Modelling and Control of Autonomous Agile Helicopter Flight using Quaternion Methods

A Complete Approach to Quaternion Use in Control and Modelling: a Bo105 Case Study

Barbaglia Riccardo

TU Delft

Modelling and Control of Autonomous Agile Helicopter Flight using Quaternion Methods

A Complete Approach to Quaternion Use in Control and Modelling: a Bo105 Case Study

Thesis report

by

Barbaglia Riccardo

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on June 25, 2025 at 09:30

Thesis committee:

Chair:	Dr. C.C. de Visser
Supervisors:	Dr. M.D. Pavel
	Dr. E. van Kampen
External examiner:	Dr. E. Mooij
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	August, 2024 - May, 2025
Student number:	5827868

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Faculty of Aerospace Engineering · Delft University of Technology

Contents

Nomenclature

List of Abbreviations

AAM	Advanced Air Mobility
ADS	Aeronautical Design Standard
AI	Artificial Intelligence
ANDI	Adaptive Nonlinear Dynamic Inversion
BP	Behaviour Planning
DCM	Direction Cosine Matrix
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V.
DOF	Degree of freedom
FSM	Finite State Machine
HFSM	Hierarchical Finite State Machine
INDI	Incremental Nonlinear Dynamic Inversion
LQG	Linear Quadratic Gaussian
LQI	Linear Quadratic Integral
LQR	Linear Quadratic Regulator
LTR	Loop Transfer Recovery
MAV	Micro Aerial Vehicle
MBB	Messerschmitt-Bölkow-Blohm
MIMO	Mulitple Input - Mulitple Output
ML	Machine Learning
MPC	Model Predictive Control
MR	Main Rotor
MSE	Mean Squared Error
MTE	Mission Task Element
MTOW	Maximum Take-Off weight
NDI	Nonlinear Dynamic Inversion
NED	North-East-Down
PID	Proportional-Integral-Derivative
PSO	Particle Swarm Optimization
SLERP	Spherical linear interpolation
TR	Tail Rotor
UAM	Urban Air Mobility
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UTTAS	Utility Tactical Transport Aircraft System
VTOL	Vertical Take-off and Landing

List of Symbols

$\bar{\mathbf{q}}$	Quaternion vector component
$\bar{\phi}$	Trim roll angle
$\bar{\psi}$	Trim yaw angle
$\bar{\theta}$	Trim pitch angle
β	Angle of sideslip
δ_0	Pilot collective command
δ_p	Pilot pedal command
δ_x	Pilot longitudinal cyclic command
δ_y	Pilot lateral cyclic command
ϵ_i	Phase of eigenvalue i
λ	Eigenvalue
$\mathbf{M}_C$	Controllability matrix
$\mathbf{M}_O$	Observability matrix
$\mathbf{n}$	Euler rotation axis
$\mathbf{q}$	Unit quaternion
$\mathbf{q}_r$	Quaternion attitude reference
$\mathbf{q}_{A2B}$	Unit quaternion describing rotation to align reference frame $\mathcal{A}$ to reference frame $\mathcal{B}$
$\mathbf{q}_{E2B}$	Quaternion attitude of the helicopter

$\mathbf{q}^{\star}_{E2B}$	Desired quaternion attitude
$\mathbf{u}$	Control input vector
$\mathbf{v}$	Eigenvector
$\mathbf{V}^{\star}$	Desired velocity measured in the $\mathcal{G}$ reference frame
$\mathbf{V}_c$	Velocity correction signal measured in the $\mathcal{G}$ reference frame
$\mathbf{X}$	Position vector
$\mathbf{x}$	Dynamic state vector
$\mathbf{X}^{\star}$	Desired position vector
$\mathbf{y}$	Dynamic output vector
ω	Angular velocity
ϕ	Roll angle
ψ	Yaw angle
Ψ_0	Cyclic phasing angle
ρ_i	Magnitude of eigenvalue i
τ	First order time constant
θ	Euler rotation angle
θ	Pitch angle
θ_0	Collective pitch
θ_c	Lateral cyclic pitch
θ_p	Tail rotor pitch
θ_s	Longitudinal cyclic pitch
ε	Correction angle for sideslip correction in attitude reference
n_u	Number of inputs
n_x	Number of states
p	Roll rate
q	Pitch rate
q_0	First (scalar) quaternion component
q_i	Second quaternion component
q_j	Third quaternion component
q_k	Fourth quaternion component
r	Yaw rate
SS_{Eul}	State-space system with Euler angles attitude parametrization
SS_{quat}	State-space system converted to the quaternion formulation
u	Surge velocity component
v	Sway velocity component
w	Heave velocity component
X	Global x-axis coordinate
Y	Global y-axis coordinate
Z	Global z-axis coordinate
f	Nonlinear dynamic state-space equations
h	Nonlinear quaternion rate equations
α	Correction angle for offline attitude reference determination
ΔT_0	Time duration of the manoeuvre forward-flight entrance segment
$\Delta\mathbf{q}$	Quaternion attitude tracking error
$\Delta\mathbf{V}$	Velocity perturbation for position reference generation
$\Delta\Phi$	Corrective roll control signal
$\Delta\Theta$	Corrective pitch control signal
$\hat{\mathbf{q}}^{\star}_{E2B}$	Required reference frame rotation to align the desired body z-axis with desired acceleration vector
$\hat{\theta}_0$	Actual collective deflection
$\hat{\theta}_c$	Actual lateral cyclic deflection
$\hat{\theta}_p$	Actual pedal deflection
$\hat{\theta}_s$	Actual longitudinal cyclic deflection
$\mathbf{A}$	State transformation matrix
$\mathbf{A}^{\star}$	Acceleration required to track a manoeuvre without accounting for gravitational acceleration g
$\mathbf{A}_T$	Total required acceleration vector to perform synthesised manoeuvres
$\mathbf{a}_T$	Normalized total required acceleration vector to perform synthesised manoeuvres

$\mathbf{B}$	Input-to-State matrix
$\mathbf{C}$	Output matrix
$\mathbf{C}_r$	Output matrix isolating $\mathbf{y}_r$ in the LQI controller
$\mathbf{D}$	Feed-through matrix
$\mathbf{g}^\star$	Overall best position across all particles of the PSO
$\mathbf{J}$	Fitness function for the PSO algorithm
$\mathbf{K}$	LQI controller gain matrix
$\mathbf{K}_i$	Integrator LQI gain matrix
$\mathbf{K}_x$	Regulator LQI gain matrix
$\mathbf{P}$	Riccati matrix
$\mathbf{p}$	Particle of the PSO
$\mathbf{p}_i^\star$	Best position of particle i of the PSO
$\mathbf{Q}$	LQI state cost weighted matrix
$\mathbf{q}_\Delta$	Corrective quaternion for attitude reference generation and attitude error calculation
$\mathbf{R}$	LQI Input cost weighted matrix
$\mathbf{R}_{E2B}^{[321]}$	Euler rotation matrix to align reference frame $\mathcal{A}$ to reference frame $\mathcal{B}$ using the 3-2-1 rotation sequence
$\mathbf{r}_1$	Random scaling vector for the PSO cognitive term
$\mathbf{r}_2$	Random scaling vector for the PSO social term
$\mathbf{r}_{x,y}$	Vector connecting point x to point y
$\mathbf{v}_i$	Velocity of particle i in the PSO
$\mathbf{V}_{trim}$	Trim velocity measured in the $\mathcal{G}$ reference frame
$\mathbf{x}_i$	Added integrator states in the LQI controller
$\mathbf{y}_r$	Tracking outputs in the LQI controller
$\mathbf{z}_E$	z-axis expressed in vector form
$\mathcal{A}$	Generic example reference frame
$\mathcal{B}$	Body reference frame
$\mathcal{E}$	North-East-Down reference frame
$\mathcal{G}$	Global reference frame
θ_0	Commanded collective deflection
θ_c	Commanded lateral cyclic deflection
θ_p	Commanded pedal deflection
θ_s	Commanded longitudinal cyclic deflection
ε	Correction angle for sideslip correction in attitude reference
ζ	Corrective generalized angle for quaternion reference generation
$\{a, \ldots, e\}$	Scalar coefficients for polynomial fitting and interpolation
A	Amplitude of sinusoidal oscillation in slalom
B	Frequency of sinusoidal oscillations in slalom
c_1	Corrective coefficient for quaternion reference generation
c_2	Corrective coefficient for quaternion reference generation
c_θ	Cosine of the θ angle
g	Gravitational acceleration
h_1	Altitude gain in the pop-up entrance manoeuvre segment
h_k	Arc length difference between waypoint $k+1$ and waypoint k
k	Waypoint index
K_P	Proportional gain
L	Distance covered in the manoeuvre forward-flight entrance segment
l	Running variable for interpolation and for the calculation of the helicopter's position on the reference trajectory
N	Number of particles in the PSO algorithm
N	Number of waypoints in trajectory reconstruction
s	Arc length
s_1	Horizontal distance covered in the pop-up entrance manoeuvre segment

s_θ	Sine of the θ angle
V_x	Helicopter's velocity in the X direction
V_y	Helicopter's velocity in the Y direction
V_z	Helicopter's velocity in the Z direction
w	Inertia scaling weight for the PSO
y	Generic polynomial interpolation for trajectory reconstruction
$\mathbf{q}_\varepsilon^\star$	Sideslip correction quaternion for desired attitude reference determination

List of Figures

List of Tables

1

Introduction

1.1. Research Formulation

Motivated by advancements in autonomous navigation technologies and with a desire to safely expand the navigational capabilities of existing aircraft designs, the field of aerospace engineering has experienced a consistent move towards the investigation of highly agile aircraft configurations that enable exceptional maneuverability. At the same time, with novel applications such as Advanced Air Mobility and a rising interest in the use of air vehicles to support other tasks, Vertical Take-Off and Landing (VTOL) capabilities have become highly valued as they allow for greater control at lower speeds and hovering potential enables a large set of possible applications for vehicles, such as search and rescue, reconnaissance, and precision transport of loads. As operational tasks become more complex, ensuring safety, efficiency and precision in navigation under a number of highly variable circumstances and flight conditions therefore becomes an increasingly more crucial requirement.

As increasing focus is put on achieving complex manoeuvres and difficult flight geometries, limitations of traditional modelling and control system development approaches have become more apparent. Notably, attitude representation with Euler angles suffers from issues like gimbal lock—a condition that creates singularities during complex maneuvers—and imposes considerable computational complexity when modelling the nuanced geometries of agile flight: while Euler angles remain highly useful for their intuitive interpretation, these deficiencies pose limitations to the implementation of controllers capable of complex manoeuvres and naturally increase the computational cost of online trajectory reconstruction operations. Addressing these reasons, engineers have begun to explore quaternions as an alternative solution for aircraft control.

Naturally avoiding the limitations of Euler angles, quaternions inherently eliminate the risk of singularities while offering more streamlined computations. Quaternions are a four-dimensional extension of complex numbers used to represent rotations in three-dimensional space, consisting of one scalar and three vector components. Quaternion algebra introduces a unique formulation of multiplications and rotations, which allow to represent attitudes and rotations without encountering singularities and only using linear equations. Because of these characteristics, quaternions facilitate more reliable and efficient tracking of dynamic, complex maneuvers: for agile helicopters, this shift enables a more complete and accurate expression of high mobility, which is crucial for developing a robust autonomous full flight control system. Despite their many advantages, the use of quaternions in modelling and controller development has noticeably remained limited to highly specific fields such as satellite attitude control and control of highly agile Micro Air Vehicles (MAV): the exploration of the implementation of quaternions to more conventional air vehicles remains a topic of active interest and a prominent trend for future developments in the field of aircraft control.

Contributing to the current academic discussion and body of research around the implementation of quaternions to the control and modelling of air vehicles, this report proposes a method to convert existing Euler-based models to quaternion-based models, and fully details the development of a quaternion-based flight control system for the autonomous execution of offline-calculated agile manoeuvres: both these contributions are applied on the MBB Bo105 agile helicopter, which is used as a test case. In the interest of creating a control system capable of leveraging the high manoeuvrability characteristics of the helicopter, quaternions provide significant benefits as they allow for an efficient and flexible method of attitude

parametrization, free of the limitations imposed by other more common approaches commonly used; these reasons make them highly desireable in modelling and control applications, especially when applied to agile aircraft and complex tracking tasks.

Further - with the rise of machine learning (ML) and artificial intelligence (AI) - many of the current research trends in the field of control engineering and especially regarding the autonomous execution of manoeuvres have become increasingly more data-driven, requiring large amounts of data and resources, the collection and processing of which is a highly expensive and time-consuming process. In contrast, this report will rely primarily on analytical methods for trajectory generation and model study, with the entire process clearly detailed throughout the text, to define an accessible and effective methodology to develop control systems for autonomous control. In particular, trajectory generation will be performed using polynomial trajectories and simple dynamic relations to extract position, velocity, and attitude references, while autonomous flight will be enabled using a Finite State Machine (FSM) autopilot. The execution of manoeuvers will be facilitated by the introduction of a full flight control system, which integrates Linear Quadratic Integral (LQI) control with quaternions to control the helicopter's attitude and Proportional-Integral (PI) control to command velocity and position. The tuning of the controller, to limit the need for hand-tuning, is performed using the Particle Swarm Optimization (PSO) method, with its implementation discussed in the thesis.

1.2. Research Questions and Research Objectives

To guide the development of the project, research questions were defined to drive the research process and research objectives were identified to identify a path for the progression of the thesis work. Here the research questions and objectives identified will be delineated and motivated, to provide a global understanding of the project's scope, the challenges being addressed, and the methodological framework adopted.

As one of the primary objectives of this report is the contribution to existing literature regarding quaternion-based control systems, it is first essential to understand and highlight what are the advantages of the quaternion formulation when compared to other - more commonly used - methods of attitude parametrization. As such, the first research question defined is:

Research Question 1

How are quaternions beneficial to the modelling of helicopter dynamics and to the development of control systems?

To answer this first question, quaternions must be introduced from a mathematical standpoint and compared to other commonly utilized means of attitude representation: Euler angles and Direction Cosine Matrices (DCM). To emphasise the use of quaternions as means of attitude parametrization, specific examples will be introduced, focusing on the practical implementation of quaternions. All considerations delineated in this analysis will be generally applicable to all air vehicles, but given the focus of this thesis, emphasis will be given to helicopters in particular.

With quaternions introduced and discussed, and their usefulness demonstrated, the baseline existing linearized model of the BO105 helicopter used for the purpose of this thesis must be presented and studied. In an effort to fully incorporate quaternions in the model and the control system, the existing linearized model of the helicopter - an identified model using Euler angles - needs to be manipulated to have an internal attitude representation using quaternions. From this, the following question arises:

Research Question 2

How can an existing state-space flight dynamics model be augmented to include quaternions for its attitude representation?

To this end, a procedure is discussed to modify the model to include the relevant quaternion representation. By addressing this research question, a methodology for the rapid conversion of already existing models to a quaternion formulation can be defined, which in turn would allow to leverage the robustness and computational advantages of quaternions and facilitate the use of advanced control algorithms without the need for a complete re-identification of the system. The validity of this procedure will be discussed by comparing the original model and the quaternion model, identifying if the modes and characteristics of the system are properly maintained throughout the procedure.

With the model defined and appropriately modified, the control strategy best suited to the control of this model needs to be identified. As such, the following research question arises:

Research Question 3

How can a quaternion-based autonomous full flight control system be optimally developed for an agile helicopter?

To answer this question, a detailed analysis of the model needs to be performed from a control theory standpoint: analysing the modes of the aircraft, the stability characteristics of the BO105 helicopter model, as well as the states and inputs of the model itself. With the characteristics of the system defined and a control strategy analysed. Further sub-questions emerging from RQ 3 are:

- What are the controllability/observability characteristics of the linearized model?
- What are possible control methods and what is the best suited for this specific system?
- How can controller parameter tuning be systematically executed to achieve desired agile maneuvering?

With these questions answered, the tuned system is to be developed and the performance of the final tuned system verified.

To verify the performance of the controller, a set of manoeuvres will be simulated via desktop simulations to test the capabilities of the controlled helicopter to execute specific manoeuvres and tracking tasks. To answer this, another fundamental research question arises:

Research Question 4

How can maneuver reference trajectories be systematically designed to evaluate the tracking performance of a helicopter control system?

Verification is fundamental in the process of development as it ensures the correct implementation of the system for the purpose of testing and simulation. For the purpose of this thesis, verification will be evaluated by simulating relevant tracking tasks that emphasise the manoeuverability of the air vehicle.

Lastly, appropriate systems need to be established to enable the helicopter to precisely navigate through the designed manoeuvers. As navigation based on continuous reference signals generally causes poor tracking and is unreliable in the presence of disturbances, to improve online tracking performance the controlled system is to be augmented with a simple autonomous navigation system. From this, the following research question is posed:

Research Question 5

How can an autonomous flight system be designed to optimize reference tracking for agile maneuvers in helicopters?

In the context of this thesis, a baseline solution using a Finite State Machine (FSM) architecture was explored to produce a functioning simple autopilot for reference tracking that could later on be expanded

further, allowing for the introduction of more sophisticated autonomous behaviours as needed.

With the research questions that will drive forward the development of the thesis defined, the following research objectives have been delineated to allow for a smooth progression of the work:

Research Objective

- Convert the linearized model into a proper quaternion formulation.
- Analyse the models and verify the quaternion conversion.
- Develop a quaternion-based control system to perform manoeuvre tracking.
- Identify a structured tuning strategy to identify the best controller parameters.
- Develop a methodology to implement manoeuvres to test the helicopter's capabilities.
- Develop a simple autopilot architecture to navigate offline-planned trajectories.
- Verify implementation and performance of the controller by simulating the manoeuvres.

1.3. Structure of the Report

With the research questions and research objectives defined in Section 1.2, the structure of the report will now be delineated to provide an understanding of the modulation of the discussion around the report. The report is divided in a total of seven parts, each dealing with a specific area of the research process and expanding upon the methodology and the intermediate and final results obtained. Here the various parts of the thesis will be briefly presented.

To start the report, Part I and Part II provide a comprehensive understanding of the thesis and of the motivations that brought about it. Part I contains the research paper resulting from the thesis study, which details a complete collection of the results as well as a complete and concise explanation of the process followed, complete with a brief analysis and discussion of the results and of the methodologies. Complementing this, Part II goes over the preliminary analysis of the thesis, with Chapter 2 containing a literature review of current research (further motivating the interest in the research topic and contextualizing this work in the broader academic discussion), and Chapter 3 detailing the methodology identified to provide comprehensive answers to the research questions discussed above. To conclude the comprehensive presentation of the results, Chapter 4 presents a punctual overview of the contributions of this thesis.

Expanding upon the overview of the work provided in the first two parts, Part III to Part VI contain the bulk of the work performed during the thesis. These parts describe the considerations and decisions made during the entire research and study, providing a more in-depth analysis of the research process and discussing the intermediate results and their relevance.

Part III and Part IV will deal with the modelling aspect of the thesis, introducing quaternions and delving into a discussion of the available linearized models and their conversion to a quaternion formulation.

To introduce the discussion on the modelling section of the thesis, Part III provides an introduction to quaternion mathematics, presenting the arithmetic formulation of various fundamental operations specific to quaternions. To clarify the applicability of quaternions in the context of control system engineering and aircraft control and provide practical and useful information, this initial part will focus on providing clear explanations of how quaternion operations apply to attitude representation and clear comparisons will be provided between quaternions and other more commonly used forms of attitude parametrization to further justify the choice of this specific attitude representation.

With quaternions fully defined and their relevance explained, Part IV will delve in a description of the helicopter model. To achieve this, the BO105 will first of all be discussed from an engineering standpoint, discussing the mechanical characteristics of the helicopter: this will justify the choice of this model for the purpose of agile manoeuvering and will introduce the helicopter itself and its characteristics. Afterwards, the available linearized state-space models will be discussed from a control theory standpoint: discussing the system's inputs, outputs, and states; detailing the modes and stability characteristics of the helicopter;

and analysing other relevant characteristics, such as the controllability and observability of the model. Concluding this part, in Chapter 7 the internal model attitude representation will be changed from the original Euler angles parametrization to the quaternion parametrization.

With the modelling aspects of the thesis discussed, Part V and Part VI will delve in the development of the control system and the execution of autonomous flight manoeuvres.

Having presented the model and discussed its characteristics, Part V will discuss the development of the controller, which for the purpose of this thesis is selected to be a nested-loop architecture mixing Linear Quadratic Integral (LQI) control and PID control. To start, the overall controller architecture is discussed in Chapter 8: here the chosen control techniques will be motivated, stating their advantages and limitations, and their implementation in the overall control system design will be discussed. Afterwards, in Chapter 9 a reliable and repeatable approach to controller tuning is provided and its algorithm representation implemented for the tuning of each of the controller's loops.

Finally, Part VI will present the discussion on the definition of manoeuvres and the implementation of an autopilot system. This part will be further divided in three chapter. Chapter 10 will first discuss the offline definition of manoeuvres. Here a systematic approach to the definition of all parameters required to fully define the manoeuvres will be identified, and this procedure will be applied to the identification of two representative manoeuvres: the slalom and the pull-up/push-down. With the manoeuvres defined, Chapter 11 will discuss the implementation of an autopilot system. To start, the reference data identified will be divided in waypoints equally spaced in time and a waypoint interpolation algorithm to reconstruct the trajectory between waypoints. Afterwards, a simple autopilot will be implemented with an FSM architecture: this enables the identification of a simple structure that can easily be augmented to enable more complex behaviours. Finally, the methodologies developed and the implementation of both the autopilot and the control system will be verified by simulating the identified manoeuvres.

Concluding the discussion of the thesis, in Part VII, Chapter 13 will serve as the concluding section of the work. First, in Section 13.1 the closing remarks of the thesis will be provided, overviewing the obtained results and providing explicit answers to the research questions posed previously, while also reviewing the research process followed. Complementing this, in Section 13.2 recommendations for future developments and continuations of this work will be provided to further expand upon the continuation of research on quaternions applications in the modelling and autonomous flight of full-scale vehicles.

Part I

Scientific Article

Modelling and Control of Autonomous Agile Helicopter Flight using Quaternion Methods

Riccardo Barbaglia*
Control and Simulation Section, Faculty of Aerospace Engineering
Delft University of Technology, Delft, The Netherlands

Demand for fully autonomous VTOL aircraft with high agility has exposed the shortcomings of Euler-angle models: limiting factors such as the presence of singularities and the need for expensive trigonometric operations impact the reliability and safety of autonomous systems and impair the execution of highly-aggressive manoeuvres. Quaternions offer a singularity-free alternative, yet their use on full-scale helicopters remains underexplored. Addressing this gap in research, this paper introduces a methodology for converting existing Euler-based linear models of a Bo105 helicopter to quaternion form, preserving model fidelity. A hierarchical flight control system is then designed, controlling attitude with a novel implementation of Linear Quadratic Integral (LQI) control with quaternions, and commanding velocity and position using PI and P controllers. The controller is tuned using a structured approach implementing the Particle Swarm Optimization (PSO) algorithm. The system is then augmented with a Finite State Machine (FSM) autopilot to follow offline-generated trajectories. The implemented system was evaluated by simulating a slalom and a pop-up manoeuvre: the simulations demonstrated maximum tracking errors of 5 m for slalom and 7 m for pop-up, primarily due to velocity-loop delays, while achieving zero steady-state error in the position. These results confirm the effectiveness of quaternion-based modelling and control for agile, autonomous VTOL operations.

I. Introduction

The evolution of autonomous navigation technologies has reignited interest in vertical take-off and landing (VTOL) aircraft, whose hovering capability and low-speed precision promise transformative applications in search and rescue, reconnaissance, and precision logistics [1, 2]. As aerospace engineering pushes the boundaries of high-agility configurations, designers face increasingly complex mission profiles that demand safety, efficiency, and accuracy under diverse conditions [3]. In this search for ever-improving agility and manoeuvrability, traditional control frameworks remain constrained by the Euler-angle parametrization of attitude, which introduces singularities (gimbal lock) during large rotations and burdens the system with computationally expensive trigonometric transformations.

In an effort to overcome the limitations of traditional modelling and control approaches, research has recently advocated for the use of attitude parametrization with quaternions [4], a mathematical construct that allows for a continuous, numerically efficient, and singularity-free representation of attitude [5], with early studies suggesting that quaternions could enable significant improvements in modelling accuracy and control performance [6, 7]. These characteristics also find application in autonomous navigation, allowing for an efficient and streamlined planning of tasks to be executed by the vehicle [8].

Motivated by these advantageous characteristics, studies have been performed on the application of quaternions to the modelling of autonomous small-scale vehicles [9–11], such as drones and quadrotors, but crucially little research is present regarding the implementation of quaternions to full-scale aircraft and autonomous control systems, two areas of relevance in the academic and industrial fields. In response to this gap in scientific literature, and to meaningfully contribute to the broader academic discourse in the aerospace engineering field, this work explores the implementation of quaternions in the modelling and autonomous control of a full-scale Bo105 helicopter, an aircraft with remarkable performance and high manoeuvrability, to enable it to perform autonomous agile navigation of offline-defined trajectories.

This work contributes to existing research in quaternion modelling and control and autonomous flight in four primary ways.

- In Section III.B, a methodology to convert existing Euler angles linearized models to have their attitude represented using quaternion elements is presented and applied to the Bo105 linearizations. This procedure is instrumental in facilitating the study of quaternions and of their implementation in control systems, as it allows the use of already existing Euler angles models without requiring re-linearizations and system identification procedures.
- In Section IV.A, an approach for the implementation of Linear Quadratic Integral (LQI) control to a quaternion-based linearized model is presented. This result serves as a guideline to the application of powerful optimization-based LQ techniques to quaternion systems, facilitating the design of complex and robust controllers.
- In Section IV.B, tuning of the control system is performed using the Particle Swarm Optimization (PSO) algorithm. This multimodal optimization is used to tune the LQI and PID controllers. The tuning is discussed for each of the control loops, clarifying the cost function used and providing an intuitive understanding of the role of each of its terms.
- In section V, the implementation of an autopilot system based on the Finite State Machine (FSM) architecture is discussed, a flexible automation framework that enables autonomous navigation. This modular design supports autonomous navigation and can be extended for advanced behaviors, such as online trajectory planning for obstacle avoidance.

*MSc Student, Department of Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology.

II. An Introduction to Quaternions

Quaternions are a mathematical construct useful in representing rotations in 3D space, which offer significant advantages over Euler angles and other attitude parametrizations. Here, quaternions will be introduced mathematically, presenting their fundamental mathematical operations and stating their advantages over other commonly used attitude parametrizations, the Euler angles and the Direction Cosine Matrix (DCM). Here only the quaternion operations essential to the modelling of systems and the integration of controllers will be presented; for a more exhaustive overview of quaternions and a comparison with other attitude formulations, consult [12].

Here the mathematical properties of quaternions are introduced for the purpose of attitude representation; for a more comprehensive understanding of their properties, a general explanation is provided in [13]. To accurately represent attitudes, quaternions are required to have unitary norm: as such, in this paper quaternions will always be assumed to be normalized [5]. The way quaternions encode attitude is well explained by considering Euler's rotation theorem [14], which states that a rigid body can be driven to any attitude by rotating around an adequate rotation axis (called *Euler axis*) by an adequate angle (called *Euler angle*). Considering two reference systems $\mathcal{A}$ and $\mathcal{B}$ and letting $\mathbf{n}_{A2B}$ and θ_{A2B} represent the Euler axis and the Euler angle required to drive $\mathcal{A}$ to coincide with $\mathcal{B}$, the quaternion representing the attitude of frame $\mathcal{B}$ with respect to frame $\mathcal{A}$, the quaternion expressing this rotation is:

$$\mathbf{q}_{A2B} = \begin{bmatrix} q_0 \\ q_i \\ q_j \\ q_k \end{bmatrix} = \begin{bmatrix} q_0 \\ \bar{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{A2B}/2) \\ \sin(\theta_{A2B}/2)\{\mathbf{n}_{A2B}\}_A \end{bmatrix} \tag{1}$$

This four-dimensional representation of rotations allows quaternions to overcome gimbal lock, the primary limitation of Euler angles which causes discontinuities when expressing rotations in 3D space, but also introduces an ambiguity relevant in control system design. From Equation 1 it follows that $\mathbf{q}_{A2B}$ and $-\mathbf{q}_{A2B}$ are *equivalent quaternions* as they represent the same relative attitude, with the only difference being that the former represents a rotation of θ_{A2B} around $\mathbf{n}_{A2B}$ while the latter represents a rotation of $2\pi - \theta_{A2B}$ around $-\mathbf{n}_{A2B}$. It follows naturally that one rotation will span a longer angle than the other: the quaternion describing the shortest rotation will be the one where $q_0 \geq 0$. This process is usually referred to as "disambiguation". To simplify notations, in this paper the quaternion encoding the shortest rotation is used.

Quaternions also enable an intuitive and computationally effective method to concatenate successive rotations. Let quaternions $\mathbf{q}_{A2B}$ and $\mathbf{p}_{B2C}$ respectively denote the relative attitude of frame $\mathcal{B}$ with respect to frame $\mathcal{A}$ and of frame C with respect to frame $\mathcal{B}$. The quaternion $\mathbf{q}_{A2C}$ denoting the relative attitude of frame C with respect to frame $\mathcal{A}$ can be immediately identified via the quaternion multiplication operation [15]:

$$\mathbf{q}_{A2C} = \mathbf{q}_{A2B}\mathbf{p}_{B2C} = \begin{bmatrix} q_0 \\ \bar{\mathbf{q}} \end{bmatrix} \begin{bmatrix} p_0 \\ \bar{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} q_0 p_0 - \bar{\mathbf{q}} \cdot \bar{\mathbf{p}} \\ q_0\bar{\mathbf{p}} + p_0\bar{\mathbf{q}} + \bar{\mathbf{q}} \times \bar{\mathbf{p}} \end{bmatrix} \tag{2}$$

Because of the cross-product operation, quaternion multiplication is not commutative except when the two quaternions being multiplied are one the inverse of the other. Considering quaternion $\mathbf{q}_{A2B}$, the inverse of $\mathbf{q}_{A2B}$, which defines the relative attitude of frame $\mathcal{A}$ with respect to $\mathcal{B}$, is defined as:

$$\mathbf{q}_{A2B}^{-1} = \begin{bmatrix} q_0 \\ -\bar{\mathbf{q}} \end{bmatrix} = \mathbf{q}_{B2A} \tag{3}$$

and satisfies the equation $\mathbf{q}_{A2B}\mathbf{q}_{A2B}{-}1 = \mathbf{q}_{A2B}{-}1\mathbf{q}_{A2B} = [1 \quad 0 \quad 0 \quad 0]^T$. Note that Equation 3 is only valid for unit quaternions.
As another point of note, rotation concatenation via quaternion elements is an entirely linear operation that does not require trigonometric or transcendental operations: because of this, rotation concatenation via quaternion multiplication can be immediately written in the matrix multiplication form shown in Equation 4.

$$\mathbf{pq} = \begin{bmatrix} p_0 & -p_i & -p_j & -p_k \\ p_i & p_0 & -p_k & p_j \\ p_j & p_k & p_0 & -p_i \\ P_k & -p_j & p_i & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_i \\ q_j \\ q_k \end{bmatrix} \tag{4}$$

Consider now reference frame $\mathcal{B}$ rotating around frame $\mathcal{A}$ with angular rate vector ω: Equation 5 shows the time propagation equation that allows expressing the quaternion rate vector $\dot{\mathbf{q}}_{A2B}$ as a function of $\mathbf{q}_{A2B}$ and of the rate vector ω [15]. By means of Equation 4, Equation 5 can also be rewritten in matrix multiplication form.

$$\dot{\mathbf{q}}_{A2B} = \frac{1}{2}\mathbf{q}_{A2B} \begin{bmatrix} 0 \\ \omega \end{bmatrix} \tag{5}$$

Another property of quaternions used extensively within this report is their ability to compute coordinate transformations using only linear equations. Discussing coordinate transformations, it is important to distinguish between *active* and *passive* rotations [5, 14].
Active rotations allow for the rotation of a vector or point expressed in a fixed coordinate system. As an example, consider vector $\mathbf{v}$ expressed in reference frame $\mathcal{A}$, and let $\mathbf{r}$ be the result of rotating $\mathbf{v}$ around the Euler axis $\mathbf{n}_{v2r}$ by the Euler angle θ_{v2r}. The coordinates of $\mathbf{r}$ in reference frame $\mathcal{A}$ may then be calculated as shown in Equation 6, where $\mathbf{q}_{v2r} = [\cos(\theta_{v2r}/2),\ \sin(\theta_{v2r}/2)\{\mathbf{n}_{v2r}\}_A]$.

$$\begin{bmatrix} 0 \\ \{\mathbf{r}\}_A \end{bmatrix} = \mathbf{q}_{v2r} \begin{bmatrix} 0 \\ \{\mathbf{v}\}_A \end{bmatrix} \mathbf{q}_{v2r}^{-1} \tag{6}$$

Passive rotations allow to perform rotations of the reference frame in which a vector is expressed. Passive rotations are highly useful when identifying the coordinates of a vector in a reference frame different from the one it is originally expressed. Consider, for example, vector $\mathbf{v}$ expressed in reference frame $\mathcal{A}$, and let $\mathcal{B}$ be the reference frame obtained when rotating $\mathcal{A}$ according to quaternion $\mathbf{q}_{A2B}$. The coordinates of vector $\mathbf{v}$ expressed in reference frame $\mathcal{B}$ can therefore be expressed as:

$$\begin{bmatrix} 0 \\ \{\mathbf{v}\}_B \end{bmatrix} = \mathbf{q}_{A2B}^{-1} \begin{bmatrix} 0 \\ \{\mathbf{v}\}_A \end{bmatrix} \mathbf{q}_{A2B} \tag{7}$$

Quaternions can also interpolate with the Spherical Linear Interpolation (SLERP) algorithm [5, 16]. Consider the quaternions $\mathbf{q}_{E2B}$ and $\mathbf{q}_{E2C}$, encoding the relative attitudes of reference frames $\mathcal{B}$ and C with respect to frame $\mathcal{E}$. The shortest rotation that transforms $\mathcal{B}$ to coincide with C is found using the SLERP algorithm in Equation 8, where $\theta = \arccos(\mathbf{q}_{E2B} \cdot \mathbf{q}_{E2C})$ and $l \in [0, 1]$ is an interpolating parameter. This efficient interpolation method allows for a singularity-free transition between two generic orientations represented by quaternions.

$$\hat{\mathbf{q}}(l) = \frac{\sin((1-l)\theta)}{\sin(\theta)} \mathbf{q}_{E2B} + \frac{\sin(l\theta)}{\sin(\theta)} \mathbf{q}_{E2D} \tag{8}$$

From these considerations, quaternions have significant advantages over other commonly used attitude parametrizations. Being fully defined by only four parameters, quaternions offer a concise attitude representation methodology, making them comparable in terms of numerical efficiency to Euler angles (which require three parameters) and far more efficient than the Direction Cosine Matrix (DCM) representation (which requires nine parameters), while not suffering from singularities, gimbal lock, and only using linear operations. Moreover, quaternions only make use of linear equations and offer a highly effective attitude interpolation algorithm using SLERP, while DCM and Euler angles require complex interpolation algorithms with inefficient operations [17] and singularities [12]. Equations 9 to 11 show the relation between quaternions and Euler angles, which will be useful in the analysis of the system presented in section III, where the available linearized models will be modified to have the internal attitude parametrization make use of the quaternions. A complete derivation of these equations is found in [18], which is only cited here for brevity.

$$\phi = \text{atan2}\left(2\left(q_0 q_i + q_j q_k\right), 1 - 2\left(q_i^2 + q_j^2\right)\right) \tag{9}$$

$$\theta = -\pi/2 + 2\,\text{atan2}\left(\sqrt{1 + 2\left(q_0 q_j - q_i q_k\right)}, \sqrt{1 - 2\left(q_0 q_j - q_i q_k\right)}\right) \tag{10}$$

$$\psi = \text{atan2}\left(2\left(q_0 q_k + q_i q_j\right), 1 - 2\left(q_j^2 + q_k^2\right)\right) \tag{11}$$

III. Conversion of the Bo105 Model to Quaternion Parametrization

For this study, two 6-DOF identified linearized models of the MBB Bo105 helicopter were provided. The Bo105 is an agile, twin-engine hingeless helicopter developed by Messerschmitt-Bölkow-Blohm in 1967 [19]. The Bo105 fits in the 2.5-ton class can carry up to 3 passengers and has found a number of applications in transport, search and rescue, offshore operations and even aerobatic flight [20, 21]: given its high agility, representing this helicopter's dynamics with quaternions would improve modelling fidelity and enable complex and unconventional manoeuvres more efficiently. The model identification and linearization procedure were performed by engineers at the German aerospace centre Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) [22, 23] and the available linearization points were in hover and in forward flight with a trim speed of 65kn.

A. Characteristics of the Linearized Models

The available data is composed of two 6-DOF system-identified linearized models, each provided in the form of only a state matrix $\mathbf{A}$ and an input matrix $\mathbf{B}$. Both models contain a state vector comprised of 11 elements: body velocities $[u, v, w]$ measured in m/s, Euler angles attitude states $[\phi, \theta, \psi]$ measured in rad, body rotational rates $[p, q, r]$ measured in rad/s, and body rotational acceleration $[\dot{p}, \dot{q}]$ measured in rad/s^2. The available inputs are the commands available to the pilot: δ_x and δ_y represent the available longitudinal and lateral stick deflections, δ_0 represents the available collective stick deflection, and δ_p represents the pedal deflection. All inputs are measured as percentages of the maximum deflections: δ_x, δ_y, and δ_p are defined in $[-100\%, 100\%]$, δ_0 is defined in $[0, 100\%]$. The models did not include any information regarding the $\mathbf{C}$ and $\mathbf{D}$ matrices used to represent the available outputs of the system. For this analysis, the systems were considered to have full-state feedback: $\mathbf{C}$ was defined as an 11×11 identity matrix, and $\mathbf{D}$ as a zero matrix of size 11×4.

Figure 1 shows the pole loci of the available linearizations, distinguishing between the hover and the 65kn models. The hover linearization is *unstable* as there is a complex pole pair with positive real part; conversely, the 65kn linearization has only a single pole at the origin of the complex plane, and as such may be considered *marginally stable*. The model was further studied by performing an eigenstructure analysis, which enabled the identification of the states involved in each of the system's motions; from this analysis, the system emerged to have significant couplings between the longitudinal and lateral states of the system. Modes were identified using the results of the eigenvector analysis and via a comparison with the theoretical Bo105 model presented in [20]. Although comparable, the modes of the system identified model show a much more significant coupling between the states.

The model was also studied in its controllability and observability by applying the Hautus lemma [24]: the results of this test show that the both linearizations are fully controllable and fully observable. This results is highly relevant in the context of controller development, as they enable the use of specific optimal control techniques such as Linear Quadratic Regulator (LQR) and of its variants. In the context of this article, they enable the application of the Linear Quadratic Integral (LQI) in particular.

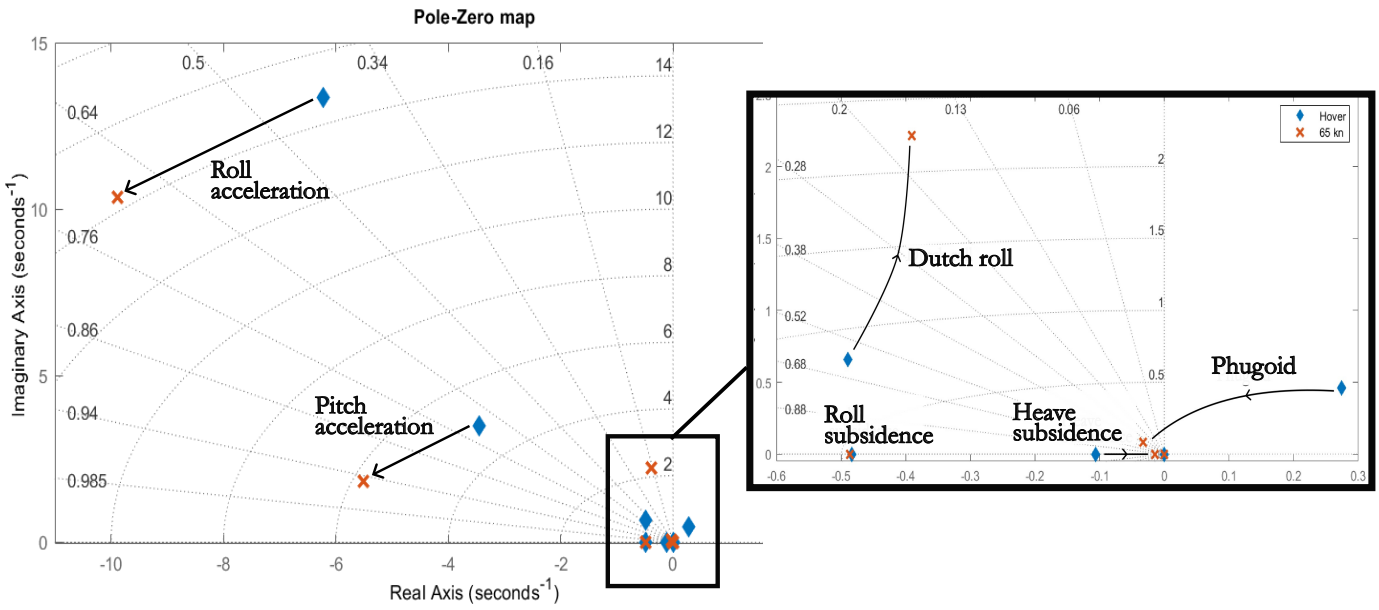


Fig. 1 Pole map of the available Bo105 models

B. Conversion of the Models to the Quaternion Attitude Parametrization

In this section, a procedure to convert the original Euler angles models to have their internal attitude be represented using quaternions is presented and discussed. This procedure facilitates the study of quaternion-based models and control strategies, removing the need to develop complex models and linearizations when Euler angles-based ones are already available. This conversion will only introduce three quaternion elements as states in q_i, q_j, and q_k, omitting the state q_0: this is because quaternions need to remain normalized and, as such, have an internal constraint. This internal constraint would negatively impact the system's controllability, as the four attitude elements could not be individually driven to any desired final state: to preserve controllability for controller analysis, only three quaternion attitude elements are included.

To convert the linearized models, the **A** and **B** matrices are modified using the chain rule. In their general formulations, the states and input matrices are defined as shown in Equation 12, where **f** indicates the general nonlinear differential set of equations describing the states such that $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, and $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$ indicate the trim points for the states and the inputs respectively.

$$\mathbf{A} = \left.\frac{d\mathbf{f}}{d\mathbf{x}^T}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\begin{bmatrix} \frac{df_1}{dx_1} & \cdots & \frac{df_1}{dx_{nx}} \\ \vdots & \ddots & \vdots \\ \frac{df_{nx}}{dx_1} & \cdots & \frac{df_{nx}}{dx_{nx}} \end{bmatrix}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \qquad \mathbf{B} = \left.\frac{d\mathbf{f}}{d\mathbf{u}^T}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\begin{bmatrix} \frac{df_1}{du_1} & \cdots & \frac{df_1}{du_{nu}} \\ \vdots & \ddots & \vdots \\ \frac{df_{nx}}{du_1} & \cdots & \frac{df_{nx}}{du_{nu}} \end{bmatrix}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{12}$$

The first modification to be applied in this methodology is the substitution of the Euler angles with the appropriate quaternion elements in the state vector **x** by changing the partial derivative terms corresponding to the attitude states. To achieve this, the chain rule of derivatives is used to change the matrix elements $\frac{d\mathbf{f}}{d\phi}$, $\frac{d\mathbf{f}}{d\theta}$, and $\frac{d\mathbf{f}}{d\psi}$. These terms are modified as shown in Equation 13.

$$\begin{aligned}
\left.\frac{d\mathbf{f}}{dq_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} &= \left.\frac{d\mathbf{f}}{d\phi}\frac{d\phi}{dq_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\theta}\frac{d\theta}{dq_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\psi}\frac{d\psi}{dq_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \\
\left.\frac{d\mathbf{f}}{dq_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} &= \left.\frac{d\mathbf{f}}{d\phi}\frac{d\phi}{dq_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\theta}\frac{d\theta}{dq_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\psi}\frac{d\psi}{dq_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \\
\left.\frac{d\mathbf{f}}{dq_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} &= \left.\frac{d\mathbf{f}}{d\phi}\frac{d\phi}{dq_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\theta}\frac{d\theta}{dq_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\psi}\frac{d\psi}{dq_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}}
\end{aligned} \tag{13}$$

where the various partial derivatives relating Euler angles and quaternions may be derived from Equations 9, 10, and 11. The state equations of the state-space model obtained after this first step of the algorithm is presented in Equation 14 in matrix form. With this, the state vector has only partially been redefined, as the derivative terms on the left-hand side of the equation still include the time derivatives of the Euler angles. Finalizing the conversion of the model to the quaternion formulation, the second step of the conversion is the substitution of the Euler angles' kinematic differential equations with the linearized quaternion differential equations (obtained linearizing the quaternion time-propagation equations presented in Equation 5). Letting $\dot{\mathbf{q}}_{E2B} = \mathbf{h}(\omega, \mathbf{q}_{E2B})$, the matrix shape of the concluded transformation is presented in Equation 15.

$$
\begin{bmatrix} \dot{u} \\ \vdots \\ \dot{r} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{p} \\ \ddot{q} \end{bmatrix} = \left[\begin{array}{ccc:ccc:cc} \frac{df_u}{du} & \cdots & \frac{df_u}{dr} & \frac{df_u}{dq_i} & \cdots & \frac{df_u}{dq_k} & \frac{df_u}{d\dot{p}} & \frac{df_u}{d\dot{q}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{df_r}{du} & \cdots & \frac{df_r}{dr} & \frac{df_r}{dq_i} & \cdots & \frac{df_r}{dq_k} & \frac{df_r}{d\dot{p}} & \frac{df_r}{d\dot{q}} \\ \hdashline \frac{df_\phi}{du} & \cdots & \frac{df_\phi}{dr} & \frac{df_\phi}{dq_i} & \cdots & \frac{df_\phi}{dq_k} & \frac{df_\phi}{d\dot{p}} & \frac{df_\phi}{d\dot{q}} \\ \frac{df_\theta}{du} & \cdots & \frac{df_\theta}{dr} & \frac{df_\theta}{dq_i} & \cdots & \frac{df_\theta}{dq_k} & \frac{df_\theta}{d\dot{p}} & \frac{df_\theta}{d\dot{q}} \\ \frac{df_\psi}{du} & \cdots & \frac{df_\psi}{dr} & \frac{df_\psi}{dq_i} & \cdots & \frac{df_\psi}{dq_k} & \frac{df_\psi}{d\dot{p}} & \frac{df_\psi}{d\dot{q}} \\ \hdashline \frac{df_{\dot{p}}}{du} & \cdots & \frac{df_{\dot{p}}}{dr} & \frac{df_{\dot{p}}}{dq_i} & \cdots & \frac{df_{\dot{p}}}{dq_k} & \frac{df_{\dot{p}}}{d\dot{p}} & \frac{df_{\dot{p}}}{d\dot{q}} \\ \frac{df_{\dot{q}}}{du} & \cdots & \frac{df_{\dot{q}}}{dr} & \frac{df_{\dot{q}}}{dq_i} & \cdots & \frac{df_{\dot{q}}}{dq_k} & \frac{df_{\dot{q}}}{d\dot{p}} & \frac{df_{\dot{q}}}{d\dot{q}} \end{array}\right]_{\mathbf{x},\mathbf{u}} \begin{bmatrix} u \\ \vdots \\ r \\ q_i \\ q_j \\ q_k \\ \dot{p} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} \frac{df_u}{d\delta_x} & \cdots & \frac{df_u}{d\delta_p} \\ \vdots & \ddots & \vdots \\ \frac{df_{\dot{p}}}{d\delta_x} & \cdots & \frac{df_{\dot{p}}}{d\delta_p} \end{bmatrix}_{\mathbf{x},\mathbf{0}} \begin{bmatrix} \delta_x \\ \vdots \\ \delta_p \end{bmatrix} \tag{14}
$$

$$
\begin{bmatrix} \dot{u} \\ \vdots \\ \dot{r} \\ \dot{q}_i \\ \dot{q}_j \\ \dot{q}_k \\ \ddot{p} \\ \ddot{q} \end{bmatrix} = \left[\begin{array}{ccc:ccc:cc} \frac{df_u}{du} & \cdots & \frac{df_u}{dr} & \frac{df_u}{dq_i} & \cdots & \frac{df_u}{dq_k} & \frac{df_u}{d\dot{p}} & \frac{df_u}{d\dot{q}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{df_r}{du} & \cdots & \frac{df_r}{dr} & \frac{df_r}{dq_i} & \cdots & \frac{df_r}{dq_k} & \frac{df_r}{d\dot{p}} & \frac{df_r}{d\dot{q}} \\ \hdashline \frac{dh_{qi}}{du} & \cdots & \frac{dh_{qi}}{dr} & \frac{dh_{qi}}{dq_i} & \cdots & \frac{dh_{qi}}{dq_k} & \frac{dh_{qi}}{d\dot{p}} & \frac{dh_{qi}}{d\dot{q}} \\ \frac{dh_{qj}}{du} & \cdots & \frac{dh_{qj}}{dr} & \frac{dh_{qj}}{dq_i} & \cdots & \frac{dh_{qj}}{dq_k} & \frac{dh_{qj}}{d\dot{p}} & \frac{dh_{qj}}{d\dot{q}} \\ \frac{dh_{qk}}{du} & \cdots & \frac{dh_{qk}}{dr} & \frac{dh_{qk}}{dq_i} & \cdots & \frac{dh_{qk}}{dq_k} & \frac{dh_{qk}}{d\dot{p}} & \frac{dh_{qk}}{d\dot{q}} \\ \hdashline \frac{df_{\dot{p}}}{du} & \cdots & \frac{df_{\dot{p}}}{dr} & \frac{df_{\dot{p}}}{dq_i} & \cdots & \frac{df_{\dot{p}}}{dq_k} & \frac{df_{\dot{p}}}{d\dot{p}} & \frac{df_{\dot{p}}}{d\dot{q}} \\ \frac{df_{\dot{q}}}{du} & \cdots & \frac{df_{\dot{q}}}{dr} & \frac{df_{\dot{q}}}{dq_i} & \cdots & \frac{df_{\dot{q}}}{dq_k} & \frac{df_{\dot{q}}}{d\dot{p}} & \frac{df_{\dot{q}}}{d\dot{q}} \end{array}\right]_{\mathbf{x},\mathbf{u}} \begin{bmatrix} u \\ \vdots \\ r \\ q_i \\ q_j \\ q_k \\ \dot{p} \\ \dot{q} \end{bmatrix} + \left[\begin{array}{ccc} \frac{df_u}{d\,\delta_x} & \cdots & \frac{df_u}{d\,\delta_p} \\ \vdots & \ddots & \vdots \\ \frac{df_r}{d\,\delta_x} & \cdots & \frac{df_r}{d\,\delta_p} \\ \hdashline \frac{dh_{qi}}{d\,\delta_x} & \cdots & \frac{dh_{qi}}{d\,\delta_p} \\ \frac{dh_{qj}}{d\,\delta_x} & \cdots & \frac{dh_{qj}}{d\,\delta_p} \\ \frac{dh_{qk}}{d\,\delta_x} & \cdots & \frac{dh_{qk}}{d\,\delta_p} \\ \hdashline \frac{df_{\dot{p}}}{d\,\delta_x} & \cdots & \frac{df_{\dot{p}}}{d\,\delta_p} \\ \frac{df_{\dot{q}}}{d\,\delta_x} & \cdots & \frac{df_{\dot{q}}}{d\,\delta_p} \end{array}\right]_{\mathbf{x},\mathbf{0}} \begin{bmatrix} \delta_x \\ \vdots \\ \delta_p \end{bmatrix} \tag{15}
$$

The methodology presented here delivers a straightforward yet mathematically rigorous pathway for transforming any existing Euler-angle–based state-space model into its quaternion-parametrized equivalent. By systematically re-expressing the attitude dependencies through the chain rule and embedding the quaternion kinematics directly into the state matrices, this conversion preserves every dynamic mode and control-relevant characteristic of the original system. Once again, the **C** and **D** matrices were integrated to be an identity matrix of size 11×11 and a zero matrix of size 11×4, allowing for full-state feedback.

C. Verification of the Quaternion Conversion Procedure

The procedure presented in subsection III.B enables a conversion of the kinematic representation of attitude of any existing linearized system to have it represented using quaternions. To verify the correctness of the procedure, three comparisons were identified: comparison of the systems' eigenstructures, comparison of the systems' controllability/observability, and comparison of the system's dynamic responses.

Starting the verification of the quaternion conversion, the poles and eigenvectors of the systems are compared. Figure 2 shows the locations of the poles of the original system. The poles are identified to be identical to the ones found for the original system, maintaining the same stability characteristics. Further, an eigenvector analysis was conducted on the two systems: again, the eigenvectors of the converted quaternion models coincide with the values of the original systems. The congruency between the eigenstructures of the original model and of the converted one supports the validity of the procedure presented.

Supporting the results obtained, the observability and controllability of the systems were analysed. These properties were checked by means of the Hautus lemma, as was already explained in subsection III.A; again, the converted models are both fully controllable and fully observable, matching the controllability and observability properties of the original linearization. These congruent characteristics support the validity of the transformation presented. Notice that the retention of the controllability property is subordinate to the inclusion of only three of the four quaternion elements as states, as the attitude states would not be independent of one another due to the unitary norm condition; including all four quaternion elements would have impaired controllability, impacting controller design.

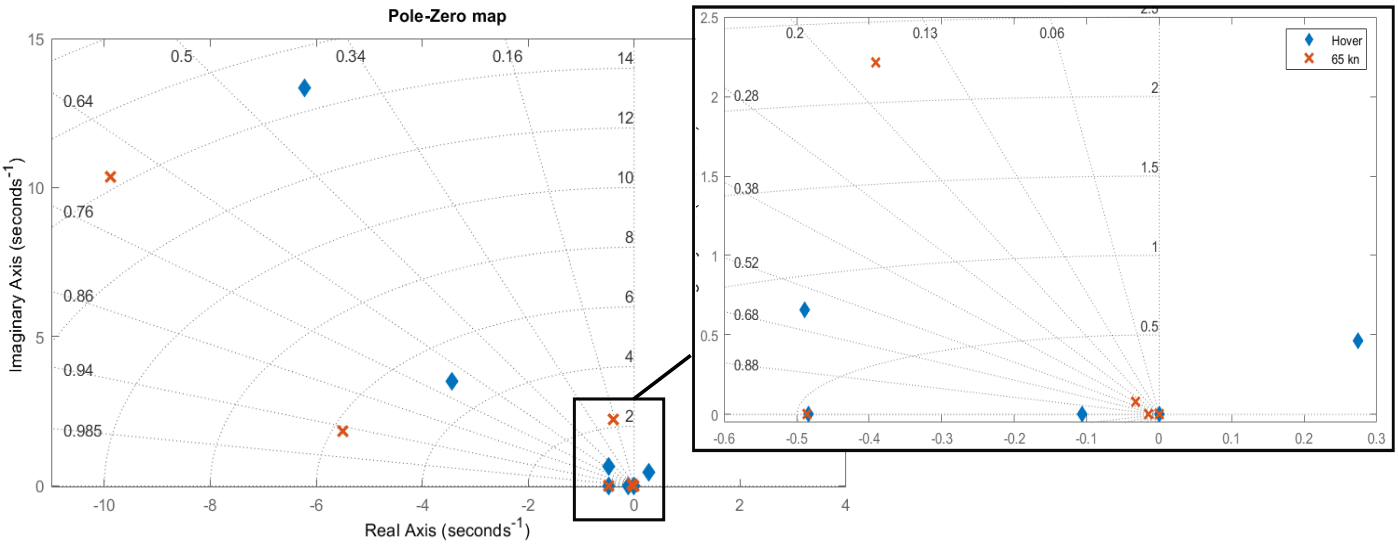


Fig. 2 Pole map of the Bo105 models converted to quaternions

As a final check of the model conversion validity, the responses of the systems to identical inputs are compared. These inputs were defined to have a total duration of 10s and are composed of a square pulse of 10% amplitude with a duration of 6s and an initial delay of 2s. A total of four inputs were defined, each activating only one of the four available model inputs $[\delta_x, \delta_y, \delta_0, \delta_p]$ at a time. The responses of the models were evaluated and compared for all four inputs and for both linearizations: for brevity, here only the results of the first input stimulation are shown (square pulse on the longitudinal cyclic in hover). Figure 3 show a comparison of the responses of the two systems in hover, with the original model represented by blue dots and the quaternion model by red squares. In all cases the states of the systems overlap exactly, further suggesting that the conversion of the model was successful in maintaining the dynamic properties of the system unaltered independently of its linearization point, while changing the internal attitude parametrization.

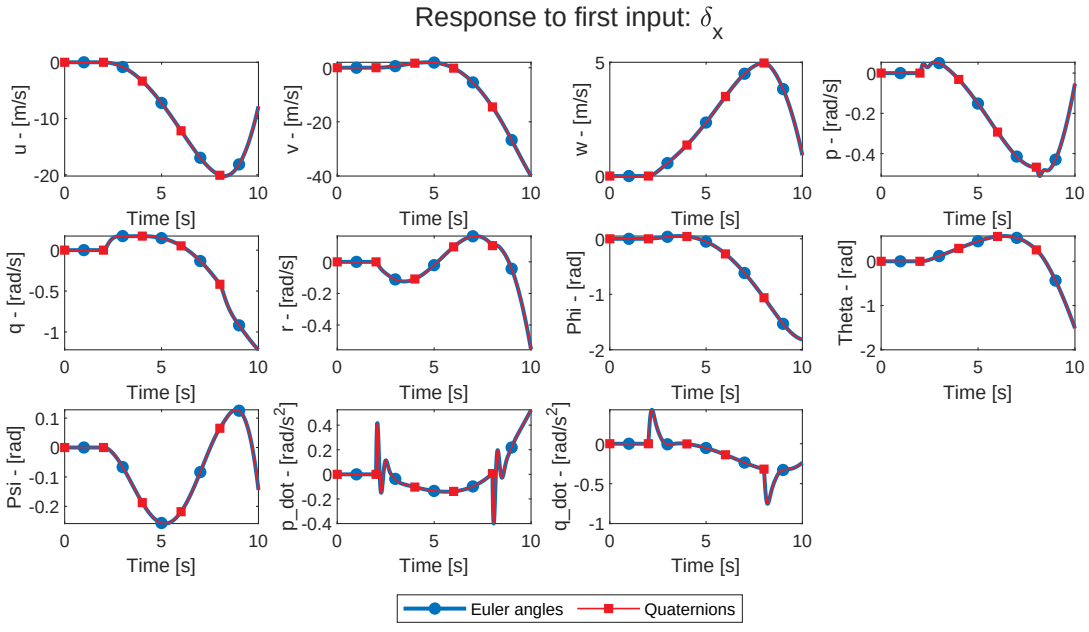


Fig. 3 Time response of the hover state-space: input applied on δ_x

With the discussion presented in section III, a complete procedure to convert an existing Euler angles-based linearized model has been presented. In subsection III.A, a description of the available linearizations is provided, discussing the characteristics of the models. In subsection III.B, a procedure to convert existing linearized models from the commonly-used Euler angles parametrization to the quaternion attitude parametrization is introduced. Further, in subsection III.C, the procedure is applied to the two available linearizations, and the conversion is verified by ensuring that the quaternion model is still representative of the original system by comparing eigenstructures, controllability and observability, and dynamic responses of the original and the converted models for both the available linearizations. The results are positive and suggest that the conversion procedure was performed correctly.

IV. Development of a Quaternion-Based Full Flight Control System

Building upon section III, the converted quaternion model is further studied to implement a quaternion-based full flight controller. Here, the multi-loop controller architecture developed will be presented in subsection IV.A. In subsection IV.B, the tuning of the controller is discussed, implementing the Particle Swarm Optimization (PSO) algorithm. As this paper investigates the implementation of quaternions in autonomous agile control, only the 65**kn** linearization will be considered.

A. Quaternion-Based Controller Architecture

Having changed the internal attitude representation of the system, the implementation of a full quaternion-based controller is now investigated. Quaternions are naturally adept at encoding smooth rotations in 3D space with no singularities, accounting for simultaneous changes in yaw, pitch, and roll; this natural flexibility makes their implementation in controllers advantageous, especially when compared to other commonly used attitude parametrizations [6, 7]. Contributing to the academic discussion about quaternion controllers, a three-loop controller with LQI for attitude control is presented in this section.

The three nested loops controller is shown in Figure 4. The innermost loop is the *Attitude Controller*: it leverages the system's full-state feedback to implement LQI for quaternion attitude tracking. The intermediate loop is the *Velocity Controller*: it leverages a Proportional-Integral (PI) controller architecture to enable good tracking performance and compensate steady-state errors in the velocity channels. The outermost loop is the *Position Controller*: it implements a Proportional (P) controller to command adequate velocity references that drive the system to track a desired trajectory. All the controller stages are augmented with feed-forward commands (denoted by the $\star$ superscript) generated by an external autopilot module, accelerating the system's response. Additionally, in both the velocity and the position loops, no derivative action is implemented: this is because derivative controllers, while they are effective in speeding up the system's response, are inadequate for application in aerospace systems due to their high sensitivity to sensor noise; to create a balanced structure that could also be applied to real systems, derivative action was avoided in this study.

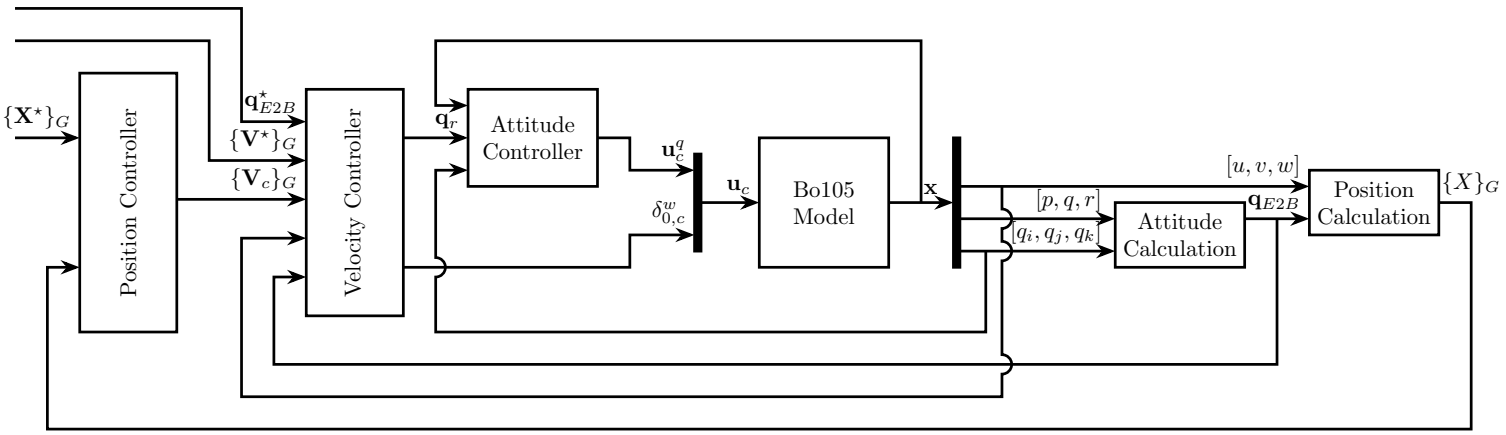


Fig. 4 Block diagram of the complete control system

1. Attitude Controller

The innermost loop is the *Attitude Controller*, tasked with controlling the attitude elements $[q_i, q_j, q_k]$ to drive the system's orientation. Leveraging the natural controllability and observability of the plant, the attitude loop implements an LQI controller for attitude control. This control technique, introduced by Young and Willems [25], is a variation of the Linear Quadratic Regulator (LQR) which modifies the system to introduce integral tracking action. As the system is a highly coupled MIMO system, the optimization-based LQI approach allows for the control of simultaneous states while granting good robustness [26] at the entrance and exit of the plant and limiting control effort [27–29].

To mimic the control behaviour of helicopter pilots, the attitude of the vehicle is controlled using only the cyclic $[\delta_x, \delta_y]$ and the pedal δ_p, isolating the collective command δ_0 to control the vertical velocity. Only for this loop, the linearized system was modified by removing the columns of the **B** and **D** associated with the δ_0 input, which will be instead controlled as a feed-forward heave command (as visible in Figure 4). Similarly, the w state is also removed by eliminating the respective row and column in the linearized matrices. This modified linearized system was verified to be fully controllable and observable, enabling the use of LQ techniques.

The LQI controller $\mathbf{K} = [\mathbf{K}_x, \mathbf{K}_i]$ integrates both a proportional regulatory action and an integral tracking action and was implemented as shown in Figure 5, where $\mathbf{x}$ indicates the complete state vector of the quaternion-converted system, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ represent the state-space matrices of the system (with δ_0 and w removed), and $\mathbf{y}_r = [q_i, q_j, q_k]$ indicates the selected tracked states ($\mathbf{C}_r$ is a matrix appropriately designed to extract the tracked states from $\mathbf{x}$). In the diagram, $\mathbf{r}$ is the reference signal vector, generated by accounting for the feedforward

attitude command and the corrective action generated by the velocity loop (as discussed in Section IV.A.2): the tracking action is applied on the integral of the tracking error $\dot{\mathbf{x}}_i = \mathbf{r} - \mathbf{y}_r$, improving tracking performance. The state-space equation of this augmented system is represented mathematically in Equation 16.

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{x}}_i(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C}_r & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{x}_i(t) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \mathbf{u}(t) + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{r}(t) \tag{16}$$

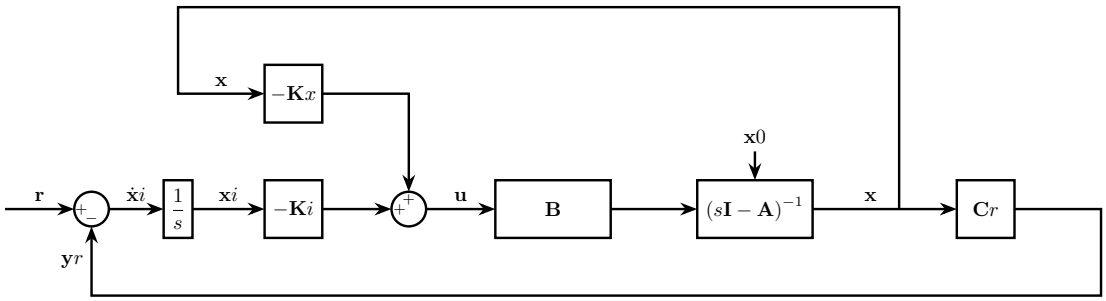


Fig. 5 Block diagram of the LQI attitude controller

To identify the best controller gains $\mathbf{K}$, LQI requires the minimization of the Quadratic Performance Index (QPI) J, where $\mathbf{Q}$ and $\mathbf{R}$ are two user-defined matrices that determine the weighting between higher performance and more conservative control actions, and $\mathbf{x}$ and $\mathbf{u}$ are the states and input vectors of the augmented system in Equation 16. In particular, higher values of $\mathbf{Q}$ will prompt a faster response, improving the dynamic behaviour of the system. Conversely, higher values of $\mathbf{R}$ will reduce control effort, limiting the activation of the controls.

$$J = \int_0^\infty \left(\mathbf{x}(\tau)^T \mathbf{Q} \mathbf{x}(\tau) + \mathbf{u}(\tau)^T \mathbf{R} \mathbf{u}(\tau) \right) d\tau \tag{17}$$

For this equation to represent a convex optimization problem with an identifiable minimum, $\mathbf{Q}$ and $\mathbf{R}$ must be, respectively, symmetric positive semi-definite and positive definite. The controller $\mathbf{K}$ that minimizes the QPI is found using Equation 18 [30, 31], where P is the solution to the Algebraic Riccati Equation (ARE) in Equation 19, and $\mathbf{A}$ and $\mathbf{B}$ are the matrices of the augmented system in Equation 16. To identify the most adequate values for $\mathbf{Q}$ and $\mathbf{R}$, an optimization-based tuning approach was selected, as discussed in Section IV.B.2.

$$\mathbf{K} = [\mathbf{K}_x, \mathbf{K}_i] = \mathbf{R}^{-1} \mathbf{B}^T \mathbf{P} \tag{18}$$

$$\mathbf{A}^T \mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T \mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{19}$$

2. Velocity Controller

The intermediate loop is the *Velocity Controller*, which controls the velocity by appropriately influencing the attitude inner loop controller and directly commanding the collective. The velocity controller is implemented using a PI architecture, which offers high flexibility in implementation and effective control action.

A representation of the controller is provided in Figure 6. The velocity reference is first calculated in the global reference frame $\mathcal{G}$ by summing the feedforward velocity signal with the corrective signal produced by the position controller. This reference velocity is then converted from the global reference frame to the body reference frame $\mathcal{B}$ in the *Global-to-Body* block: this block also takes as an input the aircraft's current attitude $\mathbf{q}_{E2B}$ and executes a passive rotation implementing Equation 7. In the $\mathcal{B}$ reference frame, the velocity error $\Delta[u, v, w]$ is calculated, and each element is then passed through a different PI controller. The heave error is fed directly to the collective command δ_0. The errors on the surge and sway velocities are used to compute the correction angles $\Delta\Theta$ and $\Delta\Phi$, which are used to produce a quaternion attitude reference in the *Quaternion Reference Generation* block.

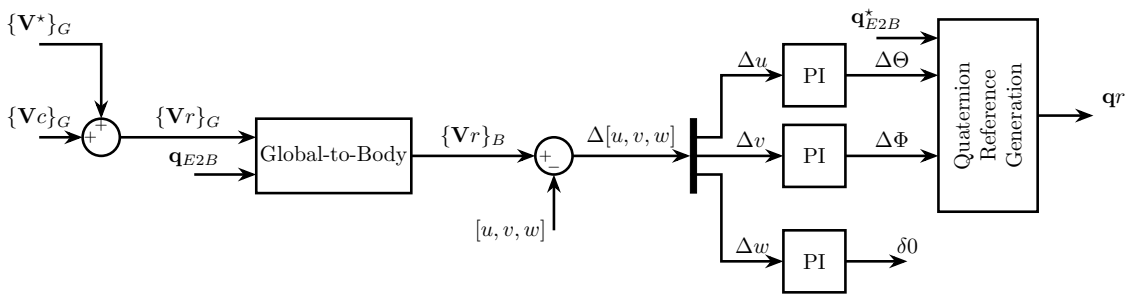


Fig. 6 Block diagram of the PI velocity controller

The quaternion attitude reference $\mathbf{q}_r$ is generated in the *Quaternion Reference Generation* by concatenating two attitude references, as per Equation 20. In the equation, $\mathbf{q}^\star_{E2B}$ is the feedforward quaternion attitude (which is generated based on offline-defined data by the

autopilot, as discussed in Section V) and $\mathbf{q}_\Delta$ is the corrective quaternion. This corrective quaternion is generated according to Equation 21, where Equations 22 to 24 show the generation of the required parameters.

$$\mathbf{q}_r = \mathbf{q}^{\star}_{E2B}\mathbf{q}_\Delta \tag{20}$$

$$\mathbf{q}_\Delta = \begin{bmatrix} \cos(\zeta/2) \\ c_1 \sin(\zeta/2) \\ c_2 \sin(\zeta/2) \\ 0 \end{bmatrix} \tag{21}$$

$$\zeta = \sqrt{\Delta\Theta^2 + \Delta\Phi^2} \tag{22}$$

$$c_1 = \begin{cases} 0 & \text{if} \quad \zeta = 0 \\ \dfrac{\Delta\Phi}{\zeta} & \text{else} \end{cases} \tag{23}$$

$$c_2 = \begin{cases} 0 & \text{if} \quad \zeta = 0 \\ \dfrac{\Delta\Theta}{\zeta} & \text{else} \end{cases} \tag{24}$$

The quaternion $\mathbf{q}_r$ must be disambiguated to ensure it commands the smallest rotation from the current attitude $\mathbf{q}_{E2B}$, following the most optimal rotation to achieve the reference attitude. To obtain this, let $\Delta\mathbf{q}$ be the quaternion describing the rotation between the current helicopter's attitude and the reference helicopter attitude $\mathbf{q}_r$ calculated with Equation 20.

$$\Delta\mathbf{q} = \mathbf{q}^{-1}_{E2B}\mathbf{q}_r \tag{25}$$

The reference quaternion $\mathbf{q}_r$ is then reconstructed using Equation 26, where $\hat{\Delta\mathbf{q}}$ is equal to $\Delta\mathbf{q}$ if its first component is is greater or equal to zero, or it is set equal to $-\Delta\mathbf{q}$ otherwise. This ensures that the reference attitude is chosen to have the shortest rotation from the original attitude, avoiding the unwinding phenomenon that would instead cause the controller to command a longer rotation to the system.

$$\mathbf{q}_r = \mathbf{q}_{E2B}\hat{\Delta\mathbf{q}} \tag{26}$$

From this corrective equation, it is also worth pointing out that, in this case, no dynamic online corrective action was implemented on the yaw direction.As such, it was chosen for yaw reference signals to be only determined offline and passed as part of the feedforward attitude. This was done to allow more flexibility and to streamline the offline trajectory planning and reference generation.

3. Position Controller

The final loop of the control system is the position controller, which generates a corrective velocity signal to drive the system to follow a predetermined path. The position controller only implements a proportional corrective action with three P controllers: this approach was chosen because of its simplicity and to avoid unnecessary integrator-like behaviours, which would otherwise slow down the system and increase its complexity. The implementation of the proportional controller is immediate, as only a proportional action is required to produce an adequate corrective velocity signal: this is shown in Figure 7.

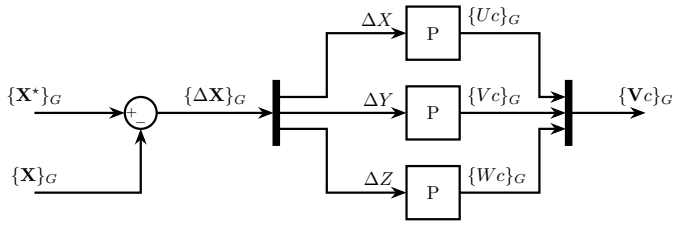


Fig. 7 Block diagram of the position controller

With this, the controller architecture has been fully introduced, presenting and motivating the specific control frameworks utilized in each loop. The implementation of the controller was also exemplified through appropriate block diagram representations, allowing for a clear interpretation of the control action and of the fundamental insight that brought to this structure. The tuning of the controller is discussed in subsection IV.B, where the optimization-based approach chosen is presented and applied to the three loops of the system.

B. Tuning of the Controller

Tuning the ontroller, an optimization based approach was followed by implementing the Particle Swarm Optimization (PSO) algorithm. In Section IV.B.1, the PSO algorithm is introduced and presented, providing guidelines for its implementation. PSO is then applied to the three loops sequentially: in Section IV.B.2, the tuning of the attitude control is discussed; in Section IV.B.3, the velocity controller tuning is analysed; and in Section IV.B.4, the tuning of the position controller is shown. For readability, the full results of the controller tuning are shown only for the position controller; the attitude and velocity controller tuning results will only be shown partially to discuss relevant trends noticed in the system's response.

1. Particle Swarm Optimization (PSO) Algorithm

To tune the controller, an optimization-based approach was chosen by employing the PSO algorithm [32]. This optimization-based approach was selected to limit the need for hand-tuning and to provide a structured methodology for the identification of controller gains that could be replicated in other control scenarios [33]. Here, the PSO algorithm is briefly introduced and its algorithmic representation discussed.

PSO is a population-based stochastic optimization technique based on the action of simple particles (agents that encode possible solutions to the optimization problem) that collaborate through information sharing to explore a complex search space effectively and converge toward optimal solutions. This concept is based on the observation that social behavior in nature often leads to emergent intelligence, where the group as a whole can solve problems more efficiently than any individual acting alone [34].

In the context of control system tuning, PSO provides a framework for adjusting controller parameters by simulating a distributed search process where each particle represents a potential set of parameters, and the collective dynamics guide the search toward improved system performance. As an added advantage, multimodal optimization is highly effective when using PSO, enabling the definition of objective functions composed of multiple contrasting terms, finding a balance between high performance and robustness. These characteristics make the PSO an effective approach to the tuning of control systems, with current research around this method suggesting its validity in the context of tuning controllers for aircraft, automotive and naval vehicles, and industrial plants alike [33].

In the PSO algorithm, possible solutions are encoded as individual particles $\mathbf{p}_i$, each used to evaluate a user-defined fitness function $\mathbf{J}(\mathbf{p})$ to be minimized. Consider the algorithm to be at iteration t and to have been initialized over N particles. The parameters contained in each particle evolve over successive iterations, and at each iteration, the algorithm keeps track of three variables: the current particle position $\mathbf{p}_i(t) \;\; \forall i = 1 \dots N$, the position $\mathbf{p}_i^\star \;\; \forall i = 1 \dots N$ that for each particle led to its personal best result $\mathbf{J}_i^\star$, and the particle position $\mathbf{g}^\star$ associated with the most desirable cost $\mathbf{J}^\star$ across all N particles. The update equation for the PSO solutions is presented in Equation 27, where $\mathbf{v}_i$ is the "particle velocity" term for the considered particle i.

$$\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + \mathbf{v}_i(t+1) \tag{27}$$

The velocity variable is calculated at each iteration of the algorithm and it is defined by an equation that encapsulates three fundamental influences: inertia, cognitive drive, and social influence. The inertia term enables the particle to maintain its current momentum, thus preserving the direction of movement. The cognitive term is a reflection of the particle's own experience, encouraging a return to positions that previously yielded high fitness values. The social term, on the other hand, represents the influence of the best-performing particle in the swarm, guiding the particle toward areas of the search space that have been successful. Mathematically, the velocity update is calculated using Equation 28. Here, an explanation of the individual terms of the equation is provided afterwards.

- $\mathbf{v}_i(t)$ is the velocity of the particle i at the previous iteration of the loop (initiated at 0 for the first iteration of the algorithm).
- w is the inertia weight, a scalar determining how much of the previous particle velocity is transferred to the current velocity.
- c_1 and c_2 are the cognitive acceleration coefficient and the social acceleration coefficient, which respectively determine how much of the velocity component is going to drive the particle towards its current best $\mathbf{p}_i^\star$ or the identified global best $\mathbf{g}^\star$ respectively.
- $\mathbf{r}_1$ and $\mathbf{r}_2$ are uniformly-distributed random values in the range $[0, 1]$ which aim at including a stochastic element to the algorithm, which in turn allows for a better exploration of the solution space.

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1(\mathbf{p}_i^\star - \mathbf{p}_i(t)) + c_2\mathbf{r}_2(\mathbf{g}^\star - \mathbf{p}_i(t)) \tag{28}$$

Following the practical recommendations listed in [34], the model's hyperparameters were selected to be $w = 1$, $c_1 = 1.5$, $c_2 = 2$, which were empirically found to yield good optimization performance across most applications [34]. By setting $w = 1$, each particle carries over a larger portion of its momentum, and as such is encouraged to fully explore the solution space of the problem. c_1 and c_2 are generally chosen between 1 and 2: to encourage the particles to have a more coherent global behaviour without neglecting the cognitive component, the values typically chosen are $c_1 = 1.5$ and $c_2 = 2$. These parameters strike a balance between exploration of the solution space and convergence, encouraging the model to explore more readily the most promising solutions without neglecting the influence of the individual history of the particles.

2. Tuning of the Attitude Controller

The tuning process started by considering the innermost loop of the system: the attitude loop. The objective of this controller is to enable the system to perform rapid and precise attitude changes, achieving both good dynamic response and steady-state performance. To achieve this, the attitude loop makes use of the LQI control framework, which was introduced in Section subsubsection IV.A.1 and is effective when controlling MIMO systems as it enforces on-axis tracking while limiting off-axis cross-activation.

To completely determine the controller, appropriate $\mathbf{Q}$ and $\mathbf{R}$ matrices must be identified and their parameters encoded as particles in the PSO algorithm, however, encoding each individual element in the PSO algorithm may cause the reconstruction of matrices that do not respect the positive definite and semi-definite properties. To avoid this, the tuning matrices were defined to be diagonal matrices with non-negative elements, an approach used in LQ tuning to achieve good performance while guaranteeing good margins in MIMO systems [26, 30].

Given the size of the system and recalling the removal of the heave state and of the collective input, the $\mathbf{Q}$ matrix must be a square 13×13 matrix and the $\mathbf{R}$ matrix must be of size 3×3. As these were defined to be purely diagonal matrices, each particle of the PSO algorithm $\mathbf{p}$ must contain 16 elements for the tuning: the particles were then defined to be 16×1 vectors, where the first 13 elements encode the diagonal elements of $\mathbf{Q}$ and the last 3 elements the diagonal elements of $\mathbf{R}$.

To determine the allowable values of the particle elements for the PSO algorithm, Bryson's rule may be used [35], which indicates that the **Q** and **R** matrices may be defined as diagonal matrices, with the diagonal elements of **Q** defined as the inverse of the squared allowed error of the respective state they map, and the diagonal elements of **R** defined as the inverse of the squared maximum acceptable value of the respective input they map. This empirical rule is summed up in Equation 29 and is commonly used to initialize **Q** and **R**.

$$\begin{aligned} Q_{j,j} &= \frac{1}{[max\ allowed\ error\ on\ x_j]^2} \qquad \forall j = 1, \dots, 14 \\ R_{k,k} &= \frac{1}{[max\ allowed\ value\ of\ u_k]^2} \qquad \forall k = 1, \dots, 3 \end{aligned} \tag{29}$$

The tuning was performed by simulating three attitude tracking tasks, each commanding a separate attitude variation along one of the three helicopter axis. In particular, in the first simulation, a rotation of 25° around the body x-axis was commanded. In the second simulation, the commanded rotation was of 45° around the body y-axis. Lastly, in the third simulation, a rotation of 60° around the body z-axis was defined. For the purpose of notation, these values will be indicated in the superscript of the variables (i.e.: $q_i^1(t)$ indicates the attitude q_i element of the first simulation run at time t).

The cost function for the optimization of the attitude controller's parameters is shown in Equation 30. The cost has three separate terms, scaled with an appropriate weight W. The first term is the total squared error in the response and aims to improve response accuracy by minimizing the tracking error over the entire response. The second term penalizes large control deviations from the trim position, avoiding saturation by adjusting the applied weights W_x, W_y, W_p by increasing their value if the commanded control deflection would saturate the respective input. The third term weights the 5% settling time of the tracked attitude channels, speeding up the system's response.

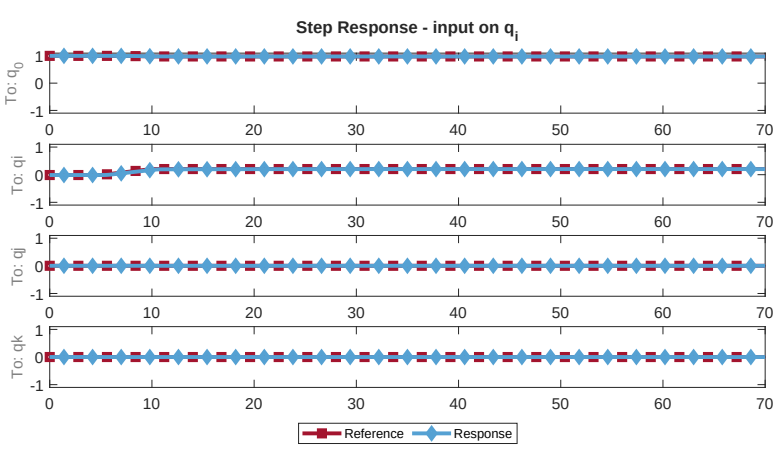
$$\begin{aligned} z = &\sum_{l=1}^{3} W_R \left(\sum_{t=1}^{T} (\Delta q_0^l(t))^2 + \sum_{t=1}^{T} (\Delta q_i^l(t))^2 + \sum_{t=1}^{T} (\Delta q_j^l(t))^2 + \sum_{t=1}^{T} (\Delta q_k^l(t))^2 \right) + \dots \\ &\sum_{l=1}^{3} \left(\sum_{t=1}^{T} \left(W_x \Delta\delta x^l(t)^2 + W_y \Delta\delta y^l(t)^2 + W_p \Delta\delta p^l(t)^2 \right) \right) + \dots \\ &W_T \left(Ts(q_i^1(t)) + Ts(q_j^2(t)) + Ts(q_k^3(t)) \right) \end{aligned} \tag{30}$$

The calculation of the attitude error $\Delta\mathbf{q}$ is carried out by first identifying the displacement quaternion. Considering $\mathbf{q}_r$ to be the reference attitude and $\mathbf{q}_{E2B}$ to be the measured attitude of the system, the difference between these two quaternions is identified by the displacement quaternion $\mathbf{q}_\Delta$, calculated according to Equation 31. This intermediate quaternion quantifies the rotational distance between the current and the desired attitude of the system. When two systems are aligned, the quaternion describing the rotation is the identity quaternion $\mathbf{q} = [1, 0, 0, 0]^T$; the attitude error is then calculated according to Equation 32.

