Trajectory Optimization of a Small Airship

by

Charles Blouin

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Mechanical Engineering

Department of Mechanical Engineering
Faculty of Engineering
University of Ottawa

Abstract

Pseudo-spectral optimal solvers are used to optimize numerically a performance index of a dynamical system with differential constraints. Although these solvers are commonly used for space vehicles and space launchers for trajectory optimization, few experimental papers exist on optimal control of small airships. The objective of this thesis is to evaluate the use of a pseudo-spectral optimal control solver for generating dynamically constrained, minimal time trajectories. A dynamical model of a small airship is presented, with its experimental virtual mass, drag and motor experimentally modeled. The problems are solved in PSOPT, a pseudo-spectral optimal control code. Experimental tests with a small scale model are performed to evaluate the generated paths. Although drift occurs, as a consequence of an open loop control, the vehicle is capable of following the path. This results of this thesis may find uses in verifying how close to optimal discreet path planners are, to plan complex trajectories on short distances, or to generate dynamic maneuverer such as take-off or landing. Ultimately, improving path planning of small airships will improve their safety, maneuverability and flight-time, which makes them fit for scientific monitoring, for search and rescue, or as mobile telecommunications platforms.

Acknowledgements

I would like to acknowledge the immense help from the staff at the University of Ottawa. In particular, I would like to thank my supervisors, Dr. Lanteigne, for his advice and support in writing this thesis and my other technical projects, and Dr. Gueaieb, for his comments and ideas. I also would like to thank Steven Reckoskie for his help and for the useful discussions we had on airships.

In addition, I would like to thank the members of my thesis committee, Dr. Spinello and Dr. Ahmadi.

Je suis arrivé ici grâce à l'aide de mes parents. Finalement, j'ai une pensée spéciale pour Angelle, ma partenaire, pour son support, son amour et ses encouragements.

Dedication

À Angelle.

Table of Contents

List of Tables vi

List of Figures vii

Nomenclature ix

1 Introduction 1

1.1 Motivation . . . 1
1.2 Problem description . . . 2
1.3 Thesis objective and contribution . . . 3
1.4 Thesis layout . . . 4
1.5 Literature review . . . 4
1.5.1 Methods of path planning . . . 4
1.5.2 Uses of optimal trajectories for flying vehicles . . . 6

2 Modeling and theory 9

2.1 Modeling of the airship . . . 9
2.1.1 Modeling of airships in the literature . . . 9
2.1.2 Modeling of a small airship . . . 10
2.2 Kinematic equations . . . 11
2.3 Kinetic equations . . . 14
2.3.1 Drag Model . . . 16
2.3.2 Thrust model . . . 18
2.4 Simulation method . . . 23
2.4.1 Description of the general optimal control problem . . . 23
2.4.2 Multiple phase problems . . . 25
2.4.3 Description of the solver used and its methods . . . 26

3 Experimental platform **32**
3.1 Tracking software and pose estimator 32
3.2 Data Transmission 34

4 Experimental tests and simulations **36**
4.1 Problem 1 36
4.1.1 Problem description 36
4.1.2 Simulation results 37
4.1.3 Experimental results 38
4.1.4 Analysis of the results 41
4.2 Problem 2 41
4.2.1 Problem description 41
4.2.2 Simulation results 42
4.2.3 Experimental results and discussion 42
4.3 Advanced simulations 46
4.3.1 Problem 3 46
4.3.2 Problem 4 46

5 Discussion of the simulations and experimental tests **50**
5.1 Error analysis 50
5.2 Limitations 51
5.3 Comparison of simulation and experimental results 52
5.4 Solving time 52
5.5 Optimal control solver for airships 53

6 Conclusion **54**
6.1 Key findings 54
6.2 Future work 54

References **56**

APPENDICES **60**

A Software Libraries **61**

List of Tables

2.1 Important dimensions of the airship 13
2.2 Experimental value characterizing the airship 22

4.1 Simulation parameter for the first experiment. 37
4.2 Experimental results of the linear test 41
4.3 Simulation parameter for the rotation experiment. 42
4.4 Experimental results of the rotation test 43
4.5 Simulation parameter for the third experiment 46
4.6 Simulation parameters for the fourth experiment 48

List of Figures

1.1 A possible path in a 2D problem 3
1.2 Discretized path in a 2D problem 5
1.3 Possible solutions to the brachistochrone problem. 6

2.1 Angle convention 9
2.2 Airship during a minimum time linear displacement test. . . . 11
2.3 Close-up of the nacelle 12
2.4 Airship dimensions 12
2.5 Relative velocities of the airship 16
2.6 Characterization of a drop test 18
2.7 Thrust model of the airship 19
2.8 Sample of the analyzed spectrum of the sound emitted by the motor. 20
2.9 Frequency spectrum of a sample of the motor sound at full power. 20
2.10 Normalized response of a thruster during simulated and experimental test 21
2.11 Multi-phase example in an orbital change problem 25
2.12 The function $\sin \Pi$, approximation with Lagrange polynomial and linear interpolation 27
2.13 The first four Legendre polynomials 28
2.14 Simplified visual representation of the problem. 31

3.1 Tracking marker used on the airship. 33
3.2 Camera used for tracking 33
3.3 Airship during a minimum time linear displacement test. . . . 33
3.4 Transmitter used for the experiments. 34
3.5 Experimental process, from the generation of the trajectory to the airship. 35

4.1 Simulated thrust and its position for the first problem. 37
4.2 Simulated command sent to the motors. 38
4.3 Simulated orientation as a function of time. 39
4.4 Simulated and experimental trajectories as a function of time for the linear acceleration test. 39
4.5 Experimental orientations as a function of time for the linear acceleration test. 40
4.6 Simulated and experimental velocities as a function of time for the linear acceleration test. 40
4.7 Simulated thrust and its position for the reorientation problem. 42
4.8 Simulated command sent to the motor for the reorientation problem. 43
4.9 Experimental orientation as a function of time for the rotation acceleration test. 44
4.10 Simulated and experimental trajectories as a function of time for the rotation acceleration test. 44
4.11 Simulated and experimental velocities as a function of time for the rotation acceleration test. 45
4.12 Path of the vehicle and thrust of the right (green) and left (red) motors at different positions along the path. 47
4.13 Thrust and yaw angle as a function of time for the third experiment. 47
4.14 Path of the vehicle and thrust of the right (green) and left (red) motors at different positions along the path. 48
4.15 Thrust and yaw angle as a function of time for the fourth experiment. 49

A.1 Program description 61

Nomenclature

Abbreviations

CG	Center of gravity
CV	Center of volume
PCB	Printed Circuit Board

Mathematical Symbols

Ω	Motor angular velocity
Φ	Cost of the initial and starting point
ϕ	Roll
Π	Normalized time
ψ	Yaw
θ	Pitch
a	Acceleration vector
B	Buoyancy force
C	Drag coefficient
C_l	Linear drag coefficient
C_p	Constant relating the square of the angular velocity of the propeller and the thrust
C_r	Rotational drag coefficient
C_r	Rotational drag
D	Drag acting on the vehicle
d_D	Distance between the CG and the center of drag
D_p	Constant relating the square of the angular velocity of the propeller and the thrust
d_p	Damping coefficient of the propeller
d_{CV}	Distance between CG and CV
d_{T_y}	y distance between CG and horizontal thruster

d_{T_z}	z distance between CG and horizontal thruster
e	Events
F_d	Drag force
F_g	Gravity force
G	Performance index
g	Gravity vector
h	Path constraint
$H(y)$	Constraints on the problem
J	Inertia matrix
J_p	Inertia of the propeller
J_z	Experimental inertia of the airship around the z axis
k	Node number
M	Moment acting on the vehicle
M_a	Applied torque for calibration
M_a	Applied torque on the airship for the damping characterization
M_T	Moment on the vehicle caused by the thrust
N	Number of nodes used in the problem
p	Body angular velocity along the x axis
q	Body angular velocity along the y axis
R	Rotation matrix
r	Body angular velocity along the z axis
S	Matrix relating a vector expressed in the inertial frame and in the body frame
T	Thrust acting on the vehicle
t	Time
V_r	Incoming wind velocity from the vehicle's frame of reference
V_v	Vehicle velocity
V_w	Wind velocity
W	Matrix relating the rate of change of the euler angles and the angular velocity in the body frame
x	x axis
$x(t)$	State vector
y	y axis
z	z axis

Superscripts

b	Frame of reference fixed to the airship.
i	Inertial frame of reference.

Subscripts

0	Initial
f	Final
L	Lower bound
U	Upper bound

Chapter 1

Introduction

1.1 Motivation

Airships were the first vehicles that allowed humans to fly while controlling their direction. They have been used extensively at the beginning of the century for transportation and military purposes. Dirigibles offered long endurance flight at high speed when compared to ground vehicles, and larger airships could move faster than 100 km/h, a revolution at the time. Technological advances in planes, as well as the Hindenburg catastrophe, reduced the use of dirigibles after the Second World War. The concept has seen a renewed interest since the beginning of the millennium. Their inherent safety, long range and flight time make these vehicles ideal as mobile telecommunications, surveillance, and monitoring platforms [1]. They can also be used for search and rescue missions, to deliver payload, to locate survivors or, thanks to their ability to hover in place, as a relay to replace destroyed telecommunication platforms [2]. Recent academic projects include project AURORA, a multiple phase endeavor which, at term, should result in the construction of an airship with over 100km range and over 24h flight duration [3, 4]. The authors want to use the vehicle for biodiversity, climate research, and monitoring. As another example, the United States army has developed a high altitude airship to provide 16kW to the surveillance payload, have an operating range of 2000 miles and stay in the air for 21 days[5].

To perform their missions, airships are required to move from one point to another in space, keep their position, take-off, land, follow a target, and other maneuvers. Whether the airship is used for surveillance or payload delivery missions, energy efficiency is important to maximize flight time and distance covered. Consequently, a wide range of techniques have been develop to plan the path and control airships, as multiple paths are generally possible [6]. Each path will have a different energy requirement, risk factor, and time associated to completion. Airships are also required to perform complex dynamical manoeuvres, such as position for storage, or coverage of a moving target.

While many path planner exist, as described in section 1.5.1, each has their own limitation and drawback. This research will attempt to use a pseudo-spectral optimal control solver to generate a path taking into account the vehicles dynamics in a wide range of conditions. The result of this work can then be used on a real airship with a controller to follow the generated path, or it could be use to compare the results to other types of path planners.

1.2 Problem description

Due to the large search space of a path planning problem, many assumptions are often made to reduce the search time and to improve the convergence of the optimizer. The airship can be described by a state vector $x(t)$ and a control vector $u(t)$. As a simplification, airships are sometimes described as a single point, with or without mass, which removes the orientation of the vehicle from the problem. The state vector $x(t)$ is then limited to a position vector. For a better approximation, the state would also include the orientation of the vehicle. When the vehicle travels at a relatively constant height, with little change in the environment in the vertical direction, the problem can be assumed to be in two dimensions under certain conditions. In three dimensions, the orientation is often described using Euler angles or quaternions. When the simulated path is short relative to the acceleration time, differential constraints on the velocity and angular velocity must be considered. Consequently, velocity and angular velocity will also be part of the state vector. Moreover, the state vector can include other variables such a motor velocity or volume of the ballasts. Those states can also have differential constraints. The control vector $u(t)$ is a vector of functions representing the inputs of the system. In the case of airships, it represents the command sent to the thrusters, the control surfaces, and the air pump of the ballast.

After the proper assumptions have been made for a particular problem, a very large or infinite number of feasible paths are still typically possible between two points. Unless the only requirement is a feasible path, the paths have to be evaluated against a certain performance index to minimize or maximize this performance index[7]. For example, airships may be required to minimize the time of arrival, the flight-time, or the energy used. Alternatively, for telecommunications or surveillance applications, maximizing the time of flight is often critical. In that case, the performance index would be the energy consumed for the total maneuver. Mathematically, the problem of path planning for a UAV can be represented by:

$$\min G(x(t), u(t)) \tag{1.1}$$

where

$$G = G_i(x_i) + G_e(x_e) + \min \int_0^t f(x(t), u(t))dt, \quad x \subset X \quad u \subset U \tag{1.2}$$

where $G_i(x_i)$ is the starting point cost, $G_e(x_e)$ is the end point cost, $f()$ is the

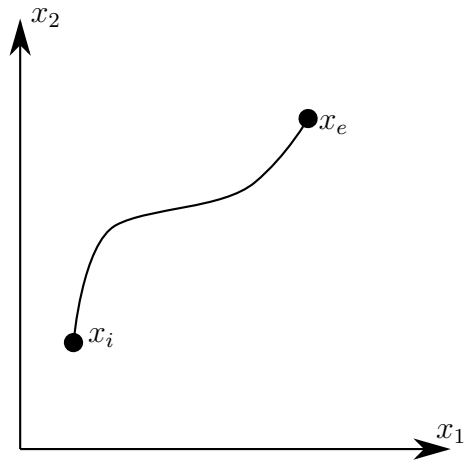


Figure 1.1: A possible path in a 2D problem

cost function, and X is the search space. The search space is restricted by the boundaries of the environment, such as the ground and the maximum height, and by the obstacles. In addition, the maximum velocity and angular velocity are limited due to the physical constraints of the airship. The search space of the control inputs is U, and it is limited by the maximum electrical command. Moreover, a model could include dynamical constraints on the derivative of the states $\dot{x}$.

1.3 Thesis objective and contribution

The objective of this thesis is to study optimal control for airship path planning and, more specifically, for small airships. To this end, a pseudo-spectral optimal control solver will be used to generate a time optimal path for a small dirigible. The feasibility of this method will then be demonstrated with experimental tests. Since a good dynamical model is required for generating a feasible trajectory, the second objective of this thesis is to model the airship based on experimental values.

Airship models traditionally use the center of volume as the frame of reference. To simplify calculations for the optimal control solver, the airship model has been redeveloped around the center of gravity. Novel techniques have been used to characterize the physical constants of the dynamics model of the airship, such as using the tracking system to track the response to an input, or recording the sound of the motor. The mass and drag model was characterized experimentally. Finally, to the authors knowledge, this is the first study of pseudo-spectral optimal control applied to airships.

1.4 Thesis layout

The thesis will be divided into four parts. First, a review of existing methods of path planning will be presented, followed by a review of airship modeling techniques and the psendo-spectral optimal control methods. Second, the airship will be modeled using Newton's equations and characterized experimentally. Third, the optimal path will be generated for some test situations and finally, the performance of the airship in these situations will be evaluated experimentally.

1.5 Literature review

1.5.1 Methods of path planning

Path planning is a method to find a suitable path, while reducing a cost function. Goerzen et al. conducted an extensive review of path planning for UAV guidance[8]. In the case of a UAVs, the cost function is typically time, energy, safety of the path, coverage, or a combination of the previous factors.

A general path for a 2D problem is seen in figure 1.1. In this problem, the state vector x can be any state, such as position, orientation, motor velocity, etc. In the case that the initial and final points are known, only the integral portion of the Eq. 1.1 is kept for the optimization. For UAVs, the search space is typically restricted by the terrain, the maximum altitude reachable by the vehicle, exclusion zones due to airspace restrictions and air traffic, as well as a security buffer.

In general, path planning for UAVs can be separated into two categories: methods that respect differential constraints, and methods that do not. Differential constraints arise from the dynamical model, the maximum speed and the acceleration. Mathematically, adding differential constraints is equivalent to adding constraints on the derivative of the state vector. The methods not using differential constraints are typically faster and more suited for large scale problem, but can lead to unfeasible paths. Using differential constraints is computationally expensive and requires an accurate dynamical model, but has the advantage of generating optimized and precise trajectories. This work will focus on numerical solutions including differential constraints.

A commonly used approach is to discretize the problem into a grid, as seen in Fig. 1.2. The airship travels on the edges between the points. The search space is then transformed from a continuous search space to a discrete space. Each edge can be assigned a weight, evaluated from the cost function, and a network search algorithm can be used.

