CRANFIELD UNIVERSITY

Ian Cowling

Towards Autonomy of a Quadrotor UAV

SCHOOL OF ENGINEERING

PhD THESIS

CRANFIELD UNIVERSITY

SCHOOL OF ENGINEERING

PhD THESIS

Ian Cowling

Towards Autonomy of a Quadrotor UAV

Supervisor: Dr J.F. Whidborne

October 2008

Abstract

As the potential of unmanned aerial vehicles rapidly increases, there is a growing interest in rotary vehicles as well as fixed wing. The quadrotor is small agile rotary vehicle controlled by variable speed prop rotors. With no need for a swash plate the vehicle is low cost as well as dynamically simple.

In order to achieve autonomous flight, any potential control algorithm must include trajectory generation and trajectory following. Trajectory generation can be done using direct or indirect methods. Indirect methods provide an optimal solution but are hard to solve for anything other than the simplest of cases. Direct methods in comparison are often sub-optimal but can be applied to a wider range of problems. Trajectory optimization is typically performed within the control space, however, by posing the problem in the output space, the problem can be simplified. Differential flatness is a property of some dynamical systems which allows dynamic inversion and hence, output space optimization.

Trajectory following can be achieved through any number of linear control techniques, this is demonstrated whereby a single trajectory is followed using LQR, this scheme is limited however, as the vehicle is unable to adapt to environmental changes. Model based predictive control guarantees constraint satisfaction at every time step, this however is time consuming and therefore, a combined controller is proposed benefiting from the adaptable nature of MBPC and the robustness and simplicity of LQR control.

There are numerous direct methods for trajectory optimization both in the output and control space. Taranenko's direct method has a number of benefits over other techniques, including the use of a virtual argument, which separates the optimal path and the speed problem. This enables the algorithm to solve the optimal time problem, the optimal fuel problem or a combination of the two, without a deviation from the optimal path.

In order to implement such a control scheme, the issues of feedback, communication and control action computation, require consideration. This work discusses the issues with instrumentation and communication encountered when developing the control system and provides open loop test results.

This work also extends the proposed control schemes to consider the problem of multiple vehicle flight rendezvous. Specifically the problem of rendezvous when there is no communication link, limited visibility and no agreed rendezvous point. Using Taranenko's direct method multiple vehicle rendezvous is simulated.

Acknowledgements

I need to thank many people who I have relied upon to complete this work. Despite the single name on the front of this work, it is by no means an individual effort.

Academically this work has been greatly influenced by two people. The first is Oleg Yakimenko from NPS California, who has provided his ideas, knowledge and expertise. He did exactly as he promised, the first time I met him he told me "18 months should be enough time to sort you out." A massive thank you has to go to James Whidborne, getting me to this stage has probably been as hard for him as it has been for me. His knowledge, vision and above all patience has been invaluable to me during this time.

I must thank the guys in my office for keeping me sane and providing plenty of distractions, especially during the summer months. Thank you to my old house mates, Giuseppe Zumpano, Frank Noppel and Ewan McAdam for sharing two very strange but good years of my life. I must also thank the people in my group at Cranfield, past and present, for interesting discussions and introducing me to the world of aerospace engineering. Special thanks should go to Alastair Cooke who was instrumental in the purchase and modelling of the quadrotor and to Barry Shaw and Vicente Martinez for their efforts in the testing and modelling of the quadrotor. I should also thank Sascha Erbsloeh for his help in the development of the experimental set up described in this work and for his strange German sense of humour. Thank you to Phill Smith and everyone at Blue Bear Systems Research for their understanding and for giving me some time to complete this work.

Of course I would not have even started at Cranfield had it not been for 21 years of support and encouragement from my family. Thank you to my 'little' brothers, Mark and Andrew, for their support and being such good role models. Of course a big thank you to my Mum and Dad, to whom I owe a great deal. My Dad for his never ending ('over') confidence in my abilities and to my Mum for doing all the worrying and sleepless nights for me.

Finally the biggest thank you has to go to my wife Lucy. Against her better judgment she agreed to marry an engineer, one studying for a PhD has really been a test for her. Thank you for your understanding over the last year, both when I have spent evenings writing this thesis and also for when I am in a world of my own. At least the thesis writing has finished.

Contents

List of Figures

List of Tables

Acronyms

DOF	Degree Of Freedom
DC	Direct Current
DCM	Direct consequence measurement
DoD	Department of Defense
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LQ	Linear Quadratic
LQR	Linear Quadratic Regulator
MAD	Matlab Automatic Differentiation
MEMS	Micro-Electro-Mechanical Systems
MBPC	Model Based Predictive Control
MILP	Mixed Integer Linear Programming
PD	Proportional and Derivative
PID	Proportional Integral and Derivative
PRM	Probabilistic Road Map
PWM	Pulse Width Modulated
RC	Radio control
RTOS	Real Time Operating System
SAM	Surface to Air Missle
SQP	Sequential Quadratic Programming
TTL	Transistor Transistor Logic
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take Off and Landing

Latin letters

Symbol	Unit	Quantity
A		Linear State Matrix
A_d		Disc Area
A_m		Force Moment Vector

B	Linear Control Matrix
C	Linear State Matrix
$\mathcal{C}$	Controllability Matrix
C_d	Drag coefficient
C_T	Thrust to torque conversion
D	Linear Control Matrix
D_{Ob}	Distance to center of obstacle
D_t	Total distance between all vehicles
F_B	Body axis forces
I	Inertia matrix
I_x	Inertia about x axis
I_y	Inertia about y axis
I_z	Inertia about z axis
J	Performance measure
K_c	Gain matrix
L	Euler rates to body rates conversion
L_p	Propulsive Moment
M	Order of basis function
M_m	Mass Matrix
M_p	Propulsive Moment
N	Number of nodes
N_p	Propulsive Moment
P_i	Power required per motor
P_n	Weighting function
$P(t)$	Parameterized function of time
$P(\tau)$	Parameterized function of virtual argument
Q	Weighting matrix
Q_i	Torque from the ith rotor
$\mathbb{R}$	Real number
R	Weighting matrix
R_b	Radius of blade
R_s	Radius of obstacle
R_x	Rotation matrix about x axis
R_y	Rotation matrix about y axis
R_z	Rotation matrix about z axis
$\mathcal{S}_c$	Stability set
T	Time horizon

T_i	Thrust from the *i*th motor
T_k	Basis function
T_{total}	Predetermined flight time
V	Speed profile
V_i	Induced Velocity
V_c	Vertical speed
V_n	Nth Vehicle
$V(\tau)$	Speed in the virtual domain
X	Position of obstacle the *x* axis
X_g	Gravitational forces along the *x* axis
X_p	Propulsive forces along the *x* axis
Y	Position of obstacle the *y* axis
Y_g	Gravitational forces along the *y* axis
Y_p	Propulsive forces along the *y* axis
Z	Position of obstacle the *z* axis
Z_g	Gravitational forces along the *z* axis
Z_p	Propulsive forces along the *z* axis
a_k	Element of free variable (λ)
b	Number of blades
c	Chord of blades
c_n	Parameterization coefficient
d	Distance between vehicles
g	Gravitational constant
l	Length of arm
m	Mass
p	Body rate about *x* axis
q	Body rate about *y* axis
r	Body rate about *z* axis
r_s	Stability set radius
s_j	Distance traveled between nodes (j and $j-1$)
t	Time
t_f	Final time
t_k	Time step
u	Velocity in body axis
$\mathbf{u}$	Control vector
$\mathbf{u_0}$	Initial control vector

u_n	Control Input
$\tilde{u}_n$	Actuator output
$\mathbf{u_{ref}}$	Reference control vector
$\mathbf{u_T}$	Terminal control vector
v	Velocity in body axis
$v(n)$	The nth Vehicle
v_i	Voltage
v_n	Automatic differentiation program
w	Velocity in body axis
$\mathbf{w}$	Weighting function
w_x	Weighting factor
w_y	Weighting factor
x	Position in earth axis
x_{error}	State error
$\hat{x}$	Parameterized position in earth axis
$\hat{x}_T$	Terminal parameterized position in earth axis
$\mathbf{x}$	State vector
$\mathbf{x}_{ref}$	Reference state vector
$\mathbf{x}_T$	Terminal state
$\mathbf{x}_0$	Initial state vector
y	Position in earth axis
$\hat{y}$	Parameterized position in earth axis
$\hat{y}_T$	Terminal parameterized position in earth axis
$\mathbf{y}$	Output vector
$\mathbf{y}_T$	Terminal output vector
z	Position in earth axis
$\hat{z}$	Parameterized position in earth axis
$\hat{z}_T$	Terminal parameterized position in earth axis

Greek letters

Symbol	Unit	Quantity

Φ	Cost function
Φ_B	Bolza cost function
Φ_L	Lagrange cost function
Φ_M	Mayer cost function
Γ	Basis function
Ω	Cross product matrix
Ω_i	Speed of *i*th rotor
α	Spectral abscissa
δv_n	Differentiated program
ϕ	Euler angle about *x* axis from R_{zxy}
$\hat{\phi}$	Euler angle about *x* axis from R_{xyz}
γ	Air density
η	Sampling time
λ	Speed factor
λ_n	Free variable
λ_{opt}	Optimal reference trajectory
θ	Euler angle about *y* axis from R_{zxy}
$\hat{\theta}$	Euler angle about *y* axis from R_{xyz}
ρ	Air density
τ	Virtual argument
τ_f	Terminal virtual argument
τ_n	Thrust from *i*th rotor
υ	Torque from *i*th rotor
ψ	Euler angle about *z* axis from R_{zxy}
$\hat{\psi}$	Euler angle about *z* axis from R_{xyz}
$\hat{\psi}$	Parameterized output

Chapter 1

Introduction

An Unmanned Air Vehicle (UAV) can be defined as (Newcome 2004) "A powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable and can carry a lethal or non lethal payload." Unmanned vehicles have been around for a considerable amount of time. In 1917 Elmer Sperry demonstrated his sea plane which was gyrostabilized and capable of navigating itself (Newcome 2004). In recent times UAV capability has increased considerably and unmanned flights are being developed for a range of military operations eg. (Gibbs 2005) as well as other specific tasks such as power line inspection (Sinha *et al.* 2006) and coast guard operations eg. (O'Donnell and Dorwar 2006). The next section discusses just a couple of many potential applications for UAVs.

There is also interest in smaller UAVs and especially UAVs capable of internal flight. For internal flight the demands are obviously very different, with agility, low speed and safety coming to the fore. There are a number of different airframes being researched for internal flight; the three major ones being low speed fixed wing vehicles, flapping winged vehicles and rotorcraft such as the quadrotor. Low speed fixed wing vehicles are simple dynamically but are not as agile as rotorcraft because minimum velocity must be maintained. Flapping-wing UAVs are capable of hover but are dynamically complex and still in the very early stages of development. The quadrotor is a small four-rotor helicopter, two examples of which are shown in Figure (1.1) and Figure (1.2). Controlled only by the speed of the four rotors, it possesses relatively simple dynamics. Its light weight and agility make it suitable for internal flight as the propeller rotors can be replaced by ducted fans. There has been extensive research into the quadrotor with its simple dynamics making it an excellent testbed for advanced control techniques. Section 1.2 discusses the quadrotor and potential applications for such a vehicle.

Autonomy when related to an aerial vehicle implies the vehicle is not controlled by others or an outside force. There are, therefore, many aspects to consider in order to achieve autonomous flight such as path planning, trajectory planning, trajectory following, health diagnosis, adaption to different flight conditions and state feedback. This work will consider trajectory generation and trajectory following in depth. Path planning is the determi-

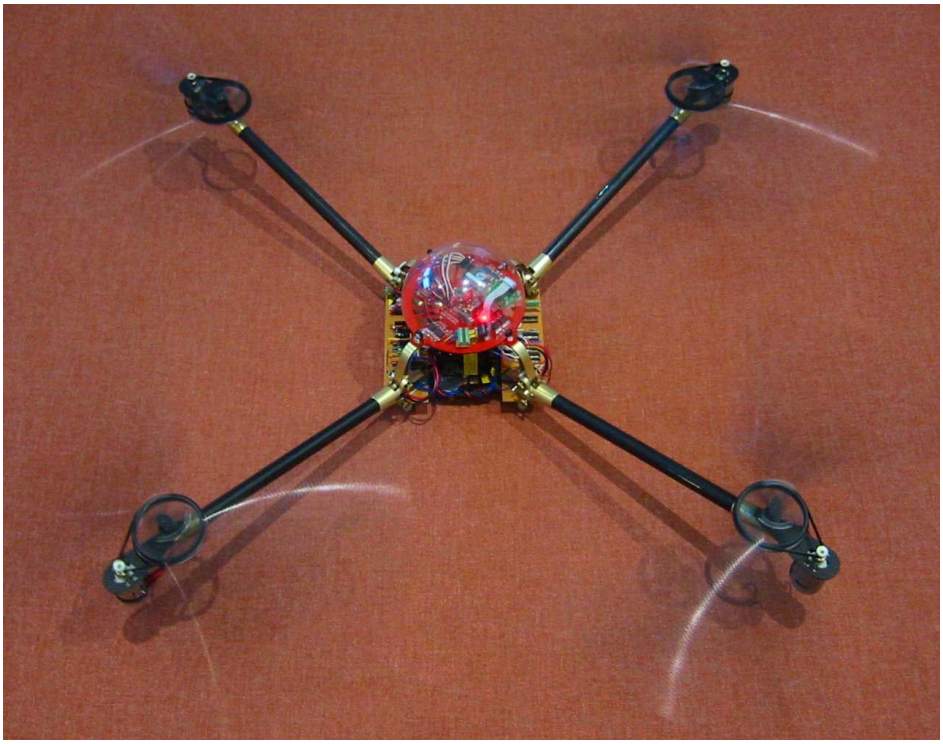


Figure 1.1: Draganflyer X-Pro at Cranfield

Figure 1.2: Draganflyer V at Cranfield

nation of a feasible and ideally an optimal flight path to reach the destination. Trajectory generation is the determination of an optimal path which is time dependent, this may also include the determination of all the vehicle's states over this time. For full autonomy a trajectory generator should be capable of running in real time and ideally on board. Trajectory optimization should also be capable of running in conjunction with a mission planner, which may be a path planning algorithm such as Probabilistic Road Maps (PRM) (Pettersson and Doherty 2004), or independently for example within a visual field. Trajectory generation will be discussed further within Section 1.5.

Once a trajectory has been determined it is obviously necessary to follow that trajectory. To do this there are many potential control schemes both linear and non-linear. The quadrotor itself has been a testbed for many different control techniques as its dynamic simplicity makes it ideally suited to the testing of advanced controllers. Trajectory following will be further discussed in Section 1.6.

There are many instances in which it could be beneficial to deploy multiple vehicles as opposed to a solitary vehicle. Advantages of multiple vehicles include; increased and improved surveillance area, increased payload and reduced risk due to a degree of vehicle redundancy, as single vehicle failure does not necessarily result in mission failure. Multiple vehicle literature is extensive and there are many potential strategies for multi-vehicle co-operation both in centralized and decentralized situations. This work will therefore look at a specific problem, to demonstrate the capabilities of the quadrotor, as well as the control schemes presented. This multiple vehicle problem will be discussed further within Section 1.7.

As discussed, trajectory generation will ideally be capable of running in real time and on board, however, the solving of a non-linear constrained optimization is usually a computationally demanding process. It is necessary therefore, to consider techniques which reduce computation time in order to achieve real-time optimization. Analysis of the trajectory generation algorithm shows that a large percentage of the computation time is used to calculate the gradients of the constraints and cost function. Automatic differentiation is a computational technique for providing these gradients that has potential for reducing computational time. This will be reviewed in depth in Section 1.8.

Once a control algorithm has been developed, issues such as feedback, instrumentation and communication need to be considered, before it can be implemented. There are many potential approaches to designing such a system with emerging technology providing low cost and lightweight alternatives to existing technology. The requirements and potential designs will be reviewed in Section 1.9

This introduction will conclude with an outline of the thesis, publications resulting from this work and contributions made within this work.

1.1 Potential Applications for UAVs

The list of potential UAV applications is endless and would include numerous surveillance, search and rescue applications as well as specific tasks such as fire fighting. Generally, UAV application is envisaged within the '3D's' environment which refers to dangerous, dull or dirty environments; in other words for occasions where it is not desirable to use a human pilot. There are of course other examples. For instance it may be cost effective in some cases to deploy UAV's for coastguard patrolling (O'Donnell and Dorwar 2006). This work is not intended to develop the quadrotor for a specific application, however, it is evident there are many potential applications, many of which will have general requirements of such a system. Therefore, it is useful to consider these requirements of the general case before considering the control system design.

Unmanned air vehicles have many potential applications but search and rescue in a hazardous environment is arguably the most likely. Rescue robots were first used in 2001 for the rescue mission in the World Trade Center disaster (Murphy 2005). 3 years later they were deployed but not used. for the rescue mission after hurricane Charley, highlighting the need for "other robot modalities (air,water)" (Murphy 2005). Rotorcraft have been used for real surveilance and rescue missions as discussed within Murphy and Stover (2006) and Murphy *et al.* (2005). Typically each mission lasts around 15 minutes and is performed within a visual field. There are also many potential indoor applications such as internal factory inspection, reconnaisance within an urban environment and observation of a structually unsafe building. At a later point within this chapter there will be a review of way point or global generation schemes such as the MILP scheme (Richards and How 2002) and roadmaps (Kavaraki *et al.* 1996). These schemes consider the global trajectory or path planning problem in which there may be several obstacles or ground threats to the vehicle. For the missions discussed here it is more likely that a short trajectory will be required, to peer around a corner for example. Such trajectories are frequently critically dependent upon time. This is a different problem to the one considered within Richards and How (2002) and Kavaraki *et al.* (1996) and therefore requires a different approach.

One potential application for an unmanned air vehicle is landmine detection and removal. Current techniques are slow, tedious and present significant threats to human life (Harris 2000). There is therefore a significant interest in a UAV capability to scan an area to detect potential landmines. Within Rajasekharan and Kambhampati (2003) some requirements are presented to describe aspects of any proposed vehicle, these have been widely accepted by researchers within the area. This list provides a useful design criteria for UAV development, which arguably, the quadrotor would potentially fit very well.

1. Low cost, lightweight and high mobility.
2. Control and communication. At least remote control or semi autonomous.
3. Sensor integration.
4. All-terrain robot.
5. Simplicity of operation, should not require extensive training to use.

The requirement for the system to be low cost is extremely important. Referring again to the 3 D's, it is unlikely that a hugely expensive UAV will be deployed into a potentially dangerous environment. The second point "at least semi autonomous" and the final point "should not require extensive training" are in many ways synonymous and this will be discussed latter within Section 1.3. However, the higher the degree of autonomy and the subsequent reduced workload on the operator, are extremely important factors within UAV development. Finally, sensor integration should be by no means a final thought, after all in the majority of situations the sensor is the reason for the UAV development in the first place, this has led to a common referal to UAV's as Unmanned Air Systems.

1.2 The Quadrotor

The quadrotors shown in Figure (1.1) and Figure (1.2) consists of a central unit housing a lithium polymer battery with 4 equal length arms supporting prop rotors which are powered by DC motors. The quadrotor is available off the shelf and the Draganflyer V is the most commonly used vehicle (RCToys 2008). It comes with 3 gyroscopes on the central unit, providing some basic stabilization for the vehicle. The vehicle is controlled using a 4-channel radio control which controls thrust, roll pitch and yaw. These vehicles typically range from around 0.5m (Figure 1.2) to 1m (Figure 1.1). There have been attempts to build in-house quadrotors (Pounds *et al.* 2002), generally though, the dynamics for all the different models remain the same. There are some exceptions, however, such as Dodd (2007) where the rotors can be tilted to achieve translational motion without rotating the vehicle. Generally, the 4 rotors control the 6 degrees of freedom, therefore, making the system under-actuated, although a certain amount of decoupling of the control inputs makes the control laws relatively simple, especially at fixed angles of yaw (Madani and Benallegue 2006). As the vehicle is controlled only by varying the speed of the 4 rotors, the vehicle possess relatively simple dynamics and is, therefore, an excellent testbed for advanced control techniques, which is evident through the diversity of publications on the subject. The advantages of such a vehicle, other than its dynamic simplicity include i.) its agility, as it is essentially omni-directional and ii.) the rotors are at fixed angles of pitch, making the construction of a quadrotor simple as well as cheap.

As discussed, the quadrotor is available off the shelf, but the Draganflyer range is not capable of closed loop guidance control without modification to achieve feedback, determination of the control action and actuator control. Instrumentation of the quadrotor is a relatively challenging problem because these models have a limited payload which prevents the addition of numerous additional sensors. Potential experimental setups will be discussed further within Section 1.9.

There are many different institutions and control groups looking at the quadrotor. A major contributor to quadrotor literature is Castillo *et al.* (2005). This group, based in Compiegne, France, has worked on backstepping with nested saturations, for real-time control of the quadrotor, for take-off, hover and landing (Kendoul *et al.* 2006). In Castillo *et al.* (2004) these are used to stabilize the quadrotor and this was possibly the first paper to demonstrate successful control of the quadrotor. In Castillo *et al.* (2006) this nested

saturation approach is compared with classical PD control, the benefit of the nested saturations algorithms is shown to be an improved handling of disturbances.

There is also significant quadrotor research from the Swiss Federal Institute (Bouabdallah *et al.* 2004*b*), where there is also interest in backstepping for stabilization, which in Bouabdallah and Siegwart (2005) is compared with sliding mode control. These are compared on a fixed rig allowing the quadrotor two degrees of freedom and also through simulation. In Bouabdallah *et al.* (2004*a*), PID and LQ control are compared in the same way. Again these results show the excellent disturbance rejection from applying backstepping control, although all four control schemes perform reasonably well.

Sliding mode control, backstepping and dynamic feedback for stabilization and simple trajectory tracking have also been considered by the Robotics Laboratory of Versailles (Mokhtari *et al.* 2006, Madani and Benallegue 2006, Mokhtari and Benallegue 2004). It was found that the dynamic feedback control is insufficient to cope with significant external disturbances, the sliding mode controller proved to be much better in simulation. The backstepping controller performed well in simulation and in a 2 degree of freedom experiment.

There appears to be a trend within these papers suggesting that traditional linear controllers are not sufficient for stabilization of the quadrotor, however, there are papers to the contrary. In Tayebi and McGilvray (2004), two different PD controllers are proposed. The first with compensation of the Coriolis and gyroscopic torques, the second without. These are shown to be effective in simulation and in Tayebi and McGilvray (2006) through experimentation again on a 2 degree of freedom test rig. The benefit of the compensation of the Coriolis and gyroscopic torques is not evident in the results, this is thought to be due to the relatively low speed, for higher speeds this may be significant.

Other examples of trajectory following controllers include Chen and Huzmezan (2003*b*), where H_∞ is considered. The results through simulation show good tracking as well as good disturbance rejection. In Driessen and Robin (2004) a two phase controller is proposed in which the first phase considers attitude and the second phase position and yaw angle. Using differential flatness, global convergence of the tracking error to zero is proven assuming positive thrust.

There are considerably fewer trajectory generation papers for the quadrotor. In Beji and Abichou (2005) output space trajectory generation is considered using differential flatness. This paper only considers straight line trajectories using a simple parameterization and is, therefore, not considering optimality and is assuming feasibility of the straight line.

Now, the papers reviewed so far are usually based on simulation results with a number also including results from a 2 degree of freedom experimental rig. The main reason for this is the problem with determining the position of the quadrotor in flight as GPS is not available indoors. There are a number of alternatives for solving this problem. In Valenti *et al.* (2006) the testbed at MIT is described. Consisting of Vicon ground cameras and a number of quadrotors with visual markers, this system provides accurate attitude and position feedback to a ground station, however, the system is certainly not cheap and does

not offer a commercial solution.

At Stanford University a multiple vehicle testbed is used for comparison of design control techniques (Waslander *et al.* 2005). This testbed with a capacity for up to 8 vehicles (Hoffmann *et al.* 2004) is only operational outdoors however as it relies on GPS.

A cheaper solution which has been considered by a number of institutions is that of visual feedback. In Altug *et al.* (2002) and Altug and Taylor (2004) two cameras are used for POSE (Position and Orientation) estimation, one camera is onboard the vehicle and the second is a ground based pan tilt camera. This set up is used to compare nonlinear control techniques. However this is not a complete solution as noted in Altug *et al.* (2002) 'A helicopter can not be fully autonomous if it depends on an external camera.' In Chitrakaran *et al.* (2006) a single onboard camera system is proposed which tracks a moving ground vehicle. Finally in Hamel *et al.* (2002) a circular trajectory is followed, using a positional estimate, obtained through an onboard camera, viewing a square target on the ground.

1.3 Autonomy

Autonomy or an autonomous vehicle are widely used phrases, of which the meaning can vary from author to author. The literal sense of the word would imply that there is absolutely no human interaction with the vehicle but this is rarely the case as human interaction is commonly required for take-off and landing as well as refueling. As advances in UAV technology enable a potentially greater degree of autonomy, the question of what level of autonomy is actually required demands increasing consideration. In the same way it is likely that commercial aircraft will continue to be controlled by highly qualified pilots, UAV's are likely to operate with a human in the loop. Safety issues are obviously a driving factor for this but also public perception is equally important. Recently Merseyside police announced plans to trial quadrotors for tracking criminals and recording antisocial behavior. This was met with a public outcry regarding an apparent 'erosion of civil liberties' (Telegraph 2007). Hence consideration of social impact of autonomous vehicle deployment is important.

Regardless of the required level of autonomy there are, as always, extreme examples. One such example can be found in Ieropoulos *et al.* (2004) where a robot searches the surrounding area for slugs which it can use as a bio-fuel. A less extreme solution for recharging of a UAV could potentially use solar cells to remove the need for any human interaction. Typically though a much lower degree of autonomy is required and is necessary for safety and public acceptance. Figure 1.3 plots the development on UAV's, from 1955 through to the current day, as well as the predicted development up to 2025, against a quantified degree of autonomy (DoD 2005). In the $1980's$, UAV's were capable of remotely guided flight, in the $1990's$ technological advances enabled vehicles such as Global Hawk to have real-time health diagnosis capabilities. As technological advances accelerate, it is predicted that UAV's will soon have the capabilities for onboard route planning, as well as the development of multiple vehicle algorithms, this could be developed to achieve multiple vehicles cooperation, enabling individual members to operate

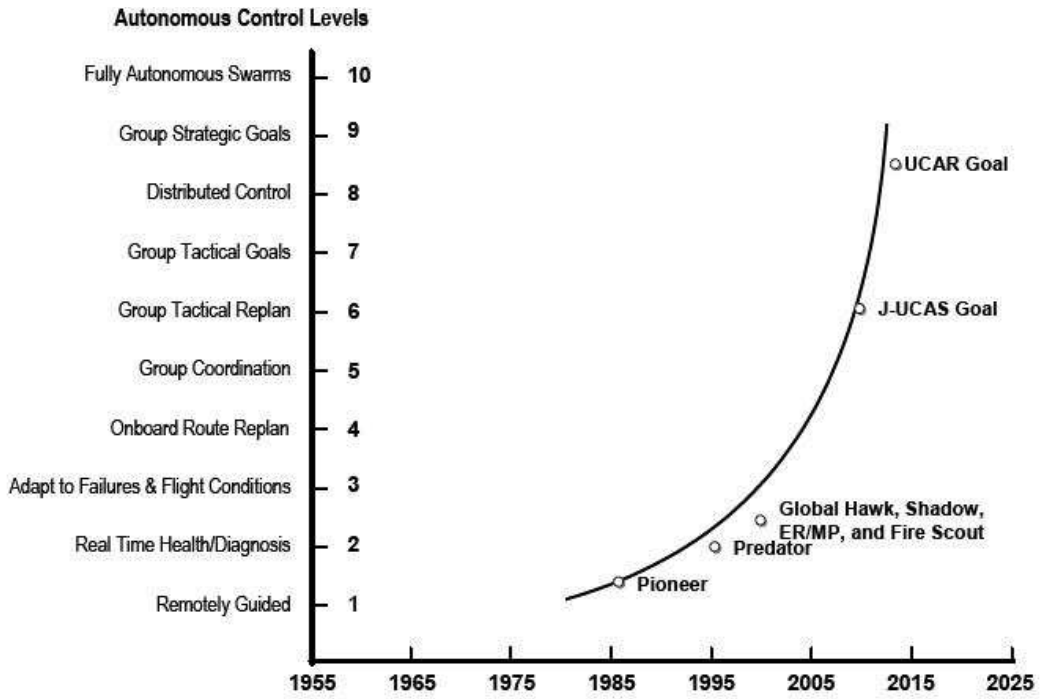


Figure 1.3: DoD Roadmap to autonomy, (DoD, 2005)

within fully autonomous swarms. Figure 1.3 splits autonomous control into 10 levels from the most basic which is remotely piloted to level 10 which is fully autonomous swarms. In this context this work looks at how a remotely piloted vehicle can be developed up to level 4 capability, this includes the ability to adapt to flight conditions and onboard trajectory planning.

The degree of autonomy is not just dependent on capability. In many cases the level of autonomy is constrained by safety regulations such as within controlled air space where the flight plan must be submitted pre-flight. Figure 1.4 shows a basic hierarchical control structure for UAV operation, however, the precise content of each box depends on the degree of autonomy of the system. The top level of the system is the operator who sets out the mission objectives, these may be specific waypoints but may be more general such area surveillance. For shorter missions the operator may simply set some terminal conditions. The next level is a mission planner, this would be required for longer missions where there are more than a single set of waypoints. The mission planner is essentially what is also know as a navigational algorithm. The role of such an algorithm is to solve a path planning problem to determine a sequential list of waypoints. The trajectory generation would determine a feasible and ideally but not necessarily optimal trajectory to get between waypoints or to the final destination. This differs from a path planning algorithm as the resulting trajectory is in 3 dimensions and a function of time. Trajectory generation could be referred to as a sophisticated guidance algorithm. Finally a trajectory follower would ensure that the vehicle follows the reference trajectory generated by the trajectory generation algorithm. This is essentially the control law within the hierarchical structure.

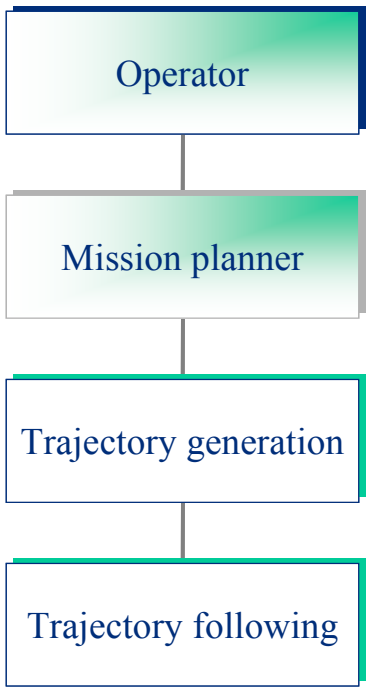


Figure 1.4: Hierarchial Controller

Ideally, out of regulated air space this control scheme would operate on board as this would enable re-optimization and redefining of the mission plan if required. In the case where the flight is within controlled airspace and the flight plan is submitted pre-flight, the trajectory planner could be implemented as part of a sense and avoid scheme, in which the vehicle may need to leave the pre-determined flight path momentarily to avoid collision, in this case the objective is to rejoin the original flight path after deviating from this path.

Figure 1.4 shows how the hierarchical structure can be built into a UAV such as the quadrotor. The mission planner determines some initial and terminal states which are fed forward to the trajectory generation algorithm, essentially this is an instruction to get from one point to another. The trajectory generation then optimizes a reference trajectory to get between these points which is then used to calculate a state error signal using full state feedback, this error is then fed into the trajectory following algorithm. The trajectory following algorithm then calculates a demanded actuator output based on this signal. Built into the quadrotor is some degree of gyroscopic stabilization, this forms an inner loop stabilizer although this is not sufficient for hands-free flight. The actual signals sent to the motors are therefore a combination of the trajectory following algorithm and this inner loop stabilization.

1.4 Path planners

There are numerous path planning algorithms, these schemes provide effective path planning schemes which are suitable, for example, within the mission planner in Figure 1.4.

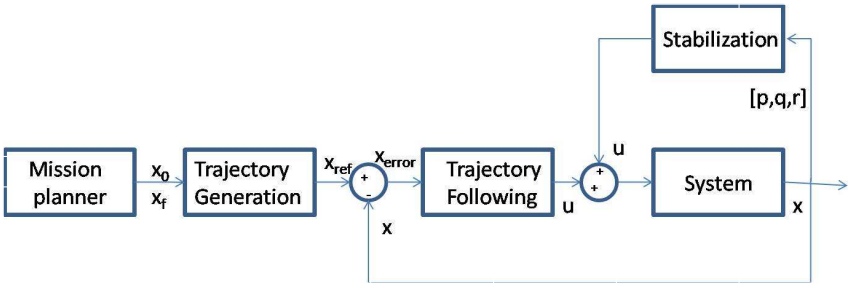


Figure 1.5: Hierarchical controller

Path planners are typically split into two categories; local and global planners (Kamal *et al.* 2005). Global planners require global knowledge beforehand and therefore any environmental changes such as a new obstacle or moving obstacle will require a reoptimization of the path. Local planners are computationally less demanding than global planners and, therefore, more likely to be able to be run on board. However, consideration of the local problem, may lead to a globally sub-optimal solution and possibly a globally infeasible flight path. As computational demand is of great importance and the globally infeasible solutions arising from local planners are to be avoided, alternatives have been considered. In Pettersson and Doherty (2004) an initial global planner is responsible for determining a flight path for the entire flight offline. An online local linear planner is then used to pass through the way points generated but also to avoid any potential collision. An alternative approach is shown in Kamal *et al.* (2005) where a local planner is used to determine a trajectory and in the event of the trajectory becoming infeasible at a future time the constraints are softened to find a new optimal path.

1.4.1 Mission planners and global planners

Roadmaps, sometimes referred to as skeletons, reflect the geometric structure of the environment in a similar way to how a skeleton reflects the geometry of a body. In the likely absence of a physical roadmap these are constructed using some probabilistic approach in which a quantative analysis of the environment provides some structure of potential or optimal pathways. Probabilistic roadmap (PRM) algorithms developed by (Kavaraki *et al.* 1996) and adapted by (Pettersson and Doherty 2004) provide an excellent example of PRM capabilities. The roadmap within Pettersson and Doherty (2004) is generated using a world model offline to generate a collision free path. The roadmap is then linearized and an A* search is performed to determine a linear path which is then smoothed or replaced with splines using a local planner. An A* algorithm performs a search within a nodal space by estimating and ranking the nodes in terms of the estimated best path through that node, it then visits the nodes in that order starting with the best estimate and is hence called a ‘best first’ algorithm.

It is likely that any global planner will be constrained. In an urban environment there are obviously buildings which must be avoided. In regulated airspace there are specified

flight paths which must be followed. In a combat situation it is likely that the vehicle will be required to fly the safest route, in this case it is constrained to fly a safe distance away from threats. In Gu *et al.* (2004), Voronoi graphs are constructed by evaluating the hit probability over an area of a number of SAM (surface to air missiles) sites. Voronoi polygons are constructed with the important property that along the edge of any polygon is the maximum distance from the perceived threat. This therefore constructs a network or roadmap of the potential paths. The optimal path through the region is then chosen using a cost function which sums the estimated risk cost and the fuel cost to fly along the path.

A common approach to path generation is Mixed Integer Linear Programming (MILP) (Richards and How 2002). This technique, in a similar way to PRM, provides a global optimal trajectory, typically through an multi-obstacle terrain such as an urban environment with many buildings. This scheme presents and solves the problem using linear programming which provides efficient route planning for a UAV especially in a complex environment where, for example, there are many buildings or threats to the UAV such as ground to air missiles. Formulating the exact problem and finding the optimal path using MILP is computationally demanding and, thus therefore, in (Kim *et al.* 2007) alternative formulations are proposed which guarantee a solution (albeit sub-optimal) close to real time.

1.5 Trajectory Generation

Trajectory optimization is the determination of a feasible and optimal time dependent path to the desired location. In order for the trajectory to be feasible it must consider dynamic constraints of the vehicle and actuator constraints. Potential dangers must also be considered such as collision with obstacles, other vehicles and in some cases threats. The algorithm must also be computationally efficient to be run on board as well as able to run in real time.

Trajectory optimization can be split into *Direct Methods* and *Indirect Methods* (von Stryk and Bulirsch 1992), an excellent review of both can be found in Betts (1998). “Direct methods can be described as discretizing the optimal control problem and then solving the resulting non-linear problem.” (Fahroo and Ross 2000). Indirect methods “rely on solving the necessary conditions derived from Pontryagin’s minimum principle” (Fahroo and Ross 2000). Direct methods are increasingly being applied to the solving of optimal control problems as the indirect methods are “very difficult to solve for all but the simplest of problems” (Conway and Larson 1998). The indirect method does have an advantage however, as the accuracy is better than for the direct method and in some cases a hybrid of the two is suggested (von Stryk and Bulirsch 1992). Typically though the direct method is preferred despite the reduced accuracy especially when considering real-time applications (Kumar and Seywald 1996*a*).

There are numerous approaches to transforming an optimal control problem into a non-linear programming problem within the control or state space using the direct method

(Hull 1997). The two most commonly used of these are the collocation method (Herman and Conway 1996, Enright and Conway 1992) and the pseudospectral method (Elnagar *et al.* 1995). The collocation method involves the reduction of the solution to a finite dimensional problem, some points are chosen within this space (collocation points) and the solution is found which solves the differential equation at these points (Hargraves and Paris 1987). The pseudospectral method is the "expanding of the underlying functions over a set of interpolating polynomials which interpolate these functions at specific nodes" (Fahroo and Ross 2002).

A common problem when solving the parameterized optimal control problem occurs when the data is non-smooth or as noted within Fahroo and Ross (2004) that even smooth data can have a non-smooth solution. In these cases pseudospectral methods exhibit Gibbs phenomenon. The Gibbs phenomenon arises from the approximation of a discontinuous function with a finite sum of continuous functions, this results in oscillations which do not die out as frequency increases. This is on the whole due to the predetermined distribution of the discretization points. Within Fahroo and Ross (2004) a knotting method is introduced to remove these problems by varying the discretization points.

Differential inclusion reduces the size of the parameter optimization problem by removing the bounded control from the optimizations, instead the states are bounded to sets of attainable time dependent states (Fahroo and Ross 2001, Seywald 1994). This removes the time histories of the controls so that the number of variables within the non-linear program is reduced, as well as the determination of the analytical gradients (Kumar and Seywald 1996*b*). A disadvantage of this approach, as argued by Conway and Larson (1998), is that the implicit nonlinear constraints are required with the lowest possible order of accuracy and, therefore, the problem is in fact larger than the collocation method which uses more sophisticated implicit non-linear constraints.

An alternative approach is the transformation of the optimal control problem into the output space and the evaluation of the objective function and constraints within the state and control space through dynamic inversion (Lane and Stengel 1988, Lu 1993, Sentoh and Bryson 1992). A dynamic system which is dynamically invertible can also be termed to be differentially flat (Fleiss *et al.* 1992).

1.5.1 Differential flatness

Differential flatness is a property of some dynamical systems which enables the expression of the state and control vectors as a function of the output function and it derivatives (Fleiss *et al.* 1992). This one-to-one relationship enables output space trajectory optimization as opposed to control space. The benefit of this approach is the simplification of the optimization problem when the majority of the optimization constraints occur within the output space. For a system to be differentially flat and therefore possessing a flat output (y) it requires (Chelouah 1997) a set of variables such that;

- The components of y are not differentially related over $\Re$;

- Every system variable may be expressed as a function of the output y;

- Conversely, every component of y may be expressed as a function of the system variables and of a finite number of their time derivatives.

It is quite common to include a positional and velocity term in the output vector of a system. In a differentially flat system however, the components must not be differentially related and therefore only velocity or position may be used in the output vector.

Within Martin *et al.* (1994) the author demonstrates differential flatness for a planar VTOL aircraft. Differential flatness has also been determined for a helicopter to achieve reference trajectory tracking within Koo and Sastry (1999). Driessen and Robin (2004) have developed a differentially flat model of the quadrotor albeit different to the one presented within this work, this paper also utilises the differentially flat property to track a reference trajectory. Within Defoort *et al.* (2007) the differential flatness property is used within the trajectory optimization to formulate the optimization problem within the output space which is parameterized using B-Splines.

1.5.2 Real time trajectory generation

Real time trajectory generation is required to produce a reference trajectory or to determine the control input dependent on the control scheme. With recent computational advances real time optimization is a feasible target and there has been numerous approaches to solving the problem. In Ross and Fahroo (2006), a review of several techniques is given including differential inclusions, control parameterization, flatness parameterization and high order inclusions. This highlights some key factors to be taken into account, when considering real-time trajectory generation such as cost function convexity, sparsity and matrix vector products.

Computation time

It is stated that it is desired that the trajectory optimization is run in real time. What is meant by real time, in this case it is the ability to redetermine the reference trajectory online when required so that the vehicle can have a smooth transition from one trajectory to the second trajectory, without having to revert to hover to wait for the new trajectory. There are many computational techniques and technologies which can reduce computation time, for example a Field Programmable Gate Array produced results 330 times faster than a 1GHz process desktop PC (Yamaguchi *et al.* 2001). In the case where the trajectory optimization is implemented within a MBPC scheme, the trajectory optimization is run at the frequency of the control system and therefore in this case real time refers to the ability to redetermine the trajectory at every control step.

1.5.3 MBPC

Model Based Predictive Control (MBPC) is an advanced control technique which is essentially a process of repeated constrained optimizations at each time step. This technique has been used extensively in the process industry but has yet to be widely applied to other sectors due to the computational demands of the algorithm. At each time step, the optimization minimizes some cost function over a time horizon subject to a set of equality and inequality constraints. Typically the time horizon is some fixed time and, therefore, at each time step as time progresses the time horizon receeds. This is why MBPC is sometimes referred to as receeding horizon control. The cost function is an approximate quantitive measure of the optimality of the solution. In the case of path planning, for example, this could be the minimum distance traveled to reach its destination. The constraints typically contain dynamic constraints acting on the system as well as user required constraints such as time constraints, safety constraints and possibly environmental constraints.

Now, again referring to the example of trajectory generation, assume the vehicle is required to reach a destination at a set time and by the shortest route possible. It would be possible to constrain the time and final destination and to have some measure of the shortest distance as the objective function (cost function.) This however applies two hard constraints to the optimization and one objective function, if the optimization routine does not find a feasible path to the destination then it will not return a solution. If, however, the time is entered as a second cost function, albeit with a high weighting, then a solution may be returned with a time slightly longer or shorter than the initial constrained time. This technique is often referred to as constraint softening where constraints are referred to as hard constraints and the objective function is referred to as a soft constraint.

Having given an overview of MBPC it is probably worth exploring the benefits and the reasons for using this computationally demanding technique. As stated by Mayne *et al.* (2000) the “raison d’etre” of MBPC is its constraint handling, whereby a feasible solution guarantees constraint satisfaction. This is obviously of great benefit for a range of tightly constrained systems where controllers may often reach saturation, or in the case where there is an environmental change. In these cases the initial feasible solution may not in fact remain feasible but at the next time step these new conditions will be considered and a new feasible solution will be found.

1.5.4 Nonlinear MBPC

If a system possess severe nonlinearities then the usefulness of linear predictive control is limited (Maciejowski 2002). The major disadvantage with nonlinear MPC is the loss of convexity of the problem, this is not necessarily an issue in terms of finding the global optimum, as this is not always necessary and in any case, a linear model only gives the illusion of finding this optimum. However, it is a major disadvantage in terms of computation time, as there is no guaranteed time in which a optimal solution will be found, if one exists at all. This leaves a problem in terms of how to proceed if the routine does

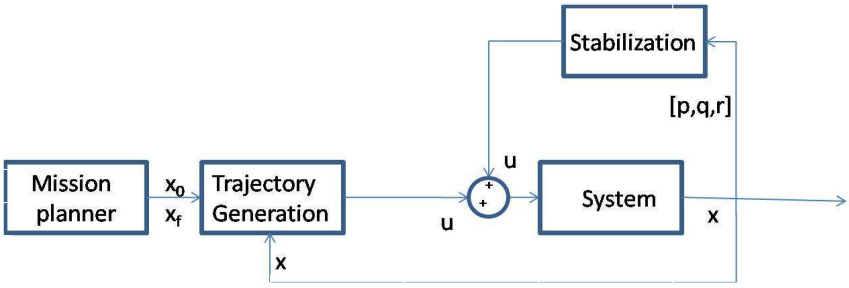


Figure 1.6: MBPC controller

not find a solution or takes too long to find it. One approach to solving this problem is considered in Scokaert *et al.* (1999) where it is shown that a feasible solution existing is sufficient for stability and in fact an optimal solution is not required.

1.5.5 MBPC for trajectory generation

Model based predictive control can be applied for trajectory generation and trajectory following (Richards and How 2004). By repeatedly solving the open loop optimal trajectory problem the next control action can be determined, effectively closing the loop as shown in Figure 1.6. This allows constraint satisfaction over a given time horizon such as obstacle avoidance, as well as initial and terminal constraints. This approach is advantageous over single trajectory optimization because at each time step the constraints are satisfied assuming a feasible solution is found, regardless of state or environmental changes since the previous time step. Within Chen and Huzmezan (2003*a*) MBPC is combined with LQ control to control a quadrotor, the MBPC controls the lateral position of the vehicle whereas the LQ control stabilizes the vehicle and controls attitude. The combining of MBPC with a linear control technique in this way in many ways is a logical step as MBPC is computationally demanding and therefore solving the full problem in real time at every time step requires large processing capabilities.

1.5.6 Parameterization techniques

Trajectory optimization typically requires a suitable control space parameterization technique. There are many potential parameterization techniques such as the simple monomial (elementary polynomial) (Yakimenko 2000, Eikenberry *et al.* 2006, Yakimenko 2006, Nieuwstadt and Murray 1995) and orthogonal polynomials such as Chebyshev polynomials (Fahroo and Ross 2000, Vlassenbroeck and Van Dooren 1988) and Laguerre polynomials (Huzmezan *et al.* 2001). Chebyshev polynomials are cylindrical in nature and therefore initial and terminal conditions can be determined analytically which make them an appealing technique especially within fluid dynamics (Canuto *et al.* 1988, McKernan *et al.* 2006). Laguerre polynomials which are the solution to Laguerre differential equa-

tion essentially provide a weighting for each term of a polynomial to reduce sensitivity and improve conditioning as the power of the basis function increases.

The choice of parameterization technique is to some degree application dependent, for example, the necessity to meet boundary conditions, although convergence properties may vary from technique to technique as discussed within Section 1.5 where non-smooth solutions prevent convergence with equally spaced nodes (Ross and Fahroo 2002). The importance of selecting the best parameterization technique varies depending on the problem and technique under consideration. In the case when the entire trajectory is approximated by a number of piecewise polynomials with the coefficients for each piece being used as a varied (free) parameter to be optimized (Fahroo and Ross 2000, Betts 2001), it makes a world of a difference because of the numerical robustness of the optimization algorithm. For example, for 10 different-length pieces with a 3rd order parameterization, for each of them, the search has potentially up to 130 free variables. The better accuracy requires more pieces (nodes). For example, SOCS (Boeing 2008), a specialised optimization package calls for as many as 807 grid points and 5,795 varied parameters to optimize a couple of minutes of a glide weapon deployment maneuver of a high-performance aircraft. No wonder that in such a case the computation time is strongly dependent upon convexity of the problem, capability to use matrix rather than scalar products, and sparsity of matrices. Therefore, the approximations that have some special properties (e.g., orthogonality) might be preferable to others.

When considering an output space optimization such as Taranenko's Direct Method (Yakimenko 2000), four polynomials are required to approximate each Cartesian coordinate as well as the speed profile along the trajectory. In this scheme the majority of polynomial coefficients are calculated analytically to satisfy the boundary conditions, which means the problem is simplified in comparison to other schemes which may require dozens or even hundreds of varied coefficients. In this algorithm, the choice of parameterization function will therefore be dependent on specific application and the vehicles dynamic properties without affecting the robustness of the algorithms. Furthermore by reducing the number of free variables the computation time is reduced, however, this does come at a cost and this is that the computed output may be sub-optimal.

When considering onboard trajectory generation it is important to run in real time, obviously the optimal trajectory is desired but if this is not computable in real time then the sub-optimal trajectory, or even any feasible trajectory, is preferable to no reference trajectory.

1.5.7 Taranenko's direct method

As discussed, it is desirable to solve non-linear optimization problems in real time but this is not often achievable using classical methods such as Bellmans equations (Yakimenko 2006). This problem, therefore, requires a different approach and several have been proposed such as discussed such as collocation method (Hargraves and Paris 1987) and method of differential inclusions (Seywald 1994). Another method which has not been discussed yet is the direct method proposed by Taranenko and Momdzhi (1968) dur-

ing the early 1960's in Russia. The majority of subsequent work on this method has been published in Russian and, therefore, is lesser known than some other techniques. Recently however, this work has been developed at the Naval Postgraduate School in California by Yakimenko (2006), where its capability for real-time on-board trajectory optimization and has been demonstrated (Dobrokhodov and Yakimenko 1999, Kaminer *et al.* 2006).

As before, direct trajectory optimization involves the discretization of the optimal control problem before solving the resulting non-linear problem (Fahroo and Ross 2000). Taranenko's scheme consists of parameterization of the output space as some function of a virtual argument allowing the separation of the position and speed profile. This, coupled with an appropriate cost function, allows easy computation of the minimum fuel or minimum time problem. The polynomial coefficients are determined analytically as functions of the initial and terminal conditions allowing for a reduction in the number of free variables, leaving only the final virtual argument as well as any unconstrained boundary conditions. As the boundary conditions are met analytically the initial guess is typically close to the optimal solution and often feasible. This improves convergence, reduces the problems caused by the non-convexity and removes 'wild' trajectories from the search space. As well as reducing the computation time, this big reduction in the number of free variables minimises the effect of other optimization considerations such as convexity and matrix sparsity which are major influences on the convergence properties of other techniques (Fahroo and Ross 2000, Betts 2001). The downside with such a scheme is the sub-optimality of the resulting trajectory, however for trajectory optimization it is assumed real time capability is of greater importance.

In fact there is a school of thought that claims, that in fact we do not want to find the optimal solution at all (Yakimenko 2006). One reason for this is due to the fragility of the optimal solution. The optimal trajectory to reach a point is generally found at the edge of the feasible search space, in other words the optimal solution results in at least one state or control input, equaling its constrained maximum. This system can be described as fragile, as the system can not increase this state. For example in the minimum time problem, the maximum speed is demanded for the entire flight, however when traveling at the maximum speed the vehicle can obviously no longer accelerate. In conventional aircraft, the edge of the flight envelope is very rarely explored except for in extreme cases such as combat situations.

1.6 Trajectory following

Once a reference trajectory has been determined, an inner loop is typically required to follow this trajectory, (Martin *et al.* 1994) although there are exceptions, such as the MBPC formulation discussed within Section 1.5.5. This can be implemented with position feedback to track this position as seen in Figure 1.5. Alternatively a multivariable controller can be implemented for stabilization as well as trajectory tracking, hence combining the two inner loops. If MBPC is possible in real time, theoretically, all three loops could be combined to achieve trajectory generation, trajectory following and stabilization but realistically the computational demand necessitates at least one inner loop. Trajectory

following or tracking is a widely publicised topic and the majority of quadrotor research to date addresses this problem as opposed to the trajectory generation problem, as discussed in Section 1.2.

1.7 Multiple Vehicles

In many cases it may be desirable to use multiple vehicles as opposed to a solitary vehicle. This may be to increase surveillance area or for improved probability of mission success as there is less dependency on a single vehicle. It is, therefore, probable at some point that the vehicles would need to rendezvous for refuelling, data transfer, close inspection or to arrange themselves into a defensive formation. The problem considered within this work is one first considered within Ando *et al.* (1999) and uses a technique called the circumcentre law. The problem considers multiple vehicles with no communication link in two dimensions, there is no predetermined rendezvous point and the only global knowledge is that of the common control scheme. Furthermore, the visibility is limited, each vehicle is assumed to have a omni-directional camera with a limited range to detect other vehicles. This problem is further considered within Lin *et al.* (2006) where graph theory is used to prove rendezvous of multiple vehicles although again only the two dimensional problem is considered.

1.8 Automatic Differentiation

When considering trajectory optimization and the desire is for real-time optimization then computation time is a major issue. Non-linear trajectory optimization is computationally demanding and this has been the prohibiting factor for techniques such as MBPC and the reason why implementation within industry has been limited. Within MATLAB (http://www.mathworks.co.uk/ n.d.), a non-linear optimization problem can be solved using a function such as *fminsearch* which uses a simplex search method. For constrained optimization, however, the problem can be easily programmed using the *fmincon* function, this uses the sequential quadratic programming method (SQP). SQP solves a quadratic programming subproblem at each iteration and is based on a quadratic approximation of the Lagrangian function. However, this SQP algorithm is gradient based and, therefore, at each time step, the sensitivity of the constraints and the cost function to a change in the free variable must be calculated. By supplying these gradients to the algorithm a significant saving in computation time can be achieved, as for a typical calculation 70% of the computation time could be used evaluating these gradients (Cao 2005). For a simple function with a few constraints, the analytical derivatives can be calculated, for a problem where there is a large number of constraints, the analytical solution is much harder to find. Calculating these derivatives using a software package such as MATLAB's Symbolic Toolbox typically leads to significant equation growth. Techniques such as finite differencing are limited as the accuracy is restricted by truncation and cancellation errors.

Automatic differentiation is a computational technique which is essentially a sequential

application of the chain rule to determine the gradient of a function. By splitting a complex function into a sequential list of arbitrary computational processes with well known derivatives, the chain rule can be used to calculate the derivatives of the complex function. Automatic differentiation is, therefore, error free unlike finite differencing, it also avoids the equation growth of symbolic toolbox software and is much quicker to implement than the analytical solution. In Cao (2005) automatic differentiation is used to evaluate dynamic sensitivity and is compared with other techniques such as CVODES, within this scheme automatic differentiation reduces the computational time by an order of 2 magnitudes. There are many software packages which enable the implementation of automatic differentiation. Through MATLAB it can be implemented directly using Matlab Automatic Differentiation (MAD) (Forth 2006), alternatively a package such as ADOL-C written in C/C++ (Griewank *et al.* 1996) can be interfaced using a MEX wrap as in Cao (2005).

1.9 Instrumentation and sensors

Two key areas of autonomy have been discussed to formulate a potential control scheme for the quadrotor. So far, however, the issue of sensors has been neglected. The recent developments in microelectromechanical systems (MEMs) sensors have provided a cost effective, lightweight solution for state measurement. A inertial measurement unit (IMU) such as the one used within Carnduff *et al.* (2007) is widely used for UAV attitude determination although these can be bulky and typically cost around U.S. $ 2000. Alternatively, the solid state gyroscopes fitted to the quadrotor can be combined with MEM's accelerometers to form an alternative and cheaper solution.

Potentially the biggest challenge for indoor flight of the quadrotor is the determination of the position. The quadrotor is likely to be used for internal flight, therefore, GPS is not a viable option. One option for obtaining state information is a positioning system similar to the one used within Valenti *et al.* (2006) which consists of six ground cameras using reflective balls placed on the vehicles to determine attitude and position. This option provides accurate information for multiple vehicles but is not easily transferable from location to location, nor is it by any means a low cost solution with system installation costing approximately $100 000. In Tournier *et al.* (2006) an alternative solution is presented, in this case the cameras are mounted on the vehicles as opposed to the ground and ground markers displaying Moire patterns. For a transportable solution the instrumentation needs to be on board and able to determine the attitude and position. The control action can be determined onboard (Altug *et al.* 2002) which is preferable for industrial application. For research purposes however, a ground station is preferable for ease of access to the control algorithm (Castillo *et al.* 2004). It is therefore, necessary to establish a communication link between the vehicle and the ground station. The state feedback can be measured on board and sent back to the ground station using a Bluetooth connection (De Nardi *et al.* 2006, Hoffmann *et al.* 2004) providing low latency.

1.10 Review

Potentially there are a vast number of applications for unmanned aerial vehicles. The quadrotor's main benefits include its agility, maneuverability and dynamic simplicity. Likely applications for such a vehicle include internal flight and short search and surveillance missions. The flight time for the vehicle is currently limited to around 20 minutes, which makes extensive urban surveillance mission unlikely. The vehicle is much more likely to be used for internal inspection, close proximity search and surveillance or simply peering around or over a visual obstacle such as a building. It can, therefore, be assumed that the quadrotor will be operating within a visual field, the flight time will be less than 20 minutes and that multiple obstacle avoidance will be unlikely, although obstacle avoidance capabilities will be essential. This thesis, therefore, concentrates on local trajectory planning schemes which work within a visual field. Path planning schemes such as MILP and roadmaps are not considered as the vehicle is intended for flight within a visual field and multiple obstacle avoidance is unlikely. Instead open-loop optimal control schemes are discussed which provide full state trajectory information. Open-loop trajectory optimization can be repeated continuously to form a MBPC controller which in effect combines the trajectory generation and trajectory following loops seen in Figure 1.5. Alternatively, given a reference trajectory an inner loop trajectory follower can be used to track this trajectory. As trajectory generation should be capable of running in real time this must be considered within the problem formulation.

1.11 Outline and Contributions

1.11.1 Publications resulting from this work

- Cowling, I.D, J.F. Whidborne and A.K. Cooke (2006). MBPC for Autonomous Operation of a Quadrotor Air Vehicle. In: Proc 21st International UAV Systems Conf. Bristol, UK.

- Cowling, I.D, J.F. Whidborne and A.K. Cooke (2006). Optimal trajectory planning and LQR control for a quadrotor UAV. In: Control 2006. Glasgow, U.K.

- Cooke, A.K, I.D. Cowling, S.D. Erbsloeh and J.F. Whidborne (2007). Low cost system design and development towards an autonomous rotor vehicle. In: Proc 22nd International UAV Systems Conf. Bristol, UK.

- Cowling, I.D, O.A. Yakimenko, J.F. Whidborne and A.K. Cooke (2007). A prototype of an autonomous controller for a quadrotor UAV. In: Proc. 2007 Euro. Contr. Conf. Kos, Greece.

