\Gamma\hat{V}_g'. \tag{4.66}$$

4.6.2 Orbit

The procedure to build up the V_g' estimator for the orbit path follows exactly the procedure just presented for the straight-line.

It is of no benefit to report all the calculations so the final $\dot{\hat{V}}_g'$ is directly written as:

$$\dot{\hat{V}}_g' = -\Gamma_o\mu_o\tilde{\chi}'\left(\frac{1}{d}\sin(\chi' - \gamma) + \lambda\beta_o\cos(\chi' - \gamma)\right) + F_{\text{orbit}} - \sigma_o\Gamma_o\hat{V}_g' \tag{4.67}$$

where F_{orbit} is the feed-forward term defined as

$$\dot{\hat{V}}_g' = \frac{\partial V_g'}{\partial\chi'}\left[\frac{\hat{V}_g'}{d}\sin(\chi' - \gamma) + \lambda\beta_o V_g'\cos(\chi' - \gamma) - \kappa_o\text{sat}\left(\frac{\tilde{\chi}'}{\varepsilon_o}\right)\right] \tag{4.68}$$

and the derivative is defined as in equation (4.65).

4.7 Conclusions

The discussed guidance law is then implemented in the simulator. Both versions, with and without adaptation, can be run. In Figure 4.11 one can see the path following sub-block, divided in straight-line and orbit. The input *GuidData* is a bus-signal which contains all the information of the desired path to be followed and allows the switch between straight-line and orbit control by means of the *Path flag* variable. In Figures 4.12 and 4.13 the specific implemented path following strategy previously discussed is shown. Both are structured in the same way: at first there is the path error computation (e_{py} or $\tilde{d}$), then the desired course χ_d is computed. Note that in the case of the orbit, the variables γ and χ needed to be unwrapped, since they are computed with the atan2 function and then always defined in $[-\frac{\pi}{2}, \frac{\pi}{2}]$. This is not a problem in the case of the simplified simulation, since the dynamics of those variables are the result of mathematical integration. In Figure 4.14 it is possible to see the whole simulator. It also includes a Path manager sub-block which can be used in the future to include a state-machine able to switch from the straight-line to the orbit path following according to the defined flight mission.

