REAL TIME AUTONOMOUS COLLISION AVOIDANCE FOR UNMANNED AERIAL VEHICLES

By

Min Prasad Adhikari, B.Eng

Aeronautical Engineering

Nanjing University of Aeronautics and Astronautics, 2012

A thesis presented to the Ryerson University

in partial fulfillment for the degree of

Master of Applied Science

in the Program of

Aerospace Engineering

Toronto, Ontario, Canada, 2015

ABSTRACT

Real Time Autonomous Collision Avoidance for Unmanned Aerial Vehicle

Min Prasad Adhikari, Master of Applied Science, Aerospace Engineering

Ryerson University, Toronto, 2015

GeoSurv II is a jointly funded project of Sander Geophysics Limited (SGL) and NSERC to develop a fixed-wing Unmanned Aerial Vehicle (UAV), capable of autonomously performing high resolution geophysical surveys at low flight altitudes over poorly known terrain. This thesis is in support of achieving this objective.

In order to achieve such a level of autonomy, the UAV must be capable of avoiding stationary, pop-up and moving obstacles while flying at low altitude. Such obstacles may include power lines, communication towers, trees, unknown flight vehicles encountered while at flight or uneven terrains which creates the situation of the pop-up obstacles. In addition to that the UAV must be able to fly as close as possible to the reference trajectory for a given geophysical survey. The development and testing of a method capable of performing such an autonomous mission is the objective of this thesis.

In this thesis, a method is developed based on a spectral method known as Legendre Pseudospectral Optimal Control, because of its capability to directly incorporate all of the mission objectives, while respecting the UAV constraints (which other methods in the literature are not capable of). The method accounts the aircraft and obstacle constraints there by capable of avoiding obstacles with feasible maneuvers for the aircraft. The objective to remain as close to the reference trajectory is fulfilled by setting the area between the flight trajectory and reference trajectory as the cost of optimization of the optimal control problem. Five different scenarios presented in this thesis show the developed method's capability to avoid the stationary, pop-up and the moving obstacles successfully while remaining close to the reference trajectory.

ACKNOWLEDGEMENTS

I am grateful to my supervisor Prof. Anton de Ruiter, who gave me such an opportunity to come to Ryerson University for the master program from Nepal. His support and encouragement on different occasion would always be appreciated. Moreover, it would have been impossible to prepare this thesis without his regular assistance and guidance.

I am grateful to my parents, Shiva Prasad Adhikari and Lila Devi Adhikari, for always supporting me to pursue my dream.

I thank my sister, Bishnu Ahikari/Kharel, for always supporting me at the time of difficulties.

My sincere gratitude to Sangita Aryal, who always encourage and supports me with care all the time.

Finally, it would have been difficult to adjust in Toronto without love and care from Deepak Lamichhane and Saru Deuja/Lamichhane. They will always remain in my heart.

“While we contemplate our needs, the universe is contemplating itself”

-Inspired by Carl Sagan.

Table of Contents

List of Tables

List of Figures

Nomenclature

AGL	Above Ground Level
CC	Complementarity Conditions
HEE	Hamiltonian Evolution Equation
HMC	Hamiltonian Minimization Condition
HVC	Hamiltonian Value Condition
KKT	Karush-Kuhn-Tucker
LGL	Legendre-Gauss-Lobatto
NL OCP	Nonlinear Optimal Control Problem
NLP	Nonlinear Programming/ Non Linear Program
OA	Obstacle Avoidance
OCM	Optimal Control Method
OD	Obstacle Detection
ODE	Ordinary Differential Equation
PS	Pseudospectral
PS OCM	Pseudospectral Optimal Control Method
RC	Radio Control
RK	Runge-Kutta
ROA	Reactive Obstacle Avoidance

RTOC	Real Time Optimal Control
SNOPT	Sparse Nonlinear OPTimizer
TP	Trajectory Planning
TTC	Terminal Transversality Condition
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

Chapter 1

Introduction

1.1 Introduction

Aerospace has always challenged nature, since the very beginning. After the controlled flight by the Wright brothers in 1903 [4], the feasibility of aircraft to carry weights heavier than air in a controlled manner was demonstrated. Even today, we seek new possibilities either driven by a new need, or simply because we can. The type of aircraft discussed in this thesis is an Unmanned Aerial Vehicle (UAV). Unmanned Aerial Vehicles, also known as drones, are flying machines without any human pilot on board. They are either controller autonomously[1] by computers on board or remotely piloted.

UAVs have several advantages and have been proven to be useful in different tasks such as domestic policing [5], disaster relief [6], scientific research [7], conservation [8] and many other up to our imagination. Among these many possible applications, is their use in geophysical surveys [9]. Because of their ability to fly at lower altitudes than manned flight [10], they become ideal for performing high resolution surveys. Moreover, driven by advancements in computer processing power, UAVs have made significant progress within the last 10 years [11]. In terms of aerial surveys, the use of UAVs in aerial surveillance has become more common these days [12,13,14,15]. While UAVs being ideal for aerial surveillance, they are mostly used by military personnel. Apart from that, UAVs are more attractive to different types of users for several reasons. Among them, first is the use of UAVs to operate in situations dangerous for human pilots. Next, due to the absence of an on board pilot, small-scaled aircraft can be used decreasing the cost of manufacturing and fuel consumption, compared to larger piloted aircraft.

UAVs are of different types based on their size and the tasks that they perform. In addition,

[1]Autonomy is, namely the ability of the UAV to fly without human intervention.

features such as range, maneuverability, autonomy determine the UAV's capabilities. Among them autonomy determines the UAV's capability to operate without human interference. At a primary level of autonomy the UAVs are able to complete the flight mission based on pre uploaded flight plans. UAVs with the ability to avoid obstacles and potential collision scenes have an even higher level of autonomy. Also, the level of autonomy can be even higher, such as in the case of forest fire monitoring [16]. For example, in the case of forest fire monitoring, a camera attached to the UAV along with an image processing system passes an image of the fire's edge points indicating the burning side. These points are further approximated by straight lines creating a perimeter around the fire area. The path planning algorithm then places waypoints along the approximated fire perimeter and finally the autopilot is used to determine the control sequence required to maneuver the UAV to the waypoints. The motivation for the study in this thesis is the GeoSurv II project, which is described in more detail in next sub-section.

1.2 GeoSurv II Project

GeoSurv II is a jointly funded project of Sander Geophysics Limited (SGL) and NSERC with the objective to develop a small UAV, capable of performing high resolution geophysical surveys at low flight altitudes [9]. The designed aircraft is a twin-boom pusher aircraft with a wingspan of 4.9m, and two magnetometers installed on its wing tips to measure the magnetic field of the terrain the UAV is flying over. Research has been conducted to reduce the interference of ferromagnetic sources of the aircraft on the survey [17] and also on the development of low cost composite structure for the airframe [18]. Figure 1.1 from [9] shows the GeoSurv II prototype.

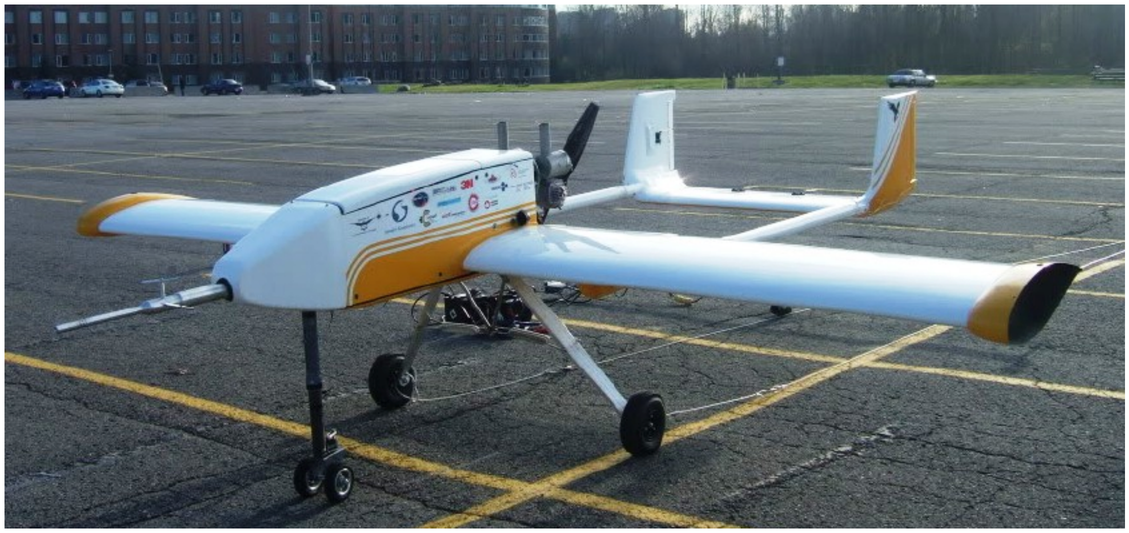


Figure 1.1: GeoSurv II prototype

Currently, Sander Geophysics Ltd. requires four crew members (pilot, co-pilot, maintenance engineer and geophysicist) to perform such surveys and the objective of the GeoSurv II project is to develop a UAV with the level of autonomy where the maintenance engineer and the geophysicist would be able to perform the survey without the pilots [9]. The autonomous system thus developed would be able to avoid obstacles detected without operator intervention. Basically the system has two separate areas: *Obstacle Detection* (OD) and *Obstacle Avoidance* (OA); this thesis is focussed on OA.

Thus, the autonomy is an essential feature of the GeoSurv II aircraft. This thesis examines the trajectory planning aspect of obstacle avoidance (OA) for the GeoSurv II project. The Obstacle detection (OD) aspect is not covered. The next paragraphs cover the mission scenario for GeoSurv II.

Survey Area and Potential Obstacles

A typical GeoSurv II mission profile involves the survey of a given area. The required flight path consists of equally spaced lines over the area of interest. To obtain high-resolution measurements of the magnetic field, it is desired to fly below 50m AGL (Above Ground Level) [9]. As such, the UAV might face different problems, such as a-priori unknown small hills or other unforeseen obstacles. Consequently this thesis addresses the autonomous collision avoidance[2] technique for such a UAV. In particular, moving, stationary and pop-up[3] obstacles

[2]Here *collision avoidance* means *obstacle avoidance*, where as in the literature collision avoidance means avoiding other UAVs/aircraft [19, 20, 21, 22] .

[3]Obstacles that appear suddenly ahead of UAVs

are considered.

Any structure or object that comes in the way of the UAV is a potential obstacle. They may range from stationary obstacles like power line towers with power lines, telecommunication towers, trees and industrial chimneys to moving obstacles such as birds, and unknown flight vehicles [23]. Pop-up obstacles become of increasing significance if the UAV is to operate in environments with uneven terrain . For example, if a small hill is encountered, the UAV cannot see what lies behind it. In cluttered environments, pop-up obstacle can be any object which is hidden by a visible obstacle and is encountered suddenly after the obstacle is avoided.

The objective for GeoSurv II, is for an operator to upload a flight plan onto the UAV, which the UAV will autonomously carry out. Any encountered obstacles are to be avoided, while staying as close as possible to the original flight plan, so as to minimize gaps in the resulting survey.

1.3 Existing Approaches to Obstacles Avoidance

Obstacles avoidance (OA) has been a topic of research required in autonomous operation. It was first researched in the field of robotics. Details of some of these approaches may be found in references [24, 25, 26, 27, 28]. Obstacles avoidance methods were later applied to the UAVs with the desire of autonomous operation [29, 30, 31]. UAVs have different properties and constraints compared to non-aerial robotics. Because of this, the methods developed in robotics often cannot be directly applied to UAVs, and need to be modified. For example, mobile robots can come to complete stop to undertake a maneuver, while fixed-wing UAVs need to maintain a minimum flight speed. Moreover, UAVs have minimum turning and pull up radii which further constrain possible UAV maneuvers. These constraints should be accounted for when designing obstacle avoidance methods for fixed-wing UAVs.

There are generally two classes of methods for obstacle avoidance:

(1) When a complete map of the environment (map with all obstacles) is known prior to the start of the mission, such method of avoiding obstacles is known as *Trajectory Planning (TP)*. Trajectory Planning methods have also been called path planning, global path planning or global guidance.

(2) When obstacles are unknown prior to the start of the mission and are encountered during flight, the method is called Reactive Obstacle Avoidance (ROA) or local path planning [9].

Obstacles avoidance methods at local and global levels, have different characteristics and are applicable in different situations. In the global approach of obstacle avoidance (Trajectory Planning), the computation time is not of concern. Since the method is run prior to the mission, it can run on a powerful computer with significant computational power. Due to the lack of a computational time constraint, optimality is of greater focus in Trajectory Planning. In contrast, computation time of the method in Reactive Obstacle Avoidance is of great importance, since a delay in response might lead to collision. Since, Reactive Obstacle Avoidance is performed in real-time[4] on board, memory resources and computational power are limited by the available on-board computational resources. In Reactive Obstacle Avoidance, robustness is of greater concern than optimality since there is very little chance to correct any errors that may have occurred in the obstacle avoidance computations.

Various obstacle avoidance methods have been developed and continue to evolve with improvement. Details of some of these approaches may be found in references [32,33,34]. These techniques include artificial potential fields, cell decompositions, road maps and pseudospectral optimal control methods, among others. Each method has advantages and disadvantages on the basis of completeness, optimality, computational complexity and scalability [35,32,34]. Completeness refers to the method's ability to find a successful solution if it exists and report if no solution exits. Optimality refers to the method's ability to select the path that best fits the given criteria. Computational complexity refers to the time taken by method to find the solution, and scalability refers to the method's ability to extend to higher dimensional complex systems [35]. OA methods are discussed in further details in the subsections below.

1.3.1 Cell Decomposition Method

In cell decomposition methods, the operational area is divided into similar shaped small regions called cells, which are non-overlapping. Possible routes are then generated using search algorithms like **A***, that pass through adjacent free cells. Cells that share a common boundary are referred to as adjacent and free cells are ones that are not occupied by obstacles. Figure 1.2 shows the route selected based on the cell decomposition method. Starting and finish points are connected by straight lines which are free from obstacles. Cells containing obstacles are shaded (cross-hatched) while the actual obstacles are solid black. Those cell containing obstacles are excluded from path development as the vehicle cannot travel through them while the free cells are traversable cells. The optimality of the solution is based on the resolution of the cell grid. Higher resolution results in improved optimality, but at the expense of increased computational burden. This increased computational burden can be overcome by implementing quad-tree or octree data structures [36]. Despite these new data structures, the inability to incorporate the vehicle constraints limits the usefulness of

[4]online or while at flight

this method for fixed-wing UAVs. Furthermore, this method is more suited for Trajectory Planning than Reactive Obstacle Avoidance, since it requires the generation of an entire path.

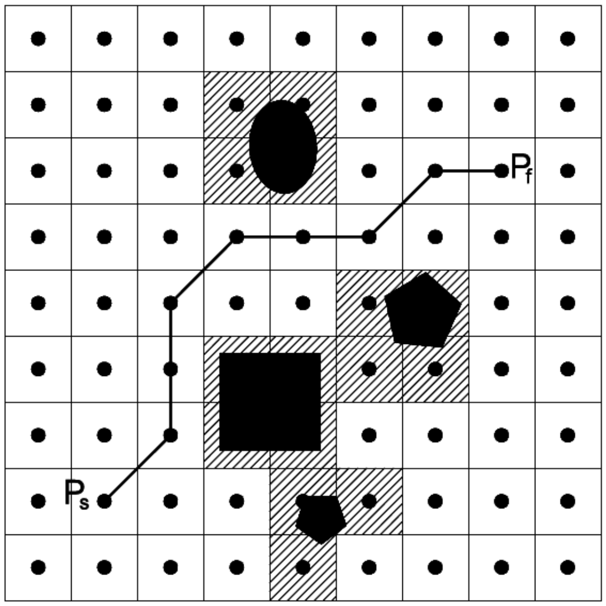


Figure 1.2: Cell Decomposition from reference [1]

1.3.2 Road Map Method

A road map is a network of straight lines, which connect the vehicle's initial and target points, without passing through any obstacles. The method work in configuration space, where the robot is treated as a point mass and the space is adjusted to accommodate the physical size of vehicle. A search algorithm is used to generate the shortest and safe route (route without intersecting obstacles) between the initial and goal points. Figure 1.3 from [1] illustrates the method. First, a map is defined as a work space (example: C_{space}). Next, the space is split into free space and obstacles. Then, a network of straight lines are generated connecting a set of points in the free space forming polygonal shapes in C_{space}. These connected lines must not intersect obstacles. The graph produces a network of possible collision free routes. Finally search algorithms like **A*** are used to select a route between the initial and final points based on merit.

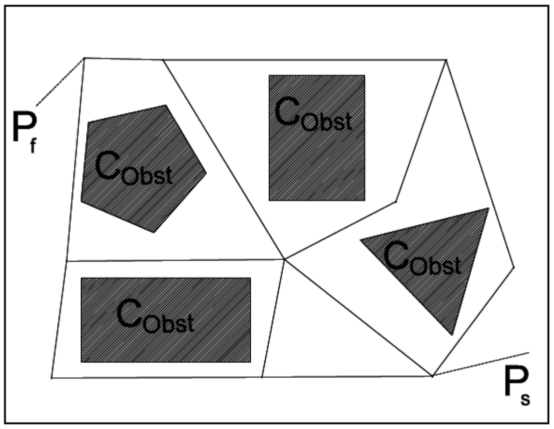


Figure 1.3: The road map method

Other road maps methods used in the literature are *visibility graphs* and *voronoi diagrams.*

Visibility Graphs

In this method, paths are formed by connecting networks of a non-directed graph of straight lines. This method consider only obstacles with polygonal shapes [1]. In the graph, vertices of polygonal obstacles are connected by straight lines called edges. The edges should not intersect obstacles. Edges that connect vertices are drawn such that each vertex can be seen from the other. Finally using a search algorithm based on some metric, a route is found connecting the start and end points. Figure 1.4 shows a visibility graph.

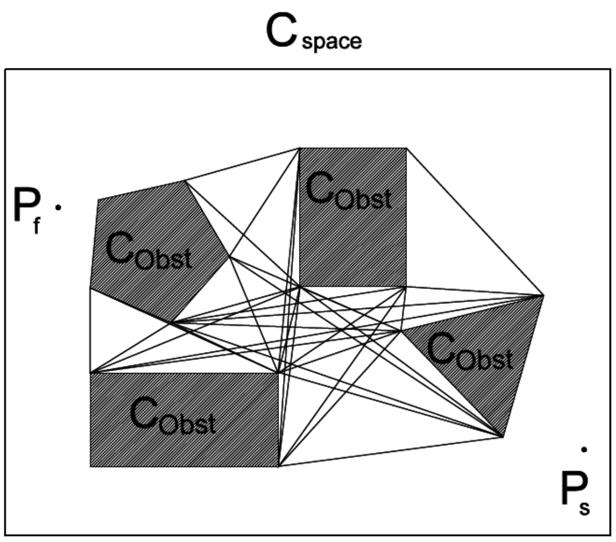


Figure 1.4: Visibility graph from [1]

Voronoi Diagrams

In Voronoi diagrams, a set of polygonal fences are constructed around each of the obstacles, by drawing lines perpendicular to the lines connecting the centers of the obstacles. The polygonal fences are then adjusted to meet at a minimum number of vertices [1]. Considering the resulting polygonal fences as connected graphs, a search algorithm can be used to find a connected route between the initial and final positions.

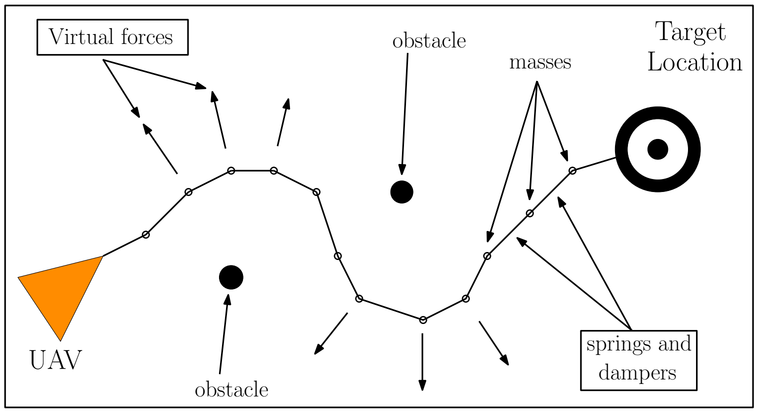


Figure 1.5: “radar site” obstacles from [2]

The use of Voronoi diagrams in reference [37] to produce paths for a UAV shows that the method does not directly account for kinematic constraints. Another approach for path generation using Voronoi diagrams is shown in Figure 1.5. In this paper, an initial path connecting the initial and final points is generated using a Voronoi diagram. The resulting path consists of a series of connected straight lines. This path is used as the starting point for generating a more smooth path. In particular, the UAV’s path is modeled as a chain of point masses, between the initial and final points, connected by springs and dampers, with the idea that the tension in the springs act to shorten the path. Each obstacle is assigned a virtual repulsive force acting on the chain. The final generated path is the equilibrium condition of the chain of point masses. However, the complexity in solving the resulting ODEs limits the use of this method in Reactive Obstacle Avoidance. Furthermore, for these types of methods, adequate global information is needed, further limiting their applications to Reactive Obstacle Avoidance. So road maps are efficient for a-priori known static environments. In the case of dynamic environments, road map implementation is more challenging. In brief, road map methods are suited for static environments and Trajectory Planning.

1.3.3 Artificial Potential Fields Method

First proposed by Khatib in 1985 [38], the artificial potential field technique treats the goal point as an attractive potential and obstacles as repulsive potentials. An artificial potential force is then computed, which is applied to the vehicle as a control input. The vehicles moving in the workspace are attracted towards the goal and repelled by the obstacles. In this approach the vehicle follows the path towards the lowest potential. This method is suitable for online path generation in robust manner with local information, since no global path planning is required.

Despite its benefits in local path planning, this method has several drawbacks. One of them is, the vehicle may get trapped at a local minimum point. For example, in the case of a **C** shaped obstacle, the potential has a minimum at the focal point of the curvature of **C.** Hence, a vehicle trapped at the focal point cannot come out of it, which is problem in the completeness of the method. This has been addressed by combining the potential field with a distance transform method, where the distance transform method is used to generate a global path from initial to goal point, while the potential field method is used to avoid obstacles in the immediate vicinity of the vehicle [39]. However, this method seems to handle vehicle constraints poorly. In addition, there have been other approaches for handling local minima. For example, in reference [40], this was addressed by imposing random motion and escape force at local minima. This is not appropriate for air-vehicles. Reference [33] discusses the development of navigation functions[5] containing only one minimum, at the goal point. However, this can only be done in presence of global information hence sacrificing the method benefits for Reactive Obstacle Avoidance.

Optimality of the APF method can be slightly improved by placing tunable gains on the repulsive forces of the obstacles [41]. However, it lacks portability. In summary, this method can be useful to avoid obstacles in presence of local information, that is well suited to reactive obstacle avoidance (ROA) in robust manner, but it is difficult to manage vehicle constraints and also it lacks optimality.

1.3.4 Potential Flow Method

Potential flow method is based on the concept that a uniform fluid flow creates a natural path around an obstacle. Unlike the Artificial Potential Field (APF) method, Potential Flow does not generate an artificial force (acceleration), rather a velocity vector is generated, which gives a direction the UAV needs to follow. Reference [9] uses the potential flow method

[5]function defined such that only the goal point has minimum potential

for Reactive Obstacle Avoidance (ROA). In addition, potential flow avoids the problem of local minima, since the flow potential is harmonic. Figure 1.6 shows the method developed to maneuver the UAV around the obstacle [9]. The actual obstacle is bounded by a box with wedges at the front and rear. The wedges were designed specifically to shape the flow in order to account for the aircraft turning and pull-up constraints for a single obstacle as shown in the Figure 1.6. The method works well if there are few obstacles which are far enough apart.

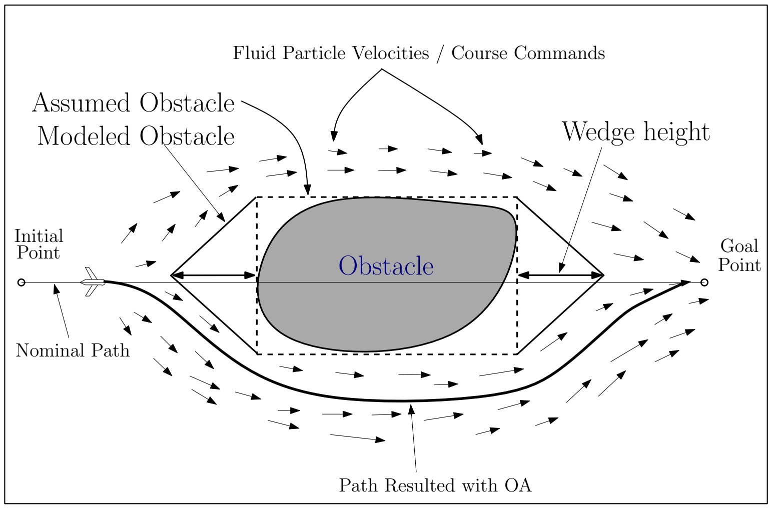


Figure 1.6: Obstacle avoidance using potential flow

The drawbacks in this method are, first, obstacles are two-dimensional, so it can either make the UAV fly around or over the obstacle. This leads to the need for heuristic rules, which become very complicated if there are many closely spaced obstacles. Second, it cannot handle moving obstacles.

1.3.5 Rapidly Exploring Random Tree

Rapidly Exploring Random Tree (RRT), is a probabilistic method. Probabilistic methods tend to form path by random selection of neighborhood points with some metric such as path safety.

In RRT, starting with a 'tree' generated from a point (initial point), random points are generated and nodes on the tree nearest to each point are established. Then new nodes are created along the lines connecting these points to the nearest nodes. At last these new nodes are added to the 'tree' if they do not hit obstacles. Figure 1.7 shows the RRT method for path planning in an urban environment.

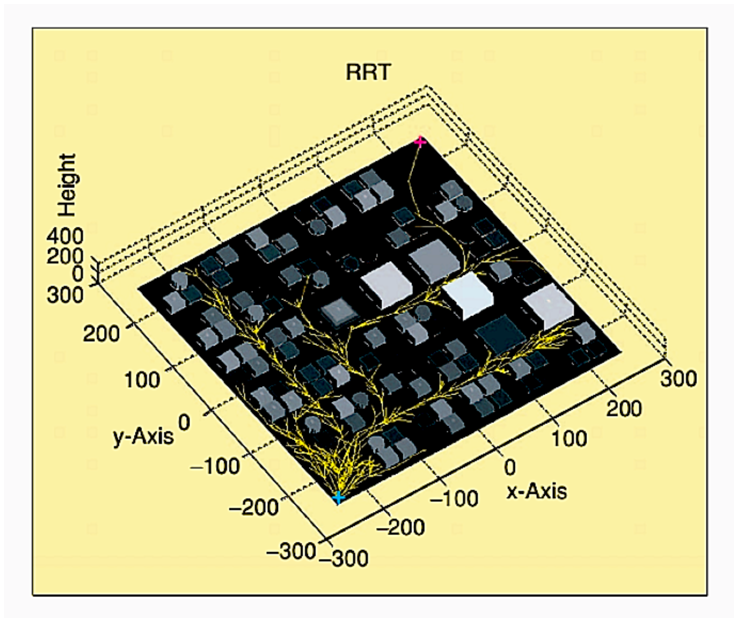


Figure 1.7: RRT method used for path planning from [3]

In reference [30], path planning achieved by RRT was added to the Dijkstra search algorithm, successfully finding a route for a UAV flying among known static obstacles. Also in reference [42], RRT was used for path planning with positional uncertainty of threat regions. In the latter, the lines joining points were adjusted with arcs to make a path flyable by a UAV. These methods are computationally intensive while checking every node for obstacles. Hence, it is more Trajectory Planning oriented than Reactive Obstacle Avoidance.

1.3.6 Pseudospectral Optimal Control Method

The optimal control method is a logical way of solving path planning problems with objective functions (that need to be minimized/maximized), constraints and boundary conditions (that need to be strictly satisfied). This method in general solves for a control history that satisfies the problem. Due to the difficulty in solving these optimal control problems analytically they are solved numerically. Despite its benefit for optimal path planning, the possible dimension and complexity of the problems and the available solution methods have in the past created heavy computational burdens with long computational times. With the advancement in computer technology and algorithmic advancements, the issue of computing burden has been reduced. Also improvements in problem formulation have further improved computation time [43].

There are various important intermediate steps before the problem can be solved numerically. It is important to correctly formulate the problem at first. Next, a proper discretization scheme for the problem is important. The type of discretization scheme, number of discretization points, proper scaling and balancing of the problem determines the correctness of the numerical solution. Details of scaling and balancing can be found in reference [43]. The problem is posed as a non-linear programming problem and finally solved using an NLP (Non Linear Program) solver.

Recent improvement in the optimal control method was made by the introduction of a discretization scheme called Legendre Pseudospectral discretization, developed by Fariba Fahroo and I. Michael Ross of the Naval Postgraduate School [44].

The Legendre Pseudospectral optimal control method has been used in offline trajectory optimization [45,46,47]. Recent applications shown in references [11,48,44,49,50] have demonstrated the implementation of the Legendre Pseudospectral optimal control method in online trajectory planning in feedback form, also accounting for disturbances. In reference [11] path planning for a multi-UAV problem has been solved successfully. The method was used for the development of collision-free multi-UAV trajectories. In this paper, a concept of spacing constraint was used between the UAVs to prevent collision. Reference [48] applies the Legendre Pseudospectral optimal control method to trajectory planning for a UGV (Unmanned Ground Vehicle). The method's potential in online trajectory planning is highlighted in reference [44]. This paper highlights the relatively low computational power needed for the Pseudospectral optimal control method. It shows that if the problem be solved fast enough, then the method can solve the problem of local path planning, thereby accounting for environmental changes. In reference [49], the method was able to solve the problem of autonomous trajectory planning for an unmanned ground vehicle (UGV), while avoiding obstacles. It shows that the method was able to avoid obstacles autonomously provided there was sufficient computational power to solve each trajectory planning problem quickly enough. In [35] the method was applied to the UAV obstacle (stationary) avoidance problem. The method was highly successful in real-time implementation for UAVs thereby demonstrating its utility for both Trajectory Planning and Reactive Obstacle Avoidance.

Reference [50] shows an implementation of the method to a re-entry and landing problem. In addition the system thus developed was able to compensate for large uncertainties and disturbances such as hurricane-force wind gusts.

1.4 Approach, Objective and Thesis Structure

Given the above discussions about the different methods for obstacle avoidance, the approach taken in this thesis will be based upon the pseudospectral optimal control method.

Unlike previous applications of the pseudospectral optimal control method to the collision avoidance problem, the objective in this thesis is not just to get from a starting point to a goal point, but rather to stay as close as possible to a pre-defined path, which is suitable for a survey[6] while avoiding stationary, moving and pop-up obstacles. Since the method is developed for the fixed wing UAVs, this thesis uses different constraints than reference [35], such as minimum pull-up and turning radii. The method developed is expected to be computationally efficient so that it could be implemented on board.

The remainder of the thesis is organized as follows.

Chapter 2 contains an introduction to the optimal control problem and the theory behind the Legendre Pseudospectral discretization method. Problem identification and formulation is covered in Chapter 3. Chapter 4 shows the results of different flight scenarios with aircraft constraints in real-time. Finally, conclusions and recommendations for future work are contained in Chapter 5.

[6]Survey area is partitioned with equally spaced lines, these lines acts as path for UAV.

Chapter 2

Pseudospectral Optimal Control Theory

2.1 Overview

This chapter presents an overview of the pseudospectral optimal control method and discusses advantages over conventional methods. In classical methods closed-form solutions can only be found in the most simple of cases. There are numerous difficulties in solving a state and control constrained nonlinear optimal control problems analytically, hence numerical methods are widely accepted and used. In particular, the Legendre pseudospectral method is able to solve complex nonlinear optimal control problems successfully with exponential convergence rates. Therefore, this method is selected for solving the problem of autonomous trajectory generation and obstacle avoidance for UAVs in this thesis.

The Legendre pseudospectral method is a spectral-based algorithm for solving nonlinear optimal control problems. It was developed by Fariba Fahroo and I. Michael Ross of Naval Postgraduate School [43]. Using Lagrange interpolating polynomials, this method discretizes the state and control trajectories, transforming the infinite-dimensional optimal control problem into a finite-dimensional nonlinear programming problem. Finally, a nonlinear programming (NLP) solver is used to solve the resulting finite-dimensional optimization problem. The pseudospectral method is commercially available in a MATLAB based software package called DIDO [51], which uses an NLP solver called SNOPT (Sparse Nonlinear OPTimizer) [52]. The DIDO software will be used to solve the different optimal control problems posed in this thesis.

Ultimately, the pseudospectral method is a means to solve optimal control problems. As such, the resulting solutions must satisfy the associated necessary conditions for optimality, given in Pontryagin's minimum principle [53]. The optimal control problems under consideration in this thesis, and the corresponding Pontryagin's minimum principle are discussed next.

2.1.1 Nonlinear Optimal Control Problem (NL OCP)

Nonlinear optimal control problems consist of a cost function, a dynamic equation, path and control constraints, and end point constraints. The cost function can consist of a Mayer term, $E(\cdot)$, which depends on the endpoint and/or the final time, a Lagrange term, $\int_{t_i}^{t_f} F(\cdot)dt$, which depends on the entire history of the state and control, or a combination of both of them, in which case the cost function is said to be in Bolza form. The dynamic equation adds a dynamic constraint to the optimal control problem, since it gives a differential equation that the state-control pair must satisfy. The problem is then solved by finding an optimal state-control function pair, $\{x(\cdot), u(\cdot)\}$ that minimizes the objective function, while satisfying all of the constraints. Such an optimal control problem is described mathematically as in equations (2.1) and (2.2) below.

Consider the optimal control problem, where it is desired to find a state-control pair $(x(t), u(t)) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_u}$and final time $\tau_f > \tau_0$ to minimize the cost function

$$J(x(\tau), u(\tau), \tau_f) = E(\tau_f) + \int_{\tau_0}^{\tau_f} F(x(\tau), u(\tau), \tau)d\tau, \tag{2.1}$$

subject to

$$\begin{array}{rcl} \dot{x}(\tau) & = & f(x(\tau), u(\tau), \tau), \\ 0 & \leq & g(x(\tau), u(\tau), \tau), \\ 0 & \leq & h(x(\tau), \tau), \\ x(\tau_0) & = & x_0, \\ x(\tau_f) & = & x_f. \end{array} \tag{2.2}$$

It is assumed that the functions

$$\begin{array}{rcl} E & : & \mathbb{R} \to \mathbb{R}, \\ F & : & \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \to \mathbb{R}, \\ f & : & \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \to \mathbb{R}^{N_x}, \\ g & : & \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \to \mathbb{R}^{N_g}, \\ h & : & \mathbb{R}^{N_x} \times \mathbb{R} \to \mathbb{R}^{N_h}, \end{array}$$

are all continuously differentiable in their arguments. These functions are often referred to as the problem data. Define the admissible control set,

$$\Omega(x, \tau) = \left\{u \in \mathbb{R}^{N_u} : g(x, u, \tau) \geq 0\right\}. \tag{2.3}$$

and define the control Hamiltonian

$$H(x,u,\lambda,\tau) = F(x,u,\tau) + \lambda^T f(x,u,\tau), \tag{2.4}$$

and the Lagrangian of the Hamiltonian

$$L(x,u,\lambda,\mu,\nu,\tau) = H(x,u,\lambda,\tau) + \mu^T g(x,u,\tau) + \nu^T h(x,\tau), \tag{2.5}$$

where $\lambda(\tau)$ are co-states and $\mu(\tau),\nu(\tau)$ are Lagrange multipliers. Then Pontryagin's necessary condition for an optimal solution to (2.1) and (2.2) is as follows [54].

Suppose that the triple $(x^*(\tau),u^*(\tau),\tau_f)$ are a minimizing solution of the problem described by equations (2.1) and (2.2). Then, there exist piecewise continuous co-state and multiplier functions $\lambda^*(\tau),\mu^*(\tau),\nu^*(\tau)$, a vector $\eta^*(\tau_i)$ for each point τ_i of discontinuity of λ^* and vectors β^*,γ^* such that:

1. *$\lambda^*(\tau)$ satisfies*

$$\dot{\lambda^*}(\tau) = -\frac{\partial L}{\partial x}(x^*(\tau),u^*(\tau),\lambda^*(\tau),\mu^*(\tau),\nu^*(\tau),\tau), \tag{2.6}$$

with Terminal Time Transversality condition

$$\lambda^*(\tau_f) = \beta^* + \left(\frac{\partial h}{\partial x}(x^*(\tau),\tau)\right)^T \gamma^*, \tag{2.7}$$

and jump conditions at the discontinuity points

$$\lambda^*(\tau_i^-) = \lambda^*(\tau_i^+) + \left(\frac{\partial h}{\partial x}(x^*(\tau),\tau)\right)^T \eta^*(\tau_i). \tag{2.8}$$

2. *At every point in the interval $\tau \in [\tau_0,\tau_f]$, $u^*(t)$ satisfies the Hamiltonian Minimization Condition*

$$u^*(\tau) = argmin_{u\in\Omega(x^*(\tau),\tau)} H(x^*(\tau),u,\lambda^*(\tau),\tau), \tag{2.9}$$

which has the associated Karush-Kuhn-Tucker (KKT) minimization condition

$$\frac{\partial L}{\partial u}(x^*(\tau), u, \lambda^*(\tau), \mu^*(\tau), \nu^*(\tau), \tau) = \frac{\partial H}{\partial u}(x^*(\tau), u, \lambda^*(\tau), \tau) + \left(\frac{\partial g}{\partial x}(x^*(\tau), u, \tau)\right)^T \mu^*(\tau) = 0. \tag{2.10}$$

3. *The Lagrange multipliers $\mu^*(\tau)$ and $\nu^*(\tau)$ satisfy the complementarity conditions*

$$\mu^*(\tau) \geq 0, \mu^{*T}(\tau) g(x^*(\tau), u^*(\tau), \tau) = 0, \nu^*(\tau) \geq 0, \nu^{*T}(\tau) h(x^*(\tau), \tau) = 0, \tag{2.11}$$

at every point in the interval $\tau \in [\tau_0, \tau_f]$, and at the terminal time, γ^* satisfies the complementarity condition

$$\gamma^* \geq 0, \gamma^{*T} h(x^*(\tau_f), \tau_f) = 0. \tag{2.12}$$

4. *Denoting the optimal Hamiltonian by $H^*(\tau) = H(x^*(\tau), u^*(\tau), \lambda^*(\tau), \tau)$, it satisfies*

$$\frac{dH^*}{d\tau} = \frac{\partial L}{\partial \tau}(x^*(\tau), u^*(\tau), \lambda^*(\tau), \mu^*(\tau), \nu^*(\tau), \tau) \tag{2.13}$$

with terminal condition

$$H^*(\tau_f) = -\frac{\partial E}{\partial \tau}(\tau_f) - \gamma^{*T} \frac{\partial h}{\partial \tau}(x^*(\tau_f), \tau_f). \tag{2.14}$$

together with the jump conditions at the points of discontinuity

$$H^*(\tau_i^-) = H^*(\tau_i^+) - \eta^{*T}(\tau_i) \frac{\partial h}{\partial \tau}(x^*(\tau_i), \tau_i). \tag{2.15}$$

Finally, discontinuity points, τ_i, can only occur when the solution $x^*(\tau)$ touches the boundary of the set $\left\{x \in \mathbb{R}^{N_x} : h(x, t) \geq 0\right\}$.

A few more points can be noted. First, differentiating the control Hamiltonian in (2.4) with respect to λ, and using the dynamic equation in (2.2) gives

$$\dot{x}^*(\tau) = \frac{\partial H}{\partial \lambda}(x^*(\tau), u^*(\tau), \tau). \tag{2.16}$$

In this thesis, the functions are all independent of time. Then, from equations (2.13), (2.14) and (2.15), the optimal control Hamiltonian will be constant with value

$$H^*(\tau) = -\frac{\partial E}{\partial \tau}(\tau_f), \forall \tau \in [\tau_0, \tau_f]. \tag{2.17}$$

The constancy of the Hamiltonian in (2.17) will be very useful to provide a check of optimality of a given solution.

2.1.2 Feasibility and Optimality of the Solution

The above sub-section shows the necessary conditions for the solution of the problem to be optimal. In this sub-section we see the methods of testing the feasibility and checking the optimality of the solution.

Feasibility

A solution is said to be feasible if it satisfies the system's dynamic equation, as well as the constraints. In the pseudo-spectral method, both the state and control trajectories are discretized, and a finite-dimensional problem approximating the original continuous optimal control problem is solved. Of primary concern for the feasibility of the solution is that the obtained discretized solution approximates a continuous-time solution that satisfies the system dynamics. To check feasibility then, the original state equations are numerically integrated (using a 4^{th} order Runge-Kutta (RK) scheme) using the obtained discretized control solution as input. The resulting state trajectory is then compared to the discretized state trajectory. If they match to within a given tolerance, the solution is considered to be feasible. This idea is illustrated in Figure 2.1. Satisfaction of the constraints can be checked directly from the discrete pseudospectral solution.

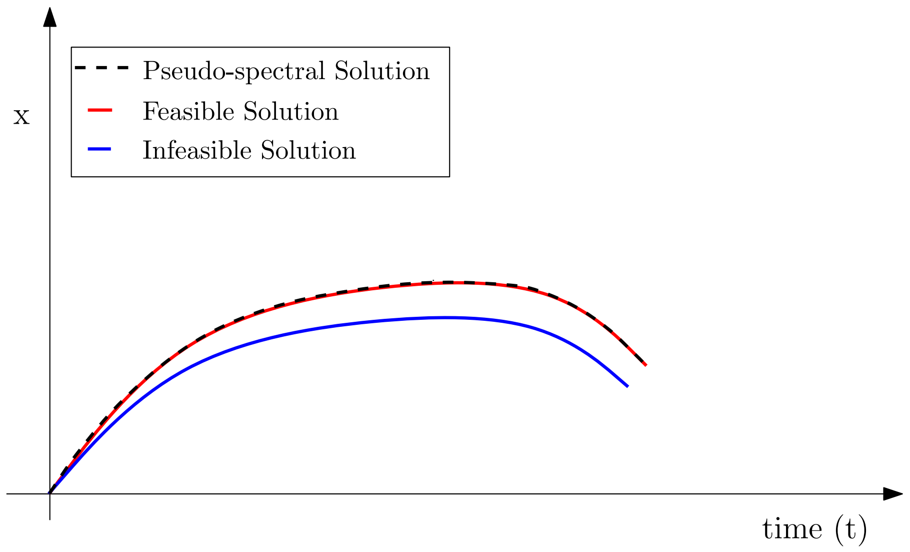


Figure 2.1: Feasibility test: pseudo-spectral solution state history, Propagated state and infeasible propagation

Optimality

In general, there is no test that can readily be performed to guarantee optimality of the obtained solution. However, there are some tests that can be performed, and can give a good indication of whether or not a solution can be trusted. The first test has already been identified in the statement of Pontryagin's minimum principle in the previous section. Namely, the Hamiltonian must be constant, with a value specified by the terminal cost. An additional test that can be performed is a check of Bellman's principle of optimality. This can be performed regardless of whether or not an obstacle field is stationary. Bellman's principle of optimality states: let there be an optimal trajectory from point A to B and consider any point C on that trajectory between points A and B. Then the optimal trajectory generated with same initial condition at A from A to C and the optimal trajectory generated with initial condition of C from C to B are equivalent to original optimal trajectory from A to B. In particular, the cost of optimal trajectory from A to B is equal to the sum of the optimal cost from A to C and C to B. Figure 2.2 shows the idea of Bellman's Principle test.

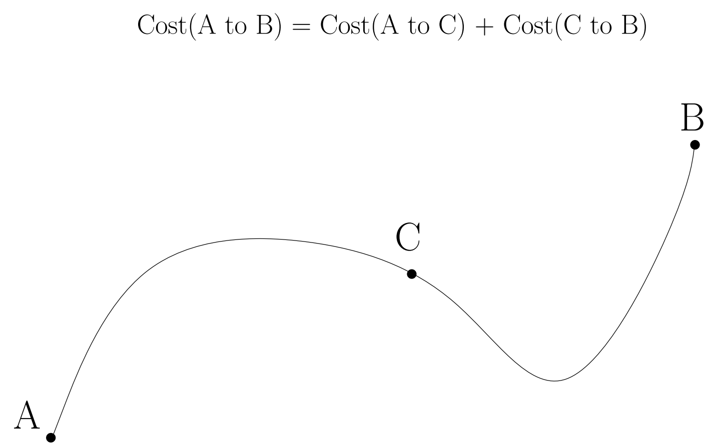


Figure 2.2: Bellman's Principle test of optimality

2.2 Legendre Pseudospectral Discretization

This section introduces the basic idea behind the theory of Pseudospectral discretization. Further details can be found in [55] and [56].

The key steps for pseudospectral discretization are contained in the following four points:

1. Select the discretization points (also known as grid or node),
2. Approximate the state and control functions by polynomials,
3. Approximate the differential equation (derivative of approximating polynomial),
4. Approximate the cost function and constraints.

These steps result in a finite-dimensional constrained optimization problem which can be solved by standard NLP solvers. The above steps are now elaborated on in some detail.

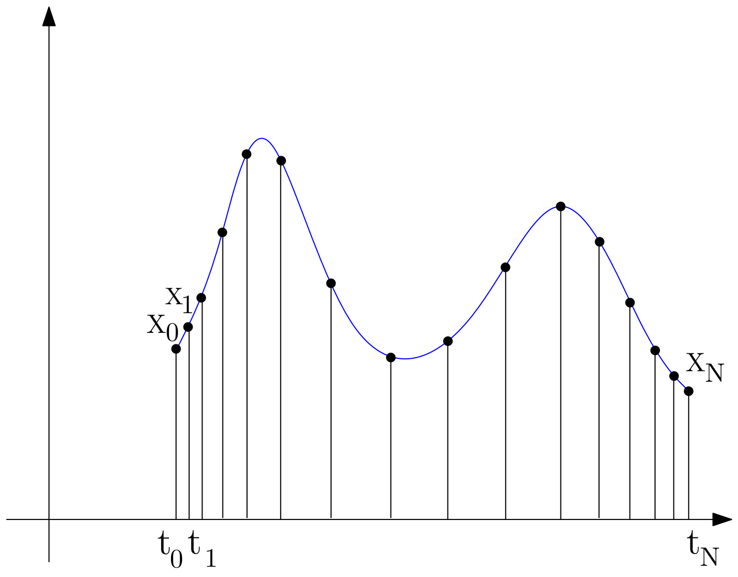
A. Node Spacing

Node selection is an important step for optimal control problems by discretization. Approximation theory suggests that optimal node placement occurs when the nodes are the roots of orthogonal polynomials such as Legendre or Chebyshev (also known as Tschebyscheff) polynomials [55][56]. One such set of points is the set of Legendre-Gauss-Lobatto (LGL) points [57]. The Legendre-Gauss-Lobatto points have very important characteristics which "concentrate" nodes at end points and spread out nodes in the middle of the domain. Figure 2.3 shows the polynomial approximation of $x(t)$ associated with Legendre-Gauss-Lobatto node distribution. The Legendre-Gauss-Lobatto points are defined on the interval $[-1, 1]$ as follows:

t_0 = -1

t_j = roots of $L'_N(t)$ $(j = 1.., N-1)$

$t_N = 1$



LGL node Points

Figure 2.3: Polynomial approximation of $x(t)$ associated with Legendre-Gauss-Lobatto node distribution

where *N+1* is the number of Legendre-Gauss-Lobatto points, and $L_i(t)$ denotes the Legendre polynomial of order i, for *i=1,...,N*. These points must be computed numerically. However, for a fixed number of nodes, they can be computed off-line beforehand and stored on-board for use in a real-time implementation. Since the Legendre-Gauss-Lobatto points are defined on the interval $[-1, 1]$, a mapping is needed from the interval of definition of the optimal control problem $[\tau_0, \tau_f]$ to $[-1, 1]$ and vice versa. The former mapping is given by equation (2.18)[58].

$$t_{LGL_i} = \frac{2(\tau_i - \tau_0) - (\tau_f - \tau_0)}{\tau_f - \tau_0} \tag{2.18}$$

where

t_{LGL_i}= the i^{th} Legendre-Gauss-Lobatto time point

$\tau_i =$ the i^{th} real time point

$\tau_0 =$ the initial real time point

$\tau_f =$ the final real time point

Conversely, the Legendre-Gauss-Lobatto time domain can be mapped to the real time domain given by

$$\tau_i = \frac{(\tau_f - \tau_0)t_{LGL_i} + (\tau_f + \tau_0)}{2} \tag{2.19}$$

Furthermore, this mapping has derivative

$$\frac{d\tau}{dt_{LGL}} = \frac{\tau_f - \tau_0}{2} \tag{2.20}$$

Equations (2.19) and (2.20) are used in the discretization of the Optimal Control Problem. After the node spacing is chosen, the problem is set, and this spacing defines the accuracy of the solution.

B. Approximation of the States and Controls

Given the N+1 Legendre-Gauss-Lobatto points, the state $x(t)$ and control $u(t)$ are approximated by N^{th} order polynomials as described in this section. First, the Lagrange interpolating polynomials of order $N+1$ are for j=0,...,N as

$$\phi_j(t) = \prod_{\substack{m=0 \\ m \neq j}}^{N} \frac{(t - t_m)}{(t_j - t_m)} \tag{2.21}$$

As each Legendre-Gauss-Lobatto point, t_i, the numerator of ϕ_j is zero except when $i = j$. The denominator normalizes the value of $\phi_j(t)$ to 1 when $i = j$. So, the following is the result at the node points:

$$\phi_j(t_i) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{2.22}$$

where δ_{ij} is the Kronecker delta.

Having defined the Lagrange interpolating polynomials, the $N+1^{th}$ order polynomial approximations for the state and control are given by

$$x^{N+1}(t) \quad := \quad \sum_{j=0}^{N} x(t_j)\phi_j(t) \tag{2.23}$$

$$u^{N+1}(t) \quad := \quad \sum_{j=0}^{N} u(t_j)\phi_j(t) \tag{2.24}$$

C. Approximation of the Differential Equation

After approximating state and control with $(N+1)^{th}$degree polynomials, we approximate their derivatives at the node points. The derivative of the approximation in (2.23) is given by

$$\dot{x}^N(t) \quad := \quad \sum_{j=0}^{N} x(t_j)\dot{\phi}_j(t) \tag{2.25}$$

For the discretization, we only require derivatives at the node points. So,

$$\dot{x}^N(t_i) \quad = \quad \sum_{j=0}^{N} x_j\dot{\phi}_j(t_i) \quad = \quad \sum_{j=0}^{N} D_{ij}x_j \tag{2.26}$$

where

$$D_{ij} = \dot{\phi}_j(t_i) \tag{2.27}$$

and

$$x_{j=}x(t_j)$$

Equation (2.26) can now be written in matrix-vector form.

$$\dot{\vec{x}}^N \quad = \quad D_N\vec{x}^N \tag{2.28}$$

where,

$x^N = \{x_0, x_1, x_2, ...x_N\}$, at the Legendre-Gauss-Lobatto node points

$$D_N = \begin{bmatrix} D_{00} & D_{01} & \dots & \dots & D_{0N} \\ D_{10} & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ D_{N0} & \dots & \dots & \dots & D_{NN} \end{bmatrix}, N+1 \times N+1 \text{ Pseudospectral differentiation}$$
matrix

Thus, differentiation is approximated by matrix multiplication. From reference [58], the elements of D$_N$ matrix is given by equation (2.29):

$$D_{ij} \quad = \quad \begin{cases} \frac{L_N(t_i)}{L_N(t_j)} \cdot \frac{1}{(t_i - t_j)} & i \neq j \\ -\frac{N(N+1)}{4} & i = j = 1 \\ \frac{N(N+1)}{4} & i = j = N \\ 0 & otherwise \end{cases} \tag{2.29}$$

Utilizing the time transformation in equation (2.20), the differential equation in (2.2), with substitution of equation (2.26) is approximated at the Legendre-Gauss-Lobatto points as :

$$\frac{\tau_f - \tau_0}{2} \left(\sum_{j=0}^{N} D_{ij} x_j \right) = f(x_i, u_i, \tau_i) \tag{2.30}$$

where

$$u_i = u(t_i)$$

for $i = 0, ..., N$

D. Approximate the Cost Function and Constraints

Given the polynomial approximations for $x(t)$ and $u(t)$ in equations (2.23) and (2.24), the cost function in (2.1) is first approximated as

$$J^N := E(\tau_f) + \int_{\tau_0}^{\tau_f} F(x^N(\tau), u^N(\tau), \tau) d\tau \tag{2.31}$$

First, the physical time domain $[\tau_0, \tau_f]$ needs to be mapped to the time-domain $[-1, 1]$ according to (2.19) and (2.20), yielding

$$\int_{\tau_0}^{\tau_f} f(\tau)d\tau = \frac{\tau_f - \tau_0}{2}\int_{-1}^{1} f(\tau(t))dt \tag{2.32}$$

Now, the cost function in (2.31) may be discretized. In general, Gaussian quadrature rules are expressed as [59]:

$$\int_{-1}^{1} f(t)dt \approx \sum_{j=0}^{N} w_j f(t_j) \tag{2.33}$$

where w_j are the weights associated with the node spacing scheme.

For the Legendre-Gauss-Lobatto node-spacing scheme, the cost function is discretized as

$$J^N := E(\tau_f) + \frac{\tau_f - \tau_0}{2}\sum_{j=0}^{N} F(x^N(\tau_j), u^N(\tau_j), \tau_j)w_j \tag{2.34}$$

Finally, the path and control constraints in (2.2) are enforced at the Legendre-Gauss-Lobatto node points and the overall discretized optimization problem becomes

$$J^N := E(\tau_f) + \frac{\tau_f - \tau_0}{2}\sum_{j=0}^{N} F(x_j, u_j, \tau_j)w_j$$

$$\begin{array}{ccl} \frac{\tau_f-\tau_0}{2}\left(\sum\limits_{j=0}^{N} D_{ij}x_j\right) & = & f(x_i, u_i, \tau_i), \\ 0 & \leq & g(x_i, u_i, \tau_i), \\ 0 & \leq & h(x_i, \tau_i), \\ x(\tau_0) & = & x_0, \\ x(\tau_f) & = & x_N. \end{array} \tag{2.35}$$

After discretization a nonlinear programming solver such as SNOPT [60] is used to solve the problem. As mentioned previously, the software package DIDO performs the aforementioned discretization, and contains the solver SNOPT. For convenience, DIDO is used to solve the optimal control problems posed in this thesis.

2.3 Real-time Trajectory Planning

The advancement in computer technology and numerical problem solving has brought a turning point in the use of optimal control in nonlinear feedback control laws. Pseudospectral optimal control laws have been successfully implemented in feedback form in a number of guidance, navigation and control problems, including observers for nonlinear systems [61], nonlinear feedback guidance of Reentry Vehicles [62], autonomous trajectory planning [49] and other evolving implementations. The use of this idea in feedback control for realtime trajectory generation is introduced below. In fact, this section summarizes the concept of realtime trajectory generation used in this thesis.

Feedback control implies that the control system has regularly updated (ideally continuous) information related to the system's states. In the case of state feedback (as in this thesis), this information consists of the states themselves. The concept of closed-loop control using the pseudospectral-optimal control method (PS-OCM) is based in the idea that the open-loop control, can be recomputed very quickly using current state information such that it is equivalent to continuous feedback. This has been done by the introduction of "$C^1 - Caratheadory$" conditions [63]. In [63], it was proved that the concept is possible if the computation time for an open-loop solution is very small.

