Advanced Space Concepts Laboratory
Department of Mechanical and Aerospace Engineering
Faculty of Engineering
University of Strathclyde

Trajectory generation and tracking for drone racing

Jonathan Jamieson

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

2018

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Jonathan Jamieson
Glasgow, Scotland, 19/01/2018

Acknowledgements

I would like to thank my supervisors: James Biggs, Malcolm Macdonald and Ivan Glesk. Dr Biggs proposed the project and has supported me throughout, even after getting a new job at Politecnico di Milano. His guidance and feedback were invaluable. I am grateful to Dr Macdonald who agreed to become my first supervisor halfway through my PhD. The meetings with his group were always enjoyable and generated many good ideas. I appreciate the advice from Professor Glesk who supported the project from the beginning. Yuliang Bai from Harbin Institute of Technology was very helpful during both of my visits to Milan. Both viva examiners, Dr Dave Anderson and Professor Matthew Cartmell, gave very helpful corrections that improved the thesis. I would also like to thank the other academic and administrative staff in the department for the help and assistance they have given.

Graham Murdoch and Lindsey Whiteford work hard to provide the facilities at the Fab Lab in Strathclyde University. I would like to thank both for showing me how to use the machines and occasionally fixing them after I have made a mistake.

I have been very fortunate to have friends, colleagues and domingueros who are keen for lots of activities. I doubt there has been a generation of postgraduate students with so many outdoor enthusiasts! My family has always encouraged any interests I have, be they academic or pleasure and I would like to thank them for their support throughout my life.

There are few things like a doctorate. Being paid to perform research on my passion was an opportunity I could not miss and I am very glad that I didn't.

Abstract

In this thesis trajectory generation for quadrotors, a type of rotorcraft UAV (Unmanned Aerial Vehicle), is considered with two different methods. The first applies the Maximum Principle of optimal control to derive closed-form analytical functions that describe the translational states for two different cases of nonholonomic constraints. Parametric optimisation is used to find the trajectories. Reachable sets are found numerically and a simple obstacle avoidance method is demonstrated for both cases.

The second motion planning method found trajectories with polynomial basis functions that are parametrised by an abstract function between zero and one. This virtual time domain trajectory satisfied conditions placed on the boundary derivatives and followed a sequence of desired waypoints. A process for finding a mapping function that converts the virtual domain trajectory into one on the standard time domain is developed to minimise the trajectory time whilst ensuring the motion remained feasible by enforcing bounds on the thrust required from each rotor.

An algorithm that uses additional waypoints where necessary to ensure the trajectory does not collide with the gates that define the course is developed. A method for minimising the accumulated angular acceleration of the heading angle whilst remaining within a desired tolerance of the velocity vector angle is also described.

Trajectory tracking is considered by modifying an existing quadrotor tracking controller on the Special Euclidean group SE(3) to include a Linear Extended State Observer that estimates and counteracts translational disturbances. The modified controller is shown to reduce the position tracking error in the presence of square wave, sinusoidal and wind disturbances.

Contents

List of Symbols and Acronyms

Constants and Variables

$\beta_1, \ \beta_2$	LESO gains
$\mu_1, \ \mu_2, \ \mu_3, \ \mu_4, \ \mu_5, \ \mu_6$	Extremal curves
$\hat{\Omega}$	Skew symmetric matrix of the angular velocity (rad/s)
θ	Heading Angle (rad)
λ	Mapping Function
ω_c	LESO bandwidth
ϕ_1, ϕ_2, ϕ_3	Euler angles in drone racing reference frame
$\phi_1^s, \phi_2^s, \phi_3^s$	Euler angles in sub-Riemannian curve reference frame
ρ_a	Density of air (kg/m^3)
σ	WSA waypoint angle (rad)
τ	Virtual domain time
χ	B-spline coefficient vector
γ	Disturbance (m/s^2)
κ	B-spline knot vector
$\boldsymbol{\nu^s}$	Translational body velocity in sub-Riemannian curve reference frame (m/s)
$\boldsymbol{\Omega^s}$	Angular body frame velocities in sub-Riemannian curve reference frame
$\boldsymbol{\Omega}$	Angular body frame velocities in drone racing reference frame
$\boldsymbol{\Xi}$	Optimisation vector
$\mathbf{b_1^s}, \mathbf{b_2^s}, \mathbf{b_3^s}$	Body frame vectors in sub-Riemannian curve reference frame
$\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}$	Body frame vectors in drone racing reference frame

$\mathbf{c_a}$	Polynomial condition vector
$\mathbf{D}$	Disturbance (N)
$\mathbf{e^s}$	Sub-Riemannian curve reference frame
$\mathbf{e_v}$	Velocity tracking error (m/s)
$\mathbf{e_x}$	Position tracking error (m)
$\mathbf{e}$	Drone racing reference frame
$\mathbf{M}$	Moment (Nm)
$\mathbf{p_a}$	Polynomial coefficient vector
$\mathbf{R^s}$	Rotation matrix in sub-Riemannian curve reference frame
$\mathbf{R}$	Rotation matrix in drone racing reference frame
$\mathbf{v^s}$	Translational velocity in sub-Riemannian curve reference frame (m/s)
$\mathbf{v}$	Translational velocity in drone racing reference frame (m/s)
$\mathbf{x^s}$	Translational position in sub-Riemannian curve reference frame (m)
$\mathbf{x}$	Translational position in drone racing reference frame (m)
A_1, A_2, A_3	Infinitesimal motions in roll, pitch and yaw directions
B_1, B_2, B_3	Infinitesimal motions in surge, sway and heave directions
c^s	Angular velocity curve weighting
$c_{\kappa f}$	Yaw due to the rotation of a rotor
d	Rotor distance (m)
E_1, E_2, E_3	Basis elements of SO(3)
e_Ω	Angular velocity tracking error (rad/s)
e_R	Attitude tracking error
f	Scalar net thrust (N)
f_1, f_2, f_3, f_4	Scalar rotor thrusts (N)
g^s	Special Euclidean group of motions
g_a	Gravitational acceleration (m/s^2)
H	Hamiltonian
$I_{3\times3}$	Identity matrix

J	Cost function
j	Scalar weighting for a cost function
J_i	Moment of inertia (kg m^2)
k_x, k_v, k_R, k_Ω	Scalar gains for position, velocity, attitude and angular velocity
l_h	Gate length (m)
L_S, L_C, L_H	Layout for Standard, Cross and H-Shape configurations
m	Mass (kg)
m^s	Translational velocity curve weighting
M_a	Polynomial linear equation matrix
r_o, $r_{h,i}$, $r_{h,o}$	Radius of obstacle, inner gate and outer gate respectively (m)
t	Time (s)
u	Spline segment number

Acronyms

ADRC	Active Disturbance Rejection Control
ESO	Extended State Observer
FPV	First Person View
GPS	Global Positioning System
LESO	Linear Extended State Observer
MAV	Miniature Air Vehicle
RRT	Rapidly-exploring Random Tree
SE(3)	Special Euclidean Group
SLAM	Simultaneous Localisation and Mapping
SO(3)	Special Orthogonal Group
UAV	Unmanned Aerial Vehicle
WSA	Waypoint Selection Algorithm

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) or Miniature Air Vehicles (MAVs) are becoming increasingly popular with many new applications made possible by the development in technology. It is still common for a pilot on the ground to control the vehicle at some or all stages of a mission but a truly autonomous system should be capable of deciding where to go and how to get there before successfully carrying out that plan. This thesis considers these requirements by investigating the motion planning and trajectory tracking problems. Drone racing is the chosen application because it is an exciting new sport that is currently in rapid development and has had little consideration from the academic literature. However, there are other applications such as package delivery, search & rescue and security in which the techniques could prove useful.

The challenging nature of the sport ensures a steep learning curve for humans and currently no autonomous drone exists that comes close to matching what a skilled pilot can achieve. However, developments in computer vision, differential GPS, inertial tracking systems and other technologies are soon likely to allow the types of aggressive trajectories performed by humans to be generated and tracked autonomously. Different applications have different criteria for rating the optimality of a trajectory such as energy efficiency, minimising snap (the fourth derivative of position with respect to time) or minimising time. The objective in this thesis is to minimise time because in the context of drone racing, maximising

a vehicle's dynamic capabilities to finish the course as fast as possible means the vehicle is more likely to win the race. It is also necessary to avoid collisions with obstacles in the environment. In the context of drone racing, gates that the vehicle must fly through can be considered as obstacles and it is important that the trajectory generation method is capable of handling them. Drone racing is often carried out in wide, open spaces with fewer obstacles, than for instance, an urban environment. However, it is important that a drone closely follows its intended path in order to avoid collisions, such as when flying through the narrow gates mentioned previously. A tracking controller is required to achieve this and must be capable of handling any disturbances such as wind by minimising the deviation from the planned course. There are few scenarios in which there is no risk of colliding with obstacles so good trajectory planning and tracking can be considered universally important.

1.2 An overview of quadrotors and multirotors

Quadrotors are the most common type of multirotor, a general term for rotorcraft vehicles with two or more propellers that don't require wings for lift. Multirotors are controlled by varying the thrust generated by each rotor which is typically done by changing the motor speed. Quadrocopters, quadcopters or even the shortened form 'quads' are common names often used for multirotors with four propellers. To avoid confusion, in this thesis 'quadrotor' shall be be used throughout when referring to multirotors with four propellers.

1.2.1 A brief history of multirotor flight

Soon after the first heavier-than-air powered aircraft –the Wright Flyer– was flown in 1903 [1], attempts were made to fly rotary wing aircraft such as the Breguet-Richet Gyroplane designed by Louis Breguet and tested in 1907 [2]. However, even by 1922 the multirotor aircraft de Bothezat helicopter built by the United States Air Service was extremely limited; with a maximum altitude ever achieved of 5 metres [3]. For manned flight, helicopters dominated the rotorcraft field of aerial vehicles during the twentieth century. It wasn't until the advent of modern electronics that multirotors saw a boom in popularity as small UAVs. The Parrot

AR.Drone was one of the first commercially available 'toy' drones targeted to the general public. The Phantom series by DJI has also proven popular with many people attaching action cameras to capture aerial footage. Other industrial applications have also been made possible and are discussed in Section 1.2.3. In the past few years multirotors have seen a return to their origins as various manufacturers have presented their concept for vehicles capable of lifting humans, an example of which is given in Figure 1.1.

Figure 1.1: Skydiving from a multirotor – An example of a vehicle capable of lifting a human skydiver was successfully tested May 2017. Image Credit: Aerones.

1.2.2 Multirotors with more or less than four propellers

The principles for multirotors with more than four propellers such as hexacopters (six rotors) [4, 5, 6] and octocopters (eight rotors) [7, 8, 9] are similar to that of quadrotors and most of the trajectory generation and control techniques are transferable. The benefit of using more propellers is increased thrust, allowing greater payloads like heavy cameras or other sensors. There is also greater fault tolerance because if the flight controller senses a rotor failure there is redundancy in the system to prevent a crash landing [10, 11]. However, these types of multirotors are more expensive than quadrotors because additional motors and speed controllers are required. They are generally bigger than the smaller size quadrotors preferred in drone racing. Trirotors or tricopters (three rotors) are

another type of multirotor [12, 13, 14], particularly popular for acrobatic stunts performed in freestyle flying. They are not typically used for drone racing because one or more of the rotors must be capable of tilting in order to maintain stability as opposing co-axial rotor pairs are not available. This means the control system and mechanical design are more complex because a tilting actuator is necessary. The active tail element is particularly vulnerable in the event of collision. For the reasons outlined, multirotors with more or less than four rotors are not considered in this thesis.

1.2.3 Typical quadrotor applications

There are many practical applications for quadrotors, examples include: remote sensing for landslides [15, 16] and agriculture [17, 18], package delivery [19] and structural inspection [20]. These developments and many others are becoming feasible thanks to the development in sensor, battery and microcontroller technology. They all require trajectory planning and tracking to accomplish their tasks although their goals vary. Applications that are not industry orientated, particularly artistic applications in mainstream entertainment, are increasingly prominent. During the 2017 Super Bowl, 300 drones formed a flag and other patterns to accompany the halftime show [21]. Another artistic application is motion planning specifically designed for cinematographers using keyframes, a concept familiar to specialists in that field [22], thereby allowing a wider use base of drone technology.

Hobbyists have played a pivotal role improving the technology by continuing to push the bounds of what is possible and helping to develop the platforms. The popular open-source flight software Pixhawk (formerly known as Ardupilot) originated in the Computer Vision and Geometry Lab of ETH Zurich [23]. However, amateurs have contributed to the code and helped test the controller on a wide variety of platforms. For hobby pilots interested in competing there are two main quadrotor disciplines; freestyle flight where acrobatic stunts are performed and drone racing, which is the focus of this thesis.

Figure 1.2: A typical racing drone – A 210mm racing drone fitted with standard equipment. Image Credit: Author.

1.2.4 Drone racing

Drone racing consists of two or more participants racing their vehicles round a predefined course. An on-board camera streams real-time video to the human operator controlling the craft. Also known as 'FPV Racing' and 'Rotorcross', it has had much media attention including the article in the popular *Make* magazine [24]. Reports such as [25] remark on the worldwide nature of the competitions, the large sums of prize money available and investment from a variety of sources which all indicated the growing popularity of the sport. Enthusiasts have been racing radio controlled vehicles for decades but it is only in the past few years the weight of the video transmitting equipment and performance of drones has improved to a level where first person view racing is feasible. Other factors include the drop in prices and general availability of parts has made the sport affordable and possible for amateurs. Initially, races were organised informally but a number of associations (both profit and non-profit) have been formed with their own rules and regulations [26, 27, 28, 29]. One of the largest races to date was held in Dubai, 2016 by the World Drone Prix [30] and the course layout is pictured in Figure 1.3.

The first and, to the author's knowledge, only autonomous drone racing event to date within the research community was the competition held at the IROS (Intelligent Robots and Systems) 2016 conference in Daejeon, South Korea [31]. Nine teams competed in a time trial event inside a small netted, indoor arena. It

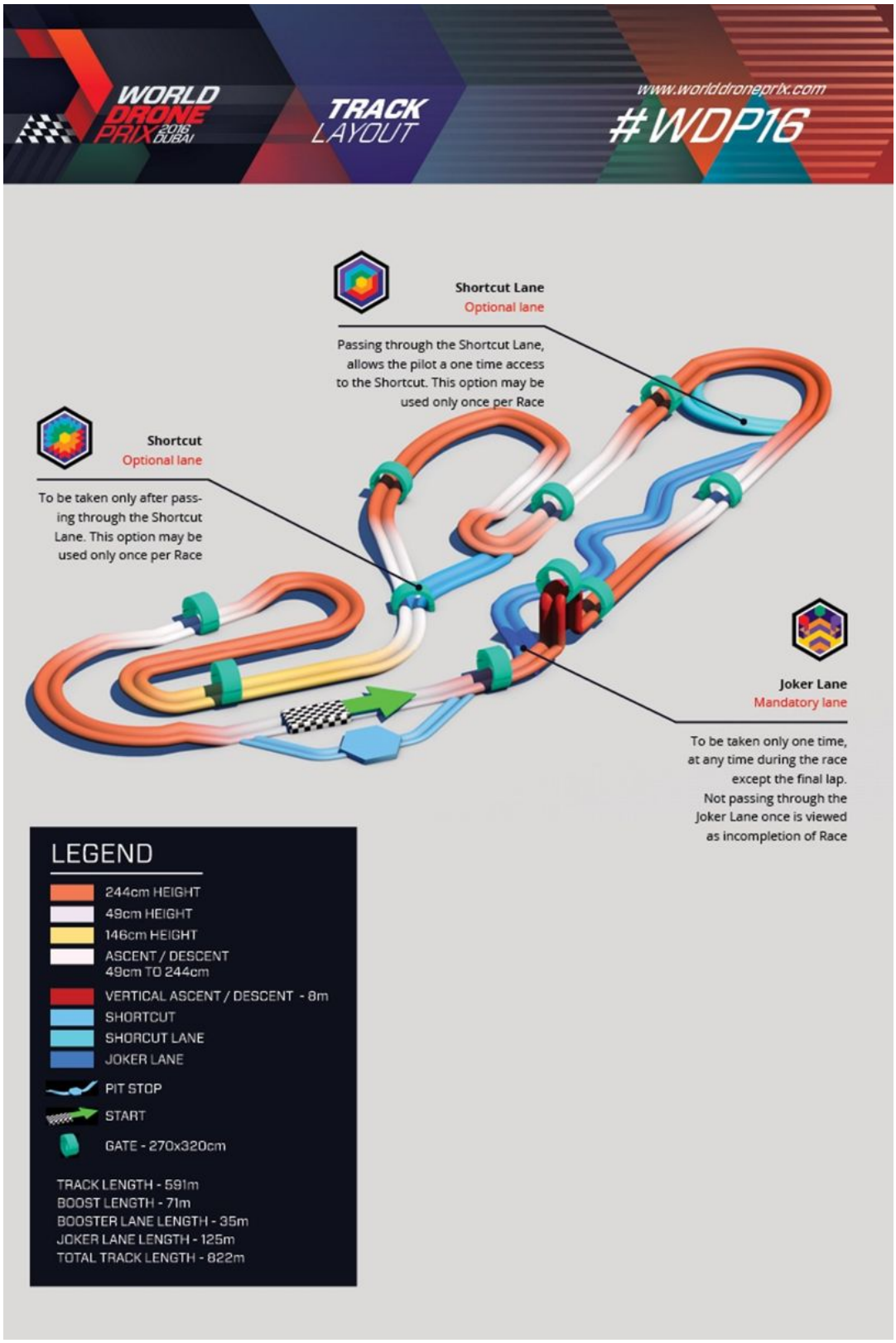


Figure 1.3: Dubai drone racing circuit – The track layout for the World Drone Prix 2016 Race. Image Credit: World Drone Prix.

was a challenging contest but the format was only loosely related to drone racing as practiced by human pilots. The density of gates and obstacles was very high and the vehicles struggled to determine their location. The vehicle performance mattered little because of the difficulties in state estimation.

Figure 1.4: IROS 2016 autonomous drone racing course – The course layout for the autonomous drone racing competition. Image Credit: Author.

1.3 Trajectory generation

Motion planning for autonomous systems is a well studied field of robotics. Generalised, it is the task of finding a trajectory in the configuration space [32] from an initial set of states to a desired set of states. It is a challenging problem that has been shown to be NP-hard in all but the simplest cases [33]. Complex environments with dynamic and static obstacles require sophisticated planning and simple point-to-point trajectories are often not enough. Generating a trajectory enables the vehicle to know where it will be in the future allowing it to calculate how it will achieve this.

In this thesis there are three criterion by which a trajectory is considered: dynamic feasibility, obstacle avoidance and optimality. The first two are essential for a trajectory to be valid. A dynamically feasible trajectory is one that the

vehicle controls are capable of performing. For instance, requiring an acceleration beyond the thrust ability of the craft is futile because it is impossible to track such a trajectory. Obstacle avoidance is also essential and simply involves avoiding collisions with the physical environment, be that trees, the ground or an artificial object. Finally, optimality is to be desired but in practice seldom achieved. It has already been mentioned that trajectory generation is an NP-hard problem and satisfying the first two criteria is challenging in itself. For this reason, 'good enough' is often acceptable and pseudo-optimal solutions are still of use. These three criterion will be referred to within the thesis and are introduced individually in the following sections. They are inherently linked when generating a trajectory but it is helpful to separate them for the purposes of this introduction.

1.3.1 Dynamic feasibility

One of the main reasons quadrotors are popular platforms, compared to other UAVs such as fixed wing aircraft, is because trajectory planning is simpler. This is due to minimal dynamic coupling and fewer constraints on the motion. For instance, a quadrotor will not stall if its airspeed drops below a certain value. The differential flatness of quadrotor dynamics has been proven [34]. A system with this property has an equal number of outputs to the number of inputs. All the states and outputs can be defined from the inputs analytically without the need to numerically integrate the system which is computationally expensive [35]. During this thesis, the quadrotor dynamics are considered a form that facilitates the flatness of the system. The implication of differential flatness for the quadrotor system is that a trajectory can be described by just the translational position and heading angle (the direction of the first axis in the body-fixed frame) as a function of time; this greatly simplifies the problem of trajectory generation for quadrotors. In other words, the motion along each axis of the inertial frame and the heading angle is enough to fully define the trajectory and a full description of attitude need not be considered at this stage.

A common method for defining the translational states of the vehicle along an axis is the use of a basis function parametrised by some argument such as time. After parametrising a basis function with the independent argument, suitable function coefficients are found to define an analytical expression that describes

the motion and satisfies any desired conditions on that axis. These conditions could be values of the higher order derivatives at the boundaries such as velocity and acceleration or waypoints that must be passed during the motion. Control of the boundary value derivatives is desirable for motion planning because separate segments can be joined to form a single, smooth trajectory without discontinuities where segments meet. The basis function can be analytically differentiated with respect to the parametrising argument and the higher order derivatives can be written as an expression. Examples of power series polynomials used for trajectory planning include those for UAVs [36, 37] and spacecraft [38, 39]. To illustrate the concept, a polynomial function is used as the basis function for position parametrised by time in Figure 1.5. Polynomials are popular because they are easy to manipulate, differentiate and implement. However, caution must be taken when higher order functions are used because stability issues can occur. The order of the polynomial must be sufficient to allow all the conditions to be achieved. It should be mentioned that the Gauss-Jordan elimination [40] used for curve fitting is inappropriate for motion planning because there is no independent parametrisation of the functions so there is no control over the boundary derivatives. This means the direction of the initial and final velocity cannot be controlled.

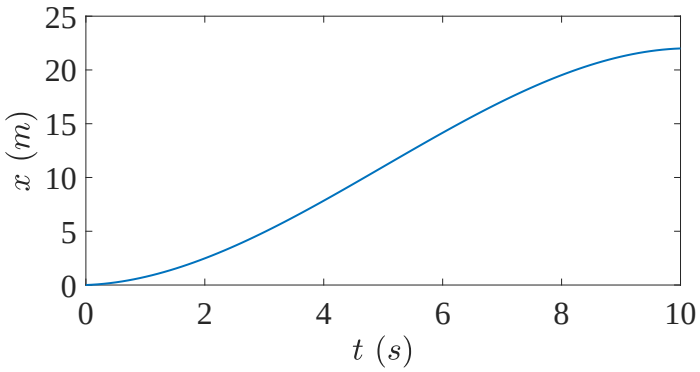


Figure 1.5: Polynomial basis function example – A power series polynomial used as a basis function to describe the position as a function of time using the equation $x(t) = 0.2t + 0.6t^2 - 0.04t^3$

Polynomials parametrised by time were used in [36] to form linear equations by matching the order of the function to the number of conditions. This is easily solved and simple to implement but the lack of flexibility means that for the trajectory to remain feasible throughout, the dynamic capabilities are not

fully exploited. If the maximum acceleration was constrained, the acceleration throughout could be reduced by increasing the trajectory time. However, for much of the manoeuvre the acceleration would be far below the limit and not exploit the capabilities of the vehicle, leading to an inefficient and sub-optimal trajectory. Alternatively, a higher order basis function can be used which allows for great flexibility. Since there is no longer a single solution for a given set of conditions, the coefficients can be chosen by minimising a cost function such as [41] that formulates an unconstrained quadratic program in which the coefficients are found that minimise the fourth derivative snap. According to [42] this is indirectly equivalent to minimising the control effort because in the system model used the inputs are functions of the fourth derivative of position so the trajectories generated minimise the integral of the square of the norm of the snap. Another method of achieving greater control over the dynamics is to parametrise the basis functions with an abstract argument instead of time. Such a trajectory is said to be in the virtual domain and must be mapped into the time domain before it is feasible.

The feasibility of a quadrotor trajectory can be assured with a variety of methods such as limiting the acceleration and rotational rates [43] or kinodynamic limits on the velocity and acceleration [44]. The disadvantage of using relatively simple constraints like this is that it is necessary to be overly conservative to ensure thrust available is actually capable of following the trajectory. One solution is calculating the actual thrust required from the trajectory [41]. In practice a safety margin between the maximum allowable thrust when generating a trajectory and the rotor's real limit should be included. This is because no model is completely accurate and unforeseen disturbances such as wind often cause more thrust to be required. The size of the safety margin is application dependent; a drone crashing during a race is of little importance when compared to a mission critical search and rescue vehicle.

1.3.2 Obstacle avoidance

Obstacle avoidance is achieved by ensuring the vehicle does not enter the forbidden regions of the configuration space [45] and stays in 'free space'. In the context of motion planning the forbidden region is some area or volume that the system

cannot pass through. This could be a physical object or an artificial constraint such as a no fly-zone near an airport. In this thesis it is assumed all the obstacles are static and known *a priori*. However, in practice a pre-existing map, combined with SLAM (Simultaneous Localization and Mapping) techniques and obstacle sensors like laser range finders would be used to provide the path planner with real-time obstacle location. For a fully autonomous vehicle, reactive collision avoidance is essential because the future motion of the other drones is unknown and cannot be planned in advance. Examples of vision-based positioning systems have been developed that also have the capability of reactive collision avoidance include [46, 47, 48, 49]. Fortunately, it's in the shared interest of all racers not to collide because mutually assured destruction is almost certain if such an event occurs.

Ideally a path planner will find a solution if a feasible path exists, or report that none exist. Such a planner is referred to as a 'complete path planner' and one of the earliest examples was proposed by Lozano-Pérez that found trajectories for polyhedral robots moving amongst polygonal/polyhedral obstacles [50]. This was generalised by Laumond for non-polygonal obstacles constructed from line sections and circular arcs [51]. Other cases based on the 'Piano Mover's Problem' were considered by Schwartz and Sharir [52, 53]. In practice, however, complete path planners are difficult to implement and costly to compute [54, 55]. For many problems in the real world, sub-optimal solutions with respect to path length, found by incomplete planners, are 'good enough'. For the application of drone racing, completing the course in the minimum time is desirable but so long as your vehicle finishes before its competitors you will still win.

The artificial potential field method is one such incomplete path planner that uses a simple, heuristic approach. An attractive potential is assigned to the goal region and repulsive potentials to forbidden regions in the configuration space. These fields can be implemented by direction equations, analogous to electro-potential fields, or equations developed specifically for the problem. Artificial potential fields have been applied to a range of applications including robotic arms [56], wheeled planetary exploration robots [57] and structural inspection using quadrotors [58]. The major limitation to the artificial potential method is the risk of becoming stuck in local minima, unable to escape [59]. In practice, this can be overcome by adding a random-walk to the algorithm capable of leaving

the local minima but this must be tuned and could be considered too risky for critical applications. Another drawback is control effort that exceeds the limits of the vehicle's capability can be required, violating the criteria that a trajectory is feasible [60].

Other approaches to the motion planning problem in the presence of obstacles include sampling based methods. These can be proven to be probabilistically complete, that is to say, as the number of points sampled tends to infinity, the probability of a path not being found when one exists approaches zero. The speed of convergence depends on the sampling method and the visibility of the free space. The Probabilistic Roadmap introduced by Kavraki [61] consists of two phases, a learning phase and a query phase. During the first phase, connections between the sampled configurations are found by a local planner and added to a topological graph to build a roadmap. Where a connection strays from the free configuration space it is rejected. The query phase then finds a path between the initial and final goal using a method such as Dijkstra's shortest path algorithm [62]. One benefit of Probabilistic Roadmap Methods is that more than one initial and final configuration can be defined so a single roadmap can be used to generate many trajectories. However, the comprehensiveness of this method is also a drawback because although the second phase is computationally efficient, the first is not. In cases where a thorough search is desirable this may not be an issue but for many practical implementations it is too slow.

Driven by the desire to reduce computation time other, less comprehensive methods, were developed. One of the most widely used is the Rapidly-Exploring Random Tree (RRT) method introduced by Lavalle [63]. Instead of comprehensively mapping the entire free space, a 'tree' of paths is incrementally constructed. In its simplest form, points are sampled in the free space and a tree of paths are formed from the initial configuration. The node on the existing graph closest to that of sampled node is used as the initial condition for a local path planner to the new node. If the new path segment enters at the forbidden region of the configuration space it can be either rejected or terminated at the boundary of the obstacle. Various extensions to RRT have been proposed, including using a potential function planner to improve the local path planner that connects nodes [64]. The sampling can also be changed to suit the problem, instead of random choosing nodes or uniform sampling it can be biased, using potential functions

for instance [65]. The RRT method can also be adapted to reactively account for obstacles by pre-computing branches off-line and rapidly switching between them as required by the presence of blockages in the scene [66]. However, the most significant development to date is RRT*, introduced by Karaman and Frazzoli [67]. Their algorithm is not only probabilistically complete, it will also converge on the optimum trajectory if given enough time. This is achieved by re-wiring the tree during the search process. When the cost to reach a node is less via a new route, the previous path is broken and an improved path is created. As soon as a feasible path is found the algorithm can be terminated at any time but by waiting the solution will continue to improve, although the rate of improvement decreases with time. For motion planning this is ideal because the best available solution can be used when the trajectory generation process must terminate.

1.3.3 Optimality

It has already been mentioned that of the three trajectory criterion optimality is the only one that can be considered non-essential. A pseudo-optimal solution is still better than one that does not consider optimality at all. Typically, optimality is judged by some measurable cost function that is desirable to minimise such as fuel expenditure, snap (the fourth derivative of position with respect to time), trajectory time or proximity to some region. It is also possible to have a multi-weighted cost function if two or more types of costs are to be considered.

Methods for finding optimal trajectories do exist, such as pseudospectral optimal control that has been demonstrated for a variety of motion planning cases in recent years, including wheeled robots [68] and the International Space Station [69]. By approximating the system state at quadrature nodes, time-dependent global polynomials, such as Legendre or Chebyshev functions are obtained. When combined with nonlinear programming methods using tools such as PSOPT [70] optimal solutions can be obtained. Constraints can be applied on the system states to ensure feasibility and path constraints to ensure obstacle avoidance. However, without a good initial guess, pseudospectral methods struggle or often fail to find the solution. For complex problems the computation time can take hours and still fail, rendering these methods unsuitable for many applications. Another method for obtaining optimal solutions is by applying the Maximum

Principle of optimal control to a kinematic description of a vehicle's motion [71]. In comparison to pseudospectral methods this technique is limited because only certain cost functions can be used and other constraints like obstacle avoidance cannot be included. However, they can be solved rapidly using parameter optimisation.

1.4 Trajectory tracking

Once a trajectory has been successfully generated a tracking controller is required to follow it. Ideally the error between desired states and the actual vehicle states are minimised to ensure the vehicle closely matches its desired motion. This section discusses the necessary elements for trajectory tracking.

1.4.1 State determination

In order calculate the error between the desired states and the actual states the actual states must be known or estimated. In simulation this is easily achieved because the outputs can be fed directly into the controller. However, this does not accurately represent the real world so depending on the fidelity of the simulation, noise or other errors may be added to the tracking controller input states. In the real world determining the translational position of the vehicle in the inertial frame is one of the most challenging aspects of autonomous UAV flight.

The IROS 2016 drone racing competition mentioned previously did not permit external positioning systems such as VICON that use infra-red cameras. These have been used for similar applications [72, 73] and are capable of tracking multiple vehicles and have been shown to be accurate to within 3 mm [74]. However, most racing takes place outdoors in areas far larger than these systems are capable of monitoring. They are also prohibitively expensive for use outside the best equipped labs. A Global Navigation Satellite System (GNNS) such as Global Positioning System (GPS) is commonly used outdoors but has a relatively low refresh rate typically between 1-10Hz which is slower than the minimum typical controller update rate of about 50Hz. Also satellite based systems are usually only accurate to within metres which is outside the tolerance necessary for flying

through small gates. Differential GPS, proposed by Morgan-Owen [75], can be used to improve the accuracy by using two receivers, one stationary in a known position and another on-board. However, the mathematics is complex and implementing such a system is non-trivial. On-board cameras have enabled machine vision to be used for positioning [76, 77]. This is a computationally demanding task and needs considerable processing power so it is often done by a base station that receives the footage wirelessly. Attitude determination is easier for sensors to measure than the translational states and is typically achieved by combining the data from a gyroscope and accelerometer using a Kalman Filter [78].

1.4.2 Quadrotor tracking controllers

Early attempts for quadrotors were based on linear control methods [79, 80] and were proven to be effective. However, quadrotors are inherently non-linear so outside the design limits the performance is uncertain. A non-linear controller was later proposed based on model predictive control [81]. However, all these controllers use Euler angles for attitude representation which make them susceptible to singularities and gimbal lock [82], particularly when performing aggressive manoeuvres that require large angles like those performed in drone racing. This is a problem for drone racing controllers and an alternative attitude representation should be used that doesn't suffer from these drawbacks. Quaternion based controllers are popular within the spacecraft community after the first controller using this attitude representation was introduced by Wie [83]. Composed of three complex elements and one real element, a quaternion vector can describe any attitude in a three-dimensional coordinate system. Quaternion based quadrotor tracking controllers have been developed [84, 85], however, these have an ambiguity due to the three-sphere (a higher-dimensional analogue of a three-dimensional sphere) S^3 double covering SO(3) and can also exhibit unwinding behaviour which causes excessively large rotations [86] or when there is noise in the system [87] if not accounted for correctly. The SE(3) nonlinear tracking controller developed in [88] suffers from none of the aforementioned drawbacks caused by the Euler or Quaternion attitude representations, making it a suitable choice for our application. It is also easily applied to an inverse dynamics approach of

determining the control thrusts needed given a reference trajectory which is convenient for efficiently checking feasibility, discussed previously, without needing to numerically integrate the equations.

1.4.3 Disturbances

Like all systems operating in the real world quadrotors are subject to disturbances and uncertainties that affect the dynamics of flight. Parametric uncertainties arise from the difficulty in accurately determining the physical parameters of the vehicle, such as inertia matrix and centre of mass. Any modification will change these parameters and it is impractical to recalculate them every time a change is made. System modelling can be used to describe the main physical effect acting upon quadrotors during flight such as propeller rotation, blade flapping, friction and inertial counter torques [89]. An accurate model allows the controller to better counteract these effects and improve performance but the nature of disturbances makes this difficult to do with a high degree of precision. External disturbances such as wind are impossible to predict.

Examples of control methods that consider disturbances include robust control, adaptive control, sliding mode control and Active Disturbance Rejection Control (ADRC). Robust controls are designed to work under the assumption that certain variables are not known but are bounded [90]. In comparison to robust controls, whose control policy remains static, adaptive controllers change with respect to measurements over time. This is usually done by adjusting the gains using simple, deterministic or fuzzy rules. Again, knowledge of the disturbance bounds and frequency are necessary. Adaptive control has been demonstrated for quadrotors [91]. Sliding mode controls switch between multiple control structures so that the trajectories move toward an adjacent region of another control structure. The crossing of the boundary of one control structure to another is called 'sliding' and in doing so generates a discontinuous control signal. Although sliding mode controls have been shown to be robust, they are susceptible to 'chattering', a phenomenon that incurs high heat loses in electrical power circuits and causes excessive wear of moving, mechanical components. Chattering can also occur in mechanical systems as vibrations, particularly in flexible appendages.

Examples of sliding mode controls for quadrotors include [92, 93, 94]. The chattering problem for altitude control using a sliding control has also been addressed [95].

The ADRC method uses a dynamic observer to estimate the disturbances at each sampling period so the controller can reject them [96, 97]. No models of the disturbances and uncertainties are necessary because an Extended State Observer (ESO) is used to estimate them. When a nonlinear ESO (that uses a nonlinear function) is used then a high estimation performance can in theory be achieved, however, extensive tuning of four parameters is required. A simpler observer, the Linear Extended State Observer (LESO) only needs two tuning parameter and is easier to apply, whilst still offering good performance [98]. ADRC has been applied to quadrotor vehicle control [99, 100, 101]. However, the first only considers attitude and all use Euler attitude representation which has the disadvantages discussed previously.

1.5 Objectives and contributions of the thesis

The following contributions have been made.

- The nonholonomic robot path planning at arbitrary speeds that uses geometric control theory first conducted by Biggs [102] and later developed by Maclean [103] for simple wheeled robots motion planning is extended for quadrotor UAVs and autonomous underwater vehicles (AUVs). For two different sets of constraints on the body velocities, a particular solution that can be expressed using standard trigonometric functions, analogous to sub-Riemmannian curves is found. For both cases reachable sets are defined and a basic obstacle avoidance method demonstrated. For the single body-velocity case, a more sophisticated obstacle avoidance method for AUVs using the Rapidly-Exploring Random Tree algorithm is designed. Relevant publications:
  - Jamieson, Jonathan, and Biggs, James. "Trajectory generation using sub-riemannian curves for quadrotor UAVs." Control Conference (ECC), 2015 European. IEEE, 2015.

  - Jamieson, Jonathan, and Biggs, James. "Path planning using concatenated analytically-defined trajectories for quadrotor UAVs." Aerospace 2.2 (2015): 155-170.
  - Jamieson, Jonathan, and Biggs, James. "Path planning on SE(3) for AUVs using the RRT method" IMA Conference on Mathematics of Robotics, 2015.

  Minimum time trajectory planning for drone racing was considered and a novel method for generating polynomial trajectories in the virtual domain and mapping them into the time domain using B-splines is developed. Non rest-to-rest manoeuvres are considered and a method for passing through the gates that describe the course is presented. A method for optimising the heading angle to ensure it points in the direction of the planar velocity vector whilst minimising accumulated acceleration is also developed. The trajectory generation methods are tested with three common quadrotor configurations. Finally, a 3D visualisation tool to view animated quadrotor trajectories within Matlab is presented. Relevant publications:

  - Jamieson, Jonathan, and Biggs, James. "Near minimum-time trajectories for quadrotor UAVs in complex environments." Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE, 2016.

- The SE(3) controller developed by Lee [88] was modified to include Active Disturbance Rejection Control (ADRC) by using a Linear Extended State Observer (LESO) and tested under square wave, sinusoidal and wind disturbances.

1.6 Thesis outline

This thesis is organised as follows. In Chapter 2 the mechanical and electrical design of quadrotor vehicles is introduced and various aspects of drone racing are defined. Also, the kinematics and dynamics of quadrotors and the autonomous underwater vehicles are given. Chapter 3 develops optimal controls for both types of vehicles and reachable sets are defined. Two types of obstacle avoidance are

also demonstrated. The curves developed in Chapter 3 are used for an additional application of AUV path planning in Chapter 4. In Chapter 5 virtual domain trajectories are defined using polynomials as basis functions. A method for navigating the gates without collision and choosing the heading angle are developed. The virtual trajectories are then mapped into the time domain in Chapter 6 to produce feasible, minimum time trajectories. Chapter 7 modifies an existing tracking controller to include disturbance rejection and tests the performance under three types of disturbances. In Chapter 8 the trajectory generation techniques developed in Chapter 5 and 6 are tested with different layout configurations and a trajectory visualisation tool is presented. Finally, in Chapter 9 the outcomes of the thesis and areas of future research are discussed.

Chapter 2

Vehicle models & design

In Chapter 1 the need for trajectory generation and trajectory tracking were introduced. However, before methods for these are developed it is necessary to give further details on quadrotor design. This chapter will cover the kinematic and dynamic models that are used in simulation to test the methods. Similarly, the methods used to define drone racing courses are described. Background theory on B-spline curves are also given because they are used multiple times during the trajectory generation process.

One of the earliest and most comprehensive works on quadrotor design and control was by Bouabdallah [89]. Other papers on quadrotor dynamics and aerodynamic effects include [104, 105]. In [37] multi-resolution modelling of quadrotors is investigated to the balance between a higher fidelity model and efficiency of the simulation. The dynamic model used in this thesis [88] is comparatively simple and neglects effects such as wing flutter and drag. Such effects could be included in the future to represent a more accurate quadrotor model but for the purposes of developing the trajectory generation and trajectory tracking method it is adequate.

The same layout of the rotors, sometimes referred to as the quadrotor configuration is typically used in most of the literature. However in the hobbyist community other layouts are sometimes used. In [24] one of the pilots being interviewed prefers a design that is bilaterally symmetric H-shape because he believes it to be more stable and capable of better performance in drone racing. To the author's knowledge, this claim has not been studied in the literature so this

thesis will define and compare three common configurations.

Original Contributions
The original contributions in this chapter are outlined as follows:

- The mechanical and electrical models of drone racing vehicles are discussed for the first time in the literature. The advantaged and disadvantages of the three commonly used layouts (standard, cross and H-Shape) are discussed, diagrams are given for each configuration.
- Typical courses used for drone racing, both indoors and outdoors are discussed for the first time in the literature.
- A method for defining drone racing gates using hollow cylinders is developed. This technique allows for the orientation and position in the inertial frame to be easily defined. Also, efficient calculations to check the trajectory is free from collisions can be performed.

The chapter is structured as follows. In Section 2.1 the quadrotor reference frames, mechanical and electrical are introduced. The kinematic and dynamic model are also given together with three common quadrotor layouts. Various aspects of drone racing are discussed in Section 2.2 include the method of modelling the gates and the scope of this thesis. Section 2.3 gives the background theory of generating B-spline curves used during the trajectory generation stage. Finally, Section 2.4 contains a summary of the chapter.

2.1 Quadrotor design

Although the focus of this thesis is on the theoretical aspects of trajectory generation and tracking, a basic understanding of how quadrotors operate is beneficial. It is also necessary to clarify the notation used to define the system's state and other parameters.

2.1.1 Reference frames

Three-dimensional reference frames can define the translational position and attitude of the vehicle. The first reference frame is used when deriving the curves in Chapters 3 and 4 is given in Figure 2.1a. States defined in this inertial frame $\mathbf{e^s}$ are denoted by superscript ^{s}. The vehicle's centre of mass is $\mathbf{x^s} = [x_1^s,\, x_2^s,\, x_3^s]^T$ and the vehicle's attitude is $R^s = [\mathbf{b_1^s},\, \mathbf{b_2^s},\, \mathbf{b_3^s}]$ where $\mathbf{b_1^s},\ \mathbf{b_2^s},\ \mathbf{b_3^s} \in \mathbb{R}^3$ are orthogonal body frame vectors. This type of attitude representation is discussed further in the following chapter. It is sometimes convenient to use the Euler angle representation when discussing rotation so the three scalar angles: roll ϕ_1^s, pitch ϕ_2^s and yaw ϕ_3^s are also given. The thrusts produced from each rotor are not given in this figure because this reference frame is only used for kinematic, not dynamic planning in this thesis.

The second reference frame $\mathbf{e}$, given in Figure 2.1b is used for the remaining chapters and is a typical NED (north-east-down) frame common in aerospace applications. The inertial position of the vehicle's centre of mass is $\mathbf{x} = [x_1,\, x_2,\, x_3]^T$ and has attitude $R = [\mathbf{b_1},\, \mathbf{b_2},\, \mathbf{b_3}]$. Scalar thrusts $f_1,\ f_2,\ f_3,\ f_4$, are produced from each rotor in the $-\mathbf{b_3}$ body frame direction and can be summed to calculate the total thrust $f = f_1 + f_2 + f_3 + f_4$. The Euler angles for roll ϕ_1, pitch ϕ_2 and yaw ϕ_3 are also given. To transform states from the first reference frame $\mathbf{e^s}$ to the second frame $\mathbf{e}$ the following relations are used:

$$\mathbf{x} = R_s \mathbf{x^s}, \quad R = R_s R^s \tag{2.1}$$

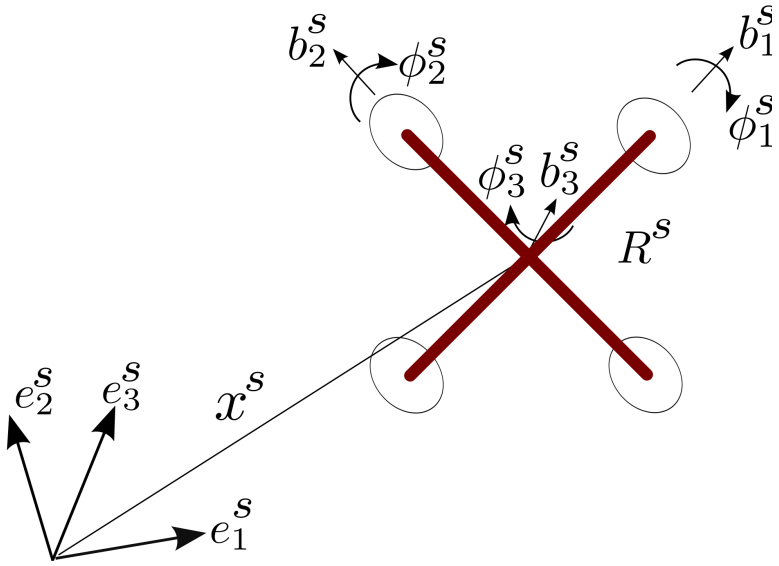
where $R_s \in \mathbb{R}^{3\times 3}$ is:

$$R_s = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.2}$$

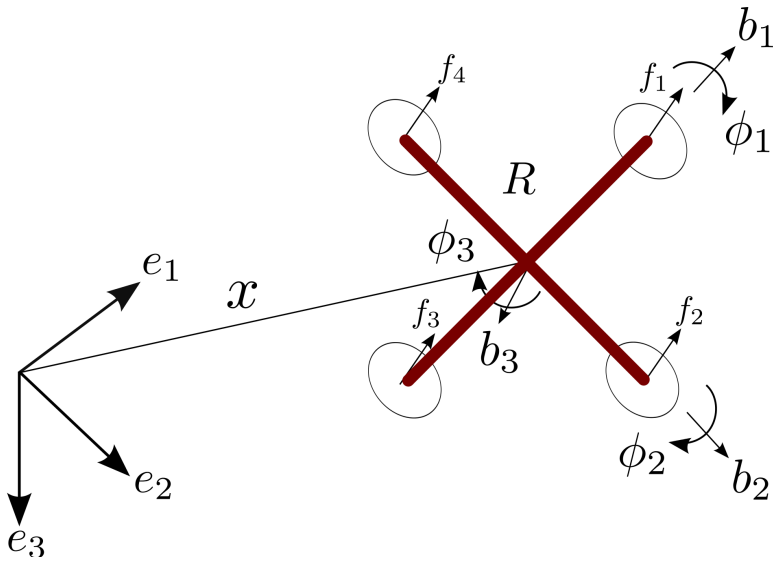
To go from the reference frame in Figure 2.1b to the reference frame Figure 2.1a the inverse of R_s can be used.

2.1.2 Mechanical design

The dynamic model of a quadrotor is considered as a rigid body and any aeroelastic effects or frame vibrations are ignored. This is a reasonable assumption



(a) Sub-Riemannian curve quadotor reference frame.



(b) Drone racing reference frame.

Figure 2.1: Quadrotor reference frames – a) This reference frame is used in Chapter 3 & 4. b) This reference frame is used in Chapters 5-8.

because significant effort is made to ensure frame stiffness by using modern, lightweight engineering materials like carbon fibre. Four rotors (vertically facing propellers) are mounted directly to motors attached to the arms some distance from the centre of mass. Common rotor layouts are given in Section 2.1.5. Opposite propellers share the same direction of rotation, so two spin clockwise and two spin counter-clockwise. This is necessary because a torque is induced about the yaw axis when the propellers rotate and the motion about this axis must be controllable. The counter-rotating blades serve a similar purpose to that of a tail rotor on a helicopter and stop the vehicle from spinning about the third body axis. A simplified explanation of how changes in the altitude and attitude are achieved is given in Figure 2.2 for a typical quadrotor in the next section. In practice, a combination of these actions is required to perform a complex manoeuvre. The method for finding each rotor thrust when the moment and total thrust are known is given in Chapter 7.

Unlike helicopters that change their thrust by adjusting the pitch of the rotor blades, most quadrotors change the angular speed of the rotors to vary the thrust. This means that there are no vulnerable linkages that can be easily damaged in crashes. Quadrotors that generate thrust in both directions of the $\mathbf{b_3}$ body frame axis are unusual but do exist, some use variable pitch blades such as the commercially available Stingray [106] or as research platforms [107, 108, 109]. Alternatively, the direction of the rotors can be reversed to change the direction of thrust [110]. A quadrotor with bi-directional thrust has enhanced dynamic capabilities, for instance, it can accelerate towards the ground faster than gravitational acceleration without flipping upside down. However, variable pitch rotors are relatively delicate and reversing the rotation direction induces a significant amount of back-EMF from the motor and requires special speed controllers. Currently, no drone racing is performed with either of these types of vehicles because the drawbacks outweigh the benefits.

2.1.3 Electrical & control design

The simplicity of the mechanical design is possible because the electronics and control system are complicated. Even when a human is controlling the vehicle, a low-level control system is necessary to convert the inputs into motor speeds. A

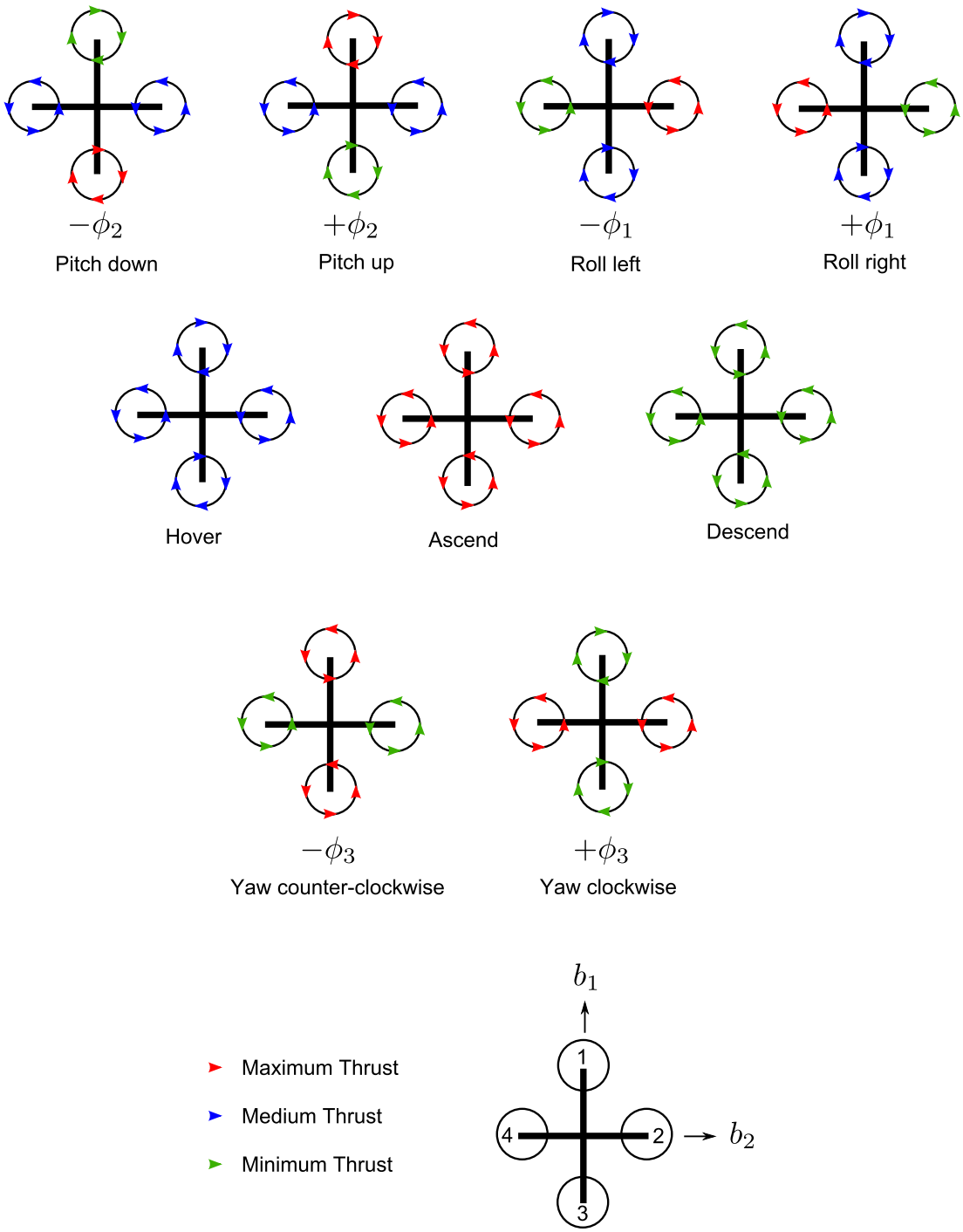


Figure 2.2: Rotor thrusts for basic manoeuvres – A simplified explanation of the quantitative rotor thrusts required to perform the basic quadrotor manoeuvres.

multirotor in which the rotor speeds were directly controlled would be impossible to fly, unlike fixed wing UAVs which can be flown manually. The minimum attitude sensor requirement is a 3-axis gyroscope when the angular rates are the control input and this is known as 'rate mode'. This type of flight is analogous to the behaviour of the roll and pitch rotations on a fixed-wing aircraft. If 'attitude mode' is desired, which is when the angle of the quadrotor is proportional to the position of the transmitter joysticks, an accelerometer is also required. In practice both a gyroscope and accelerometer are used and the data from both is filtered to produce a more accurate attitude estimation. Most pilots prefer rate mode for drone racing and tune the flight controller for aggressive, responsive flight.

The system design for a drone racing quadrotor is shown in Figure 2.3. As mentioned previously, a pilot uses a transmitter to wirelessly send commands to an on-board controller. The controller uses the information from the attitude sensors and the desired input from the pilot to compute the rotor speed. For the on-board camera, a wide-angle lens with a field of view around 148° is typical. This allows the operator a clear view of the vehicle's surroundings from the video, transmitted to them in real time. A CCD (charge-coupled device) sensor with a global shutter is preferred because it doesn't suffer from motion artefacts or 'jello' seen on a CMOS (complementary metal-oxide-semiconductor) sensor. Additionally, they offer a higher dynamic range allowing the pilot to see clearly in a variety of lighting conditions. Lithium-polymer cells are used for the batteries because they have a good energy density that minimises weight and a large maximum discharge current to meet the demands of high thrust manoeuvres. Ideally, the minimum capacity needed to complete the race is used in order to reduce the mass of the battery and improve performance.