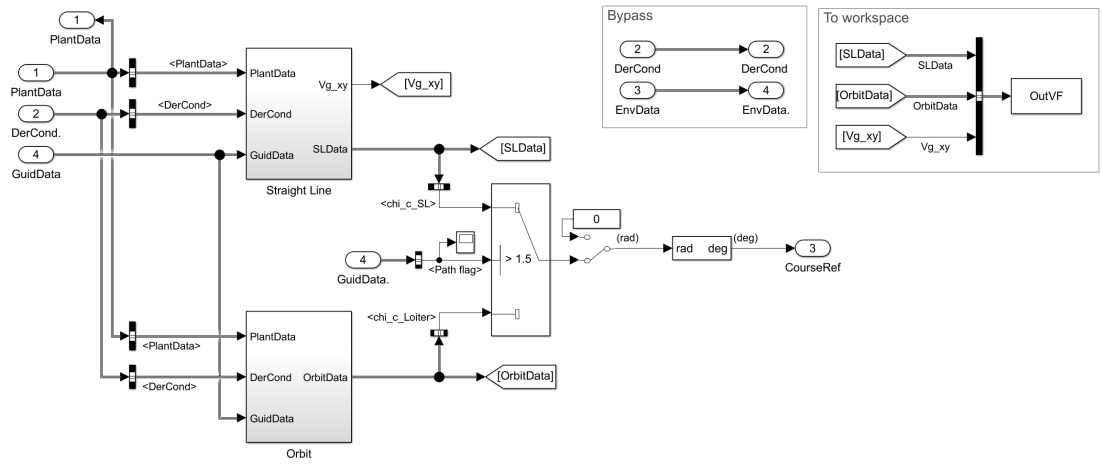


Figure 4.11: Path following sub-block

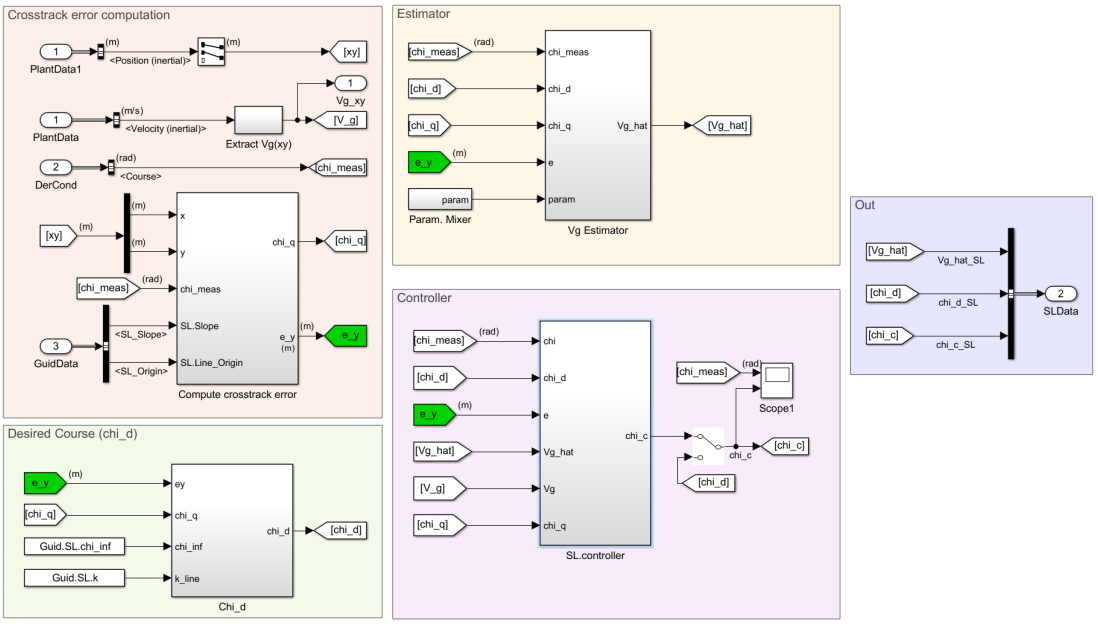


Figure 4.12: Straight-line path following sub-block

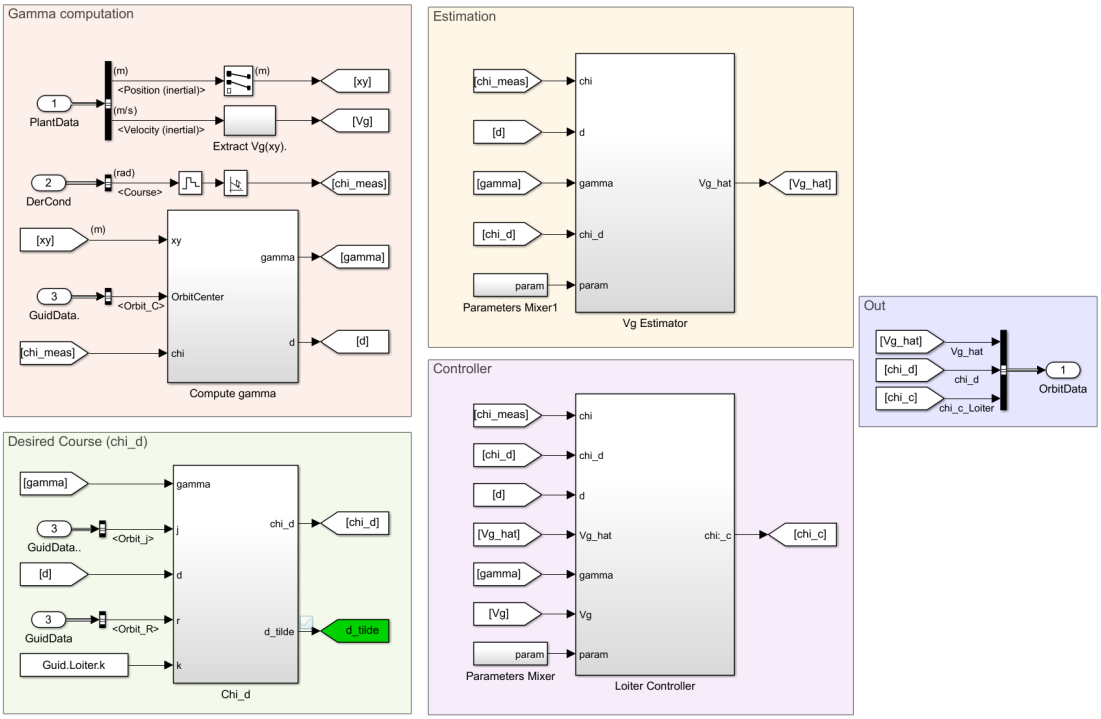


Figure 4.13: Orbit path following sub-block

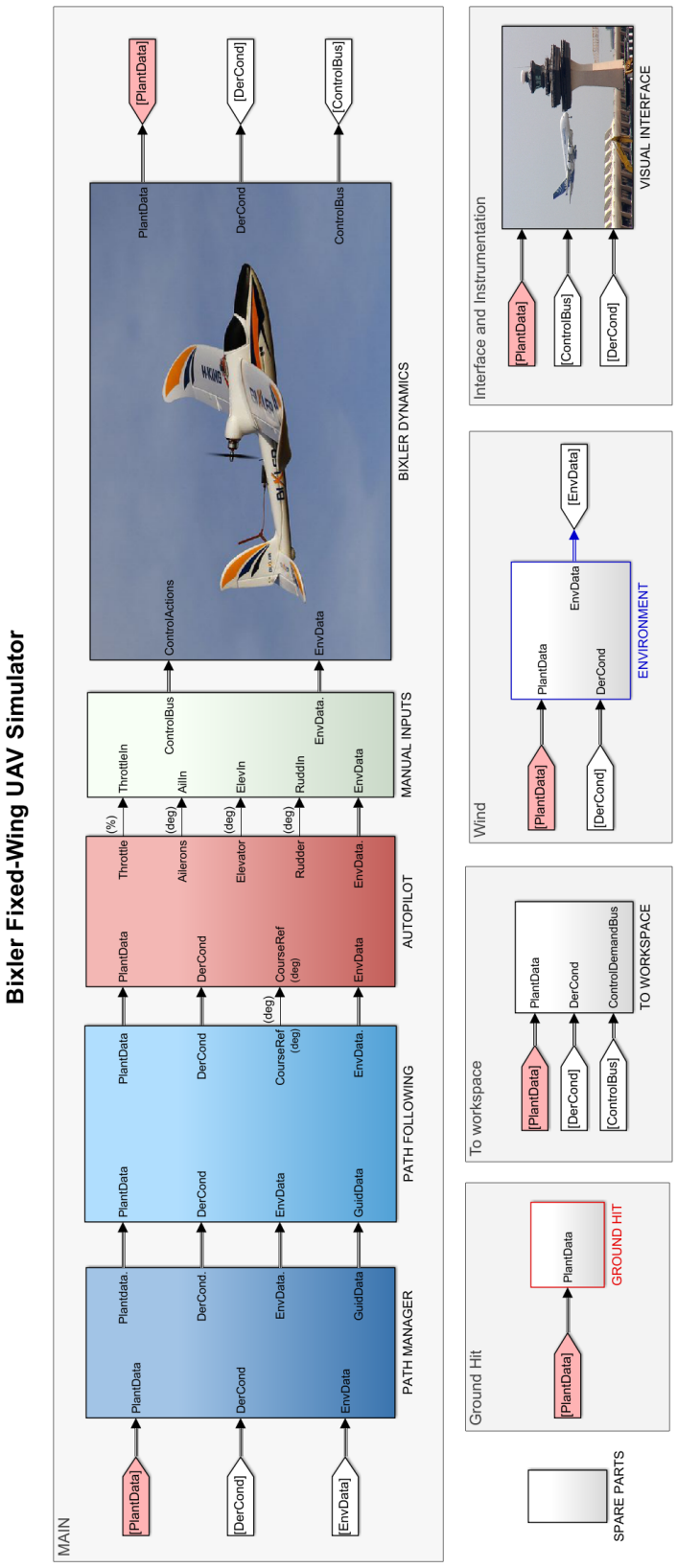


Figure 4.14: Main Simulink simulator block

Chapter 5

Simulation results

In this chapter simulations results are shown of the Vector Field guidance applied to the real UAV dynamics are presented and discussed.

At first, the Vector Field approach is tested without adaptive term. In this way, it is possible to understand the effects of k, κ and ε on the UAV path following. The chosen values are then indicated, and some simulations in different environmental conditions are presented to evaluate the robustness. Using the same conditions, soon thereafter, the Vector Field with adaptive term is considered and the results with the tuned Γ and σ values are shown. Finally, a comparison between both variants follows.

All the simulations are conducted with a constant commanded airspeed $V_{a,c} = 15$ m/s, a constant commanded height of $h_c = 50$ m, and with the initial conditions indicated in Table 5.1. Moreover, the parameter χ_∞ is set at $\frac{\pi}{2}$; it is useless to reduce it, causing the increase of the trajectory settling time.

State	Initial condition
$[p_n, p_e, p_d]$	$[0, 100, -50]$ m
$[u, v, w]$	$[15, 0, 0]$ m/s
$[\phi, \theta, \psi]$	$[0, 0, 0]$ deg
$[p, q, r]$	$[0, 0, 0]$ deg/s

Table 5.1: Initial conditions for the VF simulations

Name	Wind	Constant wind intensity	Dryden turbulences
$\mathcal{C}_0$	Absent	0 m/s	No
$\mathcal{C}_1$	Low	1 m/s	Yes
$\mathcal{C}_2$	Medium	4 m/s	Yes
$\mathcal{C}_3$	High	7.5 m/s	Yes

Table 5.2: Wind intensity for each test condition

The results are presented under four representative environmental conditions: absence of wind, low wind, medium wind, and high wind (Table 5.2). The considered average wind speeds represent, respectively the 6%, 25%, and 50% of the UAV airspeed. The constant wind direction ψ_w is set to 220 deg. The Dryden wind model, which constitutes the wind dynamic part, is configured following the Military specifications MIL-F-8785C [26], which considers at an altitude of 50 m a turbulence intensity of 2.15 m/s on the $\mathbf{i}^b$ and $\mathbf{j}^b$ axes, and 1.4 m/s on the $\mathbf{k}^b$ axis; the wavelengths σ_u, σ_v, and σ_w are of 200 m.

The performance is evaluated using the steady-state Root-Mean Square of the tracking error (e_{py} for the straight-line, and $\tilde{d}$ for the orbit) from the first moment when $e_{py} < 0.1$ m, or $\tilde{d} < 0.1$ m. The formula is:

$$X_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|X_n|^2} \tag{5.1}$$

where X_N is the considered error signal and N is the number of the simulation data samples.

5.1 VF guidance without adaptive term

5.1.1 Straight-line case

To tune the characteristic parameters k_{sl}, κ_{sl}, and ε_{sl} a preliminar study of their incidence on the guidance is conducted. A reference line directed north and passing from the origin of $\mathcal{F}_i$ is chosen for simplicity, so that the cross-track error is just $e_{py} = p_e$.

The dependency from k_{sl} will be analysed first. Parameters ε_{sl} and κ_{sl} are held constant at $\varepsilon_{sl} = 1$ and $\kappa_{sl} = \frac{\pi}{2}$. Figure 5.1 shows the plane $(\mathbf{i}^i, \mathbf{j}^i)$ where the UAV transition is moderated by the value of k_{sl}: a higher value corresponds to a faster convergence to the reference line. However, setting a too high value will result in oscillations since the demanded course angle χ_d varies quickly and the controller is not able to minimise $\tilde{\chi}$. This is shown in Figure 5.2.

Similar considerations are drawn when investigating now the sensitivity to ε_{sl} and κ_{sl}. If ε_{sl} is reduced too much, then the term $\tilde{\chi}/\varepsilon_{sl}$ increases and it is more likely that $\tilde{\chi} \geq \varepsilon_{sl}$, where the 'sat' function behaves like the 'sign' function, thus causing chattering in the control action (Figure 5.3). Since chattering does not happen when $\varepsilon_{sl} = 1$, it will be hereafter set to 1.

The value κ_{sl} determines directly the aggressiveness of the control action. Large values drive $\tilde{\chi}$ to zero quickly. Again, this parameter has to be tuned in a realistic way, according to the course tracking capability of the UAV. By exceeding, one incurs again in oscillations about the reference line. The effect of

these parameters from the sliding mode control point of view is explained in detail in [21].

After some iterations, the following parameters provided a satisfactory performance and did not lead to any oscillation in both e_{py} and $\tilde{\chi}$:

$$k_{sl} = 0.02 \tag{5.2}$$

$$\kappa_{sl} = \frac{\pi}{2} \tag{5.3}$$

$$\varepsilon_{sl} = 1. \tag{5.4}$$

Firstly, in Figure 5.4 is shown the trajectory of the UAV in absence of wind, and it is compared to the simplified dynamics; the small difference is due to the roll angle initial saturation from the complete model. Here the cross-track error goes to zero as time increases.

In $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ the errors have the behaviour of Figure 5.5 and their steady-state RMS is summarised in Table 5.3. It is possible to see how in high wind conditions the controller works hard to maintain the UAV on the reference line. Notice that wind comes from North-East and acts both on the side of the airframe, thus deviating it from the desired line, and, from the front, thus decreasing the ground speed.

Wind	Steady e_{py} RMS error
Absent	0.0357m
Low	0.1112m
Medium	0.1990m
High	1.3218m

Table 5.3: Vector Field steady RMS errors in the standard VF straight-line path following

To prove that the Vector Field approach works with any line and initial UAV posture, let us consider a generic line ($\mathbf{q} = [1, 3]$ 1/s and $\mathbf{r} = [0, 10]$ m), an initial yaw angle of 120 deg, and the $\mathcal{C}_3$ condition with constant wind coming from East (270 deg). The UAV trajectory is shown in Figure 5.6.