Once a problem is discretized, a multitude of search algorithms and techniques can be used, such as a Markov decision process, A*, D*, or a rapidly

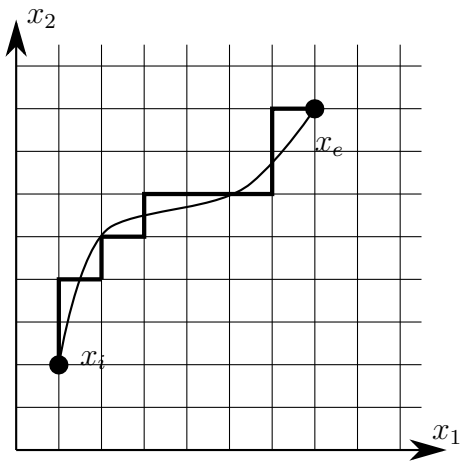


Figure 1.2: Discretized path in a 2D problem

exploring random tree[9]. A good overview of different discretized (grid based) algorithms is provided by [10]. They are part of classic path planning methods, but they have been extended to cover partially known or dynamic environments. In discrete, grid-based algorithms, a path planner is complete if it will always find a solution given a finite amount of time in a finite search space, and it is optimal if it always finds the path with the lowest cost per edge. Consequently, an optimal path planner is always complete. Mathematically, the search grid is represented as a graph. The grid is typically square, but other forms have been explored [11]. The grid points can also be moved after the search, a process called relaxation [12]. Limitations can be added on the search to respect constraints on the model. For example, Reckoskie [13] limits the search from a node to the nodes the vehicle can reach, given its turning radius. After, the discretized path is found, trajectory smoothing is often applied to the path using a polynomial function [14], a NURBS, or arcs of circles. The smoothed trajectory can be constrained to respect the turning radius of the vehicle.

The complete and optimal definition has two more weaker forms: resolution complete and probabilistically complete. A path planner can be resolution complete/optimal for the discretized space, rather than the physical space, often a subset of $\Re$. Moreover, a path planner that is probabilistically complete/optimal will find the solution given an infinite amount of time. Those path planners often use a random element to converge. Many path planners assume a static environment instead of a time varying environment. In the case of an airship, the environment varies due to the changing wind conditions during flight or other flying vehicles. Air safety is a very important concern for the cohabitation of UAVs and civil aircraft, so a path planner should take those variables into account.

In the last two decades, path planners have increasingly incorporated dynamical constraints [8], wind disturbances [13], and risk factor[15]. When the

dynamical model is well known, optimal control solvers are well suited for those operations. Those solvers have been extensively used for planning the motion of airplanes [16] and rockets [17]. Path planners using optimal control generate a locally optimal path respecting the differential constraints of the vehicle. The origin of optimal control comes from the calculus of variation, which deals with the optimization of functionals. A functional is analogous to a function for functions. One of the first calculus of variation problems was the brachistochrone problem, which was solved by Bernoulli in 1697 [18]. Given a point mass, affected only by gravity, find the path that minimizes the time to travel between two points in space. The optimal path is a cycloid, shown in Fig. 1.3.

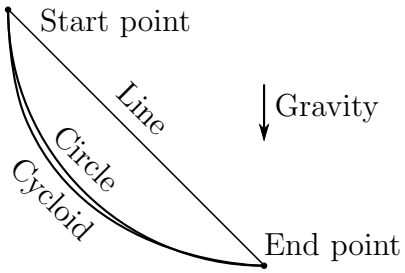


Figure 1.3: Possible solutions to the brachistochrone problem.

For example, [19] demonstrated the use of direct collocation to generate a path for a small airplane with a fixed camera pointed toward the ground. Direct collocation, which solves the optimal control problem by approximating the solution with polynomials and minimizing the differential errors, will be presented in Section 2.4. In direct collocation using a psendo-spectral method, the solution is approximated with polynomial functions, which are evaluated to check if they respect the dynamical constraints of the problem. In the case of [19], the parameter being optimized is the time that the camera is pointed at the payload. Since an aircraft cannot stay in place, there will be time that the camera will not be pointed at the target, such as when the vehicle is banking to turn around. In another example, the performance index being minimized is the accumulated heat in an Apollo capsule during descent [20]. The optimal control solver allows for a solution for varied conditions of wind and target ground speed, while respecting the physical constraints of the flying vehicle. The advantage of the direct approach is that it has good convergence and it has been applied to difficult problems.

1.5.2 Uses of optimal trajectories for flying vehicles

Historically, the formulation of the trajectory problem as a calculus of variation problem has been used for rockets and high altitude aircraft. Calculus of variation was used in 1962 at the height of the cold war to calculate the fastest ascent to high altitude of aircraft [21]. Optimal control then allowed

the calculation of optimal ascent trajectories for rockets. Each rocket has a unique optimal path that depends on its thrust to weight ratio, specific impulse, quantity of fuel, staging and mass. This problem, named the Goddard ascent problem, was described and analyzed by Robert H. Goddard in 1919 [22]. Rockets are ideal dynamical systems for trajectory generation with an optimal control solver, as they are well characterized, and are operating in a well known environment. Approximate analytical solutions for the case of an airship modeled as a point mass have also been calculated and found to be consistent with numerical solutions using collocation approach [23].

More recently, optimal control solvers have been used for fixed wings and airships. The ability of optimal control solvers to take into account wind velocity has been demonstrated before. For instance, [24] used a collocation technique on a fixed wing UAV to increase the energy of the vehicle using wind gradients, in a similar way that albatrosses are flying for hours along the coast. Optimal control solvers can also optimize the path of multiple planes for safety, while minimizing the flight time before the landing procedure at airports [25]. The fact that multiple aircraft are taken into account in the problem is reflected in the state equations and in the search space of the state of each aircraft. Adding new aircraft to the problem involves adding more states to the problem description and dynamical restrictions on those states. The possible position of each aircraft is limited by the position of the other aircraft: each plane has an exclusion zone around it. The exclusion zone around each plane is dynamic. To improve convergence of the optimal control solver, a rapidly exploring random tree can be use to first generate a rough possible path [25]. The path generated is used as a guess for the optimal control solver. Other ways of searching the solution space have been explored. Genetic algorithms have also been used to plan airship trajectory. For instance, clonal selection has been successfully applied to optimize the trajectory of a high altitude airship [26].

Optimal control for the trajectory optimization of large airship over long distances (multiple kilometers) has been demonstrated by [27]. Their airship model included wind tunnel data and CFD simulations of estimating the added mass from the mass of air displaced by the vehicle. The simulation used a wind model. The authors concluded that their optimization technique using optimal control resulted in a very realistic flight trajectory due to the precise airship model, the flight restriction, and the environment model used.

Optimal trajectories generally require a precise dynamical model. A statistical analysis of the error under varying parameters can be used to quantify how precise a model has to be and if the computed results are valid [24]. This kind of analysis is done by varying randomly important input parameters and analyzing the output.

This thesis will apply a technique that has seldom been used for airship path planning. Its other contribution to the research on small airships is the

use of an experimental model to verify the solver used as well as the accuracy of the modeling.

Chapter 2

Modeling and theory

2.1 Modeling of the airship

2.1.1 Modeling of airships in the literature

Airships are typically modeled using Newton's equations in a local frame of coordinates, with the axes fixed to the body. The center of reference typically chosen for modeling is the center of volume. The large mass of air displaced by the airship relative to the mass of the airship itself will create a virtual mass and inertia [28]. The model represented here is from [28], but similar to [27, 26, 29, 30].

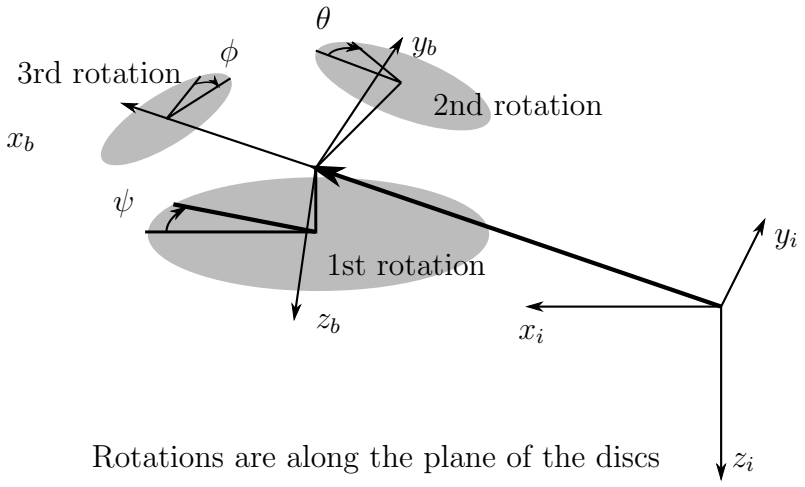


Figure 2.1: Angle convention

In most of the papers reviewed, two frames of reference are used: the inertial frame of reference and the body frame of reference, as shown in Fig. 2.1. Most path planners are using Euler angles for describing the airship

orientation, while studies on airship controllers are using both Euler angles and quaternions. The dynamical equations are expressed using two equations. The first one is the sum of forces and torques in the body frame, seen in Eq. 2.1, and the second one relates the velocity in the body frame and the velocity in the inertial frame, as seen in Eq. 2.2. Due to the modeling in the body frame, inertial forces appear in the model.

$$M_d\dot{\mu} + C(\mu)\mu + T_a(\mu_a) + g(\eta) = T_p \tag{2.1}$$

$$\dot{\eta} = S^{b\to i}(\eta)\mu \tag{2.2}$$

In the previous equations, the vector μ is the velocity and rotational velocity expressed in the body frame and μ_a is the velocity of the wind in the body frame. The term η is the orientation of the ship. M_d is a 6×6 matrix containing the mass, the inertia of the vehicle, the terms appearing from choosing a frame of reference different from the center of mass, and the added mass from the displaced fluid. The term C represents the centrifugal torque from the Coriolis force, T_a is the torque from the aerodynamic force and torque, and g is the sum the buoyancy and the gravity forces and torques. The last force is T_p is the sum of thrusters force. The matrix S converts the angular velocity in the body frame (p, w, r) to the derivative of the Euler angles in the inertial frame$(\dot{\phi}, \dot{\theta}, \dot{\psi})$.

The rotational damping terms are often ignored for large vehicles for linearizion as, at high speeds, the aerodynamic forces from the control surface become relatively large. In the case of small airships, such as the one studied in this thesis, the damping from fluid forces cannot be neglected. The modeling of the airship presented in the following section will be derived from Newton's equations and is consistent with the models available in the literature.

2.1.2 Modeling of a small airship

The airship is modeled as a six degrees of freedom mass in a viscous fluid, with viscous damping in rotation and drag in its linear movements. The airship is typically used by hobbyists and is commercially available. The balloon is an oblate spheroid with a major radius of 50cm and a minor radius of 30 cm. It is inflated using helium. The assembled airship is shown in Fig. (3.3).

The nacelle assembled on a single PCB, and it is comprised of three small motors, a receiver, the motor controllers, a small battery, and a small micro controller for the logic. The airship is sold under the name Microblimp by the company Plantraco. The modification of the transmitter is described in the experimental section. An enlarged view of the single printed circuit board (PCB) gondola is shown in Fig. 2.3. Two lateral thrusters are aligned in the longitudinal axis. During testing, it was found that the vertical axis dynamics was almost completely decoupled from the horizontal axes dynamics. Thus,

Figure 2.2: Airship during a minimum time linear displacement test.

the experiments were performed on a 2D planes. The vertical thruster was removed to prevent interference and reduce weight. The battery powering the vehicle is a single 90 mAh lithium-polymer cell delivering 4.1V at full charge. Each motor provides around 0.01 N of force. A model of the airship is shown in Fig. 2.4.

The following assumptions are used:

1. Rigid body with six degrees.
2. Lift negligible.
3. The Reynolds number is sufficiently high to apply the drag equation, i.e. drag is proportional to the square of the relative wind velocity.
4. Constant center of gravity.

The dimensions of the airship are shown in Fig. 2.1. The forces are modeled around the center of gravity (CG) and the buoyancy force is applied at the center of volume (CV). The CV is calculated using Solidworks.

2.2 Kinematic equations

The optimal solver is not restricted to a specific representation of angle or position. Quaternions have the advantage of not having a singularity. However,

Figure 2.3: Close-up of the nacelle

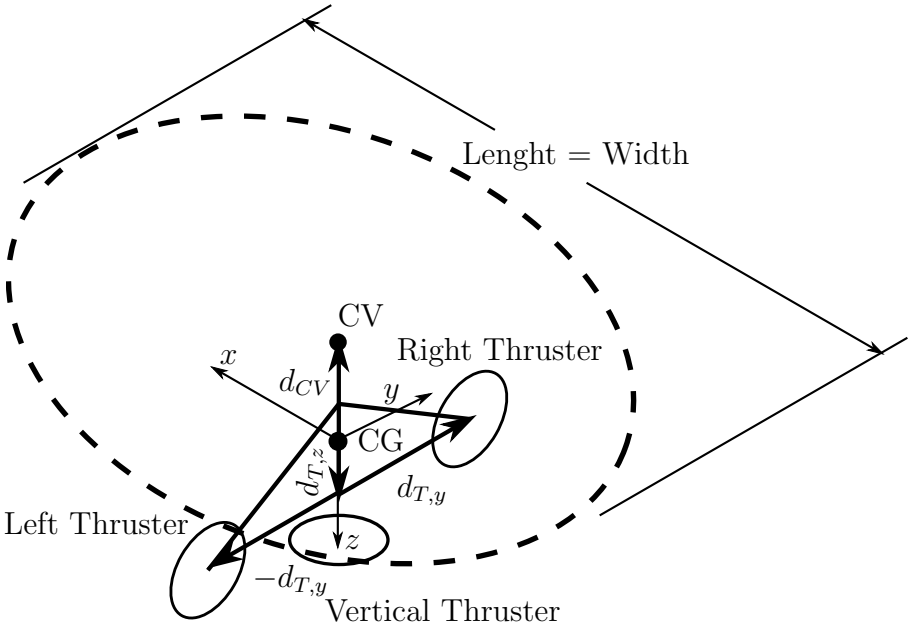


Figure 2.4: Airship dimensions

Table 2.1: Important dimensions of the airship

Variable	Description	Value (cm)
d_{CV}	Distance between CG and CV	5
d_{T_y}	y distance between CG and horizontal thruster	6
d_{T_z}	z distance between CG and horizontal thruster	5
-	Balloon length	50
-	Balloon height	30

due to the dynamical constraints on the airship, it is unlikely to be pointing directly up. Quaternions are in 4 dimensions, which would increase the number states used to describe the problem. Therefore Euler angles were used. Two reference frames are used: the body frame of reference, fixed to the airship and denoted with the superscript b, and an inertial frame of reference, fixed to the ground and denoted by the superscript i. The orientation of the vehicle is represented with three angles: ϕ, θ and ψ, which represent rotation in roll, pitch, and yaw. Fig. 2.1 shows the angle convention used and the frame of reference.

We define the following rotation matrices, where the subscripts x, y and z represent the axes along which the rotation is performed:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \; R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \; R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

To transform a vector v^b in the inertial frame to a vector v_i in the inertial frame, we apply the following transformation:

$$R_z R_y R_x v^b = S(\eta)^{b\to i} v^b = v^i \tag{2.4}$$

The matrix $S(\eta)^{b\to i}$ is defined in the following equation, where the function sin is represented by 's' and the function cos is represented by 'c'.

$$\begin{bmatrix} \mathrm{c}(\psi)\mathrm{c}(\theta) & -\mathrm{s}(\psi)\mathrm{c}(\phi) + \mathrm{c}(\psi)\mathrm{s}(\theta)\mathrm{s}(\phi) & \mathrm{s}(\psi)\mathrm{s}(\phi) + \mathrm{c}(\psi)\mathrm{s}(\theta)\mathrm{c}(\phi) \\ \mathrm{s}(\psi)\mathrm{c}(\theta) & \mathrm{c}(\psi)\mathrm{c}(\phi) + \mathrm{s}(\psi)\mathrm{s}(\theta)\mathrm{s}(\phi) & -\mathrm{c}(\psi)\mathrm{s}(\phi) + \mathrm{s}(\psi)\mathrm{s}(\theta)\mathrm{c}(\phi) \\ -\mathrm{s}(\theta) & \mathrm{c}(\theta)\mathrm{s}(\phi) & \mathrm{c}(\theta)\mathrm{c}(\phi) \end{bmatrix} v^b = v^i \tag{2.5}$$

The opposite transformation is done with a transpose of $S^{i\to b}$ such that:

$$(S(\eta)^{b\to i})^{\mathrm{T}} v^i = S(\eta)^{i\to b} v^i = v^b \tag{2.6}$$

2.3 Kinetic equations

The airship is modeled as a solid mass of mass m and of inertia J, and dynamical equations for the acceleration will be written in the body reference frame. The forces acting on the vehicle are the gravity force F_g, the buoyancy force F_b, the thrust force T_f, and the aerodynamic forces F_a. The moments acting on the vehicle are the moments due to the inertia J, the drag C_r, and the thrust forces. The angular velocities p, q, and, r are the angular velocities along the x, y and z axes, respectively.

