

POLITECNICO DI MILANO
School of Industrial and Information Engineering
Master of Science in Aeronautical Engineering



Air-to-air automatic landing for multirotor UAVs

Advisor: Prof. Marco LOVERA
Co-Advisors: Ph.D. Davide INVERNIZZI
Eng. Mattia GIURATO

Master Thesis by:
Adriano MARINI COSSETTI Matr. 863099
Pietro GIURI Matr. 853735

Academic Year 2017–2018

Acknowledgments

Alla fine di questa appassionante sfida vogliamo dedicare poche parole alle persone che ci hanno aiutato a renderla realtà. In primis, ringraziamo il Professor Marco Lovera per averci “*arruolato*” ed averci proposto una tesi tanto impegnativa quanto stimolante. Egli ci ha dimostrato estrema competenza e disponibilità, nonché incondizionata fiducia.

Il secondo immenso grazie va a Davide Invernizzi, per averci accompagnato sapientemente dall’inizio alla fine, dandoci preziosi consigli ed essendo sempre presente. Ed ancora, a Mattia Giurato, per averci guidato nell’oscuro mondo del laboratorio, anche quando non si vedeva nessuna luce in fondo al tunnel.

Inoltre, ringraziamo di cuore Simone Panza per essersi reso disponibile ogni giorno a risolvere problemi informatici più grandi di noi. Non dimenticheremo mai il tuo rassicurante: «*oplà*». Un altro compagno di avventure da ringraziare assolutamente è Paolo Gattazzo, non solo per l’aiuto costante nel risolvere problemi pratici e nell’eseguire i test, ma anche per l’allegria che ha saputo portare ogni giorno, un dono che poche persone possiedono. Inoltre, un pensiero va a tutti i ragazzi del laboratorio che hanno condiviso gioie e dolori, lamenti ed esultanze, in questi otto interminabili mesi.

Infine ringraziamo i nostri compagni di corso, che hanno reso questa laurea magistrale e questo periodo indimenticabili: Simone, che ha condiviso con noi l’intera esperienza di tesi rendendola “*straordinaria*”, e poi Alberto, Federico, Nicola e Stefano. Raramente si incontrano persone speciali come voi, e senza di voi questo percorso sarebbe stato solo un freddo corso di studi. Speriamo di ritrovarci presto tutti insieme per condividere un’altra delle nostre sobrie cene, indagando le specialità culinarie delle nostre magnifiche regioni italiane e chiacchierando, mentre sorseggiamo del buon vino.

Grazie a tutti.

Abstract

Nowadays, the Unmanned Aerial Vehicles (UAVs) are continuing to enlarge their market share and research activities about them are growing exponentially. In particular, the interaction between two or more vehicles during flight, *e.g.* formation flight and refueling, are getting more and more attention. Dealing with intelligence, surveillance, and reconnaissance missions the problem of air-to-air refueling can arise when undertaking long range flights. In the military field, the Air-to-Air Automatic Refuelling (AAAR) involving fixed-wing drones is object of large studies and research activities. Also small/medium UAVs suffer from low autonomy problems, since the overwhelming majority of them has an electric propulsion system. Another possibility to extend the range of UAVs missions could be to have a carrier drone, reasonably a fixed-wing one, with several lightweight multirotors aboard, which can take-off from and land on it. The work conducted within this thesis is focused on the implementation of a guidance law to obtain automatic air-to-air landing of a small quadcopter on a bigger hexacopter, both developed at Aerospace Systems and Control Laboratory (ASCL) in Politecnico di Milano. The purpose of the thesis is, first of all, to analyze and to simulate the guidance law in order to investigate its feasibility, then the algorithm is implemented and experimentally validated. Initially, a simulator has been realized, with a quadrotor model created using an object-oriented multi-body software. This includes both nonlinear dynamics and aerodynamics. Afterward, a number of possible algorithms for the landing trajectory generation have been studied. The feasible ones have been implemented and simulated with the previously built simulator, extended to two UAVs. Eventually, the guidance laws are validated through an experimental activity.

Sommario

Oggigiorno, i velivoli a pilotaggio remoto (UAV) continuano ad ampliare la loro quota di mercato e le attività di ricerca su di loro stanno crescendo esponenzialmente. In particolare, l'interazione tra due o più veicoli durante il volo, *e.g.* volo di formazione e rifornimento, sta ottenendo sempre più attenzione. Occupandosi delle missioni di intelligence, sorveglianza e ricognizione, il problema del rifornimento di carburante in volo può sorgere quando si intraprendono voli a lungo raggio. In campo militare, il rifornimento automatico in volo (AAAR) che coinvolge droni ad ala fissa è oggetto di ampi studi e attività di ricerca. Anche gli UAV di piccole e medie dimensioni soffrono di problemi di bassa autonomia, poiché la stragrande maggioranza di loro ha un sistema di propulsione elettrica. Un'altra possibilità di estendere l'autonomia di questa classe di UAV potrebbe essere quella di avere un drone "cargo", ragionevolmente uno ad ala fissa, con diversi multirotori leggeri a bordo, che possono decollare da e atterrare su di esso. Il lavoro condotto all'interno di questa tesi è incentrato sull'implementazione di una legge di guida per ottenere l'atterraggio automatico in volo di un piccolo quadricottero su un esacottero di dimensioni maggiori, entrambi sviluppati presso l' Aerospace Systems and Control Laboratory (ASCL) del Politecnico di Milano. Lo scopo della tesi è, prima di tutto, di analizzare e simulare la legge di guida per indagarne la fattibilità e, in seguito, di implementarla e validarla sperimentalmente. Inizialmente è stato realizzato un simulatore con il modello di un quadricottero, creato utilizzando un software di modellazione multi-corpo orientato agli oggetti. Quest'ultimo include sia la dinamica non lineare che l'aerodinamica del quadricottero. Successivamente, sono stati studiati una serie di possibili algoritmi per la generazione della traiettoria di atterraggio. Quelli fattibili sono stati implementati e simulati con il simulatore precedentemente sviluppato, esteso a due multirotori. Alla fine, le leggi di guida sono state validate attraverso un'attività sperimentale.

Contents

Acknowledgments	I
Abstract	III
Sommario	V
List of figures	XI
List of tables	XV
1 Introduction	1
2 Modelling and simulation of multirotor UAVs	5
2.1 Reference axes	6
2.1.1 Earth axes	6
2.1.2 Body axes	6
2.1.3 Rotor fixed axes	7
2.1.4 Rotor wind axes	7
2.2 Rotation formalism	7
2.2.1 Euler angles	7
2.2.2 The time derivative of Euler angles	10
2.2.3 Quaternions	11
2.3 Flight dynamics	12
2.3.1 Kinematics	12
2.3.2 Equations of motion - linear motion	13
2.3.3 Equations of motion - angular motion	14
2.3.4 Overall states of a rigid aircraft	15
2.3.5 Control forces and moments	15

2.4	Rotor aerodynamics	18
2.4.1	Rotor blades motion	19
2.4.2	Momentum theory	24
2.4.3	Blade element theory - BET	29
2.4.4	Blade element momentum theory	32
2.4.5	Dynamic inflow model (Pitt and Peters)	33
2.5	Aerodynamic modeling of a quadrotor	34
2.5.1	The multi-body model	35
2.5.2	Thrust and inflow	38
2.5.3	Torque and drag forces	43
2.5.4	Implementation	49
2.6	Simulation	54
2.6.1	Dymola-Simulink co-simulation	54
2.6.2	Results	56
2.6.3	Conclusions	69
2.7	Experimental results	70
2.7.1	Quadrotor prototype	71
2.7.2	Model comparison	72
2.7.3	Conclusions	73
3	Problem Formulation	77
3.1	Problem description	78
3.2	State of the art	78
3.3	Mathematical formulation	80
3.4	Suitable algorithms description	84
3.4.1	Three-states bang-bang algorithm	84
3.4.2	Quasi Time-Optimal algorithm	85
4	Digital implementation	87
4.1	Error monitoring and safety procedures	88
4.2	Three-states bang-bang	89
4.3	Quasi time-optimal	92
5	Simulation results	97
5.1	Simulation setup	98
5.2	Three-states bang-bang	99

5.3	Quasi time-optimal	103
5.3.1	Sensitivity analysis	105
5.4	Conclusions	118
6	Experimental activity	121
6.1	System architecture	122
6.1.1	Hardware	122
6.1.2	Software	126
6.2	Landing pad design	128
6.3	Experimental results	130
6.3.1	Three-states bang-bang	130
6.3.2	Quasi time-optimal	135
6.3.3	Velocity estimate issue	137
6.3.4	Final flight test	141
6.4	Conclusions	146
7	Conclusions	147

List of Figures

2.1	Quadcopter configuration.	16
2.2	Rotor reference planes:tip-path plane (TPP), no-feathering plane (NFP), hub plane (HP), and control plane (CP) (picture from [1]).	20
2.3	Aerodynamics of the rotor blade section (picture from [1]).	21
2.4	Air velocity relative to the blade in forward flight [1].	22
2.5	Flow model for momentum theory analysis of rotor in forward flight (picture from [1]).	24
2.6	Momentum theory results for the induced velocity in vertical flight (picture from [1]).	28
2.7	Quadcopter multi-body model overview.	37
2.8	Drag coefficient estimation.	48
2.9	Arm model overview.	53
2.10	Quadrotor multi-body model.	53
2.11	Quadrotor simulator.	56
2.12	Quadcopter subsystem.	57
2.13	Infinity trajectory.	58
2.14	Trajectory set-point.	59
2.15	Quadrotor position and velocity error (BET).	60
2.16	Quadrotor attitude and angular rates error (BET).	60
2.17	Quadrotor thrust coefficient and induced inflow ratio (BET).	61
2.18	Quadrotor H-force and torque dimensionless coefficients (BET).	62
2.19	Quadrotor H-forces (BET).	63
2.20	Quadrotor throttle percentage (BET).	64
2.21	Quadrotor position and velocity error (lumped parameter).	65
2.22	Quadrotor attitude and angular rates error (lumped parameter).	65

2.23	Quadrotor thrust coefficient and induced inflow ratio (lumped parameter)	66
2.24	Quadrotor H-force and torque dimensionless coefficients (lumped parameter).	67
2.25	Quadrotor H-forces (lumped parameter).	68
2.26	Quadrotor throttle percentage (lumped parameter).	69
2.27	Quadrotor prototype.	72
2.28	Models comparison: infinity-shape maneuver.	74
2.29	Models comparison: Step maneuver.	75
5.1	Follower, target and desired altitude (bang-bang), ideal case. . . .	100
5.2	Control variables (bang-bang), ideal case.	102
5.3	Follower, target and desired altitude (bang-bang), non-ideal case. . . .	103
5.4	Control variables (bang-bang), non-ideal case.	104
5.5	Null acceleration command iso-lines (QTO).	106
5.6	Position and velocity errors contribution to acceleration command (QTO).	107
5.7	Errors evolution with different K_p (QTO).	110
5.8	Landing trajectories (QTO).	111
5.9	Follower, target and desired altitude (QTO), ideal case.	113
5.10	Follower and target velocity (QTO), ideal case.	113
5.11	Landing trajectory error (QTO), ideal case.	114
5.12	Acceleration command (QTO), ideal case.	114
5.13	Follower, target and desired altitude (QTO), non-ideal case. . . .	115
5.14	Follower and target velocity (QTO), non-ideal case.	116
5.15	Landing trajectory error (QTO), non-ideal case.	116
5.16	Acceleration command (QTO), non-ideal case.	117
6.1	Photo of the cage with Optitrack cameras on top	123
6.2	UAVs adopted in the flight tests.	125
6.3	Landing pad CAD assembly.	129
6.4	Follower, target and desired altitude (bang-bang), flight test. . . .	131
6.5	Follower and target velocity (bang-bang), flight test.	132
6.6	Three-states bang-bang control variables.	133
6.7	Three-states bang-bang reference position, z_r	134

6.8	Three-states bang-bang reference velocity, \dot{z}_r	134
6.9	Error monitoring procedure (bang-bang), flight test.	135
6.10	Synchronization failure (bang-bang), flight test.	136
6.11	Acceleration command and u_p, u_v (QTO), unsuccessful test.	138
6.12	UAVs position and velocity (QTO), unsuccessful test.	139
6.13	IMU measurements and velocity estimate of the on board filter.	140
6.14	Follower, target and desired altitude (QTO), flight test.	141
6.15	Error monitoring procedure (QTO), flight test.	142
6.16	Acceleration command (QTO), flight test.	142
6.17	Landing trajectory error (QTO), flight test.	143
6.18	Follower and target velocity (QTO), flight test.	143
6.19	Air-to-air landing flight test.	145

List of Tables

2.1	Quadrotor mass and geometric properties.	38
2.2	Arm parameters.	51
2.3	Quadrotor model main parameters.	72
5.1	Standard deviation of measurement noise.	99
5.2	Three-states bang-bang parameters.	99
5.3	Parameters of quasi time-optimal algorithm.	112
6.1	Ant-1 model main parameters.	125
6.2	Hexa model main parameters.	126
6.3	Standard deviation of velocity estimate.	136
6.4	QTO new parameters.	137

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV) is an aircraft without a pilot aboard, which is able to fly autonomously or could be piloted from the ground. Usually called drones, in recent years this type of vehicles has met with great interest both in civil and military fields thanks to their wide range of applications, including precision agriculture, photography, policing and surveillance, search and rescue, entertainment, product delivery, aerial inspection and many others.

When referring to an UAV, generally called drone, one usually refers to the category of multi-rotor Vertical Take-Off and Landing (VTOL) vehicles provided with four, six, or eight motors, of small/medium size and remote controlled. Their versatility in technical operations have pushed the commercial and research communities towards new challenges. New research activities involve the possibility to remotely command many drones together, following the same path or performing many task simultaneously. Among the main sub-areas covering the cooperative control problem of UAVs, formation flight has attracted great interest and has been widely investigated. Besides multi-rotors formation flight, in the military field, the Air-to-Air Automatic Refueling (AAAR) involving fixed-wing drones is object of large studies and research activities. Proceeding further, one may argue that the possibility to couple formation flight with some air-to-air crucial operations has not found attention in the multi-rotors field yet.

In this thesis the focus is on the design of guidance laws aimed at providing a small multicopter with a reference descent trajectory, ending up to an air-to-air landing onto a larger one. This kind of maneuver is as risky as dangerous. The propellers wake of a multicopter generates an unsteady flow field around it.

When two UAVs fly close, they perturb each other. For what regards the air-to-air landing operation, one drone is always above the other, which constantly flies in a perturbed regime. The closer are the vehicles, the stronger will be the aerodynamic disturbances affecting the one below. It is straightforward that the aerodynamic forces developing on the multirotors vary continuously in this flight condition. This thesis also proposes a novel approach to build up the dynamic and aerodynamic model of a quadcopter, representing a further progress in the modeling procedure of multirotor UAVs. Other obvious considerations regarding the dimensions and weights of the UAVs can be taken into account, *e.g.* it is necessary that the carrier drone must be heavier and larger than the landing one, in order to stand its weight once the touch down occurs. Even though the control problem has not been studied, the controllers embedded on-board both multirotors must be capable of rejecting at least part of the high aerodynamic disturbances. The hazardous nature of the maneuver makes the design of the trajectory generation module hard and complex, as the problem is fully three-dimensional. Thanks to a decoupled approach it is possible to overcome this issue, by verifying the drones have been synchronized in the same horizontal position for a certain time at first, and then sending the reference landing path to the smaller one, provided that it respects the synchronization constraint during the landing too.

In this work, two innovative control laws are proposed for the landing trajectory generation purpose. The first one, presented in [2], is based on concept of bang-bang acceleration commands, while the second is quasi time-optimal.

The thesis is organized as follows:

- Chapter 2 deals with modeling and simulation of multirotor UAV flight; specifically the aerodynamic dependencies of rotor forces are included in the model of a quadrotor, making use of an object-oriented multi-body approach. The experimental evidence of adopting aerodynamic modeling to estimate a multirotor tracking capability is shown at the end of the chapter. In the chapter a dynamic and aerodynamic model for quadrotors is shown in details, and at the end of the chapter the experimental evidence that it reduces modeling errors during high performance aggressive maneuvers is proved.
- In Chapter 3 the actual state of the art addressing the multirotor landing

topic is presented and the problem is mathematically stated. The goal and the constraints of the landing problem are exposed, and at the end the control algorithms are introduced.

- The implementation of the trajectory generation module is discussed in detail in Chapter 4. The procedure to verify the synchronization constraint is described at first, while in the last part of the chapter the mathematical formulation of the control algorithms is presented.
- The landing maneuver has been simulated and the results are shown in Chapter 5, for each control law adopted to generate the reference landing path. Two sets of simulations are performed to quantify the difference between the ideal simulation environment and the possible execution of the real flight test, including the working limits of a ground station in the simulation setup.
- In Chapter 6, after a brief description of the laboratory environment, the flight test results are illustrated. Finally, the trajectory generation module has been tested, performing the air-to-air landing maneuver using two multirotor UAVs.

Chapter 2

Modelling and simulation of multirotor UAVs

In this first chapter a novel approach is presented to include rotor aerodynamics in the physical model of a quadrotor helicopter. Initially, the dynamic model of a UAV will be described, as well as the adopted conventions and formalism in order to avoid ambiguities. The classical rotor aerodynamic theories are introduced to address the thrust aerodynamic dependency. Then, the multi-body model of the quadcopter is depicted in details. Specifically, two models are proposed, which differ only in the adopted expression of drag force. After the implementation of the quadcopter model in Dymola [3], and of the control loop in Simulink [4], some co-simulations with a Dymola-Simulink interface are performed to evaluate the aerodynamic influence on the overall performance.

In the final, simulation results of the proposed models are compared with real flight data, to appreciate the improvement in predicting quadrotor behavior when aerodynamics is considered in the dynamical model.

2.1 Reference axes

In order to analyze correctly the unsteady motion of a flying vehicle it is necessary to define first an inertial reference system, because the basic equations for the dynamic model of a mechanical system directly apply only in an inertial reference. Secondly, another reference frame centered in the center of gravity (CG) of the flying vehicle is needed. Such reference was first introduced by Leonhard Euler (Basel 1707 - St. Petersburg 1783). More than these, when dealing with quadrotors or helicopters, suitable rotor reference systems must be used to properly identify the rotor aerodynamic environment.

2.1.1 Earth axes

Due to the fact that quadrotor UAVs fly indoor or close to the ground, the flat and non-rotating Earth is assumed. The Earth fixed frame is defined as $\mathcal{F}_E = \{O_E, e_{1E}, e_{2E}, e_{3E}\}$, where the first element is the origin point while the others are three unit vectors. The origin O_E is arbitrary: it could be the intersection of the equator, the prime meridian and mean sea level. The standard convention used in aeronautics has e_{1E} pointing North, e_{2E} pointing to the East and e_{3E} aligned with the direction of gravity, pointing downward. This type of reference frame is often referred to as NED, clearly meaning North-East-Down. These three axes are mutually perpendicular and, when referred to in the order N, E, D , form a right-handed coordinate system.

2.1.2 Body axes

Analysis of the equations of motion of an aircraft is not easy considering a reference attached to the Earth. Usually a moving reference system centered in the CG (barycentric) can be profitably used.

For this study, the body reference frame $\mathcal{F}_B = \{O_B, e_{1B}, e_{2B}, e_{3B}\}$ refers to a right-handed system of coordinates, rigidly attached to the center of gravity of the UAV, *i.e.* $O_B \equiv CG$, and changing orientation with it. The unit vector e_{1B} lies in the plane of symmetry of the vehicle and points forward, e_{2B} points to the right wing, normal to the plane of symmetry, and e_{3B} points downward the vehicle, mapping the three dimensional space around the aircraft a little bit differently

from the formal concept of NED.

2.1.3 Rotor fixed axes

The rotor fixed frame is defined as $\mathcal{F}_{R_j} = \{O_R, e_{1_R}, e_{2_R}, e_{3_R}\}$, where O_R coincides with the center of the rotor. The unit vector e_{1_R} lies in the rotor disk plane, it is aligned with the respective quadrotor arm and points outward; while e_{3_R} is aligned with motor-rotor shaft and points downward. Then, e_{2_R} completes the right-hand rule. Obviously, the subscript j indicates the j -th rotor of the UAV.

2.1.4 Rotor wind axes

Instead, the rotor wind frame is defined as $\mathcal{F}_{RW_j} = \{O_{RW}, e_{1_{RW}}, e_{2_{RW}}, e_{3_{RW}}\}$, where O_{RW} coincides again with the center of the rotor. The unit vector $e_{1_{RW}}$ is aligned with the airspeed seen by the rotor and lies in the rotor disk plane. $e_{3_{RW}}$ is orthogonal to the rotor disk plane and points downward. Lastly, $e_{2_{RW}}$ completes the right-hand rule. As before, the subscript j indicates the j -th rotor of the UAV.

2.2 Rotation formalism

2.2.1 Euler angles

One of the methods that allow to switch from a Cartesian coordinate system to another one is based on the definition of three independent parameters, able to describe the relative orientation of the two sets of reference axes. The Euler Angles (ϕ, θ, ψ) are three independent angular quantities able to do that. Assume that there are two different reference frames, with the same origin but different axes. The orientation of the axes is obtained by a sequence of three rotations so as to bring the two sets to coincide and overlap. The sequence cannot be exchanged, hence the Euler angles define the transformation of the components of a generic vector between two sets of axes. (For clarifications, see [5]).

Adopting the to-from notation, a rotation matrix from system A to system B might be named R_A^B . Thus, a vector v_A in system A can be resolved to system B,

that is v_B through the matrix operation:

$$v_B = R_A^B v_A \quad (2.1)$$

It is easier to understand how these rotation matrices work by considering rotations about one axis at a time. Rotation about the x -axis does not change the component of the vector directed along the same axis, but it does change the y and z components. The rotation matrix that does this transformation is

$$R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}. \quad (2.2)$$

In a similar fashion, the following matrices perform rotations about the y -axis and the z -axis, respectively:

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \quad (2.3)$$

$$R_Z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.4)$$

One measures a rotation angle about (around) the axis of its rotation. The user fixes the location of 0° arbitrarily, but once it is fixed it must not vary. The rotational matrices are orthonormal, which means the columns in the matrices, if considered as vectors, have unit magnitude (normal), and are mutually orthogonal. For this reason:

$$\begin{aligned} R_X^{-1}(\phi) &= R_X^T(\phi), \\ R_Y^{-1}(\theta) &= R_Y^T(\theta), \\ R_Z^{-1}(\psi) &= R_Z^T(\psi). \end{aligned} \quad (2.5)$$

Any number of rotations can be put together in any order. The resulting cascade

can be reduced to a rotation about just three axes. The matrix that performs this specific action is called an Euler rotation matrix and has the following definition:

$$T_E^B(\phi, \theta, \psi) = R_X(\phi)R_Y(\theta)R_Z(\psi) \quad (2.6)$$

The subscripts B and E stand for Body and Earth, respectively. The matrix T_E^B resolves an Earth-based vector to body axes. Expanding the matrix products, it results:

$$T_E^B(\phi, \theta, \psi) = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi - S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix}, \quad (2.7)$$

where a shorthand notation, which is $C_a = \cos(a)$ and $S_a = \sin(a)$, has been adopted.

In the following, the transformation related to velocity or acceleration vectors is mathematically explained:

$$r_e = \begin{bmatrix} N \\ E \\ D \end{bmatrix}, \quad (2.8)$$

$$v_e = \begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \dot{r}_e, \quad (2.9)$$

$$v_b = T_E^B(\phi, \theta, \psi)v_e = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad (2.10)$$

where r_e is the position vector of the aircraft center of gravity in inertial (Earth) axes, v_e is the velocity of the aircraft with respect to the Earth, and v_b is the inertial linear velocity of the aircraft, resolved to body axes.

We can also define the vector of angular position of the aircraft body axes with respect to the Earth, resolved to the Earth where the elements are roll, pitch and

yaw angles.

$$\alpha_e = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}. \quad (2.11)$$

2.2.2 The time derivative of Euler angles

The attitude of the aircraft changes with time when an aircraft maneuvers. The Euler rates are a function of the Euler angles and body-axis angular rates. Euler angles rotate Euler rates into body-axis angular rates. However, the transformation is different from the transformation for linear rates as discussed earlier. The process is complicated by the fact that the Euler angles themselves are involved in the transformation from Euler to body-axis rates. The process proceeds as described below. Define the Euler rates as

$$\dot{\alpha}_e = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \quad (2.12)$$

and the components of the body angular velocity as

$$\omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (2.13)$$

In order to get the angular rates, as illustrated in [5], Euler rates must be considered individually and resolved to intermediate axes, and then finally to the body axes. Hence an explicit formulation can be obtained and rewritten in compact form:

$$\omega_b = T^{-1}\dot{\alpha}_e, \quad (2.14)$$

where

$$T^{-1} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta) \sin(\phi) \\ 0 & \sin(\phi) & \cos(\theta) \cos(\phi) \end{bmatrix}$$

$$\rightarrow T = \begin{bmatrix} 1 & \sin(\phi) \tan(\theta) & \cos(\phi) \tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{bmatrix}.$$

Note that T does not depend on ψ and it is singular for $\theta = \pm 90^\circ$.

2.2.3 Quaternions

For what concerns the Earth and Body references frame adopted in this study, it is clear that ϕ is the roll angle, θ is the pitch angle and ψ is the yaw angle. The three Euler equations have singularities at $\theta = \pm 90^\circ$ (gimbal lock). Furthermore Euler angles can be integrated up to values outside the normal $\pm 90^\circ$ range of pitch, $\pm 180^\circ$ range of roll and yaw angles. It is clear that this causes a problem in the unique identification of aircraft attitude but it is observable that equations are linear in (p, q, r) but non-linear in terms of Euler angles. A clever solution is the use of quaternions. In [5], a full and complete description of quaternions properties is presented, and how to suitably use them in all mathematical processes involving a representation of rotating vectors in many different system of coordinates.

In the following, the relationship between quaternion

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \quad (2.15)$$

and Euler angles is depicted:

$$\begin{aligned} q_0 &= \pm (\cos(\phi/2) \cos(\theta/2) \cos(\psi/2) + \sin(\phi/2) \sin(\theta/2) \sin(\psi/2)), \\ q_1 &= \pm (\sin(\phi/2) \cos(\theta/2) \cos(\psi/2) - \cos(\phi/2) \sin(\theta/2) \sin(\psi/2)), \\ q_2 &= \pm (\cos(\phi/2) \sin(\theta/2) \cos(\psi/2) + \sin(\phi/2) \cos(\theta/2) \sin(\psi/2)), \\ q_3 &= \pm (\cos(\phi/2) \cos(\theta/2) \sin(\psi/2) - \sin(\phi/2) \sin(\theta/2) \cos(\psi/2)). \end{aligned} \quad (2.16)$$

The rates of change of the quaternion parameters in terms of the three-body-axis rotational rates (p, q, r) are given by:

$$\dot{q} = -\frac{1}{2} \begin{bmatrix} 0 & p & q & r \\ -p & 0 & -r & q \\ -q & r & 0 & -p \\ -r & -q & p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}. \quad (2.17)$$

(2.18)

2.3 Flight dynamics

2.3.1 Kinematics

Let V and a represent the linear velocity and acceleration of the CG of the aircraft respectively. Let the rotational speed of the aircraft indicated as ω . The dynamic equilibrium of the aircraft can be expressed by two vectorial equations:

$$\begin{aligned} F^{(a)} + F^{(v)} + F^{(i)} &= 0, \\ M^{(a)} + M^{(v)} + M^{(i)} &= 0, \end{aligned}$$

where the applied reaction and inertial forces sum up to zero.

Obviously, for an aircraft in flight, reaction forces are null, and we can include in the applied forces both aerodynamic forces and gravitational ones. Inertial forces in an inertial frame can be defined as:

$$\begin{aligned} F^{(i)} &= -\frac{dQ}{dt}, \\ M^{(i)} &= -\frac{dK}{dt} - v_P \times Q, \end{aligned} \quad (2.19)$$

where

$$Q = mV$$

is the momentum and

$$K = J_n \omega$$

is the angular momentum. m and J_n respectively refer to mass and inertia tensor and v_P is the linear speed of point P , used as a reference point to compute all

moments.

In case $P \equiv CG$ then $v_P \times Q = 0$, hence equations (2.19) reduce to:

$$\begin{aligned} F^{(a)} &= \frac{dQ}{dt}, \\ M^{(a)} &= \frac{dK}{dt}. \end{aligned} \quad (2.20)$$

In order to be able to write the equations of motion considering and including each term is convenient to express the dynamics with respect to a suitable reference frame, where quantities are easily readable or known. The following expressions can be adopted:

$$V = v_b = u e_{1B} + v e_{2B} + w e_{3B}, \quad (2.21)$$

$$Q = m(u e_{1B} + v e_{2B} + w e_{3B}), \quad (2.22)$$

$$\omega = \omega_b = p e_{1B} + q e_{2B} + r e_{3B}, \quad (2.23)$$

$$K = K_x e_{1B} + K_y e_{2B} + K_z e_{3B}. \quad (2.24)$$

By using Poisson's formulas, it is possible to express equilibrium equations also in a non-inertial reference system, see [5]. In this way it is possible to write equations of motion in body frame:

$$F^{(a)} = \frac{dQ}{dt} = \dot{Q} + \omega_b \times Q, \quad (2.25)$$

$$M^{(a)} = \frac{dK}{dt} = \dot{K} + \omega_b \times K. \quad (2.26)$$

2.3.2 Equations of motion - linear motion

Expanding 2.25 using 2.22 and 2.23 quantities, it is possible to obtain the dynamic equations of motion for the linear motion of the aircraft, expressed in its moving body reference frame. Assumption of constant mass for a moving aircraft is made:

$$\begin{aligned} F_x &= m(\dot{u} + qw - rv), \\ F_y &= m(\dot{v} + ru - pw), \\ F_z &= m(\dot{w} + pv - qu), \end{aligned} \quad (2.27)$$

where F_x, F_y and F_z are the non-inertial active forces applied to the aircraft body axes, defined by e_{1B} , e_{2B} and e_{3B} , respectively. In vectorial notation:

$$m\dot{v}_b + \omega_b \times mv_b = F_{ext} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}. \quad (2.28)$$

2.3.3 Equations of motion - angular motion

The same procedure shown previously applies also for (2.26) but some prior definitions have to be fixed. Keep in mind that angular momentum is function of inertia tensor of the aircraft and angular rates. The definition of inertia tensor J can be introduced

$$J_n = \begin{bmatrix} J_{xx} & -J_{xy} & -J_{xz} \\ -J_{xy} & J_{yy} & -J_{yz} \\ -J_{xz} & -J_{yz} & J_{zz} \end{bmatrix}, \quad (2.29)$$

where the terms are defined as follows:

$$\begin{aligned} J_{xx} &= \int_m (y^2 + z^2) dm, & J_{yy} &= \int_m (x^2 + z^2) dm, & J_{zz} &= \int_m (x^2 + y^2) dm, \\ J_{xy} &= \int_m (xy) dm, & J_{xz} &= \int_m (xz) dm, & J_{yz} &= \int_m (yz) dm. \end{aligned}$$

For our purposes, usually small scale UAVs show a symmetry both geometrical and in the mass distribution with respect to the same plane. If the body frame is coincident with the symmetry axes of the aircraft body, it follows that (2.29) will be diagonal because

$$J_{xy} = J_{xz} = J_{yz} = 0.$$

Expanding (2.26) using 2.23 and 2.24, the explicit dynamic equations for angular motion are obtained:

$$L = J_{xx}\dot{p} + qr(J_{zz} - J_{yy}), \quad (2.30)$$

$$M = J_{yy}\dot{q} + pr(J_{xx} - J_{zz}), \quad (2.31)$$

$$N = J_{zz}\dot{r} + pq(J_{yy} - J_{xx}), \quad (2.32)$$

where L , M and N are the non-inertial moments applied on the body-axes, defined by e_{1_B} , e_{2_B} and e_{3_B} , respectively. In vectorial notation:

$$J_n \dot{\omega}_b + \omega_b \times J_n \omega_b = M_{ext} = \begin{bmatrix} L \\ M \\ N \end{bmatrix}. \quad (2.33)$$

2.3.4 Overall states of a rigid aircraft

Nowadays, it is common sense to collect the dynamic variable of interest and their time derivatives too, in a vector called *state vector*, but not including the highest derivative. This is the so-called *state-space formulation*. The ordering of the states in the state vector is not important from a mathematical standpoint, although certain computational economies are realized if three-element sub-vectors remained grouped as just defined. Once the order of the states in the state vector has been selected, it must be preserved.

$$x = \begin{bmatrix} r_e \\ v_b \\ \omega_b \\ \alpha_e \end{bmatrix} = \begin{bmatrix} N & E & D & u & v & w & p & q & r & \phi & \theta & \psi \end{bmatrix}^T \quad (2.34)$$

Therefore, the derivative of the state vector is defined:

$$\begin{aligned} \dot{x} &= \begin{bmatrix} \dot{N} & \dot{E} & \dot{D} & \dot{u} & \dot{v} & \dot{w} & \dot{p} & \dot{q} & \dot{r} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \\ &= \begin{bmatrix} T_B^E(\phi, \theta, \psi) v_b \\ -\omega_b \times v_b + F_{ext}/m \\ J_n^{-1}(-\omega_b \times J_n \omega_b + M_{ext}) \\ T(\phi, \theta) \omega_b \end{bmatrix}. \end{aligned} \quad (2.35)$$

2.3.5 Control forces and moments

After the introduction of rigid body flight dynamics, the configuration of the UAV must be taken into account, to state which forces and moments are applied on the aircraft and be able to describe them in body axes. In Figure 2.1 the chosen configuration (X-configuration), the label of each propeller, and its rotation

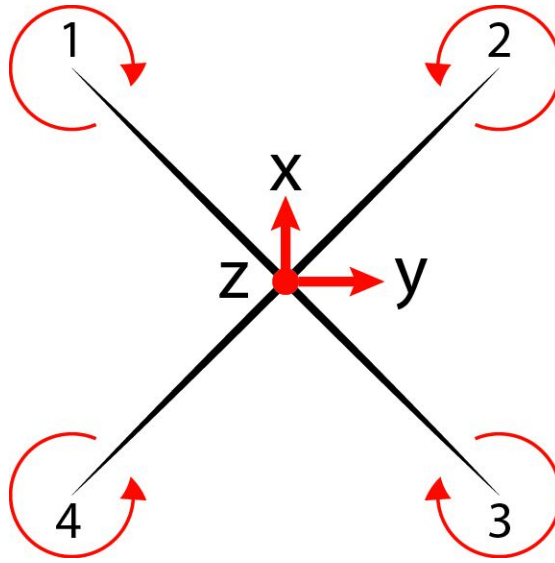


Figure 2.1: Quadcopter configuration.

direction are shown. The basic principles of rotor aerodynamics (see [6]) state that a rotor produces a thrust and a torque depending on many aerodynamic quantities. Rotor aerodynamics is very complex due to the unsteadiness of the motion of rotor blades and of the turbulent flow field it generates. For small scale UAVs thrust and power originated from each propeller is modeled in a very simple way, neglecting all aerodynamic effects that can influence rotor blades behaviour during a flight. It is straightforward that the principles of helicopter motion are very different from multirotor ones, as a helicopter rotor has a mechanical chain to vary collective and cyclic pitch, which in turn commands rotor's blades motion, producing the desired thrust. The basic principle of motion for a small scale UAV is based just on control forces and moments that arise by simply varying each rotor-motor angular speed. Recall the classical definitions for rotor thrust T and power W expressed in [6]:

$$T = C_T \rho A v_{tip}^2, \quad (2.36)$$

$$W = C_W \rho A v_{tip}^3, \quad (2.37)$$

where Ω is the angular speed of the rotor, R is rotor radius, $A = \pi R^2$ is the rotor disk area and in turn $v_{tip} = \Omega R$ is the local velocity at the blade tip when the rotor is in hovering condition, C_T and C_W are thrust and power coefficient respectively.

In this way it is possible to define also the torque Q (and torque coefficient C_Q) acting on a single rotor through the relationship between rotor torque and power W

$$W = Q\Omega \rightarrow Q = C_W\rho AR^3\Omega^2 = C_Q\rho AR^3\Omega^2, \quad (2.38)$$

so in this way

$$C_Q = C_W. \quad (2.39)$$

These definitions are often used to define a simple static relationship between each propeller angular speed ω_i and the corresponding generated thrust and torque when modeling the flight dynamics of a small scale quadrotor.

$$T_i = K_T\omega_i^2 = C_T\rho AR^2\omega_i^2, \quad (2.40)$$

$$Q_i = K_Q\omega_i^2 = C_Q\rho AR^3\omega_i^2. \quad (2.41)$$

Denoting with b the distance between the center of gravity and the i -th propeller it is possible to write the equations of the forces and the moments produced by the four propellers taking into account the symmetric X -configuration of the quadrotor.

$$F_{props} = \begin{bmatrix} 0 \\ 0 \\ K_T \sum_{i=1}^4 \omega_i^2 \end{bmatrix}, \quad (2.42)$$

$$M_{props} = \begin{bmatrix} K_T \frac{b}{\sqrt{2}} (\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2) \\ K_T \frac{b}{\sqrt{2}} (\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \\ K_Q (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix}. \quad (2.43)$$

These forces and moments acting to the quadrotor can be rearranged to realize the mixer matrix of the motors (χ). χ is a matrix that relates the required thrust and moments around each axis to the rotational speed of the propellers (that are

the actual control input of the quadrotor).

$$\begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T \\ K_T \frac{b}{\sqrt{2}} & -K_T \frac{b}{\sqrt{2}} & -K_T \frac{b}{\sqrt{2}} & K_T \frac{b}{\sqrt{2}} \\ K_T \frac{b}{\sqrt{2}} & K_T \frac{b}{\sqrt{2}} & -K_T \frac{b}{\sqrt{2}} & -K_T \frac{b}{\sqrt{2}} \\ -K_Q & K_Q & -K_Q & K_Q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \chi \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}. \quad (2.44)$$

It follows that the required rotor-motor angular speeds come from

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \chi^{-1} \begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix}. \quad (2.45)$$

Moreover, besides the control actions generated by the quadrotor, another external force is represented by gravity. Gravity force is directed downward along the Z direction of Earth frame, so it must be projected among moving body axes, *i.e.*:

$$F_g = T_E^B(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (2.46)$$

Other forces to be considered are the aerodynamic ones related to the air resistance, the so-called *drag*. This force originates from many other aerodynamic phenomena which take place in a rotor during flight and depends on several quantities which will be analyzed in the next section.

2.4 Rotor aerodynamics

As mentioned before, quadrotor dynamics is obviously influenced by aerodynamics, in particular by unsteady rotor aerodynamics. Nowadays, little literature is available regarding multirotor aerodynamic interaction. In fact, a quadrotor helicopter must face not only classical rotor aerodynamics, but also the continuous interaction between the four rotor wakes. What happens in a quadrotor wake is obviously a black box, and since it is out of the object of this thesis it will not be

analyzed or modeled.

For what concerns rotor behavior during flight, it is possible to extend the dynamic model previously introduced in Chapter 2.3, considering the variation of thrust and drag forces on each rotor, which depends on the local velocity. To be precise, the climb velocity influences the thrust while the in-plane velocity has a role in the drag loads.

As is well known in the helicopter literature, thrust depends on many factors. The overall rotor thrust depends on lift distribution over the rotor. During helicopter flight, many phenomena occur: flapping, lagging and pitch motion of each blade continuously arise to guarantee rotor equilibrium conditions; this makes rotor aerodynamics unsteady.

Depending on the flight condition, rotor performances can be analyzed. In fact the helicopter must tilt the thrust in a forward flight thanks to blade flapping. The rotor disk plane is tilted forward, hence the helicopter velocity projected on \mathcal{F}_R has three components. Consequently, the results obtained for a forward flight condition summarize all the other cases, such as hovering and vertical flight.

The next subsection contains the basic definitions needed, while the ones after present a brief summary about the well-known aerodynamic theories applied in helicopter field.

2.4.1 Rotor blades motion

When investigating the angle of attack of a blade, one has always to refer to the local angle of attack. The lift produced by one blade is distributed along the span-wise direction. So, each blade section produces a lift force according to its corresponding angle of attack and relative velocity.

Figure 2.2 summarizes the various reference planes used for the analysis of helicopter rotor in forward flight. These planes are:

- Hub Plane (HP): it is a plane perpendicular to the rotor shaft in which an observer would see both flapping and feathering during forward flight. It is the most complicated for analysis of the rotor, but it is linked to a physical part of the helicopter. The HP is often used for blade dynamic and flight dynamic analyses.
- No Feathering Plane (NFP): it is a plane where an observer sees no variation

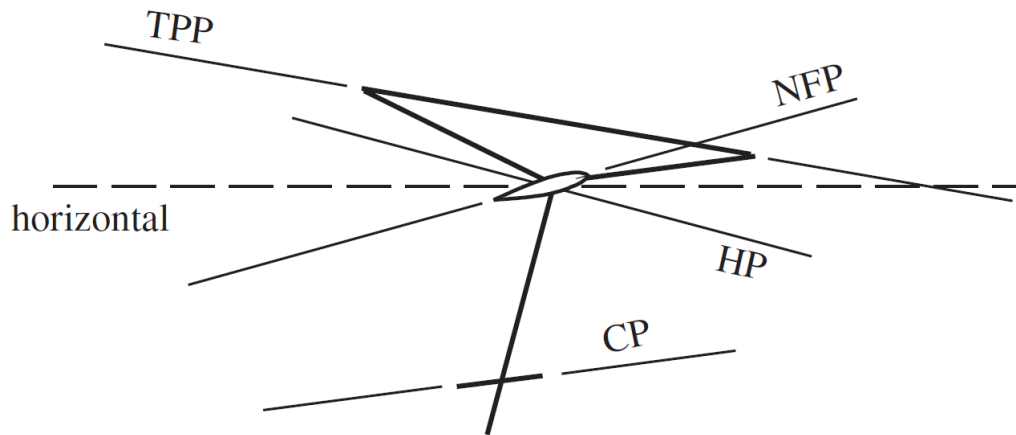


Figure 2.2: Rotor reference planes: tip-path plane (TPP), no-feathering plane (NFP), hub plane (HP), and control plane (CP) (picture from [1]).

in cyclic pitch: however, the observer will still see a cyclic variation in blade flapping angle. Normally, NFP is used for performance analysis.

- Tip Path Plane (TPP): it is the plane whose boundary is described by the blade tip positions. Therefore, an observer will see no variation in flapping. The TPP is commonly used for aerodynamic analyses, such as rotor inflow or other wake models.
- Control Plane (CP): it is the plane that represents the commanded cyclic pitch plane and is sometimes known as the swashplate plane.