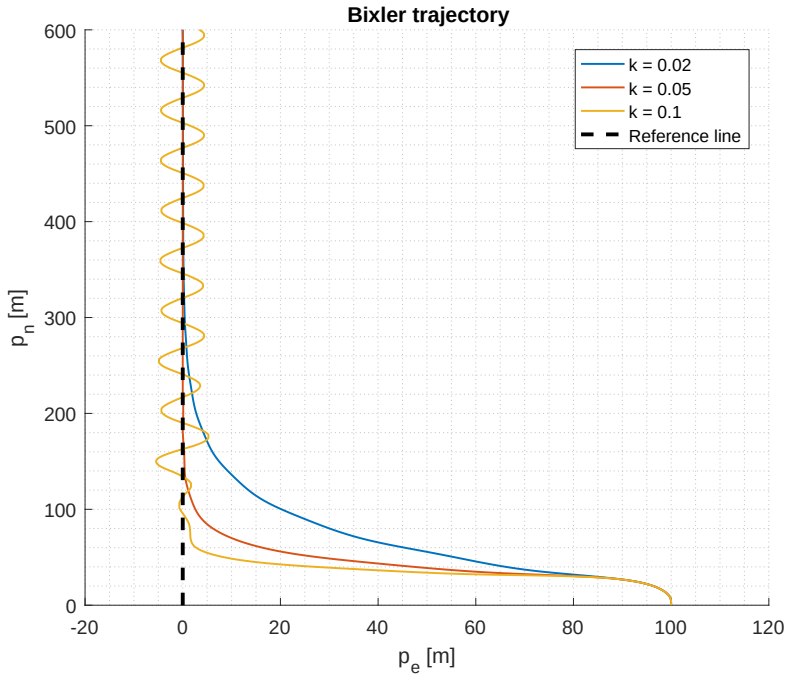


Figure 5.1: Line following transitions for different values of k_{sl}

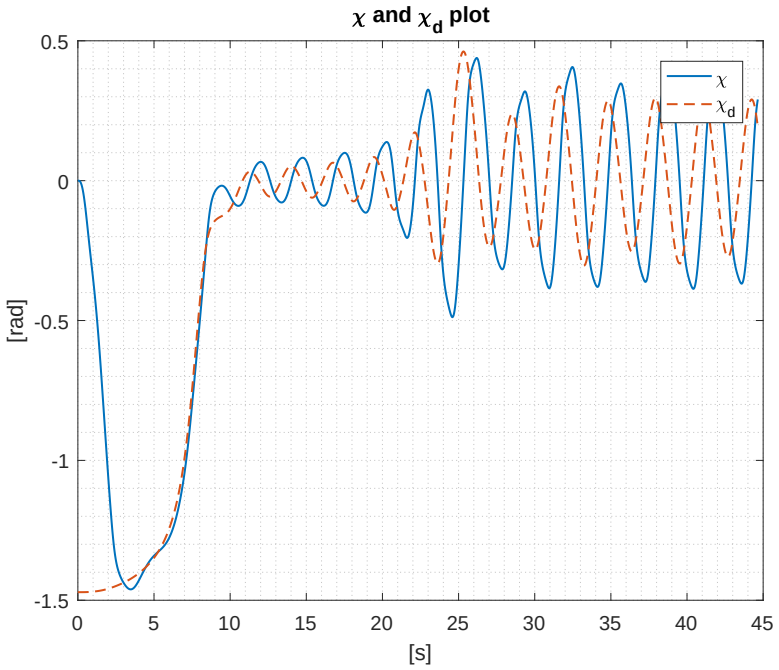


Figure 5.2: χ and χ_d when $k = 0.1$

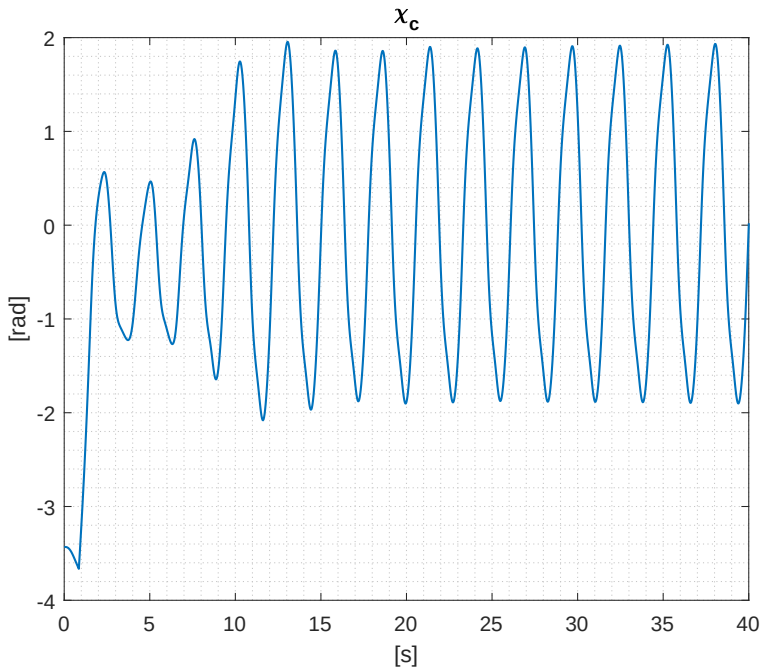


Figure 5.3: Chattering behaviour of the control variable when $\varepsilon = 0.5$

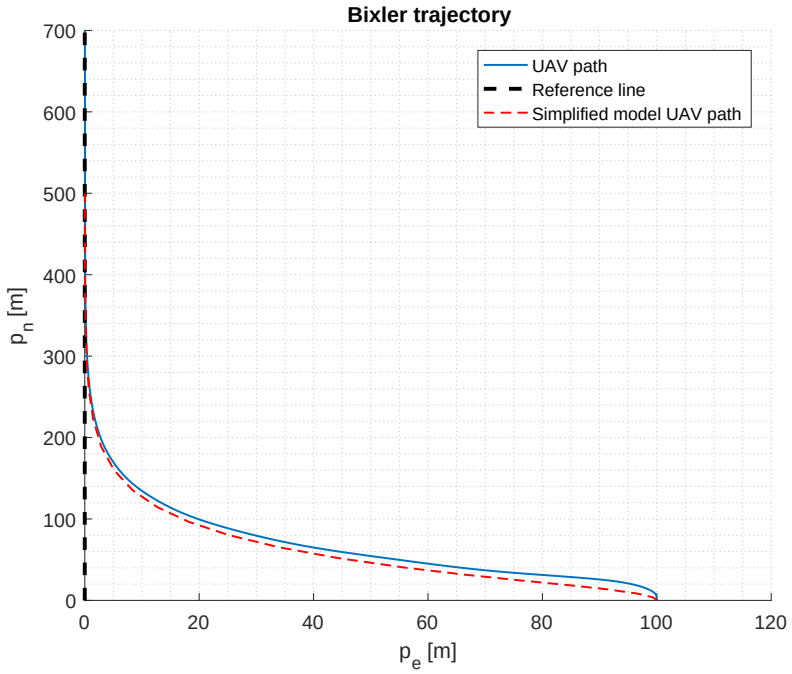


Figure 5.4: UAV trajectory from the full and simplified models during the standard VF straight-line path following (absence of wind)

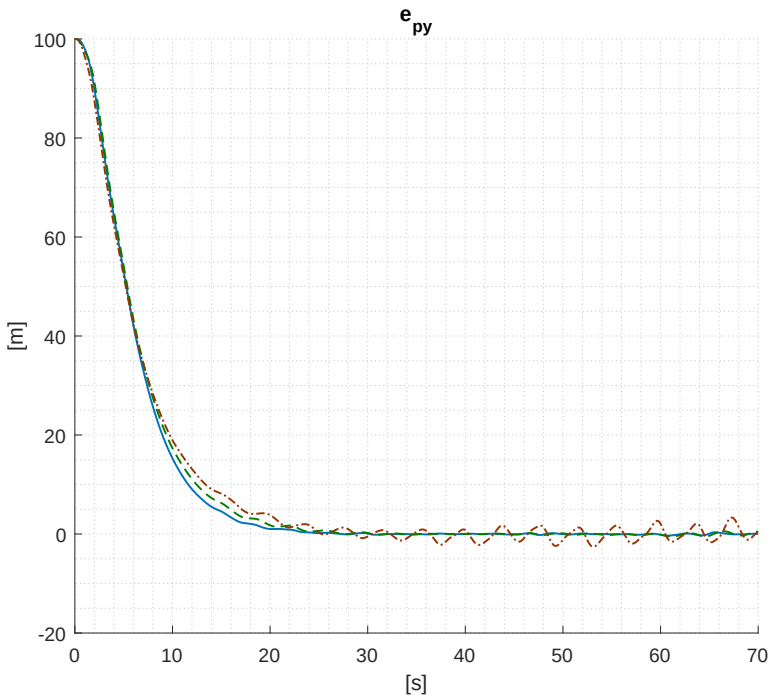


Figure 5.5: e_{py} dynamics for different wind conditions

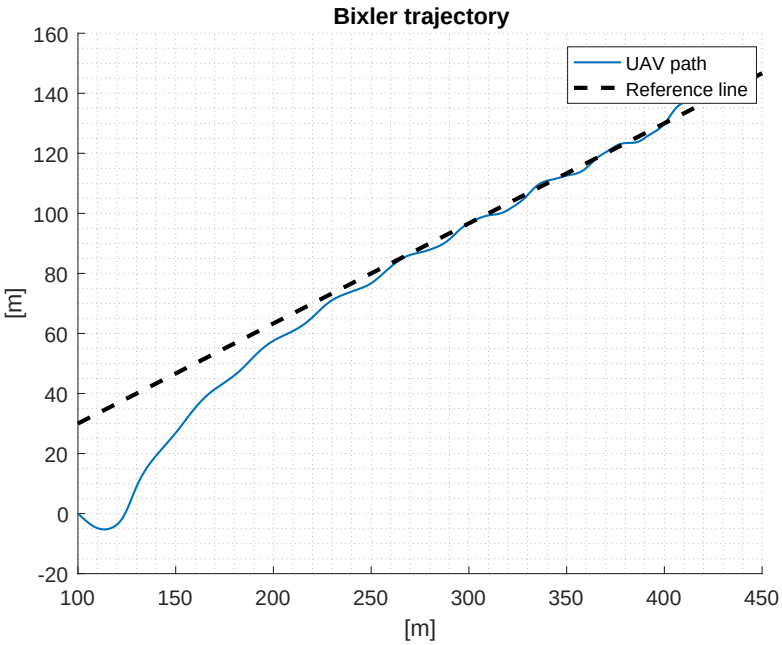


Figure 5.6: Standard VF straight-line path following with a different initial heading and reference line orientation

5.1.2 Orbit case

Varying the tuning parameters has the same effects in the orbit case; as an example, the differences in changing k_o are shown in Figure 5.7, keeping constant $\kappa_o = \frac{\pi}{2}$ and $\varepsilon_o = 1$.

Chosen values for the orbit are:

$$k_o = 0.01 \tag{5.5}$$

$$\kappa_o = \frac{\pi}{2} \tag{5.6}$$

$$\varepsilon_o = 1. \tag{5.7}$$

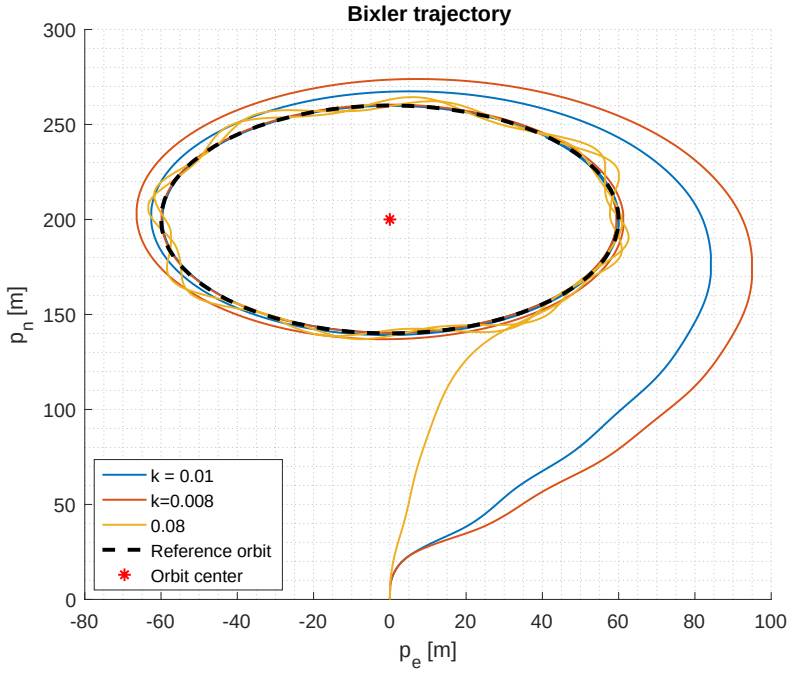


Figure 5.7: Orbit following transitions for different values of k_o

The same comparison in absence of wind between the simplified and full models is performed and shown in Figure 5.8. The minimum turning radius for the aircraft is $R_{min} = 25$ m and for the simulations a radius ϱ of 60 m is chosen.One can notice that in absence of wind the tracking error does not go to zero.