Hence, successful implementation of feedback on PS-OCM relies on the fact that the open-loop control be generated fast enough. References [64],[44],[65] show that the pseudospectral-method is capable of generating open-loop controls in fractions of a second even in legacy hardware running MATLAB.

PS-Feedback Implementation

Figure 2.4 shows the pseudospectral-feedback concept utilized in this thesis.

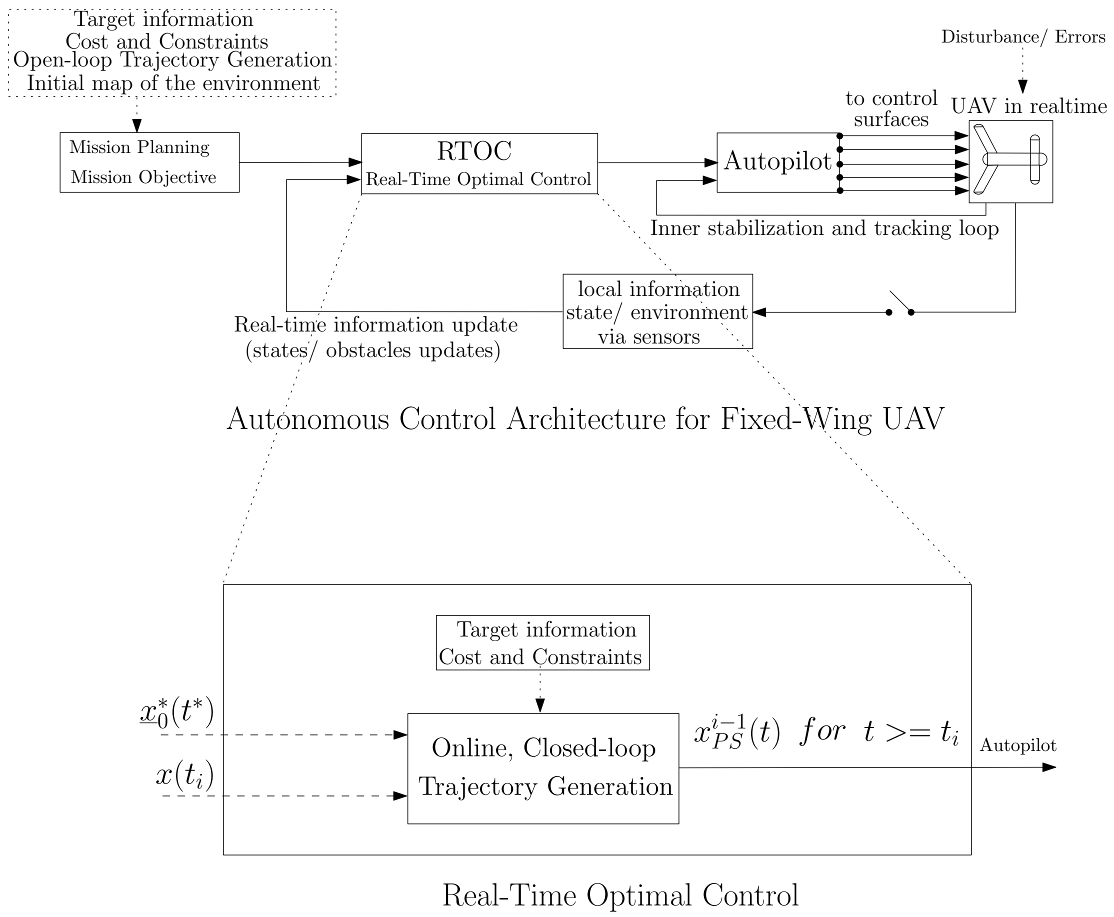

Target information
Cost and Constraints
Open-loop Trajectory Generation
Initial map of the environment
Mission Planning
Mission Objective
RTOC
Real-Time Optimal Control
Autopilot
to control surfaces
UAV in realtime
Disturbance/ Errors
Inner stabilization and tracking loop
local information state/ environment via sensors
Real-time information update (states/ obstacles updates)
Autonomous Control Architecture for Fixed-Wing UAV
Target information
Cost and Constraints
$\underline{x}_0^*(t^*)$
$x(t_i)$
Online, Closed-loop Trajectory Generation
$x_{PS}^{i-1}(t)\ \ for\ \ t >= t_i$
Autopilot
Real-Time Optimal Control

Figure 2.4: Control Architecture for Online Trajectory Generation

Information such as the mission objective, initial terrain and environment maps, cost, constraints, vehicle dynamics, initial and final conditions are given to the mission planner, which generates a mission trajectory offline. This trajectory provides a nominal flight path that needs to be stayed as close to as possible. That is, it is a target trajectory.

As shown in Figure 2.4, the online trajectory generator uses the current state information as initial condition to generate a new optimal trajectory using the pseudospectral method and the available obstacle information. This trajectory is then sent as a command to the autopilot to track. The optimal trajectory is regenerated at each sample instant using new state and obstacle information. In this way, the scheme is robust to changes in obstacle information, as well as disturbances acting on the UAV. Due to the finite amount of time taken to compute each new trajectory, the trajectory command sent to the autopilot is based upon the trajectory generated using state information from one previous sample time. This can be thought of as a predictor scheme to correct for process delays. Figure 2.5 shows the successive process of solving the OCP. In the figure, OC1 means ONLINE CONTROL 1 and so on. In theory, each initial condition (IC) will begin at final condition of the previous trajectory segment.

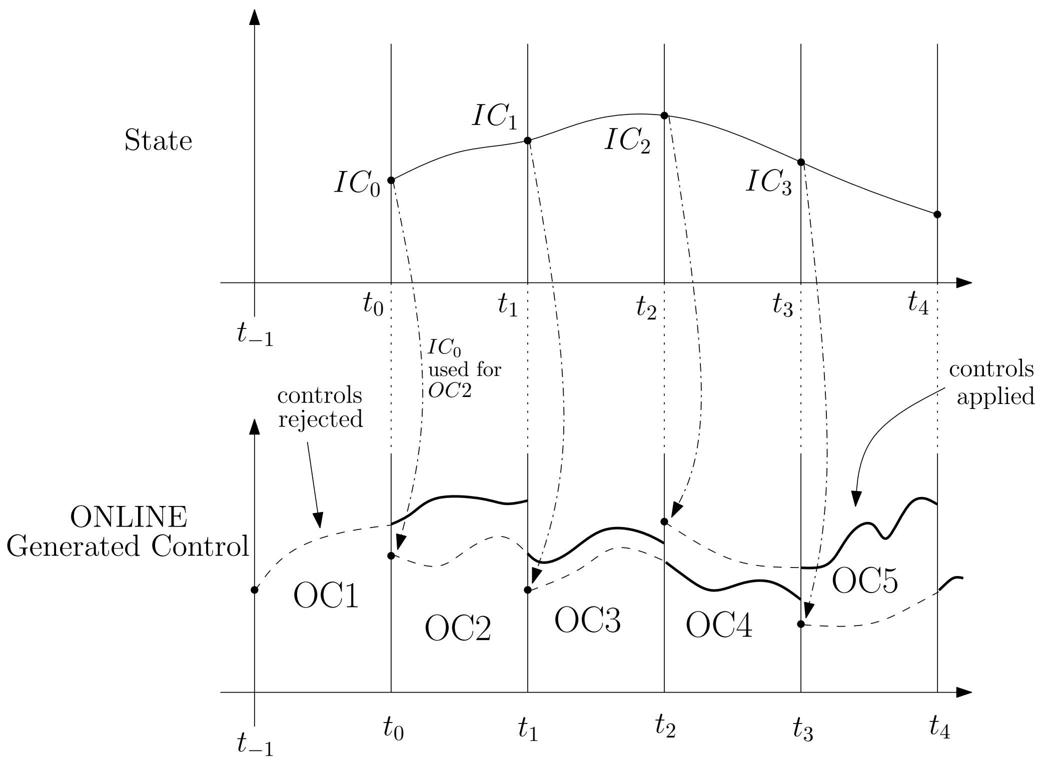


Figure 2.5: Online/Realtime Propagation

Bellman's principle provides an advantage for online trajectory generation using pseudospectral methods, in that a low accuracy solution can be de-aliased to provide a high-accuracy solution without creating a large-scale optimization problem [49],[66]. Reference [49] shows an example where an online-generated trajectory using a 15 node solution at each sample time, yields an accuracy equivalent to an off-line generated 60 node solution. This is because the Legendre-Gauss-Lobatto points are concentrated at the end points of each time interval under consideration. Thus, the accuracy of the solution is higher at the end points. In real-time implementation, only the higher accuracy initial portion of each generated control is used, and then a new control is generated. Along with that this method provides other advantages such as avoiding the need of a disturbance rejection system and no prior knowledge of the environment is required. De-aliasing Bellman (a^2B)algorithm applied in this thesis is stated below.

a^2B **algorithm: [66]**

1. *Solve the problem for a low number of nodes, n. This generates a discrete-time solution,* $\{x_i, u_i\}_{i=0}^{n}$ *corresponding to discrete time* $\{t_i\}_{i=0}^{n}$.

2. *Partition the time interval [t_0, t_n] into N_B Bellman segments, $t_0 < t^1 < \cdots < t^{N_B} = t_n$. These segments need not be uniformly spaced.*

3. *Propagate the differential equation from t_0 to t^1 using x_0 as the initial condition and any method of continuous-time reconstruction of the controls, $u^1(t)$, $t \in [t_0, t^1]$ based on $\{u_i\}_{i=0}^{n}$. That is, solve the initial value problem,*

$$\dot{x} = f(x, u^1(t)), \qquad x(t_0) = x_0 \tag{2.36}$$

This step generates a continuous-time trajectory, $x^1(t)$, $t \in [t_0, t^1]$. This propagation is done numerically via some high-precision propagator, say the standard 4/5 Runge-Kutta Method.

4. *Set $x_0 = x^1(t^1)$ and $t_0 = t^1$ and go to step 1; that is, set a new initial condition as the value of the integrated state at the end of the period $[t_0, t^1]$ and solve the problem again for n (which continues to be low). This generates a new sequence $\{x_i, u_i\}_{i=0}^{n}$ corresponding to new discrete times $\{t_i\}_{i=0}^{n}$, etc.*

5. *The algorithm stops at the N_Bth sequence when the final conditions are met. The candidate optimal trajectory is given by the Bellman chain $\big\{x^1(t), t \in [t_0, t^1]; x^2(t), t \in [t^1, t^2]; \cdot$ $x_B(t), t \in [t_0, t_f]$. Similarly, the corresponding controls are given by*

$$\big\{u^1(t), t \in [t_0, t^1]; u^2(t), t \in [t^1, t^2]; \cdots u^{N_B}(t), t \in [t^{N_B-1}, t^{N_B}]\big\} \coloneqq u_B(t), t \in [t_{0,}t_f].$$

Hence this thesis uses the above concept for realtime autonomous trajectory generation to be implemented on UAVs and its analysis in Chapter 4.

Chapter 3

Problem Formulation and Vehicle Dynamics

3.1 Introduction to Autonomous Control Architecture

This chapter includes the introduction to the thesis problem, the detailed problem formulation and the steps chosen for solution. The concept of realtime optimal control has been explained in general terms in Chapter 2. This chapter further explores the concept and specializes it for autonomous realtime trajectory generation to be used on the UAV, which has been introduced in Chapter 1.

The UAV is being designed to autonomously survey large poorly known areas. While performing a geo-physical survey of an area, the UAV is required to fly at a minimal altitude to maximize the survey resolution (50 meters above the ground level [9]). A typical survey flight path is made by connecting lines as shown in Figure 3.1. At the low flight altitude, these paths can have a number of natural (small hills or tall trees) or artificial obstacles (power lines, communication towers) on its path. Figure 3.1 shows some examples. In order to avoid these obstacles in autonomous flight mode, the UAV must have an obstacle detection and avoidance feature. This thesis focuses on the obstacle avoidance capability once the obstacles have been detected. The obstacle detection capability is beyond the scope of this thesis. It is assumed that the obstacle detection system is available providing current obstacles' sizes, shapes and positions at regular sample intervals

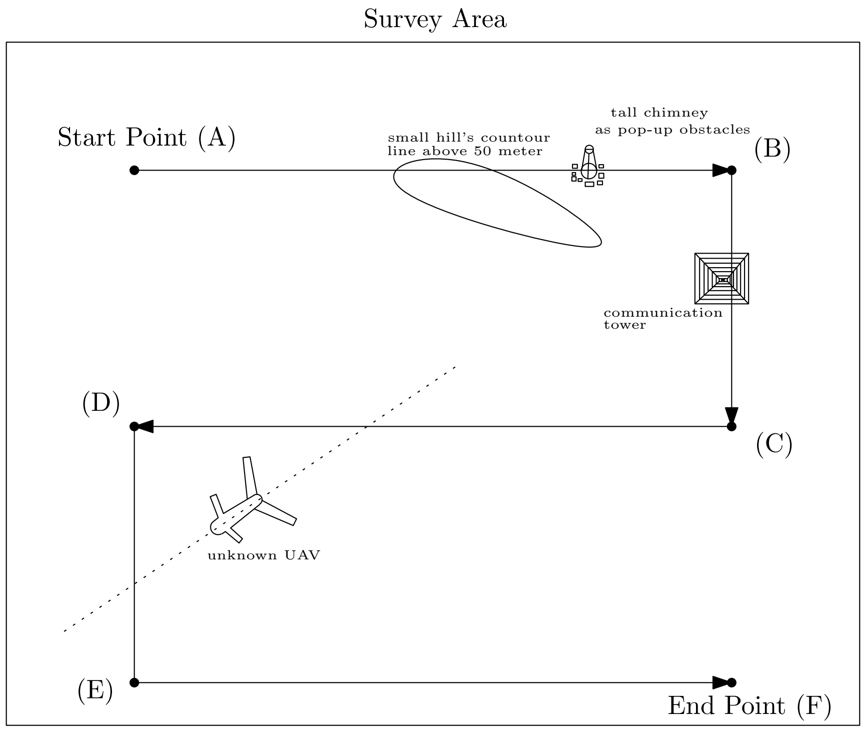


Figure 3.1: Aerial Path Designed for UAV.

The UAV under consideration this thesis is the GeoSurv II prototype, as described in Chapter 1 and once again shown in Figure 3.2 with the control surfaces indicated. The UAV with a wingspan of 4.9m, has three directional control surfaces; Aileron, Elevator and Rudder, and a Flap which is a high lift device. Overall, the aircraft has five control parameters, namely Aileron, Elevator, Rudder, Flap and the throttle. However, for the purpose of this thesis, it is assumed that the UAV is equipped with flight control laws capable of tracking a commanded trajectory. That is, it is assumed that the inner loop in Figure 2.4 has already been closed. The focus of this thesis is purely on the real-time trajectory generation for obstacle avoidance.

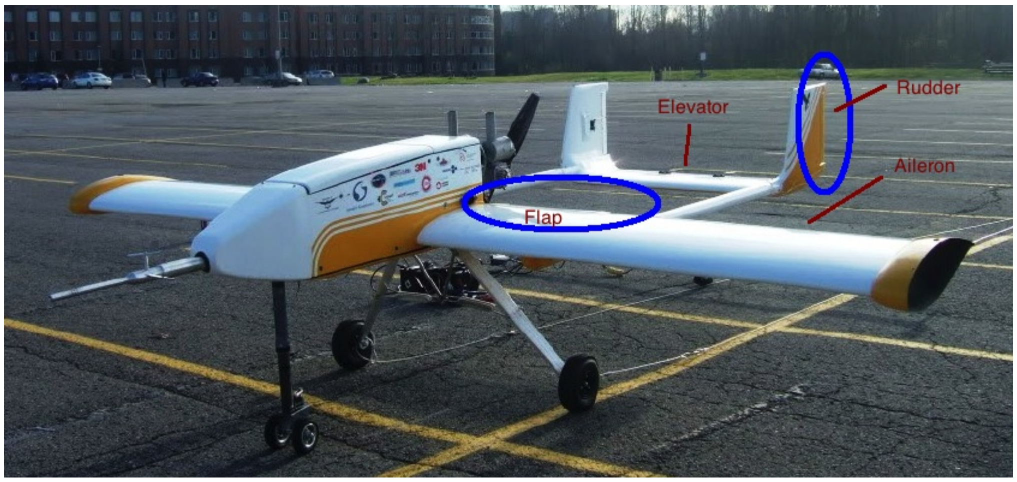


Figure 3.2: GeoSurv II Prototype with labeled control parts.

Referring to Figure 3.1, to minimize gaps in the aerial survey, the UAV is desired to stay as close as possible to the predefined path, while avoiding any encountered obstacles. With this in mind, the autonomous control architecture is developed as shown in Figure 2.4. The mission planner executes each line segment one by one. According to the control architecture shown in Figure 2.4, the mission planner creates the UAV's mission from initial point to final point of flight. Figure 3.1 shows only a typical flight plan over the survey area but in reality, the UAV takes off from a given base and flies to the survey area. It then completes the survey and flies back to base and lands. The entire flight plan is stored in the mission planning block. The mission planner keeps track of position and possibly other required states, for the duration of the mission. Each line segment in the nominal flight plan is passed from the mission planner block to the Real Time Optimal Control (RTOC) block for realtime trajectory generation, which will be expanded upon shortly. The RTOC generated trajectory is sent to the autopilot block as a commanded trajectory for the UAV to follow. As shown in Figure 2.4, the RTOC regularly updates the commanded trajectory based upon new state and obstacle information.

The obstacle avoidance method explained above needs to be tested in simulation before it can be implemented on a real aircraft. This thesis presents several simulation results for obstacle avoidance in different scenarios that may be encountered in a typical survey. The remainder of this chapter contains the details of the mathematical problem formulation.

3.2 Vehicle Kinematics

So as to avoid the need for a dynamic model of the UAV, the RTOC trajectory planning works with a kinematic model of the UAV. As illustrated in Figure 3.3, the UAV kinematics can be purely represented in terms of two angles (ξ and γ) and the UAV ground speed (V), which together specify the UAV velocity vector. The angles ξ and γ physically represent the vehicle heading and climb angles, respectively.

Representing the position of the UAV by the cartesian coordinates $[x, y, z]^T$, the UAV kinematics are given by

$$\dot{x} = V cos(\gamma) cos(\xi) \tag{3.1}$$

$$\dot{y} = V cos(\gamma) sin(\xi) \tag{3.2}$$

$$\dot{z} = V sin\gamma \tag{3.3}$$

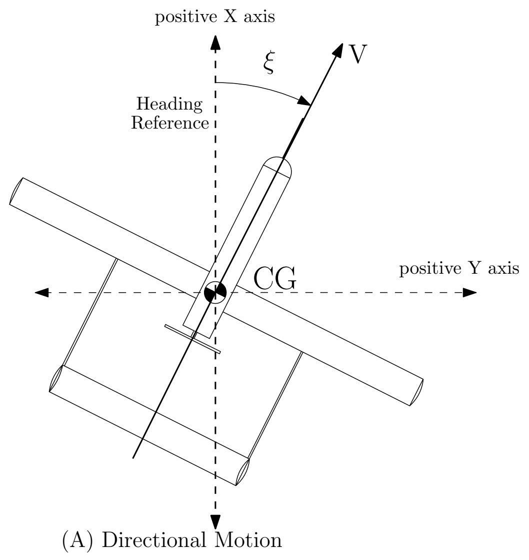
As can be seen from equations (3.1) to (3.3), the kinematics are completely governed by ξ, γ and V. As such, they can be considered to be control inputs to the kinematics. However, the RTOC generated trajectory must be feasible for a UAV to fly. Therefore, limitations must be placed upon these variables as follows:

a. There is no limitation on the heading, ξ, but its rate of change will be limited, as discussed in below.

b. The climb angle, γ, must be limited. For example, it is impossible for GeoSurv II to fly vertically upwards. Therefore, we restrict γ to the range $\gamma_{min} \leq \gamma \leq \gamma_{max}$. In addition, the rate of change of γ (the pull-up rate) will be limited, as discussed in below. For simplicity, it is assumed that the γ_{max} and γ_{min} are independent of speed.

c. The UAV's speed is limited below and above by the stall speed and the maximum speed allowed by the airframe, respectively. Consequently, we restrict $V_{min} \leq V \leq V_{max}$. In addition, the rate of change of V (the acceleration) will be limited as discussed below.[1]

[1] Strictly speaking, the limits should be on the air speed, but we assume that the UAV will only operate in calm conditions, such that the air and ground speeds are similar.



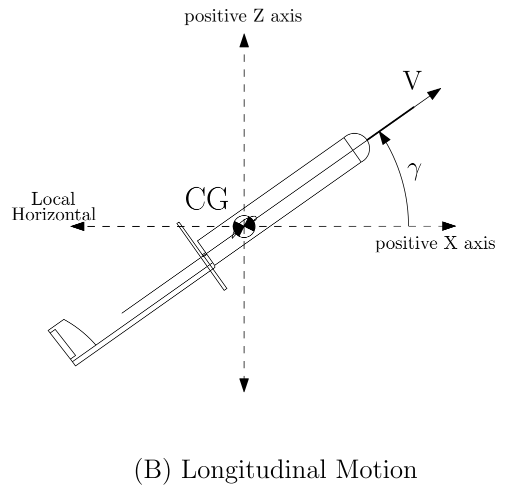


Figure 3.3: Flight Angles.

Let ω_γ, ω_ξ and a represent the rates of change of γ, ξ and V , respectively. As mentioned above, these rates need to be constrained. Consequently, the kinematic equations in (3.1) to (3.3) are augmented with the rates to obtain

$$\underline{\dot{X}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\gamma} \\ \dot{\xi} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} Vcos(\gamma)cos(\xi) \\ Vcos(\gamma)sin(\xi) \\ Vsin\gamma \\ \omega_\gamma \\ \omega_\xi \\ a \end{bmatrix} \subset \mathbb{R}^6 \tag{3.4}$$

Equation (3.4) contains the full set of state equations used for the purpose of trajectory planning. In particular, it is to be noted that γ, ξ, and V are now treated as states, while ω_γ, ω_ξ and a become the control variables.

The control variables, ω_γ and ω_ξ are limited by aircraft's maximum pull-up and turning rates, which will be discussed in Section 3.3, while the acceleration, a, is limited by the engine and airframe capabilities. Consequently, we define the control limits as.

$$C = \left\{ \begin{array}{lc} \omega_\gamma : & (\omega_\gamma)_{min} \le \omega_\gamma(t) \le (\omega_\gamma)_{max} \\ \omega_\xi : & (\omega_\xi)_{min} \le \omega_\xi(t) \le (\omega_\xi)_{max} \\ a : & a_{min} \le a(t) \le a_{max} \end{array} \right\} \subset \mathbb{R}^3 \tag{3.5}$$

3.3 Constraints and Cost

In this section, the constraints for the trajectory optimization problem are fully developed, and a cost function is formulated appropriate to the control objective. In optimal control, constraints can be of different types according to the nature of the problem under consideration. For the obstacle avoidance problem in this thesis we have two kinds of constraints: UAV constraints and obstacle constraints. UAV constraints arise due to UAV limitations, while obstacle constraints arise due to the presence of the obstacles. UAV constraints are typically a mix of path and control constraints, while obstacle constraints are purely path constraints.

UAV Constraints

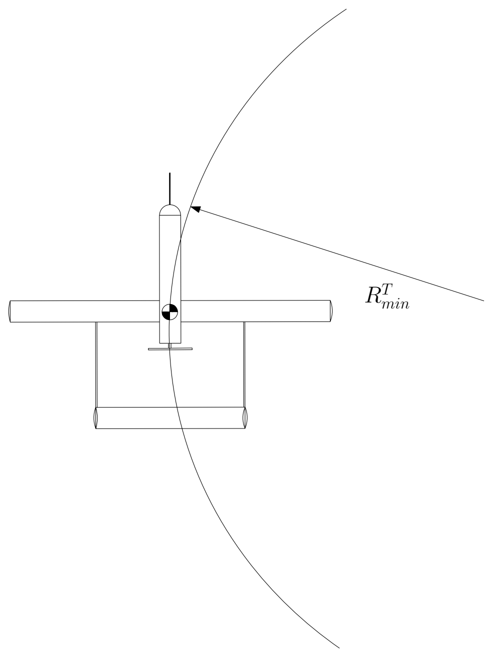
The aircraft dynamics themselves place constraints on the aircraft's maneuvering capabilities. At the fundamental level, these constraints result from the aircraft aerodynamics and limitations on the throttle and control surface deflections. However, since we do not wish to deal with the aircraft dynamics directly, we must instead incorporate these constraints into the trajectory planning process. Two of the resulting constraints have already been addressed, namely path angle and velocity constraints. In addition, as has been alluded to in equation (3.5) there are constraints on the minimum turning and pull-up rates, as explained in [9]. These translate directly into constraints on the rates of the control variables γ and ξ. It should be noted that the constraints on turning and pull-up rates are obtained under the assumption of steady-flight conditions. As such, they are somewhat conservative. However, they are simple to implement, and hence will be used.

From reference [9], the minimum turning radius is:

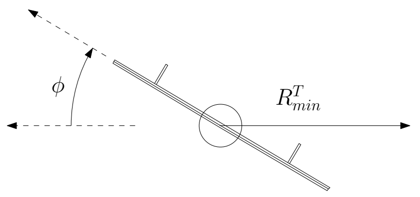
$$R^T_{min} = \frac{V^2}{g\sqrt{n^2 - 1}} \tag{3.6}$$

and the maximum turning rate is

$$\omega^T_{max} = \frac{g\sqrt{n^2 - 1}}{V} \tag{3.7}$$



(a) Top view of an UAV at tightest turn possible



(b) Front view of an UAV at tightest turn possible with ϕ as bank angle

Figure 3.4: Turning Flight

where,

g=9.81 m/s^2 is the Earth's gravitational acceleration

$n$is the load factor of the aircraft defined as

$$n = \frac{L}{W}, \tag{3.8}$$

L is lift produced (Newton)

W is the weight (Newton)

R^T_{min} is minimum turning radius in meter (meter)

ω^T_{max} is maximum rate of change of angle in turning (radian per second)

In case of turning flight, it can be shown that,

$$n = \frac{1}{cos(\phi)} \tag{3.9}$$

ϕ being the banking angle of the UAV.

From equation (3.6), it can be seen that if the load factor, n, is fixed, the minimum turning radius increases with aircraft speed. On the other hand, the load factor, n, also increases with aircraft speed (more lift can be generated at higher speeds). Since the load factor, n, appears in the denominator of (3.6), this mitigates the increase in the minimum turning radius to an extent. To avoid the need to know the relationship between the load factor and aircraft speed, we shall simply compute the minimum turning radius based upon the maximum aircraft speed, V_{max}, and keep it fixed for all aircraft speeds. To compute the maximum load factor, we shall take the maximum bank angle to be 65 [2]. Hence from equation (3.9), we get

$$n = 2.4 \tag{3.10}$$

Therefore, while turning with maximum velocity of V_{max} and load factor (n) of 2.4, from equation (3.6) we get the minimum turning radius,

$$R^T_{min} = \frac{V^2_{max}}{21.0} \tag{3.11}$$

Similarly, the minimum pull-up radius and maximum pull up rate can be calculated from [9]

$$R^p_{min} = \frac{V^2}{g(n-1)} \tag{3.12}$$

and

$$\omega^P_{max} = \frac{g(n-1)}{V} \tag{3.13}$$

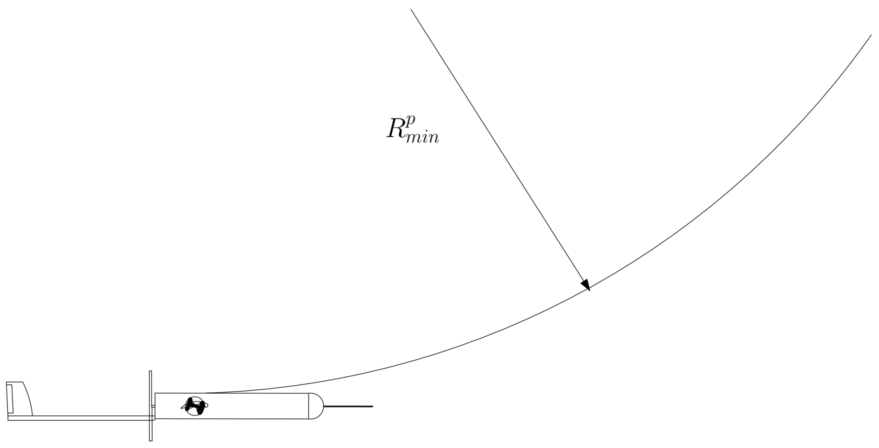


Figure 3.5: UAV at it's tightest pull-up turn possible

[2]Note that the actual GeoSurv II parameters are unknown so this bank angle is assumed as a place holder

where,

R^p_{min} is minimum pull-up radius

ω^P_{max} is maximum pull-up rate

For minimum pull-up radius, we take the value of $n = 2.4$ from equation (3.10) and with maximum velocity of V_{max}, equation (3.12) gives,

$$R^p_{min} = \frac{V^2_{max}}{13.4} \tag{3.14}$$

And, these two major concept of turning and pull-up radius of the UAV will be implemented as constraints of UAV system. These constraints are incorporated as path constraints. It is important that the generated path satisfies these constraints, such that it is feasible for the UAV to fly.

Assuming the pure turn, the turning radius R^T satisfies

$$\frac{1}{R^T} = \frac{\omega_\xi}{V} \tag{3.15}$$

Consequently, fixing the minimum turn radius obtained in (3.11), we obtain the constraint

$$-\frac{1}{R^T_{min}} \leq h(\omega_\xi, V) \leq \frac{1}{R^T_{min}} \tag{3.16}$$

where, $h(\omega, V) = \omega/V$ represents a path constraint as function of control and state.

Similarly, assuming a pure pull-up maneuver, the pull-up radius, R^P satisfies

$$\frac{1}{R^P} = \frac{\omega_\gamma}{V} \tag{3.17}$$

Consequently, fixing the minimum pull-up obtained in (3.14), we obtain the constraint

$$-\frac{1}{R^P_{min}} \leq h(\omega_\gamma, V) \leq \frac{1}{R^P_{min}} \tag{3.18}$$

and $h(\omega, V)$ is as defined previously. Note that for simplicity we assume same minimum push-down radius as pull-up radius, but this can be adjusted when more aircraft information is available.

Finally, the control limits given by equation (3.5) are updated as

$$C = \left\{ \begin{array}{cc} \omega_\gamma : & -\frac{1}{R^P_{min}} \leq h(\omega_\gamma, V) \leq \frac{1}{R^P_{min}} \\ \omega_\xi : & -\frac{1}{R^T_{min}} \leq h(\omega_\xi, V) \leq \frac{1}{R^T_{min}} \\ a : & a_{min} \leq a(t) \leq a_{max} \end{array} \right\} \subset \mathbb{R}^3 \tag{3.19}$$

Obstacle Constraints

Obstacles act as path constraints, and path constraints are generally continuous algebraic functions in states and possibly time. With this in mind, the p-norm was introduced in [35] for the purpose of modeling obstacles. The modeling of an obstacle's exact shape is complex and unnecessary, since for the purposes of path planning, we can bound each obstacle with simpler generic shapes such as spheres, cubes, or ellipsoids. Each of these shapes are readily obtained by use of an appropriate p-norm. Two-dimensional obstacles representable by a p-norm are described by equations of the form $h(x, y) = 0$, where.

$$h(x, y) = \left| \left(\frac{x - x_c}{a} \right)^p \right| + \left| \left(\frac{y - y_c}{b} \right)^p \right| - 1 \tag{3.20}$$

where,

(x_c, y_c) are the coordinates of the center of the obstacle

a is the half-length of the obstacle measured along the x axis

b is the half-length of the obstacle measured along the y axis

p is the degree of norm chosen,

Points strictly outside the obstacle are those satisfying $h(x, y) > 0$.

Figure (3.6) shows examples of a rectangle and circle, which are created using $p = 10$ and $p = 2$, respectively.

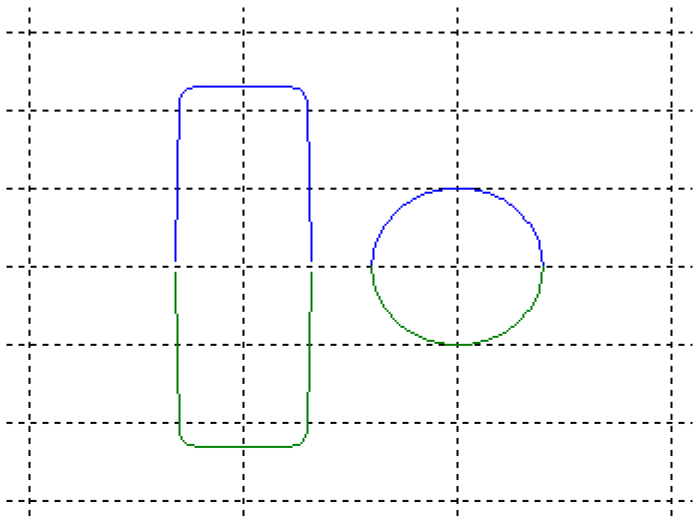


Figure 3.6: Rectangle and Circle are created by setting the exponent to 10 and 2 respectively.

Similarly for three dimensional space, obstacles can be defined as those points satisfying $h(x,y,z) = 0$, where

$$h(x,y,z) := \left|\left(\frac{x-x_c}{a}\right)^p\right| + \left|\left(\frac{y-y_c}{b}\right)^p\right| + \left|\left(\frac{z-z_c}{d}\right)^p\right| - 1 \tag{3.21}$$

with,

(x_c, y_c, z_c) the center of the obstacle

d is the half-length of the obstacle measured along the z axis.

and a, b and p have the same meaning as in the two-dimensional case.

As before, points strictly outside the obstacle satisfy $h(x,y,z) > 0$, which leads to a path constraint for the trajectory generation problem.

The path constraint corresponding to the i^{th} three-dimensional obstacle can be written equivalently as

$$h_i(x,y,z) := ln\left[\left|\left(\frac{x-x_{\{c,i\}}}{a_i}\right)^{p_i}\right| + \left|\left(\frac{y-y_{\{c,i\}}}{b_i}\right)^{p_i}\right| + \left|\left(\frac{z-z_{\{c,i\}}}{d_i}\right)^{p_i}\right|\right] > 0 \tag{3.22}$$

An example of a three-dimensional obstacle is shown in Figure 3.7. This obstacle was generated with the parameters:

$x_c = 30; \quad a = 10; \quad p = 100$

$y_c = 30; \quad b = 20;$

$z_c = 30; \quad d = 30;$

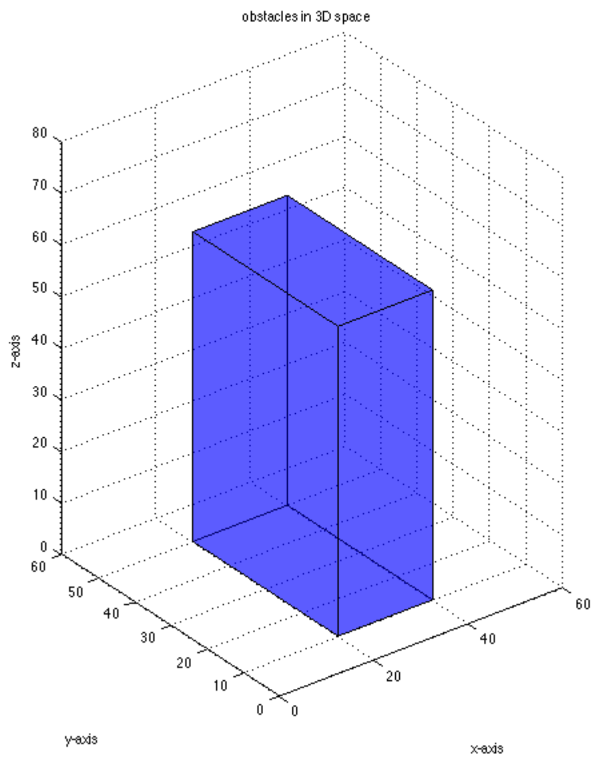


Figure 3.7: Box representing obstacle.

Cost

The cost function in an optimal control problem is defined to reflect the desired performance. As discussed previously, the objective of the control problem in this thesis is to stay as close as possible to the predefined path for the duration of the flight, while avoiding any encountered obstacles. This can be done by minimizing the area between the flight path and the predefined path as shown in Figure 3.8.

Each segment of the predefined path consists of a straight line at a fixed altitude. Therefore, there are two path deviations that need to be addressed, namely the horizontal and vertical deviations. An example is shown in Figure 3.8, where the predefined flight path is given by the equations $x = y$ and $z = 50$. Note that there is no loss in generality of this choice, since by a simple translation and rotation of coordinates, any straight horizontal flight path can be mapped to this line.

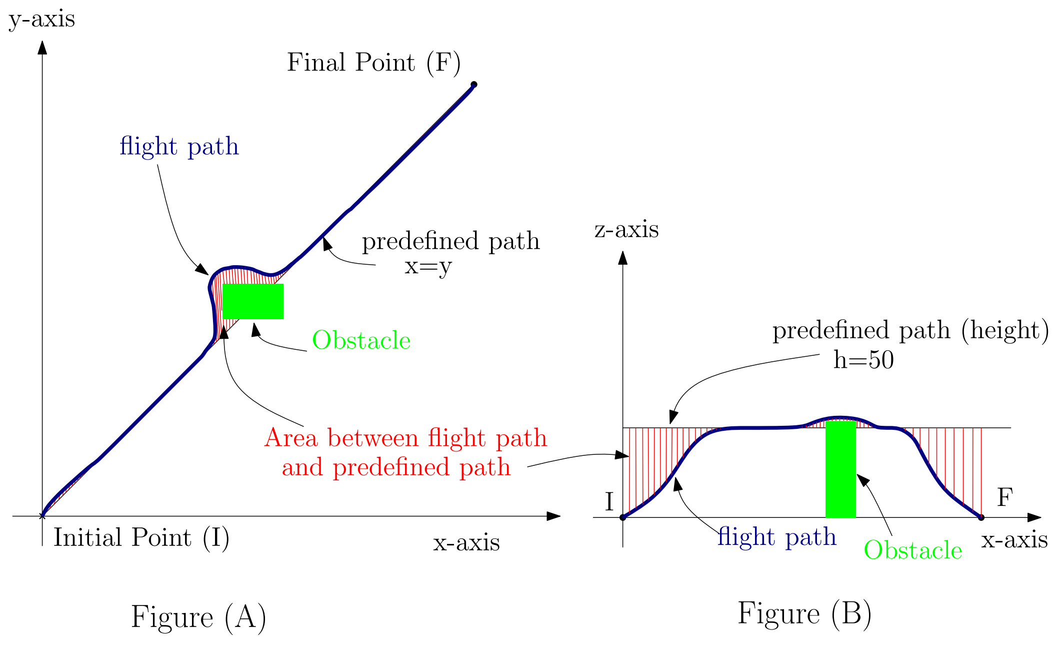


Figure 3.8: Figure showing the line $x = y$ and $h = 50$ with respective flight path and area between them.

Corresponding to the desire to minimize the area between the flight path and the predefined path $(x = y, z = 50)$, two integrals are defined

$$A1 = \int_{t_0}^{t_f} 0.5(x-y)^2 \, dt \tag{3.23}$$

and

$$A2 = \int_{t_0}^{t_f} 0.5 \, (z-50)^2 dt \tag{3.24}$$

These two integrals will form part of the cost function.

For added robustness, the obstacles are additionally penalized within the cost function. This is accomplished by the addition of a robustness function $r(x, y, z)$, given by [49]

$$r(x, y, z) = w \sum_{i=1}^{n} (e^{e^{-h_i(x,y,z)}} - 1) \tag{3.25}$$

where $w > 0$ is a weighting factor. Clearly, from (3.25), $r(x, y, z)$ increases as (x, y, z) approach an obstacle.

Finally we can assemble the cost function, in terms of an end point cost (t_f) and a running cost $(r(x, y, z) + A1 + A2)$ in the form of equation (2.1).

$$J(\underline{X}(\cdot), \underline{U}(\cdot), t_f) = t_f + A1 + A2 + \int_{t_0}^{t_f} [r(x, y, z)]dt \tag{3.26}$$

where

$$U = [\omega_\gamma, \omega_\xi, a]^T$$

Note that the inclusion of the final time t_f in the cost function is optional, and reflects whether or not it is desired to minimize the final time as well as the deviation from the predefined trajectory.

3.4 Problem Summary

Combining all of the developments so far, the trajectory optimization problem can be posed as the following optimal control problem, which is of the same form as that given in equations (2.1) and (2.2) in Chapter 2.

$$\begin{cases} \underline{Minimize:} & & J(\underline{X}(\cdot),\underline{U}(\cdot),t_f) = t_f + A1 + A2 + \int_{t_0}^{t_f}[r(x,y,z)]\,dt \\ & where, & U = [\omega_\gamma, \omega_\xi, a]^T \\ \underline{Subject} \quad \underline{to}: & & \underline{\dot{X}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\gamma} \\ \dot{\xi} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} Vcos(\gamma)cos(\xi) \\ Vcos(\gamma)sin(\xi) \\ Vsin\gamma \\ \omega_\gamma \\ \omega_\xi \\ a \end{bmatrix} \\ & and & \underline{X}(t_0) = X_0 \\ & & t_0 = 0 \\ & & \underline{X}(t_f) = X_f \\ & & x(t) \in \mathbb{R} \\ & & y(t) \in \mathbb{R} \\ & & z(t) \in \mathbb{R} \\ & & \gamma_{min} \le \gamma(t) \le \gamma_{max} \\ & & \xi(t) \in \mathbb{R} \\ & & V_{min} \le V(t) \le V_{max} \\ & & a_{min} \le a(t) \le a_{max} \\ & & -\frac{1}{R^P_{min}} \le h(\omega_\gamma, V) \le \frac{1}{R^P_{min}} \\ & & -\frac{1}{R^T_{min}} \le h(\omega_\xi, V) \le \frac{1}{R^T_{min}} \\ & & h_i(x,y,z) > 0 \\ & where, & i = 1...n \end{cases} \tag{3.27}$$

The specific parameters used in the thesis for the GeoSurv II problem are:

$$\begin{aligned} x_{min} &= 0m & x_{max} &= 1200m \\ y_{min} &= 0m & y_{max} &= 1200m \\ z_{min} &= 0m & z_{max} &= 200m \\ \gamma_{min} &= -45 & \gamma_{max} &= 45 \\ \xi_{min} &= -180 & \xi_{max} &= 180 \\ V_{min} &= 15m/s & V_{max} &= 30m/s \\ a_{min} &= -3m/s^2 & a_{max} &= 3m/s^2 \\ R^T_{min} &= \frac{V^2_{max}}{21.0375} = 43m & R^p_{min} &= \frac{V^2_{max}}{13.4024} = 67m \end{aligned} \tag{3.28}$$

and from (3.28) turning and pull up constraints becomes,

$$\begin{aligned} -1.4925m^{-1} \leq h(\omega_\gamma, V) \leq 1.4925m^{-1} \\ -2.3256m^{-1} \leq h(\omega_\xi, V) \leq 2.3256m^{-1} \end{aligned} \tag{3.29}$$

The remainder of this section presents a brief overview of different obstacle scenarios that will be considered in this thesis. Figure 3.9 shows different types of obstacles at different strategic positions with all dimensional units in meters (m). Three different types of obstacles, namely stationary, pop-up and moving obstacles, will be used to analyze the capability of RTOC using the Legendre Pseudospectral method for autonomous trajectory generation for fixed-wing UAVs such as GeoSurv II. As stated in Chapter 2, the software package DIDO will be used to perform the analysis. From Figure 3.9, obstacles placed at different strategic positions will create different scenarios for the UAV to perform. The straight line joining points $(0, 0, 50)$ m and $(1000, 1000, 50)$ m shown in Figure 3.9 is the predefined path for the UAV to fly. While flying at height of 50 m the UAV will encounter different obstacles based up on the scenario under consideration. A cylindrical shaped obstacle with the center of its base at $(350, 300, 0)$ m, diameter of 100 m and height of 80 m remains stationary through out the analysis, and is assumed to be known prior to the start of the mission. A pop-up obstacle, as its name suggests, suddenly appears in front of the UAV with two different scenarios, one when the distance between the UAV and the obstacle is 50 m and second, 4 s prior to estimated collision with the obstacle (note that the pop-up obstacle appears sooner in the first case). This pop-up obstacle is not known prior to the start of the mission and only becomes known in real-time when it appears. The pop-up obstacle is taken to be a cuboid shaped obstacle with center of its base at $(600, 650, 0)$ m and dimension $100 \times 100 \times 60$ m^3. Finally, a spherical shaped structure with radius of 30 m and center at an altitude of 50 m is taken to be a moving obstacle. This obstacle has been given two different motion scenarios. First, the obstacle moves in a straight line which crosses the UAV's pre defined path at $(800, 800, 50)$ m. Second, the obstacle moves in a circular path crossing the UAV's predefined path at around $(836, 836, 50)$ m. In any case, it is assumed that the obstacle detection subsystem can only provide the obstacle's position at each sample time. As such, for the purpose of real-time trajectory generation, at any time instant, the optimal trajectory is generated as if the moving obstacle is stationary at its latest known location.

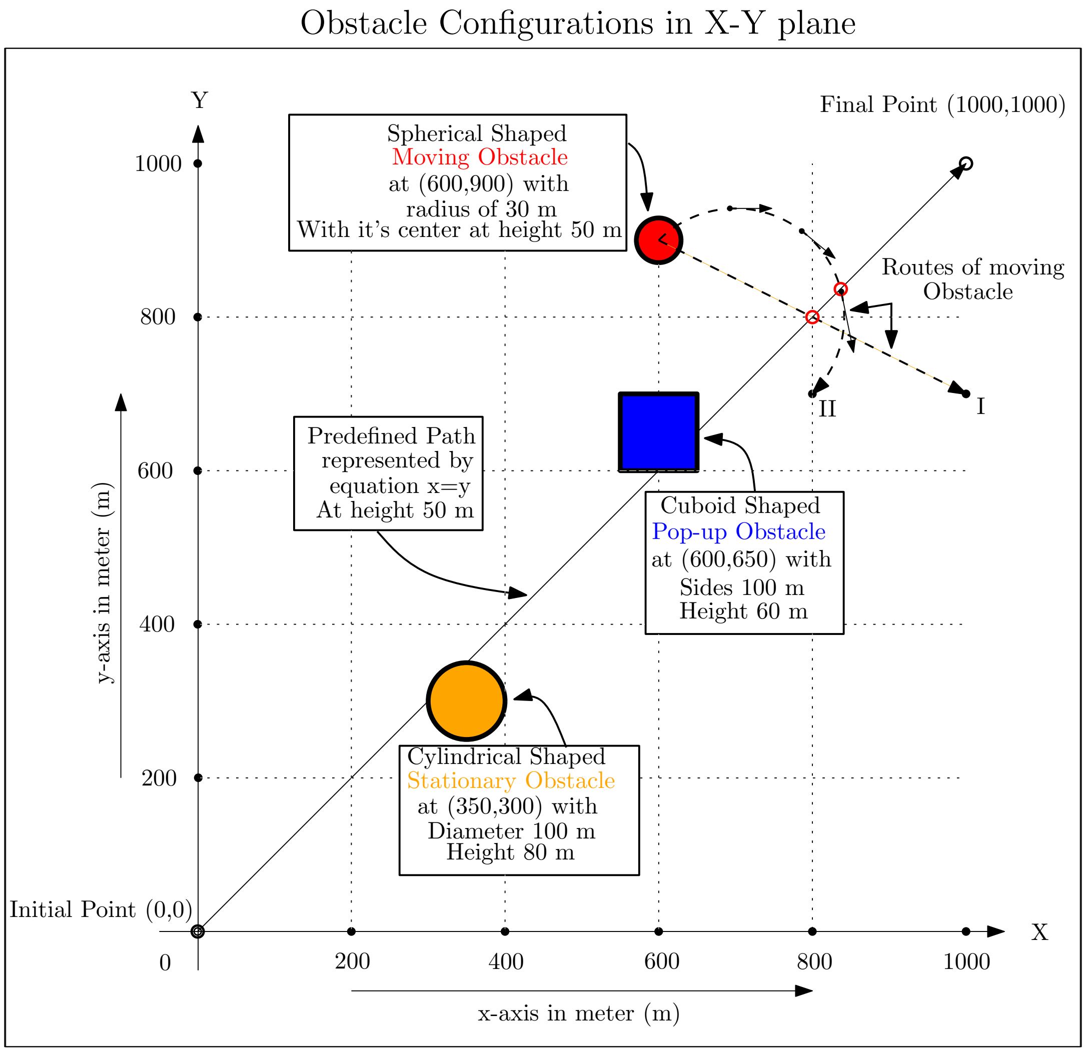


Figure 3.9: Obstacle Configurations

The UAV's task will be to fly from the initial point $(0, 0, 0)$ m to the final point $(1000, 1000, 0)$ m as close as possible to the predefined path at height of 50 m, avoiding obstacles such as stationary, pop-up and/or moving obstacles either in combination or separately based on the scenario under consideration. The UAV will start and end every autonomous mission facing towards x-axis with $\gamma = 0$, $\xi = 0$ and $V = 15$ m/s (minimum speed that UAV can fly without stalling). These numerical values are chosen to check the method's ability to incorporate constraints while generating trajectory. Based on Figure 3.9, five different obstacle scenarios are considered. They are:

1. **Without Obstacles**
2. **With Stationary Obstacles**
3. **Moving Obstacle**

4. **Pop-up Obstacle**
5. **Stationary, Pop-up and Moving Obstacles**

Chapter 4 contains detailed analyses of the five scenarios based on the optimal control problem formulation as shown in equation (3.27). Any changes to the optimal control formulation depending on the given scenario will be discussed where necessary.

Chapter 4

Results And Analysis

This Chapter presents the numerical results for the five different scenarios listed in Chapter 3. Moreover each scenario presented in the sections below has its own obstacle constraint that will be defined in that section. Equation (3.27) forms the basic optimal control problem formulation for all five scenarios. The path constraint for each obstacle defined by $h_i(x(t), y(t), z(t)) \geq 0$ and the cost function defined by $J(\underline{X}(\cdot), \underline{U}(\cdot), t_f)$, will be discussed if necessary.

Among the five scenarios, **Scenario I** and **II** are the only two cases that allow for a full open-loop solution, since full obstacle information is available beforehand (it is not updated in real time). Therefore, these two cases are solved in two ways. In the first, they are solved open-loop. In the second, they are solved in real-time. The resulting open-loop solutions are then verified and validated to ensure that they are in fact optimal. These optimal solutions are then used for comparison with the corresponding real-time solutions. **Scenarios III** to **V** represent more realistic scenarios where obstacle information is obtained and updated in real-time. Thus, for these cases, only real-time solutions are possible.

4.1 Scenario I. Without Obstacle

The obstacle free case represents an ideal scenario to see if the solution makes sense and if the method is capable of generating a trajectory to follow a nominal path, before considering the more complicated cases with obstacle present. In terms of trajectory generation for a UAV such as GeoSurv II, which is designed for a geophysical survey, flight duration is not of critical importance. The trajectory generated should be feasible and implementable in real time there by fulfilling the mission objectives. While the flight duration is not critical, it will be useful if it can be reduced, so as to potentially save fuel. To accomplish this, the final time can be added to the cost function in the trajectory optimization problem. To see if there is any benefit to doing this, **Scenario I** is further sub categorized into two cases namely, time optimal and non-time optimal. Furthermore, both sub-cases are tested in both open-loop and real-time (closed-loop) implementations. Figure 4.1 shows **Scenario I**.

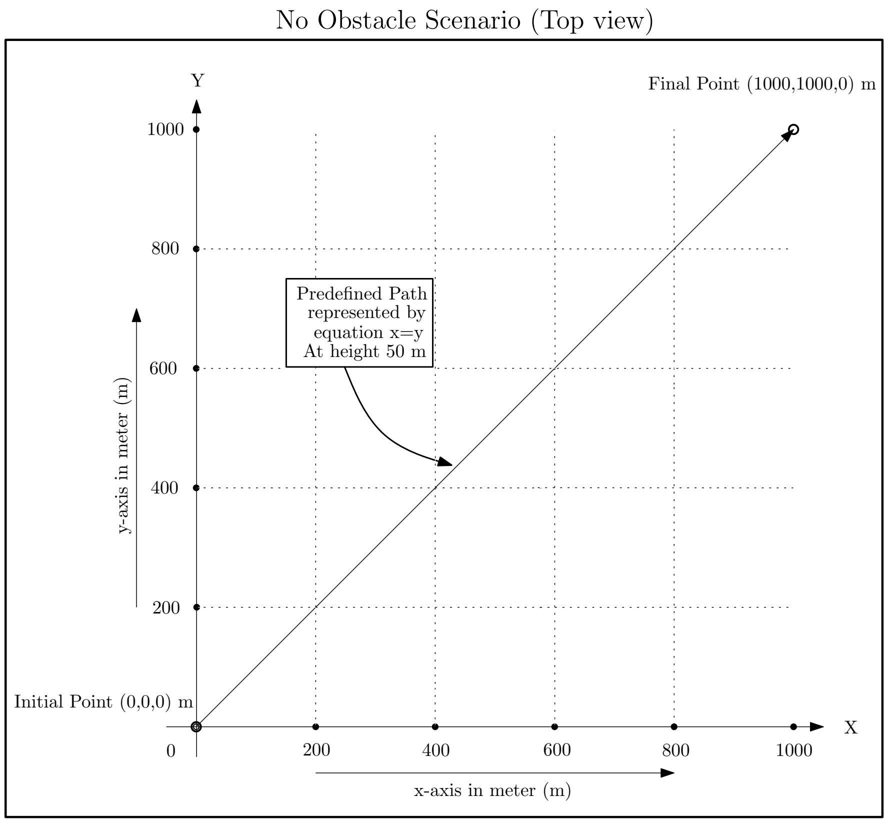

No Obstacle Scenario (Top view)
Y
Final Point (1000,1000,0) m
1000
800
600
400
200
0
Predefined Path represented by equation x=y At height 50 m
y-axis in meter (m)
Initial Point (0,0,0) m
200
400
600
800
1000
X
x-axis in meter (m)

Figure 4.1: Scenario I

IA. Open-loop, time optimal

The term time optimal in this thesis means that the final time is explicitly included in the cost function, as in the problem formulation in equation (3.27).

In **Scenario I**, there is no obstacle, and therefore the robustness factor, $r(x, y, z)$, is not included in the cost. Therefore, for **Scenario IA** the cost function in equation (3.27) reduces to

$$J(\underline{X}(\cdot), \underline{U}(\cdot), t_f) = t_f + A1 + A2 \tag{4.1}$$

The optimal control problem defined for **Scenario IA** is solved using DIDO [43] with 100 nodes. It gives $t_f = 50.3\ s$ as the total maneuver time at the total cost of $J = 51.1$. Figures

4.2-4.5 show the complete maneuver of the UAV in three-dimensional space. The states and controls shown in Figures 4.6 and 4.7 are within the given bounds as specified in equation (3.28) respectively.