In vertical flight, the natural reference plane is the horizontal one. Adding axial symmetry, the tip-path plane and no-feathering plane are horizontal. The hub plane is not necessarily horizontal in vertical flight unless the helicopter CG is on the rotor shaft axis. On the other hand, in forward flight the above described reference planes have physical meaning, and due to the asymmetry of the aerodynamics in that condition, these planes do not in general coincide with the horizontal plane or with each other. This clarification is made in order to state that all the following considerations can be done considering any of these reference planes.

As mentioned in [1], the blade section pitch θ is measured from the reference plane to the zero-lift line; θ includes the collective and cyclic pitch controls and

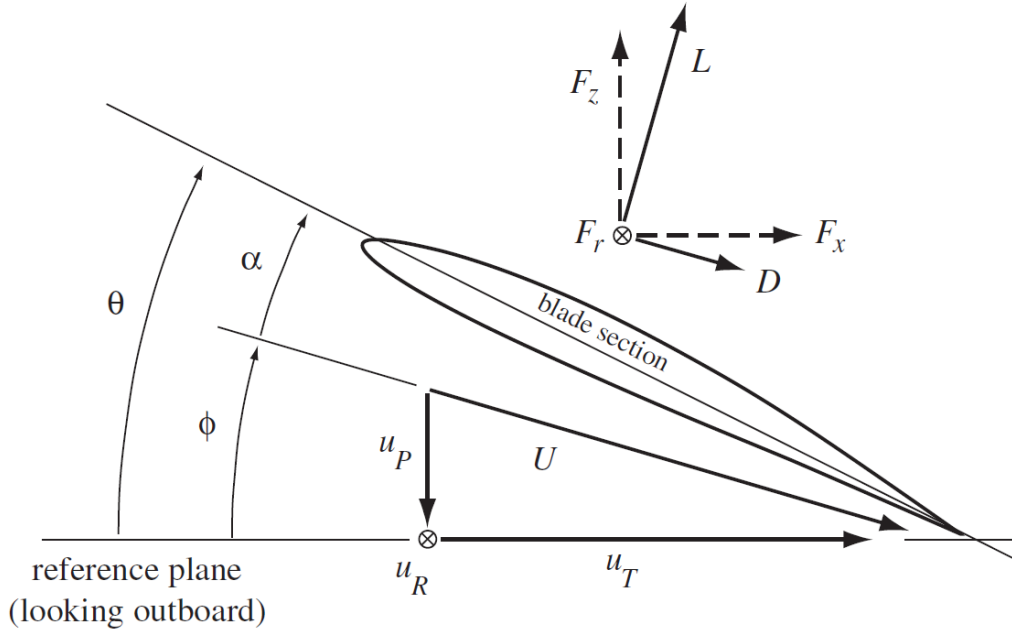


Figure 2.3: Aerodynamics of the rotor blade section (picture from [1]).

the built-in twist of the blade. As shown in Figure 2.3, the components of the airspeed relative to the blade are the in-plane velocity u_T (tangential to the disk plane, positive toward the trailing edge), u_P (perpendicular to the disk plane, positive downward), and u_R (radial, positive outward). The resultant velocity U and inflow angle ϕ of the section are:

$$U = \sqrt{u_T^2 + u_P^2 + u_R^2}, \quad (2.47)$$

$$\phi = \arctan\left(\frac{u_P}{u_T}\right). \quad (2.48)$$

Then, the section angle-of-attack is:

$$\alpha = \theta - \phi. \quad (2.49)$$

In Figure 2.3 are also depicted the aerodynamic forces on the blade section. The aerodynamic lift and drag (L and D) are, respectively, normal to and parallel to the resultant velocity U . The components of the section lift and drag resolved in the reference plane are F_z and F_x (normal and in-plane, respectively). In the non-rotating axis system of the adopted reference plane (Figure 2.4), x and

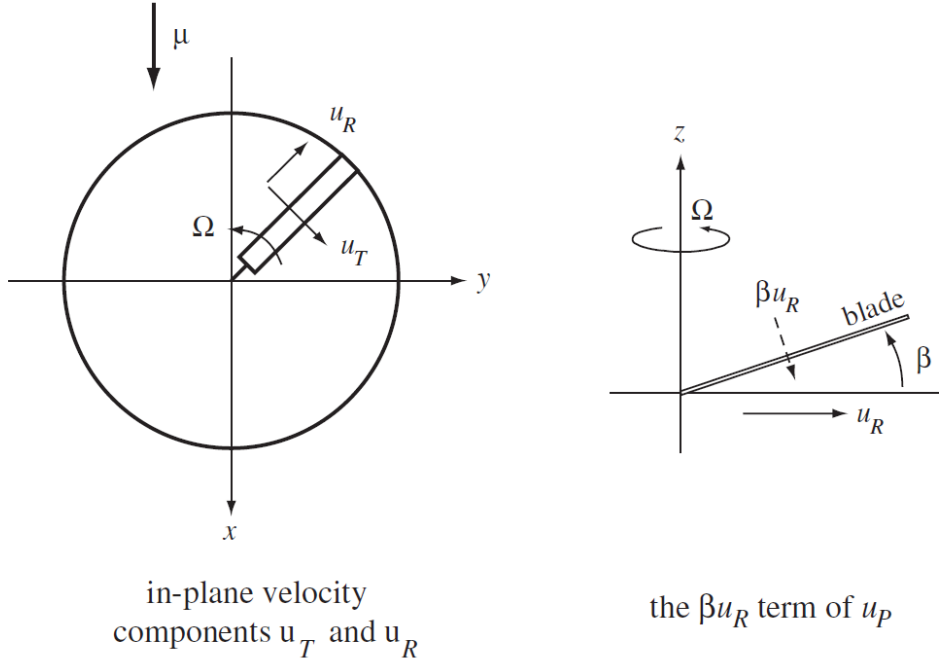


Figure 2.4: Air velocity relative to the blade in forward flight [1].

y lie in the reference plane and z is normal to it. The flap and pitch angles are measured with respect to the reference plane. The forward velocity has magnitude V and lies in the x - z plane at an incidence angle i (positive for forward tilt of the disk, see Figure 2.5). The rotor induced velocity u is assumed to be normal to the reference plane. The advance ratio μ and the total inflow ratio λ are the dimensionless velocity components parallel to and normal to the reference plane, respectively:

$$\mu = \frac{V \cos(i)}{\Omega R}, \quad (2.50)$$

$$\lambda = \frac{V \sin(i) + u}{\Omega R} = \mu \tan(i) + \lambda_i = \lambda_c + \lambda_i, \quad (2.51)$$

where λ_i is the well-known *induced inflow ratio* and λ_c is called *climb inflow ratio*.

The velocity seen by the blade section is due to the rotor rotation, the helicopter forward speed, induced velocity and the blade flap motion. To lowest order the local tangential and radial components u_T and u_R are due solely to the rotor

rotation (azimuthal position of the blade, ψ) and advance ratio. Hence:

$$u_T(r, \psi) = \Omega r + \mu \Omega R \sin(\psi), \quad (2.52)$$

$$u_R(\psi) = \mu \Omega R \cos(\psi), \quad (2.53)$$

where r is the local radial station of the blade section.

The normal velocity u_P has three terms:

1. u , which is the induced velocity of the rotor, plus the component of the free stream velocity normal to the rotor disk (equation (2.51));
2. $\frac{rd\beta}{dt}$, which is the angular velocity of the blade about the flap hinge;
3. $\Omega R \beta \mu \cos(\psi)$, which is a component of the radial velocity u_R normal to the blade when the blade is flapped up by the angle β (see Figure 2.4).

Thus, the normal velocity is

$$u_P(r, \psi) = \lambda \Omega R + \dot{\beta} r + \Omega R \beta \mu \cos(\psi). \quad (2.54)$$

In deriving the expressions of the velocity components, the flap angle β was assumed to be small, giving $\sin(\beta) \approx \beta$ and $\cos(\beta) \approx 1$. Another consequence of small angles assumption is that $\alpha = \theta - \frac{u_P}{u_T}$ and it is easily shown to be invariant during a reference plane transformation.

If there is any sideways velocity, it must be taken into account. Total velocity V will have an advance ratio μ in wind axis coordinates, and it can be decomposed in rotor coordinates into μ_1 and μ_2 : the dimensionless forward and sideward flight velocities respectively (see [7]).

It is clear that in case of hovering or climb, neither advancing ratio nor flapping motion are present, so blade section tangential velocity u_T is just due to rotational speed and u_P depends only on λ . In those cases μ includes the rate of climb too, as TPP is horizontal and perpendicular to rotor relative velocity, *i.e.*:

$$\lambda = \frac{u + v_c}{\Omega R} = \lambda_i + \lambda_c. \quad (2.55)$$

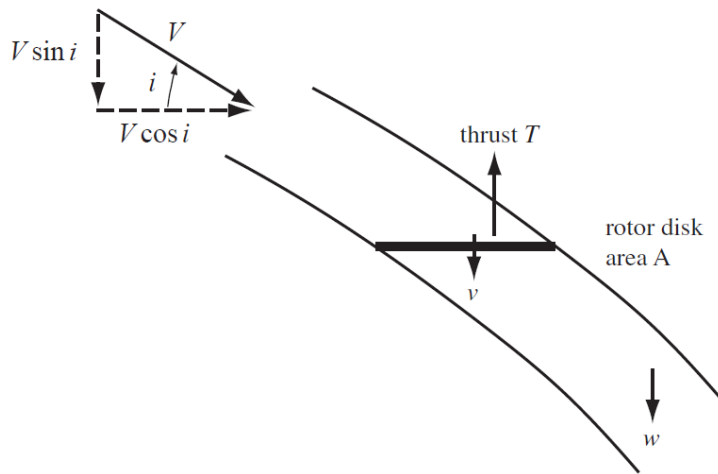


Figure 2.5: Flow model for momentum theory analysis of rotor in forward flight (picture from [1]).

2.4.2 Momentum theory

After introducing rotor blades velocity components, it is possible to look at rotor disk performance from a global point of view, considering the rotor disk as a unique rotating entity. This is the basic assumption of *momentum theory* which defines rotor thrust thanks to the *actuator disk theory*. Obviously, the theory has different results according to the flight condition, but the expression for thrust is roughly unique and dependent just on few quantities. For the sake of completeness, see [1] to understand the theoretical assumptions and the validity of this theory when applied to forward flight condition.

An appropriate solution for the induced velocity of the helicopter rotor in forward flight is:

$$u = \frac{T}{2\rho AU}, \quad (2.56)$$

where A is the rotor disk area. The thrust in forward flight can be also written as:

$$T = \dot{m}2u \quad (2.57)$$

where $\dot{m} = \rho AU$ is the mass flux through the rotor disk area. In hover and vertical

flight

$$\dot{m} = \rho A (V + u), \quad (2.58)$$

where V is the rotor velocity. Obviously V is null in hovering condition while it is the climb or descent velocity in a vertical flight.

The induced velocity at the disk is u . In the far wake, the velocity $w = 2u$ and is assumed to be parallel to the rotor thrust vector. Thus a uniformly valid expression for induced velocity can be obtained by considering the mass flux through the area A for all operating conditions. This observation was first made by Glauert (1926).

Momentum conservation gives the rotor thrust shown in equation (2.57), hence the resultant velocity U passing through the rotor disk is given by:

$$U^2 = (V \cos(i))^2 + (V \sin(i) + u)^2 = V^2 + 2Vu \sin(i) + u^2. \quad (2.59)$$

Eventually the rotor thrust can be explicitly written as

$$T = 2\rho Au \sqrt{V^2 + 2Vu \sin(i) + u^2}. \quad (2.60)$$

From equation (2.60), the expressions of thrust in hover ($V = 0, u = u_h$) and in vertical flight ($V = v_c, i = \frac{\pi}{2}$) can be derived:

- in hover, $T_h = 2\rho Au_h^2$,
- in vertical flight, $T = 2\rho A (v_c + u) u$.

As described in [1], the assumption that thrust in forward flight (or vertical flight) is the same as in hovering is usually made, *i.e.*:

$$T \cong T_h \rightarrow u_h^2 = Uu. \quad (2.61)$$

Therefore, induced velocity can be always related to the hovering induced velocity:

$$u = \frac{u_h^2}{\sqrt{(V \cos(i))^2 + (V \sin(i) + u)^2}}. \quad (2.62)$$

Expressing hovering induced velocity as a function of C_T , it results:

$$u_h^2 = \frac{T}{2\rho A} = \frac{C_T \rho A (\Omega R)^2}{2\rho A} = \frac{C_T (\Omega R)^2}{2}. \quad (2.63)$$

Now, recalling the advancing and inflow ratios definition in equations (2.50) and (2.51), an analytic implicit equation for induced velocity is obtained:

$$\lambda_i = \frac{C_T}{2\sqrt{\mu^2 + \lambda^2}}. \quad (2.64)$$

The latter is known as the Glauert inflow formula. It can be directly solved only knowing μ and λ_c , otherwise an iterative method must be used.

From equation (2.64) an important relationship is obtained between thrust and induced inflow in hovering, *i.e.*

$$\lambda_h = \sqrt{\frac{C_T}{2}}. \quad (2.65)$$

Rearranging equation (2.62) for rotor in vertical flight (climb or descent) it is possible to derive a relationship between the rate of climb and the induced velocity u , for a given hovering induced velocity u_h . The sign convention (important when the descent case is considered) is that the thrust is positive upward and the velocities are positive downward. Suppose that the rotor is climbing at velocity V :

$$\frac{u}{u_h} \left(\frac{V + u}{u_h} \right) = 1. \quad (2.66)$$

Introducing dimensionless velocities $u^* = \frac{u}{u_h}$, $V^* = \frac{V}{u_h}$, equation (2.66) has a closed form solution:

$$u^* = -\frac{V^*}{2} + \frac{\sqrt{V^{*2} + 4}}{2} \quad (2.67)$$

where the positive sign on the second term is taken since u^* must be positive. The solution is physically justified for a climb. The climb model cannot be used with $V < 0$, because in descent the free stream velocity is directed upward and therefore the far downstream wake is above the rotor disk. V is negative now, whereas T , u , and w are still positive.

The momentum theory result for the induced velocity in descent is

$$T = -2\rho A(V + u)u$$

or

$$\frac{u}{u_h} \left(\frac{V + u}{u_h} \right) = -1, \quad (2.68)$$

whose solution is:

$$u^* = -\frac{V^*}{2} - \frac{\sqrt{V^{*2} - 4}}{2}. \quad (2.69)$$

This flow condition, where $V + u < 0$ and $V + w < 0$ is called the windmill brake state. The other solution of the quadratic equation for u gives $u > 0$ and $V + u < 0$ as required, but $V + w > 0$. Thus, the flow in the far wake would be downward, in contradiction with the assumed flow model. In Figure 2.6, one can see the validity region of momentum theory solutions. The dashed portions of the curves are branches of the solution that do not correspond to the assumed flow state. The line $V + w = 0$ is where the flow through the rotor disk changes sign. The lines

$$V = 0, \quad (2.70)$$

$$V + u = 0, \quad (2.71)$$

$$V + 2u = 0 \quad (2.72)$$

divide the plane into four regions, namely the rotor operating conditions:

- the normal working state (climb and hover),
- vortex ring state (VRS),
- turbulent wake state (TWS),
- windmill brake state.

Thus, in the VRS and TWS the flow outside the slipstream is upward while the flow inside the far wake is nominally downward. Such a flow state is not possible, so there is no valid momentum theory solution for moderate rates of descent,

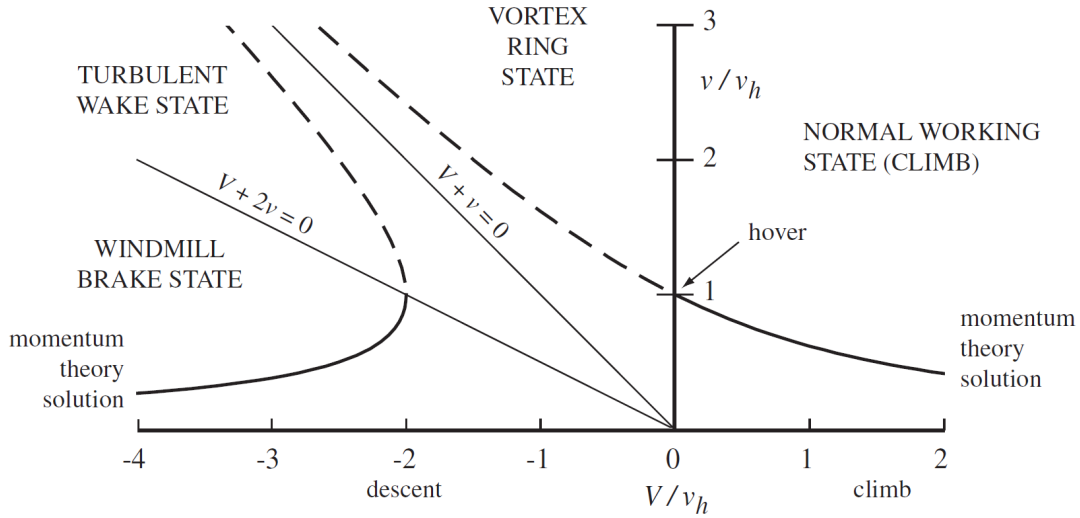


Figure 2.6: Momentum theory results for the induced velocity in vertical flight (picture from [1]).

between $V = 0$ and $V = -2u_h$. However, if the descent is combined with a forward flight motion, with sufficiently high forward speed,

$$\mu > \lambda_h,$$

no VRS exists, since the wake is swept back.

Since the induced velocity curve is not analytically predictable in this range, experimental estimates can be used to find a best-fit approximation. There are several possibilities in the literature, one could be:

$$u^* = \kappa + \kappa_1 V^* + \kappa_2 V^{*2} + \kappa_3 V^{*3} + \kappa_4 V^{*4}, \quad (2.73)$$

where $\kappa_1, \kappa_2, \kappa_3, \kappa_4$ are experimentally determined polynomial coefficients, and κ is the induced power correction factor, derived from experimental rotor measures (ideally 1, typical range $1.05 \div 1.15$). Many other models can be found in literature, see [6] and [1] for completeness.

2.4.3 Blade element theory - BET

In [6], a fully developed description of BET for both hover and forward flight is presented. This theory provides a thrust expression from a differential point of view, analysing the thrust of an infinitesimal portion of each single blade. As illustrated above in Section 2.4.1, the incremental lift force of a blade section depends on its relative velocity and local angle of attack. Recalling equation (2.48), several reasonable assumptions regarding velocity and angle of attack can be made:

- $U \approx u_T$,
- $\phi \approx \frac{u_P}{u_T}$,

because u_P is generally small with respect to u_T and so, inflow angle is considered small.

The resulting incremental lift dL per unit span on the blade element is:

$$\begin{aligned}
 dL &= \frac{1}{2}\rho U^2 c C_L dr = \frac{1}{2}\rho u_T^2 c C_{L\alpha} (\theta - \phi) dr \\
 &= \frac{1}{2}\rho u_T^2 c C_{L\alpha} \left(\theta - \frac{u_P}{u_T} \right) dr \\
 &= \frac{1}{2}\rho c C_{L\alpha} (\theta u_T^2 - u_P u_T) dr
 \end{aligned} \tag{2.74}$$

and the incremental drag is:

$$dD = \frac{1}{2}\rho u_T^2 c C_D dr, \tag{2.75}$$

where c is the local blade chord, C_L and C_D are the lift and drag coefficients respectively. These forces can be resolved in the rotor disk plane coordinates, see Figure 2.3:

$$dF_z = dL \cos(\phi) - dD \sin(\phi) \approx dL, \tag{2.76}$$

$$dF_x = dL \sin(\phi) + dD \cos(\phi) \approx \phi dL + dD, \tag{2.77}$$

because drag is one order of magnitude less than lift usually. In forward flight, blade element velocities are described by equations (2.52), (2.53), and (2.54). In BET the aerodynamic effects resulting from u_R are neglected. Therefore, the

contributions to the thrust and torque are

$$dT = N_b dF_z = N_b dL, \quad (2.78)$$

$$dQ = N_b dF_x r = N_b (\phi dL + dD), \quad (2.79)$$

where N_b is the number of blades comprising the rotor. For simplicity, the chord c and the pitch angle θ are considered constant along the blade (see [8]), and flapping angle motion is described in the following way:

$$\beta = a_0 - a_1 \cos(\psi) - b_1 \sin(\psi), \quad (2.80)$$

$$\dot{\beta} = \frac{d\beta}{dt} = \frac{d\beta}{d\psi} \frac{d\psi}{dt} = \beta' \Omega, \quad (2.81)$$

where a_0 is the coning angle and a_1, b_1 are the longitudinal and lateral flap angles respectively. Accordingly, the elementary thrust is:

$$\begin{aligned} dT &= \frac{1}{2} N_b \rho c C_{L\alpha} (\theta u_T^2 - u_P u_T) dr \\ &= \frac{1}{2} N_b \rho c C_{L\alpha} \Omega^2 R^3 [\theta (y + \mu \sin(\psi))^2 \\ &\quad - (\lambda + y\beta' + \mu\beta \cos(\psi)) \times (y + \mu \sin(\psi))] dy, \end{aligned} \quad (2.82)$$

where $y = \frac{r}{R}$ is the dimensionless radial position of the blade element. Then, the resulting thrust can be calculated in two steps:

1. averaging the incremental thrust dT over the azimuth range $\psi = 0 \div 2\pi$;
2. integrating the incremental thrust along the blade span-wise direction ($y = 0 \div 1$).

The result is:

$$T = \frac{1}{4} N_b c R \rho \Omega^2 R^2 \left(\frac{2}{3} \theta \left(1 + \frac{3\mu^2}{2} \right) - \lambda \right). \quad (2.83)$$

Therefore, in forward flight the thrust coefficient is:

$$C_T = \frac{1}{4} \sigma C_{L\alpha} \left(\frac{2}{3} \theta \left(1 + \frac{3\mu^2}{2} \right) - \lambda \right), \quad (2.84)$$

where

$$\sigma = \frac{N_b c R}{\pi R^2} \quad (2.85)$$

is the rotor solidity ratio (blades area over rotor disk area).

A useful result is provided by BET for hovering and axial flight, when the blade twist is the ideal one, *i.e.*, $\theta = \frac{\theta_{TIP}}{y}$. In this case, it is:

$$u_T = \Omega r, \quad (2.86)$$

$$u_P = v_c + u = \lambda \Omega R. \quad (2.87)$$

Recalling equations (2.74) and (2.82), it is possible to derive the elementary thrust coefficient, and to proceed further with dimensionless quantities. Because $\mu = 0$ and $\beta = \dot{\beta} = 0$, the incremental C_T is:

$$dC_T = \frac{1}{2} \sigma C_{L\alpha} y^2 dy. \quad (2.88)$$

Assuming again ideal twist and recalling equation (2.49), it becomes:

$$C_T = \frac{1}{2} \sigma C_{L\alpha} \int_0^1 \left(\theta - \frac{\lambda \Omega R}{\Omega r} \right) y^2 dy \quad (2.89)$$

$$= \frac{1}{2} \sigma C_{L\alpha} \int_0^1 \left(\frac{\theta_{TIP}}{y} - \frac{\lambda}{y} \right) y^2 dy \quad (2.90)$$

$$= \frac{1}{4} \sigma C_{L\alpha} (\theta_{TIP} - \lambda). \quad (2.91)$$

The reader can notice the likelihood between equation (2.84) and equation (2.91). In equation (2.83) a constant pitch angle θ along spanwise direction was assumed, per each blade. In reality, all blades are twisted. From a mathematical point of view, equation (2.82), with an ideal twist, gives a nonphysical solution due to the divergence of thrust at the blade root. On the other hand, equation (2.84) can be simplified as the quadratic term is negligible with respect to the advance ratio μ almost all the time, especially for quadrotors. In this manner, the first term of the AoA dependent on θ can just be formalized as a parameter when used in a dynamic model, provided that both equations (2.84) and (2.91) respect the relationship in hovering condition, always true, stated in equation (2.65).

2.4.4 Blade element momentum theory

After a careful study on the interdependence between thrust coefficient and velocity upcoming to the rotor, one can equalize *momentum theory* and *BET* solutions, obtaining suitable equations in terms of λ radial distribution. Let's see the case of hovering and axial flight.

The incremental thrust coefficient dC_T for momentum theory is:

$$\begin{aligned}
 dC_T &= \frac{dT}{\rho(\pi R^2)(\Omega R)^2} \\
 &= \frac{2\rho(v_c + u)udA}{\rho(\pi R^2)(\Omega R)^2} \\
 &= 4\left(\frac{v_c + u}{\Omega R}\right)\left(\frac{u}{\Omega R}\right)\left(\frac{r}{R}\right)d\left(\frac{r}{R}\right) \\
 &= 4\lambda\lambda_i y dy.
 \end{aligned} \tag{2.92}$$

Equalizing equations (2.92) and (2.88), and expanding lift coefficient as previously done in equation (2.74), the outcome is:

$$C_L = C_{L\alpha}\alpha = C_{L\alpha}\left(\theta - \frac{u_P}{u_T}\right); \tag{2.93}$$

and remembering that in axial flight $u_P = \lambda\Omega R$ and $u_T = \Omega R$, it becomes:

$$C_L = C_{L\alpha}(\theta - \lambda). \tag{2.94}$$

Hence, an analytic equation for the distribution of λ over the blade is obtained:

$$\lambda^2 + \left(\frac{\sigma C_{L\alpha}}{8} - \lambda_c\right)\lambda - \left(\frac{\sigma C_{L\alpha}}{8}\theta y\right) = 0, \tag{2.95}$$

which has a closed form solution. Equation (2.95) has a unique and physically admissible solution for λ (and so λ_i), if λ_c and the local pitch angle θ are known. In case of ideal twist, inflow is no more dependent on rotor coordinates and it is constant all over the blade. Just vertical velocity and blade tip twist angle are the forcing terms in the resulting equation.

2.4.5 Dynamic inflow model (Pitt and Peters)

The previous results were derived on the basis of the key assumption that just axial flight condition is considered, where the inflow distribution is axisymmetric. In reality, as described in [6], when a rotor moves forward an estimate of inflow distribution should be taken into account, besides blade pitch and rotor flapping motions, in order to estimate rotor performance. Many inflow models have been developed in the literature, aimed to better describe rotor loads in forward flight. The most famous model present in the literature, exploiting also the dynamics of inflow, is surely the one by Pitt and Peters. In [7], an article published later than the first publication about this topic, the authors study the dynamics of rotor inflow due to all velocity components approaching to rotor blades.

According to [7], the inflow distribution over the rotor is dependent on azimuth and radial coordinates. In wind-axis coordinates the inflow variation over the rotor is due to:

1. v_0 , the uniform variation;
2. v_s , the lateral variation;
3. v_c , the longitudinal variation.

Consequently, the inflow can be expressed as:

$$\lambda(r, \psi) = v_0 + v_s \frac{r}{R} \sin(\psi) + v_c \frac{r}{R} \cos(\psi). \quad (2.96)$$

The dynamic of v_0 , v_s and v_c is governed by the following first-order differential equations:

$$M \begin{bmatrix} \dot{v}_0 \\ \dot{v}_s \\ \dot{v}_c \end{bmatrix} + L_{nl}^{-1} \begin{bmatrix} v_0 \\ v_s \\ v_c \end{bmatrix} = \begin{bmatrix} C_T \\ -C_L \\ -C_M \end{bmatrix}_{aero}, \quad (2.97)$$

where M is the apparent mass matrix, L_{nl} is the non-linear version of the inflow gains matrix (completely non linear in C_T and v_0 , but linear in C_L and C_M), C_T is the instantaneous thrust coefficient, and C_L and C_M are roll and pitching moment coefficients respectively, in the wind-axis coordinates. Through a transformation

matrix T , dependent on the angles between wind-axes and natural rotor coordinates, such as those described in equation (2.2), it is possible to switch to rotor disk plane:

$$\begin{bmatrix} \lambda_0 \\ \lambda_s \\ \lambda_c \end{bmatrix} = T^T \begin{bmatrix} v_0 \\ v_s \\ v_c \end{bmatrix}. \quad (2.98)$$

Now, the dynamic inflow equation has the same form, but it involves the inflow variations in the reference plane already mentioned in previous theories:

$$M \begin{bmatrix} \dot{\lambda}_0 \\ \dot{\lambda}_s \\ \dot{\lambda}_c \end{bmatrix} + \widehat{L}^{-1} \begin{bmatrix} \lambda_0 \\ \lambda_s \\ \lambda_c \end{bmatrix} = \begin{bmatrix} C_T \\ C_1 \\ -C_2 \end{bmatrix}_{aero}. \quad (2.99)$$

The parameters C_1 and C_2 represent the lateral roll and longitudinal pitching moment coefficients. See [7] to recover all the terms belonging to the resulting non-linear inflow gains matrix in the rotor disk plane \widehat{L} . At the end, the normal induced inflow due to the effect of rotor thrust is:

$$\lambda_i = \frac{1}{2} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}^T \mathcal{L}^{-1} \begin{bmatrix} \dot{\lambda}_0 \\ \dot{\lambda}_s \\ \dot{\lambda}_c \end{bmatrix} \quad (2.100)$$

with $\mathcal{L}^{-1} = T^T L_{nl}^{-1} T$. As explained in [7], all these equations are expressed in a normalized time with respect to the rotor speed Ω , *i.e.*:

$$\dot{\lambda}_0 = \frac{d\lambda_0}{dt^*} = \frac{d\lambda_0}{dt\Omega} = \frac{1}{\Omega} \frac{d\lambda_0}{dt}. \quad (2.101)$$

2.5 Aerodynamic modeling of a quadrotor

The flight regime of small-sized UAVs is not always the hovering condition, of course. The accepted nonlinear dynamic quadrotor model is based on a thrust and torque model with constant thrust and torque coefficients derived from static thrust tests. Such a model is no longer valid when the vehicle undertakes dynamic maneuvers that involve significant displacement velocities. In these cases,

besides the continuous changes in the propellers angular speed, each rotor thrust is affected by the unsteady aerodynamic field surrounding the rotor itself. Thus, thrust aerodynamic variations must be taken into account during the flight.

At this point, a suitable aerodynamic model must be implemented, aiming to represent as close as possible the real rotor aerodynamic forces during flight. To include the dependency of thrust on the aerodynamic environment, the quadrotor physical model must comprise all the parts where aerodynamic forces develop, such as the main central body and the four arms, up to the four rotors. A multi-body approach allows to define the components of a system, and their constraints, more easily. A multi-body system can be defined as a system consisting of a number of interconnected mechanical components, moving in three-dimensional space. This includes rigid and flexible bodies, as well as free or actuated joints. Specifically, the Modelica MultiBody library has been used in this work. From the aerodynamic modeling point of view, each rotor can be considered as a stand-alone entity, as its aerodynamic performance depends only on its own aerodynamic environment. The theories presented in Section 2.4 depend on many quantities, such as velocity V and rotor incidence i , expressed in the rotor disk reference frame \mathcal{F}_R . Thanks to the multi-body approach, it is easy to define the aerodynamic environment surrounding each rotor of the quadcopter. Furthermore, it must be remembered that the model developed in this thesis does not take into account rotor wake interactions between the four propellers.

2.5.1 The multi-body model

Modelica is a language for hierarchical, equation-based, object-oriented modelling of physical systems, developed by the *Modelica Association* [9]. The main features of the language which are relevant within the scope of this work are summarized here. Modelica is an object-oriented language, supporting encapsulation, composition and inheritance. These features facilitate model development and update. Elementary models of physical elements are defined in a declarative way by their constitutive equations, and their interface with the outer world is described by physical ports (or connectors) without any implied causality, rather than by writing assignments relating inputs to outputs. The goal of the library is to simplify modeling and dynamic analysis of rigid multi-body systems, while at the same time providing efficient simulations of such systems. Arbitrarily complex multi-

body systems can be obtained by dragging and dropping the standard library components into the workspace, and by further connecting them in a suitable way. Complex models can then be built by connecting elementary models through their ports; since the ports are a-causal, any connection which is physically meaningful is allowed without restrictions. The Modelica language includes graphical annotations, which allow to use graphical user interfaces (such as the one provided by the tool Dymola [3]) to select components from a library, drag them into a diagram, connect them, and set their parameters, thus making the process of model development highly intuitive for end users.

External forces and torques acting on each rigid body can be assigned by means of dedicated library components. As an alternative, it is also possible to assign their motion, or motion, forces and torques jointly. In this latter case, attention must be paid to not specify more constraints than the system's actual degrees of freedom.

A World model must always be present at the top level of a generic Modelica MultiBody Library model. Within the World model the inertial reference frame and the gravity field models are defined, jointly with all parameters relevant to the components animation. The Modelica MultiBody library provides the opportunity to specify animation properties for all components. This feature proves an invaluable visual aid for model checking and debugging purposes.

The quadrotor physical model along with its aerodynamics is developed in Dymola 2018 [3], without the flight control unit. As illustrated in Figure 2.7, $\omega_1, \omega_2, \omega_3$ and ω_4 are the inputs of the model. These are the four desired motor-rotor angular speeds, which are computed thanks to the mixer matrix χ as depicted in equation (2.45). The output of the model is the quadrotor state, comprising:

- position in inertial frame;
- velocity in inertial frame;
- attitude expressed in quaternions;
- angular_rates (p, q, r) .

The `ground_contact` input represent the reaction force applied to the drone when it lays on ground.

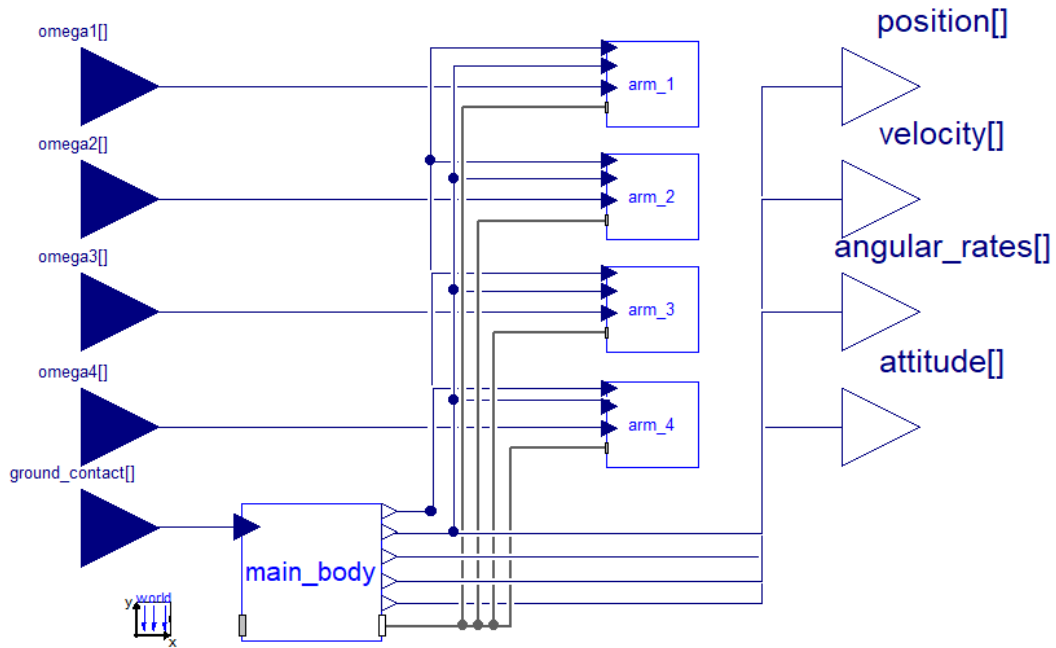


Figure 2.7: Quadcopter multi-body model overview.

The default working reference frame of the Modelica MultiBody library is different from any of the listed in Section 2.1:

- x -axis points rightward;
- y -axis points upward, opposed to gravity direction;
- z -axis completes the right-hand rule, pointing rearward.

As a consequence, all the output variables must be transformed from this triad to \mathcal{F}_E , through a suitable rotation matrix (e.g., equation (2.1)).

The quadrotor prototype modeled in this section is illustrated in details in Section 2.7.1. Mass, inertia tensor and physical dimensions of the prototype are necessary to build up the model, as well as the static thrust coefficient. All these parameters are reported in Table 2.1. It is easy to figure out that the entire model has no elastic properties, rather it is designed as rigid.

The main body of the quadcopter has been modeled with a cylindrical shape, as well as the four arms. The only difference between those rigid bodies is that the main central body has an assigned mass and inertia, whereas the four arms

Name	Description	Value	Unit of measure
m_{mb}	Main body mass	1.31	kg
J_{xx}	Main body Inertia tensor x -component	0.035	$kg\ m^2$
J_{yy}	Main body Inertia tensor y -component	0.035	$kg\ m^2$
J_{zz}	Main body Inertia tensor z -component	0.05	$kg\ m^2$
r_{mb}	Main body cylinder radius	0.06	m
h_{mb}	Main body cylinder height	0.05	m
m_{rot}	Rotor mass	0.05	kg
J_{rotxx}	Rotor Inertia tensor x -component	$8.54\ 10^{-5}$	$kg\ m^2$
J_{rotyy}	Rotor Inertia tensor y -component	$8.54\ 10^{-5}$	$kg\ m^2$
J_{rotzz}	Rotor Inertia tensor z -component	$4.58\ 10^{-5}$	$kg\ m^2$
l_{arm}	Arm length	0.275	m
d_{arm}	Arm diameter	0.025	m
K_T	Static Thrust coefficient	$2.4619\ 10^{-5}$	$N/(rad/s)^2$
R	Rotor radius	0.1524	m

Table 2.1: Quadrotor mass and geometric properties.

are assumed to have negligible mass properties. In that sense, they stand just for a rigid link connecting each rotor to the central body of the drone. Proceeding further, within the arm blocks there are the relevant components of the overall model.

2.5.2 Thrust and inflow

Many rotor aerodynamic models could have been implemented above those described in Section 2.4, but one has now to refer to quadrotor flight regimes, not to helicopter ones.

In [10], an interesting and detailed work about modeling of quadrotor aerodynamics is presented. A quadrotor, similar to the one used for this work, has been modeled including aerodynamics at first, then the latter is used along with the electrical dynamics of a motor-rotor system in the computation and estimation of electromechanical or aerodynamic mechanical power and thrust of the UAV. With the estimated power and thrust, two controllers are proposed for the regulation of aerodynamic mechanical power and thrust. In particular, blade element momentum theory is adopted to determine λ , while equation (2.91) to compute C_T .

In [11], a dynamic inflow model in perturbation version, linearized around the hovering condition, has been adopted to study the ground effect on a quadrotor.

A quad-copter varies its thrusts by continuously changing angular speed of rotors, which can be considered rigid, due to the absence of hinges and a pitch control chain. Moreover, flapping phenomenon is just a vibratory effect and does not deal with rotor equilibrium in flight. For this reason, the most suitable approach to model inflow variation is the *dynamic inflow model* proposed in [7], due to its explicit dependency on rotor rotational speed Ω , as well as rotor velocity and incidence.

The equations presented in Section 2.4.5 are built in the fixed rotor reference frame described in Section 2.1.3.

According to the small dimensions of the quad-copter, many reasonable assumptions can be made to simplify the dynamic model. The main idea is to consider just the uniform variation of inflow due to C_T variations, because the smaller is the rotor the more negligible are the inflow lateral and longitudinal contributions.

Equations (2.99) and (2.100) show how to compute λ_i , but not how to express the forcing term, that is C_T . The most reasonable approach is to express the thrust coefficient as the resulting one from *BET* in forward flight considering a constant pitch angle, namely twist, for the blade; see equation (2.84). Finally, the dynamic inflow model ends up to be fully nonlinear.

In the following the mathematical formulation of the dynamic inflow model is presented in details. Differently from the work proposed in [7], just the uniform variation of the inflow will be addressed. Before proceeding further many quantities need to be defined.

Three components of velocity in the rotor coordinates are denoted by μ_1 , μ_2 and λ_c . The components μ_1 and μ_2 which represent the nondimensionalized forward and sideward rotor velocities, are combined to define a resultant forward velocity μ in the wind-axis system.

$$\mu = \sqrt{\mu_1^2 + \mu_2^2}. \quad (2.102)$$

Thus in the wind-axis system the rotor encounters two velocity components, μ and λ_c .

