2013

Development of an Integrated Intelligent Multi -Objective Framework for UAV Trajectory Generation

Jennifer Nicole Wilburn
West Virginia University

Development of an Integrated Intelligent Multi-Objective Framework for UAV Trajectory Generation

Jennifer Nicole Wilburn

Thesis submitted to the
Benjamin M. Statler College of
Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Mario G. Perhinschi, Ph.D., Chair
Larry E. Banta, Ph.D.
Yu Gu, Ph.D.
Wade W. Huebsch, Ph.D.
Powsiri Klinkhachorn, Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
August 2013

Keywords: Path and Trajectory Planning, Clothoid, Dubins, 3-Dimensional, Artificial Immune Optimization, Immunity-Based Evolutionary Optimization

ABSTRACT

Development of an Integrated Intelligent Multi-Objective Framework for UAV Trajectory Generation

Jennifer Nicole Wilburn

This thesis explores a variety of path planning and trajectory generation schemes intended for small, fixed-wing Unmanned Aerial Vehicles. Throughout this analysis, discrete and pose-based methods are investigated. Pose-based methods are the focus of this research due to their increased flexibility and typically lower computational overhead.

Path planning in 3 dimensions is also performed. The 3D Dubins methodology presented is an extension of a previously suggested approach and addresses both the mathematical formulation of the methodology, as well as an assessment of numerical issues encountered and the solutions implemented for these.

The main contribution of this thesis is a 3-dimensional clothoid trajectory generation algorithm, which produces flyable paths of continuous curvature to ensure a more followable commanded path. This methodology is an extension of the 3D Dubins method and the 2D clothoid method, which have been implemented herein. To ensure flyability of trajectories produced by 3D pose-based trajectory generation methodologies, a set of criteria are specified to limit the possible solutions to only those flyable by the aircraft. Additionally, several assumptions are made concerning the motion of the aircraft in order to simplify the path generation problem.

The 2D and 3D clothoid and Dubins trajectory planners are demonstrated through a trajectory tracking performance comparison between first the 2D Dubins and 2D clothoid methods using a position proportional-integral-derivative controller, then the 3D Dubins and 3D clothoid methods using both a position proportional-integral-derivative controller and an outer-loop non-linear dynamic inversion controller, within the WVU UAV Simulation Environment. These comparisons are demonstrated for both nominal and off-nominal conditions, and show that for both 2D and 3D implementations, the clothoid path planners yields paths with better trajectory tracking performance as compared to the Dubins path planners.

Finally, to increase the effectiveness and autonomy of these pose-based trajectory generation methodologies, an immunity-based evolutionary optimization algorithm is developed to select a viable and locally-optimal trajectory through an environment while observing desired points of interest and minimizing threat exposure, path length, and estimated fuel consumption. The algorithm is effective for both 2D and 3D routes, as well as combinations thereof. A brief demonstration is provided for this algorithm. Due to the calculation time requirements, this algorithm is recommended for offline use.

DEDICATION

I thank my husband, Brenton, for standing by me, no matter how trying that can be.

I thank my family and friends for their love and support throughout the process of this research and always.

Most importantly, I thank my advisor and friend, Dr. Mario Perhinschi, for his boundless patience, wisdom, and guidance, and for encouraging me to pursue dreams I had not yet dared to dream.

ACKNOWLEDGEMENTS

I would like to acknowledge the funding and support provided for this research effort, both from the Army Research Laboratory and from the West Virginia NASA Space Grant Consortium Graduate Research Fellowship Program. Without the support of these generous organizations, this research could not have been conducted.

I would like to acknowledge the significant time, effort, and support of my committee, whose comments and suggestions have been both constructive and educational.

Finally, I would like to acknowledge all the people who have taken care of me and helped me through the years to make this thesis possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Symbols

Symbol	Definition
A	Aircraft cruise altitude [m]
A	Affinity
a	Aircraft commanded lateral acceleration [m/s]
a	Magnitude of the connecting vector, $\lvert\overrightarrow{a_c}\rvert$
$\overrightarrow{a_c}$	Connecting vector
$\overrightarrow{a_f}$	Vector from end of connecting vector to finish arc center
$\overrightarrow{a_s}$	Vector from beginning of connecting vector to start arc center
$\hat{b}$	Binormal unit vector of Frenet-Serret Frame
$C(h)$	Fresnel cosine integral evaluated at h
c	Centerline distance between two primary circles [m]
c_{ij}	Length of path segment i overlapped by threat j
d	Filleting distance
d_i	Direction indicator for turn i
d_{ij}	Distance from radar j to path segment i
E	Circle exit tangent point
$\vec{F}(q)$	Total force acting on aircraft at position q
g	Acceleration due to gravity [m/s^2]
h	Clothoid curve length parameter
$\bar{h}$	Clothoid curve normalized path length
h_{max}	Clothoid curve maximum length
I	Circle entry tangent point
K	Threat-to-length weighting factor
L_i	Length of path segment i
$\hat{n}$	Normal unit vector of Frenet-Serret Frame
P	Total path cost
P	Pose
P_i	Event probability of risk zone i
p_i	Cost of path segment i
$\vec{p}$	Vector from position of start pose to position of finish pose
PP	Path points for Point of Interest Observation Path Planner

$\mathbf{q}$	Current position of aircraft in 2D solution space
q	Clothoid curvature parameter
$\mathbf{q}_{goal}$	Position of the goal
$\mathbf{q}_{obs}$	Position of the obstacle center
$\bar{q}$	Clothoid curve normalized parameter
R	Turning radius [m]
R_{AB}	Rotation matrix to convert vector components from reference frame B to reference frame A
R_{min}	Aircraft minimum turning radius [m]
R_{obs}	Obstacle impact radius
RI_i	Risk intensity of risk zone i
$\vec{r_f}$	Vector from finish pose position to finish arc center
$\vec{r_s}$	Vector from start pose position to start arc center
$\overrightarrow{r_{AB}}$	Position vector of point B with respect to point A
$S(h)$	Fresnel sine integral evaluated at h
S_i	Event severity of risk zone i
$\hat{t}$	Tangent unit vector of Frenet-Serret Frame
$U(q)$	Potential function at position q
$\vec{u}$	Vector denoted u
$[\vec{u}]_A$	Components of vector $\vec{u}$ with respect to reference frame A
u_{x_A}	X-axis component of $[\vec{u}]_A$
u_{y_A}	Y-axis component of $[\vec{u}]_A$
u_{z_A}	Z-axis component of $[\vec{u}]_A$
v	Aircraft cruise velocity [m/s]
$x(h)$	Fresnel cosine integral evaluated at h
X_A, x_A	X-coordinate with respect to reference frame A
X, x	X-coordinate with respect to Earth Reference Frame
$y(h)$	Fresnel sine integral evaluated at h
Y_A, y_A	Y-coordinate with respect to reference frame A
Y, y	Y-coordinate with respect to Earth Reference Frame
Z_A, z_A	Z-coordinate with respect to reference frame A
Z, z	Z-coordinate with respect to Earth Reference Frame
α	Angle between centerline and tangent line of two primary circles [rad]
$\vec{\alpha_s}$	Vector pointing from beginning of connecting vector to start clothoid

	arc center
$\overrightarrow{\alpha_f}$	Vector pointing from end of connecting vector to clothoid finish arc center
$\hat{\alpha}$	Unit direction vector of $\overrightarrow{a_c}$
β	Slope of the centerline between two primary circles [rad]
$\vec{\beta}$	Unit vector perpendicular to $\overrightarrow{a_c}$
γ	Commanded steady-state climb angle [rad]
γ_{max}	Maximum flyable steady-state climb angle [rad]
δ	Angle between 2 adjoining path segments
η	Repulsive potential gain
θ	Pose pitch angle [rad]
κ_{max}	Aircraft maximum turning curvature [1/m]
λ_i	Length cost of path segment i
μ	Arc sweep angle [rad]
ξ	Attractive potential gain
$\overrightarrow{\rho_f}$	Vector from finish pose position to finish clothoid arc center
$\rho_{goal}(q)$	Distance between current position and goal position
ρ_0	Minimum distance of influence [m]
$\overrightarrow{\rho_s}$	Vector from start pose position to start clothoid arc center
σ_{ij}	Threat cost of path segment i with respect to radar j
τ	Tangent location angle from center of circle [rad]
ϕ	2D clothoid sweep angle [rad]
ϕ	Maneuver plane rotation angle [rad]
ϕ_{max}	Aircraft maximum bank angle [rad]
ψ	Pose heading angle [rad]

Subscripts

a, A	With respect to connecting vector reference frame
att	Attractive
Curvature	Curvature Profile Affinity
E	With respect to Earth reference frame
f, F	With respect to finish pose reference frame

Fuel Consumption	Fuel Consumption Affinity Function
goal	Related to goal location
i	Related to object i
n	Normal component
Path Length	Path Length Affinity
rep	Repulsive
s, S	With respect to start pose reference frame
t	Tangential component
Threat Exposure	Threat Exposure Affinity
x, X	X-component
y, Y	Y-component

Acronyms

IBEPO	Immunity-Based Evolutionary Pose Optimization
IBEO	Immunity-Based Evolutionary Optimization
LL	Left-Left, Left start turn direction, left finish turn direction
LR	Left-Right, Left start turn direction, right finish turn direction
POI	Point of interest
RL	Right-Left, Right start turn direction, left finish turn direction
RR	Right-Right, Right start turn direction, right finish turn direction
UAV	Unmanned Aerial Vehicle

Chapter 1. INTRODUCTION AND MOTIVATION

In recent years, the use and prevalence of Unmanned Aerial Vehicles (UAVs) have widely increased. UAVs are now quite often replacing human operators to conduct tasks which are considered to be too dangerous or tedious for a human pilot to carry out (1). As the use of UAVs increases, so too does the desire for them to perform more complicated missions. Improved adaptivity and situational awareness allow more control to be transferred to the UAV, thus enabling them to complete progressively more complex missions, relieving the burden on human pilots.

In order for the UAV to achieve greater levels of autonomy and robustness, one of the most important aspects which must be improved is path planning and trajectory generation. Path planning is defining the position in space along which the aircraft should travel. Trajectory generation involves specifying when in time the aircraft should arrive at individual points in space along the path. For practical purposes, a key problem with path planning is the number of methods available. To date, a multitude of path planning methodologies have been developed, but none which is fully capable of finding an optimal, safe, and flyable path through an unknown 3-dimensional environment, in response to varying types of threats, while observing several goals, and under abnormal flight conditions. An important aspect of this research effort lies in that it implements, investigates, and compares many of the most commonly applied planning algorithms, in an effort to narrow the selection for future applications.

The resulting selection for this research is that of pose-based planning methods. Many methods for generating the trajectory connecting two aircraft poses in space have been developed. The main contribution of this research is the development and demonstration of a three-dimensional clothoid-based path planning and trajectory generation algorithm, capable of producing paths of continuous curvature connecting two poses. For completeness, an integrated, multi-objective methodology for generating those poses has been developed, based on the clonal selection mechanism present in the biological immune system. This thesis presents the development and demonstration of an artificial immunity-based evolutionary optimization algorithm for automated selection of an adequate set of poses which allow the aircraft to safely traverse the surrounding environment while observing all points of interest along the path. To ensure that neither the pose selection algorithm nor the path planner may command a trajectory which exceeds the dynamic constraints of the aircraft, a concise set of dynamic constraints relevant to this path planning methodology are presented. All algorithms in this thesis are implemented through the Matlab®/Simulink environment. Contributions of this research have been published in the resources cited in Appendix D.

This document is organized as follows. Chapter 2 will present a literature review of the most common of the myriad path planning methodologies developed for solving the various trajectory objectives.

Chapter 3 contains a complete explanation of the environment representation scheme which will be used throughout this research. At the onset of this research, several common path planners were implemented and modified to accommodate the environment representation scheme defined for this research, in order to gain more insight into their relative performance and capabilities. Additionally, several new methodologies were developed based upon logical expansions of these existing approaches. Chapter 4 contains an in-depth description of the discrete path planning methodologies implemented in this research effort. Chapter 5 presents a thorough explanation of the 2-dimensional pose-based, or waypoint-based, trajectory generation algorithms developed and expanded throughout the course of this research. Next, Chapter 6 provides a detailed description and implementation methodology for the 3-dimensional path planners implemented and developed, including the new 3-D Clothoid trajectory generation algorithm for continuous curvature constraint. Chapter 7 then provides the background, development, implementation, and demonstration of the immunity-based optimization algorithm for pose selection. Chapter 8 is devoted to presenting a trajectory tracking comparison of the clothoid and Dubins path planning methodologies for both 2D and 3D approaches. Chapter 9 will then summarize the conclusions drawn during the course of this path planning study.

Chapter 2. LITERATURE REVIEW

Path planning is an important and widely-addressed field of research, as every mobile robotic system requires the determination of a suitable path. Unmanned aerial vehicles are no exception to this. The high dynamic complexity of the aircraft platform make path planning for UAVs a non-trivial task, beyond simple geometric path generation; a task which has been approached in countless ways over the years. Additionally, as the autonomy of UAVs is improved, human-reliance upon them increases, driving the expectation of the tasks which they should complete. As part of this process, increasingly robust and intelligent path planning is necessary. This chapter is dedicated to outlining some of the major approaches which have been performed in the past to attempt to solve the problem of UAV path planning.

A. Discrete Methods

a. Potential Field

The potential field paradigm (2) is a vastly applied discrete trajectory generation algorithm, whose popularity is due in large part to its relatively low calculation overhead and intrinsic online computation capability. The potential field is usually applied in 2-dimensions, but is also utilized in 3-dimensional navigation schemes.

The potential field paradigm focuses around treating the goal location as a point of attractive potential, treating obstacles as points of repulsive potential, and treating the vehicle as a point mass. Thus, at each time step the summation of forces, or potential, acting on the mass is calculated to determine the new velocity (magnitude and direction) of the mass within the environment, or field. Since the trajectory is recomputed at each step, the potential field lends itself to dynamic environments, as they are handled in primarily the same manner as would a static environment.

In 1986, Khatib (3) formulated a potential field path planner for application with mobile robots and robotic manipulators capable of planning with respect to moving obstacles. This was made possible by augmenting the potential functions of the obstacles with a time component such that the potential field varied with time. This scheme was demonstrated in real-time for a PUMA 560 robotic manipulator using visual sensing.

In 1993, Akishita et al. (4) proposed another potential field method for moving obstacles which utilized Laplace potential functions. This method was successfully demonstrated for a variety of simple cases combining moving and stationary obstacles using a mobile robot with a camera for positioning and connected to a computer for processing.

Potential field paradigm has one well-defined and well-documented flaw. In certain situations, the potential field will exhibit a local minimum. For instance, this often occurs directly behind an obstacle or in concave locations in the obstacles with respect to the goal. A large majority of research applying the potential field paradigm focuses on making it more robust to these issues.

In 2005, Fu-Guang et al. (5), in order to improve the potential field flaw of local minima, integrated into the potential field concept an additional virtual force which causes the object to have a tendency to gravitate toward open space, thus forcing the vehicle further from the obstacles and avoiding the location containing the minimum. Rather than this force contributing at all times, the force is incorporated in situations when the resultant force on the vehicle approaches zero, signifying a minimum. The method approaches both types of local minima commonly encountered during application of the potential field approach. First is the minimum which occurs when an obstacle is directly between the vehicle and the goal. In this case the vehicle halts directly behind the obstacle. The second situation occurs when an obstacle configuration creates a concave obstacle profile, and it becomes very likely for the vehicle to be trapped between the edges of the obstacles, since the potential function tends to force the vehicle into the depression. The intention is that the additional force component in each of these situations can help the vehicle to follow the obstacle boundary and recover from the stuck state. Although this will not produce an optimal path, which would avoid sticking altogether, the simulation results show that the traditional potential field gets stuck while the modified potential field does indeed right itself from the minima. It is noted that vehicle dynamics are not considered in this application.

Another extrapolation of the potential field concept which is intended to compensate for the occurrence of local minima is presented by Olunloyo and Ayomoh (6). They modify the architecture with the Virtual Obstacle Concept and the Virtual Goal Concept, intended to improve the performance of the algorithm in response to concave obstacle configurations. In general, these concepts involve inserting repulsive and attractive potentials which are not related to the obstacles or goal along the path of the vehicle to drive the vehicle away from local minima. Though both of these approaches had been developed prior to this work, this was the first instance in which they were used in conjunction. This method was only demonstrated in simple simulation exercises, but was shown to achieve good results.

Another phenomenon which occurs during the application of the potential field for path planning is the inability to reach the goal when obstacles in the vicinity are too close. This is the issue which Ge et al. (7) seek to rectify. In this application, a new repulsive potential function is introduced which greatly increases the attractive potential of the goal as the vehicle approaches, ensuring that the global minimum actually occurs at the goal, ensuring that the goal is reached by the vehicle. This method has an additional side effect in that if

the scaling factors in the attractive and repulsive potential functions are chosen carefully, the occurrence of local minima can be avoided or eliminated.

Hwang and Ahuja (8) apply the potential field approach to a 3-degrees of freedom robot, in a seldom-seen 3-dimensional approach to path planning. In this application, the path planner is divided into two parts. The obstacles are assigned the usual electrostatic potential field but the goal is not assigned. The global path planner, as they refer to it, then selects the path through the "valleys" in the potential field and narrows down the choice of orientations which lead to an unsafe path. Then the local path planner modifies this path for the dynamics of the vehicle to derive an actual collision-free trajectory. If it is determined that the path found by the local planner is unacceptable the global planner chooses a new path, and the process repeats until an acceptable solution is found or until there are no more possible paths. Both planners make use of the total potential seen by the robot in order to arrive at a more preferable solution.

Another 3-dimensional example of potential field path planning presented by Suzuki et al. (9) is applied to a swarm of UAVs for decentralized collision and obstacle avoidance as well as formation flight. The authors have proved stability for this system using Lyapunov's Second Theorem, and have demonstrated the viability of the method using both simulation and hardware tests.

Benefits and Drawbacks of Potential Field Methods

The potential field methods provide a low-overhead, simple solution to the trajectory generation problem, leading to their popularity. However, they suffer from several drawbacks. Potential fields have a high possibility of driving the vehicle to a local minimum which causes the method to fail. In the event that this does not occur, even properly tuned potential fields still yield typically longer-than-necessary paths through the environment. Additionally, potential field methods are a balance between goal-seeking and obstacle avoidance, so these cannot be guaranteed to satisfy both of these requirements for all situations.

b. Road Map Methods

Road map path planning methods (10) break down a continuous solution space into a finite number of discrete segments, from which the optimal path is chosen based on some criteria. Road map methods are often applied in situations in which a shortest collision-free path between two locations is desirable. In these material-point methods, position of the vehicle is of primary importance, while heading is arbitrary. Road map methods are widely applied to the area of ground-based robotics due to the 2-dimensional nature of these vehicles, though also applied for use with UAVs under simplified performance circumstances.

b.1 Visibility Diagrams

Visibility diagrams are a well-documented method for finding a shortest path through a polygonal obstacle field. In 2008, Berg et al. (11) devote a full chapter in their text Computational Geometry to detailing

the visibility diagram method of path planning for application with 2D mobile robots. In general, a visibility diagram generates paths based upon connecting the vertices of the obstacles in the environment which can be "seen" from one another, or rather whose connecting lines are not intersected by another obstacle. This algorithm grows increasingly computationally intensive as the number of obstacles, or more importantly vertices, is increased.

To highlight this drawback, Kaluder et al. (12) performed a study on the visibility diagram approach, in efforts of reducing the computational load imposed by the traditional methodology. Improvements to the computational load were implemented, versus the traditional algorithm, by utilizing a radial sweep method for detecting connections between vertices, referred to as dual transformation. This mechanism changes the entire flow of the algorithm, removing the necessity to process all vertices, only those which fall on a convex edge of an obstacle.

Gao et al. (13) applied the visibility diagram for use in an observation-based path planner for a mobile robot equipped with a limited array of sensors. The purpose of the research was to be able to compute an efficient path through which the robot may travel and be able to see all of the points of interest from at least one location along the path. In an atypical use of the visibility diagram, the path is planned by constructing the visibility diagram for the sensor of the vehicle. This diagram is used to define a "hallway" through which the vehicle should travel in order to observe all the points of interest. The optimal path must lie through this hallway, and is chosen based upon the arcs of visibility, or arcs connecting the even points. The results show that this method is fully capable of achieving the goal so long as the environment boundary is known a priori.

Finally, Omar and Gu (14) applied visibility diagrams to the area of UAV path planning due to the optimal nature of the paths produced. However, it is noted that the computational load of this algorithm is prohibitive to real-time computation. In order to combat this, a new version of the algorithm is introduced in which only the obstacles crossing the straight line connecting the start and goal positions are taken into account in the visibility diagram. The algorithm, thus, can compute a straight path to the goal while only swerving to avoid obstacles directly in the way. Additionally, this algorithm may be recomputed online such that unknown obstacles may be avoided as well.

b.2 Cell Decomposition

Cell decomposition is another elementary approach to path planning. In this method, the solution space is broken up into a uniform grid of nodes, connected by path segments, which form the cells. Cells containing the obstacles, as well as these cells' associated nodes and path segments, are removed from the solution space as invalid. Additionally, cells may be expanded or condensed by removing the unnecessary

nodes which lay in unobstructed areas and adjusting the connecting path segments accordingly. This method is commonly applied to 2-dimensional path planning, but it may also be applied in 3 dimensions, though this is seldom used due to computational intensity.

One computational downfall of cell decomposition is the need for a predetermined and accurate map. In 2007, Arney (15) enhanced the cell decomposition method by only using an approximate representation of the environment in a scheme referred to as approximate cell decomposition. In this method, the solution space is divided into a small-resolution regularly spaced grid of cells. Each cell is assigned a probability of occupancy, or how likely it is that the cell contains an obstacle. A threshold is applied, based upon experimental means, to decide which cells are occupied and which represent free navigation space. In order to group the areas of free and occupied space, a quad-tree method, or 2^m-tree with m=2, is applied. Here the solution space is divided into four cells, and each cell is again divided into four cells until each cell contains only free or occupied space, or until the maximum resolution is met. Then, all free cells which are located adjacent to each other are joined. In this methodology, the computation required to perform these steps is removed by recording relative addresses for each of the cell through the decomposition process.

Another type of cell decomposition is probabilistic cell decomposition. In this approach, the occupation status of each cell is not known a priori, but is guessed based on a series of checks. If all checks are negative, the cell is assumed to be free space. If all checks are positive, the cell is assumed to be occupied. However, if the checks are not all positive or negative, the cell is subdivided according to the dimension of the solution space and subsequent checks are performed on each of the resulting cells. As shown by Lingelbach (16) (17), this method is applicable and effective even in higher-dimension solution spaces, such as those for manipulators.

b.3 Voronoi Diagrams

A Voronoi diagram, or Dirichlet tessellation, is a manner of splitting up a 2-dimensional, or higher, space based upon a series of generating points, such that convex polygons, called Voronoi cells, are formed which each contain exactly one generating point, and any point contained within the interior of the Voronoi cell is closer to that generating point than any other (18). Voronoi diagrams are applicable to higher dimensions, but are only typically applied to path planning in 2 dimensions. Due to the nature of producing line segments, or path segments as they become in path planning, at points equally far from two impact points, these diagrams lend themselves to obstacle avoidance, particularly for the case of minimizing exposure to radar. Novy and Jacques (19) applied the Voronoi diagram to aircraft for just such a purpose. Trajectories are generated through a field of identical (equal weight) radar transmitters to generate a path of shortest length, while minimizing exposure purely due to the nature of the Voronoi diagram. Trajectories were also

generated in response to radars of varying weights, representing varying transmission power, to simulate production of a path focused upon minimizing exposure. The resulting approach produced a viable path which can be generated online in real-time.

In a similar approach by Hammouri and Matalgah (20), the Voronoi tessellation is produced to avoid radio interference which could cause loss of communication between the UAV and the ground station. Interference sources are weighted, so that path segments receive a rating based upon the highest interference point along the segment, and path segments outside of the safe communication threshold can be avoided during the path planning process. This process also incorporates a dynamic environment, as the Voronoi diagram is updated online in response to new information, allowing the UAV to reroute and avoid areas of low signal-to-interference-plus-noise-ratio.

In another radar application of the Voronoi diagram by Bortoff (21), the Voronoi diagram is generated in the standard way based on the radar locations. However, smoothing of the chosen path is performed in a different way. The path is simultaneously smoothed and optimized by treating the Voronoi nodes and path segments as a virtual mass-spring system, with the radars exerting a repulsive force on the masses.

Chandler et al. (22) provide a radar-based Voronoi path planner which is intended to cause multiple UAVs to arrive at the target point simultaneously. Pop-up radars are incorporated to assess online replanning capabilities. This methodology was demonstrated in simulation to be capable of providing rapid response to new threats while maintaining rigid arrival constraints.

Judd and McLain (23) incorporate the Voronoi diagram path planning methodology with a spline-based approach to produce flyable paths for UAVs. The Dijkstra's algorithm is used to select the optimal trajectory, based upon exposure to radars. Then, the straight-line path is smoothed to a flyable trajectory using a cubic spline smoothing mechanism. This approach was made more computationally efficient by decomposing the problem into sections. However, it still proved to be somewhat too slow for online computation. This path planning approach was then extended by McLain and Beard (24) for use with multiple UAVs for coordinated arrival. In this approach, three simulated vehicles have different start and goal points, as well as varying distances to the goals. In this smoothing method, nodes are treated as masses, and path segments are treated as chains, or springs. Threats exert repulsive forces on the masses, causing the path to curve away from threats. Additional constraints are needed to ensure that the resulting curvature is flyable by the UAV. The smoothing mechanism is used to modify the length of the paths such that all paths are of the same length, resulting in coordinated arrival times.

In a more complex application developed by Liu and Zhang (25), a pseudo-three-dimensional algorithm is derived which generates a 2-dimensional Voronoi graph based on the known threat environment. Threats are weighted to give exposure ratings to each of the Voronoi path segments. The optimal path is then chosen using a Dijkstra's algorithm. The path is then smoothed via cubic spline interpolation to remove sharp, unflyable corners. Finally, to produce the flyable 3-dimensional path, Geographic Information Systems data is used in conjunction with the limitations of the aircraft to plan the vertical path of the aircraft for avoiding geographic environmental features. Additionally, pop-up threats are incorporated using replanning if there are a number of threats in the immediate area of the new threat, and simple "steer around" mechanism to avoid replanning if the new threat is far enough from the other threats. Simulation results for this method showed it to be effective and feasible for online real-time calculation.

In a fairly straightforward implementation of the Voronoi diagram as performed by Ho and Liu (26), the Voronoi diagram is generated, a best path is chosen based upon the Dijkstra's algorithm, and the path is smoothed using Bezier curves. As an additional component of this method, the Voronoi path is intended not to minimize threat exposure to radar, but to avoid obstacles, in addition to maintaining at least a nearly optimal trajectory. This is accomplished through a four-step smoothing process. The Voronoi nodes along the path, as well as the initial and final desired locations, are chosen as the control points for the composite Bezier curve. Since using control points which are very close together can cause unnecessarily long paths, any "crowded" control points are eliminated. Next the control points are grouped into subsequences such that the resulting curve does violate an obstacle. Finally, additional control points are added to the path to satisfy the aircraft curvature constraints.

Another common method for applying the Voronoi diagram, more often seen in ground-based robotics, is the polygon obstacle implementation. This is the method applied and improved upon by Bhattacharya and Gavrilova (27) (28). In this method, obstacles are represented as disjoint polygons. The Voronoi diagram is generated based upon the start and goal points, as well as the vertices of all of the obstacle polygons. The start and goal points are then connected to the Voronoi grid by generating path segments to all of the Voronoi nodes of the Voronoi cell containing each of these end points. All Voronoi paths which violate the minimum clearance threshold, including those crossing the obstacle boundaries, are then removed. Additionally, paths were added at the minimum clearance distance around the outer edge of the road-map. The shortest path is then chosen by Dijkstra's algorithm. The final path is then optimized by removing unnecessary turns in the path which will not cause a violation of the minimum clearance criterion. Finally, the path is smoothed by introduction of Steiner points and an iterative corner-cutting technique.

b.4 Benefits and Drawbacks of Road Map Methods

In the area of UAVs, path planning in 3-dimensional space is of high importance. Accordingly, discrete methods are useful for UAVs in that they provide simple solutions to the problem of path planning and many are readily expandable to 3-dimensional solutions. However, it is often the case with discrete methods, that even in 2-dimensional space, the solutions are computationally cumbersome and impractical for online recalculation. This impedes the ability of the UAV to compensate for unknown and dynamic obstacles. Even in situations where these algorithms have been improved to provide adequately quick computation time, they typically lack the ability to specify the path the aircraft follows, but rather need a subsequent path selection method. Additionally, none of these methods directly produces either an optimal or a flyable path without post-processing.

B. Pose-Based Methods

Pose-based trajectory generation methods utilize a series of waypoints with associated headings called poses to construct a path through the environment. This paradigm is convenient as it allows the human operator to specify the waypoint the aircraft should traverse with having to define the complete path. This methodology is flexible and is often used to ensure that the aircraft avoids obstacles, as well as surveys the appropriate areas of interest in the environment.

a. Dubins

Probably the most general pose-based path planning methodology is the Dubins algorithm (29), developed by L.E. Dubins in 1957. This 2D path planning algorithm uses combinations of circular arcs and straight line segments connected at the relevant tangent points to generate a flyable trajectory. This method was featured in (10), based on a compilation of this group's previous work in cooperative UAV path planning, and has been utilized by a variety of researchers, including (30), (31) , (32) , and (33).

In 2011, Said and Sundaraj (30) applied the Dubins path planning methodology to a non-holonomic mobile robot, based on the fact that this planning mechanism produces the shortest path in a two-dimensional environment. The simple, analytical nature of this path planner led to the choice for its application in this research. Since this research involved a ground-based vehicle, the Dubins method was expanded with the Reeds-Shepp model, which incorporates the ability of the ground vehicle to stop and travel in reverse. This research covered a full explanation of the path combinations and the relative situations when one path would be selected over another.

Also in 2011, Hanson et al. (31) applied the Dubins methodology to a ground-based vehicle simulation for the purpose of target observation using discrete ranged sensors. The location on the vehicle of the sensors and the viewing angle required for each of the targets are used to specify the desired poses for the

Dubins algorithm. Additionally, the Dubins methodology used by Hanson et al. was compared to a brute-force search algorithm in the process of evaluating the effectiveness of the sensors and the optimal configurations of targets within the environment.

Prior to the previously discussed works, which capitalized on the 2-dimensional nature of ground-based vehicles, Grymin and Crassidis (32) applied the Dubins methodology for UAV trajectory generation. The goal of this research was to develop a simplified model of an aerodynamically-realistic aircraft model, based on the Dubins-vehicle paradigm (vehicle with fixed turning radius), in conjunction with the development of a Dubins-based waypoint navigation scheme. Their model simplifies the dynamics of the aircraft using the assumptions of a velocity-holding controller and an altitude holding controller, since Dubins paths assume constant velocity and are generated in 2 dimensions. This research also covered the development of a hybrid Rhumb-Line/Dubins trajectory tracking controller.

The Dubins methodology was also applied to UAVs to provide a path-planning scheme for a team of UAVs by Jeyaramen et al. (33). In that research, the goal was to produce paths for each of the aircraft such that they observe the desired points of interest and arrive at the goal at the same time without colliding. That amounts to producing a different trajectory for each aircraft, all of the same length, which do not cross each other at the same moment in time. Their efforts combined the previous work of Jeyaramen et al. (34) and Shanmugavel et al. (35).

b. Clothoid

A far more specialized and less common 2D pose-based planner is the clothoid trajectory generation method. In this path planner, the path curves produced possess a continuous curvature profile. This is an important path characteristic, since the majority of commonly-used UAV platforms lack sufficient response rates to adequately follow a Dubins trajectory. By specifying a continuous curvature profile, the acceleration command becomes a gradual or ramp command rather than the step profile obtained for Dubins paths.

In 1996, Scheuer and Fraichard (36) (37) proposed the use of clothoid arcs for path planning of non-holonomic car-like robots in order to eliminate the necessity for the robot to pause, turn the wheels, and resume moving, which is necessary when following a Dubins path. Their method used an iterative combination of two methods called "search" and "explore" to arrive at an acceptable path solution between two configurations, composed of combinations of elementary paths, where an elementary path is the concatenation of 2 symmetric clothoid curves to achieve a continuous curvature profile. "Explore" approximates the locations reachable from the initial position. "Search" seeks an acceptable solution by finding intermediate configurations between the two specified configurations through which to pass which

minimizes the length of the path. This method trades completeness of the solution for computational efficiency. However, this efficiency was demonstrated through a series of simulation results.

In a clothoid-approximation methodology by Montes et al.. (38), an online path planner capable of generating clothoid curves is created using specifically generated parametric rational Bezier curves, which are generated offline then scaled online for use in path planning. In essence, this method can be applied in the same ways and with the same benefits of the clothoid curve, but with lower online computation, assuming the rational Bezier curves have been thoroughly and accurately created offline.

One of the most published and prominent applications of the clothoid path planning methodology is that documented by Tsourdos, White, and Shanmugavel (10) (39). In this research effort, trajectory planning for a number of cooperative UAVs is desired. This approach documents the geometric approach to the solution of the clothoid system, wherein two poses are connected by the Dubins curve-straight-curve architecture, where the curves are generated to be clothoidal rather than circular arcs. This method presents basic sets of simple trajectories generated for a team of three UAVs wherein the paths intersect only at different points in time so as to avoid collision of the aircraft.

c. Pythagorean Hodograph

The Pythagorean Hodograph is a continuous curvature path generation method (10). These paths are constructed from polynomial functions similar to the B-spline curve, but with a minimum curvature constraint, as such a constraint is necessary to produce a flyable path. This method was first introduced in 1990 by Farouki and Sakkalis (40). Since then, the method has been successfully applied to UAV path planning, among other pursuits.

In 1997, Bruyninckx and Reynaerts (41) applied the Pythagorean Hodograph to path planning for mobile robots, since the computation for the curvature is low, the characteristic continuous curvature is desirable for path following of vehicles which cannot rapidly change direction, and the maximum curvature constraint ensures the vehicle will be able to successfully turn at the same rate as the generated path. This path planner uses two types of conditions to determine the desired trajectory. Either the boundary tangent vectors may be specified, or the boundary tangent directions and the boundary curvatures may be specified. Increasing the magnitudes of the boundary tangent vectors has the effect of increasing the length of the path as well as altering the curvature profile of the path. If the path curvature is specified, an iterative approximation method is used to determine the curvature of the path. There is no deterministic method for specifying the length of the path presented in this research. No results are presented outside of describing the method.

In the UAV path planning methodology presented by Lim and Bang (42), the quintic Pythagorean Hodograph is used in order to generate paths which satisfy a specified arrival orientation as well as a specified arrival location and time. The Pythagorean Hodograph is not a unique solution. Like other path planning methods, in 2 dimensions, 4 possible paths will be generates. Unlike other methods, the optimal path is chosen based upon the minimum bending energy, rather than on the shortest distance. The minimum bending energy is used to indicate which path will be the easiest for the aircraft to follow, which may well be more important than path length is many cases. The paths are generated based upon the given desired orientations and the remaining parameters are determined analytically.

The method proposed by Subchan et al. (43), applied the Pythagorean Hodograph path planning methodology to a swarm of UAVs for tracking airborne contaminants. The algorithm implemented is similar to the method proposed by Farouki and Sakkalis (40), and utilizes cubic, quartic, or quintic curves. Thus the remaining focus of the research rests not on the path planning approach but on the contaminant tracking approach. The first part of this approach is to model the boundary of the contaminant cloud, achieved using the Gaussian dispersion models. In general, the goal is for the UAVs to pass through the cloud measuring entry and exit points at specific moments in time, such that only one UAV is in the cloud at a time, but one UAV is always in the cloud at any given time.

d. Benefits and Drawbacks of Pose-Based Methods

Pose-based methods are highly useful due to their generality, which allows them to be applied to nearly any environmental situation. Additionally, these methods provide shorter paths, the shortest possible for Dubins, with guaranteed obstacle avoidance assuming proper pose selection. However, these methods require more complex calculations and are more difficult to implement than many of the other varieties. This complexity deters both practitioners and researchers and currently limits the development, investigation, evaluation, and practical implementation of these methods.

C. Optimal Search Methods

A wide variety of optimal search mechanisms exist which have been applied to the path planning problem. In general, the goal of each of these mechanisms is to find an optimal path through an environment based on some path rating criteria. Below are just a few of the more common optimal search techniques used in path planning.

a. Dijkstra's Algorithm and Extensions

One well-known optimal search technique is known as the A* algorithm. Due to its computational efficiency, this algorithm is often used for path planning, specifically for node-to-node, or discrete, path planning. This algorithm is fundamentally an extension of the commonly-applied Dijkstra's algorithm, in that

it uses the same approach but prefers a heuristic versus exhaustive search to improve computation time. In any case, while this is a widely used method, it is not a complete method, as it requires a graph of nodes and path segments from which to select the optimal path.

In 1994, Stentz (44) (45) applied the A* algorithm to path planning for a mobile robot in a partially-known environment, to produce an algorithm referred to as D* (for Dynamic A*). This method is, in essence, a more efficient approach to a brute-force path planning algorithm, since it must be capable of incorporating new sensor information as it becomes available and replanning the path accordingly. A graph of arcs is created through the environment as it is known, which are labeled with cost values according to a priori information. No specific path-segment/node generating configuration is specifically offered as the intention is to keep the approach general, as choice of the grid is arbitrary to the effect of the D* algorithm. Unlike the A* algorithm, the D* algorithm incorporates variable path segment costs, as the costs can increase or decrease as new sensor information becomes available. At each step of the algorithm, these costs are evaluated and the new best path direction is chosen based upon the information available at the current configuration. It should be noted that the path cost should account for path segments which sensor information reveal to be infeasible due to the presence of obstacles. This approach is compared to a brute force replanning algorithm. The results reveal that the D* algorithm and brute force algorithm perform equivalently, but the D* algorithm is increasingly more efficient as the complexity and size of the environment are increased.

A similar approach presented by Eppstein from 1994 (46) to 1997 (47) is that of finding a number of shortest paths on a directed graph. Conceptually, the approach recognizes the need to balance a short path with other parameters which are less easily summarized. Thus, the approach recommends choosing several shortest paths, depending upon the path cost parameters, then selecting the most desirable from these results. Like A* and D*, the algorithm is based on Dijkstra's selection. To reduce the complexity of the problem, repeated vertices are allowed. Additionally, the termination point is removed, such that all paths extending to any termination point in the graph are considered to be valid paths. The termination point is then chosen at a later point, and all paths extending to this point become candidates. In general, the algorithm is capable of finding the k-shortest paths with an order of magnitude fewer computations than comparable methods.

Along the lines of the previous methods, the A* Prune algorithm for finding k-shortest paths is presented by Liu and Ramakrishnan (48). In this method, the algorithm produces an ordered list of ascending path length of the first k-shortest paths between 2 nodes on a directed graph. The algorithm branches out producing paths which go toward the goal node, but eliminates any path which violates the imposed path constraints. This keeps only path routes which could eventually be viable shortest paths to the goal node. Paths are ordered according to length so that the shortest is expanded first. The algorithm terminates when

the required number of shortest paths have been achieved. In general, this method is more efficient than A* due to the addition of pruning. The algorithm computes at approximately the same time as the existing methods with which it was compared, but provides the k-shortest paths rather than only a single path.

b. Rapidly-Exploring Random Trees

Rapidly-exploring random trees (RRT) are an intuitive path planning technique. Sampling from the Monte Carlo and Voronoi techniques, random sample points are chosen and connected to the closest part of the existing tree. The space is quickly reduced, such that any future sample point will be within a certain distance of the existing tree. If obstacle avoidance is desired, a collision detection scheme can be implemented to reject sample points which fall within an obstacle, such that the resulting tree avoids any such pitfalls.

One common issue which arises with the RRT algorithm is the generation of a path through a smooth corridor. In general, the path produced in such a situation is jagged, difficult to follow for many types of vehicles, and inefficient at best. A solution to this problem is proposed by Rodriguez et al. (49). In this mechanism, the shape of the obstacles is used to shape the way the tree is grown in cluttered or narrow areas of the solution space. Results from this method reveal that if the switching logic is properly tuned, the method performs much better than other methods. However, tuning the corresponding gains is a non-trivial problem.

Rapidly-exploring random trees are applied to multi-UAV path planning by Kothari et al. (50) with consideration of obstacle- and collision-avoidance in a 2-dimensional partially-known and dynamic environment. A combination planner is created wherein the UAVs plan an initial path and, upon sensing an obstacle in the way, determine a new path with adequate time to maneuver away from the obstacle. The UAVs are simulated with limited communication ranges. Thus paths are communicated between UAVs as soon as they are in range of each other; whether or not the vehicles will collide is determined, and if so, the UAVs compute new paths which avoid the collision. As an additional precaution, a modified greedy RRT algorithm is implemented for finding a path in corridors between two obstacles. This produces both a safer and more efficient path. Overall, the simulation results showed the algorithm to have good performance and produce non-conflicting paths.

Another approach by Saunders et al. (51) utilized a switching logic to accommodate both static and dynamic obstacles. First, a rapidly-exploring random tree approach is used to quickly define an optimal trajectory through the environment of known obstacles. Then online obstacle detection is performed, and if an unknown obstacle is encountered, a geometry path modification is made to steer the UAV out of the path of said obstacle. Since this approach was applied to miniature UAVs with extremely limited payload capacity, scanning laser rangefinders were used to provide online obstacle detection and trigger switching to the

dynamic circumnavigation mechanism. Results of this mechanism showed that it was effective at avoiding collision with both known and unknown obstacles, though the method is not compared to any other method as a performance metric.

In an approach published by Rasmussen et al. (52), the problem of autonomous vehicle cooperation is tackled using a rapidly-exploring random tree for UAV vehicle task assignment. This algorithm considers task importance, task coordination, and flyability of the commanded trajectory, and generates piecewise optimal trajectories based on these task assignments. The approach suffers the drawback of having to generate a large portion of the total possible assignment scenarios in order to achieve reasonable confidence in the optimal solution, making it most applicable to low-dimensional problems. Unlike many similar approaches which assume only one task per target, the problem statement of this application is formulated with three tasks per target, such that different vehicles may perform the each of these tasks associated with the target, making the solution space considerably large. While this algorithm was shown to be effective at producing optimal solutions, heuristic methods may be preferred due to computation time.

One very interesting approach by LaValle and Kuffner (53) is to directly search for trajectories using the dynamics of the system, by searching the state-space rather than directly searching the solution space. This kinodynamic approach also allows the problem to be applied to a wider variety of problems, making it a useful technique for many highly complex systems. In general, this method removes many of the challenges present in the typical path planning approach such as ensuring trajectory feasibility. Thus in this technique, the controls, trajectory, and path are determined at one time. The solution space is formulated in terms of the system's state space. The rapidly-exploring random tree is used to expand the solution tree through the state space. Unfortunately, as is often the case when working in the state space, the resulting algorithm is highly complex. Although it maintains generality such that it can be broadly applied with few modifications, most research applications are limited in scope, and thus would prefer a simpler but less general solution.

c. Receding Horizon

Receding Horizon Control (RHC) is an optimal control technique in which the future trajectory is planned many steps into the future but only a small number of these steps are carried out prior to the trajectory being recalculated.

In an approach, by Kuwata et al. (54), RHC is used for UAV trajectory generation in the presence of unknown disturbances, with inclusion of flyability constraints, points of interest and no-fly-zones in the environment. In order to reduce the computational overhead of this optimization problem, such that trajectory may be calculated online, the algorithm relaxes the constraint that the target must be reached in the planning horizon. While the purpose of this is to permit a shorter planning horizon and thus reduce the

computational load, the secondary effect is that this algorithm takes on a greedy algorithm effect and it may not ultimately reach the intended target. Additionally, the initial trajectory must be planned offline (known a priori) because the computational requirement would be too great. This algorithm is demonstrated using a rotorcraft UAV, though it could be applied to others, to show that the algorithm is capable of providing a trajectory through an environment in the presence of disturbances.

In a similar approach by Schouwenaars, How, and Feron (55), paths are planned for multiple cooperative aircraft using a decentralized receding horizon algorithm based on mixed integer linear programming for computational efficiency. Each aircraft computes its own trajectory toward a waypoint, but these trajectories must be updated online with respect to the intentions of the other aircraft. Consequently, the receding horizon control technique is highly applicable. For added safety, at any point the aircraft must be able to reach a loiter circle, or a location where it can circle indefinitely without concern for obstacle collision or no-fly-zones. This is an interesting and unique mechanism as multiple aircraft teams typically have a leader-follower architecture, with one main computer planning the trajectories of all the vehicles.

d. Genetic Search Methods

Genetic algorithms (56) are highly useful, highly flexible optimal search techniques based upon Darwin's theory of evolution, which states that the strongest members of a population will ultimately thrive and produce more and better offspring than weaker members of the population. Thus, over time, the full population becomes stronger as better traits are honed and exploited and weaker traits die out. In general, a genetic algorithm solves for an optimal solution in a manner similar to this evolutionary process. The solution is encoded into a chromosome. A series of chromosomes are produced to create the initial population. The population undergoes genetic modification through genetic operators. These are usually mutation and crossover, though custom genetic operators geared toward a specific problem are commonly used as well. In order to simulate the environment, the population is rated based upon the performance of the solution which is contained in the chromosome. Finally, the population is reproduced to create the next generation, with better performing individuals having a better chance for reproduction. Often, elitist selection, or requiring that the best individual get at least one copy in the new population, is performed to eliminate the possibility of losing the best solution.

Genetic algorithms are often applied to the problem of path planning, since the goal is to produce not just a feasible path but an optimal one. These applications can take almost countless unique forms because the nature of the problem is so broad.

In a basic implementation of the genetic algorithm for path planning by Parry and Ordonez (57), the genetic algorithm is used offline to produce a path for goal-seeking and obstacle avoidance. However, this

method is intended to incorporate dynamic and unknown environments. Thus, a reactive obstacle avoidance methodology is implemented along with the path following scheme. In this trajectory generation algorithm, the path is encoded as a series of aircraft command states and their durations. For example, fly straight for 30 seconds then bank left at an angle of 30 degrees for 10 seconds. The performance index is a weighted sum of the path length, distance to goal, and a binary value for obstacle intersection. The intersection value is heavily weighted so that an invalid path is highly unlikely to survive repopulation. This algorithm incorporates mutation but no crossover, even though the chromosome representation would have readily allowed for crossover. Both offspring and parents are retained in the population for selection. The elitist selection strategy is combined with tournament selection to produce the new population. This algorithm showed poor results, as it often did not choose an optimal solution, and even occasionally produced paths with obstacle intersections. Response to unknown obstacles was simply not successful.

Similarly, a genetic algorithm path planning method presented by Rathbun et al. (58) attempts to accomplish online dynamic path planning, based on moving obstacles with unknown motion. The uncertainty level is the basis for this problem generalization. This genetic algorithm uses a similar structure to the previous approach, except that the path is encoded as a series of connected spline segments. Additionally, this method also makes use of crossover. It should be noted that due to the nature of the encoding of this problem, even a small mutation causes significant changes to the overall solution represented by the individual. In order to keep the algorithm online capable, the number of generations is limited to 20. Limited simulation results are presented, but those presented preformed favorably.

An application intended to track oceanic debris using UAVs researched by Rubio et al. (59), applies the genetic algorithm to find paths for both single and cooperative UAVs. The genetic algorithm methodology is favored because, compared to other methods, the genetic algorithm "is not as efficient in finding an optimal solution," however, "a viable solution is available at any time" (59). In this path planner, cooperative obstacle-avoidance is achieved while incorporating dynamic environment and weather conditions. To accomplish multiple-UAV path planning, each aircraft is assigned a portion of the path to fly in a given planning cycle. The method is shown to be appropriately capable of accomplishing the desired goals under simulated circumstances.

Zheng et al. (60) created a pseudo-3-dimensional genetic algorithm based path planner for dynamics environments, intended to generate paths geared toward terrain masking to hide the UAV from threats. Additionally, this algorithm aims to plan routes for multiple UAVs within this constraint. This algorithm is formulated in terms of waypoints without planning the full flyable trajectory, and is smoothed by inserting additional waypoints in the path. Two-dimensional simulation results are presented and this approach shows moderate success, though the solution is not complete.

In a 3-dimensional path planning approach presented by Nikolos et al. (61), a path is planned by the genetic algorithm through a 3-D rough environment. The path is represented by B-spline curves, where the chromosome of the algorithm represents the control points for the curves. This representation is chosen due to low computational requirements. Both terrain and moving obstacles are considered for the path planning, as well as the vehicle constraints for maximum altitude and minimum turn radius. An offline planner is used to generate the initial path in response to the environment as it is known at the time. Once the vehicle has begun following that trajectory, the online planner takes care of generating an updated path with respect to the new information. Once the online planner takes control, the future path is continuously redefined, as the planner is only designed to guide the vehicle to a safe location within the current sensor range. This is sensible, as beyond that, the likelihood of needing to replan anyway drastically increases, and only planning the path a short distance in the future lowers computational overhead. Several simulation results are provided. Only in one case does the planner fail, and the failure is caused by an unsmooth path rather than path-obstacle collision.

Genetic algorithms for path planning, like other applications, are often combined with other paradigms, where the paths are produced by one method and simply improved by the genetic algorithm. An example of such an application is presented by Benavides et al. (62), which combines the genetic algorithm with the Voronoi diagram path planner. In general, the environment is described by the Voronoi diagram and the evolutionary algorithm is tasked with choosing a valid, collision-free route through the environment. The initial population is computed using a Dijkstra's algorithm based on shortest path. The algorithm is demonstrated with physical and simulation results to perform adequately, and is compared to a potential field algorithm. It is unclear whether this implementation has made an improvement over direct use of Voronoi with Dijkstra's algorithm for selection using a more generalized cost function.

e. Benefits and Drawbacks of Optimal Search Methods

Optimal search methods take a wide variety of forms, and are extremely popular due to their immensely flexible nature. Such search techniques are noted for their ability to find a very good, if not optimal solution to a problem, which may be formulated in any number of ways. However, these methods are only as good as the problem formulation, which can lead to poor overall solutions. These methods are also notorious for high convergence times, due to the pseudo-brute force investigation methods required.

Chapter 3. ENVIRONMENT REPRESENTATION

The first and potentially most important step to path planning is to determine a scheme for properly and adequately representing the environment through which the UAV will be traveling. The environment representation scheme used in this research is discussed in this chapter. Section A discusses the earth-based reference frame notation. Section B defines the vector notation which will be followed throughout this document. Section C covers the risk zone representation scheme, and Section D explains the waypoint notation. Finally, Section E outlines the aircraft models used in this research and how their relevant parameters are calculated. These configurations will be used throughout this research, with additions where otherwise noted.

A. Reference Frames and Coordinate Systems

The Earth-based and aircraft-based reference frames are extremely important to the definition of an effective trajectory. For this application, we adopt a north-facing Earth Reference Frame. Thus, we select the initial location of the aircraft to be the origin of the Earth Coordinate Axes, though the location of the origin is relative and therefore somewhat arbitrary. For congruence with methods introduced later, a right-handed coordinate system is selected. Thus, the Earth X-axis points to the north, the Earth Y-axis points east, and the Earth Z-axis is positive in the downward direction. This is illustrated in Figure 3-1. Equivalently, for 2-dimensional planning methodologies, this can be reduced to a planar representation, as shown in Figure 3-2.

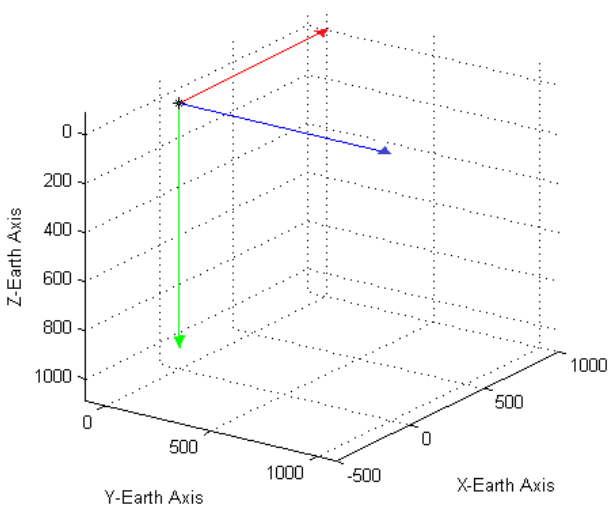

Figure 3-1—Three-Dimensional Earth Reference Frame

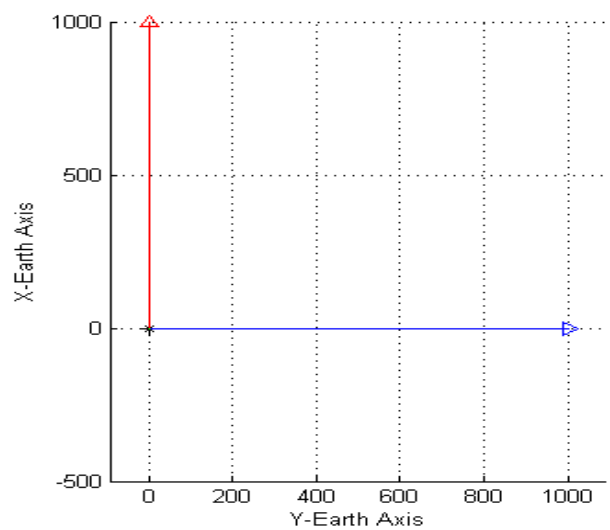

Figure 3-2—Two-Dimensional Earth Reference Frame

B. Notation

For this research, many of the methods make use of vector formulation. As such, it is necessary to define the notation which will be used to define a vector, as well as to define its coordinates with respect to the relevant coordinate system. The notation defined in this section will be used throughout this document.

A vector is characterized by a magnitude and direction within the solution space, irrespective of the frame of reference. Thus a vector will be notated with the vector arrow above the vector name, such as $\vec{p}$. Similarly, a unit vector, one with magnitude equal to 1 will be notated as $\hat{p}$. In many situations, it is convenient to formulate the problem and derive a solution with respect to vectors. However, computation of the path requires the use of the vector components with respect to a coordinate frame. Coordinates will be defined using the notation below, where the coordinates of vector $\vec{p}$ are defined with respect to coordinate system A.

$$[\vec{\mathrm{p}}]_{\mathrm{A}} = \begin{bmatrix} \mathrm{p_x} \\ \mathrm{p_y} \\ \mathrm{p_z} \end{bmatrix}_{\mathrm{A}} \qquad 1$$

Throughout calculation of a vector solution, it may be necessary to convert the coordinate of a vector from representation in one coordinate frame to another. To do this, a transition matrix is used, as shown in Equation 2. This equation illustrates the conversion of the coordinates of vector $\vec{p}$ from representation in coordinate frame A to coordinate frame B.

$$[\vec{\mathrm{p}}]_{\mathrm{B}} = \mathrm{R_{BA}}[\vec{\mathrm{p}}]_{\mathrm{A}} \qquad 2$$

Often in path planning, the use of position vectors is needed. In general, a position vector is described as the displacement of a point P from an arbitrary origin O. For purposes particular to this research, this origin O will be the origin of the Earth Reference Frame. Thus, the position vector $\vec{r}_{OP}$ of point P with respect to point O can be described as:

$$\vec{r}_{OP} = \overrightarrow{\mathrm{OP}} \qquad 3$$

with components:

$$[\vec{r}_{OP}]_{\mathrm{E}} = [\overrightarrow{\mathrm{OP}}]_{\mathrm{E}} = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{bmatrix}_{\mathrm{E}} \qquad 4$$

In order to define the normal vector to a plane containing two vectors, the cross product is needed. This is defined as:

$$\vec{a} \times \vec{b} = \begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = \tilde{a} \cdot \vec{b} \qquad 5$$

where $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ are orthogonal unit vectors for defining a coordinate system A, the a's represent the components of vector $\vec{a}$ with respect to coordinate system A, and the b's represent the components of vector $\vec{b}$ with respect to coordinate system A, and where $\tilde{a}$ is the tensor associated to vector $\vec{a}$. The components of the cross product in coordinate system A can be expressed as in Equation 6.

$$[\vec{a} \times \vec{b}]_A = \left[\begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}\right]_A = [\tilde{a}]_A \cdot [\vec{b}]_A = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}_A \cdot \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}_A \qquad 6$$

C. Risk Zone Representation

Representation of the areas of importance within the environment surrounding the UAV greatly affects the implementation and results obtained from the various path planning methodologies. Many typical path planning mechanisms focus only upon the presence of obstacles in the way of the aircraft. Others, such as the traditional Voronoi method, focus on finding the path of least exposure among radar towers. However, in most cases, the areas that a vehicle will traverse will contain not just obstacles, which absolutely must not be crossed, or radar, which has an infinite effective radius, but also less catastrophic threats such as civilian areas to which exposure should be minimized but do not inherently threaten the aircraft. For instance, it is undesirable to fly through a no-fly zone or radar, however, doing so will not necessarily produce unrecoverable catastrophic failure for the aircraft, in the same manner that flying through a building or a mountain would. For these reasons, areas of importance, which place a negative impact on the aircraft are represented in this research as risk zones with varying risk intensity.

The shape of a risk zone may be either spherical, represented by a 3-dimensional center and variable radius; or cylindrical, represented by a 3-dimensional center, with variable radius and height. These provide the flexibility to represent any object, obstacle or threat, encountered by the aircraft in a simple and general way. Similarly, if the solution is in two dimensions, these representations reduce to that of circles with a 2-dimensional center location and a variable radius. These can be seen in Figure 3-3 through Figure 3-5 below.

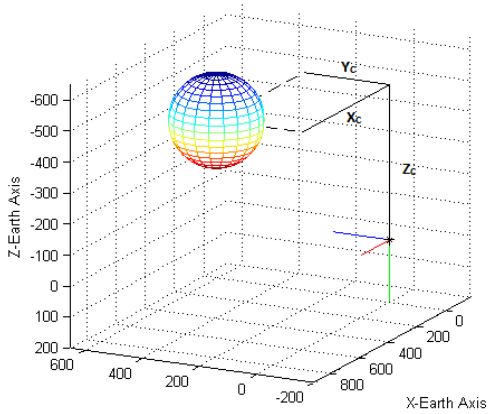

Figure 3-3—Spherical Risk Zone Geometry

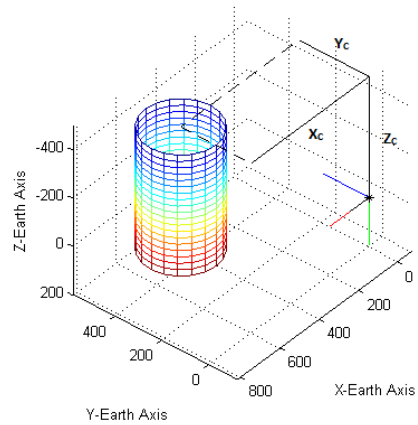

Figure 3-4—Cylindrical Risk Zone Geometry

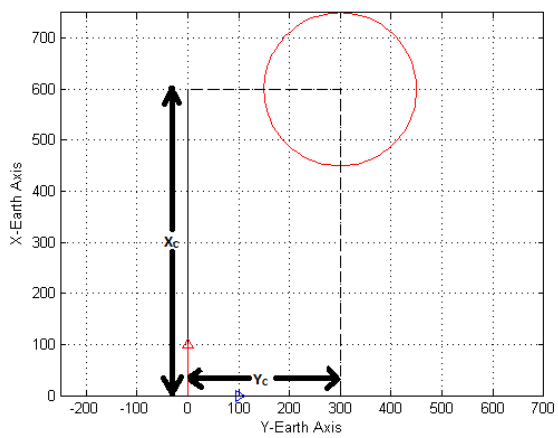

Figure 3-5—2-D Circular Risk Zone Geometry

The risk intensity, RI_i, of the risk zones is assigned based upon two parameters. These are the event probability, P_i, and the event severity, S_i, each ranging in value from 0 to 1. Event probability is the likelihood that an event will occur if the aircraft crosses the boundary. The event severity is the damage sustained by the vehicle, or rather the decrease in likelihood of the aircraft still being able to complete the mission, after an event is triggered. Three example scenarios can clarify these definitions. First, an anti-aircraft missile launcher may pose a low event probability, but a high event severity. It may be unlikely that the aircraft will be seen, but if seen, the UAV is likely to be shot down, leaving no possibility of completing the mission. Second, a no-fly zone may have a high event probability but a low event severity. The event that the aircraft enters the no-fly zone is certain to occur if the aircraft crosses the boundary of the area. However, this does not mean that the vehicle's ability to complete the mission will necessarily be impacted. Finally, a building or other solid obstacle can be defined as having an event probability and an event severity of 1, yielding a maximum risk intensity of 2. This means that if the aircraft crosses the boundary it is certain to

crash and be unable to complete its mission. Most importantly, representing risk zones in this way rather than just as obstacles allows for a higher level of path generation flexibility, with the resulting path chosen from the possibilities based upon the path cost, which depends upon the relative importance placed on path length and risk exposure. The risk intensity of a risk zone is calculated according to the following relationship:

$$\mathrm{RI_i} = (1 + \mathrm{S_i})\mathrm{P_i}\,, \qquad \text{where: } \mathrm{S_i} \in [0,1],\ \mathrm{P_i} \in [0,1],\ \mathrm{RI_i} \in [0,2] \tag{7}$$

A special case exists wherein radar towers of infinite radius can be represented, for encoding purposes, as a center with radius equal to 0. This becomes important for the discrete methods discussed in the next chapter. From this point on, the term *risk zones* will be used to refer to all risk zones of any risk intensity, but *obstacles* will refer only to risk zones of risk intensity $RI_i = 2$ which are considered to be uncompromisingly out-of-bounds.

D. Waypoint Representation

Waypoints, or poses, in this architecture are represented as a position and relevant heading. Additionally, poses signify the turning points of the path and thus need to be associated with the necessary curvatures. Thus, a 2-dimensional pose is defined by the X- and Y-position coordinates in the Earth Reference Frame, as well as the heading direction, ψ, and the maximum curvature, κ., as in Equation 8. This is illustrated below in Figure 3-6. The desired turn direction may also be specified with the pose, which is used for observation planning methods to specify which of the primary circles associated with the poses represents the area of interest for observation. This will be discussed in greater detail where relevant.

$$\mathrm{P} = \left[\mathrm{x}, \quad \mathrm{y}, \quad \frac{\pm 1}{\kappa}, \quad \psi\right] \tag{8}$$

Similarly, the 3-dimensional pose may be specified by the X-, Y-, and Z-position coordinates in the Earth Reference Frame. Not only is the heading, ψ, needed but also the pitch angle, θ. Additionally, paths are generated using the curvature constraint, κ. The full definition of the 3-dimensional pose is given in Equation 9 and illustrated in Figure 3-7.

$$\mathrm{P} = [\mathrm{x}, \quad \mathrm{y}, \quad \mathrm{z}, \quad \psi, \quad \theta, \quad \kappa] \tag{9}$$

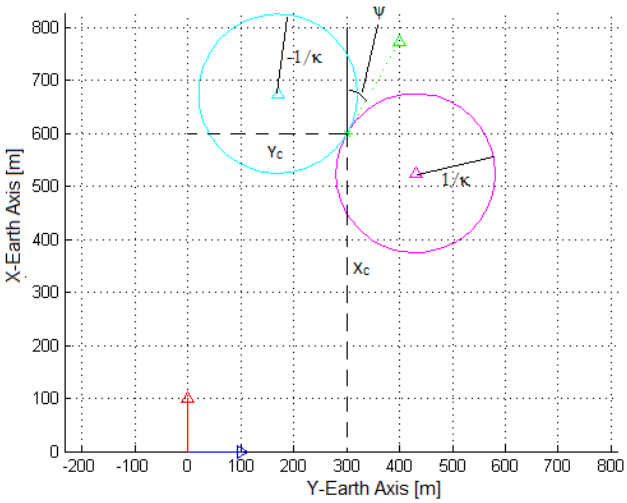

Figure 3-6—2-D Pose Definition

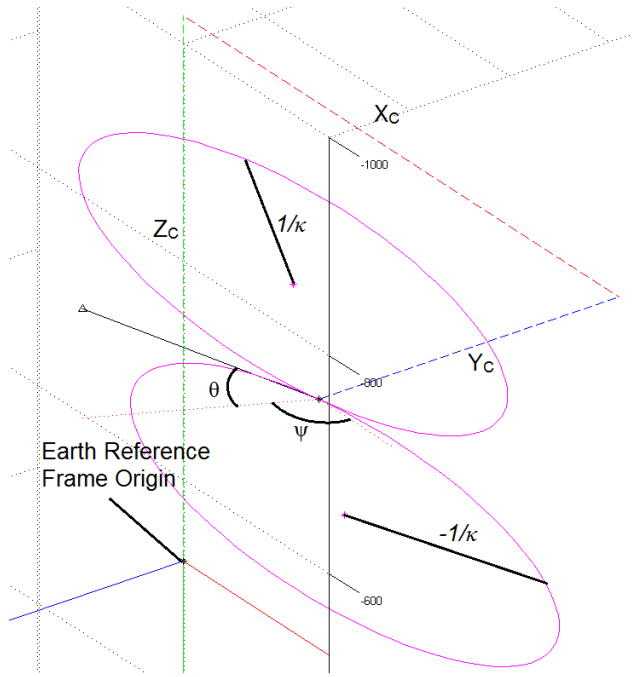

Figure 3-7—3-D Pose Definition

E. Aircraft and Dynamic Constraint Assumptions

For this research effort, several UAV aerodynamic models have been modeled (63) within the MATLAB/Simulink environment. These include the WVU YF-22 Research Aircraft (64) (65) (66) (67) (68), a scaled fighter jet; the NASA Generic Transport Model (GTM) (69), a scaled commercial passenger aircraft; and the Tigershark (70), the OX (71), and the Pioneer (72), which are low-speed high-endurance military UAVs intended for observation and payload delivery. Within this research effort, trajectory generation is performed using the assumption of constant aircraft velocity. As such, turning constraints are formulated in terms of the minimum turning radius, R_{min} and maximum climbing angle γ_{max} assuming cruise altitude. The maximum climbing angle was determined in simulation by determining the climb angle at which the aircraft speed drops below cruise speed while steadily climbing at maximum throttle. The resulting value was then moderately reduced to account for discrepancies. The path curvature constraint is calculated according to the following relationships, given in Equations 10 and 11.

$$R_{min} = \frac{v^2}{g * \tan(\phi_{max})} \qquad 10$$

$$\kappa_{max} = \frac{1}{R_{min}} \qquad 11$$

In the above equations, κ_{max} is the maximum curvature of the flyable path, v represents the cruise velocity of the UAV, g is the acceleration due to gravity, ϕ_{max} is the maximum banking angle capability of the aircraft,

and γ_{max} is the maximum climbing angle capability of the aircraft at the given altitude for constant velocity. Table 3-1 shows the relevant constraint parameters for each of the aircraft used in this research.

Table 3-1—Aircraft Dynamic Properties

	YF-22	GTM	Tigershark	OX	Pioneer
Cruise velocity, v	40m/s	38.6m/s	33.4m/s	21.1m/s	33.4m/s
Cruise altitude, a	304.8m	304.8m	300m	304.8m	300m
Maximum bank angle, ϕ_{max}	±45°	±30°	±30°	±30°	±30°
Minimum turning radius, R_{min}	114.2m	263.1m	197.0m	78.6m	197.0m
Maximum curvature, κ_{max}	0.00875/m	0.00380/m	0.00508/m	0.01272/m	0.00508/m
Steady-state climbing angle range at cruise velocity and altitude, γ_{max}	±35°	±30°	±25°	±25°	±25°

Chapter 4. ALGORITHM DEVELOPMENT WITH DISCRETE METHODS

Discrete, or road map, trajectory planning methods are simplistic in nature and are often applied to ground-based holonomic vehicles, due to a lack of turning radius constraints. However, these methods are also being applied to the area of UAV path planning due to their simple logic and expandability. Since there currently exist a seemingly endless array of interpretations of discrete path planning methods, a select few common methods were customized for implementation within the environment scheme outlined in the previous chapter. This chapter is devoted to providing the detailed implementation methodology for each of the applied path planning schemes. This chapter is divided into two segments. The first part will discuss the Potential Field Methods, which can generate the trajectory directly online or a priori; the second part discusses the various road map methods, which generate a path grid throughout the solution space, preceded by a discussion of the mechanisms used to select and smooth the final path. All algorithms are implemented through the Matlab®/Simulink environment, due to its flexibility and mathematical capabilities, and for ease of rapid prototyping. All calculation times are with respect to a Windows 7 desktop computer with a 2.2 GHz Core i7 processor and 8 GB of RAM.

A. Potential Field Methods

The methods presented in this section represent a large class of commonly-applied online trajectory generation algorithms which, for hardware implementations, use the vehicle in the loop to ensure that dynamic constraints are met. In the case of this research, the dynamic constraints are enforced within the simulation. This class of trajectory generation methods was included in this study due to their vast popularity. For these methods, the vehicle's current position, relative to threats and to the goal location, provide the inputs which drive the generation of the successive trajectory command. The popularity of such methods stems from the fact that they are inherently adaptive to dynamic and unknown environments, in addition to having traditionally low calculation overhead, also allowing for easy online implementation. However, such methods are highly susceptible to local minima, which can cause the aircraft to crash, collide with an obstacle, or lose track of the goal location.

a. Classical Potential Field

The basis of the classical potential field path planning algorithm is that the aircraft is treated as a point mass, while the goal acts as an attractive force and the threats act as repulsive forces, steering the vehicle through the environment. In general, these forcing functions can be defined as in Equation 12.

$$\vec{F}(q) = -\nabla U(q) \tag{12}$$

In the above equation, $\vec{F}(q)$ is the total force acting on the aircraft at configuration q, with $q \in \mathbb{R}^m$, and $U(q)$ is the potential function at configuration q. For this application, the potential field will be limited to 2 dimensions, thus $q \in \mathbb{R}^2 = \begin{bmatrix} x \\ y \end{bmatrix}$. This forcing function can be separated into the attractive and repulsive forces acting on the aircraft, as shown in Equations 13 and 14.

$$\vec{F}(q) = \overrightarrow{F_{att}}(q) + \overrightarrow{F_{rep}}(q) = -\nabla\left(U_{att}(q) - U_{rep}(q)\right) \tag{13}$$

$$\left[\vec{F}(q)\right]_E = \begin{bmatrix} F_x(q) \\ F_y(q) \end{bmatrix}_E = -\begin{bmatrix} \frac{\partial U_{att}(q)}{\partial x} \\ \frac{\partial U_{att}(q)}{\partial y} \end{bmatrix}_E + \begin{bmatrix} \frac{\partial U_{rep}(q)}{\partial x} \\ \frac{\partial U_{rep}(q)}{\partial y} \end{bmatrix}_E \tag{14}$$

These potential functions can take many forms depending on the application of the potential field planner. The key limitations are that the attractive potential should increase in magnitude as q moves away from q_{goal}, the goal location, and the repulsive potential should increase in magnitude as the separation between q and q_{obs} decreases.

Several common choices exist for the attractive potential function. The parabolic attractive potential function, given in Equation 15, meets the criterion, in that the force converges linearly as q approaches q_{goal}. In this equation, $\rho_{goal}(q) = \|q - q_{goal}\| = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2}$. The forcing function is derived in Equation 16 and its components are given in Equation 17. It should also be noted that this forcing function is unbounded as the distance between q and q_{goal} increases.

$$U_{att}(q) = \frac{1}{2}\xi\rho_{goal}^2(q) \tag{15}$$

$$\begin{aligned} \overrightarrow{F_{att}}(q) = -\nabla U_{att}(q) = -\nabla\left(\frac{1}{2}\xi\rho_{goal}^2(q)\right) = -\frac{1}{2}\xi\left(2\rho_{goal}(q)\right)\nabla\rho_{goal}(q) = \\ = -\frac{1}{2}\xi\left(2\rho_{goal}(q)\right)\left(\frac{q - q_{goal}}{\rho_{goal}(q)}\right) = -\xi(q - q_{goal}) \end{aligned} \tag{16}$$

$$\left[\overrightarrow{F_{att}}(q)\right]_E = \begin{bmatrix} F_{att_x}(q) \\ F_{att_y}(q) \end{bmatrix}_E = \begin{bmatrix} -\xi(x - x_{goal}) \\ -\xi(y - y_{goal}) \end{bmatrix}_E \tag{17}$$

Another common choice for the attractive potential function is the conic potential, defined in Equation 18. The forcing function associated with the conic potential field is derived in Equation 19, with components in Equation 20.

$$U_{att}(q) = \xi\rho_{goal}(q) \tag{18}$$

$$\overrightarrow{F_{att}}(q) = -\nabla U_{att}(q) = -\nabla\left(\xi\rho_{goal}(q)\right) = -\xi\frac{q - q_{goal}}{\|q - q_{goal}\|} \tag{19}$$

$$\left[\overrightarrow{F_{att}}(q)\right]_E = \begin{bmatrix} F_{att_x}(q) \\ F_{att_y}(q) \end{bmatrix}_E = \begin{bmatrix} -\dfrac{\xi(x - x_{goal})}{\|q - q_{goal}\|} \\ -\dfrac{\xi(y - y_{goal})}{\|q - q_{goal}\|} \end{bmatrix}_E \tag{20}$$

Unlike the parabolic forcing function, the conic forcing function is a scaled unit vector. Thus, this value is bounded for all $q \in \mathbb{R}^n$. However, the function is singular at the goal, which can be problematic. One way of combating this, as proposed by (73)and implemented here, is to combine these forcing functions using a switching logic based upon the distance of the current configuration from the goal, as defined in the relationship below.

$$\overrightarrow{F_{att}}(q) = \begin{cases} -\xi(q - q_{goal}), & \rho_{goal}(q) \le d \\ -\xi\dfrac{q - q_{goal}}{\|q - q_{goal}\|}, & \rho_{goal}(q) > d \end{cases} \tag{21}$$

The attractive potential composes only half of the potential field. The repulsive forces expressed by the risk zones also act on the aircraft to steer the aircraft away from the threats and prevent catastrophic collision. The repulsive potential function, given in Equation 22, acts on the vehicle any time the vehicle passes within the distance of influence of a threat, ρ_0. The repulsive forcing function is derived in Equation 23. Thus, the repulsive action of risk zone i is expressed in Equation 24. For this research effort, $\rho_{obs}(q) = \sqrt{(x - x_{obs})^2 + (y - y_{obs})^2} - R_{obs}$, since obstacles are defined to be circular.

$$U_{rep}(q) = \frac{1}{2}\eta RI_i\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 \tag{22}$$

$$\begin{aligned} \overrightarrow{F_{rep_i}}(q) = \nabla U_{rep}(q) &= \nabla\left(\frac{1}{2}\eta RI_i\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2\right) = \frac{1}{2}\eta RI_i\nabla\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 \\ &= \eta RI_i\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)\left(\frac{1}{\rho_{obs}(q)}\right)^2\left(\frac{q - q_{obs}}{\rho_{obs}(q)}\right) \end{aligned} \tag{23}$$

$$\overrightarrow{F_{rep_i}}(q) = \begin{cases} \eta RI_i\left(\dfrac{1}{\rho_{obs}(q)} - \dfrac{1}{\rho_0}\right)\left(\dfrac{1}{\rho_{obs}(q)}\right)^2\left(\dfrac{q - q_{obs}}{\rho_{obs}(q)}\right), & \rho_{obs}(q) \le \rho_0 \\ 0, & \rho_{obs}(q) > \rho_0 \end{cases} \tag{24}$$

Implementation and tuning of this potential field trajectory generation algorithm resulted in the performance shown in the figures below.

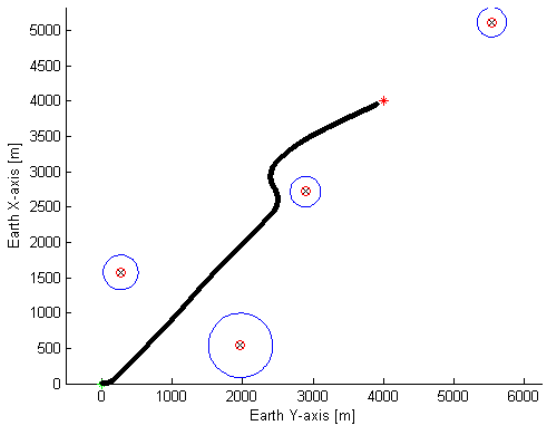

Figure 4-1—Basic Potential Field in a Simple Threat Configuration

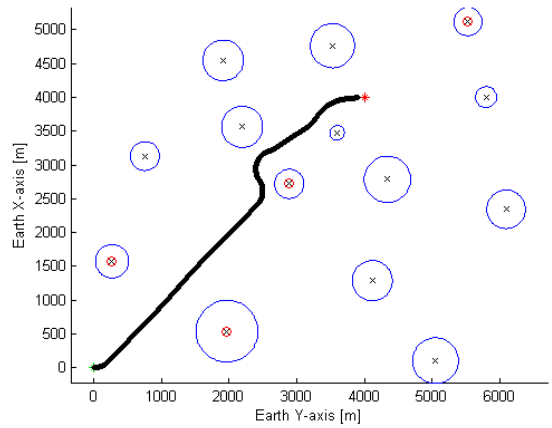

Figure 4-2—Basic Potential Field in a Complex Threat Configuration

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 4-3 and Figure 4-4. For scenario 1 the average calculation time was 3.99 seconds and the average calculation time for scenario 2 was 3.52 seconds.

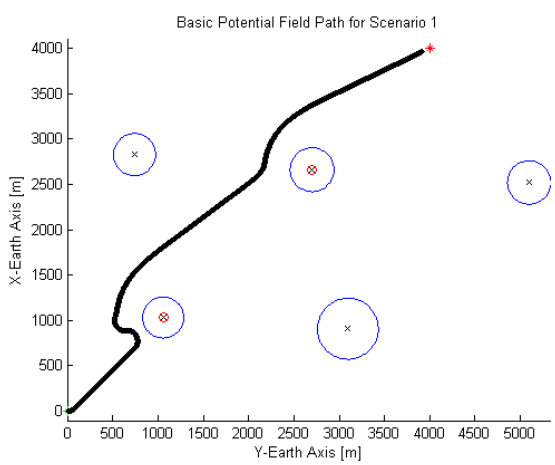

Figure 4-3—Basic Potential Field Path for Scenario 1

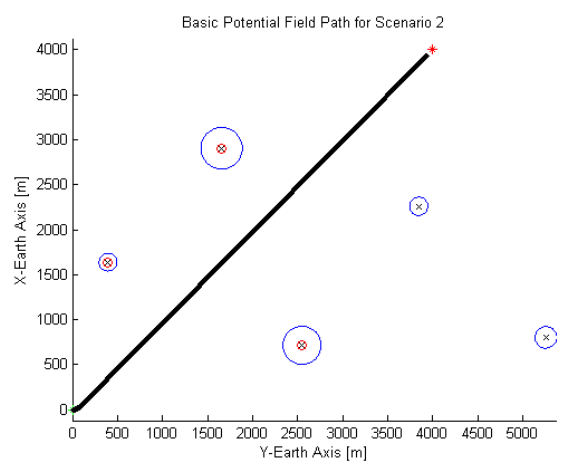

Figure 4-4— Basic Potential Field Path for Scenario 2

b. Improved Potential Field

One issue with the potential field is the presence of local minima. If the minimum potential of the field does not fall at the location of the goal, there is a distinct possibility that the vehicle could become stuck at the local minimum and never reach the goal, even if there is no obstacle between the vehicle's location and the goal. As proposed by Ge et al. (7), a simple modification to the above scheme can potentially improve this performance. As proposed, modifying the repulsive potential function to that given in Equation 25, can shift

the minimum potential location of the field to the location of the goal. The corresponding forcing function is derived in Equation 26. Equation 26 reduces to Equation 27 with the selection of n=2 for this implementation.

$$U_{rep}(q) = \frac{1}{2}\eta R I_i \left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 \rho_{goal}^n(q) \tag{25}$$

$$\begin{aligned}
\overrightarrow{F_{rep_1}}(q) = \nabla U_{rep}(q) &= \nabla\left(\frac{1}{2}\eta R I_i \left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 \rho_{goal}^n(q)\right) \\
&= -\frac{1}{2}\eta R I_i \nabla\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 \rho_{goal}^n(q) \\
&= \frac{1}{2}\eta R I_i \left[\rho_{goal}^n(q)\nabla\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 + \left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 \nabla\rho_{goal}^n(q)\right] \\
&= \frac{1}{2}\eta R I_i \left[\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 (n)\left(\rho_{goal}^{n-1}(q)\right)\left(\frac{q - q_{goal}}{\rho_{goal}(q)}\right)\right. \\
&\left. + \left(\rho_{goal}^n(q)\right)(2)\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)\left(\frac{1}{\rho_{obs}(q)}\right)^2 (-1)\left(\frac{q - q_{obs}}{\rho_{obs}(q)}\right)\right]
\end{aligned} \tag{26}$$

$$\begin{aligned}
\overrightarrow{F_{rep_1}}(q) = \eta R I_i \Bigg[&\left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)^2 (q - q_{goal}) \\
&- \left(\frac{1}{\rho_{obs}(q)} - \frac{1}{\rho_0}\right)\left(\frac{\rho_{goal}(q)}{\rho_{obs}(q)}\right)^2 \left(\frac{q - q_{obs}}{\rho_{obs}(q)}\right)\Bigg]
\end{aligned} \tag{27}$$

Implementation of this modification only requires replacing the previously-defined repulsive forcing function with the one given above. This method yields the results shown below. When comparing the relative performance of each repulsive potential function, it is seen that the enhanced function shown here exerts a much stronger response. Thus, it is necessary to separately tune the parameters of the enhanced potential field algorithm.

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 4-7 and Figure 4-8. For scenario 1 the average calculation time was 4.11 seconds and the average calculation time for scenario 2 was 3.52 seconds.

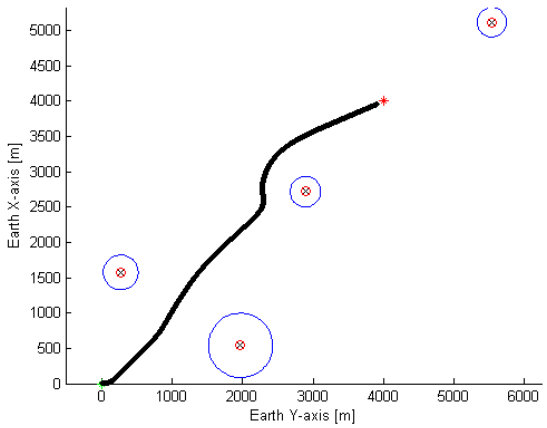

Figure 4-5—Enhanced Potential Field in a Simple Threat Configuration

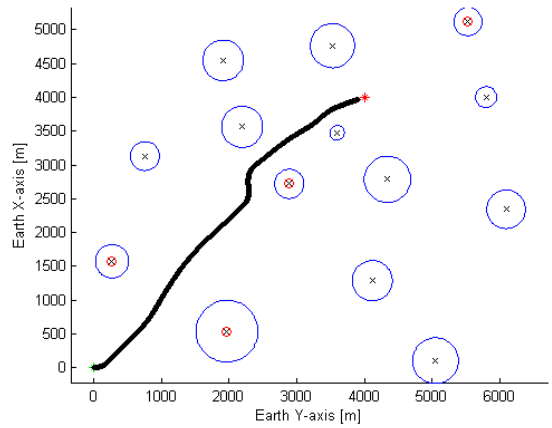

Figure 4-6—Enhanced Potential Field in a Complex Threat Configuration

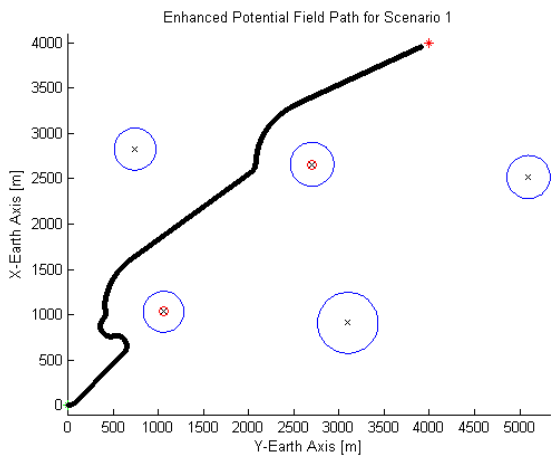

Figure 4-7—Enhanced Potential Field Path for Scenario 1

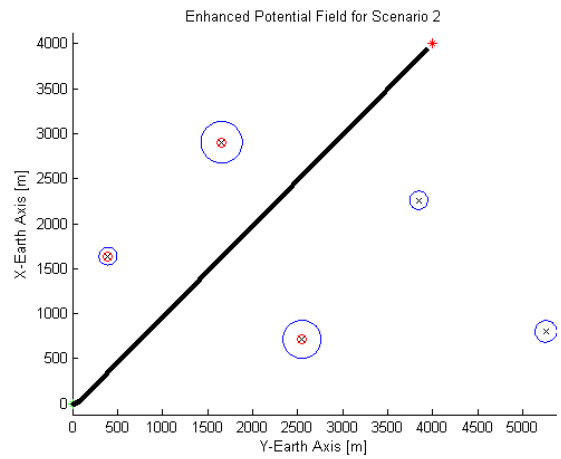

Figure 4-8— Enhanced Potential Field Path for Scenario 2

B. Road Map Generation Methods

The common feature among the discrete path planning methods is that each one produces not just one path through the environment, but a map of path segments, from which the optimal path must be chosen. In the following sections, the path selection operations will be discussed, followed by detailed implementation of the various methods for generating the road map of path segments.

a. Path Selection and Trajectory Generation

All road map methods require mechanisms for optimal path selection and subsequent path smoothing. These methods are discussed in the following sections.

a.1 Dijkstra's Algorithm for Optimal Path Selection

Each of the road map methods generates a map of nodes and paths through which the aircraft may travel. However, the selection of the path the aircraft will take relies on the use of a separate optimization algorithm. For all of these methods, a Dijkstra's algorithm (74) (75) (76) was implemented for path selection. Dijkstra's algorithm takes a predefined map of nodes and their connections, and the cost associated with each of these connections, and returns the minimum cost path from start node to goal node, knowing these start and goal node positions. The basis of Dijkstra's algorithm is that it iteratively finds the lowest cost path from the start to each node in the grid, successively branching out until the goal node is reached. Consequently, the lowest cost path from the start to the goal node is guaranteed. Dijkstra's algorithm was favored for this exploration over any of the previously documented extrapolations on this algorithm due to its relative simplicity and globally-accepted capability to efficiently solve the minimum-cost path problem.

At its core, the Dijkstra's algorithm is a path cost optimization method, which uses a grid of connected nodes and their associated costs to produce the path tree of lowest cost between a specified starting node and goal node. In order to carry out this algorithm, it is assumed that the path segments and their joining node have already been generated by an algorithm such as one of those discussed later. Additionally, a path cost function should have been applied to all paths connecting the nodes, so that each path segment is assigned a total path segment cost. This total cost of the path segment is composed of the weighted sum of the length cost and threat cost, given in Equation 28.

$$p_i = K\lambda_i + \frac{1-K}{2}\sum_{j=1}^{m}\sigma_{ij} \qquad 28$$

where the length cost is calculated in the equation below, and *m* is the number of known threats present:

$$\lambda_i = \frac{\sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}}{\sqrt{\left(x_{goal} - x_{start}\right)^2 + \left(y_{goal} - y_{start}\right)^2 + \left(z_{goal} - z_{start}\right)^2}}, \qquad 29$$

$$\lambda \in [0, \infty] \text{ (usually } \lambda \in [0, 1])$$

The threat cost imposed on a segment varies, depending upon whether the risk zone is acting as a radar or as a threat. In the case that the risk zone is a radar, having infinite boundary, the threat due to the zone on the path segment is calculated according to Equation 30.

$$\sigma_{ij} = \frac{RI_j}{1 - d_{ij}^4} \qquad 30$$

In this equation, RI_j represents the risk intensity of zone *j* and d_{ij} is the shortest distance from the radar location to the path segment.

Otherwise, the path will only be affected by the zones whose boundary contains all or part of the segment. In which case, the threat cost associated of the segment for each zone is based upon the relative amount of the path which passes through the zone and is calculated as in Equation 31.

$$\sigma_{ij} = RI_j \frac{c_{ij}}{L_i} \qquad 31$$

where c_{ij} represents the length of the portion of the path segment contained in the zone and L_i is the length of the path segment.

In performing the Dijkstra's path selection, the algorithm first initializes all nodes as unvisited and with best cost to reach it at a high value (infinity). The best cost to reach the start node is, of course, zero, since there are no steps taken to reach it. The algorithm then iterates through all unvisited nodes, and chooses the one with the lowest cost to reach. This node is now considered visited and never needs calculation again. Then all nodes connecting to this node are found and if their best cost to reach is currently higher than the cost to reach it via the path to the chosen node, this value is updated. This process continues until a full path, which will be of minimum cost, to the goal node is found. In order to better describe this process, see Pseudo-Code 1 in Appendix A. The total path cost is calculated as the sum of the cost of the q path segments contained in the path, given in Equation 32.

$$P = \sum_{k=1}^{q} p_k \qquad 32$$

a.2 Smoothing

In order for a road map based path planner to produce a flyable path, the selected series of path segments must not contain sharp corners. Thus a smoothing algorithm is used. Additionally, the smoothed path must observe a maximum curvature, or minimum turning radius, as specified by the aircraft limitations. As discussed in Chapter 2. there are several commonly applied algorithms used to smooth the path. Among the most common of these are Bezier curves, virtual-mass-and-spring, and filleting. Also as previously discussed, smoothing using either the Bezier curve or the virtual-mass-and-spring involves iteration in order to ensure that the curvature constraint is met. Since these methods are intended for use online, this additional computational load in not desirable. Additionally, these can produce final smooth paths which diverge greatly from the initial path segments used to generate them. This not only can alter which path through the system is optimal, thus removing the reliability of Dijkstra's, but can also introduce obstacle intersection and unnecessary threat exposure which was not a factor of the initial path. For all of these reasons, the filleting method will be used to create a smooth and flyable path for all of the road map methods implemented in this

research. Incidentally, this smoothing method will also result in the shortest smooth path meeting the curvature constraint (29).

In order to fillet the path segments to produce a flyable path, it must first be confirmed that the path segments are each long enough to accommodate the fillet. This distance required to allow filleting is calculated according to the following equation.

$$d = R \tan\left(\frac{\pi - \delta}{2}\right) \qquad 33$$

where R is the minimum turning radius for the aircraft and δ is the angle between the path segments. In the event that this distance is greater than the length of either path segment, no fillet may be produced. The solution to this problem will be discussed in the next section.

Assuming that the path segment lengths are sufficient to support filleting, the fillet is produced by specifying the new endpoints of the path segment. These new endpoints coupled with the arc sweep angle θ can be used to produce the necessary points for a flyable trajectory. An example of the resulting fillet geometry is shown in Figure 4-9.

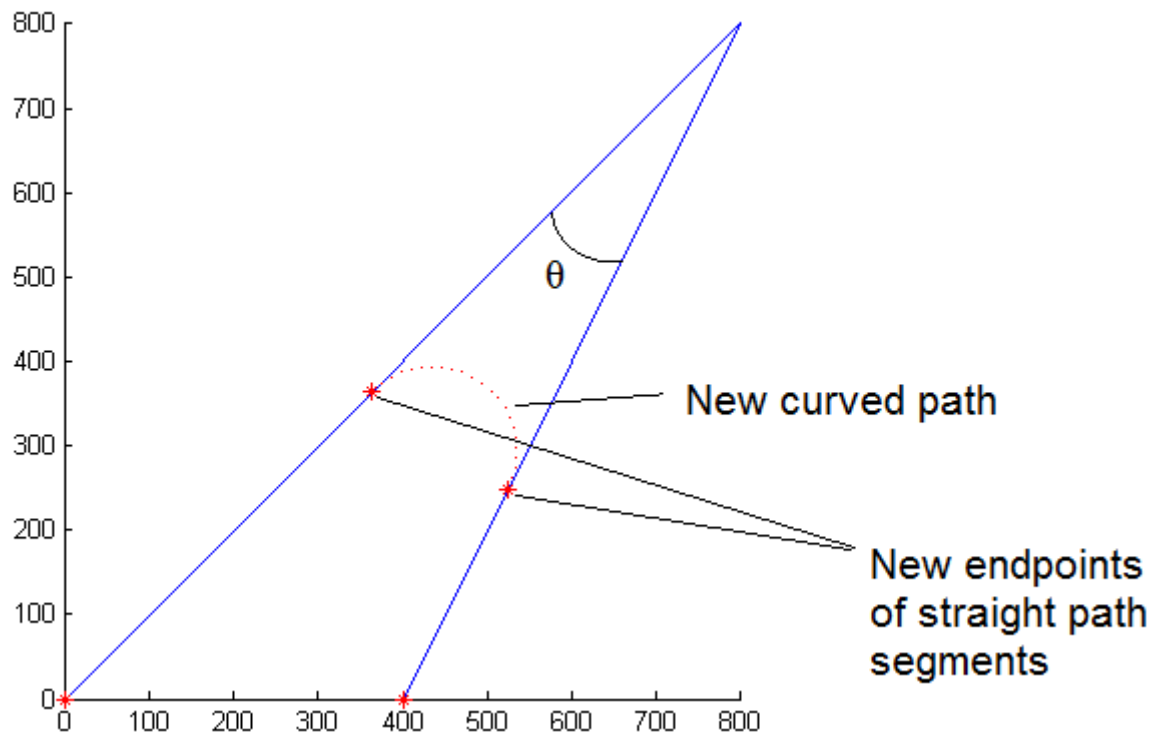

Figure 4-9—Filleted Path Segment Junction

a.3 Dijkstra's Algorithm to Ensure a Flyable Path

As discussed in the previous section, occasions arise in which the path segments in the optimal path are not sufficiently long to allow filleting of adequate turning radius for the specified aircraft. This means that not enough room exists between maneuvers to accurately follow the path. This can result in missed targets, unwanted threat exposure, and even obstacle collision. A major problem with many path planners applied to UAVs lies in the production of relatively short path segments which result in this "non-filletable" path situation. Even though a successful set of path segments is generated, it may not be feasible to fillet the path, depending upon both the path segments chosen and the angle between these. Since it is infeasible to check this during the generation of the grid, and since it is too late to check this once the optimal path has been calculated, it is reasonable that this should be accounted for during the selection of the optimal path.

In order to do this, at the point in the Dijkstra's algorithm when the next path is chosen to connect to a node, the combination of the previous path segment and the possible next path segment are checked to verify that this joint will allow filleting. If they do not, the possible next segment is rejected and the cost is set to be infinitely large; then the next lowest cost path is chosen to be the next path and the process is repeated. If none of the paths connecting to a node allow a filletable path, the node is declared to be a dead end, and the node will not ultimately be part of the chosen optimal path.

b. Regularly Spaced Grid

A grid-based path planner is a commonly-applied planning mechanism due to its simplicity and generality. For this methodology, the solution space is defined using the full range of all risk zones, plus starting and ending locations, plus a buffer around the edge. This solution space is segregated evenly into a grid of nodes with grid spacing equal to 2R, where R is the minimum turning radius of the aircraft. These are shown in Figure 4-10. Once the grid is generated, nodes which fall within the interior of obstacles are removed, as shown in Figure 4-11. The nodes are connected to each other using orthogonal and diagonal path segments, as seen in Figure 4-12 below. Remember that not all risk zones are considered to be obstacles and therefore uncrossable. Additionally, any path segments, regardless of endpoints, which intersect an obstacle are also removed from the road map. Finally, this results in a grid such as the one seen in Figure 4-13.

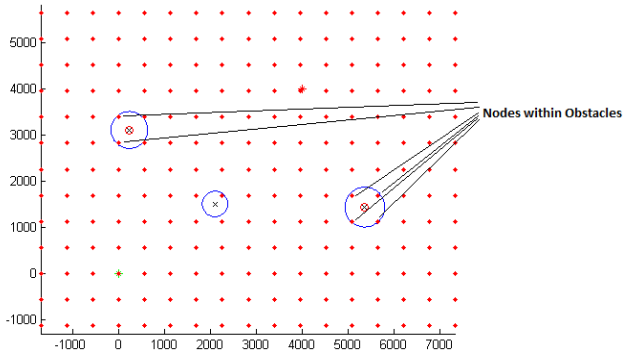

Figure 4-10—Raw Grid Nodes

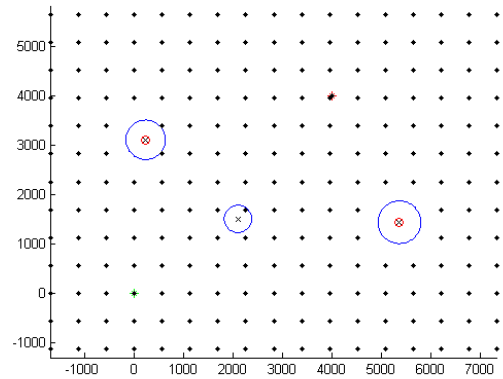

Figure 4-11—Obstacle Nodes Removed

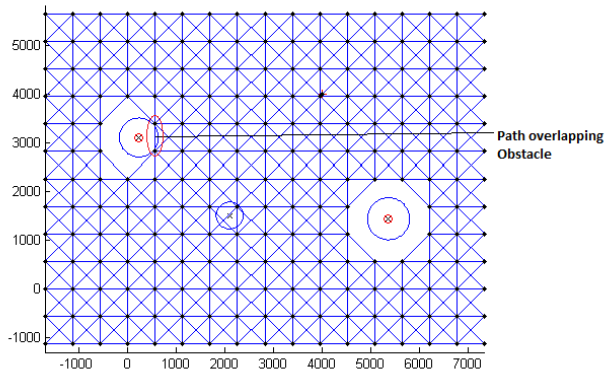

Figure 4-12—Path Based on Nodes

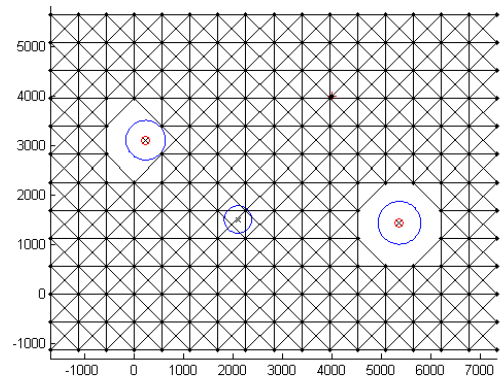

Figure 4-13—Final Paths Grid

Since this planner is intended to accommodate starting and finishing heading angles, the start and goal nodes must be connected to the grid in a special way, referred to as the start and goal sequences, respectively. This begins by connecting the terminal nodes to their nearest surrounding nodes. In order to allow adequate room for maneuvering to achieve the desired trajectory angle, nodes must lie at least 3R from the terminal node. Additionally, each of these connections is checked for obstacle intersection prior to acceptance. An example of these selections in presented in Figure 4-14.

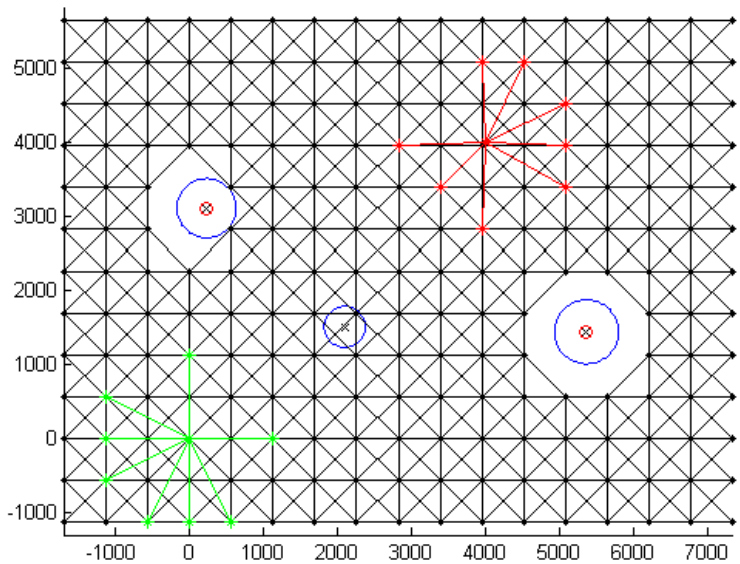

Figure 4-14—Terminal Node Connections

Once all connections are made, the optimal path is chosen using a Dijkstra's algorithm based on the path cost of each segment. This produces the "node path" for the vehicle, as seen in Figure 4-15, which still requires smoothing. The final smoothed path may be seen in Figure 4-16.

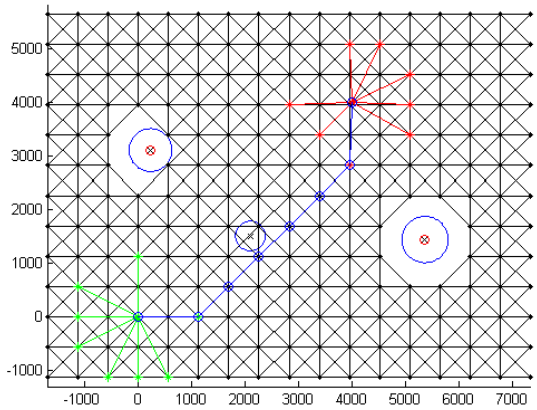

Figure 4-15—Unsmoothed Node Path

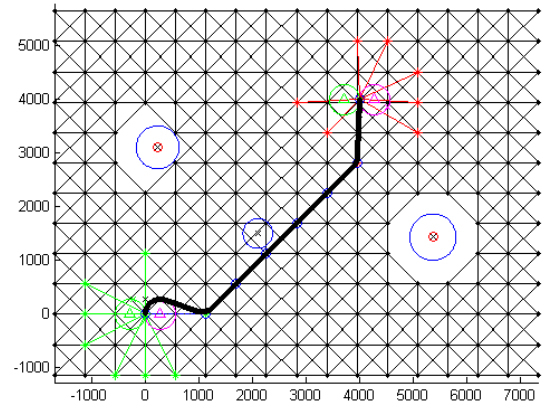

Figure 4-16—Final Smooth Path

Before the path can be smoothed, the heading angles must be accommodated. First the start and goal sequences are calculated. Working from the terminal node to the grid, the heading circles are generated tangent to the desired heading. For any heading, there exist two tangent circles which can satisfy it. Thus, the decision between these is made based upon which circle may be traversed in the correct direction to approach

or leave the terminal point. The tangent line from the circle to the grid node is then generated and becomes the path. This process is illustrated by the series of figures below.

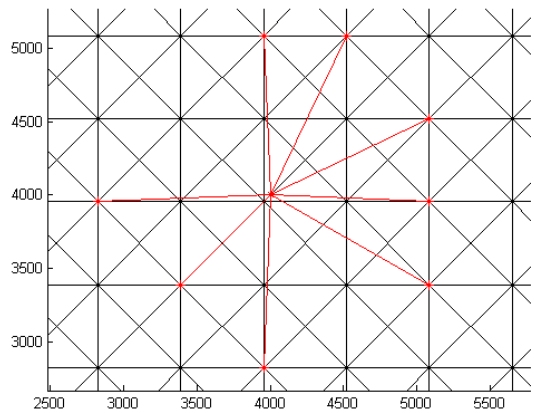

Figure 4-17—Raw Terminal Point Connection

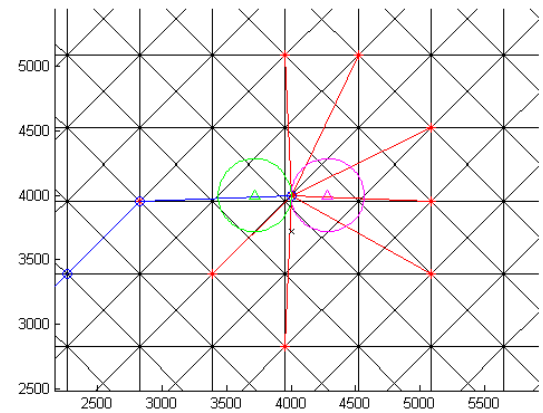

Figure 4-18—Terminal Point with Heading Circles

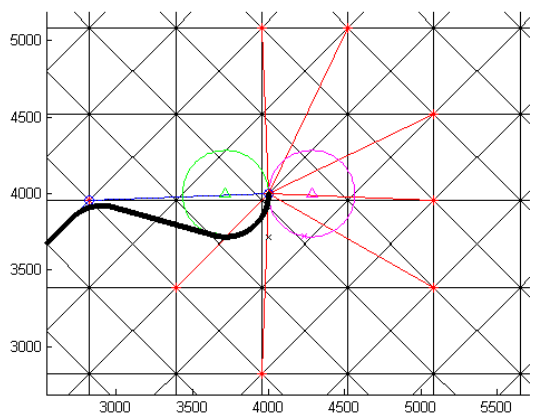

Figure 4-19—Terminal Point Connected to Grid via Heading Circle Tangent Line

At this point, the points connecting the start and goal sequences to the grid become the new effective ending nodes, and the intermediate path is smoothed to a flyable trajectory using filleting.