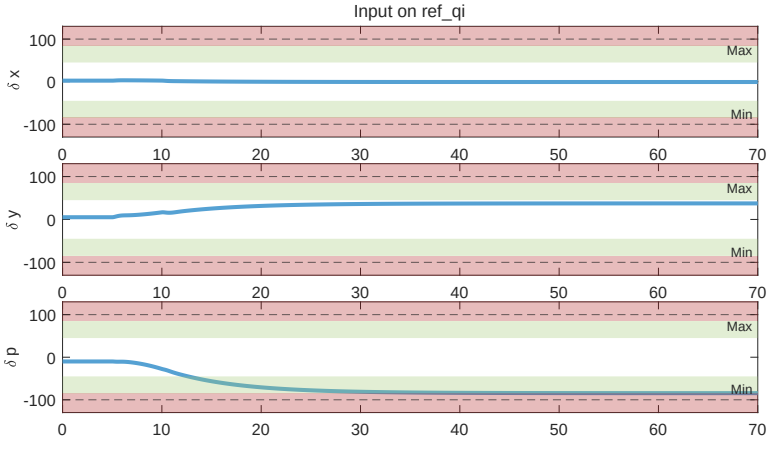
$$\mathbf{q}_\Delta = \mathbf{q}_{E2B}^{-1} \mathbf{q}_r \tag{31}$$

$$\Delta\mathbf{q} = [1, 0, 0, 0]^T - \mathbf{q}_\Delta \tag{32}$$

As performance of the tuned system are comparable commanding rotations along all three body axes, only the response of the system to a commanded rotation along the body x-axis is shown in Figure 8, where the tracked attitude states and the controls required to obtain this command are represented. In the commanded control deflections, it may be noticed that the control band is divided in three sections, differently coloured based on the weights W_x, W_y, and W_z applied in those sections: the central white regions have the smallest applied weights, as the input channels remain around the command's trim point; the green regions have a higher weighting, limiting control action if not required; the red regions have the highest weighting, discouraging the optimization algorithm from settling at solutions that would saturate the inputs. Tracking is being performed adequately, with no steady-state error and good dynamic performance without saturating the control inputs. These performance are consistent along all three control axis, with responsive tracking and contained control usage.



(a) First simulation - attitude signals



(b) First simulation - control signals

Fig. 8 First simulation: 25° rotation around the body x-axis

3. Tuning of the Velocity Controller

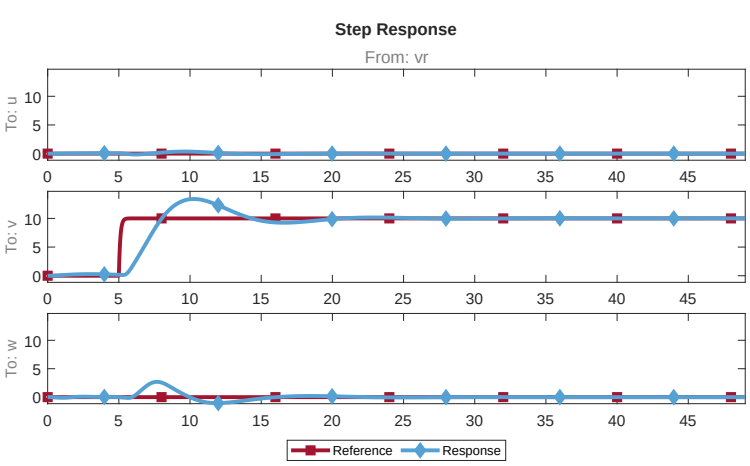
Building upon the results of the attitude controller tuning, the velocity controller is tuned next. As the controller architecture is chosen to have three PI controllers, the tuning algorithm has to identify six parameters: each particle is encoded as a 6×1 vector where each element is one of the system's gains. The tuning for the velocity controller was carried out in a manner similar to the attitude controller tuning. Three velocity tracking tasks were defined (each indicated with the index $l = 1, \ldots, 3$), each activating one of the available velocity channels and commanding a variation of 10m/s from the trim point. This variation is not commanded as a step input, but is instead filtered through a first-order lag with time constant $\tau = 0.1$s, commanding a smooth but fast variation of the commanded velocity.

The objective function is shown in Equation 33. The function is divided in four terms, each weighted by an appropriate scalar parameter W in a manner analogous to the one shown for Equation 30. The first term evaluates the squared error in each of the velocity channels for all tracking tasks, penalizing poor tracking performance. The second term discourages attitude tracking error, which is calculated using Equation 32. The third term limits command activation, with weights applied based on the entity of the activation to avoid saturation. The fourth term weighs the 5% settling time for the velocity term activated in each of the simulations, encouraging faster response.

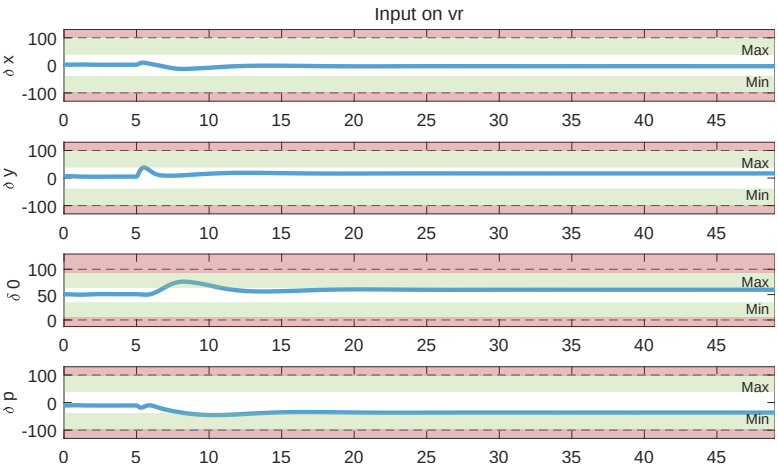
$$
\begin{aligned}
z = &\sum_{l=1}^{3} W_{Rv} \left(\sum_{t=1}^{T} (u_r^l(t) - u^l(t))^2 + \sum_{t=1}^{T} (v_r^l(t) - v^l(t))^2 + \sum_{t=1}^{T} (w_r^l(t) - w^l(t))^2 \right) + \ldots \\
&\sum_{l=1}^{3} W_{Ra} \left(\sum_{t=1}^{T} (\Delta q_0^l(t))^2 + \sum_{t=1}^{T} (\Delta q_i^l(t)(t))^2 + \sum_{t=1}^{T} (\Delta q_j^l(t))^2 + \sum_{t=1}^{T} (\Delta q_k^l(t))^2 \right) + \ldots \\
&\sum_{l=1}^{3} \left(\sum_{t=1}^{T} \left(W_x \Delta\delta x^l(t)^2 + W_y \Delta\delta y^l(t)^2 + W_p \Delta\delta p^l(t)^2 + W_0 \Delta\delta_0^l(t)^2 \right) \right) + \ldots \\
&W_T \left(Ts(u^1(t)) + Ts(v^2(t)) + Ts(w^3(t)) \right)
\end{aligned}
\tag{33}
$$

The system shows consistent behaviour when commanding velocity changes along all velocity channels: to exemplify the results of the tuning, only the response when commanding a sway input v is shown in Figure 9. The tuning highlights some criticalities in the velocity controller, with response delays and cross-activations between velocity channels; as predicted by the eigenstructure analysis, there is a noticeable activation of w when commanding inputs on u and v, causing observable differences between the reference and the response signals. Despite this, the controller demonstrates strong performance overall, achieving accurate tracking of the desired velocity channels with no steady-state error and maintaining effective attitude control with contained tracking error. The commanded actuator deflections are also within the allowed limits, allowing for the performance of velocity changes without saturating the actuators. These results confirm the suitability of the optimized velocity controller and justify the tuning approach considered, as the PSO algorithm was capable of identifying parameters that could balance dynamic performance and actuator effort for the tracking tasks presented.

When tuning the control system, it was found that the inclusion of the attitude tracking penalization term was highly beneficial to the overall performance of controlled system, as the attitude error penalization term forced the tuning algorithm to select controller gains that would not overwhelm the capabilities of the attitude inner loop, ensuring a smoother integration of the velocity controller. Without the inclusion of this term, the PSO algorithm was found to converge to solutions with higher velocity controller gains, which would cause the system to produce large attitude references that the innermost loop was unable to follow adequately: while this was effective in achieving faster velocity response, it also caused a significant degradation in the robustness of the controller, making the system less reliable.



(a) Second simulation - velocity signals



(b) Second simulation - control signals

Fig. 9 Second simulation: 10m/s **sway velocity increase**

4. Tuning of the Position Controller

The tuning procedure for the position controller mimics the one defined for the tuning of the velocity controller in Section IV.B.3. The position controller requires three proportional gains to be defined, so the particles are defined as 3×1 vectors. The tuning was performed defining three isolated tracking tasks, each exciting one position channel while leaving the others unaltered. The tracking inputs for the position channels were defined by integrating velocity commands identified by applying over a simulation time of 100s a velocity variation of 10m/s to each velocity axis in the global reference frame, according to Equation 34. The resulting position references are shown in Figure 10, where the three columns represent the various simulation runs, while the rows indicates the excitation channels X, Y, and Z.

$$\Delta V(t) = \begin{cases} 10 & \text{if} \quad 10 < t \leq 20 \ \vee \ 60 < t \leq 70 \\ -10 & \text{if} \quad 30 < t \leq 50 \\ 0 & \text{else} \end{cases} \tag{34}$$

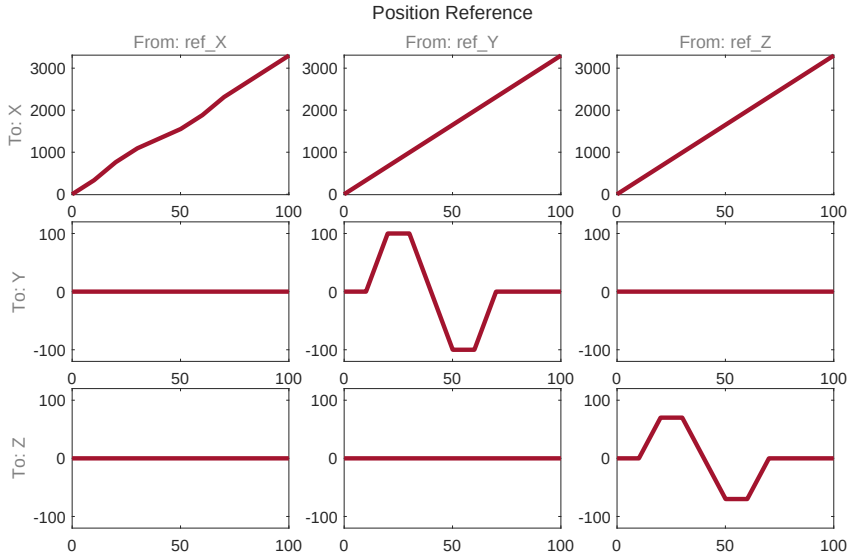


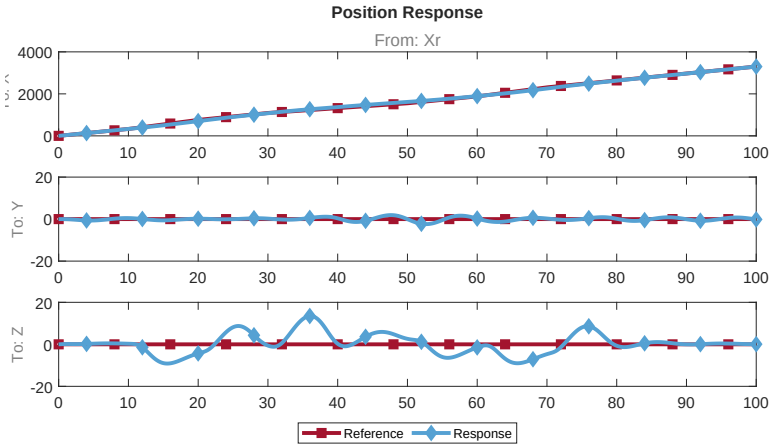
Fig. 10 Position references for the position controller tuning

Equation 35 is the cost function minimised in the position controller tuning, and it is composed of four terms. The first term calculates the squared error on the position reference tracking error along the three control axes for all simulations. The second and third terms weight the velocity and attitude reference tracking error, penalizing the closed-loop system when the position controller interferes with tracking in the two inner loops. The fourth term penalizes excessive control actuation, as done in the attitude and velocity tunings.

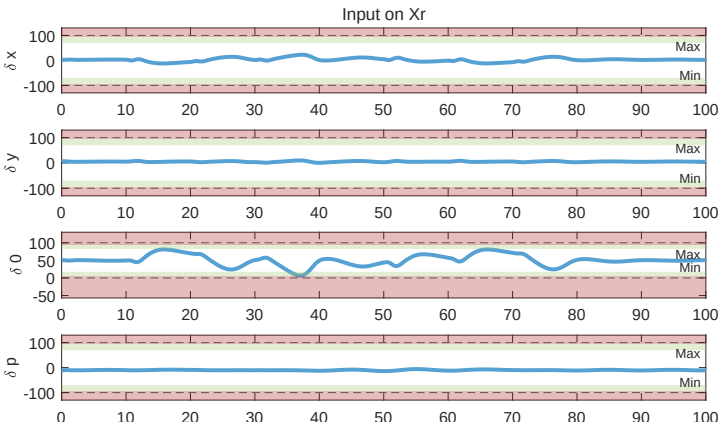
$$\begin{aligned} z = &\sum_{l=1}^{3} W_{Rp} \left(\sum_{t=1}^{T} (X_r^l(t) - X^l(t))^2 + \sum_{t=1}^{T} (Y_r^l(t) - Y^l(t))^2 + \sum_{t=1}^{T} (Z_r^l(t) - Z^l(t))^2 \right) + \dots \\ &\sum_{l=1}^{3} W_{Rv} \left(\sum_{t=1}^{T} (u_r^l(t) - u^l(t))^2 + \sum_{t=1}^{T} (v_r^l(t) - v^l(t))^2 + \sum_{t=1}^{T} (w_r^l(t) - w^l(t))^2 \right) + \dots \\ &\sum_{l=1}^{3} W_{Ra} \left(\sum_{t=1}^{T} (\Delta q_0^l(t))^2 + \sum_{t=1}^{T} (\Delta q_i^l(t)(t))^2 + \sum_{t=1}^{T} (\Delta q_j^l(t))^2 + \sum_{t=1}^{T} (\Delta q_k^l(t))^2 \right) + \dots \\ &\sum_{l=1}^{3} \left(\sum_{t=1}^{T} \left(W_x \Delta\delta x^l(t)^2 + W_y \Delta\delta y^l(t)^2 + W_p \Delta\delta p^l(t)^2 + W_0 \Delta\delta_0^l(t)^2 \right) \right) \end{aligned} \tag{35}$$

Results of the tuning procedure are shown in Figures 11 to 13; for brevity, only the position response and the control activation are presented. Predictably, there is a noticeable cross-activation on the vertical Z axis whenever commands are given on X and Y, with altitude deviations of up to 15m from trim (seen in Figure 11a): this is consistent with the velocity cross-activation noticed in the tuning of the velocity controller and, analysing the velocity response, it is consistent with delays in the velocity tracking discussed in Section IV.B.3.

Nonetheless, the on-axis performance of the control system showed great tracking effectiveness and the attitude controller remained highly responsive during the designed tracking tasks. In none of the tunings control saturation was achieved, although the high cross-activation of the vertical position caused the emergence of collective deflections to compensate for the system's natural couplings. In all cases, the on-axis input commands match the expected trends, with the especially noticeable rise-and-descent pattern in Figure 13b matching the commanded altitude variations. The pattern in the use of pedal for the second simulation (Figure 12b) is a consequence of the fact that the manoeuvre was generated as a side-step, with no heading change included in the system. Similarly to the velocity controller performance, the inclusion of terms penalizing tracking errors in the velocity and position loops was instrumental in achieving good performance while maintaining controller reliability: these terms ensure that the PSO converges to solutions that avoid overwhelming the capabilities of the inner loops, making the system more coherent and capable of tracking diverse reference trajectories.

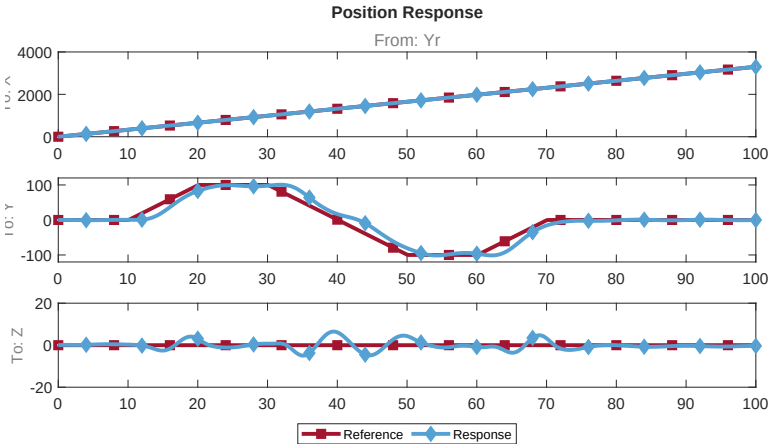


(a) First simulation - position signals

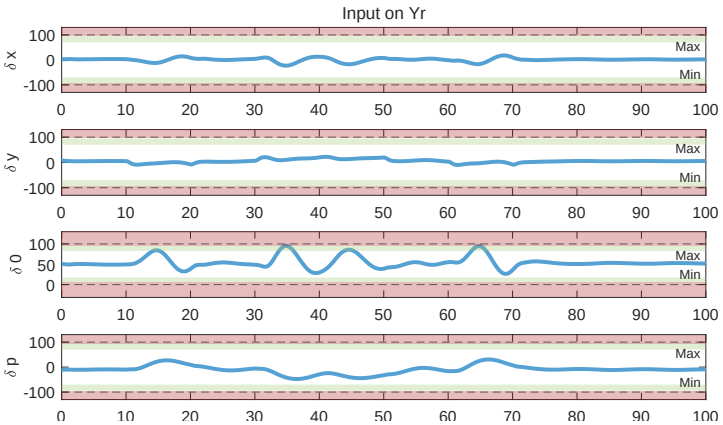


(b) First simulation - control signals

Fig. 11 First simulation: velocity variation along the global X axis

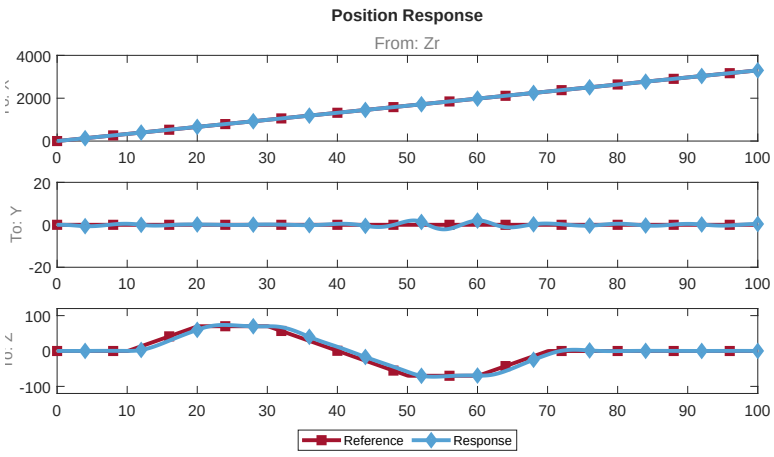


(a) Second simulation - position signals

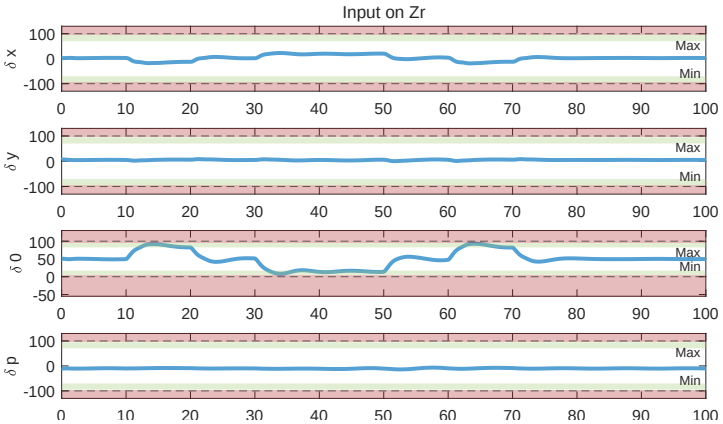


(b) Second simulation - control signals

Fig. 12 Second simulation: velocity variation along the global Y axis



(a) Third simulation - position signals



(b) Third simulation - control signals

Fig. 13 Third simulation: velocity variation along the global Z axis

V. Autonomous Flight: Trajectory Generation and Autopilot

Here the implementation of an autopilot to perform autonomous flight is discussed, integrating autonomous flight with the investigation of quaternion implementation. In subsection V.A, the procedure followed to generate manoeuvre references is discussed, and results are shown for the slalom and pop-up manoeuvres. In subsection V.B, a simple autopilot is implemented as a Finite State Machine (FSM).

A. Offline Manoeuvre Synthesis

The procedure followed to define offline manoeuvres to evaluate the system's performance is presented. The Mission Task Elements (MTEs) selected are a slalom and a pop-up, as they allow to evaluate the ability of the system to follow precise position changes in the horizontal and the vertical directions. While results will be shown for both the slalom and the pop-up, for brevity here the procedure used to identify reference signals is applied only to the slalom manoeuvre. The procedure used can be divided in two subsequent steps: first, the position, velocity, and acceleration signals are identified for the selected task; then, the acceleration signals are used to determine the desired attitude reference.

The identification of position and velocity signals is achieved by solving appropriately defined systems of Ordinary Differential Equations (ODEs). For this simulation, the slalom MTE is divided in three manoeuvre segment: a 10s forward flight entrance, a central slalom segment, and a forward flight exit of 10s, all performed at constant trim velocity $\|\mathbf{V}\| = 33\text{m/s}$. Following the recommendations identified in the Aeronautical Design Standard 33 (ADS-33) [36], the central segment consists of two turns on each side of the manoeuvre's centreline with a lateral deviation of at least 50ft separated by a horizontal distance of 500ft. For ease of implementation, the central segment was defined as a sinusoidal path with shape $Y^\star = A\sin\left(B(X^\star - L)\right)$, where $A = 25\text{m}$ (82ft), $B = \pi/500\text{ft}^{-1} = \pi/152.4\text{m}^{-1}$, and $L = 330\text{m}$ (the distance covered in 10s at 33m/s). From these, the total velocity can be expressed as in Equation 36.

$$\|\mathbf{V}\| = \sqrt{\dot{X}^2 + \dot{Y}^2} = \sqrt{\dot{X}^2 + (AB\cos(B(X-L))\dot{X})^2} \tag{36}$$

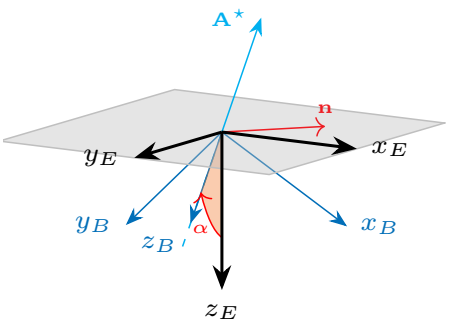
Two systems of ODEs can then be identified, as shown in Equation 37: on the left, the ODE describing the forward flight segment for the desired position signals (indicated by the superscript *) are shown; on the right, the equations for the central slalom segment are presented. In both cases, the desired arc length $s^\star$ is also included, as it will be used by the autopilot in Section V.B. Solving the systems with adequate initial conditions allows to determine the position and velocity signals required to perform the manoeuvre.

$$\begin{cases} \dot{X}^\star = \|\mathbf{V}\| \\ \dot{Y}^\star = 0 \\ \dot{Z}^\star = 0 \\ \dot{s}^\star = \|\mathbf{V}\| \end{cases} \qquad \begin{cases} \dot{X}^\star = \|\mathbf{V}\| / \sqrt{1 + (AB\cos(B(X^\star - L)))^2} \\ \dot{Y}^\star = AB\cos\left(B(X^\star - L)\right)\|\mathbf{V}\| / \sqrt{1 + (AB\cos(B(X^\star - L)))^2} \\ \dot{Z}^\star = 0 \\ \dot{s}^\star = \|\mathbf{V}\| \end{cases} \tag{37}$$

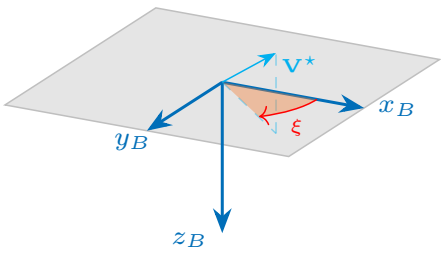
The desired attitude is determined assuming the main rotor's thrust is aligned with the helicopter's z-axis [20, 37, 38]. This hypothesis is supported by two assumptions: the magnitude of the H-forces is much smaller than the magnitude of the main rotor thrust, and the Tip Path Plane (TPP) is inclined by a small angle. These assumptions are only used to produce feed-forward attitude signals coherent with the manoeuvre: discrepancies with the desired model behaviour are corrected by the velocity controller. To identify the desired attitude, first the body-z axis is aligned with the desired acceleration, and then a rotation around z is performed to correct for heading.

The first step of the correction is shown in Figure 14a: starting from the NED frame ($\mathcal{E}$) a rotation is defined to align its z axis with the total desired acceleration $\mathbf{A}^\star$ (obtained differentiating Equation 37 and adding the gravitational acceleration $g = 9.81\text{m/s}^2$ along the global z direction). Quaternion $\hat{\mathbf{q}}^\star_{E2B}$ defines this rotation: it is defined by identifying the Euler angle α and the axis $\mathbf{n}$ shown in the figure, where $\mathbf{n}$ is the vector perpendicular to $\mathbf{A}^\star$ and $\mathbf{z}_E$ and α is the angle enclosed by $-\mathbf{A}^\star$ and $\mathbf{z}_E$. The second step is a rotation around the z-axis is performed to correct for heading. In this paper heading is corrected to align the helicopter's nose with the desired trajectory. This rotation is described by quaternion $\mathbf{q}_\xi$, where $\mathbf{v} = [0, 0, 1]^T$ is the Euler axis and ξ is the Euler angle, identified as shown in Figure 14b by projecting the desired velocity $\mathbf{V}^\star$ on the body x-y plane identified in the first step. The desired attitude is found with Equation 38, which uses Equation 1 to define the quaternions and concatenates them via quaternion multiplication Equation 2.

$$\mathbf{q}^\star_{E2B} = \hat{\mathbf{q}}^\star_{E2B}\mathbf{q}^\star_\xi = [\cos(\alpha/2), \sin(\alpha/2)\mathbf{n}]^T\,[\cos(\xi/2), \sin(\xi/2)\mathbf{v}]^T \tag{38}$$



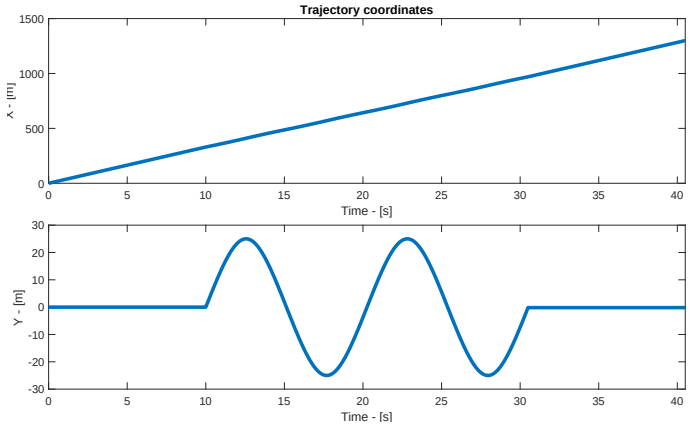
(a) Representation of α and n to align the body z-axis to the acceleration $\mathbf{A}^\star$



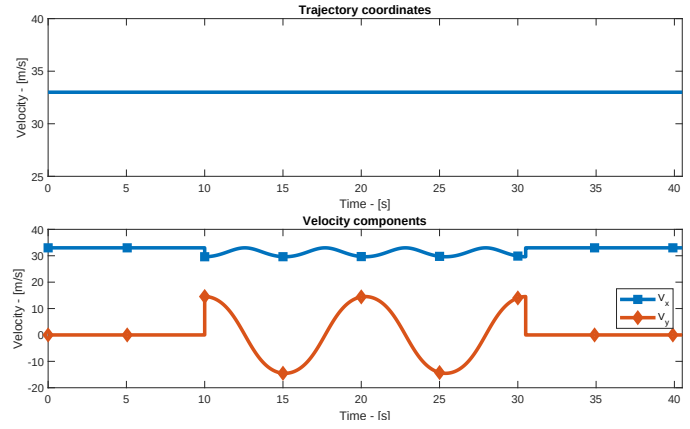
(b) Representation of ξ to align the body x-axis to the velocity projection

The second MTE studied is the pop-up, a longitudinal terrain avoidance manoeuvre described in the ADS-33 [36] and the Utility Tactical Transport Aircraft System (UTTAS) [39]. The helicopter is required to achieve a 25m altitude gain over a horizontal distance between 250m $\leq s_1 \leq$ 350m [40]. To emphasise vertical manoeuvrability, the manoeuvre was defined to gain 25m in altitude over a horizontal distance of 250m while maintaining trim velocity. As the procedure is formally identical, only the results are shown here.

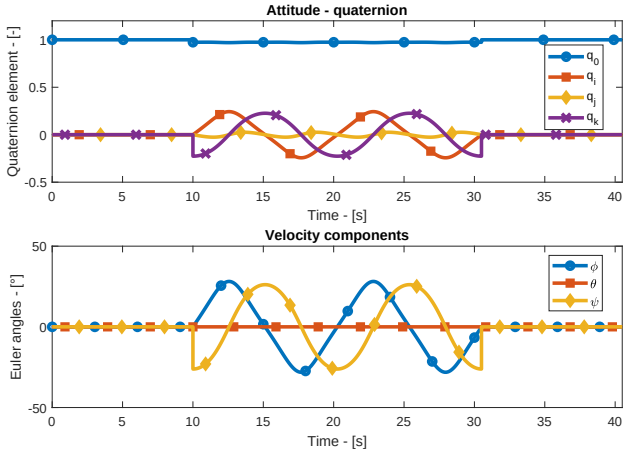
The references for both MTEs are shown in Figures 15 and 16: both the slalom and pop-up references match the expected behaviours and verify the trajectory generation approach adopted. In the slalom case (shown in Figure 15), a non-smooth reference was used to simplify trajectory generation: while this would generally be a limiting factor for accurate trajectory tracking, the presence of an online trajectory reconstruction autopilot module (presented in V.B) mitigates the impact of discontinuities in velocity and attitude profiles, improving trajectory tracking. For the pop-up manoeuvre, a smooth polynomial segment was employed, as seen in Figure 16.



(a) Slalom trajectory components

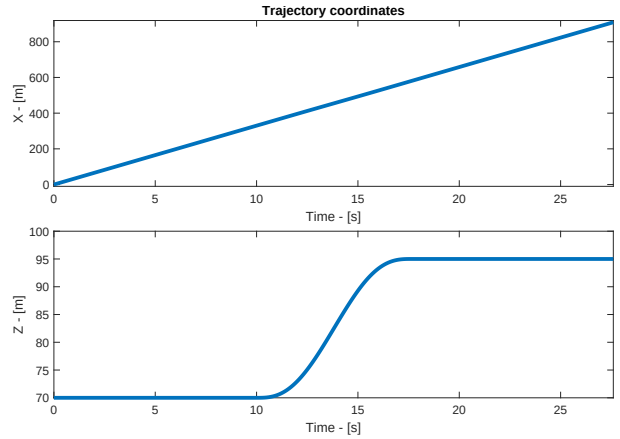


(b) Slalom velocity

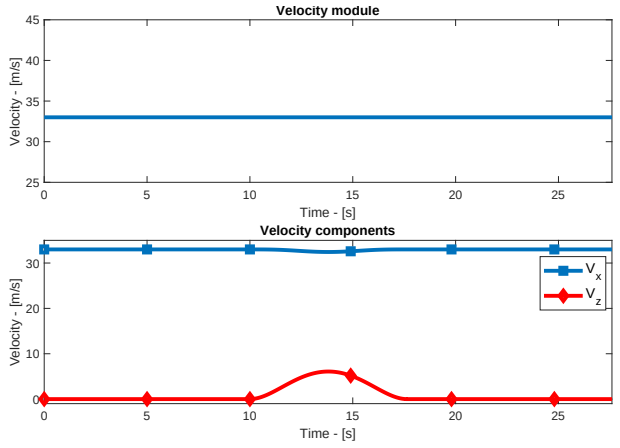


(c) Slalom attitude

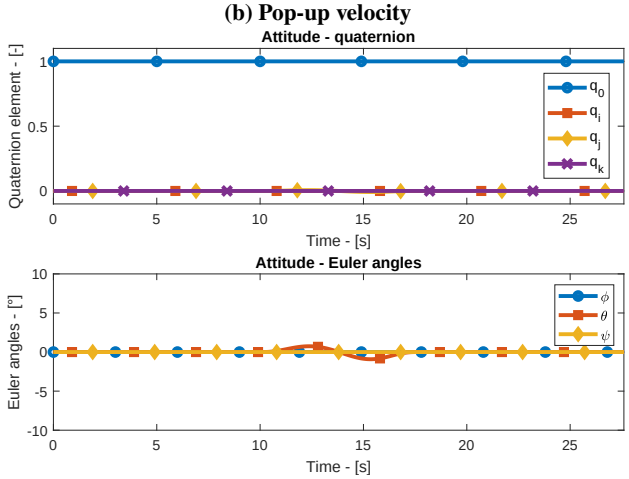
Fig. 15 Identified reference signals for the slalom manoeuvre



(a) Pop-up trajectory components



(b) Pop-up velocity



(c) Pop-up attitude

Fig. 16 Identified reference signals for the slalom manoeuvre

B. Autopilot Implementation

Here, an autopilot module is implemented using the Finite State Machine (FSM) formulation. The main purpose of the autopilot is to facilitate waypoint-based navigation, generating reference signals based on the helicopter's position on the desired trajectory. This facilitates trajectory tracking, as continuous-time references may become inadequate in the presence of system delays or external disturbances. To discuss the autopilot, first the trajectory reconstruction is discussed, and then the FSM implementation is presented.

1. Implementation of the Trajectory Interpolation

To enable efficient navigation, the reference signals calculated in Section V.A are sampled at specific waypoints and trajectory references are then reconstructed via interpolation. By using waypoint navigation, smoother trajectories can be reconstructed, overall improving the performance of the system while also allowing for a more flexible and efficient path-definition structure [41, 42]. To allow for a sufficiently accurate reconstruction with limited computational resources, waypoints were sampled at regular $\Delta T = 1$s intervals.

To increase system's responsiveness, the control system uses feed-forward signals in position $\mathbf{X}^\star$, velocity $\mathbf{V}^\star$, and attitude $\mathbf{q}^\star_{E2B}$. For quaternion attitude signals, the SLERP algorithm presented in Equation 8 is used to efficiently and smoothly interpolate attitude references without incurring in singularities or erratic reconstruction. For position and velocity values, interpolation is instead carried out using a cubic-spline interpolation on each of the reference vector signals entries [9]. Consider the trajectory to sampled in N waypoints with the helicopter in the trajectory segment delimitated by waypoints k and $k+1$ (a visual representation is given in **??**). Let $l \in [0, 1]$ be a running variable, such that: if $l = 0$, the helicopter is exactly at waypoint k; if $l = 1$, the helicopter is at waypoint $k+1$. Let h_k indicate the difference between the arc lengths at waypoints $k+1$ and k, such that $h_k = s_{k+1} - s_k$. The cubic interpolator connecting waypoint k and $k+1$ will have the shape shown in Equation 39, with $[a_k, b_k, c_k, d_k]$ unknown.

$$y_k(l) = d_k h_k^3 l^3 + c_k h_k^2 l^2 + b_k h_k l + a_k \quad \text{with} \quad 0 \le l \le 1 \tag{39}$$

The unknown parameters are computed by fitting Equation 39 to each of the individual position and velocity signals over five trajectory segments: if the helicopter is position at waypoint k, the waypoints selected to perform the interpolation are contained between $k-2$ and $k+3$. At each command loop, the algorithm computes the unknown parameters for all five available trajectory waypoints segments, and then isolates the central one within k and $k+1$ where the helicopter is positioned. In the cases where the helicopter is located in the first two waypoints $k = \{1, 2\}$ and when the helicopter has reached the final three waypoints $k = \{N-2, N-1, N\}$, the algorithm always performs calculations so that five trajectory segment are considered and then isolates the one enclosing the current helicopter's position. With the unknown parameters identified, knowing l the reference associated with the current helicopter position is determined.

2. Autopilot as a Finite State Machine

A Finite State Machine (FSM) [43] is a discrete-event system model that enables autonomous systems to execute complex behaviours by transitioning between operational states. Each state corresponds to a specific system action, and transitions occur in response to changes in the environment or internal system variables. This structured decision-making approach allows for transparent and reliable implementation of autonomous logic. The FSM was chosen for its flexibility and modularity, allowing for the possible future expansion of the autopilot to enable more complex autonomous flight behaviours.

The functioning of the autopilot FSM is explained in the state diagram in Figure 17. The autopilot is initialized at the first available waypoint $k = 1$ and also receives the current position coordinates of the helicopter. These values are used to calculate the running variable l which is used in the waypoint interpolation process: if $l < 1$ and $k < N$ (i.e.: if the helicopter is still within the trajectory segment contained within k and $k+1$ and the helicopter has not reached the last waypoint $k = N$), the calculated l is used to interpolate the reference values; if $l \ge 1$, the helicopter has already reached the next waypoint and the waypoint position is updated, setting $k := k+1$ and the running variable is recalculated. Once the helicopter has reached the final waypoint $k = N$, the simulation is stopped.

To calculate the running variable l, the autopilot approximates the completion of the current trajectory segment via Equation 40, where $\mathbf{r}_{k,k+1} = \mathbf{r}_{E,k+1} - \mathbf{r}_{E,k}$ indicates the linear distance between waypoint $k+1$ and waypoint k, and $\mathbf{r}_{k,B} = \mathbf{r}_{E,B} - \mathbf{r}_{E,k}$ indicates the linear distance between the helicopter and waypoint k. The value is also increased by 0.05 so that the autopilot produces reference signals which are always slightly ahead of the helicopter, allowing for smoother navigation. With this equation, the autopilot essentially calculates the projection of the helicopter's relative position with respect to its previous waypoint on the distance between two successive waypoints, thus allowing for a simple interpretation of the helicopter's current position within the trajectory.

$$l = \frac{\mathbf{r}_{k,k+1} \cdot \mathbf{r}_{k,B}}{\|\mathbf{r}_{k,k+1}\|^2} + 0.05 \tag{40}$$

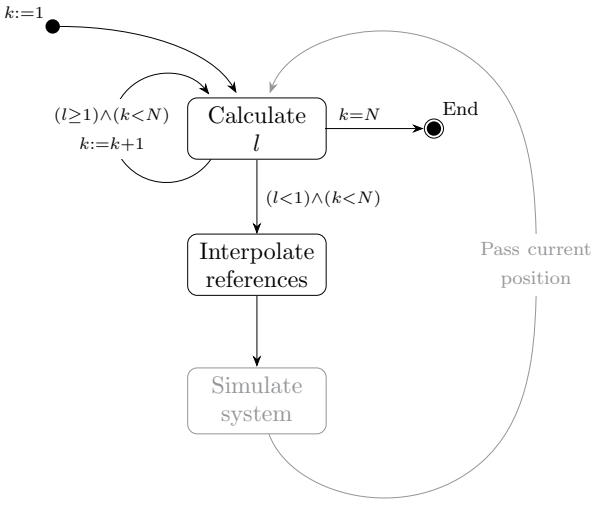


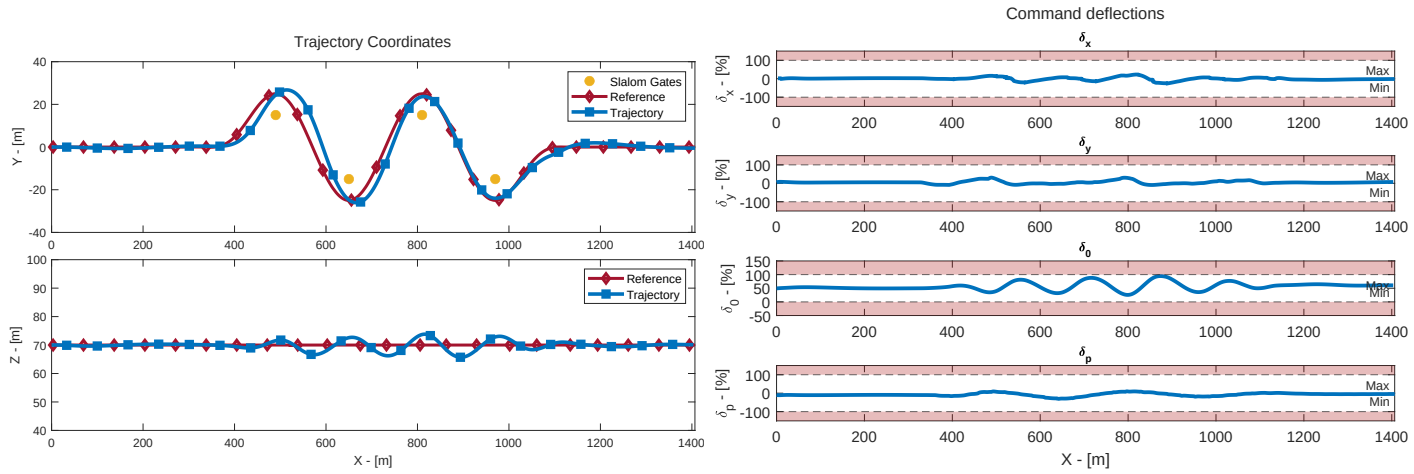
Fig. 17 State diagram of the autopilot FSM

VI. Autonomous Flight Simulations

To evaluate the controller's performance, the slalom and pop-up manoeuvre were simulated in MATLAB's Simulink environment. Simulations were run with a fixed time resolution of 1ms and using the RK4 solver. These simulation parameters were chosen as they align with the computational speed of sensors and on-board computers used in the context of aircraft control [44, 45].

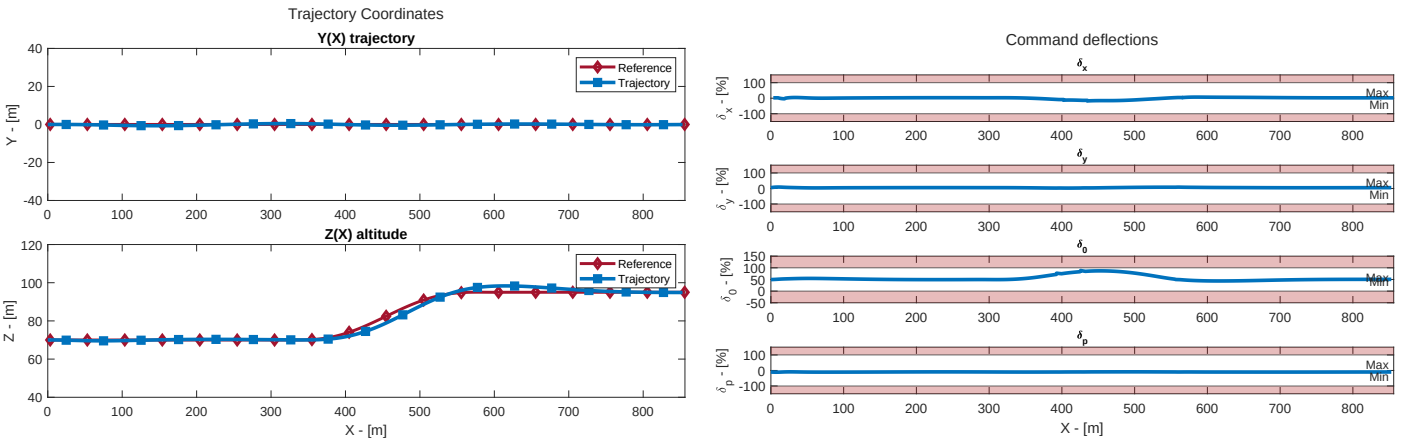
Results of the slalom simulation are collected in Figure 18: the results are positive, with good tracking performance across the position channels. The cross-activation of the vertical position in response to changes in the lateral position is also noticeable in this case, but, thanks to the feed-forward action of the velocity and attitude commands, these deviations are contained within ±7m from the trim altitude. Lateral position tracking satisfies the required minimal 15m deviation from the centerline [36], represented in Figure 18a by the addition of the slalom gates. Notably, thanks to the autopilot's ability to progressively generate position reference values, the lateral tracking performance improves over successive turns, with a maximum deviation from the desired position of 5m. Velocity tracking still shows the delays seen in the controller tuning (Figure 9): these delays are strictly linked to the position tracking performance at the entrance of the slalom, and further investigations of the central control loop performance would benefit the tracking performance. Attitude tracking remains highly effective, and commands never saturate, in spite of the aggressiveness of the manoeuvre.

Results of the pop-up manoeuvre are presented in Figure 19. The vertical position tracking is effective in following the reference trajectory with no steady-state error: when climbing, the aircraft shows a slightly delayed response causing it to slightly lag behind the reference during the manoeuvre, before overshooting the desired final attitude when exiting the climbing phase of the MTE. Lateral deviations are minimal, and heading is kept constant throughout the task. Control actuation matches the expected response, with only collective and cyclic being activated respectively to gain altitude and to provide minor corrections to heading and horizontal velocity. No command saturation is experienced, further suggesting the validity of the tuning approach used.



(a) Trajectory **(b) Control actuation**

Fig. 18 Results of the slalom manoeuvre simulation



(a) Trajectory **(b) Control actuation**

Fig. 19 Results of the Pop-up manoeuvre simulation

VII. Conclusion

Within this paper, the implementation of quaternion control in the context of modelling and autonomous control of agile helicopter flight has been examined, using the MBB Bo105 helicopter as a practical case study. In section II, a comprehensive overview of the fundamental properties of quaternions was provided. Owing to the linearity of their operations and their ability to smoothly interpolate and represent complex attitude changes in 3D space without singularities, quaternions are shown to be a powerful tool for the modelling and control of highly manoeuvrable vehicles. In section III, a procedure for converting existing linearized models from Euler angle formulations to quaternion-based attitude parametrizations was presented. As this approach is rooted in analytical derivations, it allows for the direct transformation of established models, greatly facilitating the study and integration of quaternion-based controllers while minimizing modelling effort and avoiding the need for additional linearizations. With the model reformulated using quaternions, the use of quaternion representations in control system design was discussed in section IV. The controller was structured using a nested-loop architecture, with the innermost loop implementing a novel LQI controller in quaternion space for attitude regulation, and the intermediate and outer loops employing PI and P controllers, respectively, for velocity and position tracking. The controller tuning was carried out using the PSO algorithm to optimize each loop individually; the implementation of PSO was detailed for each control loop, alongside the definition of the cost function and the most significant results obtained.

With the system extended to support trajectory tracking, autonomous flight was addressed in section V. Section V.A introduced a method for generating reference position, velocity, and attitude signals suitable for helicopters, providing a straightforward framework for creating coherent manoeuvre definitions. This method was also applied to define two representative manoeuvres—slalom and pop-up—for evaluating the system's performance. In Section V.B, a simple yet effective autopilot was implemented using an FSM architecture, enabling modular and flexible mission execution. The performance of the system was subsequently evaluated in section VI through simulation of the slalom and pop-up tracking tasks. The tracking results support the validity and effectiveness of the methodology employed across all stages of the study, with the helicopter successfully tracking both trajectories without steady-state errors and maintaining tracking errors below 5m during the most extreme phases of the slalom and below 7m at the apex of the pop-up manoeuvre.

As this work outlines a comprehensive methodology for the use of quaternions in modelling and autonomous flight control, several avenues for future research can be identified. From a modelling perspective, the conversion procedure could be applied to other systems and linearization schemes. Comparing linearizations of native nonlinear quaternion models to those obtained by converting Euler-based models would help assess the impact and efficiency of the proposed method. From a control standpoint, improving the performance of the velocity control loop could yield enhanced dynamic response, while extending the controller to accommodate multiple linearization points would allow for smoother and more robust navigation. The findings presented here may also encourage broader use of PSO in control applications, supporting the design of efficient, multimodal optimization-based controllers. Finally, in the context of autonomous flight, the FSM autopilot could be expanded to include online trajectory planning, equipping the system with the capability for dynamic environmental interaction and obstacle avoidance.

References

[1] Mishra, S., and Palanisamy, P., "Autonomous Advanced Aerial Mobility—An End-to-End Autonomy Framework for UAVs and Beyond," *IEEE Access*, Vol. 11, 2023, pp. 136318–136349.

[2] Anderson, E., Fannin, T., and Nelson, B., "Levels of aviation autonomy," *2018 IEEE/AIAA 37th Digital Avionics Systems Conference*, IEEE, 2018.

[3] Tighe, J., "Autonomy and future mobility in aerospace," , Feb. 2024.

[4] Gołąbek, M., Welcer, M., Szczepański, C., Krawczyk, M., Zajdel, A., and Borodacz, K., "Quaternion attitude control system of highly maneuverable aircraft," *Electronics (Basel)*, Vol. 11, No. 22, 2022, p. 3775.

[5] Jia, Y.-B., "Quaternions and Rotations *," 2015.

[6] Zhu, T., Li, A., Li, K., and Qin, F., "The quaternion based error model based on SE(3) of the INS," *IEEE Sens. J.*, Vol. 22, No. 13, 2022, pp. 13067–13077.

[7] Tarek, M., Zahran, S., Radi, A., Nafea, S. F., Oda, E. S., and Hafez, A. E.-D. S., "Performance analysis of quaternion vs Euler-based approaches in reduced INS/GNSS fusion techniques," *2023 International Telecommunications Conference (ITC-Egypt)*, IEEE, 2023, pp. 530–537.

[8] Kritskiy, D., Alexander, K., Koba, S., and Druzhinin, E., "Increasing the reliability of drones due to the use of quaternions in motion," *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, IEEE, 2018.

[9] Gerig, M. B., "Modeling, guidance, and control of aerobatic maneuvers of an autonomous helicopter," Ph.D. thesis, 2008.

[10] Duan, X., Yue, C., Liu, H., Guo, H., and Zhang, F., "Attitude Tracking Control of Small-Scale Unmanned Helicopters Using Quaternion-Based Adaptive Dynamic Surface Control," *IEEE Access*, Vol. 9, 2021, pp. 10153–10165.

[11] Suzuki, S., and Nonami, K., "Quaternion-based navigation and control for small unmanned helicopter," *IFAC Proc. Vol.*, Vol. 43, No. 15, 2010, pp. 37–42.

[12] Elkaim, G. H., "System Identification for Precision Control of a WingSailed GPS-Guided Catamaran," 2002.

[13] Kuipers, J. B., "Quaternions and Rotation Sequences," 1998.

[14] Zanetti, R., "Rotations, Transformations, Left Quaternions, Right Quaternions?" *Journal of the Astronautical Sciences*, 2019.

[15] Bacon, B. J., "Quaternion-Based Control Architecture for Determining Controllability/Maneuverability Limits," 2012.

[16] Dong, M., and Yao, G., "Research on attitude interpolation and tracking control based on improved orientation vector SLERP method," *Robotica*, Vol. 38, No. 4, 2020, pp. 719–731.

[17] Jonkman, B., "Interpolation of DCMs," 2020.

[18] Bernardes, E., and Viollet, S., "Quaternion to Euler angles conversion: A direct, general and computationally efficient method," *PLOS ONE*, Vol. 17, No. 11, 2022.

[19] Weiland, E. F., "Development and Test of the BO 105 Rigid Rotor Helicopter," *Journal of the American Helicopter Society*, Vol. 14, No. 1, 1969, p. 22–37.

[20] Padfield, G., *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*, AIAA education series, American Institute of Aeronautics and Astronautics, 1996.

[21] Moll, N., "To a Bo105 pilot: this thing's pretty," *Flying magazine*, Vol. 118, No. 11, 1991, pp. 96–105.

[22] Joachim Götz, *Bo105: Configuration Data*, DLR Institut für Flugsystemtechnik, Braunschweig, Germany, 2005.

[23] Joachim Götz, *Bo105: State Space Model*, DLR Institut für Flugsystemtechnik, Braunschweig, Germany, 2005.

[24] Hespanha, J. P., *Linear systems theory*, 2nd ed., Princeton University Press, Princeton, NJ, 2018.

[25] Young, P. C., and Willems, J. C., "An approach to the linear multivariable servomechanism problem," *Int. J. Control*, Vol. 15, No. 5, 1972.

[26] Kashyap, M., and Lessard, L., "Guaranteed Stability Margins for Decentralized Linear Quadratic Regulators," *IEEE control systems*, Vol. 7, 2023.

[27] Abdou, L., "Quanser's 3-DOF Helicopter control using LQR-I and MPC-based LQR controllers," *International Conference on Computer Aided Design*, 2023, p. 1–6.

[28] Shahzad, A., and Altalbe, A., "Computing of LQR Technique for Nonlinear System Using Local Approximation," *Computer Systems: Science #38; Engineering*, Vol. 46, No. 1, 2023, p. 853–871.

[29] Myala, J., Borra, J. K., and Patel, V. V., "Partially Structured LQR design for Lateral-Directional Flight Control Law," 2019.

[30] Anderson, B. D. O., and Moore, J. B., *Linear Optimal Control*, Prentice-Hall networks series, Prentice Hall, Old Tappan, NJ, 1971.

[31] Kwakernaak, H., Sivan, R., and Tyreus, B. N. D., "Linear optimal control systems," *J. Dyn. Syst. Meas. Control*, Vol. 96, No. 3, 1974, pp. 373–374.

[32] Poli, R., Kennedy, J., and Blackwell, T., "Particle swarm optimization," *Swarm Intell.*, Vol. 1, No. 1, 2007, pp. 33–57.

[33] Joseph, S. B., Dada, E. G., Abidemi, A., Oyewola, D. O., and Khammas, B. M., "Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems," *Heliyon*, Vol. 8, No. 5, 2022, p. e09399.

[34] Yang, X., *Nature-Inspired Optimization Algorithms*, Academic Press, 2020.

[35] Franklin, G. F., Powell, J. D., and Emami-Naeini, A., *Feedback control of dynamic systems*, 8th ed., Pearson, Upper Saddle River, NJ, 2018.

[36] Blanken, C. L., Tischler, M. B., Lusardi, J. A., Berger, T., Ivler, C. M., and Lehmann, R., "PROPOSED REVISIONS TO AERONAUTICAL DESIGN STANDARD – 33E (ADS-33E PRF) TOWARD ADS-33F-PRF," Tech. Rep. SR-FCDD-AMV-19-01, U.S. Army, Sep. 2019.

[37] Prouty, R., *Helicopter Performance, Stability, and Control*, Krieger, 2005.

[38] Cao, Y., Zhang, G., and Su, Y., "Mathematical modeling of helicopter aerobatic maneuvers," *Aircr. Eng.*, Vol. 76, No. 2, 2004, pp. 170–178.

[39] Yamakawa, G. M., Broadhurst, D. G., and Smith, J. R., "Utility Tactical Transport Aircraft System (UTTAS) Maneuver Criteria," Tech. Rep. AD0902767, ARMY AVIATION SYSTEMS TEST ACTIVITY EDWARDS AFB CA, Nov. 1972.

[40] Thomson, D., "An Analytical Method of Quantifying Helicopter Agility," 1986.

[41] He, S., Shin, H.-S., and Tsourdos, A., "Optimal Guidance for Integrated Waypoint Following and Obstacle Avoidance," Vol. 2019, 2019, p. 325–334.

[42] Zhou, Z., Wang, G., Sun, J., Wang, J., and Chen, J., "Efficient and Robust Time-Optimal Trajectory Planning and Control for Agile Quadrotor Flight," *IEEE robotics and automation letters*, Vol. 8, No. 12, 2023, p. 7913–7920.

[43] Zhou, Q., Shen, Z., Yong, B., Zhao, R., and Zhi, P., *Theories and practices of self-driving vehicles*, Elsevier - Health Sciences Division, Philadelphia, PA, 2022.

[44] Ren, B., Du, S., Cui, Z., Xu, Y., and Zhao, Q., "High-precision trajectory tracking control of helicopter based on ant colony optimization-slime mould algorithm," *Chinese Journal of Aeronautics*, Vol. 38, 2024.

[45] Coppa, U., Guarnieri, A., Pirotti, F., and Vettore, A., "Accuracy enhancement of unmanned helicopter positioning with low-cost system," *Applied Geomatics*, Vol. 1, 2009.

Part II

Preliminary Analysis

2

Literature Review

This work has the fundamental objective of exploring the application of quaternions to the study of helicopter motion, with particular attention to the development of a control system for the aircraft and the achievement of autonomous flight. Here, a literature review of these topics will be provided, producing a survey of current relevant research to both identify current trends in the fields of control systems and aerospace engineering and contextualizing the current study within the broader academic conversation.

To provide an orderly overview of the current research analysed, this chapter will be further divided in three sections, each focusing on a different aspect of the work. To start, Section 2.1 will provide a brief overview of current research on the applications of quaternions in the context of modelling and simulation of aircraft flight, pointing out their relevance and intrinsic advantages over other attitude formulations and highlighting the small amount of available literature on their application to full-scale systems. Further, in Section 2.2 current developments in the fields of controller design will be discussed, highlighting the approaches typically used when developing agile controllers and their characteristics. Lastly, in Section 2.3 an overview of prominent methodologies used to enable autonomous flight are presented and discussed.

2.1. Use of quaternions in modelling and control system design

As this work ultimately aims at investigating the implementation of quaternions in the development of a flight control system and at addressing their usefulness in aircraft modeling, the first central aspect the preliminary research step focused on was the investigation of quaternion properties and of their implementations in control systems, building upon current results and identifying research gaps that highlight the relevancy of this work.

In the realm of applied mathematics and engineering, quaternions are recognized as a sophisticated tool for representing rotations in three-dimensional space. First introduced by Hamilton in the 19th century, quaternions constitute a four-dimensional hypercomplex number system that elegantly represents rotations in three-dimensional space. Their formulation - comprising one real and three imaginary components - allows for compact, singularity-free encoding of orientation (thereby avoiding issues such as gimbal lock encountered with Euler angle representations) and for the generation of smooth omnidirectional orientation changes with the implementation of simple and powerful interpolation algorithms, such as Spherical Linear Interpolation (SLERP).

In practical terms, quaternions are applied across several cutting-edge fields. They enhance rotational dynamics modelling in computer graphics [1], smoothly reconstruct attitude change sequences [2, 3], and they provide critical orientation solutions in robotics for precise control and movement [4, 5]. Their role is equally significant in simulation and virtual reality environments, where they contribute to accurate spatial representations. Conclusively, quaternions have recently seen a rising interest in the fields of robotics and - most notably in the context of this thesis - aerospace engineering, where their robust and efficient attitude representation capacity has enabled the synthesis of complex and fluid motion in space.

While quaternions have been a cornerstone in the space engineering sector for many years - particularly in the context of spacecraft attitude control, owing to their ability to represent rotations without singularities -

their study in the context of aeronautics and aircraft control has not seen great interest over time, favouring the Euler angles parametrization. Inverting this trend, recent developments in the field - especially the surge in interest surrounding Micro Aerial Vehicles (MAVs) and quadcopter drones - have amplified the focus on quaternion-based methods, as their ability to represent smooth rotations in all directions allowed to fully leverage the high manoeuvrability of these systems.

With the rising interest in quaternions for the purpose of attitude representation of highly manoeuverable vehicles, researchers have argued and investigated the feasibility of quaternion-based control systems and modelling of full scale vehicles. In 2022 Gołkabek et al. [6] proposed an investigation of quaternions to the control of an agile aircraft and within their paper introduced their implementation as ”an extension to research on spacecraft attitude control”, highlighting the novelty of the application of quaternions to the field of aeronautics. Within their paper they developed a quaternion controller for an aircraft and compared the results obtained with a more traditional Euler angle based controller: the results obtained showed the quaternion controller was capable of outperforming the more conventional design, showing promise in the investigation of such a control system. These results are corroborated by other studies published in recent years - such as [7, 8] - which all go to show the promising nature of the implementation of quaternions on air vehicles and UAVs for the purpose of agile manoeuvering as they allow for higher attitude representation precision.

With the increasing interest in agile and unmanned navigation, the possibility of exploring the capabilities of helicopters becomes highly desirable, as their naturally high mobility around all axes of movement paired with their hovering flight and vertical take-off and landing abilities would greatly benefit from the implementation of novel control techniques. As of the development of this thesis, there is only a small amount of information on the applications of quaternions to the control of full-scale helicopters, with only few papers being focused on the application to scale models [9, 10, 11] at most. Additionally, implementation of quaternions in autopilots and autonomous flight controllers have been linked to improved efficiency, reliability, and performance [12], further increasing the relevance of quaternions in current research trends. To address the rising interest in the implementation of quaternions in aircraft control and modelling, and to meaningfully contribute to research topics of current interest and relevance, this work will investigate the use of quaternions in the description of the attitude and the development of a full flight control system for the purpose of autonomous helicopter navigation.

To allow for a proper investigation of quaternions and their applications to modelling and control system design, first an analytical description of their most relevant properties will be carried out within the thesis in Chapter 5, presenting the fundamental algebraic equations useful in the context of attitude representation.

2.2. Controller development for agile manoeuvering

As discussed in Section 2.1, the diffusion of small-scale drones and air vehicles has in recent years sparked great interest in the study and applications of agile controllers capable of performing manoeuvers with significant attitude changes with reduced control effort and high performance. As quaternion controllers have been praised for their flexibility and precision in attitude representation, they are a natural fit for agile manoeuvering and as such have found relevance in the current academic conversation. Here, the development of appropriate flight control systems will be discussed, providing an overview of the approaches considered in recent publications.

Agile maneuvering is the capability of an aerial vehicle to execute rapid and significant changes in attitude and trajectory, a performance attribute that is vital for operating in unpredictable and dynamic environments. This form of maneuvering is characterized by its high demand for control precision, rapid response, and minimal control effort, making it a subject of intense research. Its growing importance is evident across a broad spectrum of applications, from small unmanned aerial vehicles (MAVs) to full-scale aircraft and rotorcraft, where agile performance is increasingly a key requirement.

In response to the increasing relevance of agile and high-performing air vehicles, the rising interest in control techniques adequate for achieving this type of behaviour has motivated researchers to explore controller frameworks that could effectively augment aircraft systems to enable agile manoeuvre tracking. As the performance of a closed loop system are highly dependant on the controller’s capabilities, various

possible architectures have been analysed: here a brief review of the most prominent control architectures identified for agile manoeuvering is provided, highlighting their advantages and limitations in the context of flight control systems.

Given the large variations in attitude implied by the need to achieve agile manoeuvers, control systems developed for this purpose ought to be able to handle large variations in the system's states while maintaining robustness and reliability; with these challenges in mind, non-linear control methodologies show great promise in the obtainment of effective control behaviour.

The primary approach explored in nonlinear control is the Nonlinear Dynamic Inversion (NDI), an advanced control technique that seeks to counteract a system's inherent nonlinearities by effectively "inverting" its dynamics, thereby transforming the nonlinear control problem into one that is more tractable and akin to a linear system. While NDI control has a number of advantages and can effectively handle the task of controlling most non-linear systems, it is also subject to many limitations. To start, as NDI requires an internal model of the system to perform the inversion, the effectiveness of NDI depends upon the accuracy of the underlying dynamic model; any mismatch between the model and the real system can result in substantial performance degradation. Further, NDI is highly sensitive to external disturbances that are not accounted for in the system, which may lead to instability if not properly addressed with specific robustifying procedures [13]; this is especially noticeable in unconventional flight regimes where baseline model accuracy may falter due to complex nonlinear effects. Addressing these issues, the original NDI approach has been modified to reduce its reliance on offline-identified system models, leading to the introduction of the Adaptive Nonlinear Dynamic Inversion (ANDI) and the Incremental Nonlinear Dynamic Inversion (INDI) frameworks, which leverage advancements in sensor accuracy and online model identification strategies to allow nonlinear control systems to achieve better overall performance even in less-explored regions of the flight envelope while limiting the impact of unmodeled dynamics.

Given the novelty of NDI-based control architectures, their applications and full capabilities are still being explored. NDI has found great success in the study and control of small-scale quadcopters and other MAVs, which leverage relatively simple dynamics and extremely high maneuverability, making nonlinear control techniques especially suited to this field of applications. Outside of small air vehicles, current notable applications of NDI are the Airbus A350 in the field of civil aviation and the Lockheed Martin F-35 in the military field: additionally, nonlinear control is also being considered for the design of TU Delft's Flying-V [14]. Regarding the applicability of NDI architectures to helicopter flight, no current commercial application was identified, although research papers suggest the possibility of applying such modelling techniques to rotorcraft systems, issues with accurate modelling of vehicle dynamics appear to remain a primary limiting factor [15, 16].

Another possible control technique highly investigated for the purpose of agile manoeuvering is Model Predictive Control (MPC) [17, 18], an advanced optimization-based control strategy that utilizes a mathematical model of a system to predict its future behavior over a specified time horizon and generates appropriate control actions by solving a user-defined optimization problem at each control step. Key advantages of MPC include its ability to handle Multiple-Input Multiple-Output (MIMO) systems - a complex task for most traditional control methods - and explicitly manage constraints, which makes it a robust choice for complex applications [19]. However, these benefits come with challenges such as significant computational demands for solving optimization problems in real-time, a strong dependence on the accuracy of the underlying system model, and the complexity involved in tuning parameters like prediction horizons and cost weights [20, 21, 22].

With improvements in small-scale hardware and the introduction of highly-optimized optimization algorithms, MPCs have gained significant interest for their intuitive implementation and flexibility. As of now, the primary area of application of this control system architecture in the field of aerospace engineering is found in MAVs [23] which - similarly to NDI - tend to have simpler model descriptions and are capable of performing fast and agile manoeuvres. On a larger scale MPC has been a topic of interest in the field of UAVs given its ability to account for multiple objectives online, making it capable of easily minimizing reference tracking error while managing commanded control actuation and minimizing energy consumption: implementation of MPC to control of fixed wing aircraft has been studied in a number of research papers and has proven to be a field with great potential due to its many advantages [24], but many recognize the

limitations of such control architectures due to limited robustness [25] and reliance on identified model accuracy [25, 24]. Regarding possible applications in the specific area of rotorcraft flight, researchers have pointed to many performance advantages such as improved tracking in constrained trajectory[26] and the ability to reduce the impact of external disturbances [27], but strong couplings and requirements on model accuracy pose significant challenges to the implementation of MPC [28].

The analysis of published papers and current research trends reveals a significant interest in advanced nonlinear control methods—such as MPC and NDI—for investigating agile maneuvers. Even so, these sophisticated techniques have found only limited application in larger air vehicles, particularly in helicopters; in many cases, classic linear control approaches continue to be the most widely used, primarily because they offer greater ease of implementation, robustness, and reliability in the face of the complex dynamics characteristic of full-scale and rotary-wing platforms. Linear controllers are well-established and have a proven track record, making their implementation a much more streamlined endeavour: consequently, while nonlinear control techniques offer theoretical advantages for agile maneuvers, their transition to operational use in larger air vehicles has been constrained by practical considerations, leaving linear control techniques as the predominant solution in these domains. Given the ubiquity of linear control systems and the aim of this work to expand upon the research on quaternion implementation to helicopter use, this work will primarily focus on linear control methodologies to enable agile flight.

Linear control strategies are varied and have been studied for a number of years now, to the point where they have found impactful applications in all fields of industrial engineering. Linear control techniques are highly regarded for their simplicity, ease of implementation, and the wealth of established theoretical tools available for analysis and design. Typically, these controllers are developed based on linearized models of the underlying system, which are only valid around a specific operating point (or trim condition): this reliance on linear approximations means that, while linear controllers perform well under nominal conditions, they are not naturally well suited to handling the strong nonlinearities and rapid, large-angle maneuvers characteristic of agile flight. To overcome these limitations, researchers have explored hybrid strategies — such as gain scheduling or integrating advanced sensor fusion and adaptive mechanisms — that extend the effective range of linear controllers into more dynamic environments.

The most common and well-studied controller architecture is the Proportional-Integral-Derivative (PID). PID control combines a proportional term that reacts to current errors, an integral term that compensates for accumulated past errors, and a derivative term that anticipates future errors based on the error's rate of change: its advantages lie in its simplicity and the ease with which it can be implemented across a range of systems, making it a staple in industrial and aerospace applications. Even with this very simple design, the implementation of a PID controller requires careful consideration: PIDs are notoriously hard to hand tune, especially with highly coupled MIMO systems (like helicopters) and, to avoid being destabilized by sensor noise which would cause rapid and unpredictable changes in the error signals, the derivative element is often omitted, leading to a more robust but slower implementation.

Addressing the limitations of PI controller design, a number of strategies have been identified to increase its application range, with gain scheduling being the most prominent solution. With gain scheduling PI controllers remain effective at multiple trim points, allowing for precise flight at multiple speeds and attitudes. Moreover, the simplicity of PI design still allows them to be implemented as very effective outer-loop control systems, finding applications in systems where the inner loop may be realized with either linear or non-linear techniques [29, 30].

Another effective linear control strategy is represented by Linear Quadratic (LQ) control, a family of optimal control architectures that have received significant interest due to their potential to produce robust and effective controllers for both SISO and MIMO systems, addressing the main challenges of classic PID control at the cost of increased complexity in their implementation. The most prominent example of linear quadratic control is the Linear Quadratic Regulator (LQR), a control framework that allows for the balancing of controller performance and robustness to generate a robust controller capable of controlling complex systems, so long as they are stabilizable and that they allow full-state feedback.

With the relevance of the LQR framework, a number of modifications to it have been presented over the years. One of the most prominent modifications presented is the Linear Quadratic Gaussian (LQG), which

removes the need for full-state feedback by pairing the optimal regulator with a Kalman filter for the purpose of state reconstruction, at the cost of reduced robustness - which is addressed by using Loop Transfer Recovery (LTR) techniques.

Conversely, while LQR-based designs are normally effective in producing controllers with a strong regulating action. To enhance tracking accuracy the Linear Quadratic Integral (LQI) architecture aims at improving this aspect with the introduction of an integral action to address steady-state errors. Being a natural extension of the LQR framework, the LQI is still subject to the same requirements for application as the LQR, namely: full-state feedback and stabilizability. In spite of these limitations, LQI applications have found prominent use in the aerospace field for their ability to control MIMO systems while also providing reliable tracking performance.

While LQ control has a long and proven track record in the control of industrial plants [31, 32], applications in the field of aeronautical engineering are of great relevance and interest in present years both at an academic and industrial level in the development of tracking and stabilizing controllers for both small and large scale air vehicles. MAVs leverage the robustness of optimal control techniques and advancements in sensor hardware to produce effective controllers [33], to the point of making use of LQ controllers even in case of complex multi-body designs [34]. On a larger scale, LQ control has been studied for implementation on aircraft [35, 36], helicopters [37] and other unconventional vehicles such as tiltrotors [38].

Another widely studied approach within linear control theory is the H-infinity ($\mathcal{H}_\infty$) control framework, which belongs to the broader class of robust control techniques and has recently become one of the most polarizing control strategies in research, although it has yet to find widespread use in industry [39]. Unlike LQ controllers, which optimize performance with respect to a quadratic cost functional, $\mathcal{H}_\infty$ control is fundamentally concerned with minimizing the worst-case gain from external disturbances to controlled outputs while ensuring reference commands are replicated by the output of the system: this is obtained by controlling the $\mathcal{H}_\infty$ norm of the closed-loop transfer function that maps exogenous inputs (such as disturbances, sensor noise, or reference trajectories) to regulated outputs. This approach allows the designer to explicitly account for model uncertainties and disturbances during the controller synthesis process, making it especially attractive for aerospace applications where high reliability and robustness are critical.

The mathematical foundation of $\mathcal{H}_\infty$ control relies on satisfying specific feasibility conditions that ensure the internal stability and performance of the resulting controller [39, 40]. This is obtained by adequately shaping relevant transfer functions of the closed-loop system (such as the sensitivity and co-sensitivity transfer functions) to limit the effects of exogenous inputs (such as noise and disturbances) while ensuring references are represented in the output without saturating the actuator's bandwidths [41]. In practical terms, $\mathcal{H}_\infty$ controllers tend to be more conservative than their LQ counterparts. Nonetheless, this conservative design philosophy can offer significant benefits in aerospace systems, where safety margins and guaranteed performance bounds are crucial. In spite of these advantages, its high implementation complexity, the requirement for accurate uncertainty modelling, and limited availability of mature design toolchains have contributed to its slower transition from theory to practice, now limiting its applications in the aerospace engineering field to MAVs and small-scale aircraft [42, 43].

Given the interest in the application of LQ control, its flexibility and ability to readily control MIMO systems, within this work the baseline architecture of the controller will be based around this technique. In particular - as the aim of this work is to investigate autonomous flight applications where good reference tracking is paramount - to improve the reference following performance of the controlled system the innermost attitude loop will implement the LQI controller framework, which introduces an integral action that can limit the impact of steady-state errors.

To further leverage the flexibility of linear controller design the inner loop will be augmented with appropriate PID-based outer loops, pairing the tracking and decoupling capabilities of the LQI framework with the simplicity and effectiveness of PID controllers. To limit the interference and impact of measurement noise, the P/PI architectures will be implemented to generate an appropriate control actions that can drive the system to follow a desired trajectory.

2.3. Autonomous flight

With current researches presented in both the implementation of quaternions in modelling and controller development and with typical methods of controller development discussed, the last fundamental area of interest of this thesis lies in the augmentation of a controlled system to enable autonomous flight. To provide a better understanding of the topic and to identify the most suitable solution to implement for this thesis, here the concept of autonomous navigation will be discussed and presented, exploring relevant literature and clearly identifying its fundamental elements.

Autonomous navigation has emerged as a pivotal area of research and development, driven by the increasing demand for vehicles that can operate independently in complex and dynamic environments [44, 45]. Its relevance spans numerous sectors—from industrial automation, to military applications—and has become especially critical in the realm of aerospace engineering, where the ability to navigate without human intervention can significantly enhance operational efficiency and safety.

Recent technological advancements have accelerated progress in autonomous navigation [46], leveraging innovations in sensor technologies, machine learning, and high-speed computation to overcome previous limitations. Breakthroughs in fields such as computer vision and advancements in artificial intelligence, lidar and radar sensing, and real-time data fusion have paved the way for more accurate perception and decision-making capabilities in autonomous systems. While these advancements have found widespread application in domains such as robotics and automotive engineering, their integration into aerospace platforms is particularly noteworthy. In full-scale air vehicles—ranging from fixed-wing aircraft to helicopters—the adoption of autonomous navigation technologies is poised to revolutionize flight control, with emerging Urban Air Mobility (UAM) applications further driving the need for enhanced reliability and security in autonomous flight systems.

At its core, autonomous navigation encompasses a suite of interrelated subtasks that collectively enable a vehicle to perceive its environment, plan safe trajectories, and execute controlled maneuvers. These subtasks typically include sensing, mapping, obstacle avoidance and path planning [47, 48], while vehicle control is generally a task assigned a control system. Autonomous navigation protocols can range from relatively simple reactive systems, which respond directly to sensor inputs, to sophisticated multi-layered architectures that integrate high-level decision-making with low-level control [49]. By decomposing the overall navigation challenge into these distinct components, researchers can more effectively develop, test, and refine individual algorithms, ultimately leading to more robust and resilient autonomous systems capable of operating under a wide variety of conditions. For the purpose of this thesis, as only a simple framework for autonomous navigation is to be developed and implemented, only the specific aspects of behavior planning and path planning will be considered.

The specific task of determining the behaviours that an autonomous vehicle should have during its operations is called Behaviour Planning (BP). In particular, BP refers to the high-level decision-making process that determines a vehicle's overall actions and strategies based on its environment, mission objectives, and operational constraints. BP is responsible for selecting appropriate maneuvers - such as lane changes, obstacle avoidance, or route re-planning - by interpreting sensor data and integrating information from perception and localization modules. The challenges inherent in BP include managing dynamic and unpredictable environments, ensuring safety and compliance with navigation rules (such as road and air traffic rules), and achieving real-time performance despite computational constraints. Here notable approaches to BP will be presented, broadly divided into classical rule-based methods and more recent learning-based techniques [50, 51]: for each approach, notable methodologies will be discussed and for each their associated advantages and limitations will be mentioned.

Classical methods for behavior planning are typically based on rule-based frameworks such as Finite State Machines (FSM) and Hierarchical Finite State Machines (HFSM). FSMs model behavior as a set of discrete states with defined transitions triggered by specific events or conditions, making them intuitive and relatively simple to implement [49]. Expanding on the capabilities of the simple FSM architecture, HFSMs extend this concept by introducing hierarchical structures that allow for modularity and better management of complex behaviors. These approaches are generally easy to produce and verify, and offer clear and deterministic behaviors (a crucial feature for safety-critical applications). While these methods have great

simplicity, their main disadvantage lies in scalability: as the complexity of the environment increases, the number of states and transitions can grow exponentially, making the system brittle and less adaptable to unforeseen situations. Additionally, classical methods require extensive manual design and tuning, which may limit their performance in highly dynamic or uncertain scenarios as all possible variations in working environment should be accounted for in the design phase [50, 52].

With the rapid improvement of machine learning frameworks, learning-based methods have recently seen large interest and success for their ability to more effectively deal with dynamic environments. Learning-based methods leverage data-driven approaches such as deep learning and reinforcement learning to model and predict optimal behavior in complex environments: these methods utilize large datasets to train models capable of generalizing to a variety of scenarios, potentially overcoming the rigidity of rule-based systems. Deep learning-based techniques can extract high-level features from raw sensor data, while reinforcement learning methods enable systems to learn optimal policies through trial and error in simulated or real environments [53]. Although these approaches offer significant promise in adapting to new and unforeseen situations, they also come with challenges such as high computational requirements, the need for extensive training data, and issues related to explainability and safety assurance. Despite these hurdles, the integration of learning-based methods into BP has shown considerable potential in improving decision-making and responsiveness, with some interest in developing hybrid approaches that combine classical and learning based methods to compensate for the limitations of each approach [50, 51].

Path planning is the process by which an autonomous vehicle determines an optimal route from its starting position to its destination while avoiding obstacles and satisfying dynamic constraints. This process is critical for ensuring safe, efficient, and reliable navigation, but it is fraught with challenges such as handling uncertainties in the environment, managing high-dimensional state spaces, and coping with dynamic obstacles in real time. Notable approaches to trajectory generation have been developed to address these challenges, and they are typically divided into deterministic and learning-based methods [54]. Each category exhibits distinct characteristics: deterministic methods generally offer guarantees on optimality and completeness under well-defined conditions, while learning-based methods promise improved adaptability in complex, uncertain environments. For a comprehensive review of path planning techniques, see [55, 56].

Deterministic methods for trajectory planning are often based on graph search algorithms and optimization techniques, implemented to enable the definition of trajectories both online and offline. Classical path-planning algorithms such as A* and Dijkstra's are widely used; A* leverages heuristics to efficiently search for an optimal path in a discretized state space, whereas Dijkstra's algorithm ensures the shortest path in weighted graphs without heuristic bias. These approaches are effective in structured, static environments but may encounter scalability issues and computational challenges when applied online in complex or highly dynamic settings. In spite of their flaws, these algorithms allow for both offline and online path planning and as such are considered very valuable in guidance and trajectory definition applications [49]. In addition to these, offline kinematic motion planning methods have been developed to generate smooth, feasible trajectories that respect the vehicle's dynamic constraints, as demonstrated in [57]: these approaches tend to either generate complete paths and then sample waypoints between them, or to first place waypoints in space and then generate a trajectory connecting them by using interpolation functions such as clothoidal paths, base splines, or more complex optimization algorithms [58]. While deterministic methods provide reliable solutions and clear performance metrics, their rigidity and potential computational overhead limit their effectiveness in rapidly changing environments.

In contrast, learning-based methods for trajectory planning harness the power of data-driven models to deal with environmental uncertainties and complex dynamics. Deep learning and reinforcement learning techniques have recently emerged as promising alternatives, enabling vehicles to learn optimal or near-optimal paths directly from sensor data and environmental interactions. For instance, deep reinforcement learning approaches have been successfully applied to navigation tasks, allowing systems to adapt to dynamic obstacles and continuously evolving scenarios [54, 59]. Although learning-based methods offer significant flexibility and adaptability, they generally require large amounts of training data, extensive computational resources during training, and may lack the theoretical guarantees of optimality that deterministic methods provide. Despite these challenges, the fusion of learning-based strategies with traditional deterministic planning holds considerable promise for advancing the field of autonomous navigation.

As this thesis aims at producing an autonomous systems capable of following precise and agile manoeuvres, the trajectory generation process will not be explored in the context of being able to generate manoeuvres in a dynamic environment. To thoroughly investigate the capabilities of the helicopter, the trajectory generation problem will be handled using offline-generated trajectories identified using a kinematic approach: this will allow to purposefully generate trajectories capable of testing the effectiveness of the controlled system.

3

Methodology

With the project presented in Chapter 1 and a comprehensive review of relevant literature and the current trends in research topics presented in Chapter 2, here the methodology used in the development of the thesis is presented. As discussed in Chapter 1, to properly address the research questions driving this study, the thesis will be divided in parts, each addressing one key aspect of the overall research objective.

The aim of this project is investigating the potential of quaternion implementation in modelling and autonomous flight of helicopters, in particular exploring agile and precise manoeuvering. To better address this the thesis has been clearly divided in parts, each delving into one of the specific subtasks contained by the research objective.

The first part of the thesis work aims to introduce quaternions from a theoretical standpoint, with particular attention to attitude representation. This is a fundamental section of the work as it provides the necessary background to make use of quaternions in the context of attitude description and establishes the notation used throughout the thesis.

Within this section quaternions and other attitude parametrizations are to be compared to one another, identifying their characteristics and discussing the advantages of the quaternion formulation to justify its use in this research work. To achieve this, the following steps are taken:

- First quaternions are defined and introduced in the context of attitude parametrization, establishing the mathematical operations used to represent rotation concatenations.
- Then quaternions are compared to the two most prominent methods of attitude parametrizations - Euler angles and the Direction Cosine Matrix (DCM) - to highlight the advantages of quaternions and justify their use in this work.

With this, quaternions are introduced and their equations are defined while simultaneously justifying their choice by actively comparing them to the other most common attitude parametrization methods, showing that quaternions directly address their fundamental criticalities at the cost of increased complexity in the modelling phase.

The second part of this work aims to analyse the linearized models used in the study. Moreover, since the available helicopter models use the Euler angles parametrization while the aim of this work is to study the implementation of a quaternion-based controller, a methodology has to be defined to convert these models to have their attitude states be expressed as quaternion elements.

To effectively analyse the baseline system and identify a methodology to convert an Euler angles-based linearized model into a quaternion-based linearized model, the following steps are taken:

- First analyse the baseline linear Euler angles models and discuss their characteristics from a dynamics and control standpoint:
  - Discuss available states/inputs/outputs.
  - Study stability via pole analysis.
  - Study modes via eigenvector analysis.