Then the total inflow through the rotor is represented by $\lambda = \lambda_c + \lambda_i$. The resultant flow through the rotor can be defined as

$$V_T = \sqrt{\lambda^2 + \mu^2}, \quad (2.103)$$

and consequently the angle of attack α_d of rotor disk plane with respect to the oncoming flow V can be defined

$$\alpha_d = \tan^{-1} \frac{\lambda_c}{\mu}. \quad (2.104)$$

The next step is to relate the wind-axis coordinates with respect to rotor ones. If Δ is the angle between rotor fixed reference frame and wind-axis coordinates, it is possible to write:

$$\sin(\Delta) = \frac{\mu_2}{\mu}, \quad (2.105)$$

$$\cos(\Delta) = \frac{\mu_1}{\mu}. \quad (2.106)$$

Referring to equation (2.97) equation, the following terms must be introduced:

- the apparent mass matrix

$$M = \begin{bmatrix} \frac{8}{3\pi} & 0 & 0 \\ 0 & \frac{16}{45\pi} & 0 \\ 0 & 0 & \frac{16}{45\pi} \end{bmatrix}, \quad (2.107)$$

- the non linear version of inflow gains matrix

$$L_{nl} = L\hat{V}^{-1}, \quad (2.108)$$

where

$$L = \begin{bmatrix} \frac{1}{2} & 0 & -\frac{15}{64\pi} \sqrt{\frac{1-\sin(\alpha)}{1+\sin(\alpha)}} \\ 0 & \frac{4}{1+\sin(\alpha)} & 0 \\ \frac{15}{64\pi} \sqrt{\frac{1-\sin(\alpha)}{1+\sin(\alpha)}} & 0 & \frac{4\sin(\alpha)}{1+\sin(\alpha)} \end{bmatrix}, \quad (2.109)$$

$$\hat{V} = \begin{bmatrix} V_T & 0 & 0 \\ 0 & V & 0 \\ 0 & 0 & V \end{bmatrix}$$

and

$$V = \frac{\mu^2 + (2\lambda_i + \lambda_c)(\lambda_i + \lambda_c)}{V_T}. \quad (2.110)$$

The mass-flow parameter matrix \hat{V} denotes a weighted downstream velocity. V_T is the total resultant flow through the disk and V is the mass-flow parameter due to cyclic disturbances. The angle α is different from the above cited α_d , as this latter is the rotor angle of attack whereas the first represents the wake angle with respect to rotor disk. α is always positive, whether the flow comes from above or below.

$$\alpha = \tan^{-1} \frac{|\lambda|}{\mu}. \quad (2.111)$$

If the matrix T is used to denote the transformation from the rotor disk plane to wind-axis, then the inflow states and the force vector can be expressed as

$$\begin{Bmatrix} C_T \\ -C_L \\ -C_M \end{Bmatrix} = T \begin{Bmatrix} C_T \\ C_1 \\ C_2 \end{Bmatrix}, \quad (2.112)$$

$$\begin{Bmatrix} v_0 \\ v_s \\ v_c \end{Bmatrix} = T \begin{Bmatrix} \lambda_0 \\ \lambda_s \\ \lambda_c \end{Bmatrix},$$

where

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta) & \sin(\Delta) \\ 0 & -\sin(\Delta) & \cos(\Delta) \end{bmatrix}. \quad (2.113)$$

Passing to fixed rotor coordinate system equation (2.99) is obtained. The non linear inflow gains matrix in rotor coordinates is

$$\begin{aligned} \hat{L}^{-1} &= \hat{V} T^T L^{-1} T \\ &= \hat{V} \mathcal{L}^{-1} \end{aligned} \quad (2.114)$$

where

$$\mathcal{L} = \begin{bmatrix} \frac{1}{2} & -B \sin(\Delta) & -B \cos(\Delta) \\ B \sin(\Delta) & E \cos^2(\Delta) + D \sin^2(\Delta) & (D - E) \sin(\Delta) \cos(\Delta) \\ B \cos(\Delta) & (D - E) \cos(\Delta) \sin(\Delta) & E \sin^2(\Delta) + E \cos^2(\Delta) \end{bmatrix}, \quad (2.115)$$

and

$$\begin{aligned} E &= \frac{4}{1 + \sin(\alpha)}, & (E - D) &= 4 \frac{1 - \sin(\alpha)}{1 + \sin(\alpha)}, \\ D &= \frac{4 \sin(\alpha)}{1 + \sin(\alpha)}, & B &= \frac{15\pi}{64} \sqrt{\frac{1 - \sin(\alpha)}{1 + \sin(\alpha)}}. \end{aligned}$$

C_1 and C_2 represent the lateral roll and longitudinal pitching coefficients respectively. Focusing on the uniform variation of the inflow, the following fully nonlinear differential equation is obtained:

$$M_{(1,1)} \dot{\lambda}_0 + \hat{L}_{(1,1)}^{-1} \lambda_0 = C_T. \quad (2.116)$$

Remembering equation (2.101), the latter dynamic model can be converted to:

$$\frac{8}{3\pi} \frac{\dot{\lambda}_0}{\omega} + V_T \left(\frac{2D}{2B^2 + D} \right) \lambda_0 = C_T. \quad (2.117)$$

where now

$$\dot{\lambda}_0 = \frac{d\lambda}{dt}.$$

If one expands thrust coefficient C_T according to BET (eq. (2.84)) a fully nonlinear dynamic model for the uniform inflow variation is obtained, i.e.:

$$\frac{8}{3\pi} \frac{\dot{\lambda}_0}{\omega} + V_T \left(\frac{2D}{2B^2 + D} \right) \lambda_0 = \frac{\sigma C_{L\alpha}}{4} \left(\frac{2}{3} \theta \left(1 + \frac{3\mu^2}{2} \right) - \lambda \right). \quad (2.118)$$

The next step is to relate the uniform variation of the inflow with the induced inflow of the rotor:

$$\lambda_i = \frac{1}{2} V_T \left(\frac{2D}{2B^2 + D} \right) \lambda_0. \quad (2.119)$$

2.5.3 Torque and drag forces

For what concerns the aerodynamic model of torque and drag forces, two models are proposed.

2.5.3.1 BET model

The first model makes advantage of the application of *BET* in forward flight. In [8], a detailed result is presented for dimensionless coefficients of rotor torque and H-force in forward flight.

According to *BET*, the blade element H-force acting on a helicopter rotor is aligned with the velocity projected on the rotor-disk plane, pointing rearward. The torque arises from rotor in-plane forces originating on each blade element. Hence they are:

$$dH = dF_x \sin(\psi) = (dD \cos(\phi) + dL \sin(\phi)) \sin(\psi), \quad (2.120)$$

$$dQ = r dF_x = r (dD \cos(\phi) + dL \sin(\phi)). \quad (2.121)$$

Expanding the latter equations, as previously done for thrust in equation (2.82), the following expressions can be derived. See [8] for clarifications.

$$H = \frac{1}{2} \rho N_b c R \Omega^2 R^2 C_{L\alpha} \left(\frac{\mu C_D}{2 C_{L\alpha}} + \frac{1}{2} \mu \lambda \theta_0 \right), \quad (2.122)$$

$$Q = \rho N_b c R \Omega^2 R^3 C_D \left(1 + 4.67 \mu^2 \right) / 8 + (T \lambda - H \mu) R. \quad (2.123)$$

and the dimensionless coefficients are, respectively:

$$C_H = \frac{H}{\rho A \Omega^2 R^2} = \frac{\sigma C_{L\alpha}}{4} \left(\frac{\mu C_D}{2 C_{L\alpha}} - \frac{1}{2} \mu \lambda \theta_0 \right), \quad (2.124)$$

$$C_Q = \frac{Q}{\rho A \Omega^2 R^3} = \sigma C_D \left(1 + 4.67 \mu^2 \right) / 8 + C_T \lambda - C_H \mu, \quad (2.125)$$

where the blade section profile drag coefficient C_D can be recovered from many references (e.g. [6] and [8]).

2.5.3.2 Lumped parameter model

The second proposed model for drag forces takes inspiration from the lumped parameter model presented in [10]. In this reference, an interesting and detailed estimation of rotor drag like effects is presented. Classical drag models presented in [6] and [8], developed for full scale rotary wing aircraft, are based on steady-state forward flight conditions and are developed out primarily with a view to computing the efficiency of flight regimes, rather than modelling system dynamics.

Let's denote the in plane velocity of the rotor as

$$V_h = (V_x \ V_y \ 0)^T. \quad (2.126)$$

Each source of drag depends on this velocity. In the following, these sources are listed:

- Induced drag is due to the backward inclination of aerodynamic force with respect to the airfoil motion,

$$D_I = -TK_I V_h; \quad (2.127)$$

- Profile drag is caused by the transverse velocity of the rotor blades as they move through the air,

$$D_P = -TK_P V_h; \quad (2.128)$$

- Translational drag originates from bending of the induced velocity stream-

tube of the airflow as it goes through the rotor during translational motion,

$$D_T = \begin{cases} TK_{T_1} V_h & \text{if } V_h \leq w \\ TK_{T_2} (V \sin(i) + u)^4 V_h & \text{if } V_h > w \end{cases}, \quad (2.129)$$

where w is a constant velocity depending on the rotor;

- A drag-like force arises from blades flapping motion. If the rotor is very stiff, the blades do not flap freely to restore equilibrium condition. Flapping motion just induces a vibratory load, which can degrade rotor performance. Hence the blade flapping drag force can be modeled as

$$D_\beta = -T \left(\frac{A_{1c}}{\omega} \right) V_h, \quad (2.130)$$

where A_{1c} is a positive scalar constant depending on rotor blades geometry.

The first two sources of drag are the ones collected in the H-force defined in *BET*. Finally, a clever consideration on the above drag like effects can be made: except blade flapping, all the other forces are negligible at low velocities.

Hence, the model for the total drag force generated by the j -th rotor is given by:

$$D_j = -A_{1c} \frac{T_j}{\omega_j} V_{hj}. \quad (2.131)$$

For that a proper estimation of A_{1c} is necessary to predict each rotor drag force correctly.

For what concerns torque, just its static relation with the thrust is adopted

$$C_Q = \frac{C_T}{\kappa}, \quad (2.132)$$

where κ is the static thrust over torque ratio. This parameter is computed in the static condition (hovering) where, from equation (2.125):

$$C_{Q_{hov}} = (C_T \lambda)_{hov} = C_{T_{hov}} \sqrt{\frac{C_{T_{hov}}}{2}} = \frac{C_{T_{hov}}^{\frac{3}{2}}}{\sqrt{2}}, \quad (2.133)$$

and

$$\kappa = \frac{1}{\lambda_{hov}} = \frac{1}{\lambda_{i_{hov}}} = \sqrt{\frac{2}{C_{T_{hov}}}}. \quad (2.134)$$

2.5.3.3 Drag coefficient estimation

In [10] the drag model adopted was not referred to a multi-body model, hence the drag was considered as a resistance force applied in quadrotor central body.

$$D = - \sum_{j=1}^4 A_{1c}^b TV_{h_j}. \quad (2.135)$$

All rotors for the symmetric quadrotor exhibit the same drag properties and see the same horizontal velocity V_h . This implies that the total drag force acting on it is:

$$D = -\bar{c}TV_h, \quad (2.136)$$

where

$$\bar{c} = 4A_{1c}^b = 0.04,$$

so

$$A_{1c}^b = 0.01.$$

Then, the drag force acting on a single rotor is

$$D_j = -A_{1c}^b TV_h. \quad (2.137)$$

Due to the fact that the quadrotor modeled in [10] was very similar to the prototype object of this modeling process, it is possible to derive the corresponding A_{1c} suitable for equation (2.131). From equations (2.137), (2.131) the following equality arises:

$$A_{1c}^b TV_h = A_{1c} \frac{T_j}{\omega_j} V_h. \quad (2.138)$$

For near hovering condition it is possible to write:

$$\begin{aligned} A_{1c} &= A_{1c}^b \frac{mg}{K_T \omega_{hov}^2} \\ &= A_{1c}^b \frac{mg}{K_T \omega_{hov}} \\ &= 4A_{1c}^b \omega_{hov} = 4A_{1c}^b \sqrt{\frac{mg/4}{K_T}} = 15.51. \end{aligned} \quad (2.139)$$

This procedure is not obviously enough to have an accurate model of rotor drag. Expanding equation (2.131) and remarking that a symmetric quadrotor exhibit the same drag properties, the total drag force can be written as:

$$\begin{aligned}
 D &= \sum_{j=1}^4 D_j \\
 &= \sum_{j=1}^4 -A_{1c} \frac{C_T \rho A R^2 \omega_j^2}{\omega_j} V_h \\
 &= A_{1c} C_T \rho A R^2 \left(\sum_{j=1}^4 \omega_j \right) V_h.
 \end{aligned} \tag{2.140}$$

The onboard accelerometer measures the specific acceleration of the vehicle with respect to the inertial frame, expressed in body frame. In other words, the accelerometer measures the sum of all the exogenous accelerations applied to the UAV. If the accelerometer measurement is:

$$a = [a_x \ a_y \ a_z]^T$$

and given that drag force D acts in the horizontal plane of the quadrotor, and $T = -ma_z$, it results:

$$a = -\frac{1}{m} \begin{pmatrix} D_x \\ D_y \\ T \end{pmatrix} \tag{2.141}$$

From the drag model in equation (2.140), decomposing the translational components of the specific acceleration, it is possible to derive A_{1c} as follows:

$$\begin{aligned}
 \frac{a_x}{a_z} = \frac{D_x}{T} &= \frac{A_{1c} C_T \rho A R^2 \left(\sum_{j=1}^4 \omega_j \right) V_{hx}}{C_T \rho A R^2 \left(\sum_{j=1}^4 \omega_j^2 \right)} = A_{1c} \frac{\sum_{j=1}^4 \omega_j}{\sum_{j=1}^4 \omega_j^2} V_{hx}, \\
 \frac{a_y}{a_z} = \frac{D_y}{T} &= \frac{A_{1c} C_T \rho A R^2 \left(\sum_{j=1}^4 \omega_j \right) V_{hy}}{C_T \rho A R^2 \left(\sum_{j=1}^4 \omega_j^2 \right)} = A_{1c} \frac{\sum_{j=1}^4 \omega_j}{\sum_{j=1}^4 \omega_j^2} V_{hy}.
 \end{aligned} \tag{2.142}$$

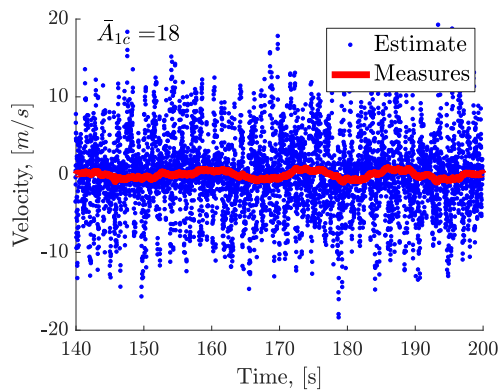
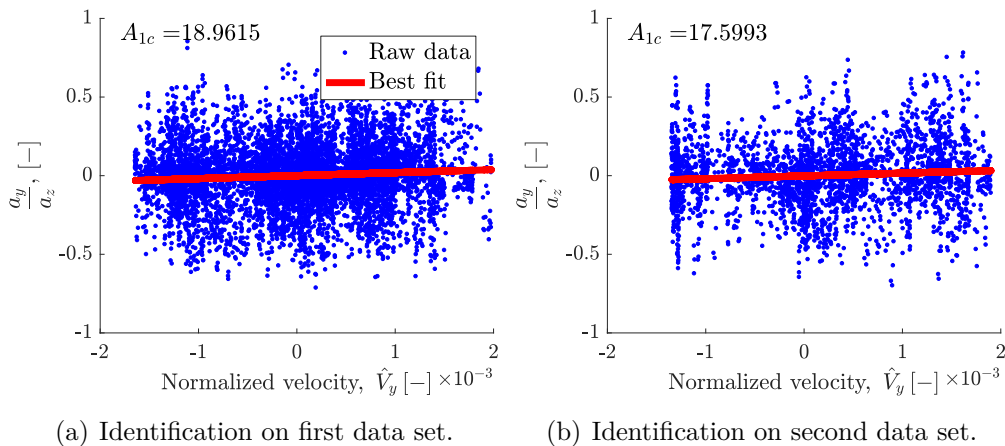


Figure 2.8: Drag coefficient estimation.

Using the vehicle velocity measurements V_m along with each motor-rotor angular speeds ω_j , a normalized velocity \hat{V} can be defined as

$$\hat{V} = \frac{\sum_{j=1}^4 \omega_j}{\sum_{j=1}^4 \omega_j^2} V_m.$$

Thanks to IMU measurements a , the drag coefficient A_{1c} can be determined. The quadrotor was asked to move left and right at constant altitude during the flight test, in order to cover a suitable range of horizontal velocities.

The result of the regressions is shown in Figures 2.8(a) and 2.8(b), where two different flight data sets are employed.

It is possible to see that the estimated lumped parameter is not so much different from the one derived analytically. Moreover, a cross validation on a third data set is performed using the mean of the coefficients previously estimated. See Figure 2.8(c).

The velocity estimates are not very close to the measured ones only due to the fact that IMU data, velocity measurements and propellers angular speed are very noisy. Nevertheless, the drag coefficient estimation can be acceptable, although better results could have been achieved with better sensors.

In the next section, where the implementation of the quadrotor model will be discussed, the drag force estimated will be always named H-force, in order to avoid misunderstanding in the developing of the model.

2.5.4 Implementation

As stated in Section 2.5.1 the advantage of using Modelica MultiBody library is that the elementary models can be connected through their ports. Moreover, the Modelica library facilitate model reusability and standardization. Thus the the rotor arm interface along with all its components were developed just once. This subsystem was just reused for each of the four rotor arms, just properly linking them to the main body.

In Figure 2.9, the multi-body implementation of the model for a single rotor arm is illustrated. The inputs of the `inflow1` block are the quadrotor angular rates ω_b and the linear velocity expressed in the body frame v_b , depicted in figure with `angular_rates_body` and `vel_body` respectively.

Through the kinematic law

$$v_r = v_b + \omega_b \times l_{arm} e_{1R}^{CG}$$

the velocity of each rotor is calculated, where e_{1R}^{CG} is oriented like e_{1R} but centered in the CG of the quadrotor.

The first order dynamics of the electric motor driving the propeller is modeled

with a transfer function with a single pole and unitary gain

$$\omega(s) = G(s)u(s) \quad (2.143)$$

$$= \frac{1}{1 + s\tau_{mot}}u(s) \quad (2.144)$$

where τ_{mot} is the time constant related to it, which has been derived experimentally, and $u(s)$ is the required propeller angular speed (see equation (2.45)). Then, the rotational speed generated by the electric motor drives the rotor through a revolute joint constraint. Lastly, the rotor is modeled as a cylinder too, with an almost negligible thickness and radius R . The chain from `frame_a` to the revolute joint represents the multi-body implementation of the arm, linking the main body with the rotor.

Aerodynamic modeling of the rotor is implemented in the `inflow1` block, while in `load` thrust, torque and H-force are calculated from their respective dimensionless coefficients. From the aerodynamic point of view, the only known quantity, as it is clear from table, is the hovering thrust coefficient K_T at first. Recalling equations (2.41) and (2.65), assuming the classical value for air density, *i.e.* $\rho = 1.225 \text{ kg/m}^3$, it turns out that

$$C_{Thov} = \frac{K_T}{\rho\pi R^4} = \frac{K_{Thov}}{\rho\pi R^4} \approx 0.01186, \quad (2.145)$$

$$\lambda_{hov} = \sqrt{\frac{C_{Thov}}{2}} = 0.077. \quad (2.146)$$

According to these results it is easy to estimate a very important parameter needed for the correct model validation, θ_0 . Remembering equation (2.84), in hovering condition, *i.e.*,

$$\begin{aligned} \mu &= 0, \\ \lambda &= \lambda_{hov}, \end{aligned}$$

it can be obtained

$$\theta_0 = \frac{3}{2} \left(\lambda_{hov} + \frac{4C_T}{\sigma C_{L\alpha}} \right) = 0.2484 \text{ rad} \approx 14^\circ. \quad (2.147)$$

Even if this twist angle is almost unfeasible on a real blade, it is the proper

parameter to match equation (2.84) with the value found in equation (2.145). In Table 2.2 all the necessary quantities to build the dynamic inflow model are listed.

As stated in Subsection 2.5.3, within the aerodynamic model two different proposals for drag and torque model may be implemented. Hereinafter, H-force will refer to the rotor drag force, whatever drag model is implemented (BET or lumped parameter), as well as its dimensionless coefficients.

The outputs of `inflow1` block are:

- Thrust coefficient, C_T ;
- Torque coefficient, C_Q ;
- H-force coefficients, C_{H_x} and C_{H_y} ;

where C_{H_x} and C_{H_y} are the components of C_H along e_{1_R} and e_{2_R} directions in the rotor coordinate system. In the same manner, also the advance ratio μ is split in μ_x and μ_y .

Now, the outputs of the quad-copter model include not only its states but also the relevant aerodynamic entities for each rotor too, namely:

- λ and λ_i ;
- C_T and thrust T ;
- C_H and H-force components H_x , H_y ;
- C_Q and torque Q .

Name	Description	Value	Unit of measure
N_b	Number of rotor blades	2	[–]
σ	Solidity ratio	0.0852	[–]
ρ	Air density	1.225	kg/m^3
C_D	Profile drag coefficient	0.012	[–]
A_{1c}	Lumped-parameter drag coefficient	18	[–]
$C_{L\alpha}$	Lift slope coefficient	2π	$1/rad$
θ_0	Blade pitch angle	14	deg
κ	Thrust over torque ratio	12.9870	[–]
τ_{mot}	Motor time constant	0.055257	rad/s

Table 2.2: Arm parameters.

Recalling equation (2.35), the definition of forces and moments applied to the UAV (see Section 2.3.5) must be extended to include each rotor H-forces and their contribution to the overall torque.

Since the quadrotor has an X -configuration, if one refers to Figure 2.1, the directions of the arms are enclosed in a vector

$$\gamma = [135^\circ \quad 45^\circ \quad 45^\circ \quad 135^\circ]^T$$

and so, according to equations (2.4) and (2.5), the rotation matrix which transfers each rotor H-force into main body axes is

$$R_{arm}^j = R_z^T(\gamma(j)) = \begin{bmatrix} \cos(\gamma(j)) & \sin(\gamma(j)) & 0 \\ -\sin(\gamma(j)) & \cos(\gamma(j)) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $j \in [1; 4]$ refers to the j -th arm. It follows that

$$F_{props} = \sum_{j=1}^4 (R_{arm}^j F_{prop}^j) = \sum_{j=1}^4 \left(R_{arm}^j \begin{bmatrix} H_x \\ H_y \\ T \end{bmatrix}_j \right), \quad (2.148)$$

$$M_{props} = \sum_{j=1}^4 (R_{arm}^j M_{prop}^j) = \sum_{j=1}^4 R_{arm}^j \left([l_{arm} \quad 0 \quad 0] \wedge F_{prop}^j + \begin{bmatrix} 0 \\ 0 \\ Q_j \end{bmatrix} \right). \quad (2.149)$$

expressed in the main body frame. Figure 2.10 illustrates the quadrotor Multi-Body model.

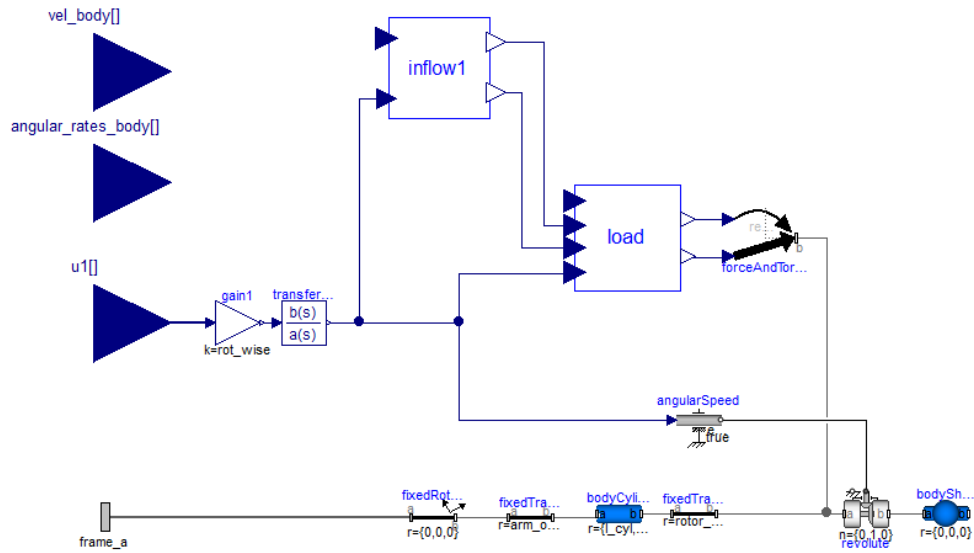


Figure 2.9: Arm model overview.

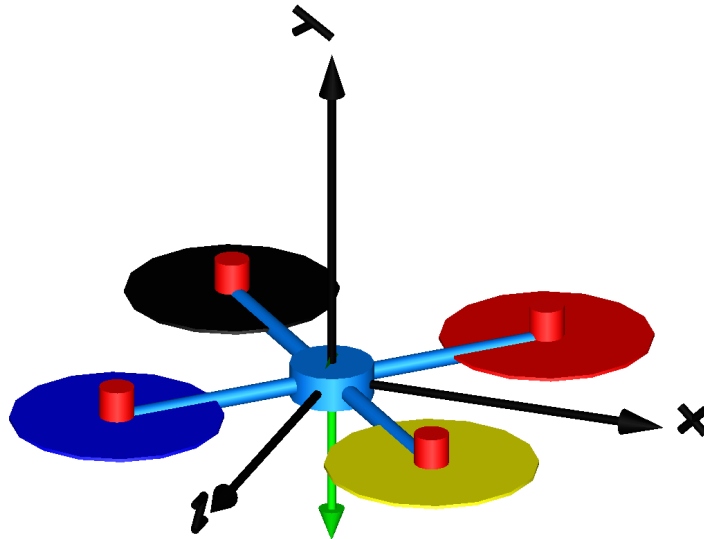


Figure 2.10: Quadrotor multi-body model.

2.6 Simulation

2.6.1 Dymola-Simulink co-simulation

The multi-body model of the quadrotor illustrated in Section 2.5 is not provided with a control algorithm. It must be intended that a multirotor may be manually controlled by a pilot, or fly autonomously when it is commanded from a ground station, sending appropriate references. The autonomous flight is definitively the hardest challenge in the UAV field. However, in both cases, the vehicle must embed a stable control law on the onboard Flight Control Unit. Once the control module has been designed, it must be written in a suitable code using procedural languages such as C or FORTRAN, to be implemented in the FCU. Procedural control algorithms, *e.g.*, written as C or MATLAB [12] code, allow the designer to reuse a wide range of available specific algorithms and routines he/she is confident with, and also can be directly implemented in the FCU.

Thus, the multi-body model of the quadrotor hardware can easily be coupled with the Simulink [4] model of the control law, along with the state feedback procedure ideally provided by sensors, without going through all the trouble of re-implementing them as Modelica code. Exploiting this flexibility in order to build simulation models of increasing complexity and accuracy, it was convenient to perform a *co-simulation* in a *Dymola-Simulink* environment. The co-simulation benefits from all the advantages of the multi-body model implemented in Dymola 2018 [3] along with MATLAB and Simulink versatility. This means that the quadcopter multi-body model is translated into a Dymola block in a quadrotor simulator implemented in Simulink.

The outputs of the multi-body model are converted in the proper systems of coordinates. Position, velocity and attitude are expressed with respect to \mathcal{F}_E reference frame, while angular rates refer to a \mathcal{F}_B body frame.

As can be seen in Figure 2.11, the model is comprehensive of:

- Quadcopter: this block contains the dynamic model implemented in Dymola 2018 [3]. It receives as input the throttle percentage required to each motor-rotor coupling, $Th\%_i$. The outputs of this block are the state vector and the aerodynamic quantities.
- Set-point: this block generates the reference trajectory that the quadrotor

is supposed to follow. The set-point specifies not only the required position and attitude, but also velocity, acceleration, jerk, angular rates, angular accelerations and angular jerks.

- Mixer: this subsystem contains the mixer matrix χ , hence the control actions are translated into required motor-rotor rotational speeds.
- Controller: it includes a suitable control law to make the quadrotor capable of tracking the given set point. It receives as inputs the state estimate and produces as outputs the four control variables $[T L M N]^T$. This block encloses the geometric controller presented in [13].
- Estimate: this block reads the output of the quad-copter block and transforms the signals by discretizing and delaying them. This is a way to take into account the sampling time of an hypothetical hardware, on which the control system should be implemented. The considered working frequency is $100Hz$.

Recalling equation (2.44), the K_T and K_Q parameters are just due to static C_T and C_Q , hence in hovering condition. See equations (2.145), (2.133). Thence, the parameters in the mixer matrix are:

$$K_T = C_{Thov} \rho A R^2, \quad (2.150)$$

$$K_Q = \left(\frac{C_{Thov}^{\frac{3}{2}}}{\sqrt{2}} \right) \rho A R^3. \quad (2.151)$$

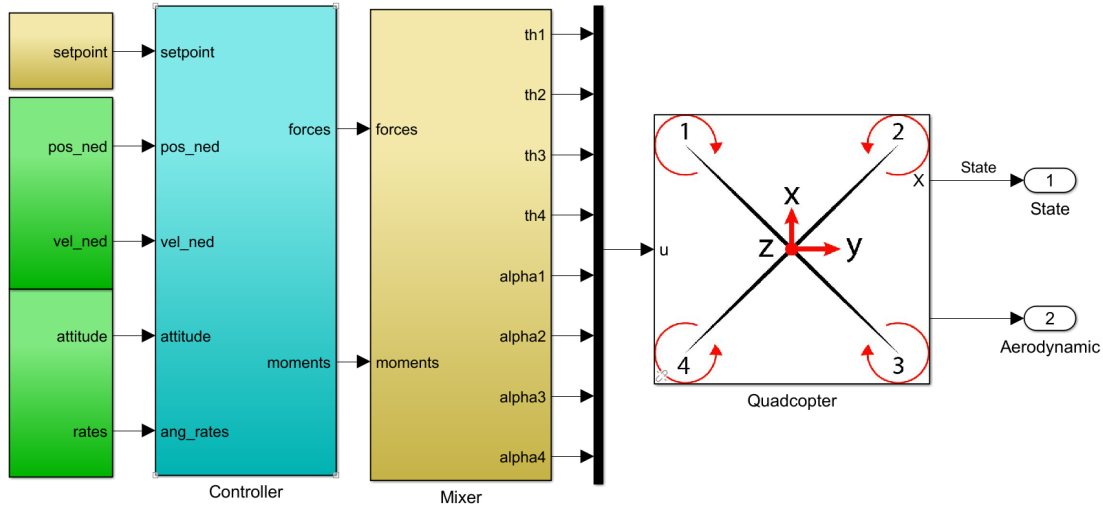


Figure 2.11: Quadrotor simulator.

2.6.2 Results

In this section, the simulation results of the co-simulation will be presented, making use of both aerodynamic models for drag forces. The task performed is a trajectory tracking and all the aerodynamic quantities will be estimated during the whole simulated flight. It will be shown that tracking capability of the quadrotor is affected by the unsteady aerodynamic variations of thrust, drag and torque.

The set-point is a smooth function of time, defined as:

$$\left[x_d \quad v_d \quad a_d \quad j_d \quad R_d \quad \omega_d \quad \dot{\omega}_d \quad \ddot{\omega}_d \right]^T \quad (2.152)$$

where:

- x_d, v_d, a_d and j_d are the linear reference quantities desired: position, velocity, acceleration and jerk respectively.
- $R_d, \omega_d, \dot{\omega}_d$ and $\ddot{\omega}_d$ instead form the angular references sent to the drone: attitude matrix, angular velocity, angular acceleration and angular jerk.

Thanks to the use of degree five polynomials, the overall desired position and attitude could be defined combining several trajectories, specifying for each one its initial and final positions, velocities, accelerations and attitudes, angular speeds and angular accelerations. This lead to a set-point with continuous accelerations and angular accelerations.

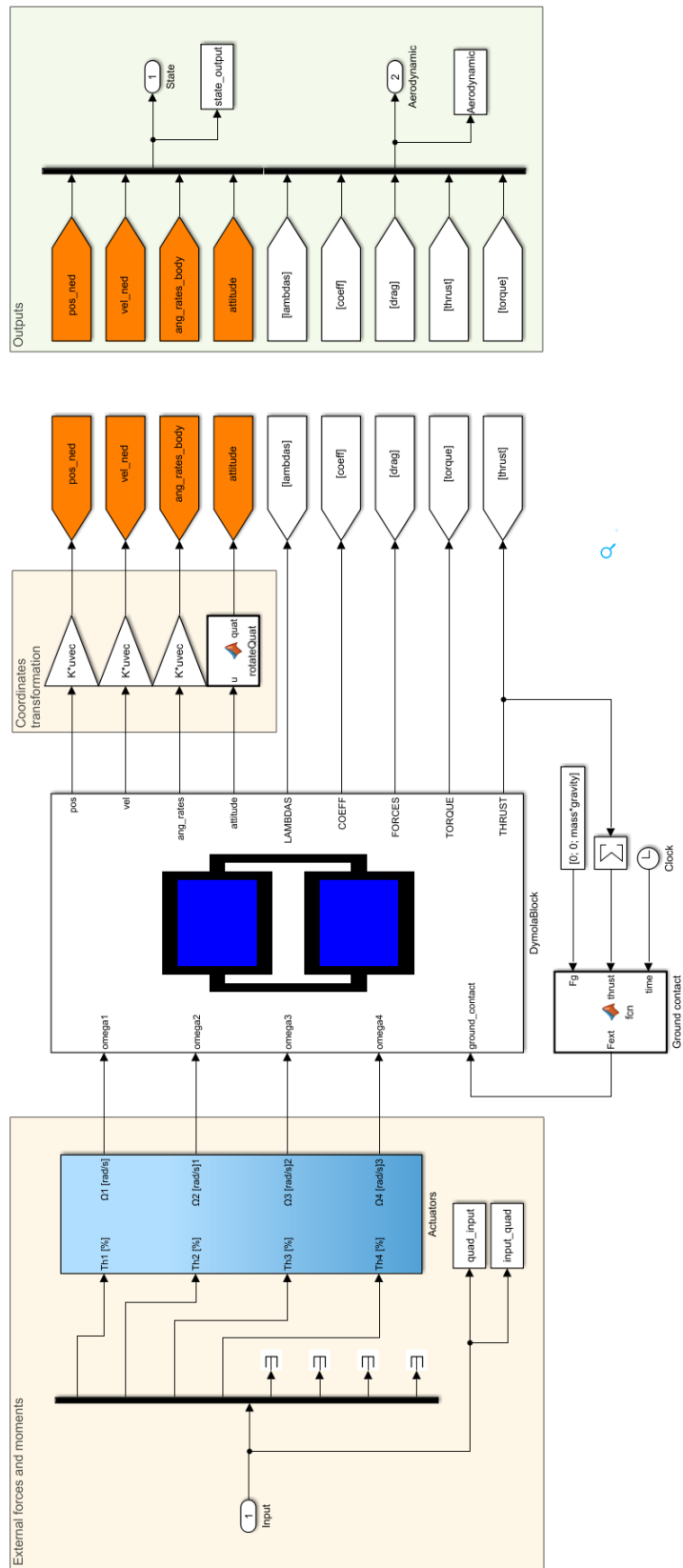


Figure 2.12: Quadcopter subsystem.

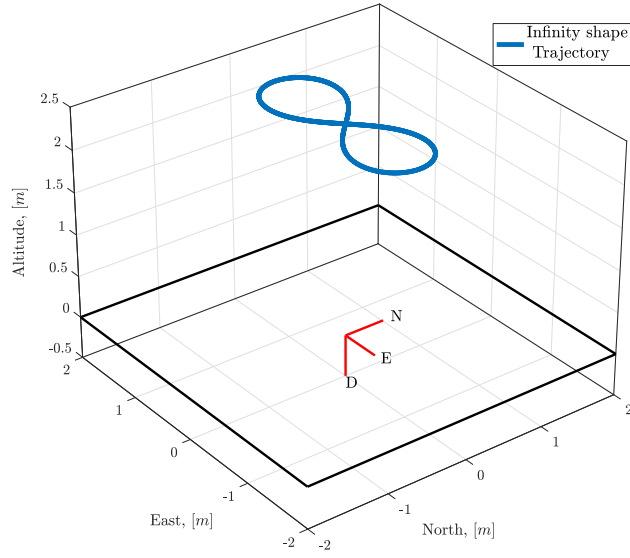


Figure 2.13: Infinity trajectory.

Infinity-shape trajectory

Initially the trajectory prescribes the take-off maneuver from the ground, moving the quad-rotor from a generic point on the ground with non-null yaw angle to the hovering point in $p = [0, 0, h]^T$ with null attitude. Then, at time $t = 20s$ the quadrotor is requested to move to the starting point of the infinity-shape trajectory. At time $t = 25s$ the maneuver starts:

$$x_d = \begin{bmatrix} \frac{\sin(2\omega t)}{3 - \cos(2\omega t)} \\ \frac{\cos(2\omega t)}{3 - \cos(2\omega t)} \\ h \end{bmatrix}. \quad (2.153)$$

where t is the simulation time, ω is the angular frequency of the maneuver and h is the altitude at which the infinity-shape trajectory is performed. The velocity, acceleration and jerk set-points v_d , a_d and j_d are obtained differentiating with respect to time equation (2.153). In this case $h = -2.5m$. In Figure 2.13, the planar part of the trajectory is plotted. Besides the position set-point, the drone is required also to keep the roll and pitch angles equal to zero and to change yaw angle to maintain the e_{1B} direction tangent to the trajectory. This is achieved by

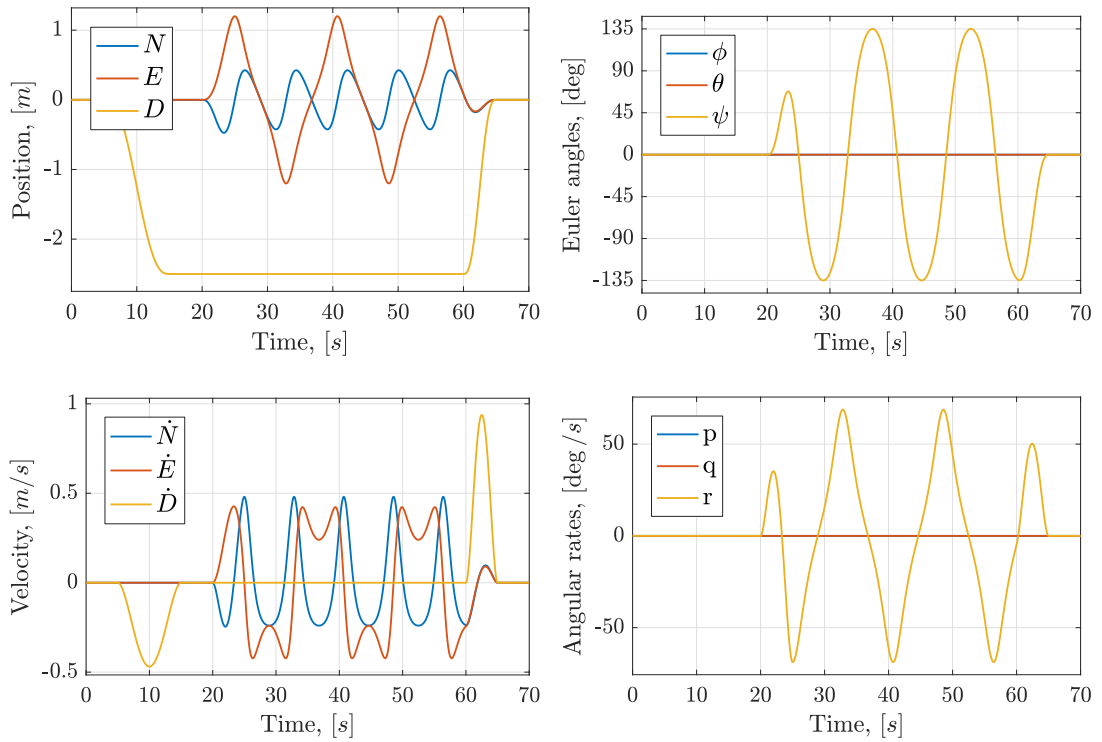


Figure 2.14: Trajectory set-point.

imposing:

$$\psi_d = \arctan(v_d(2), v_d(1)), \quad (2.154)$$

$$\dot{\psi}_d = \frac{a_d(2)v_d(1) - a_d(1)v_d(2)}{v_d(1)^2 + v_d(2)^2}.$$

In Figure 2.14, the main references of the infinity-shape trajectory are shown.

BET based model

The model has been described in Section 2.5.3.1.

Figure 2.15 shows position and velocity errors, while in Figure 2.16 attitude and angular rates errors are depicted. It is worth noticing that these errors refer to the control errors, hence the difference between the quadrotor state and the set-point. For what concerns aerodynamics, Figure 2.17 shows the induced inflow ratio and thrust coefficients, whereas Figure 2.18 depicts H-force and torque coefficients. Finally, Figures 2.19 and 2.20 illustrate the H-force in-plane components for the four rotors and the throttle percentages required to the motor-rotor system.

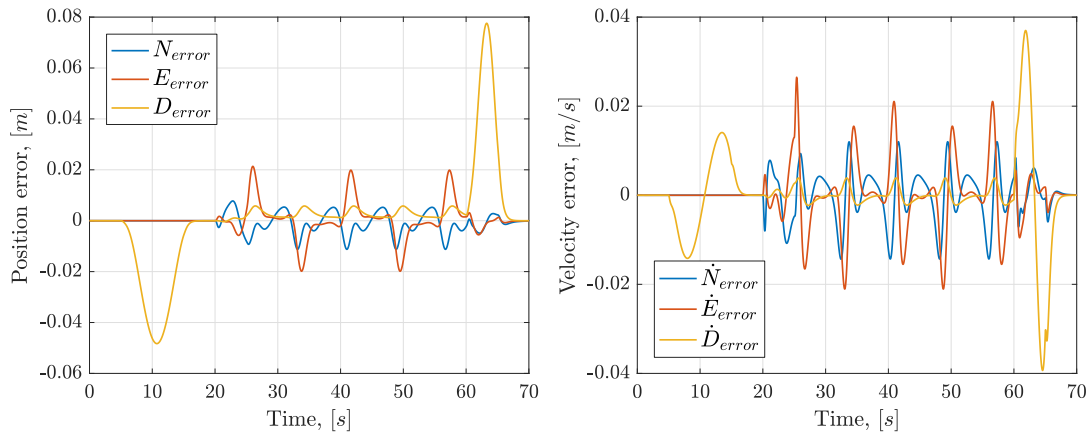


Figure 2.15: Quadrotor position and velocity error (BET).

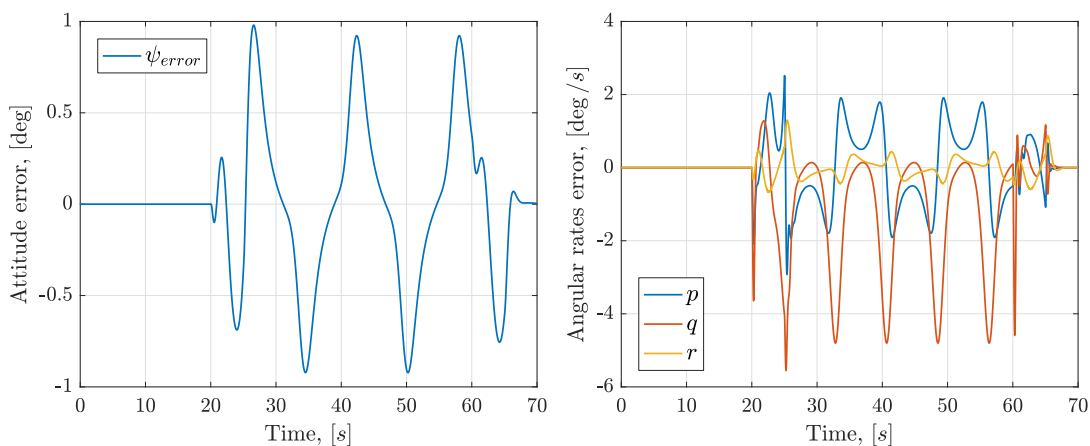


Figure 2.16: Quadrotor attitude and angular rates error (BET).