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 4-20 and Figure 4-21. For scenario 1 the average calculation time was 10.40 seconds and the average calculation time for scenario 2 was 10.91 seconds.

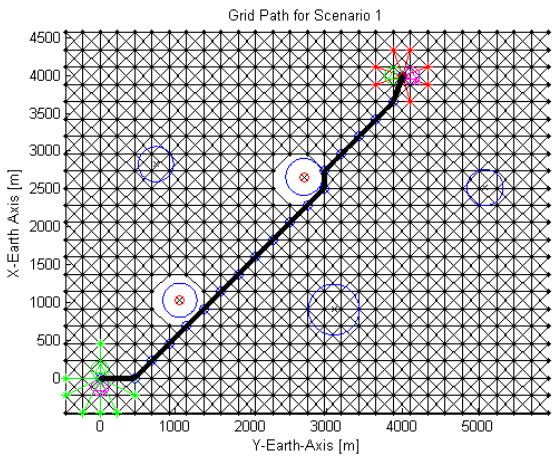

Figure 4-20—Grid Path for Scenario 1

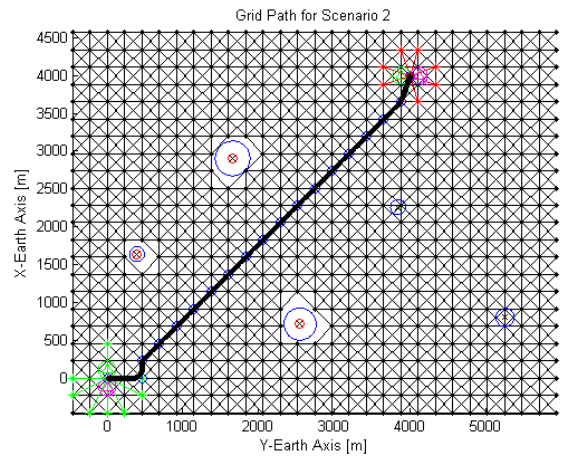

Figure 4-21— Grid Path for Scenario 2

c. Cell Decomposition

Similar to the regularly spaced grid formulation for discretizing the solution space, the cell decomposition algorithm also uses a discrete grid-like approach to partition the solution space into cells which are considered either free or occupied. Cell decomposition can be considered either exact, in which the occupied cells take exactly the space of the obstacles they represent, or approximate, in which a generalized shape is used to construct the cells instead. Due to its lower computational overhead and decreased complexity, approximate cell decomposition is usually the favored approach. One of the most commonly applied approximate cell decomposition methods is known as the 2^m-tree decomposition. In this approach, the bounds of the solution space are generalized as a rectangloid of dimension m. If the cell is not found to be purely free or occupied, the cell is split evenly into 2^m smaller cells. These cells are again checked for belonging entirely to free or occupied space, and the process continues until the cell size threshold is met. Here, a 2-dimensional, thus m=2, cell decomposition planner was implemented. As with the grid-based planner, the cell decomposition planner also accomplishes generation of paths which observe the desired start and goal heading, using the same approach as discussed in the previous section. The full pseudo-code for this algorithm is given in Pseudo-Code 2 in Appendix A.

For the cell decomposition methods, the vehicle may travel either along the edges of the cells, or through the interior of the cells belonging to free space. For consistency with the other discrete methods implemented in this study, the edges of the cells are treated as paths, and the optimal path is chosen via the Dijkstra's selection algorithm, discussed later. Below are some example paths generated using this planning method. Figure 4-22 contains an example path through a field containing purely obstacles. Figure 4-23 contains a path generated using a field of variable risk intensity risk zones.

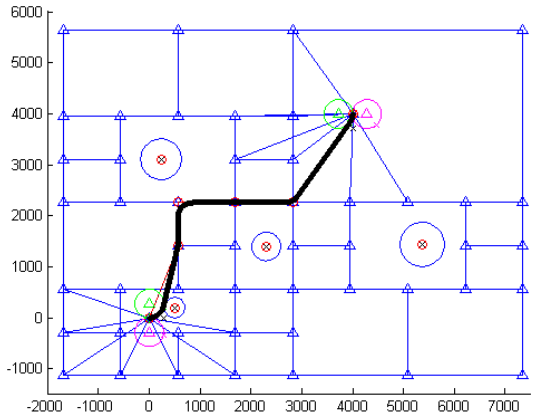

Figure 4-22—Cell Decomposition Path Using a Purely Obstacle Field

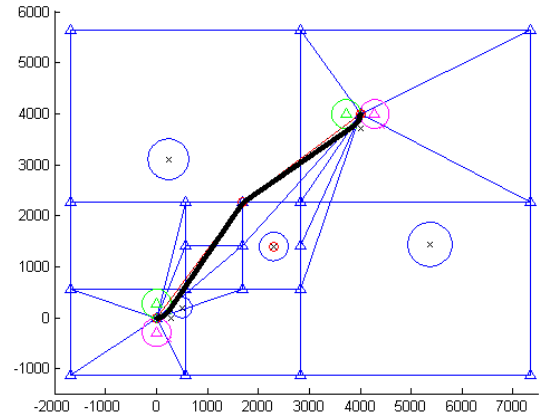

Figure 4-23—Cell Decomposition Path Using the Risk zone Approach

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 4-24 and Figure 4-25. For scenario 1 the average calculation time was 0.6289 seconds and the average calculation time for scenario 2 was 0.7243 seconds.

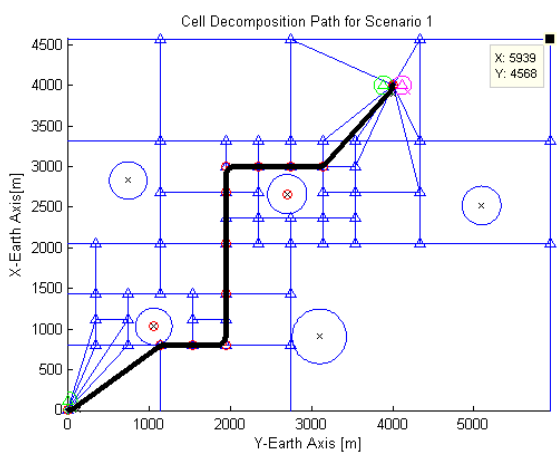

Figure 4-24—Cell Decomposition Path for Scenario 1

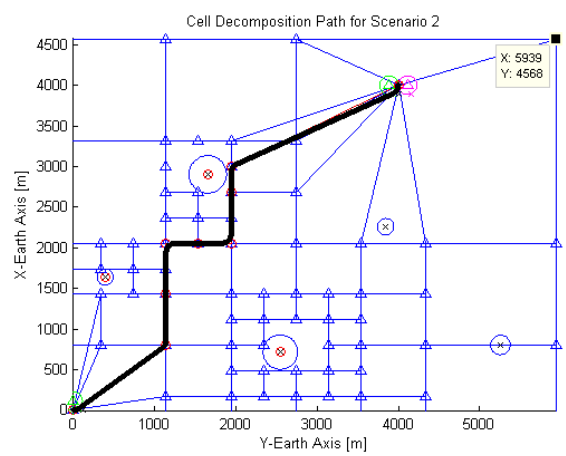

Figure 4-25— Cell Decomposition Path for Scenario 2

d. Classic Voronoi

Due to the wealth of information and applications of the Voronoi diagram to UAV path planning, a classic Voronoi algorithm was implemented. This algorithm naturally accommodates radar locations of infinite radius. A classic Voronoi diagram is a geometric mapping of the locations in the solution space which are equidistant from the nearest input points. An example Voronoi diagram is shown below in Figure 4-26. It can be seen that for the 2-dimensional case, this set of equidistant locations form straight lines between two input points. Locations where these lines join are equidistant to 3 input points. These connections are

typically referred to as Voronoi nodes. Lines which diverge are said to be connected to a node located an infinite distance away. Each input point results in a Voronoi cell, which is the closed shape produced by the equidistance lines surrounding the point. All points within this cell are therefore closer to this input point than to any other input point considered in the diagram. Equidistant lines which go to infinity are considered to be connected at the same infinite node, and thus Voronoi cells containing these are also considered to be closed shapes.

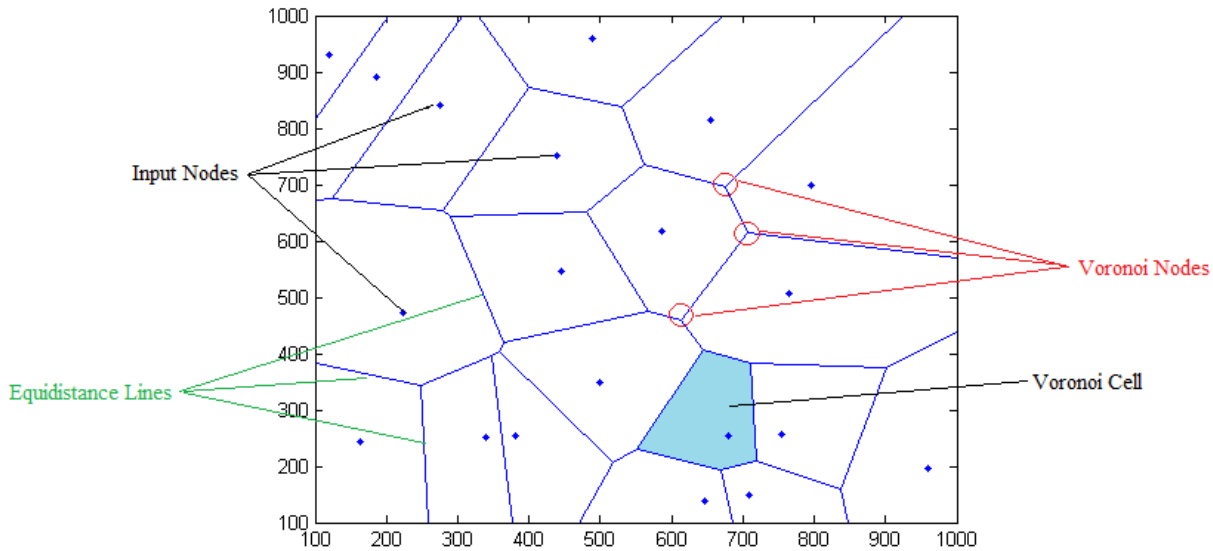

Figure 4-26—General Voronoi Diagram

It is logical to apply such a mathematical formulation to the area of path planning. In the event that the locations we wish to minimize exposure to can be represented as points, such as for the case of radar which has an origin but no definite boundary, we may generate a grid of potential paths which expose the aircraft to these points in the least possible way. Figure 4-27 illustrates the effectiveness of the basic Voronoi planner to find a suitable trajectory through an environment considering only radar as threats. Figure 4-28 illustrates the insufficiency of this method if variable-radius risk zones are considered.

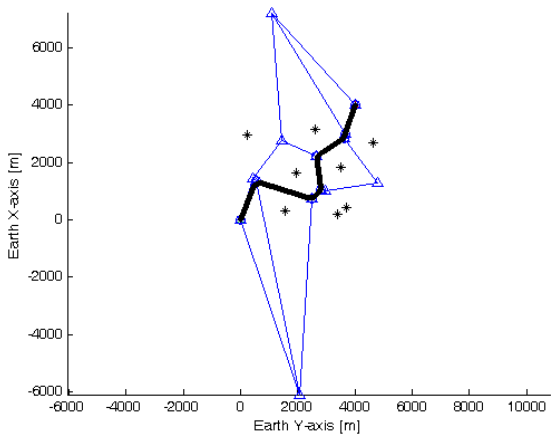

Figure 4-27—Voronoi Path with Only Radar

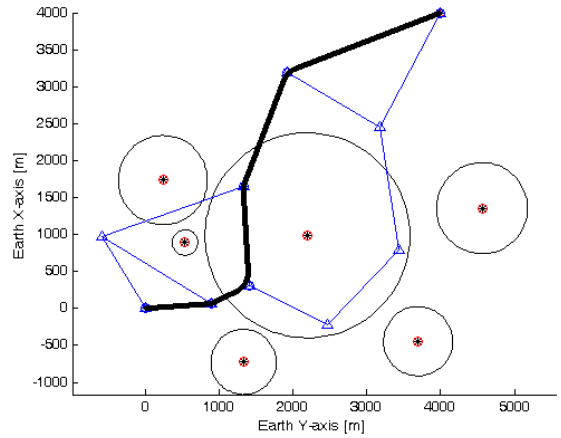

Figure 4-28—Voronoi Path with Obstacles

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 4-29 and Figure 4-30. For scenario 1 the average calculation time was 0.1197 seconds and the average calculation time for scenario 2 was 0.1211 seconds. Note that for these scenarios, short path segments would make a flyable path impossible. Thus, a filleting mechanism which "skips ahead" until filleting is possible was used. This mechanism is neither recommended nor preferred as it can introduce overlapping with obstacles that was not present in the path as it was defined. This tendency and necessity is eliminated using the Obstacle Avoidance Voronoi, developed in a future subsection, which accounts for both obstacle size in path generation and path feasibility in path selection.

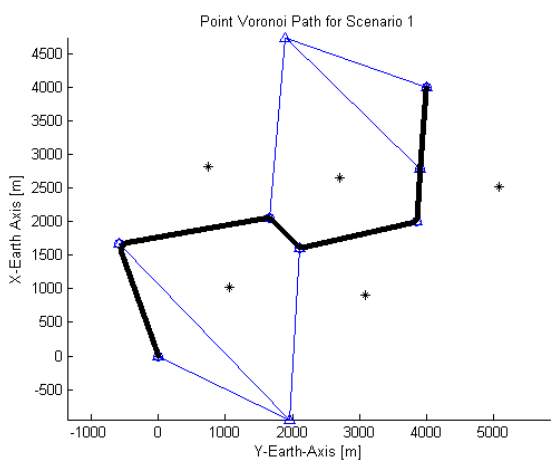

Figure 4-29— Point Voronoi Path for Scenario 1

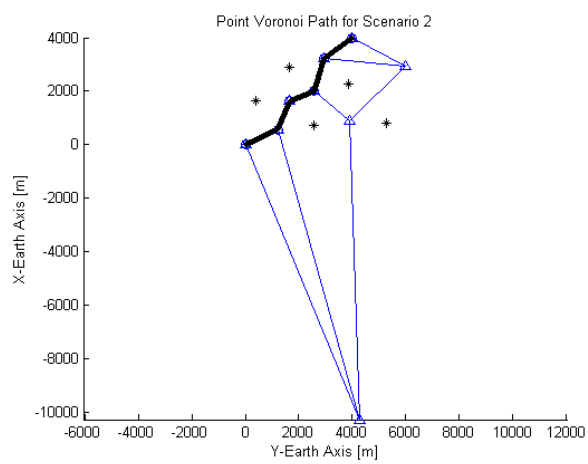

Figure 4-30— Point Voronoi Path for Scenario 2

e. Polygon Voronoi

In the area of UAVs more often, obstacles will be present in the field of the aircraft, introducing locations which absolutely must be avoided. One common method of applying the Voronoi diagram to the

problem of obstacle avoidance is to formulate obstacles as polygons with definite vertices. An example of such is shown in Figure 4-31. These obstacles are represented as hexagons, though any other polygons could be applied with the same resulting characteristics, regardless of the number of vertices, whether the shape is regular or irregular, convex or concave.

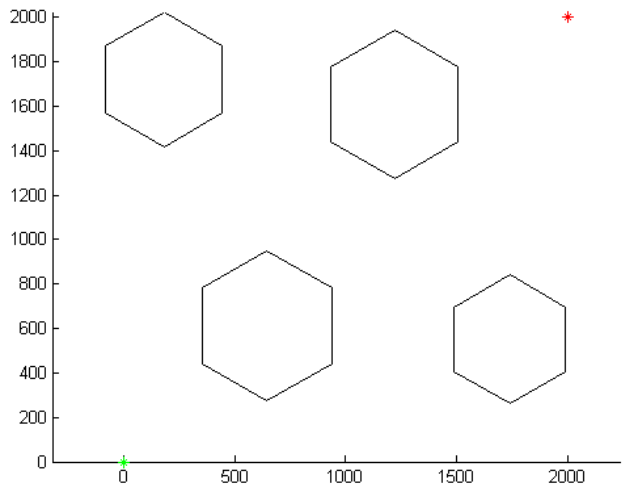

Figure 4-31—Example of Hexagonal Polygon Obstacles

In order to produce the Voronoi diagram for the polygon obstacle avoidance scheme, the input points become the vertices of the obstacle polygons. This produces the full Voronoi grid as shown in Figure 4-32. It is clear that many of these paths will need to be removed due to overlapping with the obstacles. The Voronoi grid is shown again in Figure 4-33, but with the path segments highlighted which will remain after all invalid paths are removed. Figure 4-34 illustrates the finalized grid after the start and goal positions have been connected.

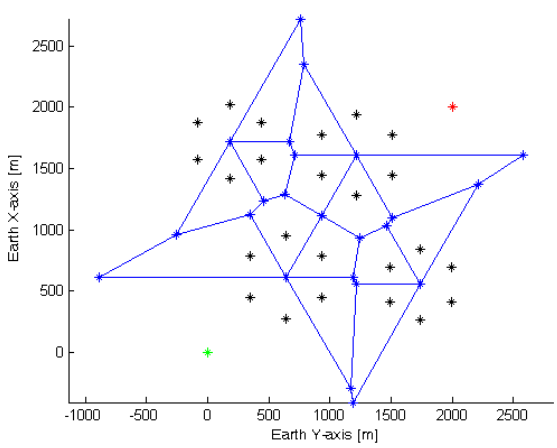

Figure 4-32—Full Example Voronoi Grid Using Polygon Obstacle Representation

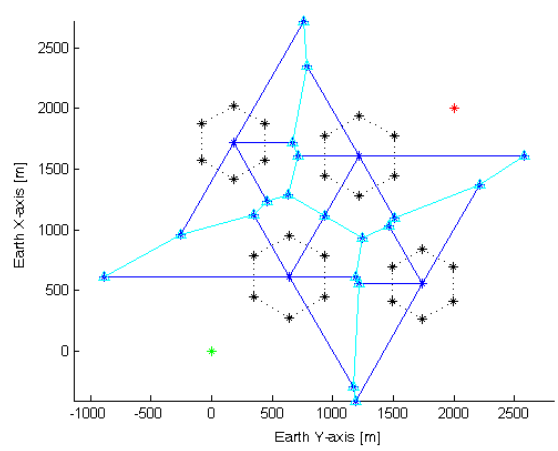

Figure 4-33—Voronoi Grid Using Polygon Obstacle Representation With Valid Paths Highlighted

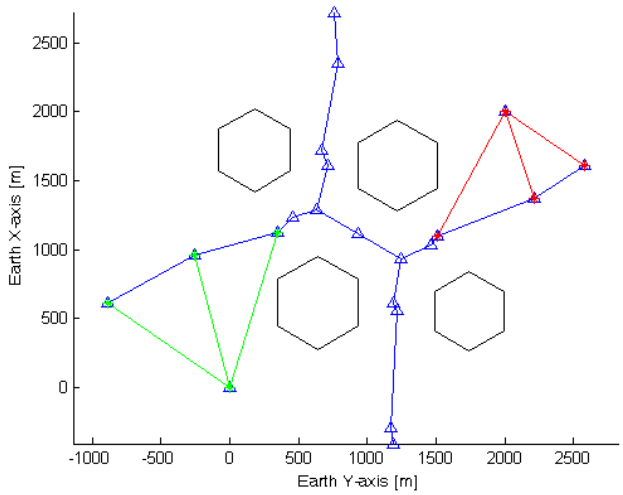

Figure 4-34—Finalized Grid Produced Using the Voronoi Diagram and Polygon Obstacle Definitions

Representing obstacles as polygons presents several problems. First, since the vertices are used to produce the Voronoi diagram, a secondary process is needed to remove the equidistance lines which cross the boundaries of the obstacles and any of the Voronoi nodes which are rendered unnecessary by this path removal. This severely limits the solution space available for the aircraft. In fact, this process makes it highly likely that the set of remaining path choices will not contain a full, flyable path. This is to say that the path is highly likely to contain short segments which will not allow room to curve the path to produce a trajectory which the aircraft is capable of following. Even in the above example, several short segments are present for which only an aircraft with a very small turning radius would be able to find a flyable trajectory. Additionally, joining the start position and goal position to the resulting Voronoi path grid becomes non-trivial, as there is no simple way of ensuring that a joining link, after heading and path curving are considered, will not intersect an obstacle. Figure 4-35 and Figure 4-36 below illustrate examples of the polygon Voronoi algorithm. For the very simple case shown in Figure 4-35, the algorithm is capable of finding a flyable path. However, for the second, more complex environment shown in Figure 4-36, though a grid is obtained no flyable path through it exists.

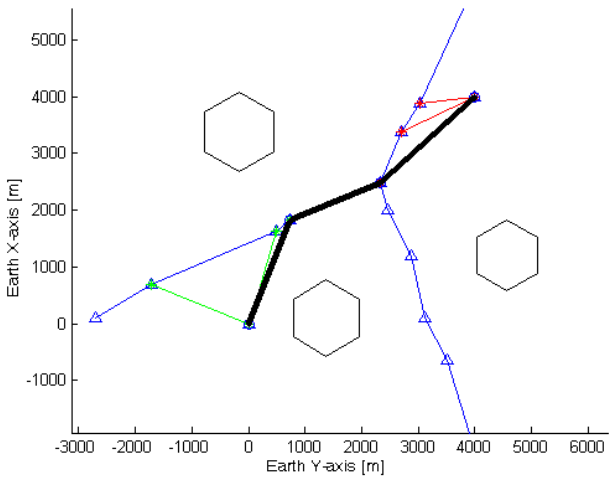

Figure 4-35—Polygon Voronoi Path Using Three Obstacles

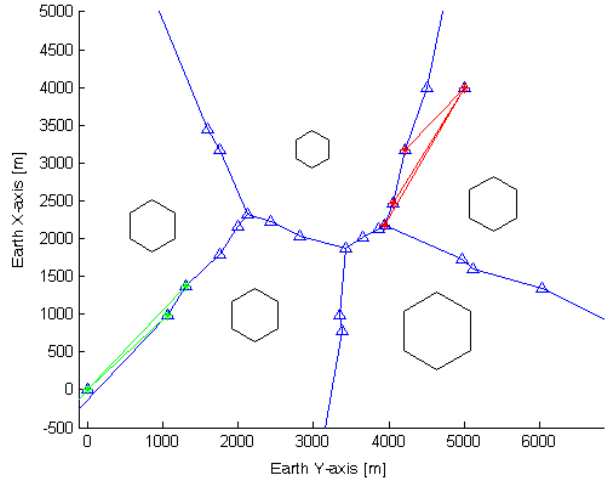

Figure 4-36—Polygon Voronoi Grid with Unsuccessful Path

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 4-37 and Figure 4-38. For scenario 1 the average calculation time was 0.2049 seconds and the average calculation time for scenario 2 was 0.1558 seconds. This method, like the Classic Voronoi, also suffers from a tendency to produce relatively short path segments which prohibit filleting. Thus, for the purposes of this calculation time assessment, the alternative filleting mechanism was used, which could introduce overlapping with obstacles.

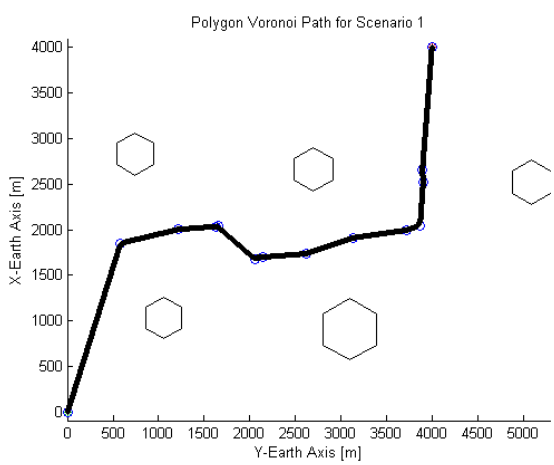

Figure 4-37— Polygon Voronoi Path for Scenario 1

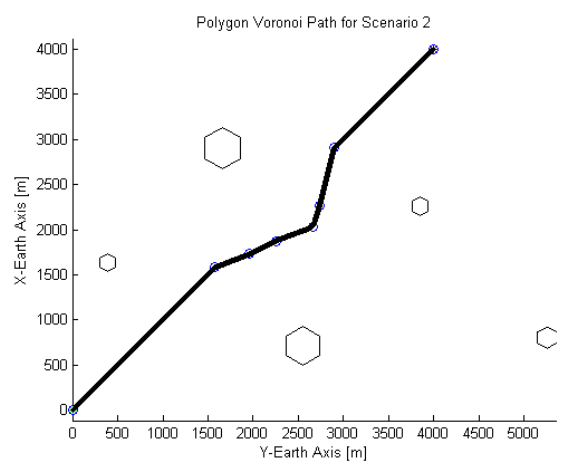

Figure 4-38— Polygon Voronoi Path for Scenario 2

f. Obstacle Avoidance Voronoi

The drawbacks discussed above for the Classic and Polygon Voronoi implementations lead to the development of a modified Voronoi-based path planning scheme, referred to as the Obstacle Avoidance

Voronoi (OAV) (77) (78). In this scheme, path generation considerations may include radars, elevated risk areas, and obstacles, through the using of cylindrical risk zones of variable radius and risk intensity.

As a property of generating a Voronoi diagram, the OAV methodology requires the presence of at least 3 risk zones in the environment in order to generate a path. If fewer than this exist, a more elementary method should be applied. The algorithm generates the Voronoi diagram using the centers of the risk zones as the generating points. The key modification of this algorithm is that it generates the path segments so that any obstacles in the environment are avoided, and in such a way that only one obstacle is present within each Voronoi cell. Additionally, path segments are generated which may potentially pass through lower intensity risk zones, allowing the vehicle to maintain a safe distance from obstacles while limiting threat exposure. If zero threat exposure is desired, all risk zones may be specified to be obstacles. These considerations give the algorithm maximum flexibility. A restriction is that obstacles may not overlap, and this restriction must be checked prior to computation of the path grid.

The OAV algorithm generates the Voronoi diagram such that path segments are produced equidistant with respect to obstacles present in the field of the aircraft. Several steps must be taken to generate the OAV path grid. Modifications to the traditional Voronoi will be emphasized where applicable. The OAV algorithm begins by retrieving and plotting the risk zones. Since the purpose of this methodology is to isolate each obstacle within a Voronoi cell, overlapping of obstacles is undesirable, and therefore, prohibited. These requirements are checked prior to computation of the OAV path grid.

The OAV algorithm first computes the Delaunay triangulation for the set of risk zones using the centers of the risk zones as input points. This Delaunay triangulation produces the original connecting links relevant to a classic Voronoi algorithm. This is illustrated below in Figure 4-39. One key modification to this algorithm is the adjustment of the connecting links. The connecting links are shortened to connect at the edges of obstacles. No modification is made for general risk zones, including radar, with risk intensity less than 2, since some threat exposure may be allowable and can be accounted for with the path selection algorithm once the path choices are generated. Because we require that paths be generated outside of obstacles, the effective length of the connecting link which connects with an obstacle will be reduced to coincide with the edge of the obstacle. This is illustrated in Figure 4-40. Further elaboration on effective length modification for obstacles is illustrated in Figure 4-41.

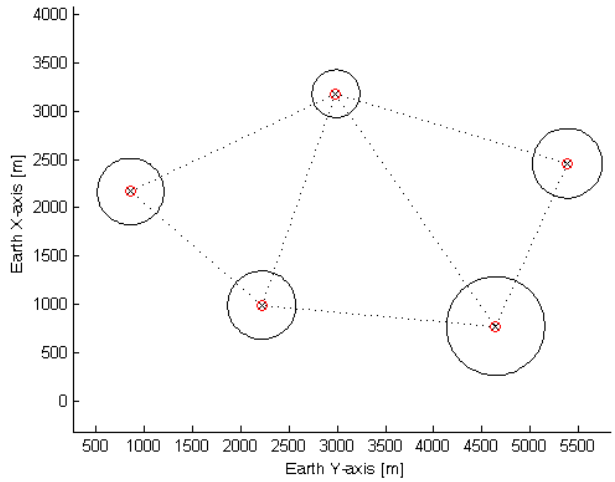

Figure 4-39—Delaunay Triangulation

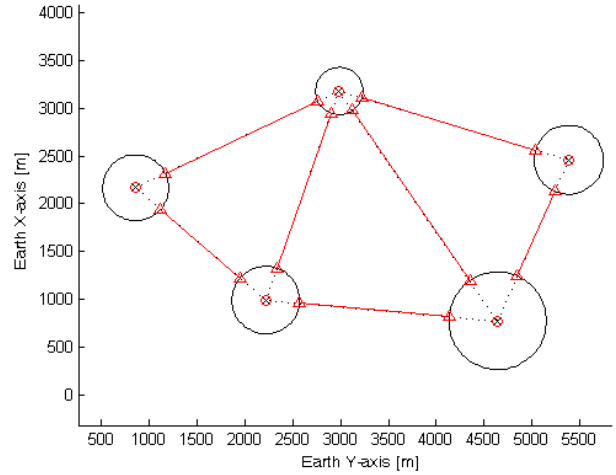

Figure 4-40—Connecting Links Reduced to Effective Length in the Presence of Obstacles Only

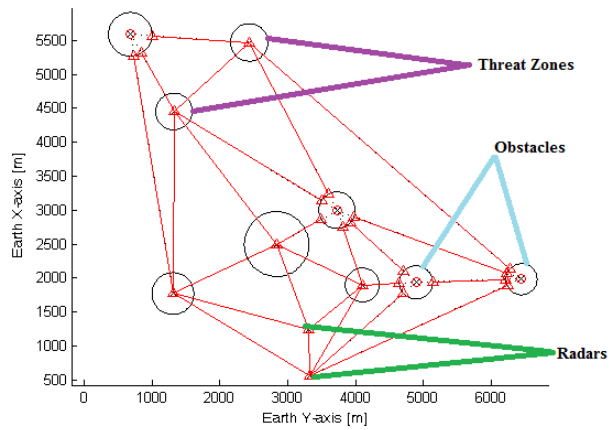

Figure 4-41—Connecting Links Reduced to Effective Length with Varying Types of Zones

According to the traditional Voronoi, the equidistance lines are created by bisecting the connecting links and generating perpendicular lines through these bisection points. Ultimately, the relevant portions of these perpendiculars become the equidistance lines. For the classic Voronoi, in which effective lengths of the connecting segments are not utilized, three of these perpendicular lines would intersect at the same precise geometric location, the circumcenter of the respective Delaunay triangle, which becomes the Voronoi node, as shown in Figure 4-42 (except for a regular input grid, where four perpendiculars would intersect). Introduction of effective connecting segment lengths changes this property, however, and these segments no longer align precisely. This is illustrated in Figure 4-43. Misalignment of these intersections causes issues in terms of producing flyable paths, as these create very short path segments which, as discussed in previous sections, do not accommodate the curving of the path needed to produce a flyable trajectory.

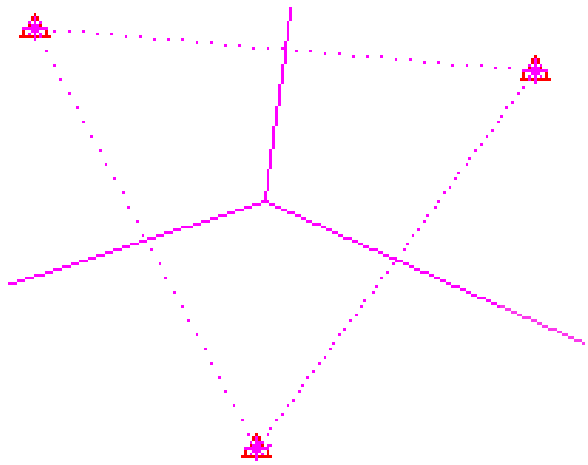

Figure 4-42—Voronoi Perpendicular Intersection to Produce Concise Voronoi Node

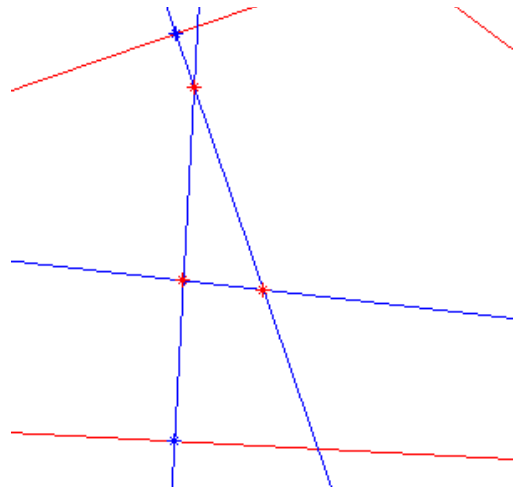

Figure 4-43—Misalignment of Perpendiculars Caused by Introduction of Connection Segment Effective Length

To counter this problem, a single node should instead be generated based upon the triangle geometry rather than at the point of intersection, noting that for the classic Voronoi diagram, these two methods would produce the same point. Therefore, the Voronoi node is generated as the centroid of the Delaunay triangle. However, using the original triangulation would produce the same centroid as the traditional Voronoi methodology, negating the purpose of the effective connecting links. Thus, the effective triangle concept is introduced which places the vertex of the triangle corresponding to an obstacle at the edge of the obstacle bisecting the angle made by the effective connecting links. This vertex then replaces the center of the obstacle in defining the Delaunay triangle. To better illustrate the effect of this procedure, see Figure 4-44 and Figure 4-45.

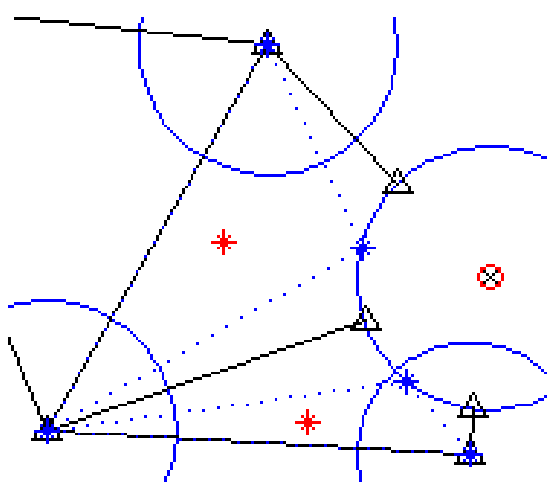

Figure 4-44—Effective Triangles as Produced by Three Zones and One Obstacle

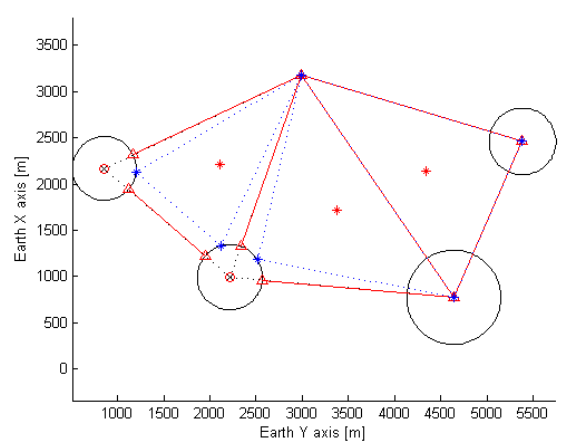

Figure 4-45—Effective Triangles as Produced by Three Zones and One Obstacle

Once the Voronoi nodes have been defined, the equidistance lines are generated to connect them. These form the basis of the path segments grid. First, the internal links are generated, which definitively span two adjacent Voronoi nodes. In the traditional Voronoi method, these would be the only path segments available, which can be a very limited selection. These internal links are illustrated in Figure 4-46. Note that at this point no additional measures have been taken to prevent geometric obstacle crossing. This will be performed by two separate processes discussed later: one to move Voronoi nodes that are generated inside obstacles and one to circumvent obstacles when a segment crosses one. Figure 4-46 specifically illustrates both of these situations.

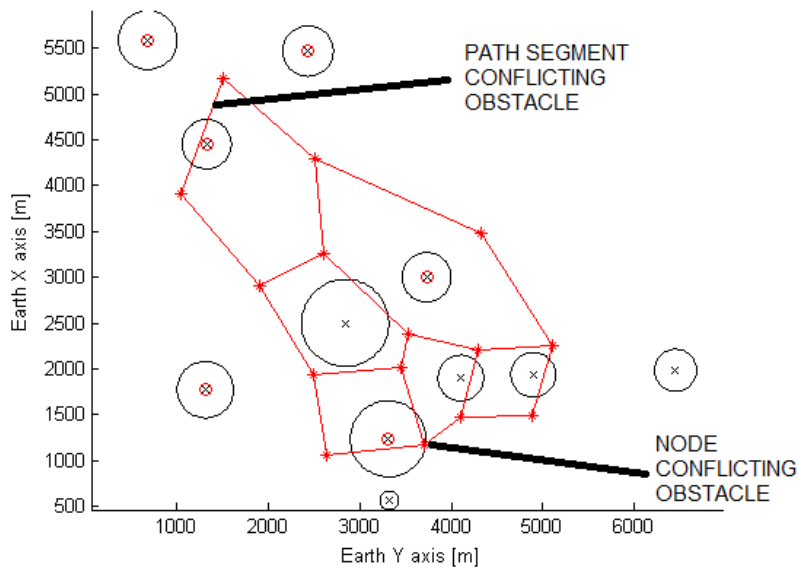

Figure 4-46—Raw Internal Voronoi Path Segments

At this point, any nodes which have been inadvertently generated within obstacles are moved. To do so, the path segments belonging to the cell corresponding to the obstacle are moved to be tangent to the obstacle, thus shifting the node outside the obstacle and ensuring that the obstacle is completely enclosed by the cell. An illustration of the path segments before and after this process can be seen in Figure 4-47.

Knowing the final location of the internal Voronoi nodes, the process of adding the exterior paths begins. First, the Voronoi segments which extend to "infinity" are generated as a vector extending from the Voronoi node through the bisection point of the exterior connecting link, with a sufficiently large magnitude, say 10000m. These additional path segments are illustrated in Figure 4-48. Ultimately, these path segments will join the interior path segments to the exterior path segments.

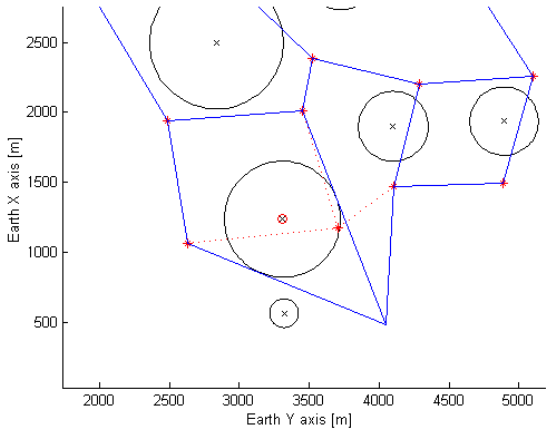

Figure 4-47—Node Moved From Inside an Obstacle

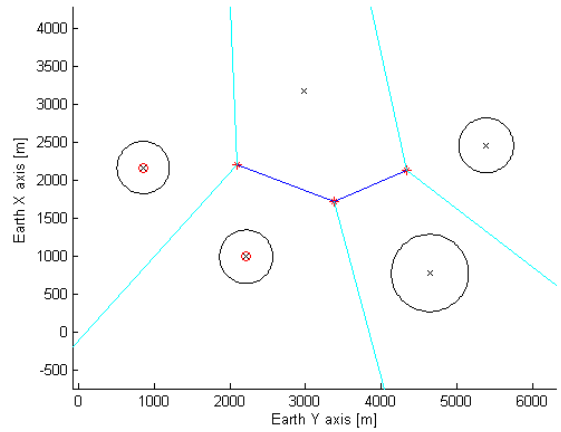

Figure 4-48—Raw External "Infinite" Path Segments

At this point, the external Voronoi segments may be generated. These path segments are a feature of the OAV method which is lacked by the traditional Voronoi. The external Voronoi segments are generated as the connecting tangent lines surrounding the risk zones encompassed by the external Voronoi cells which are not closed, having links which extend to "infinity". These tangent lines are generated, and Voronoi nodes are placed at the intersection of the tangent lines at the locations of the risk zones, and at the intersections of the "infinite" links with the tangent lines. An example of the results of this procedure is presented in Figure 4-49. The final step to generating the OAV path segments grid is to eliminate any obstacle crossing among the path segments. For this purpose, a segment which crosses an obstacle is split into two paths, tangent to the obstacle, with a new Voronoi node located at the intersection of these tangent lines. This process can be seen in Figure 4-50. The OAV grid is now complete. The remaining activity is to connect the start and goal locations to the grid, while observing the desired initial and final headings. These terminal locations are connected to the 3 nearest nodes which are at least the minimum turning radius away from the terminal point, to allow room for the heading maneuver. Due to the mechanism used to generate the grid, it is certain that at most one obstacle may lie between one of these terminal locations and the connecting node. In the event that this connecting segment would cross an obstacle, an additional node is generated to move the path around the obstacle, similar to the mechanism used in the generation of the grid, illustrated in Figure 4-50.

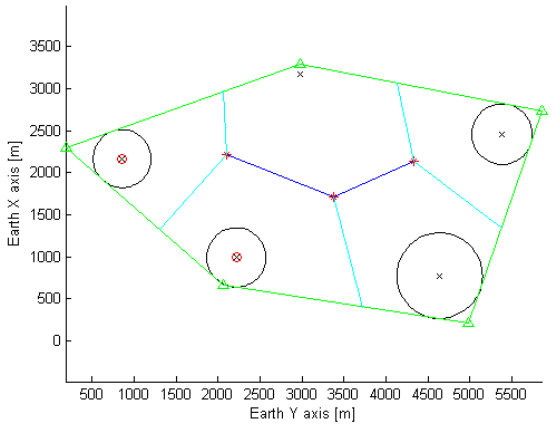

Figure 4-49—Exterior Nodes and Path Segments Added

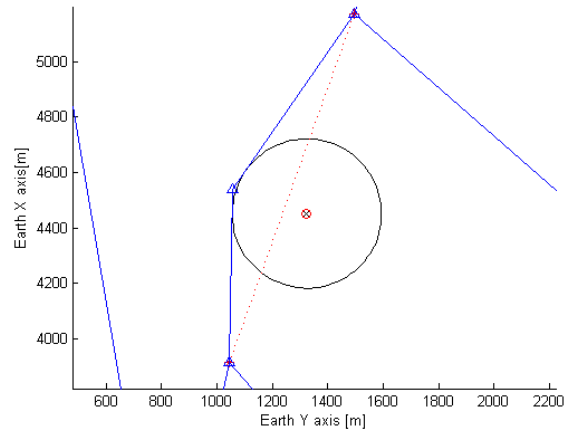

Figure 4-50—Path Segment Moved to Eliminate Obstacle Intersection

Once the Voronoi grid is finalized, it is necessary to connect the start and goal points to the grid. Since the aircraft has an initial heading, and since we may specify a desired heading for the goal location, these must be accounted for when connecting to the grid to ensure that the final path will not result in obstacle intersection. Since this method ensures that only one obstacle may lie within each Voronoi cell, we may be certain that only a maximum of one obstacle may lie between the start or goal position and the Voronoi grid. The heading line is generated and connected with the desired node in such a manner that the resulting path is filletable. Additionally, if this path does cross an obstacle, the path will be moved such that the path is curved around the edge of the obstacle. Several connecting paths are generated for the start and goal positions. Once the grid is completed, a Dijkstra's algorithm is used to choose the optimal path through the grid. Since production of a flyable path is the ultimate goal, this Dijkstra's algorithm implementation verifies that a path may be geometrically filleted prior to its selection. Thus, in situations resulting in sharp turns, it is possible to obtain no flyable path, although a grid can be generated. Finally, the selected path is filleted, and then the trajectory points are generated and sent to the aircraft to define the desired path. Figure 4-51 illustrates the grid generated and path chosen by this algorithm in response to a obstacle-sparse environment. Similarly, Figure 4-52 shows the differences in the grid generated by the algorithm is response to an obstacle-dense environment.

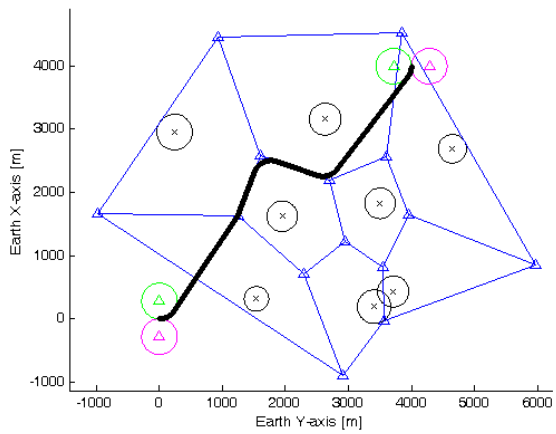

Figure 4-51—Path Selected by the Zone Voronoi Algorithm in a Threat-Filled Environment

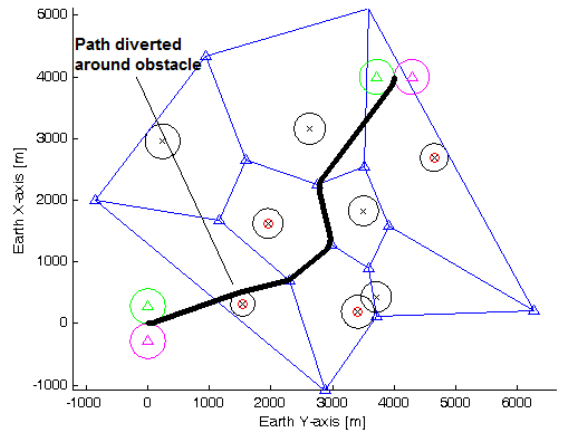

Figure 4-52—Path Selected by the Zone Voronoi Algorithm in a Complex Environment

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 4-53 and Figure 4-54. For scenario 1, the average calculation time was 0.112 seconds and the average calculation time for scenario 2 was 0.1439 seconds.

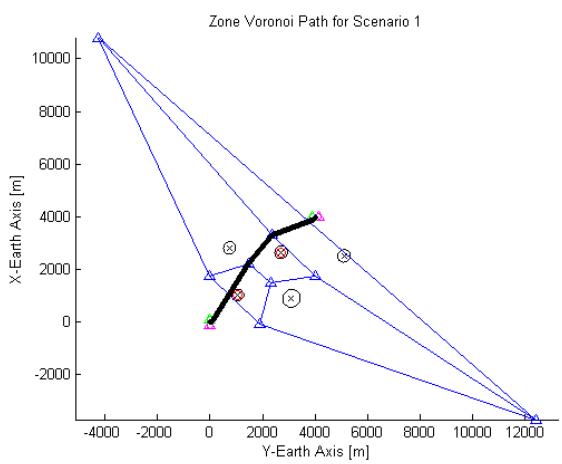

Figure 4-53—Zone Voronoi Path for Scenario 1

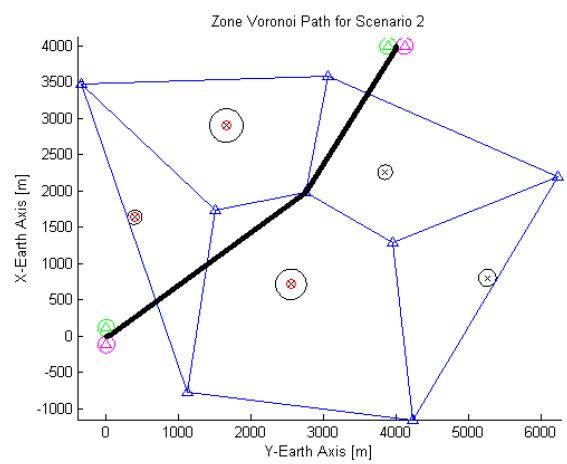

Figure 4-54— Zone Voronoi Path for Scenario 2

Chapter 5. ALGORITHM DEVELOPMENT WITH WAYPOINT TRACKING METHODS

Pose-based, or waypoint tracking, methods provide a more flexible solution to the path planning problem, as nearly all situations can be handled, depending upon proper selection of an adequate and optimal series of poses, regardless of the representation of the environment. Thus, the specification of poses becomes a separate problem, handled as needed for the specific scenario, but generation of a flyable trajectory can always be accomplished in the same manner. Additionally, one of the most important uses for UAVs is observation and surveillance. Whether it is an area which is unsafe for human pilots, a vast area of meager interest, or simply a tedious area to scout, UAVs do a great deal of being "eyes in the sky." As a consequence, generation of effective target observation trajectories is crucial. This section focuses upon a series of 2-dimensional pose-based planners, geared toward various purposes, with two specific implementations. First, waypoint tracking can be accomplished by finding the path that satisfies 2 poses. However, the area traversed will not be specified for this approach. If an area of interest is to be observed, the radius of the area should be used to specify the aircraft turning radius, as flyable by the aircraft, and the direction of turn should be specified in addition to the pose, to ensure that the appropriate geography is traversed for observation to take place. This will be explained in more detail as applicable to the following pose-based path planning mechanisms. All algorithms are implemented through the Matlab®/Simulink environment. All calculation times are with respect to a Windows 7 desktop computer with a 2.2 GHz Core i7 processor and 8 GB of RAM.

A. Point of Interest Observation

The following method is intended to generate the trajectory for the aircraft, in which the path follows the edges of the particular points of interest to allow for effective observation and data collection. The algorithm assumes a circling radius of each point equal to the minimum turning radius of the craft, R, for simplicity. This algorithm is intended to make an observational loop, such that the UAV takes off from a ground station and returns to this same ground station after all points have been observed. The ending point is not required to be the starting point for the algorithm; this is done for convenience with the intended purpose of the aircraft.

The inputs to this planner consist of a series of ground-based points in 2 dimensions which need to be observed, as specified in Equation 34. For the purposes of this algorithm, the first point in this array will be considered to be the starting position of the aircraft and the last point in the array will be considered to be the goal position of the aircraft.

$$\mathrm{POI} = \begin{bmatrix} X_1 & Y_1 \\ X_2 & Y_2 \\ \vdots & \vdots \\ X_n & Y_n \end{bmatrix} \tag{34}$$

Since this algorithm produces a pattern of straight and arc maneuver segments, the output of this algorithm is represented as a series of "path points" which define the initial point of each maneuver, as shown in Equation 35. In this equation, $p = 2n - 2$, where n is the number of points of interest including the start and goal locations.

$$\mathrm{PP} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_p & y_p \end{bmatrix} \tag{35}$$

This algorithm begins by assigning the turn direction of each maneuver which will allow the aircraft to best view the point of interest. This algorithm is performed entirely within the Earth Reference Frame, and relies heavily upon the cross product vector operation. The initial and final headings are not specified for this algorithm; thus, these points will not be associated with maneuver circles. In order to determine the turn direction of each maneuver, a set of 3 consecutive points (A, B, C) is used to define 2 vectors, as shown below, where $\mathrm{POI}_j = [X_j \quad Y_j]$ and $i = 2 \rightarrow n - 1$.

$$[\vec{r}_{BA}]_E = \begin{bmatrix} X_{i-1} - X_i \\ Y_{i-1} - Y_i \\ 0 \end{bmatrix}_E \tag{36}$$

$$[\vec{r}_{BC}]_E = \begin{bmatrix} X_{i+1} - X_i \\ Y_{i+1} - Y_i \\ 0 \end{bmatrix}_E \tag{37}$$

Using the cross product operation, the turn direction of each arc maneuver, d, can be specified according to the following relationships:

$$\begin{aligned} [\vec{c}]_E = [\vec{r}_{BA} \times \vec{r}_{BC}]_E &= \left[\begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ X_{i-1} - X_i & Y_{i-1} - Y_i & 0 \\ X_{i+1} - X_i & Y_{i+1} - Y_i & 0 \end{vmatrix}\right]_E \\ &= \begin{bmatrix} 0 \\ 0 \\ (X_{i-1} - X_i)(Y_{i+1} - Y_i) - (X_{i+1} - X_i)(Y_{i-1} - Y_i) \end{bmatrix}_E \end{aligned} \tag{38}$$

$$d_i = \mathrm{sign}\big((X_{i-1} - X_i)(Y_{i+1} - Y_i) - (X_{i+1} - X_i)(Y_{i-1} - Y_i)\big) \tag{39}$$

$$\text{if } d_i = \begin{cases} +1, & \text{turn direction is clockwise} \\ -1, & \text{turn direction is counterclockwise} \end{cases} \quad 40$$

Once the turn directions have been assigned, the points of interest must be checked for path overlapping. Since each point of interest is observed from a radius equal to R, then overlap is defined as:

$$\sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} < 2R \quad 41$$

If the above inequality is true for any 2 consecutive POI, then the turn direction for these must be the same. Thus, $d_i \neq d_{i+1}$, then the sign of d_{i+1} and all subsequent turn directions must be made opposite, illustrated in Equation 42.

$$\text{if } d_i \neq d_{i+1}, \text{then for } j = i + 1 \rightarrow n - 1, d_j = -d_j \quad 42$$

With the maneuver directions assigned, the path can be defined. This is done by calculating the tangent lines shared by the pair of observation circles. Four tangents are shared by any set of 2 circles, as illustrated below in Figure 5-1. The appropriate tangent may be determined based upon the turn directions of each of the maneuver circles. If the turn directions are the same, a straight tangent, or one whose slope is equal to the slope of the centerline connecting the points of interest, will be chosen. If they are different, one of the crossing tangents will be needed. This eliminates a pair of the tangents from the possible choices, and the remaining tangent may be eliminated using the turn direction of the first maneuver.

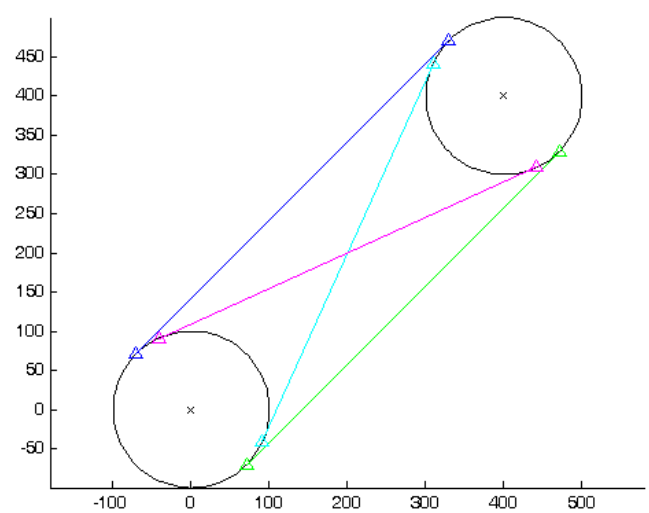

Figure 5-1—Four Connecting Tangents

To calculate the four tangents, the solution is reduced to finding the intersection points between two circles. For the start and goal maneuvers, since there is not a maneuver circle associated with these points,

this geometry will be defined as in Figure 5-2. For all other points of interest, there will exist a pair of maneuver circles, with the resulting geometry shown in Figure 5-3. If desired heading is to be considered for the start and goal positions, this may be accomplished by also associating a maneuver circle with these points, wherein the start and goal positions are the tangent point of the circle, with the centers of the circles located normal to the heading vector.

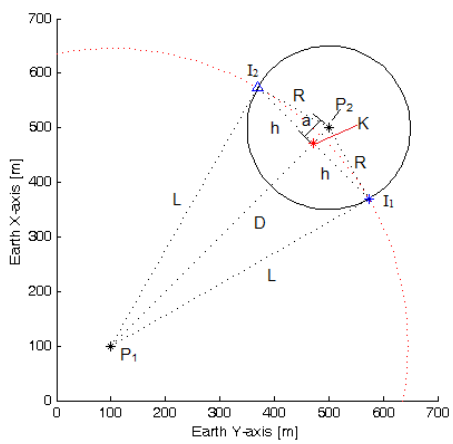

Figure 5-2—Termination Point Geometry

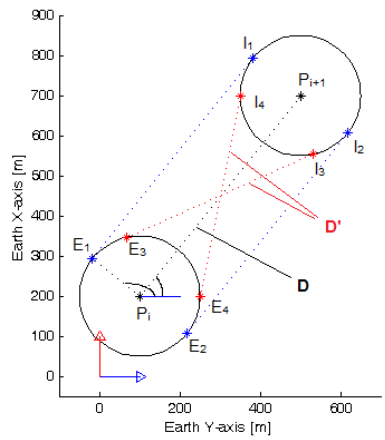

Figure 5-3—Point of Interest Geometry

The geometry in Figure 5-2 shows that the solution to the following system of equations will yield the coordinates in the Earth Reference Frame for the tangent entry points of the first maneuver circle.

$$[\vec{r}_{OP_1}]_E = \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix}_E \tag{43}$$

$$[\vec{r}_{OP_2}]_E = \begin{bmatrix} X_2 \\ Y_2 \end{bmatrix}_E \tag{44}$$

$$\begin{aligned} (X_I - X_2)^2 + (Y_I - Y_2)^2 &= R^2 \\ (X_I - X_1)^2 + (Y_I - Y_1)^2 &= L^2 \end{aligned} \tag{45}$$

where:

$$D = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \tag{46}$$

$$L = \sqrt{D^2 - R^2} \tag{47}$$

Thus, using this geometry, we can simplify the solution to a geometric rather than algebraic one, to yield the following coordinate solution for the tangent entry points.

$$a = \frac{R^2 - L^2 + D^2}{2D} \tag{48}$$

$$h = \sqrt{R^2 - a^2} \tag{49}$$

$$[\vec{r}_{OK}]_E = \begin{bmatrix} X_K \\ Y_K \end{bmatrix}_E = \begin{bmatrix} X_2 \\ Y_2 \end{bmatrix}_E + \frac{a}{D}\begin{bmatrix} X_1 - X_2 \\ Y_1 - Y_2 \end{bmatrix}_E \tag{50}$$

$$[\vec{r}_{OI_1}] = \begin{bmatrix} X_{I1} \\ Y_{I1} \end{bmatrix}_E = \begin{bmatrix} X_K + \frac{h}{D}[Y_1 - Y_2] \\ Y_K - \frac{h}{D}[X_1 - X_2] \end{bmatrix}_E, \text{if } d_1 = -1 \tag{51}$$

$$[\vec{r}_{OI_2}] = \begin{bmatrix} X_{I2} \\ Y_{I2} \end{bmatrix}_E = \begin{bmatrix} X_K - \frac{h}{D}[Y_1 - Y_2] \\ Y_K + \frac{h}{D}[X_1 - X_2] \end{bmatrix}_E, \text{if } d_1 = +1 \tag{52}$$

If a right-hand turn is desired for the first maneuver, tangent point I_2 should be selected. Otherwise, I_1 should be selected.

With the first entry point computed, the tangents for the remaining pairs of maneuver circles will be computed using the position vectors and relative vectors defined in Figure 5-3. This procedure is outline below. In order to determine the straight tangents between maneuver circles i and $i + 1$, the following relationships are needed. The slope of the centerline is angle α:

$$\alpha = \tan^{-1}\left(\frac{Y_{i+1} - Y_i}{X_{i+1} - X_i}\right) \tag{53}$$

The normal to the centerline is:

$$\beta = \alpha + \frac{\pi}{2} \tag{54}$$

The magnitude of the centerline is:

$$D = \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} \tag{55}$$

The straight tangent exit points for maneuver circle i may be determined from the geometric relationships:

$$[\vec{r}_{OE_1}]_E = \begin{bmatrix} X_i + R\cos(\beta) \\ Y_i + R\sin(\beta) \end{bmatrix}_E \tag{56}$$

$$[\vec{r}_{OE_2}]_E = \begin{bmatrix} X_i - R\cos(\beta) \\ Y_i - R\sin(\beta) \end{bmatrix}_E \tag{57}$$

The straight tangent entry points for maneuver circle $i + 1$ may be defined as:

$$\left[\vec{r}_{OI_1}\right]_E = \begin{bmatrix} X_{i+1} + R\cos(\beta) \\ Y_{i+1} + R\sin(\beta) \end{bmatrix}_E \quad 58$$

$$\left[\vec{r}_{OI_2}\right]_E = \begin{bmatrix} X_{i+1} - R\cos(\beta) \\ Y_{i+1} - R\sin(\beta) \end{bmatrix}_E \quad 59$$

These points may also be defined using the following vector relationships.

$$\left[\vec{r}_{OP_i}\right]_E = \begin{bmatrix} X_i \\ Y_i \end{bmatrix}_E \quad 60$$

$$\left[\vec{r}_{OP_{i+1}}\right]_E = \begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix}_E \quad 61$$

$$[\hat{u}]_E = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} E \quad 62$$

The centerline vector connecting the points of interest is then:

$$\left[\vec{r}_{P_iP_{i+1}}\right]_E = \left[\vec{r}_{OP_{i+1}}\right]_E - \left[\vec{r}_{OP_i}\right]_E = \begin{bmatrix} X_{i+1} - X_i \\ Y_{i+1} - Y_i \end{bmatrix}_E \quad 63$$

and the vector connecting E_i to P_k is:

$$\left[\vec{r}_{P_iE_1}\right]_E = \left[\frac{R}{D}\hat{u} \times \vec{r}_{P_iP_{i+1}}\right]_E \quad 64$$

Then the straight tangent exit points for maneuver circle i can be defined as:

$$\left[\vec{r}_{OE_1}\right]_E = \left[\vec{r}_{OP_i}\right]_E + \left[\vec{r}_{P_iE_1}\right]_E \quad 65$$

$$\left[\vec{r}_{OE_2}\right]_E = \left[\vec{r}_{OP_i}\right]_E - \left[\vec{r}_{P_iE_1}\right]_E \quad 66$$

Then the corresponding straight tangent entry points can be defined as:

$$\left[\vec{r}_{OI_1}\right]_E = \left[\vec{r}_{OE_1}\right]_E + \left[\vec{r}_{P_iP_{i+1}}\right]_E \quad 67$$

$$\left[\vec{r}_{OI_2}\right]_E = \left[\vec{r}_{OE_2}\right]_E + \left[\vec{r}_{P_iP_{i+1}}\right]_E \quad 68$$

Next the crossing tangent exit points for maneuver circle i are may be determined from the geometric relationships:

$$\left[\vec{r}_{OE_3}\right]_E = \begin{bmatrix} X_i - R\cos(\alpha + \beta) \\ Y_i - R\sin(\alpha + \beta) \end{bmatrix}_E \quad 69$$

$$\left[\vec{r}_{OE_4}\right]_E = \begin{bmatrix} X_i - R\cos(\alpha - \beta) \\ Y_i - R\sin(\alpha - \beta) \end{bmatrix}_E \quad 70$$

And the crossing tangent entry points for maneuver circle $i+1$ may be defined as:

$$\left[\vec{r}_{OI_3}\right]_E = \begin{bmatrix} X_{i+1} + R\cos(\alpha+\beta) \\ Y_{i+1} + R\sin(\alpha+\beta) \end{bmatrix}_E \quad 71$$

$$\left[\vec{r}_{OI_4}\right]_E = \begin{bmatrix} X_{i+1} + R\cos(\alpha-\beta) \\ Y_{i+1} + R\sin(\alpha-\beta) \end{bmatrix}_E \quad 72$$

The crossing tangent entry points for maneuver circle $i+1$ may also be computed as such:

$$D' = \left|\vec{r}_{E_3I_3}\right| = \left|\vec{r}_{E_4I_4}\right| = 2\sqrt{\frac{D^2}{4} - R^2} \quad 73$$

$$\left[\vec{r}_{E_3P_i}\right]_E = \left[\vec{r}_{OE_3}\right]_E - \left[\vec{r}_{OP_i}\right]_E \quad 74$$

$$\left[\vec{r}_{E_3I_3}\right]_E = \left[\frac{D'}{R}\vec{r}_{E_3P_i} \times \hat{u}\right]_E \quad 75$$

$$\left[\vec{r}_{OI_3}\right]_E = \left[\vec{r}_{OE_3}\right]_E + \left[\vec{r}_{E_3I_3}\right]_E \quad 76$$

$$\left[\vec{r}_{E_4P_i}\right]_E = \left[\vec{r}_{OE_4}\right]_E - \left[\vec{r}_{OP_i}\right]_E \quad 77$$

$$\left[\vec{r}_{E_4I_4}\right]_E = \left[\frac{D'}{R}\vec{r}_{E_4P_i} \times \hat{u}\right]_E \quad 78$$

$$\left[\vec{r}_{OI_4}\right]_E = \left[\vec{r}_{OE_4}\right]_E - \left[\vec{r}_{E_4I_4}\right]_E \quad 79$$

Finally, the remaining exit tangent point will be computed using the same geometry as for computing the initial entry tangent. Once all of the path points have been defined, the sweep angle for the arc maneuvers should be computed to allow for the generation of the aircraft trajectory from the geometry. For each maneuver circle i, there exists a selected entry tangent point, I, and exit tangent point, E. Using these, the sweep angle can be computed as:

$$\mu = cos^{-1}\frac{\vec{r}_{P_iI} \cdot \vec{r}_{P_iE}}{\left|\vec{r}_{P_iI}\right|\left|\vec{r}_{P_iE}\right|} \quad 80$$

Using the straight segments as rays pointing toward the maneuver circle, if the rays intersect, the sweep angle equals theta. If they diverge, the sweep angle equal 2π-θ.

The path examples shown below were computed using the method discussed in this section.

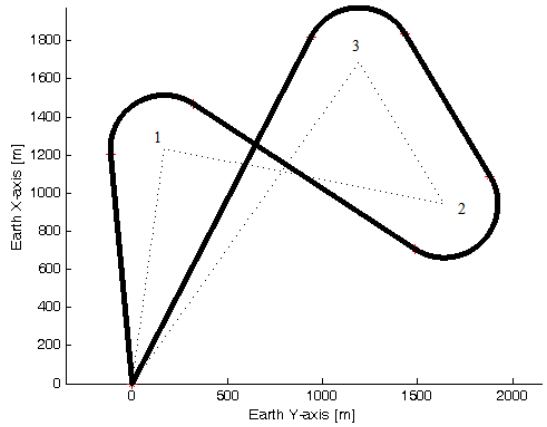

Figure 5-4—Basic Path Using Points of Interest

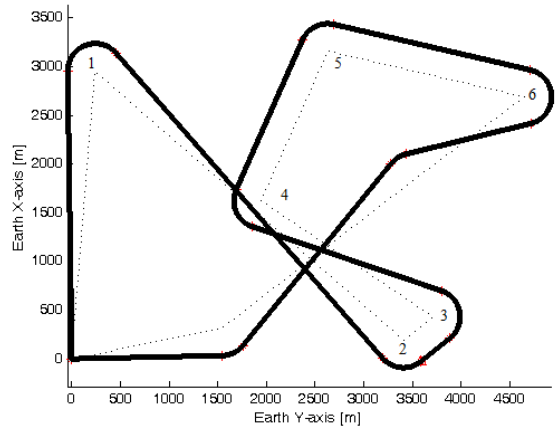

Figure 5-5—Complex Path Using Points of Interest

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 5-6 and Figure 5-7. For scenario 1, the average calculation time was 2.171 seconds and the average calculation time for scenario 2 was 2.1543 seconds.

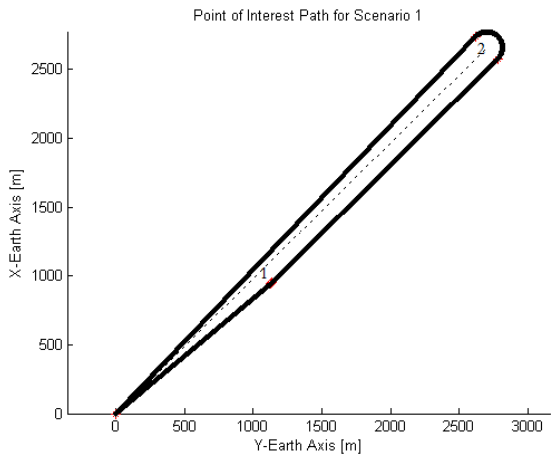

Figure 5-6—Point of Interest Path for Scenario 1

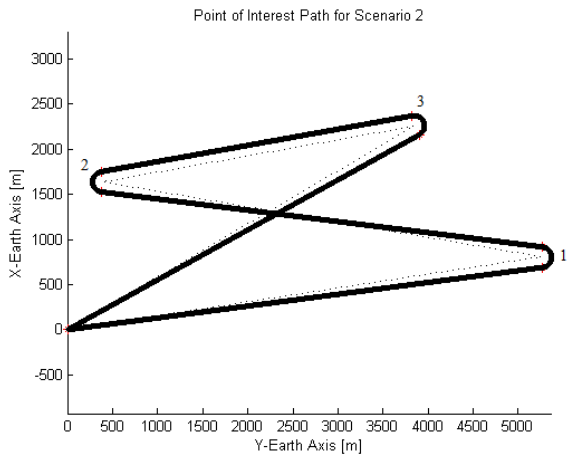

Figure 5-7— Point of Interest Path for Scenario 2

B. Dubins Waypoint and Observation Trajectory Generation

The Dubins path planner was originally developed by L.E. Dubins in 1957 and proven to provide the shortest path of a defined curvature constraint between two positions with defined tangent, referred to here as a set of poses. This method is fundamentally similar to the above described Point of Interest method derived by the author, though the Dubins planner is more general and relies on computation of the four possible paths due to turn combinations to choose the shortest path, rather than selecting the turn-directions which will produce a path circling the desired location as the Point of Interest method does. However, turn directions may also be specified for the Dubins planner to produce a similar effect of observing a desired

location, similar to the Point of Interest method. The standard Dubins method is widely used and represents a conceptual starting point for the clothoid planner discussed in the next section.

For any set of two poses, there exist 4 possible paths which will pass from the start pose smoothly to the finish pose, based upon the traditional Dubins curve-straight-curve path structure. These combinations are, of course, right-right, right-left, left-left, and left-right. Of these, one choice of poses will yield the shortest continuous path of specified curvature. However, for the methodology discussed, the turn directions need to be arbitrarily initially assigned to generate the path. Thus, finding the shortest combination involves computing all four and choosing the one which is the shortest. To compute these, the start and finish poses and their associated curvatures must be specified, with the curvature generally chosen to be the minimum turning radius of the aircraft for all poses. For each pose, there exist two heading circles which lie tangent to the pose, usually referred to as the primary circles. These circles may be specified by their centers located at:

$$[\vec{r}_{OC}]_E = \begin{bmatrix} x \\ y \end{bmatrix}_E \pm \frac{1}{\kappa} \begin{bmatrix} \cos\left(\psi + \frac{\pi}{2}\right) \\ \sin\left(\psi + \frac{\pi}{2}\right) \end{bmatrix} \qquad 81$$

where the choice of a positive sign is associated with a right turn and the choice of a negative sign is associated with the left turn.

Segments of the primary circles will ultimately form the curve segments of the curve-straight-curve path architecture. The straight segments are thus provided by the common tangents between these primary circles. For each set of two circles, there exist 4 common tangents: 2 straight-tangents and two cross-tangents. These are shown in Figure 5-8. Fortunately, the start pose heading direction immediately eliminates two of these leaving only one remaining straight tangent and one remaining cross tangent. Additionally, the finish pose heading eliminates one more of these. A full diagram of these tangents is provided below in Figure 5-9. Thus, the four turn direction combinations dictate what the four paths will be:

- Right-Right (RR): The start right primary circle connected via a straight-tangent line to the finish right primary circle.
- Left-Left (LL): The start left primary circle connected via a straight-tangent line to the finish left primary circle.
- Right-Left (RL): The start right primary circle connected via a cross-tangent line to the finish left primary circle.
- Left-Right (LR): The start left primary circle connected via a cross-tangent line to the finish right primary circle.

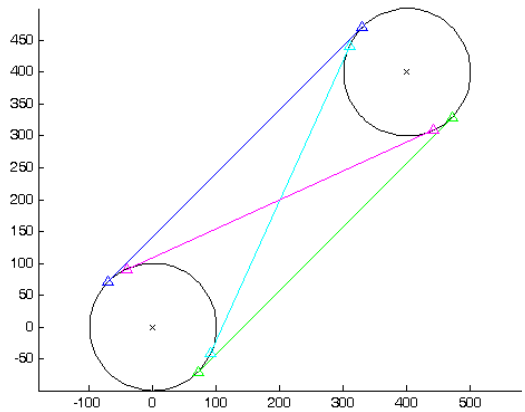

Figure 5-8—Tangent Lines Between 2 Circles

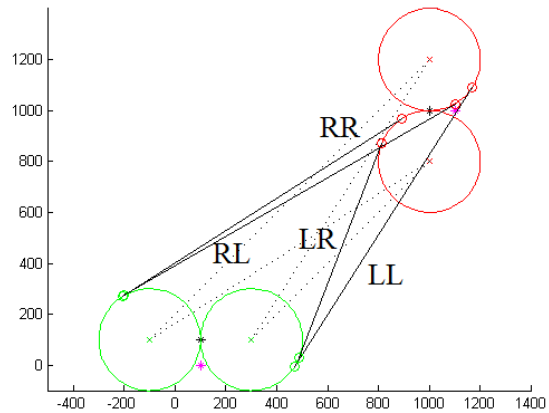

Figure 5-9—Relevant Path Tangents

It should be noted that the radius of curvature of the primary circles will dictate which of these tangents is available. If the right and left primary circle of the poses overlap then only the straight tangents exist, and thus this combination produces no feasible path. Equally, if one of the right primary circles completely contains the other, or likewise for the left primary circles then only the cross tangents exist and the combination will not produce a feasible path. In general, if the following criteria is met then a path exists for the turn combination, where *c* is the centerline distance between the relevant primary circles and the signs are specified based upon the turn direction choices, as above.

$$\frac{1}{c^2}\left(\frac{\pm 1}{\kappa_f} - \frac{\pm 1}{\kappa_s}\right) < 1 \qquad 82$$

a. Computing the Straight-Tangent Solutions

The right-right path combination consists of two clockwise arcs, and the left-left path combination consists of two counter-clockwise arcs, each connected by the common straight tangent. The centerline distance is given by:

$$c = \left|[\vec{r}_{OC_f}]_E - [\vec{r}_{OC_s}]_E\right| = \sqrt{\left(x_{cf_E} - x_{cs_E}\right)^2 + \left(y_{cf_E} - y_{cs_E}\right)^2} \qquad 83$$

with the circle centers calculated according to Equation 81. The angles α and β are intermediate geometric relationships used in this calculation. The angle α is the angle between the centerline and the tangent line, and the angle β is the slope of the centerline. These are defined below in Equations 84 and 85, where $R = \frac{1}{\kappa}$. Additionally, this geometry can be seen in Figure 5-10.

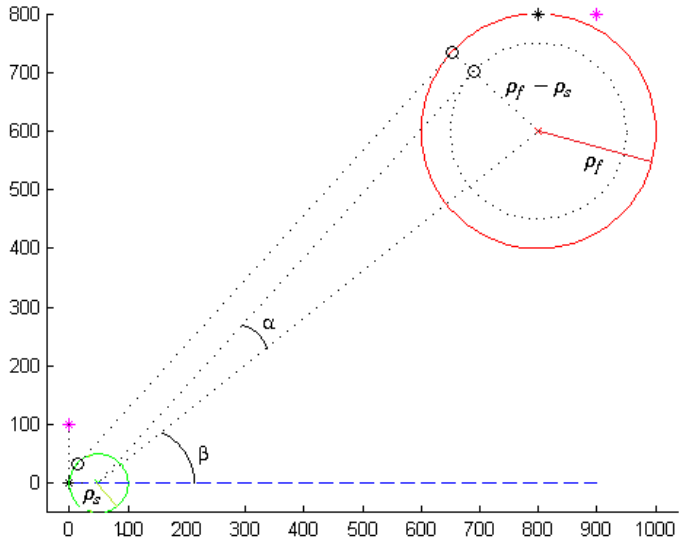

Figure 5-10—Straight-Tangent Construction Geometry

$$\alpha = \sin^{-1}\left(\frac{R_f - R_s}{c}\right) \tag{84}$$

$$\beta = \tan^{-1}\left(\frac{y_{fc} - y_{sc}}{x_{fc} - x_{sc}}\right) \tag{85}$$

For the right-right path combination,

$$\tau_s = \beta - \alpha + \frac{3\pi}{2}, \qquad \tau_s \in [0, 2\pi] \tag{86}$$

$$\tau_f = \beta - \alpha + \frac{3\pi}{2}, \qquad \tau_f \in [0, 2\pi] \tag{87}$$

And for the left-left path combination,

$$\tau_s = \alpha + \beta + \frac{\pi}{2}, \qquad \tau_s \in [0, 2\pi] \tag{88}$$

$$\tau_f = \alpha + \beta + \frac{\pi}{2}, \qquad \tau_f \in [0, 2\pi] \tag{89}$$

Thus, the start arc exit tangent is calculated as:

$$\left[\vec{r}_{OT_s}\right]_E = \begin{bmatrix} x_{tx} \\ y_{tx} \end{bmatrix}_E = \begin{bmatrix} x_{sc} \\ y_{sc} \end{bmatrix}_E + R_s \begin{bmatrix} \cos(\tau_s) \\ \sin(\tau_s) \end{bmatrix} \tag{90}$$

and the finish arc entry tangent is:

$$\left[\vec{r}_{OT_f}\right]_E = \begin{bmatrix} x_{tn} \\ y_{tn} \end{bmatrix}_E = \begin{bmatrix} x_{fc} \\ y_{fc} \end{bmatrix}_E + R_f \begin{bmatrix} \cos(\tau_f) \\ \sin(\tau_f) \end{bmatrix} \tag{91}$$

The sweep angle for the start and finish arcs can then be calculated using the following relationships given in Equations 92 through 99. Note that the conditional statement switches directions depending on whether the solution is right-right or left-left.

$$\left[\vec{r}_{OA_S}\right]_E = \begin{bmatrix} x_s - x_{cs} \\ y_s - y_{cs} \end{bmatrix}_E \tag{92}$$

$$\left[\vec{r}_{OB_S}\right]_E = \begin{bmatrix} x_{tx} - x_{cs} \\ y_{tx} - y_{cs} \end{bmatrix}_E \tag{93}$$

$$\left[\vec{r}_{OC_S}\right]_E = \left[\vec{r}_{OA_S} \times \vec{r}_{OB_S}\right]_E = \begin{bmatrix} x_{C_S} \\ y_{C_S} \\ z_{C_S} \end{bmatrix}_E \tag{94}$$

$$\begin{aligned} &\text{if } \operatorname{sign}(z_{C_S}) == \text{start turn direction} \\ &\quad \mu_s = \cos^{-1}\left(\frac{\left[\vec{r}_{OA_S}\right]_E^T \cdot \left[\vec{r}_{OB_S}\right]_E}{\left|\vec{r}_{OA_S}\right|\left|\vec{r}_{OB_S}\right|}\right), \qquad \mu_s \in [0, \pi] \\ &\text{elseif } \operatorname{sign}(z_{C_S}) == \text{opposite start turn direction} \\ &\quad \mu_s = 2\pi - \cos^{-1}\left(\frac{\left[\vec{r}_{OA_S}\right]_E^T \cdot \left[\vec{r}_{OB_S}\right]_E}{\left|\vec{r}_{OA_S}\right|\left|\vec{r}_{OB_S}\right|}\right), \qquad \mu_s \in [\pi, 2\pi] \end{aligned} \tag{95}$$

$$\left[\vec{r}_{OA_F}\right]_E = \begin{bmatrix} x_{tn} - x_{cf} \\ y_{tn} - y_{cf} \end{bmatrix}_E \tag{96}$$

$$\left[\vec{r}_{OB_F}\right]_E = \begin{bmatrix} x_f - x_{cf} \\ y_f - y_{cf} \end{bmatrix}_E \tag{97}$$

$$\left[\vec{r}_{OC_F}\right]_E = \left[\vec{r}_{OA_F} \times \vec{r}_{OB_F}\right]_E = \begin{bmatrix} x_{C_F} \\ y_{C_F} \\ z_{C_F} \end{bmatrix}_E \tag{98}$$

$$\begin{aligned} &\text{if } \operatorname{sign}(z_{C_F}) == \text{finish turn direction} \\ &\quad \mu_f = \cos^{-1}\left(\frac{\left[\vec{r}_{OA_F}\right]_E^T \cdot \left[\vec{r}_{OB_F}\right]_E}{\left|\vec{r}_{OA_F}\right|\left|\vec{r}_{OB_F}\right|}\right), \qquad \mu_f \in [0, \pi] \\ &\text{elseif } \operatorname{sign}(z_{C_F}) == \text{opposite finish turn direction} \\ &\quad \mu_f = 2\pi - \cos^{-1}\left(\frac{\left[\vec{r}_{OA_F}\right]_E^T \cdot \left[\vec{r}_{OB_F}\right]_E}{\left|\vec{r}_{OA_F}\right|\left|\vec{r}_{OB_F}\right|}\right), \qquad \mu_f \in [\pi, 2\pi] \end{aligned} \tag{99}$$

Using the known arc endpoints and sweep angles, the aircraft trajectory can then be generated from the geometry for either of the straight-tangent paths.

b. Computing the Cross-Tangent Solutions

For the cross tangent solutions, it becomes highly necessary to check for the existence of the path. Due to the nature of application of this path planner, it is far more likely that the two relevant primary circles will overlap, eliminating the possible cross-tangent solution, than one of the relevant primary circles completely containing the other, which is the criterion to eliminate a straight tangent solution. Once it has been verified that the solution exists, the solution can be calculated, following a similar procedure as above. First, calculate the centerline distance between the two relevant primary circles, according to Equation 83. Then calculate the intermediate angle values according to the following relationships. The geometry for this scheme is illustrated in Figure 5-11.

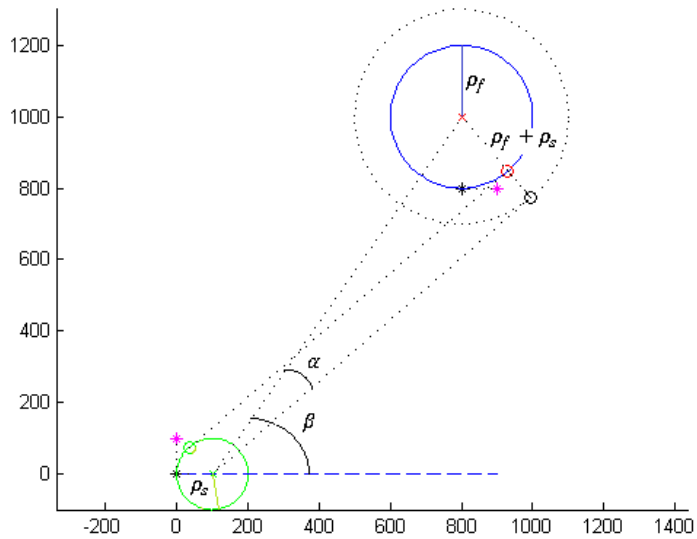

Figure 5-11--Cross-Tangent Construction Geometry

$$\alpha = \sin^{-1}\left(\frac{R_f + R_s}{c}\right) \qquad 100$$

$$\beta = \tan^{-1}\left(\frac{y_{fc} - y_{sc}}{x_{fc} - x_{sc}}\right) \qquad 101$$

From this we can calculate the total angles τ_s and τ_f. For the right-left path combination:

$$\tau_s = \beta + \alpha - \frac{\pi}{2}, \qquad \tau_s \in [0, 2\pi] \qquad 102$$

$$\tau_f = \beta + \alpha + \frac{\pi}{2}, \qquad \tau_f \in [0, 2\pi] \qquad 103$$

And for the left-right path combination:

$$\tau_s = \beta - \alpha + \frac{\pi}{2}, \qquad \tau_s \in [0, 2\pi] \tag{104}$$

$$\tau_f = \beta - \alpha + \frac{3\pi}{2}, \qquad \tau_f \in [0, 2\pi] \tag{105}$$

And again, the entry points are defined by Equations 90 and 91. Similarly, using these endpoints, the sweep angles can be calculated according to Equations 92 through 99, and the resulting geometry can be used to calculate the flyable trajectory.

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 5-12 and Figure 5-13. For scenario 1, the average calculation time was 0.0736 seconds and the average calculation time for scenario 2 was 0.0809 seconds.

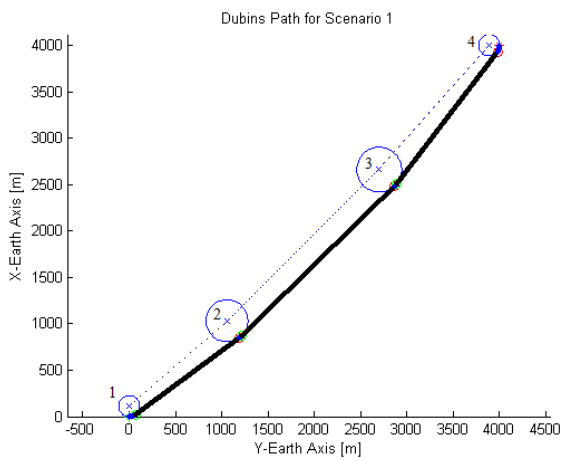

Figure 5-12—Dubins Path for Scenario 1

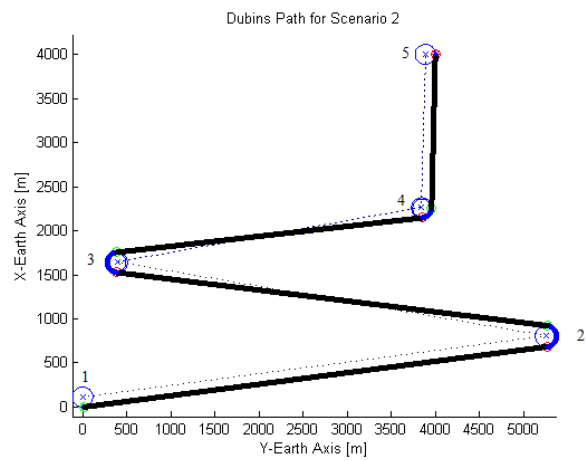

Figure 5-13— Dubins Path for Scenario 2

C. Clothoid Trajectory Generation

The Dubins trajectory generation methodology is frequently applied due to the simple geometry and guaranteed smooth path it produces. However, although a Dubins path is continuous in position and velocity, the angular acceleration is not. For aircraft possessing very fast response rates, following a path with such instantaneous changes in commanded lateral acceleration can be achieved with an adequate level of accuracy. However, the vast majority of UAVs are designed to maximize flight time rather than performance, and thus do not possess such quick responses. Following a Dubins trajectory becomes much more difficult and results in a much higher tracking error, and in some cases, can even cause the aircraft to lose the trajectory entirely. In such situations, a commanded trajectory which is continuous in acceleration as well as velocity, or second-order continuous, is desirable. This easier-to-track path becomes all the more crucial once failure conditions are introduced on the aircraft. The commanded lateral acceleration is proportional to the curvature of the

path, by the following relationship shown in Equation 106, where a represents the lateral acceleration of the aircraft, v is the forward velocity of the aircraft, and κ is the path curvature. Therefore, in order to obtain a path which is second-order continuous, the curvature must be a continuous function.

$$a = v^2\kappa \tag{106}$$

Mimicking the Dubins trajectory generation methodology, it is desired to produce a trajectory which directs the aircraft through a series of desired poses using a piecewise continuous path. Poses are defined in the same manner as for the Dubins trajectory generation, where P_S is the start pose, P_F is the finish pose for a set of maneuvers, X is the position with respect to the Earth X-axis, Y is the position with respect to the Earth Y-axis, ψ is the heading orientation angle with respect to the Earth X-axis, and $\varkappa$ is the maximum curvature:

$$P_S = [X_S\ Y_S\ \psi_S\ \kappa_S] \tag{107}$$

$$P_F = [X_F\ Y_F\ \psi_F\ \kappa_F] \tag{108}$$

However, as illustrated in Figure 5-14 below, is can be seen that generating a path based upon circular arcs will not result in a path of continuous curvature. One method of ensuring a continuous curvature profile, as originally developed by (79), is to substitute Euler curves, or clothoids, in place of circular arcs as used in Dubins trajectory generation. The following derivation of this method will follow the discussion as presented in (80) (81). Clothoids are curves for which the curvature is varied linearly as a function of path length, according to the Fresnel integrals. In this way, the curvature of the path may be made to match at the connections of each segment, resulting in a piecewise trajectory of continuous curvature. This is illustrated in Figure 5-15. It should be noted that between two adjoining sets of poses in a fully developed path, the turn direction from one set of maneuvers transitioning to another set of maneuvers must be consistent in order to maintain this continuous curvature command profile.

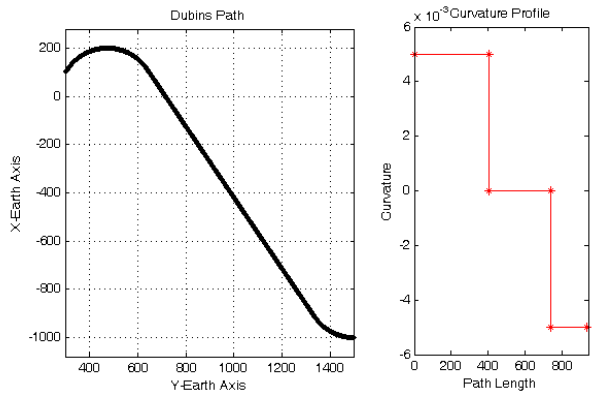

Figure 5-14—Dubins Path with Curvature Profile

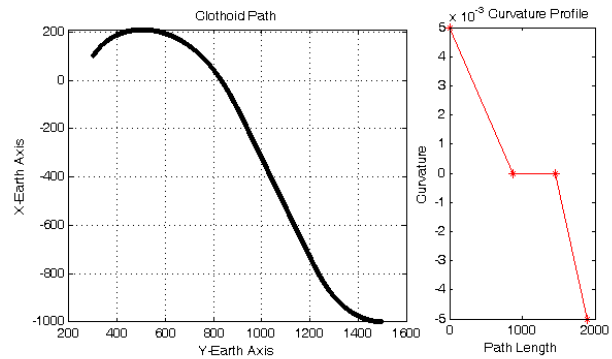

Figure 5-15—Clothoid Path with Curvature Profile

a. Coordinate Axes and Notation

In order to generate the clothoid trajectory between the specified poses, four important coordinate axes will be needed. These are the Earth coordinate axes, denoted by subscript E, start coordinate axes based upon the start pose and denoted by subscript S, finish coordinate axes based upon the finish pose and denoted by subscript F, and the connection coordinate axes based upon the straight line connecting the two clothoid arcs and denoted by subscript A. These coordinate systems are illustrated in Figure 5-16 below.

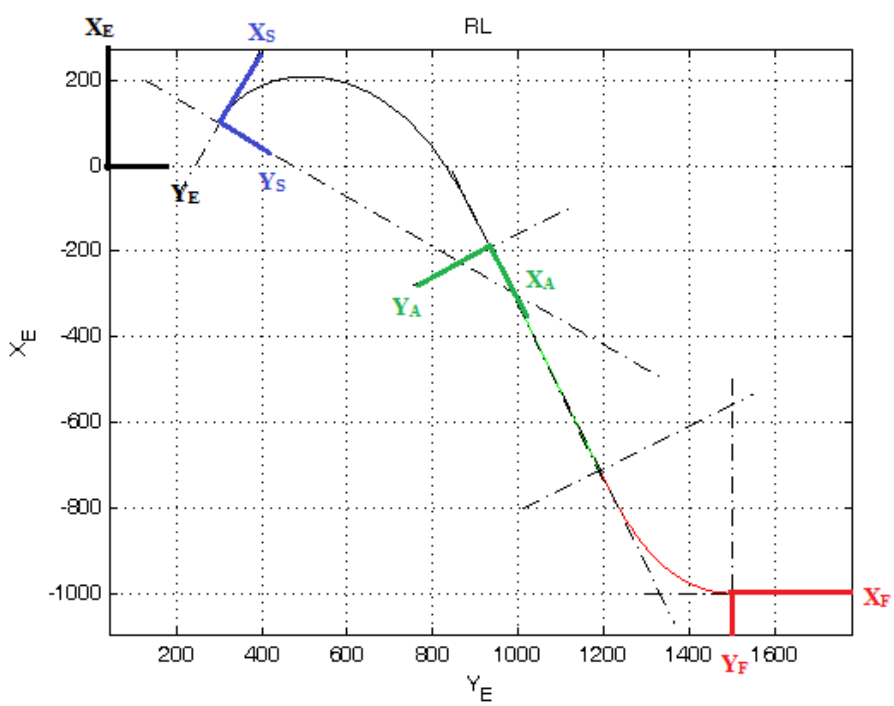

Figure 5-16—Coordinate Systems

Conversion from the start coordinate system to the Earth coordinate system is accomplished using the transformation matrix in Equation 109. Likewise, Equation 110 gives the transition matrix which will allow for coordinate conversion from the finish coordinate system to the Earth coordinate system.

$$R_{ES} = \begin{bmatrix} \cos(\psi_S) & \sin(\psi_S) \\ -\sin(\psi_S) & \cos(\psi_S) \end{bmatrix} \tag{109}$$

$$R_{EF} = \begin{bmatrix} \cos(\psi_F) & \sin(\psi_F) \\ -\sin(\psi_F) & \cos(\psi_F) \end{bmatrix} \tag{110}$$

The connecting reference frame, A, depends on the positioning of the start reference frame. Thus,

$$R_{SA} = \begin{bmatrix} \cos(\phi_S) & \sin(\phi_S) \\ -\sin(\phi_S) & \cos(\phi_S) \end{bmatrix} \tag{111}$$

where ϕ_S represents the arc sweep angle of the start clothoid. Thus, conversion from the connecting coordinate system to the Earth coordinate system is a composite transition matrix, as defined in Equation 112.

$$\begin{aligned} R_{EA} = R_{ES} * R_{SA} &= \begin{bmatrix} \cos(\psi_S) & \sin(\psi_S) \\ -\sin(\psi_S) & \cos(\psi_S) \end{bmatrix} \begin{bmatrix} \cos(\phi_S) & \sin(\phi_S) \\ -\sin(\phi_S) & \cos(\phi_S) \end{bmatrix} \\ &= \begin{bmatrix} \cos(\psi_S + \phi_S) & \sin(\psi_S + \phi_S) \\ -\sin(\psi_S + \phi_S) & \cos(\psi_S + \phi_S) \end{bmatrix} \end{aligned} \tag{112}$$

The final set of coordinate axes relevant to this problem is independent from the above system. This set of coordinate axes is the system of coordinates in which the clothoid arc is generated, according to the Fresnel Integrals, as discussed in the next section.

b. Defining a Clothoid Curve

This section restates the derivation provided by Tsourdos et al. (39) Euler curves are generated from and based upon the solution of the Fresnel integrals, given in the Equations 113 and 114 below.

$$x(h) = \int_0^h \cos(\phi)\, dq \tag{113}$$

$$y(h) = \int_0^h \sin(\phi)\, dq \tag{114}$$

where x and y are coordinates in the clothoid axes. The length of the clothoid arc h and its total sweep angle ϕ are related as given below, in Equation 115. Thus, it is a scaled version of the Fresnel integrals which yields the coordinate path for the clothoid arc. These pertinent relationships are given in Equations 116 and 117.

$$\phi(q) = \int_0^q \kappa \frac{k}{h} dk = \frac{k}{2h} q^2 \tag{115}$$

$$x(h) = \int_0^h \cos(\phi)\, dq = \int_0^h \cos\left(\frac{k}{2h} q^2\right) dq = \int_0^h \cos\left(\frac{k}{2h} q^2\right) dq = C(h) \tag{116}$$

$$y(h) = \int_0^h \cos(\phi)\, dq = \int_0^h \sin\left(\frac{k}{2h} q^2\right) dq = \int_0^h \sin\left(\frac{k}{2h} q^2\right) dq = S(h) \tag{117}$$

In order to more easily evaluate these complex integrals, the following substitution may be made.

$$\bar{q} = \sqrt{\frac{\kappa}{2h}}\, q \tag{118}$$

Since $q \in [0, h]$,

$$\bar{h} = \sqrt{\frac{\kappa}{2h}}\, h = \sqrt{\frac{\kappa h}{2}} \tag{119}$$

$$dq = \sqrt{\frac{2h}{\kappa}}\, d\bar{q} \tag{120}$$

Resulting in the following format for Equations 116 and 117:

$$C(h) = \sqrt{\frac{2h}{\kappa}} \int_0^{\bar{h}} \cos(\bar{q}^2)\, d\bar{q} \tag{121}$$

$$S(h) = \sqrt{\frac{2h}{\kappa}} \int_0^{\bar{h}} \sin(\bar{q}^2)\, d\bar{q} \tag{122}$$

Equations 121 and 122 are referred to as the scaled Fresnel Integrals, which are solved in order to generate the clothoid curve in the clothoid axes, which are defined in Section b.3 later.

b.1 Numerical Solution of the Fresnel Integrals

Since there is no explicit solution to Equations 121 and 122, the scaled Fresnel Integrals, as needed to generate the clothoid curve, these integrals may be solved using a numerical approximation. Recall from the Taylor's Series Expansion that

$$\cos(u) = 1 - \frac{u^2}{2!} + \frac{u^4}{4!} - \frac{u^6}{6!} + \cdots \tag{123}$$

thus,

$$\cos(x^2) = 1 - \frac{x^4}{2!} + \frac{x^8}{4!} - \frac{x^{12}}{6!} + \cdots \tag{124}$$

$$\int \cos(x^2)\,dx = \int 1\,dx - \int \frac{x^4}{2!}dx + \int \frac{x^8}{4!}dx - \int \frac{x^{12}}{6!}dx + \cdots \tag{125}$$

$$\int \cos(x^2)\,dx = x - \frac{x^5}{(2!)(5)} + \frac{x^9}{(4!)(9)} - \frac{x^{13}}{(6!)(13)} + \cdots \tag{126}$$

and the term-wise summation may be written as:

$$C(x) = \int \cos(x^2)\,dx = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!\,(4n+1)} x^{4n+1} \tag{127}$$

Repeating this process for the sine integral we obtain:

$$\sin(u) = u - \frac{u^3}{3!} + \frac{u^5}{5!} - \frac{u^7}{7!} + \cdots \tag{128}$$

$$\sin(x^2) = x^2 - \frac{x^6}{3!} + \frac{x^{10}}{5!} - \frac{x^{14}}{7!} + \cdots \tag{129}$$

$$\int \sin(x^2)dx = \int x^2 dx - \int \frac{x^6}{3!}dx + \int \frac{x^{10}}{5!}dx - \int \frac{x^{14}}{7!}dx + \cdots \tag{130}$$

$$\int \sin(x^2)dx = \frac{1}{3}x^3 - \frac{x^7}{(3!)(7)} + \frac{x^{11}}{(5!)(11)} - \frac{x^{15}}{(7!)(15)} + \cdots \tag{131}$$

$$S(x) = \int \sin(x^2)\,dx = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!\,(4n+3)} x^{4n+3} \tag{132}$$

The summations in Equations 127 and 132 can be carried out for an increasing number of terms until a desired accuracy is met. Once the solutions are sufficiently accurate, the results may be scaled to the appropriate values using the following relationships:

$$C(h) = \sqrt{\frac{2h}{\kappa}}C(x) \qquad 133$$

$$S(h) = \sqrt{\frac{2h}{\kappa}}S(x) \qquad 134$$

b.2 Generating the Clothoid

Using the derived approximation to calculate the Fresnel integrals allows generation, plotting and creation of trajectories based on the clothoid curve. These curves are initially generated within the clothoid coordinate system independent of the Earth axes. The format of the Fresnel Integrals is such that the curves are always produced with the initial point at the origin, a positive sweep or rather as a right-hand turn, and with curvature profile ranging from zero to maximum curvature, similar to those shown in Figure 5-17. Since this is not the only curve configuration which will be needed in this path planning methodology, the raw curve must be converted to the relevant system axes for use in the trajectory. Such conversion involves several processes and will be outlined later.

To initially generate the curve within the clothoid, two parameters need to be defined. These are the total sweep angle of the arc, ϕ, and the maximum curvature of the arc, κ . The total path length of the curve, h_{max}, must then be defined, using Equation 135.

$$h_{max} = \frac{2\phi}{\kappa} \qquad 135$$

The clothoid path may be discretely defined based on incrementing either the path sweep angle or the path length. For either case, the X and Y path components are defined at discrete points using the numerical approximations yielded in Equations 127 and 132, where the input, x, is equal to the square root of the sweep angle as the desired point. Each of these X and Y path components must be scaled using the relationships from Equations 133 and 134. Depending on the parameters of the desired clothoid curve, this process will produce a curve similar to those shown in Figure 5-17.

Due to the relationship between arc length and arc sweep for the clothoid curve, if the total arc sweep angle is altered while the maximum curvature remains constant, the entire curve profile will be altered, as illustrated in Figure 5-17. This is in contrast to the circular arcs used for the Dubins method, for which the curve profile is only dependent upon curvature. This is in contrast to the circular arcs used for the Dubins method, for which the curve profile is only dependent upon curvature.

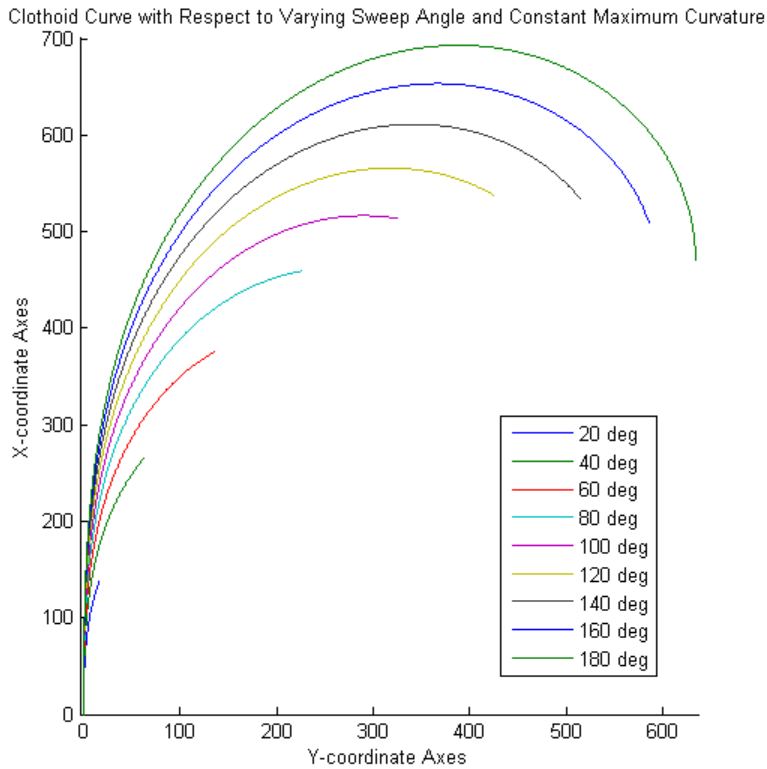

Figure 5-17—Clothoid Arc Profile with Maximum Curvature Held Constant and Sweep Angle Increasing

b.3 Conversion of Clothoid to Earth Coordinate System

Since the clothoid arcs are created externally, it is necessary to convert them to the Earth coordinate system in generating the trajectory. Initially, the clothoid is obtained as a right hand (clockwise or positive) turn arc with the curvature and length varying from 0 to maximum. To generate curves to represent left-hand turns, or negative turns, the signs of the Y coordinates must be negated. This yields the effect seen in Figure 5-18. In this way, the same relationships may be used to generate either a right- or left-hand turn depending upon necessity.