2.1.4 Kinematics & dynamics

Quadrotors can be modelled as a rigid body of mass m_w with an inertia matrix $J_i \in \mathbb{R}^{3\times3}$ under constant gravitational acceleration g_a. The inertial position and velocity are $\mathbf{x} = [x_1, x_2, x_3]^T$ and $\mathbf{v} = [v_1, v_2, v_3]^T$ respectively. The angular velocities in the body-fixed frame are $\mathbf{\Omega} = [\Omega_1, \Omega_2, \Omega_3]^T$. The propellers generate a total scalar thrust f in the direction of the body frame thrust vector $\mathbf{e_3} = [0, 0, 1]^T$ and body frame moments $\mathbf{M} = [M_1, M_2, M_3]^T$. Any translational

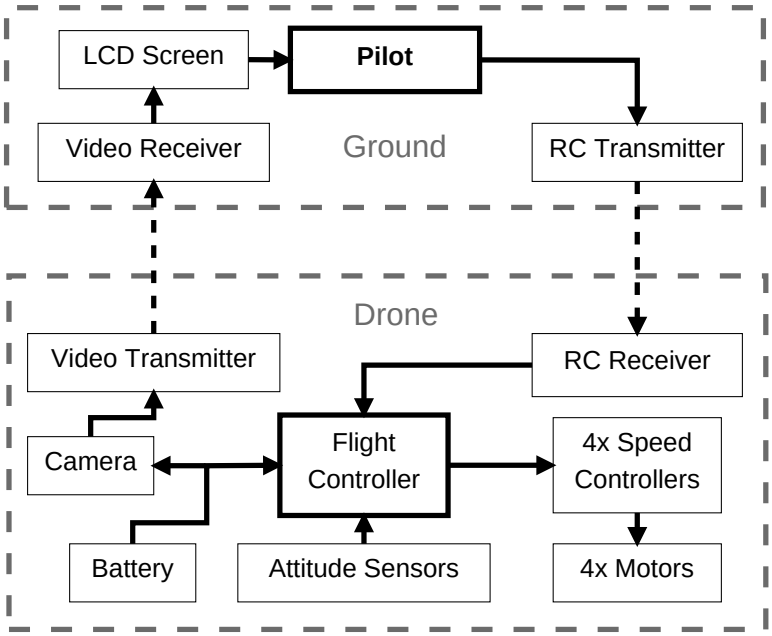


Figure 2.3: Drone racing system design – A block diagram of the drone racing system when controlled by a human pilot.

disturbances, such as aerodynamic forces, turbulence and wind are described by $\mathbf{D} = [D_1,\, D_2,\, D_3]^T$. The equations of motion are [88]:

$$\dot{\mathbf{x}} = \mathbf{v} \tag{2.3}$$

$$m\dot{\mathbf{v}} = mg_a\mathbf{e_3} - fR\mathbf{e_3} + \mathbf{D} \tag{2.4}$$

$$\dot{R} = R\hat{\Omega} \tag{2.5}$$

$$J_i\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times J_i\mathbf{\Omega} = \mathbf{M} \tag{2.6}$$

with $R \in SO(3)$ where:

$$\mathrm{SO(3)} \triangleq \{R \in \mathbb{R}^{3\times3} :\ \ R^TR = I_{3\times3} \quad \text{and} \quad \det(R) = 1\} \tag{2.7}$$

2.1.5 Layouts

There are three common quadrotor layouts: Standard, Cross and H-shape (colloquially referred to as 'dead-cat'). Photographs and diagrams of quadrotors with each of the layouts are given in Figure 2.4 and Figure 2.5 respectively. The Stand-

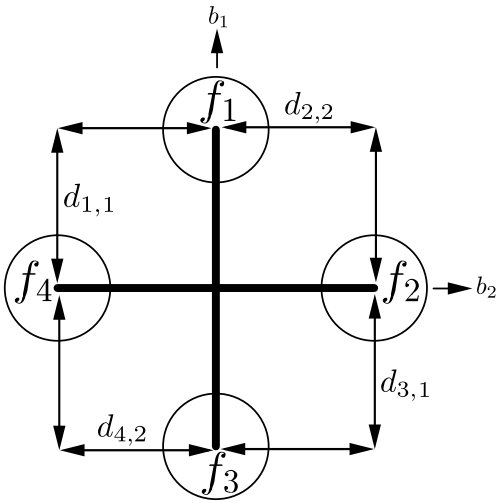
(a) Standard layout. Image credit: mydronelab.com

(b) Cross layout. Image credit: firstpersonview.co.uk

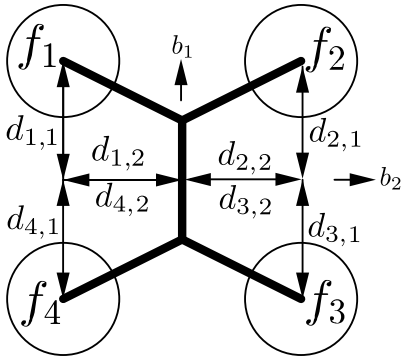
(c) H-shape layout. Image credit: Graham Dyer.

Figure 2.4: Photos of quadrotor layouts – Examples of three typical quadrotor layouts.

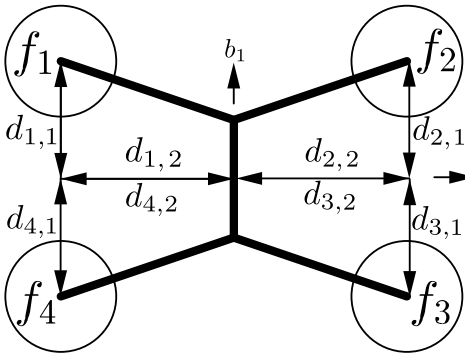
ard layout is the configuration typically found in the literature. It is simple to design and is supported by the majority of flight controllers on the market. Heavy payloads such as remote sensing equipment are usually mounted underneath to ensure a low centre of mass and improve stability. The Cross layout is similar to the Standard layout except each rotor is rotated 45° about the $\mathbf{b_3}$ axis. Compared to the Standard layout this configuration is superior for FPV flight because the view is unobstructed along the $\mathbf{b_1}$ axis where the camera is normally mounted. Since the battery is the heaviest single component for drone racing, it must be carefully positioned and the central bar in this design facilitates this. Finally, the H-shape is similar to the Cross configuration except that the arm lengths on the $\mathbf{b_2}$ axis are greater than those on the $\mathbf{b_1}$ axis.



(a) Standard layout.



(b) Cross layout.



(c) H-shape layout.

Figure 2.5: Quadrotor layout diagrams – The distances from the rotors used in this thesis are given in Table 2.1.

	Rotor 1		Rotor 2		Rotor 3		Rotor 4	
	$d_{1,1}$(m)	$d_{1,2}$(m)	$d_{2,1}$(m)	$d_{2,2}$(m)	$d_{3,1}$(m)	$d_{3,2}$(m)	$d_{4,1}$(m)	$d_{4,2}$(m)
Standard	0.315	0	0	0.315	0.315	0	0	0.315
Cross	0.223	0.223	0.223	0.223	0.223	0.223	0.223	0.223
H-Shape	0.256	0.183	0.256	0.183	0.256	0.183	0.256	0.183

Table 2.1: Layout configurations: properties – The rotor distances for the Standard, Cross and H-shape configurations used during the simulations.

In order to compare the configurations fairly it is necessary to ensure the sum of the rotor distances from the centre are the same for each configuration:

$$d_{total} = \sum_{n=1}^{4} \sqrt{d_{n,1}^2 + d_{n,2}^2} \tag{2.8}$$

This is because the induced torque about the third body axis $\mathbf{b_3}$ from a rotor is proportional to its distance from the centre of mass. In other words, if one configuration had disproportionately larger distances, then smaller thrusts would be required from the rotors to induce an equivalent moment. Table 2.1 gives the rotor distances for each layout configuration used during the simulations. Since the distance from the centre of each rotor to the centre of the quadrotor mass is the same for all rotors on all layouts the same torque coefficient $c_{\kappa f} = 8.004 \times 10^{-4}\,\mathrm{m}$, a measure of the torque about the yaw axis when the propellers rotate, can be used. This coefficient and the other physical properties including mass $m_w = 4.34\,\mathrm{kg}$, gravitational acceleration $g_a = 9.81\,\mathrm{m/s^2}$ are based on the quadrotor model in [88]. Finally, the moment of inertia is:

$$J_i = \begin{bmatrix} 0.0820 & 0 & 0 \\ 0 & 0.0845 & 0 \\ 0 & 0 & 0.1377 \end{bmatrix} \mathrm{kg\ m^2} \tag{2.9}$$

In reality the moment of inertia will change slightly depending on the layout. However, because the difference in arm lengths is relatively small and the majority of a quadrotor's mass is near the centre, the moment of inertia was assumed to be constant. The thrust limit of each rotor is constrained to the following bound:

$$0 \leq f_{1,2,3,4} \leq 12\,\mathrm{N} \tag{2.10}$$

2.1.6 Obstacles

The obstacles are modelled as spheres whose volumes are considered regions of the collision space that valid trajectories must avoid. Figure 2.6 gives the notation used where $\mathbf{x_o^s}$ is the centre of the sphere in the inertial reference frame and r_o^s is the radius of the sphere. Although comparatively simple when compared to objects in the real world, spheres were chosen because accurate collision checking is fast and efficient. Also, by combining spheres it is trivial to represent and approximate many different types of obstacles.

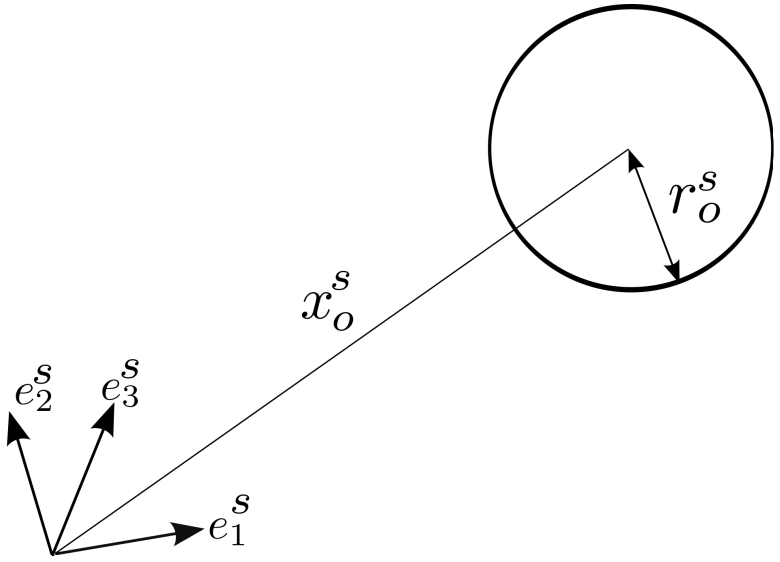


Figure 2.6: Obstacle reference frame – A diagram of the notation used to describe the position and size of an obstacle.

2.2 Drone racing

In this section, the types of course used in drone racing and how they can be modelled virtually are discussed. The scope of drone racing considered in this thesis is also discussed.

2.2.1 Courses

Drone racing courses are the equivalent to racetracks used for motorsports. The vehicles will usually start at rest on the ground, or hovering in front of a line before

(a) An indoor drone racing course. Image Credit: thedronegirl.com

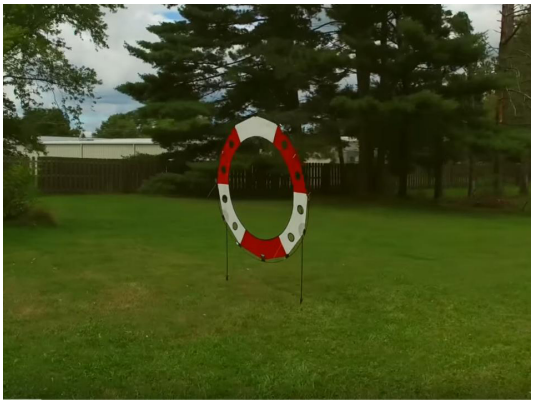
(b) An outdoor drone racing course. Image Credit: newswire.com

Figure 2.7: Types of drone racing course – An example of an indoor and outdoor course typically used for drone racing.

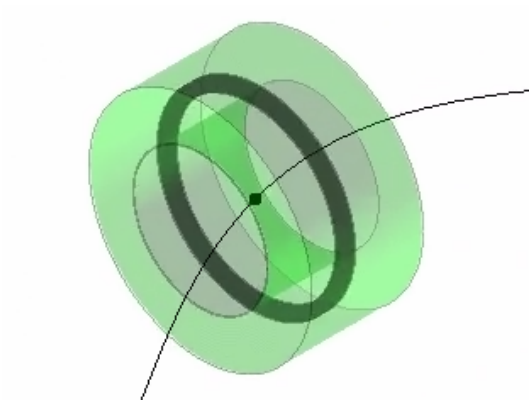
navigating through a series of gates and obstacles. One or more laps (complete circuits of the entire course) are raced before finishing at some predefined point. Both indoor and outdoor courses (Figure 2.7) are used for drone racing. Existing features such as pillars, gradients in the terrain and trees can be deliberately incorporated to make the course more challenging. Currently, no standard track layouts have emerged and the course designers are given free reign when laying out the gates. The courses used to test the trajectory planner and tracking control in this thesis are representative to those used in actual races.

2.2.2 Gates

The basic building blocks of the courses are gates that must be flown through sequentially. Closed gates like the ones described in this section make it easier to check if a vehicles has navigated the gate because the acceptable region is clearly defined. Open gates that don't have a closed shape can cause confusion and doubt that a vehicle successfully passed through if travelling at high speeds. The gates are modelled as 'hoops' and are toroids with square cross-sections. Figure 2.8a is a an example of a commercially available gate similar to the kind rendered in Figure 2.8b. The black torus that represents the physical hoop contained within a virtual, transparent green, hollow cylinder. This green volume is used by the motion planner as the collision region that the trajectory may not enter. The process of calculating the green volume is as follows.



(a) Premier RC Key Hole gate. Image Credit: Mini Quad Club.



(b) A rendering of the type of gates used in this thesis.

Figure 2.8: Type of drone racing gates – The rendered gate is modelled around the type of gate shown in the photograph. The green volume represents the collision region in which the trajectory should not enter.

Figure 2.9 defines the notation used to model the gate in the motion planner and Figure 2.10 explains how the parameters are calculated. Each gate is modelled as a small cylinder inside a larger one. Any point outside the large, outer cylinder or inside the small, inner cylinder is considered safe from collision for a given gate. Any point inside the outer cylinder and outside the inner cylinder is assumed to collide with the gate and the trajectory considered infeasible. Subscript h for 'hoop' was used to denote parameters related to the modelling of the gates. The inner and outer radii are $r_{h,i}$ and $r_{h,o}$ respectively and the length of both cylinders is l_h. For a given gate, the cylinders are defined from their centre with a translational position $\mathbf{x_h} \in \mathbb{R}^3$ and orientation $R_h \in SO(3)$. The first body fixed axis of the gate $\mathbf{b_{h,1}}$ is aligned with the central axes of the cylinders, with the direction of $\mathbf{b_{h,2}}$ and $\mathbf{b_{h,3}}$ defined in the diagram. The rotation of the gate can also be specified using the Euler angles $\phi_{h,1}$, $\phi_{h,2}$ and $\phi_{h,3}$ about the body axes $\mathbf{b_{h,1}}$, $\mathbf{b_{h,2}}$ and $\mathbf{b_{h,3}}$ respectively. By modelling the gates in this way, complex, three dimensional configurations in which gates are rotated about multiple axes can be represented. Currently, the gates of most real courses are perpendicular to the ground $\mathbf{b_{h,3}} = \mathbf{e_3}$ but as the sport develops gates angled about the $\mathbf{b_{h,2}}$ axis can be used to force more sophisticated trajectories that offer a greater challenge.

Figures 2.10a and 2.10b are diagrams that explain the method for determining the parameters of the virtual gate's collision region based on the properties of the

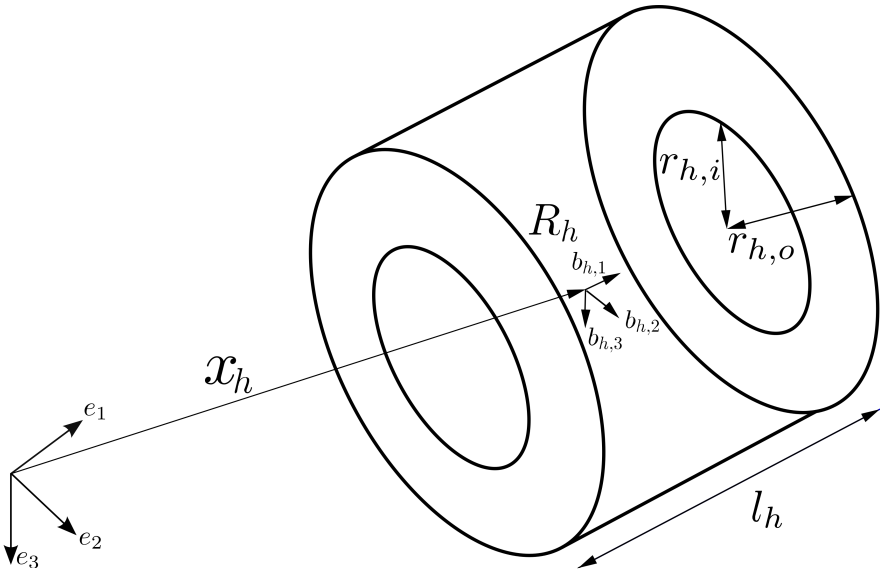
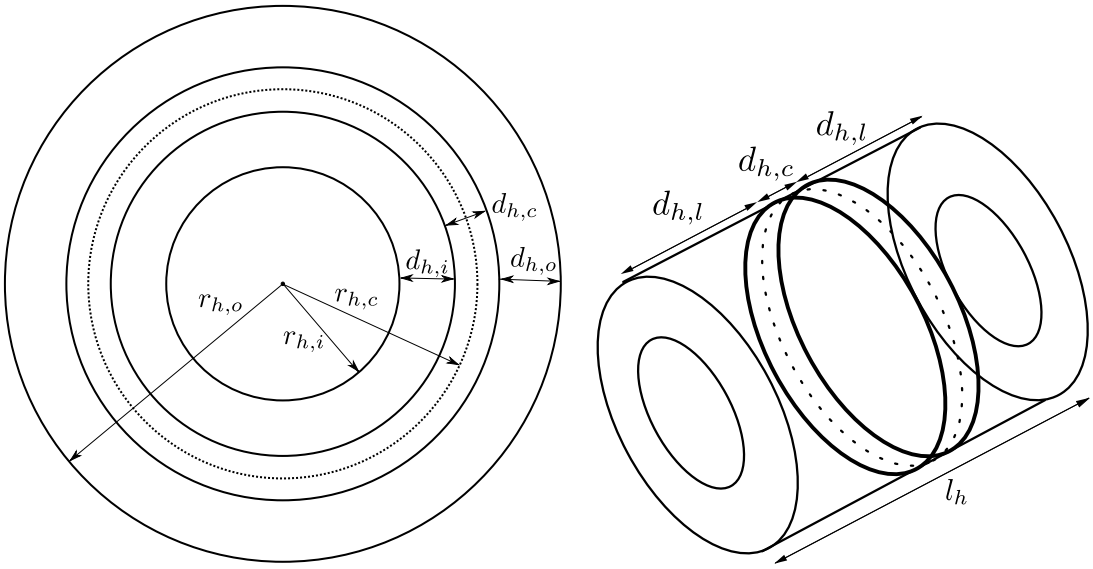


Figure 2.9: Drone racing gate: reference frame – The notation used to define the size and shape of the collision region surrounding the gate.

hoop and the vehicle's characteristics. If $r_{h,i} = r_{h,c} - \frac{1}{2}d_{h,c}$ is the inner radius of physical hoop and $r_{h,o} = r_{h,c} + \frac{1}{2}d_{h,c}$ is the outer radius of the physical hoop then the radii of the virtual collision region can be calculated. If d_{max} is the largest distance between the centre of mass and the tip of the rotor then the conditions $d_{h,i} > d_{max}$ and $d_{h,o} > d_{max}$ must be observed to ensure the collisions are avoided. Similarly, along the length of the cylinder the condition $d_{h,l} > d_{max}$ must be maintained. The size of the internal radius, outer radius and length of the cylinder can now be defined:

$$\begin{aligned} r_{h,o} &= r_{h,c} + \frac{d_{h,c}}{2} + d_{h,o}, \\ r_{h,i} &= r_{h,c} - \frac{d_{h,c}}{2} - d_{h,i}, \\ l_{h,o} &= 2d_{h,l} + d_{h,c} \end{aligned} \tag{2.11}$$

In practice a safety margin should be included in $d_{h,i}$, $d_{h,o}$ and $d_{h,l}$ because the trajectory tracking is unlikely to be perfect and an error in the translational position is more likely to cause a crash if a small tolerance is used.



(a) End view of the gate. **(b)** Isometric view of the gate

Figure 2.10: Drone racing gate: parameters – The notation used to find the properties that define the collision region of the gate.

2.2.3 Objectives & scope

There are a variety of quadrotor simulators available with various levels of fidelity; some are aimed at researchers [111, 112] and others for amateur pilots [113, 114]. In this thesis a simple dynamic simulation that neglected aerodynamic and other higher order effects was used to test the tracking controllers because it was adequate for the purpose of comparing them. A truly autonomous quadrotor must be capable of accurately determining its translational position so the tracking controller can follow the planned trajectory. The available methods and sensors discussed in the introduction are not currently capable of doing so with the required accuracy. However, location estimation is vital to many UAV applications and is an active area of research that is continuing to improve the capabilities of multirotor flight. Attitude determination has progressed to the point were even low cost sensors are capable of accurate measurement.

Trajectory planning for drone racing racing is essentially a minimum time problem because the winner is the vehicle that finishes the course first. In this thesis, moving obstacles such as other vehicles are not considered and the only obstacle present are the gates. This assumption is made because many courses take place in open areas. Future work may also consider other obstacles in the

environment. The generated trajectories are benchmarks of what a quadrotor is capable of with a given layout configuration, mass and thrust capabilities. This is useful for vehicle design and modification because the impact on the trajectory time and performance can be calculated.

2.3 B-spline curves

The definition of B-splines given in this section should be considered an introduction to the subject of such curves [115]. The full derivation is lengthy and modern numerical software typically has inbuilt functions to handle their computation so custom routines do not need to be written. For the purposes of this thesis a brief explanation of the curves and their properties is sufficient. A B-spline is a type of piecewise polynomial spline function and is defined as follows:

$$x(u) = \sum_{i=0}^{n} N_{i,k}(u)\chi_i \tag{2.12}$$

where u is the segment number $0 \leq u \leq n-k+2$, the control coefficients are χ, $n+1$ is the number of control coefficients, k is the order of the curve and $N_{i,k}$ is the basis function and expressed using the Cox-de Boor recursion formula:

$$N_{i,k}(u) = \frac{(u-\kappa_i)N_{i,k-1}(u)}{\kappa_{i+k-1}-\kappa_i} + \frac{(\kappa_{i+k}-u)N_{i,k-1}(u)}{\kappa_{i+k}-\kappa_{i+1}} \tag{2.13}$$

and the special case

$$\begin{gathered} N_{i,1} = 1 \text{ if } \kappa_i \leq u \leq \kappa_{i+1} \\ N_{i,1} = 0 \quad \text{otherwise} \end{gathered} \tag{2.14}$$

The knot values κ_i $(0 \leq i \leq n+k)$ are as follows:

$$\begin{gathered} \kappa_i = 0 \text{ if } i < k \\ \kappa_i = i-k+1 \text{ if } k \leq i \leq n \\ \kappa_i = n-k+2 \text{ if } i > n \end{gathered} \tag{2.15}$$

The basis function has the continuity of the segments inherently built into it. Since a B-spline is fully defined by the N_κ number of knots κ and the coefficients χ, there are three ways of changing the curve shape: 1) the positions of the

control points; 2) knot location; 3) order of the curve. The order of the curve is simply $k = N_\kappa - n$ and the curve is C^{k-2} continuous between segments. The knots within the knot vector must be given in ascending order and define the position of the control points. An example B-spline showing the control points is given in Figure 2.11.

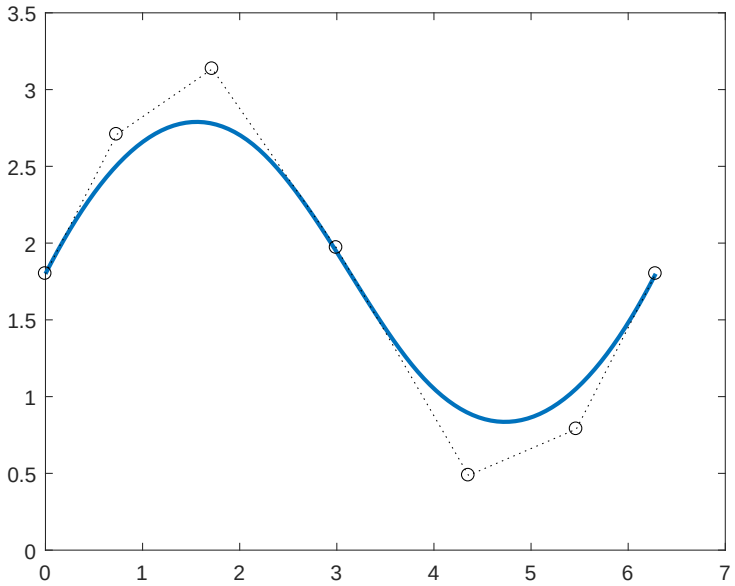


Figure 2.11: B-spline example – A B-spline with a knot vector $\kappa = [0, 0, 0, 2.2, 2.9, 3.8, 6.3, 6.3, 6.3, 6.3]$, coefficient vector $\chi = [1.8, 2.7, 3.1, 1.9, 0.48, 0.79, 1.8]$ and order $k = 4$.

There are various types of knot vector. A cardinal B-spline, for instance, has knots that are equidistant from each other. However, these are not commonly used because it is not possible to clamp knots to a specific location with multiple coincident knots which is desirable for most applications. The continuity of the derivative order is reduced by 1 for each additional knot. This knot multiplicity is typically used at the boundaries to ensure the curve finishes and ends at the desired location. The interior knots can have a uniform or non-uniform distribution between the boundary knots.

The main benefit of B-splines is their local control modification scheme. Each control point only affects a limited number of nearby segments so changes in one part of the curve leave the rest of curve untouched. This is in comparison

to Bézier curves which are a small subset of B-spline curves and have a strict relationship between curve degree and number of control points. Since the knot vector does not demand uniformity and because of the local control modification scheme it is trivial to insert knots where required without changing other parts of the curve. For instance, if only a small section of the curve needs a fine level of control knots only need to be added in this location. B-splines are also flexible in allowing continuity of the curve to be specified. This means that whatever degree of smoothness is required can be satisfied. Finally, the convex hull property of B-splines guarantees the curve will remain bounded in the region of the control points. This is in contrast to polynomials which can exhibit large amplitude of oscillations if not defined well. There are extensions to the basic B-spline method. For instance, NURBS (Non-Uniform Rational B-Spline) adds a weighting to each basis function which can be thought of as a spring that pulls the curve closer to the control points [116]. However, in this thesis the basic B-spline approach was used because it was adequate for the requirements, future work may consider the use of NURBS.

2.4 Chapter summary

In this chapter the vehicles models used for trajectory planning were introduced. The inertial reference frames used to derive the trajectories were also given. Three common quadrotor layout configurations were presented: Standard, Cross and H-Shape. The physical parameters used in the simulations for each type were also given. Drone racing system design was discussed to provide a background for these types of vehicles. A simple yet flexible method for modelling the gates to define the courses was introduced so the collision region can be clearly defined. Finally the basic theory of construction B-splines was introduced because some of the trajectory generation methods use this type of curve.

Chapter 3

Trajectory planning on SE(3) with sub-Riemannian curves

In this chapter a trajectory generation method is developed using optimal curves analogous to sub-Riemannian curves. They are found by solving an optimal control problem via Pontryagin's Maximum Principle on a 3D-Lie group. The method for motion planning on the Special Euclidean Group SE(3) is an inherently kinematic approach to describing the motion of a vehicle. The trajectory obtained will not only contain the translational position but also the attitude as a function of time. This attitude is purely kinematic, in a system with forces such as gravity in the case of the quadrotor, the rotation of the vehicle required to track the position over time is different to the kinematic rotation obtained by the planner. However, the translational position on SE(3) can be projected to $\mathbf{x}^{\mathbf{s}} \in \mathbb{R}^3$ and used as the translational position basis function for the tracking controller. In Chapter 4 the curves derived in this chapter are used to plan motions for an AUV. In this application the full translational and rotational information contained within the 3D-Lie group can be used for trajectory tracking.

One of the earliest motion planning problems to be considered was that of simple wheeled robots in two dimensions. The Dubins car is named after the mathematician who derived a method for generating trajectories for car-like vehicles with arbitrary initial and final positions and rotations [117]. Dubins curves describe the shortest length path connecting the initial and final position when it is assumed the vehicle can move at a fixed speed and turn at a fixed

angular velocity. These trajectories consist of straight line segments and arcs of constant curvature. Were these segments meet, the trajectory is only smooth to C^1. It is possible to extend Dubins curves into three dimensions and plan trajectories for quadrotors [118]. Similarly, they have been used as the basis for AUV motion planning [119].

Dubins curves are just one way of finding controls for vehicles with kinematic constraints. These constraints are a type of nonholonomic system, which is a system that has constraints not only on position but also velocity. Compared to holonomic systems that are only constrained by the configuration variables position and attitude [120] nonholonomic systems are more challenging motion planning problems. This is because the velocity constraints prevent certain paths. A common example of a nonholonomic system is that of a standard car which must use a parallel parking manoeuvre to fit into a space, it cannot simply stop next to the gap and transversely slide into position. It should also be noted that it is not possible to integrate a nonholonomic constraint and obtain holonomic constraints because they are non-integrable.

Two examples of methods that can be used to plan motions for nonholonomic systems are describing the controls using sinusoids [121] and numerical based pseudospectral optimal control [122]. Sinusoidal controls have been demonstrated for a number of systems including snakeboards [123], AUVs [124] and spacecraft [125]. However, there are no obvious ways to include obstacles or optimality so feasibility is the only motion planning criterion discussed in Chapter 1 that is satisfied. Pseudospectral optimal control is capable of producing feasible, optimal and obstacle free trajectories for nonholonomic systems by specifying the constraints as differential equations. Unfortunately, as discussed in the introduction the computation times can be excessively large, especially when obstacles are included and a reasonable initial guess is usually necessary for the solution to converge.

A nonholonomic system can be formulated as an optimal control problem by incorporating the Maximum Principle of optimal control into Geometric control theory. The trajectories produced will satisfy some cost function and the kinematic constraints of the vehicle. This method is best suited to local planning in the presence of only a few obstacles because the optimal control method cannot directly account for them. However, it is possible to incorporate the local planner

into a global sampling-based planner to navigate complex environments. This is similar to the use of Dubins curves within an RRT framework [118].

In this chapter two kinematics models are used to define the motions represented using Lie groups and their associated algebra. After defining a quadratic cost function to the kinematics, the Maximum Principle of optimal control is applied to define an optimal Hamiltonian from the kinematics and cost function. The final step lifts the dual of the Lie algebra using the Poisson bracket to obtain Hamiltonian vector fields which are solved to find the analytical optimal controls. These controls describe trajectories that are analogous to sub-Riemannian curves.

Original Contributions

The original contributions in this chapter are outlined as follows:

- A closed-form optimal solution for a rigid-body with a nonholonomic sliding constraint i.e. where the body fixed translational velocity is fixed to motion in one direction is derived. This analytical solution was then utilized to design motions for a quadrotor. The solution is also applied to AUVs in Chapter 4.
- A closed-form optimal solution for a rigid-body where the body fixed translational velocity is not fixed in one direction is derived.
- Reachable sets for both cases are derived numerically under unit speed and unit time conditions to demonstrate the ability of the analytical solutions to reach a variety of final translational states in the inertial reference frame.
- A method that uses parametric optimisation to find a curve with a desired final position is presented and demonstrated for both types of curves.
- A method that uses parametric optimisation to find a curve that considers both the final position and final velocity is presented and demonstrated for both types of curves. Although the desired velocity could not be matched in either case it is possible to influence the shape of the curve and this is used to practical effect by developing a simple obstacle avoidance method.

The chapter is structured as follows. In Section 3.1 the Special Euclidean groups of motions SE(3) and Special Orthogonal group SO(3) are introduced and their

respective algebras $\mathfrak{se}(3)$ and $\mathfrak{so}(3)$ are given. A particular solution of the optimal curves for when only the first body-velocity is non-zero is derived in Section 3.2. Similarly, a particular solution for when three body-velocities are present is derived in 3.3. Reachable sets are found for both types of curve in Section 3.4. The method for finding the final conditions using parametric optimisation is developed in Section 3.5. In Section 3.6 obstacle avoidance by biasing the final velocity is investigated. Finally a summary of the chapter is given in Section 3.7.

3.1 Lie groups and Lie algebra

Before deriving the curves it is necessary to give a brief explanation of Lie groups and Lie algebra that are used to describe the motions. The Special Euclidean group of motions $g^s \in SE(3)$ defines the position in six-dimensional space:

$$g^s = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \mathbf{x^s} & & R^s & \end{pmatrix} \tag{3.1}$$

where $\mathbf{x^s} \in \mathbb{R}^3$ is the translational position in the inertial $\mathbf{e^s}$ frame and R^s is the rotational position formulated in the Special Orthogonal group SO(3), the set of every 3×3 rotation matrix, which is formally defined as:

$$\mathrm{SO}(3) \triangleq \{R^s \in \mathbb{R}^{3\times 3} : \ R^{sT} R^s = I_{3\times 3} \quad \text{and} \quad \det(R^s) = 1\} \tag{3.2}$$

where $I_{3\times 3} \in \mathbb{R}^{3\times 3}$ is the identity matrix. The matrix R defines an orientation by specifying the direction of each axis in the body frame relative to another frame. The first column is for the first body axis, the second column for the second axis and the third column for the third axis.

The basis elements of the Lie algebra $\mathfrak{se}(3)$ of the matrix Lie group SE(3) are

given by:

$$A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \; A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \; A_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
$$B_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \; B_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \; B_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{3.3}$$

where A_1, A_2, A_3, B_1, B_2, B_3 describe the infinitesimal motion in the roll, pitch, yaw, surge, sway and heave directions of the vehicle respectively. That is to say, the derivatives of rotation and translation around each of the standard axes evaluated at the identity. The Lie bracket is defined as $[X, Y] = XY - YX$ and the corresponding Lie bracket table is then:

[,]	A_1	A_2	A_3	B_1	B_2	B_3
A_1	0	A_3	$-A_2$	0	B_3	$-B_2$
A_2	$-A_3$	0	A_1	$-B_3$	0	B_1
A_3	A_2	$-A_1$	0	B_2	$-B_1$	0
B_1	0	B_3	$-B_2$	0	0	0
B_2	$-B_3$	0	B_1	0	0	0
B_3	B_2	$-B_1$	0	0	0	0

Table 3.1: Commutative table for basis on $\mathfrak{se}(3)$ – The commutative table is used for calculating the Lie bracket.

The Lie bracket can be thought of as the derivative of Y along the flow generated by X. Similarly, the basis elements for SO(3) are:

$$E_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \; E_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \; E_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.4}$$

where E_1, E_2 and E_3 describe infinitesimal motion in the roll, pitch and yaw directions of the vehicle respectively. The corresponding Lie bracket table for SO(3) is given in Table 3.2.

[,]	E_1	E_2	E_3
E_1	0	E_3	$-E_2$
E_2	$-E_3$	0	E_1
E_3	E_2	$-E_1$	0

Table 3.2: Commutative table for basis on $\mathfrak{so}(3)$ – The commutative table is used for calculating the Lie bracket.

From the Wei-Norman representation [126], R^s can be expressed as:

$$R^s(t) = e^{\phi_1^s(t)E_1}\, e^{\phi_2^s(t)E_2}\, e^{\phi_3^s(t)E_3} \tag{3.5}$$

where $\phi_1^s(t)$, $\phi_2^s(t)$ and $\phi_3^s(t)$ are scalar functions of time t. The rotation matrix (3.5) can be considered using the Euler attitude representation as the motion of a freely rotating rigid body spinning about the first body-axis by ϕ_1^s, the second body-axis by ϕ_2^s and the third body-axis by ϕ_3^s. The equivalent Euler angle rotation sequence for the rotation matrix (3.5) is $\phi_3^s \leftarrow \phi_2^s \leftarrow \phi_1^s$. A diagram of the reference frame can be found in Figure 2.1a.

A vehicle with angular velocities about the body frame $\mathbf{\Omega^s} = [\Omega_1^s, \Omega_2^s, \Omega_3^s]^T$ and body frame translational velocities $\nu^{\mathbf{s}} = [\nu_1^s, \nu_2^s, \nu_3^s]^T$ has kinematics that can be expressed as a left-invariant system on the Lie group $SE(3)$:

$$\frac{dg^s}{dt} = g^s(B_1\nu_1^s + B_2\nu_2^s + B_3\nu_3^s + A_1\Omega_1^s + A_2\Omega_2^s + A_3\Omega_3^s) \tag{3.6}$$

where B_1, B_2, B_3, A_1, A_2, A_3 are basis elements of the Lie algebra of the Lie group $SE(3)$ defined previously in (3.3).

3.2 Single body velocity case

The curves derived in this section assume the vehicle can only produce thrust in the direction of ν_1^s, and the angular rotational velocity about the same axis should be minimised. Therefore, the body velocities $\nu_2^s = \nu_3^s = \Omega_1^s = 0$. These conditions are necessary to enable the initial direction of the velocity to be specified and is a requirement for the sampling-based path planning RRT method implemented in Chapter 4. After applying these assumptions on the body velocities, the kinematics on SE(3) given in (3.6) can be reduced to the following

kinematic constraint:

$$\frac{dg^s}{dt} = g^s(\nu_1^s B_1 + A_2\Omega_2^s + A_3\Omega_3^s) \tag{3.7}$$

It should be noted that although there is no direct control over the translational motion in B_2 and B_3 direction the Lie bracket allows motion in these directions.

A quadratic cost function that is analogous to defining a sub-Riemannian metric on SE(3) but parametrised by virtual time rather than arc length is then defined:

$$J_{s,1} = \frac{1}{2}\int_0^T \nu_1^{s2} + c^s(\Omega_2^{s2} + \Omega_3^{s2})dt \tag{3.8}$$

subject to the given boundary conditions $g^s(0) = g_0^s$ and $g^s(T) = g_T^s$ and where c^s is a constant weight and ν_1^s, Ω_2^s, and Ω_3^s are measurable and bounded. The weighting c^s is required because the distance metric on SE(3) is degenerate. Since the cost function is multi-objective, the weighting can be manipulated to increase or decrease the angular velocity of the planned reference trajectory relative to the translational velocity in the forward direction. For instance, if c^s is reduced then the trajectories become more helical.

From [127, 128] an application of the coordinate free Maximum Principle that minimises (3.8) subject to (3.7) yields the Hamiltonian:

$$H = \nu_1^s\mu_1 + \Omega_2^s\mu_5 + \Omega_3^s\mu_6^s - \frac{\rho_0}{2}(\nu_1^{s2} + c^s(\Omega_2^{s2} + \Omega_3^{s2})) \tag{3.9}$$

where $\rho_0 = 1$ for regular extremals and $\rho_0 = 0$ for abnormal extremals. This thesis only considers regular extremals because it has been shown that abnormal extremals are a subset of regular extremals [127], therefore $\rho_0 = 1$ is used throughout. Noting that (3.9) is a concave function and from Pontryagin's Maximum Principle that if:

$$\frac{\partial H}{\partial \nu_1^s} = \frac{\partial H}{\partial \Omega_2} = \frac{\partial H}{\partial \Omega_3^s} = 0,\ \frac{\partial^2 H}{\partial^2 \nu_1^s} < 0,\ \frac{\partial^2 H}{\partial^2 \Omega_2^s} < 0,\ \frac{\partial^2 H}{\partial^2 \Omega_3^s} < 0 \tag{3.10}$$

then ν_1^s, Ω_2^s, and Ω_3^s are optimal. To satisfy these conditions the following controls were chosen:

$$\nu_1^s = \mu_1^s,\ \ \Omega_2^s = \frac{\mu_5}{c^s},\ \ \Omega_3^s = \frac{\mu_6}{c^s} \tag{3.11}$$

and substituted into (3.9) to yield the optimal Hamiltonian:

$$H^* = \frac{1}{2}(\mu_1^2 + \frac{\mu_5^2}{c^s} + \frac{\mu_6^2}{c^s}) \tag{3.12}$$

The corresponding Hamiltonian vector fields describing the necessary conditions for optimality are calculated using the Poisson bracket $\frac{d\mu_i}{dt} = [\mu_i, H^*]$ where $i = 1, \ldots, 6$ [127]. For example, the derivative of first extremal curve written out fully is:

$$\begin{aligned}\dot{\mu}_1 = \frac{\partial H^*}{\partial \mu_1}\{\mu_1, \mu_1\} + \frac{\partial H^*}{\partial \mu_2}\{\mu_1, \mu_2\} + \frac{\partial H^*}{\partial \mu_3}\{\mu_1, \mu_3\} + \frac{\partial H^*}{\partial \mu_4}\{\mu_1, \mu_4\} \\ + \frac{\partial H^*}{\partial \mu_5}\{\mu_1, \mu_5\} + \frac{\partial H^*}{\partial \mu_6}\{\mu_1, \mu_6\}\end{aligned} \tag{3.13}$$

The vector fields are then:

$$\begin{aligned}\dot{\mu}_1 = \frac{\mu_2\mu_6}{c^s} - \frac{\mu_3\mu_5}{c^s},\ \dot{\mu}_2 = -\frac{\mu_1\mu_6}{c^s},\ \dot{\mu}_3 = \frac{\mu_1\mu_5}{c^s}, \\ \dot{\mu}_4 = 0,\ \dot{\mu}_5 = \mu_1\mu_3 - \frac{\mu_4\mu_6}{c^s},\ \dot{\mu}_6 = -\mu_1\mu_2 + \frac{\mu_4\mu_5}{c^s}\end{aligned} \tag{3.14}$$

In addition to the conserved quantity (3.12) the Casimir functions:

$$I_2 = \mu_1^2 + \mu_2^2 + \mu_3^2 \tag{3.15}$$

$$I_3 = \mu_1\mu_4 + \mu_2\mu_5 + \mu_3\mu_6 \tag{3.16}$$

are constant along the Hamiltonian flow (3.14). If $\dot{\mu}_3 = 0$ is assumed then a particular solution can be identified. This means that only a subset of the possible solutions can be obtained from the following analysis. However, the final functions that describe the motion of the vehicle will not contain elliptic integrals and are considerably easier for software to evaluate.

To solve the Hamiltonian equations, the derivative of the first extremal curve from (3.14) is squared:

$$(\dot{\mu}_1)^2 = \frac{\mu_2^2\mu_6^2}{c^{s2}} - 2\frac{\mu_2\mu_3\mu_5\mu_6}{c^{s2}} + \frac{\mu_3^2\mu_5^2}{c^{s2}} \tag{3.17}$$

and from the conserved quantities the following is true:

$$c^s(2H^* - \mu_1^2) = \mu_5^2 + \mu_6^2,\ \ I_2 - \mu_1^2 = \mu_2^2 + \mu_3^2 \tag{3.18}$$

therefore:

$$c^s(2H^* - \mu_1^2)(I_2 - \mu_1^2) = \mu_2^2\mu_5^2 + \mu_2^2\mu_6^2 + \mu_3^2\mu_5^2 + \mu_3^2\mu_6^2; \tag{3.19}$$

From (3.16):

$$(I_3 - \mu_1\mu_4)^2 = \mu_2^2\mu_5^2 + \mu_3^2\mu_6^2 + 2\mu_2\mu_3\mu_5\mu_6 \tag{3.20}$$

and therefore:

$$(\dot{\mu}_1)^2 = \frac{1}{c^2}\left(c(2H^* - \mu_1^2) - (I_3 - \mu_1\mu_4)^2\right) \tag{3.21}$$

Equation (3.21) can be solved in terms of elliptic functions. However, just as the derivative of the third extremal was set to zero, the same will be done to the first, so $\dot{\mu}_1 = 0$. This ensures only trigonometric functions are needed to solve the final equations. Particular solutions at the singularity are found using the following condition:

$$(\dot{\mu}_1)^2 = \frac{1}{c^{s2}}\left(c^s(2H^* - \mu_1^2) - (I_3 - \mu_1\mu_4)^2\right) = 0 \tag{3.22}$$

The real root of this equation is denoted by $\mu_1(0)$ which is a constant velocity in the body fixed frame so $\mu_1 = \nu_1^s$. The conserved quantities (3.18) can written in the terms of the constants η_1 and η_2:

$$\eta_1^2 = \mu_5(0)^2 + \mu_6(0)^2, \;\; \eta_2^2 = \mu_2(0)^2 + \mu_3(0)^2 \tag{3.23}$$

where $\eta_1^2 = c^s(2H^* - \mu_1(0)^2)$ and $\eta_2 = I_2 - \mu_1(0)^2$. These conserved quantities suggest using polar coordinates to solve the differential equations (3.14), so the ansatz solution is attempted:

$$\begin{aligned} \mu_2 &= \eta_2 \sin(\alpha t + \beta), \;\; \mu_3 = -\eta_2 \cos(\alpha t + \beta) \\ \mu_5 &= \eta_1 \sin(\alpha t + \beta), \;\; \mu_6 = -\eta_1 \cos(\alpha t + \beta) \end{aligned} \tag{3.24}$$

Substituting (3.24) into (3.14), two solutions for α are obtained:

$$\alpha = \frac{s\nu_1^s}{c^s\eta_2}, \;\; \alpha = \frac{\Omega_1}{c^s} - \frac{\nu_1^s\eta_2}{\eta_1} \tag{3.25}$$

where $\Omega_1^s = \mu_4$ and is constant since $\dot{\Omega}_1^s = 0$. Thus, (3.24) is only a particular solution, if and only if:

$$\Omega_1^s = \mu_4 = \nu_1^s\frac{\eta_1^2 + c^s\eta_2^2}{\eta_1\eta_2} \tag{3.26}$$

The particular solution for the Hamiltonian vector flow (3.14) can now be expressed as:

$$\begin{gathered}\mu_1 = \nu_1^s, \;\; \mu_2 = \eta_2 \sin\theta, \;\; \mu_3 = -\eta_2 cos\theta, \\ \mu_4 = \nu_1^s \frac{\eta_1^2 + c^s \eta_2^2}{sr}, \;\; \mu_5 = \eta_1 \sin\theta, \;\; \mu_6 = -\eta_1 \cos\theta\end{gathered} \tag{3.27}$$

where $\theta = \frac{\eta_1 \nu_1^s}{c^s \eta_2} t + \beta$ and $\beta = atan2\left(-\mu_5(0), \mu_6(0)\right)$. For convenience, define a constant $K^2 = I_2$ then $R^s \in SO(3)$ is known to satisfy the equation [127]:

$$R^s P R^{s-1} = \rho \tag{3.28}$$

where $P = \mu_1 E_1 + \mu_2 E_2 + \mu_3 E_3$ and $\rho \in \mathfrak{so}(3)$ is a constant matrix. This fact implies that any orbit $R^s P R^{s-1} = \rho$ is conjugate to $\rho = K E_1$ and therefore it suffices to integrate the particular orbit:

$$R^s P R^{s-1} = K E_1 \tag{3.29}$$

Then let ϕ_1^s, ϕ_2^s and ϕ_3^s denote the coordinates of a point on SO(3) according to the formula:

$$R^s = e^{\phi_1^s E_1} e^{\phi_2^s E_2} e^{\phi_3^s E_1} \tag{3.30}$$

then substituting (3.30) into (3.29) yields:

$$P = K e^{-\phi_1^s E_1} \, e^{-\phi_2^s E_2} \, E_1 e^{\phi_2^s E_2} \, e^{\phi_1^s E_1} \tag{3.31}$$

which yields:

$$P = K \begin{pmatrix} 0 & -\cos\phi_1^s(t)\cos\phi_2^s(t) & \cos\phi_2^s(t)\sin(\phi_1^s(t) \\ \cos\phi_1^s(t)\cos\phi_2^s(t) & 0 & \sin\phi_2^s(t) \\ -\cos\phi_2^s(t)\sin(\phi_1^s(t) & -\sin\phi_2^s(t) & 0 \end{pmatrix} \tag{3.32}$$

then:

$$\begin{gathered}\nu_1^s = K \sin\phi_2^s, \\ \eta_2 \sin\theta = K \cos\phi_2^s \sin\phi_1^s, \\ \eta_2 \cos\theta = \cos\phi_1^s \cos\phi_2^s\end{gathered} \tag{3.33}$$

therefore:

$$\sin\phi_2^s = \frac{\nu_1^s}{\sqrt{\eta_2^2 + \nu_1^{s2}}}, \;\; \cos\phi_2^s = \frac{\eta_2}{\sqrt{\eta_2^2 + \nu_1^{s2}}} \tag{3.34}$$

and as $\tan\theta = \tan\phi^s$ then $\phi_3^s = \theta$. To calculate ϕ_1^s, substitute (3.30) into (2.5)

and obtain the following relationships:

$$\begin{aligned} \sin\phi_2^s \sin\phi_3^s \dot{\phi}_1^s + \cos\phi_3^s \dot{\phi}_2^s = \Omega_2^{s*}, \\ -\sin\phi_2^s \cos\phi_3^s \dot{\phi}_1^s + \sin\phi_3^s \dot{\phi}_2^s = \Omega_1^{s*} \end{aligned} \tag{3.35}$$

which can be manipulated to give:

$$\dot{\phi}_1^s = \frac{\Omega_2^{s*} \sin\phi_3^s - \Omega_1^{s*} \cos\phi_3^s}{\sin\phi_2^s} \tag{3.36}$$

which can be simplified and integrated assuming $\phi_1^s(0) = 0$ to yield:

$$\phi_1^s = \left(\frac{\eta_1 \sqrt{r^2 + \nu_1^{s2}}}{\eta_2 c^s} \right) t \tag{3.37}$$

Substituting all the values found into (3.30) to obtain a particular optimal solution for R^s, we can then obtain $\mathbf{x^s}$ by integration of the equation $\frac{d\mathbf{x^s}}{dt} = R^s \nu^{\mathbf{s}}$. These analytical solutions are then used to form a particular solution g_p^s which for convenience is pulled back to the identity by computing $g^s(t) = g_p^s(0)^{-1} g_p^s(t)$ so the trajectory starts at the origin. The translational position $\mathbf{x^s}$ from the particular solution is obtained via the projection $[1, \mathbf{x^s}]^T = g^s[1, 0, 0, 0]^T$. The analytical functions that describe the motion in each axis are:

$$\begin{aligned} x_1^s &= \frac{j_2^2 \nu_1^s}{\gamma} \sin\gamma t + j_1^2 \nu_1^s t \\ x_2^s &= \frac{j_2 \nu_1^s}{\gamma} ((\cos\gamma t - 1)\cos\beta - j_1 \sin\beta\,(\gamma t - \sin\gamma t)) \\ x_3^s &= \frac{j_2 \nu_1^s}{\gamma} ((\cos\gamma t - 1)\sin\beta + j_1 \cos\beta\,(\gamma t - \sin\gamma t)) \end{aligned} \tag{3.38}$$

where:

$$\gamma = \frac{\eta_1 \sqrt{\eta_2^2 + \nu_1^{s2}}}{\eta_2 c^s},\ j_1 = \frac{\nu_1^s}{\eta_2^2 + \nu_1^{s2}},\ j_2 = \frac{\eta_2}{\eta_2^2 + \nu_1^{s2}} \tag{3.39}$$

Differentiating (3.38) gives the translational velocities in the inertial frame:

$$\begin{aligned} \dot{x}_1^s &= j_1^2 \nu_1^s + j_2^2 \nu_1^s \cos\gamma t \\ \dot{x}_2^s &= -\frac{j_2 \nu_1^s}{\gamma} \big(- j_1(\gamma - \gamma\cos\gamma t)\sin\beta + \gamma\cos\beta\sin\gamma t \big) \\ \dot{x}_3^s &= \frac{j_2 \nu_1^s}{\gamma} \big(j_1(\gamma - \gamma\cos\gamma t)\cos\beta - \gamma\sin\beta\sin\gamma t \big) \end{aligned} \tag{3.40}$$

The projection of a particular optimal curve $g_p^s \in SE(3)$ onto R^s is given by:

$$R^s = \begin{pmatrix} R^s_{11} & R^s_{12} & R^s_{13} \\ R^s_{21} & R^s_{22} & R^s_{23} \\ R^s_{31} & R^s_{32} & R^s_{33} \end{pmatrix} \tag{3.41}$$

where:

$$\begin{aligned}
R^s_{11} &= j_1^2 + j_2^2 \cos t\gamma \\
R^s_{12} &= -j_1 j_2 (\cos t\gamma - 1) \sin(t\alpha + \beta) - j_2 \cos(t\alpha + \beta) \sin t\gamma \\
R^s_{13} &= -j_1 j_2 (\cos t\gamma - 1) \cos(t\alpha + \beta) + j_2 \sin(t\alpha + \beta) \sin t\gamma \\
R^s_{21} &= -j_1 j_2 (-1 + \cos t\gamma) \sin\beta + j_2 \cos\beta \sin t\gamma \\
R^s_{22} &= \cos\beta \cos(t\alpha + \beta) \cos t\gamma + (j_2^2 + j_1^2 \cos t\gamma) \sin\beta \sin(t\alpha + \beta) - j_1 \sin t\alpha \sin t\gamma \\
R^s_{23} &= \cos(t\alpha + \beta)(j_2^2 + j_1^2 \cos t\gamma) \sin\beta - \cos\beta \cos t\gamma \sin(t\alpha + \beta) - c_1 \cos t\alpha \sin t\gamma \\
R^s_{31} &= -j_1 c_j (-1 + \cos t\gamma) \cos\beta - j_2 \sin\beta \sin t\gamma \\
R^s_{32} &= -\cos(t\alpha + \beta) \cos t\gamma \sin\beta + \cos\beta (j_2^2 + j_1^2 \cos t\gamma) \sin(t\alpha + \beta) + j_1 \cos t\alpha \sin t\gamma \\
R^s_{33} &= \cos\beta \cos(t\alpha + \beta)(j_2^2 + j_1^2 cost\gamma) + \cos t\gamma \sin\beta \sin(t\alpha + \beta) - j_1 \sin t\alpha \sin t\gamma
\end{aligned} \tag{3.42}$$

The initial conditions of the extremal curves will be found using parametric optimisation later in Section 3.5.

3.3 Multiple body velocities case

The curves derived in this section are not limited to a single translational body-fixed velocity like those in the previous section. The benefit of this is that when the initial velocity direction is not constrained these curves are more flexible and allow for a greater range of trajectory shapes. The derivation in this section is explained in less detail than that of the previous one because the process of deriving the curves is similar to the previous section.