The steady-state errors $\tilde{d}$ are represented in Figure 5.9, while steady RMS errors are reported in Table 5.4.

Again, the generalised trajectory during high wind coming from East, initial yaw angle of 120 deg, and circle center at $\mathbf{c} = [100, 0]$ m, is depicted in Figure 5.10.

Wind	Steady $\tilde{d}$ RMS error
Absent	0.4437 m
Low	1.3691 m
Medium	2.5433 m
High	4.3791 m

Table 5.4: Steady RMS errors using the standard VF orbit path following

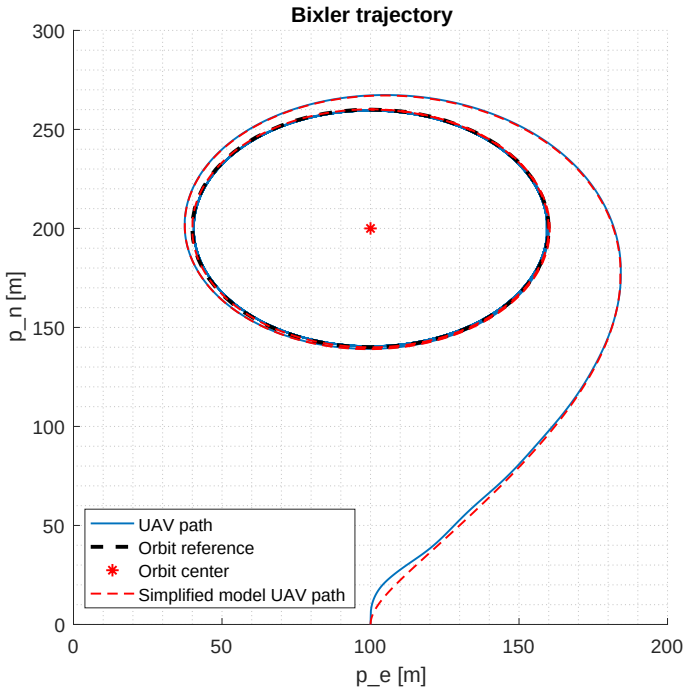


Figure 5.8: UAV trajectory from the full and simplified models during the standard VF orbit path following (absence of wind)

5.2 VF guidance with adaptive term

5.2.1 Straight-line case

Before showing the results of the Vector Field strategy using the estimate $\tilde{V}_g'$, the gains Γ and σ have to be tuned. The former is the estimator gain and determines its dynamics, while the latter is a particular damping factor which makes $\hat{V}_g'$ decrease proportionally to its magnitude. A too high Γ_{sl} would lead to an estimate drift and then instability.

After some iterations the chosen value is:

$$\Gamma_{sl} = 0.1. \tag{5.8}$$

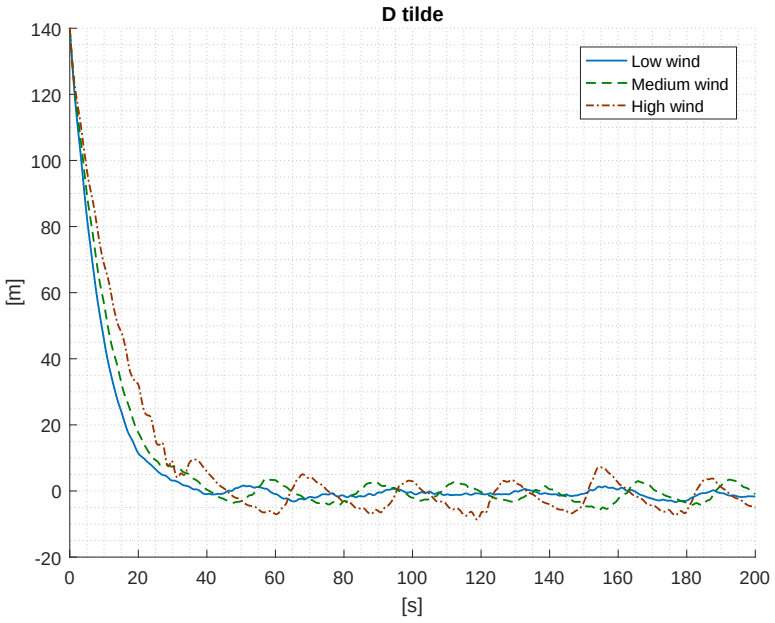


Figure 5.9: $\tilde{d}$ dynamics for different wind conditions

With this Γ_{sl} value, the estimates did never diverge. Therefore, $\sigma_{sl} = 0$.

It is also necessary to define the parameter μ_{sl}. Being μ_{sl} a scaling term aimed to make the tracking error and the course error comparable, it is approximated as:

$$\mu_{sl} \approx \left(\frac{e_{py,\max}}{\pi}\right)^2. \tag{5.9}$$

In other words, μ_{sl} is the squared ratio between the maximum lateral error $e_{py,\max}$ and the maximum possible course error $\chi - \chi_d$. In the simulations $e_{py,\max}$ is set as the distance of the UAV from the line in the first time instant (when usually the error e_{py} is maximum).

Again, we take the initial conditions of Table 5.1. In absence of wind, the estimator behaves as in Figure 5.11.

For each wind condition, the estimator is not able to track in any way the dynamic part of the wind, which is generated by the Dryden turbulence model. An example for medium wind is given in Figure 5.13. This do not surprises since, from the analysis made in [1] using the simplified point mass model of the UAV, the steady RMS estimation error is always large. This essentially makes the performance of the adaptive Vector Field approach for straight-lines unvaried with respect to the standard Vector Field case. Table 5.5 provides a summary of the simulation results.

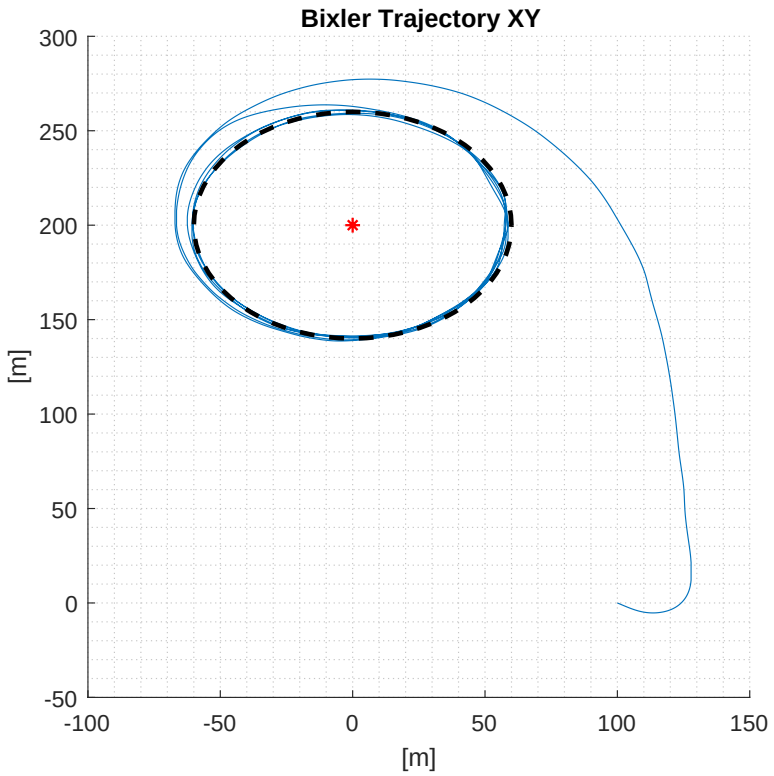


Figure 5.10: Standard VF orbit path following with a different initial heading and reference line orientation

5.2.2 Orbit case

The orbit path case offer a slightly different scenario.

Before starting, here coefficient μ_o, for the same principles explained in the previous section, is:

$$\mu_o \approx \left(\frac{d_{\max}}{\pi}\right)^2 \tag{5.10}$$

where $d_{\max}$ is d when $t = 0$.

In this case, by trial and error, the gain Γ_o is chosen as:

$$\Gamma_o = 0.1 \tag{5.11}$$

and σ_o is set again to zero since no drift is present.

At first, it is worth seeing what happens in presence of wind. It was shown that zero steady-state error was not achieved in the orbit case. With this adaptive term, since its dynamics is determined primarily by $\tilde{\chi}$, the error is driven to zero. The estimation performance and the $\tilde{d}$ evolution are shown in Figures 5.14 and 5.15.

Wind	Steady RMS estimation error	Steady e_{py} RMS error
Absent	0.3550 m/s	0.0266 m
Low	1.4548 m/s	0.1111 m
Medium	1.6660 m/s	0.1942 m
High	2.7428 m/s	1.4385 m

Table 5.5: Steady RMS tracking and estimation errors using the adaptive VF straight-line path following

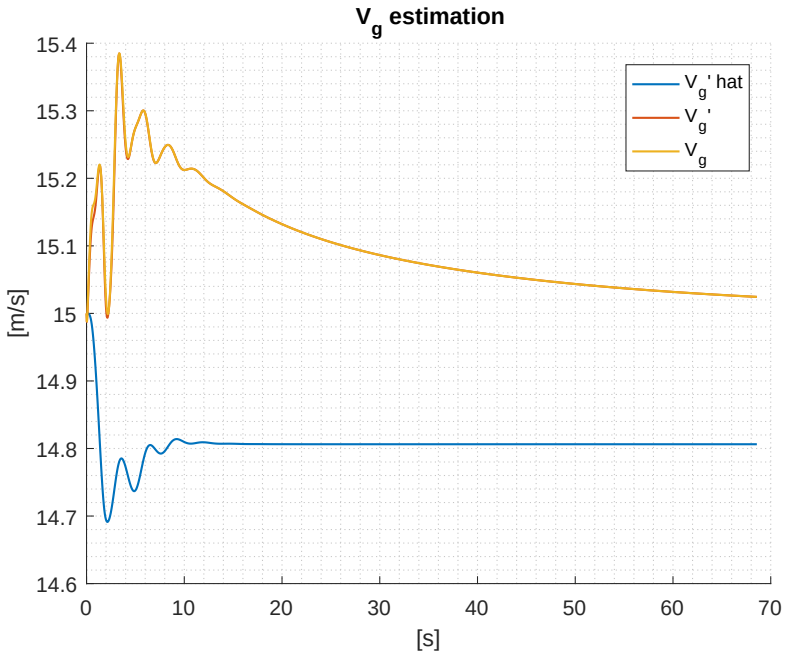


Figure 5.11: Estimator behaviour (absence of wind) in the straight-line case using the adaptive VF

When wind is present, the reference tracking is better as well, as shown in Table 5.6. The steady-state error is less than the standard VF approach even if the V_g' tracking is achieved with a large estimation error. For example, Figure 5.16 reports the trend of $\hat{V}_g$ with medium wind.