The clothoid generated by solving the Fresnel integrals in the clothoid axes has a curvature varying from zero to a maximum value. This is exactly the profile desired for the finish clothoid arc. However, the start clothoid should have a curvature profile ranging from maximum curvature to 0 curvature as the path length is varied from 0 to maximum. In this way the curvature profile will be smooth throughout the maneuver, as the connecting link is a straight line of zero curvature. To accomplish this, the X-coordinate in the clothoid coordinate system should be negated. Additionally, the order of the coordinates is reversed to yield the order needed for the final trajectory. Furthermore, if a left-hand-turning start clothoid arc is needed, both manipulations may be made. A full diagram of these processes is outlined in Figure 5-18.

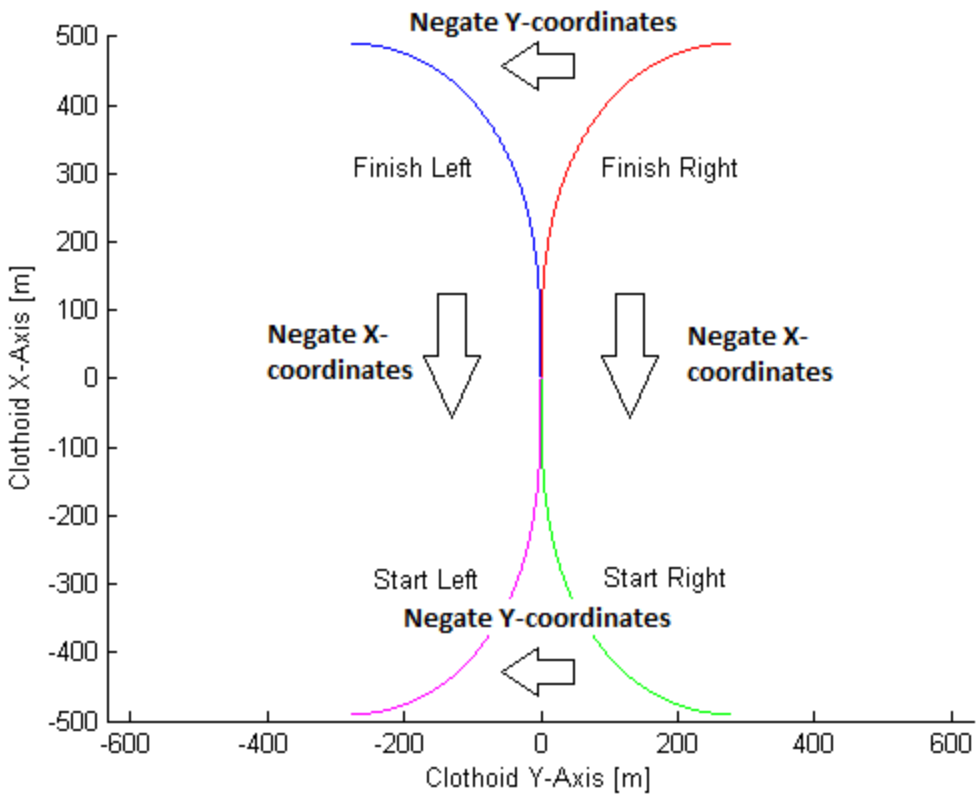

Figure 5-18—Generation Right and Left Turns for Start and Finish Arcs

For both the start and finish clothoids, a translation and rotation are needed to convert the clothoids from the clothoid axes, denoted by subscript C, to the start and finish axes, respectively. For the start clothoid, the curve is rotated by the start heading angle, ψ_S, and the start sweep angle, ϕ_S. Then the curve is translated a distance equal to the difference between the start pose position and the first point in the start curve. This process is given in Equation 136 and shown in Figure 5-19. For the finish clothoid, the curve also undergoes a rotation by the start heading angle and the start sweep angle. Then the clothoid is translated the distance between the goal pose position and the position of the last point in the finish curve. The mathematical representation of this process is given in Equation 137. This effect is shown in Figure 5-20.

$$\begin{bmatrix} X_{CS} \\ Y_{CS} \end{bmatrix}_E = \left(\begin{bmatrix} X_S \\ Y_S \end{bmatrix}_E - \begin{bmatrix} X_{CS_{first}} \\ Y_{CS_{first}} \end{bmatrix}_C \right) + R_{EA} \begin{bmatrix} X_{CS} \\ Y_{CS} \end{bmatrix}_C \qquad 136$$

$$\begin{bmatrix} X_{CF} \\ Y_{CF} \end{bmatrix}_E = \left(\begin{bmatrix} X_F \\ Y_F \end{bmatrix}_E - \begin{bmatrix} X_{CF_{last}} \\ Y_{CF_{last}} \end{bmatrix}_C \right) + R_{EA} \begin{bmatrix} X_{CF} \\ Y_{CF} \end{bmatrix}_C \qquad 137$$

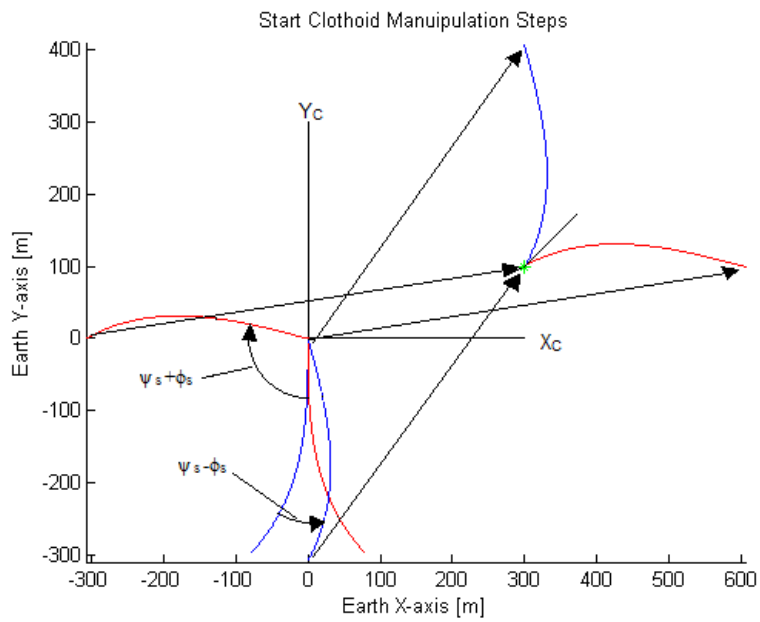

Figure 5-19—Clothoid Translated to Start Coordinates with Right and Left Turns

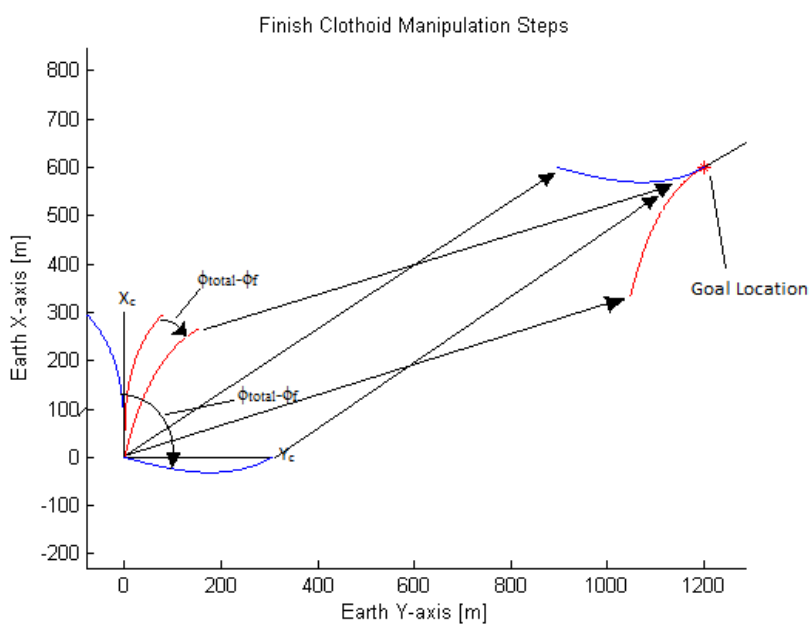

Figure 5-20—Clothoid Translated to Finish Coordinates with Right and Left Turns

In order to generate the connecting vector, the length of the vector is needed, as will be discussed in more detail later. Knowing this magnitude, the vector can be generated at the origin. It is then rotated by the start heading angle and the start sweep angle, and translated the distance to the end of the converted start clothoid curve. If the solution to the system has been properly derived, the endpoint of the connecting vector should coincide with the beginning point of the finish clothoid curve. A finalized path, with trajectory points shown, is given in Figure 5-21, where the start clothoid arc is displayed in black, and the finish clothoid arc is

displayed in red. Note that this set of turn directions does not yield the shortest clothoid path between these poses. Rather this set of turn directions was selected for this example because of the exaggerated nature of the arcs, which clearly illustrate the divergence of the clothoid arc from a circular arc.

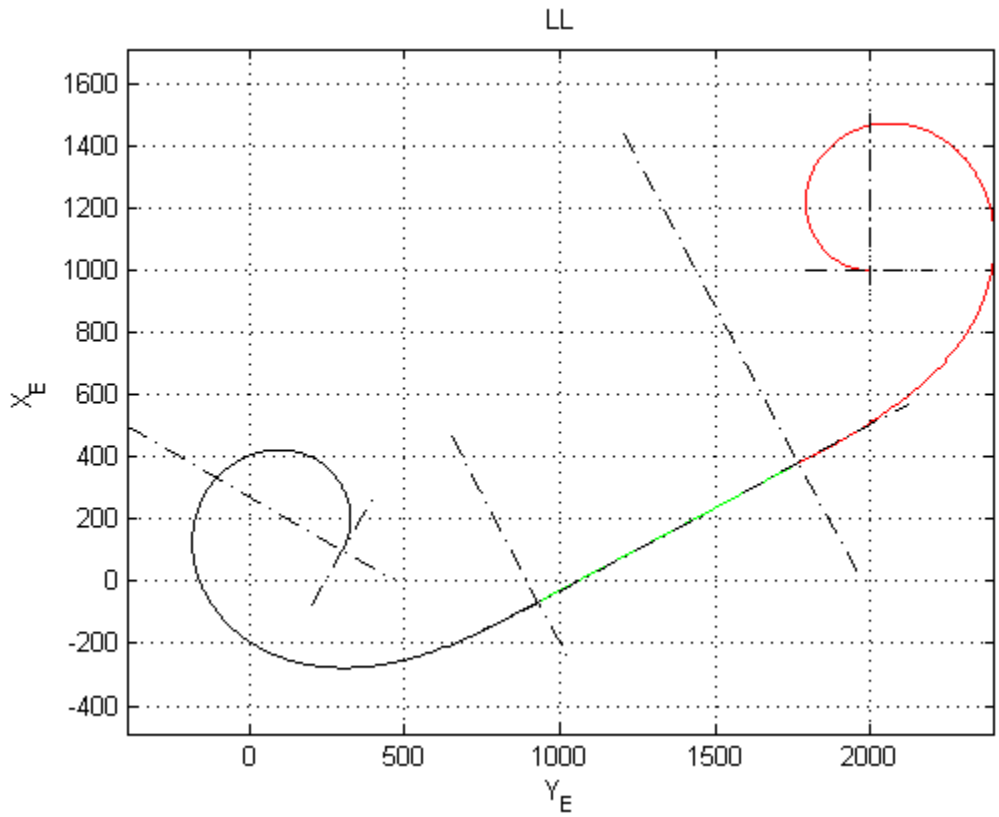

Figure 5-21—Finalized Converted Path Including Start Clothoid, Connecting Segment, and Finish Clothoid

c. Solution of the Vector Equation

In order to find a suitable solution for the path between the two given poses using clothoid arcs in place of circular arcs, the required start and finish sweep angles must be determined. There is no analytical solution to this system; therefore, a numerical solution is determined iteratively. The steps to solving this system will be discussed in the following subsections.

c.1 Definition of Solution Space Quadrants

For any set of two pose vectors, there exist four possible combinations of turns through which a continuous path may be defined, assuming ample spacing between these poses. These are right-right, right-left, left-left, and left-right, where the first direction is the direction of turn for the start clothoid and the second direction is the direction of turn for the finish clothoid. It has been determined that for each set of poses, however, there is a natural selection of turn combinations which will yield the shortest path. In fact, the solution space can be divided into four quadrants based upon the position of the finish pose relative to

the start pose, which, when compared to the sign of the total sweep angle, will yield the natural choice for turn directions.

The natural choices for turn directions, based upon quadrant and sign of ϕ_{total}, are listed in the table below. However, it is important to note that, assuming ample spacing, all four turn direction combinations may be used to generate a flyable path for any set of poses. Lacking ample spacing, one or more of these combinations will be impossible.

Table 5-1—Direction Choices Based Upon Quadrant and Sign of Total Sweep Angle

Quadrant	$\phi_{total} \geq 0$	$\phi_{total} < 0$
I	RR	RL
II	RL	RR
III	LL	LR
IV	LR	LL

In order to define these quadrants and determine to which quadrant a set of poses belongs, the first value which must be found is the magnitude of the total sweep angle, ϕ_{total}. The total sweep angle can be calculated based on the cross and dot products of the start and finish pose tangential unit vectors.

$$[\hat{t}_s]_E = \begin{bmatrix} \cos(\psi_s) \\ \sin(\psi_s) \\ 0 \end{bmatrix}_E \tag{138}$$

$$[\hat{t}_f]_E = \begin{bmatrix} \cos(\psi_f) \\ \sin(\psi_f) \\ 0 \end{bmatrix}_E \tag{139}$$

$$[\hat{t}_s \times \hat{t}_f]_E = \left[\begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ \cos(\psi_s) & \sin(\psi_s) & 0 \\ \cos(\psi_f) & \sin(\psi_f) & 0 \end{vmatrix}\right]_E = \begin{bmatrix} 0 \\ 0 \\ \cos(\psi_s)\sin(\psi_f) - \sin(\psi_s)\cos(\psi_f) \end{bmatrix}_E \tag{140}$$

$$\phi_{total} = \text{sign}(\cos(\psi_s)\sin(\psi_f) - \sin(\psi_s)\cos(\psi_f)) \cdot \cos^{-1}([\hat{t}_s]_E^T \cdot [\hat{t}_f]_E) \tag{141}$$

Once the magnitude and sign of the total sweep angle are known, the quadrant axes may be generated. In principle, these axes are defined based upon the principle of using only the start clothoid to obtain the total sweep angle, and using only the finish clothoid to obtain the total sweep angle. First, look at the case for $\phi_s = \phi_{total}$. Generate the final point of the start clothoid, such that path length, curvature, and sweep angle are all maximum, using Equations 133 and 134, with scaling factors relevant to the start clothoid. Being that this represents the start clothoid, the sign of the x-coordinate should also be reversed. Rotate this point by the angle ϕ_{total}, as instructed in the previous section. This will result in a point referred to as F', noting that it is an abstraction of the finish pose position, point F. This point has been labeled on the

following Figure 5-22 through Figure 5-29. The F' division is generated by constructing a line through point F' at an angle ψ_f, or the black dotted line in the figures below.

To generate the second axis, the point F" is needed. This point is created in a similar manner to F', except that instead $\phi_f = \phi_{total}$ is used to generate the endpoint of the finish clothoid. Since this is a finish clothoid, the sign of the x-coordinate should not be switched. Point F" has also been labeled on the following figures, though it should be noted that this point as well as the full finish clothoid have been translated along the start X-axes in order to make these figures more legible. Location of this point is arbitrary so long as it lies on this line, as the origin point Q is the point of importance. The translation is shown in cyan, with the finish clothoid in red and the start clothoid in green.

Finally, with the quadrant axes defined, the quadrant location of the finish pose can be determined based upon which segment, or quadrant, of the graph it lies in. Note that these graphs are generated based upon the start coordinate axes, thus the finish pose, which is specified in Earth axes, must be converted.

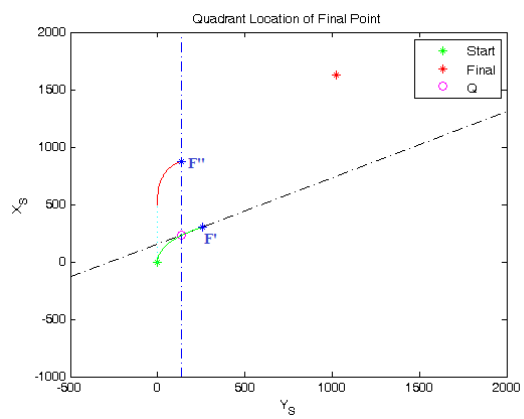

Figure 5-22—Quadrant I with $\phi_{total} \geq 0$

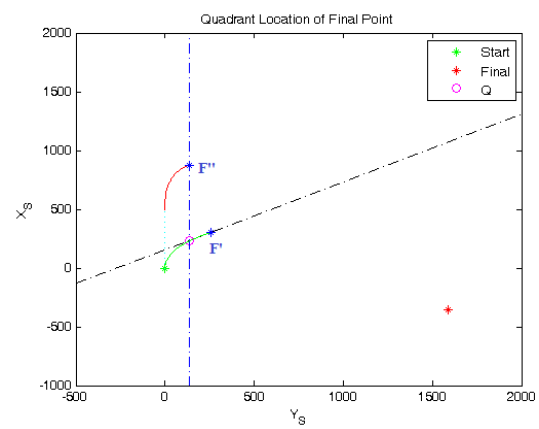

Figure 5-23—Quadrant II with $\phi_{total} \geq 0$

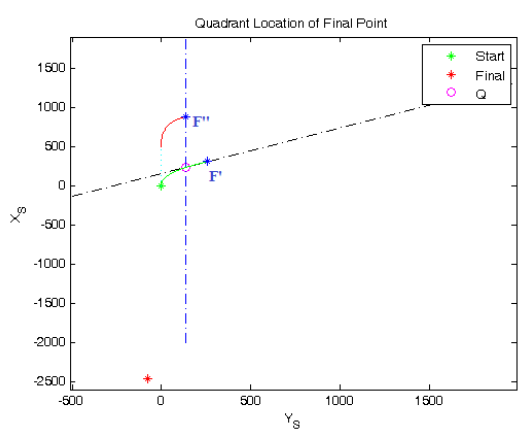

Figure 5-24—Quadrant III with $\phi_{total} \geq 0$

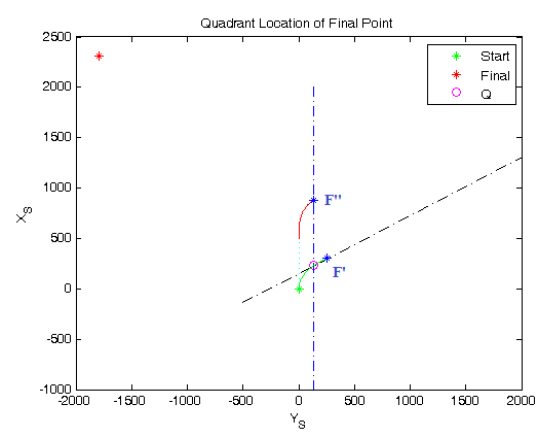

Figure 5-25—Quadrant IV with $\phi_{total} \geq 0$

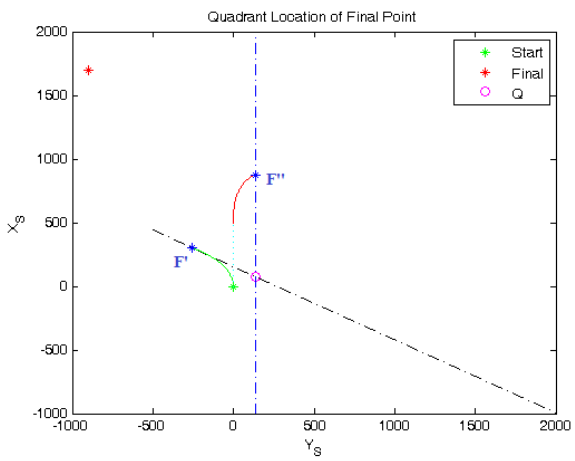

Figure 5-26—Quadrant I with $\phi_{total} < 0$

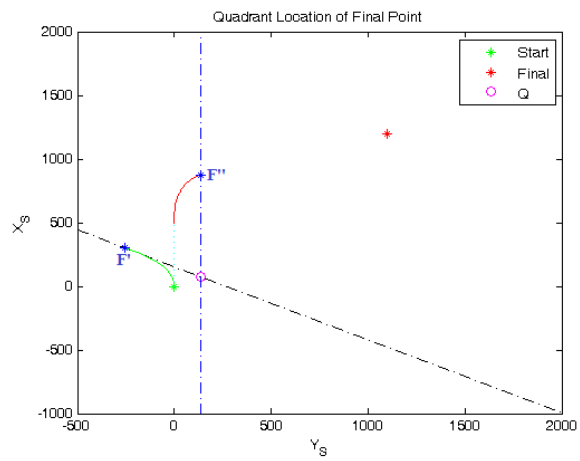

Figure 5-27—Quadrant II with $\phi_{total} < 0$

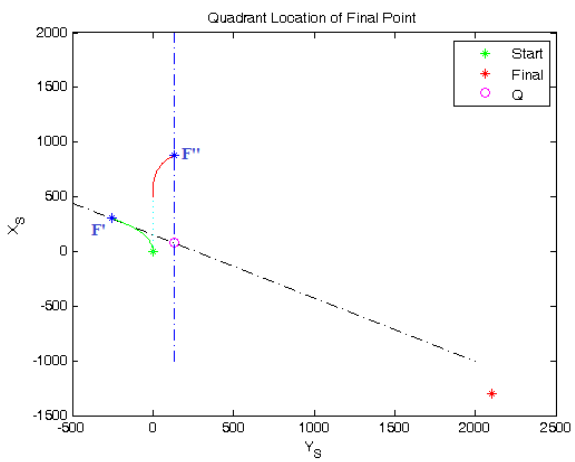

Figure 5-28—Quadrant III with $\phi_{total} < 0$

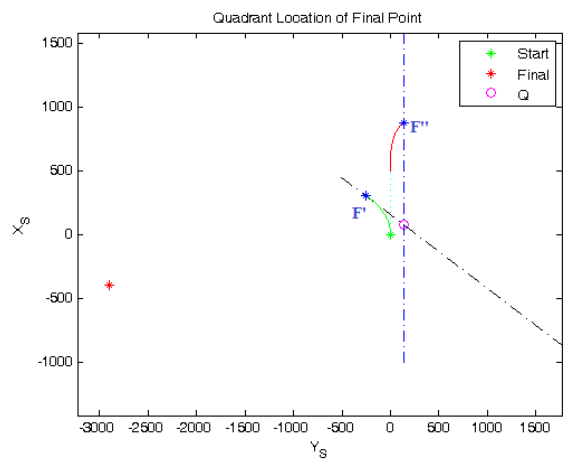

Figure 5-29—Quadrant IV with $\phi_{total} < 0$

Based upon the quadrant location and sign of ϕ_{total}, the relationship among ϕ_s, ϕ_f, and ϕ_{total} will vary. These relationships have been derived and confirmed, and are provided in Table 5-2 below.

Table 5-2—ϕ_f Expression Based Upon Quadrant, Sign of ϕ_{total}, and Turn Directions

Quadrant	Sign ϕ_{total}	Relationship for ϕ_f Based Upon Turn Directions			
		RR	RL	LL	LR
I	+	$\phi_{total} - \phi_s$	$2\pi - \phi_{total} + \phi_s$	$4\pi - \phi_{total} - \phi_s$	$-2\pi + \phi_{total} + \phi_s$
	-	$4\pi + \phi_{total} - \phi_s$	$-2\pi - \phi_{total} + \phi_s$	$-\phi_{total} - \phi_s$	$2\pi + \phi_{total} + \phi_s$
II	+	$2\pi + \phi_{total} - \phi_s$	$\phi_s - \phi_{total}$	$2\pi - \phi_{total} - \phi_s$	$\phi_{total} + \phi_s$
	-	$2\pi + \phi_{total} - \phi_s$	$\phi_s - \phi_{total}$	$2\pi - \phi_{total} - \phi_s$	$\phi_{total} + \phi_s$
III	+	$2\pi + \phi_{total} - \phi_s$	$\phi_s - \phi_{total}$	$2\pi - \phi_{total} - \phi_s$	$\phi_{total} + \phi_s$
	-	$2\pi + \phi_{total} - \phi_s$	$\phi_s - \phi_{total}$	$2\pi - \phi_{total} - \phi_s$	$\phi_{total} + \phi_s$
IV	+	$2\pi + \phi_{total} - \phi_s$	$\phi_s - \phi_{total}$	$2\pi - \phi_{total} - \phi_s$	$\phi_{total} + \phi_s$
	-	$2\pi + \phi_{total} - \phi_s$	$\phi_s - \phi_{total}$	$2\pi - \phi_{total} - \phi_s$	$\phi_{total} + \phi_s$

c.2 Definition of Solution Vectors

In order to find a smooth, flyable path between the start pose and finish pose, the following vector relationship must be satisfied:

$$\vec{p} - \overrightarrow{\rho_S} + \overrightarrow{\rho_F} = -\overrightarrow{\alpha_S} + \overrightarrow{a_c} + \overrightarrow{\alpha_F} = \vec{c} \qquad 142$$

This equation is illustrated below in Figure 5-30. Using these relationships, we can rearrange this crucial equation into the more usable format given in Equation 143.

$$\vec{p} - \overrightarrow{\rho_S} + \overrightarrow{\rho_F} + \overrightarrow{\alpha_S} - \overrightarrow{a_c} - \overrightarrow{\alpha_F} = 0 \qquad 143$$

Since this equation is not readily solvable, this format lends itself to the application of a numerical root-finding method. Application of this will be discussed in the next section. In order to solve this vector equation, it becomes necessary to rewrite the individual vectors in terms of their components with respect to the same coordinate axes. It is most feasible to relate the vector components with respect to the start coordinate axes. Definition of these components will be given below.

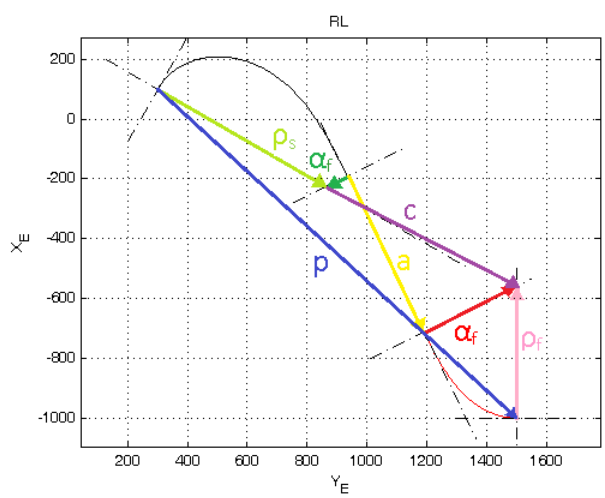

Figure 5-30—Vector Geometry

Vector $\vec{p}$ represents the vector extending from the start pose to the finish pose. We can most readily define this vector relative to the Earth axes as the vector difference of the position vectors of each pose.

$$[\vec{p}]_E = [\vec{r}_{OF}]_E - [\vec{r}_{OS}]_E \qquad 144$$

Where:

$$[\vec{r}_{OS}]_E = \begin{bmatrix} x_{s_E} \\ y_{s_E} \end{bmatrix} \qquad 145$$

and

$$[\vec{r}_{OF}]_E = \begin{bmatrix} x_{f_E} \\ y_{f_E} \end{bmatrix} \tag{146}$$

However, it is desired to write this vector in terms of the start coordinate axes, as in the following equation.

$$\begin{aligned}[\vec{p}]_S = R_{ES}^T[\vec{p}]_E &= \begin{bmatrix} \cos(\psi_s) & -\sin(\psi_s) \\ \sin(\psi_s) & \cos(\psi_s) \end{bmatrix} \begin{bmatrix} x_{f_E} - x_{s_E} \\ y_{f_E} - y_{s_E} \end{bmatrix}_E = \\ &= \begin{bmatrix} \cos(\psi_s)(x_{f_E} - x_{s_E}) + \left(-\sin(\psi_s)\right)(y_{f_E} - y_{s_E}) \\ \sin(\psi_s)(x_{f_E} - x_{s_E}) + \left(\cos(\psi_s)\right)(y_{f_E} - y_{s_E}) \end{bmatrix}_S \end{aligned} \tag{147}$$

Vector $\overrightarrow{\rho_S}$ is in the start coordinate axes as:

$$[\overrightarrow{\rho_S}]_S = \begin{bmatrix} 0 \\ \pm\rho_s \end{bmatrix}_S \tag{148}$$

where the magnitude of the vector is defined in Equation 149, and the sign of the component is assigned based upon the desired start clothoid turn direction.

$$\rho_s = \frac{C\left(\frac{2\phi_s}{\kappa_s}\right)}{\sin(\phi_s)} \tag{149}$$

where $C\left(\frac{2\phi_s}{\kappa_s}\right)$ is calculated according to Equation 127.

Vector $\overrightarrow{\alpha_S}$ is defined in the connecting vector axes as:

$$[\overrightarrow{\alpha_S}]_A = \begin{bmatrix} 0 \\ \pm\alpha_s \end{bmatrix}_A \tag{150}$$

Where:

$$\alpha_s = S\left(\frac{2\phi_s}{\kappa_s}\right) + \frac{C\left(\frac{2\phi_s}{\kappa_s}\right)}{\tan(\phi_s)} \tag{151}$$

$C\left(\frac{2\phi_s}{\kappa_s}\right)$ and $S\left(\frac{2\phi_s}{\kappa_s}\right)$ are defined according to Equations 127 and 132, and the sign of the component is assigned based upon the desired start clothoid turn direction. This yields the relationship below.

$$[\overrightarrow{\alpha_S}]_S = R_{SA}[\overrightarrow{\alpha_S}]_A = \begin{bmatrix} \cos(\phi_s) & \sin(\phi_s) \\ -\sin(\phi_s) & \cos(\phi_s) \end{bmatrix} \begin{bmatrix} 0 \\ \pm\alpha_s \end{bmatrix}_A = \begin{bmatrix} \pm\alpha_s\left(-\sin(\phi_s)\right) \\ \pm\alpha_s\left(\cos(\phi_s)\right) \end{bmatrix}_S \tag{152}$$

Conveniently, vector $\overrightarrow{\alpha_F}$ is also defined in the connection coordinate axes. Thus, a similar procedure can also be followed for deriving $[\overrightarrow{\alpha_F}]_S$.

$$[\overrightarrow{\alpha_F}]_A = \begin{bmatrix} 0 \\ \pm\alpha_f \end{bmatrix}_A \tag{153}$$

where,

$$\alpha_f = S\left(\frac{2\phi_f}{\kappa_f}\right) + \frac{C\left(\frac{2\phi_f}{\kappa_f}\right)}{\tan(\phi_f)} \tag{154}$$

and likewise, the sign is assigned based on the desired finish clothoid turn direction. This yields:

$$[\overrightarrow{\alpha_F}]_S = R_{SA}[\overrightarrow{\alpha_F}]_A = \begin{bmatrix} \cos(\phi_s) & \sin(\phi_s) \\ -\sin(\phi_s) & \cos(\phi_s) \end{bmatrix} \begin{bmatrix} 0 \\ \pm\alpha_f \end{bmatrix}_A = \begin{bmatrix} \pm\alpha_f(-\sin(\phi_s)) \\ \pm\alpha_f(\cos(\phi_s)) \end{bmatrix}_S \tag{155}$$

Vector $\overrightarrow{\rho_f}$ must be converted from the finish coordinate axes to the start coordinate axes. In finish coordinates:

$$[\overrightarrow{\rho_F}]_F = \begin{bmatrix} 0 \\ \pm\rho_f \end{bmatrix}_F \tag{156}$$

where:

$$\rho_f = \frac{C\left(\frac{2\phi_f}{\kappa_f}\right)}{\sin(\phi_f)} \tag{157}$$

and the sign of the component is assigned based upon the desired finish clothoid turn direction. In order to convert these vector coordinates from the finish axes to the start axes, two transition matrices are needed. These are the transition matrix from finish to connection reference frame, given below in Equation 158, and the transition matrix from connection to start reference frame given above in Equation 111.

$$R_{AF} = \begin{bmatrix} \cos(\phi_f) & \sin(\phi_f) \\ -\sin(\phi_f) & \cos(\phi_f) \end{bmatrix} \tag{158}$$

However, the multiplication of these transition matrices can be simplified by noting that the total rotation from start to finish coordinates is ϕ_{total}. This yields a single transition matrix,

$$R_{SF} = \begin{bmatrix} \cos(\phi_{tot}) & \sin(\phi_{tot}) \\ -\sin(\phi_{tot}) & \cos(\phi_{tot}) \end{bmatrix} \tag{159}$$

which can then be used to convert $[\overrightarrow{\rho_F}]_F$ to $[\overrightarrow{\rho_F}]_S$, as shown in Equation 160.

$$[\overrightarrow{\rho_F}]_S = R_{SF}[\overrightarrow{\rho_F}]_F = \begin{bmatrix} \cos(\phi_{tot}) & \sin(\phi_{tot}) \\ -\sin(\phi_{tot}) & \cos(\phi_{tot}) \end{bmatrix} \begin{bmatrix} 0 \\ \pm\rho_f \end{bmatrix}_F = \begin{bmatrix} \pm\rho_f(-\sin(\phi_{tot})) \\ \pm\rho_f(\cos(\phi_{tot})) \end{bmatrix}_S \quad 160$$

Slightly different, vector $\overrightarrow{a_c}$ is also defined in terms of the connection axes, as given below in Equation 161.

$$[\overrightarrow{a_c}]_A = \begin{bmatrix} a \\ 0 \end{bmatrix}_A \quad 161$$

By inspection of the vector geometry, the magnitude of the connecting vector a can be defined as:

$$a = \sqrt{c^2 - (\pm\rho_f - \pm\rho_s)^2} \quad 162$$

where the signs are selected to correspond with the desired start and finish turn directions, and c refers to the magnitude of $\vec{c}$. This yields start coordinate components:

$$[\overrightarrow{a_c}]_S = R_{SA}[\overrightarrow{a_c}]_A = \begin{bmatrix} \cos(\phi_s) & \sin(\phi_s) \\ -\sin(\phi_s) & \cos(\phi_s) \end{bmatrix} \begin{bmatrix} a \\ 0 \end{bmatrix}_A = \begin{bmatrix} \cos(\phi_s)\sqrt{c^2 - (\pm\rho_f - \pm\rho_s)^2} \\ \sin(\phi_s)\sqrt{c^2 - (\pm\rho_f - \pm\rho_s)^2} \end{bmatrix}_S \quad 163$$

Inserting the above derived relationships for the start coordinate vector components into vector Equation 143, yields the following:

$$\begin{aligned} \begin{bmatrix} f(\phi_s) \\ g(\phi_s) \end{bmatrix}_S &= \begin{bmatrix} \cos(\psi_s)(x_{f_E} - x_{s_E}) + (-\sin(\psi_s))(y_{f_E} - y_{s_E}) \\ \sin(\psi_s)(x_{f_E} - x_{s_E}) + (\cos(\psi_s))(y_{f_E} - y_{s_E}) \end{bmatrix}_S - \begin{bmatrix} 0 \\ \pm\rho_s \end{bmatrix}_S \\ &+ \begin{bmatrix} \pm\rho_f(-\sin(\phi_{tot})) \\ \pm\rho_f(\cos(\phi_{tot})) \end{bmatrix}_S + \begin{bmatrix} \pm\alpha_s(-\sin(\phi_s)) \\ \pm\alpha_s(\cos(\phi_s)) \end{bmatrix}_S \\ &- \begin{bmatrix} \cos(\phi_s)\sqrt{c^2 - (\pm\rho_f - \pm\rho_s)^2} \\ \sin(\phi_s)\sqrt{c^2 - (\pm\rho_f - \pm\rho_s)^2} \end{bmatrix}_S - \begin{bmatrix} \pm\alpha_f(-\sin(\phi_s)) \\ \pm\alpha_f(\cos(\phi_s)) \end{bmatrix}_S = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned} \quad 164$$

This vector equation, when written as components, results in a system of 2 equations with one unknown, ϕ_s, which can now be solved using a suitable numerical method. In order to find a flyable, smooth solution, both of these equations must be satisfied simultaneously.

c.3 Iterative Solution Using Modified Bisection Method

When solving the vector equation derived in the previous section, it is necessary to utilize numerical means to determine the appropriate combination of start and finish clothoid sweep angles which will yield a continuous and flyable path. In order to do so, a modified bisection algorithm is implemented.

c.3.1 Traditional Bisection Numerical Root-Finding Method

The bisection algorithm is a common numerical method for finding the roots of an equation. It is known as a bracketing method, since in order to obtain a solution, a bound in which exactly one root exists must be known a priori. In a typical bisection algorithm, a known but usually difficult-to-solve equation is arranged in the format:

$$f(x) = 0 \tag{165}$$

This function may have multiple roots over its full range, however, the algorithm needs to be bounded to a limited range in order to find only a particular root at a time. Thus, the following criteria must be met by the initial bounds:

$$f(a) * f(b) < 0 \tag{166}$$

where a is the lower bound, b is the upper bound, and f(x) is continuous for [a, b]. In order to determine the root, the bound is divided at the center and the value of the function is assessed. If the value of the function changes signs in the left half of the bound, the center point becomes the new upper bound and the old upper bound is no longer considered. Likewise, if the value of the function changes signs for the right half of the bound, the left half is neglected and the center point becomes the new lower bound. This process continues until the value of the function is acceptably small. The pseudo-code for this process is included as Pseudo-Code 3 in Appendix A.

c.3.2 Modified Bisection Method for Multiple Equations

The traditional root-finding method works well when obtaining the roots for a single function, however, this situation requires simultaneous satisfaction of a system of two equations: one for the x-coordinate and one for the y-coordinate of the summation of the vector components. Thus, the bisection method above was modified to adjust for this.

In a typical clothoid system, both the x-equation and y-equation will have multiple roots. Conveniently, however, only one of these roots will be common to both equations. This is illustrated by an example system in Figure 5-31. In order to find the common root, in this case, the bounds will be defined on the start sweep angle but the value of the function will be assessed for both the x- and y-equations. If a sign changes over a given bound for either or both equations this bound is retained as possibly containing the solution. This bound is then subdivided, and the process is repeated. Otherwise, if no sign change is present, all bounds are retained with the assumption than an even number of bounds is present in each. This assumption is based upon the fact the bounds for the solution are known definitely and also known to contain multiple solutions for each equation; this procedure has been demonstrated to work well in practice. This solution logic is outlined in Pseudo-Code 4 in Appendix A.

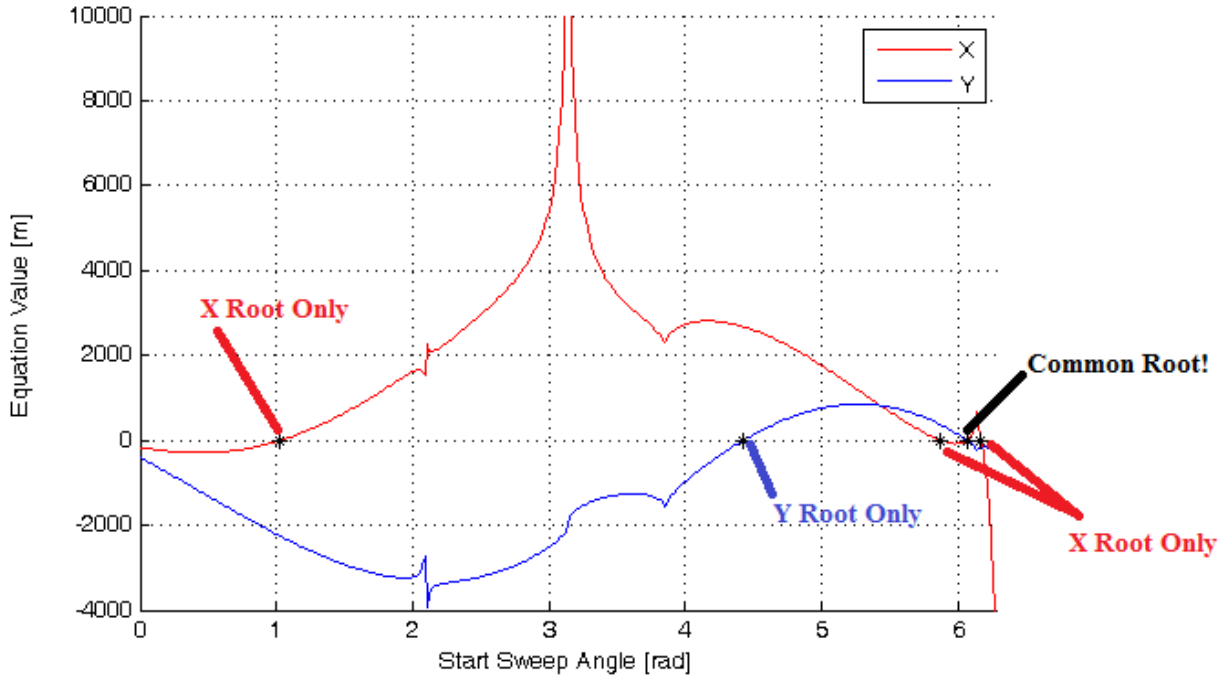

Figure 5-31—Example of Vector Component X and Y Equations

d. Clothoid Planner Demonstration

Combining the techniques discussed, it is possible to generate a full path through a series of poses. The full pseudo-code for the clothoid Path Planner is given in Appendix A. As per the discussion in Chapter 5. c.1, of the four solution quadrants, it happens that when a path is generated for a set of poses, the shortest path will be generated by the combination of turn directions dictated by the sign convention of the quadrant to which the set of poses corresponds. This will be illustrated in the following examples for all four quadrants.

For Quadrant I, the optimal choice of turn directions is right-right. The paths below in Figure 5-32 through Figure 5-35 are generated for the same set of poses, but based on the four different turn direction combinations. As shown in Table 5-3, the predicted path choice indeed produces the shortest path. A similar procedure has been performed for a set of poses corresponding to each of the remaining 3 quadrants. Paths generated for Quadrant II are shown in Figure 5-36 through Figure 5-39. Paths for Quadrant III are illustrated by Figure 5-40 through Figure 5-43, and Figure 5-44 through Figure 5-47 shown the paths generated for the Quadrant IV set of poses. The resulting path lengths are organized in Table 5-4 through Table 5-6, respectively.

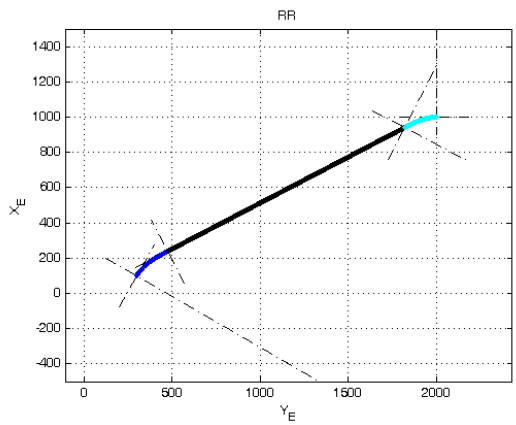

Figure 5-32—Quadrant I, Right-Right Path

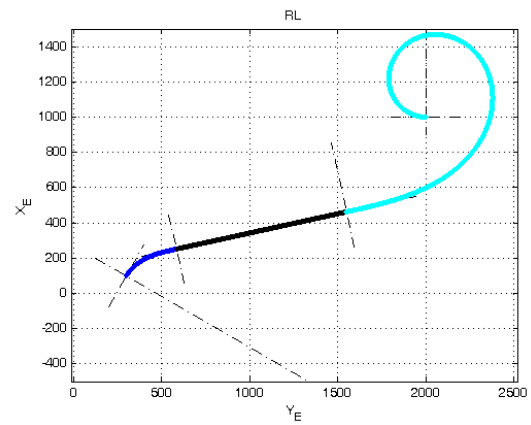

Figure 5-33—Quadrant I, Right-Left Path

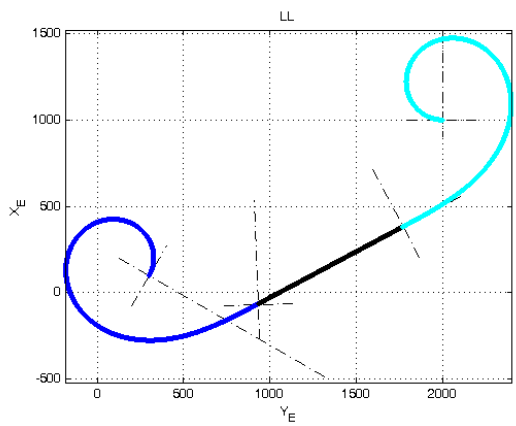

Figure 5-34—Quadrant I, Left-Left Path

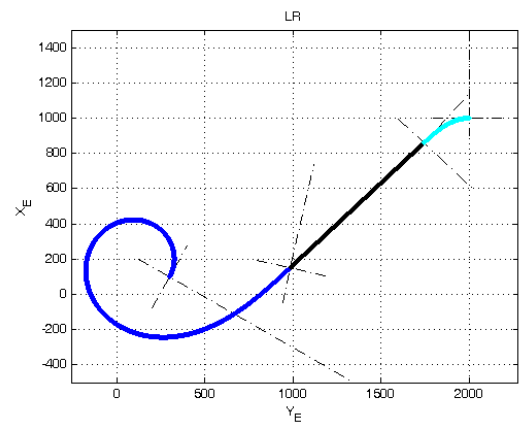

Figure 5-35—Quadrant I, Left-Right Path

Table 5-3—Quadrant I Resulting Path Lengths

QUADRANT I	
Turn Direction Combination	**Path Length [m]**
Right-Right	**1935.1**
Right-Left	3741.8
Left-Left	5551.8
Left-Right	3726.9

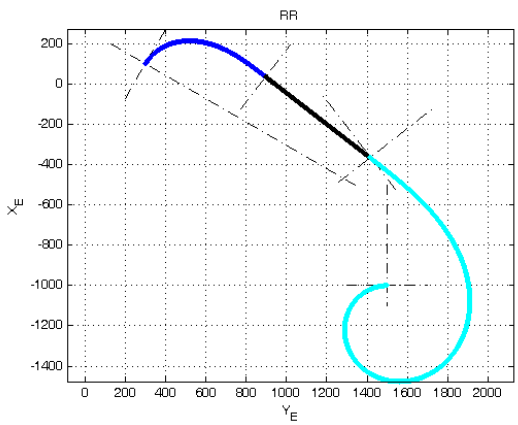

Figure 5-36—Quadrant II, Right-Right Path

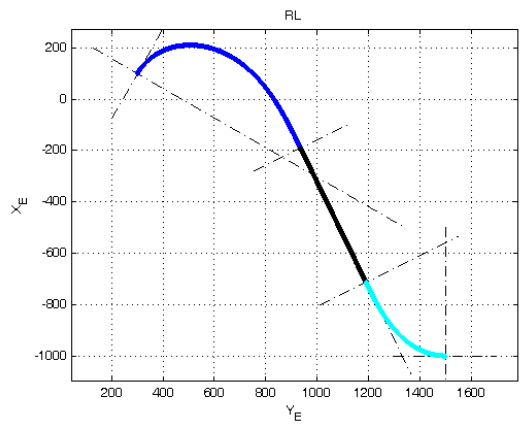

Figure 5-37—Quadrant II, Right-Left Path

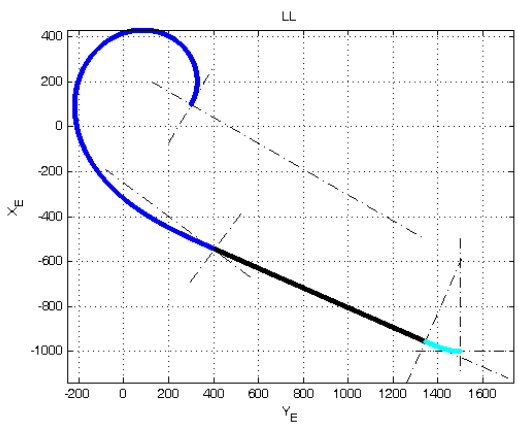

Figure 5-38—Quadrant II, Left-Left Path

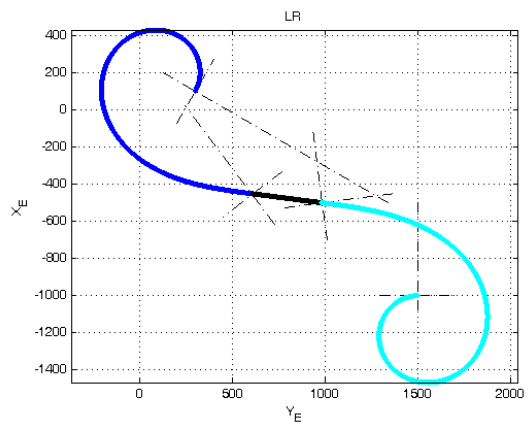

Figure 5-39—Quadrant II, Left-Right Path

Table 5-4—Quadrant II Resulting Path Lengths

QUADRANT II	
Turn Direction Combination	**Path Length [m]**
Right-Right	3584.0
Right-Left	**1893.8**
Left-Left	3118.0
Left-Right	4874.9

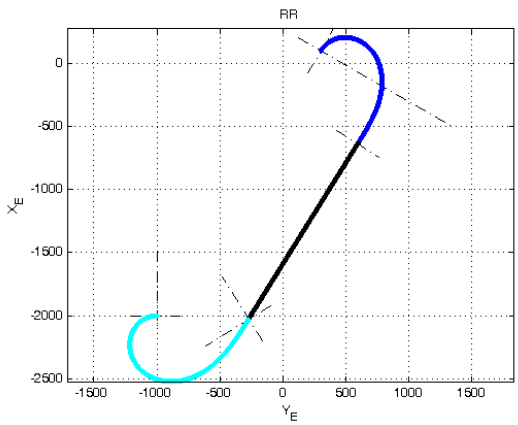

Figure 5-40—Quadrant III, Right-Right Path

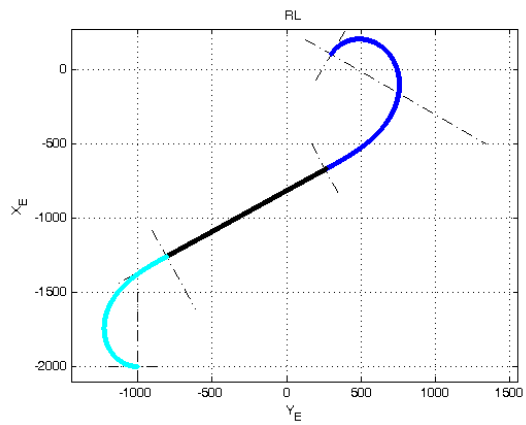

Figure 5-41—Quadrant III, Right-Left Path

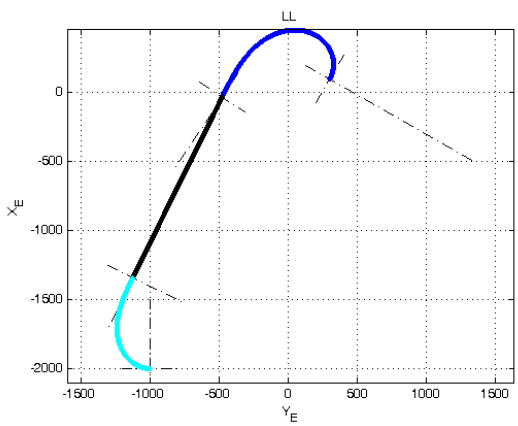

Figure 5-42—Quadrant III, Left-Left Path

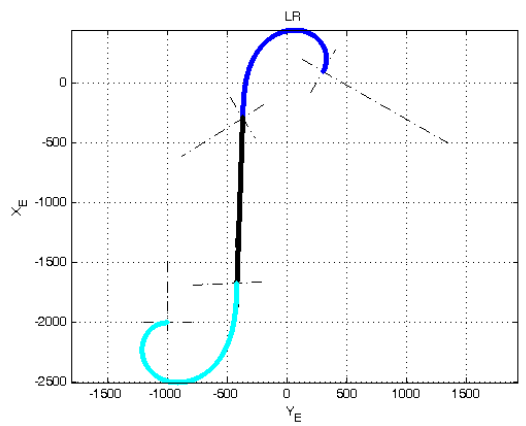

Figure 5-43—Quadrant III, Left-Right Path

Table 5-5—Quadrant III Resulting Path Lengths

QUADRANT III	
Turn Direction Combination	**Path Length [m]**
Right-Right	4578.5
Right-Left	3756.9
Left-Left	**3555.9**
Left-Right	4711.1

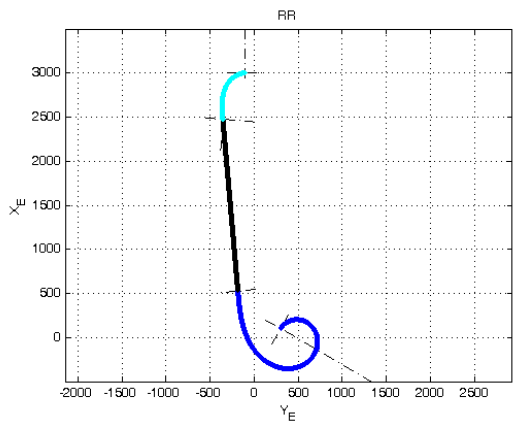

Figure 5-44—Quadrant IV, Right-Right Path

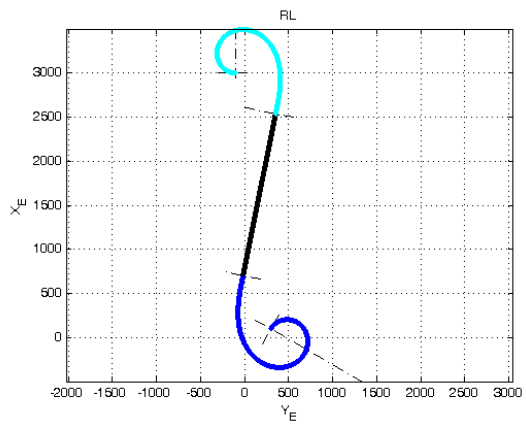

Figure 5-45—Quadrant IV, Right-Left Path

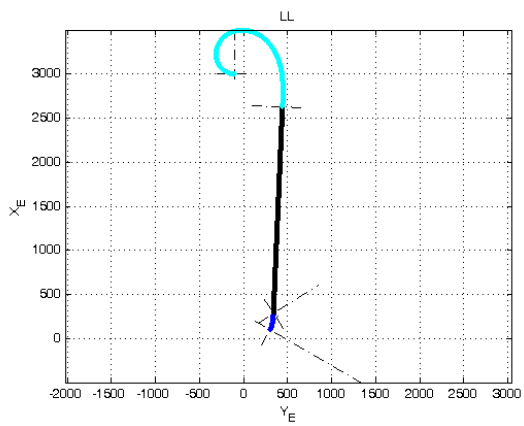

Figure 5-46—Quadrant IV, Left-Left Path

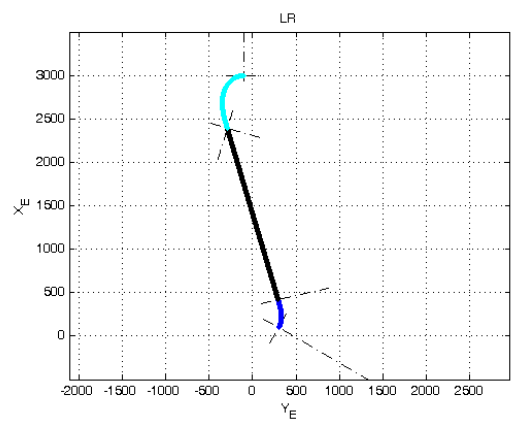

Figure 5-47—Quadrant IV, Left-Right Path

Table 5-6—Quadrant IV Resulting Path Lengths

QUADRANT IV	
Turn Direction Combination	**Path Length [m]**
Right-Right	4882.1
Right-Left	6216.5
Left-Left	4435.4
Left-Right	**3133.8**

It can be seen in the above results that for each pose quadrant situation the predicted turn direction combination does produce the shortest path combination. Since the paths are identical, though reversed, for the case when $\phi_{total} < 0$, illustrating these properties for $\phi_{total} \geq 0$ is sufficient to conclude that the

quadrant formulation correctly chooses the shortest path without the need to calculate all four paths. Knowing this is highly important, as it cuts the computational load of the method to one-fourth, making the clothoid trajectory generation method viable for online recomputation. The following path contains an example of each turn direction combination. Figure 5-48 contains a plotting of the entire composite trajectory. It should be noted that this trajectory was generated using a waypoint scheme which observes the pose but turn directions are not specified. It happens, however, that for this situation we obtain the same resulting path which would be produced for a point-of-interest scheme intended to observe the locations marked by the circles. It can be seen that without proper turn direction selection, the length of this path could become significantly longer. Figure 5-49 shows the path followed by the WVU YF-22 UAV simulation model, using the outer-loop non-linear dynamic inversion trajectory tracking controller.

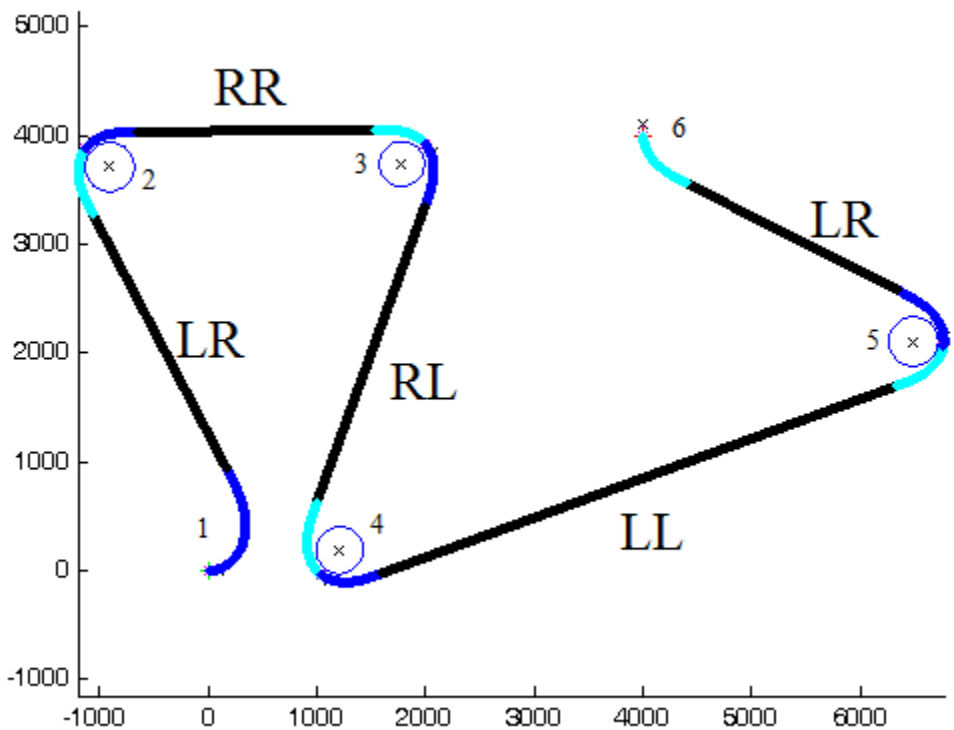

Figure 5-48—Clothoid Full Trajectory with Each Turn Combination

Figure 5-49—Full Clothoid Path with Aircraft Tracking

To assess the calculation overhead of this algorithm, two basic scenarios were tested for computation time. These are shown below in Figure 5-50 and Figure 5-51. For scenario 1 the average calculation time was 4.1684 seconds and the average calculation time for scenario 2 was 5.1325 seconds.

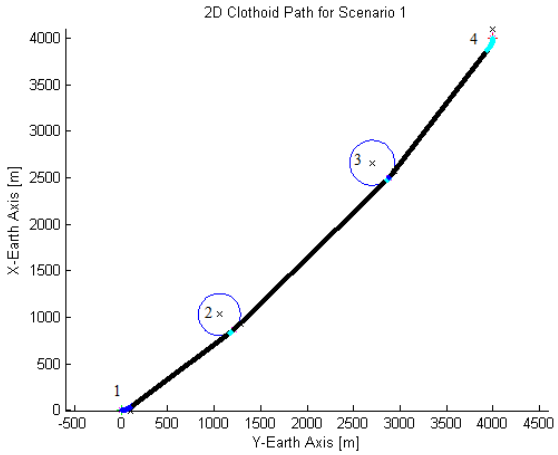

Figure 5-50—Clothoid Path for Scenario 1

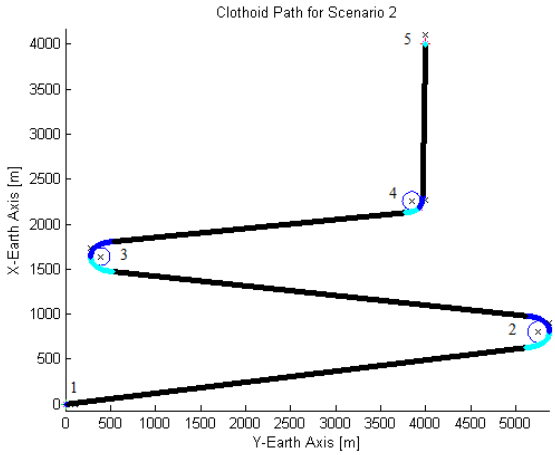

Figure 5-51— Clothoid Path for Scenario 2

D. Hybrid Clothoid and Dubins Trajectory Generation

One of the drawbacks to generating a clothoid trajectory occurs when any of the pose positions lie too close together. If this occurs, the solution for the connecting vector results in a negative magnitude, which is obviously invalid. If this occurs, it is not possible to generate a valid clothoid trajectory between the two poses. However, while the path loses the benefit of a continuous curvature profile, it is always possible to generate a valid Dubins trajectory between 2 poses. In the hybrid Clothoid-Dubins trajectory generation methodology, the clothoid trajectory profile is used in all valid situations. However, rather than returning no path or requiring the user to modify the desired poses (which in a real-world application may not be possible), the Dubins path between the two poses will be utilized in the event that the clothoid profile cannot be constructed in the available space. An example of such a trajectory is displayed in Figure 5-52. In this path, the third set of poses, indicated on the figure, are too close together to allow for the Clothoid-profile trajectory to be possible. Thus, the Dubins-profile trajectory was substituted for this set of poses only. A formulation of this hybrid trajectory generation algorithm is utilized with the Immunity-Based Evolutionary Pose Optimization algorithm, discussed later in Chapter 7.

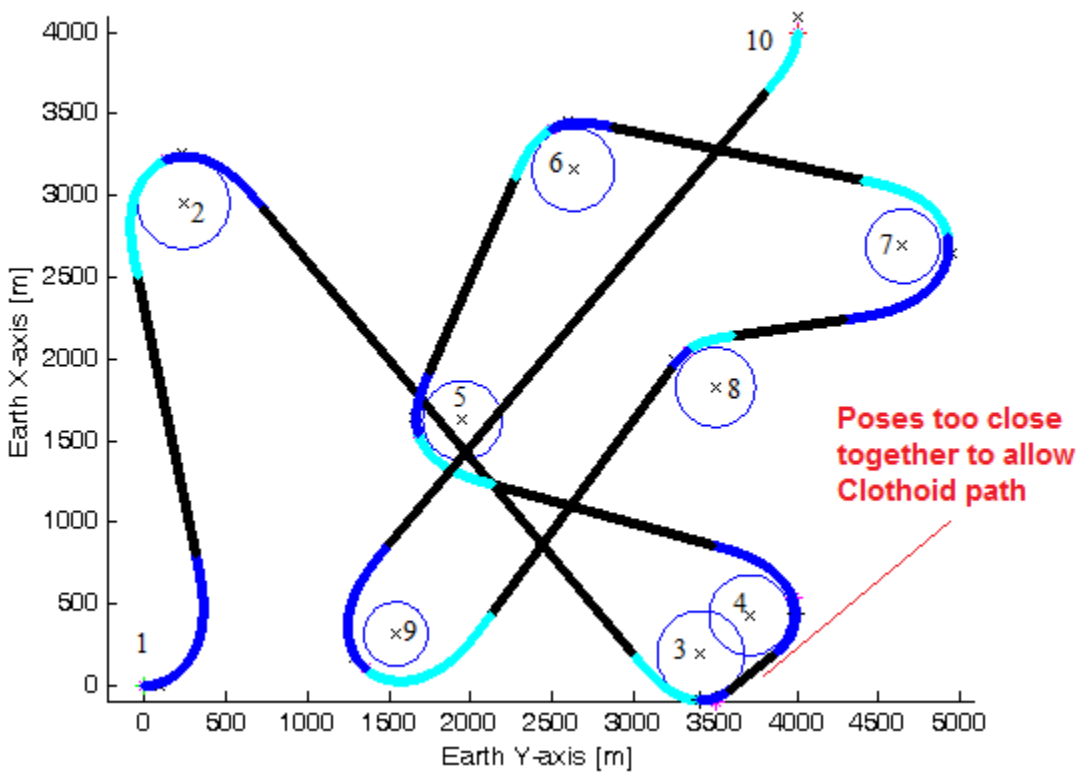

Figure 5-52—Hybrid Clothoid-Dubins Trajectory

Since this method is a combination of the 2D Dubins and 2D clothoid path planners, the computational overhead will depend on the computational time of the clothoid planner, as this path is checked first. For consistency, the computation time for this method was still assessed for the following 2 scenarios. These are shown below in Figure 5-53 and Figure 5-54. For scenario 1, the average calculation time was 4.2647 seconds and the average calculation time for scenario 2 was 5.2282 seconds.

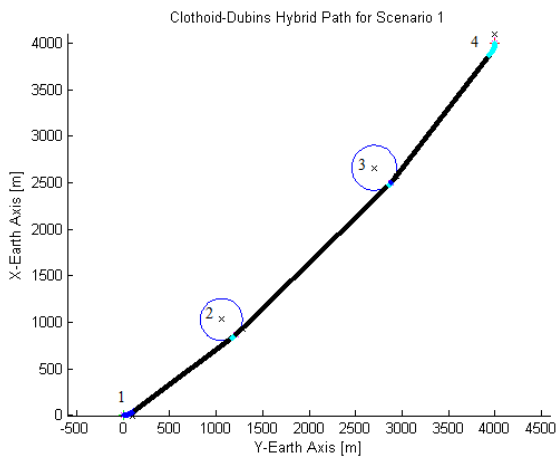

Figure 5-53—Clothoid-Dubins Hybrid Field Path for Scenario 1

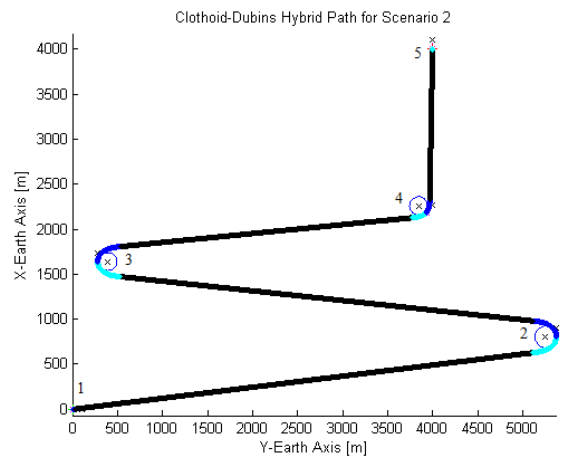

Figure 5-54— Clothoid-Dubins Hybrid Path for Scenario 2

Chapter 6. 3-DIMENSIONAL METHODS

Aircraft are inherently 3-dimensional systems. As such, to utilize the full potential and capabilities of the UAV platform, it is necessary to plan flyable trajectories in the 3-dimensional solution space. Up to this point all methods which have been discussed have been limited to a 2-dimensional solution space. While requiring the aircraft maintain constant altitude is a valid and viable solution to the path planning problem for UAVs, it is not always practical, as there are a variety of situations in which a constant altitude solution is inadequate. Additionally, many stealth techniques, such as terrain masking, require the aircraft to follow the altitude of the topography. In this chapter, 3-dimensional implementations of the Dubins and clothoid planners will be discussed.

A. Problem Definition

The 3D path planners generate a trajectory between 2 poses in 3-dimensional space, the start pose and the finish pose. Each of these poses can be defined using the following definition.

$$P = [x\ y\ z\ \psi\ \theta\ \kappa] \tag{167}$$

where x, y, and z are the components of the pose position vector with respect to the Earth Coordinate System, ψ defines the pose heading, θ defines the pose pitch, κ defines the maximum curvature.

Similar to the 2-dimensional Dubins and clothoid path planning algorithms, there exist 4 combinations of possible solutions for a set of poses. In the event that:

$$\left|\left[\vec{r}_{OP_f}\right]_E - \left[\vec{r}_{OP_s}\right]_E\right| \geq \frac{1}{\kappa_s} + \frac{1}{\kappa_f} \tag{168}$$

is not satisfied, only a subset of these solutions may be available. These solutions depend upon the turn directions chosen for each of the arcs. Thus, the solutions are right-right, right-left, left-left, and left-right, noting that the first direction is that specified for the start arc and the second direction is that specified for the finish arc. This criterion only specifically applies to the Dubins path planner, since the curvature profile of the Dubins path does not depend on arc sweep angle. A similar phenomenon occurs for the clothoid path planners, but the above relationship does not define its occurrence.

Three important coordinate systems are defined in order to determine the possible path solutions between two non-coplanar poses. First is the Earth Coordinate System, which is defined with the x-axis pointing North, the y-axis pointing East, and, by the right-hand convention, the z-axis pointing downward. This is visualized in Figure 6-1 below. Two maneuver planes are defined based upon the Earth Coordinate

System, the Start Coordinate System and the Finish Coordinate System. Examples of these are shown in Figure 6-2.

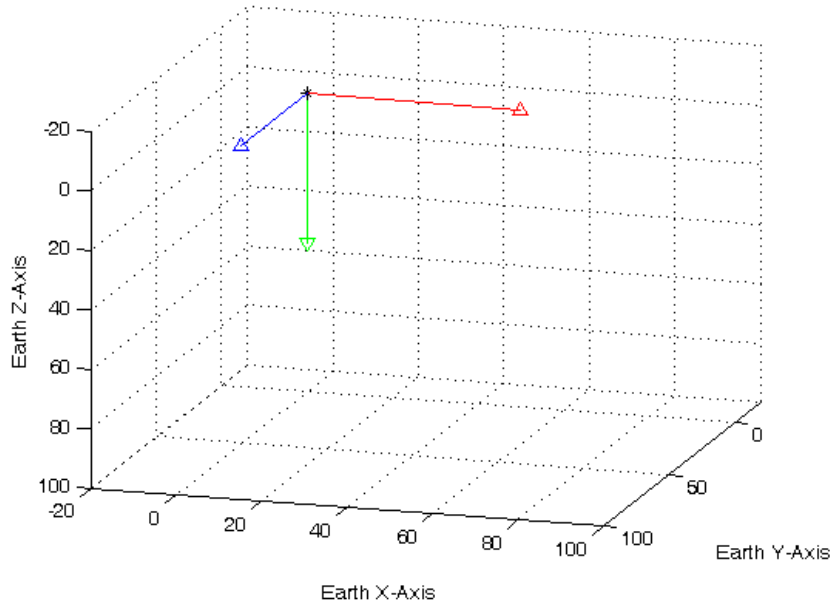

Figure 6-1—3-Dimensional Earth Coordinate System

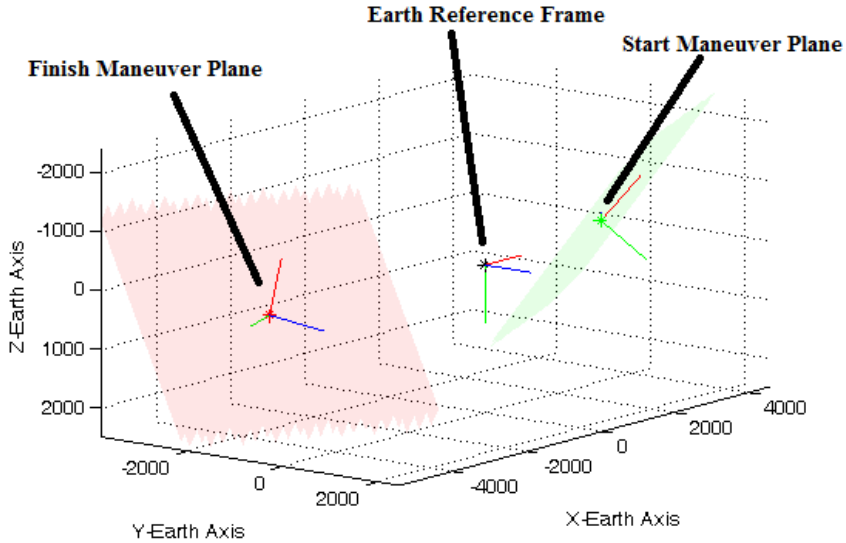

Figure 6-2—3-Dimensional Dubins Coordinate Systems

In order to convert the coordinates of a vector in space between the three coordinate systems, two important transition matrices are needed. Equation 169 contains the transition matrix to convert the coordinates of a vector in Start Coordinate System, or start coordinates, to the Earth Coordinate System, or Earth coordinates.

$$
\begin{aligned}
R_{ES} &= R(\psi_s)\, R(\theta_s)\, R(\psi_s) \\
&= \begin{bmatrix} \cos\psi_s & -\sin\psi_s & 0 \\ \sin\psi_s & \cos\psi_s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_s & 0 & \sin\theta_s \\ 0 & 1 & 0 \\ -\sin\theta_s & 0 & \cos\theta_s \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_s & -\sin\phi_s \\ 0 & \sin\phi_s & \cos\phi_s \end{bmatrix} \\
&= \begin{bmatrix} \cos\psi_s\cos\theta_s & \cos\psi_s\sin\theta_s\sin\phi_s - \sin\psi_s\cos\phi_s & \cos\psi_s\sin\theta_s\cos\phi_s + \sin\psi_s\sin\phi_s \\ \sin\psi_s\cos\theta_s & \sin\psi_s\sin\theta_s\sin\phi_s + \cos\psi_s\cos\phi_s & \sin\psi_s\sin\theta_s\cos\phi_s - \cos\psi_s\sin\phi_s \\ -\sin\theta_s & \cos\theta_s\sin\phi_s & \cos\theta_s\cos\phi_s \end{bmatrix}
\end{aligned}
\tag{169}
$$

In this equation, ψ_s, θ_s, and ϕ_s represent the rotation angle of the Start Maneuver Plane with respect to the Earth Coordinate System.

Similarly, the transition matrix which converts vector components from Finish Coordinate System, or finish coordinates, to Earth coordinates is given in Equation 170.

$$
\begin{aligned}
R_{EF} &= R(\psi_f)\, R(\theta_f)\, R(\psi_f) \\
&= \begin{bmatrix} \cos\psi_f & -\sin\psi_f & 0 \\ \sin\psi_f & \cos\psi_f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_f & 0 & \sin\theta_f \\ 0 & 1 & 0 \\ -\sin\theta_f & 0 & \cos\theta_f \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_f & -\sin\phi_f \\ 0 & \sin\phi_f & \cos\phi_f \end{bmatrix} \\
&= \begin{bmatrix} \cos\psi_f\cos\theta_f & \cos\psi_f\sin\theta_f\sin\phi_f - \sin\psi_f\cos\phi_f & \cos\psi_f\sin\theta_f\cos\phi_f + \sin\psi_f\sin\phi_f \\ \sin\psi_f\cos\theta_f & \sin\psi_f\sin\theta_f\sin\phi_f + \cos\psi_f\cos\phi_f & \sin\psi_f\sin\theta_f\cos\phi_f - \cos\psi_f\sin\phi_f \\ -\sin\theta_f & \cos\theta_f\sin\phi_f & \cos\theta_f\cos\phi_f \end{bmatrix}
\end{aligned}
\tag{170}
$$

Equivalently, in this equation, ψ_f, θ_f, and ϕ_f represent the rotation angle of the Finish Maneuver Plane with respect to the Earth Coordinate System.

B. 3-D Dubins Waypoint

The Dubins 3-D path planning methodology discussed in this section is based on and extrapolated from the method presented by Tsourdos et al. (10). In the 2-dimensional Dubins path planner, a solution is produced between 2 coplanar poses which is composed of 2 circular arcs connected via a common tangent line. Similarly, the 2-dimensional Dubins path planner is expanded to the 3-dimensional methodology by no longer requiring the start and finish poses to be coplanar. A 3-dimensional solution is produced by connecting two circular arcs via a common tangent line, but with the arcs lying in the start and finish maneuver planes, respectively, connected by the common tangent along the intersection of these start and finish maneuver planes. This approach requires that the start and finish poses may not be coplanar.

Consideration of this situation to provide a complete and robust planner is discussed at the end of this section, following derivation of the 3D geometric solution. This complete methodology has also been documented in (82).

a. Derivation of the Vector Formulation

In order to find a suitable flyable path between two pose vectors in 3-dimensional space using the Dubins architecture, it is necessary that the following vector equation be satisfied. These vectors may be defined as shown in Figure 6-3 for an example situation.

$$\vec{p} - \vec{r_s} + \vec{r_f} + \vec{a_s} - \vec{a_c} - \vec{a_f} = 0 \qquad 171$$

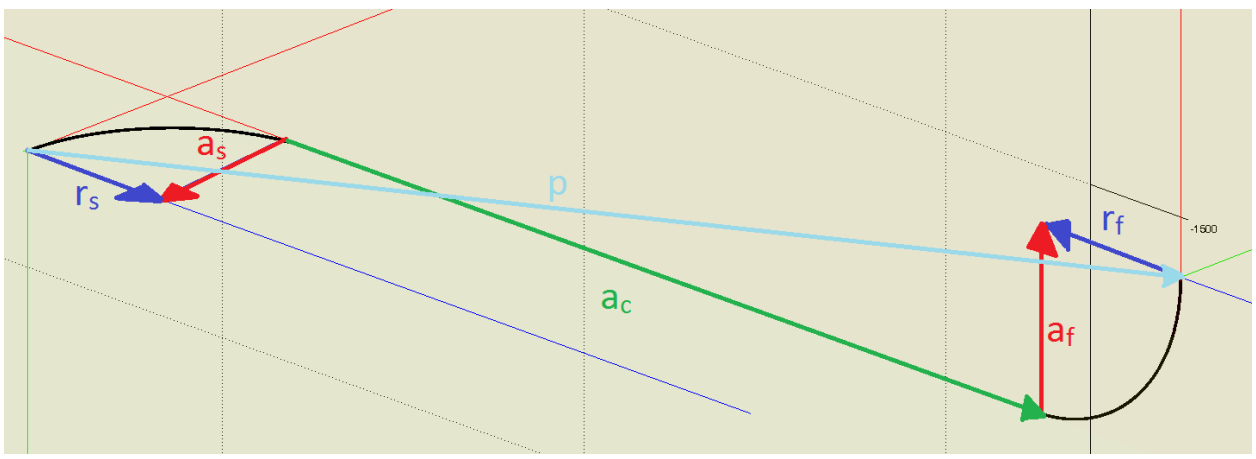

Figure 6-3—Vector Solution of the 3D Dubins System

This can most easily be formulated by defining all of the vector components in terms of the start coordinate axes, shown in Equation 172.

$$[\vec{p}]_S - [\vec{r_s}]_S + [\vec{r_f}]_S + [\vec{a_s}]_S - [\vec{a_c}]_S - [\vec{a_f}]_S = 0 \qquad 172$$

The relative position vector $\vec{p}$ is most simply represented in the Earth Coordinate System according to Equation 173 below.

$$[\vec{p}]_E = [\vec{p_f}]_E - [\vec{p_s}]_E = \begin{bmatrix} x_f - x_s \\ y_f - y_s \\ z_f - z_s \end{bmatrix}_E \qquad 173$$

Using the transition matrix given in Equation 169, the components of the vector $\vec{p}$ can be written in terms of the Start Coordinate System according to the following relationship.

$$[\vec{p}]_S = R_{SE}[\vec{p}]_E = R_{ES}^T \begin{bmatrix} x_f - x_s \\ y_f - y_s \\ z_f - z_s \end{bmatrix}_E \qquad 174$$

Vector $\vec{r_s}$ is described in terms of the Start Coordinate System; thus:

$$[\vec{r_s}]_S = \begin{bmatrix} 0 \\ \pm\frac{1}{\kappa_s} \\ 0 \end{bmatrix}_S \qquad 175$$

where the sign of the y-coordinate is defined based upon the desired turn direction of the start arc, where + corresponds to a right turn and – corresponds to a left turn for this sign convention.

Similarly, the vector $\vec{r_f}$ is defined in the Finish Coordinate System by Equation 176.

$$[\vec{r_f}]_F = \begin{bmatrix} 0 \\ \pm\frac{1}{\kappa_f} \\ 0 \end{bmatrix}_F \qquad 176$$

where the sign of the y-coordinate is defined based upon the desired finish turn direction. In order to convert this vector to start coordinates, the use of both of the transition matrices given in Equations 167 and 168 is needed. This conversion is given as follows:

$$[\vec{r_f}]_S = R_{SF}[\vec{r_f}]_F = R_{SE}R_{EF}[\vec{r_f}]_F = R_{ES}^T R_{EF} \begin{bmatrix} 0 \\ \pm\frac{1}{\kappa_f} \\ 0 \end{bmatrix}_F \qquad 177$$

The connecting vector is located on the intersection of the two maneuver planes. Knowing that: the binormal vectors for the maneuver planes are defined as:

$$[\hat{b_s}]_S = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_S \qquad 178$$

and

$$[\hat{b_f}]_F = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_F \qquad 179$$

the line of action of the connection vector, $\hat{\alpha}$, can be determined from:

$$\hat{\alpha} = \frac{\left(\overrightarrow{b_s} \times \overrightarrow{b_f}\right)}{\left|\overrightarrow{b_s} \times \overrightarrow{b_f}\right|} \tag{180}$$

$$[\hat{\alpha}]_S = \frac{\left[\overrightarrow{b_s} \times \overrightarrow{b_f}\right]_S}{\left|\overrightarrow{b_s} \times \overrightarrow{b_f}\right|} = \frac{\left[\widetilde{b_s}\right]_S \cdot \left[\overrightarrow{b_f}\right]_S}{\left|\left[\widetilde{b_s}\right]_S \cdot \left[\overrightarrow{b_f}\right]_S\right|} = \frac{\begin{bmatrix} 0 & -b_{s_z} & b_{s_y} \\ b_{s_z} & 0 & -b_{s_x} \\ -b_{s_y} & b_{s_x} & 0 \end{bmatrix}_S \cdot \begin{bmatrix} b_{f_x} \\ b_{f_y} \\ b_{f_z} \end{bmatrix}_S}{\left|\begin{bmatrix} 0 & -b_{s_z} & b_{s_y} \\ b_{s_z} & 0 & -b_{s_x} \\ -b_{s_y} & b_{s_x} & 0 \end{bmatrix}_S \cdot \begin{bmatrix} b_{f_x} \\ b_{f_y} \\ b_{f_z} \end{bmatrix}_S\right|} \tag{181}$$

and

$$[\hat{\alpha}]_F = \frac{\left[\overrightarrow{b_s} \times \overrightarrow{b_f}\right]_F}{\left[\left|\overrightarrow{b_s} \times \overrightarrow{b_f}\right|\right]_F} = \frac{\left[\widetilde{b_s}\right]_F \cdot \left[\overrightarrow{b_f}\right]_F}{\left|\left[\widetilde{b_s}\right]_F \cdot \left[\overrightarrow{b_f}\right]_F\right|} = \frac{\begin{bmatrix} 0 & -b_{s_z} & b_{s_y} \\ b_{s_z} & 0 & -b_{s_x} \\ -b_{s_y} & b_{s_x} & 0 \end{bmatrix}_F \cdot \begin{bmatrix} b_{f_x} \\ b_{f_y} \\ b_{f_z} \end{bmatrix}_F}{\left|\begin{bmatrix} 0 & -b_{s_z} & b_{s_y} \\ b_{s_z} & 0 & -b_{s_x} \\ -b_{s_y} & b_{s_x} & 0 \end{bmatrix}_F \cdot \begin{bmatrix} b_{f_x} \\ b_{f_y} \\ b_{f_z} \end{bmatrix}_F\right|} = R_{FS}[\hat{\alpha}]_S \tag{182}$$

The magnitude of the connecting vector, a, is unknown at this point. This magnitude is, by definition, the distance from the end of the start arc to the beginning of the finish arc. In order to define this distance, however, the sweep angles of the arcs must be known. Since the centers of the start and finish primary circles are known, vectors can be defined to extend from the center of the circle to the pose location $\vec{r}_{CP}$, and from the center of the circle to the nearest point on the line of intersection $\vec{r}_{C\alpha}$. The sweep angles can then be calculated using the following relationships:

$$\mu_S = \cos^{-1}\left(\frac{\left[\vec{r}_{C_S P_S}\right]_S^T \cdot \left[\vec{r}_{C_S \alpha_S}\right]_S}{\left|\vec{r}_{C_S P_S}\right|\left|\vec{r}_{C_S \alpha_S}\right|}\right) \tag{183}$$

$$\mu_F = \cos^{-1}\left(\frac{\left[\vec{r}_{C_F \alpha_F}\right]_F^T \cdot \left[\vec{r}_{C_F P_F}\right]_F}{\left|\vec{r}_{C_F P_F}\right|\left|\vec{r}_{C_F \alpha_F}\right|}\right) \tag{184}$$

It is important to note that these equations only yield values $\mu \in [0, \pi]$. Thus, these relationships only yield the "small sweep" of the circle, with the "large sweep" given by $2\pi - \mu$. Since the arc angle can often take the value of the "large sweep", an additional step is needed to determine which sweep to choose. Taking the cross product of the vector components represented in terms of the native maneuver planes respectively, the

sign of the resulting z-component, compared with the desired turn direction, can be used to make this distinction. Thus, these cross products are:

$$\left[\overrightarrow{d_s}\right]_S = \left[\vec{r}_{C_sP_S} \times \vec{r}_{C_S\alpha_S}\right]_S = \left[\tilde{r}_{C_SP_S}\right]_S \cdot \left[\vec{r}_{C_S\alpha_S}\right]_S$$
$$= \begin{bmatrix} 0 & -r_{C_SP_{S_z}} & r_{C_SP_{S_y}} \\ r_{C_SP_{S_z}} & 0 & -r_{C_SP_{S_x}} \\ -r_{C_SP_{S_y}} & r_{C_SP_{S_x}} & 0 \end{bmatrix}_S \cdot \begin{bmatrix} r_{C_S\alpha_{S_x}} \\ r_{C_S\alpha_{S_y}} \\ r_{C_S\alpha_{S_z}} \end{bmatrix}_S = \begin{bmatrix} d_{s_x} \\ d_{s_y} \\ d_{s_z} \end{bmatrix}_S \tag{185}$$

$$\left[\overrightarrow{d_f}\right]_F = \left[\vec{r}_{C_F\alpha_F} \times \vec{r}_{C_FP_F}\right]_F = \left[\tilde{r}_{C_F\alpha_F}\right]_F \cdot \left[\vec{r}_{C_FP_F}\right]_F$$
$$= \begin{bmatrix} 0 & -r_{C_F\alpha_{F_Z}} & r_{C_F\alpha_{F_y}} \\ r_{C_F\alpha_{F_Z}} & 0 & -r_{C_F\alpha_{F_x}} \\ -r_{C_F\alpha_{F_y}} & r_{C_F\alpha_{F_x}} & 0 \end{bmatrix}_F \cdot \begin{bmatrix} r_{C_FP_{F_x}} \\ r_{C_FP_{F_y}} \\ r_{C_FP_{F_z}} \end{bmatrix}_F = \begin{bmatrix} d_{f_x} \\ d_{f_y} \\ d_{f_z} \end{bmatrix}_F \tag{186}$$

Finally, the criterion for the sweep choice becomes:

$$\text{if sign}\left(d_{s_z}\right) == \text{start turn direction,}$$

$$\text{then } \mu_S = 2\pi - \cos^{-1}\left(\frac{\left[\vec{r}_{C_SP_S}\right]_S^T \cdot \left[\vec{r}_{C_S\alpha_S}\right]_S}{\left|\vec{r}_{C_SP_S}\right|\left|\vec{r}_{C_S\alpha_S}\right|}\right) \tag{187}$$

$$\text{if sign}\left(d_{f_z}\right) == \text{finish turn direction,}$$

$$\text{then } \mu_F = 2\pi - \cos^{-1}\left(\frac{\left[\vec{r}_{C_F\alpha_F}\right]_F^T \cdot \left[\vec{r}_{C_FP_F}\right]_F}{\left|\vec{r}_{C_FP_F}\right|\left|\vec{r}_{C_F\alpha_F}\right|}\right) \tag{188}$$

Once the sweep angles are determined, the endpoint of the arcs can be generated. From these points, Euclidean distance is used to determine the length of the connecting vector, a.

Finally, $\overrightarrow{a_c}$ can be computed as:

$$\overrightarrow{a_c} = a \cdot \hat{\alpha} \tag{189}$$

$$\left[\overrightarrow{a_c}\right]_S = a \cdot \left[\hat{\alpha}\right]_S \tag{190}$$

Now, the transition matrices can be used to convert the binormal vectors to the necessary coordinates to arrive at component representations for $\overrightarrow{a_s}$ and $\overrightarrow{a_f}$. If it is defined that:

$$[\hat{\alpha}]_S = \begin{bmatrix} \alpha_{s_t} \\ \alpha_{s_n} \\ 0 \end{bmatrix}_S \tag{191}$$

Equivalently,

$$[\hat{\alpha}]_F = \begin{bmatrix} \alpha_{f_t} \\ \alpha_{f_n} \\ 0 \end{bmatrix}_F \tag{192}$$

Now the vectors $\overrightarrow{a_s}$ and $\overrightarrow{a_f}$ remain to be defined. Since these are each perpendicular to $\overrightarrow{a_c}$, unit perpendicular vectors are used to define the vectors $\overrightarrow{a_s}$ and $\overrightarrow{a_f}$. These unit perpendiculars are defined below in Equations 193 and 194.

$$[\overrightarrow{\beta}]_S = \frac{1}{\sqrt{\alpha_{s_t}^2 + \alpha_{s_n}^2}} \begin{bmatrix} -\alpha_{s_n} \\ \alpha_{s_t} \\ 0 \end{bmatrix}_S \tag{193}$$

$$[\overrightarrow{\beta}]_F = \frac{1}{\sqrt{\alpha_{f_t}^2 + \alpha_{f_n}^2}} \begin{bmatrix} -\alpha_{f_n} \\ \alpha_{f_t} \\ 0 \end{bmatrix}_F \tag{194}$$

Using this notation, vector $\overrightarrow{a_s}$ can be written directly in the Start Coordinate System as:

$$[\overrightarrow{a_s}]_S = \pm\frac{1}{\kappa_s}[\overrightarrow{\beta}]_S = \pm\frac{1}{\kappa_s}\frac{1}{\sqrt{\alpha_{s_t}^2 + \alpha_{s_n}^2}} \begin{bmatrix} -\alpha_{s_n} \\ \alpha_{s_t} \\ 0 \end{bmatrix}_S \tag{195}$$

Vector $\overrightarrow{a_f}$ can be written as:

$$[\overrightarrow{a_f}]_F = \pm\frac{1}{\kappa_f}[\overrightarrow{\beta}]_F = \pm\frac{1}{\kappa_f}\frac{1}{\sqrt{\alpha_{f_t}^2 + \alpha_{f_n}^2}} \begin{bmatrix} -\alpha_{f_n} \\ \alpha_{f_t} \\ 0 \end{bmatrix}_F \tag{196}$$

This will need to be converted to the Start Coordinate System using the transition matrices. This results in:

$$\begin{aligned} [\overrightarrow{a_f}]_S &= R_{SF}[\overrightarrow{a_f}]_F = R_{ES}^T R_{EF}[\overrightarrow{a_f}]_F = R_{ES}^T R_{EF}\left(\pm\frac{1}{\kappa_f}\right)[\overrightarrow{\beta}]_F \\ &= R_{ES}^T R_{EF}\left(\pm\frac{1}{\kappa_f}\right)\frac{1}{\sqrt{\alpha_{f_t}^2 + \alpha_{f_n}^2}} \begin{bmatrix} -\alpha_{f_n} \\ \alpha_{f_t} \\ 0 \end{bmatrix}_F \end{aligned} \tag{197}$$

At this point, all of the relevant vectors have been defined in the Start Coordinate System. Using these formulations Equation 172 can be solved. The unknown values in this equation are the rotations of the start and finish maneuver planes about the pose vectors, ϕ_s and ϕ_f. Since the complexity of these equations precludes direct solution, these values may be determined using a valid numerical method.

b. Solution of the Vector Equation Using Modified 2-Dimensional Bisection

In order to solve the system of equations, a valid numerical method is needed. Several methods were investigated, including various formulations of the Secant Method and Jacobi/Gauss-Seidel Iteration. However, adequate formulation of such complex and non-linear equations leads to significant numerical issues which cause lack of convergence in many situations, or inability to properly represent the functions for use with the method. Thus, a 3-dimensional approach to the bisection method, influenced by Eberly (83), was implemented, which is capable of simultaneously solving the system of three highly-nonlinear equations. This method will be described for the particular case of solving the 3D Dubins vector equation.

This method requires the system to be formulated as follows. Conveniently, this is the representation which naturally occurs from the problem statement.

$$F(\phi_s, \phi_f) = \mathbf{0} \tag{198}$$

Similar to the 1-D bisection algorithm, bounds must be provided for each of the unknowns of the system. The bounds for this system are given below.

$$\phi_s \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \tag{199}$$

$$\phi_f \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \tag{200}$$

At each iteration of the algorithm, the bounds for each unknown are subdivided equally, producing a total of 4 "squares" on whose vertices the solution of the function has been evaluated. Each set of bounds for which the value of all three equations change sign among the 4 vertices signifies a solution is present, and this boundary is kept. In this case, ultimately only one solution is present within the given bounds. When the error of the solutions at the vertices of the bound becomes sufficiently small, the bounds are subdivided a final time, and this central set of values becomes the accepted solution to the set of equations. The full pseudo-code for this modified numerical method is given in Pseudo-Code 6 in Appendix A.

c. Full Algorithm Implementation

Once solution of the vector equations is obtained, the full trajectory may be produced. The 3D Dubins methodology for non-coplanar poses is given in Pseudo-Code 7 in Appendix A.

An additional consideration for producing a robust and fully capable 3D Dubins path planner is that the 3D Dubins solution methodology breaks down for the situation of two coplanar poses. Consequently, any fully developed 3D Dubins planner will need to be able to diagnosis this situation, and thus produce a path between the coplanar poses using the commonly applied 2D Dubins methodology discussed in Chapter 5. B. In order to determine whether two poses are coplanar, three vectors are needed:

$$\left[\overrightarrow{p_s}'\right]_E = \begin{bmatrix} \cos(\theta_s)\cos(\psi_s) \\ \sin(\psi_s) \\ -\sin(\theta_s) \end{bmatrix}_E \qquad 201$$

$$\left[\overrightarrow{p_f}'\right]_E = \begin{bmatrix} \cos(\theta_f)\cos(\psi_f) \\ \sin(\psi_f) \\ -\sin(\theta_f) \end{bmatrix}_E \qquad 202$$

$$[\vec{p}]_E = [\overrightarrow{p_f}]_E - [\overrightarrow{p_s}]_E \qquad 203$$

Arranging these vector components into a matrix and taking the determinant, as in Equation 204, reveals if the vectors are coplanar. If the determinant of the matrix is 0, then the poses are indeed coplanar and the 2D solution methodology should be used for this set of poses.

$$\text{if}\det\begin{vmatrix} \cos(\theta_s)\cos(\psi_s) & \cos(\theta_f)\cos(\psi_f) & x_{f_E} - x_{s_E} \\ \sin(\psi_s) & \sin(\psi_f) & y_{f_E} - y_{s_E} \\ -\sin(\theta_s) & -\sin(\theta_f) & z_{f_E} - z_{s_E} \end{vmatrix} = 0, \qquad \text{poses are coplanar} \qquad 204$$

Once it has been confirmed that the poses are coplanar, the orientation of the plane is needed. Since one plane is enough to describe the full solution, only one set of coordinate axes, the Start Coordinate Axes, is used to describe the solution. The rotation of the plane, angle ϕ_s is calculated from the following relationships.

$$[\hat{n}_s]_E = \frac{\left[\overrightarrow{p_s}' \times \vec{p}\right]_E}{\left|\overrightarrow{p_s}' \times \vec{p}\right|} = \begin{bmatrix} n_{s_x} \\ n_{s_y} \\ n_{s_z} \end{bmatrix}_E \qquad 205$$

$$\phi_s = \cos^{-1}\left(\frac{n_{s_z}}{\cos(\theta_s)}\right), \qquad \phi_s \in [-\pi, \pi] \qquad 206$$

The in-plane heading of each pose, μ, is then calculated with respect to the plane tangent, $\hat{t}_s$. These relationships are given in Equations 207 through 216.

$$[\hat{t}_s]_S = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_S \tag{207}$$

$$R_{ES} = \begin{bmatrix} \cos\psi_s\cos\theta_s & \cos\psi_s\sin\theta_s\sin\phi_s - \sin\psi_s\cos\phi_s & \cos\psi_s\sin\theta_s\cos\phi_s + \sin\psi_s\sin\phi_s \\ \sin\ \psi_s\cos\theta_s & \sin\psi_s\sin\theta_s\sin\phi_s + \cos\psi_s\cos\phi_s & \sin\psi_s\sin\theta_s\cos\phi_s - \cos\psi_s\sin\phi_s \\ -\sin\theta_s & \cos\theta_s\sin\phi_s & \cos\theta_s\cos\phi_s \end{bmatrix} \tag{208}$$

$$\left[\overrightarrow{p_s}'\right]_S = R_{ES}^T \left[\overrightarrow{p_s}'\right]_E \tag{209}$$

$$\mu_s = \cos^{-1}\left(\frac{\left[\overrightarrow{p_s}'\right]_S^T \cdot [\hat{t}_s]_S}{\left|\overrightarrow{p_s}'\right|}\right), \qquad \mu_s \in [0, \pi] \tag{210}$$

$$\left[\overrightarrow{d_s}\right]_S = \left[\overrightarrow{p_s}' \times \hat{t}_s\right]_S = \begin{bmatrix} d_{s_x} \\ d_{s_y} \\ d_{s_z} \end{bmatrix}_S \tag{211}$$

if $\mathrm{sign}(d_{s_z}) < 0$

$$\mu_s = 2\pi - \cos^{-1}\left(\frac{\left[\overrightarrow{p_s}'\right]_S^T \cdot [\hat{t}_s]_S}{\left|\overrightarrow{p_s}'\right|}\right), \qquad \mu_s \in [\pi, 2\pi] \tag{212}$$

$$\left[\overrightarrow{p_f}'\right]_S = R_{ES}^T \left[\overrightarrow{p_f}'\right]_E \tag{213}$$

$$\mu_f = \cos^{-1}\left(\frac{\left[\overrightarrow{p_f}'\right]_S^T \cdot [\hat{t}_s]_S}{\left|\overrightarrow{p_f}'\right|}\right), \qquad \mu_f \in [0, \pi] \tag{214}$$

$$\left[\overrightarrow{d_f}\right]_S = \left[\overrightarrow{p_f}' \times \hat{t}_s\right]_S = \begin{bmatrix} d_{f_x} \\ d_{f_y} \\ d_{f_z} \end{bmatrix}_S \tag{215}$$

$$\text{if } \operatorname{sign}(d_{f_z}) < 0$$

$$\mu_f = 2\pi - \cos^{-1}\left(\frac{\left[\overrightarrow{p_f'}\right]_S^T \cdot [\hat{t}_s]_S}{\left|\overrightarrow{p_f'}\right|}\right), \qquad \mu_f \in [\pi, 2\pi] \tag{216}$$

The reduced 2-D poses then become:

$$P_s' = \left[\left[\overrightarrow{p_s'}\right]_S^T, \quad \frac{1}{\kappa_s}, \quad \mu_s\right] \tag{217}$$

$$P_f' = \left[\left[\overrightarrow{p_f'}\right]_S^T, \quad \frac{1}{\kappa_f}, \quad \mu_f\right] \tag{218}$$

These pseudo-2-D poses are then used to generate the 2-D aircraft trajectory. The trajectory, Q, is then converted from Start Plane Coordinates to Earth Axes Coordinates according to the following relationship.

$$Q_E = R_{ES} Q_S \tag{219}$$

The fully-developed 3D Dubins path planner pseudo-code can be seen in Pseudo-Code 8 in Appendix A.

C. 3-D Clothoidal Waypoint

As previously discussed, the Dubins path planner does not yield a path of continuous curvature, neither in 2 or 3 dimensions. Typical UAV platforms lack the responsiveness to accurately track a trajectory which does not comply with the continuous curvature constraint, as supported by the results displayed in Chapter 8. C. To date, very few researchers have sought to incorporate the clothoid curve into a path planner which produces a path of continuous curvature. Additionally, all efforts to do so have been limited to 2-dimensional path planning situations. Extrapolating from the results of the 2-dimensional Dubins and clothoid path planners, the 3-dimensional Dubins planner can serve as the basis for a 3-dimensional clothoidal planner, which makes use of planar maneuvers connected along the intersection of the maneuver planes.