The kinematic constraints for this curve are given in (3.6) and without any simplifications and the quadratic cost function is defined as:

$$J_{s,2} = \frac{1}{2} \int_0^T (m_1^s \nu_1^{s2} + m_2^s \nu_2^{s2} + m_3^s \nu_3^{s2} + c_1^s \Omega_1^{s2} + c_2^s \Omega_2^{s2} + c_3^s \Omega_3^{s2})\, dt \tag{3.43}$$

and the corresponding Hamiltonian is:

$$\begin{aligned} H &= \mu_1\nu_1^s + \mu_2\nu_2^s + \mu_3\nu_3^s + \mu_4\Omega_1^s + \mu_5\Omega_2^s + \mu_6\Omega_3^s \\ &-\frac{\rho_0}{2}(m_1^s\nu_1^{s2} + m_2^s\nu_2^{s2} + m_3^s\nu_3^{s2} + c_1^s\Omega_1^{s2} + c_2^s\Omega_2^{s2} + c_3^s\Omega_3^{s2}) \end{aligned} \tag{3.44}$$

with the controls chosen:

$$\begin{aligned} \nu_1^s = \frac{\mu_1}{m_1^s},\ \ \nu_2^s = \frac{\mu_2}{m_2^s},\ \ \nu_3^s = \frac{\mu_3}{m_3^s}, \\ \Omega_1^s = \frac{\mu_4}{c_1^s},\ \ \Omega_2^s = \frac{\mu_5}{c_2^s},\ \ \Omega_3^s = \frac{\mu_6}{c_3^s}. \end{aligned} \tag{3.45}$$

Substituting the controls into (3.44) yields the optimal Hamiltonian:

$$H^* = \frac{1}{2}\left(\frac{{\mu_1}^2}{m_1^s} + \frac{{\mu_2}^2}{m_2^s} + \frac{{\mu_3}^2}{m_3^s} + \frac{{\mu_4}^2}{c_1^s} + \frac{{\mu_5}^2}{c_2^s} + \frac{{\mu_6}^2}{c_3^s}\right) \tag{3.46}$$

The Hamiltonian vector fields are:

$$\begin{aligned} \dot{\mu}_1 &= \frac{\mu_2\mu_6}{c_3^s} - \frac{\mu_3\mu_5}{c_2^s}, \\ \dot{\mu}_2 &= \frac{\mu_3\mu_4}{c_1^s} - \frac{\mu_1\mu_6}{c_3^s}, \\ \dot{\mu}_3 &= \frac{\mu_1\mu_5}{c_2^s} - \frac{\mu_2\mu_4}{c_1^s}, \\ \dot{\mu}_4 &= \left(\frac{m_2^s - m_3^s}{m_2^s m_3^s}\right)\mu_2\mu_3 + \left(\frac{c_2^s - c_3^s}{c_2^s c_3^s}\right)\mu_5\mu_6, \\ \dot{\mu}_5 &= \left(\frac{m_3^s - m_1^s}{m_1^s m_3^s}\right)\mu_1\mu_3 + \left(\frac{c_3^s - c_1^s}{c_1^s c_3^s}\right)\mu_4\mu_6, \\ \dot{\mu}_6 &= \left(\frac{m_1^s - m_2^s}{m_1^s m_2^s}\right)\mu_1\mu_2 + \left(\frac{c_2^s - c_1^s}{c_1^s c_2^s}\right)\mu_4\mu_5 \end{aligned} \tag{3.47}$$

The Casimir functions for this case are similar to those given in the previous one. They can be written as conserved quantities in terms of the constants η_1 and η_2:

$$\eta_1^2 = \frac{\left(\mu_1(0)\mu_4(0) + \mu_2(0)\mu_5(0)\right)^2}{\mu_1(0)^2 + \mu_2(0)^2},\ \ \eta_2^2 = \mu_1(0)^2 + \mu_2(0)^2 \tag{3.48}$$

The ansatz solutions used for solving the conserved quantities are:

$$\begin{aligned} \mu_1 = -\eta_2\cos(\alpha t + \beta),\ \ \mu_2 = \eta_2\sin(\alpha t + \beta) \\ \mu_4 = -\eta_1\cos(\alpha t + \beta),\ \ \mu_5 = \eta_1\sin(\alpha t + \beta \end{aligned} \tag{3.49}$$

and the third and sixth extremals are constant and defined respectively as $\mu_3 = \nu_3^s$ and $\mu_6 = \Omega_3^s$. Two expressions for α can then be obtained:

$$\alpha = \frac{\Omega_3^s}{c_3^s} - \frac{\nu_3^s \eta_1}{c_2^s \eta_2}, \quad \alpha = \frac{\Omega_3^s}{c_3^s} - \frac{\nu_3^s \eta_1}{c_1^s \eta_2} \tag{3.50}$$

Equating α from both results in the following expression:

$$\frac{\Omega_3^s}{c_3^s} - \frac{\nu_3^s \eta_1}{c_2^s \eta_2} = \frac{\Omega_3^s}{c_3^s} - \frac{\nu_3^s \eta_1}{c_1^s \eta_2} \tag{3.51}$$

which is true if, and only if $c_1^s = c_2^s = c^s$. By taking the derivative of (3.49) and substituting into (3.47) two more expressions for α can be obtained:

$$\alpha = \left(\frac{m_2^s - m_3^s}{m_2^s m_3^s}\right)\frac{\eta_2 \nu_3^s}{\eta_1} + \left(\frac{c^s - c_3^s}{c^s c_3^s}\right)\Omega_3^s, \quad \alpha = \left(\frac{m_1^s - m_3^s}{m_1^s m_3^s}\right)\frac{\eta_2 \nu_3^s}{\eta_1} + \left(\frac{c^s - c_3^s}{c^s c_3^s}\right)\Omega_3^s \tag{3.52}$$

Again, both expressions for α are equated:

$$\left(\frac{m_2^s - m_3^s}{m_2^s m_3^s}\right)\frac{\eta_2 \nu_3^s}{\eta_1} + \left(\frac{c^s - c_3^s}{c^s c_3^s}\right)\Omega_3^s = \left(\frac{m_1^s - m_3^s}{m_1^s m_3^s}\right)\frac{\eta_2 \nu_3^s}{\eta_1} + \left(\frac{c^s - c_3^s}{c^s c_3^s}\right)\Omega_3^s \tag{3.53}$$

which is true if and only if $m_1^s = m_2^s = m^s$. Rearranging (3.53) for Ω_3^s:

$$\Omega_3^s = \nu_3^s \left(\frac{c^s\, \eta_2 (m^s - m_3^s)}{m^s\, m_3^s \eta_1} + \frac{\eta_1}{\eta_2}\right) \tag{3.54}$$

The following condition must also be true:

$$\frac{\mu_1}{\mu_2} = \frac{\mu_4}{\mu_5} \tag{3.55}$$

The particular solution for the Hamiltonian vector flow (3.47) can now be expressed as:

$$\begin{aligned} \mu_1 &= -\eta_2 \cos\theta, \quad \mu_2 = \eta_2 \sin\theta, \quad \mu_3 = \nu_3^s, \\ \mu_4 &= -\eta_1 \cos\theta, \quad \mu_5 = \eta_1 \sin\theta, \quad \mu_6 = \Omega_3^s, \end{aligned} \tag{3.56}$$

where $\theta = \left(\frac{\Omega_3^s}{c_3^s} - \frac{\nu_3^s s}{c^s \eta_2}\right) t + \beta$ and where $\beta = \operatorname{atan2}\left(-\mu_5(0),\, \mu_4(0)\right)$. The rotation matrix R^s in terms of Euler angles ϕ_1^s, ϕ_2^s and ϕ_3^s can be expressed according to the formula:

$$R^s = e^{\phi_1^s E_3} e^{\phi_2^s E_2} e^{\phi_3^s E_3} \tag{3.57}$$

then substituting (3.57) into (3.29) yields:

$$P = Ke^{-\phi_3^s E_3}\, e^{-\phi_2^s E_2}\, E_3 e^{\phi_2^s E_2}\, e^{\phi_3^s E_3} \tag{3.58}$$

where $K = I_2^2$ and yields:

$$P = K \begin{pmatrix} 0 & -\cos\phi_2^s & \sin\phi_2^s \sin\phi_3^s \\ \cos\phi_2^s & 0 & \sin\phi_2^s \cos\phi_3^s \\ -\sin\phi_2^s \sin\phi_3^s & -\sin\phi_2^s \cos\phi^s & 0 \end{pmatrix} \tag{3.59}$$

then:

$$\begin{aligned} \eta_2 \cos\theta &= \sin\phi_2^s \cos\phi_3^s, \\ \eta_2 \sin\theta &= \sin\phi_3^s \sin\phi_2^s, \\ \nu_3^s &= K\cos\phi_2^s \end{aligned} \tag{3.60}$$

From (3.60) the following expressions can be obtained:

$$j_1 = \cos\phi_2^s = \frac{\mu_3(0)}{\sqrt{\eta_2^2 + \mu_3(0)^2}} \tag{3.61}$$

$$j_2 = \sin\phi_2^s = \frac{\eta_2}{\sqrt{\eta_2^2 + \mu_3(0)^2}} \tag{3.62}$$

$$\phi_3^s = \theta \tag{3.63}$$

To find ϕ_1^s, substitute (3.57) into (2.5) to obtain the following relationships:

$$\begin{aligned} \sin\phi_2^s \sin\phi_3^s \dot{\phi}_1^s + \cos\phi_3^s \dot{\phi}_2^s &= \Omega_2^{s*}, \\ -\sin\phi_2^s \cos\phi_3^s \dot{\phi}_1^s + \sin\phi_3^s \dot{\phi}_2^s &= \Omega_1^{s*} \end{aligned} \tag{3.64}$$

which can be manipulated and simplified to:

$$\dot{\phi}_1^s = \frac{\eta_1 \sqrt{\eta_2^2 + \mu_3(0)^2}}{\eta_2 c^s} \tag{3.65}$$

If it is assumed $\phi_1^s(0) = 0$, the particular solution is:

$$\phi_1^s = \gamma t \tag{3.66}$$

where $\gamma = \frac{\eta_1 \sqrt{\eta_2^2 + \mu_3(0)^2}}{\eta_2 c^s}$. The translation position $\mathbf{x^s}$ is found as an analytical

function using the method described in the previous section:

$$\begin{aligned}
x_1^s &= \frac{1}{m^s m_3^s \gamma}\Bigg(2\big(j_2 m^s \nu_3^s - j_1 m_3^s \eta_2\big)\sin\beta \sin\left(\frac{t\gamma}{2}^2\right) + \cos\beta\Big(- j_2\big(j_1 m^s \nu_3^s + j_2 m_3^s \eta_2\big)t\gamma \\
&\qquad + j_1\big(j_2 m^s \nu_3^s - j_1 m_3^s \eta_2\big)\sin t\gamma\Big)\Bigg), \\
x_2^s &= \frac{1}{m^s m_3^s \gamma}\Bigg(2\big(j_2 m^s \nu_3^s - j_1 m_3^s \eta_2\big)\cos\beta \sin\left(\frac{t\gamma}{2}^2\right) + \sin\beta\Big(j_2\big(j_1 m^s \nu_3^s + j_2 m_3^s \eta_2\big)t\gamma \\
&\qquad + j_1\big(j_1 m_3^s \eta_2 - j_2 m^s \nu_3^s\big)\sin t\gamma\Big)\Bigg), \\
x_3^s &= \frac{1}{m^s m_3^s \gamma}\Bigg(j_1\big(j_1 m^s \nu_3^s + j_2 m_3^s \eta_2\big)t\gamma + j_2 \sin t\gamma\big(j_2 m^s \nu_3^s - j_1 m_3^s \eta_2\big)\Bigg)
\end{aligned} \tag{3.67}$$

Analytical expressions for the translational velocities are obtained by differentiating (3.67):

$$\begin{aligned}
\dot{x}_1^s &= \frac{1}{m^s m_3^s \gamma}\Bigg(2\big(j_2 m^s \nu_3^s - j_1 m_3^s \eta_2\big)\sin\beta \sin\left(\frac{t\gamma}{2}^2\right) + \cos\beta\Big(- j_2\big(j_1 m^s \nu_3^s + j_2 m_3^s \eta_2\big)t\gamma \\
&\qquad + j_1\big(j_2 m^s \nu_3^s - j_1 m_3^s \eta_2\big)\sin t\gamma\Big)\Bigg), \\
\dot{x}_2^s &= \frac{1}{m^s m_3^s \gamma}\Bigg(2\big(j_2 m^s \nu_3^s - j_1 m_3^s \eta_2\big)\cos\beta \sin\left(\frac{t\gamma}{2}^2\right) + \sin\beta\Big(j_2\big(j_1 m^s \nu_3^s + j_2 m_3^s r\big)t\gamma \\
&\qquad + j_1\big(j_1 m_3^s r - j_2 m^s \nu_3^s\big)\sin t\gamma\Big)\Bigg), \\
\dot{x}_3^s &= \frac{1}{m^s m_3^s \gamma}\Bigg(j_1\big(j_1 m^s \nu_3^s + j_2 m_3^s \eta_2\big)t\gamma + j_2 \sin t\gamma\big(j_2 m^s \nu_3 - j_1 m_3^s \eta_2\big)\Bigg)
\end{aligned} \tag{3.68}$$

The acceleration for each axis can be obtained by differentiating (3.68) but is not given here. The magnitude of velocity and acceleration are:

$$\|\nu^{\mathbf{s}}\| = \sqrt{\frac{\nu_3^{s2}}{m_3^{s2}} + \frac{\eta_2^2}{m^{s2}}} \tag{3.69}$$

$$\|\dot{\nu}^{\mathbf{s}}\| = \sqrt{\frac{\gamma^2}{m^{s2} m_3^{s2}}\Big(m_3^{s2}\eta_2^2 - 2j_1 j_2 m^s m_3^s \nu_3^s \eta_2 + j_2\big(m^{s2}\nu_3^{s2} - m_3^{s2}\eta_2^2\big)\Big)} \tag{3.70}$$

It is possible to project the attitude R^s, as a function of t from the solution on SE(3) as given for the case of a single translation body velocity in the previous section. However, it is a large expression and not used in later sections so it is

has been omitted from this thesis.

3.4 Reachable sets

For the purpose of this thesis, reachable sets are defined as the states achievable starting from the origin, for a single trajectory given some bounds applied to the controls. It is assumed that the configuration space is free from obstacles for the purposes of defining reachable sets. The reachable sets for both types of curves defined in Section 3.2 and Section 3.3 are found numerically using a Monte Carlo approach. The trajectory time is fixed as $T = 1$ and the magnitude of the translational body velocity is also fixed at unit speed, $\|\nu^{\mathbf{s}}\| = 1$. Any weightings relevant to the curve (c^s, c_3^s, m^s, m_3^s) are fixed and the same for all trajectories.

3.4.1 Single body velocity reachable set

The curve weighting and surge direction body velocity were fixed as $c^s = 1$ and $\nu_1^s = 1\,\mathrm{m/s}$ respectfully. The three extremals that describe the shape of the curve were defined using the random number generator **rand** built into Matlab and constrained with the following bounds:

$$0 < \mu_3(0) < 1, \;\; 0 < \mu_5(0) < c^s, \;\; 0 < \mu_6(0) < c^s \tag{3.71}$$

Extremals $\mu_5(0)$ and $\mu_6(0)$ have upper bounds of c^s because that limits the controls Ω_2^s and Ω_3^s to have a maximum value of 1 from (3.11). A thousand combinations of (3.71) were generated and plotted in Figure 3.1. All trajectories start at the origin, with an initial velocity vector equal to that of x_1^s before diverging. It should be noted that the reason all the trajectories remain within $x_2^s(t) \geq 0$ is due to the bounds on the extremals having minimums of zero (3.71).

3.4.2 Multiple body velocities reachable set

The weightings for all the trajectories generated were fixed as $c^s = 1$, $c_3^s = 1$, $m^s = 2$, $m_3^s = 1$. The process of choosing the random extremals was slightly more

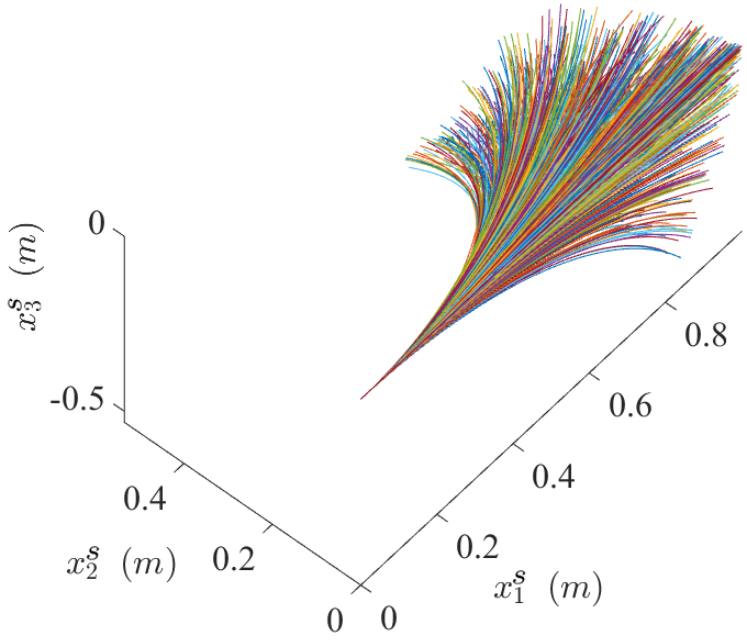


Figure 3.1: Single velocity sub-Riemannian curve reachable set – 1000 random trajectories generated with the bounds (3.71) to numerically illustrate the reachable set for the curves defined in Section 3.2.

involved than that of the previous case because the magnitude of the velocity is a function of all the individual body-axis velocities (3.69). Consequently the choices of the extremals were dependent upon each other, therefore the following process was used to ensure the magnitude of translational velocity was unit speed. The first step was to generate a random velocity in the third body direction within the following bounds:

$$0 < \nu_3^s < 1 \tag{3.72}$$

Then the first body-fixed velocity is generated within the following bounds:

$$0 < \mu_1(0) < m^s \sqrt{\|\nu^{\mathbf{s}}\|^2 - \frac{\nu_3^{s2}}{m_3^{s2}}} \tag{3.73}$$

The third body velocity can be found by rearranging (3.69):

$$\mu_2(0) = m^s \sqrt{\|\nu^{\mathbf{s}}\|^2 - \frac{\nu_3^{s2}}{m_3^{s2}} - \frac{\mu_1(0)^2}{m^{s2}}} \tag{3.74}$$

The fourth extremal is generated within the bounds:

$$0 < \mu_4 < c^s \tag{3.75}$$

which means $\mu_5(0)$ is calculated by satisfying (3.55) rearranged to:

$$\mu_5(0) = \frac{\mu_2(0)\mu_4(0)}{\mu_1(0)} \tag{3.76}$$

All the parameters of the curve have now been defined. Again, 1000 combinations of parameters were randomly generated and the results plotted in Figure 3.2. Compared to the single speed case (Figure 3.1) which has similar constraints, there is a greater degree of flexibility because the initial direction of the velocity in the inertial frame has more freedom. This reachable set is also limited by the minimum bounds of the conditions applied when randomly generating the parameters so the trajectories remained bounded to $x_1^s(T) < 0$. For the purposes of motion planning these bounds are relaxed, so a greater set of final positions can be met.

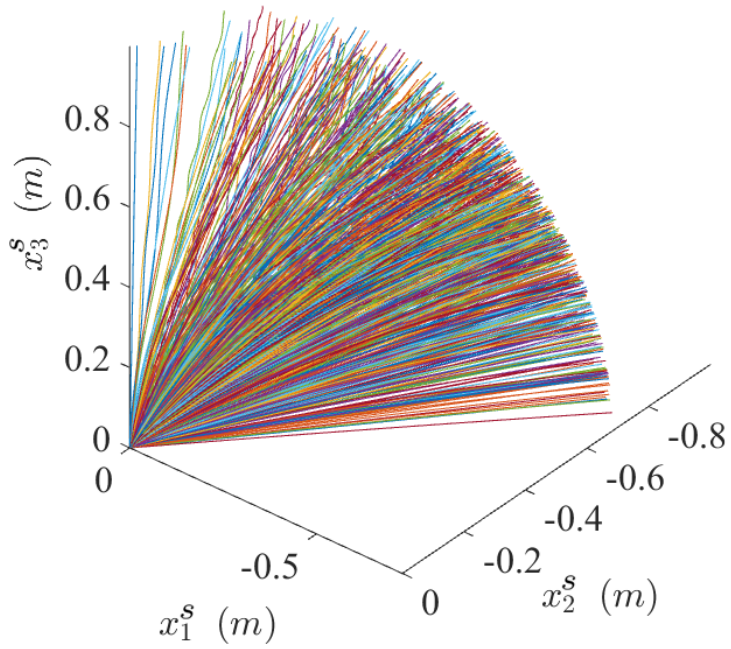


Figure 3.2: Multiple velocity sub-Riemannian curve reachable set – 1000 random trajectories generated using bounds given in (3.72)-(3.76) to numerically illustrate the reachable set for the curves defined in Section 3.3.

3.5 Parametric optimisation

Parametric optimisation can be used to find a curve that goes from the origin to a desired translational state. The free parameters in the analytical expressions that describe the states form the optimisation vector $\Xi_{1,2}$, which for the single velocity case is:

$$\Xi_1 = [\mu_1(0),\ \mu_5(0),\ \mu_6(0),\ T]^T \tag{3.77}$$

and for the multiple velocity case:

$$\Xi_2 = [\mu_2(0),\ \mu_4(0),\ \mu_5(0),\ T]^T \tag{3.78}$$

where T is the final time, recalling $t \in [0, T]$. When the optimisation vector and the appropriate weightings are chosen the curve is fully defined because any other parameters can calculated from these. During the optimisation process the optimisation vector is controlled by the solver that tries to minimise some cost function. In this section two cost functions are introduced, the first optimises for the final translational position and the second also considers the final translational velocity.

3.5.1 Solving for final position

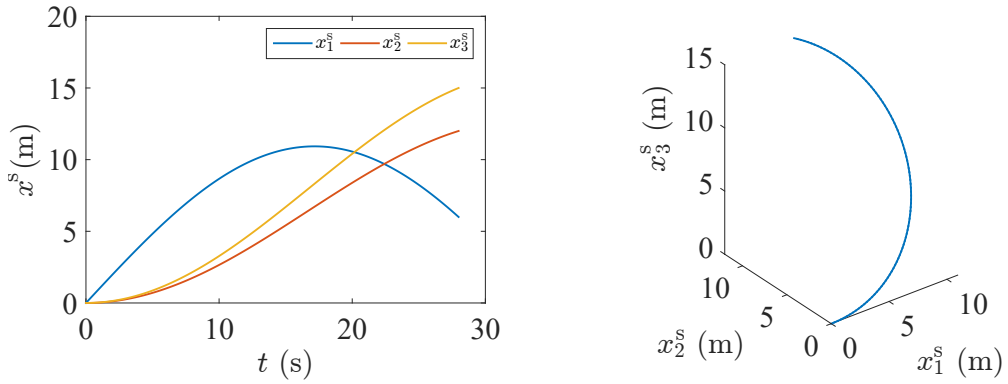
The cost function for matching the final position is simply the Euclidean distance between the final position of a curve and the desired position:

$$J_x(\Xi_{1,2}) = \|\mathbf{x^s}_{des} - \mathbf{x^s}(T)\| \tag{3.79}$$

where $\mathbf{x^s}_{des}$ is the desired final position. The minimisation problem is therefore:

$$\begin{aligned} &\underset{\boldsymbol{\Xi}_{1,2}}{\text{minimise}} && J(\boldsymbol{\Xi}_{1,2})_x \\ &\text{subject to} && \boldsymbol{\Xi}_{1,2,min} \leq \boldsymbol{\Xi}_{1,2} \leq \boldsymbol{\Xi}_{1,2,max} \end{aligned} \tag{3.80}$$

where $\boldsymbol{\Xi}_{1,2,min} = [-\infty, -\infty, -\infty, 0]^T$ and $\boldsymbol{\Xi}_{1,2,max} = [\infty, \infty, \infty, T]^T$ are the minimum and maximum bounds on the optimisation vector. A variety of numerical optimisers were tested and Matlab's **fmincon** was found to perform well for both types of curves and converged on the solution in a reasonable amount of

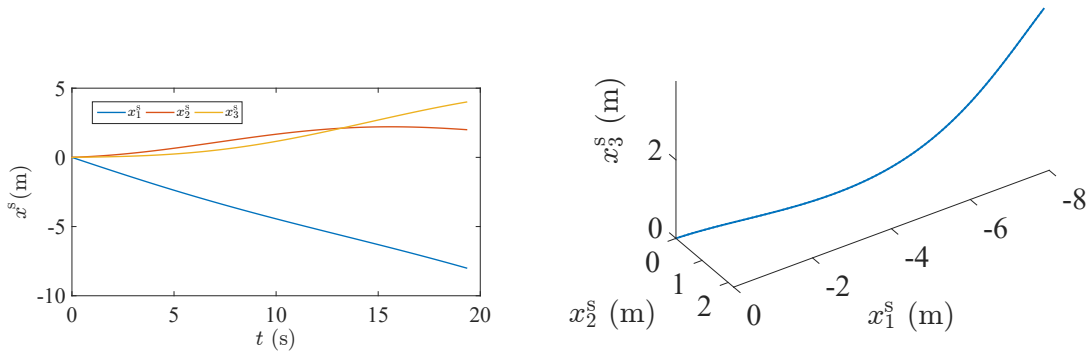


(a) Position against time. **(b)** Trajectory.

Figure 3.3: Single velocity position optimisation example – The trajectory starts at the origin and finishes at the desired translational position $\mathbf{x^s_{des}} = [6\,\text{m},\ 12\,\text{m},\ 15\,\text{m}]^T$.

computation time. It is capable of handling the linear inequality constraint on the solution that the trajectory time must be greater than or equal to zero. An initial guess of $\mathbf{\Xi}_{1,2} = [1,\ 1,\ 1,\ 1]^T$ was found to be effective for a wide range of desired positions and was therefore chosen as the initial guess for both types of curve.

If $\mathbf{x}^s_{des} = [6\,\text{m},\ 12\,\text{m},\ 15\,\text{m}]^T$ is the desired position, $\nu_1 = 1\,\text{m/s}$ and weighting $c = 200$ is chosen, then for the single velocity case the numerical optimiser found $\mathbf{\Xi}_1 = [16.6665,\ -14.1895,\ 11.5624,\ 27.9885]^T$ to be the optimisation vector that generated a curve finished closest to the target position. The trajectory and position against time is shown in Figure 3.3. The solution time was $0.56\,s$ on a standard 3.8 Ghz desktop computer. Similarly, for the multiple velocities case with weightings $\nu^s_3 = 1\,\text{m/s}$, $c^s = 20$, $c^s_3 = 1$, $m^s = 5$, $m^s_3 = 50$ and target position $\mathbf{x^s_{des}} = [-8\,\text{m},\ 2\,\text{m},\ 4\,\text{m}]^T$, the optimisation vector was found to $\mathbf{\Xi}_2 = [0.2574,\ 3.9942,\ -0.4156,\ 19.3300]^T$. The solution time was $0.20\,s$ and the trajectory and position against time was plotted in Figure 3.4. These are two simple examples but it is important to appreciate that although the derivation of the curves is mathematically involved, using them in practice is simple.



(a) Position against time. **(b)** Trajectory.

Figure 3.4: Multiple velocity position optimisation example – The trajectory starts at the origin and finishes at the desired translational position $\mathbf{x^s_{des}} = [-8\,\text{m},\, 2\,\text{m},\, 4\,\text{m}]^T$.

3.5.2 Solving for final position and biasing the velocity

The following cost function considers both the final translational position and velocity:

$$J_{x^s,\dot{x}^s}(\Xi_{1,2}) = j_{x^s} \left\|\mathbf{x^s}_{des} - \mathbf{x^s}(T)\right\| + j_{\dot{x}^s} \left\|\dot{\mathbf{x}}^{\mathbf{s}}_{des} - \dot{\mathbf{x}}^{\mathbf{s}}(T)\right\| \tag{3.81}$$

where $\mathbf{x^s}_{des}$ is the desired final position and $\dot{\mathbf{x}}^{\mathbf{s}}_{\mathbf{des}}$ is the desired velocity. The weightings j_{x^s} for position and $j_{\dot{x}^s}$ for velocity are necessary because it is important that the velocity matching does not dominate the minimisation. If this occurs then the position will not be matched because using the curves derived in this chapter it is not possible to match the final velocity. As discussed previously, the inclusion of velocity into the cost function allows the shape of the curve to be biased while still reaching the desired final position if suitable weightings are chosen. The minimisation problem for this cost function is:

$$\begin{aligned} &\underset{\boldsymbol{\Xi}_{1,2}}{\text{minimise}} \quad J(\boldsymbol{\Xi}_{1,2}) \\ &\text{subject to} \quad \boldsymbol{\Xi}_{1,2,min} \leq \boldsymbol{\Xi}_{1,2} \leq \boldsymbol{\Xi}_{1,2,max} \end{aligned} \tag{3.82}$$

where $\boldsymbol{\Xi}_{1,2,min} = [-\infty, -\infty, -\infty, 0]^T$ and $\boldsymbol{\Xi}_{1,2,max} = [\infty, \infty, \infty, T]^T$ are the minimum and maximum bounds on the optimisation vector. The same minimiser used for optimising the position that was discussed previously is used for this minimisation problem. One option to improve the convergence when both posi-

tion and velocity are considered would be to use a general solution instead of the particular solutions that were derived. However, since numerical minimisers must recalculate the functions that describe the position and velocity for every iteration of the optimisation vector, the hyperbolic functions required for the general solution would take longer to solve compared to the trigonometric functions needed for the particular solution. Example trajectories for the minimisation problem are not given for this section because the simple obstacle avoidance method in the next section uses the minimisation problem (3.82).

3.6 Obstacle avoidance motion planning

The cost function introduced in Section 3.5.2 is used to find a curve with a final position and also influence the shape of the curve by including an error term for the direction of the final velocity. As discussed previously, with these curves it is not possible to satisfy a desired final velocity direction in the inertial frame but the bias introduced can be used as a simple obstacle avoidance method. This is demonstrated for both types of curves.

After defining the obstacles and desired final position the first step is generating random unit velocity vectors:

$$\mathbf{v_{des}} = \begin{bmatrix} \cos\phi_1^r \cos\phi_3^r \\ \sin\phi^r \sin\phi_1^r \cos\phi_3^r - \cos\phi_2^r \sin\phi_3^r \\ \cos\phi_2^r \sin\phi_1^r \cos\phi_3^r + \sin\phi_2^r \sin\phi_3^r \end{bmatrix} \tag{3.83}$$

where ϕ_1^r, ϕ_2^r and ϕ_3^r are scalar constants randomly generated between the bounds $0 \leq \phi_{1,2,3}^r \leq 2\pi$. In this example 50 trajectories are then generated using a numerical optimiser solving the minimisation problem (3.82) and ranked by their path length. Any trajectories that enter the collision space are rejected. In this thesis the shortest path is selected but a different metric could be used.

Two examples of this obstacle avoidance method are given. The first uses the curves with a single body velocity direction starting at the origin and finishing at $\mathbf{x_{des}^s} = [6\,\text{m},\ 12\,\text{m},\ 6\,\text{m}]^T$. The obstacles are three spheres with the properties are given in Table 3.3 and the weighting for all curves was $c^s = 200$ and the surge velocity $\nu_1^s = 1\,\text{m/s}$. Figure 3.5 shows all the trajectories generated, ranked in

colour by their path length. The next step of removing those trajectories which collide with the obstacles is given in Figure 3.6 and the shortest trajectory plotted in Figure 3.7.

$x_1^s\,(\mathsf{m})$	$x_2^s\,(\mathsf{m})$	$x_3^s\,(\mathsf{m})$	$r_o\,(\mathsf{m})$
8.4	7.0	3.8	0.6
9.1	8.7	5.4	0.7
10.7	7.0	7.0	5.0

Table 3.3: Obstacle Avoidance: single body-velocity case obstacles – The translational position and size of the obstacles.

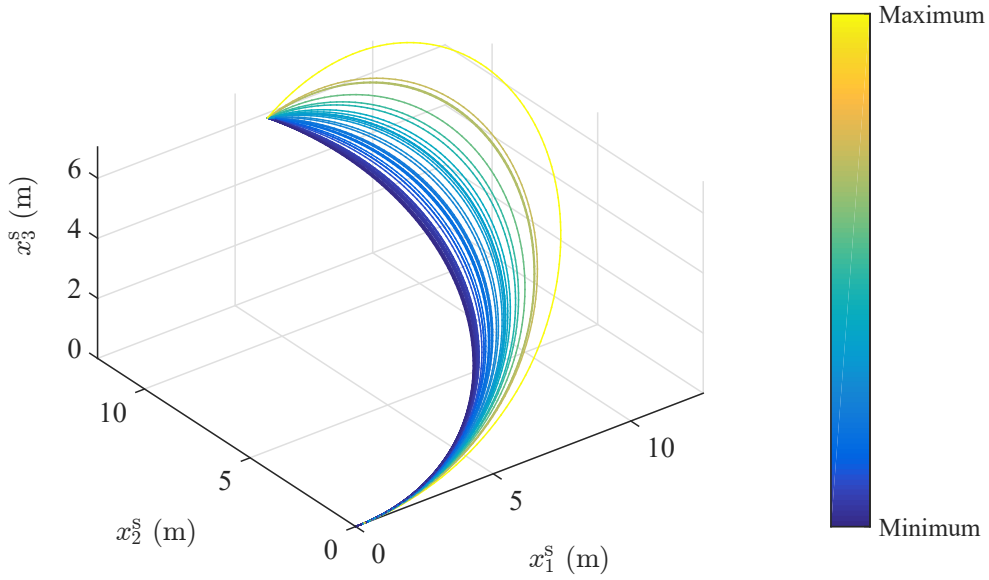


Figure 3.5: Obstacle avoidance single body velocity: all trajectories – 50 trajectories generated with randomly chosen biasing velocities to change their shape. The paths are coloured coded from the shortest to longest geometrical lengths.

The simple obstacle avoidance method can also be applied to the multiple body-velocity curves. In this example the trajectory starts at the origin and the desired final position is $\mathbf{x}_{\mathbf{des}}^{\mathbf{s}} = [-8\,\mathrm{m},\, 2\,\mathrm{m},\, 4\,\mathrm{m}]^T$. The weightings chosen are $c^s = 20$, $c_3^s = 1$, $m^s = 5$, $m_3^s = 50$ and the body-velocity was fixed at $\nu_3^s = 1\,m/s$. Figure 3.8 contains the 50 randomly generated trajectories with the velocity biasing vector found using (3.83). The trajectories were then checked

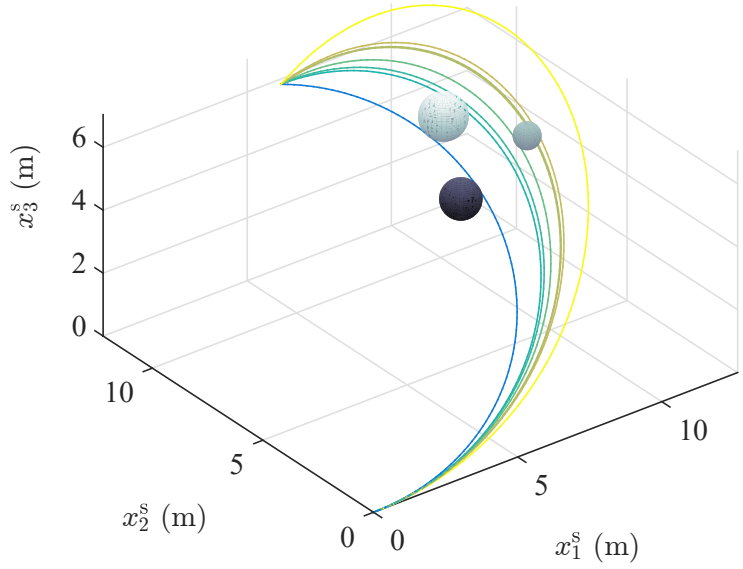


Figure 3.6: Obstacle avoidance single body-velocity: collision free trajectories – The trajectories that collide with obstacles in the collision space are rejected and those that remain plotted with the spherical obstacles present.

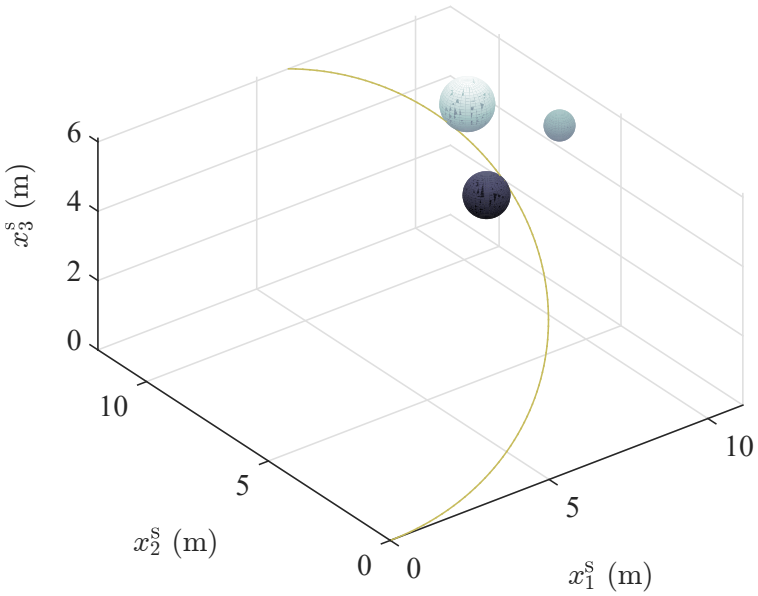


Figure 3.7: Obstacle avoidance single body-velocity: shortest, collision free path – The shortest trajectory generated from the origin to desired final position $\mathbf{x}_{\mathbf{des}}^{\mathbf{s}} = [6\,\text{m},\ 12\,\text{m},\ 6\,\text{m}]^T$.

with the obstacles (Table 3.4) and those that collided were removed (Figure 3.9). The shortest, collision-free trajectory is then given in Figure 3.10.

Comparing the range of trajectories for the two different types of curves (Figure 3.8 and Figure 3.5) it can be seen that the multiple body-velocity type produces a greater variety of curves so where possible it should be used for this kind of collision avoidance. However, both types of curves are unable to directly account for the obstacles and instead rely on at least one trajectory being feasible. If this is not so, a different approach must be taken such as the RRT method described in the next section.

$x^s_{o,1}\,(\mathsf{m})$	$x^s_{o,2}\,(\mathsf{m})$	$x^s_{o,3}\,(\mathsf{m})$	$r_o\,(\mathsf{m})$
-3.0	1.4	0.5	0.2
-5.4	1.1	2.7	0.3
-5.7	2.0	2.0	0.4
-4.0	1.6	2.5	0.7
-4.6	1.0	3.0	0.6

Table 3.4: Obstacle Avoidance: multiple body-velocity case obstacles – The translational position and size of the obstacles.

3.7 Chapter summary

In this chapter motion planning methods for vehicles in three dimensions defined on Lie groups with kinematic constraints were considered using optimal control theory. Two variations of kinematic constraints were developed: (i) when two body angular velocities and one translational body-velocity was non zero and; (ii) when all translational and angular body velocities are non-zero. The curves are analogous to sub-Riemmanian curves but are parametrised by time, not length. The analytical functions that describe the states are trigonometric because particular solutions are chosen. This means only a subset of the general solutions to the optimal curves possible are available given the kinematic constraints applied. Future work could consider the particular solutions but if so, hyperbolic functions would be required that are not well supported by numerical computing software, in particular that of on-board hardware.

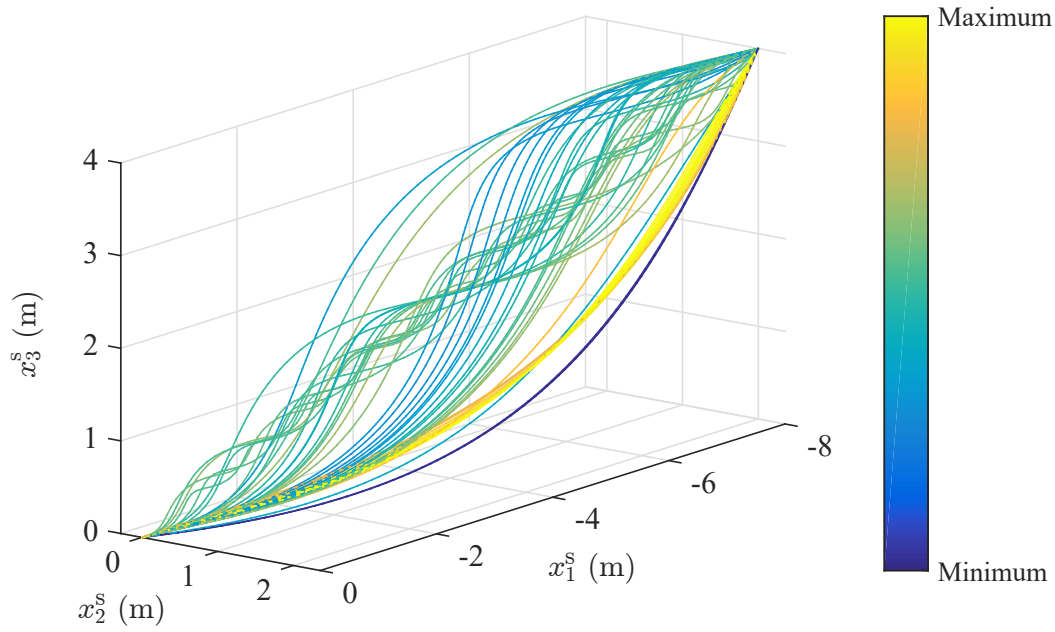


Figure 3.8: Obstacle avoidance multiple body-velocities: all trajectories – 50 trajectories generated with randomly chosen biasing velocities to change their shape.

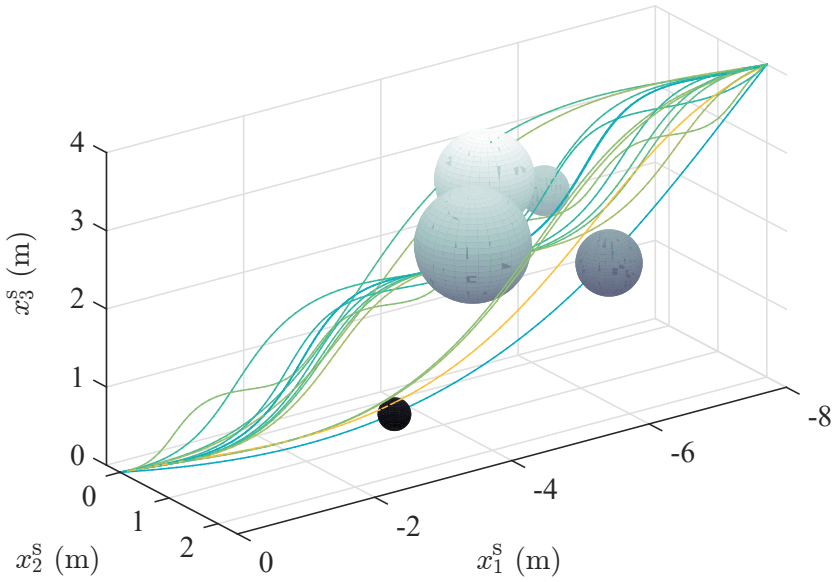


Figure 3.9: Obstacle avoidance multiple body-velocities: collision free trajectories – The trajectories that collide with obstacles in the collision space are rejected and those that remain plotted with the spherical obstacles present.

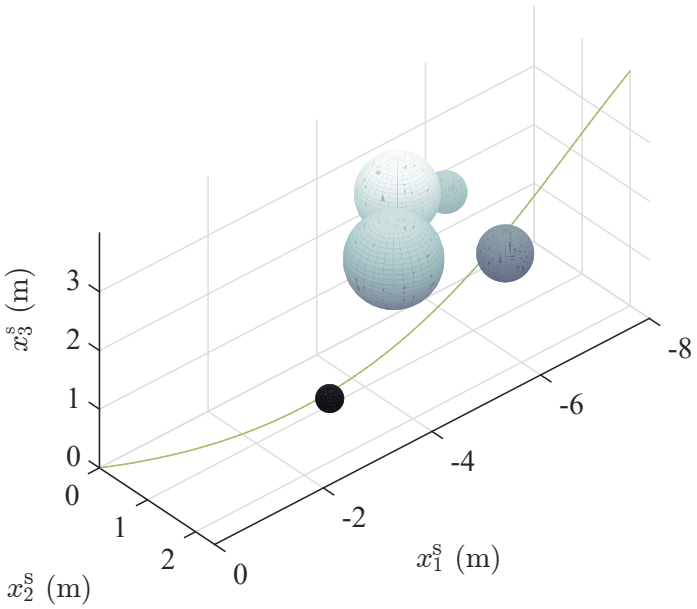


Figure 3.10: Obstacle avoidance multiple body-velocities: shortest, collision free path – The shortest trajectory generated from the origin to desired final position $\mathbf{x}^{\mathbf{s}}_{\mathbf{des}} = [-8\,\text{m},\, 2\,\text{m},\, 4\,\text{m}]^T$.

Reachable sets for unit speed and unit time conditions for both cases were found. Unsurprisingly, the case of multiple body velocities is capable of reaching a greater range of final translational states for similar conditions compared to the single velocity case. Therefore, when the vehicle kinematics and dynamics allow for it, the multiple body velocity case should be chosen over the single body velocity case.

Two parametric optimisations were introduced: i) when a final translational position in the inertial frame is desired ii) when a final translational position in the inertial frame is desired and the final velocity is also biased in the cost function. Optimising for the final translational position was demonstrated for both types of curves and the case when the final velocity is biased was used as the basis of an obstacle avoidance strategy for both types of curve. This simple obstacle avoidance method used spheres to represent obstacles in the environment. Again, the variation of curve shapes was greater for the multiple translational body velocity, making it more suitable for this kind of obstacle avoidance if the kinematics of the vehicles permits it. However, this method is limited when many obstacles are present because a single trajectory found using this method

is unable to significantly change its shape. To overcome the limitations of the simple obstacle avoidance method, the single velocity curves were combined with a sampling-based path planner in Chapter 4.

Chapter 4

Additional application: AUV path planning

Although the focus of this thesis is on quadrotor flying vehicles, the curves developed using optimal control theory in the previous chapter lend themselves to trajectory planning for underwater vehicles so this chapter will explore their use in that area. The sub-Riemannian curves from this chapter were defined on SE(3) which includes the translational and rotational position as a function of time. In that chapter the solution was projected onto $\mathbf{x^s} \in \mathbb{R}^3$ but the AUVs used in this chapter can use the rotational information contained within the curve for the purposes of trajectory tracking. For neutrally buoyant, long, slender AUVs such as C-SCOUT [129], the lateral body-fixed motions ν_2^s and ν_3^s are quickly damped out and it can only produce thrust in the direction of the first body-axis. This means that the nonholonomic constraints in Section 3.2 match those of the vehicle so the rotational position from those curves can be used.

It was noted that the obstacle avoidance method in Chapter 3 was limited in the presence of a dense obstacle field. This is because the obstacles are not directly accounted for by the path planner, instead it relies upon at least one trajectory being valid. In this chapter the sub-Riemannian curves are used as a local planner within the sampling-based Rapidly-exploring Random Tree (RRT) framework. The trajectories no longer need to be limited to a single curve but can consist of concatenated segments allowing more flexibility and range of trajectory shapes.

Original Contributions

The original contribution in this chapter is:

- A path planning framework for AUVs that considers both attitude and translational position is developed by combining the single velocity curves with a sampling-based motion planner. The method is capable of finding feasible paths through a field of obstacles by concatenating individual segments into a single trajectory.

The chapter is structured as follows. In Section 4.1 the reference frame and kinematic model of the AUV is introduced. Section 4.2 introduces the RRT method and uses it to develop an obstacle avoidance framework for the sub-Riemannian curves. Finally, a summary of the chapter is given in Section 4.3.

4.1 AUV model

In this section a basic model of an AUV is presented. Figure 4.1 shows a long, slender AUV that is representative of the C-SCOUT [129]. The vehicle's centre of mass is $\mathbf{x^s} = [x_1^s, x_2^s, x_3^s]^2$ and the vehicle's attitude is $R^s = [\mathbf{b_1^s}, \mathbf{b_2^s}, \mathbf{b_3^s}]$ where $\mathbf{b_{1,2,3}^s} \in \mathbb{R}^3$ are orthogonal body frame vectors. This reference frame is equivalent to the quadrotor reference frame presented in Figure 2.1a but for an AUV instead.

If the vehicle is assumed to be neutrally buoyant and it is also assumed that the dynamic effects are damped out, then the dynamics can be neglected and a simple, kinematic model can be used to express its motion:

$$\dot{\mathbf{x}}^{\mathbf{s}} = \mathbf{v}^{\mathbf{s}} \tag{4.1}$$

$$\dot{R}^s = R^s \hat{\Omega}^s \tag{4.2}$$

where $\mathbf{v^s} = [v_1^s, v_2^s, v_3^s]^T$ is the translation velocity in the inertial frame and the angular velocities about the body axes $\hat{\Omega}^s$ which is the skew symmetric matrix of $\mathbf{\Omega^s} = [\Omega_1^s, \Omega_2^s, \Omega_3^s]^T$.

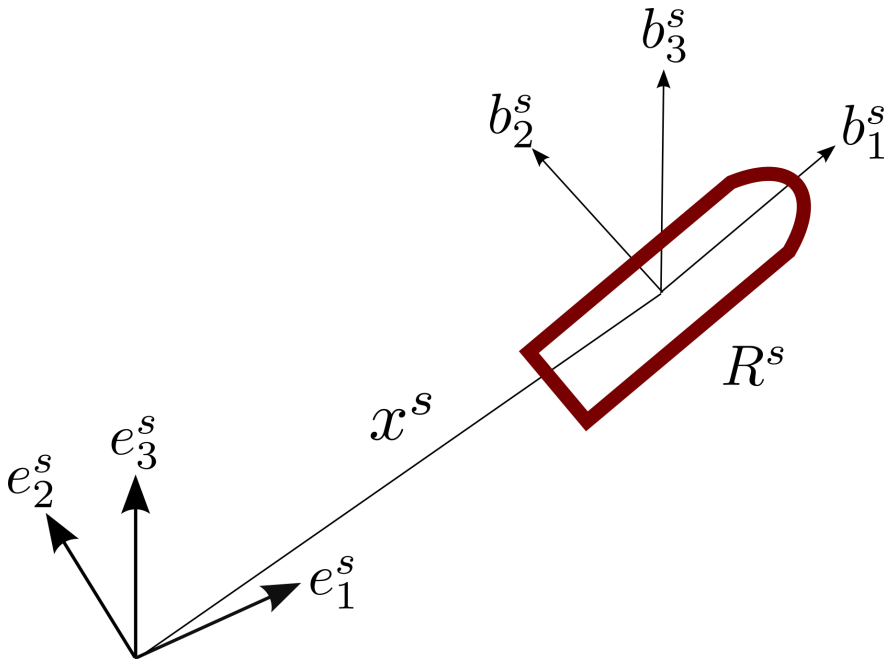


Figure 4.1: Sub-Riemmanian curve AUV reference frame – The AUV has translational position $\mathbf{x^s}$ and rotational position R^s.

4.2 RRT for underwater vehicles

In this section the curves derived in Section 3.2 in which the body fixed translational velocity is fixed to motion in one direction are used within a sampling-based planning framework, RRT. The trajectories are formed by concatenating local curves that are generated using the parametric optimisation described in Section 3.5.

Before introducing the method used to incorporate the sub-Riemmanian curves into the RRT algorithm it is useful to introduce the standard RRT algorithm which is constructed as follows. A starting position is defined with the vehicle's initial state in the inertial reference frame. For this problem the initial state is position $\mathbf{x^s}(0)$ and orientation $R^s(0)$. The goal position $\mathbf{x^s_{goal}}$ is also chosen which is the desired position at the end of the trajectory. There are two methods for generating target positions $\mathbf{x^s_{target}}$ in order to extend the tree. They can either be sampled randomly from the free configuration space or chosen with a deterministic algorithm. Examples of deterministic sampling algorithms include the simple uniform spacing method or one that accounts for the environment and

current solution by biasing the location of $\mathbf{x}^{\mathbf{s}}_{\mathbf{target}}$ towards where it believes it should be. Once $\mathbf{x}^{\mathbf{s}}_{\mathbf{target}}$ has been chosen a search is performed on the existing nodes to determine which is closest to the target node according to some metric and is designated $\mathbf{x}^{\mathbf{s}}_{\mathbf{closest}}$. This node is chosen as the point the tree will branch out from during that particular iteration. A segment between this node and the target node is computed using a time integrated simulation of the vehicle. If the segment is deemed acceptable (that is, it doesn't collide with obstacles and is feasible according to any other constraint) then it is added to the tree. A given number of points along that segment are defined as nodes that can be used as $\mathbf{x}^{\mathbf{s}}_{\mathbf{closest}}$ in future iterations. This basic algorithm works well for vehicles with holonomic constraints. However, for the majority of systems in the real world like AUVs with kinematic constraints the time integration simulation can be computationally intensive because the controls must be calculated for each time step. Since this integration is performed many times to create a tree of trajectories, the overall computational time is significant, especially for embedded on-board hardware.

In this chapter the standard RRT algorithm is used, with one significant change. Instead of performing a time integrated simulation to generate segments, parametric optimisation of the minimisation problem (3.80) is performed. This is achieved by generating a curve from the closest node to the target node. In doing so, a segment can be calculated faster than a time integration simulation. Since the curves are defined analytically, they can be efficiently discretised to check for proximity to obstacles and rejected if a collision is detected. A random search of the environment was used to generate target nodes because although it may take longer than a deterministic algorithm, no heuristics need to be tuned that are specific to the problem. A certain percentage of the target nodes are located at $\mathbf{x}^{\mathbf{s}}_{\mathbf{goal}}$. The optimum value for this scalar bias is referred to as $\mathbf{x}^{\mathbf{s}}_{\mathbf{bias}}$ and could be obtained through numerical experimentation, using a Monte Carlo approach for instance because it is not possible to determine analytically. Optimising x^s_{bias} was left as future work for the purposes of this thesis and instead a few values were tested before settling upon the one used. If x^s_{bias} is too large, then the environment will not be properly explored and tend to get stuck in dead-ends. If x^s_{bias} is too small then it can take a long time for the algorithm to converge on the goal state.

A Euclidean distance metric is used to determine which of the nodes on the tree is closest to the target:

$$J_{RRT} = \left\| \mathbf{x}^{\mathbf{s}}_{\mathbf{target}} - \mathbf{x}^{\mathbf{s}}_{\mathbf{n}} \right\| \tag{4.3}$$

where $\mathbf{x}^{\mathbf{s}}_{\mathbf{n}}$ is the position vector of the node on the tree currently being considered. Although this metric is simple to compute, the orientation of the vehicle is not considered. Given the nonholonomic constraints, the Euclidian distance metric (4.3) can recommend a node $\mathbf{x}^{\mathbf{s}}_{\mathbf{closest}}$ that does not result in the shortest path to $\mathbf{x}^{\mathbf{s}}_{\mathbf{target}}$. Other metrics have been proposed [54, 130] but for the purposes of demonstrating the method it was decided that the Euclidean metric is adequate. Further research is necessary to determine the benefits of using a more sophisticated metric that accounts for the rotational position, not just translational.

To demonstrate the RRT path planner with the single body-velocity curves an example shall be be given. Further examples can be found in the paper [131]. The initial position of vehicle is at the origin $\mathbf{x}^{\mathbf{s}}(0) = [0\,\mathrm{m},\, 0\,\mathrm{m},\, 0\,\mathrm{m}]^T$, $R^s(0) = I_{3\times 3}$ and the goal position is $\mathbf{x}^{\mathbf{s}}_{\mathbf{goal}} = [5\ \mathrm{m}, 5\,\mathrm{m}, 5\,\mathrm{m}]^T$. The obstacles were defined as spheres and are given in Table 4.1. The entire tree and obstacles are plotted in Figure 4.2. At this stage an exploration of the state space has been performed and a path has been found that reaches the goal state and avoids the obstacles. The final path (4.3) consisting of 4 segments was found after 119 iterations and has total length 16.7 m.

$x^s_{o,1}\,(\mathsf{m})$	$x^s_{o,2}\,(\mathsf{m})$	$x^s_{o,3}\,(\mathsf{m})$	$r_o\,(\mathsf{m})$
1.6	4.4	1.8	1
1.0	−3.0	4.0	1
5.0	0.0	3.0	1
−1.4	5.0	2.0	1
−4.0	1.0	3.0	1
−1.0	−2.0	1.5	1
1.0	1.0	3.1	1
3.0	3.0	3.1	1
−3.0	1.0	−1.0	1
1.0	−3.0	−1.0	1

Table 4.1: RRT obstacle parameters – The translational position and size of the obstacles.