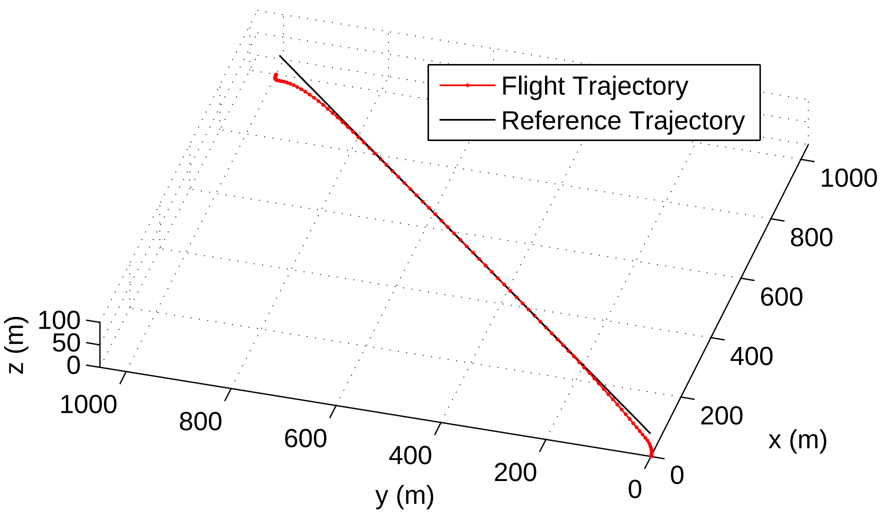


Figure 4.2: Isometric View of Total Maneuver

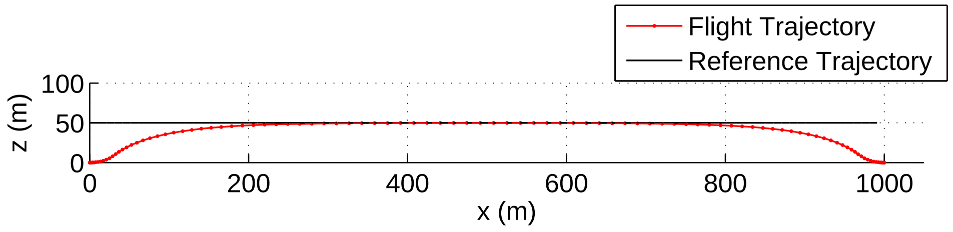


Figure 4.3: View in XZ Plane

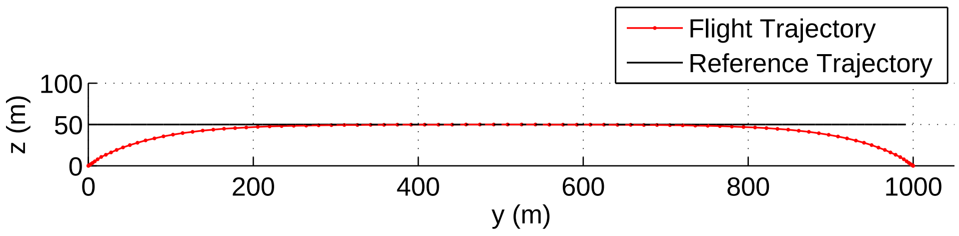


Figure 4.4: View in YZ Plane

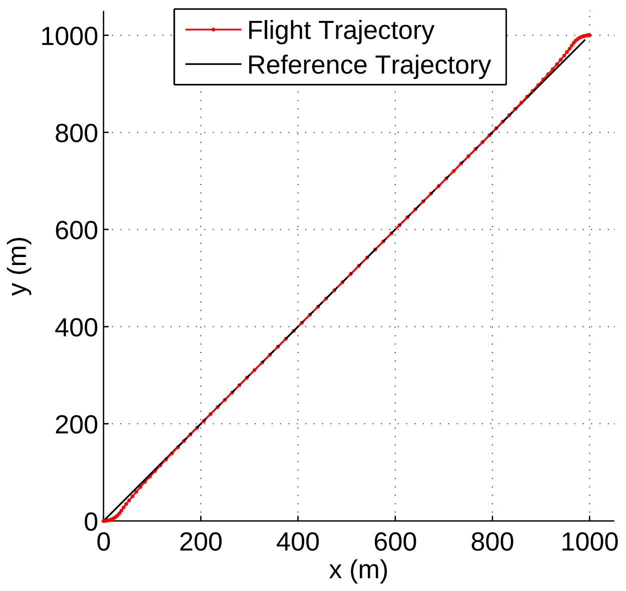


Figure 4.5: View in XY Plane

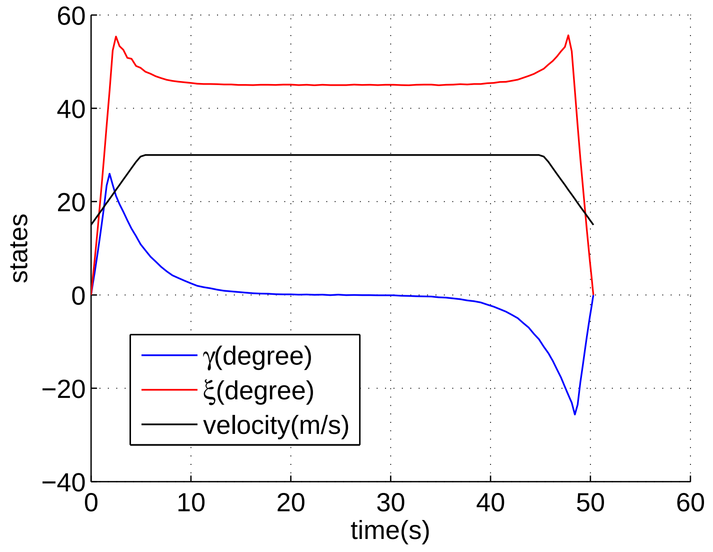


Figure 4.6: States [γ,ξ,V] vs time (t)

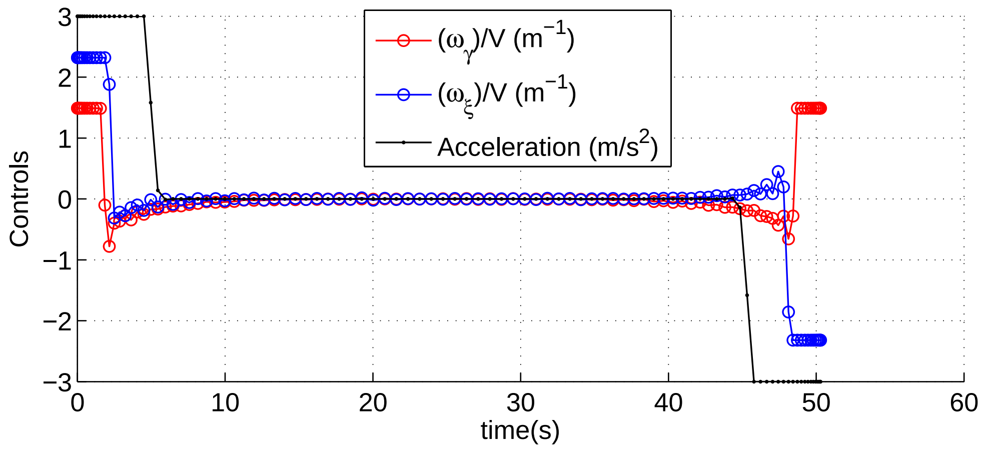


Figure 4.7: Control Trajectory

Feasibility Check:

The first step in verification is to show the feasibility of the generated solution. This can be done by control trajectory interpolation and state propagation using a Runge-Kutta algorithm as explained in subsection 2.1.2. The initial conditions and system dynamics of the UAV state variables were propagated using the control solution shown in the Figure 4.7**.** Then, the propagated state variables were superimposed on the DIDO generated solution, which demonstrates the feasibility of the solution if they are close. Figures 4.8-4.14 show the propagated states versus the DIDO generated states for the feasibility check.

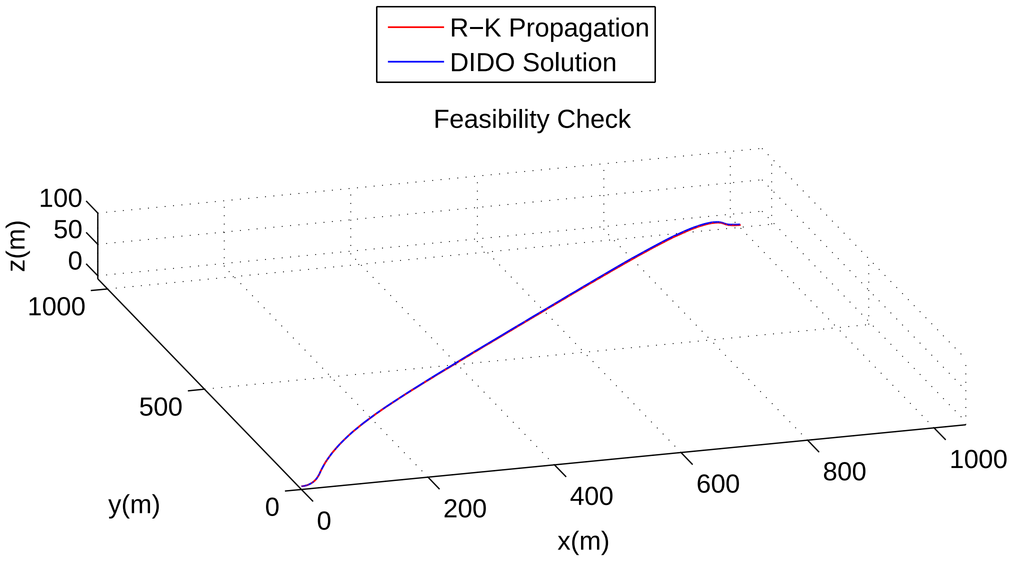


Figure 4.8: Isometric View of Complete Maneuver

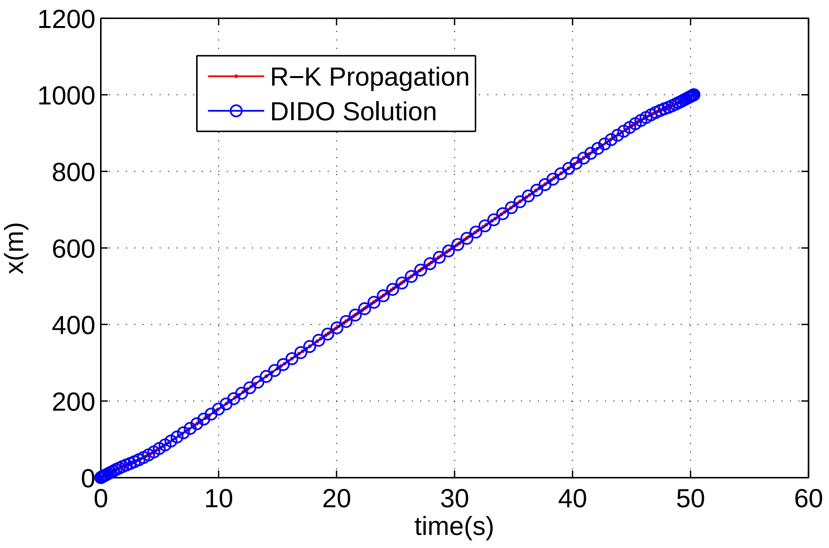


Figure 4.9: Feasibility Check (x vs t)

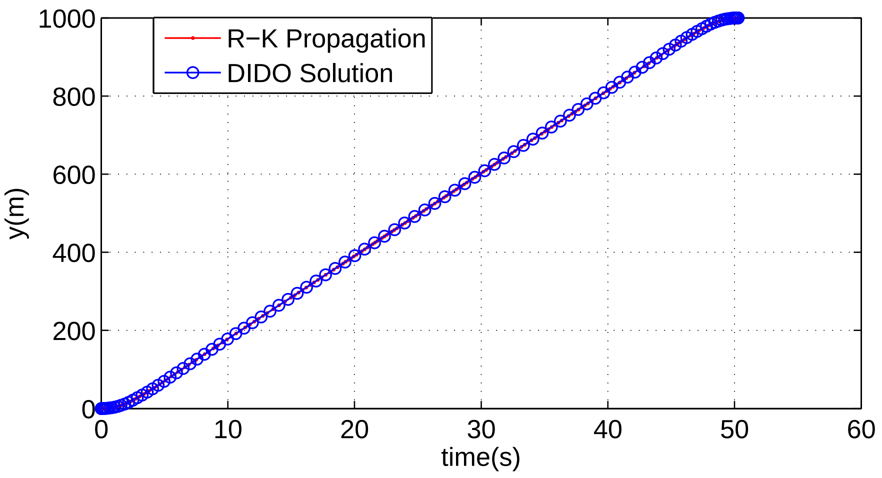
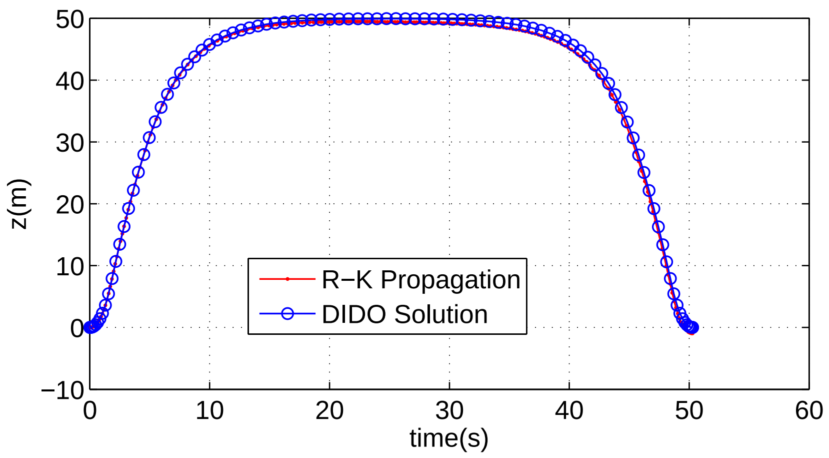


Figure 4.10: Feasibility Check (y vs t)



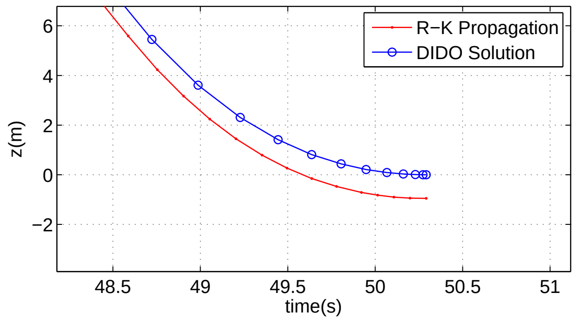


Figure 4.11: Feasibility Check (z vs t)

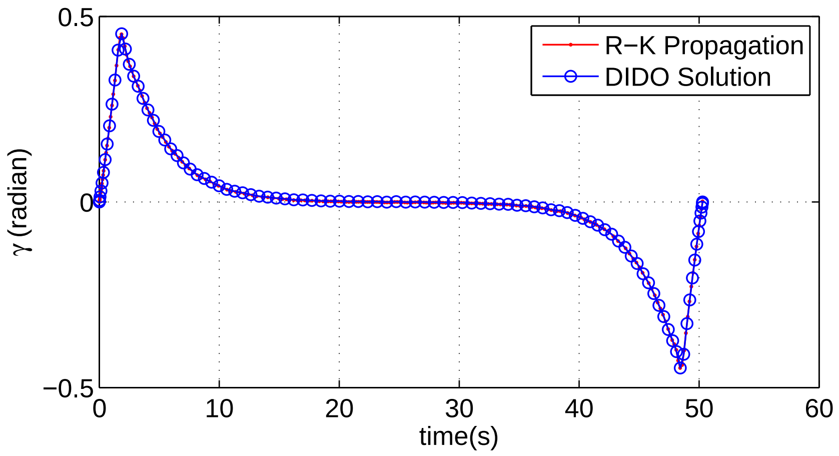


Figure 4.12: Feasibility Check (γ vs t)

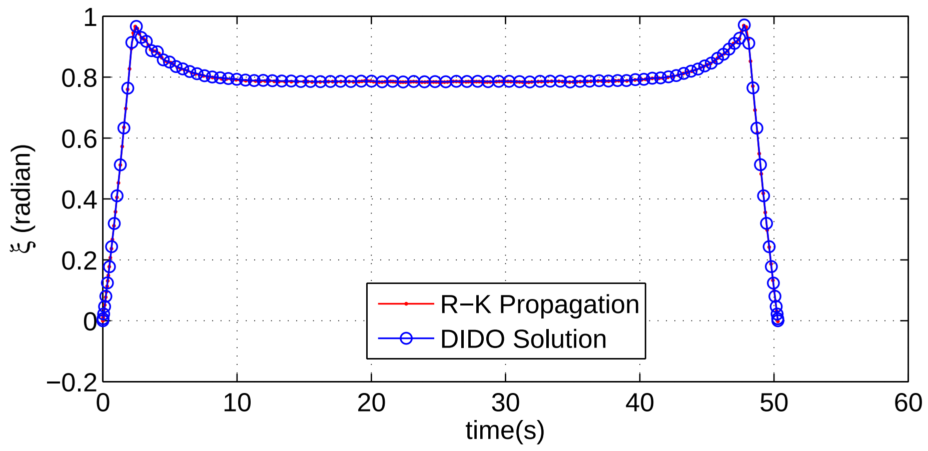


Figure 4.13: Feasibility Check (ξ vs t)

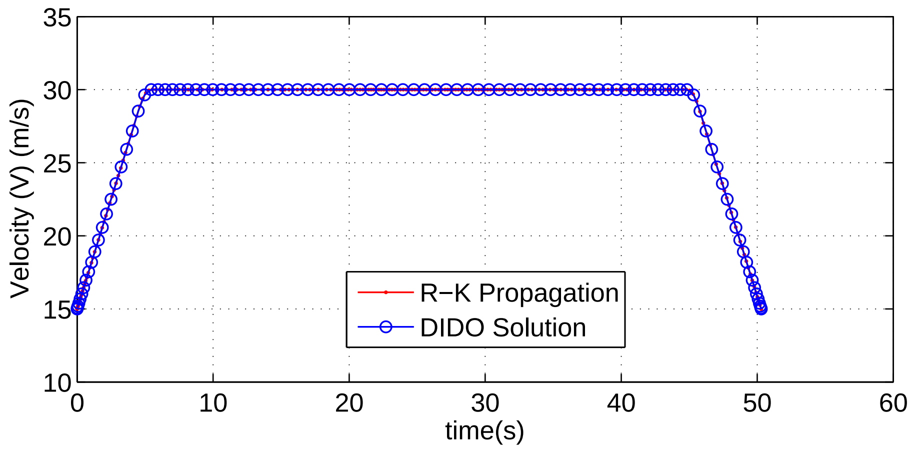


Figure 4.14: Feasibility Check (V vs t)

The Runge-Kutta propagated states deviate slightly from the DIDO generated states. Figure 4.11 shows increase in error as it propagates to the end. This is because the PS (DIDO) generated states and controls only enforce a discretization of the dynamic equation at the node points. However, the deviation of the Runge-Kutta propagated solution from the DIDO solution is small, giving confidence in the feasibility of the DIDO generated solution.

Hamiltonian Value Condition:

Pontryagin's principle does not give sufficient conditions for optimality, just necessary ones. That is, just because Pontryagin's principle is satisfied, it does not mean the solution is optimal. However, if Pontryagin's principle is not satisfied, then we know that the solution isn't optimal. We do not check all conditions in Pontryagin's principle, instead, we only check the Hamiltonian condition, since this is readily done, and it gives a quick indication of whether or not the solution can be trusted. From equation (2.17), the optimal control Hamiltonian value for problem **Scenario IA** is $H^*(\tau) = -1$. As shown in Figure 4.15, Pontryagin's Hamiltonian value is nearly constant with value $H^*(\tau) = -1$, as required.

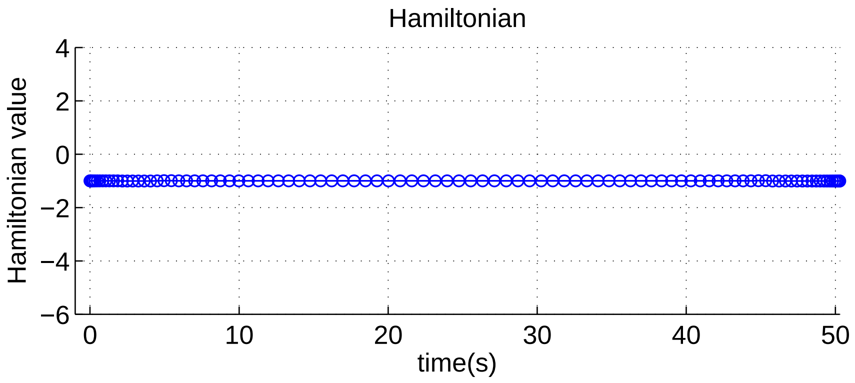


Figure 4.15: Hamiltonian Value

Bellman Principle Test:

From the Bellman's test of optimality described in subsection 2.1.2, the problem is broken down into two halves from the 50th node of original problem. The first half from $(0, 0, 0)$ m to $(508.43, 508.39, 49.93)$ m and the second from $(508.43, 508.39, 49.93)$ m to $(1000, 1000, 0)$ m. The problem is solved by taking the final condition of the first half as the initial condition of the second half. The cost and time taken for these two halves were noted.

These two halves were then combined to show an overall trajectory. This combined trajectory is compared with the original trajectory. Figures 4.16-4.19 show the state and control trajectories of the original problem with the 1^{st}half and 2^{nd} half problem superimposed on the same graph. The cost of the1^{st} half problem was $J_1 = 25.9719$ with 50 nodes and the cost of the 2^{nd} half problem was $J_2 = 25.1770$ also with 50 nodes. And on adding of 1^{st} and 2^{nd} halves gives $J_1 + J_2 = 51.1489$ which almost matches the original problem cost of $J = 51.1442$. The slight difference in cost is due to the difference in node distributions between the original and split problems.

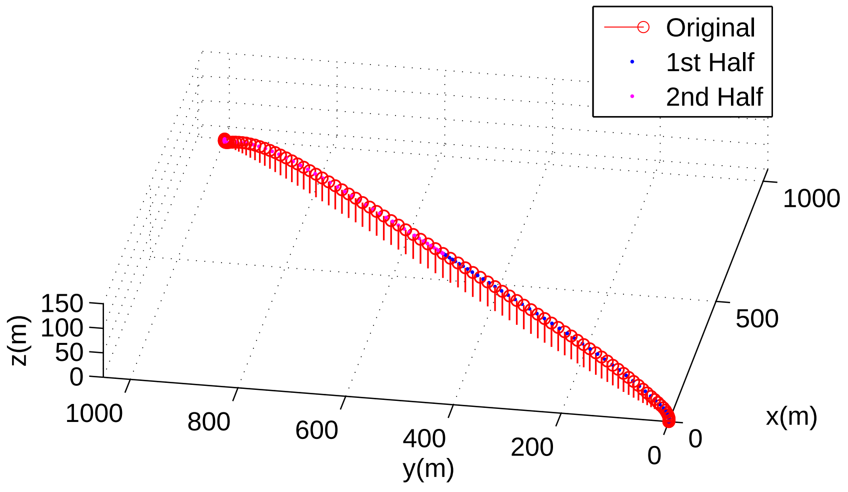


Figure 4.16: Bellman's Test in 3D Maneuver

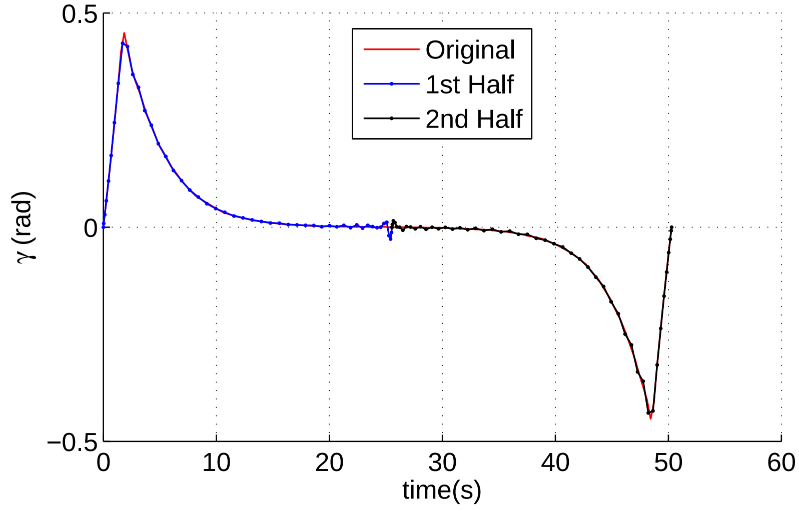


Figure 4.17: Bellman's Principle Test (γ vs t)

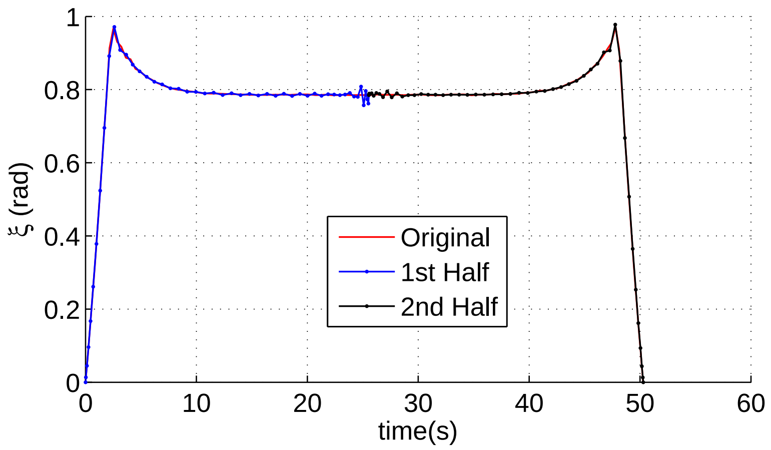


Figure 4.18: Bellman's Principle Test (ξ vs t)

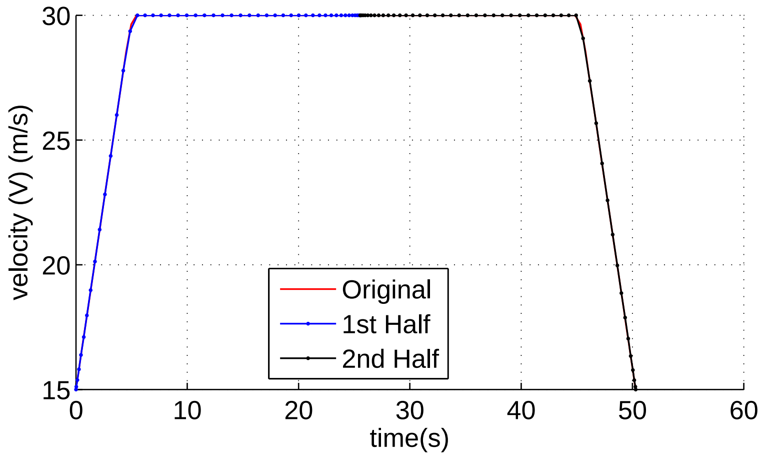


Figure 4.19: Bellman's Principle Test (V vs t)

This sub-section has presented the results of a time optimal open-loop UAV trajectory planning problem. Next, the non-time optimal case is examined, where the trajectory is optimized to stay as close as possible to the desired trajectory while satisfying the UAV and obstacle constraints, without any consideration for the total maneuver time.

IB. Open-loop, non-time optimal

As stated in subsection 4.1, non-time optimal means that the final time is excluded from the cost to be minimized. As for Scenario IA, there are no obstacles, and therefore no robustness factor. Consequently, the cost function in equation (3.27) becomes

$$J(\underline{X}(\cdot), \underline{U}(\cdot)) = A1 + A2 \tag{4.2}$$

The problem solved with 100 nodes gives $t_f = 52.9\ s$ as the total maneuver time with a the total cost of $J = 0.65$.

Removing the contribution of final time from the cost in **Scenario IA** and comparing it to the cost in the **Scenario IB** shows that removing the final time from the cost in the trajectory optimization, results in a 23% decrease in the cost, while only a 5.2% increase in the final time.

Figures 4.20-4.23 show the complete maneuver of the UAV in three-dimensional space. The states and controls shown in Figures 4.24 and 4.25 are within the given bounds as specified in (3.28) respectively.

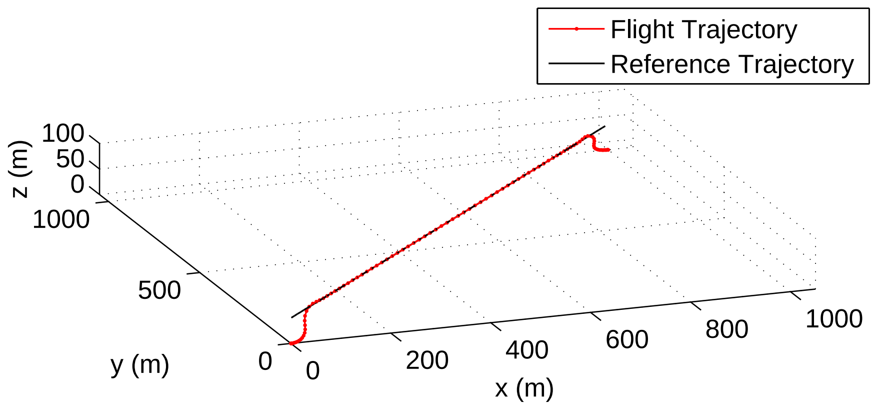


Figure 4.20: Isometric View of Total Maneuver

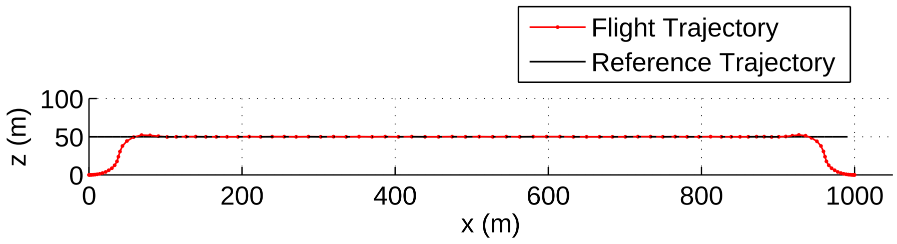


Figure 4.21: View in XZ Plane

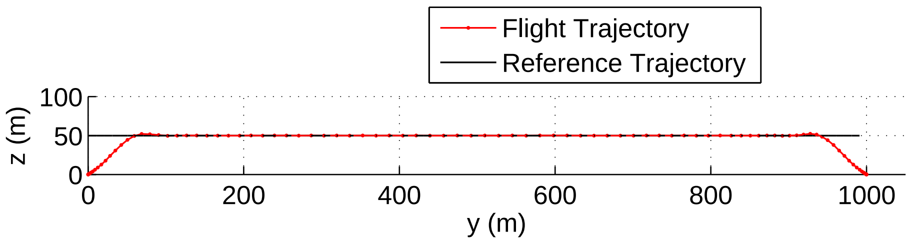


Figure 4.22: View in YZ Plane

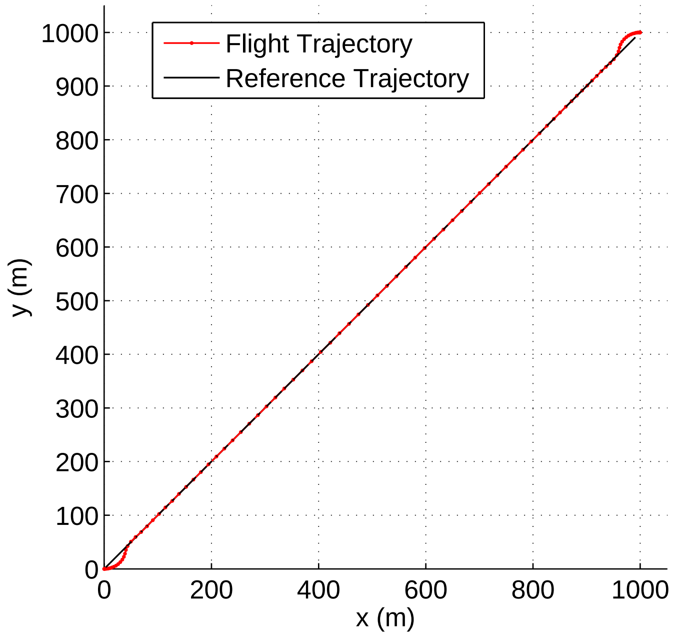


Figure 4.23: View in XY Plane

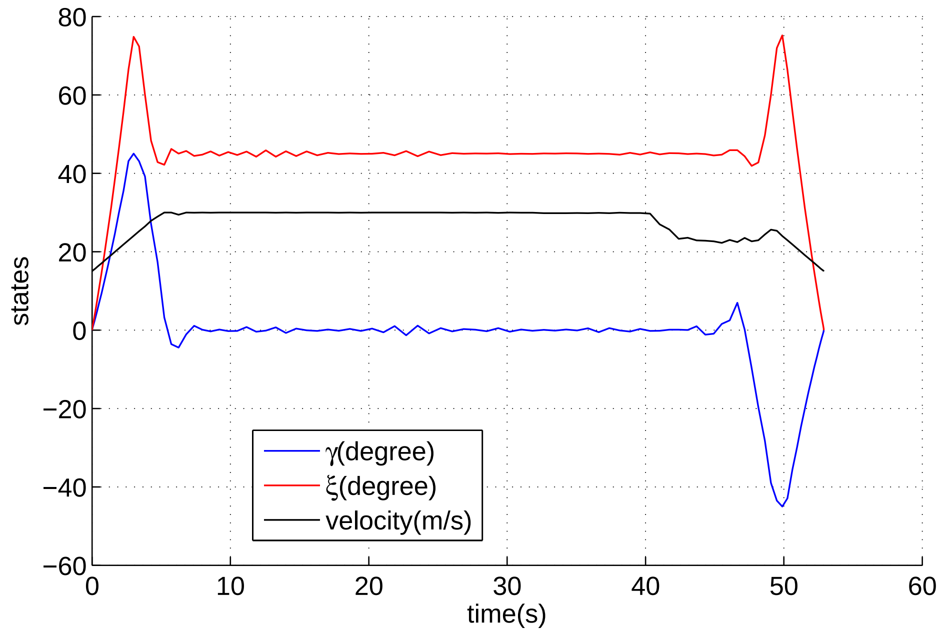


Figure 4.24: States [γ,ξ,V] vs time (t)

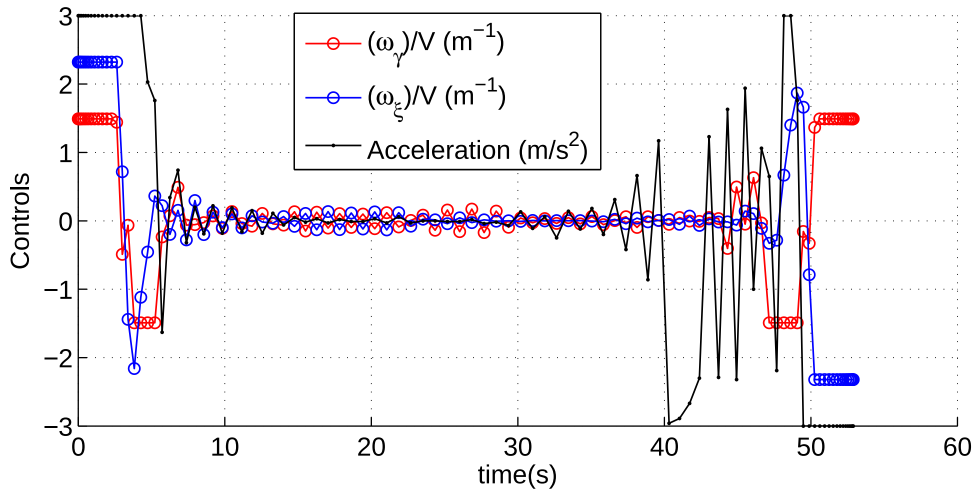


Figure 4.25: Control Trajectory

Feasibility Check:

Analogous to **Scenario IA**, Figures 4.26 to 4.32 show the feasibility check for **Scenario IB**. These figures show that the DIDO generated solution is clearly feasible with minor error in Figure 4.29, with similar reason to **IA**.

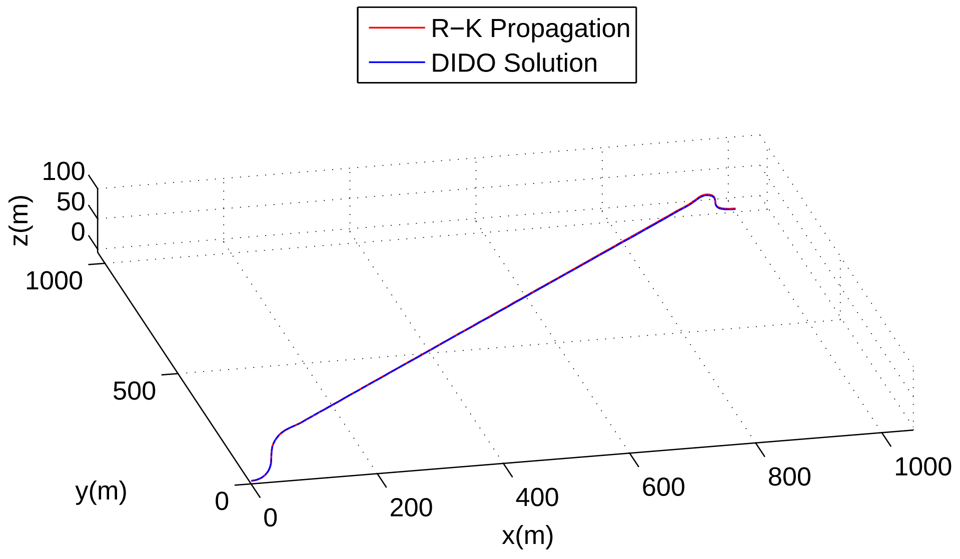


Figure 4.26: Isometric View of Complete Maneuver

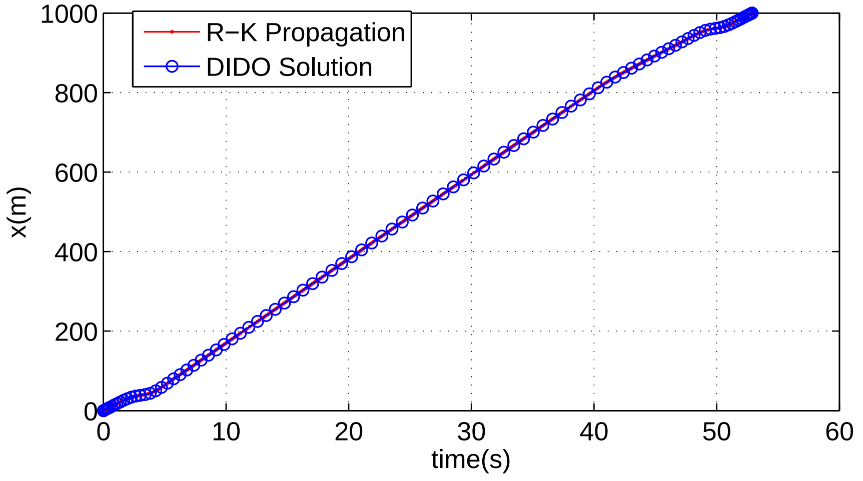


Figure 4.27: Feasibility Check (x vs t)

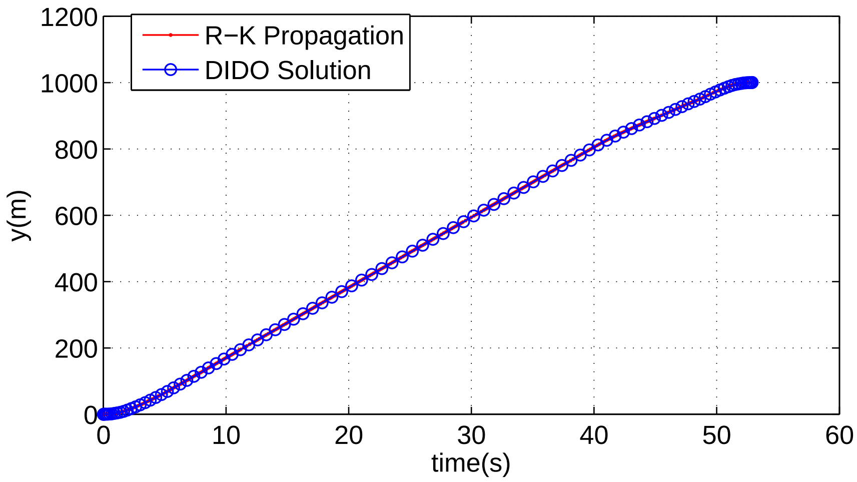


Figure 4.28: Feasibility Check (y vs t)

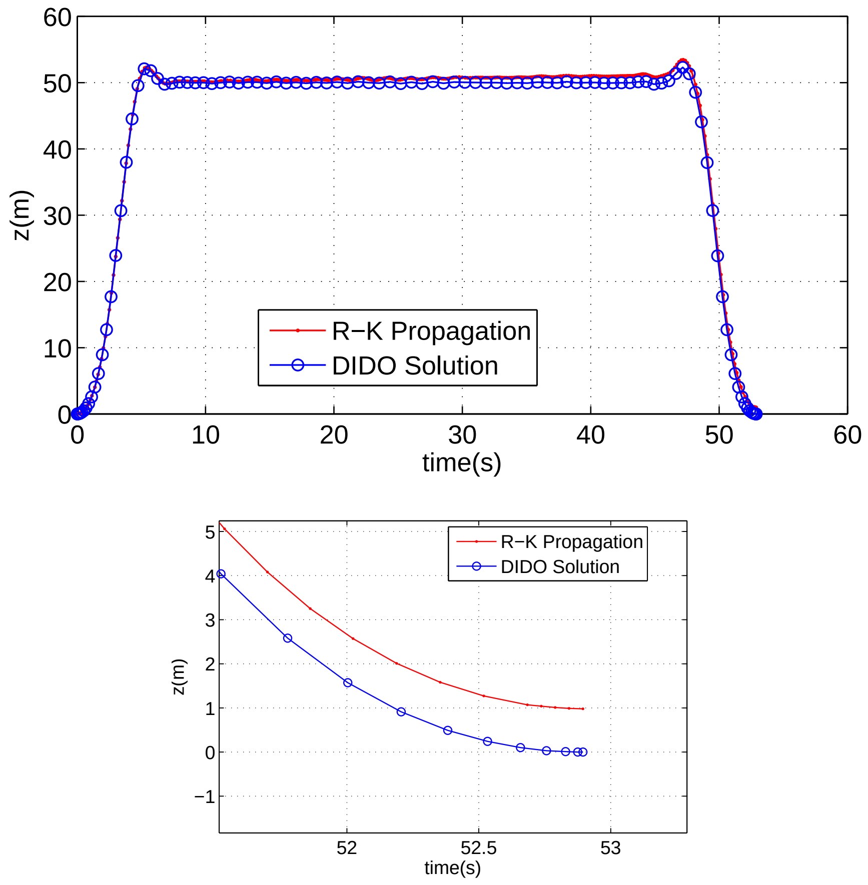


Figure 4.29: Feasibility Check (z vs t)

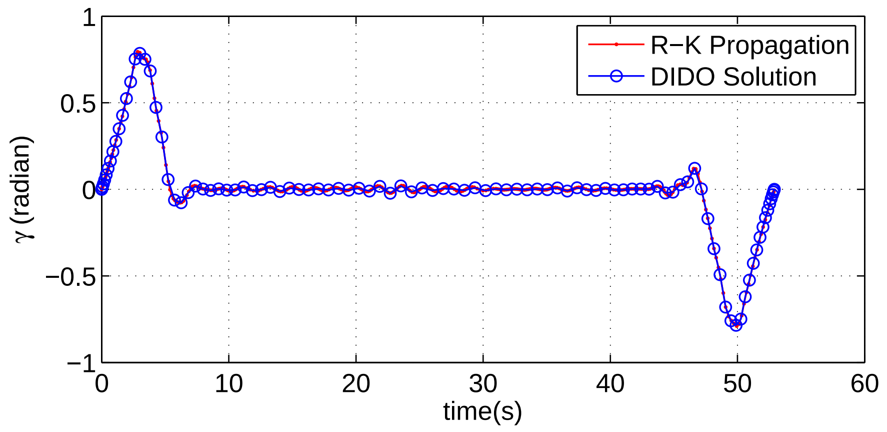


Figure 4.30: Feasibility Check (γ vs t)

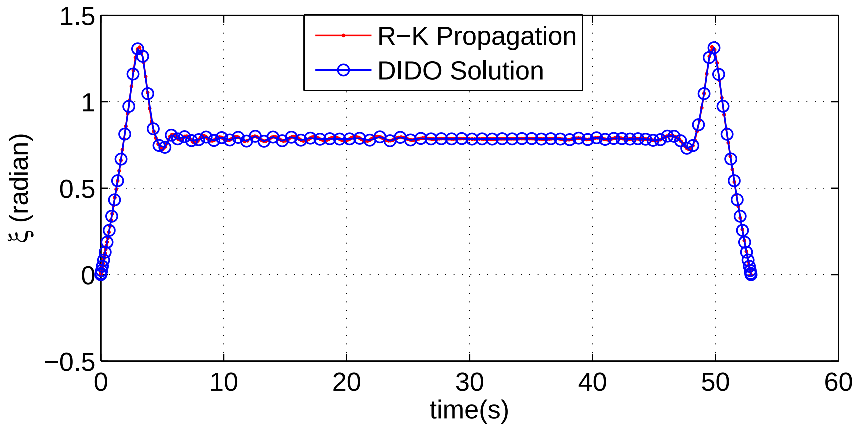


Figure 4.31: Feasibility Check (ξ vs t)

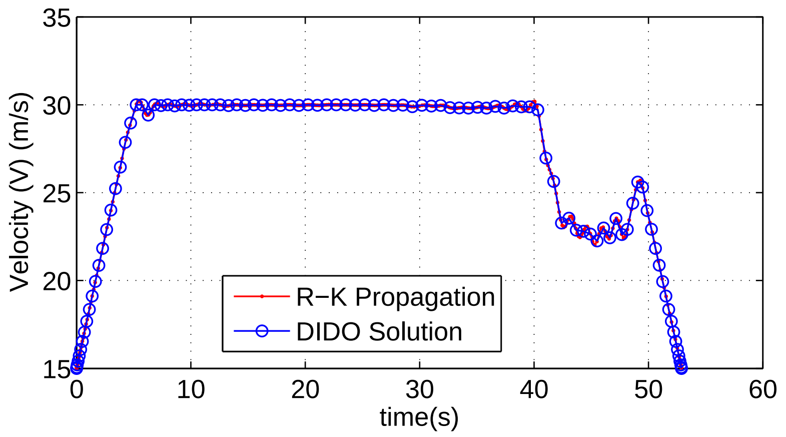


Figure 4.32: Feasibility Check (V vs t)

Hamiltonian Value Condition:

From equation (2.17), the optimal control Hamiltonian value for problem **Scenario IB** is $H^*(\tau) = 0$. As shown in Figure 4.33, Pontryagin's Hamiltonian value is nearly constant with value $H^*(\tau) = 0$, as required.

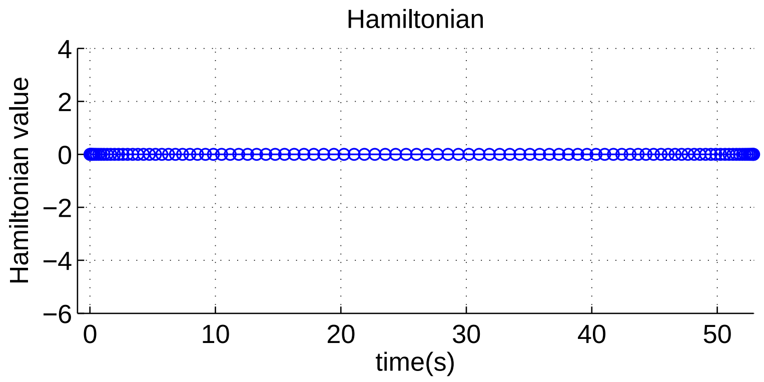


Figure 4.33: Hamiltonian Value

Bellman Principle Test:

Just as in **Scenario IA**, the problem is broken down into two halves. The first half from $(0, 0, 0)$ m to $(527.38, 527.37, 49.85)$ m and the second from $(527.38, 527.37, 49.85)$ m to $(1000, 1000, 0)$ m. The cost and time taken for these two halves were noted.

Figures 4.34-4.37 show the state trajectories of the original problem with the 1^{st} half and the 2^{nd} half problems superimposed on the same graph. The cost for the 1^{st} half problem was $J_1 = 0.3272$ with 50 nodes; the cost for the 2^{nd} half problem was $J_2 = 0.3279$ also with 50 nodes; and addition of the 1^{st} and 2^{nd} halves gives $J_1 + J_2 = 0.6551$ which almost matches the original problem cost of $J = 0.6558$. Figures 4.34, 4.35 and 4.36 show good agreement in the UAV trajectory as well as angles ξ and γ. However, figure 4.37 shows significant differences in UAV speed. This can be understood by the fact in the non-time optimal case the cost is unaffected by the speed of the UAV once it is flying along the reference trajectory, since the integrands of $A1$ and $A2$ are zero during this portion. Hence, there are multiple solutions for the UAV speed, yielding the same cost.

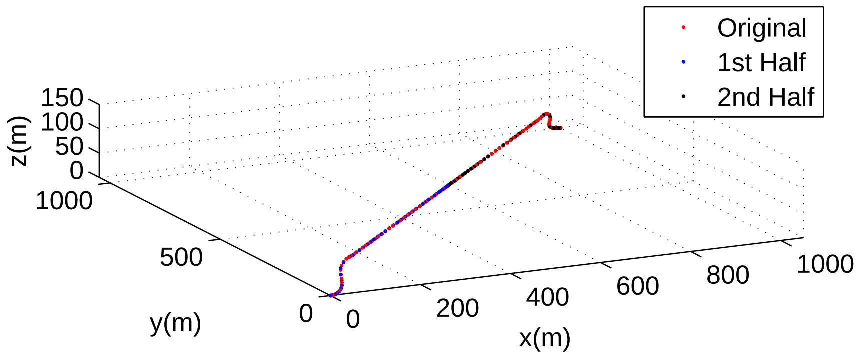


Figure 4.34: Bellman's Test in 3D Maneuver

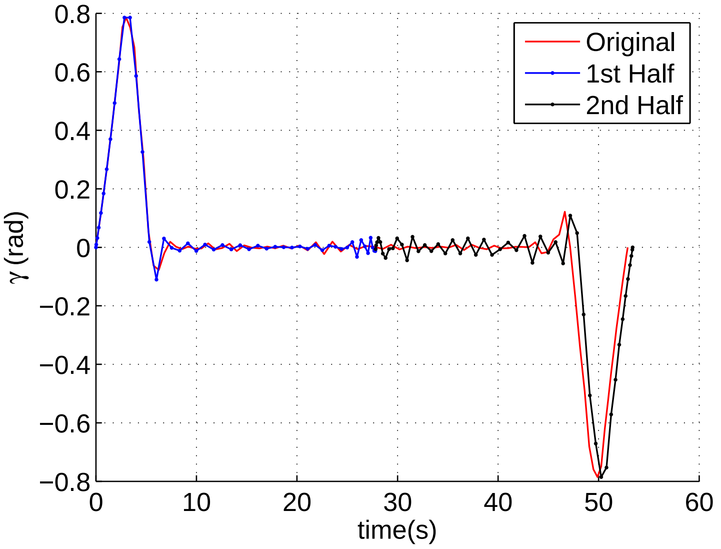


Figure 4.35: Bellman's Principal Test (γ vs t)

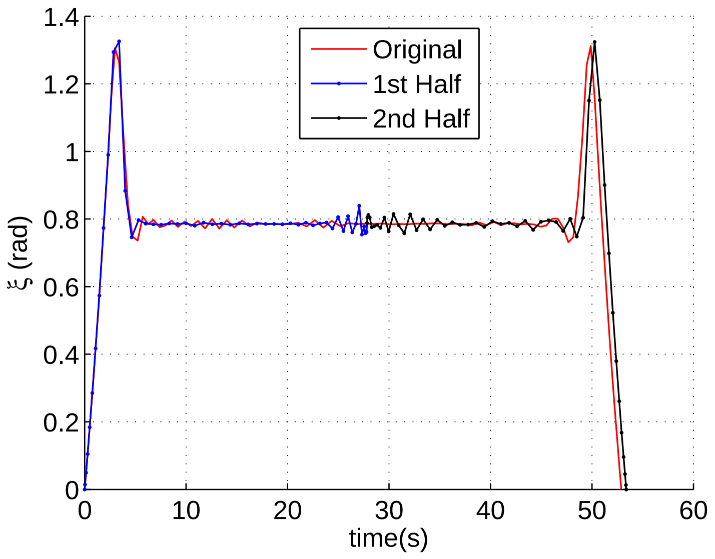


Figure 4.36: Bellman's Principle Test (ξ vs t)

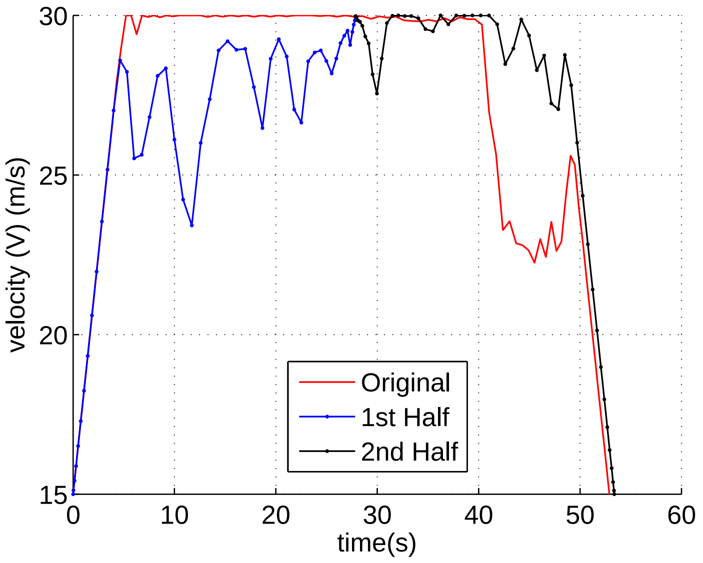


Figure 4.37: Bellman's Principle Test (V vs t)

This sub-section has presented the results of a non time optimal open-loop UAV trajectory planning problem. In the next section, we further check the feasibility of this method in realtime path planning.

IC. Realtime, time optimal

Realtime path planning uses the de-aliasing principle as explained in subsection 2.3. Before presenting the real-time results for **Scenario IC**, an appropriate sample rate and number of nodes for each solution must be selected. Because of the high convergence rate of PS methods, we simply use low node solutions for online calculation, which further lowers the computation time [49]. Figure 4.38 shows the optimal open-loop trajectories corresponding to **Scenario IB** using 15, 30, 50 and 100 nodes with $4s$, $11s$, $28s$, $388s$ computation time respectively. For real-time implementation, a lower computation time allows one to increase the sampling frequency. In general, according to the de-aliasing principle, a lower sample time leads to less propagation error and higher accuracy. To support this statement Figure 4.29, is considered, which shows that the error increases with increasing propagation time.