In the 2-dimensional clothoid path planner, a piecewise continuous solution is produced between 2 coplanar poses which is composed of 2 clothoid arcs connected via a common tangent line. This methodology is expanded to 3-dimensions by allowing the start and finish poses to be non-coplanar. A 3-dimensional solution is produced by connecting two clothoid arcs via a common tangent located along the

intersection of these start and finish maneuver planes, similar to the 3D Dubins path planning methodology. The substitution of clothoid arcs in place of circular arcs for the planar maneuvers introduces new complications into the solution. Since the clothoid profile depends upon the sweep angle as well as the curvature, these sweep angles become additional unknowns in the solution. The full solution of this system is discussed in the following subsection.

a. Derivation of the Vector Formulation

The 3-dimensional clothoid path planning architecture is inspired by combining the 2D clothoid path planner with the 3D Dubins path planner. Thus, the solution depends upon the solution of the vector equation in Equation 220. These vectors may be defined as shown in Figure 6-4 for an example situation.

$$\vec{p} - \vec{r_s} + \vec{r_f} + \vec{a_s} - \vec{a_c} - \vec{a_f} = 0 \qquad 220$$

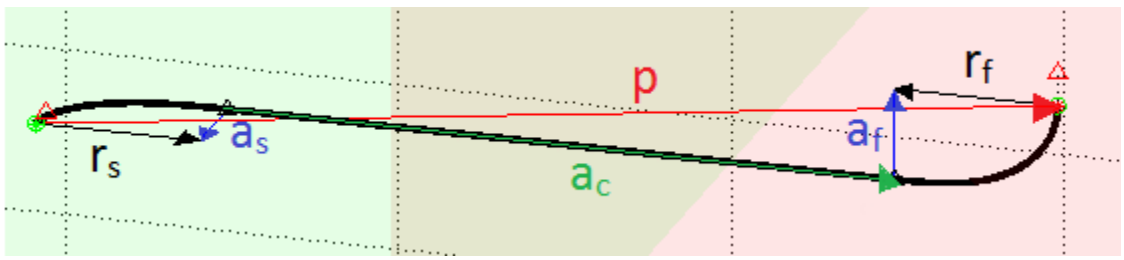

Figure 6-4—Vector Solution of the 3D Clothoid System

As in previous methods, these components will be represented in terms of the start maneuver plane reference frame.

$$[\vec{p}]_S - [\vec{r_s}]_S + [\vec{r_f}]_S + [\vec{a_s}]_S - [\vec{a_c}]_S - [\vec{a_f}]_S = 0 \qquad 221$$

The relative position vector $\vec{p}$ is most simply represented in the Earth Coordinate System according to Equation 222 below.

$$[\vec{p}]_E = [\vec{r}_{OP_f}]_E - [\vec{r}_{OP_s}]_E = \begin{bmatrix} x_f - x_s \\ y_f - y_s \\ z_f - z_s \end{bmatrix}_E \qquad 222$$

Using the transition matrix given in Equation 169, the components of the vector $\vec{p}$ can be written in terms of the Start Coordinate System according to the following relationship.

$$[\vec{p}]_S = R_{SE}[\vec{p}]_E = R_{ES}^T \begin{bmatrix} x_f - x_s \\ y_f - y_s \\ z_f - z_s \end{bmatrix}_E \qquad 223$$

Unlike the Dubins methodology which has a constant turning radius, the clothoid radius of curvature is a continuous function of its length. As discussed above, the profile of the clothoid arc is dependent upon the sweep angle of the arc as well as maximum curvature. This sweep angle depends upon the orientation of the connecting vector $\overrightarrow{a_c}$. Thus, this must be computed next. The magnitude of this vector cannot yet be determined, but the line of action can be defined by:

$$\hat{\alpha} = \frac{\left(\overrightarrow{b_s} \times \overrightarrow{b_f}\right)}{\left|\overrightarrow{b_s} \times \overrightarrow{b_f}\right|} \tag{224}$$

$$[\hat{\alpha}]_S = \frac{\left[\overrightarrow{b_s} \times \overrightarrow{b_f}\right]_S}{\left|\overrightarrow{b_s} \times \overrightarrow{b_f}\right|} = \frac{\left[\widetilde{b_s}\right]_S \cdot \left[\overrightarrow{b_f}\right]_S}{\left|\left[\widetilde{b_s}\right]_S \cdot \left[\overrightarrow{b_f}\right]_S\right|} = \frac{\begin{bmatrix} 0 & -b_{s_z} & b_{s_y} \\ b_{s_z} & 0 & -b_{s_x} \\ -b_{s_y} & b_{s_x} & 0 \end{bmatrix}_S \cdot \begin{bmatrix} b_{f_x} \\ b_{f_y} \\ b_{f_z} \end{bmatrix}_S}{\left|\begin{bmatrix} 0 & -b_{s_z} & b_{s_y} \\ b_{s_z} & 0 & -b_{s_x} \\ -b_{s_y} & b_{s_x} & 0 \end{bmatrix}_S \cdot \begin{bmatrix} b_{f_x} \\ b_{f_y} \\ b_{f_z} \end{bmatrix}_S\right|} \tag{225}$$

where:

$$\left[\widehat{b_s}\right]_S = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_S \tag{226}$$

$$\left[\widehat{b_f}\right]_F = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_F \tag{227}$$

knowing that the connecting vector lies on the intersection line of the maneuver planes and must therefore be perpendicular to both. However, it is important to note that this only defines the line of action of this vector, but not its direction. The direction must be defined separately, according to the convention that the connecting vector should extend from the start maneuver and point toward the finish maneuver.

Using the unit vector $\hat{\alpha}$ to define the direction and orientation of the connecting vector, it is possible to compute the sweep angles of the start and finish maneuvers. These calculations must be performed in 2-dimensions within the start or finish maneuver plane, respectively. The perpendicular vector $Q_\perp$ to a vector in a plane Q is defined as:

$$Q_{\perp} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} Q \tag{228}$$

Using this relationship, perpendicular vectors are produced for the pose vector P and for the connecting unit vector $\hat{\alpha}$. The sweep angle can then be calculated according to the following relationships:

$$\mu_S = \cos^{-1}\left(\frac{\left[\overrightarrow{P_{S_\perp}}\right]_S^T \cdot \left[\overrightarrow{\alpha_\perp}\right]_S}{\left|\overrightarrow{P_{S_\perp}}\right|\left|\overrightarrow{\alpha_\perp}\right|}\right) \tag{229}$$

$$\mu_F = \cos^{-1}\left(\frac{\left[\overrightarrow{P_{F_\perp}}\right]_S^T \cdot \left[\overrightarrow{\alpha_\perp}\right]_S}{\left|\overrightarrow{P_{F_\perp}}\right|\left|\overrightarrow{\alpha_\perp}\right|}\right) \tag{230}$$

It is important to note that these equations only yield values $\mu \in [0, \pi]$. Thus, these relationships only yield the "small sweep" of the arc, with the "large sweep" given by $2\pi - \mu$. Since the arc sweep angle can often take the value of the "large sweep", an additional step is needed to determine which sweep to choose. Taking the cross product of the vector components represented in terms of the native maneuver planes respectively, the sign of the resulting z-component, compared with the desired turn direction, can be used to make this distinction. Thus, these cross products are:

$$\begin{aligned}\left[\overrightarrow{d_s}\right]_S = \left[\overrightarrow{P_{S_\perp}} \times \overrightarrow{\alpha_\perp}\right]_S = \left[\widetilde{P_{S_\perp}}\right]_S \cdot \left[\overrightarrow{\alpha_\perp}\right]_S &= \begin{bmatrix} 0 & -P_{S_{\perp z}} & P_{S_{\perp y}} \\ P_{S_{\perp z}} & 0 & -P_{S_{\perp x}} \\ -P_{S_{\perp y}} & P_{S_{\perp x}} & 0 \end{bmatrix}_S \cdot \begin{bmatrix} \alpha_{\perp x} \\ \alpha_{\perp y} \\ \alpha_{\perp z} \end{bmatrix}_S \\ &= \begin{bmatrix} d_{s_x} \\ d_{s_y} \\ d_{s_z} \end{bmatrix}_S\end{aligned} \tag{231}$$

$$\left[\overrightarrow{d_f}\right]_F = \left[\overrightarrow{\alpha_\perp} \times \overrightarrow{P_{F_\perp}}\right]_F = \left[\widetilde{\alpha_\perp}\right]_F \cdot \left[\overrightarrow{P_{F_\perp}}\right]_F = \begin{bmatrix} 0 & -\alpha_{\perp z} & \alpha_{\perp y} \\ \alpha_{\perp z} & 0 & -\alpha_{\perp x} \\ -\alpha_{\perp y} & \alpha_{\perp x} & 0 \end{bmatrix}_F \cdot \begin{bmatrix} P_{F_{\perp x}} \\ P_{F_{\perp y}} \\ P_{F_{\perp z}} \end{bmatrix}_F = \begin{bmatrix} d_{f_x} \\ d_{f_y} \\ d_{f_z} \end{bmatrix}_F \tag{232}$$

Finally, the criterion for the sweep choice becomes:

$$\text{if } \operatorname{sign}\left(d_{s_z}\right) \mathrel{!}= \text{start turn direction,}$$

$$\text{then } \mu_S = 2\pi - \cos^{-1}\left(\frac{\left[\overrightarrow{P_{S_\perp}}\right]_S^T \cdot \left[\overrightarrow{\alpha_\perp}\right]_S}{\left|\overrightarrow{P_{S_\perp}}\right|\left|\overrightarrow{\alpha_\perp}\right|}\right) \tag{233}$$

$\text{if sign}\left(d_{f_z}\right) == \text{finish turn direction,}$

$$\text{then } \mu_F = 2\pi - \cos^{-1}\left(\frac{\left[\overrightarrow{P_{F\perp}}\right]_S^T \cdot \left[\overrightarrow{\alpha_\perp}\right]_S}{\left|\overrightarrow{P_{F\perp}}\right|\left|\overrightarrow{\alpha_\perp}\right|}\right) \tag{234}$$

It is important to note that the sweep angle cannot be 0 radians, because then the situation would be coplanar and would require special treatment, as will be discussed later. It is also important to note that it is possible to achieve clothoid-based solutions with sweep-angles greater than 2π radians; since the profile of the clothoid arc varies with sweep angle, unlike a circular arc, the clothoid solution for a sweep angle of, for instance, π would be different than the solution corresponding to the sweep angle 3π. For the Dubins planner using circular arcs, these solutions would be identical. Since this means that there are an infinite number of possible clothoid solutions, this methodology only addresses solutions requiring sweep angles of $\mu \in (0, 2\pi]$.

With the sweep angles defined, the clothoid arcs may be generated. This is performed in the start and finish maneuver planes respectively using the methodology discussed in Chapter 5. C.b Defining a Clothoid Curve. Once in Earth coordinates, the zero-curvature point of each curve can be used to determine the magnitude of the connecting vector, $\overrightarrow{a_c}$. The C and S values returned for these points are also used in Equations 149 and 151 to find the magnitudes of the vectors $\overrightarrow{r_s}$, $\overrightarrow{a_s}$, $\overrightarrow{r_f}$, and $\overrightarrow{a_f}$ according to the following relationships, repeated here for convenience and updated for this usage.

$$C_{max} = \frac{2}{\kappa}\sqrt{\gamma_{max}}\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n)!\,(4n+1)}\sqrt{\gamma_{max}}^{4n+1} \tag{235}$$

$$S_{max} = \frac{2}{\kappa}\sqrt{\gamma_{max}}\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)!\,(4n+3)}\sqrt{\gamma_{max}}^{4n+3} \tag{236}$$

These points are converted from the in-plane clothoid axes to the Earth Reference frame, then the distance a between them yields the magnitude of the connecting vector. Then $\overrightarrow{a_c}$ can be computed as:

$$[\overrightarrow{a_c}]_S = [a \cdot \vec{\alpha}]_S = a \cdot [\vec{\alpha}]_S \tag{237}$$

Magnitudes of the vectors $\overrightarrow{r_s}$, $\overrightarrow{a_s}$, $\overrightarrow{r_f}$, and $\overrightarrow{a_f}$ are found from the following relationships:

$$|\overrightarrow{r_s}| = r_s = \frac{C_{m_{axS}}}{\sin(\gamma_S)} \tag{238}$$

$$|\overrightarrow{a_s}| = a_s = S_{m_{axS}} + \frac{C_{m_{axS}}}{\tan(\gamma_S)} \tag{239}$$

$$|\overrightarrow{r_f}| = r_f = \frac{C_{m_{axF}}}{\sin(\gamma_F)} \tag{240}$$

$$|\overrightarrow{a_f}| = a_f = S_{m_{axF}} + \frac{C_{m_{axF}}}{\tan(\gamma_F)} \tag{241}$$

Now the vectors can be calculated. Vectors $\overrightarrow{r_s}$ and $\overrightarrow{r_f}$ are defined as:

$$[\overrightarrow{r_s}]_S = \begin{bmatrix} 0 \\ \pm r_s \\ 0 \end{bmatrix}_S \tag{242}$$

$$[\overrightarrow{r_f}]_F = \begin{bmatrix} 0 \\ \pm r_f \\ 0 \end{bmatrix}_F \tag{243}$$

where the signs are selected based upon the specified desired start and finish turn directions. Vector $[\overrightarrow{r_f}]_F$ must be converted to start coordinates using the following operation.

$$[\overrightarrow{r_f}]_S = R_{SF}[\overrightarrow{r_f}]_F \tag{244}$$

Next, the vectors $\overrightarrow{a_s}$ and $\overrightarrow{a_f}$ need to be computed, which is performed similarly as for the 3D Dubins path planning method. The equations are reiterated here for legibility.

$$[\overrightarrow{\beta}]_S = \frac{1}{\sqrt{\alpha_{s_t}^2 + \alpha_{s_n}^2}} \begin{bmatrix} -\alpha_{s_n} \\ \alpha_{s_t} \\ 0 \end{bmatrix}_S \tag{245}$$

$$[\overrightarrow{\beta}]_F = \frac{1}{\sqrt{\alpha_{f_t}^2 + \alpha_{f_n}^2}} \begin{bmatrix} -\alpha_{f_n} \\ \alpha_{f_t} \\ 0 \end{bmatrix}_F \tag{246}$$

$$[\overrightarrow{a_s}]_S = \pm\frac{1}{\kappa_s}[\overrightarrow{\beta}]_S = \pm\frac{1}{\kappa_s}\frac{1}{\sqrt{\alpha_{s_t}^2 + \alpha_{s_n}^2}} \begin{bmatrix} -\alpha_{s_n} \\ \alpha_{s_t} \\ 0 \end{bmatrix}_S \tag{247}$$

$$[\overrightarrow{a_f}]_F = \pm\frac{1}{\kappa_f}[\overrightarrow{\beta}]_F = \pm\frac{1}{\kappa_f}\frac{1}{\sqrt{\alpha_{f_t}^2 + \alpha_{f_n}^2}} \begin{bmatrix} -\alpha_{f_n} \\ \alpha_{f_t} \\ 0 \end{bmatrix}_F \tag{248}$$

Vector $[\overrightarrow{a_f}]_F$ will need to be converted to the Start Coordinate System via the transition matrices as follows:

$$[\overrightarrow{a_f}]_S = R_{SF}[\overrightarrow{a_f}]_F = R_{ES}^T R_{EF}[\overrightarrow{a_f}]_F = R_{ES}^T R_{EF}\left(\pm\frac{1}{\kappa_f}\right)[\overrightarrow{\beta}]_F$$
$$= R_{ES}^T R_{EF}\left(\pm\frac{1}{\kappa_f}\right)\frac{1}{\sqrt{\alpha_{f_t}^2 + \alpha_{f_n}^2}}\begin{bmatrix}-\alpha_{f_n}\\ \alpha_{f_t}\\ 0\end{bmatrix}_F \quad 249$$

At this point, all of the relevant vectors have been defined in the Start Coordinate System. Using these formulations Equation 220 can be solved. The unknown values in this equation are the rotations of the start and finish maneuver planes about the pose vectors, ϕ_s and ϕ_f. Since the complexity of these equations precludes direct solution, these values must be determined using a valid numerical method.

b. Solution of the Vector Equation Using 2-Dimensional Bisection

Drawing from the experience of solving the 3D Dubins vector equation, a 2-dimensional bisection algorithm was utilized to solve for the maneuver plane rotation angles, ϕ_s and ϕ_f. The full pseudo-code for this numerical method is found in Pseudo-Code 6 in Appendix A. The total solution ranges of the variables are:

$$\phi_s \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \quad 250$$

$$\phi_f \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \quad 251$$

Numerical discontinuities are present in this range, including at the "edges" where either or both ϕ_s and ϕ_f are $\pm\frac{\pi}{2}$. Consequently, to avoid erroneous sign changes in the bisection method, the range was narrowed on all limits by 0.001 degrees, or approximately 1.7×10^{-5} radians, with the assumption that this is not a large enough error that if a solution were to lie one of these edges that a solution could not be found. In order to illustrate the nature of this function and its discontinuities, four surfaces may be created by solving the equation at a fixed grid. These surfaces, representing the X-coordinate error, Y-coordinate error, Z-coordinate error, and total error, are provided below in Figure 6-5 through Figure 6-8 for one particular problem, consisting of a set of poses and the desired turn directions. Note that this is only an example; the surface will be different for any other set of poses and/or turn directions. Also note that producing these surfaces is computationally expensive, but not required for solution of the vector equation using bisection. They are primarily for visual purposes and verification.

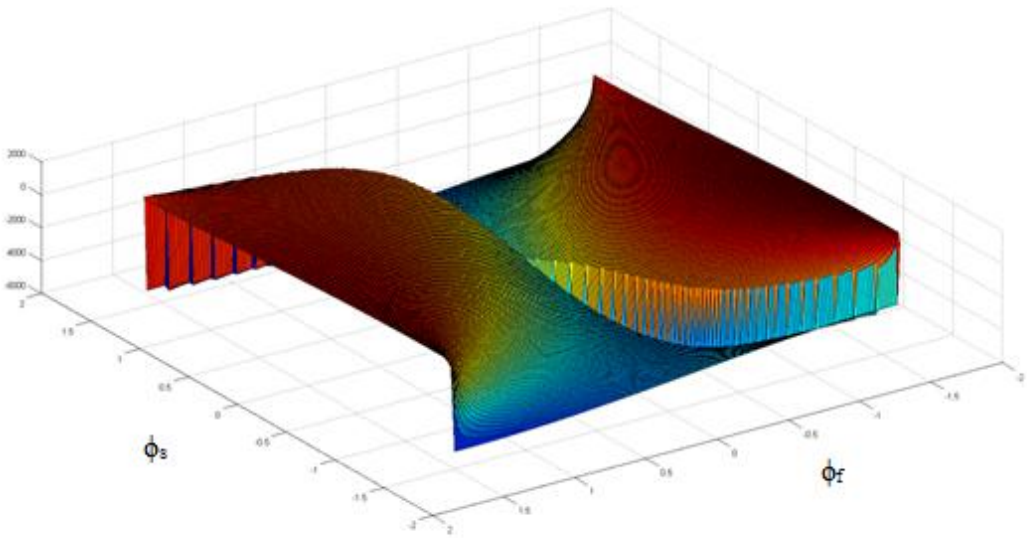

Figure 6-5—X-Coordinate Error Surface for 3-D Clothoid Over Full Bisection Range

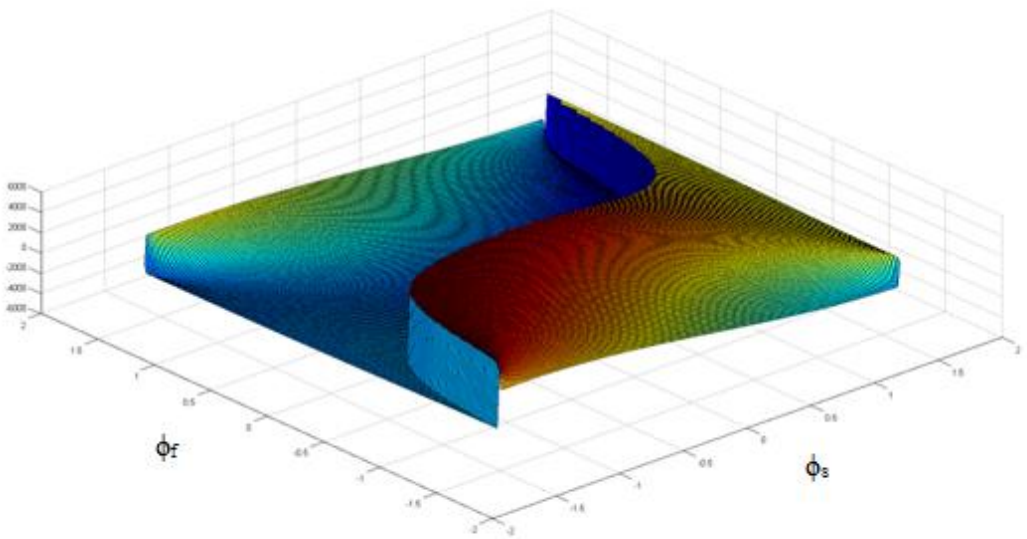

Figure 6-6—Y-Coordinate Error Surface for 3-D Clothoid Over Full Bisection Range

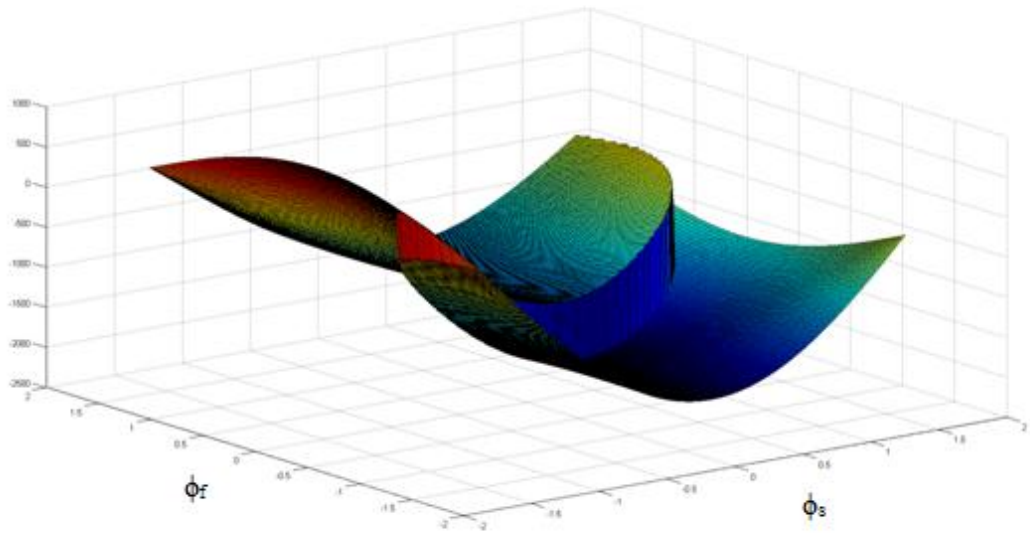

Figure 6-7—Z- Coordinate Error Surface for 3-D Clothoid Over Full Bisection Range

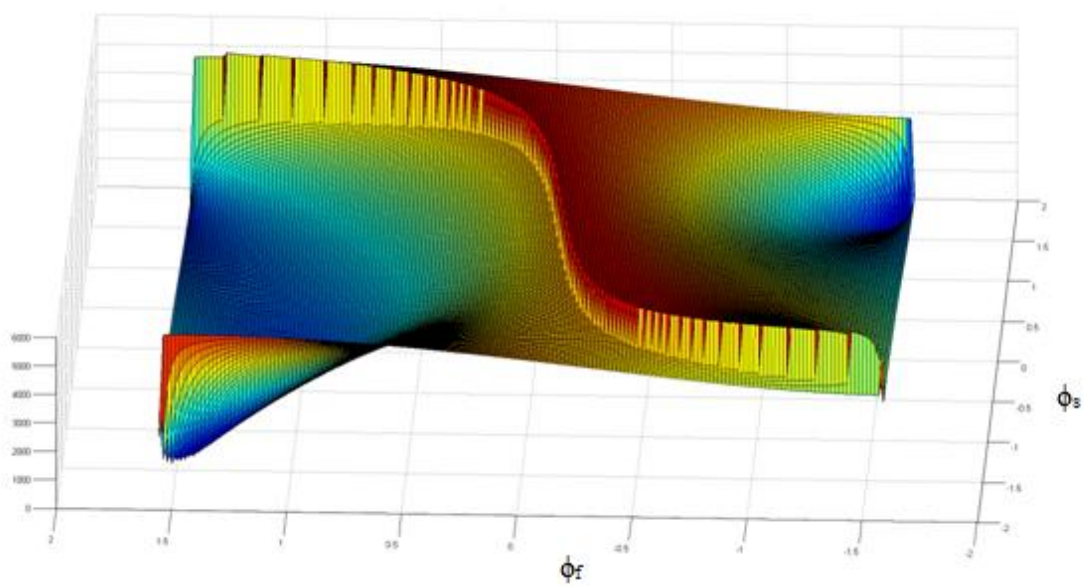

Figure 6-8—Total Error Surface for 3-D Clothoid Over Full Bisection Range

c. Full Algorithm Implementation

Once solution of the vector equations is obtained the full trajectory may be produced. The 3D clothoid methodology for non-coplanar poses is given in Pseudo-Code 7 in Appendix A.

An additional consideration for producing a robust and fully capable 3D clothoid path planner is that the 3D clothoid solution methodology, like the 3D Dubins methodology, breaks down for the situation of

two coplanar poses. Thus, the fully-implemented 3D clothoid path planner is capable of diagnosing this situation and producing a path between the coplanar poses using the 2D clothoid methodology discussed in Chapter 5. C. The coplanar pose combination can be diagnosed via the same mechanism used for the 3D Dubins path planner, detailed in Section B.c above. The fully-developed 3D clothoid path planner pseudo-code can be seen in Pseudo-Code 8 in Appendix A.

D. Numerical Issues

An inherent problem in using numerical solving techniques consists of the lack of convergence in certain cases. As has been previously stated, the functions which define the clothoid path solution are highly nonlinear and not well-behaved. The same can be said for the 3D Dubins equations. Consequently, solution of these systems of equations through typical numerical methods such as Jacobi Iteration, Gauss-Seidel Iteration, Newton-Raphson Method, or Secant Method is not possible. Due to the implicit representation of these systems of equations, it is not possible to represent these equations in a format suitable for solution by these methods. This realization led to the selection of bisection for solution of the highly-complex systems of equations for generating either 3D Dubins or 3D Clothoid trajectories.

The choice of bisection for solution is not without its disadvantages. It is well-known that the bisection method is susceptible to missing solutions in the presence of discontinuities and multiple solutions. Conveniently, either the 3D Dubins or 3D clothoid system of equations has only one solution within the named bounds, for each set of turn combinations. However, due to the ill-behaved nature of these systems, it is possible for a solution to exist which the bisection method is incapable of obtaining, due to sign-shift errors at the discontinuities. In order to help reduce the occurrence of this issue, the solutions are checked at a regular interval of reasonably calculable resolution, and the bounds containing the minimum-error solutions are maintained as the starting bounds of the bisection algorithm. This is similar to the method used to narrow the bounds for the 2D clothoid bisection algorithm. This is an optional step which adds approximately 6 seconds to the calculation time of the path, but does increase its success rate.

These systems can also suffer from bound-runaway. This occurs when the discontinuity crosses the zero value for all 3 equations within a bound. Consequently, rather than erroneously eliminating a bound, the algorithm may find a continuously-growing number of candidate bounds in which a solution could lie. For these situations, the maximum number of total bounds is limited, and if this limit is reached, the algorithm terminates and reports no solution.

Chapter 7. IMMUNITY-BASED EVOLUTIONARY POSE OPTIMIZATION ALGORITHM FOR UAV TRAJECTORY GENERATION

This section will discuss the initiative and general architecture of the Immunity-Based Evolutionary Pose Optimization (IBEPO) algorithm. This will include the data and scenario formulation provided to the algorithm and the various algorithms modules.

A. Biological Immune System

In 1955, the Danish immunologist Niels K. Jerne developed the theory of clonal selection (84), the mechanism by which the biological immune system produces antibodies, or lymphocytes, in response to harmful antigen, which earned him a Nobel Prize in 1984 (85). The clonal selection mechanism gets its name due to the fact that the antigen determines the fitness of the lymphocytes, composed of B-sell and T-cells, for clonal expansion (86). In general, the immune system generates a basis population of antibodies containing an enormously diverse variety of receptors. The antibodies are generated at random and checked for matching with the body's own cells. Assuming that the antibody does not contain receptors that react to the self cells, it is matured.

When an antigen is introduced into the system, the small number of lymphocytes capable of binding to the antigen's particular epitope will do so. This triggers the clonal selection process in which the antibodies fit to bind with the antigen rapidly reproduces identical copies of themselves. During the course of this rapid mitosis, T-helper cells produce identical copies of the activated receptor. However, B-cells undergo a process referred to as somatic mutation, a directed rapid mutation which seeks to make the offspring of the B-cells better at binding the particular antigen (87). Of this population of cells, those with better abilities bind to the antigen more frequently and for longer, thus producing more copies of themselves.

B. Immunity-Based Evolutionary Optimization Paradigm

The biological immune system is effective at rapidly searching the solution space of receptors and arriving at an effective solution to ridding the body of invading antigen. This efficient solving mechanism has become inspiration for a new optimization technique called immunity-based evolutionary optimization (IBEO), also referred to as artificial immune optimization (sic). This artificial intelligence-based optimization technique is noted for its ability to quickly converge to an effective solution to a given problem. Implementations of this algorithm are particularly effective in the situation when a problem may have multiple equally-effective solutions (88)

Implementations of this artificial intelligence technique have been undertaken by several researchers, including (89), (90), and (91). Gaspar et al. (89) applied immunity-based optimization to pattern recognition, which solved the problem presented robustly and efficiently. Coello et al. (90), applied the immunity-based optimization mechanism for constrained and unconstrained multi-objective optimization problems, using Pareto dominance and feasibility, an economic principle taken here to mean that one objective cannot be better fulfilled without worsening another objective, to determine the fitness of solutions. When compared to similar approaches the immunity-based optimization typically performed as well or better, based on the problems assessed. Finally, deCastro and Timmis (91) combined the immunity-based optimization with other artificial immune system approached for data clustering and pattern recognition, which results proved it was well-suited to do.

C. Immunity-Based Evolutionary Pose Optimization Algorithm Layout

IBEO is favored over other optimization techniques due to its marked ability to quickly achieve a very good solution to a multi-objective problem. This makes IBEO a good choice for path planning, as this problem requires balancing many factors including exposure to threats, collision avoidance, goal-seeking, point-of-interest observation, path length, flight time, and fuel consumption. Choice of an ideal set of poses to achieve these goals is non-trivial; thus a mechanism such as this is required.

In essence, the immunity-based optimization is a specialized form of an evolutionary algorithm (EA). However, since it draws its inspiration from the immune-system's adaptive response mechanism, key differences of this method make it a better choice than EA for the problem of path planning. In IBEO, there is no exchange of genetic material between B-cells. This reproduction occurs asexually. Additionally, the somatic mutation process calls for mutation rates significantly higher than would be seen in nature through genetic accident. In fact, mutation rates of the B-cell reproduction are inversely proportional to the affinity of the B-cell for binding with the epitope. The number of copies of a particular B-cell is also directly proportional to its affinity.

a. Representation

The individual candidate solutions represent the B-cells in the biological immune system. A solution to the optimization problem will consist of a set of poses connecting the start pose to the goal pose, which observes all points of interest and does cause intersection with obstacles. These will be encoded as real values, as the quantity of parameters would be too cumbersome to represent using binary representation.

While this algorithm is intended for use with 3-dimensional environments, coplanar solutions are handled as a special case. Thus, this algorithm is applicable to both 2-dimensional and 3-dimensional environments, depending upon the nature of the poses and threats specified.

b. 3-D Trajectory Generation and Dynamic Constraints

Additional dynamic constraints must be considered for the situation of 3-dimensional path planning. In the above discussed methods, paths are formulated in terms of piecewise continuous segments, with two arcs located within the maneuver planes and the connecting segments situated at the intersection of these planes. Although these algorithms incorporate maximum curvature constraints on the arc segments they generate, this may not always be sufficient to ensure a flyable trajectory is produced by the algorithm. Thus, at two points in these algorithms, it is necessary to apply constraints to ensure a flyable trajectory. The simpler of these is that the poses specified may not exceed the dynamic limitations of the aircraft. This amounts to ensuring that the pitching angle of the pose does not exceed that of the aircraft.

However, the more difficult implication comes with determining the trajectory of the aircraft during the turning maneuvers. These maneuvers take place within the respective maneuver planes which are not generally parallel with the Earth axis. Thus, the relative climbing angle of the commanded trajectory points also may not exceed the maximum climbing rate. Limiting the maneuver plane angle is not sufficient, as the plane angle may exceed the allowable climb angle, while the trajectory does not. Thus the commanded climb angle of trajectory itself must be verified not to exceed the maximum climbing angle. This is simply computed by checking the pitch angle of the direction vector pointing from the previous to current trajectory point.

c. Algorithm Logic

The full IBEPO algorithm begins by generating a full population of viable individuals. These are then rated and stored as parents. The parents then produce a set of offspring, proportional to the parent's affinity. These offspring then undergo somatic mutation, with a mutation rate inversely proportional to the affinity of the parent. The affinity of these offspring is then reassessed. The parents and the offspring are recombined to form the complete population. Then, tournament selection is performed to fill a new population. Finally, to reduce the risk of stagnation, increase variability in the population, and improve convergence time, a specified number of new random individuals is added to the population at each generation using the same mechanism used to generate the initial population. This process continues for a specified number of generations or until a minimum performance index of the best individual is reached. The complete logic of this optimization algorithm can be seen in Figure 7-1.

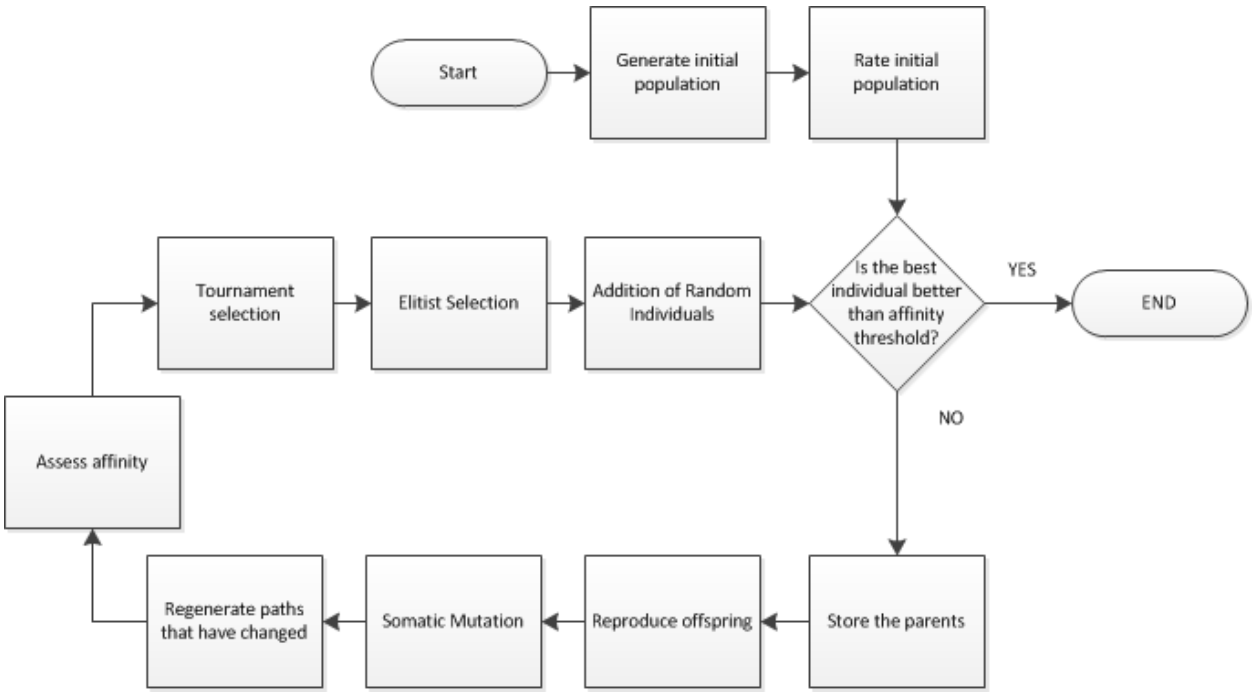

Figure 7-1—IBEPO Algorithm Logic Diagram

Several modules are needed to perform the IBEPO algorithm. These will include generation of the initial population, affinity assessment, reproduction, somatic mutation, and selection. These components of the algorithm will be discussed in the following sections.

c.1 Generating the Initial Population

The initial population is generated such that all candidate solutions are viable, but not necessarily optimal solutions. This means that they must meet the basic requirements of non-collision with obstacles and observance of all points of interest. To generate the population, all permutations of the order of the intermediate poses (those between the start and goal poses) are generated. Since the number of permutations produced is $n!$, where n is the number of intermediate poses, it is immediately apparent that all possible pose order permutations cannot be included in the initial population if a large number of intermediate poses is considered, without considering a very large population size, and therefore drastically increasing the computational overhead of the algorithm. Thus, the first step to generating an individual is to randomly select one of these pose order permutations to populate the pose order of the individual. Thus, the poses are stored within the structure of the individual. For each of these poses, a variety of parameters may be constant or variable, including heading and pitch angles, radius, and turn direction. If these parameters are allowed to vary, they are specified for the individual at random, observing flyability constraints. Once the poses are set, the paths connecting them must be generated. For each set of poses, the clothoid path is computed. If no flyable clothoid path is found, the Dubins path is computed instead. If neither a flyable clothoid nor Dubins

path is available for any set of two poses within an individual, the individual is rejected. Once a flyable path is generated between each set of poses in an individual, the trajectory is checked for obstacle intersection. If none of the paths in an individual conflict with an obstacle, the paths are stored in the structure of the individual and the individual is accepted as part of the population. If any path segment does conflict with an obstacle, an additional pose is added in order to fly around the obstacle. However, if this is not readily feasible, the individual is rejected. This process is repeated until a full population of viable individuals is found. The following flowchart depicts this process.

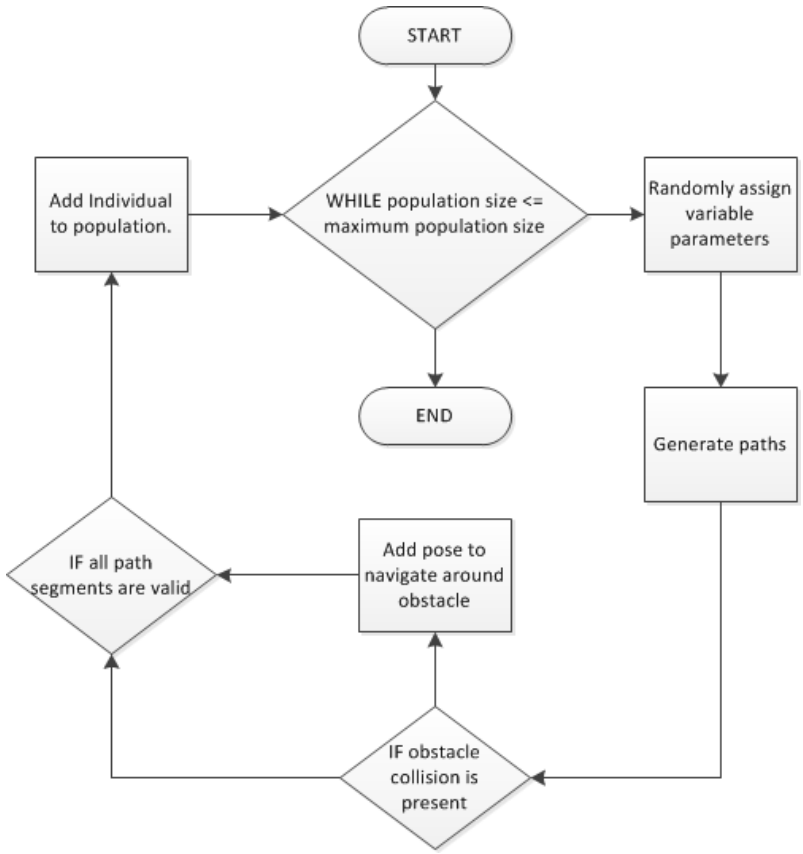

Figure 7-2—Logic for Generating the Initial Population

c.2 Affinity Assessment

The affinity of the solutions will be assessed based on several parameters. These will include fuel path length, fuel consumption, risk exposure, and curvature profile. The affinity of an individual is rated from 0 to 1, where 1 is perfectly good and 0 is perfectly bad. In order to determine these parameters, the path for each set of poses in an individual will need to be generated.

To assess the path length of an individual and to assign an affinity to it based on the total path length, the path length is summed for all path segments in the individual. An affinity value is then assigned to the individual for the path length based upon the relationship in Figure 7-3. In this calculation, the minimum

path length is assigned to be the furthest distance from the start pose to any of the other mandatory poses. The maximum distance is set to be this distance multiplied by the number of mandatory poses.

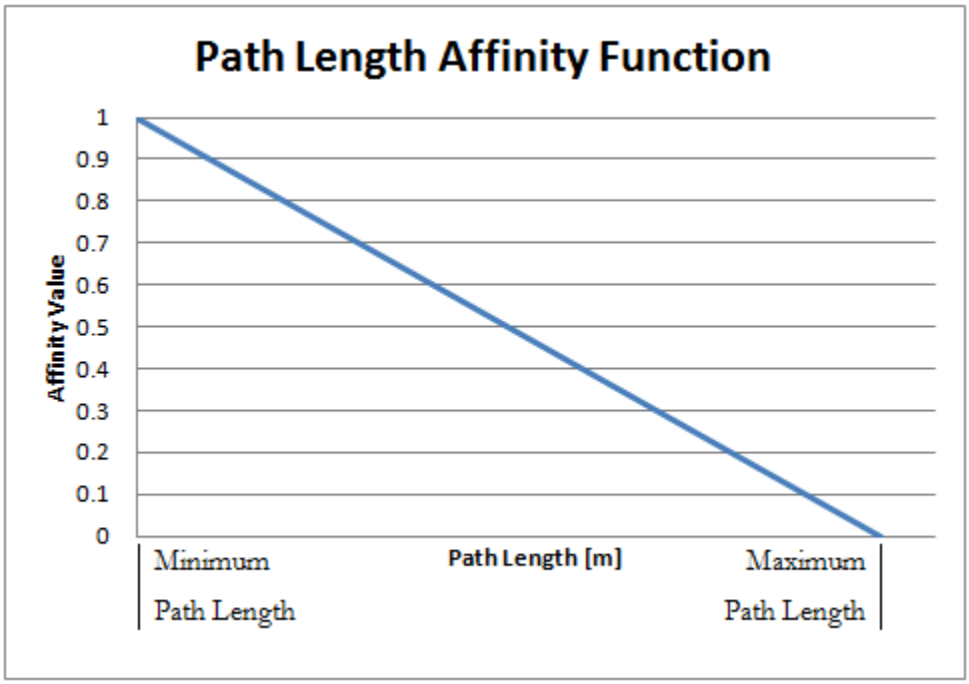

Figure 7-3—Path Length Affinity Function

Fuel consumption is not a simple calculation for a path. It depends upon a variety of factors, including altitude, velocity, and throttle setting, and is a nonlinear function. For this application, the concern is not placed upon the actual quantity of the fuel consumed in flying a trajectory, but rather which paths will be more fuel-conscious. Consequently, the fuel consumption for this application is approximated to be proportional to the commanded climb angle of the path, taken at the straight-line connecting vector. Steeper commanded climb angles will require more throttle to maintain cruise velocity, while shallower angles will require less. Since the concern is only on relative fuel consumption, this is a reasonable assumption, whose assessment can be performed quickly. Thus, for each path segment in an individual, an affinity based on fuel consumption is assigned according to the affinity function shown in Figure 7-4. This affinity value is then averaged over the path segments, to yield an affinity for the individual.

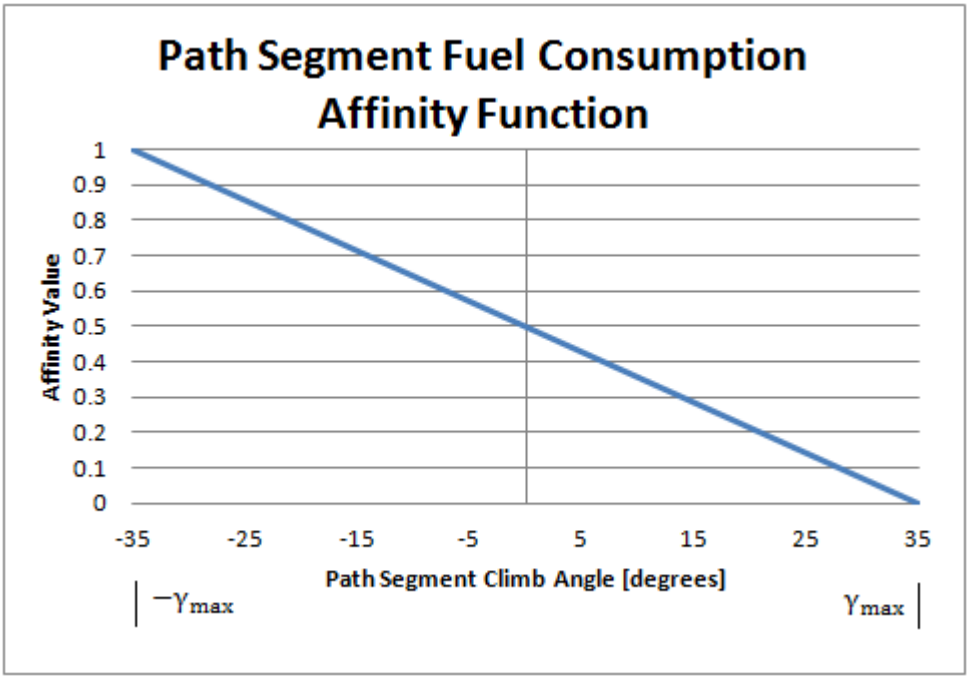

Figure 7-4—Path Segment Fuel Consumption Affinity Function

The affinity for minimum threat exposure is assigned to an individual based upon the threat exposure exerted on the aircraft at each trajectory point. Thus, for each point in the trajectory, the risk intensity is summed for all threats inside whose impact radius the point falls. This risk exposure is summed and averaged over all the points in the trajectory, to yield a total threat exposure value between 0 and 2. This is applied to the affinity function shown in Figure 7-5 to assign the threat exposure affinity to the individual.

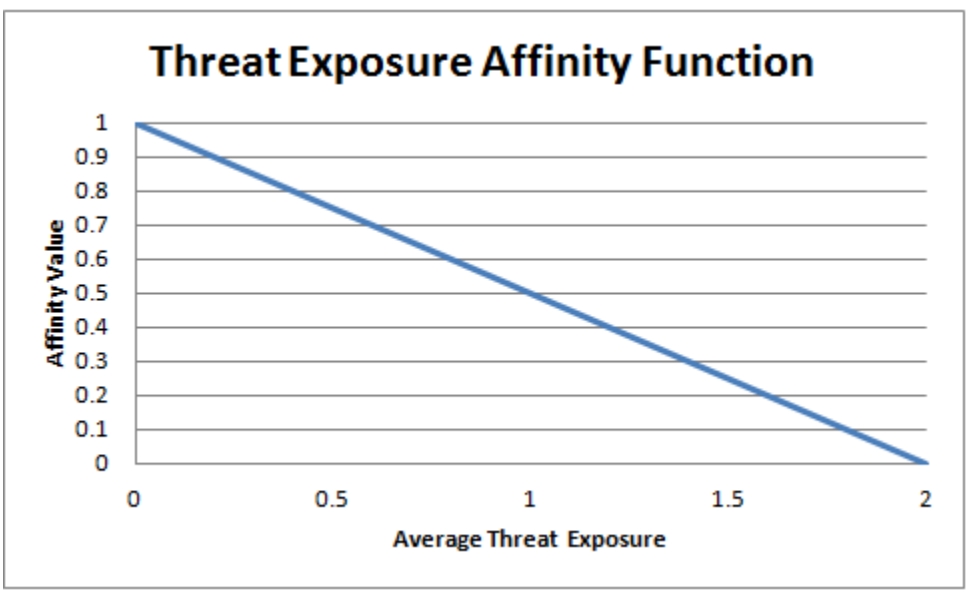

Figure 7-5—Threat Exposure Affinity Function

Finally, the curvature affinity is assigned based upon the number of clothoid-based path segments that are present in the individual. As discussed previously, clothoid arcs are preferred as they yield a more easily followable path. Thus, the curvature affinity is assigned to be the number of clothoid path segments divided by the total number of path segments for an individual. This affinity function is shown in Figure 7-6.

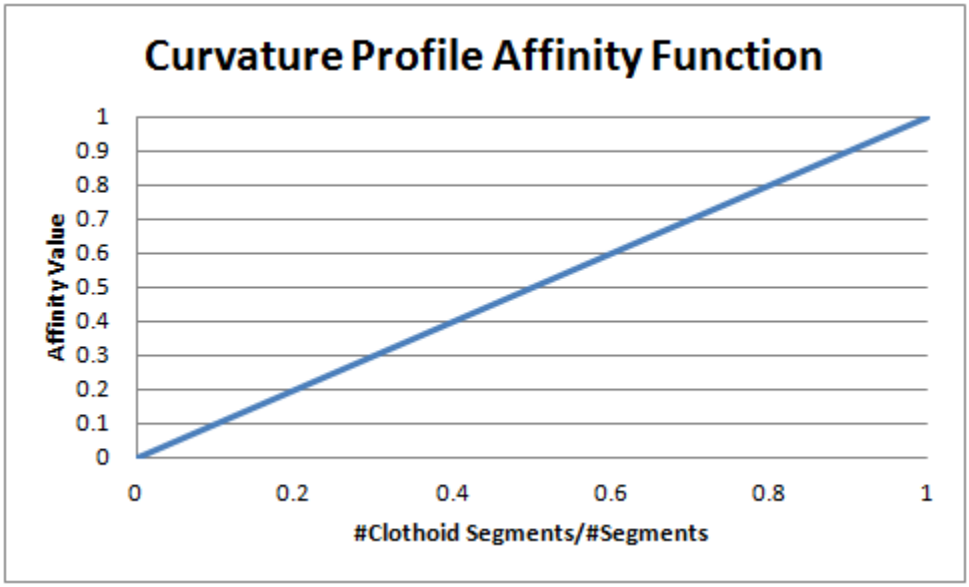

Figure 7-6—Curvature Profile Affinity Function

Once these four affinity values are assigned, the individual's total affinity is assigned using the following weighted average.

$$A_{total} = 0.2A_{Path\ Length} + 0.2A_{Fuel\ Consumption} + 0.2A_{Curvature} + 0.4A_{Threat\ Exposure} \quad 252$$

c.3 Somatic Mutation and Reproduction

Reproduction in the biological immune system is performed based upon the affinity of a particular B-cell in the population to bind with a particular antigen. B-cells with higher affinity produce more copies of themselves, while those with lower affinity may reproduce only once. These copies then undergo the process of somatic mutation, a type of mutation characteristic to the biological immune system which is marked by very aggressive mutation whose intensity is inversely proportional to the affinity of the parent B-cell. Thus, solutions achieving higher affinity will be mutated only a small amount, while those with low affinity are mutated much more drastically in an attempt to reach a better solution. In this way, the immunity-based optimization paradigm balances exploration of new solutions with exploitation of existing solutions. The high mutation rate combined with parent cells remaining in the population is responsible for the quick convergence of this optimization technique. For this algorithm, mutation will consist of random changes to

the position, heading, pitch angle, curvature, and turn direction associated with the poses which make up a solution. Note that mutation will not be performed on any of the parameters of the poses which are specified to be constant. Additionally, a pose may be added between two of the existing poses. A single mutation is considered to be a change in any one of these parameters for one pose within one solution; however, the high mutation rates may result in multiple mutations being performed to an offspring before its affinity is assessed.

Each time an individual is modified, its affinity must be assessed. This requires the full trajectory to be generated. To minimize computational expenditure, the path is stored as a series of path segments related to a set of poses. Consequently, when a pose is altered, only the path segments connecting to it need to be recomputed. When these path segments are recomputed, individuals whose mutations do not result in flyable paths or cause overlapping with obstacles are rejected.

c.4 Selection

Selection of the new population is performed to simulate the lifespan of the B-cells. Requiring that individuals expire after a number of generations could negatively impact the solution convergence, as the best individual could die out prematurely. Additionally, without a selection mechanism to eliminate poorly performing individuals, the population size could grow rapidly, increasing the overall computational overhead of the algorithm without improving performance. Instead, the new population is chosen using a tournament selection algorithm. In tournament selection, a number of individuals is selected with equal probability to compete for a slot in the new population. The individual in the tournament with the highest affinity gets placed in the new population. The minimum tournament size is 2. However, using a larger tournament size decreases the chance that worse individuals will make it into the new population. For this algorithm, a tournament size of 3 is used. This tournament process is repeated until the new population is filled to a specified size, which eliminates uncontrolled population growth. This process is depicted in the flowchart in Figure 7-7 below.

At each generation, the elitist selection mechanism is implemented to ensure that the best individual in a population has at least one offspring in the new population. This is ensured by allocating one space in the new population for the best individual from the current population. Additionally, a small number of spaces in the new population are allocated to new random individuals, which are generated through the same mechanism as the individuals in the initial population. The introduction of a small number of random individuals in each population helps to improve exploration of the solution space and avoid stagnation in the population variability. It has also been shown in the literature (91) to help improve the convergence rate of the optimization with respect to number of generations which must be performed.

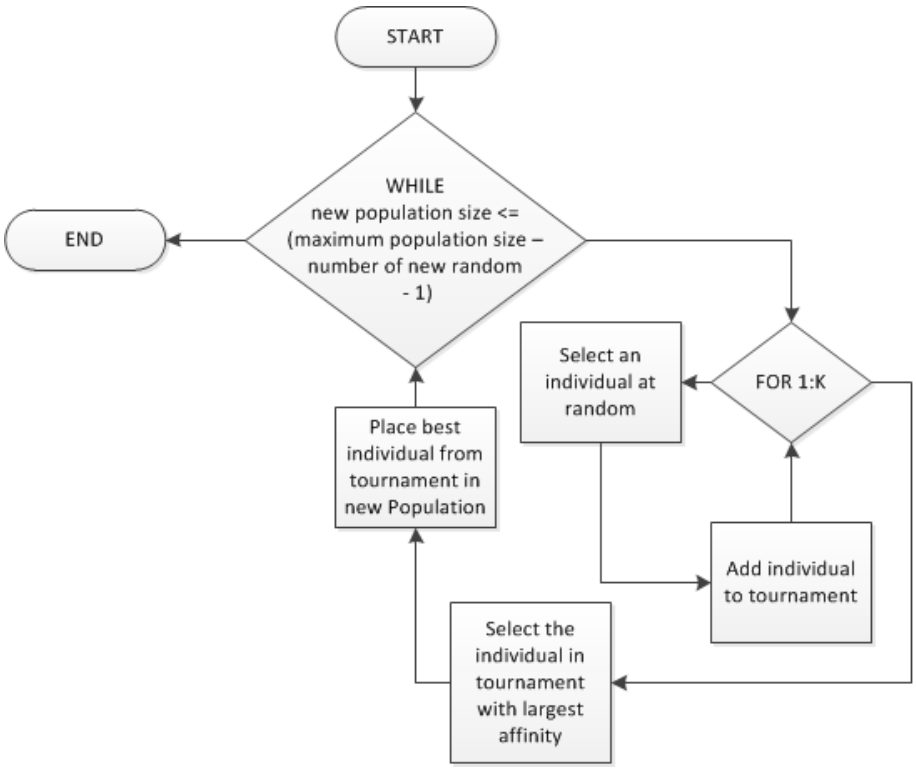

Figure 7-7—Tournament Selection Logic

d. Environment Specification Interactive GUI

In order to permit selection of both threats and poses in 3 dimensions, as needed for the IBEPO algorithm, an interactive environment Guided User Interface (GUI) was created. This GUI allows the user to create, delete, and modify the threats contained in the environment. Additionally, the user can specify the start and goal poses for the aircraft, any waypoints to traverse through, parameters for these poses, and which of their parameters may undergo optimization, and whether the pose visitation order is fixed or variable. These may not only be created, edited, and removed, but may also be reordered, except for the start and goal, which must remain first and last in the array. This GUI can be seen in Figure 7-8 below.

To discuss these parameters, threat zones may be either spherical or cylindrical. For either of these shapes, an X, Y, and Z location must be specified. Spheres and cylinders both have an impact radius, but cylinders also require a height. For both of these shapes, the risk intensity and risk severity must be specified; these are used to compute the risk intensity.

For poses, there are four types to select from. A start pose and goal pose must be selected for any set. These have constant defined heading and pitch. The start is required to be first in the order, and the goal is required to be last. Intermediate points may be either poses, with a fixed heading and pitch, or waypoints, where only the position is constant. For any of these, the turning radius will be limited by the minimum

aircraft turning radius, but a fixed or variable radius and its value may be specified. Likewise, the turn direction may be either specified to be right or left, or may be left variable. This parameter helps define points of interest around which the aircraft should fly, by selecting whether to turn left or right to circle the area.

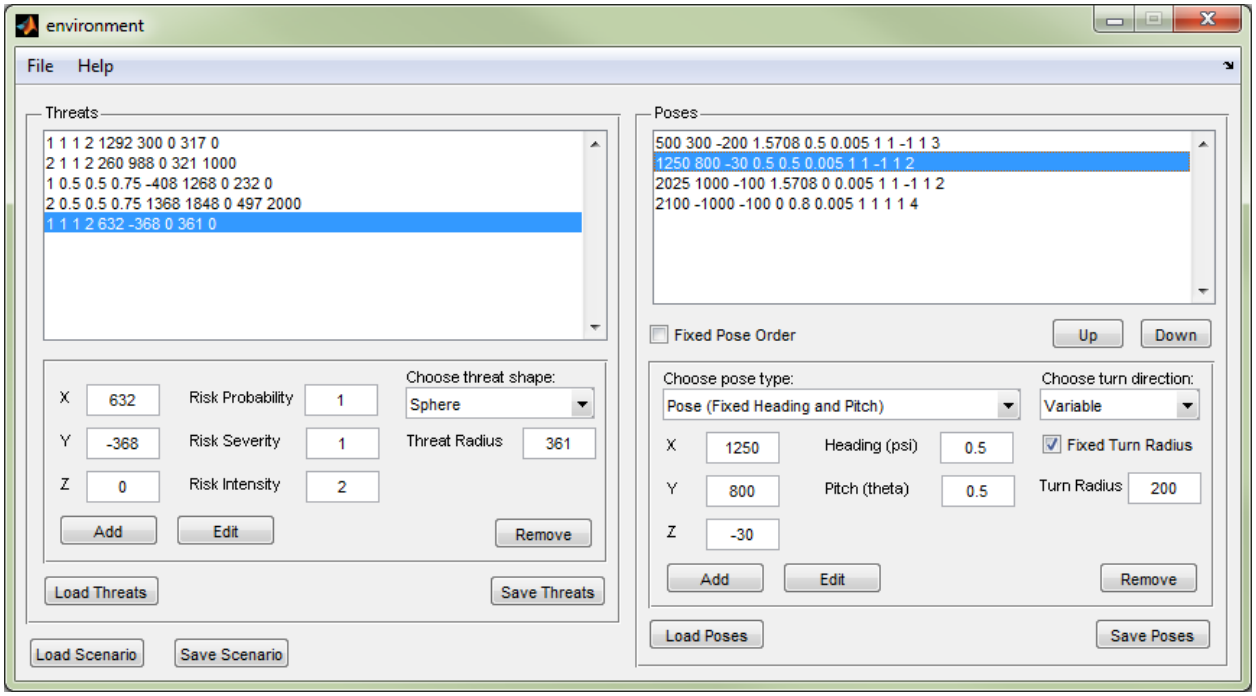

Figure 7-8—IBEPO Environment GUI

The key feature of this GUI is that, as the environment is altered, it is dynamically plotted for the user to visualize and inspect. An example environment plot is shown in Figure 7-9. The ground plane, or Earth Reference Frame, is plotted in green. Threats are plotted in red with their alpha value (brightness) varying with risk intensity, so that worse threats appear darker. Poses are plotted as a position and a direction vector. To indicate their pose order, a dotted line is plotted between them. Again, as any of these parameters are changed in the GUI, the plot is automatically updated to reflect these changes. Figure 7-10 shows the environment plot rotated to highlight these features.

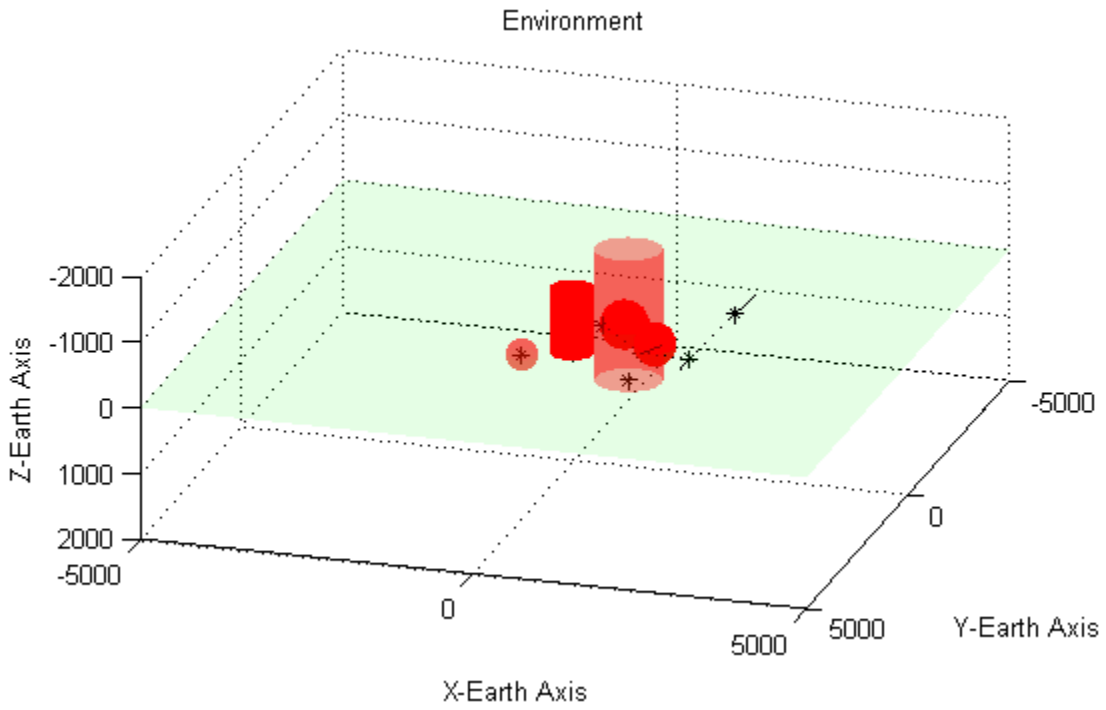

Figure 7-9—3D Environment Plot Native View

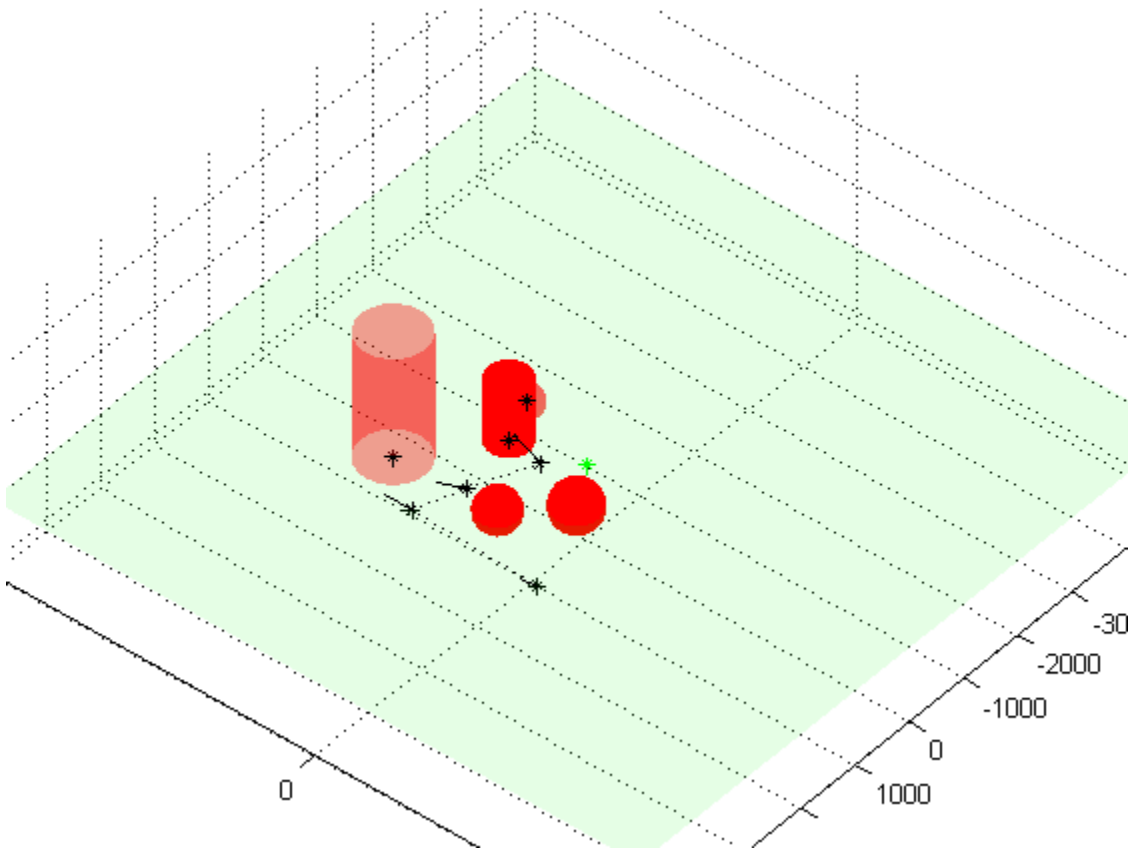

Figure 7-10—3D Environment Plot Rotated

e. Algorithm Performance

An example IBEPO optimization using the environment seen in the above figures was performed for a population size of 20 individuals for 15 generations. The intermediate poses were specified to be waypoints, such that their positions were fixed but their visitation order, heading and pitch angles, and turning radius and direction were allowed to vary. These poses were intentionally chosen such that the path would require significant turning maneuvers to visit all waypoints, rather than allowing an apparent path straight through the obstacle field. This exaggeration is intended to highlight the characteristics of the Dubins and clothoid 3-dimensional paths, but may not be representative of a likely real-world scenario.

Generation of the initial population took approximately 3 hours, while subsequent generations computed in approximately 1 hour each, using a Windows 7 Core i7 desktop computer with 8GB of RAM running the Matlab software. These results indicate that this optimization algorithm requires significant computational time, which prohibits use for online pose selection. However, use of a lower-overhead programming language such as C could substantially improve the convergence speed of the algorithm.

During this exploration of the algorithm's capabilities, the affinity value of the best individual improved from 0.7954 to 0.8533 in just 15 generations. The affinity values of the best individual at each generation are plotted in Figure 7-11. It must be recognized that the relative weighting of the four affinity components determines the definition of an optimal solution for the optimization problem, and selecting these weights differently would cause the same solutions to perform differently within the optimization.

The best individual in the initial population is plotted in Figure 7-15, with a top-down view of this solution shown in Figure 7-13. It is important to note that this is a 3-dimensional solution space, so although the top-down view may appear to show obstacle intersection, this is a 2-D projection, meaning that the actual path does not conflict the obstacle in the 3-D space, but instead may fly over the obstacle. It can be seen in these figures that the path of this individual is longer than necessary, but more importantly, the path crosses a threat zone. For this optimization problem, the criterion of highest importance is threat exposure. Thus, the presence of this threat crossing significantly decreases the affinity of this individual. In comparison, the best individual in generation 1, which is shown in Figure 7-14 and Figure 7-15, is longer that the initial best individual, however, it does not cross a threat, giving it a very good score for threat exposure but a lower score for path length. The best individual in generation 15 is plotted in Figure 7-16, and again a top-down view is seen in Figure 7-17. It can be seen that the final best individual's path is shorter and with shorter turning maneuvers than the best individual from generation 1, and it also avoids crossing a threat zone which is present in the initial individual, leading to lower threat exposure for the final path. Thus, according to the affinity function weighting criteria, this individual provides the best balance among the various affinity considerations of all solutions explored.

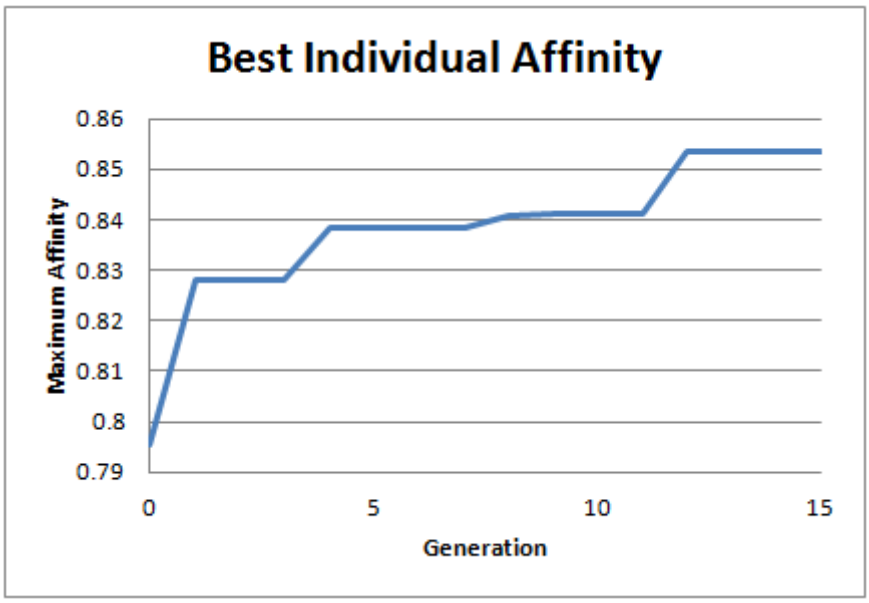

Figure 7-11—Best Infinity Improvement

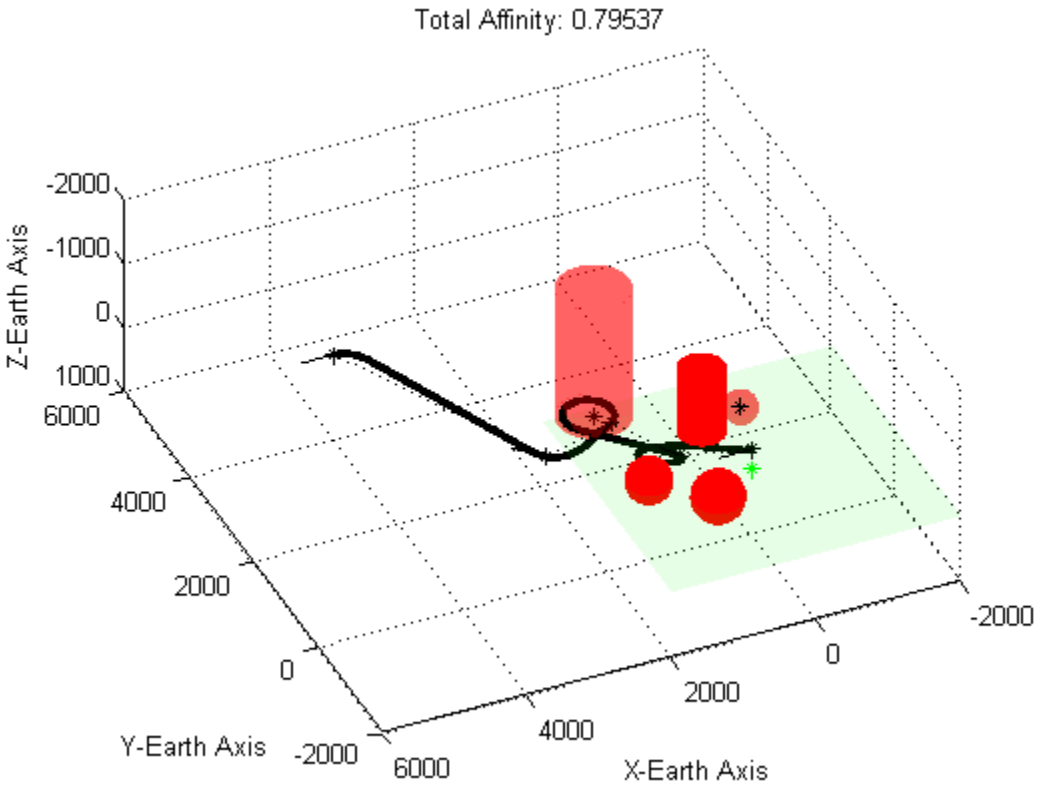

Figure 7-12—Best Initial Individual

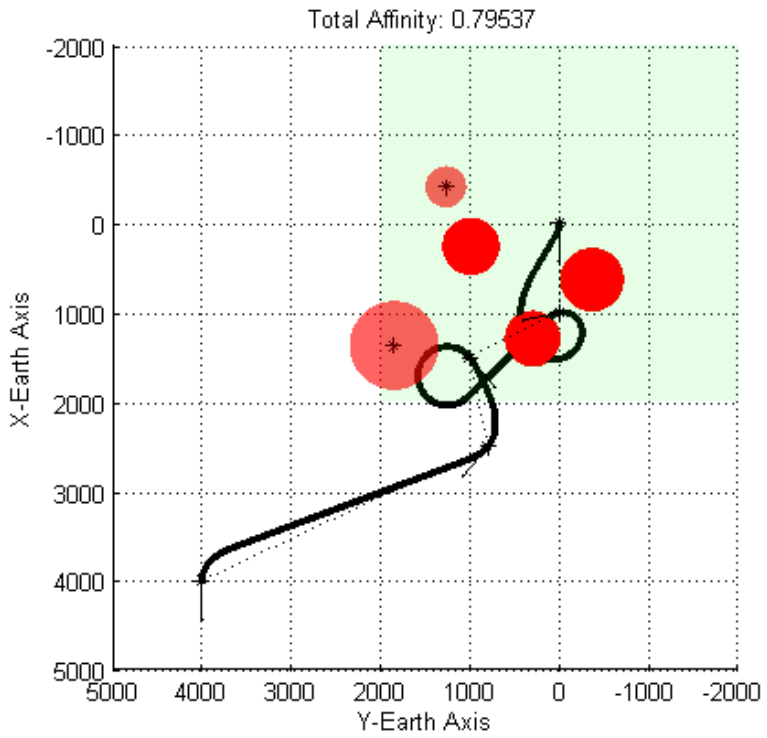

Figure 7-13—Best Initial Individual, Top-Down Projection

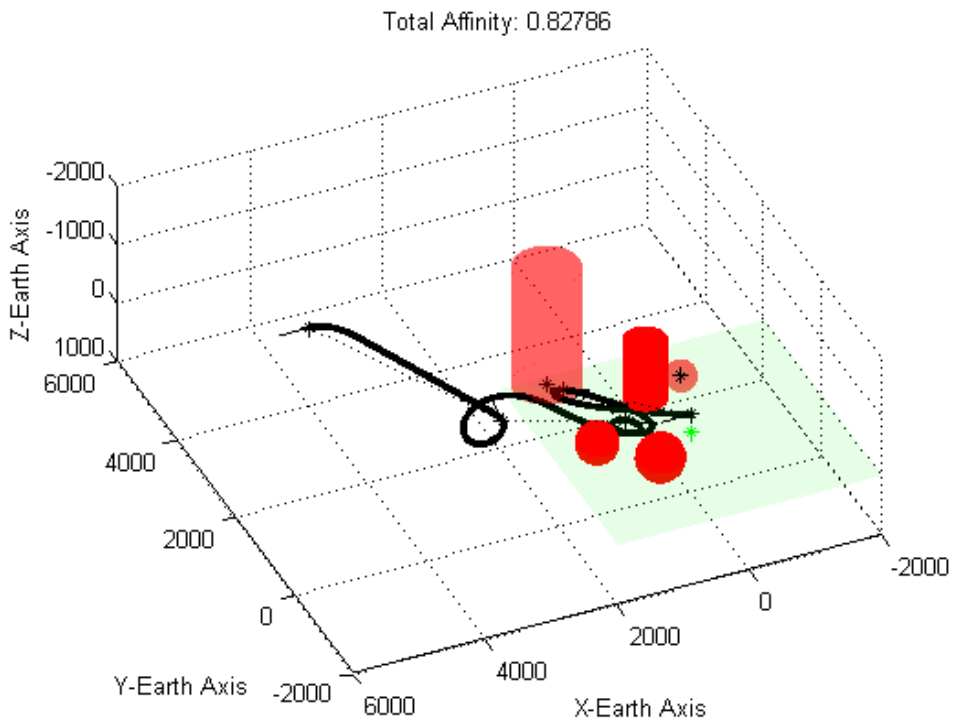

Figure 7-14—Best Individual in Generation 1

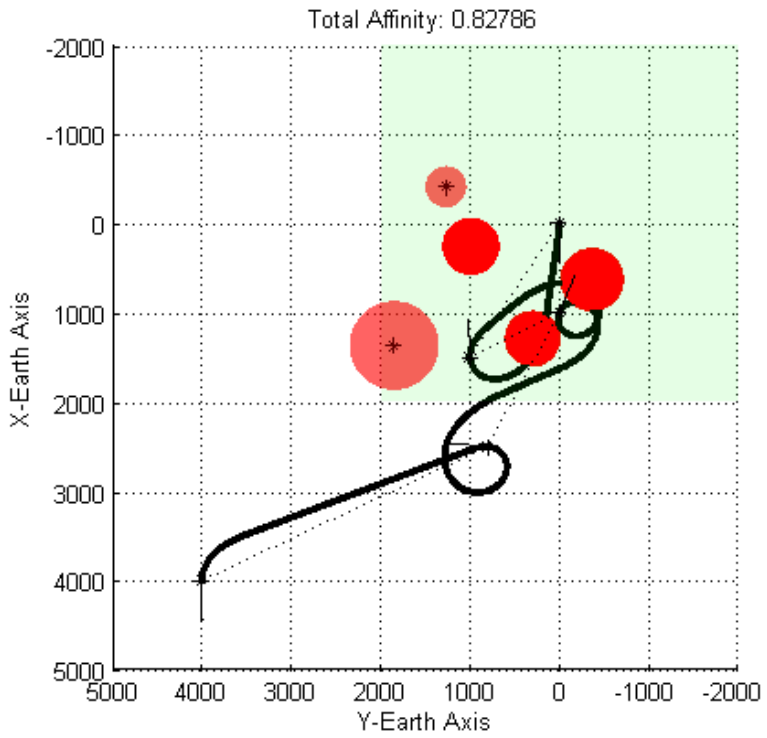

Figure 7-15—Best Individual in Generation 1, Top-Down Projection

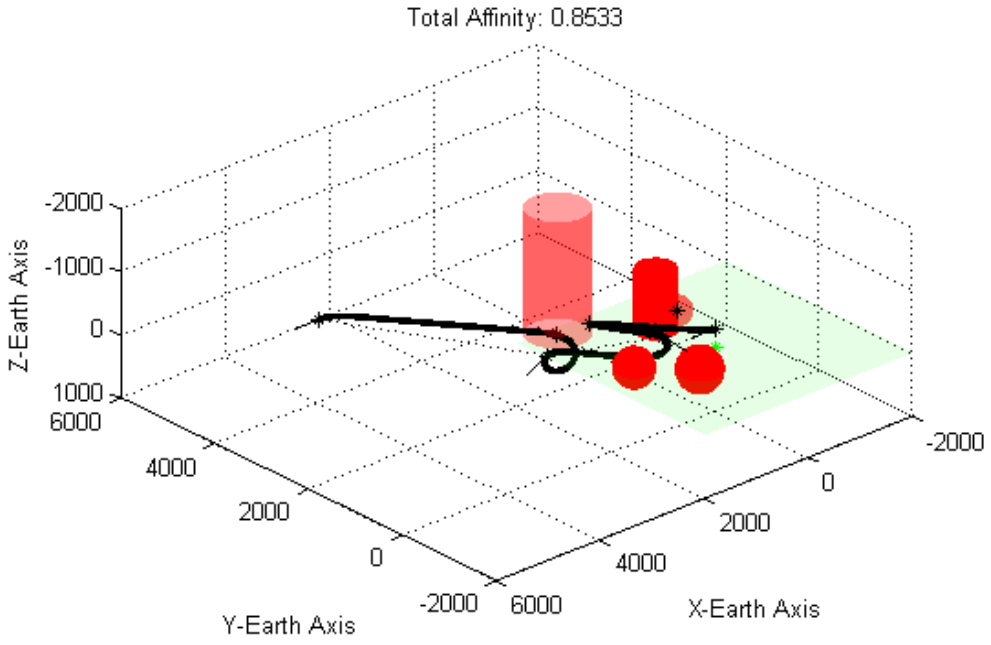

Figure 7-16—Best Final Individual

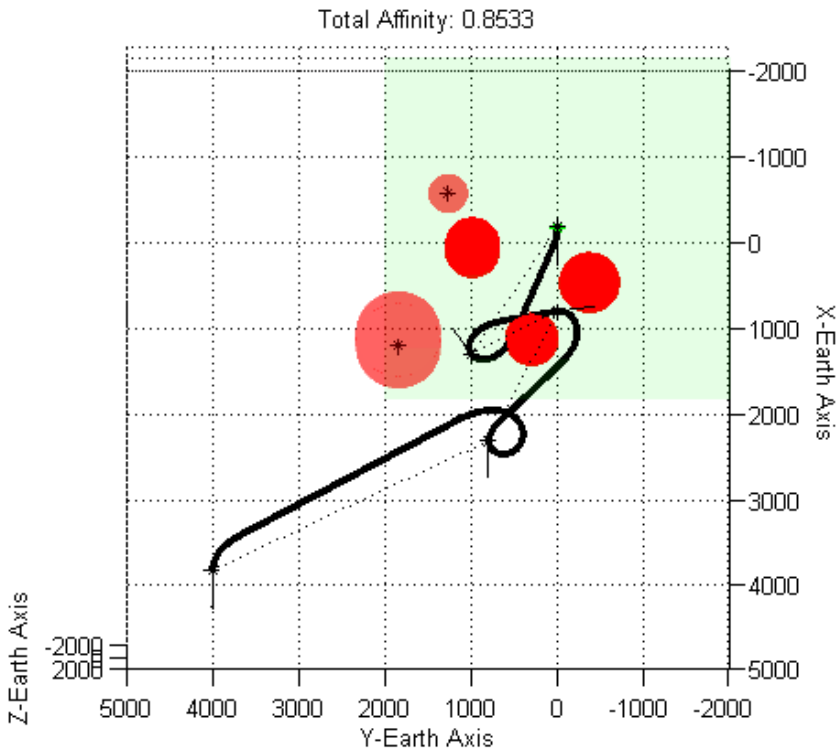

Figure 7-17—Best Final Individual, Top-Down Projection

In general, for a highly-complex environment wherein many points of interest should be visited, selection of poses which adequately and efficiently observe all of these areas without imposing risk on the aircraft is a non-trivial task. Thus, a mechanism such as this is needed to produce the poses, and consequently the path, through which the UAV should fly. However, in less stringent or less complex circumstances, computational time may indicate that heuristic methods may be preferable for pose selection.

In addition to reimplementing the method using a more efficient coding language, the computation time required for this method could be significantly improved if a more efficient numerical approach to solving the paths could be developed. However, as discussed in previous sections, the availability of an efficient numerical approach capable of solving these equations is limited due to the highly-nonlinear, complex nature of the problem. Minor improvements in convergence time could also be achieved through parallelization of the algorithm.

Chapter 8. DEMONSTRATION AND RESULTS

This chapter will be used to highlight the capabilities of the clothoid path planning methodology, in both 2 and 3 dimensions, by comparing it with the Dubins path planners. First, the WVU UAV Simulation Environment used to generate the results will be discussed, including some of its capabilities not necessary to this comparison. A set of performance metrics was defined in order to assess the relative performance of the clothoid and Dubins paths, which will be discussed. Then, results are presented for the 2-dimensional clothoid planner compared to the Dubins path planner. A similar comparison will then be presented for the 3-dimensional algorithms.

A. WVU Simulation Environment

An integrated simulation environment has been developed at West Virginia University (WVU) to support the complete development and testing of UAV autonomous flight control methodologies (92). This simulation environment and its user-friendly interface were developed in Matlab/Simulink for maximum portability and flexibility. It integrates with the FlightGear (93) flight simulation software to provide the user with visual cues. Additionally, a scenario definition dashboard was created within C# to allow the user to define the locations of risk zones and of the various aircraft. This dashboard also provides aerial visual feedback to the user in real-time. The modular architecture incorporated into the simulation environment includes the following components and capabilities. A more detailed description can be found in (63). Below in Figure 8-1 is a composite figure showing the typical desktop interface of the Simulation Environment, including UAVDashboard, FlightGear, Simulink models, and Guided User Interface. Figure 8-2 shows an example of the visualization provided by FlightGear.

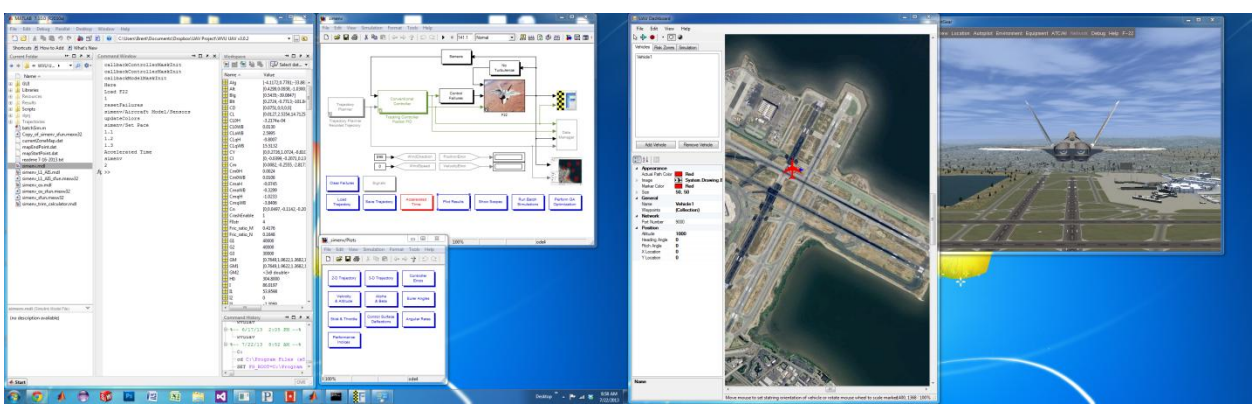

Figure 8-1—Simulation Environment Interface

Figure 8-2—FlightGear Environment

a. Aircraft Aerodynamic Models

The simulation environment has been outfitted with several different UAVs with which to experiment. These include the WVU YF-22 research aircraft, the NASA Generic Transport Model, the Pioneer, the Tigershark, and the OX. Various failure conditions have been integrated into many of these models to support the development and testing of fault tolerant control laws. Failures include actuator, sensor, structural, propulsion system failures, wing icing, GPS drop out, and severe weather. Figure 8-3 below shows an example of a UAV aerodynamic model within the Simulink environment. Each of the aircraft models have been updated to incorporate a modular architecture to allow more easy integration of the cooperating functionality, including the environment, path planners, and trajectory tracking control laws.

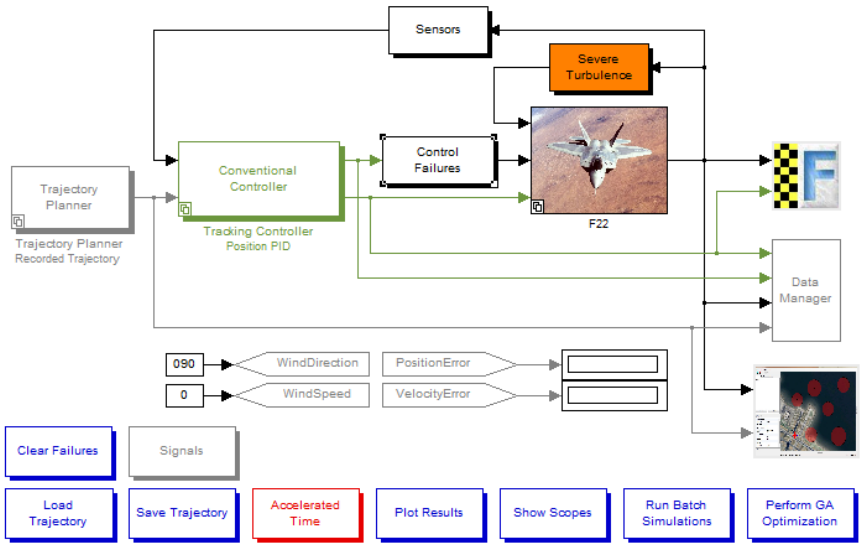

Figure 8-3—Simulink Aerodynamic Model of the WVU UAV Simulation Environment

In this environment, the blocks containing the aircraft dynamic model, path planner, trajectory tracking control laws, and the various failure categories are all switchable, meaning that to choose a different algorithm, a selection menu is available where the user needs simply select the desired algorithm, model, failure, etc. An example of such a menu is shown in Figure 8-4. This menu activates the control surface failures, whose model is shown in Figure 8-5. This generality drastically increases the usability and flexibility of the WVU UAV Simulation Environment.

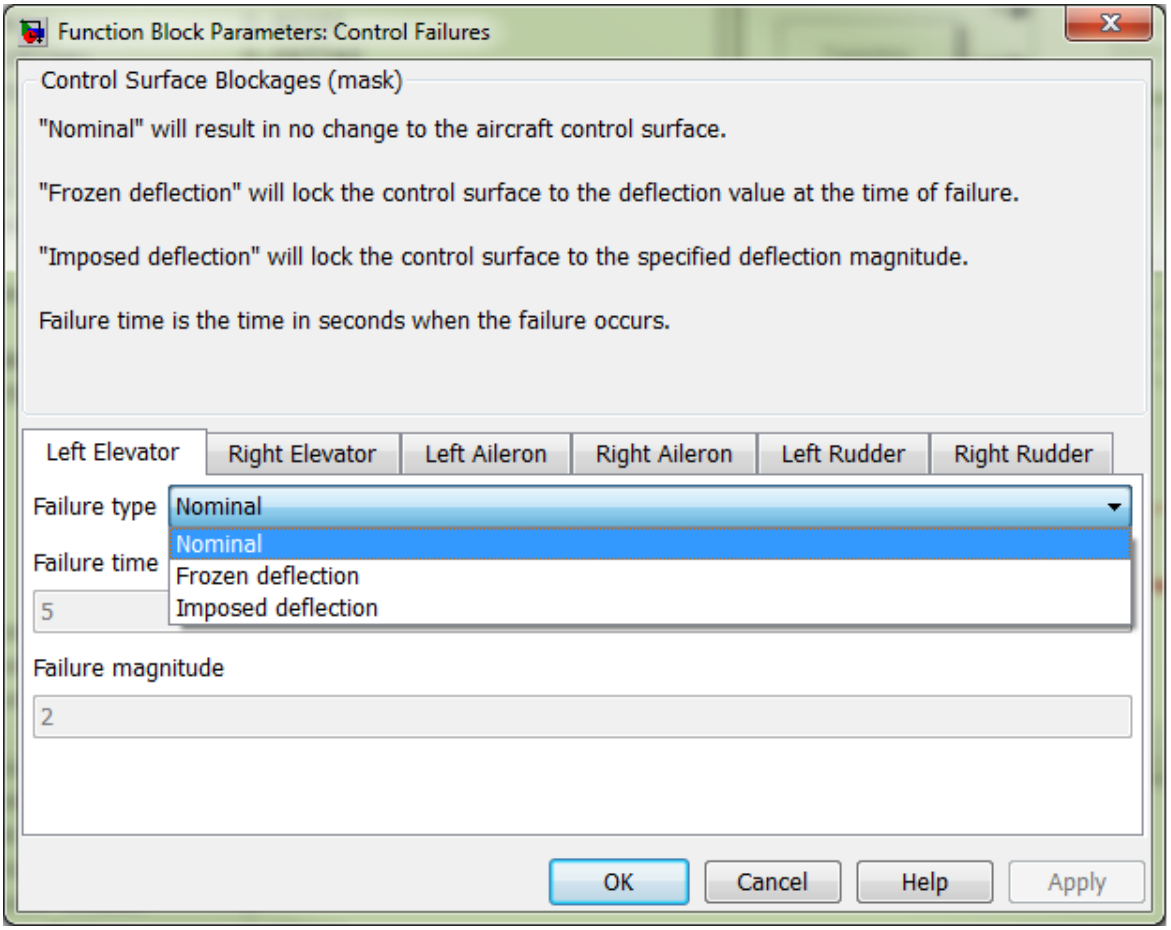

Figure 8-4—Control Surface Failure Selection Menu

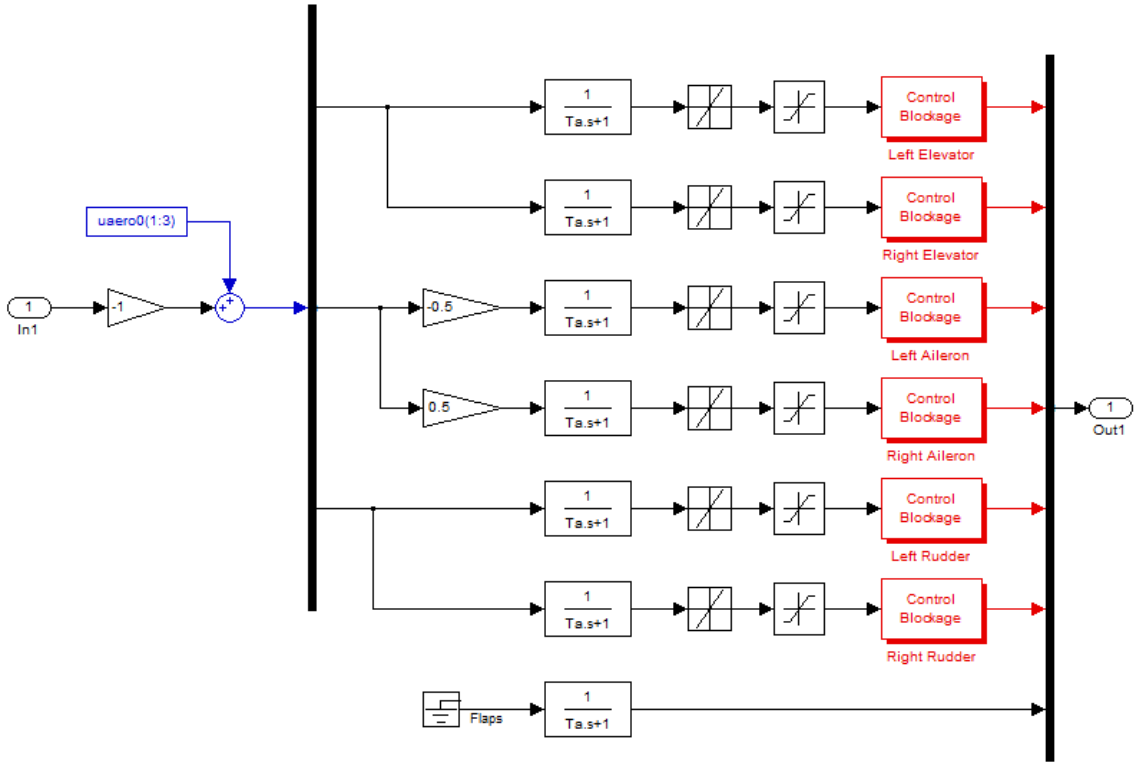

Figure 8-5—Control Surface Failure Model

b. Map Generation

As previously stated, the map generation software, called UAVDashboard, allows the user to select the initial position of the simulated vehicle, its objectives and waypoints, and, finally, the location, size, shape, and risk intensities of risk zones including obstacles. Once the initial conditions of the simulation vehicles and environment have been set, the UAVDashboard passes the information to the Simulink modules. During execution of the simulation, a 2-D visualization of the position and heading of the vehicles is presented to the user in real-time from a top-down perspective provided by the UAVDashboard. Figure 8-6 below displays the interface for the UAVDashboard.

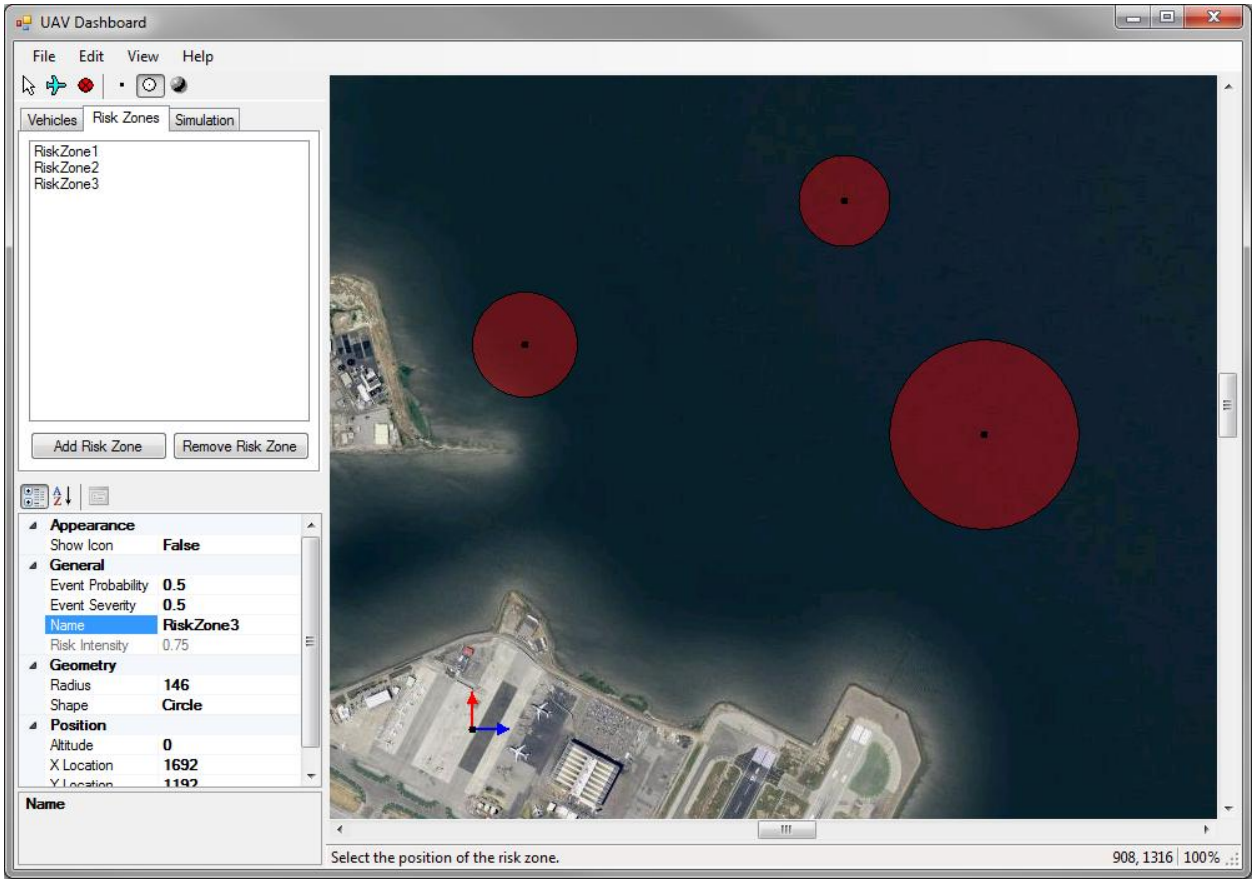

Figure 8-6—UAVDashboard Interface

c. Path Planning and Trajectory Generation

A complete library of trajectory generation methods relying on a host of path planning algorithms has been integrated into the simulation environment. These include all of the algorithms discussed previously in this document: Radar Voronoi, Polygon Voronoi, Zone Voronoi, Grid, Cell Decomposition, Potential Field, Enhanced Potential Field, Point of Interest Observation, Dubins Observation, Dubins Waypoint Tracking, 2D Clothoid Waypoint Tracking, 3D Dubins Waypoint Tracking, and 3D Clothoid Waypoint Tracking. The pose-based methods can be utilized with or without first applying the IPO algorithm to choose the poses. Thus these algorithms may take as inputs either the environment or the commanded poses. Figure 8-7 below illustrates the outermost block which works with the aircraft modules to provide the trajectory to the aircraft. Figure 8-8 then shows the model which runs inside the block to determine recalculation and path generation.

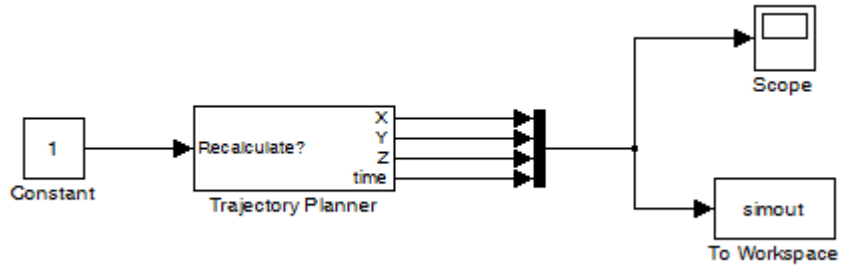

Figure 8-7—Simulink Trajectory Generation Block

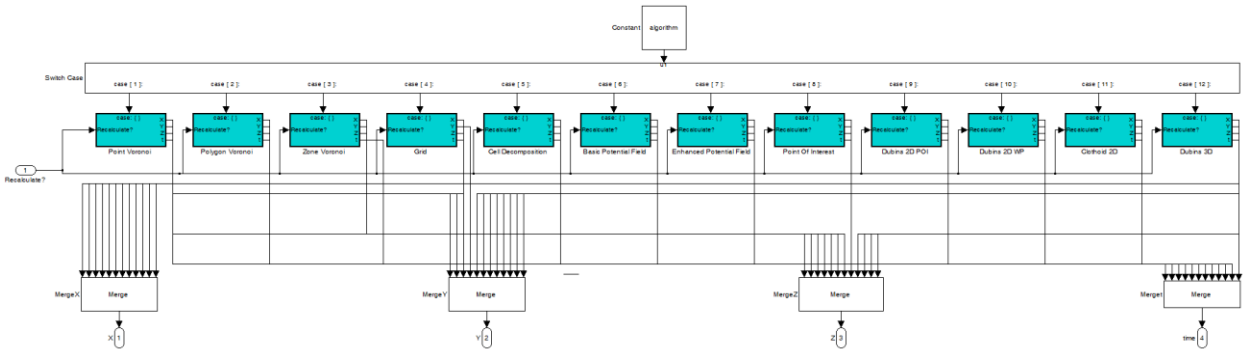

Figure 8-8—Trajectory Generation Simulink Model

d. Trajectory Tracking

A wide range of algorithms for autonomous flight control laws based on different strategies are implemented, which are expected to follow the waypoints produced by the trajectory generation module. They include conventional approaches such as linear, non-linear, and model predictive control; and adaptive approaches such as non-linear dynamic inversion and artificial neural network augmentation. In addition, the artificial immune system approach to failure detection, identification, and evaluation has been integrated to explore fault tolerance.

B. Performance Indices

In order to assess the various performance considerations relevant to trajectory tracking and to adequately compare the performance of the clothoid planners with the Dubins planners, as set of performance metrics were developed. These are discussed in greater detail in Wilburn et al. (94). A total performance index is assigned to the trajectory tracking performance of each trial based upon a number of factors. These are divided into two major categories: tracking accuracy and control surface activity. Thus, the total performance index is calculated as:

$$\mathrm{PI}_{\mathrm{UAV}} = \overline{\mathrm{w}}_{\mathrm{TT}}\mathrm{PI}_{\mathrm{TT}} + \overline{\mathrm{w}}_{\mathrm{CA}}\mathrm{PI}_{\mathrm{CA}} \qquad 253$$

where PI_{UAV} is the total performance index for the trial, PI_{TT} is the trajectory tracking performance index for the trial, PI_{CA} is the control activity performance index for the trial, and $\overline{w}_{TT}$ and $\overline{w}_{CA}$ are relative importance weights.

In order to obtain the trajectory tracking and control activity performance indices, many factors were considered. The trajectory tracking performance index is determined based upon 9 trajectory tracking specific performance parameters: the maximum absolute error, average absolute error, and standard deviation of tracking error in the XY-plane, along the Z-axis, and in 3-dimensional space.

The XY-plane tracking error is defined as:

$$e_{XY}(t) = \sqrt{[x_c(t) - x(t)]^2 + [y_c(t) - y(t)]^2} \tag{254}$$

the Z-axis error is defined as:

$$e_Z(t) = |z_c(t) - z(t)| \tag{255}$$

and the combined tracking error is define as:

$$e_{XYZ}(t) = \sqrt{[x_c(t) - x(t)]^2 + [y_c(t) - y(t)]^2 + [z_c(t) - z(t)]^2} \tag{256}$$

In these expressions, $x_c(t)$, $y_c(t)$, and $z_c(t)$ represent the commanded trajectory positions at time t, and $x(t)$, $y(t)$, and $z(t)$ represent the corresponding actual position values of the aircraft.

To define the tracking metrics. the average absolute tracking error is defined as:

$$\bar{e}_Q = \text{mean}\left(\left|e_Q(t)\right|\right) \tag{257}$$

the maximum absolute tracking error is defined as:

$$e_{max_Q} = \max\left(\left|e_Q(t)\right|\right) \tag{258}$$

and the tracking error standard deviation is defined as:

$$\hat{e}_Q = \text{STD}\left(e_Q(t)\right) \tag{259}$$

where the subscript Q represents XY, Z, or XYZ respectively. Thus, the trajectory tracking specific performance vector is defined as:

$$PV_{TT} = \left[\bar{e}_{XY}\ \bar{e}_Z\ \bar{e}_{XYZ}\ e_{max_{XY}}\ e_{max_Z}\ e_{max_{XYZ}}\ \hat{e}_{XY}\ \hat{e}_Z\ \hat{e}_{XYZ}\right]^T \tag{260}$$

The trajectory tracking specific performance index is then calculated as:

$$PI_{TT} = w_{TT} PV_{TT} \tag{261}$$

where w_{TT} is a 9-element row vector containing the normalization weights for each of the elements in the performance vector.