- Cowling, I.D. and J.F.Whidborne (2008). Multiple vehicle rendezvous using the circumcenter law. In: Proc. 23rd International UAV Systems Conf. Bristol, UK.

- Cowling, I.D., O.A. Yakimenko and J.F.Whidborne (2008). A Direct Method for UAV Guidance and Control. In: Proc. 23rd International UAV Systems Conf. Bristol, UK.

- Cowling, I.D., O.A. Yakimenko, J.F.Whidborne and A.K. Cooke (2008). Prototype control system for autonomous quadrotor UAV. In preparation for submission to an international journal.

- Cowling, I.D. and J.F.Whidborne. Rendezvous of multiple quadrotors using the circumcenter law. In preparation for submission to an international journal.

1.11.2 Chapter 2

Chapter 2 derives the nonlinear dynamic equations of motion for the quadrotor. These are simplified using a non-conventional rotational matrix. A linear model is also presented with some linear analysis of the vehicle. Finally, a full dynamic model is described which has been developed at Cranfield University (Shaw 2005) and is used for simulation of the control algorithms.

1.11.3 Chapter 3

Chapter 3 defines three missions for testing the control algorithm. The differentially flat equations of motion are derived for the quadrotor which enables output space optimization. A suitable cost function is presented and the convex constraints are defined. Different basis functions are considered for output space parameterization and Laguerre polynomials are shown to reduce computation time for the trajectory optimization. Finally, some trajectories are optimized to demonstrate the Laguerre polynomial based algorithm.

1.11.4 Chapter 4

Chapter 4 considers the problem of closing the loop. A Model Based Predictive Control algorithm is presented which repeatedly solves the trajectory optimization presented in Chapter 3. An alternative scheme which consists of a single trajectory optimization and a linear tracking controller (LQR). Results for both of these schemes are compared and a combined controller is proposed. This combined controller consists of an outer loop trajectory generator and an inner loop trajectory follower, the loops are switched on and off by an update switch which measures the reference trajectory feasibility and the vehicles drift from this trajectory. The combined controller is then demonstrated for the 3 missions defined previously.

1.11.5 Chapter 5

Chapter 5 considers an alternative trajectory optimization scheme. Taranenko's direct method was developed in the 1960's in Russia but with the majority of papers being published in Russian it is generally overlooked with a few notable exceptions (Kaminer *et al.* 2006, Yakimenko 2000). This chapter looks at the major advantages of such a scheme and incorporates this scheme into the control architecture presented previously to simulate the same 3 missions.

1.11.6 Chapter 6

Chapter 6 considers a decentralized rendezvous algorithm. The circumcenter law as presented in Ando *et al.* (1999) is extended to 3 dimensions. The combined control architecture and Taranenko's direct method are applied to simulate the rendezvous of multiple quadrotors. This is also extended to consider the conflicting problems of rendezvous and collision avoidance and a new measurement of direct consequence is presented.

1.11.7 Chapter 7

Obviously to achieve autonomous flight it is necessary to consider issues other than trajectory generation and following. Chapter 7 considers the issues of feedback on the quadrotor with limited payload. The experimental set up at Cranfield University is described and some open loop test results are shown. Finally there is a discussion on the challenges that remain, in order to achieve practical autonomy of a small UAV such as the quadrotor.

In Chapter 8 some final conclusions are made and suggestions for future work within the area. This thesis concludes with details of the dynamic model used for simulation of the control schemes in Appendix A and Appendix B.

Chapter 2

Dynamic Modelling

This chapter describes the dynamics of the quadrotor, the subsequent equations of motion and dynamic modelling of the quadrotor. This chapter starts by stating the approximations and derives the non-linear equations of motion which are used for trajectory optimization. This model is then linearized so that a linear multi-variable controller can be designed for the purposes of path following. Section 2.4 states the equations within the full dynamic model of the quadrotor which have been developed for simulation and validation purposes. Finally this chapter concludes with a discussion of the constraints acting on the quadrotor and quantifies the necessary control constraints which are required for control design.

2.1 Dynamic model for control system design

2.1.1 Assumptions when modelling

There are certain approximations used when modelling the quadrotor when deriving the equations for control system design.

- The arms and body are rigid
- The body axes system coincides with the principal moment of inertia axes, therefore $[I_{xy}, I_{yz}, I_{zx}] \approx 0$.
- The body rate around the z axis will be kept small under closed loop control $r \approx 0$
- 90° rotational symmetry about the z axis, therefore $I_x = I_y$
- Gyroscopic rotor forces are considered to be negligible
- Motor inertia is small and therefore lag within the motors is negligible

2.1.2 Actuator outputs and control inputs

The quadrotor is controlled only by independently varying the speed of the four rotors. A pitch moment is achieved by varying the ratio of the front and back rotor speeds, a roll by varying the left and right rotor speeds (see Figure 2.1). A yaw moment is obtained from the torque resulting from the ratio of the clockwise (left and right) and the anti-clockwise (front and back) speeds.

These inputs are clearly working in the body axis, however, when following a ground track the positional terms in the state space matrix are in the earth axis. It is possible to relate these earth positional terms to the actuator inputs for the benefit of this state space derivation, however, doing so involves the introduction of a rotational transform adding to the complexity. Therefore, this work uses a state space matrix entirely based in the earth axis ($[x, y, z, \phi, \theta, \psi]$). The control inputs are therefore also defined in this axis ($[\ddot{\phi}, \ddot{\theta}, \ddot{\psi}]$). The demands going to the motors however are obviously in the body axis, these are called the actuator outputs which are functions of the body accelerations $(\dot{p}, \dot{q}, \dot{r})$

The first actuator output ($\tilde{u}_1$) is a combination of the total thrust from the four rotors (τ_i), however, to simplify the state space equations of motion this is divided by the vehicle mass (m). The first actuator output is therefore a body acceleration defined by:

$$\tilde{u}_1 = \frac{\tau_1 + \tau_2 + \tau_3 + \tau_4}{m} \quad (m/s^2) \tag{2.1}$$

where τ_i is the thrust from the ith rotor.

The pitching and roll moments are generated through the differential thrust between the front and back and left and right rotors. These pitching moments are defined:

$$\tilde{u}_2 = l(\tau_4 - \tau_3) \quad (Nm) \tag{2.2}$$
$$\tilde{u}_3 = l(\tau_1 - \tau_2) \quad (Nm) \tag{2.3}$$
$$\tag{2.4}$$

where l is the distance from the centre of mass.

The final actuator output is a combination of the individual torques (υ_i) generated by each of the four rotors.

$$\tilde{u}_4 = \upsilon_3 + \upsilon_4 - \upsilon_1 - \upsilon_2 \quad (Nm) \tag{2.5}$$

where υ_i is the torque from the ith rotor.

Actuator outputs $(\tilde{u}_2 - \tilde{u}_4)$ can be related to the body rates

$$\tilde{u}_2 = I_x \dot{p} + qr(I_{zz} - I_{yy}) \tag{2.6}$$
$$\tilde{u}_3 = I_y \dot{q} - pr(I_{xx} - I_{zz}) \tag{2.7}$$
$$\tilde{u}_4 = I_z \dot{r} - pq(I_{xx} - I_{yy}) \tag{2.8}$$

Assuming symmetry about the z axis resulting in $I_{xx} = I_{yy}$ and the body angular rate about the z axis (r) is small under closed loop control the equations can be simplified resulting

in:

$$\tilde{u}_2 = I_x \dot{p} \tag{2.9}$$
$$\tilde{u}_3 = I_y \dot{q} \tag{2.10}$$
$$\tilde{u}_4 = I_z \dot{r} \tag{2.11}$$

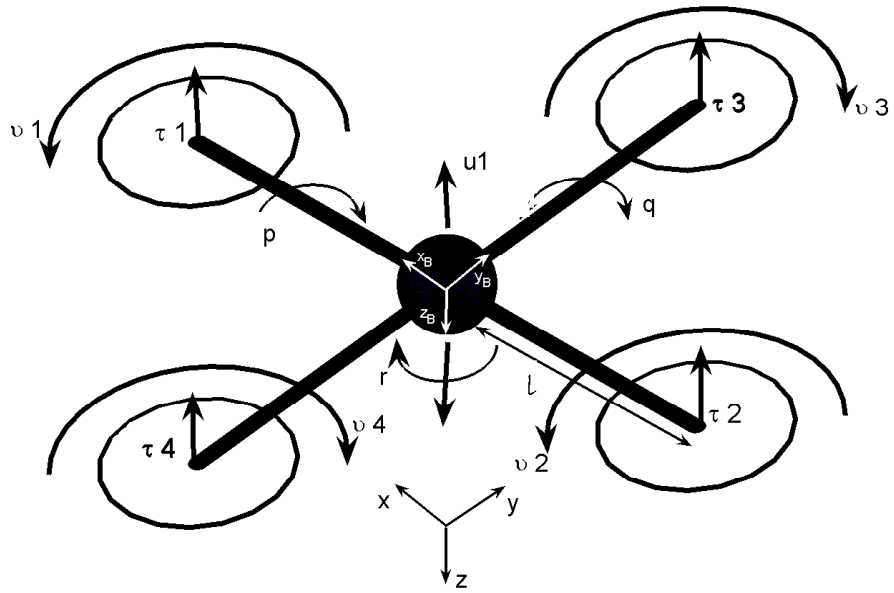


Figure 2.1: Quadrotor schematic

2.1.3 State Variables

The state variable vector, **x**, is defined as

$$\mathbf{x}^T = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix} \tag{2.12}$$

where x, y and z are the translational positions (see Figure 2.1) and ϕ, θ, ψ are the roll, pitch and yaw respectively. For surveillance operations the camera position and pointing direction are important, hence the desired outputs for the vehicle are the translational positions (x,y,x) and the yaw angle (ψ). The output vector, **y**, is hence defined as

$$\mathbf{y}^T = \begin{bmatrix} x & y & z & \psi \end{bmatrix}. \tag{2.13}$$

2.1.4 Rotation Matrix

Aerospace engineers traditionally use the rotation matrix R_{xyz} (Cook 1997) to relate the body axis to the earth axis. This rotational matrix or Directional Cosine Matrix (DCM) is derived by successively rotating about the x axis, y axis and finally the z axis as shown in Figure 2.2.

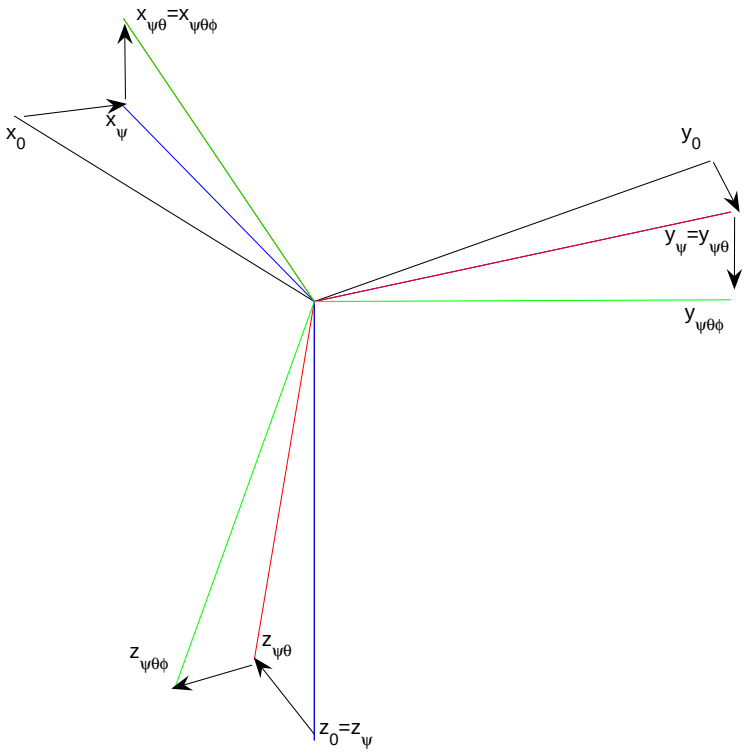


Figure 2.2: Rotations R_{xyz}

$$R_{xyz} = R_x R_y R_z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_{\hat{\phi}} & s_{\hat{\phi}} \\ 0 & -s_{\hat{\phi}} & c_{\hat{\phi}} \end{bmatrix} \begin{bmatrix} c_{\hat{\theta}} & 0 & -s_{\hat{\theta}} \\ 0 & 1 & 0 \\ s_{\hat{\theta}} & 0 & c_{\hat{\theta}} \end{bmatrix} \begin{bmatrix} c_{\hat{\psi}} & s_{\hat{\psi}} & 0 \\ -s_{\hat{\psi}} & c_{\hat{\psi}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

$$R_{xyz} = \begin{bmatrix} c_{\hat{\theta}} c_{\hat{\psi}} & c_{\hat{\theta}} s_{\hat{\psi}} & -s_{\hat{\theta}} \\ s_{\hat{\phi}} s_{\hat{\theta}} c_{\hat{\psi}} - s_{\hat{\psi}} c_{\hat{\phi}} & s_{\hat{\phi}} s_{\hat{\theta}} s_{\hat{\psi}} + c_{\hat{\phi}} c_{\hat{\psi}} & s_{\hat{\phi}} c_{\hat{\theta}} \\ c_{\hat{\phi}} s_{\hat{\theta}} c_{\hat{\psi}} + s_{\hat{\phi}} s_{\hat{\psi}} & c_{\hat{\phi}} s_{\hat{\theta}} s_{\hat{\psi}} - c_{\hat{\psi}} s_{\hat{\phi}} & c_{\hat{\theta}} c_{\hat{\phi}} \end{bmatrix} \tag{2.15}$$

where $c_{\hat{\theta}} = \cos\hat{\theta}$ and $s_{\hat{\psi}} = \sin\hat{\psi}$ etc.

This directional cosine matrix can translate body forces or accelerations from the earth axis (P_{EA}) to the body axis P_{BA}:

$$P_{BA} = DCM \times P_{EA} \tag{2.16}$$

For the state space equations of motion the thrust (u_1), is acting in the body axis. Relating this body axis acceleration to accelerations in the earth axis can be done through the transpose of the DCM matrix and recalling that the upward thrust is providing a negative acceleration in the downward pointing z axis:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = R_{xyz}^{T} \begin{bmatrix} 0 \\ 0 \\ -u_1 \end{bmatrix} \tag{2.17}$$

The translational equations of motion can therefore be derived recalling the control inputs 2.26

$$\ddot{x} = \tilde{u}_1(c_{\hat{\psi}} s_{\hat{\theta}} c_{\hat{\phi}} + s_{\hat{\psi}} s_{\hat{\phi}}) \tag{2.18}$$

$$\ddot{y} = \tilde{u}_1(s_{\hat{\psi}} s_{\hat{\theta}} c_{\hat{\phi}} - c_{\hat{\psi}} s_{\hat{\phi}}) \tag{2.19}$$

$$\ddot{z} = g - \tilde{u}_1 c_{\hat{\theta}} c_{\hat{\phi}} \tag{2.20}$$

When considering trajectory generation, in order to reduce computation time, it is desired to have the translational equations in the simplest form. When using the rotation matrix R_{xyz} for a fixed wing vehicle, the first rotation is about the axis of thrust. This simplifies the equations of motion. This scheme is adopted throughout the industry even for other vehicles in which the thrust acts along a different axis such as a helicopter. If for the quadrotor a rotational matrix is chosen with the first rotation about the axis of thrust i.e the z axis, then the equation can be simplified. In Figure 2.3 the rotations occur around the z axis the x axis and finally the y axis, giving

$$R_{zxy} = R_z R_x R_y = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \begin{bmatrix} c_\theta & 0 & -s_\theta \\ 0 & 1 & 0 \\ s_\theta & 0 & c_\theta \end{bmatrix} \tag{2.21}$$

with the resulting rotational matrix

$$R_{zxy} = \begin{bmatrix} s_\theta s_\phi s_\psi + c_\psi c_\theta & c_\phi s_\psi & c_\theta s_\psi s_\phi - s_\theta c_\psi \\ s_\phi s_\theta c_\psi - c_\theta s_\psi & c_\theta c_\phi & c_\theta s_\phi c_\psi + s_\theta s_\psi \\ s_\theta c_\phi & -s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.22}$$

This results in the following translational equations of motion for the quadrotor which are a significant reduction in complexity compared with (2.18-2.20).

$$\ddot{x} = -\tilde{u}_1 s_\theta c_\phi \tag{2.23}$$

$$\ddot{y} = \tilde{u}_1 s_\phi \tag{2.24}$$

$$\ddot{z} = g - \tilde{u}_1 c_\theta c_\phi \tag{2.25}$$

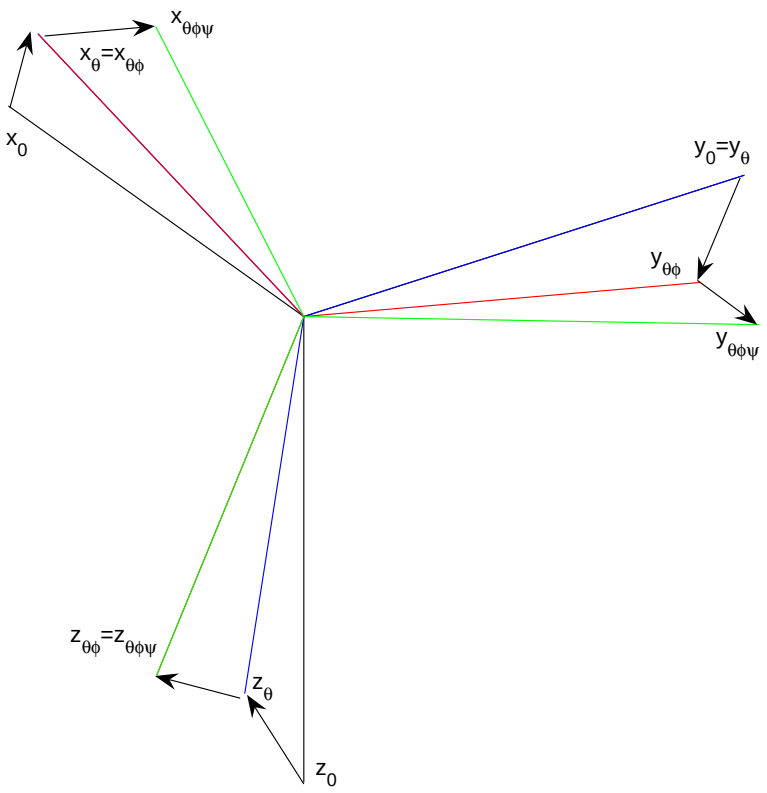


Figure 2.3: Rotations R_{zxy}

2.1.5 Input Transforms

As will be seen later the control inputs are not chosen to equal the actuator outputs. Instead, the control vector **u**, is expressed as a function of the Euler angles and defined as

$$\mathbf{u}^T = [u_1,\ u_2,\ u_3,\ u_4]\,. \tag{2.26}$$

where $u_1 = \tilde{u}_1$, $u_2 = \ddot{\phi}$, $u_3 = \ddot{\theta}$ and $u_4 = \ddot{\psi}$. The actuator outputs are therefore acting in the body axis unlike the control inputs which are functions of Euler angles. The Euler angles can be expressed as a function of the body rates using

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = L \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.27}$$

where

$$L = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & s_\phi \\ 0 & -s_\phi & c_\phi \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \tag{2.28}$$

$$= \begin{bmatrix} \cos\psi & \sin\psi\cos\phi & 0 \\ -\sin\psi & \cos\psi\cos\phi & 0 \\ 0 & -\sin\phi & 1 \end{bmatrix} \tag{2.29}$$

The actuator outputs $(\tilde{u}_2)$,$(\tilde{u}_3)$ and $(\tilde{u}_4)$ are the turning moments acting in the body axis and resulting in a rotational acceleration $(\dot{p},\dot{q},\dot{r})$. These can therefore be expressed as a transform of the Euler angles. Recalling (2.26) these actuator outputs are now expressed as a function of the control inputs $(\ddot{\phi},\ddot{\theta},\ddot{\psi})$ and its integrals $(\dot{\phi},\dot{\theta},\dot{\psi})$.

$$\begin{bmatrix} I_x\tilde{u}_2 \\ I_y\tilde{u}_3 \\ I_z\tilde{u}_4 \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi\cos\phi & 0 \\ -\sin\psi & \cos\psi\cos\phi & 0 \\ 0 & -\sin\phi & 1 \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} -\sin\psi\dot{\psi} & \cos\phi\cos\psi\dot{\psi} - \sin\psi\sin\phi\dot{\phi} & 0 \\ -\cos\psi\dot{\psi} & -\cos\psi\sin\phi\dot{\phi} - \cos\phi\sin\psi\dot{\psi} & 0 \\ 0 & -\cos\phi\dot{\phi} & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.30}$$

Taking the state equation (2.12), the control inputs (2.26) and the equations of motion (2.23,2.24,2.25), a state space model can be defined as

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ -u_1 s_\theta c_\phi \\ u_1 s_\phi \\ g - u_1 c_\theta c_\phi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{2.31}$$

2.1.6 Linear approximations of the rotor dynamics

In order to control the actual quadrotor the rotor voltage needs to be determined for each rotor. To determine the individual thrust for each motor from the input it is possible to manipulate the equations (2.1-2.5).

$$\tau_1 = \frac{\tilde{u}_1 + 2\tilde{u}_3/l - \tilde{u}_4/C_T}{4} \tag{2.32}$$

$$\tau_2 = \frac{\tilde{u}_1 - 2\tilde{u}_3/l - \tilde{u}_4/C_T}{4} \tag{2.33}$$

$$\tau_3 = \frac{\tilde{u}_1 + 2\tilde{u}_2/l + \tilde{u}_4/C_T}{4} \tag{2.34}$$

$$\tau_4 = \frac{\tilde{u}_1 + 6\tilde{u}_2/l + \tilde{u}_4/C_T}{4} \tag{2.35}$$

where C_T is an approximate thrust to torque conversion factor. The following voltage v_i to thrust τ_i relationship can then be determined from experimental data see Figure(A.1).

$$v_i = \sqrt{37\tau_i} \tag{2.36}$$

2.2 Linear Control Model

In order to apply a suitable linear controller for trajectory following, it is necessary to have a linear model of the quadrotor. A linear model is derived by taking partial derivatives of each state and control variable with respect to the state matrix.

$$\dot{x} = Ax + Bu \tag{2.37}$$

where

$$A = \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & u_1\sin\theta\sin\phi & -u_1\cos\theta\cos\phi & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & u_1\cos\phi & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & u_1\sin\theta\cos\phi & u_1\cos\theta\sin\phi & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{2.38}$$

$$B = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
-\sin\theta\cos\phi & 0 & 0 & 0 \\
\sin\phi & 0 & 0 & 0 \\
-\cos\theta\cos\phi & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{2.39}$$

$$y = Cx \tag{2.40}$$

where

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.41}$$

This model is linearized about the hover conditions, in this case the body is level in the earth axis ($\psi = 0, \theta = 0, \phi = 0$) and the total thrust is equal to the gravitatinal force ($u_1 = mg$) leaving:

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & -u_1 & 0 & 0_{3\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & u_1 & 0 & 0_{3\times3} \\ 0_{1\times3} & 0_{1\times3} & 0 & 0 & 0 & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times3} \end{bmatrix} \tag{2.42}$$

$$B = \begin{bmatrix} 0_{3\times1} & 0_{3\times3} \\ 0 & 0_{1\times3} \\ 0 & 0_{1\times3} \\ -1 & 0_{1\times3} \\ 0_{3\times1} & 0_{3\times3} \\ 0_{3\times1} & I_{3\times3} \end{bmatrix} \tag{2.43}$$

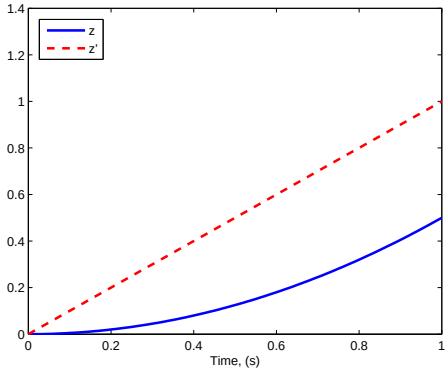
2.3 Linear Analysis

2.3.1 Controllability

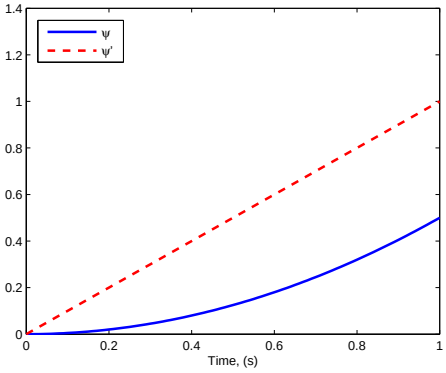
In order for the system to be controllable the controllability matrix must have full rank. Taking the linearized state space equations (2.42, 2.43), where A is an $n \times n$ matrix and B is an $n \times m$ matrix, the controllability matrix $\mathcal{C}$ is defined:

$$\mathcal{C} = [B\, AB\, A^2B\, ...A^{n-1}B] \tag{2.44}$$

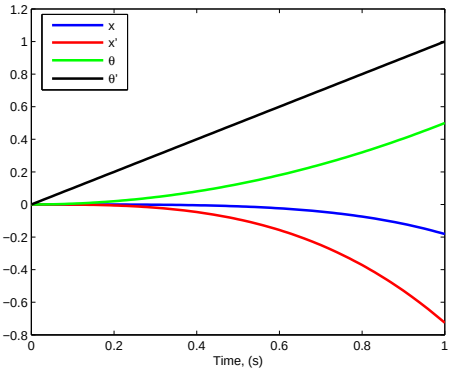
The controllability matrix has full rank and is therefore fully controllable for any linearized model which is linearized with a control input $u_1 > 0$. Therefore as long as the linear model is about a point with a positive thrust and not in free fall the vehicle is controllable. This is an expected result as if there is zero thrust then each motors individual and relative thrust is zero.



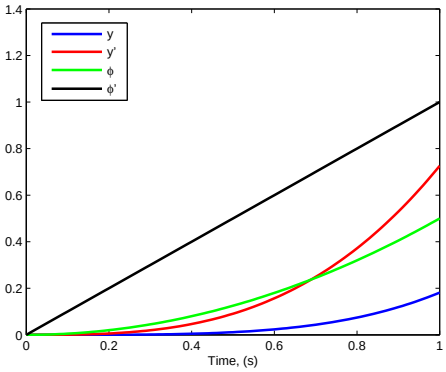
(a) System time response to a small peturbation in u_1



(b) System time response to a small peturbation in u_2



(c) System time response to a small peturbation in u_3



(d) System time response to a small peturbation in u_4

Figure 2.4: System time response

2.3.2 Stability

The stability of the vehicle can be analysed by finding the eigenvalues of the linearized A matrix. This produces 12 poles all of which are found at the origin showing that the system is marginally stable. The time response of the linear system to a step change is shown in Figure 2.4.Therefore closed loop control is required in order to stabilise the system.

2.4 Full Dynamic Model

In order to test the control algorithms it is necessary to have a full dynamic model of the quadrotor for simulation purposes. A model of the small quadrotor (Figure 1.2) has been developed at Cranfield University (Shaw 2005) and contains experimental data details of which can be found within Appendix A. This model is used for the majority of simulations within this work although this model has also been adapted to model the Draganflyer X-Pro (Figure 1.1), details of this can be found within Appendix B.

2.4.1 Assumptions when modelling

A revised list of assumptions is required for assumptions for the full dynamic model as opposed to the equations derived for control system design. These assumptions apply to the models described in Appendix A and Appendix B.

- The arms and body are rigid
- Gyroscopic forces are considered to be negligible
- Motor inertia is small and therefore lag within the motors is negligible

2.4.2 Powerplant

The powerplant converts the input voltages into the total thrust and turning moments of the quadrotor. The thrust and speed of each rotor is calculated by interpolating the experimental data and from this the torque can be calculated. Using blade element theory (Cooke 2002) and making the usual assumptions we can state that the total power required per rotor is ;

$$P_i = T_i(V_C + V_I) + 0.125\rho bcR_b^4\Omega_i^3 C_d \tag{2.45}$$

where T_i in this section is the thrust from the ith rotor, V_I is the induced velocity, V_C is the vertical speed, ρ is the air density, b is the number of blades, c is the chord, R_b is the radius, Ω_i is the speed of the ith rotor and C_d is the drag coefficient. The induced velocity for each rotor can be determined by:

$$V_I = \sqrt{\frac{T_i}{2\rho A_d}} \tag{2.46}$$

where A_d is the disk area. Taking $P_i = Q_i\Omega_i$, the torque Q_i from the ith rotor can be calculated by:

$$Q_i = \frac{T_i(V_C + V_I) + 0.125\rho bcR_b^4\Omega_i^3 C_d}{\Omega_i} \tag{2.47}$$

The propulsive forces (X_p, Y_p, Z_p) and moments (L_p, M_p, N_p) acting on the vehicle can be calculated using

$$Z_p = -T_1 - T_2 - T_3 - T_4 \tag{2.48}$$

$$L_p = (T_3 - T_4)l \tag{2.49}$$

$$M_p = (T_1 - T_2)l \tag{2.50}$$

$$N_p = (Q_1 + Q_2) - (Q_3 + Q_4) \tag{2.51}$$

2.4.3 Equations of motion

The gravitational forces acting on the body can be calculated using the directional cosine matrix as shown;

$$\begin{bmatrix} X_g \\ Y_g \\ Z_g \end{bmatrix} = \begin{bmatrix} c_{\hat{\theta}}c_{\hat{\psi}} & s_{\hat{\phi}}s_{\hat{\theta}}c_{\hat{\psi}} - s_{\hat{\psi}}c_{\hat{\phi}} & c_{\hat{\phi}}s_{\hat{\theta}}c_{\hat{\psi}} + s_{\hat{\phi}}s_{\hat{\psi}} \\ c_{\hat{\theta}}s_{\hat{\psi}} & s_{\hat{\phi}}s_{\hat{\theta}}s_{\hat{\psi}} + c_{\hat{\phi}}c_{\hat{\psi}} & c_{\hat{\phi}}s_{\hat{\theta}}s_{\hat{\psi}} - c_{\hat{\psi}}s_{\hat{\phi}} \\ -s_{\hat{\theta}} & s_{\hat{\phi}}s_{\hat{\theta}} & c_{\hat{\theta}}c_{\hat{\phi}} \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{2.52}$$

By defining the body forces (F_B) acting on the vehicle:

$$F_B = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.53}$$

The body accelerations can be derived (Stevens and Lewis 1992):

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = -\Omega \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} X_g/m \\ Y_g/m \\ Z_g/m \end{bmatrix} + \begin{bmatrix} X/m \\ Y/m \\ Z/m \end{bmatrix} \tag{2.54}$$

where Ω is the cross product matrix:

$$\Omega = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{2.55}$$

resulting in:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw - g\sin\theta + X/m \\ -ru + pw + g\sin\phi\cos\theta + Y/m \\ qu - pv + g\cos\theta\cos\phi + Z/m \end{bmatrix} \tag{2.56}$$

In a similar way the body axis rotations can be derived (Stevens and Lewis 1992):

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = -I^{-1}\Omega I \begin{bmatrix} p \\ q \\ r \end{bmatrix} + I^{-1} \begin{bmatrix} L \\ M \\ N \end{bmatrix} \tag{2.57}$$

where L, M and N are the torques acting in the body axis and I is the inertia matrix:

$$\begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{2.58}$$

The body rates are translated into the earth axis using the rotational matrix

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_{xyz} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.59}$$

where

$$R_{xyz} = \begin{bmatrix} c_{\hat{\theta}}c_{\hat{\psi}} & s_{\hat{\phi}}s_{\hat{\theta}}c_{\hat{\psi}} - s_{\hat{\psi}}c_{\hat{\phi}} & c_{\hat{\phi}}s_{\hat{\theta}}c_{\hat{\psi}} + s_{\hat{\phi}}s_{\hat{\psi}} \\ c_{\hat{\theta}}s_{\hat{\psi}} & s_{\hat{\phi}}s_{\hat{\theta}}s_{\hat{\psi}} + c_{\hat{\phi}}c_{\hat{\psi}} & c_{\hat{\phi}}s_{\hat{\theta}}s_{\hat{\psi}} - c_{\hat{\psi}}s_{\hat{\phi}} \\ -s_{\hat{\theta}} & s_{\hat{\phi}}s_{\hat{\theta}} & c_{\hat{\theta}}c_{\hat{\phi}} \end{bmatrix} \tag{2.60}$$

Finally, the rates of change of the Euler attitude angles are determined by;

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \\ q\sin\phi\sec\theta + r\cos\phi\sec\theta \end{bmatrix} \tag{2.61}$$

2.5 Constraints

The quadrotor has very few constraints on motion, there is no minimum speed for the vehicle as it is capable of hovering and it is omnidirectional in the sense that from hover it can move is any direction required. There are however constraints on the control inputs in terms of maximum voltage and therefore maximum thrust, roll moment, pitch moment and yaw moment. To avoid singularities and consider stability the attitude is constrained although this is not a physical constraint it is present within the control algorithm, details of this can be found within Section 3.4. From experimental data at the maximum voltage of $7.5v$:

$$\tau_n \leq 1.364(N) \tag{2.62}$$
$$\Omega_i \leq 216(rad/s) \tag{2.63}$$

Therefore if $u_2 = u_3 = u_4 = 0$ then the maximum acceleration is

$$u_{1MAX} = \frac{1.364 \times 4}{m} = 12.28m/s^2 \tag{2.64}$$

Obviously if the turning moments are increased from zero then the maximum acceleration will be reduced and therefore a reasonable trade off is required to determine the maximum control inputs. Assuming the body rotational accelerations $(\dot{p}, \dot{q}, \dot{r})$ are constrained to $0.5rad/s^2$, whereas, in reality the rates would be considerably smaller than this, the actuator outputs are constrained to

$$\tilde{u}_{2MAX} = 0.5I_x \tag{2.65}$$
$$\tilde{u}_{3MAX} = 0.5I_y \tag{2.66}$$
$$\tilde{u}_{4MAX} = 0.5I_z \tag{2.67}$$

therefore

$$\tilde{u}_{2MAX} = 0.0042 \quad (Nm) \tag{2.68}$$
$$\tilde{u}_{3MAX} = 0.0036 \quad (Nm) \tag{2.69}$$
$$\tilde{u}_{4MAX} = 0.0062 \quad (Nm) \tag{2.70}$$

Recalling (2.32)-(2.35), it can be seen if the maximum rolling, pitching and yawing moments are applied then the maximum acceleration is reduced

$$u_{1MAX} \approx 12.18 m/s^2 \tag{2.71}$$

In conclusion it can be seen that by appying a constraint to the rolling, pitching and yawing moments the total acceleration is not significantly reduced and therefore with a suitable approximation of the control inputs the following can be applied without a significant loss in performance.

$$u_1 \leq 12.18 m/s^2 \tag{2.72}$$

$$u_2 \leq 0.5 rad/s^2 \tag{2.73}$$

$$u_3 \leq 0.5 rad/s^2 \tag{2.74}$$

$$u_4 \leq 0.5 rad/s^2 \tag{2.75}$$

Chapter 3

Trajectory Generation

Trajectory generation is the determination of the optimal path as well as the optimal speed to travel along this path. With respect to the quadrotor, the trajectory is typically a short flight lasting maybe a few seconds. However, it is likely the quadrotor will be used for an internal flight or within an urban environment which introduces the need for obstacle avoidance. For surveillance purposes it is not necessary to arrive at the destination in the minimum time, instead, maybe the vehicle is required to arrive at a specific time or with minimum fuel cost. Indirect methods yield a problem which becomes increasingly hard to solve when constraints such as obstacles are added or the cost function is modified to accommodate different requirements.

Direct methods involve the conversion of this optimal control problem into a non-linear programming problem typically within the state or control space for which there are many techniques for solving. Differential flatness is a property of some dynamical systems which allows the expression of the state and control vectors as functions of the output space and hence the optimization problem can be re-posed within the output space. This chapter will start with defining three mission which will be used for testing the trajectory optimization scheme. The general problem will then be formulated within the control space. Differential flatness will then be considered and the differentially flat equations will be derived allowing for a reformulation of the optimization problem within the output space and hence simplifying the problem. Trajectory optimization is subject to a set of inequality and equality constraints placed on the controls and states, these will be discussed within Section 3.4. Obviously for optimization a quantative measure of optimality is required and therefore within Section 3.5 a suitable cost function is defined. To reduce the problem to a finite dimensional problem a suitable parameterization is required for which there are many options including polynomials, Laguerre polynomials and Chebyshev polynomials, a few of these are considered and compared within Section 3.6. With the resulting output space problem with a suitable parameterization the topology can be examined and the missions previously defined can be used to test the optimization algorithm.

3.1 Missions

In order to demonstrate the trajectory generation algorithm and to later demonstrate the control schemes three missions have been defined. All three missions start at hover at (0,0,0) and consist of a reasonably short flight which is less than 30 seconds. Within this section the missions are defined with predetermined flight times in this case the optimization is concerned with the optimal path to reach the destination at this time and not considering the minimum time problem.

Mission (i) The first mission involves a vertical flight of 5m in 7 seconds. This mission is obviously very simple but this enables comparison with well known analytical results for minimum time or minimum fuel.

Mission (ii) The second mission involves navigation around an obstacle to reach a destination at $(6,0,0)$ in 15 seconds. The obstacle is modelled as a sphere centered at (3,0,0) with a radius of 1m.

Mission (iii) The third mission involves a horizontal flight to the top of a mineshaft or well located at $(10,0,0)$, before descending down the mineshaft to reach the bottom at $(10,0,-5)$ in 25 seconds. The mineshaft is modelled as a cylinder with radius of 2m. If a wind is applied in this mission it is acting only above ground, there is no wind acting on the vehicle within the mineshaft.

3.1.1 Disturbances and uncertainties

Applying a controller to follow a trajectory in the absence of disturbances is obviously a trivial problem, therefore the missions are simulated in the presence of wind and gusting of varying strength. Each mission is further defined dependent on the wind and gusting strength

Mission (i)

- Mission (ia) Constant wind of 0.1m/s acting along the *x* axis.
- Mission (ib) Constant wind of 0.01m/s acting along the *x* axis.
- Mission (ic) Constant wind of 0.1m/s acting along the *x*, *y* and *z* axis.
- Mission (id) No disturbances acting on the vehicle.
- Mission (ie) Constant wind of 0.1m/s acting along the *x*, *y* and *z* axis, gusting of mean 0m/s with variance of 0.01m/s acting on *x*,*y* and *z* axis.

Mission (ii)

- Mission (iia) Constant wind of 0.25m/s acting along the *x* axis.
- Mission (iib) Constant wind of 0.01m/s acting along the *x* axis.
- Mission (iic) Constant wind of 0.1m/s acting along the *x*, *y* and *z* axis,

Mission (iii)

- Mission (iiia) Constant wind of 1m/s acting along the *x* axis.
- Mission (iiib) Constant wind of 0.05m/s acting along the *x* axis.
- Mission (iiic) Constant wind of 0.1m/s acting along the *x*, *y* and *z* axis,

The model used for the majority of simulations is not a full aerodynamic model and therefore it is not possible to accurately incorporate wind into the model. Instead it is assumed that the vehicle travels in the wind axis and offers no resistance to the acceleration due to the wind. Furthermore this wind has no effect on the dynamics of the vehicle. In a similar way the gusting acts in the translational axis only and will accelerate the vehicle in this axis. As an example if the vehicle is trying to hover and a wind is acting along the *x* axis at 1m/s then the vehicle will travel at this speed along the x axis. As it is trying to hold position in the earth axis, it will try to correct this and travel back but will need an airspeed of 1m/s to remain stationary above ground.

In this chapter it is assumed throughout that there are no model uncertainties or modelling errors. In practice there are likely to be at least slight discrepancies between the model and the vehicle, these will be considered in more detail in Section 4.2.4.

3.2 General Problem

The direct method requires a transformation of the optimal control problem into a nonlinear programming problem, this is typically within the control space. By a choice of suitable control space parameterization $(\mathbf{u} = f(t))$ and from the initial states and controls $(\mathbf{x}_0, \mathbf{u}_0)$ the Cauchy problem can be solved to determine the states $\mathbf{x} = f(\mathbf{u}, t, \mathbf{x}_0, \mathbf{u}_0)$. The control input can then be optimized to meet some degree of optimality in which constraints placed on the controls, states and outputs are met.

$$\begin{aligned} \min_{\hat{\mathbf{u}}(t)\in\mathbf{u}} \ & \Phi \text{ for } t \in [0,T] \\ & \text{s.t. } c(\mathbf{u}) \leq 0 \\ & \text{s.t. } \mathbf{x}_0 - g_1(\mathbf{u}_0) = 0 \\ & \text{s.t. } \mathbf{y}_T - g_2(\mathbf{u}_T) = 0 \\ & \text{s.t. } \mathbf{y} = g_3(\mathbf{u}, \mathbf{x}_0) \end{aligned} \tag{3.1}$$

where Φ is the cost function, $c(\mathbf{u})$ is a set of functions that express inequality constraints on the state and output, $\mathbf{x}_0$ is the initial state at $t = 0$, the state is a function g_1 of the input, $\mathbf{y}_T$ represents the terminal output at $t = T$, the output is some a function g_2 of the input and some dynamic constraints are applied so that the output $\mathbf{y}$ is a function g_3 of the input $\mathbf{u}$ and the initial state $\mathbf{x}_0$.

3.3 Differential Flatness

Differential flatness as proposed by (Fleiss *et al.* 1992) is a property of some dynamical systems which allows the inversion of the dynamics and therefore the expression of the state and control inputs in terms of the output space.This allows for the optimization to occur within the output space as opposed to the control space. The output space parameterization with a one-to-one mapping into the state and control space removes the necessity to solve the Cauchy problem and instead these are evaluated analytically. Another benefit of the differentially flat approach is within the constraint analysis and specifically when a large number of constraints are present within the output space. Through differential flatness the output space constraints can be expressed as direct functions of the parameterization whereas within the general formulation these will be functions of the outputs which are determined through the Cauchy problem. Potentially in a general control problem where the majority of constraints are acting in the control space this is of no benefit. But for trajectory optimization, many constraints such as obstacle avoidance, are within the output space, so in this case the problem in simplified. It is also possible to satisfy some constraints analytically such as initial constraints $\mathbf{x}_0 = f(t_0)$ and therefore again simplify the problem.

By manipulation of the equations of motion and recalling Equation (2.31), the state vector and input vector can be expressed as a function of the output vector.

$$\theta = \arctan\left(\frac{\ddot{x}}{g-\ddot{z}}\right) \tag{3.2}$$

$$\phi = \arcsin\left(\frac{-\ddot{y}}{u_1}\right) \tag{3.3}$$

where

$$u_1 = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (g-\ddot{z})^2}. \tag{3.4}$$

From Equation 3.4:

$$\frac{\ddot{y}}{u_1} \leq 1 \tag{3.5}$$

Therefore singularities in this model only appear when

$$g - \ddot{z} = 0 \tag{3.6}$$

in other words when the vehicle is in free fall. This can be avoided by constraining the input such that $u_1 > 0$ and the pitch and roll such that $\theta < 90^0$ and $\phi < 90^0$. The output space must be defined ensuring each component is not differentially related to another

component, as discussed in Secion 1.5.1. The translational positions of the vehicle are chosen as out components but this prevents velocity also being an output parameter as this is clearly differentially related to position. Instead the yaw angle (ψ) is chosen as the fourth component. The output space will be parameterized using a suitable choice of basis function and will be therefore defined as $[\hat{x},\hat{y},\hat{z},\hat{\psi}]$. As the system is differentially flat, the full state and control space can now be expressed as a function of these parameterized outputs and their derivatives:

$$x = \hat{x} \tag{3.7}$$
$$y = \hat{y} \tag{3.8}$$
$$z = \hat{z} \tag{3.9}$$
$$\dot{x} = \dot{\hat{x}} \tag{3.10}$$
$$\dot{y} = \dot{\hat{y}} \tag{3.11}$$
$$\dot{z} = \dot{\hat{z}} \tag{3.12}$$
$$\phi = \arcsin\left(\frac{-\ddot{\hat{y}}}{\sqrt{\ddot{\hat{x}}^2+\ddot{\hat{y}}^2+(g-\ddot{\hat{z}})^2}}\right) \tag{3.13}$$
$$\theta = \arctan\left(\frac{\ddot{\hat{x}}}{g-\ddot{\hat{z}}}\right) \tag{3.14}$$
$$\psi = \hat{\psi} \tag{3.15}$$
$$\dot{\phi} = \frac{\left(\dot{u}_1\ddot{\hat{y}}-u_1\dddot{\hat{y}}\right)}{(u_1)\sqrt{(u_1)^2-\ddot{\hat{y}}^2}} \tag{3.16}$$
$$\dot{\theta} = \frac{\dddot{\hat{x}}\left(g-\ddot{\hat{z}}\right)+\ddot{\hat{x}}\dddot{\hat{z}}}{\left(\left(g-\ddot{\hat{z}}\right)^2+\ddot{\hat{x}}^2\right)} \tag{3.17}$$
$$\dot{\psi} = \dot{\hat{\psi}} \tag{3.18}$$

where

$$u_1 = \sqrt{\ddot{\hat{x}}^2+\ddot{\hat{y}}^2+(g-\ddot{\hat{z}})^2} \tag{3.19}$$

$$\dot{u}_1 = \frac{1}{2}(\ddot{\hat{x}}^2+\ddot{\hat{y}}^2+(g-\ddot{\hat{z}})^2)^{-1/2}(2\ddot{\hat{x}}\dddot{\hat{x}}+2\ddot{\hat{y}}\dddot{\hat{y}}+2(g-\ddot{\hat{z}})(-\dddot{\hat{z}}) \tag{3.20}$$

The second control input $\dot{u}_2 = \ddot{\phi}$ can therefore be expressed:

$$\dot{u}_2 = \frac{N_{u2}}{D_{u2}} \tag{3.21}$$

where

$$\begin{aligned} N_{u2} = {} & (\ddot{u}_1\ddot{\hat{y}}-u_1\ddddot{\hat{y}})(u_1\sqrt{u_1^2-\ddot{\hat{y}}^2}) \\ & -(\dot{u}_1\ddot{\hat{y}}-u_1\dddot{\hat{y}})(\dot{u}_1\sqrt{u_1^2-\ddot{\hat{y}}^2}+u_1(\frac{1}{2}(u_1^2-\ddot{\hat{y}}^2)^{-\frac{1}{2}}(2u_1\dot{u}_1-2\ddot{\hat{y}}\dddot{\hat{y}}))) \end{aligned} \tag{3.22}$$

and

$$D_{u2} = (u_1\sqrt{u_1^2-\ddot{\hat{y}}^2})^2 \tag{3.23}$$

The third control input $\dot{u}_3 = \ddot{\theta}$ can be expressed:

$$\dot{u}_3 = \frac{N_{u3}}{D_{u3}} \tag{3.24}$$

where

$$N_{u3} = (\dddot{\hat{x}}(g-\ddot{\hat{z}}) + \ddot{\hat{x}}\dddot{\hat{z}})((g-\ddot{\hat{z}})^2 + \ddot{\hat{x}}^2) - (\dddot{\hat{x}}(g-\ddot{\hat{z}}) + \ddot{\hat{x}}\dddot{\hat{z}})(2(g-\ddot{\hat{z}})(-\dddot{\hat{z}}) + 2\ddot{\hat{x}}\dddot{\hat{x}}) \tag{3.25}$$

and

$$D_{u3} = ((g-\ddot{\hat{z}})^2 + \ddot{\hat{x}}^2)^2 \tag{3.26}$$

Finally the fourth control input is expressed as a function of the output:

$$\dot{u}_4 = \ddot{\hat{\psi}} \tag{3.27}$$

3.3.1 Differential flatness discussion

Differential flatness has been shown for the quadrotor between the output space ($[x, y, z, \psi]$) and the control inputs ($[u_1, u_2, u_3, u_4]$.) These control inputs are functions of the Euler angles in the earth axis and hence not equal to the actuator outputs ($[\tilde{u_1}, \tilde{u_2}, \tilde{u_3}, \tilde{u_4}]$) which are acting in the body axis. The actuator outputs can be assumed in some cases to approximate the control inputs (Castillo *et al.* 2005), this assumes small angle deviation in roll and pitch and fixed zero yaw angle. This approach is obviously limited however, for trajectory optimization in the four dimensional output space (Section 3.6) as the yaw angle is clearly not always zero and therefore this assumption no longer holds.

As seen within the differentially flat equations rapid equation growth is present as the differential flat equations are derived for the state and control spaces. For true differential flatness the actuator outputs would be calculated as a function of the output space, this would however result in further equation growth. Instead the differential flatness is considered only within the earth axis and hence simplifies the problem. The actuator outputs (body axis) are then calculated from the control inputs (earth axis) using the algebraic relationship in equations (2.32) to (2.35). For trajectory optimization the constraints are placed on the control inputs in the earth axis using a suitable approximation as discussed within Section 2.5. The whole problem can now be posed within the output space as opposed to the control space (Equation 3.1.)

$$\begin{aligned} &\min_{\mathbf{y}(t)} \Phi \quad \text{for } t \in [0, T] \\ &\quad s\text{s.t. } d(\mathbf{y}) \leq 0, \\ &\qquad \mathbf{x}_0 - h_1(\mathbf{y}(0)) = 0, \\ &\qquad \mathbf{y}_T - \mathbf{y}(T) = 0 \end{aligned} \tag{3.28}$$

Now the state is a function of the output space obtained from the differential flatness $h_1(\mathbf{y})$. Furthermore, any inequality constraints on the problem are also expressed as a function of the output space $d(\mathbf{y})$. These inequality constraints contain dynamic, control and environmental constraints such as maximum velocity, thrust limits and obstacles.

3.4 Constraints

Obviously when determining the optimal or quasi-optimal trajectory it is important that the trajectory is feasible. The trajectory must satisfy a set of equality and inequality constraints which ensure dynamic, actuator and obstacle avoidance constraints are met. Also obviously the vehicle must start at its current position and reach its destination at the predetermined flight time, therefore:

$$\hat{\mathbf{x}}_\mathbf{0} = \mathbf{x}_\mathbf{0} \tag{3.29}$$

where $\hat{\mathbf{x}}_\mathbf{0}$ is the initial parameterized state and $\mathbf{x}_\mathbf{0}$ is the initial state of the vehicle. Similarly:

$$\hat{\mathbf{x}}_\mathbf{T} = \mathbf{x}_\mathbf{T} \tag{3.30}$$

where $\hat{\mathbf{x}}_\mathbf{T}$ and $\mathbf{x}_\mathbf{T}$ are the parameterized state and required state at the predetermined time (T). As with any equations of motion, singularities exist at roll and pitch angles exceeding 90^o. While these can be avoided by adopting quaternions, it is unlikely that these angles will be exceeded with any rotor craft. Therefore to avoid these singularities within the differentially flat equations the roll angle and pitch angle are constrained. Furthermore singularities also occur at zero thrust and therefore this is also constrained:

$$-90^o \leq \phi \leq 90^o \tag{3.31}$$
$$-90^o \leq \theta \leq 90^o \tag{3.32}$$
$$u_1 > 0 \tag{3.33}$$

As discussed within Section 2.5 the constraints are approximated as functions of the control inputs:

$$u_1 \leq 5.4 \tag{3.34}$$
$$u_2 \leq 0.5 \tag{3.35}$$
$$u_3 \leq 0.5 \tag{3.36}$$
$$u_4 \leq 0.5 \tag{3.37}$$

As this model contains no aerodynamic effects there is no maximum speed within the model, therefore the maximum speed of the vehicle is constrained:

$$\sqrt{\dot{\hat{x}}^2 + \dot{\hat{y}}^2 + \dot{\hat{z}}^2} \leq 5(m/s) \tag{3.38}$$

3.4.1 Obstacle Modelling

In order for the trajectory to be feasible, the trajectory must avoid obstacles such as other vehicles, buildings and the ground. For computational simplification it is desirable to model all obstacles as convex smooth obstacles. In the simplest case this would be in the form of a sphere in which the inequality is presented as

$$R_s - \sqrt{(x-X)^2 + (y-Y)^2 + (z-Z)^2} \leq 0 \tag{3.39}$$

where R_s is the radius of the sphere, (x,y,z) are vehicle co-ordinates and (X,Y,Z) are obstacle co-ordinates. This is used within the simplest obstacle avoidance mission described within Section 3.1. Buildings on the other hand which are clearly not spherical are modelled as infinitely high cuboid approximations, as it is assumed they are too high to fly over.

$$R_s - \sqrt[N]{(x-X)^N + (y-Y)^N} \leq 0 \tag{3.40}$$

where N is a large number and as $N \longrightarrow \infty$ then the obstacle approaches a convex square. To avoid computational difficulties obviously N is not chosen to be ∞, instead a reasonable approximation is made by setting $N = 12$. Figure 3.1 shows the approximation of a square for $N = 12$. Obviously not all obstacles are circular or square, however, weightings can be

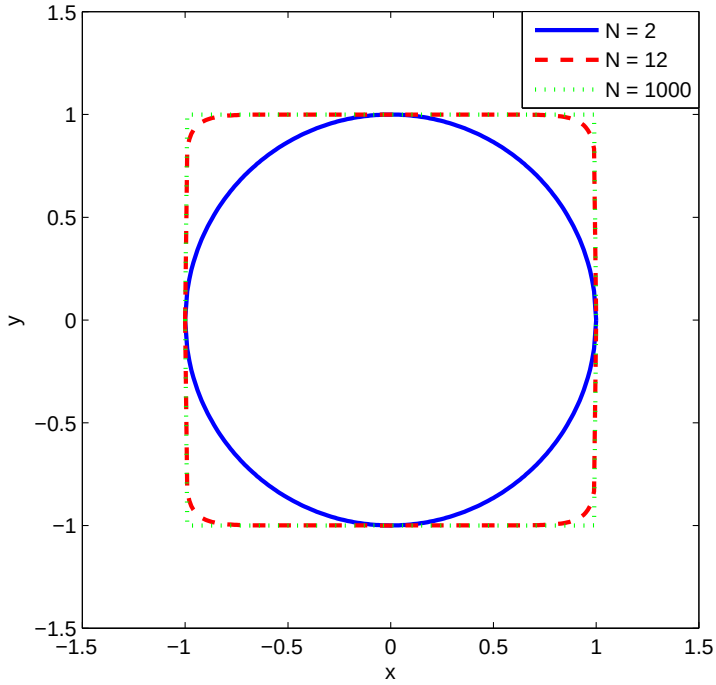


Figure 3.1: Constraint approximation of a square

applied to the approximations to obtain approximations for different shapes. By applying a simple weighting to the x and y coordinate a rectangular obstacle can be modelled

$$R_s - \sqrt[N]{\frac{(x-X)^N}{w_x} + \frac{(y-Y)^N}{w_y}} \leq 0 \tag{3.41}$$

where w_x and w_y are weighting factors. Figure 3.2 shows the approximation of various rectangles by varying these weightings. This approach is a simple method in which any obstacle avoidance can be modelled as a concave inequality regardless of the actual shape of the obstacle, spherical approximation accounts for a factor of safety for the trajectory and improves the convergence on an feasible solution.

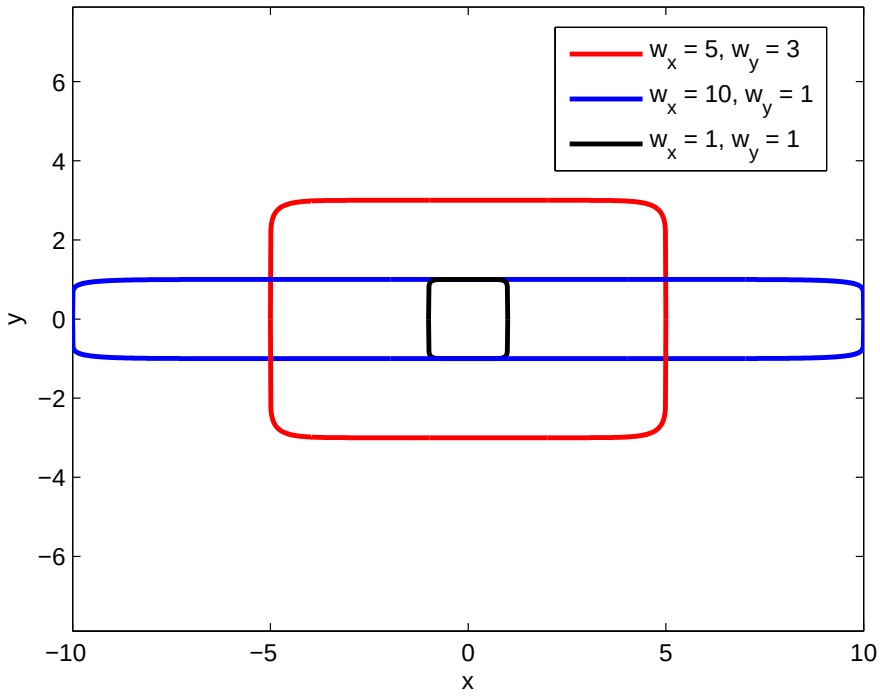


Figure 3.2: Constraint approximation of a rectangle

3.4.2 Terminal constraints

The optimization is required to minimize the cost function subject to the dynamic, initial and terminal constraints. This is solved numerically and it is found that the convergence properties of the algorithm can be greatly improved by relaxing the terminal constraints, this is done by creating a box around the destination in which the trajectory must finish. Equation 3.30 is therefore relaxed, such that

$$\begin{bmatrix} \hat{x}_T \\ \hat{y}_T \\ \hat{z}_T \end{bmatrix} = \begin{bmatrix} x_T \pm 0.25 \\ y_T \pm 0.25 \\ z_T \pm 0.25 \end{bmatrix} \tag{3.42}$$

where $[\hat{x}_T, \hat{y}_T, \hat{z}_T]$ are the relaxed terminal parameterized positions and $[x_T, y_T, z_T]$ is the destination coordinates.