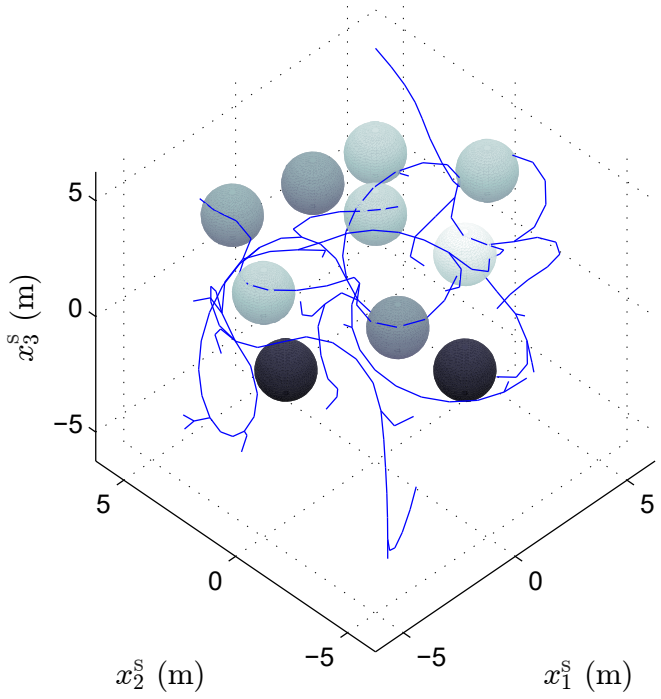


Figure 4.2: AUV RRT complete tree – The entire tree for the motion planning problem of going from the origin to the goal position in the presence of 10 obstacles.

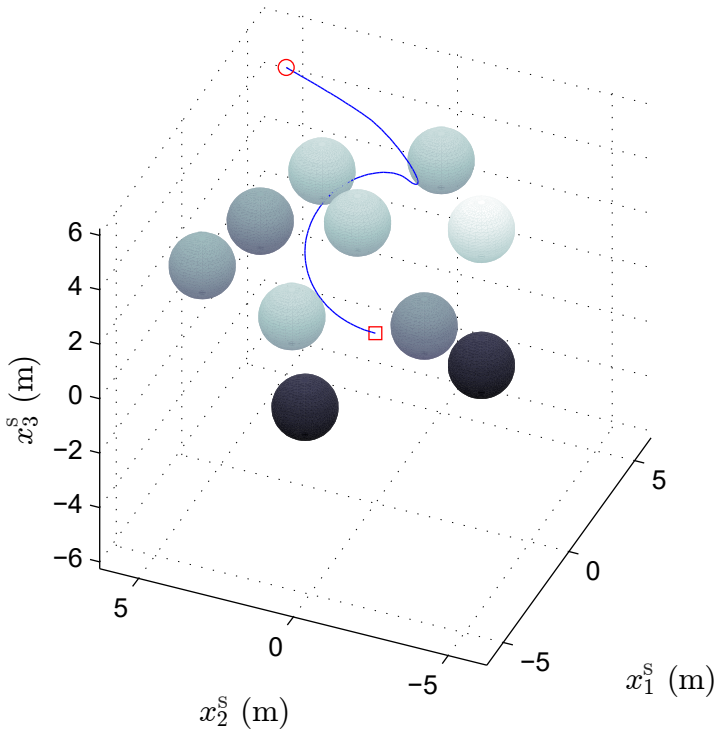


Figure 4.3: AUV RRT feasible path – The path from the origin to the desired position consisting of 4 segments.

The RRT based path planning algorithm is capable of planning a collision-free, feasible path from an initial position to final position and is easy to implement. The main requirement is the availability of an efficient numerical function minimiser. However, like Dubins curves, the curves developed here have translational velocity between segments that is C^1 smooth and the angular velocities are discontinuous between trajectory segments. It may be possible to smooth the transitions between the segments so the trajectory is continuous to a higher degree, however, this is left as further work.

4.3 Chapter summary

By concatenating individual trajectory segments to form a single path from the origin to the desired final position, far greater flexibility is possible compared to the simple obstacle avoidance method previously demonstrated in Chapter 3. The discontinuity of the derivatives at boundaries of the segment are the same as that of Dubins curves. That is to say, the translational states are C^1 smooth and the angular states C^0 smooth. A tracking controller should be able to handle the discontinuity between segments, however, this was not tested and ideally a method to smooth the transition during the planning stage should be designed. The RRT sampling-based path planner was chosen because manually choosing the start and end points of each segment be impractical. However, no obvious method for making the RRT trajectories not only feasible, but also optimal with respect to path was found. This is because the rotational position at the end of the curves cannot be controlled so extensions such as RRT* are not suitable.

Chapter 5

Trajectory generation for drone racing

In Chapter 2 the basic concepts of drone racing were introduced. The trajectory generation method obtained by applying Pontryagin's Maximum Principle in Chapter 3 lacked the flexibility necessary to navigate the types of courses found in drone racing and control the derivatives of the translational position at the boundaries. The equations that defined the translational motion were not decoupled for each axis, any change on one axis would affect the others because the parameters in the optimisation vector were shared between them. The simple obstacle avoidance method from Section 3.6 was suitable for simple obstacles but lacked the control needed to fly through narrow gates. The RRT path planning method described in Section 4.2 is more applicable to vehicles with limited kinematic abilities than quadrotors and the path length was longer than necessary. A trajectory generation method more suitable for drone racing is desired and was the motivation for the polynomial based, virtual domain trajectories developed in this chapter.

Polynomials are used as the basis functions for the trajectory translational states in this chapter because they are simple to implement and constraints on the boundary derivatives can be applied easily. As a function of time, polynomials for the translational positions can be expressed as:

$$x_a(\tau) = \sum_{n=0}^{N_a} p_{a,n} t^n \quad , \quad a \in \{1, 2, 3\} \tag{5.1}$$

where $p_{a,n}$ are the coefficients, N_a is the degree of the polynomial and a is the axis number. Trajectory planning for various systems have been developed using power series polynomials including that for fixed-wing UAVs [36] and spacecraft [38, 39]. For the UAV case, polynomials were parametrised by time to form linear equations by matching the order of the function to the number of conditions. This approach is similar to the one taken in this chapter. However, here waypoints can also be included between the boundary conditions which is possible because the polynomials are parametrised in the virtual domain. Another disadvantage of parametrising polynomial by time as is done in [36] is that there is little flexibility and control over the derivatives during the manoeuvre. For the trajectory to remain feasible throughout, the dynamic capabilities are not fully exploited. If for instance, the maximum acceleration was constrained, the acceleration throughout could be reduced by increasing the trajectory time. However, for much of the manoeuvre the acceleration would be far below the limit and not exploit the capabilities of the vehicle, leading to an inefficient and sub-optimal trajectory with respect to trajectory time. A higher order basis function can be used which allows for greater flexibility but since there is no longer a single solution for a given set of conditions, the coefficients can't be obtained by solving a linear equation. Instead, they must be found by some other method. For example, the coefficients can be chosen to minimise a cost function such as [41] that formulates an unconstrained quadratic program in which the coefficients are found that minimise the fourth derivative snap, which is equivalent to minimising the control effort.

As discussed in Chapter 1, obstacle avoidance in cluttered environments is challenging. Sampling-based path planning is one method that has shown good potential for solving these types of problems in a reasonable amount of time. The application of these approaches in relation to quadrotors is covered in [132]. Most outdoor drone racing does not occur in environments with many obstacles. If this is the case and there is a sparse field of obstacles then they could be ignored until the trajectory entered the collision space. The trajectory could be modified between the two gates where the collision with the offending obstacle occurred and in this way limit the sampling space. However, in this thesis the only obstacles that are considered are gates and a path planning algorithm that is more problem specific is developed to reduce the computation time.

Recalling the discussion on trajectory generation from Chapter 1, the three criteria that must be considered when generating a trajectory are feasibility, obstacle avoidance and optimality. The polynomial based, virtual domain trajectory generation method developed in this chapter considers feasibility and optimality but these are only properly accounted for during the mapping process in the following chapter. To do this the boundary state derivatives must be controllable in the virtual domain. The final trajectory time in the time domain does not need to be considered at this stage. The virtual trajectory also needs to define the heading angle (the direction of the first body axis). Assuming the heading angle is not fixed at a static position, as is the case in much of the literature [88], it can be optimised to improve the trajectory performance.

Original Contributions

The original contributions in this chapter are outlined as follows:

- A polynomial path planning method on the virtual time domain that is suitable for quadrotors and can be mapped to the real time domain is developed. Conditions on the boundary derivatives and multiple, sequential waypoints can be specified for the trajectory to satisfy. The locations of the waypoints on the virtual time domain are chosen by a numerical optimiser that minimises a cost function related to the trajectory length.

- The Waypoint Selection Algorithm (WSA) is developed to find the locations of corrections waypoints that shape the curve sufficiently to ensure it passes through drone racing gates without collisions. Compared to the simple conservative method of ensuring the trajectory passes through hoops by using fixed waypoints, this method generates trajectories that can be flown faster.

- A method for minimising the accumulated angular acceleration of the heading angle when a camera is used to navigate is investigated. The direction of the first body axis is bounded to remain within a defined region of the velocity vector direction. This optimisation is achieved by assigning the heading angle to a B-spline parameterised in the virtual domain. The number of nodes required to find a reasonable solution with the least amount of computation time is investigated through a Monte Carlo simulation.

The chapter is structured as follows. In Section 5.1 a power series polynomial is introduced and a method to find polynomial coefficients by forming a linear equation which contains the conditions of the trajectory is given. A cost function that minimises the length of the path is defined and a simple example is provided to demonstrate the method. Section 5.2 introduces the Waypoint Selection Algorithm that is used to shape the trajectory in order to avoid collisions with the gates. Again, a simple example of the method is provided. The method for optimising the heading angle is given in Section 5.3. Finally, Section 5.4 concludes the chapter and provides a summary of the results.

5.1 Polynomial basis functions

In this section a method of generating polynomial basis functions for each translational axis is developed using the inertial reference frame from Figure 2.1b in Chapter 2. The trajectories are planned on the virtual domain by parametrising the polynomial functions with an abstract parameter $\tau \in [0, 1]$ instead of the actual time $t \in [0, T]$ and they satisfy the given waypoint and boundary conditions. Waypoint conditions in this thesis are defined as translational position states that must be reached, in order, during the trajectory. The location of the waypoints in the virtual domain are found using a numerical optimiser that minimises the geometrical path length. The trajectories found in the virtual domain are not guaranteed or designed to satisfy any feasibility constraints throughout the motion. In order to achieve a feasible trajectory, the virtual domain trajectories must be mapped onto the time domain using the techniques developed in Chapter 6.

5.1.1 Polynomial series

A polynomial for each translational axis describes the position in the inertial frame as a function of τ:

$$P_a(\tau) = p_{a,n}\tau^{N_a} + p_{a,n-1}\tau^{N_a-1} + \cdots + p_{a,0} = \sum_{n=0}^{N_a} p_{a,n}\tau^n \quad , \quad a \in \{1, 2, 3\} \tag{5.2}$$

where $p_{a,n}$ are the coefficients, N_a is the degree of the polynomial and a is the axis number. Any conditions that the polynomial must satisfy are given in the form $P_a^{(r)}(\tau)$ where r is the r^{th} derivative with respect to τ. The initial and final boundary conditions occur at $\tau_i = 0$ and $\tau_f = 1$ respectively. These conditions are used to control the states of the vehicle at the beginning and end of the trajectory. The values of the boundary derivatives $P_a^{(r)}(\tau)$ do not necessarily equal the desired values in the time domain $x_a^{(r)}\big(t(\tau)\big)$ but must be found using the method described in Section 6.1.1. Between the boundary conditions, waypoints in the form of translational position conditions are used to shape the curve for the purposes of obstacle avoidance. The value of τ for each waypoint is found numerically using the procedure described in Section 5.1.3.

5.1.2 Formulating the polynomial matrix

It is convenient to find the coefficients by formulating the polynomial using matrices. The degree of the polynomial that describes the motion on a particular axis is such that a linear equation can be formed when all the conditions are applied:

$$M_a \mathbf{p_a} = \mathbf{c_a} \tag{5.3}$$

where $M_a \in \mathbb{R}^{(N_a+1)\times(N_a+1)}$ is a matrix in which each row is a condition on the path containing the terms of the polynomial at the relevant derivative. The vectors containing the polynomial coefficients and the values of the conditions are $\mathbf{p_a} \in \mathbb{R}^{N_a+1}$ and $\mathbf{c_a} \in \mathbb{R}^{N_a+1}$ respectively. Equation (5.3) can be solved using Gaussian elimination to find the polynomial coefficients. This is more efficient than solving $\mathbf{p_a} = M_a^{-1}\mathbf{c_a}$ because the inverse of M_a does not need to be found. The motion along each axis on the virtual domain is now defined and can be combined to form a multi-dimensional trajectory that satisfies the conditions on the boundaries and passes through the waypoints specified.

5.1.3 Minimising path length

The position of the intermediate waypoints, those between the first and last waypoint, on the virtual domain form the waypoint optimisation vector $\boldsymbol{\Xi}_w = [\tau_{i+1}, \ldots, \tau_{f-1}]^T$. The optimisation vector that minimises the geometrical path

length is obtained using a numerical optimiser. Any given optimisation vector has a unique solution for the motion on each axis, found by solving (5.3). It is desirable that the optimisation vector which results in the shortest path length is found but a sub-optimal optimisation vector will still produce a trajectory that satisfies the conditions. However, a poor choice will cause the trajectory length to be excessively large. This is demonstrated by the planar trajectory example given in Figure 5.1.

The optimisation vector is found as follows. A cost function related to the path length is defined:

$$J(\boldsymbol{\Xi}_{\mathbf{w}}) = \int_0^1 \sum_{a=1}^{A} \dot{P}_a(\tau)^2 \; d\tau \tag{5.4}$$

where A is the number of axes. The optimisation function $J(\boldsymbol{\Xi}_{\mathbf{w}})$ can be calculated numerically or analytically because the integrand is a polynomial. The analytical solution requires less computation and is more accurate. The code developed for this thesis solves (5.4) analytically but since the polynomials are problem specific a single expression cannot be given here. Remembering $\tau_i = 0$ and $\tau_f = 1$, for the waypoints to be passed in the correct order the following must be true:

$$\tau_i < \tau_{i+1} < \tau_2 < \cdots < \tau_{f-1} < \tau_f \tag{5.5}$$

which can be written as a linear constraint:

$$L_w \boldsymbol{\Xi}_{\mathbf{w}} \leq \mathbf{b}_{\mathbf{w}} \tag{5.6}$$

where L_w is a matrix of the associated τ_w terms and $\mathbf{b}_{\mathbf{w}}$ is a column vector that implements the inequality constraints given in (5.5). Finding the optimisation vector that minimises the geometric path length is formulated as:

$$\begin{aligned} \underset{\boldsymbol{\Xi}_{\mathbf{w}}}{\text{minimise}} \quad & J(\boldsymbol{\Xi}_{\mathbf{w}}) \\ \text{subject to} \quad & L_w \boldsymbol{\Xi}_{\mathbf{w}} \leq \mathbf{b}_{\mathbf{w}} \end{aligned} \tag{5.7}$$

The initial guess for the optimisation vector is taken from the previous solution if available. When this is not possible, a uniform distribution of the τ elements for each waypoint is used. A non-uniform distribution based on the Euclidean

distance between waypoints was tested but found to offer no improvement of the solution time compared to the uniform case. Standard optimisation tools such as Matlab's **fmincon** or **patternsearch** can be used to solve this problem. Experimentation showed the latter performing best for the types of trajectories found in this thesis. Gradient based-methods methods such as **fmincon** are prone to becoming stuck in local minima and this was occasionally observed. The minimisation problem (5.7) is continuous but not easily differentiable so providing an analytical expression for the gradient was not possible. In the case of problems like this where the gradient is unavailable, the optimiser must numerically estimate the gradient which can be inaccurate and computationally expensive. Pattern search methods do not require the gradient while optimising and are suitable for functions that are not continuous or differentiable so it is a direct search method. The general method and name were introduced by Hooke and Jeeves [133]. The principle is to vary parameters by steps of a given magnitude until no improvement occurs, the step size is then reduced and the process repeated. The algorithm continues until some termination condition is reached, such as minimum step size.

5.1.4 Polynomial formulation example

To illustrate the method of finding the polynomial trajectories in the virtual domain, an example will be given. A planar trajectory that passes through the waypoints given in Table 5.1 and satisfies the conditions in Table 5.2 is planned. A two-dimensional trajectory is chosen to aid understanding but the process is the same for three-dimensional planning. The waypoints were chosen arbitrarily and are given in Table 5.1.

τ	x_1	x_2
τ_0 (τ_i)	0	5
τ_1 ($\Xi_{w,1}$)	0	4
τ_2 ($\Xi_{w,2}$)	1	3
τ_3 ($\Xi_{w,3}$)	4	6
τ_4 (τ_f)	5	5

Table 5.1: Planar polynomial trajectory example: waypoints – The waypoint parameters for the polynomial formation example.

In this example there are three waypoints between the initial and final point that require τ values (τ_1, τ_2, τ_3) and form the optimisation vector $\mathbf{\Xi_w} = [\Xi_{w,1},\ \Xi_{w,2},\ \Xi_{w,3}]$. The boundary conditions required are given in Table 5.2. The vector of the con-

Condition	Value
$P_1^{(1)}(\tau_0)$	-5
$P_1^{(2)}(\tau_0)$	0
$P_1^{(3)}(\tau_0)$	0
$P_1^{(4)}(\tau_0)$	0
$P_1^{(1)}(\tau_4)$	0
$P_2^{(1)}(\tau_0)$	4
$P_2^{(2)}(\tau_0)$	0
$P_2^{(3)}(\tau_0)$	0
$P_2^{(1)}(\tau_4)$	-5

Table 5.2: Planar polynomial trajectory example: conditions – The conditions in the virtual time domain that the trajectory must satisfy.

ditions on the first axis is therefore:

$$\begin{aligned}\mathbf{c_1} = [&P_1^{(0)}(\tau_0),\ P_1^{(0)}(\tau_1),\ P_1^{(0)}(\tau_2),\ P_1^{(0)}(\tau_3),\ P_1^{(0)}(\tau_4),\\ &P_1^{(1)}(\tau_0),\ P_1^{(2)}(\tau_0),\ P_1^{(3)}(\tau_0),\ P_1^{(4)}(\tau_0),\ P_1^{(1)}(\tau_4)\ ]^T\end{aligned} \tag{5.8}$$

and on the second axis:

$$\begin{aligned}\mathbf{c_2} = [&P_2^{(0)}(\tau_0),\ P_2^{(0)}(\tau_1),\ P_2^{(0)}(\tau_2),\ P_2^{(0)}(\tau_3),\ P_2^{(0)}(\tau_4),\\ &P_2^{(1)}(\tau_0),\ P_2^{(2)}(\tau_0),\ P_2^{(3)}(\tau_0),\ P_2^{(1)}(\tau_4)\ ]^T\end{aligned} \tag{5.9}$$

To ensure the waypoints were followed in this correct order the linear constraint (5.6) was applied. The linear inequality constraints matrix in this example is written as:

$$L_w = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

and the constraints vector is:

$$\mathbf{b_w} = [0,\ 0,\ 0,\ 0,\ 1]^T \tag{5.11}$$

After minimising the cost function (5.4) to find the minimum path length, the optimisation vector for this problem is $\mathbf{\Xi_w} = [0,\ 0.38,\ 0.51,\ 0.86,\ 1]^T$. The matrix containing the polynomial conditions of the first axis is

$$M_1 = \begin{bmatrix} {\tau_0}^0 & {\tau_0}^1 & {\tau_0}^2 & {\tau_0}^3 & {\tau_0}^4 & {\tau_0}^5 & {\tau_0}^6 & {\tau_0}^7 & {\tau_0}^8 & {\tau_0}^9 \\ {\tau_1}^0 & {\tau_1}^1 & {\tau_1}^2 & {\tau_1}^3 & {\tau_1}^4 & {\tau_1}^5 & {\tau_1}^6 & {\tau_1}^7 & {\tau_1}^8 & {\tau_1}^9 \\ {\tau_2}^0 & {\tau_2}^1 & {\tau_2}^2 & {\tau_2}^3 & {\tau_2}^4 & {\tau_2}^5 & {\tau_2}^6 & {\tau_2}^7 & {\tau_2}^8 & {\tau_2}^9 \\ {\tau_3}^0 & {\tau_3}^1 & {\tau_3}^2 & {\tau_3}^3 & {\tau_3}^4 & {\tau_3}^5 & {\tau_3}^6 & {\tau_3}^7 & {\tau_3}^8 & {\tau_3}^9 \\ {\tau_4}^0 & {\tau_4}^1 & {\tau_4}^2 & {\tau_4}^3 & {\tau_4}^4 & {\tau_4}^5 & {\tau_4}^6 & {\tau_4}^7 & {\tau_4}^8 & {\tau_4}^9 \\ 0 & {\tau_0}^0 & 2{\tau_0}^1 & 3{\tau_0}^2 & 4{\tau_0}^3 & 5{\tau_0}^4 & 6{\tau_0}^5 & 7{\tau_0}^6 & 8{\tau_0}^7 & 9{\tau_0}^8 \\ 0 & 0 & 2{\tau_0}^0 & 6{\tau_0}^1 & 12{\tau_0}^2 & 20{\tau_0}^3 & 30{\tau_0}^4 & 42{\tau_0}^5 & 56{\tau_0}^6 & 72{\tau_0}^7 \\ 0 & 0 & 0 & 6{\tau_0}^0 & 24{\tau_0}^1 & 60{\tau_0}^2 & 120{\tau_0}^3 & 210{\tau_0}^4 & 336{\tau_0}^5 & 504{\tau_0}^6 \\ 0 & 0 & 0 & 0 & 24{\tau_0}^0 & 120{\tau_0}^1 & 360{\tau_0}^2 & 840{\tau_0}^3 & 1680{\tau_0}^4 & 3024{\tau_0}^5 \\ 0 & {\tau_4}^0 & 2{\tau_4}^1 & 3{\tau_4}^2 & 4{\tau_4}^3 & 5{\tau_4}^4 & 6{\tau_4}^5 & 7{\tau_4}^6 & 8{\tau_4}^7 & 9{\tau_4}^8 \end{bmatrix} \tag{5.12}$$

and for the second axis:

$$M_2 = \begin{bmatrix} {\tau_0}^0 & {\tau_0}^1 & {\tau_0}^2 & {\tau_0}^3 & {\tau_0}^4 & {\tau_0}^5 & {\tau_0}^6 & {\tau_0}^7 & {\tau_0}^8 \\ {\tau_1}^0 & {\tau_1}^1 & {\tau_1}^2 & {\tau_1}^3 & {\tau_1}^4 & {\tau_1}^5 & {\tau_1}^6 & {\tau_1}^7 & {\tau_1}^8 \\ {\tau_2}^0 & {\tau_2}^1 & {\tau_2}^2 & {\tau_2}^3 & {\tau_2}^4 & {\tau_2}^5 & {\tau_2}^6 & {\tau_2}^7 & {\tau_2}^8 \\ {\tau_3}^0 & {\tau_3}^1 & {\tau_3}^2 & {\tau_3}^3 & {\tau_3}^4 & {\tau_3}^5 & {\tau_3}^6 & {\tau_3}^7 & {\tau_3}^8 \\ {\tau_4}^0 & {\tau_4}^1 & {\tau_4}^2 & {\tau_4}^3 & {\tau_4}^4 & {\tau_4}^5 & {\tau_4}^6 & {\tau_4}^7 & {\tau_4}^8 \\ 0 & {\tau_0}^0 & 2{\tau_0}^1 & 3{\tau_0}^2 & 4{\tau_0}^3 & 5{\tau_0}^4 & 6{\tau_0}^5 & 7{\tau_0}^6 & 8{\tau_0}^7 \\ 0 & 0 & 2{\tau_0}^0 & 6{\tau_0}^1 & 12{\tau_0}^2 & 20{\tau_0}^3 & 30{\tau_0}^4 & 42{\tau_0}^5 & 56{\tau_0}^6 \\ 0 & 0 & 0 & 6{\tau_0}^0 & 24{\tau_0}^1 & 60{\tau_0}^2 & 120{\tau_0}^3 & 210{\tau_0}^4 & 336{\tau_0}^5 \\ 0 & {\tau_4}^0 & 2{\tau_4}^1 & 3{\tau_4}^2 & 4{\tau_4}^3 & 5{\tau_4}^4 & 6{\tau_4}^5 & 7{\tau_4}^6 & 8{\tau_4}^7 \end{bmatrix} \tag{5.13}$$

After solving the linear equation (5.3), the polynomial coefficients for the first axis were:

$$\mathbf{p_1} = [0,\ -5,\ 0,\ 0,\ 0,\ 1.7066e3,\ -7.2016e3,\ 1.1699e4,\ -8.5341e3,\ 2.304e3]^T \tag{5.14}$$

and for the second axis:

$$\mathbf{p_2} = [5,\ 4,\ 0,\ 0,\ -473.67,\ 1.2834e3,\ -911.59,\ -155.3180,\ 253.19]^T \tag{5.15}$$

The solid, black line in Figure 5.1 plots the trajectory of the optimisation vector that minimised the path length. To illustrate non-optimal solutions, three other trajectories represented by dashed lines are shown in the same figure. These

satisfy the waypoint and boundary conditions (Table 5.1 & Table 5.2) but have longer path lengths. Figure 5.2 shows the translational position and its derivatives against the abstract parameter τ. All the boundary conditions are satisfied.

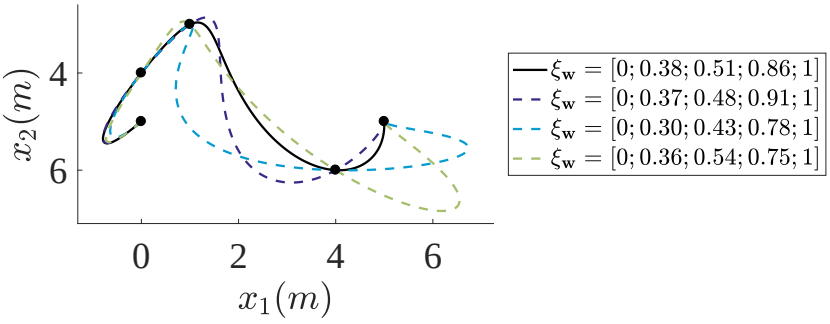
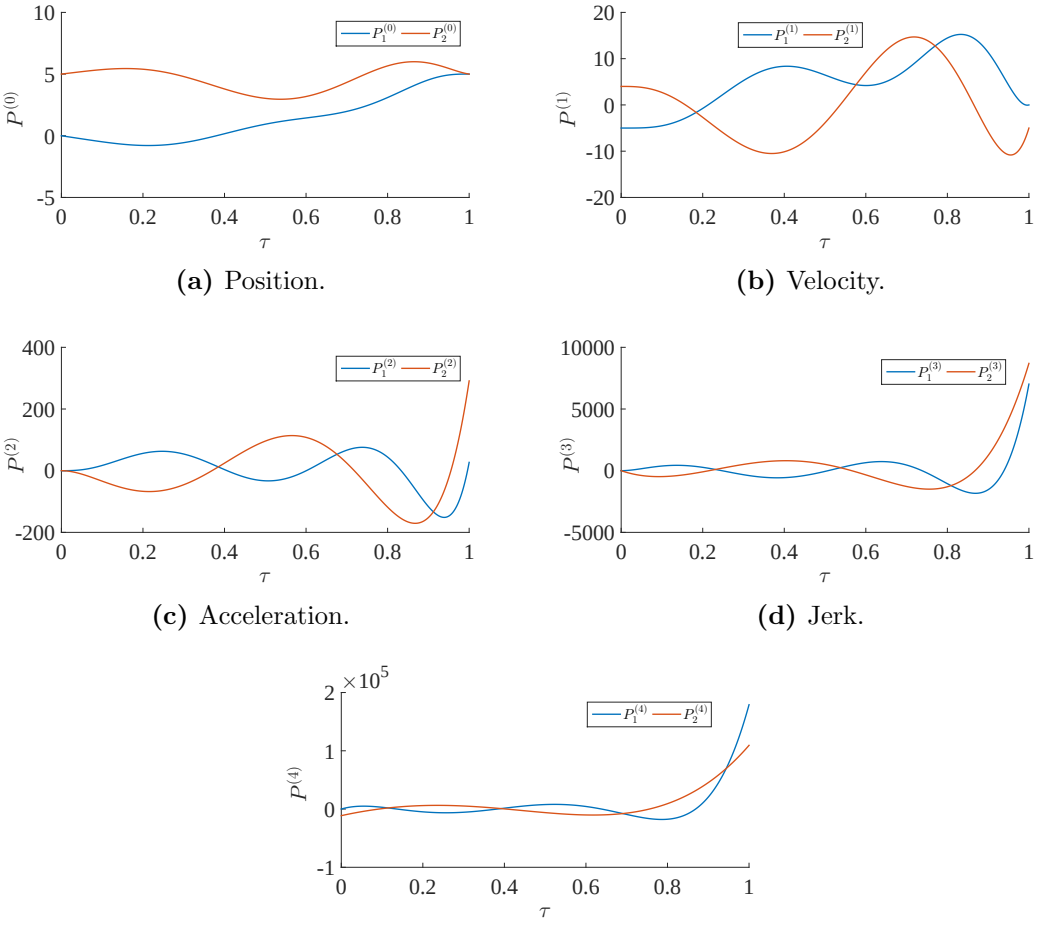


Figure 5.1: Polynomial example: trajectories – The three dashed curves represent sub-optimal trajectories with path lengths longer than the optimal trajectory as found by the numerical optimiser, represented by the solid black line.

5.2 Navigating through gates

A feasible trajectory must pass through each of the gates, modelled in Section 2.2.2, without entering the collision region. Simply placing a single waypoint in the centre of a gate will not guarantee the trajectory safely passes through the gate because there is no control over the direction during entry and exit. To avoid collisions, additional waypoints can be added, as required, before and after the centre of the gate. Waypoints added before the centre waypoint of a gate are known as entry waypoints and the ones after centre waypoint are exit waypoints. The Waypoint Selection Algorithm developed in this section finds the locations of these waypoints in the inertial frame to pass through the gates without entering the collision region. It also minimises the change in the shape of the curve so the path length is not increased unnecessarily. Early tests of the algorithm simply put two waypoints on the $\mathbf{b_{h,1}}$ axis at either side of the gate's centre. However, it was observed that this caused tight turns which increased the trajectory time because the speed needed to be reduced for the rotor thrusts to stay within the feasibility bounds. By using an iterative process that gradually changes the shape of the curve, the increase in trajectory time can be minimised.



(a) Position. **(b)** Velocity. **(c)** Acceleration. **(d)** Jerk. **(e)** Snap.

Figure 5.2: Polynomial formation example: states – The translational states from position to snap in the virtual domain.

5.2.1 Waypoint Selection Algorithm

For each gate, the Waypoint Selection Algorithm (WSA) can add entry and exit waypoints when required. In this section the algorithm is described in the context of adding an entry waypoint but the process is similar for the exit waypoint case. The translational position of the gate in the inertial frame $\mathbf{x_h}$ is automatically added to the list of waypoints a trajectory must pass through when it is defined. Assuming the trajectory collides with the gate upon entering the following method is used to find the position of the entry waypoint $\mathbf{x_{W,a}} \in \mathbb{R}^3$. A unit vector $\mathbf{v_{W,\dot{P}}} \in \mathbb{R}^3$ pointing in the same direction as the inertial velocity vector $\dot{\mathbf{P}}$ in the virtual domain at $\mathbf{x_h}$ is defined. The unit vector that defines the direction of the first body axis of the gate $\mathbf{b_{h,1}}$ is also shown. By taking the cross product these unit vectors a third vector is obtained: $\mathbf{b_{W,3}} = \mathbf{b_{h,1}} \times \mathbf{v_{W,\dot{P}}}$. This vector is used as an axis of rotation about which the vector $\mathbf{v_{W,a}}$ is rotated by an angle σ from $\mathbf{v_{W,\dot{P}}}$. The Euclidean distance between $\mathbf{x_{W,a}}$ and $\mathbf{x_h}$ can be altered by varying the magnitude $\|\mathbf{v_{W,a}}\|$. The maximum rotation angle σ_{max} can be calculated using the dot product:

$$\sigma_{max} = cos^{-1}\left(\frac{\mathbf{v_{W,\dot{P}}} \cdot \mathbf{b_{h,1}}}{\left\|\mathbf{v_{W,\dot{P}}}\right\| \times \|\mathbf{b_{h,1}}\|}\right) \tag{5.16}$$

If there is a collision when entering the gate, σ is gradually increased to σ_{max} until there is no collision. If σ_{max} is reached and a collision still occurs σ is reset to zero and the magnitude $\|\mathbf{v_{W,a}}\|$ is increased and the process begins again. A limit on the maximum magnitude ensures the algorithm terminates with an error message if there is a problem and the geometry needs to be checked. This algorithm only considers gate collisions for the segment of trajectory between the gate before and the gate after the one in question. If a collision occurs with a gate outside of these conditions another technique like a sampling based method would need to be used. In practice this rarely occurs with racing circuits because having the course pass through a previous section of the course significantly increases the likelihood of a collision with other drones.

As discussed in the next chapter, when mapping from the virtual domain to the time domain the geometrical shape of the trajectory can change so it is necessary to ensure that when this occurs the new trajectory is free from collision. The block diagram in Figure 5.3 describes the process. In practice it was found

that the WSA algorithm needed to be run after changing the mapping routine but it is important to check the trajectory is still collision free.

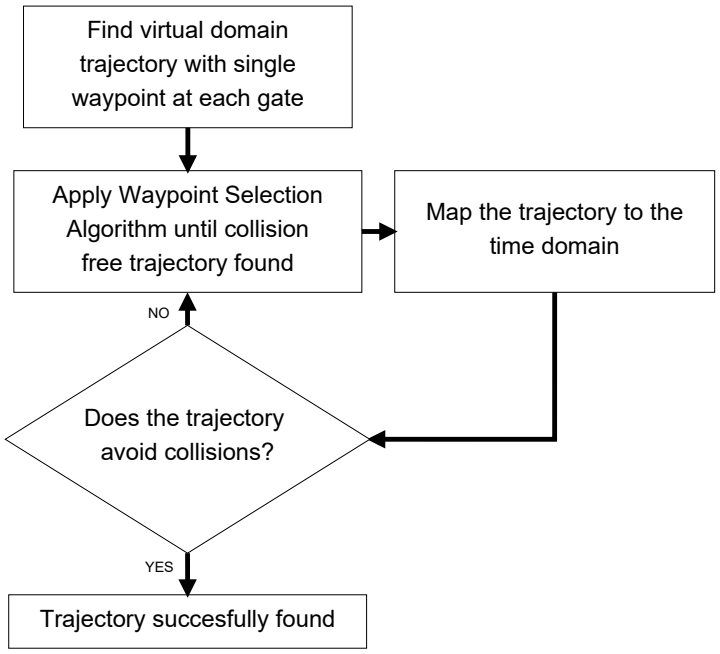


Figure 5.3: Waypoint Selection Algorithm: block diagram – A block diagram of the process used to determine if a waypoint should be added (applicable to both the entry and exit case).

5.2.2 Single gate example

To demonstrate the Waypoint Selection Algorithm an example is given in Figure 5.4. Starting from rest at the origin $\mathbf{x}(0) = [0\,\mathrm{m}, 0\,\mathrm{m}, 0\,\mathrm{m}]^T$, the trajectory must pass through a single gate and finish at $\mathbf{x}(T) = [8\,\mathrm{m}, 0\,\mathrm{m}, 0\,\mathrm{m}]^T$. The hoop parameters are as follows: $R_h = [0, -1, 0; 0.866, 0, 0.5, -0.5, 0, 0.8666]$, $\mathbf{x_h} = [5\,\mathrm{m}, 0\,\mathrm{m}, 0\,\mathrm{m}]^T$, $r_{h,i} = 1.2\,\mathrm{m}$, $r_{h,o} = 2\,\mathrm{m}$ and $l_h = 2\,\mathrm{m}$. Four iterations were required to find the correction waypoints that allowed a collision free trajectory to be obtained. The first iteration in Figure 5.4a used a single waypoint at the centre of the gate and collisions occurred upon entering and exiting the cylinders that represented the collision region. A correction waypoint was added on the second iteration in Figure 5.4b but collisions still occurred at both sides. The location of this waypoint was changed on the third iteration shown in Figure 5.4c and there was no longer a collision when entering the gate. The fourth iteration in Figure 5.4d added a waypoint that corrected for the collision upon exiting to

obtain the collision free trajectory. This example demonstrates that a 'worst case scenario' in which the trajectory collides with the gate perpendicularly to the correct entrance direction can be quickly resolved with the Waypoint Selection Algorithm.

5.2.3 Multiple gate example

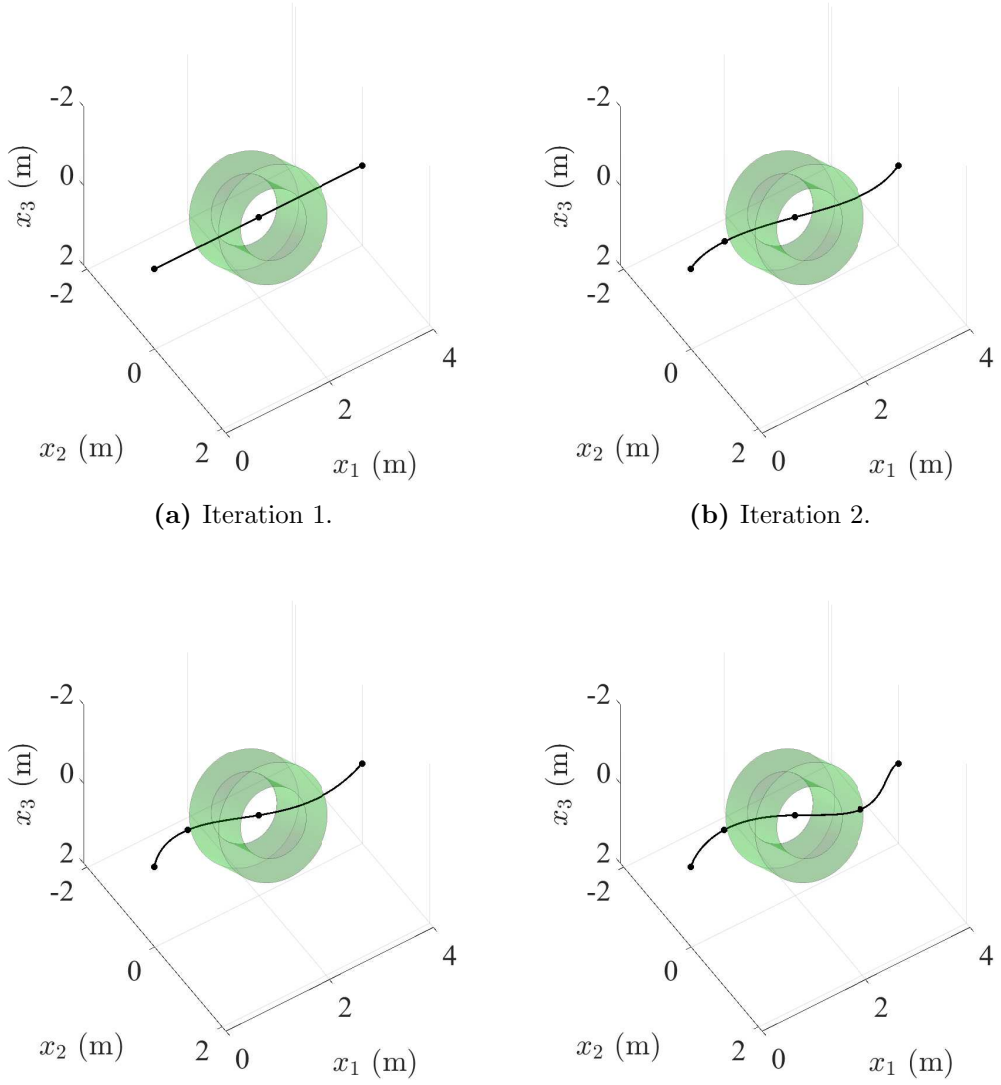
In this example a trajectory that must pass through 3 gates rotated about multiple axes is generated. As in the previous example, the trajectory starts from rest but the derivatives at the final waypoint are not fixed. The inner radius, outer radius and length for all the gates are $r_{h,i} = 0.3\,\text{m}$, $r_{h,o} = 0.9\,\text{m}$ and $l_h = 0.3\,\text{m}$ respectively. The position and orientation of each gate is given in Table 5.3. The initial and final translation position are chosen as $\mathbf{x}(0) = [0\,\text{m}, 0\,\text{m}, 0\,\text{m}]^T$ and $\mathbf{x}(T) = [2\,\text{m}, -4\,\text{m}, 0\,\text{m}]^T$ respectively. Figure 5.5a shows the trajectory with no additional waypoints and collisions occur at every gate. At the start of this section it was noted that one method of using waypoints to navigate through the gates is to place them along the axis defined by $\mathbf{b_{h,1}}$ and an example of this is given in Figure 5.5c as a 'simple approach'. Compared to the trajectory found by the WSA in Figure 5.5b it can be seen that this method produces inferior results. Although the path length is slightly shorter, the turns are tighter which forces the vehicle to reduce its speed in order to stay within the thrust limits and increase the trajectory time.

$x_{h,1}\,(\mathsf{m})$	$x_{h,2}\,(\mathsf{m})$	$x_{h,3}\,(\mathsf{m})$	$\phi_{h,1}\,(^\circ)$	$\phi_{h,2}\,(^\circ)$	$\phi_{h,3}\,(^\circ)$
2	0	0	0	10	−50
6	2	0	0	−10	20
6	−2	0	0	−10	−160

Table 5.3: Multiple gate example: gate parameters – The gate parameters for the multiple gate example.

5.3 Heading angle optimisation

The heading angle θ is the angle between the first body axis of the vehicle and $\mathbf{e_1}$ of the inertial reference frame about the third axis $\mathbf{e_3}$ of the inertial frame. For



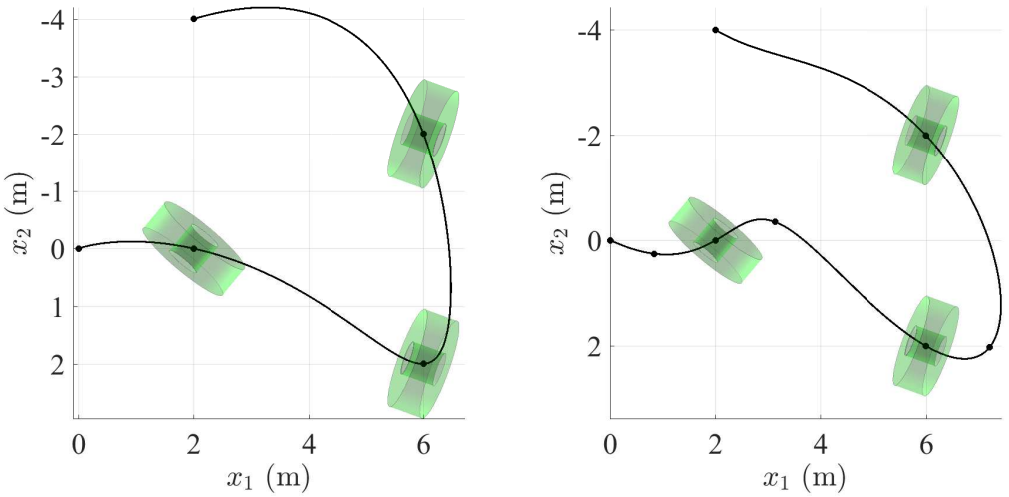
(a) Iteration 1.

(b) Iteration 2.

(c) Iteration 3.

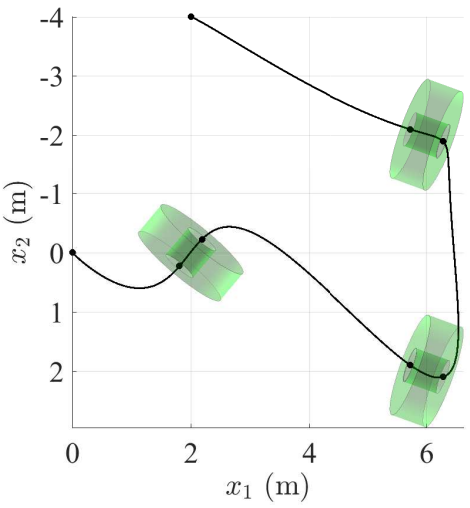
(d) Iteration 4.

Figure 5.4: Waypoint Selection Algorithm: single gate example – The four iterations required to find a collision free trajectory are shown.



(a) Centre waypoints only.

(b) Waypoint Selection Algorithm.



(c) Simple approach.

Figure 5.5: Multiple gate example: trajectories – Both the WSA and the simple approach produce collision free trajectories but the latter has tight turns which result in a slower trajectory time.

many multi-rotor applications the direction of the first body axis can be fixed throughout the trajectory. For instance, if a sensor such as an infra-red camera for crop monitoring is mounted underneath the vehicle pointing in the direction of the third body axis there is little need to have the heading angle vary. However, if a camera is pointing in the direction of the first body axis, the heading angle can be changed so it is pointing towards the area of interest. For drone racing the camera is used to detect obstacles and should point in the velocity vector direction of the planar motion on the $\mathbf{e_1}-\mathbf{e_2}$ plane. Since wide-angle lenses are typically used there is a region of tolerance between the heading angle and the planar motion. Consequently, it is not required that the heading angle is exactly equal to the planar velocity vector direction and can be optimised to reduce the amount of angular acceleration about the third body axis. This is desirable because additional thrust must be used to make these changes (see Chapter 8), thereby reducing the translational capabilities of the vehicle and increasing the trajectory time.

5.3.1 Determining the planar velocity vector direction

The planar velocity vector direction as an angle about $\mathbf{e_3}$ can be found using the arctangent function:

$$\theta_v(\tau) = \text{atan2}\left(v_2(\tau),\, v_1(\tau)\right) \tag{5.17}$$

where θ_v is velocity vector angle in radians. This is equivalent to side slip heading in conventional flight mechanics. When $v_1 = v_2 = 0$ the previous non-zero θ_v is used if available, otherwise the next non-zero value of θ_v is used. This is necessary because if the trajectory at a given time has zero velocity then (5.17) is undefined. This commonly occurs in the context of drone racing when the vehicle starts at rest or zero velocity is desired at the end. The arctangent function **atan2** has a range $\theta \in [-\pi, \pi]$, so in order to avoid discontinuities $\theta_v(\tau)$ must be unwrapped by adding multiples of $\pm 2\pi$ to jumps between consecutive elements equal or bigger than π. Figure 5.6 is an example of the function **unwrap** being used on a rest-to-rest rotation with two revolutions. At $\tau = 0.38$ and $\tau = 0.62$ the angle calculated using **atan2** jumps from $+\pi$ to $-\pi$ but $\theta_{unwrapped}$ remains smooth. By unwrapping the velocity vector angle there are no discontinuities at

these points that would cause problems with the optimisation when the tolerance region is defined using the method in the next section.

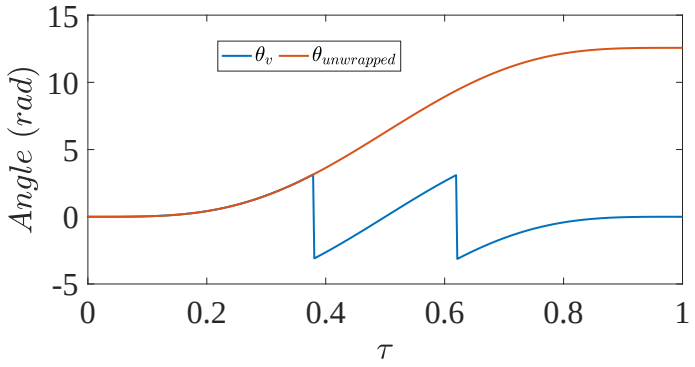


Figure 5.6: Angle unwrapping example – A rest-to-rest rotation with two revolutions. The unwrapped angle $\theta_{unwrapped}$ has no discontinuites, unlike θ_v with discontinuities at $+\pi$ and $-\pi$ radians.

5.3.2 Optimised heading angle

If it is assumed that it is acceptable for the heading angle to track but not exactly equal the planar velocity vector angle, then it can be optimised to minimise the accumulated angular acceleration cost, defined as:

$$J_{\ddot{\theta}_{opt}} = \int_0^1 \ddot{\theta}_{opt}(\tau)^2 \ d\tau \tag{5.18}$$

where $\ddot{\theta}_{opt}$ is the angular acceleration and is the second derivative of the optimised heading angle θ_{opt} with respect to τ. This cost function was chosen because it minimises the accumulated amount of rate of change in the heading angle. Accelerating about the yaw axis requires thrust and can limit the translational acceleration capabilities of the vehicle at times. An analytical expression could be used to calculate $J_{\ddot{\theta}_{opt}}$ but it would need to be re-derived for each unique B-spline so instead (5.18) was discretised and the integral evaluated numerically. The accuracy of this calculation is dependent upon the coarseness of the discretisation so a suitable resolution should be chosen.

The allowable deviation of the optimised heading angle from the planar velo-

city angle is constrained throughout the trajectory such that:

$$\theta_{min}(\tau) \leq \theta_{opt}(\tau) \leq \theta_{max}(\tau) \tag{5.19}$$

where $\theta_{min}(\tau)$ and $\theta_{max}(\tau)$ are the minimum and maximum angles at a given time on the virtual domain respectively. They are calculated using a tolerance angle θ_{tol} that is chosen by the user:

$$\begin{aligned} \theta_{min}(\tau) &= \theta_{unwrapped}(\tau) - \theta_{tol} \\ \theta_{max}(\tau) &= \theta_{unwrapped}(\tau) + \theta_{tol} \end{aligned} \tag{5.20}$$

An example optimisation of the heading angle is given in Figure 5.7. The process used to generate the optimised heading angle is described in the following section.

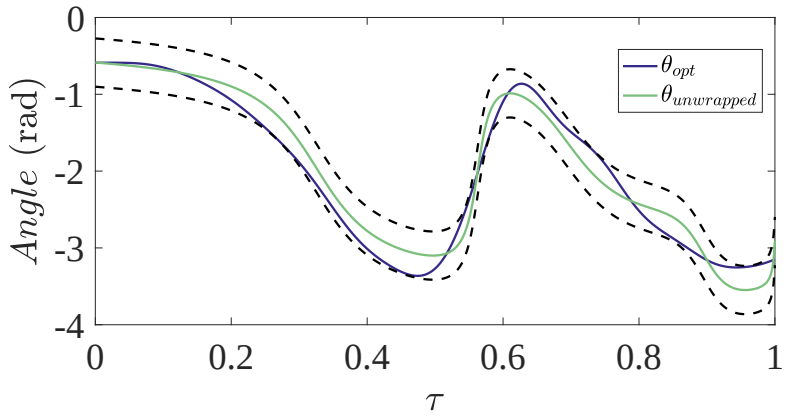


Figure 5.7: Heading angle optimisation example – The heading angle before and after is has been optimised. The dashed lines represent the boundaries in which the optimised angle must remain.

The optimised heading angle $\theta_{opt}(\tau)$ was parametrised using a B-spline. The properties and formulation of B-splines were introduced in Section 2.3 but to summarise for this application, the main benefits are the local modification scheme and the fine degree of control achievable when required. The coefficients of the B-spline that describe $\theta_{opt}(\tau)$ are found using a numerical minimiser but first the location of the knots must be chosen. The smoothness of the B-spline curve up to a given derivative is dependent upon the order. To achieve a continuously smooth function up to the r^{th} derivative, the order is $n = r + 3$. To clamp the boundary values of $\theta_{opt}(\tau)$ there must be n repeated values of either 0 or 1 for the initial and final knots respectively. The boundary coefficients must also be repeated and

this is discussed below. The knot vector is composed of three vectors:

$$\kappa = [\kappa_{\mathbf{initial}},\ \kappa_{\mathbf{interior}},\ \kappa_{\mathbf{final}}] \tag{5.21}$$

where $\kappa_{\mathbf{initial}} \in \mathbb{R}^n$, $\kappa_{\mathbf{interior}} \in \mathbb{R}^k$ and $\kappa_{\mathbf{final}} \in \mathbb{R}^n$ are vectors composed of the initial, interior and final knots. The number of interior knots k can be freely chosen by the user. Initially, the interior knots are evenly distributed in ascending order between the boundary knots but as mentioned below it is sometimes necessary to include additional knots when forming an initial guess if the constraint (5.19) is being violated. Increasing the number of interior knots will give more control over the spline but a numerical minimiser will struggle to converge in a reasonable time when too many parameters must be optimised. This is explored further for the context of this problem in Section 5.3.4. The total number of knots or length of the knot vector κ is N_κ.

The length of the coefficient vector is calculated by $N_\chi = N_{knots} - n$. Before applying a numerical optimiser to find the heading angle it is beneficial to generate an initial guess for the coefficient vector χ. A good initial guess is one which is close to $\theta_{unwrapped}(\tau)$ and always remains within the minimum and maximum tolerances throughout. This is important because the cost function only considers the largest breach of the limits. So if there are many violations only the largest has a bearing on the optimisation cost which causes the convergence time to be excessive because between iterations there is no penalty applied to the smaller violations.

An efficient method to calculate a guess for the coefficients is by using the fixed knot variation of the least-squares spline approximation [115] that is available in Matlab as the **spap2** function. This method requires the virtual domain parameter τ and velocity vector heading $\theta_{unwrapped}(\tau)$ to be discretised and provided to the algorithm in conjunction with the knots vector κ and order of the B-spline n. The function returns a unique coefficient vector χ that corresponds to a B-spline approximation of the heading angle $\theta_{leastSqares}(\tau)$. In the event of $\theta_{leastSqares}(\tau)$ not remaining within the tolerance values a new knot location is added to $\kappa_{\mathbf{interior}}$ at the position where the maximum violation occurred and the least-squares spline approximation is performed again. This process continues until a feasible heading angle represented by a B-spline can be found. It is possible in extreme cases for this approach to fail and no feasible heading angle is

found. This typically occurs when there are discontinuities in the velocity vector heading angle, for instance when the vehicle comes to a complete rest during the trajectory before continuing in a different direction. This is a dynamically valid trajectory but the method described here fails to represent it. The trajectory can either be rejected as unsuitable or, more practically, it can be accepted that for a short time it is impossible for the heading angle to track the velocity vector direction exactly. This may or may not be an issue and is left to the operators judgement.

The least-square approximation does not consider any conditions placed upon the heading angle and its derivatives at the boundaries. In order to fix the heading angle and drive its derivatives up to the r^{th} degree to zero, repeated coefficients can be used. For the initial boundaries to be set, the first $r+1$ coefficients must be equal to the the desired initial heading angle. Likewise, for the final boundary conditions to be set, the last $r+1$ coefficients must equal the final desired heading angle. The coefficients that are free to be chosen by the optimiser form the optimisation vector $\mathbf{\Xi}_\chi$. It should be noted that if either of these conditions on the boundaries is not required, for instance if the final angle is not fixed, then the associated coefficients of that boundary condition are included as free parameters in the optimisation. Being able to choose the precise angle at the boundary and specify that the derivatives are to be zero is useful when concatenating separate trajectory segments because then it is possible to ensure the controls are smooth between them.