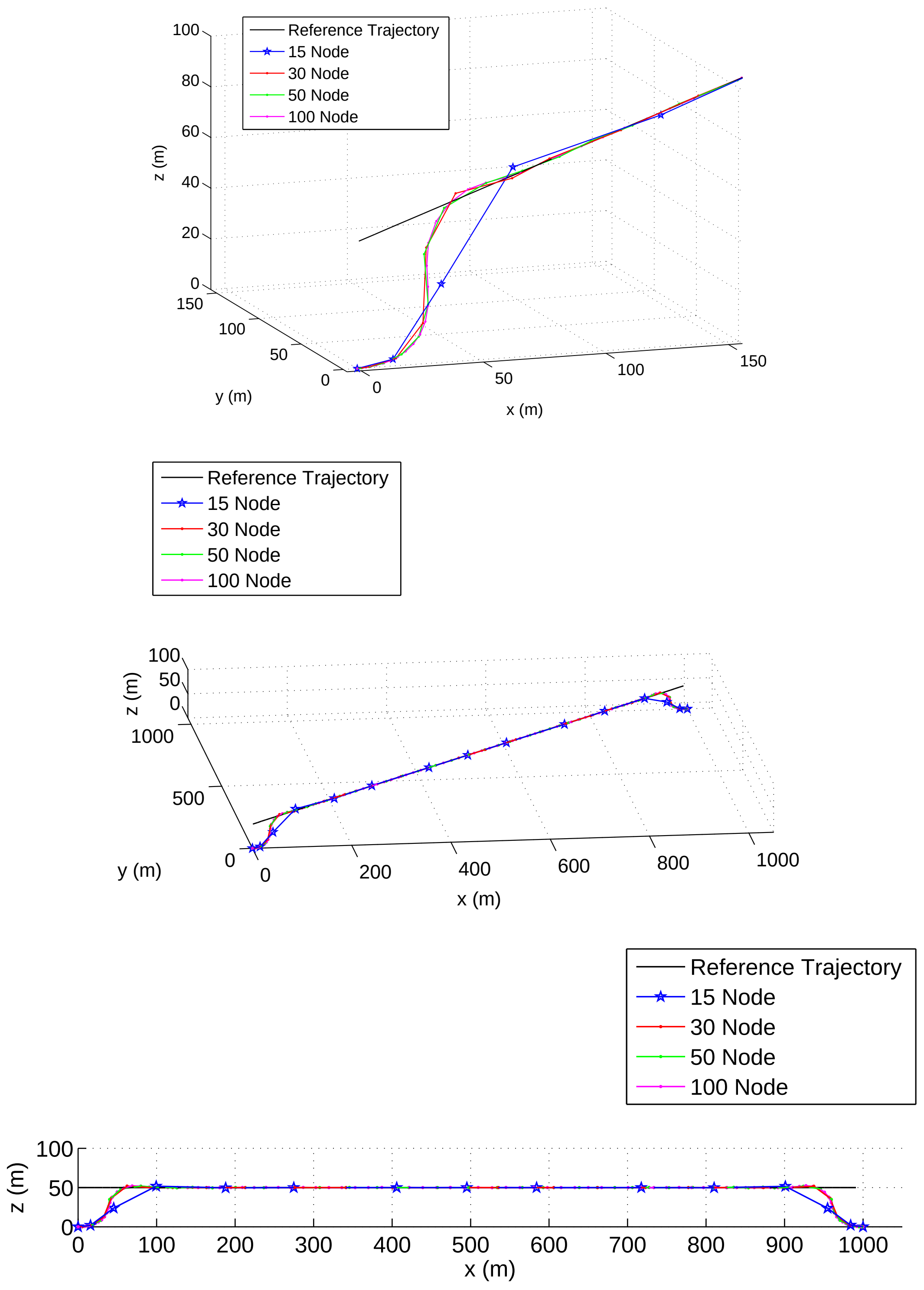


Figure 4.38: Trajectory comparision with different nodes.

Regarding the computation time, reference [49] states that, excluding the Windows platform and MATLAB overheads, the computation time can be decreased by a factor of 100, with optimized actual code. For implementation on a real UAV, the program would be customized excluding unnecessary platform and overheads. Hence, with this in mind, a sampling frequency of 5 Hz is assumed, that is, we assume the computer on board with UAV can compute each open-loop solution in 0.2 seconds. This is conservatively based on a 30 nodes solution, which as mentioned before was computed in 11 s. Figure 4.39 shows the state trajectory from 0.2 to 0.4 seconds for the 30 nodes and 100 nodes solutions.

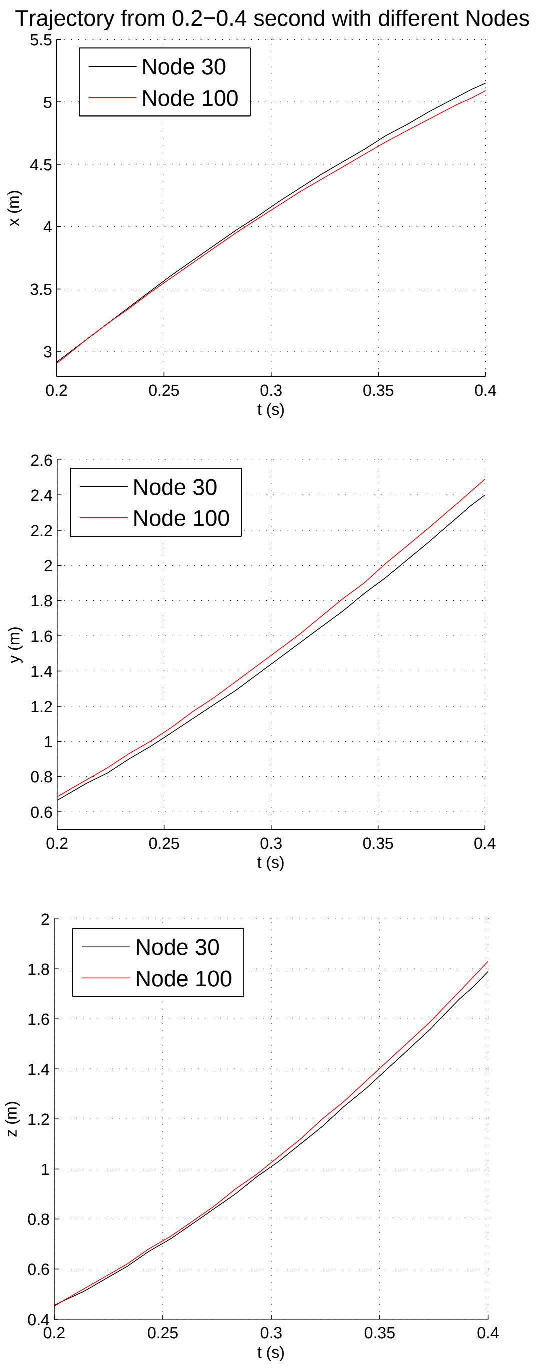


Figure 4.39: Trajectory Propagation using controls from 0.2-0.4 second

Scenario IC uses the same problem formulation as **Scenario IA**, with the difference being that the solution is generated in real-time with a 5 Hz sampling frequency and 30 nodes used for the solution at each sample instant. The simulation results are shown in Figures 4.40 to 4.46. The R-K propagated, open-loop time optimal solution generated with 100 nodes from **Scenario IA** is shown for comparison. The average time for computing each open-loop is

4.1026 *s*. A Computer with Intel CPU @ 2.9 GHz and 4 GB RAM was used to obtain all the results in this thesis.

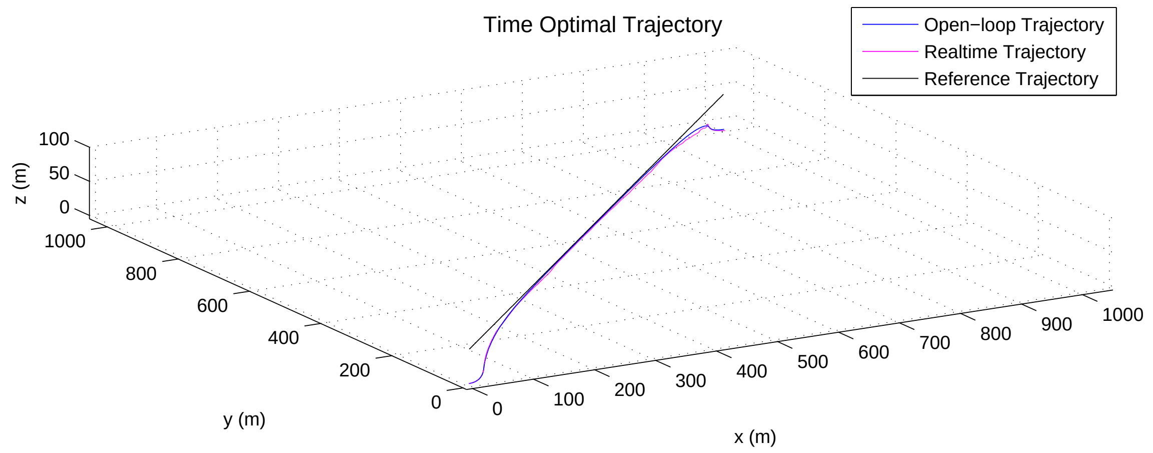


Figure 4.40: Realtime vs Open-loop Maneuver in Isometric View

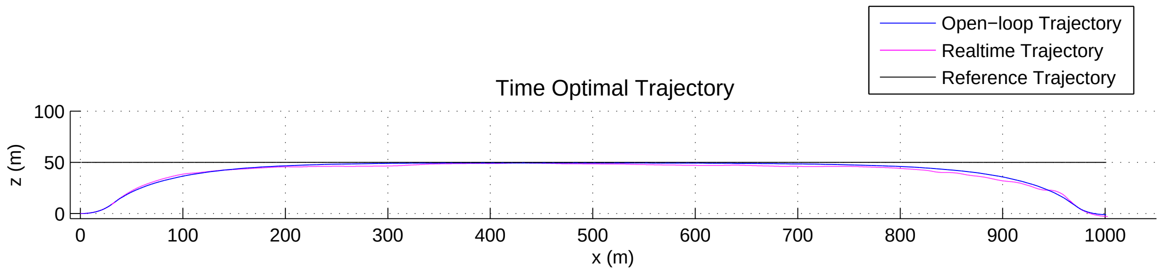


Figure 4.41: Realtime vs Open-loop Maneuver in XZ Plane

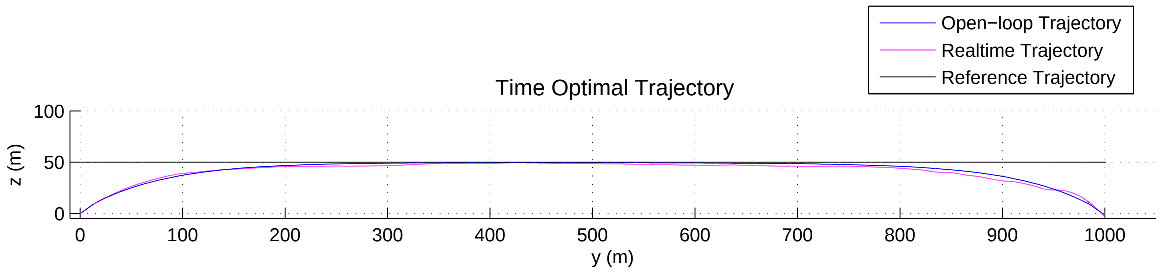


Figure 4.42: Realtime vs Open-loop Maneuver in YZ Plane

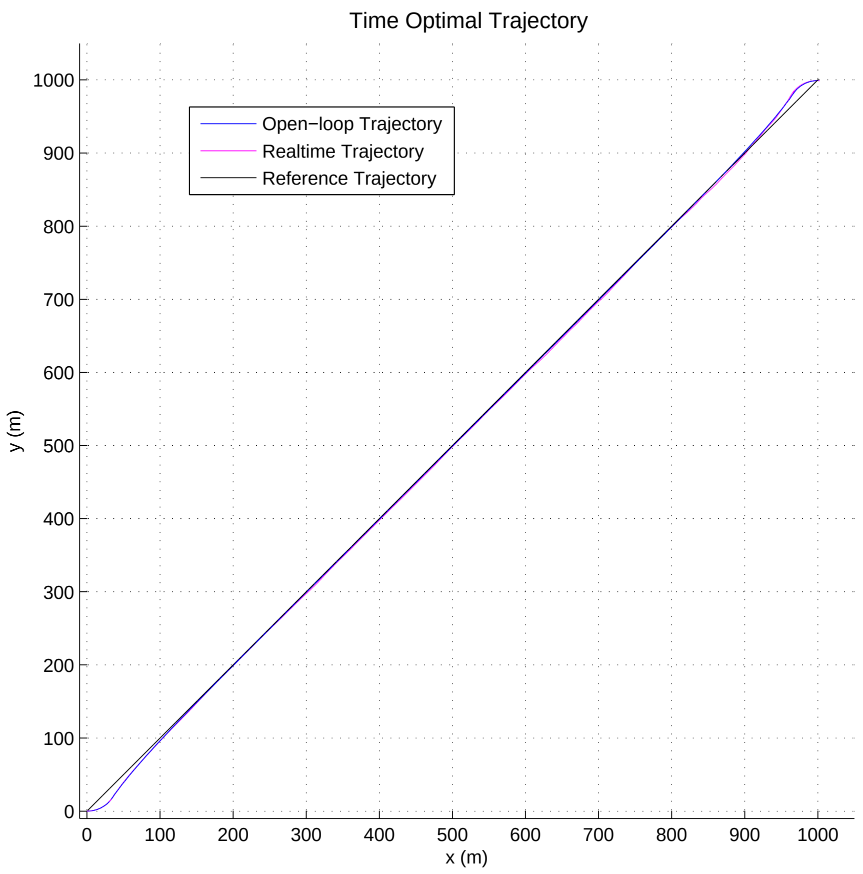


Figure 4.43: Realtime vs Open-loop Maneuver in XY Plane

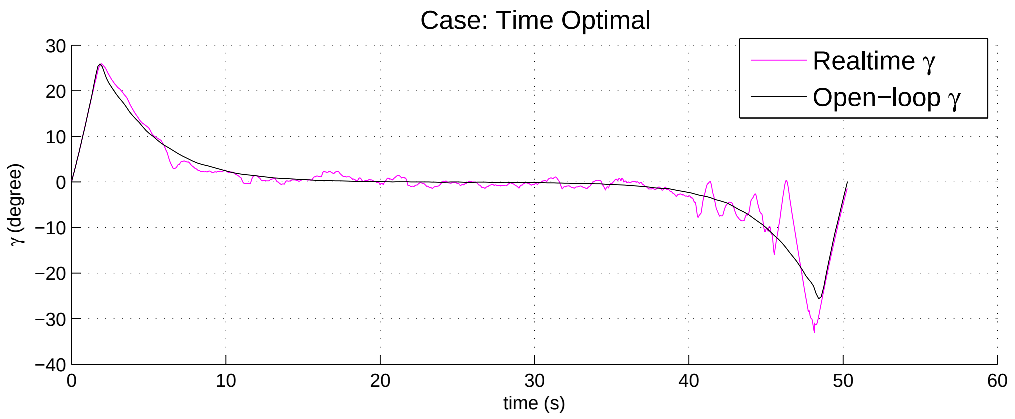


Figure 4.44: Realtime vs Open-loop γ angle

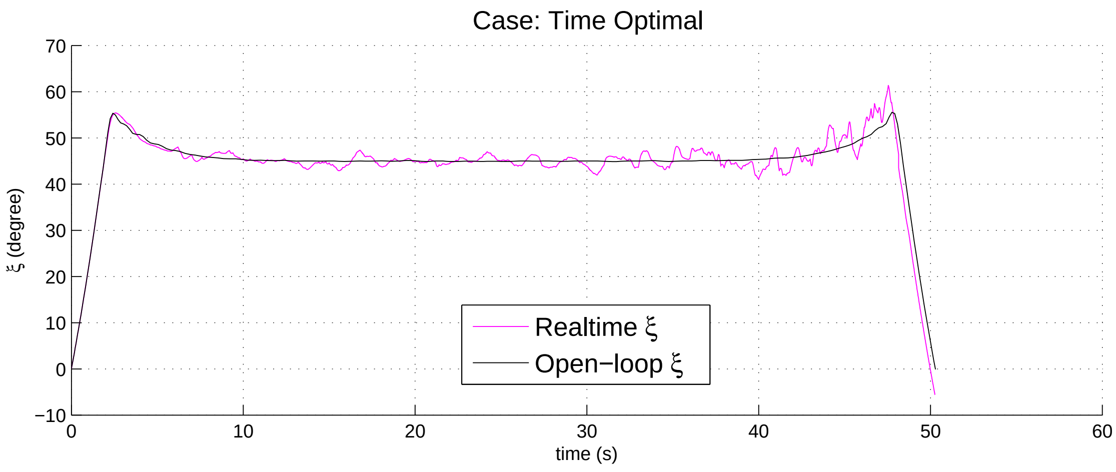


Figure 4.45: Realtime vs Open-loop ξ angle

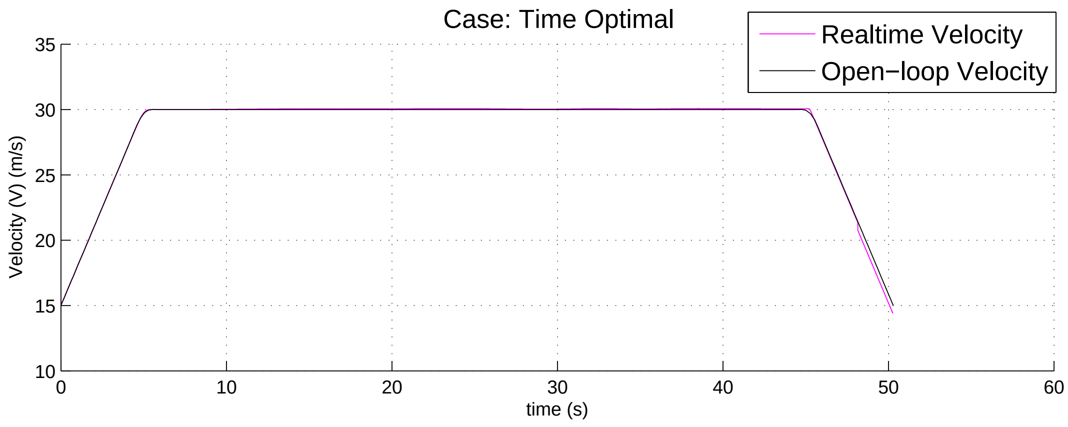


Figure 4.46: Realtime vs Open-loop (V) Velocity

It can be seen from the results that the real-time generated solution with 30 nodes matches the open-loop 100 node solution very well, justifying its use for real-time trajectory generation. During realtime simulation if the realtime updates are allowed to happen until the goal point, the final update may become infeasible, since it could extend beyond the required final position. To prevent the in-feasibility the realtime updates are stopped 30m ahead of the goal point and allowed to propagate with the last open-loop controls of realtime simulation. Because of this reason the real-time updated states have propagation errors at the end which are visible in Figures 4.46, 4.111, 4.112, 4.118 and 4.119.

ID. Realtime, non-time optimal

Scenario ID uses the same problem formulation as **Scenario IB**, with the difference being that the solution is generated in real-time with a 5 Hz sampling frequency and 30 nodes used for the solution at each sample instant. The simulation results are shown in Figures 4.47 to 4.53. The open-loop optimal solution generated with 100 nodes from **Scenario IB** is shown for comparison. The average time for computing each open-loop problem is 9.8170 s.

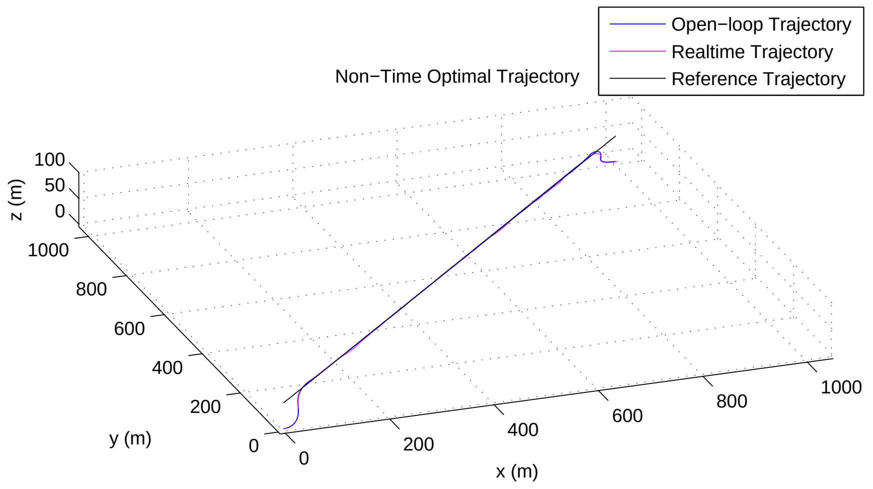


Figure 4.47: Realtime vs Open-loop Maneuver in Isometric View

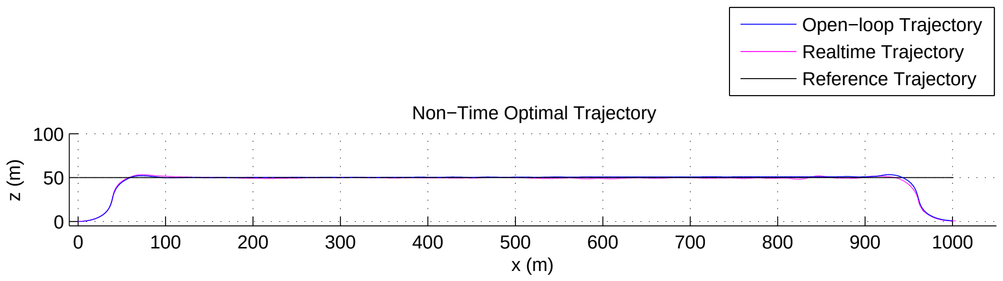


Figure 4.48: Realtime vs Open-loop Maneuver in XZ Plane

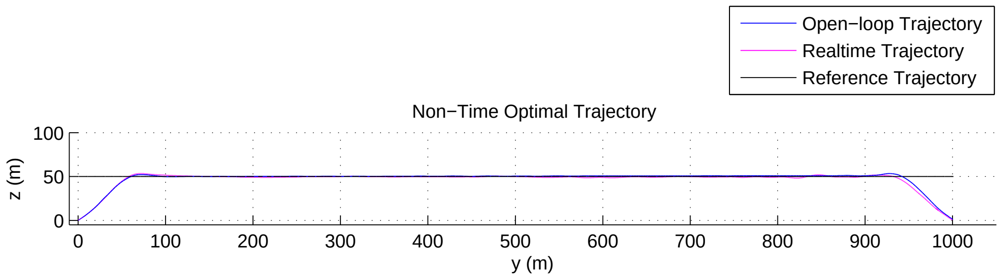


Figure 4.49: Realtime vs Open-loop Maneuver in YZ Plane

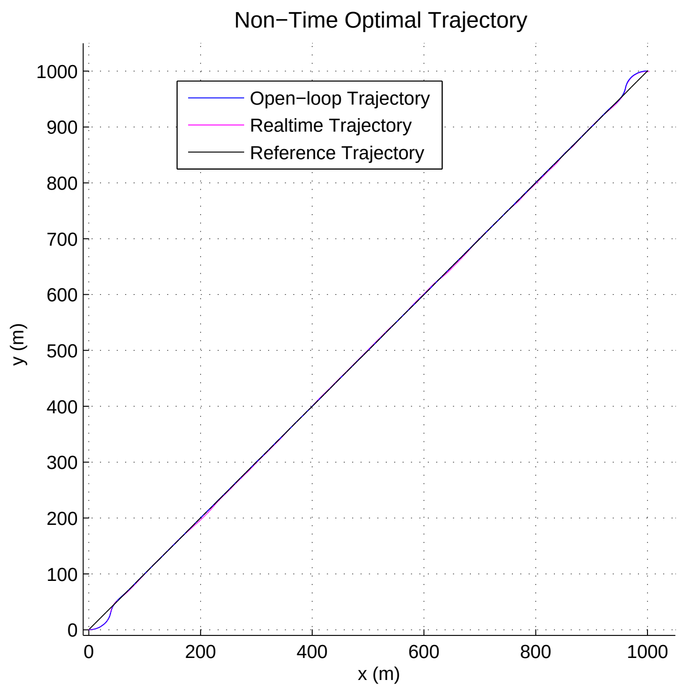


Figure 4.50: Realtime vs Open-loop Maneuver in XY Plane

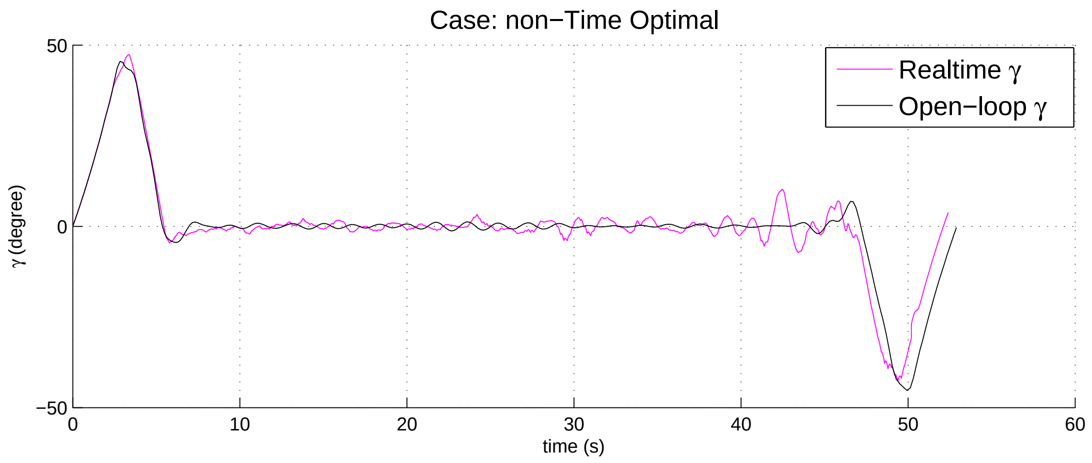


Figure 4.51: Realtime vs Open-loop γ angle

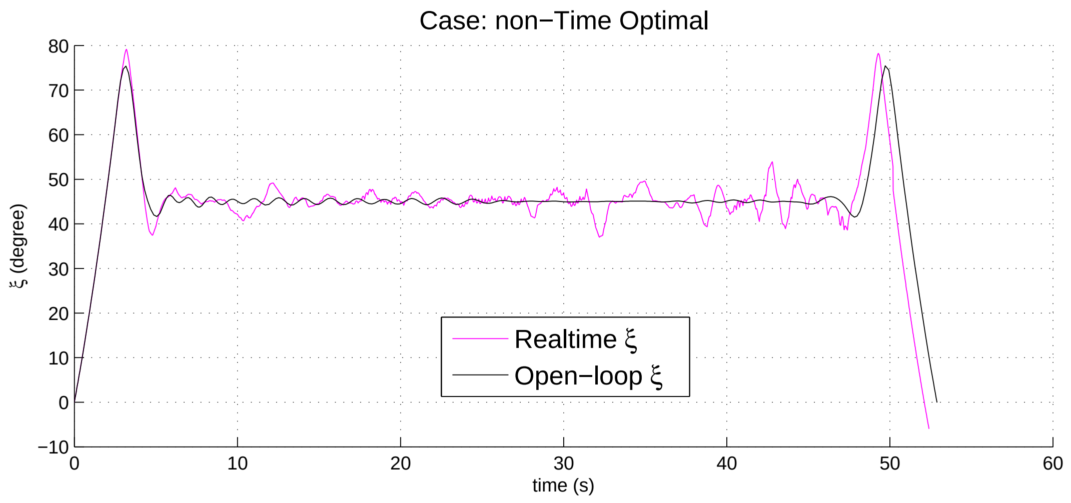


Figure 4.52: Realtime vs Open-loop ξ angle

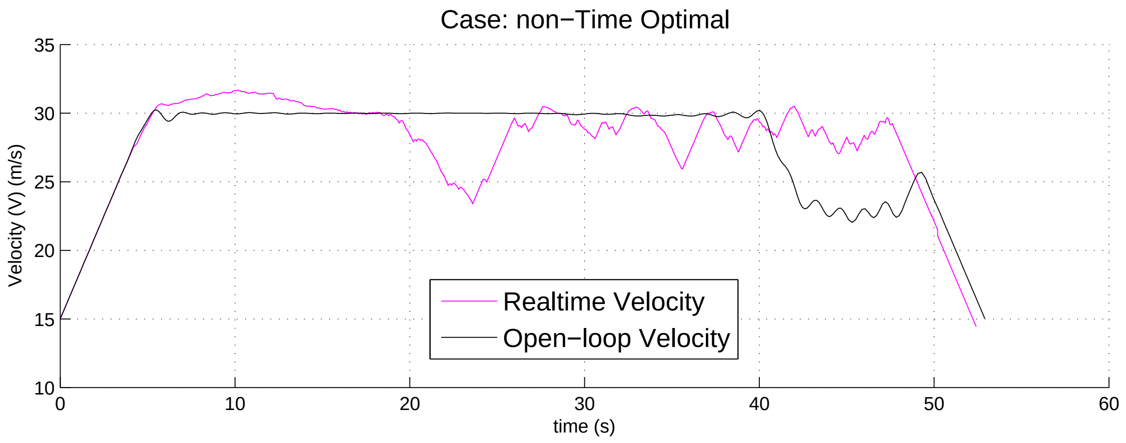
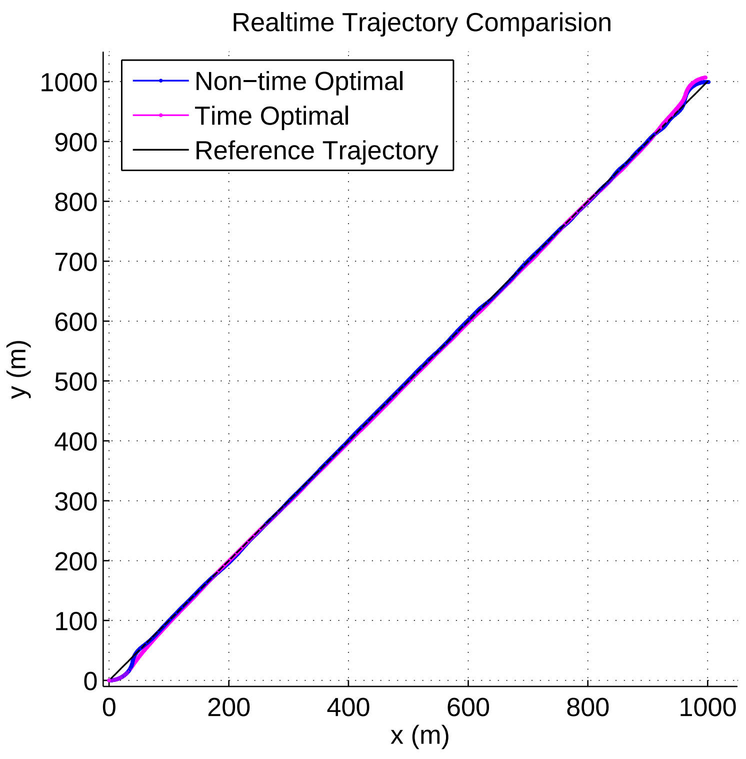


Figure 4.53: Realtime vs Open-loop (V) Velocity

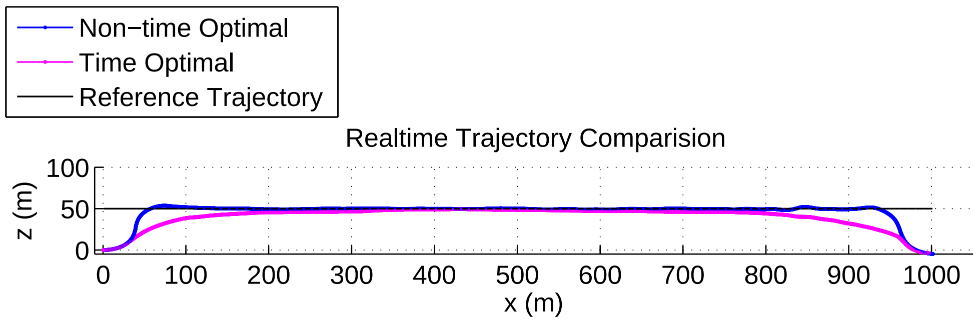
From Figure 4.53 it can be seen that the realtime velocity goes higher than its upper limit, this is due to the propagation error when controls applied are at the upper limits, similar to that of Figure 4.29. When any of the states are out of the required range the tool (DIDO) provides an infeasible solution with controls within the bounds given to the problem, these controls tend to pull the values of the states within the bounds by lowering the values of controls. This is demonstrated in Figure 4.53.

Conclusion:

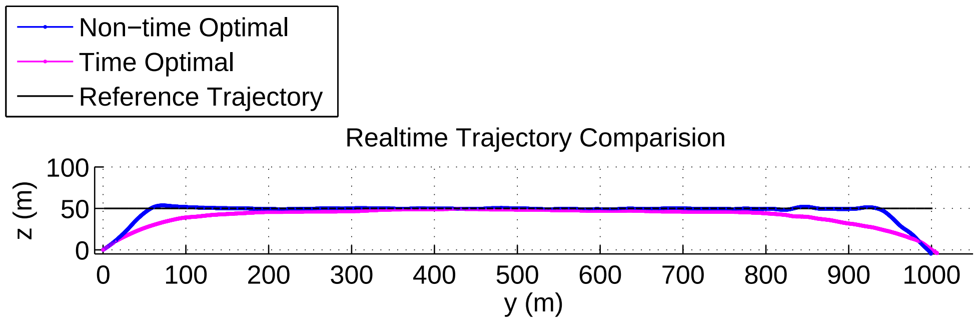
Figure 4.54 compares the real-time time-optimal and non time-optimal trajectories. It can be seen that the non time-optimal solution follows the reference trajectory closer than the time-optimal one. Table 4.1 provides a comparison between all four scenarios. Despite the lower computation and slightly lower maneuver time in the time optimal case, the non-time optimal formulation is chosen for realtime trajectory generation through out this thesis, due to the fact that it generates a trajectory that follows the reference trajectory significantly closer.



(a) XY Maneuver



(b) XZ Maneuver



(c) YZ Maneuver

Figure 4.54: Comparison of the trajectory generated.

Table 4.1: **Scenario I** Computational Summary.

Case	Node	Computation Time	Cost	Maneuver Time
IA	100 (offline)	$229.8s$	0.8523(without t_f)	$50.2919s$
IB	100 (offline)	$388.05s$	0.6558	$52.8946s$
IC	30, 5 Hz (realtime)	$4.1026s$ (a.c.t)[1]	-	$50.2512s$
ID	30, 5 Hz (realtime)	$9.8170s$ (a.c.t)	-	$52.7172s$

4.2 Scenario II. With Stationary Obstacles

Scenario II presents the case where obstacles are stationary, and their positions and shapes are known beforehand. The detailed scenario is shown in Figure 4.55. This section checks the method's capability to avoid stationary obstacles while remaining close to the predefined trajectory. As explained in Section 3.4, two stationary obstacles are placed with base centered at $(350, 300, 0)$ m and $(600, 650, 0)$ m. These two obstacles form path constraints as explained in Section 3.3. From equation (3.22), the path constraints due to the obstacles are given by

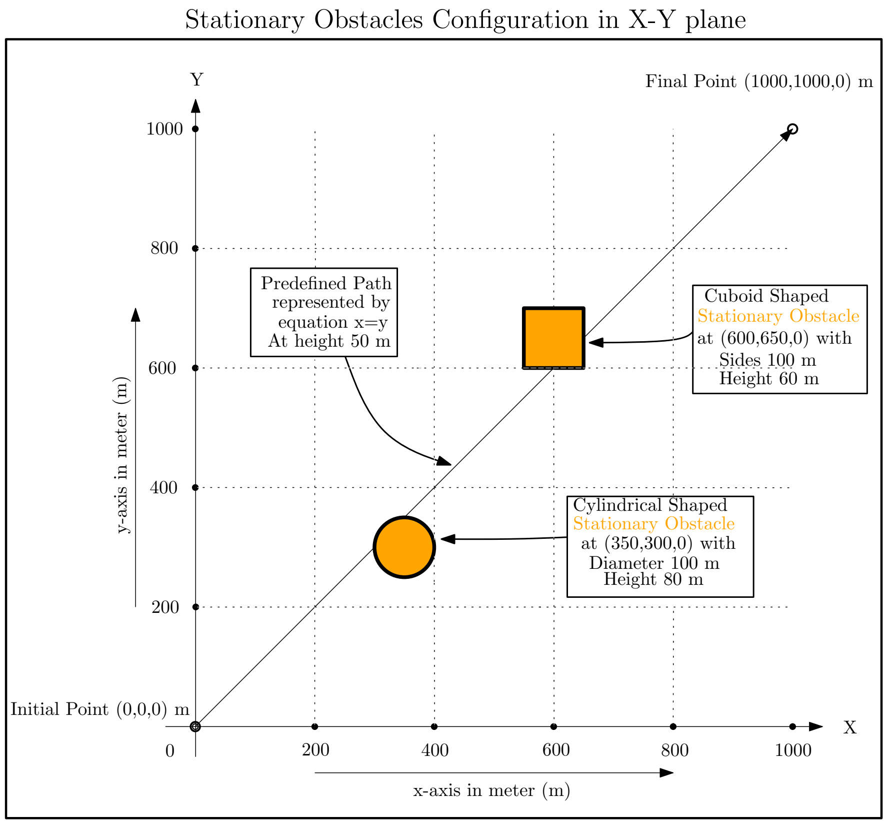
$$h_1(x,y,z) := ln\left[\left|\left(\frac{x-350}{50}\right)^2\right| + \left|\left(\frac{y-300}{50}\right)^2\right| + \left|\left(\frac{z-40}{40}\right)^8\right|\right] \geq 0 \tag{4.3}$$

and

$$h_2(x,y,z) := ln\left[\left|\left(\frac{x-600}{50}\right)^8\right| + \left|\left(\frac{y-650}{50}\right)^8\right| + \left|\left(\frac{z-30}{30}\right)^8\right|\right] \geq 0 \tag{4.4}$$

As explained in Section 3.3, when obstacles are present, the cost function includes a corresponding robustness factor. The effect of adding the robustness factor to the cost is that the optimal trajectory is pushed away from the obstacles, which is beneficial from the point of view of creating a safer trajectory. On the other hand, it means that the trajectory does not truly minimize the area between the optimal and reference trajectories.

[1]average computation time per solution at a sample time



Stationary Obstacles Configuration in X-Y plane
Y
Final Point (1000,1000,0) m
1000
800
600
400
200
0
200
400
600
800
1000
X
Predefined Path represented by equation x=y At height 50 m
Cuboid Shaped Stationary Obstacle at (600,650,0) with Sides 100 m Height 60 m
Cylindrical Shaped Stationary Obstacle at (350,300,0) with Diameter 100 m Height 80 m
y-axis in meter (m)
Initial Point (0,0,0) m
x-axis in meter (m)

Figure 4.55: Scenario II

IIA. Open-loop:

Equation (3.27) defines the problem statement for **Scenario IIA** with the obstacles' constraints given in equations (4.3) and (4.4) and the cost function given by equation (4.5).

$$J(\underline{X}(\cdot), \underline{U}(\cdot)) = \int_{t_0}^{t_f} [r(x,y,z)]dt + A1 + A2 \tag{4.5}$$

where, from equation (3.25), with $w = 0.5$, the robustness function is given by

$$r(x,y,z) = 0.5\left[(e^{e^{-h_1(x,y,z)}} - 1) + (e^{e^{-h_2(x,y,z)}} - 1)\right] \tag{4.6}$$

The problem is solved with 100 nodes giving a total cost of $J = 2.8328$ and $t_f = 57.1669$ s as the total maneuver time. Figures 4.56-4.62 show the state and control trajectories respectively. It can be seen that the optimal trajectory stays close to the reference trajectory, while avoiding the obstacles.

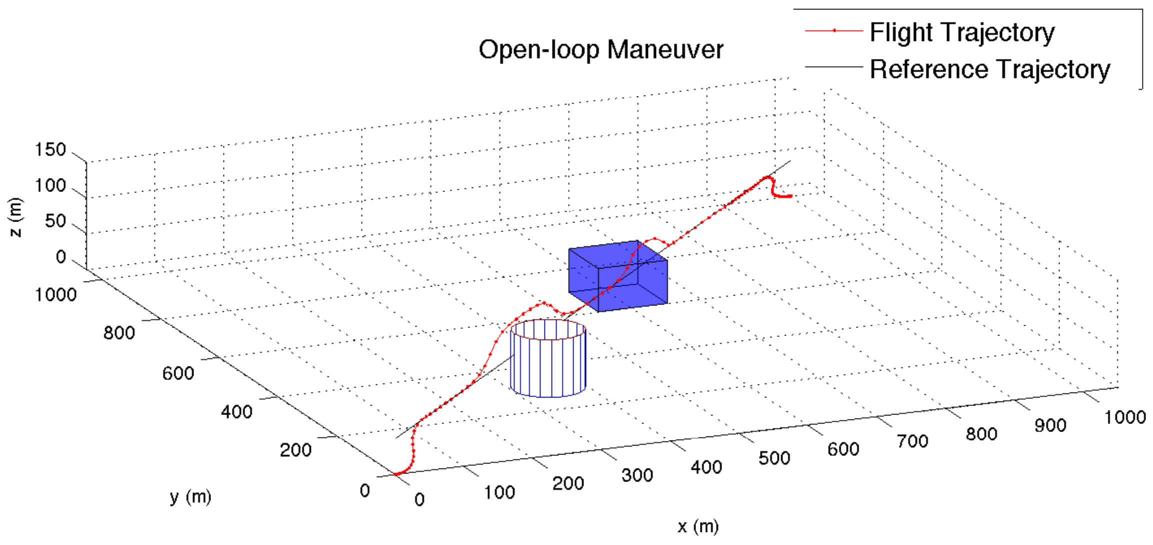


Figure 4.56: Isometric View of Total Maneuver

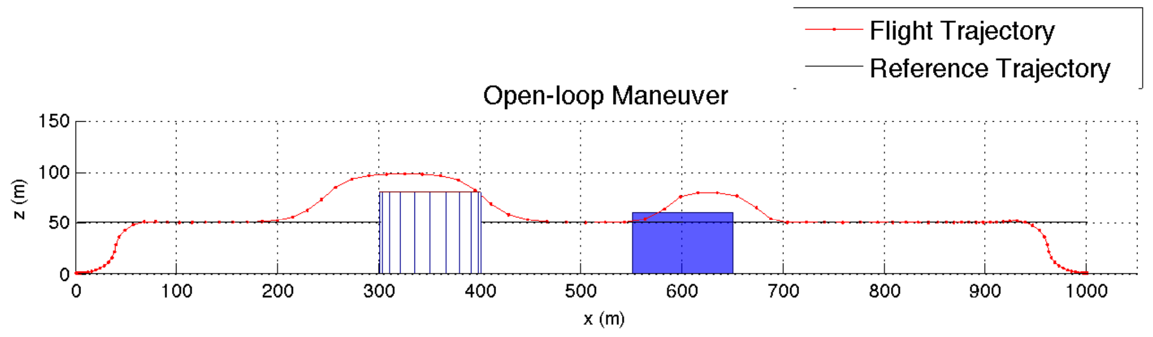


Figure 4.57: View in XZ Plane

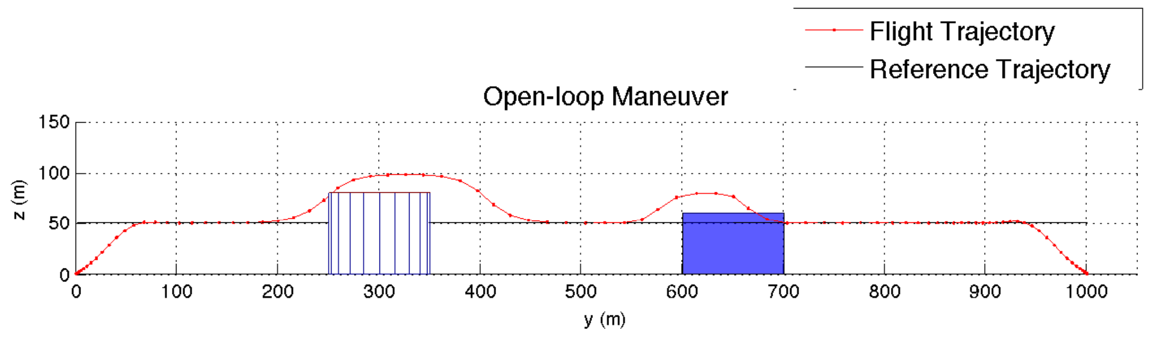


Figure 4.58: View in YZ Plane

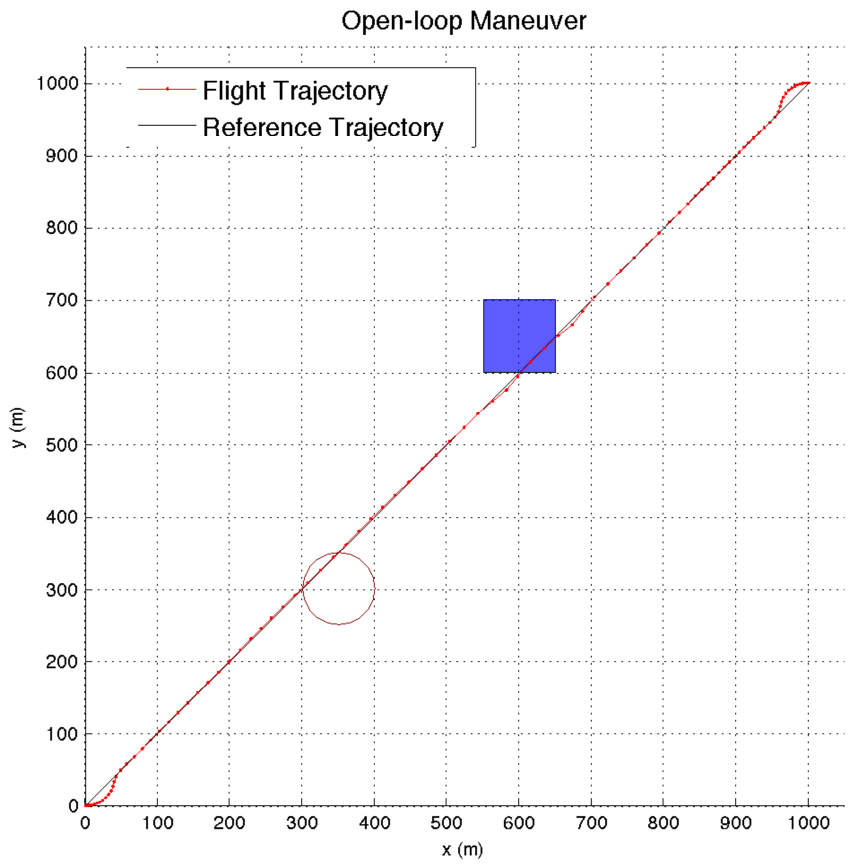


Figure 4.59: View in XY Plane

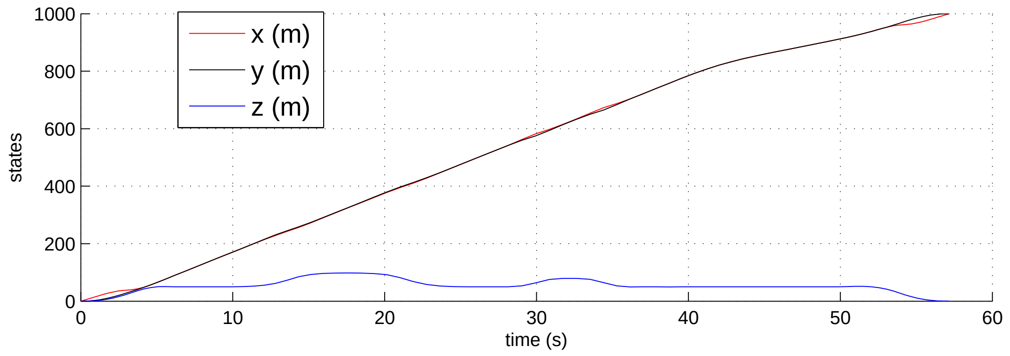


Figure 4.60: States $[x, y, z]$ vs time (t)

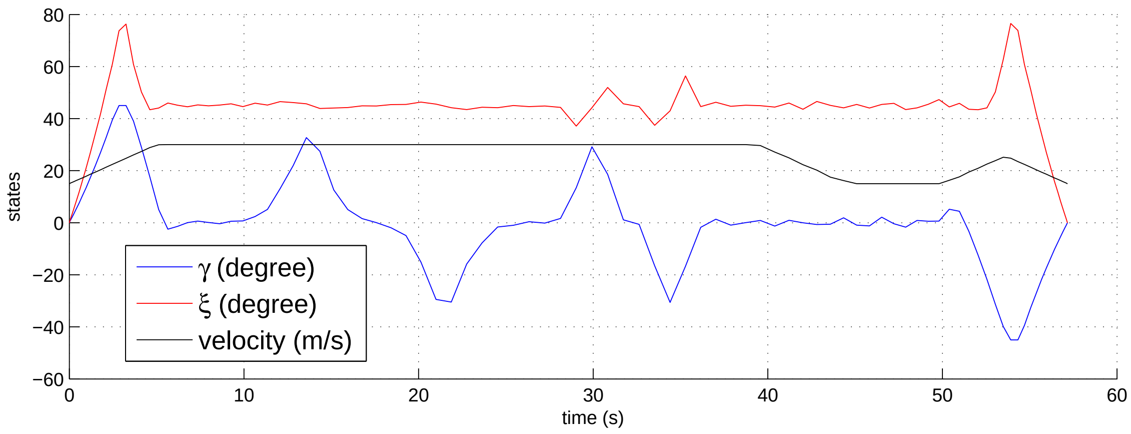


Figure 4.61: States $[\gamma, \xi, \mathrm{V}]$ vs time (t)

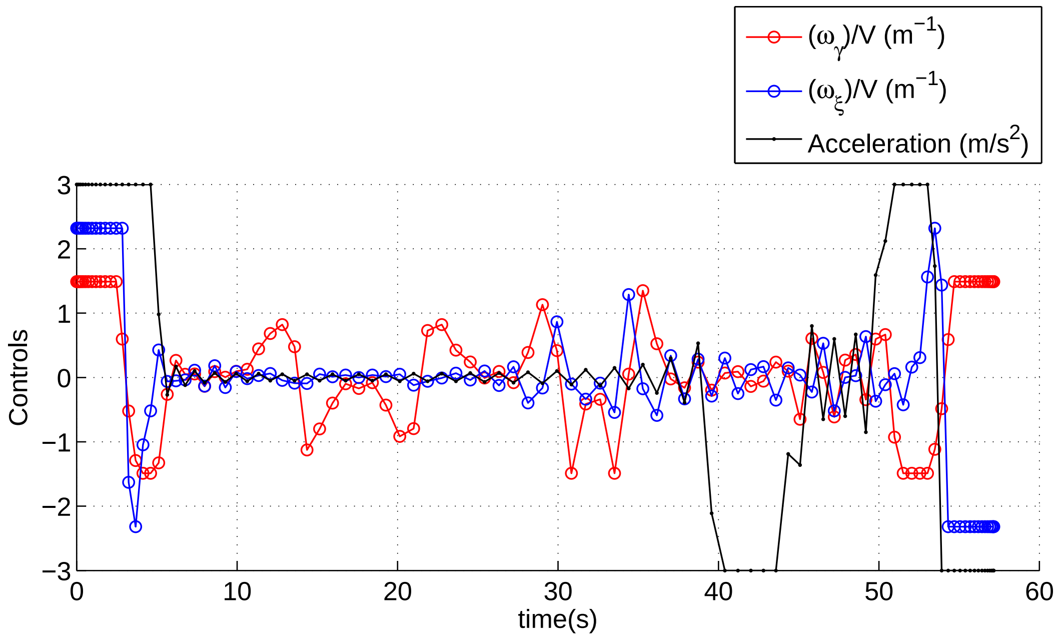


Figure 4.62: Control Trajectory

Feasibility Check:

Figures 4.63 to 4.69 show the feasibility check for **Scenario IIA**. It can be seen that the Runge-Kutta propagated solution follows the DIDO solution very closely.