The velocity change in the body reference frame is:

$$\begin{bmatrix} \dot{V}_x^b \\ \dot{V}_y^b \\ \dot{V}_z^b \end{bmatrix} = F^b/m - \begin{bmatrix} p \\ q \\ r \end{bmatrix}^T \cdot \begin{bmatrix} V_x^b \\ V_y^b \\ V_z^b \end{bmatrix} \tag{2.7}$$

The right-most term of the previous equation is due to the modeling in the body frame. The position change in the inertial frame is:

$$\begin{bmatrix} \dot{\mathbf{x}}_\mathbf{x}^\mathbf{i} \\ \dot{\mathbf{x}}_\mathbf{y}^\mathbf{i} \\ \dot{\mathbf{x}}_\mathbf{z}^\mathbf{i} \end{bmatrix} = R_z R_y R_x V^b \tag{2.8}$$

where V^b is $[V_x^b, V_y^b, V_z^b]^T$. The angular acceleration in the body frame is:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \dot{\omega} = J^{-1}(M^b - \omega \times J\omega) \tag{2.9}$$

In the previous equation, J is inertia matrix including the virtual mass. Since the frame of reference for the angular acceleration is in the body frame and the description of the angular positions is in the inertial frame, it is necessary to transform the angles from one frame to another. This transformation will be done using the matrix W. The airship angular rate of change in the body frame is found by rotating each vector of the Euler angular rate of change in the body frame.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & \sin -\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & \sin -\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = W^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.11}$$

It is possible to calculate W by inverting the matrix (W^{-1}). Thus, the equation that defines the rate of change of the orientation of the vehicle is:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = W \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \; W = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \tag{2.12}$$

In summary, the state space equations are Eq. (2.7), Eq. (2.8), Eq. (2.9) and Eq. (2.12).

For the simulation, another state vector, $\mathbf{\Omega}$, will be added to represent the propeller velocity, since the propeller has inertia and does not react instantly. The dynamic equations governing those state will be described in Section 2.3.2.

To complete the dynamic model, the input forces in the body frame, F^b, and the input moments in the body frame, M^b, have to be added. The input forces are calculated around the CG.

The input forces are due to the buoyancy force B, the gravity g, the aerodynamic drag and lift force D, as well as the thrust from the propellers T. The buoyancy force and the gravity force, which always stay vertical, have to be transformed to the body frame by the matrix $S^{i\to b}$.

$$F^b = S^{i\to b}B^i + S^{i\to b}g^i + D^b + T^b \tag{2.13}$$

For the experiments, the vehicle operates in a neutrally buoyant plane due to small weights added to the airship. Therefore, the buoyancy force is equal and opposite to the gravity force and can be neglected.

The thrust vector T is the sum of the thrust from the three thrusters on board (i.e., T_1, T_2, and T_3).

$$T = T_1 + T_2 + T_3 \tag{2.14}$$

Where,

$$T_1 = \begin{bmatrix} \bar{T_1} \\ 0 \\ 0 \end{bmatrix}, \quad T_2 = \begin{bmatrix} \bar{T_2} \\ 0 \\ 0 \end{bmatrix}, \quad T_3 = \begin{bmatrix} 0 \\ 0 \\ \bar{T_3} \end{bmatrix}, \tag{2.15}$$

In the previous equation, the magnitude if the thrust is represented with a bar over the variable. Note that there is no term in the y direction, which means that the vehicle cannot accelerate laterally on its own. The forces in Eq. (2.13) can also produce a torque on the airship. This torque is modeled by Eq. (2.16).

$$M^b = d_{CV}S^{i\to b}B^i + d_D D^b + M_T^b \tag{2.16}$$

where d_{CV} is the distance between the center of gravity and the center of volume, and d_D is the distance between the center of gravity and the center of drag. The term $d_{CV}S^{i\to b}B^i$ is due to the the torque from the buoyancy force. Since the buoyancy force is applied at a certain distance from the CG, a torque is generated using the matrix $S^{i\to b}B^i$. The orientation of B^i also has to be transformed in the body frame. The center of drag is at the position described by d_D. Thus, the drag causes a torque $d_D D^b$ on the airship. For this model, the CV and the center of drag will be assumed to be the same. The variable M_T^b is the sum of moments caused by the thrust forces from the thrusters. The torque of each thruster is the cross product of the distance from the center of gravity to the thruster and the force applied by the thruster.

2.3.1 Drag Model

The drag D is modeled mathematically in Eq. (2.13). Its magnitude is a function of the square of the magnitude of the incoming airflow velocity V_r, and is parallel to the incoming airflow V_r in the body frame. Consequently, the drag expressed in the body frame is approximated by:

$$D^b = C_l(\phi, \theta, \psi, V_w)V_r^b|V_r^b| \tag{2.17}$$

The drag coefficient $C_l(\phi, \theta, \psi, V_w)$ is a function of the angle of attack of the airship and the incoming inertial wind velocity V_w, and V_r is the relative wind velocity from the body frame of reference. The variable V_v is the vehicle velocity. Fig. 2.5 shows graphically the relationship between the velocities. The figure is true in the inertial frame and in the body frame. The relative wind velocity in the body frame is calculated in Eq. (2.18).

$$V_r^b = V_v^b - S^{i\to b}V_w^i \tag{2.18}$$

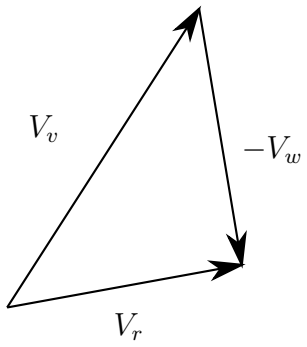


Figure 2.5: Relative velocities of the airship

As explained in the introduction, the drag coefficient is usually measured in wind tunnels or approximated using fluid theory. Due to cost and time

constraints, wind tunnel measurements could not be performed on the airship model. Instead, the response to a known input was measured using the vision tracking system. This response was fitted to the theoretical curve to find the coefficients.

The drag coefficient was found experimentally by capturing the airship's position in free fall. The vehicle was weighted with a known weight, which allowed to precisely calculate the force applied on the airship. The fall was recorded using the optical positioning system explained in the experimental section. By varying the position of the weight, it is possible to control the orientation of the vehicle during the fall and find the drag coefficient at multiple angles. The equation governing the drop experiment is the following:

$$ma = F_g - D \tag{2.19}$$

where m is the mass of the vehicle, including the virtual mass from the air, F_g is the gravity force, F_d is the drag force, and a is the acceleration vector. Each drag experiment experiment is performed in one dimension, so only the magnitude of the vector is considered. The drag force is:

$$|F_d| = C_l |V_r|^2 \tag{2.20}$$

The solution to Eq. (2.19) and Eq. (2.20) equation is:

$$x(t) = \frac{m \log\left(cosh \frac{\sqrt{C_l}\sqrt{F_g}(c_1 m + t)}{m}\right)}{C_l} + c_2 \tag{2.21}$$

The equation was fitted to the experimental results of the drop tests. The drop tests were performed by launching the airship vertically. From the moment the vehicle is released, its movement is described by equation 2.21. There are four unknowns in this equation: c_1, c_2, m and C. The value of c_1 and c_2 are not important here, since they depend on the time the experiment was started and the absolute value of the height measured. The value of C however is the drag coefficient of the orientation measured, and m_e is the mass of the airship added plus the mass of the added weight. Since the amount of air displaced is not the same during travel along the x and z axes due to different geometry, the experimental mass depending on the direction of acceleration. Multiple drop tests were perform to find the drag and the mass constants shown in Table 2.2 on page 22. An example of one of those fits can be seen in Fig. 2.6. The mass is determined experimentally from the measured curves, which means it includes the added mass from the displaced airflow.

The characterization of the rotational damping followed the same procedure as the linear damping. The dynamic equation has the same form, but

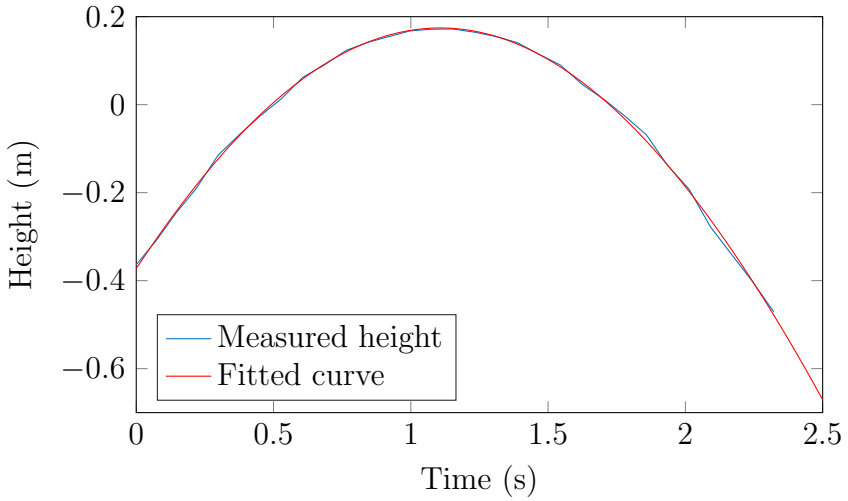


Figure 2.6: Characterization of a drop test

the velocity is replaced by the angular velocity ω:

$$M_a = C_r \omega^2 \tag{2.22}$$

In the previous equation, M_a is the applied torque and C_r is the drag coefficient in rotation. Instead of the gravity however, a constant torque M_a was applied using the thrusters of the ship. The resulting data was then fitted in Matlab to find the experimental inertia and the experimental drag. The solution for the z axis is then:

$$\psi(t) = \frac{J_z \log\left(cosh \frac{\sqrt{C_r}\sqrt{M_a}(d_1 J_z + t)}{J_z}\right)}{C_r} + d_2 \tag{2.23}$$

The solution for the x and y axes is the same, but the constants have different values due to the different geometry. The torque M_a is applied by the two thrusters. The thrusters are speed up to 100% and the airship is released. The pose of the airship is then recorded, and the unknown constants C_r, J_z, d_1, and d_2 are computed using a fit on the experimental data.

2.3.2 Thrust model

The thrust is assumed to be proportional to the square of the propeller rotational speed. Experimental tests on the gondola model showed that a model for the thrust was needed, since the motor required about one second to reach full thrust. This delay is not negligible when compared to the thrust time of the simulated maneuverer.

The thrust model is based on one similar to [31]. The variable J_p is the inertia of the motor, Ω is the angular velocity, d_p is a damping constant due to the drag of the air, and C_p is a constant that relates the square of the angular velocity of the propeller and the thrust.

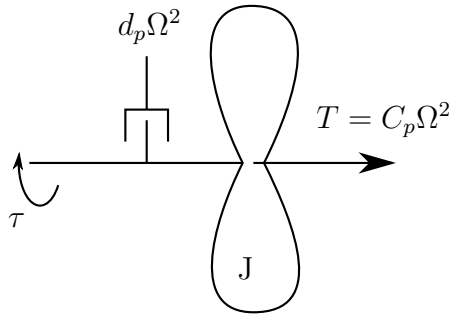


Figure 2.7: Thrust model of the airship

The governing equations are

$$J_p\dot{\Omega} = \tau - D_p\Omega^2 \tag{2.24}$$

$$T = C_p\Omega^2 \tag{2.25}$$

The torque τ is assumed to be controlled directly by the input sent to the airship. This torque is caused by the application of a DC voltage on the motor.

The motors are very small, therefore measuring their velocity directly is difficult. Since the frequency of the sound of the motor is proportional to the rotational speed, the normalized velocity profile for speeding up and down the motor was acquired using a sound recorder. The sound files generated were then analyzed using the software Audacity to sample the frequency change over time. An example of such a sample is shown in Fig. 2.8.

The frequency power spectrum of the previous signal is shown in Fig. 2.9. The peak frequency is 419 Hz.

Multiple short samples of the sound of the motor accelerating and decelerating were analyzed using frequency spectrum plot. For each sample, the frequency with the highest amplitude was recorded. This dominant frequency was plotted as a function of time. Since this dominant frequency is proportional to the motor speed, it was fitted with theoretical results of the differential equations as seen in Fig. 2.10 to obtain the normalized constants D_p. The drag constant D_p is normalized by setting the inertia J_p to 1.

The normalization can be done since only the propeller velocity normalized with respect to the maximum propeller velocity is needed to model the airship, as we are looking to establish the relationship between the input and the motor thrust. The absolute propeller velocity will not affect the result, as the thrust

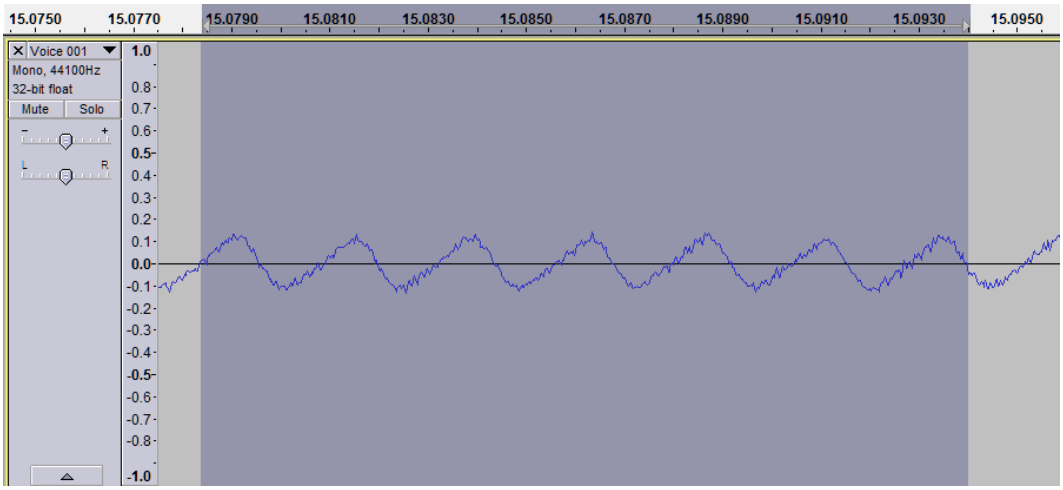


Figure 2.8: Sample of the analyzed spectrum of the sound emitted by the motor.

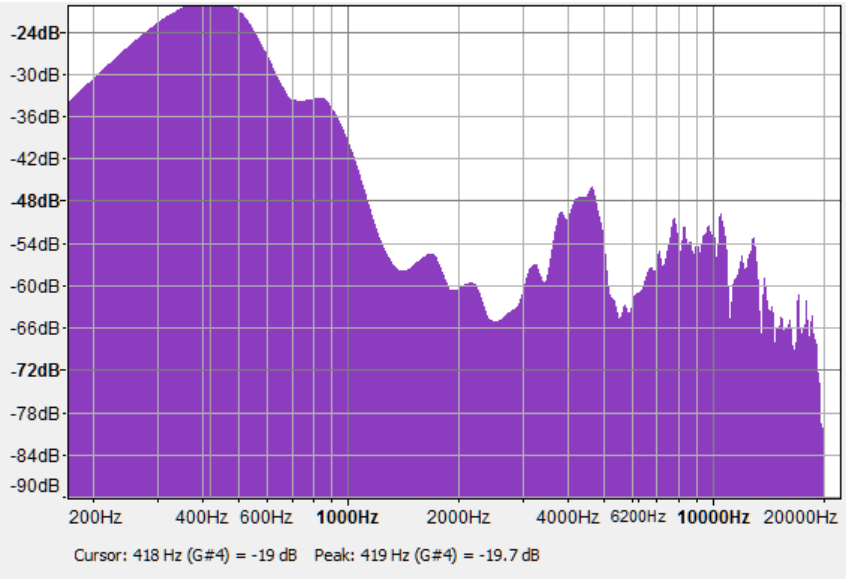
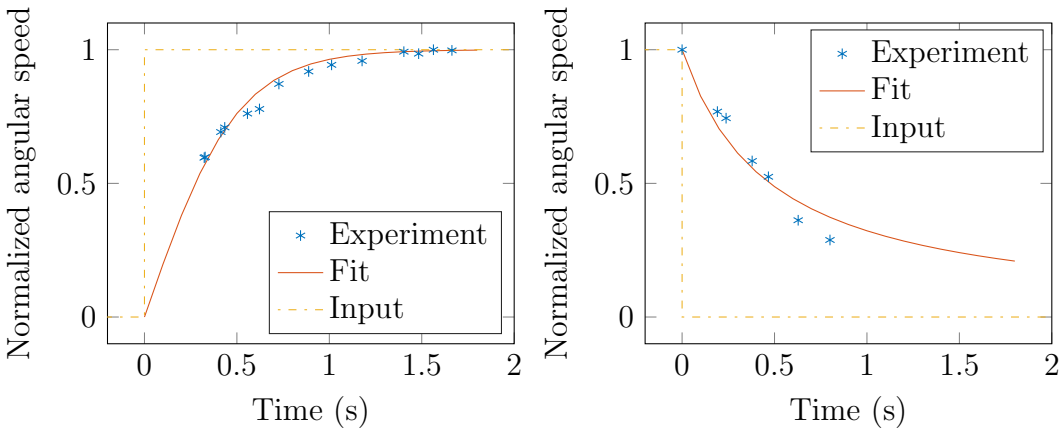


Figure 2.9: Frequency spectrum of a sample of the motor sound at full power.