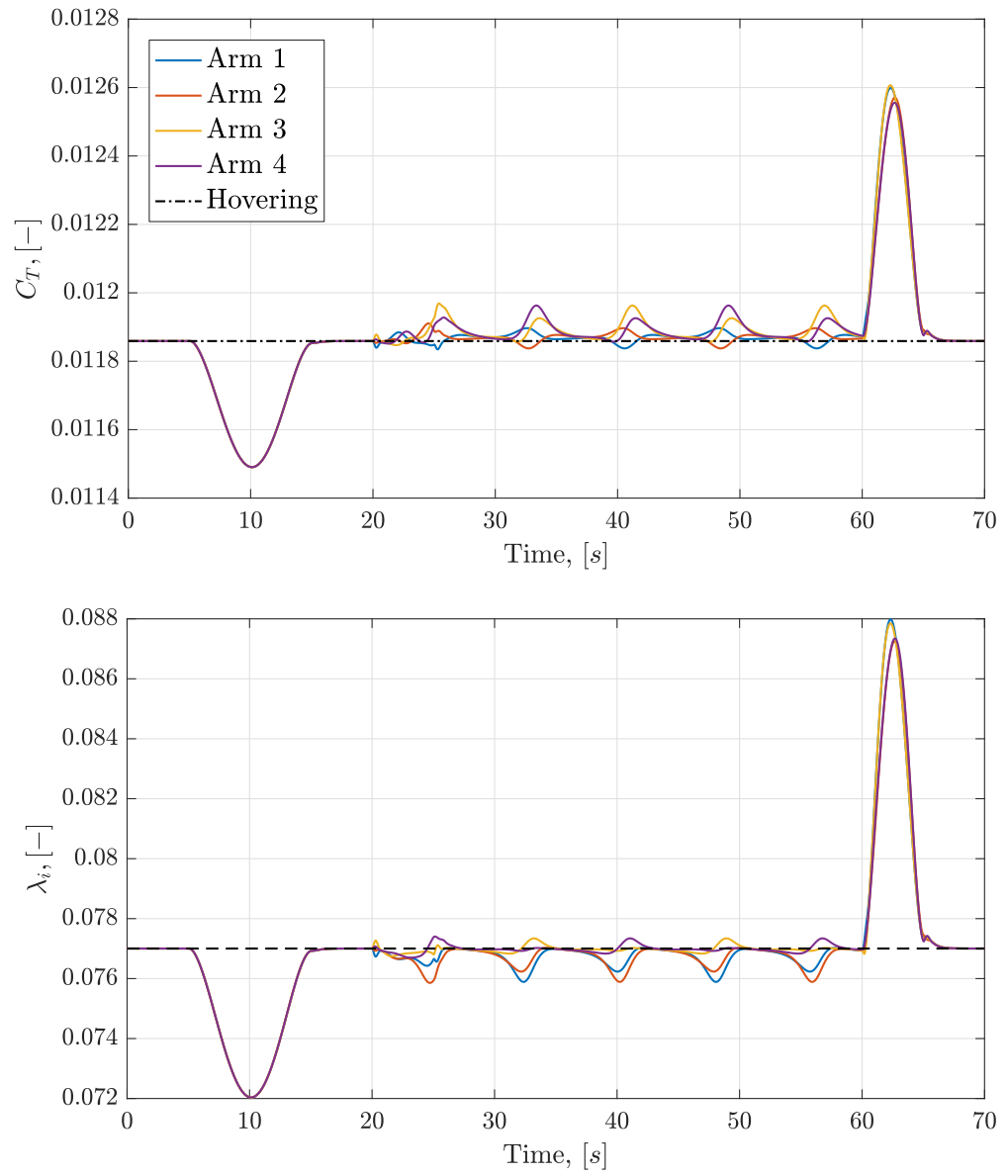


Figure 2.17: Quadrotor thrust coefficient and induced inflow ratio (BET).

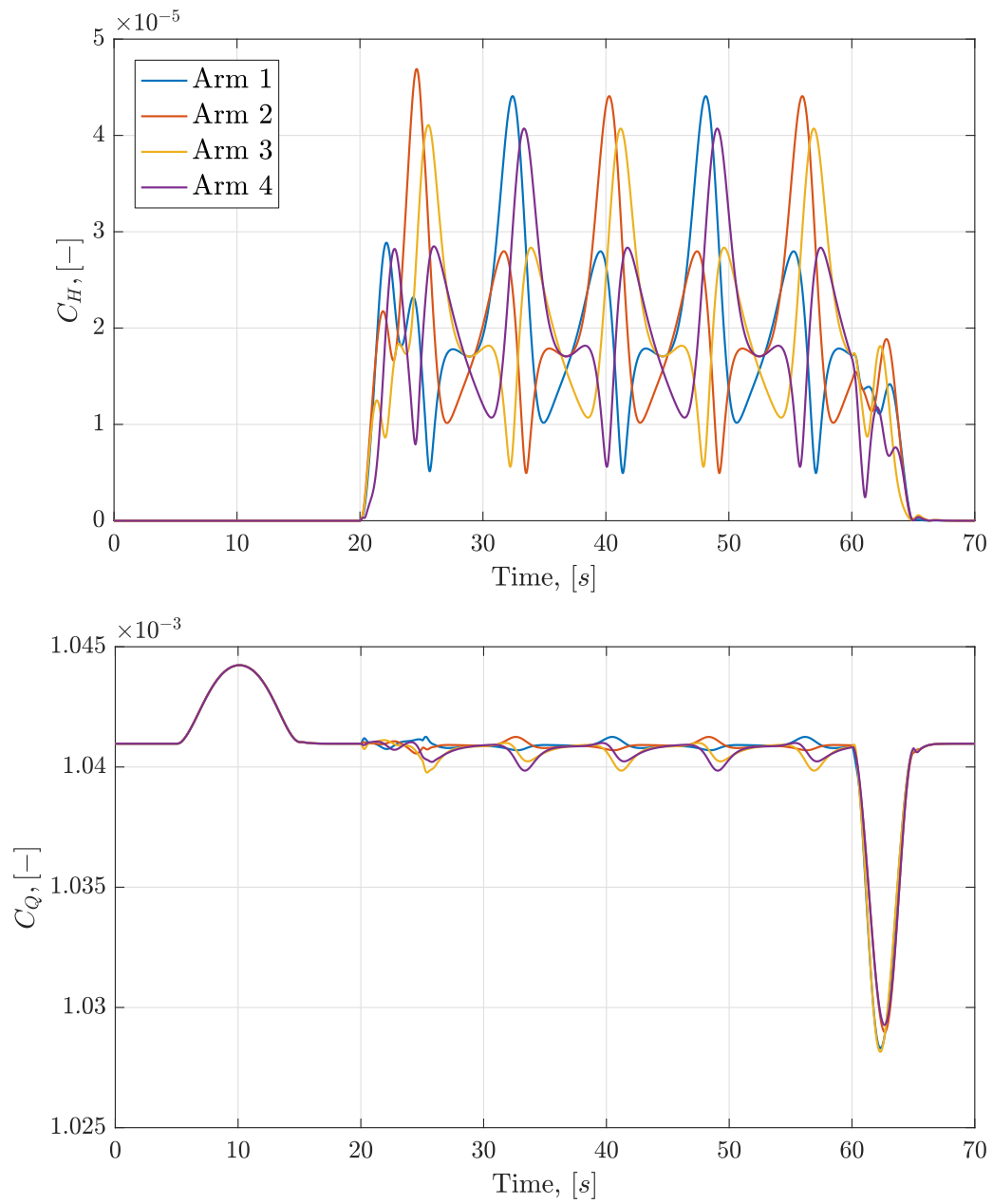


Figure 2.18: Quadrotor H-force and torque dimensionless coefficients (BET).

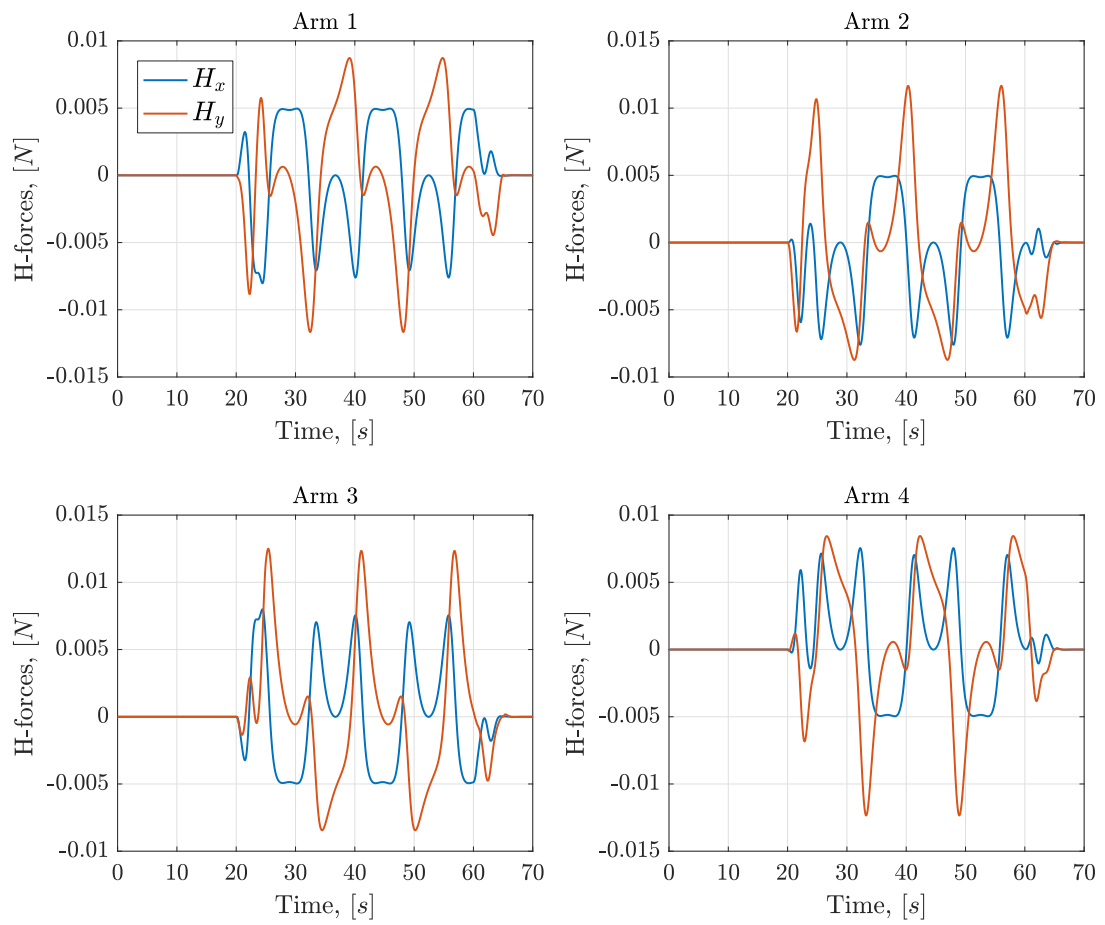


Figure 2.19: Quadrotor H-forces (BET).

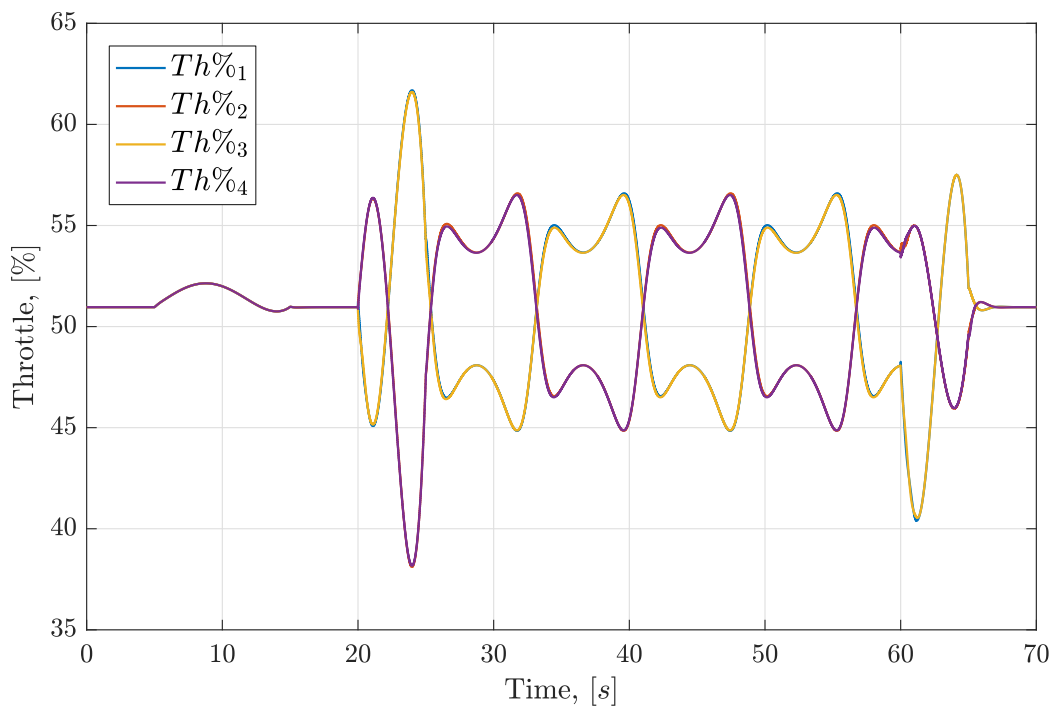


Figure 2.20: Quadrotor throttle percentage (BET).

Lumped parameter model

The simulation results of the quadrotor model provided with the lumped parameter drag model are shown in the following. As done in the previous paragraph, Figure 2.21 shows position and velocity errors, while Figure 2.22 attitude and angular rates errors.

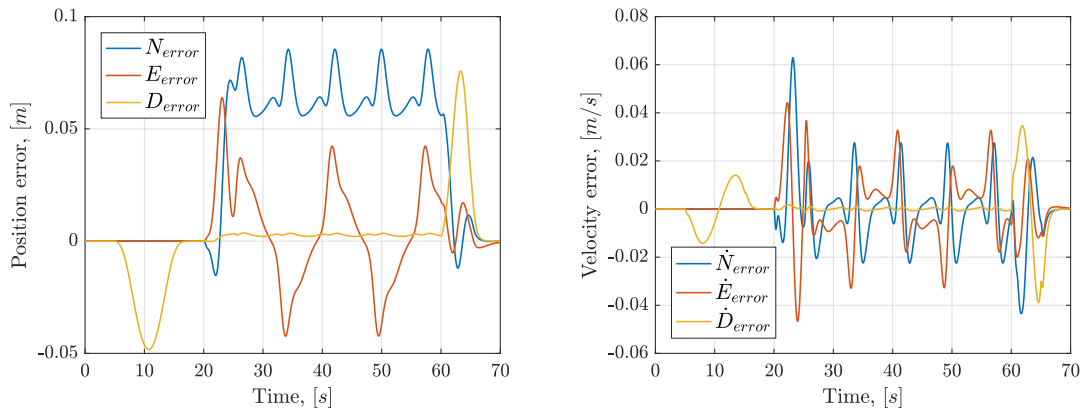


Figure 2.21: Quadrotor position and velocity error (lumped parameter).

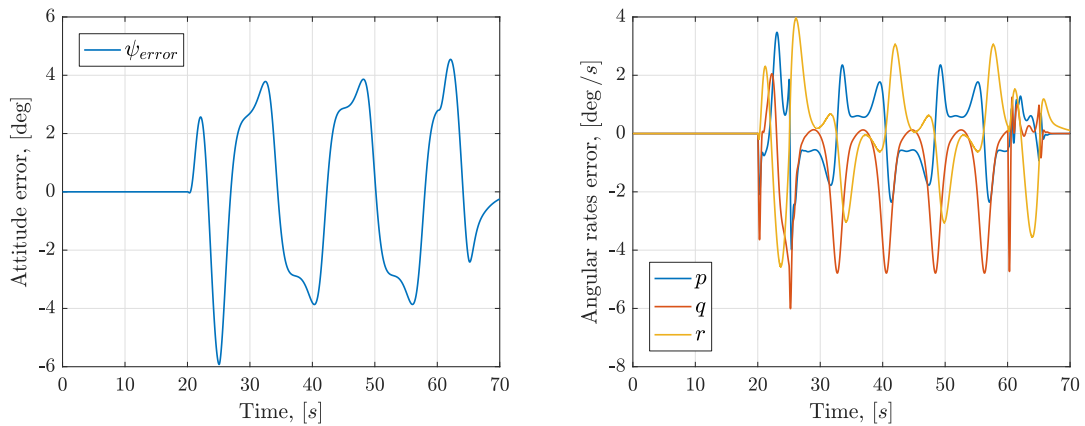


Figure 2.22: Quadrotor attitude and angular rates error (lumped parameter).

And again, Figures 2.23 and 2.24 show the aerodynamic quantities, inflow and coefficients. Lastly, Figure 2.25 illustrate the in-plane components of the drag force on the four rotors. Throttle percentages of the four motor-rotor systems are depicted in Figure 2.26.

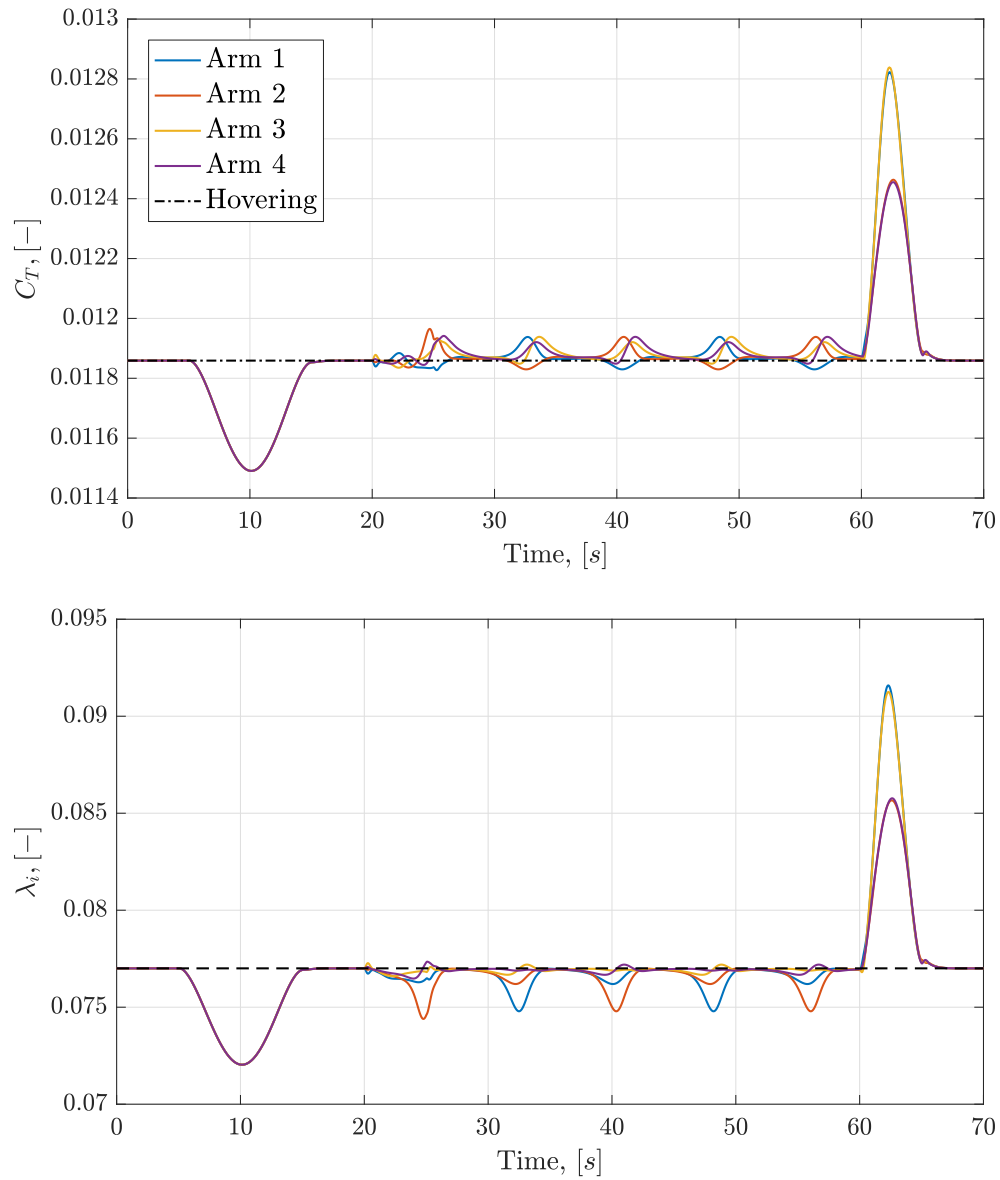


Figure 2.23: Quadrotor thrust coefficient and induced inflow ratio (lumped parameter).

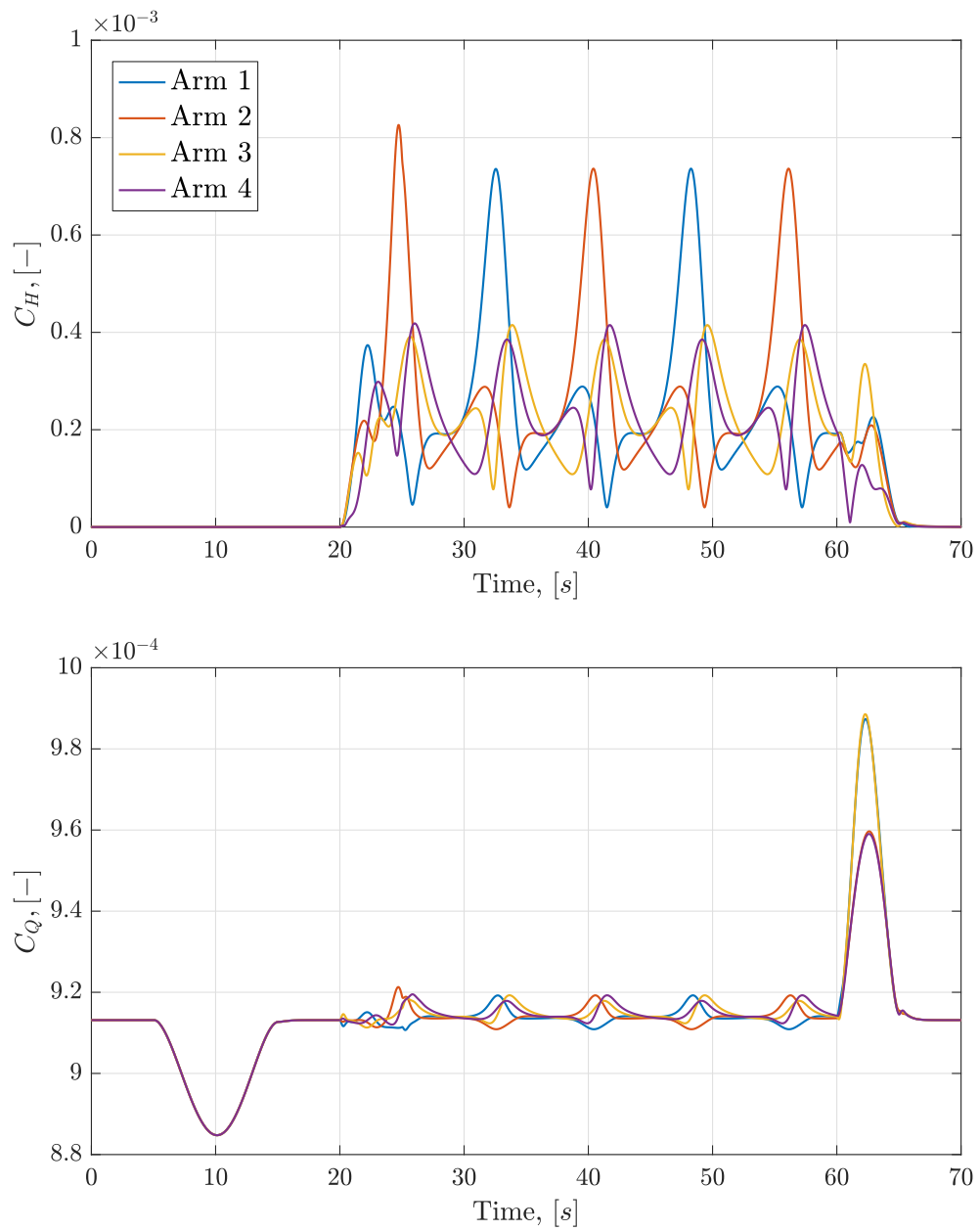


Figure 2.24: Quadrotor H-force and torque dimensionless coefficients (lumped parameter).

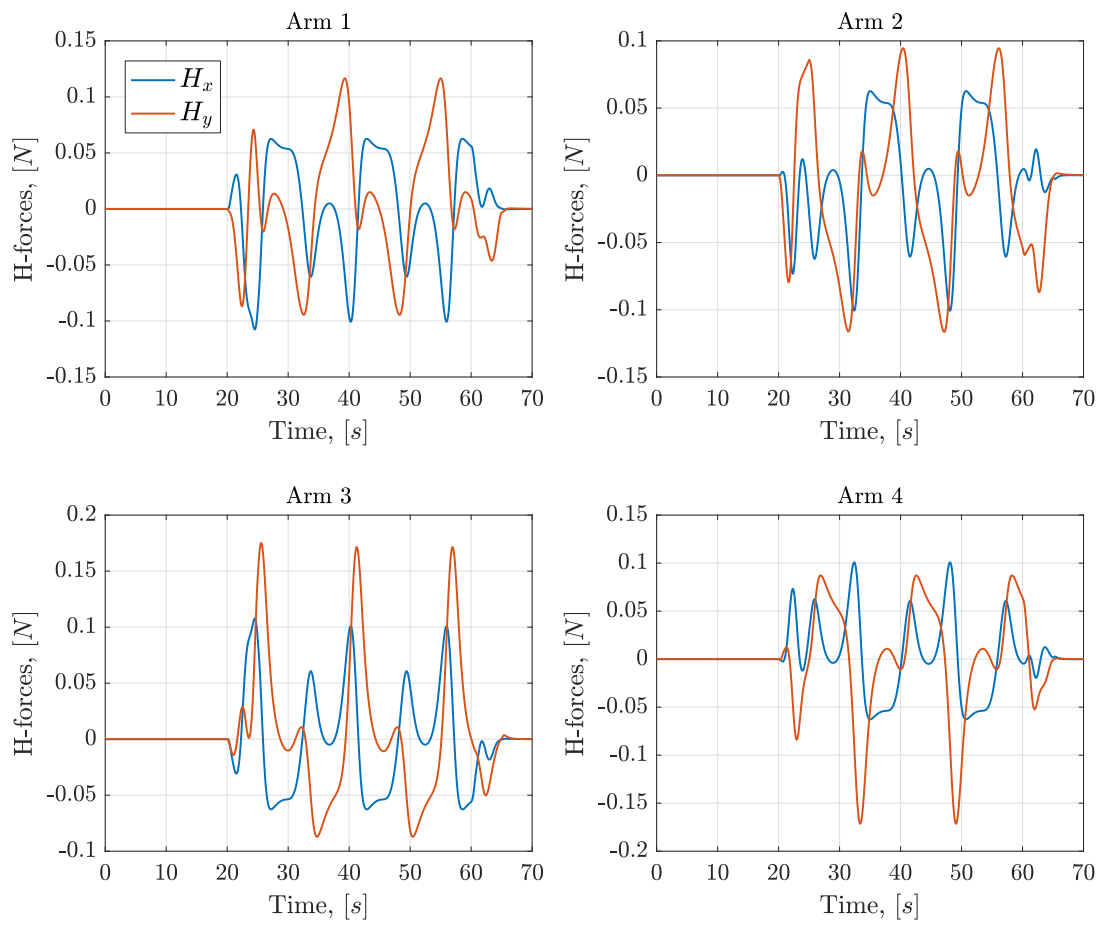


Figure 2.25: Quadrotor H-forces (lumped parameter).

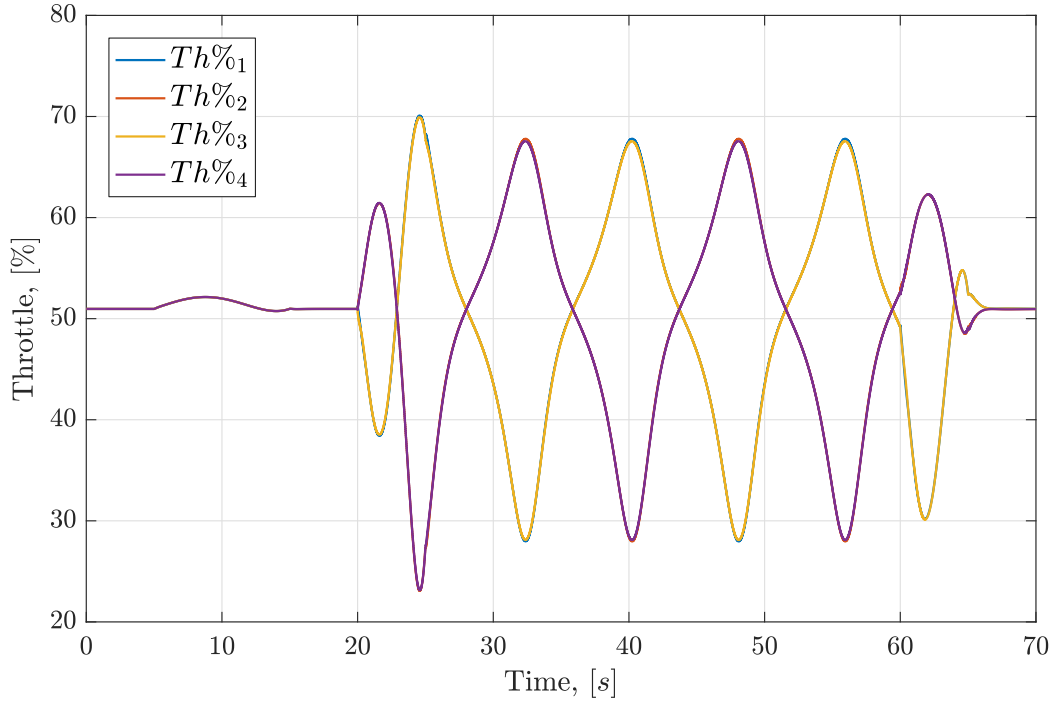


Figure 2.26: Quadrotor throttle percentage (lumped parameter).

2.6.3 Conclusions

It is clear that the controller aims at minimizing the errors between quadrotor states and its reference set-points. Focusing on Figures 2.20 and 2.26 the differences in throttle percentage commands between the two models is evident. This is mostly due to the fact that, as depicted in Figures 2.19 and 2.25, the resistance forces opposing quadrotor motion are larger for the drag lumped parameter model than the BET one. This is clearly remarked in Figures 2.18 and 2.24 where the difference in C_H is about one order of magnitude. It is possible to state that including blade flapping phenomena as a resistant load has a strong effect on quadrotor performances. This difference in drag forces influence also the tracking capability of the quadrotor. During the infinity trajectory each rotor tries to self-equilibrate itself through blade flapping phenomena, but this never happens due to rotor rigidity. For this reasons, an higher resistance contribution arises, reflecting in higher position and attitude errors especially in the curvilinear stretch of the trajectory. See Figures 2.21 and 2.22. The higher error is on the north position, namely the coordinate along which the tighter turn has to be performed. Lastly, the variation of λ_i , and consequently of C_T and C_Q , is significant during

climb and descent phases, while it is almost negligible during in-plane motion except for the turning points. In these precise stretches a consistent variation of the thrust coefficient is experienced, mostly due to the combined effects of drag forces and advance ratio μ on C_T (see equation (2.84)).

Moreover, the torque coefficient has different behavior according to the adopted model (see Figure 2.18 and Figure 2.17). Focusing on the take-off and landing maneuvers, the BET based model provides an opposite variation of the torque with respect to C_T and λ ones, even if it is due to the combination of these latter. Instead, looking at Figure 2.24 and Figure 2.23 C_Q is smaller than the one predicted by BET model and has obviously the same behavior of thrust coefficient (see equation (2.132)).

2.7 Experimental results

In this section both quadrotor models are compared to flight data. The quadrotor vehicle, object of the modeling process, is asked to perform the same maneuvers designed in simulation environment.

The reference trajectories made in the flight tests are:

- Infinity-shape trajectory illustrated in details in Subsection 2.6.2,
- Step maneuver, consisting in an sudden change of the horizontal position set-point.

The first trajectory consist of continuous and differentiable set of smooth references, while the step maneuver does not ensure these properties.

The errors between state and references will be analyzed along all the time history of the flight tests, focusing on position and attitude errors. Then, they are compared to the corresponding simulation results in order to show which model, BET or lumped parameter, predicts the real flight performance better.

It will be also show that using a quadrotor simulator entirely developed in Simulink [4], devoid of any aerodynamic model, which will be referred as *Original simulator*, it is not possible to estimate real performance at all.

2.7.1 Quadrotor prototype

Here it is described only the specific hardware of the quadcopter prototype; for what concern the whole hardware and software see Section 6.1. The drone is a quadrotor in X-configuration, *i.e.* the principal axes of the body frame are not aligned with the rotor arms, see Figure 2.27. The frame is a Talon V2.0 (HobbyKing), made of carbon fibre tubes and aluminium parts. A detailed description of quadcopter onboard systems is provided in the following paragraphs, while the main parameters of the drone are reported in Table 2.3.

Companion

It is equipped with Raspberry Pi3 [14], whose specifications are:

- Processor: Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- Memory: 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) onboard
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera connector
- DSI display port
- Micro SD port
- Upgraded switched Micro USB power source up to 2.5A



Figure 2.27: Quadrotor prototype.

Item	Value	Unit of Measure
MTOW	1.51	kg
Rotor Radius	152.4	mm
Frame Diagonal	550	mm

Table 2.3: Quadrotor model main parameters.

Flight Control Unit

The FCU mounted on this UAV is the R2P (Rapid Robot Prototyping) [15]. It is an open source HW/SW framework that allows to implement real time architectures. Another feature is that is composed by modules, to be precise four in this case: USB, RC, IMU and proximity. For further details, see [16] and [17].

2.7.2 Model comparison

For what concerns the infinity trajectory, the performance index chosen to compare the models and the flight data is the evaluation of the control error between the quadcopter state and the set-point illustrated in Figure 2.14. Focusing on position and attitude errors, the norm of the position errors and the yaw angle error are the comparison indexes. The maximum and the mean values of these latter are evaluated for all the models in simulation environment, and subsequently compared to flight data.

Due to the fact that drag force always opposes to the motion, the response to a step position set-point may be different according to the adopted drag model. With this in mind, the response of the models can be compared to the flight data

when the quadrotor is asked to perform the maneuver.

Infinity trajectory

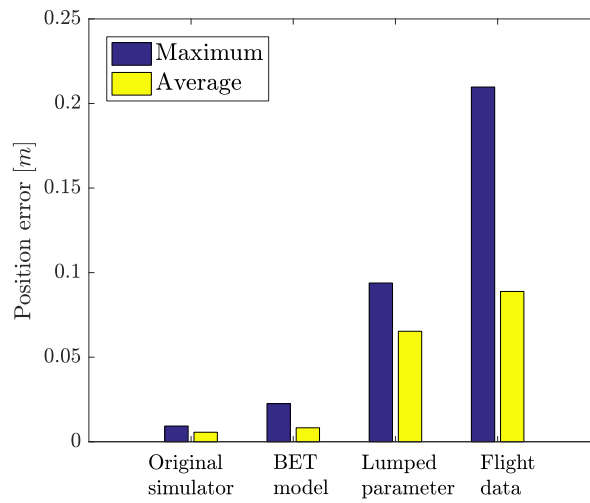
In Figure 2.28(a), the histograms for position error norm are shown, while in Figure 2.28(c) the same quantities are reported in percentage with respect to flight data values. Lastly, in Figure 2.28(b) the mean and maximum of yaw error is plotted.

Step maneuver

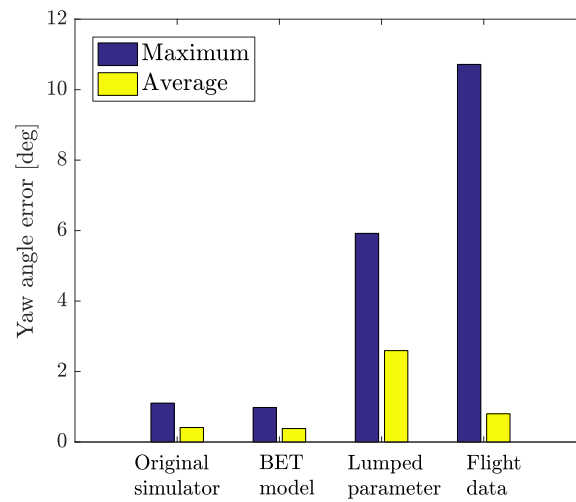
In Figure 2.29, the time responses of the quadrotor to the step set-point are reported. The maneuver is performed at a constant height.

2.7.3 Conclusions

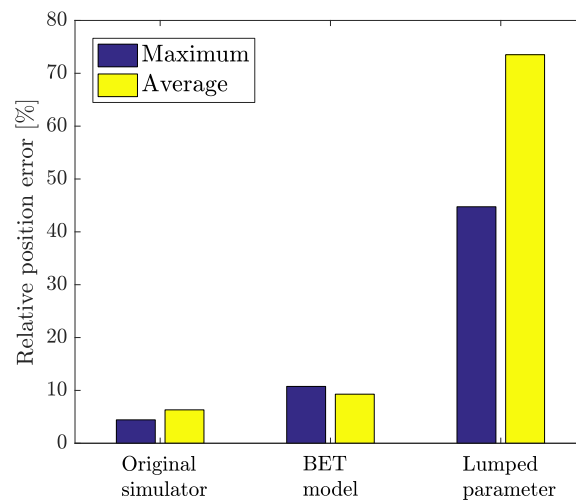
The importance of including the rotor unsteady aerodynamics in the quadcopter model is proved in Figure 2.28. Specifically, Figure 2.28(c) shows the relative percentage error between simulation results and flight data, in terms of the norm of position errors. From the original simulator to the lumped parameter one there is a clear tendency to get closer to real flight data. With a lumped parameter model, based on the experimental observations of drag effects, the mean of the norm of the position control errors is estimated up to 70 %, while its maximum is close to the 50 %. Comparing 2.28(a) and 2.28(b), one can notice that the same tendency is not verified for the attitude error too. This is probably due to the static model adopted for torque, which is unable to counteract the drift caused by higher drag forces experienced on the external rotors during a curvilinear trajectory. As a result, including blade flapping motion as a source of drag proved to be crucial when estimating the control errors of a quadrotor, because the aerodynamic theories adopted for full-sized helicopters may show some limitations when dealing with multirotor UAVs. The BET H-force is just part of the overall aerodynamic drag arising during a quadrotor flight, but BET model still provides better results regarding attitude errors. The combination of the two adopted models, making use of BET to model the aerodynamic torque as well as the lumped parameter model for drag forces, will probably match more and more the flight data in terms of both position and attitude.



(a) Norm of position error.



(b) Yaw error.



(c) Percentage error between the models and flight data performance

Figure 2.28: Models comparison: infinity-shape maneuver.

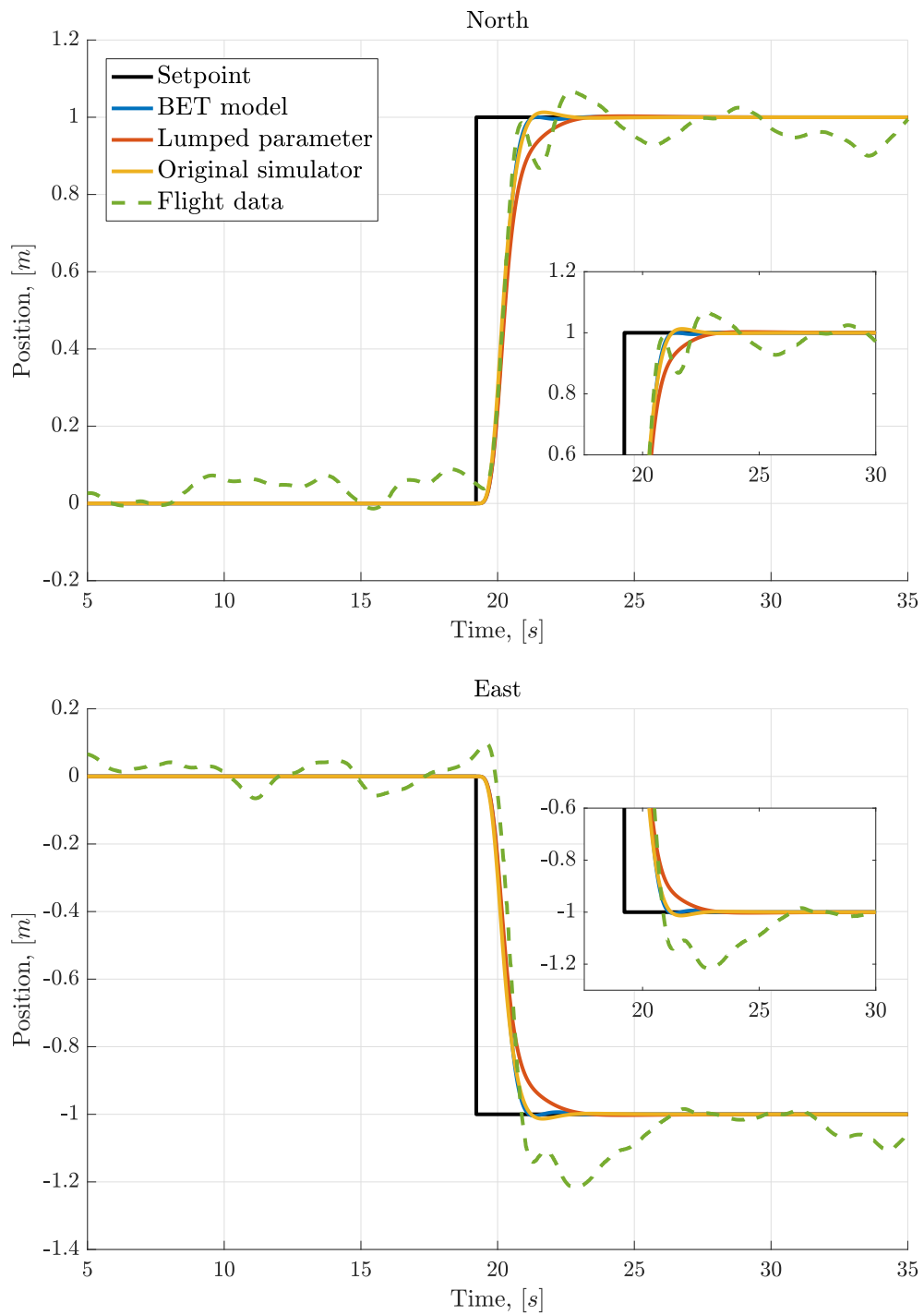


Figure 2.29: Models comparison: Step maneuver.

When the quadcopter is asked to perform a step maneuver, the relation between the aerodynamic drag and the aerodynamic damping is manifest. Even if the behavior of the UAV was not so well-defined during the flight test, the rise time of the models is almost equal to the real one, but according to the different drag model adopted, the step response is different. In Figure 2.29 it is possible to see that. For what concerns the flight tests, along the North position step the Lumped parameter model seems to fit the flight data better, while in East, the set-point was not reached very well by the quadrotor prototype. As a matter of fact the higher is the drag, the higher is the aerodynamic damping introduced in the motion. This latter will affect the overall response of the quadrotor to the step change in horizontal position. It is possible to see that the lumped parameter model provides an highly damped response, while BET and the Original Simulator show a critically-damped response, which is just due to the control structure embedded in the FCU (see [13]).