5.3 Final considerations

It is clear that the Vector Field adaptive variant has some pros and some cons. We start considering that in any case it does not worsen the standard Vector Field approach based on the knowledge of the constant wind field.

In the straight-line case, the estimator is not able to track the rapidly changing dynamic component of the wind, also because, as shown in [1], the line tracking

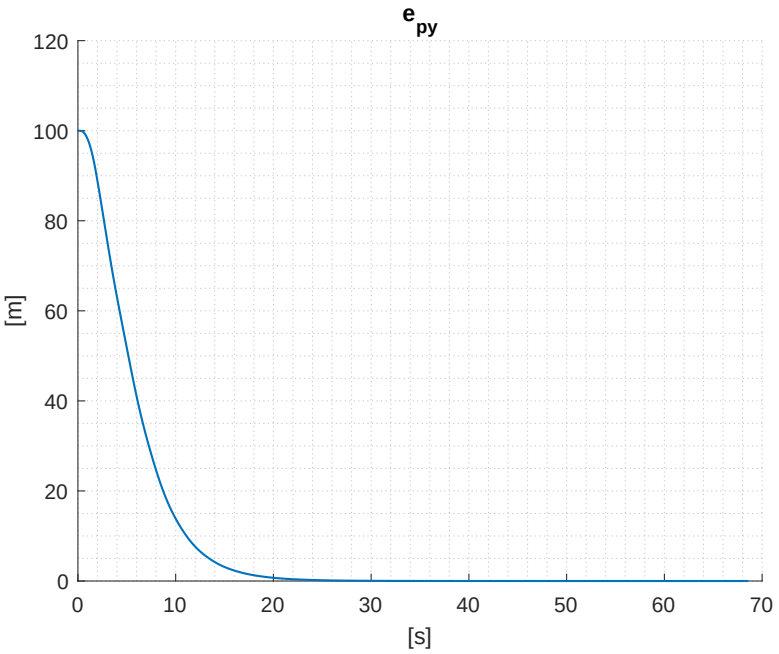


Figure 5.12: Error e_{py} behaviour (absence of wind) in the straight-line case using the adaptive VF

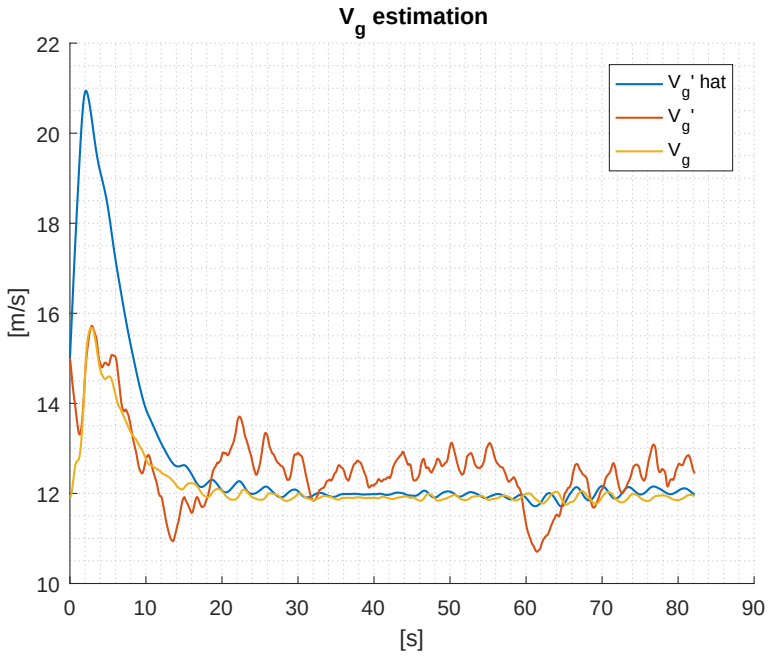


Figure 5.13: Estimator behaviour (medium wind) in the straight-line case using the adaptive VF

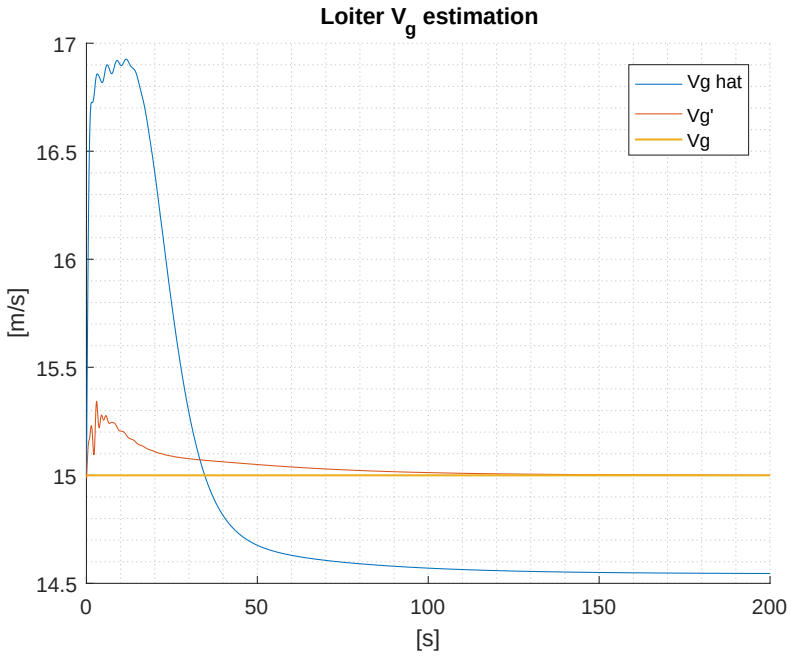


Figure 5.14: Estimator behaviour (absence of wind) in the orbit case using the adaptive VF

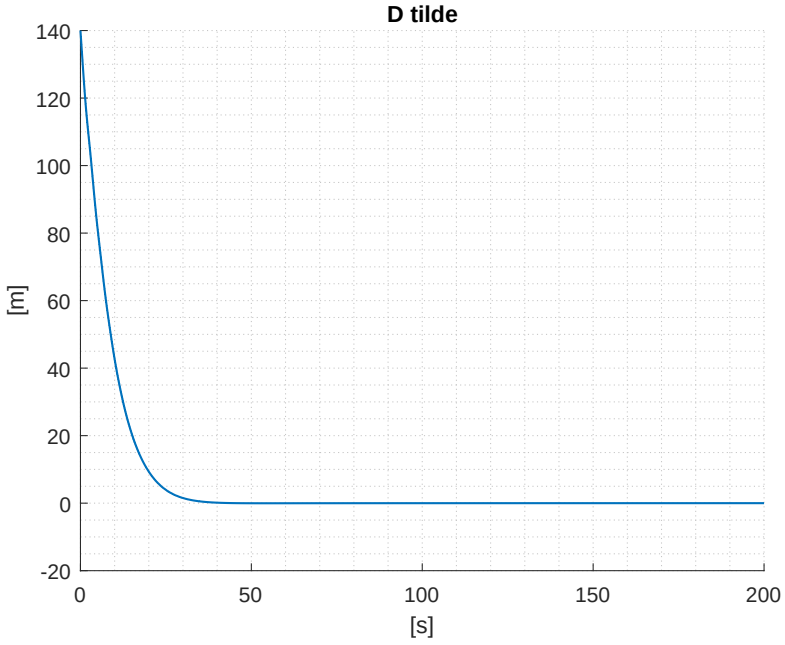


Figure 5.15: Error $\tilde{d}$ in the orbit case using the adaptive VF

Wind	Steady RMS estimation error	Steady $\tilde{d}$ RMS error
Absent	1.5785 m/s	0.0145 m
Low	1.7258 m/s	0.5877 m
Medium	2.7043 m/s	0.8987 m
High	6.6419 m/s	1.5072 m

Table 5.6: Steady RMS tracking and estimation errors using the adaptive VF orbit path following

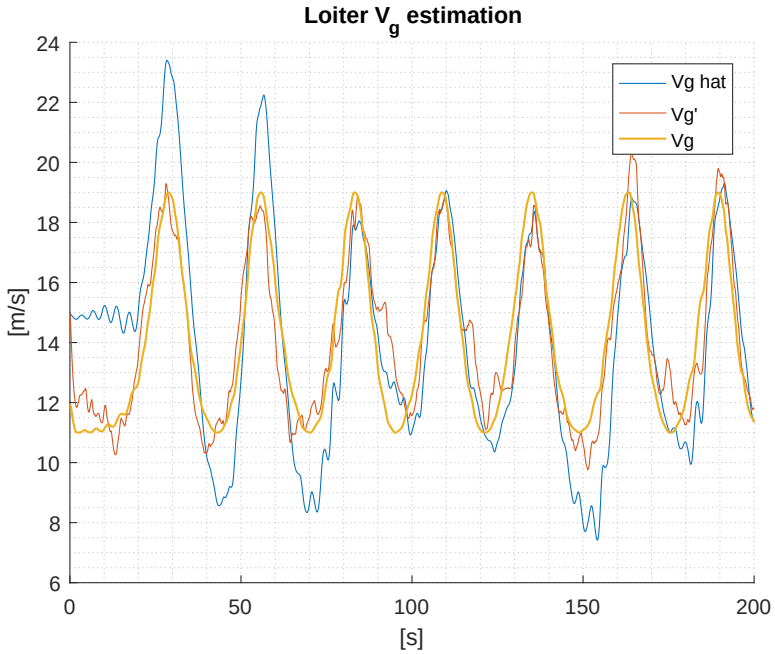


Figure 5.16: Estimator behaviour (medium wind) in the orbit case using the adaptive VF

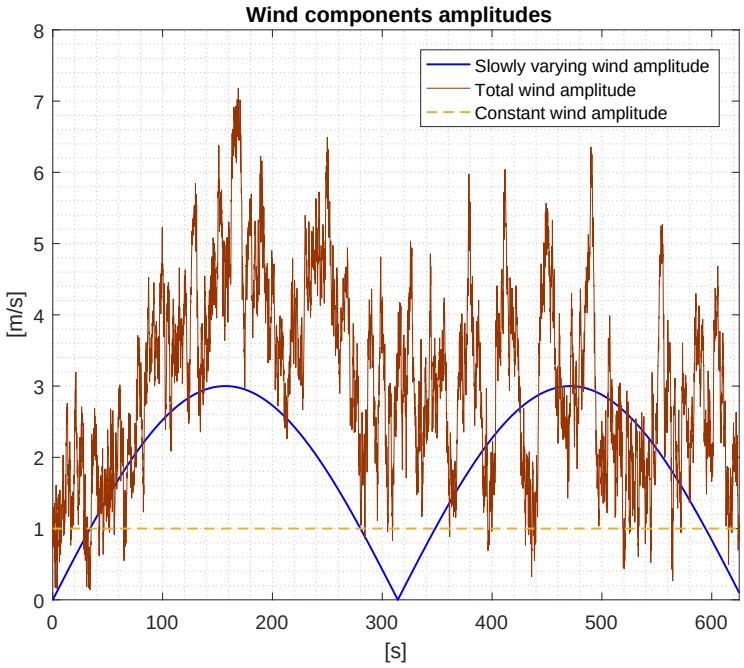


Figure 5.17: Wind amplitudes with a new slowly varying component

implies some lack in persistent excitation. The estimator just tracks the known V_g amplitude.