If the boundary coefficients are changed then the B-spline is recalculated with the new knot and coefficient vector to ensure it remains within the tolerance bounds. If this is not so, a new knot is added at the location of the maximum violation and the least-squares spline approximation method is performed again. Since only the coefficients at the boundaries have been changed from the previous step, it will only be at the beginning and end of the spline that problems can occur. This is due to the local modification scheme that is characteristic of the B-spline function. In practice it is seldom necessary to rerun steps because the heading angle limits have been breached.

Assuming a valid initial guess has been obtained, the optimisation of the heading angle can be performed using the following multi-parameter minimisation:

$$\underset{\Xi_\chi}{\text{minimize}} \quad j_\theta J_\theta(\Xi_\chi) + j_{\ddot{\theta}} J_{\ddot{\theta}_{opt}}(\Xi_\chi) \tag{5.22}$$

where j_θ and $j_{\ddot{\theta}}$ are scalar weightings for the boundary violation and the accumulated acceleration respectively. The accumulated acceleration cost $J_{\ddot{\theta}}(\Xi_\chi)$ was defined previously and the boundary violation cost $J_\theta(\Xi_\chi)$ is found as follows. The logical statements that determines the undershoot cost is:

$$J_{\theta,undershoot} = \begin{cases} \max(\theta_{min}(\tau) - \theta_{opt}(\tau)), & \text{if } \max(\theta_{min}(\tau) - \theta_{opt}(\tau)) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

and similarly for overshoot:

$$J_{\theta,overshoot} = \begin{cases} \max(\theta_{opt}(\tau) - \theta_{max}(\tau)), & \text{if } \max(\theta_{opt}(\tau) - \theta_{max}(\tau)) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

The largest of either the undershoot and overshoot is used as the boundary cost violation, or if identical they are identical (including zero) then the undershoot is used:

$$J_\theta = \begin{cases} J_{\theta,undershoot}, & \text{if } J_{\theta,undershoot} \geq J_{\theta,overshoot} \\ J_{\theta,overshoot}, & \text{otherwise} \end{cases}$$

The maximum values are obtained using the **max** function included in Matlab. Theoretically, the coarseness of the discretisation will affect the computation time of this function but in practice it is negligible compared to other functions so any reasonable resolution can be used.

In the multi-objective cost function (5.22) there is no obvious metric to compare the two types of cost. Therefore the size of the weightings j_θ and $j_{\ddot{\theta}}$ must be chosen with a heuristic approach. Since it is desirable for there to be no violation of the heading angle limits, j_θ should be large to reflect the importance of this requirement. However, since numerical minimisers such as **fmincon** and **patternsearch** can't include the heading angle limits as hard constraints, very small violations may occur. Including a safety margin on the θ_{tol} is one way of negating this and it is trivial to check a solution to ensure it is valid.

5.3.3 Optimised heading example

An example will be given to illustrate the process of finding the optimal heading angle that minimises angular acceleration. A planar trajectory (Figure 5.8) was

randomly generated using the method described in Section 5.3.4. There are a mixture of turns with different radii of curvature, the smallest being at the end of the trajectory. The unwrapped heading angle that tracks the velocity vector direction is plotted in Figure 5.9. The dashed lines represent the maximum and minimum angles when $\theta_{tol} = 0.2618\,\text{rad}$.

Before calculating the coefficients using the least-squares spline approximation, the knot vector and order must be chosen. The number of unique knot vector nodes was chosen to be 18. The knot vector nodes are control points on the B-spline that influence its shape. The order was chosen as $n = 7$ because it desired that up to the fourth derivative of the angle should be smooth and continuous. The initial derivatives of the heading angle should be zero up to the same derivative. The only constraint on the final conditions is that the heading angle should be within the tolerance bounds. The length of the knot vector (including the repeated boundary knots) is $N_\kappa = 30$:

$$\kappa = [0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0.06,\, 0.12,\, 0.18,\, 0.24,\, 0.30,\, 0.35, 0.41,\, 0.47,\, 0.53,\, 0.59,\\ 0.65,\, 0.71,\, 0.76,\, 0.82,\, 0.88,\, 0.94,\, 1,\, 1,\, 1,\, 1,\, 1,\, 1,\, 1] \tag{5.23}$$

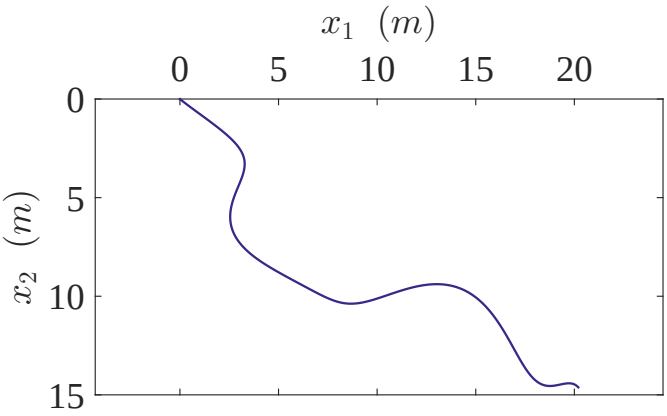


Figure 5.8: Heading angle optimisation example: trajectory – A planar trajectory starting at the origin that was randomly generated using the process described in Section 5.3.4. It contains turns of different radii and is parametrised in the virtual domain.

There are 23 coefficients and the initial guess for these is found using the

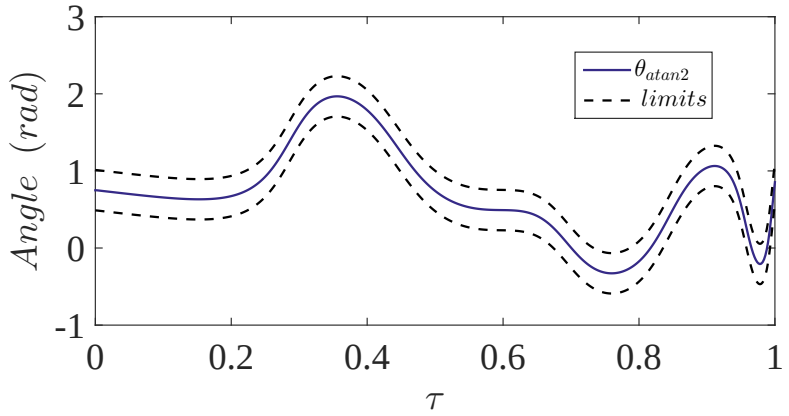


Figure 5.9: Heading angle optimisation example: velocity vector angle – The heading angle of the trajectory in Figure 5.8, calculated using the **atan2** function. The dashed lines mark the maximum and minimum allowable angles during the trajectory.

least-squares approximation:

$$\chi = [0.75,\ 0.75,\ 0.70,\ 0.72,\ 0.61,\ 0.65,\ 0.61,\ 0.77,\ 2.29,\ 2.02,\ 1.42,\\ 0.53,\ 0.46,\ 0.56,\ 0.44,\ -0.52,\ -0.54,\ 0.67\,1.15,\ 1.66,\ -0.67,\ -0.33,\ 0.92] \tag{5.24}$$

A plot of the heading angle as found by the least-squares approximation is shown in Figure 5.10. It can be seen that the approximation closely matches the velocity vector heading angle and $\theta_{leastSquares}$ remains within the tolerance limits.

After replacing the coefficients to force zero derivative initial boundary conditions and clamp the starting angle, the coefficient vector is:

$$\chi = [0.75,\ 0.75,\ 0.75,\ 0.75,\ 0.75,\ 0.65,\ 0.61,\ 0.77,\ 2.29,\ 2.02,\ 1.42,\\ 0.53,\ 0.46,\ 0.56,\ 0.44,\ -0.52,\ -0.54,\ 0.67,\ 1.15,\ 1.66,\ -0.67,\ -0.33,\ 0.92] \tag{5.25}$$

These coefficients and knot vector describe a B-spline that is plotted as $\theta_{leastSquaresConditions}$ in Figure 5.11. Compared to $\theta_{leastSquares}$ in Figure 5.10 the heading angle doesn't track the velocity vector heading as closely at the beginning which is to be expected because the coefficients in that region were changed. However, since the entire trajectory is still within the tolerance limits, it is a valid initial guess for the optimiser and no additional knots need to be added.

The following weightings were chosen, $j_\theta = 10000$ and $j_{\ddot{\theta}} = 0.0001$ which were

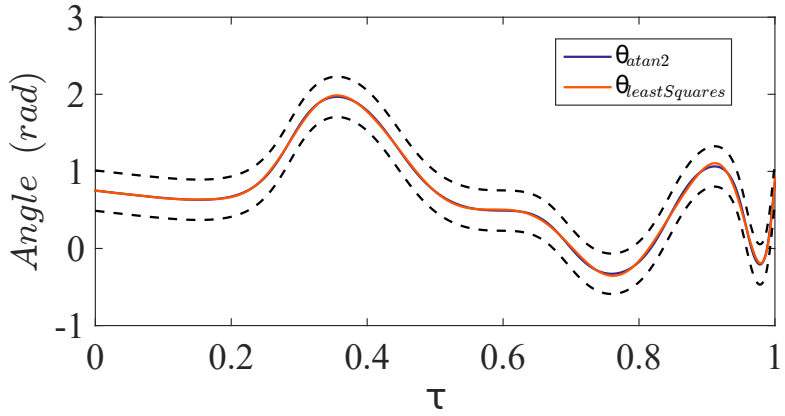


Figure 5.10: Heading angle optimisation example: least-squares spline approximation – A B-spline approximation of the heading angle (Figure 5.9) calculated using the least-squares spline approximation.

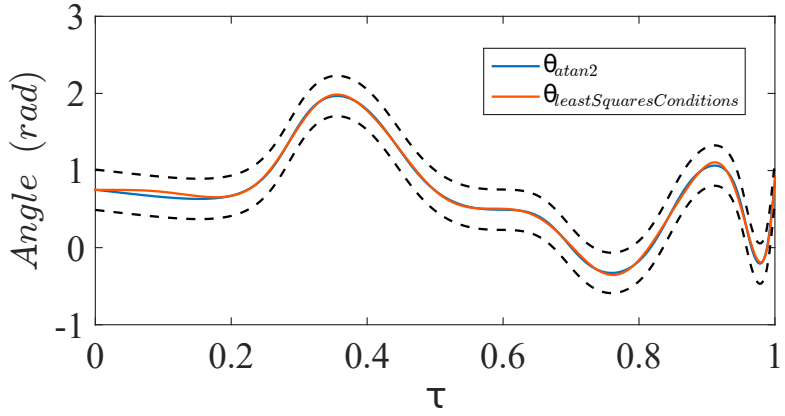


Figure 5.11: Heading angle optimisation example: initial guess – The coefficients from $\theta_{leastSquaresCondition}$ are used as the initial guess for the numerical minimiser after replacing the coefficients that are fixed.

obtained by experimenting with a variety of trajectories and found to work for a variety of scenarios. The large difference in the order of magnitude was used to ensure the heading angle remained within the specified bounds. Since the initial guess for the optimiser is within these bounds the optimised solution should also be so. After solving the minimisation problem (5.22), the optimised coefficient vector was obtained:

$$\chi = [0.75,\ 0.75,\ 0.75,\ 0.75,\ 0.75,\ 0.72,\ 0.98,\ 1.02,\ 1.80,\ 1.90,\\ 1.42,\ 1.03,\ 0.50,\ 0.34,\ -0.06,\ 0.14,\ -0.51,\ 1.17,\ 0.27,\ 1.66,\ -0.67,\ 0.23,\ 0.60] \tag{5.26}$$

The optimised heading angle $\theta_{opt}(\tau)$ is shown in Figure 5.12. The accumulated angular acceleration cost for $\theta_{leastSquaresCondition}(\tau)$ was $J_{\ddot{\theta}_{opt}} = 9.70 \times 10^5$. For $\theta_{opt}(\tau)$ the accumulated angular acceleration cost was $J_{\ddot{\theta}_{opt}} = 1.26 \times 10^5$ which is a decrease of 87%.

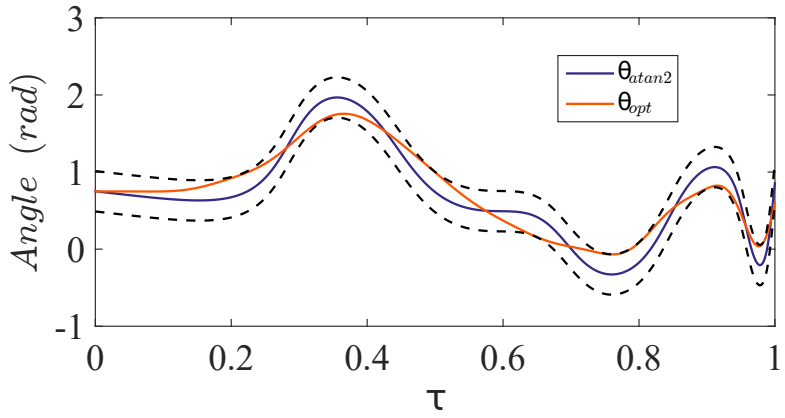


Figure 5.12: Heading angle optimisation example: optimised spline – The optimised heading angle θ_{opt} described by a B-spline with the coefficients found using a numerical minimiser.

5.3.4 Monte Carlo test of the knot vector length

Choosing the number of unique knots in the knot vector is a compromise between solution quality and computational time. In the case when a very low number of unique knots is used it may be impossible to find a B-spline that remains within the tolerance bounds throughout the trajectory. At the other extreme when too many knots are used the increased computational challenge can cause the numerical optimiser to struggle to converge on the solution. A numerical, Monte

Carlo simulation was designed to experimentally test and investigate the effect of varying the number of knots. Figure 5.14 shows the 1000 planar trajectories that were created with varying numbers of waypoints, chosen randomly, between 5 and 10. The Euclidean distance ΔW between adjoining points was constrained to be $3\,\mathrm{m} < \Delta W < 5\,\mathrm{m}$. The final constraint was on the angle between the vector of the previous waypoint and the current waypoint, and the vector between the current waypoint and the one about to be added. The angle between the vectors ψ_m was constrained to be within $-\frac{\pi}{2}\,\mathrm{rad} \leq \psi_m \leq \frac{\pi}{2}\,\mathrm{rad}$ as defined by Figure 5.13. The purpose of this constraint was to develop a variety heading angle profiles that were representative of the types of trajectory found in circuit racing. It is unlikely for the vehicle to double-back on itself as would sometimes happen if the angle was unconstrained.

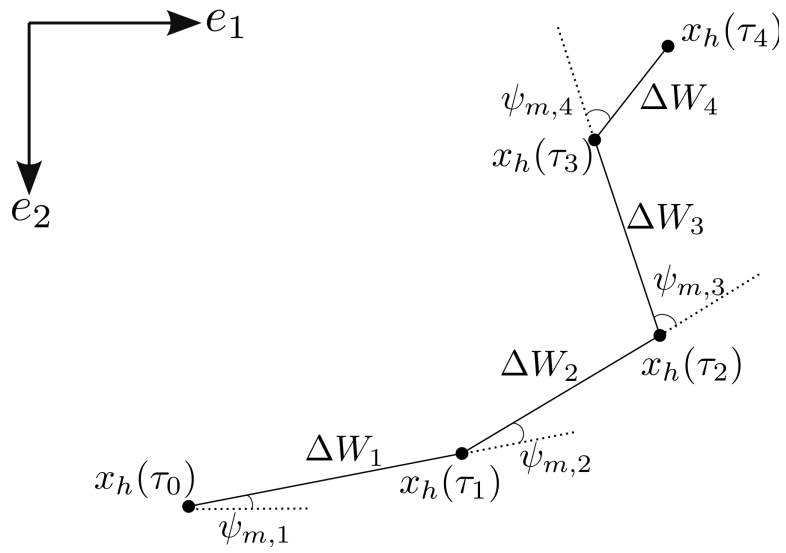


Figure 5.13: Heading angle Monte Carlo test geometry – The locations of the waypoints are defined in manner similar to polar coordinates. The distance between consecutive coordinates is ΔW with an angle between the vectors ψ_m.

For each trajectory, the unwrapped heading angle $\theta_{unwrapped}$ was calculated and five different θ_{tol} values were defined in the vector:

$$\theta_{\mathbf{tol,range}} = [0.26,\ 0.35,\ 0.44,\ 0.52,\ 0.61]\,\mathrm{rad} \tag{5.27}$$

Five different knot vectors were also defined, these will be referred to by the number of unique knots in each which are as follows $\kappa_{\mathbf{range}} = [10,\ 15,\ 20,\ 25,\ 30]$. For each combination of $\theta_{\mathbf{tol,range}}$ and $\kappa_{\mathbf{range}}$ an optimised heading angle using the

process described in the previous section is found for each of the 1000 trajectories. The stopping criteria for the numerical minimiser was a precision tolerance on the free parameters and function cost of $1e{-}8$.

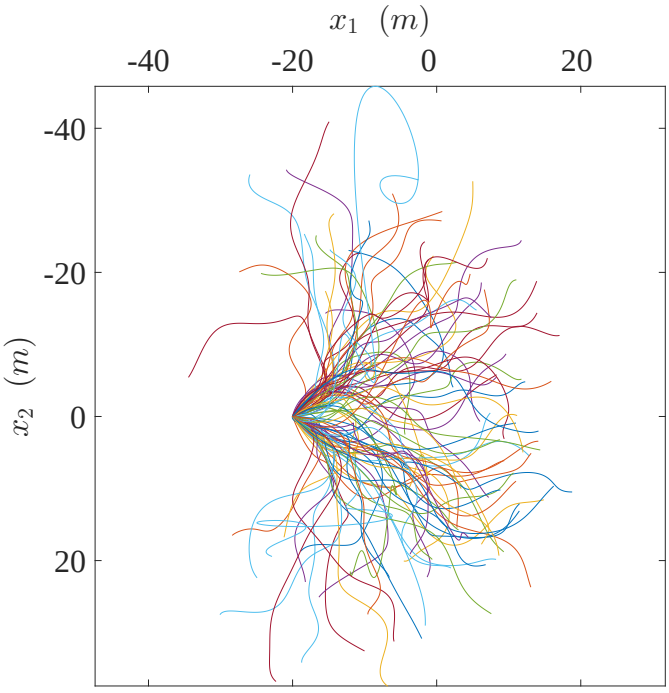


Figure 5.14: Heading angle parameter selection: 1000 trajectories – All the planar trajectories generated by the Monte Carlo process.

The computation time T_{comp} for each parameter combination and trajectory were recorded. Since these computation times varied considerably for the different trajectories a method of normalising the times was devised in order to prevent outliers from skewing the results. This was achieved by taking the maximum computation time $T_{comp,max}$ of all the $\kappa_{\mathbf{range}}$ for a given trajectory. The normalised computation time is then calculated for each trajectory:

$$T_{comp,norm} = \frac{T_{comp}}{T_{comp,max}} \tag{5.28}$$

The mean of the normalised computation times $\bar{T}_{comp,norm}$ are then taken for each combination of parameters. Figure 5.15 shows this against the heading angle tolerance for each of the different numbers of nodes. Generally, stricter tolerance bounds mean the solution is found faster because there is less scope for

improving the heading angle. However, for the case of 10 unique knots there is little difference in the computation time between the smallest tolerance bounds and the largest tolerance bounds.

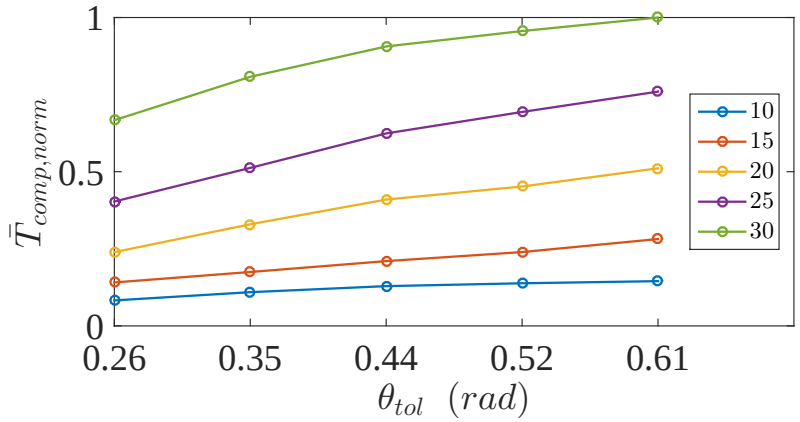


Figure 5.15: Heading angle Monte Carlo test: computational cost – The computational cost for each combination of κ and θ_{tol}. Using fewer knots reduces the computation time.

To compare the acceleration cost improvement for each combination of parameters the following method was used. The angular acceleration cost when $\theta_{leastSquaresConditions}$ described the heading angle was found for each trajectory when the maximum number of unique nodes, in this case 30, was used. The maximum number of unique nodes was used because the approximated curve is closer to $\theta_{unwrapped}$. The angular acceleration cost percentage decrease was then found for every parameter combination for that particular trajectory. The same $\theta_{leastSquaresConditions}$ was used for all the parameter combinations in each trajectory to ensure the comparison between them was fair. The mean angular cost $\bar{J}_{\ddot{\theta},norm}$ from all the trajectories for each combination was then calculated and shown in Figure 5.16. When θ_{tol} is small there is little difference between $\bar{J}_{\ddot{\theta},norm}$ for each of the different numbers of knots because again, there is little scope for improvement. As θ_{tol} increases having fewer nodes performs better in this Monte Carlo simulation. This is most likely due to convergence issues within the numerical minimisation because there are too many free parameters. It is possible that in long trajectories with many sharp bends increasing using a large number of knots is necessary. However, for the trajectories considered here it is unnecessary. Therefore, from this Monte Carlo simulation the results show that using a relatively small number of knots provides the best solutions.

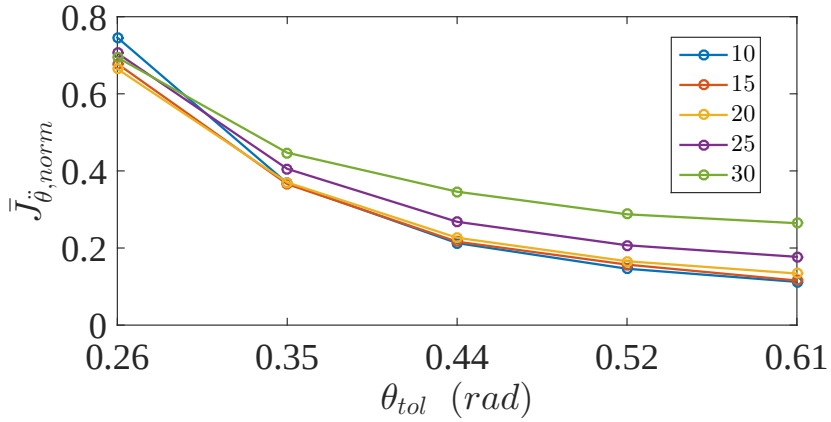


Figure 5.16: Heading angle Monte Carlo test: angular acceleration cost – The normalised, average angular acceleration costs for the range of θ_{tol} and number of coefficients used.

5.4 Chapter summary

In this chapter the trajectory generation within the virtual domain for drone racing was developed. Starting with the translational motion, polynomials were used as basis function to describe the position on each axis as the function of the abstract parameter τ. A method of formulating linear equations that could be solved to find the polynomial coefficients which satisfied the conditions placed on the trajectory was given. An example trajectory was given which demonstrated the use of a numerical minimiser to find the trajectory that satisfied all the conditions and had the shortest geometrical path path length. Unlike the sub-Riemannian curves derived in Chapter 3 the derivatives at the boundaries can be chosen making this trajectory generation method more suitable for drone racing.

A method for ensuring the trajectories do not collide with the gates used to mark drone racing courses was developed. The Waypoint Selection Algorithm adds correction waypoints where necessary and two examples were given in this chapter of its use. In the first example a single gate was placed between the initial and final position and the algorithm quickly found extra waypoints to avoid colliding with the obstacle. The second example had three gates rotated about multiple axes placed in a three-dimensional environment. Once again, a collision free trajectory was found using the Waypoint Selection Algorithm. To demonstrate the benefits of using this algorithm an alternative approach was also used that simply placed waypoints on the first body axis of the gate. A collision

free trajectory was found with this method but it required tighter turns than those found using the Waypoint Selection Algorithm.

Finally a method for optimising the heading angle was developed. A cost function that minimised the accumulated acceleration whilst staying within prescribed tolerance limits was defined. B-splines were used as function to parametrise the heading angle and the coefficients were found using a numerical minimiser. An example that demonstrated how to formulate the knot vector and coefficient vector that define a B-spline was given. A Monte Carlo simulation was performed to numerically investigate the effect of increasing the number of unique nodes in the knot vector and the size of the tolerance angle limits.

In the following chapter the virtual domain trajectories developed in this chapter are mapped to the time domain. This is required for the trajectories to be feasible and address optimality. During the mapping process it is sometimes necessary to recompute the virtual domain trajectory, the reasons for this are given in the next chapter. When this does occur, the same method described in this chapter is used.

Chapter 6

Mapping trajectories to the time domain

In the previous chapter, trajectories were planned in the virtual domain and were parametrised by the abstract parameter τ. In order for these trajectories to be feasible and trackable by the controller in Chapter 7 they must be mapped into the time domain. This is achieved through the use of a mapping function that allows the trajectory to be parametrised in the real time domain t. This trajectory can then be used as a reference trajectory because during the mapping process the kinodynamic limits, such as the limit of thrust available or maximum acceleration can be accounted for, unlike the virtual domain trajectory found previously.

Mapping a trajectory from the virtual domain to the time domain is equivalent to separating the temporal information from the spatial information, a technique developed by Bobrow for robotic manipulators [134]. Time mapping is also known as adjusting the velocity profile, planning in the virtual domain, time parametrisation and time scaling. The geometric path remains the same before and after mapping but the rate at which the vehicle moves along it is different. This has advantages and disadvantages for the motion planning. For obstacle avoidance it is a useful property to have because a trajectory in the virtual domain that avoids the collision region will do the same in the time domain. The disadvantage is that the shape of the trajectory is not directly considered when optimising the motion so while the mapped solution may be optimal for one virtual domain trajectory, it may not be globally optimal. However, as discussed in Chapter 1, pseudo-optimal solutions are still useful.

The same mapping must be applied to the motion along each axis. For this reason it is impossible to change derivatives of position (such as velocity and acceleration) independently so appropriate derivatives must be be chosen in the virtual domain. Rotational motion can also be mapped, as is necessary for the heading angle in the case of quadrotors. Examples of different systems where time mapping has been applied to include spacecraft [135, 136] and robotic arms [137]. A point-to-point quadrotor virtual trajectory generation method was presented in [138] but the manoeuvres are rest-to-rest and the yaw angle is not considered. A well-chosen mapping function has a high level of control over the relationship between the virtual domain and the time domain. The kinodynamic limits of the vehicle can be pushed as much or as little as desired. One of the appealing aspects of time mapping is its suitability for minimising trajectory time because the final trajectory time does not need to be known in advance. As discussed in Chapter 2, drone racing is essentially a minimum time problem so this method of optimisation is ideal. This chapter will develop methods for optimising the trajectory time whilst remaining feasible for a drone to fly. The vehicle model and layout is that of the Standard configuration with the parameters defined in Chapter 2. In this chapter it is assumed the thrust required from each rotor can be calculated by the translational states and the heading angle. The method for doing this is called inverse dynamics and how this is achieved in this thesis is given in Chapter 7.

Original Contributions

The original contributions in this chapter are outlined as follows:

- A multi-weighted cost function that includes the trajectory time and feasibility of the trajectory with respect to the thrust capabilities of the rotors is given.
- An algorithm that quickly finds the two boundary mapping values with the minimum trajectory time when only boundary nodes are present is developed. This enables dynamically feasible, non-zero boundary conditions on the derivatives to be found.
- Three methods for finding the internal mapping nodes (those between the boundary mapping nodes) are developed and compared. The first is a heuristic approach that seeks to lower the mapping value of each node by an algorithmic approach. The second also uses power series polynomials as

the basis function but the multi-weighted cost function is applied with a numerical optimiser. The final approach uses the same cost function but instead of polynomials a B-spline is used as the basis function.

- The best parameter choice for when B-splines are used as the basis functions for the time mapping was investigated numerically with a Monte Carlo simulation.

The chapter is structured as follows. In Section 6.1 the concept of mapping functions that convert a trajectory in the virtual domain onto the time domain is introduced. A multi-weight cost function is given in Section 6.2.1 that accounts for the dynamic feasibility of the trajectory and minimises the final trajectory time. Then, a method for algorithmically determining the boundary values efficiently is presented in Section 6.2.2. Three methods are developed in Section 6.2.3 to find the interior mapping nodes and compared against each other. A Monte Carlo simulation in Section 6.2.4 is used to investigate the choice of parameters when a B-spline is used to parametrise the mapping function. Section 6.3 concludes the chapter with a summary of the findings.

6.1 Mapping function

The mapping function $\lambda(\tau)$ that maps the virtual domain τ to the real time domain t is defined as:

$$\lambda(\tau) = \frac{dt}{d\tau} \tag{6.1}$$

or equivalently:

$$t(\tau) = \int_0^{\tau} \lambda(\tau)\, d\tau \tag{6.2}$$

A valid mapping function must satisfy the constraint $\lambda(\tau) > 0$ because time increases monotonically. The trajectory states in the time domain can now be written as a function of virtual domain states and the mapping function. In this thesis expressions up to the fourth derivative of position with respect to time are provided because the inverse dynamics formulation in Chapter 7 requires control up to this degree. The states in the virtual domain are no longer power series polynomials but they are still continuously differentiable C^{∞} and smooth. The

translational position on axis a in the inertial reference frame is simply:

$$x_a\big(t(\tau)\big) = P_a(\tau) \tag{6.3}$$

By applying the chain rule, the velocity on a given axis in the time domain is found to be:

$$\dot{x}_a(t(\tau)) = \frac{dP_a(\tau)}{d\tau}\frac{1}{\lambda(\tau)} \tag{6.4}$$

Likewise, the acceleration is:

$$\ddot{x}_a(t(\tau)) = \frac{d^2P_a(\tau)}{d\tau^2}\frac{1}{\lambda(\tau)^2} - \frac{dP_a(\tau)}{d\tau}\frac{d\lambda}{d\tau}\frac{1}{\lambda(\tau)^3} \tag{6.5}$$

and jerk:

$$\begin{aligned}\dddot{x}_a(t(\tau)) = 3\frac{d\lambda(\tau)}{d\tau}^2\frac{dP_a(\tau)}{d\tau}\frac{1}{\lambda(\tau)^5} + \frac{d^3P_a(\tau)}{d\tau^3}\frac{1}{\lambda(\tau)^3} \\ -3\frac{d\lambda(\tau)}{d\tau}\frac{d^2P_a(\tau)}{d\tau^2}\frac{1}{\lambda(\tau)^4} - \frac{d^2\lambda(\tau)}{d\tau^2}\frac{dP_a(\tau)}{d\tau}\frac{1}{\lambda(\tau)^4}\end{aligned} \tag{6.6}$$

Finally, the fourth derivative jounce is:

$$\begin{aligned}\ddddot{x}_a(t(\tau)) = 15\frac{d^2P_a(\tau)}{d\tau^2}\frac{d\lambda(\tau)}{d\tau}^2\frac{1}{\lambda(\tau)^6} - \frac{dP_a(\tau)}{d\tau}\frac{d^3\lambda(\tau)}{d\tau^3}\frac{1}{\lambda(\tau)^5} \\ +10\frac{dP_a(\tau)}{d\tau}\frac{d\lambda(\tau)}{d\tau}\frac{d^2\lambda(\tau)}{d\tau^2}\frac{1}{\lambda(\tau)^6} - 15\frac{dP_a(\tau)}{d\tau}\frac{d\lambda(\tau)}{d\tau}^3\frac{1}{\lambda(\tau)^7} \\ -6\frac{d\lambda(\tau)}{d\tau}\frac{d^3P_a(\tau)}{d\tau^3}\frac{1}{\lambda(\tau)^5} - 4\frac{d^2\lambda(\tau)}{d\tau^2}\frac{d^2P_a(\tau)}{d\tau^2}\frac{1}{\lambda(\tau)^5} \\ +\frac{d^4P_a(\tau)}{d\tau^4}\frac{1}{\lambda(\tau)^4}\end{aligned} \tag{6.7}$$

The mapping function $\lambda(\tau)$ can be parametrised using a basis function just as the translational position and heading angle were in Chapter 5. In this thesis B-splines and polynomials are both candidate functions for the mapping and are compared in Section 6.2.3. The boundary value finding method in 6.2.2 can also be performed with either candidate function. The mapping optimisation vector Ξ_χ contains the mapping values currently being optimised. In this chapter the method for forming the polynomials and B-splines from the conditions is not given because the process is very similar to that described in Chapter 5.

6.1.1 Matching the boundary conditions

The required boundary values of the velocity $\frac{dP_a(\tau)}{d\tau}$, acceleration $\frac{d^2P_a(\tau)}{d\tau^2}$, jerk $\frac{d^3P_a(\tau)}{d\tau^3}$ and jounce $\frac{d^4P_a(\tau)}{d\tau^4}$ in the virtual domain can be obtained by rearranging (6.4), (6.5), (6.6) and (6.7) respectively. However, all derivatives before a desired constraint must be known if rearranged directly. For instance if only the acceleration was to be constrained a velocity would have to be specified. This undesirable limitation can be avoided by specifying that any derivatives of $\lambda(\tau)$ at the boundaries are zero. In our formulation this means:

$$\dot{\lambda}(\tau_i) = \dot{\lambda}(\tau_f) = \ddot{\lambda}(\tau_i) = \ddot{\lambda}(\tau_f) = \dddot{\lambda}(\tau_i) = \dddot{\lambda}(\tau_f) = 0 \tag{6.8}$$

This assumption means that if $\lambda(\tau)$ at the boundaries is known, the condition on the derivatives at the boundary expressed in the time domain can be expressed as:

$$P_a{}^{(r)}(\tau) = x_a(t(\tau))^{(r)}\lambda^r(\tau) \tag{6.9}$$

The initial value of $\lambda(\tau)$ at the boundaries are found numerically because if the value is too small then it is impossible to find a mapping function that produces a feasible trajectory since the kinodynamic limits are exceeded. This is because when a larger value of $\lambda(\tau)$ is needed, $\dot{\lambda}(\tau)$ must be increased. This causes the higher order derivatives of the position to also increase, making the trajectory infeasible. This can be easily illustrated with a planar example on the first two axes. The boundary conditions are given in Table 6.1 and the three example trajectories are plotted in Figure 6.1. Since the direction of the initial velocity is specified, the trajectory planner is not free to optimise when attempting to minimise the path length. Three combinations of boundary mapping values are used. The first trajectory in Figure 6.1a has initial and final boundary mapping values of $\lambda(0) = 1$ and $\lambda(1) = 1$, so all the derivatives throughout the trajectory in the time domain are identical to those of the virtual domain. If a relatively small maximum acceleration was applied there is no choice of mapping function between the prescribed boundary mapping values that would enable a feasible trajectory. This is because the trajectory rapidly changes direction which requires an acceleration before the vehicle would have time to decelerate to a slower speed that would require a smaller acceleration for the same manoeuvre. In Figure 6.1c the boundary mapping values have increased to $\lambda(0) = 2$ and $\lambda(1) = 3$ and the curve has changed shape. The geometrical length is now longer but

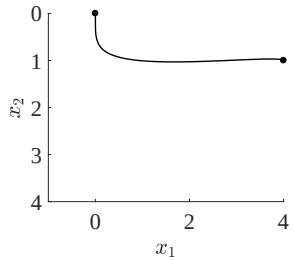
the distance from the starting point to when the turn begins has also increased. The boundary mapping values are further increased in Figure 6.1e to $\lambda(0) = 4$ and $\lambda(1) = 8$. Again, the distance from the starting point to the first turn increases and the radius of curvature of the final turn to meet the boundary conditions on the velocity vector at the end is increased. All three trajectories match the boundary conditions in the time domain as required in Table 6.1 but as this example demonstrates, the boundary mapping values affect the shape of the trajectory and by choosing them carefully, the motion can be made feasible.

Condition	Value
$x_1^{(0)}(0)$	0
$x_1^{(1)}(0)$	0
$x_1^{(2)}(0)$	0
$x_1^{(3)}(0)$	0
$x_1^{(4)}(0)$	0
$x_1^{(0)}(T)$	4
$x_1^{(1)}(T)$	1
$x_2^{(0)}(0)$	0
$x_2^{(1)}(0)$	2
$x_2^{(2)}(0)$	0
$x_2^{(3)}(0)$	0
$x_2^{(4)}(0)$	0
$x_2^{(0)}(T)$	1
$x_2^{(1)}(T)$	1

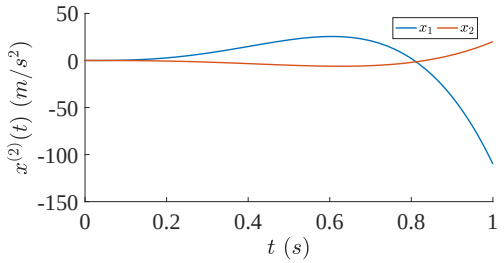
Table 6.1: Boundary mapping values: conditions – The boundary conditions placed upon the translational states when parametrised in the time domain.

6.2 Time mapping algorithms

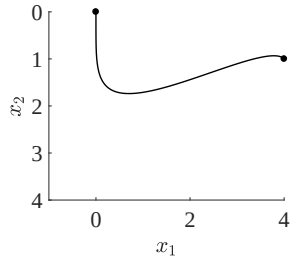
The optimal mapping function is one that maps a trajectory from the virtual domain into the time domain with the minimum trajectory time whilst remaining feasible. In this section the methods for finding this function are developed. In Section 6.2.1 a cost function is defined, then an algorithm for finding an initial guess of the boundary mapping values is given given in Section 6.2.2. Finally, Section 6.2.3 addresses how mapping values between the boundaries are found.



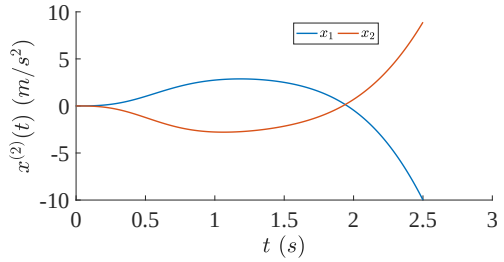
(a) Trajectory: $\lambda(0) = 1$ and $\lambda(1) = 1$.



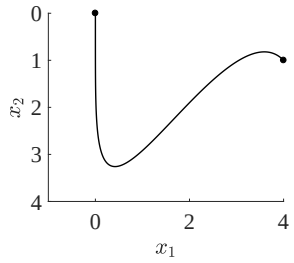
(b) Acceleration: $\lambda(0) = 1$ and $\lambda(1) = 1$.



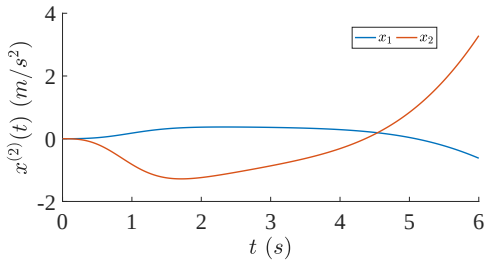
(c) Trajectory: $\lambda(0) = 2$ and $\lambda(1) = 3$.



(d) Acceleration: $\lambda(0) = 2$ and $\lambda(1) = 3$.



(e) Trajectory: $\lambda(0) = 4$ and $\lambda(1) = 8$.



(f) Acceleration: $\lambda(0) = 4$ and $\lambda(1) = 8$.

Figure 6.1: Effect of change the boundary mapping values on the trajectory shape – The combinations of boundary mapping values are applied to the same problem. The shape of the curve is dependent upon the choice and by increasing the values dynamic feasibility can be achieved.

6.2.1 Defining the mapping cost function

The multi-objective optimisation problem of minimising time and remaining feasible can be converted into single-objective problem by scalarising it:

$$J_m = j_T T + j_f {J_f}^2 \tag{6.10}$$

where j_T and j_f are scalar weightings for time and feasibility respectively. The trajectory feasibility cost is J_f and T is the final trajectory time. The trajectory time is calculated using (6.2) and the method for finding the feasibility cost is given below.

In this thesis a feasible trajectory is one in which the thrust required from each rotor is within specified limits throughout the entire motion. If each rotor satisfies the condition $f_{min} < f_{1-4}(t) < f_{max}$ then the feasibility cost is zero, $J_f = 0$. Effectively this means that so long as the trajectory remains feasible it does not contribute to the cost function (6.10). If one or more of the thrusts do not stay within the limits then the maximum violation is used as the feasibility cost. The violation of the lower and upper thrust limit, $J_{f,lower}$ and $J_{f,upper}$ respectively are calculated as follows:

$$\begin{aligned} J_{f,lower} &= |min(f_{1-4}(t)) + f_{min}| \\ J_{f,upper} &= |max(f_{1-4}(t)) - f_{max}| \end{aligned} \tag{6.11}$$

If $J_{f,lower} > J_{f,upper}$ then the feasibility cost is $J_f = J_{f,lower}$ otherwise it is $J_f = J_{f,upper}$. In practice the latter is most often the case because it is only when the trajectory requires an acceleration in the direction of $\mathbf{e_3}$ that is similar in magnitude to the gravitational acceleration $g_a = 9.8\,\mathrm{m/s^2}$ that the lower thrust limit is reached.

6.2.2 Determining boundary mapping values

The method described in this section to find the boundary mapping values is universal to the three methods for finding the interior mapping nodes that are developed in Section 6.2.3. It has already been explained in Section 6.1.1 that to match the boundary conditions the boundary mapping values must be sufficiently large to ensure the trajectory is dynamically feasible throughout. If a mapping

function that produces a feasible trajectory is defined by some function (such as a polynomial or B-spline) with only boundary mapping the interior mapping values can be found later. This ensures that a feasible but sub-optimal solution is quickly produced that can be improved by optimising the interior nodes to reduce the trajectory time.

Early attempts at finding the boundary mapping values used standard numerical optimisers such as **fmincon** and **patternsearch**. However, because the virtual trajectory needs to be recomputed at every iteration the solution time was excessively long because standard optimisers have no knowledge of the problem being solved. For instance, there are usually two peaks of maximum dynamic infeasibility, one near the beginning and one near the end. The maximum infeasibility of a given peak can be reduced by increasing the corresponding mapping value. Additionally, the standard optimisers require upper and lower bounds to be given for the optimisation vector. The values of the optimisation vector are problem dependent and it is difficult to predict their size in advance. Although an infinitely large bound can be applied this will result in iterations with excessively large guesses that will extend the computation time. The problem is exacerbated if the interior mapping values are also found at the same time as the boundary values. However, if a good initial guess is obtained the standard numerical optimisers perform better. The method in this section can be thought of as finding that initial guess.

Since the optimisation vector used to find the boundary mapping values only has two parameters, the initial and final mapping value, developing a time minimisation algorithm specific to this problem is relatively simple. Figure 6.2 explains the method that was developed. The first step is to initialise the boundary mapping values to some minimum value. A convenient starting point is $\lambda(0) = \lambda(1) = 1$ so the trajectory in the time domain is identical to that of the one in the virtual domain. For the purposes of checking the dynamic feasibility the trajectory was divided into two halves. The feasibility cost of the first half is $J_{f,initial} = J_f([0, T/2])$ and for the second half is $J_{f,final} = J_f([T/2, T])$ where $J_f([0, T/2])$ is the maximum feasibility cost when $t \in [0, T/2]$ and $J_f([T/2, T])$ is the maximum feasibility cost when $t \in [T/2, T]$. If the trajectory in infeasible during the first half, $J_{f,initial} > 0$, then the initial mapping value $\lambda(0)$ is doubled at the next iteration. Similarly, if $J_{f,final} > 0$ then the final mapping value $\lambda(1)$ is doubled.

A new virtual domain trajectory is obtained with the new mapping values and the same process for checking the dynamic feasibility is applied and the process is repeated. If one half of the trajectory is feasible then the mapping value on that side is held constant for the next iteration. The other mapping value is then increased until the entire trajectory is dynamically feasible. Then the boundary values are refined by taking the average of the current value and the largest mapping value of when trajectory was still infeasible. This is repeated until one of two tolerance criteria is reached. The first is a minimum step size for the boundary values and the second is when the maximum thrust is within some defined region from the maximum allowable thrust. The first criteria stopping criteria is required because sometimes the maximum thrust on one side does not come close to the maximum allowable thrust. Decreasing the step size and maximum thrust region will result in an improved trajectory time but the algorithm takes longer to converge.

To illustrate the method for finding the boundary mapping values an example is given in Figure 6.3. In this example a B-spline curve was used as the basis function for the mapping function (Section 6.2.3.3). However, the same method is used when polynomials are used to define the mapping function (Section 6.2.3.1 and Section 6.2.3.2). The dashed lines represent the boundary mapping values (left vertical axis) and the solid lines represent the logarithmic of the maximum thrusts (right vertical axis). The red circle marks represent values related to the initial boundary condition and the blue square marks relate to the final boundary condition. The logarithmic applied to the maximum thrust was necessary because the thrusts that correspond to small mapping values are several orders of magnitude larger than the thrust limit. In this respect, the graph understates the ability of the method to rapidly converge on a feasible solution. During iterations one through to nine the algorithm is finding the mapping values that give a feasible trajectory throughout. At iteration six a feasible thrust for the first half is found and the algorithm no longer attempts to increase the mapping value. At iteration nine the maximum thrust for the second half of the trajectory is found and the refining stage can begin. The mapping values for both the initial and final boundary nodes were reduced, with the greatest improvement shown for the latter. This is to be expected due to the exponential increase of the mapping value caused by doubling them in the previous stage. The number of iterations required to find the boundary mapping values is a fraction of the

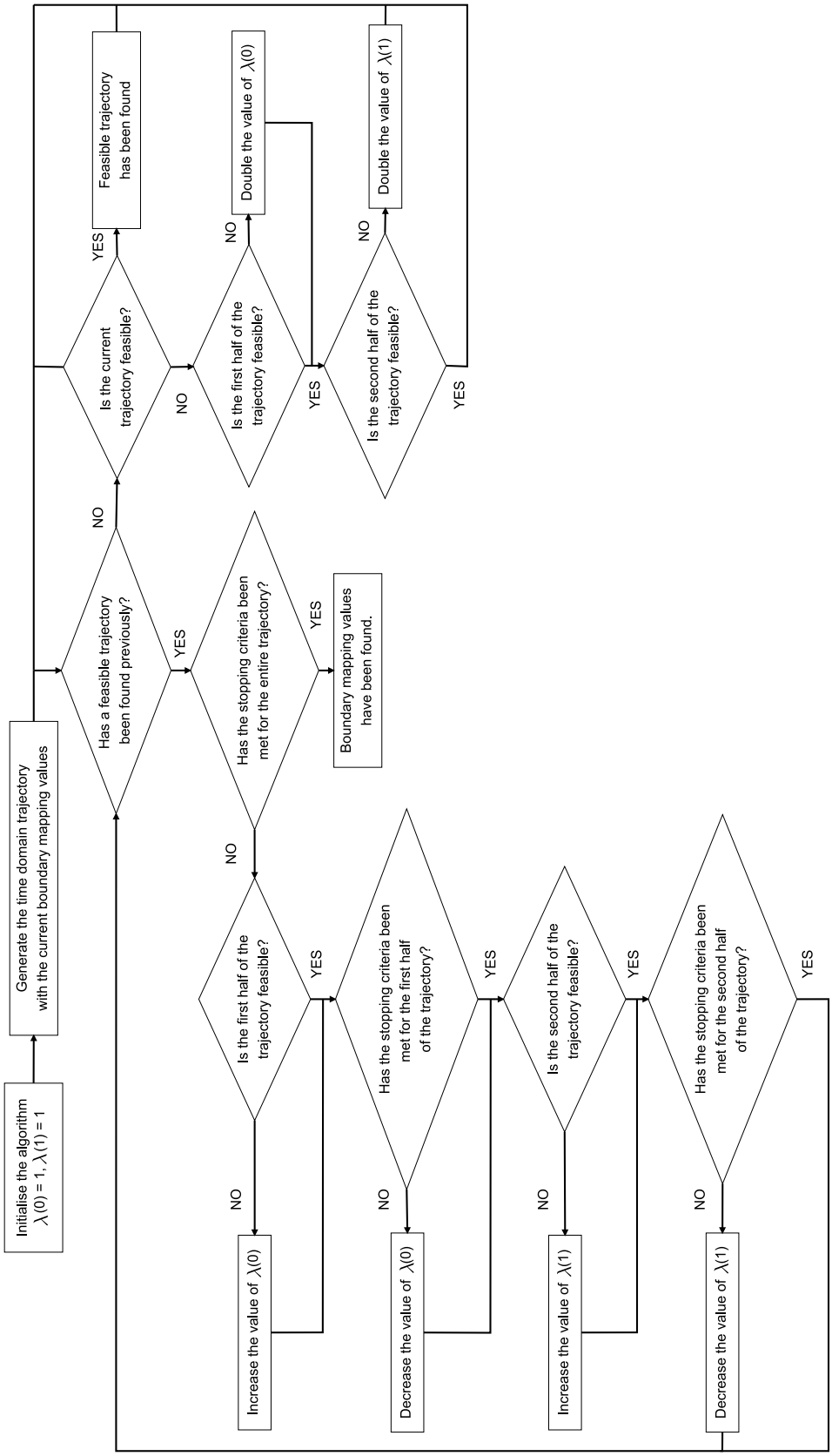


Figure 6.2: Boundary mapping algorithm – A flowchart of the boundary mapping algorithm used to find the mapping values at the beginning and end of the trajectory.

number required when using standard numerical optimisers. The solution found using the boundary mapping algorithm and a B-spline as the basis function was $\lambda(0) = 9.50$, $\lambda(1) = 39.75$ which resulted in a trajectory time $T = 24.63\,\mathrm{s}$.

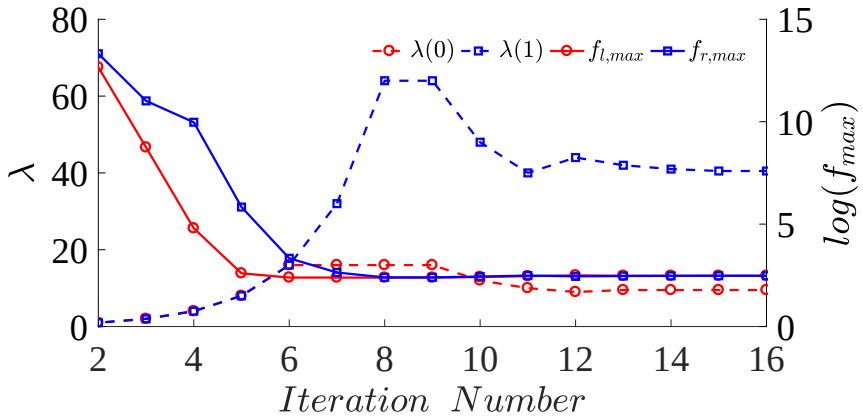


Figure 6.3: Boundary mapping algorithm example – The solid lines denote the maximum thrust violation in the first and second half of the trajectory. The dashed lines denote the boundary mapping values.

For the same boundary mapping problem addressed in Figure 6.3, a two-dimensional contour plot of the final trajectory time T is given in Figure 6.4. The purpose of this figure is to demonstrate the ability of the method used to determine the near-optimum boundary mapping values. It was generated by starting with mapping values $\lambda(0) = \lambda(1) = 1$ and increasing them in unit increments to $\lambda(0) = \lambda(1) = 100$ whilst ensuring all combinations in between were tested. If a given combination of $\lambda(0)$ and $\lambda(1)$ did not produce a dynamically feasible trajectory then it was not included in the contour plot. The best solution from this exhaustive search was $\lambda(0) = 10$, $\lambda(1) = 37$ that resulted in a trajectory time $T = 23.50\,\mathrm{s}$ which is slightly smaller than the one found by the algorithm. This discrepancy is due to the assumption made by the algorithm that $J_{f,initial}$ and $J_{f,final}$ can be treated independently. At the twelfth iteration the condition $f_{limit} - f_{tol} \leq f_{l,max} \leq f_{limit}$ is true so the initial mapping value is fixed at $\lambda(0) = 9.5$, however, if it was slightly larger then the final $\lambda(1)$ mapping value could be lower and result in reduced trajectory time. One solution to this would be to try a range of values of $\lambda(0)$ whilst choosing $\lambda(1)$. However, this would make the algorithm more complicated and harder to implement. It would also require more iterations to solve, thereby increasing the computation time. It was decided that the algorithm performed satisfactorily and further improve-

ments would be left to future work. Recalling the discussion in Chapter 1 on the subject of optimality, for motion planning problems, pseudo-optimal solutions are still 'good enough'. Additionally, since the interior mapping values still need to be found further improvements of the boundary mapping values may be possible after this is performed.

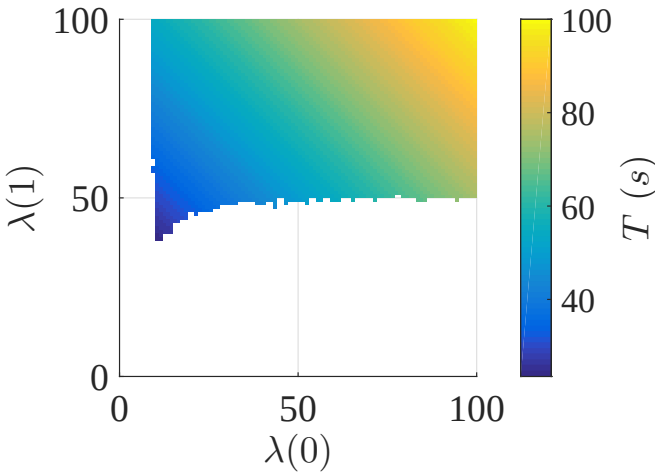


Figure 6.4: Boundary mapping value algorithm trajectory times – A contour plot of the final times relating to feasible trajectories. Trajectories that were not feasible and violated the thrust limits were ignored.

6.2.3 Three time mapping methods

In this section three methods for finding the interior mapping values – after the boundary mapping values have been determined – are compared. This is achieved by defining a drone racing scenario and running the motion planner using each of the three methods. The position and orientation of the gates are defined in Table 6.2 and they are all the same size: $r_{h,i} = 0.6\,\text{m}$, $r_{h,o} = 1.4\,\text{m}$ and $l_h = 0.5\,\text{m}$. The boundary conditions are given in Table 6.3. The starting and finishing translational position are $\mathbf{x}(0) = [0\,\text{m}, 0\,\text{m}, 0\,\text{m}]^T$ and $\mathbf{x}(T) = [14\,\text{m}, -2\,\text{m}, 0\,\text{m}]^T$ respectively. In this example the heading angle is fixed throughout the trajectory at $\mathbf{b_1} = [1,\ 0,\ 0]^T$. Time mapping when the heading angle is not static is demonstrated in the next chapter. As mentioned earlier in this chapter, all the simulations assume the standard layout quadrotor from Chapter 2 is being used.

Gate	h_1 (m)	h_2 (m)	h_3 (m)	$\phi_{h,1}$ (°)	$\phi_{h,2}$ (°)	$\phi_{h,3}$ (°)
1	0	−6	−1	0	0	−90
2	4	−10	−4	0	20	75
3	15	−5.5	−2.5	0	15	0

Table 6.2: Time mapping test: gate parameters – The translational position and attitude of the gates for this test.