The control surface activation performance index is computed based upon the integral of the absolute value of the rate of change of deflection of the aileron, stabilator, rudder, and throttle, as well as the percentage of samples at saturation for each of these control surfaces. Letting δ_a, δ_e, δ_r, and δ_t be the commanded deflections of the aileron, stabilator, rudder, and throttle respectively, the control-activity related parameters listed above are defined as follows:

Integral of aileron deflection rate of change:

$$I\dot{\delta}_a = \frac{1}{T}\int_0^T \left|\dot{\delta}_a(t)\right| dt \tag{262}$$

Integral of stabilator deflection rate of change:

$$I\dot{\delta}_e = \frac{1}{T}\int_0^T \left|\dot{\delta}_e(t)\right| dt \tag{263}$$

Integral of rudder deflection rate of change:

$$I\dot{\delta}_r = \frac{1}{T}\int_0^T \left|\dot{\delta}_r(t)\right| dt \tag{264}$$

Integral of throttle command rate of change:

$$I\dot{\delta}_t = \frac{1}{T}\int_0^T \left|\dot{\delta}_t(t)\right| dt \tag{265}$$

Aileron saturation index:

$$S_{\delta_a} = \frac{100}{T}\int_0^T \tilde{\delta}_a(t) dt \tag{266}$$

where:

$$\tilde{\delta}_a(t) = \begin{cases} 0 \text{ for } \delta_a < \delta_{a_{max}} \\ 1 \text{ for } \delta_a \geq \delta_{a_{max}} \end{cases} \tag{267}$$

Stabilator saturation index:

$$S_{\delta_e} = \frac{100}{T}\int_0^T \left(\tilde{\delta}_{e1}(t) + \tilde{\delta}_{e2}(t)\right) dt \tag{268}$$

where:

$$\tilde{\delta}_{e1}(t) = \begin{cases} 0 \text{ for } \delta_e < \delta_{e_{max}} \\ 1 \text{ for } \delta_e \geq \delta_{e_{max}} \end{cases} \tag{269}$$

and

$$\tilde{\delta}_{e2}(t) = \begin{cases} 0 \text{ for } \delta_e > \delta_{e_{min}} \\ 1 \text{ for } \delta_e \leq \delta_{e_{min}} \end{cases} \tag{270}$$

Rudder saturation index:

$$S_{\delta_r} = \frac{100}{T}\int_0^T \tilde{\delta}_r(t)dt \tag{271}$$

where:

$$\tilde{\delta}_r(t) = \begin{cases} 0 \text{ for } \delta_r < \delta_{r_{max}} \\ 1 \text{ for } \delta_r \geq \delta_{r_{max}} \end{cases} \tag{272}$$

Throttle saturation index:

$$S_{\delta_t} = \frac{100}{T}\int_0^T \tilde{\delta}_t(t)dt \tag{273}$$

where:

$$\tilde{\delta}_t(t) = \begin{cases} 0 \text{ for } \delta_t < \delta_{t_{max}} \\ 1 \text{ for } \delta_t \geq \delta_{t_{max}} \end{cases} \tag{274}$$

Thus, the control activity specific performance vector is defined as:

$$PV_{CA} = \left[\, I\dot{\delta}_e \;\; I\dot{\delta}_a \;\; I\dot{\delta}_r \;\; I\dot{\delta}_t \;\; S_{\delta_e} \;\; S_{\delta_a} \;\; S_{\delta_r} \;\; S_{\delta_t} \right]^T \tag{275}$$

From this equation, the control activity specific performance index is computed as:

$$PI_{CA} = w_{CA} P V_{CA} P \quad (276)$$

where w_{CA} is an 8-element row vector containing the normalization weights corresponding to each element in the performance vector, and P is the percentage of trajectory points which were within a threshold of the commanded path. This percentage multiplier is necessary to ensure that a trial which crashes cannot get an artificially inflated performance index due to the lack of control activity throughout the trial.

Using the above defined values, the global trajectory tracking performance index is calculated.

C. Comparison of 2D Clothoid and Dubins Trajectory Tracking Performance

In order to verify and justify any performance improvements yielded by using a clothoid arc rather than a circular arc, a series of comparisons were conducted, both under nominal and failure conditions. Three different paths were generated, based upon the same poses, for each the clothoid and the Dubins 2-dimensional path planners. These can be seen in Figure 8-9 through Figure 8-14 below. These were flown using the WVU YF-22 UAV model, which has reasonably quick response rates, and is also outfitted with an extensive suite of failure conditions. Several trajectory tracking controllers are available for this UAV, but the simple position proportional-integral-derivative controller was used for this comparison, as it contains no additional failure compensation and represents a common trajectory tracking approach. This comparison is intended to demonstrate the potential performance improvements possible without alteration of the aircraft hardware or controller. For all nominal conditions, the trajectory tracking performance of the clothoid path was significantly higher than the trajectory tracking performance of the Dubins path planner. Additionally, under failure conditions, the clothoid path consistently yielded better trajectory tracking performance than the Dubins path, although for two of the stuck elevator failures, both simulations still ultimately crashed. This can be attributed to the lack of additional failure compensation present in the control laws; this occurrence could be mitigated assuming a more robust trajectory tracking controller may be used. In spite of this, even on these failed missions, the trajectory tracking performance yielded by the clothoid path was still higher than the performance of the Dubins path. The full trajectory tracking results can be seen in Table 8-1. Following this table, several specific examples of these results will be illustrated and discussed. The full results of this comparison are included in Appendix B.

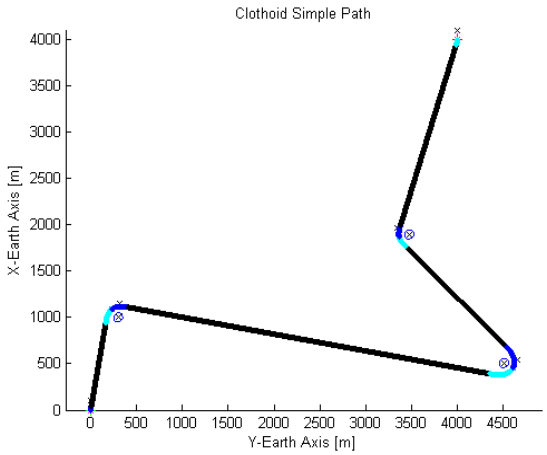

Figure 8-9—Clothoid Simple Trajectory

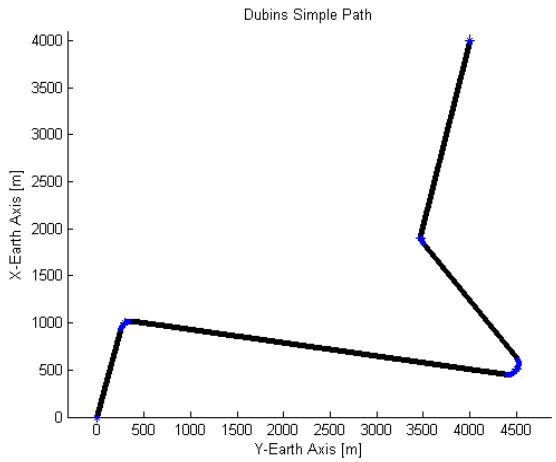

Figure 8-10—Dubins Simple Trajectory

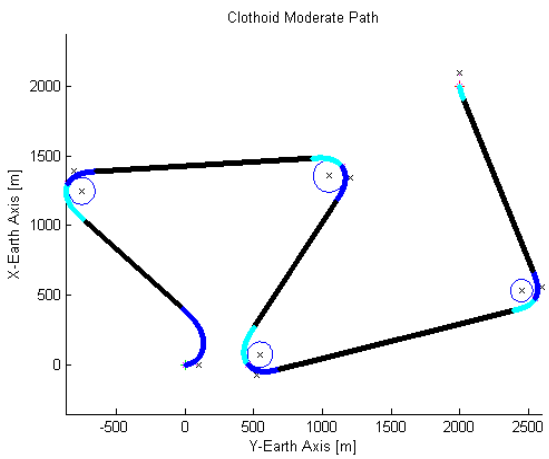

Figure 8-11—Clothoid Moderate Trajectory

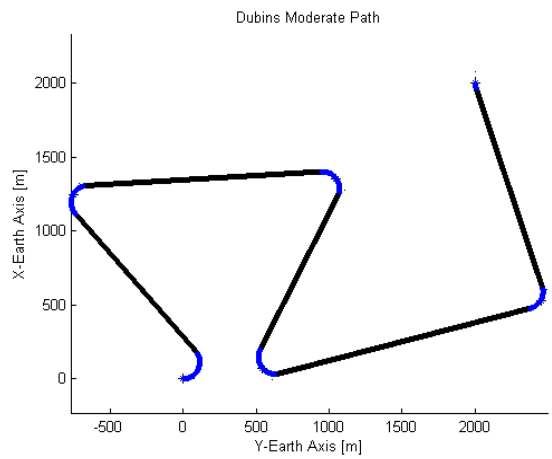

Figure 8-12—Dubins Moderate Trajectory

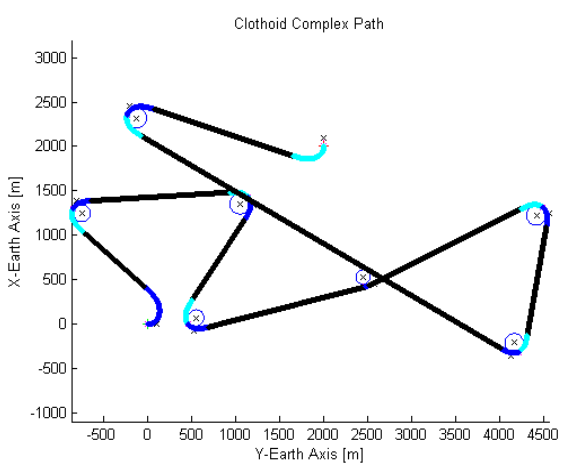

Figure 8-13—Clothoid Complex Trajectory

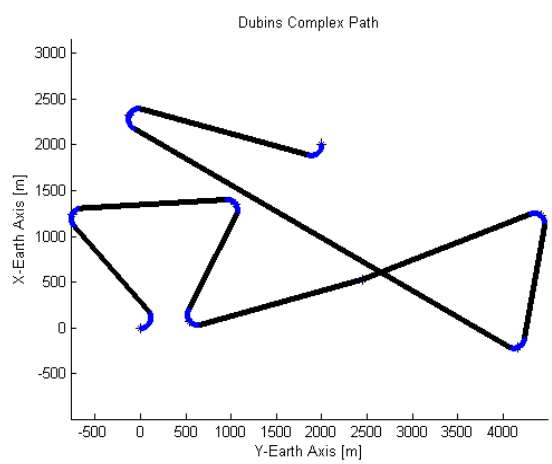

Figure 8-14—Dubins Complex Trajectory

Table 8-1—2-D Clothoid vs Dubins Trajectory Tracking Comparison Results

Clothoid VS Dubins Path Tracking Results (Tracking Controller=PPID)							
	Condition	**Tracking PI**		**Control PI**		**Total PI**	
		Dubins	Clothoid	Dubins	Clothoid	Dubins	Clothoid
Simple	Nominal	0.359	0.791	0.950	0.981	0.536	0.848
	Left Aileron stuck at 7deg	0	0.303	0.764	0.796	0.229	0.451
	Left Elevator stuck at 7deg	0	0.0206	0.879	0.962	0.264	0.303
	Left Rudder Stuck at 8deg	0.148	0.487	0.939	0.973	0.385	0.633
	High Turbulence	0.0464	0.111	0.741	0.753	0.255	0.304
Moderate	Nominal	0.347	0.419	0.866	0.962	0.503	0.582
	Left Aileron stuck at 2deg	0	0.317	0.801	0.913	0.240	0.496
	Left Elevator stuck at 2deg	0	0	0.775	0.886	0.233	0.266
	Left Rudder Stuck at 8deg	0	0.291	0.787	0.942	0.236	0.486
	High Turbulence	0.0126	0.188	0.690	0.731	0.216	0.351
Complex	Nominal	0.349	0.483	0.903	0.975	0.515	0.631
	Left Aileron stuck at 2deg	0	0.353	0.814	0.956	0.244	0.534
	Left Elevator stuck at 2deg	0	0	0.782	0.914	0.235	0.274
	Left Rudder Stuck at 8deg	0	0.314	0.787	0.944	0.236	0.503
	High Turbulence	0.119	0.135	0.724	0.752	0.301	0.321

To demonstrate the nominal trajectory tracking performance comparison between these two path planners, Figure 8-15 and Figure 8-16. These figures show the commanded, in magenta, and actual, in black, paths of the aircraft in response to the two complex trajectories under nominal conditions. It is evident, not just from the numerical performance index, but visibly, that the clothoid path was significantly easier for the aircraft to track.

To accentuate the capabilities of the clothoid path versus the Dubins path, Figure 8-17 and Figure 8-18 show the trajectory tracking performance of the moderate paths with the aircraft suffering an elevator stuck at 2 degrees deflection. For the clothoid path, the aircraft was able to reach the goal location, while for the Dubins path the aircraft lost the path entirely. Similar occurrences were also observed for the aileron and elevator failures for the simple pose scenario, aileron and rudder failures for the moderate pose scenario, and for aileron and rudder failures for the complex pose scenario. These results indicate that the clothoid path was easier to track under these failed circumstances than the Dubins path, however, this effect may also be contributed to the controller being inadequate. Thus, additional testing using a more robust controller is necessary to verify the source of these improvements.

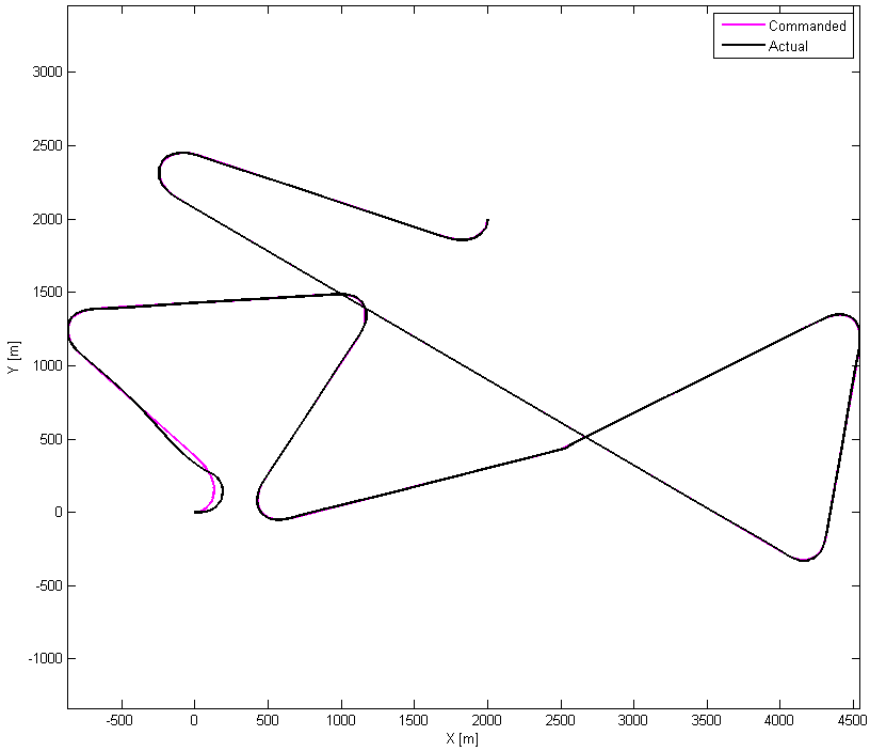

Figure 8-15—Commanded and Actual Clothoid Trajectories for the Complex Pose Scenario at Nominal Conditions

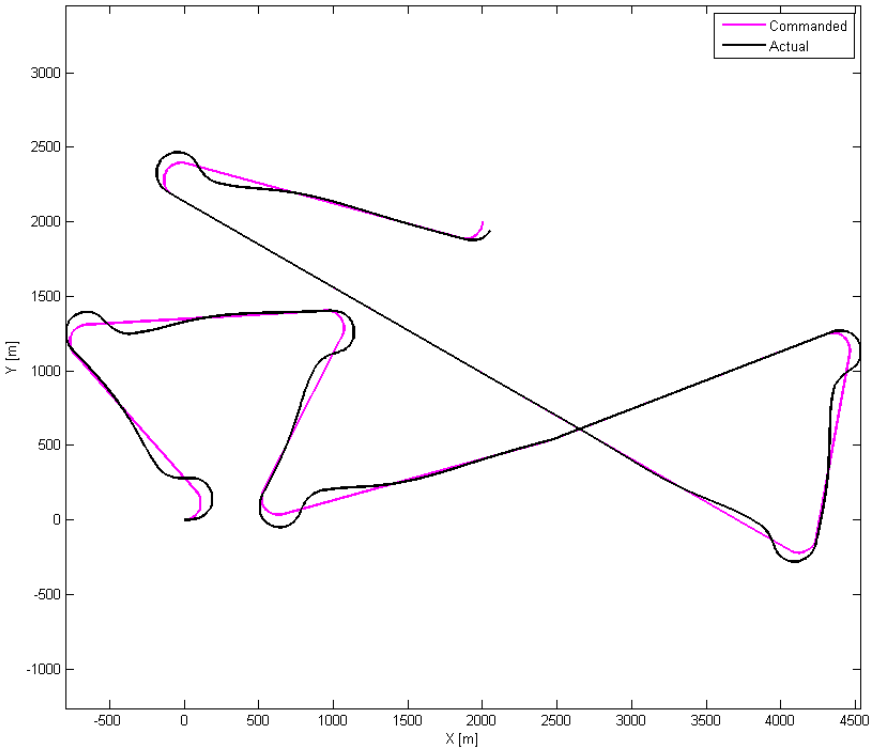

Figure 8-16—Commanded and Actual Dubins Trajectories for the Complex Pose Scenario at Nominal Conditions

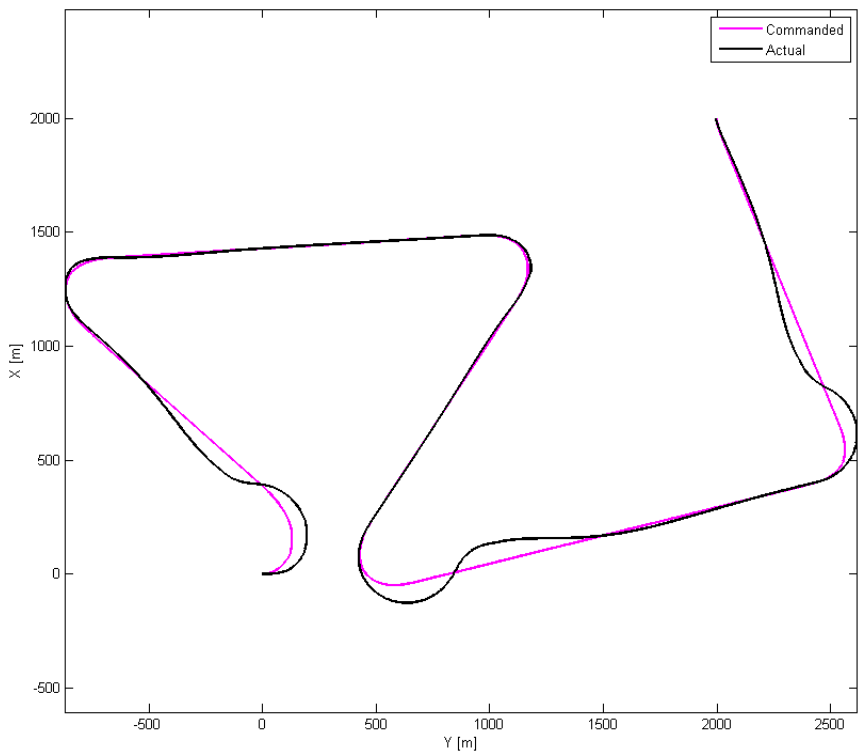

Figure 8-17—Commanded and Actual Clothoid Trajectories for the Moderate Pose Scenario with Elevator Stuck at 2 Degrees Deflection

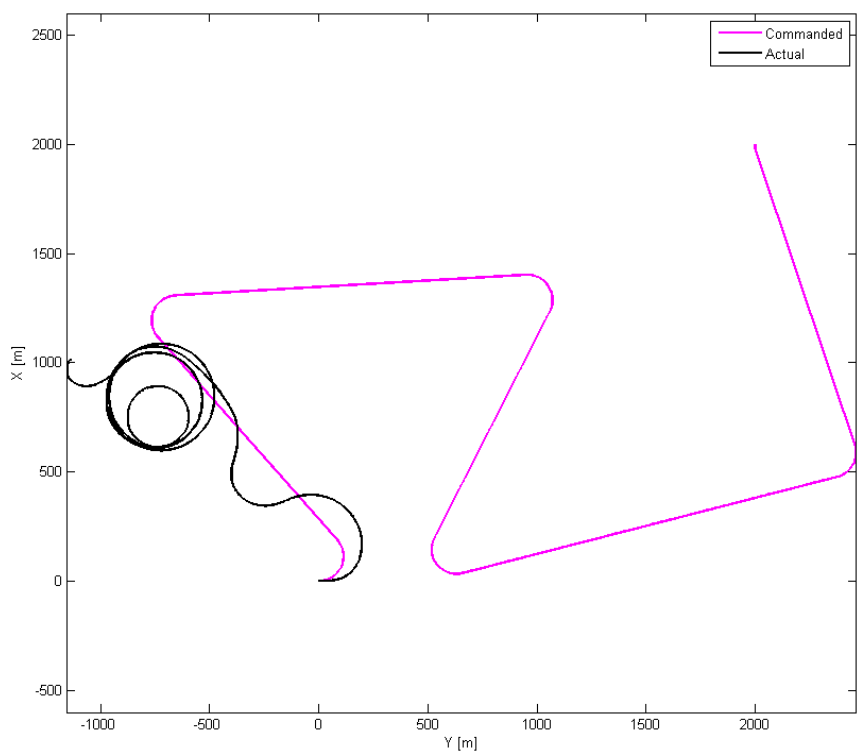

Figure 8-18— Commanded and Actual Dubins Trajectories for the Moderate Pose Scenario with Elevator Stuck at 2 Degrees Deflection

Other improvements were noted for many of the cases. For instance, observable trajectory tracking improvements are yielded by the clothoid as compared to the Dubins for the situation of high turbulence for this situation. These tracking results are shown in Figure 8-19 and Figure 8-20. Although turbulence is a highly variable effect, the improvements observed for this trial provide an encouraging result, indicating that use of the clothoid path planner as opposed to the Dubins path planner could provide improved resilience to turbulence. Turbulence is a highly likely occurrence for an aircraft, which is an issue of even higher concern for UAVs which tend to be smaller and therefore more susceptible to wind disturbances. Consequently, turbulence causes degradation in trajectory tracking performance, which can lead to obstacle collision in close environments, and therefore, catastrophic failure for the aircraft. Improved resilience to turbulence will lead to an overall more robust, more reliable UAV platform. Thus, further investigation should be performed in order to verify if improvements in trajectory tracking performance are in reality capable with use of the clothoid-based path.

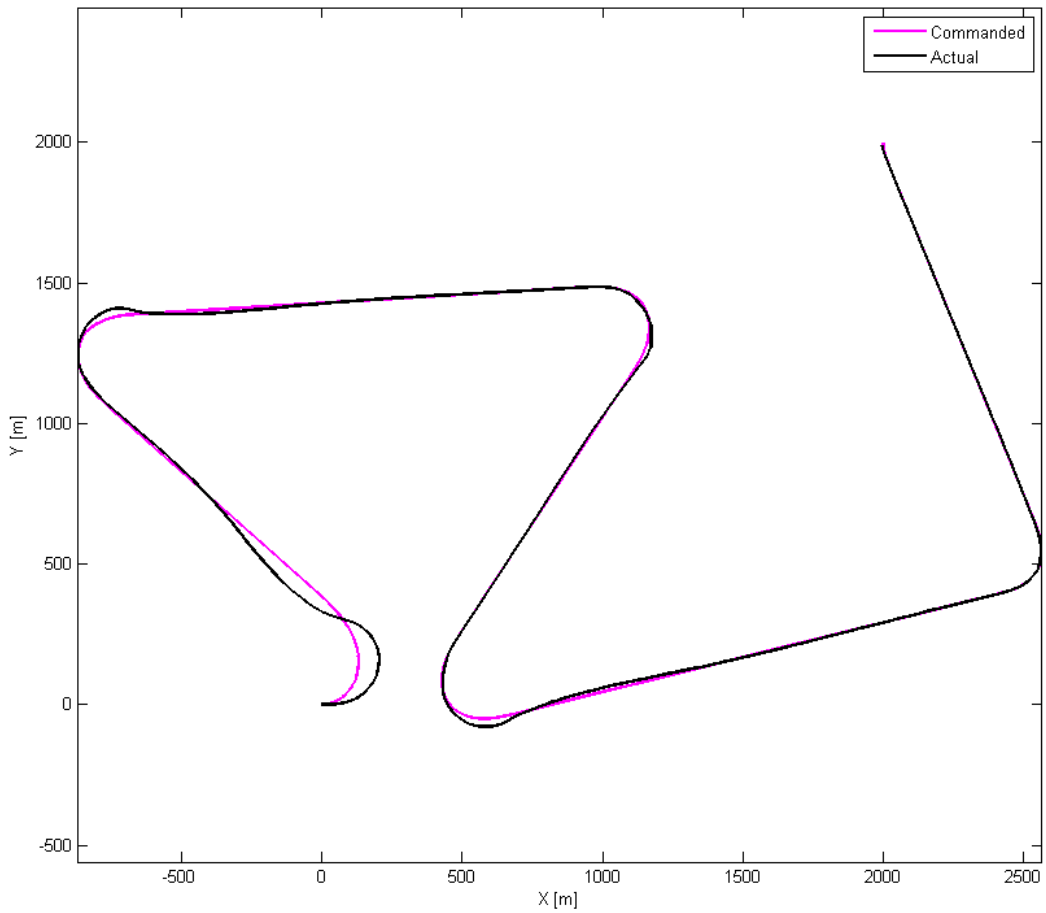

Figure 8-19—Commanded and Actual Clothoid Trajectories for the Moderate Pose Scenario with High Wind Turbulence

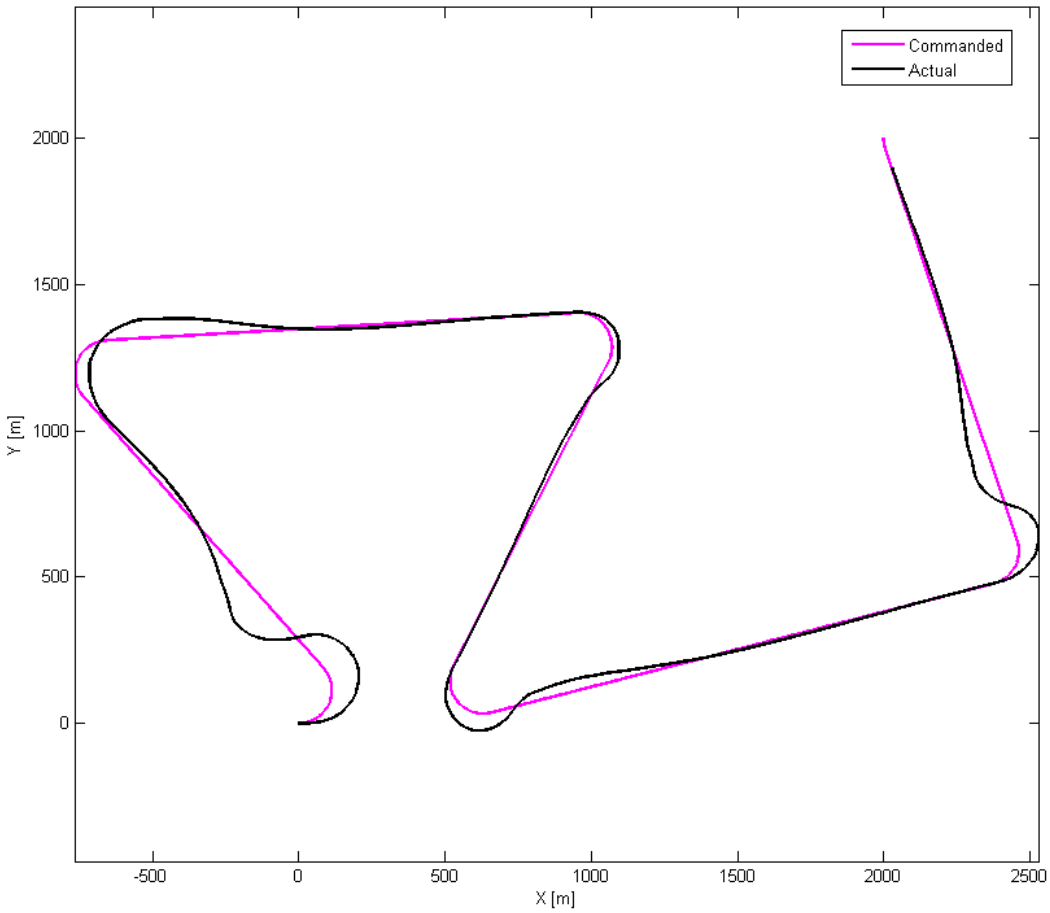

Figure 8-20—Commanded and Actual Dubins Trajectories for the Moderate Pose Scenario with High Wind Turbulence

D. Comparison of 3D Clothoid and Dubins Trajectory Tracking Performance

To support and justify the benefits of the 3-dimensional clothoid path planner, paths were generated for a set of poses for both the 3-dimensional clothoid and the 3-dimensional Dubins path planners. These were then flown in simulation using the WVU YF-22 UAV model under both nominal and off-nominal conditions. These tests were conducted for the position proportional-integral-derivative (PPID) trajectory tracking controller and the outer-loop non-linear dynamic inversion controller (ONLDI). The PPID controller was selected for this comparison since it represents an elementary trajectory tracking approach which generates control surface deflection commands directly based upon the lateral, vertical, and forward position errors. The ONLDI controller was selected for testing as it is a more complex and robust trajectory tracking mechanism which uses the dynamic inversion of the linearized aircraft model to generate the commanded surface deflections. The paths used for this comparison were generated from three sets of poses, of increasing complexity. The paths produced by the 3D clothoid and 3D Dubins planners are shown below in Figure 8-21 through Figure 8-26.

Each of these paths was tested against the YF-22 model in response to normal conditions, control surface failures, and high wind turbulence. It may often be the case that a system and its control laws are unalterable by the user of the system, but better tracking performance is desired. In such a situation, only the external commands to the system may be altered. Thus, these comparisons are intended to represent the potential performance outcomes possible without alteration of the aircraft or its control laws and only modifying the commanded trajectory. This "black box" comparison is also intended to isolate and highlight the differences imposed on the trajectory tracking performance by the path geometries. The complete performance index results for the PPID controller are shown in Table 8-2 and the performance index results of the same tests using the ONLDI controller are displayed in Table 8-3. It can readily be seen from these results that this aircraft was better able to track the clothoid-based path than the Dubins-based path, as hypothesized. The unabridged results of this comparison have been included in Appendix C.

To assess the nominal trajectory tracking performance of the two planners, see Figure 8-27 through Figure 8-34. These figures show the 3D trajectory tracking results, and the trajectory tracking results projected into the X-Y plane, for each of the planners in response to the circling path above, with the commanded path in magenta and the flown path in black. It is readily visible from these plots that the clothoid trajectory was more closely tracked by the aircraft for both the PPID and ONLDI trajectory tracking controllers.

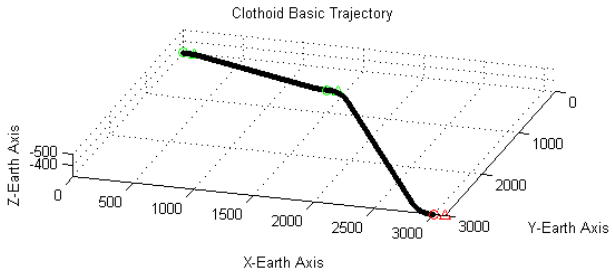

Figure 8-21—3D Clothoid Basic Trajectory

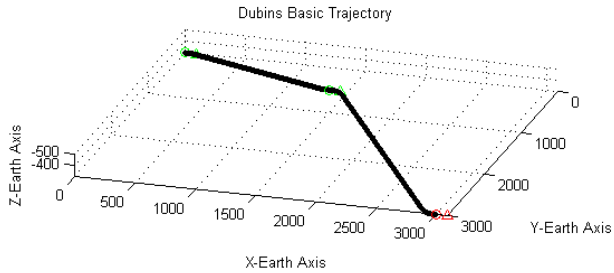

Figure 8-22—3D Dubins Basic Trajectory

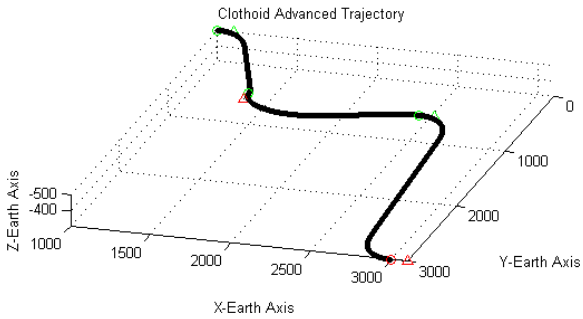

Figure 8-23—3D Clothoid Advanced Trajectory

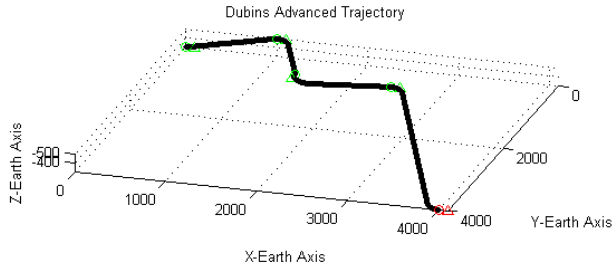

Figure 8-24—3D Dubins Advanced Trajectory

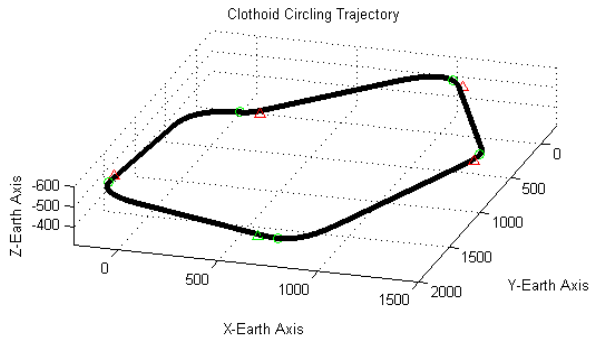

Figure 8-25—3D Clothoid Circling Trajectory

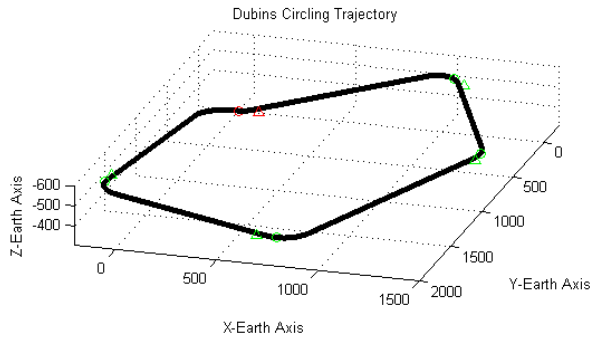

Figure 8-26—3D Dubins Circling Trajectory

Table 8-2—3-D Clothoid vs Dubins Trajectory Tracking Comparison Results for PPID Controller

	Condition	Tracking PI		Control PI		Total PI	
		Dubins	Clothoid	Dubins	Clothoid	Dubins	Clothoid
Basic	Nominal	0.570	0.651	0.959	0.976	0.687	0.749
	Left Aileron stuck at 2deg	0.560	0.579	0.943	0.959	0.675	0.693
	Left Elevator stuck at 2deg	0.596	0.691	0.964	0.975	0.707	0.777
	Left Rudder Stuck at 8deg	0.376	0.417	0.787	0.957	0.500	0.579
	High Turbulence	0.334	0.334	0.637	0.626	0.425	0.421
Advanced	Nominal	0.384	0.456	0.850	0.928	0.524	0.597
	Left Aileron stuck at 2deg	0.389	0.392	0.339	0.334	0.374	0.375
	Left Elevator stuck at 2deg	0.336	0.058	0.340	0.336	0.337	0.141
	Left Rudder Stuck at 8deg	0.373	0.512	0.846	0.933	0.515	0.638
	High Turbulence	0.300	0.327	0.570	0.642	0.381	0.422
Circle	Nominal	0.468	0.522	0.908	0.926	0.600	0.643
	Left Aileron stuck at 2deg	0.376	0.522	0.874	0.919	0.526	0.641
	Right Aileron stuck at 2 deg	0.373	0.501	0.755	0.916	0.487	0.625
	Left Elevator stuck at 2deg	0.135	0.487	0.871	0.922	0.356	0.618
	Right Elevator stuck at 2deg	0.318	0.503	0.802	0.924	0.464	0.629
	Left Rudder Stuck at 8deg	0.000	0.479	0.226	0.920	0.068	0.611
	High Turbulence	0.269	0.261	0.617	0.595	0.373	0.361

Table 8-3—3-D Clothoid vs Dubins Trajectory Tracking Comparison Results for ONLDI Controller

	Condition	Tracking PI		Control PI		Total PI	
		Dubins	Clothoid	Dubins	Clothoid	Dubins	Clothoid
Basic	Nominal	0.551	0.594	0.755	0.768	0.612	0.646
	Left Aileron stuck at 2deg	0.449	0.470	0.692	0.719	0.522	0.545
	Left Elevator stuck at 2deg	0.457	0.626	0.732	0.763	0.539	0.667
	Left Rudder Stuck at 8deg	0.375	0.465	0.640	0.713	0.455	0.540
	High Turbulence	0.580	0.600	0.577	0.584	0.579	0.595
Advanced	Nominal	0.458	0.532	0.689	0.745	0.527	0.596
	Left Aileron stuck at 2deg	0.375	0.375	0.523	0.555	0.419	0.429
	Left Elevator stuck at 2deg	0.280	0.325	0.564	0.581	0.365	0.402
	Left Rudder Stuck at 8deg	0.376	0.619	0.635	0.743	0.453	0.656
	High Turbulence	0.434	0.494	0.538	0.569	0.465	0.517
Circle	Nominal	0.620	0.884	0.726	0.760	0.652	0.847
	Left Aileron stuck at 2deg	0.365	0.617	0.612	0.731	0.439	0.651
	Right Aileron stuck at 2 deg	0.374	0.576	0.538	0.740	0.423	0.625
	Left Elevator stuck at 2deg	0.394	0.868	0.688	0.751	0.482	0.833
	Right Elevator stuck at 2deg	0.373	0.704	0.642	0.754	0.454	0.719
	Left Rudder Stuck at 8deg	0.373	0.684	0.650	0.751	0.456	0.704
	High Turbulence	0.550	0.814	0.577	0.582	0.558	0.744

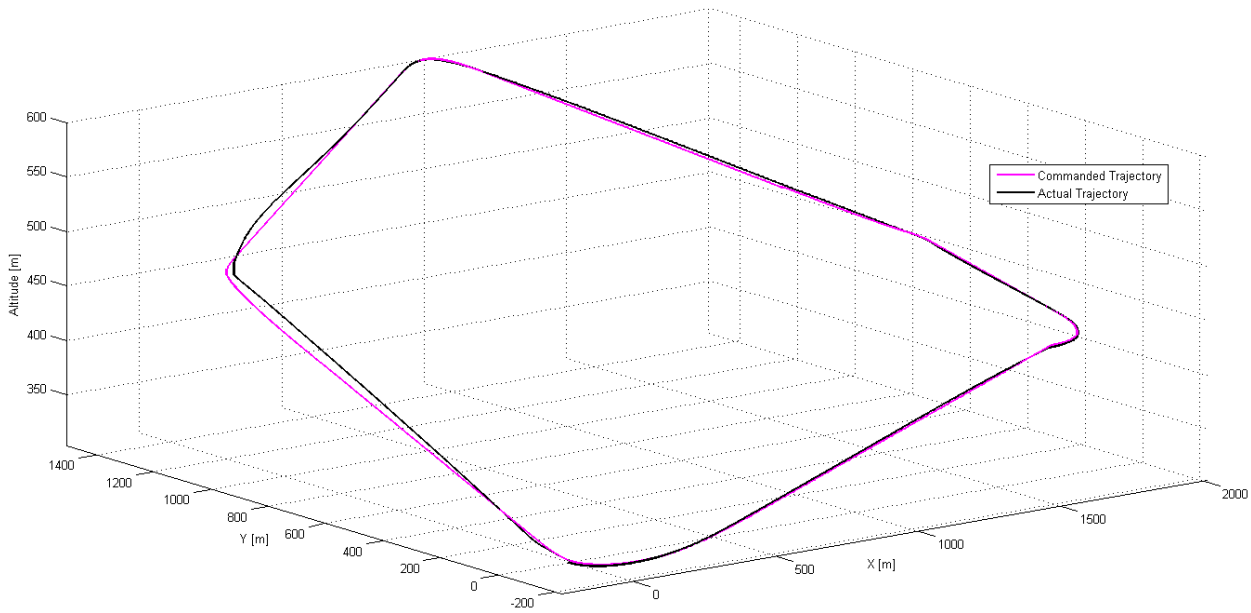

Figure 8-27—Clothoid Nominal Trajectory Tracking with PPID, 3D View

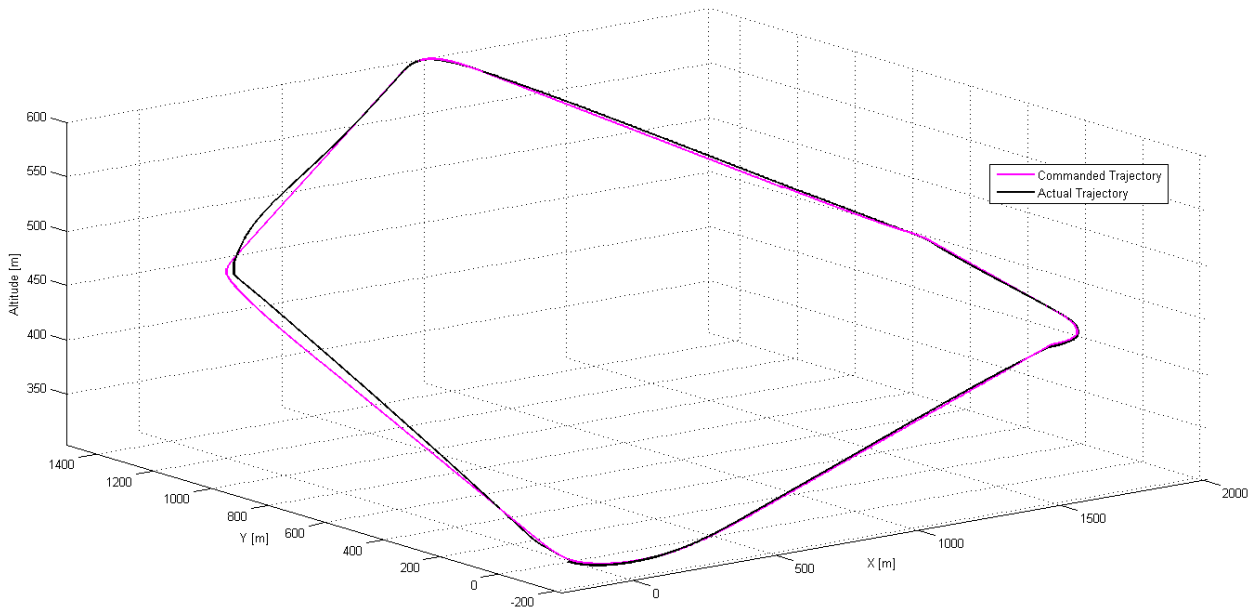

Figure 8-28—Dubins Nominal Trajectory Tracking with PPID, 3D View

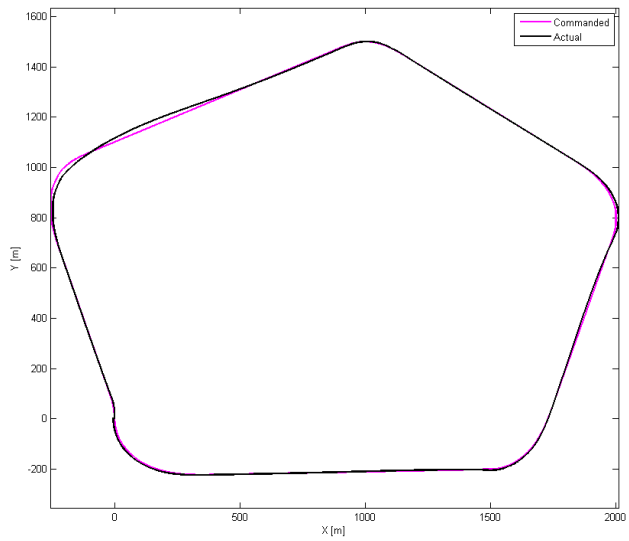

Figure 8-29—Clothoid Nominal Trajectory Tracking with PPID, 2D View

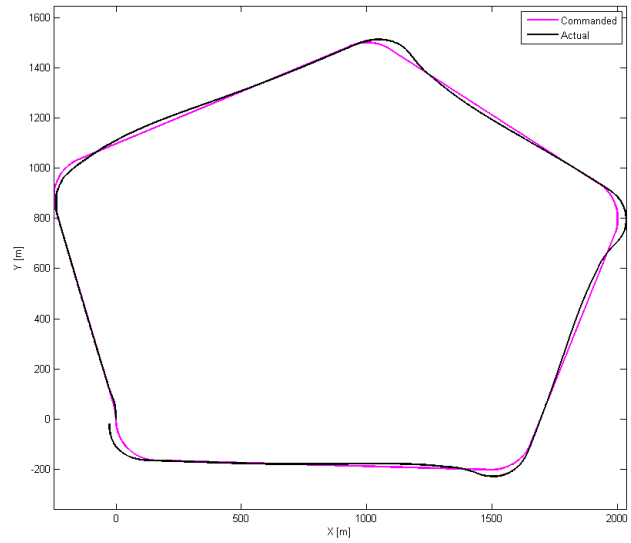

Figure 8-30—Dubins Nominal Trajectory Tracking with PPID, 2D View

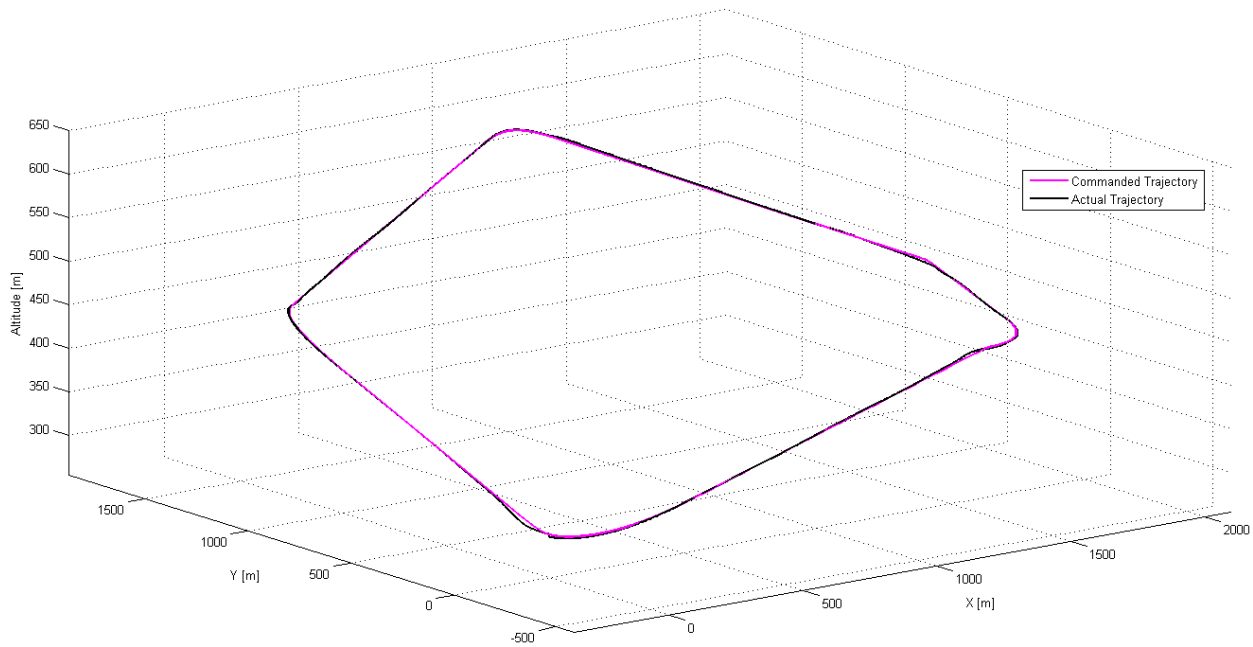

Figure 8-31—Clothoid Nominal Trajectory Tracking with ONLDI, 3D View

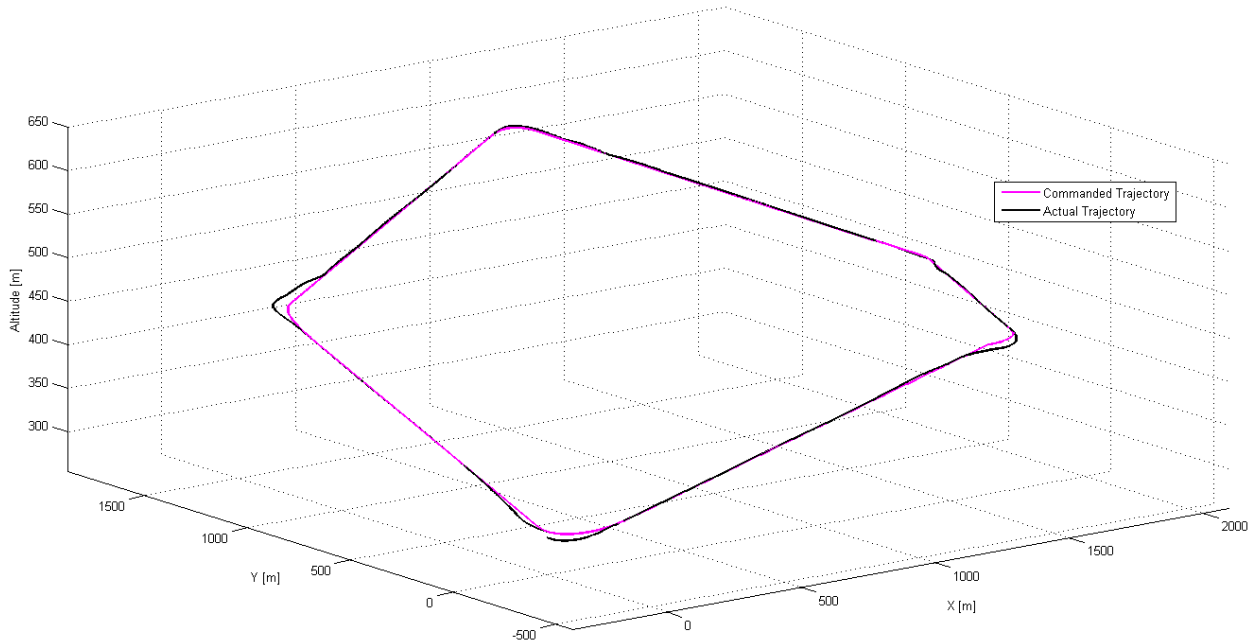

Figure 8-32—Dubins Nominal Trajectory Tracking with ONLDI, 3D View

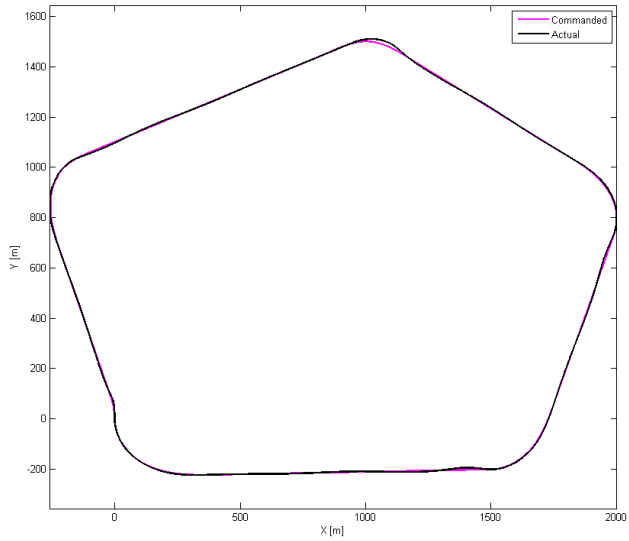

Figure 8-33—Clothoid Nominal Trajectory Tracking with ONLDI, 2D View

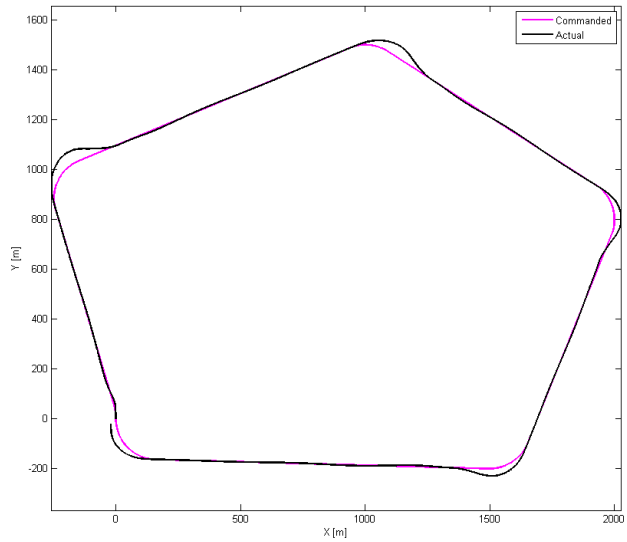

Figure 8-34—Dubins Nominal Trajectory Tracking with ONLDI, 2D View

For the particular case of the left rudder failure on the circling trajectory, the clothoid-based path was successfully tracked using the PPID controller, whereas the aircraft lost track of the path in the XY-plane when trying to track the Dubins-based path and could track only path in the vertical direction for this controller. Due to the failure, the Dubins trajectory became too demanding for the PPID controller, causing

the position error to grow rapidly until the path was lost. Although this still occurs for the clothoid, results suggest that this phenomenon does not occur as easily for the clothoid-based path in response to the PPID controller. However, no true conclusion may be drawn from this result as this loss-of-path is a weakness of the PPID controller rather than a characteristic of the path shapes. This controller decouples XY-path tracking and Z-axis path tracking and uses only the lateral error to generate the commanded bank angle. If the lateral error increases beyond a reasonable level, the logic of this controller breaks down in the XY-plane and the controller is unable to rejoin the commanded path. However, it can be seen that the aircraft continued to track the altitude profile as the logic for vertical error does not exhibit this breakdown. This phenomenon occurred for several trials in these results for the PPID controller. These results are displayed in Figure 8-35 through Figure 8-38. It can be supported that this performance difference is due to the PPID controller, as the results for the ONLDI controller did not exhibit this problem. More importantly, for the ONLDI controller, it can be seen that, while both paths were tracked, the trajectory tracking performance of the clothoid-based path was improved versus the performance of the Dubins-based path. The results of this test for the ONLDI controller can be seen in Figure 8-39 through Figure 8-42.

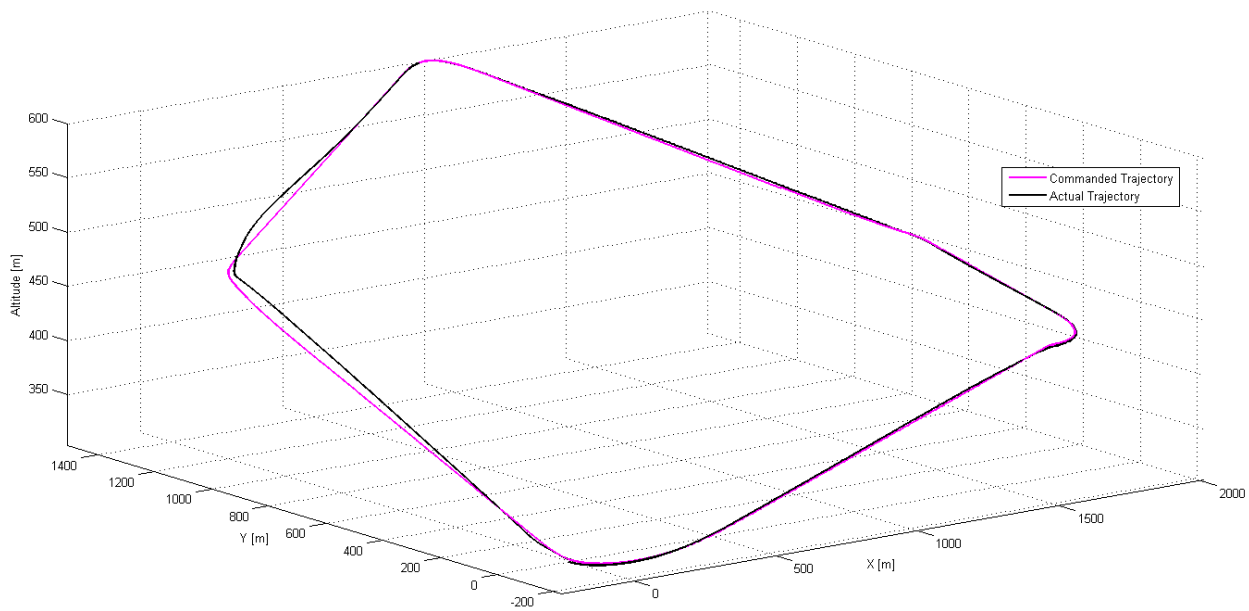

Figure 8-35—Clothoid Trajectory Tracking for Left Rudder Failure with PPID, 3D View

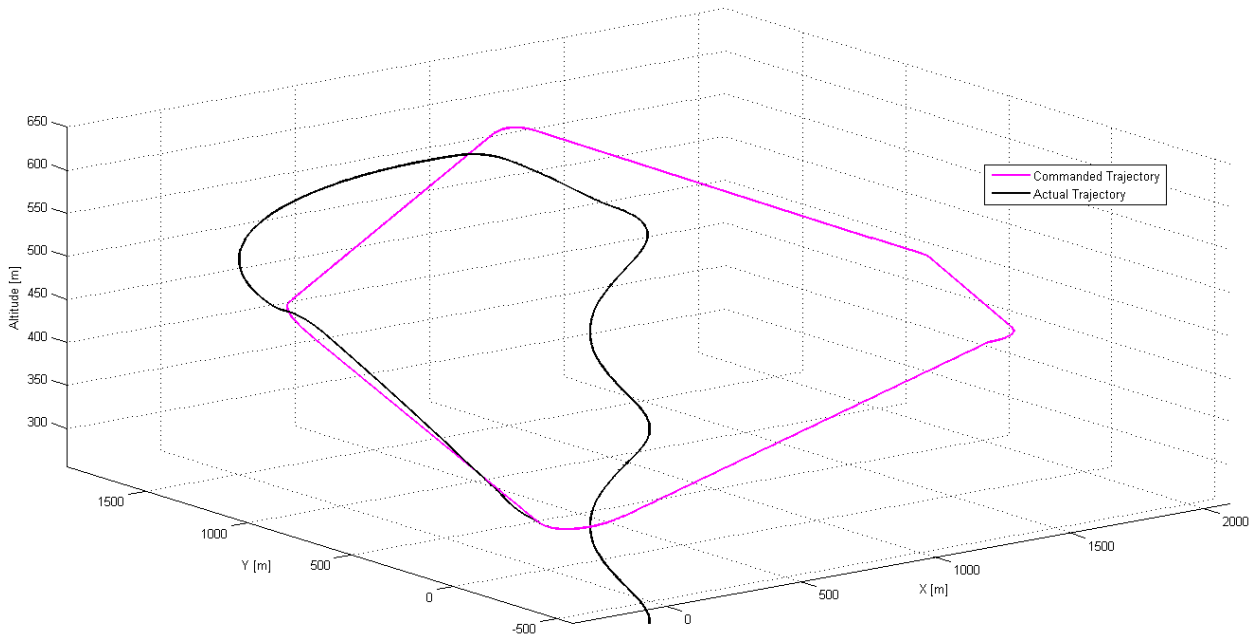

Figure 8-36—Dubins Trajectory Tracking for Left Rudder Failure with PPID, 3D View

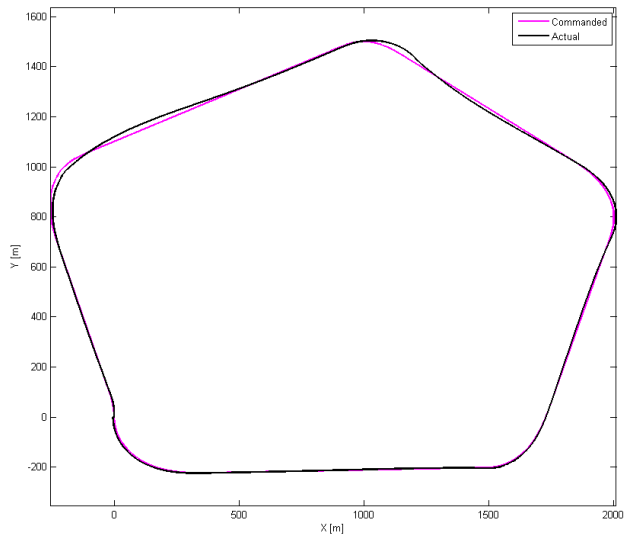

Figure 8-37—Clothoid Trajectory Tracking for Left Rudder Failure with PPID, 2D View

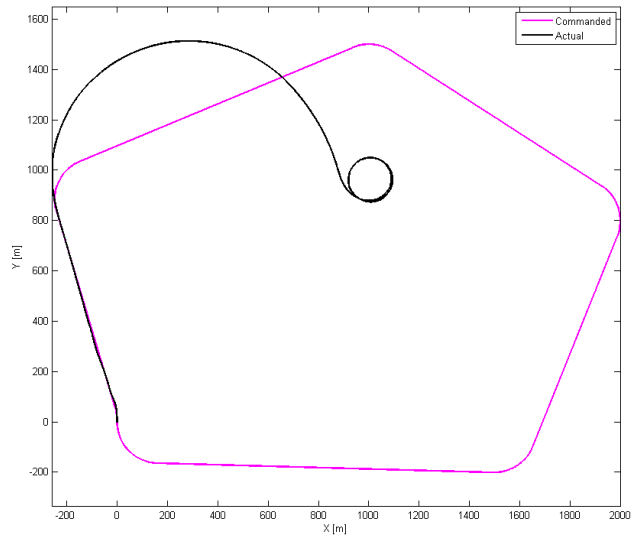

Figure 8-38—Dubins Trajectory Tracking for Left Rudder Failure with PPID, 2D View

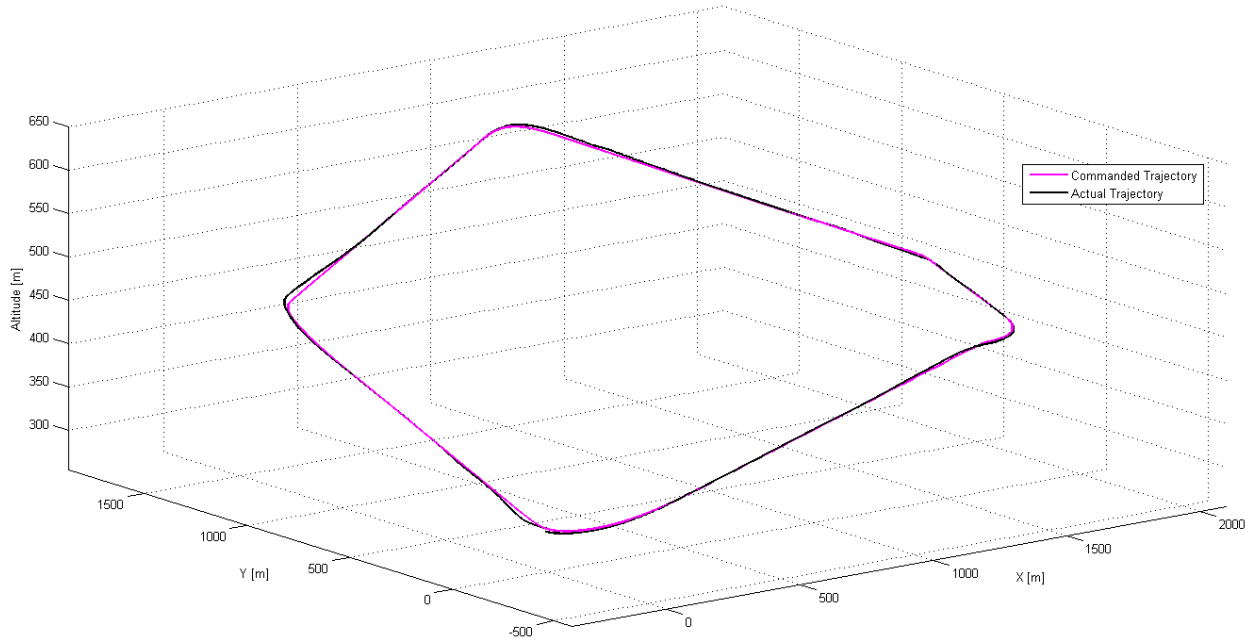

Figure 8-39—Clothoid Trajectory Tracking for Left Rudder Failure with ONLDI, 3D View

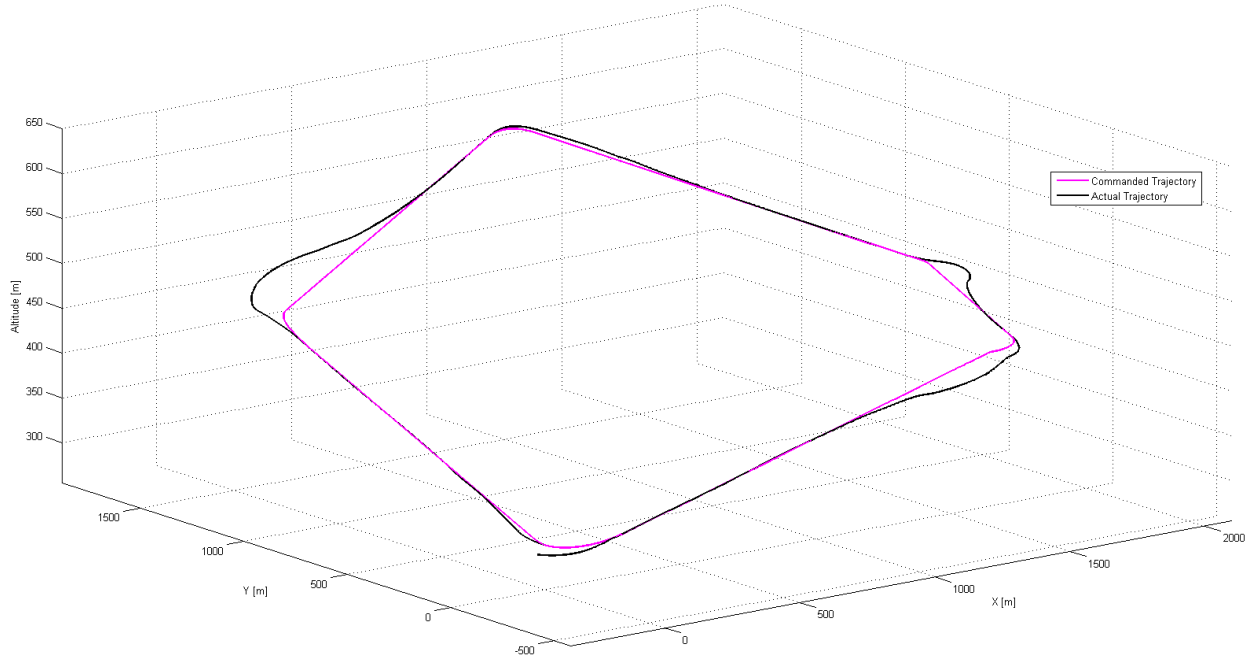

Figure 8-40—Dubins Trajectory Tracking for Left Rudder Failure with ONLDI, 3D View

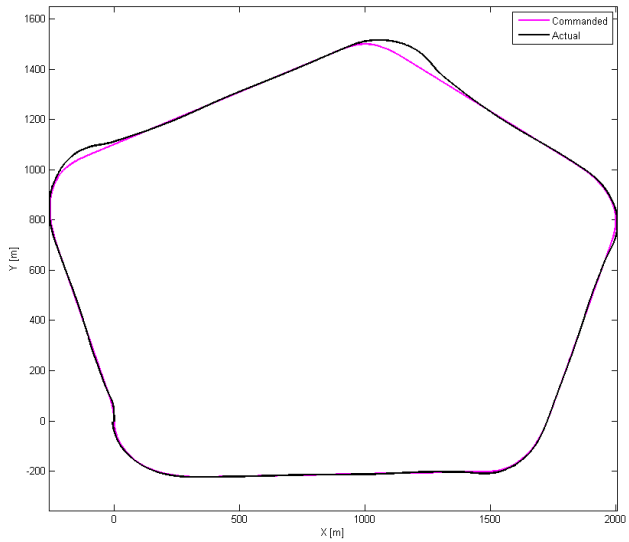

Figure 8-41—Clothoid Trajectory Tracking for Left Rudder Failure with ONLDI, 2D View

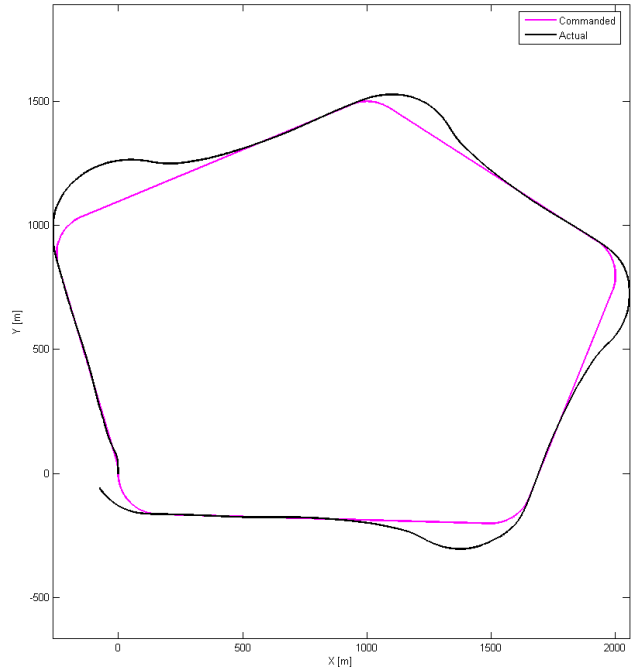

Figure 8-42—Dubins Trajectory Tracking for Left Rudder Failure with ONLDI, 2D View

For the case of the advanced path with respect to a left elevator stuck at 2 degrees, the Dubins planner achieved a better trajectory tracking performance index for the PPID controller. However, as this controller lost the path in the XY-plane and ultimately tracked only vertical error, no conclusions may be drawn from these results. The trajectory tracking results for this failure can be seen in Figure 8-43 through Figure 8-46. However, for this situation flown using the ONLDI controller, which exhibits considerably better robustness to large errors, both paths were successfully navigated, with the clothoid-based path achieving a significantly higher trajectory tracking performance that the Dubins-based path. These results can be seen in Figure 8-47 through Figure 8-50.

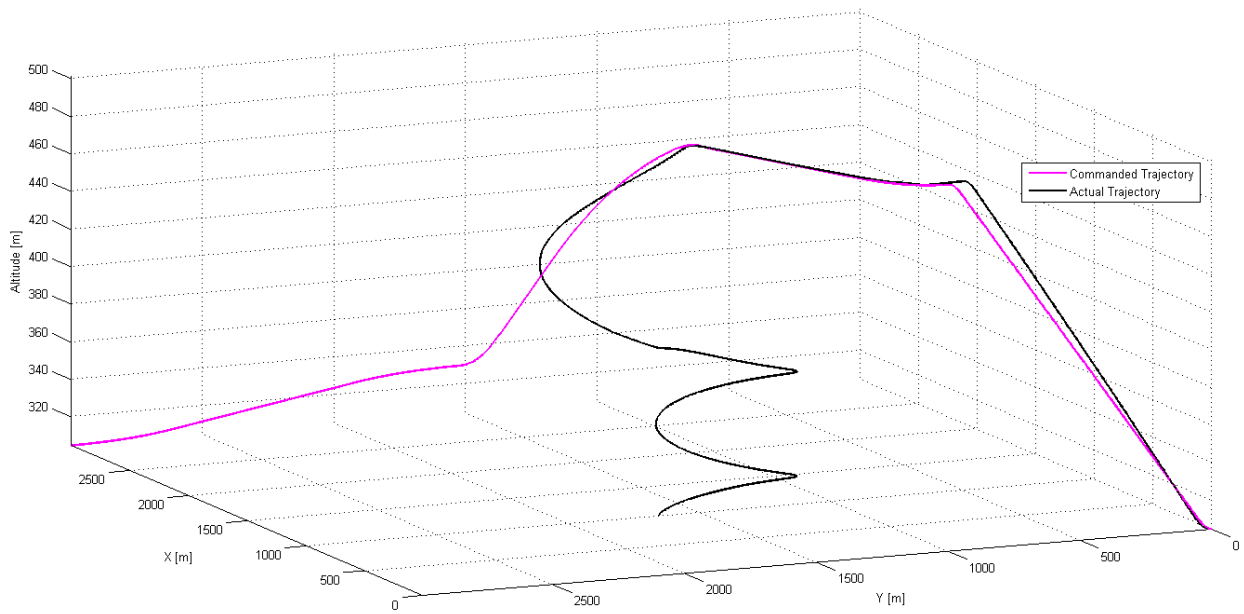

Figure 8-43—Clothoid Trajectory Tracking for Left Elevator Failure with PPID, 3D View

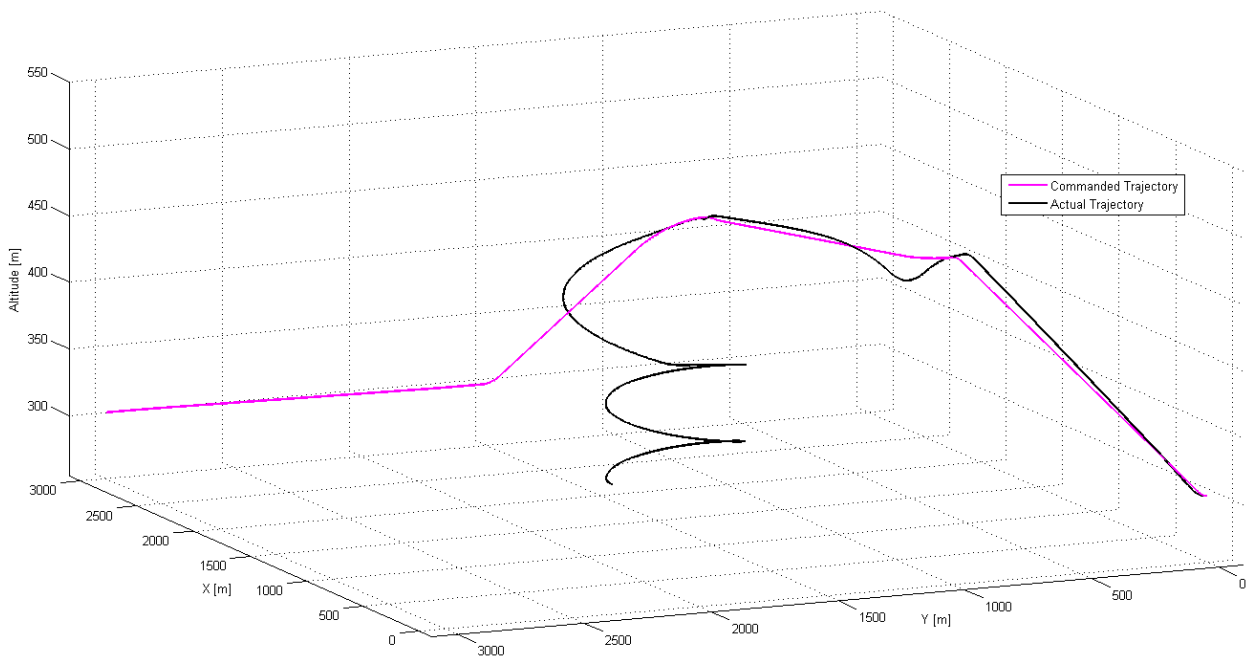

Figure 8-44—Dubins Trajectory Tracking for Left Elevator Failure with PPID, 3D View

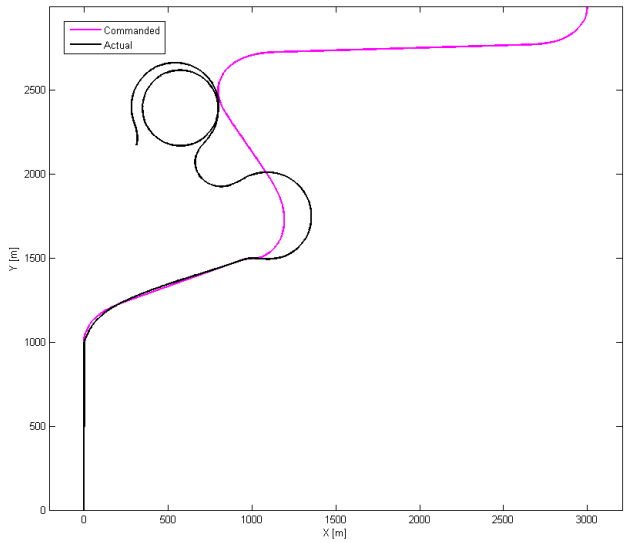

Figure 8-45—Clothoid Trajectory Tracking for Left Elevator Failure with PPID, 2D View

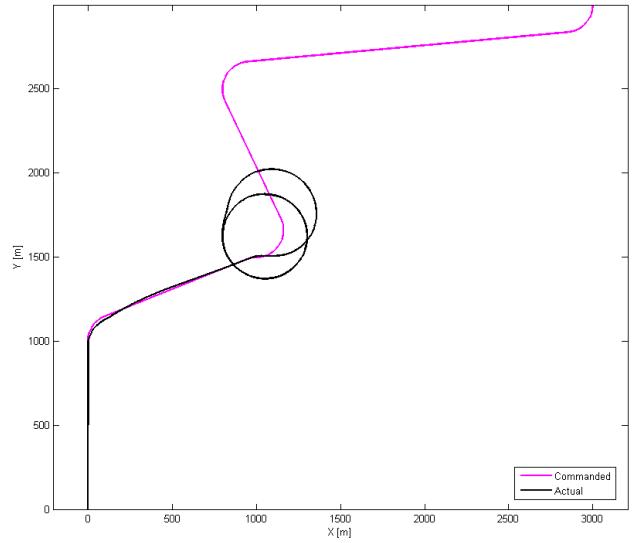

Figure 8-46—Dubins Trajectory Tracking for Left Elevator Failure with PPID, 2D View

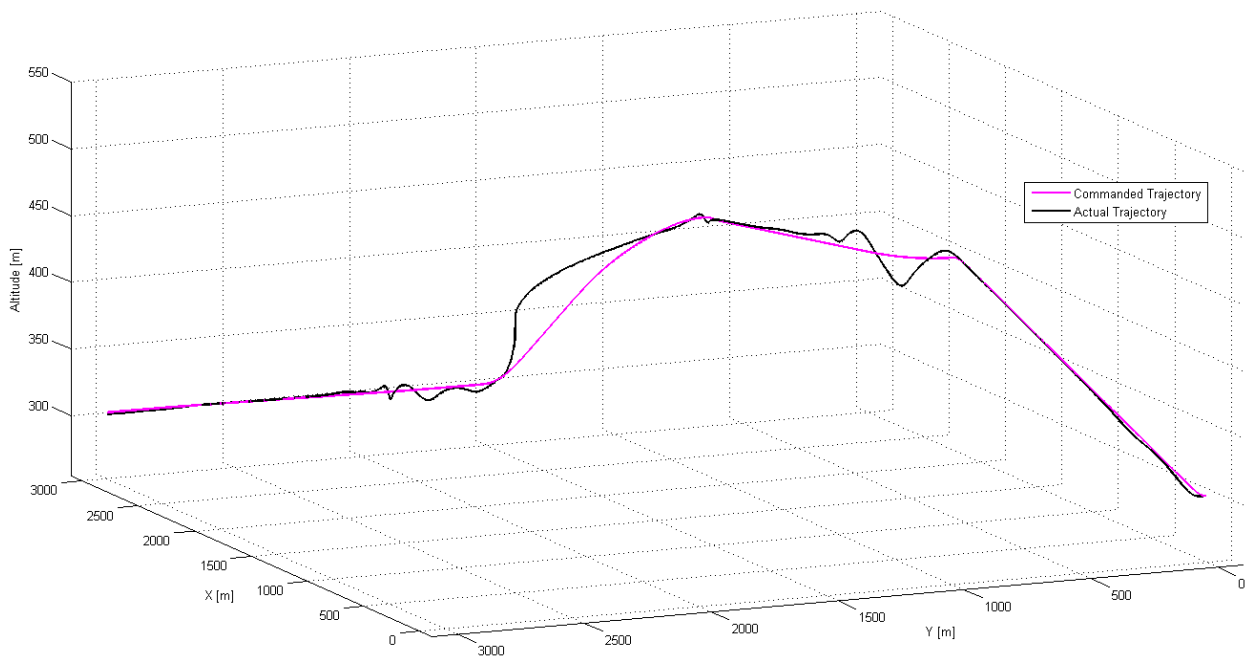

Figure 8-47—Clothoid Trajectory Tracking for Left Elevator Failure with ONLDI, 3D View

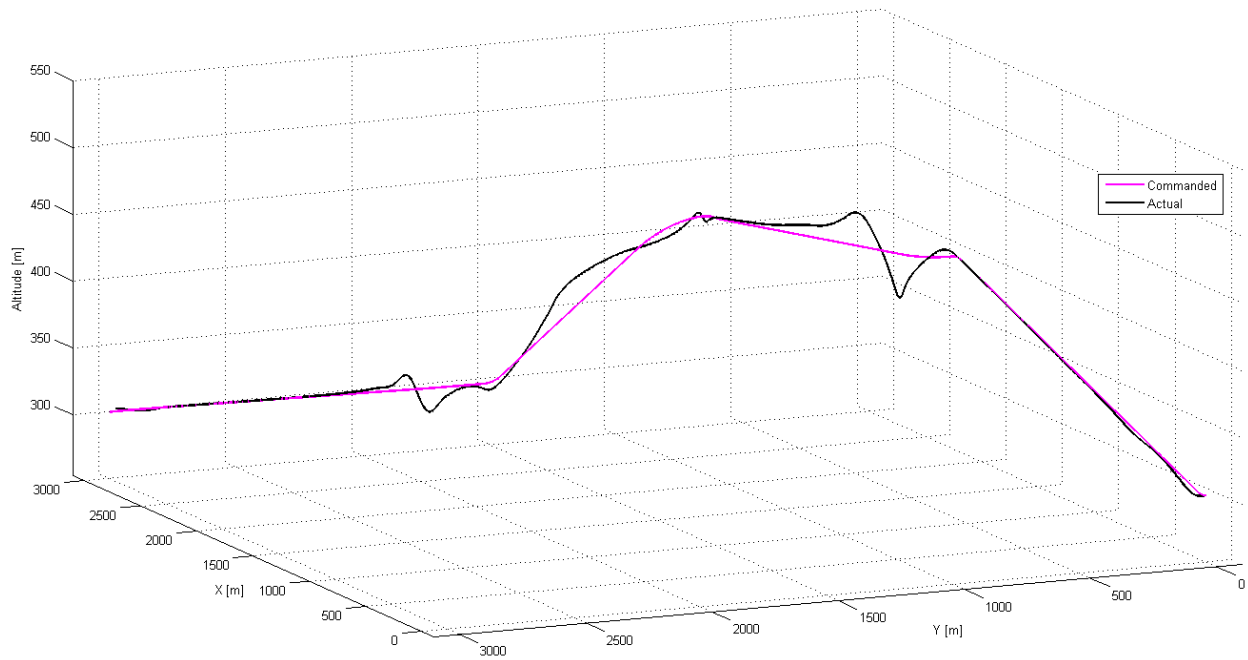

Figure 8-48—Dubins Trajectory Tracking for Left Elevator Failure with ONLDI, 3D View

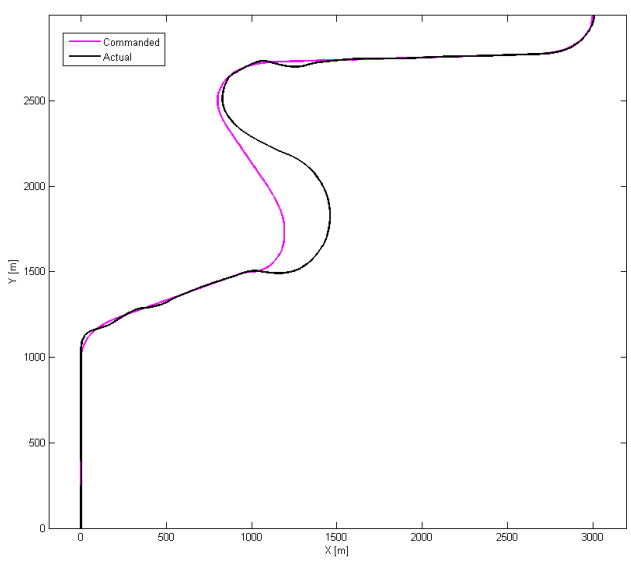

Figure 8-49—Clothoid Trajectory Tracking for Left Elevator Failure with ONLDI, 2D View

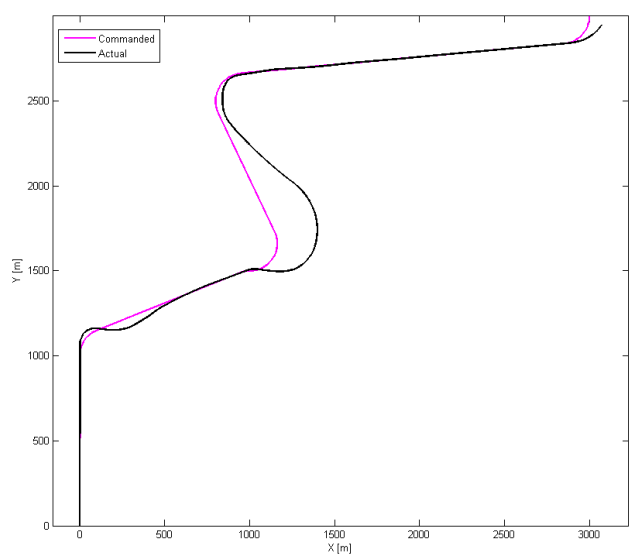

Figure 8-50—Dubins Trajectory Tracking for Left Elevator Failure with ONLDI, 2D View

The PPID controller was successfully able to follow both the clothoid and the Dubins paths for the circling path scenario, under a stuck elevator of 2 degrees. This supports that the loss-of-path seen above for this controller is a property of the controller and not of the path shape or failure type. Additionally, the

clothoid path achieved better trajectory tracking performance than the Dubins path for both the PPID and ONLDI controllers. The results of these trials may be seen below in Figure 8-51 to Figure 8-58.

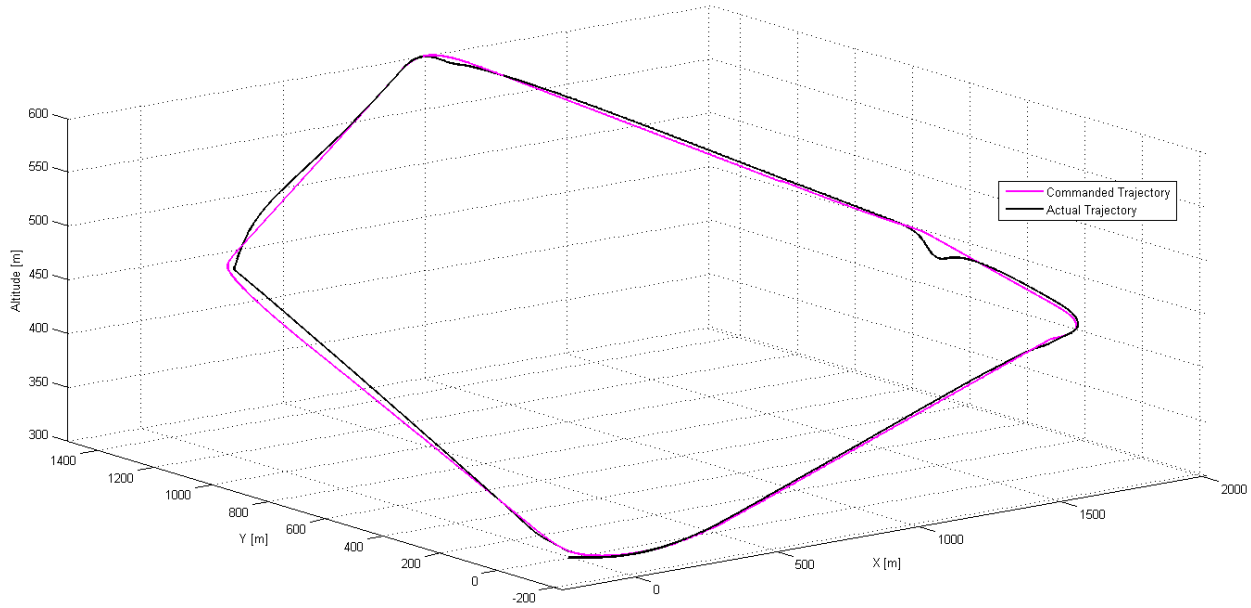

Figure 8-51—Clothoid Trajectory Tracking for Left Elevator Failure for PPID, 3D View

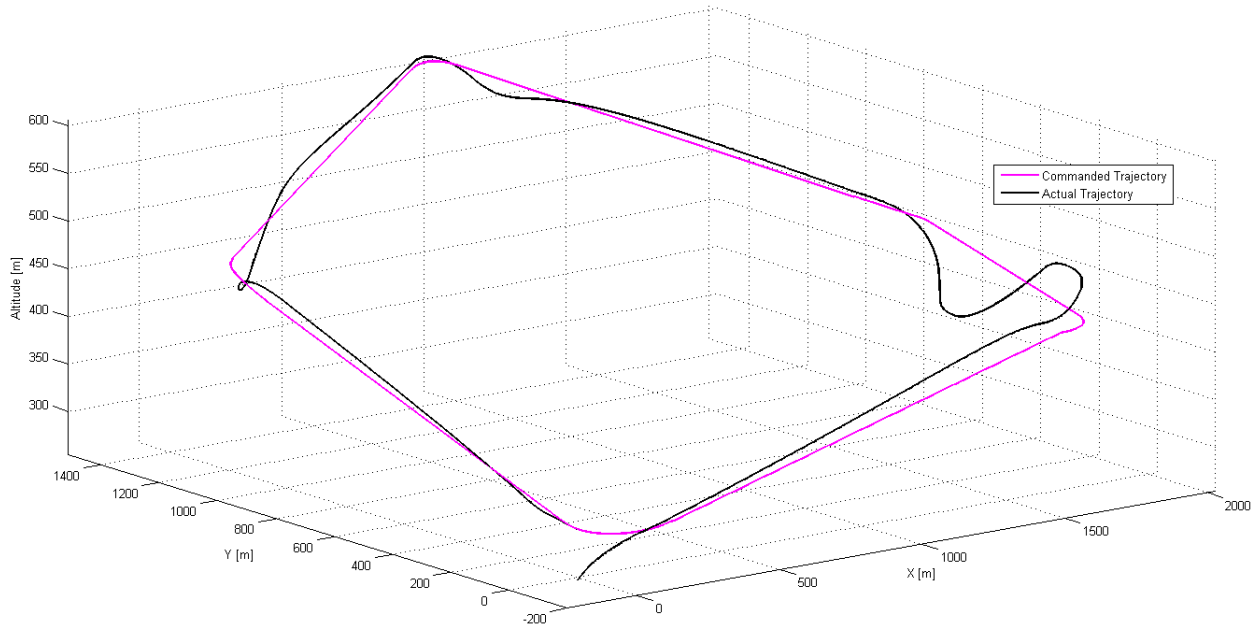

Figure 8-52—Dubins Trajectory Tracking for Left Elevator Failure for PPID, 3D View

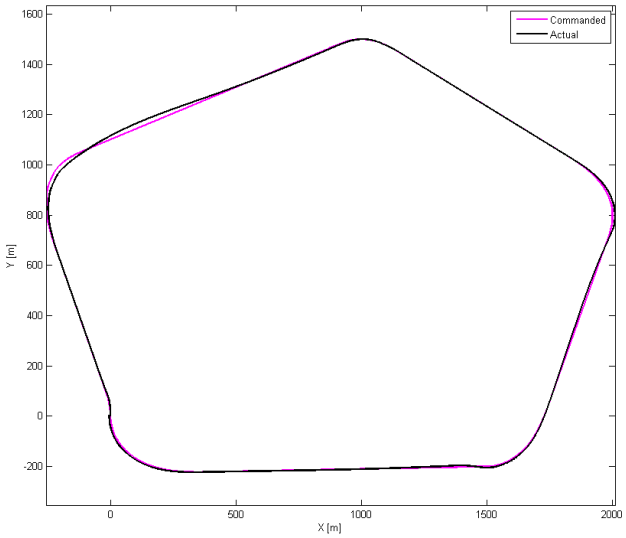

Figure 8-53—Clothoid Trajectory Tracking for Left Elevator Failure for PPID, 2D View

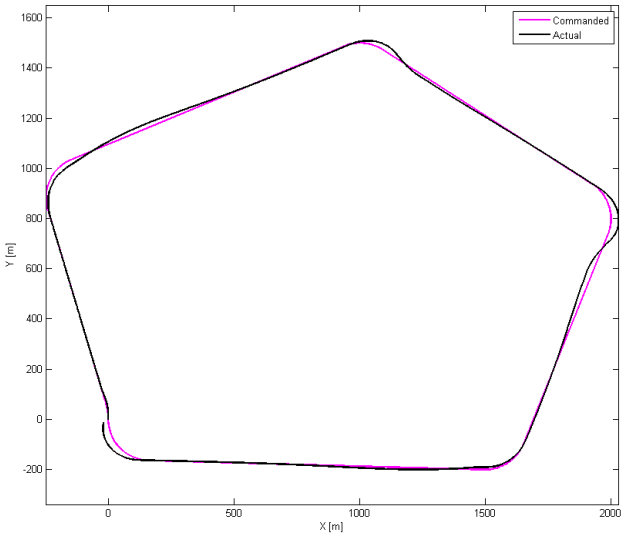

Figure 8-54—Dubins Trajectory Tracking for Left Elevator Failure for PPID, 2D View

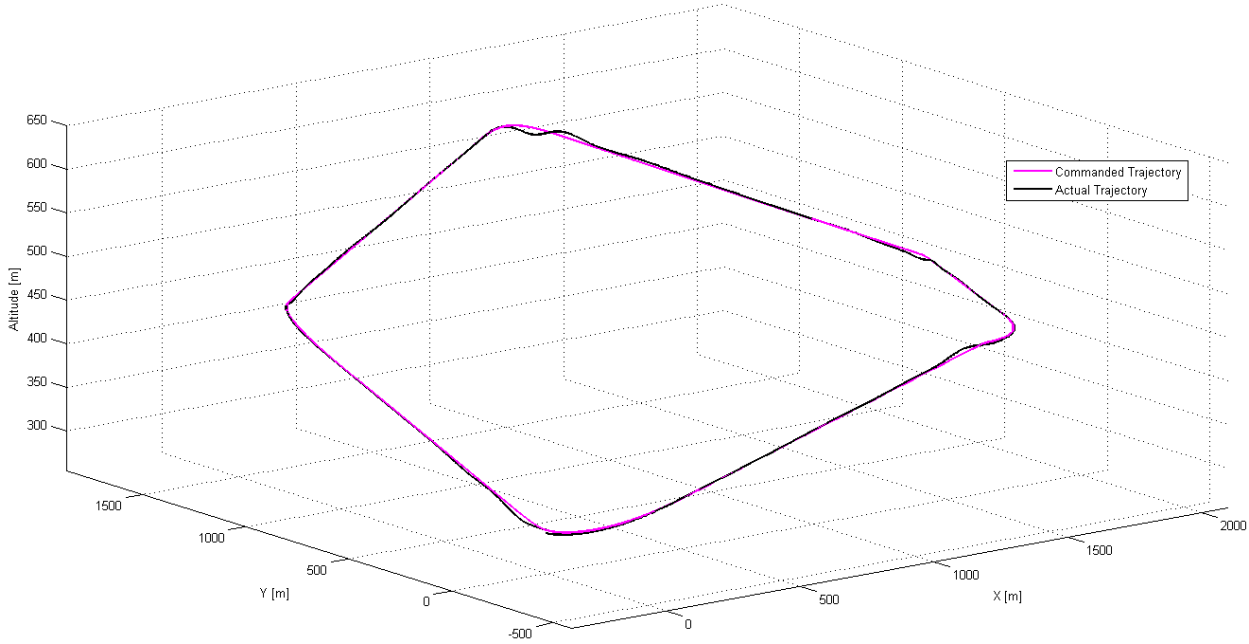

Figure 8-55—Clothoid Trajectory Tracking for Left Elevator Failure for ONLDI, 3D View

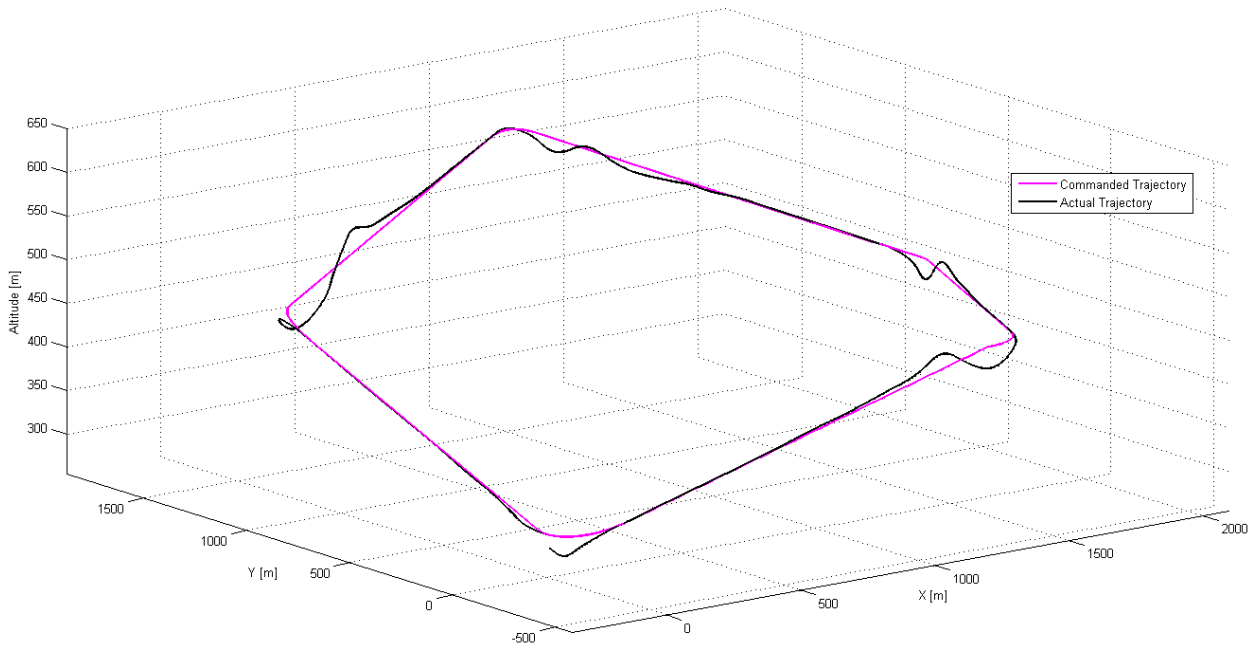

Figure 8-56—Dubins Trajectory Tracking for Left Elevator Failure for ONLDI, 3D View

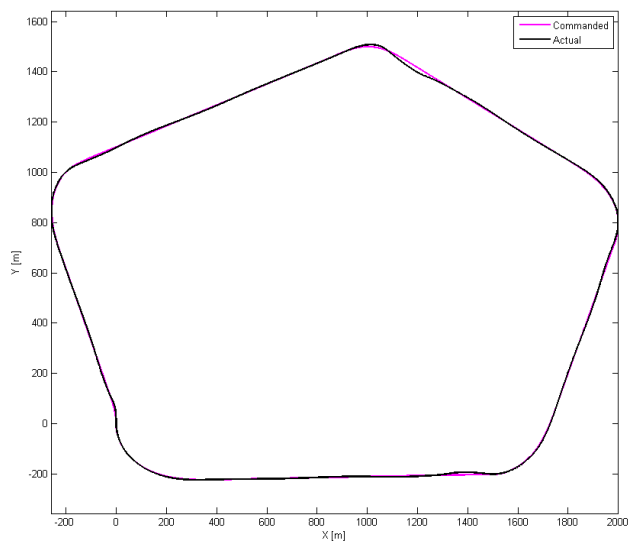

Figure 8-57—Clothoid Trajectory Tracking for Left Elevator Failure for ONLDI, 2D View

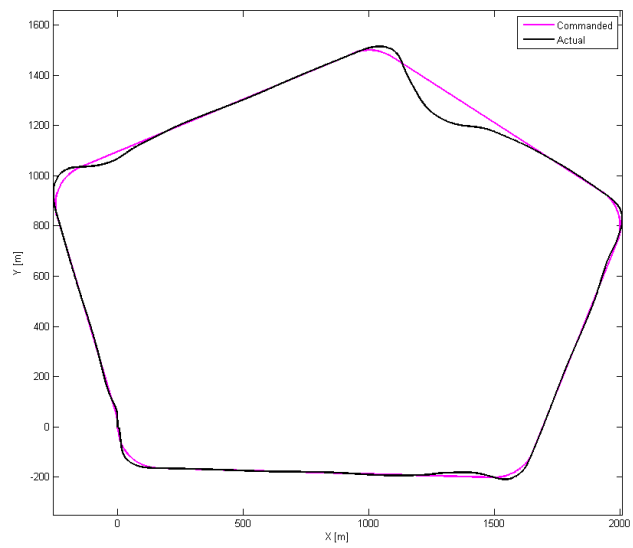

Figure 8-58—Dubins Trajectory Tracking for Left Elevator Failure for ONLDI, 2D View

E. Comparison Conclusions

It can be seen from the above results that both the 2-dimensional and 3-dimensional clothoid path planners result in equivalent or, more often, better trajectory tracking performance in terms of tracking error and control activity for both nominal and abnormal conditions. The clothoid planner yields paths which are easier for the aircraft to track. Since for the ONLDI controller no trials lost track of the path or crashed for

either the clothoid or the Dubins paths, it may be concluded from the full plots and the resulting performance indices that for the YF-22 model combined with the ONLDI controller use of the clothoid path does yield tangible trajectory tracking improvements versus use of a Dubins path. It is suggested by these results that the controller does have a significant impact on the recoverability of the aircraft to abnormal conditions, regardless of the path geometry. Thus, when possible, a controller which is more robust to upset conditions should be utilized. However, if modification of the aircraft or its controller is not feasible, commanding a clothoid-based trajectory versus a Dubins-based trajectory can provide better trajectory tracking performance. These results suggest that coupling the clothoid-based path planning methodology with a suitable failure-compensating adaptive trajectory tracking controller could significantly improve the recoverability of the aircraft to off-nominal conditions.