  - Discuss controllability/observability via matrix analysis.
- Identify the differential relations between the existing states of the system and the quaternion states.
- Modify the existing linearized matrices with the differential relations identified in the previous step by incorporating the chain rule of derivatives to have the states be represented as quaternions.
- Analyse the new quaternion-based model and discuss its characteristics:
  - Discuss available states/inputs/outputs.
  - Study stability via pole analysis.
  - Study modes via eigenvector analysis.
  - Discuss controllability/observability via matrix analysis.
- Verify the conversion from Euler angles to quaternions is successful by comparing the characteristics of the two models (stability, controllability, observability, eigenvectors).
- Verify the conversion by also comparing open-loop response to inputs and verify they overlap.

With these steps, a methodology is presented to convert an existing Euler-angles based model into a quaternion-based model. Further, the models are discussed and analysed: the properties identified will be crucial in the choice of an appropriate flight control system architecture, which will be handled later in the study.

With the linearized model discussed and converted to incorporate a quaternion formulation, an appropriate control structure is then introduced. The methodology selected will be highly dependent on the linearized system's characteristics. To ensure the controller is understandable in both the implementation and tuning, the following methodology will be followed:

- Identify a suitable control architecture based on the discussion of the plant performed in the previous step.
- Develop a control law and plan the implementation of the control system with a block diagram of the controller, highlighting the signals required and produced by each block to identify how they are to interact with one another.
- Identify a tuning strategy for the controller loops and identify parameters to evaluate the optimality of the tuning.
- Perform the tuning of the control system and verify the tuning by simulating the response to sample signals.

This description allows to produce and test via desktop simulations a control system capable of tracking signals in position, velocity and attitude, allowing for complete control of the system during flight.

As a last element of this work, the final part of the thesis will focus on enabling autonomous flight and testing the capabilities of the system to handle agile manoeuvres. This involves the identification of appropriate manoeuvres to be performed and the implementation of an autopilot to allow for autonomous navigation. The methodology followed to achieve this is defined hereafter:

- Identify and characterize appropriate agile manoeuvres to test the controlled helicopter, use reference manuals (e.g.: ASD-33, UTTAS) to define performance metrics.
- Generate the reference signals for position, velocity and attitude to follow these manoeuvres:
  - Define via a set of differential equations position and velocity profiles to describe the manoeuvres.
  - Identify the acceleration required to follow such references by differentiating the velocity profile.
  - Generate an initial attitude reference around the manoeuvre by assuming the main rotor thrust to be aligned to the helicopter's body z axis.
  - Correct the attitude reference identified to account for heading angle information.
  - Sample the manoeuvre signals into waypoints to allow for smoother navigation around the manoeuvre.
- Implement an FSM to serve as a baseline autopilot for the closed-loop system.
- Simulate the manoeuvres using MATLAB and Simulink.
- Evaluate the performance of the controller by discussing the tracking error around the manoeuvre.

4

Contributions

This thesis serves as an addition to the current body of work produced by the scientific community in two primary areas of research: quaternion implementation in modelling and flight control systems, and autonomous flight through agile trajectories. Here the specific contributions of this thesis are highlighted:

- **Chapter 7:** A complete methodology for the conversion of an identified linearized state-space model using Euler angles for parametrization into a linearized model using quaternions for attitude parametrization is presented. The conversion is justified via mathematical derivation of the equations and the fidelity of the conversion is verified by comparing the principal characteristics of the two systems: pole location, eigenvectors, controllability and observability, and response to open-loop signals. The procedure proved beneficial as it enabled the retention of already identified dynamics and facilitated controller design by integrating quaternion attitude elements directly in the state-space system's state vector.
- **Chapter 8:** A full flight control system architecture is developed to enable controlled flight of the helicopter vehicle: here the overall structure is identified and each element of the flight control system is presented and discussed, and particular attention is placed on highlighting the motivation behind the choice of specific control techniques. To ensure the interpretability of the system and facilitate the tuning process, the flight control system is developed using nested loops to control. Particular attention is given to the novel proposed development of an LQI controller for the attitude of the system, providing a repeatable approach to its implementation.
- **Chapter 9:** A complete methodology for the tuning of the controller is presented using an optimization-based approach based on the Particle Swarm Optimization (PSO) algorithm. The optimization algorithm is applied on an LQI controller, a PI controller, and a P controller sequentially; for each control loop, a cost function is defined for the optimization of the parameters and its terms are discussed and analysed. The approach presented is repeatable and can be applied to different systems with minimal modifications, offering a structured and flexible approach to tuning of MIMO systems based on closed-loop performance.
- **Chapter 11:** An implementation of a simple Finite State machine (FSM) for the purpose of autonomous flight and trajectory tracking is provided, allowing for trajectory tracking of offline predefined signals. The FSM formulation is such that the structure presented can be augmented to allow for more complex behaviours, such as the implementation of algorithms to determine trajectories online capable of adapting to dynamic environments. The approach used to generate the autopilot references is based on kinematic relations and simple dynamic equations, and its application to manoeuvre definition is presented in Chapter 10.

Part III

Quaternions

5

An introduction to quaternions

In the context of aerospace engineering, the problem of representing a rigid body's orientation in 3D space is fundamental: air vehicles and space vehicles are set apart by their ability to function in a number of highly different flight configurations, even assuming highly unconventional attitudes during complex and agile manoeuvering. When trying to model a vehicle's motion through an agile and complex trajectory, classical methods, such as Euler angles and rotation matrices, have well-known limitations: Euler angles suffer from gimbal lock, and rotation matrices are computationally expensive and redundant. These criticalities pose limitations to the application of conventional attitude parametrizations in highly-manoeuvrable systems, where smooth attitude representation and fast computations are required. In contrast, unit quaternions offer a compact, efficient, and non-singular representation of attitude, making them an attractive solution for various applications.

This chapter aims at introducing the fundamental properties of quaternions; providing a simple and clear description of their mathematical formulation, while also highlighting the advantages of using quaternions for attitude representation over other more commonly used methods. To orderly introduce quaternions and their properties, here a description of the layout of the chapter is given. To start, a comprehensive introduction to quaternions will be provided, giving an understanding of their basic structure and providing a way for quickly interpreting how they can be used to represent the relative attitude of two reference frames in 3D space. Further, a discussion of their fundamental mathematical properties will be given by discussing: mathematical operations between quaternions, rotation concatenation, and the relation between quaternions and Euler angles. To conclude, the reason behind using quaternions for attitude representation in aerobatic flight will be highlighted by comparing quaternions with two other commonly used attitude representation methods: Euler angles and the Direction Cosine Matrix (DCM).

5.1. An introduction to quaternions for attitude representation

Quaternions are a number system invented by mathematician William Rowan Hamilton in 1843, originally created to propose a type of hypercomplex number that could be described using three-dimensional space in the same way conventional complex numbers could be described using planar geometry [60].

Quaternions are four-dimensional numbers and - in their most general form - they are generally presented with the formula:

$$q = q_0 1 + q_i i + q_j j + q_k k \tag{5.1}$$

where $[q_0, q_i, q_j, q_k] \in \mathbb{R}$ are scalar coefficients of the quaternions while $1, i, j, k$ are the fundamental quaternion units, which also serve as a basis to $\mathbb{R}^4$ and essentially act as a multidimensional extension of the imaginary unit i commonly used for complex numbers. The basis elements i, j, k abide by the following multiplication rules [60, 9]:

$$\begin{cases} i^2 = j^2 = k^2 = ijk = -1 \\ ij = -ji = k \\ jk = -kj = i \\ ki = -ik = j \end{cases} \tag{5.2}$$

Alongside the extended formula of the quaternion shown in Equation 5.1, an equivalent and more compact vector representation can easily be defined, as shown in Equation 5.3. In this notation, the quaternions are generally defined as the concatenation of two distinct parts: a *real part* (also referred to as *scalar part*) q_0 and an *imaginary part* (also referred to as *vector part*) $\bar{\mathbf{q}} = [q_i, q_j, q_k]^T$. This notation is widely used in scientific papers and its is useful as it allows for the transcription of more complex quaternion operations as simple matrix multiplications (as discussed in Section 5.2).

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_i \\ q_j \\ q_k \end{bmatrix} = \begin{bmatrix} q_0 \\ \bar{\mathbf{q}} \end{bmatrix} \tag{5.3}$$

5.1.1. Pure quaternions and unit quaternions

Of all quaternions belonging to $\mathbb{R}^4$, two subcategories of quaternions are of particular interest when investigating attitude representation: *pure quaternions* and *unit quaternions*.

Pure quaternions are a subset of quaternions that have a real part equal to 0. By considering the imaginary quaternion units i, j, k to be the standard orthonormal basis of $\mathbb{R}^3$:

$$i = [1, 0, 0]^T \qquad j = [0, 1, 0]^T \qquad k = [0, 0, 1]^T$$

it follows that every vector element $\hat{\mathbf{n}} \in \mathbb{R}^3$ can be expressed as a pure quaternion in $\mathbb{R}^4$, with the real part set to zero $q_0 = 0$ and the $\mathbb{R}^3$ vector as the imaginary part $\bar{\mathbf{q}} = \hat{\mathbf{n}}$. This representation of vectors in $\mathbb{R}^3$ as pure quaternions in $\mathbb{R}^4$ is instrumental in the inclusion of 3D coordinates in quaternion algebra, as discussed in Section 5.2.

Unit quaternions are essential in the representation of attitudes in 3D space [61] and they are characterised by the additional internal constraint:

$$\|\mathbf{q}\| = q_0^2 + q_i^2 + q_j^2 + q_k^2 = 1 \tag{5.4}$$

To intuitively understand how a quantity in $\mathbb{R}^4$ - the unit quaternion - may be used to represent an orientation in $\mathbb{R}^3$, one can simply recall Euler's rotation theorem, which states that the attitude of a rigid-body can be changed from any given orientation to any other orientation by rotating by a certain angle (called the *Euler angle*) the body about an axis (called the *Euler axis*) that is fixed to the rigid body, as shown in Figure 5.1. By considering θ to be the *Euler angle* and the unit vector $\mathbf{n} \in \mathbb{R}^3$ to be the *Euler axis*, the unit quaternion in Equation 5.5 can be defined such that it includes all relevant information for the representation of a body's relative attitude in 3D space [62].

$$\mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\,\mathbf{n} \end{bmatrix} \tag{5.5}$$

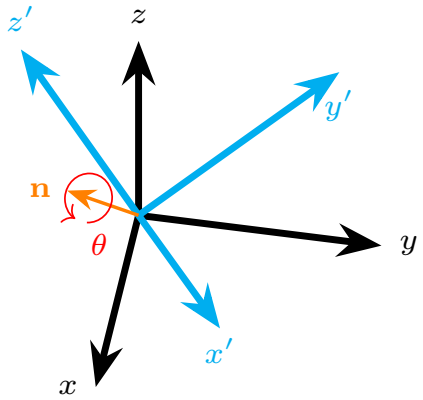


Figure 5.1: Graphical rendering of Euler's rotation theorem, the Euler angle θ and the Euler axis $\mathbf{n}$ are shown explicitly.

5.2. Quaternion operations

With quaternions introduced in Section 5.1, essential mathematical operations involving them will now be discussed. To start, Section 5.2.1 the notation used to indicate quaternions throughout the thesis will be given, providing an explicit and immediate relation between the quaternion itself and its physical interpretation. After this, in Section 5.2.2 some essential mathematical operations will be discussed: quaternion inverse, quaternion multiplication, and active and passive rotations. Finally equivalent quaternions will be introduced, discussing error representation and the ”unwinding phenomenon” in Section 5.2.3.

5.2.1. Quaternion notation

As discussed in Section 5.1.1, unit quaternions expressed as shown in Equation 5.5 can directly be related to a rotation as per Euler’s rotation theorem. Now - to highlight the physical significance of quaternions and provide an immediate interpretation - the notation used in this thesis to indicate quaternions will be presented. Consider the reference frames $\mathcal{A}$ and $\mathcal{B}$: the quaternion describing the relative attitude of $\mathcal{B}$ with respect to $\mathcal{A}$ is expressed as

$$\mathbf{q}_{A2B} = \begin{bmatrix} \cos\left(\theta_{A2B}/2\right) \\ \sin\left(\theta_{A2B}/2\right) \{\mathbf{n}_{A2B}\}_A \end{bmatrix} \tag{5.6}$$

meaning that, to align $\mathcal{A}$ to $\mathcal{B}$, reference frame $\mathcal{A}$ has to undergo a right-handed rotation of angle θ_{A2B} around the unit vector $\mathbf{n}_{A2B}$ with coordinates measured in reference frame $\mathcal{A}$. Note that since $\mathcal{A}$ rotates around $\mathbf{n}_{A2B}$ to obtain $\mathcal{B}$, the coordinates of the unit vector remain the same in the two reference systems, i.e.: $\{\mathbf{n}_{A2B}\}_A = \{\mathbf{n}_{A2B}\}_B$.

5.2.2. Mathematical operations

Here, the fundamental mathematical operations of quaternions will be discussed, alongside a physical interpretation of them. For the purposes of attitude representation, there are four essential operations that must be highlighted: quaternion multiplication, quaternion inversion, active rotations and passive rotations.

Quaternion multiplication

Consider the arbitrary quaternions **p** and **q**, expressed as:

$$\mathbf{p} = \begin{bmatrix} p_0 \\ \bar{\mathbf{p}} \end{bmatrix}$$

$$\mathbf{q} = \begin{bmatrix} q_0 \\ \bar{\mathbf{q}} \end{bmatrix}$$

let quaternion **r** be the result of the Hamilton’s product operation (hereby referred to simply as ”quaternion multiplication” or ”quaternion product”) [62] of quaternions **p** and **q**. The product of **p** and **q** may be expressed using the following equation [63]:

$$\mathbf{r} = \mathbf{pq} = \begin{bmatrix} q_0 p_0 - \bar{\mathbf{q}} \cdot \bar{\mathbf{p}} \\ p_0 \bar{\mathbf{q}} + q_0 \bar{\mathbf{p}} + \bar{\mathbf{p}} \times \bar{\mathbf{q}} \end{bmatrix} \tag{5.7}$$

Note that, as evidenced by the presence of the cross product ($\times$), quaternion multiplication is non-commutative. Another way of expressing Equation 5.7 is by means of matrix multiplication, resulting in the formulation shown in Equation 5.8.

$$\mathbf{r} = \mathbf{pq} = \begin{bmatrix} p_0 & -p_i & -p_j & -p_k \\ p_i & p_0 & -p_k & p_j \\ p_j & p_k & p_0 & -p_i \\ P_k & -p_j & p_i & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_i \\ q_j \\ q_k \end{bmatrix} \tag{5.8}$$

Quaternion multiplication is significant for attitude representation as it allows for a simple and intuitive way of concatenating successive rotation. Consider reference frames $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, where $\mathcal{B}$ is obtained by rotating $\mathcal{A}$ around the Euler axis $\{\mathbf{n}_{A2B}\}_A$ by the Euler angle θ_{A2B} and $\mathcal{C}$ is obtained by rotating $\mathcal{B}$ around the Euler axis $\{\mathbf{n}_{B2C}\}_B$ by the Euler angle θ_{B2C}. With the notation described in Section 5.2.1, the relative attitude of $\mathcal{B}$ with respect to $\mathcal{A}$ and of $\mathcal{C}$ with respect to $\mathcal{B}$ are described by quaternions $\mathbf{q}_{A2B}$ and $\mathbf{q}_{B2C}$ respectively. By using Equation 5.5, the two attitude quaternions can be written as:

$$\mathbf{q}_{A2B} = \begin{bmatrix} \cos(\theta_{A2B}/2) \\ \sin(\theta_{A2B}/2)\{\mathbf{n}_{A2B}\}_A \end{bmatrix}$$
$$\mathbf{q}_{B2C} = \begin{bmatrix} \cos(\theta_{B2C}/2) \\ \sin(\theta_{B2C}/2)\{\mathbf{n}_{B2C}\}_B \end{bmatrix}$$

By means of quaternion multiplication it is possible to immediately find quaternion $\mathbf{q}_{A2C}$ expressing relative attitude of reference frame $\mathcal{C}$ with respect to frame $\mathcal{A}$, as shown in Equation 5.9.

$$\mathbf{q}_{A2C} = \mathbf{q}_{A2B}\mathbf{q}_{B2C} \tag{5.9}$$

Quaternion conjugation and quaternion inversion
Consider the arbitrary quaternion $\mathbf{q} \in \mathbb{R}^4$, with real part q_0 and vector part $\bar{\mathbf{q}}$: the complex conjugate of $\mathbf{q}$ - denoted as $\mathbf{q}^*$ - is immediately identified by Equation 5.10.

$$\mathbf{q}^* = \begin{bmatrix} q_0 \\ -\bar{\mathbf{q}} \end{bmatrix} \tag{5.10}$$

Quaternion inversion is another important mathematical operation when it comes to quaternion algebra. Consider the arbitrary quaternion $\mathbf{q} \in \mathbb{R}^4$: the inverse of $\mathbf{q}$ - denoted as $\mathbf{q}^{-1}$ - is identified via Equation 5.11 [61].

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} \tag{5.11}$$

By multiplying an arbitrary quaternion with its inverse, the result is the *identity quaternion*, a peculiar unit quaternion which has unitary real part and null vector part. It is worth noting that - when considering quaternions as means of expressing relative attitude as per Equation 5.5 - the identity quaternion indicates that the two reference systems being considered are actually aligned, as it is obtained only when setting the Euler angle θ to zero.

$$\mathbf{q}^{-1}\mathbf{q} = \mathbf{q}\mathbf{q}^{-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.12}$$

Better analysing Equation 5.11, it is immediate to notice that - for a unit quaternion - the inverse $\mathbf{q}^{-1}$ and the complex conjugate $\mathbf{q}^*$ are the same.

Quaternion inversion is also relevant in the context of attitude representation. Consider for example the quaternion $\mathbf{q}_{A2B}$, which expressed the attitude of reference frame $\mathcal{B}$ with respect to frame $\mathcal{A}$: the inverse quaternion $\mathbf{q}_{A2B}^{-1}$ is still a unit quaternion and it represents the relative attitude of frame $\mathcal{A}$ with respect to frame $\mathcal{B}$ [64]. Using the quaternion notation discussed in Section 5.2.1, it follows that:

$$\mathbf{q}_{A2B}^{-1} = \mathbf{q}_{B2A} = \begin{bmatrix} \cos(\theta_{B2A}/2) \\ \sin(\theta_{B2A}/2)\{\mathbf{n}_{B2A}\}_B \end{bmatrix} \tag{5.13}$$

Active and passive rotations
With the introduction of quaternions as means of attitude representation, interest in their for coordinate transformation rose and intuitive formulations for the application of rotations were identified. Here the more relevant operations for the application of rotation will be presented, providing clear mathematical formulas for the application of both *active* and *passive rotations* via unit quaternions: in the interest of brevity proof for these mathematical operations will not be discussed, but note that further information and clear mathematical derivations may be found in [61, 63, 62].

Active rotations (also known as *point rotations* or *vector rotations*) are mathematical operations which allow for the rotation of a point (or a vector) in a fixed coordinate system. Active rotations are especially useful when trying to determine the coordinates of a point after a certain rotation has been applied to it.

Consider vector $\mathbf{v} \in \mathbb{R}^3$ with its coordinates expressed in reference frame $\mathcal{A}$: using the notation discussed in Section 5.2.1 this quantity would be expressed as $\{\hat{\mathbf{v}}\}_A$. Let $\mathbf{r} \in \mathbb{R}^3$ be the result of rotating $\mathbf{v}$ around the unit vector $\{\mathbf{n}_{v2r}\}_A$ by angle θ_{v2r}, such that the quaternion describing this rotation is:

$$\mathbf{q}_{v2r} = \begin{bmatrix} \cos(\theta_{v2r}/2) \\ \sin(\theta_{v2r}/2)\{\mathbf{n}_{v2r}\}_A \end{bmatrix}$$

Expressing $\mathbf{v}$ and $\mathbf{r}$ as pure quaternions, the coordinates of $\{\mathbf{r}\}_A$ may then be calculated by applying the active rotation operator, as shown in Equation 5.14.

$$\begin{bmatrix} 0 \\ \{\mathbf{r}\}_A \end{bmatrix} = \mathbf{q}_{v2r} \begin{bmatrix} 0 \\ \{\mathbf{v}\}_A \end{bmatrix} \mathbf{q}^*_{v2r} \tag{5.14}$$

Passive rotations (also known as *coordinate frame rotations*, or more simply *frame rotations*) are mathematical operations that allow for the rotation of the coordinate frame in which a vector is expressed. Passive rotations are especially useful when trying to express the coordinates of a vector, know in a certain reference frame, in another reference frame.

Consider vector $\mathbf{v} \in \mathbb{R}^3$ of know coordinates in reference frame $\mathcal{A}$, expressed as $\{\mathbf{v}\}_A$. Let $\mathcal{B}$ be the reference frame obtained by rotating $\mathcal{A}$ around the unit vector $\{\mathbf{n}_{A2B}\}_A$ by angle θ_{A2B}, such that the relative attitude of $\mathcal{B}$ with respect to $\mathcal{A}$ may be expressed via the quaternion:

$$\mathbf{q}_{A2B} = \begin{bmatrix} \cos(\theta_{A2B}/2) \\ \sin(\theta_{A2B}/2)\{\mathbf{n}_{A2B}\}_A \end{bmatrix}$$

Expressing $\mathbf{v}$ as a pure quaternion, the coordinates of $\mathbf{v}$ in coordinate frame $\mathcal{B}$ (i.e.: $\{\mathbf{v}\}_B$) may then be calculated by applying the passive rotation operator, as shown in Equation 5.15.

$$\begin{bmatrix} 0 \\ \{\mathbf{v}\}_B \end{bmatrix} = \mathbf{q}^*_{A2B} \begin{bmatrix} 0 \\ \{\mathbf{v}\}_A \end{bmatrix} \mathbf{q}_{A2B} \tag{5.15}$$

5.2.3. Equivalent quaternions: error representation and unwinding

A final important concept when discussing quaternions, especially in the context of attitude representation, is the concept of *equivalent quaternions*. To properly introduce them, consider the reference reference systems $\mathcal{A}$ and $\mathcal{B}$, with $\mathcal{B}$ obtained by performing a right-hand rotation of $\mathcal{A}$ around the Euler axis $\mathbf{n}_{A2B}$ by the Euler angle θ_{A2B}. As per Equation 5.6, the relative attitude of $\mathcal{B}$ with respect to $\mathcal{A}$ is expressed by the following quaternion.

$$\mathbf{q}_{A2B} = \begin{bmatrix} \cos(\theta_{A2B}/2) \\ \sin(\theta_{A2B}/2)\{\mathbf{n}_{A2B}\}_A \end{bmatrix} \tag{5.16}$$

Notice that rotating around a certain axis by a certain angle is exactly the same as rotating by a negative angle about the negative axis [65]. With this in mind, letting $\mathcal{C}$ be the result of rotating $\mathcal{A}$ around the axis

$\mathbf{n}_{A2C} = -\mathbf{n}_{A2B}$ by the explementary angle θ_{A2C} (such that $\theta_{A2B} - \theta_{A2C} = \pm 2\pi$ when $|\theta_{A2C}| \leq 2\pi$ [64]), it follows naturally that the reference systems $\mathcal{B}$ and $\mathcal{C}$ will have the same final attitude. From Equation 5.6 and considering $\theta_{A2C} = 2\pi - \theta_{A2B}$, the following relation for equivalent quaternions is derived:

$$\begin{aligned} \mathbf{q}_{A2C} &= \begin{bmatrix} \cos(\theta_{A2C}/2) \\ \sin(\theta_{A2C}/2)\{\mathbf{n}_{A2C}\}_A \end{bmatrix} = \begin{bmatrix} \cos((2\pi - \theta_{A2B})/2) \\ \sin((2\pi - \theta_{A2B})/2)\{-\mathbf{n}_{A2B}\}_A \end{bmatrix} \\ &= \begin{bmatrix} -\cos(\theta_{A2B}/2) \\ \sin(\theta_{A2B}/2)\{-\mathbf{n}_{A2B}\}_A \end{bmatrix} = -\mathbf{q}_{A2B} \end{aligned} \tag{5.17}$$

With the considerations discussed above, it follows that for every unit quaternions $\mathbf{q}$ there exists and *equivalent quaternion* $-\mathbf{q}$ that expresses the same relative attitude via a different rotation.

In the specific context of control systems, equivalent quaternions are especially important for the purpose of attitude error representation; to clearly understand this, consider a simple generic control scenario. Let $\mathcal{B}$ be the body frame of the aircraft, $\mathcal{E}$ to be the local NED frame and $\mathcal{D}$ to be the desired attitude of the aircraft: the objective of the control system is to move $\mathcal{B}$ such that it becomes aligned with $\mathcal{D}$. To perform its task, the control system must first quantify the difference in $\mathcal{B}$ and $\mathcal{D}$, which will then be used to generate appropriate control inputs for the helicopter. With the notation introduced in Section 5.2.1, the relative attitudes of $\mathcal{B}$ and $\mathcal{D}$ with respect to $\mathcal{E}$ are described by quaternions $\mathbf{q}_{E2B}$ and $\mathbf{q}_{E2D}$ respectively, and - as per Equations 5.9 and 5.13 - the quaternion describing the relative attitude of $\mathcal{D}$ with respect to $\mathcal{B}$ is given by Equation 5.18.

$$\mathbf{q}_{B2D} = \mathbf{q}_{B2E}\mathbf{q}_{E2D} = \mathbf{q}_{E2B}^{-1}\mathbf{q}_{E2D} \tag{5.18}$$

As per the concept of equivalent quaternions, $\mathbf{q}_{B2D}$ and $-\mathbf{q}_{B2D}$ effectively express the same relative attitude and as such they may both be used to define the attitude error for the control system. Even so, the two quaternions encode fundamentally different information: $\mathbf{q}_{B2D}$ represents a rotation by angle θ_{B2D} around the unit vector $\mathbf{n}_{B2D}$, while $-\mathbf{q}_{B2D}$ represents a rotation by the explementary angle of θ_{B2D} around the opposite vector $-\mathbf{n}_{B2D}$.

This ambiguity is the root cause of the ”unwinding phenomenon” [66], a peculiar behaviour shown by quaternion-controlled vehicles in which the vehicle is driven to perform an unnecessarily long rotation around the target attitude, producing large control displacements and not minimising the angle of rotation. For the purpose of developing a control system, it is essential to define the desired attitude change in a way that allows for a rotation by the smallest possible angle. This condition is achieved when the Euler angle of rotation θ is less than π radians [67, 64], which - for a generic unit quaternion $\mathbf{q} = [q_0, \bar{\mathbf{q}}]^T$ - translates to the condition:

$$q_0 \geq 0 \tag{5.19}$$

With Equation 5.19, a clear condition to discern which equivalent quaternion allows for the smallest desired rotation has been identified.

5.3. Comparison of various attitude parametrizations

With the information presented above, quaternions have been discussed from a mathematical standpoint and a focus on their relevance for the field of attitude representation has been given, discussing in particular the specific application of different mathematical operators (such as quaternion multiplication, inversion, and active and passive rotations) for the purpose of practical attitude representation and reconstruction.

To add to the information presented thus far in the chapter, this final section aims at comparing quaternions with other more common methods of attitude representation, briefly discussing each of them individually and clearly pointing out their advantages and disadvantages over each other. To do so, three forms of attitude representation will be considered: Euler angles, DCM, and quaternions. Note that, for the purpose of this thesis, only an overview of the specific formulation will be provided, discussing only the more relevant information and mathematical equations. The following sources provide a more complete and in-depth background [68, 69, 70], that is only mentioned here for completeness and to provide other more in-depth resources.

5.3.1. Euler angles

Of the three methods of attitude parametrization considered, Euler angles are the most intuitive and most commonly used. The fundamental concept behind Euler angles is that any orientation in 3D space is uniquely described by an ordered sequence of rotations around certain axis, also referred to as a ”set”.

Overview and fundamental equations

In the field of aerospace engineering the most common set of rotations in the so-called [3-2-1] Euler angles set, which is defined as follows. Consider a vehicle aligned with the local North-East-Down (NED) reference frame $\mathcal{E}$, such that $\mathcal{B}_0 \equiv \mathcal{E}$: first rotate the vehicle around the z-axis of $\mathcal{B}_0$ by the yaw angle ψ, obtaining the reference frame $\mathcal{B}_z$; then, rotate the current body reference frame $\mathcal{B}_z$ around its y-axis by the pitch angle θ, obtaining the reference frame $\mathcal{B}_y$; finally, rotate $\mathcal{B}_y$ around its x-axis by the roll angle ϕ, obtaining the final attitude of the body reference frame $\mathcal{B}$. The numbers identifying the rotation sequence correspond to the respective axis of the intermediate reference systems produced by the rotation sequence, and because of this the [3-2-1] Euler angles set in sometimes also referred to as a [Z-Y-X] rotation set.

As an example Figure 5.2 shows a [3-2-1] Euler rotation set, with first a rotation of $45°$ around the z-axis of $\mathcal{B}_0$, then a rotation of $30°$ around the y-axis of $\mathcal{B}_z$ and finally a rotation of $40°$ around the x-axis of $\mathcal{B}_y$, obtaining the final reference frame $\mathcal{B}$.

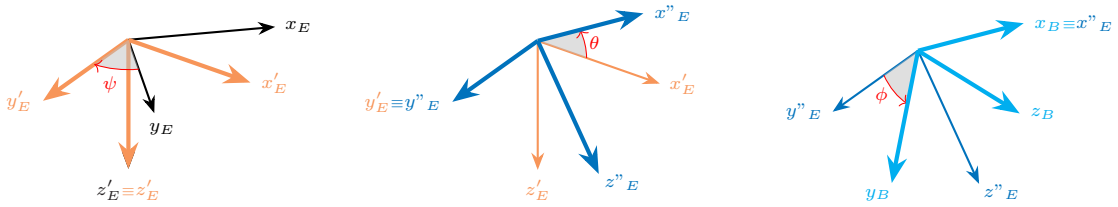


Figure 5.2: Reference system rotation using a [3-2-1] Euler angles set

To translate a vector from the NED reference frame $\mathcal{E}$ to the body reference frame $\mathcal{B}$, rotation matrices are used. The mathematical expression of the rotation matrix to be used depends heavily on the specific Euler angles set considered: for the purpose of translating a vector from $\mathcal{E}$ to $\mathcal{B}$, the rotation matrix to be used is given in Equation 5.20 [68] (for readability c and s respectively indicate the cosine and the sine of the subscript Euler angle):

$$R_{E2B}^{[321]} = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\theta + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & c_\theta s_\phi \\ s_\psi s_\theta + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\theta c_\phi \end{bmatrix} \tag{5.20}$$

where the superscript $[321]$ indicates the Euler angles set and the subscript $E2B$ the starting and final reference frames. Note that the matrix describing the opposite transformation, i.e.: $\mathbf{R}_{B2E}^{[321]}$, is simply the inverse of $\mathbf{R}_{E2B}^{[321]}$. It is also with noting that since the matrix $\mathbf{R}_{E2B}^{[321]}$ is an orthonormal matrix, the inverse operation is trivial as the inverse of an orthonormal matrix is the same as its transpose. These considerations are summed in Equation 5.21.

$$\mathbf{R}_{B2E}^{[321]} = \left(\mathbf{R}_{E2B}^{[321]}\right)^{-1} = \left(\mathbf{R}_{E2B}^{[321]}\right)^{T} \tag{5.21}$$

Other equations of great importance for the purpose of control system design are the time-propagation equations, which relate the time rate of change of the specific parametrization used to the body rates of the vehicle considered. For the Euler angles, the time-propagation equations are the following [71].

$$\frac{d}{dt}\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{5.22}$$

where $[p,\, q,\, r\,]$ are the body-fixed rotation rates.

Advantages and disadvantages
While Euler angles provide an intuitive way of expressing attitude and are commonly used in many control systems, there are a number of disadvantages related to them that make their use undesirable, especially in the context of high-performance control systems and especially aerobatic control systems.

To start, in order to make the Euler angles parametrization unique, the values that may be assumed by the Euler angles are limited as shown in Table 5.1.

Table 5.1: Limits of the Euler angles parametrization

Angle	Limits
ϕ	$[-\pi,\, \pi]$
θ	$[-\pi/2,\, \pi/2]$
ψ	$[-\pi,\, \pi]$

These limitations are strictly related to the problem of ”gimbal lock”, a singularity that occurs when two of the three rotation axes align reducing the degrees of freedom from three to two. This is especially problematic in aerospace applications, where vehicles perform continuous, arbitrary rotations. When gimbal lock occurs, control algorithms based on Euler angles can lose the ability to distinguish between different orientations, leading to instability or control failure.

Another criticality related to the Euler angles parametrization is the frequent use of transcendental functions. Even considering the simple operation of propagating attitude measurements in time, the resulting functions are highly non-linear and heavily rely on the use of matrix inversion (which is a very computationally expensive and prone to inaccuracies) and on transcendental functions (shown clearly in Equation 5.22).

5.3.2. Direction Cosine Matrix (DCM)
To solve some of the problems of Euler angles, the Direction Cosine Matrix approach is here presented.

Overview and fundamental equations
The fundamental idea behind this representation is that, instead of identifying the Euler angles $[\phi,\, \theta,\, \psi]$ and then calculating the rotation matrix, an alternative way of expressing the attitude of a body is by direct calculation of the coefficients of the matrix operator defined in Equation 5.20. This rotational matrix is referred to as the Direction Cosine Matrix (DCM).

By directing calculating the coefficients of the DCM, the time propagation equation for the Euler angles is not applicable directly and needs to be redefined. Once again considering $[p,\, q,\, r]$ to be the body-fixed rotation rates of the vehicle considered and considering $R_{E2B}^{[321]}$ to be the DCM calculated at a certain time step, the time propagation equations of the rotation matrix is derived to be [71]:

$$\frac{d}{dt}R_{E2B}^{[321]} = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} R_{E2B}^{[321]} \tag{5.23}$$

Advantages and disadvantages
While the fundamental advantage of the DCM parametrization is that the transformation matrix is formed directly, without any need for additional transcendental equations, there are many disadvantages that may discourage from its use.

Two immediately noticeable disadvantages of this approach are a loss of interpretability and a higher computational load. Firstly, by directly constructing the transformation matrix, the immediate interpretation of the Euler angles is lost in favour of a more mathematically robust implementation, which causes the information of the DCM to need further transformation if its data is to be displayed to pilots or other operators in real time. Secondly, another immediate disadvantage of this parametrization is the fact that, while Euler angles use only three parameters to completely and uniquely define the attitude of a body, the DCM uses nine, making its calculation and storage much more computationally expensive.

A final disadvantage of this attitude parametrization is the added susceptibility to error propagation and the expensive correction procedure. The fact that the DCM uses 9 unique elements to represent information that the Euler angles display with only 3 independent parameters suggests that there are 6 additional constraining equations that the DCM must adhere to: these constraints serve to ensure that the matrix computed is always orthonormal. These additional constraints are highly nonlinear and computationally expensive, and in practice, mean that to have a proper representation of attitude, the DCM must be continually normalised and re-orthogonalized.

5.3.3. Quaternions

With the criticalities of the Euler angles and the DCM parametrizations defined, quaternions are introduced as an alternative method for attitude parametrization that is capable of overcoming a great number of the disadvantages discussed thus far.

Overview and fundamental equations

The role and applicability of quaternions in the context of attitude representation has been extensively discussed in Sections 5.1 and 5.2, with their fundamental idea being connected to Euler's rotation theorem and their fundamental mathematical operations defined, with their relevance clearly outlined in the context of attitude calculations.

An additional equation not discussed thus far for the quaternions is the time propagation, which allows to express the derivative of a quaternion as a function of its current value and of the body rates of the vehicle considered. Considering $\mathbf{q}_{E2B}$ to be the quaternion describing the relative attitude of the body-fixed reference frame $\mathcal{B}$ with respect to the local NED frame $\mathcal{E}$, and $[p,\, q,\, r]$ to be the body-fixed rotation rates of the vehicle, the time-propagation equation that allows to express $\dot{\mathbf{q}}_{E2B}$ as a function of $\mathbf{q}_{E2B}$ and $[p,\, q,\, r]$ is [64]:

$$\dot{\mathbf{q}}_{E2B} = \frac{1}{2}\mathbf{q}_{E2B}\begin{bmatrix}0\\ \omega\end{bmatrix} \tag{5.24}$$

where ω is the body rate vector $[p,\, q,\, r]^T$ and $[0, \omega]^T$ represents the pure-quaternion interpretation of the body-rate vector in 3D space.

An additional advantage of quaternions is the possibility of using the spherical linear interpolation (SLERP) algorithm for efficient and smooth interpolation. Consider the control task of driving the aircraft from its initial attitude to a desired attitude, expressed respectively by quaternions $\mathbf{q}_{E2B}$ and $\mathbf{q}_{E2D}$. By using the SLERP algorithm, the shortest rotation between the two quaternions is easily identified, as shown in Equation 5.25.

$$\hat{\mathbf{q}}(l) = \frac{\sin\left((1-l)\theta\right)}{\sin(\theta)}\mathbf{q}_{E2B} + \frac{\sin\left(l\theta\right)}{\sin(\theta)}\mathbf{q}_{E2D} \tag{5.25}$$

where $\theta = \arccos\left(\mathbf{q}_{E2B} \cdot \mathbf{q}_{E2D}\right)$ and l is an interpolating parameter varying between 0 and 1. This highly efficient interpolation method allows for a smooth and singularity-free transition between two generic orientations represented by quaternions.

For attitude interpolation, the DCM parametrization also supports smooth, singularity-free attitude reconstruction techniques, but these are fairly complex and computationally expensive [72], making the procedure costly and with limited effectiveness (especially when compared to the immediacy of the SLERP algorthm for quaternions).

Advantages and disadvantages
Similarly to the DCM, the lack of an immediately understandable interpretation of quaternions is indeed a disadvantage: while the axis-angle interpretation of the rotation shown in Equation 5.5 is easy to understand, the presence of transcendental functions and the complexity of the rotation makes understanding the attitude of the vehicle in 3D space fairly difficult for the pilot or for other operators, especially in real-time applications.

Even so, the advantages of quaternions are clear to understand. First of all, the attitude is completely defined by only 4 parameters, making storing and computational load comparable to those of Euler angles and much more efficient that DCM. Further, the only additional constraint introduced by quaternions is the fact that - to properly express an attitude - the quaternion must always remain normalised, which is computationally more advantageous than performing the re-orthogonalization required by the DCM.

Another clear advantage of the quaternions over Euler angles is the lack of nonlinear and transcendental functions when discussing quaternion rotations (Equations 5.14 and 5.15) and quaternion multiplications (Equation 5.7), making rotation concatenations and time-propagation of quaternions extremely efficient, compact, and robust to computational errors and numerical integration.

One of the most essential qualities of quaternions is the complete avoidance of gimbal lock. This makes quaternions especially useful when dealing with manoeuvres that require large changes in the vehicle's attitude - such as aerobatic control of aircraft and spacecraft attitude control - as there is no need for additional switching logic. By parametrizing the attitude in 3D space using 4 parameters, quaternions provide a smooth, non-singular representation of any possible orientation in 3D space, even for large or continuous rotations. Finally, the SLERP algorithm allows for smooth, singularity-free interpolation of attitudes.

With this, an exhaustive comparison between the three attitude parametrization methods has been presented, showcasing the advantages of quaternions over the Euler angles and DCM representation in terms of numerical stability, memory usage, and computational efficiency, motivating and further advocating for their use in the modelling and control of complex and agile systems. A summary of the comparison between quaternions, Euler angles, and DCM is provided in Table 5.2.

Table 5.2: Comparison of attitude representation methods

Feature	Euler Angles	Direction Cosine Matrix	Quaternions
Number of parameters	3	9	4
Singularities	Yes	No	No
Computational efficiency	Low (transcendental functions, matrix inversion)	Low (9 elements, normalization)	High (simple algebra, normalization only)
Storage efficiency	High	Low	High
Ease of interpretation	Intuitive (roll, pitch, yaw)	Non-intuitive	Non-intuitive
Time propagation	Complex (nonlinear)	Simple (matrix multiplication)	Simple (linear in quaternion)
Numerical robustness	Poor (singularities, nonlinearity)	Poor (requires re-orthogonalization)	High (only normalization required)
Interpolation support	Difficult and non-smooth	Not practical	SLERP: smooth and efficient

5.4. Relation between quaternions and Euler angles

Now that quaternions have been defined in terms of their mathematical operations and their role in attitude representation has been discussed and their advantages over the more conventional Euler angles parametrization highlighted, here the equations that relate quaternions and Euler angles will be presented (considering a [3-2-1] rotation set). These equations allow for a robust conversion between the two attitude parametrizations, which is especially useful as Euler angles remain the preferred parametrization for intuitive interpretation and visualization of an aircraft's attitude in terms of roll, pitch, and yaw.

5.4.1. Euler angles to quaternion conversion

The fundamental equations governing the conversion between Euler angles and quaternions do not require any specific additional formula, as they can easily be derived from the concatenation of subsequent rotations via quaternion multiplication.

Considering a [3-2-1] rotation set with angle sequence $[\psi, \theta, \phi]$ describing the rotation between the NED reference frame $\mathcal{E}$ to the body reference frame $\mathcal{B}$, the equivalent quaternion $\mathbf{q}_{E2B}$ describing this rotation may be obtained by using the following equation:

$$\begin{aligned}\mathbf{q}_{E2B} &= \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix}\end{aligned} \tag{5.26}$$

A more complete source detailing all possible rotation sequences and the respective equations for the quaternion elements can be found in [73]. Even so, note that Equation 5.26 is essentially a quaternion multiplication of the three intermediate Euler angles rotations (appropriately ordered as to represent the specific Euler angles set considered): $\mathbf{q}_{E2B_z}$ describing a rotation of ψ around the reference system $\mathcal{E}$'s z-axis, $\mathbf{q}_{B_z2B_y}$ describing a rotation of θ around $\mathcal{B}_z$'s y-axis, and $\mathbf{q}_{B_y2B}$ describing a rotation of ϕ around $\mathcal{B}_y$'s x-axis, with the quaternions having the following equations.

$$\mathbf{q}_{E2B_z} = \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix} \quad \mathbf{q}_{B_z2B_y} = \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \quad \mathbf{q}_{B_y2B} = \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix}$$

With the equations described above, the three successive rotations shown in Figure 5.2 can now be condensed into a single rotation around an axis, as shown in Figure 5.3.

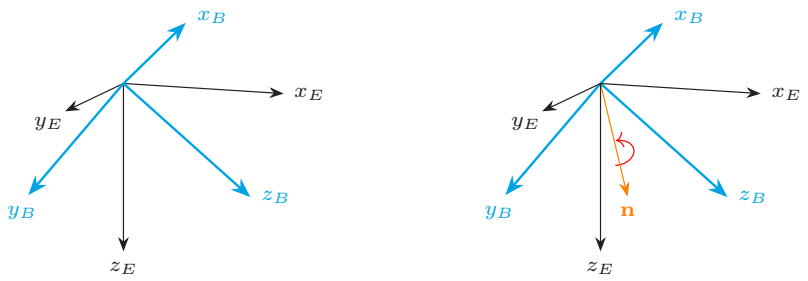


Figure 5.3: [3-2-1] Euler angles rotation shown in Figure 5.2 condensed in a single rotation using the appropriate quaternion

5.4.2. Quaternion to Euler angles conversion

The inverse conversion, expressing the equations that allow to express Euler angles as a function of the quaternion elements, requires the solution of the non-linear system of equations of Equation 5.26. These equations are highly non-linear and heavily dependant on the specific Euler angles set, making their solution non-trivial. Given this, here only the results for the specific [3-2-1] Euler angles set will be provided (a more general algorithm discussing the solution for any arbitrary sequence of rotation may be found in [74]).

Let the relative attitude of $\mathcal{B}$ with respect to $\mathcal{E}$ be determined by the quaternion $\mathbf{q}_{E2B}$. Consider that the same relative attitude may be obtained via a [3-2-1] rotation sequence with the following rotations angles: ψ around the z-axis of the first reference system $\mathcal{E}$, θ around the y-axis of the first reference system $\mathcal{B}_z$, ϕ around the x-axis of the last reference system $\mathcal{B}_y$ to obtain the final attitude $\mathcal{B}$. The equations that allow to find the Euler angles $[\,\phi,\,\theta,\,\psi\,]$ from the quaternion components $[\,q_0,\,q_i,\,q_j,\,q_k\,]$ are shown in Equations 5.27 to 5.29.

$$\phi = \text{atan2}\left(2\left(q_0 q_i + q_j q_k\right), 1 - 2\left(q_i^2 + q_j^2\right)\right) \tag{5.27}$$

$$\theta = -\pi/2 + 2\,\text{atan2}\left(\sqrt{1 + 2\left(q_0 q_j - q_i q_k\right)}, \sqrt{1 - 2\left(q_0 q_j - q_i q_k\right)}\right) \tag{5.28}$$

$$\psi = \text{atan2}\left(2\left(q_0 q_k + q_i q_j\right), 1 - 2\left(q_j^2 + q_k^2\right)\right) \tag{5.29}$$

Part IV

Model analysis

6

Analysis of the Bo105 model

This chapter will delve in a description of the Bo105 helicopter and of the available 6 DOF model, providing a complete overview of their characteristics and emphasising their qualities. In order to do so, the chapter will be divided in two major sections: Section 6.1 will delve in an analysis of the Bo105 linearizations, presenting the available state-space models and -noting that the attitude representation was incorrect in the original data- correcting the attitude representation part of the algorithm; finally, Section 6.2 will discuss the linearized models form a control theory standpoint, performing a stability and modes analysis of the system and discussing the system's controllability and observability.

Additional considerations regarding the model analysis of the Bo105 helicopter are found in Appendix A. In particular, in Section A.1, a description of the Bo105 helicopter from a mechanical and architectural standpoint is provided to better contextualize the interest in this helicopter for agile flight applications. Further, in Section A.2, a comparison between the pole maps of the available system-identified linearized models and of a theoretical model is provided, highlighting the complexity of the system-identified model.

6.1. Discussion of the linearized model

The linearized models available for this study will be here presented and discussed, detailing their characteristics and providing a discussion of their properties. To allow for an orderly discussion of the topic at hand, the original raw data for the model will be presented and discussed, including a discussion of the available states and linearization points. Afterwards, the presented data will be further analysed and processed, with small modifications and corrections made. With the required modifications and corrections performed, the model will be studied from a control theory standpoint, discussing crucial aspects of observability and controllability, as well as an eigenmode analysis.

6.1.1. Description of the linearized model

At the start of the thesis study, two system-identified linearized models of the Bo105 helicopter were made available at the following velocities: 0 and 65 kn (approximately 33 m/s). A description of these models - together with some of the data used for their implementation - is provided in the manuals [76, 77] and the general characteristics of the models will be reported here for completeness. These linearized models are a set of system-identified models developed by DLR Braunschweig and are provided only in the form of a pair of matrices, corresponding to the **A** and **B** matrices of a state-space system. Considering the general equation given in Equation 6.1 - where **x** indicates a vector containing the states of the system, **u** the system's inputs, and **y** the system's outputs - to complete the state-space system the remaining **C** and **D** matrices need to be defined.

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{cases} \tag{6.1}$$

For the purpose of this thesis, the state-space representation was considered to have full-state feedback, meaning that for all flight configurations **C** and **D** matrices were considered to be respectively and identity and a zero matrix of appropriate sizes: **C** is therefore set to be a $n_x \times n_x$ identity matrix and **D** and $n_x \times n_u$ zero matrix, with n_x being the number of states and n_u the number of inputs. With this consideration, it follows naturally that $\mathbf{y} = \mathbf{x}$. With the models defined, their analysis is carried out.

Figure 6.1: A Canadian Coast Guard MBB Bo105 over St. Lawrence River [75]

6.1.2. Model inputs, states, and outputs

The available Bo105 models are multiple input-multiple output (MIMO) systems with 12 states **x** and 4 inputs **u**. Table 6.1 shows the states (and outputs) of the system. Note that the system used in this analysis only details the rigid-body states and does not include the higher-order dynamics associated with the rotor: this absence, while it would reduce model accuracy, still allows for the development of a control system based on the study of the helicopter response in terms of its rigid body characteristics [78, 79], controlling the system's orientation and velocity.

Table 6.1: States and outputs of the linearized model

Description	Symbol	Unit
Surge velocity	u	[m/s]
Sway velocity	v	[m/s]
Heave velocity	w	[m/s]
Roll rate	p	[rad/s]
Pitch rate	q	[rad/s]
Yaw rate	r	[rad/s]
Roll angle	ϕ	[rad]
Pitch angle	θ	[rad]
Yaw angle	ψ	[rad]
Roll acceleration	$\dot{p}$	[rad/s^2]
Pitch acceleration	$\dot{q}$	[rad/s^2]
Yaw acceleration	$\dot{r}$	[rad/s^2]

Table 6.2 shows the inputs of the model. As the Bo105 is a conventional helicopter, the inputs vector is comprised of four commands corresponding respectively to the longitudinal and lateral cyclic, the collective and the pedal pilot commands. Of these, the first three commands interact with the main rotor (MR), changing the pitch of the blade by interacting with the feathering hinge via the swashplate. Complementing their action, the last command δ_p interacts with the blades of the tail rotor (TR), changing their pitch and as such impacting the moment around the helicopter's body z axis. All inputs are expressed as percentages of the total available deflection, and their allowable ranges are specified in the table:

Table 6.2: Inputs of the linearized model

Description	Symbol	Unit	Range	
Longitudinal cyclic	δ_x	[%]	-100% pulled	+100% pushed
Lateral cyclic	δ_y	[%]	-100% left	+100% right
Collective	δ_0	[%]	0% pushed down	+100% pulled up
Pedal	δ_p	[%]	-100% pushed left	+100% pushed right

As per the model description, it is also worth noting that the input vector is not the commanded deflection of the control surfaces, but instead the deflections of the pilot's control stick. While conventionally the actuator's dynamics would have to be introduced as a separate dynamic subsystem, with this configuration the dynamics of the actuators are already included in the model and no further modifications are required to account for them. Nonetheless, while these dynamics are not to be explicitly modeled, they are still implied in the model and will be described in Section 6.1.3.

6.1.3. Implied actuator dynamics

As discussed above, the available linearized models of the Bo105 helicopter already account for actuator dynamics, which are implied in the state-space representations. Even so, the dynamics of the helicopter's actuator represent an important part of the system's representation and, as such, their characteristics are briefly explained hereafter.

The relation between the pilot's stick deflections [δ_x, δ_y, δ_0, δ_p] and the actuator's demands [θ_0, θ_s, θ_c, θ_p] as described as per Equations 6.2 to 6.5, where the subscripts U and L indicate respectively the upper and lower deflections limits of the respective control surface defined in [76] to represent the actuator's allowed deflections (values are shown in Table 6.3).

$$\theta_0 = (\theta_{0U} - \theta_{0L})\frac{\delta_0}{100} + \theta_{0L} \tag{6.2}$$

$$\theta_s = (\theta_{sU} - \theta_{sL})\frac{\delta_x}{100} + \theta_{sL} \tag{6.3}$$

$$\theta_c = (\theta_{cU} - \theta_{cL})\frac{\delta_y}{100} + \theta_{cL} \tag{6.4}$$

$$\theta_p = (\theta_{pU} - \theta_{pL})\frac{\delta_p}{100} + \theta_{pL} \tag{6.5}$$

The dynamics of the control surfaces - according to [77] - may be defined as either first or second order transfer functions [76], shown respectively in Equations 6.6 and 6.7 and Equations 6.8 and 6.9, where τ indicates the first-order actuator time constant, while ε and ω_0 indicate respectively the second order damping factor and natural frequency (the values of these factors are found in Table 6.4).

$$\hat{\theta}_X = \frac{\theta_X}{\tau_{MR}\, s + 1} \qquad X = \{0, s, c\} \tag{6.6}$$

$$\hat{\theta}_p = \frac{\theta_p}{\tau_{TR}\, s + 1} \tag{6.7}$$

$$\hat{\theta}_X = \frac{\theta_X}{s^2 + 2\,\varepsilon_{MR}\,\omega_{0MR}\, s + \omega_{0MR}^2} \qquad X = \{0, s, c\} \tag{6.8}$$

$$\hat{\theta}_p = \frac{\theta_p}{s^2 + 2\,\varepsilon_{TR}\,\omega_{0TR}\, s + \omega_{0TR}^2} \tag{6.9}$$

In the context of the thesis, the models were already linearized and the first-order formulation was implemented in the model definition process. Notice that the first and second-order transfer functions map

similar frequency responses of the system, essentially acting as delays to the control actuator's response when manipulating the position of the swashplate and the tail rotor collective.

With the relation between the pilot's inputs and the actuator deflections clarified, the cyclic pitch at the blade can be obtained from a phase combination of the cyclic, collective and pedal values as per Equation 6.10 (where Ψ_0 is the cycling phasing angle).

$$\begin{bmatrix} \theta_{Bl,0} \\ \theta_{Bl,s} \\ \theta_{Bl,c} \\ \theta_{Bl,p} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Psi_0) & \sin(\Psi_0) & 0 \\ 0 & -\sin(\Psi_0) & \cos(\Psi_0) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_s \\ \hat{\theta}_c \\ \hat{\theta}_p \end{bmatrix} \tag{6.10}$$

With this, the relation between the helicopter's commands and the pitch at the blades has been clarified. These relations were used in the identification of the model and were naturally implied in the model made available at the start of the thesis.

Table 6.3: Upper and lower deflection limits

Description	Symbol	Value	Unit
Collective			
Upper limit	θ_{0U}	18.00	[deg]
Lower limit	θ_{0L}	2.00	[deg]
Longitudinal cyclic			
Upper limit	θ_{sU}	-5.50	[deg]
Lower limit	θ_{sL}	10.00	[deg]
Lateral cyclic			
Upper limit	θ_{cU}	4.00	[deg]
Lower limit	θ_{cL}	-6.00	[deg]
Pedal			
Upper limit	θ_{pU}	-6.00	[deg]
Lower limit	θ_{pL}	18.00	[deg]

6.1.4. Analysing and processing of the model data

The discussion carried out so far has presented the Bo105 helicopter and has introduced the characteristics of its available linearized models. With the linearized systems studied in terms of their inputs and outputs, the data of the linearized model was further analysed to better understand its dynamic characteristics. From this initial analysis, two observations were made which prompted the modification of the system's original **A** and **B** matrices: these modifications will be described and justified in this section.

The objective of these adjustments was to refine the matrix representation to better reflect the physical dynamics of the helicopter, ensuring the model's suitability for subsequent control analysis and design tasks. To this end, two modifications were implemented:

1. removal of the yaw rate derivative $\dot{r}$ and its associated dynamics.
2. modification of the attitude submatrix to incorporate trigonometric dependencies on the trim angles.

These refinements ensure that the linearized model captures the relevant dynamics accurately, excluding states that do not contribute meaningfully to the system's evolution while enhancing the precision of attitude coupling effects around the trim condition.

Removal of the $\dot{r}$ state

Upon analysis of the original A and B matrices, it was observed that the entries interacting with the yaw rate derivative state $\dot{r}$ were entirely populated with zeros, indicating that $\dot{r}$ had no effect on, nor was it affected by, other states or inputs in the linearized model at the specified operating points (hover and 65

Table 6.4: Control system data [76]

Description	Symbol	Value	Unit
Cyclic phasing angle	Ψ_0	10.0	[deg]
Main rotor actuator rate limit	λ_{MRlim}	180	[deg/s]
Tail rotor actuator rate limit	λ_{TRlim}	360	[deg/s]
First order assumption			
Main rotor actuator time constant	τ_{MR}	0.040	[s]
Tail rotor actuator time constant	τ_{TR}	0.020	[s]
Second order assumption			
Main rotor actuator natural frequency	$\omega_{0\text{MR}}$	80	[rad/s]
Main rotor damping factor	ε_{MR}	0.4	[−]
Tail rotor actuator natural frequency	$\omega_{0\text{TR}}$	120	[rad/s]
Tail rotor damping factor	ε_{TR}	0.5	[−]

knots). In state-space representation, a row and column of zeros in the state matrix generally imply that the respective state is dynamically decoupled from the system, meaning that it neither influences nor is influenced by the dynamics of other states in this configuration. Consequently, retaining $\dot{r}$ in the state vector introduces an unnecessary dimension to the system, complicating the model without contributing to its accuracy, while also hindering the controllability and observability of the state-space system since an internal dynamic structurally unrelated to the rest of the system is introduced.

To streamline the model and focus on the dynamics directly relevant to the helicopter's behaviour in hover and forward-flight conditions, the $\dot{r}$ state was removed from the **A** and **B** matrices. This operation entailed eliminating both the row and column associated with $\dot{r}$ in **A** and the respective row of the **B** matrix, resulting in a reduced-order system. By removing this extraneous state, the dimensionality of the **A** matrix is decreased to reflect only the essential dynamics, facilitating a more focused and efficient approach to control design.

Note that, as the size of the state and input matrices was changed by this transformation, this change should also be reflected on all other matrices as to keep the state-space system coherent. In particular, in an effort to maintain the full-state feedback characteristic mentioned above, the **C** and **D** matrices had to be redefined to be respectively and identity matrix of size $(n_x - 1) \times (n_x - 1)$ and a zero matrix of size $(n_x - 1) \times n_u$. As all following calculations in this text will consider this modified form of the state-space system, the values of n_x and n_u used hereafter will always refer to the number of states and inputs of the modified system respectively.

Modification of the attitude submatrix

When generating a linearized model, the rates of change of the attitude angles $[\dot{\phi}, \dot{\theta}, \dot{\psi}]$ are derived from kinematic relations starting from the trim values of the attitude Euler angles and the helicopter's body rates $[p, q, r]$. Letting $\bar{\phi}$ and $\bar{\theta}$ respectively be the trim point's roll angle and pitch angle, these angular rates are related to the body rates $[p, q, r]$ in proximity of the trim point by Equation 6.11:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\bar{\phi})\tan(\bar{\theta}) & \cos(\bar{\phi})\tan(\bar{\theta}) \\ 0 & \cos(\bar{\phi}) & -\sin(\bar{\phi}) \\ 0 & \sin(\bar{\phi})\sec(\bar{\theta}) & \cos(\bar{\phi})\sec(\bar{\theta}) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{6.11}$$

This relationship describes how the helicopter's body rates contribute to the changes in its attitude, and the matrix of Equation 6.11 (computed with the appropriate trim attitude angles) is included in the state

space model's **A** matrix to indicate the dynamics of the attitude states.

In the initial formulation of the **A** matrix, the submatrix representing these kinematic relationships was set as an identity matrix in both the hover and forward flight trim. This simplification caused the model to overlook the coupling between roll and pitch rates, leading to a misrepresentation of the dynamics near the trim point. This meant that changes in roll angle did not reflect any dependency on pitch orientation and vice versa, leading to errors in the predicted response of the helicopter's attitude, particularly in scenarios requiring precise control.

To address this issue, the above-mentioned identity matrix was replaced with the correct kinematic relationship that incorporates the relevant trigonometric terms. Specifically, the modified submatrix includes terms such as $\cos(\bar{\theta})$, $\sin(\bar{\phi})$, and $\cos(\bar{\phi})$, where $\bar{\theta}$ and $\bar{\phi}$ represent the trim values of the pitch and roll angles, respectively. These trigonometric terms reflect the influence of the aircraft's orientation on its attitude dynamics, accurately capturing the way in which the body rates p and q interact with the helicopter's current orientation. The corrected submatrices are non-dimensional and are shown in Equation 6.12 for hover and Equation 6.13 for 65 knots.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & -1.53 \cdot 10^{-5} & 6.73 \cdot 10^{-2} \\ 0 & 1 & 2.28 \cdot 10^{-4} \\ 0 & -2.28 \cdot 10^{-4} & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{6.12}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & -1.26 \cdot 10^{-4} & 5.22 \cdot 10^{-3} \\ 0 & 1 & 2.42 \cdot 10^{-2} \\ 0 & -2.42 \cdot 10^{-2} & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{6.13}$$

By replacing the identity matrix with the correct kinematic terms, the modified **A** matrices now accurately reflect the dependency of the attitude dynamics on the trim angles $\bar{\phi}$ and $\bar{\theta}$. This change improves the model's fidelity by incorporating the coupling effects that naturally arise between the pitch and roll dynamics, providing a more realistic representation of the helicopter's attitude around the available trim points. This modification is particularly beneficial for control analysis, as it ensures that the model more closely captures the helicopter's true response to control inputs, enabling more accurate prediction and tuning of control strategies around the trim condition.

6.2. Stability and controllability analysis of the plant

With the model discussed and the modifications applied to it detailed in Section 6.1, an analysis of the linearized model was performed to understand the eigenmodes of the problem and the characteristics of the system itself. To perform this analysis, this section will have the following structure: first Section 6.2.1 will deal with the modal analysis of the system, analysing the model by looking at its poles and zeros and providing an interpretation of the modes themselves; afterwards, Section 6.2.2 will study the controllability and observability of the model, discussing its effects on the system's properties.

6.2.1. Stability and modes analysis

Having access to the linearized state-space models of the system, an analysis of the stability of the plant can be performed by simply analysing its poles (i.e.: the eigenvalues of the **A** matrix). As a general condition for stability, a dynamic state-space system can be considered asymptotically stable if all the poles of the system have a negative real part [41]. The poles of the system at both available trim points are shown in Figure 6.2, with a detail of the modes shown in Figure 6.3 to better distinguish the poles around the origin: in this figure, the primary modes of the system were identified by analysing the eigenvector components and also comparing this model with the theoretical formulation identified by Padfield in [80] (see also Appendix A). Table 6.5 details all the eigenvalues for both linearizations.

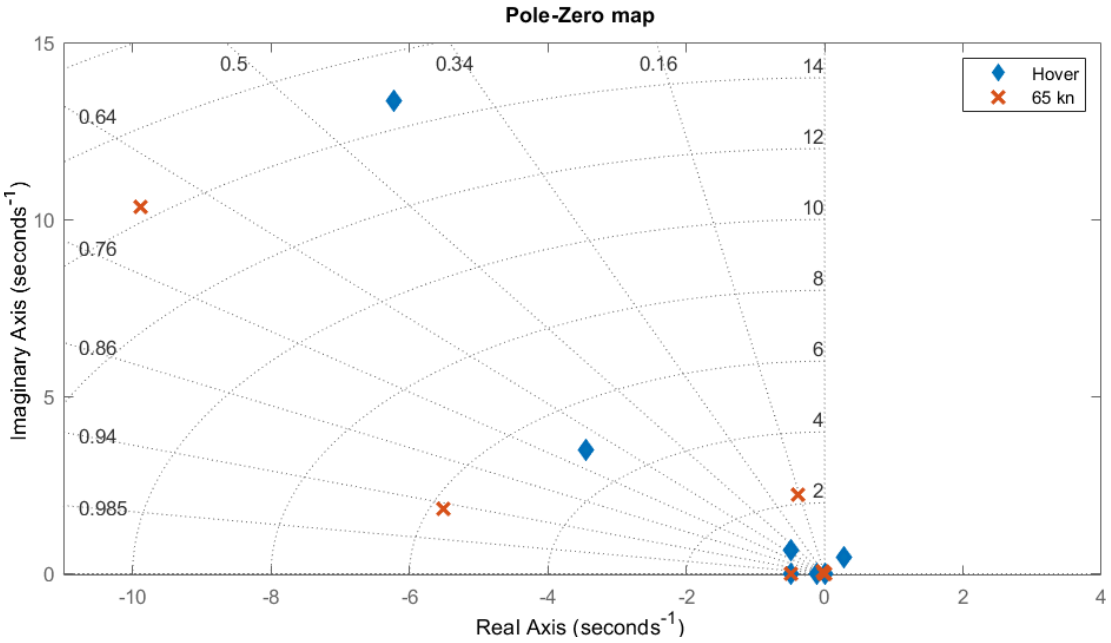


Figure 6.2: Pole map of the Bo105 model

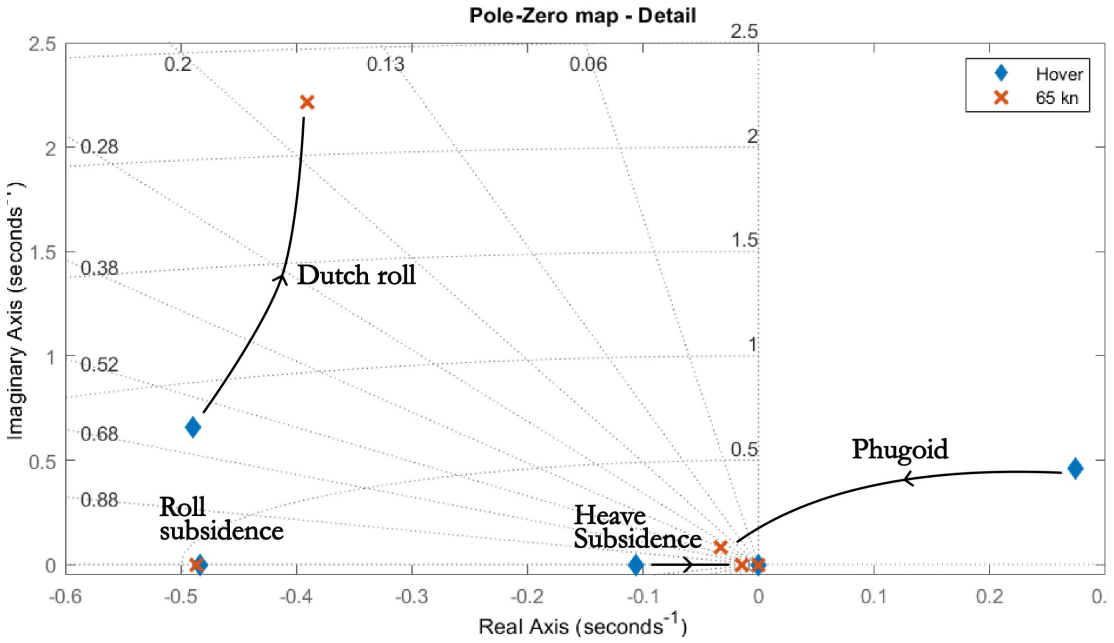


Figure 6.3: Pole-zero map of the Bo105 model (detail)

Immediately by looking at the graphs it can be seen that the hover model of the Bo105 is unstable as there is a pair of complex-conjugate poles with real part greater than 0. Conversely, the stability of the 65 kn model appears to be improved, as the root locus corresponding to the unstable complex-conjugate pair appears to have moved to have a negative real part. Note that - although the asymptotically unstable modes of the hover model become naturally stable in the forward flight configuration - the state-space system linearized at 65 kn can only be considered *marginally stable* because of the presence of a pole at the origin of the complex plane. To provide a proper interpretation of the modes of the system and what they represent, it is crucial to understand which states are primarily involved in their motions. To achieve this, an eigenvector analysis is appropriate.

Table 6.5: Eigenvalues of the hover and 65 kn linearizations

Hover [rad/s]	**65 kn** [rad/s]
0	0
-6.23+13.36i	-9.88+10.35i
-6.23-13.36i	-9.88-10.35i
-3.44+3.5i	-5.5+1.83i
-3.44-3.5i	-5.5-1.83i
-0.49+0.66i	-0.39+2.22i
-0.49-0.66i	-0.39-2.22i
0.27+0.46i	-0.03+0.08i
0.27-0.46i	-0.03-0.08i
-0.48	-0.49
-0.11	-0.01

The purpose of the eigenvector analysis is to identify the states that primarily contribute to the modes identified by each of the eigenvalues. As eigenvectors are generally vectors of complex numbers, they ought to be analysed by accounting for the effects of the real and imaginary parts: to do this, the elements of the eigenvectors associated with each eigenvalue have been expressed as a magnitude-phase pair. Let $\mathbf{v} \in \mathbb{C}^{n_x \times 1}$ be the eigenvector associated with the generic pole $\lambda \in \mathbb{C}$ such that:

$$\lambda \mathbf{v} = \mathbf{A}\mathbf{v} \tag{6.14}$$

then, each entry v_i of the eigenvector $\mathbf{v}$ can be expressed as a combination of two real values: the amplitude (or modulus) ρ_i and the phase angle (or - more simply - the phase) ϵ_i such that $v_i = \rho_i e^{i\epsilon_i}$. The magnitude and phase of each eigenvector entry can be defined as per the following equations:

$$\rho_i = \sqrt{\mathfrak{Re}(v_i)^2 + \mathfrak{Im}(v_i)^2} \tag{6.15}$$

$$\epsilon_i = \arctan\left(\frac{\mathfrak{Im}(v_i)}{\mathfrak{Re}(v_i)}\right) \tag{6.16}$$

With the magnitude and phase angles defined, the contribution of each state to the various modes can be quantified. Figures 6.4 and 6.5 show the results of the eigenvector analysis: each plot corresponds to a certain eigenvalue λ indicated at the top of the graph, the vertical blue bar correspond to the modulus of each entry, while the number written on top of each bar corresponds to the phase ϵ (expressed in degrees). The state corresponding to each modulus-phase pair is found on the x axis of each plot (note the absence of $\dot{r}$, as per Section 6.1).

Further observations can be made by comparing the identified pole loci of Figure 6.2 and Figure 6.3 with the eigenvector analysis. By comparing the two complementary information, a more complete understanding of the behaviour of the Bo105 helicopter model can be obtained. To start, from Figure 6.3 an initial understanding of the system's modes can be obtained. To start, the two complex-conjugate pole pairs can be identified to be the phugoid and the Dutch roll.

The phugoid, which is unstable in hover (with an eigenvalue of $0.27 \pm 0.46i$) but later becomes stable at 65kn (with an eigenvalue of $-0.03 \pm 0.08i$), is generally associated with an excitation of the u, w, and θ states (producing the longitudinal oscillatory motion typically associated with the phugoid); the eigenvector analysis of the identified model differs from the expected theoretical behaviour, with the largest contribution being in the surge u sway v velocity states when in hover and with a very large contribution in u at 65kn, suggesting that for this specific model a higher coupling between the longitudinal and lateral dynamics is to be expected. Analysing the contribution of the attitude angles, the largest activation is measured in θ and ψ, but their overall contribution to the eigenvector remains minimal. Additionally, phugoids in helicopter flight tend to become gradually more stable at moderate speeds, as is confirmed by theoretical models such as the Helisim [80]: although generally phugoids tend to remain unstable for helicopters, the

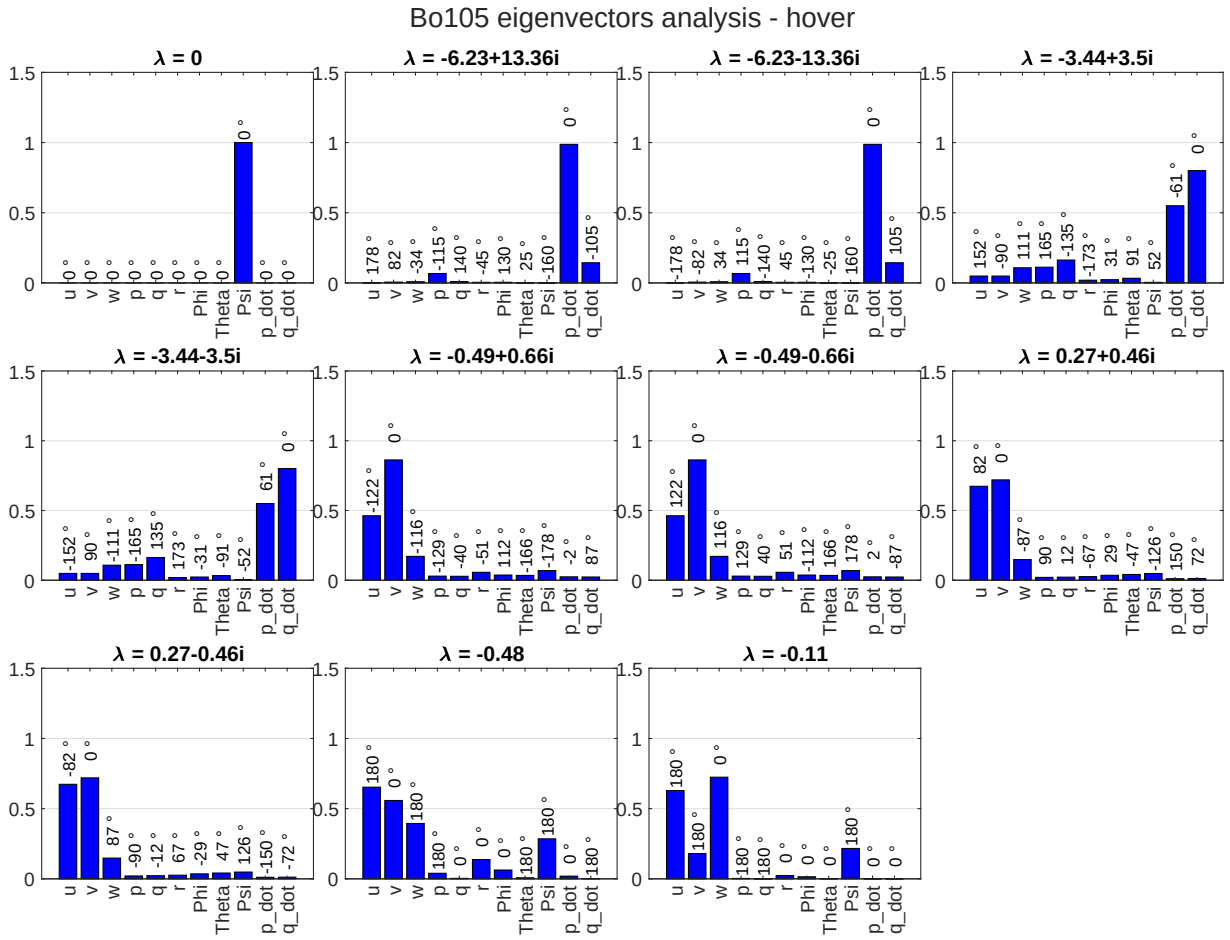


Figure 6.4: Eigenvector analysis: model trimmed at hover

stabilization of these poles in the 65kn linearization remains consistent with the theoretical behaviour of helicopter pole loci.

Another point of comparison between the two models is the Dutch roll mode, which is typically a stable lateral mode that is primarily identified by an out-of-phase combination of the yaw ψ and the bank ϕ angles. The modes of the identified model more closely matching the behaviour associated with the Dutch roll are eigenvalues $-0.49 \pm 0.66i$ in hover and $-0.39 \pm 2.22i$ at 65kn, which from the eigenvector analysis show large contributions of the surge and sway velocities u and v, alongside the expected out-of-phase participation of the roll and yaw attitude angles ϕ and ψ. As expected, this mode remains stable around the flight envelope.

By studying the results of the eigenvector analysis and comparing them with the pole-zero maps identified above it becomes immediately apparent that the two pole pairs with higher natural frequencies identified in Figure 6.2 primarily excite the rotational accelerations states $\dot{p}$ and $\dot{q}$: this is to be expected, as poles with higher natural frequency components will indicate higher-frequency dynamics, which is consistent with the rotational acceleration terms.

The eigenvector analysis also highlighted a clear interplay between the longitudinal and the lateral states for the available modes, suggesting a highly-coupled behaviour of the system, which will become more relevant in the controller development and manoeuvre simulation sections of the thesis. In particular, the three velocity channels seem to have significant out-of-phase coupling across all modes, with both lower and higher frequency modes showing significant cross-excitations.

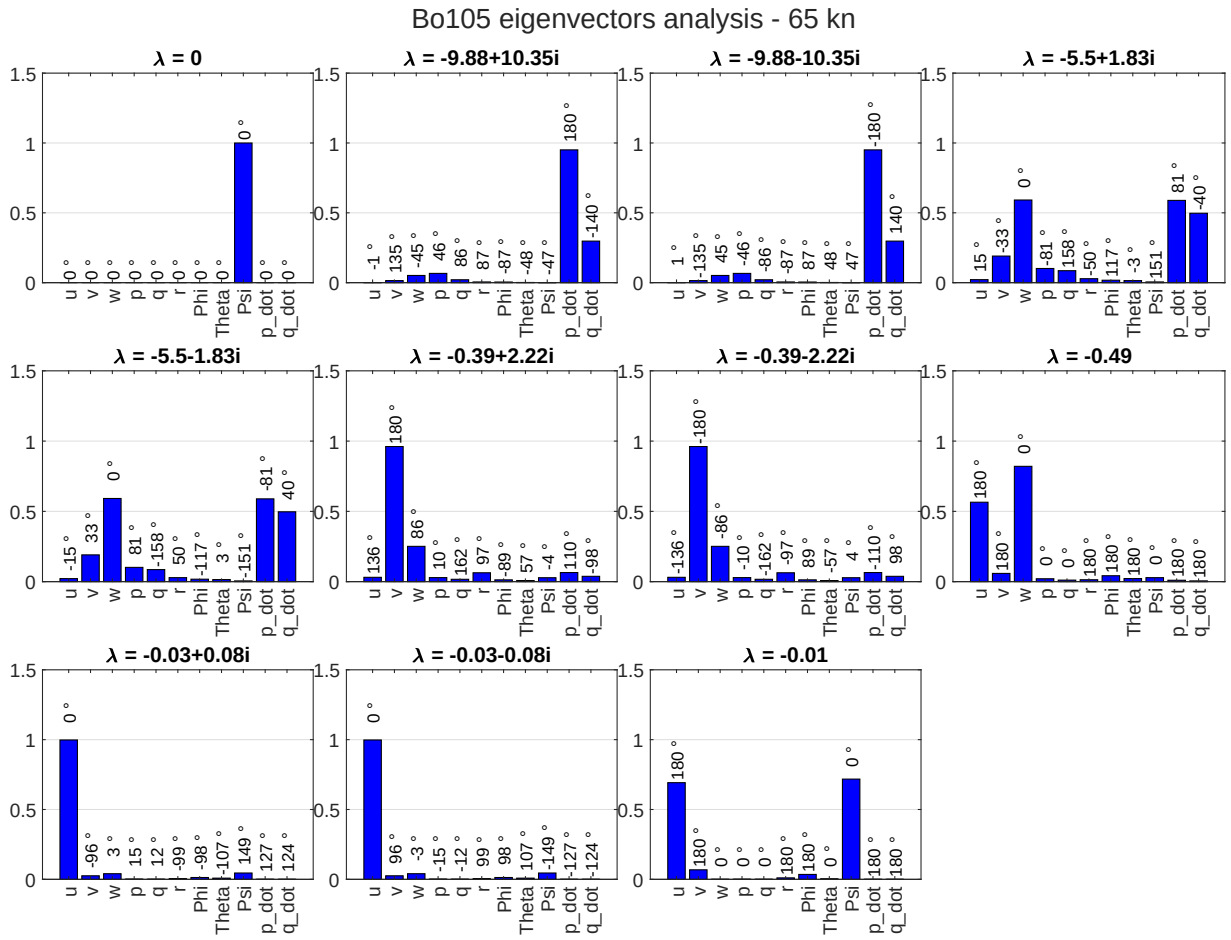


Figure 6.5: Eigenvector analysis: model trimmed at 65 kn

Overall, the pole loci of the available identified Bo105 models show some congruency with the expected behaviour of a typical helicopter, with complex pole pairs identifying the phugoid and the Dutch roll, and with other uncoupled modes encoding excitation on the heave and on the roll states. All modes show significant cross-excitations between the longitudinal and lateral states, suggesting complex behaviours and possible cross-activations, which will become more relevant in the study of the response of the system and the controller development and will be further discussed in Part V.

To expand upon the analysis of the modes of the system, in Section A.2 a comparison is carried out between the identified pole loci and modes of the identified system and a theoretical model of the Bo105 helicopter obtained using the Helisim general model [80], highlighting the similarities and differences between the modes of the systems.

With the system's linearized model defined and studied via a stability and modal analysis, other relevant characteristics of the linearized model - its controllability and its observability - will be discussed in Section 6.2.2, as they are fundamental information for the design of the control system and the characterization of the model.

6.2.2. Controllability and observability analysis

Another set of essential characteristics for the analysis and synthesis of a control system are the concepts of controllability and observability. These allow to determine whether the states of the helicopter can be effectively controlled and measured using available inputs and outputs, and the knowledge of these properties is instrumental in the choice of the appropriate control architecture.

Controllability - generally speaking - means that it is always possible to steer the states of a system from an arbitrary initial condition to an arbitrary final condition using the admissible sets of controls in a finite time [81, 82]. Whether a given state-space system is controllable or not can be evaluated by analysing the controllability matrix $\mathbf{M}_C(\mathbf{A}, \mathbf{B})$, defined as:

$$\mathbf{M}_C = \left[\mathbf{B} \,|\, \mathbf{AB} \,|\, \mathbf{A}^2\mathbf{B} \,|\, \ldots \,|\, \mathbf{A}^{n_x-1}\mathbf{B} \right] \tag{6.17}$$

where **A** and **B** are the states and inputs matrices of the state space system, and n_x is the number of states of the system. In particular, a state-space system is defined ”controllable” when the controllability matrix $\mathbf{M}_C$ has full row rank.

Observability, on the other hand, is generally defined to indicate the ability of all states of the system to be precisely identified from the information of the available system’s inputs and outputs [82]. Similarly to the controllability property, observability may be measured by studying the observability matrix $\mathbf{M}_O(\mathbf{A}, \mathbf{C})$:

$$\mathbf{M}_O = \begin{bmatrix} \mathbf{C} \\ \hline \mathbf{CA} \\ \hline \mathbf{CA}^2 \\ \hline \vdots \\ \hline \mathbf{CA}^{n_x-1} \end{bmatrix} \tag{6.18}$$

with **A** and **C** being the respective matrices of the state-space models. Similarly to controllability, a system is said to be ”observable” if the observability matrix has full column rank.

The procedure for evaluating controllability and observability described above was performed on the available linearizations to evaluate the system’s controllability and observability, showing that for all linearizations, the system is both controllable and observable. Other than contextualizing the linearized models in terms of their general characteristics, these characteristics are highly relevant for the development of an appropriate control system, as the fact that the system is both controllable and observable suggests the possibility of investigating Linear Quadratic control techniques, which are highly advantageous in the context of MIMO systems.

7

Conversion of the Bo105 model to the quaternion formulation

So far, the Bo105 aircraft has been introduced from a structural and mathematical standpoint, analysing the characteristics that make this specific helicopter fit for agile manoeuvering and then presenting the available system linearizations. In particular, the system has been discussed from a modelling and control-theory standpoint, describing it as a MIMO system with highly-coupled modes that is both controllable and observable. With the choice of the Bo105 helicopter for agile manoeuvering and its available linearized models described, to work towards a seamless integration of the available system with the quaternion attitude parametrization, it is essential for the models to be modified to make use of this parametrization instead of conventional Euler angles.

In an effort to allow for a seamless inclusion of the quaternion-based control system, in this chapter, the linearized models will be reviewed and modified such that their internal representation of the attitude also uses the quaternion formulation introduced in Chapter 5. With the process described hereafter, existing Euler angles-based models can effectively be converted to quaternion-based models, facilitating the study of possible quaternion controller formulations.

To discuss and motivate this procedure, the chapter will be divided in two major sections: to start, Section 7.1 will propose a methodology for the conversion of the model, starting from the process of linearization and appropriately manipulating the entries of the state-space matrices to have change the state representation; afterwards, Section 7.2 will discuss the validity of the results of the conversion, comparing the responses of the two systems to certain inputs and comparing the eigenvalues and eigenvectors to verify the correct conversion of the model.

7.1. Conversion of the model

This section will discuss the linearization logic and the modification of the state-space matrices to have the internal attitude representation be represented by means of quaternion elements $\mathbf{q}_{E2B} = [\, q_0,\, q_i,\, q_j,\, q_k \,]$ instead of Euler angles $[\, \phi,\, \theta,\, \psi \,]$.