Condition	Value
$x_1^{(0)}(0)$	0 m
$x_1^{(1)}(0)$	4 m/s
$x_1^{(2)}(0)$	0 m/s^2
$x_1^{(3)}(0)$	0 m/s^3
$x_1^{(4)}(0)$	0 m/s^4
$x_1^{(0)}(T)$	14 m
$x_1^{(1)}(T)$	0 m/s
$x_2^{(0)}(0)$	0 m
$x_2^{(1)}(0)$	−0.8 m/s
$x_2^{(2)}(0)$	0 m/s^2
$x_2^{(3)}(0)$	0 m/s^3
$x_2^{(4)}(0)$	0 m/s^4
$x_2^{(0)}(T)$	−2 m
$x_2^{(1)}(T)$	0 m/s
$x_3^{(0)}(0)$	0 m
$x_3^{(1)}(0)$	0 m/s
$x_3^{(2)}(0)$	0 m/s^2
$x_3^{(3)}(0)$	0 m/s^3
$x_3^{(4)}(0)$	0 m/s^4
$x_3^{(0)}(T)$	0 m

Table 6.3: Time mapping comparison: boundary conditions – The boundary conditions placed upon the translational states when parametrised in the time domain. The values were chosen arbitrarily but selected to show the derivatives can be specified to different values independently.

6.2.3.1 Mapping value heuristic optimisation using polynomials

An earlier version of this algorithm was presented by the author in [44]. In that paper the dynamic feasibility was considered using simple kinodynamic conditions

on the maximum speed and acceleration. This is a valid approach but the results tend to be overly-conservative because the kinodynamic constraints are simplified in comparison to the inverse dynamic approach used to find the thrust in this thesis. This version of the algorithm also assumes the boundary mapping values have already been found using the method described in the previous section.

Of the three methods presented in this section this is the only one that does not use the multi-objective cost function previously defined. Instead, the approach taken to minimise the trajectory time is to recognise that the area under the mapping function when $\lambda(\tau)$ is plotted against τ is the trajectory time in the time domain (6.2). Therefore, reducing this area will improve the trajectory time. Using a polynomial function to describe $\lambda(\tau)$, the mapping values that control the shape of this curve can be reduced where possible whilst ensuring the trajectory remains feasible. The heuristic approach taken alternates between the initial mapping value side and the final mapping value side and works in towards the middle. Each node is considered individually and reduced as much as possible before the trajectory until the point where the trajectory becomes infeasible. The method used to form and solve the polynomial is identical to how the motion on the translational axes is defined in Chapter 5. In summary, the conditions on the mapping function, including on its derivatives (6.8), are formed into a linear equation and solved to find the coefficients of the polynomial.

Figure 6.5 is a flow chart of the heuristic algorithm that is split into two stages. The first stage, 'Interior Value Finding', begins by taken the boundary mapping values found previously. The mapping value that corresponds to the second node is interpolated by numerically discretising the current solution. It is reduced until any further reduction would cause the trajectory to become infeasible. Next, the penultimate mapping value is considered and reduced in a similar manner. The third mapping value, then the third-last and so on are reduced until all the values have been minimised. The alternating approach is taken at this stage because it was found that when the order was linear from the initial value to the final value the trajectory time at the end of this stage was inferior. Now that all of the mapping nodes have values, they can be considered from the initial to final value repeatedly in the final stage, known as 'Value Refinement'. Once again, each node is considered individually and reduced when possible. When no further reductions can be made the algorithm terminates.

Using the scenario described by (Table 6.2 and Table 6.3) an example mapping

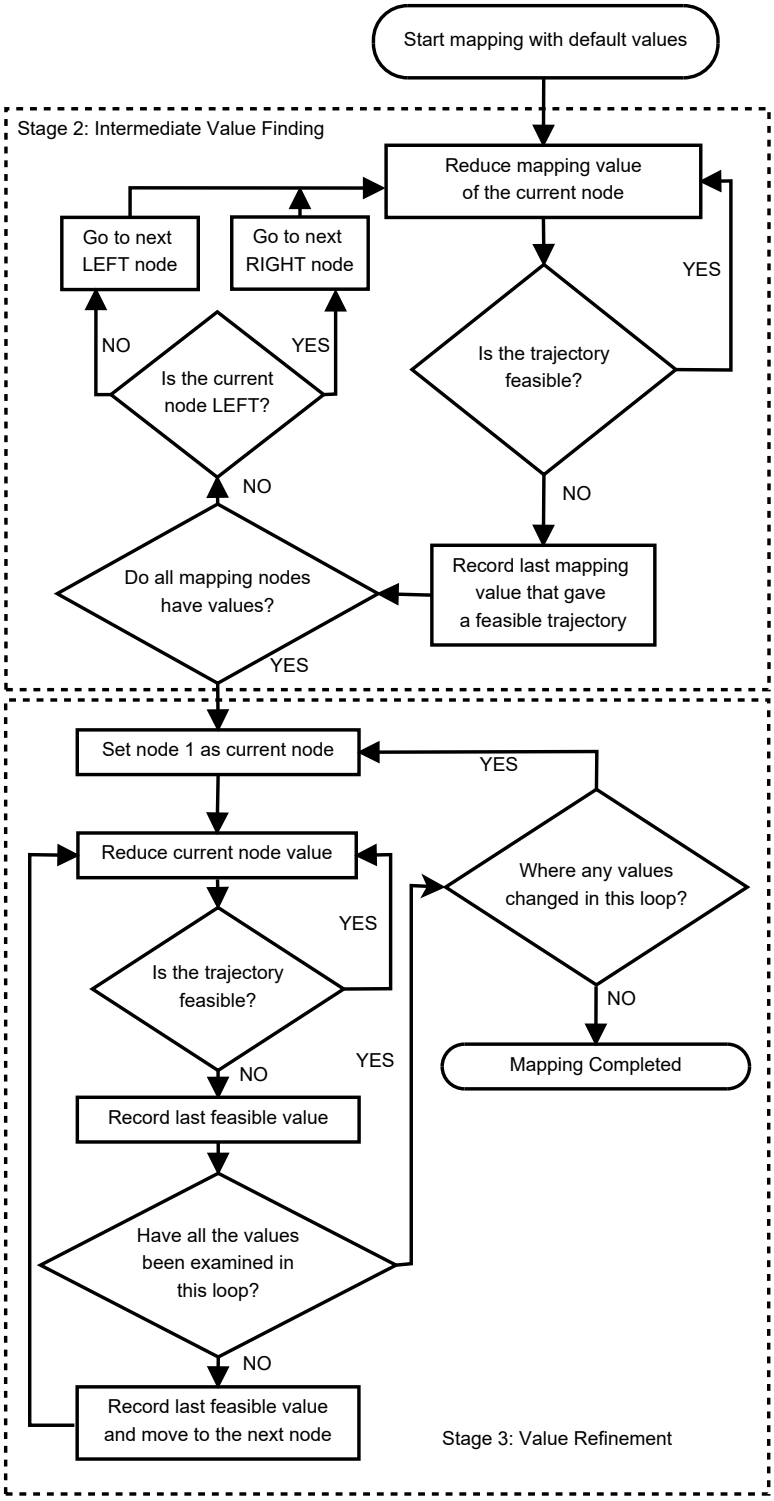


Figure 6.5: Heuristic time mapping algorithm – A flowchart of the logic behind the second and third stage of the Heuristic time mapping method. Before using this algorithm the boundary mapping values must be found first.

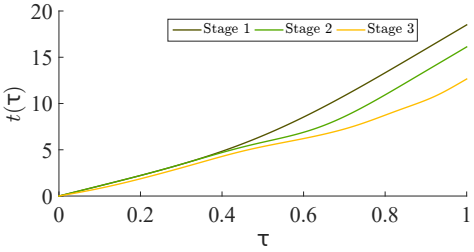
from the virtual domain to the time domain is performed using the heuristic approach described above. Twelve mapping nodes were distributed uniformly in the virtual domain and the minimum reduction allowed was 0.1. The relationship between the virtual domain and the time domain for each stage is given in Figure 6.6a. A planar view of the trajectory after the third mapping stage is shown in Figure 6.6b. The mapping functions for Stage 1, Stage 2 and Stage 3 are given in Figures 6.6c, 6.6e and 6.6g respectively. Similar the thrust profiles for each stage are given in Figures 6.6d, 6.6f and 6.6h

The trajectory time in the first, second and third stage is 18.50 s, 16.20 s and 12.65 s respectively. The greatest improvements to the trajectory time occurred when $\tau > 0.6$, which is to be expected because the final mapping value is considerably higher than the first so a greater proportion of the area under $\lambda(\tau)$ is after that point. In Figure 6.6d the two peaks of maximum thrust that occur when the boundary values are found are both visible. After defining and reducing the interior mapping nodes in Figure 6.6e the trajectory time is improved but in the region $7\,\text{s} < t < 15\,\text{s}$ the thrust capabilities are not being fully utilised because the thrusts are well below the limit of $12\,N$. The final refinement stage mapping function shown in Figure 6.6g improves the trajectory time, mainly by altering this region.

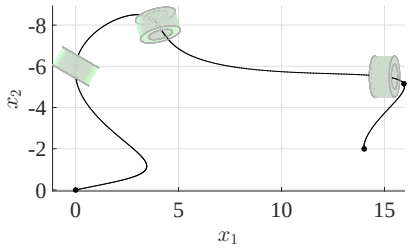
The translational states on each axis from position to the fourth derivative as found by in the final mapping stage are plotted against time in Figure 6.7. It can be seen that all the boundary state conditions from Table 6.3 are matched. The first axis requires the greatest number of changes in acceleration direction (Figure 6.7c) so it dominates the thrust magnitude.

6.2.3.2 Mapping value optimisation using polynomials

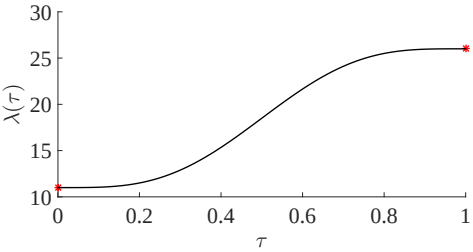
Like the heuristic mapping method, the approach introduced in this section also uses a polynomial to describe the mapping function. However, a numerical optimisation is applied to find the mapping values that minimise trajectory time by using the cost function defined in Section 6.2.1. There are three stages to this method. The first stage is to find the boundary mapping values with the method previously discussed. In the second stage the interior mapping values are found by minimising the cost function and in the final stage all the mapping values are considered.



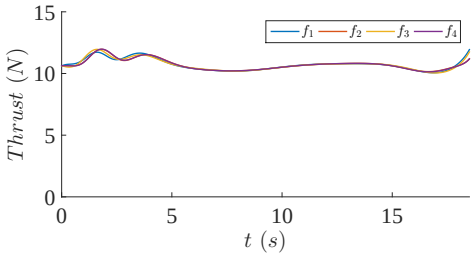
(a) Trajectory time against the virtual domain.



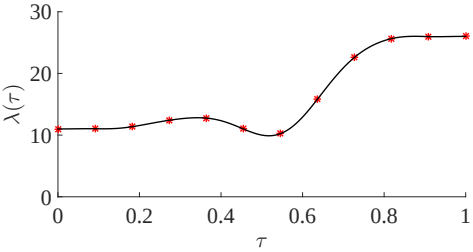
(b) The trajectory after Stage 3.



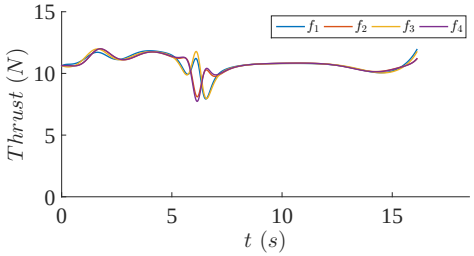
(c) Mapping function: Stage 1.



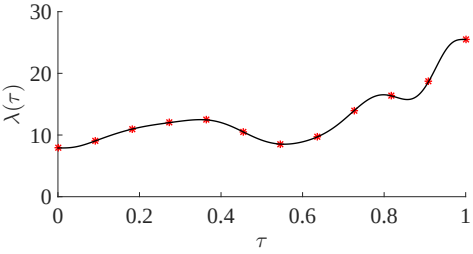
(d) Rotor thrusts: Stage 1.



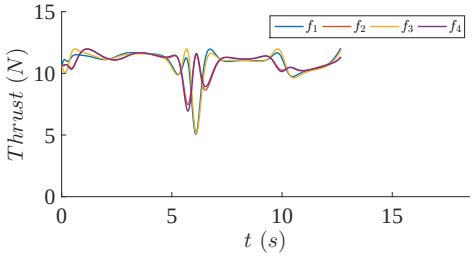
(e) Mapping function: Stage 2.



(f) Rotor thrusts: Stage 2.

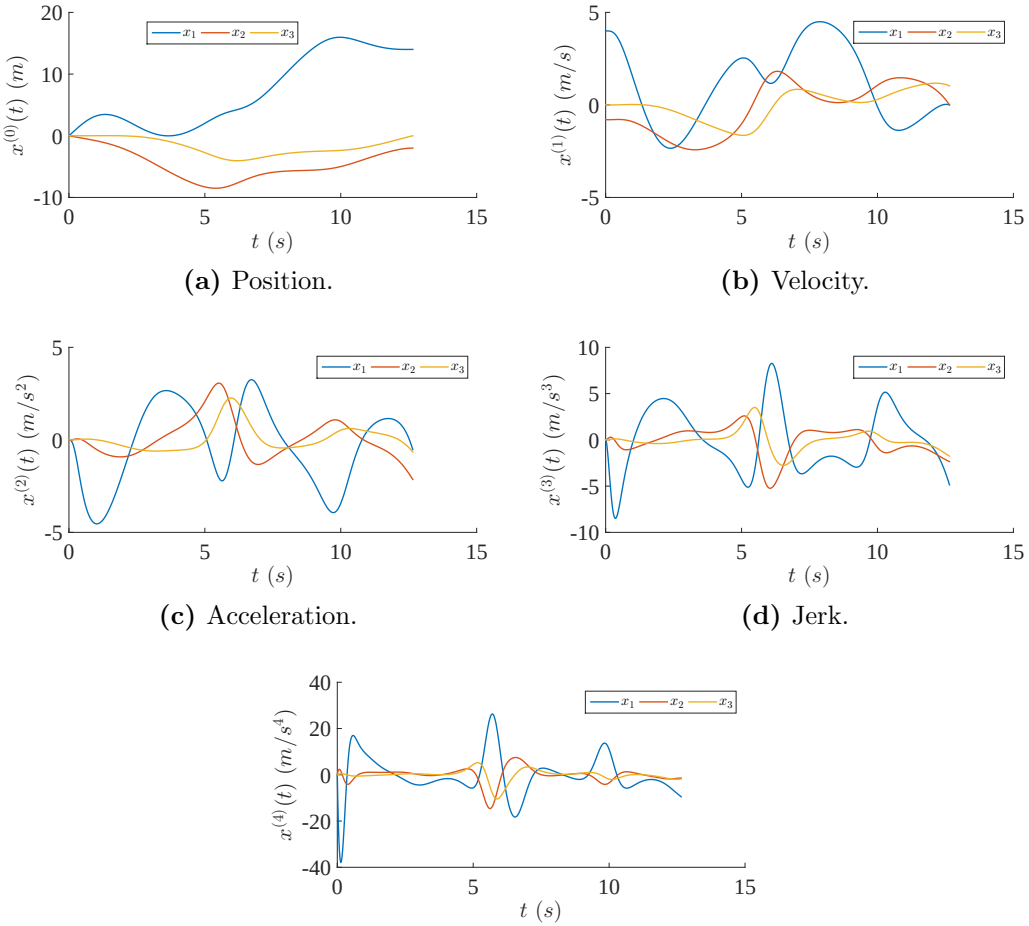


(g) Mapping function: Stage 3.



(h) Rotor thrusts: Stage 3.

Figure 6.6: Heuristic time mapping: example – The mapping stages for the heurstic time mapping example.



(a) Position. **(b)** Velocity.

(c) Acceleration. **(d)** Jerk.

(e) Jounce.

Figure 6.7: Heuristic time mapping: example states – The translational states up to the fourth derivative.

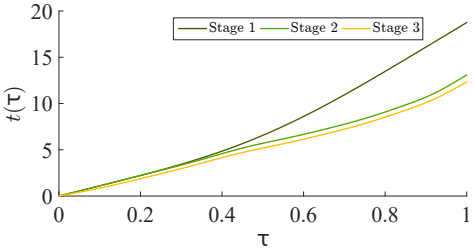
An initial guess for the interior mapping values in Stage 2 is obtained by interpolating between the discretised mapping function from the previous stage. The boundary mapping values are not considered because if they are included the virtual trajectory must be recalculated whenever either are changed which is computationally optimal and interior mapping values have not been optimised yet. The final stage is identical to the second except all the mapping values including the boundary ones are considered because a good guess should be available at this point and observations have shown the boundary values do not change significantly in this step. Therefore, the solution for the virtual domain trajectory can be found reasonably quickly. For this reason, the bounds placed on the initial and final mapping guessed values in the optimisation vector only need to allow small changes from the previous stage.

The same trajectory scenario described in Table 6.2 and Table 6.3 was used to test this method. Again, twelve nodes are used distributed across the virtual domain. The scalar weightings for the cost function are $j_T = 0.1$ and $j_f = 20$. The relationship between time and the virtual domain for each stage is given in Figure 6.8a. The trajectory times after Stage 1, Stage 2 and Stage 3 were respectively 18.50 s, 13.09 s and 12.32 s. The trajectory after the final mapping stage is given in Figure 6.8b. Mapping functions in the virtual domain and the rotor thrusts in the time domain for each stage are plotted in Figures 6.8c, 6.8e & 6.8g and 6.8d, 6.8f & 6.8h respectively.

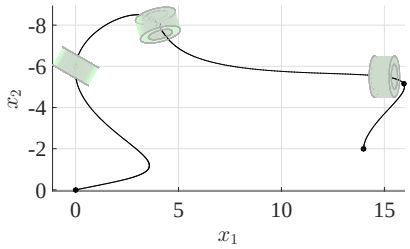
The first stage in this method is identical to the first stage of the previous method because both are defined using polynomials using the same algorithm to find the boundary values. Interestingly, there is no qualitative difference between the mapping functions and rotor thrusts in Stages 2 & 3, which is reflected by both having similar trajectory times. In this case there was minimal benefit to refining the boundary values. The translational states up to the fourth derivative in the time domain are given in Figure 6.9. All the boundary conditions were matched successfully.

6.2.3.3 Mapping value optimisation using B-splines

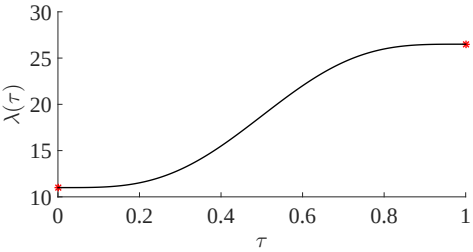
The method for optimising the mapping functions in this section is identical to that of the previous one, except B-splines are used as the basis functions instead of polynomials. The boundary nodes are clamped by repeating knots at both ends of



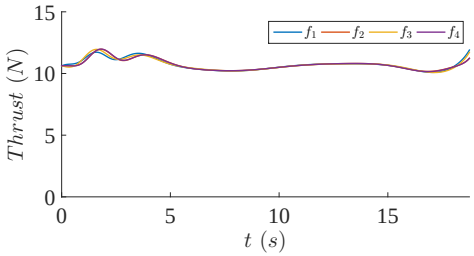
(a) Trajectory time against the virtual domain.



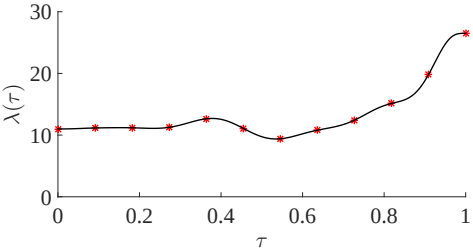
(b) The trajectory after Stage 3



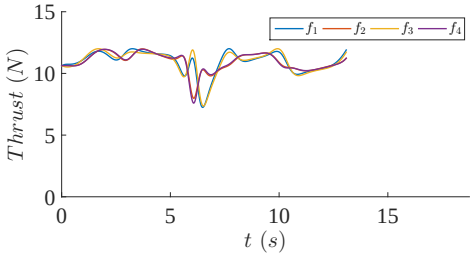
(c) Mapping function: Stage 1.



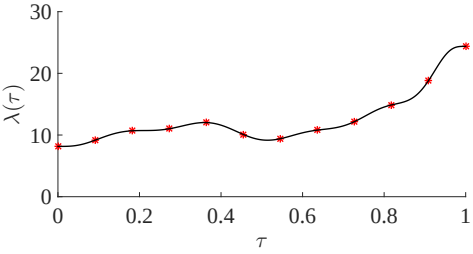
(d) Rotor thrusts: Stage 1.



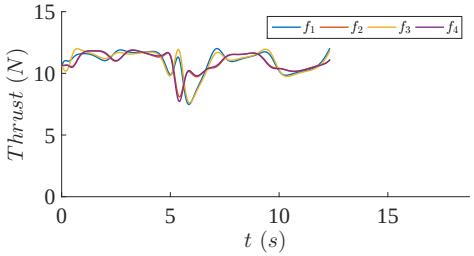
(e) Mapping function: Stage 2.



(f) Rotor thrusts: Stage 2.

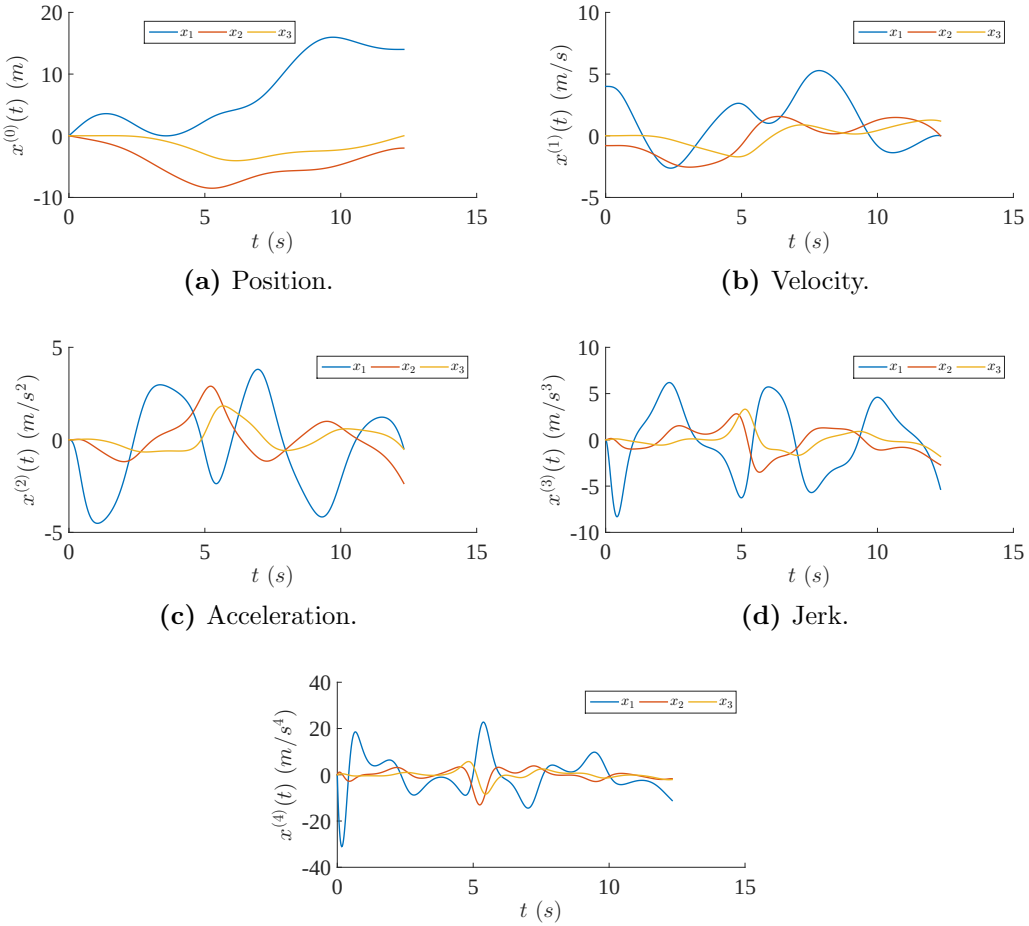


(g) Mapping function: Stage 3.



(h) Rotor thrusts: Stage 3.

Figure 6.8: Polynomial time mapping: example – The mapping stages for the polynomial time mapping example.



(a) Position.

(b) Velocity.

(c) Acceleration.

(d) Jerk.

(e) Jounce.

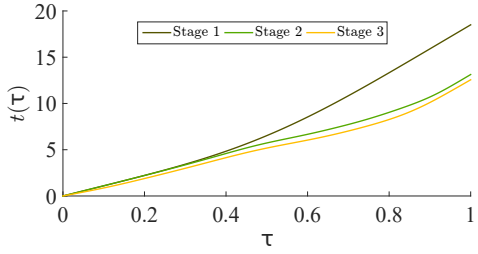
Figure 6.9: Polynomial time mapping: example states – The translational states up to the fourth derivative.

the mapping function to ensure the conditions on the derivatives of the mapping function (6.8) were satisfied and the boundary values could be specified exactly. The interior values control the shape of the curve but the mapping function is not required to pass through them exactly, as is the case for polynomials. This means that an interior mapping value could be less than zero, so long as the condition $\lambda(\tau) > 0$ was still met. Therefore, the upper and lower bounds on the optimisation vector should not be so restrictive as to prevent valid solutions.

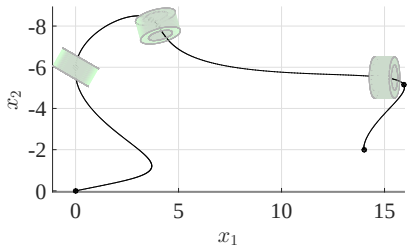
Like the previous methods, the process was split into three stages. The first stage finding the boundary values, the second finding the interior values and the third refining all values. In Chapter 5 it was necessary to find an initial guess for the B-spline coefficients when the heading angle was optimised using the least-squares approximation. This was not required for the mapping function because the shape of the curve after the boundary values were found was simple enough to simply discretise the solution to obtain the interior values that were used as guesses for the coefficients.

The same trajectory scenario described in Table 6.2 and Table 6.3 was used to test this method. Again, 12 unique mapping values were optimised (at each boundary the same value was repeated 5 times to ensure the boundary conditions were met. The same scalar weightings as the previous polynomial method were used: $j_T = 0.1$ and $j_f = 20$. Figure 6.10a shows the relationship between the time domain and the virtual domain for each stage. The trajectory times after Stage 1, Stage 2 and Stage 3 were respectively 18.75 s, 12.81 s and 11.61 s. In Figure 6.10b the trajectory after the final stage is shown.

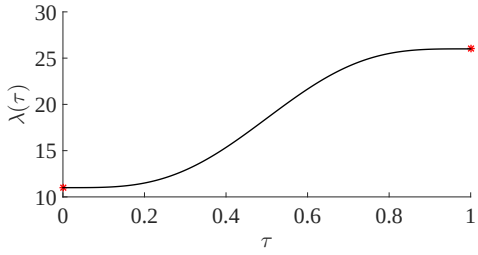
Mapping functions in the virtual domain and the rotor thrusts in the time domain for each stage are plotted in Figures 6.10c, 6.10e & 6.8g and 6.10d, 6.10f & 6.10h respectively. As mentioned above, the mapping function after the first stage in Figure 6.10c is simple enough for the B-spline to approximate with interpolation rather than the least-squares approximation method. The improvement in trajectory time between the second and final stage is minimal for this case. The translational states up to the fourth derivative in the time domain are given in Figure 6.11. All the boundary conditions were matched successfully.



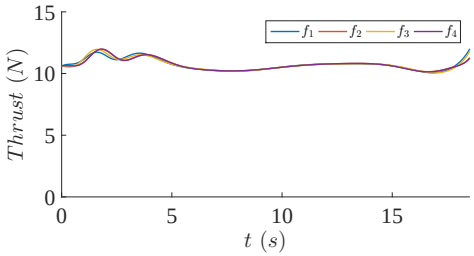
(a) Mapping function: Stage 1.



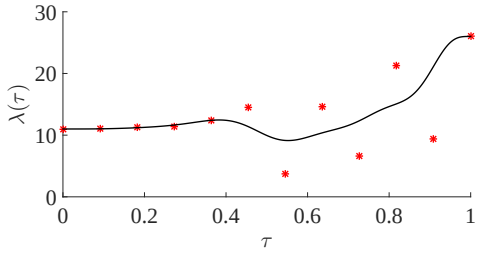
(b) Rotor thrusts: Stage 1.



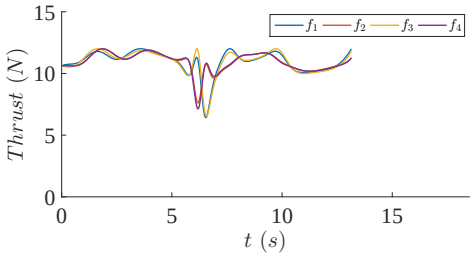
(c) Mapping function: Stage 1.



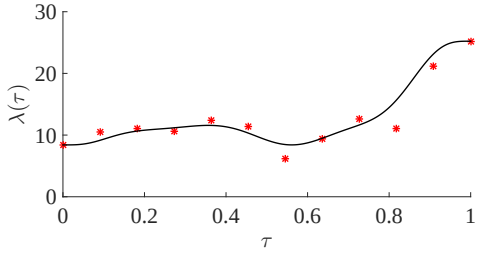
(d) Rotor thrusts: Stage 1.



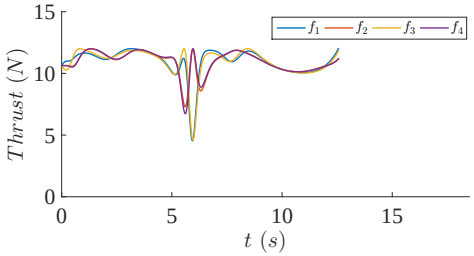
(e) Mapping function: Stage 2.



(f) Rotor thrusts: Stage 2.

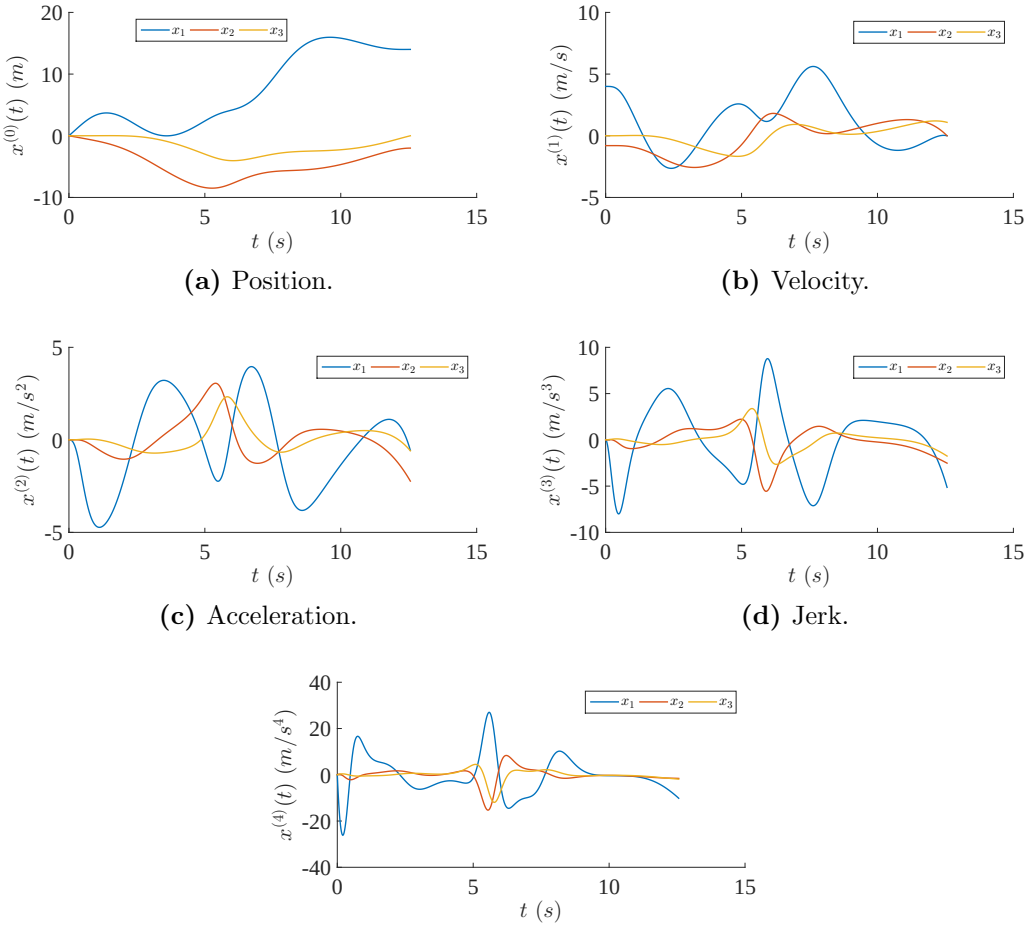


(g) Mapping function: Stage 3.



(h) Rotor thrusts: Stage 3.

Figure 6.10: B-spline time mapping: example – The mapping stages for the B-spline time mapping example.



(a) Position.

(b) Velocity.

(c) Acceleration.

(d) Jerk.

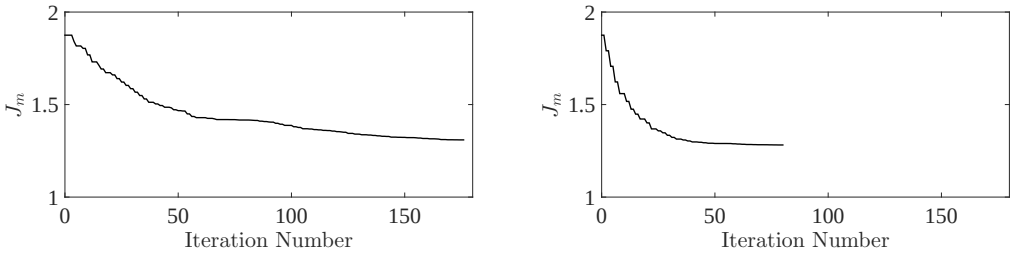
(e) Jounce.

Figure 6.11: B-spline time mapping: example states – The translational states up to the fourth derivative.

6.2.3.4 Algorithm comparison

Three methods for optimising the mapping values to minimise the trajectory time have been presented. The final trajectory times found by each method are similar but the B-spline approach is marginally the best. Currently the virtual domain trajectories require a numerical optimiser to find the shortest path lengths but if an alternative method was used and combined with the heuristic mapping approach then no numerical optimiser would be required. In some situations this could be beneficial, for instance if the flight software did not support or have one built in. However, the heuristic approach is just that, heuristic. The other two methods change the mapping values based on a cost function that includes the trajectory time. There could be cases when reducing an individual mapping value does not reduce the trajectory time so it is not possible to prove the strategy of the heuristic method is always the best one. For this reason and despite the simplicity of this method it was not chosen

B-splines are more complicated to calculate compared to polynomials but modern numerical software such as Matlab includes built-in dedicated functions to handle the mathematics. One significant benefit of B-splines is that there is no risk of ill-conditioning, as can happen when too many constraints are applied to polynomials. This means that the number of mapping values for polynomials are limited, whereas B-splines can add mapping values indefinitely and are only limited by convergence issues within the numerical optimiser. Additionally, power-series polynomials suffer from Runges phenomenon that can cause the function to oscillate near the boundaries [139]. In Figure 6.12a and Figure 6.12b the cost function against iteration number are plotted for both the polynomial and B-spline method respectively. It can be seen that the B-spline basis function converges to the minimum much more quickly than the polynomial basis function. This is most likely because B-splines have a local modification scheme so it is easier for the numerical optimiser to improve the mapping values without affecting the rest of the solution. For the reasons given, B-splines were chosen as the basis function for the mapping function and a further investigation is given in the next section.



(a) Polynomial basis function. **(b)** B-spline basis function.

Figure 6.12: Second numerical optimiser performance – A comparison between using the polynomial and B-spline as the basis function during the second stage of the mapping process.

6.2.4 Finding the best parameters for B-Spline algorithm

Of the three methods presented for the finding the mapping function, the B-spline parametrisation with the numerical optimisation was chosen as the preferred method. In this section it will be explored further to ascertain: the benefits of running the third stage and how many unique mapping values should be used. The third stage requires the virtual domain trajectory to be recalculated at each step because the boundary mapping values are not fixed. In the example scenario there was a marginal improvement of the trajectory time when the third stage was used but this was only a single test case. Intuitively, one would expect that increasing the number of mapping nodes would improve the solution. However, in practice, multi-parameter optimisation is challenging and by increasing the number of free variables it is possible that the minimiser will struggle or fail to converge.

In order to test both of these issues numerically a Monte Carlo simulation was created. This simulation is similar to the one performed in Chapter 5 that investigated the heading angle optimisation. However, in this example the boundary conditions on the translational derivatives were randomly chosen between given bounds. 50 trajectories were generated using the following process. Velocity and acceleration conditions were considered at the beginning and end of the trajectories. A 50% chance for each type of condition being applied was chosen so that wide variety of scenarios and types of boundary cases was generated. When a velocity or acceleration condition was applied to an axis it was constrained within

the limits:

$$-5\,\mathrm{m/s} \leq \dot{x}_{1,2,3} < 5\,\mathrm{m/s}, \quad -1\,\mathrm{m/s}^2 \leq \ddot{x}_{1,2,3} < 1\,\mathrm{m/s}^2 \tag{6.12}$$

The waypoints were randomly chosen with the same set of rules as Section 5.3.4. The trajectories that are feasible in the time domain after the boundary mapping values were found are given in Figure 6.13. The second stage of optimisation was then applied to each trajectory for 5, 10, 15, 20, 25 and 30 unique mapping values. To find the average improvement in trajectory time between the first and second stage the percentage decrease was found for each mapping value case during the first 100 iterations. The mean percentage decrease at every iteration was then calculated. Percentage decrease was chosen as the metric because short trajectories have less time available to improve so the longer trajectories would have a greater influence on the results if it was just the average trajectory time that was found.

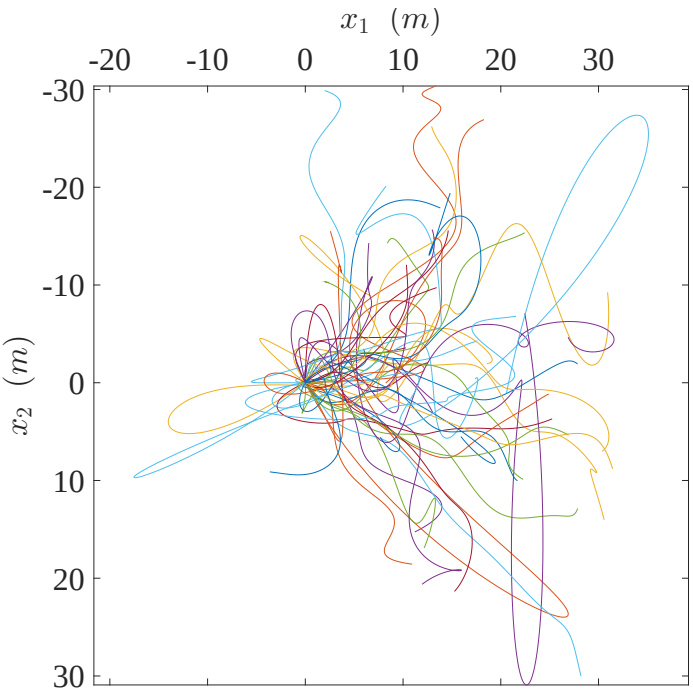


Figure 6.13: B-spline time mapping parameter selection: 50 trajectories – All the trajectories generated by the Monte Carlo process after the time mapping process has been applied.

The results after running the second optimisation stage are shown in Figure 6.14. It can be seen that the case of 5 mapping values converges the fastest but shows the smallest percentage decrease of all the tests. If there is only a very short optimisation period available then it is possible that 5 mapping values would be a reasonable choice, however the others have a similar trajectory time improvement rate during the early optimisations but then continue to get better later. At the other end of the spectrum, 30 mapping values is clearly too many because in 100 iterations it has only just converged to a result similar to when only 10 mapping values are used. The mid-range shows the best results and either 15 or 20 mapping values have a sufficient amount of nodes to find an good solution but not so many that the optimiser struggles to converge. One possibility to improve the method is to start with small number of mapping values and then add more to refine the solution. It may also be possible to find where the trajectory time has the most scope for improvement and focus the computational efforts there. Due to the local modification scheme of B-splines the rest of the solution would be unaffected by these changes. However, this is left as future work.

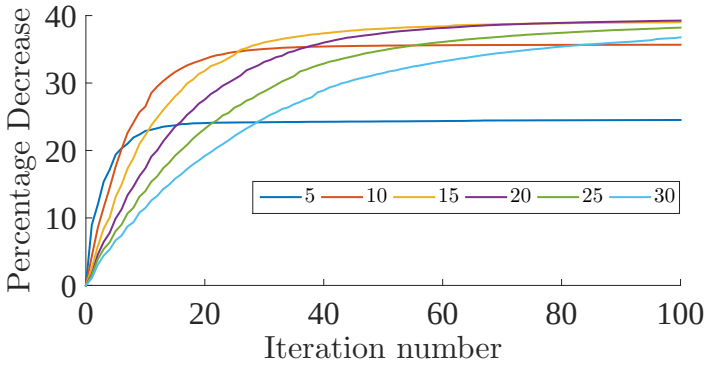


Figure 6.14: B-spline time mapping parameter selection Stage 2 performance – The percentage decrease for the various numbers of mapping nodes used for each iteration of the optimisation algorithm are given.

To test the improvement of the final mapping stage, 15 unique mapping nodes were used on the 50 trajectories previously generated. The same method for finding the average percentage decrease was used from Stage 2 to Stage 3. Figure 6.15 shows the percentage decrease against the iteration number when 15 nodes were used. It is clearly beneficial to apply this stage if possible. However, after about 40 iterations the trajectory time is no longer improved significantly.

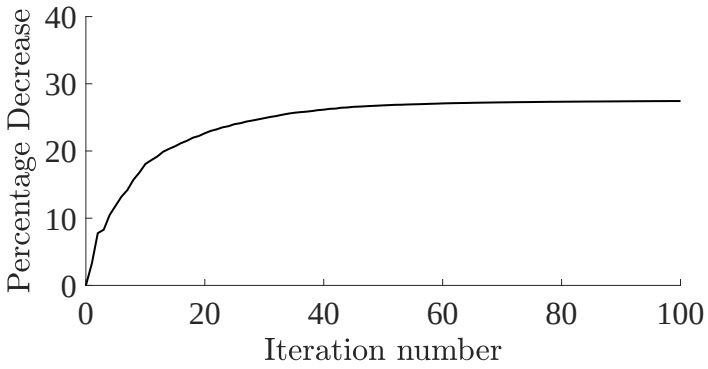


Figure 6.15: B-spline time mapping parameter selection Stage 3 performance – The performance of Stage 3 when 15 unique mapping nodes are used.

6.3 Chapter summary

In this chapter the methods for mapping a trajectory from the virtual domain into the time domain have been giving. The concept of a mapping function was defined and a simple, multi-objective cost function was defined that considered trajectory time and dynamic feasibility. An example showing the need to carefully choose the boundary mapping values was given and the algorithm for finding these was shown to find a good solution efficiently and was compared to an exhaustive search. Of the three methods demonstrated the B-spline optimisation was chosen as the best performer. An investigation into the number of mapping values to use suggested that 15 is a good amount. It was also shown that refining all the mapping values including the boundary ones was beneficial if it was possible.

This chapter concludes the trajectory generation methods. Dynamically feasible, collision avoiding and pseudo-optimal trajectories can now be generated for drone racing scenarios. The next chapter will use these motion planning methods as references for tracking controllers to follow. In Chapter 8 further simulations are performed with the trajectory generation methods in order to gain a better understanding of them.

Chapter 7

Disturbance rejection tracking controller

Chapters 5 and 6 developed a framework for generating drone racing trajectories. Once these trajectories have been obtained they can be used as reference trajectories for a tracking controller. The need for tracking controllers was introduced in Chapter 1 and they are required to calculate the control actions necessary to follow a planned motion. As explained in Chapter 2, quadrotors have four outputs (f_1, f_2, f_3, f_4) that consist of rotors that produce varying amounts of thrust depending on the motor speed. These thrusts were used for determining dynamic feasibility in the previous chapter. The dynamic equations of motion require the quadrotor outputs as a total scalar thrust f and moment about each body axis $\mathbf{M}$. This is also the format of the controls that the tracking controller uses but it easy to transform them into four thrusts if the layout configuration is known and the method for doing so will be given in this chapter.

Lee designed a nonlinear tracking controller for quadrotors on the SE(3) group [88]. This shall be referred to as the Standard Controller in this thesis and is the basis of the Modified Controller developed in this chapter. The Modified Controller uses a Linear Extended State Observer (LESO) which has been shown to be effective on nonlinear systems [140] to estimate the translational disturbance $\mathbf{D}$ acting upon the vehicle. LESOs have been applied to quadrotors previously [99, 100, 101] but in all these cases the tracking controllers were based on Euler angles which suffer from the limitations discussed in the first chapter. A nonlinear extended state observer is another kind of Active Disturbance Rejection

Control (ADRC) but it requires more parameters to tune and also requires more calculations. If the estimator struggles to track the disturbance then a nonlinear ESO should be considered. When a disturbance estimator is used it is important to demonstrate that the estimator error remains bounded and converges within a known region [141].

Original Contributions

The original contributions in this chapter are outlined as follows:

- The conversion of the total thrust and moment about each axis into four individual thrusts for the Cross and H-Shape layout configuration is developed. The conversion for the Standard layout configuration is also given but this has been found previously.
- The Standard Controller is used to develop and inverse dynamics model that can be used to find the thrust required of each rotor given a time based reference trajectory and assuming there are no disturbances.
- The Standard SE(3) Controller is modified to include a Linear Extended State Observer that improved the tracking performance when disturbances are present.
- A convergence proof for the LESO is given to show the bounds on the estimation error of the disturbance.
- Three types of disturbances were used to test and compare the tracking controllers: i) square wave, ii) sinusoidal, iii) wind model.

The chapter is structured as follows. In Section 7.1 the Standard Controller is introduced. It is then used as the basis for an inverse dynamics formulation that allows the individual rotor thrusts to be calculated for a variety of layout configurations. Section 7.2 develops a LESO that is used to modify the Standard Controller to include ADRC. The two tracking controllers are compared numerically in simulation with three types of disturbances in Section 7.3. The chapter is concluded and summarised in Section 7.4.

7.1 Standard controller and inverse dynamics

This section introduces the Standard SE(3) tracking controller and uses it to form an inverse dynamics method for find the control thrusts required to follow a quadrotor trajectory.

7.1.1 Standard SE(3) controller

The nonlinear tracking controller developed on the special Euclidean group SE(3) presented in [88] is used as the basis for the disturbance rejection controller in the next section. The scalar magnitude of thrust is calculated by the controller by trying to minimises the tracking errors in both the translational and rotational position:

$$f = -(-k_x \mathbf{e_x} - k_v \mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d}) \cdot R_d \mathbf{e_3} \tag{7.1}$$

where $\mathbf{e_3}$ is the direction of the third body axis. Similarly for the moment control:

$$\begin{aligned} \mathbf{M} = &-k_R \mathbf{e_R} - k_\Omega \mathbf{e_\Omega} + \mathbf{\Omega} \times J\mathbf{\Omega} \\ &-J(\hat{\Omega} R^T R_d \mathbf{\Omega_d} - R^T R_d \dot{\mathbf{\Omega}}_\mathbf{d}) \end{aligned} \tag{7.2}$$

where R is the actual rotation and R_d is the desired rotation and where $\mathbf{e_x}$, $\mathbf{e_v}$, $\mathbf{e_R}$ and $\mathbf{e_\Omega}$ are the tracking error vectors in position, velocity, attitude and angular velocity respectively and are calculated respectively:

$$\mathbf{e_x} = \mathbf{x} - \mathbf{x_d} \tag{7.3}$$

$$\mathbf{e_v} = \mathbf{v} - \mathbf{v_d} \tag{7.4}$$

$$\mathbf{e_R} = \frac{1}{2}\big(R_d^T R - R^T R_d\big)^\vee \tag{7.5}$$

$$\mathbf{e_\Omega} = \mathbf{\Omega} - R^T R_d \mathbf{\Omega_d} \tag{7.6}$$

Their associated scalar gains are k_x, k_v, k_R and k_Ω. The desired states are denoted with subscript d and $\hat{\Omega}$ is the skew-symmetric matrix of the angular velocity:

$$\hat{\Omega} = \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \tag{7.7}$$

The desired rotation can be expressed as $R_d = [\mathbf{b_{1_d}}, \mathbf{b_{3_d}} \times \mathbf{b_{1_d}}, \mathbf{b_{3_d}}]$ where the desired direction of the third body axis is:

$$\mathbf{b_{3_d}} = \frac{-k_x\mathbf{e_x} - k_v\mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d}}{\|-k_x\mathbf{e_x} - k_v\mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d}\|} \tag{7.8}$$

7.1.2 Inverse dynamics

If it is assumed that the error vectors of the position, velocity, orientation and angular velocity are zero so no control gains are required. The controls can then be calculated analytically from the translational motion and the direction of the first body-axis without needing to numerically integrate the equations of motion (2.3)-(2.6).

The translational dynamics are controlled by the net thrust f directed along the $\mathbf{b_3}$ body axis. The direction of this axis is dependent on the translational acceleration desired, the actual translational acceleration and the gravitational acceleration in the inertial frame:

$$\mathbf{b_{3_c}} = \frac{-mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d}}{\|-mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d}\|} \tag{7.9}$$

where subscript d is the desired value and subscript c is the commanded value. After choosing the desired direction of the $\mathbf{b_1}$ either by fixing it at a constant or using the heading angle optimisation developed previously, the second body axis $\mathbf{b_2}$ can be computed:

$$\mathbf{b_{2_c}} = \frac{\mathbf{b_{3_c}} \times \mathbf{b_{1_d}}}{\|\mathbf{b_{3_c}} \times \mathbf{b_{1_d}}\|} \tag{7.10}$$

The commanded direction along the first body axis is:

$$\mathbf{b_{1_c}} = \mathbf{b_{2_c}} \times \mathbf{b_{3_c}} \tag{7.11}$$

The desired attitude as a rotation matrix:

$$R_d = [\mathbf{b_{1_c}}, \mathbf{b_{2_c}}, \mathbf{b_{3_c}}]^T \tag{7.12}$$

And the desired angular body velocity:

$$\hat{\Omega}_d = R_d^T \dot{R}_d \tag{7.13}$$

Since the first derivative of the desired rotation matrix is required it can be seen that $\hat{\Omega}_d$ is dependent upon the translational acceleration and jerk. Using the chain rule, the rate of change of $\hat{\Omega}_d$ can be found:

$$\dot{\hat{\Omega}}_d = R_d^T \ddot{R}_d + \dot{R}_d \dot{R}_d^T \tag{7.14}$$

which is a function of translational acceleration, jerk and jounce. The attitude is controlled by applying a suitable moment about each axis and is computed by:

$$\mathbf{M} = \mathbf{\Omega_d} \times J\mathbf{\Omega_d} - J(\hat{\Omega}_d \mathbf{\Omega_d} - \dot{\mathbf{\Omega}}_\mathbf{d}) \tag{7.15}$$

The net thrust that controls the translational dynamics can be expressed as:

$$f = (mg\mathbf{e_3} - m\ddot{\mathbf{x}}_\mathbf{d}) \cdot R_d \mathbf{e_3} \tag{7.16}$$

If the controls of trajectory segments are to be smoothly joined then the states $\mathbf{x}(t)$ must be considered to the fourth derivative. In this way continuity of f and $\mathbf{M}$ can be ensured during the transfer between segments.

7.1.3 Calculating rotor thrusts

For a given set of controls, f and M, the required thrust from each rotor can be calculated by the relation:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = L^{-1} \begin{bmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} \tag{7.17}$$

where $L \in \mathbb{R}^{4\times4}$ is a matrix that relates the controls to the thrust for each rotor. The composition of L is dependent on the layout configuration. In order for L to be invertible the determinant of L must be non-zero. For the Standard layout (Figure 2.5a) the following simplifications can be made:

$$\begin{aligned} d_{2,2} = d_{4,2} = d_{1,1} = d_{3,1} = d, \\ d_{1,2} = d_{3,2} = d_{2,1} = d_{4,1} = 0 \end{aligned} \tag{7.18}$$

so the layout configuration is:

$$L_s = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\kappa f} & c_{\kappa f} & -c_{\kappa f} & c_{\kappa f} \end{bmatrix} \tag{7.19}$$

where d is the distance between the centre of mass and the rotor and $c_{\kappa f}$ is a constant related to the torque produced about the yaw axis of the quadrotor caused by the rotors angular motion. The determinant of L_s is $8c_{\kappa f}d^2$ so the matrix is invertible when $c_{\kappa f} \neq 0$ and $d \neq 0$.

For the Cross layout (Figure 2.5b) the following simplifications can be made:

$$\begin{aligned} d_{1,2} = d_{2,2} = d_{3,2} = d_{4,2} = d, \\ d_{2,1} = d_{2,1} = d_{3,1} = d_{4,1} = d \end{aligned} \tag{7.20}$$

so the layout configuration is:

$$L_c = \begin{bmatrix} 1 & 1 & 1 & 1 \\ d & -d & -d & d \\ d & d & -d & -d \\ -c_{\kappa f} & c_{\kappa f} & -c_{\kappa f} & c_{\kappa f} \end{bmatrix} \tag{7.21}$$

and the determinant is $16c_{\kappa f}d^2$, so again, the matrix is invertible when $c_{\kappa f} \neq 0$ and $d \neq 0$.

Finally, for the H-shape (Figure 2.5c) the following simplifications can be made:

$$\begin{aligned} d_{1,2} = d_{2,2} = d_{3,2} = d_{4,2} = d_2, \\ d_{2,1} = d_{2,1} = d_{3,1} = d_{4,1} = d_1 \end{aligned} \tag{7.22}$$

so the layout configuration is:

$$L_h = \begin{bmatrix} 1 & 1 & 1 & 1 \\ d_2 & -d_2 & -d_2 & d_2 \\ d_1 & d_1 & -d_1 & -d_1 \\ -c_{\kappa f} & c_{\kappa f} & -c_{\kappa f} & c_{\kappa f} \end{bmatrix} \tag{7.23}$$

where the determinant is $16c_{\kappa f}d_1d_2$. Therefore the L_h is invertible when $c_{\kappa f} \neq 0$,

$d_1 \neq 0$ and $d_2 \neq 0$.

7.2 Modified controller

The scalar thrust calculated by the modified tracking controller is:

$$f = -\left(-k_x \mathbf{e_x} - k_v \mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d} - \hat{\mathbf{D}}\right) \cdot R_d \mathbf{e_3} \tag{7.24}$$

and the control moment is:

$$\begin{aligned} \mathbf{M} = &-k_R \mathbf{e_R} - k_\Omega \mathbf{e_\Omega} + \boldsymbol{\Omega} \times \mathbf{J}\boldsymbol{\Omega} \\ &-J(\hat{\Omega} R^T R_d \boldsymbol{\Omega}_\mathbf{d} - R^T R_d \dot{\boldsymbol{\Omega}}_\mathbf{d}) \end{aligned} \tag{7.25}$$

and desired direction of the third body axis is:

$$\mathbf{b_{3_c}} = -\frac{-k_x \mathbf{e_x} - k_v \mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d} - \hat{\mathbf{D}}}{\left\| -k_x \mathbf{e_x} - k_v \mathbf{e_v} - mg\mathbf{e_3} + m\ddot{\mathbf{x}}_\mathbf{d} - \hat{\mathbf{D}} \right\|} \tag{7.26}$$

It can be seen that the Modified Controller is very similar to the Standard Controller, the only difference being that the estimated disturbance $\hat{\mathbf{D}}$ is included in the control thrust f and the commanded direction of the third body axis $\mathbf{b_{3_c}}$. The method for calculating the estimated disturbance is given in the next section. The moment control for the Modified Controller (7.25) is identical to the moment control for the Standard Controller (7.2) because in this thesis rotational disturbances were not considered and were left as future work.