3.5 Cost Function

Trajectory generation involves the minimization of some cost function, Φ, with respect to a set of constraints. The cost function is required to be some quantitative measure of optimality for any given trajectory. There are three classes of problem (Pierre 1969), the

Lagrange problem which is a function of the running costs over the mission

$$\Phi_L = \int_0^T f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{t}) \quad \mathbf{dt} \tag{3.43}$$

the Mayer problem which is a function of the terminal costs

$$\Phi_M = f(\mathbf{x}, \mathbf{t})|_{\mathbf{0}}^{\mathbf{T}} \tag{3.44}$$

and the Bolza problem which is a function of the running and terminal costs

$$\Phi_B = f(\mathbf{x}, \mathbf{t})|_{\mathbf{0}}^{\mathbf{T}} + \int_{\mathbf{0}}^{\mathbf{T}} \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{t}) \quad \mathbf{dt}. \tag{3.45}$$

For a predetermined flight time the vehicle should travel to the destination by taking the shortest possible route or use the minimum amount of fuel. There is obviously some correlation between shortest distance and minimum fuel, the further the vehicle travels the more fuel it needs. When considering missions such as mission (ii), then the relationship is less straightforward as dropping below an obstacle requires a different control action to going around it, although both are minimum distance. The difference is obviously the thrust required to overcome the gravitational pull, this is only acting in one plane and therefore the symmetry of minimum distance problem is lost.

In all three missions there is a desired terminal position, which requires the trajectory to finish at a certain height, this simplifies the problem somewhat. Consider mission (ii) as an example, the terminal state requires the vehicle to return to its initial altitude, therefore

$$\frac{1}{T}\int_0^T u_1 dt \approx mg \tag{3.46}$$

Now consider a minimum fuel cost function, in which the cost is a direct function of the total thrust u_1

$$\Phi = \frac{1}{T}\int_0^T u_1^2 P u_1^{2\prime} \tag{3.47}$$

where P is a weighting factor. It follows therefore that the minimum fuel problem in this case, with a fixed mission time, is essentially minimizing the deviation from the average required thrust (g). Recalling the differentially flat equation (3.4), it can be seen that minimizing u_1 is achieved by minimizing the vehicles acceleration as opposed to the minimum distance. In Chapter 5 this topic will be revisited but for now minimum distance is used for a number of reasons. Firstly the optimal solution for minimum distance is reasonably obvious, whereas minimum fuel requires careful consideration. The major advantage is however due to the output space optimization, as the problem is posed within the output space, by expressing the cost function as a direct function of the output space as opposed to the control space, the cost function is much simplified and does not include the large differentially flat equations (3.19-3.27). The following cost function is proposed and therefore the problem is formulated as a Lagrange problem:

$$\Phi = \frac{1}{T}\int_0^T (P_1\dot{x}^2 + P_2\dot{y}^2 + P_3\dot{z}^2)^{\frac{1}{2}} \quad dt \tag{3.48}$$

where P_1, P_2, P_3 are weighting factors, T is the predetermined mission time.

3.6 Parameterization

In order to reduce the optimization problem to a finite dimensional problem a parameterization of the output space is required. The simplest starting point for this is a polynomial or monomial (elementary polynomial) (Yakimenko 2006, Yakimenko 2000, Eikenberry *et al.* 2006, Nieuwstadt and Murray 1995), these are not necessarily the most effective solution however, as the conditioning is poor and there is a greater sensitivity at larger values of time. Other techniques which have been considered include Laguerre polynomials (Huzmezan *et al.* 2001), Chebyshev polynomials (Fahroo and Ross 2000, Vlassenbroeck and Van Dooren 1988) and polynomials obtained from the Taylor series expansion.

All techniques are essentially a product of a free variable (a_k) and a basis function Γ_k,

$$f(t) = \sum_{k=0}^{M} a_k \Gamma_k(t) \tag{3.49}$$

where M is the order of the basis function. The search space, assuming that there is no requirement to optimize the yaw angle (ψ), becomes $\mathbb{R}^{3(M+1)}$, with a 5th order basis function the problem becomes a 18 dimensional optimization problem. This in comparison is much lower than the search space within the work by Fahroo and Ross (2000), where the search space over G piecewise polynomials of magnitude M is $\mathbb{R}^{G(3(M+1)+1)}$ and although each polynomial may only be a 3rd order polynomial the dimension of the problem is still 130. While the dimension of the search space is not the only factor when considering convergence and the resulting computation time, with such a reduced search space other issues which are important within the work by Fahroo and Ross (2000) become less significant.

3.6.1 Laguerre polynomials

Laguerre polynomials which are derived from Laguerre's differential equations improve conditioning by weighting the individual terms of the simple polynomial. Laguerre polynomials can be derived from the following recurrence relationship.

$$\begin{aligned} \Gamma_0(t) &= 1 \\ \Gamma_1(t) &= 1 - t \\ (k+1)\Gamma_{K+1}(t) &= (2K+1-t)\Gamma_k(t) - k\Gamma_{k-1}(t). \end{aligned} \tag{3.50}$$

Laguerre polynomials (Γ_n, for $n = 0, 1 \ldots 5$) can be seen in Figure 3.3

3.6.2 Chebyshev polynomials

Chebyshev polynomials are used extensively within fluid dynamics as their cylindrical nature and the ability to determine both set of boundary conditions analytically make them

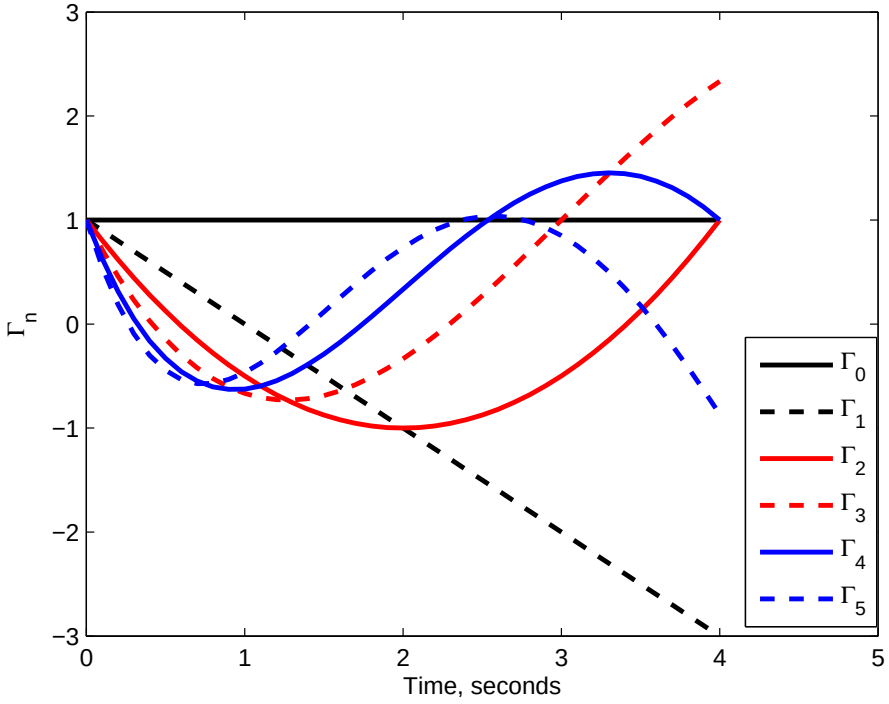


Figure 3.3: Laguerre polynomials

ideal for certain fluid flow modelling problems. This can be beneficial when considering trajectory optimization problems as the initial and terminal constraints can be determined analytically.

$$\Gamma_k(t) = \cos(ny) \quad \text{where } t = \cos(y). \tag{3.51}$$

Chebyshev polynomials (Γ_n, for $n = 0, 1...6$) can be seen in Figure 3.4

3.6.3 Taylor series polynomials

The conditioning of the polynomials can be also improved using weightings obtained from the Taylor series.

$$\Gamma_k = (1/k!)t^k. \tag{3.52}$$

The benefit of this can be seen when comparing the sensitivity to a change in the free variable at a large value of t. Starting with the basic polynomial:

$$\Gamma_k = t^k. \tag{3.53}$$

The sensitivity to a change in the free variable a_k is:

$$\frac{\partial \Gamma}{\partial a_k} = t^k \tag{3.54}$$

whereas for the polynomial obtained from the Taylor series this is:

$$\frac{\partial \Gamma}{\partial a_k} = \frac{t^k}{k!} \tag{3.55}$$

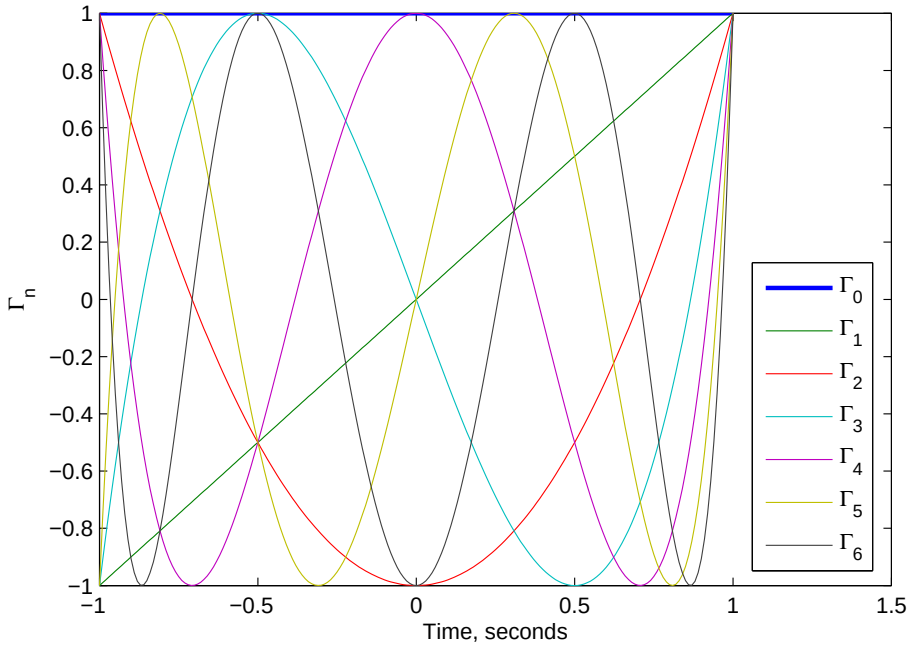


Figure 3.4: Chebyshev polynomials

To show the benefit of this improved conditioning, the sensitivity for both the Taylor polynomials and the standard polynomials, are shown in Figures 3.5-3.9. For different time values $(t = [1, 2, ...5])$ the sensitivity is shown at different powers (k). This shows the increased sensitivity for large values of k for the standard polynomial but this is reduced when the Taylor series weighting are applied.

3.6.4 Comparison

There are many potential parameterization techniques and therefore in order to come to a decision as to which is the best one to use, some method of testing is required. In Section 3.1, three missions are defined to test the optimization algorithm, it is therefore possible to test each basis function for each mission by solving the optimization problem stated in Equation 3.28. Essentially we are interested in finding the optimal path in the shortest possible time, assuming the total optimization time is a product of the number of iterations and the single iteration time then we can evaluate the effectiveness of each basis function. As the computation time for a single iteration is small and not particularly varied from one basis function to the next then it can be assumed we are interested in the technique which converges to the solution in the smallest number of iterations.

Table 3.1 shows the number of iterations for each mission with the four different basis functions. It is assumed that all techniques converge on the same optimal solution as the final cost function is the same. It can be seen that the basic polynomial generally requires more iterations than the other techniques to converge on the solution. The weighting from the Taylor series reduces the number of iterations for the Vertical mission especially but

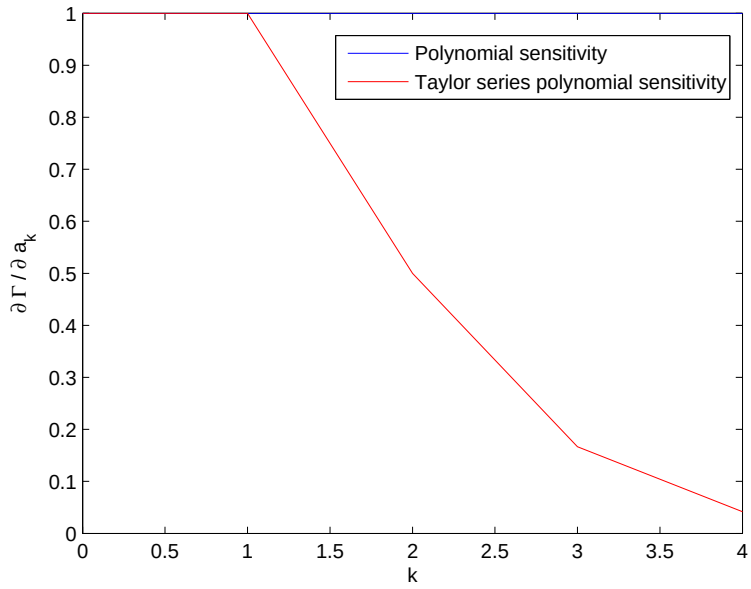


Figure 3.5: Improved conditioning with Taylor series (t=1)

the best results are achieved when using Laguerre polynomials and Chebyshev polynomials. Chebyshev polynomials are cylindrical in nature and therefore initial and final

Table 3.1: Number of iterations for various parametrization techniques

Iteration	The well	Vertical	Obstacle
Polynomial	30	85	17
Laguerre	21	33	18
Chebyshev	15	33	17
Taylor	30	30	20

boundary conditions can be determined analytically and applied to reduce computation time although this proves complex for variable horizon times. Laguerre polynomials are shown to be an efficient parametrization technique for this problem and therefore have been implemented.

3.6.5 Topology plots

Having decided upon a 5th order Laguerre polynomial and recalling the cost function (Equation 3.48) it is possible to plot small changes in the free variables within the parameterization against the cost function to get some idea of any potential convexity that exists within the optimization. The search space when optimizing over translational position and yaw angle becomes $\mathbb{R}^{4(M+1)}$ which obviously can not be shown on a single graph, instead just the six coefficients on the x position are varied in pairs $(a_0, a_1, a_2, a_3, a_4, a_5)$ to produce 3 sets of graphs (Figure 3.10).

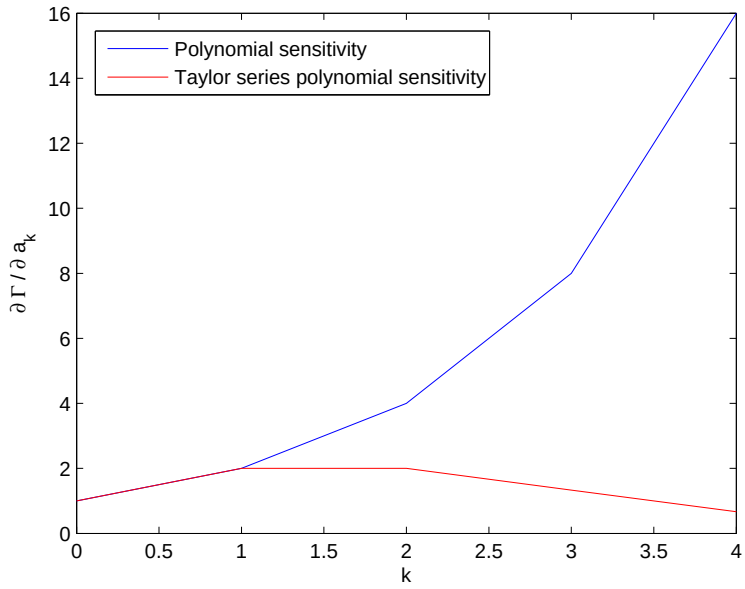


Figure 3.6: Improved conditioning with Taylor series (t=2)

These results indicate convexity over this region but also show the sensitivity variation within the Laguerre polynomials despite the improved conditioning due to weightings. These results can be compared to the cost function convexity using a basic polynomial parameterization (Figure 3.11). It is clear that the conditioning for the polynomial parameterization is very poor, as it appears that the cost function is almost totally dependent of only one of the two free variables in each of the plots. This is understandable considering the basic sensitivity analysis shown in Figures 3.5-3.9.

3.7 Results

Mission (i)

Figure.3.12 shows an example of the trajectory optimization for the vertical mission. Using Laguerre polynomials and an initial guess of $\lambda_n = 0_{4x6}$ the optimal path and states are calculated. Obviously the optimal horizontal displacement for a vertical flight when minimizing distance traveled is zero and therefore the optimization essentially determines the speed profile and required thrust to reach the destination at 7 seconds. The minimum time solution for such a problem is well known (Pontrjagin *et al.* 1962), where the optimal control signal appearing as a 'bang-bang' with maximum thrust followed by minimum thrust. The minimum fuel problem which converges to infinite time is small positive thrust followed by small negative thrust, the control profile in this case appears to be a flattened replica of the minimum time problem. With fixed time at 7 seconds it can be seen that the problem lies in between minimum time and minimum fuel and therefore the maximum thrust is considerably below the constraint of 5.4N and the 'flatter' control profile is

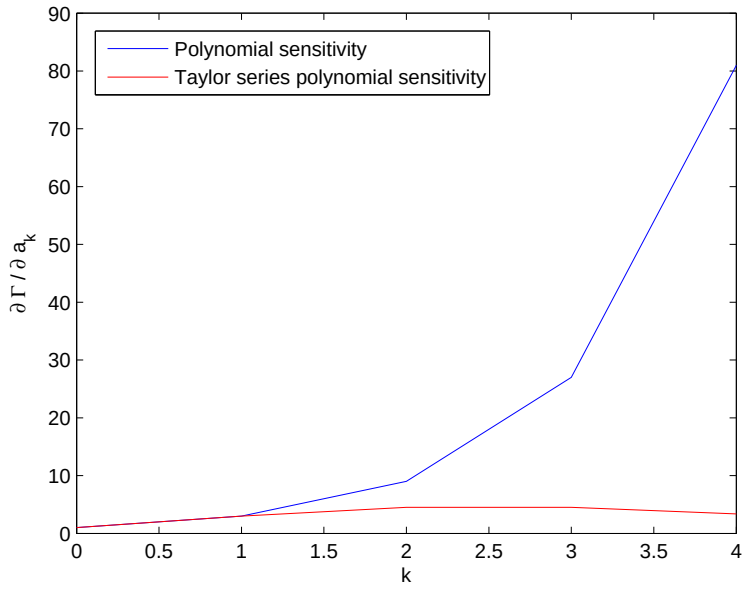


Figure 3.7: Improved conditioning with Taylor series (t=3)

evident.

Mission (ii)

The second mission is the obstacle avoidance and this can be seen in Figure.3.13. As seen the reference trajectory passes along the surface of the obstacle as this is the shortest route to the destination. As the cost function is a function of the distance traveled, the optimal trajectory is along the surface of the sphere but it could equally be underneath or around the side. The reference trajectory is therefore dependent on the starting point for the optimization, i.e the free variables' initial values.

Mission (iii)

The third mission is the mineshaft mission and this can be seen in Figure.3.14. This mission is possibly the most challenging mission in terms of finding a feasible solution as the walls severely increase the number of constraints within the optimization as the vehicle must pass down a thin mineshaft, this accounts for the increased computation time which will be discussed in the next section. If the diameter of the mineshaft was smaller, then a 5th order polynomial may struggle to meet the constraints, this could be solved by increasing the order of the polynomial although this in turn would increase computation time. An alternative to this problem would be to set waypoints to determine a trajectory to the top on the mineshaft and to then drop down the mineshaft after a new trajectory generation.

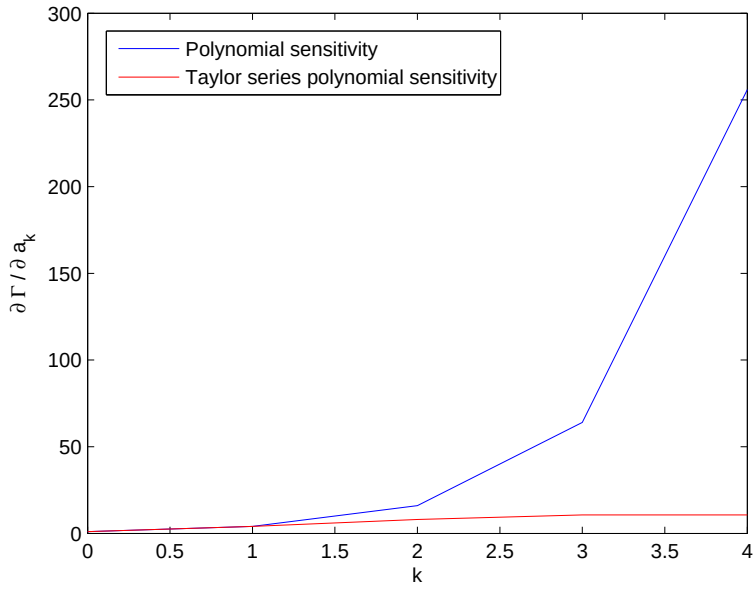


Figure 3.8: Improved conditioning with Taylor series (t=4)

3.7.1 Computation time results

The computation times for the optimization are calculated on a desktop PC with a 3GHz processor, 1GB of RAM and running a Windows operating system. The MATLAB function *fmincon* is used to optimize Equation 3.28 with 5th order Laguerre polynomials. The free variables require an initial guess at which the optimization search begins, in all three missions for the polynomials of order $\mathbb{R}^{4(M+1)}$ the free variable $\lambda_0 = 0_{4x6}$. At subsequent optimizations in the case the trajectory needs to be redetermined it is possible to use λ_{opt} as the starting point in the optimization and this generally reduces computation time.

Table 3.2: Computation time for 3 missions

Mission	Computation time
(i)	1.8 seconds
(ii)	2 seconds
(iii)	7.2 seconds

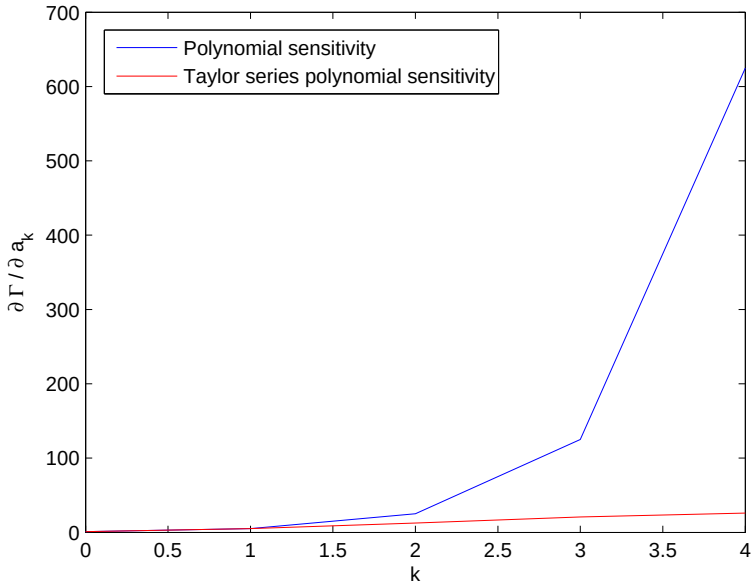
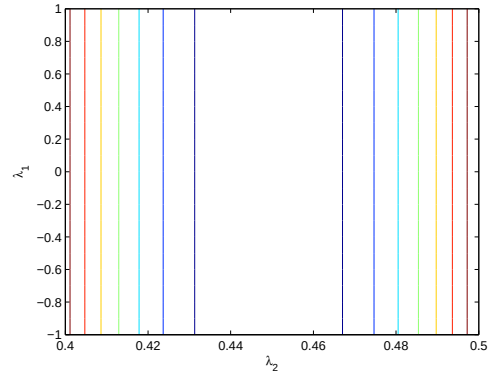
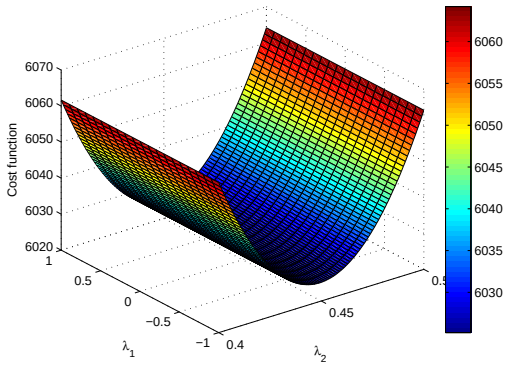


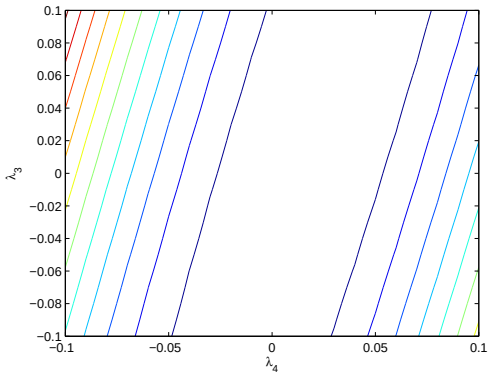
Figure 3.9: Improved conditioning with Taylor series (t=5)



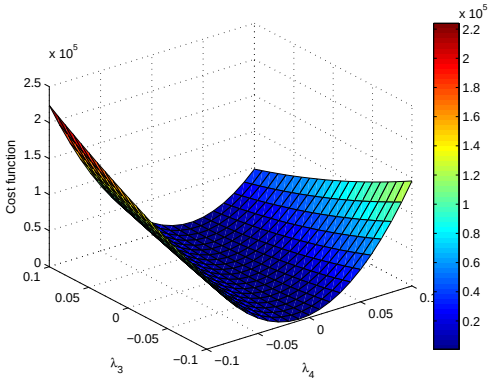
(a) λ_1, λ_2 contour plot



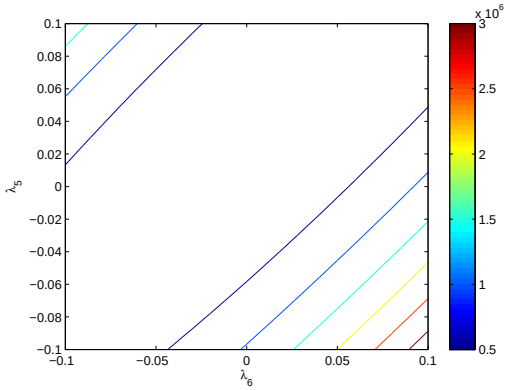
(b) λ_1, λ_2 surface plot



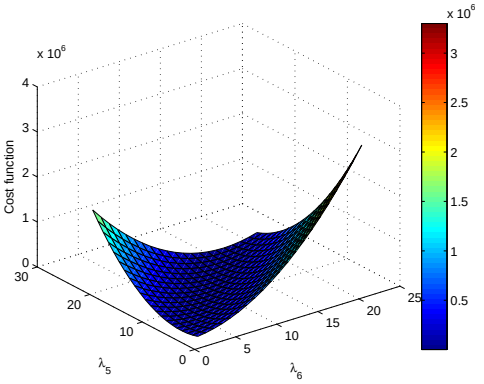
(c) λ_3, λ_4 contour plot



(d) λ_3, λ_4 surface plot

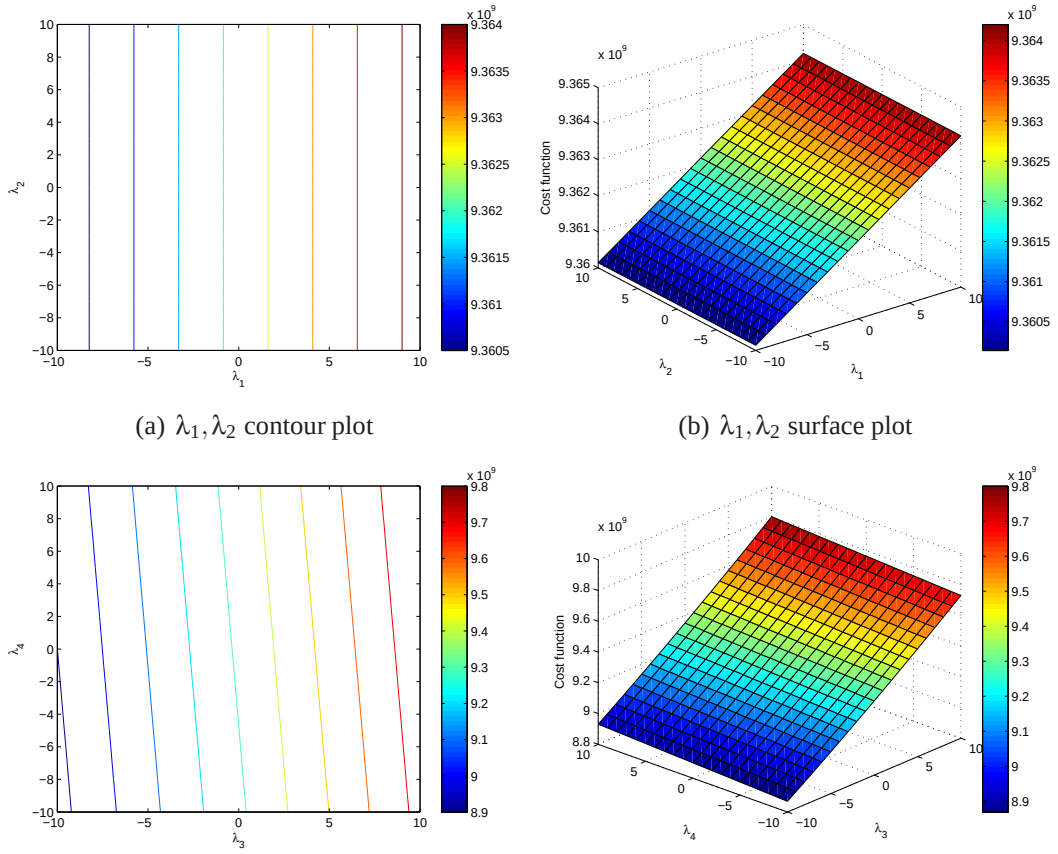


(e) λ_5, λ_6 contour plot



(f) λ_5, λ_6 surface plot

Figure 3.10: Cost function convexity using Laguerre polynomials as function of $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6$,



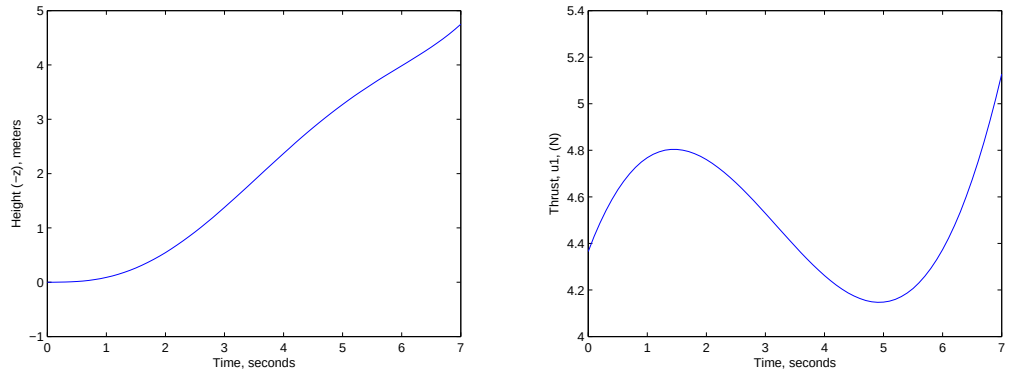
(a) λ_1, λ_2 contour plot

(b) λ_1, λ_2 surface plot

(c) λ_3, λ_4 contour plot

(d) λ_3, λ_4 surface plot

Figure 3.11: Cost function convexity using polynomials as function of $\lambda_1, \lambda_2, \lambda_3, \lambda_4$



(a) Height against time

(b) U_1 against time

Figure 3.12: Mission (i)

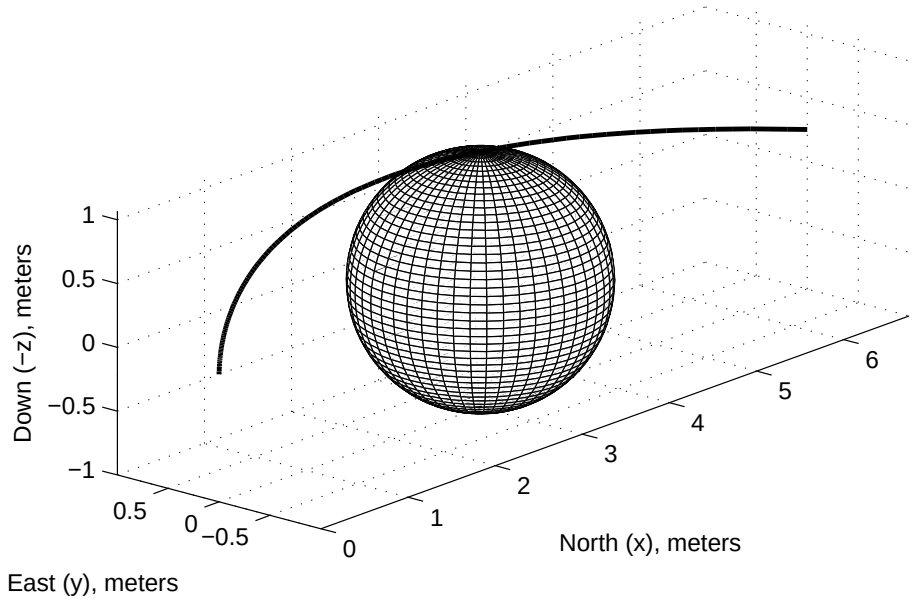


Figure 3.13: Mission (ii)

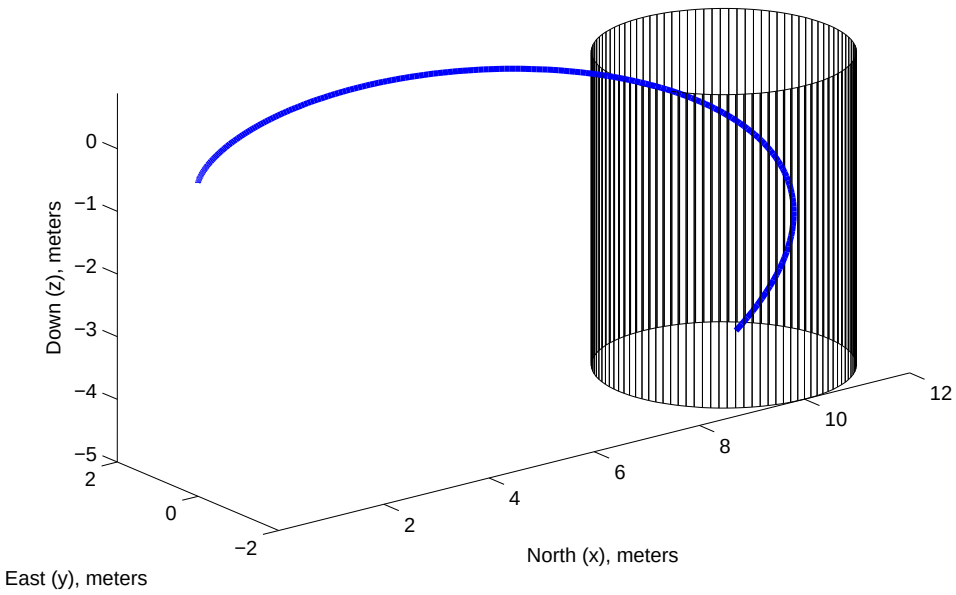


Figure 3.14: Mission (iii)

Chapter 4

Control Schemes

The previous chapter discussed trajectory generation of the quadrotor, by performing a non-linear optimization, within the output space, to calculate the reference state and control values, for a given mission. By feeding these reference control signals forward, it is theoretically possible to fly the vehicle open loop. In reality, this is obviously not the case however, as disturbances, noise and model uncertainty will cause tracking errors. Furthermore, the vehicle is unstable hence feedback is required to stabilize the system.

By repeatedly optimizing the trajectory at every time step, and updating state information through feedback, it is possible to close the loop and formulate a MBPC controller. MBPC optimizes the trajectory at each time step providing guaranteed constraint satisfaction. However the disadvantage of such a scheme is significant computational demand. Alternatively, a dual loop controller, as shown in Figure 1.5, can be used to achieve closed loop flight, with reduced computational demand. Trajectory optimization produces reference states and control for the duration of the flight, feeding forward these reference values and combining with a standard multivariable controller (LQR), produces a robust trajectory following controller. However, in the event of environmental changes or significant disturbances the reference trajectory may become infeasible and therefore it is necessary to determine a new optimal trajectory. The two schemes discussed therefore offer advantages and disadvantages, the MBPC scheme offers constraint satisfaction but with significant computational demand, whereas, the LQR inner loop does not guarantee constraint satisfaction but is less computationally demanding. It is therefore possible to consider a combination of the two control schemes where an outer loop trajectory generation is combined with an inner loop trajectory follower and these loops are controlled by a switch which monitors the trajectory feasibility.

This chapter consider both MBPC and singular trajectory generation with trajectory following and compares the results of both schemes using the three standard missions defined in Section 3.1. A dual loop controller is then developed which combines the benefits of both schemes and this is demonstrated using the same missions but with environmental changes, mission changes and model uncertainty to show the adaptability of the new scheme.

It is clear that the computation time for trajectory generation is of great importance. Within a gradient based optimization routine such as *fmincon*, a large percentage of the computation time is consumed calculating the gradients of the constraints and the cost function with respect to a change in the free variable. Computation time can be reduced significantly by supplying these gradients to the routine, however, determining these gradients analytically is a time consuming process. Automatic differentiation is a computational process, which, utilises the step by step nature of a computer program, and systematically uses the chain rule, to determine the derivative of a function. Automatic differentiation software is available in MATLAB (Forth 2006) and this is used to evaluate the gradients of the constraints and cost function and the computation time is compared with the analytical solution.

4.1 MBPC

A single trajectory optimization provides reference state and control values for the required mission. Closed loop control can be achieved by continually reoptimizing the trajectory and forming in effect a MBPC controller. MBPC is a process of successive optimizations over a given time horizon. The initial control input is inputed into the system, the process is then repeated at the next time step. Typically the time horizon is fixed and at every time step receeds, this is why MBPC is sometimes referred to as receeding horizon control, an excellent review of MBPC can be found in Mayne *et al.* (2000). The major benefits of MBPC is its online capability to satisfy constraints and therefore the real benefit of this for trajectory generation is it capability to handle dynamically changing environments with some measure of optimality. MBPC can be formulated for trajectory generation by optimizing the reference trajectory at every time step by solving Equation (3.28) over some time horizon. This formulation is a nonlinear problem and therefore requires non-linear MBPC. This presents several problems such as the convexity will be lost possibly resulting in local minima or ‘wild’ trajectories. Also there is no guarantee of computation time if a feasible solution is found at all. Furthermore MBPC is dependent on the accuracy of the model and therefore if the model changes due to for example a change in mass then the resulting control input may be unsuitable. The advantages and disadvantages can therefore be summarised:

MBPC advantages

- Constraint satisfaction at every time step
- Measure of optimality at every time step

Non linear MBPC disadvantages

- No guarantee of a feasible solution

- Sub-optimal or 'wild' solutions resulting from local minima
- Large computational demand
- Poor robustness to model inaccuracy

4.1.1 MBPC for trajectory generation

At any given time step t_k, given by;

$$t_k = t_0 + k\eta \qquad t_k \in [t_0\ t_f] \tag{4.1}$$

where η is the sampling time. The optimal trajectory is determined over $[t_k\ t_k + T]$ where T is a constant time horizon and the control action u_{k+1} is determined. This process is then repeated at the next time step t_{k+1}, over the time horizon $[t_{k+1}\ t_{k+1} + T]$ to determine the control input u_{k+2}. This is why predictive control is also referred to as receding horizon control. This can be applied to autonomy of UAV flight by successive optimization of the flight trajectory with respect to a set of constraints such as moving obstacles and actuator limits. The benefits of this control technique are therefore its ability to combine the tasks of the path planner and controller as well as its improved constraint handling abilities.

4.1.2 Time Horizon

The trajectory optimization determines the optimal path from the current state to the destination, this is obviously not therefore a fixed time, as is the norm for MBPC in which a fixed time horizon receeds into the horizon (hence receeding horizon control). The time horizon is initially set to the full flight time $T = [t_k\ , t_f]$ where $t_k = t_0$ and reduces with time, with the vehicle reaching the destination when $t_k = t_f$ and $T = 0$, in this case the time horizon is no longer receding. Thus the predictive controller has a varying time horizon which can be expressed as;

$$T = t_f - t_k \tag{4.2}$$

t_f can be set to any value and this depends upon the requirements of the mission although if the minimum time is required this can be determined off-line by starting at an arbitrary value, checking feasibility and adjusting the mission time accordingly and repeating until a minimum is found.

4.1.3 Algorithm

Having defined the cost function, constraints and time horizon it is possible implement the control algorithm. At each time step t_k for $k = [0, 1.... f]$:

1. Obtain measurement $\mathbf{y}_k$;

2. Minimize Φ, subject to constraints as in Equation (3.28);

3. Determine control input $\mathbf{u}_{k+1}$;

4. Repeat until $T = 0$.

In practice for small values of the time horizon $T < 0.5$s, it is very hard to find optimal feasible solutions. This is improved by relaxing the terminal constraints with the addition of a destination box as discussed in Section 3.4.2. Despite this however a feasible solution may not be found, in this case a practical solution is to use a linear control scheme to follow a reference trajectory determined by the last feasible solution. When optimizing the trajectory it is necessary to provide a start point for the search (λ_0) from which the optimal trajectory is found (λ_{opt}). Initially this is an arbitrary initial guess ($\lambda_0 = 0_{4\times 6}$), but for values $k > 1$ the search time can be significantly reduced by recalling the previous optimal solution ($\lambda_0(k) = \lambda_{opt}(k-1)$).

4.1.4 Feasibility and stability

Stability for non-linear MBPC is generally very difficult to show (Maciejowski 2002). Global stability requires that a feasible solution can always be found and although this could be shown for very simple missions, when obstacle avoidance is considered this is certainly not a trivial problem. Non-linear stability can be shown in the case where a feasible solution is always possible by the addition of terminal constraints and convergence to these constraints through a Lyaponov function. Although this work contains no non-linear stability proof, stability is demonstrated through simulation and terminal constraints are also placed on the trajectory.

4.1.5 MBPC results

Mission (ia)

The first mission was the vertical flight, to climb 5m in 7 seconds. For all missions the sampling rate is 10Hz and all target destinations have a tolerance of 25cm and this is modeled as a box around the destination. The result of this test can be seen in Figure 4.1. A feasible solution is found at every time step until $T < 0.5$s. At this point the vehicle is quite close to the target, and using the last feasible solution for the control, results in the vehicle passing through the target box at 7 seconds. It should be noted that the scheme generally does not find feasible solutions when the horizon gets small, but the vehicle is usually very close to the target at this time.

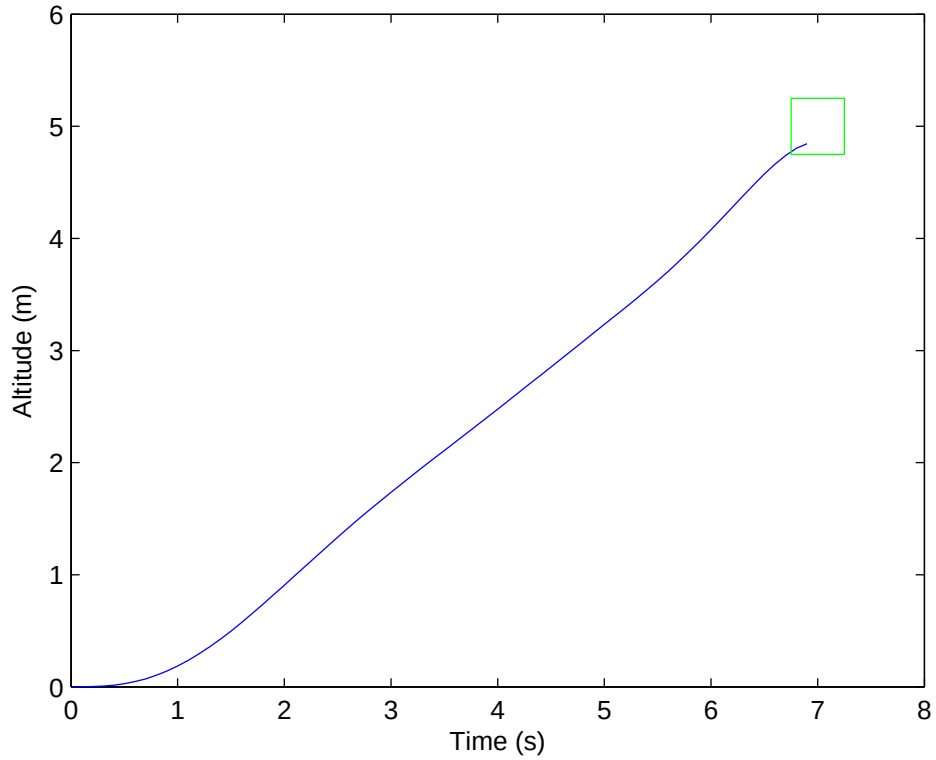


Figure 4.1: MBPC for vertical climb

Mission (iia)

The second mission involves a flight to a target 6m north of the start point within 15 seconds. An obstacle modeled as a sphere with radius 1m is centered at $(3, 0, 0)$. The flight path can be seen in Figure 4.2. As disturbances such as wind are present in the simulation then the optimal path may change in flight, as discussed this is a major advantage with MBPC. This plot shows an initial optimal trajectory but due to a tail wind blowing the vehicle towards the obstacle, the optimal trajectory changes at the next sampling interval. The vehicle then attempts to follow this trajectory but due to the continual tail wind and resulting errors it is necessary to constantly replot the trajectory. The optimal path is very close to the obstacle as seen, this is due to the optimization routine attempting to find the shortest route to the destination. In practice a factor of safety would be required when defining the obstacles.

Mission (iiia)

The final mission involves a flight to a waypoint at the top of a mineshaft in a strong wind followed by a vertical flight down to the bottom of the mineshaft. There is no wind disturbance in the mineshaft. The flight path can be seen in Figure 4.3. After the first optimization the vehicle takes into account the disturbance due to the strong wind. As

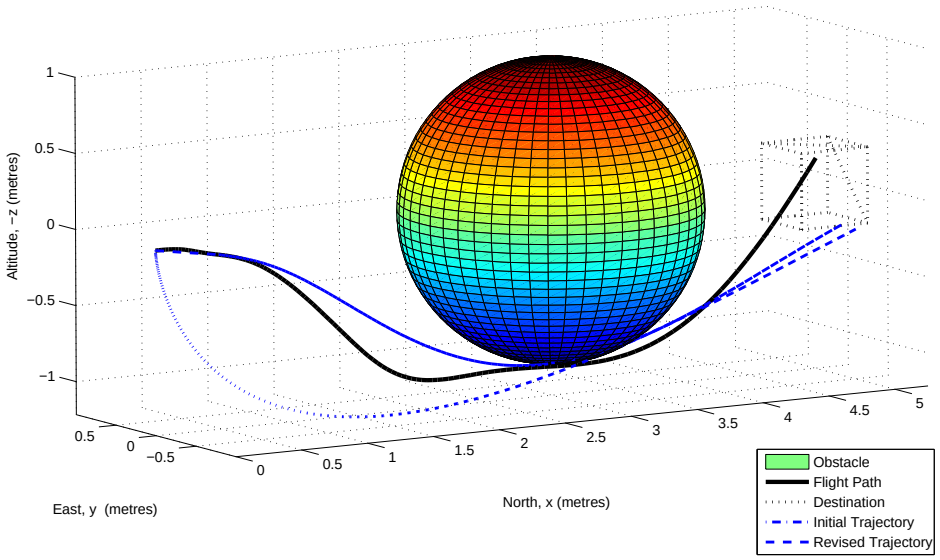


Figure 4.2: MBPC for obstacle mission

the vehicle has a induced velocity due to the tail wind there is no need to pitch to reach the top of the mineshaft, instead the vehicle drifts in the wind to the top of the mineshaft whereupon, in order to slow down, it pitches and descends down the mineshaft. Within the mineshaft however the wind drops, leaving the vehicle pitching and drifting towards the mineshaft wall. At this point it is necessary to pitch the other way to avoid collision and to return to the center. This mission highlights the advantage of MBPC over a linear tracking controller over a pre-defined trajectory, as it accounts for environmental changes such as the presence of a strong wind.

Computation time

The computational demands are the obvious disadvantage when considering MBPC, and this is the major contributing factor, as to why its application in industry is limited to the process industry where the time constants are typically large (Al Seyab 2006). The quadrotor on the other hand has short time constants making real time optimization a very challenging problem. The average computation time for each trajectory generation presented in this section is under 0.5 seconds, on a standard desktop PC, but this is still not sufficient to operate at 10Hz. Significant reductions in computation time can be achieved using techniques such as automatic differentiation, as will be discussed within Section 4.4. However, even with significant computational reductions the computation time is still too large for realistic real-time MBPC operation. Also as the problem is a nonlinear MBPC problem a feasible solution is not guaranteed. In the event of an optimization not finding a feasible solution, it is possible to revert back to the previous optimal solution, and use

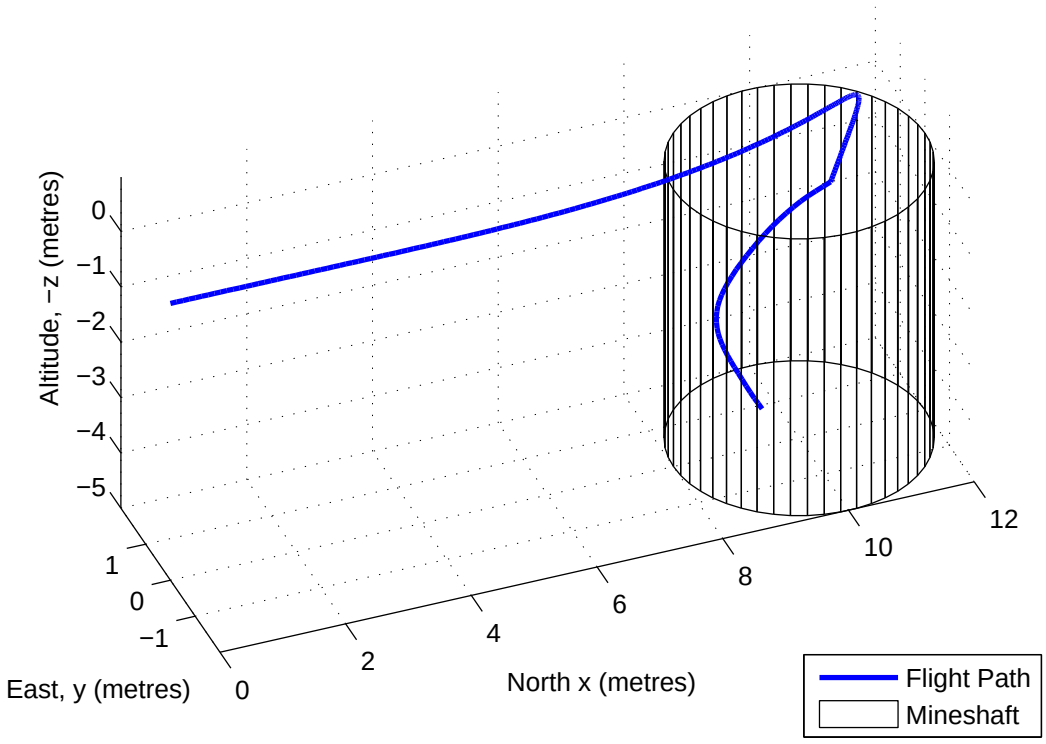


Figure 4.3: MBPC for mineshaft mission

the reference control signal, but this is in effect reverting back to open loop control which is not desirable, especially as the quadrotor is open loop unstable. These two problems require the investigation of an alternative approach such as an extra loop as previously shown in Figure 1.5.

4.2 Trajectory Following using LQR control

With a single trajectory optimization full optimal reference states and controls are found for the entire flight. Closed loop control can be achieved by feeding forward these values and applying a standard multi variable controller (LQR) to follow this trajectory as shown in Figure 1.5. Assuming a reference trajectory x_{ref} from the trajectory optimization, as discussed within Chapter 3, an LQR controller can be used to track the time dependent reference trajectory. This approach is of negligible computational demand once the reference trajectory has been determined compared to a MBPC scheme. Also once a trajectory has been determined a feasible solution exists which the vehicle can follow and is therefore not dependent on a new trajectory being determined before the next control action. There are obviously disadvantags with this approach as well such as a reliance on the initial reference trajectory optimality. Furthermore if the problem changes after the initial optimization, due to for example a new obstacle appearing, the control algorithm will not

reoptimize a trajectory to avoid this obstacle. As before the advantages and disadvantages are summarised below

LQR trajectory following advantages

- Negligible computational demand
- Does not require a feasible solution after initial optimization
- Inherent robustness and stability characteristics

LQR trajectory following disadvantages

- Could violate constraints due to environmental changes such as a moving obstacle
- Relies on the initial trajectory optimality

4.2.1 The LQR control problem

Linear Quadratic Regulator (LQR) control is a widely used control technique (Starin *et al.* 2001, Wu *et al.* 1998), which can be applied to linear state space systems

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t). \tag{4.3}$$

The control gains are found by solving the Riccati equation and minimising the performance measure (J),

$$J = \int_0^\infty (\mathbf{x}(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)\mathbf{R}\mathbf{u}(t)\,)\,dt \tag{4.4}$$

where **Q** and **R** are weighting matrices. The control input is then defined:

$$\mathbf{u}(t) = \mathbf{u_{ref}} - \mathbf{K}_c\mathbf{x}(t), \tag{4.5}$$

resulting in a stable closed loop system

$$\dot{\mathbf{x}} = [\mathbf{A} - \mathbf{B}\mathbf{K}_c]\mathbf{x}(t) + \mathbf{B}\mathbf{u}_{ref} \tag{4.6}$$

Now, assuming a time-dependent reference trajectory $x_{ref}(t)$, the LQR control can be applied as a trajectory follower to minimize small errors between the full measured state x and the reference state x_{ref} , such that the applied control becomes

$$\mathbf{u}(t) = \mathbf{u}_{ref}(t) - \mathbf{K}_c(\mathbf{x}(t) - \mathbf{x}_{ref}(t)) \tag{4.7}$$

The control gains $\mathbf{K}_c$ are determined with the plant linearized at hover (Equation 2.42 and Equation 2.43).

The weightings for performance measure can be chosen to minimize deviation from the reference trajectory or to minimize control action. For the non-normalized case where the **A** and **B** matrices are defined in (2.38,2.39), the **Q** and **R** can be used to minimize control action as well as ensure constraint satisfaction for the control inputs. By defining

$$\mathbf{Q} = \text{diag}(1\times10^{-5}, 1\times10^{-5}, 1\times10^{-4}, 1\times10^{-5}, 1\times10^{-5}, 1\times10^{-4}, 1\times10^{-3}, \quad (4.8)$$

$$1\times10^{-3}, 1\times10^{-3}, 1\times10^{-5}, 1\times10^{-5}, 1\times10^{-5}) \quad (4.9)$$

$$\mathbf{R} = \text{diag}(1\times10^{-5}, 1\times10^{8}, 1\times10^{8}, 1\times10^{8}) \quad (4.10)$$

the control gains are suitably low to reduce the likelihood of constraint violation. It is worth noting that this is not the same as setting the weightings to zero, as the **Q** and **R** matrix must remain positive definite. Alternatively the **A** and **B** matrix can be normalised with respect to the inertia of the vehicle and similar control gains can be determined using a identity matrix. The resulting gain matrix is:

$$\mathbf{K}_c = \begin{bmatrix} 0 & 0 & 3.16 & 0 & 0 & 4.04 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3.1\times10^{-6} & 0 & 0 & 2.5\times10^{-3} & 0 & 0 \\ -0.32 & 0 & 0 & -1 & 0 & 0 & 0 & 6.13 & 0 & 0 & 3.50 & 0 \\ 0 & 1.00 & 0 & 0 & 2.81 & 0 & 0 & 0 & 15.0 & 0 & 0 & 5.48 \end{bmatrix} \quad (4.11)$$

The closed loop stability of the system can now be analysed by calculating the eigenvalues of the closed loop $(\mathbf{A} - \mathbf{B} * \mathbf{K}_c)$, these are shown in Figure 4.4.

4.2.2 Stability Analysis

Clearly, to follow a trajectory, the system does not remain at hover. A simplified analysis is hence performed to determine an envelope of operation where the vehicle will remain stable. The analysis is not rigorous and is is hence only an indicator, however the analysis is simple and provides a convex bound on the state **x**. Stability for a calculated trajectory can be subsequently checked by simulation. From Equation 2.42 and Equation 2.43, the linearized dynamics depend on three variables, θ, ϕ and u_1. We define the linearized stability set $\mathcal{S}$ to be

$$\mathcal{S} = \left\{\theta, \phi : \alpha(\mathbf{A}(\theta, \phi, u_1) - \mathbf{B}(\theta, \phi)\mathbf{K}_c) < 0, 0.5 < u_1 < u_{1(\max)}\right\} \quad (4.12)$$

where $\alpha(\cdot)$ is the spectral abscissa (most positive real part of the eigenvalues). The set is plotted in Figure 4.6 . By inspection, we can fit a disk inside the set, hence it is clear that $\mathcal{S}_c \subset \mathcal{S}$ where

$$\mathcal{S}_c = \left\{\theta, \phi : \theta^2 + \phi^2 \leq r_s^2\right\} \quad (4.13)$$

with $r_s = 60^\circ$. $\mathcal{S}_c$ is also shown in Figure. An extra constraint can be inserted into the trajectory planner, which maintains the angles within this set and therefore ensures linearized time-invariant stability.