(a) Response to a step input (b) Response to a reversed step input

Figure 2.10: Normalized response of a thruster during simulated and experimental test

is proportional to the square of the propeller velocity using the constant C_p, and this constant is computed experimentally.

In practice, the damping constant D_p was computed first. The sound of the motor responding to an inverse step input of full throttle to zero throttle was recorded. Since no torque is applied during deceleration, $\tau = 0$. Using a normalized rotational inertia of $J_p = 1$, the differential equation representing this situation is the following:

$$J_p \dot{\Omega} = -D_p \Omega^2 \tag{2.26}$$

The solution to Eq. 2.26 is:

$$\Omega(t) = \frac{1}{D_p t + c_1} \tag{2.27}$$

The solution to Eq. (2.24) is:

$$\Omega(t) = \frac{\sqrt{\tau}\tanh(\sqrt{\tau}\sqrt{D_p}c_1 + \sqrt{\tau}\sqrt{D_p}t)}{\sqrt{D_p}} \tag{2.28}$$

The value of D_p is found by fitting this function in Eq. (2.28) to the experimental data. The value of the unknown parameters were also found with non-linear fit to the experimental data. The results for a normalized step input and a normalized output are $d_p = \tau = 2.1$ with a 95% confidence interval range of $[1.5, 2.6]$.

The precision of the model decreases at low speed, but the low speed regime is less important for small motors, since the thrust produced at these speeds is very small (the thrust is proportional to the square of the angular velocity).

Finally, all the parameters used for modeling, and for which the derivation was demonstrated in this section, are shown in Fig. 2.2.

Table 2.2: Experimental value characterizing the airship

Variable	Description	Value	unit	95 % conf.
$\bar{T}_1$	Maximum thrust of a thruster	0.010	N	
C_x	Horizontal linear drag coefficient	0.046		± 0.04
C_z	Vertical linear drag coefficient	0.13		± 0.04
m_{ex}	Experimental total mass of the vehicle in x	0.077	kg	± 0.010
m_{ez}	Experimental total mass of the vehicle in z	0.117	kg	± 0.020
m_b	Mass of the gondola, battery, and balloon	0.0357	kg	± 0.005
m_g	Theoretical mass of the gas	0.0258	kg	± 0.001
$m_b + m_g$	Mass of the vehicle without virtual mass	0.0565	kg	± 0.005
$\frac{m_{ex}}{m_b+m_g}$	Horizontal virtual mass ratio	1.4		± 0.2
$\frac{m_{ez}}{m_b+m_g}$	Vertical virtual mass ratio	2.1		± 0.4
I_{ez}	Experimental inertia around the z axis	$2.7 * 10^{-3}$	$\text{kg} \cdot \text{m}^2$	$\pm 0.2 * 10^{-3}$
I_{ex}	Experimental inertia around the x and y axes	$6.0 * 10^{-3}$	$\text{kg} \cdot \text{m}^2$	$\pm * 10^{-3}$
C_{rz}	Rotational drag coefficient around the z axis	$2.7 * 10^{-4}$		$\pm 0.7 * 10^{-4}$
C_{rx}	Rotational drag coefficient around the x axis	$9.7 * 10^{-4}$		$\pm 2 * 10^{-4}$
d_p	Damping constant of the motor and propeller	2.1		± 0.6

2.4 Simulation method

The problem of finding a path for aerial vehicles is equivalent to minimizing a cost function with constraints. The cost function is generally composed of an integral over the path, and a cost associated with the start and end position. As described in the introduction, many methods have been used to minimize the cost function. The method presented here is a pseudo-spectral method, which takes into account the differential constraints of the problem and finds a local minimum solution.

2.4.1 Description of the general optimal control problem

Optimal control for trajectory optimization has historically been used in aerospace to solve the ascent problem of rockets. Since analytical solutions are only possible in very simple and specific cases, numerical solutions are used. Mathematically, the general optimal control problem is defined as follows, and presented using the same notation as the one adopted by the PSOPT. Given the state trajectories $x(t)$, $t \subset [t_0, t_f]$, and times t_o and t_f, find the optimal control trajectory $u(t)$, $t \subset [t_0, t_f]$ to minimize the following performance index:

$$G = \Phi(x(t_0), x(t_f), t_0, t_f) + \int_{t_o}^{t_f} L(x(t), u(t), t)dt \tag{2.29}$$

where the function Φ is the cost associated to the initial and final starting points and time, L is the cost associated to an infinitesimal section of the path, and G is the cost function. The state vectors $x(t)$ and the control vector $u(t)$ have the following form:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \dots \end{bmatrix} = \begin{bmatrix} x^i(t) \\ y^i(t) \\ z^i(t) \\ v_x^b(t) \\ v_y^b(t) \\ v_z^b(t) \\ p(t) \\ q(t) \\ r(t) \\ \phi \\ \theta \\ \psi \\ \Omega_1 \\ \Omega_2 \\ \Omega_3 \end{bmatrix} \qquad u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \dots \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \tag{2.30}$$

The Ω components are related to the motor command. The components of the control vector $u(t)$ are the controls sent to the three motors. The states $x(t)$ of the airship studied are position, velocity, and motor speed. If the airship had ballasts, the volume of those ballast could also be a state. The states have dynamic constraints due to Newton's equations. That means that the derivatives of the vehicle's states are functions of the vehicle's states, control inputs, parameters p, and time, as expressed in the following equation:

$$\dot{x}(t) = f(x(t), u(t), t), t \subset [t_o, t_f] \tag{2.31}$$

A simple example of the previous equation is that the derivative of the position in the inertial frame is equal to the velocity in the inertial frame. The solver will have to respect that constraint.

Typically, there are maximum and minimum values associated to the control inputs, the states, the parameters, the start time and the end time. Those constraints will be called bound constraints and are given by Eqs. 2.32 to 2.36 for the present airship.

$$\begin{aligned}
\text{Control:} \quad & u_L \leq u(t) \leq u_U, t \subset [t_o, t_f] && (2.32)\\
\text{States:} \quad & x_L \leq x(t) \leq x_U, t \subset [t_o, t_f] && (2.33)\\
\text{Start time:} \quad & t_{0L} \leq t_0 \leq t_{0U} && (2.34)\\
\text{End time:} \quad & t_{fL} \leq t_f \leq t_{fU} && (2.35)\\
\text{Total time:} \quad & t_f - t_0 \geq 0 && (2.36)
\end{aligned}$$

The subscript U is for the upper limit and the subscript L is for the lower limit. Additionally, it is possible to specify a path constraint h function of the states, the controls and the parameters as shown in Eq. (2.37).

$$h_L \leq h(x(t), u(t), t) \leq h_U, t \subset [t_o, t_f] \tag{2.37}$$

In the case of an airship, a path constraint may be a limit on the sum of the electrical current input of all motors, due to the maximum current draw of the battery. It could also be a restriction on the position of the vehicle.

Moreover, the initial and final values of the problem may be specified. For instance, the vehicle may have to start in a certain configuration, and the starting and ending values may be specified. Such condition are called events and are represented by the letter e.

$$e_L \leq e(x(t_o), u(t_o), x(t_f), u(t_f), t_o, t_f) \leq e_U, t \subset [t_o, t_f] \tag{2.38}$$

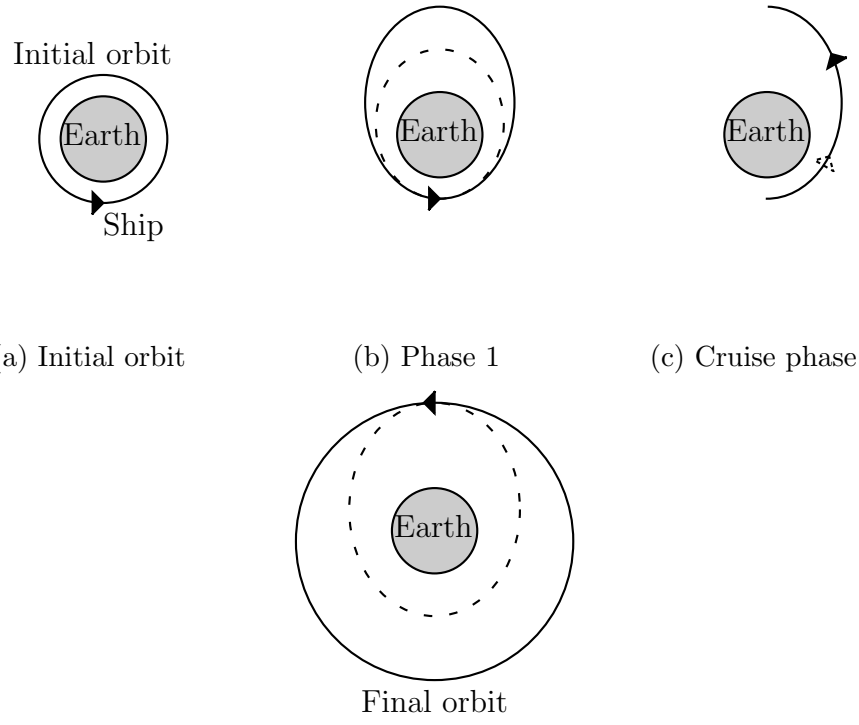
In the problem studied, the events are the upper and lower initial and final states. The upper and lower event can be the same, or different. For example,

in the experimental section, the set of initial upper event was the same as the set of initial lower event. However, the final pitch of the vehicle was not restricted, so the upper and lower events associated with the pitch were the maximum and minimum possible pitch expected for the vehicle.

2.4.2 Multiple phase problems

The problem can be described in multiple phases, which is useful when discontinuities appear in the problem or when the equations governing the system change. The phase will then be described by the superscript j, with $j = 1...N_p$, where N_p is the number of phases.

The different phases are related by the phase linkage. For instance, a spacecraft going from low circular orbit (Fig. 2.11a) to a higher circular orbit will require two phases of powered flight. The first phase (Fig. 2.11b) will raise the orbit to an ellipse and the second phase (Fig. 2.11d) will circularize it. The linkage in this case is the cruising phase between the two parts of the flight (Fig. 2.11c). Using Kepler's equations, the starting conditions of phase 2 are related to the final condition of phase 1.



(a) Initial orbit (b) Phase 1 (c) Cruise phase

(d) Phase 2

Figure 2.11: Multi-phase example in an orbital change problem

Another common example of multiple phase uses is the Goddard rocket maximum ascent problem, as seen in [32] and [33]. Use of multiple phase

problems for airships can be useful when the dynamical equations representing the system change suddenly (eg. dropping cargo), or to help convergence. For this thesis, the problem is implemented in the software as a multiple phase problem, but convergence was better when solving with a single phase.

2.4.3 Description of the solver used and its methods

Pseudo-spectral solution solvers were initially investigated for solutions to PDEs in fluid dynamics. For example, solutions to the well known 1D burger equation, as well as the 3D Euler equations, were obtained using those solvers [34].

This section will describe how the optimal control problem is transformed into a non-linear optimization problem. This process is well described in [32] and will be summarized in the present section, with added examples. Understanding the solver is important to analyse the solution, and to understand which problem are well suited for optimal control. The objective is to minimize the performance index G. The search space is the space of functions that respect the differential constraints on the problems and the functions that are control inputs. The first part of this section will describe how to approximate smooth functions using a weighted sum of Legendre polynomials. The second part will show how this approximation allows for very fast and accurate differentiation. The third part will demonstrate how to evaluate quickly if a function respects the differential constraints (state equations) of the problem using parts one and two. The last part will combine the previous parts to express an optimal control problem as a non-linear optimization problem, which minimizes the performance index G, while respecting certain constraints, such as differential constraints on $\dot{x}(t)$, the time constraints t_U and t_L, and the maximum and minimum state of $x(t)$, and control inputs $u(t)$.

The states and the control inputs of the system, which are functions of time, will be represented by polynomials. The problem will be discretized in N points Π_k, and the value of the states and controls will be interpolated between those points using Legendre polynomial a $P(\Pi)$. Let's define the auxiliary function f such that $f(\Pi_k) = P(\Pi_k)$. Thus, the Lagrange polynomial is:

$$P(\Pi) = \sum_{k=0}^{N} f(\Pi_k) \prod_{i=0, i\neq k}^{N} \frac{\Pi - \Pi_i}{\Pi_k - \Pi_i} \tag{2.39}$$

For example, the function $\sin\Pi$ discretized at point 0,1,2 and 3 is:

$$\sin\Pi \approx \sin 0 \left(\frac{\Pi-1}{0-1}\right)\left(\frac{\Pi-2}{0-2}\right)\left(\frac{\Pi-3}{0-3}\right) + \sin 1 \left(\frac{\Pi-0}{1-0}\right)\left(\frac{\Pi-2}{1-2}\right)\left(\frac{\Pi-3}{1-3}\right) + \tag{2.40}$$

$$\sin 2 \left(\frac{\Pi-0}{2-0}\right)\left(\frac{\Pi-1}{2-2}\right)\left(\frac{\Pi-3}{2-3}\right) + \sin 3 \left(\frac{\Pi-0}{3-0}\right)\left(\frac{\Pi-1}{3-2}\right)\left(\frac{\Pi-2}{3-3}\right) + \tag{2.41}$$

An example of the resulting function can be seen in figure 2.12.

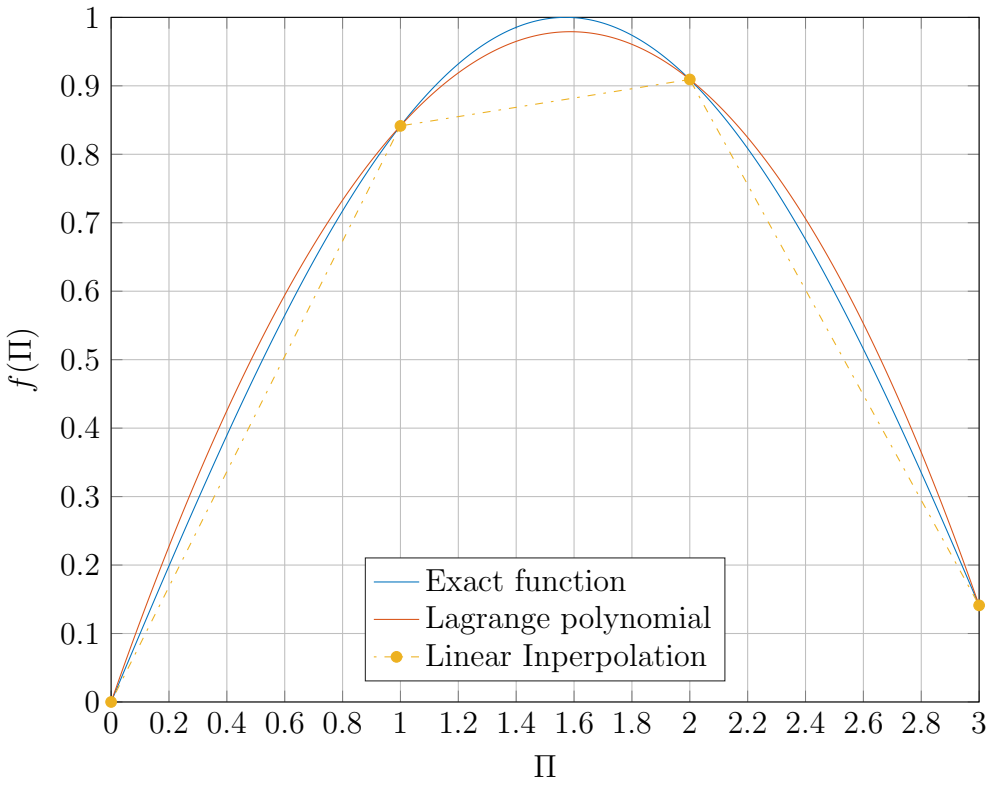


Figure 2.12: The function $\sin\Pi$, approximation with Lagrange polynomial and linear interpolation

A special case of Lagrange polynomials, the Legendre polynomials, will be used to represent states of the system. Legendre polynomials are orthogonal to each other and they are the solutions to the Sturm-Liouxville problem, which is a differential equation problem. They are defined explicitly as:

$$P_n(x) = 2^n \cdot \sum_{k=0}^{n} x^k \binom{n}{k} \binom{\frac{n+k-1}{2}}{n} \tag{2.42}$$

The first four polynomials are shown below and plotted in Fig. 2.13.