The process of deriving the LESO to estimate the disturbance and show the estimation error is bounded is as follows. After rearranging (2.4) the disturbance term as an acceleration is $\gamma = \frac{\mathbf{D}}{m}$. Using a linear second-order extended state observer [140] the estimated disturbance can be found:

$$\dot{\hat{\mathbf{v}}} = \hat{\gamma} + \beta_1(\mathbf{v} - \hat{\mathbf{v}}) - \frac{fR\mathbf{e_3}}{m} + \frac{mg\mathbf{e_3}}{m} \tag{7.27}$$

$$\dot{\hat{\gamma}} = \beta_2(\mathbf{v} - \hat{\mathbf{v}}) \tag{7.28}$$

where the estimated velocity and disturbance are $\hat{\mathbf{v}}$ and $\hat{\gamma}$ respectively. The LESO scalar gains β_1 and β_2 must satisfy the constraint ${\beta_1}^2 > 4\beta_2$. To ensure

this they were defined in terms of scalar bandwidth $\omega_c > 0$ such that $\beta_1 = 4\omega_c$ and $\beta_2 = 3\omega_c$.

The estimation errors for the velocity $\widetilde{\mathbf{v}} = \mathbf{v} - \hat{\mathbf{v}}$ and disturbance $\widetilde{\gamma} = \gamma - \hat{\gamma}$ are differentiated with respect to time to find the estimation error dynamics:

$$\dot{\widetilde{\mathbf{v}}} = \dot{\mathbf{v}} - \dot{\hat{\mathbf{v}}} = -\beta_1 \widetilde{\mathbf{v}} + \widetilde{\gamma} = -4\omega_c + \widetilde{\gamma} \tag{7.29}$$

$$\dot{\widetilde{\gamma}} = \dot{\gamma} - \dot{\hat{\gamma}} = -\beta_2 \widetilde{\mathbf{v}} + \dot{\gamma} = -3\omega_c^2 \widetilde{\mathbf{v}} + \dot{\gamma} \tag{7.30}$$

Equations (7.29) and (7.30) can be rewritten by setting $\epsilon = [\epsilon_{\mathbf{1}},\ \epsilon_{\mathbf{2}}]^T$ where $\epsilon_{\mathbf{1}} = \widetilde{\mathbf{v}}$ and $\epsilon_{\mathbf{2}} = \frac{\widetilde{\gamma}}{\omega_c}$:

$$\dot{\epsilon}_{\mathbf{1}} = -4\omega_c \epsilon_{\mathbf{1}} + \omega_c \epsilon_{\mathbf{2}} \tag{7.31}$$

$$\dot{\epsilon}_{\mathbf{2}} = -3\omega_c^2 \epsilon_{\mathbf{1}} + \frac{\dot{\gamma}}{\omega_c} \tag{7.32}$$

which can be combined using matrices to form a single equation:

$$\dot{\epsilon} = \omega_c A \epsilon + B \frac{\dot{\gamma}}{\omega_c} \tag{7.33}$$

where $\epsilon_{\mathbf{1}} = [\epsilon_{11},\ \epsilon_{12},\ \epsilon_{13},]^T$, $\epsilon_{\mathbf{2}} = [\epsilon_{21},\ \epsilon_{22},\ \epsilon_{23},]^T$, $A = \begin{bmatrix} -4I_{3\times3} & I_{3\times3} \\ -3I_{3\times3} & 0_{3\times3} \end{bmatrix}$, $B = \begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix}$. New estimation error dynamic variables are defined by taking elements from $\epsilon_{\mathbf{1}}$ and $\epsilon_{\mathbf{2}}$ such that $\widetilde{\epsilon}_{\mathbf{i}} = [\epsilon_{1i},\ \epsilon_{2i}]^T$ for $i = 1, 2, 3$ so (7.33) becomes:

$$\dot{\widetilde{\epsilon}}_{\mathbf{i}} = \widetilde{A}\widetilde{\epsilon}_{\mathbf{i}} + \widetilde{B}\frac{\dot{\gamma}_{\mathbf{i}}}{\omega_c} \tag{7.34}$$

where $\widetilde{A} = \begin{bmatrix} -4\omega_c & \omega_c \\ -3\omega_c & 0 \end{bmatrix}$ and $\widetilde{\mathbf{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. The general solution of a differential equation in the form of (7.34) can be expressed as:

$$\widetilde{\epsilon}_{\mathbf{i}}(t) = e^{\widetilde{A}t}\widetilde{\epsilon}_{\mathbf{i}}(0) + \int_0^t e^{\widetilde{A}(t-\tau)} \frac{\widetilde{\mathbf{B}}\dot{\gamma}}{\omega_c} d\tau \tag{7.35}$$

From the Cayley-Hamiltonian theorem $e^{\widetilde{A}t}$ can be written as:

$$e^{\widetilde{A}t} = \frac{1}{2} e^{-3\omega_c t} \begin{bmatrix} 3 - e^{2\omega_c t} & -1 + e^{2\omega_c t} \\ 3(1 - e^{2\omega_c t}) & -1 + 3e^{2\omega_c t} \end{bmatrix} \tag{7.36}$$

therefore, $\lim_{t\to\infty}$ then $e^{\widetilde{A}t} = 0$. Assuming $||\dot{\gamma}|| \leq \delta$ and as $t \to \infty$ the limit of the magnitude of the estimation error dynamics can be reduced to:

$$\begin{aligned} \|\widetilde{\epsilon}_i(t)\| &\leq \frac{\delta}{\omega_c} \left\| -\widetilde{A}^{-1} \right\|_F \\ &\leq \frac{\delta\sqrt{26}}{3\omega_c^2} \end{aligned} \tag{7.37}$$

In order to derive a bound on the estimation error of the disturbance it is necessary to remember $\epsilon_{\mathbf{2}} = \frac{\widetilde{\gamma}}{\omega_c}$. From the inequality $\|\widetilde{\epsilon}_{\mathbf{i}}(t)\| \leq \frac{\delta\sqrt{26}}{3\omega_c^2}$ it must be true that $|\widetilde{\gamma}_{\mathbf{i}}| \leq \frac{\delta\sqrt{26}}{3\omega_c}$ and therefore the Euclidean norm of $\widetilde{\gamma}$ is given as:

$$\|\widetilde{\gamma}\| = \sqrt{\widetilde{\gamma}_1^2 + \widetilde{\gamma}_2^2 + \widetilde{\gamma}_3^2} \leq \frac{\delta\sqrt{78}}{3\omega_c} \tag{7.38}$$

Therefore, as $t \to \infty$ the estimation error of the disturbance is bounded by

$$\|\widetilde{\gamma}\| \leq \sigma_2 \tag{7.39}$$

where $\sigma_2 = \frac{\delta\sqrt{78}}{3\omega_c}$ is a constant.

7.3 Disturbances

Three types of disturbance were used to compare the tracking performance of the Modified Controller and the Standard Controller. The reference trajectory for all the simulations in this section was the final mapping stage using the B-spline as the basis function for the mapping function from Section 6.2.3.3, the translational states being given in Figure 6.11. The error gains for both types of controllers were the same and chosen as follows: $k_x = 10m$, $k_v = 5.6m$, $k_R = 15$, $k_\Omega = 2.54$. These values were found from experimentation and hand-tuning. For the Modified Controller the estimator gains $\beta_1 = 15$ and $\beta_2 = 50$ were chosen, which satisfy the condition $\beta_1^2 > 4\beta_2$.

7.3.1 Square wave

The square wave is a discontinuous function that is particularly challenging for the observer to track due to the step changes. The convergence proof presented in

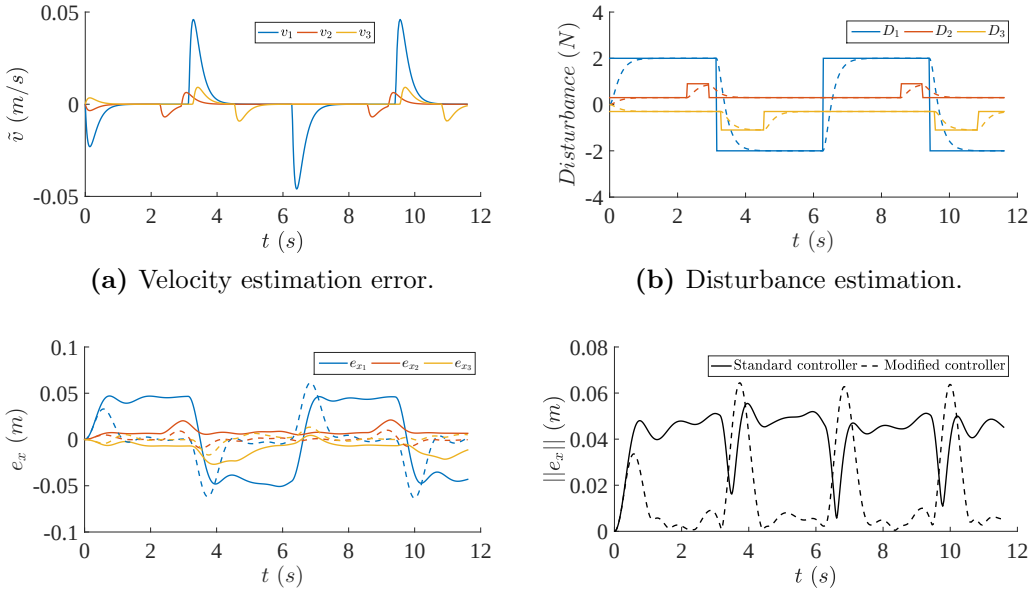
the previous section assumes the disturbance is continuous but for the purposes of testing the LESO under difficult conditions the square wave is of interest in order to see how the controllers behave. Different square wave functions were applied to each translational axis in the inertial frame independently. On the first axis $\mathbf{e_1}$ a square wave with amplitude 2 N, period 6 s and zero phase offset is applied. The square wave disturbance on the second axis $\mathbf{e_2}$ has an amplitude 0.6 N, period 0.7 s and a phase offset of 2.2 s. The third axis $\mathbf{e_3}$ disturbance square wave has an amplitude 0.8 N, period of 1.259 s and phase offset of 3.3 s.

The solid lines in Figure 7.1b plot the disturbance against time. The dashed lines represent the estimated disturbance $\hat{\mathbf{D}}$ as calculated by the Modified Controller. The estimated values track the actual values reasonably well. The step changes cause periods in which the estimated values must catch up but this happens quickly and there is no overshoot so the estimator gains are tuned correctly. Figure 7.1a shows the estimation error of the velocity $\hat{\mathbf{v}}$ is small throughout the trajectory. The translational position error for each axis is shown for the Standard Controller (solid lines) and the Modified Controller (dashed lines) in Figure 7.1c. The largest errors occur on the first axis due to the magnitude of the disturbance being greatest there. On the Standard Controller the errors remain steady when a disturbance is present. This is in contrast to the Modified Controller that converges back to the reference trajectory once the disturbance estimator has regained the correct value. The magnitude of the position error is given in Figure 7.1d. The only times that the Modified Controller (dashed line) shows inferior tracking performance compared to the Standard Controller (solid line) is just after the step changes. This suggests that if the disturbance is highly discontinuous then an extended state observer may not be suitable because the disturbance estimator will be unable to track the actual disturbance.

7.3.2 Sinusoidal

Unlike the previous disturbance, the sinusoidal disturbances are continuous and defined for each axis by the following:

$$\mathbf{D} = \begin{bmatrix} 2\ \sin(0.4330t) \\ 1\ \sin(0.2165t) \\ 0.5\ \sin(1.7321t) \end{bmatrix} \text{N} \tag{7.40}$$



(a) Velocity estimation error.

(b) Disturbance estimation.

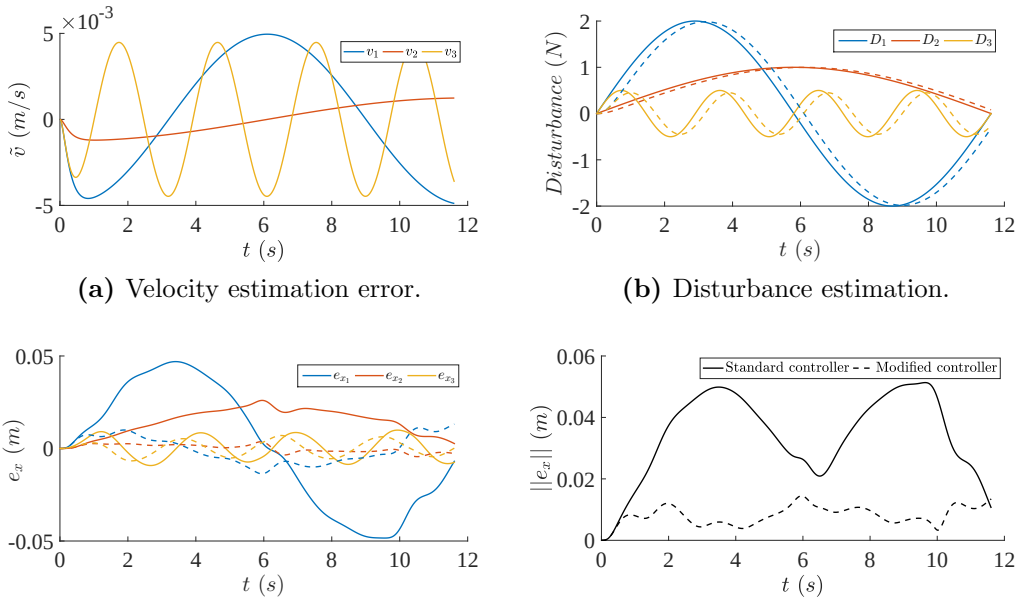
(c) Translational position tracking error.

(d) Magnitude of position error.

Figure 7.1: Square wave disturbance tracking controller performance – The Modified Controller demonstrates better tracking performance compared to the Standard Controller under the influence of a square wave disturbance.

The choices for the amplitude, and frequency for each of the three axes was arbitrary but a range of values were chosen to enable a comparison.

The performance of the tracking controllers under sinusoidal disturbances is given in Figure 7.2. The actual (solid lines) and estimated (dashed lines) disturbances are plotted in Figure 7.2b. It can be seen that the disturbance with the smallest period on the second axis is tracked best by the estimator. This is to be expected because as discussed previously, slow changes are easier to track. However, even the relatively high frequency disturbance on the third axis is tracked reasonably well, showing only a small offset. The velocity estimation error in Figure 7.2a is small throughout the trajectory. The position tracking error $\mathbf{e_x}$ is smaller on all axes for the Modified Controller compared to the Standard Controller. This is also demonstrated by the magnitude of the tracking error in Figure 7.2d.



(a) Velocity estimation error.

(b) Disturbance estimation.

(c) Translational position tracking error.

(d) Magnitude of position error.

Figure 7.2: Sinusoidal disturbance tracking controller performance – The Modified Controller demonstrates better tracking performance compared to the Standard Controller under the influence of a sinusoidal disturbance.

7.3.3 Wind

The final disturbance used to test the tracking controllers is from a wind model and is continuous but non-cyclical. A spatially correlated turbulent wind simulation in Figure 7.3 was produced using Cheynet's open source code [142] that is based on the models in [143, 144]. The disturbance force caused by the wind for each axis is calculated by [145]:

$$D_a = -\frac{1}{2}\rho_{air} C_{d,a} A_a (v_a - v_{w,a})^2 \operatorname{sgn}(v_a - v_{w,a}) \tag{7.41}$$

where $v_{w,a}$ is the wind speed for the given axis, $\rho_{air} = 1.225\,\mathrm{kg/m^3}$ is the air density, $C_{d,a}$ is the drag coefficient, A_a is the cross sectional area for the given axis denoted by subscript a. In this simulation the drag coefficient and cross sectional area were assumed to be the same for all axes and were $C_{d,1} = C_{d,2} = C_{d,3} = 0.5$ and $A_1 = A_2 = A_3 = 0.202\mathrm{m}^2$ respectively. The **sgn** function returns 1 if the input is greater than 0, -1 if the input is less than 0, and 0 if the input is equal to zero.

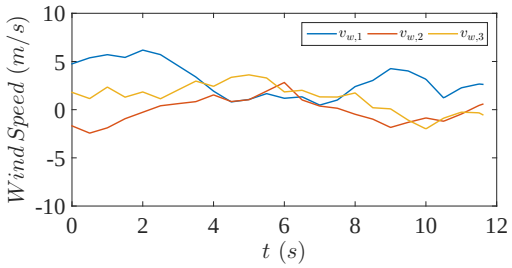


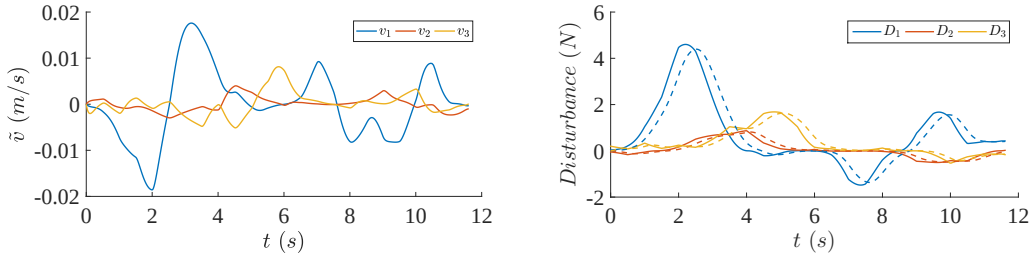
Figure 7.3: Wind speed simulation – The wind velocities generated in simulation and used as the disturbance.

The performance of the Standard Controller and the Modified Controller under wind disturbances is given in Figure 7.4. As was the case for previous disturbances, the velocity estimation error in Figure 7.4a is small throughout the trajectory. The performance of the disturbance estimator in Figure 7.4b shows that it is capable of tracking the wind disturbance and there is only a small lag between the actual disturbance and the estimated value. The tracking error in the translational position shown in Figure 7.4c is greatest on the first axis, corresponding to largest disturbance force. The magnitude of the translational position error is given in Figure 7.4d. The Modified Controller (solid line) has a maximum magnitude of error that is approximately half of the Standard Controller (dashed line).

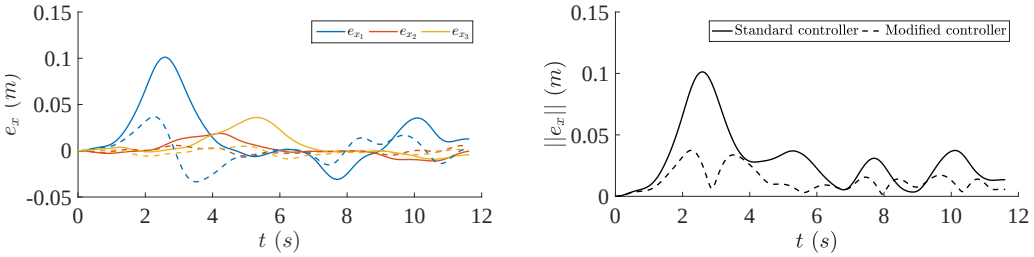
7.4 Chapter summary

In this chapter the trajectory tracking problem was addressed. A Standard Controller based on SE(3) was used to develop an inverse dynamics formulation for calculating the thrust and moment required to follow a trajectory. The method for calculating the required thrusts for each rotor was given for three layout configurations: Standard, Cross and H-Shape. This inverse dynamics formulation was used to ensure the feasibility of trajectories during the trajectory generation in earlier chapters.

A LESO was added to the Standard Control in order to provide better translational disturbance rejection. A convergence proof was given for the LESO that gave the bound of the error on the disturbance estimate. The Modified Controller was compared to the Standard Controller with numerical simulations under



(a) Velocity estimation error. **(b)** Disturbance estimation.



(c) Translational position tracking error. **(d)** Magnitude of position error.

Figure 7.4: Wind disturbance tracking controller performance – The Modified Controller demonstrates better tracking performance compared to the Standard Controller under the influence of a wind disturbance.

three types of disturbances using a reference trajectory from Chapter 6. The square wave disturbance showed the worst disturbance estimation due to the discontinuous nature of the signal. However, the tracking error performance in the translation position for the Modified Controller was still superior than that of the Standard Controller. The sinusoidal and wind disturbance also showed the Modified Controller performed better than the Standard Controller. Rotational disturbances were not considered for this controller, however, the same methods used for the translational disturbances could be applied. Future work can address this and the controllers should also be tested on a higher fidelity simulation and onboard an actual quadrotor.

Chapter 8

Simulations and further testing

In Chapters 3, 5 and 6 the motion planning methods for quadrotors were developed. Examples were given to illustrate the methods, however, these test cases did not explore the wide variety of scenarios. For instance, the Standard quadrotor layout was used throughout for consistency but two other layouts were also given in Chapter 2 and inverse dynamics found in Chapter 7. Also, example trajectories were given for the heading optimisation method but a comparison between the trajectory time of fixed heading angles was not possible because the time mapping method had not yet been introduced. Finally the dynamic feasibility of trajectories in this thesis is considered by limiting the maximum thrust of each rotor but it is possible to add other kinodynamic constraints such as a maximum velocity magnitude.

The quadrotor simulators [113, 114] mentioned previously show the vehicle in a 3D environment but are not suitable for visualising the trajectories generated in this thesis. This is because the vehicle states that have been calculated cannot be used as inputs. The Quadrotor Trajectory Analyser (QTA) introduced in this chapter seeks to address this by providing a simple method for generating an animated three dimensional trajectory whilst simultaneously showing the graphs of any vehicle states that are of interest.

Original Contributions

The original contributions in this chapter are outlined as follows:

- The final trajectory times with different quadrotor configurations are found for a variety of heading angle scenarios.

- The relationship between the final trajectory time and size of the bounds on the optimised heading angle is investigated.
- Trajectory time mapping with additional feasibility constraints on velocity and acceleration is demonstrated.
- The Quadrotor Trajectory Analyser that visualises quadrotor trajectories using a 3D animation is presented.

The chapter is structured as follows. In Section 8.1 three layout configurations are compared under different heading angle conditions. Section 8.2 investigates the relationship between the trajectory time and the size of the heading angle constraints when the heading angle is free to be optimised. Additional kinodynamic feasibility constraints are added to the time mapping method in Section 8.3. The Quadrotor Trajectory Analyser is presented in Section 8.4. Finally, a summary of the chapter is given in Section 8.5.

8.1 Comparison of quadrotor configurations

During the previous chapters the Standard layout has been used as the default configuration for the simulations. However, two other configurations, Cross and H-Shape were also given. In this section the layouts are compared with each other by defining a drone racing course in which the vehicles start at rest $\mathbf{x}^{(0)}(0) = \mathbf{x}^{(1)}(0) = \mathbf{x}^{(2)}(0) = \mathbf{x}^{(3)}(0) = \mathbf{x}^{(4)}(0) = 0$ and pass through five gates defined in Table 8.1. All the gates were the same size and had the following parameters: $r_{h,i} = 0.35\,\text{m}$, $r_{h,o} = 1\,\text{m}$ and $l_h = 0.5\,\text{m}$.

The trajectories start and finish at $x(0) = [0\,\text{m}, 0\,\text{m}, 0\,\text{m}]^T$ and $x(T) = [0\,\text{m}, 2\,\text{m}, -1\,\text{m}]^T$ respectively. The vehicle properties for each layout are given in Chapter 2.

8.1.1 Fixed heading angle

A minimum time, dynamically feasible trajectory, that satisfies the boundary conditions and passes through the gates is found for each configuration when the desired first body angle is fixed at $\mathbf{b_{1d}} = [1, 0, 0]^T$. The trajectory results for the Standard layout, Cross Layout and H-Shape are given in Figures 8.1, 8.2 and 8.3

Gate	h_1 (m)	h_2 (m)	h_3 (m)	$\phi_{h,1}$ (°)	$\phi_{h,2}$ (°)	$\phi_{h,3}$ (°)
1	6	0	−1	0	0	0
2	15	3	−2	0	10	−50
3	22	1	−2	0	30	90
4	20	8	−6	0	0	110
5	8	8	−2	0	0	220

Table 8.1: Layout comparison: gate parameters – The parameters for each of the 5 gates used to define a 3D drone racing course.

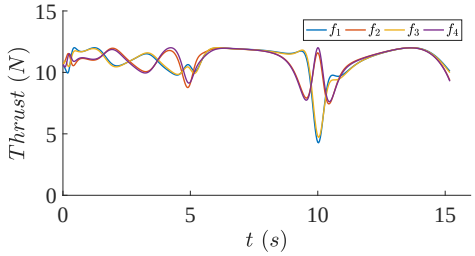
respectively. In the same order, the final trajectory times after mapping were 15.18 s, 15.00 s, 15.08 s respectively. The small improvement for the Cross and H-Shape layout compared to the Standard is likely because the first axis in the inertial frame $\mathbf{e_1}$ has the greatest translational motion. Initially, when at rest, the Standard layout has one rotor that can produce a thrust to tilt the quadrotor in the desired direction compared to the other layouts which have two. However, the figures demonstrate that the trajectories are very similar when a fixed heading angle is desired. All layouts maintain high thrust from all rotors for much of the trajectory, showing that the dynamic capabilities of the vehicle are being utilised.

8.1.2 Exact velocity heading direction

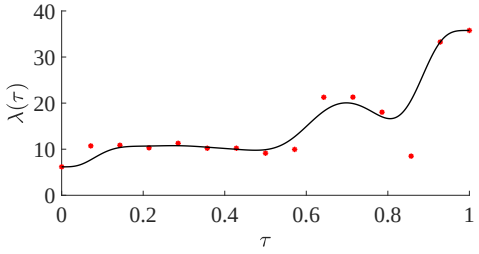
For this scenario the heading angle was always in the direction of the planar velocity θ_v, as calculated by (5.17). The trajectory results for the Standard layout, Cross layout and H-Shape are given in Figures 8.4, 8.5 and 8.6 respectively. In the same order, the final trajectory times after mapping were 24.03 s, 24.07 s, 24.55 s. These trajectory times and the trajectory states were similar for each layout. The thrust at the early stage of the trajectory spikes to the maximum bound but soon drops for the first five seconds. This is due to the exact velocity angle lacking sufficient smoothness, so the initial mapping function value is relatively large.

8.1.3 Optimised heading angle

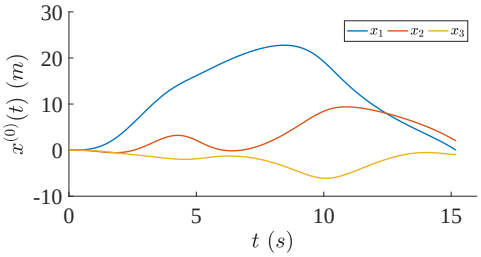
The final comparison between the quadrotor configurations is the case when the heading angle is optimised with a tolerance angle $\theta_{tol} = 15°$. The initial derivatives of the optimised angle were fixed at zero to the forth derivative



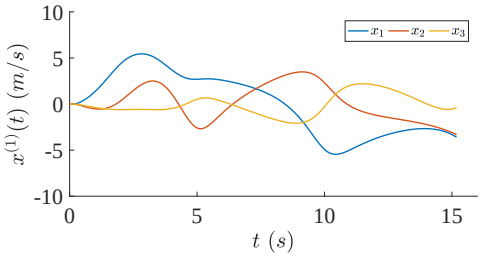
(a) Thrust profile.



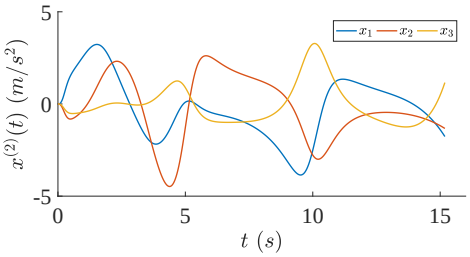
(b) Mapping function.



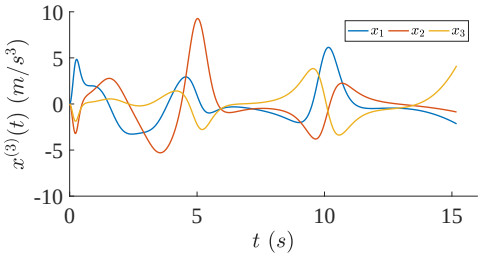
(c) Position.



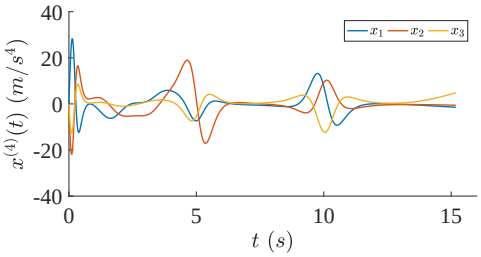
(d) Velocity.



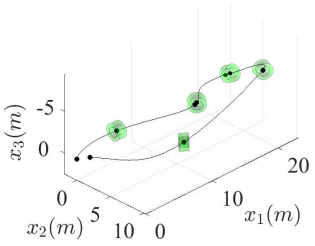
(e) Acceleration.



(f) Jerk.

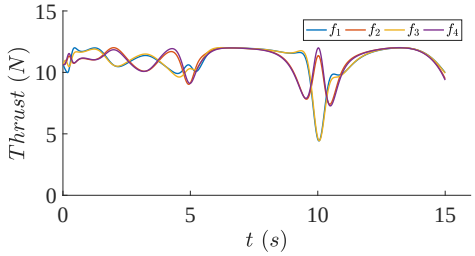


(g) Snap.

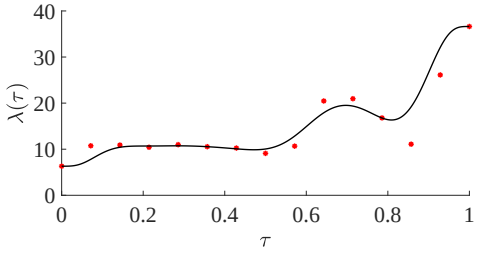


(h) Trajectory.

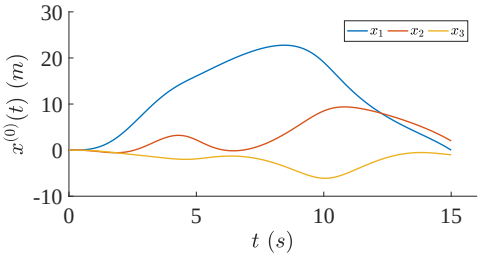
Figure 8.1: Layout comparison: Standard layout, fixed heading – The trajectory results for the gates given in Table 8.1.



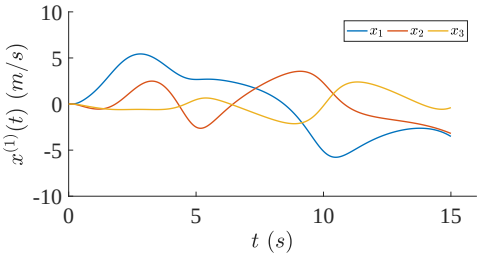
(a) Thrust profile.



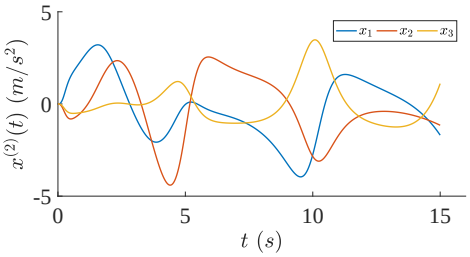
(b) Mapping function.



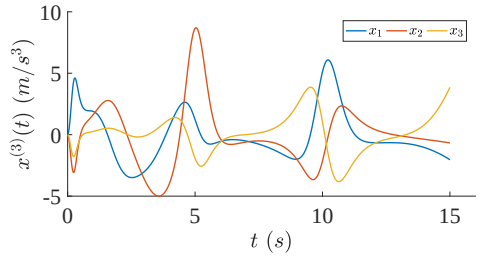
(c) Position.



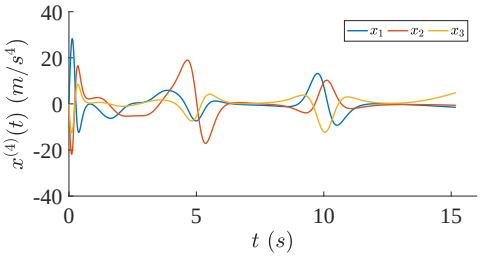
(d) Velocity.



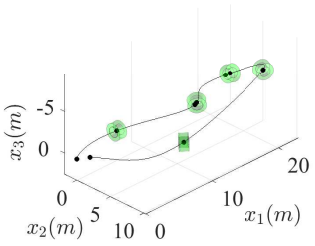
(e) Acceleration.



(f) Jerk.

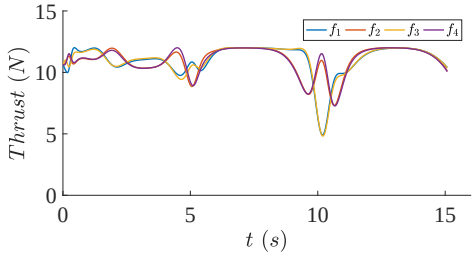


(g) Snap.

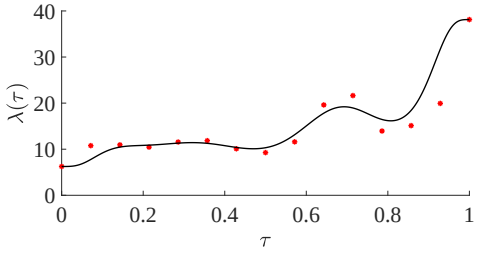


(h) Trajectory.

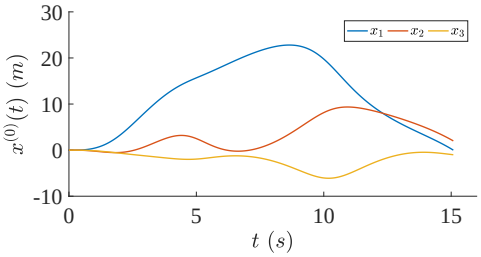
Figure 8.2: Layout comparison: Cross layout, fixed heading – The trajectory results for the gates given in Table 8.1.



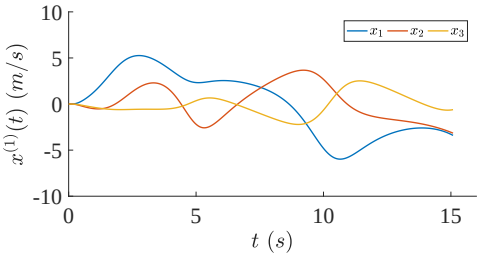
(a) Thrust profile.



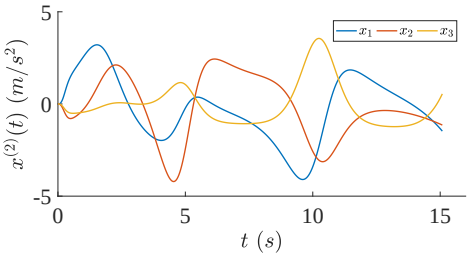
(b) Mapping function.



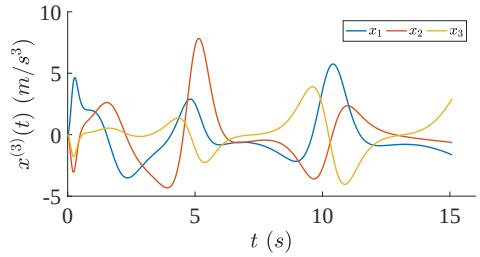
(c) Position.



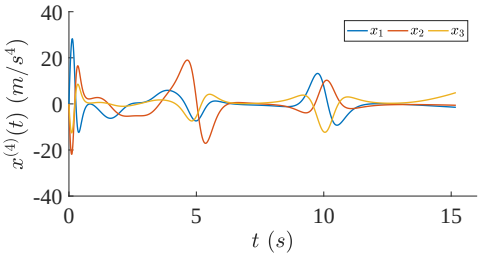
(d) Velocity.



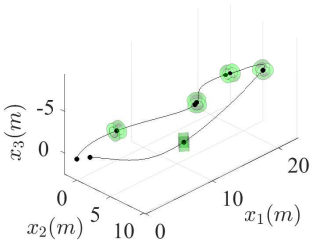
(e) Acceleration.



(f) Jerk.

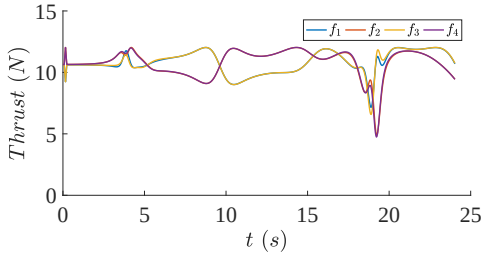


(g) Snap.

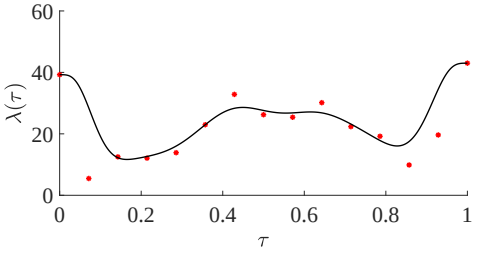


(h) Trajectory.

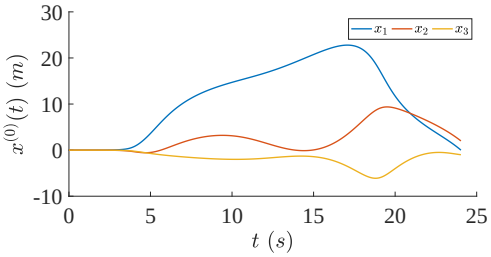
Figure 8.3: Layout comparison: H-shape layout, fixed heading – The trajectory results for the gates given in Table 8.1.



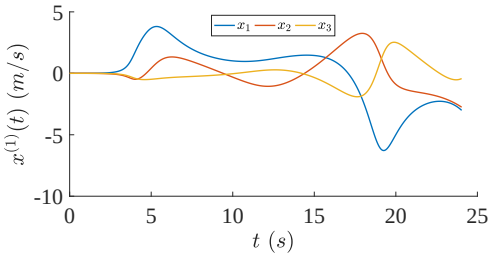
(a) Thrust profile.



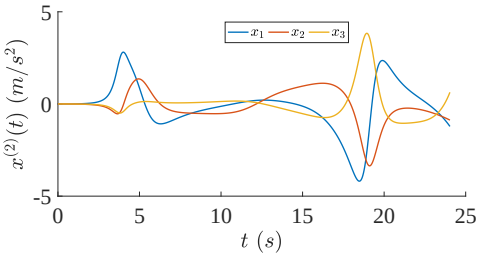
(b) Mapping function.



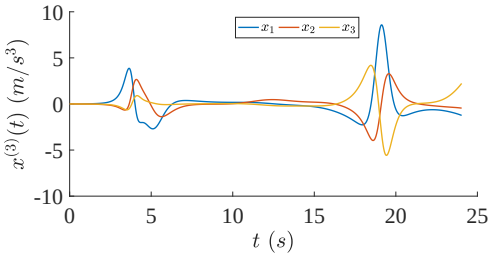
(c) Position.



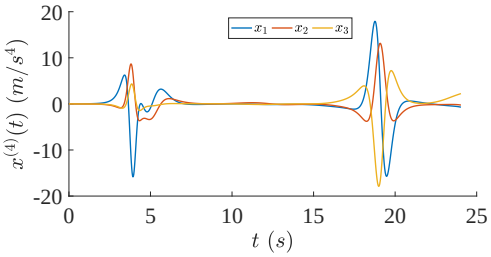
(d) Velocity.



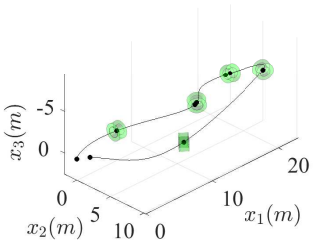
(e) Acceleration.



(f) Jerk.

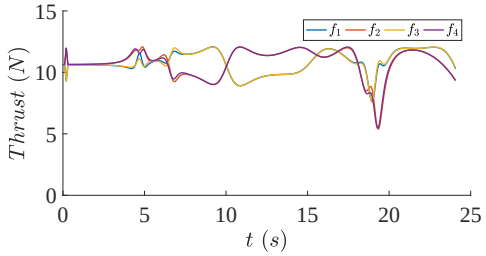


(g) Snap.

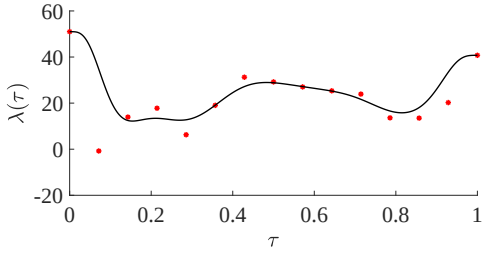


(h) Trajectory.

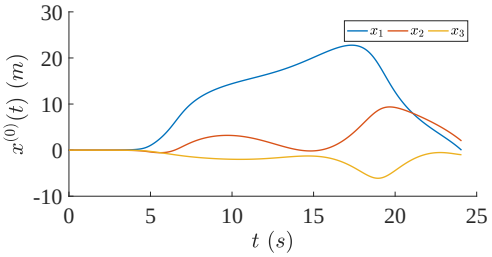
Figure 8.4: Layout comparison: Standard layout, exact heading – The trajectory results for the gates given in Table 8.1.



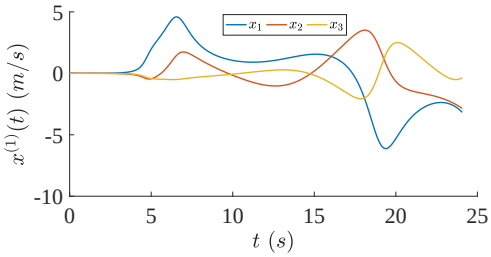
(a) Thrust profile.



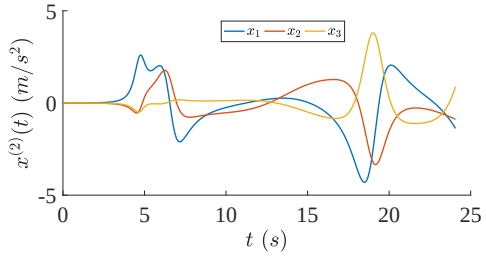
(b) Mapping function.



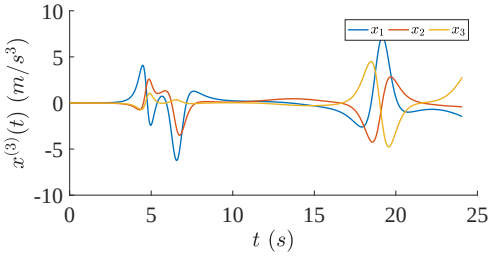
(c) Position.



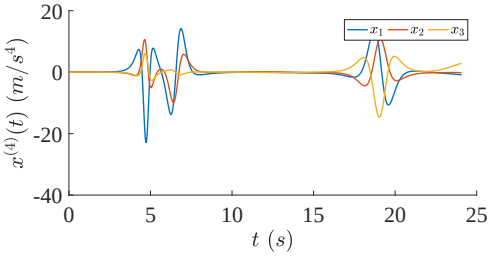
(d) Velocity.



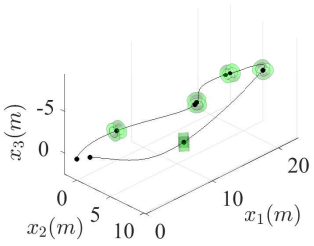
(e) Acceleration.



(f) Jerk.

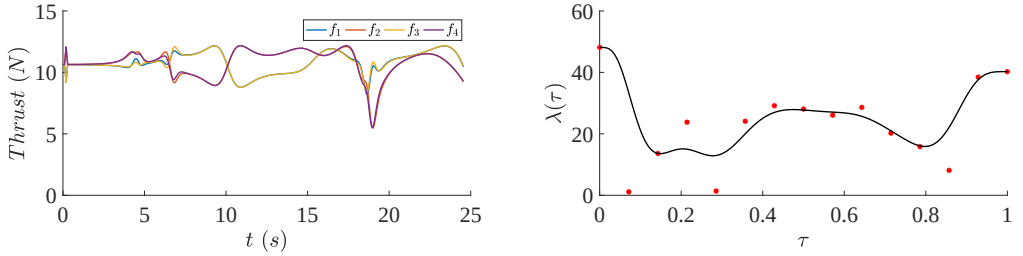


(g) Snap.



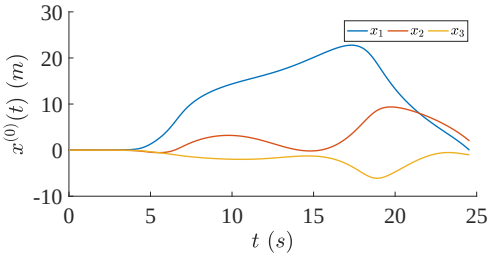
(h) Trajectory.

Figure 8.5: Layout comparison: Cross layout, exact heading – The trajectory results for the gates given in Table 8.1.

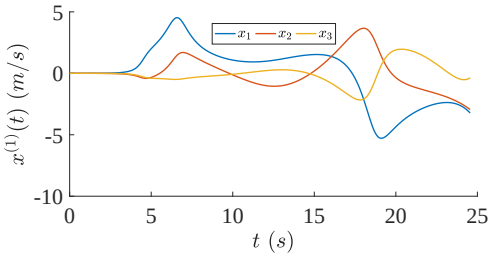


(a) Thrust profile.

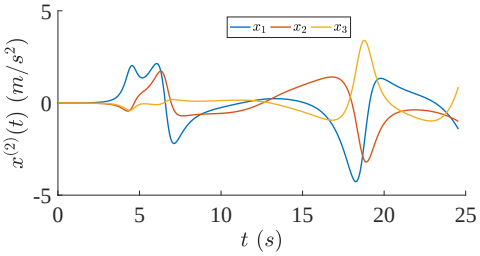
(b) Mapping function.



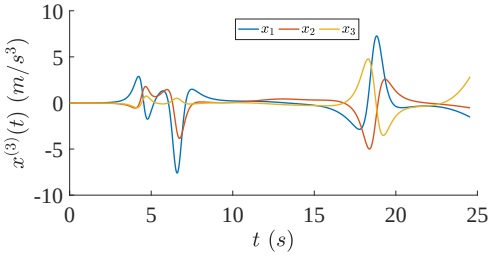
(c) Position.



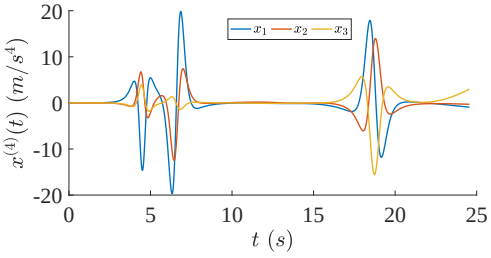
(d) Velocity.



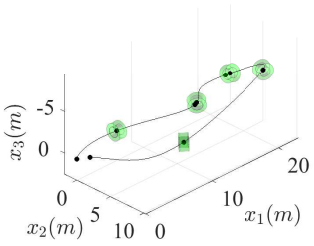
(e) Acceleration.



(f) Jerk.



(g) Snap.



(h) Trajectory.

Figure 8.6: Layout comparison: H-shape layout, exact heading – The trajectory results for the gates given in Table 8.1.

	Fixed (s)	Velocity heading (s)	Optimised heading (s)
Standard	15.18	24.03	19.55
Cross	15.00	24.07	19.23
H-shape	15.08	24.55	19.50

Table 8.2: Layout comparison: trajectory times – The final trajectory times for all the layouts and heading angle methods.

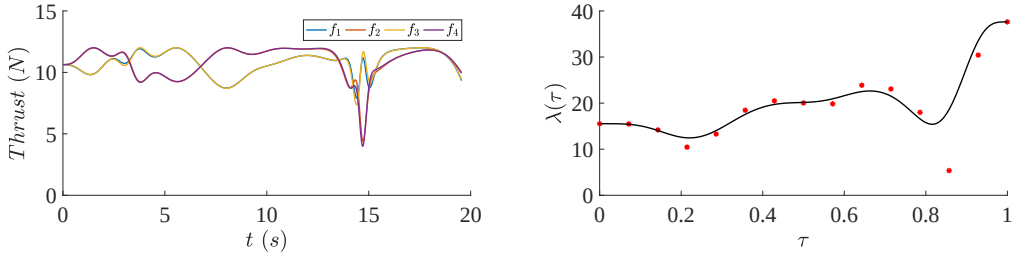
$\theta^{(4)}(0) = \theta^{(3)}(0) = \theta^{(2)}(0) = \theta^{(1)}(0) = 0$ and the final derivatives and angle were not fixed. The trajectory results for the Standard layout, Cross layout and H-Shape are given in Figures 8.7, 8.8 and 8.9 respectively. In the same order, the final trajectory times after mapping were 19.55 s, 19.23 s, 19.50 s respectively. These trajectory times and the trajectory states were similar for each layout. The thrust profiles remained near the maximum rotor limits for most of the trajectory.

8.1.4 Layout configuration summary

The final trajectory times for all the simulations in this section are given in Table 8.2. When the heading angle is fixed the fastest times are obtained and the slowest times occur when the heading angle tracks the velocity direction exactly. The optimised heading angle trajectory times sits between these two cases. If the tolerance value in the heading was relaxed it would be expected that the optimised heading times would approach the fixed heading trajectory time, this is investigated further in the next section. No layout configuration showed any clear advantage and the trajectory times between them were very similar. From the simple dynamic model and tracking controller used in this thesis, none of them could be judged better than the others. However, future work should consider a more complex quadrotor model when simulating their dynamics because it is possible that the simple one used here is not fully capturing their behaviour.

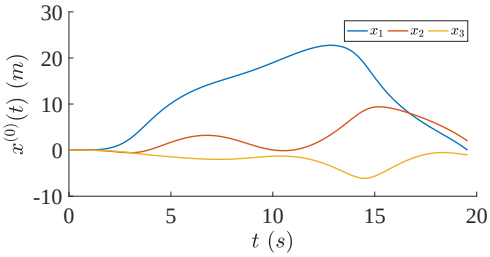
8.2 Optimised heading and trajectory time

Using the same course defined in the previous section, an analysis of the relationship between the trajectory time and θ_{tol} when the heading angle is optimised can be performed. In this section the Standard layout configuration is used to find the trajectory time with a range of θ_{tol} values from 5° to 180°. In total 17 different

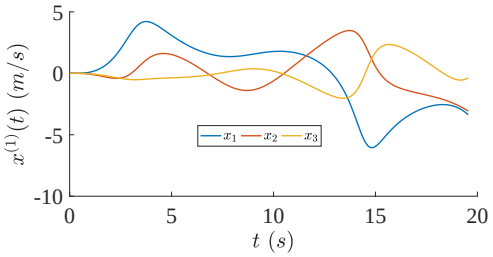


(a) Thrust profile.

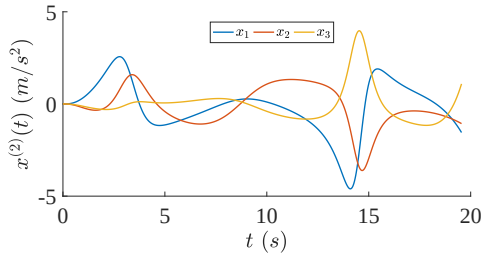
(b) Mapping function.



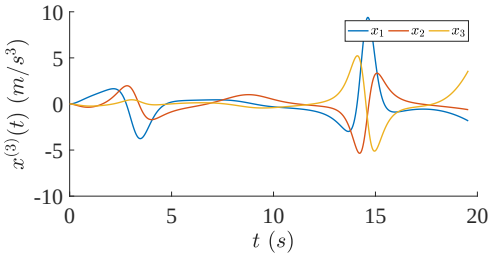
(c) Position.



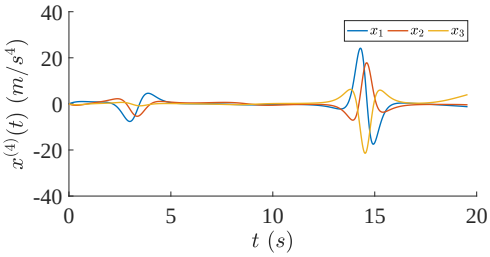
(d) Velocity.



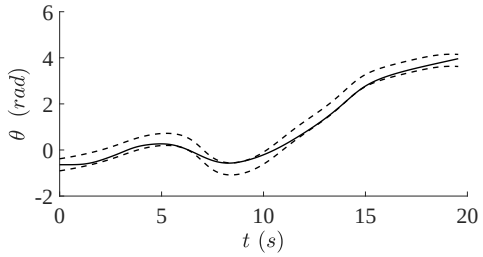
(e) Acceleration.



(f) Jerk.

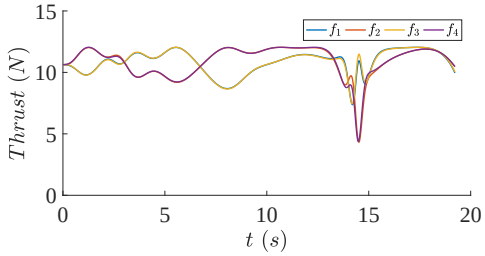


(g) Snap.

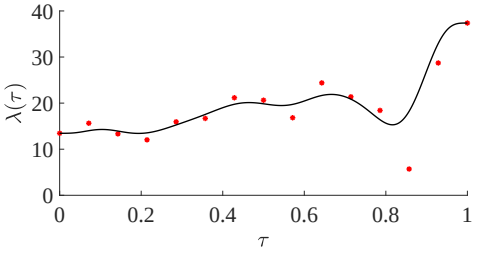


(h) Heading Angle.

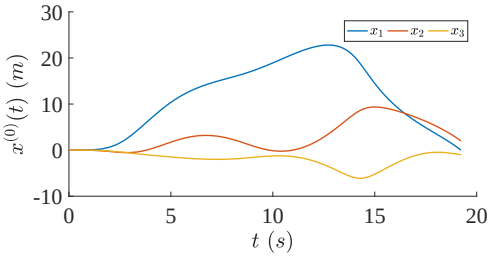
Figure 8.7: Layout comparison: Standard layout, tracked heading – The trajectory results for the gates given in Table 8.1.



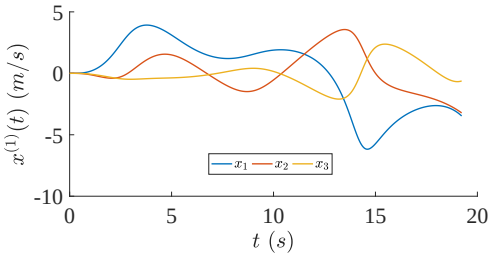
(a) Thrust profile.



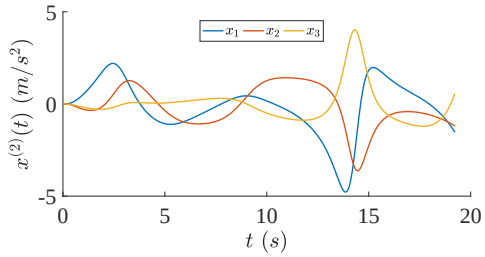
(b) Mapping function.