Finally, it is possible to conclude that the estimation of quadrotor tracking capability must necessarily make use of a detailed dynamic and aerodynamic model, which reduces modeling errors during high performance aggressive maneuvers. In fact, the control error is mostly due to the mismatch between the control actions and the actual thrusts developed by each rotor, which depend on the aerodynamic environment. More than the aerodynamic thrust variation, the drag forces play a crucial role on the quadcopter dynamics. Despite the validity of the classical aerodynamic theories on full-sized helicopters, they can not be directly applied on small scale multirotor UAVs. Indeed, a drag model based on the experimental observations during flights is recommended, to properly catch all the drag sources developing on a multirotor UAV. For this reason, further work has to be done to refine the aerodynamic model of a UAV, especially as regards the drag coefficient estimation.

Chapter 3

Problem Formulation

In the first part of this chapter, the problem is described in general terms. Thus, an overview of works related to our problem is presented. In the last but one section, the problem is formulated. The adopted approach is described and motivated, as well as the objective and constraints. In the last part of the chapter, the suitable algorithms adopted for the procedure are presented.

3.1 Problem description

The objective of this work is to perform autonomous landing of a small UAV, hereinafter referred to as *follower*, on a larger one, hereinafter referred to as *target*, while both are flying.

The problem is clearly three-dimensional but, for the sake of simplicity, is decoupled: the first part deals with trajectories synchronization in the horizontal plane, while the second regards the descent of the follower drone on the target one. The main focus of this thesis is on the latter problem.

In-plane synchronization is the first safety constraint. Only when the follower is in the same (N, E) position of the target, or in its neighborhood, the landing maneuver can start. This constraint must be verified for the entire duration of the descent, otherwise the procedure is stopped.

The safety constraints do not concern just in-plane synchronization. As a matter of fact, continuous feedback of the target drone altitude is used to design real-time reference landing trajectory. The main reason of this approach is that the target may voluntarily vary its altitude or experience vertical oscillations caused by the aerodynamic perturbations, originating from the follower's rotors wake.

3.2 State of the art

Surprisingly, nothing similar has been performed yet. Many articles treating the UAV landing topic can be found in literature but none of them deals with a flying landing target. In this section, an overview of some recent research studies is provided.

[2] proposes a novel controller structure to achieve fast, safe and precise landing of a quadrotor on a vertically oscillating platform. The control structure consists of three modules: motion estimation, trajectory generation and tracking control. In the tracking control module, an ARC (Adaptive Robust Controller) is designed to perform non-linear ground effect compensation and, hence, to enable accurate trajectory tracking. In the trajectory generation module, a time-optimal

reference trajectory for the quadrotor is generated such that it converges from the initial height precisely to the platform height with zero relative velocity (for smooth landing). The landing time duration is as short as possible, and physical safety constraints (position, velocity, acceleration bounds etc.) are satisfied during the entire landing process. Lastly, in the motion estimation module, UAV and platform positions are estimated on-line from only the measurement of the relative distance between them, as well as the inertia measurement of the UAV. An UKF (Unscented KalmanFilter) is constructed and the estimated parameters are fed to the other two modules in real time.

[18] describes a vision-based algorithm to control a VTOL UAV while tracking and landing on a moving platform. It makes use of image-based visual servoing (IBVS) to track the platform in two-dimensional image space and generate a velocity reference command, used as the input to an adaptive sliding mode controller. IBVS is computationally cheaper since it is less sensitive to the depth estimation allowing for a faster method to obtain this estimate. To enhance velocity tracking of the sliding mode controller, an adaptive rule is described to account for the ground effect experienced during the maneuver. Finally, the IBVS algorithm is integrated with the adaptive sliding mode controller for tracking and landing.

Outside the UAV field, [19] discusses non-linear methodologies that can be employed to devise real-time algorithms suitable for guidance and control of spacecrafts during asteroid close-proximity operations. A combination of optimal and sliding control theory provide the theoretical framework for the development of guidance laws that generates thrust commands as function of the estimated spacecraft state. Such algorithms can be employed for autonomous targeting of points on the asteroid surface (soft landing , Touch-And-Go (TAG) maneuvers). The guidance algorithm has its root in the generalized ZEM/ZEV feedback guidance and its mathematical equations are naturally derived by properly defining a sliding surface as function of Zero-Effort-Miss and Zero-Effort-Velocity. The latter enables the augmentation of the energy-optimal guidance law by a sliding mode that ensures global stability for the proposed algorithm.

Moreover, many research studies about trajectory tracking rely on quasi time-

optimal control algorithms.

[20] proposes a hybrid controller that ensures global convergence of the error dynamics to zero for a saturated multidimensional double integrator in presence of a time-varying reference. Such a controller combines a local linear feedback, which ensures the tracking in the presence of small errors (local mode), and a global stabilizing quasi time-optimal controller, which handles large errors and the saturation (global mode). The proposed switching hybrid logic is applied also to the reference. In global mode, the reference trajectory is slowed down to promote the stabilization task. Once in local mode, the controller speeds up the reference to restore the nominal behavior.

[21] proposal regards nonlinear proportional-integral (PI) and proportional-integral-derivative (PID) controllers combining time-(sub)optimality with linear control robustness and anti-windup properties for first-order and second-order integrator systems. A complementary contribution is the introduction of an integral action with anti-windup properties (wind-up may cause multiple bouncing between minimal and maximal values of the control, which worsens the overshoot problem) into the control law, under the constraint of ensuring global asymptotic stability. For illustration purposes, the proposed PID solution is applied to the longitudinal headway control of a vehicle following another vehicle.

Last, but not the least, [22] proposes a family of nonlinear state feedback global stabilizers for all planar linear systems which are globally stabilizable by bounded inputs. This family is parametrized by a nonlinear function whose selection can yield quasi time-optimal responses, where the “quasi” is required to achieve local exponential stability of the closed-loop. The arising trajectories are quasi time-optimal for arbitrarily large initial conditions; in this sense the proposed nonlinear control law may be very useful for embedded control applications with strong computational constraints.

3.3 Mathematical formulation

In this thesis, a control structure capable of performing an autonomous landing of a UAV on another one, while the second is flying (to be precise hovering, at least for a first attempt) is proposed. The task is challenging and hazardous.

The proposed structure consists of: a motion monitoring module, a trajectory gen-

eration module and a tracking control module. In the tracking control module, there are the built-in controller of the adopted flight control units. They are capable of tracking a given position and yaw reference with very good performances. The design of an ad hoc controller for this task is beyond the topic of this thesis. In the trajectory generation module, the desired set-point is generated, such that the follower converges from the initial height to the landing pad, mounted on the target. Last, but not least, the motion monitoring module manages the feedback position measurements and checks the safety of the whole procedure. Then, the state measurements and the safety variables feed the trajectory generation module. The last two modules run on the ground control station.

Now, the problem is formulated from a mathematical point of view. As seen in Chapter 2, the states of the UAVs are collected in a vector (2.34); so, with same meaning of notation but referring to the two drones, the states vectors are:

$$\begin{aligned} x_t, \\ x_f, \end{aligned} \tag{3.1}$$

where the subscripts t and f respectively refer to target and follower.

The landing objective is:

$$\lim_{t \rightarrow t_f} \left(\begin{bmatrix} N_f(t) \\ E_f(t) \\ D_f(t) \end{bmatrix} - \begin{bmatrix} N_t(t) \\ E_t(t) \\ D_t(t) \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ -\epsilon_D \end{bmatrix} \tag{3.2}$$

where t_f is the time in which the the procedure is terminated and ϵ_D is a positive defined tolerance. To be more precise, at $t = t_f$ the follower is hovering at a desired height (ϵ_D) above the target.

Clearly, the landing procedure is subject to physical constraints. The first one is on the down component of position:

$$D_f < D_t \quad t > t_0 \tag{3.3}$$

where t_0 is the start time of the landing maneuver. This means that the follower must be always above the target from starting time onward. In addition, the

relative velocity between the vehicles must be zero at final time, *i.e.*:

$$\lim_{t \rightarrow t_f} \left(\begin{bmatrix} \dot{N}_f(t) \\ \dot{E}_f(t) \\ \dot{D}_f(t) \end{bmatrix} - \begin{bmatrix} \dot{N}_t(t) \\ \dot{E}_t(t) \\ \dot{D}_t(t) \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.4)$$

At this point, there are two possible approach to solve the problem:

- a fully three dimensional solution,
- a decoupled one.

Considering that the UAVs involved are multi-rotors, one can understand that the more reasonable method to move one close to the other is on a vertical path. Hence, the chosen strategy is to decouple the problem in:

- horizontal plane synchronization,
- vertical approach.

Lastly, the relative navigation problem has not been studied in this work. The absolute positions of the drones are known by means of a motion capture system. Thanks to this, it will be possible to obtain requirements for the design of the relative navigation module, which is one of the future works to be done.

Now, the two parts of the problem are described in details.

Horizontal plane synchronization

The horizontal plane synchronization is continuously checked before and for all the duration of the landing maneuver. This is done through an averaging procedure. The mean error computation at time t takes into account the relative in-plane errors occurred in the past time history, within a time interval of length T_a :

$$\begin{aligned} \bar{e}_N(t) &= \frac{1}{T_a} \int_{t-T_a}^t (N_f(\tau) - N_t(\tau)) d\tau, \\ \bar{e}_E(t) &= \frac{1}{T_a} \int_{t-T_a}^t (E_f(\tau) - E_t(\tau)) d\tau. \end{aligned} \quad (3.5)$$

In order to respect the synchronization constraint the following relation

$$\bar{e}_p(t) = \sqrt{\bar{e}_N^2(t) + \bar{e}_E^2(t)} \leq \epsilon, \quad (3.6)$$

must be satisfied for a given time T_m , where ϵ is a small positive value. If both conditions are verified the procedure is considered safe, and so permitted. The detailed explanation of the safety procedures will be discussed in Section 4.1. It follows that the circular area, centered in (N_t, E_t) and with radius ϵ , is the acceptable region in which the follower drone should be for all the duration of the landing maneuver.

Vertical approach

Now looking at the vertical approach, the objective described in equation (3.2) turns out to be:

$$\lim_{t \rightarrow t_f} (D_f(t) - D_t(t)) = -\epsilon_D. \quad (3.7)$$

As long as the synchronization is obtained and checked, the focus of the procedure is on the generation of a vertical landing trajectory for the follower.

The landing trajectory is computed starting from an acceleration command, whose characteristics will be described in the next sections. As a consequence, both the algorithms needs a second-order integrator to get the desired trajectory:

$$\ddot{x} = u, \quad (3.8)$$

where u is the control variable, namely the reference acceleration. For different reasons, it has upper and lower bounds, hence:

$$a_{min} < u < a_{max}, \quad (3.9)$$

where a_{min} and a_{max} are respectively the climb and descent acceleration limits (signs due to *NED* convention).

In the next section, the trajectory generation algorithms will be described in

details.

3.4 Suitable algorithms description

Among the studied algorithms, only two proved to be suitable for our purpose. In the next sections, they will be presented.

They are used for the landing trajectory generation. The implementation is revisited because UAVs have their own controllers onboard, which can receive in input just the position set-point, and aim to minimize position tracking errors at best.

3.4.1 Three-states bang-bang algorithm

The algorithm is taken from [2], where it was originally used together with an ad hoc controller, but here only the trajectory generation module is retained and re-adapted taking into account safety factors.

Firstly, notice that the algorithm is one dimensional, built on z-axis pointing upward from ground, and it works with relative quantities.

The objective and the criteria listed in Section 3.3 lead to an optimization problem, as stated in [2], to obtain fast, safe and precise landing. According to Pontryagin's theorem, the optimal solution is a bang-bang type control law. The novelty is that it involves three states, instead of the classical two. This peculiarity is very useful in the case of multi-rotors, because it avoids abruptly changes in the angular speed of rotors, therefore the aerodynamic disturbances generated are reduced. Hence, the acceleration has three possible values:

- maximum downward acceleration,
- zero acceleration,
- maximum upward deceleration.

The decision process, whereby the proper input is chosen, is based on a test trajectory, *i.e.* the prediction of the future relative vertical motion between follower and target. Starting from arbitrary initial conditions (position and velocity) assuming that the maximum upward acceleration is applied, the algorithm predicts

when the relative velocity will be zero. Based on the relative position at that time instant, the control action is chosen:

- if the future relative position is lower or equal to zero it means that the follower will hit the target, so the maximum downward deceleration (hence upward acceleration) should be applied,
- if the future relative position is greater than zero two are the possible situations:
 - if the velocity limit has already been reached then no acceleration is applied, maintaining the maximum descent velocity,
 - else apply downward acceleration to reach it.

Then, as stated in equation (3.8) the reference acceleration is integrated twice to get the reference velocity and position. Eventually, the reference trajectory to feed the follower is calculated.

3.4.2 Quasi Time-Optimal algorithm

The second proposed algorithm for the landing trajectory generation is based on the concept of sub-optimal, or quasi time-optimal control illustrated in [22].

Since the double integrator depicted in equation (3.8) is a linear system, several techniques are suitable for control design. In fact, it is the basic model for several real systems, such as electrical and mechanical system. Therefore, its stabilization is widely studied in the literature; however, in practical implementations, non linearities have to be considered.

The family of nonlinear state feedback global stabilizers can be parameterized by a nonlinear function whose selection can yield quasi time-optimal responses, where the “quasi” is required to achieve local exponential stability of the closed-loop.

A nonlinear state feedback control law is proposed for the purpose of landing trajectory generation. The output of the control algorithm is an acceleration command depending on position and velocity errors between the two drones, which in turn it is integrated twice to get the landing trajectory.

For what concerns saturation levels, different acceleration bounds are set. In

order to provide a quasi-time optimal descent phase in a safe manner, the downward acceleration is bounded to a very low value. On the contrary the upward acceleration limit is as high as to have a fast response whenever the target moves up suddenly.

When being close to obtain the landing objective depicted in equation (3.7), the switching between the saturation levels is not discontinuous as for a bang-bang type solution, but it has a linear dependency on the position error. It follows that the proposed control law is locally Lipschitz and extremely close to being time-optimal, thus yielding quasi optimal responses for all signal ranges while preserving the robustness properties of a Lipschitz state-feedback. Then the arising reference trajectories are quasi time-optimal for arbitrarily large initial conditions. Hereinafter this algorithm will be referred as to QTO.

Chapter 4

Digital implementation

In the first part of this chapter, the motion monitoring module is described. The first element of the module computes the errors, which are used consequently to define some safety flags. Then, the trajectory generation module is presented. In particular, the three-states bang-bang and the QTO algorithms are presented from the mathematical point of view.

4.1 Error monitoring and safety procedures

Given the fact that the air-to-air landing maneuver is hazardous, a number of safety checks are performed before and during the execution.

The first step is to compute position and velocity errors, namely:

$$e_p = \begin{bmatrix} N_f \\ E_f \\ D_f \end{bmatrix} - \begin{bmatrix} N_t \\ E_t \\ D_t \end{bmatrix} = \begin{bmatrix} e_N \\ e_E \\ e_D \end{bmatrix}, \quad (4.1)$$

$$e_v = \begin{bmatrix} \dot{N}_f \\ \dot{E}_f \\ \dot{D}_f \end{bmatrix} - \begin{bmatrix} \dot{N}_t \\ \dot{E}_t \\ \dot{D}_t \end{bmatrix} = \begin{bmatrix} \dot{e}_N \\ \dot{e}_E \\ \dot{e}_D \end{bmatrix}. \quad (4.2)$$

As one can understand, position synchronization is investigated in order to make the follower land precisely inside the central part of the target drone, as close as possible to its center of gravity. In reality the target should be equipped with a landing pad with the same size as the acceptable safe region.

Knowing the previous information, the safety flag can be defined. The latter variable indicates if the landing is safe or not. The procedure to compute it is the following:

1. Calculate the mean value of the N and E errors in the interval $[t(k-n); t(k)]$:

$$\begin{aligned} \bar{e}_N &= \frac{1}{n} \sum_{i=k-n}^k (N_f^{(i)} - N_t^{(i)}), \\ \bar{e}_E &= \frac{1}{n} \sum_{i=k-n}^k (E_f^{(i)} - E_t^{(i)}), \end{aligned} \quad (4.3)$$

where n is the number of samples included in the average computation and k is the sample index at the current time. It is straightforward that n is related to the averaging time T_a (see Section 3.3) through the working frequency. In turn the absolute value of the mean position error is

$$|\bar{e}| = \sqrt{\bar{e}_N^2 + \bar{e}_E^2}. \quad (4.4)$$

2. From the first time instant in which it is true that:

$$\begin{aligned} |\bar{e}| &< \epsilon, \\ t &\geq t_0, \end{aligned} \tag{4.5}$$

update a counter variable c , which is initially zero for $t < t_0$. If the norm of the in-plane position errors exceeds the bound ϵ , namely

$$\sqrt{e_N^2 + e_E^2} > \epsilon,$$

c is immediately reset to zero.

3. If the counter value is greater than a defined threshold \bar{c} , then the flag takes unit value, meaning that the operation is safe, otherwise it is set to zero. Given the working frequency of the algorithm, the time interval over which the above conditions are satisfied is known. Hence, one can choose the counter threshold in order to monitor in-plane synchronization for the desired amount of time, T_m .

When the safety flag equals one, the in-plane synchronization is assumed to be acceptable to perform the landing because the follower is in a small neighborhood of the target. If during the landing maneuver the safety flag switches back from one to zero, the follower is asked to stop the descent. Whenever the safety flag switches again to be one, the initial conditions for the second-order integrator are reset to the actual relative position and velocity.

Another variable indicates if the landing procedure is started; its value becomes one as soon as safety flag equals one for the first time. This variable is necessary to switch the guidance law from the general set point to the landing one.

The last variable checks if the follower has arrived below the minimum relative position ϵ_D . When it happens the value is one, otherwise it is zero.

4.2 Three-states bang-bang

In this section, the three-states bang-bang algorithm is presented. Before it, convention and useful definitions must be given.

Convention

As already explained, the algorithm works only in the z -coordinate. The convention followed in the implementation is opposite to the NED one, i.e., the z -axis is positive upward, so that a positive acceleration is needed to stop while approaching the target. For this reason, error vectors and state vectors in input to the algorithm have sign changed.

Definitions

Here is a list of the main definitions:

z_r is the reference trajectory, which is given as set point to the follower,

z_d is the target altitude,

$z_a := z_r - z_d$ is the relative trajectory.

Algorithm

The reference trajectory $z_r(t)$ is generated respecting the following conditions and constraints:

1. The initial values are the follower position and velocity:

$$z_r(t_0) = -D_f(t_0), \quad (4.6)$$

$$\dot{z}_r(t_0) = -\dot{D}_f(t_0). \quad (4.7)$$

2. The final value for position is equal to the target one increased by the safety tolerance ϵ_D , while for velocity it is equal to the target one:

$$z_r(t_f) = -D_t(t_f) + \epsilon_D, \quad (4.8)$$

$$\dot{z}_r(t_f) = -\dot{D}_t(t_f), \quad (4.9)$$

such that the follower reaches a certain height above the target and hovers maintaining the relative distance till the thrust-off command.

3. The reference position, velocity and acceleration are constrained, as already mentioned:

$$\begin{aligned} z_r(t) &> -D_t(t) + \epsilon_D, \\ |\dot{z}_r(t)| &\leq \dot{z}_{rmax}, \quad t \in [t_0; t_f] \\ \ddot{z}_{rmin} &\leq \ddot{z}_r \leq \ddot{z}_{rmax}, \end{aligned} \quad (4.10)$$

where \dot{z}_{rmax} is the limit reference velocity, \ddot{z}_{rmin} is the maximum downward acceleration and \ddot{z}_{rmax} is the maximum upward acceleration. In the above, the position constraint means the follower is always above the target UAV.

4. The landing is as fast as possible.

Now, from the initial relative position and velocity, the problem is reduced to the generation of \ddot{z}_a to satisfy the above criteria, and then integrate to get z_a , \dot{z}_a . Subsequently, the references z_r , \dot{z}_r are computed by adding actual position and velocity of the target, z_d , \dot{z}_d respectively.

According to Pontryagin's theorem, the optimal solution \ddot{z}_a is a bang-bang type control law, with three states: downward, zero or upward relative acceleration. In the following, it is explained the decision process through which one of the possible values is assigned to the relative acceleration. If $z_a(t) = 0$, it means that the follower is already on the target so that the landing process is terminated. Else, a future "test trajectory" is defined $(z_a(\tau), \dot{z}_a(\tau))$, $\forall \tau \geq t$, starting from the current state $(z_a(t), \dot{z}_a(t))$ and considering that the limit upward acceleration, \ddot{z}_{rmax} , is given:

$$\begin{aligned} z_a(\tau) &= z_a(t) + \dot{z}_a(t)(\tau - t) + \int_t^\tau \int_t^\tau (\ddot{z}_{rmax} - \ddot{z}_d(\tau_2)) d\tau_2 d\tau_1 \\ &= z_a(t) + \dot{z}_a(t)(\tau - t) + \frac{1}{2}\ddot{z}_{rmax}(\tau - t)^2 - z_d(\tau) + z_d(t) \\ &\quad + \dot{z}_d(t)(\tau - t) \\ \dot{z}_a(\tau) &= \dot{z}_a(t) + \int_t^\tau (\ddot{z}_{rmax} - \ddot{z}_d(\tau_1)) d\tau_1 \\ &= \dot{z}_a(t) + \ddot{z}_{rmax}(\tau - t) - \dot{z}_d(\tau) + \dot{z}_d(t) \end{aligned} \quad (4.11)$$

If $(\ddot{z}_{rmax} - \ddot{z}_d(\tau)) > 0$, then this trajectory will intersect the x -axis (*i.e.*, $\dot{z}_a(\tau) = 0$) only once. The time instant of this intersection is denoted as t_s , which is the

solution of the equation $\dot{z}_a(\tau) = 0$ for τ , *i.e.*,

$$\dot{z}_a(t) + \ddot{z}_{rmax}(\tau - t) - \dot{z}_d(\tau) + \dot{z}_d(t) = 0. \quad (4.12)$$

Hence, the corresponding future relative position is $z_a(t_s)$. Now, there are three cases:

- If $z_a(t_s) \leq 0$, even producing the full deceleration the follower will still hit the target. Thus, the full deceleration has to be applied.
- If $z_a(t_s) > 0$ and $\dot{z}_a(t) \leq -\dot{z}_{rmax} - \dot{z}_d(t)$, it means that with the full deceleration the follower will stop at a position higher than the desired one. Thus, for time-optimality purposes, full deceleration does not have to be made at this point. However, the velocity of the follower already reaches its maximum downward limit. Thus, $\ddot{z}_a(t)$ is taken as $-\ddot{z}_d(t)$ to maintain the landing velocity at its limit.
- If $z_a(t_s) > 0$ and $\dot{z}_a(t) > -\dot{z}_{rmax} - \dot{z}_d(t)$, the case is the same as the previous one except for the fact that the velocity limit has not been reached yet. Thus, $\ddot{z}_a(t)$ is taken as its lower limit to make a full downward acceleration.

Concisely, the decision process is written in mathematical terms as:

$$\ddot{z}_a = \begin{cases} \ddot{z}_{rmin} - \ddot{z}_d(t) & \text{if } z_a(t) > 0 \wedge z_a(t_s) > 0 \wedge \dot{z}_a(t) > -\dot{z}_{rmax} - \dot{z}_d \\ \ddot{z}_d(t) & \text{if } z_a(t) > 0 \wedge z_a(t_s) > 0 \wedge \dot{z}_a(t) \leq -\dot{z}_{rmax} - \dot{z}_d \\ \ddot{z}_{rmax} - \ddot{z}_d(t) & \text{if } z_a(t) > 0 \wedge z_a(t_s) \leq 0. \end{cases} \quad (4.13)$$

The last part of the procedure requires a double integration to get $(z_a(t), \dot{z}_a(t))$. The initial conditions are the relative position and velocity measured at t_0 .

4.3 Quasi time-optimal

In this section, the quasi time-optimal control for trajectory generation is presented. The double integrator is the basic model for several real systems, such as electrical and mechanical systems. Therefore, the stabilization of the quasi time-optimal control of a double integrator has been widely studied in the literature. When designing controllers for these systems, in light of saturation, quite often

one seeks for solutions of the bang-bang or time-optimal types, so that the control input authority is fully exploited most of the time. These considerations place the basis for the development of the quasi time-optimal control law to generate the reference landing trajectory. The control effort is named u , which is the acceleration command to be integrated twice to get the landing path for the follower drone.

Definitions

Let $M > 0$ and $m < 0$ denote two real numbers. In what follows, sat_m^M denotes the saturation function defined on \mathbb{R} by

$$sat_m^M(x) = \begin{cases} M & x \geq M \\ x & \text{if } x \in (m, M) \\ m & x \leq m \end{cases} \quad (4.14)$$

Remark: if $m = -M$ the saturation function is denoted by $sat^M(x)$.

Also define the discontinuous $sign_m^M$ function as follows:

$$sign_m^M(x) = \begin{cases} M & x > 0 \\ 0 & \text{if } x = 0 \\ m & x < 0. \end{cases} \quad (4.15)$$

Algorithm

Since a double integrator is a linear system, several techniques are suitable for control design. Particular efforts have been placed in the design of controllers that explicitly consider the saturation phenomenon. Considering the second-order integrator

$$\ddot{x} = u, \quad (4.16)$$

with the bound constraints as previously, *i.e.*, $m \leq u \leq M$. As stated in [21] the time-optimal control associated with this system that takes x to zero in minimal

time can be written as

$$u(x, \dot{x}) = \text{sign}_m^M \left(- \left(x + \frac{\dot{x}|\dot{x}|}{2a} \right) \right), \quad a = \begin{cases} M & \text{if } x \geq 0 \\ -m & \text{if } x < 0 \end{cases} \quad (4.17)$$

This feedback law is discontinuous at points (x, \dot{x}) where $x + \frac{\dot{x}|\dot{x}|}{2a} = 0$ and also on the line $x = 0$.

It is in particular discontinuous at the desired equilibrium point $(x, \dot{x}) = (0, 0)$.

As pointed out before, the advantage of having a locally Lipschitz (instead of a discontinuous, bang-bang) feedback consists in better robustness to noise and disturbances; moreover, in a neighborhood of the origin it is desirable to have a linear control law in order to have at least local exponential stability.

In order to ensure continuity one may consider the following approximation,

$$u(x, \dot{x}) = \text{sat}_m^M \left(-K_p \left(x + \frac{\dot{x}|\dot{x}|}{2a(x, \varepsilon)} \right) \right), \quad (4.18)$$

where K_p plays the role of a “ proportional gain ”, with

$$a(x, \varepsilon) = \frac{M - m}{2} + \frac{M + m}{2} \text{sat}^1 \left(\frac{x}{\varepsilon} \right), \quad (4.19)$$

and ε a small positive number.

Note that $a(x, \varepsilon)$ is constant and equal to M in the case when $M + m = 0$, hence with symmetric saturation levels.

Now, a shortcoming of the above approximation is that it does not yield a (local) rate of convergence uniformly as fast as exponential, due to the quadratic velocity correction term involved in the time-optimal feedback law. In [21] this issue is taken care of by adding a complementary linear velocity term as follows:

$$u(x, \dot{x}) = \text{sat}_m^M \left(-K_p \left(x + \frac{\dot{x}|\dot{x}|}{2a(x, \varepsilon)} \right) - K_v \dot{x} \right), \quad (4.20)$$

with $K_v > 0$ playing the role of a “ derivative gain ”. Rearranging (4.20) it is possible to write:

$$u(x, \dot{x}) = \text{sat}_m^M \left(-K_p \left(x + \dot{x} \left(\frac{|\dot{x}|}{2a(x, \varepsilon)} + \frac{K_v}{K_p} \right) \right) \right), \quad (4.21)$$

At this point, one aims to benefit of the properties of both control theories, so the Lipschitz nonlinear control law and the local linear one can be blended by choosing the maximum value between $\frac{|\dot{x}|}{2a(x,\varepsilon)}$ and $\frac{K_v}{K_p}$. This is obtained by introducing a maximum function

$$\gamma_{max} = \max \left(\left(\frac{|\dot{x}|}{2a(x,\varepsilon)} \right), \frac{K_v}{K_p} \right). \quad (4.22)$$

In order to reduce the computational effort, the generic maximum function $\max \{a_1, a_2\}$ can be approximated as

$$\sqrt[n]{a_1^n + a_2^n},$$

with n a positive value greater than one.

Applying this property it is possible to rearrange equation (4.22) as follows

$$\gamma_{max} = \sqrt[n]{\left(\frac{|\dot{x}|}{2a(x,\varepsilon)} \right)^n + \left(\frac{K_v}{K_p} \right)^n}. \quad (4.23)$$

Indeed, the linear approximation of the above feedback at the desired equilibrium ($x = 0, \dot{x} = 0$) is the PD controller

$$u(x, \dot{x}) = -K_p x - K_v \dot{x}$$

whose proportional and derivative gains, K_p and K_v , can be determined by applying classical rules of linear control theory. Then, the designed control law is Lipschitz continuous, ensuring global convergence, and is a quasi time-optimal control of the double integrator. However, in a neighborhood of the origin the above nonlinear control law can induce a highly oscillatory behaviour; hence, it is advisable to introduce a local linear feedback inducing a critically damped local response. For instance, the choice

$$K_v = 2\sqrt{K_p} \quad (4.24)$$

yields two closed-loop poles equal to $-\sqrt{K_p}$ and ensures a critically-damped response with no overshoot. Thus, combining equation (4.24) with equation (4.21),

the nonlinear quasi time-optimal control law can be written as

$$\begin{aligned} u(x, \dot{x}) &= \text{sat}_m^M(-K_p(x + \dot{x}\gamma_{max})) = \\ &= \text{sat}_m^M\left(-K_p\left(x + \dot{x}\left(\sqrt[n]{\left(\frac{|\dot{x}|}{2a(x, \varepsilon)}\right)^n + \left(\frac{2}{\sqrt{K_p}}\right)^n}\right)\right)\right). \end{aligned} \quad (4.25)$$

Given the landing objective named in equation (3.2) for the landing purposes it is clear that

$$\begin{aligned} x &= D_f - (D_t - \epsilon_D) = e_p, \\ \dot{x} &= \dot{D}_f - \dot{D}_t = e_v. \end{aligned} \quad (4.26)$$

Recalling the double-order integrator definition in equation (4.16), in order to improve local convergence another term must be included in equation (4.25), when tracking a varying reference trajectory is required. See [20].

This additional term is simply the acceleration of the reference to follow u_r , that is the target drone vertical acceleration for this problem.

$$u_r = \ddot{D}_t. \quad (4.27)$$

From equations (4.16),(4.26) it can be proved that

$$\ddot{D}_f = \ddot{D}_t + u = u + u_r. \quad (4.28)$$

Finally the overall acceleration command for the follower, resulting from the quasi time-optimal non linear state feedback control law for landing trajectory generation, is

$$\begin{aligned} a_c(e_p, e_v) &= u_r + u = \\ &= \ddot{D}_t + \text{sat}_m^M\left(-K_p\left(e_p + e_v\left(\sqrt[n]{\left(\frac{|e_v|}{2a(e_p, \varepsilon)}\right)^n + \left(\frac{2}{\sqrt{K_p}}\right)^n}\right)\right)\right). \end{aligned} \quad (4.29)$$

Then, the reference landing trajectory z_{d_f} for the follower is computed by means of a double integration of the acceleration command a_c .

Chapter 5

Simulation results

In this chapter, the simulation results for the landing maneuver are shown, according to the control algorithm embedded in the trajectory generation module. The two multirotors are assumed to be quadrotors. In simulation environment the two quadcopters involved in the air-to-air landing maneuver are identical, because the goal is just to analyze and verify the efficiency of both algorithms. For the sake of simplicity, the models of the two UAVs refer to the aerodynamic multi-body model described in details in Chapter 2. The maneuver is undertaken just in the vertical direction, hence no in-plane drag force arises even though each rotor thrust of the follower is influenced by its descent flight. According to the conclusions stated in Section 2.7.3 about the two drag models proposed, both of them will show the same aerodynamic effects for a vertical flight, hence each model accurately predicts the follower performance.

The simulations are performed in ideal and non-ideal conditions; in this latter case reasonable measurement noise is added and the set-point is discretized. In the last part of the chapter, some considerations are made about differences between the two cases.

5.1 Simulation setup

In this section, the trajectories commanded to the two drones are delineated. Furthermore, the conditions of the simulations are described.

The target is asked to perform a vertical oscillating trajectory, *i.e.*:

$$\begin{cases} N_t = 0 \\ E_t = 0 \\ D_t = A \sin\left(\frac{2\pi t}{\tau}\right) + h_t \end{cases} \quad (5.1)$$

where $A = 0.05 \text{ m}$ is the amplitude of the oscillation, $\tau = 9 \text{ s}$ the period of the oscillations and $h_t = -2.5 \text{ m}$ the mean altitude.

On the other hand, the follower climbs to $h_f = -4.5 \text{ m}$ and then it receives as in-plane set-point:

$$\begin{cases} N_f = N_t \\ E_f = E_t. \end{cases} \quad (5.2)$$

Thereafter, when it is safe, it starts to land.

For the sake of clarity, all the plots presented start at the time t_0 , so that take-off and initial positioning are cut away from the representation.

Firstly, both algorithms are simulated in ideal conditions. Secondly, two actions are taken to perform simulations that are as close as possible to reality, namely:

- discretize the set-point at 10Hz
- add white noise to the state estimate feedback, to reproduce real sensors measurements. Table 5.1 shows the estimated standard deviation of the measurement noise.

For what concern safety procedures, recalling Section 3.3, the averaging time is $T_a = 10 \text{ s}$, whilst the monitoring time for the average errors is $T_m = 5 \text{ s}$. Instead, the bound for in-plane synchronization is $\epsilon = 0.05 \text{ m}$, this value is chosen to have the follower safely inside the landing pad area (see Section 6.2).

Quantity	Standard deviation	Unit of measure
n	0.001	m
e	0.001	m
d	0.01	m
\dot{n}	0.01	m/s
\dot{e}	0.01	m/s
\dot{d}	0.02	m/s
ϕ	0.1	deg
θ	0.1	deg
ψ	0.1	deg
p	0.1	deg /s
q	0.1	deg /s
r	0.1	deg /s

Table 5.1: Standard deviation of measurement noise.

Parameter	Value	Unit of Measure
\ddot{z}_{rmax}	0.001	$[m/s^2]$
\ddot{z}_{rmin}	-0.05	$[m/s^2]$
\dot{z}_{rmax}	0.05	$[m/s]$

Table 5.2: Three-states bang-bang parameters.

5.2 Three-states bang-bang

In the bang-bang control algorithm just three parameters need to be set to have a safe and slow descent. The maximum and minimum vertical acceleration limits are definitely responsible for the landing trajectory. Imposing a small value for \ddot{z}_{rmax} , the decision process forces the follower to decelerate earlier. Otherwise, with a large value, there will be a strong brake just when the follower is close to the target. On the other hand, the downward acceleration limit \ddot{z}_{rmin} determines how fast the maximum downward velocity will be reached. Specifically, the velocity limit \dot{z}_{rmax} has been chosen to have a slow descent. In Table 5.2 the selected parameters are shown. The target acceleration feedback \ddot{z}_a necessary to compute the relative acceleration \ddot{z}_a (see equation (4.13)) is provided by the quadrotor multi-body model referred to the target, implemented in Dymola [3].

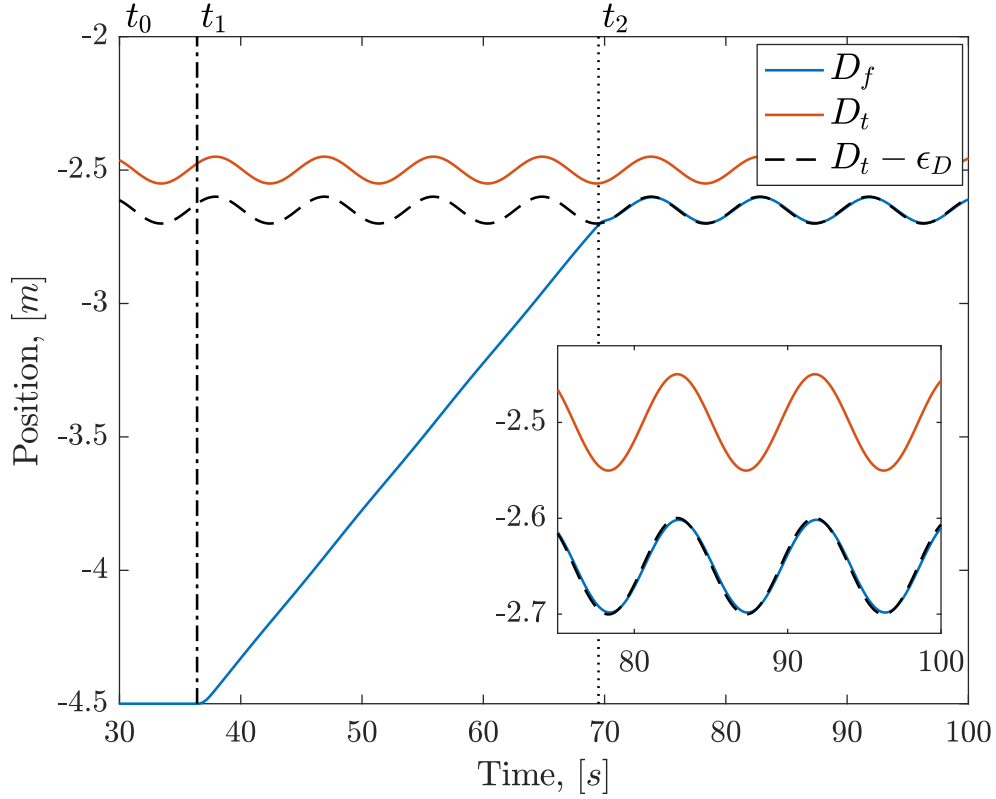


Figure 5.1: Follower, target and desired altitude (bang-bang), ideal case.

Ideal environment simulation

In Figure 5.1, one can see the follower and target down position, as well as the final desired altitude for the follower, namely $D_t - \epsilon_D$. The vertical lines indicate key time instants of the procedure, precisely:

t_0 : already described in Section 3.3, is the start time of the landing maneuver.

t_1 : is the time instant in which the safe flag assumes unit value for the first time.

t_2 : is the first time in which the relative position set point z_a computed in the landing algorithm is less or equal to zero.

The time necessary to complete the maneuver, called t_{go} , is:

$$t_{go} = t_2 - t_1 = 38 \text{ s.}$$

The control variables computed by the algorithm are depicted in Figure 5.2. The relative trajectory z_a starts at a value equal to $z_a(t_1) = |D_f| - |D_t| - \epsilon_D$, in this way the follower will stop the maneuver at $z_a(t_2) = |D_t| + \epsilon_D$. Moreover, in the three plots is clearly visible the influence of the target motion, in this case oscillating. The algorithm is clearly able to converge to the target, by means of a slow and safe descent trajectory. The guidance law, thanks to the target acceleration feedback, reflects the target motion onto the velocity and position set-points of the follower, generating a landing trajectory which perfectly matches the desired altitude at the end.

Non-ideal environment simulation

As already mentioned, the second simulation environment analyzed is built considering discrete set-point and white noise as disturbance. The reference position is computed through a discrete time double integration on \ddot{z}_a , making use of backward Euler method.

In Figure 5.3, the landing maneuver with these changes is shown. Looking at Figure 5.4, one can clearly see the effect of noisy estimate on the control variables, especially acceleration and velocity. The main difference with respect to the ideal case is in terms of the time of descent, namely $t_{go} = 23 s$. The reason lies in the difference between the initial conditions. Even though the standard deviation of the white noise added to the position and velocity estimates is small, when the control law is switched on by the safety flag, the initial conditions for the double integration are noisy enough to affect the algorithm convergence time. The generated trajectory has a very small slope, due to the value assigned to velocity and acceleration limits, listed in Table 5.2. Although the difference in the initial conditions is not so relevant, the time needed to reach the final conditions, having a small acceleration can significantly change. This reflects the possibility to have a faster or slower descent trajectory, according to the values of position and velocity errors when the descent starts. In particular, the velocity error is the most influential term: comparing Figure 5.4(b) and Figure 5.2(b) one can notice that the initial value of \dot{z}_a not only changes in value, but also in sign.

Secondly, the follower trajectory after t_2 does not precisely match the desired altitude (dashed line) several times more than in the continuous case. However,

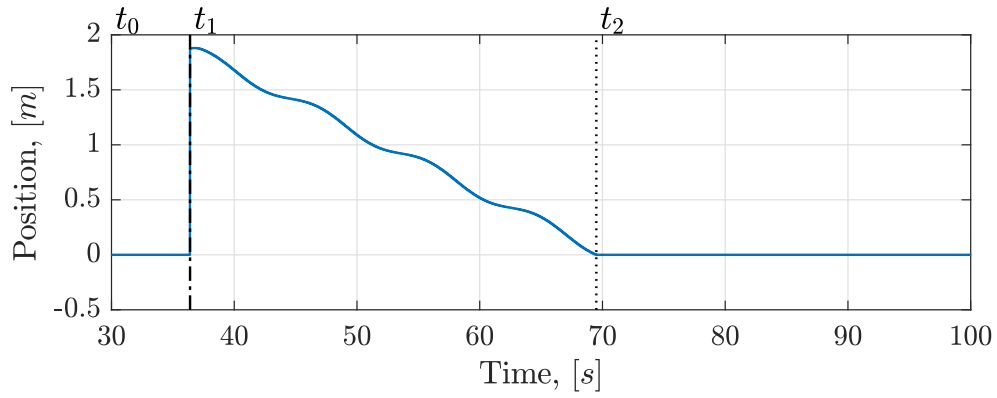
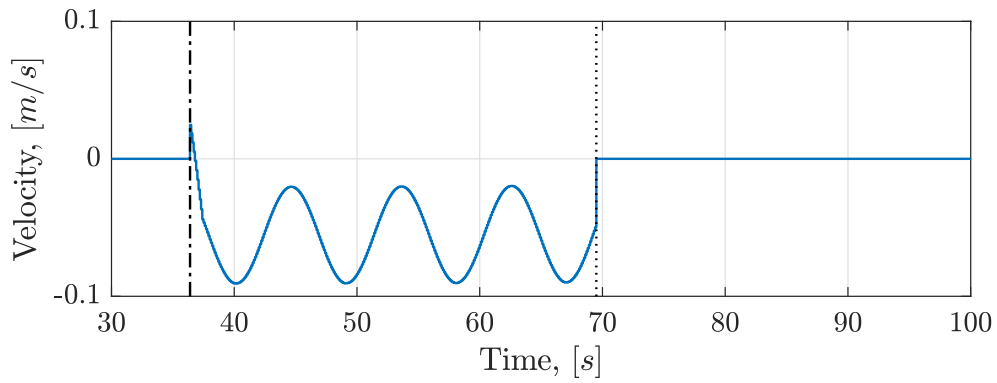
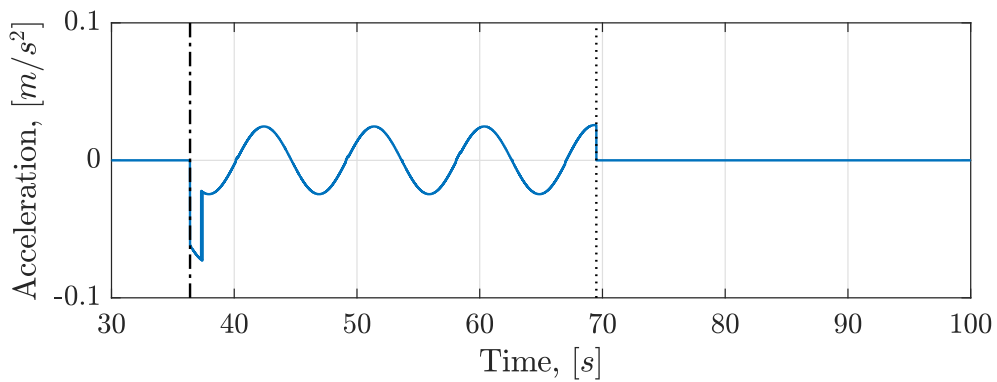
(a) Relative position, z_a (b) Relative velocity, \dot{z}_a (c) Relative acceleration, \ddot{z}_a

Figure 5.2: Control variables (bang-bang), ideal case.