To perform this task, it is first essential to consider what each entry of the state-space matrices represent, as to appropriately modify them to include the desired attitude representation. Considering $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ to be the general non-linear system of dynamic equations of the Bo105 helicopter and $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u})$ to be the system of output equations, the state-space matrices are defined as follows:

$$\mathbf{A} = \left.\frac{d\mathbf{f}}{d\mathbf{x}^T}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\begin{bmatrix} \dfrac{df_1}{dx_1} & \cdots & \dfrac{df_1}{dx_{nx}} \\ \vdots & \ddots & \vdots \\ \dfrac{df_{nx}}{dx_1} & \cdots & \dfrac{df_{nx}}{dx_{nx}} \end{bmatrix}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{7.1}$$

$$\mathbf{B} = \left.\frac{d\mathbf{f}}{d\mathbf{u}^T}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\begin{bmatrix} \frac{df_1}{du_1} & \cdots & \frac{df_1}{du_{nu}} \\ \vdots & \ddots & \vdots \\ \frac{df_{nx}}{du_1} & \cdots & \frac{df_{nx}}{du_{nu}} \end{bmatrix}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{7.2}$$

$$\mathbf{C} = \left.\frac{d\mathbf{g}}{d\mathbf{x}^T}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\begin{bmatrix} \frac{dg_1}{dx_1} & \cdots & \frac{dg_1}{dx_{nx}} \\ \vdots & \ddots & \vdots \\ \frac{dg_{ny}}{dx_1} & \cdots & \frac{dg_{ny}}{dx_{nx}} \end{bmatrix}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{7.3}$$

$$\mathbf{D} = \left.\frac{d\mathbf{g}}{d\mathbf{u}^T}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\begin{bmatrix} \frac{dg_1}{du_1} & \cdots & \frac{dg_1}{du_{nu}} \\ \vdots & \ddots & \vdots \\ \frac{dg_{ny}}{du_1} & \cdots & \frac{dg_{ny}}{du_{nu}} \end{bmatrix}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{7.4}$$

To explain the procedure followed to achieve this conversion, the following steps will be performed: first, the columns of the **A** and **C** matrices will be modified to allow for the substitution of the Euler angles in the state vector; then, the rows of **A** and **B** will be modified to include the time propagation equations for the quaternions. Note that - for the purpose of verification - the modifications applied will first of all aim at changing the state vector **x** but not the output vector **y** to later compare the outputs of the system to the same inputs. After the verification is complete, the **C** and **D** matrices will be re-included separately, and they will be kept to be respectively an identity matrix and a zero matrix of appropriate size to allow for full state feedback.

As an additional point of note, while the quaternions are composed of four elements, the modification of the state-space matrices will only aim at introducing three in the state-space vector. This is because of the presence of the additional internal unitary constraint shown in Equation 5.4: because of this internal constraint, were all four quaternion elements to be introduced in the state vector the system would loose its controllability property, as the states could not - in principle - be steered to achieve an arbitrary final configuration and the controllability property would therefore be hindered. With this in mind, the state-space vector will be modified to only include the quaternion vector elements $[\, q_i,\, q_j,\, q_k \,]$; the remaining element q_0 can always be reconstructed by applying the unitary norm condition.

To modify the columns of the **A** matrix to include the quaternion terms in the state vector, the matrix entries that need to be changed are those related to the Euler angles. In particular, the columns:

$$\left.\frac{d\mathbf{f}}{d\phi}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \qquad \left.\frac{d\mathbf{f}}{d\theta}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \qquad \left.\frac{d\mathbf{f}}{d\psi}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}}$$

These need to be changed to express the derivatives of the non-linear dynamic equations with respect to the quaternion elements. To this end, leveraging the chain rule and the linearity of the derivation operator, the new columns of the **A** matrices may be defined as follows:

$$\left.\frac{d\mathbf{f}}{d\,q_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\frac{d\mathbf{f}}{d\phi}\frac{d\phi}{d\,q_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\theta}\frac{d\theta}{d\,q_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\psi}\frac{d\psi}{d\,q_i}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{7.5}$$

$$\left.\frac{d\mathbf{f}}{d\,q_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\frac{d\mathbf{f}}{d\phi}\frac{d\phi}{d\,q_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\theta}\frac{d\theta}{d\,q_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\psi}\frac{d\psi}{d\,q_j}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{7.6}$$

$$\left.\frac{d\mathbf{f}}{d\,q_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} = \left.\frac{d\mathbf{f}}{d\phi}\frac{d\phi}{d\,q_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\theta}\frac{d\theta}{d\,q_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} + \left.\frac{d\mathbf{f}}{d\psi}\frac{d\psi}{d\,q_k}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \tag{7.7}$$

where the partial derivatives of the Euler angles with respect to the quaternion elements are derived from Equations 5.27 to 5.29. Note that to effectively apply this modification the derivatives of the Euler angles with respect to the quaternion elements have to be linearized around the trim point: to identify the

equilibrium points of the quaternion components, the following equation may be used:

$$\bar{\mathbf{q}}_{E2B} = \begin{bmatrix} \cos(\bar{\psi}/2) \\ 0 \\ 0 \\ \sin(\bar{\psi}/2) \end{bmatrix} \begin{bmatrix} \cos(\bar{\theta}/2) \\ 0 \\ \sin(\bar{\theta}/2) \\ 0 \end{bmatrix} \begin{bmatrix} \cos(\bar{\phi}/2) \\ \sin(\bar{\phi}/2) \\ 0 \\ 0 \end{bmatrix} \tag{7.8}$$

with $[\,\bar{\phi},\,\bar{\theta},\,\bar{\psi}\,]$ being the attitude at the trim point considered. With the columns of A modified as such, all the dependencies of the dynamic equations on the Euler angles $[\,\phi,\,\theta,\,\psi\,]$ have effectively been modified to instead show dependency on the states. These modification were also applied in an identical manner to the respective columns of the C matrix.

With these initial modifications applied, the shape of the state-space matrices is now the following:

$$\begin{bmatrix} \dot{u} \\ \vdots \\ \dot{r} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{p} \\ \vdots \\ \dot{r} \end{bmatrix} = \left[\begin{array}{ccc:ccc:ccc} \frac{df_1}{du} & \cdots & \frac{df_1}{dr} & \frac{df_1}{d\,q_i} & \cdots & \frac{df_1}{d\,q_k} & \frac{df_1}{d\,\dot{p}} & \cdots & \frac{df_1}{d\,\dot{r}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{df_{nx}}{du} & \cdots & \frac{df_{nx}}{dr} & \frac{df_{nx}}{d\,q_i} & \cdots & \frac{df_{nx}}{d\,q_k} & \frac{df_{nx}}{d\,\dot{p}} & \cdots & \frac{df_{nx}}{d\,\dot{r}} \end{array}\right]\Bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \begin{bmatrix} u \\ \vdots \\ r \\ q_i \\ q_j \\ q_k \\ \dot{p} \\ \vdots \\ \dot{r} \end{bmatrix} + \begin{bmatrix} \frac{df_1}{d\,\delta_x} & \cdots & \frac{df_1}{d\,\delta_p} \\ \vdots & \ddots & \vdots \\ \frac{df_{nx}}{d\,\delta_x} & \cdots & \frac{df_{nx}}{d\,\delta_p} \end{bmatrix}\Bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \begin{bmatrix} \delta_x \\ \vdots \\ \delta_p \end{bmatrix} \tag{7.9}$$

$$\begin{bmatrix} u \\ \vdots \\ r \\ \phi \\ \theta \\ \psi \\ \dot{p} \\ \vdots \\ \dot{r} \end{bmatrix} = \left[\begin{array}{ccc:ccc:ccc} \frac{dg_1}{du} & \cdots & \frac{dg_1}{dr} & \frac{dg_1}{d\,q_i} & \cdots & \frac{dg_1}{d\,q_k} & \frac{dg_1}{d\,\dot{p}} & \cdots & \frac{dg_1}{d\,\dot{r}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{dg_{nx}}{du} & \cdots & \frac{dg_{nx}}{dr} & \frac{dg_{nx}}{d\,q_i} & \cdots & \frac{dg_{nx}}{d\,q_k} & \frac{dg_{nx}}{d\,\dot{p}} & \cdots & \frac{dg_{nx}}{d\,\dot{r}} \end{array}\right]\Bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \begin{bmatrix} u \\ \vdots \\ r \\ q_i \\ q_j \\ q_k \\ \dot{p} \\ \vdots \\ \dot{r} \end{bmatrix} + \begin{bmatrix} \frac{dg_1}{d\,\delta_x} & \cdots & \frac{dg_1}{d\,\delta_p} \\ \vdots & \ddots & \vdots \\ \frac{dg_{nx}}{d\,\delta_x} & \cdots & \frac{dg_{nx}}{d\,\delta_p} \end{bmatrix}\Bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \begin{bmatrix} \delta_x \\ \vdots \\ \delta_p \end{bmatrix} \tag{7.10}$$

After having modified the columns of **A** and **C** appropriately, to fully include the quaternion elements as states the rows of the **A** and **B** matrices have to be modified to remove the Euler angles dynamic equations and instead include the time-propagation equations of the selected quaternion entries $[\,q_i,\,q_j,\,q_k\,]$. The time propagation equations are described in Equation 5.24: these have to be linearized around the trim

point, where the linearized values of the quaternion elements may be found as shown in Equation 7.8. Letting $\mathbf{h}(\mathbf{x}, \mathbf{u})$ be the function describing the time-propagation equations of the selected quaternion entries such that:

$$\begin{bmatrix} \dot{q}_i \\ \dot{q}_j \\ \dot{q}_k \end{bmatrix} = \mathbf{h}(\mathbf{x}, \mathbf{u}) = \frac{1}{2} \begin{bmatrix} q_0 & -q_k & q_j \\ q_k & q_0 & -q_i \\ -q_j & q_i & q_0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{7.11}$$

by introducing them in the **A** and **B** matrices from Equation 7.9, the final modified matrices are shown in Equation 7.12. In this final modified state-space system, the output equations remain the same as shown in Equation 7.10.

$$\begin{bmatrix} \dot{u} \\ \vdots \\ \dot{r} \\ \hdashline \dot{q}_i \\ \dot{q}_j \\ \dot{q}_k \\ \hdashline \dot{p} \\ \vdots \\ \ddot{r} \end{bmatrix} = \left[\begin{array}{ccc:ccc:ccc} \frac{df_u}{du} & \cdots & \frac{df_u}{dr} & \frac{df_u}{d\,q_i} & \cdots & \frac{df_u}{d\,q_k} & \frac{df_u}{d\,\dot{p}} & \cdots & \frac{df_u}{d\,\dot{r}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{df_r}{du} & \cdots & \frac{df_r}{dr} & \frac{df_r}{d\,q_i} & \cdots & \frac{df_r}{d\,q_k} & \frac{df_r}{d\,\dot{p}} & \cdots & \frac{df_r}{d\,\dot{r}} \\ \hdashline \frac{dh_i}{du} & \cdots & \frac{dh_i}{dr} & \frac{dh_i}{d\,q_i} & \cdots & \frac{dh_i}{d\,q_k} & \frac{dh_i}{d\,\dot{p}} & \cdots & \frac{dh_i}{d\,\dot{r}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{dh_k}{du} & \cdots & \frac{dh_k}{dr} & \frac{dh_k}{d\,q_i} & \cdots & \frac{dh_k}{d\,q_k} & \frac{dh_k}{d\,\dot{p}} & \cdots & \frac{dh_k}{d\,\dot{r}} \\ \hdashline \frac{df_{\dot{p}}}{du} & \cdots & \frac{df_{\dot{p}}}{dr} & \frac{df_{\dot{p}}}{d\,q_i} & \cdots & \frac{df_{\dot{p}}}{d\,q_k} & \frac{df_{\dot{p}}}{d\,\dot{p}} & \cdots & \frac{df_{\dot{p}}}{d\,\dot{r}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{df_{\dot{r}}}{du} & \cdots & \frac{df_{\dot{r}}}{dr} & \frac{df_{\dot{r}}}{d\,q_i} & \cdots & \frac{df_{\dot{r}}}{d\,q_k} & \frac{df_{\dot{r}}}{d\,\dot{p}} & \cdots & \frac{df_{\dot{r}}}{d\,\dot{r}} \end{array}\right]\Bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \begin{bmatrix} u \\ \vdots \\ r \\ \hdashline q_i \\ q_j \\ q_k \\ \hdashline \dot{p} \\ \vdots \\ \dot{r} \end{bmatrix}$$
$$+ \left[\begin{array}{ccc} \frac{df_u}{d\,\delta_x} & \cdots & \frac{df_u}{d\,\delta_p} \\ \vdots & \ddots & \vdots \\ \frac{df_r}{d\,\delta_x} & \cdots & \frac{df_r}{d\,\delta_p} \\ \hdashline \frac{dh_i}{d\,\delta_x} & \cdots & \frac{dh_i}{d\,\delta_p} \\ \vdots & \ddots & \vdots \\ \frac{dh_k}{d\,\delta_x} & \cdots & \frac{dh_k}{d\,\delta_p} \\ \hdashline \frac{df_{\dot{p}}}{d\,\delta_x} & \cdots & \frac{df_{\dot{p}}}{d\,\delta_p} \\ \vdots & \ddots & \vdots \\ \frac{df_{\dot{r}}}{d\,\delta_x} & \cdots & \frac{df_{\dot{r}}}{d\,\delta_p} \end{array}\right]\Bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \begin{bmatrix} \delta_x \\ \vdots \\ \delta_p \end{bmatrix} \tag{7.12}$$

7.2. Verification of the conversion

With the models appropriately modified to introduce quaternion elements as part of the state vector, the results of the conversion have to be verified to ensure the system's characteristics are not modified by the mathematical operations described above. This verification will be performed in two ways: Section 7.2.1 will compare the eigenvalues of the system and the system's controllability and observability properties, while Section 7.2.2 will compare the responses of the system to a set of simple inputs. By performing this verification, the model can be assessed to have been accurately converted and the implementation of the quaternion controller can be performed. Note that, for ease of understanding, in this section the Euler angles system will be referred to as SS_{Eul} and the converted system will be referred to as SS_{quat}.

7.2.1. Comparison of the systems' eigenvalues

Here, the eigenvalues and eigenvectors of the two state-space systems - the original Euler angles SS_{Eul} described and modified in Section 6.1 and the quaternion converted system SS_{quat} shown in Section 7.1 - will be compared. To achieve a correct conversion of the model, as the quaternion and the Euler angles parametrizations essentially codify the same information, the dynamics of the two systems should remain identical and, as such, the eigenvalues and their respective eigenvectors should not show significant changes.

The eigenvalues of the converted quaternion system SS_{quat} have been evaluated and are shown in Figure 7.2 for both the hover and 65 knots linearizations. The numerical values of the eigenvalues are also shown explicitly in Table 7.1. As an additional point of comparison, the eigenvectors for the respective systems can similarly be compared, with the results of the eigenvector analysis of the SS_{quat} state-space system shown in Figures 7.3 and 7.4.

Comparing the eigenvalues of the hover and 65 knots linearizations for SS_{quat} with those for SS_{Eul} shown in Figure 6.2 and Table 6.5, it can be noticed that the systems prior and after the conversion shows the same eigenvalues for both the linearizations, suggesting that the conversion is effective and has been performed correctly. The same considerations can also be done when comparing the eigenvectors of the systems at the different linearizations. Further, the fact that the eigenvector components corresponding to $[\phi, \theta, \psi]$ in the SS_{Eul} system match exactly the ones of $[q_i, q_j, q_k]$ in the SS_{quat} system suggests that the chosen quaternion elements have a good correspondence with the respective Euler angles.

The fact that eigenvalues and eigenvectors match before and after the conversion suggests that the procedure has successfully transitioned the system's internal attitude states to the quaternion parametrization without altering its intrinsic dynamic properties. This result serves as an important initial validation of the methodological robustness of the conversion process, and also confirms that the system's fundamental characteristics remain intact. To provide further validation of the proposed analytical process, in the next section the success of this conversion will be verified by comparing the responses of the original and converted systems to the same input signals, thereby demonstrating that the internal attitude parametrization has been preserved.

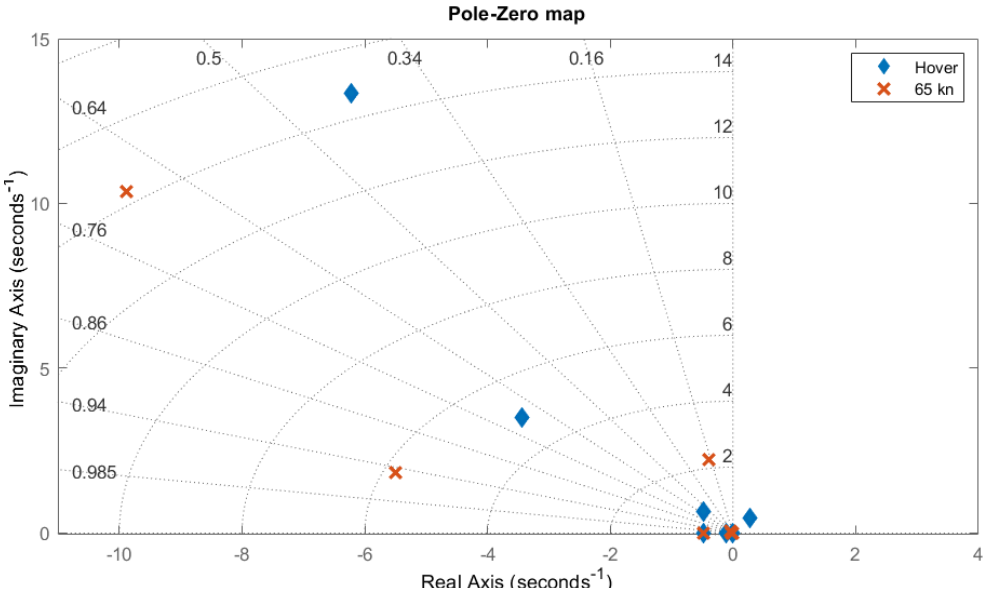


Figure 7.1: Pole-zero map of the SS_{quat} system: hover and 65 kn linearizations

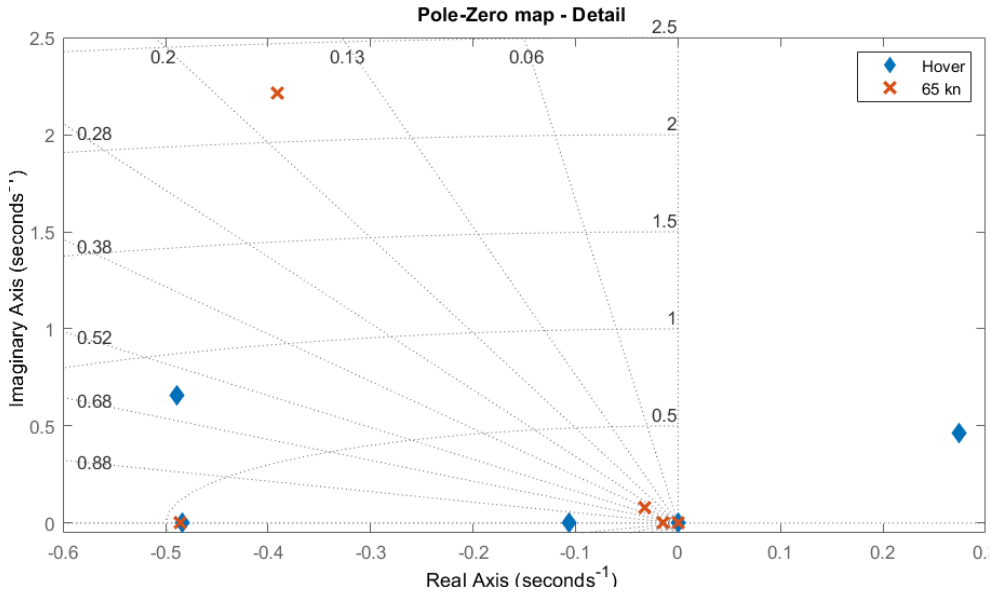


Figure 7.2: Pole-zero map of the SS_{quat} system: hover and 65 kn linearizations (detail)

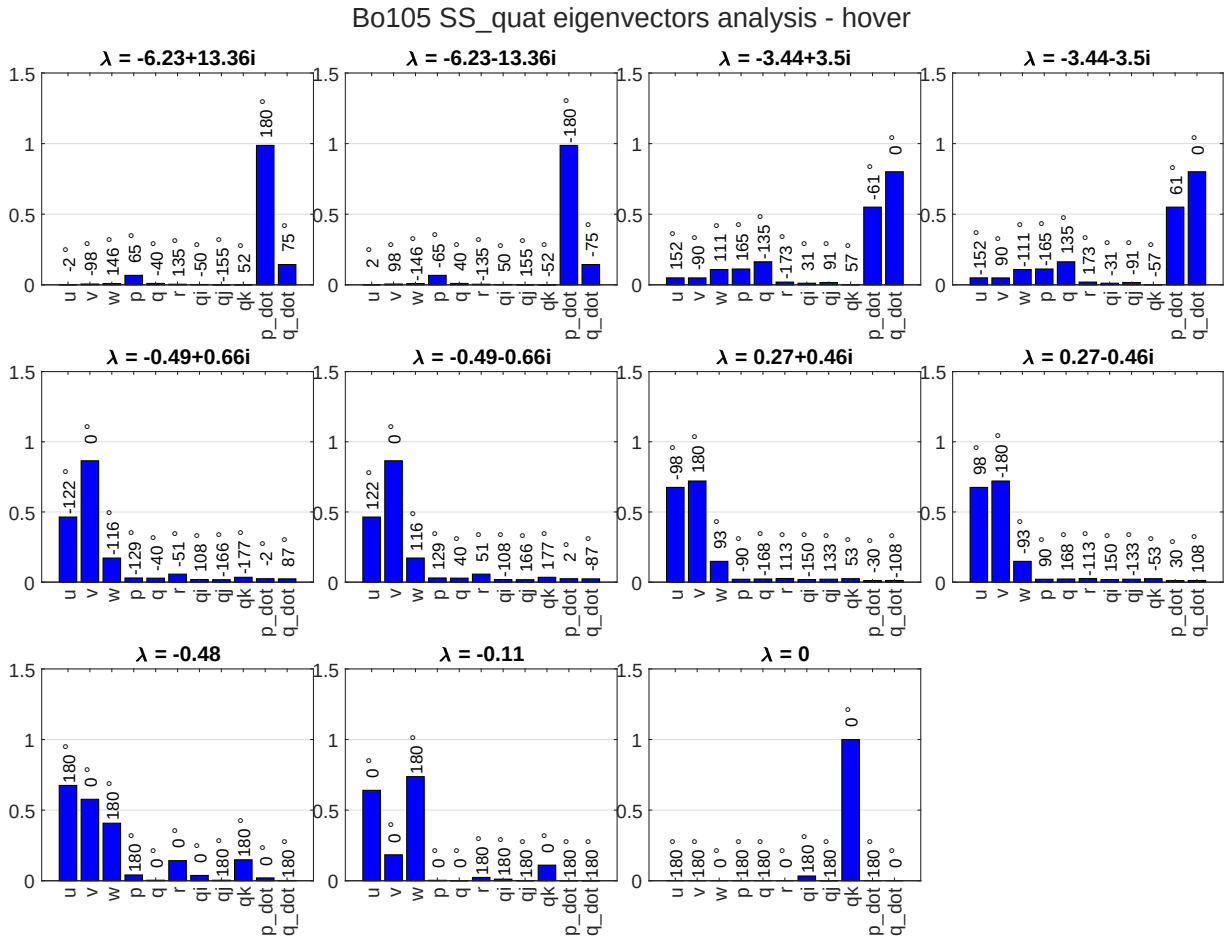


Figure 7.3: Eigenvector analysis of SS_{quat}: model trimmed at hover

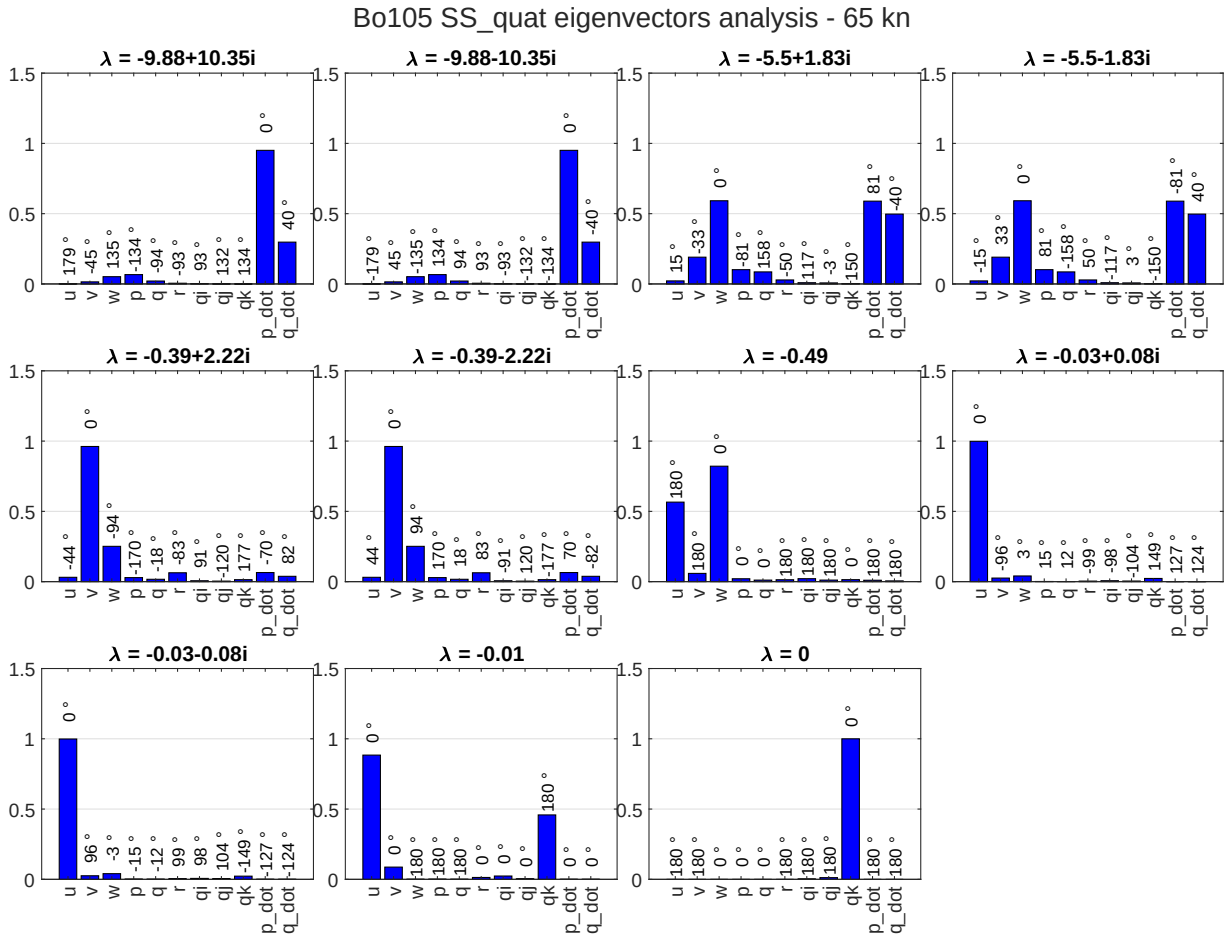


Figure 7.4: Eigenvector analysis of SS_{quat}: model trimmed at 65 kn

Table 7.1: Eigenvalues of the hover and 65 kn linearizations for SS_{quat}

Hover [rad/s]	**65 kn** [rad/s]
0	0
-6.23+13.36i	-9.88+10.35i
-6.23-13.36i	-9.88-10.35i
-3.44+3.5i	-5.5+1.83i
-3.44-3.5i	-5.5-1.83i
-0.49+0.66i	-0.39+2.22i
-0.49-0.66i	-0.39-2.22i
0.27+0.46i	-0.03+0.08i
0.27-0.46i	-0.03-0.08i
-0.48	-0.49
-0.11	-0.01

7.2.2. Comparison of the systems' response

After confirming that the eigenvalue spectra and corresponding eigenvector orientations of the state-space systems SS_{Eul} and SS_{quat} are congruent, the subsequent step involves verifying the practical implications of this equivalence. Within this section, the time responses of both models are compared when subjected to identical inputs. This analysis is pivotal, as it reinforces the accuracy of the conversion and assures that

the quaternion-based representation faithfully replicates the dynamic performance of the original Euler angle-based system, thereby setting the stage for appropriate controller evaluation.

To isolate and examine each input channel's impact on the system, a set of distinct square pulse signals was applied as inputs to the models. These pulse signals were designed to excite one control input channel at a time while keeping the other channels inactive. This selective excitation method allows for a clear interpretation of the system's response to each individual input, ensuring that any discrepancies between the Euler and quaternion models would be readily observable. Each input pulse was configured with the following characteristics:

- Pulse Duration: 6s
- Amplitude: 10%
- Initial delay: 2s
- Simulation duration: 10s

The amplitude was chosen to be an additional 10% deflection to command a small yet sufficient excitation to produce discernible dynamic responses without pushing the system into very non-linear behaviours. Further, given the trim points of the inputs shown in Table 7.2, the selected amplitude is adequate as it allows to keep the actuators far from the saturation point, improving the fidelity of the simulations.

Table 7.2: Trim points of the state-space inputs

Input	Hover trim point [%]	65 kn trim point [%]
δ_x	50.68	2.40
δ_y	34.82	5.17
δ_0	38.26	50.61
δ_p	45.83	-9.97

Given that both SS_{Eul} and SS_{quat} have four control inputs, four separate simulations were performed, each applying the square pulse to one of the inputs while maintaining the other three at zero. This testing approach ensures that each input channel's influence is isolated, making it easier to compare the responses of the two models on a per-channel basis. As an example of the provided inputs, Figure 7.5 shows the shape of the square pulse inputs described: in the figure, only the δ_x channel is activated, while the other three channels are kept inactive.

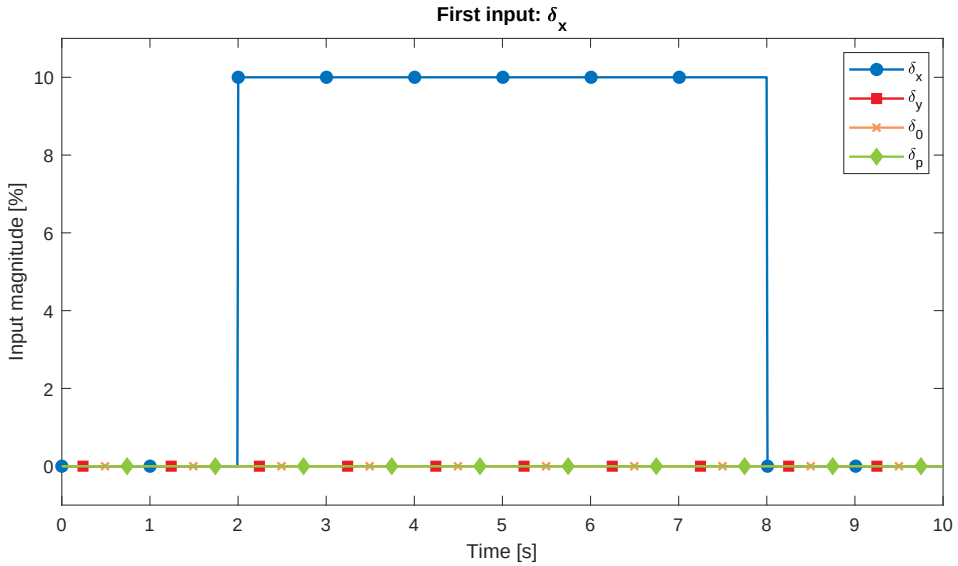


Figure 7.5: Example input activating the δ_x channel

For brevity and readability, the results of the above-mentioned simulations (as well as a graphical representation of the other inputs selected) are shown Appendix B. Here, only the responses to the input shown above will be presented to show the validity of the model, with Figure 7.6 and Figure 7.7 showing the time-responses of the SS_{Eul} and SS_{quat} systems, linearized in hover and at 65 knots respectively. The results reveal that the time responses of the two systems overlap exactly: this outcome confirms that the conversion to an internal quaternion attitude representation was executed successfully and that all dynamic characteristics have been faithfully maintained.

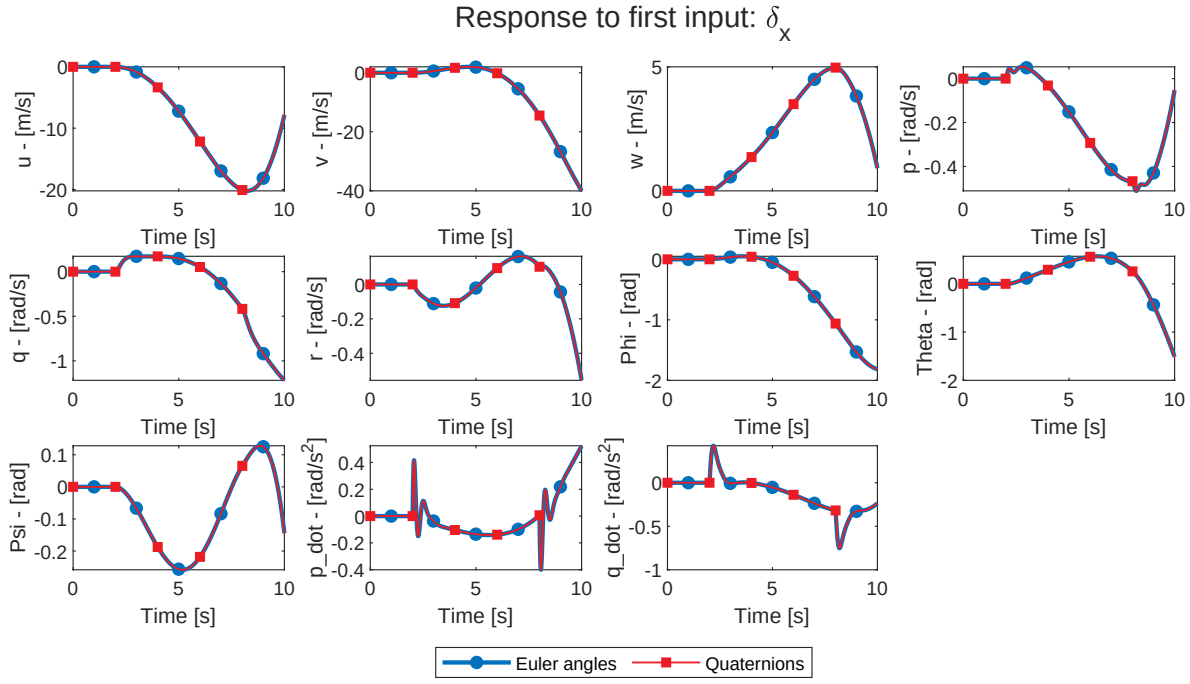


Figure 7.6: Time response of the hover state-space: input applied on δ_x

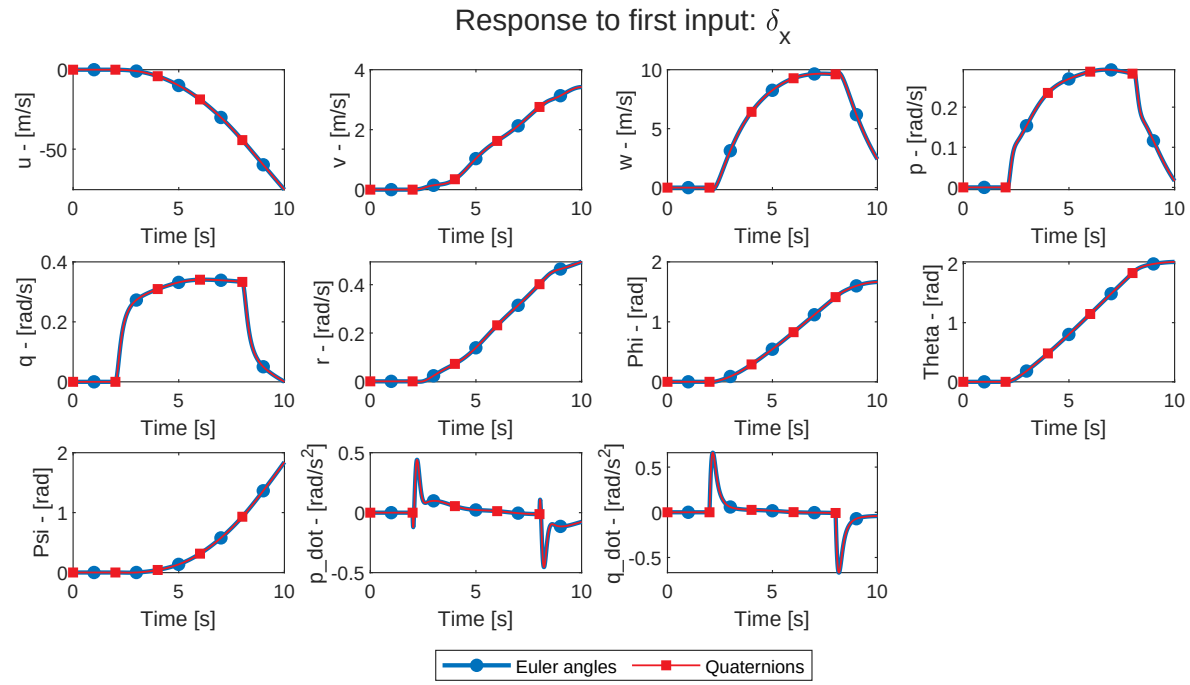


Figure 7.7: Time response of the 65 kn state-space: input applied on δ_x

7.3. Concluding remarks

This chapter has shown the conversion of the state-space system from an internal Euler angle parametrization of the attitude to a quaternion representation. This was achieved not by acting directly on the non-linear model or by re-defining it, but instead it was achieved by modifying the available linearization points and changing the internal configuration of the model. This procedure is inexpensive and shows good results, providing an effective and efficient way of modifying the state-space systems to achieve a conversion of the internal attitude parametrization.

The procedure developed to obtain such results is based entirely on mathematical considerations and as such is based entirely on analytical considerations without needing any experimental data (other than the data used for the identification of the already existing model). To further verify the correctness of the study, the original and the modified systems were compared for both the available linearizations in their modes and characteristics (such as their controllability and observability) and also in their dynamic response to selected inputs, showing good results that suggest the validity of the approach developed.

With the conversion to quaternions discussed in Section 7.1 and verified in Section 7.2, the rest of the report will only use the appropriately modified SS_{quat} system (which will be from now on simply referred to as "state-space model"): as the rest of the thesis aims at investigating the implementation of quaternion control for the purpose of autonomous agile manoeuvering, the system that will be considered is the linearization at 60 knots, to allow for a more accurate representation of the characteristics of the helicopter during flight. In particular, the **A** and **B** matrices will be the ones discussed in Equation 7.12, while - to maintain the full state feedback hypothesis made in Section 6.1 - the **C** and **D** matrices will now be reset to be an identity matrix and a zero matrix of appropriate sizes.

Part V

Controller development

8

Control system architecture

With the model studied and augmented in Part IV, to properly evaluate the capabilities of quaternion-based control, a control system was constructed to allow the helicopter to track references in attitude, velocity, and position appropriately. This chapter aims at presenting and discussing the developement of the control system architecture, providing an explanation of its individual components, and detailing their contribution to the achievement of the desired tracking performance.

To provide a comprehensive discussion of the control system, this chapter is divided as follows. To start, Section 8.1 will provide a general overview of the control system architecture, highlighting the inputs and outputs of each major block to provide a high-level understanding of how each subsystem interacts with the rest of the system. Following this, Section 8.2 will discuss in greater detail the attitude controller architecture, justifying the choice of the LQI framework and providing a more detailed view of its inner workings. With the attitude controller presented, the velocity and heave controllers will be discussed in Section 8.3, showing how they contribute to the generation of appropriate control signals and attitude references. Finally, Section 8.4 will conclude the presentation of the control system architecture by presenting the structure of the outer-loop position controller.

8.1. Control structure overview

The control architecture selected for the Bo105 helicopter is structured as a cascade of three nested loops, as illustrated in Figure 8.1. This allows for a combination of desired feedforward signals (identified by the * superscript) produced by the autopilot with corrective signals (identified by the $_c$ subscript) produced by previous loops of the control system, thus enabling the obtainment of a faster response while integrating the online-calculated corrective signals to ensure the model follows a desired trajectory.

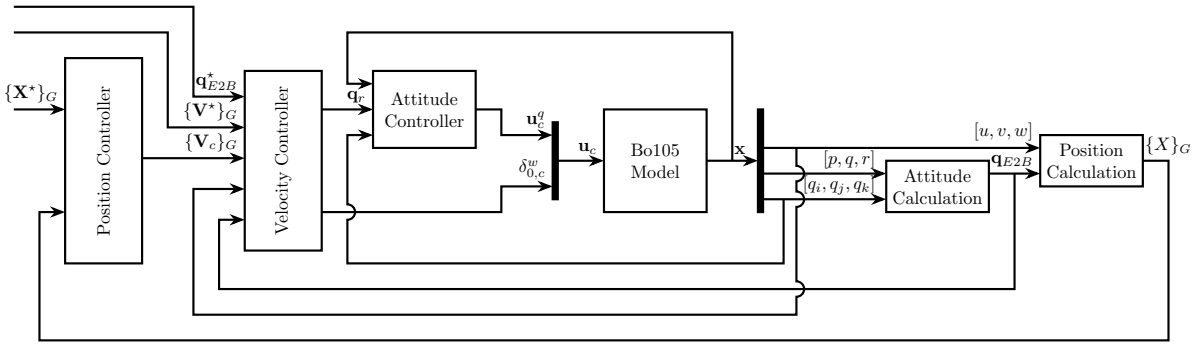


Figure 8.1: Block diagram of the complete control system

To better understand the overall controller architecture and the role of each block in controlling the system, a brief explanation of the components is provided hereafter, starting from the innermost loop and moving outwards.

The innermost loop consists of an attitude controller that regulates the helicopter's orientation $\mathbf{q}_{E2B}$ to track a reference attitude $\mathbf{q}_r$, which is determined by the velocity controller. To follow this attitude, the controller commands deflections in the helicopter's longitudinal and lateral cyclic $[\delta_x, \delta_y]$ and in the pedal δ_p. The innermost control loop is typically more exposed to the couplings of the plant, as it is in direct contact with it. Because of this reason, the innermost loop will be developed using an optimal control technique, the Linear Quadratic Integral (LQI), as it allows for a structured way of developing controllers for MIMO systems and naturally enables an attitude tracking behaviour.

The second loop is composed of the velocity controller, which performs two concurring tasks: horizontal velocity control and altitude control. The horizontal velocity controller is tasked with ensuring that the helicopter's horizontal velocity can track a reference horizontal velocity signal by generating an appropriate reference attitude $\mathbf{q}_r$ for the inner loop. The altitude controller is instead tasked with ensuring the vertical velocity of the helicopter tracks an appropriate reference signal by directly commanding the collective δ_0. For both controllers, the generation of the appropriate reference values and outputs is determined by combining the feed-forward reference signals for the attitude ($\mathbf{q}^{\star}_{E2B}$) and the velocity ($\mathbf{V}^{\star}$) provided by the autopilot module with a velocity correction signal ($\mathbf{V}_c$) produced by the outer loop position controller. The velocity controller will be based on the Proportional-Integral architecture, as it is a widely used controller technique that is flexible and capable of reducing steady-state errors.

The outermost loop is composed of a position controller, which takes as inputs the helicopter's current position $\mathbf{X}$ (computed by the "Position Calculation" block) and the desired position value $\mathbf{X}^{\star}$ (provided by the autopilot). This last control loop will be constructed using a set of simple on-axis proportional controllers, enabling the generation of representative velocity correction signals without requiring unnecessary integrator-like behaviours that would needlessly slow the system's response.

With a high-level understanding of the control system architecture now provided, the following sections will discuss in more detail the functioning of the individual controller blocks and loops, detailing their structures with the aid of appropriate block diagrams and motivating the approach used for the definition of each loop's controllers.

8.2. Attitude controller architecture

The first loop directly acting on the helicopter model is the attitude loop, which has the purpose of generating appropriate command signals along the helicopter's input channels $\mathbf{u} = [\delta_x, \delta_y, \delta_0, \delta_p]$ such that the helicopter's attitude $\mathbf{q}_{E2B}$ will follow a certain reference attitude $\mathbf{q}_r$. Given that the main helicopter model is a MIMO system and considering the controllability and observability characteristics identified in Section 7.2 and accounting for the need of a reference tracking action while stabilizing the system, the controller structure chosen is the Linear Quadratic Integral (LQI).

To properly introduce and justify the choice of this architecture, Section 8.2.1 will first highlight the reasons behind the choice of the LQI controller structure, emphasising its ability to handle MIMO systems while providing good stability margins. Then, Section 8.2.2 will briefly explain the mathematical foundations of the LQI controller architecture, discussing its relation with the more commonly used Linear Quadratic Regulator (LQR) and discussing its implementation in this specific case.

8.2.1. Motivation of the LQI architecture

The choice of the LQI structure has a number of significant advantages, which make it very desirable for the control of complex and coupled systems. Here, the choice of the LQI architecture for the attitude inner loop is motivated by providing a concise description of three fundamental beneficial characteristics of this approach: inherent decoupling, guaranteed robustness margins, and the presence of an integral action.

To start, LQI control is well-suited for controlling MIMO systems as it inherently balances control effort and system performance while handling the coupling between multiple inputs and outputs [83, 84]. This is especially useful in this case, as helicopters are heavily coupled systems which often require the addition of cross-fed signals in the controllers or of a control allocation module to obtain good performance. By means of the LQI approach in the innermost loop, the attitude controllers computed are capable of inherently

decoupling the system's motion, facilitating the augmentation of the controller via the introduction of additional outer loops.

Additionally, LQI controller architectures have the advantage of producing controllers which can guarantee good phase and gain margins even in the case of MIMO systems [85], providing a measure of robustness which other control architecture would not be able to naturally obtain. This characteristic is especially useful in the innermost loops, which are in direct contact with the plant and are tasked with controlling the vehicle's faster dynamics: by guaranteeing good stability margins at the plant, the inner loop provides a stable and responsive foundation that enhances the overall robustness and performance of the control architecture [86, 84]. This ensures that rapid changes and disturbances are effectively managed at the lowest level, allowing the outer loop to focus on higher-level objectives - such as trajectory tracking and mission-specific tasks - without being destabilized by internal oscillations or delays.

Further, compared to the more commonly used LQR control architecture, the added integral action of the LQI compensates for constant disturbances and model uncertainties, making it reliable for complex control applications in which reference-tracking manoeuvres have to be performed. The integral action extends the state-space by incorporating an additional state for the accumulated error (as explained in Section 8.2.2), reducing steady-state errors and ensuring that the system can maintain accurate reference tracking even during longer and more complex manoeuvres.

In light of the reasons discussed above, the LQI is a highly desirable control system architecture for the innermost attitude loop of the system, as it allows for decoupling of selected control signals while guaranteeing good stability margins and having an improved tracking action thanks to the integral action on selected signals. With the choice of the LQI architecture now justified, a mathematical description of the LQI control methodology will be provided in Section 8.2.2.

8.2.2. Description of the LQI implementation

Here a mathematical description of the LQI framework is provided, detailing the construction of the controller and the fundamental motivations behind its functioning. Note that, as this controller is applied on a linearized model, all variables discussed in this section are actually variations on the trim values: for the purpose of readability, here the notation Δ will be omitted.

To provide a clear explanation of the process used in developing the LQI controller, this section will first explain the baseline modifications applied to the original quaternion state-space system presented in Equation 7.12 to obtain the formulation shown in Equation 8.4. With the model modification applied, the LQI cost minimization method is briefly introduced and the equations to determine the controller **K** are presented. Concluding this section, a graphical representation of the complete integrated closed-loop system is presented in Figure 8.3 to clearly show the introduction of the LQI controller. Notice that here only a description of the controller implementation is provided. The tuning procedure and its results are explained in Section 9.2

As discussed previously, LQI is essentially a natural extension of the more commonly used LQR framework [87]: while the latter method is based on the use of a full-state feedback controller to achieve state regulation, LQI pairs this proportional action with an integral tracking action to ensure that a specially computed performance signal $\mathbf{y}_r(t)$ tracks a user-defined reference signal $\mathbf{r}(t)$, where the performance signal is defined as a linear combination of the base system's states.

$$\mathbf{y}_r = \mathbf{C}_r \mathbf{x} \tag{8.1}$$

Given that the purpose of the attitude controller is ensuring that the attitude quaternion elements included in the state vector $[q_i, q_j, q_k]$ track appropriate reference values, the matrix $\mathbf{C}_r$ was computed such that:

$$\mathbf{y}_r = \begin{bmatrix} q_i \\ q_j \\ q_k \end{bmatrix} \tag{8.2}$$

To effectively ensure that the performance signal $\mathbf{y}_r$ tracks a user-defined reference signal **r**, LQI aims at introducing a performance metric $\mathbf{x}_i$ as an additional state of the system: by then applying the LQR framework to this augmented system, the regulatory action will ensure that the new error state is driven to zero, allowing for proper tracking of a reference signal [41, 88]. To this end, the performance metric presented above has the following equation:

$$\mathbf{x}_i(t) = \int_{t_0}^{t} (\mathbf{r}(\tau) - \mathbf{y}_r(\tau))\, d\tau \tag{8.3}$$

As this new state essentially represents the integral of the tracking error, by regulating it to 0 the LQI controller is capable of ensuring reference tracking while maintaining the guaranteed stability margins of the traditional LQR controller.

Another point of note for the development of the attitude controller, as discussed in the previous sections, is the fact that the attitude loop is only tasked with modifying the helicopter's pedal and cyclic commands to attain the desired body orientation in space, while the task of maintaining the desired vertical speed is assigned to the velocity controller by means of a direct feed-forward to the collective control. To achieve this, in addition to the inclusion of $\mathbf{x}_i$, another modification was applied to the baseline system presented in Equation 7.12 is the removal - for the purpose of tuning - of the the collective input δ_0 and the heave velocity state w by appropriately removing the respective rows and columns of the A and B matrices. By applying this modification, the controller computed using LQI will not drive any additional control effort towards the regulation of the heave state and will only have access to the longitudinal and lateral cyclic commands and the pedal.

Applying both modifications to the system, the resulting augmented state-space model has the following state-space representation:

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{x}}_i(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C}_r & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{x}_i(t) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \mathbf{u}(t) + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{r}(t) \tag{8.4}$$

where **A** and **B** are the state and input matrices of the original system given in Equation 7.12 that have had the heave state and the collective control input removed, and $\mathbf{C}_r$ is the matrix used in Equation 8.1 to isolate the attitude states to be controlled.

A graphical representation of the system augmentation described above is shown in Figure 8.2. It is shown that the resulting system has two external inputs: **u** representing the control inputs to the system, and **r** representing the generated reference values. Matrices **B** and **A** are the state and input matrices of the baseline system, while $\mathbf{C}_r$ is the output matrix described in Equation 8.1.

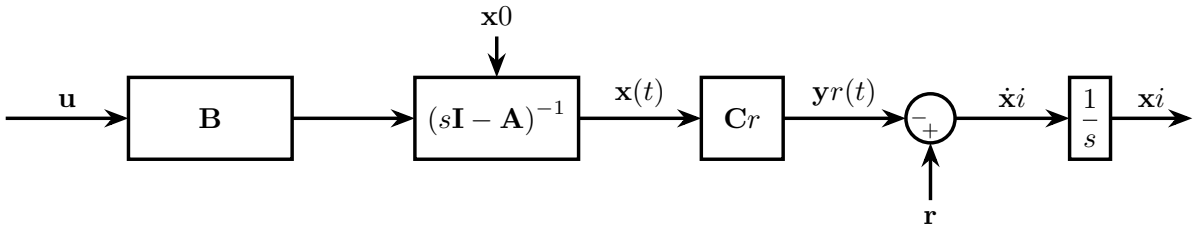


Figure 8.2: Block diagram of the augmented system

Note that the system here described was also analysed in terms of its controllability and observability in a method analogous to the ones discussed in Section 6.2: the results showed that the system maintained its controllability even after the transformation, meaning that the LQR control framework could indeed be applied to the augmented system [41, 89].

With the system modified as discussed, a controller can be produced by applying the LQR method to the augmented system. This revolves around the minimization of a Quadratic Performance Index (QPI), a convex cost function described by the equation:

$$J = \int_0^\infty \left(\mathbf{x}(\tau)^T \mathbf{Q}\mathbf{x}(\tau) + \mathbf{u}(\tau)^T \mathbf{R}\mathbf{u}(\tau)\right) d\tau \tag{8.5}$$

where **x** represents the augmented system's state vector, **u** represents the augmented system's input vector, and **Q** and **R** are user-defined performance matrices that aim at balancing the control system's performance and control effort: in particular, **Q** is a symmetric positive semi-definite matrix that penalizes deviations from the trim state and **R** is a symmetric positive-definite matrix that penalizes large control activation. By selecting appropriate values for **Q** and **R** the performance of the closed-loop system can be modified ensuring a good balance between control effort and performance. By selecting **Q** and **R** to have these characteristics, the optimization problem shown in Equation 8.5 is well posed and allows for the identification of an optimal stabilizing feedback controller **K**.

With the optimization problem described above, the equation for the controller **K** that can minimize the QPI is [90, 91]:

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{8.6}$$

where **P** is the solution to the Algebraic Riccati Equation (ARE):

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = 0 \tag{8.7}$$

The controller **K** obtained with this procedure is then included in the augmented control system via negative feedback. Notice that, to emphasize the presence of the integral action, the controller will be represented as $\mathbf{K} = [\mathbf{K}_x, \mathbf{K}_i]$ [87].

$$\mathbf{u} = -[\mathbf{K}_x, \mathbf{K}_i] \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_i \end{bmatrix} \tag{8.8}$$

While $\mathbf{K}_x$ focuses on immediate dynamic performance (ensuring the system responds promptly), $\mathbf{K}_i$ serves as a corrective mechanism that eliminates steady-state errors. This separation simultaneously allows the achievement of a good transient response and long-term accuracy, which is particularly beneficial in applications such as helicopter attitude control, where both quick responsiveness and precise steady-state tracking are critical. The controlled system with the LQI attitude control loop implemented is shown in Figure 8.3.

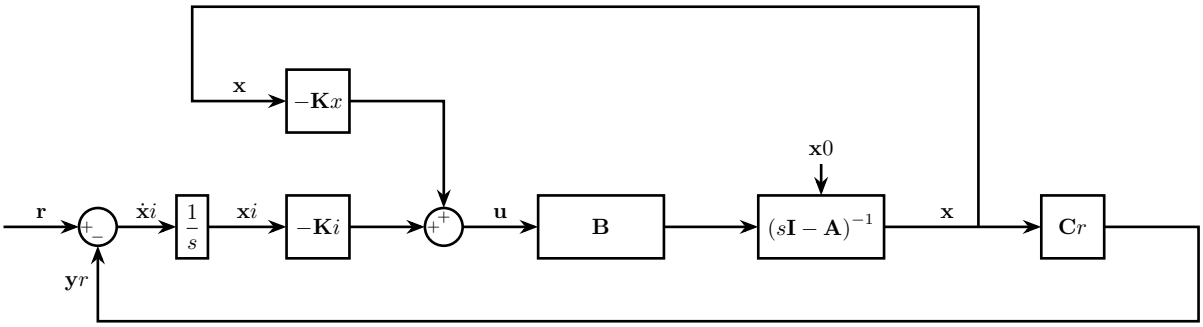


Figure 8.3: Block diagram of the controlled augmented system

8.3. Velocity ad heave controller architecture

Building upon the attitude controller implemented as discussed in Section 8.2, an additional outer loop aimed at controlling the velocity signals of the helicopter is developed.

Here an overview of the velocity controller is provided: first, in Section 8.3.1 the use of Proportional-Integral (PI) controllers is motivated, leveraging their flexibility and seamless integration with the inner attitude loop; afterwards, a description of the implementation is provided in Section 8.3.2, showing the control architecture of the system with this additional loop by means of an appropriate block diagram.

8.3.1. Motivation the PI architecture

To take advantage of the stability and robustness provided by the inner LQI attitude loop, this second loop makes use of PI controllers, leveraging their simplicity and widespread use in practical application. Here, the choice of the PI controller framework is justified.

The decision to employ PI controllers in the velocity loop is primarily driven by their simplicity and flexibility, which make them attractive for practical applications, as they are relatively easy to design and tune. Because of these characteristics, PI controllers find many practical applications in all engineering fields, making them a preferrable option over more complex design frameworks.

Further, PI controllers are especially useful when designing tracking controllers, as they are capable of providing both an immediate correction through their proportional term while also integrating errors over time via their integral action: this dual action is essential for compensating for persistent disturbances and steady state errors, which might otherwise lead to sustained velocity errors causing deviations from the desired trajectory

Moreover, the hierarchical control structure paired with the PI's flexibility allow to fully leverage the reliable and robust performance of the inner attitude loop, which is designed using an LQI controller. By using the control architecture here presented, the velocity controller is capable of integrating seamlessly with this robust inner loop, ensuring that any adjustments in velocity are accurately translated into corresponding attitude changes. This integration results in a control system that is both resilient to disturbances and capable of precise, responsive behavior.

Overall, the PI framework is a simple and effective solution, capable of achieving good tracking response while having a simple architecture that can interface seamlessly with the previously-designed LQI attitude controller: in light of these reasons. With the choice of the PI controller architecture now justified, an in-depth description of the velocity controller implementation is discussed in Section 8.3.2.

8.3.2. Description of the controller implementation

As described in Section 8.1, this second loop can be divided in two parallel structures: a velocity controller which aims at controlling the helicopter's surge and sway velocities $[u, v]_B$ by generating an appropriate reference attitude corrective signal $\mathbf{q}_c$, and a heave controller controlling the helicopter's heave velocity w_B to maintain altitude.

This differentiation in the controls is primarily inspired by the approach used by human pilots when controlling a helicopter: when human pilots wish to adjust horizontal velocity, they typically alter the helicopter's attitude to re-orient the main rotor's thrust; conversely, adjustments in vertical velocity are generally achieved by modifying the collective pitch directly.

A graphical representation of the interplay between these two controllers is provided in the block diagram representation of Figure 8.4. In the figure, it can be seen that the reference velocity signals $\mathbf{V}_r = [U_r, V_r, W_r]$ are obtained by summing a feed-forward velocity signal $\mathbf{V}^\star = [U^\star, V^\star, W^\star]$ generated by the autopilot to a correction signal $\mathbf{V}_c = [U_c, V_c, W_c]$ generated by the outer-loop position controller such that:

$$\mathbf{V}_r = \mathbf{V}^\star + \mathbf{V}_c \tag{8.9}$$

this configuration allows for an online correction of the velocity reference values to compensate for real-time errors in the helicopter's position, as the correction signals generated by the outer position loop are included in the generation of the velocity reference.

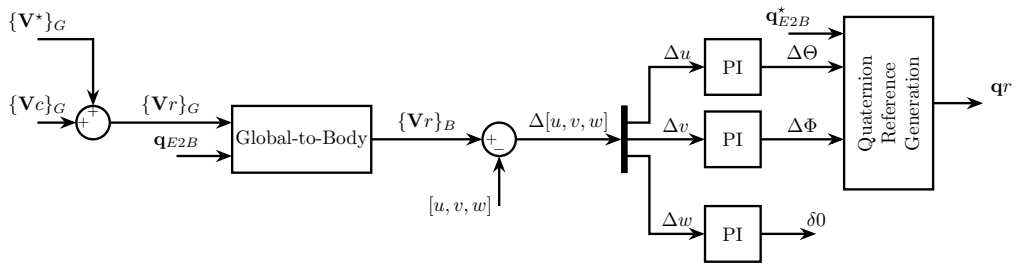


Figure 8.4: Block diagram of the velocity controller

Another point of note is the reference frame in which the various signals are generated: while the reference velocity signals generated with this approach are expressed in the global reference frame $\mathcal{G}$, before these are sent to the rest of the system they are converted to the current helicopter's body frame $\mathcal{B}$ by means of the "Global-to-Body" coordinate rotation block, which rotates the coordinates to the body frame by means of the quaternion passive rotation implemented in Equation 8.10.

$$\begin{bmatrix} 0 \\ \{\mathbf{V}\}_B \end{bmatrix} = \mathbf{q}_{G2B} \begin{bmatrix} 0 \\ \{\mathbf{V}\}_G \end{bmatrix} \mathbf{q}_{G2B}^{-1} \tag{8.10}$$

where $\mathbf{q}_{G2B}$ is the quaternion representing the relative attitude of the body reference frame $\mathcal{B}$ with respect to the global reference frame $\mathcal{G}$. Knowing the relative attitude of the NED reference frame $\mathcal{E}$ with respect to the global reference frame $\mathcal{G}$, the quaternion $\mathbf{q}_{G2B}$ is obtained by means of the concatenation quaternion rotations:

$$\mathbf{q}_{G2B} = \mathbf{q}_{G2E}\mathbf{q}_{E2B} \tag{8.11}$$

From this, the reference signal is split between the horizontal velocity controller and the heave controller. The reference signal W_r - now expressed in the body frame - is compared to the helicopter's heave velocity w and the error is elaborated by the heave PI controller, which generates a correction signal that directly commands the helicopter's collective δ_0.

The errors on the reference signals $[U_r, V_r]$ are used to compute the angle variations $\Delta\Theta$ and $\Delta\Phi$, which are in turn used to compute the quaternion attitude reference $\mathbf{q}_r$ via the "Quaternion Reference Generation" block. These two correction signals ($\Delta\Theta$ and $\Delta\Phi$) should be interpreted as commanded variations of helicopter attitude around its current body Y and X axes respectively, as this mimics the approach used by human pilots during flight: errors in the surge velocity u are usually corrected by tilting the helicopter's nose forward or backward, while errors in the sway velocity v are handled by commanding a rolling motion on the helicopter.

The "Quaternion Reference Generation" block is the last component of the velocity controller and has the purpose of generating an appropriate attitude reference for the inner-loop attitude controller by merging the corrective signals coming from the horizontal velocity controller $[\Delta\Theta, \Delta\Phi]$ with the attitude feed-forward signal $\mathbf{q}^{\star}_{E2B}$ produced by the autopilot module. The generation of the attitude reference follows the approach proposed by Gerig [9] and revolves around the rotation concatenation property of quaternions to correct the feed-forward attitude signal $\mathbf{q}^{\star}_{E2B}$ to account for errors in the horizontal velocity signals. This attitude correction is obtained by producing a correction quaternion $\mathbf{q}_{\Delta}$, which unifies the corrective signals $\Delta\Theta$ and $\Delta\Phi$ produced by the horizontal velocity controller in the following structure:

$$\mathbf{q}_{\Delta} = \begin{bmatrix} \cos(\zeta/2) \\ c_1 \sin(\zeta/2) \\ c_2 \sin(\zeta/2) \\ 0 \end{bmatrix} \tag{8.12}$$

where the parameters ζ, c_1 and c_2 are calculated according to the equations:

$$\zeta = \sqrt{\Delta\Theta^2 + \Delta\Phi^2} \tag{8.13}$$

$$c_1 = \begin{cases} 0 & \text{if} \quad \zeta = 0 \\ \frac{\Delta\Phi}{\zeta} & \text{else} \end{cases} \tag{8.14}$$

$$c_2 = \begin{cases} 0 & \text{if} \quad \zeta = 0 \\ \frac{\Delta\Theta}{\zeta} & \text{else} \end{cases} \tag{8.15}$$

The generation of the reference quaternion leverages the rotation concatenation property, as shown in Equation 8.16: in this manner the pitch, roll, and yaw angles desired values contained by the desired

attitude quaternion are seamlessly modified to also consider the current errors in the velocity components of the control system, allowing for fluid tracking of the planned helicopter motion in both the velocity and the attitude.

$$\mathbf{q}_r = \mathbf{q}^{\star}_{E2B}\mathbf{q}_{\Delta} \tag{8.16}$$

With this, the velocity controller structure has been fully motivated and described, justifying the choice of the PI controller framework and the distinction between horizontal velocity control and vertical velocity control to mimic the behaviour of a human pilot. Further, the interaction with the inner-loop attitude controller was explained by incorporating the desired attitude signal produced by the autopilot and a corrective attitude signal produced by the PI controllers in the horizontal velocity.

8.4. Position controller architecture

With the attitude and the velocity controllers motivated and explained, the final element in the design of the helicopter's flight control system is the position controller. Being the outermost loop of the control system, the position controller bridges the gap between high-level commands generated by the autopilot and the inner-loop controllers by converting a desired position into a velocity reference.

In this section, the position controller will be presented and discussed to provide an overview of its inner structure. To start, in Section 8.4.1 the choice of a proportional (P) structure for the position loop is motivated by explaining the underlying rationale and highlighting the key design considerations. Afterward, Section 8.4.2 provides a detailed description of how the controller functions, with emphasis on the roles of various signals and their contribution to generating the velocity reference.

8.4.1. Motivation of the P architecture

Given the internal functioning of the helicopter's inner loops, the outer position loop was developed to make use of the simple proportional (P) framework. Within this section, the choice of the P controller architecture is motivated by highlighting the physical relation between the position and the velocity and the avoidance of needless integration of references.

To start, the fundamental reason behind the choice of the simple P framework for the position controller is the fact that - as the velocity is the derivative of the position - a command in the velocity channels will generally be proportional to a delta in the position channels for the helicopter. As the purpose of the position controller is to make use of an error in the helicopter's position to generate a reference in the velocity, a proportional mapping was deemed sufficient and adequate for the outer-loop controller.

Another reason for the choice of a proportional architecture is the avoidance of redundant signal integration. The inner velocity loop already incorporates a PI controller that addresses steady-state errors and allows for accurate reference tracking [92]; adding an additional integrator in the outer loop could lead to excessive phase lag and potential issues such as integrator windup, especially under persistent disturbances, while only providing limited improvements in the tracking of position commands.

Lastly, the choice of the proportional design is also motivated by its inherent simplicity. By developing the outer loop using only a P controller reduces the number of tuning parameters, thereby streamlining the design process. This simplicity is particularly valuable in aerospace applications where robustness and ease of implementation are critical.

For the reasons proposed above, the utilization of a proportional controller for the outer position loop is motivated as it allows to leverage the natural relation between position and velocity while maintaining a simpler controller structure that emphasises ease of tuning and quickness in the response. With the choice of controller architecture justified, in Section 8.4.2 an in-depth discussion of the outer loop implementation is provided.

8.4.2. Description of the controller implementation

Concluding the discussion of the control system design presented in Section 8.1, here a detailed description of the controller implementation is provided.

The position controller block - of which a block diagram representation is portrayed in Figure 8.5- is fundamentally tasked with generating an appropriate velocity correction signal $\mathbf{V}_c$ that will be fed to the velocity controller to ensure proper tracking of the desired position command. As explained in Section 8.4.1, the velocity correction signal is generated by means of a simple proportional controller according to the following control law:

$$\mathbf{V}_c = K_P(\mathbf{X}^\star - \mathbf{X}) \tag{8.17}$$

where $\mathbf{X}^\star$ and $\mathbf{X}$ are - respectively - the desired helicopter position (produced by the autopilot) and the actual helicopter position and K_P represents the introduction of the proportional controller.

As the helicopter's current position is not a directly available signal, it is calculated in the "Position Calculation" block, as shown in Figure 8.1. This block obtains the helicopter's complete position by first applying a passive rotation to the body velocity signals $[u, v, w]$ (here not defined as variations but as complete values) to obtain the respective velocity components in the global reference frame $\mathcal{G}$ and then integrating them over time.

As both the helicopter's current position and the desired position produced by the autopilot are expressed in the global reference frame, the resulting velocity corrective signal $\mathbf{V}_c$ will also be expressed in the $\mathcal{G}$ reference frame. As the following loop is the velocity loop, which functions in the body reference frame, after the position controller an appropriate passive rotation is to be applied to convert the velocity reference signal from the $\mathcal{G}$ reference frame to the $\mathcal{B}$ reference frame.

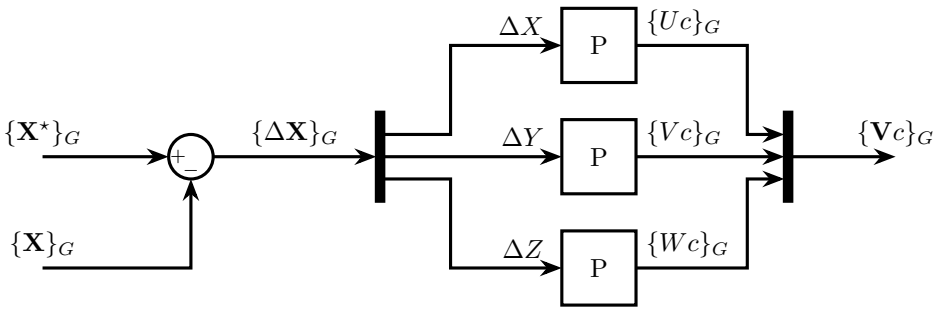


Figure 8.5: Block diagram of the position controller

8.5. Concluding remarks

Within this chapter, the overall control system architecture was presented and explained, highlighting the structure of each controller loop and motivating the controller development approach used.

To start, in Section 8.1 the complete control system architecture was presented, showing the three-loop structure of the controller composed of an inner-loop attitude controller, a middle-loop velocity controller, and an outer-loop position controller, each integrating corrective signals generated by the external loops and feed-forward signals produced by the autopilot, which will be discussed in Chapter 11.

With a high-level understanding of the controller provided, the rest of the chapter focused on the construction of the individual controller blocks. First in Section 8.2 the attitude controller was analysed, motivating the choice of the use of the LQI control algorithm and clearly detailing its implementation.

Afterwards, in Section 8.3 the second control loop was analysed. This control loops was composed of two parallel controllers: a horizontal velocity controller, tasked with elaborating the errors in the body horizontal velocity signals to produce an attitude correction signal; and a vertical velocity controller tasked with directly commanding the helicopter's collective command. Moreover, the use of a PI architecture for this controller was motivated, as it is advantageous for its simplicity, flexibility and ability to allow effective tracking while compensating for constant disturbances.

Lastly, the position controller was discussed in Section 8.4, first motivating the use of only proportional controllers and then discussing the generation of the velocity reference signal, and then providing an explanation of the controller integration with the rest of the system.

With the structure presented the system is capable of handling tracking tasks by leveraging the integral action of the attitude and velocity loops, while having a flexible and applicable design that leverages on its simplicity and flexibility to ensure the system is controlled and capable of appropriately tracking desired reference signals.

9

Tuning of the controller

With the controller structure defined in Chapter 8 and the functioning of each of the control loops presented, this chapter will discuss the tuning process followed to identify the appropriate gains and tuning parameters for the system.

In an effort to reduce the need for hand-tuning and to define a solid and robust methodology that could be flexible and applied to multiple study-cases, the tuning of the controller was performed using an optimization-based approach. Of the various existing tuning algorithms and procedure used in industry and in research, Particle Swarm Optimization (PSO) is widely praised for its effectiveness, having successfully been applied a number of times to the tuning of control systems for plants, vehicles and aircraft alike [93]: because of these reasons and the supporting evidence available, the PSO tuning algorithm was chosen for the purpose of this thesis.

To illustrate the tuning process and appropriately showcase the results obtained, this chapter will be structured as follows. To start, Section 9.1 will provide an overview of the PSO algorithm, introducing the terminology used and providing a systematic description of the algorithm itself. Following this, Section 9.2 will discuss the utilization of the previously described PSO algorithm to perform the tuning for the LQI attitude control system and showcasing the results of the tuning process. Afterward, in Section 9.3 the tuning of the velocity controller is performed, discussing how the PSO algorithm was applied to the tuning problem and discussing the results of the tuning. To conclude, Section 9.4 will discuss the tuning of the position controller and the application of the PSO tuning algorithm to the study case described.

9.1. Implementation of the PSO algorithm

As a fundamental element in the tuning of the Particle Swarm Algorithm, here an overview of the tuning algorithm is provided, discussing its fundamental intuition and implementation.

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique that draws inspiration from the collective behavior observed in flocks of birds or schools of fish [93]. The fundamental intuition behind PSO is that simple agents, or particles, can collaborate through information sharing to explore a complex search space effectively and converge toward optimal solutions. This concept is based on the observation that social behavior in nature often leads to emergent intelligence [94], where the group as a whole can solve problems more efficiently than any individual acting alone. In the context of control system tuning, PSO provides a framework for adjusting controller parameters by simulating a distributed search process where each particle represents a potential set of parameters, and the collective dynamics guide the search toward improved system performance.

At the core of the algorithm is the idea of a swarm, which is composed of N particles that traverse the multidimensional search space. Each particle i is characterized by a position $\mathbf{p_i}$ and a velocity $\mathbf{v_i}$ (with $i = 1 \dots N$), where the position corresponds to a candidate solution for the control system's tuning

problem, and the velocity determines the direction and rate at which the particle moves through the space. A user-defined fitness function $\mathbf{J} = f(\mathbf{p})$ is used to evaluate the quality of each candidate solution: this fitness function quantifies the performance of the control system given a specific set of parameters, thereby providing a measure to compare different solutions. By means of the cost function, the algorithm also keeps track of the particle position $\mathbf{g}^\star$ associated to the most desirable cost $\mathbf{J}^\star$ and of the position $\mathbf{p}_i^\star$ that for each particle lead to its personal best result $\mathbf{J}_i^\star$.

Initially particles are generated with a uniform random distribution in the search space, afterwards, the algorithm proceeds in an iterative manner. At the commencement of each iteration, the fitness of every particle is evaluated at their current position: Each particle then compares its current fitness $\mathbf{J}_i = f(\mathbf{p}_i)$ with the historical global best fitness $\mathbf{J}^\star$, updating its personal best $\mathbf{p}_i^\star$ if an improvement is observed.

Finally, at each iteration an update to the current particle's candidate solution is performed by updating its position. Consider for example the particle i at iteration t: the current solution associated with this particle will be identified by its position, which is identified by the notation $\mathbf{p}_i(t)$. The update of particle i's position is performed by calculating a velocity auxiliary variable $\mathbf{v}_i(t+1)$, which is then used to produce the candidate particle's position at the next iteration $t+1$ according to Equation 9.1

$$\mathbf{p}_i(t+1) = \mathbf{p}_i(t) + \mathbf{v}_i(t+1) \tag{9.1}$$

The velocity variable is calculated at each iteration of the algorithm and it is defined by an equation that encapsulates three fundamental influences: inertia, cognitive drive, and social influence. The inertia term enables the particle to maintain its current momentum, thus preserving the direction of movement. The cognitive term is a reflection of the particle's own experience, encouraging a return to positions that previously yielded high fitness values. The social term, on the other hand, represents the influence of the best-performing particle in the swarm, guiding the particle toward areas of the search space that have been successful. Mathematically, the velocity update is represented as:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1(\mathbf{p}_i^\star - \mathbf{p}_i(t)) + c_2\mathbf{r}_2(\mathbf{g}^\star - \mathbf{p}_i(t)) \tag{9.2}$$

where:

- $\mathbf{v}_i(t)$ is the velocity of the particle i at the previous iteration of the loop (initiated at 0 for the first iteration of the algorithm).
- w is the inertia weight, a scalar term which determines how much of the previous particle velocity is transferred to the current velocity.
- c_1 and c_2 are the cognitive acceleration coefficient and the social acceleration coefficient, which respectively determine how much of the velocity component is going to drive the particle towards its current best $\mathbf{p}_i^\star$ or the identified global best $\mathbf{g}^\star$ respectively.
- $\mathbf{r}_1$ and $\mathbf{r}_2$ are uniformly-distributed random values in the range $[0, 1]$ which aim at including a stochastic element to the algorithm, which in turn allows for a better exploration of the solution space.

A systematic representation of the PSO is given in the pseudocode provided in algorithm 1.

PSO offers several advantages that contribute to its widespread use in the field of control system tuning. Its implementation is relatively straightforward, and it does not require gradient information, which is particularly beneficial when dealing with non-differentiable or noisy fitness landscapes. The ability of PSO to balance exploration and exploitation through the coordinated adjustment of inertia, cognitive, and social influences allows it to effectively navigate complex, multidimensional search spaces. Moreover, the PSO algorithm is highly capable of handling complex optimization cost functions and is well suited for multi-modal optimization, allowing for the simultaneous optimization of different objectives.

Despite these strengths, PSO is not without limitations. One of the principal challenges is the potential for premature convergence, wherein the particles may become trapped in a local optimum rather than exploring the broader search space; this risk is further amplified in problems characterized by highly rugged fitness landscapes with many local optima. Additionally, the performance of PSO is highly sensitive to the choice of hyperparameters such as the inertia weight and acceleration coefficients: improper tuning of these parameters can lead to suboptimal convergence behavior. Note however that all these

Algorithm 1 PSO pseudocode implementation

Init: $\mathbf{J} = f(\mathbf{p})$
Hyperparameters: $N, MaxIter, w, c_1, c_2$
Result: $\mathbf{g}^\star$

for $i = 1$ **to** N **do**
  Initialize particle i *with random position* $\mathbf{p}_i$ *and velocity* $\mathbf{v}_i = 0$
  Evaluate fitness $f(\mathbf{p}_i)$
  Set personal best $\mathbf{p}_i^\star \leftarrow \mathbf{p}_i$
end

Set $\mathbf{g}^\star \leftarrow \arg\min_i f(\mathbf{p}_i^\star)$
for $t = 1$ **to** $MaxIter$ **do**
  for *each particle* i *in the swarm* **do**
    Update velocity:
    $\mathbf{v}_i \leftarrow w \cdot \mathbf{v}_i + c_1 \cdot \mathbf{r}_1 \cdot (\mathbf{p}_i^\star - \mathbf{p}_i) + c_2 \cdot \mathbf{r}_2 \cdot (\mathbf{g}^\star - \mathbf{p}_i)$
    Update position:
    $\mathbf{p}_i \leftarrow \mathbf{p}_i + \mathbf{v}_i$
    Evaluate fitness $f(\mathbf{p}_i)$

    if $f(\mathbf{p}_i)$ *is better than* $f(\mathbf{p}_i)$ **then**
      Set $\mathbf{p}_i^\star \leftarrow \mathbf{p}_i$
    end

    if $f(\mathbf{p}_i^\star)$ *is better than* $f(\mathbf{g}^\star)$ **then**
      Set $\mathbf{g}^\star \leftarrow \mathbf{p}_i^\star$
    end
  end
end
Return $\mathbf{g}^\star$

disadvantages are generally shared by most other optimization algorithms.

Overall, the PSO algorithm is an effective and intuitive approach to optimizing complex optimization problems. While sharing many of the limitations of other metaheuristic optimization algorithms, PSO has a number of significant advantages that make it a suitable and desireable choice for the purpose of control system tuning. To start, its collaborative search mechanism makes it particularly appealing for applications such as control system tuning, where the objective is to identify optimal parameter settings within a challenging search space. Additionally, its flexible design and lack of reliance on gradient-based methods make this approach highly desirable for non-smooth multi-modal optimizations, adding significant freedom to the cases in which this optimization may be applied.

With the PSO algorithm presented and its use motivated, the application of the PSO algorithm will now be discussed to the specific tuning of the control system presented in Chapter 8, highlighting how the optimization is implemented and what cost function is used for the tuning of each control loop.

9.2. Tuning of the attitude controller

With the PSO algorithm introduced and discussed in Section 9.1, the implementation of the optimization algorithm will be discussed in the context of the tuning of the inner-loop attitude controller. To start, in Section 9.2.1 the implementation of the optimization algorithm to the specific attitude controller will be discussed, showing how the tunable parameters of the LQI controller are encoded in the optimization algorithm. Further, in Section 9.2.2 the cost function used to evaluate the performance of the attitude controller is motivated and discussed. Afterwards, in Section 9.2.3 the results of the tuning process will be shown.

9.2.1. Attitude tuning: PSO implementation

As the inner attitude loop makes use of a LQI control framework (discussed in Section 8.2.2), the PSO algorithm is implemented to compute a controller capable of regulating the system and tracking specific attitude references. To properly compute a controller,t two performance matrices - **Q** and **R** - have to be designed: these act as weighting matrices for the calculation of the LQI cost function discussed in Equation 8.5, with the former determining the quickness of the system's response and the latter influencing the amount of control usage.

To encode the matrices in the system, the dimensions and characteristics of these matrices have to be considered. To start, the **Q** matrix is a $n_x \times n_x$ positive-semidefinite symmetrical matrix, where n_x is the number of states of the augmented system obtained in Equation 8.4 (in this specific case, $n_x = 13$). Conversely, the **R** matrix is a $n_u \times n_u$ symmetric positive definite matrix, where n_u is the number of inputs of the augmented system (for the augmented system described in Equation 8.4, $n_u = 3$). As the performance matrices have the characteristics of being positive semidefinite and definite respectively, a common way of defining them in the context of tuning via optimization is by simply encoding them as positive diagonal matrices [95].

With this consideration, the particles for the PSO algorithm were chosen to be 16×1 vectors, with the first 11 entries of the particle encoding the diagonal elements of the states performance matrix **Q** and the last 3 elements encoding the diagonal elements of the input matrix **R**. To determine the allowable values of the particle elements for the PSO algorithm, Bryson's rule may be used [96], which indicates that the **Q** and **R** matrices may be defined as diagonal matrices, with the diagonal elements of **Q** defined as the inverse of the squared allowed error of the respective state they map, and the diagonal elements of **R** defined as the inverse of the squared maximum acceptable value of the respective input they map. This rule is summed up in Equation 9.3.

$$\begin{aligned} Q_{j,j} &= \frac{1}{[max\ allowed\ error\ on\ x_j]^2} \qquad \forall j = 1, \ldots, 14 \\ R_{k,k} &= \frac{1}{[max\ allowed\ value\ of\ u_k]^2} \qquad \forall k = 1, \ldots, 3 \end{aligned} \tag{9.3}$$

9.2.2. Attitude tuning: definition of the cost function

With the encoding of the **Q** and **R** matrix elements discussed, the cost function used in the tuning of the LQI controller is presented. The fundamental objective of the tuning process is to produce a controller that is capable of allowing for simultaneous decoupling and reference tracking in the three attitude axes $[q_i, q_j, q_k]$, all while avoiding saturation in the available control channels $[\delta_x, \delta_y, \delta_p]$. To evaluate these characteristics, the cost function is determined by simulating the control system and evaluating the closed-loop characteristics of the system's response to three different input sets, each generated to command a rotation around the body's axes.

To produce the simulations, appropriate reference signals must first be generated. As the objective is to produce a controller capable of independently providing rotations around each of the helicopter's body axes, the reference signals are generated by first calculating a final desired attitude using the rotation concatenation property of quaternions and then using SLERP to produce an attitude path leading from the initial trim attitude to the calculated final attitude.

For this purpose, three reference sets were produced: for the first reference set, the final attitude is obtained by commanding a rotation of $25°$ around the current body x axis; for the second reference set, the final attitude is obtained by generating a $45°$ rotation around the body y axis; for the third reference set, the final attitude is obtained by generating a $60°$ rotation around the body z axis. With the starting and final attitudes defined for each iteration, the SLERP interpolation (see Equation 5.25) was used to generate the appropriate intermediate references for the helicopter to follow.

The reference signals resulting from this process are shown in Figure 9.1. In the figure, each column shows the references for each input set l; the rows indicate the attitude channels $[q_i, q_j, q_k]$. From the figure it can be noticed that, because of the SLERP algorithm used to define the rotation, some small cross activation on the off-axes is present: even so, as expected, the largest reference activation is on the on axis elements representing rotations on the $[q_i,\ q_j,\ q_k]$ elements. It is also worth noting that in all cases

the q_0 element in the reference also changes its magnitude: this is to reflect the changes in the imaginary quaternion components and maintain unit module to have an accurate representation of attitude.