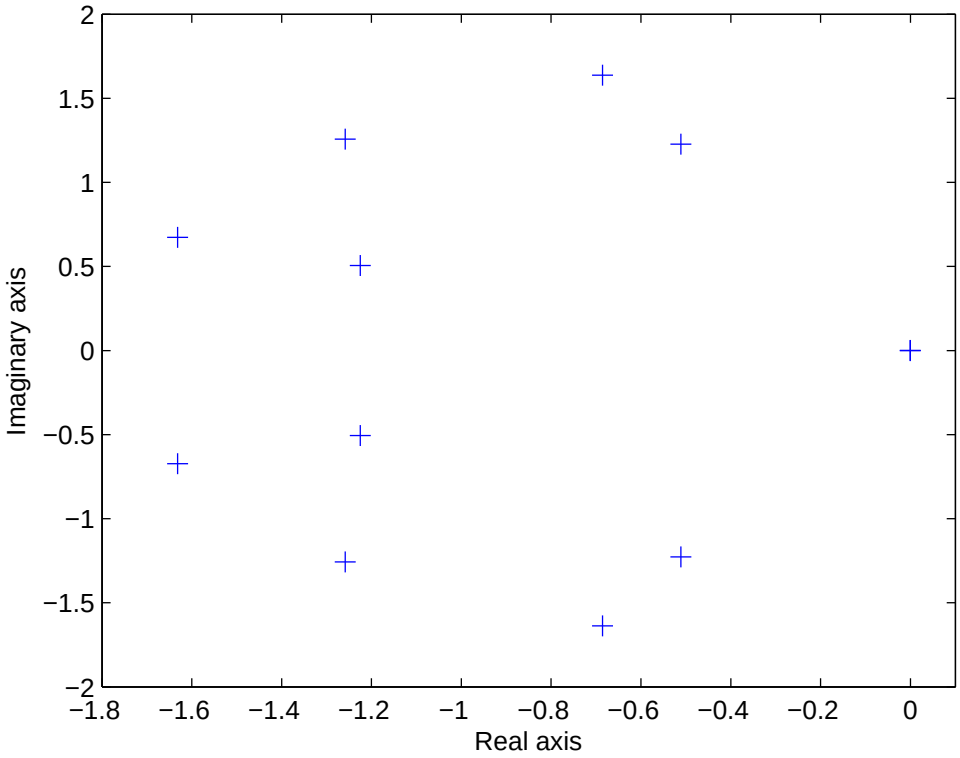


Figure 4.4: Closed loop poles with LQR control

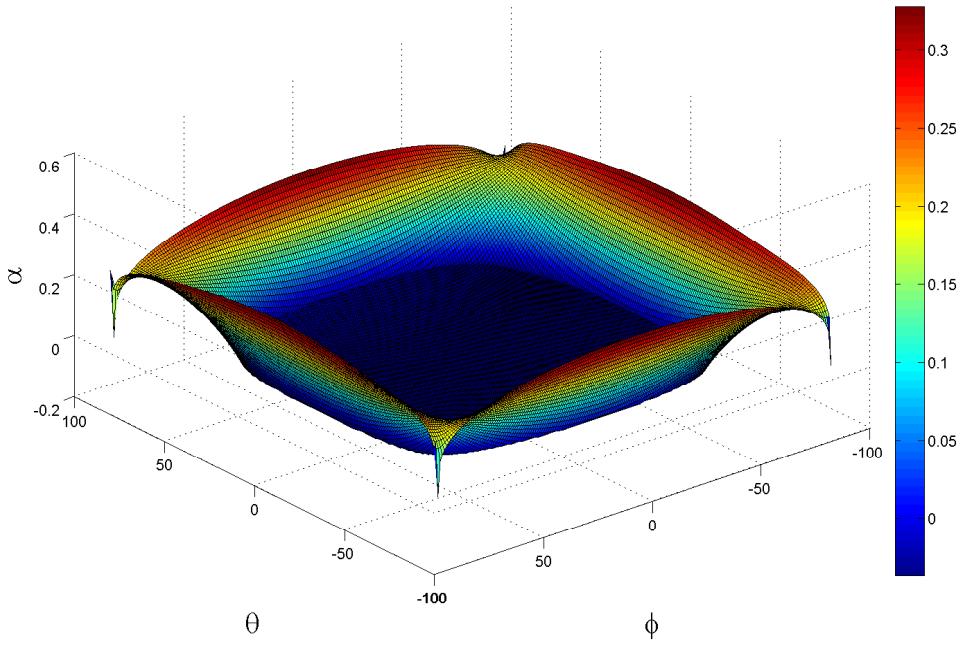


Figure 4.5: Stability surface for varying θ and ϕ

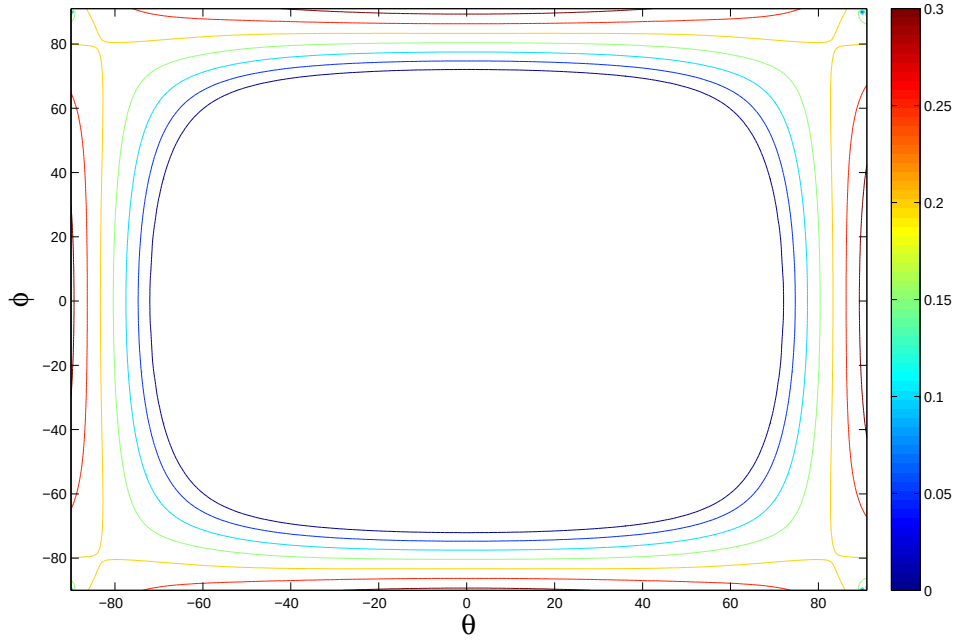
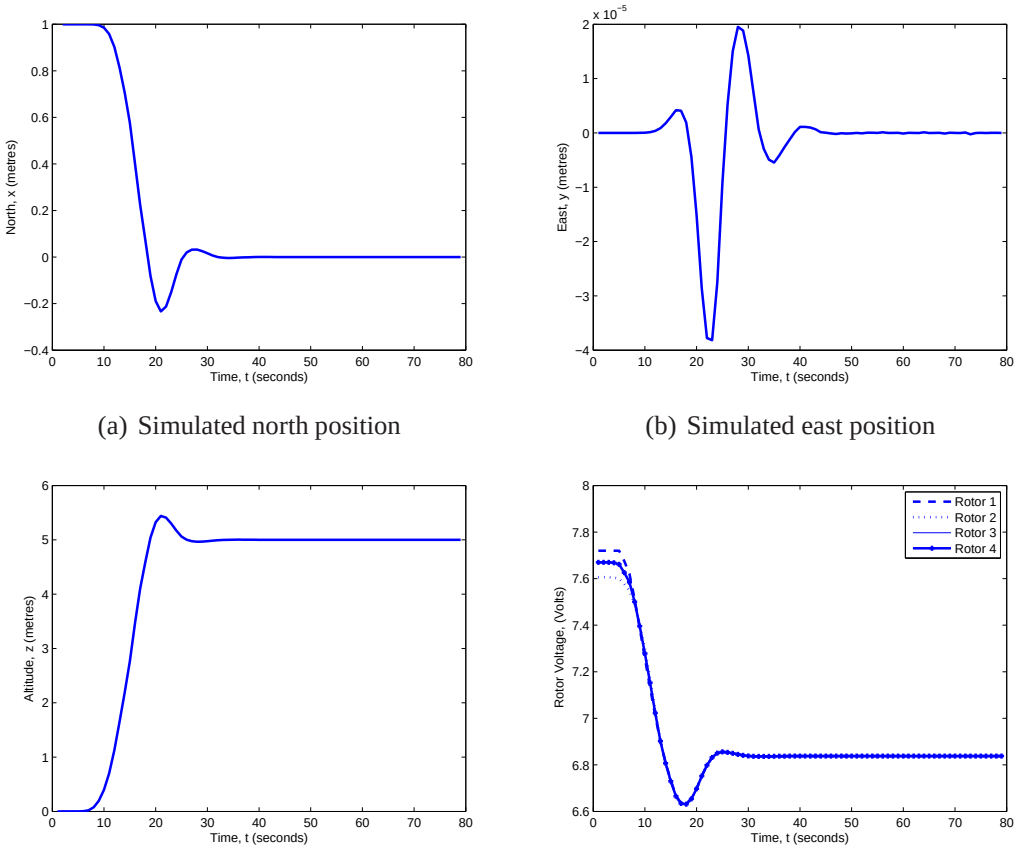


Figure 4.6: Stability region contour for varying θ and ϕ

4.2.3 Results

LQR control for hover of the Draganflyer X - Pro

Before attempting to attempt trajectory following of the smaller quadrotor, the larger quadrotor model detailed in Appendix B will be used to demonstrate a hover controller for the Draganflyer X-Pro. Essentially these models are the same except for some obvious differences such as the vehicles inertias, mass and rotor properties. Obviously as the model is different the required control gains will be different, these are determined in the same way the controller is designed for the smaller quadrotor but with different weightings. In this case the control inputs are normalized with respect to the vehicles inertia and mass and this allows a identity matrix to be chosen for the weightings without saturating the control inputs. The LQR weighting for the normalized model for this controller have been chosen simply as $\mathbf{Q} = I_{12}, \mathbf{R} = I_4$. The reference state is set such that the vehicle hovers at $[0,0,5]$, with the initial state set to $[1,0,0]$. For this simple hover simulation the wind is applied as a constant drift of 0.1m/s along the *x*, *y* and *z* axis, gusting is applied as a random noise with a mean value of 0.1m/s and a variance of 0.01m/s. There is also a time delay between the state measurement and the control action of 0.1 seconds to simulate the communication lags with the experimental set up described in Chapter 7. As seen in Figure 4.7, the vehicle flies to the hover condition within 30 seconds. The LQR weightings are chosen for demonstration and performance can be improved by modifying these weightings.



(a) Simulated north position

(b) Simulated east position

(c) Simulated altitude

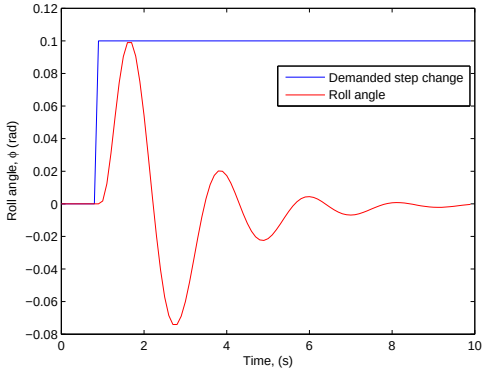
(d) Simulated rotor inputs

Figure 4.7: Quadrotor hover using LQR control

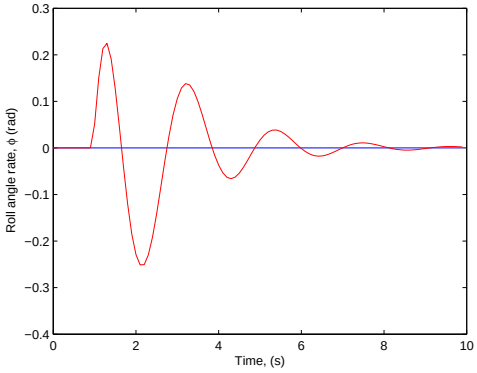
LQR control with step changes

It is typical to test a linear controller by inputing step changes in the demanded state of the system. This is not so straightforward for the quadrotor however. As the system is highly coupled a step change in one state is not ideal. For example, a step input into the roll angle alone, requires the vehicle to roll while maintaining zero roll rate and at a fixed position, clearly this is not possible. These step changes are considered using the smaller quadrotor model detailed in Appendix A. In Figure 4.8, initially the roll angle increases as demanded but this leads to a big error in the roll rate and the east position. To reduce these errors the vehicle the vehicle rolls the other way reducing the horizontal velocity. There is then a sequence of oscillations as the vehicle tries to reduce all the state errors which of course is not possible. The vehicle then returns to essentially a hover condition, whereby the demanded roll angle in not met but the other state errors are minimized.

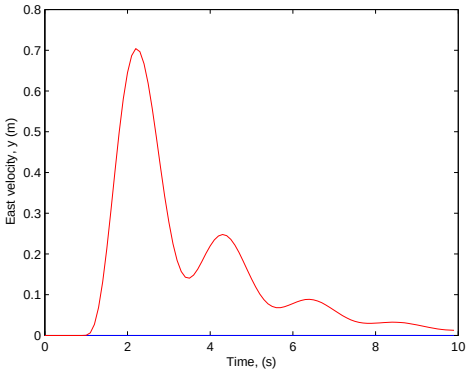
In a similar manner a demand in a change of position requires errors in other states to occur. In Figure 4.9 a change in position is required. This is possible as in this case the vehicle can hover at a different location although state errors are inevitable in achieving this. As before there is a quick initial response as the vehicle pitches. Again this leads to errors in other states however, so the vehicle pitches the other way to reduce these



(a) Simulated roll angle



(b) Simulated roll rate



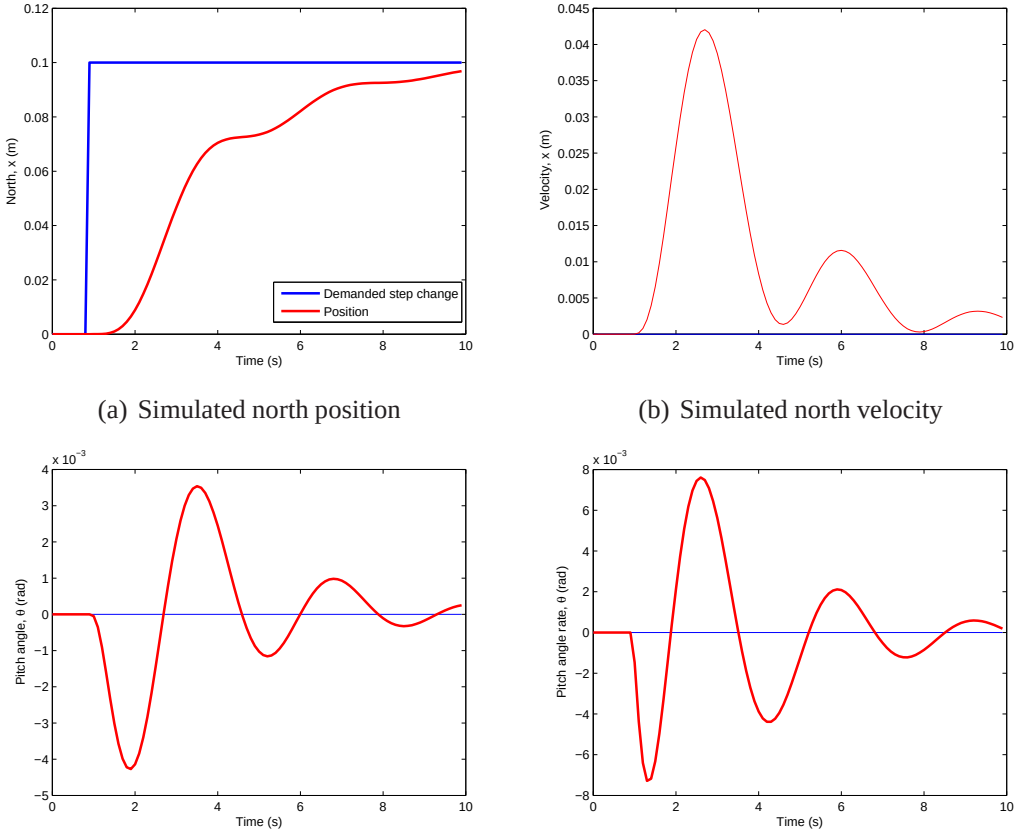
(c) Simulated east velocity

Figure 4.8: Roll angle step change

and again this leads to oscillations. In this case, convergence to the demanded state is possible although this is quite slow. These results show that the coupled nature of the dynamics, can result in poor tracking, of a step change in the vehicles state. To improve this tracking performance to a step change the LQR weightings could be modified to reduce this coupling effect, however, for reference trajectory following, where the full state is fed-forward this problem is reduced. It is likely however, that an state offset, could remain constant as the vehicle tracks the other states.

LQR tracking of reference trajectory

To demonstrate closing the loop with a LQR controller the three missions described previously will be shown. This includes a single trajectory optimization, the reference states and controls are then fed-forward to the LQR controller, which follows this trajectory. These results are simulated using the model of the smaller Draganflyer detailed in Appendix A.



(a) Simulated north position

(b) Simulated north velocity

(c) Simulated pitch angle

(d) Simulated pitch rate

Figure 4.9: North position step change

Mission (ib)

The vertical climb mission can be seen in Figure 4.10 and the error is plotted in Figure 4.11. To increase the chance of finding a feasible path a tolerance of 25cm is introduced into the terminal positional constraints. As seen the LQR controller tracks the reference trajectory well despite the presence of noise and disturbances.

Mission (iib)

The obstacle mission can be seen in Figure 4.12. To increase the chance of finding a feasible path a tolerance of 25cm is introduced into the terminal positional constraints. As seen the vehicle drifts slightly off the reference trajectory but this is understandable as a constant wind is acting on the vehicle. The reference trajectory passes on the surface of the obstacle as this is the shortest path to the destination, in reality a factor of safety would be introduced here. Despite the disturbances acting on the vehicle, the vehicle does not pass through the obstacle.

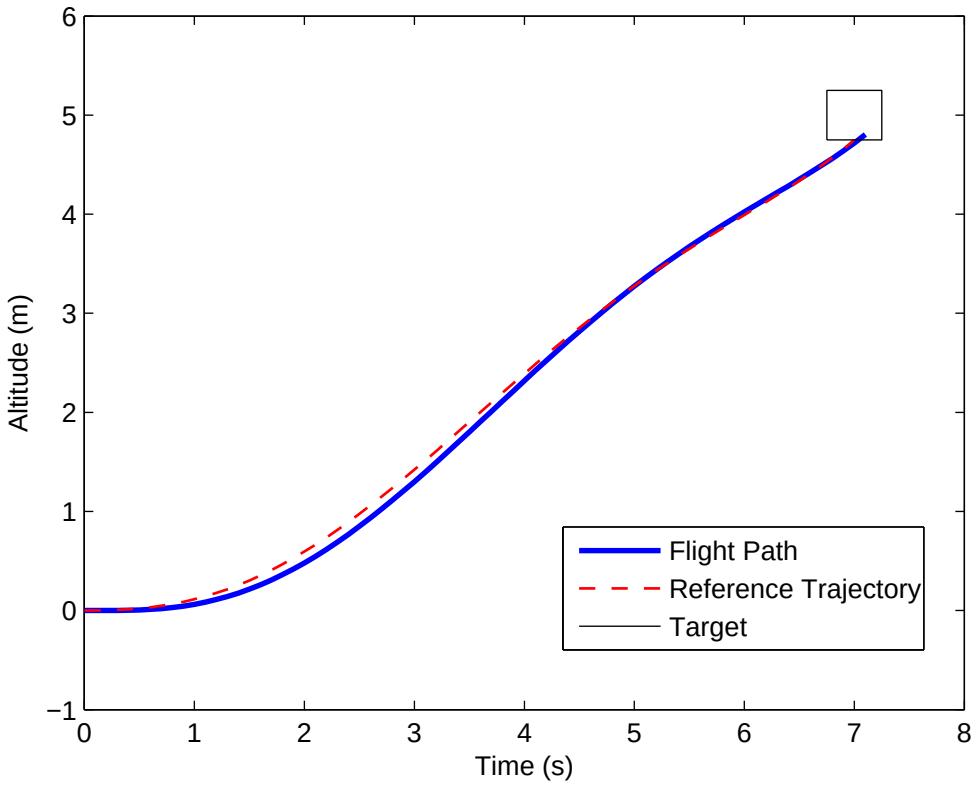


Figure 4.10: LQR trajectory following for vertical climb

Mission (iiib)

The final mission is the mineshaft mission where the vehicle must fly to the top of the mineshaft before dropping down to the bottom. As seen in Figure 4.13 the vehicle again drifts off the reference trajectory slightly due to the tailwind acting on the vehicle although this is only the case above ground. Despite this tailwind the vehicle reaches the bottom of the mineshaft without hitting the walls of the mineshaft.

4.2.4 Disturbances and model uncertainties

Up to this point we have considered a number of missions, with some moderate wind applied in one direction only, causing the vehicle to momentarily depart from the reference trajectory. Obviously this is not always going to be the case with the addition of a severe wind likely to prevent the vehicle from reaching its destination. To demonstrate the effect of various types of wind, mission (ii) is simulated. Figure 4.14 shows mission (ii) with the reference trajectory as before passing below the obstacle. Initially various steady winds are applied along the x axis in a northerly and southerly direction. As seen despite an initial large disturbance and subsequent tracking error the vehicle returns to the reference trajectory. A magnitude of $2m/s$ is recoverable in a reasonable time but if the

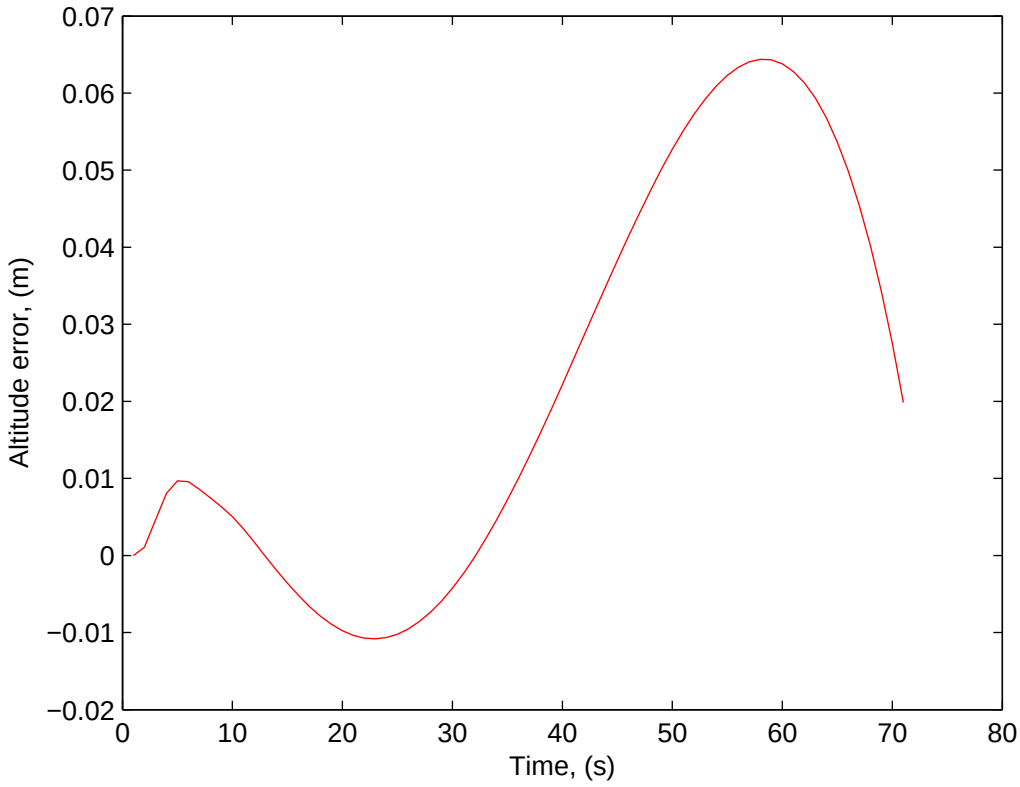


Figure 4.11: Vertical climb error

wind is stronger than this then the demanded control exceeds the control limits resulting in problems from controller saturation. It is unlikely that only a steady wind will be acting on the vehicle, there is also likely to be gusting acting on the vehicle, this is modelled as a random noise with a mean of 0 and variance of $0.01m/s$. The simulated flight path with gusting acting in all three axes is shown and it can be seen that the vehicle departs significantly from the reference trajectory. The vehicle is unable to recover from gusting with a variance of $0.1m/s$. Also shown is flight path with gusting acting in all three axis with variance of $0.01m/s$ and steady wind in all three axis of magnitude $0.1m/s$, as seen the vehicle tracks reaches the destination despite a constant tracking error.

At the start of the mission the vehicle is required to determine the optimal trajectory. This requires determining the current state, determining the optimal trajectory and then tracking this trajectory. Trajectory generation will take some time to calculate, and therefore, the vehicles state is unlikely to be the same as the initial conditions stated in the optimization algorithm, resulting in initial state errors. In Figure 4.15 the initial state is varied to model this time lag due to the trajectory optimization. A postion error in the x axis is shown, as seen the vehicle returns to the reference trajectory by flying back towards to the start point before proceeding with the forward flight. A position error in the y axis is also shown, here the vehicle is required to roll to minimize this positional error, again the vehicle quickly passes to the reference trajectory. Also shown is the case where the vehicle has an initial velocity along the x axis, this result appears to be very similar to the

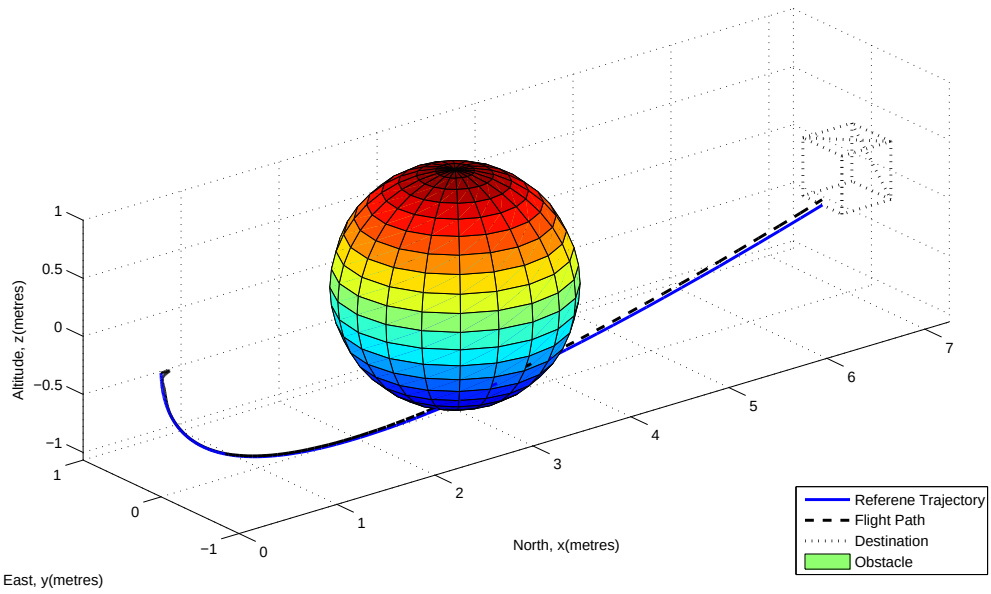


Figure 4.12: LQR trajectory following for obstacle mission

presence of a steady wind acting along the same axis, the vehicle is able to quickly return to the reference trajectory. Finally the flight path is shown where the vehicle is initialized with a large roll angle. This causes the vehicle to drift off the reference trajectory as the roll angle is corrected, the vehicle then rolls the other way to rejoin the reference trajectory and this is done reasonably quickly.

The scheme presented in this work is model based and up to this point the model is assumed to be correct. It is however, inevitable that the model will contain errors and inaccuracies due to modeling assumptions, it may also change due to the flight conditions or be required to carry a particular payload. For example the model presented includes the weight of the standard battery, sometimes a larger battery may be required or sometimes a smaller battery may be used. In this case the model could have significant errors. In Figure 4.16 the obstacle mission is again simulated but this time the vehicle's mass is changed to show the effect on the tracking controller. As expected if the mass is varied there is a large difference between the reference control value and that of the control required to track the reference trajectory. As the mass is increased from $0.44kg$ to $0.5kg$ the vehicle drops below the reference trajectory and is unable to track the reference trajectory accurately, this is due to the error in the reference control signals and the necessity to also track the reference speed profile, which it does after the initial drop in height. If the mass is reduced the vehicle climbs at the start as again the reference control signal is inaccurate. Although the vehicle then compensates for this change by reducing the total thrust the vehicle remains a constant distance from the reference trajectory, and in fact passes through the obstacle.

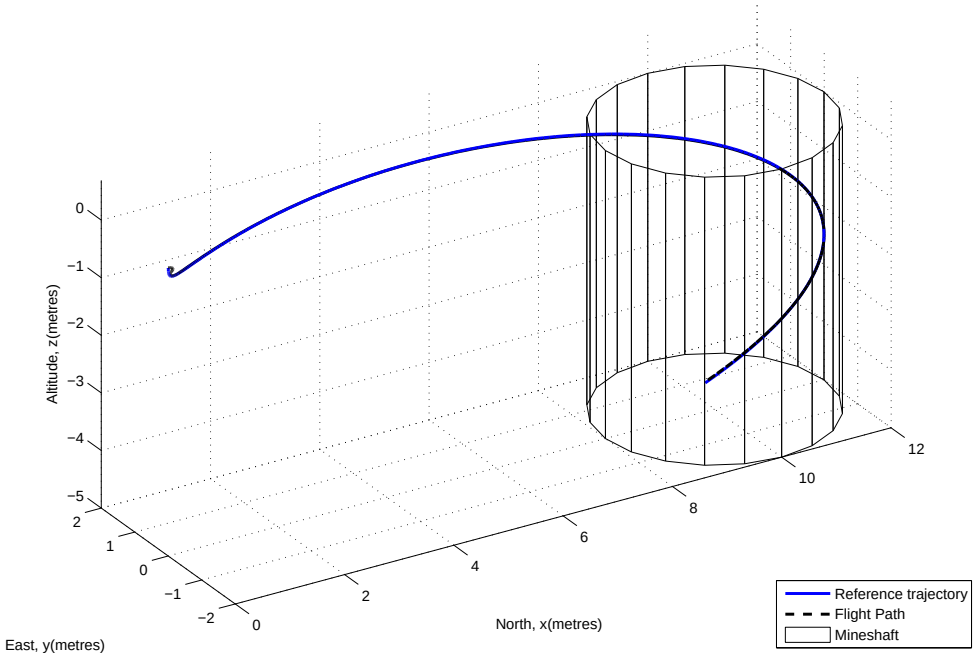


Figure 4.13: LQR trajectory following for mineshaft mission

4.3 Combined Control

In the previous section two schemes have been proposed. The first is a MBPC controller which re-determines the optimal trajectory at each time step. The second scheme is an LQR controller which follows a trajectory regardless of environmental changes or disturbances. The MBPC scheme provides optimal control with guaranteed constraint satisfaction and is also adaptive in terms of taking into account environmental changes. By closing the loop with a linear controller such as LQR the computational demand is reduced but constraint satisfaction is not guaranteed in the event of any environmental change.

This section discusses a combined controller which benefits from the online adaptability and constraint satisfaction of MBPC as well as the computational simplicity of LQR trajectory following. In Figure 4.17 a controller framework is presented consisting of two loops which in effect combines the two previous controllers in Figure 1.5 and Figure 1.6, this new framework is controlled by an update switch. LQR control forms the inner loop which can run at a high frequency (1 - 100Hz), trajectory generation forms the outer loop which determines a new optimal path when activated by the update switch and feeds the reference values through to the inner loop at a much lower frequency (0.01 - 10Hz). This controller therefore introduces no new additional terms but simply combines the MBPC algorithm presented in Section 4.1.3 and the inner loop LQR control from Equation 4.7. The only new addition to the theory so far presented in this work is in fact the switch. The switch itself compares the current state with the reference state as well as ensuring all constraints are satisfied. If at any point the vehicle drifts off the reference trajectory or

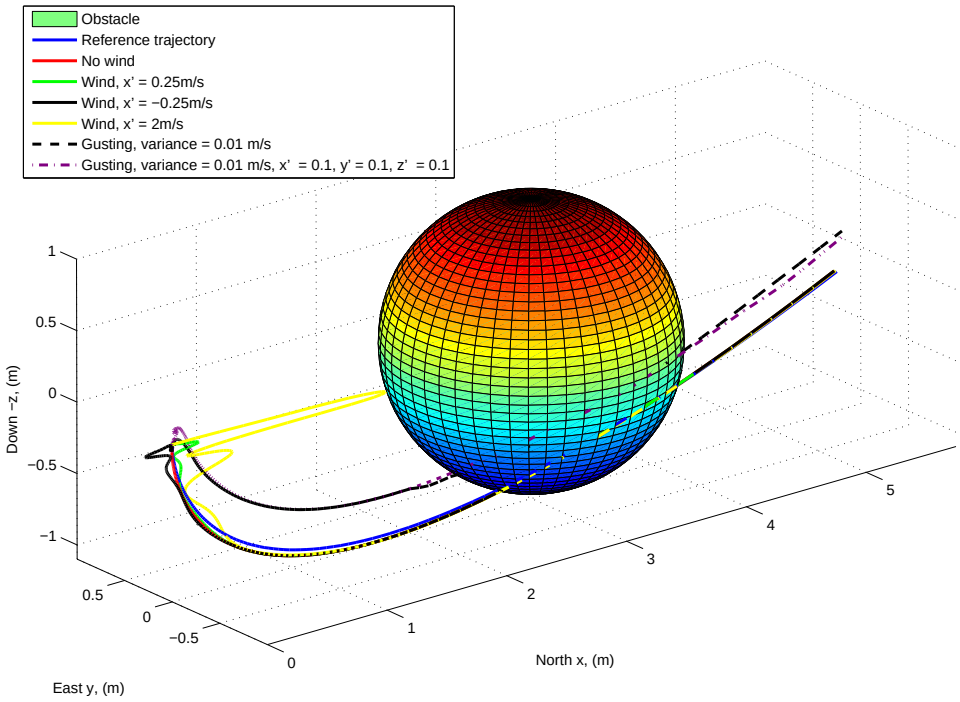


Figure 4.14: LQR trajectory following for obstacle mission with different wind strengths

a new obstacle appears so that the constraints are no longer satisfied the trajectory generation is switched on. The new algorithm now incorporates both loops, at each time step (k):

1. If k = 1, determine initial reference trajectory by solving Equation (3.28);
2. Check feasibility of reference trajectory against time varying constraints C(k);
3. If the reference trajectory is infeasible then reoptimize by solving Equation (3.28);
4. Follow reference trajectory by determining control input (Equation 4.7);
5. $T = T - \Delta T$, If $T \leq 0$ then end, else go to step 2.

4.3.1 Update switch

The update switch decides when the reference trajectory is to be reoptimized. There are two issues to consider when evaluating the existing reference trajectory, feasibility and optimality. Feasibility is easy to evaluate analytically by evaluating the reference trajectory against a set of constraints, which would include any new constraints introduced, due to for example, a new obstacle becoming visible. Optimality is harder to evaluate analytically, to determine the optimal trajectory the optimization problem has to be solved (Equation 3.28). Obviously it is desirable to determine and follow the optimum trajectory

but this is not always possible, in these circumstances a quasi optimal trajectory or even a feasible trajectory will suffice. Assuming an initial trajectory is feasible and has some degree of optimality then as long as this trajectory remains feasible then it will be followed. Potentially periodic optimizations may be performed on board to evaluate the trajectory optimization and evaluate the optimality but this is not possible at every time step.

4.3.2 Results

To test the control architecture three scenarios have been simulated with the full dynamic model.

Mission (iv)

Mission (iv) is bsaed on the obstacle mission with an environmental change. Unlike the previous example a second obstacle is detected after 5 seconds and therefore the initial trajectory becomes infeasible. In Figure 4.18 it can be seen that the initial trajectory passes under the obstacle. After 5 seconds the update switch detects the infeasibility of the reference trajectory due to the presence of a second obstacle and a new trajectory is determined (Figure 4.19). The vehicle then follows the new reference trajectory which passes over the top of the obstacles. In this mission there is a constant tailwind acting along the x axis of magnitude 0.1m/s. This result also shows the dual optimality of passing under or over the obstacle in this mission, the calculated optimum is dependent on the search direction within the optimization algorithm and the initial values of the free variable.

Mission (v)

Mission (v) is also based on the obstacle mission (Figure 4.18), but this time the mission scenario changes. In this case after a short time the desired destination moves from $[7,0,0]$ to $[7,2,2]$, this requires a reoptimization of the reference trajectory as the terminal constraints are no longer met (Figure 4.20). As before the LQR controller shows good reference tracking of the trajectory despite the presence of disturbances which is again a constant tailwind of magnitude 0.1m/s. As the new reference trajectory has initial conditions which are the vehicles current state then a smooth transition is evident in the flight.

Mission (vi)

Mission (vi) demonstrates the case where the initial information which is presented to the controller is incorrect, this mission is the mineshaft mission as seen in Figure 4.21. In this example after 5 seconds the position of the mineshaft moves from $[10,0,0]$ to $[12,0,0]$ making the reference trajectory infeasible. A new trajectory is then required as

seen in Figure 4.22 and this is then followed using the LQR controller despite the constant tailwind of 0.1m/s.

4.3.3 Combined control conclusion

A control structure has been presented which combines an inner loop LQR controller with an outer loop trajectory generation. These loops are controlled by an update switch which monitors the feasibility of the current reference trajectory. In the event of the trajectory becoming infeasible a new reference trajectory is determined in real time from the current state to the destination allowing for a smooth transition. In the event that the optimization routine is unable to determine a feasible reference trajectory then the current reference trajectory is tracked until a new reference trajectory can be determined, if there is an immediate threat of collision then the quadrotor is capable of hover.

4.4 Computation time

The major disadvantage of MBPC as discussed in Section 1.5.3 is its computational demand. This is possibly the main contributing factor to the reason MBPC has only had limited application and typically only within the process industry where the time constants are longer. Technological advances are enabling the deployment of MBPC into an increasingly wider range of applications. However, for a vehicle such as the quadrotor, significant computational time reduction is still required.

The trajectory optimization routine can be easily implemented using the *fmincon* within the MATLAB optimization toolbox. This optimization routine uses a sequential quadratic programming method (SQP) to find the optimal trajectory, this routine is however gradient based, requiring the constraint and cost function gradients to be calculated or supplied. Analysis of the *fmincon* algorithm shows a large percentage of the computation time consists of the constraint analysis and more specifically determining the gradient of these constraints with respect to the free variable λ.

4.4.1 Analytical Solution

Recalling the constraints are placed on the full state and control matrix then it is necessary at each evaluation of the constraints and cost function to also analyse the gradient of each state. The obvious starting point for this problem is to determine an analytical solution. There are 12 states and 4 control inputs and either 18 or 24 independent free variables (depending whether ψ is optimized or fixed), this therefore requires up to 384 analytical solutions of the gradient. It can be appreciated that providing so many analytical solutions by hand to some complex functions is not feasible and therefore alternatives are required. The analytical solution can be determined using mathematical software such as Mathematica or the symbolic toolbox within MATLAB, these solutions can result in

large equation growth and therefore lead to significant programming time and evaluation time within the optimization routine. Other techniques such as finite differencing provide an adequate solution but often increase computation time significantly and are subject to rounding errors.

4.4.2 Automatic Differentiation

Automatic Differentiation (AD) is an alternative approach to calculating the derivative analytically. Automatic differentiation was first considered in the 1960's (Wengert 1964) but later popularised by Rall (1981) and Griewank (1992). Automatic Differentiation is a systematic application of the chain rule to determine the gradient of a function (f). AD is similar to finite differencing in the fact it only needs the original function (f) to calculate the derivative. Instead of executing (f) on different sets of inputs it builds a new function (f'), that calculates the analytical derivative of the original function. This new function is typically referred to as the differentiated program. The scheme utilises the step by step nature of a computer program, as a elementary function is performed on the original program a corresponding function is performed on the differentiated program to obtain partial derivatives. The partial derivatives are then accumulated using the chain rule.

Example

As an example consider the first control input:

$$u_1 = \sqrt{\ddot{y}_1^2 + \ddot{y}_2^2 + (g - \ddot{y}_3)^2} \tag{4.14}$$

where the accelerations can be expressed as a function of the free variable λ and some basis function $\ddot{\Gamma}$

$$\begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \\ \ddot{y}_3 \end{bmatrix} = \begin{bmatrix} \lambda_{1n} \\ \lambda_{2n} \\ \lambda_{3n} \end{bmatrix} \ddot{\Gamma}_n \tag{4.15}$$

The analytical derivative with respect to λ_{1n}

$$\frac{\partial u_1}{\partial \lambda_{1n}} = \frac{1}{2}\left(\ddot{y}_1^2 + \ddot{y}_2^2 + (g - \ddot{y}_3)^2\right)^{-\frac{1}{2}} 2\ddot{y}_1 \frac{\partial \ddot{y}}{\partial \lambda_{1n}} \tag{4.16}$$

As seen the analytical solution leads to equation growth, this is only the gradient of u_1, the gradients of $\dot{u}_1$,$\ddot{u}_1$ and $\dddot{u}_1$ as well as the other states and derivatives are also required for analysis of the constraints. Clearly an alternative solution is desirable.

As elementary functions are performed on the program (v_n), a corresponding function is performed on the differentiated program (δv_n). For example v_1 is defined as the first free variable λ_{1n}, the differentiated program δv_1, is subsequently defined as $\delta\lambda_{1n}$. Now to

evaluate the first variable y_1, the free variable is multiplied by the basis function $\ddot{\Gamma}_n$. The differentiated program is updated in parallel, recalling $\ddot{y}_1 = \lambda_{1n}\ddot{\Gamma}_n$ the new differentiated function δv_2 is a product of the previous differentiated function ($\delta v_1 = \delta\lambda_{1n}$) and the basis function $\ddot{\Gamma}_n$. This process is repeated throughout the program as shown until the sensitivity for the complex function u_1 is calculated.

$$v_1 = \lambda_{1n} \qquad \delta v_1 = \delta\lambda_{1n} \tag{4.17}$$
$$v_2 = y_1 = v_1\ddot{\Gamma}_n \qquad \delta v_2 = \delta v_1\ddot{\Gamma}_n \tag{4.18}$$
$$v_3 = y_1^2 = v_2^2 \qquad \delta v_3 = 2v_2\delta v_2 \tag{4.19}$$
$$v_4 = \lambda_{2n} \qquad \delta v_4 = \delta\lambda_{2n} \tag{4.20}$$
$$v_5 = y_2 = v_4\ddot{\Gamma}_n \qquad \delta v_5 = \delta v_4\ddot{\Gamma}_n \tag{4.21}$$
$$v_6 = y_2^2 = v_5^2 \qquad \delta v_6 = 2v_5\delta v_5 \tag{4.22}$$
$$v_7 = \lambda_{3n} \qquad \delta v_7 = \delta\lambda_{3n} \tag{4.23}$$
$$v_8 = v_7\ddot{\Gamma}_n \qquad \delta v_8 = \delta v_7\ddot{\Gamma}_n \tag{4.24}$$
$$v_9 = y_2 = g - v_8 \qquad \delta v_9 = -\delta v_8 \tag{4.25}$$
$$v_{10} = y_3^2 = v_9^2 \qquad \delta v_{10} = 2v_9\delta v_9 \tag{4.26}$$
$$v_{11} = v_3 + v_6 + v_{10} \qquad \delta v_{11} = \delta v_3 + \delta v_6 + \delta v_{10} \tag{4.27}$$
$$v_{12} = \sqrt{v_{11}} \qquad \delta v_{12} = \frac{1}{2}v_{11}^{-\frac{1}{2}}\delta v_{11} \tag{4.28}$$

4.4.3 Results

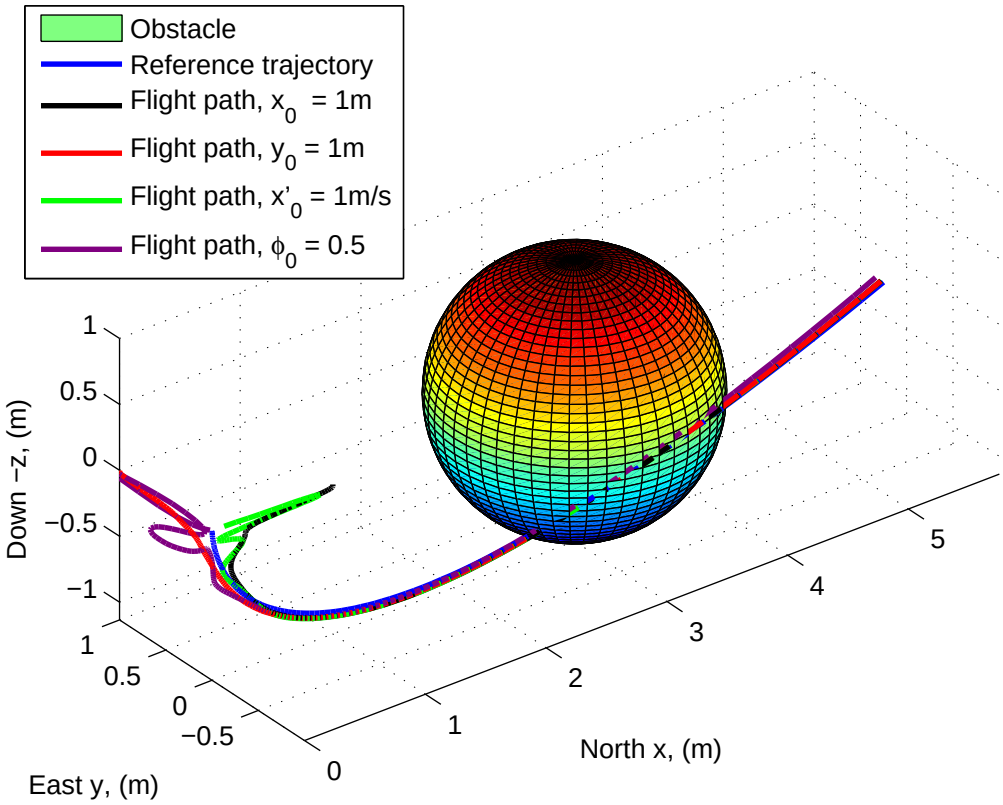
Automatic differentiation can be easily implemented into a MATLAB optimization routine using the MATLAB Automatic Differentiation toolbox (MAD). This is done by initialising λ as a *fmad* object, so that at every stage, the gradient is evaluated along with the function, as demonstrated in Equations (4.18-4.28). The computation times for the evaluating the gradients analytically and through the MAD toolbox can be compared by evaluating the script 100 times and then calculating the average computation time. The mean computation time for the analytical solution is 0.0058 seconds as opposed to the mean MAD time which is 0.1618 seconds. Although MAD does reduce the programming time as the analytical gradients do not need to be calculated these results demonstrate a significant increase in computation time which makes the application of MAD prohibitive.

It is worth considering the reason for this increase in the computation time. In Cao (2005) significant reductions in computation time are achieved using AD and therefore the increase in time is unlikely to be from the technique. The technique can be compared with the analytical solution by hand for a simple example. By calculating by hand the gradient of each state in a similar way as shown in Equations (4.18-4.28) the constraint gradients can be evaluated. Again the mean evaluation time for 100 repeats can be compared with the analytical solution, to reduce the programming time, ϕ and it derivatives are not evaluated. The mean average time for the analytical solution is 0.0028 seconds with the mean

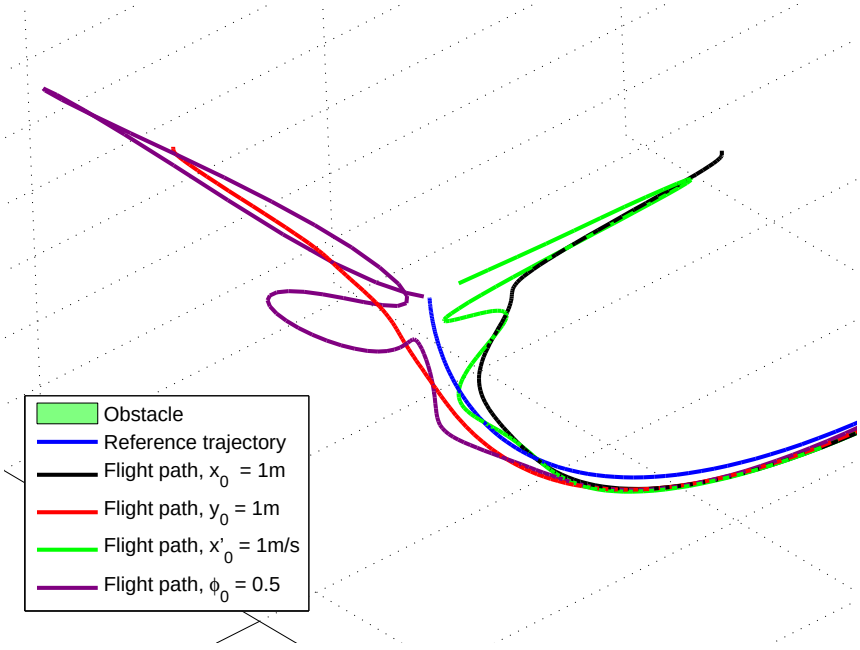
average time for the recursive relationship being 0.00025 seconds. This shows that the recursive process is not inherently any slower than the analytical solution. This also implies that any increase in computation time when using MAD is likely to be from the program as opposed to the method.

4.4.4 Automatic Differentiation conclusion

Automatic differentiation is a computational technique which can be used to provide a functions gradient to a optimization routine. Automatic differentiation can be implemented using any number of different software packages including MAD. In this example the overheads from the MAD toolbox negates any benefit from the AD technique. This does not necessarily show that AD can not be applied to calculate the gradients of the constraints within this MBPC algorithm. There are many alternatives to MAD to evaluate the sensitivity such as ADOL-C, which has been successfully applied for a similar application as demonstrated by Cao (2005).



(a) Full flight path



(b) Beginning of flight path

Figure 4.15: LQR trajectory following for obstacle mission with initial position errors

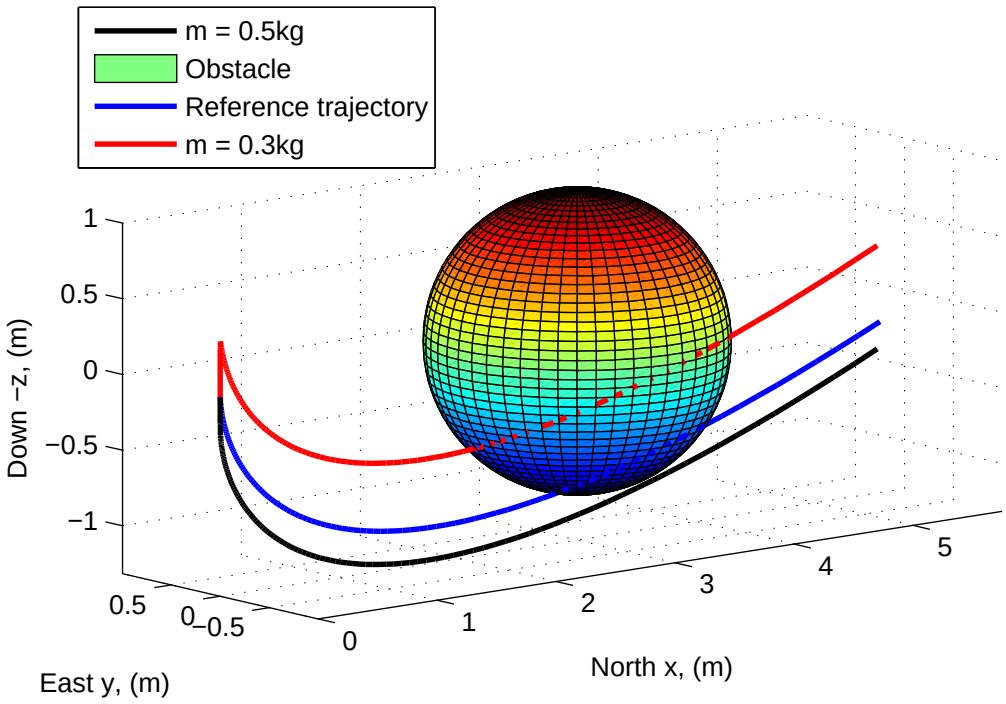


Figure 4.16: LQR trajectory following for obstacle mission with model innaccuracy

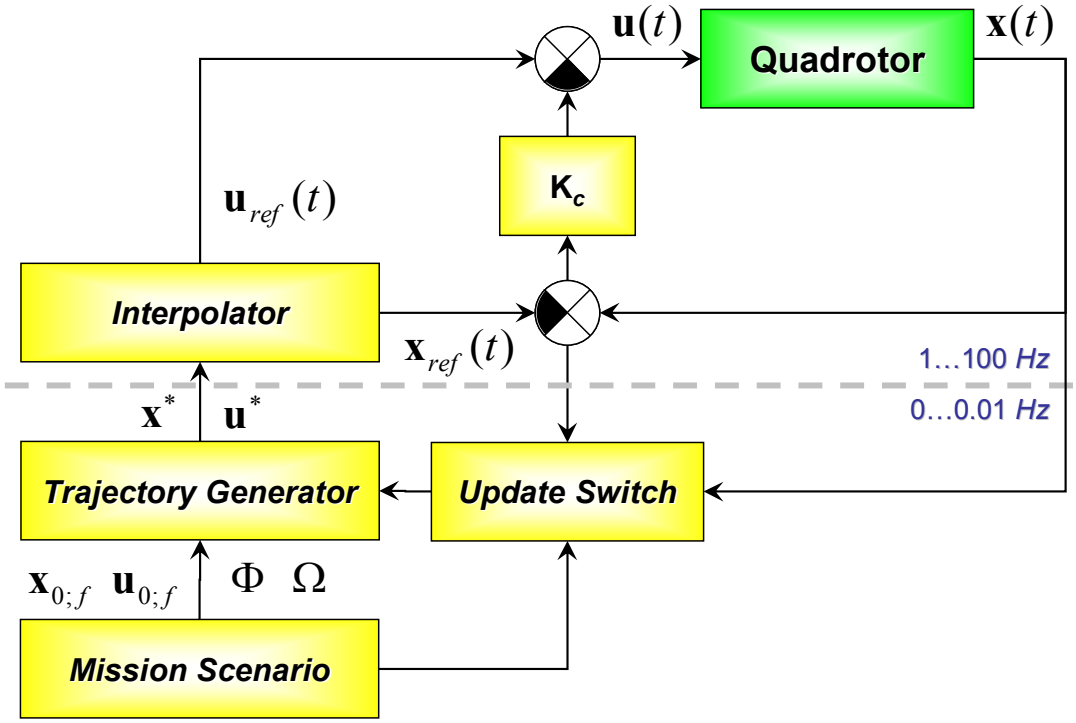


Figure 4.17: Controller framework

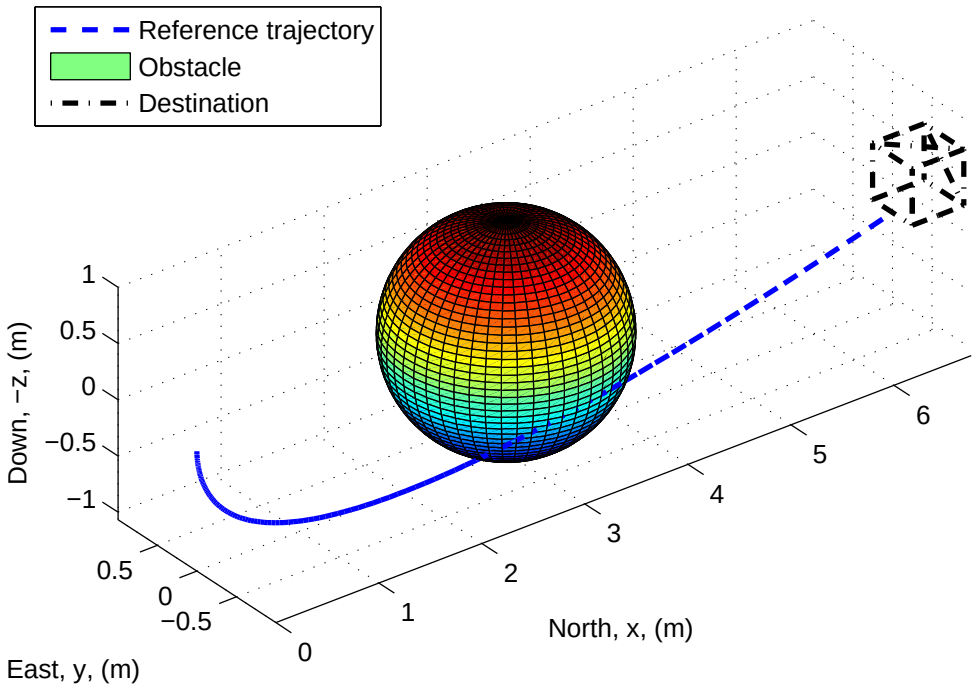


Figure 4.18: Obstacle mission initial trajectory

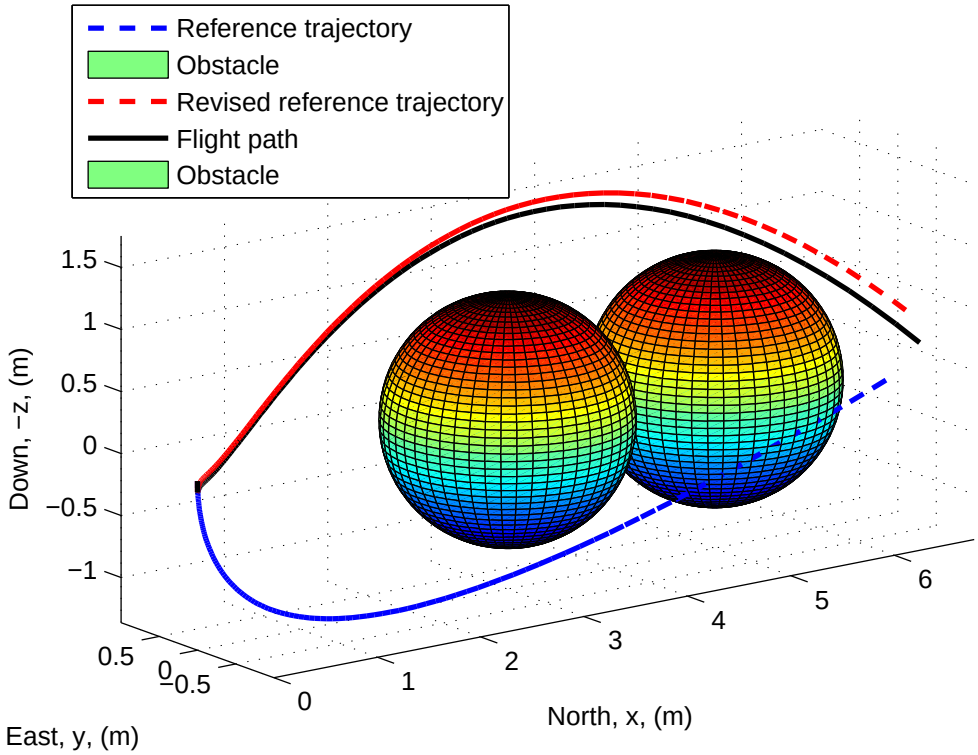


Figure 4.19: Revised reference trajectory for obstacle mission

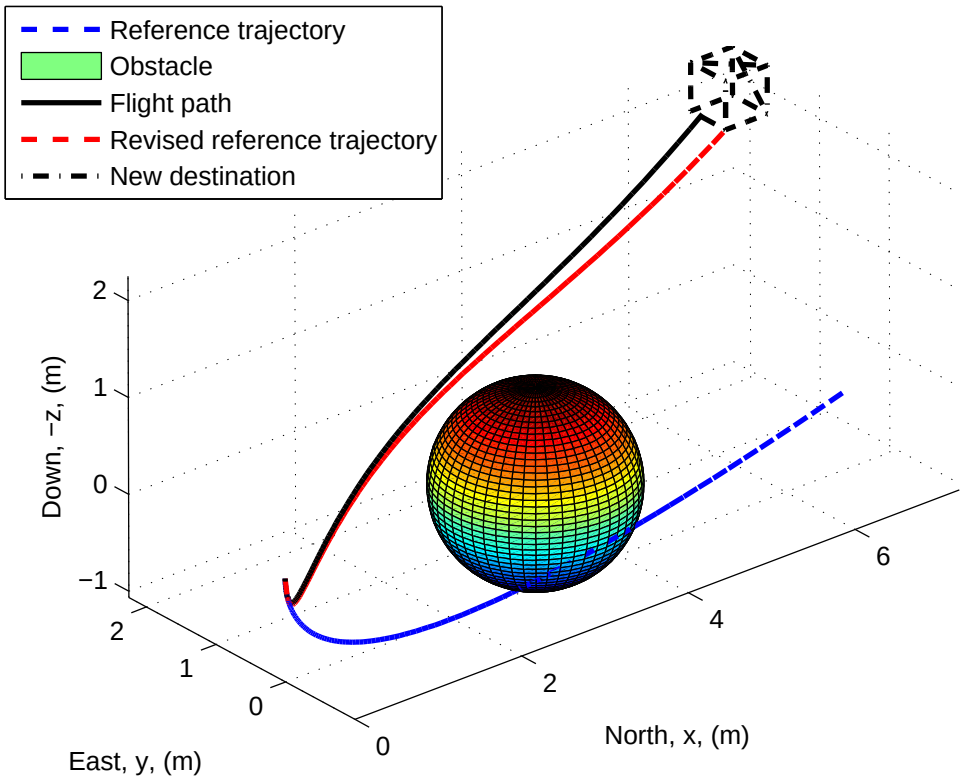


Figure 4.20: Moving target mission, revised trajectory

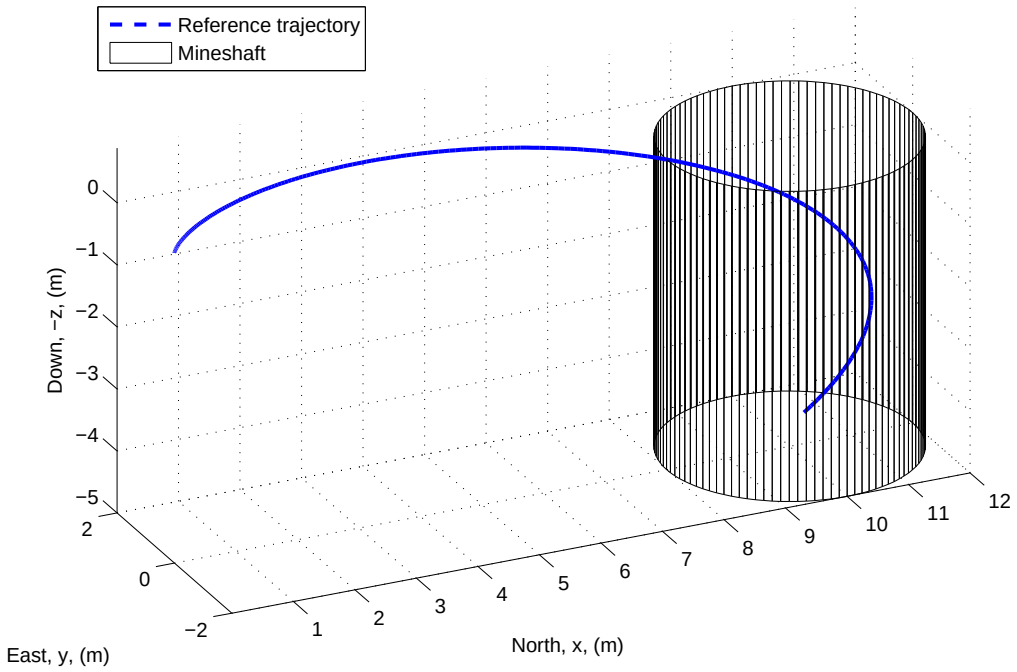


Figure 4.21: Original reference trajectory

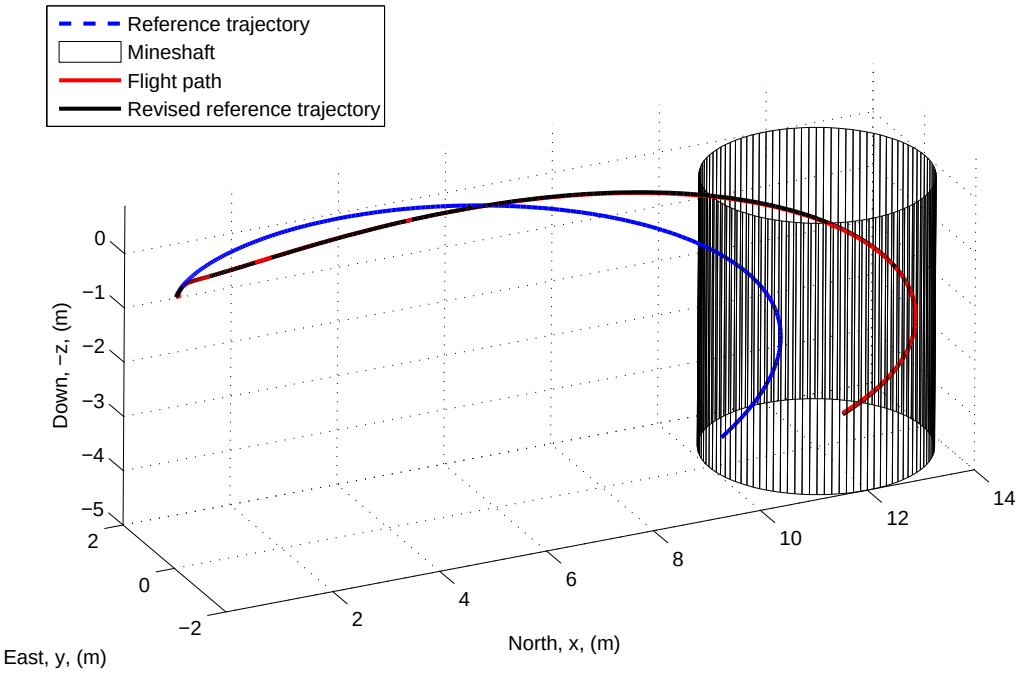


Figure 4.22: Determination of new trajectory

Chapter 5

Taranenko's Direct Method

5.1 Introduction

In Chapter 3, trajectory optimization was considered within the output space using the differential flatness property of the system's dynamics and parameterizing using Laguerre polynomials. The idea of output space optimization has been around for a long time, for example Taranenko and Momdzhi (1968) solved optimal control problems in the output space through dynamic inversion. This idea was very popular in the 1990's with the proposal of differentially flat systems (Fleiss *et al.* 1992, Martin *et al.* 1994, Koo and Sastry 1999, Driessen and Robin 2004, Milam *et al.* 2000). Despite this sudden interest in output space optimization Taranenko's techniques were largely ignored as the majority of literature was published in Russian.