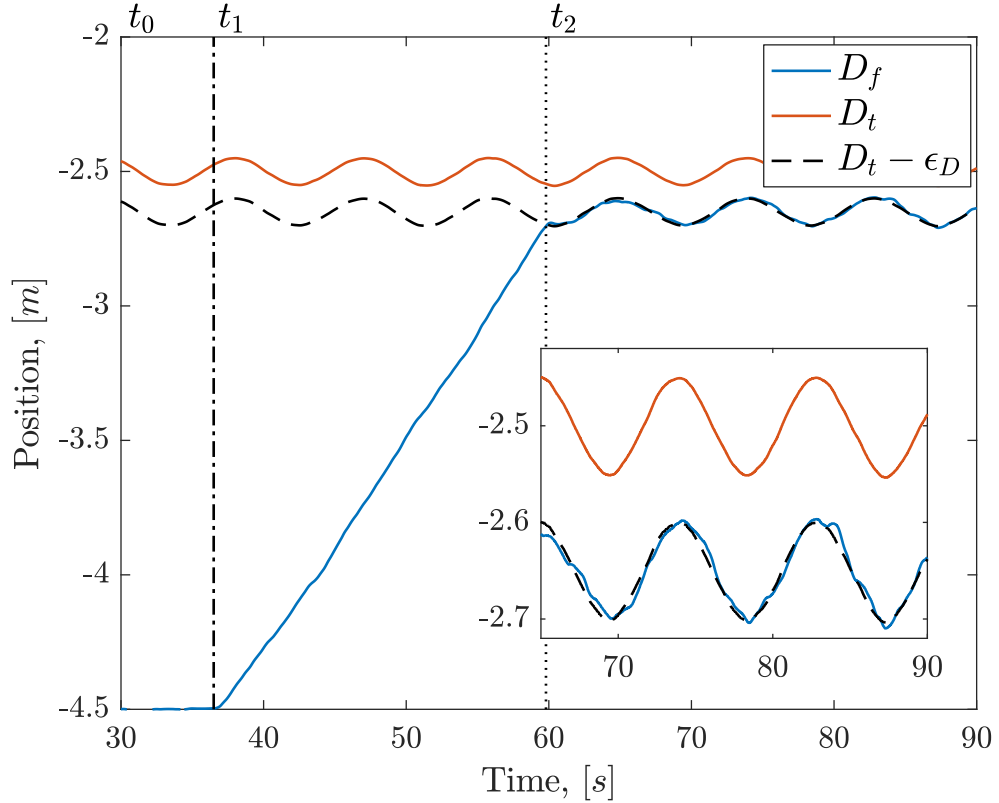


Figure 5.3: Follower, target and desired altitude (bang-bang), non-ideal case.

these crossings are not affecting the effectiveness of the algorithm since they are of negligible entity.

5.3 Quasi time-optimal

The Quasi time-optimal algorithm relies on many parameters, hence a performance investigation is necessary. At first saturation levels M and m must be defined, as well as the neighborhood ε of the desired position $x = 0$ in which the switching function $a(x, \varepsilon)$ makes the control law Lipschitz. When being close to the landing pad, the follower must track any target vertical motion. Then, a fast response with the maximum upward acceleration is desired whenever the target moves up. In that way the follower would not hit the target on its landing pad. This stringent requirement is the reason for choosing ε as small as possible.

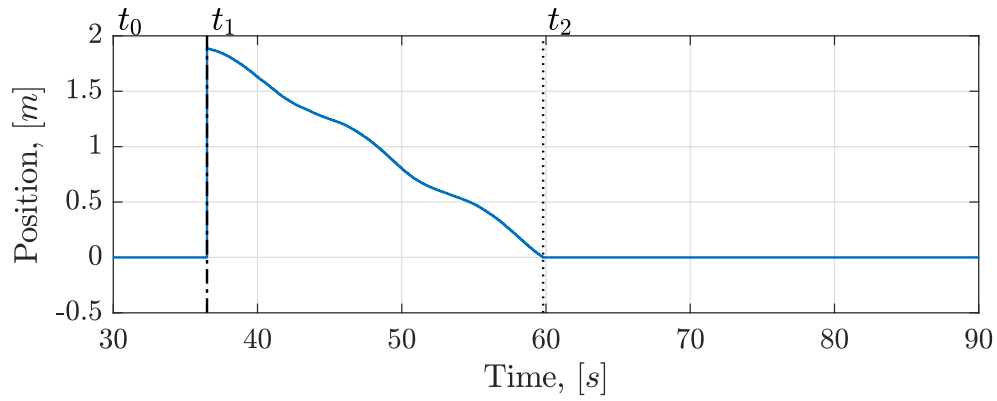
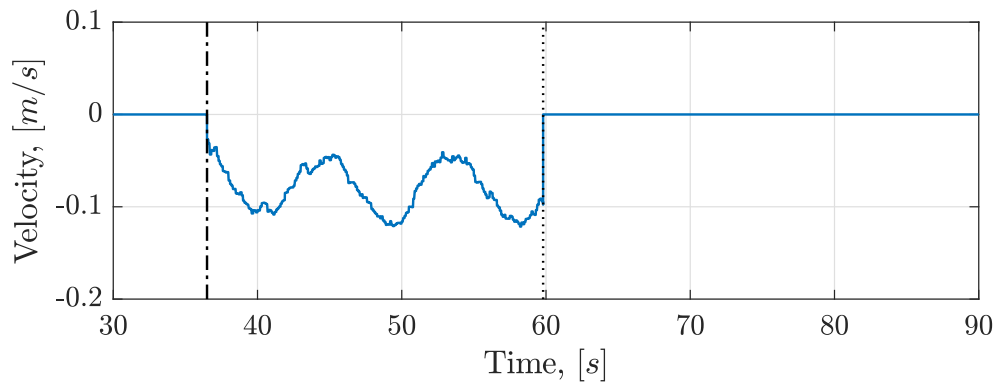
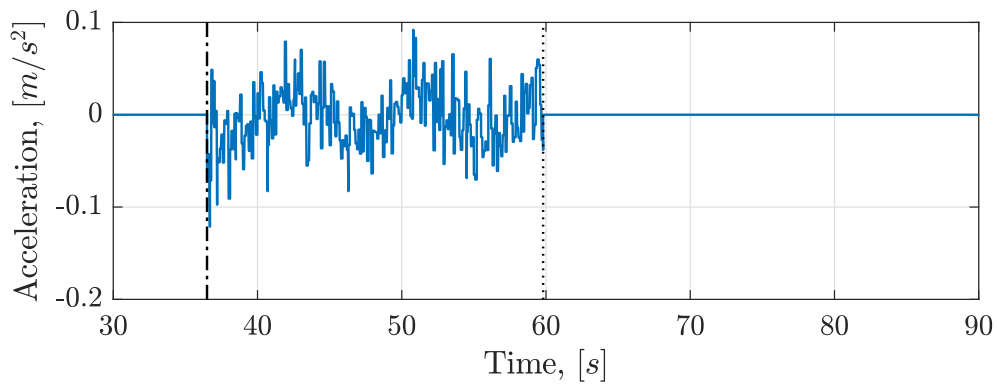
(a) Relative position, z_a (b) Relative velocity, \dot{z}_a (c) Relative acceleration, \ddot{z}_a

Figure 5.4: Control variables (bang-bang), non-ideal case.

It follows that the maximum upward acceleration m (minimum saturation level according to \mathcal{F}_E) must be larger. Instead, the maximum downward acceleration must be small to have a safe and slow descent, so to have the least dangerous possible impact with the landing pad.

5.3.1 Sensitivity analysis

The most arduous choice regards K_p . Recalling equations (4.16), (4.25) it is clear that the proportional gain plays the role of the natural frequency of the double integrator, while K_v is responsible of the damping in the control law. The relation between critical damping and natural frequency in a second order system

$$\ddot{x} + 2\xi\omega_n\dot{x} + \omega_n^2x = 0$$

is the same provided by equation (4.24) to ensure a critically damped local response.

A performance analysis is required on global convergence (arrive close to the target without overcoming it in a safe manner, in reasonable time) and local convergence (track any target vertical motion when being close to it) of the quasi time-optimal algorithm.

The QTO control should decrease both position and velocity errors smoothly, avoiding the bouncing around the desired equilibrium $(x, \dot{x}) = (0, 0)$. Once it has been reached, hence $e_p = e_v = 0$, the acceleration command u will be zero.

However, a null acceleration command may occur even before reaching the equilibrium, if position and velocity errors compensate each other, *i.e.*:

$$e_v = -\frac{\sqrt{K_p}}{2}e_p. \quad (5.3)$$

In this situation, if $\sqrt{K_p} < 2$, the velocity error is lower than the position one. According to this consideration, one may seek to set K_p very low, but this choice has its drawbacks too. When position and velocity terms compensate each other within the control law (equation (5.3) is verified), for a given position error close to zero, there could be some residual velocity error, depending on the value for

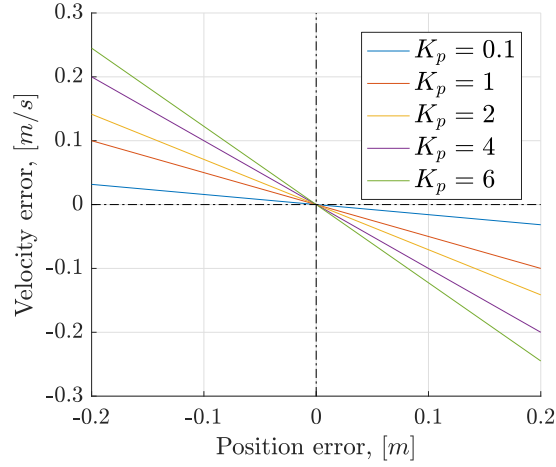


Figure 5.5: Null acceleration command iso-lines (QTO).

K_p adopted in the algorithm. See Figure 5.5.

With a given small velocity error, the smaller is the K_p , the larger is the position error.

From this first observation, one may conclude that a small K_p frequently provides a null acceleration command during the descent, even when the follower has just started the landing maneuver, but it is still far from the target.

On the contrary, with a given small position error, the larger is the proportional gain, the larger is the velocity error.

Then, it is possible to state that when coming closer to the target, a high K_p may give rise to a null acceleration command with large velocity error, resulting in a continuously bouncing motion of the follower.

Now let's consider separately the position and velocity errors contribution to the control law, and call them

$$u_p = -K_p e_p, \quad (5.4)$$

$$u_v = -2\sqrt{K_p} e_v. \quad (5.5)$$

When $u \approx 0$ these contributions are the same but opposed in sign

$$u_p = -u_v.$$

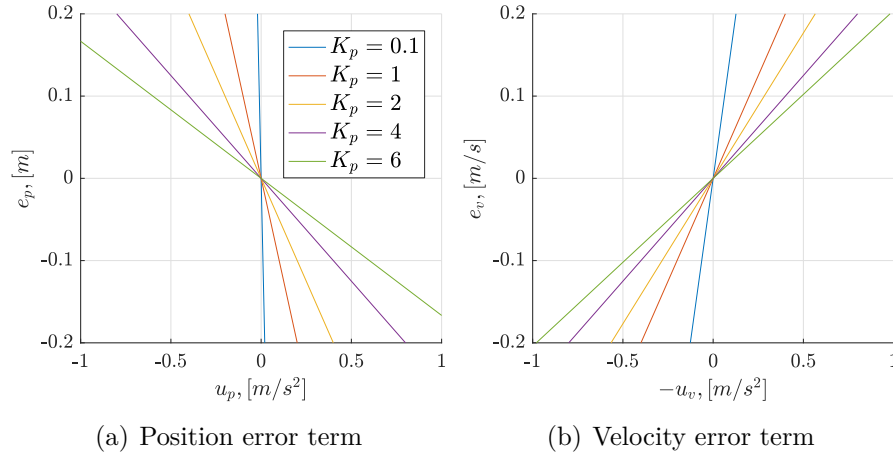


Figure 5.6: Position and velocity errors contribution to acceleration command (QTO).

It is possible to compare the position and velocity errors whenever the acceleration command is null and so, when both contributions compensate each other.

From Figures 5.6(a), 5.6(b) it is clear that a control law with $K_p < 1$ provides very weak control efforts u_p and u_v , regardless of both errors. This means that $u = 0$ occurs many times during the descent and that the local convergence will be reached in an infinite time, confirming what previously stated. On the other hand, $K_p \geq 1$ provides larger slope of the control efforts with respect to position and velocity errors. This means that a null acceleration command is given only when the follower is close to the target, resulting in a faster local convergence.

There is a drawback of choosing a much large K_p . When being close to the target ($e_p \approx 0$), the control effort u_p is high. It follows that to have a null acceleration command, a consistent velocity error is induced. The control law aims to minimize the latter with a significant control effort u_v , and later on, when the next null acceleration command occurs, there is still a residual position error. This phenomenon happens continuously: while the control aims to minimize one of the errors, the other one increases, and viceversa.

That is the cause of the bouncing motion around the equilibrium point when adopting large K_p values.

From this result one may conclude that $K_p \geq 4$ provides a faster descent rate ($u = 0$ only close to the target), but makes the follower bounce locally around the target, whereas $K_p < 1$ cause a slow descent with no local convergence in a finite

time.

Hence it is reasonable to conclude that

$$1 \leq \sqrt{K_p} < 2 \quad (5.6)$$

is the acceptable range in order to have a safe, smooth descent in a reasonable time and a good local convergence of position and velocity errors.

A set of simulations were performed to understand which is the most suitable K_p ensuring the satisfactory behavior. In reality, accelerometer measurements of a flying vehicle are very noisy and unavailable in real time to be used in feedback. As a consequence, referring to equation (4.25), the target acceleration \ddot{D}_t , *i.e.* u_r , will not be used when analyzing the performances of the QTO algorithm. Finally, the QTO control has been tuned assuming that

$$u_r \approx 0, \quad (5.7)$$

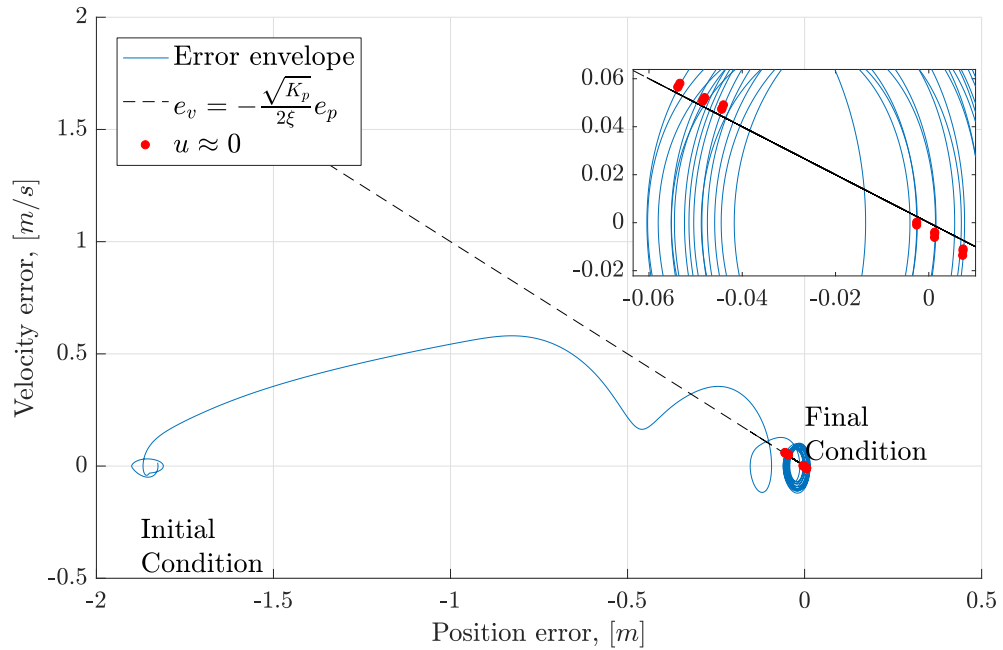
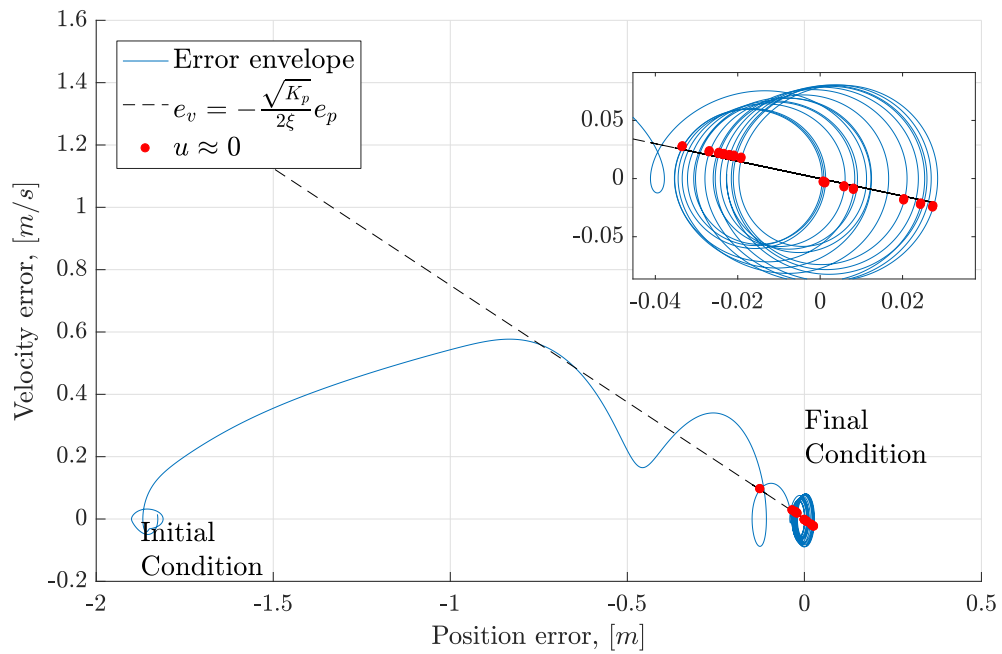
hence

$$a_c(e_p, e_v) = u. \quad (5.8)$$

In Figures 5.7(a), 5.7(b), 5.7(c), 5.7(d) the envelope of position and velocity errors during the landing maneuver is shown. According to previous statements, when using large proportional gains, the local convergence is affected by bouncing effects. See figures 5.7(a), 5.7(b). The error envelope revolves around the equilibrium point, just crossing it without reaching it.

Making use of a smaller K_p instead increases the local convergence. Looking at Figure 5.7(d) it is possible to notice that the error path crosses many times the $u = 0$ iso-line due to errors compensation, having null acceleration commands even when the follower is far from the target.

At the final condition the limits of a control law with $K_p = 1$ are clear: (e_p, e_v) is closely surrounding the equilibrium point, never reaching it. This behavior reflects the phase lag between target and follower trajectories.

(a) $K_p = 2^2$ (b) $K_p = 1.5^2$

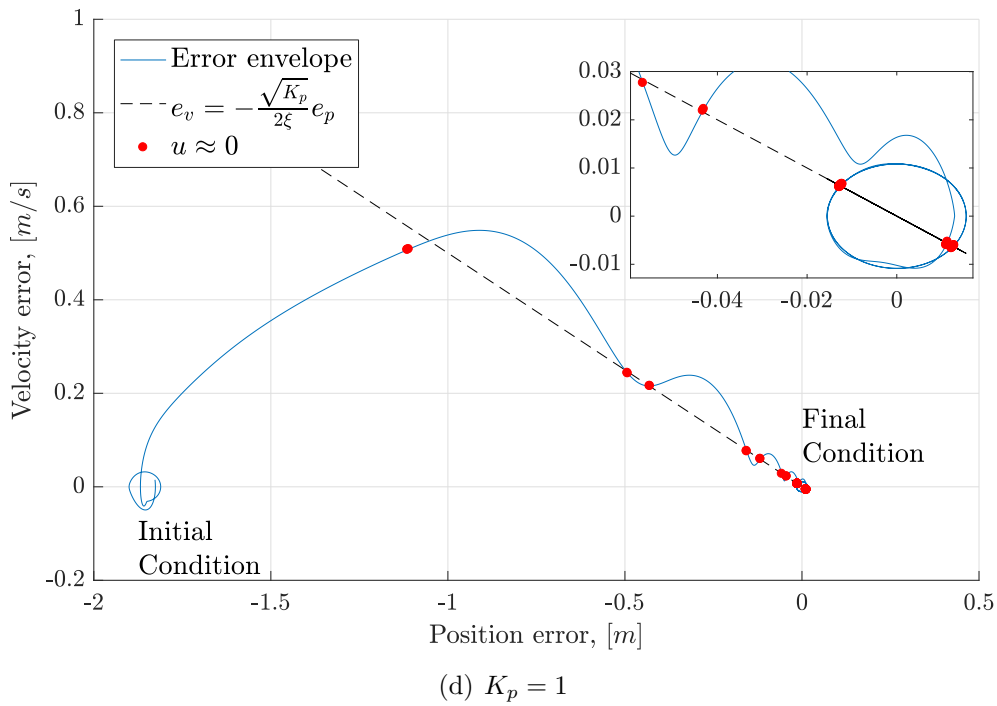
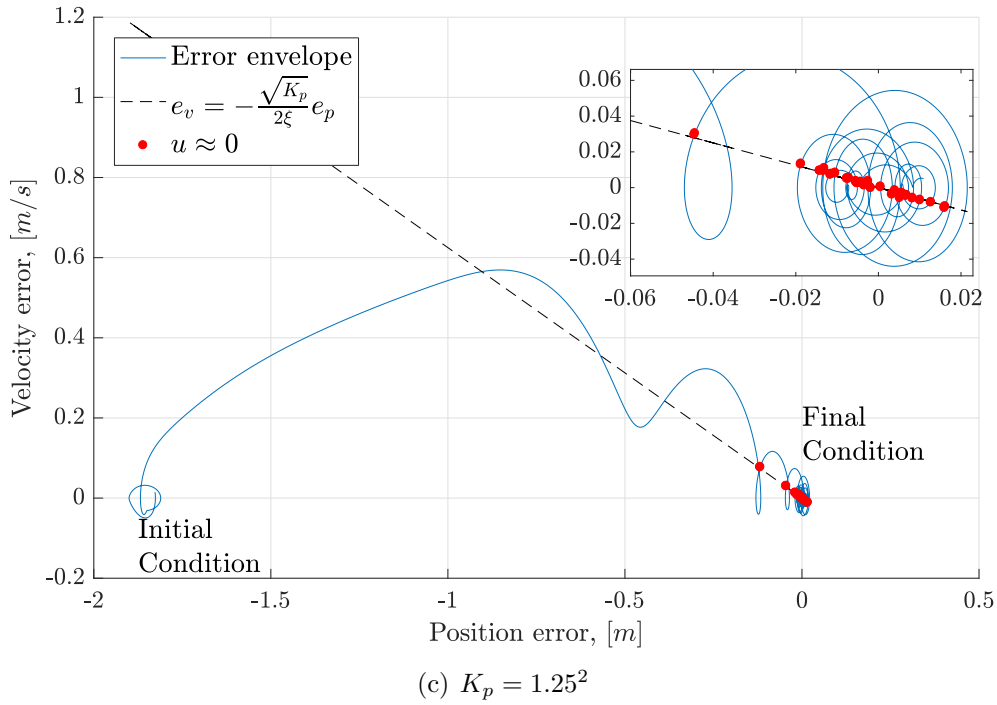


Figure 5.7: Errors evolution with different K_p (QTO).

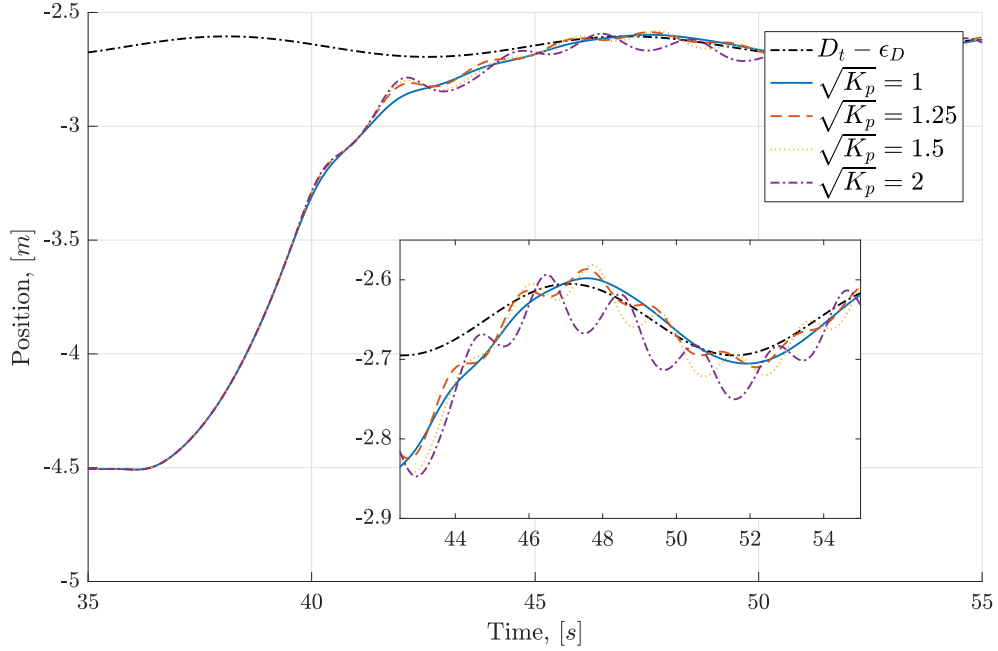


Figure 5.8: Landing trajectories (QTO).

The best performance is achieved with $K_p = 1.25^2$ as shown in Figure 5.7(c). The evolution of (e_p, e_v) crosses the $u = 0$ iso-line having always a non-null acceleration command until the follower get close to the target. Many null acceleration command can be found in the final condition, once the control law has minimized both position and velocity errors, reaching the local convergence. The follower is properly tracking the target drone now: global and local convergence are fully satisfied. Finally Figure 5.8 compares the landing trajectories of each simulation, showing in details all the peculiarities due to different choice of K_p , from the bouncing effect and the phase lag up to the local convergence satisfied. The compromise between the performances shown in Figures 5.7(d), 5.7(c) and Figure 5.8 dictates the choice for the proportional gain. K_p is set to 1.15^2 .

Referring to equation (4.23) n is high enough to let the nonlinear part of the control law work just during the initial part of the descent. In Table 5.3 the control law parameters used in this sensitivity analysis are reported, together with the optimal K_p found. In the next sections, the same parameters are adopted to simulate the maneuver.

Parameter	Value	Unit of measure
M	0.15	$[m/s^2]$
m	-0.6	$[m/s^2]$
ϵ	0.01	$[m]$
K_p	1.3225	$[1/s^2]$
n	4	$[-]$

Table 5.3: Parameters of quasi time-optimal algorithm.

Ideal environment simulation

As previously mentioned, the reference trajectory is computed integrating twice the acceleration command u , with no u_r in feedback. In Figure 5.9 the target and follower vertical positions are shown, along with the follower desired altitude, referred to as $D_t - \epsilon_D$.

Unlike the case of the three-states bang-bang algorithm, the quasi time-optimal control law is continuously fed by the position and velocity errors, so the definition of t_2 is slightly different. In this case t_2 is the first time in which the follower has arrived to the desired vertical position, namely

$$D_f = D_t - \epsilon_D.$$

As depicted in Figure 5.9, now the time necessary to reach the local convergence

$$t_{go} = t_2 - t_1 = 10 \text{ s}. \quad (5.9)$$

It is worth noticing that the small phase lag between target and follower position may be caused by the combination of QTO control algorithm performance and flight control unit delay implemented in the model. This negligible phase lag can be seen in Figure 5.10 too. In Figure 5.11 the error between the landing set point and the target altitude is illustrated, with a close zoom in the final part of the simulation. At the end, the reference trajectory seems to have a small delay, resulting in small and acceptable error of less than 1 *cm*. The acceleration command depicted in Figure 5.12 clearly oscillates around its zero value at regime, simply due to the fact that is not provided with the target acceleration u_r . That is the main cause of the small delay between reference landing trajectory and target motion.

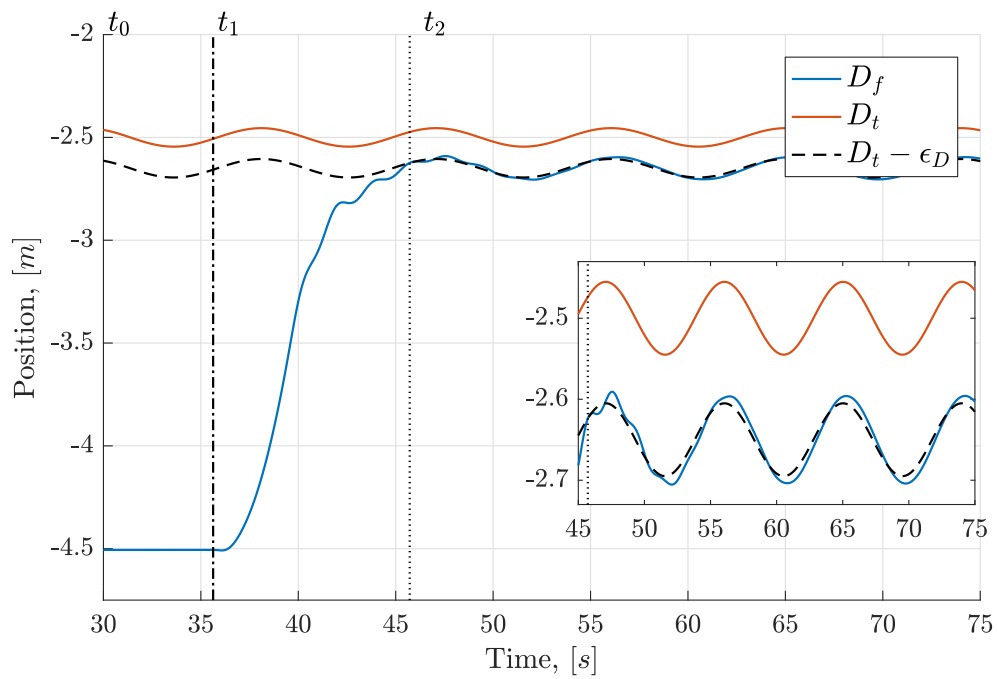


Figure 5.9: Follower, target and desired altitude (QTO), ideal case.

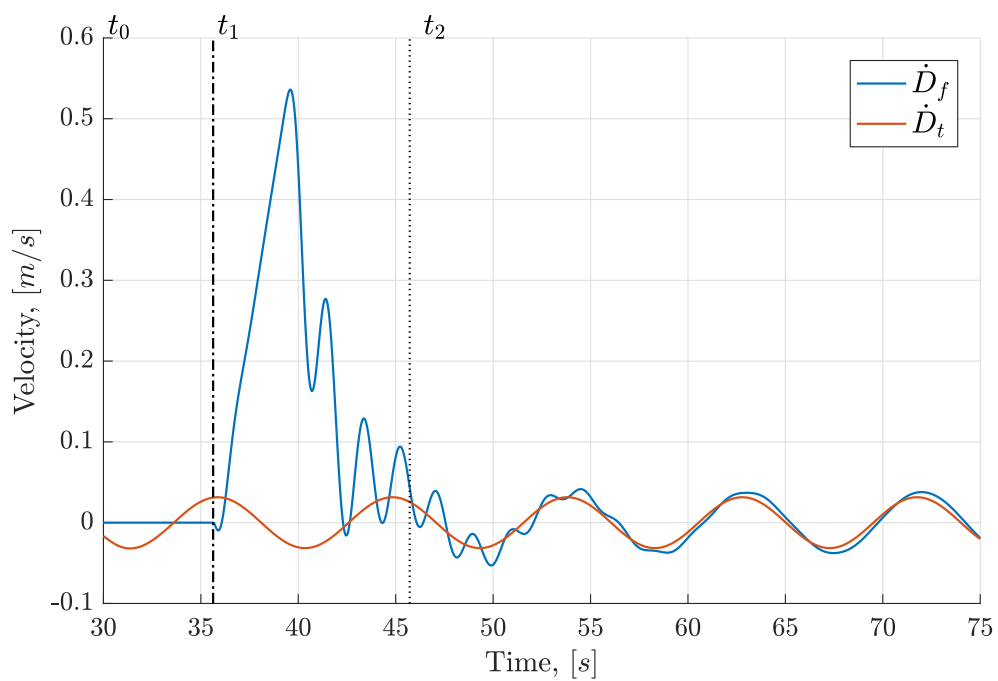


Figure 5.10: Follower and target velocity (QTO), ideal case.

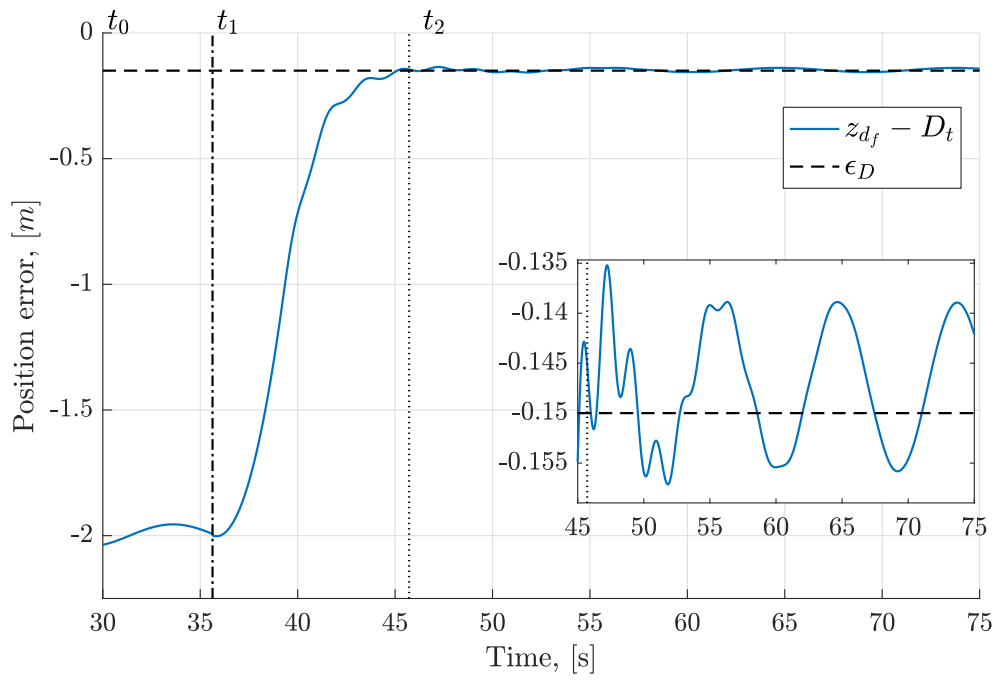


Figure 5.11: Landing trajectory error (QTO), ideal case.

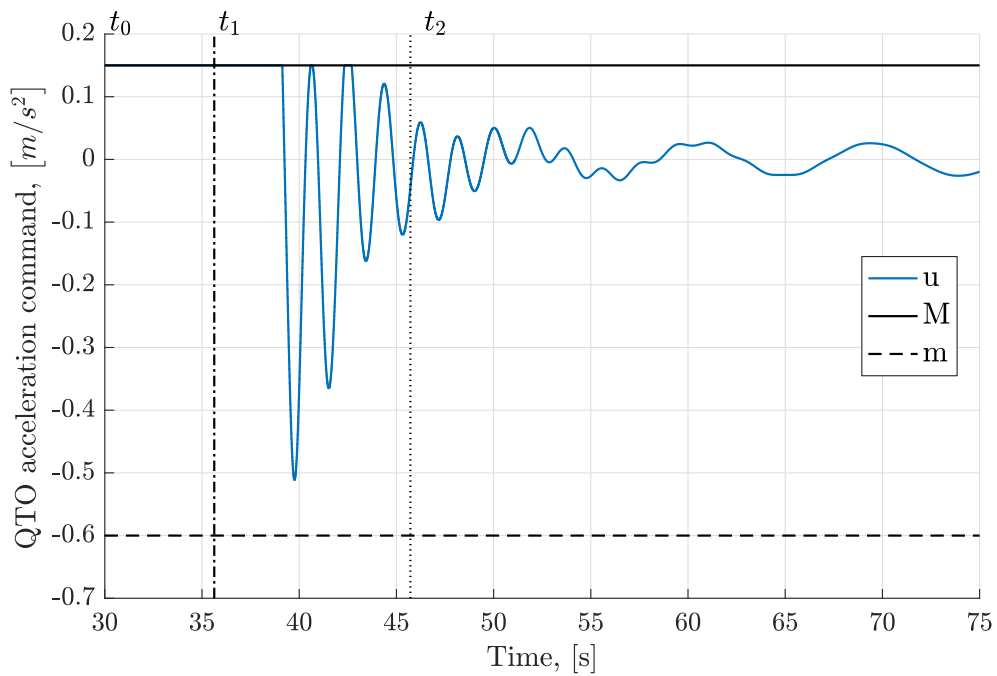


Figure 5.12: Acceleration command (QTO), ideal case.

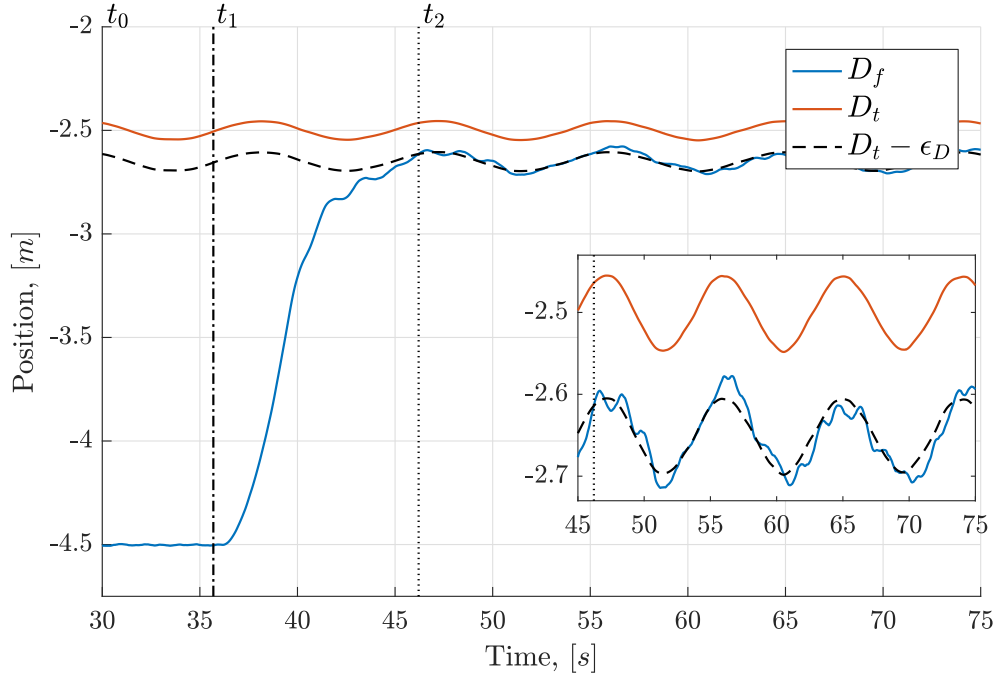


Figure 5.13: Follower, target and desired altitude (QTO), non-ideal case.

Non-ideal environment simulation

Now the same trajectory illustrated in the previous section is simulated supposing to have real sensors aboard of the drone, providing measurements and noisy estimates regarding the state of both UAVs. The reference position is computed through a discrete time double integration on u , not having the target acceleration as a feedback again and making use of backward Euler method. Clearly the time to reach the local convergence is slightly increased. Now

$$t_{go} \approx 11 \text{ s.}$$

From the results illustrated in Figures 5.13 and 5.15, the algorithm is able to reject the noise disturbance included in the model and the air-to-air landing maneuver is successfully performed. Instead, Figure 5.14 shows the large impact of having noisy measurements for both drones velocity. The velocity error is not always minimized because the acceleration command, depicted in Figure 5.16, is fed with both the position and velocity noisy measurements, and results to be very different from the ideal one (see Figure 5.12). For this reason, the velocity feedback may give some problems when the noise is added to the estimate. One may obviously

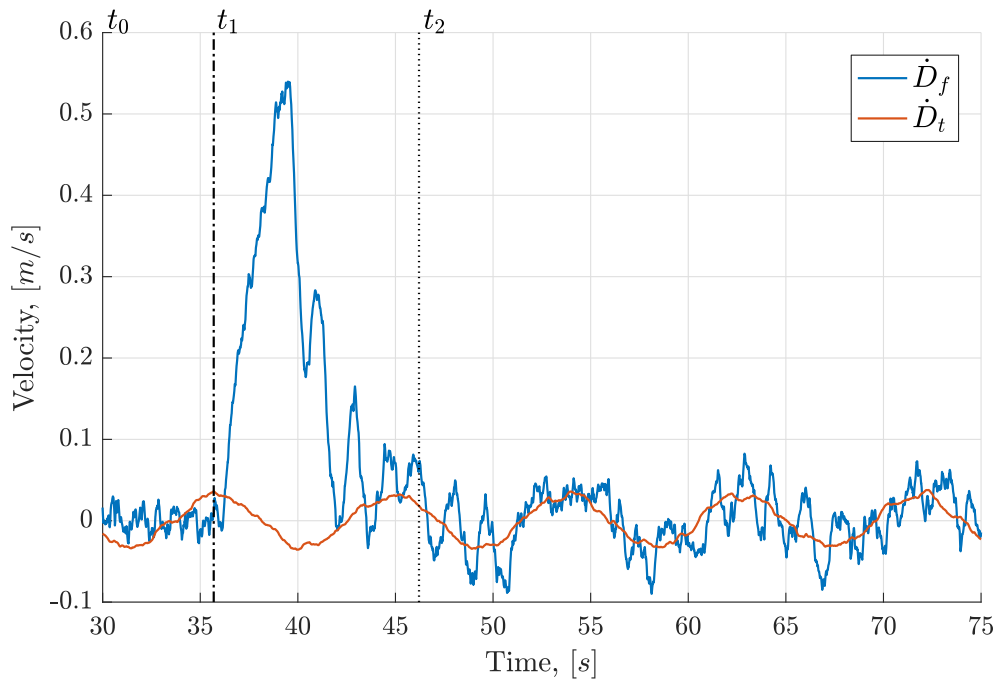


Figure 5.14: Follower and target velocity (QTO), non-ideal case.