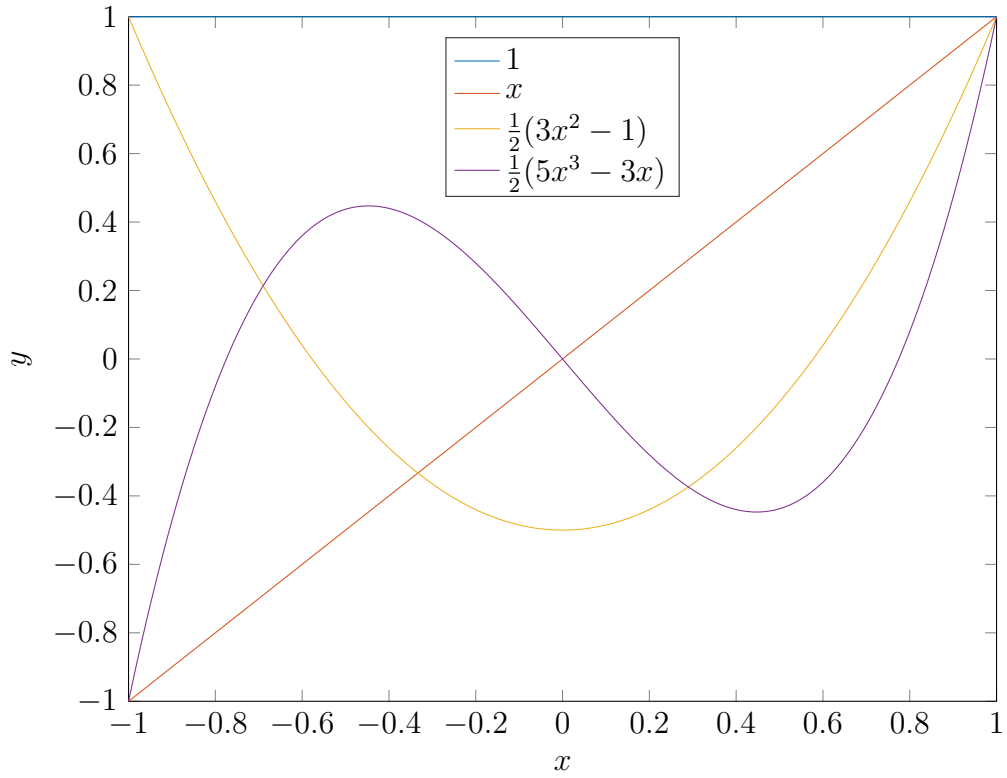


Figure 2.13: The first four Legendre polynomials

$$P_0(x) = 1 \tag{2.43}$$
$$P_1(x) = x \tag{2.44}$$
$$P_2(x) = \frac{1}{2}(3x^2 - 1) \tag{2.45}$$
$$P_3(x) = \frac{1}{2}(5x^3 - 3x) \tag{2.46}$$

The Legendre polynomials are orthogonal to each other, which means:

$$\int_{-1}^{1} P_n * P_m = 0, \quad n \neq m \tag{2.47}$$

The orthogonality has important applications with respect to the numerical stability and to approximate accurately the functions via spectral decomposition. The Legendre polynomials have the property of being very fast to integrate from $[-1, 1]$ using the Legendre-Gauss quadrature which states:

$$\int_{-1}^{1} f(\Pi) \approx \sum_{i=1}^{N} w_i f(\Pi_i) \tag{2.48}$$

In summary, the integral of a function can be approximated by a weighted sum of this function evaluated at the nodes Π_i, which are the roots of the Legendre polynomial. The weights w_i are calculated only once using the Gauss-Legendre quadrature, which makes the integral very fast to perform. Note that for the rest of the problem, the functions of time t will be mapped to Π, where the start time t_0 and the end time t_f will become -1 and 1, respectively. The transformation used is the following:

$$\Pi \leftarrow \frac{2}{t_f - t_0} t - \frac{t_f + t_0}{t_f - t_0} \tag{2.49}$$

Any node Π could be chosen to represent the discretized function. However, for numerical stability reasons explained in [32], the Legendre-Gauss-Labatto(LGL) nodes will be used. The LGL nodes are more precise than equidistant nodes. In general, the coefficient of the spectral decomposition of a function $f(\Pi)$ over a certain interval is performed by integrating $f(\Pi)$ multiplied by the spectral functions. It is also possible to differentiate quickly the function approximated by a Legendre polynomial using a differentiation matrix as shown in Eq. 2.50 [32].

$$\dot{f}(\Pi_k) \approx \dot{n}^N(\Pi_K) = \sum_{i=0}^{N} D_{ki} f(\Pi_i) \tag{2.50}$$

Consequently, the function to being optimized can be rewritten as

$$G = \phi(x(-1), x(1), , t_0, t_f) + \frac{t_f - t_0}{2} \int_{-1}^{1} L(x(\Pi), u(\Pi), \Pi dt \tag{2.51}$$

$$\approx \phi(x^N(-1), x^N(1), t_0, t_f) + \frac{t_f - t_0}{2} \sum_{k=0}^{N} \left[L(x^N(\Pi_k), u^N(\Pi_k), p, \Pi_k \right] \omega_k dt \tag{2.52}$$

In the previous equation, x^N and u^N are the states and the control inputs approximating the exact solution with Legendre polynomials of order N. For smooth functions, the integral can be approximated with very few points accurately. In the case of an airship, the function G will typically be the time, the power required, or a combination of both, such as in [13]. For the experiment presented in this thesis, the performance index is the final time t_f.

The solver will minimize the function G, while respecting the differential constraints imposed on x^N, $\dot{x}^N$ and u^N.

The vector representing the problem is y:

$$y = \begin{bmatrix} \text{vec}(U^N) \\ \text{vec}(X^N) \\ t_0 \\ t_f \end{bmatrix} \tag{2.53}$$

The function 'vec' takes column vectors and concatenates them vertically in a larger column vector. At the same time, the solver has to respect the constraints $H(y)$, such as the differential constraints, the path constraints, and the constraints on the time $t_f - t_0$.

Finally, the problem in Eq. (2.54), Eq. (2.55) and Eq. (2.56) is solved by the non-linear programming problem IPOPT.

$$\min_y G(y) \tag{2.54}$$

subject to:

$$H_l < H(y) < H_u \tag{2.55}$$

$$y_l < y < y_u \tag{2.56}$$

The vector $H(y)$ is composed of the events, the path constraints, the time $t_f - t_0$ and the differential defects. The differential defects are the differences between the evaluated differential constraints from the dynamic equations of the problem and the differentiation of the state vector. For example, if a problem sets $\dot{x}_1 = x_2$, the differential defect will be $\dot{x}_1 - x_2$.

The results of the optimization are the state vector y, which contains the optimized control inputs and states, expressed as a combination of Lagrange polynomial, the start time t_o and the end time t_f. The solution is not guaranteed to be the absolute minimum, but it is a local minimum that respects the constraints set on the system to the tolerance given of the solver. The process of solving the problem is showed in Fig. 2.14 as a simplified diagram, while the software and libraries used to solve the program are shown in Appendix A.

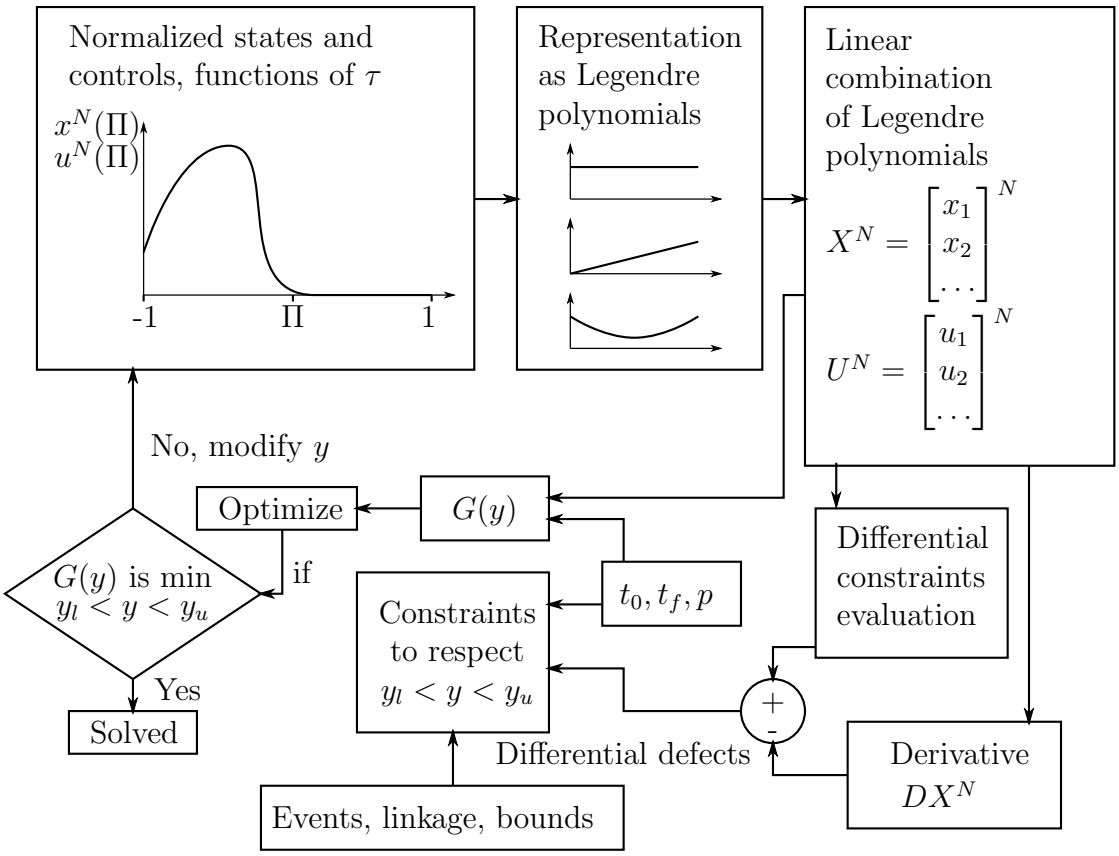


Figure 2.14: Simplified visual representation of the problem.

Chapter 3

Experimental platform

This section will discuss the equipment used for the experiment. The computer sends commands to the airship in open loop. The objective of the research is to evaluate the generated path directly and the drift caused by differences in the physical and mathematical model, as well as numerical error in the solution. Therefore, the implementation for real applications would include a controller or a high frequency replanning of the path to correct for drift.

3.1 Tracking software and pose estimator

Tracking of the airship is essential for the experimental tests, as it allows comparison of the airship's real trajectory to the simulation results. The first solution explored for pose tracking was using two cameras and active infrared markers made of LEDs embedded in small empty translucent spheres. Extensive work was done to build the system and integrate it with an open source code, Track-it-yourself, written by Andreas Pflaum, version 1.0. Unfortunately, the active marker system proved too heavy to install on the airship and the tracking was often lost. Instead, a system based on marker tracking was used. The software used for tracking is named ARUCO, and it is the result of the research of group 'Aplicaciones de la Visión Artificial' at the University of Córdoba, Spain. The tracking software tracks a 25cm by 25cm marker, such as the one seen in figure 3.1. The camera is a Microsoft Livecam, seen in Fig. 3.2, running in 640x480 pixel mode. Before use, the camera's lense model was estimated using ARUCO's camera calibration tool. When launching the software, the position of the marker(s) tracked by the camera is output in real time to a text file, which is then processed by Matlab for plotting. The camera, the marker, and the airship are shown in Fig. 3.3.

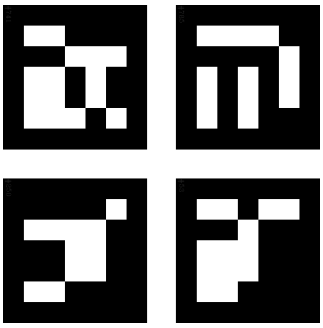


Figure 3.1: Tracking marker used on the airship.

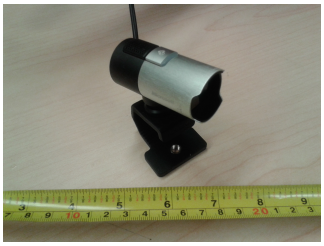


Figure 3.2: Camera used for tracking

Figure 3.3: Airship during a minimum time linear displacement test.

3.2 Data Transmission

The small airship is controlled with the original transmitter from the manufacturer, in which a signal is injected to replace the original commands. The control is sent by the computer, transmitted through the COM port using the serial protocol and read by an Arduino board running an Atmega368. The microcontroller then generates a pulse width modulation signal, with a cycle time proportional to the motors' command. The transmitter expects an analog voltage input, so the pulse width modulation signal is smoothed using an electrical low pass filter. Also, it was discovered that the impedance of the filter is high, so an operational amplifier is added to the circuit to force the input of the transmitter to follow the output of the low pass filter. The transmitter, low pass filters, microcontroler, and OP-amps are shown in Fig. 3.4.

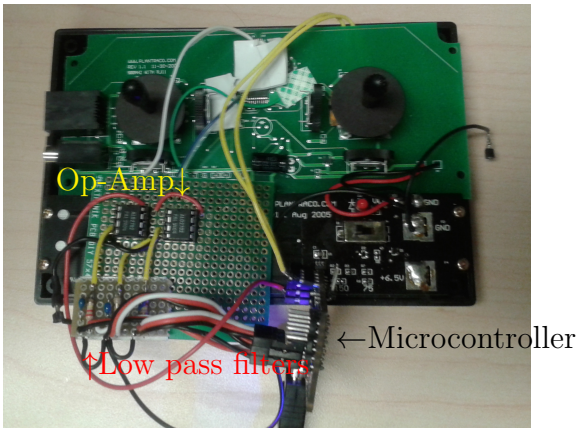


Figure 3.4: Transmitter used for the experiments.

The controller sends a pre-mixed signal to the airship. For instance, the forward command of the transmitter activates the two forward thrusters. The input had to be mix by the computer to be properly interpreted by the airship, since the optimal solver is calculating the input of each motor. The whole process, from the solving of the problem to the output to the motors, is illustrated in Fig. 3.5.

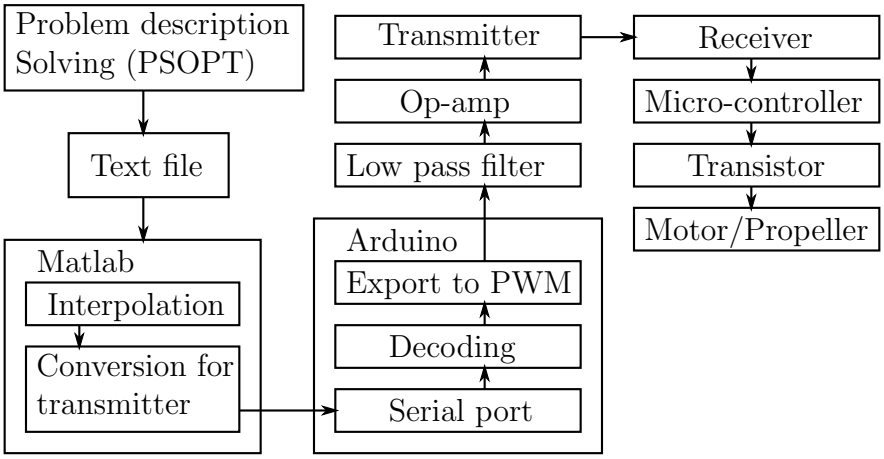


Figure 3.5: Experimental process, from the generation of the trajectory to the airship.

Chapter 4

Experimental tests and simulations

The objective of this section is to demonstrate that the simulated paths, generated using the experimentally defined parameters, are achievable, and that the airship behaves as predicted. Eventually, the same method for trajectory generation and parameter identification could be used in large airships. To achieve this, the small airship modeled in the previous section was subjected to the control vector computed by the pseudo-spectral optimal control method. The resulting paths are compared to the paths generated by the solver. The experiments are divided into two class. In the first class, simple trajectories are computed and executed experimentally. In the second class, more complex trajectories are demonstrated in simulation only.

4.1 Problem 1

4.1.1 Problem description

The objective of the first problem is to analyze how precisely and accurately the airship executes a linear displacement generated by the optimizer. The simulation and the experimental results will be compared. The test verifies the accuracy of the model, the modeling, and trajectory generator. Note that there is no controller and the test were executed in an open loop manner only. A real system would need a controller, as drift and disturbances accumulate over time.

For this test, the vehicle starts immobile and has to finish immobile at a point one meter straight ahead. The initial and final conditions are shown in Table 4.1.

The final roll and pitch are not considered important for this test and are therefore considered "free". Consequently, they can take any value. In the

Table 4.1: Simulation parameter for the first experiment.