The majority of this work has been focused on solving the optimization problem in the minimum amount of time, in order that a trajectory can be generated on-line. However, inevitably there is a trade off between optimality and computation time. The majority of research (Conway and Larson 1998), attempts to find the optimal solution in 'pseudo real time', which is generally too long to run in real time. Taranenko's method based on the optimal control work done by Taranenko and Momdzhi (1968) and developed by Yakimenko (2000) and Alekhin and Yakimenko (1999) is designed, to solve 'pseudo optimal' trajectory planning algorithms in real time. 'pseudo optimal' is an approximation of the optimal solution, which can be found in less time than it would take to find the optimal solution. The difference is clearly a trade-off between optimality and real time capability.

Now for practical implementation it is necessary for the algorithm to run in real time. 'Pseudo real time' is defined as for a mission lasting T seconds, anything up to $1000T$, whereas 'real time' is defined as $0.01T$. Clearly 'pseudo real tine' is not sufficient for this purpose. Therefore it is necessary to trade optimality for real time capability and solve the 'pseudo optimal' problem. As argued by Yakimenko (2000) the optimal path is not necessarily desirable in the first place, as there is normally a trade off between optimality and robust stability.

Therefore in reality we are much more concerned with first finding an feasible path and then some degree of optimality is preferred, but is not as important as providing a solution in the required time. Since the mid 1990's Yakimenko has been demonstrating real time optimization, using a standard IBM386 computer, for trajectory generation (Yakimenko 2000) as well a range of other applications including the inverted pendulum (Yakimenko 2006).

There are a number of advantages that this direct method has over other techniques such as those discussed in Section 1.5 and the technique presented in Chapter 3. These benefits arise from just two changes to the optimization routine, the first being the parameterization as a function of a virtual argument as opposed to time. The second change being the determination of boundary conditions analytically. The virtual argument within the parameterization allows for the velocity to be determined independently of position as opposed to the velocity being tied to the Cartesian coordinates through the time derivative. Determining the boundary conditions analytically reduces the number of constraints within the optimization routine. This ensures exact satisfaction of the boundary states so there is no longer any need for the boxes at the terminal states Equation 3.4.2. Furthermore, this analytical determination of the boundary conditions provides a smooth transition from one trajectory to the next as the current state and its derivatives are fixed as the initial conditions for a trajectory. This determination of the boundary conditions also reduces the number of free variables. This technique is capable of optimizing any cost function without the need to consider differentiability or dimensionality. This method will be looked at more detail in this chapter and demonstrated in simulation using the same controller as presented in Figure 4.17.

This chapter starts by introducing Taranenko's direct method and crucially the virtual argument. This introduction will include the key elements of this method; the decoupling of velocity and position and boundary condition satisfaction. In the same way as before, a cost function is required, this is discussed in more detail in Section 5.4. Finally the topology is considered for the cost function and state constraints, these are then compared with the previous scheme. The missions developed previously are then used to test the resulting control algorithm.

5.2 Separating Trajectory from Speed Profile

Parameterized trajectory optimization typically uses a basis function which is some function of time. This results in a preset single speed profile which is tied to the trajectory and, therefore, as the trajectory is determined the speed profile is also determined by

$$V = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \tag{5.1}$$

By transferring the parameterization into a virtual domain, the path and the speed profile can be optimized independently, if the speed profile needs changing it can be done easily while maintaining the same path.

To do this, first we replace time with a virtual argument (τ), which is the virtual arc length,

this is related to time through a variable speed factor λ where

$$\lambda = \frac{d\tau}{dt} \tag{5.2}$$

The output space parameterization is now expressed as a function of this virtual argument as opposed to time

$$P(\tau) = a_0 + a_1\tau + a_2\tau^2 ... a_M\tau^M \tag{5.3}$$

where $P(\tau) = [x(\tau), y(\tau), z(\tau)]^T$. The derivatives are now taken with respect to this virtual argument

$$x' = \frac{dx}{d\tau} \tag{5.4}$$

$$y' = \frac{dx}{d\tau} \tag{5.5}$$

$$z' = \frac{dx}{d\tau} \tag{5.6}$$

The velocity can now be expressed as a function of this virtual argument and the speed factor λ

$$V(\tau) = \lambda\sqrt{x'^2\tau + y'^2\tau + z'^2\tau} \tag{5.7}$$

and the acceleration likewise

$$\dot{V} = \frac{\lambda}{\sqrt{x'^2 + y'^2 + z'^2}}\left(\lambda'\left(x'^2 + y'^2 + z'^2\right) + \lambda(x'x'' + y'y''' + z'z'')\right) \tag{5.8}$$

However, to optimize the trajectory and the velocity independently, the velocity can also be parameterized as a function of this virtual argument. In this case the order of the polynomials is less than the polynomial in 5.3, this will be discussed further in Section 5.3.

$$V = a_0 + a_1\tau + a_2\tau^2 + \ldots + a_{M-2}\tau^{M-2} \tag{5.9}$$

Now, we obviously need to relate these output vectors as functions of time in order to evaluate constraints, cost functions and to provide time dependent reference trajectories. By evaluating the position and velocity polynomials at a number of equidistant nodes in τ

$$P(\tau)|_{j=0\ldots N} \tag{5.10}$$

$$V(\tau)|_{j=0\ldots N} \tag{5.11}$$

where N is the number of nodes. The distance travelled (s) between nodes can be easily approximated for $j = 1\ldots N$

$$s_j = \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2 + (z_j - z_{j-1})^2} \tag{5.12}$$

The time elapsed between nodes (Δt_j) for $j = 1\ldots N$ can be approximated using the obvious relationship

$$\Delta t_j = 2\frac{s_j}{V(\tau)_j + V(\tau)_{j-1}} \tag{5.13}$$

Therefore, the time at every node (j) for $j = 2 \ldots N$ is calculated by

$$t_j = t_{j-1} + \Delta t_j \tag{5.14}$$

The speed factor for $j = 2 \ldots (N-1)$ can be approximated at every node assuming small time steps

$$\lambda_j = \frac{\Delta\tau}{\Delta t_{j-1}} \tag{5.15}$$

Determining the boundary conditions for λ when $j = 0$ and $j = N$ will be discussed in Section 5.3. The speed factor derivatives $(\lambda' \quad \lambda'')$ are calculated numerically at each time step using the backward step method.

Now the time derivatives can be calculated for each of the output states, from the virtual derivatives $[x \quad x' \quad x'' \quad x''']$ and the speed factor approximation $[\lambda \quad \lambda' \quad \lambda'']$

$$\dot{x} = \lambda x' \tag{5.16}$$

$$\ddot{x} = \lambda^2 x'' + \lambda\lambda' x' \tag{5.17}$$

$$\dddot{x} = \lambda^3 x''' + 3\lambda^2\lambda' x'' + (\lambda^2\lambda'' + \lambda\lambda'^2)x' \tag{5.18}$$

As before, with the output vector and its derivatives, the full state and control vectors can be evaluated using the differentially flat equations.

5.3 Parameterization

In Section 3.6 there was a comparison of various parameterization techniques. Each scheme was defined as a function of a free variable and a particular basis function such as a Laguerre polynomial. Within this approach the free variable was varied in order to meet initial, terminal and actuator constraints as well as minimizing some cost function. The next section will demonstrate how the problem can be reposed by satisfying the boundary conditions analytically. In this case any basis function can be chosen without affecting algorithm robustness. The minimum order of the parameterization (M) in this case is defined by the number of boundary conditions which need to be satisfied.

Satisfying Boundary Conditions

So if the order of the parameterization is now determined by the number of constraints on the boundary conditions, how is the order determined and how are these constraints met? Taking for example a simple polynomial

$$P(t) = a_0 + a_1\tau + a_2\tau^2 + \ldots + a_M\tau^M \tag{5.19}$$

If the initial and terminal position, velocity and acceleration are to be constrained then M must be at least 5 as that will give 6 free variables $(a_0 \ldots a_5)$ and 6 conditions to be met $(x_0, \dot{x}_0, \ddot{x}_0, x_f, \dot{x}_f, \ddot{x}_f)$. The analytical solution will now be shown for this case where the order of the polynomial is the minimum order to achieve these boundary conditions analytically.

5th Order Polynomials

The fifth order polynomial is used to parameterize the position of the vehicle over time, in this example only the x coordinate is demonstrated but this can obviously be extended for the 4 dimensional optimization problem (x, y, z, ψ). For the second derivative of the polynomial where the order is now 3rd order, x'' is expressed as a function of the free variable (b)

$$x'' = b_2 + b_3\tau + b_4\tau^2 + b_5\tau^3 = \sum_{k=2}^{5} b_k\tau^{k-2} \tag{5.20}$$

Integrating to get the first derivative of x and again to get x produces

$$x' = \sum_{k=1}^{5} \frac{b_k\tau^{k-1}}{\max(1, k-1)} \tag{5.21}$$

$$x = \sum_{k=0}^{5} \frac{b_k\tau^{k}}{\max(1, k(k-1))} \tag{5.22}$$

Now as the six boundary conditions are met analytically the free variables b_k can be determined from the following relationship:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 \\ 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau_f^4 \\ 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_0' \\ x_0'' \\ x_f \\ x_f' \\ x_f'' \end{bmatrix} \tag{5.23}$$

In this case, where the six boundary conditions are determined analytically, the only free variable is arc length τ_f. As will be shown shortly, varying τ_f gives only a limited degree of flexibility of the reference trajectory and, therefore, further flexibility is required. There are two approaches to this problem, firstly, the constraints on the second derivative can be relaxed, this would, therefore, become a free variable. The second approach is to increase the order of the parameterization, this would allow for the determination of the third derivative or jerk boundary conditions $(\dddot{x}_0, \dddot{x}_f)$. As we do not wish to constrain the third derivatives boundary conditions then these become additional free variables.

Boundary conditions in the virtual space

In order to determine the boundary conditions, they must be posed as a function of the virtual argument. Transforming for the time domain to the virtual domain requires a value of λ. By rearranging (5.18)

$$x' = \dot{x}\lambda^{-1} \tag{5.24}$$

$$x'' = \ddot{x}\lambda^{-2} - \dot{x}\lambda'\lambda^{-1} \tag{5.25}$$

now with the initial and final values of λ, these can be calculated. For a fixed wing vehicle the following boundary values of λ are suggested (Yakimenko 2000):

$$\lambda_0 = V_0 \tag{5.26}$$

$$\lambda_f = V_f \tag{5.27}$$

$$\lambda_0' = \dot{V}_0 V_0^{-1} \tag{5.28}$$

$$\lambda_f' = \dot{V}_f V_f^{-1} \tag{5.29}$$

However choosing the boundary values for hovering vehicle needs to be considered carefully. A rotor craft can hover, resulting in singularities at $V_0 = 0$. In practice the velocity of a vehicle is rarely zero in flight. The velocity of a hovering vehicle may be close to zero but very rarely zero. In this work, it is assumed that $V = 0$ only at t_0 and t_f. Furthermore it is assumed at these time the acceleration is also equal to zero. In this case the boundary conditions are simply fixed to zero in both the virtual and time domain and the boundary conditions for λ are not required. Between these points the value is calculated without the singularities as the velocity and accelerations are non-zero.

$$x_0' = \dot{x}_0 = x_f' = \dot{x}_f = 0 \tag{5.30}$$

$$x_0'' = \ddot{x}_0 = x_f'' = \ddot{x}_f = 0 \tag{5.31}$$

7th Order Polynomials

Now for additional flexibility the order of the polynomial can be increased to a 7th order polynomial. In the same way as for the 5th order parameterization, the analytical solution is found for the boundary conditions of the 7th order parameterization where the coefficients (c) are now determined by:

$$c_0 = x_0 \tag{5.32}$$

$$c_1 = x_0' \tag{5.33}$$

$$c_2 = x_0'' \tag{5.34}$$

$$c_3 = x_0''' \tag{5.35}$$

$$c_4 = -\frac{2x_f''' + 8x_0'''}{\tau_f} + \frac{30x_f'' - 60x_0''}{\tau_f^2} - \frac{180x_f' + 240x_0'}{\tau_f^3} + 420\frac{x_f - x_0}{\tau_f^4} \tag{5.36}$$

$$c_5 = \frac{10x_f''' + 20x_0'''}{\tau_f^2} - \frac{140x_f'' - 200x_0''}{\tau_f^3} + \frac{780x_f' + 900x_0'}{\tau_f^4} - 1680\frac{x_f - x_0}{\tau_f^5} \tag{5.37}$$

$$c_6 = -\frac{15x_f''' + 20x_0'''}{\tau_f^3} + \frac{195x_f'' - 225x_0''}{\tau_f^4} - \frac{1020x_f' + 1080x_0'}{\tau_f^5} + 2100\frac{x_f - x_0}{\tau_f^6} \tag{5.38}$$

$$c_7 = 7\frac{x_f''' + x_0'''}{\tau_f^4} - 84\frac{x_f'' + x_0''}{\tau_f^5} + 420\frac{x_f' + x_0'}{\tau_f^6} - 840\frac{x_f + x_0}{\tau_f^7} \tag{5.39}$$

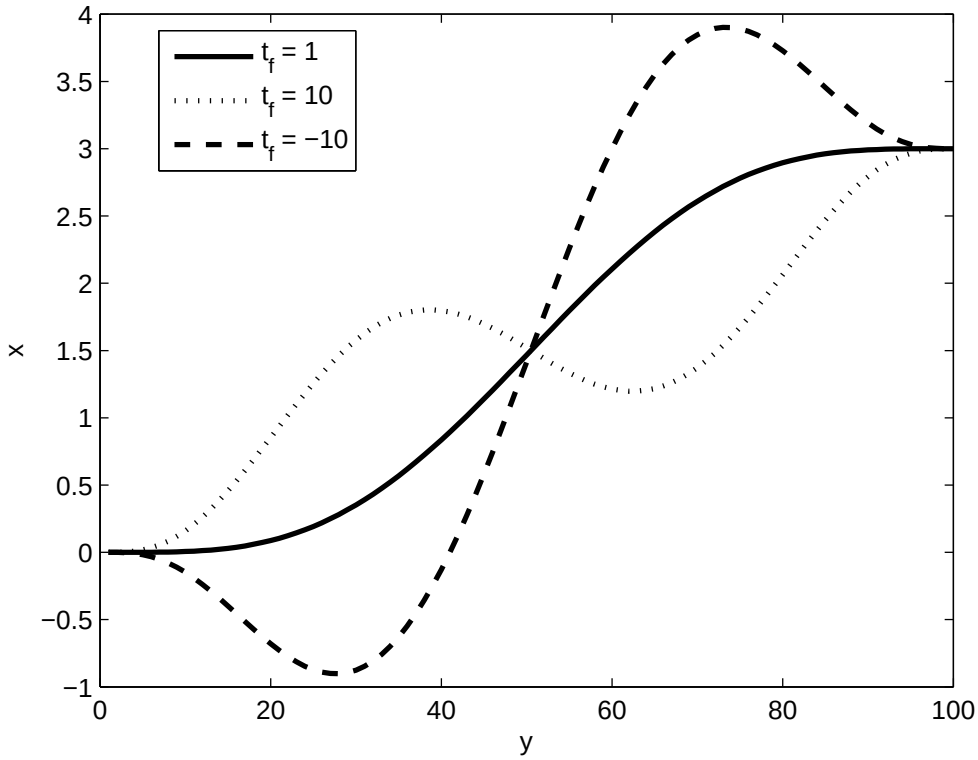


Figure 5.1: Varying t_f

Again the positions, velocities and accelerations at the initial and terminal state are constrained but now the third derivative (jerk) is introduced and this becomes a free variable. Of course, if the third derivative needs to be constrained then this can also be done analytically and the polynomial order could be increased to 9th order or higher for additional flexibility. Alternatively, further free parameters can be included by increasing the polynomial order. Inevitably, this is at the expense of an increased search space dimension and longer convergence times for the non-linear program. The additional flexibility can be demonstrated in the following example, initially only τ_f is varied as shown in Figure 5.1, where

$$x_0 = 0, \tag{5.40}$$
$$\dot{x}_f = 0, \tag{5.41}$$
$$\ddot{x}_0 = 0, \tag{5.42}$$
$$\dddot{x}_0 = 2, \tag{5.43}$$
$$x_f = 0, \tag{5.44}$$
$$\dot{x}_f = 0, \tag{5.45}$$
$$\ddot{x}_f = 0 \tag{5.46}$$
$$\dddot{x}_f = 2 \tag{5.47}$$

Obviously in practice however, τ_f would remain positive. Now, by taking the same initial and terminal conditions but varying the third derivatives $(\dddot{x}_0, \dddot{x}_f)$ as well as the virtual

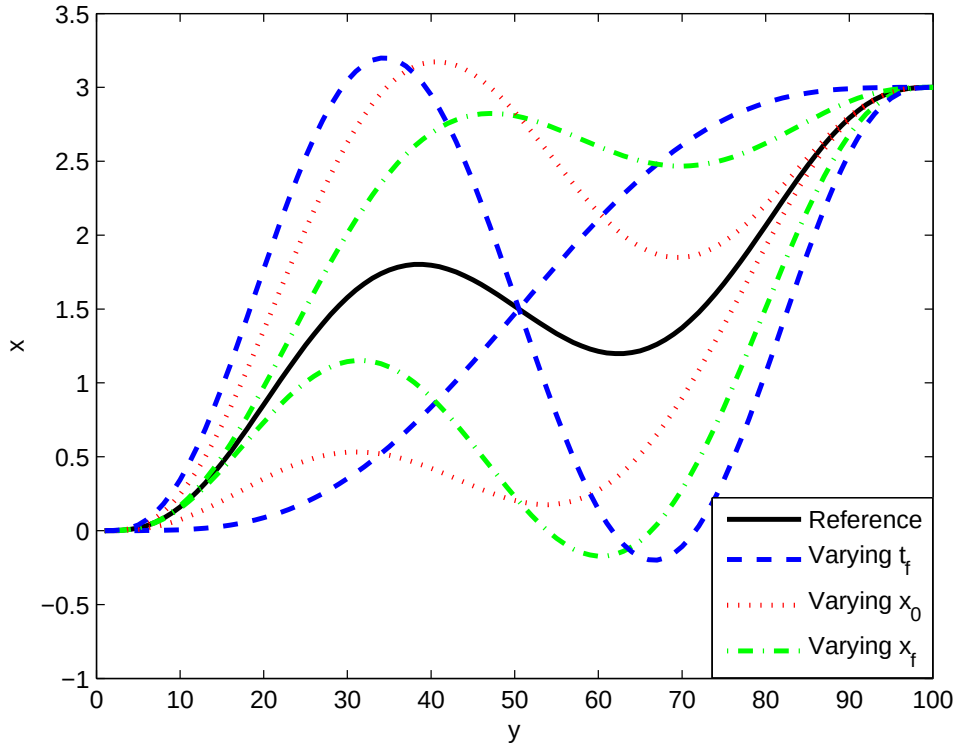


Figure 5.2: Varying t_f and third derivative

argument (τ_f) results in a greater degree of flexibility as seen in Figure 5.2.

Velocity polynomial

The order of the velocity polynomial is dependent on the boundary conditions that need to be met. If the velocities and accelerations are constrained at the boundary conditions then the polynomial is fifth order

$$V = \sum_{k=0}^{5} \frac{a_k \tau^k}{\max(1, k(k-1))} \tag{5.48}$$

5.4 Cost function

The cost function, Φ , is a quantitative measure of the optimality of the trajectory and can be approximated by the sum of the running costs and the terminal cost. Assuming that the running costs (fuel consumption) are proportional to the average velocity, the objective function can be defined as:

$$\Phi = (1-\mathbf{w})\frac{1}{t_f}\int_0^{t_f} \sqrt{(P_1\dot{x}^2 + P_2\dot{y}^2 + P_3\dot{z}^2)}\, dt + \mathbf{w}(t_f - T)^2 \tag{5.49}$$

where $\mathbf{w}$, P_1,P_2,P_3 are weighting factors, and T is the desired time of arrival. In particular, the case when $\mathbf{w} = 1$ and $T = 0$ corresponds to the minimum-time problem; and the case when $\mathbf{w} = 0$, $P_1 = P_2 = P_3$ approximates the minimum-fuel problem which is discussed in Section 5.4.1.

5.4.1 Alternative cost functions

The assumption that the average velocity is a reasonable approximation of fuel cost holds for certain scenarios but is not always the case. One advantage of using a trajectory optimization algorithm such as Taranenko's direct method is the ability to pose the problem with different cost functions or constraints. It is, therefore, worth considering the minimum fuel problem where the objective function is now in the control space. Starting with the new objective function which is now formulated as the minimum fuel problem by simply taking the scalar product of the control input:

$$\Phi_u = \frac{1}{t_f} u_1 P_1 u_1' \tag{5.50}$$

where $P_1 = 1$

The reference trajectory can be determined for mission (ii) as before, with this new objective function and compared with the original reference trajectory for the minimum distance problem. Figure 5.3 shows the minimum distance and the minimum fuel trajectories as well as the corresponding control profiles. It can be seen that although the trajectories and control profiles are similar there is a difference between the two problems. It can also be seen that the path length for the minimum fuel problem is longer than that of the minimum distance but the fuel costs are harder to visualise. Table 5.1 shows both the minimum fuel (Φ_u) and the minimum distance (Φ) cost functions for each trajectory.

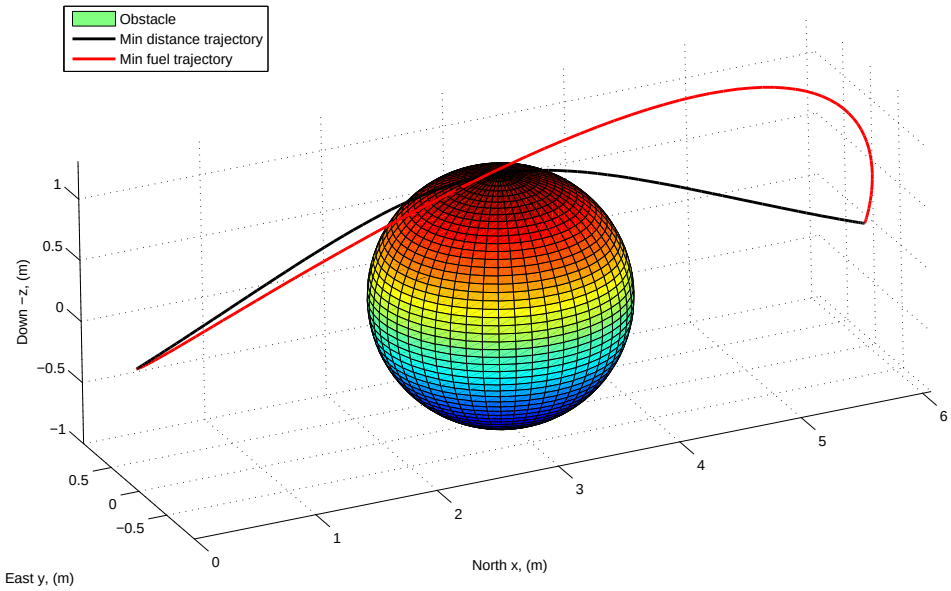
Table 5.1: Minimum fuel and distance comparison

Objective function	Fuel cost(Φ_u)	Distance cost (Φ)
Min Fuel	33.3732	2.8791
Min Distance	36.848	0.2797

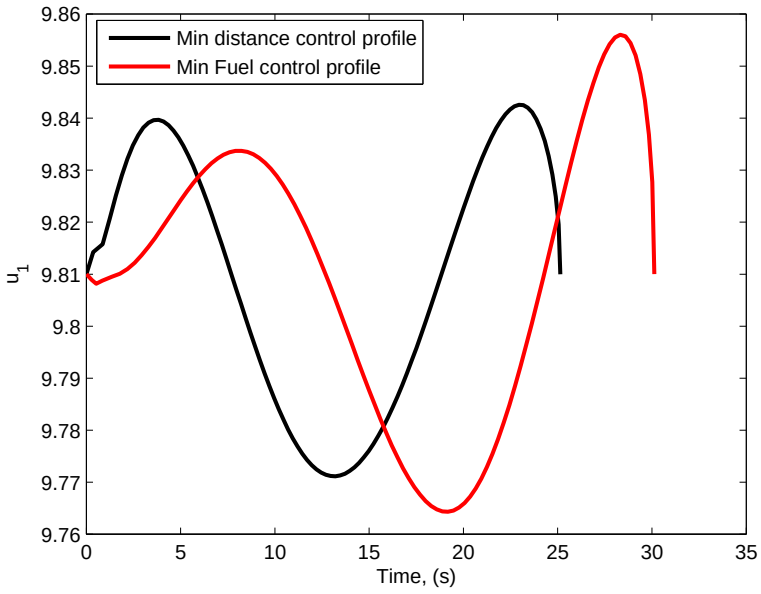
5.4.2 Convexity

Non-linear trajectory optimization is generally computationally demanding, yet Taranenko's direct method has been demonstrated in real time (Yakimenko 2000). The major reason for this is the decoupling of the time and path problems as well as the analytical solving of the boundary conditions, both of which enable the problem to be solved with as few as 9 free variables. It would logically follow therefore, that by reducing the number of free variables the problem reduces in complexity, this is not necessarily the case.

In fact, it is well known (Ross and Fahroo 2006) that the computation time of a trajectory optimization is not simply dependent on the number of variables but also on factors such



(a) Reference flight path



(b) Control profile

Figure 5.3: Alternative cost functions

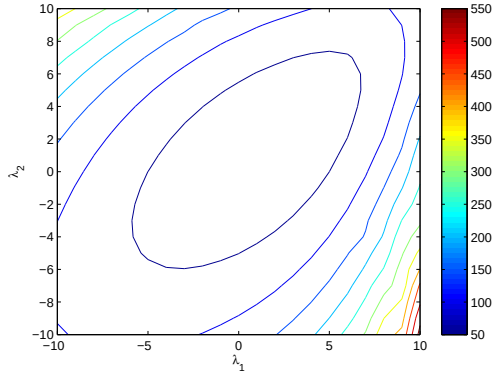
as convexity, sparsity and matrix vector products. For example SOCS, is a high performance optimization package which is used for trajectory optimization for amongst other things a high performance aircraft (Boeing 2008). In this package as many as 807 grid points are required with 5795 free variables, as opposed to the 9 presented here. In this example where there are so many free variables, the issue of convexity is of massive importance. A large number of free variables also introduces other important issues which are of less importance when considering a problem with 9 free variables such as: matrix sparsity, capability to use matrix rather than scalar products and parameterization orthogonality. As discussed in (Betts 1998), the number of free variables can be increased to reduce computation time if this improves on some of the other factors which influence computation time. In this work however, where the number of free variables is so low, the dependency on other factors is reduced.

Another example to demonstrate the influence of other factors is found in (Ross and Fahroo 2006) where there is a comparison of several different techniques, including differential flatness. They show that the differential flatness approach under-performs in comparison with other techniques, such as differential inclusion, as the convexity of the problem was lost when transferring the problem from the control to the output space. Again, this approach uses a greater number of free variables than Taranenko's direct method but the point remains, that other factors influence the computation time, not purely the number of free variables. However, by having as few as 9 free variables the computation time can be significantly reduced. This can be backed up by the work carried out primarily by Yakimenko (2000) in which the cost function can be any non-convex function and yet has been demonstrated working in real time.

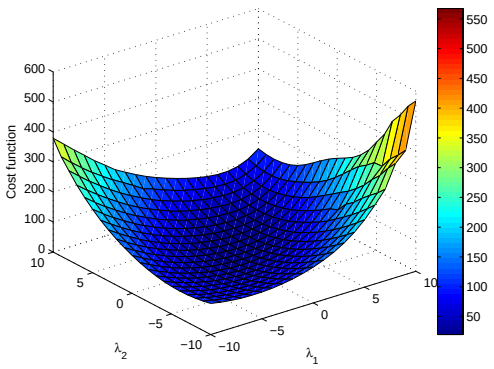
Ross and Fahroo (2006) also found, that the differential flatness approach worked well with a good initial guess of free variable values. Using a suitable parameterization technique, where the initial and terminal conditions are met, the initial guess is often a feasible solution and therefore, a good initial guess is inherent within the formulation.

5.4.3 Topology plots

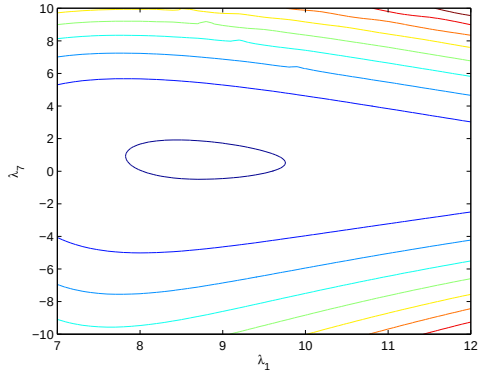
The cost function can be plotted as a function of the free variables $(\lambda_1, \lambda_2, \lambda_7)$, as shown in Figure 5.4. Of interest in these plots is the problem conditioning, a well conditioned problem will react equally to a change in different free variables. As seen the problem appears to be convex within this region and the conditioning is significantly better than in the previous example in Section 3.6.5. The speed and obstacle avoidance constraints can also be plotted as a function of the free variables although as singularities exist at $\tau_f = 0$, this is constrained such that $\lambda_7 > 0$. In Figure 5.5 the maximum velocity is plotted against the same free variables, the conditioning appears not to be as good as for the cost function. The final plot in Figure 5.6 is the obstacle avoidance constraint against the free variables.



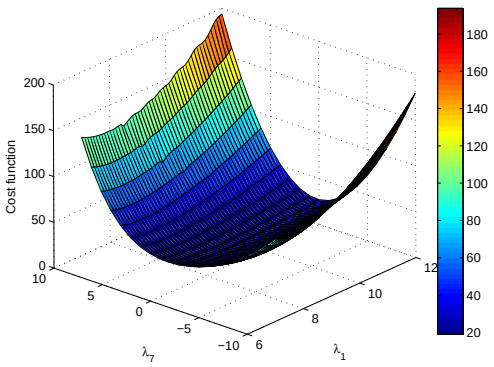
(a) λ_1, λ_2 contour plot



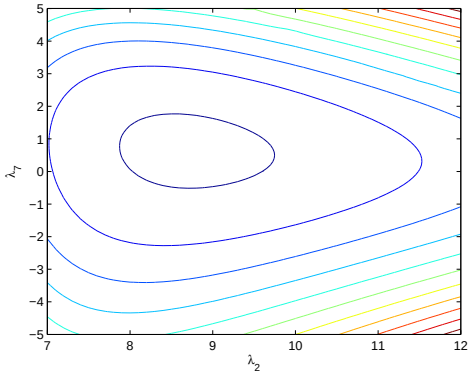
(b) λ_1, λ_2 surface plot



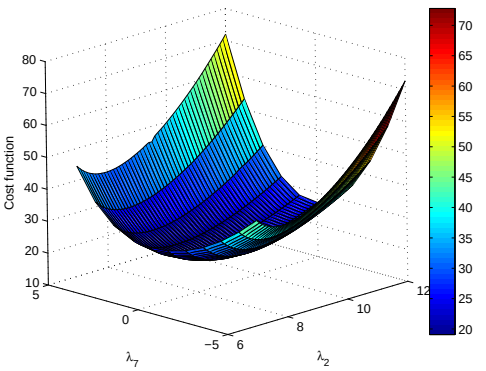
(c) λ_1, λ_7 contour plot



(d) λ_1, λ_7 surface plot

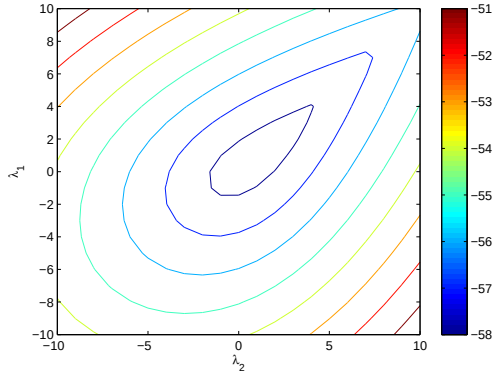


(e) λ_2, λ_7 contour plot

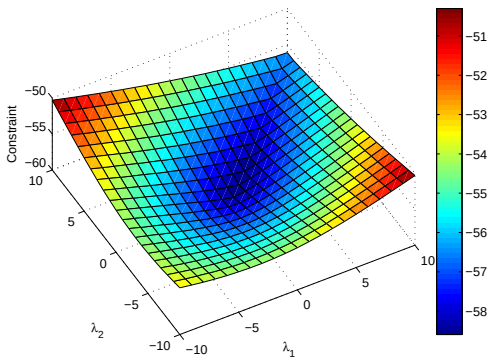


(f) λ_2, λ_7 surface plot

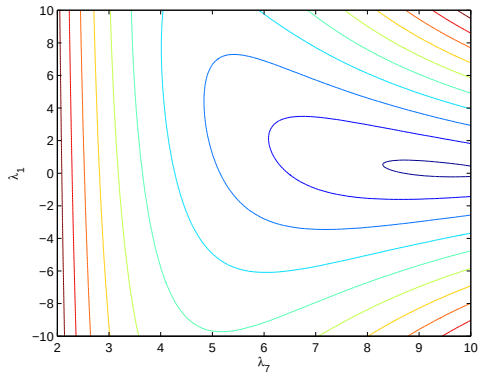
Figure 5.4: Cost function convexity as function of $\lambda_1, \lambda_2, \lambda_7$



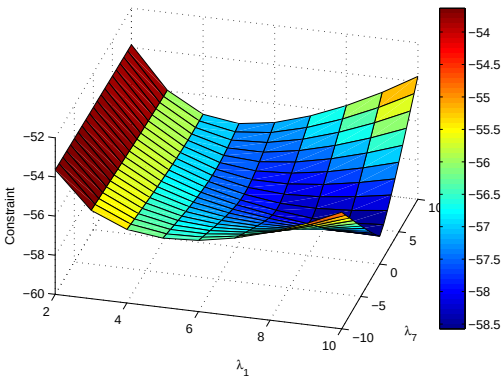
(a) λ_1, λ_2 contour plot



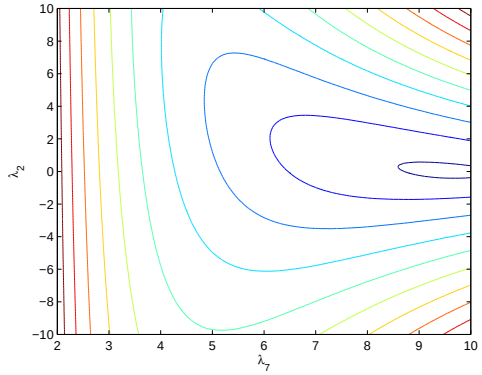
(b) λ_1, λ_2 surface plot



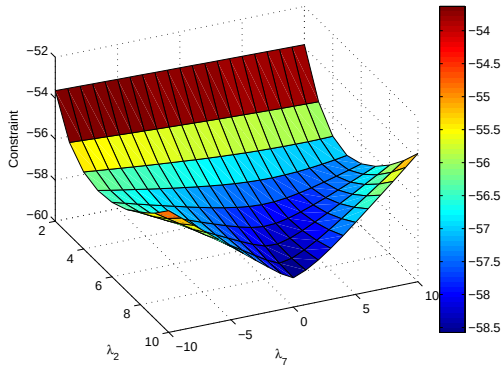
(c) λ_1, λ_7 contour plot



(d) λ_1, λ_7 surface plot

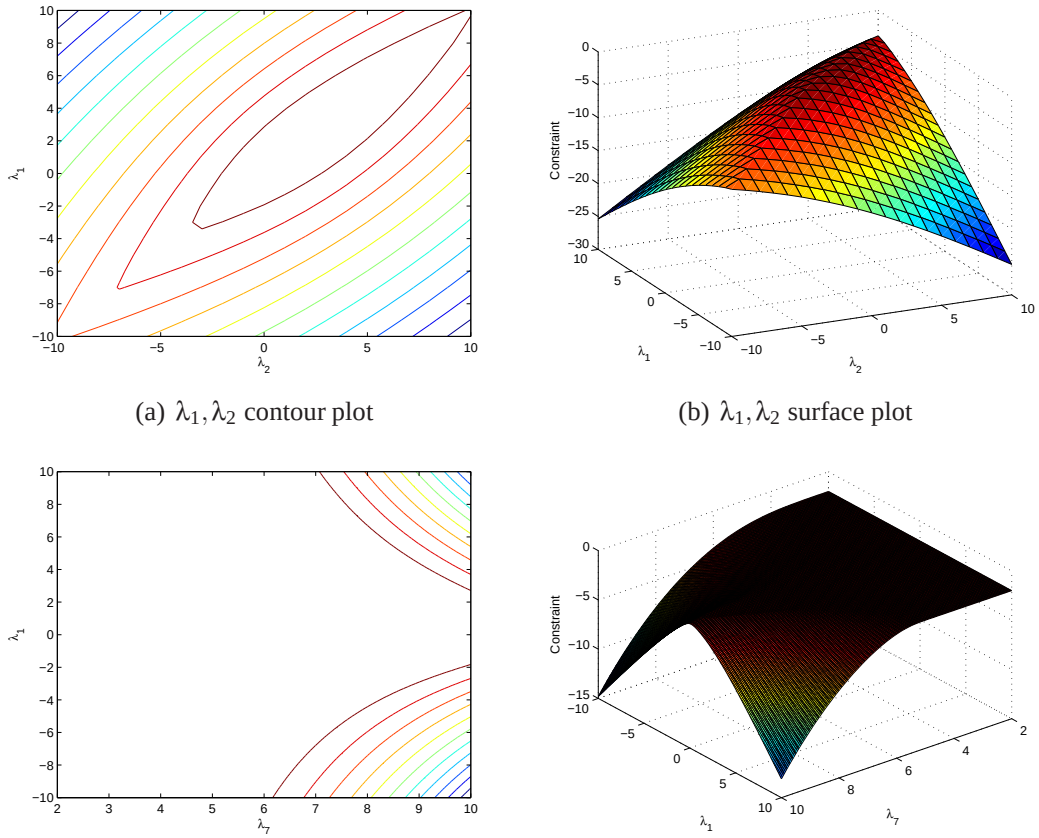


(e) λ_2, λ_7 contour plot



(f) λ_2, λ_7 surface plot

Figure 5.5: Maximum speed constraint convexity as function of $\lambda_1, \lambda_2, \lambda_7$



(a) λ_1, λ_2 contour plot

(b) λ_1, λ_2 surface plot

(c) λ_1, λ_7 contour plot

(d) λ_1, λ_7 surface plot

Figure 5.6: Minimum distance from obstacle constraint convexity as function of $\lambda_1, \lambda_2, \lambda_7$

5.5 Results

Mission (i (c))

The first mission is a simple vertical flight of 7m finishing at hover at the destination. The vertical flight in itself is not a challenging mission as the analytical solution for minimum time and minimum fuel is easily achievable from Pontryagin's principle (Pontrjagin *et al.* 1962). However, when the flight time is pre-determined or a combination of minimum fuel and minimum time is required the problem requires a more complex approach. Recalling the cost function (Equation 5.49) it can be seen that by varying the weighting (w) it is possible to optimize any combination of minimum fuel and optimal time, therefore, we can decide we want optimal time but not at any cost. Optimal time can refer to minimum time or in this case predetermined flight time. For surveillance missions the camera it may be required that the camera arrives at a specific time.

In the case of minimum time it is easy to compare with the analytical optimal solution. The optimal solution is well known (Tou 1964) to be the bang bang solution given by

$$u_1(t) = 5.4 \quad \text{for} \quad 0 \leq t \leq 1.44 \tag{5.51}$$

$$u_1(t) = 0 \quad \text{for} \quad 1.44 \leq t \leq 1.8 \tag{5.52}$$

This is an optimal solution and while it is not expected that the solution will be exactly this it gives an indication of the approximation capabilities of the optimization routine which is dependent on the tolerances set within the optimization, choice of basis function and convergence properties of the algorithm. Figure 5.7 shows the altitude against time for the flight which has a total flight time of 3.26 seconds. Figure 5.8 shows the first control input over this time. As seen the thrust is a polynomial approximation of the maximum constraint followed by a short reduction in thrust. It should be noted though that while the flight time t_f is 1.5 times greater than the optimal one (obtained for the simplified dynamics) all boundary conditions (including those imposed on the rotor fans dynamics) are satisfied, the reference trajectory and nominal control are feasible and (which is probably the most important) the solution can be obtained in the real time.

Now, consider the optimal flight time problem of $T = 5$, $\mathbf{w} = 1$ and $T = 10$, $\mathbf{w} = 1$. In this case the resulting optimal flight time is exactly the desired flight time and the reference and control inputs can be seen in Figure 5.9. To demonstrate constraint satisfaction the reference and actual control inputs were plotted against the constraint. As seen the flight time in this case is equal to the required mission time T and all constraints are met. The sharp peak at 0.1s of the actual control inputs are the result of a strong down wind, this causes the vehicle to initially accelerate downwards and therefore an increase in the thrust is required to remove the positional error. As the wind is constant this explains the constant extra thrust applied for the duration of the flight. Once the initial transient has overcome the downwind, the trajectory is tracked closely despite the constant wind, as

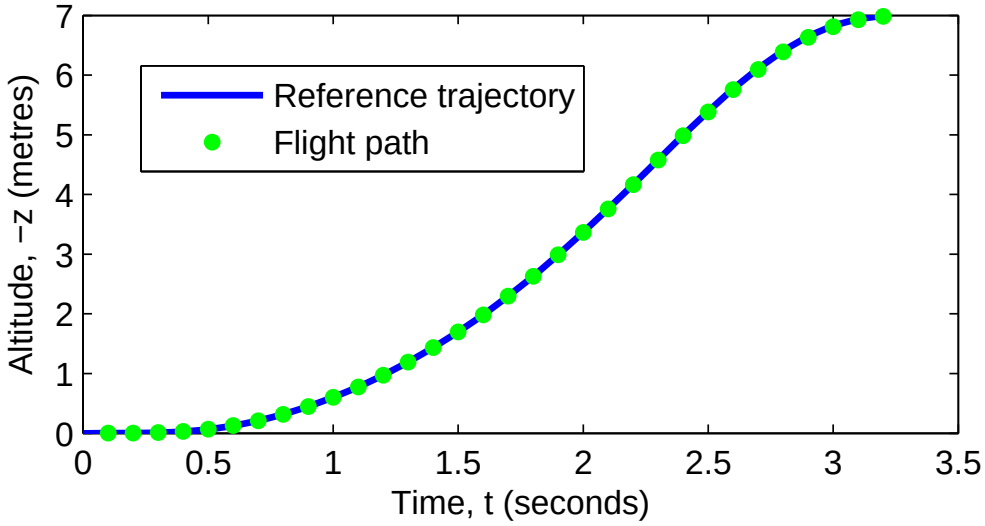


Figure 5.7: Minimum time for vertical flight

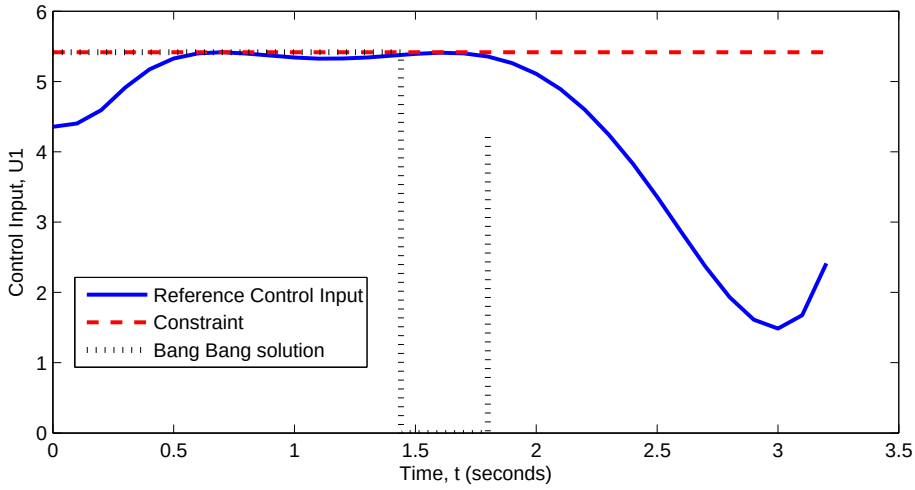


Figure 5.8: Minimum time for vertical flight speed profile

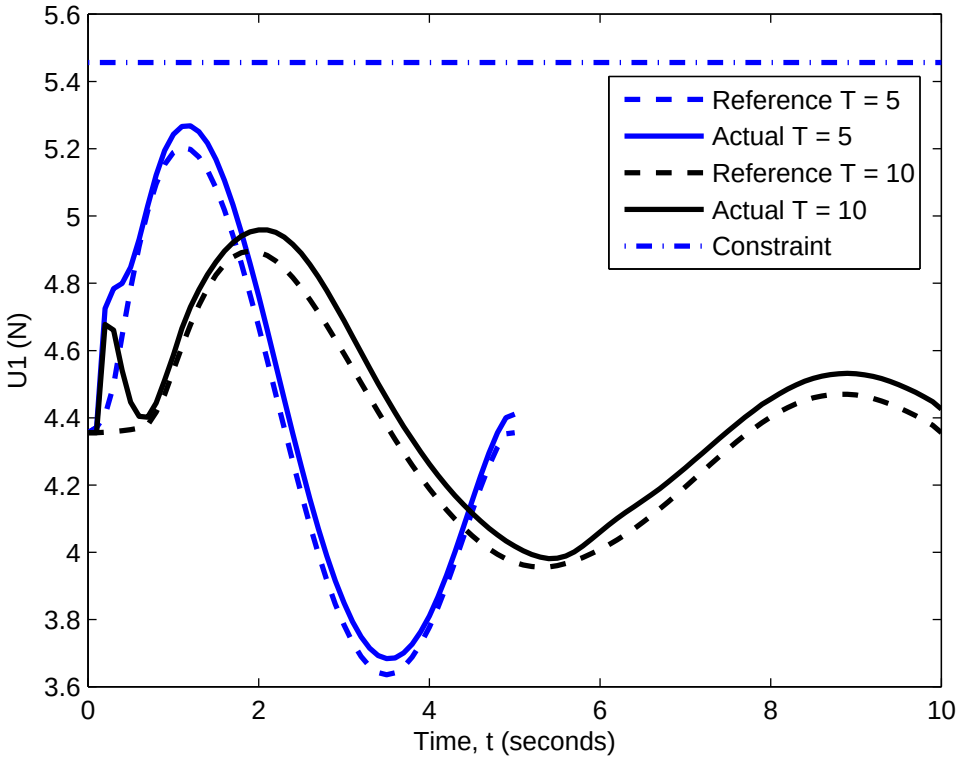


Figure 5.9: Vertical flight, $\mathbf{w} = 1$

can be see in the speed profiles in Figure 5.10 and Figure 5.11.

Now assuming optimal time is still required but not at any cost. The weighting, $\mathbf{w}$, is reduced to 0.75 in order to have some measure of minimum fuel and the resulting control profile can be seen in Figure 5.12. For the first mission where $T = 5$s, the actual optimal flight time is higher at 6.2 seconds, this is due to the fuel consideration and this can be seen by the flatter control profile. The second mission flight time is 10.2 seconds which is close to the desired flight time but again the control profile is flatter than for the previous case. Again a sharp peak at 0.1s is due to a constant downwind.

This can further be demonstrated by showing the control profiles, for decreasing values of the weighting, with the desired time of 10 seconds as seen in Figure 5.13. As the weighting decreases the flight time increases resulting in a flatter control profile. These are in effect approximations of the well-known analytical solutions. At a weighting of one the solution is an approximation of the bang-bang minimum time problem and as the weighting approaches zero the solution approaches the infinite time solution. So decreasing the weighting flattens the control profile as the result tends to the infinite time analytical solution, this can be seen in Figure 5.14 which plots the maximum thrust against the weighting. As the weighting is reduced then the flight time increases, again towards the infinite time analytical solution, this is shown in Figure 5.15. Data from Figure 5.14 and Figure 5.15 is combined on a single plot in Figure 5.16. This represents the Pareto frontier or the set of solutions that are all Pareto efficient (no further improvements can be made).