Chapter 9. CONCLUSIONS

From the path planning methodology investigation performed during this research, it can be concluded that there is no perfect method for every situation which a UAV may encounter. However, it is observed that the pose-based path planning methods demonstrate the greatest representation flexibility. For this reason, pose-based path planning is the primary focus of this research. A full 2-dimensional clothoid-based path planner was developed to provide continuous-curvature trajectories. This planner was compared favorably to the 2-dimensional Dubins-based path planner, in terms of overall followability under nominal and abnormal flight conditions.

In order to take full advantage of the capabilities of the UAV platform, the path planning must be performed in 3 dimensions. Equivalently, a great number of UAV platforms exhibit slow responses to input commands. Thus, path planning with a continuously-varying curvature profile is necessary to ensure flyability of the path. Inspired by the performance of the 2-dimensional clothoid trajectory's followability, the major contribution of this thesis is a clothoid-based 3-dimensional path planning methodology capable of providing this. An additional contribution, a set of simplified dynamic constraints on such a path have also been developed to ensure that the 3D path does not violate the capabilities of the aircraft, while maintaining reasonable simplicity in solving for the path. The capabilities of this path planning methodology in terms of aircraft followability have been favorably compared to an equivalent 3D Dubins-based path planner, demonstrated for the PPID and ONLDI trajectory tracking controllers on the YF-22 UAV model. The 3D clothoid-based path planner has been shown to offer paths which are significantly easier for this aircraft to follow under nominal conditions, and which may provide additional robustness under failure conditions.

Future work regarding the 3-D clothoid-based path planner involves verifying the results obtained in this thesis against a variety of trajectory tracking controllers and for a variety of aircraft. Additionally, flight testing should be performed to verify that the improvements seen in simulation are also realized in hardware-in-the-loop tests. Along with this, computational improvements should be made to improve online calculation capabilities, beginning with encoding the methodology in a lower-overhead programming language. In addition, online recalculation methods should be investigated such as computing the next path segment while flying the current one or implementing a maneuver look-up table of precomputed solutions to eliminate the need to compute paths online.

The final contribution of this thesis is an optimization algorithm based on the clonal selection mechanism of the biological immune system for the purpose of selecting the optimal pose combination to achieve the aircraft goals of obstacle avoidance and point-of-interest observation. This optimization algorithm has been briefly demonstrated to be capable of automatically selecting viable solutions from the solutions

space, and iteratively improving these solutions over time. This methodology is recommended for offline use due to computational overhead. Future work includes investigating improvements to computational efficiency to make this optimization feasible for online calculation. Contributions of this research relevant to this thesis have been published in the resources listed in Appendix D.

BIBLIOGRAPHY

1. **Anonymous.** *Unmanned Systems Roadmap 2011-2036.* Washington, D.C. : Office of the Secretary of Defense, 2011.

2. **Goodrich, M A.** *Potential Fields Tutorial.* 2000.

3. *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.* **Khatib, O.** 1, s.l. : Massachusetts Institute of Technology, Spring 1986, International Journal of Robotics Research, Vol. 5, pp. 90-98.

4. *Fast Path Planning Available for Moving Obstacle Avoidance by Use of Laplace Potential.* **Akishita, Sadao, Hisanobu, Takashi and Kawamura, Sadao.** Yokohama, Japan : s.n., 26-30 July 1993. Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robot Systems. pp. 673-678.

5. *AUV Local Path Planning Based on Virtual Potential Field.* **Fu-guang, D, et al.** s.l. : IEEE, 2005. IEEE International Conference on Mechatronics and Automation. Vol. 4, pp. 1711-1716.

6. *Autonomous Mobile Robot Navigation Using Hybrid Virtual Force Field Concept.* **Olunloyo, V O S and Ayomoh, M K O.** 2, s.l. : EuroJournals Publishing, Inc., 2009, European Journal of Scientific Research, Vol. 31, pp. 204-228.

7. *New Potential Functions for Mobile Robot Path Planning.* **Ge, S S and Cui, Y J.** 5, s.l. : IEEE, October 2000, IEEE Transactions on Robotics and Automation, Vol. 16, pp. 615-620.

8. *A Potential Field Approach to Path Planning.* **Hwang, Y K and Ahuja, N.** s.l. : IEEE, 1992. IEEE Transactions on Robotics and Automation. Vols. 8, No. 1, pp. 23-32.

9. *Three-Dimensional Formation Flying Using Bifurcating Potential Fields.* **Suzuki, Masayuki, et al.** Chicago, IL : AIAA, 10-13 August 2009. Proceedings of the 2009 AIAA Guidance, Navigation, and Control Conference.

10. **Tsourdos, Antonois, White, Brian and Shanmugavel, Madhavan.** *Cooperative Path Planning of Unmanned Aerial Vehicles.* Reston, VA : American Institute of Aeronautics and Astronautics, 2011.

11. **Berg, Mark, et al.** Visibility Graphs. *Computational Geometry.* s.l. : Springer Berlin Heidelberg, 2008, pp. 323-333.

12. *A Visibility Graph Based Method for Path Planning in Dynamic Environments.* **Kaluder, H, Brezak, M and Petrovic, I.** Croatia : IEEE, 2011. 2011 Proceedings of the 34th International Convention MIPRO. pp. 717-721.

13. *Constructing Visibility Graph and Planning Optimal Path for Inspection of 2D Workspace.* **Gao, Bo, et al.** s.l. : IEEE, 2009. IEEE International Conference on Intelligent Computing and Intelligent Systems. pp. 693-698 .

14. *Visibility Line Based Methods for UAV Path Planning.* **Omar, Rosli and Gu, Da-Wei.** Fukuoka International Congress Center, Japan : IEEE, 2009. Proceedings of the 2009 ICROS-SICE International Joint Conference.

15. *An Efficient Solution to Autonomous Path Planning by Approximate Cell Decomposition.* **Arney, T.** s.l. : IEEE, 2007. Third International Conference on Information and Automation for Sustainability. pp. 88-93.

16. *Path Planning Using Probabilistic Cell Decomposition.* **Lingelbach, F.** New Orleans, LA : IEEE, 2004. Proceedings of the 2004 IEEE International Conference on Robotics and Automation. pp. 467-472.

17. *Path Planning for Mobile Manipulation Using Probabilistic Cell Decomposition.* **Lingelbach, F.** Sendai, Japan : s.n., 2004. Proceedings of the 2004 IEEE International Conference on Intelligent Robotics and Systems. pp. 2807-2812.

18. **Weisstein, Eric.** Wolfram Math World. *Voronoi Diagram.* [Online] September 6, 2012. http://mathworld.wolfram.com/VoronoiDiagram.html.

19. *Air Vehicle Optimal Trajectories Between Two Radars.* **Novy, M C, Jacques, D R and Pachter, M.** s.l. : IEEE, 2002. Proceedings of the 2002 American Control Conference. Vol. 1, pp. 785-790.

20. *Voronoi Path Planning Technique for Recovering Communication in UAVs.* **Hammouri, O M and Matalgah, M M.** s.l. : IEEE, 2008. IEEE International Conference on Computer Systems and Applications. pp. 403-406.

21. *Path Planning for UAVs.* **Bortoff, Scott A.** Chicago, IL : AACC, 2000. Proceedings of the American Control Conference . pp. 364-368.

22. *UAV Cooperative Path Planning.* **Chandler, P R, Rasmussen, S and Pachter, M.** s.l. : AIAA, 2000. Proceedings of the 2000 AIAA Guidance, Navigation, and Control Conference.

23. *Spline-Based Path Planning for Unmanned Air Vehicles.* **Judd, Kevin B and McLain, Timothy W.** s.l. : AIAA, 2001. Proceedings of the 2001 Guidance, Navigation, and Control Conference.

24. *Trajectory Planning for Coordinated Rendezvous of Unmanned Air Vehicles.* **McLain, Timothy W and Beard, Randal W.** s.l. : AIAA, 2000. Proceedings of the 2000 AIAA Guidance, Navigation, and Control Conference.

25. *Voronoi Diagram and GIS-Based 3D Path Planning.* **Liu, L and Zhang, S.** s.l. : IEEE, 2009. 17th International Conference on Geoinformatics. pp. 1-5.

26. *Collision-Free Curvature-Bounded Smooth Path Planning Using Composite Bezier Curve Based on Voronoi Diagram.* **Ho, Y J and Liu, J S.** s.l. : IEEE, 2009. IEEE International Symposium on Computational Intelligence in Robotics and Automation. pp. 463-468.

27. *Voronoi Diagram in Optimal Path Planning.* **Bhattacharya, P and Gavrilova, M L.** s.l. : IEEE, 2007. 4th International Symposium on Voronoi Diagrams in Science and Engineering. pp. 38-47.

28. *Roadmap-Based Path Planning-Using the Voronoi Diagram for a Clearance-Based Shortest Path.* **Bhattacharya, P and Gavrilova, M L.** 2, s.l. : IEE, 2008, IEEE Robotics and Automation Magazine, Vol. 15, pp. 58-66.

29. *On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents.* **Dubins, L E.** 3, s.l. : Johns Hopkins University Press, 1957, American Journal of Mathematics, Vol. 79, pp. 497-516.

30. *Simulation of Nonholonomic Trajectory for a Car-Like Mobile Platform using Dubins Shortest Path Model.* **Said, Z and Sundaraj, K.** The University of Nottingham, Semenyih, Selangor, Malaysia : IEEE, 20-21 October 2011. 2011 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology. pp. 127-132.

31. *Path Planning of a Dubins Vehicle for Sequential Target Observation with Ranged Sensros.* **Hanson, Clarence, Richardson, Jeremy and Girard, Anouck.** San Francisco, CA : ACC, June 29-July 1. 2011. Proceedings of the 2011 American Control Conference. pp. 1698-1703.

32. *Simplified Model Development and Trajectory Determination for a UAV using the Dubins Set.* **Grymin, David J and Crassidis, Agamemnon L.** Chicago, IL : AIAA, 2009. Proceeding of the 2009 AIAA Atmospheric Flight Mechanics Conference.

33. *Formal Techniques for the Modelling and Validation of a Co-operating UAV Team that uses Dubins Set for Path Planning.* **Jeyaraman, Suresh, et al.** Portland, OR : s.n., 2005. Proceedings of the 2005 American Control Conference. pp. 4690-4695.

34. *Formalized Hybrid Control Scheme for a UAV Group using Dubins Set and Model Checking.* **Jeyaramen, S, et al.** Atlantis, Paradise Island, Bahamas : IEEE, December 14-17, 2004. Proceedings of the 43rd IEEE Conference on Decision and Control. Vol. 4, pp. 4299-4304.

35. *Path Planning of Multiple UAVs using Dubins Sets.* **Shanmugavel, M, et al.** San Francisco, CA : AIAA, 15-18 August 2005. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

36. *Planning Continuous-Curvature Paths for Car-Like Robots.* **Scheuer, A and Fraichard, T.** s.l. : IEEE, 1996. Proceedings of the 1996 Internation Conference on Intelligent Robots and Systems.

37. *Continuous-Curvature Path Planning for Car-Like Vehicles.* **Scheuer, A and Fraichard, T.** s.l. : IEEE, 1997. Proceedings of the 1997 International Conference on Intelligent Robots and Systems. pp. 997-1003.

38. *Trajectory Generation based on Rational Bezier Curves as Clothoids.* **Montes, Nicolas, Mora, Marta C and Tornero, Josep.** Istanbul, Turkey : IEEE, 13-15 June 2007. Proceedings of the 2007 IEEE Intelligent Vehicles Symposium. pp. 505-510.

39. *Co-operative Path Planning of Multiple UAVs Using Dubins Paths with Clothoid Arcs.* **Shanmugavel, Madhavan, et al.** s.l. : Elsevier, ScienceDirect, 2010, Control Engineering Practice, Vol. 18.

40. *Pythagorean Hodographs.* **Farouki, R T and Sakkalis, T.** 5, s.l. : IBM, 1990, IBM Journal of Research and Development, Vol. 34, pp. 736-752.

41. *Path Planning for Mobile and Hyper-Redundant Robots Using Pythagorean Hodograph Curves.* **Bruyninckx, Herman and Reynaerts, Dominiek.** Monterey, CA : IEEE, 7-9 July 1997. Proceedings of the 1997 International Conference on Advanced Robotics. pp. 595-600.

42. *UAV Guidance Laws to Arrival at Sesired Position and Time from Desired Direction.* **Lim, S and Bang, H.** Korea : IEEE, 2011. 11th International Conference on Control, Automation and Systems. pp. 299-304.

43. *Pythagorean Hodograph (PH) Path Planning for Tracking Airborne Contaminant Using Sensor Swarm.* **Subchan, S, et al.** Victoria, Vancouver Island, Canada : IEEE, 2008. IEEE International Instrumentation and Measurement Technology Conference.

44. *Optimal and Efficient Path Planning for Partially-Known Environments.* **Stentz, Anthony.** May 1994. Proceedings of the IEEE International Conference on Robotics and Automation.

45. *The Focussed D* Algorthim for Real-Time Replanning.* **Stentz, Anthony.** 1995. Proceedings of the International Joint Conference on Artifiical Intelligence.

46. **Eppstein, David.** *Finding the k-Shortest Paths.* Department of Information and Computer Science, University of California. Irvine, CA : University of California, 1994. pp. 154-165, Technical Report.

47. —. *Finding the k-Shortest Paths.* Department of Information adn Computer Science, University of California. Irvine, CA : University of California, 1997.

48. *A*Prune: An Algorithm for Finding K Shortest Paths Subject to Multiple Constraints.* **Liu, Gang and Ramakrishnan, K G.** s.l. : IEEE, 2001. IEEE International Conference on Computer Communications.

49. *An Obstacle-Based Rapidly-Exploring Random Tree.* **Rodriguez, Samuel, et al.** Orlando, FL : IEEE, 2006. Proceedings of the 2006 IEEE International Conference on Robotics and Automation.

50. *Multi-UAV Path Planning in Obstacle Rich Environments Usin Rapidly-Exploring Random Trees.* **Kothari, Mangal, Pstlethwaite, Ian and Gu, Da-Wei.** Shanghai, P.R. China : s.n., 2009. Proceedings of the 48th IEEE Joint Conference on Decision and Control.

51. *Static and Dynamic Obstacle Avoidance in Miniature Air Vehicles.* **Saunders, Jeffery B, et al.** s.l. : AIAA, 2005. Infotech@Aerospace.

52. *Optimal vs. Heuristic Assignment of Cooperative Autonomous Unmanned Air Vehicles.* **Rasmussen, S, et al.** s.l. : AIAA, 2003. Proceedings of the 2003 AIAA Guidance, Navigation, and Control Conference.

53. *Randomized Kinodynamic Planning.* **LaValle, Steven M and Kuffner, James J.** s.l. : IEEE, 1999. Proceedings of the IEEE International Conference on Robotics and Automation.

54. *Robust Constrained Receding Horizon Control for Trajectory Planning.* **Kuwata, Y, et al.** s.l. : AIAA, 2005. Proceedings of the 2004 AIAA Guidance, Navigation, and Control Conference.

55. *Decentralized Cooperative Trajectory Planning of Multiple Aircraft with Hard Safety Guarantees.* **Schowenaars, T, How, J and Feron, E.** s.l. : AIAA, 2004. Proceedings of the 2004 AIAA Guidance, Navigation, and Control Conference.

56. **Goldberg, David E.** *Genetic Algorithms in Search, Optimization, and Machine Learning.* 1. s.l. : Addison-Wesley Professional, 1989.

57. *Intelligent Path Planning with Evolutionary Computation.* **Parry, Adam C and Ordonez, Raul.** Toronto, Ontario, Canada : AIAA, 2-5 August 2010. Proceedings of the 2010 AIAA Guidance, Navigation, and Control Conference.

58. *An Evolution BAsed Path Planning Algorithm for Autonomous Motion of A UAV Through Uncertain Environments.* **Rathbun, David, et al.** Irvine, CA : s.n., 2002. Proceedings of the 21st Digital Avionics Systems. Vol. 2, pp. 8D2-1-8D2-12.

59. *Adaptive Path Planning for Autonomous UAV Oceanic Search Missions.* **Rubio, Juan Carlos, Vagners, Juris and Rysdyk, Rolf.** Chicago, IL : AIAA, 20-22 September 2004. Proceedings of the AIAA 1st Intelligent Systems Conference.

60. *Evolutionary Route Planner for Unmanned Aerial Vehicles.* **Zheng, Changwen, et al.** 4, August 2005, IEEE Transactions on Robotics, Vol. 21, pp. 609-620.

61. *Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation.* **Nikolos, Ioannis K, et al.** s.l. : IEEE, December 2003. IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics. Vol. 33, pp. 898-912.

62. *Real Path Planning Based on Genetic Algorithm adn Voronoi Diagrams.* **Benavides, Facundo, et al.** s.l. : IEEE, 1-4 October 2011. Proceedings of the 2011 IEEE IX Latin American Robotics Symposium and IEEE Columbian Conference on Automatic Control and Industry Applications. pp. 1-6.

63. **Karas, Ondrej.** *UAV Simulation Environment for Autonomous Flight Control Algorithms.* Morgantown, WV : West Virginia University, 2012.

64. **Napolitano, M R.** *Development of Formation Flight Control Algorithms Using 3 YF-22 Flying Models.* Morgantown, WV : s.n., April 2005.

65. *Design of Formation Flight Control Laws for Maneuvered Flight.* **Campa, G, et al.** March 2004, The Aeronautical Journal, pp. 125-134.

66. **Gu, Yu.** *Design and Flight Testing Actuator Failure Accommodation Controllers on WVU YF-22 Reserach UAVs.* Morgantown, WV : West Virginia University, 2004.

67. *Development of Fault-Tolerant Flight Control Laws for the WVU YF-22 Model Aircraft.* **Perhinschi, M G, et al.** Hilton Head, SC : AIAA, 2007. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

68. *Formation Flight Test Results for UAV Research Aircraft Models.* **Seanor, B, et al.** Chicago, IL : AIAA, 2004. Proceedings of the AIAA Intelligent Systems Technical Conference.

69. **Jordan, Thomas, et al.** *Development of a Dynamically Scaled Generic Transport Model Testbed for Flight Research Experiments.* Hampton, VA : NASA Langley Research Center.

70. Unmanned Aircraft Systems. *Navmar Applied Sciences Corporation.* [Online] http://www.nasc.com/ps-uas.php.

71. OX Unmanned Aerial Vehicle (UAV). *CLMax Engineering.* [Online] 2011.

72. United States Navy Fact File: RQ-2A PIONEER UNMANNED AERIAL VEHICLE (UAV). *America's Navy.* [Online] February 18, 2009. http://www.navy.mil/navydata/fact_display.asp?cid=1100&tid=2100&ct=1.

73. **Amato, Nancy.** *Potential Field Methods (Chapter 7 and Papers).* 1996.

74. **Kinahan, Kelly and Pryor, Jennifer.** Example Networks1: Dijkstra's Algorithm for Shortest Route Problems. *Algorithm Animations for Practical Optimization: A Gentle Introduction.* [Online] http://optlab-server.sce.carleton.ca/POAnimations2007/DijkstrasAlgo.html.

75. Lecture 18-Solving Shortest Path Problem: Dijkstra's Algorithm. [Online] October 23, 2009. https://netfiles.uiuc.edu/angelia/.

76. **Wang, Xiaodong.** Dijkstra Shortest Path Routing. *Matlab Central.* [Online] July 23, 2004. http://www.mathworks.com/matlabcentral/fileexchange/5550-dijkstra-shortest-path-routing.

77. *Development of a Modified Voronoi Algorithm for UAV Path Planning and Obstacles Avoidance.* **Wilburn, J, et al.** Minneapolis, MN : AIAA, 2012. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

78. *Enhanced Modified Voronoi Algorihtm for UAV Path Planning and Obstacle Avoidance.* **Wilburn, Jennifer N, Perhinschi, Mario G and Wilburn, Brenton K.** February, s.l. : International Review of Aerospace Engineering, 2013, Vol. 2013.

79. **Tsourdos, A, White, B and Shanmugavel, M.** Path Planning in Two Dimensions. [ed.] Frank K Lu. *Cooperative Path Planning of Unmanned Aerial Vehicles.* Chichester West Sussex : AIAA, Wiley, 2011, Vol. 235, 2, pp. 29-63.

80. *Implementation of Composite Clothoid Paths for Continuous Curvature Trajectory Generation for UAVs.* **Wilburn, Jennifer N, Perhinschi, Mario G and Wilburn, Brenton K.** Boston, MA : AIAA, 2013. Proceedings of the 2013 AIAA Guidance, Navigation, and Control Conference.

81. *UAV Clothoid Path Generation Algorithm with Shortest Path Criterion.* **Wilburn, Jennifer N, Perhinschi, Mario G and Wilburn, Brenton K.** s.l. : AIAA, submitted to AIAA Journal of Guidance, Control, and Dynamics, March 2013.

82. *Implementation of a 3-Dimensional Dubins-Based UAV Path Generation Algorithm.* **Wilburn, Jennifer N, Perhinschi, Mario G and Wilburn, Brenton K.** Boston, MA : AIAA, 2013. Proceedings of the 2013 AIAA Guidance, Navigation, and Control Conference.

83. **Eberly, David.** *Bisection in 1D, 2D, and 3D.* s.l. : Geometric Tools, LLC, 1999-2008.

84. *The Natural-Selection Theory of the Immune System.* **Jerne, Niels K.** 1955. Proceedings of the National Academy of Sciences of the United States of America. pp. 849-857.

85. **Jerne, Niels K.** The Generative Grammar of the Immune System. [Online] 1984. Nobelprize.org.

86. Clonal Selection. [Online] January 10, 2011. http://users.rcn.com.

87. **Kaiser, Gary E.** Doc Kaiser's Microbiology Homepage. *The Community College of Baltimore County.* [Online] June 2012.

88. *Artificial Immune Optimization Methods and Applications--A Survey.* **Wang, X, Gao, X Z and Ovaska, S J.** s.l. : IEEE, 2004. IEEE International Conference on Systems, Man and Cybernetics. pp. 3415-3420.

89. *Two Models of Immunization for Time Dependent Optimization.* **Gaspar, Alessio and Collard, Philippe.** Nashville, TN : IEEE, October 2000. IEEE International Conference on Systems, Man, and Cybernetics. pp. 113-118.

90. *Solving Multiobjective Optimization Problems Using an Artificial Immune System.* **Coello, C A C and Cortes, N C.** Canterbury, UK : s.n., September 2002. Proceedings of the First International Conference on Artificial Immune Systems. pp. 212-221.

91. *An Artificial Immune Network for Multimodal Function Optimization.* **de Castro, L N and Timmis, j.** Honolulu, HI : s.n., 2002. Proceedings of the IEEE Congress on Evolutionary Computation. pp. 699-704.

92. *Development of a Simulation Environment for Autonomous Flight Control Algorithms.* **Perhinschi, M G, et al.** Portland, OR : AIAA, 2011. Proceedings of the AIAA Modeling and Simulation Technologies Conference.

93. **Olson, Curtis.** Introduciton to Flight Gear. [Online] FlightGear, n.d. [Cited:] flightgear.org.

94. *Unmanned Aerial Vehicle Trajectory Tracking Algorithm Comparison.* **Wilburn, Brenton K, et al.** International Journal of Unmanned Intelligent Systems.

95. **Tsourdos, A, White, B and Shanmugavel, M.** Introduction. [ed.] Frank K Lu. *Cooperative Path Planning of Unmanned Aerial Vehicles.* Chichester West Sussex : AIAA, Wiley, 2011, 1, pp. 1-28.

96. **Waller, M C, et al.** *Considerations in the Application of Dynamic Programming to Optimal Aircraft Trajectory Generation.* Melboume, Australia : IEEE, 1990.

97. **Yen, J and Langari, R.** *Fuzzy Logic: Intelligence, Control, and Information.* s.l. : Prentice Hall, 1999.

98. **Kane, Thomas R and Levinson, David A.** *Dynamics: Theory and Applications.* s.l. : McGraw Hill, Inc. , 1985.

99. *The Elastic Bending Energy of Pythagorean-Hodograph Curves.* **Farouki, Rida T.** s.l. : Elsevier, February 1995, Computer Aided Geometric Design, Vol. 13, pp. 227-241.

100. *Hermite Interpolation By Pythagorean Hodograph Quintics.* **Farouki, R T and Neff, C A.** 212, October 1995, Mathematics of Computation, Vol. 64, pp. 1589-1609.

101. *On Curves of Minimum Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents.* **Dubins, L E.** 3, July 1957, American Journal of Mathematics, Vol. 79, pp. 497-516.

102. **Celik, Ismail B.** *Introductory Numerical Methods for Engineering Applications.* 2001.

103. *UAV Adaptive Control Laws Using Dynamic Inversion Augmented with an Immunity-Based Mechanism.* **Moncayo, H, et al.** Minneapolis, MN : AIAA, 2012. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

104. *Extended Nonlinear Control Laws for Unmanned Air Vehicles.* **Moncayo, H, et al.** Minneapolis, MN : s.n., 2012. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

105. *Comparison of a Fuzzy Logic Controller to a Potential Field Controller for Real-Time UAV Navigation.* **Wilburn, J, et al.** Minneapolis, MN : AIAA, 2012. Proceedings of the AIAA Guidance, Navigation, and Control Conference.

APPENDICES

A PSEUDO-CODE

Pseudo-Code 1: Standard Dijkstra's Algorithm

```
INPUTS:
nodes       nx2 list of node locations
pathCost    nxn matrix containing path cost at indices corresponding to
            connected nodes and "inf" at unconnected node
startIndex  node index of starting node
goalIndex   node index of goal node

OUTPUTS:
indexPath   list of node indices from start to goal for lowest cost path
_____________________________________________________________________________
FOR all nodes {
      set visited to false
      set bestCost to inf
      set previousNode to null
}
set bestCost(startIndex) to 0
FOR nodes 1 through n-1 {
      find all unvisted nodes
      choose the one with the least cost whose index is q
      set node q as visited
      find nodes connected to node q
      for each connected node whose index is r
      if bestCost(r) is greater than bestCost(q) + pathCost(q to r) {
            set bestCost(r)=bestCost(q) + pathCost(q to r)
            set previous(r)=q
      }
}
currentIndex=goalIndex
WHILE currentNode != startIndex{
      add currentNode to indexPath array
      set currentNode to previous(currentNode)
}
add startIndex to indexPath array
```

Pseudo-Code 2: Cell Decomposition

```
INPUTS:
Start             starting location and heading
Goal              final location and heading
Zones             list of all risk zones in the solution space
Gridsize          minimum allowable path segment size
Velocity          aircraft cruise velocity
SampleRate        aircraft controller sample rate

OUTPUTS:
Trajectory        [X Y Z t] commanded aircraft trajectory
TotalLength       total path length
_____________________________________________________________________
Generate cell decomposition paths grid: {
      Determine location of obstacles.
      Generate initial limits of the solution space based on zone locations.
      Initialize cells array with one cell around the limits. Set active.
      WHILE active cells exist{
            FOR each cell{
                  IF obstacles inside cell and new cells will be larger than
                  the gridsize, subdivide. Keep each new cell active.
                  ELSE set cell inactive.
            }
      }
      Assemble a list of nodes (vertices) and path segments (edges) which do
      not cross obstacles.
}

Calculate the cost of each path segment.
Choose the lowest cost path with Dijkstra's algorithm.
Smooth the path.
Generate trajectory points.
```

Pseudo-Code 3: Traditional Bisection

```
WHILE accuracy is insufficient {
      c = (a+b)/2
      IF f(a) * f(c) < 0,
            THEN b = c;
      ELSE a = c
}
```

Pseudo-Code 4: Modified Bisection Algorithm for 2 Equations and 1 Unknown

```
Plot f(φ_s) = 0 and g(φ_s) = 0, φ_s ∈ [0,2π], with resolution 2π/400.

Search each interval for sign changes in either f(φ_s) or g(φ_s).

Record bounds for which both functions have a sign change. Set these as
initial bounds.

Subdivide bounds at the midpoint using c = (a+b)/2. Compute the solution to the
equations for this point.

WHILE accuracy is insufficient {

IF no sign changes are found for either x or y, THEN subdivide all existing
bounds and repeat.

IF a set of bounds is found for x but not y, THEN: set the x bounds as the
new bounds. Subdivide the new bounds and repeat.

IF a set of bounds is found for y but not x, THEN: set the y bounds as the
new bounds. Subdivide the new bounds and repeat.

IF a set(s) of bounds matches, THEN: select this as part of new bounds.
Subdivide the new bounds and repeat.

IF none of the bounds found for x and y match, retain all bounds. Subdivide
and repeat.

}

Subdivide the remaining bound and take this to be the solution.
```

Pseudo-Code 5: Clothoid Path Planner

```
FOR each set of poses P {

Initialize the relevant variables (for convenience).

Compute the Transformation Matrices.

Compute ϕ_total.

Compute the quadrants and determine which quadrant the solution falls into.

FOR each path direction combination {

IF a valid path is possible, THEN compute ϕ_s and ϕ_f using the Modified
Bisection Algorithm.

Compute the finalized path length.

}

Choose the shortest valid path as the final solution.
Record the solution in the Total Path.

}

Compute the Trajectory Points for the Total Path.
```

Pseudo-Code 6: Modified Bisection Algorithm for 3-Equations and 2 Unknowns

```
Specify the bounds $\phi_s = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, $\phi_f = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

Optional: Calculate all solutions to the vector equation at a resolution of 3
            degrees. Select the 4 bounds surrounding the point of lowest
            error as the new bounds.

WHILE error > tolerance {

      FOR each current bound {

            Subdivide the bound into 4 each squares.

            Compute the 9 solutions at the corners of the subdivided bounds.

            FOR each of the 4 squares {

                  IF all 3 equations have a sign change among the 4 corners {
                        Store the bound.
                  }
            }
      }
}

Subdivide the resulting solution bound.

Compute the solution in the center.

Compare this solution to the 4 corners and return the solution with least
      error.
```

Pseudo-Code 7: 3D Dubins and Clothoid Path Generation Methodology

```
INPUTS:
Poses       [x y z ϕ θ κ] for each desired waypoint
Velocity    cruise velocity for the aircraft
SR          On-board computer sampling rate, determines trajectory resolution

OUTPUTS:
totalLength total length of the finalized trajectory
trajectory  full trajectory for the aircraft
_____________________________________________________________________________
For each turn direction combination {
      Compute the plane angles by iteratively solving the vector equation.
      If a solution exists{
            Compute the length of the path. Store the solution.
      }
}
Compare the path lengths and choose the shortest.
Compute the trajectory for this combination.
```

Pseudo-Code 8: 3D Dubins and Clothoid Full Path Planner

```
INPUTS:
Poses       [x y z ϕ  θ  κ  τ] for each desired waypoint
Velocity    cruise velocity for the aircraft
SR

OUTPUTS:
smoothPath  list of arc and straight maneuvers used to generate the
            trajectory, containing positions, headings, and sweep angles
totalLength total length of the finalized trajectory
trajectory  full trajectory for the aircraft
_____________________________________________________________________________
For each set of poses {

            Check if the combination is coplanar.
            If coplanar {
                  Determine the orientation of the solution plane.
                  Generate the 2D solutions, choosing the shortest path.
                  Generate the trajectory.
                  Use the plane orientation to convert the 2D solution to the
                        Earth coordinate system.
            }
            Else {
                  For each combination of turn directions {
                        Solve the vector equations, and determine sweep
                              angles for each set of pose combinations.
                        }
                  Choose the shortest path.
                  Calculate the trajectory.
                  }
            }
            Increment the total path length.
      }
}
```

B UNABRIDGED 2D CLOTHOID VERSUS DUBINS TRAJECTORY TRACKING COMPARISON RESULTS

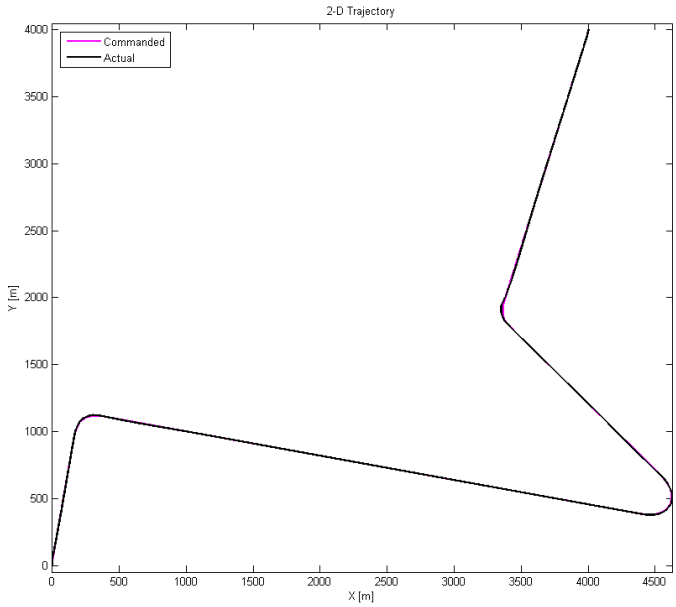

Figure B-1—Clothoid Trajectory Tracking Results for Simple Path at Nominal Conditions

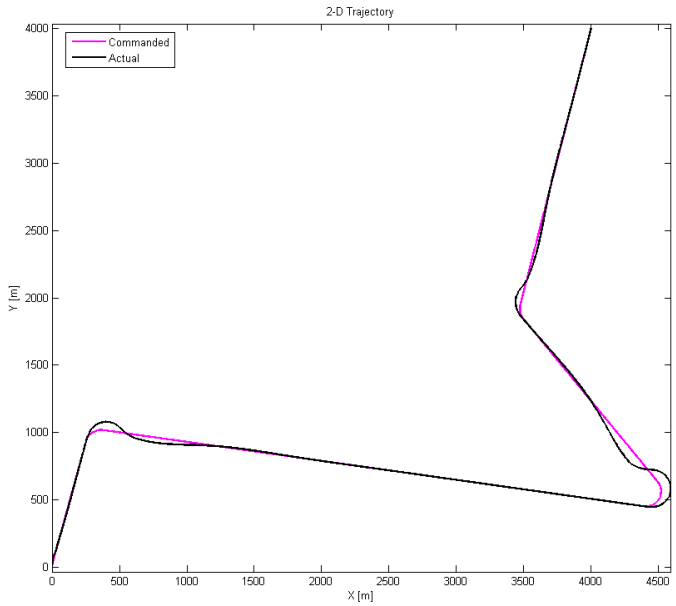

Figure B-2—Dubins Trajectory Tracking Results for Simple Path at Nominal Conditions

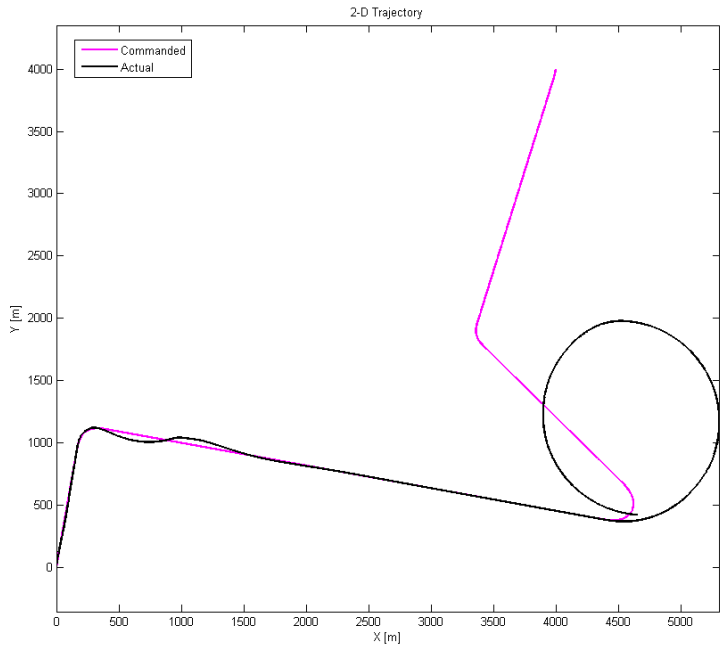

Figure B-3—Clothoid Trajectory Tracking Results for Simple Path with Left Aileron Stuck at 7°

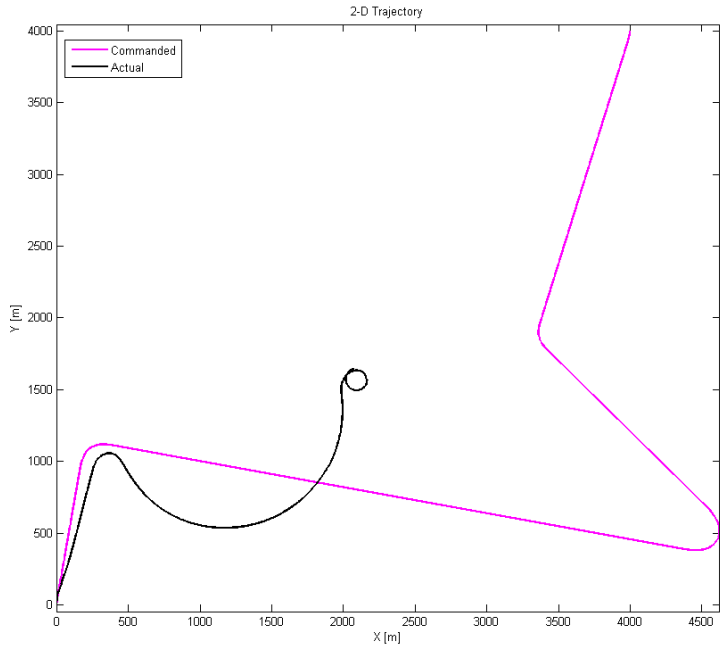

Figure B-4—Dubins Trajectory Tracking Results for Simple Path with Left Aileron Stuck at 7°

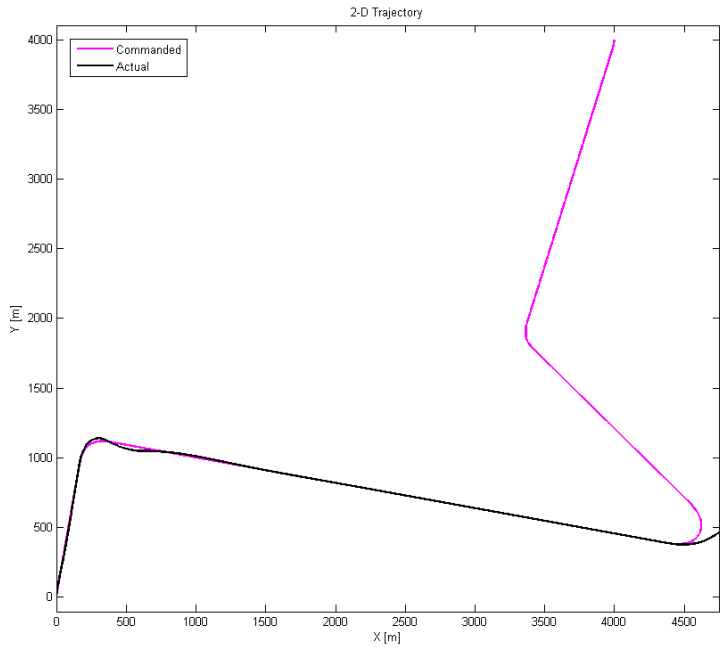

Figure B-5—Clothoid Trajectory Tracking Results for Simple Path with Left Elevator Stuck at 7°

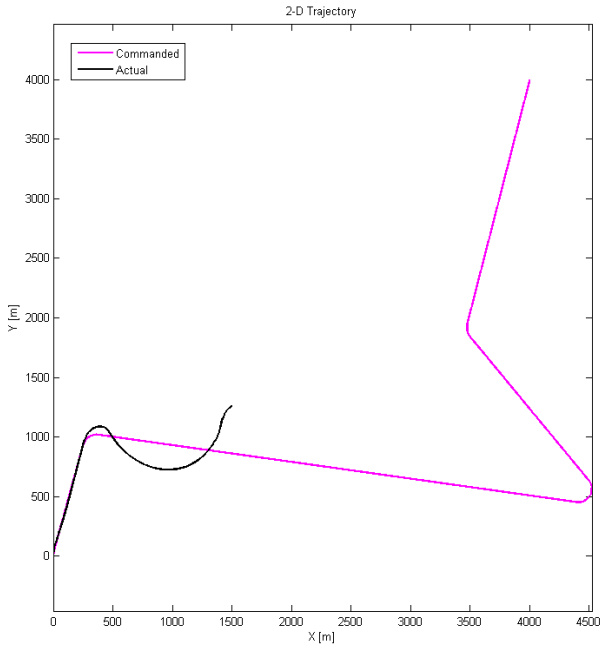

Figure B-6—Dubins Trajectory Tracking Results for Simple Path with Left Elevator Stuck at 7°

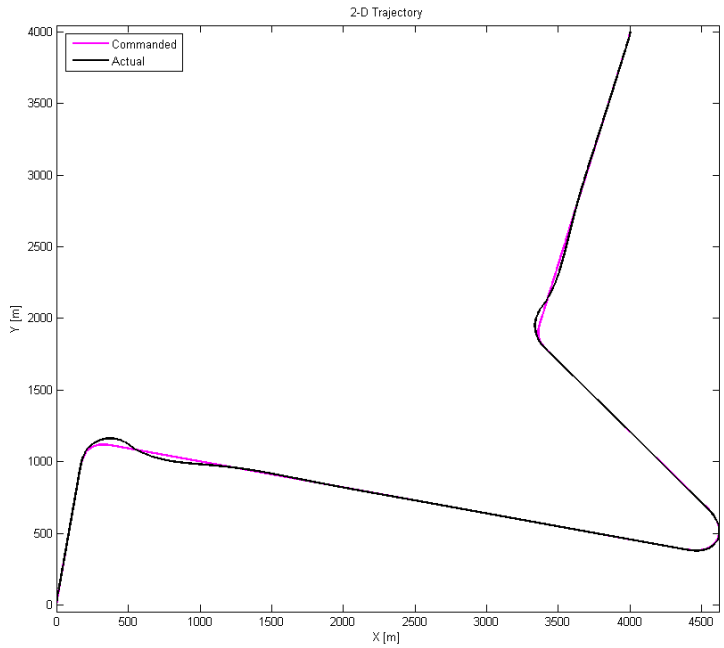

Figure B-7—Clothoid Trajectory Tracking Results for Simple Path with Left Rudder Stuck at 8°

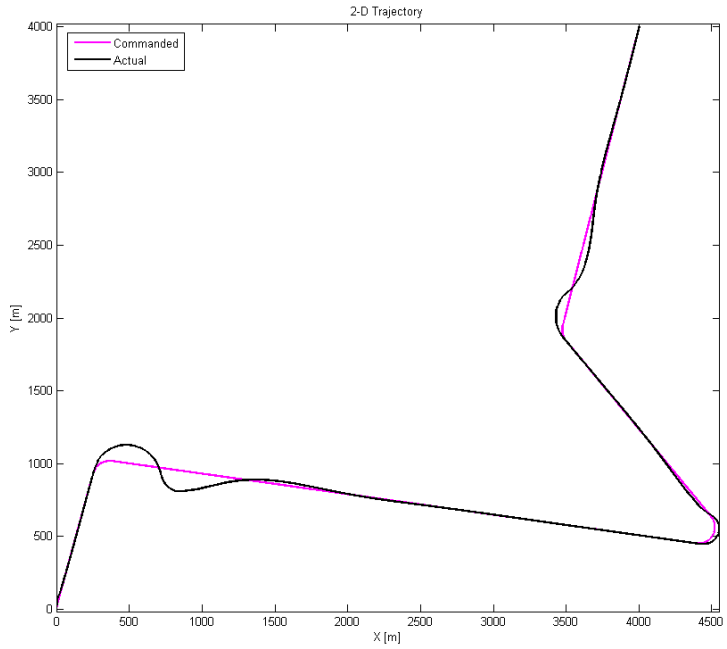

Figure B-8—Dubins Trajectory Tracking Results for Simple Path with Left Rudder Stuck at 8°

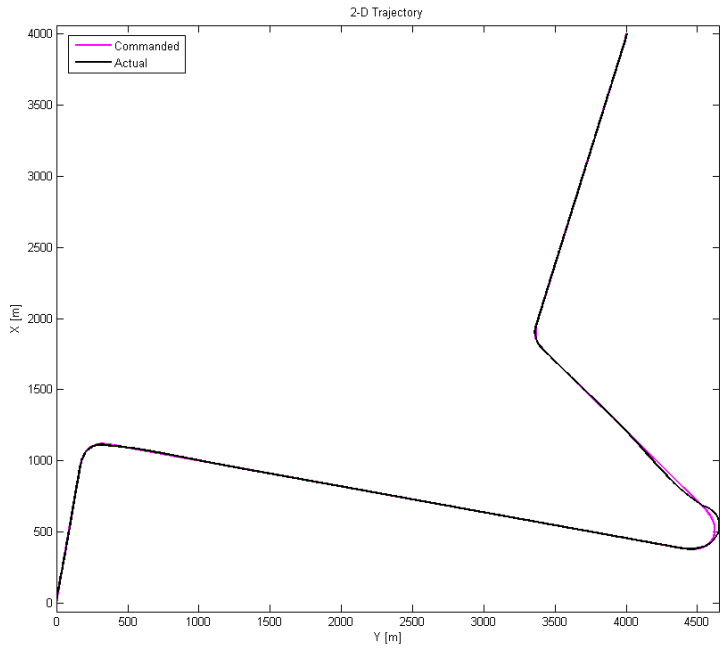

Figure B-9—Clothoid Trajectory Tracking Results for Simple Path with High Wind Turbulence

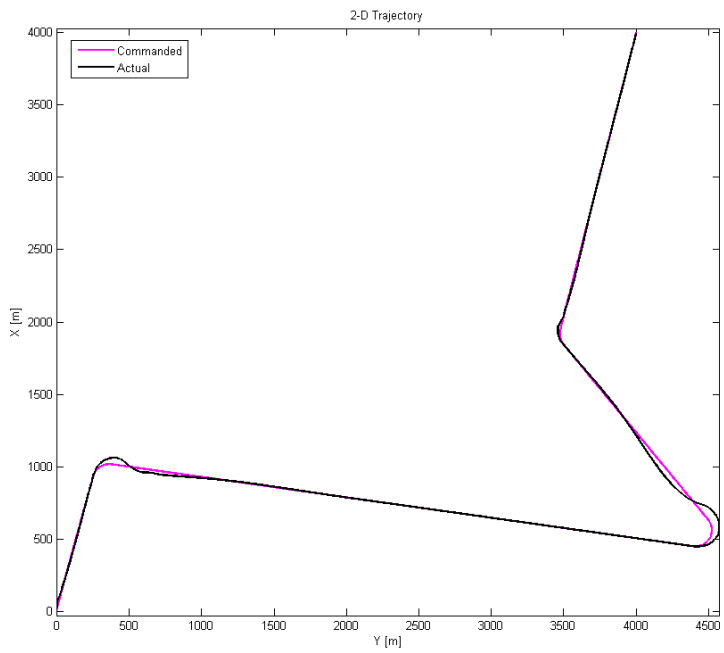

Figure B-10—Dubins Trajectory Tracking Results for Simple Path with High Wind Turbulence

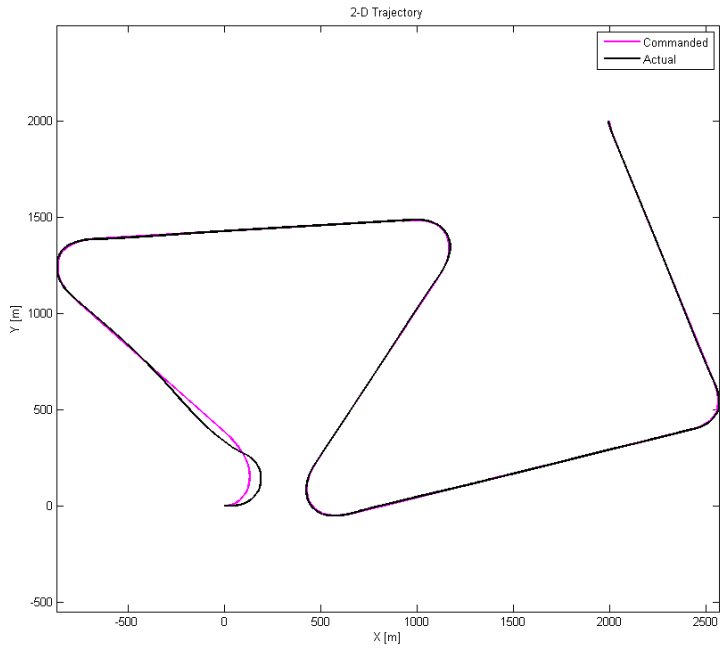

Figure B-11—Clothoid Trajectory Tracking Results for Moderate Path at Nominal Conditions

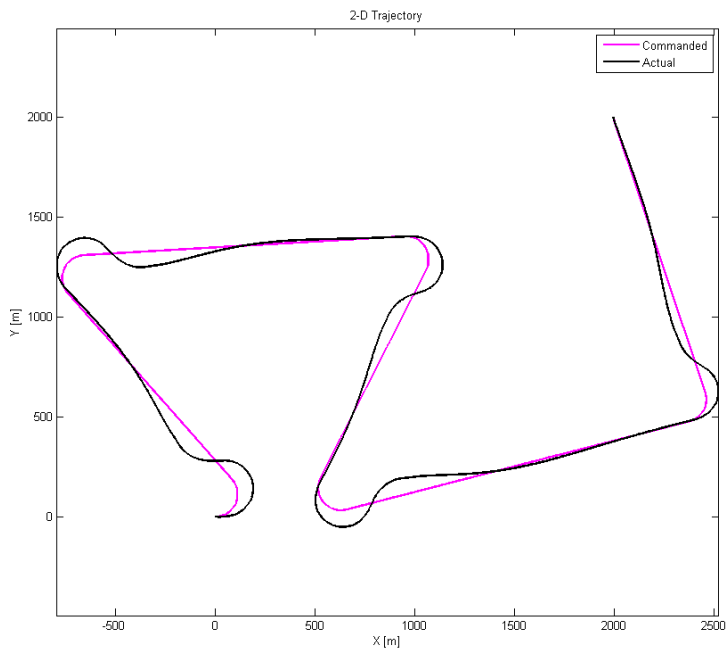

Figure B-12—Dubins Trajectory Tracking Results for Moderate Path at Nominal Conditions

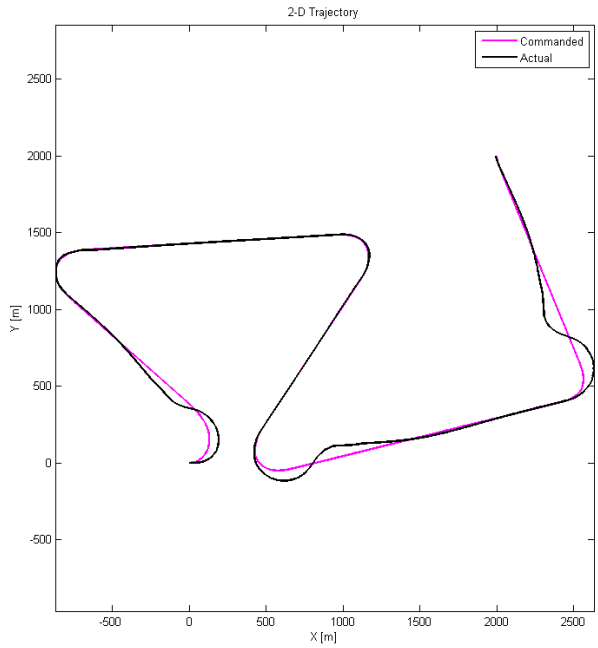

Figure B-13—Clothoid Trajectory Tracking Results for Moderate Path with Left Aileron Stuck at 2°

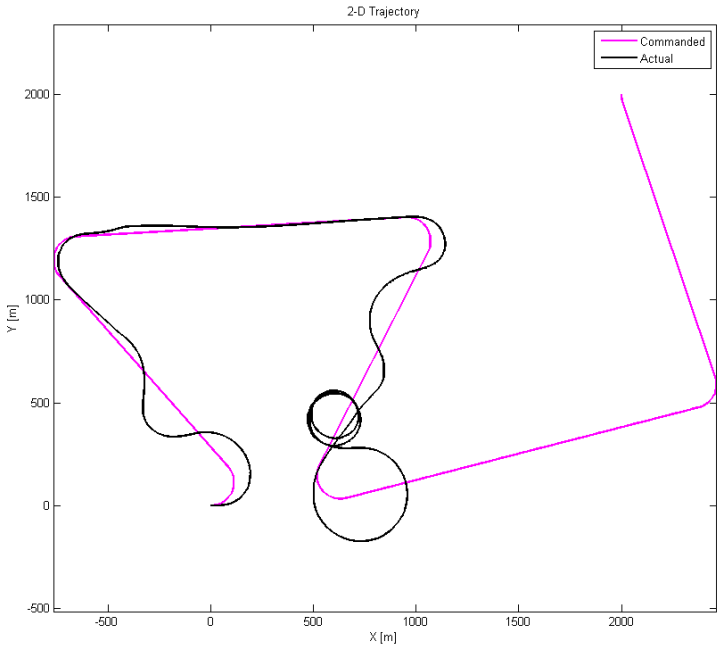

Figure B-14—Dubins Trajectory Tracking Results for Moderate Path with Left Aileron Stuck at 2°

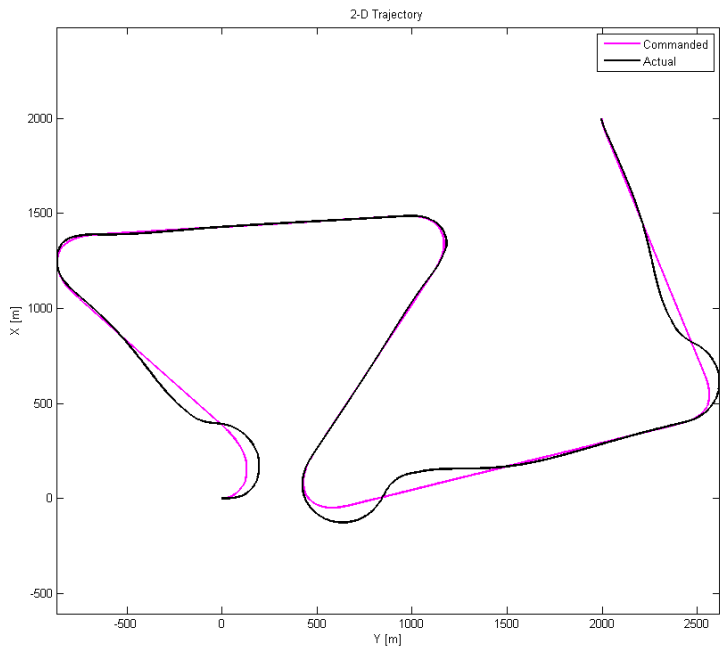

Figure B-15—Clothoid Trajectory Tracking Results for Moderate Path with Left Elevator Stuck at 2°

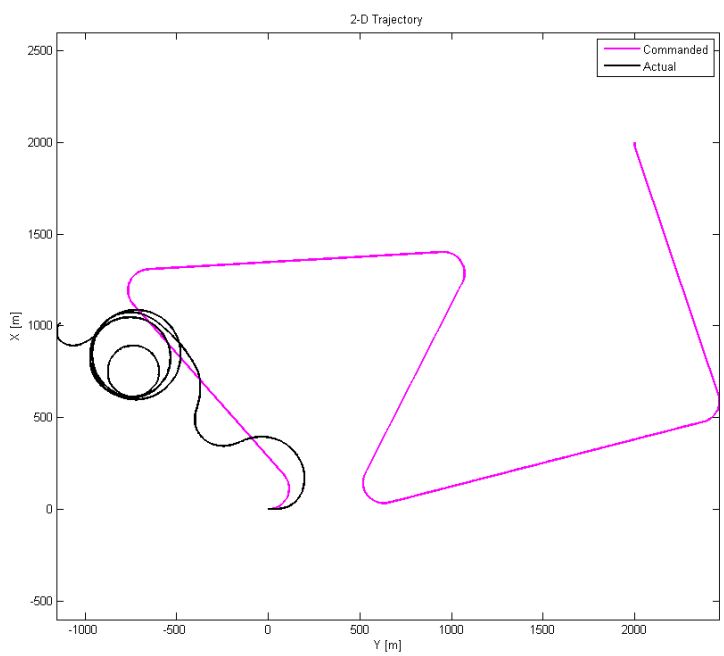

Figure B-16—Dubins Trajectory Tracking Results for Moderate Path with Left Elevator Stuck at 2°

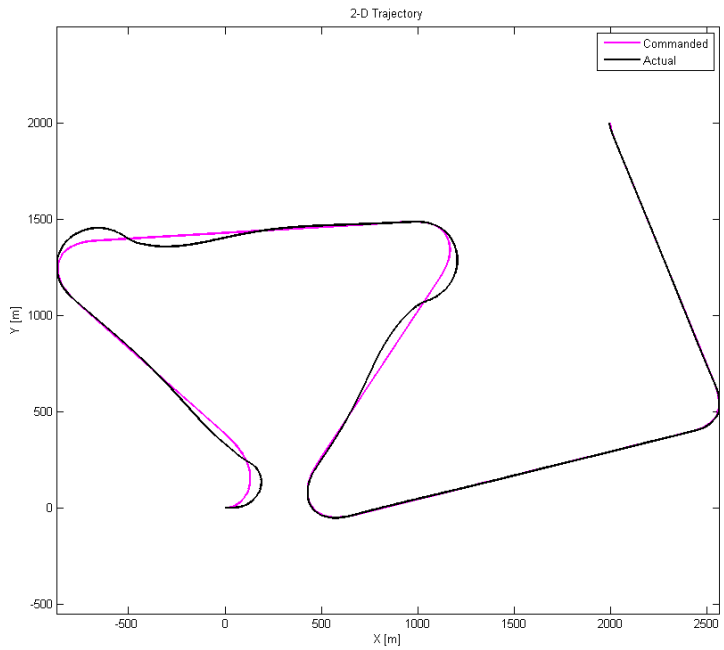

Figure B-17—Clothoid Trajectory Tracking Results for Moderate Path with Left Rudder Stuck at 8°

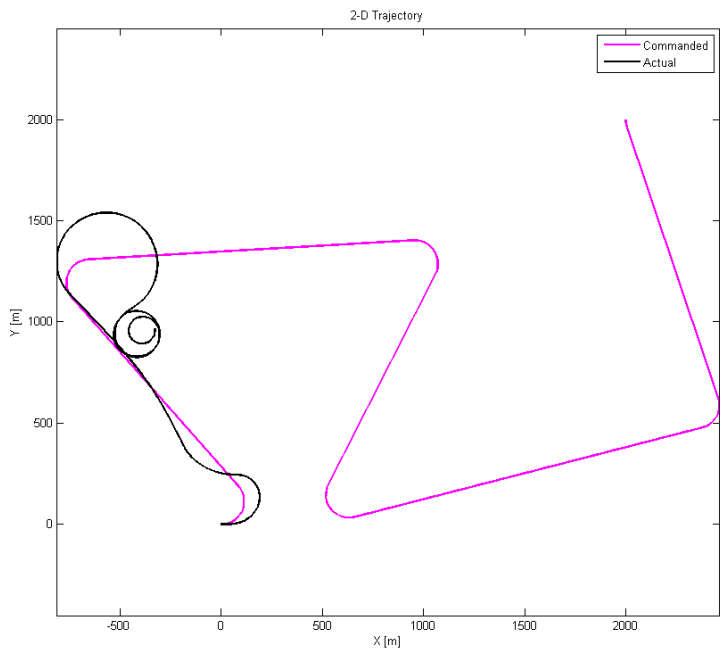

Figure B-18—Dubins Trajectory Tracking Results for Moderate Path with Left Rudder Stuck at 8°

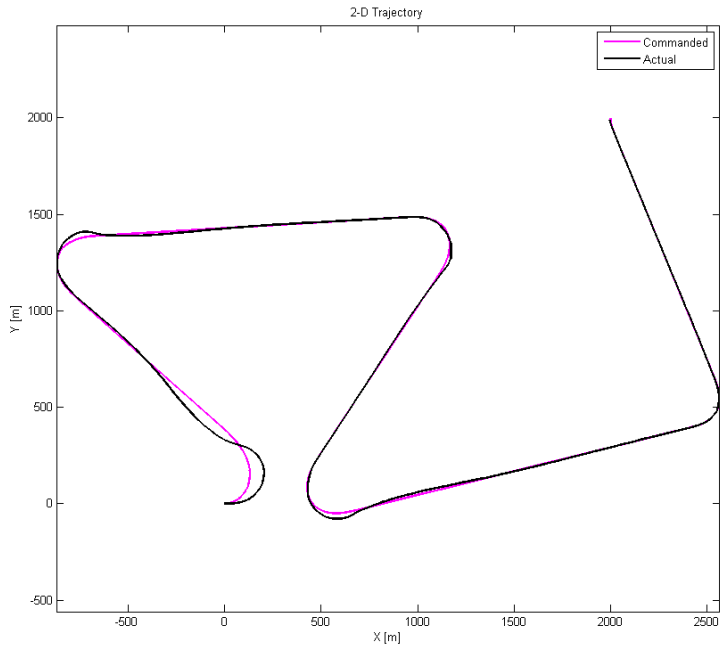

Figure B-19—Clothoid Trajectory Tracking Results for Moderate Path with High Wind Turbulence

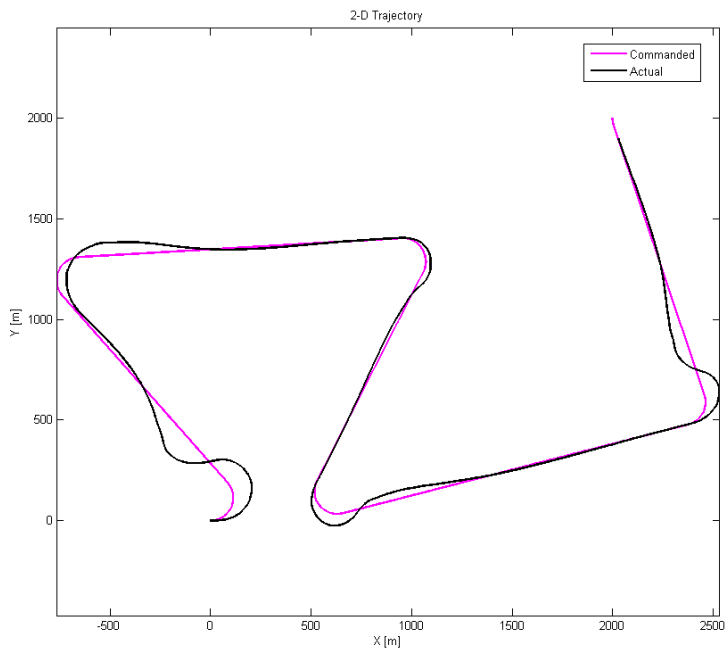

Figure B-20—Dubins Trajectory Tracking Results for Moderate Path with High Wind Turbulence

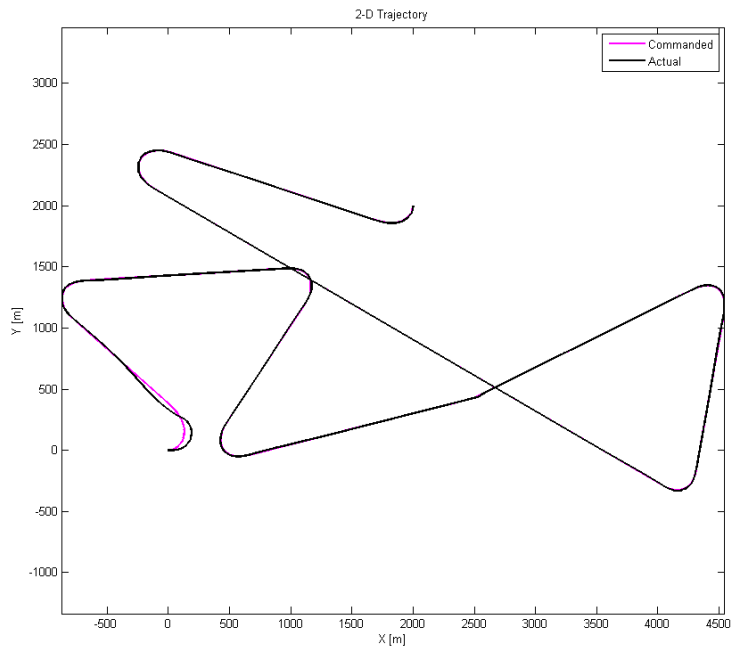

Figure B-21—Clothoid Trajectory Tracking Results for Complex Path at Nominal Conditions

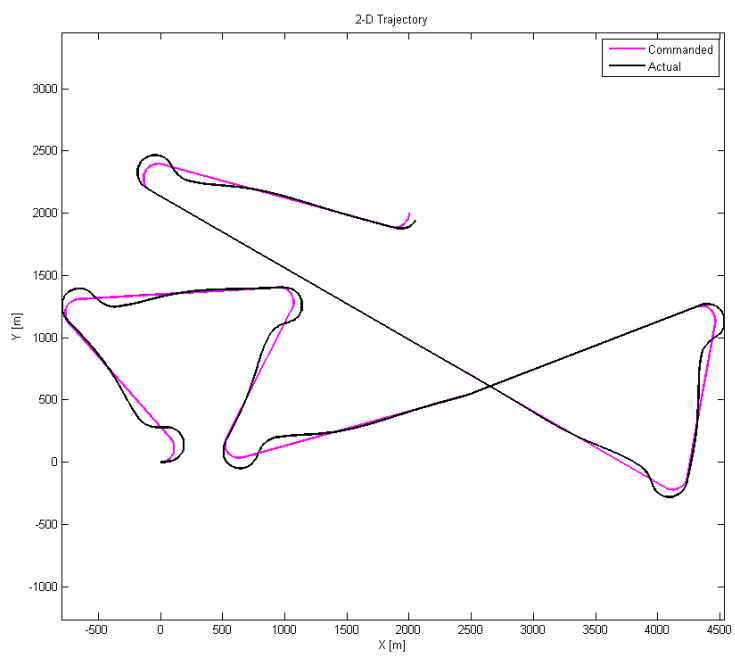

Figure B-22—Dubins Trajectory Tracking Results for Complex Path at Nominal Conditions

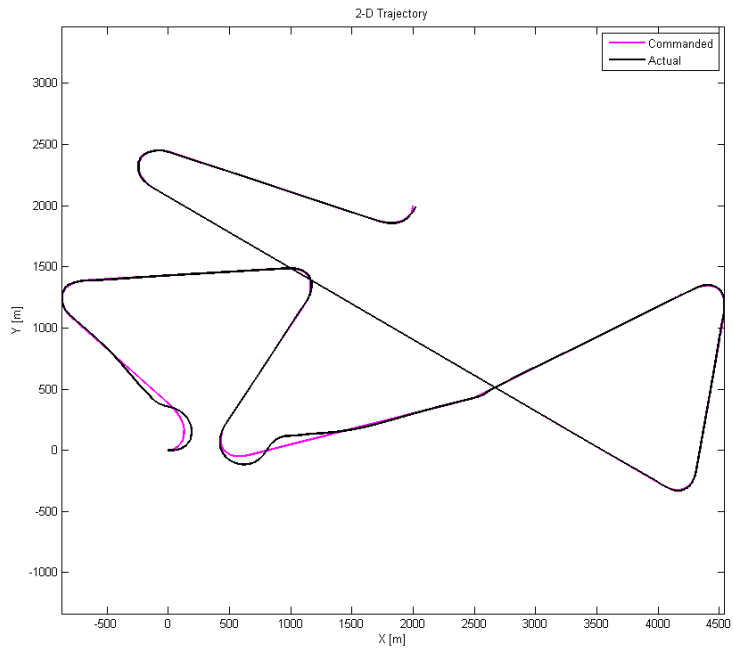

Figure B-23—Clothoid Trajectory Tracking Results for Complex Path with Left Aileron Stuck at 2°

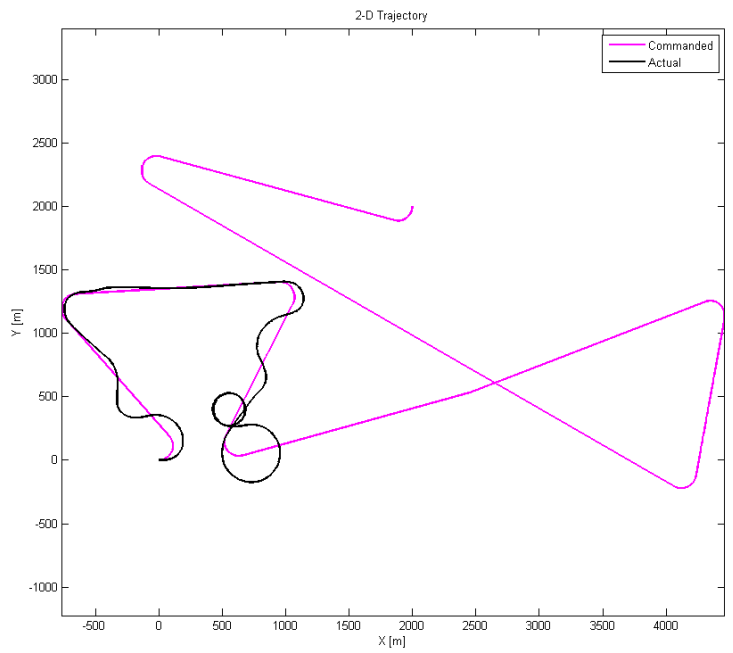

Figure B-24—Dubins Trajectory Tracking Results for Complex Path with Left Aileron Stuck at 2°

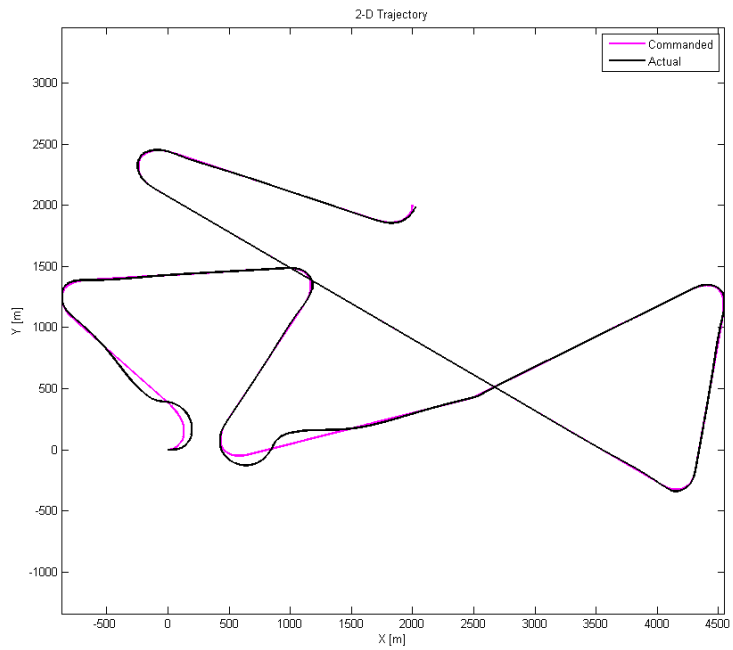

Figure B-25—Clothoid Trajectory Tracking Results for Complex Path with Left Elevator Stuck at 2°

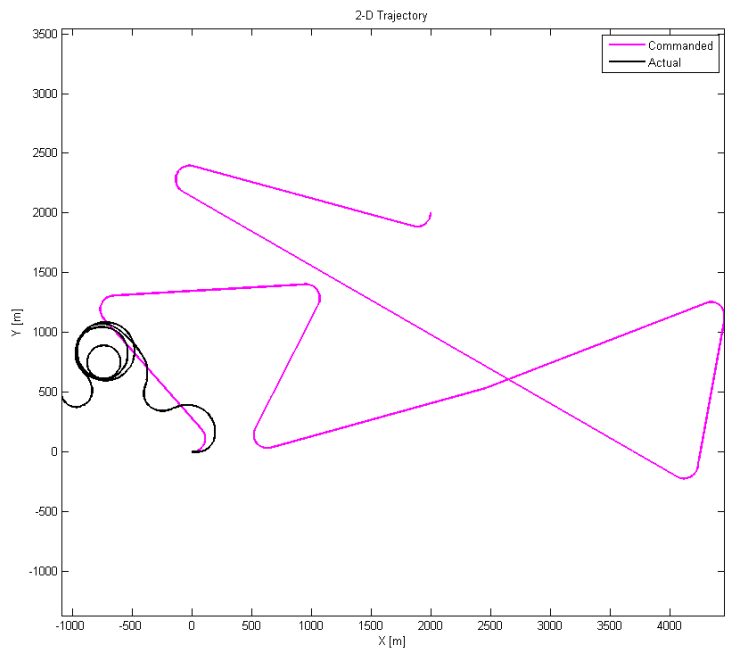

Figure B-26—Dubins Trajectory Tracking Results for Complex Path with Left Elevator Stuck at 2°

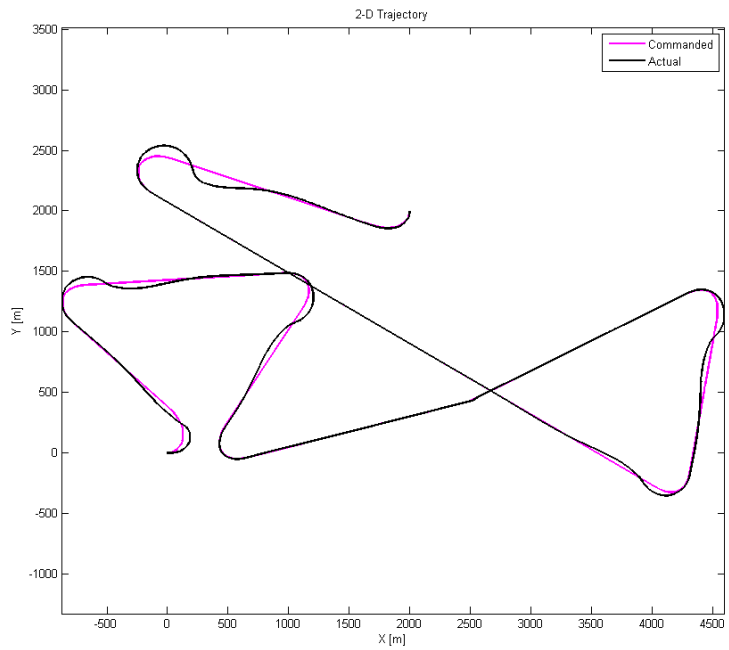

Figure B-27—Clothoid Trajectory Tracking Results for Complex Path with Left Rudder Stuck at 8°

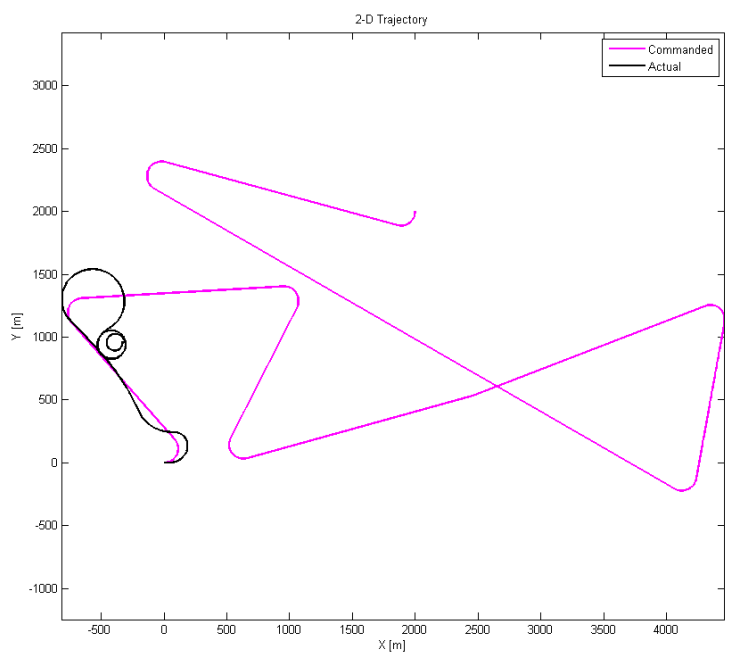

Figure B-28—Dubins Trajectory Tracking Results for Complex Path with Left Rudder Stuck at 8°

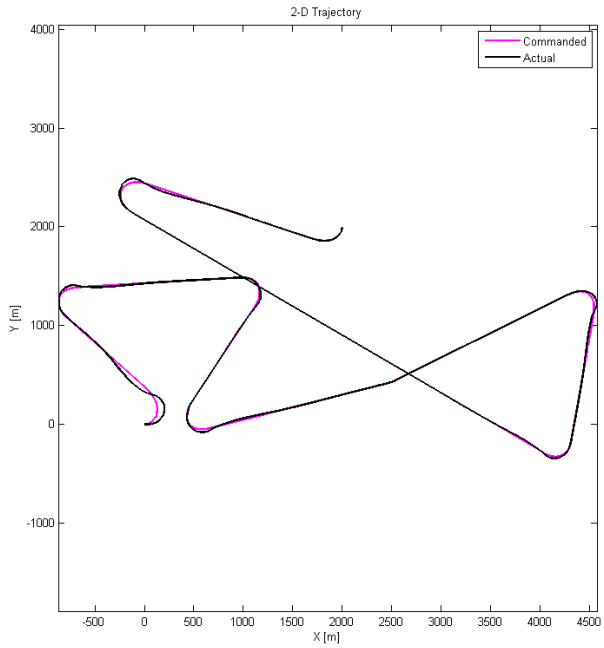

Figure B-29—Clothoid Trajectory Tracking Results for Complex Path with High Wind Turbulence

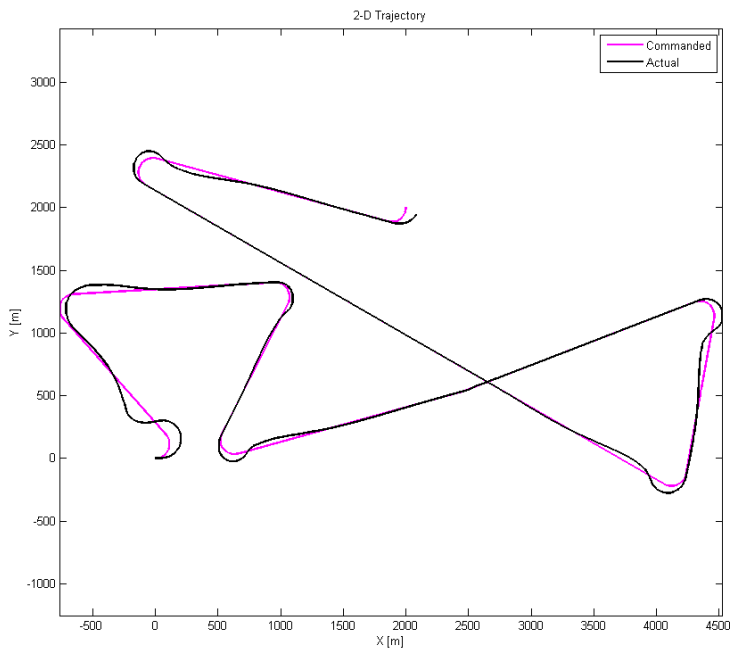

Figure B-30—Dubins Trajectory Tracking Results for Complex Path with High Wind Turbulence

C UNABRIDGED 3D CLOTHOID VERSUS DUBINS TRAJECTORY TRACKING COMPARISON RESULTS

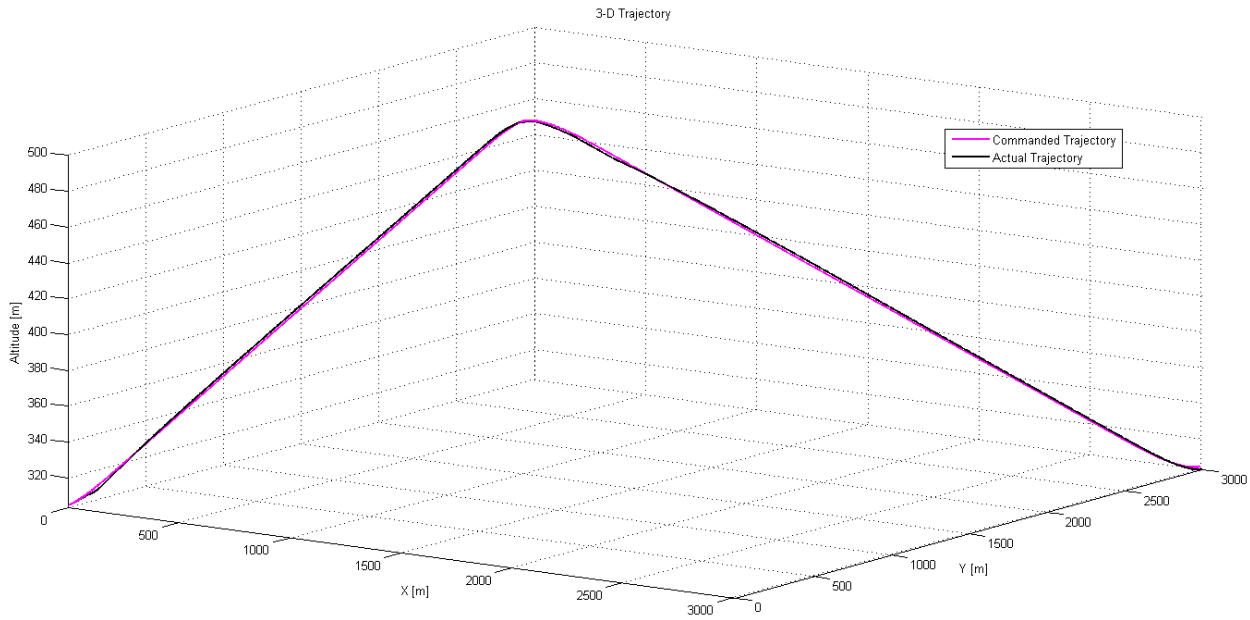

Figure C-1—Clothoid Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for PPID Controller, 3D View

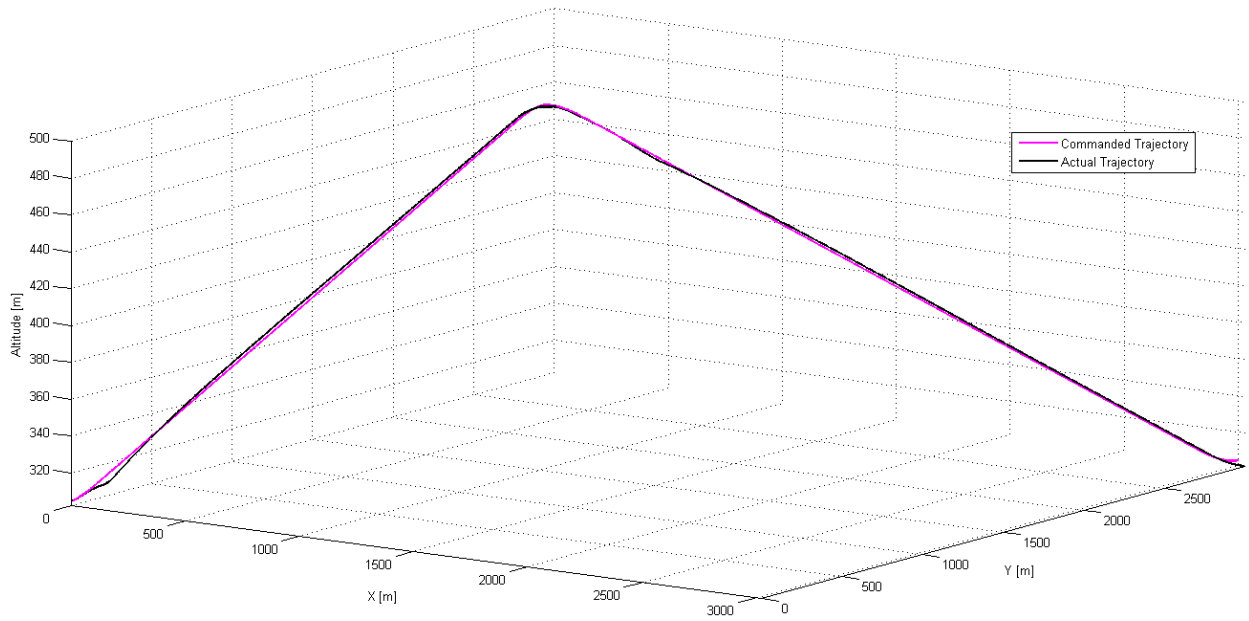

Figure C-2—Dubins Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for PPID Controller, 3D View

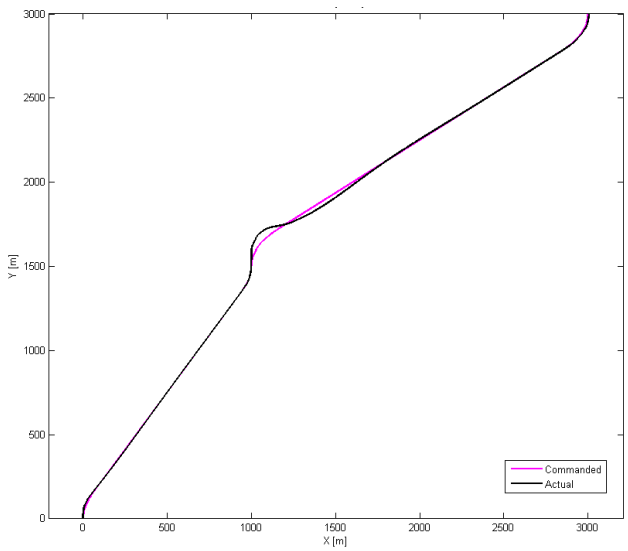

Figure C-3—Clothoid Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for PPID Controller, 2D View

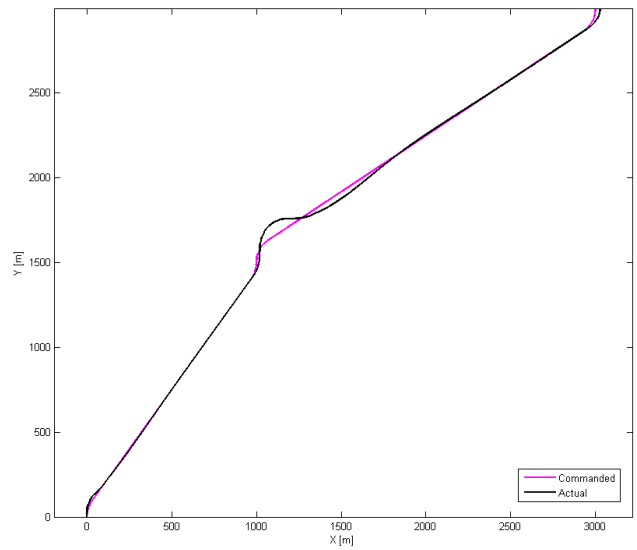

Figure C-4—Dubins Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for PPID Controller, 2D View

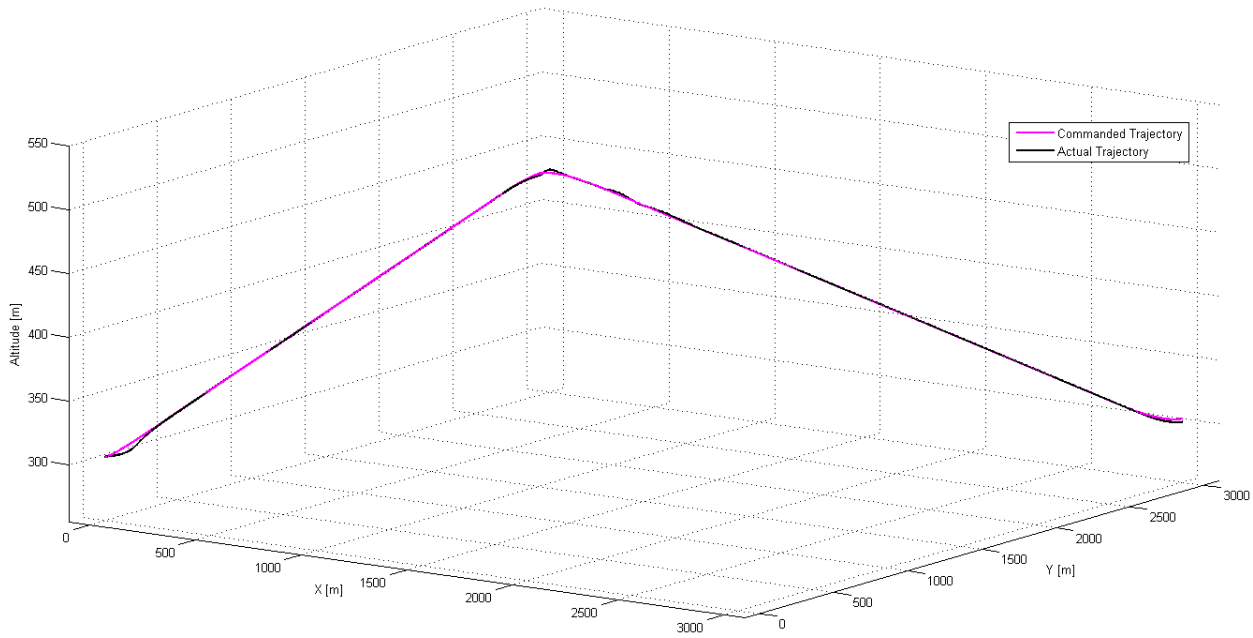

Figure C-5—Clothoid Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for ONLDI Controller, 3D View

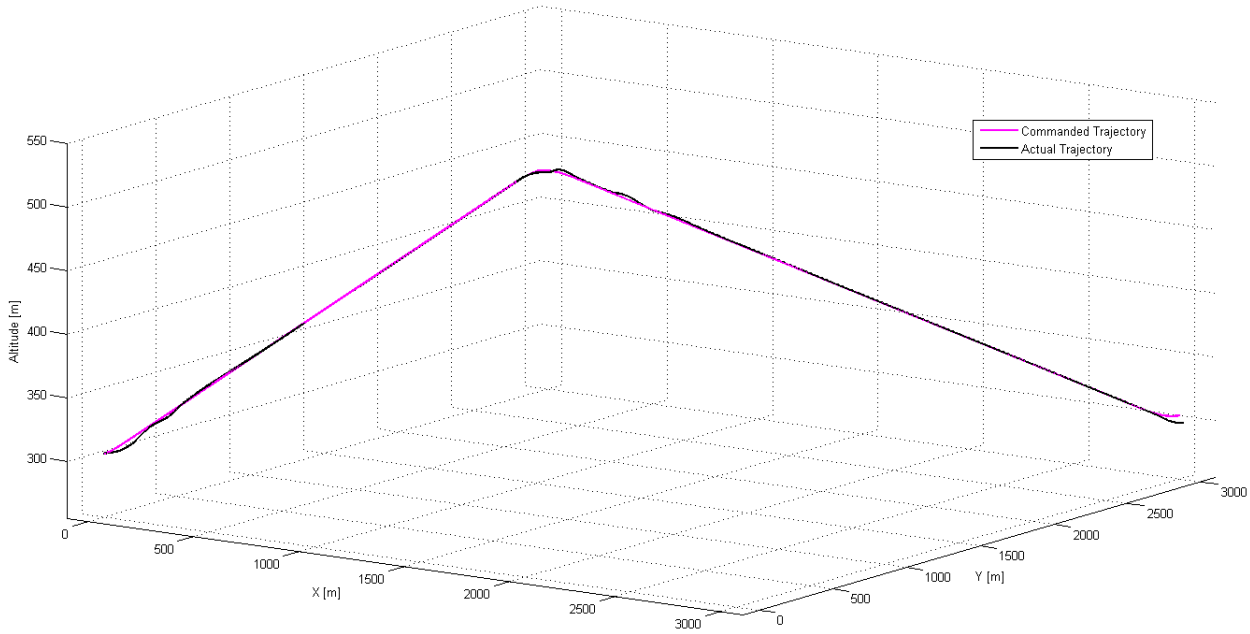

Figure C-6—Dubins Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for ONLDI Controller, 3D View

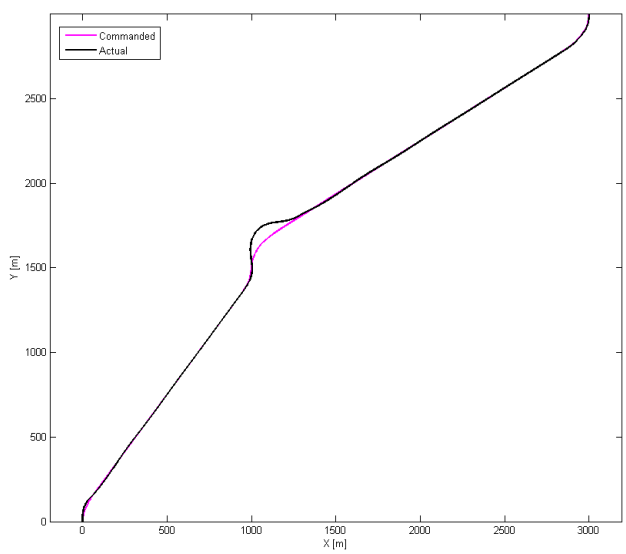

Figure C-7—Clothoid Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for ONLDI Controller, 2D View

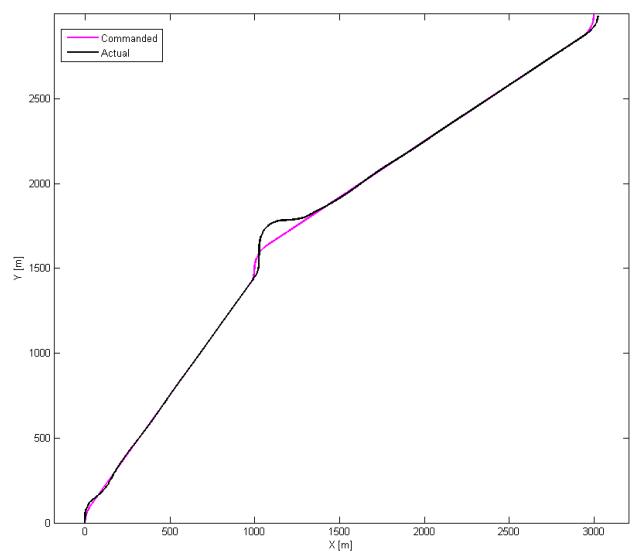

Figure C-8—Dubins Trajectory Tracking Results for Basic 3D Path at Nominal Conditions for ONLDI Controller, 2D View

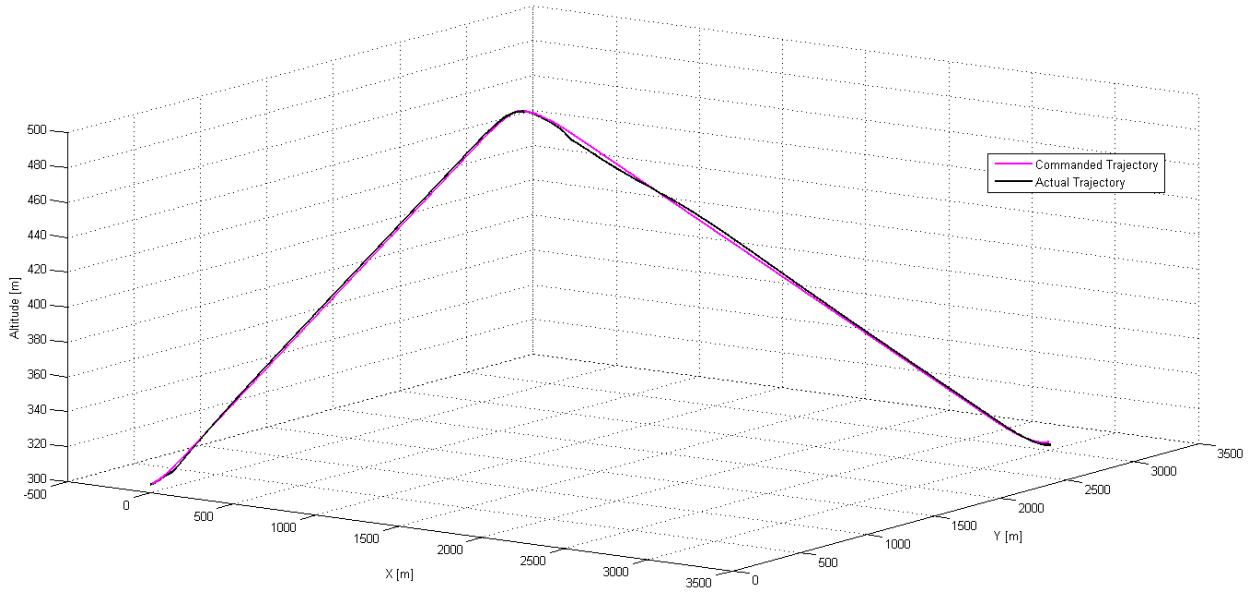

Figure C-9—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for PPID Controller, 3D View

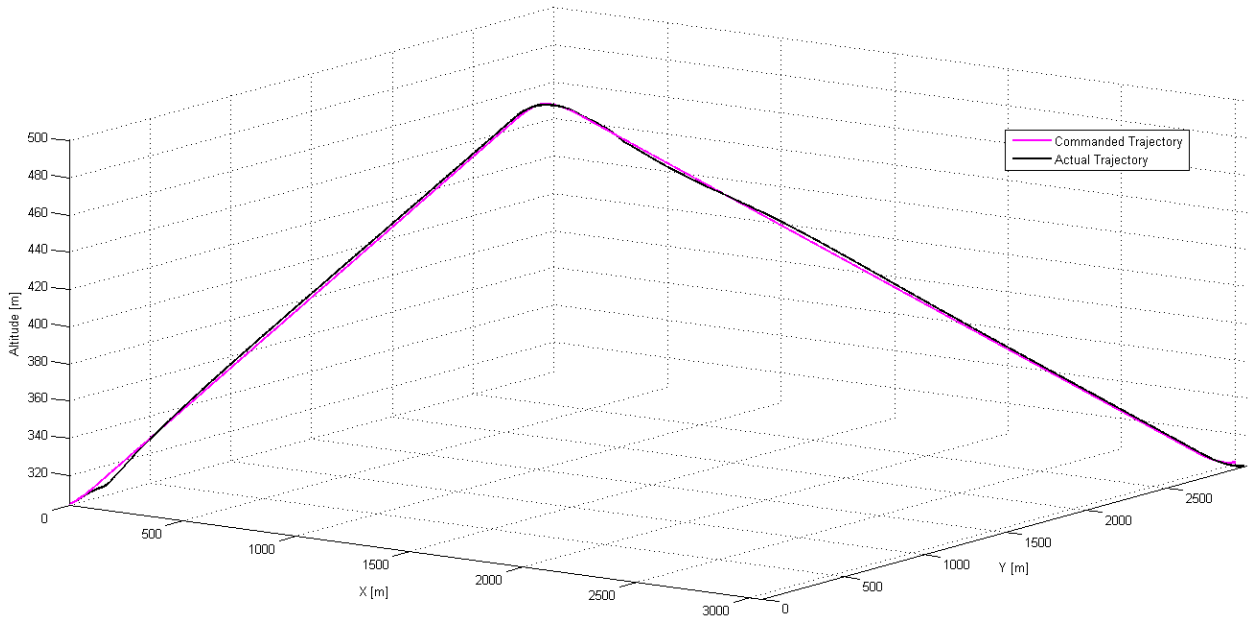

Figure C-10—Dubins Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for PPID Controller, 3D View

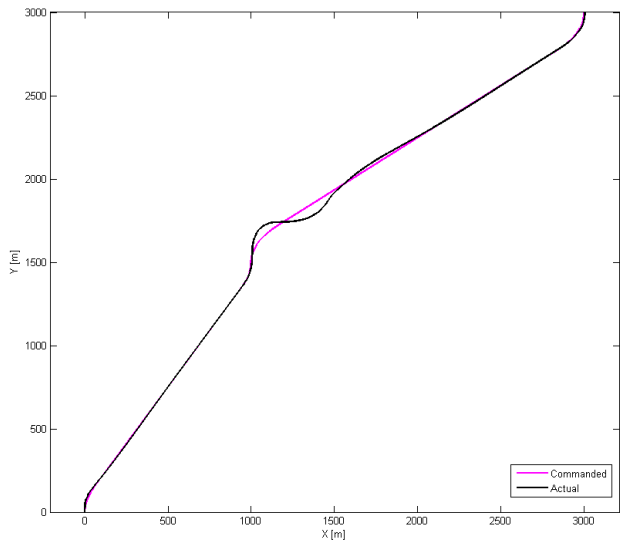

Figure C-11—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for PPID Controller, 2D View

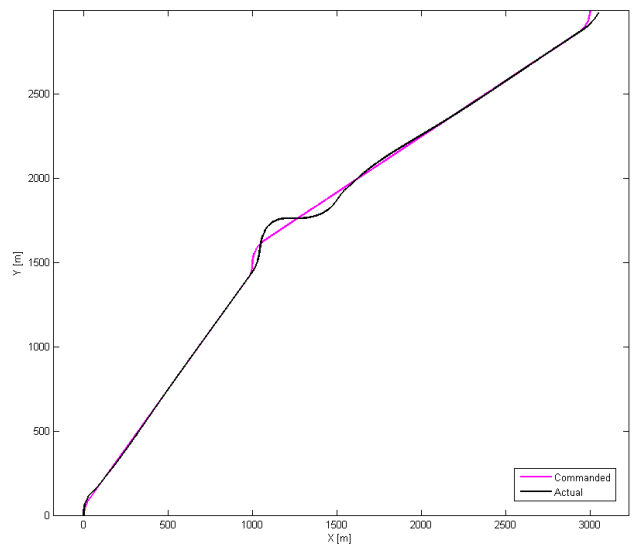

Figure C-12—Dubins Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for ON PPID LDI Controller, 2D View