Time	x (m)	y (m)	z (m)	ϕ (rad)	θ (rad)	ψ (rad)
Initial	0	0	0	0	0	0
Final	1	0	0	free	free	0
	$\dot{x}$ (m/s)	$\dot{y}$ (m/s)	$\dot{z}$ (m/s)	p (rad/s)	q (rad/s)	r (rad/s)
Initial	0	0	0	0	0	0
Final	0	0	0	free	free	0

code, their upper and lower limits are set to a large value, such as ± 10 rad, as it relaxes the constraints on the solver. The initial conditions were input in the code and the problem was solved using PSOPT. The model used is the model presented in Section 2.3.2.

4.1.2 Simulation results

Without the propeller inertia, the solution would be very close to a simple bang-bang control, with a full forward thrust followed by a full reverse thrust [35]. However, this inertia increases greatly the complexity of the problem. In Fig. 4.1, the solid blue line represents the optimal path generated, while the base of the arrows represent the point of the thrust application. The arrows are spaced at equidistant time. The length of the arrows represent the relative magnitude of the thrust. The interpretation of the graph is an acceleration followed by a deceleration, after the airship traveled 0.8m.

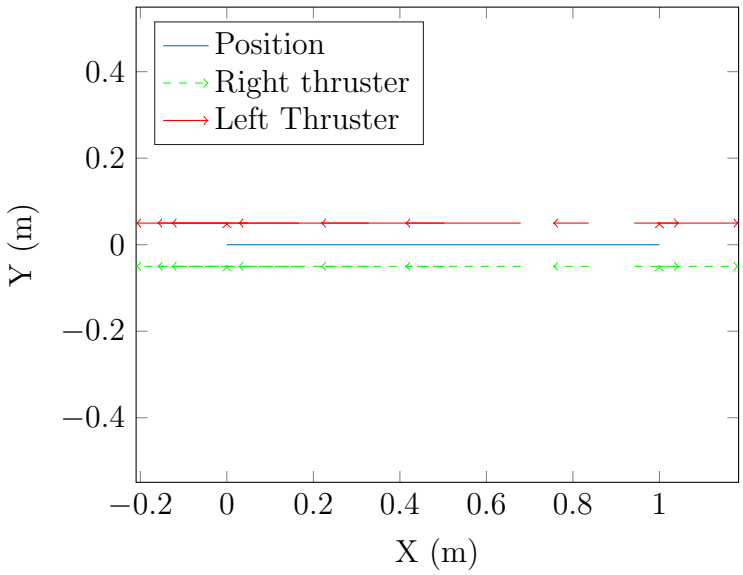


Figure 4.1: Simulated thrust and its position for the first problem.

According to the optimizer, the solution is a full power forward command followed by a full power reverse command, and, finally, a short full forward command to stop the propellers. Fig. 4.2 shows the command and the resulting simulated thrust sent to the individual motors. The command and the thrust of the left and right motors are exactly equal within the simulation's tolerance at all times during the problem. The consequence of having equal thrust on both sides is that the vehicle keeps its heading at all times during the problem.

It can be noted that the thrust from the propeller lags behind the command. This is due to the propeller inertia and the damping of the air, implemented in the model presented in the previous section. The significant lag between the command and the thrust relative to the simulation time demonstrates that a propeller model was needed for accurate simulation. If such model was not included, the results would underestimate the time required to perform a maneuver.

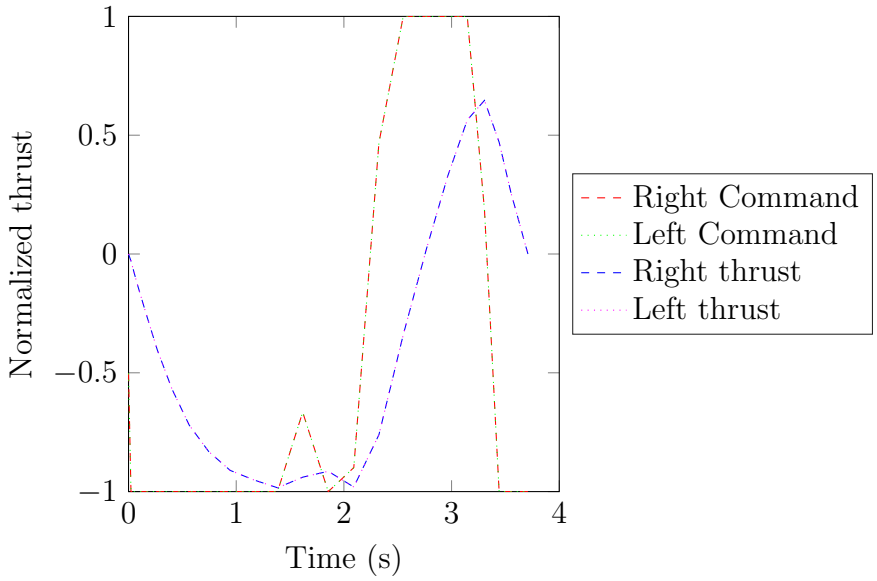


Figure 4.2: Simulated command sent to the motors.

The simulated orientation is shown in Fig. 4.3. Considering the tolerance of the modeling, the commanded yaw angle is a straight line.

4.1.3 Experimental results

The simulation command were sent to the vehicle and the trajectory was tracked. The experiment was repeated eight times and the experimental trajectories plotted alongside the simulated trajectory. Fig. 4.4 shows the position of the vehicle at 50% and 100% of the simulation time.

The heading (yaw) of the vehicle is shown in Fig. 4.5. The yaw angular acceleration is positive and the yaw finishes at +10deg. The experimental and

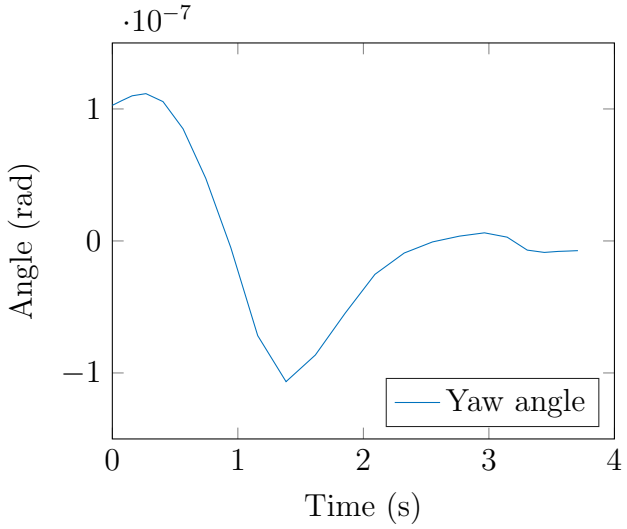


Figure 4.3: Simulated orientation as a function of time.

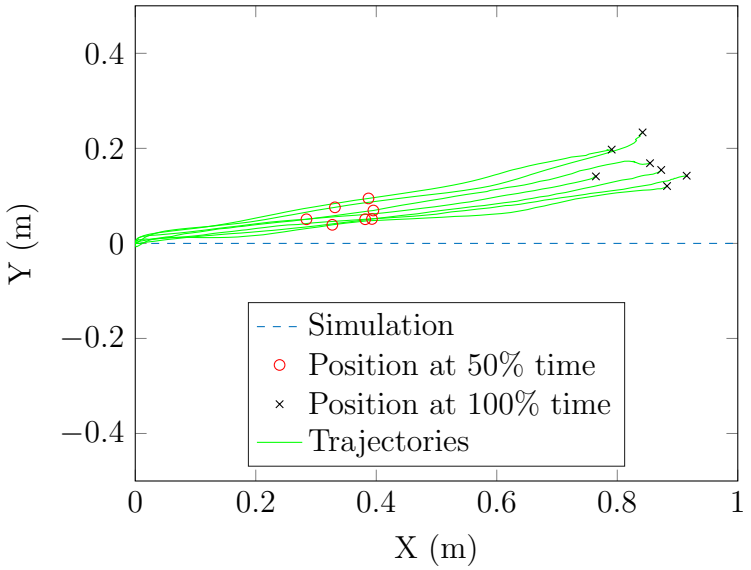


Figure 4.4: Simulated and experimental trajectories as a function of time for the linear acceleration test.

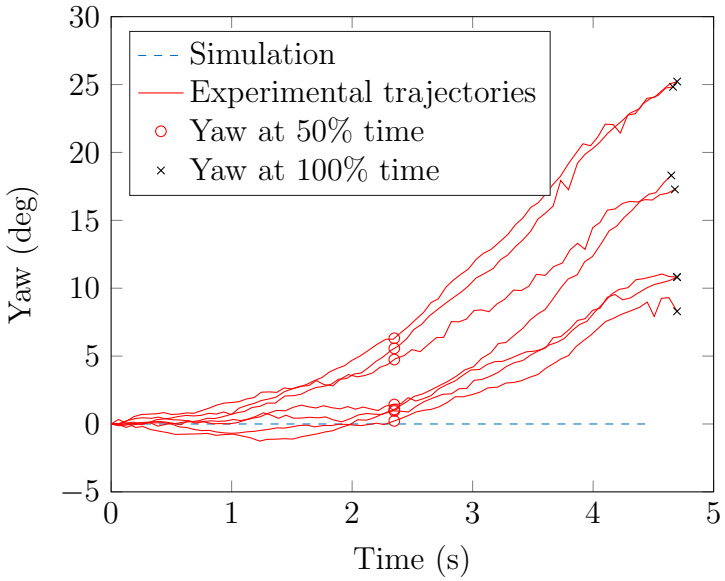


Figure 4.5: Experimental orientations as a function of time for the linear acceleration test.

simulated speeds are shown in 4.6.

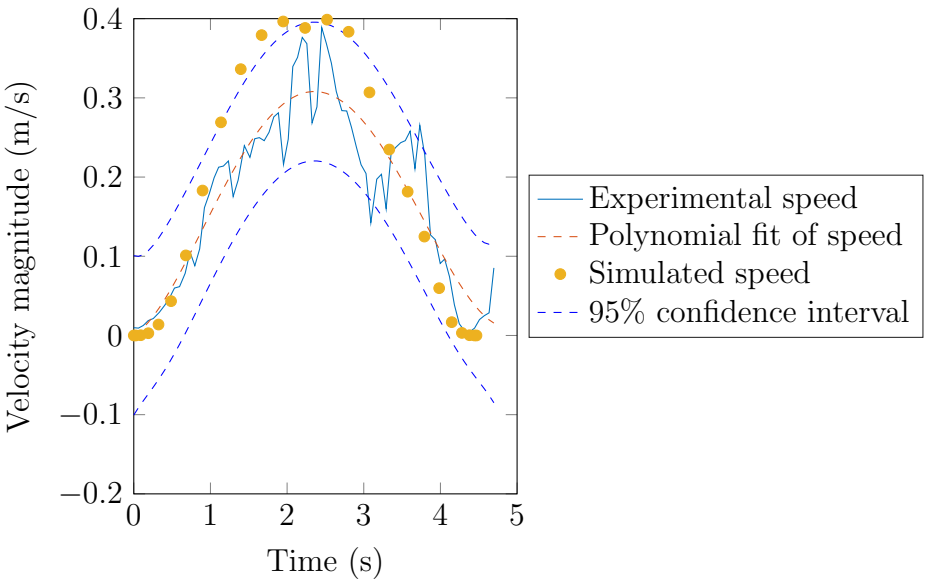


Figure 4.6: Simulated and experimental velocities as a function of time for the linear acceleration test.

On average, the error at 50 % time, after 1.6 seconds, is 3 % in the x direction and 6 % in the y direction. At the end of the experiment, after 3.8 seconds, the error is 7 % in the x direction and 15 % in the y direction. A statistical analysis of the experiment is presented in Table 4.2.

Table 4.2: Experimental results of the linear test

	50 % time			100 % time		
	Speed (m/s)	x (m)	y (m)	Speed (m/s)	x (m)	y (m)
Simulated state	0.39	0.54	0	0	1	0
Average error	-0.06	-0.03	0.06	0.07	-0.15	0.17
Standard deviation	0.05	0.04	0.02	0.04	-0.05	0.04

4.1.4 Analysis of the results

There are artifacts caused by the use of a spectral method, as seen in the command plot at around 1.5s in Fig. 4.2. In this case, it is clear that the optimal solution is, as know in the literature, a bang-bang control. Specifically, the command should be in order, full forward thrust, full reverse thrust, and full forward thrust. The first part accelerates the vehicle, the second part decelerates it, and the last part stops the propellers as quickly as possible.

Polynomials, while very accurate for continuous solutions, will show artifacts when approximating discontinuous changes, such as a bang-bang control. In this case, considering uncertainty in the model, the result is very close to the optimal bang-bang solution. In this simulation, a single phase was used. For a more accurate solution, three phases could be used. Since each phase does not require continuity, the solution could be approximated with the discontinuities properly modeled. Using multiple phases would require manual input and prior knowledge on the form of the solution.

The yaw acceleration was consistently positive. This was probably due to the right motor being stronger than the left motor. Due to the non-linear behavior of the command, it was very difficult to calibrate the motors. Consequently, due to manufacturing tolerances, one motor happened to be stronger that the other.

4.2 Problem 2

4.2.1 Problem description

The second problem demonstrates a reorientation of the airship. In this case, the reorientation is of 180 degrees. The start and end parameters are shown in Fig. 4.3. Once again, the final roll and pitch are considered 'free' as their final values have little effect on the intended maneuverer.

Table 4.3: Simulation parameter for the rotation experiment.

Time	x (m)	y (m)	z (m)	ϕ (rad)	θ (rad)	ψ (rad)
Initial	0	0	0	0	0	0
Final	0	0	0	free	free	π
	$\dot{x}$ (m/s)	$\dot{y}$ (m/s)	$\dot{z}$ (m/s)	p (rad/s)	q (rad/s)	r (rad/s)
Initial	0	0	0	0	0	0
Final	0	0	0	free	free	0

4.2.2 Simulation results

The simulated thrust and position of the airship is shown in Fig. 4.7. The graph shows that the vehicle stays immobile while the two thrusters rotate the vehicle. Fig. 4.8 shows that the command sent to each motor is exactly the

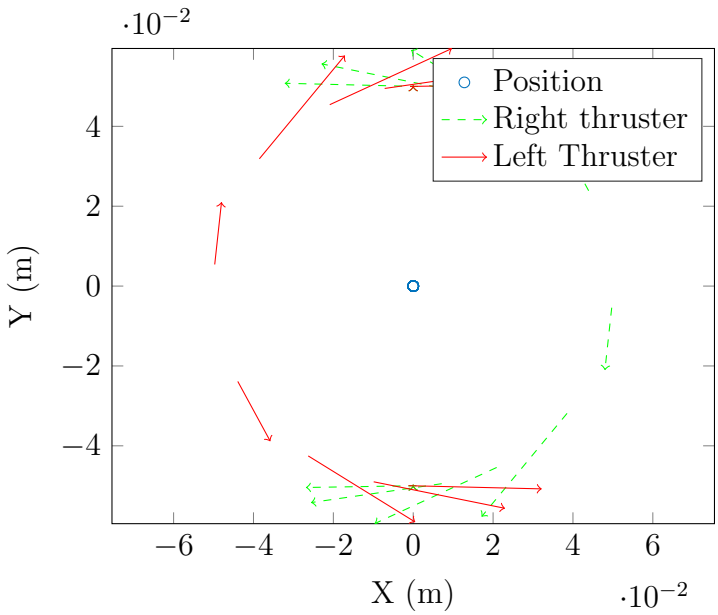


Figure 4.7: Simulated thrust and its position for the reorientation problem.

same, but opposite. This is an expected solution due to the symmetry of the problem.

4.2.3 Experimental results and discussion

The experiments were repeated multiples times and demonstrated excellent repeatability between each test for the orientation of the yaw. The standard deviation at the end of the test was 1.2 degrees for the yaw orientation, while the error was 18 degrees. The vehicle consistently overshoots, which implies

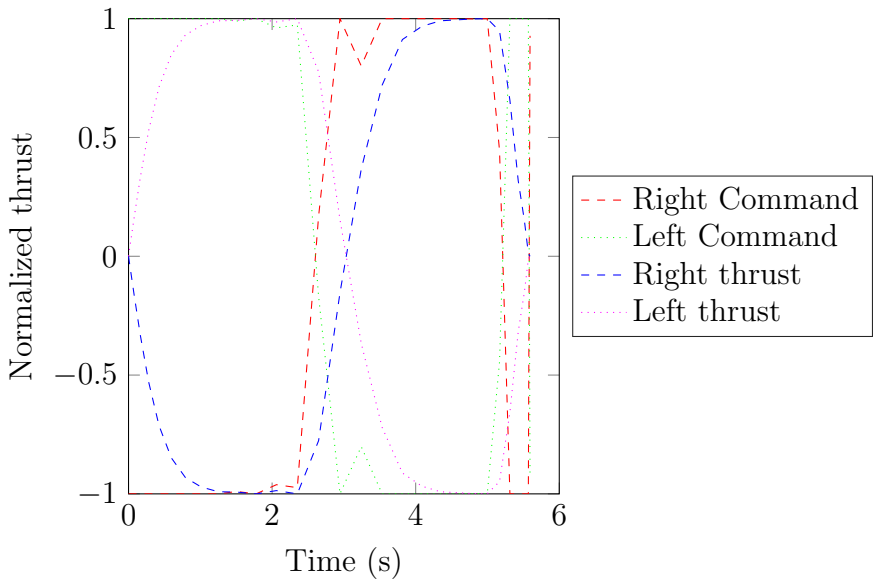


Figure 4.8: Simulated command sent to the motor for the reorientation problem.

that the rotational inertia or the rotational drag coefficient was overestimated. Table 4.4 shows the error and the standard deviation at 50 % of the time of the complete test, and at the end the test.