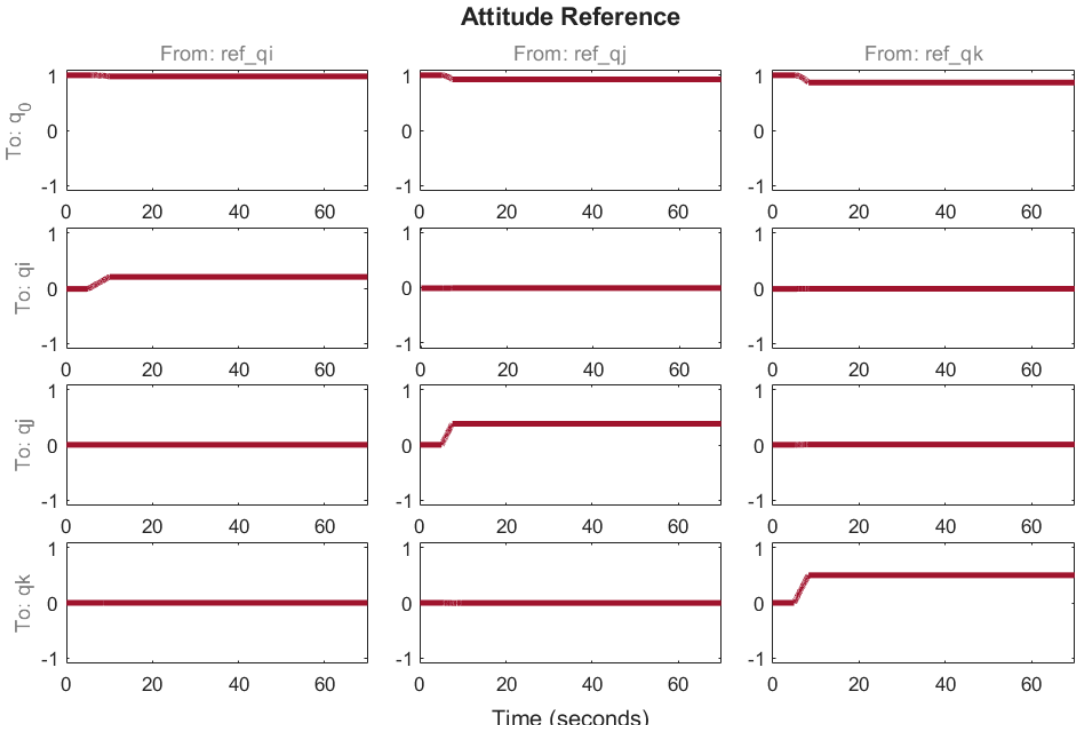


Figure 9.1: Attitude references for the attitude controller tuning

The cost function used for the optimization of the attitude controller's parameters has been developed to contain three separate terms, as shown in Equation 9.4, each scaled with an appropriate weight W.

$$\begin{aligned} z = &\sum_{l=1}^{3} W_R \left(\sum_{t=1}^{T} (\Delta q_0^l(t))^2 + \sum_{t=1}^{T} (\Delta q_i^l(t))^2 + \sum_{t=1}^{T} (\Delta q_j^l(t))^2 + \sum_{t=1}^{T} (\Delta q_k^l(t))^2 \right) + ... \\ &\sum_{l=1}^{3} \left(\sum_{t=1}^{T} \left(W_x \Delta\delta x^l(t)^2 + W_y \Delta\delta y^l(t)^2 + W_p \Delta\delta p^l(t)^2 \right) \right) + ... \\ &W_T \left(Ts(q_i^1(t)) + Ts(q_j^2(t)) + Ts(q_k^3(t)) \right) \end{aligned} \tag{9.4}$$

The first term penalizes the mean squared error (MSE) on the on-axis reference signal tracking error: the use of the MSE as a measure of the signal performance in this case is especially advantageous, as it allows to give a higher penalization on higher deviations from the reference, ensuring the attitude follows the reference more closely. In the notation described above, the attitude error is quantified by the error quaternion $\Delta\mathbf{q}$, which represents the distance between the reference attitude $\mathbf{q}_r$ and the current attitude $\mathbf{q}_{E2B}$. To calculate the error quaternion, first the difference between the reference attitude and the current attitude has to be quantified by means of an intermediate quaternion $\mathbf{q}_\Delta$, obtained via the equation:

$$\mathbf{q}_\Delta = \mathbf{q}_{E2B}^{-1} \mathbf{q}_r \tag{9.5}$$

Note that, when the two attitudes coincide, the intermediate quaternion produced will be the identity quaternion:

$$\mathbf{q}_\Delta = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{9.6}$$

The attitude error quaternion $\Delta\mathbf{q}$ is therefore quantified by the difference between the intermediate quaternion $\mathbf{q}_\Delta$ and the identity quaternion, as shown in Equation 9.7 [64].

$$\Delta\mathbf{q} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \mathbf{q}_\Delta \tag{9.7}$$

The second term is related to the input activation and aims at penalizing too-large input activations and preventing actuator saturations: this is achieved by appropriately selecting the weights W_x, W_y, W_p to have different values depending on the total amplitude of their respective signal, effectively creating desired areas for the input actuation. An example of the input activation areas is shown in Figure 9.2, where a sample input signal is shown (in this case, the longitudinal collective δ_x signal). From the figure, it is clearly noticed that the area is divided in three sections:

- The central white region is positioned around the actuator's trim position and is assigned the smallest weight W, as the actuator activation is not significant.
- The green areas are regions with a higher penalization weight, aimed at discouraging the controller to command inputs that enter these regions for too long.
- The outside red areas have a very high associated weight and are set to start slightly before the actuator upper and lower bounds: the aim of these regions is to avoid saturation during the commanded manoeuvres.

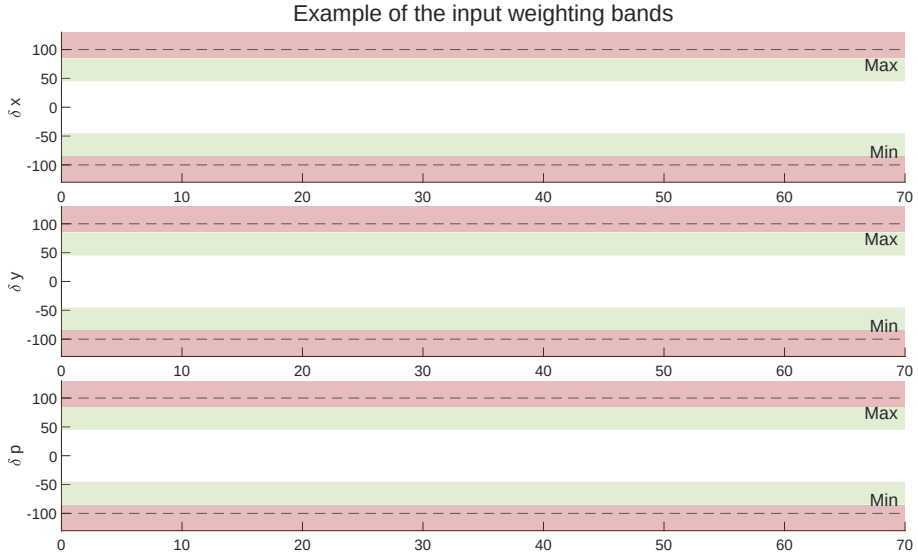


Figure 9.2: Visual representation of the weight selection bands for the input actuation

Lastly, the third term of the cost function is aimed at encouraging the system to achieve a small settling time Ts on the attitude signal that is primarily activated in each of the simulations. In particular, it is noticed from Figure 9.1 that in the first simulation $l = 1$ the attitude signal primarily excited is the q_i signal, the second $l = 2$ simulation will primarily concern the q_j signal, and the third $l = 3$ simulation will be related to the q_k signal.

The weights used for this tuning were selected to more heavily prioritize control accuracy over control quickness, as this was found to lead to overall more robust performances over different tracking tasks. As a hard limitation of the control system was also to avoid control saturation, the weights for the different control regions were selected to apply a noticeably heavier penalization to the cases in which allowed control limits were violated: this essentially applies implements the "big M" method to avoid control saturation in the command channels [97, 98], discouraging the convergence to solutions with large control actuations. With the cost function described above, the PSO optimization algorithm will try to find the diagonal Q and R matrices that are capable of producing a closed-loop system whose response can achieve a good settling time and match a desired reference while ensuring avoiding control saturation and encouraging small control deflections.

9.2.3. Attitude tuning: results of the tuning

With the PSO algorithm implementation discussed and the cost function analysed in Sections 9.2.1 and 9.2.2 respectively, the results of the attitude controller tuning are discussed next. In this case, the tuning of the PSO was performed with a population of 50 particles over 250 iterations, ensuring a good exploration of the solution space with the random initialization and allowing for enough time for the particles to reach optimal solutions. The evolution of the cost over the tuning iterations are shown in Figure 9.3, where the contribution of each of the cost terms of the final identified global best is also shown in the lower section of the figure.

By analysing the evolution of the cost over the iterations it is noticed that the PSO algorithm implemented according to the formulation provided in Section 9.1 is able to converge at an optimal solution over the various iterations of the tuning. By analysing the evolution of the cost over time is also noticed that there are noticeable drops at various intervals of the tuning: this suggests that the cost function has discontinuities which are most likely caused by the changes in cost coefficients for the input activation, making the cost function non-smooth. As the PSO algorithm is not gradient-based, the impact of these discontinuities is limited and can be compensated for by using a large enough amount of particles: with the selected weights, it was deemed appropriate to use 50 to allow for a large exploration of the solution space. By analysing the contribution of each objective to the final cost function it is noticed that the cost objective is the smaller of the three components, suggesting that the solution found does not cause control saturation.

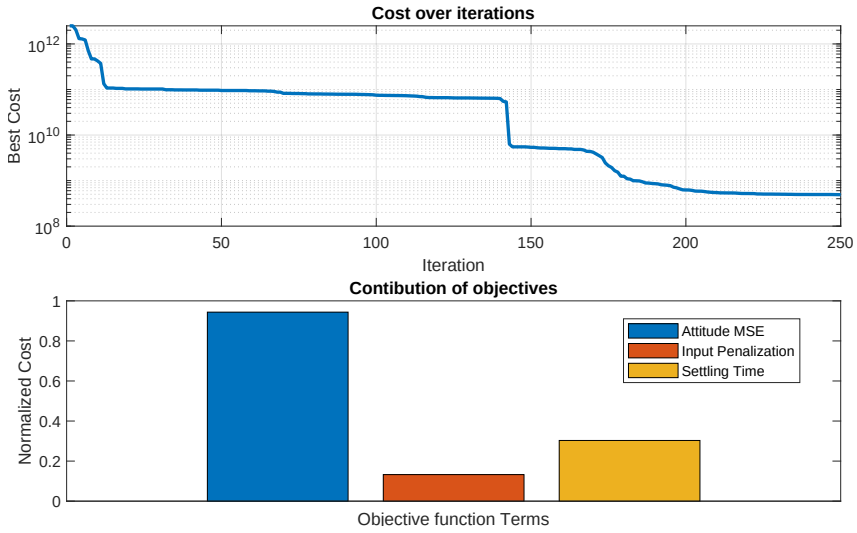
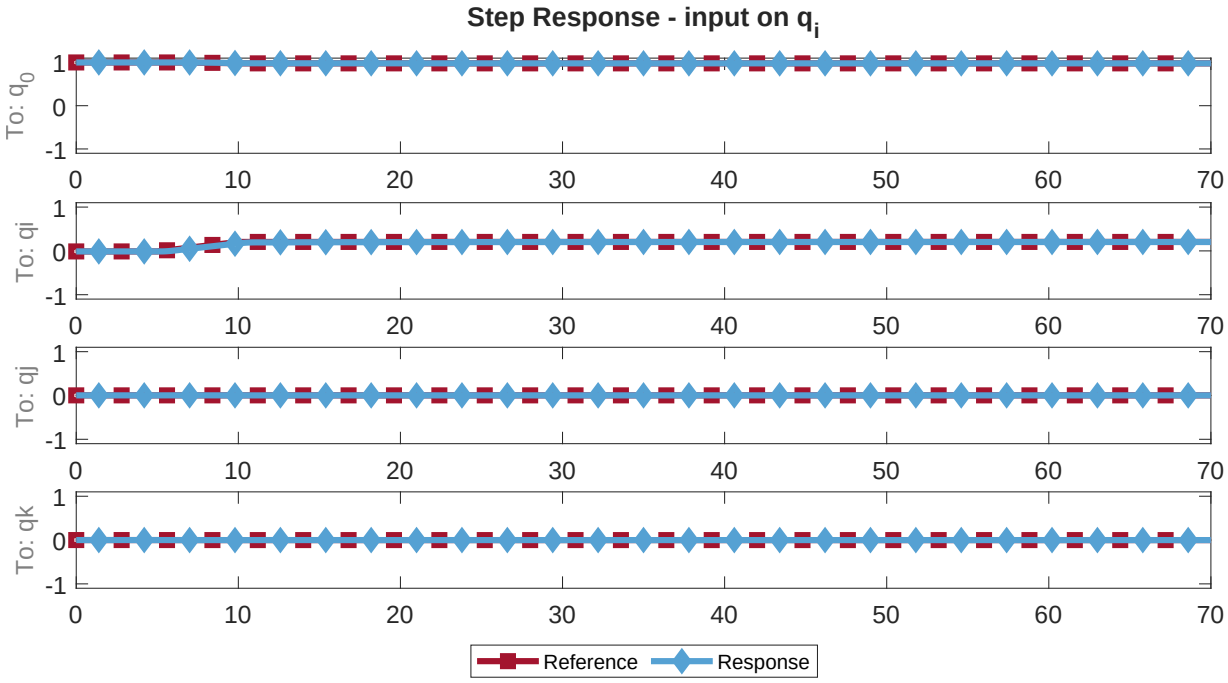


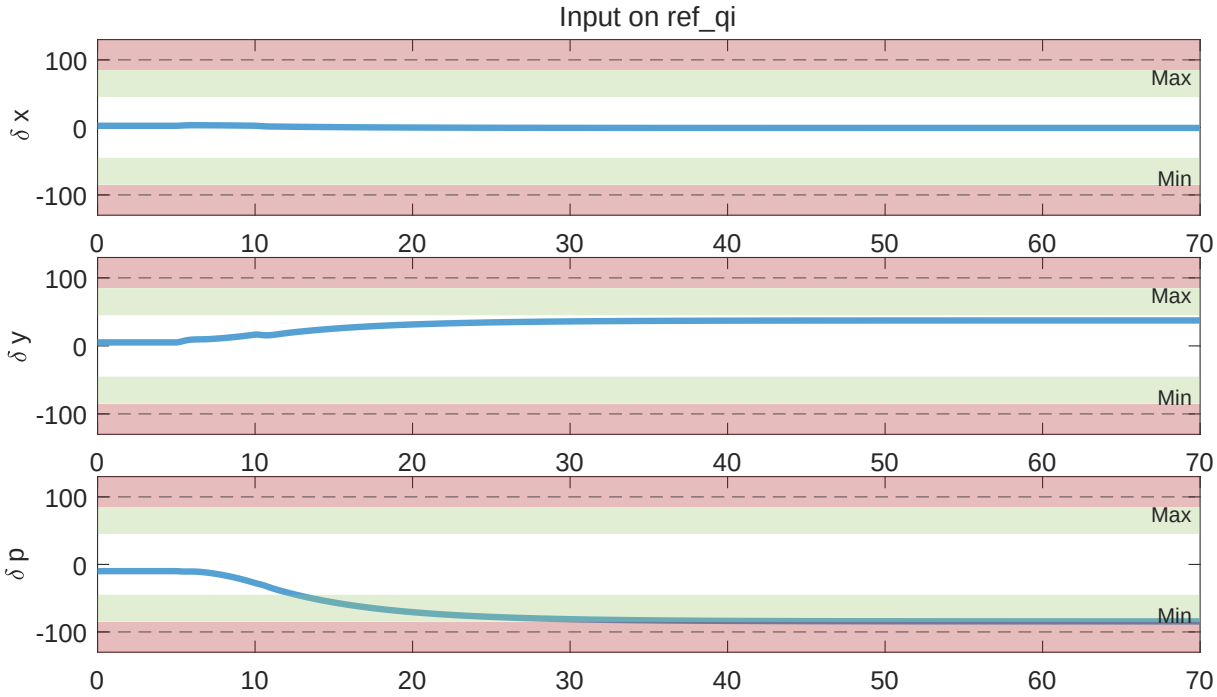
Figure 9.3: Evolution of the cost function of the attitude controller tuning

Overall, the inner-loop tuning algorithm was able to obtain good tracking performance, with a quick response and no command saturation. The figures presented hereafter show the results of the tuning process. Figures 9.4a and 9.4b respectively show the attitude signal response and the control usage for the first simulation set - a commanded $25°$ rotation around the trim body x axis. The results for the second reference set - a $45°$ rotation around the trim body y axis - are shown in Figure 9.5a, while the respective inputs required to obtain such performance are provided in Figure 9.5b. Lastly, the results for the third simulation set are shown in Figures 9.6a and 9.6b, where the attitude signal response and the respective control inputs required to obtain it are shown for a commanded rotation of $60°$ around the trim body z axis. The tracking performance is good on all three inputs, with a prompt response on the attitude to follow the references. The control axis showing the largest activation in all cases is the pedal, which is especially activated to compensate for the required input on q_i (i.e.: the $25°$ rotation along the ϕ Euler angle). This response is expected, as a right turn (i.e.: a positive bank angle) will generally reduce the antitorque action required by the pedal - as seen in Figure 9.4b [99]. Additionally, this behaviour is justified by the fact that, because of the structure of the controller, the reference yaw angle ψ to be held by the helicopter is to be defined as a feedforward attitude command encoded in the quaternion reference: since in these tuning scenarios no feedforward yaw commands are given, the helicopter is using the pedal to naturally counter the change in orientation of the helicopter's nose to maintain its trim yaw even while performing a sidestep

movement.

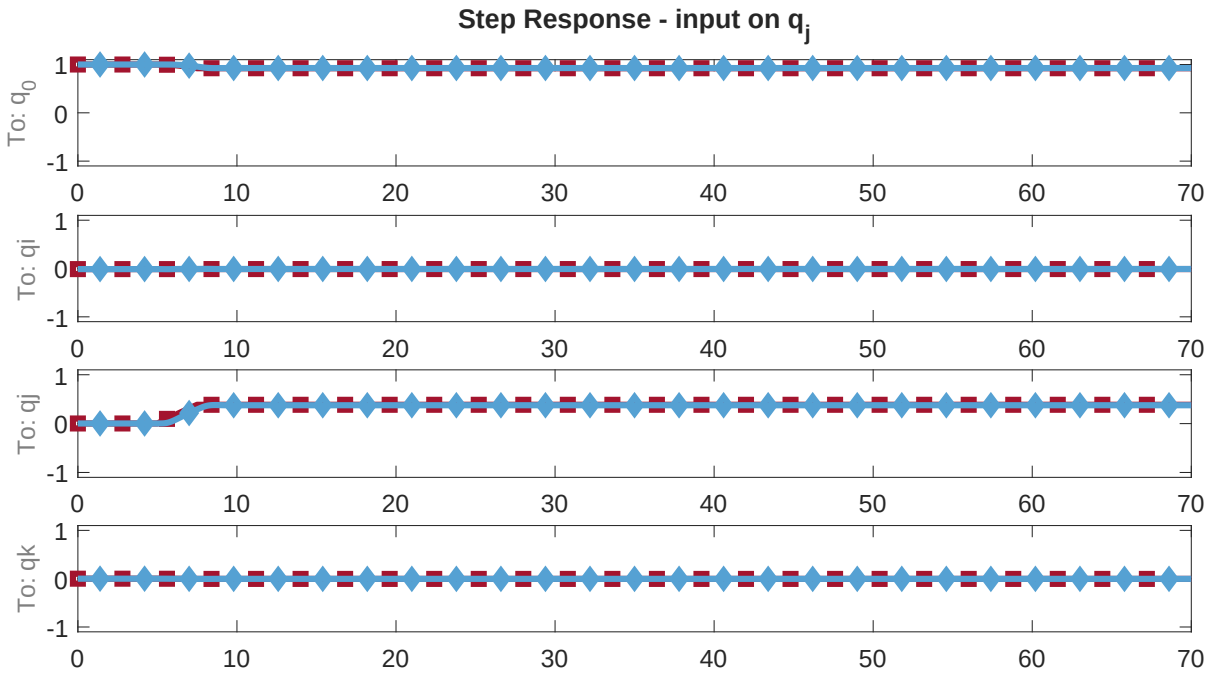


(a) First simulation - attitude signals

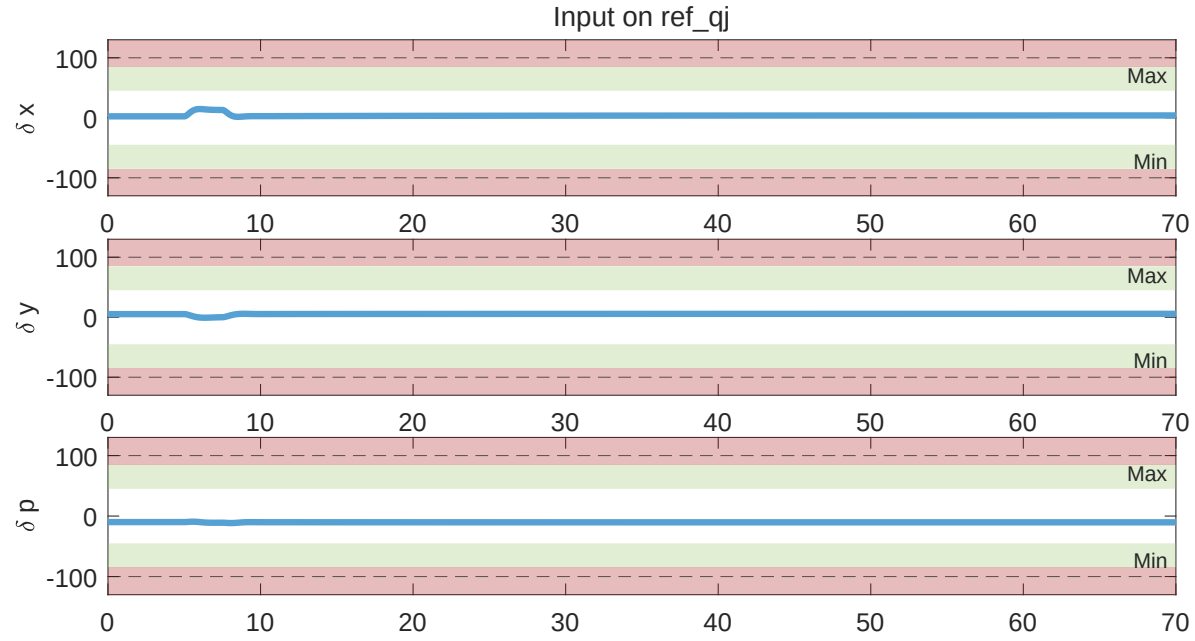


(b) First simulation - control signals

Figure 9.4: First simulation: $25°$ rotation around the body x axis

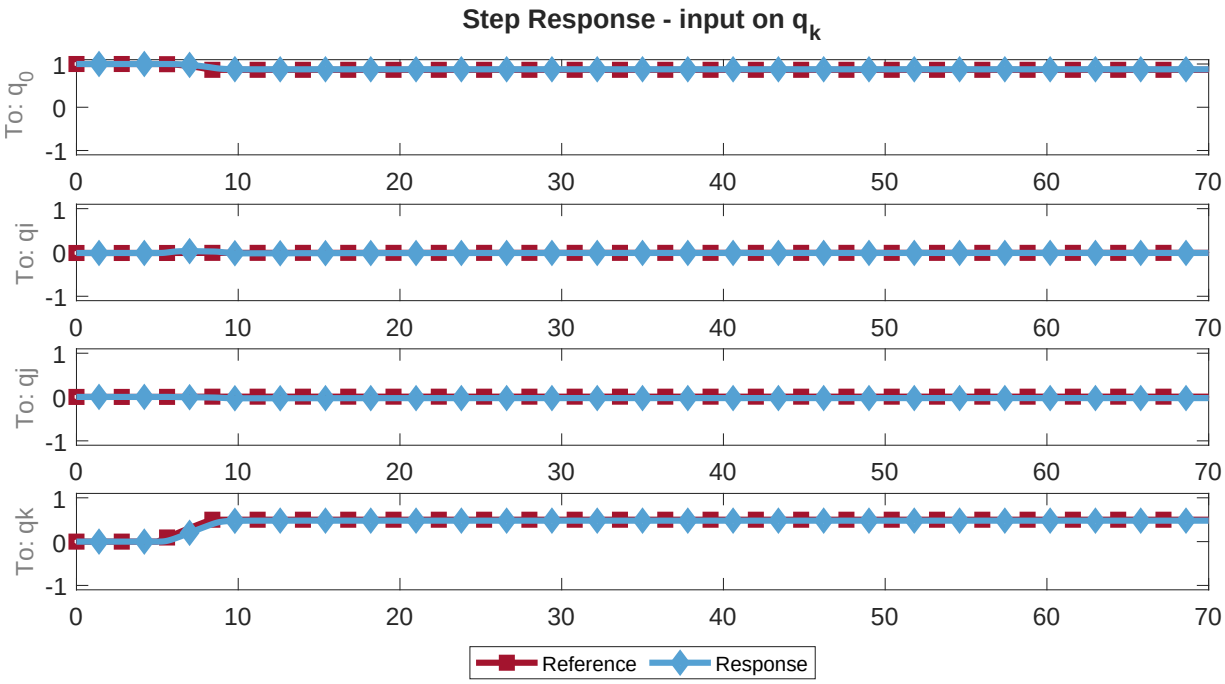


(a) Second simulation - attitude signals

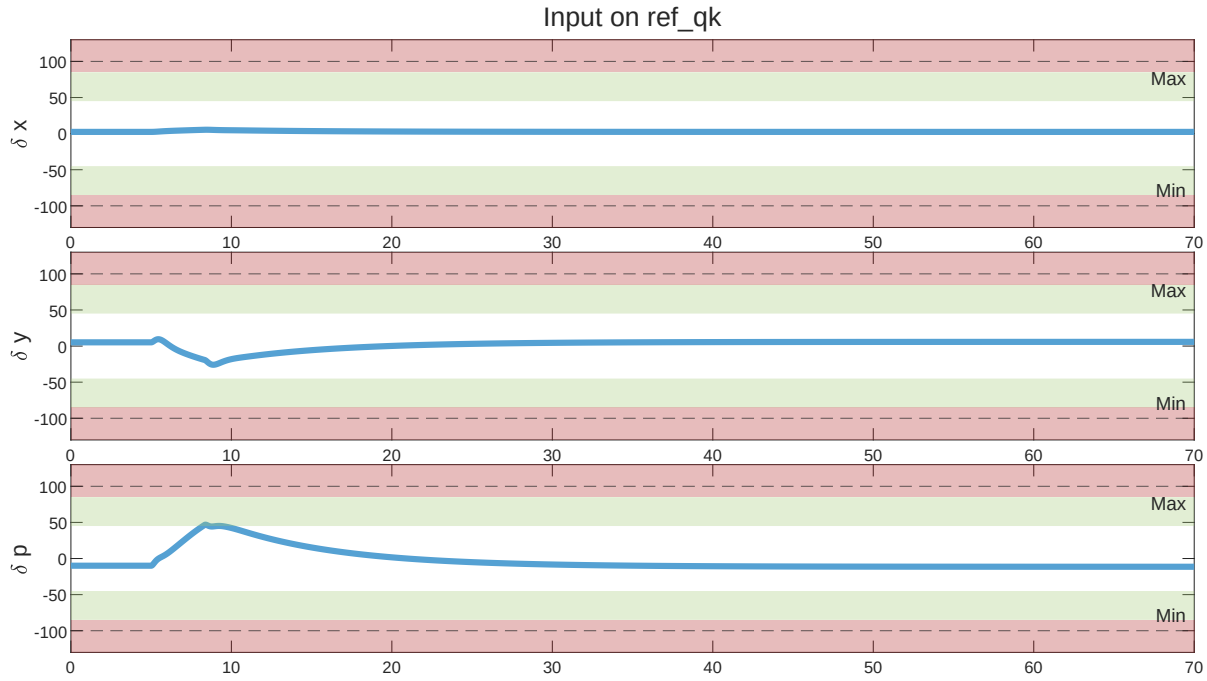


(b) Second simulation - control signals

Figure 9.5: Second simulation: $45°$ rotation around the body y axis



(a) Third simulation - attitude signals



(b) Third simulation - control signals

Figure 9.6: Third simulation: $90°$ rotation around the body z axis

9.3. Tuning of the velocity controller

Building upon the results of the attitude controller tuning discussed in Section 9.2, here the tuning of the velocity controller is discussed. A discussion of the velocity controller architecture is provided in Section 8.3, where the interplay between the various controller blocks was discussed and the PI framework was motivated.

To provide an orderly discussion of the tuning process, this section has been further divided in two parts. To start, in Section 9.3.1 the implementation of the optimization algorithm and the cost function used for the optimization process are discussed, demonstrating how the controller parameters are encoded as PSO agents and providing an intuitive understanding of the motivation behind the construction of the cost function. Afterwards, in Section 9.3.2 the results of the velocity controller tuning are provided and briefly analysed, discussing the characteristics of the controller tuning results.

9.3.1. Velocity tuning: PSO implementation and cost function

As the velocity loop makes use of the widely-adopted PI formulation on three velocity channels, the controller will be in total composed of six tunable parameters. As these gains are directly included in the controller and there is no need of any intermediate auxiliary parameter (as was the case for the LQI attitude controller in Section 9.2.1, which required the definition of the **Q** and **R** matrices), the implementation of the PI gains as agents of the PSO algorithms is straightforward: in particular, each agent was defined as a 6×1 vector where the first three elements encoded the values of the proportional controllers, and the last three elements the values of the integral controllers. With the PSO agents defined, the cost function used in the tuning of the velocity control system is discussed.

To start, the velocity controller is tasked with enabling the system to perform appropriate tracking of velocity reference signals, while limiting control usage to avoid saturation and ensuring that the signals fed to the attitude controller are also being effectively tracked: to evaluate these characteristics, the cost function is evaluated by simulating the performance of the closed-loop system to three distinct reference sets, each activating one of the body velocity channels independently of the others.

For this purpose, $l = 3$ reference sets were produced, each commanding a 10 m/s variation in velocity on of the three body velocity axes - u, v, and w - while leaving the two other velocity signals at 0. To avoid sharp variations in the velocity references and provide some for of dynamics to the reference signal, the reference signal is generated by passing a step velocity command through a first-order filter with a time-constant of $\tau = 0.1$ s. A graph of the reference signals is provided in Figure 9.7. In the figures, the columns represent the reference sets $l = 1 \dots 3$ while the rows represent the respective velocity channels $[u, v, w]$.

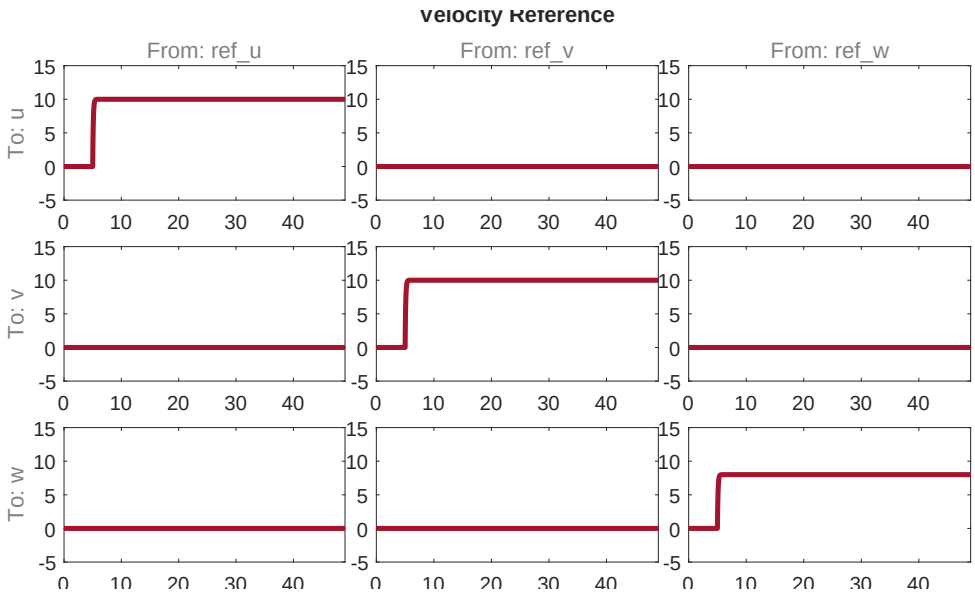


Figure 9.7: Velocity references for the velocity controller tuning

To appropriately ensure the velocity controller is able to obtain its design objective, an appropriate cost function was developed, taking into account the requirement of tracking velocity signals while limiting control actuation and ensuring the correct functioning of the inner attitude loop. The cost function developed is presented in Equation 9.8: as it can be seen, the equation can be divided in four separate terms, which will individually be explained hereafter.

$$
\begin{aligned}
z = & \sum_{l=1}^{3} W_{Rv} \left(\sum_{t=1}^{T} (u_r^l(t) - u^l(t))^2 + \sum_{t=1}^{T} (v_r^l(t) - v^l(t))^2 + \sum_{t=1}^{T} (w_r^l(t) - w^l(t))^2 \right) + ... \\
& \sum_{l=1}^{3} W_{Ra} \left(\sum_{t=1}^{T} (\Delta q_0^l(t))^2 + \sum_{t=1}^{T} (\Delta q_i^l(t)(t))^2 + \sum_{t=1}^{T} (\Delta q_j^l(t))^2 + \sum_{t=1}^{T} (\Delta q_k^l(t))^2 \right) + ... \\
& \sum_{l=1}^{3} \left(\sum_{t=1}^{T} \left(W_x \delta x^l(t)^2 + W_y \delta y^l(t)^2 + W_p \delta p^l(t)^2 + W_0 \delta_0^l(t)^2 \right) \right) + ... \\
& W_T \left(Ts(u^1(t)) + Ts(v^2(t)) + Ts(w^3(t)) \right)
\end{aligned}
\tag{9.8}
$$

The first two terms of the cost function ensure the velocity and attitude signals follow the respective references: this system behaviour is obtained by encouraging the minimization of the respective MSEs on the reference signals. By using the MSE as a measure of performance, larger deviations from the reference values are more heavily penalized, which encourages solutions that lead to a closer following of the references. These terms are calculated once for each simulation run l and appropriately weighted with the scalar weights W_{Rv} for the velocity signals and W_{Ra} for the attitude signals. Note that the attitude tracking error $\Delta\mathbf{q}$ is calculated using the same methodology discussed in Equation 9.7.

The third term of the cost function weighs the input command activation to avoid command saturation and discourage excessive control usage. This is obtained using the same method presented in Section 9.2.2, where each available input band is divided into three regions, each with a different performance weight assigned.

The fourth term, similarly to the cost equation presented in Equation 9.4, is a weighting on the on-axis settling time for the activated velocity signals. This term emphasises response quickness which, when paired with the accuracy penalization provided by the MSE weighting, allows for a more effective matching of the desired reference response.

As the main objective of the controller is to ensure velocity tracking, higher penalizations were applied to the velocity MSE term and the settling time term, ensuring the tuning process prioritized them. The attitude tracking term was associated with a proportionally smaller weight. On the other hand, analogously to what was done for the attitude tuning, the control actuation weights were selected to very heavily penalize solutions that involve actuator saturations, ensuring the control system converged to solutions that limit commanded control deflections to remain within the allowed bounds.

9.3.2. Velocity tuning: results of the tuning

With the implementation of the tunable parameters as position agents in the PSO algorithm discussed and the construction of the cost function presented. For the velocity controller tuning, the simulation was conducted over a population of 30 agents over 50 iterations of the algorithm. Figure 9.8 shows the results of the tuning process, with the evolution of the identified global best $\mathbf{g}^\star$ over the iterations presented at the top of the figure and the individual components of the cost function presented at the bottom of the figure. Figures 9.9 to 9.11 instead show the results of the three simulations used in the tuning process.

From Figure 9.8 it is noticed that the cost follows the typical trend of metaheuristic and learning algorithms, showing a large reduction in the cost value in the first iterations and then gradually settling at an optimal solution. As no noticeable improvements were noticed in the last iterations, the tuning was stopped after 50 iterations. Moreover, by analysing the cost terms shown in the lower half Figure 9.8 it is observed that the primary contribution to the final cost is provided by the tracking error on the velocity: this is due to its relatively higher weight, as this term is of primary importance to the tuning of the velocity controller.

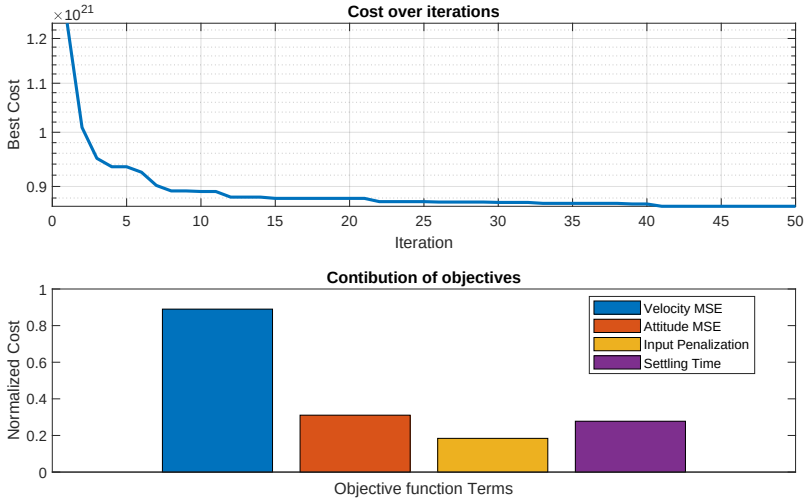


Figure 9.8: Evolution of the cost function of the velocity controller tuning

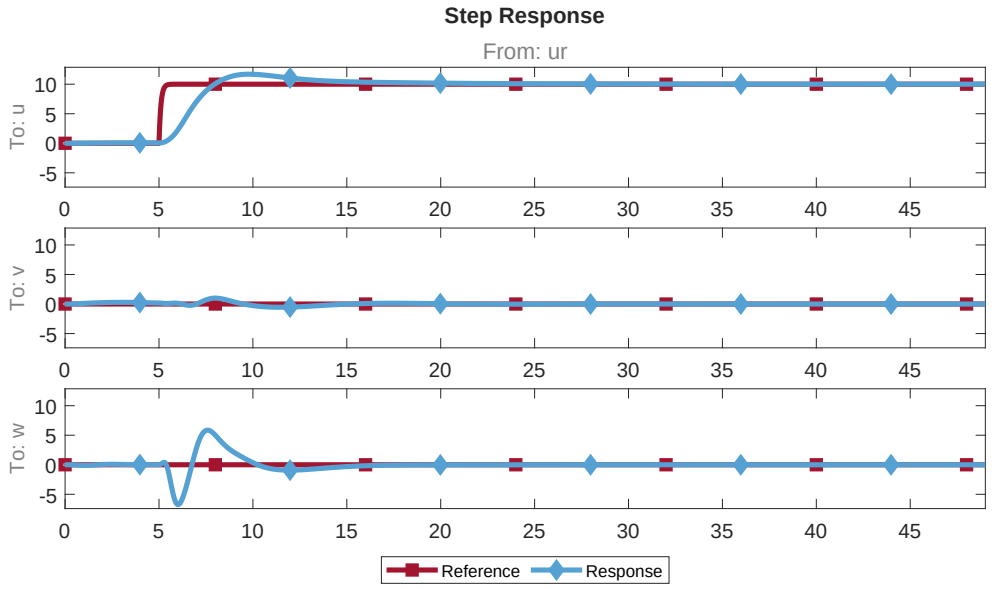
Another important factor observed during both the tuning process and the formulation of the cost function was the critical role played by the attitude tracking penalization term. In its absence, the tuning process generally yielded faster responses in the velocity channels. However, this apparent improvement in performance was achieved predominantly by commanding abrupt and large attitude changes within the innermost control loop. This observation highlights the importance of the attitude tracking penalization term for attitude tracking errors within the cost function; without such a term, the controller tuning process may prioritize rapid transient responses at the expense of the overall tracking accuracy and stability, thereby compromising the system's ability to maintain the desired attitude over time.

As expected, the relatively small contribution of the control usage term, on the other hand, is justified by the control actuation graphs shown in figures 9.9c, 9.10c, and 9.11c: in these graphs the commanded control deflections generally lie within the first penalization region, which has the lowest weighting of the three zones and as such does not produce a large contribution to the cost.

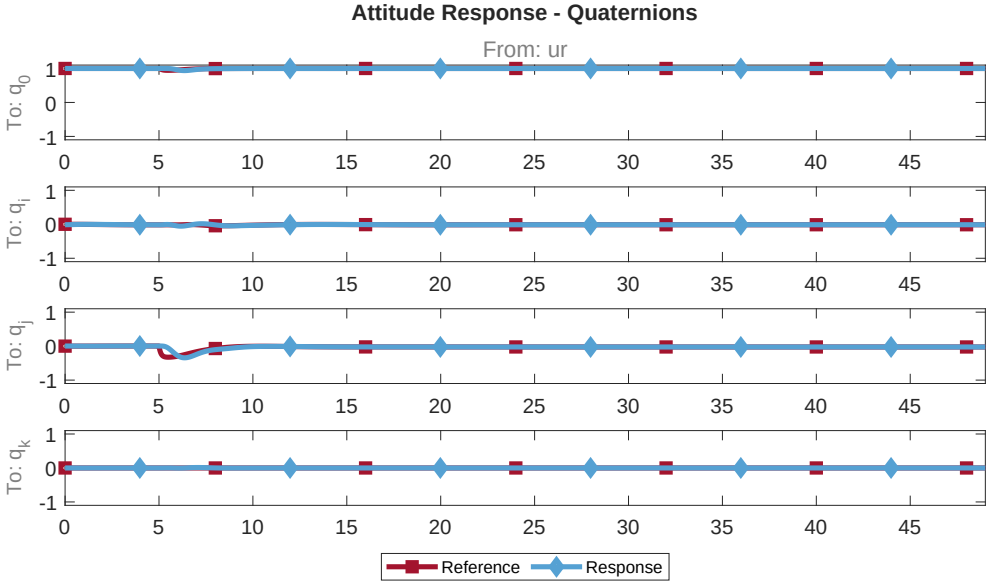
In terms of dynamic performance, the velocity controller is effective in tracking velocity reference signals, even with steep commanded variations such as $10\text{m}/\text{s}$. In particular, the results of the tuning suggest a good decoupling between the u and v axis, which is likely due to the attitude controller (which is directly involved in the correction of horizontal velocity commands, as was explained in Chapter 8). Noticeably worse cross-coupling effects are noticed when analysing the heave w response when inputs are commanded on the surge and sway axis: this result is somewhat expected, as the eigenmode analysis in Section 6.2.1 showed that the velocity channels of the helicopters were highly coupled to one another. By further analysing the response, the system shows overshoots and delays in the velocity signal: this would suggest that an additional derivative action in the control is desired, as it would contribute in significantly speeding up the response. This was however avoided, as in practical applications derivative actions are highly subject to signal noise and may cause de-stabilising actions.

Suggesting the validity of the obtained results, the attitude response matches the expected behaviour: an increase in the surge velocity commands a reduction of the pitch command θ tilting the helicopter forward, seen as a decrease of the q_j quaternion component; an increase in the sway velocity is obtained with an increase of the bank angle ϕ tilting the helicopter to the right, seen as an increase in the q_i quaternion component. Variations in the vertical velocity component have little-to-no effect on the attitude and are obtained only via collective commands δ_0: this is expected, as the vertical velocity control has been completely decoupled from the horizontal velocity control, in a manner analogous to the control actions that a human pilot would show.

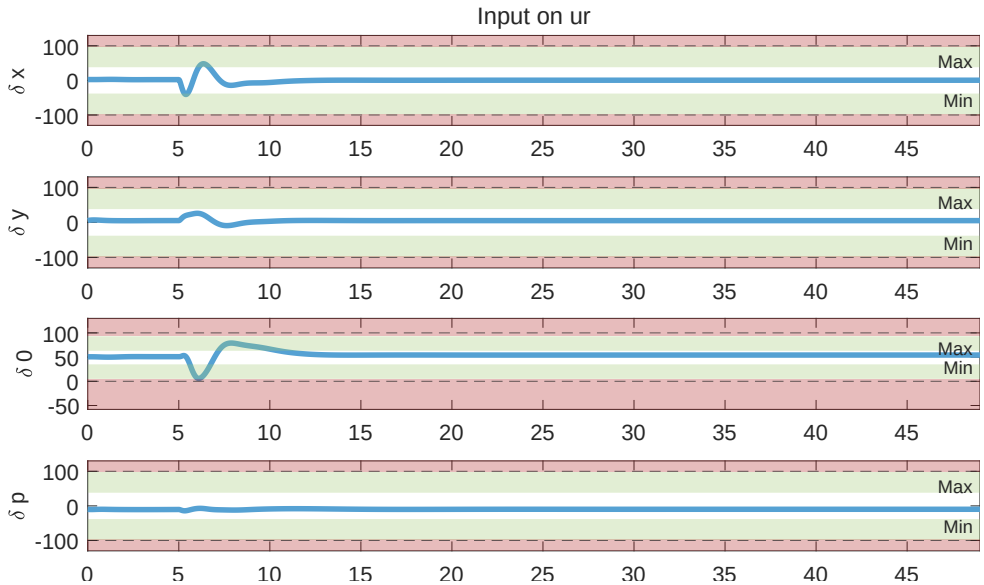
Overall, the velocity loop tuning process was able to produce a responsive control system capable of following velocity commands with accuracy and limited actuator usage. The velocity tracking penalization term and the settling time penalization were instrumental in achieving a good performance in the tracking of velocity commands, while the opposing attitude tracking and input penalization terms ensured the efficacy of the inner loop attitude controller while reducing the control effort required to achieve the commanded velocities.



(a) First simulation - velocity signals

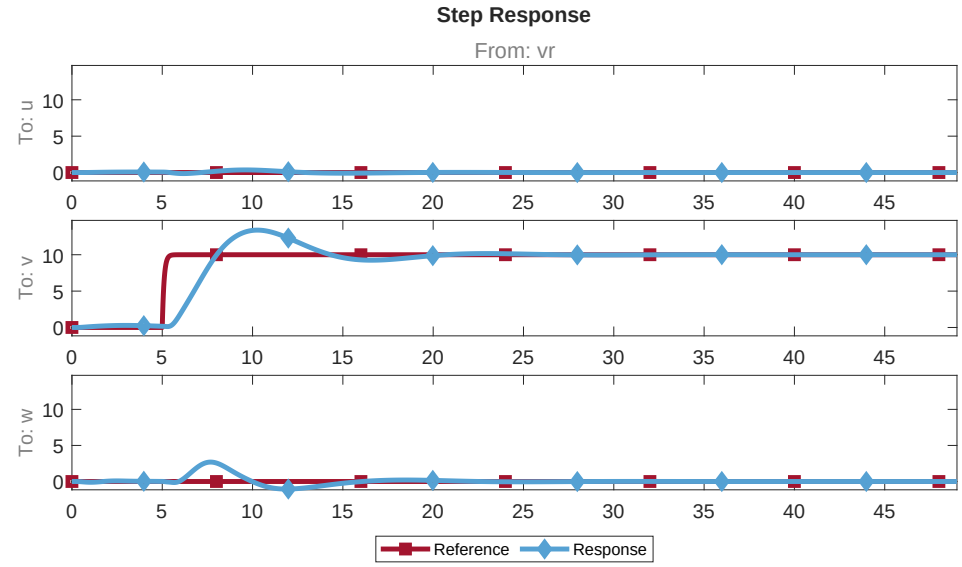


(b) First simulation - attitude signals

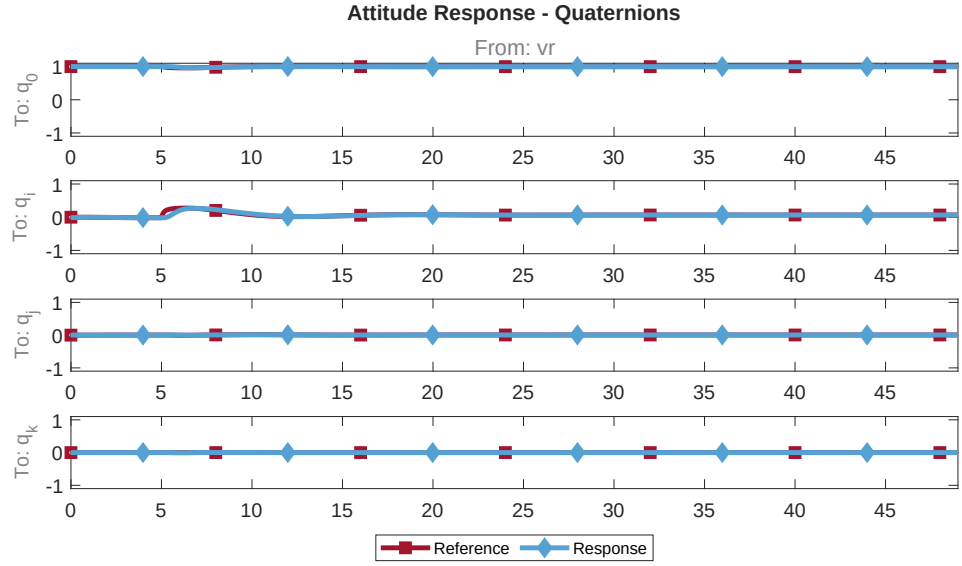


(c) First simulation - control signals

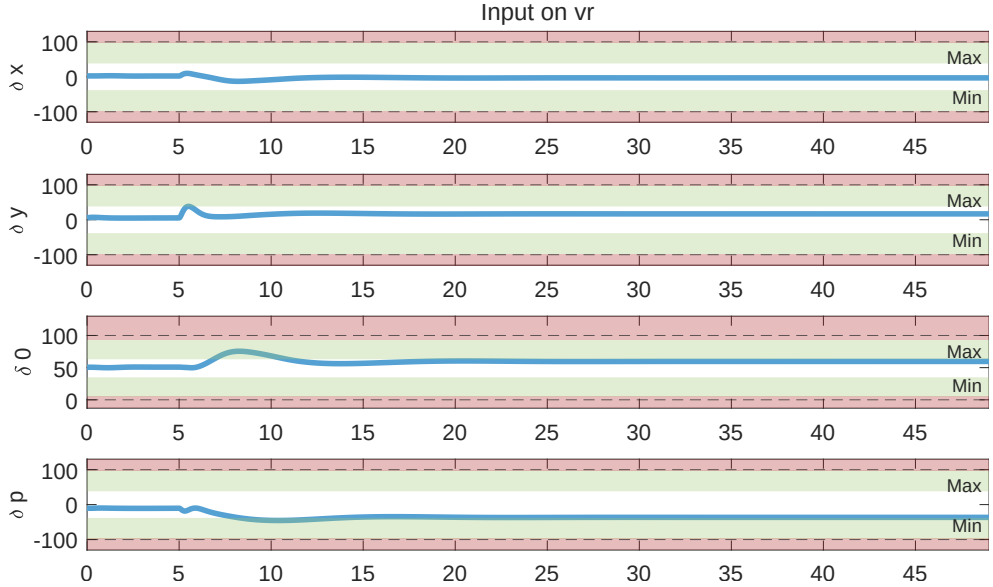
Figure 9.9: First simulation: 10 m/s surge u command



(a) Second simulation - velocity signals

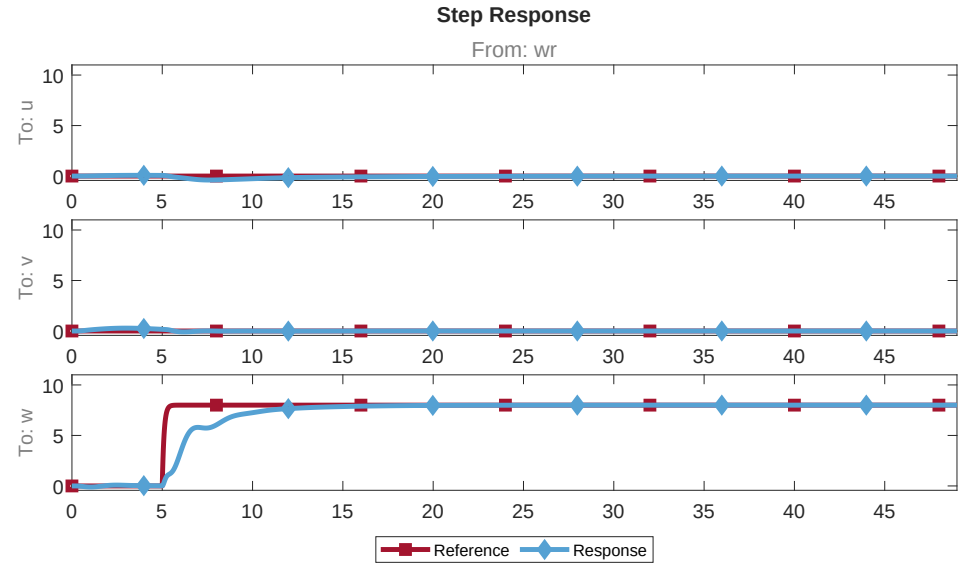


(b) Second simulation - attitude signals

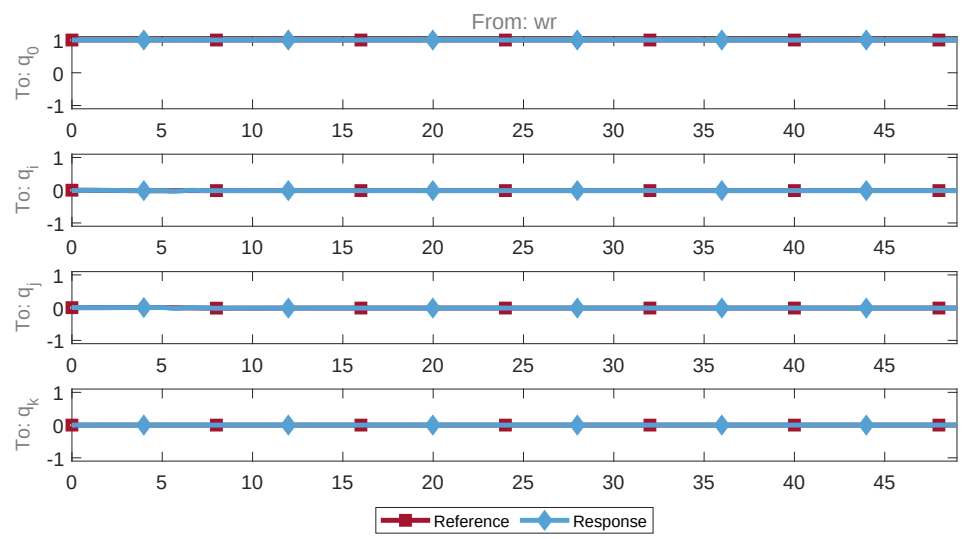


(c) Second simulation - control signals

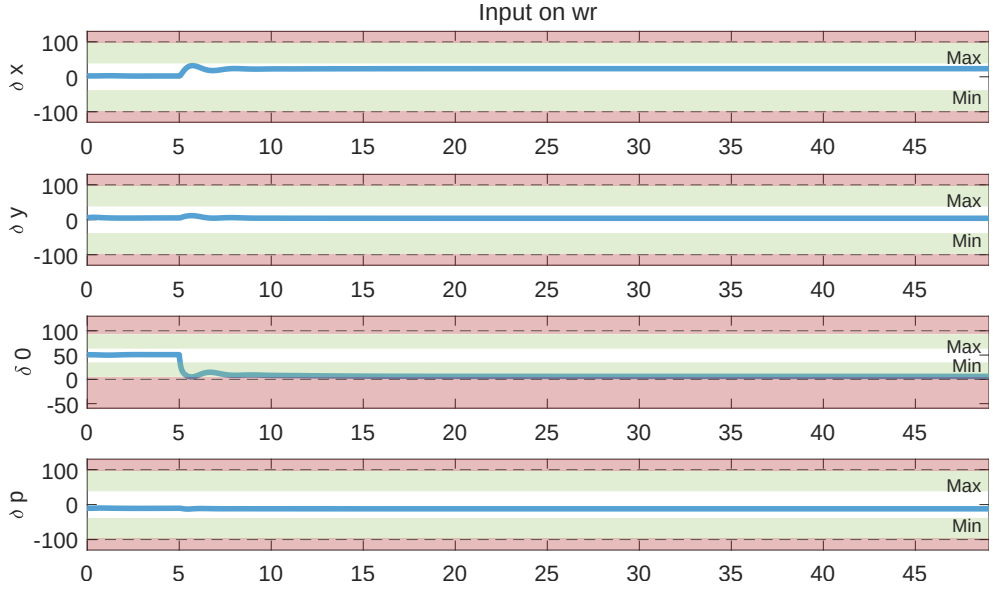
Figure 9.10: Second simulation: 10 m/s sway v command



(a) Third simulation - velocity signals



(b) Third simulation - attitude signals



(c) Third simulation - control signals

Figure 9.11: Third simulation: 10 m/s heave w command

9.4. Tuning of the position controller

Now that the inner attitude and velocity loops have been modelled and defined, the tuning of the outer loop position controller is expanded on in this section. A discussion of the position controller architecture is provided in detail in Section 8.4, where the use of proportional controllers is motivated and a description of the inclusion in the rest of the control system is provided.

Hereafter the controller tuning process is discussed. First, in Section 9.4.1 the implementation of the PSO optimization algorithm is discussed for this specific design case, explaining how the controller tunable parameters were encoded as agents of the optimization algorithm and introducing the cost function used for the tuning process. Afterwards, with the tuning setup defined, the results of the tuning will be presented in Section 9.4.2.

9.4.1. Position tuning: PSO implementation and cost function

Similarly to the velocity controller, the position controller proportional formulation allows for a straightforward implementation of the controller tunable parameters as elements of the PSO agents. As the position controller includes three proportional gains (one per position channel), the PSO agents were defined as 3×1 vectors where each element represents the value of the controller gain. With the PSO agents defined, the tuning process and the cost function is discussed hereafter.

As discussed in Section 8.4.2, the purpose of the position controller is to ensure the helicopter follows a desired trajectory by generating an appropriate velocity correction term $\mathbf{V}_c$ to drive the rest of the system, while ensuring the correct functioning of the inner loop velocity and attitude controllers. To evaluate these characteristics, the tuning process was set up to perform three separate simulations - each with a different reference set - to then evaluate the system's obtained performances by means of an appropriately-defined cost function.

As the model is linearized in forward flight at 33 m/s, to ensure the simulations remained as representative of the model as possible, the reference positions were generated by integrating appropriately-defined velocity signals $\mathbf{V}_r$. The procedure used to generate the velocity reference signals $\mathbf{V}_r$ is fully described by Equation 9.9:

$$\{\mathbf{V_r}^l\}_G = \{\mathbf{V_{trim}}\}_G + \{\Delta\mathbf{V}^l\}_G \tag{9.9}$$

where $\{\mathbf{V_{trim}}\}_G$ indicates the helicopter's trim velocity expressed in the global reference frame $\mathcal{G}$ and $\{\Delta\mathbf{V}^l\}_G$ indicates a variation term which - for each simulation run $l = 1 \dots 3$ - is applied to a different channel of the velocity trim vector, such that:

$$\{\Delta\mathbf{V}^1\}_G = \begin{bmatrix} \Delta V(t) \\ 0 \\ 0 \end{bmatrix} \qquad \{\Delta\mathbf{V}^2\}_G = \begin{bmatrix} 0 \\ \Delta V(t) \\ 0 \end{bmatrix} \qquad \{\Delta\mathbf{V}^3\}_G = \begin{bmatrix} 0 \\ 0 \\ \Delta V(t) \end{bmatrix} \tag{9.10}$$

Here the velocity perturbation element $\Delta V(t)$ is a user-defined time-dependant velocity deviation. In an effort to introduce both positive and negative deviations from the trim velocity, the velocity signals were defined each over a $t_{max} = 100$s duration and have the equation defined in Equation 9.11. A graphical representation of this is provided in Figure 9.12. The resulting velocity signals are then integrated over time to generate accurate position reference signals for controller tuning.

$$\Delta V(t) = \begin{cases} 10 & \text{if} \quad 10 < t \leq 20 \ \vee \ 60 < t \leq 70 \\ -10 & \text{if} \quad 30 < t \leq 50 \\ 0 & \text{else} \end{cases} \tag{9.11}$$

The position references calculated using this method are shown in Figure 9.13, where the columns indicate the different simulation runs, and the rows indicate the three available position channels X, Y, and Z.

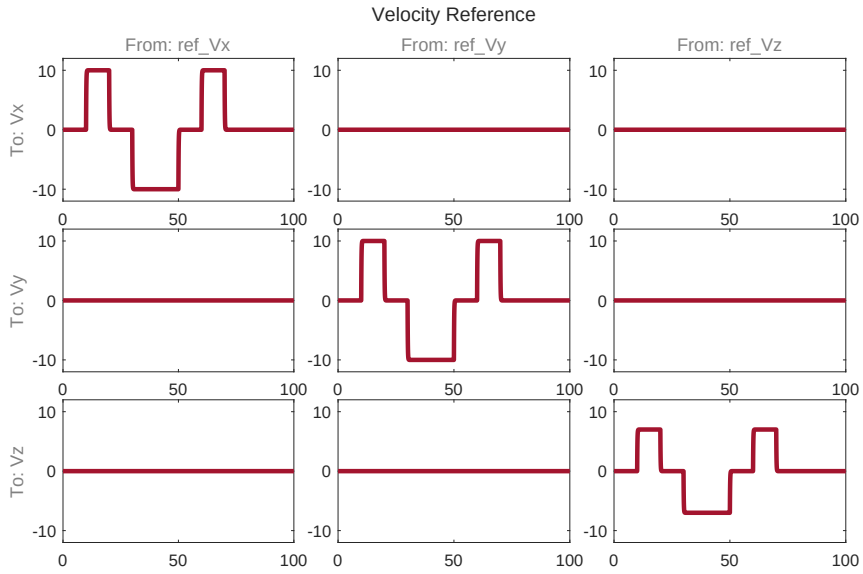


Figure 9.12: Velocity perturbation term $\Delta V(t)$ used in position reference generation

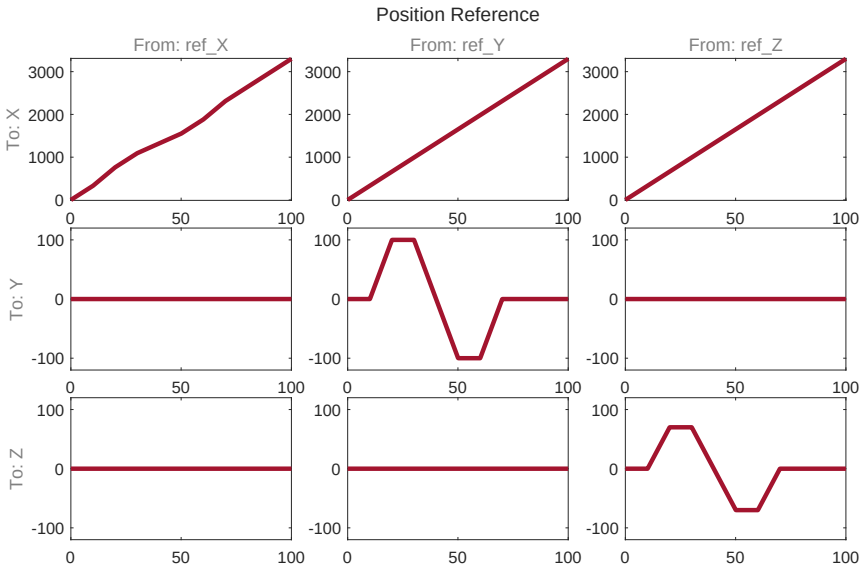


Figure 9.13: Position references for the position controller tuning

With the procedure followed to generate the reference signals for the various simulation runs l, the cost function used in the PSO tuning algorithm will be discussed hereafter. Similarly to the tuning process followed for both the attitude and velocity controllers, the cost function is defined by a number of terms, each encoding a desired aspect of the controlled system's behaviour: to achieve this, the cost function used for the tuning of the position controller is given in Equation 9.12.

$$
\begin{aligned}
z = &\sum_{l=1}^{3} W_{Rp} \left(\sum_{t=1}^{T} (X_r^l(t) - X^l(t))^2 + \sum_{t=1}^{T} (Y_r^l(t) - Y^l(t))^2 + \sum_{t=1}^{T} (Z_r^l(t) - Z^l(t))^2 \right) + ... \\
&\sum_{l=1}^{3} W_{Rv} \left(\sum_{t=1}^{T} (u_r^l(t) - u^l(t))^2 + \sum_{t=1}^{T} (v_r^l(t) - v^l(t))^2 + \sum_{t=1}^{T} (w_r^l(t) - w^l(t))^2 \right) + ... \\
&\sum_{l=1}^{3} W_{Ra} \left(\sum_{t=1}^{T} (\Delta q_0^l(t))^2 + \sum_{t=1}^{T} (\Delta q_i^l(t)(t))^2 + \sum_{t=1}^{T} (\Delta q_j^l(t))^2 + \sum_{t=1}^{T} (\Delta q_k^l(t))^2 \right) + ... \\
&\sum_{l=1}^{3} \left(\sum_{t=1}^{T} \left(W_x \delta x^l(t)^2 + W_y \delta y^l(t)^2 + W_p \delta p^l(t)^2 + W_0 \delta_0^l(t)^2 \right) \right)
\end{aligned}
\tag{9.12}
$$

The first term of the equation aims at ensuring position reference tracking by minimizing the associated tracking MSE: in this manner, it is ensured that the control system is capable of tracking reference trajectory signals generated by the autopilot module.

The second and third terms, on the other hand, are tasked with ensuring the position controller does not interfere with the functioning of the attitude and velocity loops by minimizing the tracking errors on their respective reference commands. The application of these terms was found to be beneficial for the general closed loop system behaviour and ensured the position gains do not cause malfunctioning in the inner loops.

The fourth term, similarly to the cost function discussed in Equation 9.8, ensures the position controller does not saturate the controller commands while tracking the reference signals. This is ensured by once again applying different weights depending on the commanded values, in a manner analogous to the one discussed in Section 9.2.2.

The weights were selected to prioritize the performance of the position controller, assigning a large penalization to the position MSE and adjusting the weights on the velocity and attitude MSEs to have comparable contribution to the cost function. As one of the objectives of the control system is also to avoid command saturation, the weight on the outermost band of the control saturation is set to be extremely large to discourage solutions that would lead to excessive control usage, similarly to what was done in the previous tunings of the system.

9.4.2. Position tuning: results of the tuning

With the structure of the control system defined and the implementation of the PSO optimization discussed, the results of the controller tuning are discussed hereafter. The tuning was performed on the complete closed-loop model shown in Figure 8.1; the algorithm was initialized over a population of 30 random PSO agents and a total of 50 generations. This allows for a good exploration of the solution space while also giving time to the optimization algorithm to converge to an optimum solution.

The results of the tuning are shown in Equation 9.12, where the top of the figure shows the evolution of the cost over the various generations of the algorithm and the bottom figure details the contribution of the various terms of the cost function. The first figure clearly shows the model's converging behaviour, meaning that the PSO algorithm has successfully stabilized at an optimum solution: once again, the cost trend over the iterations mimics the typical behaviour of metaheuristic models, with the largest improvements being located at the beginning of the iterative process and more marginal improvements at higher iterations once an optimal solution has been located. The second figure shows the magnitude of the individual terms of the cost function: again, the contribution of the various terms is comparable to one another, suggesting that the optimization has found a best compromise between the multiple objectives of the cost function. The largest contribution is provided by the position tracking error term (which is also the primary objective of the control system), while the velocity and attitude tracking errors are comparatively more limited in magnitude. The input penalization is comparable to the velocity and attitude penalizations, suggesting that the tuning process was able to converge to a solution that does not incur in command saturation in the representative manoeuvres used to generate the tuning simulations.

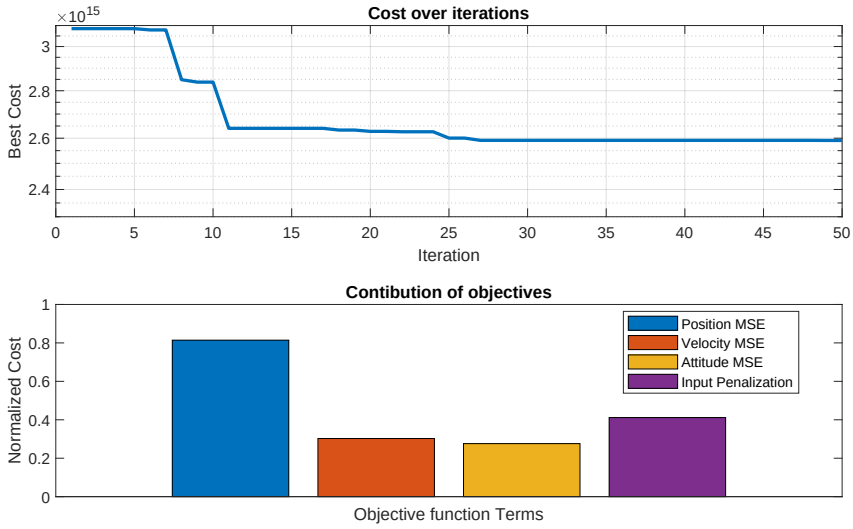


Figure 9.14: Evolution of the cost function of the position controller tuning

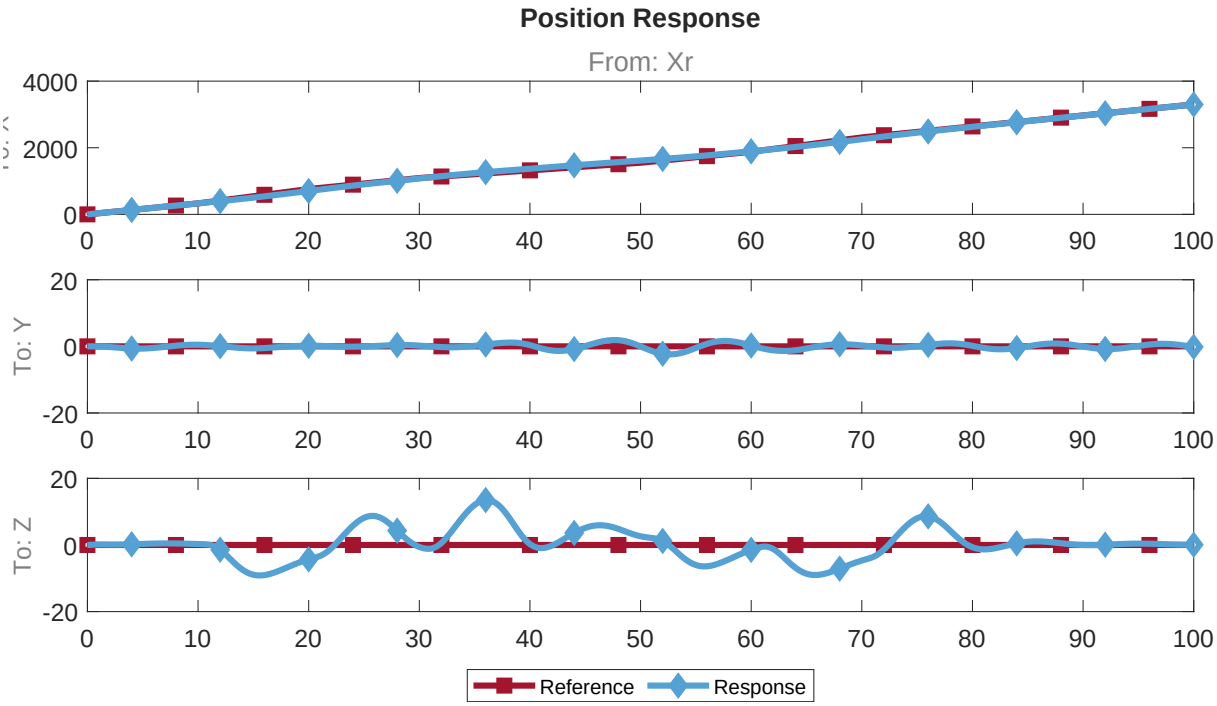
To further show the results of the tuning process, the closed-loop responses of the helicopter for the different simulated maneuvers are shown in Figures 9.15 to 9.17. The tuning process showed positive results, with the controlled system capable of following the commanded position signals over the three simulations. When performing the tuning process, the terms included on the velocity and attitude error penalizations proved especially useful, as they ensured that the position controller produced coherent signals that would not excessively load the two inner loops, ensuring a coherent response in all the channels.

Analysing the individual simulations, the cross-coupling behaviours noticed in the velocity controller tuning remain apparent, with position commands along the X and Y directions also exciting a noticeable response in the Z direction. The negative effects of the cross couplings are also identifiable in the control actuations, with all manoeuvres requiring some use of the collective command δ_0 to maintain the appropriate altitude. Additionally, off-axis controls of more moderate magnitude are noticed on the lateral cyclic δ_y and in the pedal δ_p when exciting the X control axis and, similarly, on the longitudinal cyclic δ_x when exciting the Y control axis. As was also discussed in the velocity controller, these behaviours are reflective of the cross-couplings identified during the study of the system's eigenmodes in Section 6.2.1 and are further exacerbated by delays in the velocity controller response.

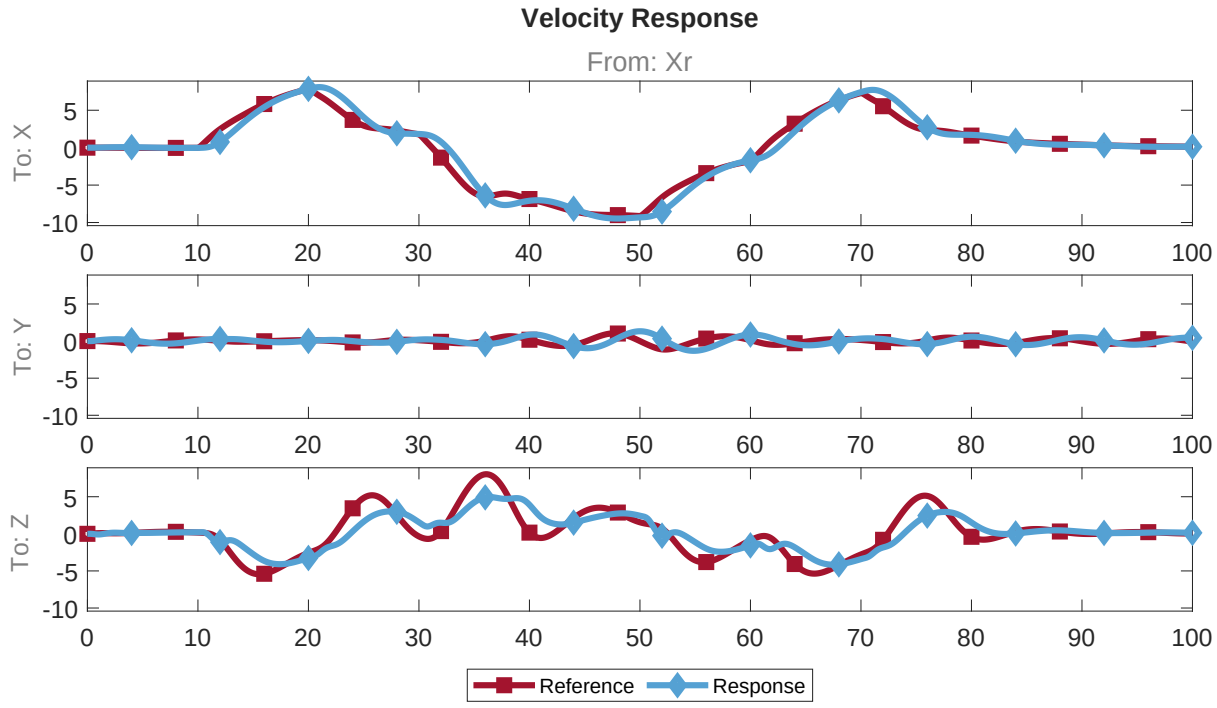
The smallest cross-coupling between the position channels is noticed when altitude inputs are provided to the system, leaving the other axes untouched: this condition is shown in Figure 9.17. This is consistent with the results shown in Figure 9.11, where inputs along the V_z direction are tracked accurately with limited coupling along the V_x and V_y axes.

As expected from the analysis of the cost function terms, the actuator deflections never reach complete saturation, instead remaining within the allowed control margins: in all the simulations good tracking performances were obtained with limited control usage, suggesting the viability of this tuning strategy when handling complex and highly coupled systems.

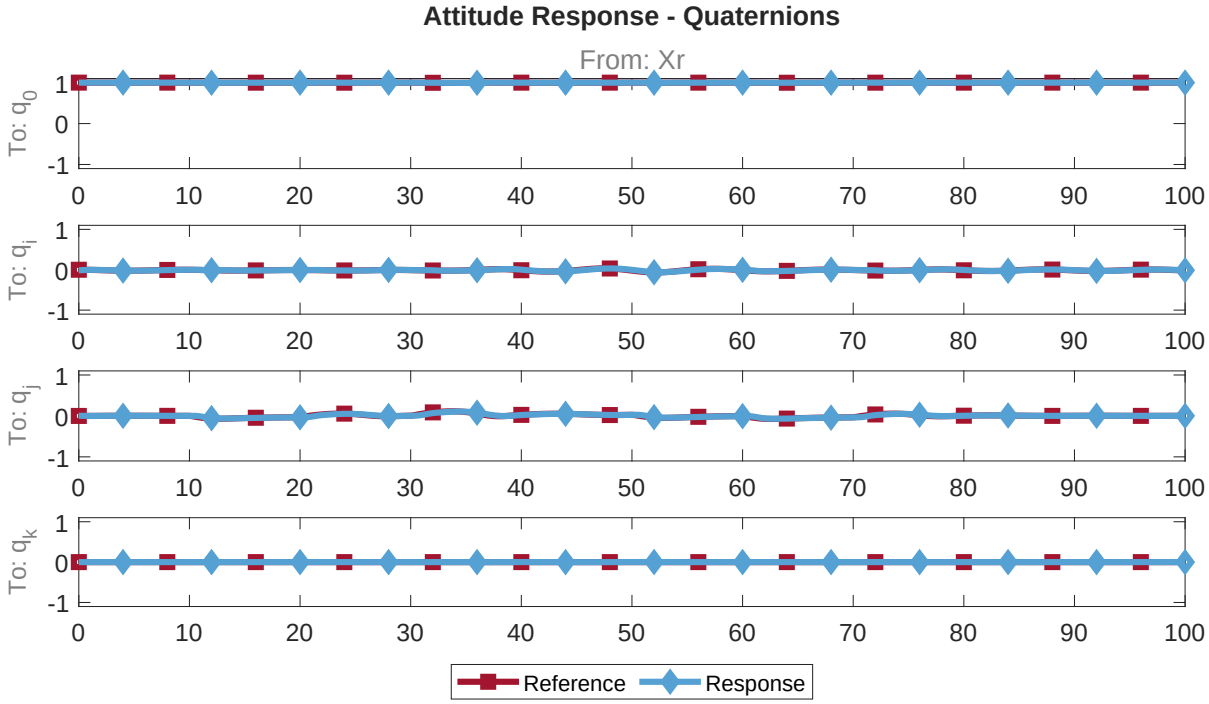
Overall, the position controller tuning procedure successfully allowed for the definition of a control strategy capable of tracking user-defined velocity commands by leveraging the good performance of the innermost velocity and attitude loops while always maintaining moderate control usage. By analysing the system and its response, the cross-couplings between the various control axes can be led back to the identified system modes discussed in Section 6.2.1 and to the delays in the velocity command response. The most immediate way of addressing this is via the introduction of an additional derivative control action in the inner loops, which would however make the controls more heavily susceptible to perturbations and noise and would likely cause a dynamic destabilization effect due to rapidly changing commands and control saturation.