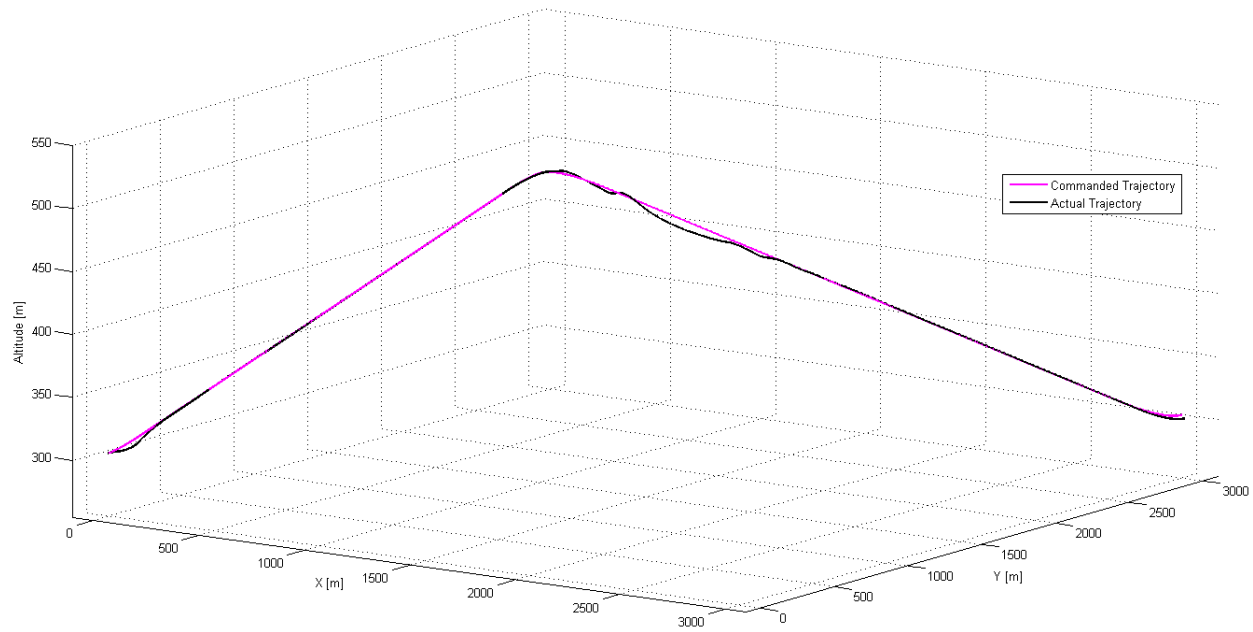

Figure C-13—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 3D View

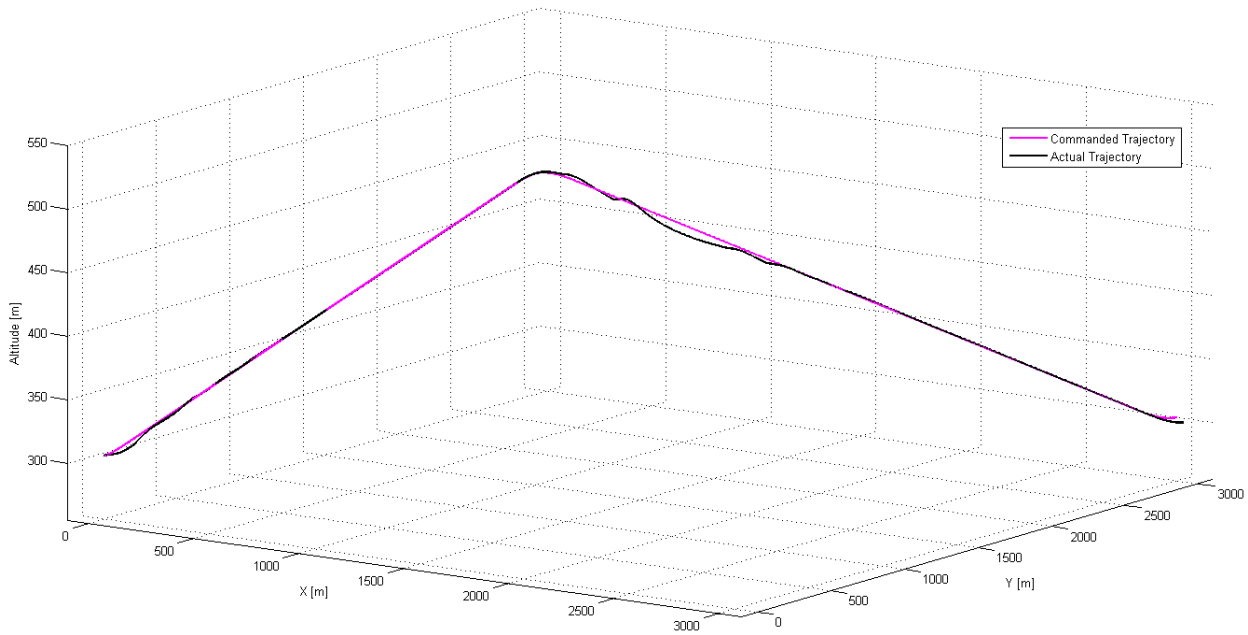

Figure C-14—Dubins Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 3D View

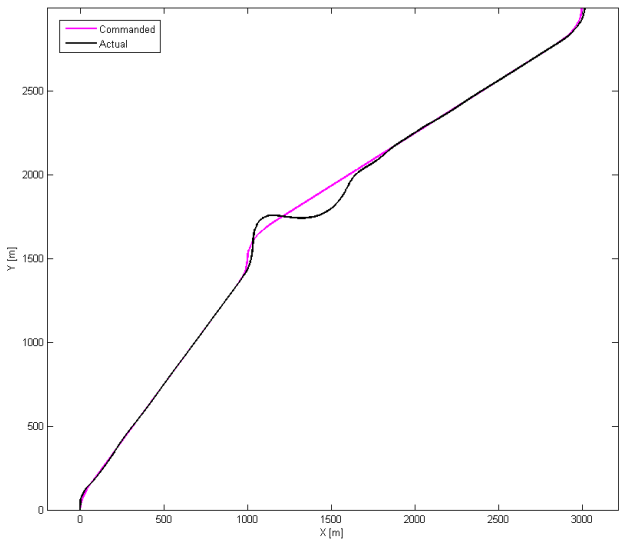

Figure C-15—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 2D View

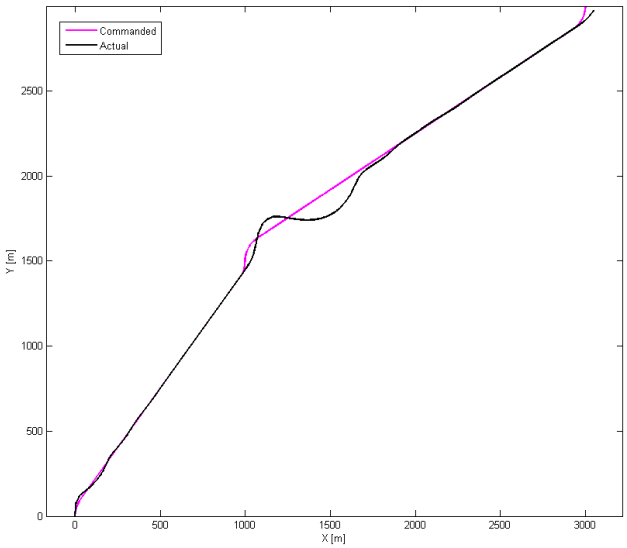

Figure C-16—Dubins Trajectory Tracking Results for Basic 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 2D View

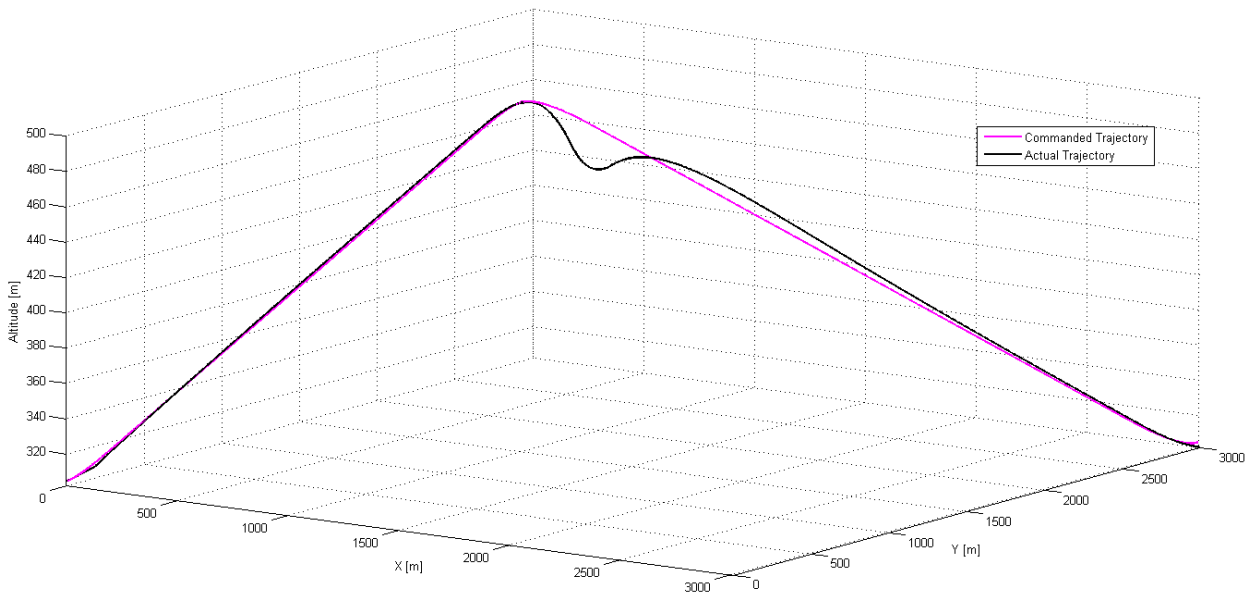

Figure C-17—Clothoid Trajectory Tracking Results for Basic 3D with Left Elevator Stuck at 2° for PPID Controller, 3D View

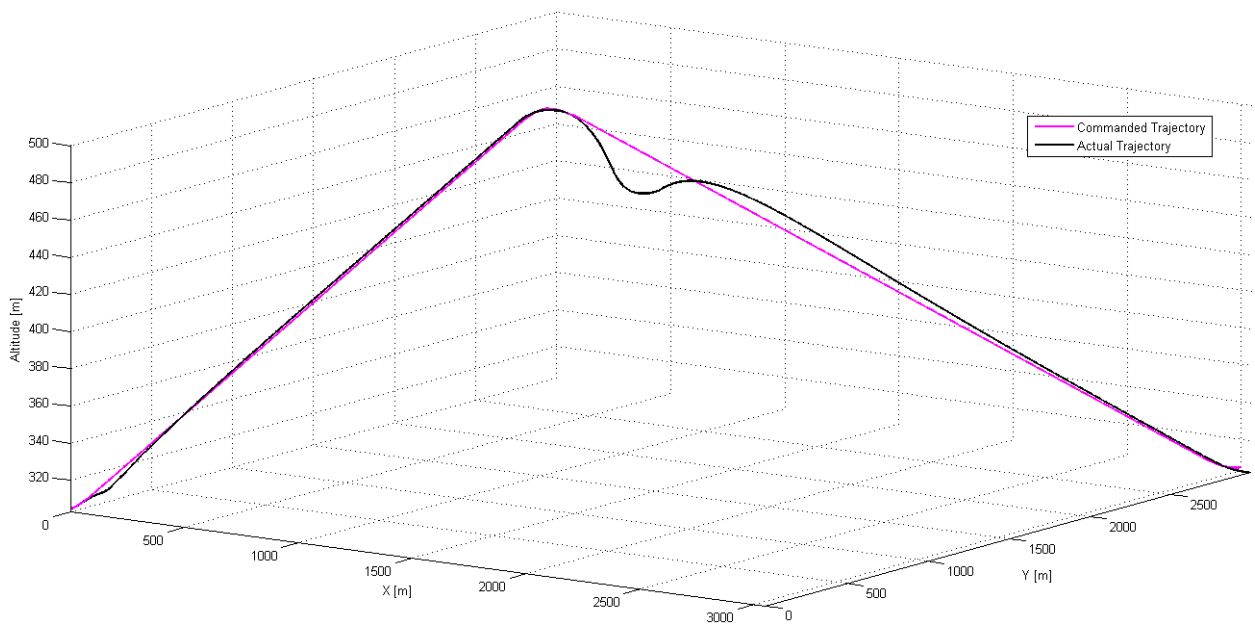

Figure C-18—Dubins Trajectory Tracking Results for Basic 3D Path with Left Elevator Stuck at 2° for PPID Controller, 3D View

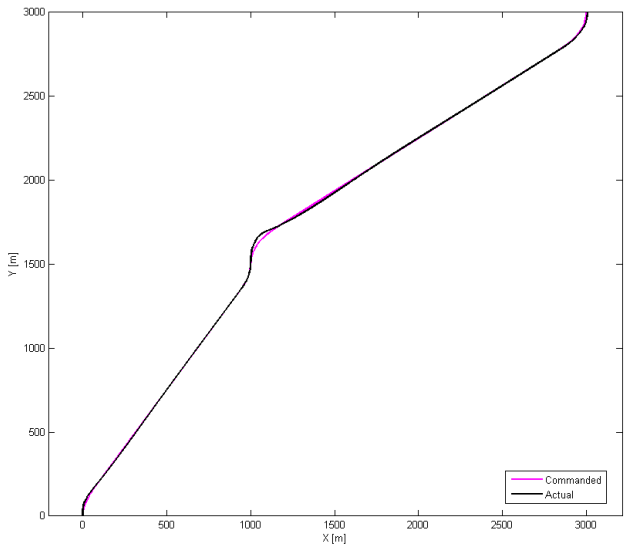

Figure C-19—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Elevator Stuck at 2° for PPID Controller, 2D View

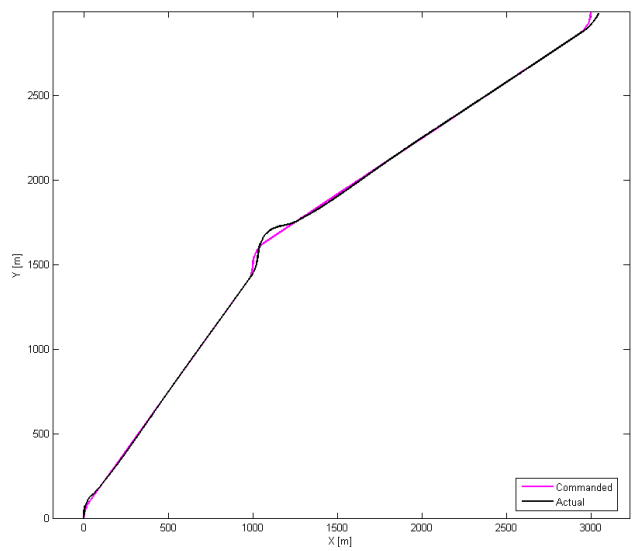

Figure C-20—Dubins Trajectory Tracking Results for Basic 3D Path with Left Elevator Stuck at 2° for PPID Controller, 2D View

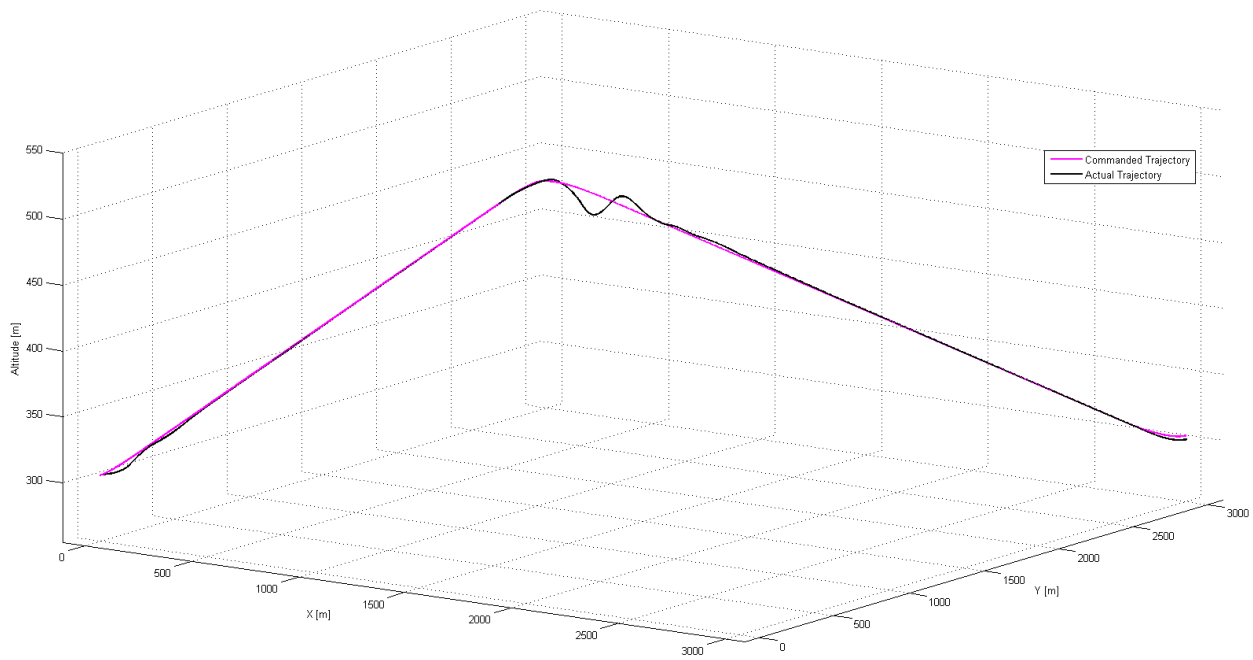

Figure C-21—Clothoid Trajectory Tracking Results for Basic 3D with Left Elevator Stuck at 2° for ONLDI Controller, 3D View

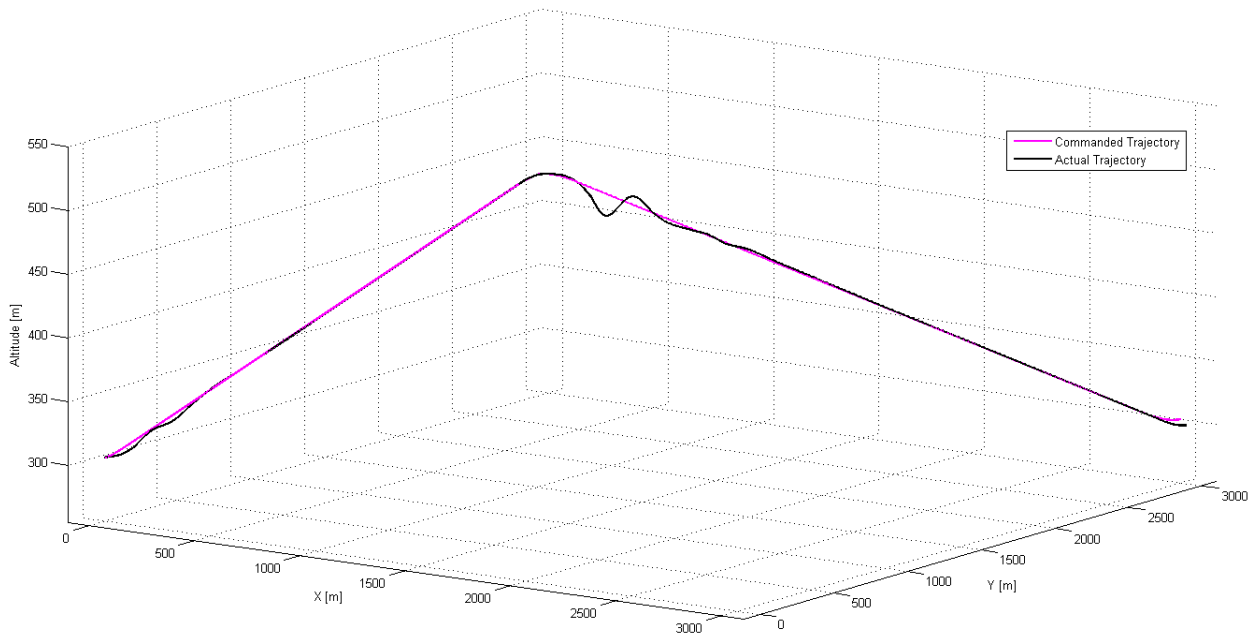

Figure C-22—Dubins Trajectory Tracking Results for Basic 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 3D View

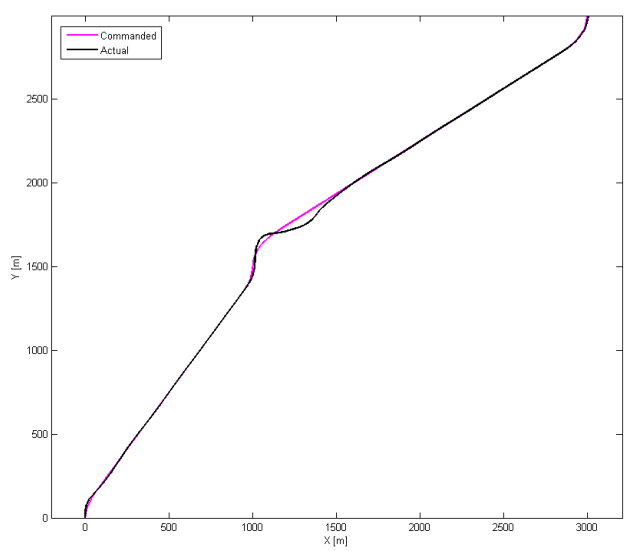

Figure C-23—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 2D View

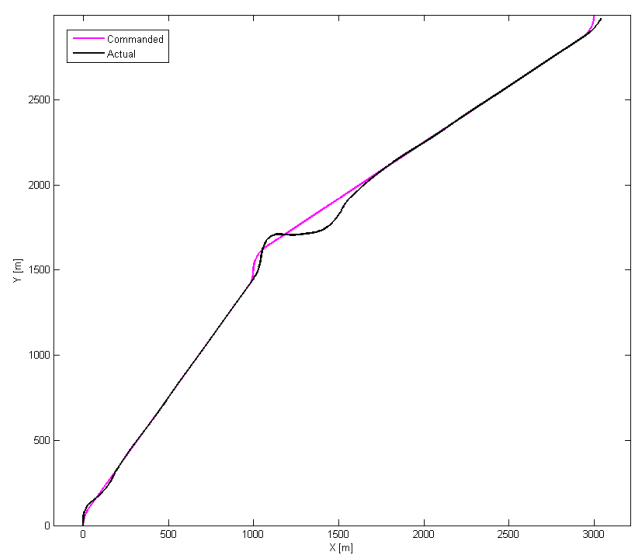

Figure C-24—Dubins Trajectory Tracking Results for Basic 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 2D View

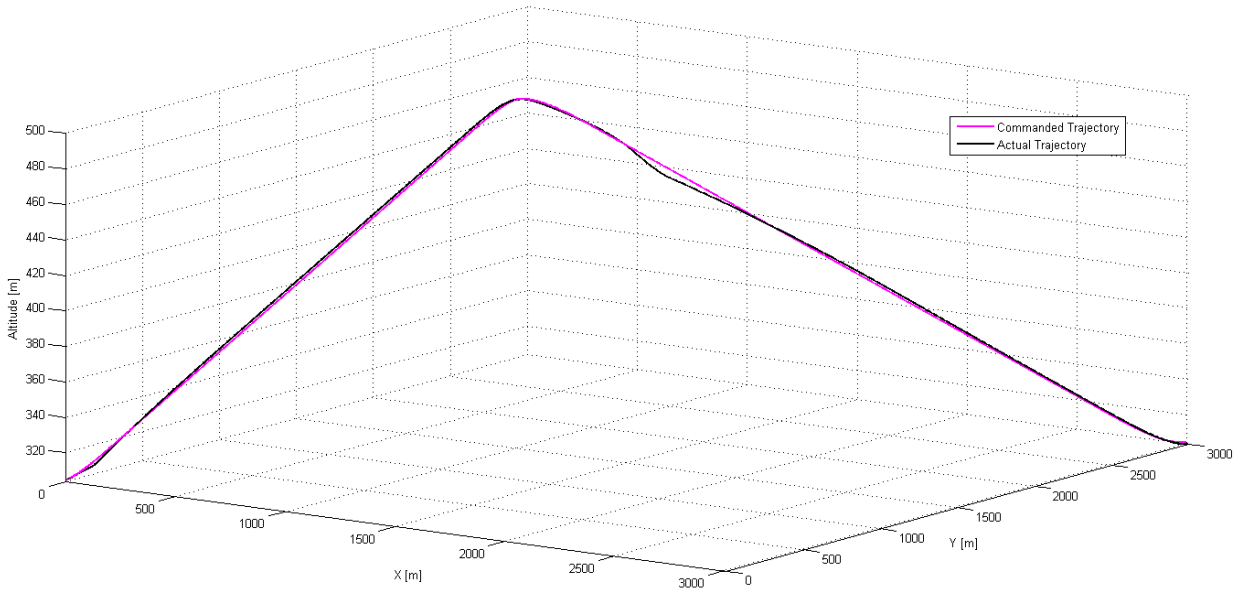

Figure C-25—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for PPID Controller, 3D View

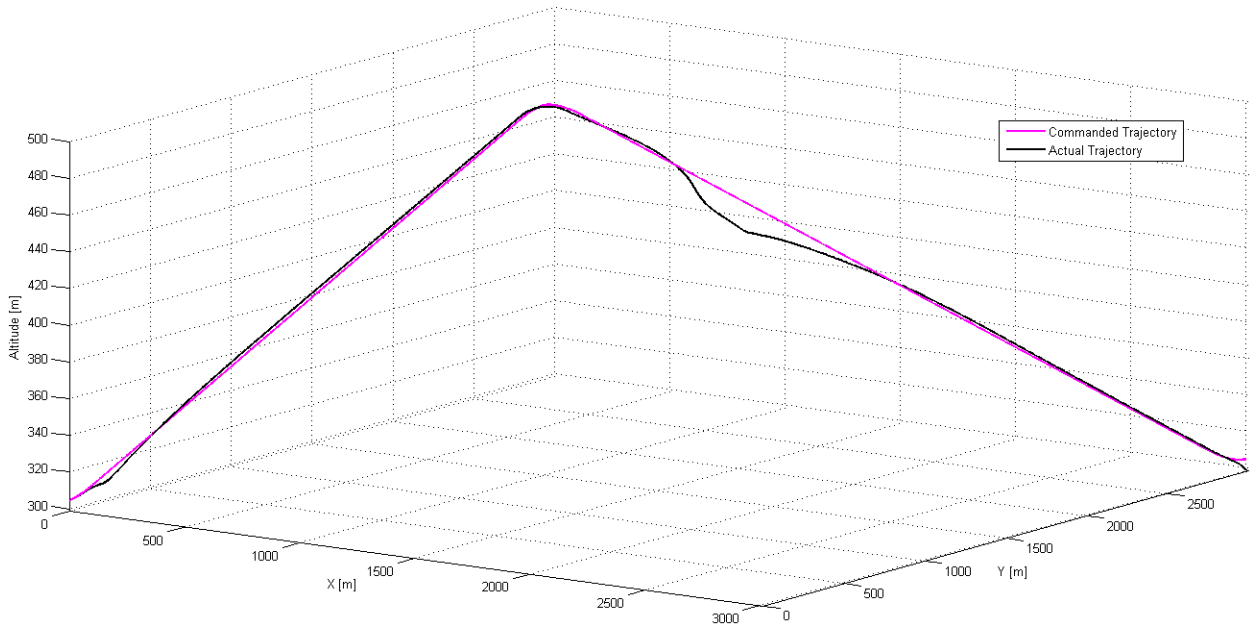

Figure C-26—Dubins Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for PPID Controller, 3D View

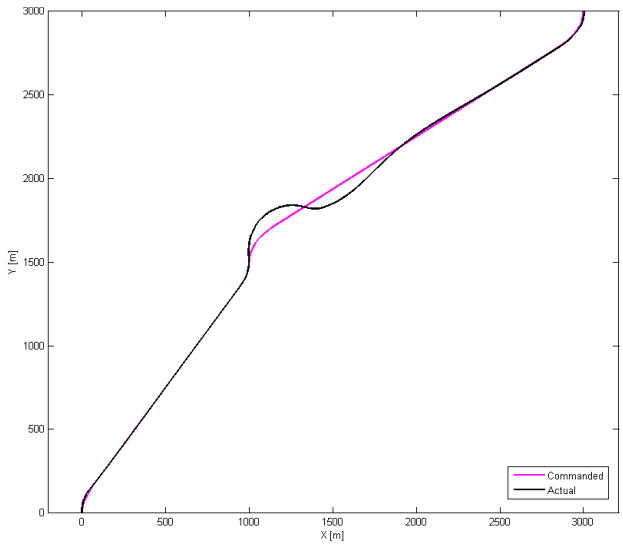

Figure C-27—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for PPID Controller, 2D View

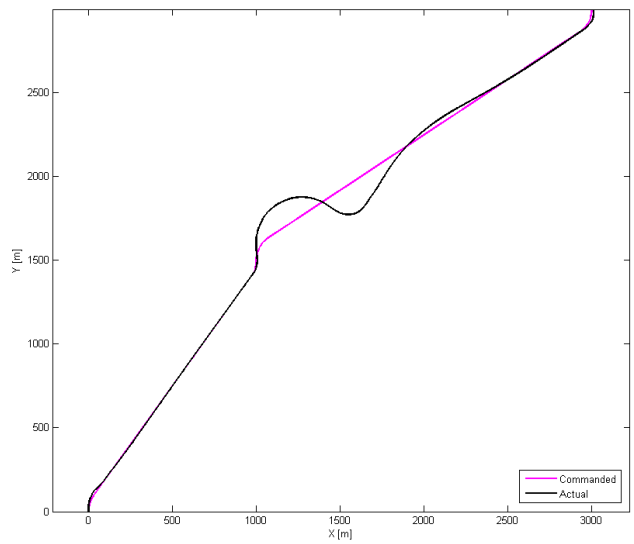

Figure C-28—Dubins Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for PPID Controller, 2D View

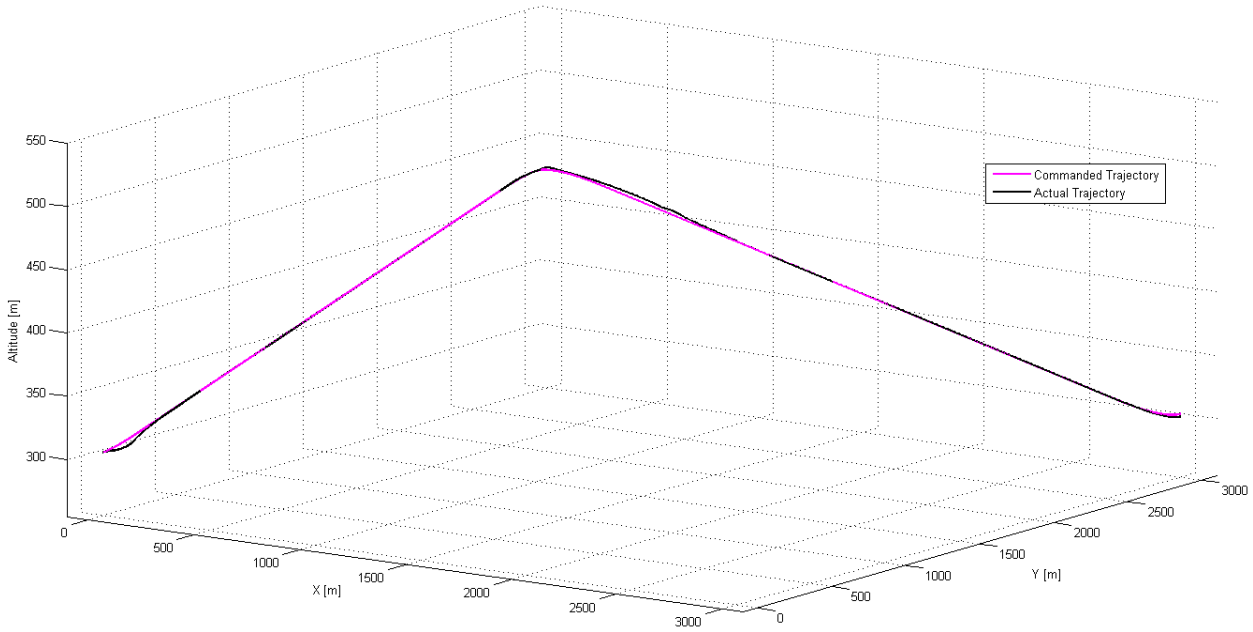

Figure C-29—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 3D View

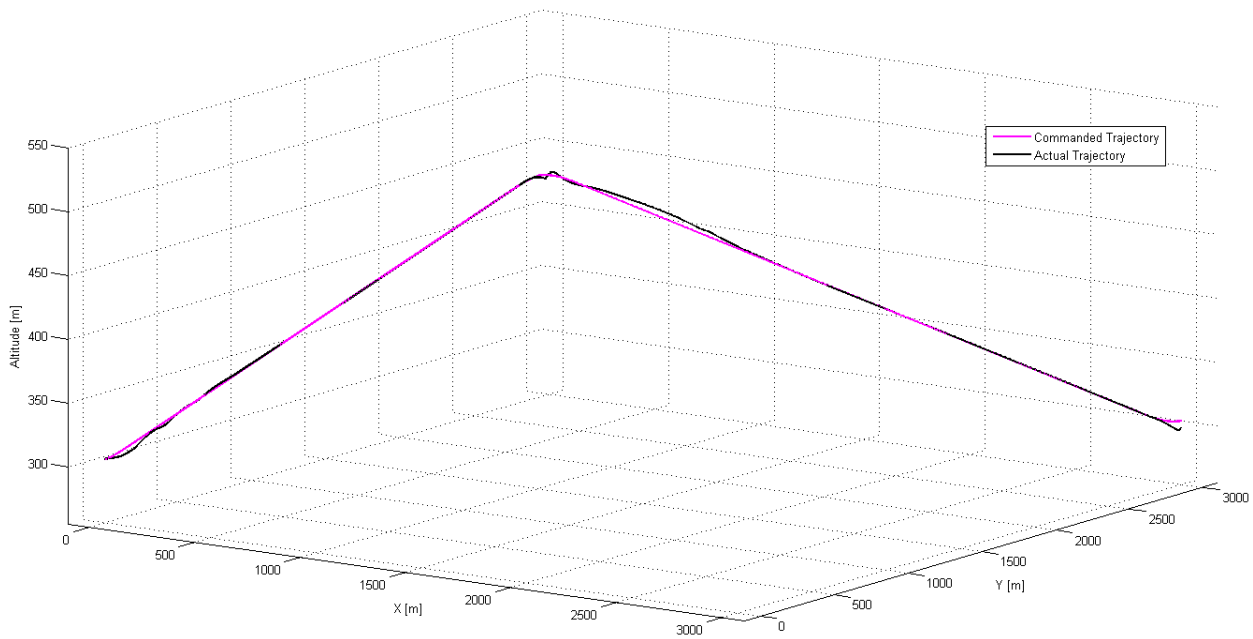

Figure C-30—Dubins Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 3D View

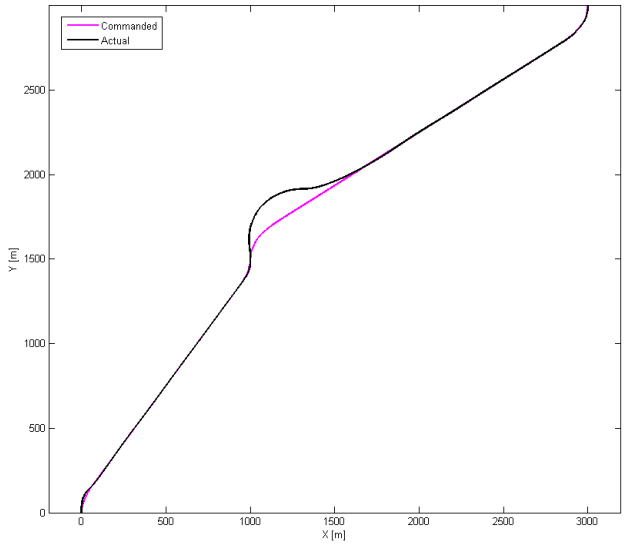

Figure C-31—Clothoid Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 2D View

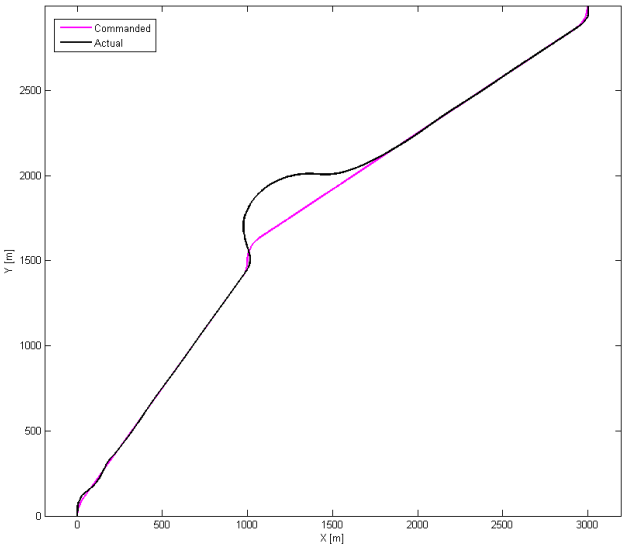

Figure C-32—Dubins Trajectory Tracking Results for Basic 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 2D View

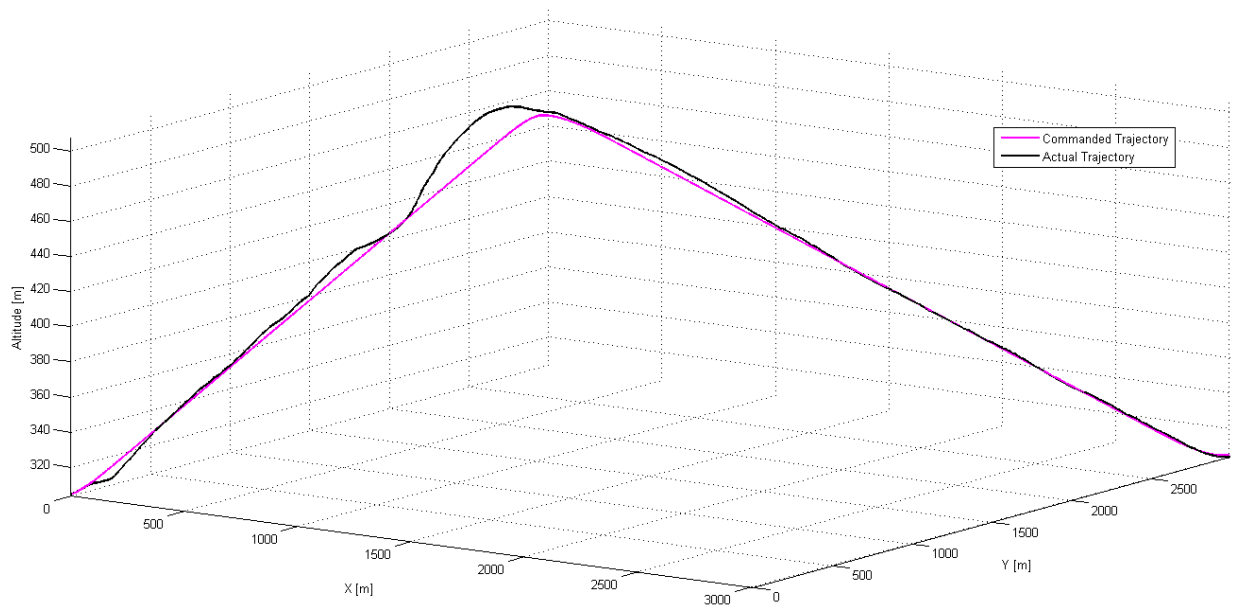

Figure C-33—Clothoid Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for PPID Controller, 3D View

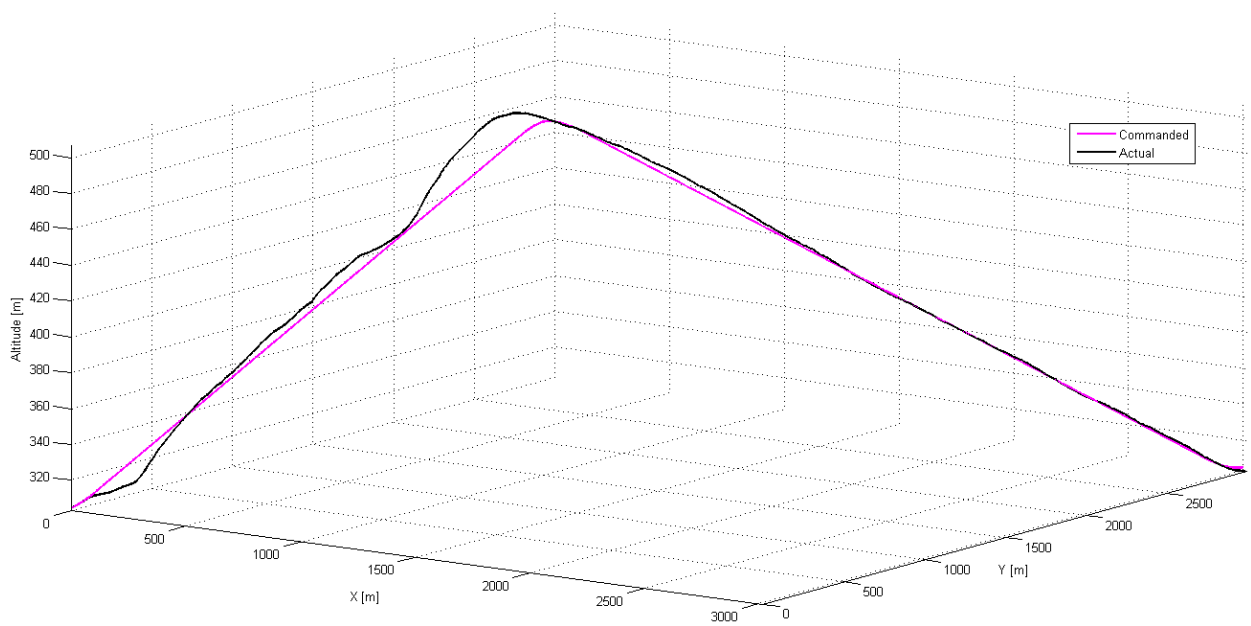

Figure C-34—Dubins Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for PPID Controller, 3D View

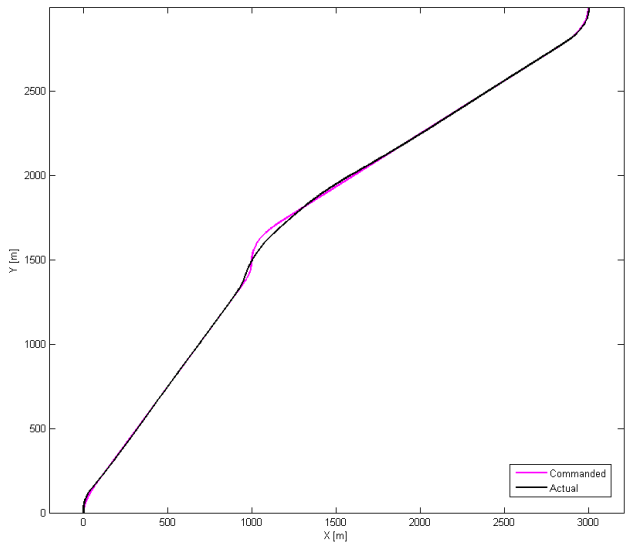

Figure C-35—Clothoid Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for PPID Controller, 2D View

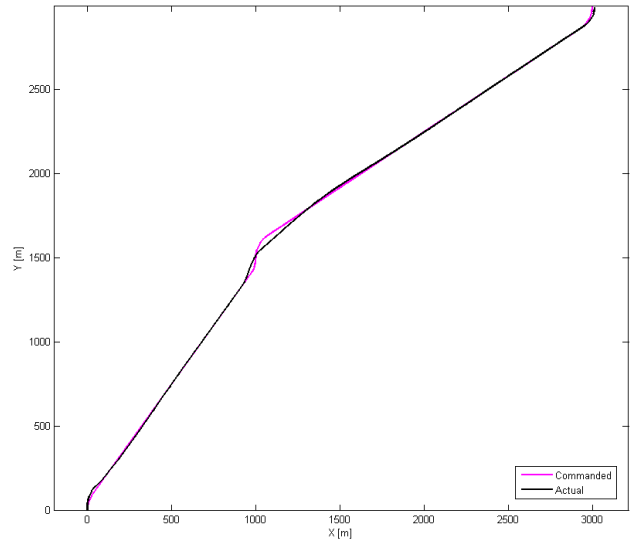

Figure C-36—Dubins Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for PPID Controller, 2D View

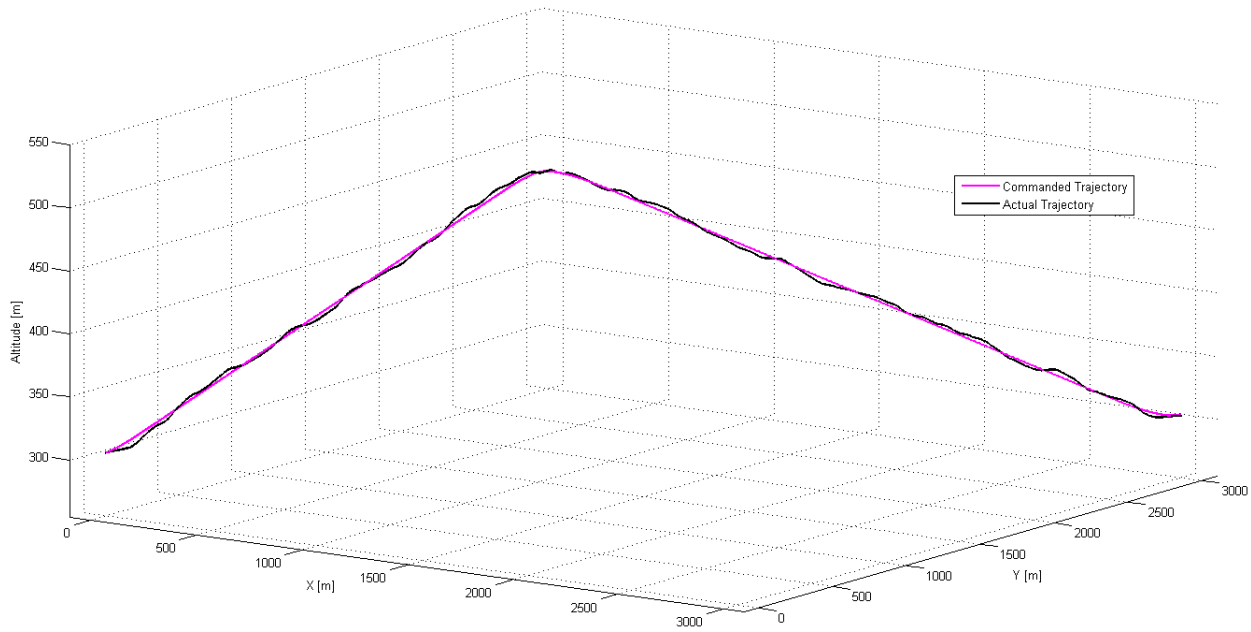

Figure C-37—Clothoid Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for ONLDI Controller, 3D View

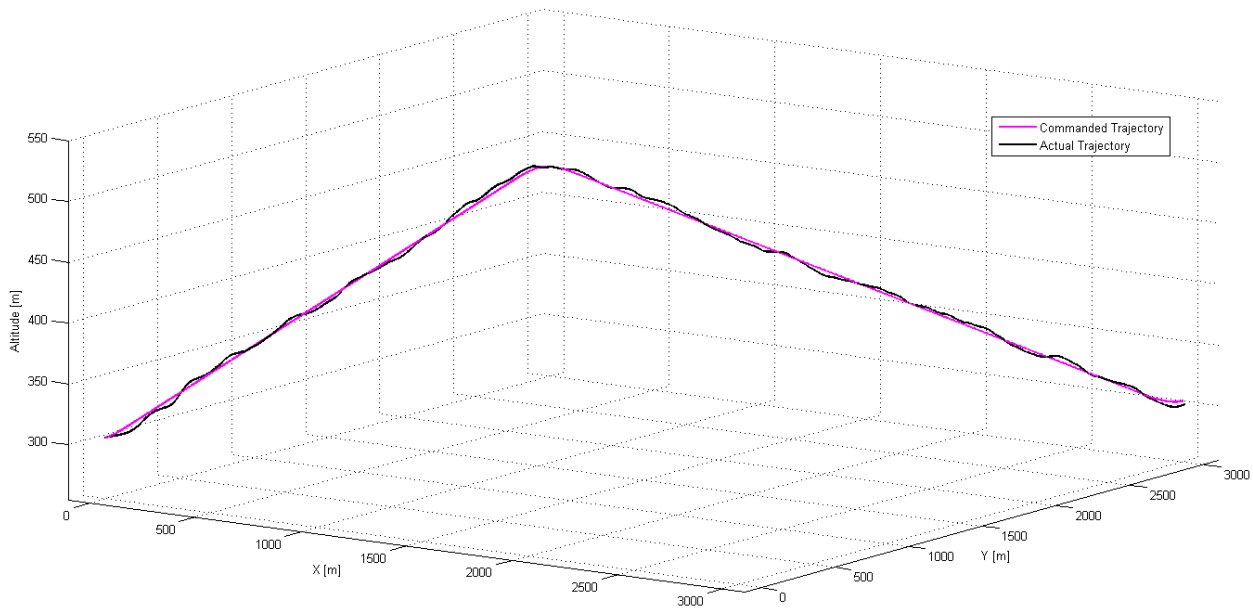

Figure C-38—Dubins Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for ONLDI Controller, 3D View

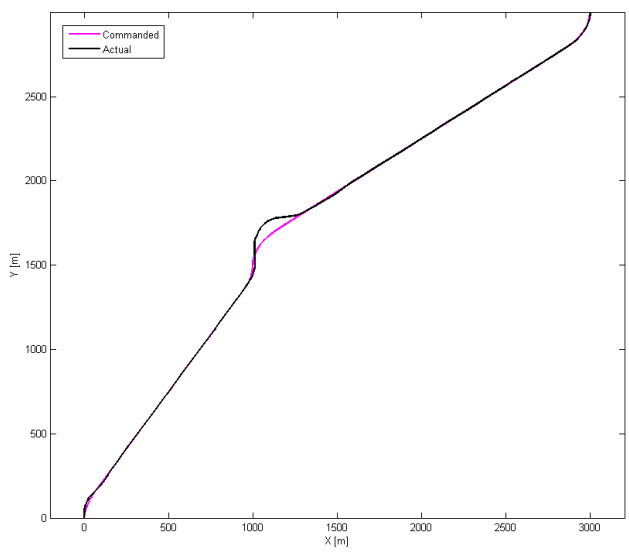

Figure C-39—Clothoid Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for ONLDI Controller, 2D View

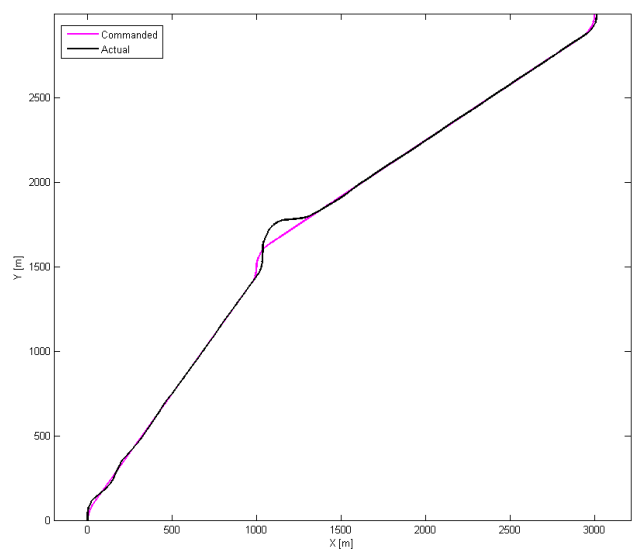

Figure C-40—Dubins Trajectory Tracking Results for Basic 3D Path with High Wind Turbulence for ONLDI Controller, 2D View

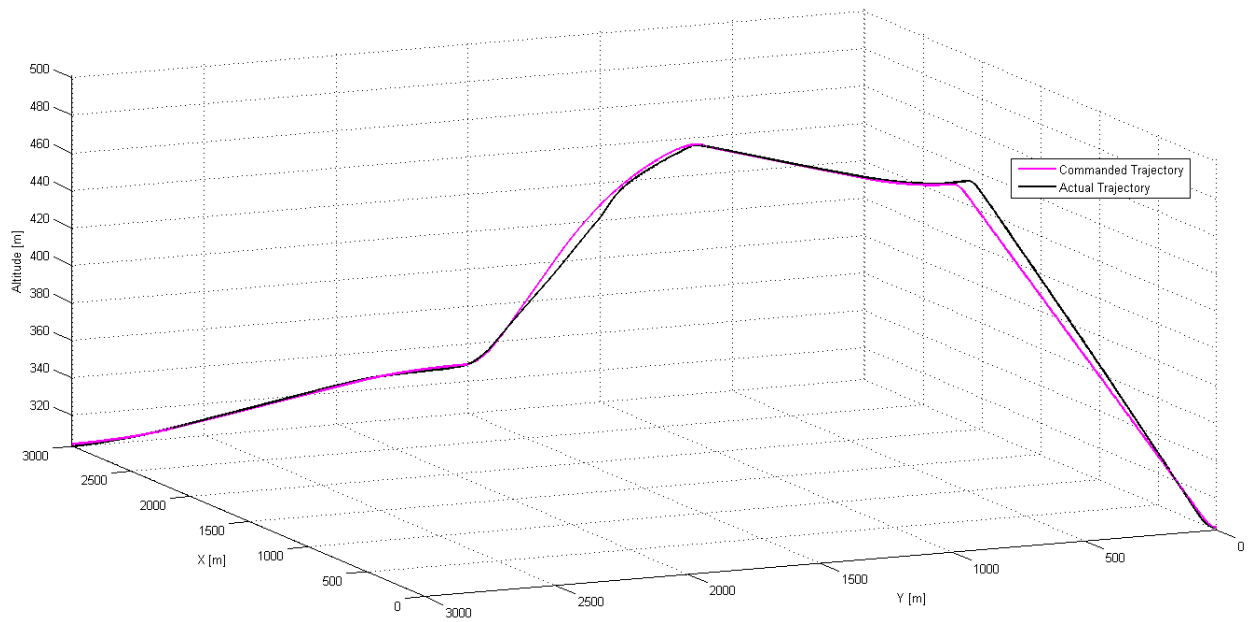

Figure C-41—Clothoid Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for PPID Controller, 3D View

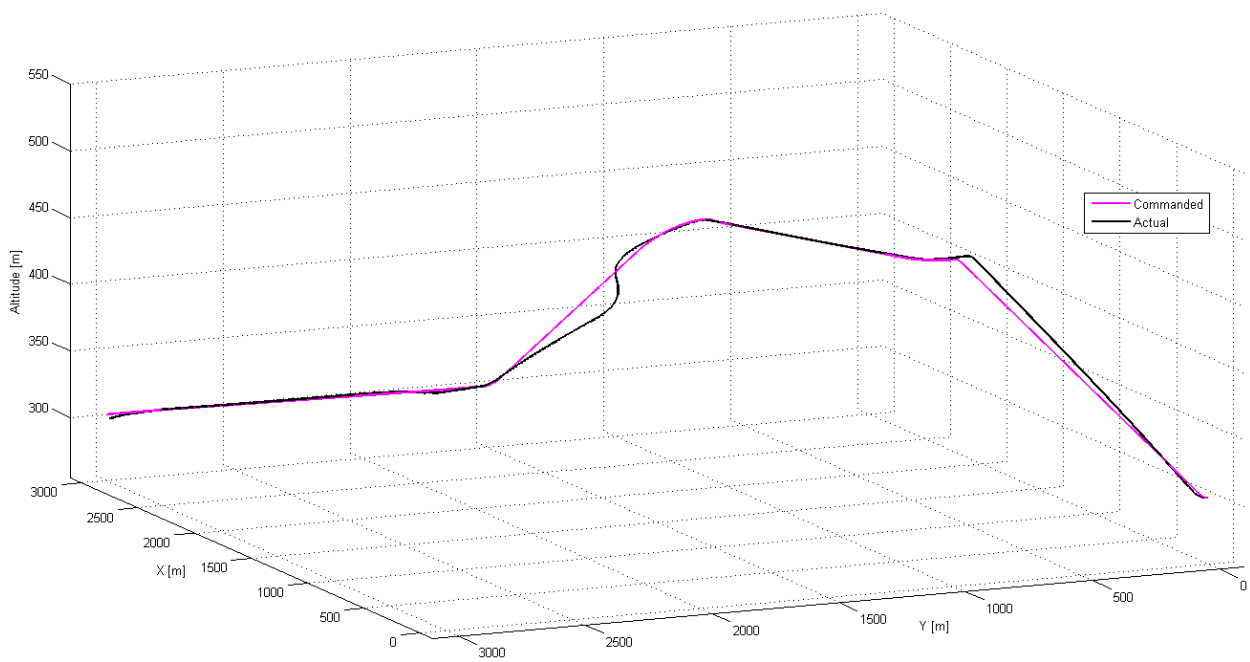

Figure C-42—Dubins Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for PPID Controller, 3D View

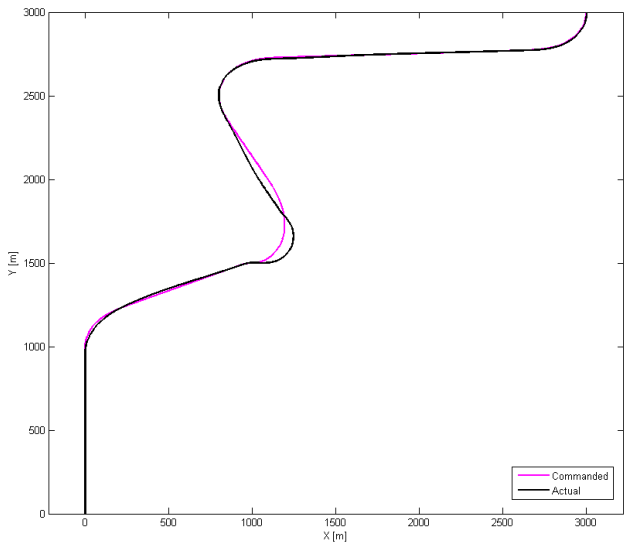

Figure C-43—Clothoid Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for PPID Controller, 2D View

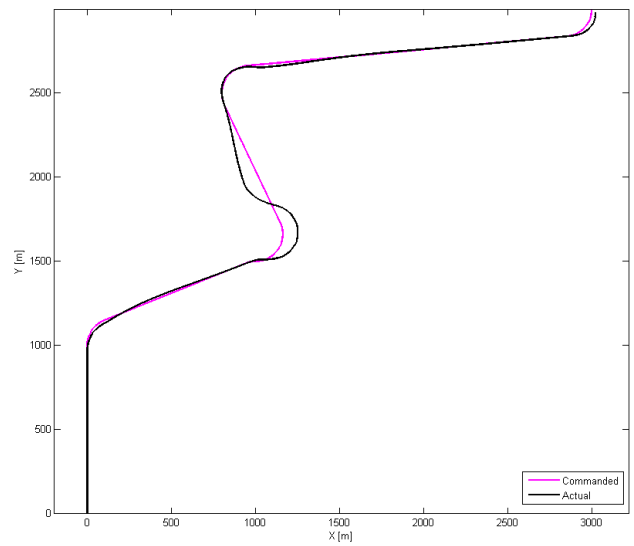

Figure C-44—Dubins Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for PPID Controller, 2D View

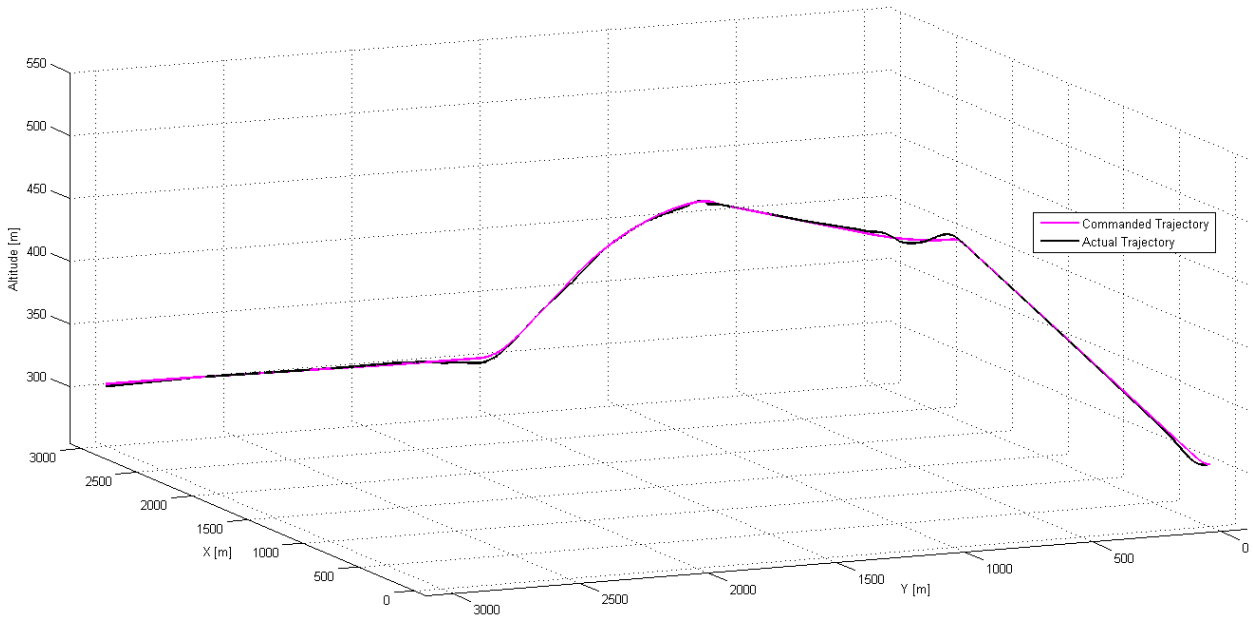

Figure C-45—Clothoid Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for ONLDI Controller, 3D View

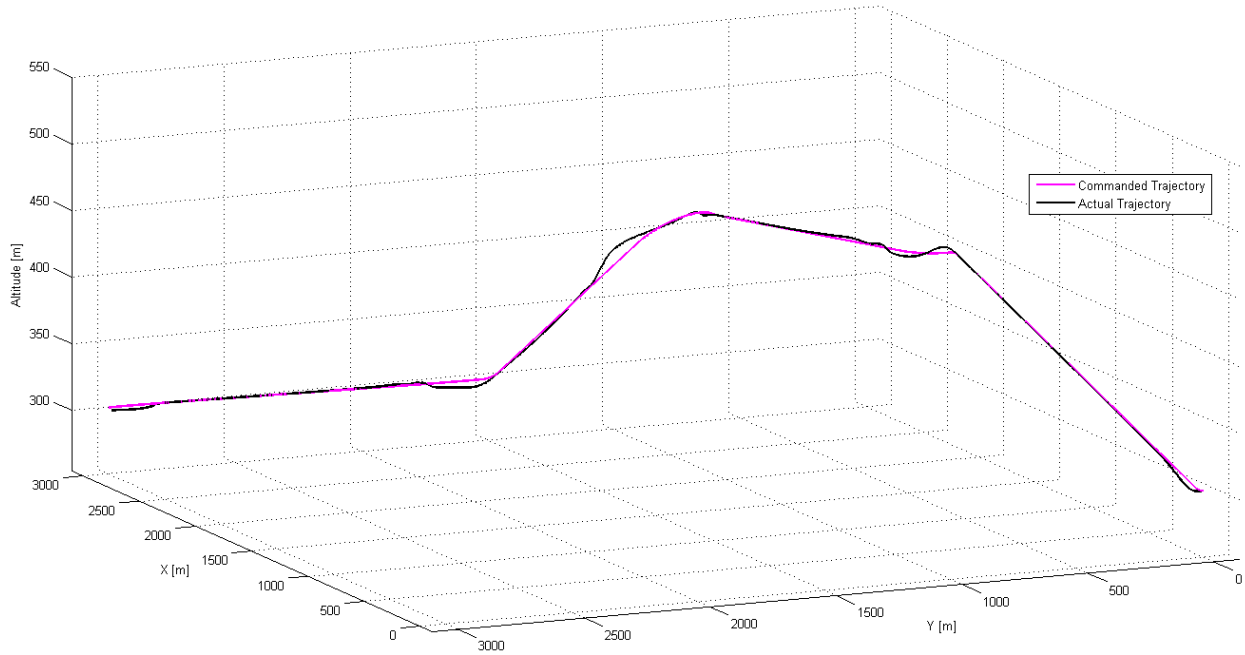

Figure C-46—Dubins Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for ONLDI Controller, 3D View

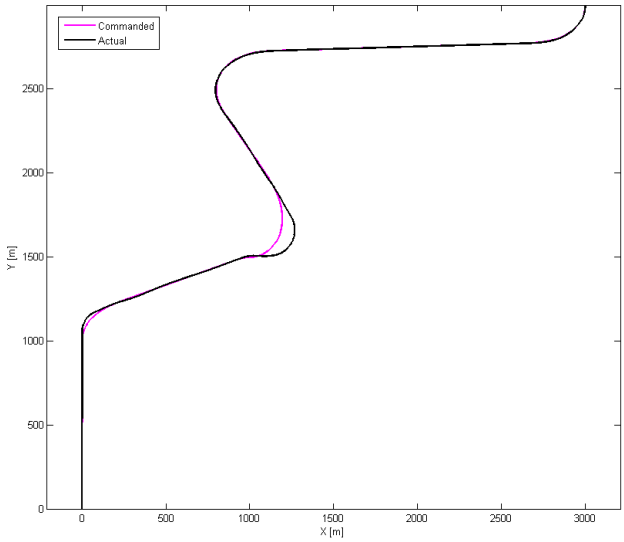

Figure C-47—Clothoid Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for ONLDI Controller, 2D View

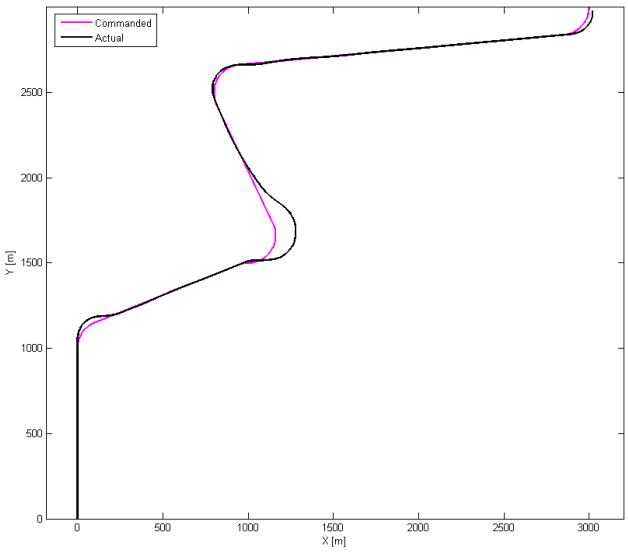

Figure C-48—Dubins Trajectory Tracking Results for Advanced 3D Path at Nominal Conditions for ONLDI Controller, 2D View

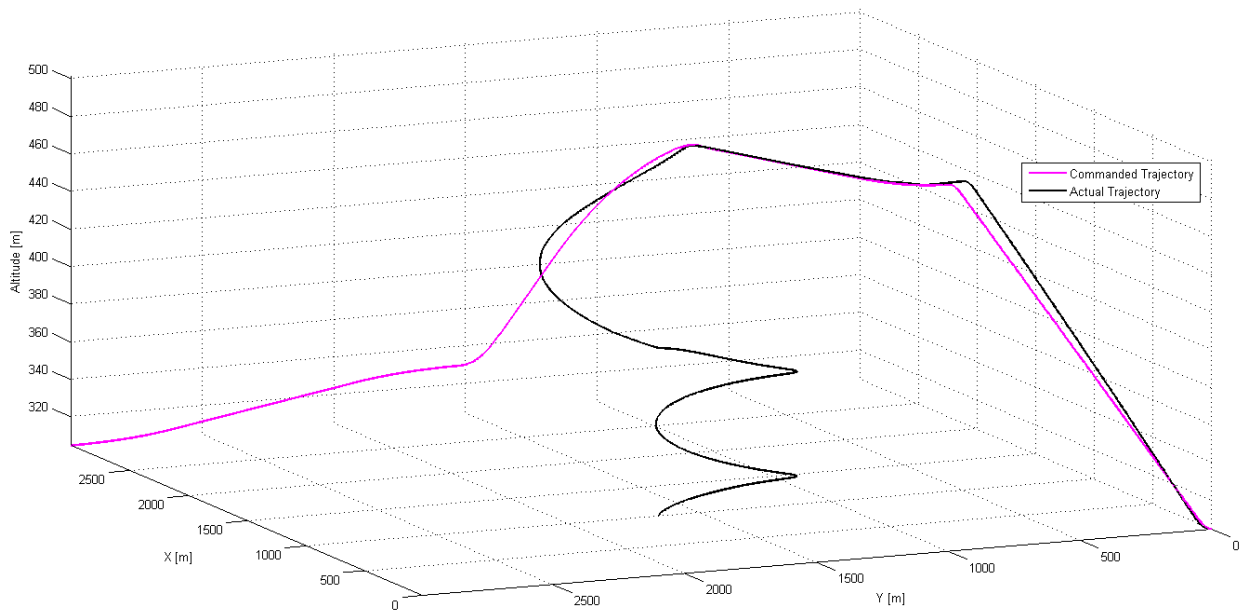

Figure C-49—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for PPID Controller, 3D View

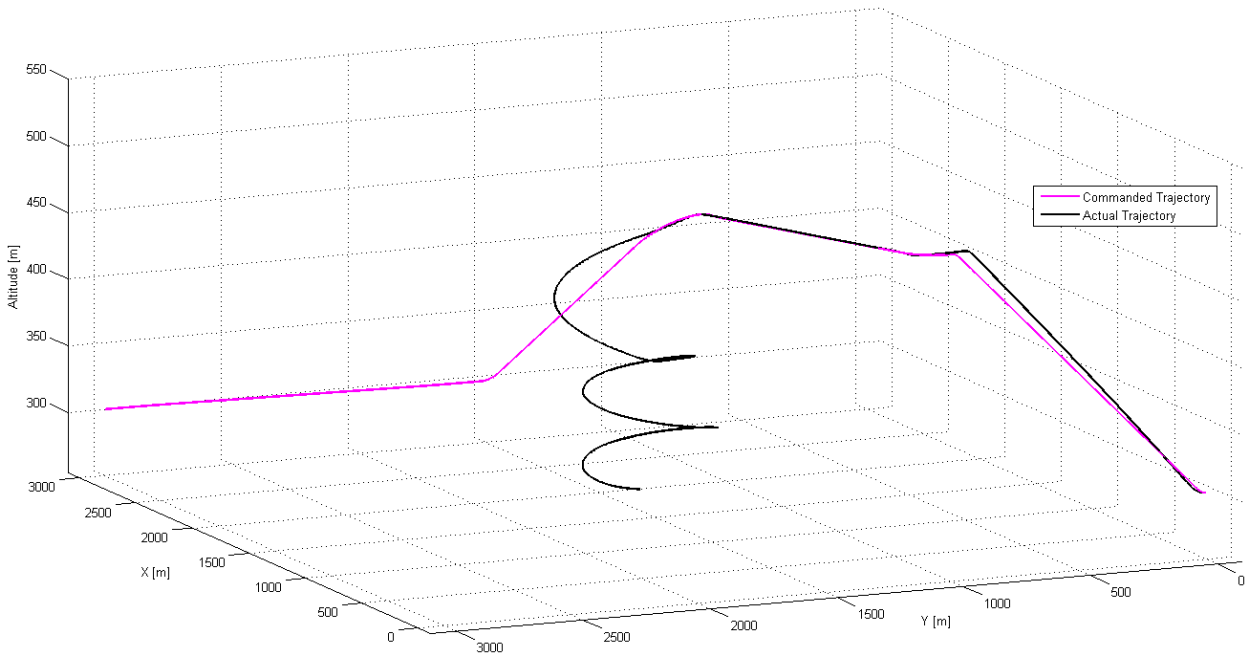

Figure C-50—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for PPID Controller, 3D View

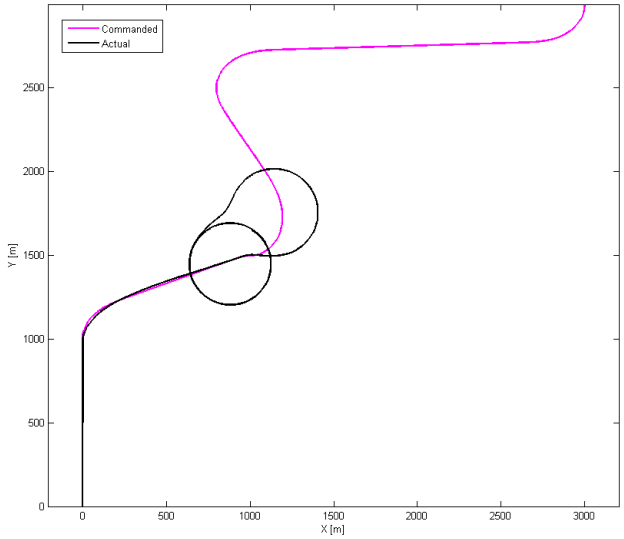

Figure C-51—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for PPID Controller, 2D View

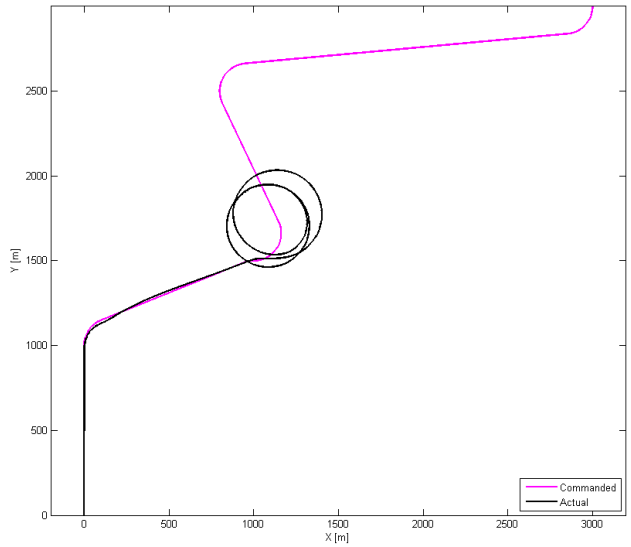

Figure C-52—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for PPID Controller, 2D View

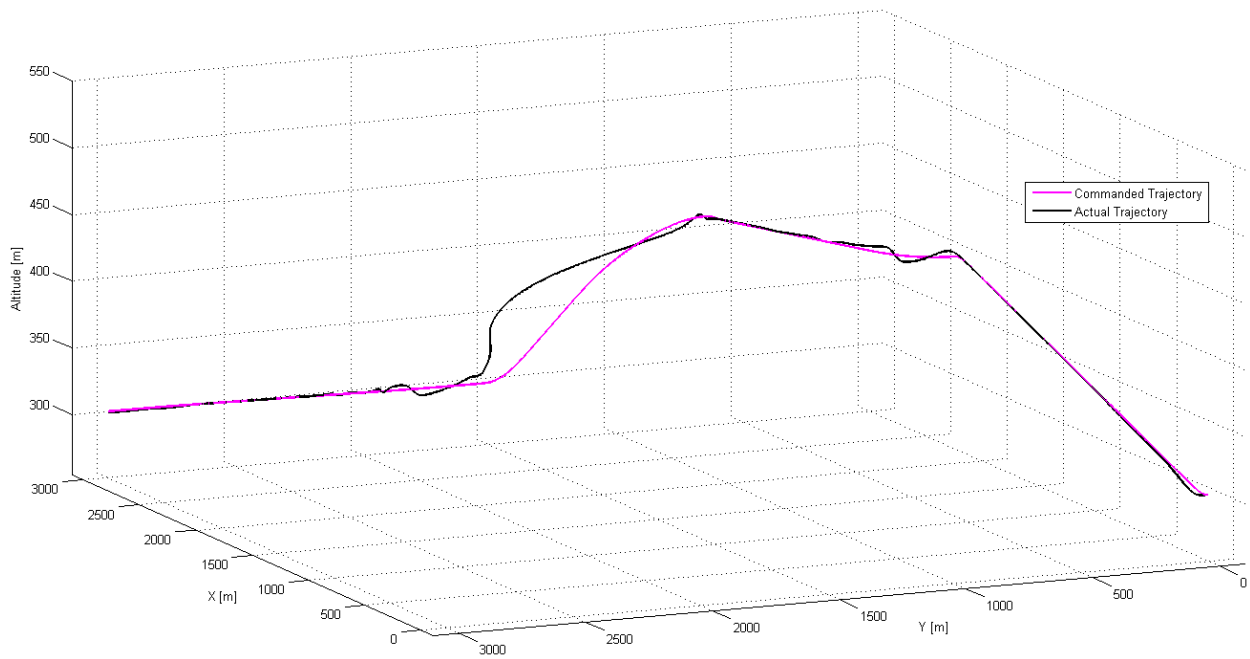

Figure C-53—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 3D View

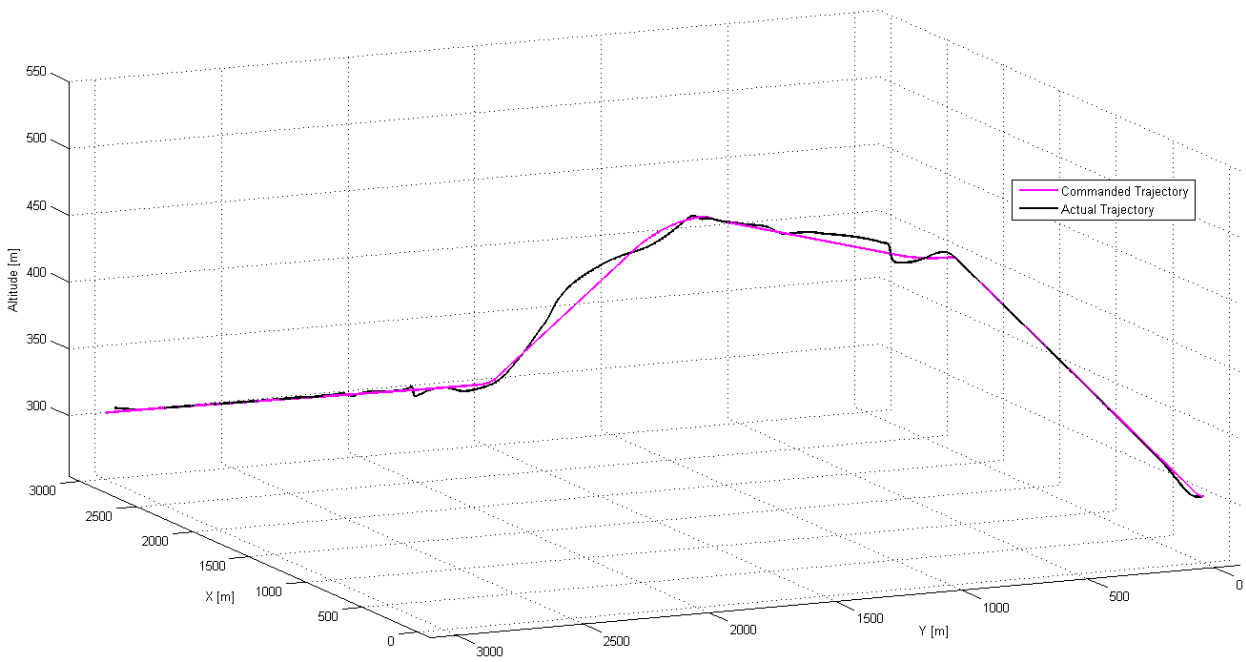

Figure C-54—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 3D View

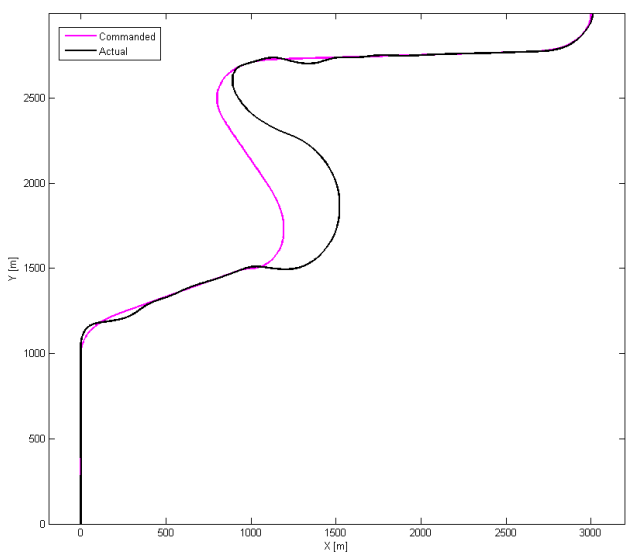

Figure C-55—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 2D View

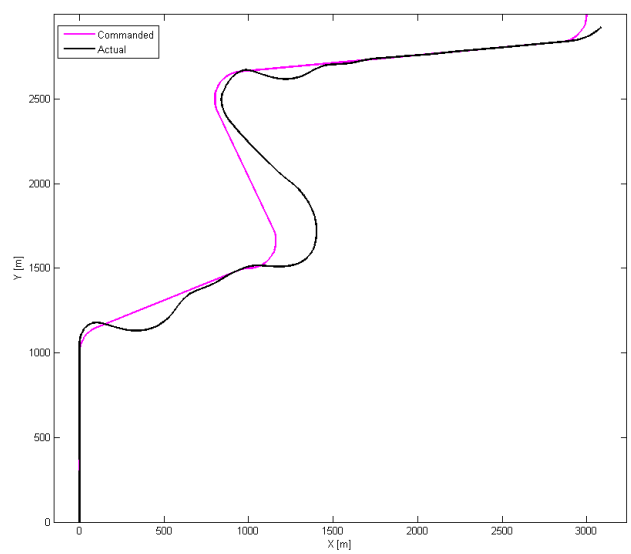

Figure C-56—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 2D View

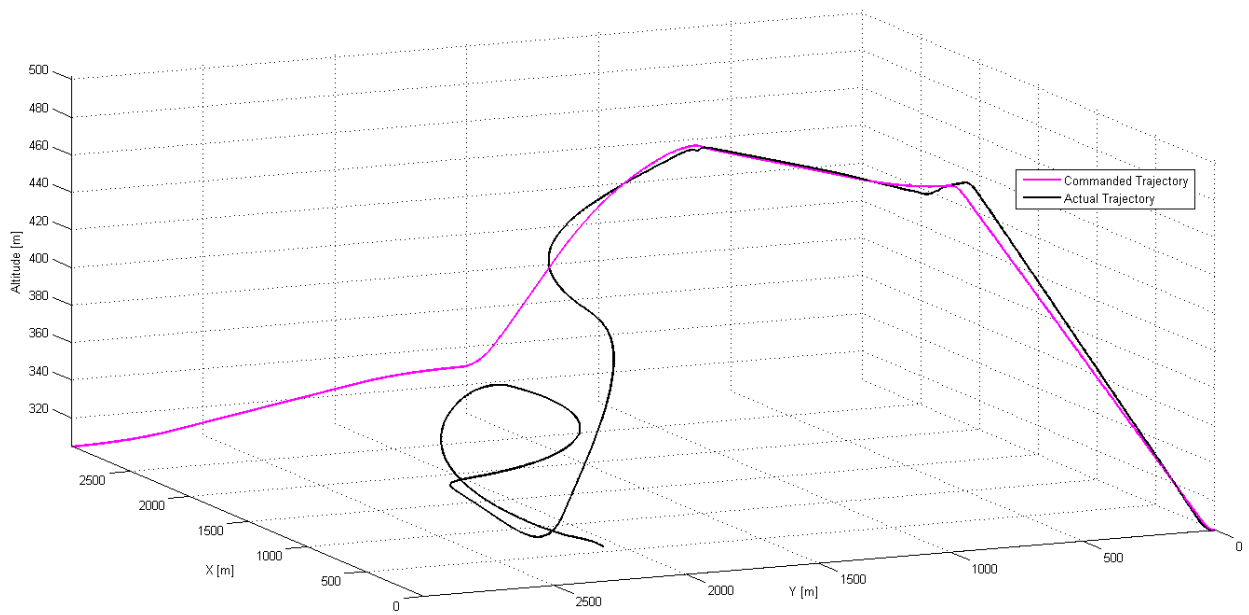

Figure C-57—Clothoid Trajectory Tracking Results for Advanced 3D with Left Elevator Stuck at 2° for PPID Controller, 3D View

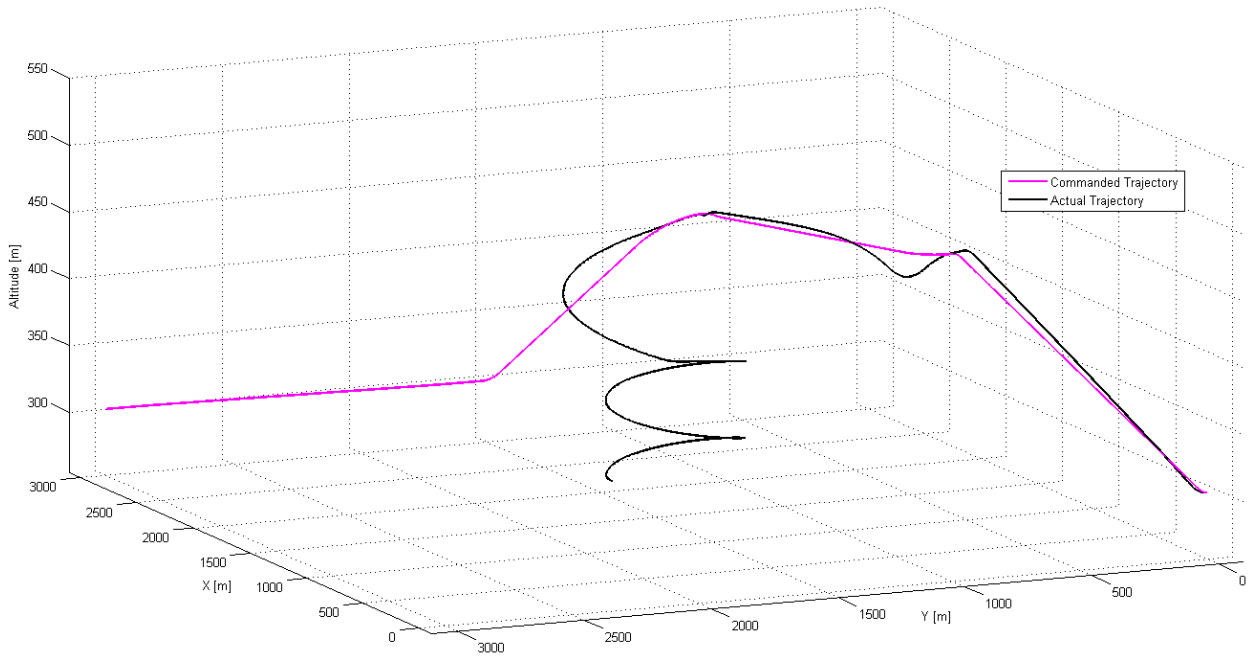

Figure C-58—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Elevator Stuck at 2° for PPID Controller, 3D View

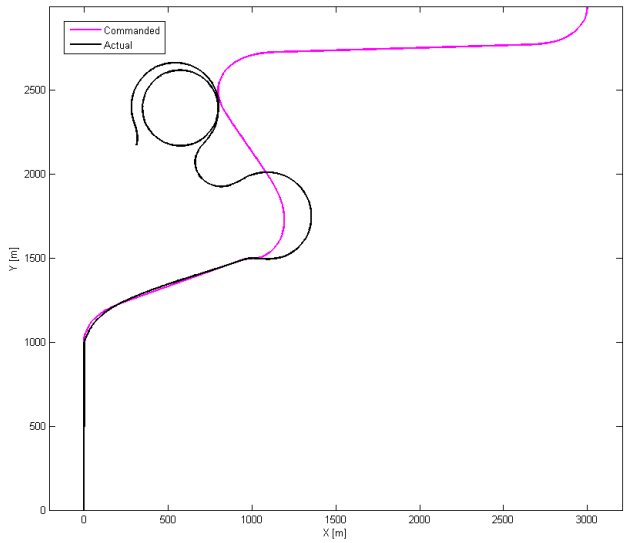

Figure C-59—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Elevator Stuck at 2° for PPID Controller, 2D View

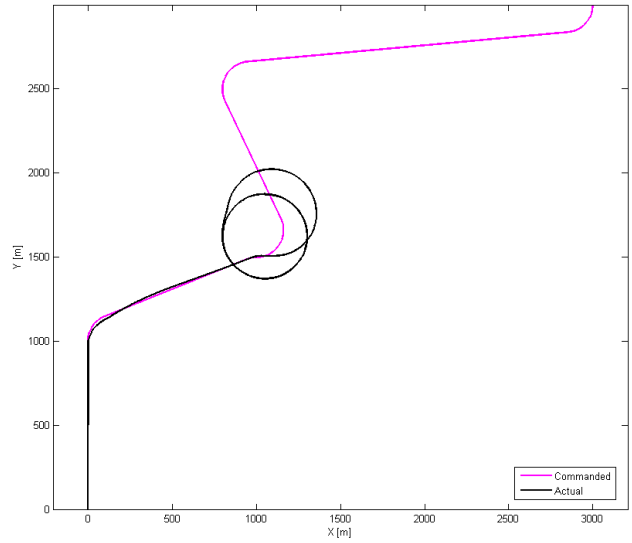

Figure C-60—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Elevator Stuck at 2° for PPID Controller, 2D View

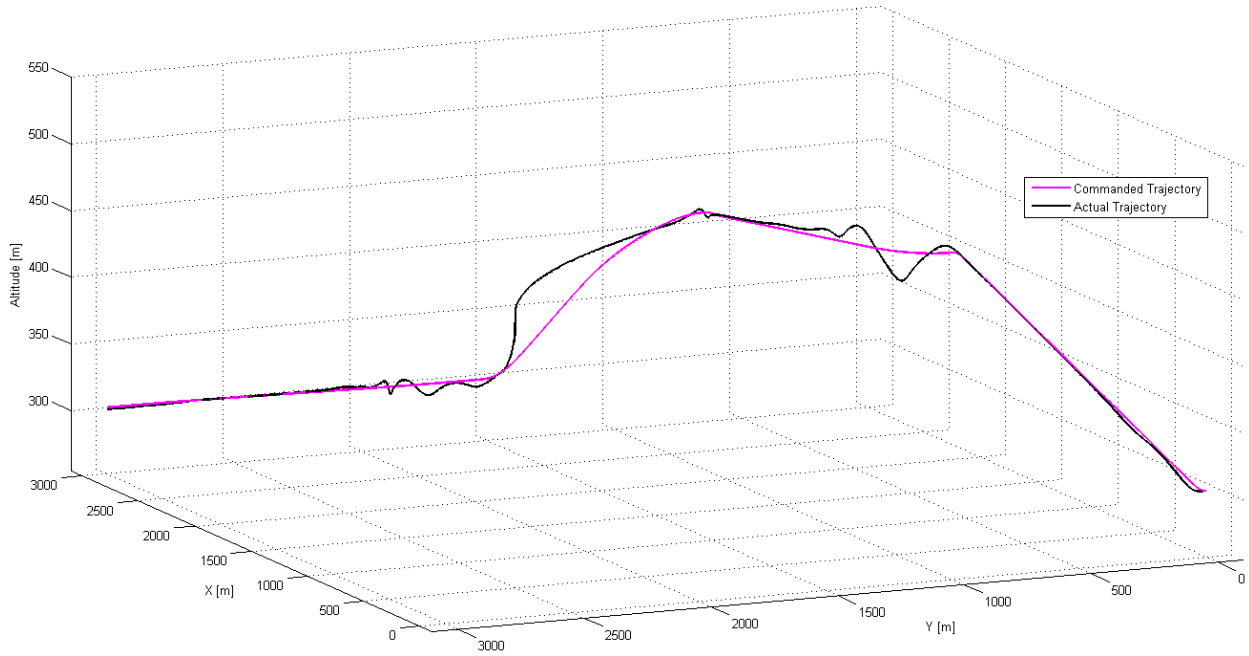

Figure C-61—Clothoid Trajectory Tracking Results for Advanced 3D with Left Elevator Stuck at 2° for ONLDI Controller, 3D View

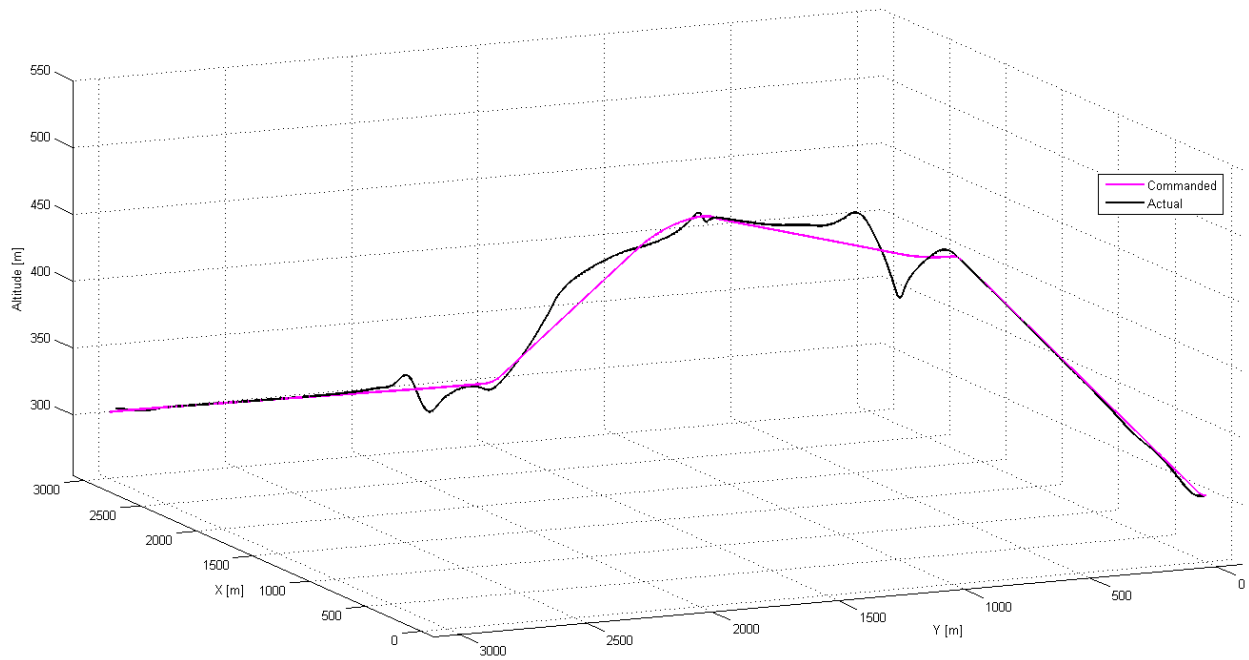

Figure C-62—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 3D View

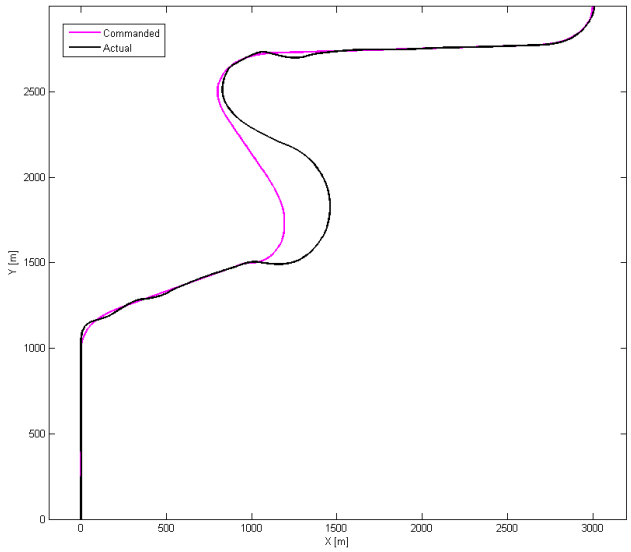

Figure C-63—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 2D View

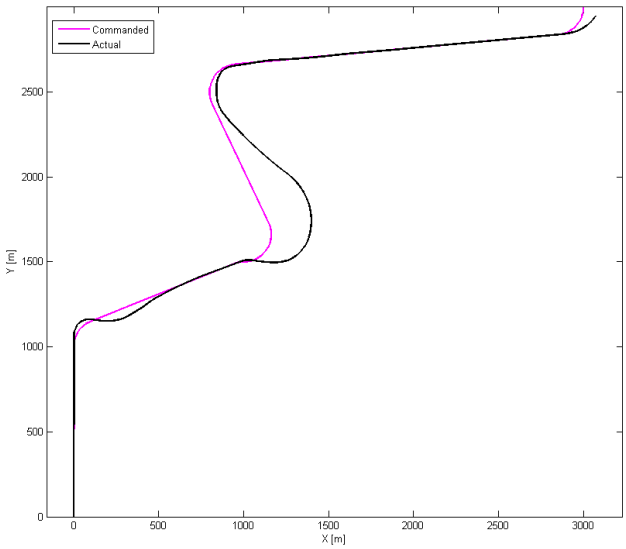

Figure C-64—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 2D View

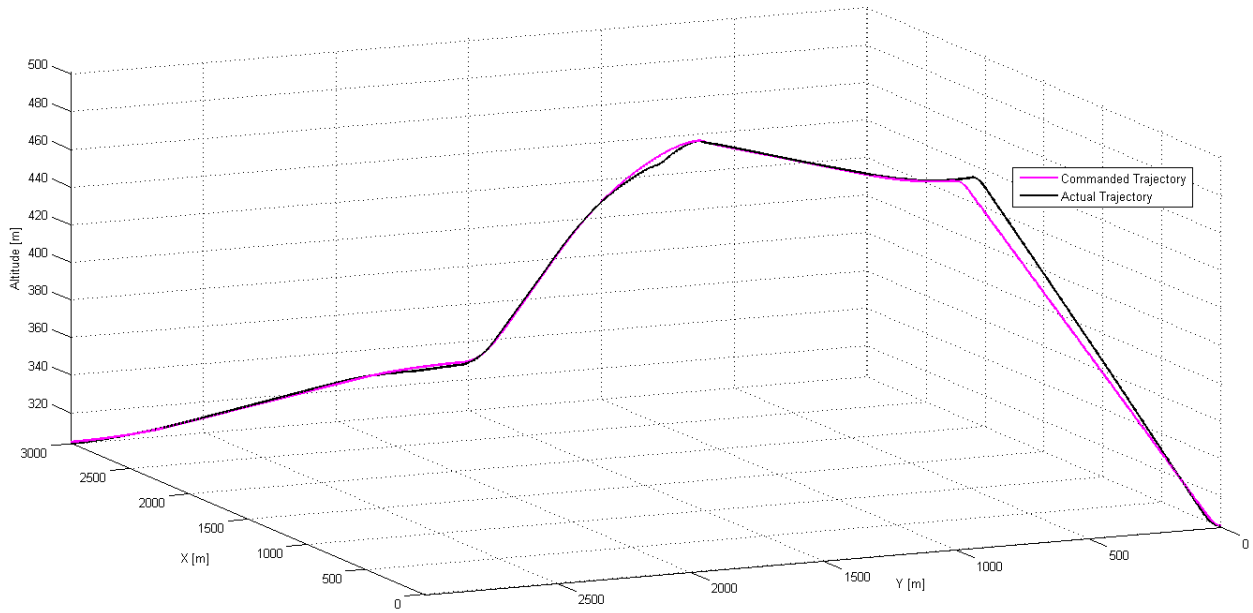

Figure C-65—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for PPID Controller, 3D View

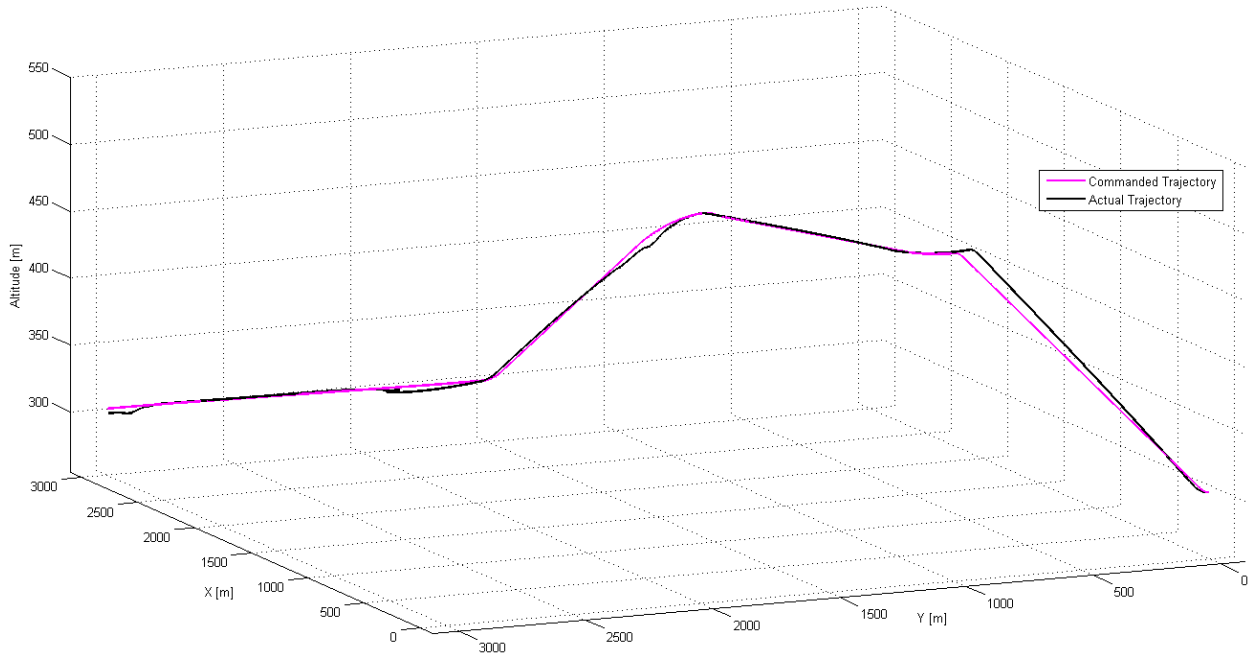

Figure C-66—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for PPID Controller, 3D View

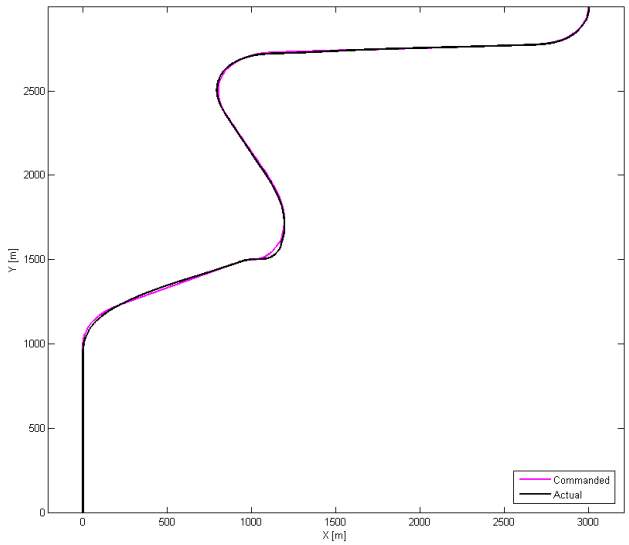

Figure C-67—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for PPID Controller, 2D View

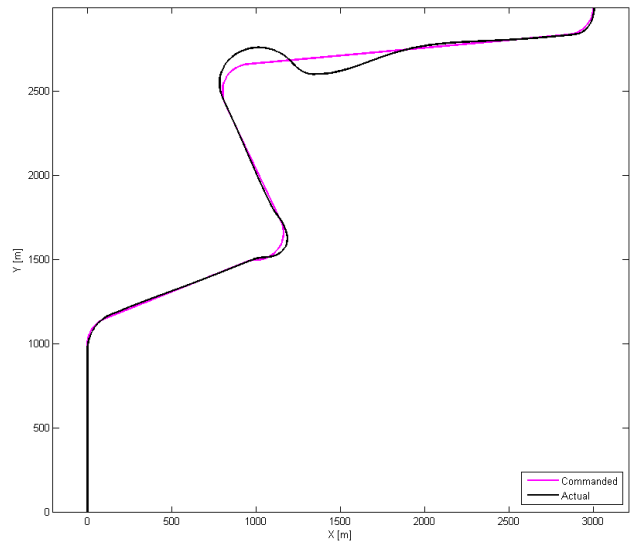

Figure C-68—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for PPID Controller, 2D View