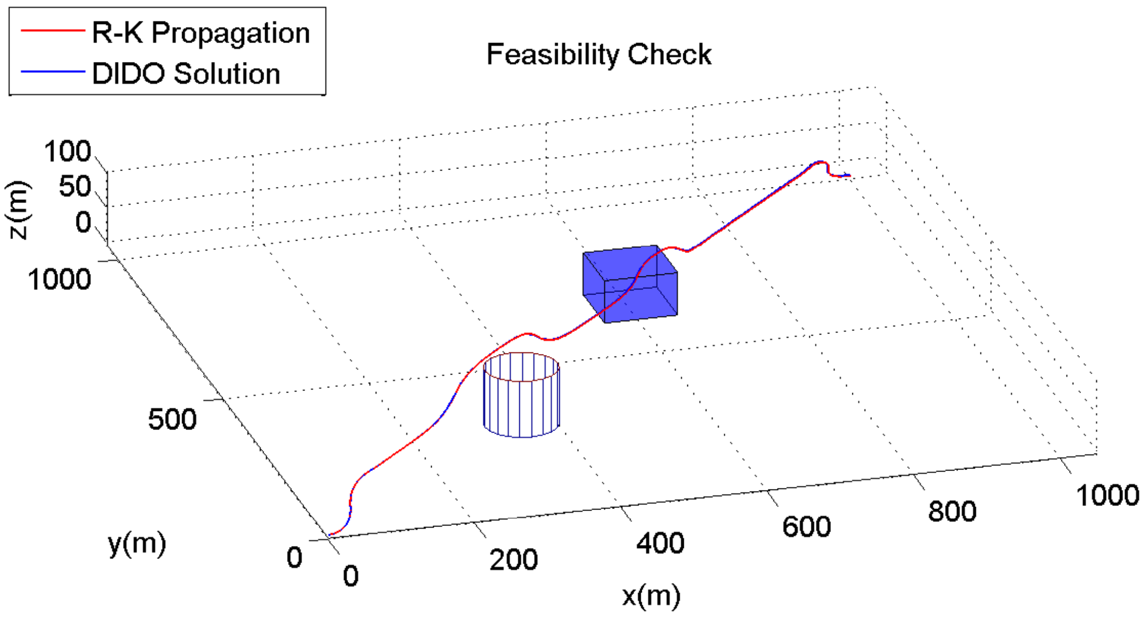


Figure 4.63: Isometric View of Complete Maneuver

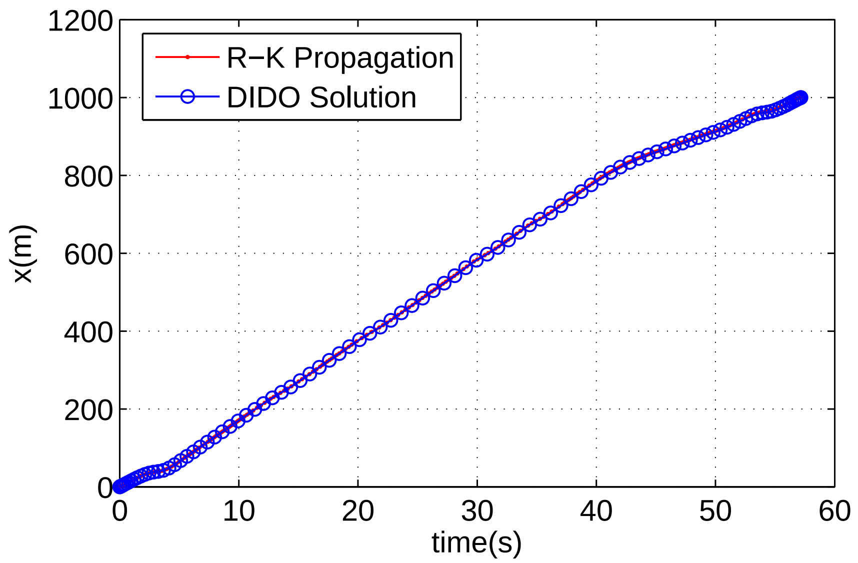


Figure 4.64: Feasibility Check (x vs t)

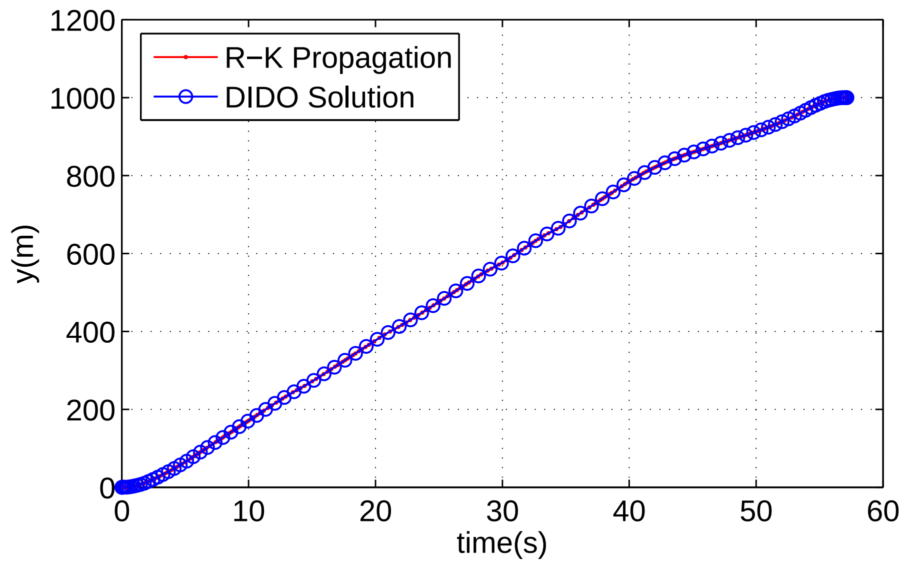


Figure 4.65: Feasibility Check (y vs t)

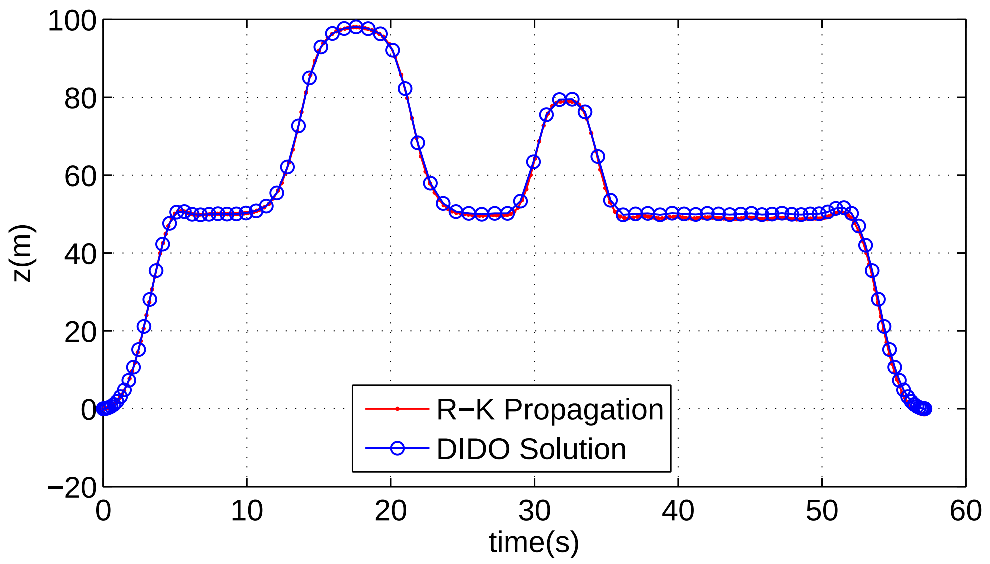


Figure 4.66: Feasibility Check (z vs t)

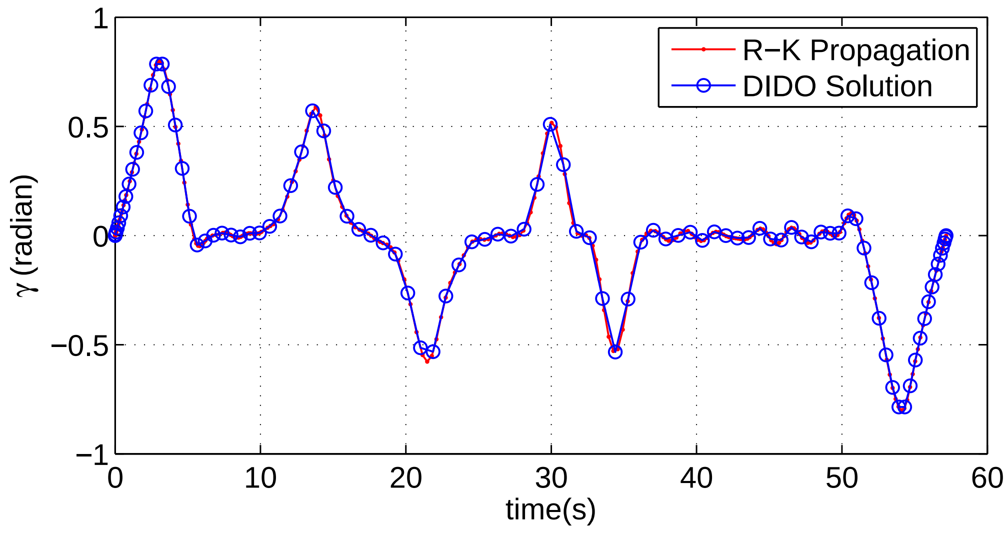


Figure 4.67: Feasibility Check (γ vs t)

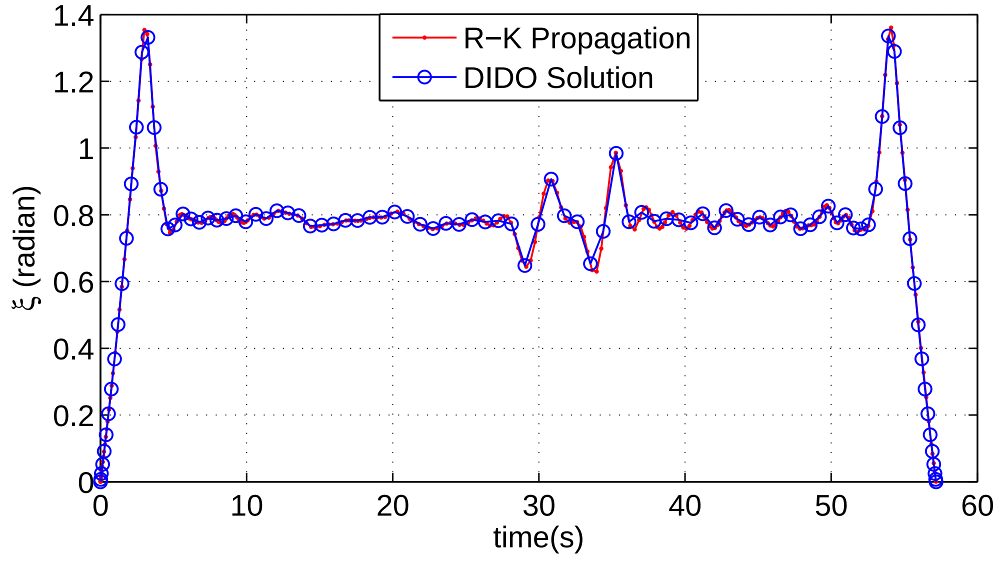


Figure 4.68: Feasibility Check (ξ vs t)

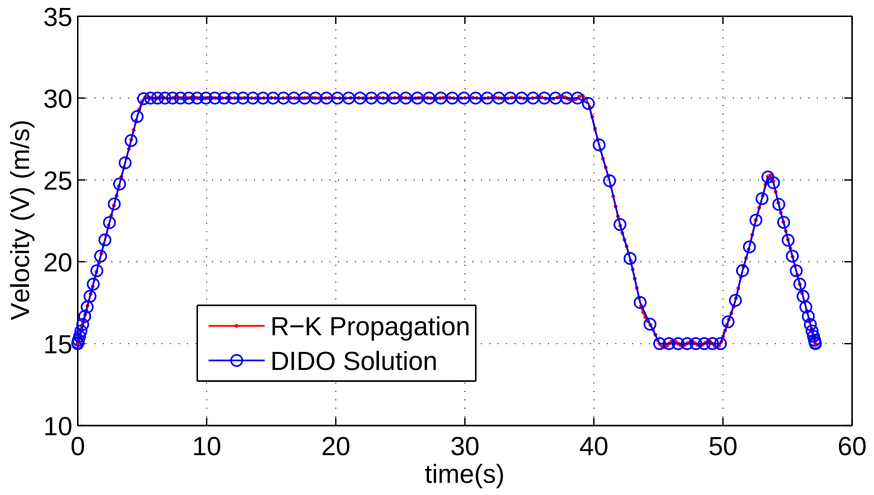


Figure 4.69: Feasibility Check (V vs t)

Hamiltonian Value Condition:

Similar to that of **Scenario IB**, Figure 4.70 shows Pontryagin's Hamiltonian value is nearly constant with value $H^*(\tau) = 0$, as required.

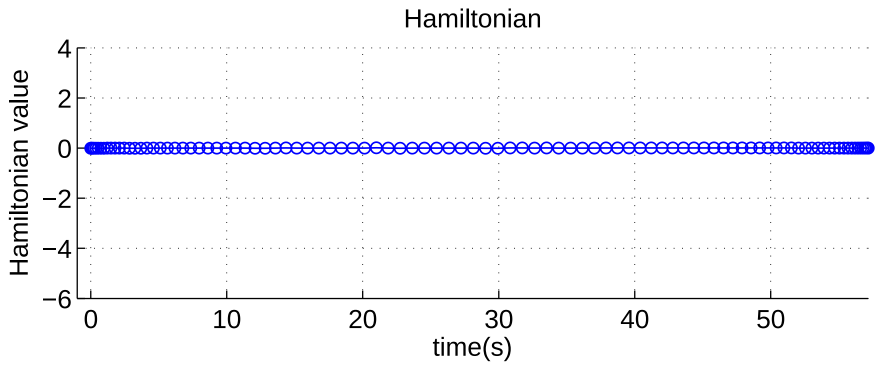


Figure 4.70: Hamiltonian Value

Bellman Principle Test:

The problem is broken down into two halves. The first half is from $(0, 0, 0)$ m to $(8.0750, 8.0762, 0.499)$ m and the second from $(8.0750, 8.0762, 0.499)$ m to $(1000, 1000, 0)$ m. The cost and time taken for these two halves were noted.

Figures 4.71-4.74 show the states trajectories of the original problem with the 1^{st}half and 2^{nd} half problem superimposed on the same graph. The cost of the 1^{st} half problem was $J_1 = 2.4670$ with 50 nodes; the cost of the 2^{nd} half problem was $J_2 = 0.3482$ also with 50 nodes; and addition of the 1^{st} and 2^{nd} halves gives $J_1 + J_2 = 2.81$ which almost matches the original problem cost of $J = 2.83$. The difference in the costs is due to the node distribution scheme and the dividing point of the original trajectory. Figure 4.75 shows an example of the node distribution if the original trajectory is divided into two from the midpoint of the interval $[0, 10]$.

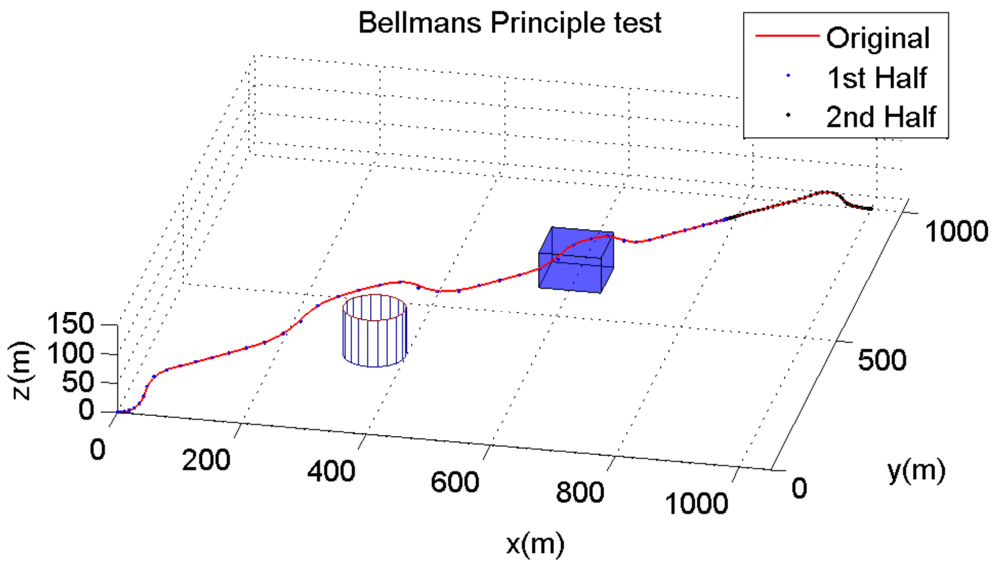


Figure 4.71: Bellman's Test in 3D Maneuver

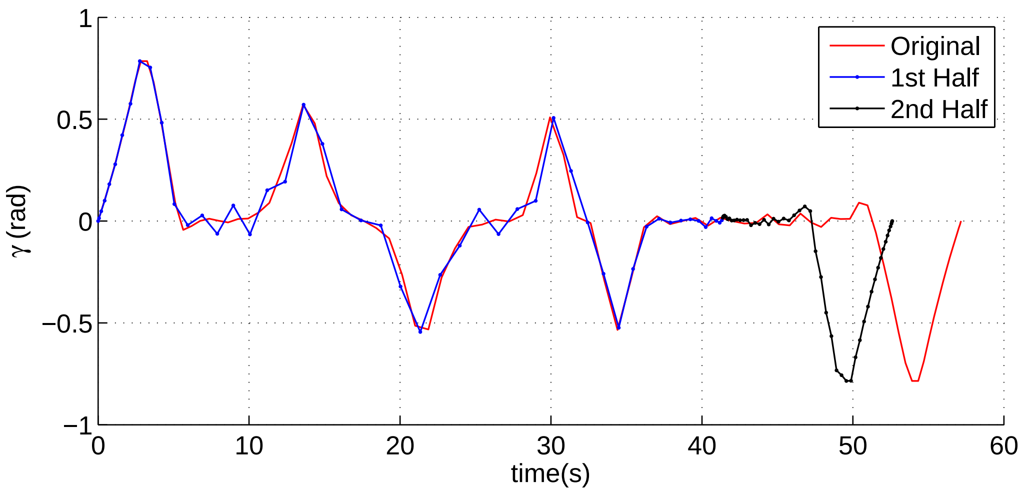


Figure 4.72: Bellman's Principle Test (γ vs t)

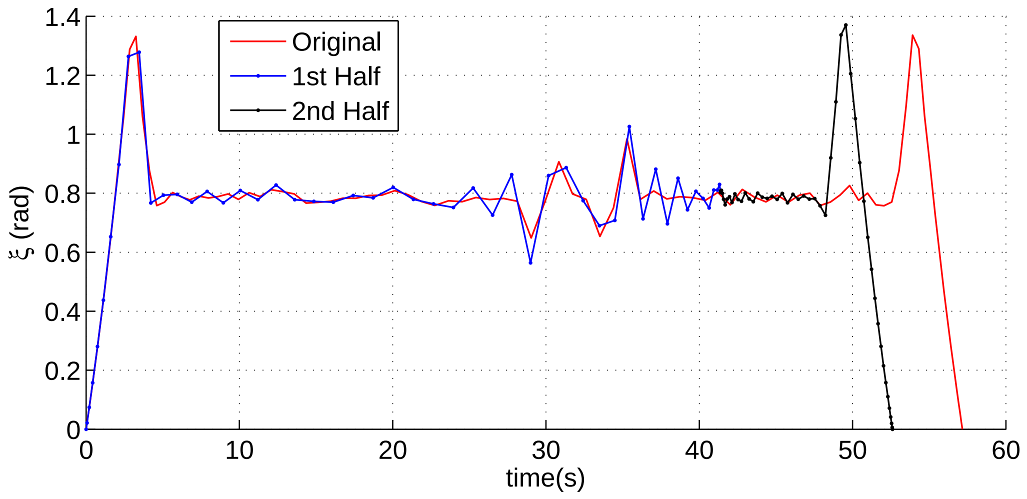


Figure 4.73: Bellman's Principle Test (ξ vs t)

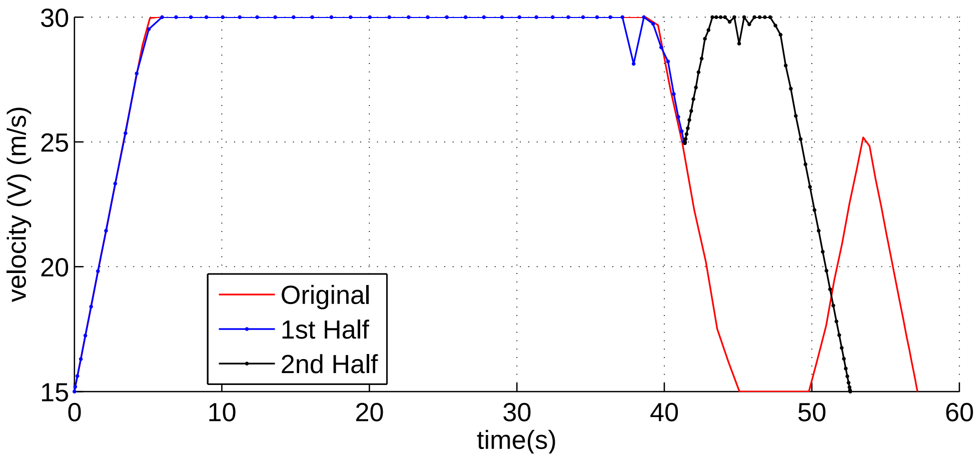


Figure 4.74: Bellman's Principle Test (V vs t)

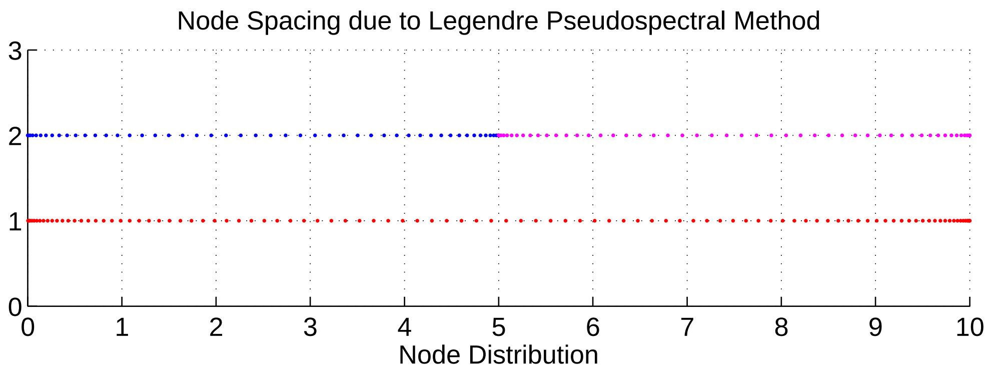
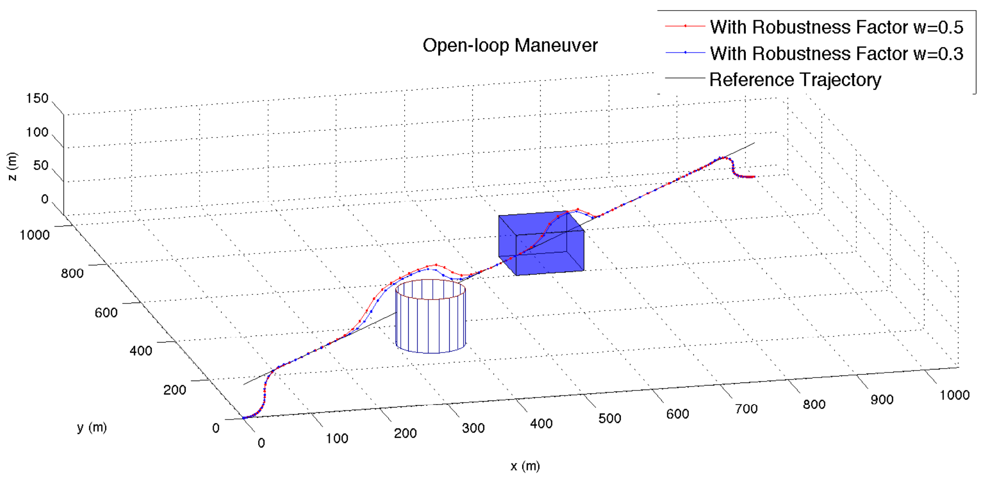


Figure 4.75: Node Spacing Example

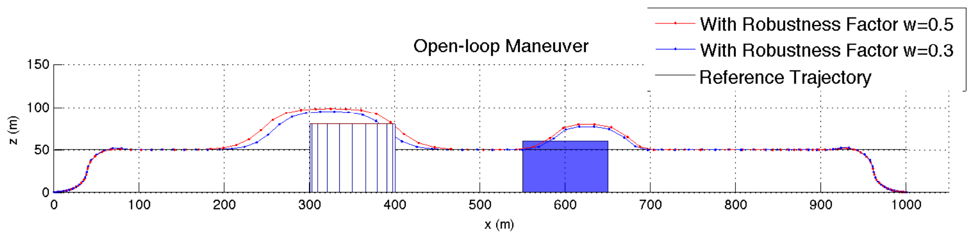
Discrepancies in Figures 4.72 to 4.74 are because of the polynomial approximation of the control in equation (2.24), which actually means that the control is required to be continuous over the entire trajectory. This includes the acceleration. Hence, sharp changes in velocity are not possible. However, when we break the trajectory into two, as in the Bellman test, we introduce the possibility of a discontinuity in the controls at the break point. From Figure 4.74, this is exactly what has happened to the acceleration. This discontinuity in velocity has allowed the 2^{nd} half trajectory some freedom in seeking a trajectory that gives a lower total cost. This can be seen in the costs $J_1 + J_2 < J_3$.

IIA (a). Change of Robustness Factor (w)

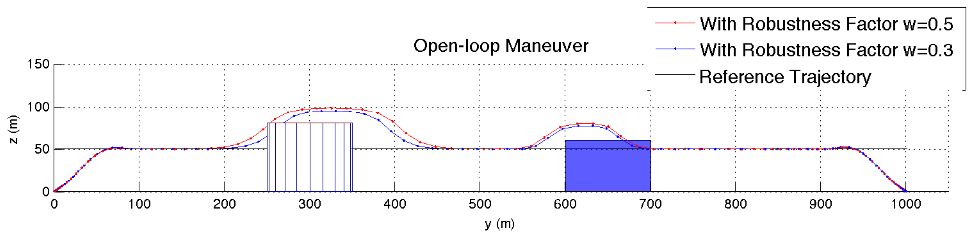
As mentioned previously, the robustness factor pushes the resulting optimal trajectory away from the obstacles. In this subsection, we compare the trajectory generated with one generated with a reduced robustness factor (w). The results are shown in Figure 4.76. As expected, the trajectory generated with a reduced robustness factor stays closer to the reference trajectory. However, it also passes closer to the obstacles, and hence is less safe.



(a) 3D Maneuver



(b) XZ Maneuver



(c) YZ Maneuver

Figure 4.76: Comparison of Trajectory with different Robustness Factor (w).

IIB. Realtime, $(w = 0.5)$:

Scenario IIB contains the real-time solution of the problem formulated in **Scenario IIA**. In this case, the sampling frequency is 5 Hz, and 40 nodes are used to compute the solution at each sample instant. Compared to **Scenarios IC** and **ID**, the number of nodes is increased, since it was found that 30 nodes did not perform sufficiently well. Figures 4.77 to 4.83 show the resulting real-time trajectories, with the corresponding open-loop trajectories shown for comparison. It can be seen that spatially (x, y, z), the real-time and open-loop trajectories are quite similar.

The average time for solving each realtime problem is 67.3459 s.

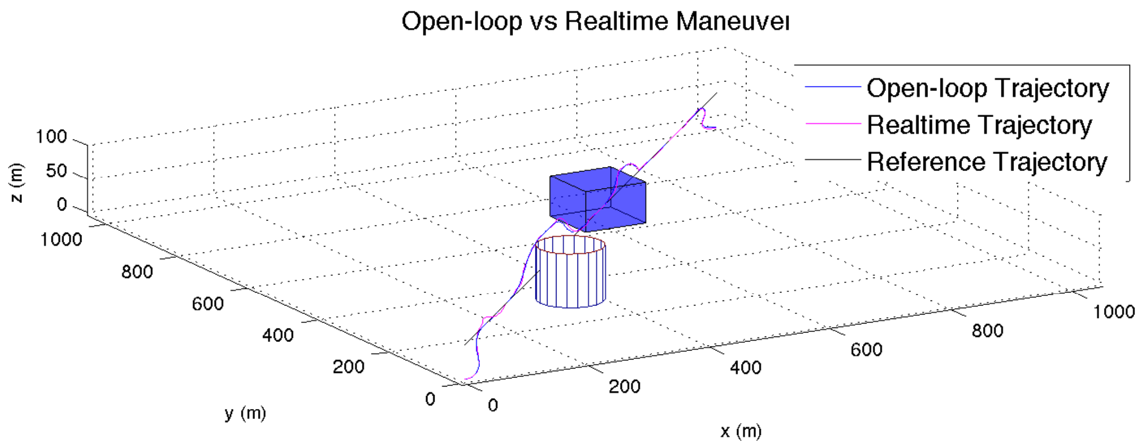


Figure 4.77: 3D Maneuver

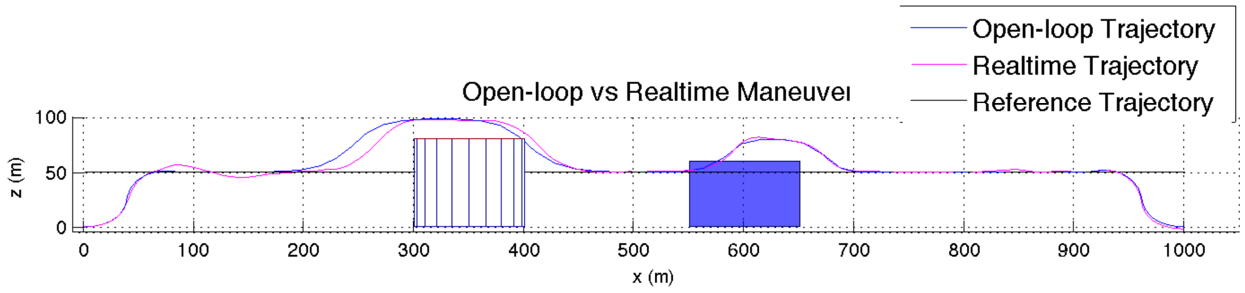


Figure 4.78: XZ Maneuver

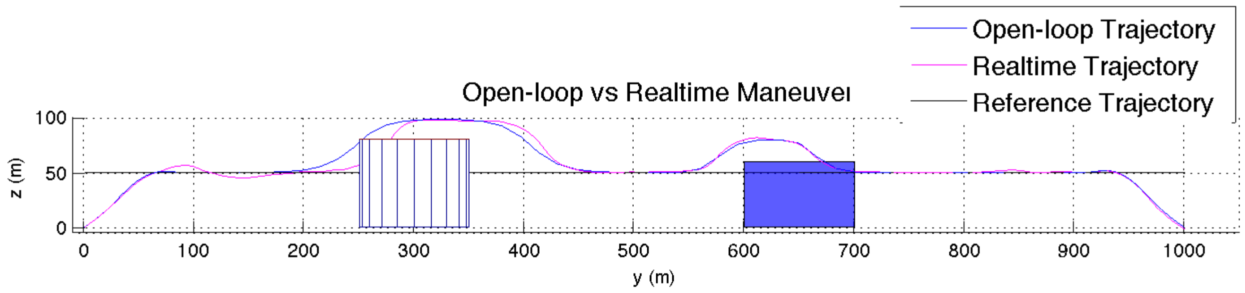


Figure 4.79: YZ Maneuver

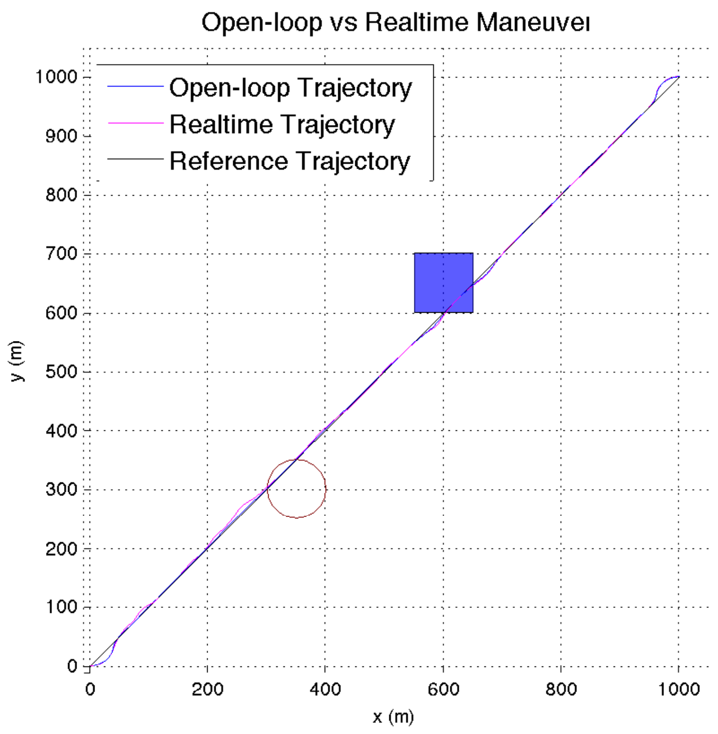


Figure 4.80: XY Maneuver

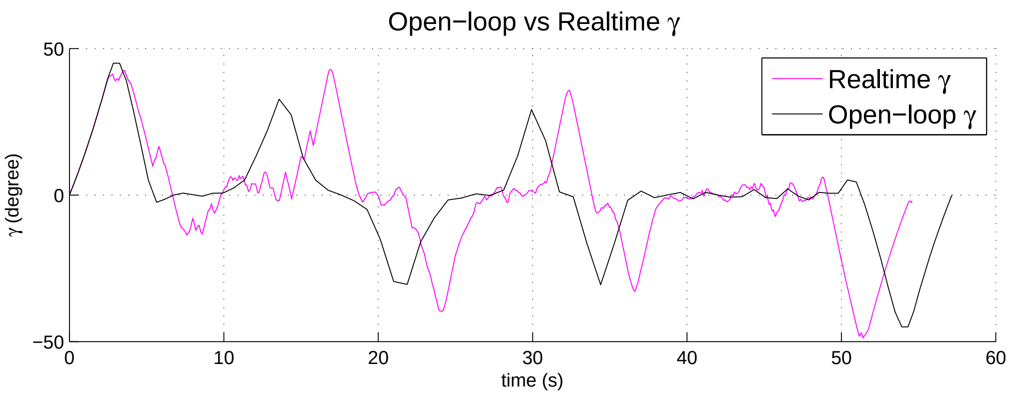


Figure 4.81: γvs t

In Figure 4.81, real-time γ actually goes outside the allowable range at about 52 s, this is due to propagation error similar to that of Figure 4.53.

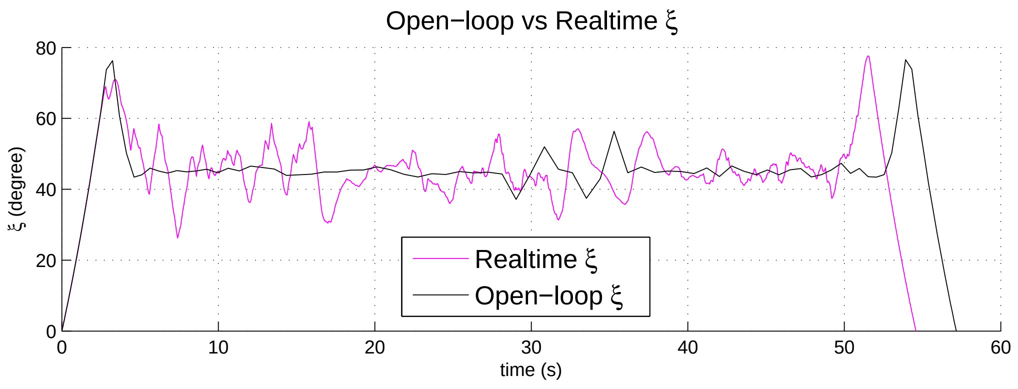


Figure 4.82: ξ vs t

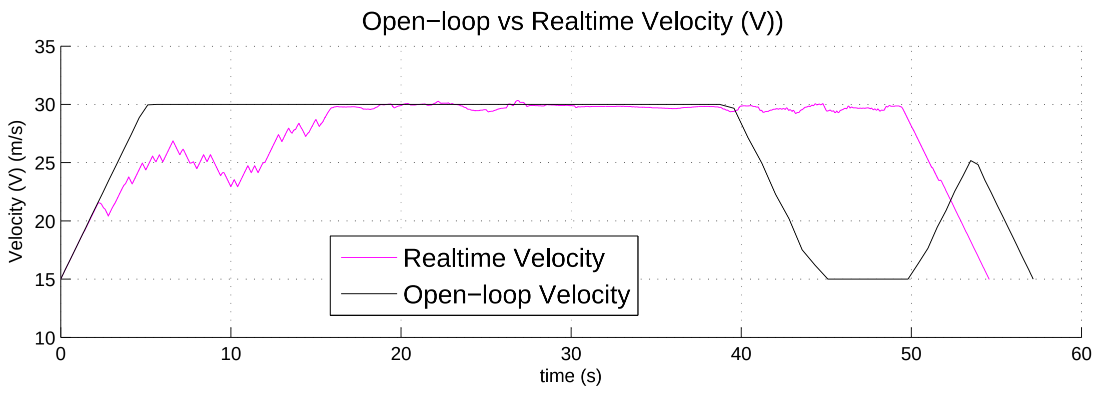


Figure 4.83: Velocity (V) vs t

IIC. Realtime, $(w = 0.3)$:

Scenario IIC is a case similar to **IIB** but with decreased robustness factor $(w = 0.3)$. Figures 4.84 to 4.87 compares the **IIC** real-time trajectory with **IIB** realtime trajectory. It shows that **IIC** remains close to reference trajectory and also decreasing maneuver and computation time.

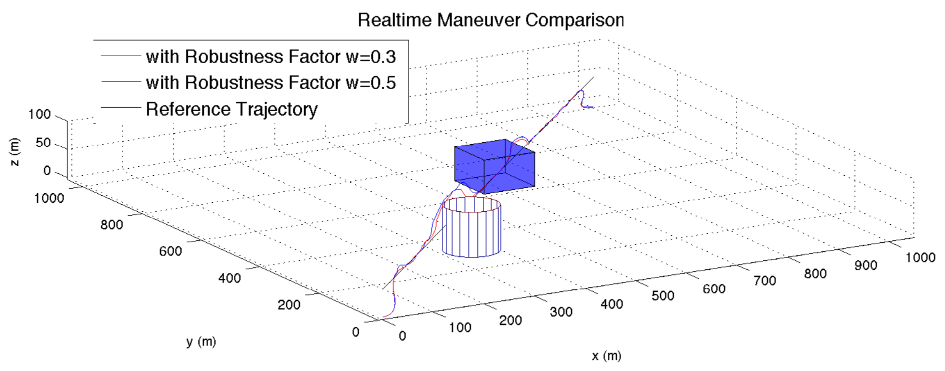


Figure 4.84: Trajectory comparison with different robustness factor (3D view)

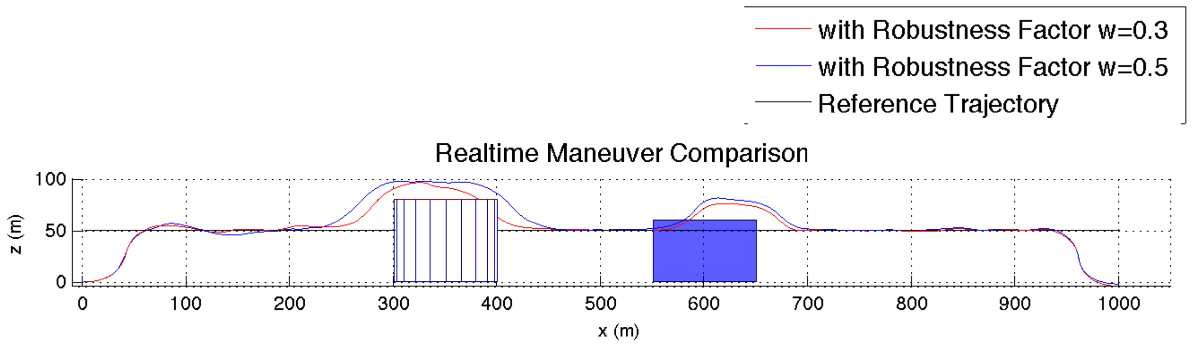


Figure 4.85: Trajectory comparison with different robustness factor (XZ Plane)

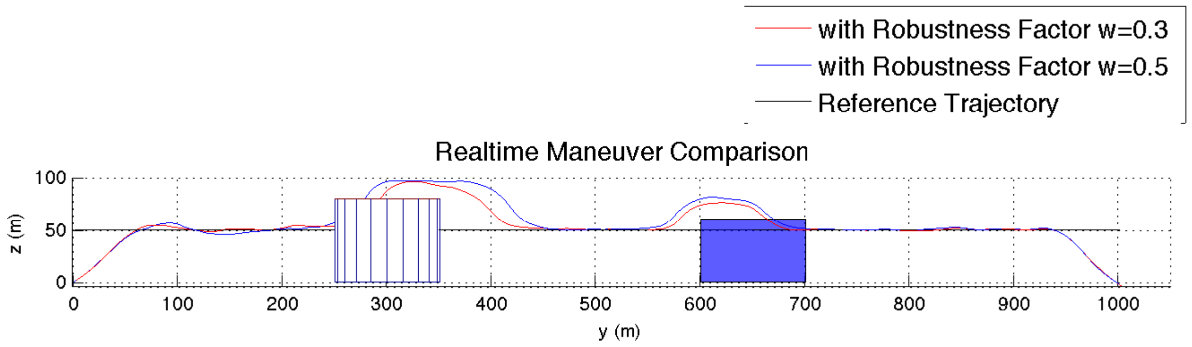


Figure 4.86: Trajectory comparison with different robustness factor (YZ Plane)

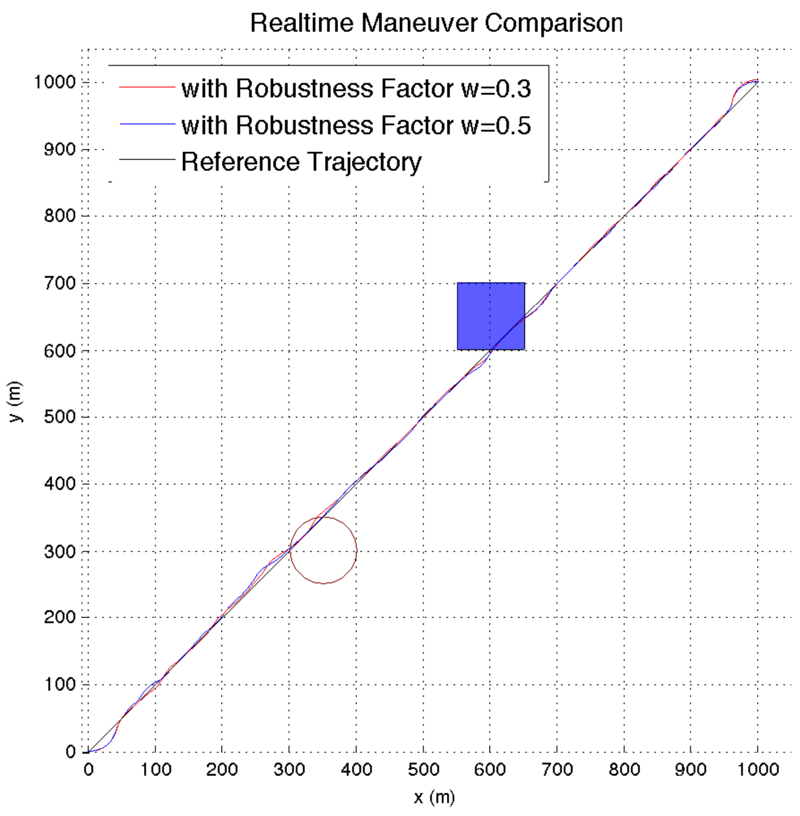


Figure 4.87: Trajectory comparison with different robustness factor (XY Plane)

Table 4.2 summarizes the results for **Scenario II.**

Table 4.2: **Scenario II** Computational Summary

Case	Node	Computation Time	Cost	Maneuver Time
IIA	100 (offline) ($w = 0.5$)	2228.2 s	2.8328	57.1669 s
IIA (a)	100 (offline) ($w = 0.3$)	1238.5 s	2.2659	52.7784 s
IIB	40, 5 Hz (realtime) ($w = 0.5$)	67.3459 s (a.c.t)[2]	-	54.5815 s
IIC	40, 5 Hz (realtime) ($w = 0.3$)	48.652 s (a.c.t)	-	49.6174 s

It can be seen that with $w = 0.3$ the computation time was significantly less. hence a way to recover some safety while reducing computation time would be to reduce the weighting factor, but increase the size of the modeled obstacle (make it larger than the true obstacle).

[2]average computation time per solution at a sample time

4.3 Scenario III. Moving Obstacle

After the successful implementation of Legendre Pseudospectral method in **Scenario I** and **Scenario II**, we now test the method with moving obstacles. Because the motion of the obstacles is not known beforehand, it must be implemented in real-time. Since, this method works on the basis of obstacles seen at each snapshot taken, it is necessary that the obstacles be detected. Regarding the avoidance of moving obstacles, the more rapidly the obstacle is moving and/or the later the obstacle is detected, the quicker the on-board computation of a new trajectory and the more maneuverable the aircraft needs to be. To test this method for moving obstacles, two different cases are considered. First, the obstacle moves in a straight line and second in a curved path as defined in the subsections below.

IIIA. Realtime: An Obstacle Moving in Straight Line

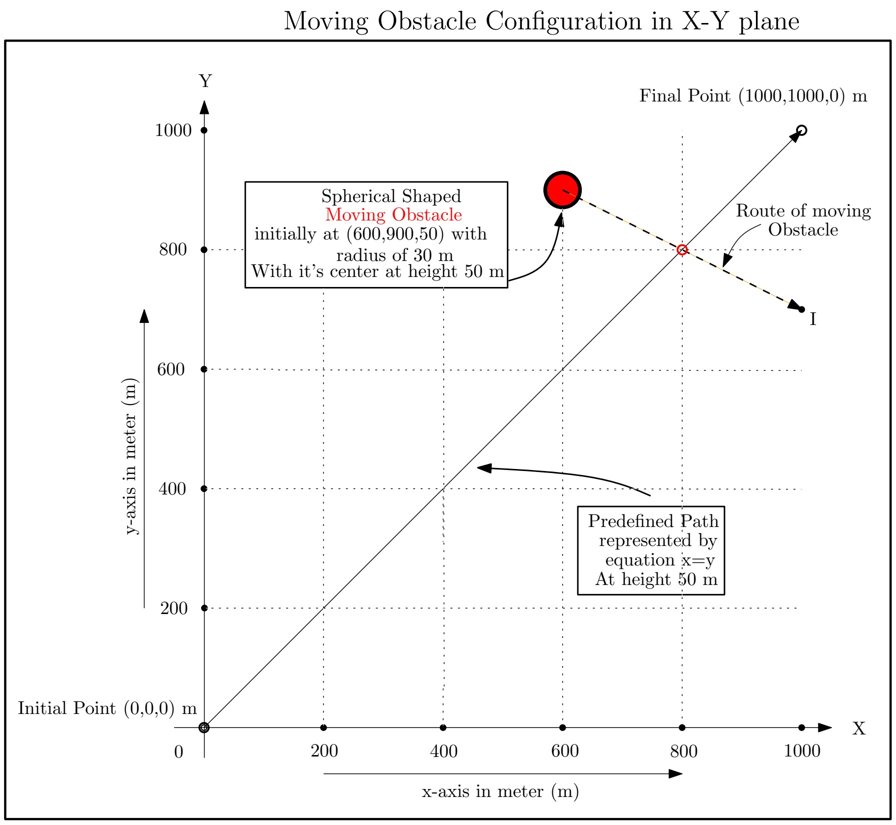


Figure 4.88: Scenario IIIA

In this case the spherical shaped obstacle has its center initially stationed at $(600, 900, 50)$ m, and starts moving after $32.7s$ of simulation time to meet the UAV at $(800, 800, 50)$ m. It is difficult to predict the time that the UAV would arrive at (800,800,50) m, but from the realtime simulation result of **Scenario ID**, it is assumed that the UAV reaches $(800, 800, 50)$ m at 40.7 s. Hence, the obstacle's path is designed in such a way that it remains stationary till 32.7 s, and start moving towards the reference trajectory to reach $(1000, 700, 50)$ m, after 16 s. On its way the obstacle passes through $(800, 800, 50)$ m at 40.7 s, that is 8 s after starting to move.

Equation (4.7) shows the obstacle constraint as the function of time.

$$h_1(x,y,z) := ln\left[\left|\left(\frac{x-100(6+\frac{(t-32.7)}{4})}{30}\right)^2\right| + \dots\right.$$

$$\left.\left|\left(\frac{y-100(9-\frac{(t-32.7)}{8})}{30}\right)^2\right| + \left|\left(\frac{z-50}{30}\right)^2\right|\right] \geq 0 \tag{4.7}$$

$$32.7 \leq t \leq 48.7$$

However, it must be emphasized that the UAV is assumed to have knowledge only of the obstacle position at each sample instant. Hence, at a given sample instant, the new trajectory is computed with t in (4.7) held fixed at the current sample time.

With the path constraint in (4.7), the cost function for **Scenario IIIA** in equation (3.27) becomes

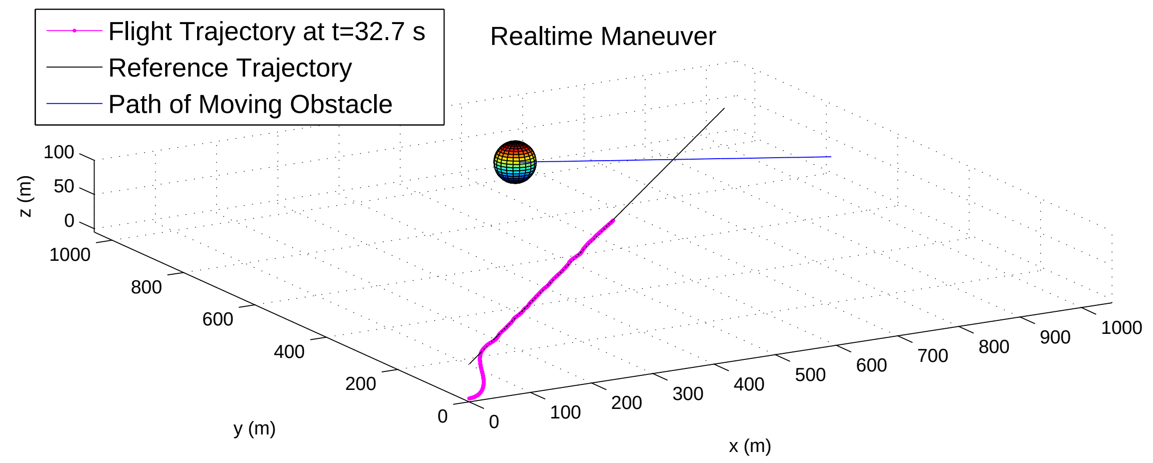
$$J(\underline{X}(\cdot), \underline{U}(\cdot)) = \int_{t_0}^{t_f} [r(x,y,z)]dt + A1 + A2 \tag{4.8}$$

where, from equation (3.25), with $w = 0.5$, the robustness function is given by

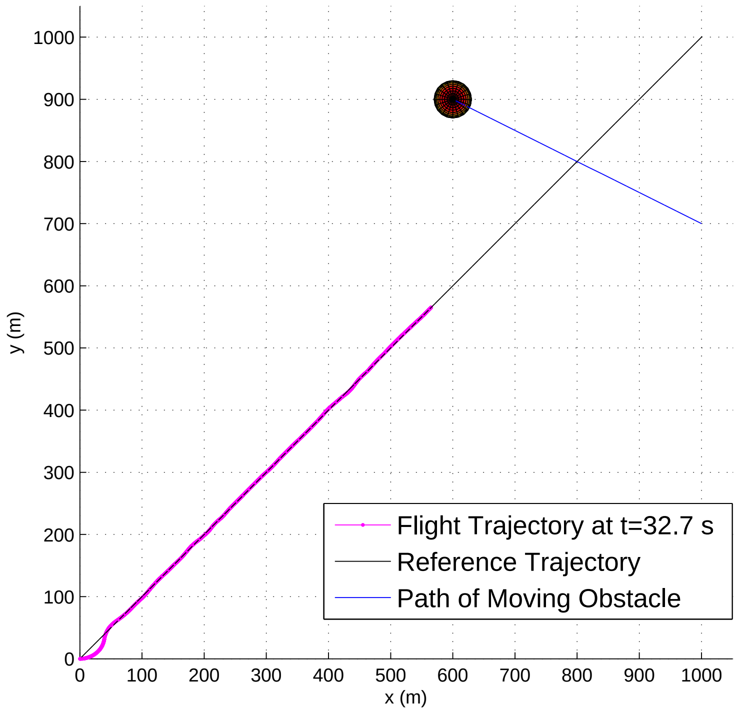
$$r(x,y,z) = 0.5\left[(e^{e^{-h_1(x,y,z)}} - 1)\right] \tag{4.9}$$

In case of moving obstacle, it was found to be necessary to increase the robustness factor (w) because it was found that with the reduced robustness factor, each real-time trajectory (which was obtained under the assumption that the obstacle is stationary) passed too close to the obstacle, and since the obstacle was actually moving, collision occurred.

As for **Scenario IIB**, the sampling frequency is 5 Hz, and a 40 nodes are used for each real-time solution. The resulting state trajectories are shown in Figures 4.89 to 4.98. The average time for solving the realtime problem is 27.329 s and the total maneuver time is 55.54 s. As shown in the figures, the UAV successfully avoids the moving obstacle. Of particular interest is figure 4.93, where the UAV is avoiding the obstacle after it has already passed. This is because the realtime trajectory is determined based on instantaneous position of the obstacle only, without knowledge of its motion. The robustness function acts to push the UAV further to the left even though the obstacle has passed.