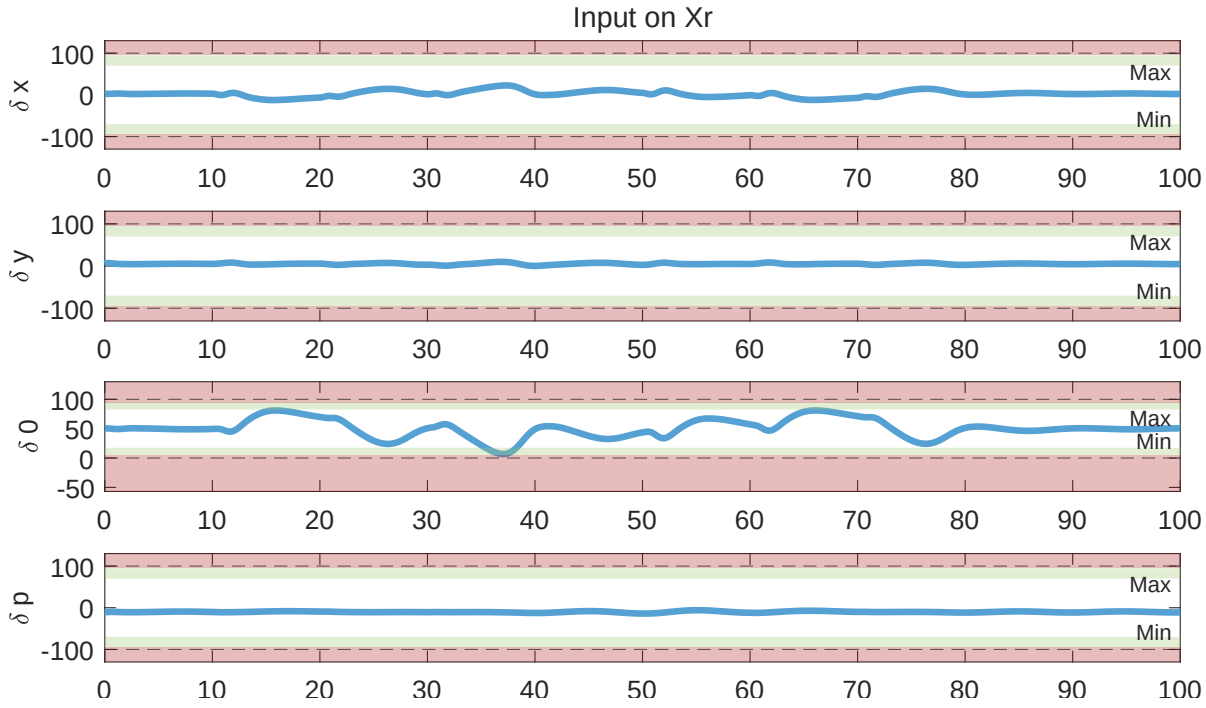
(a) First simulation - position signals



(b) First simulation - velocity signals

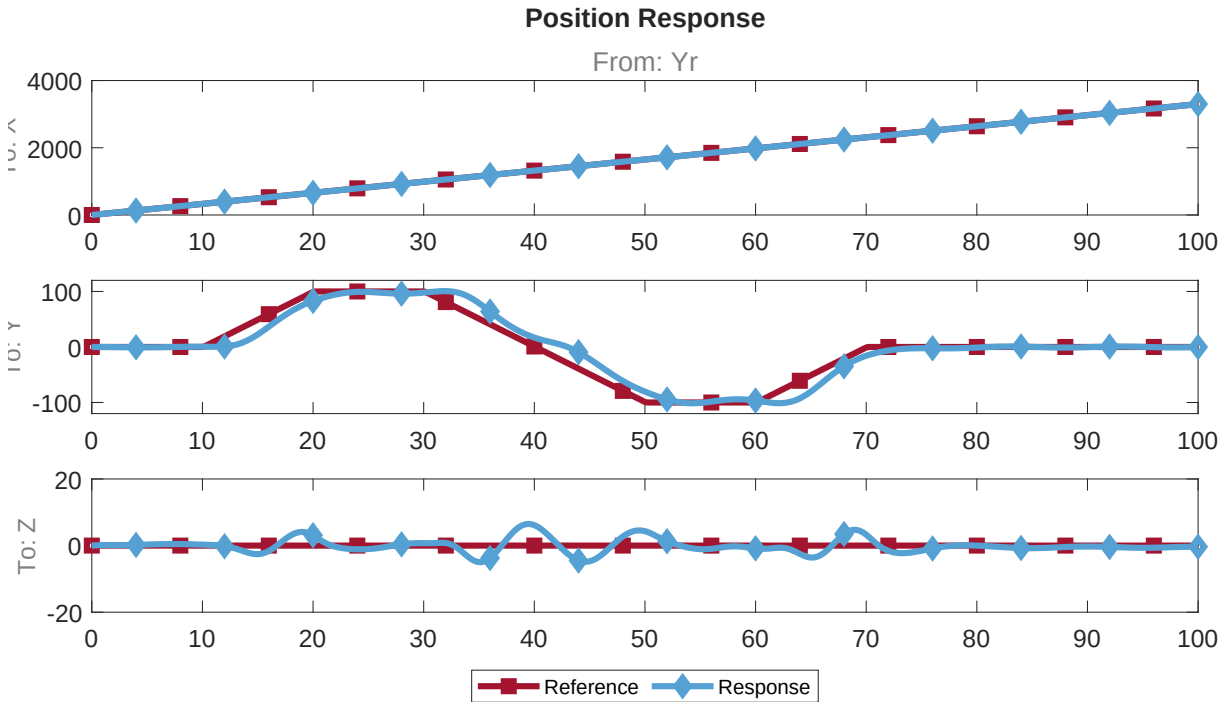


(c) First simulation - attitude signals

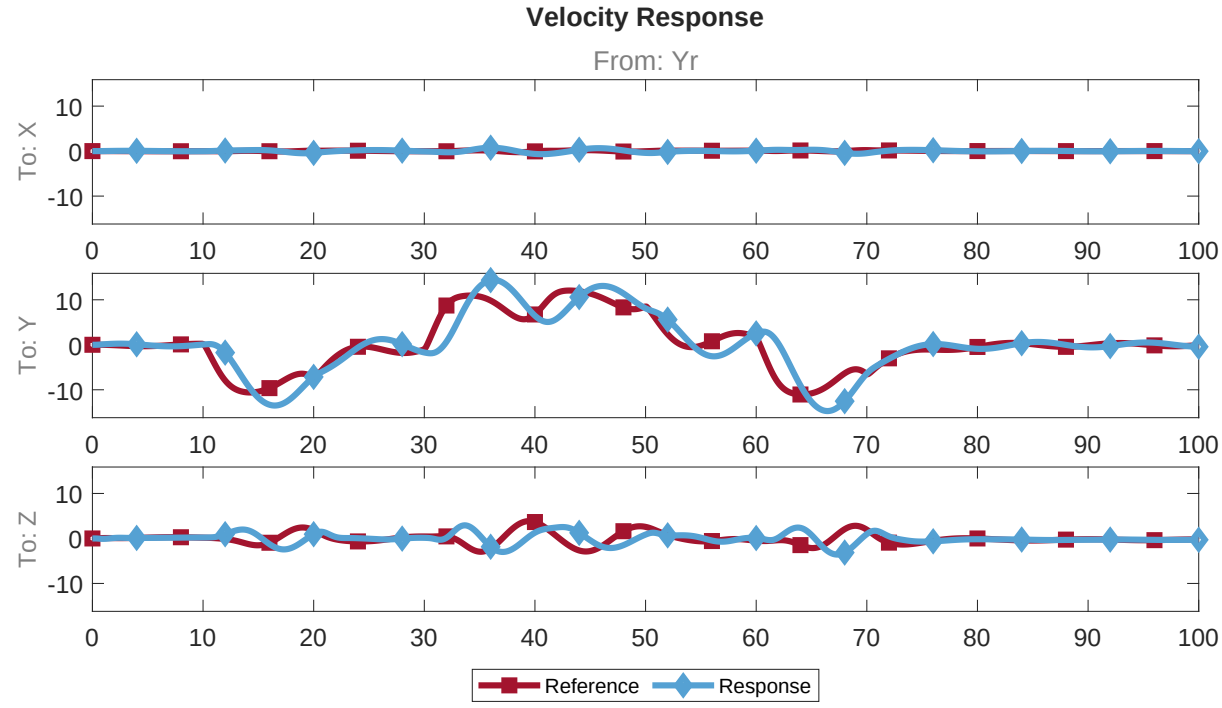


(d) First simulation - control signals

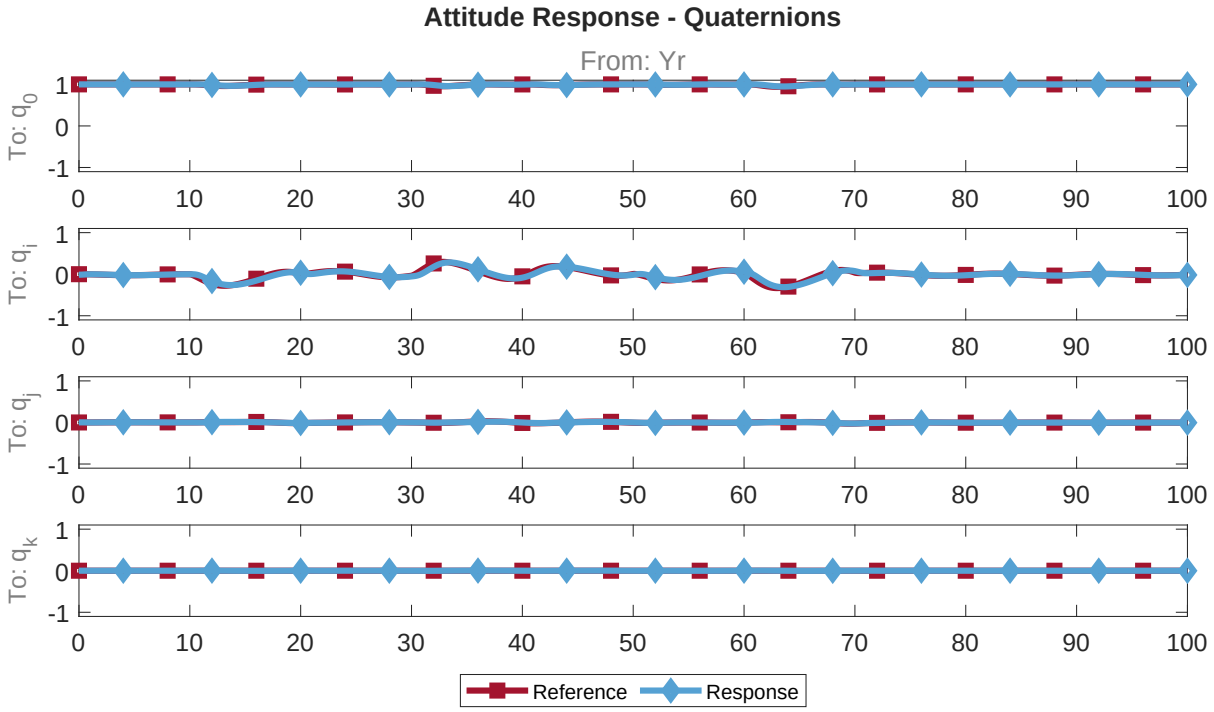
Figure 9.15: First simulation: longitudinal acceleration and deceleration



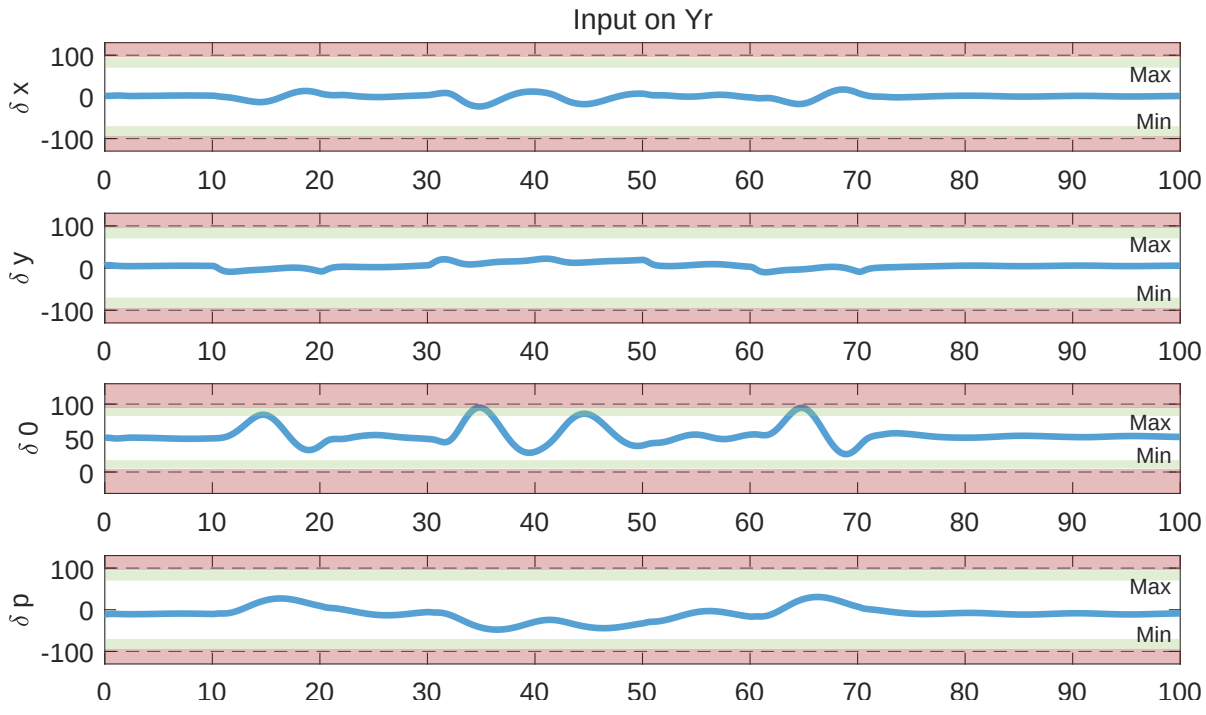
(c) Second simulation - position signals



(d) Second simulation - velocity signals

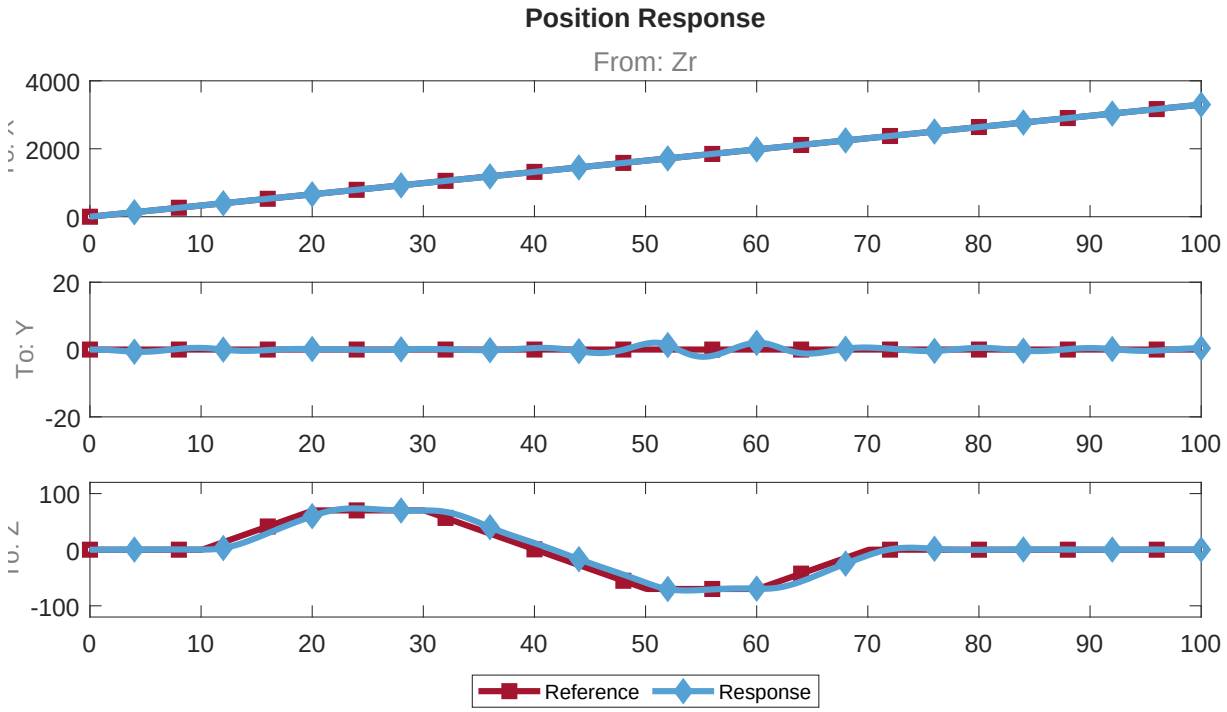


(a) Second simulation - attitude signals

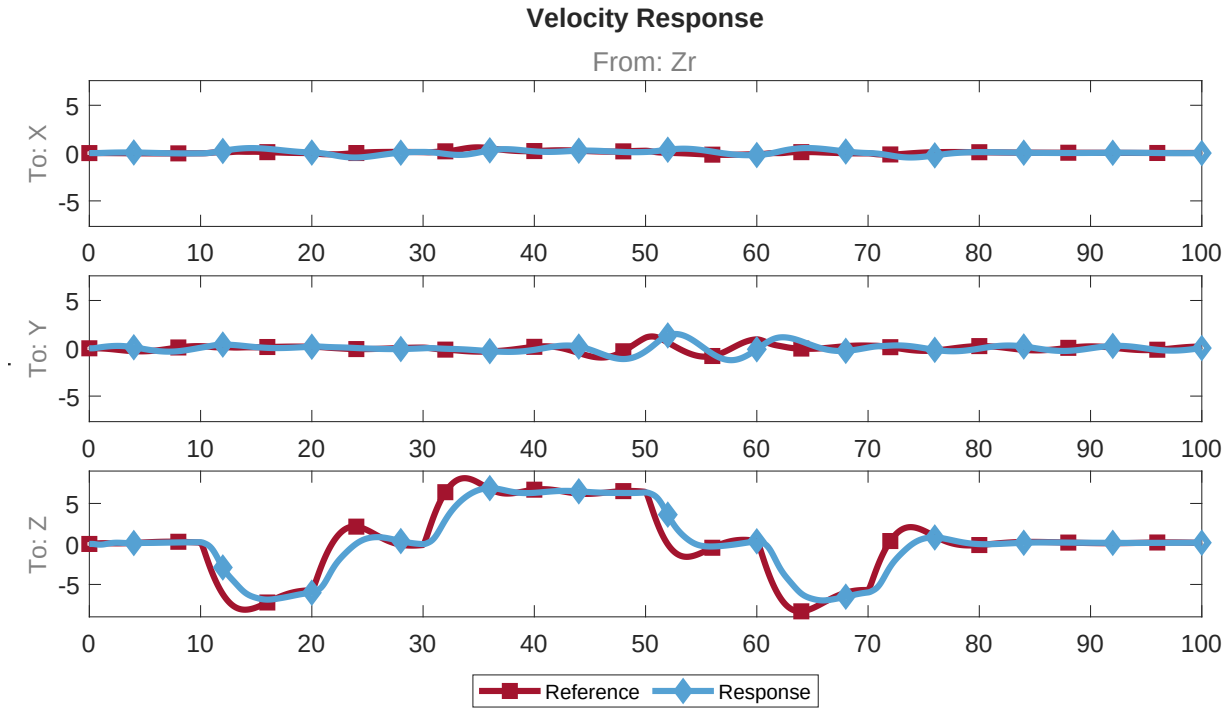


(b) Second simulation - control signals

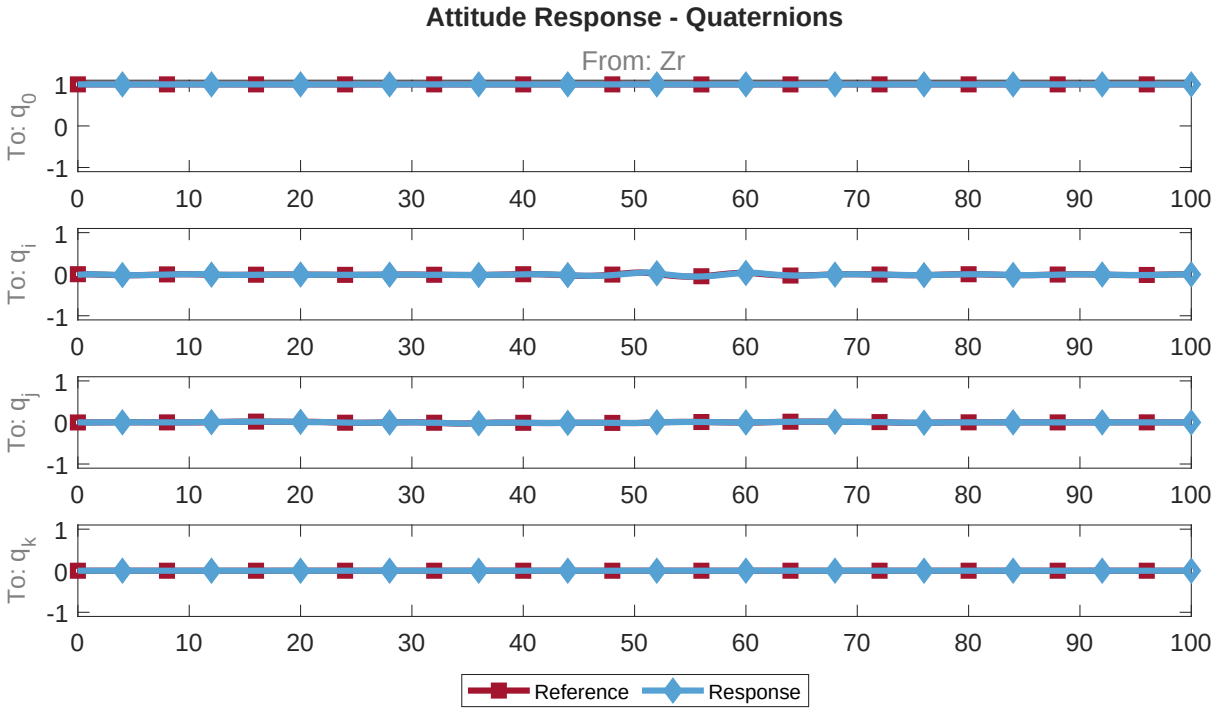
Figure 9.16: Second simulation: lateral acceleration and deceleration



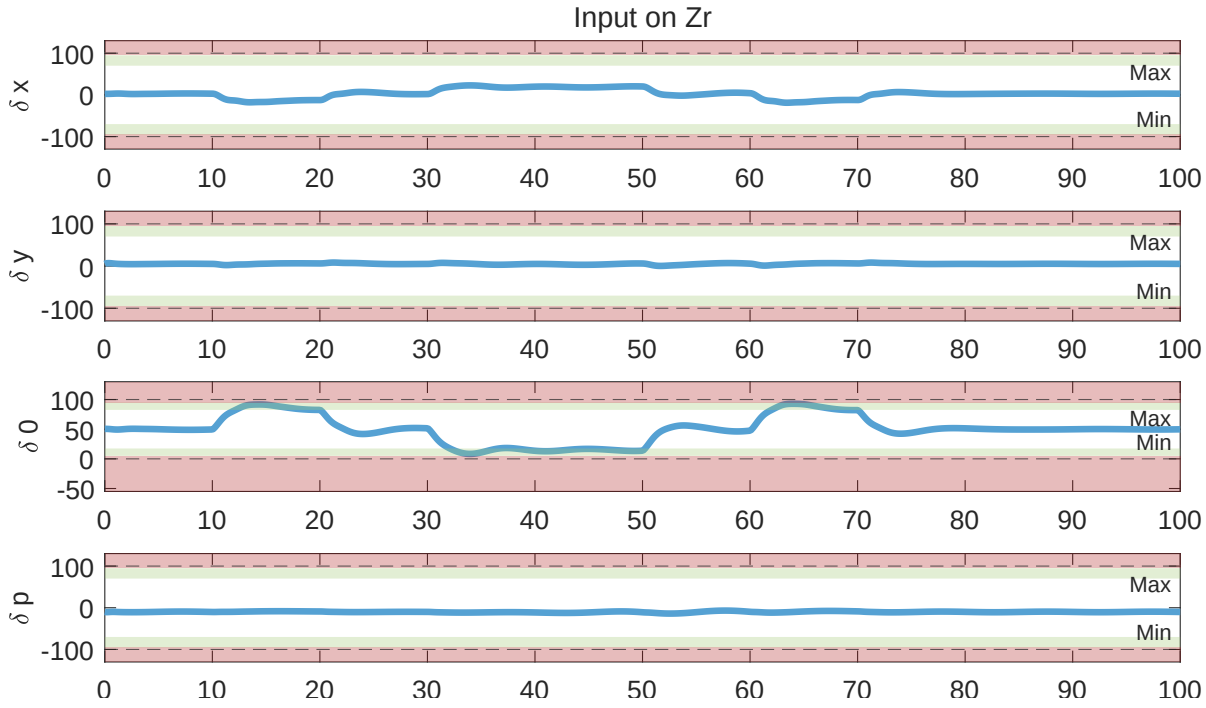
(a) Third simulation - position signals



(b) Third simulation - velocity signals



(c) Third simulation - attitude signals



(d) Third simulation - control signals

Figure 9.17: Third simulation: vertical acceleration and deceleration

9.5. Final considerations on the controller tuning

Within this chapter the tuning of the controller was discussed in its entirety, providing a comprehensive understanding of the procedure used and of its implementation. Here, a brief summary and outline of the contents of the chapter is provided.

In an effort to provide a systematic and repeatable approach to controller tuning that could also be generalized to other control-system architectures, the tuning process followed in the context of this thesis made use of the Particle Swarm Optimization (PSO) algorithm, which was introduced in Section 9.1. The PSO algorithm had already been proven adequate for controller tuning, often obtaining better results than more complex optimization algorithms while also having a sizeable computational cost advantage [93]. Further, the multimodal capabilities of the algorithm and the ability to define an arbitrarily complex cost function allowed for the simultaneous optimization of contrasting objectives, an essential characteristic when tuning controllers that need to balance a fast response with a moderate control usage.

Leveraging the flexibility and the simplicity of the PSO algorithm, the tuning of the controllers was discussed afterwards, starting from the innermost attitude loop and then moving outwards to tune the velocity and position loops.

The attitude controller tuning was discussed first in Section 9.2, discussing the implementation of the PSO algorithm and the definition of the cost function. As this control loop is in direct contact with the helicopter model and given that the model is a coupled MIMO system, the attitude controller was developed using the LQI framework to simultaneously achieve good tracking performance while also allowing for a measure of decoupling between the attitude channels. The flexibility of the PSO algorithm enabled the optimization of the controller performance, producing a controller capable of following rapid and substantial attitude changes without incurring in control saturation.

The velocity controller tuning was analysed in Section 9.3. Building upon the results of the attitude loop tuning, the velocity controller was developed using the simpler and more widely adopted PI architecture. Even in this case, the flexibility of the PSO tuning algorithm was instrumental as it allowed for the definition of the most appropriate gains for the velocity controller that could allow for tracking of velocity commands while ensuring the functioning of the attitude inner loop and limiting control usage. The tuned system was successful in tracking velocity reference commands with moderate actuator deflection, but showed noticeable couplings, especially considering the large cross-activation of the vertical velocity channel when inputs are provided on the horizontal velocity components.

Lastly, the tuning of the position controller was discussed in Section 9.4, where the PSO algorithm was applied to the selection of the appropriate proportional gains for the outer loop position controller. Once again, the tuning procedure established for the two inner loops was successful in obtaining controller gains that could provide stable tracking of position commands, without interfering with the functioning of the inner loops and limiting control usage. Even in this case, some cross-activation was noticed, caused by the delays in the velocity response and by the inherent dynamic modes of the helicopter.

In all three test cases, the results of the tuning verify the proposed methodology for helicopter controller development: the PSO algorithm consistently achieved a balanced trade-off between tracking performance and control effort, while remaining an understandable and versatile tuning approach. Overall, the optimization-based tuning strategy developed in this thesis has proven both effective and flexible, allowing the user to define desired closed-loop characteristics and multiple counteracting objectives.

The full flight control system developed using this approach will serve as a basis for the remaining parts of the the thesis. In particular, the closed loop system defined in such manner will be augmented with a simple autopilot subsystem and manoeuvres will be identified and calculated offline in order to create more complex tasks for the evaluation of closed-loop performance.

Part VI

Definition of manoeuvres and autonomous flight

10

Helicopter manoeuvres synthesis

In the previous chapters the control system for the Bo105 helicopter has been developed and its tuning discussed, applying a structured optimization methodology to determine the most appropriate tunable parameters to obtain a desired system behaviour. With an appropriate closed-loop system designed and tuned, its capabilities are to be evaluated by having the helicopter follow a set of reference manoeuvres aimed at determining its performances.

The issue of trajectory generation and path planning has been considered and confronted a number of times in literature, with a number of different approaches and optimization techniques developed to take into account various mission parameters. Many such strategies make use of complex dynamical systems [100] to identify suitable references and command sets that remain within the operating limits of the system, while other use data-driven probabilistic approaches [101] to produce optimal trajectories that can drive air vehicles in challenging environments. While these approaches are certainly of interest and capable of producing accurate trajectories in complex flight situations, they have the fundamental limitation of requiring a very precise knowledge of the system's behaviour in all relevant regions of the flight envelope (which would require a large modelling effort to have an accurate mathematical description of the system in various flight conditions) and to have access to a large number of flight data (which would require large and expensive simulations campaigns or special access to obtain appropriate information on the systems).

To avoid relying on such complex models and equations and streamline the simulation process, for the purpose of this thesis the trajectory references are generated using simple kinematic and dynamic equations to define a path and a set of corresponding velocities and accelerations in 3D space to drive the motion of the system [102, 57]. With an acceleration profile defined throughout the manoeuvre, appropriate attitude references are then generated to allow the helicopter to follow the desired trajectory with a specified heading.

This chapter will be centred around the definition of a methodology to properly define manoeuvres in 3D space for the helicopter to follow. To start, in Section 10.1 the baseline procedure followed in the generation of the manoeuvres is outlined, clearly detailing the steps taken in defining the required reference values for position, velocity and attitude. Afterwards, in Section 10.2, reference manoeuvres will be selected for the purpose of controller performance evaluation: the slalom and the pop-up manoeuvres. These manoeuvres were identified to determine the control system's capacity to fly while tracking offline-created reference signals.

10.1. Baseline trajectory synthesis procedure

The issue of generating appropriate manoeuvre references is a critical task in motion planning as the generated reference values have to be correct and coherent with the general capabilities of the system. Within this work, in an effort to provide an effective and replicable methodology to trajectory generation, a systematic approach to produce appropriate references is discussed and provided, leveraging the properties of quaternions and of simple dynamic systems to generate reference trajectory, velocity and attitude signals to drive the motion of the helicopter.

As discussed in Chapter 8, the closed loop system comprised of the helicopter and the control system is equipped to make use of references in the position channels $\mathbf{X}^\star$, the velocity channels $\mathbf{V}^\star$, and the attitude channels $\mathbf{q}^\star_{E2B}$: the problem of trajectory generation therefore is defined by outlining a methodology that can allow for the definition of these variables.

The first step in the trajectory generation approach is the identification of an appropriate position and velocity profile to drive the motion of the system: as these variables are strictly related to each other, their identification is a simple matter of defining a trajectory over time in 3D space and differentiating to extrapolate the required velocity signals. This differential relationship between velocity and position encourages the formulation of the problem in terms of differential equations (as will be further exemplified in Section 10.2), which also facilitates the seamless introduction of additional constraints to the manoeuvre (e.g.: constant total velocity, following of a pre-defined velocity profile, tracking of a specified load factor profile). Note that, while only position $\mathbf{X}^\star$ and velocity $\mathbf{V}^\star$ references are required to drive the closed loop system in Chapter 8, in this step the acceleration profile $\mathbf{A}^\star$ is also calculated for the purpose of attitude feed-forward calculation.

With the position and velocity signals identified to drive the motion of the system in space, the generation of appropriate attitude references to enable this motion is required. In helicopter flight, attitude changes are primarily used to change the direction of the helicopter's main rotor thrust, enabling accelerations in the horizontal plane: to reflect this, the reference attitude is calculated to align the helicopter's main rotor thrust to the acceleration profile defined previously.

For helicopter flight, the direction and magnitude of the main rotor thrust vector is determined by the interaction of a vertical and a horizontal component: the vertical component is the thrust force T, perpendicular to the tip path plane (TPP), which represents the vertical force component generated by the motion of each blade element over the rotation of the main rotor; the horizontal force component is determined by the H-forces H_i and H_0 - parallel to the TPP - which represent the difference in inclination of the lift and of the profile drag on the left and right side of the main rotor hub respectively [103].

To simplify trajectory determination, here the assumption that the main rotor thrust is aligned to the body z axis [80, 103, 104] is considered. This hypothesis is primarily motivated by two fundamental assumptions: that the magnitude of the H-forces is much smaller than the magnitude of the main rotor thrust, and that the inclination of the TPP can be identified by a small relative angle. While these simplifications may not hold in general cases when studying helicopter motion, for the purpose of trajectory generation they are only used to produce feed-forward attitude signals coherent with the commanded manoeuvre: the attitude signals generated in this manner therefore are only used to speed up the closed-loop system, and any discrepancy with the desired model behaviour will be compensated for by the velocity controller's corrective attitude signal.

It is worth noting that, since the objective of this procedure is to generate coherent reference trajectories rather than compute realistic system responses or control inputs, the use of a point-mass model is not required, as this approach relies only on desired velocity profiles and their derivatives to infer compatible attitude references. The simulation of the trajectory's execution is performed separately using the complete linearized model, as detailed in Chapter 12, thereby ensuring that the system response accounts for the full set of dynamic effects.

To identify the feed-forward attitude $\mathbf{q}^\star_{E2B}$, first the total acceleration $\mathbf{A}_T$ required to maintain course is calculated. This is calculated by subtracting the gravitational acceleration term $g = -9.81\mathrm{m/s^2}$ to the acceleration $\mathbf{A}^\star$ calculated in the previous trajectory generation step. As both $\mathbf{A}^\star$ and the gravitational acceleration are both measured in the global reference frame $\mathcal{G}$, the total acceleration is calculated according to Equation 10.1.

$$\{\mathbf{A}_T\}_G = \{\mathbf{A}^\star\}_G - \left\{ \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right\}_G \tag{10.1}$$

As the quaternion attitude is always expressed relative to the NED frame $\mathcal{E}$ - the total acceleration is then expressed in the $\mathcal{E}$ reference frame by means of a passive rotation, obtaining $\{\mathbf{A}_T\}_E$.

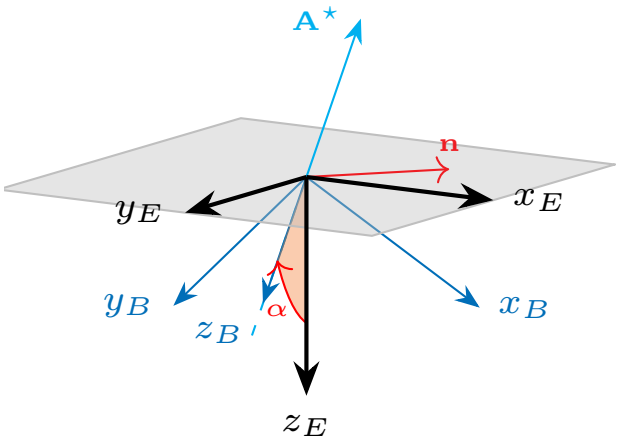


Figure 10.1: Representation of α and **n** for the determination of the helicopter's desired attitude around a manoeuvre, the horizontal plane defined by vectors x_E and y_E has been highlighted in gray

With the acceleration identified as a vector in 3D space, an appropriate rotation has to be identified to align the helicopter's body z axis to the total acceleration vector: this is equivalent to determining the rotation that aligns the z axis of the $\mathcal{E}$ reference frame to the opposite of the required acceleration (as in a general flight condition the helicopter's thrust is directed upwards while the body z axis is positive downwards).

The problem of identifying the rotation required to align two vectors is easily determined and solved by using quaternions and leveraging the linearity of their operations [105]. First, let $\{\mathbf{z}\}_E$ be the z axis of the NED reference frame, such that:

$$\mathbf{z}_E = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}_E \tag{10.2}$$

and, to simplify calculations, consider the normalized acceleration vector:

$$\{\mathbf{a}_T\}_E = \frac{\{\mathbf{A}_T\}_E}{\|\mathbf{A}_T\|} \tag{10.3}$$

To align $\{\mathbf{z}\}_E$ to $\{-\mathbf{a}_T\}_E$ two parameters need to be identified: the enclosed angle α between the two vectors and the orthogonal direction **n**. A graphical representation of these two values is provided in Figure 10.1. By then leveraging the definition of quaternions provided in Equation 5.5, the desired attitude is immediately identified by computing the quaternion $\hat{\mathbf{q}}^{\star}_{E2B}$ presented in Equation 10.4.

$$\hat{\mathbf{q}}^{\star}_{E2B} = \begin{bmatrix} \cos(\alpha/2) \\ \sin(\alpha/2)\mathbf{n} \end{bmatrix} \tag{10.4}$$

The dot product of the NED z axis and of the normalized total acceleration vector is defined as:

$$\mathbf{z}_E \cdot (-\mathbf{a}_T) = \|\mathbf{z}_E\| \| -\mathbf{a}_T \| \cos(\alpha) = \cos(\alpha) \tag{10.5}$$

By then leveraging the trigonometric properties of half angles, the sine and cosine required to compute $\hat{\mathbf{q}}^{\star}_{E2B}$ are immediately determined by means of the equations:

$$\cos(\alpha/2) = \sqrt{\frac{1 + \cos(\alpha)}{2}} \tag{10.6}$$

$$\sin(\alpha/2) = \sqrt{\frac{1 - \cos(\alpha)}{2}} \tag{10.7}$$

The last remaining component of the quaternion, **n**, is trivially identified by computing the cross product of $\mathbf{z}_E$ and $-\mathbf{a}_T$:

$$\mathbf{n} = \mathbf{z}_E \times -\mathbf{a}_T \tag{10.8}$$

Notice that the attitude quaternion identified in this manner $\hat{\mathbf{q}}^{\star}_{E2B}$ is still not the complete desired attitude quaternion, as it has only constrained the helicopter's body z axis to be aligned to the required acceleration of the manoeuvre but encodes no information about the desired orientation of the helicopter's body around its z axis. While this step may not be required in a general case as the control system presented in Chapter 8 could easily be augmented with a sideslip angle controller, to fully leverage the added manoeuvrability of the helicopter around its vertical axis here the desired reference yaw angle is computed offline to drive the motion of the system.

To complete the calculation of the desired attitude, the quaternion $\mathbf{q}^{\star}_{\xi}$ describing a rotation around the new body z axis is calculated. As the quaternion only describes a rotation around the z axis of the system, it will have the following equation:

$$\mathbf{q}^{\star}_{\xi} = \begin{bmatrix} \cos(\xi/2) \\ 0 \\ 0 \\ \sin(\xi/2) \end{bmatrix} \tag{10.9}$$

Different approaches may be used for the calculation of this angle and these will enable very different behaviours from the helicopter itself. For example, $\mathbf{q}^{\star}_{\xi}$ may be defined to ensure the vehicle maintains the same yaw angle throughout the manoeuvre or it may be defined to minimize a certain sideslip angle (calculated offline from the desired velocity). Once the appropriate quaternion has been determined, the complete desired attitude is immediately calculated using the rotation concatenation property:

$$\mathbf{q}^{\star}_{E2B} = \hat{\mathbf{q}}^{\star}_{E2B}\mathbf{q}^{\star}_{\xi} \tag{10.10}$$

With this, the procedure used to determine the desired position, velocity and attitude signals to drive the motion of the system has been outlined, discussing the approach used for the generation of all values and the assumptions adopted in generating the desired attitude signal. Expanding upon these results, Section 10.2 will apply this methodology for the generation of appropriate reference manoeuvres which will later be used to test the capabilities of the closed-loop system.

10.2. Definition of the evaluation manoeuvres

To properly evaluate the performance of the controller and the helicopter's capacity for autonomous flight, a set of well-defined flight control tasks must be established: the set of tasks identified for the purpose of evaluating the performance and capabilities of a system compose the Mission Task Elements (MTEs), a systematic framework for delineating the specific operational scenarios that the helicopter is expected to encounter during its operations. MTEs comprise distinct tasks that encapsulate critical flight objectives and challenges, ranging from precision manoeuvreing to transitions between various flight regimes, and as such defining appropriate control tasks for a system is of crucial importance.

The selection of appropriate MTEs is governed by an analysis of mission requirements and safety considerations, to provide a metric of the capabilities of a vehicle to perform aggressive tasks and manoeuvres. Typically, MTEs are categorized in specific manuals and regulations, such as the Aeronautical Design Standard – 33 (ADS-33) [106] and the Utility Tactical Transport Aircraft System (UTTAS) manoeuvre Criteria [107], that are used in industry as standards of vehicle performance.

In this section, the derivation of the reference trajectories for two key manoeuvres is discussed: the slalom and the pop-up. These manoeuvres were selected to evaluate both the lateral agility and vertical dynamic performance of the helicopter's quaternion-based control system, requiring simultaneous changes in velocity and attitude while precisely tracking reference signals. In particular, the slalom manoeuvre challenges the control system's ability to handle rapid, oscillatory lateral deviations, thereby testing its agility and responsiveness. Conversely, the pop-up manoeuvre targets the vertical dynamics of the helicopter, assessing its capacity to execute altitude transitions. These manoeuvres not only represent realistic flight scenarios but also provide a structured framework for examining the performance of the controller under aggressive and dynamic conditions: by using them, the controller can be tested in realistic flight challenges that emphasize the controller's ability to track complex reference trajectories in both the horizontal and vertical planes while ensuring smooth transitions in position, velocity, and attitude.

When identifying the manoeuvre references, two valid approaches will be presented. For the slalom manoeuvre, the trajectory and velocity will be obtained by connecting three separate trajectory segments without imposing any additional continuity constraints: this will generate discontinuous references for the velocity and attitude variables when changing between the manoeuvre's segments. Conversely, the pop-up manoeuvre will be generated using a continuous fitting polynomial approach, which will guarantee a smoother trajectory and references. While the latter approach focuses on reference quality and flexibility, the former approach focuses on ease of implementation, as no system needs to be solved to identify the reference variables. Both these approaches are valid and will not cause great losses in the helicopter's performance, as any discontinuity in the references will be mitigated by the integrated autopilot module, which will perform real-time interpolation between segment boundaries (as will be better explained in Section 11.1.2). This online interpolation smooths abrupt transitions and also dynamically adjusts reference signals to maintain continuity in the control loops, ensuring the autonomous system is robust to errors in the trajectory definition segment.

10.2.1. Slalom manoeuvre

The slalom manoeuvre entails a series of oscillatory lateral movements superimposed on a constant forward velocity. This dynamic task requires continuous adjustments in roll, pitch, and yaw, as the helicopter negotiates alternating lateral deviations: because of these characteristics, such a manoeuvre is particularly effective in evaluating the controller's capability to track rapidly changing reference trajectories and to maintain stability during aggressive lateral manoeuvres. Here the references to command the slalom manoeuvre are derived from a kinematic and dynamic approach.

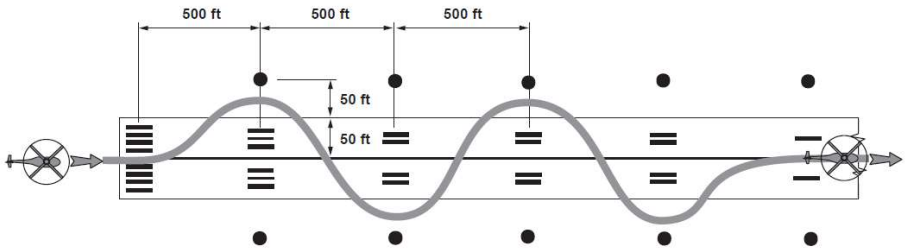


Figure 10.2: Suggested course for slalom manoeuvre [106]

A formal definition of the slalom manoeuvre for the purpose of handling qualities identification is provided in the ADS-33 specifications [106]. The manoeuvre there is defined as follows: "initiate the manoeuvre in level unaccelerated flight and lined up with the centerline of the test course. Perform a series of smooth turns at 500-ft intervals (at least twice to each side of the course). The turns shall be at least 50 ft from the centerline, with a maximum lateral error of 50 ft. The manoeuvre is to be accomplished below the reference altitude. Complete the manoeuvre on the centerline, in coordinated straight flight". The suggested path for this manoeuvre is shown in Figure 10.2. Following the methodology presented in Section 10.1, the reference values for the slalom manoeuvre are identified.

For the purpose of simulation, the manoeuvre path can be defined as a simple sinusoidal trajectory connecting an initial and a final straight sections lasting 10s. The addition of an initial and final forward flight segments has the primary purpose of allowing for the evaluation of the aircraft's ability to maintain its trim condition and as such serves as a demonstration of the closed-loop stability of the system.

To orderly exemplify the reference generation procedure, the manoeuvre will be divided in three segments and the trajectory equations will be derived individually. Segment 1 will represent the initial straight section, segment 2 will indicate the central slalom section, and segment 3 will indicate the final straight section.

Segment 1: manoeuvre entrance

The entrance to the manoeuvre is indicated as a forward flight segment at constant speed. As the model is trimmed at $\|\mathbf{V}\| = 33\text{m/s}$, the manoeuvre is executed at the trim speed.

With this consideration in mind, the reference set of inputs is immediately identified via a set of differential equations. Assuming the helicopter to be facing north in the simulations (such that the body x axis is aligned to the global x axis), the position and velocity references are identified according to the following set of equations:

$$\begin{cases} \dot{X}^\star = \|\mathbf{V}\| \\ \dot{Y}^\star = 0 \\ \dot{Z}^\star = 0 \\ \dot{s}^\star = \|\mathbf{V}\| \end{cases} \tag{10.11}$$

where X, Y, and Z indicate the positions in the global reference frame, and s indicates the trajectory arc length.

The system of equations is solved setting initial conditions. For the purpose of simulation, the initial values set for this system of equations are:

Table 10.1: Initial values of the initial slalom segment

$X_{r,0}$	0m
$Y_{r,0}$	0m
$Z_{r,0}$	70m
$s_{r,0}$	0m

The initial segment is defined for $X = [0, L]$, where $L = 330$m to allow the helicopter to maintain forward fight for the desired 10s interval.

As the entrance segment is a forward flight segment, there will be no variations in the velocity's direction or magnitude: as such, the only a vertical acceleration to counter the gravitational acceleration $g = -9.81$m$/$s is required. Because of this, the acceleration required in the global reference frame is the following:

$$\{\mathbf{A}_T^\star\}_G = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{10.12}$$

Segment 2: slalom

The central segment is the oscillatory segment, where the manoeuvreability of the system will be tested. The equations defining the oscillatory trajectory segment in the global reference frame can therefore be modeled as a sinusoidal path:

$$\begin{cases} X^\star(t) = f(t) \\ Y^\star(t) = A \sin\left(B(X^\star - L)\right) \\ Z^\star(t) = h \end{cases} \tag{10.13}$$

where h indicates the manoeuvre's constant altitude, A represents the amplitude of the trajectory oscillation, B modulated the frequency of the oscillations. In accordance with the manoeuvre proposed in the ADS-33 documentation, the amplitude of the oscillations is set to $A = 25$m, to cover the required 50ft $= 15$m deviation from the center line of the manoeuvre. To allow for the turns to happen at 500ft intervals, the frequency parameter is modulated to be $B = \pi/500\text{ft}^{-1} = \pi/152.4\text{m}^{-1}$. As the slalom segment only starts after $L = 155$m, the argument of the sine function contains the $X^\star - L$ term.

To introduce a velocity dependence, the equations in Equation 10.13 are differentiated with respect to time:

$$\begin{cases} \dot{X}^\star = \dot{f}(t) \\ \dot{Y}^\star = AB \cos\left(B(X^\star - L)\right) \dot{X}^\star \\ \dot{Z}^\star = 0 \end{cases} \tag{10.14}$$

Further, as the available linearized model is trimmed at a speed of 33m/s, to ensure the simulated manoeuvre are representative of realistic manoeuvres, it is beneficial to maintain an approximately constant total speed throughout the path. To obtain this, the differential equations presented above are augmented with an additional condition to maintain a constant total velocity $\|\mathbf{V}\|$.

The total velocity of the vehicle, as there are no components along the global z axis, is easily identified with the following equation:

$$\|\mathbf{V}\| = \sqrt{\dot{X}^2 + \dot{Y}^2} = \sqrt{\dot{X}^2 + (AB\cos(B(X-L))\dot{X})^2} \tag{10.15}$$

Rearranging the equation an expression for $\dot{X}$ is identified:

$$\dot{X} = \frac{\|\mathbf{V}\|}{\sqrt{1 + (AB\cos(B(X-L)))^2}} \tag{10.16}$$

Substituting Equation 10.16 in Equation 10.14, the complete system of equations defining the position and velocity values for the central segment of the slalom manoeuvre is identified, as shown in Equation 10.17. Note that the system was also augmented with the additional differential equation $ds/dt = \|\mathbf{V}\|$, where s is the trajectory arc length.

$$\begin{cases} \dot{X}^\star = \dfrac{\|\mathbf{V}\|}{\sqrt{1 + (AB\cos(B(X^\star - L)))^2}} \\ \dot{Y}^\star = AB\cos\left(B(X^\star - L)\right) \dfrac{\|\mathbf{V}\|}{\sqrt{1 + (AB\cos(B(X^\star - L)))^2}} \\ \dot{Z}^\star = 0 \\ \dot{s}^\star = \|\mathbf{V}\| \end{cases} \tag{10.17}$$

The set of equations is solved using the initial conditions provided in Table 10.2, ensuring continuity with the entrance trajectory obtained by solving Equation 10.11. The differential equations are solved $X = [330, 930]$m: as the curves are to be placed 500ft apart, this range for the X coordinate value allows for two turns to be performed on each side of the centerline.

Table 10.2: Initial values of the central slalom segment

$X_{r,0}$	330m
$Y_{r,0}$	0m
$Z_{r,0}$	70m
$s_{r,0}$	330m

The last element to consider is the acceleration required throughout the course, which will be used to determine the desired attitude. The acceleration required at each instant of the manoeuvre can be identified by differentiating with respect to time Equation 10.17 and by adding the gravitational acceleration g (required by the helicopter to maintain level altitude in the Earth's gravitational field). The resulting equations are provided in Equation 10.18.

$$\begin{cases} \ddot{X}^\star = \dfrac{\|\mathbf{V}\|^2 A^2 B^3 \cos(B(X^\star - L))\sin(B(X^\star - L))}{(1 + (AB\cos(B(X^\star - L)))^2)^{5/2}} \\ \ddot{Y}^\star = -AB^2 \sin\left(B(X^\star - L)\right)\dot{X}^\star + AB\cos(B(X^\star - L))\ddot{X}^\star \\ \ddot{Z}^\star = -g \end{cases} \tag{10.18}$$

where $\dot{X}$ is defined in Equation 10.16.

Segment 3: manoeuvre exit

The final part of the manoeuvre is another 330m unaccelerated forward flight segment at the trim speed $\|\mathbf{V}\| = 33$m/s. The equations used to identify the coordinates in this segment are identical to the ones provided in Equation 10.11, but they are now solved with the initial values in Table 10.3

Table 10.3: Initial values of the final slalom segment

$X_{r,0}$	930m
$Y_{r,0}$	0m
$Z_{r,0}$	70m
$s_{r,0}$	$\hat{s}$

where $\hat{s}$ is the total trajectory arc length of the first and second segments of the manoeuvre.

Similarly to the velocity and position values, the total acceleration required is identical to the first segment. As this is an unaccelerated part of the manoeuvre, the helicopter will only have to compensate for the Earth's gravitational acceleration to maintain altitude; as such, the acceleration required in the final part of the manoeuvre is:

$$\{\mathbf{A}_T^\star\}_G = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{10.19}$$

Attitude throughout the manoeuvre

The attitude around the manoeuvre is identified following the procedure described in Section 10.1. Here, the procedure followed will be exemplified using the slalom manoeuvre as an example.

As discussed previously, the attitude is determined starting from the acceleration signals identified above. By assuming that the main rotor thrust of the helicopter is aligned to the aircraft's body z axis, the desired attitude is identified as the orientation that allows the thrust of the main rotor (i.e. the body z axis) to be aligned with the desired acceleration throughout the manoeuvre. In particular, as the main rotor thrust is directed upwards while the body z-axis is directed downwards, the body z axis will have to be aligned to the negative of the acceleration. Moreover, as attitude measures the relative orientation of the body reference frame to the NED reference frame, here all quantities will be expressed in the NED reference frame.

Let $\mathbf{z}_E$ be the NED frame z-axis and let $\{\mathbf{A}^\star\}_E$ be the required acceleration measured in the NED frame, as obtained from Equations 10.12, 10.18 and 10.19 - depending on the manoeuvre section the helicopter is found in. The initial attitude correction can be immediately identified by computing the quaternion:

$$\hat{\mathbf{q}}_{E2B}^\star = \begin{bmatrix} \cos(\alpha/2) \\ \sin(\alpha/2)\mathbf{n} \end{bmatrix} \tag{10.20}$$

where $\cos(\alpha/2)$ and $\sin(\alpha/2)$ are found by implementing Equations 10.6 and 10.7, and $\mathbf{n}$ is found with Equation 10.8. The attitude quaternion identified in this manner allows for the alignment of the required acceleration and the body z axis, allowing the helicopter to manoeuvre through the desired path.

With the thrust aligned, the heading of the helicopter is to be controlled by rotating the identified intermediate quaternion $\hat{\mathbf{q}}_{E2B}^\star$ around its z axis. This is obtained by imposing a rotation of angle ξ, as shown in Equation 10.10. The heading correction for the slalom manoeuvre was performed to have the nose of the helicopter follow the instantaneous velocity reference around the manoeuvre: here, the procedure followed to achieve this is explained.

To start, the orientation of the desired velocity is to be expressed in the identified intermediate reference frame (represented by the attitude quaternion $\hat{\mathbf{q}}_{E2B}^\star$) by means of a passive rotation:

$$\begin{bmatrix} 0 \\ \{\mathbf{V}^\star\}_B \end{bmatrix} = (\mathbf{q}_{G2E}\hat{\mathbf{q}}_{E2B}^\star)^{-1} \begin{bmatrix} 0 \\ \{\mathbf{V}^\star\}_G \end{bmatrix} (\mathbf{q}_{G2E}\hat{\mathbf{q}}_{E2B}^\star) \tag{10.21}$$

where $\mathbf{q}_{G2E}$ indicates the quaternion describing the relative orientation of the NED frame with respect to the Global reference frame. Having identified the velocity vector expressed in the intermediate reference frame,

the projection on its xy-plane is isolated by only considering the first two elements of $\{\mathbf{V}\}_B = \{[V_x, V_y, V_z]\}_B$. A graphical representation of this is given in Figure 10.3.

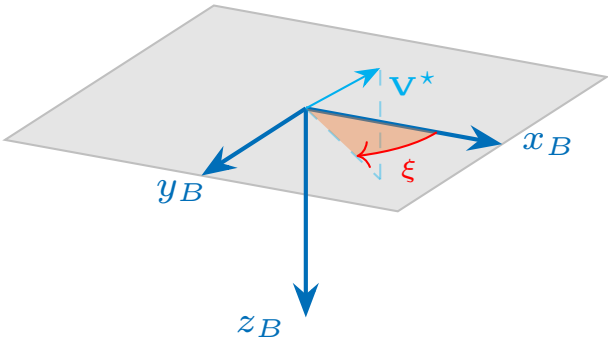


Figure 10.3: Representation of the sideslip correction, the horizontal plane defined by vectors x_B and y_B has been highlighted in gray

The identified velocity projection then allows for the identification of the orientation of the desired velocity with respect to the intermediate orientation of the helicopter. Let β_i be the intermediate sideslip angle, the angle can be identified by using the following equation:

$$\beta_i = \arctan\left(\frac{V_y}{V_x}\right) \tag{10.22}$$

where V_x and V_y are the x and y velocity components of $\{\mathbf{V}\}_B$. By then letting $\xi = \beta_i$, the correction quaternion $\mathbf{q}_\xi$ is identified using Equation 10.10.

Note that by calculating the desired heading offline additional flexibility is provided to the entire tracking algorithm, as this methodology allows to align the helicopter's body axis to any arbitrary reference during flight.

Complete manoeuvre

With the methodology described above, the slalom trajectory is fully defined, with appropriate position, velocity and attitude signals to be used in guiding the system. The results of these procedures are shown in Figures 10.4 to Figure 10.8 and will hereafter be briefly discussed to asses the validity of the calculations followed.

The position references are shown in Figure 10.5 and Figure 10.6. It is noticed that the manoeuvre lasts a total of 40.5s and covers a total horizontal displacement of 1260m. Variations in the lateral position coordinate are clearly divided in three segments: an initial and a final segment with no desired lateral displacement (i.e.: the initial and final forward flight segments) and a sinusoidal segment in the central part of the manoeuvre. The initial and final segments, as desired, both last 10s; the central manoeuvre segment lasts 20.5s. Figure 10.4 shows a 3D render of the trajectory path, with slalom gates placed 50ft from each side of the path's centerline to show the minimum required path deviation.

The velocity references around the manoeuvre are shown in Figure 10.7. As expected, the total velocity around the manoeuvre is kept constant at the trim speed of 33m/s, while the forward V_X and lateral V_Y velocity components in the global reference frame show moderate variations around their baseline values in the central manoeuvre section. In particular, the lateral velocity component shows large variations around zero, with peaks of ± 14.5m/s. These values are as expected and follow co-sinusoidal trends, complementary to the sinusoidal patterns of the trajectory: this suggests that the calculations were implemented correctly in the MATLAB coding environment and that the results are representative of the manoeuvre implemented.

The attitude to be followed around the trajectory is shown in Figure 10.8 and was found through the procedure described above to ensure the heading of the helicopter follows the trajectory. As expected, the

yaw ψ and roll ϕ angles are commanded to vary with sinusoidal patterns that match the trajectory and have amplitudes of respectively ± 25° and ± 27°, while the pitch angle θ exhibits only minimal variations. These results match the expectations, as the primary axis that will be excited during this manoeuvre are the lateral control axis, while the longitudinal effort required during this motion is minimal. These variations are also shown in the quaternion elements: the largest variations are found in the q_i and q_k elements (which roughly correspond to the ϕ and ψ Euler attitude angles), while q_j only displays minor shifts around 0. Variations in the q_0 term are only a consequence of the unity norm constraint on attitude quaternions, which was explained in Section 5.1.1.

Notice that the velocity and attitude values show discontinuities at the edges of the central section of the manoeuvre: this is to be expected from the procedure followed to generate the reference values, as they are obtained by connecting three independently-generated trajectory segments only considering continuities in the position channel and not accounting for continuity along the first and second derivatives of the position signals. When controlling a system, it is more desirable to have smooth references to drive the system, as smooth signals inevitably lead to smoother responses with less control effort placed on the actuators. This incongruence will be addressed in Section 11.1.2, where an online interpolation procedure for the reference signals is discussed to allow for online smoothening of the interpolation signals during simulated flight: the addition of this interpolation step allows for added flexibility in the trajectory generation phase and for a smoother and more robust integration with the autopilot system, as will be further expanded upon in Chapter 11.

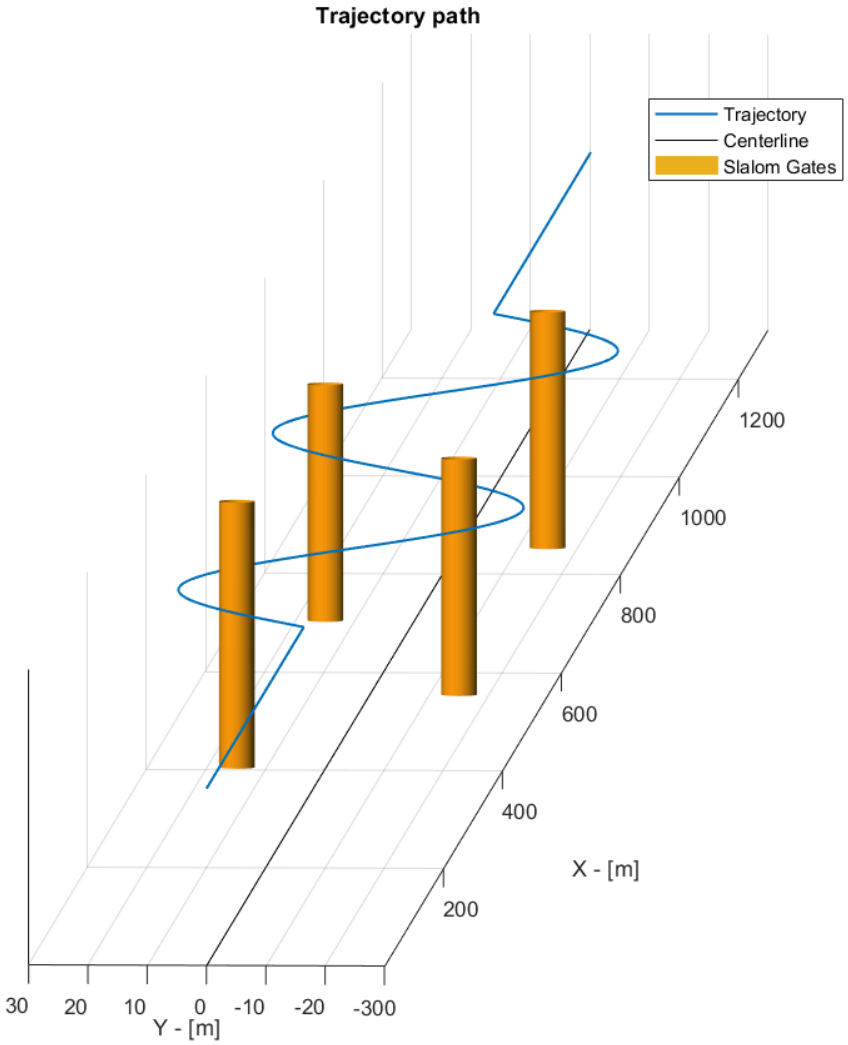


Figure 10.4: 3D view of the slalom trajectory with gates placed 50ft from the path's centerline

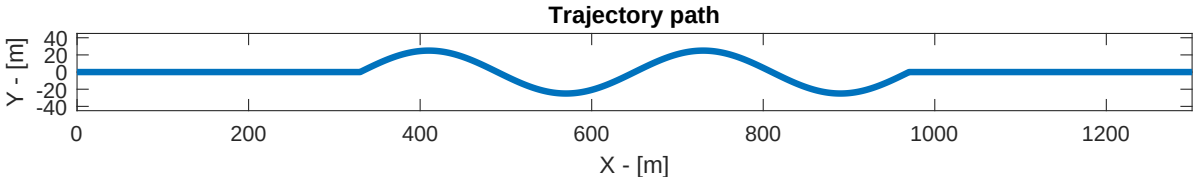


Figure 10.5: Vertical view of the slalom trajectory

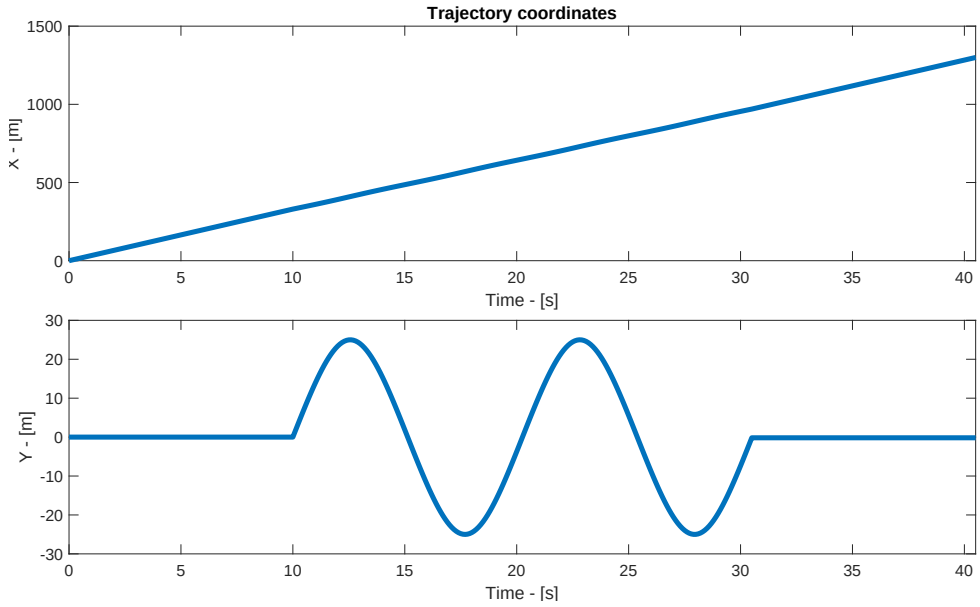


Figure 10.6: Slalom trajectory components

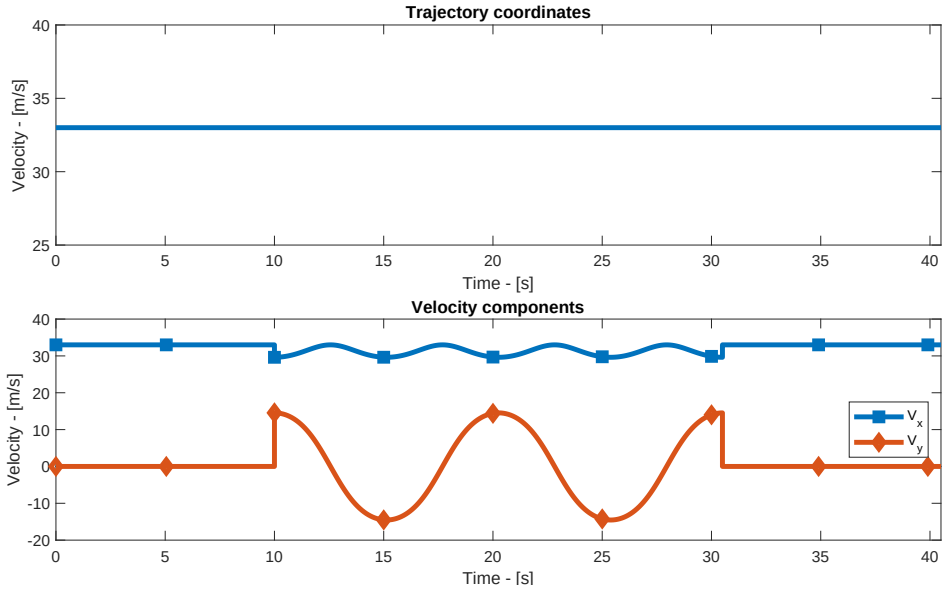


Figure 10.7: Slalom velocity

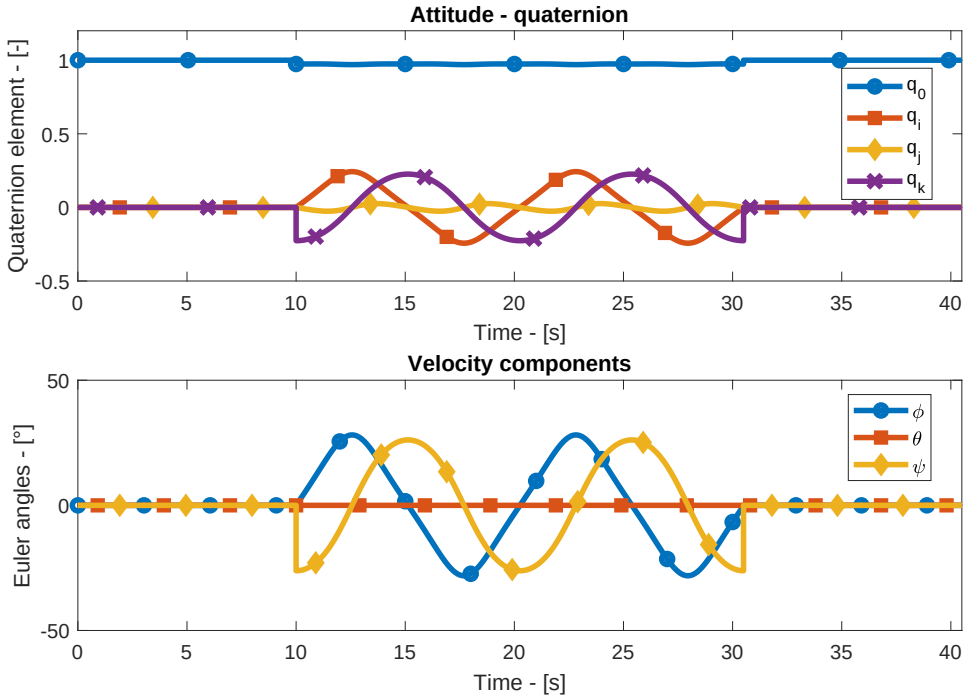


Figure 10.8: Slalom attitude

10.2.2. Pop-Up manoeuvre

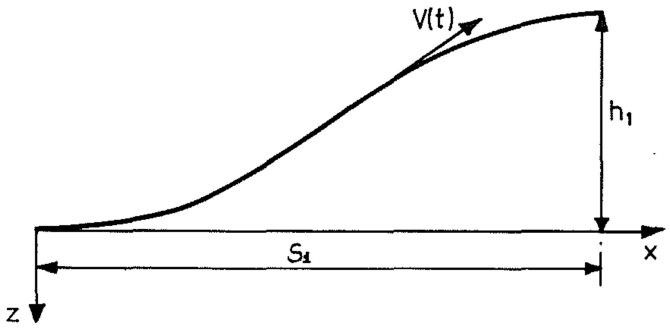
The second manoeuvre implemented for the purpose of helicopter performance evaluation is the pop-up manoeuvre, which is also discussed in a number of resources, including the ADS-33 [106] and the UTTAS [107]. Here, a definition of the pop-up manoeuvre and the methodology used to define an offline reference generation methodology for this specific manoeuvre case are provided.

The pop-up manoeuvre is a longitudinal manoeuvre consisting in a height change performed at a constant forward velocity, and it is used for obstacle clearance. This manoeuvre aims at qualifying the ability of the helicopter to obtain an altitude change sufficient to allow the vehicle to avoid possible obstacles that would otherwise impede its horizontal movement, while covering a limited horizontal distance to ensure a quick response.

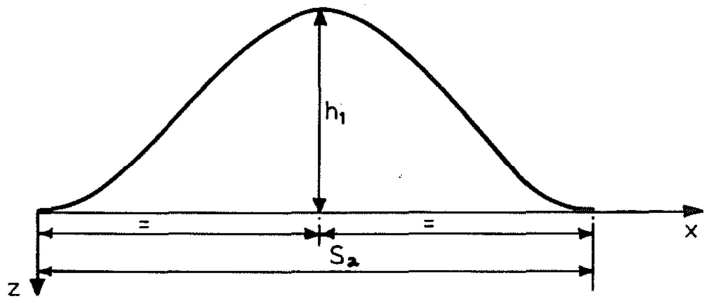
It is worth stating that the pop-up manoeuvre is often considered as a variation of the more common hurdle-hop manoeuvre [108]: the essential difference between the two manoeuvres is the fact that, while the hurdle-hop manoeuvre requires the helicopter to then return to its original altitude, the pop-up manoeuvre requires the helicopter to maintain the altitude obtained. In an effort to test the ability of the control system to perform altitude changes and to maintain the obtained altitudes, for the purpose of this thesis the pop-up manoeuvre was considered. A graphical comparison between the pop-up and the hurdle hop manoeuvre is provided in Figure 10.9.

Another commonly-used alternative to the pop-up manoeuvre when testing the longitudinal and vertical maneuverability of a helicopter is the pull-up/push-over manoeuvre, which aims at qualifying a vehicle's agility by having it track a specified load-factor profile consisting of a rapid climb to 2g and a quick descent to 0g flight [107]. While this manoeuvre was also considered as a possible candidate to test the controlled system discussed in this thesis, in the end the pop-up manoeuvre was deemed more adequate. This is because the control system architecture proposed in Part V mainly focuses on tracking position reference commands instead of load-factor profiles, thus making the pull-up/push-over manoeuvre (which is effectively defined as a load factor profile tracking MTE) less adequate for the system in question.

Similarly to the slalom manoeuvre, the pop-up manoeuvre is divided into three segments. To start, the helicopter will maintain forward flight for a duration of 10s holding the trim speed 33m$/$s at the set trim



(a) Pop-up example trajectory [108]



(b) Hurdle-hop example trajectory [108]

Figure 10.9: Comparison between the pop-up and hurdle-hop manoeuvre profiles

altitude h. Afterwards, the helicopter will perform the pop-up manoeuvre, gaining an elevation of h_1 over a horizontal distance s_1 while maintaining constant speed. Concluding the manoeuvre, the helicopter will settle at the altitude $h + h_1$, maintaining trim velocity.

In this manoeuvre, the trajectory requirements presented by Thomson in [108] will be used, as it provides a comprehensive overview of the desired manoeuvre characteristics. In [108], the pop-up manoeuvre is defined as a manoeuvre that should guarantee a 25m altitude gain over a horizontal distance comprised between $250\text{m} \leq s_1 \leq 350\text{m}$. In this work, to emphasise the vertical manoeuvrability of the helicopter, the helicopter will be required to gain $h_1 = 25\text{m}$ in altitude over a horizontal distance of $s_1 = 250\text{m}$.

Manoeuvre entrance and exit

As the initial and final segments of the manoeuvre are fundamentally identical to the ones of the slalom manoeuvre presented in Section 10.2.1, the set of differential equations used to identify the coordinates profiles is the same as the one presented in Equation 10.11, which are solved with the initial conditions presented in Table 10.4 for the manoeuvre entrance and Table 10.5 for the manoeuvre's exit. In the tables, the trim altitude for the helicopter was set to $h = 70\text{m}$ (consistent with the slalom manoeuvre), the initial X coordinate for the final segment was obtained by solving the equation:

$$X_{r,0} = \|\mathbf{V}^\star\| \Delta T_0 + s_1 \tag{10.23}$$

where $\|\mathbf{V}^\star\| = 33\text{m/s}$ is the trim velocity, $\Delta T = 10\text{s}$ is the duration of the entrance manoeuvre segment, and $s_1 = 250\text{m}$ is the horizontal distance to be covered in the pop-up manoeuvre segment. The trajectory arc length initial condition $\hat{s}$ is found by solving the pop-up manoeuvre segment.

Table 10.4: Initial values of the initial pop-up segment

$X_{r,0}$	0m
$Y_{r,0}$	0m
$Z_{r,0}$	70m
$s_{r,0}$	0m

Table 10.5: Initial values of the final pop-up segment

$X_{r,0}$	580m
$Y_{r,0}$	0m
$Z_{r,0}$	95m
$s_{r,0}$	$\hat{s}$

Pop-up manoeuvre segment
The central part of the manoeuvre requires the identification of an appropriate smooth trajectory connecting the initial and final manoeuvre segments while maintaining the trim velocity.

To provide an alternative approach to the one used in Section 10.2.1, here the trajectory will be generated using a smooth polynomial function. While with the approach used for the slalom manoeuvre the trajectory is generated by simply splicing together two separate and easy-to-model trajectory segments (obtaining a trajectory with local discontinuities that will be naturally smoothened by the autopilot interpolation, as will be discussed in Section 11.1), for the pop-up manoeuvre the trajectory reference will be generated using a more general trajectory definition approach which will naturally produce a smooth function by generating a connecting polynomial.

Letting X_0 be the initial condition for the X coordinate in the pop-up manoeuvre segment, the conditions the polynomial function needs to satisfy to produce a smooth transition from the initial altitude to the final altitude are the following:

$$\begin{aligned} Z^\star(X_0) &= h & \frac{dZ^\star(X_0)}{dX^\star} &= 0 & \frac{d^2Z^\star(X_0)}{dX^{\star 2}} &= 0 \\ Z^\star(X_0+s_1) &= h+h_1 & \frac{dZ^\star(X_0+s_1)}{dX^\star} &= 0 & \frac{d^2Z^\star(X_0+s_1)}{dX^{\star 2}} &= 0 \end{aligned} \tag{10.24}$$

As the polynomial trajectory needs to satisfy six conditions, the altitude can be defined as a function of the covered horizontal distance as a fifth-order polynomial $Z^\star(X)$ defined over the range $X^\star \in (X_0, X_0+s_1]$. The polynomial can then be defined as a function of six unknown parameters $\{a,b,c,d,e,f\}$: equations 10.25 to 10.27 show the shape of the polynomial equations for $Z^\star(X)$ and its relevant derivatives.

$$Z^\star(X) = a(X^\star-X_0)^5 + b(X^\star-X_0)^4 + c(X^\star-X_0)^3 + d(X^\star-X_0)^2 + e(X^\star-X_0) + f \tag{10.25}$$

$$\frac{dZ^\star(X)}{dX^\star} = 5a(X^\star-X_0)^4 + 4b(X^\star-X_0)^3 + 3c(X^\star-X_0)^2 + 2d(X^\star-X_0) + e \tag{10.26}$$

$$\frac{d^2Z^\star(X^\star)}{dX^{\star 2}} = 20a(X^\star-X_0)^3 + 12b(X^\star-X_0)^2 + 6c(X^\star-X_0) + 2d \tag{10.27}$$

By using the equations and conditions defined above, the six unknown polynomial parameters can be identified by solving a linear system of equations, completely determining the evolution of the altitude as a function of the horizontal position.

With the polynomial parameters determined, a system of differential equations is defined to completely identify the manoeuvre. To start, the total velocity at any point of the equation is determined by the following equation:

$$\|\mathbf{V}\| = \sqrt{\left(\dot{X}^\star\right)^2 + \left(\dot{Z}^\star\right)^2} \tag{10.28}$$

where the velocity component along the Y direction was omitted since the manoeuvre is entirely longitudinal. The vertical velocity component $\dot{Z}$ can be determined by using the chain rule for derivative functions, meaning that:

$$\dot{Z}^\star = \frac{dZ^\star}{dX^\star}\dot{X}^\star \tag{10.29}$$

With these considerations and recalling that the manoeuvre is carried out with a constant total velocity $\|\mathbf{V}^\star\|$, Equation 10.28 can be rewritten in Equation 10.30.

$$\|\mathbf{V}^\star\| = \sqrt{\dot{X^\star}^2 \left(1 + \frac{dZ^\star}{dX^\star}^2\right)} \tag{10.30}$$

Which can further be rearranged to provide an equation for the horizontal velocity $\dot{X}$.

$$\dot{X}^\star = \frac{\|\mathbf{V}^\star\|}{\sqrt{\left(1 + \frac{dZ^\star}{dX^\star}^2\right)}} \tag{10.31}$$

This also allows for an immediate identification of the vertical velocity by including Equation 10.31 in Equation 10.29.

The equations identified above allow for the definition of the following set of differential equations:

$$\begin{cases} \dot{X}^\star = \dfrac{\|\mathbf{V}^\star\|}{\sqrt{\left(1 + \frac{dZ^\star}{dX^\star}^2\right)}} \\ \dot{Y}^\star = 0 \\ \dot{Z}^\star = \dfrac{dZ^\star}{dX^\star}\dot{X}^\star \\ \dot{s}^\star = \|\mathbf{V}^\star\| \end{cases} \tag{10.32}$$

which can then be solved using the initial conditions shown in Table 10.6. In the table, the initial conditions for the desired horizontal coordinate $X^\star$ and arc length $s^\star$ indicate the distance covered in forward flight by the helicopter when travelling at the trim speed $33\text{m}/\text{s}$ for 10s.

Table 10.6: Initial values of the central pop-up segment

$X_{r,0}$	330m
$Y_{r,0}$	0m
$Z_{r,0}$	70m
$s_{r,0}$	330

Attitude throughout the manoeuvre

With the position and velocity signals identified, the only remaining references to be determined are the attitude signals to allow for the manoeuvre to be followed. The methodology and assumptions made have already been presented in Section 10.1 and a first applied example of them has also already been discussed in Section 10.2.1: here, to avoid any further repetition, only the most crucial point of the attitude reference determination process will be discussed.

As already discussed, the fundamental idea behind the attitude determination procedure is the identification of a required acceleration profile (compensating also for gravity) that allows to follow the identified trajectory path and then the correction of the helicopter's orientation to align the main rotor thrust to the desired acceleration vector. To allow for an effective solution of the equations with reduced computational cost and limited model knowledge, the fundamental assumption made in this case is that the helicopter's main rotor thrust is exactly aligned with the body z-axis of the vehicle, which was justified in Section 10.1.

The first fundamental step in the attitude identification procedure is the determination of the required attitude around the trajectory. As discussed, this will need to account for two components: an acceleration component required to follow the path itself $\mathbf{A}^\star$, and an additional acceleration component compensating

for gravity's influence, according to equation Equation 10.33 where $g = -9.81\text{m/s}^2$.

$$\{\mathbf{A}_T^\star\}_G = \{\mathbf{A}^\star\}_G - \left\{ \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right\}_G \tag{10.33}$$

The desired acceleration components used to allow for path following can be identified by differentiating the desired velocity components along the global reference frame's axes $[X, Y, Z]$. For the initial and final segments of the manoeuvre, the acceleration profile is trivially identified: since the manoeuvres are unaccelerated and in forward flight, the accelerations for the initial and final manoeuvre segments are shown in Equation 10.34.

$$\{\mathbf{A}_T^\star\}_G = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{10.34}$$

For the central part of the manoeuvre, the velocity equations have already been identified in Equation 10.32. With this, the resulting equations, already correcting for the gravitational acceleration component, are shown in Equation 10.35.

$$\{\mathbf{A}_T^\star\}_G : \begin{cases} \ddot{X}^\star = -\dfrac{\|\mathbf{V}^\star\|}{\sqrt{\left(1 + \dfrac{dZ^\star}{dX^\star}^2\right)^3}} \dfrac{dZ^\star}{dX^\star} \dfrac{d^2Z^\star}{dX^{\star 2}} \\ \ddot{Y}^\star = 0 \\ \ddot{Z}^\star = \dfrac{dZ^\star}{dX^\star}\ddot{X}^\star + \dfrac{d^2Z^\star}{dX^{\star 2}}\dot{X}^2 - g \end{cases} \tag{10.35}$$

By solving the equations above, the acceleration profile required during the manoeuvre is identified and used for the purpose of attitude determination. In this case, as the pop-up manoeuvre is defined as an entirely longitudinal manoeuvre, attitude is determined to allow the helicopter to perform the manoeuvre while keeping the helicopter's longitudinal axis aligned with the trajectory itself. This condition is equivalent to once again imposing a desired sideslip angle of $\beta = 0$. As the procedure followed to determine the attitude around the manoeuvre is formally identical to the one used for the slalom manoeuvre, for brevity and readability of the document, it will not be presented again.

Complete manoeuvre

With the reference position, velocity and attitude profiles determined throughout the manoeuvre, the pop-up MTE has been completely determined. In its entirety, the manoeuvre lasts 27.6s and covers a horizontal distance of 910m with an elevation change of 25m: these parameters are exactly aligned with the expected manoeuvre characteristics and suggest the correct nature of the obtained results. The total velocity is maintained constant throughout the entire manoeuvre: as expected, the gain in altitude is accompanied by a reduction of the horizontal velocity component V_X and a proportional increase in the vertical velocity element V_Z, with the former decreasing to approximately 32m/s while the latter increases to 6m/s.

The attitude throughout the maneuver is shown in Figure 10.13, which is divided into two to show both the desired attitude in quaternion elements and in Euler angles (for ease of interpretability). Since the manoeuvre is entirely in the longitudinal plane, the references are expected to show variations only in the pitch angle θ (which is roughly represented by the q_j quaternion element): by analysing the figure, it is noticed that the results obtained following the presented procedure match the expectations, with variations only on the pitch angle θ and with no variations on the roll and yaw angles ϕ and ψ. Predictably, variations in the quaternion element references also follow the expected behaviour, with the largest variations being expressed in the q_j quaternion component; the small variations found in q_0 are a simple response to the variations in q_j, as the attitude is represented via unit quaternions which is to maintain unitary norm.

The trend of the variation also matched the expected values, with an initial rise in the value of θ at the entrance of the manoeuvre followed by a symmetric reduction in the pitch angle magnitude to allow for

settling at the achieved altitude: this also matches the trend in the changes of the horizontal velocity V_X, which will first decrease in magnitude before increasing again to maintain a constant total velocity. The magnitude of the attitude change is also consistent with the predicted trajectory: as the manoeuvre primarily excites changes in the vertical velocity, only minor variations in attitude are expected, since changes in vertical velocity will not provoke large changes in the direction of the acceleration vector, but will primarily change its magnitude.

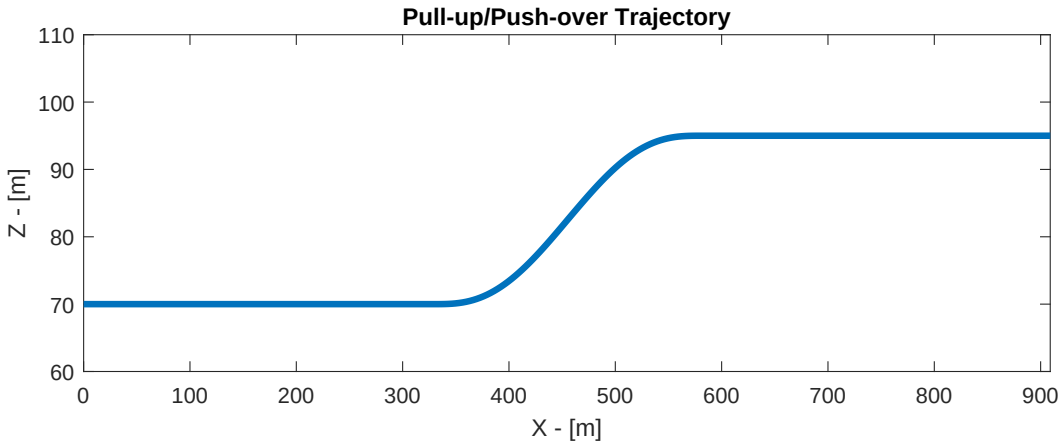


Figure 10.10: Pop-up trajectory

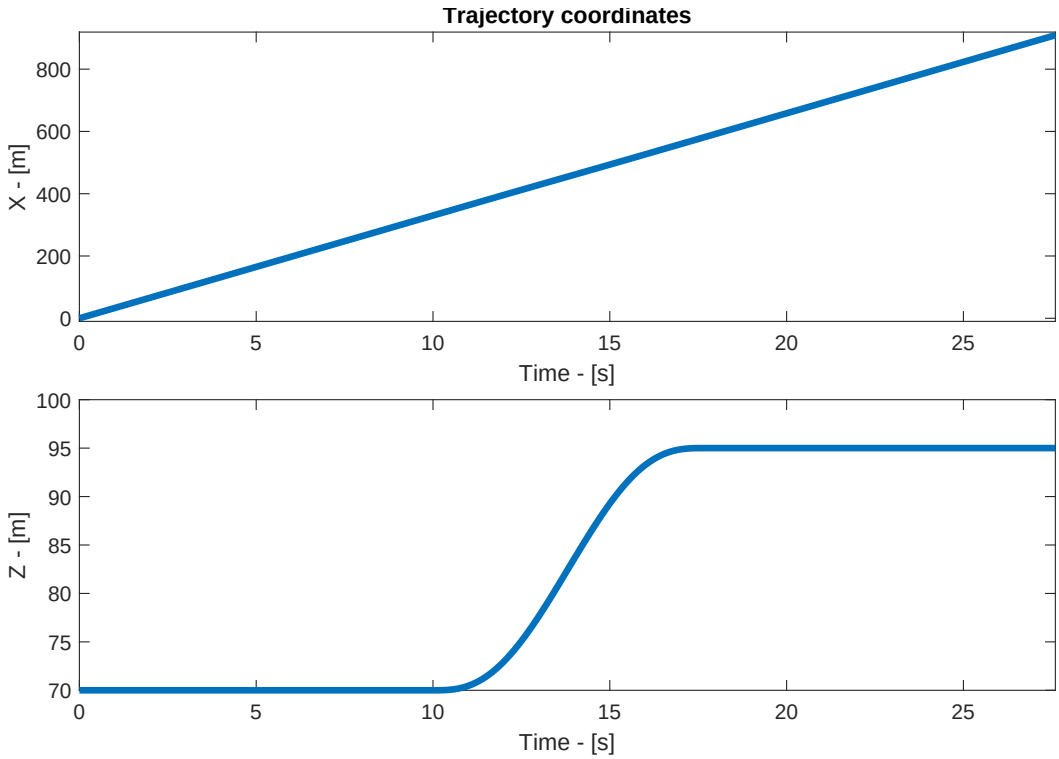


Figure 10.11: Pop-up trajectory components

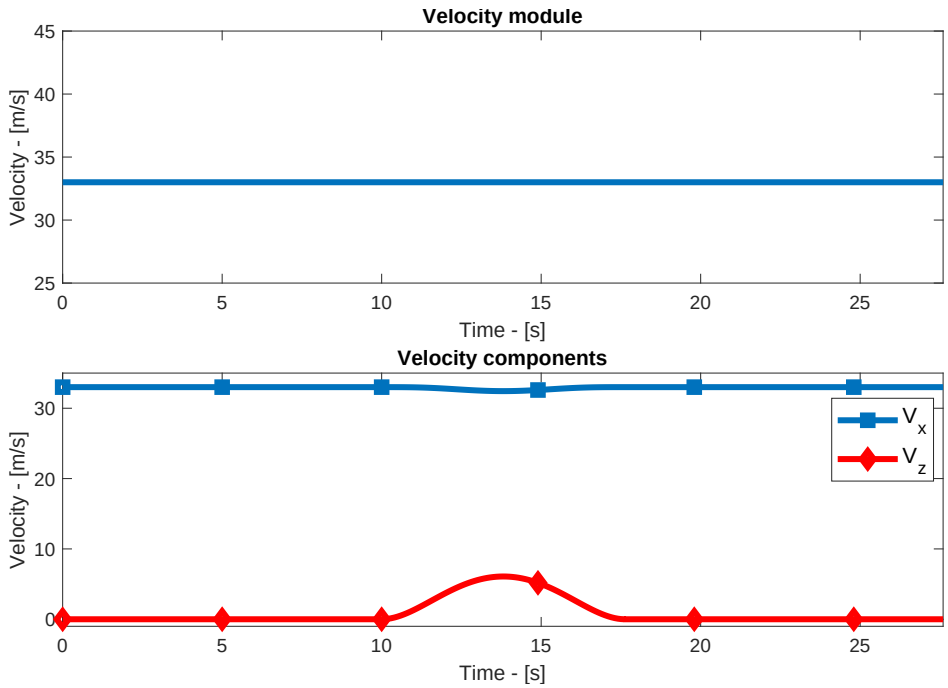


Figure 10.12: Pop-up velocity

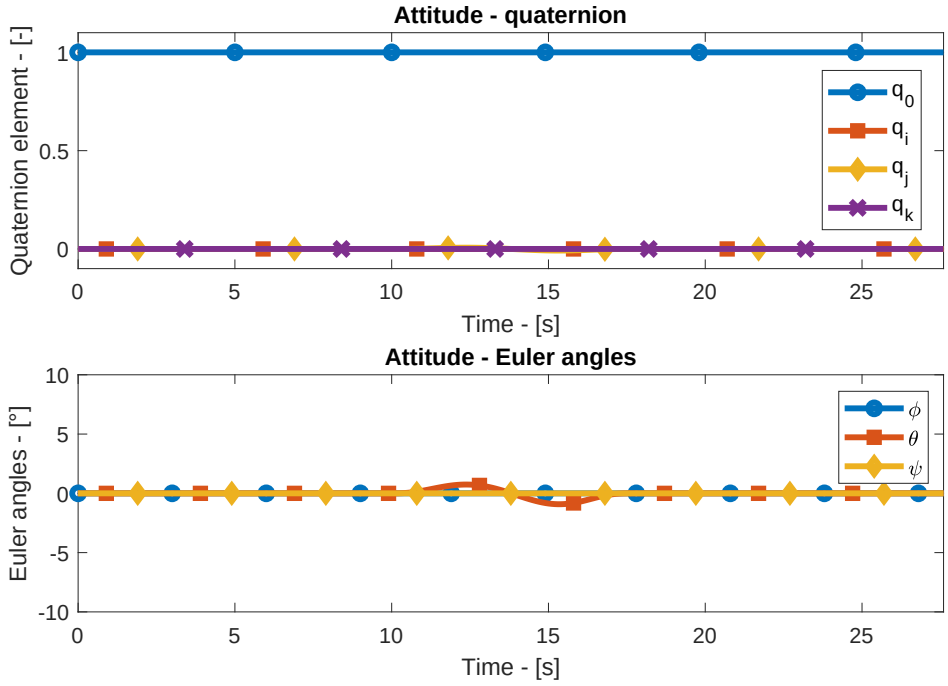


Figure 10.13: Pop-up attitude

11

Helicopter autopilot module

To properly evaluate the performance of the controller developed in Part V in the manoeuvres defined in Chapter 10, an additional structure has to be introduced to the system to appropriately feed the reference values to the system.