In the orbit case, the estimator is naturally stimulated since the amplitude of the wind vector that acts on the airframe varies as the UAV orbits around the path.

When wind is absent, the steady state error of $\tilde{\chi}$ enters the estimator dynamics which can counteract it and leads the UAV on the desired path with zero steady-state error. This shows also how the estimator dynamics can have a beneficial effect in a way not generally related to the wind disturbance presence.

In order to verify the guidance performance in presence of a larger dynamic wind component, let us define a new wind vector $\mathbf{V}_{\text{new}}$ as:

$$\mathbf{V}_{\text{new}} = \begin{bmatrix} V_{\text{new},x} \\ V_{\text{new},y} \end{bmatrix} = A_{\text{new}}(t) \begin{bmatrix} \cos \psi_{w,\text{new}}(t) \\ \sin \psi_{w,\text{new}(t)} \end{bmatrix} \tag{5.12}$$

where $A_{\text{new}}(t) = 3 \sin(0.01t)$ and $\psi_{w,\text{new}}(t) = \pi \sin(0.01t)$. New simulations are then run with the constant, slowly, and rapidly varying wind terms in order to perform a new comparison between the standard and adaptive Vector Field. The three wind amplitudes are represented in Figure 5.17.

The results are summarised in Table 5.7. It can be seen that the performance in the straight-line is again not improved, while in the orbit case there is a reduction of the steady-state RMS error. Other simulations showed that removing the

Straight-line		Orbit	
Standard	Adaptive	Standard	Adaptive
0.0964 m	0.0963 m	2.5016 m	2.1785 m

Table 5.7: Comparison of the standard VF and adaptive VF in both straight-line and orbit cases with a new slowly time-varying wind component

Dryden wind model, lowering the $A_{\text{new}}(t)$ and $\phi_{w,\text{new}}(t)$ frequencies from 0.01 rad/s to 0.001 rad/s, and increasing Γ, the steady-state error decreases and estimation error consistently decreases.

The above discussion allows us to draw some conclusions:

1. The adaptive VF gives great benefits in the orbit following rather than the straight-line case.
2. The estimation shows its benefits in scenarios where there is a slowly-varying wind and an orbit path reference;
3. The Vector Field method has three parameters to tune, and the introduction of the estimator requires two more parameters to be chosen;
4. The UAV dynamics plays a role in limiting the performance of the estimator, since it limits the maximum Γ value;
5. The estimation has usually a burst during the initial transient which can be limited with a suitable choice of σ. One solution can be letting $\sigma = \hat{V}_g' - V_g$, when $\hat{V}_g' - V_g$ overcome a certain threshold, otherwise it is zero.

It is then evident that introducing a new dynamics is a powerful element that has to be treated with care and that this strategy has to be studied and deepened more, in particular with the objective of a future implementation on a real UAV. Given these results, it is possible to affirm that the adaptive Vector Field can be effectively used in contexts where a precise orbit path following is essential for the mission goals.

Conclusions

In this thesis, several aspects of the guidance and control for a fixed-wing UAV have been tackled. The primary goal of the thesis was build a reliable framework where the new Vector Field guidance algorithm could be implemented in simulation and then tested on a real vehicle. An accurate path tracking feature is a key aspect for the new frontier of applications for the fixed-wing UAVs.

A variety of tasks have been conducted, requiring both a theoretical understanding of the phenomena governing the flight mechanics, its control, and the on-field practice. A functional and modular simulator has been built using the complete dynamics UAV model; it can be interfaced with FlightGear, thus helping in the flight visualization. Moreover, it integrates the low-level controllers structure of ArduPilot, one of the most common flight software suite, that gives two benefits at the same time: the use of the ArduPilot software features, such as already implemented state estimators, sensor drivers, failsafe features, and so on, and the possibility to implement a new guidance law on a real UAV while being able to simulate its behaviour with a good accuracy. Even if there is the lack of an open-loop validation, the closed-loop lateral response comparison between simulations and flight tests gave good results.

Relatively to the adaptive Vector Field, the results shows an improvement in the orbit path tracking performances: this can be useful to improve performance in missions where a loiter flight above a ground target is crucial, for example in the search and rescue, during aerial rendezvous, or surveillance. It would be interesting to continue improving the adaptation provided by the V_g' estimator trying to improve also the straight-line tracking performances.

Future work

The possible further developments of this project are several and are left here as suggestions:

- Implement the adaptive Vector Field on the UAV;
- Perform a comparison between the ArduPilot guidance strategy and the Vector Field approach;

- Study possible improvements in the adaptive Vector Field approach;
- Study possible improvements in the V_g' estimation, for example trying to exploit the ground speed GPS measurements, after a study of its accuracy and uncertainty;
- Perform the UAV model identification and validation;
- Implement a Path Manager to evaluate the adaptive Vector Field in presence of complicated paths with multiple waypoints, hence a situation close to a real mission.

Bibliography

[1] Bingyu Zhou, Harish Satyavada, and Simone Baldi. Adaptive path following for unmanned aerial vehicles in time-varying unknown wind environments. In *American Control Conference (ACC), 2017*, pages 1127–1132. IEEE, 2017.

[2] Mattia Giurato. Design, integration and control of a multirotor uav platform. Master's thesis, Politecnico di Milano, 2015.

[3] University of Sydney, 2006. URL `http://www-mdp.eng.cam.ac.uk/web/library/enginfo/aerothermal_dvd_only/aero/propeller/prop1.html`.

[4] Robert F Stengel. *Flight dynamics*. Princeton University Press, 2015.

[5] Jack W Langelaan, Nicholas Alley, and James Neidhoefer. Wind field estimation for small unmanned aerial vehicles. *Journal of Guidance Control and Dynamics*, 34(4):1016, 2011.

[6] FlightGear Flight Simulator, 2017. URL `http://www.flightgear.org/`.

[7] University of Stanford. Aa241x problem set 2 report, 2006. URL `http://web.stanford.edu/~rsur/Chimera%20data/AA241X_Chimera_PS2_Report.pdf`.

[8] John E Williams and Steven R Vukelich. The usaf stability and control digital datcom. volume i. users manual. Technical report, MCDONNELL DOUGLAS ASTRONAUTICS CO ST LOUIS MO, 1979.

[9] Description of digital datcom, 2017. URL `http://www.pdas.com/datcomDescription.html`.

[10] Anwar Ul Haque, Waqar Asrar, Ashraf Ali Omar, Erwin Sulaeman, and JS Ali. Comparison of digital datcom and wind tunnel data of a winged hybrid airship's generic model. In *Applied Mechanics and Materials*, volume 629, pages 36–41. Trans Tech Publ, 2014.

[11] Naca airfoil generator, 2017. URL `http://airfoiltools.com/airfoil/naca4digit`.

[12] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[13] Davide Del Cont Bernard, Fabio Riccardi, Mattia Giurato, and Marco Lovera. A dynamic analysis of ground effect for a quadrotor platform.

[14] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[15] Paolo Bolzern, Riccardo Scattolini, and Nicola Schiavoni. *Fondamenti di controlli automatici*. McGraw-Hill Libri Italia, 1998.

[16] Katsuhiko Ogata and Yanjuan Yang. *Modern control engineering*. Prentice-Hall Englewood Cliffs, NJ, 1970.

[17] Roll, pitch and yaw controller tuning, 2016. URL `http://ardupilot.org/plane/docs/roll-pitch-controller-tuning.html`.

[18] John H Blakelock. *Automatic control of aircraft and missiles*. John Wiley & Sons, 1991.

[19] LF Faleiro and AA Lambregts. Analysis and tuning of a 'total energy control system'control law using eigenstructure assignment. *Aerospace science and technology*, 3(3):127–140, 1999.

[20] Sanghyuk Park, John Deyst, and Jonathan P How. A new nonlinear guidance logic for trajectory tracking. In *AIAA guidance, navigation, and control conference and exhibit*, pages 16–19, 2004.

[21] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.

[22] PB Sujit, Srikanth Saripalli, and Joao Borges Sousa. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems*, 34(1):42–59, 2014.

[23] Hassan K Khalil. *Nonlinear systems*. Prentice-Hall, New Jersey, 2002.

[24] Petros A Ioannou and Jing Sun. *Robust adaptive control*, volume 1. PTR Prentice-Hall Upper Saddle River, NJ, 1996.

[25] Eugene Lavretsky and Kevin A Wise. Robust adaptive control. In *Robust and Adaptive Control*, pages 317–353. Springer, 2013.

[26] Military Specification. Flying qualities of piloted airplanes. *United States Department of Defense*, 1980.

[27] Thomas R Kane and David A Levinson. *Dynamics, theory and applications*. McGraw Hill, 1985.

[28] Datcom by holy cows, 2010. URL `http://www.holycows.net/datcom/index.html`.

[29] Ardupilot complete parameters list, 2016. URL `http://ardupilot.org/plane/docs/parameters.html`.

Appendix A

Differentiation of a vector

Suppose to have two reference frames, namely $\mathcal{F}_0$, which is defined by the unit vectors $(\mathbf{i}^0, \mathbf{j}^0, \mathbf{k}^0)$, and $\mathcal{F}_1$, defined by $(\mathbf{i}^1, \mathbf{j}^1, \mathbf{k}^1)$. A vector $\mathbf{p}$ is moving in $\mathcal{F}_1$, and $\mathcal{F}_1$ is rotating, but not translating with respect to $\mathcal{F}_0$. The goal is to find the time derivative of $\mathbf{p}$ as seen from frame $\mathcal{F}_0$.