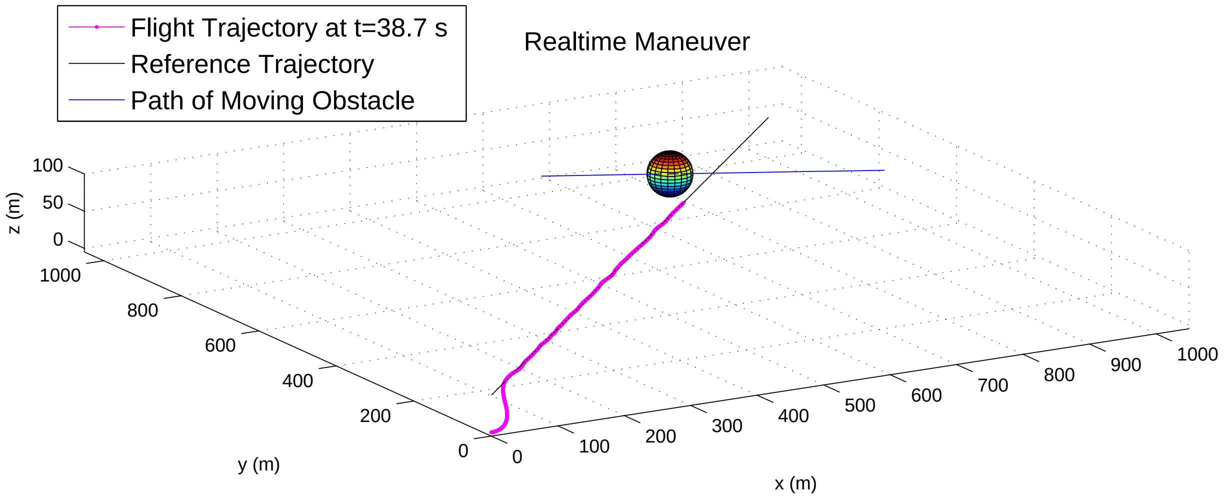


(a) 3D view

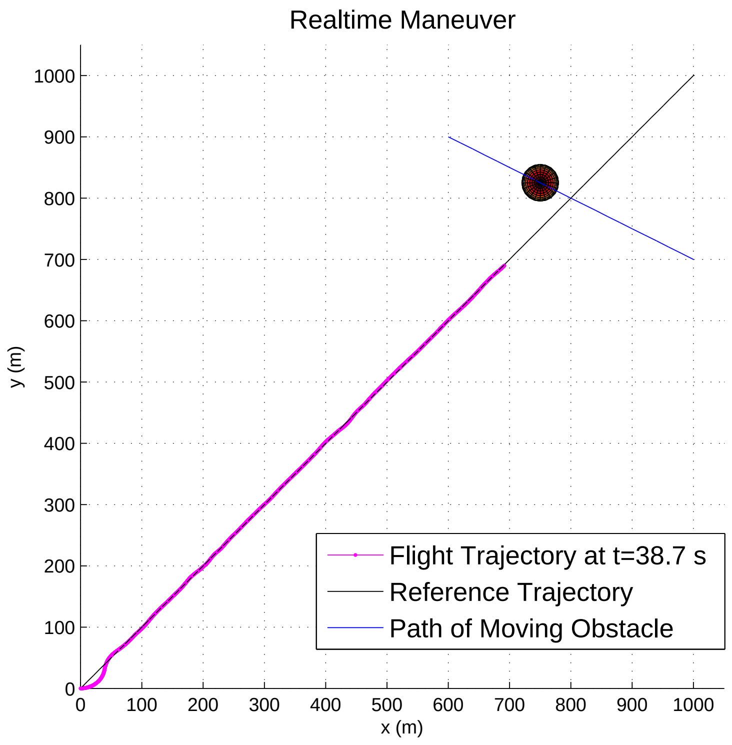


(b) View in XY Plane

Figure 4.89: Realtime Maneuver, flight path at $t = 32.7s$

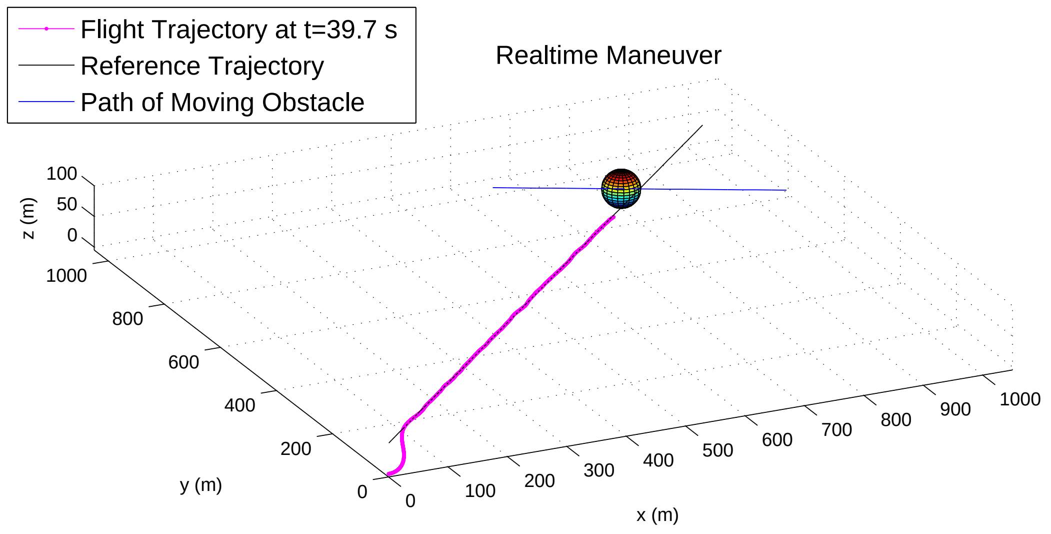


(a) 3D View

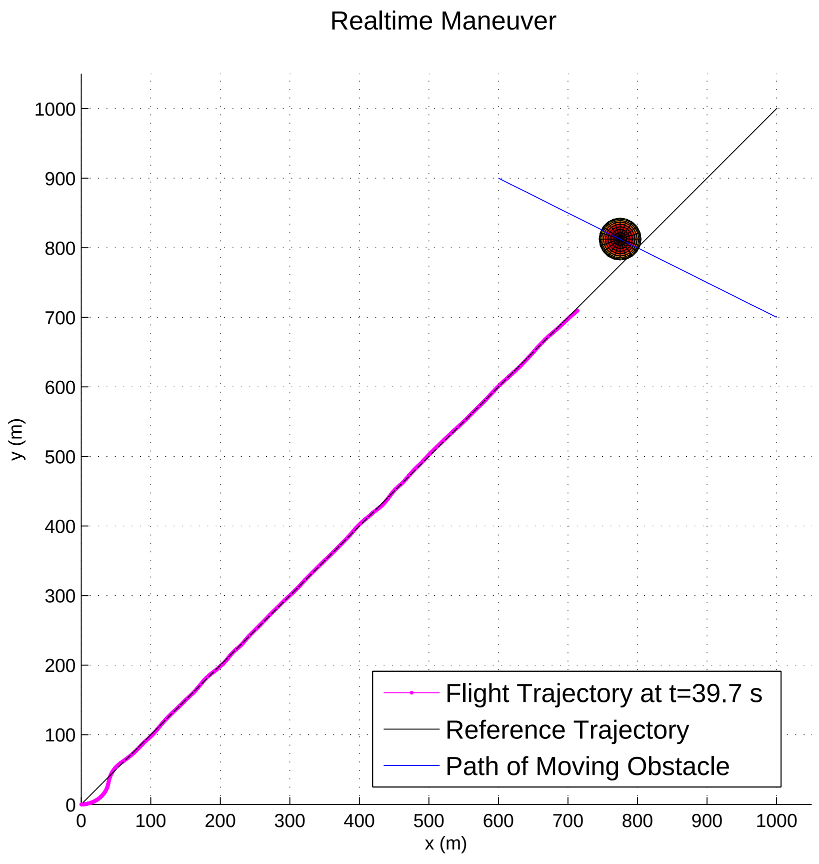


(b) View in XY Plane

Figure 4.90: Flight path at $t = 38.7\ s$

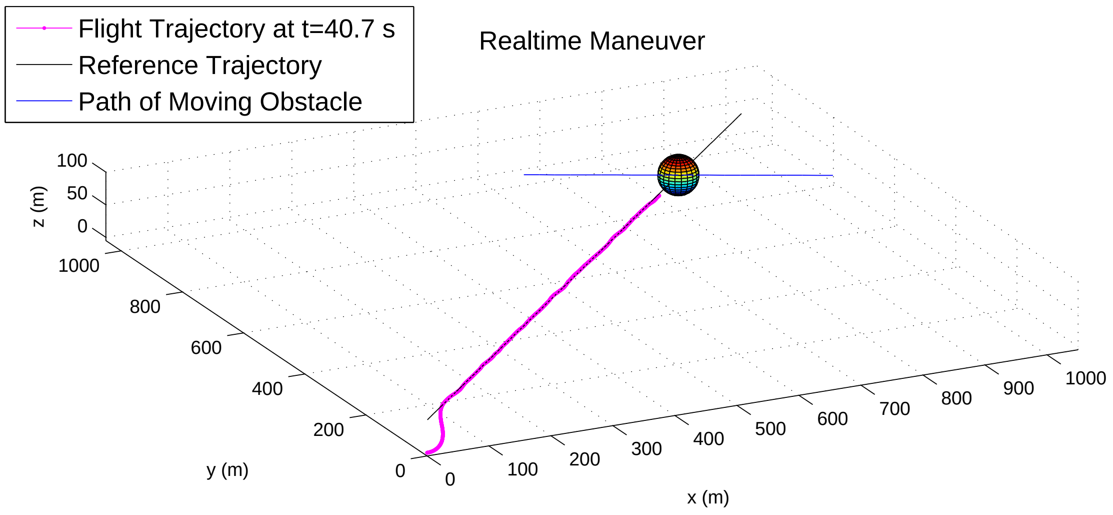


(a) 3D view

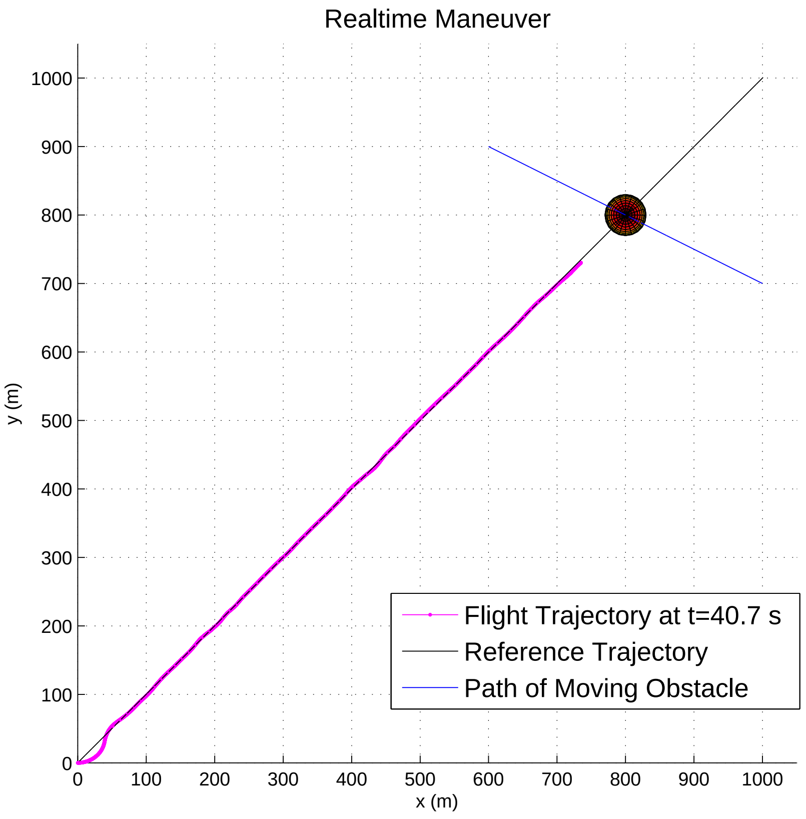


(b) View in XY Plane

Figure 4.91: Flight path at $t = 39.7s$

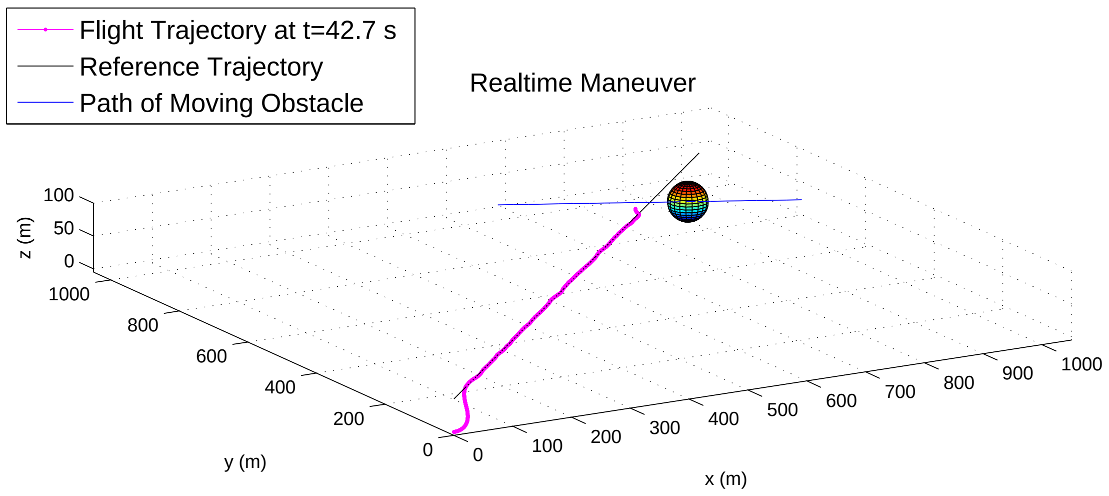


(a) 3D View

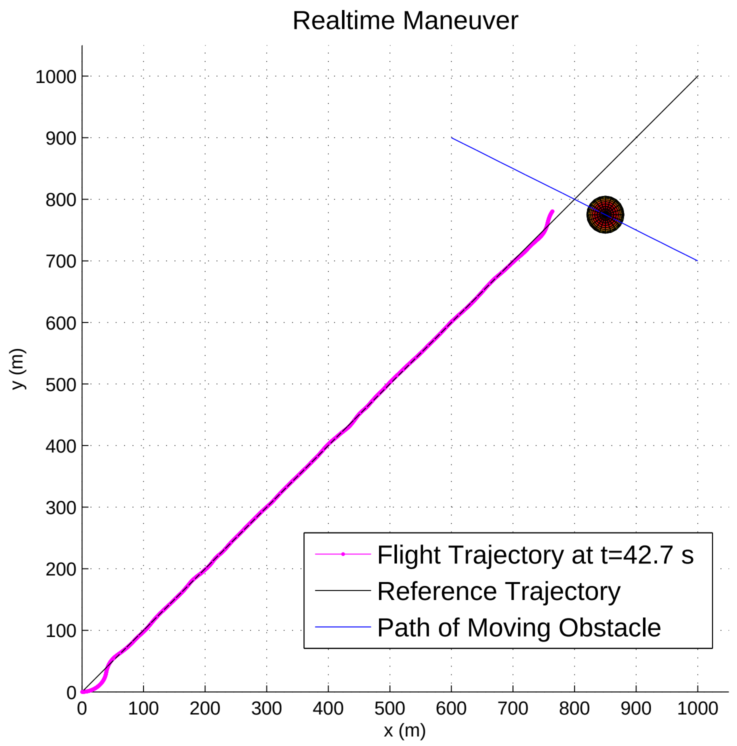


(b) View in XY Plane

Figure 4.92: Flight path at $t = 40.7\ s$

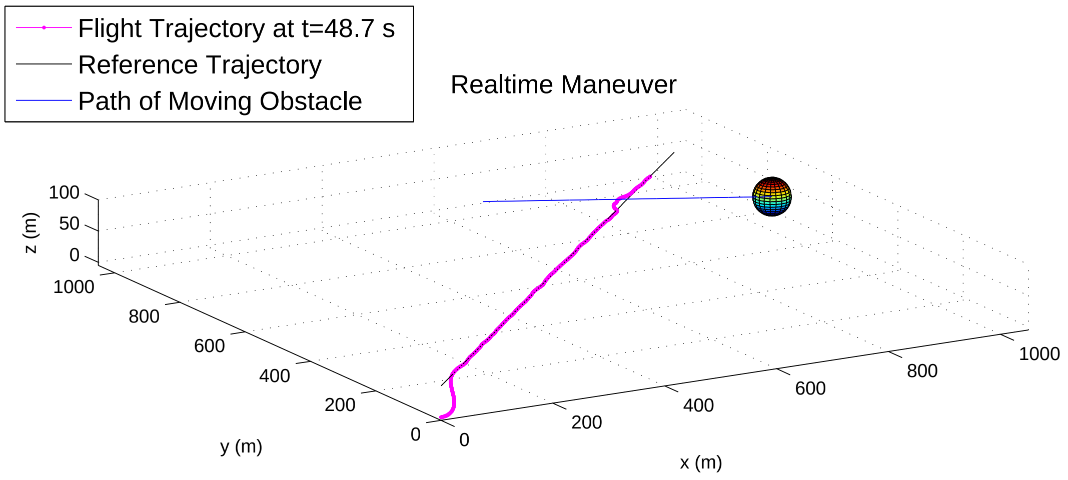


(a) 3D View

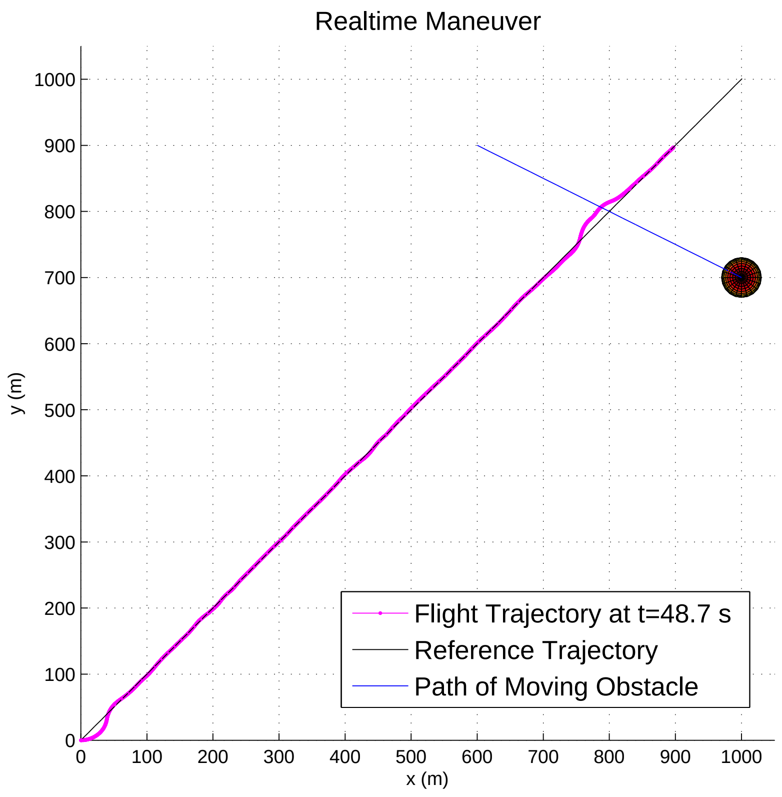


(b) View in XY Plane

Figure 4.93: Flight path at $t = 42.7\ s$

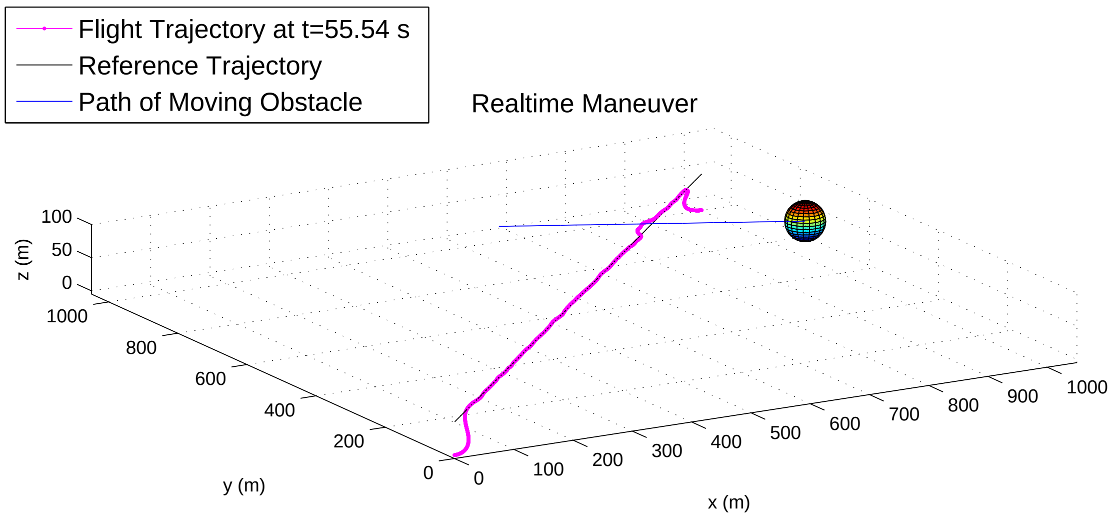


(a) 3D view

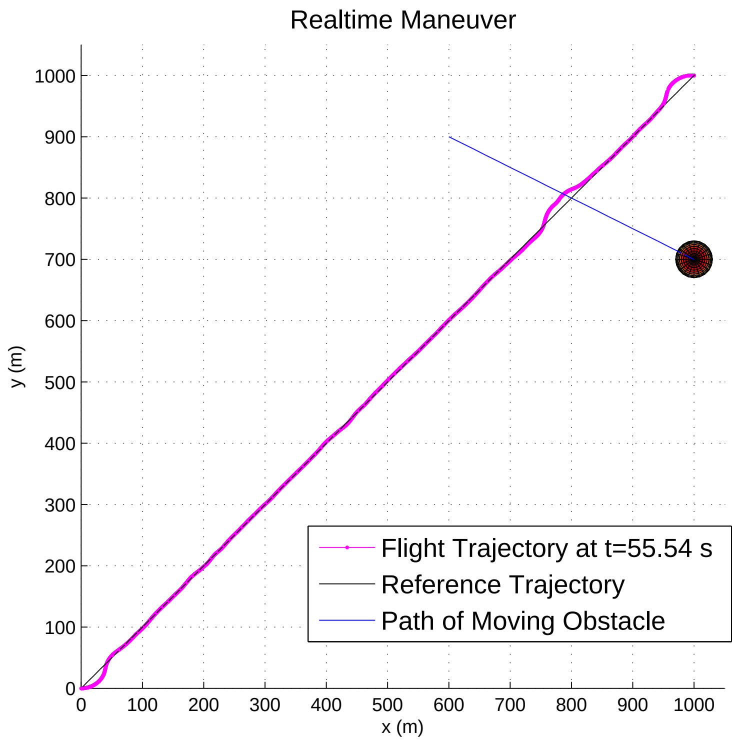


(b) View in XY Plane

Figure 4.94: Flight path at $t = 48.7s$



(a) 3D view



(b) View in XY Plane

Figure 4.95: Flight path at $t = 55.54s$

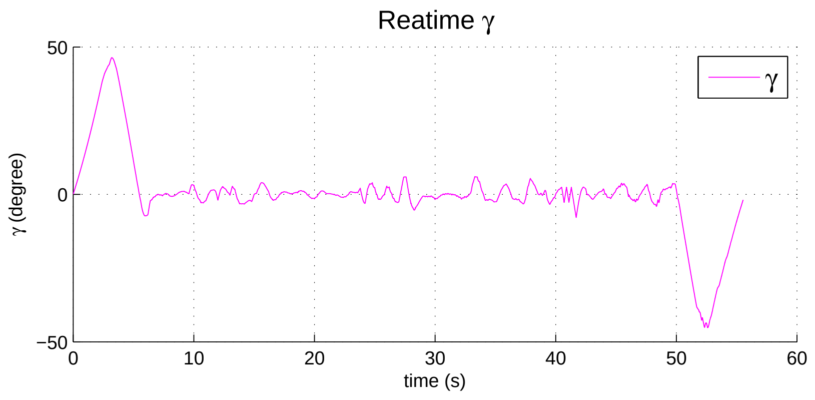


Figure 4.96: γvs t

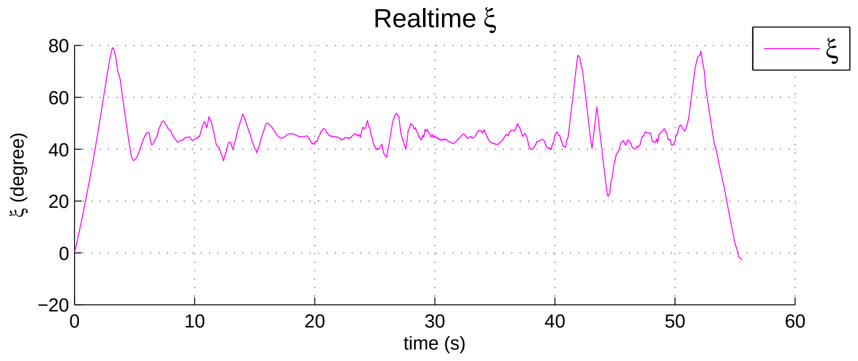


Figure 4.97: ξ vs t

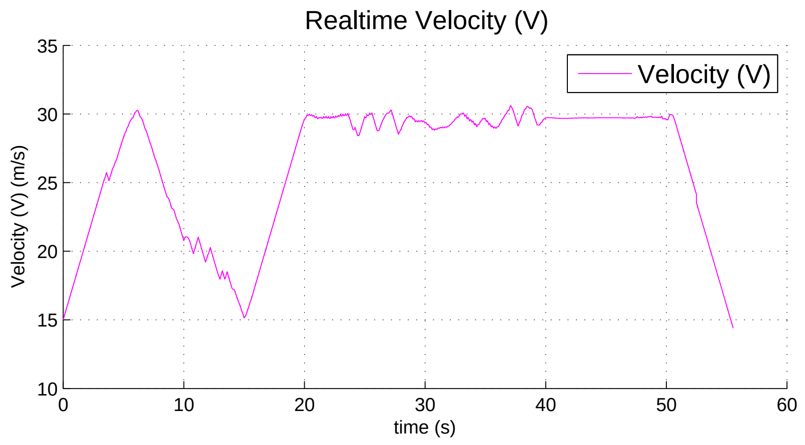


Figure 4.98: V vs t

IIIB. Realtime: An Obstacle Moving in Curve Path

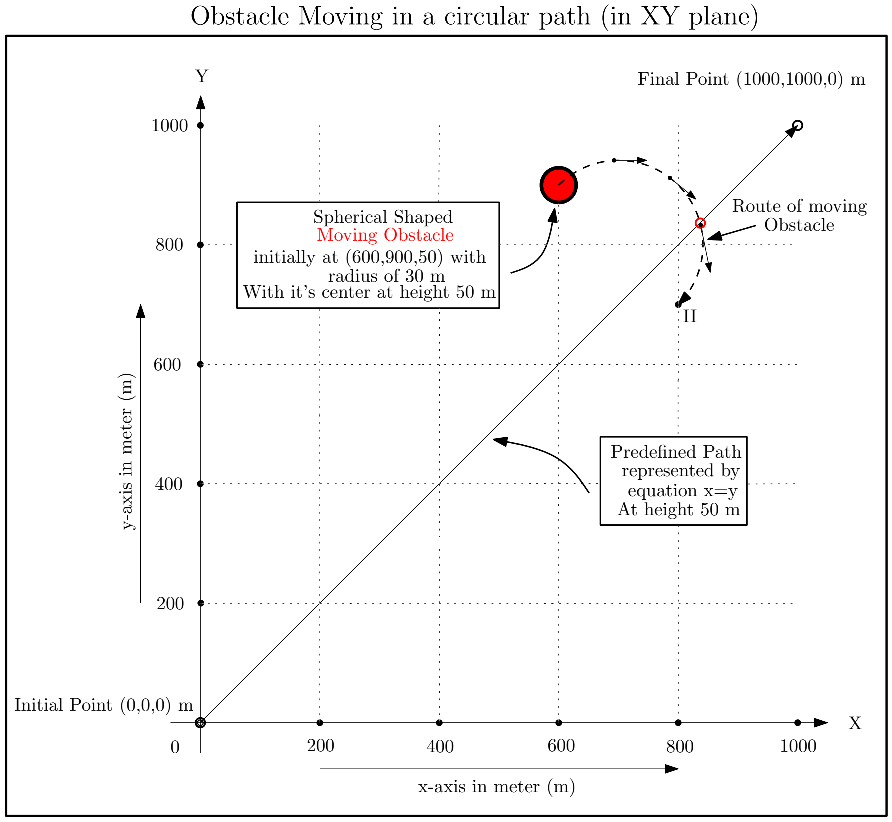


Figure 4.99: Scenario IIIB

Similar to **Scenario IIIA**, the obstacle is defined in such a way that it remains stationary with its center at $(600, 900, 50)$ m until 35 s and then starts moving in circular path with center of curvature at $(700, 800, 50)$ m reaching $(800, 700, 50)$ m 16 s later after passing through the reference trajectory at $(836, 836, 50)$ m. The radius of curvature is found to be $r = 100\sqrt{2}$ m. Initially the position of the obstacle in *meters* is given by

$$x_{ip} = 700 + r\cos(\theta)$$
$$y_{ip} = 800 + r\sin(\theta)$$
$$z_{ip} = 50$$

where,

$$\theta = 135$$

Let $\triangle\theta$ be the angle from the center of curvature between the initial position $(600, 900, 50)$ m and the final position $(800, 700, 50)$ m. Then, since it takes 16 s to travel from the start to finish positions, the obstacle constraint is given by.

$$h_1(x,y,z) := ln \left[\left| \left(\frac{x-(700+r\cos(135-(t-35)(\triangle\theta/16)))}{30} \right)^2 \right| \right.$$

$$\left. + \left| \left(\frac{y-(800+r\sin(135-(t-35)(\triangle\theta/16)))}{30} \right)^2 \right| + \left| \left(\frac{z-50}{30} \right)^2 \right| \right] \geq 0 \qquad (4.10)$$

$$35s \leq t \leq 51s$$

$$r = 100\sqrt{2}m$$

As in **Scenario IIIA**, the UAV is assumed to have knowledge only of the obstacle position at each sample instant. Hence, at a given sample instant, the new trajectory is computed with t in (4.10) held fixed at the current sample time.

With the path constraint in (4.10), the cost function for **Scenario IIIB** in equation (3.27) becomes

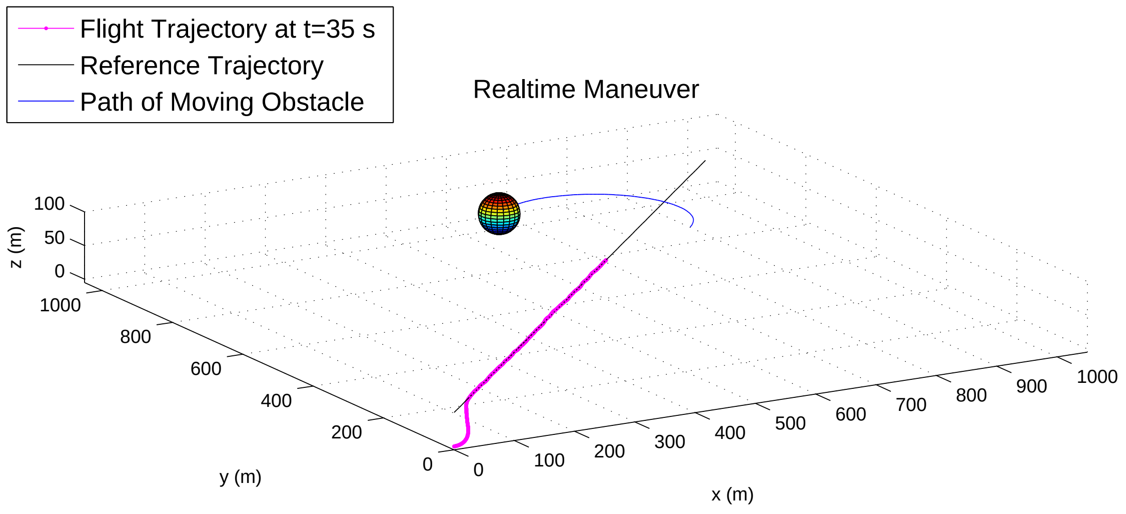
$$J(\underline{X}(\cdot), \underline{U}(\cdot)) = \int_{t_0}^{t_f} [r(x,y,z)]dt + A1 + A2 \qquad (4.11)$$

where, from equation (3.25), with $w = 0.5$, we get

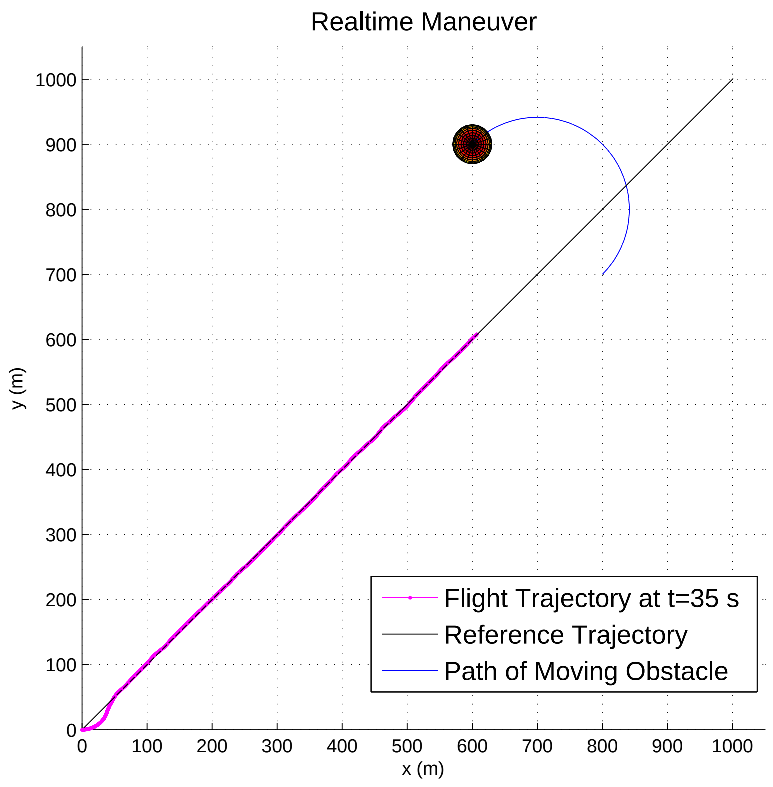
$$r(x,y,z) = 0.5 \left[(e^{e^{-h_1(x,y,z)}} - 1) \right] \qquad (4.12)$$

As for **Scenario IIIA**, the sampling frequency is 5 Hz, and a 40 nodes are used for each real-time solution. The resulting state trajectories are shown in Figures 4.100 to 4.108.

The average time for solving the realtime problem is 74.65 s, and the total maneuver time is 56.59 s.

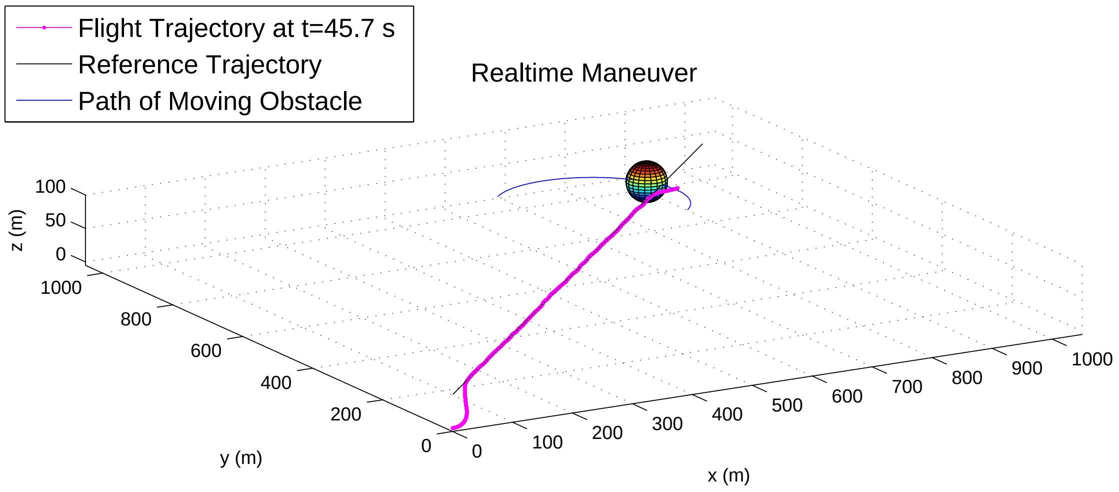


(a) 3D view

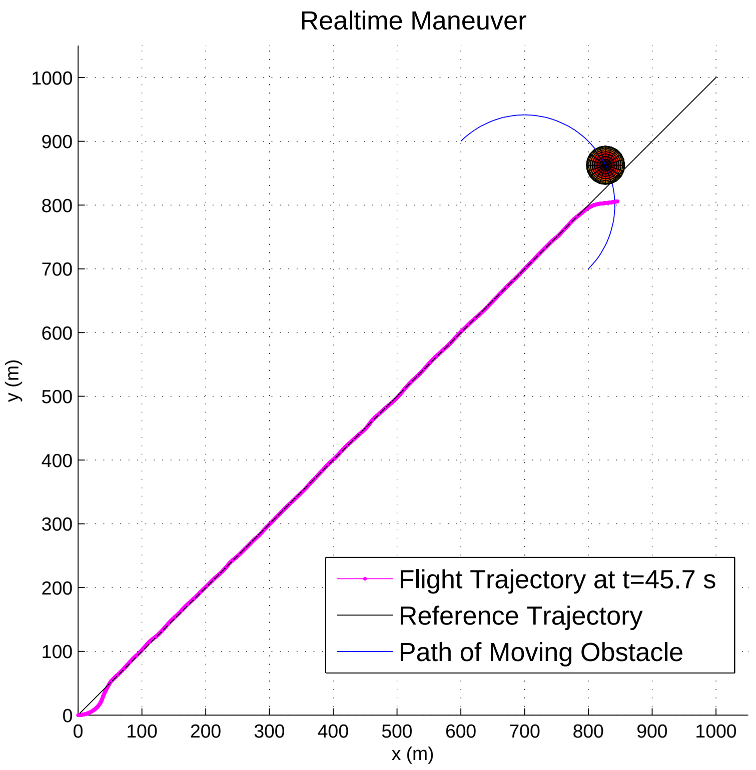


(b) View in XY Plane

Figure 4.100: Flight path at $t = 35s$

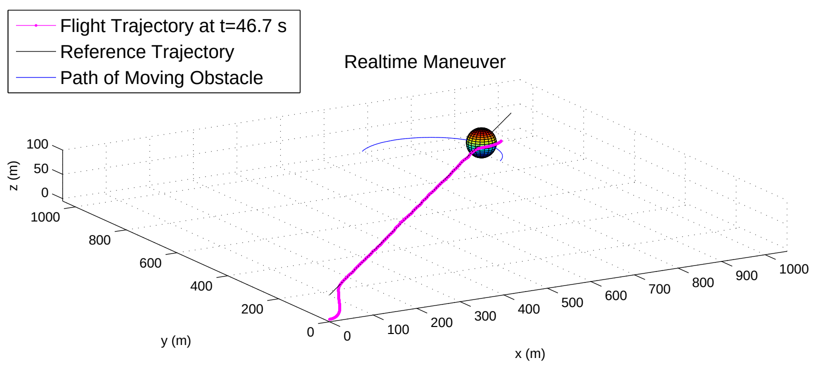


(a) 3D View

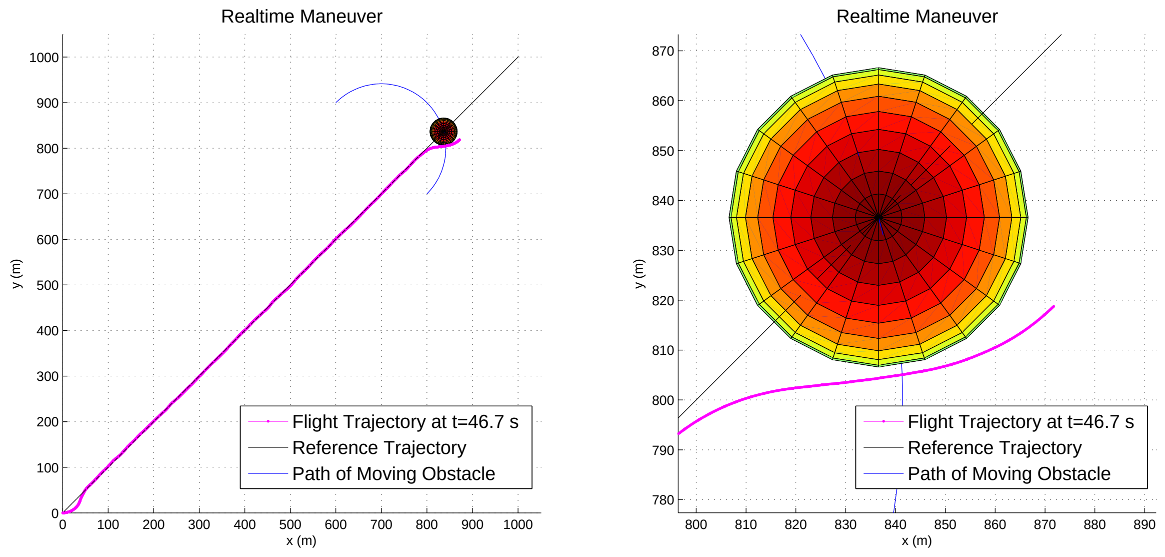


(b) View in XY Plane

Figure 4.101: Flight path at $t = 45.7\ s$

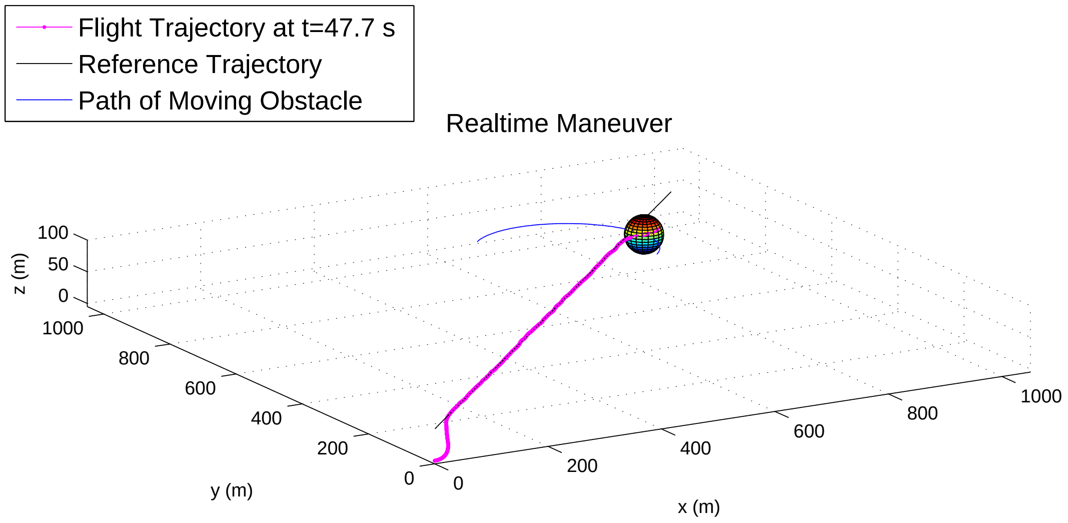


(a) 3D view

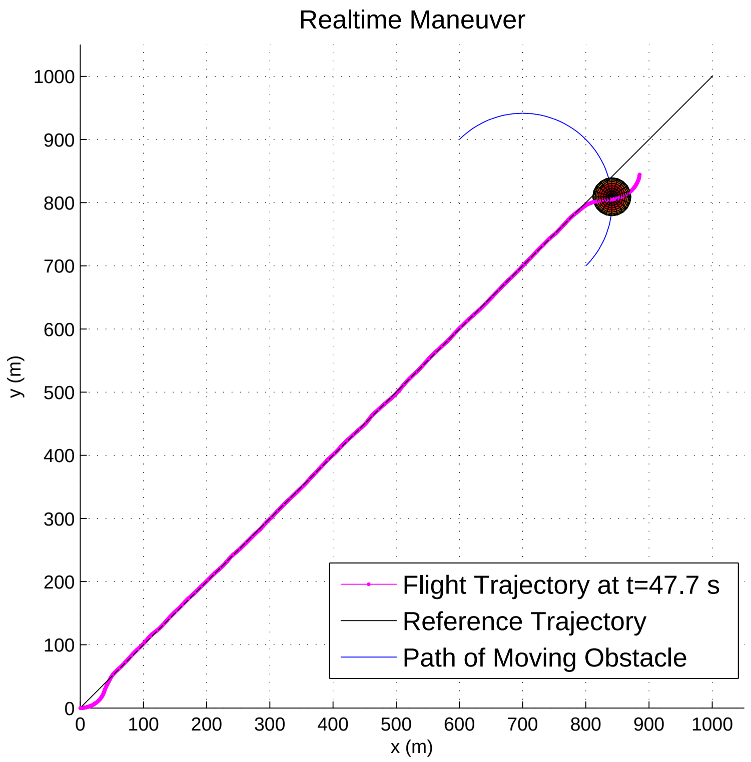


(b) View in XY Plane

Figure 4.102: Flight path at $t = 46.7s$

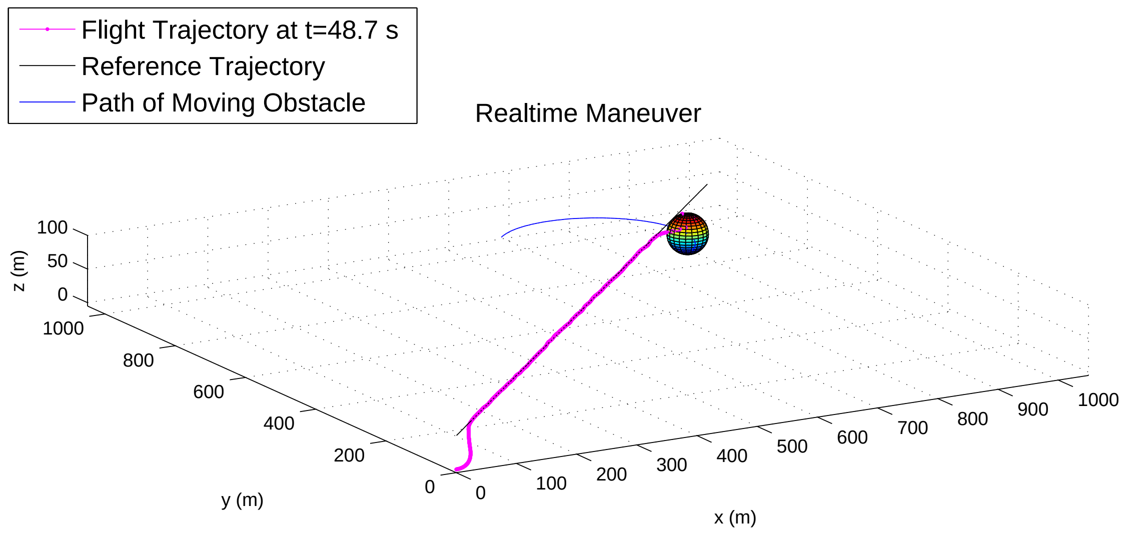


(a) 3D View

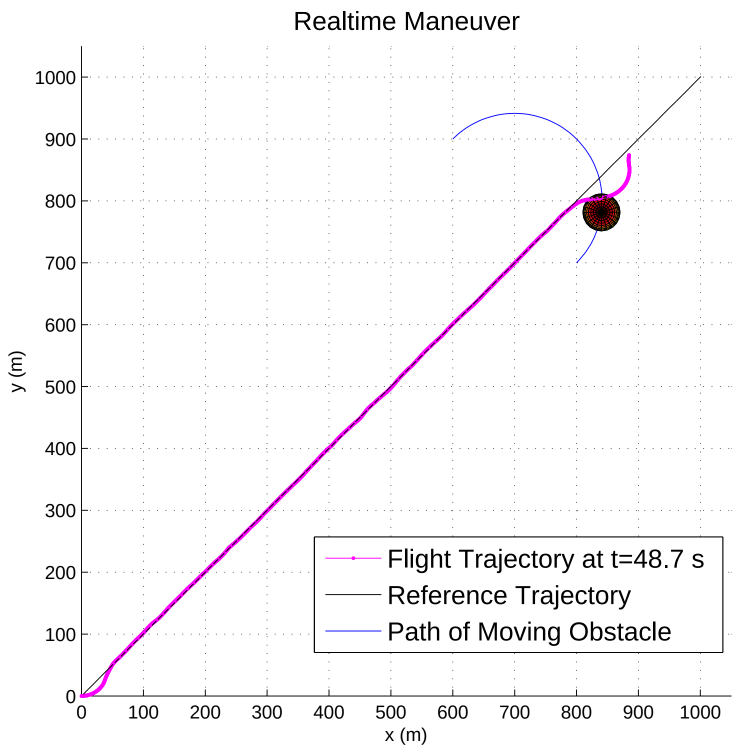


(b) View in XY Plane

Figure 4.103: Flight path at $t = 47.7\ s$

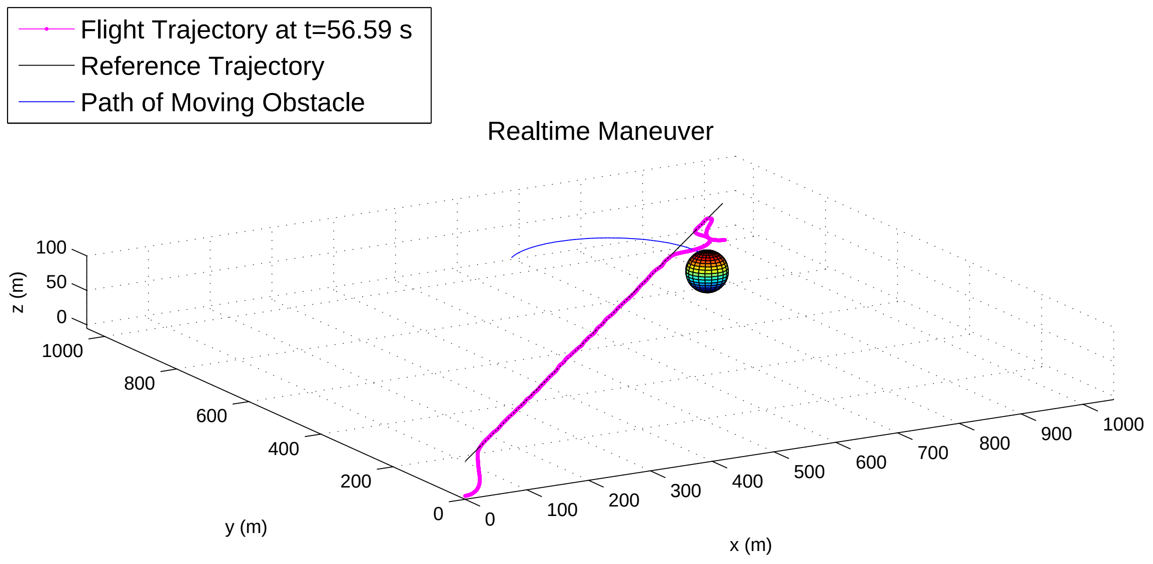


(a) 3D View

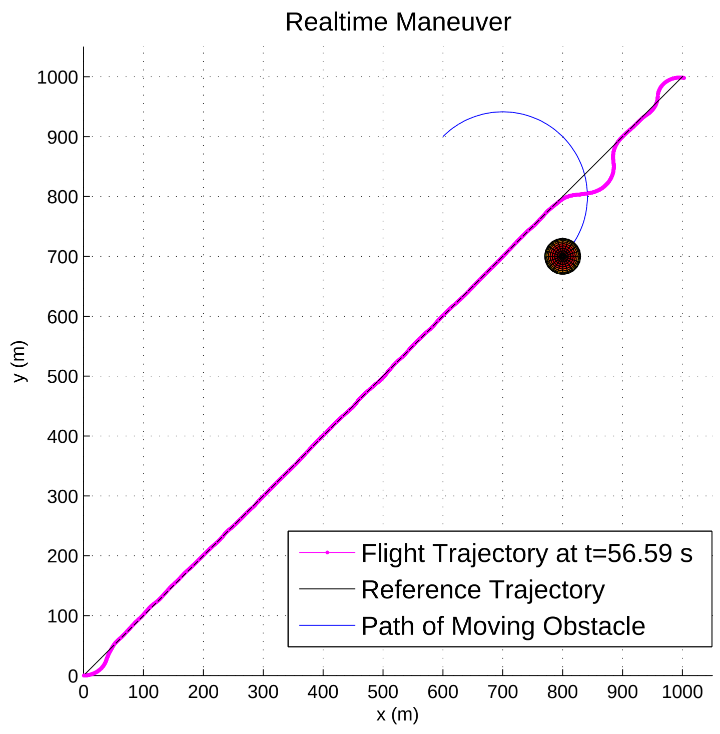


(b) View in XY Plane

Figure 4.104: Flight path at $t = 48.7\ s$



(a) 3D view



(b) View in XY Plane

Figure 4.105: Flight path at $t = 56.59s$

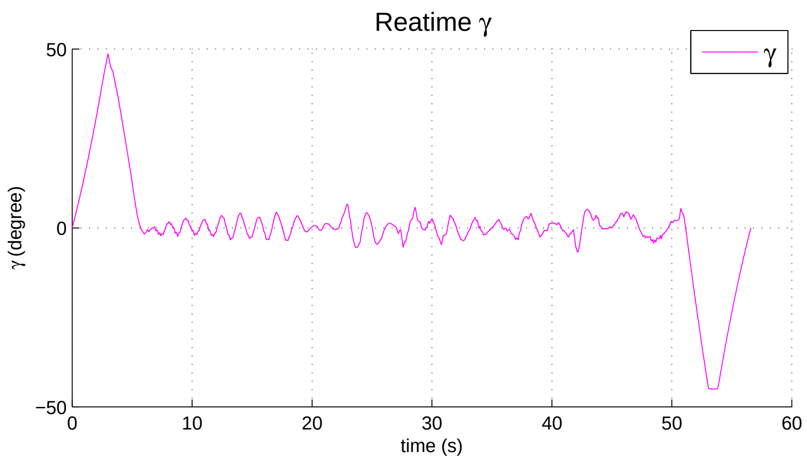


Figure 4.106: γvs t

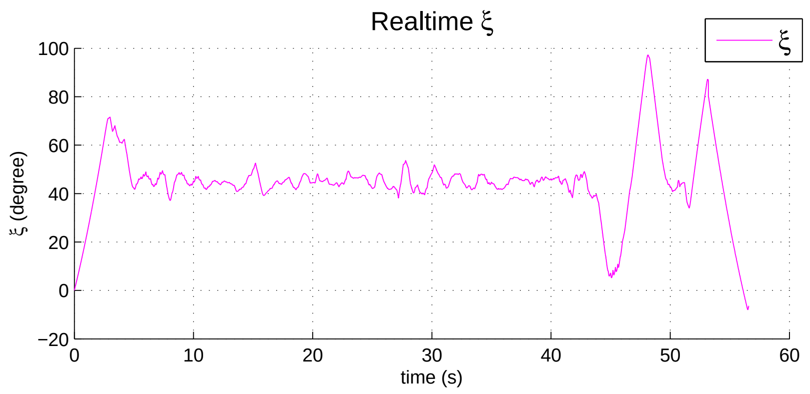


Figure 4.107: ξ vs t

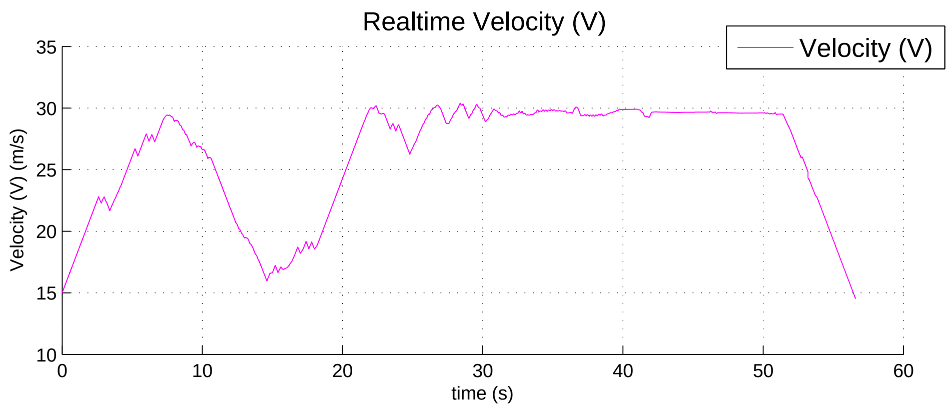


Figure 4.108: V vs t

From **Scenarios IIIA** and **IIIB**, it can be seen that the method is capable of generating trajectories in real-time to avoid moving obstacles, with only information about the obstacle's instantaneous position. However, it can require increasing the robustness factor, which increases the computation time, as well as the total time for the maneuver.

4.4 Scenario IV. Pop-up Obstacle

Apart from stationary and moving obstacles, it is a practical requirement that this method can handle obstacles that are detected suddenly. This particular situation can occur for example in crowded environments when one building is right behind the next building and is not detected until the first building is passed. This situation can also be seen when the environment is foggy and the visibility of the aircraft is very low. Thus, autonomous avoidance of pop up obstacles is of significant practical importance. In **Scenario IV** we consider such a scenario. In this section the method is implemented with two sub cases. First, when the obstacle is detected 50 m ahead of UAV and second, when obstacle appears on the reference trajectory 4 s prior to expected collision. A cuboid shaped structure with its base centered at $(600, 650, 0)$ m and height of 60 m is considered as the pop-up obstacle. The obstacle has equal width and length of 100 m. Equation (4.13) shows the corresponding obstacle constraint equation.

$$h_1(x,y,z) := ln\left[\left|\left(\frac{x-600}{50}\right)^8\right| + \left|\left(\frac{y-650}{50}\right)^8\right| + \left|\left(\frac{z-30}{30}\right)^8\right|\right] \geq 0 \tag{4.13}$$

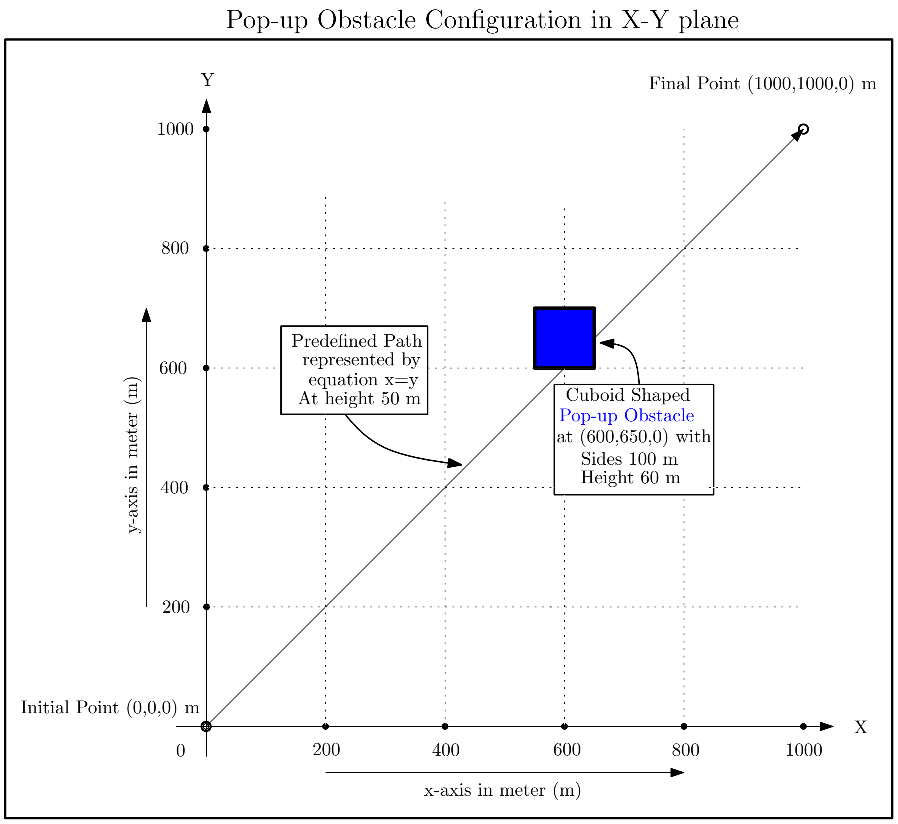

Pop-up Obstacle Configuration in X-Y plane
Y
Final Point (1000,1000,0) m
1000
800
600
400
200
0
y-axis in meter (m)
Predefined Path represented by equation x=y At height 50 m
Cuboid Shaped Pop-up Obstacle at (600,650,0) with Sides 100 m Height 60 m
Initial Point (0,0,0) m
200
400
600
800
1000
X
x-axis in meter (m)

Figure 4.109: Scenario IV

IVA. Realtime: An obstacle appears $50\ m$ ahead

In this case the obstacle is detected when it is $50m$ ahead of the aircraft. Since the reference trajectory first passes through the obstacle's surface at $(600, 600, 50)\ m$, the obstacle is detected when the UAV is $50m$ ahead of this point. Hence the distance from the UAV to the obstacle along the reference trajectory is from UAV to obstacle is given by

$$d(x, y, z) = \sqrt{(x - 600)^2 + (y - 600)^2 + (z - 50)^2}$$

and the obstacle constraint in (4.13) comes into effect when $d \leq 50m$.