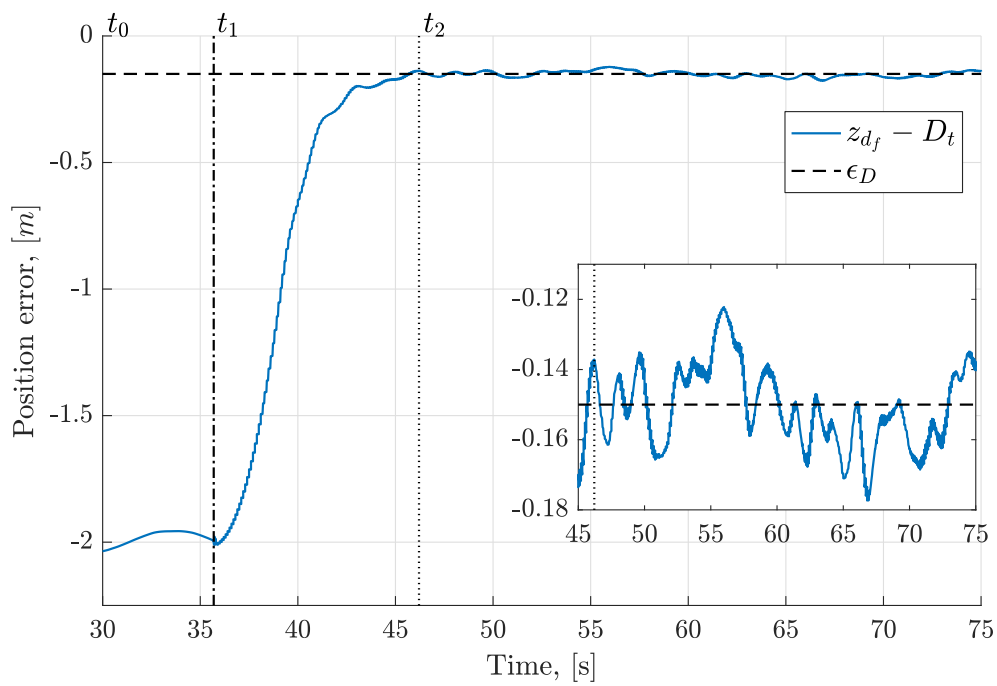


Figure 5.15: Landing trajectory error (QTO), non-ideal case.

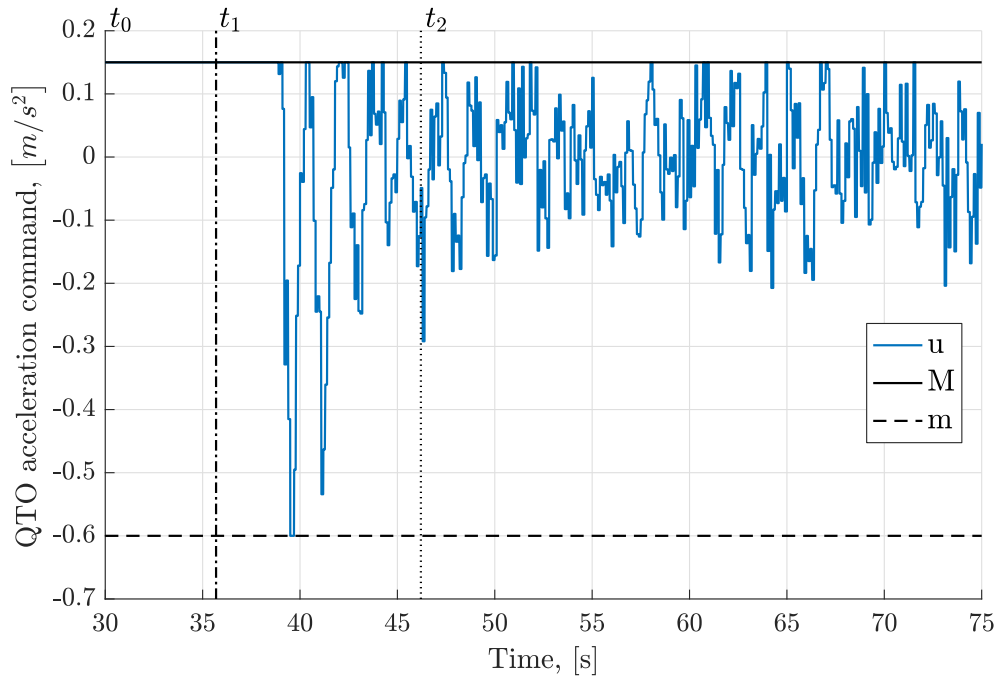


Figure 5.16: Acceleration command (QTO), non-ideal case..

assert that also this control law, as the three-states bang-bang, is more sensitive to the velocity noisy data instead the position ones, even if the time to descent t_{go} is not so affected. It is worth reminding that the global convergence of the control law is provided by the nonlinear term in equation (4.21), which is relevant in the initial part of the maneuver. Instead, close to the target, the local convergence is provided only by the linear term ensuring the critical damping. As a results, dealing with significant velocity errors (Figure 5.14) and negligible position ones (Figure 5.13), the guidance law rejects the noise disturbance in a longer time with respect to the ideal case.

5.4 Conclusions

In this chapter, the results of simulations in ideal and non-ideal conditions have been presented. As expected, there are some differences between the two cases.

Looking at the three-states bang-bang algorithm, one can notice that the time to perform the landing maneuver decreases when noise and discrete set-point are considered. This is imputable to the noise added to the state estimates, specifically for position and velocity. The actual state measurements are used only as initial conditions for the double integration, and the open-loop nature of the algorithm means that the added noise could lead to faster or slower descent trajectories, according to the random value assumed by the velocity error at the start of the descent. Even this may be seen as a disadvantage, the control law rejects this initial perturbation and provides always a safe, neither steep nor aggressive landing maneuver.

On the other side, QTO algorithm does not show the same drawback but the trajectory generated is steeper and faster than the one provided by the first guidance law. Unfortunately, the velocity error is not fully compensates locally when approaching to the target, and, if one has to deal with greater noise standard deviations, the landing may be aborted. On the other hand, if the landing maneuver happens without any wind-up behavior, the time needed to conclude the procedure is almost equal in the two cases.

One may argue that the saturation levels of QTO, M and m , and the constraints of three-states bang-bang, \dot{z}_{rmax} , \ddot{z}_{rmin} and \ddot{z}_{rmax} , play a crucial role. Unfortunately, these limits are not comparable. As a matter of fact, in the first algorithm \ddot{z}_{rmin} and \ddot{z}_{rmax} are the only possible values of the acceleration used to compute the position set-point, while in the second one the acceleration could assume each value inside the range $[m; M]$. Furthermore, in the three-states bang-bang the decision process is based on a future test trajectory, this implies that the lower is \ddot{z}_{rmax} the earlier the upward acceleration starts and the longer is the time needed to perform the landing (see Section 4.2). In addition, the QTO guidance law is a closed-loop nonlinear state feedback control, so the acceleration command continuously feels the position and velocity errors during the landing, and it does everything to rapidly decrease the errors. The time t_{go} mainly depends on the descent saturation level M , which it cannot be small as \ddot{z}_{rmin} . In fact, the nonlinear

part of the control law prevails on the linear one when the follower is far from the target, and the acceleration reaches its maximum value. Truly, both saturation levels must be chosen to guarantee the local convergence too, whenever the target trajectory is vertically oscillating.

Chapter 6

Experimental activity

This chapter is divided into three sections. In the first one the system architecture is presented: firstly the hardware used and then the respective software adopted. The central section illustrates the design constraints for the landing pad construction, and the adopted solution to face both mechanical and aerodynamic requirements is proposed. The last part of the chapter shows the experimental results of the air-to-air automatic landing between two multirotors, making use of both guidance laws previously tested in simulation environment. The chapter ends with the evaluation of the experimental results, comparing the performance of the two guidance laws.

6.1 System architecture

6.1.1 Hardware

The hardware used to run the system is heterogeneous and here is provided a detailed description of the machines involved in the project.

6.1.1.1 Ground station

A desktop pc, whose characteristics are listed below, work as ground station:

- Processor: Intel Core(TM) i56500 CPU @ 3.20GHz
- Memory: 16 GB
- Storage: 230 GB
- Network: Intel Gigabit CT Network Adapter

On this machine, we will run an Ubuntu virtual machine which has the following specifications:

- Processor: 1 Core
- Memory: 4 GB
- Storage 25 GB
- Network: Virtual adapter

6.1.1.2 Motion capture

The motion capture system is Motive Optitrack [23], we use eight Opitrack Prime 13 cameras [24], arranged on a square. The cameras define a flight volume of 25 m^3 and the drones can fly without obstacles inside it, see Figure 6.1.

The cameras are connected to a Netgear Prosafe 28PT GE POE [25] switch through Gigabit Ethernet cables.

In order to be seen by the Optitrack system, every drone must be equipped with markers set in different configurations, which differentiate the drones from each other.



Figure 6.1: Photo of the cage with Optitrack cameras on top

6.1.1.3 Flight control unit

The flight control unit (FCU) adopted is the Pixfalcon, which belongs to the family of the Pixhawk [26]:

- Main SystemonChip: STM32F427
 - CPU: 180 MHz ARM Cortex M4 with single precision FPU
 - RAM: 256 KB SRAM (L1)
- Failsafe SystemonChip: STM32F100
 - CPU: 24 MHz ARM Cortex M3
 - RAM: 8 KB SRAM
- Wifi: ESP8266 external
- GPS: U-Blox 7/8 (Hobbyking) / U-Blox 6 (3D Robotics)
- Connectivity:
 - 1x I2C
 - 1x CAN (2x optional)
 - 1x ADC
 - 4x UART (2x with flow control)
 - 1x Console

- 8x PWM with manual override
- 6x PWM / GPIO / PWM input
- S.BUS / PPM / Spektrum input
- S.BUS output

6.1.1.4 Companions

Intel Edison The companion computers are of two types. The first kind is the Intel Edison [27], which is a general purpose computer with the following specifications:

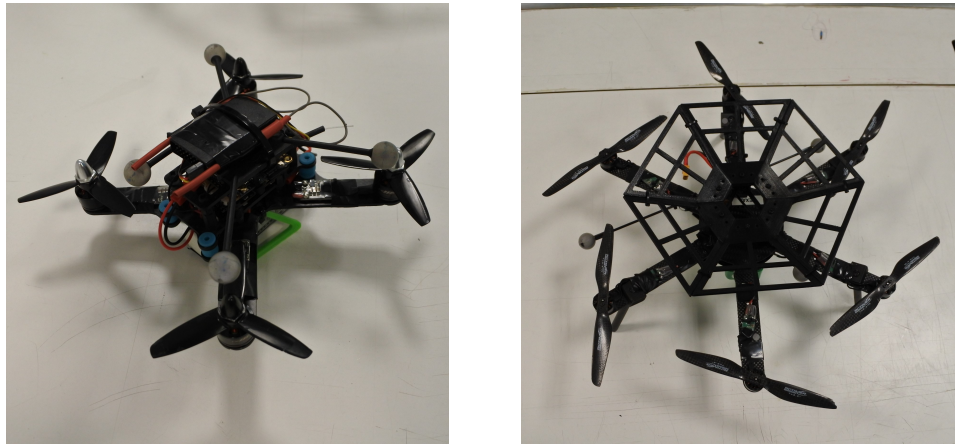
- Atom 2Core (Silvermont) x86 @ 500 MHz
- Memory: LPDDR3 1 GB
- Storage: 4 GB EMMC

Raspberry Pi Zero The second one is Raspberry Pi Zero [28], with the following specifications:

- Processor:
 - Broadcom BCM2835
 - contains an ARM1176JZFS (ARM11 using an ARMv6architecture core)
- Memory: 512MB LPDDR2 SDRAM
- USB OnTheGo port
- Mini HDMI
- 40pin GPIO header
- CSI camera connector

6.1.1.5 Drones

The two multirotors involved in the mission adopt the same general configuration, but have different hardware. Indeed, they are equipped with a flight control unit connected to a companion microcomputer by the serial port. The microcomputer communicates with the ground station through the Wi-Fi connection.



(a) Ant-1 drone.

(b) Hexa drone.

Figure 6.2: UAVs adopted in the flight tests.

Item	Value	Unit of measure
Rotor Radius	35	mm
MTOW	0.200	kg
Frame Diagonal	160	mm

Table 6.1: Ant-1 model main parameters.

Ant-1 This drone is a quadrotor in X-configuration, *i.e.* the principal axes of body frame are not aligned with motor arms, and it is the follower drone in the mission. The multirotor is illustrated in Figure 6.2(a). The frame is cut from a laminated carbon fiber plate of 3mm thickness. It is equipped with Raspberry Pi Zero and PixFalcon. The main parameters of the machine are summed up in Table 6.1.

Hexa This drone is an exacopter, *i.e.* it has six coplanar rotors at 60° one from the adjacent ones, see Figure 6.2(b). The frame is cut from a laminated carbon fiber plate of 3mm thickness too. It is equipped with Intel Edison and PixFalcon. It refers to the target UAV in the operation, and the main parameters of the drone are listed in Table 6.2.

Item	Value	Unit of measure
Rotor Radius	100	mm
MTOW	0.9	kg
Frame Diagonal	400	mm

Table 6.2: Hexa model main parameters.

6.1.2 Software

We now list the software adopted to execute the algorithm in a real environment. The principal software used to manage the distributed architecture is ROS Kinetic Kame [29]. ROS is a robotic middleware with a structure which is mainly publisher-subscriber which can manage more machines in a distributed environment. The central part of the ROS architecture is a node called ROS core, which manages the topics of the system and the subscriptions. The ROS core offers also other functionalities, such as the Parameter Server or the possibility to advertise services. The Parameter Server is a central infrastructure, which is responsible for storing configuration parameters loaded by the nodes of the system. These parameters can be retrieved by other nodes and used if necessary. Instead, a ROS service is a sort of remote function call. One node can advertise the service, which can be called by any other node. The call is synchronous, so the caller is blocked until the caller has executed its callback function. The ROS architecture is based on queues, threads and callback functions, but most of the provided tools hide part of the implementations of the distributed environment.

6.1.2.1 Ground station

The ground station runs Windows 10 Pro [30] and the software used to virtualize a Desktop machine is VMware [31]. On the virtual machine is installed Ubuntu 16.04 LTS [32] in order to run software needed and available only for Unix systems.

On Windows operating system we launch the Motive Optitrack software [23], which allows to calibrate and control the cameras for position tracking. It then provides the streaming of the positions of the markers identified by the cameras and sends it to the Ubuntu operating system using a multi-cast IP address. Here, the information is converted by a ROS node and sent through the ROS topics, which are read by the drones. In this way, each drone knows exactly its position.

This conversion node is an open source node called Mocap which can be found on GitHub [33]. On Ubuntu side, we launch the ROS core, which manages all the ROS nodes and topics.

6.1.2.2 Raspberry Pi Zero

The Raspberry Pi Zero executes a dedicated version of Debian operating system, which is Raspbian. The version used is Raspbian Jessie 4.4 [34].

6.1.2.3 Intel Edison

The Intel Edison runs a version of Debian called Jubinux, at version 0.1.1 [35].

6.1.2.4 Companions

Both companions, the Raspberry and the Edison, are provided with ROS Kinetic and both have to execute some ROS nodes in order to communicate with the other drones.

Both of them run Mavros nodes [36] which can be downloaded from GitHub and manage the conversion of the information taken from the ROS topics to the serial port and vice versa. Indeed, the ROS messages are converted into Mavlink messages and sent through the serial port to the Pixfalcon autopilot. The same is done for the Mavlink messages from the autopilot, which are published on ROS topics.

The second kind of ROS node run by the companions is a custom consensus node, which loads the desired trajectory and sends the next set point to the Mavros node.

6.1.2.5 Pixfalcon

The Pixfalcon FCU is flashed with PX4 Pro Autopilot [37], an open source firmware downloadable from GitHub. The release used is the v1.5.5.

6.1.2.6 Additional software

We use Matlab R2016B [12] to process the data, to plot the graphs and to validate some theoretical results. This document is written in LATEX [38], while the versioning control platform used are GitHub [39] and GitLab [40].

6.2 Landing pad design

It is clear that make the follower UAV land on the target one is not a straightforward issue. The practical execution of the air-to-air automatic landing maneuver does not deal with only the trajectory generation issue, but also with the necessity to have a support which land onto. For this reason, a landing pad is created to assist in the procedure.

The design requirements are divided in two fields:

- mechanical,
- aerodynamic.

The mechanical ones are:

- avoid interference between the rotors,
- the structure must be stiff enough to withstand the impact at the touchdown and support the follower weight,
- furthermore, it should be light, axisymmetric and of low height in order not to affect the CG position of the target drone,
- the shape of the pad should be concave to keep the follower inside, once it is landed. Moreover, this let the target maneuver safely once the land is concluded.

On the other hand, in the final part of the approach the aerodynamic requirements come into play, namely the follower should stop far enough from the target:

- to avoid “ground effect” caused by the pad surface,
- to minimize the aerodynamic disturbances due to the propellers wake.

As always, the requirements are conflicting, so a compromise is needed. The mechanical requirements are easily satisfied by realizing the structure thanks to the use of 3D printing. The material used is polylactic acid (PLA), a common commercial, low cost, biodegradable and bio-active material. This guarantees very light structure, considering also the low weight of the follower UAV (see Section

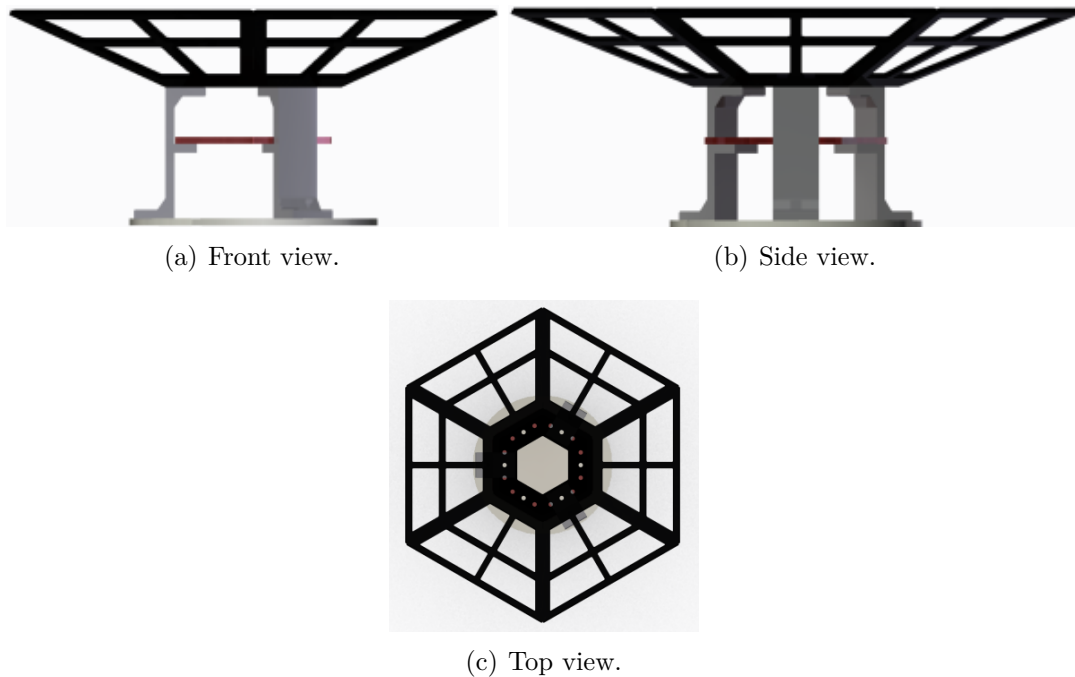


Figure 6.3: Landing pad CAD assembly.

6.1.1.5), and availability. The height selected is about 65 mm , giving priority to maintain the CG of the target drone as close as possible to the nominal position.

It is necessary to describe the final part of the maneuver to address the aerodynamic interaction problem. For clear safety reason, the land should happen in the most safe way, neither dangerously hitting nor bouncing onto the landing pad: the follower should stop and hover slightly above the pad, then the thrust-off command is sent and the follower lands. While hovering above the pad, the wake exiting from the follower rotors impinging on the pad surface is a source of disturbances for the target. On the other hand, when being close to the platform, the “ground effect” may influence the follower actual position, destabilizing the final crucial operation. These effects are limited by piercing the surface. Their minimization is improved by larger holes, which let the follower propellers wake flow downward. Due to the axis-symmetry of the pad, this airflow equally affects the thrust of each rotor of the target, according to what shown in Section 2.5.2, reducing the disturbances caused by the airflow. The landing pad designed is depicted in Figure 6.3.

6.3 Experimental results

In this last section, experimental results are shown. In Section 5.1 the target was asked to perform a vertical oscillatory motion just to have a fictitious representation of the perturbed flight condition during the whole operation. In the flight test, on the contrary, the target drone is asked to stay in hovering at constant altitude. For what concerns the follower, the maneuver is exactly equal to the one simulated, *i.e.*: take-off, positioning above the target, move to the target in-plane position and land on it.

The trajectory generation module needs both position and velocity of the two multirotors and the acceleration of the target one. Unfortunately, the measurement of acceleration is too noisy to be used in the landing control law (see also Section 5.3.1). So, the target acceleration is assumed to be null, during the whole procedure. Instead, the velocity estimate, available on-line, comes from an on-board filter, fusing the position measurements and the IMU data. Lastly, the position measurement is directly taken from the motion capture system.

In the figures regarding the flight tests, a time t_3 is indicated. It is defined as the time instant in which the thrust-off command is given.

6.3.1 Three-states bang-bang

In [2], the target platform motion was estimated using an UKF, assuming it is sinusoidal. On the contrary here the position raw measurement is used to feed directly the algorithm, together with the velocity estimate, obtained as described above. The parameters of the algorithm are the one used in simulation environment, see Table 5.2. Since the acceleration of the target is not taken into account, it results: $\ddot{z}_a = \ddot{z}_r$. Clearly, only \ddot{z}_a is reported in Figure 6.6(c).

In Figure 6.4, the down position of the two drones are reported, together with the desired final altitude. The time to complete the land is higher than in simulation, *i.e.* $t_{go} = 52$ s. The reason lies in the absence of the acceleration term, \ddot{z}_d . Since the acceleration is bounded by the constraint values $[\ddot{z}_{rmin}, \ddot{z}_{rmax}]$, the landing trajectory has a smaller slope so the maneuver will be slower and the time-to-land will obviously increase. Additionally, it is evident that the target drone is flying in a perturbed airflow field during the whole landing procedure. Furthermore, the aerodynamic interaction between the two is visible in the final

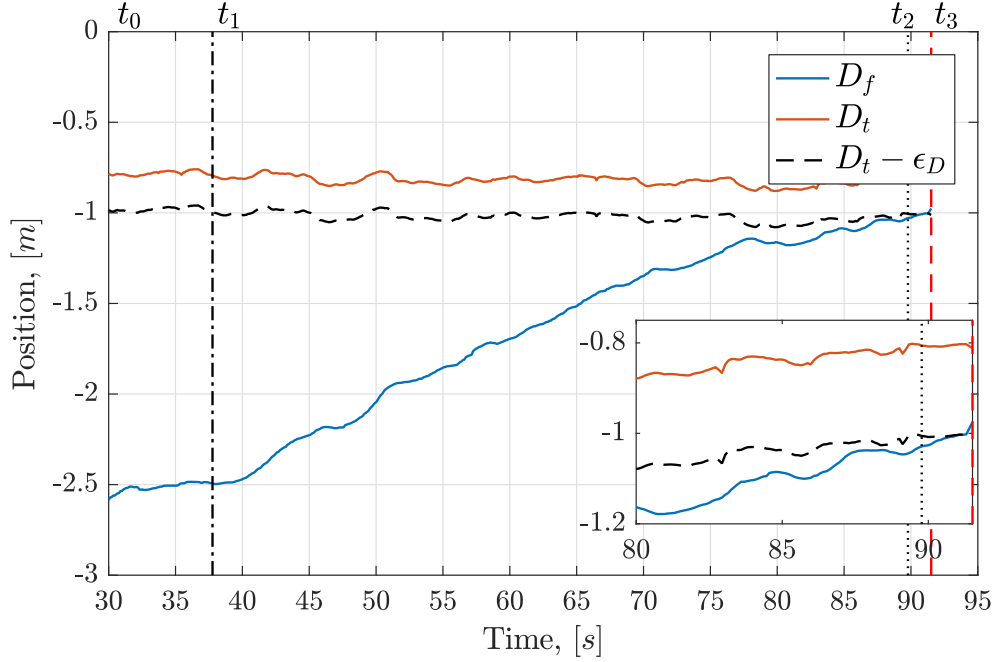


Figure 6.4: Follower, target and desired altitude (bang-bang), flight test.

part of the approach, where the target drone suddenly moves upward. The maneuver is successfully completed even though the velocity estimates have a little bias and are quite noisy, specially for the target multirotor. See Figure 6.5. In Figures 6.6(a), 6.6(b) and 6.6(c), the control variables of the algorithm, z_a , \dot{z}_a and \ddot{z}_a are reported. They respects perfectly the theory illustrated in Section 4.2. It is evident that the integration of the acceleration without the target one \ddot{z}_d is smoother and devoid of oscillations. From Figure 6.6(c), one can see that the downward and the null acceleration commands are given for a very limited amount of time, while the deceleration one is prevailing during the descent. This behavior perfectly reflects what explained in the theory and in the simulation, *i.e.* a small value of \ddot{z}_{rmax} implies the deceleration command is given earlier and for a longer time. Now, looking to Figures 6.7 and 6.8, the reference position and velocity are plotted. Referring to Figure 6.7, the aerodynamic effects described in Figure 6.4 are once again evident, and when the target drone suddenly moves upward the set-point adapts consequently. In Figure 6.8, the reference velocity \dot{z}_r exceeds a little its maximum limit \dot{z}_{rmax} since the algorithm works in discrete time. Comparing Figures 6.6(b) and 6.7 it is possible to observe the mismatch in the sign of \dot{z}_a and \dot{z}_r respectively. This is caused by the large noise affecting the

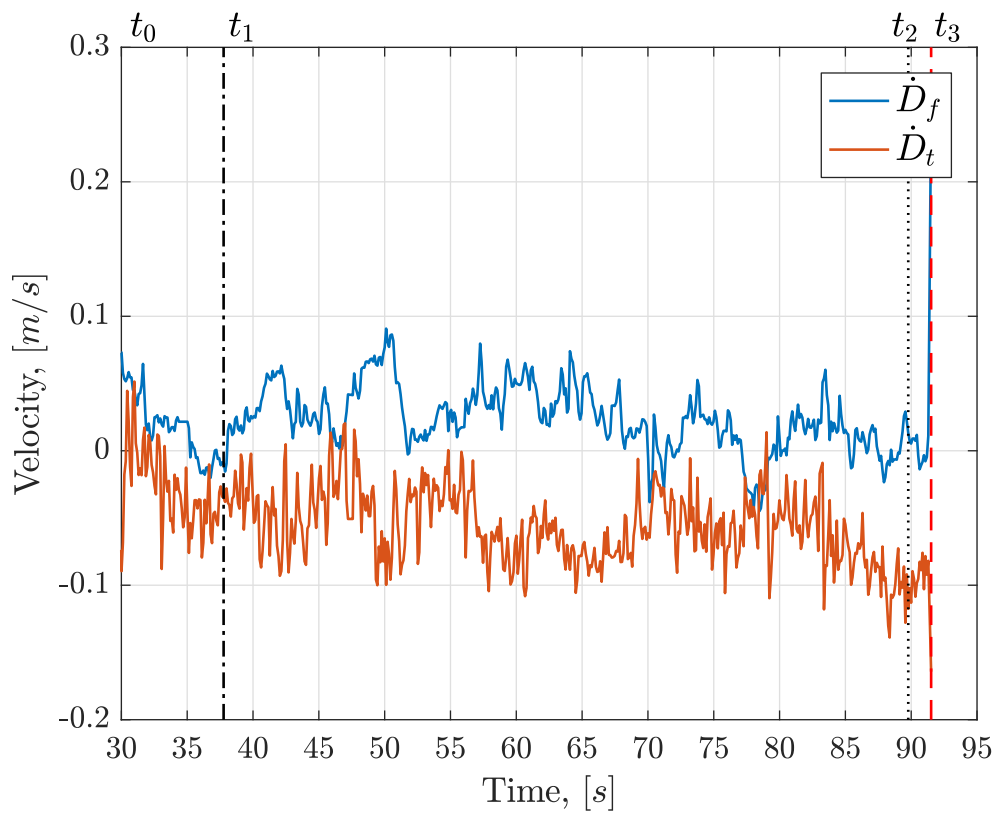


Figure 6.5: Follower and target velocity (bang-bang), flight test.

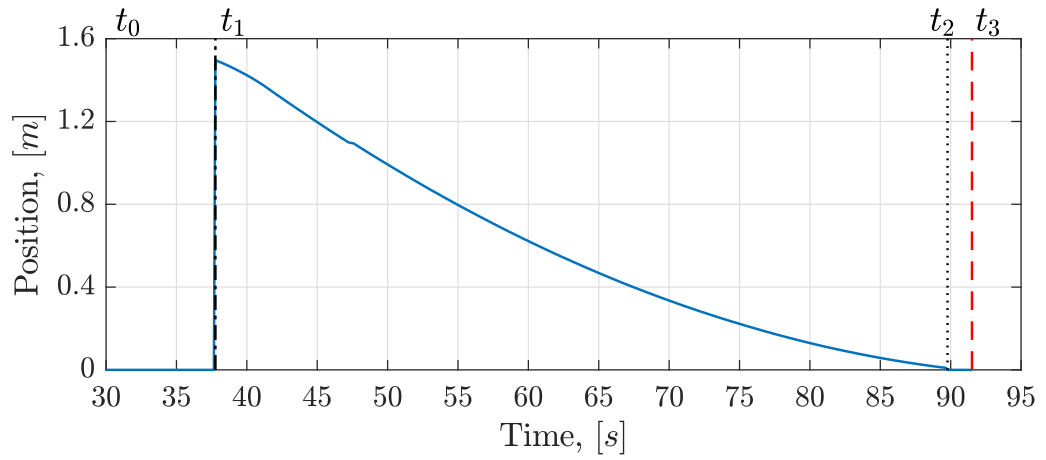
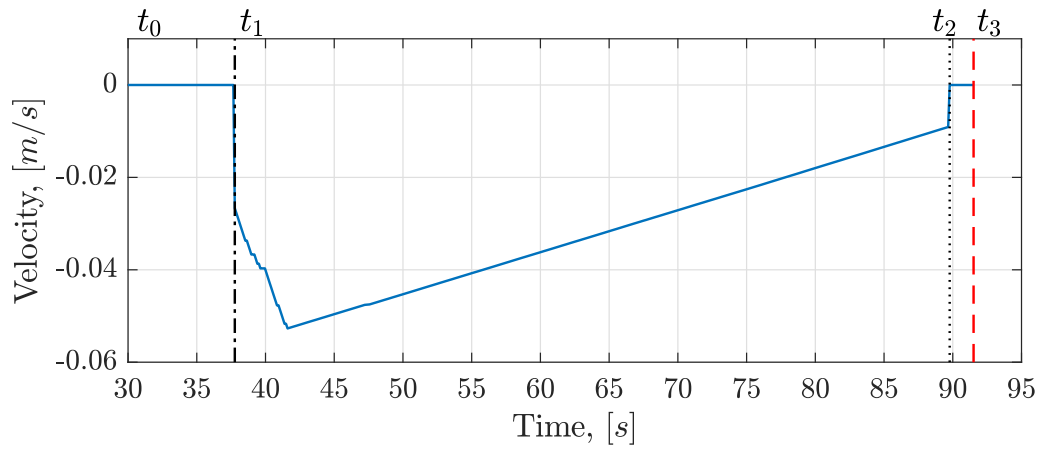
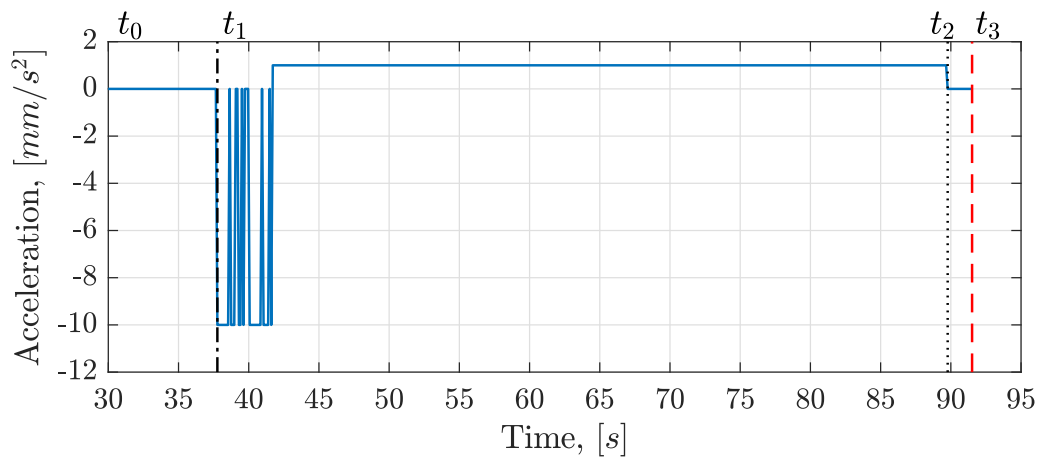
(a) z_a .(b) \dot{z}_a .(c) \ddot{z}_a .

Figure 6.6: Three-states bang-bang control variables.

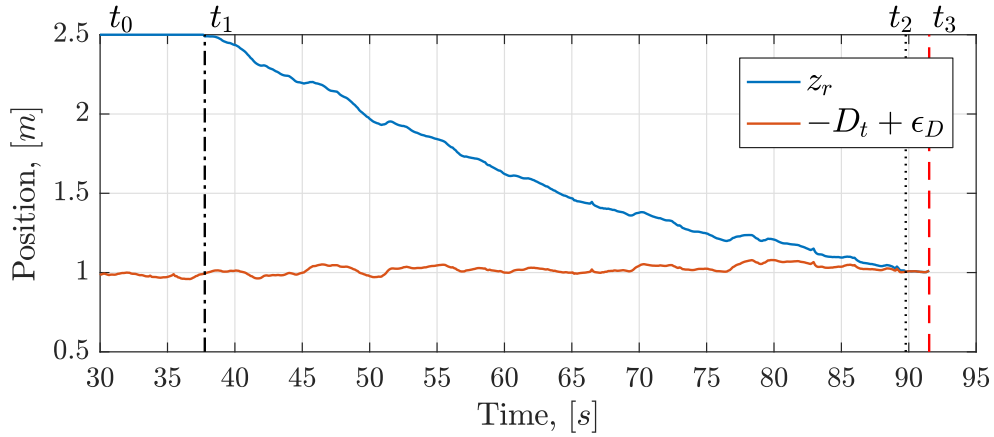


Figure 6.7: Three-states bang-bang reference position, z_r .

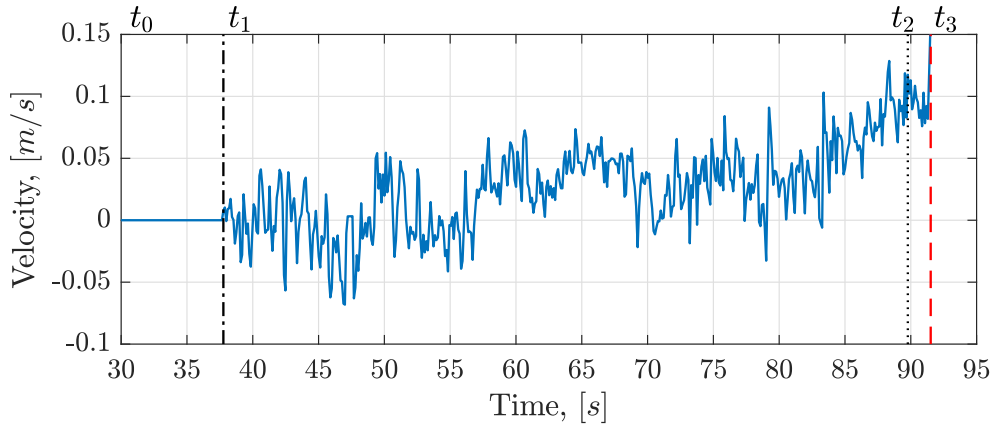


Figure 6.8: Three-states bang-bang reference velocity, \dot{z}_r .

velocity estimate of the target drone, reported in Figure 6.5. The advantage of using only relative quantities and adding the target position z_d just at the end, is that this guidance law is unaffected by the unsatisfactory estimate of the target velocity.

For what concerns the error monitoring, in Figure 6.9 the norm of the average errors, \bar{e} , is reported in the time interval $[t_0; t_3]$. The dashed line indicates the ϵ bound, described in Section 3.3. From the first time \bar{e} does not exceed the bound, the means are monitored for the time interval T_m , indicated between the dotted lines, and if they respect the ϵ limit, the safety flag switches on. The norm of the average errors shows a clear tendency to go towards ever smaller values. In addition, in the top-right part of the figure, the in-plane positions of the follower

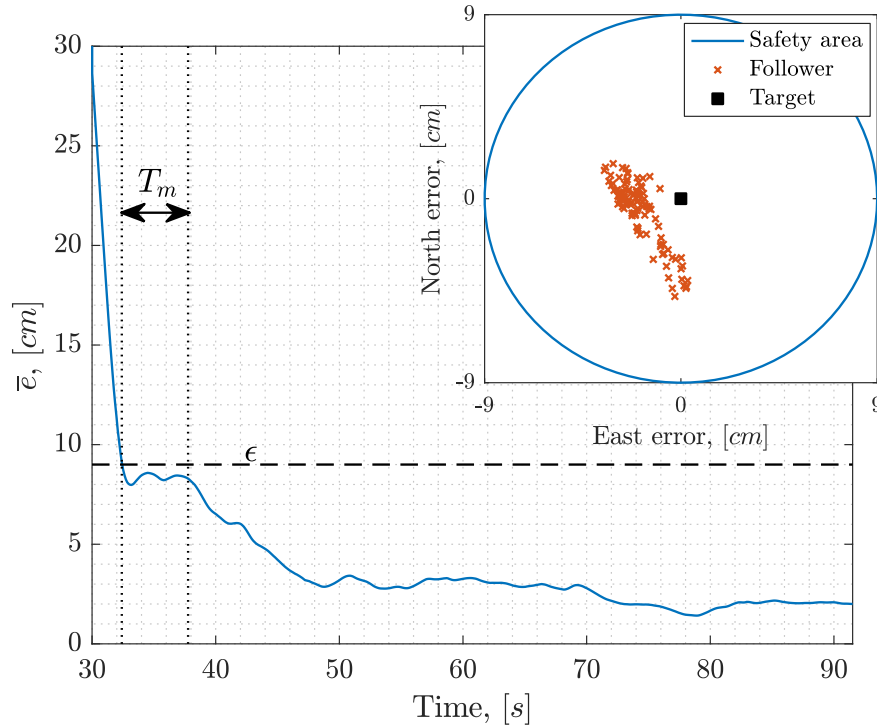


Figure 6.9: Error monitoring procedure (bang-bang), flight test.

in the same time interval are plotted over the safety region. Lastly, an extract of another test is reported in order to clarify the safety procedure in case of failed synchronization. In Figure 6.10, one can see what happens when the safety flag returns to zero value after the starting of the landing maneuver, *i.e.* the norm of the actual in-plane position errors exceed the chosen bound. In this situation, the set-point is “frozen”, namely it is assumed equal to the last valid value. This is observably in the region between the dotted lines.

6.3.2 Quasi time-optimal

In the experimental tests of the quasi time-optimal algorithm, some problems have been encountered, despite it perfectly works in simulation environment. The major problem again is the velocity estimate of the target, because it is biased and very noisy. Due to the closed-loop nature of the guidance law, receiving both position and velocity errors in feedback, the convergence of the algorithm is not granted. With this in mind one may seek to tune again the relevant parameters of this guidance law, namely ξ , ϵ and K_p . Referring to the non-ideal environment

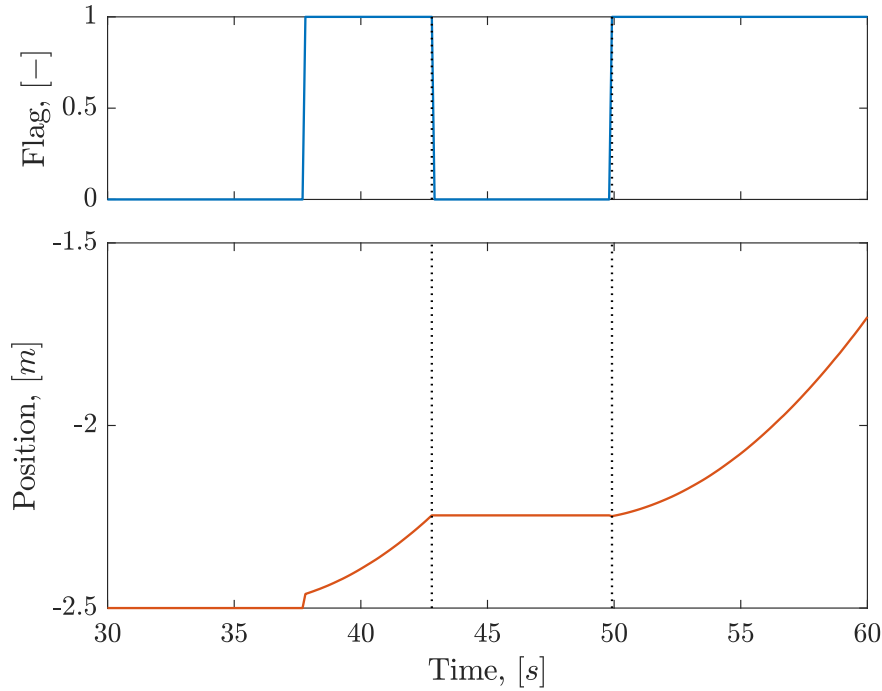


Figure 6.10: Synchronization failure (bang-bang), flight test.