The fundamental motivation behind the implementation of such a system lies in the fact that controlling the motion of the helicopter through a previously computed trajectory would not account for possible mismatches between the system's assumed behaviour and the system's real response. By commanding motion with continuous-time signals, errors caused by the inability of the system to maintain the pace dictated by the reference signals (which may be due to non-minimum phase behaviours, response transients, or even external disturbances) would cause the instantaneous reference values to be inaccurate, causing large error build-ups and making the trajectory followed by the helicopter inaccurate.

To create a flexible structure that is robust to external disturbances, the system ought to be capable of generating reference signals on-line to appropriately follow defined reference trajectories: to achieve this reliably, the control system previously presented and tuned has been augmented with an autopilot module, which has the primary purpose of calculating the instantaneous desired signals the helicopter should follow to track a desired trajectory based on the current measured position of the helicopter.

The fundamental advantage of this approach is the decoupling of the trajectory planning and the controller design. By allowing the autopilot to provide references to the vehicle based on the helicopter's current position, the offline definition of trajectories and paths will not require to account for transient behaviours of the helicopter, significantly simplifying the design process. Further, by having the controller references be generated online, the system is now equipped to account for external disturbances that could cause unwanted deviations in the system's behaviour, improving tracking of precise paths.

Within this chapter, the development of such an autopilot module will be discussed. To provide an orderly discussion of the reference generation module, Section 11.1 will explain the procedure followed to encode the manoeuvres generated in Chapter 10 in the autopilot module to initialize the simulation and the interpolation method used to streamline the generation of reference signals. Afterwards, in Section 11.2 the baseline rationale behind the functioning of the autopilot will be presented, providing a global understanding of the functioning of the autopilot and expanding on its implementation in the Simulink environment.

11.1. Waypoint navigation and reference interpolation logic

To start the discussion of the autopilot, first the necessary manipulation of the reference data identified in Chapter 10 is discussed.

Initiating the discussion of the available data, in Section 11.1.1 the continuous-time signals describing the manoeuvres will be sampled in waypoints to increase tracking efficiency while limiting the data required by the system to perform manoeuvres in 3D space. Afterwards, in Section 11.1.2 the interpolation strategy used to reconstruct the reference signals online will be discussed, with particular attention placed on the reconstruction of the position and velocity desired values.

11.1.1. Identification of waypoints

In Chapter 10, a procedure for defining testing manoeuvres has been defined to generate a set of reference positions, velocities and attitudes starting from basic kinematics equations: by using this approach the reference values throughout the trajectory are defined as a continuum as functions of time. However, studies suggest that waypoint-based navigation provides for much better dynamic performances and flexibility [109, 110], as they reduce both computational burden and required data while also providing a more general structure that can be augmented with a number of very efficient algorithms for optimization. Another advantage of waypoint navigation lies in the implied generation of smoother trajectories, as interpolation enforces continuous transitions and derivatives between waypoints limiting abrupt transitions between trajectory segments, a limiting factor of the approach presented in Chapter 10.

To allow for smoother navigation and a more flexible controller structure, the autopilot module was not built to store continuum trajectories, but rather to store waypoints that could then be used to reconstruct the trajectory via interpolation of the signals.

For simplicity of implementation, waypoints were chosen to be equally spaced in time. With this, the definition of a correct time spacing ΔT between waypoints is crucial for the functioning of the control system: in fact, selecting a large ΔT leads to waypoints too distant from one another, causing losses in trajectory accuracy and making the controller ineffective for tracking precise manoeuvres; on the other hand, selecting a small ΔT produces waypoints that are very close to one another, making interpolation ineffective and forcing the helicopter to follow a tighter trajectory. To allow for some flexibility in the manoeuvre and avoid having strict references, the time between waypoints was set to be $\Delta T = 1$s.

The waypoints sampled in this manner are then stored in appropriate data structures. The data required by the autopilot to allow for navigation are: position $\mathbf{X}^{\star}_{wp}$, velocity $\mathbf{V}^{\star}_{wp}$, trajectory arc length $\mathbf{s}_{wp}$, and attitude $\mathbf{q}^{\star}_{E2B,wp}$. A graphical representation of this interpolation is provided in Figure 11.1, where the reference values are represented by the continuous line and the sampled waypoints are indicated by the red dots.

With the manoeuvres sampled following the procedure described above, the remaining step is defining how the reconstruction of the trajectory and velocity references will be performed based on the available waypoint data. This procedure will be further explained in Section 11.1.2.

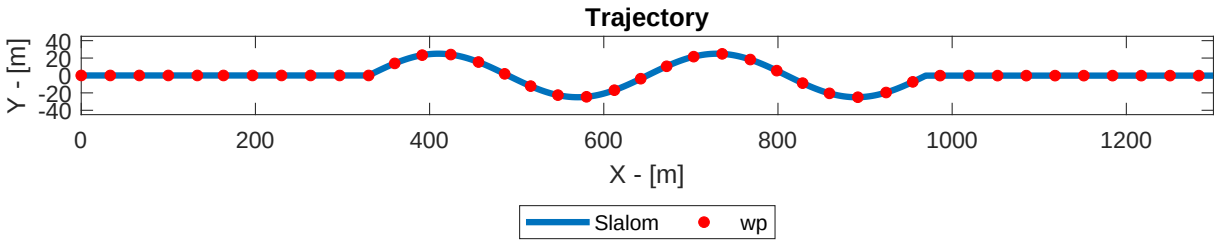


Figure 11.1: Example of waypoint selection over a trajectory

11.1.2. Definition of the interpolation strategy

With the waypoint identification methodology defined, the last remaining point of note is the identification of an appropriate interpolation logic to allow for on-line trajectory reconstruction.

As discussed, the data that need to be interpolated for the correct functioning of the autopilot are three: the helicopter's position coordinates $[X, Y, Z]$, the helicopter's velocity signals $[U, V, W]$, and lastly the helicopter's attitude $\mathbf{q}_{E2B}$.
By leveraging the properties of quaternions, a very efficient and effective algorithm for their interpolation is easily found in the SLERP algorithm, which has already been presented in Equation 5.25 and allows to map the shortest rotation in 3D space to drive the system from a defined initial attitude to a defined final attitude.
With an adequate strategy already identified for the interpolation of the attitude, the identification of an interpolation methodology for the position and velocity signals is discussed hereafter.

Regarding the choice of an interpolation strategy, there are three primary issues to account for:

- To start, the interpolation strategy chose needs to be adequate for the type of data that need to be interpolated, respecting any kind of internal constraint within the variables themselves. While this may not be an issue for the interpolation of the position and velocity variables, quaternion attitude functions differently as the quaternion representation requires the quaternion elements remain normalized throughout the flight.
- Moreover, the selected interpolation strategy should be capable of adapting to a generic trajectory, where in principle local variations of the reference values should not impact other regions of the signal. For this purpose, local interpolation methods are preferable over global interpolation methods: while the latter are more adept at generating smoother surfaces over large regions, they are also more susceptible to numerical instability [111]. While other methodologies have been proposed to impose local constraints on global interpolations for the specific purpose of trajectory generation (such as the "Global-to-Local" approach presented in [112]), local interpolation methods are generally preferred as they provide a simple and reliable approach to generate reference signals without losing accuracy [113].
- Lastly, the waypoint interpolation strategy selected should allow for a correct reconstruction of the waypoint with minimal computational cost: as this algorithm would need to function in real time as part of a larger system, it is crucial for the interpolation approach chosen to be reliable and responsive.

Considering the points of interest discussed above, the generation of the references between waypoints is handled with the use of a windowed cubic spline interpolation algorithm [114] expanding on the approach presented in [9]: this approach is described hereafter.

Consider the trajectory sampled at waypoints $k = 1 \dots N$, and in particular consider the helicopter to be found in the segment delimited by waypoints k and $k+1$. Typically, the interpolation polynomial segments are expressed using the formulation shown in Equation 11.1, where x_k indicates the independent variable's value at waypoint k (in this case, the arc length s_k) and $\hat{y}_k$ indicates the dependent variable's value at the same waypoint (in this case, this may indicate the desired position or the velocity).

$$\bar{y}_k(x) = d_k(x - x_k)^3 + c_k(x - x_k)^2 + b_k(x - x_k) + a_k \qquad \text{with} \quad x_k \leq x < x_{k+1} \tag{11.1}$$

In this specific case, to allow for the implementation of the interpolation system within the autopilot, the polynomial segments are all expressed as function of the functional l, a monotonously increasing running variable in the interval $[0, 1]$ such that when $l = 0$ the helicopter is at waypoint k and when $l = 1$ the helicopter is at waypoint $k+1$. By letting $h_k = (x_{k+1} - x_k)$, the relation between the functional l and the independent variable x is clearly identified by the equation:

$$(x - x_k) = l(x_{k+1} - x_k) = lh_k \tag{11.2}$$

By incorporating Equation 11.2 in Equation 11.1, the individual spline polynomial equations can be expressed as function of l, according to Equation 11.3, note that the notation $\bar{y}_k$ is used to express the polynomial coefficient as a function of the independent variable x, while y_k is used to express the same polynomial as a function of the functional l.

$$y_k(l) = d_k h_k^3 l^3 + c_k h_k^2 l^2 + b_k h_k l + a_k \quad \text{with} \quad 0 \leq l \leq 1 \tag{11.3}$$

The polynomial coefficients of Equation 11.3 are calculated online for each trajectory segment using waypoints $k-2$ to $k+3$. Note that at the extremities of the trajectory no sufficient waypoints would be available: as such, for $k \leq 2$ the waypoints used for the calculation of the coefficients are the waypoints $k = 1 \dots 6$; conversely, for $k \geq N - 2$, the waypoints used for the calculation of the coefficients are waypoints $k = N - 5 \dots N$.

The process described above produces five polynomial segments connecting each waypoint to the next. Of these, only one polynomial segment is selected at a time, corresponding to the current waypoint segment the helicopter is positioned in. To determine the coefficients of the polynomials, a system of equations can be set up to impose the required continuity on the various polynomial segments. As a total of five polynomials

have to be defined to properly generate the trajectory and since each is fully determined by four coefficients, a total of 20 coefficients need to be identified to consider the trajectory generation problem solved.

The first set of equations is obtained by ensuring each polynomial segment passes through its limiting waypoints. Let $\hat{y}_k$ be the value assumed at waypoint k by the variable being interpolated: as the running variable l is defined in the interval $[0, 1]$, this condition is expressed as:

$$\begin{cases} y_i(0) = \hat{y}_i & \forall i = k-2 : k+2 \\ y_i(1) = \hat{y}_{i+1} & \forall i = k-2 : k+2 \end{cases} \tag{11.4}$$

This set of constraints, as it is defined over all five polynomial segments, yields a total of 10 constraining equations.

The second set of constraint is found by imposing a first and second order continuity to the polynomial segments. Considering the polynomial segment $y_k(l)$ connecting waypoints k and $k+1$ defined according to Equation 11.3, its first and second derivatives are identified by Equations 11.5 and 11.6 respectively.

$$\bar{y}'_k(x) = \frac{d\bar{y}_k(x)}{dx} = 3d_k(x - x_k)^2 + 2c_k(x - x_k) + b_k \tag{11.5}$$

$$\bar{y}''_k(x) = \frac{d^2\bar{y}_k(x)}{dx^2} = 6d_k(x - x_k) + 2c_k \tag{11.6}$$

These equations can also be expressed as functions of the functional l to allow for implementation in the autopilot, leading to equations:

$$y'_k(l) = 3d_k h_k^2 l^2 + 2c_k h_k l + b_k \tag{11.7}$$

$$y''_k(l) = 6d_k h_k l + 2c_k \tag{11.8}$$

Once again considering the waypoints $k-2$ to $k+3$, this second set of constraints is expressed by the following equations:

$$\begin{cases} y'_i(1) = y'_{i+1}(0) & \forall i = k-2 : k+1 \\ y''_i(1) = y''_{i+1}(0) & \forall i = k-2 : k+1 \end{cases} \tag{11.9}$$

As these equations apply two constraints at each of the four internal nodes, Equation 11.9 yields a total of eight constraining equations.

Lastly, the third set of constraint is determined by imposing the Not-A-Knot condition on the second and second-to-last nodes $k-1$ and $k+2$: this condition imposes a third-order continuity at the identified nodes. By following the same procedure presented above, the third order derivative of the generic interpolating polynomial is determined as a function of l, obtaining the two equalities in Equation 11.10.

$$\begin{cases} d_{k-2}h_{k-2} = d_{k-1}h_{k-1} \\ d_{k+1}h_{k+1} = d_{k+2}h_{k+2} \end{cases} \tag{11.10}$$

By solving the system of equations determined by Equation 11.4, Equation 11.9 and Equation 11.10, a total of 20 equations are identified, allowing for the proper identification of the 20 required parameters. Note that as the equations are all linear in the parameters, the system of equations can be expressed in matrix form to find the unknown parameters $\{a_i, b_i, c_i, d_i\}$ with $k = \{k-2, \ldots, k+3\}$.

A graphical representation of the interpolation process is given in Figures 11.2, 11.3, where the interpolation algorithm described above has been applied respectively to the trajectory and to the velocity reference signals of the slalom manoeuvre presented in Section 10.2.1. It is immediately noticed that the interpolation is effective in generating an accurate representation of the original reference signal with minimal available information, as the reconstructed curve reference only deviates slightly from the original data: in particular, the trajectory interpolation is especially effective, showing only minimal variations at the points where the discontinuities produced by the simple trajectory generation procedure are located. The velocity interpolation is similarly effective in the trajectory segments where the reference is continuous and only shows noticeable variations from the desired signal at the points of discontinuity. As already

anticipated, the waypoint interpolation process allows for a natural smoothening of the feed-forward reference signals, with contained incongruences localized where discontinuities are present: while these incongruences will likely negatively impact the navigation accuracy, they will only have limited impact as they remain representative of the overall trends of the reference signals and any small discrepancy between the reference and feed-forward values will be compensated for by the nested-loop control system, significantly simplifying the trajectory generation process.

As an added advantage, the proposed interpolation algorithm relies entirely on linear operations, allowing the interpolation to be performed with only limited computational effort: this is especially important as the reduced number of operations will naturally limit the energy requirements of the autopilot module if implemented in a real system while also allowing for faster computation of reference data, enabling integration with faster sensors for more frequent measurements.

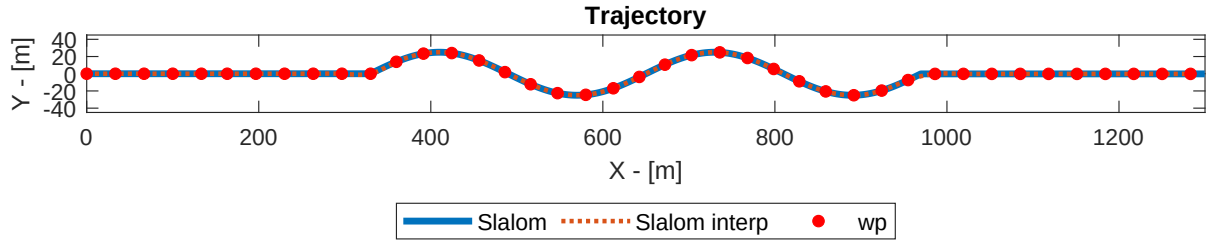


Figure 11.2: Example of the interpolation process applied to the slalom trajectory

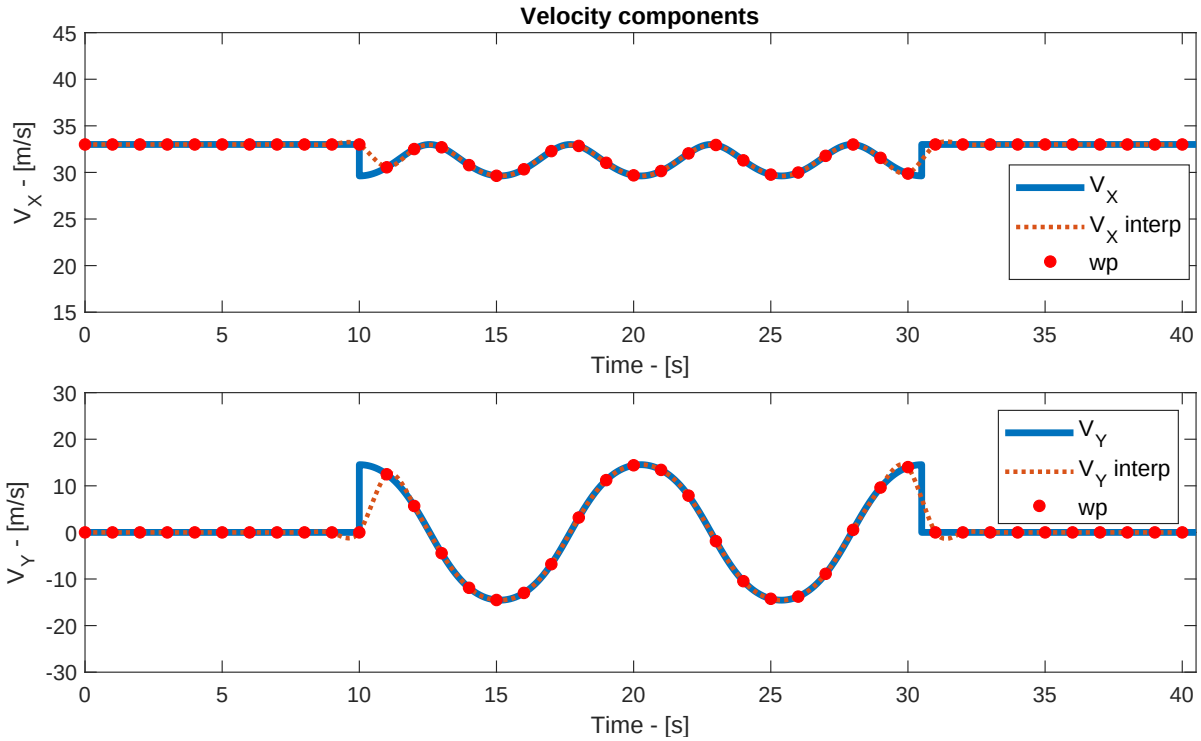


Figure 11.3: Example of the interpolation process applied to the slalom velocity references

11.2. Implementation of the autopilot module

With the references sampled via waypoints and an interpolation strategy defined to reconstruct the complete trajectory online, here the definition of the autopilot module will be discussed.
The purpose of the autopilot is the online generation of an appropriate reference signal based on the system's current position, allowing the system to navigate along a previously-defined trajectory: by

allocating this task to a separate block in the system navigation is greatly simplified, as users would only need to define waypoints in 3D space and the autopilot block would then autonomously generate a path connecting them and have the helicopter fly following the identified trajectory.

In an effort to produce a simple and reliable autopilot architecture that could also be modified by including more sophisticated trajectory generation techniques [115, 116], the module developed for the purpose of this thesis serves as an understandable and effective architecture that can produce accurate references in real time for the purpose of generating appropriate helicopter motions.

The structure used for the autopilot is that of the Finite State Machine (FSM) [9]. An FSM (also called a "Finite Automata") is an abstract computational model defined by a finite set of states, a finite set of input symbols, a state transition function that maps state–input pairs to subsequent states, an initial state, and (in some cases) a set of terminal or accepting states [49]. In the context of autonomous vehicle control, the FSM provides a formal framework to represent discrete modes of operation and the conditions under which the system transitions from one mode to another: this abstraction is particularly valuable in complex control applications, as it enforces a clear, modular separation between various high-level behavioural modes and simplifies both design and verification processes.

The modular nature of the FSM architecture makes this autopilot formulation particularly desirable for the formulation of a simple autopilot, as it provides a basic framework for the functioning of the system that can easily be augmented with additional modes or behavioural sets: this type of complex finite automata is referred to as a Hierarchical Finite State Machine (HFSM). Note that while this structure finds wider usage in self-driving vehicles because of its ability to handle more different behavioural conditions, it is often criticized for its limited reusability and often complex architecture [49]. For the purpose of this thesis a simple FSM has been developed, providing a functional baseline structure that can further be augmented to incorporate more complex high-level behaviours.

A graphical representation of the autopilot's FSM architecture is provided in Figure 11.4, where the black lines indicate the actual autopilot module and the grey lines indicate the behaviour of the controlled system. The autopilot is initiated at the starting waypoint $k = 1$ and has available all the waypoint positions and the respective desired velocities and attitudes, as discussed in Section 11.1.1. At each navigation step, the autopilot receives from the helicopter model the current position of the vehicle and uses it to identify the waypoint segment $\{k, k+1\}$ it is currently located in by calculating the running variable $l \in [0, 1)$.

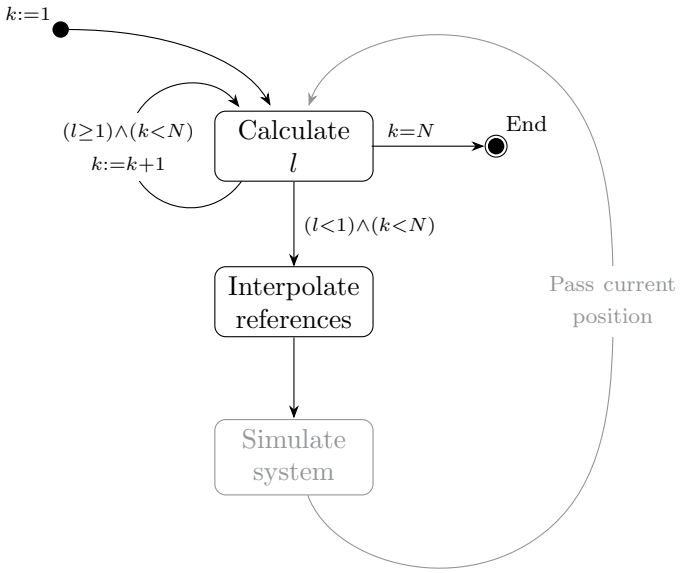


Figure 11.4: Autopilot represented as a Finite State Machine

To explain the functioning of the autopilot, consider the generic case represented in Figure 11.5, which represents the motion of the helicopter through a sample curved trajectory. In the figure, the helicopter's body B (represented by the red dot) is moving along a trajectory where the waypoints k, $k+1$, and $k+2$

have been highlighted, and in particular it is currently located in the trajectory segment $\{k, k+1\}$. The vectors $\mathbf{r}_{E,k}$ and $\mathbf{r}_{E,k+1}$ represent the positions of waypoints k and $k+1$ in 3D space with respect to the NED reference system (represented in the bottom-left of the figure). The position of the helicopter is provided by vector $\mathbf{r}_{E,B}$.

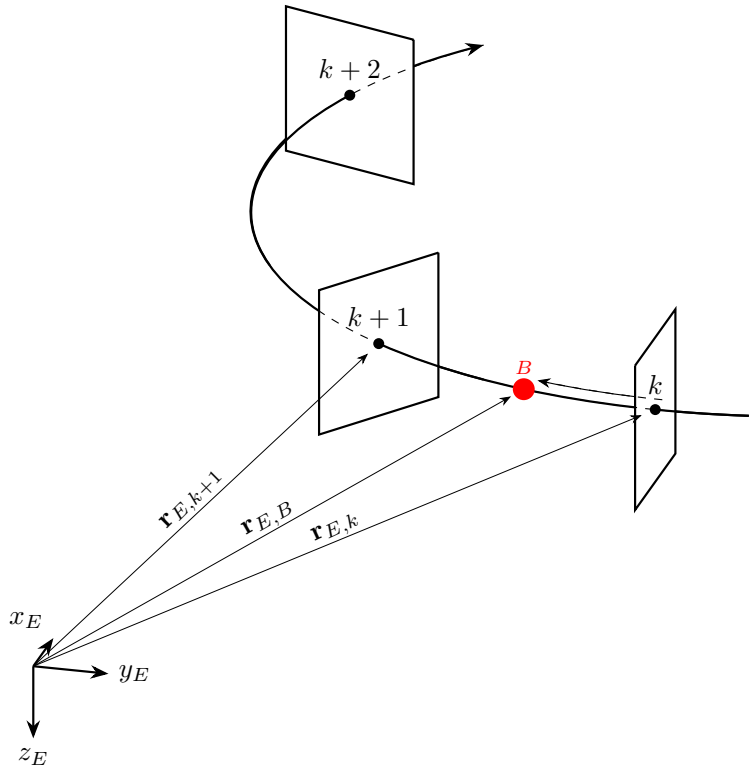


Figure 11.5: Representation of the helicopter's motion through a sample trajectory, as interpreted by the autopilot

To calculate the running variable l, the autopilot first assumes the helicopter is located in the waypoint segment $\{k, k+1\}$ (identified by the previous step of the simulation) and then uses this information to approximate the completion of the trajectory segment between the waypoints k and $k+1$ by calculating l according to Equation 11.11.

$$l = \frac{\mathbf{r}_{k,k+1} \cdot \mathbf{r}_{k,B}}{\|\mathbf{r}_{k,k+1}\|^2} + 0.05 \tag{11.11}$$

where $\mathbf{r}_{k,k+1}$ indicates the relative position vector of waypoint $k+1$ with respect to waypoint k and $\mathbf{r}_{k,B}$ indicates the relative position of the helicopter's body B with respect to waypoint k. The value of l is also increased by a value corresponding to 5%: this enables the autopilot to place the reference slightly in front of the instantaneous position of the vehicle, allowing for the system to move forward at a more regular pace in the trajectory. As shown in Figures 11.5 and 11.6, the relative position of the vectors can be immediately identified using the equations:

$$\mathbf{r}_{k,k+1} = \mathbf{r}_{E,k+1} - \mathbf{r}_{E,k} \tag{11.12}$$

$$\mathbf{r}_{k,B} = \mathbf{r}_{E,B} - \mathbf{r}_{E,k} \tag{11.13}$$

Defined in this manner, l provides an estimate of the helicopter's position within the trajectory segment: as l approaches unity, the helicopter's body B will move away from waypoint k and towards $k+1$. With an initial calculation of l, the FSM checks if the running variable is still within the desired range:

- if $0 \leq l < 1$, the running variable is considered valid and the helicopter's position is still found within waypoints k and $k+1$.
- if $l \geq 1$, the helicopter's position is found within the trajectory segment limited by $\{k+1, k+2\}$: the FSM then increases the waypoint counter $k = k+1$ and re-calculates l in the new waypoint range.

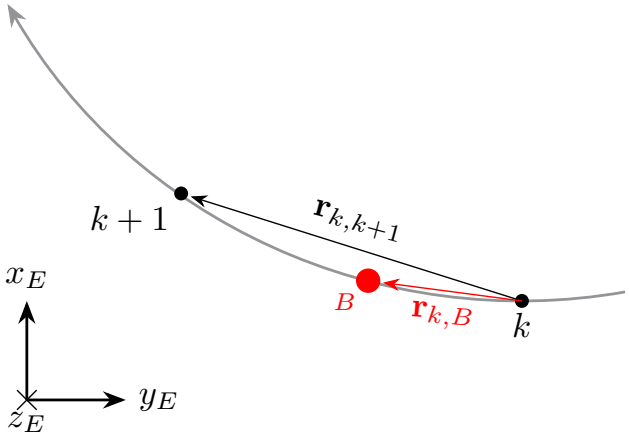


Figure 11.6: Top view of the helicopter's motion through a sample trajectory

The running variable defined here is then used by the autopilot to calculate the instantaneous reference values according to the interpolation strategy discussed in Section 11.1.2: for the position and velocity data, the windowed cubic spline approach is used; for the attitude references, the SLERP algorithm is used.
As during the identification of l the FSM architecture also determines the trajectory segment $\{k, k+1\}$ the helicopter is currently traversing, before performing the interpolation, the necessary waypoints required by the algorithm are isolated. As the SLERP interpolation only requires two values, the FSM only needs to isolate the previous and next waypoint, k and $k+1$ respectively. On the other hand, the interpolation of the velocity and the position data for the purpose of reference generation requires a total of six waypoints. Consider the helicopter to be in the segment limited by waypoints k and $k+1$ along a trajectory with N total waypoints:

- if $k \leq 2$, the waypoints used to generate the trajectory are the ones in the range $\{1, \ldots, 6\}$.
- if $2 < k < N-2$, the algorithm has sufficient waypoints both before and after the current trajectory segment, and as such the waypoints used are contained in the range $\{k-2, \ldots, k+3\}$.
- if $k \geq N-2$, the waypoints used for the interpolation are delimited by $\{N-5, \ldots, N\}$.

The operations described above are carried out for all steps of the manoeuvre simulation, which is considered completed when the helicopter's position has reached the final waypoint $k+1 = N \wedge l = 1$.

With this, the helicopter's autonomous guidance has been thoroughly described, proposing a simple and reliable structure that is capable of performing the task of generating the instantaneous references required by the helicopter to function. To achieve this, the FSM framework has been first introduced and motivated, highlighting its use in modelling the behaviour of self driving vehicles, and its advantages and limitations have been discussed. Afterwards, the specific implementation of the autopilot has been discussed, first providing an overview of the behaviour of the model and then highlighting the algorithms applied in each step of the simulation.

12

Manoeuvre simulation

To conclude the evaluation of the helicopter's autonomous flight capabilities and to discuss the complete system's ability to follow offline-calculated reference signals, relevant tracking tasks need to be simulated to test the system's behaviour during flight. To this end, this chapter investigates the dynamic performance of the flight control system and onboard autopilot by simulating two representative manoeuvers: a lateral slalom and a longitudinal pop-up. The reference signals that will be used for the simulation of these two manoeuvres, alongside the methodology used to obtain them, have been detailed in Chapter 10.

The chapter is divided in three sections. Section 12.1 discusses the simulation of the slalom manoeuvre: this MTE will then be discussed by analysing the system's response and the controls used, evaluating how closely the references are tracked and quantifying the control effort used in the MTE. Section 12.2 covers the pop-up manoeuvre, offering similar analyses to the one performed in the previous section for the slalom MTE. To conclude the chapter, in Section 12.3 final considerations regarding the system's tracking performance are presented, synthesising the findings, highlighting strengths and identifying potential faults of the system.

To evaluate the system's performance, the simulations for each manoeuvres were carried out using MATLAB's Simulink tool, a block-diagram environment adequate for desktop simulation testing of dynamic systems. All tests were performed using a fixed integration time step of 1ms with a Fourth-Oder Runge-Kutta (RK4) solver. The selected time step allows for an appropriate simulation of system's dynamics, granting a good resolution and being within the update rates for aircraft controllers [117, 118] even in the context of novel autonomous flight applications. The selection of the RK4 solver also serves as an adequate balance of computational efficiency and model accuracy, enabling the system to function with contained computational effort.

By discussing these simulations, the capability of the quaternion-based architecture and the three-loop controller to achieve precise reference tracking in both the horizontal and vertical axes is critically assessed. The results illustrate the interplay between the attitude, velocity, and position loops and quantify the improvements in tracking accuracy as well as the trajectory smoothness enabled by the autopilot's real-time interpolation capabilities.

12.1. Slalom MTE simulation

Here the simulation of the slalom MTE is discussed. A complete description of the manoeuvre has already been provided in Section 10.2.1, where the reference signals have also been identified.

As was explained in Section 10.2.1, the slalom MTE is a manoeuvre with constant total velocity that entails a quick succession of left and right turns at constant altitude, requiring simultaneous changes in both the roll angle ϕ and the yaw angle ψ to execute the trajectory tracking task correctly. The manoeuvre was implemented following the [106] specifications, which states that the turns have to be 500ft apart and have a deviation from the track's centerline of at least 50ft. With the required attitude and velocity changes along two of the body axes, the slalom is an effective lateral manoeuvre that can test the manoeuvrability of the helicopter.

The results of the simulations are shown in Figures 12.1 to 12.5, where all the most relevant simulation states are shown. Here a discussion of the results will be provided, highlighting their most relevant characteristics.

In Figure 12.1 the evolution of the position coordinates over the trajectory is shown. Within the figure, the evolution of the lateral position Y and the vertical position Z are shown over the horizontal coordinate X, allowing for an immediate interpretation of the evolution of the trajectory. Overall, the tracking performance of the system show promising results, with a generally good matching of the reference signals in both the longitudinal and lateral planes.

The figure shows that the controller is effective in following the desired altitude, with variations of at most ± 7m in the central segments of the manoeuvre. Slightly worse tracking performance are identified in the lateral direction, with the helicopter's trajectory lagging behind the desired response during the first two turns: this behaviour is consistent with the intermediate controller tuning results for the velocity and the position loops, discussed in Section 9.3 and Section 9.4 respectively, where it was shown that the controller had non-minimum phase behaviours in the lateral response especially, causing lagging. Even with these delays in the response, the helicopter is still capable of effectively tracking the position reference, always remaining within ± 5m of the reference path.

Noticeably, the position tracking is also much closer to the reference signal in the section between the third and the fourth slalom gates: this improvement in performance can be attributed to the introduction of the simple autopilot discussed in Chapter 11, as it allows for a dynamic generation of the reference signal based not on the amount of time passed (as would happen when performing simulations with continuous reference signals) but on the instantaneous position of the aircraft.

Continuing the discussion of the simulation results, Figure 12.2 shows the evolution of the velocity reference tracking during the manoeuvre. In the figure, the velocity references are shown in the instantaneous body reference frame of the vehicle and are generated via the interaction of the feed-forward term calculated in the offline definition of the manoeuvre with the velocity correction term computed online by the controller, as was explained in Section 8.3.

The velocity controller performance shows evident delays in the system's response, with the sway v and heave w components being especially slower than their respective references, leading to poor matching of the response. This behaviour also matches the criticalities discussed when analysing the system's tuning results, as similar delays were also noticeable even when using simple step reference signals. A possible way of addressing this would be the introduction of a derivative term in the velocity control loop, as it would make the system more responsive to variations in the reference: this, however, was avoided, as derivative controller actions are heavily susceptible to noise and signal errors, and as such would not be advisable when developing controllers for real systems.

The attitude around the manoeuvre is discussed in Figures 12.3 and 12.4, which respectively show the attitude evolution around the manoeuvre using the quaternion parametrization and the Euler angles parametrization. As expected, the highest variations in attitude are identified to be in the roll angle ϕ and in the yaw angle ψ, with the pitch angle θ having variations of only up to $10°$, matching the surge velocity reference.

The system's response appears satisfactory in both figures, with good simultaneous tracking on all axes and a smooth evolution of the references. This suggests the correct implementation of the LQI controller and the effective conversion to the quaternion formulation, as the attitude of the vehicle is responsive and is coherent with the trajectory and velocity signals.

Concluding the discussion of the slalom manoeuvre simulation results, Figure 12.5 shows the control actuation of the helicopter around the manoeuvre. It is immediately noticed from the figure that all control signals always remain within the allowed margins, never reaching complete saturation; this suggests that the optimization-based tuning approach used in Chapter 9 was effective in producing controllers capable of limiting control usage while maintaining good tracking performance. The control actuation also matches the expected behaviour of the controller, with simultaneous pedal and lateral cyclic actuations to regulate the yaw angle of the controller while navigating the curved section of the MTE, and longitudinal cyclic to

modulate the total velocity around the track. Noticeably, the collective has very large actuations, which make it almost reach saturation: these actuations match the variations in altitude measured in Figure 12.1 and are consistent with the cross-couplings identified when tuning the controller loops.

As an additional point of note, observing the evolution over the trajectory of the attitude and velocity references, it may be noticed that the reference signals at times show spiking and irregular behaviours, which appear inconsistent with the relatively smooth evolution of the position references. These small spikes in reference values, from a closer inspection, appear to match trends in the autopilot's running variable l and waypoint counter k, which are shown in Figure 12.6. As was explained in Section 11.2, these two running variables are used by the autopilot module to identify the position of the helicopter around the trajectory and calculate a reference via online waypoint interpolation.

As it may be noticed in the figure, the running variable l does not have a smooth evolution: this may be caused by the method used to calculate the value l, which, in curved trajectory segments, may cause the system to rapidly switch between adjacent waypoints. This is also consistent with the fact that these spikes are only present in the central section of the manoeuvre and are not present in the initial and final forward flight segments of the MTE. From the figure, the waypoint counter k shows consistent and regular increases, corroborating the correct implementation of the autopilot and its effectiveness in navigating the MTE selected.

Overall, from the results of the simulation, the controller and autopilot developed within this thesis are effective in following the slalom trajectory synthesized using the methodology provided in Chapter 10. The tracking performance of the autonomous system appear satisfactory, with the position reference being tracked with a maximum error of 7m along the vertical axis and of 5m in the longitudinal direction. The attitude of the system is also matched effectively, with attitude changes being effectively replicated along all three Euler angles with no control saturation. The results of the interpolation procedure also suggest that the trajectory interpolation is effective in producing a smooth reference for the system, allowing the aircraft to seamlessly navigate a trajectory that may have been identified offline with discontinuities in the velocity or attitude references.

In summary, the simulation demonstrates that the controller and autopilot achieve accurate tracking of the synthesized slalom path, verifying the tuning methodology, the autopilot implementation, and the interpolation strategy. The tight adherence to the reference trajectory and smooth attitude transitions highlight the system's readiness for agile flight tasks. A visual representation of the system's tracking performance in the slalom path is provided in Figure 12.7, which shows a 3D visualization of the slalom course, providing an immediate understanding of the system's performance during simulated flight.

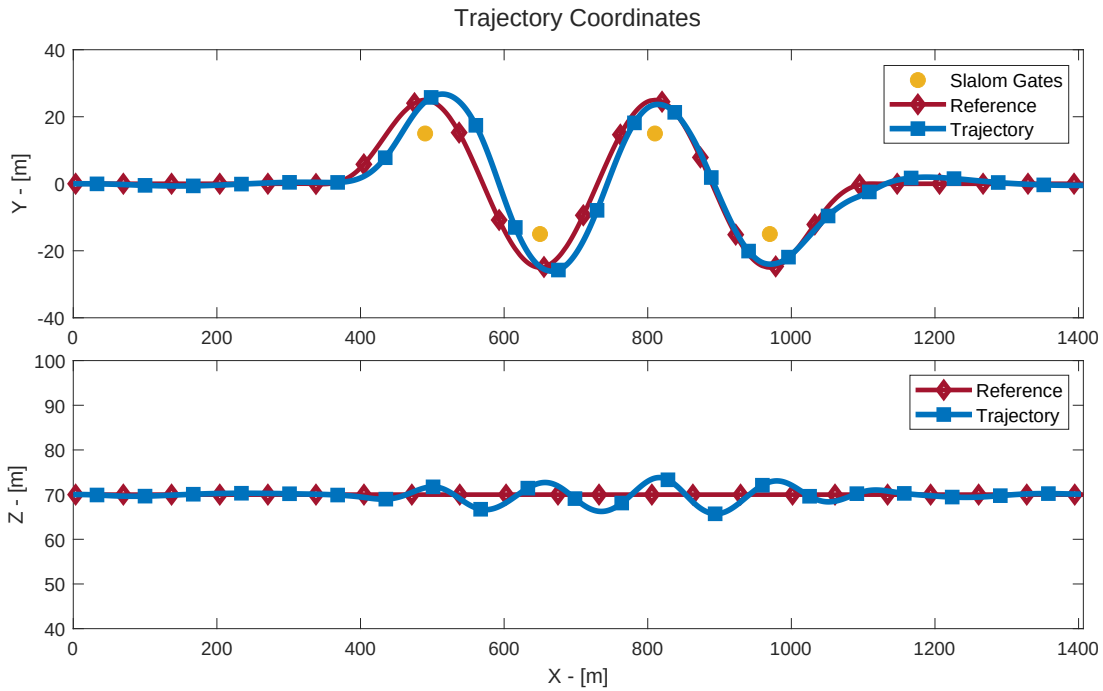


Figure 12.1: Autonomous flight tracking of slalom trajectory reference

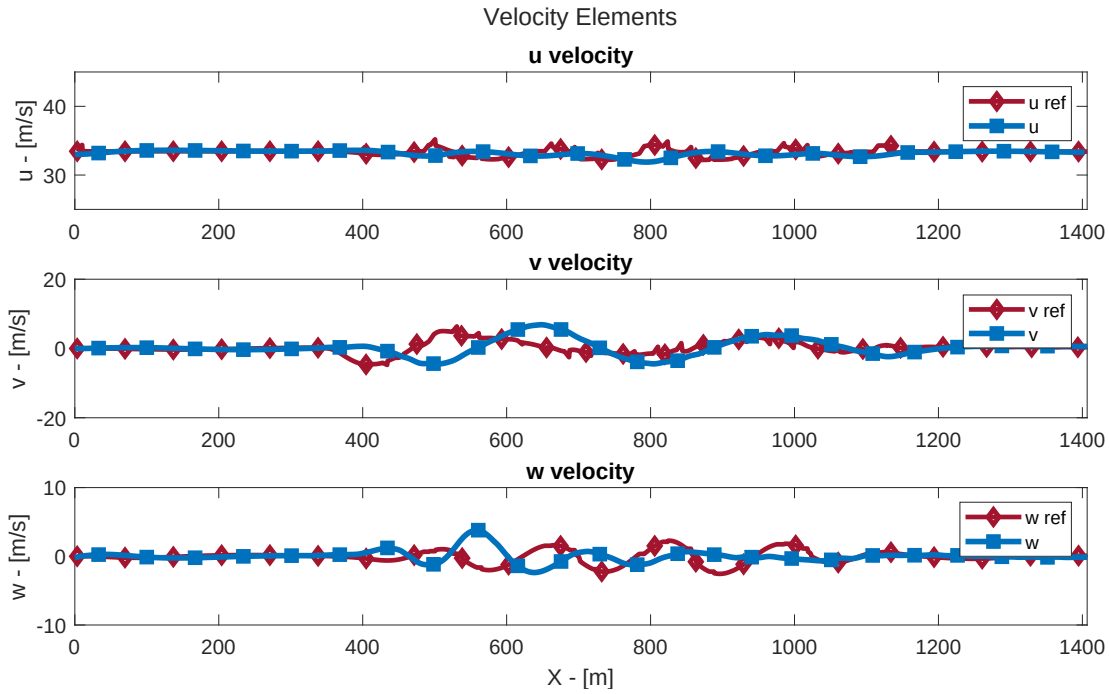


Figure 12.2: Velocity references during the slalom MTE

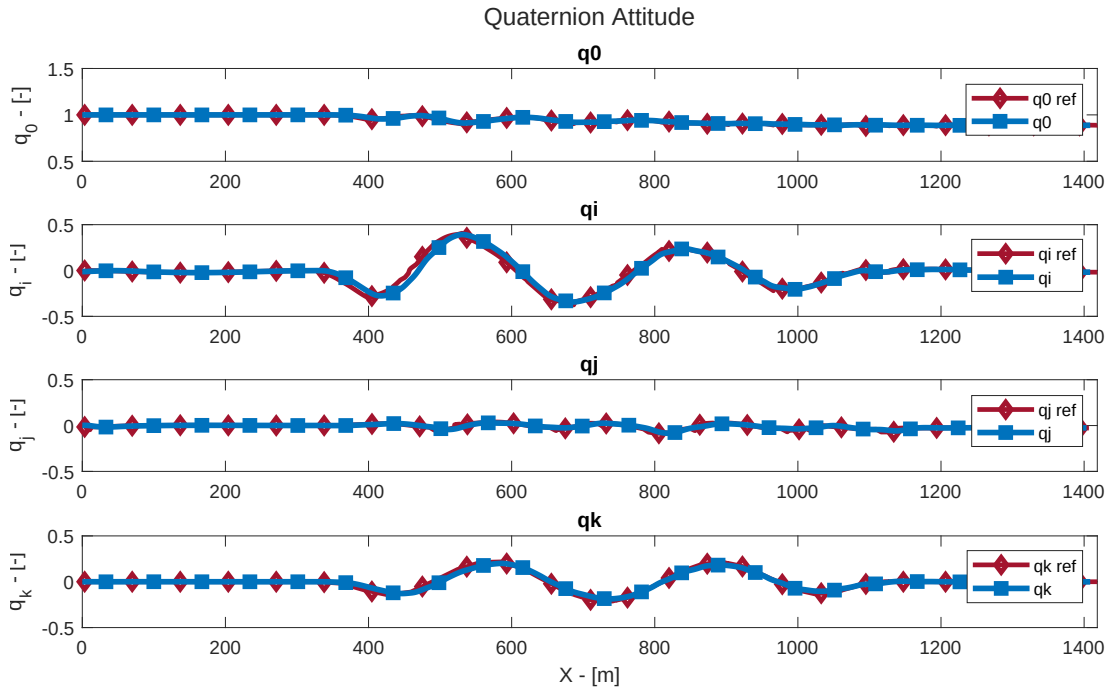


Figure 12.3: Quaternion attitude references during the slalom MTE

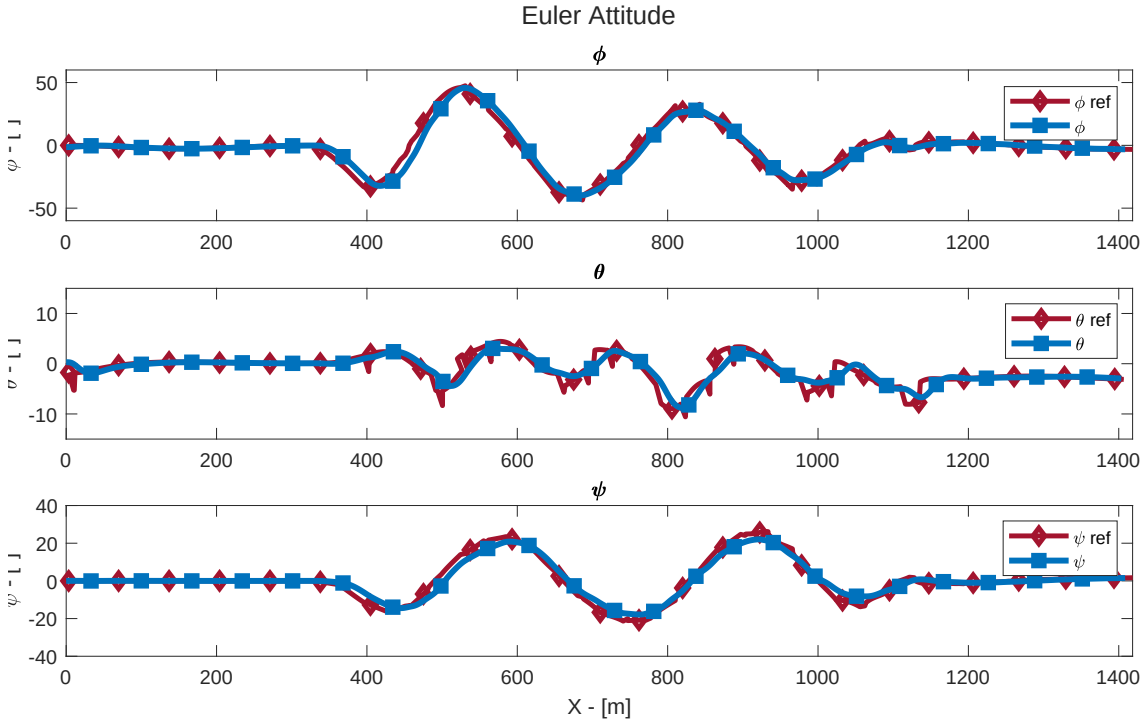


Figure 12.4: Euler angles attitude references during the slalom MTE

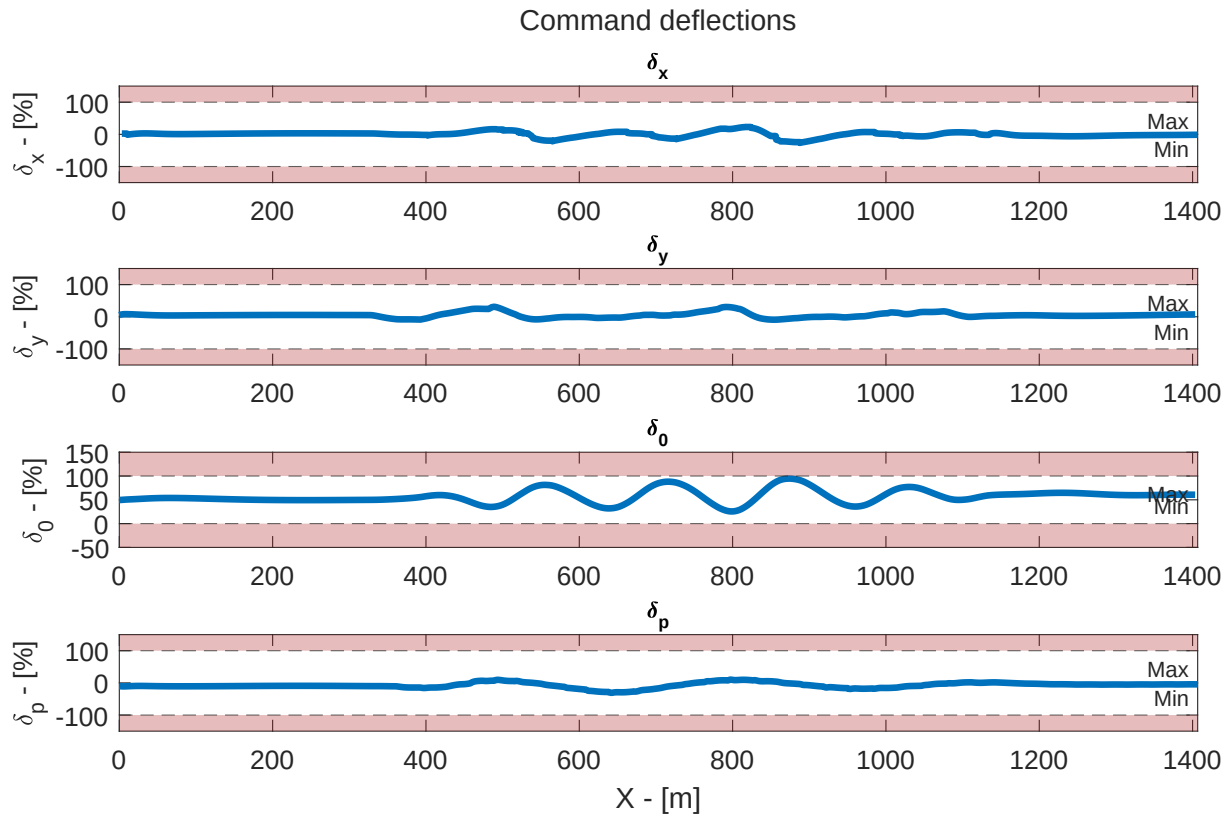


Figure 12.5: Control actuation during the slalom MTE

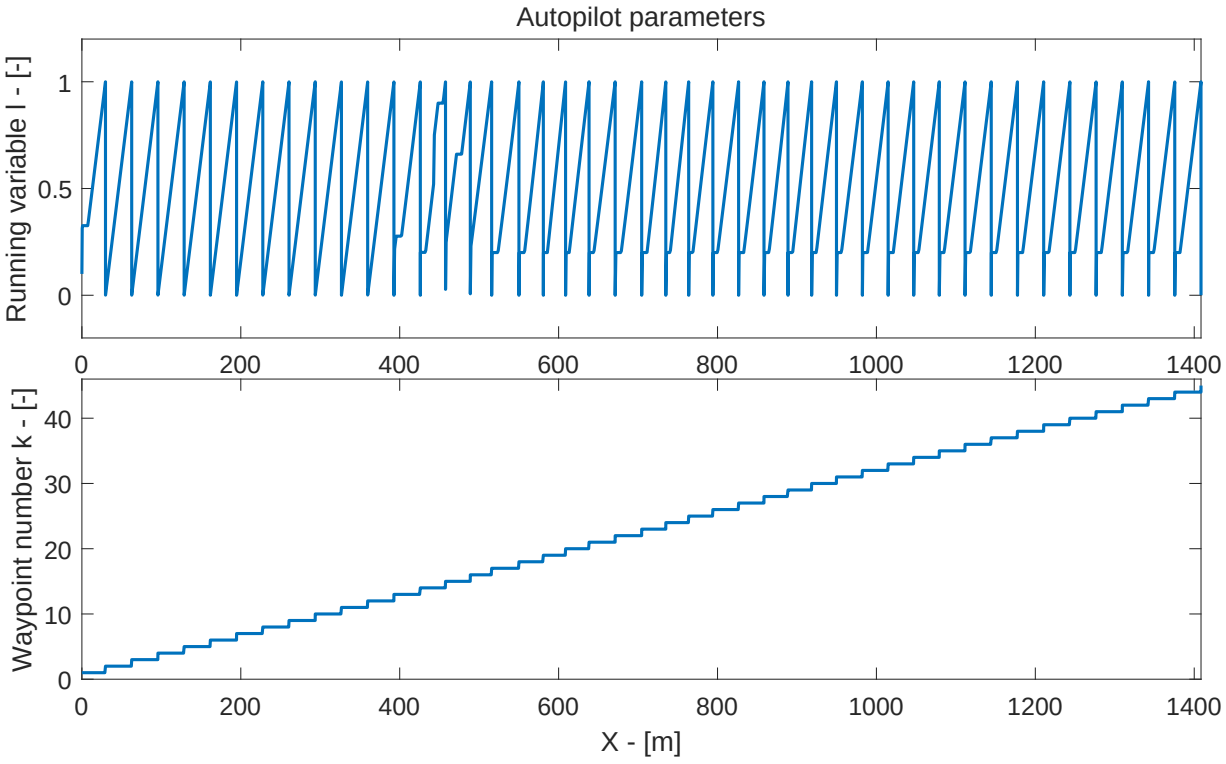


Figure 12.6: Autopilot running variable l and waypoint number k

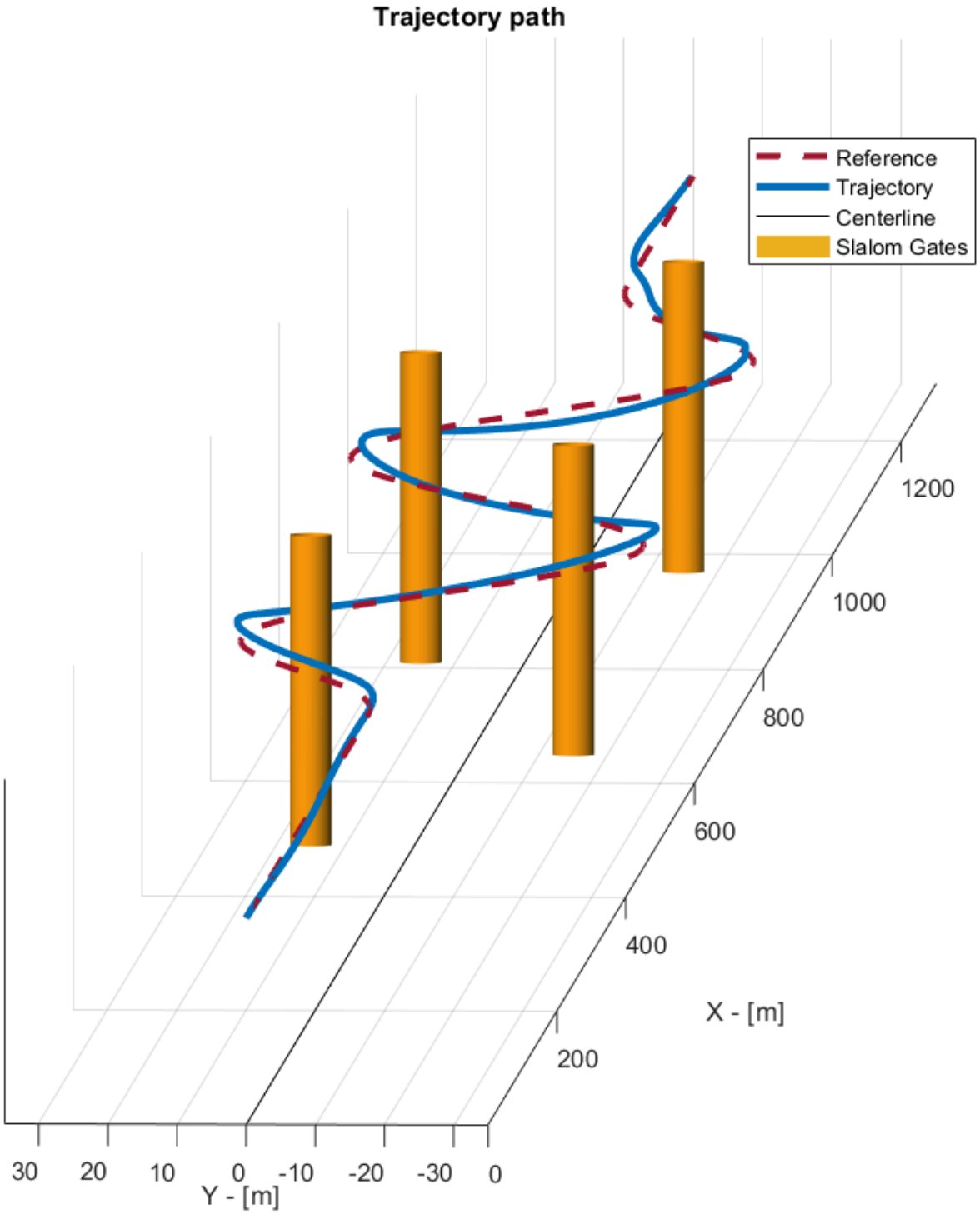


Figure 12.7: 3D representation of the helicopter's autonomous flight in the slalom MTE

12.2. Pop-up MTE simulation

Within this section, an analysis of the system's simulated response to the pop-up MTE is carried out. The reference signals used for the purpose of the MTE simulation have been identified using the procedure shown in Section 10.2.2 and the overall system's performance will be discussed by analysing how closely the reference trajectory is being matched and the control effort required by the system to perform this tracking task.

As explained in Section 10.2.2, the pop-up manoeuvre is a completely longitudinal manoeuvre consisting of an altitude change and hold at constant heading. Similarly to the slalom manoeuvre, the MTE is divided in three segments: an initial forward flight segment, the central pop-up segment where the helicopter is tasked with changing its altitude, and a final forward flight segment while holding the commanded altitude. Given the nature of the manoeuvre, the primary channel that will be excited is the heave velocity w, with only minor expected changes in the helicopter's pitch θ to enter and exit the altitude change segment of the MTE.

Hereafter, the results of the simulations are presented and discussed. The results of the simulation for the pop-up manoeuvre are shown in Figures 12.8 to 12.12: these values will be analysed and discussed individually, highlighting the most relevant characteristics.

To start, in Figure 12.8 the evolution of the position signals is presented. As the manoeuvre is entirely longitudinal, the highest excitation of the system is found alongside the Z axis, with only minimal variations on the lateral direction corresponding to the beginning and the end of the central manoeuvre portion: this is consistent with the much smaller coupling, which was identified during the position loop tuning in Section 9.4. The response of the system is slightly delayed compared to the reference position signals: this is consistent with the results of the tuning and was also noticed in the lateral response during the slalom MTE. This lagging behaviour in the position is caused by the delayed response in the velocity control on the vertical axis, which causes the system to begin gaining altitude later than what would have been expected. This delay in the response is also noticed when the helicopter transitions from the pop-up manoeuvre to the final altitude hold segment, where it is noticed that the aircraft first overshoots the desired altitude by about 3m before slowly descending and settling at the desired 95m altitude.

The velocity response of the system is shown in Figure 12.9, where the measured simulated helicopter velocity is compared to its reference. As expected, the most noticeable velocity activation is found in the heave, which is commanded to reach a velocity of $-9\text{m}/\text{s}$ before ascending again to have no vertical velocity component. This figure clearly shows the lagging vertical velocity response that was discussed above and highlights how the limits and delays in the system's velocity response are reflected in the position tracking performance.

Lastly, Figures 12.10 and 12.11 respectively show the helicopter's attitude response expressed using the quaternion parametrization and the Euler angles parametrization. As expected, variations in the attitude of the system are minimal, with the largest variations being found in the pitch angle θ. Once again, this behaviour is expected of the system: since the heave controller is separate from the attitude controller (as explained in Section 8.3), the vertical velocity component is almost entirely controlled by varying the helicopter's collective command, and variations in the pitch axis are entirely due to the required variations in the surge velocity commanded during climb.

Concluding the analysis of the system's response to the pop-up manoeuvre reference, Figure 12.12 shows the command deflections during the manoeuvre. Once again, the results match the expected behaviour of the system, with the primary control input used being the collective command to determine changes in the heave velocity w; only minor variations are found in the longitudinal cyclic control to match the commanded changes in velocity. The lateral controls (pedal and lateral cyclic) remain almost unvaried from their trim positions, as would be expected in a lateral manoeuvre. In all cases, the control inputs never saturate the available command limits, suggesting that the tuning procedure proposed in this thesis shows effectiveness also in the longitudinal direction.

Overall, the simulation results suggest that the approach used when identifying and tuning the controller and the methodology developed to generate appropriate reference values for the execution of the pop-up

manoeuvre are effective in enabling the system to navigate altitude changes. The response of the system, although slightly delayed from the reference, shows good accuracy, with a maximum deviation of 3m from the desired vertical position due to an overshoot at the peak of the manoeuvre. Additionally, thanks to the integral action of the velocity controller and the addition of a position control loop, the helicopter is capable of reducing errors in the position when remaining in a steady-state flight configuration, as is shown by the evolution of the altitude in the final segment of Figure 12.8, where the helicopter recovers from the overshoot before settling at the desired altitude.

Furthermore, the attitude tracking during the pop-up manoeuvre remains robust, with the pitch angle commands being tracked effectively by the system to regulate the surge velocity, while roll and yaw angles are maintained near zero throughout the climb and descent phases. Control inputs, displayed in Figure 12.12, exhibit smooth collective variations that mirror the altitude profile without exceeding actuator limits. Additionally, as the reference position matches the offline-calculated pop-up trajectory, it also follows that the autopilot's real-time interpolation ensures is effective in smoothly reconstructing the complete trajectory, allowing the entire manoeuvre to not be stored as a set of continuous reference signals, but instead as a set of relevant waypoints. Collectively, these results demonstrate that the integrated control and autopilot architecture delivers both precision and stability across rapid vertical flight profiles.

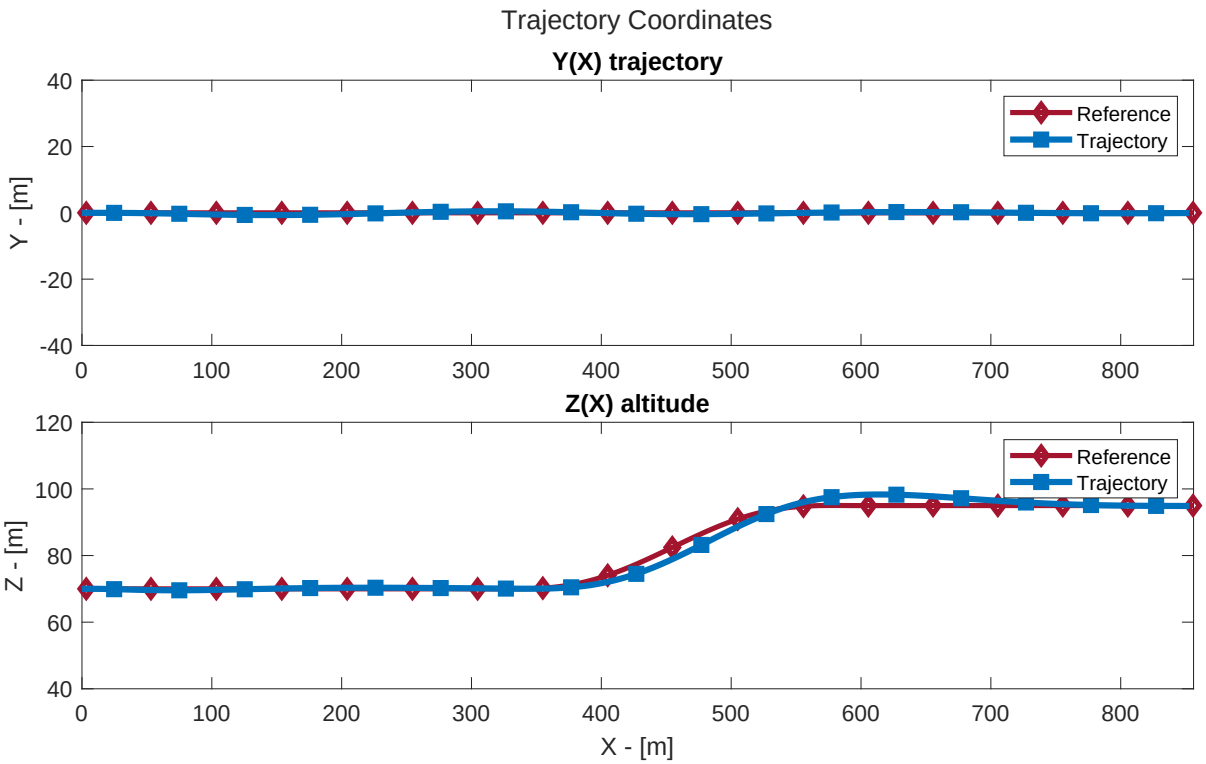


Figure 12.8: Autonomous flight tracking of pop-up trajectory reference

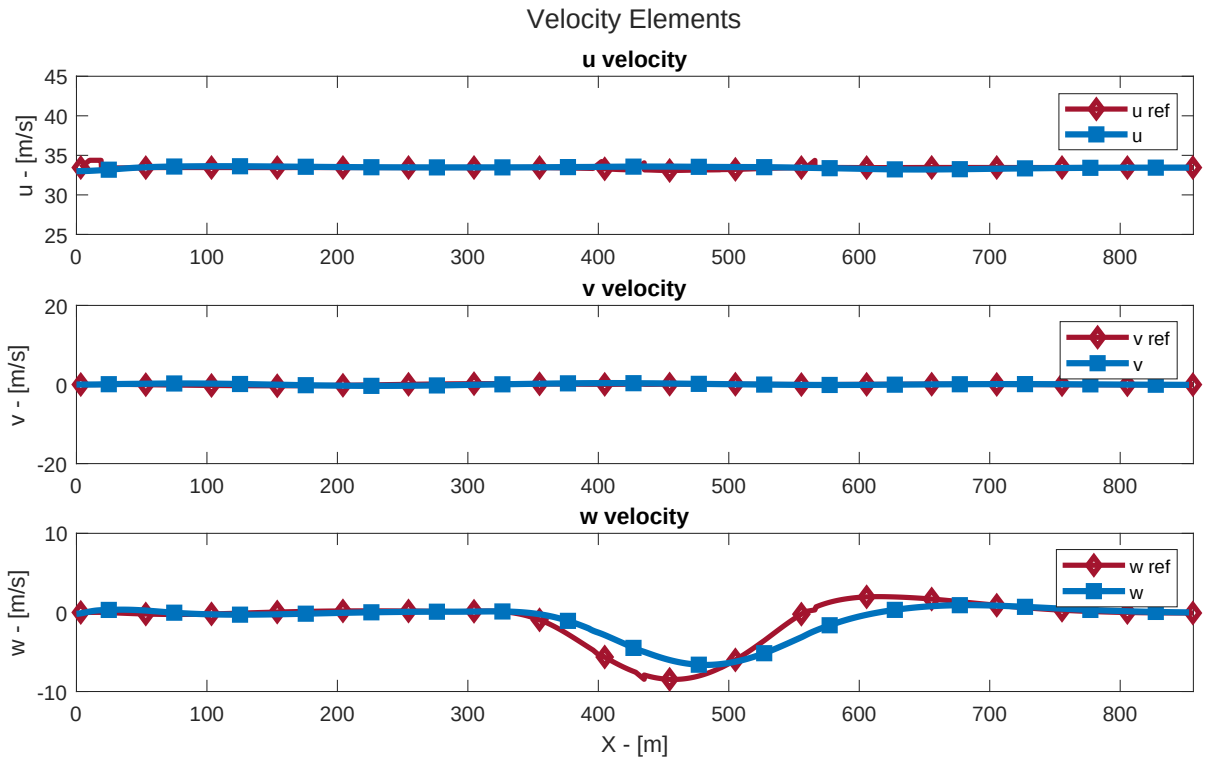


Figure 12.9: Velocity references during the pop-up MTE

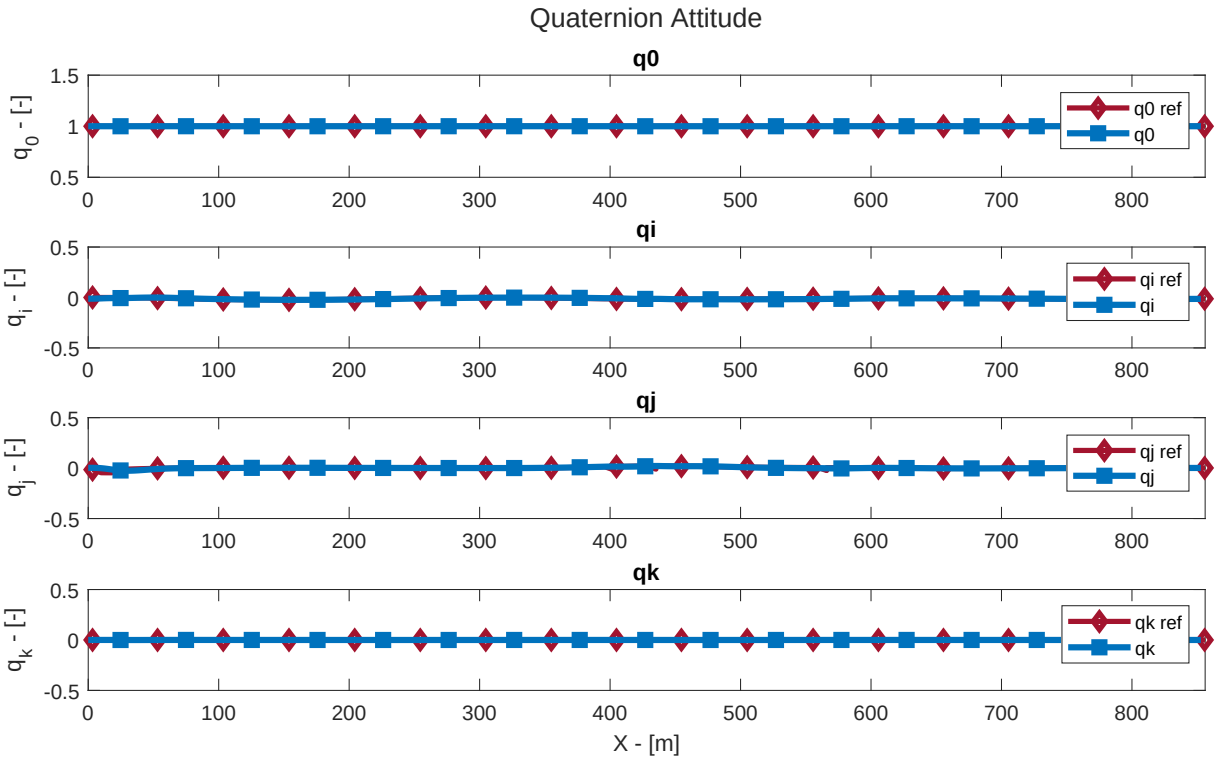


Figure 12.10: Quaternion attitude references during the pop-up MTE

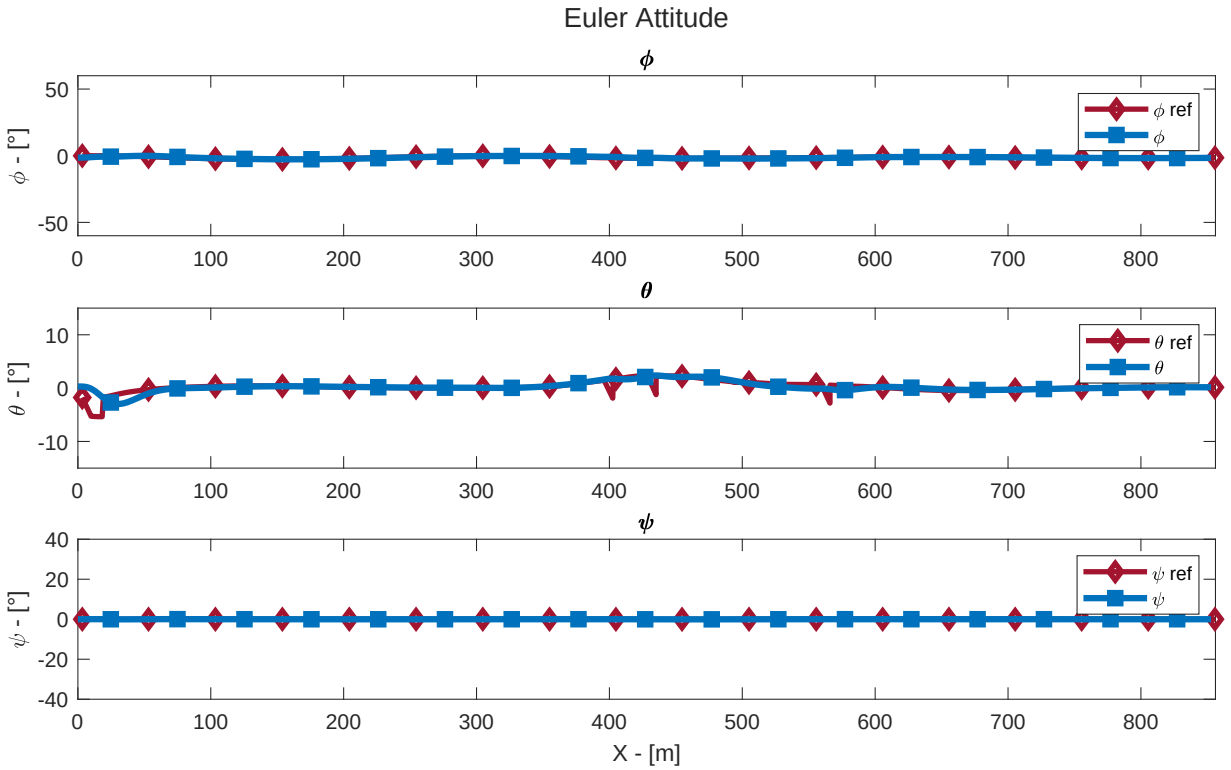


Figure 12.11: Euler angles attitude references during the pop-up MTE

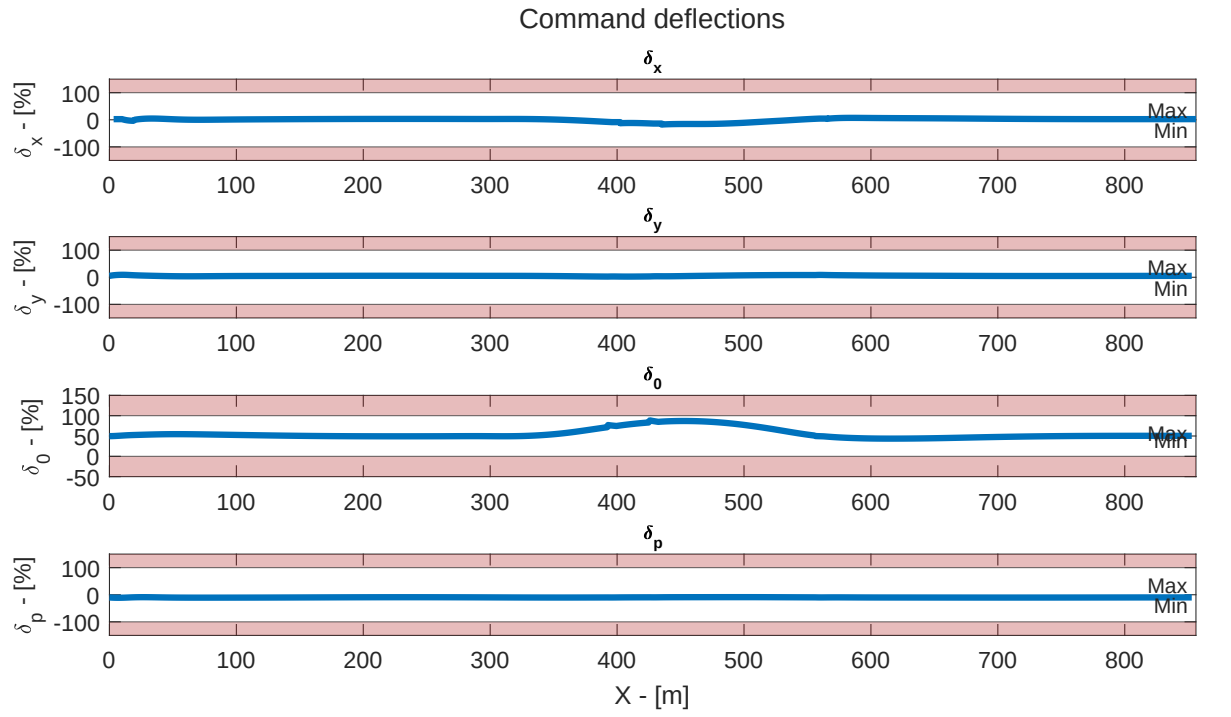


Figure 12.12: Control actuation during the pop-up MTE

12.3. Concluding remarks on helicopter tracking performance

Within this chapter, the controller and the autonomous flight systems developed within the thesis were verified in their implementation by simulating the execution of two complementary manoeuvres: the slalom and the pop-up. The simulations were carried out making use of the Simulink environment, which enables the simulation of dynamical systems, and the simulation parameters were set up to have a simulation update timestep of 1ms using a Fourth-Order Runge-Kutta (RK4) solver, enabling the system to function at a rate that is consistent with the achieved rates of sensors and computers that may be used in flight control scenarios while using a solver algorithm that strikes a balance between computational effort and simulation accuracy.

The manoeuvres simulated were a slalom and a pop-up. These two manoeuvres were selected as they are commonly used in the context of piloted flight to test the agility of an air vehicle and because they allow testing of both the lateral and the longitudinal manoeuverability of the system.

The slalom manoeuvre was simulated first, commanding continuous variations in both the roll angle ϕ and the yaw angle ψ while navigating a continuously curving trajectory that aimed at achieving a sinusoidal flight path with curves spaced by 150m achieving a 25m variation from the centerline of the track. The manoeuvre references were identified offline using the methodology discussed in Section 10.2.1 and leveraging the online-smoothing capabilities of the autopilot module discussed in Section 11.1.2. Results of the simulation show that the aircraft was successful in achieving good trajectory tracking, always remaining within 5m of the desired lateral position and with similar maximum displacements along the vertical axis. While the velocity reference tracking performance suggest that improvements in the tuning of the intermediate controller loop would benefit the system, the attitude tracking performance show remarkable accuracy, with only minimal variations and deviations from the reference signals, showing the effectiveness of quaternions in encoding simultaneous changes along all attitude axes with a robust methodology and minimal computational effort. The inputs measured around the manoeuvre are consistent with the vehicle's response and never reach saturation, validating the multi-objective tuning methodology introduced in this work: the large activations of the helicopter's collective command are a consequence of the high cross-activation between lateral and vertical velocity variations.

The pop-up manoeuvre was simulated next, commanding the helicopter to perform an altitude variation while maintaining constant heading. In particular, the manoeuvre was implemented to command a smooth 25m altitude gain and hold, requiring contained variations in the aircraft's pitch angle θ and commanding a collective activation to increase the vertical velocity component of the system. The manoeuvre reference was generated using the approach presented in Section 10.2.2, where the pop-up manoeuvre was introduced and identified. The response was overall satisfactory, with the helicopter being able to perform the altitude variation calculated offline. Similarly to the slalom manoeuvre, the velocity response of the system showed non-negligible delays in the heave response, which also impact the position tracking performance of the helicopter, causing a momentary overshoot of the desired altitude. Even with this, the helicopter was able to later settle at the desired 95m altitude, showing the effectiveness of the integrated autopilot and position controller. The attitude response is again able to follow the desired references, and the manoeuvre is executed without saturating the available controls, with the collective control showing a smooth rise and descent during the pop-up section of the manoeuvre before settling at a value that is very close to its original trim during the final altitude hold phase, as expected.

Part VII

Closure

13

Conclusion and Recommendations

Within this chapter, concluding remarks regarding the work performed and the results obtained in the context of the thesis will be presented, synthesising the findings of this work and providing explicit answers to the research questions previously identified.

To provide an orderly and effective conclusion to the work performed, this chapter will be divided in two sections. Section 13.1 will provide a contextualized overview of the work performed, recalling the research questions identified in Section 1.2 and synthesising the methodology followed and the results obtained. Afterwards, Section 13.2 will highlight potential future development of the work performed and present useful recommendations to guide the possibility of improving the methodologies and approaches developed within this work. To provide an immediate reference of the relevance of this work, a concise overview of the main contributions of this work has already been provided in Chapter 4, where the main contributions of the work have been listed, referencing the sections of the work where these results have been discussed.

13.1. Overview of the work and research questions

With a growing interest in the application of quaternions to full-scale flight controllers, but with little works performed in the study of quaternion models and quaternion control system for atmospheric flight, this thesis sets out to contribute to the current research landscape by exploring, with a practical implementation case study, the inclusion of quaternion modelling techniques, and the development and implementation of a quaternion-based full flight control system for autonomous flight.

Here the methodology developed and the results obtained are summarized, providing a complete overview of the work done. Additionally, the research questions driving this work, which had been identified in Section 1.2, are also included in the overview of the work, allowing the reader to gain immediate understanding of how they were developed and the answers identified in this work.