With the path constraint in (4.13), the cost function for **Scenario IVA** in equation (3.27) becomes

$$J(\underline{X}(\cdot), \underline{U}(\cdot)) = \int_{t_0}^{t_f} [r(x, y, z)]dt + A1 + A2 \tag{4.14}$$

where, from equation (3.25), with $w = 0.25$, we get

$$r(x,y,z) = 0.25\left[(e^{e^{-h_1(x,y,z)}} - 1)\right] \tag{4.15}$$

Unlike **Scenario III**, since the obstacle is not moving, we can get away with a smaller robustness factor (as will be seen), thereby decreasing computational time.

As for **Scenario II** and **III**, the sampling frequency is 5 Hz, and 40 nodes are used for each real-time solution. The resulting state trajectories are shown in Figures 4.110 to 4.116.

The average time for solving the realtime problem is 44.1311 s and a total maneuver time of 73.503 s. In this case even with decreased robustness factor it can be seen that the UAV successfully avoids the obstacle and returns to the reference trajectory.

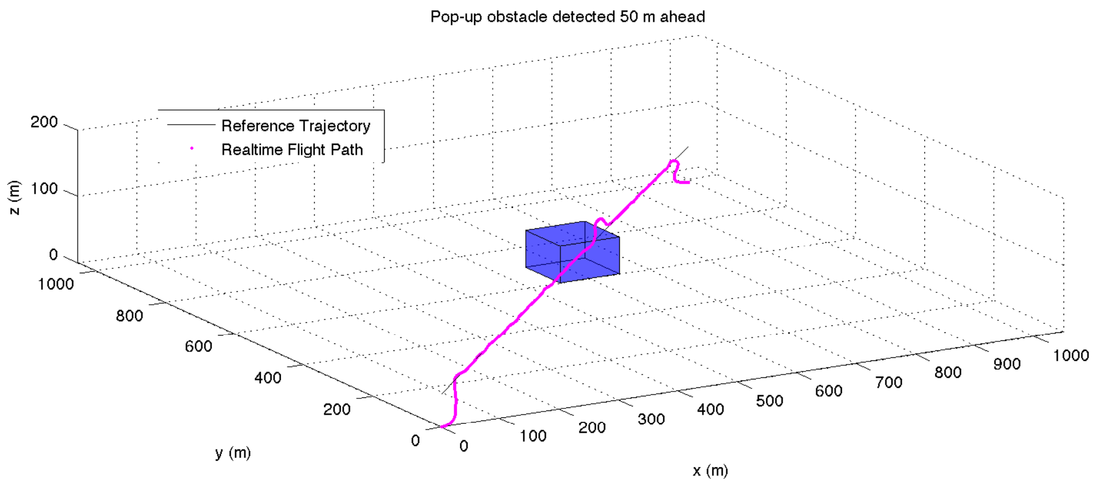


Figure 4.110: Pop-up Obstacle detected 50 m ahead: 3D view

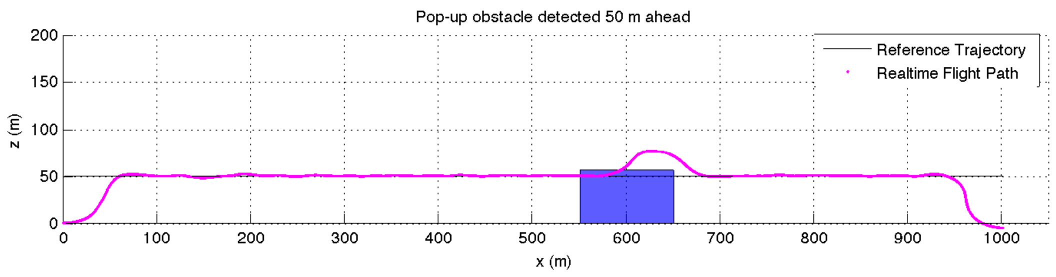


Figure 4.111: Pop-up Obstacle detected 50 m ahead: view in XZ plane

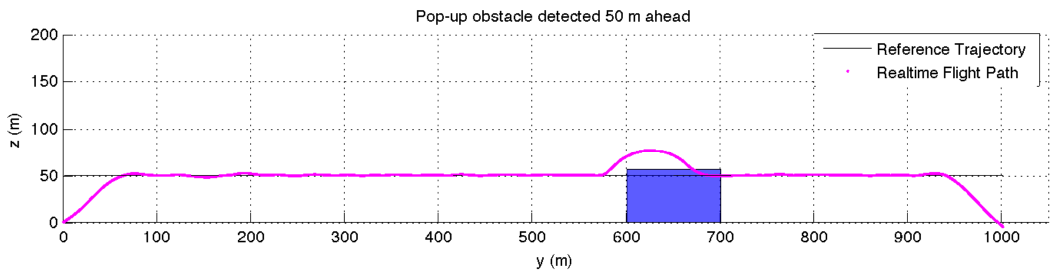


Figure 4.112: Pop-up Obstacle detected 50 m ahead: view in YZ plane

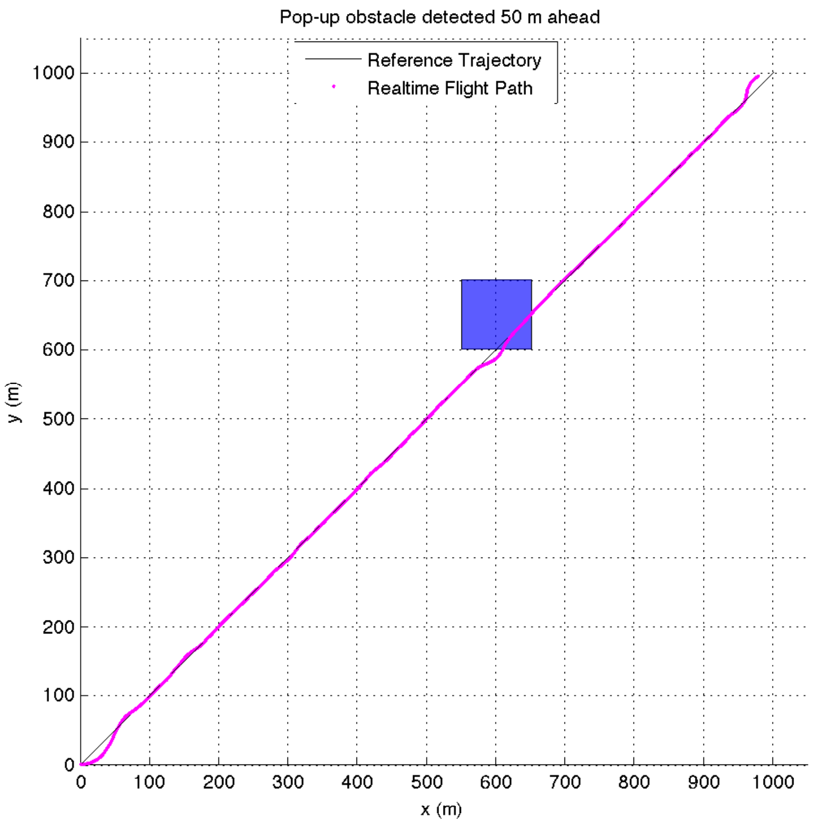


Figure 4.113: Pop-up Obstacle detected 50 m ahead: view in XY plane

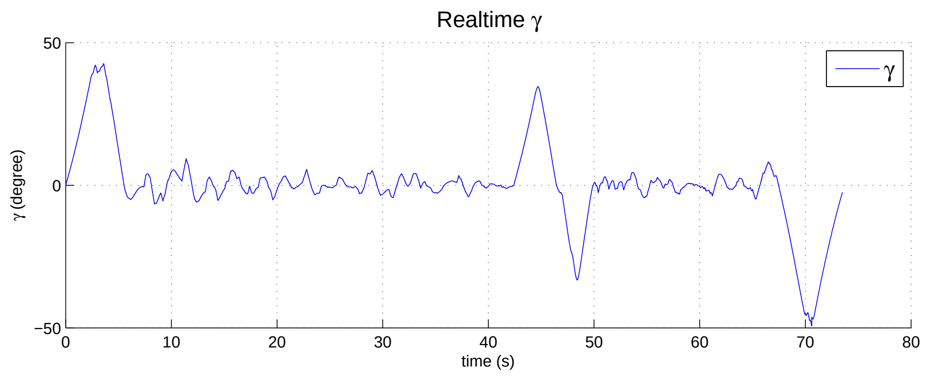


Figure 4.114: γ vs t : with pop-up obstacle detected 50 m ahead

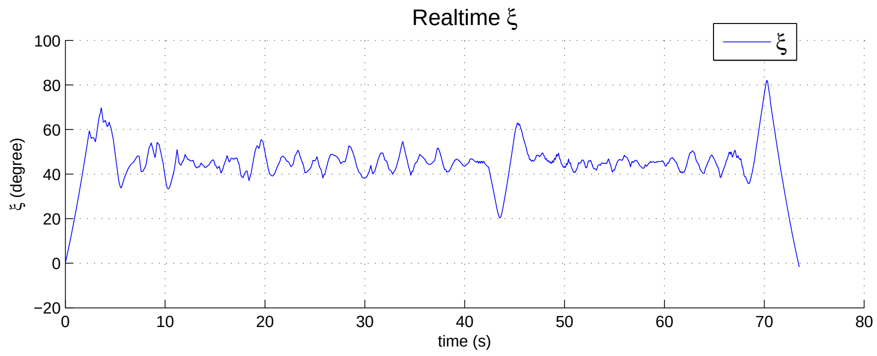


Figure 4.115: ξ vs t : with pop-up obstacle detected 50 m ahead

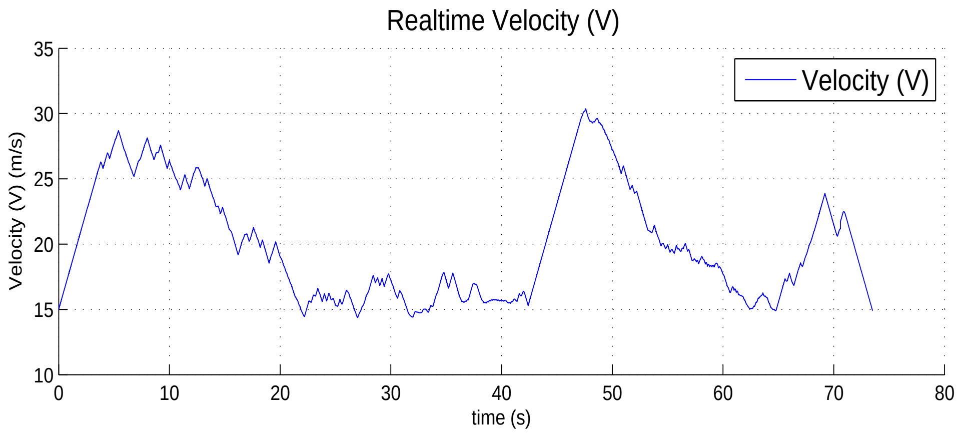


Figure 4.116: V vs t : with pop-up obstacle detected 50 m ahead

The discrepancies seen in the Figures 4.111 to 4.113 at the end of the trajectory is due to the propagation error as explained in subsection 4.1.

IVB. Realtime: An obstacle appears $4s$ prior to expected collision

In this case the obstacle is detected $4s$ prior to expected collision. From **Scenario ID**, the UAV reaches the point $(600, 600, 50)$ m at $30s$. Hence, with only the pop-up obstacle on its path, we assume the obstacle is detected at $26s$ of simulation time. Hence the obstacle constraint in equation (4.13) comes into effect after $t = 26$ s.

The cost function for **Scenario IVB** is consequently the same as for **Scenario IVA**, with the only difference being the time at which the obstacle constraint comes into effect.

As for **Scenario IVA**, the sampling frequency is 5 Hz, and 40 nodes are used for each real-time solution. The resulting state trajectories are shown in Figures 4.117 to 4.123. The average time for solving the realtime problem is $47.32s$ and a total maneuver time of $70.6s$.

It can be seen that the UAV successfully avoids the obstacle and returns to the reference trajectory.

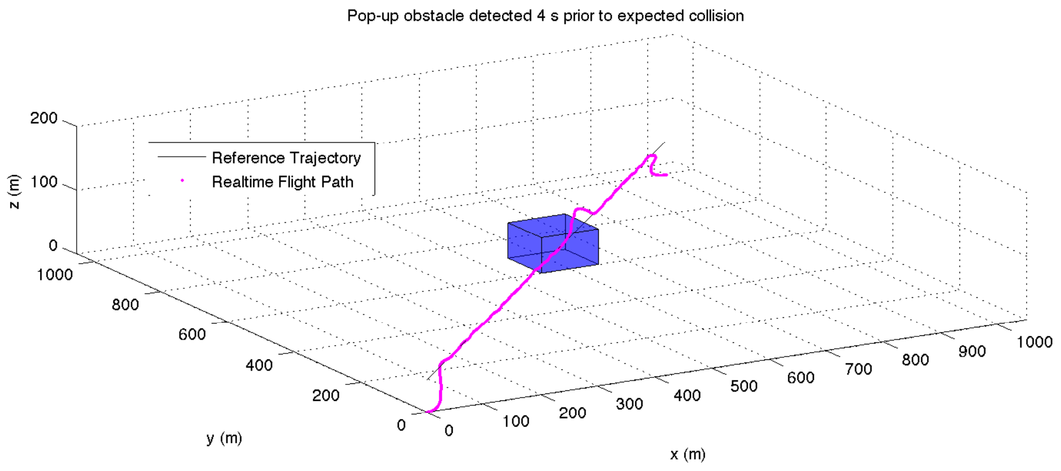


Figure 4.117: Pop-up Obstacle detected 4 s prior to expected collision: 3D view

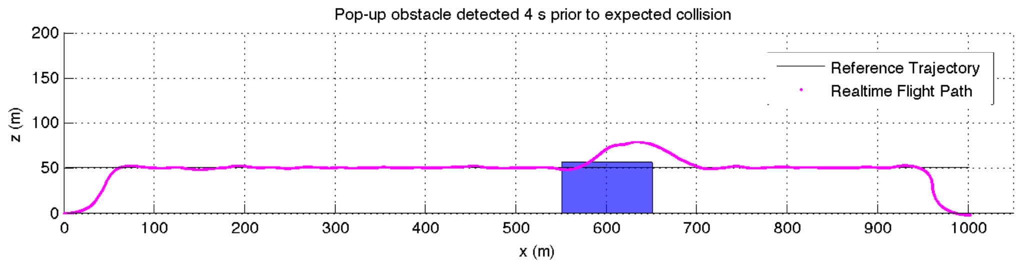


Figure 4.118: Pop-up Obstacle detected 4 s prior to expected collision: view in XZ plane

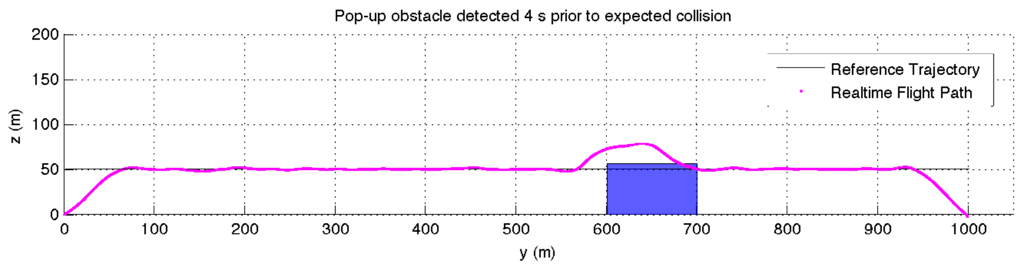


Figure 4.119: Pop-up Obstacle detected 4 s prior to expected collision: view in YZ plane

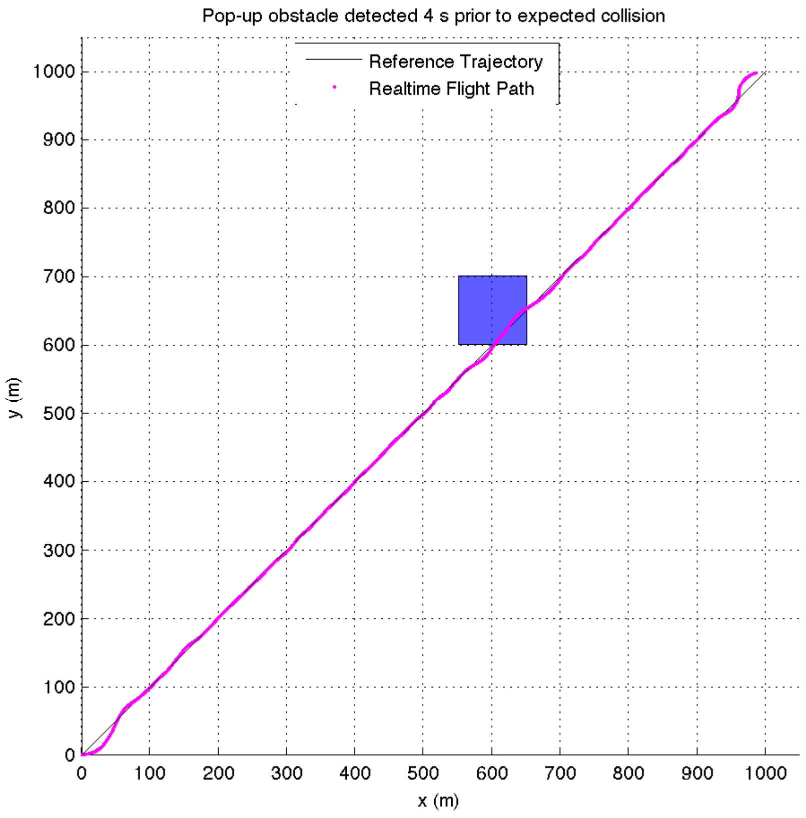


Figure 4.120: Pop-up Obstacle detected 4 s prior to expected collision: view in XY plane

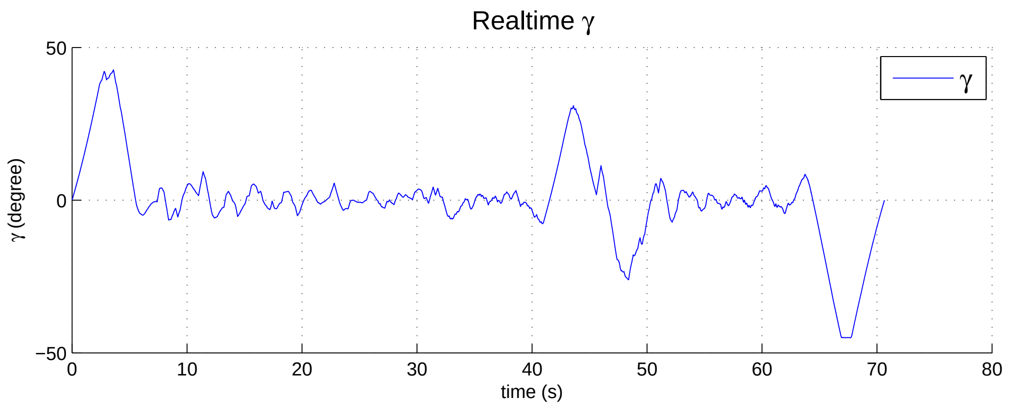


Figure 4.121: γ vs t : with pop-up obstacle detected 4 s prior to expected collision:

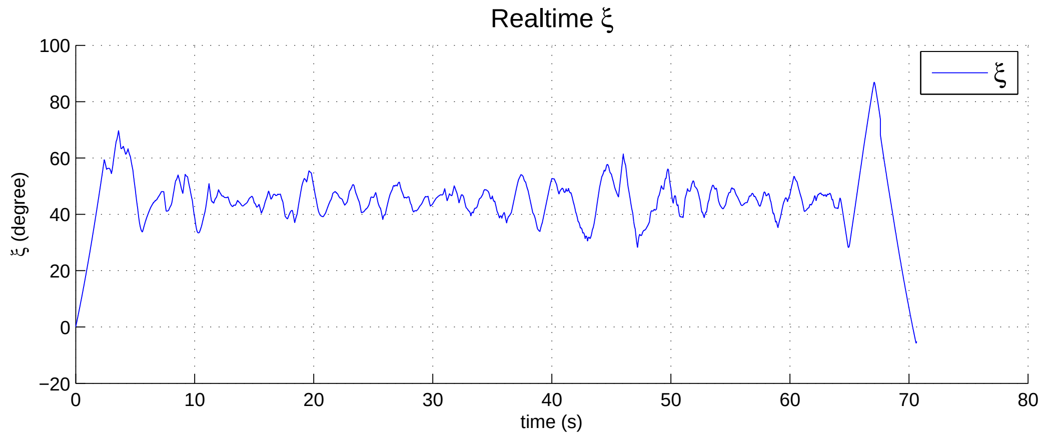


Figure 4.122: ξ vs t : with pop-up obstacle detected 4 s prior to expected collision:

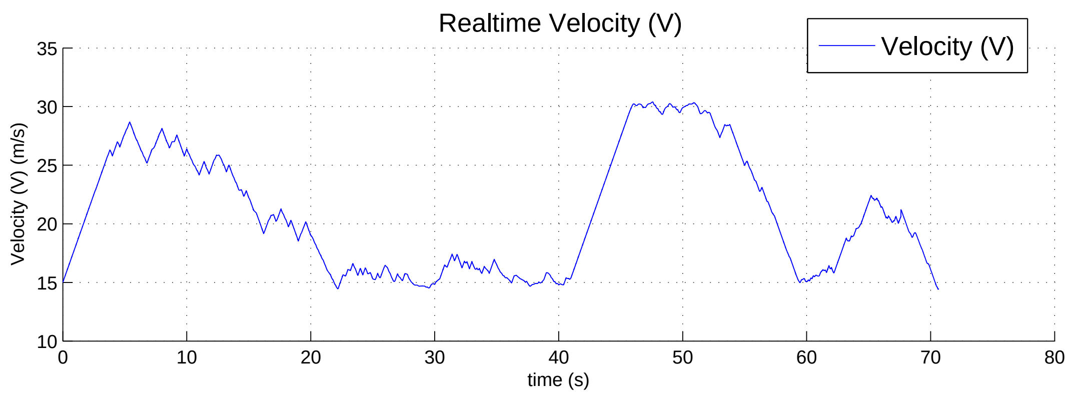


Figure 4.123: V vs t : with pop-up obstacle detected 4 s prior to expected collision:

To summarize both cases, the real-time trajectories generated for **Scenarios IVA** and **IVB** are shown in Figures 4.124 to 4.127.

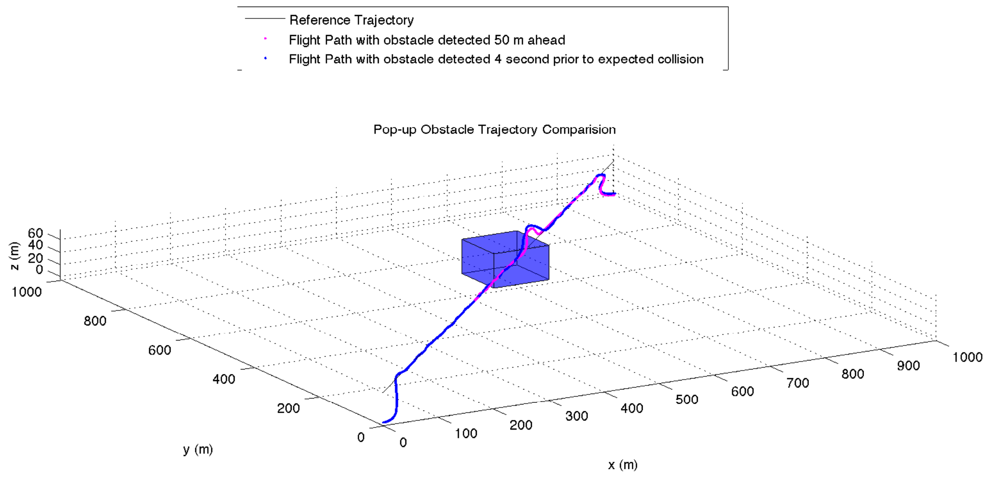


Figure 4.124: Comparison of case IVA and IVB: 3D view

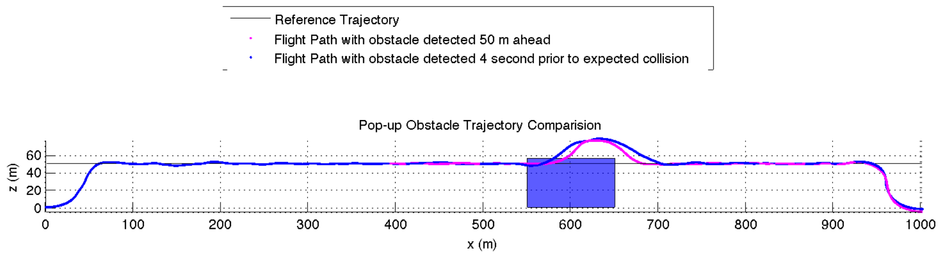


Figure 4.125: Comparison of case IVA and IVB: XZ view

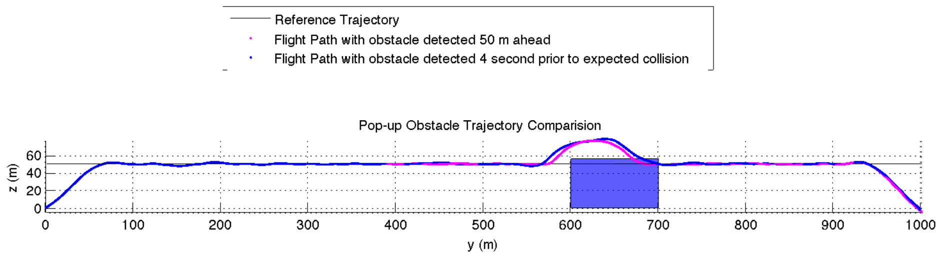


Figure 4.126: Comparison of case IVA and IVB: YZ view

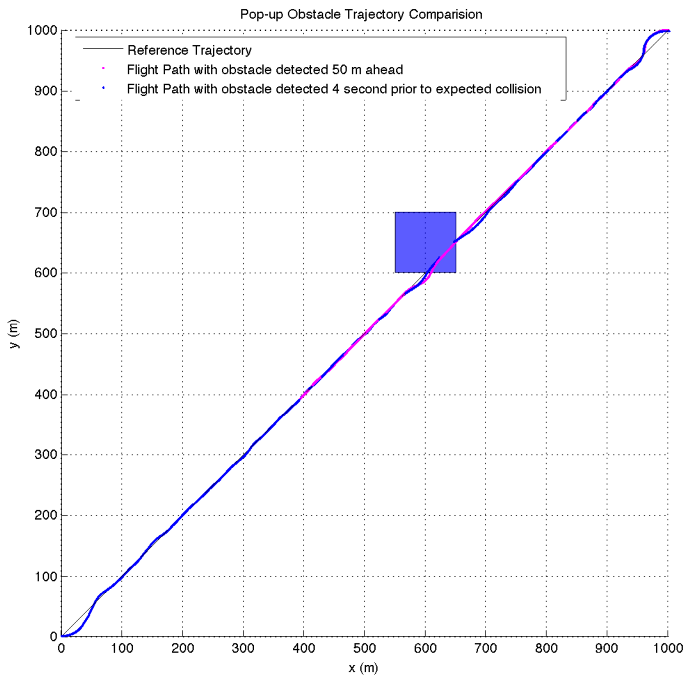


Figure 4.127: Comparison of case IVA and IVB: XY view

The trajectories in both cases are similar and avoid the pop-up obstacles successfully.

4.5 Scenario V. Stationary, Pop-up and Moving Obstacles

Finally, **Scenario V** includes stationary, pop-up and moving obstacles together, as shown in Figure 4.128 and 4.137. In this case, the stationary obstacle with its base centered at $(350, 300, 0)$ m is cylindrical shaped with a height of 80 m. The path constraint due to the stationary obstacle is given in equation (4.16). The pop-up obstacle is formed by a cuboid shaped structure with its base centered at $(600, 650, 0)$ m and dimension of $100 \times 100 \times 60$ m. The pop-up obstacle is detected just 50 m ahead by UAV on reference trajectory similar to **Scenario IVA.** Similarly, the path constraint due to pop-up obstacle is given in equation (4.17). The moving obstacle is spherical shaped with radius of 30 m. According to the path of the moving obstacle, **Scenario V** is further divided into two sub-cases. First, when the moving obstacle's path is a straight line and second when the path of moving obstacle is a curve.

$$h_1(x,y,z) \coloneqq ln\left[\left|\left(\frac{x-350}{50}\right)^2\right| + \left|\left(\frac{y-300}{50}\right)^2\right| + \left|\left(\frac{z-40}{40}\right)^8\right|\right] \geq 0; t \geq 0 \qquad (4.16)$$

$$h_2(x,y,z) \coloneqq ln\left[\left|\left(\frac{x-600}{50}\right)^8\right| + \left|\left(\frac{y-650}{50}\right)^8\right| + \left|\left(\frac{z-30}{30}\right)^8\right|\right] \geq 0; d \leq 50m \qquad (4.17)$$

where,

$$d(x,y,z) = \sqrt{(x-600)^2 + (y-600)^2 + (z-50)^2}$$

VA. Realtime: Stationary, Pop-up and Moving (in straight line) obstacles.

Moving Obstacle in Straight Line (in X-Y plane)

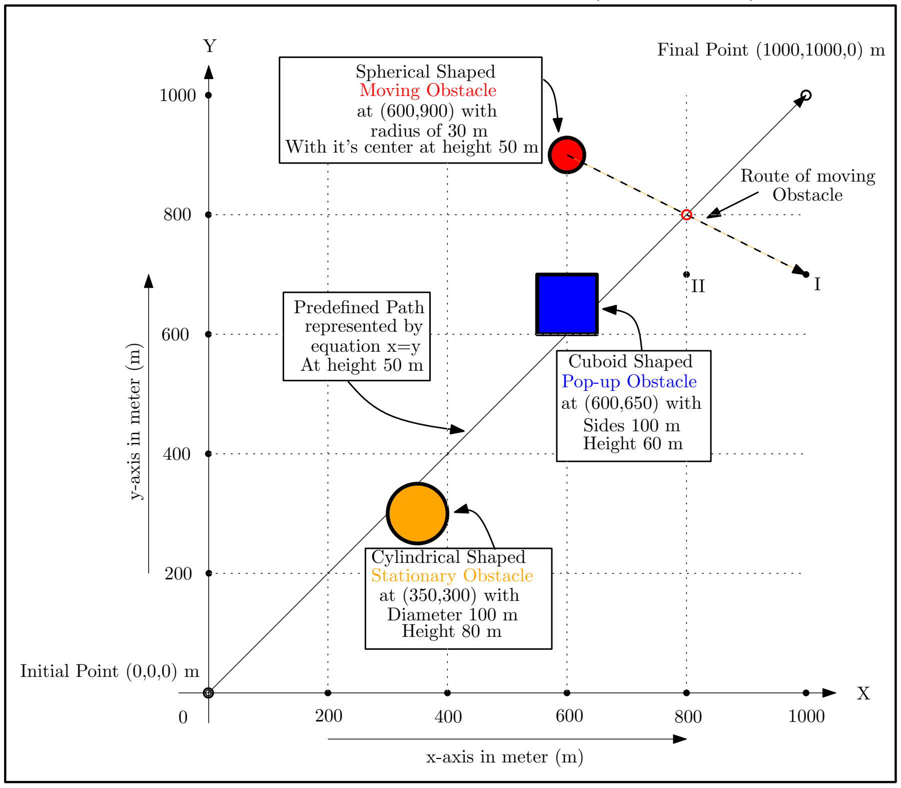


Figure 4.128: Scenario VA

In this case, the moving obstacle remains stationary till 38 s with its center at $(600, 900, 50)$ m and starts moving after 38 s to reach the point $(1000, 700, 50)$ m at 54 s passing through the reference trajectory at 46 s. The constraint due to moving obstacle is

$$h_3(x,y,z) := ln \left[\left| \left(\frac{x - 100(6 + \frac{(t-40)}{4})}{30} \right)^2 \right| + \right.$$

$$\left. \left| \left(\frac{y - 100(9 - \frac{(t-40)}{8})}{30} \right)^2 \right| + \left| \left(\frac{z-50}{30} \right)^2 \right| \right] \geq 0; 38 \leq t \leq 54 \tag{4.18}$$

However, as before, it must be emphasized that the UAV is assumed to have knowledge only of the obstacle position at each sample instant. Hence, at a given sample instant, the new trajectory is computed with t in (4.18) held fixed at the current sample time.

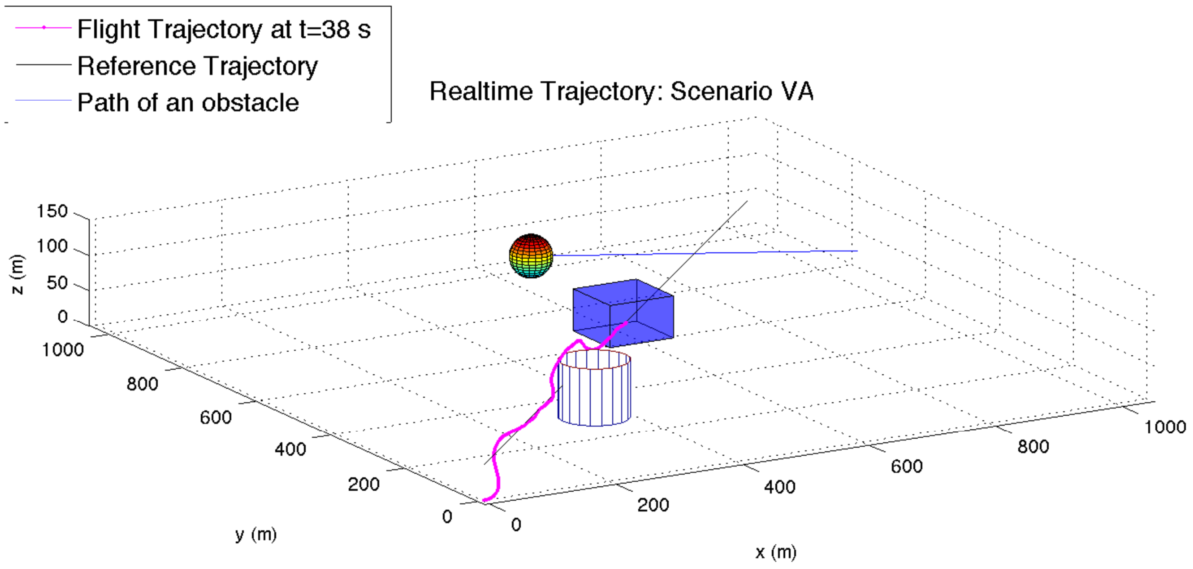
Similarly, equation (3.27) defines the problem formulation for **Scenario VA**, with the path constraint due to obstacles given by equations (4.16), (4.17) and (4.18), and the cost function given by equation (4.19) implemented in realtime.

$$J(\underline{X}(\cdot), \underline{U}(\cdot)) = \int_{t_0}^{t_f} [r(x,y,z)]dt + A1 + A2 \tag{4.19}$$

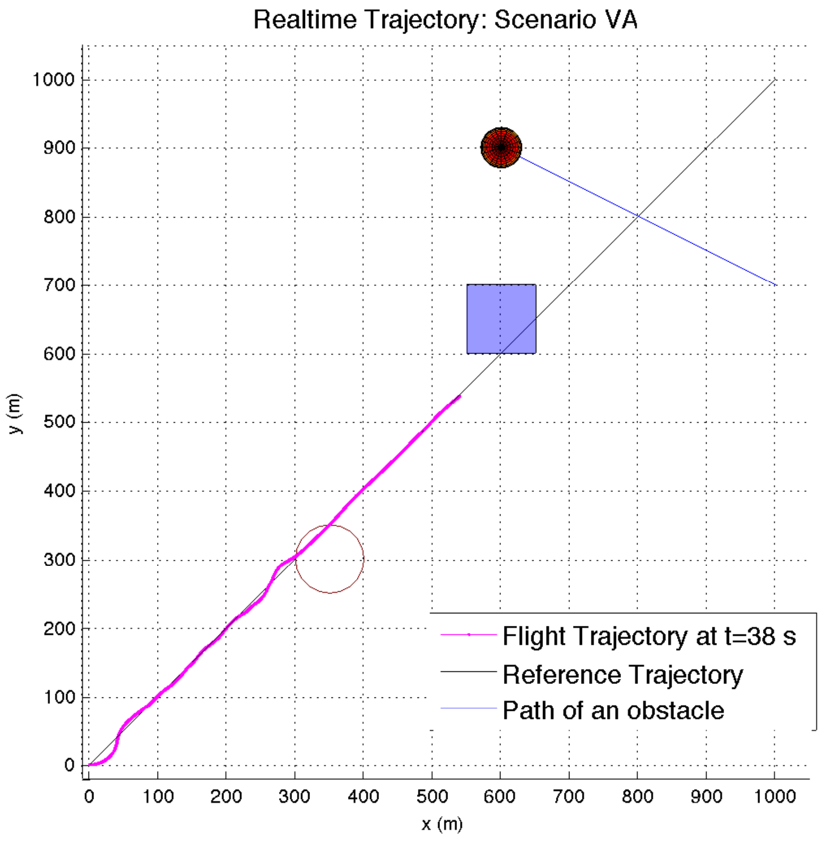
where, from equation (3.25), with ($w = 0.25$) on stationary and pop-up obstacle and ($w = 0.5$) on moving obstacle, we get

$$r(x,y,z) = 0.25 \left[(e^{e^{-h_1(x,y,z)}} - 1) + (e^{e^{-h_2(x,y,z)}} - 1) \right] + 0.5 \left[(e^{e^{-h_3(x,y,z)}} - 1) \right] \tag{4.20}$$

With 5 Hz updating frequency, and 40 nodes, the realtime simulation results are shown in Figures 4.129-4.136. The average time for solving the realtime problem is 24.32s and total maneuver time is 62.365s.

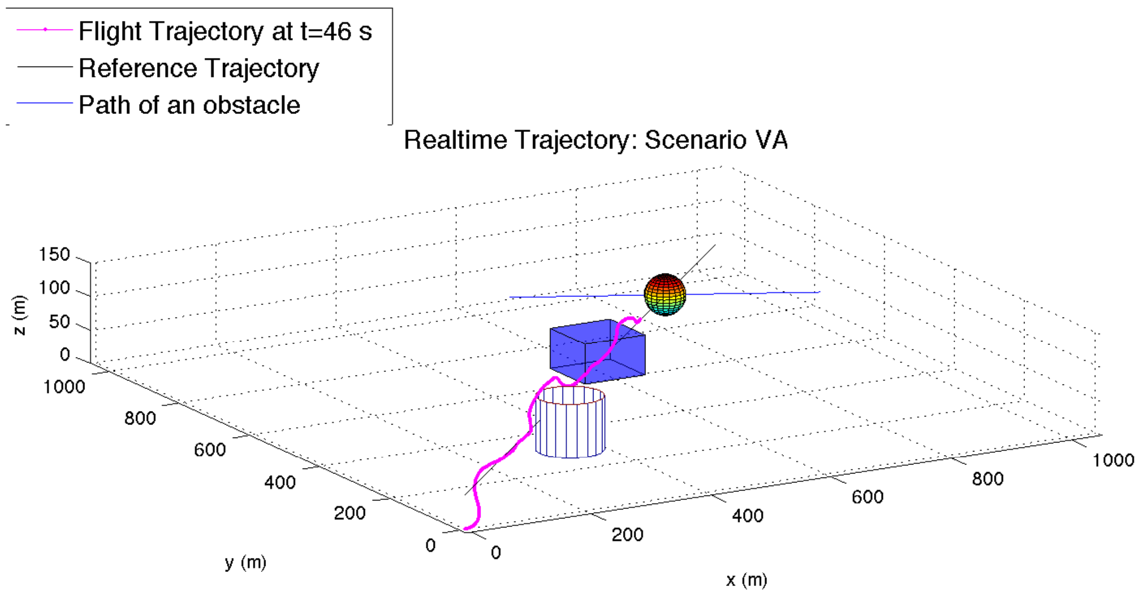


(a) 3D view

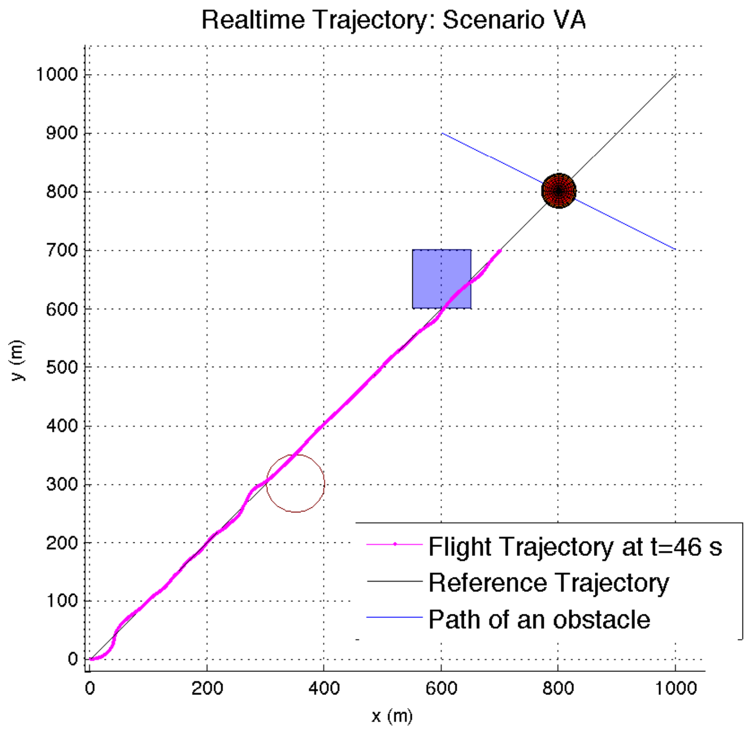


(b) View in XY Plane

Figure 4.129: Flight path at $t = 38s$

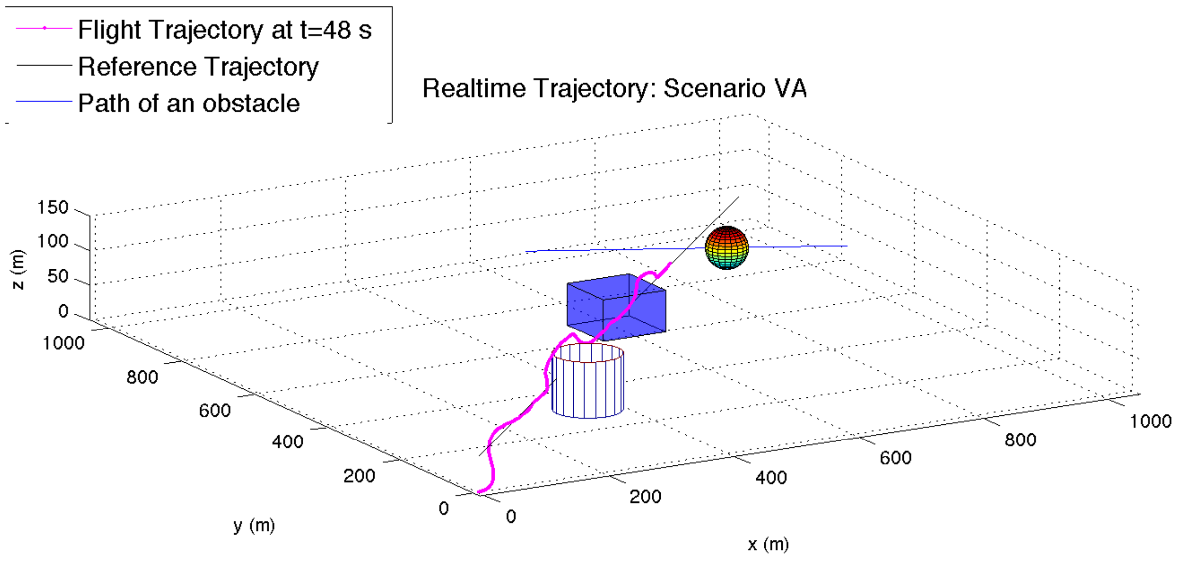


(a) 3D View

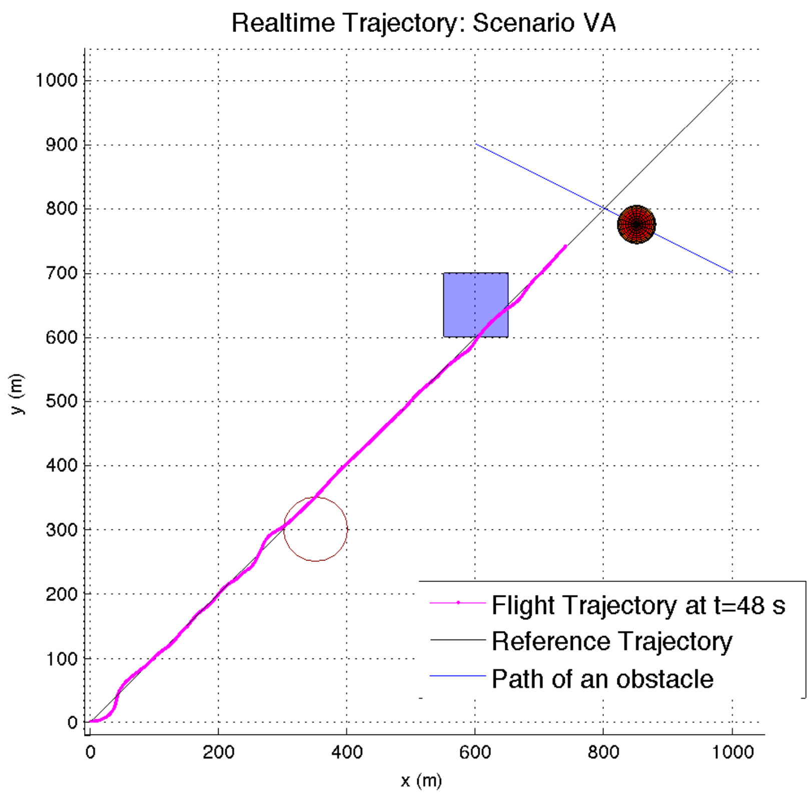


(b) View in XY Plane

Figure 4.130: Flight path at $t = 46\ s$

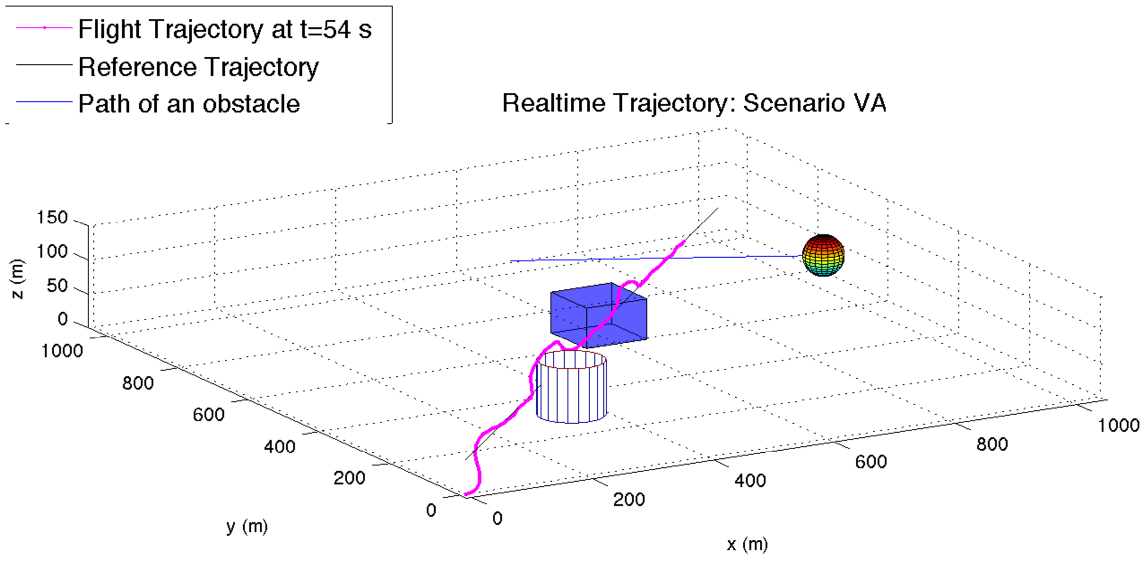


(a) 3D view

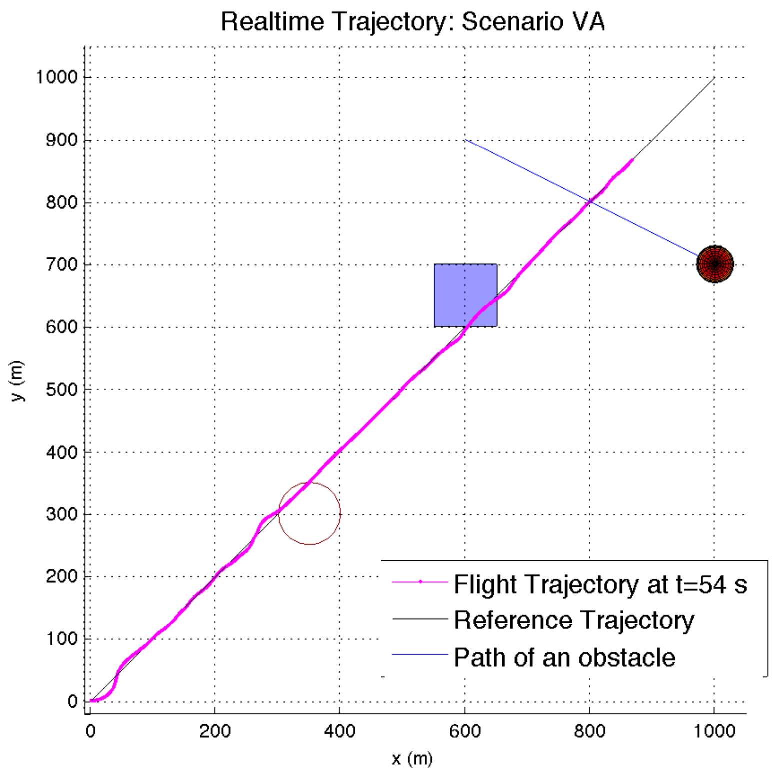


(b) View in XY Plane

Figure 4.131: Flight path at $t = 48s$

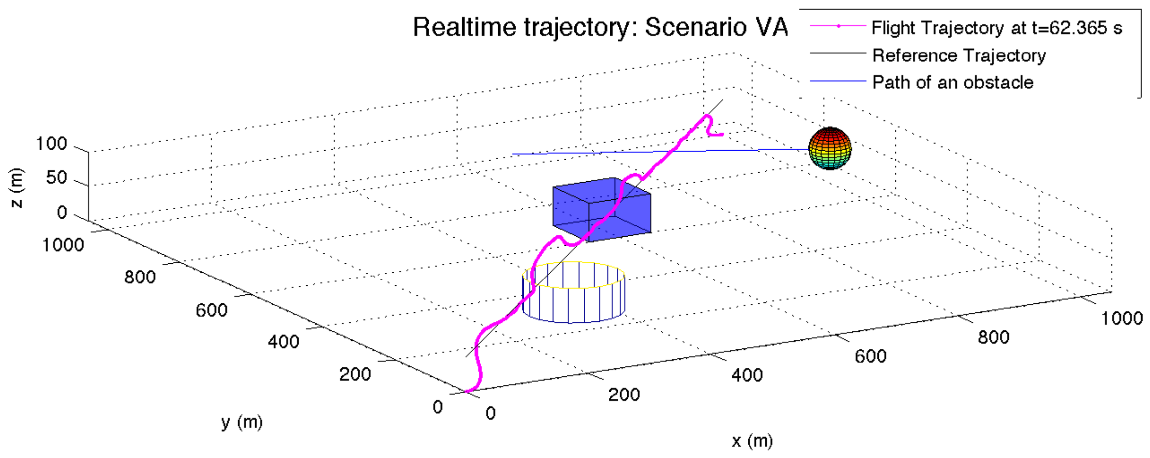


(a) 3D view

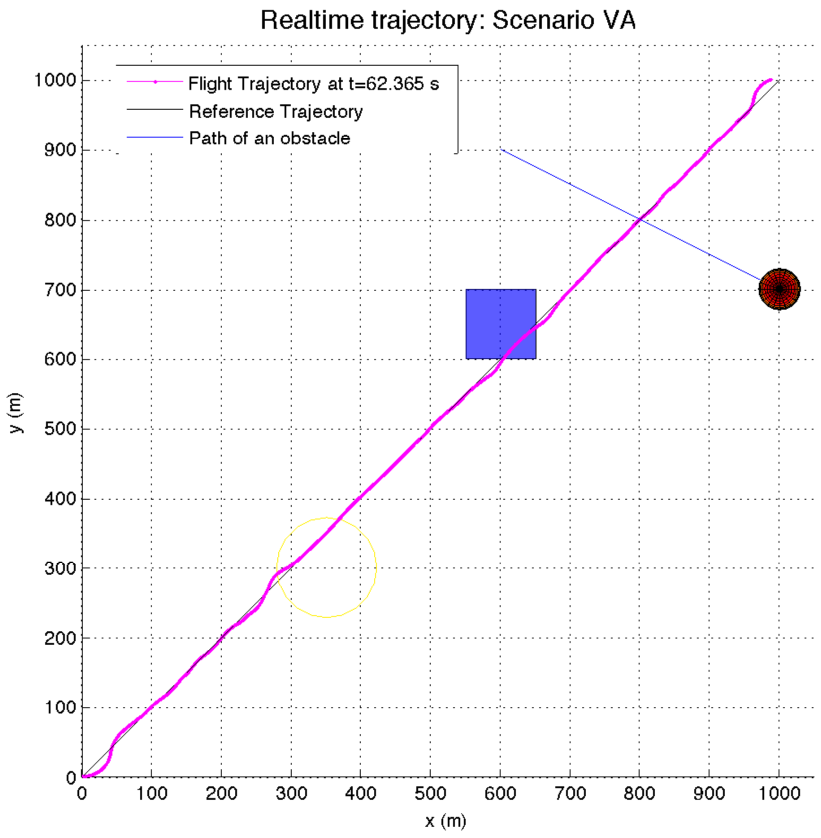


(b) View in XY Plane

Figure 4.132: Flight path at $t = 54s$



(a) 3D view



(b) View in XY Plane

Figure 4.133: Flight path at $t = 62.365s$

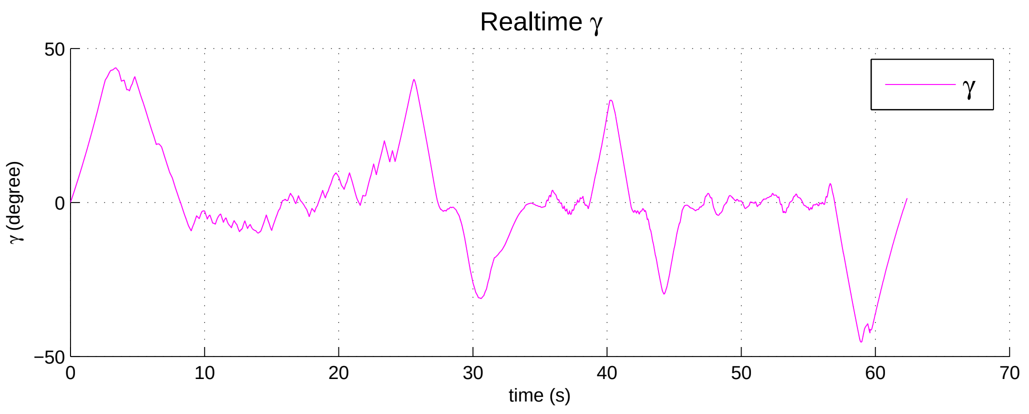


Figure 4.134: γvs t

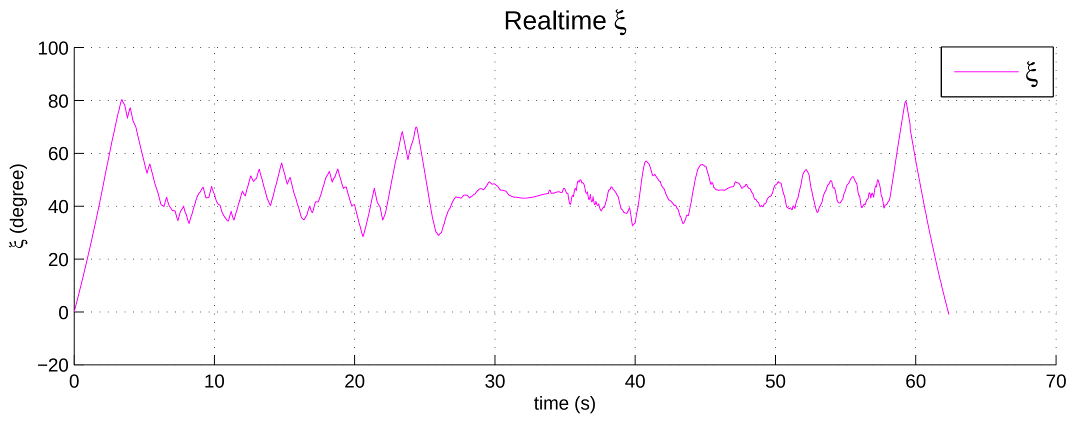


Figure 4.135: ξ vs t

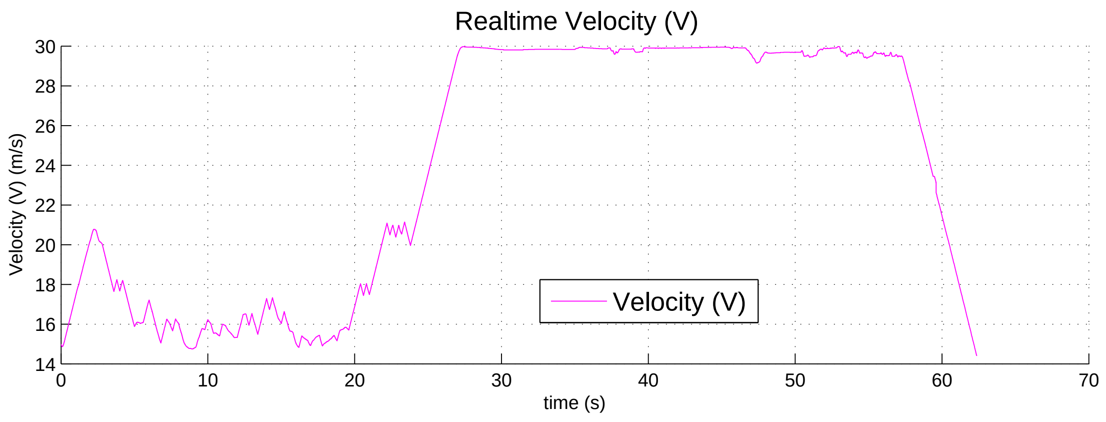


Figure 4.136: V vs t

As can be seen, the UAV successfully avoids all obstacles while staying close to the reference trajectory.