To do this, call the angular velocity of frame $\mathcal{F}_1$ seen from $\mathcal{F}_0$ as $\boldsymbol{\omega}_{1/0}$. One can express the vector $\mathbf{p}$ in terms of vector components as:

$$\mathbf{p} = p_x\mathbf{i}^1 + p_y\mathbf{j}^1 + p_z\mathbf{k}^1. \tag{A.1}$$

Differentiating with respect to $\mathcal{F}_0$ (indicated with superscript '0'):

$$\left(\frac{d\mathbf{p}}{dt}\right)^0 = \dot{p}_x\mathbf{i}^1 + \dot{p}_y\mathbf{j}^1 + \dot{p}_z\mathbf{k}^1 + p_x\left(\frac{d\mathbf{i}^1}{dt}\right)^0 + p_y\left(\frac{d\mathbf{j}^1}{dt}\right)^0 + p_z\left(\frac{d\mathbf{k}^1}{dt}\right)^0. \tag{A.2}$$

The first three terms on the right-end side of last equation, represent the change in $\mathbf{p}$ as viewed by an observer in the rotating $\mathcal{F}_1$, while the last three terms represent the change in $\mathbf{p}$ due to the rotation of $\mathcal{F}_1$ relative to $\mathcal{F}_0$. As shown in [27], since $\mathbf{i}^1$, $\mathbf{j}^1$ and $\mathbf{k}^1$ are fixed in $\mathcal{F}_1$, their derivatives are:

$$\left(\frac{d\mathbf{i}}{dt}\right)^0 = \boldsymbol{\omega}_{1/0} \times \mathbf{i}^0 \tag{A.3}$$

$$\left(\frac{d\mathbf{j}}{dt}\right)^0 = \boldsymbol{\omega}_{1/0} \times \mathbf{j}^0 \tag{A.4}$$

$$\left(\frac{d\mathbf{k}}{dt}\right)^0 = \boldsymbol{\omega}_{1/0} \times \mathbf{k}^0. \tag{A.5}$$

Differentiating $\mathbf{p}$ with respect to $\mathcal{F}_1$ (indicated with superscript '1') leads to:

$$\left(\frac{d\mathbf{p}}{dt}\right)^1 = \dot{p}_x\mathbf{i}^1 + \dot{p}_y\mathbf{j}^1 + \dot{p}_z\mathbf{k}^1. \tag{A.6}$$

Combining equations (A.2), (A.3), (A.4), (A.6) and (A.6) the final result is:

$$\left(\frac{d\mathbf{p}}{dt}\right)^{0} = \left(\frac{d\mathbf{p}}{dt}\right)^{1} + \boldsymbol{\omega}_{1/0} \times \mathbf{p}. \tag{A.7}$$

Appendix B

DATCOM input

```
DIM M
DERIV RAD
DAMP
BUILD
PLOT

 $FLTCON WT=1.0,NMACH=1.0,MACH=0.03,
NALPHA=20.0,
ALSCHD(1)=-2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0,
             8.0, 9.0, 10.0, 12.0, 14.0, 16.0, 18.0, 19.0, 20.0, 25.0,
NALT=1.0,ALT(1)=10.0,
GAMMA=0.0,LOOP=1.0$

 $SYNTHS XCG=0.28,ZCG=0.0,
XW=0.24,ZW=0.05,
ALIW=1.0,
XH=0.80,ZH=0.035,ALIH=0.0,
XV=0.77,ZV=0.035,
VERTUP=.TRUE.$

 $BODY NX=9.0,X(1)=0.0,0.01,0.05,0.14,0.18,0.22,0.46,0.47,0.85,
R(1)=0.0,0.02,0.04,0.045,0.045,0.045,0.045,0.04,0.02,
ZU(1)=0.0,0.008,0.03,0.053,0.053,0.1,0.12,0.04,0.03,
ZL(1)=0.0,-0.008,-0.03,-0.038,-0.040,-0.0385,-0.022,-0.021,0.0,
BNOSE=2.0$
 $WGPLNF CHRDTP=0.15,CHRDR=0.198,
SSPNE=0.61,SSPN=0.655,
TWISTA=0.0,
SAVSI=0.0,SAVSO=0.0,
CHSTAT=0.25,TYPE=1.0$
```

```
NACA-W-4-1410

SAVE
CASEID MAIN: Bixler
NEXT CASE

 $ASYFLP STYPE=4.0,
NDELTA=9.0,
DELTAL(1)=0.0,1.0,3.0,5.0,8.0,10.0,15.0,20.0,25.0,
DELTAR(1)=0.0,-1.0,-3.0,-5.0,-8.0,-10.0,-15.0,-20.0,-25.0,
SPANFI=0.28,SPANFO=0.51,
CHRDFI=0.055,CHRDFO=0.043$

CASEID AILERONS
SAVE
NEXT CASE

NACA-H-4-0010

 $HTPLNF CHRDTP=0.0883,CHRDR=0.1036,
SSPNE=0.2194,SSPN=0.247,
TWISTA=0.0,
SAVSI=0.0,
CHSTAT=0.25,TYPE=1.0$

NACA-V-4-0010

 $VTPLNF CHRDTP=0.0792,CHRDR=0.128,
SSPNE=0.179,SSPN=0.209,
TWISTA=0.0,
SAVSI=10.0,
CHSTAT=0.25,TYPE=1.0$

 $SYMFLP FTYPE=1.0,NTYPE=1.0,
NDELTA=9.0,
DELTA(1)=-20.0, -10.0, -5.0, -1.0, 0.0, 1.0, 5.0, 10.0, 20.0,
CHRDFI=0.035,CHRDFO=0.034,
SPANFI=0.01,SPANFO=0.173$

 CASEID TOTAL: Bixler
```

Appendix C

Bixler v1.1 parameters

Physical parameters are reported in the following table:

Parameter	Value	Unit
m	1.01	kg
b	1.31	m
c	0.175	m
S	0.228	m^2
$\mathbf{J}$	diag(0.020, 0.026, 0.053)	kg·m^2
S_p	0.031	m^2

Table C.1: List of physical parameters values

Concering aerodynamic coefficients, Datcom+ [28] is an extension of the Digital DATCOM program, which incorporates some tools to make it easier to handle the Digital Datcom program, and therefore was used for the computation of the aerodynamic coefficients. In the following, the coefficients composition is reported in the fashion outputted from Datcom+:

Coefficients not listed are not computed by DATCOM or neglected. All inputs are in degrees, while the derivatives are per radian. Tables are reported in the exact order of the above list.

C_D	$\underbrace{\text{CD\_Basic}}_{C_{D_0}+C_{D_\alpha}}$	$\underbrace{\text{CD\_Elevator}}_{C_{D_{\delta_e}}}$		
C_L	$\underbrace{\text{CL\_Basic}}_{C_{L_0}+C_{L_\alpha}}$	$\underbrace{\text{CL\_PitchRate}}_{C_{L_q}}$	$\underbrace{\text{CL\_Elevator}}_{C_{L_{\delta_e}}}$	
C_m	$\underbrace{\text{Cm\_Basic}}_{C_{m_0}+C_{m_\alpha}}$	$\underbrace{\text{Cm\_PitchRate}}_{C_{m_q}}$	$\underbrace{\text{Cm\_Elevator}}_{C_{m_{\delta_e}}}$	
C_Y	$\underbrace{\text{CY\_Beta}}_{C_{Y_\beta}}$	$\underbrace{\text{CY\_RollRate}}_{C_{Y_r}}$		
C_l	$\underbrace{\text{Cl\_Beta}}_{C_{l_\beta}}$	$\underbrace{\text{Cl\_RollRate}}_{C_{l_p}}$	$\underbrace{\text{Cl\_YawRate}}_{C_{l_r}}$	$\underbrace{\text{Cl\_Aileron}}_{C_{l_{\delta_a}}}$
C_n	$\underbrace{\text{Cn\_Beta}}_{C_{n_\beta}}$	$\underbrace{\text{Cn\_RollRate}}_{C_{n_p}}$	$\underbrace{\text{Cn\_YawRate}}_{C_{n_r}}$	$\underbrace{\text{Cn\_Aileron}}_{C_{n_{\delta_a}}}$

Table C.2: CD_Basic

α	CD_Basic	α	CD_Basic
-2	0.02612	8	0.07022
-1	0.02615	9	0.08072
0	0.02705	10	0.09255
1	0.02890	12	0.11490
1	0.03168	14	0.13790
3	0.03544	16	0.15980
4	0.04017	18	0.17540
5	0.04599	19	0.17630
6	0.05291	20	0.15880
7	0.06097	25	0.09811

Table C.3: CD_Elevator

$\alpha \downarrow$	δ_e								
	-20	-10	-5	-1	0	1	5	10	20
-2	0.0035	0.0017	0.0005	0.0001	0.0000	0.0000	0.0000	0.0007	0.0020
-1	0.0031	0.0014	0.0004	0.0000	0.0000	0.0000	0.0002	0.0009	0.0024
0	0.0027	0.0011	0.0003	0.0000	0.0000	0.0000	0.0003	0.0012	0.0028
1	0.0024	0.0009	0.0002	0.0000	0.0000	0.0000	0.0004	0.0014	0.0032
2	0.0020	0.0006	0.0000	0.0000	0.0000	0.0001	0.0006	0.0017	0.0036
3	0.0016	0.0004	-0.0001	-0.0001	0.0000	0.0001	0.0007	0.0019	0.0040
4	0.0012	0.0001	-0.0002	-0.0001	0.0000	0.0001	0.0008	0.0022	0.0044
5	0.0008	-0.0001	-0.0004	-0.0001	0.0000	0.0001	0.0009	0.0024	0.0047
6	0.0004	-0.0004	-0.0005	-0.0001	0.0000	0.0002	0.0011	0.0027	0.0051
7	0.0000	-0.0006	-0.0006	-0.0002	0.0000	0.0002	0.0012	0.0030	0.0055
8	-0.0004	-0.0009	-0.0007	-0.0002	0.0000	0.0002	0.0013	0.0032	0.0059
9	-0.0008	-0.0012	-0.0009	-0.0002	0.0000	0.0002	0.0014	0.0035	0.0063
10	-0.0012	-0.0014	-0.0010	-0.0002	0.0000	0.0003	0.0016	0.0037	0.0068
12	-0.0021	-0.0020	-0.0013	-0.0003	0.0000	0.0003	0.0019	0.0043	0.0077
14	-0.0032	-0.0027	-0.0016	-0.0004	0.0000	0.0004	0.0022	0.0050	0.0088
16	-0.0044	-0.0035	-0.0020	-0.0005	0.0000	0.0005	0.0026	0.0058	0.0099
18	-0.0058	-0.0044	-0.0025	-0.0005	0.0000	0.0006	0.0030	0.0067	0.0113
19	-0.0066	-0.0049	-0.0027	-0.0006	0.0000	0.0006	0.0033	0.0072	0.0122
20	-0.0076	-0.0056	-0.0031	-0.0007	0.0000	0.0007	0.0037	0.0079	0.0132
25	-0.0148	-0.0102	-0.0054	-0.0011	0.0000	0.0011	0.0060	0.0125	0.0203