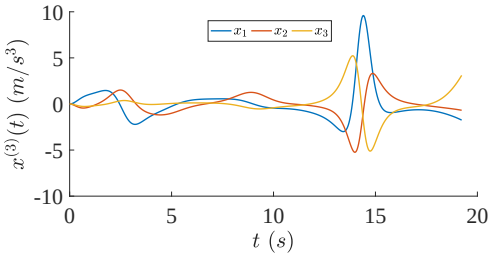
(c) Position.



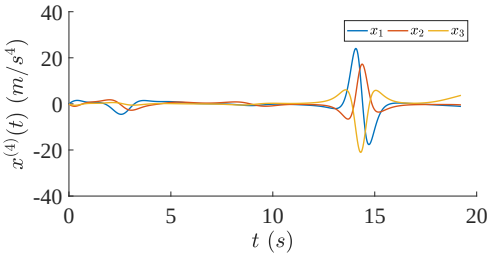
(d) Velocity.



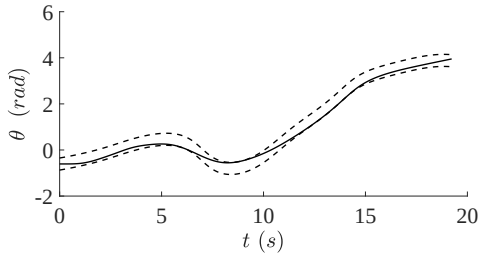
(e) Acceleration.



(f) Jerk.

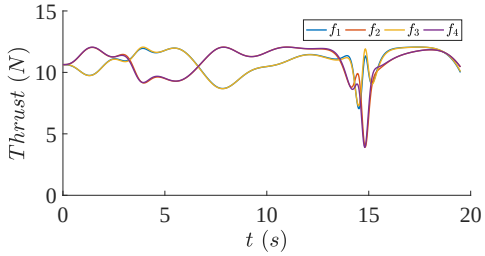


(g) Snap.

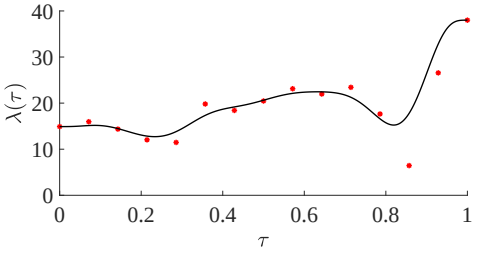


(h) Heading Angle.

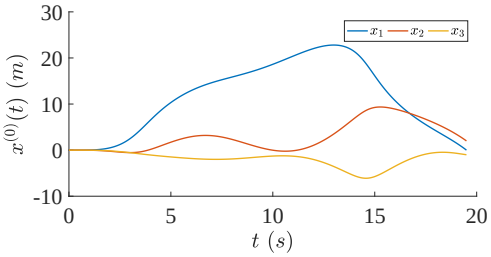
Figure 8.8: Layout comparison: Cross layout, tracked heading – The trajectory results for the gates given in Table 8.1.



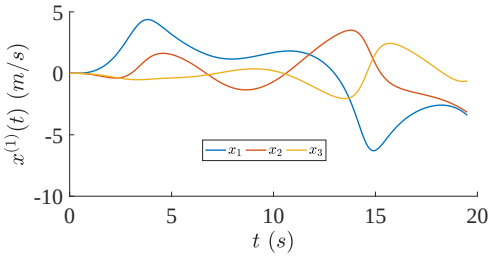
(a) Thrust profile.



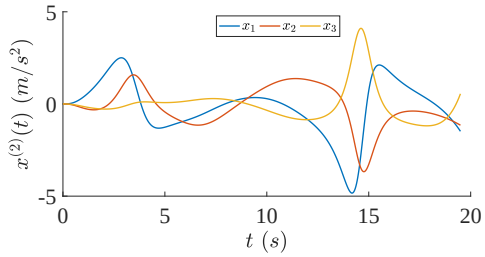
(b) Mapping function.



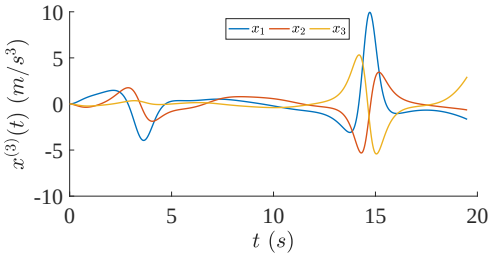
(c) Position.



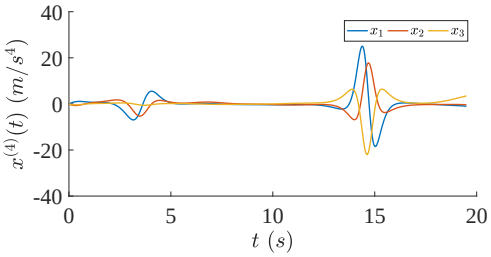
(d) Velocity.



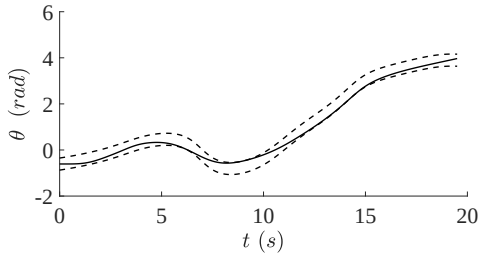
(e) Acceleration.



(f) Jerk.



(g) Snap.



(h) Heading Angle.

Figure 8.9: Layout comparison: H-shape layout, tracked heading – The trajectory results for the gates given in Table 8.1.

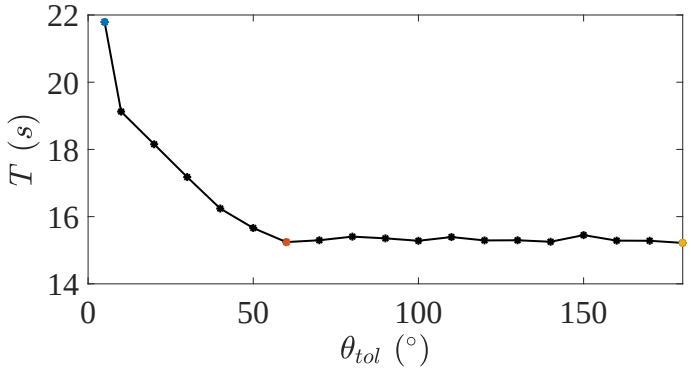
angles were used and these are plotted against their respective trajectory time in Figure 8.10a. The markers locate the angles in which a simulation was performed, three of which are colours other than black and are used to mark angles that were plotted in Figure 8.10b. The trajectory times continue to improve until $\theta_{tol} = 60°$, at which point there is minimal scope to reduce the accumulated angular acceleration. It would be expected that when $\theta_{tol} = 180°$ the trajectory time would be equal to when the heading was fixed $T = 15.18\,\text{s}$, however, the actual time is $T = 15.22\,\text{s}$. This small discrepancy can be explained by Figure 8.10b. The final heading angle when $\theta_{tol} = 180°$ is close to the final angle when the heading follows the velocity vector direction even though it is not constrained to do so. This is because the numerical minimiser has found a local optimum instead the global optimum that would be a straight, horizontal line. In practice the heading optimisation method was intended for stricter tolerances so the initial guess was designed to match the velocity vector heading. If large value is chosen for θ_{tol} then a more suitable initial guess should be found before using the numerical optimiser.

8.3 Additional kinodynamic constraints

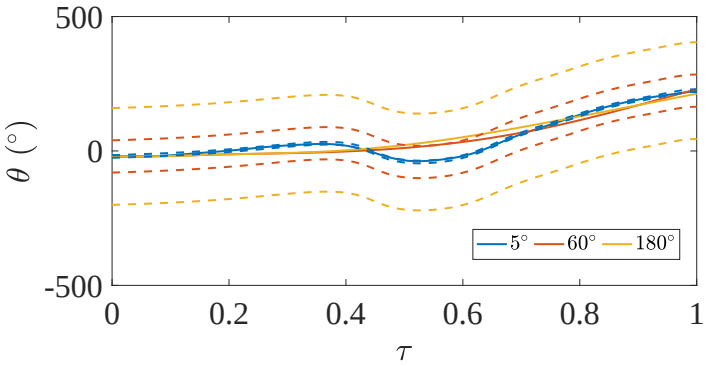
The feasibility of a trajectory has been considered by keeping the rotor thrust within prescribed limits during the time mapping process. However, it is possible to include other kinodynamic limits such as the magnitude of the velocity and acceleration. In this section the gate parameters and boundary conditions from Section 8.1 were used to create a trajectory with a fixed heading angle for the Standard quadrotor layout with the additional limits on the velocity and acceleration magnitudes:

$$\|\mathbf{v}\| \leq 3\,\text{m/s}, \quad \|\dot{\mathbf{v}}\| \leq 1.5\,\text{m/s}^2 \tag{8.1}$$

This is achieved by adding additional penalties to the cost function if either of these constraints is violated. Figure 8.11 shows the trajectory that was created under the above conditions. The thrust profile in Figure 8.11a does not stay near the maximum allowable thrust limit per rotor, instead the trajectory is mainly limited by the constraint on the velocity magnitude (dashed line), shown in Figure 8.11d. There are also periods when the acceleration magnitude (dashed line in



(a) The final trajectory time against the tolerance angle.



(b) Three optimised heading examples.

Figure 8.10: The effect of θ_{tol} on trajectory time T – As the tolerance angle is increased the trajectory times reduce until they are approximately equal to the fixed heading angle case.

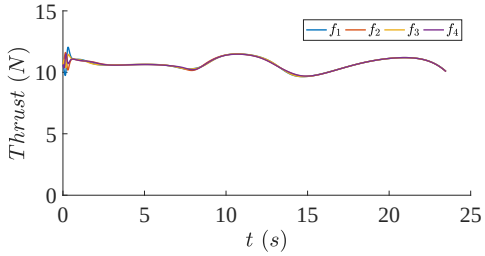
Figure 8.11e) is the limiting factor. Further work will allow kinodynamic limits to be specified at only certain regions of the trajectory. This could be useful for defining safe zones in which the speed must be limited. There may also be other applications where it is desirable for the trajectory to remain steady, for instance if a remote sensor such as a camera was collecting data.

8.4 Quadrotor Trajectory Analyser

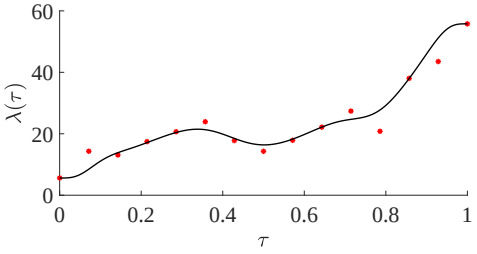
Motion planning is inherently dynamic, the vehicle's states change over time and representing this in a static figure has its limitations. Typically, states are plotted against time but this can be unintuitive, especially to the layperson. For instance, attitude can be represented by the Euler angles roll, pitch and yaw but it is difficult to gain a sense of how this rotation relates to the translational position over time. In order to aid understanding a visualisation tool was developed in Matlab. This software was chosen because it's commonly used amongst researchers and available at most universities. Additionally, since the trajectory generation and tracking algorithms were coded in this environment it is convenient to use it. Outreach and other activities that bring research to a wider audience are becoming increasingly important so a clear and simple way to visualise the trajectory generation and trajectory tracking is useful.

The Quadrotor Trajectory Analyser combines a three dimensional model of a quadrotor following a trajectory with animated plots of its states. The user can decide what outputs to display to ensure the animation is not cluttered by unnecessary information. For example, a simplified version that only gives the speed may be desirable for demonstration purposes but the velocity vector containing the speed of each axis is useful when examining a trajectory in more detail. The translational trajectory is represented by a solid line in space. The position and orientation of the virtual camera can remain static or follow the motion of the vehicle. A stationary camera that displays the entire trajectory gives a broader picture but by tracking the vehicle closer up more detail can be observed.

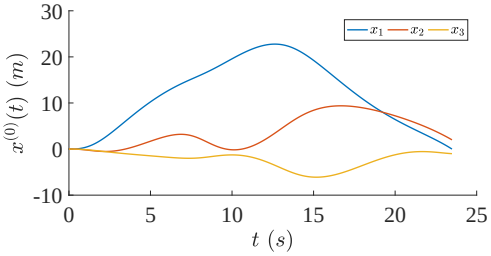
Although it is possible to use a detailed model of a quadrotor in Matlab, for instance by importing a 3D STL (STereoLithography) file, it was found that a simple model consisting of four rods that represented the arms sufficed. The arms



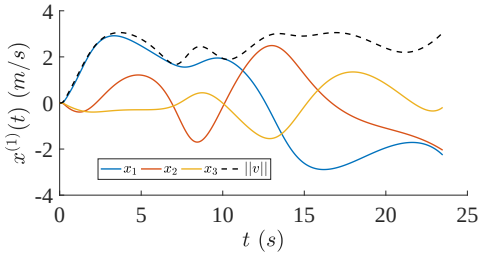
(a) Thrust profile.



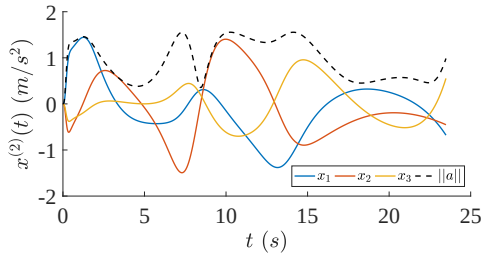
(b) Mapping function.



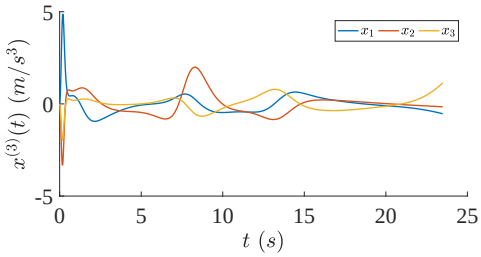
(c) Position.



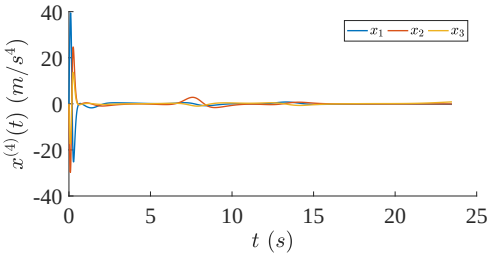
(d) Velocity.



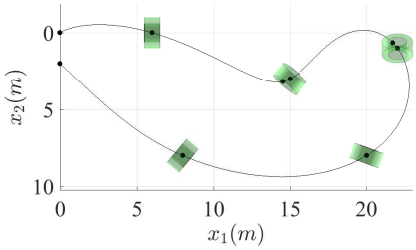
(e) Acceleration.



(f) Jerk.

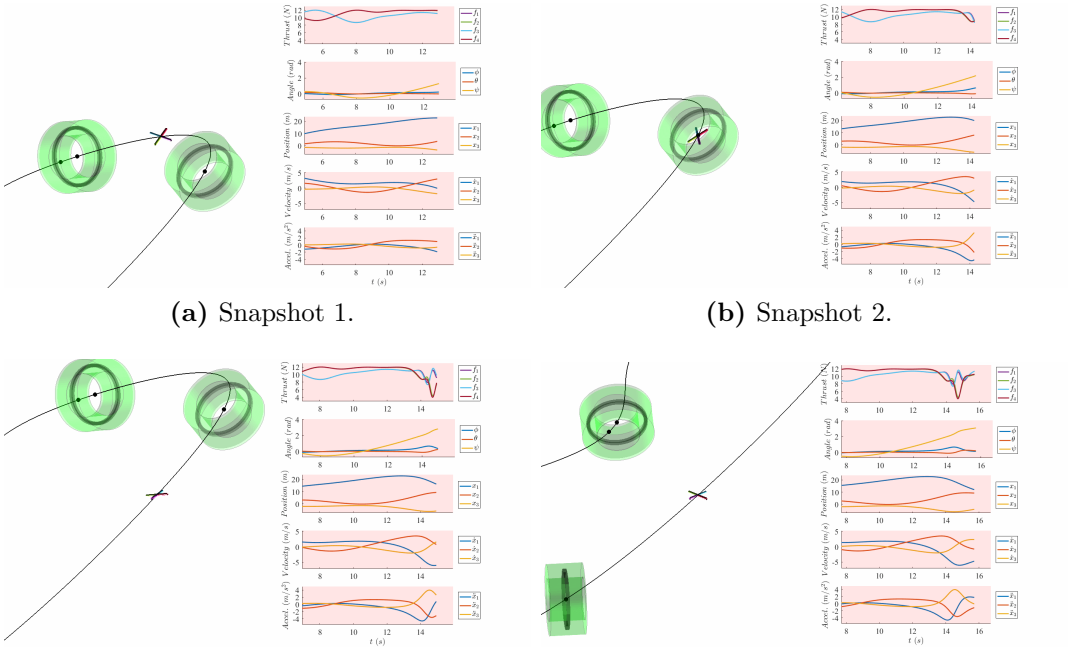


(g) Snap.



(h) Trajectory.

Figure 8.11: Additional kinodynamic constraints trajectory results – The trajectory results when the additional kinodynamic constraints on the maximum magnitude of acceleration and velocity are applied.



(a) Snapshot 1. **(b)** Snapshot 2.

(c) Snapshot 3. **(d)** Snapshot 4.

Figure 8.12: Quadrotor Trajectory Analyser snapshots – Four snapshots from the Quadrotor Trajectory Analyser.

were colour coded to the thrust profile line colours to aid understanding. The process of generating the animation file is generally slower than real-time because Matlab is an interpreted language and the code was not optimised for efficiency. However, the video file containing the animation can have an arbitrary playback speed and frame rate. The position and rotation of the quadrotor is represented and any gates or obstacles can be included.

It was designed to have flexible inputs, if only the position and rotation are given then an inverse dynamics simulation is used to find the thrust controls. Alternatively, if a trajectory has already been full simulated then the vehicle and control states can be plotted directly. The analyser was intended for animated applications rather than static print so in order to demonstrate it, snapshots from an animation are shown in Figure 8.12. The change in position and rotation can be seen clearly. Finally, Figure 8.13 gives the full animation including graphs of the thrust, attitude, position, velocity and acceleration. A video of which can be found at `http://personal.strath.ac.uk/xbb08162/PhD/trajectory.mp4`.

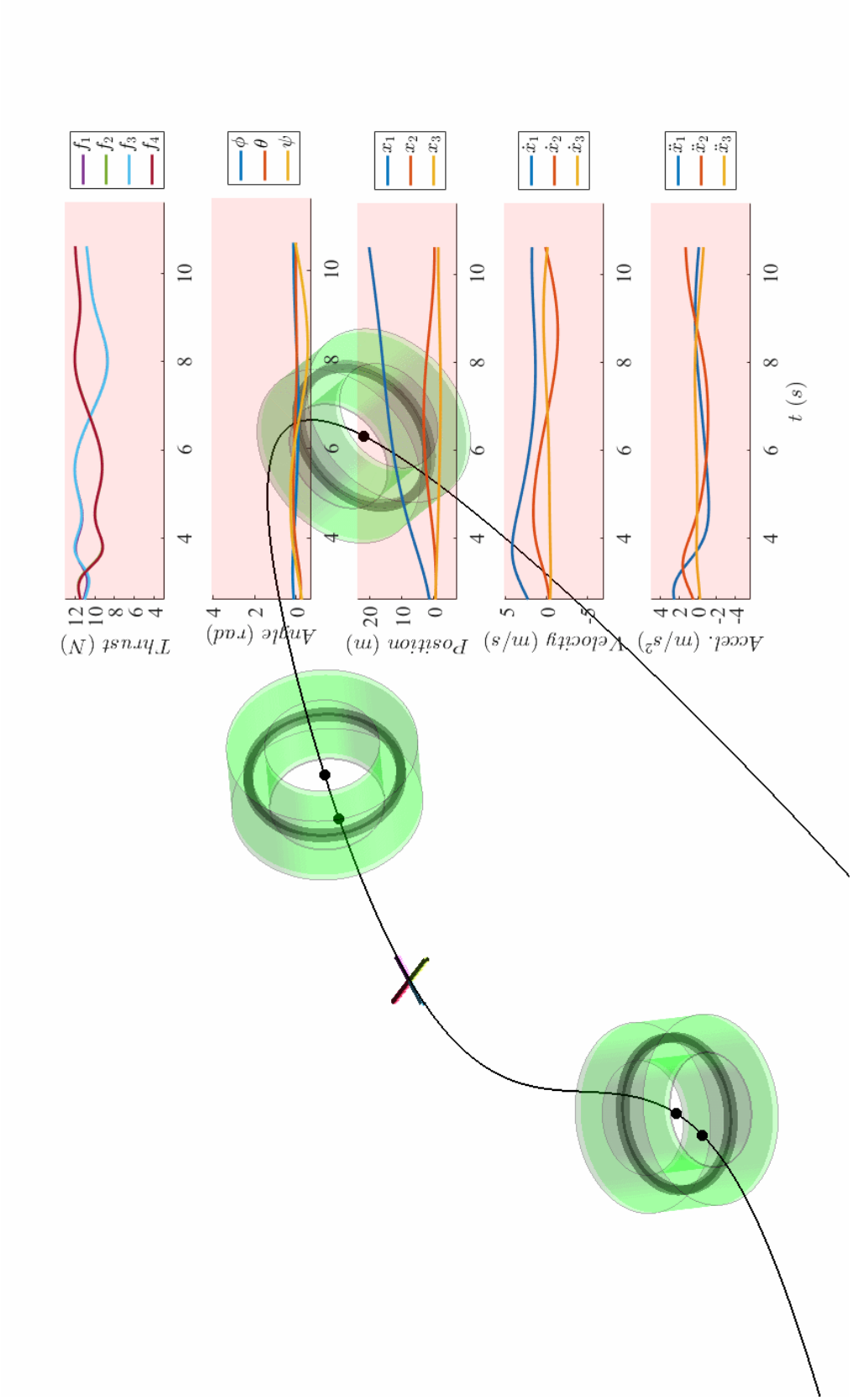


Figure 8.13: Quadrotor Trajectory Analyser – The graphs are overlay the 3D animation and are synchronised to the vehicle's motion.

8.5 Chapter Summary

In this chapter three different quadrotor configurations were compared. It was found that the Standard, Cross and H-Shape layouts all produced similar trajectory times. It was noted that only a simple dynamic model was used and further work would be needed in simulation and real world testing to determine if aerodynamic effects that were not considered would influence the result. Using the same trajectory scenario, different heading angle cases were compared by finding the minimum trajectory time possible for each. Leaving the heading angle fixed and constant was found to produce the fastest time, then the optimised heading angle and finally the heading angle that tracks the planar velocity vector diagram was the slowest. Further testing of the tolerance angle when the heading angle is optimised showed that if the bounds are large a better initial guess may be needed compared to the method developed in Chapter 5. However, the optimised heading angle is capable of handling a wide variety of bounds and improves the minimum trajectory time considerably compared to directly tracking the velocity direction.

The time mapping method was revisited with additional constraints on the kinodynamics. An example in which the magnitude of the velocity and magnitude of velocity were considered in addition to the limits on the rotor thrusts. This may be useful for certain applications in which the vehicle's must be controlled tightly. It was also noted that future work should consider only apply additional constraints at certain points during the trajectory.

Finally the Quadrotor Trajectory Analyser was presented. The QTA can animate a quadrotor trajectory in 3D in order to aid understanding of the motion and is a useful tool for researchers, users and outreach to the general public. Future versions of the QTA will consider modelling other vehicles such as fixed wing aerial vehicles.

Chapter 9

Conclusions and future work

The previous chapters developed trajectory planning and trajectory tracking methods for quadrotor UAVs with particular attention paid to the drone racing application. Trajectory planning for long, slender AUVs was also considered as an additional application. The following chapter will summarise the contributions of this thesis, discuss the limitations of this thesis and finally, suggest future areas of research.

9.1 Reasearch outcomes and limitations

This thesis was motivated by the need to plan dynamically feasible, collision free, optimal trajectories for autonomous quadrotor UAVs. The main focus was finding minimum time trajectories suitable for drone racing. To this end, geometrical control theory methods with inherent optimality were used to derive basis functions for two types of non-holonomic constraints. These basis functions were analogous to sub-Riemannian curves and have been used for trajectory planning in previous works but not for the case of quadrotors. The final translational position of the vehicle could be specified for both types of non-holonomic constraints using parametric optimisation. Although the final velocity direction could not be arbitrarily chosen, including it in a numerical minimisation allowed for the shape of the curve to be influenced. This enabled simple obstacle avoidance to be demonstrated for both cases. It was found that the case of non-holonomic constraints that constrained the relation between the first and second translational body velocity, and the first and second angular body velocity gave the greatest

flexibility in the curve shape. This result was obtained numerically using Monte Carlo simulations. Since a quadrotor is capable of tracking trajectories for both cases, the multiple body velocities case is recommended if a point-to-point manoeuvre is required. However, the single body velocity case can be combined with a sampling based motion planning method, RRT, that concatenates trajectory segments into a single trajectory. This was demonstrated with the application of AUVs but could also be used for quadrotor trajectories. A neutrally buoyant AUV was used because the kinematic description of the curve included the translational position and attitude. To the author's knowledge, this is the first time these types of curves have been used within an RRT framework.

The main limitation of the curves found using these geometric control methods is that the speed is constant and there is little control over the derivatives at the boundaries. For drone racing courses, the lack of flexibility in the curves rendered them unsuitable for that application. Therefore a method for defining trajectories in the virtual time domain using polynomials as the basis functions was developed. Polynomials have been used previously for trajectory planning but the method for defining consecutive waypoints and minimising for the shortest path length in this thesis is novel. In order to fly through the gates without collision, a method for changing the shape of the trajectory was required. This was achieved using the Waypoint Selection Algorithm that added additional waypoints according to a method designed to ensure the curve remained smooth by not creating tight turns that increase trajectory time. The Waypoint Selection Algorithm was demonstrated with a variety of test cases including a single gate, multiple gates and a drone racing course.

Another aspect of drone racing was considered whilst generating the virtual domain trajectories, namely, the direction of the heading angle. It was assumed that a camera fixed to point in the direction of the first body axis was being used to navigate so the heading angle should be chosen such that it points in the direction of motion. However, due to the use of wide angled lenses it is not necessary for the heading angle to point exactly in the direction of the planar velocity vector and can instead be optimised to remain within a defined tolerance. A method for optimising the heading angle during the trajectory to minimise the accumulated angular acceleration was developed and tested. Previously in the literature, the heading angle was either fixed or defined by some arbitrary function such as a sinusoid. It was found that for the types of trajectories in this

thesis 15 unique knots for the B-spline function that defines the heading angle is best for the numerical optimiser. It was shown that the optimised heading angle resulted in shorter trajectory times when compared to the case of the heading angle directly tracking the planar motion.

The polynomial based trajectory generation was separated into two parts, the previously discussed virtual domain trajectories and the mapping method for converting these to time domain trajectories. The time mapping method has been used in a variety of applications but in this thesis it is used to find feasible trajectories that minimise trajectory time for quadrotors with the feasibility ensured by checking the thrust required from each rotor using a nonlinear inverse dynamics simulation. Since the value of the boundary derivatives in the virtual domain are dependent on the boundary mapping values, the geometrical shape of the trajectory changes when the boundary mapping values are adjusted. This is necessary to ensure a feasible trajectory is found but recomputing the virtual trajectory is computationally costly so it is desirable for the boundary values to be found efficiently. To this end, a novel algorithm was developed to find boundary mapping values that produced a feasible trajectory with minimal trajectory time. These mapping values were then used as the basis of the initial guess for refining the rest of the mapping values in order to further reduce the trajectory time. Three different methods were tested and it was found that defining the mapping function with a B-spline and numerically minimising according to a cost function produced the best results with respect to minimising the trajectory time. This is similar to the case for optimising the heading angle, 15 unique knots for the B-spline was found to be the optimum number according to a Monte Carlo simulation based on 50 trajectories. It was also demonstrated that additional kinodynamic constraints, such as those on the velocity and acceleration, could be included in the time mapping method.

Further simulations were performed to test three common quadrotors layouts because some racing drone operators express a preference for particular configurations. However, in the simulations using the dynamic model from this thesis no significant difference of the trajectory times could be found between the layouts.

The Quadrotor Trajectory Analyser was created to enable a better visualisation of the trajectories by animating the motion of the vehicle in 3D. This should be a useful tool for research and outreach because it is more suitable than commercially available simulation software for playing back and demonstrating

planned trajectories.

The trajectory tracking problem in the presence of disturbances was also addressed. An existing nonlinear tracking controller on SE(3) was modified to include a LESO that estimated and attempted to reject any translational disturbance. This is achieved by using the estimated disturbance within the tracking controller so it can account for the actual disturbance and remain closer to the desired, planned trajectory. A convergence proof was offered for the LESO and an expression for the bound given. The Modified Controller and Standard Controller were compared using three types of disturbances. It was found that the Modified Controller rejected the continuous disturbances like a sinusoid or wind simulation best, however, it also showed improved translational tracking performance for square waves compared to the Standard Controller. Previous attempts in the literature had only considered Extended State Observers for attitude only or linearised controller dynamics.

9.2 Future work

From the outcomes of this thesis given above, this section suggests further work and extensions for the methods developed.

It has been noted previously that the curves derived with geometric control theory in this thesis cannot have their velocity vector direction specified at the end of the trajectory. This limitation is probably not due to the parametric optimiser being unable to find the correct optimisation vector but because a particular solution was found during the curve derivation that is a subset of the general solution. This meant that the analytical functions defining the trajectories could be written as standard trigonometric functions whereas the general solution would have required elliptic functions. The particular solutions are a subset of the general solutions so if the latter had been used more trajectory shapes would have been available to the optimiser. Future work should investigate using the general solution for quadrotor trajectories. The ability for efficient, onboard calculation of elliptical functions could also be necessary. If the general solution were used it may also be possible to integrate the trajectory generation method with the RRT* framework to find trajectories that are optimal in path length. The parametric optimisation used to minimise the cost function related to the

final position could also be investigated. If a better initial guess was provided then the solution should be found more rapidly.

The polynomial trajectory generation method would also benefit from an improvement in calculation speed, especially during the mapping process when it is called repeatedly. Currently, the position on the virtual domain of the waypoints during the mapping process is not retained and is only passed from one optimisation to the next. If all the previous waypoint optimisation vectors were available then the closest could be used as an initial guess for any new boundary mapping values being attempted. There are also numerical issues when large numbers of conditions are required because the order of the polynomials that define the virtual domain trajectory grow. If the polynomials become badly scaled then the numerical results may be inaccurate. The condition of a polynomial can be checked to ensure it is valid but if a high order is necessary then it may be more appropriate to use splines, a series of connected polynomials that are smooth to a given degree at the boundaries. Future work should consider the use of these to avoid numerical issues.

It may also be possible to improve the numerical efficiency of the mapping function optimisation. This could be achieved by designing an optimiser that has an inbuilt knowledge of the problem that avoids unnecessary iterations in which obviously invalid optimisation vectors are attempted. Currently, the mapping function tries to minimise the final trajectory time and ensure the thrust from each rotor remains within the defined limits. However, other costs functions could also be considered, such as minimising the amount of energy required to complete a manoeuvre. The drone racing application is just one of many, so improving the abilities of the methods in this thesis should benefit other areas.

With regards to trajectory tracking, the Modified Controller only considered the rejection of translational disturbances. Future work should also investigate the use of an observer for the rotational dynamics and disturbances. This should improve the ability of the controller to reduce the tracking error in both the translational and rotational position. A nonlinear ESO is another possibility for improving the performance. In theory, a well tuned nonlinear observer would be capable of estimating a disturbance better than a linear one. However, as mentioned previously, more parameters must be tuned compared to the linear case. A more realistic quadrotor simulation to test the controllers is another important area for future work. Currently, aerodynamic effects are not included

in the dynamic model and they may have an effect on the controller performance. Ideally, both the Standard and Modified Controller should be tested on a real quadrotor. This would allow the best possible comparison to be made between the two and also enable their performance to be characterised, so the simulations can be improved.

Finally, the trajectory generation methods developed in this thesis are theoretical because the state-of-the-art quadrotors at this point in time are unable to reliably and accurately track the aggressive trajectories that have been generated. This is particularly true for the case of outdoor flying where it is difficult to obtain an accurate, frequently updated information of the vehicle's position for the controller. In the short term, strict kinodynamic limits can be applied during the motion planning process in order to generate trajectories that are easier to follow. As the technology improves these limitations can be relaxed and greater use of the vehicle performance can be gained. Human pilots are capable of flying at fast speeds with aggressive turns and manoeuvre through small gates, future autonomous drones should be able to participate at the same level or higher.

Bibliography

[1] H. S. Wolko and J. D. Anderson. The wright flyer: an engineering perspective. 1983.

[2] J. G. Leishman. The breguet-richet quad-rotor helicopter of 1907. *Vertiflite*, 47(3): 58–60, 2002.

[3] A. Gessow and G. C. Myers. *Aerodynamics of the Helicopter*. Frederick Ungar, 1967.

[4] R. Baránek and F. Šolc. Modelling and control of a hexa-copter. *Carpathian Control Conference (ICCC), 2012 13th International*, pages 19–23. IEEE, 2012.

[5] A. Alaimo, V. Artale, C. Milazzo, A. Ricciardello and L. Trefiletti. Mathematical modeling and control of a hexacopter. *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 1043–1050. IEEE, 2013.

[6] V. Artale, C. Milazzo and A. Ricciardello. Mathematical modeling of hexacopter. *Appl. Math. Sci*, 7(97): 4805–4811, 2013.

[7] M. J. Er, S. Yuan and N. Wang. Development control and navigation of octocopter. *Control and Automation (ICCA), 2013 10th IEEE International Conference on*, pages 1639–1643. IEEE, 2013.

[8] A. Chamseddine, D. Theilliol, I. Sadeghzadeh, Y. Zhang and P. Weber. Optimal reliability design for over-actuated systems based on the MIT rule: Application to an octocopter helicopter testbed. *Reliability Engineering & System Safety*, 132: 196–206, 2014.

[9] M. Oczipka, J. Bemmann, H. Piezonka, J. Munkabayar, B. Ahrens, M. Achtelik and F. Lehmann. Small drones for geo-archaeology in the steppe: locating and documenting the archaeological heritage of the Orkhon Valley in Mongolia. *SPIE Europe Remote Sensing*, pages 747806–747806. International Society for Optics and Photonics, 2009.

[10] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim and G. Sanahuja. Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor. *Robotics and Automation (ICRA), 2015 IEEE International Conference*, pages 5266–5271. IEEE, 2015.

[11] T. Schneider, G. Ducard, R. Konrad and S. Pascal. Fault-tolerant control allocation for multirotor helicopters using parametric programming. *International Micro Air Vehicle Conference and Flight Competition (IMAV)*. 2012.

[12] S. Salazar-Cruz and R. Lozano. Stabilization and nonlinear control for a novel trirotor mini-aircraft. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2612–2617. IEEE, 2005.

[13] D.-W. Yoo, H.-D. Oh, D.-Y. Won and M.-J. Tahk. Dynamic modeling and stabilization techniques for tri-rotor unmanned aerial vehicles. *International Journal Aeronautical and Space Sciences*, 11(3): 167–174, 2010.

[14] J.-S. Chiou, H.-K. Tran and S.-T. Peng. Attitude control of a single tilt tri-rotor UAV system: dynamic modeling and each channel's nonlinear controllers design. *Mathematical Problems in Engineering*, 2013, 2013.

[15] U. Niethammer, M. James, S. Rothmund, J. Travelletti and M. Joswig. UAV-based remote sensing of the Super-Sauze landslide: Evaluation and results. *Engineering Geology*, 128: 2–11, 2012.

[16] M. Walter, U. Niethammer, S. Rothmund and M. Joswig. Joint analysis of the Super-Sauze (French Alps) mudslide by nanoseismic monitoring and UAV-based remote sensing. *First Break*, 27(8), 2009.

[17] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz and J. Valente. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5): 667–689, 2011.

[18] J. Valente, D. Sanz, A. Barrientos, J. d. Cerro, Á. Ribeiro and C. Rossi. An air-ground wireless sensor network for crop monitoring. *Sensors*, 11(6): 6088–6108, 2011.

[19] D. Wang. The economics of drone delivery, 2016. Accessed: 17-May-2017. Available at `https://www.flexport.com/blog/drone-delivery-economics/`.

[20] R. A. Clark, G. Punzo, C. N. MacLeod, G. Dobie, R. Summan, G. Bolton, S. G. Pierce and M. Macdonald. Autonomous and scalable control for remote inspection with multiple aerial vehicles. *Robotics and Autonomous Systems*, 87: 258–268, 2017.

[21] BBC. Lady Gaga dives into Super Bowl history, 2016. Accessed: 16-May-2017. Available at http://www.bbc.co.uk/news/entertainment-arts-38875699.

[22] N. Joubert, M. Roberts, A. Truong, F. Berthouzoz and P. Hanrahan. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics (TOG)*, 34(6): 238, 2015.

[23] L. Meier, P. Tanskanen, F. Fraundorfer and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2992–2997. IEEE, 2011.

[24] N. Hurst. Formula FPV. *Make*, 44: 25–28, 2015.

[25] BBC. Boy wins first World Drone Prix in Dubai, 2016. Accessed: 7-September-2016. Available at `http://www.bbc.co.uk/newsround/35973020`.

[26] The drone racing league. Accessed: 16-03-2017. Available at `https://thedroneracingleague.com/`.

[27] British FPV racing association. Accessed: 16-03-2017. Available at `http://bfpvra.org/`.

[28] Freedom class giant drone racing. Accessed: 16-03-2017. Availavle at `http://www.fcracing.com/`.

[29] First person view racing. Accessed: 16-03-2017. Available at `http://fpvr.org/`.

[30] British teen wins 250,000 dollars in world's biggest drone race. Accessed: 16-03-2017. Available at `http://www.wired.co.uk/article/british-teenager-luke-bannister-wins-worlds-biggest-drone-race`.

[31] Iros 2016 autonomous drone racing challenge. Accessed: 16-03-2017. Available at `http://ris.skku.edu/home/iros2016racing.html`.

[32] T. Lozano-Perez. The design of a mechanical assembly system. Technical report, MIT, 1976.

[33] M. Hoy, A. S. Matveev and A. V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(03): 463–497, 2015.

[34] D. Mellinger, N. Michael and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 31(5): 664–674, 2012.

[35] R. M. Murray, M. Rathinam and W. Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. *ASME International Mechanical Engineering Congress and Exposition*. ASME, 1995.

[36] L. Babel. Three-dimensional route planning for unmanned aerial vehicles in a risk environment. *Journal of Intelligent & Robotic Systems*, 71(2): 255–269, 2013.

[37] M. L. Ireland. *Investigations in multi-resolution modelling of the quadrotor micro air vehicle*. Ph.D. thesis, University of Glasgow, 2014.

[38] A. Caubet and J. Biggs. A motion planning method for spacecraft attitude maneuvers using single polynomials. *AAS/AIAA Astrodynamics Specialist Conference*. 2015.

[39] A. Caubet and J. D. Biggs. An inverse dynamics approach to the guidance of spacecraft in close proximity of tumbling debris. *International Astronautical Congress*. 2015.

[40] A. Johnson. Curvefitting. *Techniques in the behavioral and neural sciences*, 5: 309–336, 1991.

[41] C. Richter, A. Bry and N. Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. *Proceedings of the International Symposium of Robotics Research (ISRR 2013)*. Singapore, 2013.

[42] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.

[43] M. Hehn and R. D'Andrea. Quadrocopter trajectory generation and control. *IFAC Proceedings Volumes*, 44(1): 1485–1491, 2011.

[44] J. Jamieson and J. Biggs. Near minimum-time trajectories for quadrotor UAVs in complex environments. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1550–1555. 2016.

[45] H. M. Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[46] J. Leonard, A. Savvaris and A. Tsourdos. Distributed reactive collision avoidance for a swarm of quadrotors. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, page 0954410016647074, 2016.

[47] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell and M. Hebert. Learning monocular reactive UAV control in cluttered natural environments. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[48] E. Anderson. *Quadrotor Implementation of the Three-Dimensional Distributed Reactive Collision Avoidance Algorithm*. Ph.D. thesis, University of Washington, 2011.

[49] H. Alvarez, L. Paz, J. Sturm and D. Cremers. Collision avoidance for quadrotors with a monocular camera. *Experimental Robotics*, pages 195–209. Springer, 2016.

[50] T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10): 560–570, 1979.

[51] J.-P. Laumond. Obstacle growing in a nonpolygonal world. *Information processing letters*, 25(1): 41–50, 1987.

[52] J. T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research*, 2(3): 46–75, 1983.

[53] J. T. Schwartz and M. Sharir. On the piano movers' problem: V. the case of a rod moving in three-dimensional space amidst polyhedral obstacles. *Communications on Pure and Applied Mathematics*, 37(6): 815–848, 1984.

[54] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, New York, NY, USA, 2006.

[55] J.-C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[56] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[57] G. Punzo, G. Dobie, D. J. Bennet, J. Jamieson and M. Macdonald. Low-cost, multi-agent systems for planetary surface exploration. *63rd International Astronautical Congress*. 2012.

[58] R. Clark, G. Punzo, G. Dobie, C. MacLeod, R. Summan, G. Pierce, M. Macdonald, G. Bolton, D. E. Chimenti and L. J. Bond. 3D model generation using an airborne swarm. *AIP Conference Proceedings*, volume 1650, pages 1460–1467. AIP, 2015.

[59] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6): 628–649, 1991.

[60] G. Radice and I. Ali. Autonomous attitude using potential function method under control input saturation. 2008.

[61] L. E. Kavraki, P. Svestka, J.-C. Latombe and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4): 566–580, 1996.

[62] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1): 269–271, 1959.

[63] S. M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[64] I. Garcia and J. P. How. Improving the efficiency of rapidly-exploring random trees using a potential function planner. *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 7965–7970. IEEE, 2005.

[65] S. Khanmohammadi and A. Mahdizadeh. Density avoided sampling: An intelligent sampling technique for Rapidly-Exploring Random Trees. *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*, pages 672–677. IEEE, 2008.

[66] J. Hall and D. Anderson. Reactive route selection from pre-calculated trajectoriesapplication to micro-uav path planning. *Aeronautical Journal*, 115(1172): 635, 2011.

[67] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7): 846–894, 2011.

[68] Q. Gong, W. Kang, N. S. Bedrossian, F. Fahroo, P. Sekhavat and K. Bollino. Pseudospectral optimal control for military and industrial applications. *Decision and Control, 2007 46th IEEE Conference on*, pages 4128–4142. IEEE, 2007.

[69] W. Kang and N. Bedrossian. Pseudospectral optimal control theory makes debut flight, saves NASA 1Min under three hours. *SIAM news*, 40(7): 1–3, 2007.

[70] V. M. Becerra. Solving complex optimal control problems at no cost with PSOPT. *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*, pages 1391–1396. IEEE, 2010.

[71] C. D. Maclean. *Analytical motion planning on 3D-Lie groups*. Ph.D. thesis, University of Strathclyde, 2014.

[72] G. Ducard and R. D'Andrea. Autonomous quadrotor flight using a vision system and accommodating frames misalignment. *2009 IEEE International Symposium on Industrial Embedded Systems*, pages 261–264. IEEE, 2009.

[73] M. Achtelik, T. Zhang, K. Kuhnlenz and M. Buss. Visual tracking and control of a quadcopter using a stereo camera system and inertial sensors. *2009 International Conference on Mechatronics and Automation*, pages 2863–2869. IEEE, 2009.

[74] G. Dobie, R. Summan, C. MacLeod and S. G. Pierce. Visual odometry and image mosaicing for NDE. *NDT & E International*, 57: 17–25, 2013.

[75] G. J. Morgan-Owen and G. T. Johnston. Differential GPS positioning. *Electronics Communication Engineering Journal*, 7(1): 11–21, 1995.

[76] O. Shakernia, Y. Ma, T. J. Koo and S. Sastry. Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control. *Asian journal of control*, 1(3): 128–145, 1999.

[77] O. Dunkley, J. Engel, J. Sturm and D. Cremers. Visual-inertial navigation for a camera-equipped 25g nano-quadrotor.

[78] C. Kownacki. Optimization approach to adapt Kalman filters for the real-time application of accelerometer and gyroscope signals' filtering. *Digital Signal Processing*, 21(1): 131–140, 2011.

[79] P. Castillo, R. Lozano and A. Dzul. Stabilization of a mini-rotorcraft having four rotors. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2693–2698 vol.3. 2004.

[80] G. M. Hoffmann, H. Huang, S. L. Waslander and C. J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. 2007.

[81] G. V. Raffo, M. G. Ortega and F. R. Rubio. An integral predictive/nonlinear control structure for a quadrotor helicopter. *Automatica*, 46(1): 29 – 39, 2010.

[82] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. 2006.

[83] B. Wie and P. M. Barba. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, 8(3): 360–365, 1985.

[84] A. Tayebi and S. McGilvray. Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Transactions on control systems technology*, 14(3): 562–571, 2006.

[85] E. Fresk and G. Nikolakopoulos. Full quaternion based attitude control for a quadrotor. *Control Conference (ECC), 2013 European*, pages 3864–3869. 2013.

[86] S. P. Bhat and D. S. Bernstein. A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon. *Systems & Control Letters*, 39(1): 63–70, 2000.

[87] C. G. Mayhew, R. G. Sanfelice and A. R. Teel. Quaternion-based hybrid control for robust global attitude tracking. *IEEE Transactions on Automatic Control*, 56(11): 2555–2566, 2011.

[88] T. Lee, M. Leoky and N. H. McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425. IEEE, 2010.

[89] S. Bouabdallah. *Design and control of quadrotors with application to autonomous flying*. Ph.D. thesis, Ecole Polytechnique Federale de Lausanne, 2007.

[90] C. Nicol, C. Macnab and A. Ramirez-Serrano. Robust adaptive control of a quadrotor helicopter. *Mechatronics*, 21(6): 927–938, 2011.

[91] Z. T. Dydek, A. M. Annaswamy and E. Lavretsky. Adaptive control of quadrotor UAVs: A design trade study with flight evaluations. *IEEE Transactions on control systems technology*, 21(4): 1400–1406, 2013.

[92] D. Lee, H. J. Kim and S. Sastry. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. *International Journal of control, Automation and systems*, 7(3): 419–428, 2009.

[93] R. Xu and U. Ozguner. Sliding mode control of a quadrotor helicopter. *Decision and Control, 2006 45th IEEE Conference on*, pages 4957–4962. IEEE, 2006.

[94] S. Bouabdallah and R. Siegwart. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2247–2252. IEEE, 2005.

[95] I. González, S. Salazar and R. Lozano. Chattering-free sliding mode altitude control for a quad-rotor aircraft: Real-time application. *Journal of Intelligent & Robotic Systems*, 73(1): 137–155, 2014.

[96] J. Han. From PID to active disturbance rejection control. *IEEE transactions on Industrial Electronics*, 56(3): 900–906, 2009.

[97] Z. Gao. Active disturbance rejection control: a paradigm shift in feedback control system design. *2006 American control conference*, pages 7–pp. IEEE, 2006.

[98] B.-Z. Guo and Z.-l. Zhao. On the convergence of an extended state observer for nonlinear systems with uncertainty. *Systems & Control Letters*, 60(6): 420–430, 2011.

[99] D. J. Zhao and B. Y. Jiang. Robust attitude control of quadrotor vehicle via extended state observer. *Applied Mechanics and Materials*, volume 373, pages 1445–1448. Trans Tech Publ, 2013.

[100] K. Chang, Y. Xia, K. Huang and D. Ma. Obstacle avoidance and active disturbance rejection control for a quadrotor. *Neurocomputing*, 190: 60 – 69, 2016.

[101] X. Gong, Y. Tian, Y. Bai and C. Zhao. Trajectory tacking [sic] control of a quadrotor based on active disturbance rejection control. *2012 IEEE International Conference on Automation and Logistics*, pages 254–259. IEEE, 2012.

[102] J. Biggs. Optimal path planning for nonholonomic robotic systems via parametric optimisation. *Conference Towards Autonomous Robotic Systems*, pages 207–218. Springer, 2011.

[103] C. Maclean and J. D. Biggs. Path planning for simple wheeled robots: sub-Riemannian and elastic curves on SE(2). *Robotica*, 31(08): 1285–1297, 2013.

[104] V. Martinez. Modelling of the flight dynamics of a quadrotor helicopter. *A MSc Thesis in Cranfield University,*, 2007.

[105] M. Amir and V. Abbass. Modeling of quadrotor helicopter dynamics. *Smart Manufacturing Application, 2008. ICSMA 2008. International Conference on*, pages 100–105. 2008.

[106] C. Youngblood. Stingray 500, 2016. Accessed: 21-March-2017. Available at `http://www.curtisyoungblood.com/legacy-product-support-curtis-youngblood/attachment/stingray-500/`.

[107] M. J. Cutler. *Design and control of an autonomous variable-pitch quadrotor helicopter*. Ph.D. thesis, Citeseer, 2012.

[108] B. Michini, J. Redding, N. K. Ure, M. Cutler and J. P. How. Design and flight testing of an autonomous variable-pitch quadrotor. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2978–2979. IEEE, 2011.

[109] S. Sheng and C. Sun. Control and optimization of a variable-pitch quadrotor with minimum power consumption. *Energies*, 9(4): 232, 2016.

[110] Hobbico. Helimax: Voltage 500 3D, 2015. Accessed: 8-September-2016. Available at `http://www.helimaxrc.com/quadcopters/hmxe0864-voltage500/index.html`.

[111] J. Meyer. Hector quadrotor for ROS, 2017. Accessed 28-March-2017. Available at `http://wiki.ros.org/hector_quadrotor`.

[112] R. De Nardi. The qrsim quadrotors simulator. *RN*, 13(08): 08, 2013.

[113] DRL. Drone Racing League simulator, 2017. Accessed: 28-March-2017. Available at `https://thedroneracingleague.com/simulator/`.

[114] ImmersionRC. Liftoff Simulator, 2017. Accessed: 28-March-2017. Available at `https://www.immersionrc.com/fpv-products/liftoff-drone-race-simulator/`.

[115] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor and C. De Boor. *A practical guide to splines*, volume 27. Springer-Verlag New York, 1978.

[116] D. F. Rogers. *An introduction to NURBS: with historical perspective*. Elsevier, 2000.

[117] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3): 497–516, 1957.

[118] Y. Lin and S. Saripalli. Path planning using 3D Dubins curve for unmanned aerial vehicles. *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 296–304. IEEE, 2014.

[119] N. Mahmoudian and C. Woolsey. Underwater glider motion control. *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 552–557. IEEE, 2008.

[120] A. M. Bloch. *Nonholonomic mechanics and control*, volume 24. Springer, 2015.

[121] R. M. Murray and S. S. Sastry. Steering nonholonomic systems using sinusoids. *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, pages 2097–2101. IEEE, 1990.

[122] I. M. Ross and M. Karpenko. A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2): 182–197, 2012.

[123] J. Ostrowski, A. Lewis, R. Murray and J. Burdick. Nonholonomic mechanics and locomotion: the snakeboard example. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 2391–2397. IEEE, 1994.

[124] N. E. Leonard and P. S. Krishnaprasad. Motion control of drift-free, left-invariant systems on lie groups. *IEEE Transactions on Automatic control*, 40(9): 1539–1554, 1995.

[125] N. E. Leonard and P. Krishnaprasad. Averaging for attitude control and motion planning. *Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*, pages 3098–3104. IEEE, 1993.

[126] J. Wei and E. Norman. On global representations of the solutions of linear differential equations as a product of exponentials. *Proceedings of the American Mathematical Society*, 15(2): 327–334, 1964.

[127] V. Jurdjevic. *Geometric control theory*, volume 52. Cambridge University Press, 1997.

[128] H. J. Sussmann. An introduction to the coordinate-free maximum principle. *In Geometry of Feedback and Optimal Control, B. Jakubczyk and W. Respondek Eds., M*, pages 463–557. Dekker, Inc, 1997.

[129] D. Perrault, N. Bose, S. OYoung and C. D. Williams. Sensitivity of AUV added mass coefficients to variations in hull and control plane geometry. *Ocean engineering*, 30(5): 645–671, 2003.

[130] A. A. Neto, D. G. Macharet and M. F. M. Campos. Feasible RRT-based path planning using seventh order Bézier curves. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1445–1450. IEEE, 2010.

[131] J. Jamiesonn and J. Biggs. Path planning on SE(3) for AUVs using the RRT method. *IMA Conference on Mathematics of Robotics*. Institute of Advance Mathematics, 2015.

[132] D. Devaurs, T. Siméon and J. Cortés. Efficient sampling-based approaches to optimal path planning in complex cost spaces. *Algorithmic Foundations of Robotics XI*, pages 143–159. Springer, 2015.

[133] R. Hooke and T. A. Jeeves. "Direct Search" Solution of numerical and statistical problems. *J. ACM*, 8(2): 212–229, 1961.

[134] J. E. Bobrow, S. Dubowsky and J. Gibson. Time-optimal control of robotic manipulators along specified paths. *The international journal of robotics research*, 4(3): 3–17, 1985.

[135] G. Boyarko, M. Romano and O. Yakimenko. Time-optimal reorientation of a spacecraft using an inverse dynamics optimization method. *Journal of Guidance, Control, and Dynamics*, 34(4): 1197–1208, 2011.

[136] J. D. Biggs and L. Colley. Geometric attitude motion planning for spacecraft with pointing and actuator constraints. *Journal of Guidance, Control, and Dynamics*, pages 1–6, 2016.

[137] S. Dubowsky, N. Norri and Z. Shille. Time optimal trajectory planning for robotic manipulators with obstacle avoidance: a CAD approach. *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1906–1912. IEEE, 1986.

[138] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne and A. K. Cooke. A prototype of an autonomous controller for a quadrotor UAV. *Control Conference (ECC), 2007 European*, pages 4001–4008. IEEE, 2007.

[139] J. F. Epperson. On the Runge example. *Amer. Math. Monthly*, 94(4): 329–341, 1987.

[140] Q. Zheng, L. Q. Gaol and Z. Gao. On stability analysis of active disturbance rejection control for nonlinear time-varying plants with unknown dynamics. *Decision and Control, 2007 46th IEEE Conference on*, pages 3501–3506. IEEE, 2007.

[141] Y. Bai, J. D. Biggs, X. Wang and N. Cui. A singular adaptive attitude control with active disturbance rejection. *European Journal of Control*, 35: 50–56, 2017.

[142] E. Cheynet. Wind field simulation, 2016. Accessed: 2-June-2017. Available at `https://uk.mathworks.com/matlabcentral/fileexchange/50041-wind-field-simulation`.

[143] M. Shinozuka. Monte Carlo solution of structural dynamics. *Computers & Structures*, 2(5): 855–874, 1972.

[144] G. Deodatis. Simulation of ergodic multivariate stochastic processes. *Journal of engineering mechanics*, 122(8): 778–787, 1996.

[145] A. Aboudonia, R. Rashad and A. El-Badawy. Time domain disturbance observer based control of a quadrotor unmanned aerial vehicle. *Information, Communication and Automation Technologies (ICAT), 2015 XXV International Conference on*, pages 1–6. IEEE, 2015.