Quantity	Standard deviation	Unit of measure
\dot{n}	0.05	$[m/s]$
\dot{e}	0.05	$[m/s]$
\dot{d}	0.1	$[m/s]$

Table 6.3: Standard deviation of velocity estimate.

simulations performed in Chapter 5 the standard deviations of the white noise related to velocity, listed in Table 5.3 can be rearranged to match the estimates provided by the on-board filters. The new values assumed are listed in the following in Table 6.3. Then a Monte Carlo simulation of the landing maneuver has been conducted on the controlled system with the uncertain values of the parameters. By evaluating per each simulation the amount of time in which the follower is located in the neighborhood of the desired position, namely $D_t - \epsilon_D$, the optimal parameters have been found. Moreover, for practical reasons, the downward acceleration limit M has been reduced. The new parameters involved in the control algorithm are listed in Table 6.4. After this additional tuning, a flight test has been made. The designed maneuver is the same described in Section 6.3.1. Because the landing reference trajectory is obtained by integrating

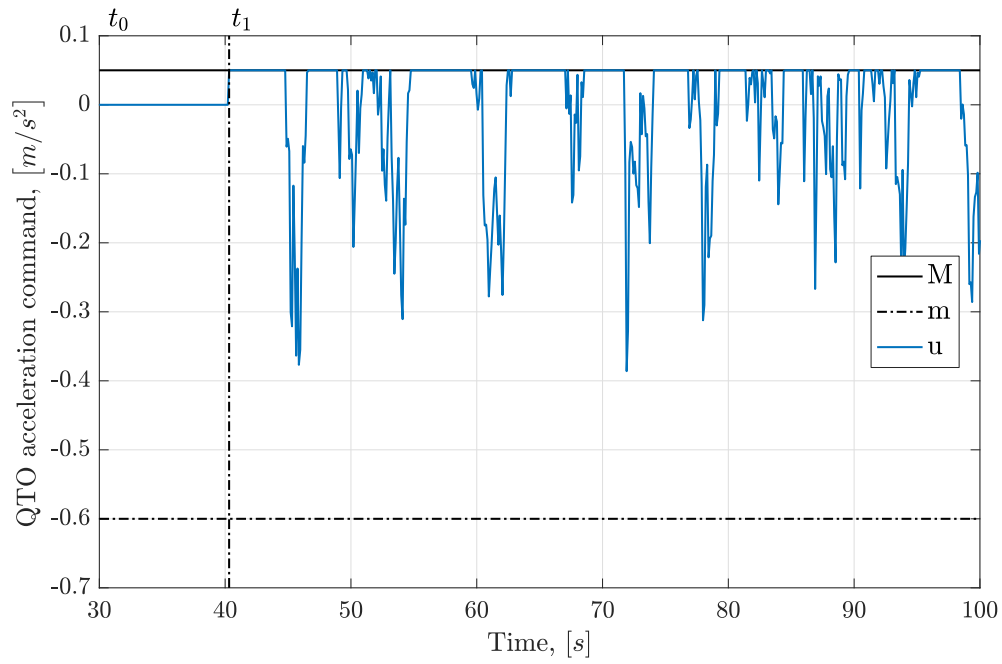
Parameter	Value	Unit of measure
K_p	0.89	$[1/s^2]$
ξ	0.35	$[-]$
ε	0.027	$[m]$
M	0.05	$[m/s^2]$
m	-0.6	$[m/s^2]$

Table 6.4: QTO new parameters.

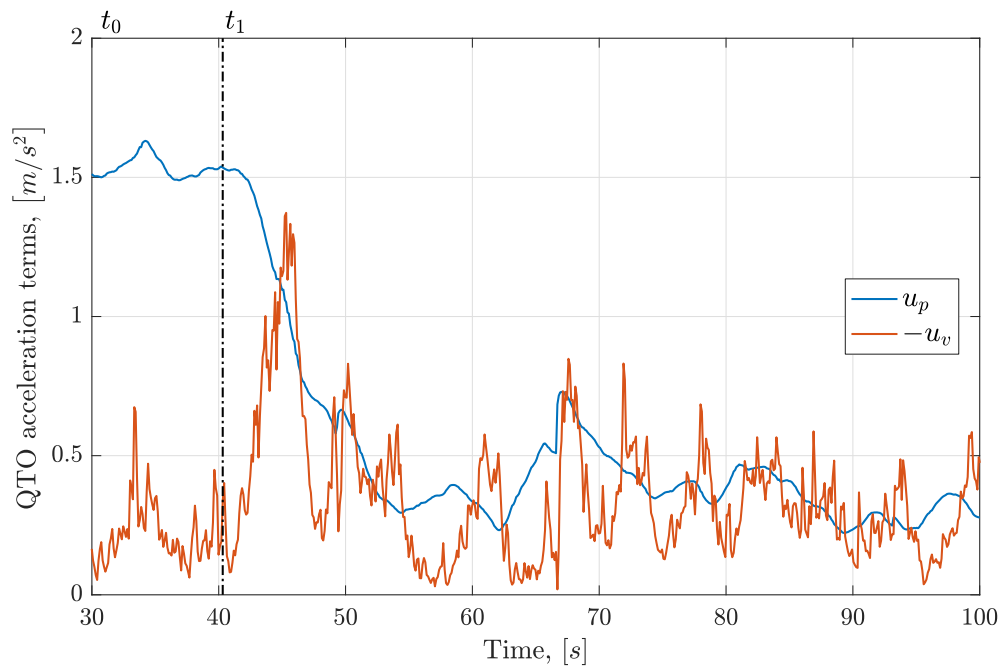
twice the acceleration command, the noise affecting the velocity estimates enters in the loop and degrades the performance of the algorithm. In Figure 6.11(a) the acceleration command is reported. When the follower is close to the target the acceleration terms related to position and the velocity errors often compensate each other, resulting in the acceleration command crossing many times the zero value. Recalling the definition of the acceleration terms declared in Chapter 5, respectively u_p and u_v , Figure 6.11(b) shows their anomalous behavior. Besides, the position and velocity errors are not minimized at all. As a consequence, the landing maneuver is not fully accomplished, and, the follower stops at a certain altitude above the target as depicted in Figure 6.12(a). In Figure 6.12(b) it is possible to see the nonsense values assumed by the target velocity estimate \dot{D}_t , which seems to stabilize around $0.1 m/s$, even if its altitude is almost constant.

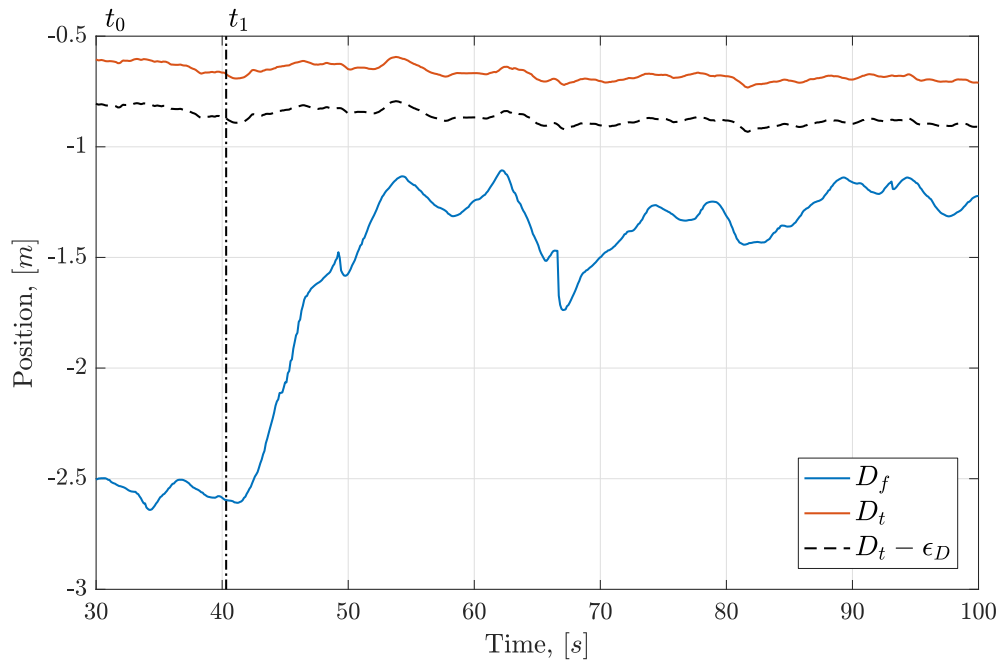
6.3.3 Velocity estimate issue

This issue has been investigated by analyzing the target on board IMU measurements when the target is still on ground. At the beginning, the follower is hovering far from the target, and then moves above it. Look at Figure 6.13(a): until 300 s the accelerometer measurement detects only the gravity field, but when the follower suddenly moves above the target, the acceleration sensor is influenced by the follower's propellers wake. Once the accelerations are integrated in the on board filter, the drift in the velocity estimate arise, as shown in Figure 6.13(b). That is the reason of the offset of the velocity estimate visible in Figure 6.12(b). Since no other velocity estimator was available for this kind of tests, to overcome this issue the algorithm has been modified to work without the target velocity.

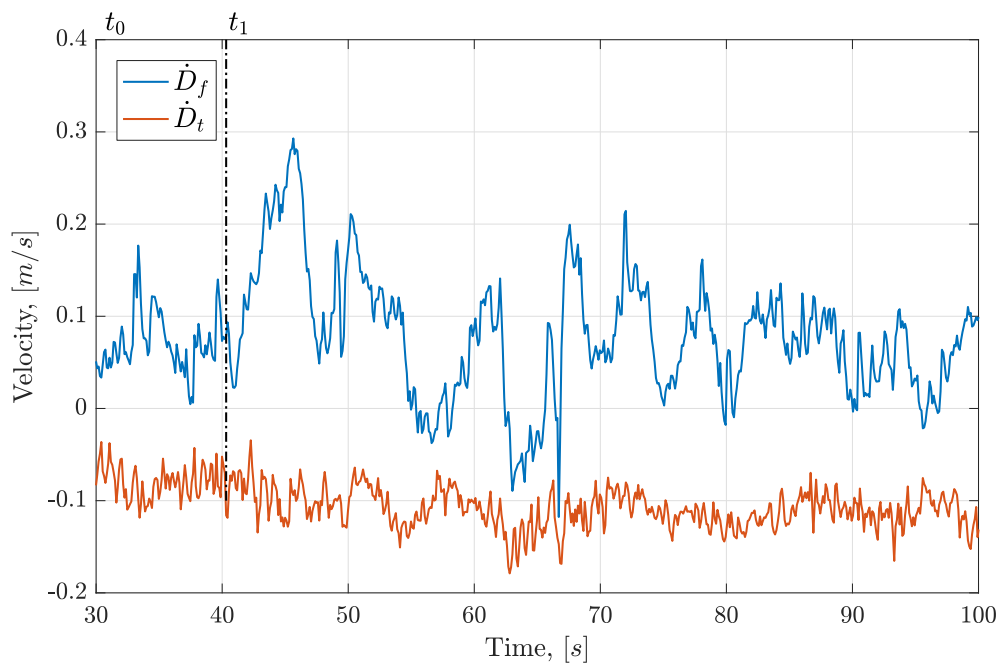


(a) Acceleration command.

(b) u_p, u_v .Figure 6.11: Acceleration command and u_p, u_v (QTO), unsuccessful test.

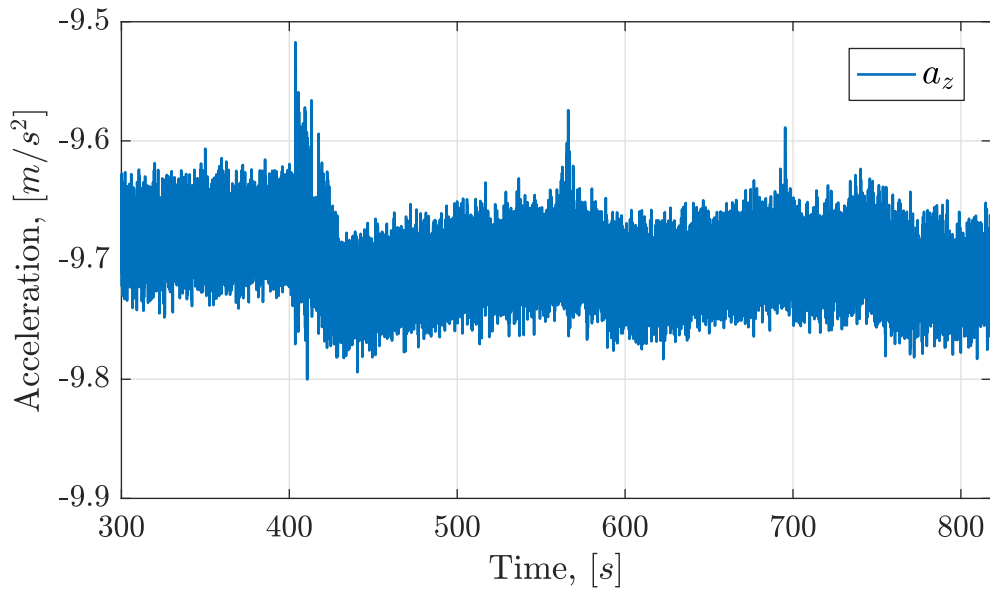


(a) Position.

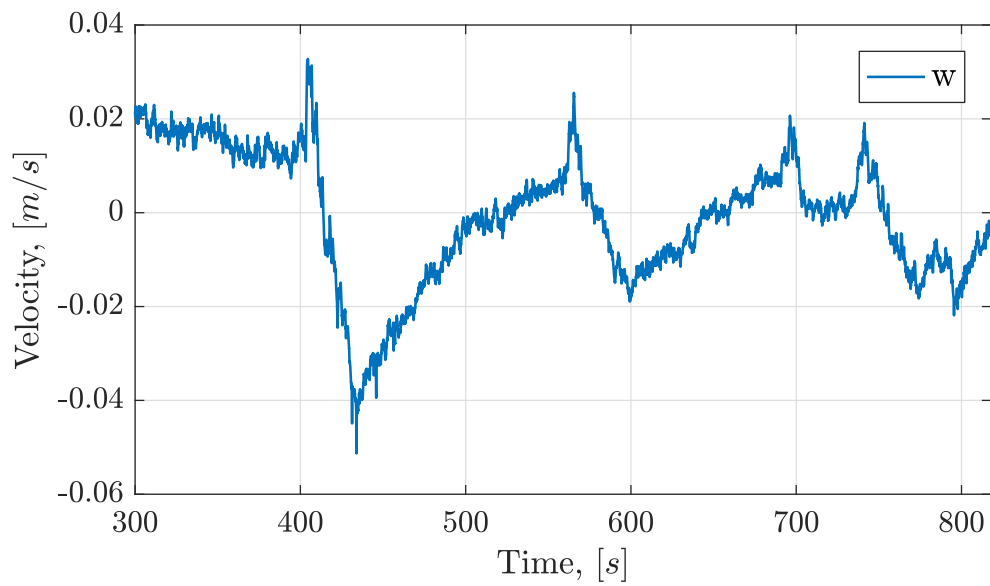


(b) Velocity.

Figure 6.12: UAVs position and velocity (QTO), unsuccessful test.



(a) Accelerometer measure.



(b) Velocity estimate.

Figure 6.13: IMU measurements and velocity estimate of the on board filter.

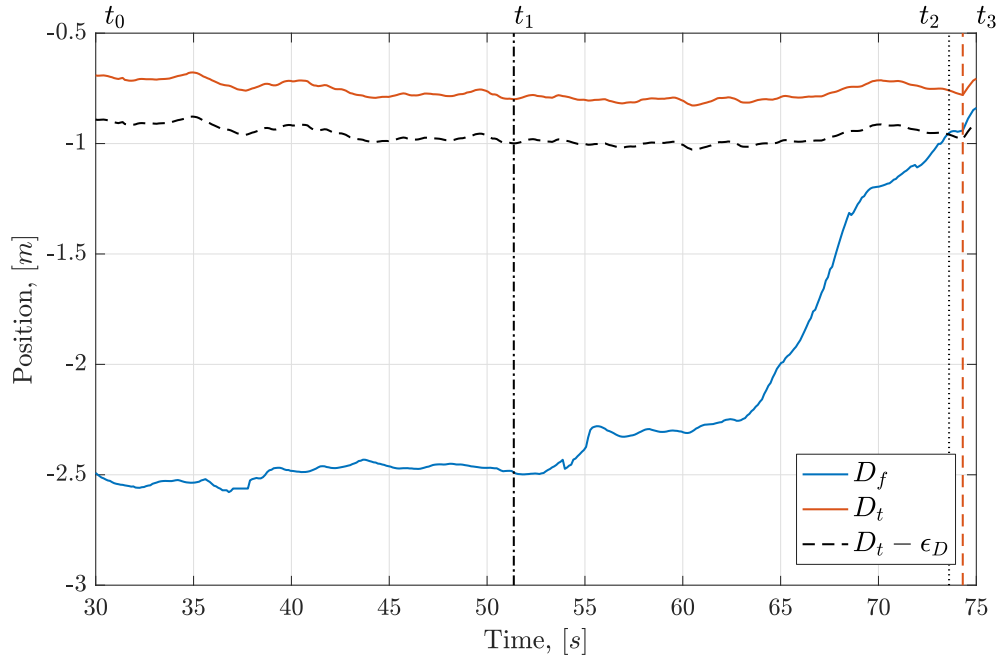


Figure 6.14: Follower, target and desired altitude (QTO), flight test.

6.3.4 Final flight test

Another identical test has been conducted by imposing a null target velocity in the control loop. Figure 6.3.4 depicts the altitude of both multirotors during the landing maneuver. Now the air-to-air automatic landing has been successful. The follower starts the descent at time t_1 but after few seconds it has to face a sudden synchronization failure. The locus of points of the in-plane position errors between the multirotors is depicted in Figure 6.3.4 as well as their average \bar{e} . With this safety check it is clear now the whole safety procedure: even if the average is acceptable, the safety flag is reset to zero whenever the follower exceeds the position error bound; the follower stops the descent, positioning above the hexacopter as expected. Looking at Figure 6.3.4 it can be appreciated that the acceleration command is reset to zero during this time frame. Then, the position error between the reference landing trajectory and the target altitude is shown in Figure 6.3.4, where the stop of the descent is evident between 55 s 60 s. Referring again to Figure 6.3.4, once the landing maneuver starts again, the follower approaches to the target drone, and at time t_2 it enters in the tolerance band ϵ_D .

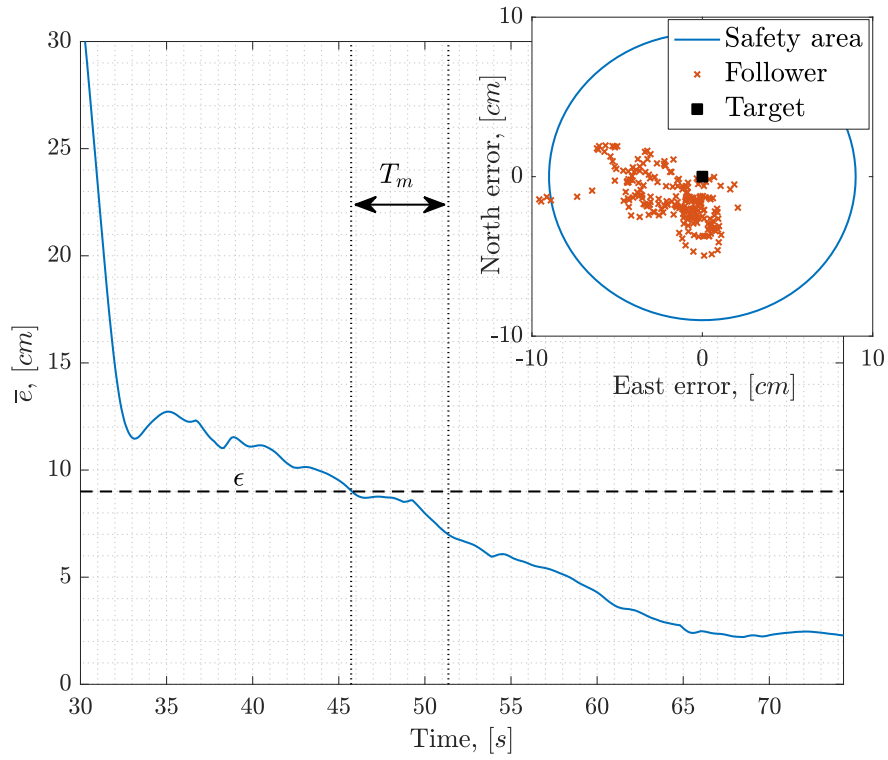


Figure 6.15: Error monitoring procedure (QTO), flight test.

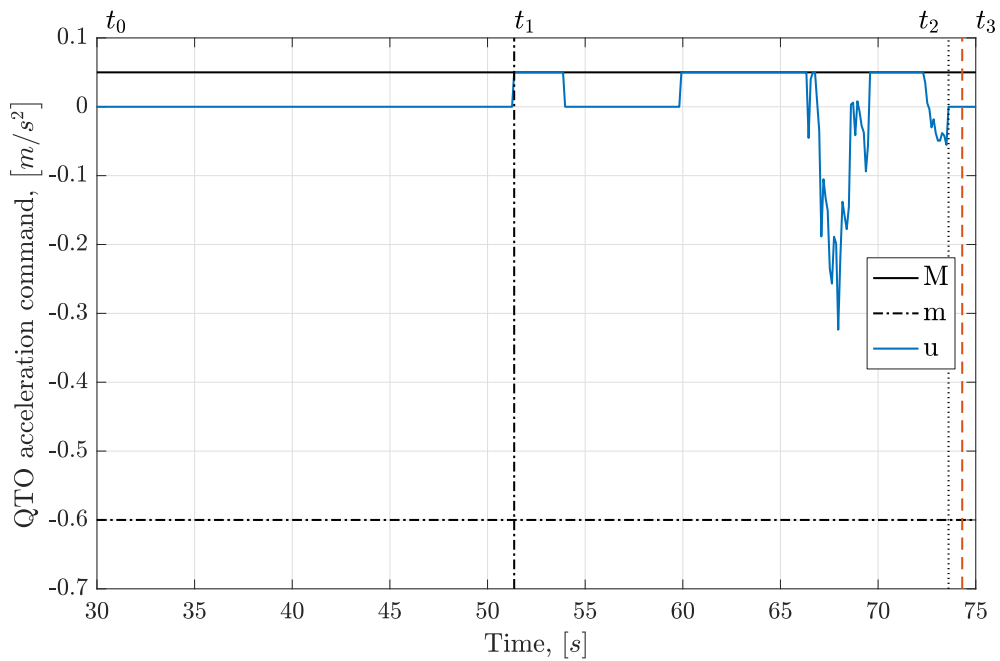


Figure 6.16: Acceleration command (QTO), flight test.

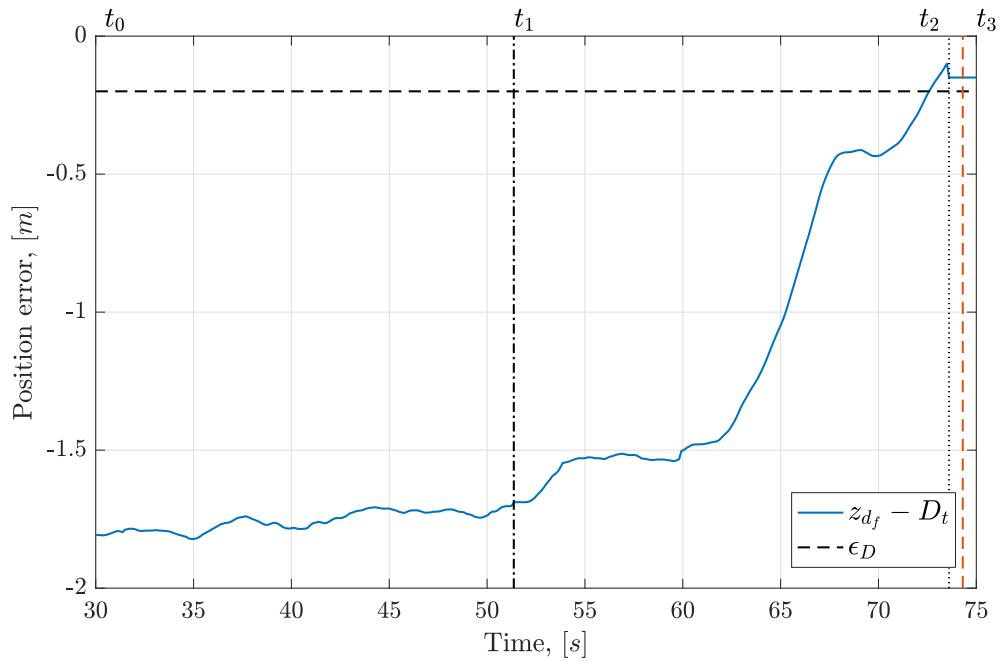


Figure 6.17: Landing trajectory error (QTO), flight test.

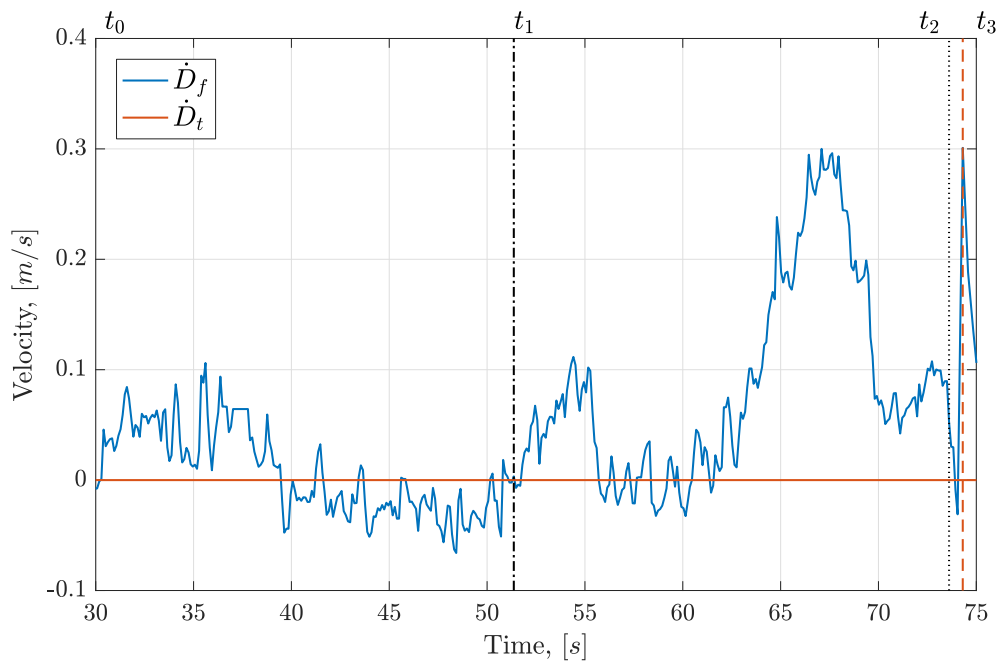


Figure 6.18: Follower and target velocity (QTO), flight test.

Similarly to what prescribed by the three-states bang-bang guidance law, after t_2 the landing set-point is fixed to the hexacopter altitude minus a positive tolerance, smaller than ϵ_D for practical reasons. Then, after waiting for the follower to hover for 0.5 s inside the bound, the thrust-off command is sent. The evidence of the touch down is clearly visible in the last seconds, looking at Figure 6.3.4. Finally, it can be appreciated that the the whole landing maneuver lasts almost 23 s , due to the synchronization failure occurred soon after the start. Anyway, if one considers just when the descent starts again at 60 s , the descent time is greater than the one predicted by the simulation results, hence $t_{go} \approx 15\text{ s}$. As one can imagine, this is obviously due to the reduction of the descent saturation level M .

Figure 6.19 shows the key points of the maneuver.

- (a) The two UAVs are taking-off.
- (b) The follower is positioning above the target, waiting for synchronization.
- (c) The follower is approaching the target.
- (d) The follower is hovering inside the bound ϵ_D , waiting for thrust-off command.
- (e) The thrust-off command is executed.
- (f) The target drone is landing with the follower aboard.

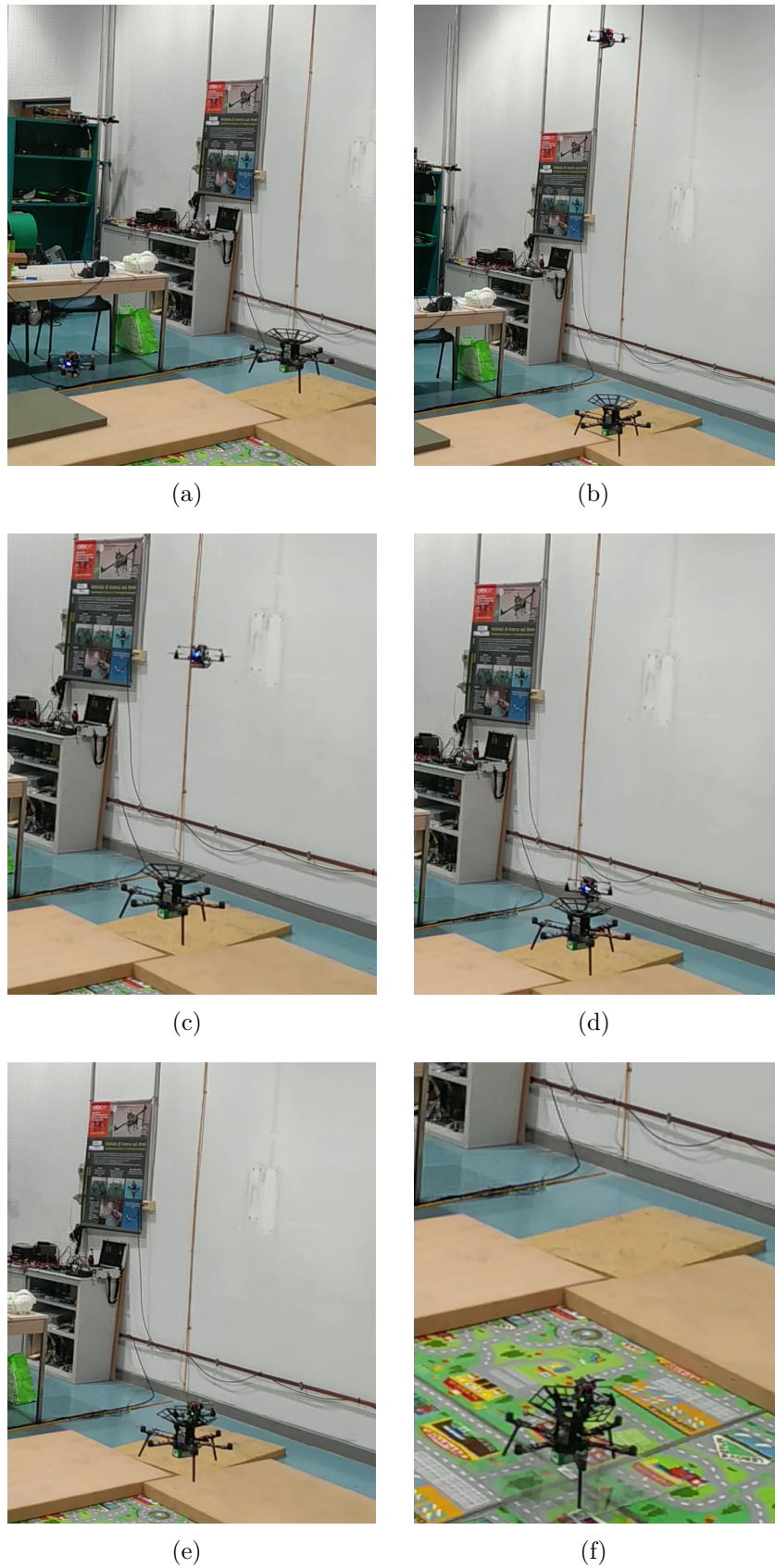


Figure 6.19: Air-to-air landing flight test.

6.4 Conclusions

The three-states bang-bang control concludes the landing even though the acceleration \ddot{z}_d is not considered. The actual position and velocity of the follower are taken just as initial conditions for the double integration, while the target ones are continuously used to generate the reference landing path. Since the velocity estimate is significantly noisy the initial conditions can randomly vary, and, as a consequence, the maneuver results to be slower than the simulated one. Instead, the quasi time-optimal guidance law needs to be fed back by both position and velocity errors during the landing maneuver. The problem of having noisy and rough velocity estimates in the closed-loop has been analyzed and initially faced by means of a new tuning of the control parameters via a Monte Carlo simulation. In the first attempt, the descent was not even completed, so a test campaign to identify the estimation problem regarding the target velocity has been conducted. Once the problem has been identified and referred to the on board estimator of the hexacopter, the algorithm has been modified and the task has been accomplished successfully. One may conclude that both guidance laws have been satisfactory for the landing purpose, providing the air-to-air automatic landing in two different manners. The three-states bang-bang algorithm generates a slow descent trajectory, and the control variable are unaffected by the poor performance of the on-board filters. On the contrary, despite the initial synchronization failure, the QTO algorithm provides an higher descent rate to the landing trajectory, remarking the time-optimality idea behind itself.

Chapter 7

Conclusions

In this thesis, the air-to-air landing maneuver for multirotor UAVs has been studied. The problem is of great interest in UAV operations, such as search and rescue and surveillance. The purpose of the work was to simulate, to implement and to experimentally validate a guidance law able to generate a landing reference trajectory. Two different solutions were proposed.

The conducted activities start with an introductory chapter, in which the modeling of a quadrotor is presented. Firstly, the formalisms and the classical aerodynamic theories for rotors, namely blade element theory, momentum theory and dynamic inflow, are recalled. Secondly, the model is built with an object-oriented multi-body approach, including the rotor aerodynamics, which influences thrust and drag forces. Then, the model is imported in Simulink, in which the controller and state feedback procedure is implemented, and a co-simulation is performed. This chapter ends with the evaluation of the simulation results, along with the experimental validation of the model.

In the third chapter, the air-to-air landing problem is introduced in general terms and all the investigated algorithms are listed and briefly explained. Afterward, the problem is formulated from a mathematical point of view. The adopted approach is presented and motivated. In the last section, the two suitable control laws, a three-states bang-bang and a quasi time-optimal one, are described.

In the fourth chapter, the digital implementation of the error monitoring module and of the trajectory generation module is addressed. For what concerns the latter, the two algorithms proposed are described in details.

The fifth chapter deals with the simulation of the guidance laws. The simulation is

performed by means of the tool described in the first chapter, simply extended to two UAVs. Initially, the desired trajectories are specified. To enhance the reality of the simulation, white noise on estimates feedback and set-point discretization are added to the model. Furthermore, in the case of the quasi time-optimal algorithm, a sensitivity analysis, regarding the choice of the relevant parameters, is performed and commented. Since only the Simulink model of the quadrotor was available, the two UAVs simulated are identical. Hence, the performance of the control laws are commented from a qualitatively point of view.

The final chapter reports the results of the experimental activity. At the beginning, a detailed description of the system architecture is provided. In the middle section, the landing pad design is shown. Finally, the flight test results for both control laws are presented and commented.

The realization of the automatic air-to-air landing maneuver, has been an enormous challenge. To help future developments, here some recommendations are gathered:

- the velocity estimate is of crucial importance. Therefore, high precision hardware is advised to obtain an higher quality estimator, *e.g.* IMU, when fusing it with a motion capture system measurement in a Kalman filter. Otherwise, one can filter the estimates afterwards.
- To complete the landing, a thrust-off command must be given. Hence, a landing pad is needed to avoid collision between the drones (rotors) in the case of not perfectly vertical land. Furthermore, the pad helps to keep the follower UAV inside, when the target is maneuvering.
- The major problems encountered regard the communication network (ROS), so it is suggested to find a lighter and more reliable middle software.

While working on this thesis, it was not sure that the results would be positive. Since the feasibility of the air-to-air landing has been proven, the future works can now focus on the improvement of performances. Some of the topics that is worth to develop are listed.

- Firstly, a pre-processing of the acceleration data is needed to avoid drift in velocity estimate. In alternative, the vertical velocity estimator could be tuned to make it robust with respect to the follower wake disturbance.

-
- The relative navigation problem could be addressed by means of an onboard camera, able to recognize a graphic pattern printed on the target drone, such as a QR code. This would make the full procedure autonomous.
 - The usage of a camera will also solve the problem of in-plane synchronization. Otherwise, a consensus control law could be implemented to perform independently the synchronization.
 - The most interesting application of this work is the search and rescue operation. It is reasonable to imagine that a fixed-wing or a carrier multirotor UAV will carry a certain number of lightweight sensors-equipped multirotors from the base of (the) operation to the site of the emergency; then, they will take-off, inspect the area, provide a first medical aid and come back and land on the carrier. Finally, the drone will return to the base to let the rescuers analyze the collected data and decide a rescue plan. Clearly, a fixed-wing drone can not hover, therefore the landing procedure must be executed while the carrier is loitering. Thus, in laboratory that real case could be reproduced by sending a circular reference trajectory to the target drone.

Bibliography

- [1] W. Johnson, *Rotorcraft Aeromechanics*. Cambridge University Press, 2013.
- [2] B. Hu, L. Lu, and S. Mishra, “Fast, safe and precise landing of a quadrotor on an oscillating platform,” in *American Control Conference, Chicago, USA*, 2015.
- [3] Dassault Systemes, “Dymola.” <https://www.3ds.com/it/prodotti-e-servizi/catia/prodotti/dymola>.
- [4] MathWorks, “Simulink.” <https://www.mathworks.com/products/simulink.html>.
- [5] G. Guglieri and C. E. Riboldi, *Introduction to Flight Dynamics*. Celid, 2014.
- [6] G. J. Leishman, *Principles of helicopter aerodynamics (second edition)*. Cambridge University Press, 2006.
- [7] D. A. Peters and N. HaQuang, “Dynamic inflow for practical applications,” *Journal of the American Helicopter Society*, vol. 33(4), pp. 64–68, 1988.
- [8] A. R. S. Bramwell, D. Balmford, and G. Done, *Bramwell’s Helicopter Dynamics*. Elsevier Butterworth-Heinemann, 2001.
- [9] The Modelica Association, “Modelica Home Page.” <https://www.modelica.org>.
- [10] M. Bangura, *Aerodynamics and Control of Quadrotors*. PhD thesis, The Australian National University, 2017.
- [11] F. Riccardi and M. Lovera, “Dynamic inflow and ground effect in multirotor UAV attitude dynamics,” in *43rd European Rotorcraft Forum, Milan, Italy*, 2017.

-
- [12] MathWorks, “Matlab.” <https://www.mathworks.com/products/matlab.html>.
- [13] D. Invernizzi, M. Lovera, and L. Zaccarian, “Geometric tracking control of underactuated VTOL UAVs,” in *American Control Conference, Milwaukee, USA*, 2018.
- [14] R. Pi Foundation, “Raspberry Pi 3 model b.” <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>, 2015.
- [15] A. Bonarini, M. Matteucci, M. Migliavacca, and D. Rizzi, “R2P: An open source hardware and software modular approach to robot prototyping,” *Robotics and Autonomous Systems*, vol. 62, no. 7, pp. 1073–1084, 2014.
- [16] P. Gattazzo, “Nonlinear control of a tilt-arm quadrotor UAV,” Master’s thesis, Politecnico di Milano, School of Industrial and Information Engineering, 2017.
- [17] M. Giurato, “Design, integration and control of a multirotor UAV platform,” Master’s thesis, Politecnico di Milano, School of Industrial and Information Engineering, 2015.
- [18] D. Lee, T. Ryan, and H. J. Kim, “Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing,” in *IEEE International Conference on Robotics and Automation, Saint Paul, USA*, 2012.
- [19] R. Furfaro, B. Gaudet, D. Wibben, J. Kidd, and J. Simo, “Development of non-linear guidance algorithms for asteroids close-proximity operations,” in *AIAA Guidance, Navigation and Control Conference, Boston, USA*, 2013.
- [20] M. Andreetto, D. Fontanelli, and L. Zaccarian, “Quasi time-optimal hybrid trajectory tracking of an n-dimensional saturated double integrator,” in *IEEE Conference on Control Applications, Buenos Aires, Argentina*, 2016.
- [21] M.-H. Hua and C. Samson, “Time sub-optimal nonlinear PI and PID controllers applied to longitudinal headway car control,” *International Journal of Control*, vol. 84, no. 10, pp. 1717–1728, 2011.

-
- [22] F. Forni, S. Galeani, and L. Zaccarian, “A family of global stabilizers for quasi-optimal control of planar linear saturated systems,” *IEEE Transactions on Control of Network Systems*, vol. 55, no. 5, pp. 1175–1180, 2010.
- [23] Motive, “Optitrack.” <http://optitrack.com/products/motive>.
- [24] Prime, “Motive Prime 13.” <http://optitrack.com/products/prime-13>.
- [25] Netgear, “Prosafe 28PT GE POE.” <https://www.netgear.com>.
- [26] L. Meier, “Px4 Autopilot.” <http://www.pixhawk.org>, 2008.
- [27] Intel, “Intel Edison Compute Module.” <https://software.intel.com/en-us/iot/hardware/edison>, 2014.
- [28] R. Pi Foundation, “Raspberry Pi Zero.” <https://www.raspberrypi.org/products/pi-zero/>, 2015.
- [29] O. S. R. F. OSRF, “ROS.” <http://www.ros.org>.
- [30] Microsoft, “Windows 10 Pro.” https://www.microsoftstore.com/store/mseea/it_IT/pdp/Windows-10-Pro/productID.320433200.
- [31] Vmware, “VMware.” <https://www.vmware.com>.
- [32] Canonical, “Ubuntu 16.04 lts.” <https://www.ubuntu.com>.
- [33] K. Gravel and A. Bencz, “Mocap Optitrack.” https://github.com/ros-drivers/mocap_optitrack.
- [34] R. Pi Foundation, “Raspbian Jessie 4.4.” <https://www.raspberrypi.org/downloads/raspbian/>.
- [35] Emutex, “Jubinux.” <http://www.jubinux.org>.
- [36] L. Meier, “Mavros.” <https://github.com/mavlink/mavros>.
- [37] L. Meier, “Px4 Firmware.” <https://github.com/PX4/Firmware>, 2008.
- [38] L. Project, “LaTeX.” <https://www.latex-project.org>.
- [39] GitHub.com, “GitHub.” <https://github.com>.
- [40] GitLab.com, “GitLab.” <https://about.gitlab.com>.