Table 4.4: Experimental results of the rotation test

	50 % time			100 % time		
	Yaw (°)	x (m)	y (m)	Yaw (°)	x (m)	y (m)
Simulated state	74	0	0	180	0	0
Average error	19	-0.05	-0.03	18	-0.08	-0.17
Standard deviation	1.2	0.05	0.04	3.7	-0.13	0.18

The position of the airship was drifts over time. Since the command sent to each motor was the same, but inverted, the drift in position shows that each motor has a different response to the input. Additionally, the part of the positional error may be due to the initial velocity being slightly different from zero, or due to air movement in the test room.

Fig. 4.10 shows the position of the airship during the tests. It shows that the airship will generally move to the lower left of its initial position. Fig. 4.11 shows the velocity of the airship during a test. The velocity reaches a maximum of 15 cm/s and the vehicle ends the test with a velocity of 10cm/s. The discrepancy could be reduced with a controller, an individualized motor model, and an accurate representation of the wind velocity around the airship. While the wind velocity is difficult to estimate for small airships, larger airships can use wind data and on board sensors to estimate the local air velocity.

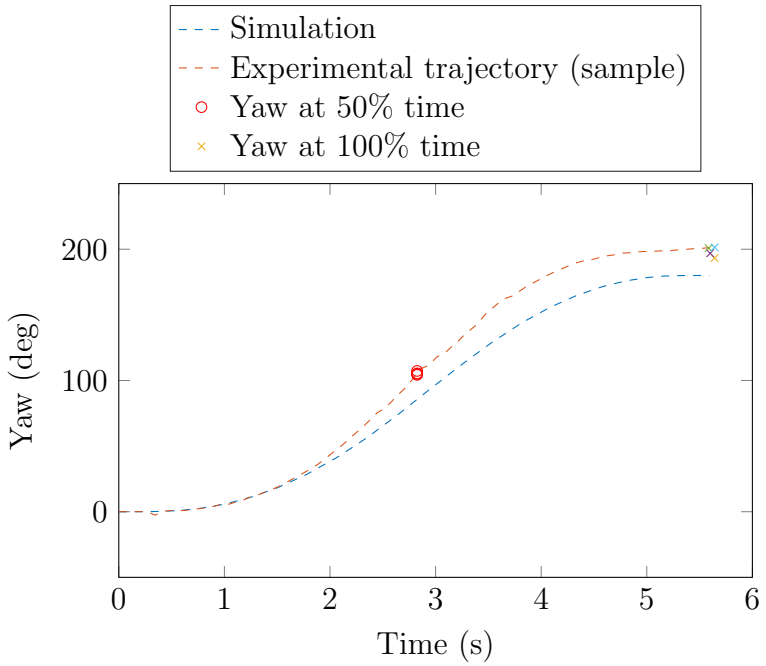


Figure 4.9: Experimental orientation as a function of time for the rotation acceleration test.

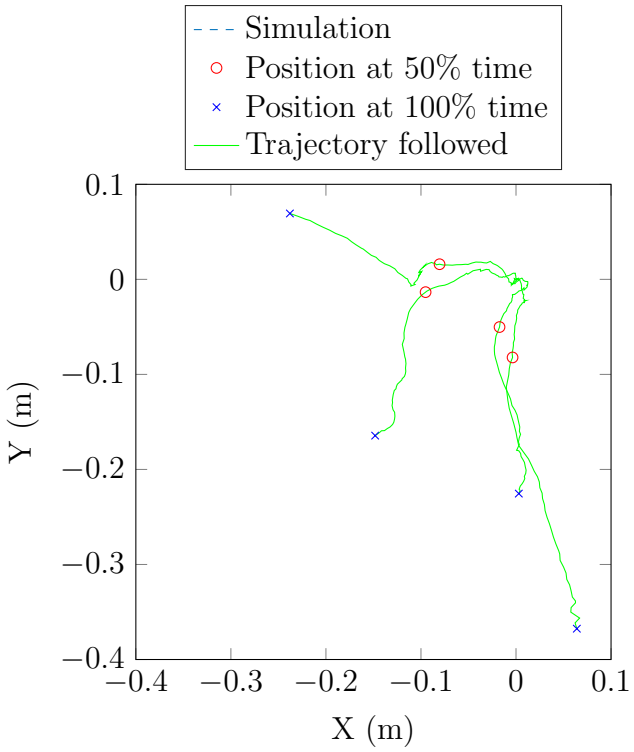


Figure 4.10: Simulated and experimental trajectories as a function of time for the rotation acceleration test.

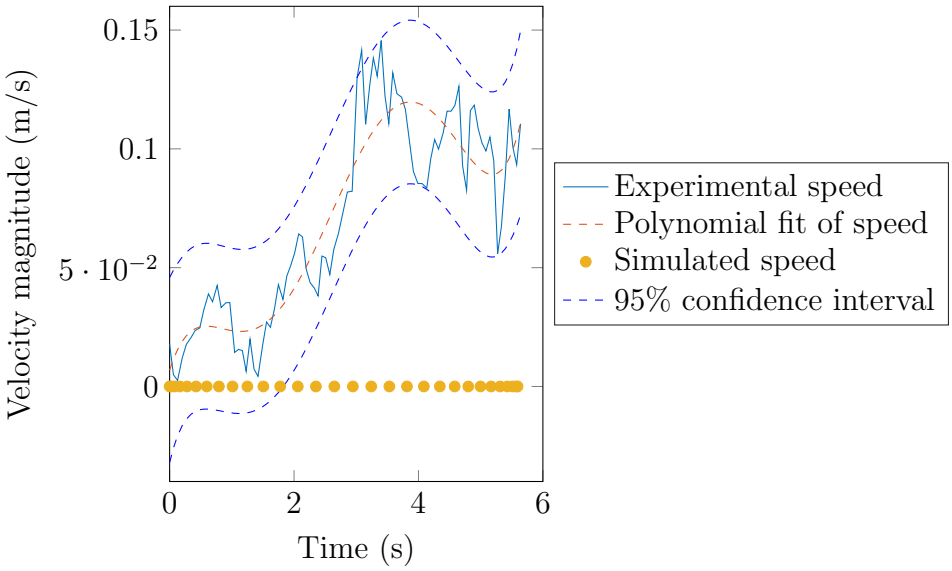


Figure 4.11: Simulated and experimental velocities as a function of time for the rotation acceleration test.

4.3 Advanced simulations

The first two problems demonstrated simple trajectories. However, the optimal control solver can handle much more complex problems. The limitation for practical experiments is that the drift causes an error between the trajectory to follow and the actual path. This drift is caused by the airflow in the test room, imperfection in the motors, and inaccuracy in the model. Therefore, in these problems, a controller must be used. Consequently, more complex trajectories will be demonstrated in simulation only.

4.3.1 Problem 3

The third problem demonstrates a case where the vehicle moves and reorients itself. The simulation parameters are shown in Fig. 4.5.

Table 4.5: Simulation parameter for the third experiment

Time	x (m)	y (m)	z (m)	ϕ (rad)	θ (rad)	ψ (rad)
Initial	0	0	0	0	0	0
Final	1.5	0.5	0	0	0	1
	$\dot{x}$ (m/s)	$\dot{y}$ (m/s)	$\dot{z}$ (m/s)	p (rad/s)	q (rad/s)	r (rad/s)
Initial	0	0	0	0	0	0
Final	0	0	0	0	0	0

The resulting trajectory with the relative thrust orientation and magnitude is shown in Fig. 4.12, while the command sent to the thrusters as a function of time is shown in Fig. 4.13a.

The vehicle accelerates in a straight line in the first second of the trajectory. As seen in Fig. 4.13b, the vehicle points outside the curve and reverses its thrust to reorient the vehicle's velocity vector toward the final position. The final part of the path is in a straight line, where the airship accelerates and decelerates to finish in the position and orientation desired. The normalized thrust for each motor is shown in Fig. 4.13b. This kind of optimal maneuver would be very difficult to generate with traditional linear controller. The dynamical model has to be taken into account to minimize the time: the optimal solver uses the inertial of the vehicle.

4.3.2 Problem 4

The fourth simulation demonstrates how lateral movements are possible with the airship even though it is under-actuated. The airship is commanded to

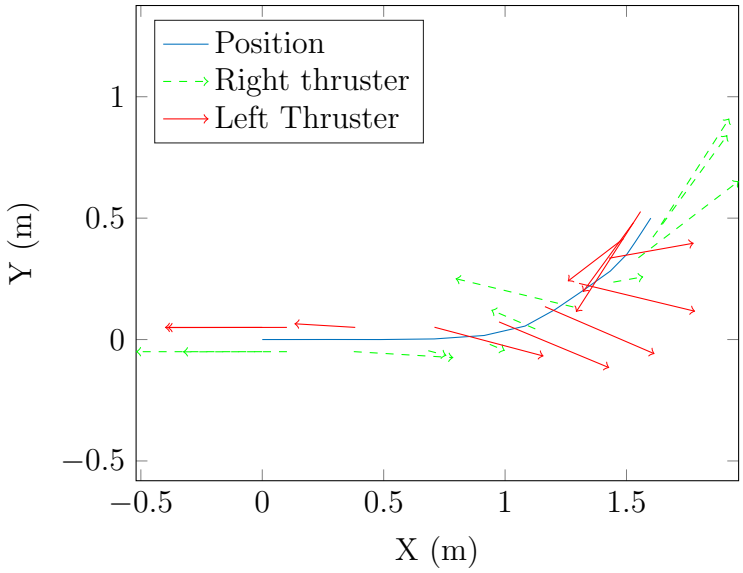
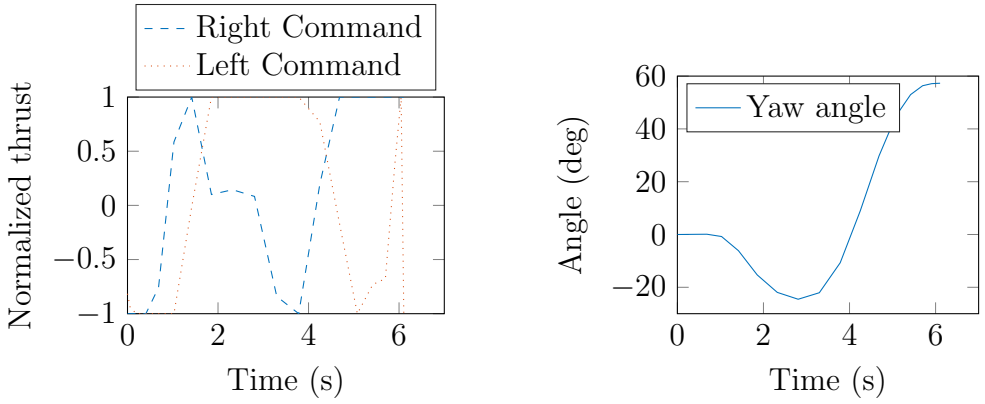


Figure 4.12: Path of the vehicle and thrust of the right (green) and left (red) motors at different positions along the path.



(a) Left and right thrusts as a function of time.

(b) Yaw orientation of the vehicle as a function of time.

Figure 4.13: Thrust and yaw angle as a function of time for the third experiment.

finish the simulation to the left of the starting point, in the same orientation, but with a lateral speed. This kind of maneuver could be useful for storing the airship in a similar manner to parallel parking for cars. Due to the lateral movement, generating this path requires a path planner taking into account the dynamics of the airship. The initial and final conditions for the fourth problem are presented in Table 4.6. The trajectory generated is shown in Fig. 4.14, the commands sent to the motor are shown in Fig. 4.15a, and the orientation of the vehicle as a function of time is shown in Fig. 4.15b.

In the Fig. 4.14, the vehicle is shown accelerating, then turning to its

Table 4.6: Simulation parameters for the fourth experiment

Time	x (m)	y (m)	z (m)	ϕ (rad)	θ (rad)	ψ (rad)
Initial	0	0	0	0	0	0
Final	0	0.3	0	0	0	0
	$\dot{x}$ (m/s)	$\dot{y}$ (m/s)	$\dot{z}$ (m/s)	p (rad/s)	q (rad/s)	r (rad/s)
Initial	0	0	0	0	0	0
Final	0	0.1	0	0	0	0

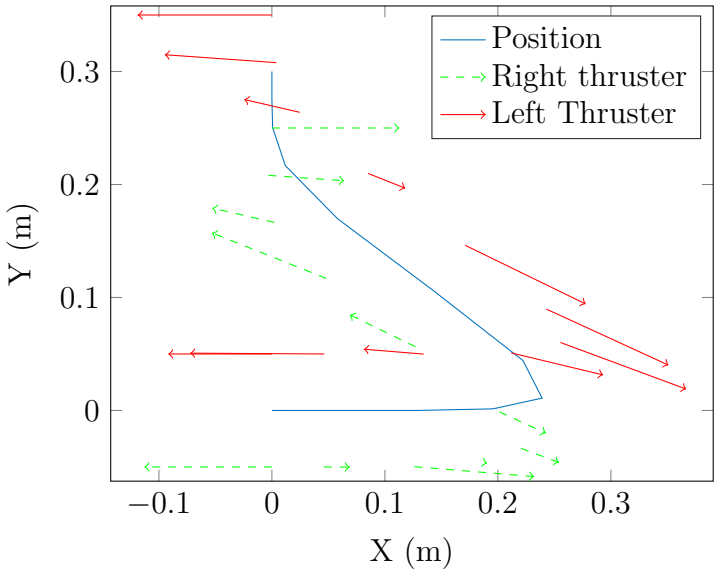
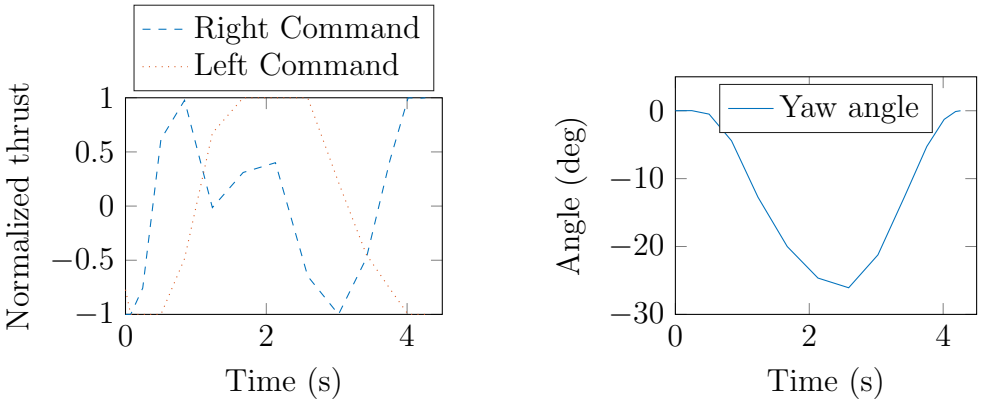


Figure 4.14: Path of the vehicle and thrust of the right (green) and left (red) motors at different positions along the path.

right and reversing thrust. This maneuver causes the vehicle to gain lateral velocity. In the last part of the path, the vehicle cancels out its velocity in the x direction and orients itself to 0 radians, while keeping a velocity in the y direction. The air resistance slows down the vehicle to a rest position.



(a) Left and right thrust. as a function of time.

(b) Yaw orientation of the vehicle as a function of time.

Figure 4.15: Thrust and yaw angle as a function of time for the fourth experiment.

Chapter 5

Discussion of the simulations and experimental tests

5.1 Error analysis

The model is not perfectly accurate; consequently, the vehicle drifts due to small errors in the model and due to external disturbances. Errors in the modeling will cause differences in the acceleration time, the time of arrival, the terminal speed and other characteristic of the airship. In addition, some of the model assumptions are simplifications. For instance, with real propellers, the thrust is function of airspeed, which was neglected. The drag is based on a very simple model, not taking into account the angle of attack or the change in Reynold number when the speed increases, which means that the drag is not perfectly proportional to the square of the velocity. Since the airship is not a rigid hull, there is also deformation of the helium's envelop, changing the dynamical characteristics. The thrust of the nacelle was not symmetric, as shown by the rotation and linear acceleration test. The structure itself is not perfectly rigid, as it is supported by helium. Consequently, the mass matrix and the orientation of the thrust may change during acceleration.