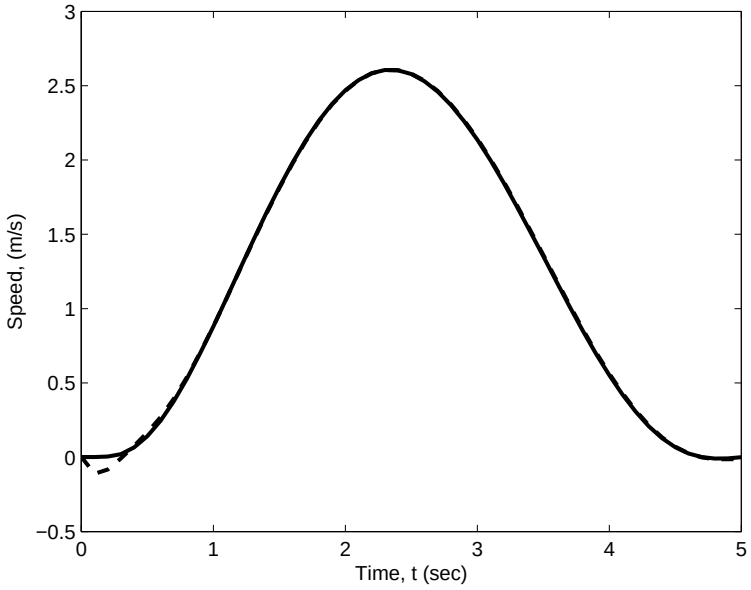


Figure 5.10: Speed profile, T = 5

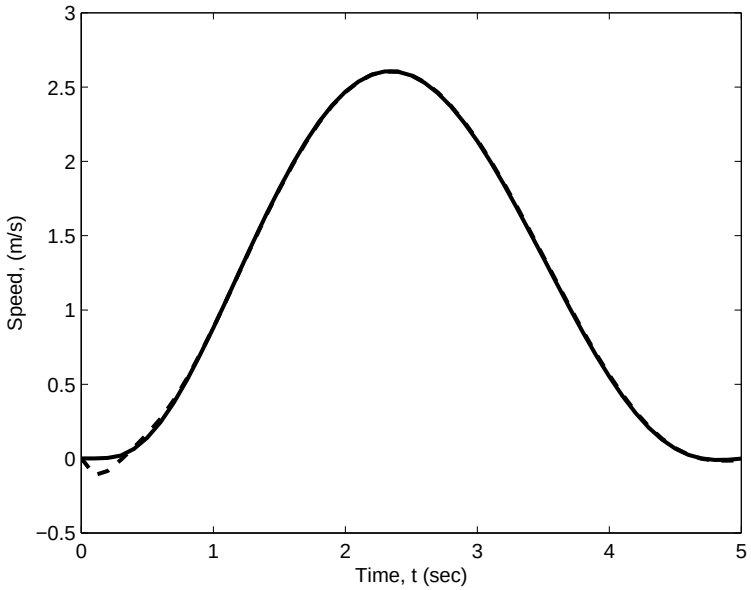


Figure 5.11: Speed profile, T = 10

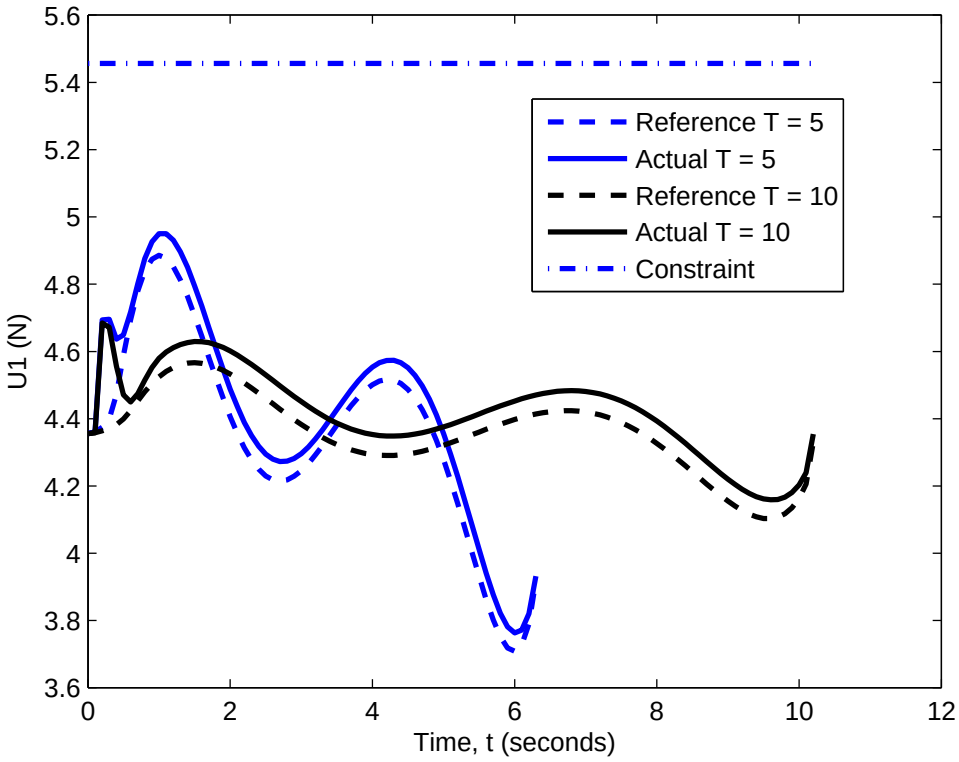


Figure 5.12: Vertical flight, **w** = 0.75

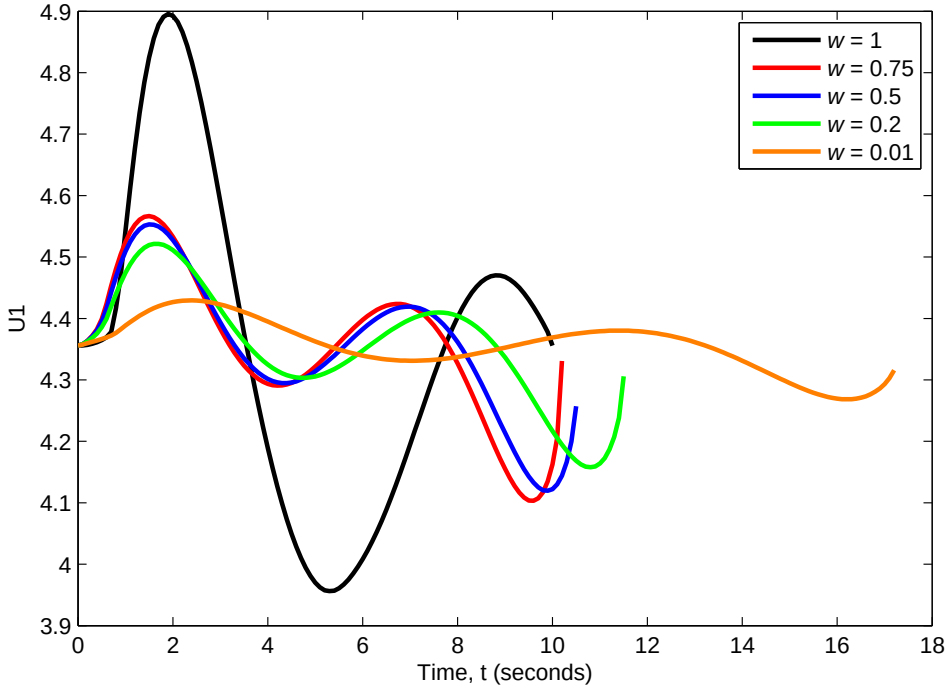


Figure 5.13: Vertical flight with varying **w**

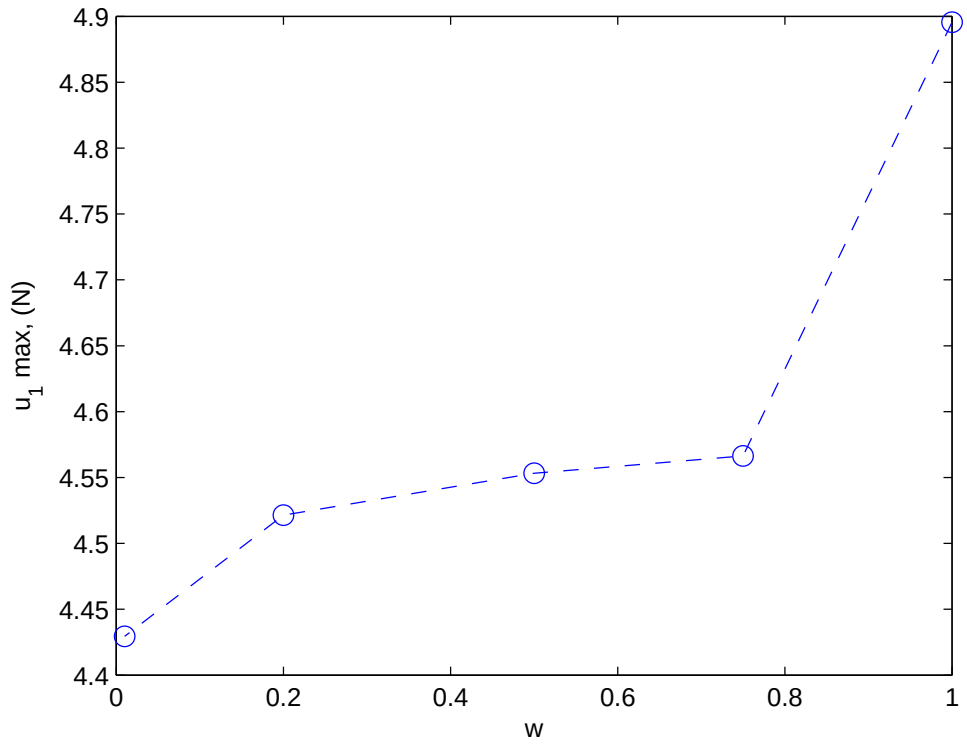


Figure 5.14: The effect of varying the weighting coefficient **w** on the absolute maximum control input u_1

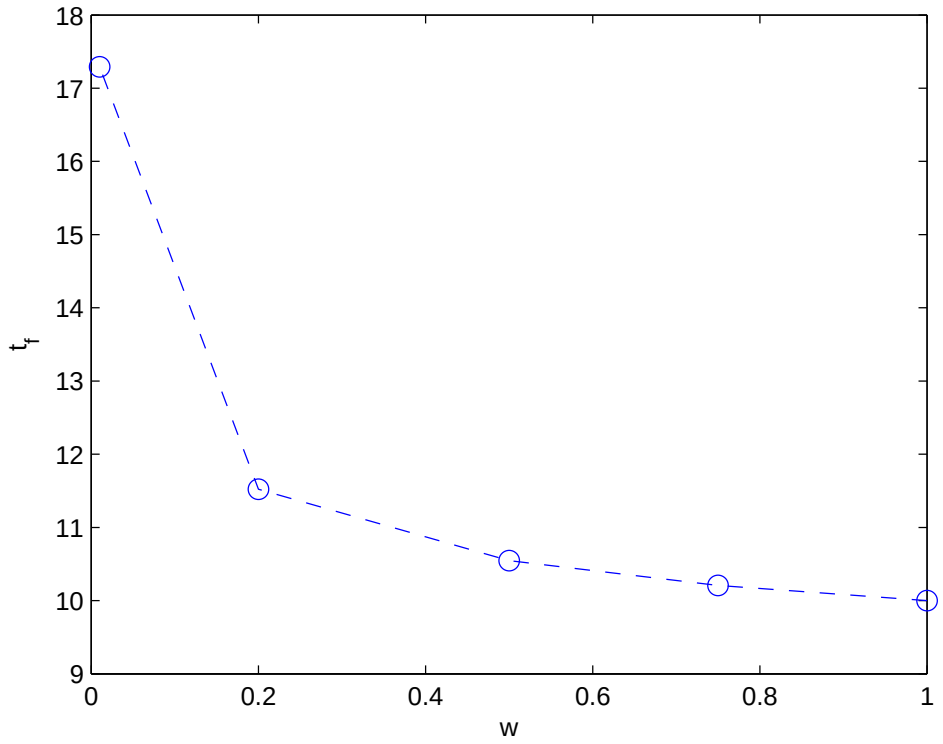


Figure 5.15: The effect of varying the weighting coefficient **w** on the final time t_f

We start optimization at some point above this curve and meet it at a certain point defined by **w**. The two well known analytical solutions, minimum time and minimum fuel are the two extremes of this plot but in practice we are more likely to be interested in the region between these points. Having this curve a designer can introduce proper scaling coefficients to reshape this curve or make focused tradeoffs.

Mission (i (d))

Figures 5.17-5.20 demonstrate the quality of LQR controller in tracking trajectories produced by the trajectory generator. Figure 5.17 presents the case of the following a six-metre vertical transition reference trajectory with no disturbances. As seen, the simple LQR controller does the job fairly well. Moreover, the intentionally introduced discrepancy in the initial acceleration is being corrected in a timely manner (the speed profile shown in Figure 5.18 corrects in a half of a second).

Mission (i (c))

Figure 5.19 shows the case of the same vertical transition trajectory but in case of constant downward wind of 0.1m/s. As seen, despite the larger discrepancy in the speed profile due to the constant downwind in the beginning of the trajectory, the altitude profile re-

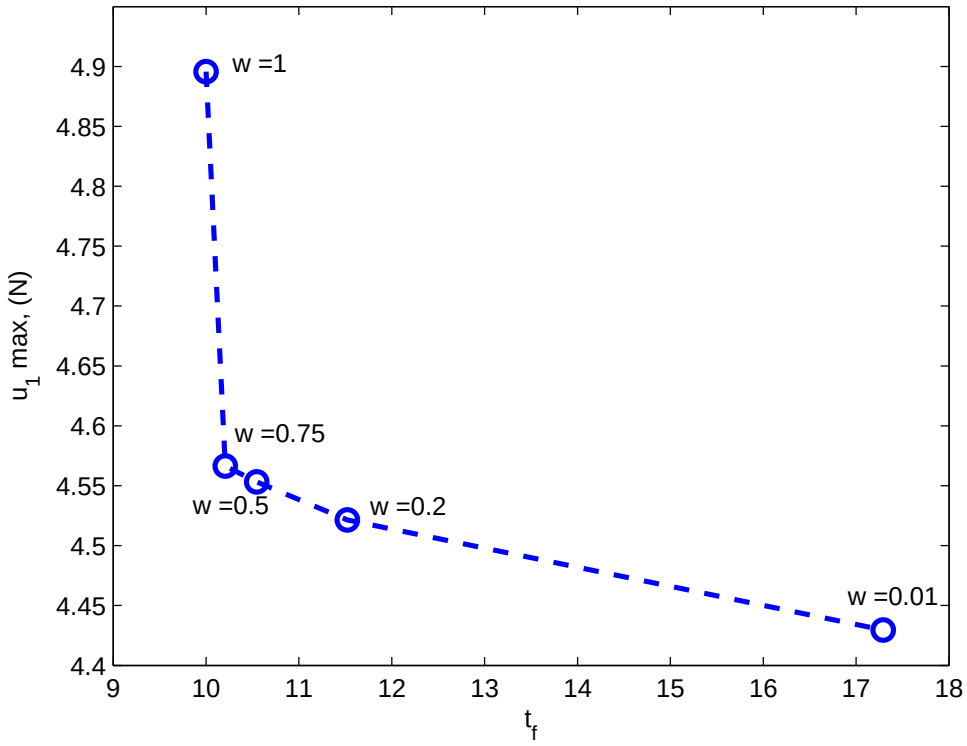


Figure 5.16: The Pareto frontier for the formulation including two conflicting terms in the cost function

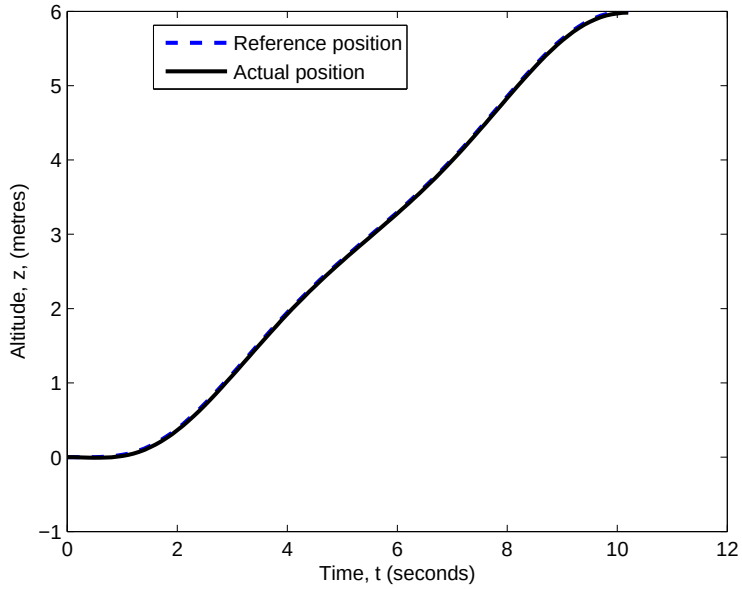


Figure 5.17: Vertical flight

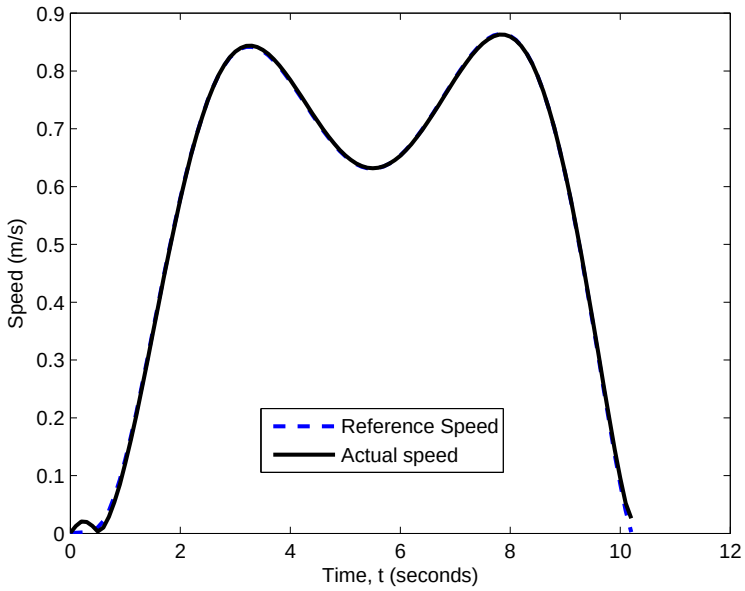


Figure 5.18: Vertical flight speed profile

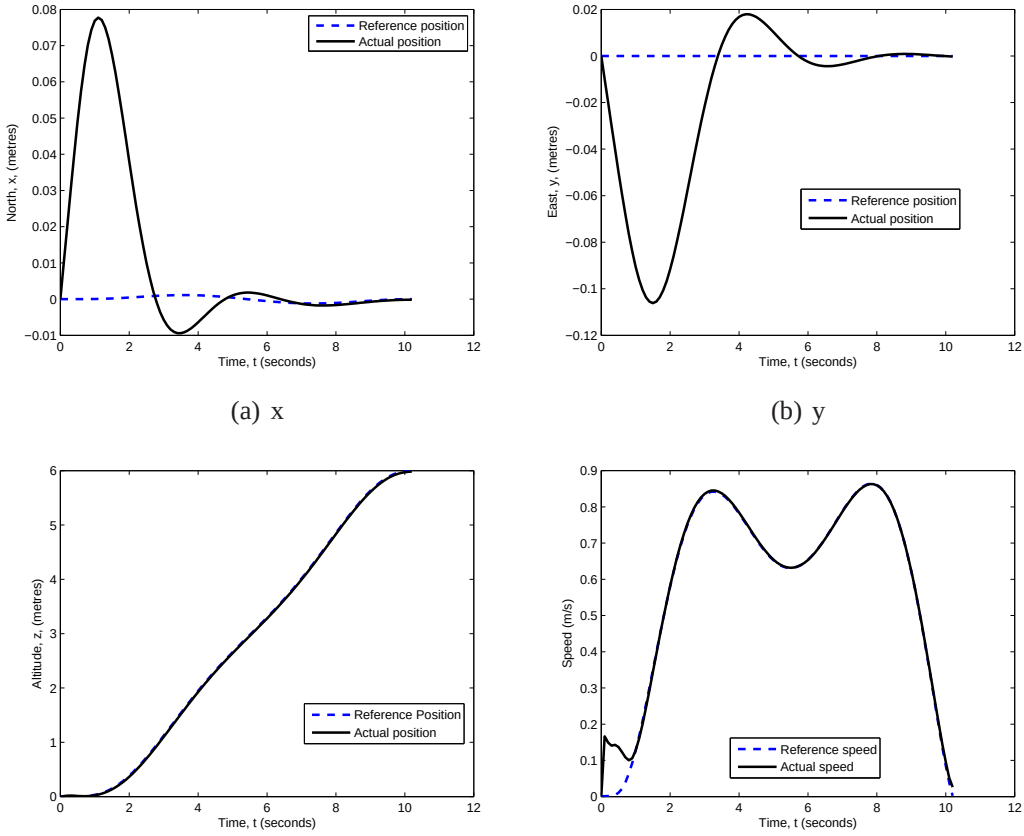
mains untouched. The speed profile tracks the reference profile well after the controller compensates for this disturbance and the thrust is increased. A constant wind of the same magnitude is also applied along the x and y axes, despite this constant disturbance the controller compensates and the trajectory is followed closely.

Mission (i (e))

Finally, Figure 5.20 presents the case of gusty winds. A constant downward and cross wind along the x and y axis is applied as before with a magnitude of 0.1m/s. As well as this, turbulence is added along all three axes in the form of a Gaussian random displacement, with a mean of 0m/s and a variance of 0.01 m/s. A small constant error is present within the x and y position but the altitude profile remains unchanged despite the turbulence. Within Figure 5.20 (d) there are two sharp peaks on the actual speed, this is due to combined effect of the strong wind and the gusting. Within Figure 5.20(d) there is an initial downward peak as the downward wind accelerates the vehicle downward, the controller then compensates by inputting a large thrust shown by the second peak, the consistent offset is then present as the vehicle accurately tracks the reference position as seen in Figure 5.20(c).

Mission (ii (c))

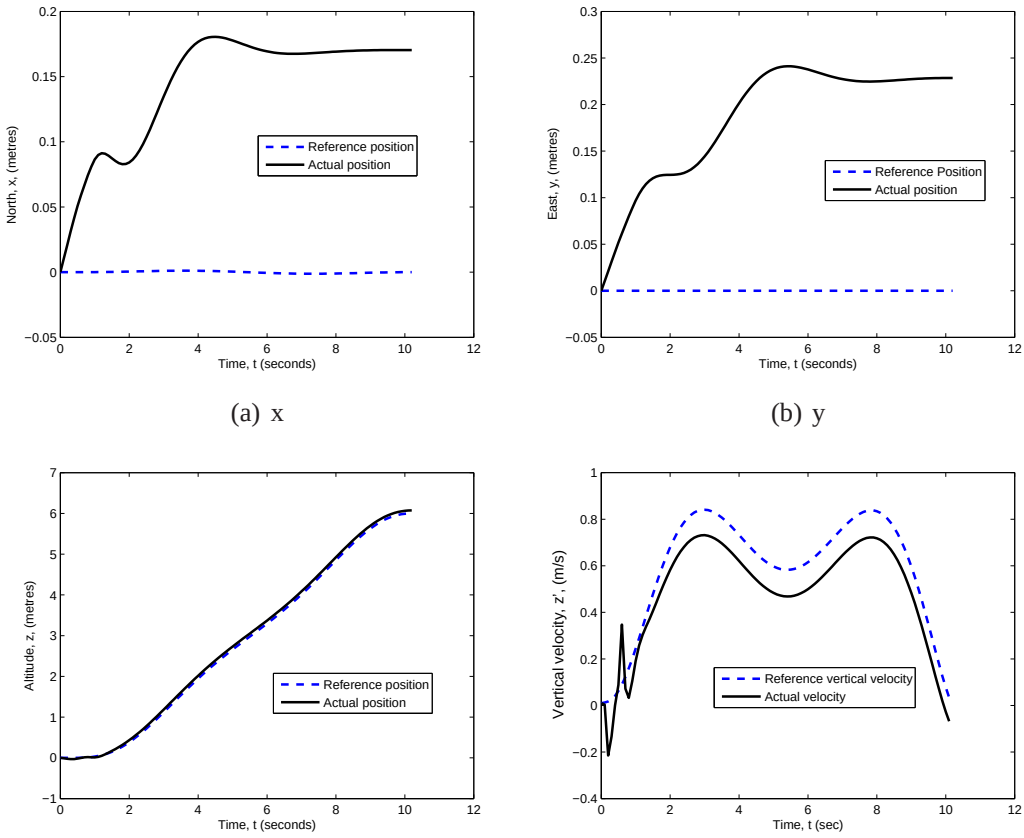
The second mission is an obstacle avoidance mission, the vehicle must fly from the origin to the destination at $[6,0,0]$. An obstacle modeled as a sphere is centered at $[3,0,0]$ with a radius of 1m. Consider the case where to improve disturbance rejection around



(a) x (b) y

(c) z (d) speed

Figure 5.19: Vertical flight with constant wind



(a) x (b) y

(c) z (d) speed

Figure 5.20: Vertical flight with constant wind and turbulance

the obstacle it is desired to reduce the radial speed V around the obstacle to minimize the chance of collision. This scheme can easily incorporate a modified cost function to deal with such a problem by modifying the existing cost function as shown:

$$\Phi = (1-\mathbf{w})\frac{1}{t_f}\left(\int_0^{t_f}\sqrt{(P_1\dot{x}^2+P_2\dot{y}^2+P_3\dot{z}^2)}\,dt + P_\gamma\int_0^{t_f}\frac{V_\gamma}{D_{Ob}^2}\,dt\right)+\mathbf{w}(t_f-T)^2 \quad (5.53)$$

where

$$V_\gamma = \frac{\sqrt{\dot{x}^2+\dot{y}^2+\dot{z}^2}([x,y,z]-[3,0,0])^T}{D_{Ob}} \quad (5.54)$$

and D_{Ob} is the distance to the centre of the sphere

$$D_{Ob} = \sqrt{(x-3)^2+(y-0)^2+(z-0)^2} \quad (5.55)$$

The factor D_{Ob}^{-2} assures that we are only concerned about the radial velocity when we are close to the obstacle. Figures 5.21 and 5.22 show the results of the simulation with $T=0$, $P_1=P_2=P_3=P_\gamma$ and $\mathbf{w}=0.1$. The optimal trajectory is now slightly further away from the obstacle due to the cost function now being a function of the distance, as seen in Figure 5.21. The speed is reduced around the obstacle as can be seen from Figure 5.22 compared with the same mission with the original cost function. Figure 5.21 also shows the flight path for the vehicle and demonstrated close tracking of the reference trajectory despite the presence of the disturbances.

Mission (vii)

Up to this point obstacles have been modelled as spheres; a new mission is now defined which simulates flight in an urban environment. Mission (vii) is a horizontal flight around some tall buildings, therefore the vehicle must find a trajectory around the buildings and not over. In this mission we also consider the case of incorrect or incomplete information. As the vehicle navigates around the first building a second building comes into view as seen in Figure 5.23. These buildings are assumed to be square based cuboids. The vehicle must reach a destination of $[7,0,0]$. In this mission there is a constant wind acting on the vehicle of 0.1m/s along the x, y, and z axis. It is possible to approximate a square obstacle and still maintain the convex smooth problem as discussed (3.4.1) through the relationship:

$$R_s = \sqrt[n]{x^n+y^n} \quad (5.56)$$

where n is a large number, as n tends to infinity the square approximation improves; however as n increases and the distance from the center becomes small, rounding errors cause numerical problems within the routine. For the simulation n, was chosen to be 12, to approximate square buildings. Initially only one building is in view of the vehicle and the trajectory is planned accordingly. As the vehicle navigates around the building another building comes into view making the reference trajectory infeasible as it passes through the second building. A new trajectory is then determined around both buildings and the vehicle follows the new trajectory. As the current state is set as the initial conditions in

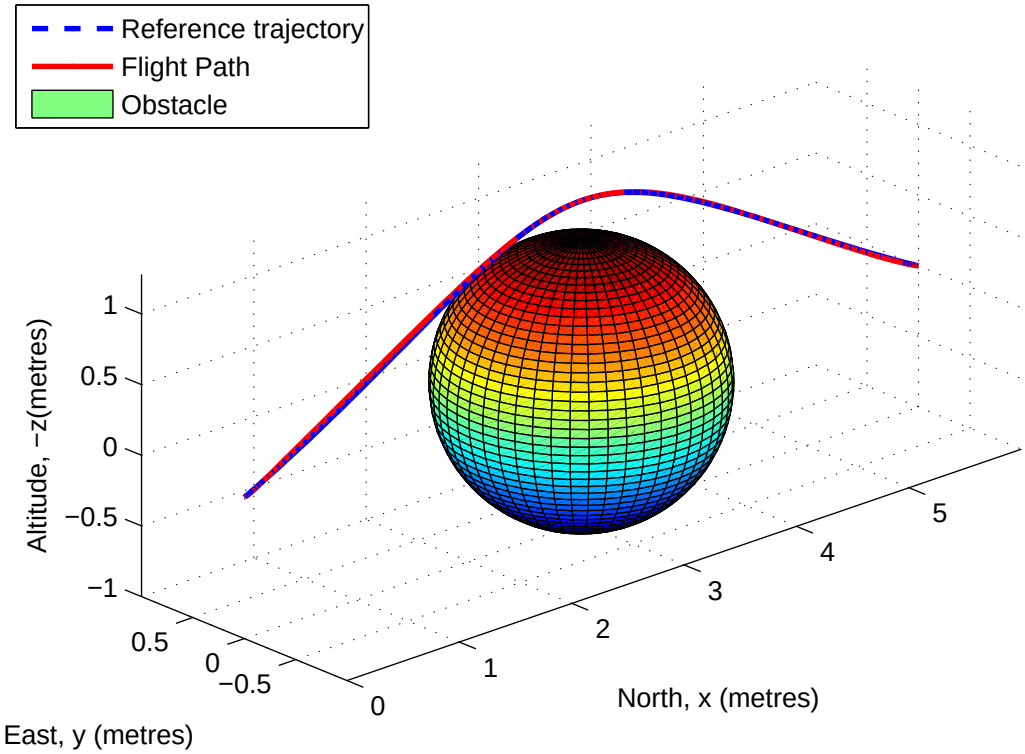


Figure 5.21: Direct Method obstacle mission

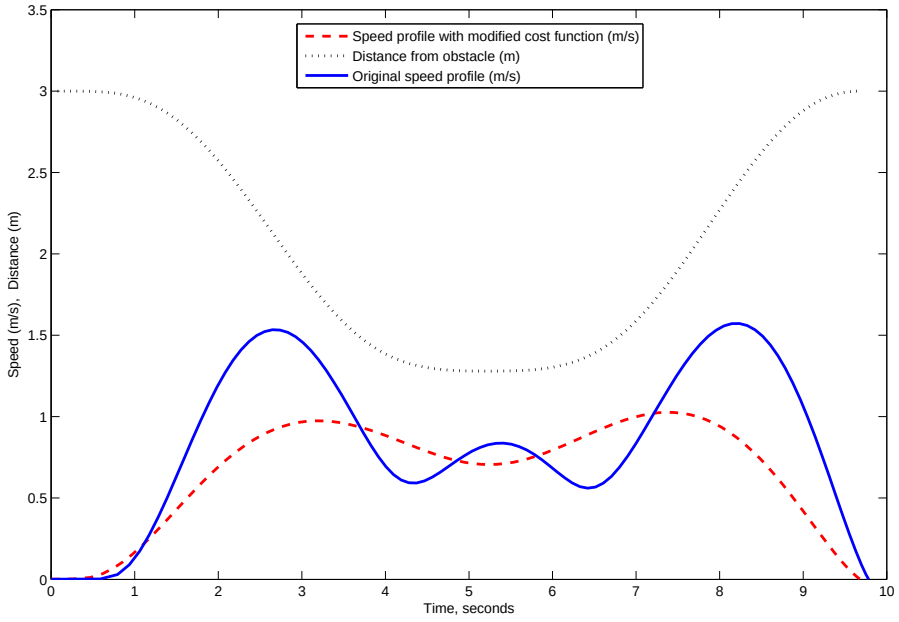
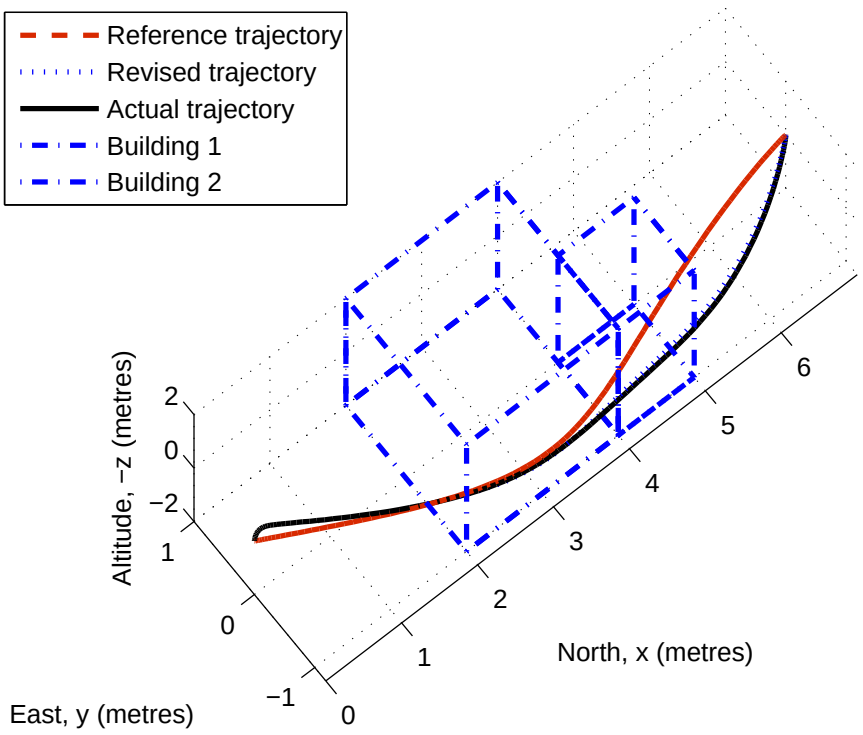


Figure 5.22: Speed profile for obstacle mission

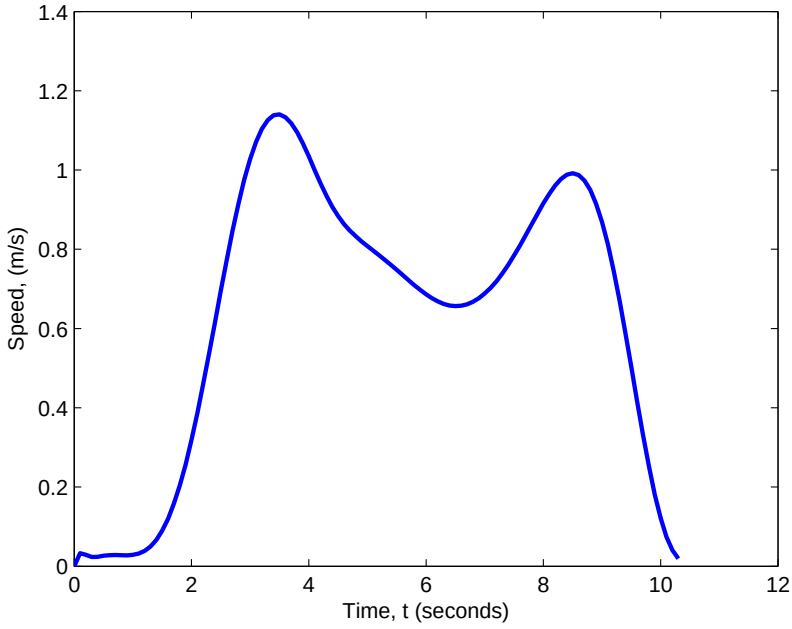
the trajectory optimization a smooth transition from one trajectory to another occurs this can be seen in the speed profile for the flight in Figure 5.23.

Mission (iii (c))

The fourth mission is a mineshaft mission with retargeting mid-flight. The vehicle initially determines a trajectory to fly 6m north and then down a mineshaft with a pre-determined time of arrival of 10 seconds. After 4 seconds however the target moves and a new target in introduced at $(10m, 5m, 0m)$, which requires a new trajectory to be determined (Figure 5.24). This mission demonstrates a smooth transition for dynamic retargeting as well as dynamic change of constraints shown by the urban environment mission. The initial reference flight time is 10.3 seconds but after retargeting this increases to 12.9 seconds. The initial drift from the reference speed profile is due to the constant wind along the x axis, this results in the vehicle drifting ahead of the reference position so the vehicle slows down as seen in 5.24(b), which explains why the actual speed drops below the reference speed.

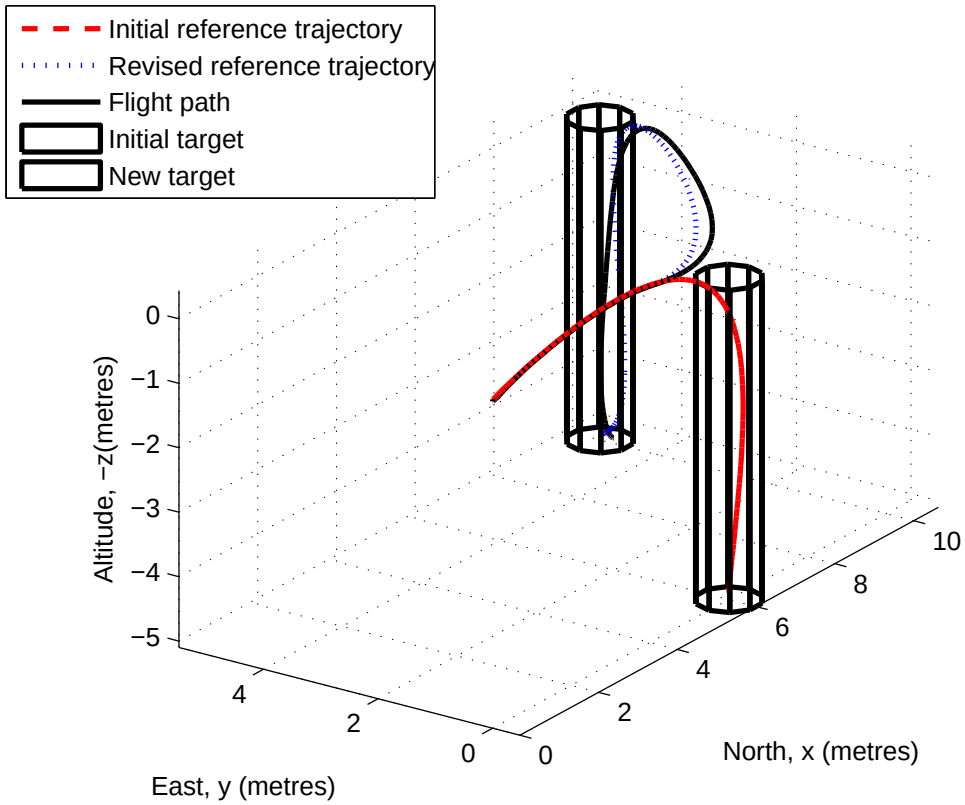


(a) Building mission

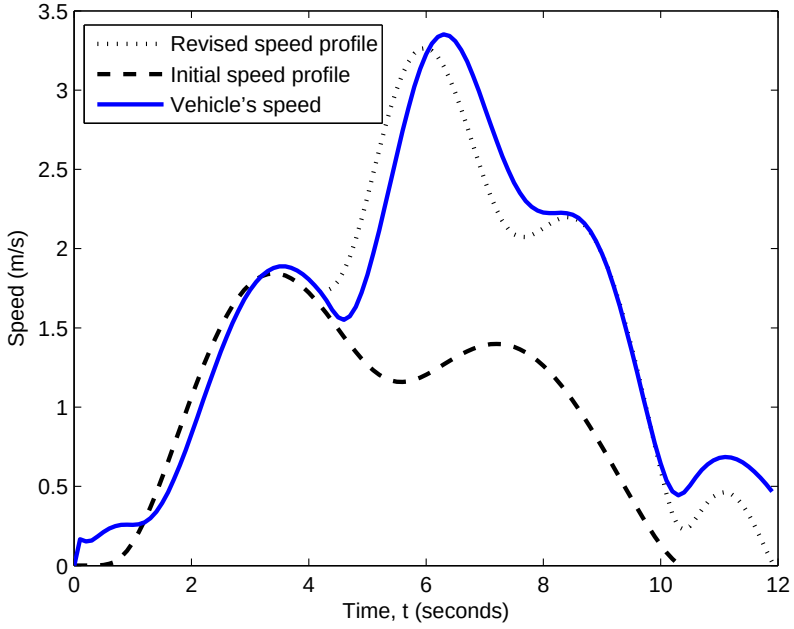


(b) Building mission speed profile

Figure 5.23: Building mission



(a) Mineshaft mission



(b) Speed profile

Figure 5.24: Mineshaft mission

Chapter 6

Multiple Vehicles

6.1 Introduction

When considering small UAV's such as the quadrotor there has been considerable interest in the control of multiple vehicles. When considering UAV applications, there are many possible advantages to deploying multiple vehicles as opposed to a single vehicle. these include an increased surveillance area, inspection from different angles and a reduced dependency on a single vehicle.

The term rendezvous implies a meeting at a common point at a set time, the motivation for this could be, for example, the completion of a more complex task or for battery recharging. For multiple vehicle flight the issue of rendezvous is a challenging problem, which is complicated by the necessary conflicting problem of collision avoidance. Also, in order to rendezvous, determining the position of the other vehicles is necessary, however, there are a number of scenarios where this might not be possible. The quadrotor due to its size has a relatively limited payload and therefore communication hardware may be an infeasible load. Also as discussed in Chapter 7, determining the position of a vehicle indoors is a challenging problem as GPS signal is not available. In this case visual sensors may be guiding the vehicle with respect to a relative position, such as a nearby doorway, as opposed to absolute position. There are also bandwidth considerations, a vehicle may have sufficient bandwidth to communicate with another vehicle but for rendezvous it would need to communicate with every other vehicle. Finally there is growing interest in guidance within GPS denied regions and even if GPS is not denied then the signal is variable and only has limited accuracy. Therefore it is unlikely that any rendezvous algorithm is going to be purely based on GPS information. It is therefore possible, for a number of reasons, that the quadrotor will not be aware of other vehicles in the vicinity, and the only method of detection is through an on-board sensor, such as an omni-directional camera. Any camera will only be able to detect other vehicles within a finite range determined by the camera resolution and the size of vehicle.

This chapter will look at one particular decentralized rendezvous algorithm. A decentralized algorithm is one which is run on board each vehicle independently, as opposed to at a

central ground station. This scheme assumes there is no communication between vehicles and no predetermined meeting place. In fact there is no global knowledge between the vehicles except that all the vehicles use the same algorithm. It is also assumed that there is a limited sensor range and therefore each vehicle may not be able to see all of the other vehicles, although each vehicle can see at least one other.

6.2 Circumcenter law

This section demonstrates a two dimensional point convergence algorithm as developed by Ando *et al.* (1999) for autonomous vehicles with limited visibility. The algorithm is memoryless, as at each step the next position is determined by the position of the vehicles in the visibility range, at that time, and independent of the previous history of the system. By initially ignoring the problems of collision, and assuming that each vehicle is within visibility range of at least one other vehicle, an algorithm can be developed to rendezvous all of the vehicles. Since each vehicle has a limited visibility it is likely that initially only some of the entire group will be in visibility range. The method is a 4 step algorithm which is repeated until the rendezvous is complete.

1. Observe position of other vehicles
2. Determine next position.
3. Move to the new position.
4. Repeat until the rendezvous is complete

6.2.1 Determining the next position

An algorithm is required at the beginning of each step, to determine the next position of each vehicle. Consider a group of four vehicles at $t = 0$, with limited visibility such that only one other vehicle is visible to vehicle A as seen in Figure 6.1. The smallest encompassing circle is determined for these two vehicles (Figure 6.2). The vehicle then proceeds to the assembly point. This process is repeated by all vehicles simultaneously until all vehicles meet at the same point as shown in Figure 6.3.

6.2.2 Centre of the smallest possible circle

The smallest enclosing ball is a well known, non-trivial, computational problem (Gartner 2008). One approach to the problem is by first removing from consideration, the vehicles which have no influence on the center of the smallest circle (inactive vehicles). To do this it is necessary to first calculate the distance between all vehicles within the visibility

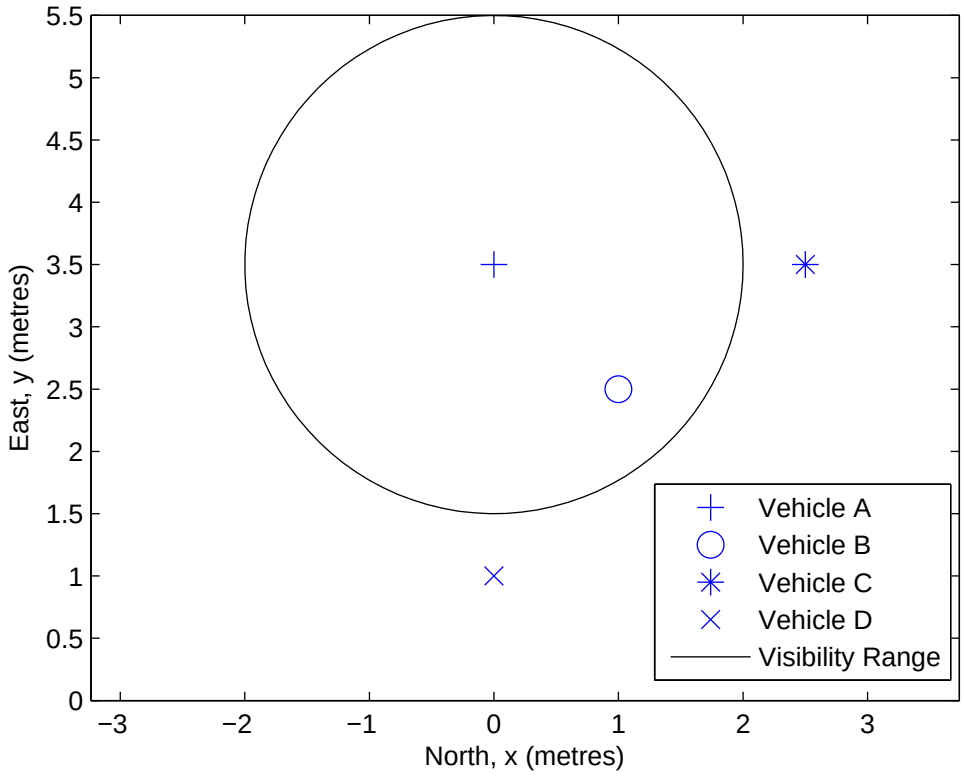


Figure 6.1: Multiple Vehicles, starting points.

range. In 2D, for a circle center to be calculated requires between 2 and 4 points, the other vehicles are not influential on this circle center.

In the most simple case, where only 2 vehicles are within the visibility range, the center of the smallest encompassing circle lies at the mid point between the two vehicles as shown in Figure 6.4.

In the case of 3 vehicles, the smallest circle center is calculated in a number of steps. Firstly the two greatest distances are calculated and the mid-point co-ordinate of each distance is found. In the case where 3 distances are all the same length, then any two of the three distances are used. The center of the smallest circle is then found at the meeting of the lines perpendicular to these points.

Finally for the case of 4 or more vehicles, again the first step is to calculate the two longest distances. If the two longest distances form a triangle between 3 points, then the 4th point is discarded for this calculation as it is inactive and the center is found as before for 3 points. If the two longest distances do not have a common point and 4 points are 'active' then 2 mid-points exist. In this case the center of the smallest circle is the mid-points of these mid-points as seen in Figure 6.6.

The smallest center is therefore calculated with the following algorithm.

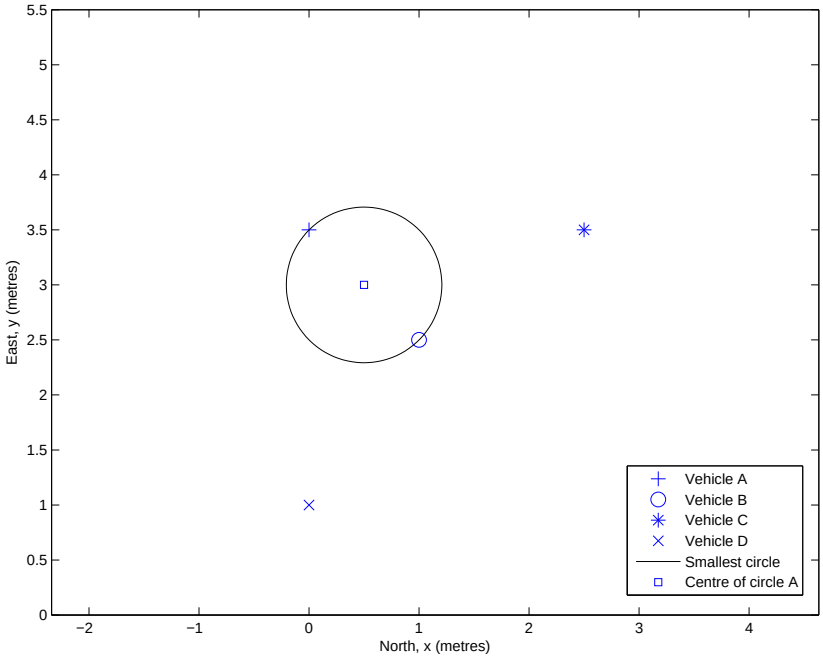


Figure 6.2: Center of encompassing circle A

1. Determine all vehicles in the visible set $[V_1V_2V_3....V_{n-1}V_n]$

2. Calculate the distance between all vehicles $[d_{12}d_{13}....d_1n....d_{n-1,n}]$ in the visible set

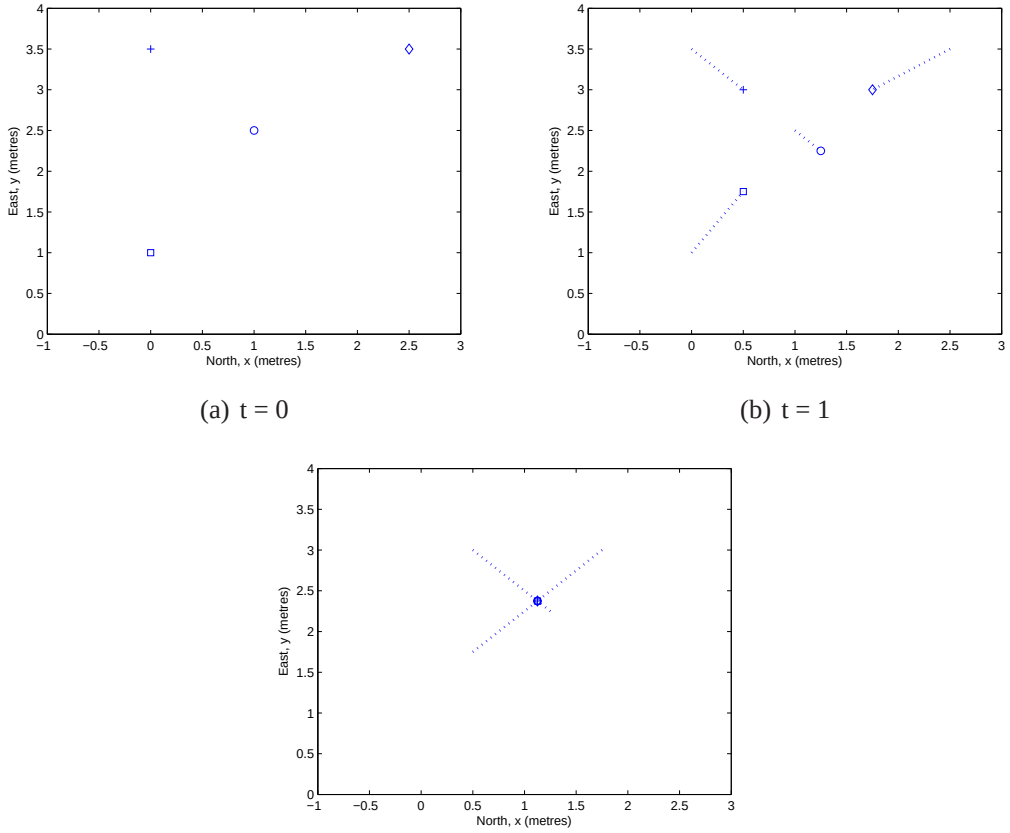
3. Determine the 2 greatest distances between points $[d_{a,b}d_{c,d}]$

4. Remove 'inactive' vehicles, leaving vehicles $[V_aV_bV_cV_d]$

5. Calculate centre of smallest circle

6.3 Extension to 3 dimensions

The previous section discussed the circumcentre law to ensure the rendezvous of multiple vehicles. This law ensures that for the two dimensional problem all vehicles will rendezvous at a given point. For UAV operation it is likely vehicles will be at different altitudes and therefore the problem becomes a 3D problem. With the addition of a omnidirectional camera the visibility range would now take the form of a sphere. This problem differs from the two dimensional one as now it is necessary to determine the smallest possible sphere encompassing all of the visible vehicles. In the same way as the 2D problem,



(a) t = 0

(b) t = 1

(c) t = 2

Figure 6.3: Rendezvous in 2 dimensions

the 3D problem can be solved by considering only the active vehicles. In this case however, the three greatest distances need to be calculated producing between 2 and 6 active points. If two of the greatest distances share a point (common point), a plane is defined with these 3 points on the surface and the center of this plane is found, as discussed for the 2D case (Figure 6.5). However this is not necessarily the center of the sphere as there are now 3 greatest distances. If 2 lines do not share a common point, then as before both mid-points are calculated and the center is the average of these mid-points.

1. If two distances share a common point, a plane is found and the resulting center is calculated using the 2D rules.

2. If no common point is shared between distances, the center is the average of the mid-points.

6.4 The control algorithm

To demonstrate the practicality of such a scheme this work simulates the flight of a number of quadrotors which are required to rendezvous. This simulation uses Taranenko's direct

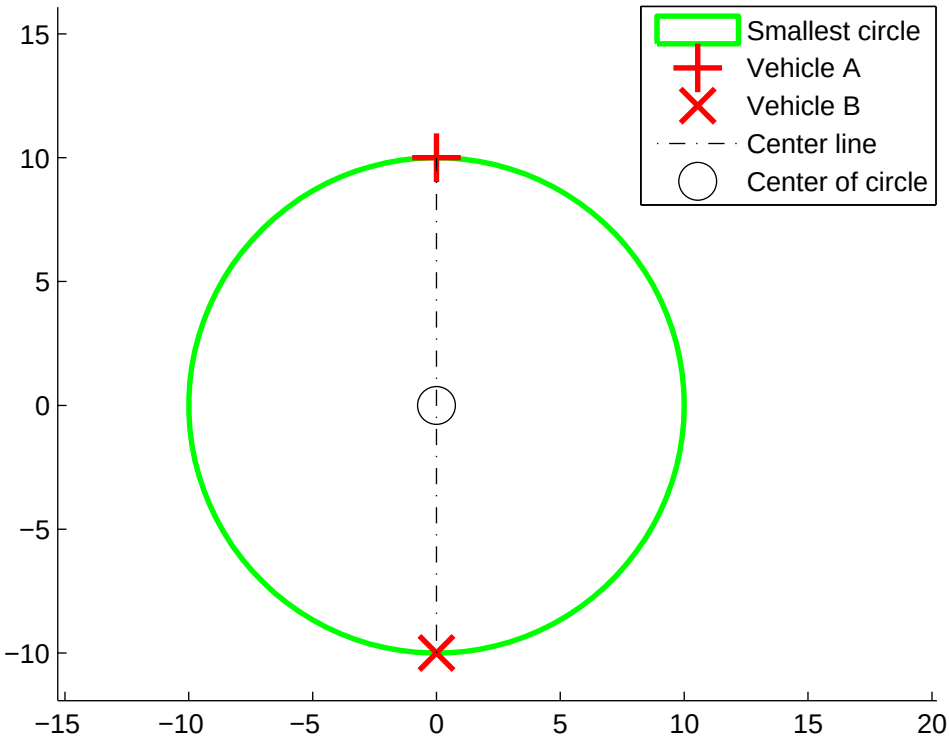


Figure 6.4: 2 point circle centre

method, as discussed within Chapter 5, and is implemented within the control structure presented in Figure 4.17. The mission planner will determine the destination, which is dependent on the position of the other vehicles. The trajectory planner will determine the optimal trajectory to get to this destination before the next time step and the trajectory follower will track this trajectory. The benefit of using Taranenko's method for this problem, is its analytical boundary condition satisfaction. At the first time step (j) the mission planner determines the new reference position of the vehicle at the next time step $(j+1)$ and the trajectory determines the reference trajectory between these time steps where

$$\mathbf{x}_j = \begin{bmatrix} x_j & y_j & z_j & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.1}$$

and

$$\mathbf{x}_{j+1} = \begin{bmatrix} x_{j+1} & y_{j+1} & z_{j+1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.2}$$

This is repeated for until the vehicles rendezvous. The update switch will monitor the feasibility of this trajectory and drift from this trajectory and switch on the trajectory planner if required.

The vehicle must travel to the center of the sphere by the end of the time horizon. As the problem is now multiple vehicles then it is obviously necessary to consider collision avoidance. This problem has not been considered by (Ando *et al.* 1999) and is complicated by the decentralized nature of the problem, in other words, no vehicle has prior knowledge of the others vehicles trajectory. The trajectory optimization scheme

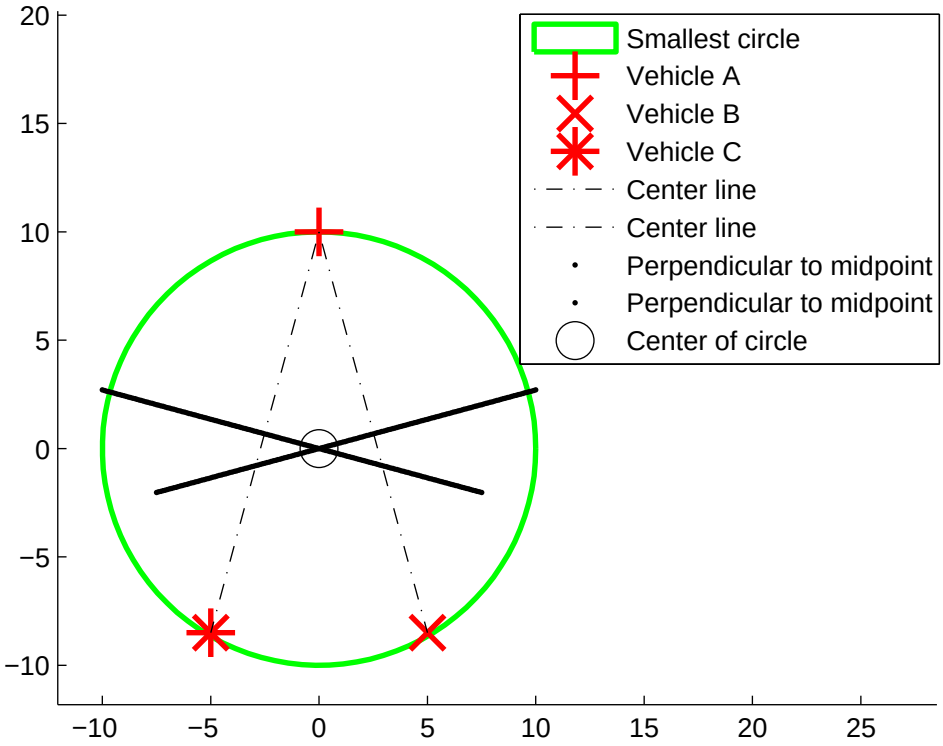


Figure 6.5: 3 point circle centre

can be applied to plot an initial trajectory to reach the required destination and to avoid the static obstacles. However other vehicles with unknown trajectories provide unknown constraints on the vehicles trajectory optimization. There are two potential approaches to this problem considered within this work, a stochastic analysis of a safe path to the destination or a rule based approach such as if two vehicles are within a certain distance then one vehicle climbs 2 meters or hovers. Stochastic analysis of an omni-directional vehicle such as the quadrotor is a challenging problem due to the vehicles high bandwidth and the required length of the prediction horizon, this would possibly result in sub-optimal or infeasible results. In this scheme a simple flight path law is applied which considers the direct consequence of the other vehicles and applies a simple flight path law, this will be discussed in Section 6.4.1.