Table C.4: CL_Basic

α	CL_Basic	α	CL_Basic
-2	-0.0381	8	0.9515
-1	0.0516	9	1.0590
0	0.1432	10	1.1680
1	0.2372	12	1.3480
2	0.3338	14	1.5050
3	0.4327	16	1.6320
4	0.5335	18	1.6990
5	0.6360	19	1.6810
6	0.7399	20	1.5210
7	0.8451	25	0.0707

Table C.5: CL_PitchRate

α	CL_PitchRate
all	7.9520

Table C.6: CL_Elevator

$\alpha \downarrow$	δ_e								
	-20	-10	-5	-1	0	1	5	10	20
all	-0.0924	-0.0598	-0.0299	-0.0060	0.0001	0.0060	0.0299	0.0598	0.0924

Table C.7: Cm_Basic

α	Cm_Basic	α	Cm_Basic
-2	0.0862	8	-0.2291
-1	0.0598	9	-0.2682
0	0.0337	10	-0.3098
1	0.0068	12	-0.3995
2	-0.0225	14	-0.5032
3	-0.0538	16	-0.6163
4	-0.0863	18	-0.6163
5	-0.1200	19	-0.6163
6	-0.1550	20	-0.6163
7	-0.1914	25	-0.6163

Table C.8: Cm_PitchRate

α	Cm_PitchRate
all	-16.5800

Table C.9: Cm_Elevator

$\alpha \downarrow$	δ_e								
	-20	-10	-5	-1	0	1	5	10	20
all	0.2636	0.1695	0.0847	0.0170	-0.0002	-0.0170	-0.0847	-0.1695	-0.2638

Table C.10: Cy_Beta

α	Cl_Beta
all	-0.3073

Table C.11: Cy_RollRate

α	Cy_RollRate	α	Cy_RollRate
-2	-0.0059	8	0.0101
-1	-0.0042	9	0.0116
0	-0.0026	10	0.0130
1	-0.0010	12	0.0163
2	0.0006	14	0.0200
3	0.0022	16	0.0239
4	0.0038	18	0.0268
5	0.0054	19	0.0512
6	0.0070	20	0.0434
7	0.0085	25	0.0806

Table C.12: Cl_Beta

α	Cl_Beta	α	Cl_Beta
-2	-0.0397	8	-0.0610
-1	-0.0415	9	-0.0634
0	-0.0434	10	-0.0659
1	-0.0454	12	-0.0694
2	-0.0474	14	-0.0718
3	-0.0495	16	-0.0729
4	-0.0517	18	-0.0715
5	-0.0540	19	-0.0688
6	-0.0563	20	-0.0609
7	-0.0586	25	0.0015

Table C.13: Cl_RollRate

α	Cl_RollRate	α	Cl_RollRate
-2	-0.4666	8	-0.5639
-1	-0.4802	9	-0.5676
0	-0.4934	10	-0.5278
1	-0.5059	12	-0.4056
2	-0.5175	14	-0.3176
3	-0.5280	16	-0.1707
4	-0.5374	18	0.1513
5	-0.5458	19	0.6986
6	-0.5530	20	1.2260
7	-0.5591	25	2.1430

Table C.14: Cl_YawRate

α	Cl_YawRate	α	Cl_YawRate
-2	0.0187	8	0.2125
-1	0.0361	9	0.2334
0	0.0541	10	0.2544
1	0.0726	12	0.2864
2	0.0916	14	0.3120
3	0.1109	16	0.3299
4	0.1307	18	0.3320
5	0.1508	19	0.3197
6	0.1711	20	0.2727
7	0.1917	25	-0.0692

Table C.15: Cl_Aileron

$\alpha \downarrow$	$\lvert\delta_a\rvert$								
	0	2	6	10	16	20	30	40	50
all	0.0000	0.0043	0.0130	0.0217	0.0348	0.0435	0.0636	0.0706	0.0740

Table C.16: Cn_Beta

α	Cn_Beta
all	0.0709

Table C.17: Cn_RollRate

α	Cn_RollRate	α	Cn_RollRate
-2	0.0027	8	-0.0750
-1	-0.0047	9	-0.0833
0	-0.0121	10	-0.0940
1	-0.0197	12	-0.1150
2	-0.0274	14	-0.1329
3	-0.0351	16	-0.1517
4	-0.0429	18	-0.1698
5	-0.0508	19	-0.2573
6	-0.0588	20	-0.2137
7	-0.0669	25	-0.0583

Table C.18: Cn_YawRate

α	Cn_YawRate	α	Cn_YawRate
-2	-0.0737	8	-0.0867
-1	-0.0744	9	-0.0888
0	-0.0752	10	-0.0910
1	-0.0762	12	-0.0949
2	-0.0772	14	-0.0982
3	-0.0784	16	-0.1007
4	-0.0798	18	-0.1010
5	-0.0813	19	-0.0994
6	-0.0829	20	-0.0937
7	-0.0847	25	-0.0769

Table C.19: Cn_Aileron

$\alpha \downarrow$	$\lvert\delta_a\rvert$								
	0	2	6	10	16	20	30	40	50
-2	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-1	0	0.0000	-0.0001	-0.0002	-0.0003	-0.0004	-0.0006	-0.0006	-0.0007
0	0	-0.0001	-0.0002	-0.0004	-0.0007	-0.0008	-0.0012	-0.0013	-0.0014
1	0	-0.0001	-0.0004	-0.0006	-0.0010	-0.0013	-0.0019	-0.0021	-0.0022
2	0	-0.0002	-0.0005	-0.0009	-0.0014	-0.0017	-0.0025	-0.0028	-0.0029
3	0	-0.0002	-0.0007	-0.0011	-0.0018	-0.0022	-0.0032	-0.0035	-0.0037
4	0	-0.0003	-0.0008	-0.0013	-0.0021	-0.0027	-0.0039	-0.0043	-0.0045
5	0	-0.0003	-0.0009	-0.0016	-0.0025	-0.0031	-0.0046	-0.0051	-0.0053
6	0	-0.0004	-0.0011	-0.0018	-0.0029	-0.0036	-0.0053	-0.0059	-0.0061
7	0	-0.0004	-0.0012	-0.0021	-0.0033	-0.0041	-0.0060	-0.0067	-0.0070
8	0	-0.0005	-0.0014	-0.0023	-0.0037	-0.0046	-0.0068	-0.0075	-0.0078
9	0	-0.0005	-0.0015	-0.0026	-0.0041	-0.0051	-0.0075	-0.0083	-0.0087
10	0	-0.0006	-0.0017	-0.0028	-0.0045	-0.0056	-0.0082	-0.0091	-0.0095
12	0	-0.0006	-0.0019	-0.0032	-0.0051	-0.0064	-0.0094	-0.0103	-0.0108
14	0	-0.0007	-0.0021	-0.0035	-0.0056	-0.0070	-0.0103	-0.0114	-0.0119
16	0	-0.0007	-0.0022	-0.0037	-0.0060	-0.0075	-0.0110	-0.0121	-0.0127
18	0	-0.0008	-0.0023	-0.0038	-0.0061	-0.0076	-0.0111	-0.0123	-0.0129
19	0	-0.0007	-0.0022	-0.0037	-0.0059	-0.0074	-0.0108	-0.0119	-0.0125
20	0	-0.0006	-0.0019	-0.0032	-0.0051	-0.0063	-0.0093	-0.0102	-0.0107
25	0	0.0001	0.0004	0.0006	0.0010	0.0012	0.0018	0.0020	0.0021

Appendix D

ArduPilot parameters

In this appendix, the control parameters for the low-level autopilot structure, after the on-field tuning, are reported. Some of them were described in Chapter 3; for the meaning of the other implemented parameters, see [29].

Roll control loop

Parameter	Symbol	Value
RLL2SRV_TCONST	τ_ϕ	0.45
RLL2SRV_P	K_{P_ϕ}	0.7
RLL2SRV_I	K_{I_ϕ}	0.1
RLL2SRV_D	K_{D_ϕ}	0.01
RLL2SRV_FF	K_{FF_ϕ}	0
RLL2SRV_RMAX	-	75
RLL2SRV_IMAX	-	30

Pitch control loop

Parameter	Symbol	Value
PTCH2SRV_TCONST	τ_θ	0.45
PTCH2SRV_P	K_{P_θ}	1.055
PTCH2SRV_I	K_{I_θ}	0.15
PTCH2SRV_D	K_{D_θ}	0.08
PTCH2SRV_IMAX	-	30
PTCH2SRV_RMAX_UP	-	75
PTCH2SRV_RMAX_DN	-	75
PTCH2SRV_RLL	-	1

Side-slip control loop

Parameter	Symbol	Value
YAW2SRV_SLIP	-	0
YAW2SRV_INT	-	0
YAW2SRV_DAMP	-	0
YAW2SRV_RLL	$K_{r_{coord}}$	0
KFF_RDDRMIX	$K_{r_{mix}}$	0.5
YAW2SRV_IMAX	-	15

TECS

Parameter	Symbol	Value
TECS_CLMB_MAX	-	5
TECS_SPD_OMEGA	-	2
TECS_SINK_MIN	-	2
TECS_RLL2THR	K_{T_ϕ}	10
TECS_TIME_CONST	-	5
TECS_SPDWEIGHT	ϵ_{ke}	1
TECS_THR_DAMP	$K_{D_{thr}}$	0.5
TECS_PTCH_DAMP	K_{D_Θ}	0
TECS_INTEG_GAIN	$K_{I_{thr}}$	0.1
TECS_SINK_MAX	-	5
TECS_VERT_ACC	-	7
TECS_PITCH_MAX	-	20
TECS_HGT_OMEGA	-	3
TECS_PITCH_MIN	-	-20

Course control loop

Parameter	Symbol	Value
NAV_ROLL_P	K_{P_χ}	0.7