As the primary interest of the thesis is the investigation and the implementation of the quaternion attitude parametrization to highly-manoeuverable air vehicles (such as helicopters), in Chapter 5 an introduction to the quaternion formulation was given, providing a concise overview of the applications of quaternions in the context of aerospace engineering. In this chapter, the first research question was also addressed:

Research Question 1

How are quaternions beneficial to the modelling of helicopter dynamics and to the development of control systems?

To answer this question, quaternions were compared with two of the most commonly-used attitude parametrization parametrizations, Euler angles and Direction Cosine Matrices, showing that quaternions provide a computationally-efficient and highly-robust approach to attitude parametrization, removing the risk of gimbal lock and enabling faster computations: the two primary limitations of traditional attitude parametrization approaches.

With the advantages of quaternions established, in Chapter 6 and Chapter 7 their implementation to existing systems was investigated to provide a reliable approach to investigate quaternion models and to develop quaternion-based control systems. To this end, Research Question 2 was investigated:

Research Question 2

How can an existing state-space flight dynamics model be augmented to include quaternions for its attitude representation?

To answer this question, in Section 7.1 a methodology to convert existing linearized models from the Euler angles to the quaternion attitude parametrization based on mathematical considerations was provided, and the conversion of the model was verified by comparing the system's poles, eigenvectors, controllability/observability characteristics, and dynamic response before and after the quaternion conversion. The results are positive, suggesting that the attitude parametrization conversion procedure is effective in maintaining the fundamental system's characteristics, enabling the study of quaternion-based linearized models without the need to perform complex identification procedures if an existing Euler angles-based model is available.

Building upon the results of Chapter 7, which allowed for the construction of a linearized system model with an integrated quaternion attitude parametrization, the task of controlling the existing helicopter model is then investigated. To this end, Research Question 3 further guides the development of the thesis by investigating the inclusion of quaternions in flight control systems.

Research Question 3

How can a quaternion-based autonomous full flight control system be optimally developed for an agile helicopter?

To answer this question, in Chapter 8 an architecture for a full flight control system for helicopter control is presented. The proposed control system is based on a modular architecture composed of three nested loops: the innermost loop controls the quaternion attitude of the system, the middle loop controls the system's velocity, and the outer loop controls the position of the vehicle. While the middle and outer loops are generated using a P/PI formulation, the inner attitude loop leverages a novel implementation of LQI control to the quaternion attitude parametrization, which enables good decoupling of the attitude channels while ensuring responsiveness and moderate control usage.

Expanding upon the controller architecture discussed, Chapter 9 delves into the tuning process. In an effort to reduce the need for manual tuning, the tuning was performed for the three loops using the Particle Swarm Optimization (PSO) algorithm, which allows for the identification of multiple performance objectives that can be optimized simultaneously to find a set of controller parameters to achieve the desired performance. The tuning was performed minimizing tracking error with appropriate reference signals while limiting control usage to avoid saturations, resulting in an effective and repeatable tuning procedure that can be applied to multiple different systems. The tuning was performed and discussed for all three loops, every time discussing how the PSO algorithm was implemented, allowing for a solid understanding of the system's performance and of the methodology used.

With the control system identified and tuned, to investigate the system's performance, appropriate tracking tasks had to be identified to evaluate controller performance, and as such adequate reference signals had to be investigated to allow for the execution of selected Mission Task Elements. To this end, Research Question 4 was identified:

Research Question 4

How can maneuver reference trajectories be systematically designed to evaluate the tracking performance of a helicopter control system?

To this end, in Chapter 10 a deterministic approach to manoeuvre reference identification for the helicopter is developed and presented, building a simple framework that, from simple kinematic and dynamic equations, allows to determine the required position, velocity, and attitude references that the helicopter should follow to track a desired manoeuvre in 3D space. The proposed algorithm does not rely on any model knowledge and integrates well with the nested-loop control system to allow for online compensation of the simplifying hypotheses utilized when defining the manoeuvres.

The methodology proposed was then utilized to generate reference values for two complementary tracking tasks: a slalom manoeuvre and a pop-up manoeuvre. These manoeuvre were identified as they allow for an exploration of both the lateral and the longitudinal responses of the vehicle, testing the autopilot's and the controller's ability to adequately perform under various flight conditions.

With adequate manoeuvres identified to test the controlled helicopter's performance, the ability of the system to follow the offline-defined reference signals is tested via desktop simulations. In the context of autonomous navigation based on identified trajectories, however, continuous control signals are inadequate as they do not allow for the consideration of possible delays in the system's response, and will lead to inaccurate trajectories that will not perform well in tracking tasks where precise position tracking is desired. To this end, Research Question 5 was defined:

Research Question 5

How can an autonomous flight system be designed to optimize reference tracking for agile maneuvers in helicopters?

Answering this question, in Chapter 11 an autopilot system is introduced and clearly detailed in its functioning and implementation. The autopilot is build as a Finite State Machine (FSM) that is capable of tracking the system's current position and, collaborating with the controller, dynamically produce adequate position, velocity, and attitude references to control the motion of the vehicle. The choice of the FSM is instrumental in enabling a simple but effective architecture, that operated reliably and that could be easily expanded upon in future research to allow for more complex behaviour like real-time trajectory-planning to allow for obstacle avoidance.

Concluding the description of the system, the two manoeuvres identified with the methodology introduced in Chapter 10 are simulated in MATLAB's Simulink environment. The simulations are both performed with a time resolution that matches the performance of aircraft on-board computer and using a solver algorithm that balances good accuracy with limited computational load, and the results of the simulations are discussed in Chapter 12.

In both cases, the helicopter is able to adequately track the desired trajectories, showing only limited deviations from the offline-defined trajectory at the most aggressive points of the manoeuvres. Analysing the results, it was identified that the primary limiting factors were the velocity response, which in the tuning process showed non-negligible delays across all control axes, and the high cross-activation of the vertical velocity when inputs were provided in the horizontal and lateral velocity directions. Conversely, the inner-loop attitude tracking appeared to be effective in its implementation, showing remarkable responsiveness, suggesting the validity of the model conversion technique to have the attitude be internally represented with quaternions discussed in Chapter 7, and of the novel LQI implementation to quaternion attitude control introduced in Chapter 8.

Further, in both simulations, the helicopter was capable of effectively limiting control usage, never saturating the system's input commands. This, paired with the good tracking performance discussed above, supports the applicability of the objective-based tuning procedure utilized in Chapter 9, showing that the PSO algorithm is capable of tuning multi-loop system and providing clear cost function definitions that allow for the effective tuning of multi-loop controllers for dynamic systems.

In closing, this thesis has delivered an effective framework for quaternion-based full flight control in agile helicopters, eliminating Euler angle limitations and ensuring singularity-free attitude representation.

A procedure to convert existing linearized models from the internal Euler angles parametrization to the quaternion attitude parametrization has been presented, eliminating the need to perform complex identification procedures, thus enabling a more immediate study of a system's performance using a quaternion approach. The hierarchical three-loop architecture, tuned via Particle Swarm Optimization and supported by a modular FSM autopilot, has demonstrated exceptional tracking performance in both lateral slalom and longitudinal pop-up manoeuvres, achieving sub-5m lateral and sub-3m vertical errors without control saturation for two selected manoeuvres: the slalom and the pop-up. By bridging quaternion methodologies from UAV and spacecraft research to full-scale rotorcraft, this work contributes to the academic discussion in autonomous VTOL flight of agile air mobility systems.

13.2. Future developments and recommendations

As discussed above, this thesis work covers a wide variety of topics, providing a comprehensive discussion of the entire procedure followed to analyse and discuss a helicopter model, implement and tune a full flight control system, and build a simple but effective algorithm to allow for autonomous navigation through offline-defined paths. Here, a brief overview of the primary recommendations for the continuation of this research is provided.

Rec 1: Evaluation of linearization effectiveness at multiple operating points and comparison with full nonlinear quaternion models
While the present work focuses on the discussion of the model conversion at two operating points (hover and 65kn linearizations), future developments could investigate how the accuracy and usefulness of the linearized quaternion-based model vary across a broader range of operating points. In particular, linearizing the system around different flight regimes—such as low-speed forward flight, aggressive climb, or descent—would allow for a better understanding of how well the linearized dynamics capture the true system behavior in varying conditions. This analysis could guide the design of gain-scheduled or regionally-tuned controllers that adapt their parameters based on flight state, thereby improving robustness and tracking performance throughout the entire flight envelope.

Building upon this, from a modelling standpoint, another useful development of this work would be a comparison between the linearization of a complete non-linear quaternion-based helicopter model and a linearized Euler angles model converted to the quaternion parametrization using the proposed methodology. This would serve to further validate the procedure presented in this work, and would also provide useful insight between the accuracy of the two linearization procedures, comparing computational effort, assessing eventual linearization incongruences, and evaluating how well the dynamics of the non-linear quaternion model are captured by the converted linearized Euler angles model.

Rec 2: integration of velocity cross-feedforward gains
One crucial limitation of the control system developed is the high coupling between lateral and vertical response, where sway velocity commands or lateral position changes are also reflected in significant vertical displacements. One possible area of future expansion of this work would be the application of the proposed tuning methodology to a different control architecture that may introduce cross-feedforward gains to limit the influence of the helicopter's natural coupling.

Rec 3: integration of robustness goals in the tuning
The flexibility of the PSO control system may also be used to include design considerations regarding the robustness of the closed-loop system, thus adapting the control strategy developed to also account for stability margins and disturbance-rejection criteria. This would significantly improve the performance of the tuning strategy, as it would allow extend the tuning approach to consider both time-domain and frequency-domain characteristics of any system this approach is applied to.

Rec 4: integration of q_0 element in the attitude controller
As an additional point of note in controller architecture design, while the inner-loop LQI was highly effective at tracking attitude variations around the trim point of the vehicle, it is worth remembering that the attitude is now being tracked using only three of the four quaternion elements, neglecting the q_0 term. As explained in Chapter 7 and Chapter 8, this was done to maintain the system's controllability condition, thus allowing for the investigation of the application of LQI. In an effort to enable more accurate and robust attitude tracking

policies, an attitude controller capable of accounting for references across all four quaternion elements is therefore desirable and worth investigating. While to this end a change in the controller paradigm may be useful, the tuning procedure presented here may still be used and applied tot he tuning of another, different control system.

Rec 5: expansion of autopilot FSM capabilities
The autopilot controller presented in Chapter 11 is a simple but functional autonomous flight system, with a highly versatile and modular structure. Because of this simplicity and modularity, the FSM presented is well-suited to being expanded upon by including additional subsystems to enable more complex behaviour. As possible examples, the FSM could be augmented with an appropriate online trajectory planning algorithm to enable obstacle avoidance, or by expanding its capabilities to travel through trajectories that can seamlessly merge different manoeuvres.

Further, as the autopilot FSM is model-agnostic (meaning that it does not rely on any existing model knowledge and it was presented using a general formulation) it could be applied to a number of different-scale and different-purpose vehicles, such as small drones, conventional aircraft, or even automotive. This allows for a wide range of possible applications of this seemingly basic autopilot, enabling its investigation in more complex systems.

References

[1] J. Vince. *Quaternions for Computer Graphics*. SpringerLink : Bücher. Springer London, 2011. URL: https://books.google.co.uk/books?id=CDgrYqDsSBAC.

[2] J. Kuipers. “Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality”. In: (2002). DOI: 10.5860/choice.37-0370.

[3] Z. Terze et al. “Aircraft attitude reconstruction via novel quaternion-integration procedure”. In: *Aerospace Science and Technology* 97 (2020), p. 105617. DOI: 10.1016/j.ast.2019.105617.

[4] Peidong Wang et al. “Dynamic Modeling and Control of Multibody Systems Using Dual Quaternions”. In: *Journal of Guidance, Control, and Dynamics* (2024). DOI: 10.2514/1.g008104.

[5] A. Valverde et al. “Spacecraft Robot Kinematics Using Dual Quaternions”. In: *Robotics* 7 (2018), p. 64. DOI: 10.3390/ROBOTICS7040064.

[6] Michał Gołąbek et al. “Quaternion attitude control system of highly maneuverable aircraft”. en. In: *Electronics (Basel)* 11.22 (Nov. 2022), p. 3775.

[7] Tiangao Zhu et al. “The quaternion based error model based on SE(3) of the INS”. In: *IEEE Sens. J.* 22.13 (July 2022), pp. 13067–13077.

[8] Maher Tarek et al. “Performance analysis of quaternion vs Euler-based approaches in reduced INS/GNSS fusion techniques”. In: *2023 International Telecommunications Conference (ITC-Egypt)*. Alexandria, Egypt: IEEE, July 2023, pp. 530–537.

[9] Marco B Gerig. “Modeling, guidance, and control of aerobatic maneuvers of an autonomous helicopter”. PhD thesis. 2008.

[10] Xiaojun Duan et al. “Attitude Tracking Control of Small-Scale Unmanned Helicopters Using Quaternion-Based Adaptive Dynamic Surface Control”. In: *IEEE Access* 9 (2021), pp. 10153–10165. DOI: 10.1109/ACCESS.2020.3043363.

[11] Satoshi Suzuki et al. “Quaternion-based navigation and control for small unmanned helicopter”. en. In: *IFAC Proc. Vol.* 43.15 (2010), pp. 37–42.

[12] Dmitriy Kritskiy et al. “Increasing the reliability of drones due to the use of quaternions in motion”. In: *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. Kyiv, Ukraine: IEEE, May 2018.

[13] Dinesh D. Dhadekar et al. “Robust Control of Quadrotor using Uncertainty and Disturbance Estimation”. In: *Journal of Intelligent Robotic Systems* 101 (2021). DOI: 10.1007/s10846-021-01325-1.

[14] Jurian Stougie et al. “Incremental nonlinear dynamic inversion control with flight envelope protection for the flying-V”. In: *AIAA SCITECH 2024 Forum*. Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2024.

[15] Joseph F Horn. “Non-linear dynamic inversion control design for rotorcraft”. en. In: *Aerospace* 6.3 (Mar. 2019), p. 38.

[16] Pedro Simplício. “Helicopter nonlinear flight control: An acceleration measurements-based approach using incremental nonlinear dynamic inversion”. In: 2011. URL: https://api.semanticscholar.org/CorpusID:113692845.

[17] Michael Nikolaou. “Model predictive controllers: A critical synthesis of theory and industrial needs”. In: *Advances in Chemical Engineering* 26 (2001), pp. 131–204. URL: https://api.semanticscholar.org/CorpusID:1402032.

[18] L. Wang. *Model Predictive Control System Design and Implementation Using MATLAB®*. Advances in Industrial Control. Springer London, 2009. URL: https://books.google.co.uk/books?id=vcc_AAAAQBAJ.

[19] Julian Berberich et al. “Linear Tracking MPC for Nonlinear Systems—Part I: The Model-Based Case”. In: *IEEE Transactions on Automatic Control* 67.9 (2022), pp. 4390–4405. DOI: 10.1109/TAC.2022.3166872.

[20] Johan Ubbink et al. “From Instantaneous to Predictive Control: A More Intuitive and Tunable MPC Formulation for Robot Manipulators”. In: *IEEE Robotics and Automation Letters* 10 (2024), pp. 748–755. DOI: 10.1109/LRA.2024.3511439.

[21] M. Alhajeri et al. “Tuning Guidelines for Model-Predictive Control”. In: *Industrial Engineering Chemistry Research* (2020). DOI: 10.1021/acs.iecr.9b05931.

[22] Qiugang Lu et al. “MPC Controller Tuning using Bayesian Optimization Techniques”. In: *ArXiv* abs/2009.14175 (2020).

[23] P. Bouffard. “On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments”. In: (2012). DOI: 10.21236/ada572108.

[24] Utku Eren et al. “Model predictive control in aerospace systems: Current state and opportunities”. en. In: *J. Guid. Control Dyn.* 40.7 (July 2017), pp. 1541–1566.

[25] Labane Chrif et al. “Aircraft control system using model predictive controller”. In: *TELKOMNIKA Indones. J. Electr. Eng.* 15.2 (Aug. 2015), p. 259.

[26] Tuğrul Oktay et al. “Constrained predictive control of helicopters”. In: *Aircraft Engineering and Aerospace Technology* 85 (2013), pp. 32–47. DOI: 10.1108/00022661311294021.

[27] Ying-Chih Lai et al. “Adaptive Learning-Based Observer With Dynamic Inversion for the Autonomous Flight of an Unmanned Helicopter”. In: *IEEE Transactions on Aerospace and Electronic Systems* 57 (2021), pp. 1803–1814. DOI: 10.1109/TAES.2021.3050653.

[28] Cunjia Liu et al. “Tracking control of small-scale helicopters using explicit nonlinear MPC augmented with disturbance observers”. In: *Control Engineering Practice* 20 (2012), pp. 258–268. DOI: 10.1016/J.CONENGPRAC.2011.10.015.

[29] Vishal et al. “GA tuned LQR and PID controller for aircraft pitch control”. In: *2014 IEEE 6th India International Conference on Power Electronics (IICPE)*. 2014, pp. 1–6. DOI: 10.1109/IICPE.2014.7115839.

[30] Muhammet Ali Güneş et al. “Stabilization of a Fixed-Wing Aircraft with a Nonlinear Dynamic Inversion-Based PI Controller”. In: *2024 15th National Conference on Electrical and Electronics Engineering (ELECO)*. 2024, pp. 1–6. DOI: 10.1109/ELECO64362.2024.10847254.

[31] Amelia Chindrus et al. “Continuous Manufacturing using Linear Quadratic Regulator in the Context of Cyber-Physical Systems”. In: *2022 26th International Conference on System Theory, Control and Computing (ICSTCC)*. 2022, pp. 231–236. DOI: 10.1109/ICSTCC55426.2022.9931791.

[32] Wolfgang Birk. “Industry Applications of Multivariable Control”. Available at https://www.diva-portal.org/smash/get/diva2:990581/FULLTEXT01.pdf. PhD thesis. SE-971 87 Luleå, Sweden: Luleå University of Technology, Aug. 2002.

[33] Nurul Dayana Salim et al. “PID plus LQR attitude control for hexarotor MAV in indoor environments”. In: *2014 IEEE International Conference on Industrial Technology (ICIT)*. 2014, pp. 85–90. DOI: 10.1109/ICIT.2014.6894977.

[34] Titilayo Ogunwa et al. “Modeling and control of an articulated multibody aircraft”. en. In: *Appl. Sci. (Basel)* 12.3 (Jan. 2022), p. 1162.

[35] Labane Chrif et al. “Aircraft control system using LQG and LQR controller with optimal estimation-Kalman filter design”. en. In: *Procedia Eng.* 80 (2014), pp. 245–257.

[36] Taha Hamid et al. “Pitch Attitude Control for an Aircraft using Linear Quadratic Integral Control Strategy”. In: *2019 22nd International Multitopic Conference (INMIC)*. 2019, pp. 1–6. DOI: 10.1109/INMIC48123.2019.9022787.

[37] Erkan Abdulhamitbilal et al. “Gain scheduled LQ optimal control of a parametric light commercial helicopter model at sea level”. In: *Proceedings of the 9th WSEAS International Conference on Automatic Control, Modelling and Simulation*. ACMOS’07. Istanbul, Turkey: World Scientific, Engineering Academy, and Society (WSEAS), 2007, pp. 299–306.

[38] Christos Papachristos et al. “Linear Quadratic optimal position control for an unmanned Tri-TiltRotor”. In: *2013 International Conference on Control, Decision and Information Technologies (CoDIT)*. Hammamet, Tunisia: IEEE, May 2013.

[39] M. Green et al. *Linear Robust Control*. Dover Books on Electrical Engineering. Dover Publications, Incorporated, 2012. URL: https://books.google.co.uk/books?id=CI-DyLffACcC.

[40] A. Marcos et al. “Flight testing of an structured H-infinity controller: An EU-Japan collaborative experience”. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)* (2017), pp. 2132–2137. DOI: 10.1109/CCTA.2017.8062768.

[41] Sigurd Skogestad et al. *Multivariable Feedback Control: Analysis and Design (second edition)*. Wiley, Jan. 2005.

[42] R. Hyde et al. “The application of scheduled H∞ controllers to a VSTOL aircraft”. In: *IEEE Trans. Autom. Control.* 38 (1993), pp. 1021–1039. DOI: 10.1109/9.231458.

[43] D. Jeong et al. “H-Infinity Attitude Control System Design for a Small-Scale Autonomous Helicopter with Nonlinear Dynamics and Uncertainties”. In: *Journal of Aerospace Engineering* 25 (2012), pp. 501–518. DOI: 10.1061/(ASCE)AS.1943-5525.0000176.

[44] Jim Tighe. “Autonomy and future mobility in aerospace”. Feb. 2024.

[45] Sakshi Mishra et al. “Autonomous Advanced Aerial Mobility—An End-to-End Autonomy Framework for UAVs and Beyond”. In: *IEEE Access* 11 (2023), pp. 136318–136349. DOI: 10.1109/ACCESS.2023.3339631.

[46] Eric Anderson et al. “Levels of aviation autonomy”. In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. London: IEEE, Sept. 2018.

[47] Antonio Loquercio et al. “Learning high-speed flight in the wild”. In: *arXiv [cs.RO]* (2021).

[48] Abderrahim Waga et al. “Efficient autonomous navigation for mobile robots using machine learning”. In: *IAES Int. J. Artif. Intell. (IJ-AI)* 13.3 (Sept. 2024), p. 3061.

[49] Qingguo Zhou et al. *Theories and practices of self-driving vehicles*. en. Philadelphia, PA: Elsevier - Health Sciences Division, July 2022.

[50] Nada S. Kassem et al. “Behavior Planning for Autonomous Driving: Methodologies, Applications, and Future Orientation”. In: *MSA Engineering Journal* 2.2 (2023), pp. 1–12. DOI: 10.21608/MSAENG.2023.291918.

[51] Wilko Schwarting et al. “Planning and decision-making for autonomous vehicles”. en. In: *Annu. Rev. Control Robot. Auton. Syst.* 1.1 (May 2018), pp. 187–210.

[52] Simon Ulbrich et al. “Towards a functional system architecture for automated vehicles”. In: (2017). eprint: 1703.08557 (eess.SY).

[53] Changxi You et al. “Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning”. en. In: *Rob. Auton. Syst.* 114 (Apr. 2019), pp. 1–18.

[54] Billy Elly et al. “Autonomous Navigation in Dynamic Environments: A Comparative Study of AI-Driven Robotics Approaches”. In: (Nov. 2024).

[55] Sushil Kumar Sahoo et al. “A review of methodologies for path planning and optimization of mobile robots”. In: *Journal of Process Management and New Technologies* 11 (Jan. 2023), pp. 122–140. DOI: 10.5937/jouproman2301122S.

[56] Xavier Olive et al. “A Framework to Evaluate Aircraft Trajectory Generation Methods”. In: Oct. 2021.

[57] Zoe Mbikayi et al. “Online deterministic 3D trajectory generation for electric vertical take-off and landing aircraft”. en. In: *Aerospace* 11.2 (Feb. 2024), p. 157.

[58] Mithun Goutham et al. “Optimal path planning through a sequence of waypoints”. In: *IEEE Robot. Autom. Lett.* 8.3 (Mar. 2023), pp. 1509–1514.

[59] Nora Schimpf et al. “A Generalized Approach to Aircraft Trajectory Prediction via Supervised Deep Learning”. In: *IEEE Access* 11 (2023), pp. 116183–116195. DOI: 10.1109/ACCESS.2023.3325053.

[60] João Pedro Morais et al. “An Introduction to Quaternions”. In: *Real Quaternionic Calculus Handbook*. Basel: Springer Basel, 2014, pp. 1–34.

[61] Yan-Bin Jia. “Quaternions and Rotations *”. In: 2015. URL: https://api.semanticscholar.org/CorpusID:7075201.

[62] Renato Zanetti. “Rotations, Transformations, Left Quaternions, Right Quaternions?” In: *Journal of the Astronautical Sciences* (Jan. 2019). DOI: 10.1007/s40295-018-00151-2.

[63] Jack B. Kuipers. “Quaternions and Rotation Sequences”. In: 1998. URL: https://api.semanticscholar.org/CorpusID:116093978.

[64] Barton J. Bacon. “Quaternion-Based Control Architecture for Determining Controllability/Maneuverability Limits”. In: 2012. URL: https://api.semanticscholar.org/CorpusID:108509985.

[65] Rick Parent. “Chapter 2 - Technical Background”. In: *Computer Animation (Third Edition)*. Ed. by Rick Parent. Third Edition. Boston: Morgan Kaufmann, 2012, pp. 33–60. DOI: https://doi.org/10.1016/B978-0-12-415842-9.00002-2. URL: https://www.sciencedirect.com/science/article/pii/B9780124158429000022.

[66] Ce Liu. “Nonsingular Hierarchical Approach for Trajectory Tracking Control of Miniature Helicopter with Model Uncertainties”. In: *Journal of Intelligent & Robotic Systems* 110 (Mar. 2024). DOI: 10.1007/s10846-024-02072-9.

[67] Dario Brescianini et al. *Nonlinear Quadrocopter Attitude Control. Technical Report*. en. Report. Zürich, 2013. DOI: 10.3929/ethz-a-009970340.

[68] James Diebel. “Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors”. In: *Matrix* 58 (Jan. 2006).

[69] Donald T Greenwood. *Principles of dynamics*. en. 2nd ed. Upper Saddle River, NJ: Pearson, July 1987.

[70] Myron Kayton et al. *Avionics Navigation Systems*. en. Ed. by Myron Kayton et al. 2nd ed. Nashville, TN: John Wiley & Sons, Apr. 1997.

[71] Gabriel Hugh Elkaim. “System Identification for Precision Control of a WingSailed GPS-Guided Catamaran”. In: (Jan. 2002).

[72] Bonnie Jonkman. “Interpolation of DCMs”. In: 2020. URL: https://openfast.readthedocs.io/en/v3.5.3/_downloads/1a893afc619c27248e5e48031ab6d8b2/DCM_Interpolation.pdf.

[73] D. M. Henderson. *Shuttle Program. Euler angles, quaternions, and transformation matrices working relationships*. Tech. rep. JSC-12960. McDonnell Douglas Tech. Services Co., Inc., July 1997.

[74] Evandro Bernardes et al. “Quaternion to Euler angles conversion: A direct, general and computationally efficient method”. In: *PLOS ONE* 17.11 (Nov. 2022). Ed. by Yaodong Gu, e0276302. DOI: 10.1371/journal.pone.0276302. URL: http://dx.doi.org/10.1371/journal.pone.0276302.

[75] Simon Pierre Barrette. *Canadian Coast Guard Bo105*. [Online; accessed February, 2025]. 2011. URL: https://commons.wikimedia.org/wiki/File:C-GCFU_(cropped).jpg.

[76] Joachim Götz. *Bo105: Configuration Data*. DLR Institut für Flugsystemtechnik. Braunschweig, Germany, 2005.

[77] Joachim Götz. *Bo105: State Space Model*. DLR Institut für Flugsystemtechnik. Braunschweig, Germany, 2005.

[78] Tom Berger. “Handling Qualities Requirements and Control Design for High-Speed Rotorcraft”. Dec. 2019.

[79] P.V. Kokotovic et al. “Singular perturbations and order reduction in control theory — An overview”. In: *Automatica* 12.2 (Mar. 1976), pp. 123–132. DOI: 10.1016/0005-1098(76)90076-5. URL: http://dx.doi.org/10.1016/0005-1098(76)90076-5.

[80] G.D. Padfield. *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling*. AIAA education series. American Institute of Aeronautics and Astronautics, 1996.

[81] Jerzy Klamka. “Controllability of dynamical systems”. eng. In: *Mathematica Applicanda* 36.50/09 (2008), null. URL: http://eudml.org/doc/293174.

[82] Walter Murray Wonham. *Linear Multivariable Control*. Springer Berlin Heidelberg, 1974. DOI: 10.1007/978-3-662-22673-5. URL: http://dx.doi.org/10.1007/978-3-662-22673-5.

[83] Latifa Abdou. “Quanser’s 3-DOF Helicopter control using LQR-I and MPC-based LQR controllers”. In: *International Conference on Computer Aided Design* (May 2023), pp. 1–6. DOI: 10.1109/ICCAD57653.2023.10152411.

[84] Jayalakshmi Myala et al. “Partially Structured LQR design for Lateral-Directional Flight Control Law”. In: (Mar. 2019). DOI: 10.1109/I-PACT44901.2019.8960152.

[85] Mruganka Kashyap et al. “Guaranteed Stability Margins for Decentralized Linear Quadratic Regulators”. In: *IEEE control systems letters* 7 (Apr. 2023), pp. 1778–1782. DOI: 10.1109/LCSYS.2023.3280868.

[86] Aamir Shahzad et al. “Computing of LQR Technique for Nonlinear System Using Local Approximation”. In: *Computer Systems: Science #38; Engineering* 46.1 (Jan. 2023), pp. 853–871. DOI: 10.32604/csse.2023.035575. URL: https://doi.org/10.32604/csse.2023.035575.

[87] P C Young et al. “An approach to the linear multivariable servomechanism problem”. In: *Int. J. Control* 15.5 (May 1972), pp. 961–979.

[88] W.S. Levine. *The Control Systems Handbook: Control System Advanced Methods, Second Edition*. The Electrical Engineering Handbook. CRC Press, 2018. URL: https://books.google.it/books?id=GVjvBQAAQBAJ.

[89] Guang-Ren Duan et al. *LMIs in control systems*. Boca Raton, FL: CRC Press, June 2013.

[90] Brian David Outram Anderson et al. *Linear Optimal Control*. en. Prentice-Hall networks series. Old Tappan, NJ: Prentice Hall, Apr. 1971.

[91] Huibert Kwakernaak et al. “Linear optimal control systems”. en. In: *J. Dyn. Syst. Meas. Control* 96.3 (Sept. 1974), pp. 373–374.

[92] Teng Fei Sun et al. “PI Parameter Tuning of Proportional Valve based on Improved PSO”. In: *Proceedings of the 7th International Conference on Cyber Security and Information Engineering* (Sept. 2022). DOI: 10.1145/3558819.3565170.

[93] Stephen Bassi Joseph et al. “Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems”. en. In: *Heliyon* 8.5 (May 2022), e09399.

[94] X.S. Yang. *Nature-Inspired Optimization Algorithms*. Academic Press, 2020. URL: https://books.google.it/books?id=La_YDwAAQBAJ.

[95] Martin Kambushev et al. “Optimizing the Selection of the Weighting Matrices Q and R in Linear Quadratic Regulator”. In: (Oct. 2019). DOI: 10.1109/HITECH48507.2019.9128263.

[96] Gene F Franklin et al. *Feedback control of dynamic systems*. en. 8th ed. Upper Saddle River, NJ: Pearson, Jan. 2018.

[97] Xia Li et al. “Model-free adaptive robust control based on TDE for robot with disturbance and input saturation”. In: *Robotica* 41 (Aug. 2023), pp. 3426–3445. DOI: 10.1017/s0263574723001078.

[98] Michael John Browne. “A unified strategy for windup prevention in control systems with multiple saturating actuators”. In: (Dec. 2000). URL: https://scholar.sun.ac.za:443/handle/10019.1/51629.

[99] Federal Aviation Administration (FAA). *Helicopter flying handbook*. en. New York, NY: Skyhorse Publishing, July 2021.

[100] John T Betts. “Survey of numerical methods for trajectory optimization”. en. In: *J. Guid. Control Dyn.* 21.2 (Mar. 1998), pp. 193–207.

[101] Xuhao Gui et al. “Improved data-driven trajectory optimization method utilizing deep trajectory generation”. en. In: *J. Aerosp. Comput. Inf. Commun.* 22.1 (Jan. 2025), pp. 28–42.

[102] Douglas Thomson et al. “Mathematical Definition of Helicopter Maneuvers”. In: *Journal of the American Helicopter Society* 42 (Oct. 1997), p. 307. DOI: 10.4050/JAHS.42.307.

[103] R.W. Prouty. *Helicopter Performance, Stability, and Control*. Krieger, 2005.

[104] Yihua Cao et al. “Mathematical modeling of helicopter aerobatic maneuvers”. en. In: *Aircr. Eng.* 76.2 (Apr. 2004), pp. 170–178.

[105] Robert Eisele. *Quaternion from two vectors*. 2022. URL: https://raw.org/proof/quaternion-from-two-vectors/.

[106] C. L. Blanken et al. *PROPOSED REVISIONS TO AERONAUTICAL DESIGN STANDARD – 33E (ADS-33E PRF) TOWARD ADS-33F-PRF*. Tech. rep. SR-FCDD-AMV-19-01. U.S. Army, Sept. 2019.

[107] George M. Yamakawa et al. *Utility Tactical Transport Aircraft System (UTTAS) Maneuver Criteria*. Tech. rep. AD0902767. ARMY AVIATION SYSTEMS TEST ACTIVITY EDWARDS AFB CA, Nov. 1972.

[108] Douglas Thomson. “An Analytical Method of Quantifying Helicopter Agility”. In: Sept. 1986.

[109] Shaoming He et al. “Optimal Guidance for Integrated Waypoint Following and Obstacle Avoidance”. In: 2019 (Nov. 2019), pp. 325–334. DOI: 10.1109/REDUAS47371.2019.8999721. URL: https://jglobal.jst.go.jp/en/detail?JGLOBAL_ID=202002266935532777.

[110] Zongchuan Zhou et al. “Efficient and Robust Time-Optimal Trajectory Planning and Control for Agile Quadrotor Flight”. In: *IEEE robotics and automation letters* 8.12 (Dec. 2023), pp. 7913–7920. DOI: 10.1109/lra.2023.3322075.

[111] Hanno Ackermann et al. “Trajectory reconstruction for affine structure-from-motion by global and local constraints”. In: *Computer Vision and Pattern Recognition* (June 2009), pp. 2890–2897. DOI: 10.1109/CVPR.2009.5206664. URL: https://www.tnt.uni-hannover.de/papers/data/768/768_1.pdf.

[112] Z. Zhang et al. “G2LTraj: A Global-to-Local Generation Approach for Trajectory Prediction”. In: (Apr. 2024). DOI: 10.48550/arxiv.2404.19330. URL: https://arxiv.org/pdf/2404.19330.

[113] Sergej Vital’evich Znamenskij. “Local competing in interpolation problems”. In: 11.4 (Dec. 2020), pp. 99–122. DOI: 10.25209/2079-3316-2020-11-4-99-122. URL: https://cyberleninka.ru/article/n/local-competing-in-interpolation-problems.

[114] Helmuth Späth. *One dimensional spline interpolation algorithms*. en. London, England: CRC Press, Dec. 2019.

[115] Jongho Park et al. “Reactive Trajectory Generation of Unmanned Aerial Vehicle Incorporating Fuzzy C-Means Clustering and Optimization Problem-based Guidance”. In: *IEEE Access* (Jan. 2024). DOI: 10.1109/access.2024.3510736.

[116] Baris Baspinar. “A Generic Model of Nonlinear Helicopter Dynamics for Intelligent Avionics Systems and Trajectory Based Applications”. In: *AIAA SCITECH 2022 Forum* (Jan. 2022). DOI: 10.2514/6.2022-2100.

[117] Binwu Ren et al. “High-precision trajectory tracking control of helicopter based on ant colony optimization-slime mould algorithm”. In: *Chinese Journal of Aeronautics* 38 (Aug. 2024). DOI: 10.1016/j.cja.2024.08.003.

[118] Ugo Coppa et al. “Accuracy enhancement of unmanned helicopter positioning with low-cost system”. In: *Applied Geomatics* 1 (Sept. 2009). DOI: 10.1007/s12518-009-0009-x.

[119] Emil F. Weiland. “Development and Test of the BO 105 Rigid Rotor Helicopter”. In: *Journal of the American Helicopter Society* 14.1 (Jan. 1969), pp. 22–37. DOI: 10.4050/jahs.14.22. URL: http://dx.doi.org/10.4050/JAHS.14.22.

[120] Rotor Leasing. *BO-105 - Helicopter Specifications*. https://rotorleasing.com/Specifications/BO105.html. [Accessed 20-10-2024].

[121] M. Kerler et al. “Modeling of BO 105 flight dynamics for research on fuel savings due to single-engine operation”. English. In: *38th European Rotorcraft Forum 2012, ERF 2012*. 38th European Rotorcraft Forum 2012, ERF 2012. 38th European Rotorcraft Forum 2012, ERF 2012 ; Conference date: 04-09-2012 Through 07-09-2012. 2012, pp. 659–670.

[122] Murat Şenipek. “DEVELOPMENT OF AN OBJECT-ORIENTED DESIGN, ANALYSIS AND SIMULATION SOFTWARE FOR A GENERIC AIR VEHICLE”. Sept. 2017. DOI: 10.13140/RG.2.2.24882.15042.

[123] N. Moll. “To a Bo105 pilot: this thing’s pretty”. In: *Flying magazine* 118.11 (1991), pp. 96–105.

[124] G.D. Padfield. *A Theoretical Model of Helicopter Flight Mechanics for Application to Piloted Simulation*. Technical Report RAE TR 81048. Royal Aircraft Establishment, Bedford, UK, 1981.

[125] G.D. Padfield. *Theoretical modelling for helicopters flight dynamics: development and validation*. Tech. rep. 16th ICAS congress. Royal Aircraft Establishment, Bedford, UK, 1988.

A

Description of the Bo105 helicopter

The appendix serves as an addition to the analysis presented in Chapter 6, providing a modal and structural analysis to contextualize the choice of the Bo105 helicopter for agile flight. In particular, this appendix will be divided in two major sections.

To start, in Section A.1 a complete description of the Bo105 helicopter is provided, overviewing its structure and its most relevant peculiarities. In particular, the aircraft will be presented in terms of its rotor system, its fuselage and engines. This discussion serves to justify the interest in using the Bo105 helicopter in the context of agile flight, while presenting its most relevant mechanical characteristics and highlighting their contribution to the overall performance of the helicopter.

Further expanding upon the modal analysis carried out in Chapter 6, in Section A.2, a comparison between the system-identified model and a theoretical model obtained by Padfield via the Helisim modelling approach [80] is provided. This analysis is instrumental in highlighting the complexity of the Bo105 helicopter in its available linearizations and emphasises the intricacy of the existing couplings.

A.1. The Bo105 helicopter

The Messerschmitt-Bölkow-Blohm (MBB) Bo105 is a lightweight, twin-engine, multi-role helicopter that has seen extensive use in both civilian and military applications. Developed in Germany and first flown in 1967, the Bo105 was on of the first helicopters to feature a hingeless rotor system, which allowed for highly agile manoeuvrability and precise control. The Bo105 was considered revolutionary for its time and it was a result of an extensive research process that aimed at obtaining a versatile and highly manoeuvrable helicopter with low maintenance costs and high power reserves [119].

To provide an overall understanding of the fundamental mechanical properties of the Bo105, the most relevant characteristics of the vehicle will be briefly described hereafter.

A.1.1. General characteristics

The main characteristics of the Bo105 are summarised in Table A.1, while a three-ways diagram of the helicopter is given in Figure A.1.

From the data in Table A.1, it can be noticed that the Bo105 fall comfortably in the light-weight helicopter class, given its weight and its 9.8 m rotor radius. Even so, the twin-engine design allows for a significant degree of available power and safety, as in case of engine loss the helicopter can still be controlled even with only one of the two engines, albeit in a reduced speed range [121]. Moreover, while the twin engine design allows for a remarkable available power and top speed for its weight class, the relative high efficiency of the two Allison 250 C20 engines helicopter allows the Bo105 to maintain a good maximum range.

A.1.2. Rotor system

The Bo105's rotor system is one of its most defining features. The main rotor consists of four composite blades, arranged on a rigid rotor head without traditional hinges and dampers: this rigid configuration (also referred to as "hingeless") was especially revolutionary at the time and its introduction was allowed by

Table A.1: Bo105 configuration data

Description	Symbol	Value	Unit
Gross weight	m_g	2200	kg
Helicopter moments of inertia (reference point: CG_H)	I_{xx}	1433	[kgm^2]
	I_{yy}	4973	[kgm^2]
	I_{zz}	4099	[kgm^2]
	I_{xz}	660	[kgm^2]
Main Rotor (MR)			
Radius	R_{MR}	4.912	[m]
Angular velocity	Ω_{MR}	44.4	[rad/s]
Blade number	N_{MR}	4	[-]
Blade chord	c_{MR}	0.27	[m]
Blade mass	$m_{Bl,MR}$	24.2	[kg]
Tail Rotor (TR)			
Radius	R_{TR}	0.95	[m]
Angular velocity	Ω_{TR}	233	[rad/s]
Blade number	N_{TR}	2	[-]
Blade chord	c_{TR}	0.179	[m]
Blade mass	$m_{Bl,TR}$	0.47	[kg]
Engine data			
Engine	Allison 250 C20		
Number of engines	N_{en}	2	[-]
Horsepower	P_{en}	420	[shp]
Normal range	R_{en}	300	[nm]
Max range	R_{en}^{max}	550	[nm]

advancements in the field of material studies. The design of a rigid rotor is shown in Figure A.2: as shown, the rigid rotor design has no flapping hinges and no lead-lag hinges, with the only degree of freedom of the blades being provided by their pitch horn to allow for feathering.

Compared to articulated and semi-rigid rotor systems, the hingeless configuration treats the blades and the main rotor mast as a single structural element, which allows for an increase in responsiveness and a reduction in control inputs [99]. In addition to this, another significant characteristic of the hingeless rotor configuration is the reduction in maintenance costs, which is a direct consequence of the simplified design of the rotor. While the hingeless rotor structure boasts many advantages in responsiveness over the articulated design, it is worth noting that rigid rotors typically transfer more vibrations to the cabin (as the rotor blades are directly connected to the main rotor mast) and the removal of the lead-lag and flapping hinges may reduce the manoeuvrability in heavily gusty environments, as external forces caused by wind are transferred rigidly to the body of the vehicle.

Because of these characteristics - while there may be degradation in flight quality in especially turbulent air - the hingeless rotor allows the Bo105 to achieve high manoeuvrability and a crisp response in tactical and emergency operations, while also having an improved reliability due to the reduce maintenance effort, thus justifying its choice over other rotor types for the development of the Bo105 helicopter.

A.1.3. Fuselage and engine

One of the critical design objectives of the Bo105 helicopter was its flexibility, as it was intended to be used for both military and civil purposes. For this reason, in addition to the considerations made for the design of the rigid rotor system, the design also involved serious considerations about the fuselage geometry and the engine. In an effort to provide a proper description of the Bo105 helicopter, these considerations will

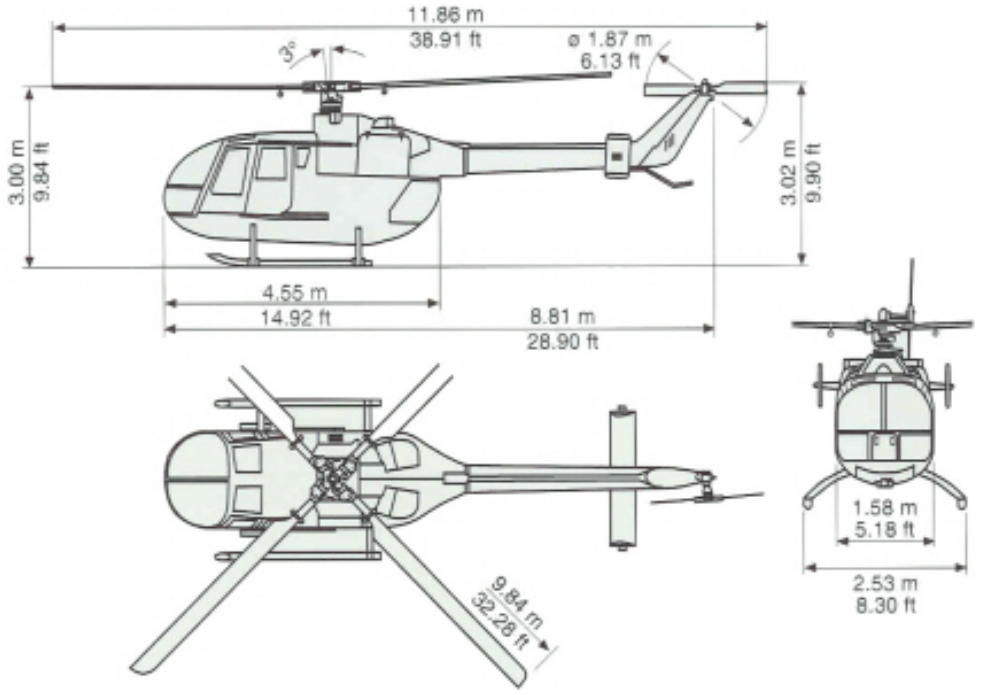


Figure A.1: Bo105 three-view drawing [120]

be briefly discussed hereafter.

While the rigid rotor design allowed for high manoeuvrability and a reduction in control effort, the generation of sufficient power and its efficient transmission to the main and tail rotors was a critical matter. To this end, a twin-engine design was chosen, with two turbines providing power to the main rotor.

The twin-engine configuration is characterised by both advantages and disadvantages. On one hand, the twin engine design allows for an increase in available power and an additional degree of redundancy: allowing for a direct increase in maximum take-off weight (MTOW) and increasing safety (as in case of engine failure, the other engine will still provide the required power to control the flight and allow for a controlled emergency landing [121]). On the other hand, the twin engine design causes an increase in weight and complexity, while also causing both engines to function at partial load, which reduces the engine efficiency and increases fuel consumption. Even with these considerations, the increased power and safety of the twin-engine configuration outweighed its limitation.

Regarding its structural components, the Bo105 helicopter is optimized for durability, weight efficiency, and adaptability across various operational contexts, making it a remarkably usable aircraft and meeting its multiple-use-scenario requirements.

The fuselage is constructed primarily from riveted aluminium panels and sandwich panels, providing robust structural support while minimizing weight and allowing for the capacity of transporting up to 3 passengers (in addition to the two pilots) [119]. For landing gear, the Bo105 uses a durable twin-skid configuration. This simple design enhances the helicopter's resilience in rough or uneven terrain, making it suitable for operations in remote areas. The skids provide stable support for landing on both hard and soft surfaces, while the compact configuration minimizes weight, contributing to the helicopter's overall manoeuvrability.

The introduction of a twin-engine design and the choice of lightweight materials for the main structure of the helicopter contribute greatly to the overall manoeuvrability of the helicopter itself, allowing for the responsiveness of the hingeless rotor design to be fully leveraged.

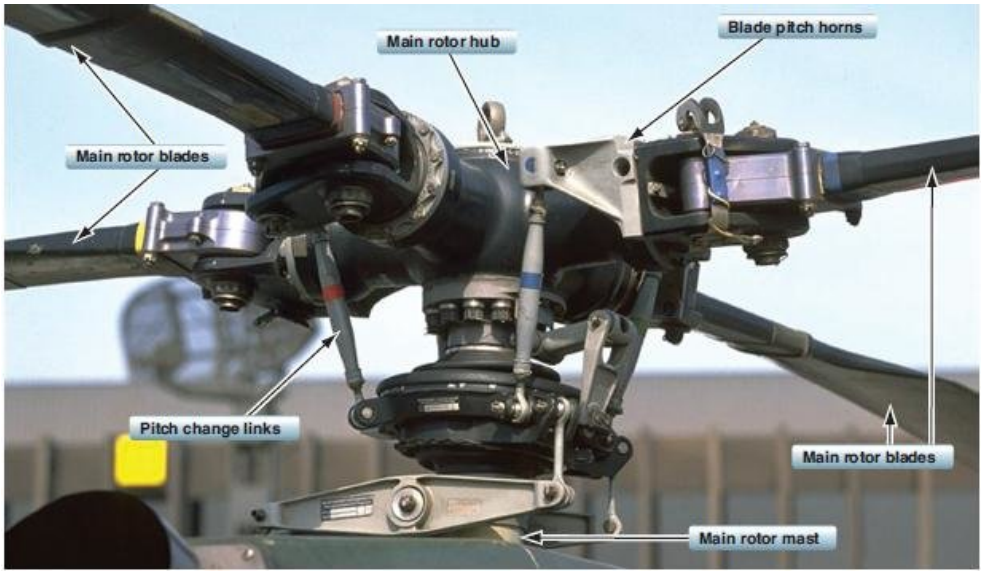


Figure A.2: Four-bladed hingeless main rotor system [122]

A.1.4. Concluding remarks

Here the characteristics of the Bo105 helicopter have been discussed, with particular attention given to the design of the hingeless rotor. The analysis performed highlighted the advantages of this rigid configuration over the articulated and semi-rigid designs and justified other significant choices, such as the twin-engine design and the lightweight structure to allow for a complete expression of the high control power and manoeuvrability of the helicopter.

The high available power and the responsiveness of the hingeless rotor configuration allow the safe execution of quick and agile manoeuvres which would not be practicable by other similar helicopters. Thanks to its characteristics, the Bo105 can execute rapid ”nap-of-the-earth” manoeuvres (very low altitude flight, commonly used in military applications to make use of cover and avoid detection) which would be dangerous or impossible for teetering and articulated rotors due to potential loss of controllability [123], and the helicopter was even capable of ”full aerobatic routines, complete with loops, hammerheads, bunts and rolls” [123, p. 101].

With the characteristics described above, the choice of the Bo105 helicopter for the purpose of this thesis is justified from a mechanical standpoint, as its design characteristics make it a lightweight helicopter capable of high manoeuvrability and even aerobatics manoeuvres. Because of this, the introduction of a quaternion-based control system is of interest, as it would allow to overcome the limitations of the Euler angles parametrization, allowing for the execution of autonomous aerobatic manoeuvres and making rapid changes in attitude possible, thus fully expressing the high-manoeuvrability potential of the Bo105.

A.2. Comparison between system-identified and theoretical modes

The mode analysis of the linearized systems produced in Section 6.2 highlighted the peculiarity of the available identified models, with eigenvector elements showing high cross-activations and unconventional dynamics. To emphasise the unconventional nature of the identified dynamics of the available systems, a comparison between the identified model and a theoretical model is appropriate.

One of the most prominent helicopter model structures used in research is Helisim, a generic model proposed in [124] by Padfield: this model serves as an evolution of early helicopter models and was developed to allow for simulation of both ”gentle” and ”agile flight” [125]. In particular, the Helisim model has also been applied to the Bo105 helicopter in [80], obtaining a theoretical pole map of the rigid-body system with theoretical parameters: a graphical representation of the theoretically identified pole loci for the Bo105 helicopter is given in Figure A.3, where the approximate pole locations are shown over the speed range from hover to 160kn.

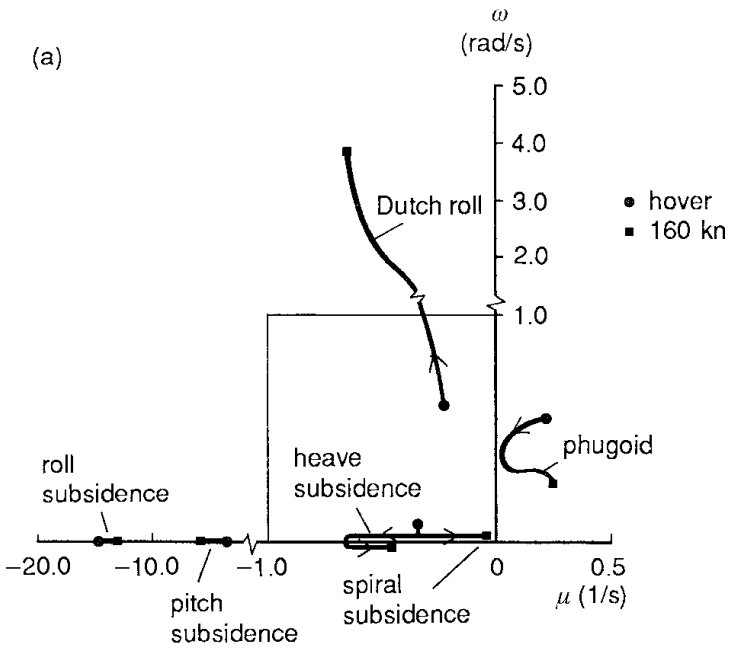


Figure A.3: Theoretical pole map of the Bo105 helicopter [80]

The modes identified are aligned with the typical behaviours expected from a helicopter. Two oscillatory modes are found, a phugoid and a Dutch roll: the phugoid mode is an unstable longitudinal mode describing an exchange in airspeed and altitude; the Dutch roll is a more highly damped lateral mode which involves oscillations in the yaw and roll angles. Two non-oscillatory modes are also present, the roll and pitch subsidence, which relate respectively to the lateral and to the longitudinal dynamics of the system. An additional mode emerges in the coupled heave and spiral subsidence modes, which are coupled at lowed speeds and become decoupled as velocity increases: this behaviour is also to be expected from helicopters, as hover is a much more complex flying condition and couplings are more prevalent.

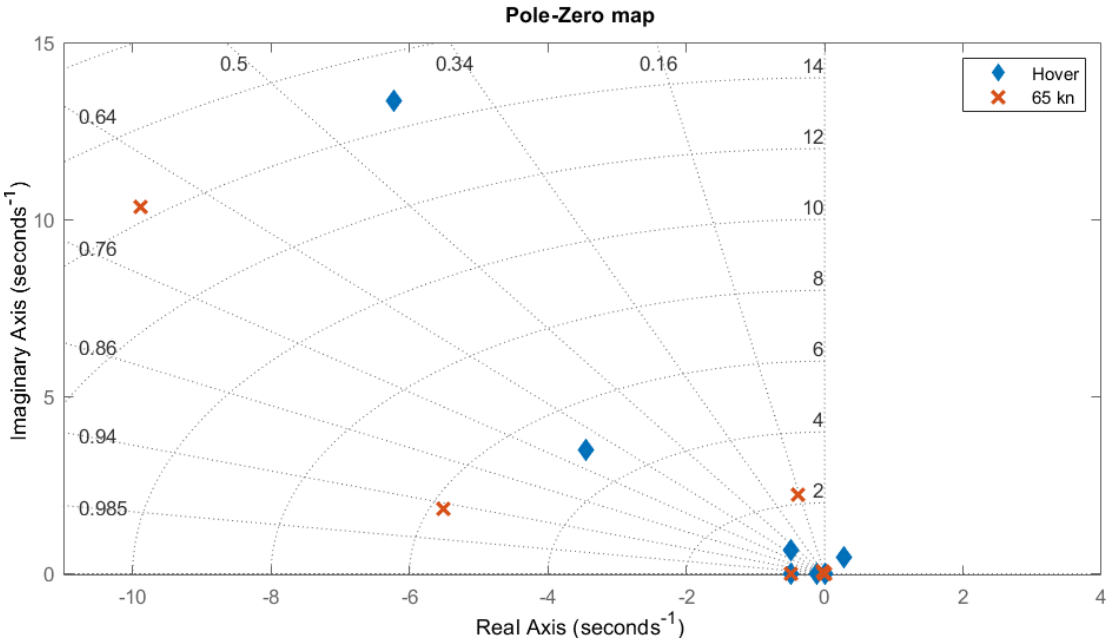


Figure A.4: Pole map of the Bo105 model

Comparing the theoretical pole loci identified in Figure A.3 with the pole loci of the available linearized model in Figure A.4, it is apparent that the two images show noticeable differences.

The first noticeable discrepancy between the modes is the presence of two additional high-frequency complex pole pairs, both in the hover and in the 65kn knots linearizations: from the mode analysis in Section 6.2.1 it is apparent that the eigenvectors associated with these poles have large contributions from the $\dot{p}$ and $\dot{q}$ states, which have very fast dynamics and are not modelled as states in the theoretical Helisim model.

As a further point of difference, the identified model shows more noticeable couplings between the modes. This is expected, as theoretical model are based on fundamental simplifying assumptions and are unable to account for uncertainties in the parameters used when computing the models. These limitations prevent the theoretical Helisim model from capturing the complete dynamics of the helicopter and highlight the added precision granted by system identification techniques.

To provide a more immediate comparison between the identified poles and the theoretical ones, Figure A.5 shows the pole loci for both systems in hover, with a more detailed view of the map provided in Figure A.6. By comparing the two systems, it is noticed that phugoid and Dutch roll modes appear to have been correctly identified, with the pole location being highly similar to the expected modes. Even so, the eigenvector analysis carried out in Section 6.2.1 suggests that the poles associated with the phugoid and the Dutch roll show a high cross-activation between the longitudinal and lateral states, which appear to not have been taken into account in the theoretical Helisim model. Conversely, the Helisim model in hover suggests a coupling between the heave and spiral subsidence modes, which is not detected in the identified hover linearization.

These discrepancies highlight the limitations of the Helisim theoretical model, which were also noted by Padfield in [125], where notable modelling deficiencies were noted with validation data in agile flight cases, although the general parametric trends appeared to be correct. Additionally, these discrepancies become even more relevant in the context of agile flight for the specific Bo105 helicopter, as the Bo105 was developed with the fundamental objective of creating a fast and highly maneuverable vehicle and, as such, is more prone to quick cross-coupled responses.

Overall, the study of the model highlighted the complicated nature of the system's dynamic responses, with noticeable unusual couplings in the longitudinal and lateral channels. While these cross-couplings are expected in helicopters due to the large values assumed by the aerodynamic derivatives [80], theoretical models are often unable to provide a complete modelling of these conditions due to their fundamental underlying simplifications and due to the physical complexity of helicopter flight.

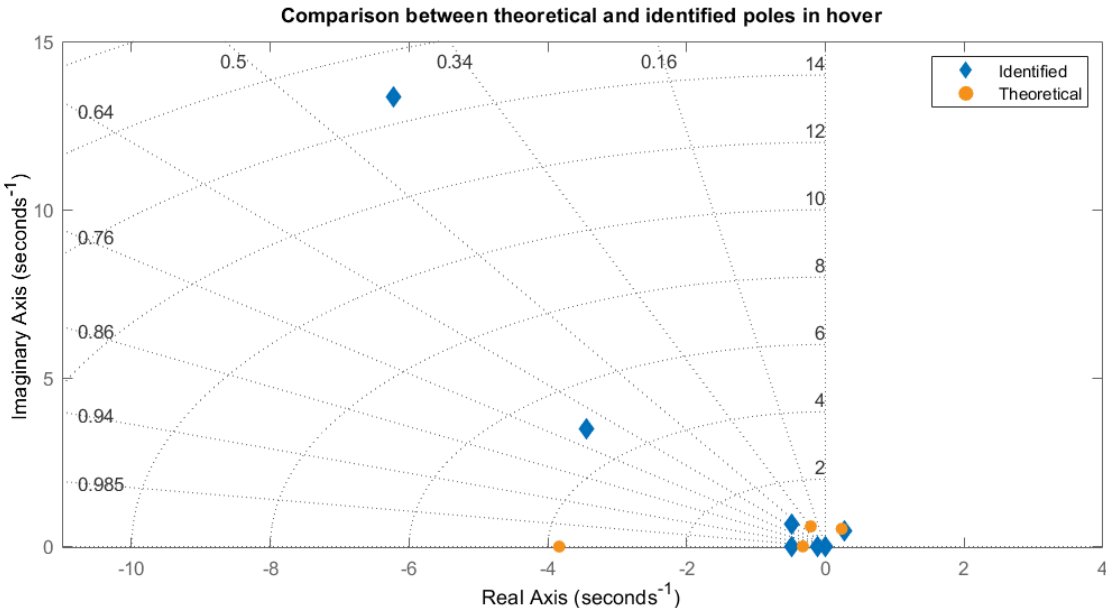


Figure A.5: Comparison between theoretical and system-identified pole loci in hover

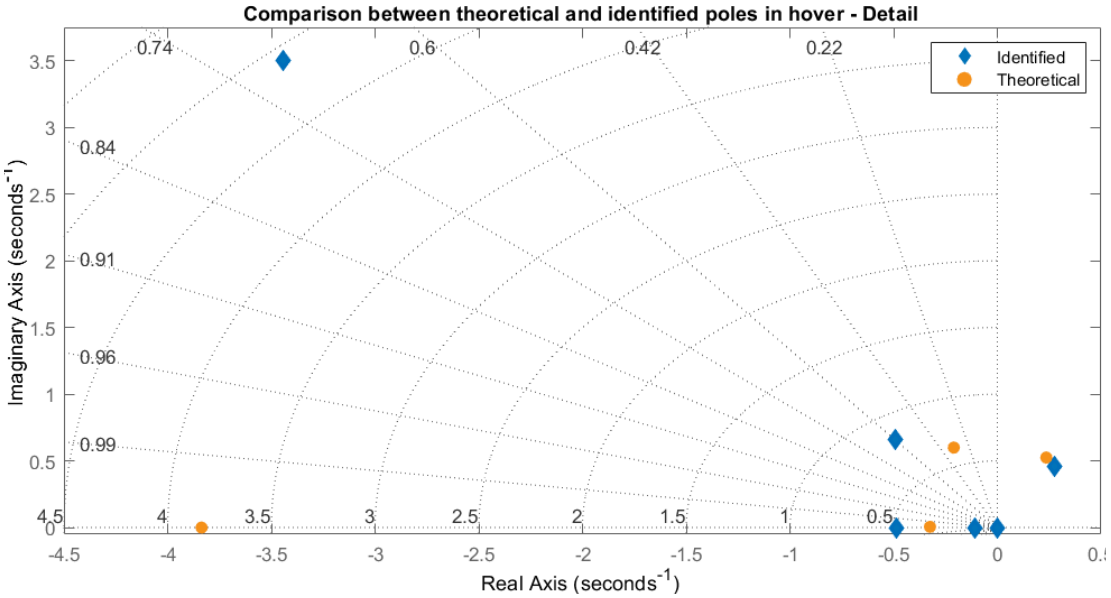


Figure A.6: Comparison between theoretical and system-identified pole loci in hover (detail)

B

Results of the quaternion conversion

Here the results of the time simulations discussed in Section 7.2.2 will be shown in their entirety. The figures presented aim at showing the overlap between the time responses of the Euler angles system SS_{eul} and the quaternion system SS_{quat}. The figures will be grouped by input, showing both the results for the hover linearization and those for the 65 knots linearization.

As discussed, the responses of the two systems exactly overlap, suggesting that the conversion has been successful in introducing quaternions as an internal parametrization for attitude. This, in addition to the comparisons performed about the system's eigenvalues and eigenvectors characteristics, suggests that the conversion logic used in this report is valid and maintains the system's fundamental dynamics.

For each simulation, the selected input is defined as a square pulse starting at $T = 2$s and lasting for 6s. The amplitude of the pulse is 10% for all inputs, meaning that the inputs are commanded to provide a deflection of a positive 10% from their trim position at either the hover trim point or at the 65kn trim point.

In order to properly test the responses of the system in all the most representative cases, here the results of four simulations will be presented, where the square pulse reference signal defined above is applied once on each of the available inputs. As such, the pulse input will be applied: in Section B.1 on the longitudinal cyclic δ_x, in Section B.2 on the lateral cyclic δ_y, in Section B.3 on the collective δ_0, and in Section B.4 on the pedal δ_p. In all the presented scenarios, all other inputs are left inactive.

B.1. Longitudinal cyclic δ_x input

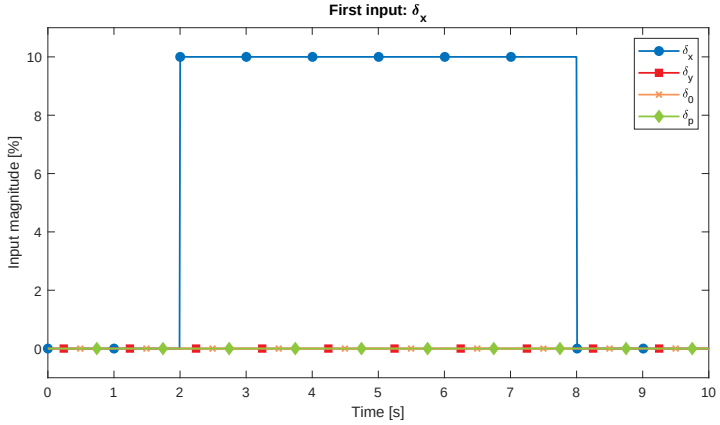


Figure B.1: Example input activating the δ_x channel

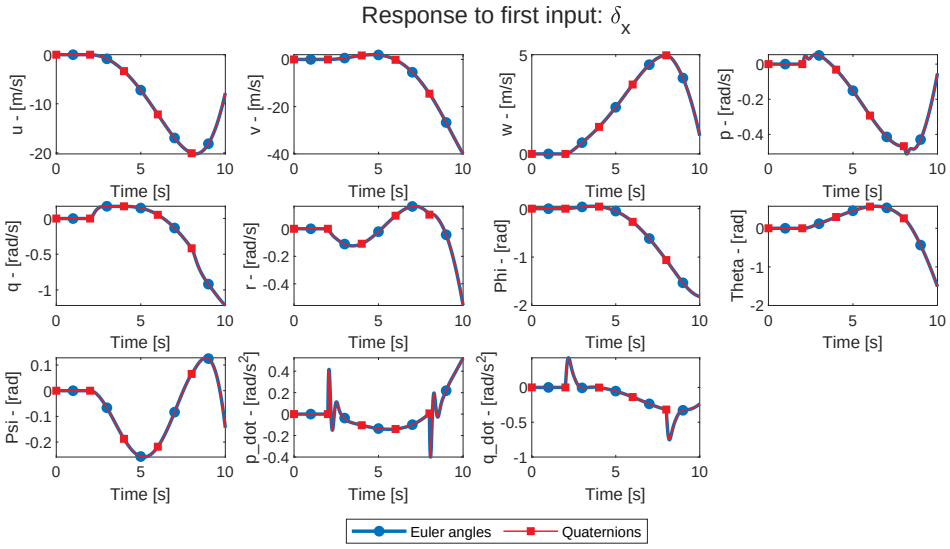


Figure B.2: Time response of the hover state-space: input applied on δ_x

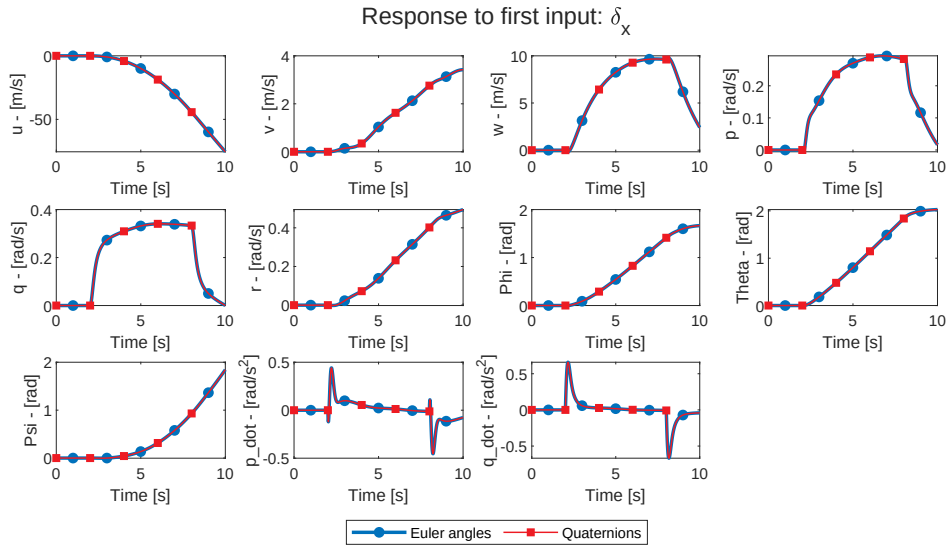


Figure B.3: Time response of the 65 kn state-space: input applied on δ_x

B.2. Lateral cyclic δ_y input

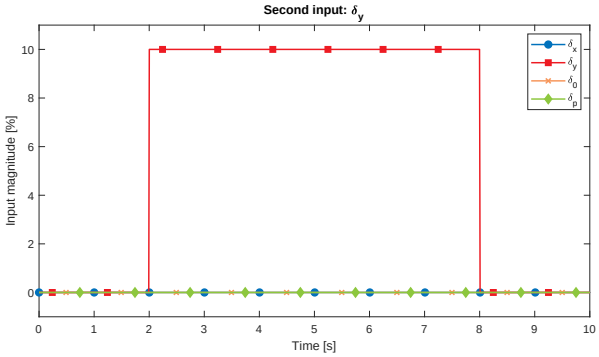


Figure B.4: Input activating the δ_y channel

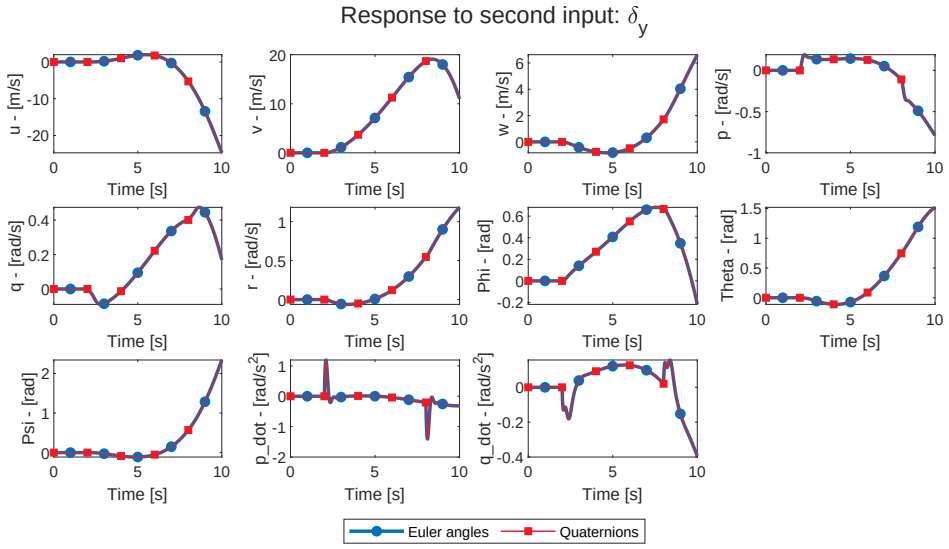


Figure B.5: Time response of the hover state-space: input applied on δ_y

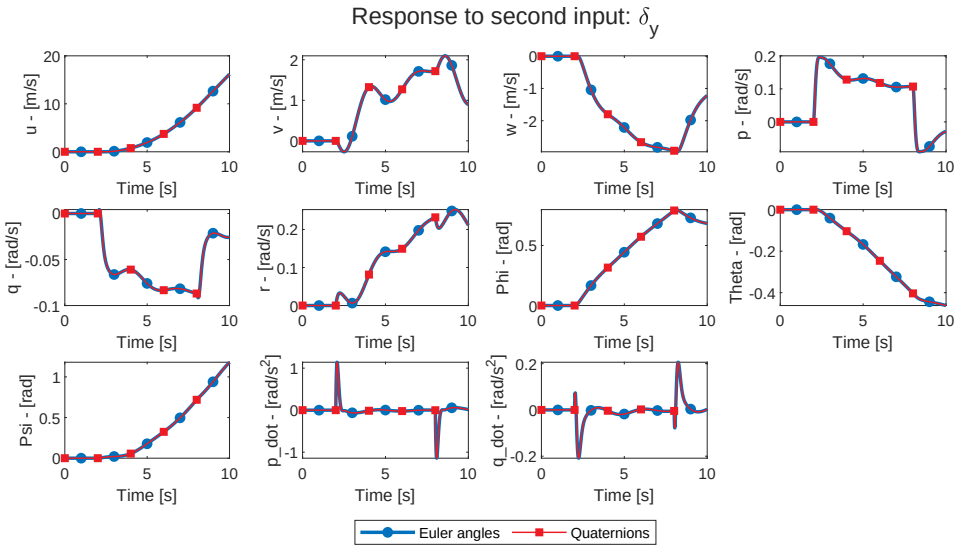


Figure B.6: Time response of the 65 kn state-space: input applied on δ_y

B.3. Collective δ_0 input

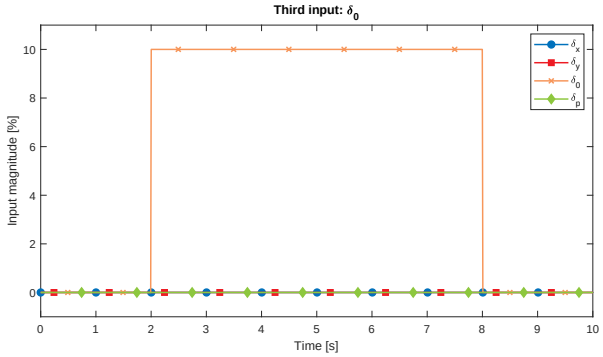


Figure B.7: Input activating the δ_0 channel

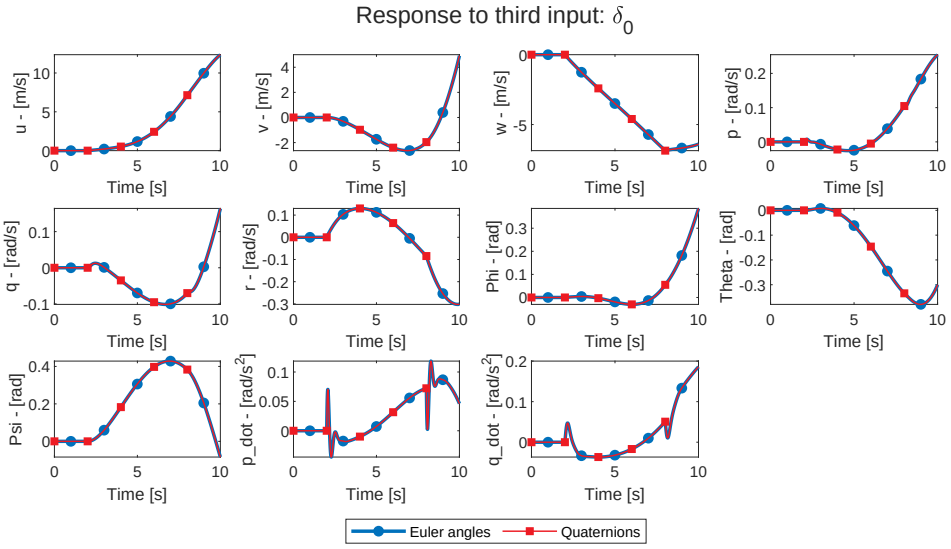


Figure B.8: Time response of the hover state-space: input applied on δ_0

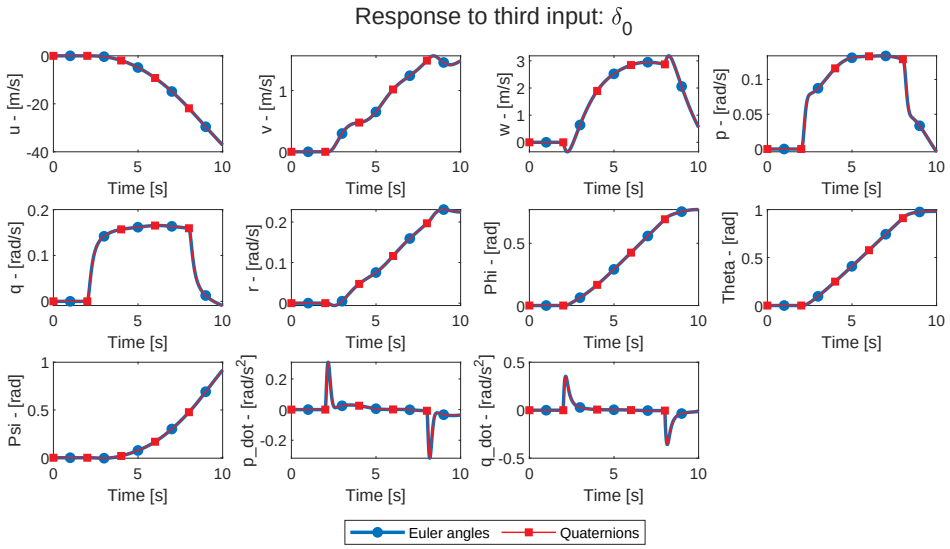


Figure B.9: Time response of the 65 kn state-space: input applied on δ_0

B.4. Pedal δ_p input

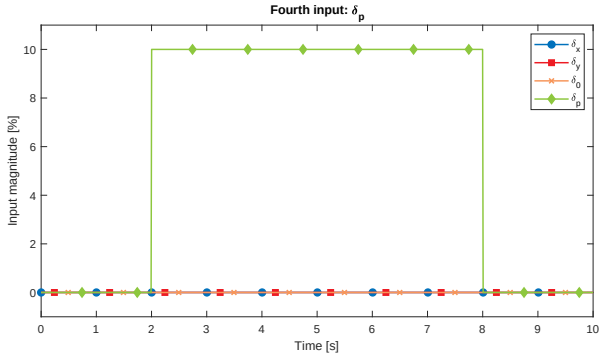


Figure B.10: Input activating the δ_p channel

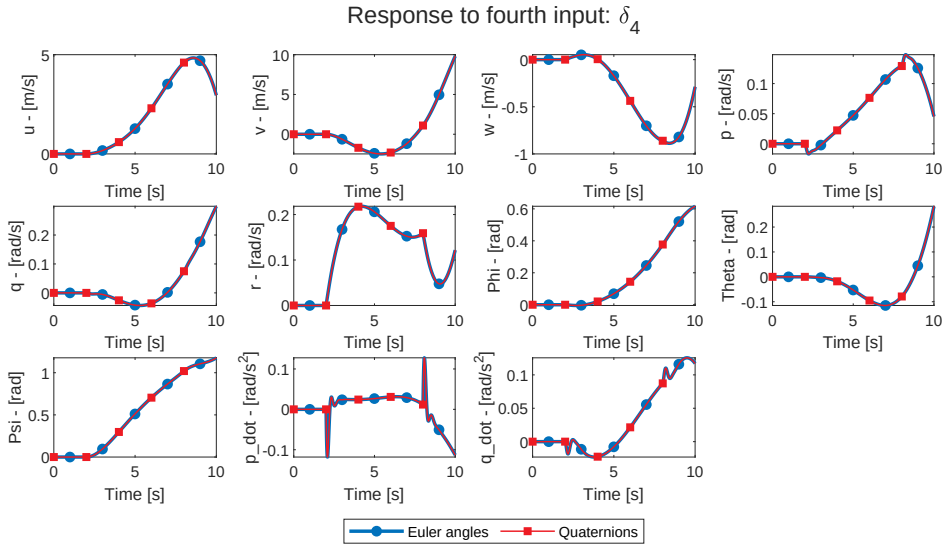


Figure B.11: Time response of the hover state-space: input applied on δ_p

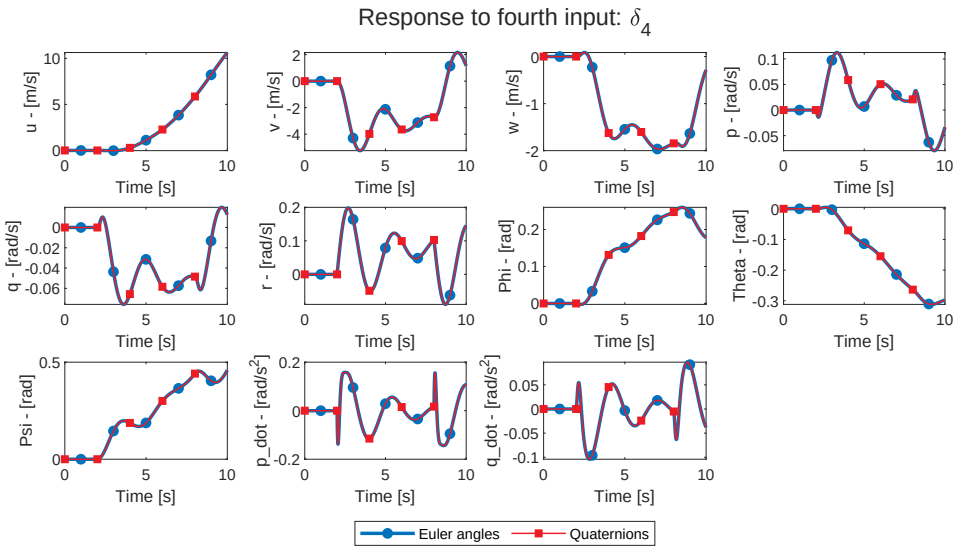
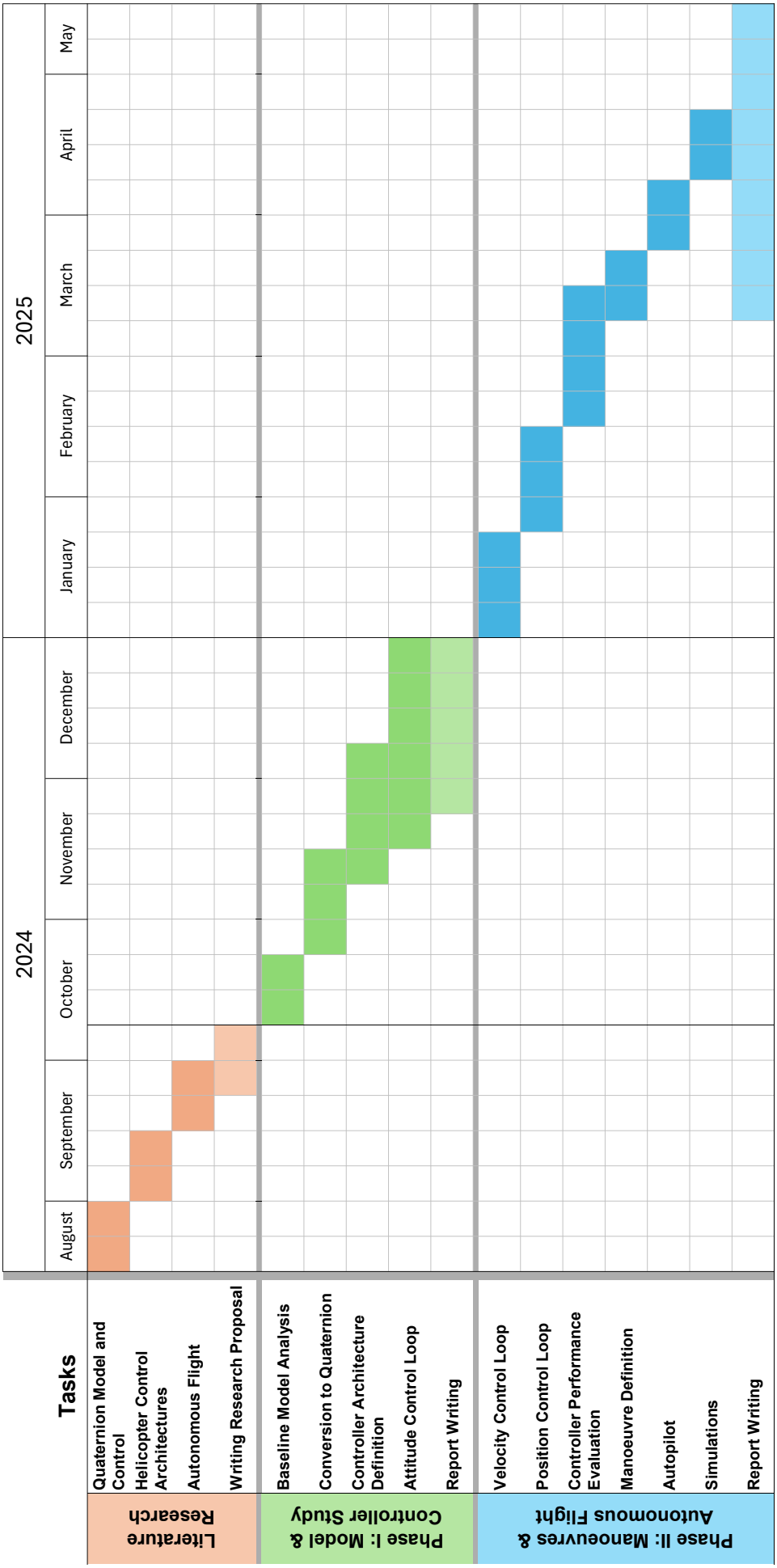


Figure B.12: Time response of the 65 kn state-space: input applied on δ_p

C

Gantt Chart


2024
2025
August
September
October
November
December
January
February
March
April
May
Tasks
Literature Research
Quaternion Model and Control
Helicopter Control Architectures
Autonomous Flight
Writing Research Proposal
Phase I: Model & Controller Study
Baseline Model Analysis
Conversion to Quaternion
Controller Architecture Definition
Attitude Control Loop
Report Writing
Phase II: Manoeuvres & Autonomous Flight
Velocity Control Loop
Position Control Loop
Controller Performance Evaluation
Manoeuvre Definition
Autopilot
Simulations
Report Writing