The inner loop trajectory follower will again be a simple LQR controller which tracks the reference trajectory to the center of the circle or until the trajectory becomes infeasible. In the event the vehicle needs to hover, for example if another vehicle is very close then the LQR controller can switch to hover very easily by setting the reference trajectory equal to the current position of the vehicle and with the attitudes set to zero.

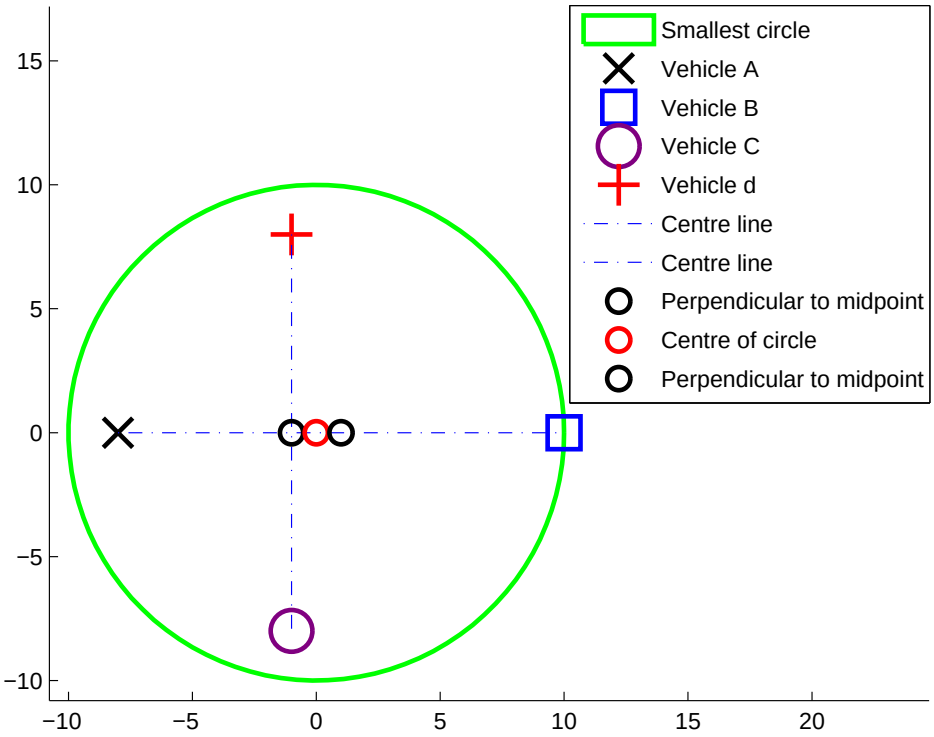


Figure 6.6: 4 point circle centre

Step time

The step time is the time required for the vehicle to reach its destination, i.e the center of the sphere. This time must be constrained, because at the end of the step time all the vehicles must progress onto the next step. It is however, possible for the vehicles to arrive at the destination early and hover until the next step. The problem is posed therefore as a constrained time problem, where the step time is fixed and the vehicles mus arrive before this time. Ideally all vehicles will converge in the minimum possible time, however, if the time is too short a feasible trajectory may not be found. Assuming a visibility range of 20 meters then the maximum distance to the center of the new sphere is under 20m, assuming this distance can be covered within 10 seconds then this provides a suitable time horizon.

6.4.1 Avoid collision but rendezvous

Now as it can be seen in Figure 6.3 the circumcentre law provides, at each time step, co-ordinates for each vehicle's destination. This guarantees rendezvous of all vehicles,

but specifically it means all vehicles will arrive at the same point at the same time, which will result in multi-vehicle collision. This presents a problem as rendezvous and collision avoidance are conflicting problems. The best way to avoid collision is to maximise distance between vehicles and not reduce it.As discussed an estimator could be used to predict the likely trajectory of the other vehicles and optimize the path around these. However due to the high bandwidth of the vehicles this is deemed unsuitable. Instead collision avoidance is removed from the initial optimization and instead collision avoidance is only considered on a direct consequence basis. If the vehicle detects another obstacle within a certain range on the reference trajectory it is able to hover, until the vehicle passes, and if the other vehicle does not pass by, then it is assumed that the vehicles have reached the rendezvous point. The update switch therefore determines the 'direct consequence measurement' which is the distance to other vehicles, which are on the reference trajectory, if this is below 2m then the vehicle hovers.

6.4.2 Multiple vehicle algorithm

The multiple vehicle algorithm reflects the same control structure presented within the single vehicle case in Figure 4.17. In addition to this model is the direct consequence measurement within the update switch which ensures collision avoidance by switching the reference state to hover in the event of another vehicle on the reference trajectory within a certain range. The full multiple vehicle algorithm is shown in Figure 6.7.

6.5 Results

To demonstrate the suitability of the circumcenter law, 3 scenarios have been chosen. The circumcenter law guarantees convergence for any number of vehicles but for computational time considerations, simulation of 5 vehicles are shown within this work.

6.5.1 Test case 1

The first test case is for a set of randomly distributed vehicles as shown in Table 6.1. As with all of the scenarios the visibility is 20m with a DCM tolerance of 2m. Despite the presence of the conflicting problem the vehicles rendezvous and the final position are reasonably close, easily within a tolerable range to establish a communication link. As seen in Figure 6.8 and Table 6.2 all the vehicles converge to the rendezvous point within 5 steps.

In Figure 6.9 the total distance (D_t) between all vehicles is calculated over the complete flight time, by:

$$D_t = \sum_{vk=1}^{vk=5} \sum_{vm=1}^{vm=5} \sqrt{(vk - vm).^2} \tag{6.3}$$

Table 6.1: Starting coordinates for test case 1

	x	y	z
v1	10	10	-10
v2	10	10	10
v3	30	10	10
v4	30	10	30
v5	10	30	10

Table 6.2: Final coordinates for test case 1

	x	y	z
v1	16.4554	13.3915	9.4396
v2	19.0220	12.4534	10.9103
v3	19.3220	12.4868	11.1195
v4	20.2634	12.0541	11.7497
v5	17.2330	13.6866	9.9507

where vk and vm are the vehicle numbers. It can be seen that despite the constraints on the vehicles to avoid collisions, that the total distance between vehicles is reduced. The discrete time steps can also be seen. As the vehicle returns to hover at each step the velocity reduces and hence the total distance gradient flattens. The theoretical minimum value for this total distance for (n) vehicles is $2n(n-1)$, in this case 40m. This theoretical minimum is assuming all vehicles can form a shape where the distance between each vehicle is the minimum distance between all vehicles (2m). In reality this will not be possible or desirable as there is little margin for error and instead a value close to this would be acceptable.

6.5.2 Test case 2

The second test case is where the vehicles start in a straight line as shown in Table 6.3. Obviously in the case where collision avoidance is not considered the vehicles would fly to the exact coordinates of the central vehicle but with collision avoidance considerations this is not the case. Although the first vehicle initially can't see the third vehicle it will eventually need to fly towards it, the problem is the second vehicle is in the direct line to this point. This scenario is therefore interesting as the conflicting problems of rendezvous and collision avoidance are very evident. As seen in Table 6.4 and Figure 6.10 the vehicles are slowly converging to a final point but very slowly, this however can be explained by the fact that the first vehicle has only just become visible to the third vehicle and therefore by adding an additional step to this simulation the convergence speed should increase. In Table 6.5 and Figure 6.11 the additional step is shown. Now the smallest encompassing sphere for the first vehicle includes the third vehicle and it is therefore closer to the theoretical rendezvous at $[38 \;\; 10 \;\; 10]$. As the first vehicle approaches this point at a greater speed then the target for the second vehicle also converges on this point, the only constraint now being the collision avoidance constraint, this can be seen by the first vehicle hovering 2m away from the second vehicle in Figure 6.5.

Table 6.3: Starting coordinates for test case 2

	x	*y*	*z*
v1	0	10	10
v2	19	10	10
v3	38	10	10
v4	57	10	10
v5	76	10	10

Table 6.4: Final coordinates for test case 2

	x	*y*	*z*
v1	20.0753	10.0000	9.9751
v2	27.0802	10.0002	9.9849
v3	38.0444	10.0000	9.9727
v4	49.0208	10.0000	9.9713
v5	55.9254	9.9999	9.9646

Table 6.5: Final coordinates with additional step for test case 2

	x	*y*	*z*
v1	26.4626	10.0020	9.9476
v2	29.0051	10.0000	9.9799
v3	38.0038	10.0067	9.9701
v4	47.1428	9.9999	9.9675
v5	49.4763	10.0000	9.9427

Table 6.6: Analytical solution for test case 2

	J = 0	J = 1	J = 2	J = 3	J = 4	J = 5
v1	0	9.5	14.25	19	28.5	38
v2	19	19	23.75	26.125	28.5	38
v3	38	38	38	38	38	38
v4	57	57	52.25	49.875	47.5	38
v5	76	66.5	61.75	57	47.5	38

This test can be compared with the analytical solution where it is assumed the vehicle moves to the center of the smallest enclosing sphere and no collision avoidance is considered. As all vehicles are on the same plane then we know they will converge as this is essentially the 2D problem which is proved within Ando *et al.* (1999). Table 6.6 shows this analytical solution. It is shown that in fact for the analytical solution the vehicles do not rendezvous until the 5th step. It can be seen though that the third vehicle becomes visible to the fifth vehicle after three steps, which is the point at which the convergence rate increases, this is delayed by one step in the results as collision avoidance is considered. It can therefore be seen that for the straight line scenario then convergence is slow and this is understandable as it takes a long time for the third vehicle to become visible to the first and fifth vehicle. It is also fair to say that convergence is never in doubt within this scenario as the vehicles only move towards the rendezvous point.

As before the total distance between all vehicles is shown in Figure 6.12. Obviously for the straight line case the total distance between all vehicles is the maximum value while retaining the minimum visibility links. In this case although convergence to the rendezvous point is at a reasonable rate, after 50 seconds the total distance between all vehicles is still large. This reiterates the fact that, despite a seemingly slow convergence rate the convergence is not in doubt using the circumcenter law.

6.5.3 Test case 3

For the third test case the vehicles are positioned strategically to ensure no two vehicles start by flying towards each other. This is achieved by placing two vehicles the maximum distance apart and then turning through 90 degrees from one vehicle and placing the third vehicle the maximum distance within range as seen in Figure 6.13. In Figure 6.14 the multiple vehicle rendezvous problem is shown with the no collision avoidance strategy. In Figure 6.15 the same mission is shown with the collision avoidance strategy. The coordinates for both scenarios is shown in Table 6.7. The collision avoidance strategy has no impact on the final coordinates of the vehicles until t_2 and at this point only the 1st vehicle is affected. By the final time step 3 vehicles have a different final position compared with the first scenario but in this case they still rendezvous but without collision.

Table 6.7: Test case 3 for A) No collision avoidance consideration. B) Collision avoidance consideration

		A			B		
		x	y	z	x	y	z
t_0	v1	0	0	0	0	0	0
	v2	20	0	0	20	0	0
	v3	20	20	0	20	20	0
	v4	20	20	20	20	20	20
	v5	40	20	20	40	20	20
t_1	v1	6.8	0	0	6.8	0	0
	v2	14	5.97	0	14	5.97	0
	v3	20	13.65	5.99	20	13.65	5.99
	v4	23.68	20	13.8	23.68	20	13.8
	v5	33.09	20	19.98	33.09	20	19.98
t_2	v1	11.27	4.72	2	10.88	4.2	1.85
	v2	13.4	6.85	2.86	13.4	6.85	2.86
	v3	16.1	10.58	6.73	16.1	10.58	6.73
	v4	26.28	17.14	13.06	26.28	17.14	13.06
	v5	29.21	20	17.34	29.21	20	17.34
t_3	v1	13.44	7.48	4.13	13.19	7.16	4.01
	v2	17.32	10.03	6.31	17.25	9.89	6.31
	v3	19.64	12.18	9.25	19.53	11.98	9.22
	v4	22.44	14.22	10.65	22.44	14.22	10.65
	v5	24.94	16.77	13.53	24.94	16.77	13.53

Finally the total distance is again measured over the flight time. In this case the convergence rate is again reasonably fast, but in this case, the final time step produces only a small reduction in the total distance. At this point the total distance between all vehicles is small and therefore the collision avoidance constraints are active, thus limiting the convergence rate. By this point however, the vehicles are deemed sufficiently close. This result highlights the benefits of the direct consequence measurement, where the impact of the additional constraints are only active when the vehicles are sufficiently close.

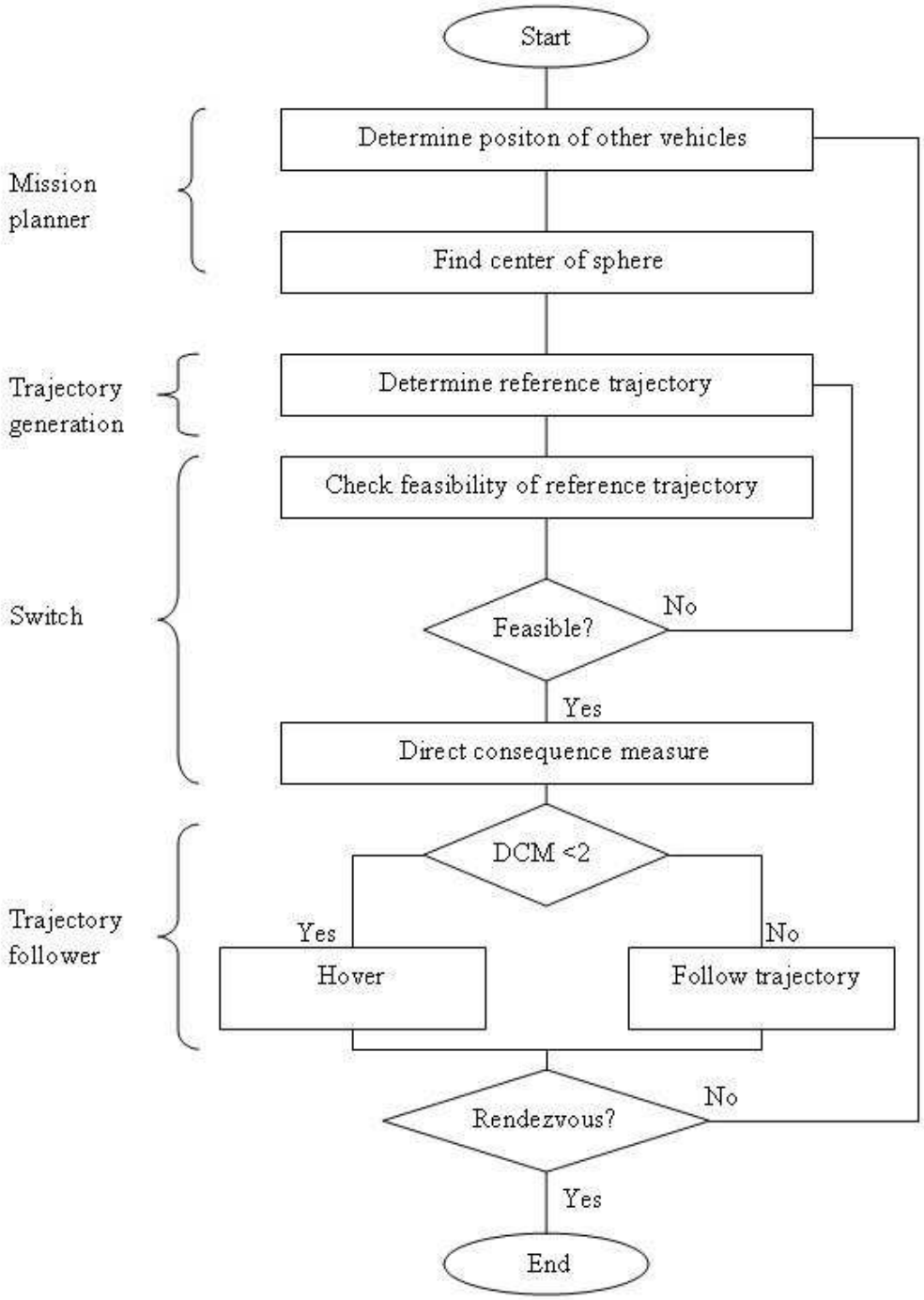
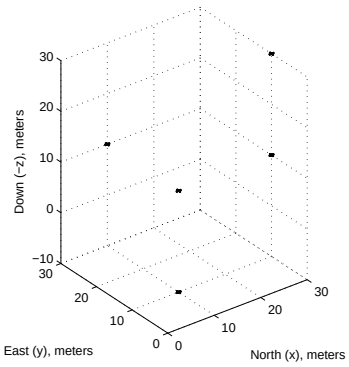
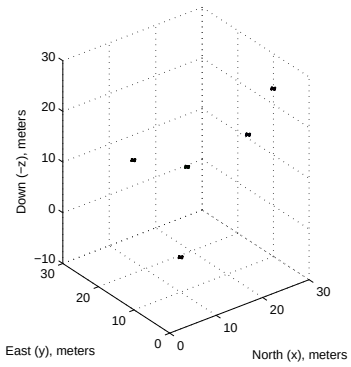


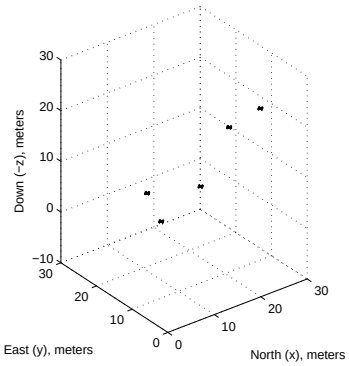
Figure 6.7: Multiple vehicle rendezvous algorithm



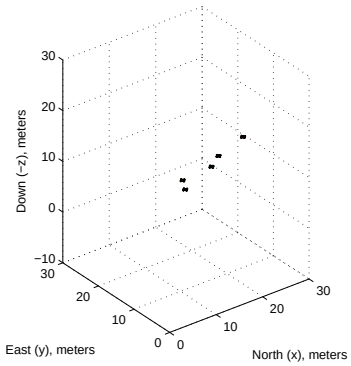
(a) J = 0



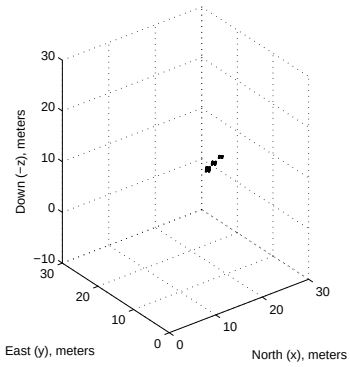
(b) J = 1



(c) J = 2



(d) J = 3



(e) J = 4

Figure 6.8: Multiple vehicle rendezvous for test case 1

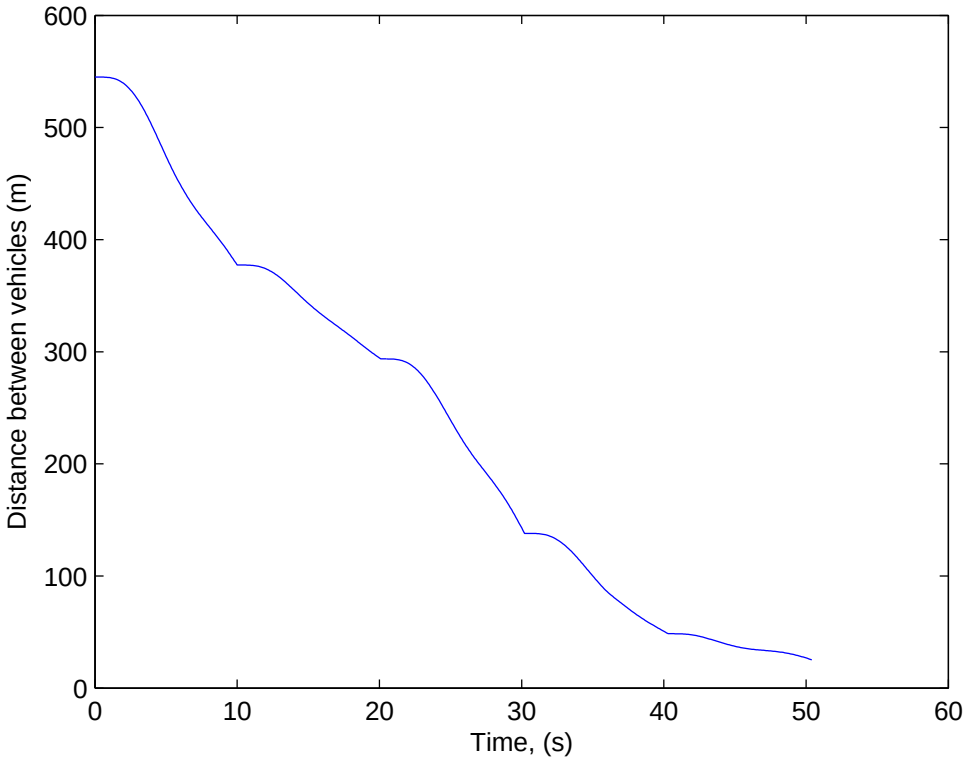
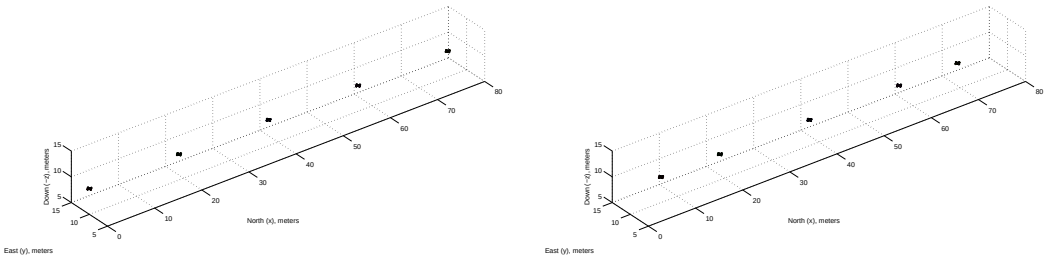
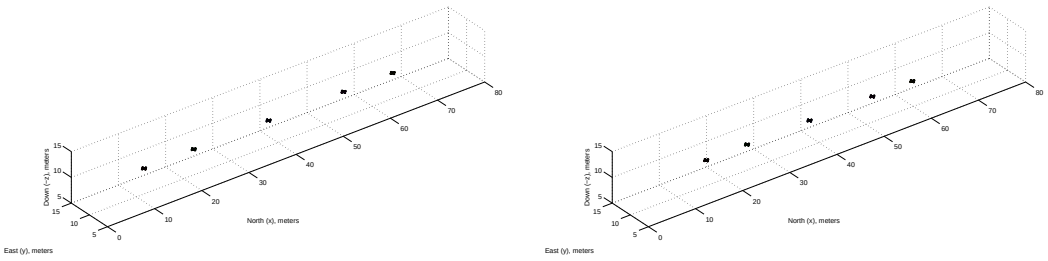


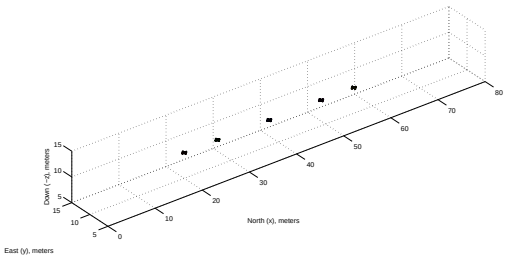
Figure 6.9: Total distance between all vehicles for mission 1



(a) J = 0 (b) J = 1



(c) J = 2 (d) J = 3



(e) J = 4

Figure 6.10: Multiple vehicle rendezvous starting from straight line

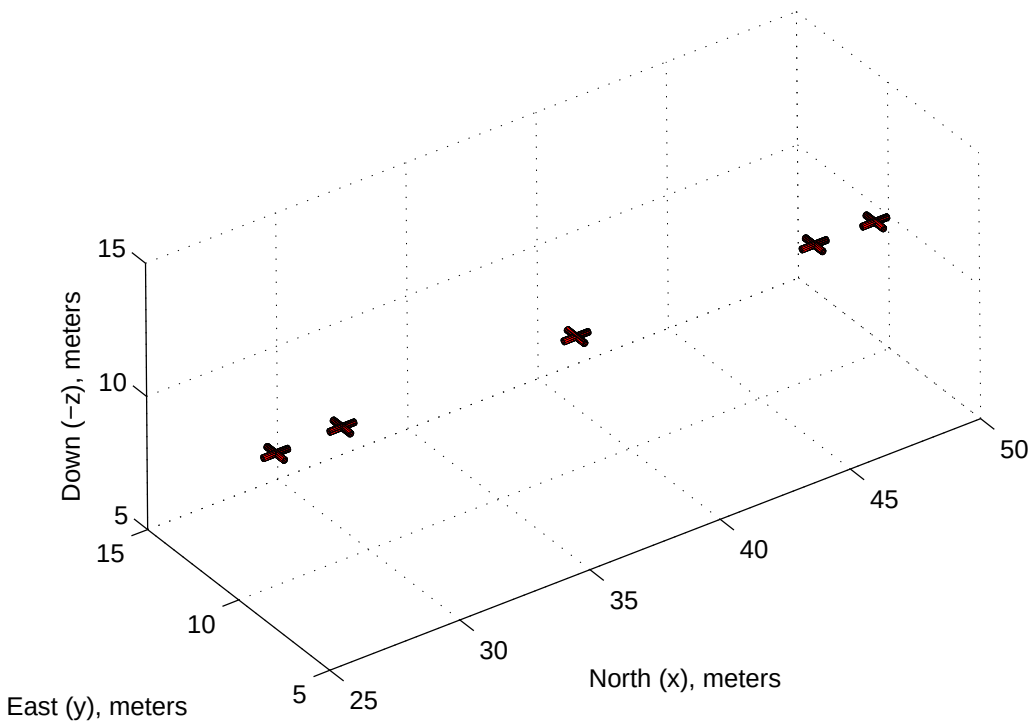


Figure 6.11: Additional step for test case 2

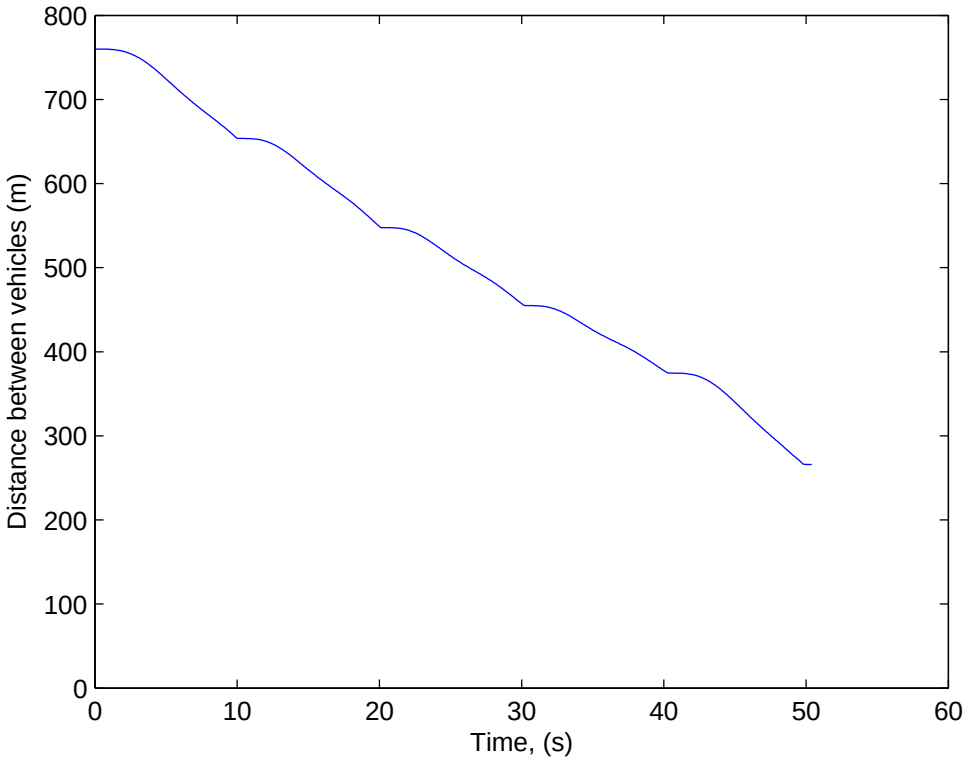


Figure 6.12: Total distance between all vehicles for mission 2

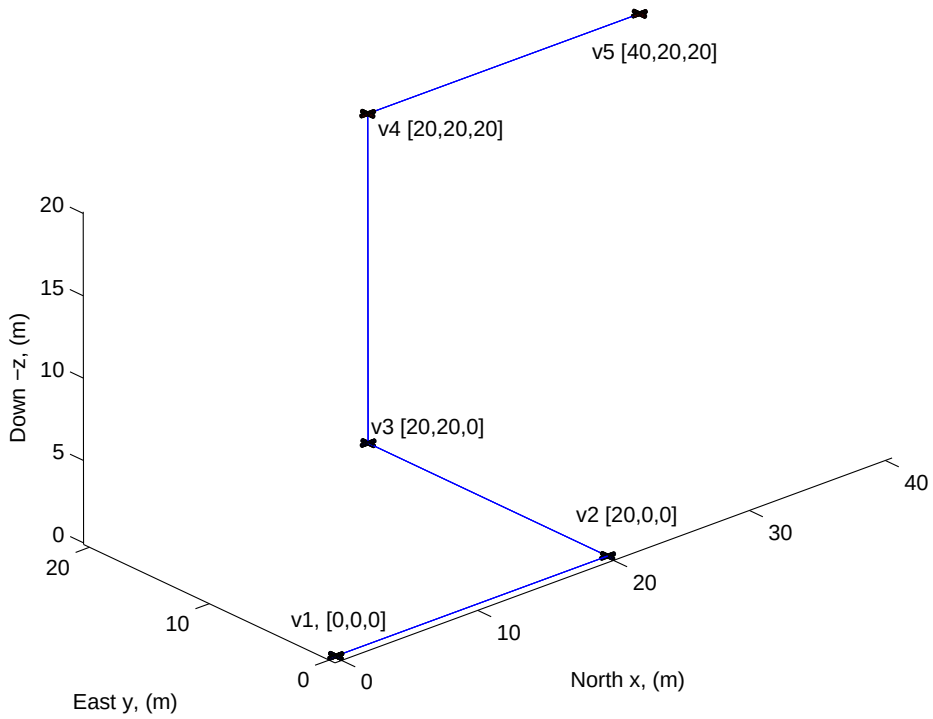
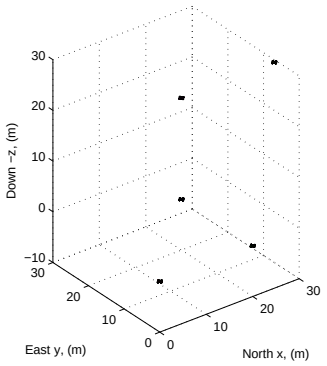
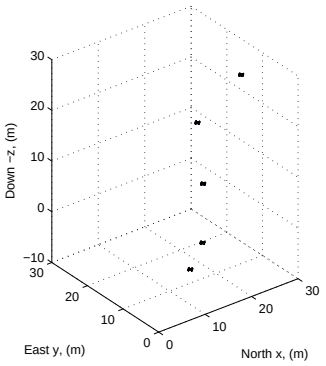


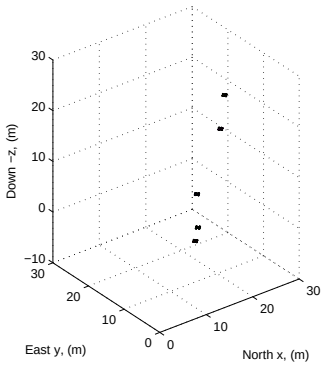
Figure 6.13: Starting positions for test case 3



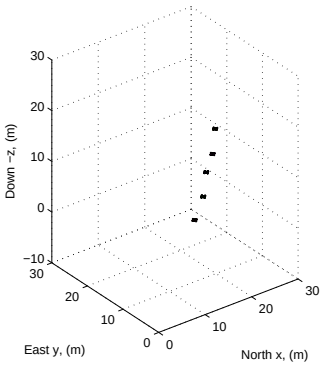
(a) J = 0



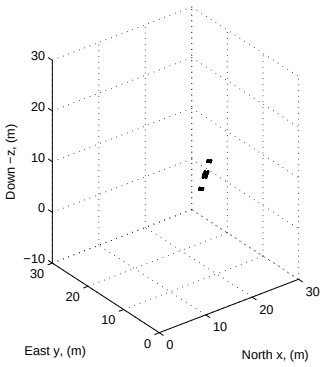
(b) J = 1



(c) J = 2

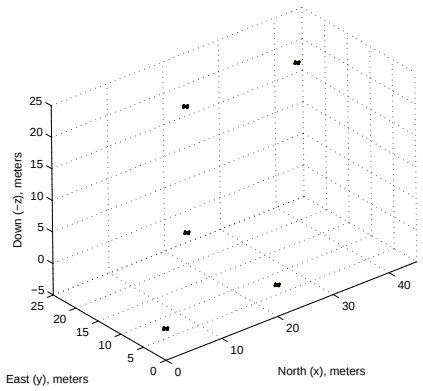


(d) J = 3

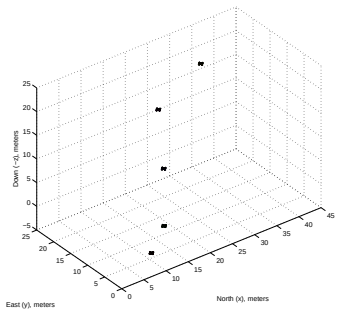


(e) J = 4

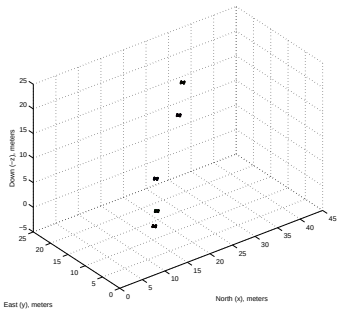
Figure 6.14: Multiple vehicle rendezvous starting from worst case scenario with no collision avoidance



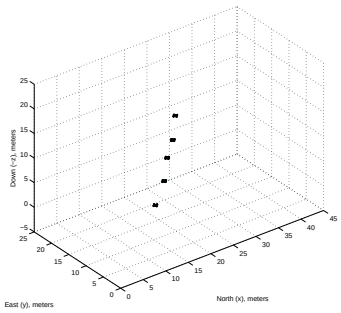
(a) J = 0



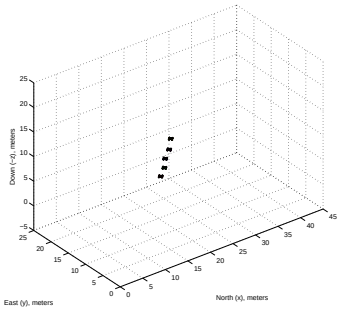
(b) J = 1



(c) J = 2



(d) J = 3



(e) J = 4

Figure 6.15: Multiple vehicle rendezvous starting from worst case scenario with collision avoidance

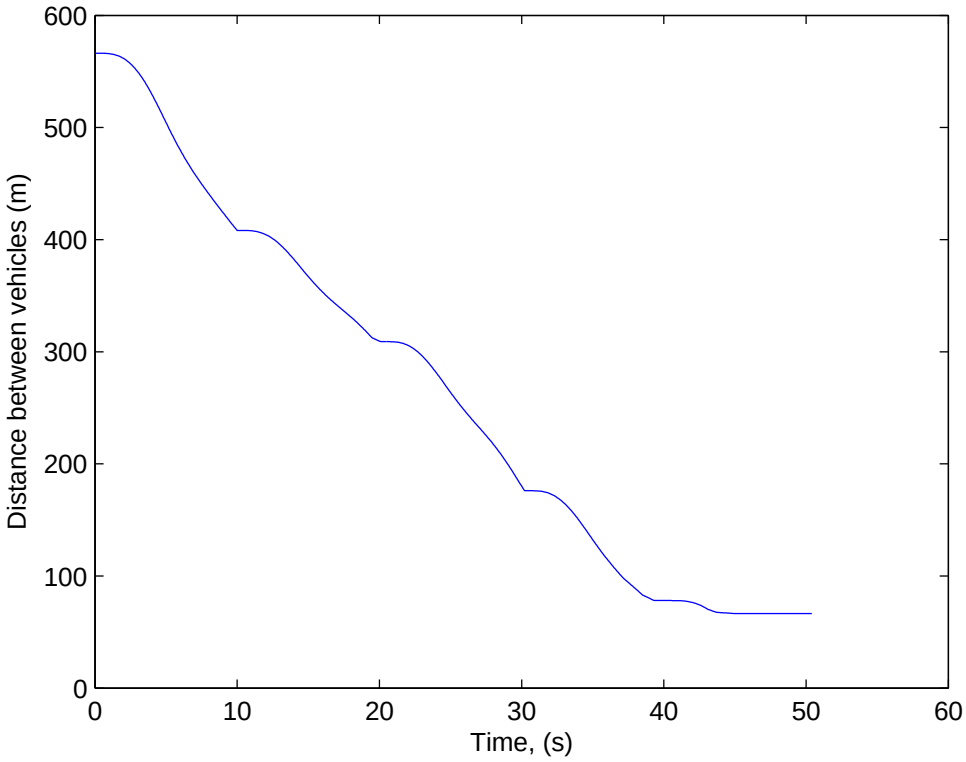


Figure 6.16: Total distance between all vehicles for mission 3

Chapter 7

Towards implementation

7.1 Introduction

To achieve autonomy of a UAV there are many issues to consider, these include issues already discussed within this work such as trajectory generation and trajectory following. Of equal importance is the issue of feedback and the practical implementation of the control algorithms, which have been presented within this work.

There are two obvious potential solutions for providing real time state feedback. One approach is to use ground based sensors to determine the vehicles states as for example demonstrated in Valenti *et al.* (2006). These systems usually are based on visual identification and can provide very accurate state information for the control laws as they are free from vibration and electrical interference. However they are typically very expensive and not portable which prevents widespread use. Very precise state feedback provides an excellent testbed for comparison of different vehicles and control laws. However, these results are only of real benefit if the near perfect state information can be obtained all of the time, in reality this is not the case as flight outside of the test facility is probably desirable.

The second approach is to obviously measure the states onboard but this introduces further problems such as payload, size and interference. The quadrotor is a small UAV and has a limited payload and therefore can not accommodate a large or heavy sensor set and control processor. Furthermore with each motor drawing up to 10A each, the issue of electromagnetic interference is likely to be an issue.

The practical task of providing feedback and hence closing the loop is a challenging problem especially on such a small vehicle. A large air vehicle would typically carry a Inertial Navigational System (INS) which provides accurate rate and acceleration information as well as a GPS unit which gives positional information. Also on a larger vehicle, ring laser gyros for example provide very accurate rate information but these are too large to incorporate onto a small UAV. GPS is also typically relied upon to provide accurate positional information, the quadrotor is however intended for internal flight and therefore GPS is

unlikely to be a viable solution. Furthermore, GPS is typically only accurate to a few meters which is probably not sufficient for a small UAV such as the quadrotor. Alternative solutions are therefore clearly required for feedback on such a small UAV.

There is a reasonable amount of interest in state estimation using visual feedback. There are a number of merits to this approach. Firstly a reduced dependency on GPS is advantageous for both internal flight or in GPS denied regions. Secondly cameras are reasonably cheap and lightweight as well as being the primary reason for the majority of UAV flights in the first place; if a camera is being used for surveillance, then why not also use it to provide state information. However, this is not a trivial problem and developing this system is beyond the realistic scope of this work.

The recent rapid development of Micro-Electro-Mechanical Systems (MEMS) sensors, provide another alternative to these traditional systems and are commonly used in the majority of MAV autopilots. One option for the quadrotor would be to fit a commercial off the shelf (COTS) autopilot. There are many commercial autopilots on the market today, offering a range of functionality (Blue-Bear-Systems-Research 2008, Cranfield-Aerospace 2008, MicroPilot 2008, Procerus-Technologies 2008). The majority include MEM's accelerometers, gyroscopes, magnetometers and numerous analogue to digital converters for connection with barometric pressure sensors and dynamic pressure sensors. Furthermore some autopilots come fitted with GPS and all are designed to accomodate GPS. The majority of this functionality is therefore not applicable to the quadrotor as for internal flight barometric pressure, dynamic pressure, magnetometers and GPS is either not required or not suitable. In fact these autopilots are almost too sophisticated for the quadrotor as with this range of functionality they are much larger than the key components, which are actually required for a small UAV flying internally. However, MEMS sensors for acceleration and rates are possibly the only feasible solution for on-board state measurement of a quadrotor. These do however come at a cost and this is typically precision.

The deployment of the control algorithm also requires consideration. On-board computation of the control action is no doubt the preferred solution for industry as this reduces time lags, however, in a experimental set up, the requirements are different. The control law may need modification in flight or at least a quick turn-around in between flights. Furthermore, the control algorithms presented in this thesis are computationally demanding and therefore, while on-board implementation is the eventual aim, off-board computation is the initial preferred option. However, this in turn presents communication challenges with lag time being the critical factor.

It is therefore evident there is a considerable amount of work involved in the development of a suitable control system. While this is not the main focus of this thesis, an appreciation of the challenges in achieving autonomy is essential. This chapter, will therefore describe the development of the Cranfield experimental set up, and the issues that arise from this work. This set up has been demonstrated by performing some open loop flights, where the quadrotor is piloted using a joystick via a PC, and the state feedback is recorded on the groundstation.

7.2 Experimental set up

For both trajectory generation and trajectory following, full state feedback is required. This requires a measure or estimation of the vehicles position (x, y, z), velocity $(\dot{x}, \dot{y}, \dot{z})$, Euler angles (ϕ, θ, ψ) and Euler rates $(\dot{\phi}, \dot{\theta}\dot{\psi})$. Measuring these states can be done from the ground using a sophisticated camera system such as the one presented in Valenti *et al.* (2006), however, in the majority of cases, it is done onboard, using an inertial measurement unit (IMU) (Carnduff *et al.* 2007).

There are many inertial measurement units on the market today but they all consist of essentially the same core components; accelerometers and gyroscopes. MEM's technology has allowed for the development of very small accelerometers which are capable of measuring the body accelerations along 3 axis $(\ddot{x}, \ddot{y}, \ddot{z})$. Solid state gyroscopes are also typically incorporated into a IMU to provide body rates about 3 axes (p, q, r).

The Draganflyer X-Pro is sold with an onboard stabilisation loop which consists of rate feedback through some solid state gyroscopes, this enables a pilot to fly the vehicle through a radio control although this is not sufficient for hands free flight. It has been decided that this inner loop stabilization should remain on the vehicle for enhanced stability and be incorporated into the new control scheme As these sensors are already on-board, it also makes sense in terms of payload, to utilize these sensors for rate measurement, rather than to add additional sensors. With these gyroscopes already on the vehicle, with the addition of a 3-axis accelerometer, the core components of a typical IMU are in place.

Position measurement is a challenging problem and is discussed in more detail in Section 7.4. However, there are several feasible solutions for height measurement. The cheapest and most available option is an ultrasonic sensor (Active-Robots 2008), which measures straight line distance by sending and receiving ultrasonic pulses. Another option is a laser altimeter which have a greater range but are generally heavier, more expensive and require more power (MDL 2008). For a low cost experimental set up, an ultrasonic sensor is the preferred solution. A mahor concern with the ultrasonic sensor, was that the propellor down-wash would interfere with the signal. A number of bench tests were performed with the quadrotor however and this was not found to be a problem. For internal flight in a room with known dimensions and clear walls it could be possible to utilize these sensors for positional determination by placing a number of sensors of the side of the vehicle.

Closing the outer loop can be done onboard or via a groundstation. The inner stabilization loop is hard wired into the circuit board and therefore left on the vehicle. For a commercial application the control action would be determined onboard, this is preferable as opposed to a groundstation because of smaller communication delays. For research purposes however, it is desirable to determine the control action on a groundstation, as the control laws can be easily accessed. This requires a reliable communication link for downlink and up-linking, where latency is a primary consideration.

Figure 7.1 shows the proposed experimental set up for validating the work discussed in this thesis. A standard hobbyist joystick is connected to the RTOS which converts the body axis demands from the pilot into serial outputs. These serial outputs are converted

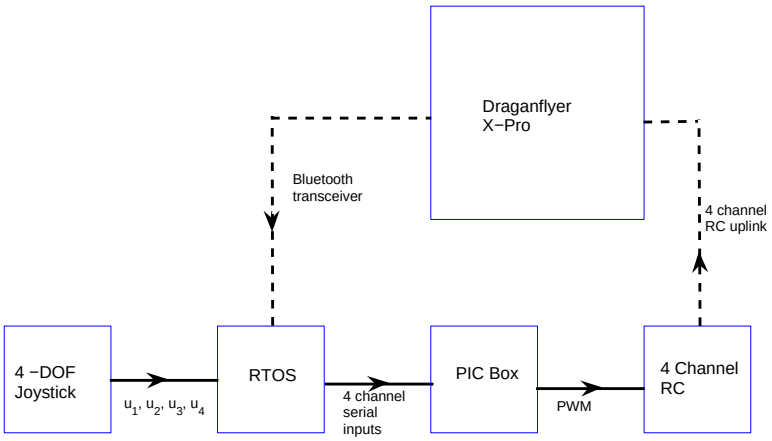


Figure 7.1: Experimental Set up

by a bespoke standalone PIC box into a PWM signal and sent to the RC controller through the buddy port. The RC controller sees this PWM signal as it would another RC controller and can switch between RC control and the signal coming from the joystick. This signal is then received by the 2.4GHz receiver onboard the vehicle and the state information is then sent via Bluetooth transceiver to the RTOS.

7.2.1 Sensors

For implementation of the control law, the following information must be measured directly:

- Angular rates in roll pitch and yaw
- Linear acceleration along the x,y and z axis
- Altitude
- Horizontal displacement

The angular rates are measured using the onboard gyroscopes. Signals from these sensors were taken directly from the circuit board of the XPro and a low-pass filter removes the high frequency noise from the signal.

A MEM's analogue 3 axis accelerometer has been fitted onto the vehicle (Analog-Devices 2008). These are capable of measuring up to $3g$, with a sensitivity of $300mV/g$, and a

sensitivity accuracy of 10%. This one was chosen as it offers reasonable accuracy and a limit of $3g$ is deemed sufficient for the quadrotor which has a maximum thrust of around $1g$. Analogue sensors have been chosen for ease of implementation and this signal is also filtered with a simple anti-aliasing filter with a adjustable cutoff frequency. The major benefit of this sensor is that it is extremely lightweight, draws very little power and also costs less than 50GBP. Further details of this component can be found in Table 7.1.

Table 7.1: ADXL330

Number of axis	3
Size	4mm x 4mm x 1.45mm
Range	$\pm$ 3g
Sensitivity	300mV/g
Supply voltage	3.6V
Supply current	$320\mu A$
Noise density	280/350 $\mu g/\sqrt{Hz}rms$
Bandwidth (x,y axis)	0.5Hz to 1600 Hz
Bandwidth (z axis)	0.5Hz to 550 Hz
Turn on time	1ms

A simple ultrasonic sensor is fitted to the bottom of the vehicle (Active-Robots 2008). This sensor is very cheap (30GBP) but provides an accurate estimate of the vehicles height above ground. Its effective range is approximately 5m, which is deemed suitable for internal flight within a lab. Initial fears that the down wash from the blades would cause interference were proven to be unfounded. Furthermore the sensor is capable of detecting the ground at roll or pitch angles of up to 45 degrees. This sensor is triggered by a simple 555 timer circuit and operates at 20Hz. The output from this sensor is a Pulse Width Modulated (PWM) signal which has been calibrated in house (accuracy: $\pm 1cm$). No additional filtering was required for this sensor. If triggered the ranger will output 8 bursts of ultrasound (40kHz) and raise its echo line high until an echo is received. It therefore outputs a pulse width signal proportional to the height of the vehicle and is able to measure distances of 3cm up to 5m.

All of the sensors installed are analogue sensors. In order to communicate with a computer ground station it is obviously necessary to convert to a digital signal. To convert these signals, PIC microcontrollers with 12bit resolution are used. PICAXE controllers are programmed in BASIC and are very low cost and easy to use. One PIC chip is used to convert the accelerometer and gyroscopic data. A pulse width measure from the ultrasonic is performed by a second microcontroller. All data is converted into a TTL level serial data stream at a maximum baud rate of 19600Baud and an update rate of 20Hz which is limited by the ultrasonic sensor. All timing is done using the timer on the ultrasonic, once the PIC chip receives a signal from the ultrasonic it sends the digital signal to the TTL level shifter and then triggers the second chip which immediately sends the accelerometer and gyroscopic data. The onboard instrumentation setup can be seen in Figure 7.2. The total cost of all additional hardware installed on the quadrotor amounts to about 300GBP. This whole system is small, lightweight and consumes around 2Watts of power and is supplied by a separate lithium polymer battery.

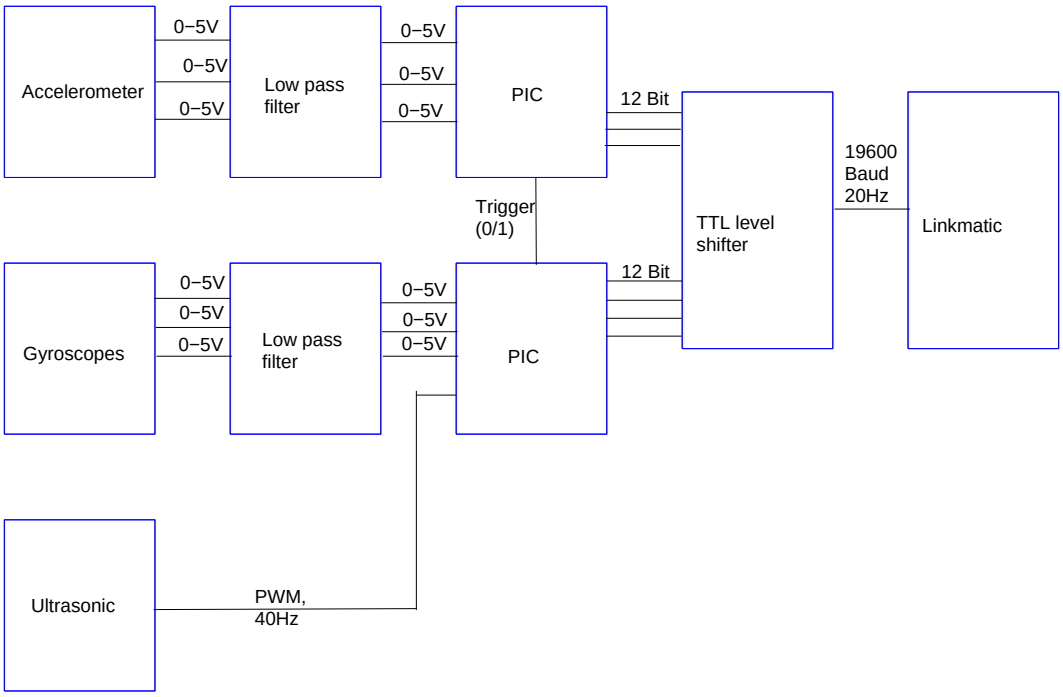


Figure 7.2: Onboard instrumentation

A major issue with any sensor is that of noise. On the quadrotor this is especially an issue due to the large currents supplying the rotors as these large currents induce significant electromagnetic interference. As the rotors are variable speed, the noise is of variable frequency which makes filtering a non-trivial problem. Low-pass filters consisting of basic resistor capacitor circuits are installed, these filter out all high frequency noise. A necessary extension to this work would need to consider the removal of the electromagnetic noise at low frequencies, however, reducing the cut off frequency would be likely to result in the loss of the vehicles dynamic data. Notch filters could be applied to remove all noise outside of a given frequency but in all likelihood there will still be a rotor speed which introduces noise at the same frequency as the vehicles dynamics. Filtering could be performed on the groundstation to remove some of the noise however, there are possible simpler solutions including repositioning the sensors, shield the sensors from the EM noise and screening of the motor power cables. This requires further investigation.

7.2.2 Groundstation

Successful control of the vehicle may be achieved by closing the loop either within the vehicle or remotely via a ground station. An onboard control system holds the advantage of stability as well as small delay times and is undoubtedly the preferred solution for a commercial vehicle. For research purposes though, a high degree of flexibility is needed with algorithms being constantly improved and updated. To ensure fast turnaround times a control system on a ground station provides a number of advantages. These include ease

of access, reduced payload, cost, the ability to rapidly deploy updated code and room for future expansion.

The ground station itself consists of generic personal computing hardware used as a hardware target to run a real time operating system. It is networked to a standard PC from which code can be updated and deployed instantly. The standard PC runs a LABVIEW interface, which, is capable of reading serial data and plotting the sensor data in real time. It is also possible to deploy a control algorithm within LABVIEW and output the required control action through the serial port. The base station also has a joystick which enables open loop control of the vehicle which has been found to be an easier method of flying than the radio control.

7.2.3 Downlink

The difficulty introduced by the concept of a ground station is to establish a deterministic communication link with manageable transmission delays. There are many ways of establishing a communication link between a UAV and a groundstation. For a given power limit however, the trade off is typically between bandwidth and range. In an experimental set up, the maximum range is unlikely to exceed 50m. It is also assumed that bandwidth is a major factor with a requirement for as much state information to be transferred to the groundstation as possible. Furthermore any device used on board the vehicle must be small, lightweight and low cost.

A standard 2.4GHz Bluetooth transceiver offers high transfer rates as well as being small, lightweight and low cost. Another major benefit of the Bluetooth transceiver is the serial interface allowing easy connectivity with the PIC controllers. Sensor data is streamed to the ground station with a transmission latency of 50ms ($\pm 5ms$)). Furthermore the range of the Bluetooth transceiver is up to approximately 100m, comfortably exceeding the likely maximum. At the time of writing a new Bluetooth transceiver came onto the market, this new transceiver is expected to reduce the latency to 15ms. Sensor data is read by a 2.4GHz Bluetooth transceiver directly via a RS232 port.

7.2.4 Uplink

The uplink is via a 4 channel radio control, the 4 channels being total thrust, roll control, pitch control and yaw control. These 4 control actions are determined on the PC either via the joystick or control law. For safety reasons it is necessary to keep a pilot in the loop, so that the control law can be switched off at any time and the safety pilot can regain control. Control via a PC is one option but in the event of a computer crash, control by a pilot or autopilot would be lost. Instead an RC transmitter used for model aircraft control, is used to fly the vehicle as normal, this is more reliable than a PC and is the preferred solution for many UAV experimental set ups (Valenti *et al.* 2006). In order to activate the control law the PC is connected into the buddy port at the back of the transmitter and can be given control of the vehicle by the RC transmitter.

The buddy port of the transmitter requires a PWM signal. PWM generated from the groundstation is possible but was found to be too slow for practical implementation. Instead a PWM converter box has been produced consisting of two PIC chips. The first chip reads the RS232 signal from the computer, the second PIC chip then converts this signal to a 4 channel PWM signal. This box then connects to the back of the transmitter. An FM receiver is then placed on the vehicle.

7.3 Open loop Results

An open loop flight test has been performed in order to test the uplink and feedback. This involved a flight using the uplink and downlink described within this chapter. The normalised thrust going into the quadrotor against height from the ultrasonic sensor can be seen in Figure 7.3. There is a non-linear relationship between the thrust and the altitude which can be explained by the presence of ground effect.

Although not considered within this work, ground effect is an inevitable disturbance which the controller must overcome. However, without a detailed model of this effect, it is impossible to determine the suitability of the controller to cope with this. A test stand has been constructed with a large wire grid surface. This surface holds the vehicle well above the ground (1.5m) but allows the down wash from the props to flow through the surface. This test stand will enable control development outside of ground effects.

The control inputs for this flight were also recorded and shown in Figure 7.4. This figure gives some indication of the high pilot workload required to fly the quadrotor which explains why it is so hard to manually pilot. Pilot Induced Oscillations (PIO) are a common problem when flying the quadrotor, the oscillatory demands which can lead to this, can clearly be seen in Figure 7.4.

Data from the gyroscopes and accelerometers, was also recorded during this flight. As the supply current to the motors is large however, large electro-magnetic forces are present and therefore significant noise is present within the data. As the rotors are of variable speed the electro-magnetic noise is of variable frequency and therefore filtering the noise out without removing the dynamic data is not a trivial problem. This has been reduced by adding low pass filters to the vehicle to remove noise at the frequency of the motors but this is something that will need modification before autonomous flight can be achieved.