VB. Realtime: Stationary, Pop-up and Moving (in curve path) obstacles.

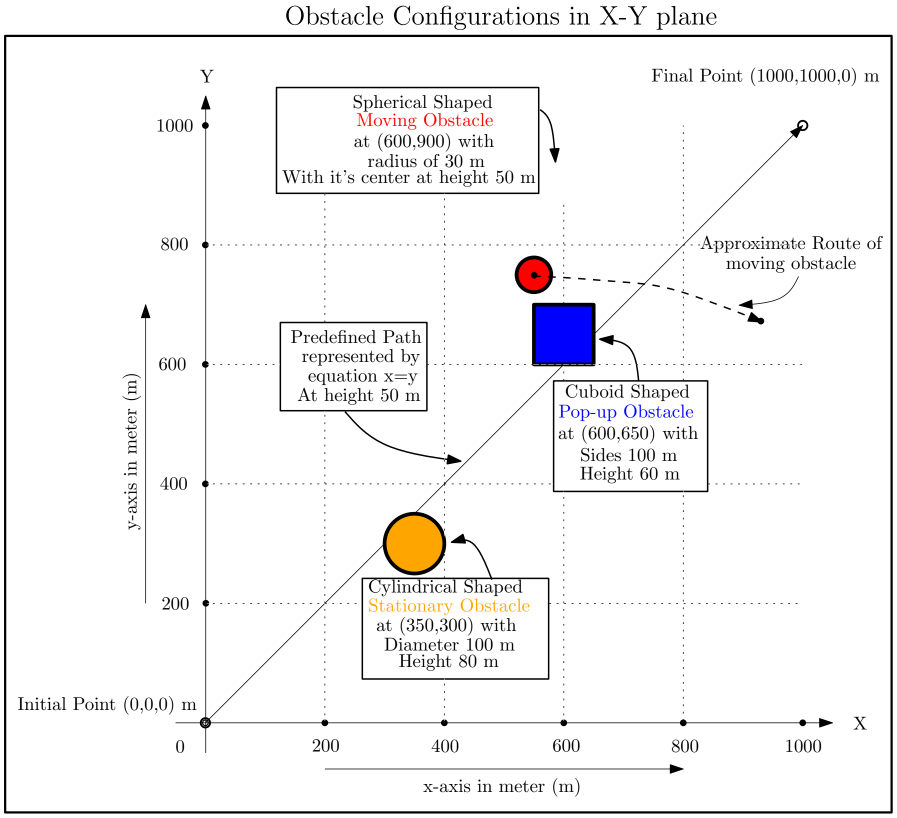


Figure 4.137: Scenario VB

In this case the moving obstacle changes its position in such a way that the $x-position$ of the UAV becomes the $x-position$ of the moving obstacle from 31 s until 51 s of simulation time and $y-position$ is given by

$$y = 600 + rsin(n)$$

at height of 50 m.

where,

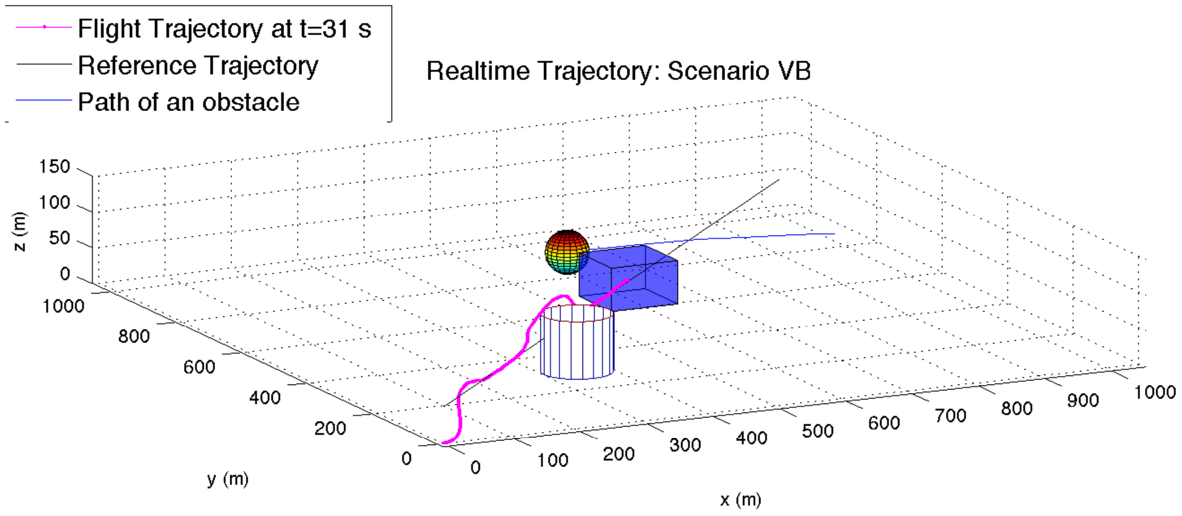
$$r = 100\sqrt{2}m$$

$$n = (90 - 0.5625(t - 31)5)$$

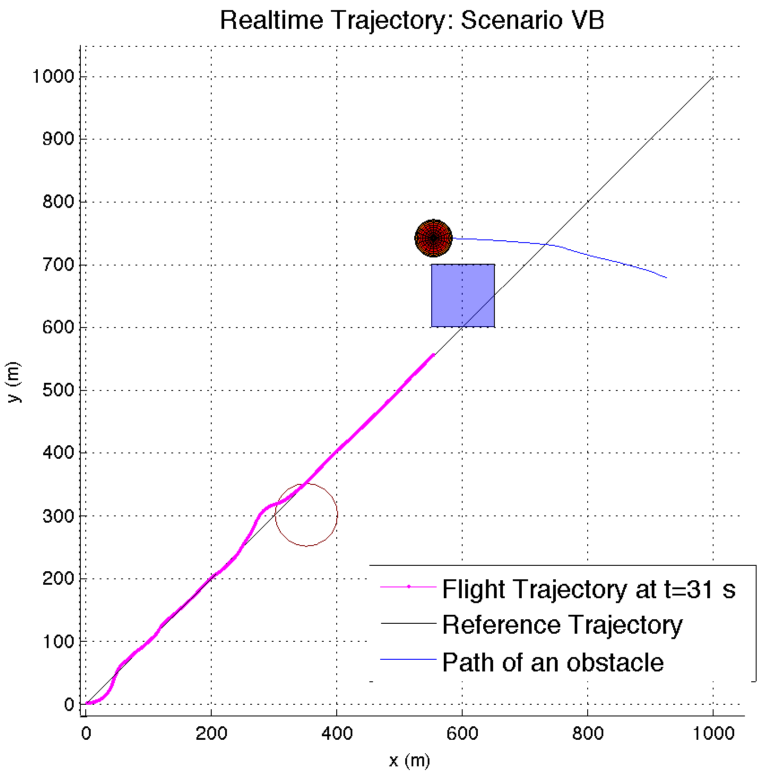
The decrease in the value of n by -0.5625 represents the change in the $y-position$ of the obstacle for every 0.2 s of simulation time from 31 s to 51 s. With this choice of obstacle $x-position$, the obstacle tends to target the UAV as it crosses its path. This represents a challenging obstacle for the UAV to avoid. The above expression for the $y-position$ is similar to **Scenario IIIB**. The problem formulation for **Scenario VB** is similar to that for **Scenario VA**, with the only difference being the path constraint due to the moving obstacle which is given in equation (4.21) and its associated robustness function. Similar to **Scenario IIIA**, it was found that with robustness factor (w) of 0.5 the moving obstacle collided with the UAV, hence the robustness factor (w) of the moving obstacle is further increased to 1.

$$h_1(x,y,z,t) := ln\left[\left|\left(\frac{x-x}{30}\right)^2\right| + \left|\left(\frac{y-(600+r\sin(n))}{30}\right)^2\right| + \cdots \right.$$
$$\left.\left|\left(\frac{z-50}{30}\right)^2\right|\right] \geq 0; 31s \leq t \leq 51s \tag{4.21}$$

As in the previous scenarios, a 5 Hz updating frequency and 40 nodes are used for each open-loop computation in realtime simulation. Figures 4.138 to 4.147 show the results. The average time for solving each open-loop solution in the realtime problem is 63.72 s with a total maneuver time of 55.43 s.

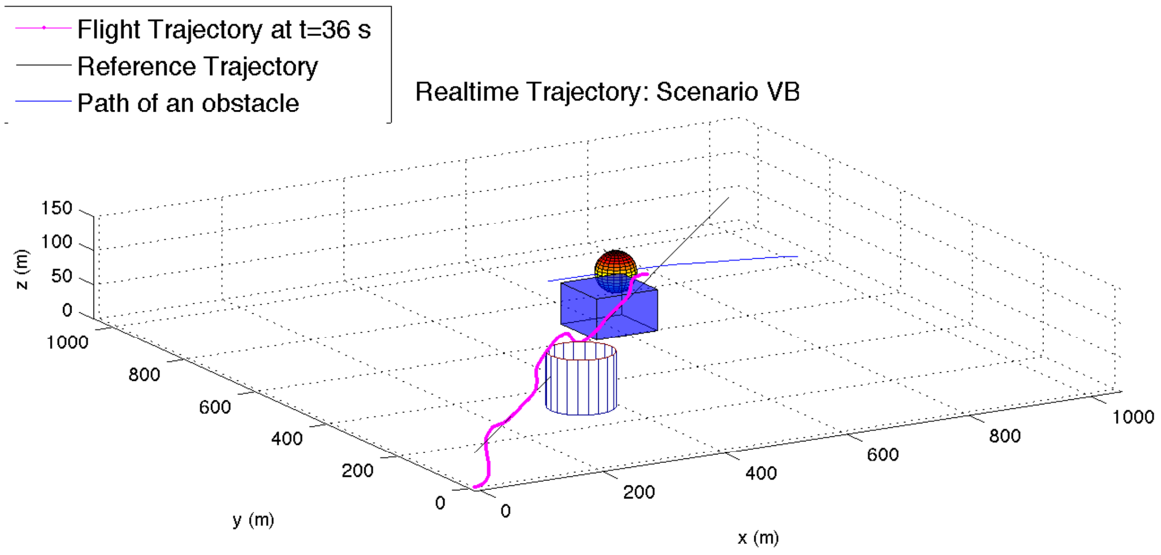


(a) 3D View

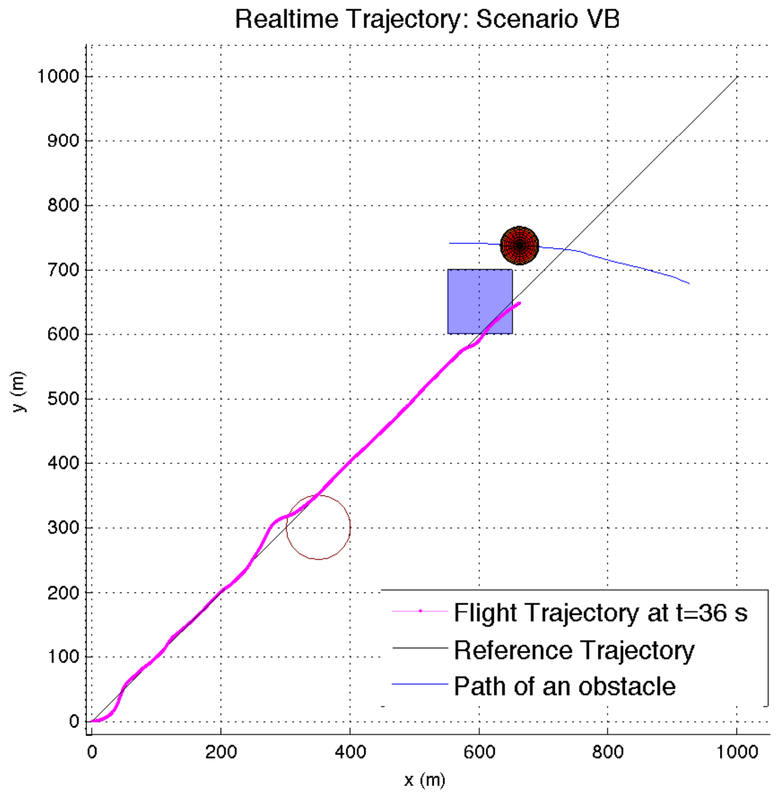


(b) View in XY Plane

Figure 4.138: Flight path at $t = 31s$

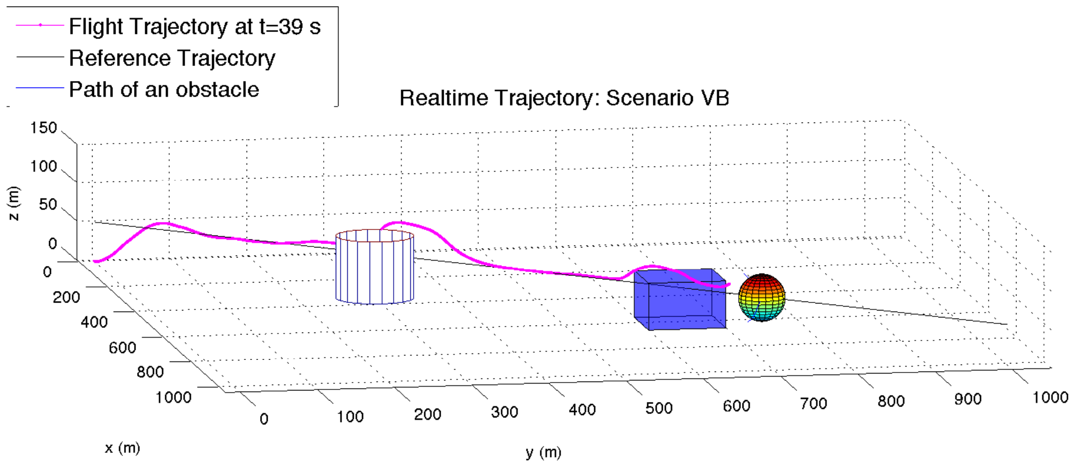


(a) 3D View

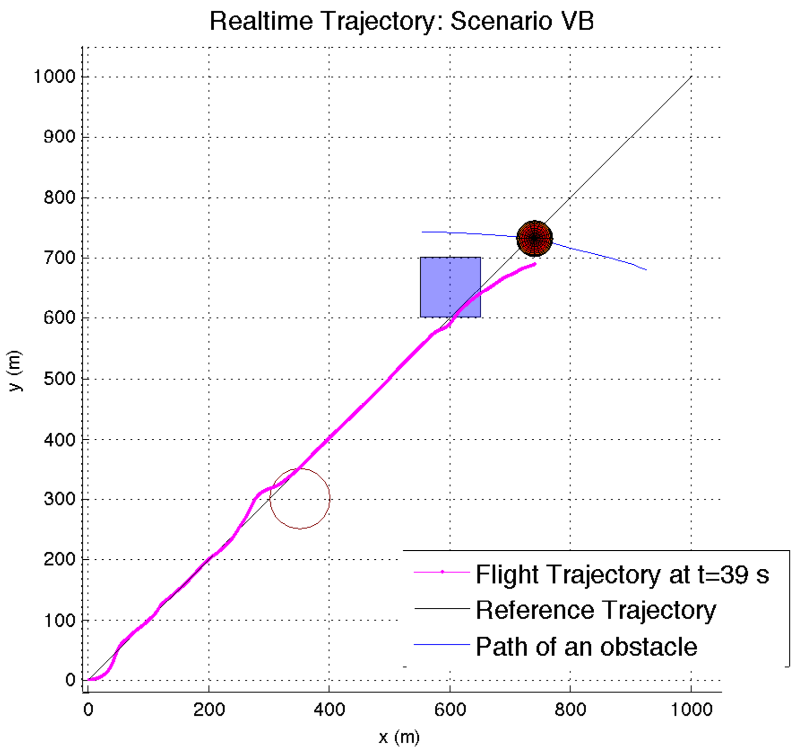


(b) View in XY Plane

Figure 4.139: Flight path at $t = 36s$

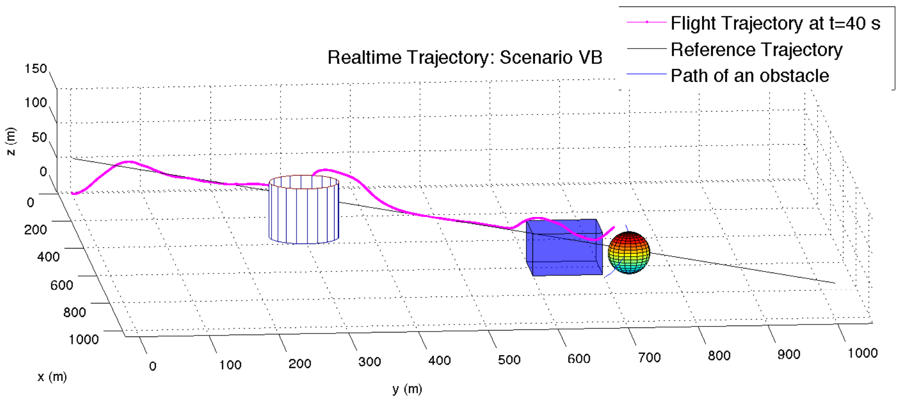


(a) 3D View

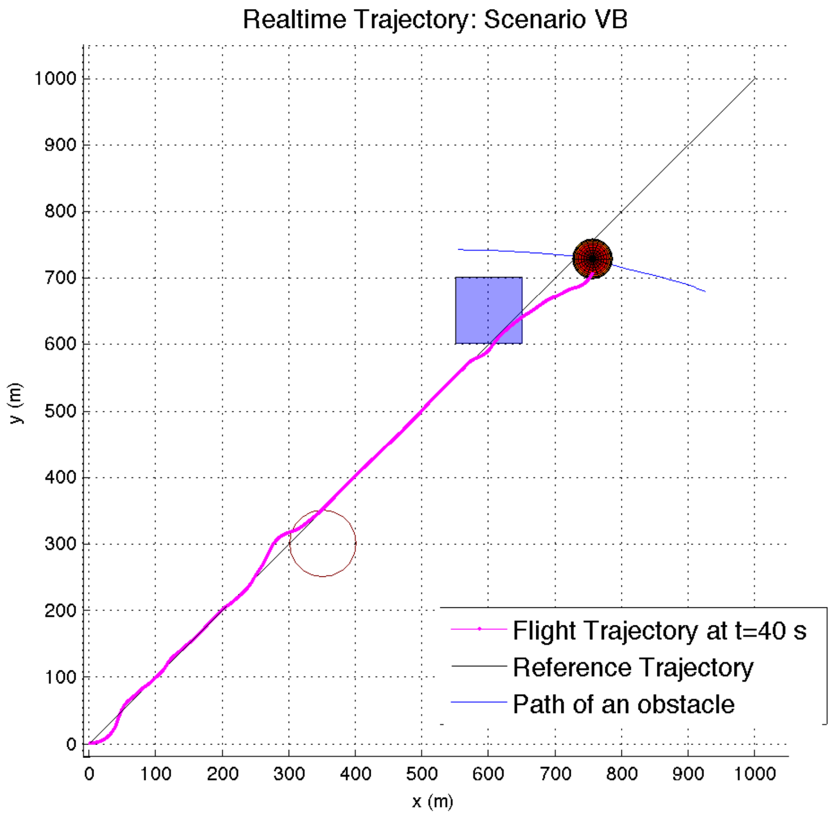


(b) View in XY Plane

Figure 4.140: Flight path at $t = 39s$

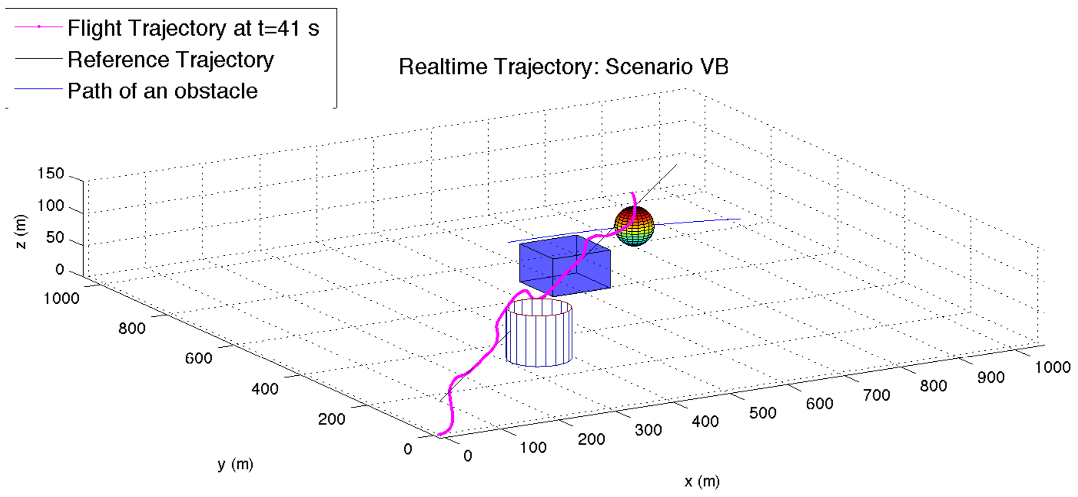


(a) 3D View

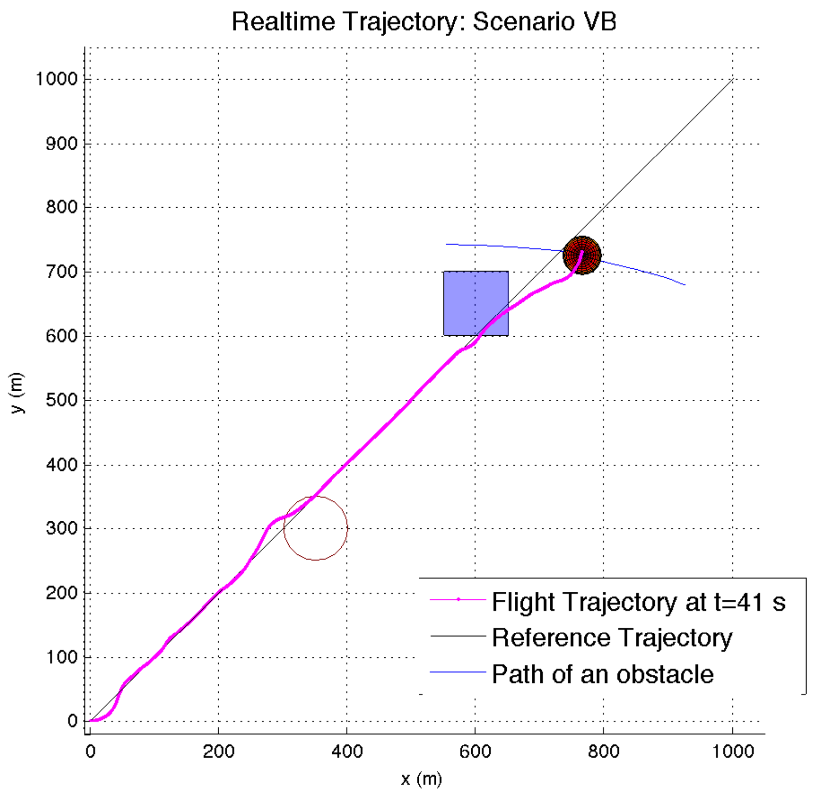


(b) View in XY Plane

Figure 4.141: Flight path at $t = 40s$

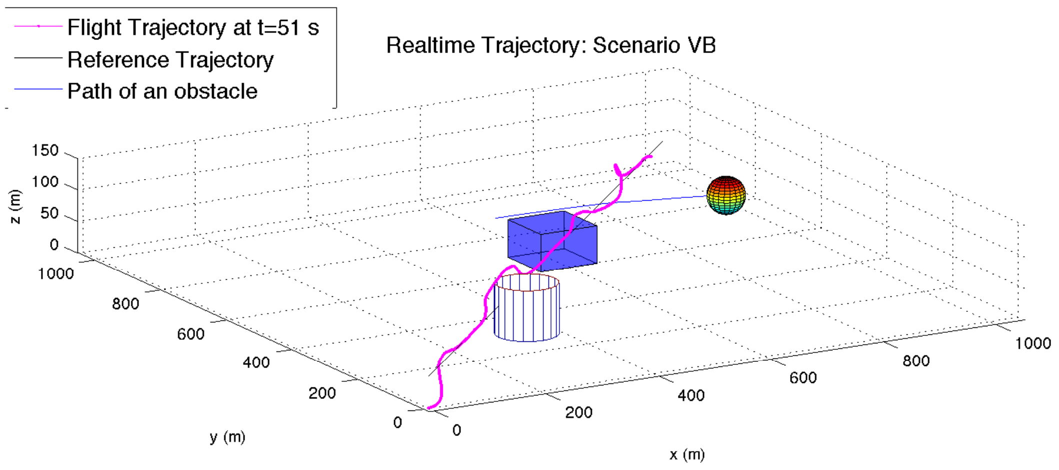


(a) 3D View

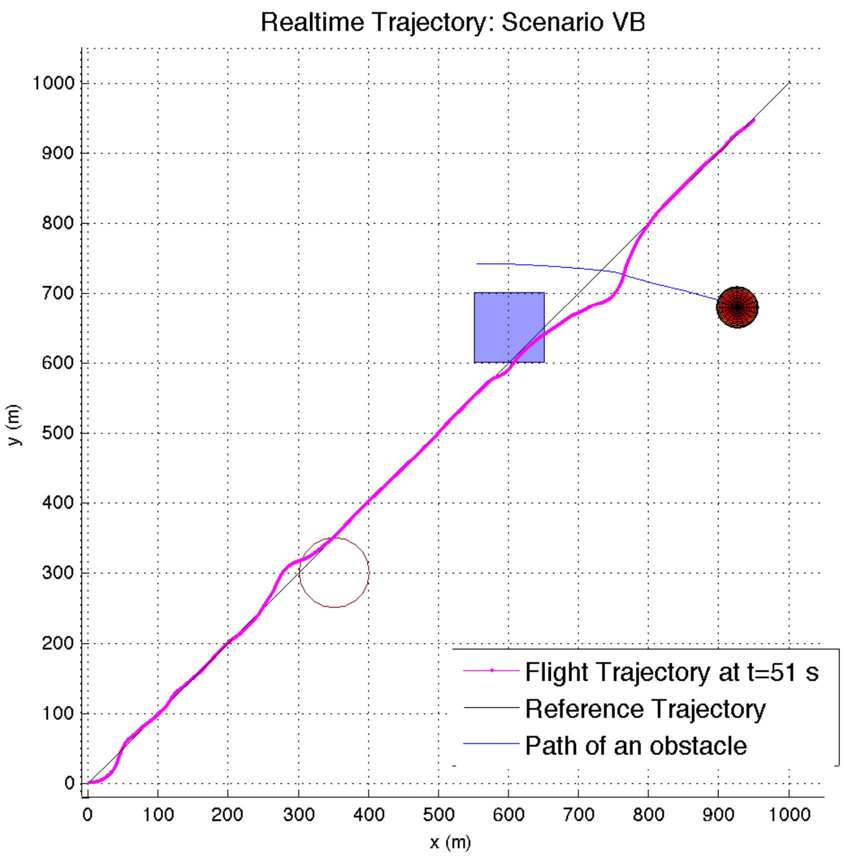


(b) View in XY Plane

Figure 4.142: Flight path at $t = 41s$

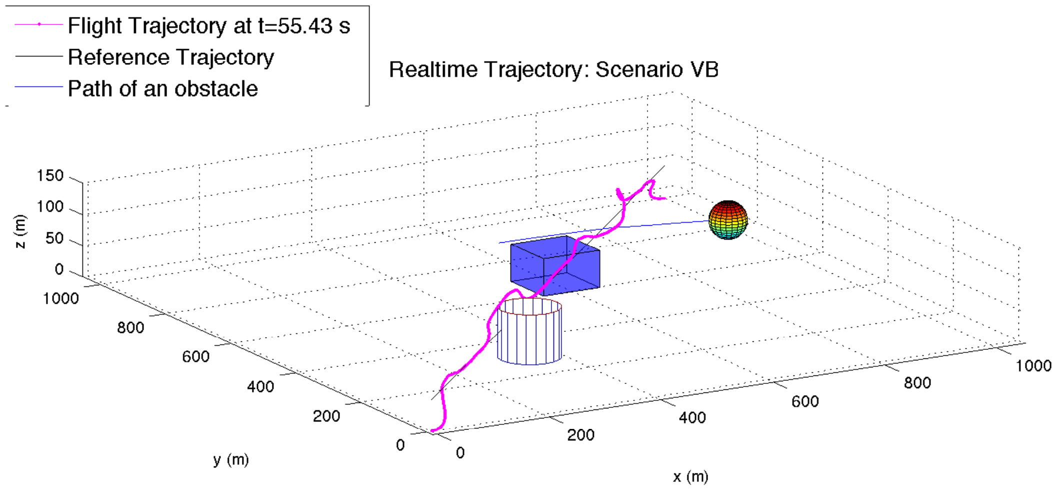


(a) 3D View

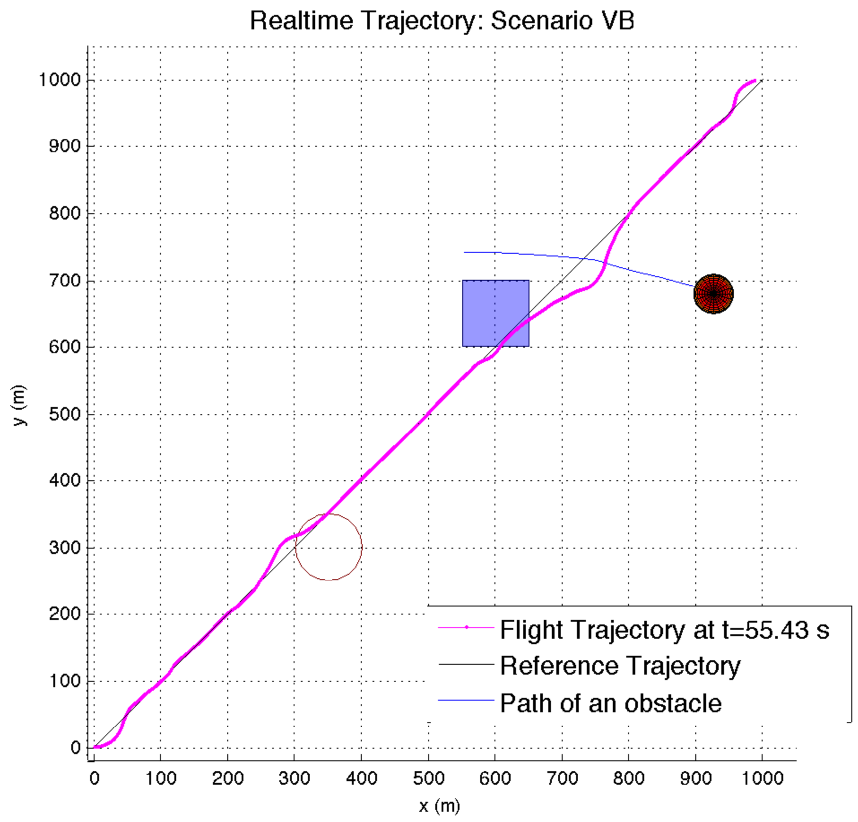


(b) View in XY Plane

Figure 4.143: Flight path at $t = 51s$



(a) 3D View



(b) View in XY Plane

Figure 4.144: Flight path at $t = 55.43s$

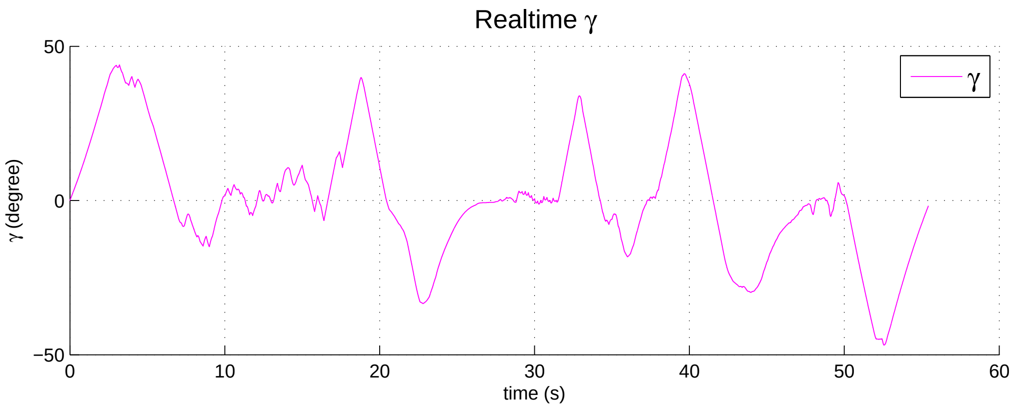


Figure 4.145: γvs t

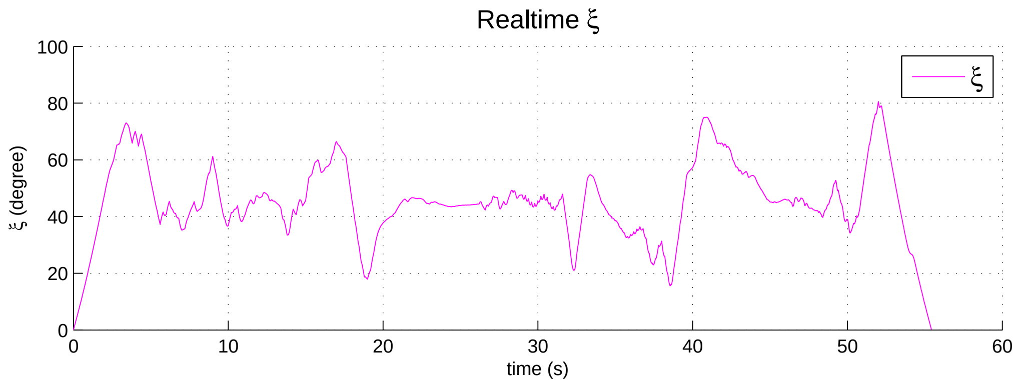


Figure 4.146: ξ vs t

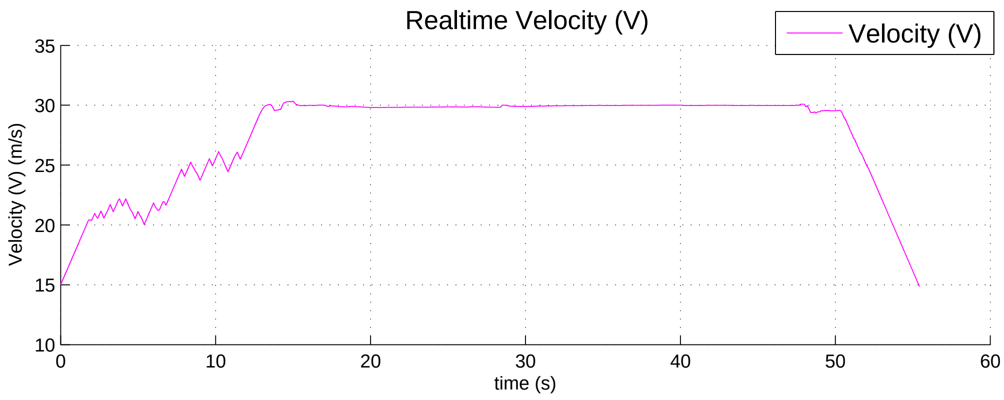


Figure 4.147: V vs t

As can be seen, the method can successfully avoid the stationary, pop-up and moving obstacles while remaining close to the reference trajectory even in this challenging scenario.

This scenario is even more challenging than would be expected to encounter in practice. For example, moving obstacle would be much smaller, e.g. birds.

Chapter 5

Conclusion

5.1 Research Conclusion

The objective of this thesis was to determine the capability of the Legendre Pseudospectral method of optimal control for use in real-time autonomous obstacle avoidance by fixed-wing UAVs performing geophysical surveys. The UAV is required to avoid different types of obstacles such as stationary, moving and pop-up obstacles, while remaining close to the pre-determined nominal survey flight path. At the same time, the method needs to respect the aircraft's physical and dynamic limitations.

In order to solve this problem, an appropriate optimal control problem was formulated, with the cost function defined so as to minimize the area between the actual flight path and the reference trajectory. As seen in the results, this particular definition directly leads to a flight path that remains close to the reference trajectory. A number of aircraft constraints were incorporated into the optimal control problem, including constraints on the aircraft turning and pull-up radii, climb and descend angles, aircraft acceleration and maximum and minimum speeds. Obstacles were modeled as path constraints. In addition to the obstacle path constraints, a robustness factor for each obstacle was included in the cost function, pushing the flight path further away from the obstacles for increased safety. Finally, the cost function has the option of including final maneuver time, so as to reduce the overall amount of time taken to perform an avoidance maneuver.

The formulated method has been tested numerically in a number of different scenarios, including stationary obstacles alone, moving obstacles alone, pop-up obstacles alone, and a combination of all three types of obstacles considered simultaneously. In every scenario considered, the UAV was able to successfully avoid the obstacles, while remaining close to the reference trajectory. Thus, this work has answered a key question regarding the applicability of the Legendre pseudospectral method of optimal control for real-time obstacle avoidance for the type of UAV mission under consideration. Namely, this work has demonstrated that provided the computational requirements of the Legendre pseudospectral method can be met on-board a UAV, it will be able to successfully guide the UAV around a number of different types of obstacles, while staying close to the nominal flight path. This represents a significant increase in capability over previously developed methods.

5.2 Future Work

The successful implementation of the developed method of obstacle avoidance for a fixed-wing UAV depends on the computational power on board as well as computational burden of the method and the maneuverability of the UAV. This thesis does not develop the end product to be implemented directly into UAV, but provides a necessary stepping stone to build one. The next important step would be to develop a customized tool to solve the optimal control problem. The following is a list of recommendations for future work before this method can be actually applied on the UAVs.

1. Development of the customized tool for the UAV to solve the optimal control problem, with a particular focus on reducing the computational burden. This would include looking at specific nonlinear program solvers, and the generation of suitable initial guesses.

2. Develop high fidelity UAV simulation with Autopilot and another realistic conditions including wind, measurement errors, obstacle detection errors and UAV limitations.

3. Integration of hardware and the customized tool to be used on the UAV

4. Test of the method in flight.

References

[1] F. K. Lu, ed., *Cooperative Path Planning of Unmanned Aerial Vehicles*, vol. 235, AIAA, A John Wiley and Sons, Ltd., Publication, November 2010.

[2] S. Bortoff, "Path-planning for unmanned air vehicles," *Proc. American Control Conf., Chicago*, pp. 364–368, June 28-30 2000.

[3] S. Griffiths and J. Saunders, "Maximizing miniature aerial vehicles," *IEEE Robotics and Automation Magazine*, vol. 13, Issue: 3, pp. 34–43, September 2006.

[4] J. D. Anderson, "Inventing flight: The wright brothers and their predecessors.," *Johns Hopkins University Press*, 2004.

[5] R. M. Thompson II, "Drones in domestic surveillance operations: Fourth amendment implications and legislative responses," *Congressional Research Service*, April 3 2013.

[6] T.-Y. Chou, M.-L. Yeh, Y.-C. Chen, and Y.-H. Chen, "Disaster monitoring and management by the unmanned aerial vehicle technology," *ISPRS TC VII Symposium*, vol. XXXVIII, Part 7B, July 5-7 2010.

[7] A. C. Watts, V. G. Ambrosia, and E. A. Hinkley, "Unmanned aircraft systems in remote sensing and scientific research: Classification and consideration of use," *Remote Sensing*, vol. 4, pp. 1671–1692, June 8 2012.

[8] L. P. Koh and S. A. Wich, "Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation," *Tropical Conservation Science*, vol. 5, pp. 121–132, July 9 2012.

[9] S. Owlia, "Real-time autonomous obstacles avoidance for low-altitude fixed-wing aircraft," *M.A.Sc Thesis, Carleton University*, January 2013.

[10] J. Barnard, "Use of unmanned air vehicles in oil, gas and mineral exploration activities," *AUVSI Unmanned Systems North America 2010 Conference*, vol. Session 44, August 26 2010, Colorado Conference Center.

[11] K. P. Bollino and L. R. Lewis, “Collision-free multi-uav optimal path planning and cooperative control for tactical applications,” *AIAA Guidance, Navigation and Control*, August 18-21, Honolulu, Hawaii 2008.

[12] A. Cavoukian, *Privacy and Drones: Unmanned Aerial Vehicles.* Toronto: Information and Privacy Commissioner, Ontario, Canada, August 2012.

[13] C. C. Haddal and J. Gertler, “Homeland security: Unmanned aerial vehicles and border surveillance,” *Congressional Research Service*, July 8 2010.

[14] B. Coifman, M. McCord, R. G. Mishalani, and K. Redmill, “Surface transportation surveillance from unmanned aerial vehicles,” *Proceedings of the 83rd Annual Meeting of the Transportation Research Board*, January 11-15 2004, Washington, DC.

[15] W. Akihisa, Y. Toshiaki, M. Masaaki, A. Takanari, A. Hideo, and T. Hirokazu, “A surveillance system using small unmanned aerial vehicle (uav) related technologies,” *NEC Technical Journal*, vol. 8, September 2013.

[16] D. W. Casbeer, S. M. Li, R. W. Beard, R. K. Mehra, and T. W. McLain, “Forest fire monitoring with multiple small uavs,” *Proceedings of the 2005 American Control Conference (IEEE)*, pp. 530–535, 8-10 June 2005, Portland Hilton and Towers in downtown Portland, Oregon.

[17] R. Forrester, S. Huq, M. Ahmadi, and P. Straznicky, “Magnetic signature attenuation of an unmanned aircraft system for aeromagnetic survey,” *IEEE/ASME Transactions on Mechatronics*, vol. 19, pp. 1436–1446, 2013.

[18] J. Maley, “An investigation into low-cost manufacturing of carbon epoxy composites and a novel "mouldless" technique using the vacuum assisted resin transfer moulding (vartm),” *MASc thesis, Carleton University*, May 2008.

[19] G. Fasano, D. Accardo, A. Moccia, C. Carbone, U. Ciniglio, F. Corraro, and S. Luongo, “Multi-sensor based fully autonomous non-cooperative collision avoidance system for uavs,” *Journal of Aerospace Computing, Information and Communication*, vol. 5, no. 10, pp. 338–360, 2008.

[20] A. Smith, D. Coulter, and C. Jones, “Uas collision encounter modeling and avoidance algorithm development for simulating collision avoidance,” *In "AIAA Modeling and Simulation Technologies Conference And Exhibit," AIAA*, 18-21 August 2008, Honolulu, Hawaii.

[21] P. Angelov, C. Bocaniala, C. Xideas, C. Patchett, D. Ansell, M. Everett, and G. Leng, "A passive approach to autonomous collision detection and avoidance," *In "Tenth International Conference on Computer Modeling and Simulation"*, pp. 64–69, 1-3 April 2008 IEEE Computer Society (2008), Cambridge, UK.

[22] L. Peng and Y. Lin, "A closed-form solution of horizontal maneuver to collision avoidance system for uavs.," *In "Control and Decision Conference (CCDC)"*, pp. 4416–4421, 26-28 May IEEE (2010), Xuzhou, China.

[23] S. Karaman and E. Frazzoli, "High-speed flight in an ergodic forest," *Robotics and Automation (ICRA), 2012 IEEE International Conference*, pp. 2899–2906, May 14-18 2012, Saint Paul, MN.

[24] L. A. Loeff and A. H. Soni., "An algorithm for computer guidance of a manipulator in between obstacles," *Transactions of the ASME. Series B, Journal of Engineering for Industry 97(3)*, pp. 836–842, 1975.

[25] L. Marce, M. Julliere, and H. Place, "Strategy of obstacle avoidance for a mobile robot," *RAIRO Automatique 15(1)*, pp. 5–18, 1981.

[26] K. L. Doty and S. Govindaraj, "Robot obstacle detection and avoidance determined by actuator torques and joint positions," *In "Conference Proceedings of IEEE Southeastcon '82"*, pp. 470–473, April 4-7 IEEE (1982), Sandestin, Destin, Florida.

[27] B. L. Chapman and N. Perreira, "Algorithm for intelligent control of a robot manipulator," *In "Proceedings of the International MOTORCON Conference"*, pp. 334–344, 19-21 April Intertec Communications Inc, Orlando, FL, USA (1983).

[28] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, pp. 90–98, March ISSN 0278-3649 (1986).

[29] M. Hwangbo, J. Kuffner, and T. Kanade, "Efficient two-phase 3d motion planning for small fixed-wing uavs," *In "IEEE International Conference on Robotics and Automation"*, pp. 1035–1041, 10-14 April Roma, Italy (2007).

[30] J. Amin, J. Boskovic, and R. Mehra, "A fast and efficient approach to path planning for unmanned vehicles," *In "AIAA Guidance, Navigation and Control Conference and Exhibit"*, 21-24 August AIAA 2006-6103. Keystone, Colorado (2006).

[31] J. Saunders, B. Call, and A. Curtis, “Static and dynamic obstacle avoidance in miniature air vehicles,” *In "Infotech@Aerospace"*, September AIAA 2005-6950. Arlington, Virginia (2005).

[32] S. M. La Valle, *Planning Algorithms.* Cambridge University Press, 2005.

[33] H. e. a. Choset, *Principles of Robot Motion: Theory, Algorithms and Implementation.* Intelligent Robotics and Autonomous Agents series, A Bradford Book, 2005.

[34] M. Zefran, *Continuous Methods for Motion Planning.* PhD thesis, University of Pennsylvania, December 1996.

[35] L. P. R. Lewis, “Rapid motion planning and autonomous obstacles avoidance for unmanned vehicles,” *Doctoral Dissertation, Naval Postgraduate School*, December 2006.

[36] B. Xu, D. Z. Chen, and R. J. Szczerba, “A new algorithm and simulation for computing optimal paths in a dynamic and weighted 2-d environment,” *IEEE*, vol. 1, pp. 313–318, 8-11 October 2000, Nashville, TN.

[37] P. Chandler, S. Rasmussen, and M. Pachter, “Uav cooperative path planning,” *AIAA Guidance, Navigation and Control Conf., Denver. CO, AIAA-2000-4370*, pp. 1301–1306, August 14-17 2000, Denver. CO.

[38] O. Khatib, “Real time obstacle avoidance for manipulators and mobile robots,” *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 500–505, 25-28 March 1985, St. Louis, MO, USA.

[39] L. C. Wang, L. S. Yong, A. Jr., and M. H., “Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment,” *In "International Symposium on Intelligent Control", Vancouver, Canada*, pp. 27–30, October 2002.

[40] F. Lingelbach, “Path planning for mobile manipulation using probabilistic cell decomposition,” *Proc. of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendal, Japan*, 2 October 2004.

[41] A. Poty, P. Melchior, and A. Oustaloup, “Dynamic path planning for mobile robots using fractional potential field,” *Control, Communications and Signal Processing, 2004. First International Symposium on, IEEE*, pp. 557–561, March 21-24 2004, Hammamet, Tunisia.

[42] M. Jun and R. D'Andrea, "Path planning for unmanned aerial vehicles in uncertain and adversarial environments.," *In "Cooperative Control: Models, Applications and Algorithms"*, vol. 1, pp. 95–110, 2003.

[43] I. M. Ross, "A begineer's guide to dido," *A MATLAB Application Package for Solving Optimal Control Problems (Ver. 7.3)*, 2007, Elissar, LLC.

[44] I. M. Ross and F. Fahroo, "Issues in the real-time computation of opitmal control," *Mathematical and Computer Modelling, ELSEVIER*, vol. 43, pp. 1172–1188, May 4 2006.

[45] J. R. Rea, "A legendre pseudospectral method for rapid optimization of launch vehicle trajectories," *Master Thesis*, 2001.

[46] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, March-April 1998.

[47] I. M. Ross, C. D'Souza, F. Fahroo, and J. B. Ross, "A fast approach to multi-stage launch vehicle trajectory optimization," *AIAA Guidance, Navigation and Control Conference and Exhibit, Austin, Texas*, August 11-14 2003.

[48] M. A. Hurni, P. Sekhavat, and I. M. Ross, "Issues on ugv optimal motion planning and obstacle avoidance," *AIAA Infotech@Aerospace Conference, Seattle, Washington*, April 6-9 2009.

[49] M. A. Hurni, P. Sekhavat, and I. M. Ross, "Autonomous trajectory planning using real-time information updates," *AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii*, August 18-21 2008.

[50] K. P. Bollino, "High-fidelity real-time trajectory optimization for resuable launch vehicles," *Doctoral Dissertation, Naval Postgraduate School*, 2006.

[51] I. M. Ross, "User's manual for dido: A matlab application package for solving optimal control problems,," *Tomlab Optimization, Sweden*, February 2004.

[52] P. Gill, W. Murray, and M. Saunders, "User's guide for snopt 5.3: A fortran package for large-sclae nonlinear programming.," *Technical Report, Stanford Business Software, Inc., Palo Alto, CA*, July 1998.

[53] I. M. Ross, "A primer on pontryagin's principle in optimal control," *Collegiate Publishers, San Francisco, Carmel*, 2009.

[54] R. F. Hartl, S. P. Sethi, and R. G. Vickson, "A survey of maximum principles for optimal control problems with state constraints," *SIAM*, vol. 37, no. 2, pp. 181–218, 1995.

[55] B. Fornberg, *A Practical Guide to Pseudospectral Methods,*. Cambridge Monographs on Applied and Computational Mathematics, Cambridge, United Kingdom: Cambridge University Press, 1998.

[56] L. Trefethen, *Spectral Methods in MATLAB.* Society for Industrial and Applied Mathematics,, Philadelphia, Pennsylvania: SIAM, 2000.

[57] I. M. Ross and M. Karpenko, "A review of pseudospectral optimal control: From theory of flight," *Annual Reviews in Control*, vol. 36, Issue: 2, pp. 182–197, December 2012.

[58] I. M. Ross and F. Fahroo, "Costate estimation by a legendre pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 24, pp. 270–277, March-April 2001.

[59] G. N. Elnagar and M. A. Kazemi, "Pseudospectral legendre-based optimal computation of nonlinear constrained variational problems," *Journal of Computational and Applied Mathematics, ELSEVIER*, no. 88, pp. 363–375, 1997.

[60] P. E. Gill, W. Murray, and M. A. Saunders, "User's guide for snopt 7: Software for large-scale nonlinear programming," June 16 2008.

[61] Q. Gong, I. M. Ross, and W. Kang, "A pseudospectral observer for nonlinear systems," *Discrete And Continuous Dynamical Systems-Series B*, vol. 8, October 2007.

[62] K. P. Bollino, I. M. Ross, and D. D. Doman, "Optimal nonlinear feedback guidance for reentry vehicles," *AIAA Guidance, Navigation and Control Conference and Exhibit*, 21-24 August 2006, Keystone, Colorado.

[63] I. Ross, P. Sekhavat, Q. Gong, and A. Fleming, "Pseudospectral feedback control: Foundations, examples and experimental results," *Proceedings of the 2006 AIAA Guidance, Navigation and Control Conference*, August 21-24 2006, Keystone Colorado.

[64] Q. Gong, W. Kang, and I. M. Ross, "A pseudospectral method for the optimal control of constrainted feedback linearizable systems," *IEEE Transactions on Automatic Control*, vol. 51, pp. 1115–1129, 10 July 2006.

[65] I. M. Ross and F. Fahroo, "A unified framework for real-time optimal control," *Proceedings of the IEEE Conference on Decision and Control, Maui*, vol. 3, pp. 2210–2215, 9-12 December 2003.

[66] I. M. Ross, Q. Gong, and P. Sekhavat, “Low thrust, high-accuracy trajectory optimization,” *Journal of Guidance, Control, and Dynamics*, vol. 30, pp. 921–933, July-August 2007.