Some errors were caused by measurement system. The resolution of the camera is limited, which causes a finite precision of the position. The position measurement accuracy is limited by the resolution of the sensor. It was observed that the noise is lower in the plane parallel to the camera sensor than in the direction perpendicular to it. The former one relies on the position of the marker, which can be latter one relies on the size of the marker. Consequently, the camera was placed directly above the marker to reduce the noise. The noise is due to the noise in the image and to imprecise corner recognition of the software. There are also constant turbulences in the experiment room due to ventilation, convection, and movement. Even under no power, the vehicle drifts fairly quickly due to these turbulences.

Nonetheless, a controller has to be used for real airships. Additionally to a controller, it may be necessary to re-plan the path using the optimal solver when significant error has accumulated. The dynamical model can be updated to better reflect the changing conditions, such as new wind conditions, changing mass, or even a broken actuator. The errors are due to multiple factors.

5.2 Limitations

Convergence was an issue. Due to the very wide search space, the solution is not guaranteed to converge, especially if the initial guess is poor. To help convergence, the problem is resolved in multiple steps. The solver initially finds a solution with a low number of points, which corresponds to a low order of Legendre polynomials. The low resolution solution is then used as a guess for a higher order of polynomial approximation. The process is repeated until the desired resolution is achieve. For reference, in the first two problems, the solver started with 8 nodes, followed by 9, 15, 19 and 25 nodes.

An initial guess has to be provided. The optimizer starts from the guess and modify it to respect the differential constraints while minimizing the performance index. For all the experiments, a linear interpolation between the initial and final conditions is used as a guess for every state, and the controls are guess to be zero at all time. This initial guess was sufficient for the problem used. In more complex cases, a refined guess may be necessary, especially if the solution diverges significantly from the guess. The solver may fail to converge, which is unacceptable for real time application if the optimal solver is the only path planner used on an airship. When the solution can be planned in advance, manually changing the number of points used for each step of the solving can help. The solver work by solving the problem with a low order polynomial first, and then it uses this solution as a guess for a higher order polynomial. For example, instead of solving the problem in two steps with 8 and 20 nodes, four steps of 6, 12, 16 and 20 nodes could be used. The solving time increases, but there is more chances of convergence.

In the case of the problem studied, part of the convergence issues were due to the sharp discontinuities in the solution. Attempts to solve directly with 25 nodes failed. Consequently, to use a pseudo-spectral solver on a real flying vehicle, a method for generating a guess be required. This would have two advantages: first, if the optimal solver does not converge, the guess can be used by the path follower as a less optimal solution. This way, the vehicle will not be left without a path. The method for generating the guess can be a discrete path planner using a grid search algorithm. Second, the guess can be used by the optimal control solver, which helps convergence.

As explained earlier, an issue with spectral solvers is that discontinuities are approximated with error, as polynomials have to be continuous. Around

discontinuities, artifacts appeared in the solution, as seen at 3.5 seconds in Fig. 4.8. Considering the error already present in the model, those artifacts are small. It has been proved that convergence is possible even in the presence of discontinuities [36]. In practice however, a very large number of nodes will be required for a good approximation of discontinuities, which may take a long time to solve. If a very precise solution is required, multiple phases would be required. The controls don't have to be continuous between two phases, which allows discontinuities. The problem is then to guess how many phases are required for the optimal solution. This step can be performed manually. Future work could include automatic recognition of the discontinuities to allow solving automatically for use in embedded systems.

5.3 Comparison of simulation and experimental results

Over the period of time studied, the experimental model showed drift. This is due to the open loop nature of the control. An open loop control was necessary to evaluate the performance of the path planner without interference from a controller. The main problem with the experimental model was a constant drift toward the left. This is due to a difference in the right and left motor thrust, as well as the aerodynamic dissymmetry of the vehicle. There are multiple way of solving this problem. The command could be changed to make both motors match thrust. Another possibility is to use small fins to correct the orientation during forward movement. The fins would change the aerodynamic characteristics of the airship, which would required changes to the model. Finally, the controller could also correct directly for the error in the model. The experiment of reorientation showed lateral movement for every test. Those movements are probably partly due to the turbulence, but an imbalance in thrust would also cause them.

5.4 Solving time

The solving time was between 1s and 30s on an Intel i7 2600 processor for the problems presented. The main factor affecting the convergence time was the number of node used for solving as well as the number of steps. Many parameters affect the time of convergence. The time can be affected due to technical details of implementation as many code libraries used for solving. In the experiment performed, it was found that the number of nodes was the most important parameter affecting the time to solve the problem. A poor initial guess also affected the time of convergence for the first step of solving. It is possible to change the tolerance accepted by the solver, which may improve convergence time. However, the tolerance accepted has a relatively

small impact due to the exponential rate of convergence of the pseudo-spectral method [32].

5.5 Optimal control solver for airships

The optimal solver used generates very complex paths locally optimal, and respecting the differential equations. An important advantage of the optimal solver is the flexibility it offers, as almost any dynamical equation can be used. For instance, a propeller model taking into account the inertia and the drag of the propeller could be added by adding states and dynamical constraints. The solver and problem formulation would not change.

Given the strengths and limitations of the optimal solver, the results could be used in real applications to precalculate specific open loop maneuvers. For instance, parts of the landing and take-off procedure could be open loop, and the optimal solver could be used to generate the command. There could also be precalculated commands to reorient the airship quickly. The optimal control solver can also be used to verify the result of other path planners that use approximations, discretization, genetic algorithms, or other. An important advantage of an optimal path planner is that it is very flexible and updates to the model are easy. For example, adding a second airship in the simulation is simply a matter of duplicating all the states, controls and dynamic constraints, as well as adding a path constraint to prevent collisions.

Chapter 6

Conclusion

6.1 Key findings

Airships are complex dynamical systems. The modeling of a small airship was covered theoretically and experimentally. The experimental modeling successfully included a technique to model drag, virtual mass and the response of the thrusters. This thesis demonstrated that pseudo-spectral optimal control solvers can be used to optimize small airship trajectories in complex environments. Simple solutions we computed and used to control a small airships. The experimental results matched the simulations results with a quantified and explained error. Finally complex maneuverer were computed and presented. The results of this thesis may find applications in research, to compare optimal solutions to simpler models, and in helping to compute complex maneuver such as landing, take-off and parking.

6.2 Future work

The objective of this work was to generate an optimal trajectory for a small airship and create a good dynamical model of the experimental platform. As demonstrated by the results, in open loop, the airship will diverge from its trajectory. The solution to the accumulating error is a controller. Although a linear controller might work for simple trajectories, the non-linear dynamics of the system indicate that a more advanced controller would be required. Dynamic re-planning may also be necessary if the environment changes significantly from the initial assumptions.

The airship model used proved useful for indoor testing in a research environment. Improvements could be made to the mechanical system to facilitate tests. The most significant problem discovered is that the voltage changes significantly after a few minutes of tests. Consequently, tests were all done

with a fully charged battery. It would be possible to correct for this by monitoring the battery on-board and sending a command to the motor function of the voltage. This implementation would be very difficult with the tested off-the-shelf airship, which means a custom airship would have to be made. The second improvement is on the characterization of the propeller. Since the motors used are very small DC motors, it is difficult to control their speeds, and dust or dirt significantly affects their performance. To ensure repeatable tests, small brushless motors may be preferable.

References

[1] J. Waishek, A. Dogan, and Y. Bestaoui, "Investigation into the time varying mass effect on airship dynamics response," in *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition*, no. January, 2009.

[2] J. Rao, Z. Gong, J. Luo, and S. Xie, "Unmanned airships for emergency management," in *Proceedings of the IEEE International Safety, Security and Rescue Robotics Workshop*, 2005, pp. 125–130.

[3] A. Elfes, S. S. Bueno, M. Bergerman, J. A. J. G. Ramos, and S. A. B. V. Gomes, "Project AURORA: Development of an Autonomous Unmanned Remote Monitoring Robotic Airship," *Journal of the Brazilian Computer Society*, vol. 4, 1998.

[4] S. Bueno, J. Azinheira, and J. R. Jr, "Project AURORA: Towards an autonomous robotic airship," *Institutional Repository of the Center for Information Technology Renato Archer*, 2010.

[5] J. S. Cummings, "Long Endurance Multi-Intelligence Vehicle (LEMV) Agreement Signed," 2010.

[6] H. Chao, Y. Cao, and Y. Chen, "Autopilots for small unmanned aerial vehicles: A survey," *International Journal of Control, Automation and Systems*, vol. 8, no. 1, pp. 36–44, 2010.

[7] G. Huang, Y. Lu, and Y. Nan, "A survey of numerical algorithms for trajectory optimization of flight vehicles," *Science China Technological Sciences*, vol. 55, no. 9, pp. 2538–2560, 2012.

[8] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 57, no. 1-4, pp. 65–100, 2010.

[9] W. H. Al-sabban, L. F. Gonzalez, R. N. Smith, and G. F. Wyeth, "Wind-Energy based Path Planning For Electric Unmanned Aerial Vehicles Using Markov Decision Processes," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[10] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems*, 2005.

[11] B. Y. P. Bhattacharya and M. L. Gavrilova, "Roadmap-Based Path Planning," *IEEE Robotics & Automation magazine*, no. June, 2008.

[12] C. E. Thorpe, "Path Relaxation: Path Planning for a Mobile Robot," in *Association for Avancement of Artificial Intelligence*. Washington, DC: IEEE, 1984, pp. 318–321.

[13] S. Recoskie, "Autonomous Hybrid Powered Long Ranged Airship for Surveillance and Guidance," Ph.D. dissertation, University of Ottawa, 2014.

[14] W. Adamski, P. Herman, Y. Bestaoui, and K. Kozłowski, "Control of airship in case of unpredictable environment conditions," in *Conference on Control and Fault-Tolerant Systems, SysTol'10 - Final Program and Book of Abstracts*, 2010, pp. 843–848.

[15] J. Kim and J. Hespanha, "Discrete approximations to continuous shortest-path: application to minimum-risk path planning for groups of UAVs," in *Decision and Control*, 2003.

[16] J. a. S. J. a. Sorensen, S. a. M. S. a. Morello, and H. E. H. Erzberger, "Application of trajectory optimization principles to minimize aircraft operating costs," *1979 18th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, vol. 18, pp. 415–421, 1979.

[17] Y. Zhuang, G. Ma, H. Huang, and C. Li, "Real-time trajectory optimization of an underactuated rigid spacecraft using differential flatness," *Aerospace Science and Technology*, vol. 23, no. 1, pp. 132–139, 2012.

[18] L. Haws and T. Kiser, "Exploring the Brachistochrone Problem," *The American Mathematical Monthly*, vol. 102, no. 4, 1995.

[19] B. R. Geiger, J. F. Horn, A. M. Delullo, and L. N. Long, "Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming," in *American Institute of Aeronautics and Astronautics GNC conference.*, vol. 2, 2006.

[20] O. von Stryk and R. Burlirsch, "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operations Research*, vol. 37, pp. 357–373, 1992.

[21] A. Bryson and W. Denham, "A Steepest-Ascent Method for Solving Optimum Programming Problems," *Journal of Applied Mechanics*, vol. 29, p. 447, 1962.

[22] R. H. Goddard, *A Method of Reaching Extmeme Altitudes.* Baltimore, MD, U.S.A.: Smithsonian Institution, 1919.

[23] Y. J. Zhao, W. Garrard, and J. Mueller, "Benefits of Trajectory Optimization in Airship Flights," in *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, no. September, Chicago, 2004.

[24] T. C. Flanzer, G. C. Bower, and I. M. Kroo, "Robust Trajectory Optimization for Dynamic Soaring," in *AIAA Guidance, Navigation, and Control Conference*, 2012, pp. 1–22.

[25] K. Mohan, M. Patterson, and A. V. Rao, "Optimal Trajectory and Control Generation for Landing of Multiple Aircraft in the Presence of Obstacles," *Guidance, Navigation, and Control Conference*, no. August, pp. 1–16, 2012.

[26] Y. Liu, Y. Zhang, and Y. Hu, "Optimal path planning for autonomous airship based on clonal selection and direct collocation algorithm," in *Proceedings of 2008 IEEE International Conference on Networking, Sensing and Control, ICNSC*, 2008, pp. 1828–1832.

[27] S. Lee and H. Bang, "Three-Dimensional Ascent Trajectory Optimization for Stratospheric Airship Platforms in the Jet Stream," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1341–1351, Sep. 2007.

[28] E. Hygounenc, I.-K. Jung, P. Souères, and S. Lacroix, "The Autonomous Blimp Project of LAAS-CNRS: Achievements in Flight Control and Terrain Mapping," *The International Journal of Robotics Research*, vol. 23, pp. 473–511, 2004.

[29] S. B. V. Gomes and J. J. R. Ramos, "Airship dynamic modeling for autonomous operation," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, vol. 4, no. May, 1998, pp. 3462–3467.

[30] D. T. Liesk and P. M. Nahon, "An investigation of the equations of motion of an airship in a wind field," *Control*, no. July, pp. 24–29, 2011.

[31] D. R. Yoerger, J. G. Cooke, and J.-J. E. Slotine, "The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 167–178, 1990.

[32] V. M. Becerra, "PSOPT Optimal Control Solver User Manual (Release 3)," p. 421, 2011.

[33] A. V. Rao, D. a. Benson, C. Darby, M. a. Patterson, C. Francolin, I. Sanders, and G. T. Huntington, "Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Psendospectral Method." *ACM Transactions on Mathematical Software*, vol. 37, no. 2, pp. 1–39, 2010.

[34] D. G. D. Bernard Legras, "A Comparison of the Contour Surgery and Pseudo-spectral Methods," *Journal of Computational Physics*, vol. 104, no. 2, pp. 287–302, 1993.

[35] L. C. Evans, "An Introduction to Mathematical Optimal Control Theory Version 0.2." [Online]. Available: https://math.berkeley.edu/~evans/control.course.pdf

[36] W. Kang, Q. Gong, I. M. Ross, and F. Fahroo, "On the convergence of nonlinear optimal control using pseudospectral methods for feedback linearizable systems," *International Journal of Robust and Nonlinear Control*, vol. 17, no. 14, pp. 1251–1277, 2007.

APPENDICES

Appendix A

Software Libraries

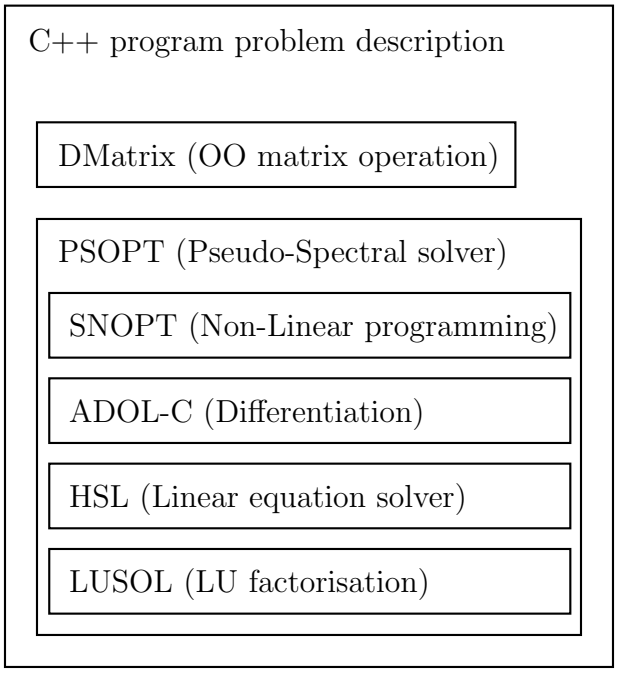


Figure A.1: Program description

The problem is described in C++ and the libraries used are shown in Fig. A.1. Multiple objects are defined by PSOPT and modified to describe the problem. The library DMatrix, which uses heavy operation overloading, facilitates the manipulation of matrices. PSOPT also has helper functions to simplify the problem description. The objects describing the problem are passed to the optimizer. Internally, the software will transform the problem into a non-linear programming problem (NLP). NLP is the process of solving an optimization problem constrained by inequalities. Solving the problem involves evaluating a large number of derivatives due to the dynamical constraints. Those derivatives are solved using ADOL-C. The NLP problem is

solved by SNOPT, which transforms the problem into a linear problem involving large matrices. Those matrices are solved using the HSL library and LUSOL for LU factorisation.