7.4 Discussion

This chapter has presented some open loop test flight results from the Cranfield experimental set up of the Draganflyer X-Pro. This data provides accelerations, gyroscopic rates and height measurement. Before trajectory planning techniques, such as the ones discussed within this thesis, can be implemented, position feedback of the vehicle is required. For a vehicle such as the quadrotor, which is intended for internal flight, this

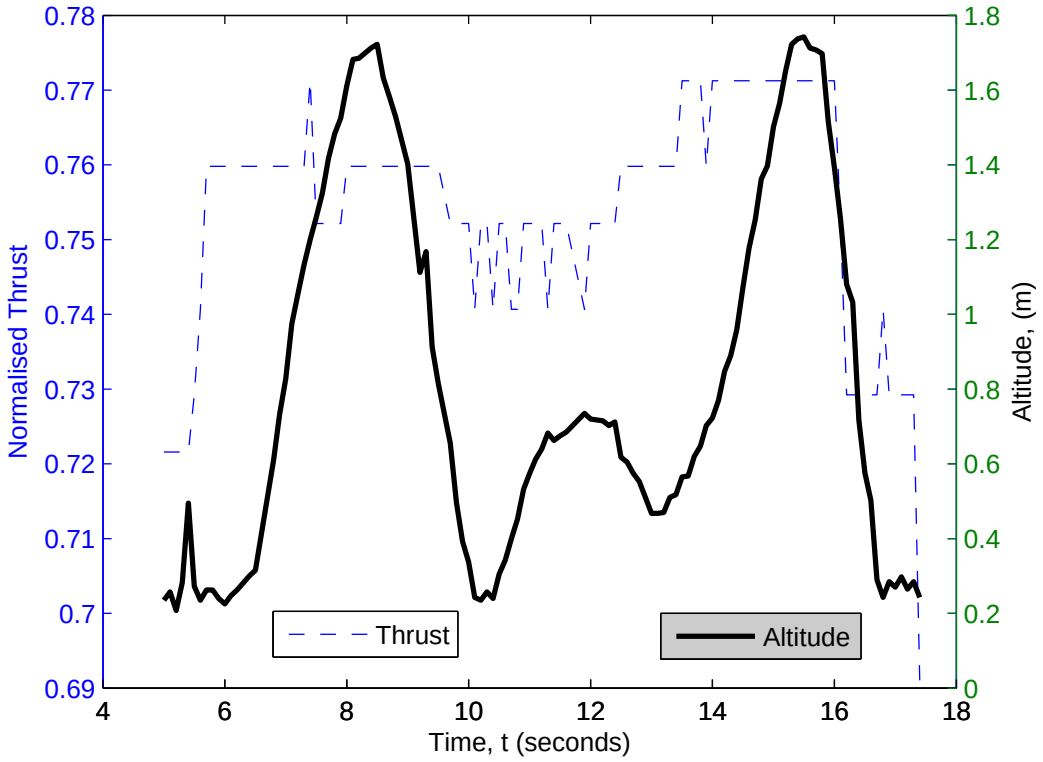


Figure 7.3: Experimental thrust against measured height

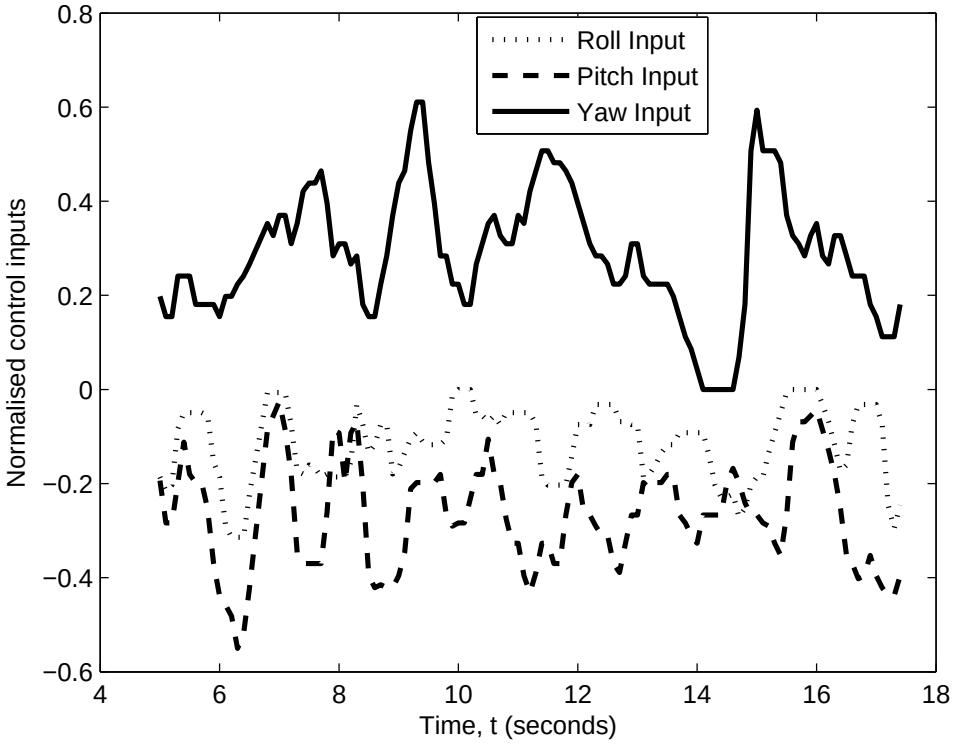


Figure 7.4: Experimental normalized control inputs

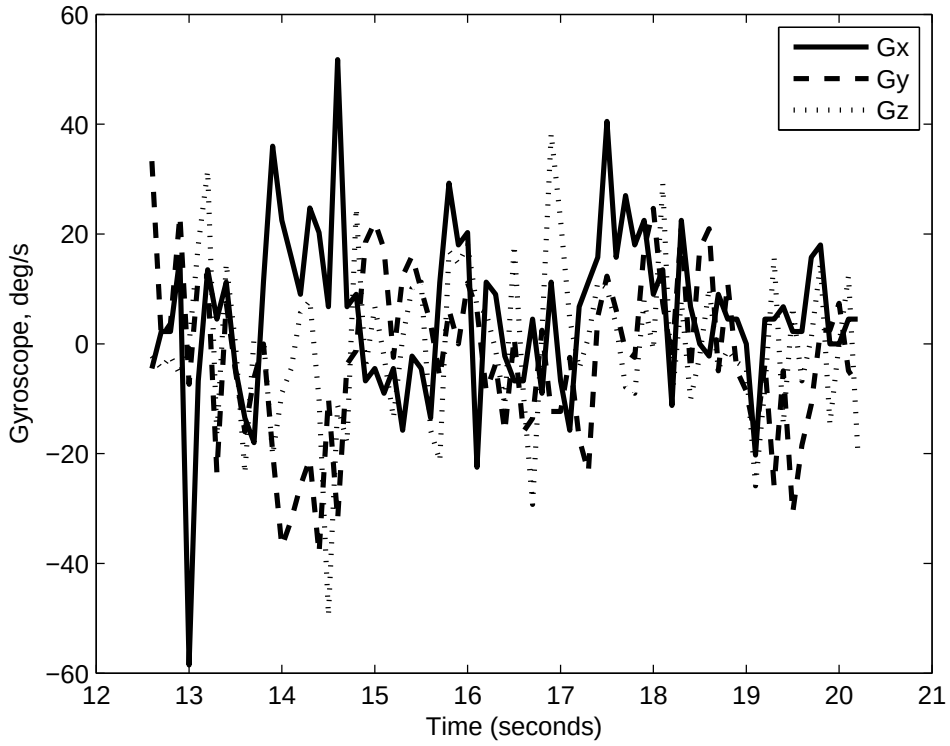


Figure 7.5: Experimental gyroscopic readings

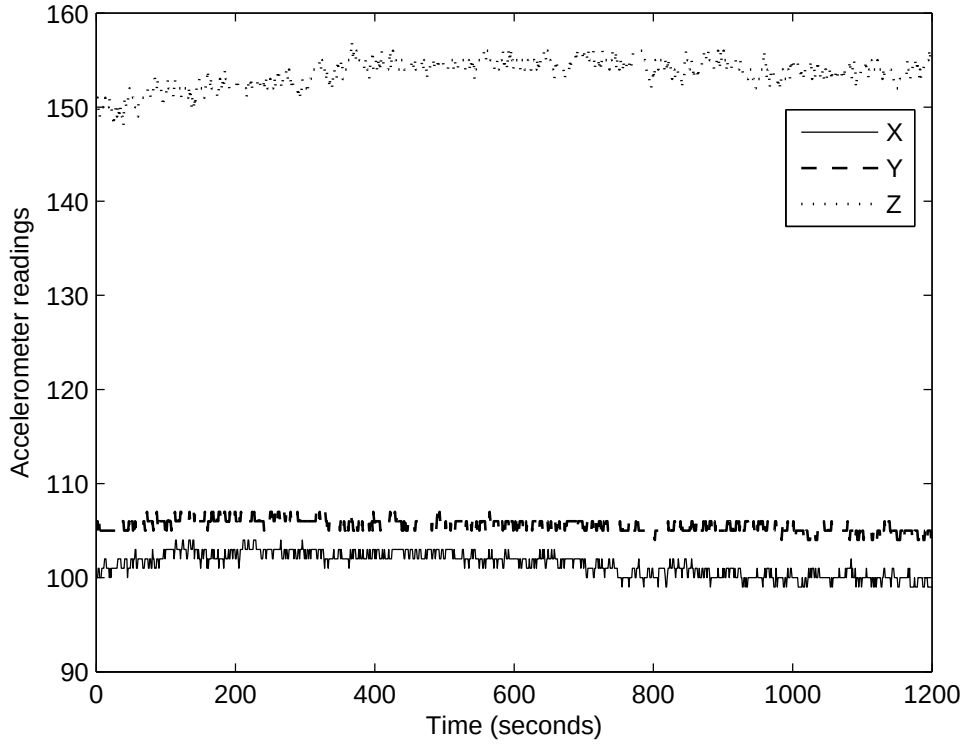


Figure 7.6: Experimental accelerometer readings

problem is made harder by the fact horizontal displacement can not be measured by GPS.

There are a few options for position feedback for such a vehicle: a indoor camera system such as the Viacon system can be installed which gives accurate state feedback for all vehicles within the room to a high degree of accuracy, unfortunately the system is very expensive costing typically $100 000 and this does not solve the problem of how to obtain this information in practice. An alternative portable solution is to attach on board cameras on to the vehicle and use visual recognition software to determine position (Altug *et al.* 2002), this however is by no means a trivial task and could warrant an independent thesis alone, so is not investigated within this work. A simpler solution in a room of known geometry as discussed in (Park *et al.* 2005), is to use side mounted ultrasonic sensors to measure the distance to the walls, however, this would only be feasible in an empty room with regular walls.

The practical results presented in this chapter demonstrate the feasibility of such an experimental set up. Control of the vehicle is achieved through a PC, albeit under manual control, thus validating the uplink control. Data is also streamed in real time to the ground station with minimal latency and therfore closing the loop. Potentially of greater importance this work has highlighted some key challenges which need to be considered before autonomy of the quadrotor can be achieved. Electromagnetic noise from the motors is an issue which requires consideration before the sensor data can be used for determining the control action. This work also demonstrates the non-linear behavior within ground effect. Without extensive modelling this is a very difficult thing to consider in control law design,

however these results show it is significant and therefore will need consideration before flight testing. It is also theoretically possible with the current set up to use flight test data to validate the full dynamic model used in this work. This chapter therefore despite not demonstrating closed loop control of the quadrotor, does highlight some key issues with achieving autonomy of the quadrotor. Furthermore, whilst not demonstrating autonomy it would not be complete without considering these challenges to achieve this. In doing so, it is now a reasonable claim that this work truly considers the problem of autonomy and makes considerable strides to achieving it in a low cost experimental set up.

7.5 Multiple vehicle implementation considerations

This work considers the challenges of implementing the control laws developed in this work on a single quadrotor. In the previous chapter the problem of multiple quadrotor control was considered. It is therefore worth considering the suitability of the experimental set up described in this chapter for the control of multiple vehicles.

The first issue, is the scalability of the current set up, although the states are measured onboard the vehicle the control action is determined on the ground station. This will obviously limit the number of vehicles that can be controlled from a single PC as the computation time will grow with each vehicle controlled by a PC. Multiple ground stations are a possible solution but it is more likely that the vehicles would need to determine the control action onboard the vehicle. For safety reasons however, it is likely that a pilot would need to remain in the loop. For a small number of quadrotors however, it is feasible that the current set up is sufficient and would be very cost effective.

Another issue is that of collision avoidance. The previous chapter considered the multiple vehicle rendezvous problem and proposed a control scheme which was dependent on a visual or similar system, to rendezvous the vehicles as well as to avoid collisions. If the control laws are decentralized and the control actions are computed onboard then it is necessary to also determine the position of other vehicles onboard or through a telemetry link. However, in the current set up, with the addition of positional measurement, the ground station could prevent collisions through direct control of each vehicle.

In conclusion while the experimental set up described in this work is not a long term industrial solution, there are a some merits when considering the problem of multiple vehicles. Firstly as the number of vehicles increases the instrumentation cost per vehicle remains low as the computational power is in the ground station. Secondly, when considering collision avoidance the addition of extra sensors is not necessary, as the vehicles can be kept apart by the ground station which has a position estimate for each vehicle.

Chapter 8

Conclusions

8.1 Quadrotor

As UAV's are reaching an unprecidented level of sophistication, there is a growing interest in hovering vehicles. The quadrotor is one possible solution for this, its dynamic simplicity results in low manufacturing costs as well as being an ideal test-bed for advanced control algorithms.

In Chapter 2, the quadrotors' dynamics are considered. The six degree of freedom, equations of motion for the vehicle are presented. This work simplifies these equations by using a different rotational matrix as opposed to the rotational matrix traditionally used by aerospace engineers. This simplification is of great benefit when manipulating these equations to evaluate the differentially flat property.

As well as these simplified equations, a full dynamic model has been developed at Cranfield University for the testing of the control laws. This model is based on experimental data as well as theoretical analysis.

8.2 Trajectory optimization

In order to achieve autonomy of a UAV several issues need to be considered, these include trajectory generation and trajectory following. Trajectory generation is essentially the determination of a feasible, time dependent, three dimensional trajectory to reach the destination. In order to determine the optimal trajectory for the three missions defined (3.1), it is necessary to solve a non-linear constrained optimization, typically this would be solved within the control space. Differential flatness is a property of some dynamical systems which essentially means the dynamics can be inverted to repose such an optimization problem within the output space. In Section 3.3 the dynamic flat equations of the quadrotor are shown enabling such a formulation.

To determine a feasible trajectory the optimization must be constrained, by approximating the control inputs as a function of the actuator outputs suitable control approximations can be made to avoid control saturation. Initial and terminal constraints are also required to ensure the vehicle reaches its destination at some predetermined flight time. Singularities are present within the differentially flat equations when the vehicle pitches or rolls through 90 degrees, therefore the trajectory is constrained within this range. Finally obstacles and buildings are constrained using a convex approximation. Within this work, it is assumed that there exists a feasible trajectory to reach the destination although that in reality is not always possible.

In order the reduce the order of the optimization to a finite amount, a suitable output space parameterization is required. The simplest form of output space parameterization is the simple polynomial but the conditioning of this is poor. Other techniques such as Laguerre polynomials and Chebyshev polynomials are compared and it is found that for this type of problem, Laguerre polynomials provide the fastest convergence properties.

Using Laguerre polynomials and the differentially flat equations, trajectory optimizations are posed within the output space and shown for the 3 missions (3.7). The computation time for these optimization is typically a couple of seconds on a standard desktop PC. There are however a number of disadvantages to this approach. The initial and terminal constraints are determined numerically, which can lead to long computational time to be solved exactly. To remove this problem a bounding box is used to relax the terminal constraints and hence reduce computation time, enabling a feasible trajectory to be found. Also in this formulation the flight time must be predetermined and can not be optimized, preventing the solution to the minimal time problem.

8.3 Control system

Once a reference trajectory has been found, the reference states and controls, can be fed forward. For stability and accurate tracking however, closed loop control is also needed. Two different control schemes are initially considered in this work, the first a non-linear advanced control technique (MBPC) and the second a standard multivariable controller (LQR).

8.3.1 MBPC

Model Based Predictive Control is an advanced control scheme, which essentially consists of repeated optimizations, over a given time horizon, to determine the current control input. By repeatedly solving the non-linear constrained optimization proposed in Chapter 3 a non-linear MBPC problem is formulated. At each time step the initial constraints are set to the vehicles current state and hence closing the loop.

Typically the time horizon receeds in MBPC as time progresses, hence the alternative name of receeding horizon control. In this work however, the whole trajectory until the

final predetermined time is considered and therefore the time horizon reduces with time.

MBPC is used as a combined trajectory generation and trajectory following algorithm. The three missions shown demonstrate the suitability of MBPC for UAV control. The major advantage of this approach is the ability to satisfy state constraints especially in the event of an environmental change, this is shown by the reoptimization of a trajectory to account for a strong wind. A disadvantage of this approach however, is the computational time required to solve the nonlinear optimization problem. Furthermore there is no guarantee when considering nonlinear optimization that the algorithm will converge on a feasible solution.

8.3.2 LQR

Once a reference trajectory is found, if this remains feasible for the duration of the flight there is no need to reoptimize. Instead a standard multivariable controller (LQR) can be applied to track this reference trajectory. The computational time of trajectory following is negligible and it can be easily implemented. The disadvantage of this scheme is that constraint satisfaction is not guaranteed after the first time step. Therefore in the event of environmental changes such as an obstacle moving, the trajectory may become infeasible resulting in a collision or large tracking errors.

8.3.3 Combined control

As MBPC is computationally demanding and simple trajectory following does not guarantee constraint satisfaction, an alternative solution is required. A combined controller is proposed which follows the reference trajectory when it is feasible to do so, but reoptimizes if it not feasible. An update switch monitors the reference trajectory and the vehicle's drift from this trajectory and switches between the inner loop trajectory follower and outer loop trajectory generation. The resulting controller is capable of near to real time simulation on a standard desktop PC as well as 'reacting' to environmental changes as shown in Section 4.3.2.

8.3.4 Automatic Differentiation

In this work the optimization is programmed into a gradient based routine (*fmincon* in MATLAB.) Evaluation of the optimization routine shows that upto 70% of the computation time is used in the evaluation of the constraints and the gradients of these constraints. Significant reductions in computation time can therefore be theoretically achieved by supplying the gradients to this routine but determining these analytically by hand is a time consuming process. Automatic Differentiation (AD) is a computational technique which utilizes the step by step nature of the computer programme and the chain rule to evaluate the gradients of a complex function. AD can be implemented easily into MATLAB using

MATLAB Automatic Differentiation (MAD) (Forth 2006). Unfortunately in this example there is no saving in computational time using MAD as the overheads of the software seem to negate the benefits of the technique. Future work in this area would look to seek alternative software packages such as ADOL-C as used in (Cao 2005), where significant computational time reductions are shown.

8.4 Taranenko's direct method

Output space trajectory optimization through dynamic inversion was heavily researched during the 1990's (Martin *et al.* 1994, Koo and Sastry 1999, Driessen and Robin 2004, Defoort *et al.* 2007, Chelouah 1997) following the work on differential flatness proposed by Fleiss (Fleiss *et al.* 1992). Output space optimization via dynamic inversion was not a new idea however, in the 1960's Taranenko was also looking at direct methods in the output space to solve optimal control problems. A significant difference in this work, is the use of the virtual argument, which greatly improves the convergence properties of the scheme. Unfortunately the majority of publications on this approach are in Russian and therefore lesser known.

By introducing a virtual argument the optimal path and speed problems are separated, allowing for example, easy variation of the speed profile along a trajectory. Another benefit of this technique is the analytical determination of the boundary conditions, which were previously solved numerically within the routine presented in Chapter 3. As the boundary conditions are met analytically, the initial guess of the remaining free variables are less crucial, and often provide a feasible solution, unlike the Laguerre based scheme. The virtual argument also enables the solving of the optimal time problem as opposed to arriving at a predetermined arrival time. The big difference observed from a programming point of view is a much reduced dependency on the optimization settings, such as the initial values for the free variables.

When considering fixed wing vehicles using Taranenko's direct method, there is no need to consider zero velocity. The quadrotor however, is capable of zero velocity during hover. This creates a problem because the speed factor (λ) becomes undefined (zero divided by zero) when the velocity is zero. The work in this thesis assumes that zero velocity can occur only at the boundaries. The assumption is made that the speed factor is zero at the boundary and this worked satisfactorily. This is because the approximations that are used means that the sigularities do not occur in the calculations. Although this approach does not allow for the possibility of zero velocity during a flight this was not found to be a limitation for the case cases studied. The question of how best to deal with zero velocity remains for future work. In (Whidborne *et al.* 2008), an iterative scheme is proposed for defining the boundary conditions of the speed factor, this requires further testing and has not been covered in this thesis.

This work assumes that this zero velocity only occurs at the first and last time steps therefore, allowing for the speed factor to be equal to zero at the boundary conditions. Although this does not account for the possibility of the zero velocity during the flight this

was not found to be a problem. This case does require further consideration and especially the case where the initial or terminal velocity are zero, as in-flight zero velocity is less likely.

This work therefore, through the use of the differentially flat equations applies Taranenko's direct method to the quadrotor successfully. This technique was found to be a significant improvement on the algorithm presented in Chapter 3. From these results, two different conclusions can be drawn. The first is that Taranenko's direct method is a viable trajectory generation algorithm for control of the quadrotor. However, Taranenko's method can be applied to any UAV which posses differentially flat dynamics. This scheme has a number of benefits and it envisaged that this scheme is equally applicable for control of other rotary and fixed wing UAVs.

While a scheme of this complexity may not be deployed in standard fixed wing flight, where a classical control structure following waypoints may be equally suitable, it may be deployed for maneuvers which require the states to be constrained or optimized. One example state optimization is time critical problems, where for example a vehicle is required to reach a destination at a predetermined or optimal time. Another example of state optimization is the velocity optimization problem considered in this work where a vehicle passes over an obstacle with reduced velocity. An example of state constraints is also shown in this work where the vehicle is required to avoid obstacles, drop down a mineshaft and satisfy dynamic and control constraints. For fixed wing flight, similar constraints may be encountered when considering for example automated landing, sense and avoid problems, formation flight and multiple vehicle rendezvous.

8.5 Multiple vehicles

When considering small UAVs such as the quadrotor there are a number of advantages to deploying multiple UAVs. This work considers a decentralized rendezvous problem. In Ando *et al.* (1999), a two dimensional point convergence algorithm is presented which guarantees convergence of multiple vehicles with no communication link and only limited visibility. In Chapter 6 this scheme is demonstrated for the 3D case with 5 quadrotors. Furthermore this work considers the conflicting problem of collision avoidance and with the application, of a 'direct consequence measurement,' collision avoidance and rendezvous are demonstrated.

The results show that despite the collision avoidance considerations within the controller the vehicles rendezvous successfully. For the straight line scenario the convergence is slower than for the analytical solution but rendezvous is not in doubt. This work also provides another example of a potential application for a trajectory optimization scheme such as Taranenko's. In this example, the analytical determination of the boundary conditions is especially beneficial as well as the extra state constraints imposed by the collision avoidance algorithms.

8.6 System design

To achieve autonomy there are many issues to consider including trajectory generation and following, which have been considered within this work. Other important issues include that of feedback and the implementation of the control law. In Chapter 7 the experimental set up at Cranfield University is discussed with some open loop experimental data. It is shown that it is feasible to use a groundstation for determining the control action, as a Bluetooth transceiver is used to stream data with a reasonably small latency. Lightweight MEM's sensors are used to measure linear acceleration along the 3 axes, steady state gyroscopes are used for attitude feedback and an ultrasonic sensor is used for altitude. Successful open loop flight results are shown which show the feasibility of such a system as well as highlighting the difficulties when considering instrumentation of a small UAV such as the quadrotor. These difficulties include the position determination of an internally flying vehicle where GPS is not available. Another issue is the filtering of the vehicles sensor data where variable frequency electromagnetic noise is present. While this work does not go as far as demonstrating autonomy of a quadrotor, by considering the implementation of the control laws it is a reasonable claim that the full problem of autonomy is considered.

8.7 Future work

Suggested future work following the work presented within this thesis can be split into two key areas, that of trajectory generation and implementation. While the trajectory generation issues consist of potential applications of the schemes discussed for other vehicles, the implementation side involves the demonstration of the techniques discussed onto the quadrotor itself.

8.7.1 Trajectory optimization

Future improvements of the control algorithm presented could include focus on the reduction of computation time. In this work AD proved unsuccessful, but there are examples of significant computational time reductions using AD (Cao 2005). Using a package such as ADOL-C and the MEX wrap in MATLAB could potentially significantly reduce computation time.

Taranenko's direct method is easily extendable to other vehicles. Consider the case of the fixed wing rendezvous problem, this is different to the problem discussed previously, as there is a constraint on the minimum velocity. The direct method allows for the velocity profile to be varied along a given trajectory which simplifies this problem. As the boundary conditions are met analytically, and the states are explicit functions of these boundary conditions, then it is easily extendable to the case of a critical terminal state, such as landing on a slanted roof.

8.7.2 Implementation

A major issue encountered with the instrumentation is the electromagnetic noise within the sensor data. The 4 rotors are variable speed rotors and therefore the noise is of variable frequency making a simple, low-pass filter inadequate. Future work will be required to remove this noise from the sensor data without the removal of the vehicles dynamic data. Removing this noise from the signal will provide a good estimate for body rates $([p,q,r])$ and accelerations $[\ddot{x},\ddot{y},\ddot{z}]$. Other state parameters such as Euler angles can potentially be obtained from this information.

Before trajectory tracking can be demonstrated it is first necessary to determine the vehicles position. As the vehicle is intended for internal flight, GPS is typically not viable. Currently the only options on the market are very expensive such as the ground based camera system used by Valenti *et al.* (2006) and an indoor GPS system (Naviva 2008). This provides very accurate state feedback but is expensive and does not solve the problem for a transportable UAV. There is therefore substantial future work required to obtain position feedback onboard potentially via visual feedback from onboard cameras.

Therefore to validate the work presented in this thesis, the following tasks are required:

- Shielding or filtering of electromagnetic noise
- Positional feedback through additional sensors
- Attitude determination from onboard sensors through a suitable filter
- Deployment of control law on RTOS

References

Active-Robots (2008). *http://www.active-robots.com/products/sensors/sensors-devantech.shtml*.

Al Seyab, R. K. S. (2006). Nonlinear model predictive control using automatic differentiation. PhD thesis. Cranfield University.

Alekhin, D. V. and O. A. Yakimenko (1999). Synthesis of optimization algorithm for planned-route flight trajectory by direct variational method. *Journal of Computer and Systems Sciences International* **38**, 650–666.

Altug, E. and C. Taylor (2004). Vision-based pose estimation and control of a model helicopter. In: *Proceeding of IEEE Mechatronics*. Istanbul, Turkey.

Altug, E., J.P. Ostrowski and R. Mahony (2002). Control of a quadrotor helicopter using visual feedback. In: *Proc. IEEE Conf. Robo. Auto.*. Washington, DC.

Analog-Devices (2008). *www.analog.com*.

Ando, H., Y. Oasa and I. Suzuki (1999). Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Trans. Robotics. Automation* **15**, 818–828.

Beji, L. and A. Abichou (2005). Streamlined rotors mini rotorcraft: Trajectory generation and tracking. *International Journal of Control, Automation and systems* **3**(1), 87–99.

Betts, J. T. (2001). *Practical Methods for Optimal Control Using Nonlinear Programming (Advances in Design and Control)*. Society for Industrial Mathematics.

Betts, J.T. (1998). Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics* **21**(2), 193–207.

Blue-Bear-Systems-Research (2008). *http://www.bluebearsystems.com/Default.asp?page=374*.

Boeing (2008). Sparse optimal control software. The Boeing Company.

Bouabdallah, A., A. Noth and R. Siegwart (2004*a*). PID vs LQ control techniques applied to an indoor micro quadrotor. In: *Proc. IEEE Conf. on Intel. Robot. Systems*. Sendai, Japan.

Bouabdallah, A. and R. Siegwart (2005). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. In: *IEEE International Conference on Robotics and Automation*. Barcelona, Spain,.

Bouabdallah, A., P. Murrieri and R. Siegwart (2004*b*). Design and control of an indoor micro quadrotor. In: *Proceedings. IEEE International Conference on Robotics and Automation*. pp. 4393– 4398.

Canuto, C., M.Y. Hussaini and A. Quarteroni (1988). *Spectral Methods in Fluid Dynamics*. Springer Verlag. New York.

Cao, Y. (2005). A formulation of nonlinear model predictive control using automatic differentiation. *Journal of Process Control* **15**, 851–858.

Carnduff, S., S.D. Erbsloeh, M.V. Cook and A.K. Cooke (2007). Experimental investigation into UAV flight dynamics. In: *Proc 22nd International gUAV Systems Conf*.

Castillo, P., A. Dzul and R. Lozano (2004). Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Trans. Control Syst. Tech.* **Vol.12 NO.4**, 510– 516.

Castillo, P., P. Albertos, P. Garcia and R. Lozano (2006). Simple real-time attitude stabilization of a quad rotor aircraft with bounded signals. In: *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA, USA. pp. 1533–1538.

Castillo, P., R. Lozano and A.E. Dzul (2005). *Modelling and Control of Mini-Flying Machines*. Springer.

Chelouah, A. (1997). Extensions of differential flat fields and Liouvillian systems. In: *Proc. 36th IEEE Conf. Decision Contr.*. San Diego, Caliornia.

Chen, M. and M. Huzmezan (2003*a*). A combined MBPC/2 dof H_∞ controller for a quad rotor UAV. In: *Proc. AIAA. Atmos. Flight. Mechanics. Conf (afmc 2003)*. Austin, Texas.

Chen, M. and M. Huzmezan (2003*b*). A simulation model and H_∞ loop sharing control of a quadrotor unmanned air vehicle. In: *IASTED International Conference on Modelling and Simulation*. Palm Springs, California.

Chitrakaran, V.K., D.M. Dawson, H. Kannan and M. Feemster (2006). Vision assited autonomous path following for unmanned aerial vehicles. In: *45th IEEE Conference on Decision and Control*. San Diego, CA. pp. 63–68.

Conway, B. A. and K. M. Larson (1998). Collocation versus differential inclusion in direct optimization. *Journal of Guidance, Control, and Dynamics* **21**, 780–785.

Cook, M.V. (1997). *Flight Dynamics Principles*. Butterworth Heinemann, Oxford, England.

Cooke, A.K. (2002). *Helicopter test and evaluation*. Blackwell and QinetiQ, Oxford, England.

Cranfield-Aerospace (2008). *http://www.cranfieldaerospace.com/miniuav.aspx*.

De Nardi, R., O. Holland, J. Woods and A. Clark (2006). Swarmav: A swarm of miniture aerial vehicles. In: *Proc 21st International UAV Systems Conf*.

Defoort, M., T. Floquet, A. Kokosy and W. Perruquetti (2007). Decentralized robust control for multi-vehicle navigation. In: *Proceedings of the European Control Conference*. Kos, Greece. pp. 2150–2157.

Dobrokhodov, V.N. and O.A. Yakimenko (1999). Synthesis of trajectorial control algorithms at the stage of rendezvous of an airplane with a maneuvering object. *Journal of Computer and Systems Sciences International*, **38**(2), 262–277.

DoD (2005). Unmanned air systems roadmap 2005 - 2030. Technical report. Office of the secretary of defense. USA.

Dodd, T. (2007). Development of a mini uninhabited air vehicle for urban environments. In: *IET Seminar on Micro UAV's*. Austin Court, Birmingham, UK.

Driessen, B.J. and A.L. Robin (2004). A globally convergent tracking controller for the X4 flyer rotor craft for reference trajectories with positive thrust. In: *Robotica Part 4*. pp. 375–388.

Eikenberry, B., O. Yakimenko and M. Romano (2006). A vision based navigation among multiple flocking robots: Modeling and simulation,. In: *AIAA Modeling and Simulation Technologies Conference*. Keystone, CO.

Elnagar, J., M. A. Kazemi and M. Razzaghi (1995). The pseudospectral legendre method for discretizing optimal control problems. *IEEE Transactions on Automatic Control* **40**, 17931796.

Enright, P. J. and B. A. Conway (1992). Discrete approximations to optimal trajectories using direct transcription and nonlinear programming. *Journal of Guidance, Control, and Dynamics*, **15**, 994–1002.

Fahroo, F. and I. M. Ross (2001). A second look at approximating differential inclusions. *Journal of Guidance, Control, and Dynamics* **24**, 131133.

Fahroo, F. and I. M. Ross (2002). A direct method for solving nonsmooth optimal control problems. In: *IFAC, 15th Triennial World Congress,*. Barcelona, Spain.

Fahroo, F. and I. M. Ross (2004). Pseudospectral knotting methods for solving nonsmooth optimal control problems. *Journal of Guidance, Control, and Dynamics* **27**, 397–405.

Fahroo, F. and I.M. Ross (2000). Direct trajectory optimization by a chebyshev pseudospectral method. In: *Proc. 2000 Amer. Contr. Conf.*. Chicago, Illinois.

Fleiss, M., J. Levine, Ph. Martin and P. Rouchon (1992). Sur les systemes non linerities differentiellement plats. *C.R.Acad Sci*.

Forth, S.A. (2006). An efficient overloaded implementation of forward mode automatic differentiation in matlab. *ACM Transactions on Mathematical Software* **32**(2), 195 – 222.

Gartner, B. (2008). Fast and robust smallest enclosing balls. www.inf.ethz.ch/personal/gaertner/miniball.html.

Gibbs, D.G. (2005). The predator in operation iraqi freedom – a pilot's perspective. In: *Infotech@Aerospace, AIAA*.

Griewank, A. (1992). Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Opt. Methods Softw.* **1**, 3554.

Griewank, A., D. Juedes and J. Utke (1996). Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software* **22**(2), 131 – 167.

Gu, D.W., W. Kamal and I. Postlethwaite (2004). A UAV waypoint generator. In: *Proc. AIAA 1st. Intelligent Sys. Tech Conf.*

Hamel, T., R. Mahony and A. Chriette (2002). Visual servo trajectory tracking for a four rotor VTOL aerial vehicle. In: *IEEE International Conference on Robotics and Automation*. Washington, DC. pp. 2781–2786.

Hargraves, C. R. and S. W. Paris (1987). Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, **10**(4), 338–342.

Harris, G. (2000). The economics of landmine clearance: Case study of cambodia. *Journal of International Development* **12**, 219–225.

Herman, A. L. and B. A. Conway (1996). Direct optimization using collocation based on high-order gauss-lobatto quadrature rules. *Journal of Guidance, Control, and Dynamics*, **19**(3), 592–599.

Hoffmann, G., D.G. Rajnarayan, S.L. Waslander, D. Dostal, J.S. Jang and C.J. Tomlin (2004). The stanford testbed of autonmous rotorcraft for multi agent control (STARMAC). In: *Proc. 23rd Digital Avionics systems conf.*. Salt Lake City, Utah.

http://www.mathworks.co.uk/ (n.d.). *MATLAB*.

Hull, D. G. (1997). Conversion of optimal control problems into parameter optimization problems. *Journal of Guidance, Control, and Dynamics* **20**, 57–60.

Huzmezan, M., G.A. Dumont, W.A. Gough and S. Kovac (2001). Multivariable Laguerre-based indirect adaptive predictive control a reliable practical solution for process control. In: *IASTED Modelling and Control*. Innsbruck, Austria.

Ieropoulos, I., C. Melhuish and J. Greenman (2004). Energetically autonomous robots. In: *Proceedings of the 8th Intelligent Autonomous Systems Conference*. Amsterdam, The Netherlands. pp. 128–35.

Kamal, W.A., D.-W. Gu and I. Postlethwaite (2005). A decentralized probabilistic framework for the path planning of autonomous vehicles. In: *Proc. 16th IFAC World Congress*. Prague, Czech Republic.

Kaminer, I.I., Yakimenko. O.A. and A.M. Pascoal (2006). Coordinated control of multiple UAVs for time-critical applications. In: *Proceedings of the 27th IEEE Aerospace Conference*. Big Sky, Montana,.

Kavaraki, L.E., P. Svestka, J.C. Latombe and M.H. Overmars (1996). Probabilistic roadmaps for path planning in high- dimesional configuration spaces. *IEEE Trans. Robotics. Automation.* **12**, 566–580.

Kendoul, F., D. Lara, I. Fantoni and R. Lozano (2006). Nonlinear control for systems with bounded inputs: Real time embedded control applied to UAVs.. In: *Proceedings of the 45th IEEE Conference on Decision and Control*. San Diego, CA, USA. pp. 5888–5893.

Kim, Y., D.W. Gu and Postlethwaite (2007). Real-time optimal mission scheduling and flight path selection. *IEEE Transactions on Automatic Control* **52**(6), 1119–1123.

Koo, T.J. and S. Sastry (1999). Differential flatness based full authority helicopter design. In: *Proc. 38th IEEE Conf. Decision Contr.*. Phoenix, Arizona.

Kumar, R. and H. Seywald (1996*a*). Dense-sparse discretization for optimization and real-time guidance. *Journal of Guidance, Control, and Dynamics* **19**, 501–503.

Kumar, R. and H. Seywald (1996*b*). Should controls be eliminated while solving optimal control problems via direct methods?. *Journal of Guidance, Control, and Dynamics* **19**(2), 418–423.

Lane, S. H. and R. F. Stengel (1988). Flight control design using nonlinear inverse dynamics. *Automatica* **24**(4), 471–483.

Lin, Z., B. Francis and M. Maggiore (2006). Getting mobile autonomous robots to rendezvous. In: *Control of Uncertain Systems: Modelling, Approximation and Design.* (B.A. Francis, M.C. Smith and J.C. Willems, Eds.). Springer. pp. 119–137. Workshop on the occasion of Keith Glovers 60th Birthday.

Lu, P. (1993). Inverse dynamics approach to trajectory optimization for an aerospace plane. *Journal of Guidance, Control, and Dynamics* **16**, 726–732.

Maciejowski, J.M. (2002). *Predictive Control with Constraints*. Prentice Hall, Harlow, England.

Madani, T. and A. Benallegue (2006). Control of a quadrotor mini-helicopter via full state backstepping technique. In: *Proc. 45th IEEE Conf. Decision Contr.*. San Diego, CA, USA. pp. 1515–1520.

Martin, P., S. Devasia and B. Paden (1994). A different look at output tracking: control of a VTOL aircraft. In: *Proc. 33rd IEEE Conf. Decision Contr.*. Lake Buena Vista, Florida.

Martinez Martinez, V. (2007). Modelling of the flight dynamics of a quadrotor helicopter. Master's thesis. School of Engineering, Cranfield University,.

Mayne, D.Q., J.B. Rawlings, C.V. Rao and P.O.M. Scokaert (2000). Constrained model predictive control: Stability and optimality. *Automatica* **36**, 789–814.

McKernan, J., G. Papadakis and J.F. Whidborne (2006). A linear state-space representation of plane Poiseuille flow for control design – a tutorial. *Int. J. Model. Ident. and Control* **1**(4), 272–280.

MDL (2008). *http://www.mdl.co.uk/laser*$_m$*odules/laserace − im/index.html*.

MicroPilot (2008). *http://www.micropilot.com/*.

Milam, M.B., K. Mushambi and R.M. Murray (2000). A new computational approach to real time trajectory generation for constrained mechanical systems. In: *Proc. 39th IEEE Conf. Decision Contr.*. Sydney, Australia.

Mokhtari, A., A. Benallegue and Y. Orlov (2006). Exact linearization and sliding mode observer for a quadrotor unmanned aerial vehicle. *International Journal of Robotics and Automation* **21**(1), 39 – 49.

Mokhtari, A. and A. Benallegue (2004). Dynamic feedback controller of euler angles and wind parameters estimation for a quadrotor unmanned air vehicle. In: *IEEE International Conference on Robotics and Automation*. New Orleans, LA. pp. 2359–2366.

Murphy, R. and S. Stover (2006). Gaps analysis for rescue robots,. In: *CRASAR-TR2005-12, 2006*.

Murphy, R., S. Stover and H. Choset (2005). Lessons learned on the uses of unmanned vehicles from the 2004 florida hurricane season,. In: *CRASAR-TR2005-11, AUVSI Unmanned Systems North America*. Baltimore, MD.

Murphy, R.R. Burke, J.L. (2005). Up from the rubble: Lessons learned about HRI from search and rescue. In: *Proceedings of the 49th Annual Meetings of the Human Factors and Ergonomics Society*. Orlando, Florida.

Naviva (2008). *http://www.naviva.fi/navindoor/*.

Newcome, L.R. (2004). *Unmanned Aviation*. AIAA Inc. Virginia.

Nieuwstadt, M.J. and R.M. Murray (1995). Approximate trajectory generation for differentially flat systems with zero dynamics. In: *Proc. 34th IEEE Conf. Decision Contr.*. New Orleans.

O'Donnell, J.E.D. and J.B. Dorwar (2006). Evaluation of medium altitude long endurance unmanned aerial vehicles for U.S coast guard operations. In: *Proc 21st International UAV Systems Conf.*

Park, S., D.H. Won, M.S. Kang, T.J. Kim, H.G. Lee and S.J. Kwon (2005). RIC (robust internal-loop compensator) based flight control of a quad-rotor type UAV. In: *Intelligent Robots and Systems, IEEE/RSJ International Conference*. pp. 3542– 3547.

Pettersson, P.O. and P. Doherty (2004). Probabilistic roadmap based path planning for an autonomous unmanned air vehicle. In: *Proc. Works. Connecting. Planning and theory with Practice. ICAPS*.

Pierre, D.A. (1969). *Optimization theory with applications*. John Wiley and Sons,.

Pontrjagin, L., V. Boltjanskiy, R. Gamkrelidze and E. Mishenko (1962). *Mathematical Theory of Optimal Processes*. Interscience. New York, NY.

Pounds, P., R. Mahoney, P. Hynes and J. Roberts (2002). Design of a four-rotor aerial robot. In: *Proc. Austr. Conf. Robo. Auto.*. Auckland, New Zealand.

Procerus-Technologies (2008). *http://www.procerusuav.com/*.

Rajasekharan, S. and C. Kambhampati (2003). The current opinion on the use of robots for landmine detection. In: *In Proceeding of the 2003 IEEE International Conference on Robotics & Automation*. Taipei, Taiwan. pp. 4252–4257.

Rall, L. B. (1981). *Automatic Differentiation: Techniques and Applications. Lecture Notes in Computer Science*. Vol. 120. Springer-Verlag, Berlin.

RCToys (2008). www.rctoys.com.

Richards, A. and J.P. How (2002). Aircraft trajectory planning with collision avoidance using Mixed Integer Linear Programming. In: *Proc. 2002 Amer. Contr. Conf.*. Anchorage, Alaska.

Richards, A. and J.P. How (2004). Decentralized model predictive control of cooperating uavs. In: *In Proceeding of IEEE Conference on Decision and Control*. Bahamas.

Ross, I.M. and F. Fahroo (2002). A direct method for solving nonsmooth optimal control problems. In: *Proc. 15th IFAC Triennial World Congress.*}. *Barcelona, Spain.*

Ross, I.M. and F. Fahroo (2006). Issues in the real-time computation of optimal control. Mathematics and Computer Modelling **43**, *1172–1188.*

Scokaert, P.O.M., D.Q. Mayne and J.B. Rawlings (1999). Suboptimal model predictive control (feasibility implies stability). IEEE TRANSACTIONS ON AUTOMATIC CONTROL, **44***(3), 648–654.*

Sentoh, E. and A. E. Bryson (1992). Inverse and optimal control for desired outputs. Journal of Guidance, Control, and Dynamics **15**, *687–691*.

Seywald, H. (1994). Trajectory optimization based on differential inclusion. Journal of Guidance, Control, and Dynamics, **17***(3), 480–487.*

Shaw, B.C. (2005). Development of a non-linear model of the draganflyer iv quad-rotor. Master's thesis. Cranfield University.

Sinha, A.K., S. Atyeo, A. Schauenburg, A. Rathore, B. Quinn and T. Christoffersen (2006). Investigation on the application of uav technology in power line inspection. In: Proc 21st International UAV Systems Conf.

Starin, S. R., R. K. Yedavalli and A.G. Sparks (2001). Design of a LQR controller of reduced inputs for multiple spacecraft formation flying. In: Proceedings of the American Control Conference.

Stevens, B.L. and F.L. Lewis (1992). Aircraft Control and Simulation. *Vol. 1. John Wiley & Sons, Inc.*

Taranenko, V. T. and V. G. Momdzhi (1968). Direct Variational Method in Boundary Problems of Flight Dynamics. *Mashinostroenie Press. (in Russian).*

Tayebi, A. and S. McGilvray (2004). Attitude stabilization of a four-rotor aerial robot. In: Proc. 43rd IEEE Conf. Decision Contr.. *Bahamas.*

Tayebi, A. and S. McGilvray (2006). Attitude stabilization of a VTOL quadrotor aircraft. IEEE Transactions on Control Systems Technology **14***(3), 562–571*.

Telegraph (2007).

Tou, J.T. (1964). Modern Control Theory. *McGraw-Hill. New York, NY.*

Tournier, G.P., M. Valenti and J.P. How (2006). Estimation and control of a quadrotor vehicle using monocular vision and moire patterns. In: AIAA Guidance, Navigation, and Control Conference and Exhibit. *Keystone, Colorado.*

Valenti, M., B. Bethke, G. Fiore and J.P. How (2006). Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In: Proc. AIAA Conference. Guidance. Navigation. Control.

Vlassenbroeck, J. and R. Van Dooren (1988). A Chebyshev Technique for Solving Nonlinear Optimal Control Problems. IEEE Trans. Autom. Control **33, No. 4**, *333–340.*

von Stryk, O. and R. Bulirsch (1992). Direct and indirect methods for trajectory optimization. Annals of Operations Research **37**, *357–373.*

Waslander, S.L., G. Hoffmann, J.S. Jang and C.J. Tomlin (2005). Multi-agent quadrotor testbed control design: Integral sliding mode vs. reinforcement learning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. *pp. 468–473.*

Wengert, R. E. (1964). A simple automatic derivative evaluation program. Commun. ACM 7, **8**, *463464.*

Whidborne, J.F., I.D. Cowling and O. Yakimenko (2008). A direct method for UAV control. In: Proc 23rd International UAV Systems Conf. *Bristol, U.K.*

Wu, J.C., J.N. Yang and W.E. Schmitendorf (1998). Reduced-order H_∞ and LQR control for wind-excited tall buildings. Engineering Structures **20**, *222–236.*

Yakimenko, O. (2000). Direct method for rapid prototyping of near-optimal aircraft trajectories,. Journal of Guidance, Control, and Dynamics, **23**, *865–875.*

Yakimenko, O. (2006). Direct method for real-time prototyping of optimal control. In: International Conference Control-2006. *Glasgow, Scotland.*

Yamaguchi, Y., T. Maruyama and A. Konagaya (2001). An approach for homology search with reconfigurable hardware. Genome Informatics **12**, *374–375.*

Appendix A

Small Quadrotor Model

A.1 Simulink Model

A full dynamic model of the quadrotor is required for a number of reasons. For trajectory optimization, a model enables the realistic constraints to be included in the algorithm. Also with a full dynamic model of the quadrotor, the control schemes can be tested as shown throughout this thesis. This modelling has been done by Barry Shaw, as part of his MSc thesis (Shaw 2005). A full quadrotor model has been developed using experimental data and theoretical analysis and compiled in SIMULINK. Experiments have been carried out to model the rotor dynamics as well as determine the mass and inertias. The model can be seen in Figure A.3. This model is used for all simulations within this thesis with the one exception of Section 4.2.3 which uses the model for the larger quadrotor detailed in Appendix B.

This model essentially consists of five major parts; the powerplant, gravitational calculations, equations of motion, navigation and angular motion calculations. The powerplant uses look up tables constructed from the experimental data to convert voltage into rotor speed and thrust. Using blade element theory detailed in Section 2.4.2 the torque per rotor can be calculated. These are then converted to body forces using Equations (2.48-2.51.) In a similar way the gravitational force is converted into body axis forces within the gravitational calculations. The equations of motion then convert these forces using the experimental mass and inertial data into body rates $(u, v, w, p, q, r, \dot{p}, \dot{q}, \dot{r})$. The angular motion calculations convert these into Euler angles (ϕ, θ, ψ) and the navigational calculations converts these into positions and speeds $(x, y, z, \dot{x}, \dot{y}, \dot{z})$.

A.1.1 Experimental data

In order to determine the total thrust and the turning moments acting on the vehicle it is necessary to first determine the individual rotor speed and thrust from each rotor. Experimental data (see Figure A.1) from a pendulum test has been used to determine the voltage

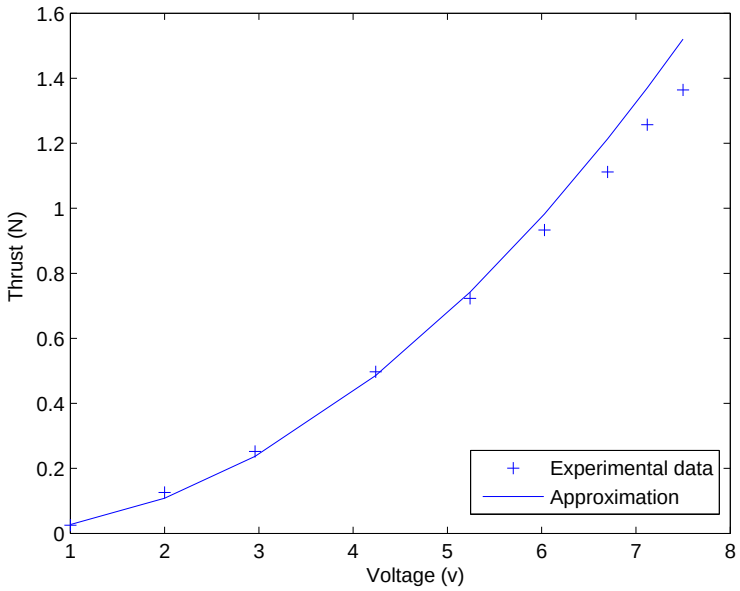


Figure A.1: Voltage to thrust

to thrust relationship. The following approximate relationship has been determined:

$$\tau = \frac{v^2}{37} \tag{A.1}$$

The voltage to rotor speed relationship has been determined using two different experimental procedures. The first was using a hand held tachograph and the second a stroboscope. The results can be seen in Figure A.2.

The moments of inertia have been calculated using a compound pendulum and a bifilar pendulum and are;

$$I_x = 8.33gm^2 \tag{A.2}$$

$$I_y = 7.14gm^2 \tag{A.3}$$

$$I_z = 1.24gm^2 \tag{A.4}$$

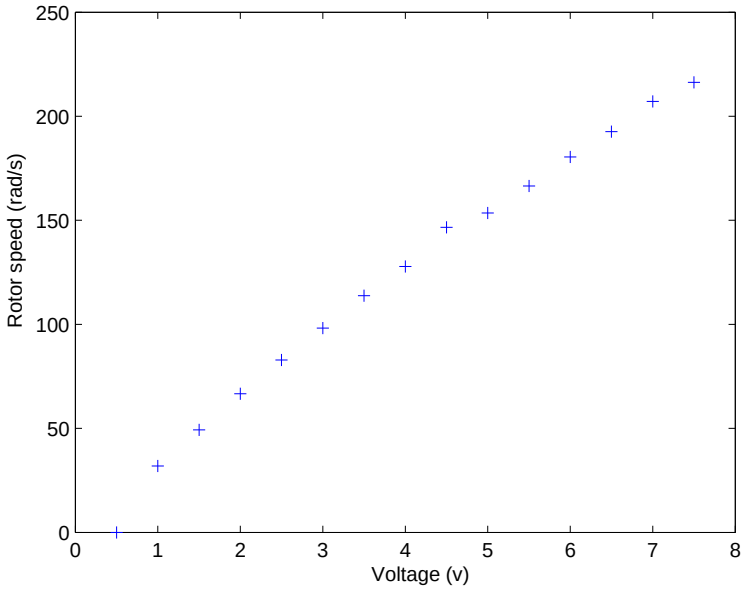


Figure A.2: Voltage to rotor speed

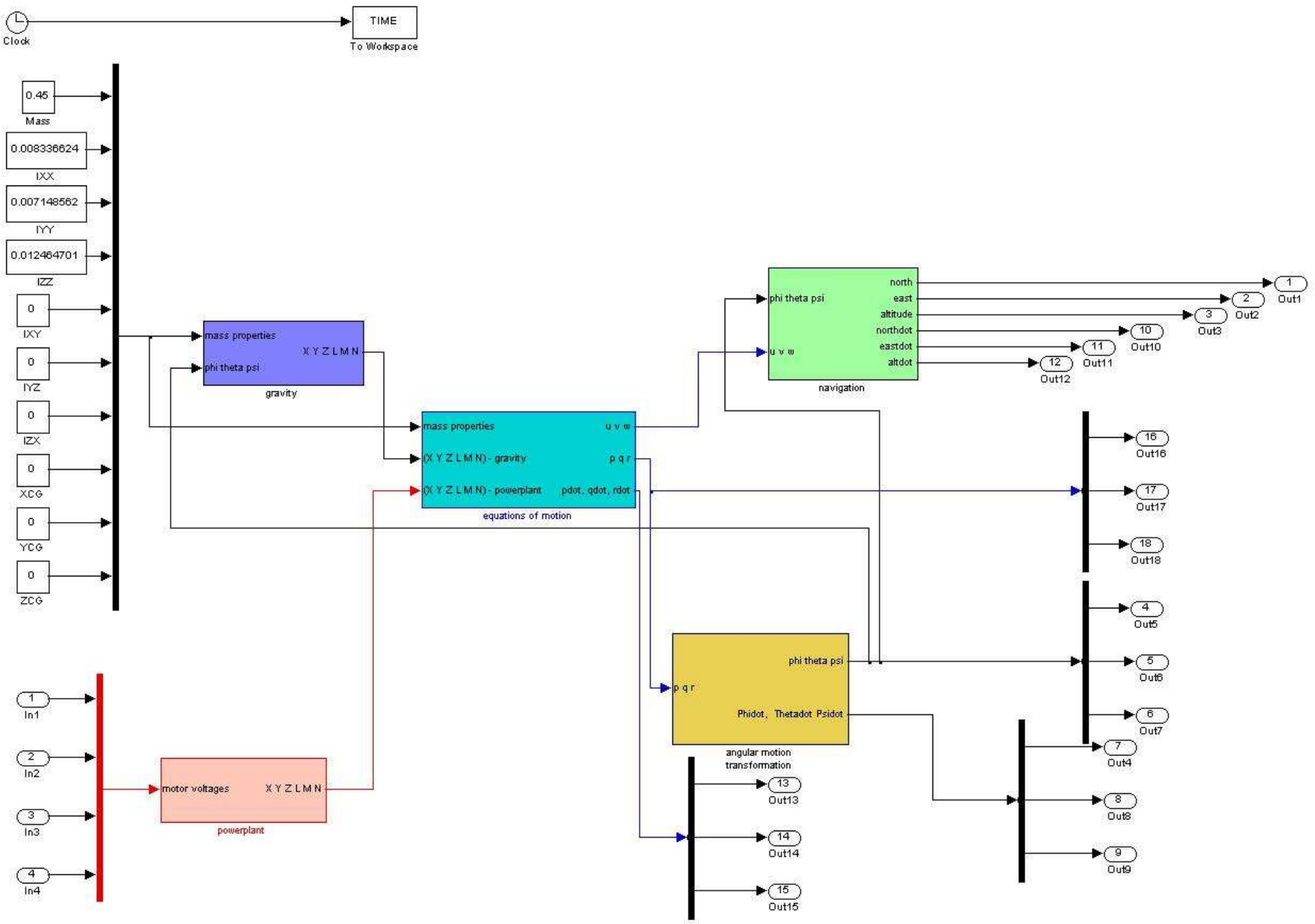


Figure A.3: Simulink Model

Appendix B

Draganflyer X Pro Model

In order to test the control schemes on the larger quadrotor a similar model to the one in Appendix A is required. Experimental data has been obtained from wind tunnel testing to obtain the voltage to thrust and voltage to speed relationship of the motors. As seen in Figure B.3 the motors are placed on a force balance within the open section of the wind tunnel and connected to a regulated power supply. This is tested for a range of angles at different wind speeds to obtain the dynamic properties of the model. For this model however, we are only interested in the zero pitch angle results. This work has been done by Vincent Martinez Martinez towards his MSc thesis Martinez Martinez (2007). The model presented in Appendix A has been adapted with this new data to test the LQR controller in Section 4.2.3.

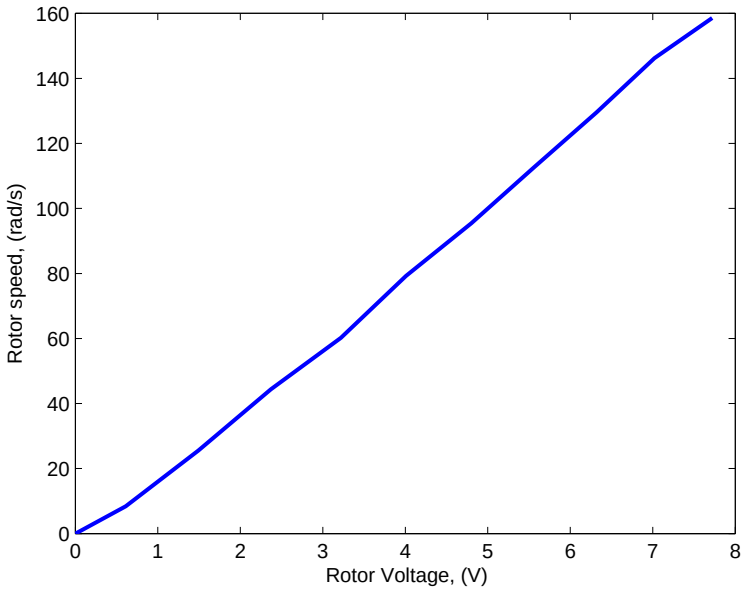


Figure B.1: Voltage to thrust

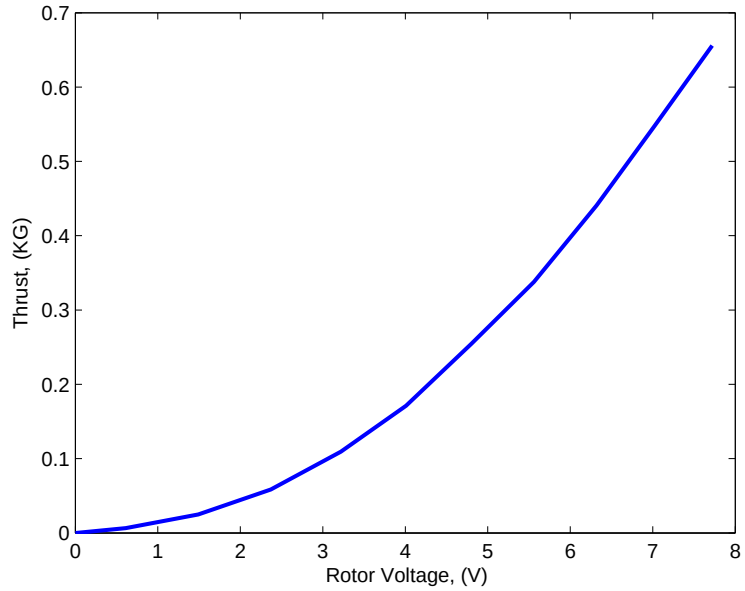


Figure B.2: Voltage to rotor speed

Figure B.3: Wind Tunnel testing of the Draganflyer X Pro