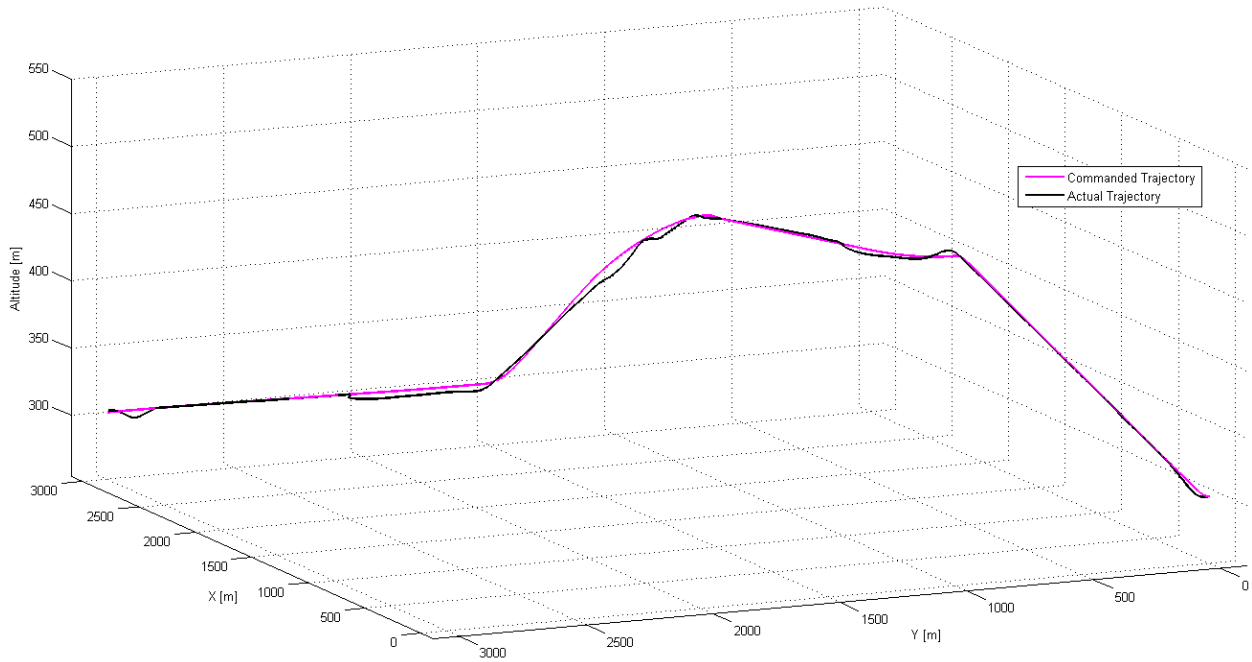

Figure C-69—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 3D View

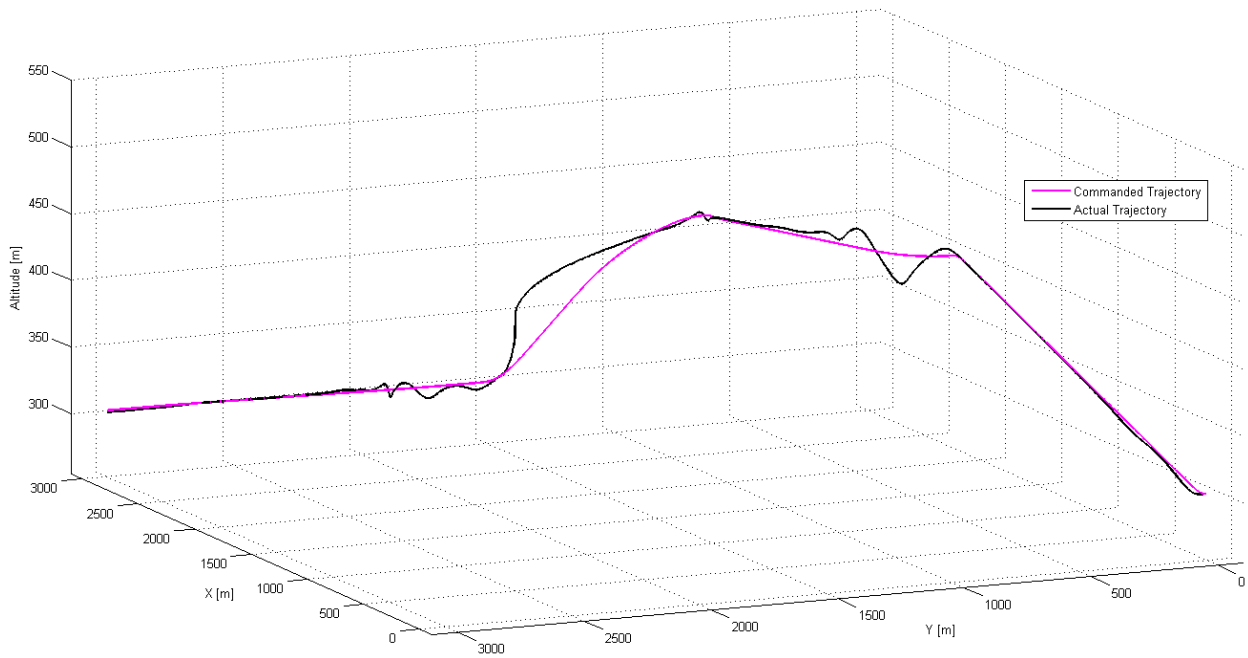

Figure C-70—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 3D View

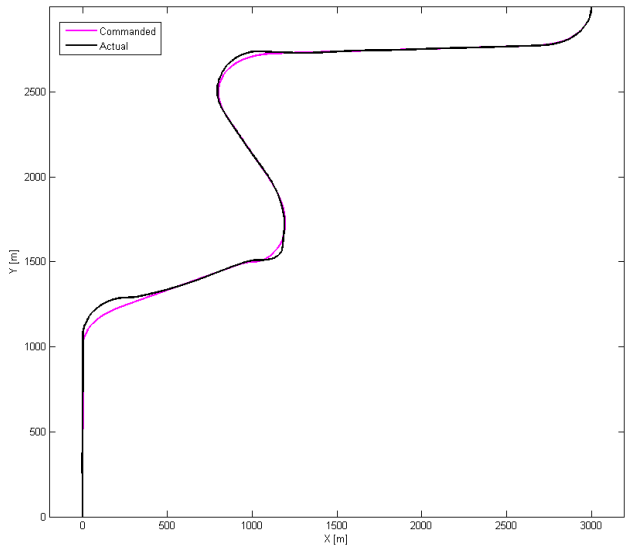

Figure C-71—Clothoid Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 2D View

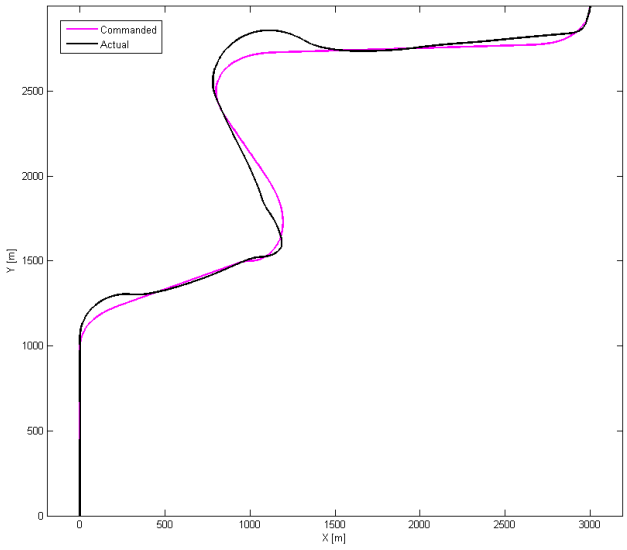

Figure C-72—Dubins Trajectory Tracking Results for Advanced 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 2D View

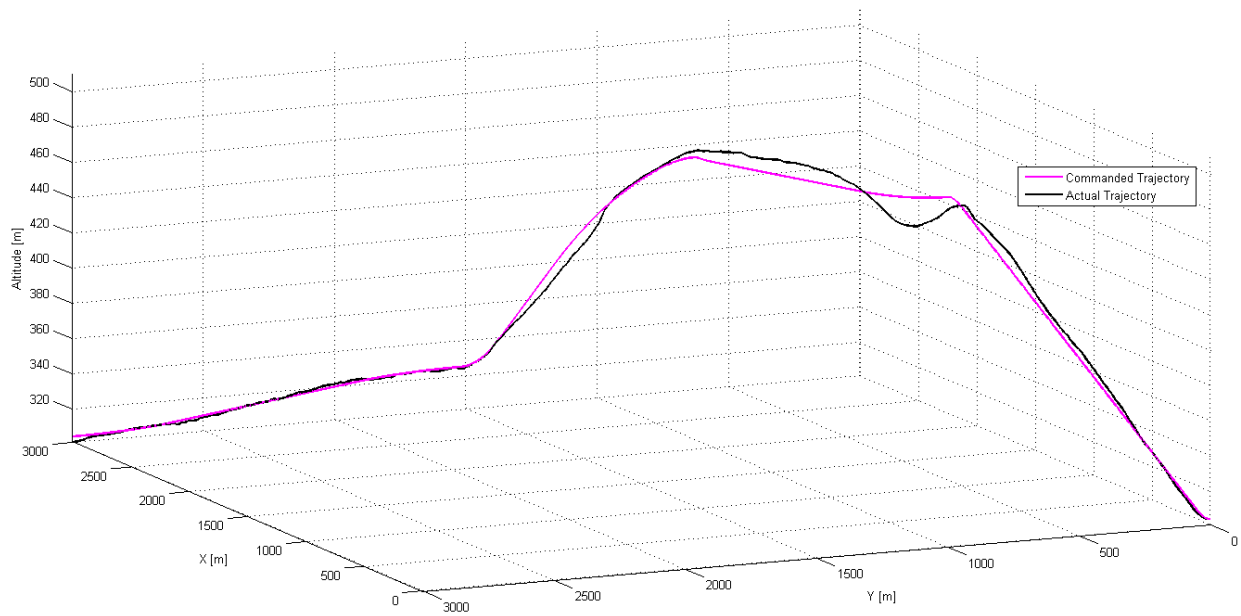

Figure C-73—Clothoid Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for PPID Controller, 3D View

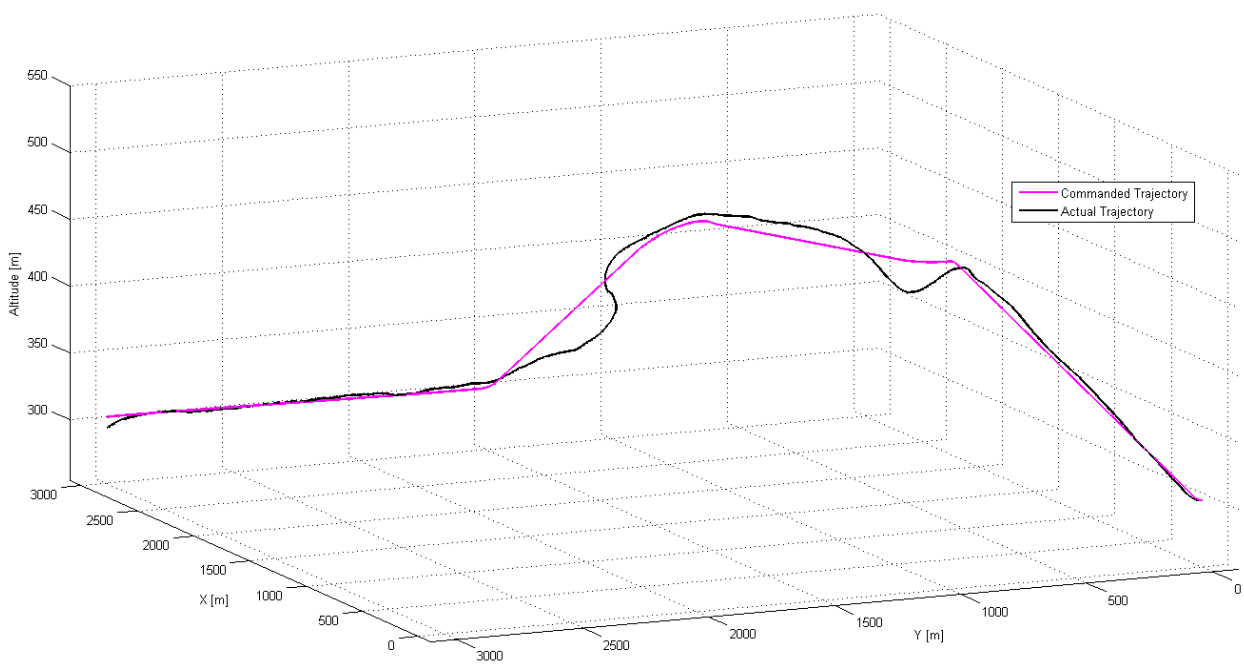

Figure C-74—Dubins Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for PPID Controller, 3D View

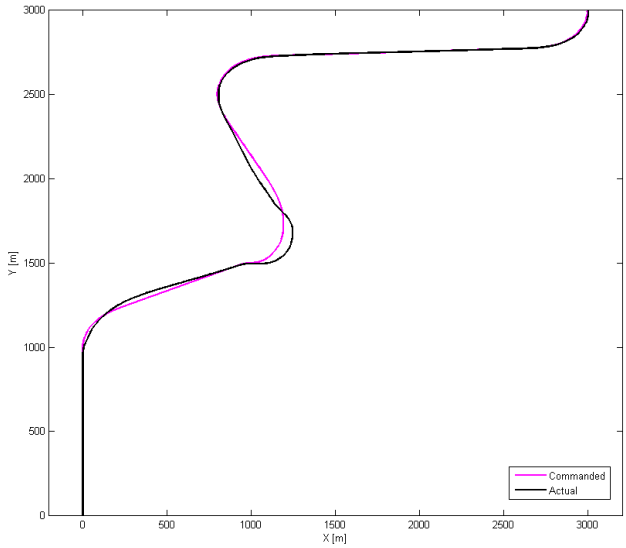

Figure C-75—Clothoid Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for PPID Controller, 2D View

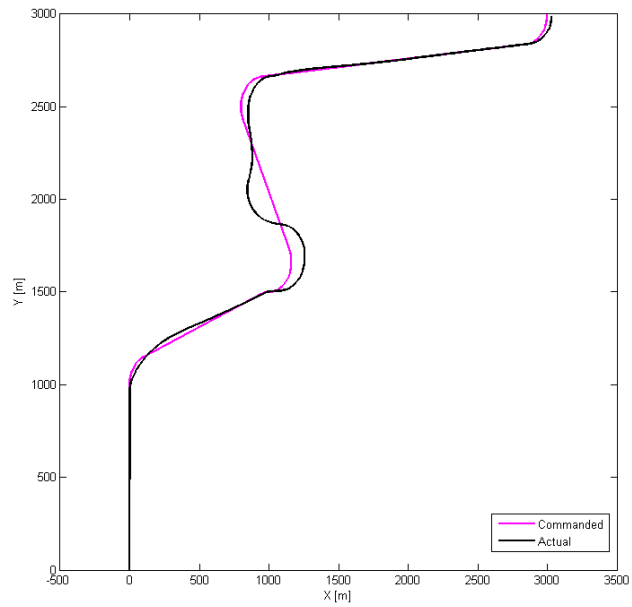

Figure C-76—Dubins Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for PPID Controller, 2D View

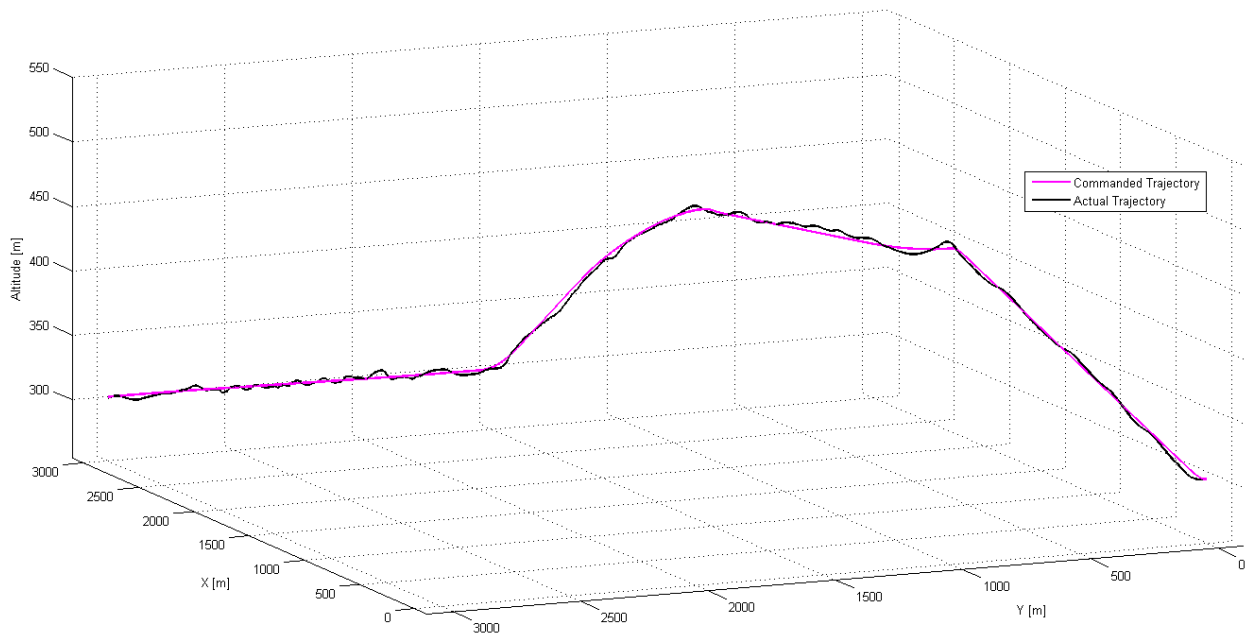

Figure C-77—Clothoid Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for ONLDI Controller, 3D View

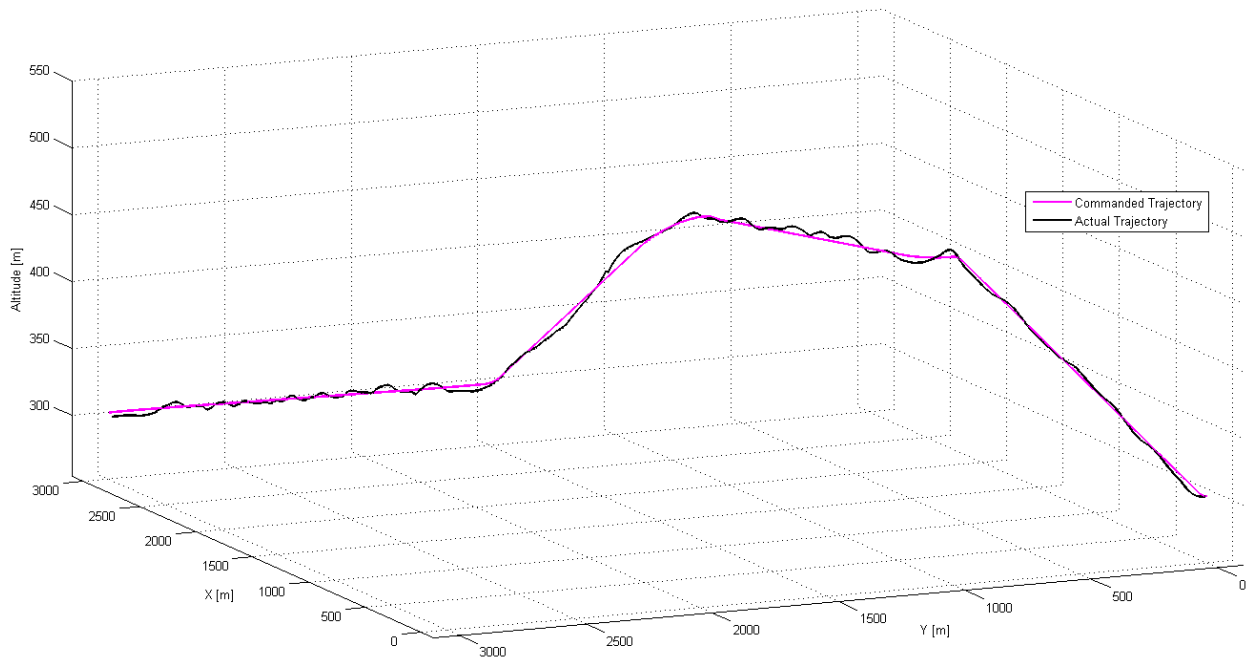

Figure C-78—Dubins Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for ONLDI Controller, 3D View

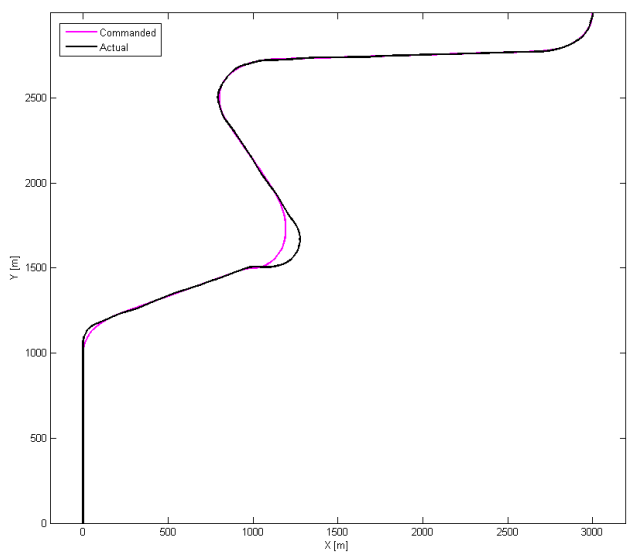

Figure C-79—Clothoid Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for ONLDI Controller, 2D View

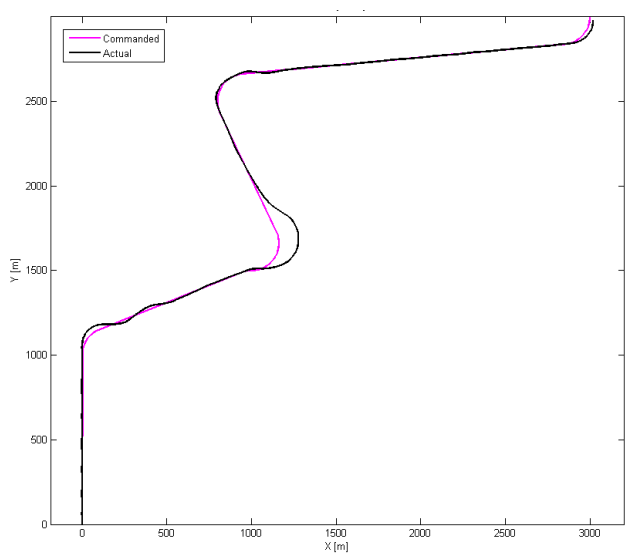

Figure C-80—Dubins Trajectory Tracking Results for Advanced 3D Path with High Wind Turbulence for ONLDI Controller, 2D View

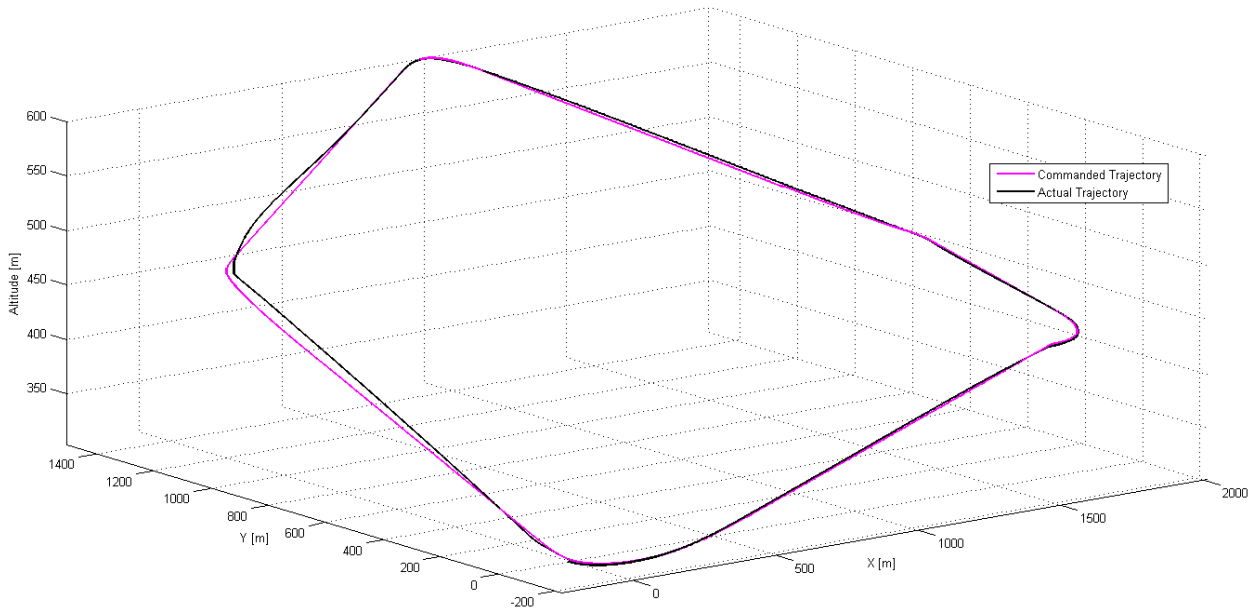

Figure C-81—Clothoid Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for PPID Controller, 3D View

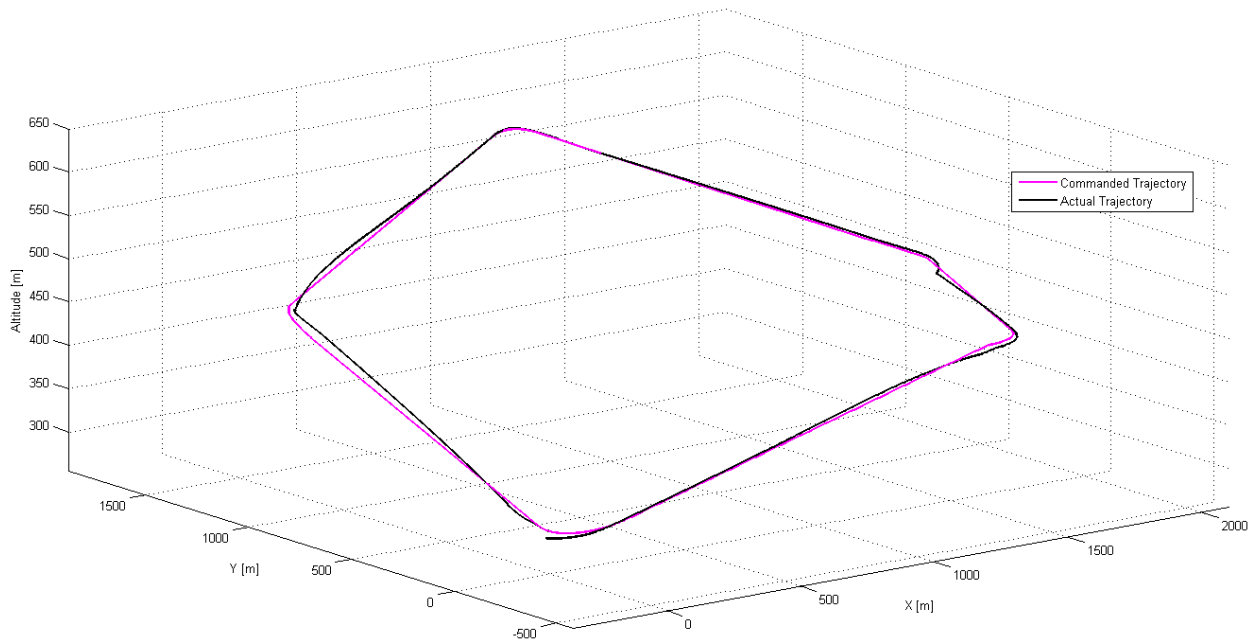

Figure C-82—Dubins Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for PPID Controller, 3D View

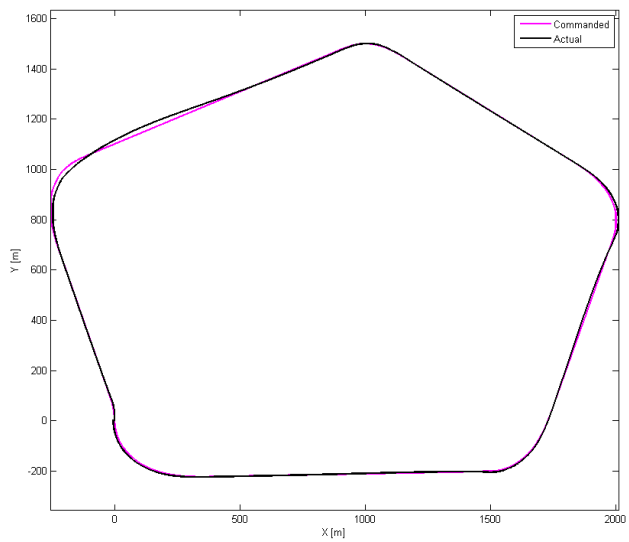

Figure C-83—Clothoid Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for PPID Controller, 2D View

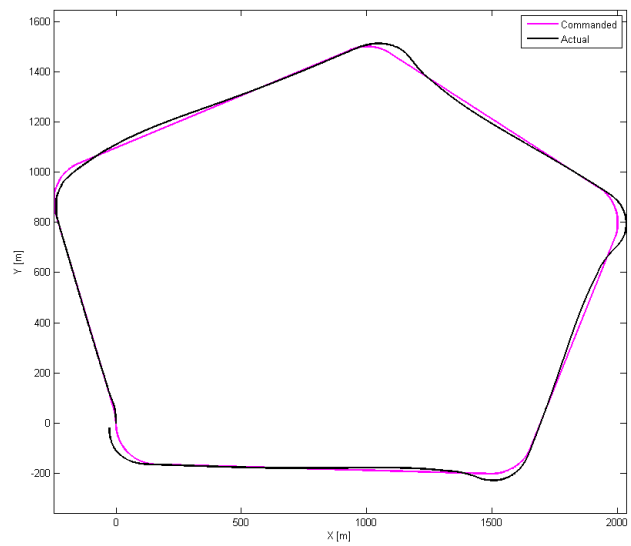

Figure C-84—Dubins Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for PPID Controller, 2D View

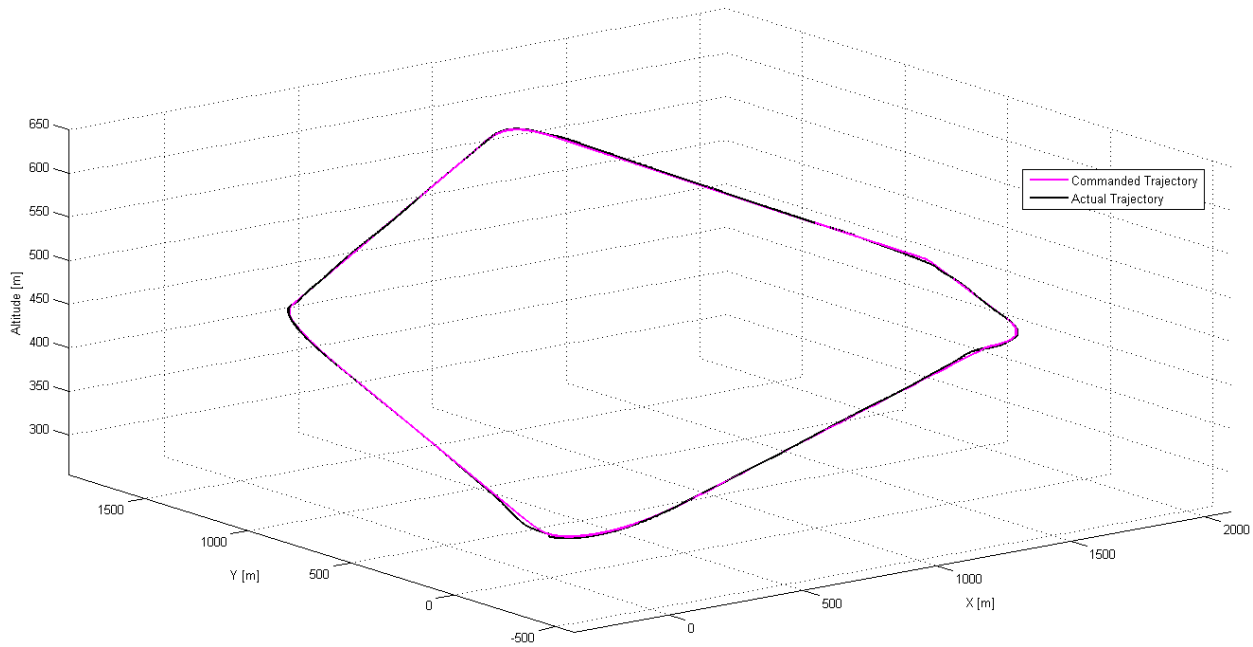

Figure C-85—Clothoid Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for ONLDI Controller, 3D View

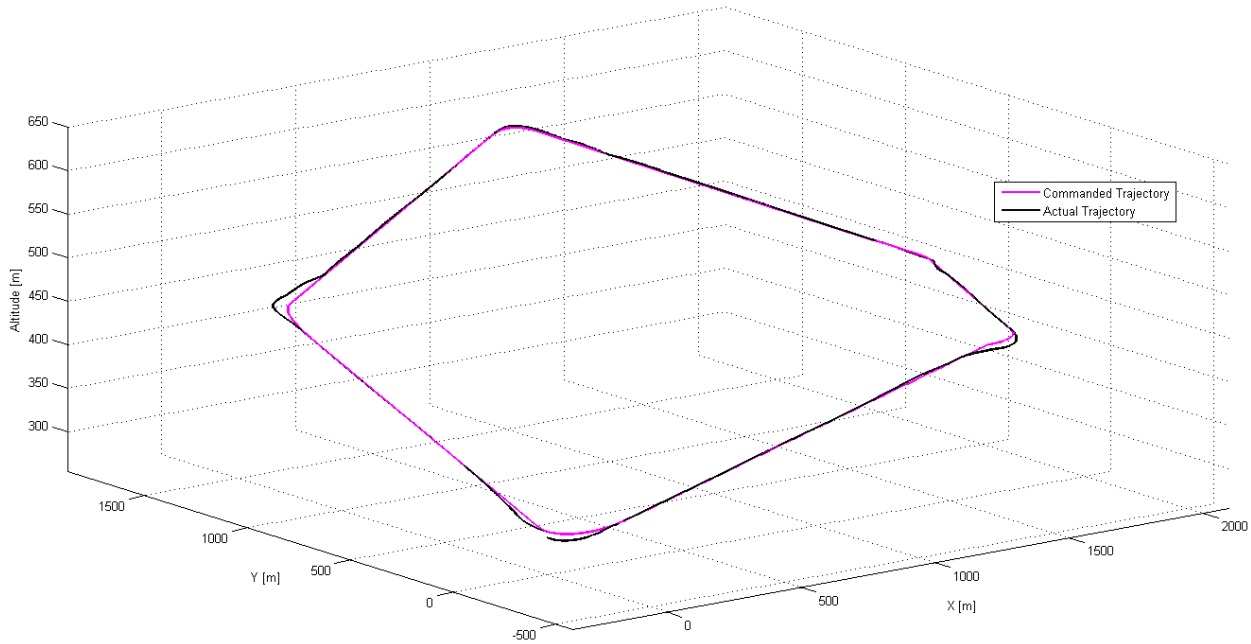

Figure C-86—Dubins Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for ONLDI Controller, 3D View

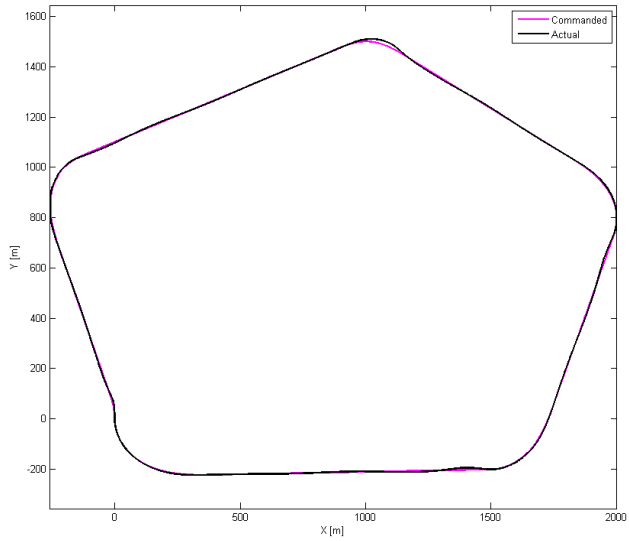

Figure C-87—Clothoid Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for ONLDI Controller, 2D View

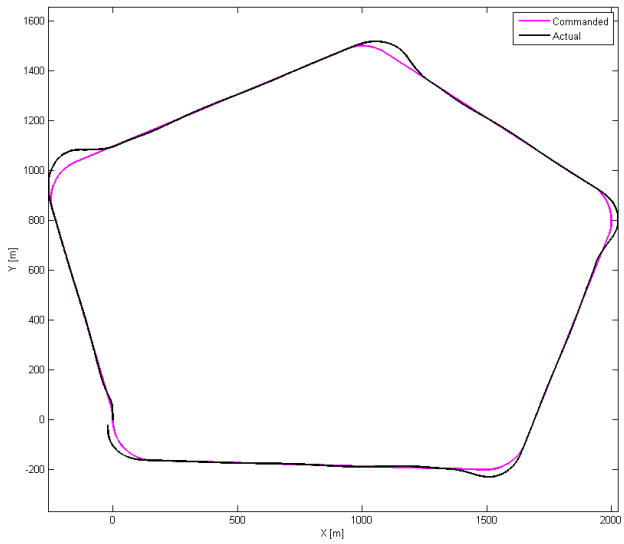

Figure C-88—Dubins Trajectory Tracking Results for Circling 3D Path at Nominal Conditions for ONLDI Controller, 2D View

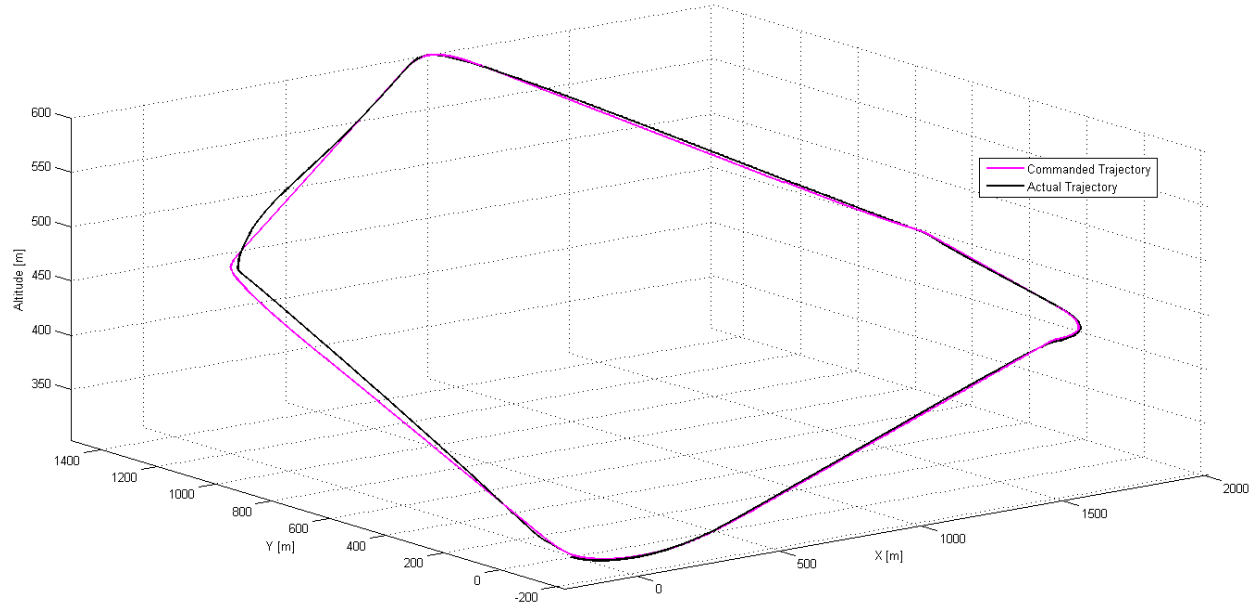

Figure C-89—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for PPID Controller, 3D View

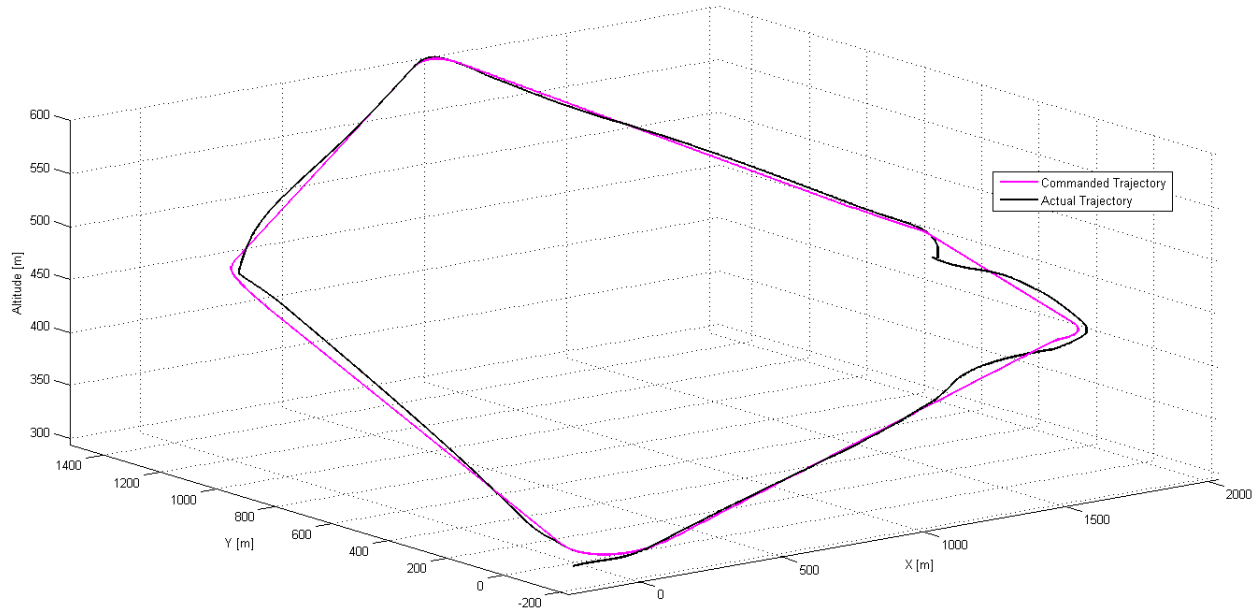

Figure C-90—Dubins Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for PPID Controller, 3D View

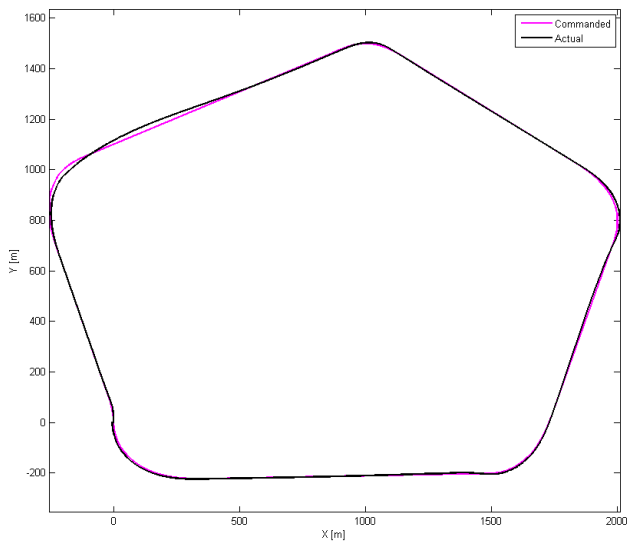

Figure C-91—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for PPID Controller, 2D View

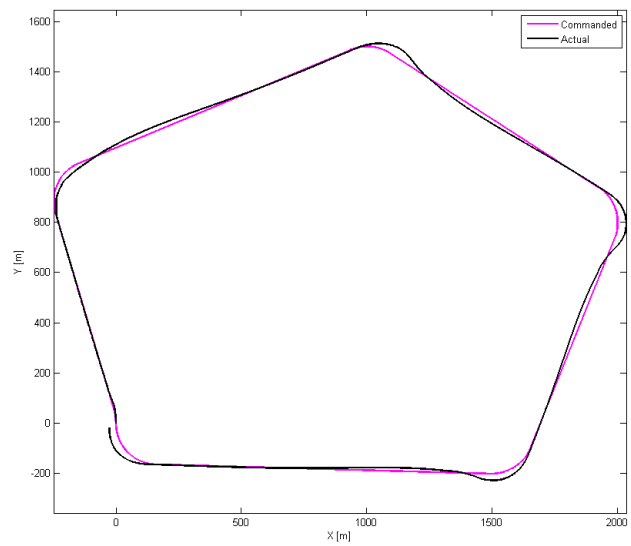

Figure C-92—Dubins Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for PPID Controller, 2D View

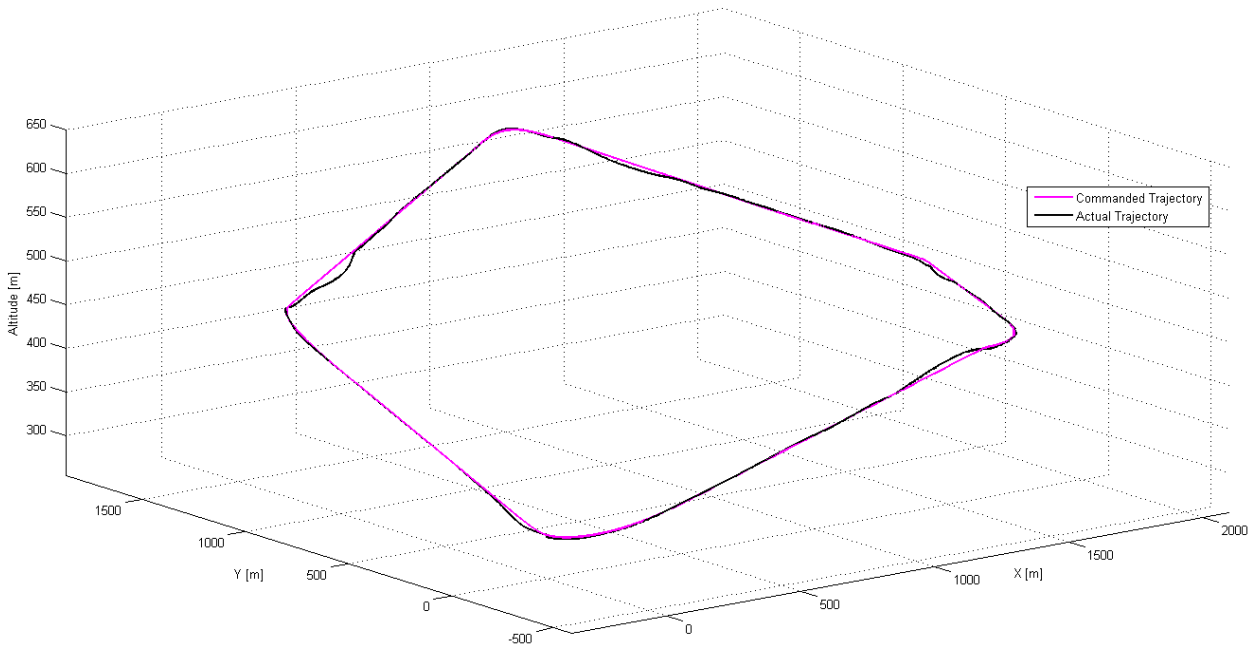

Figure C-93—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 3D View

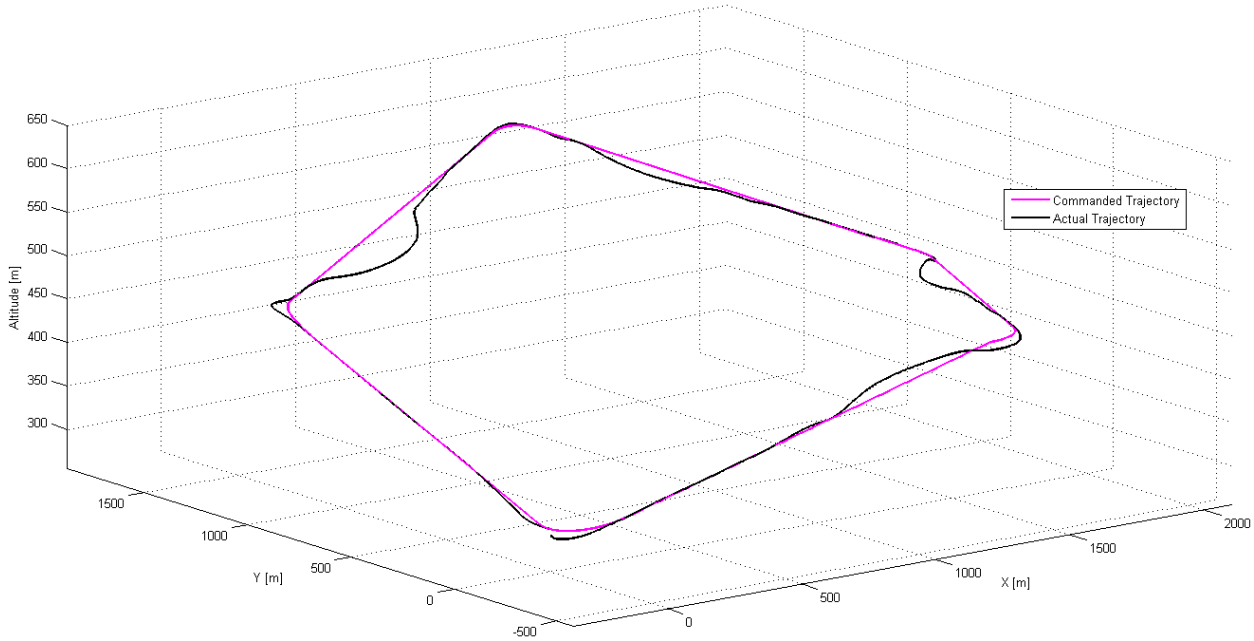

Figure C-94—Dubins Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 3D View

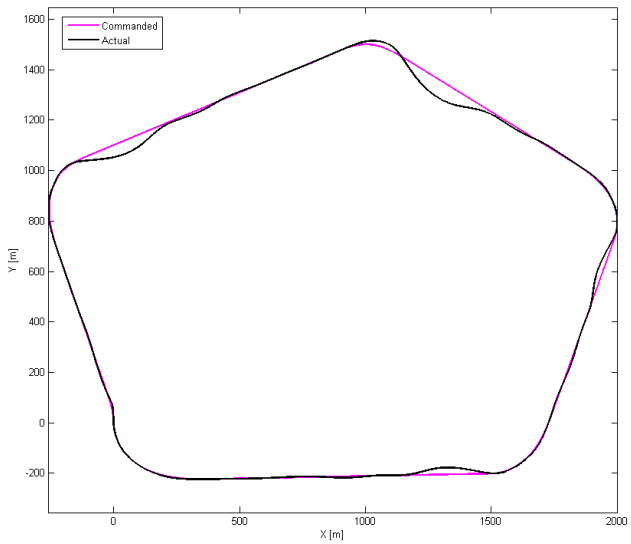

Figure C-95—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 2D View

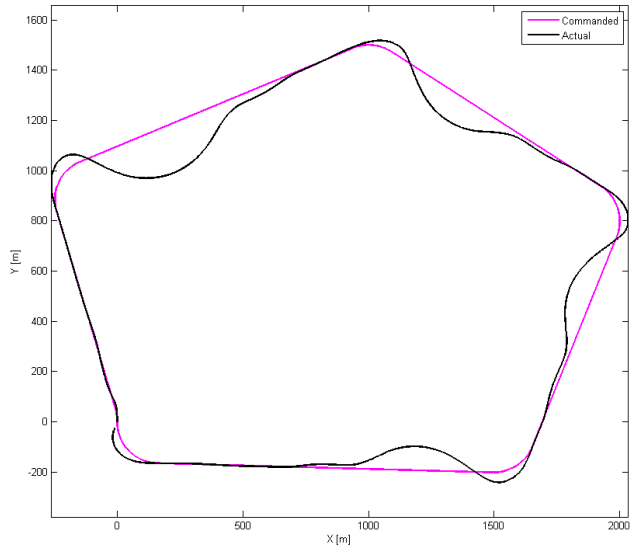

Figure C-96—Dubins Trajectory Tracking Results for Circling 3D Path with Left Aileron Stuck at 2° for ONLDI Controller, 2D View

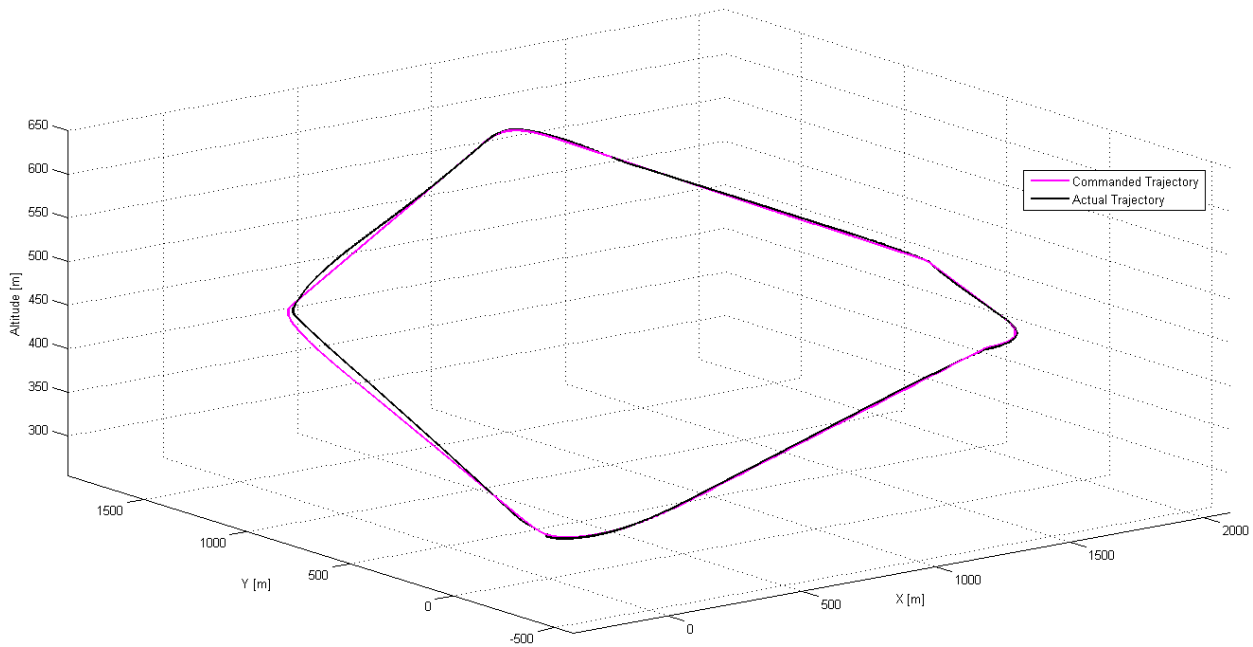

Figure C-97—Clothoid Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for PPID Controller, 3D View

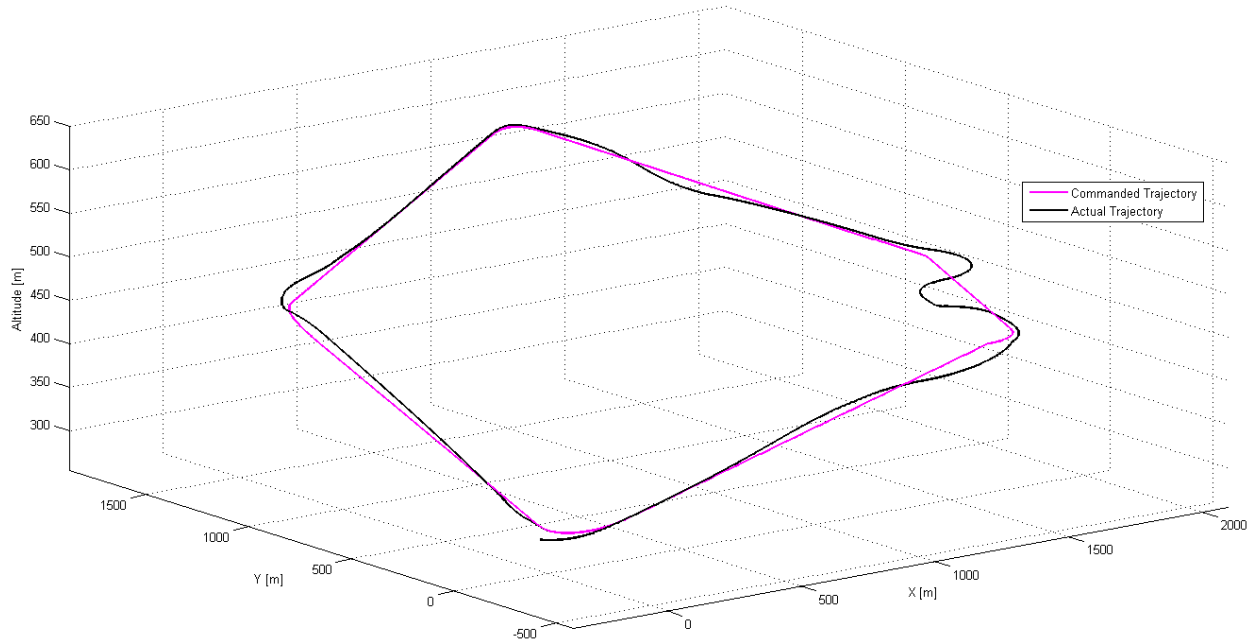

Figure C-98—Dubins Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for PPID Controller, 3D View

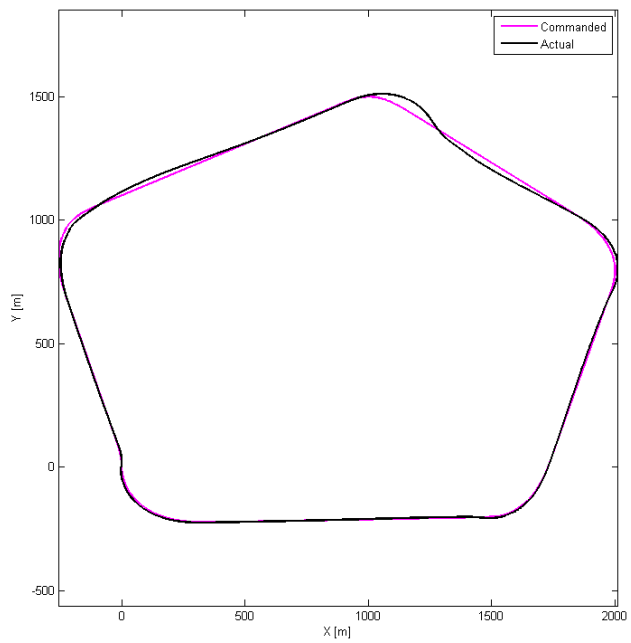

Figure C-99—Clothoid Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for PPID Controller, 2D View

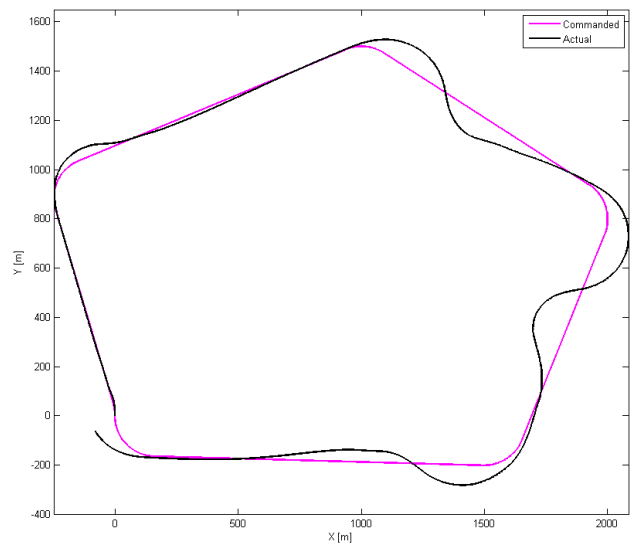

Figure C-100—Dubins Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for PPID Controller, 2D View

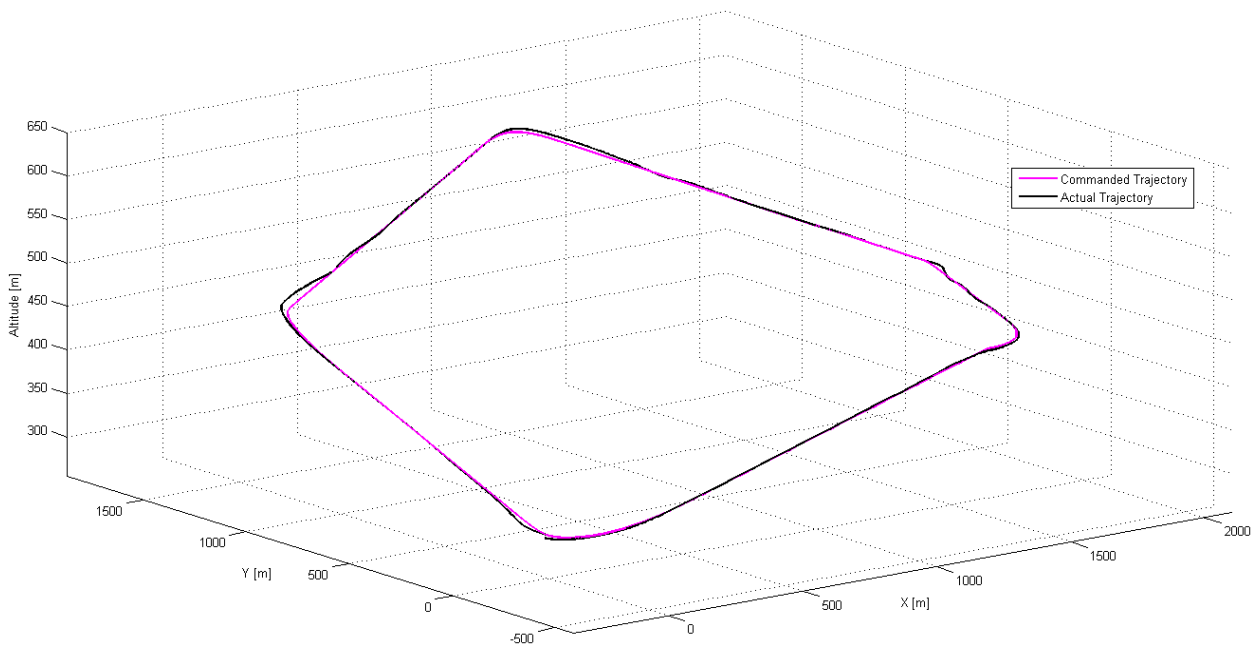

Figure C-101—Clothoid Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for ONLDI Controller, 3D View

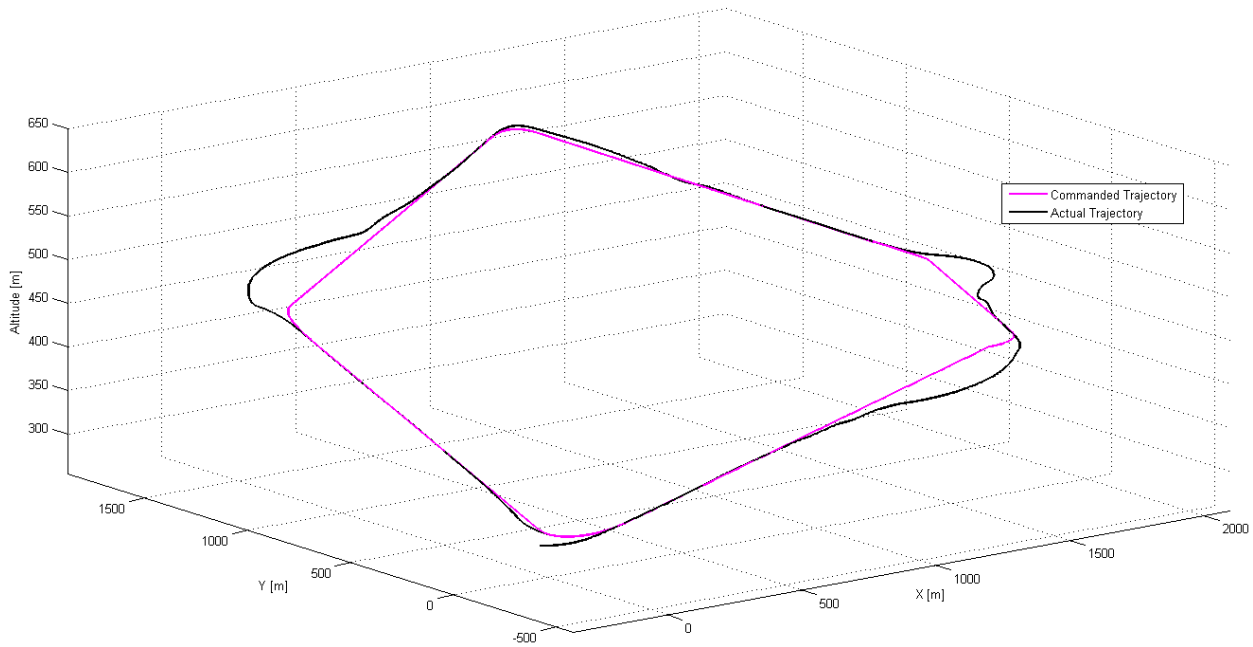

Figure C-102—Dubins Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for ONLDI Controller, 3D View

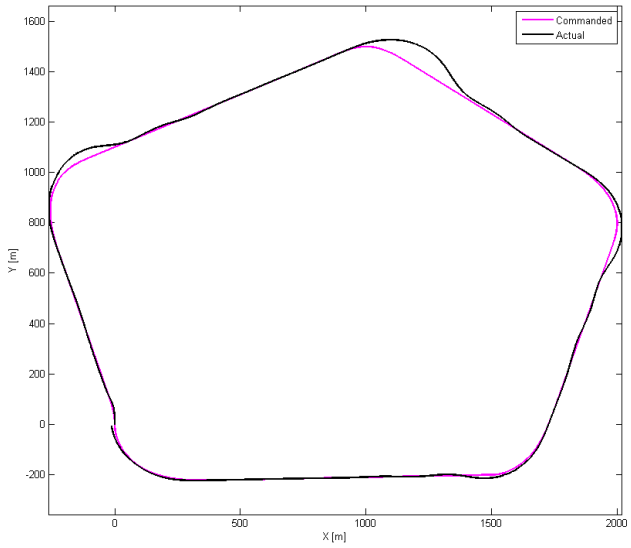

Figure C-103—Clothoid Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for ONLDI Controller, 2D View

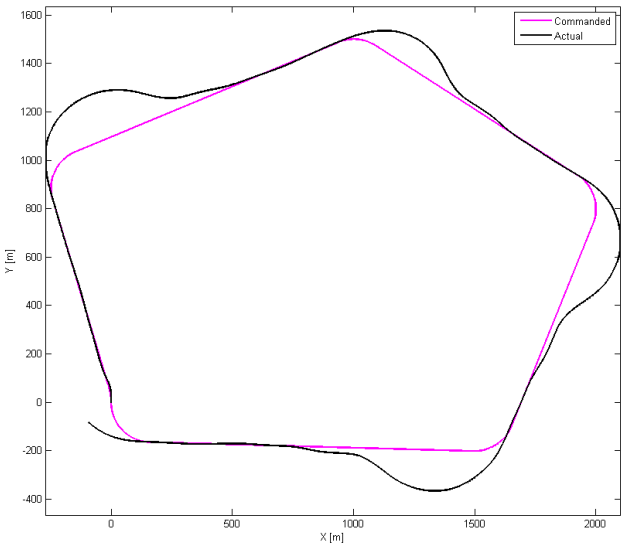

Figure C-104—Dubins Trajectory Tracking Results for Circling 3D Path with Right Aileron Stuck at 2° for ONLDI Controller, 2D View

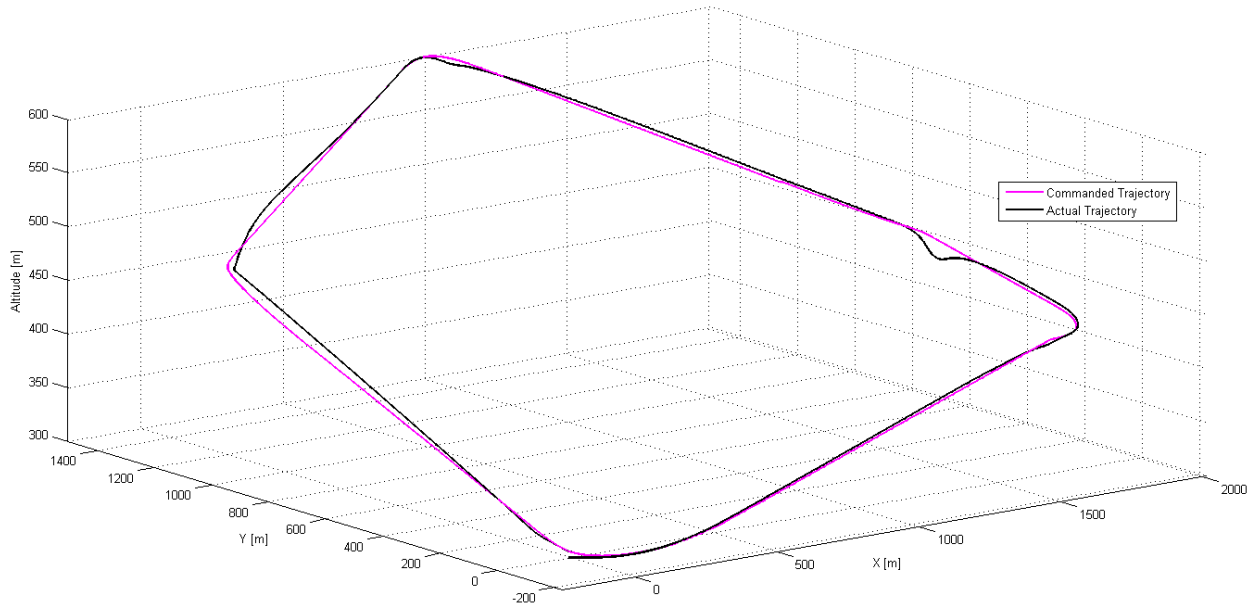

Figure C-105—Clothoid Trajectory Tracking Results for Circling 3D with Left Elevator Stuck at 2° for PPID Controller, 3D View

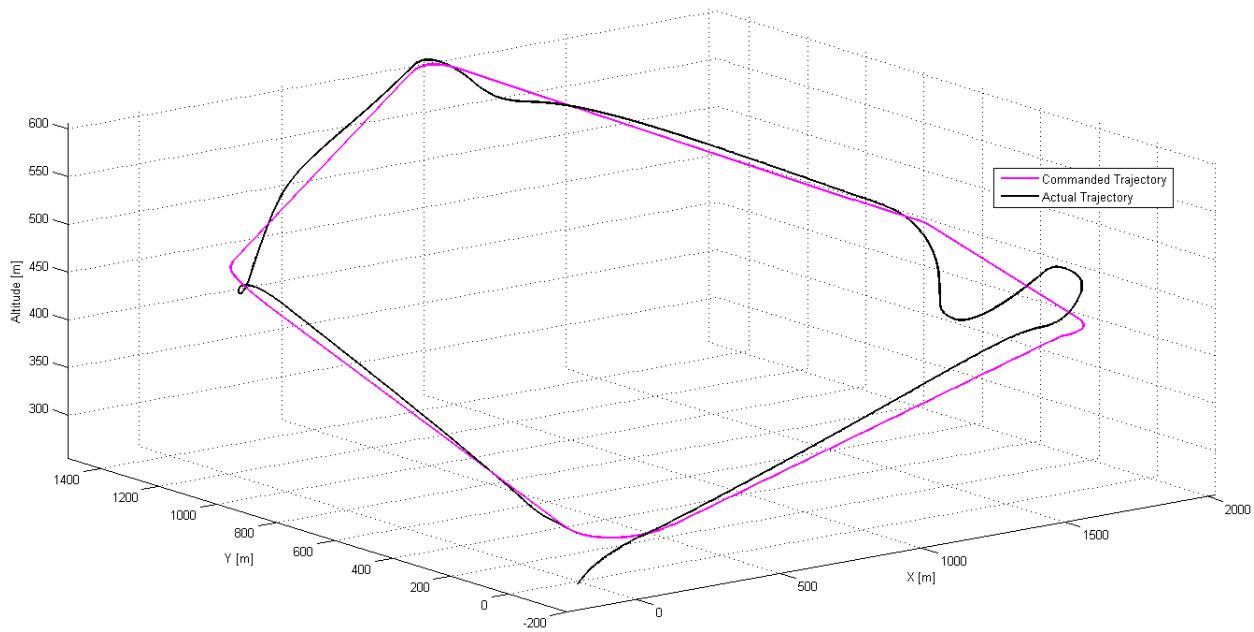

Figure C-106—Dubins Trajectory Tracking Results for Circling 3D Path with Left Elevator Stuck at 2° for PPID Controller, 3D View

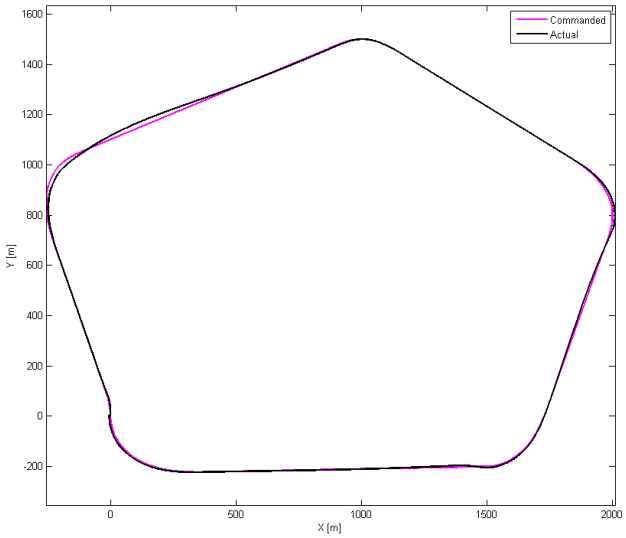

Figure C-107—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Elevator Stuck at 2° for PPID Controller, 2D View

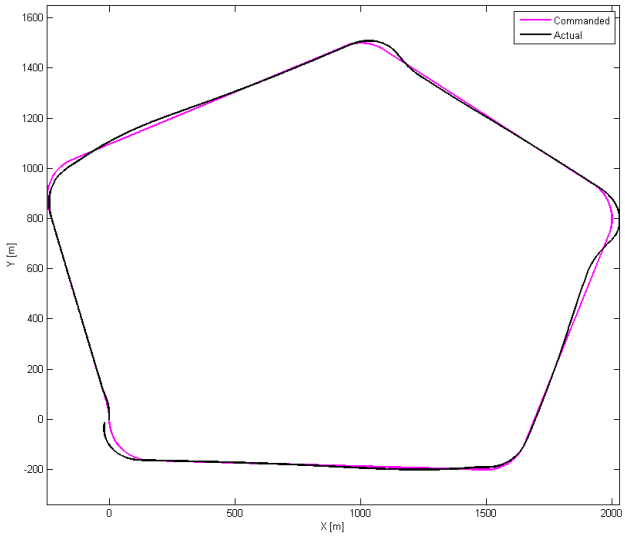

Figure C-108—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Elevator Stuck at 2° for PPID Controller, 2D View

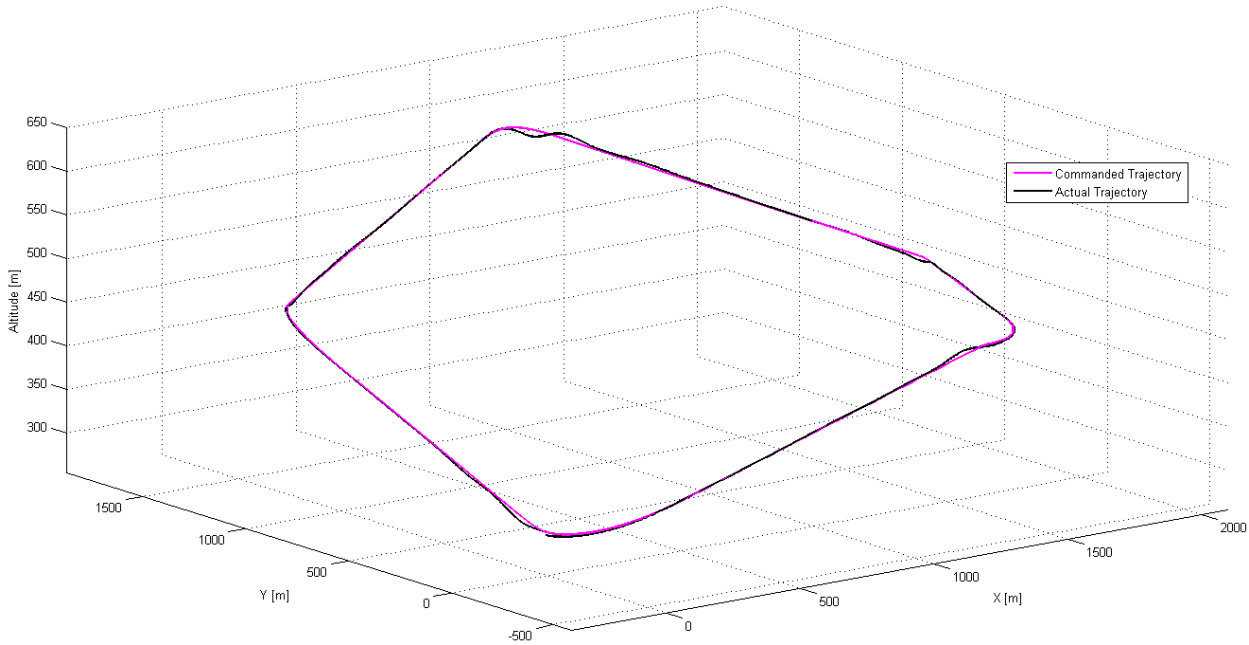

Figure C-109—Clothoid Trajectory Tracking Results for Circling 3D with Left Elevator Stuck at 2° for ONLDI Controller, 3D View

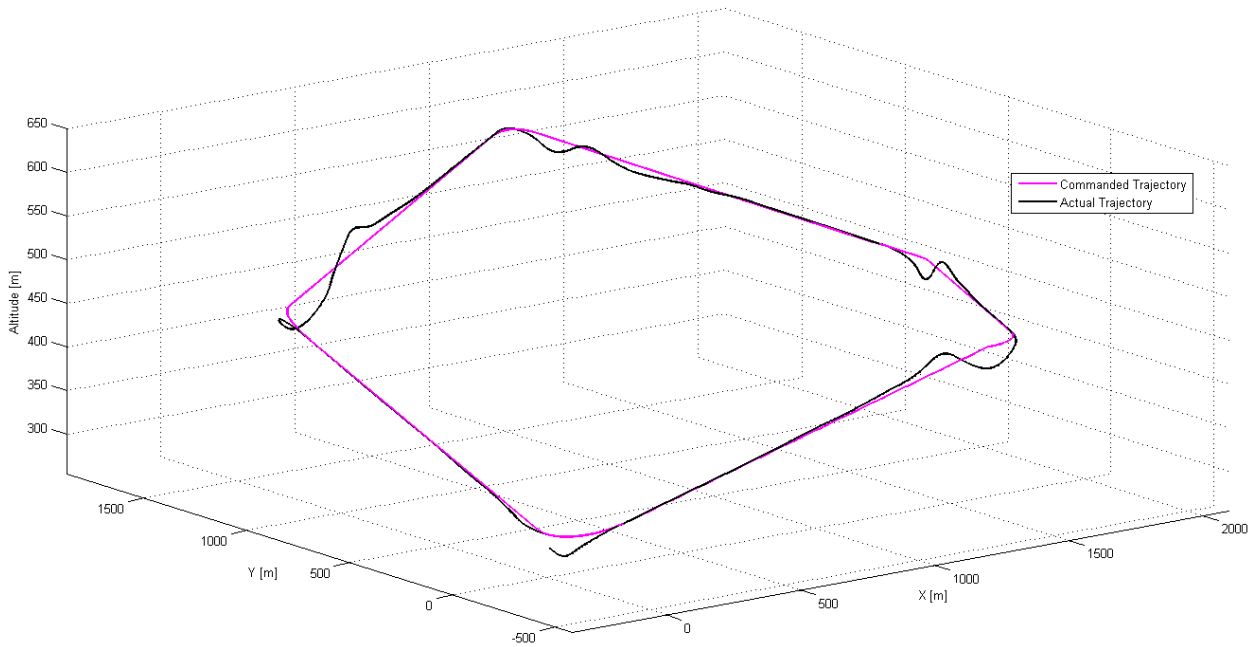

Figure C-110—Dubins Trajectory Tracking Results for Circling 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 3D View

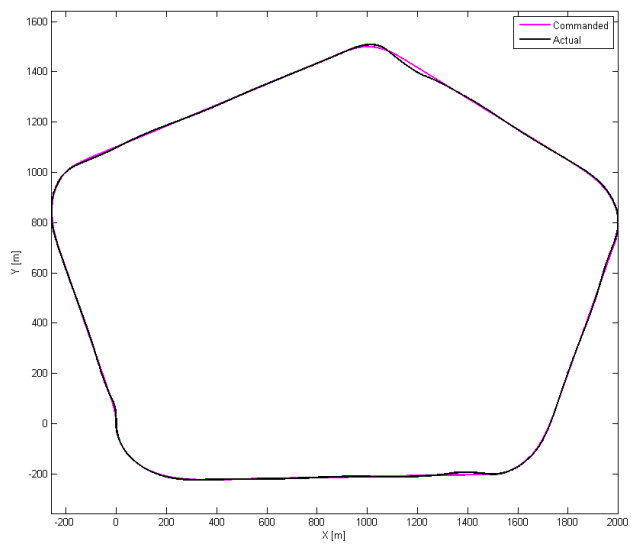

Figure C-111—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 2D View

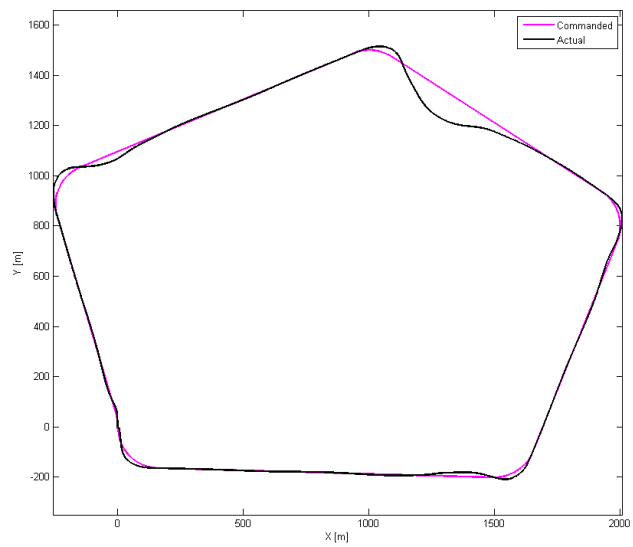

Figure C-112—Dubins Trajectory Tracking Results for Circling 3D Path with Left Elevator Stuck at 2° for ONLDI Controller, 2D View

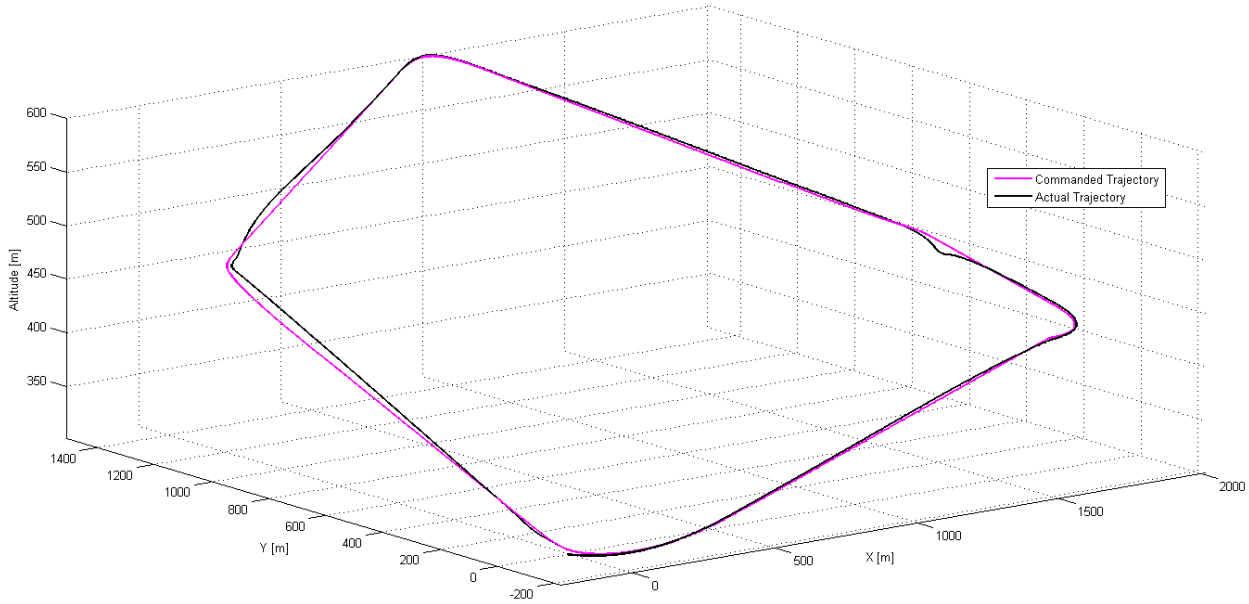

Figure C-113—Clothoid Trajectory Tracking Results for Circling 3D with Right Elevator Stuck at 2° for PPID Controller, 3D View

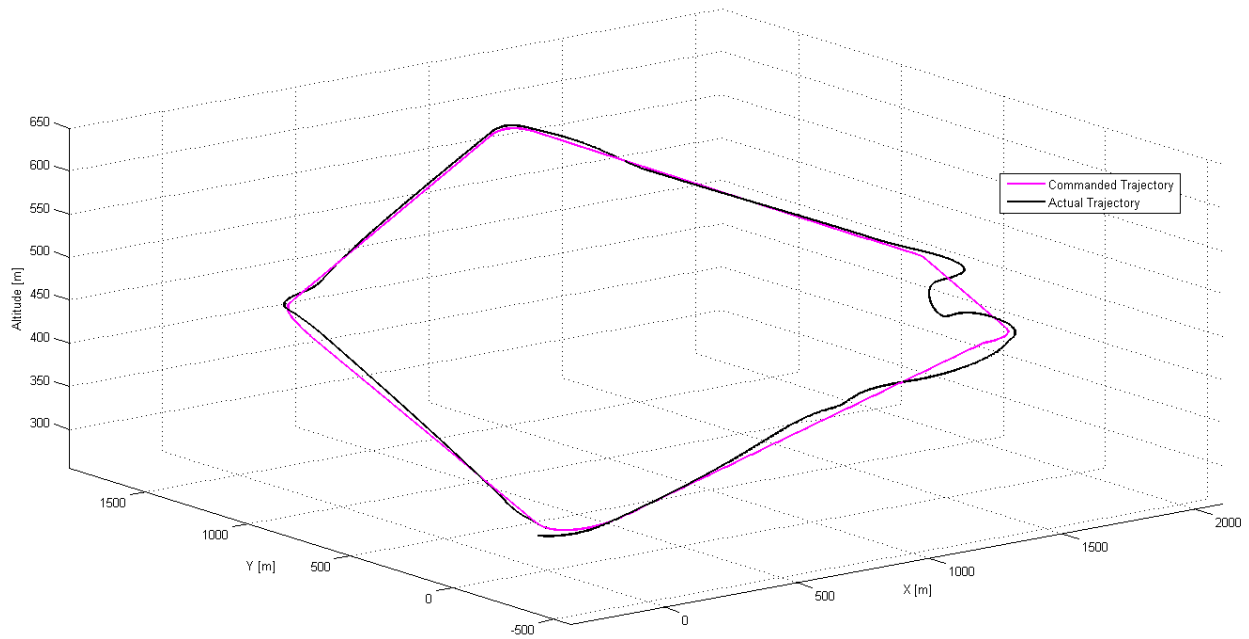

Figure C-114—Dubins Trajectory Tracking Results for Circling 3D Path with Right Elevator Stuck at 2° for PPID Controller, 3D View

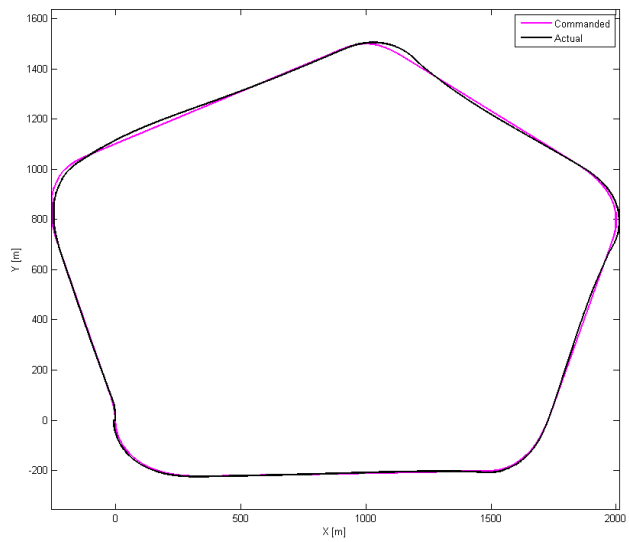

Figure C-115—Clothoid Trajectory Tracking Results for Circling 3D Path with Right Elevator Stuck at 2° for PPID Controller, 2D View

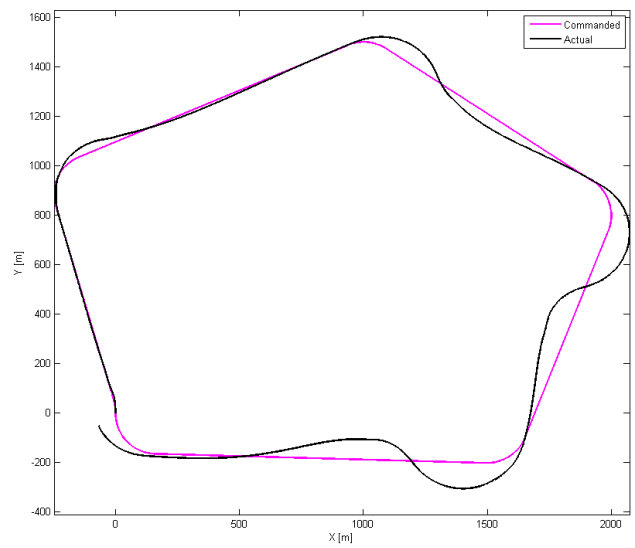

Figure C-116—Dubins Trajectory Tracking Results for Circling 3D Path with Right Elevator Stuck at 2° for PPID Controller, 2D View

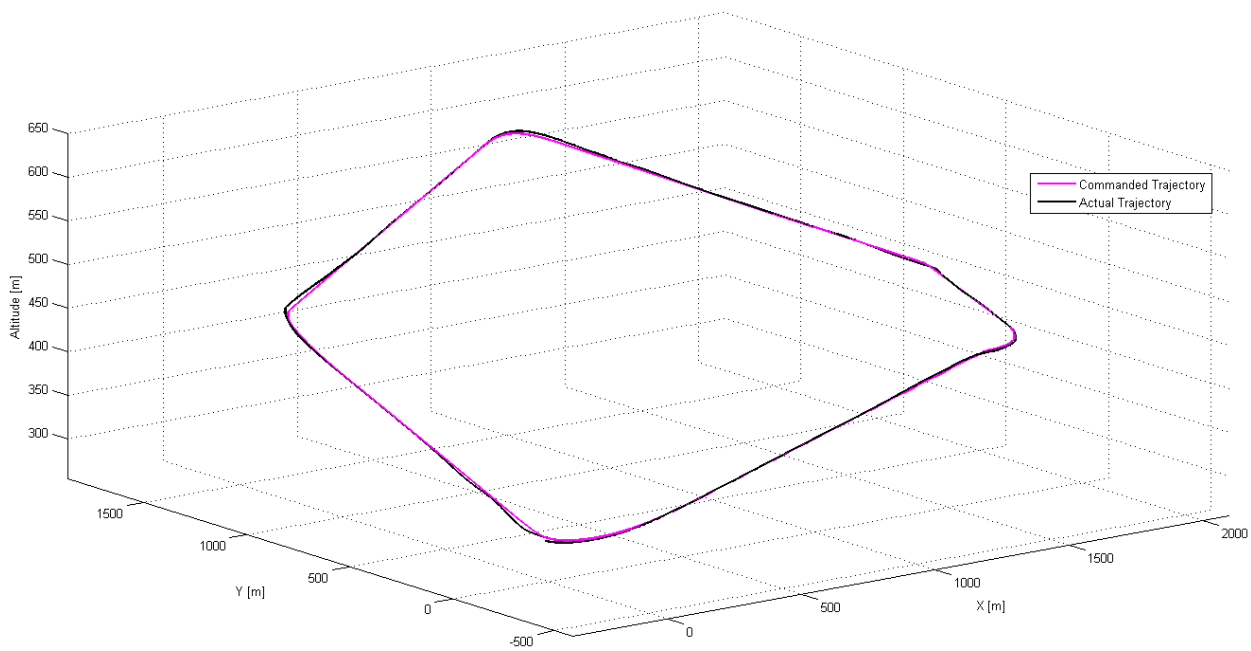

Figure C-117—Clothoid Trajectory Tracking Results for Circling 3D with Right Elevator Stuck at 2° for ONLDI Controller, 3D View

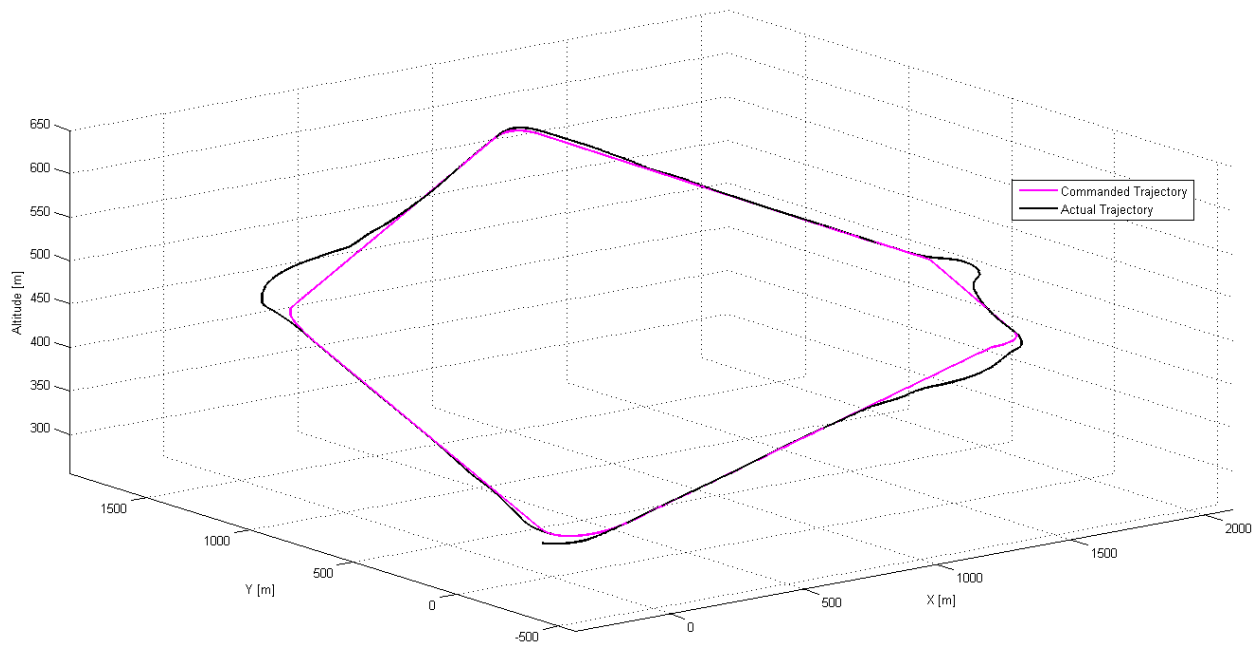

Figure C-118—Dubins Trajectory Tracking Results for Circling 3D Path with Right Elevator Stuck at 2° for ONLDI Controller, 3D View

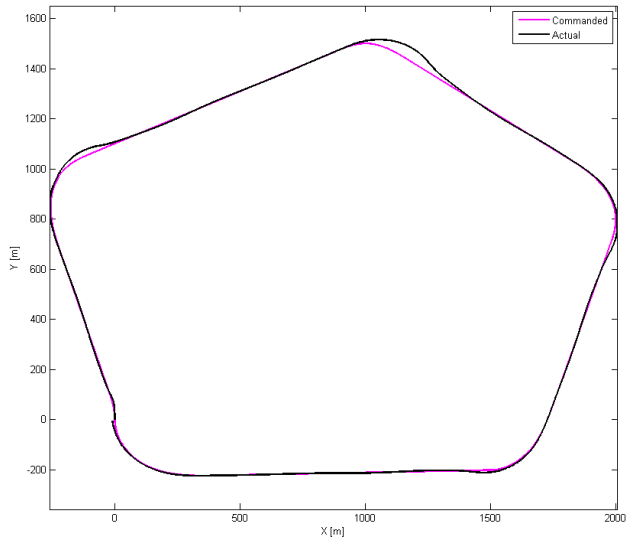

Figure C-119—Clothoid Trajectory Tracking Results for Circling 3D Path with Right Elevator Stuck at 2° for ONLDI Controller, 2D View

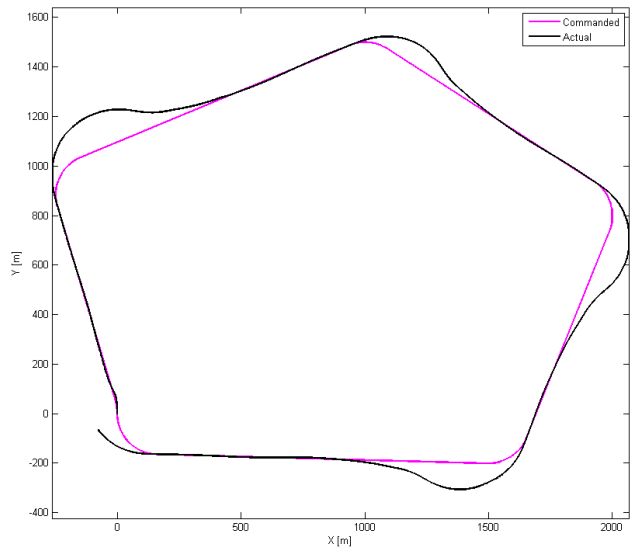

Figure C-120—Dubins Trajectory Tracking Results for Circling 3D Path with Right Elevator Stuck at 2° for ONLDI Controller, 2D View

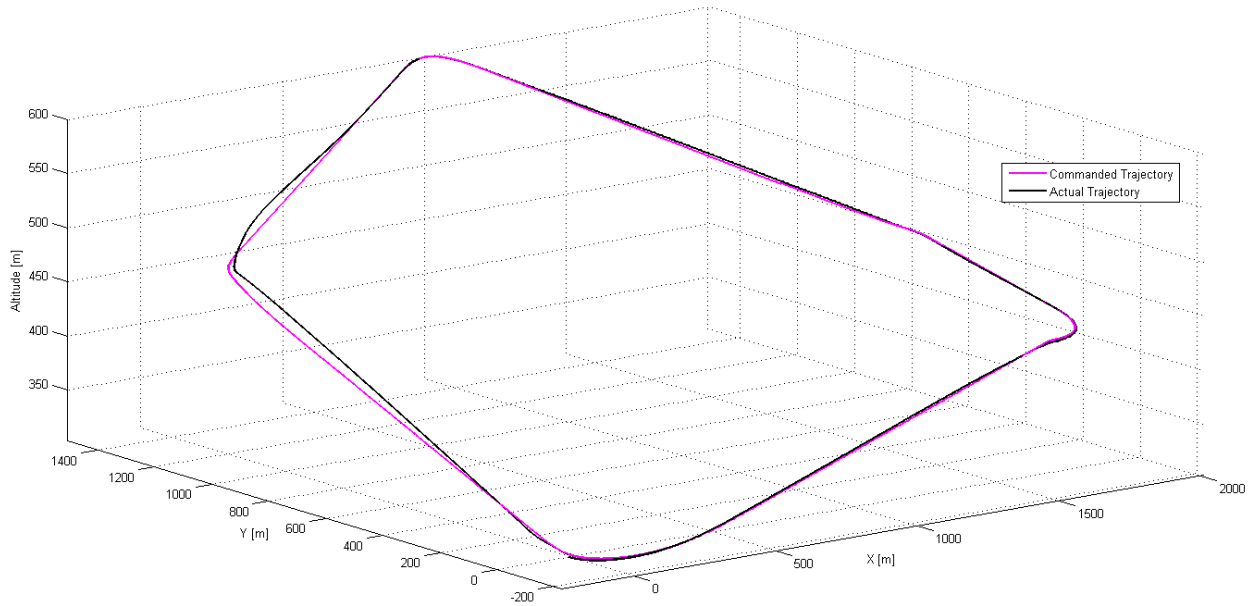

Figure C-121—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for PPID Controller, 3D View

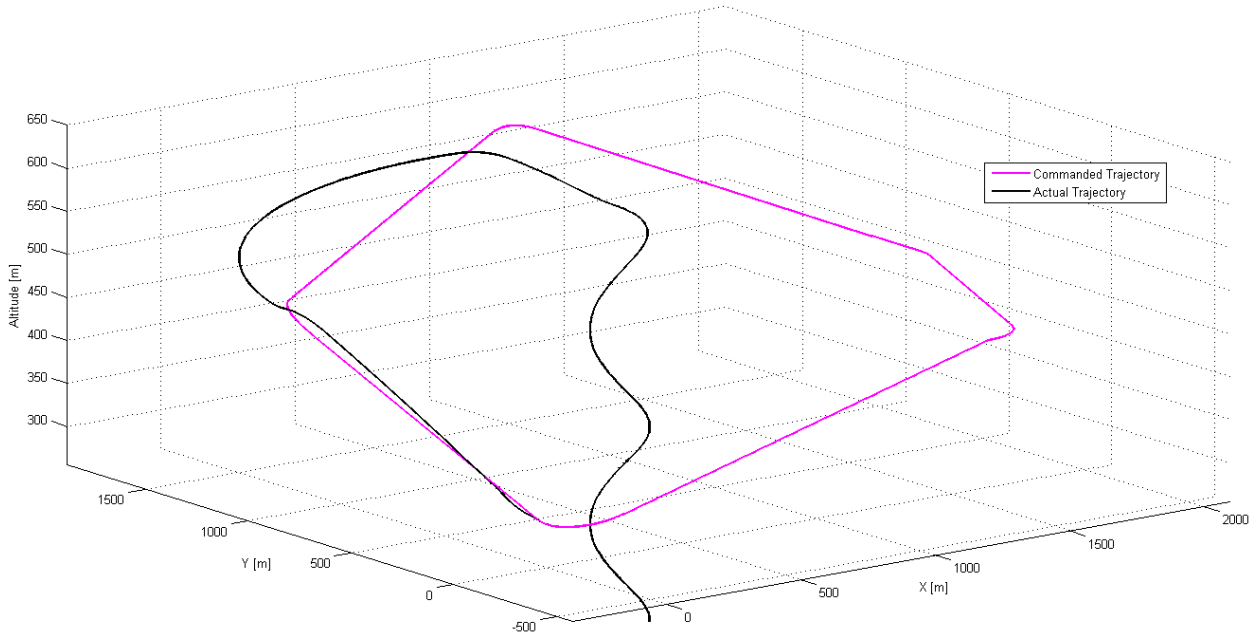

Figure C-122—Dubins Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for PPID Controller, 3D View

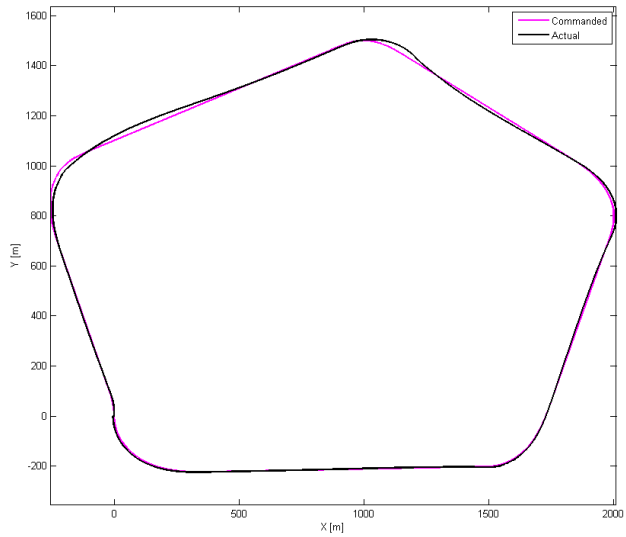

Figure C-123—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for PPID Controller, 2D View

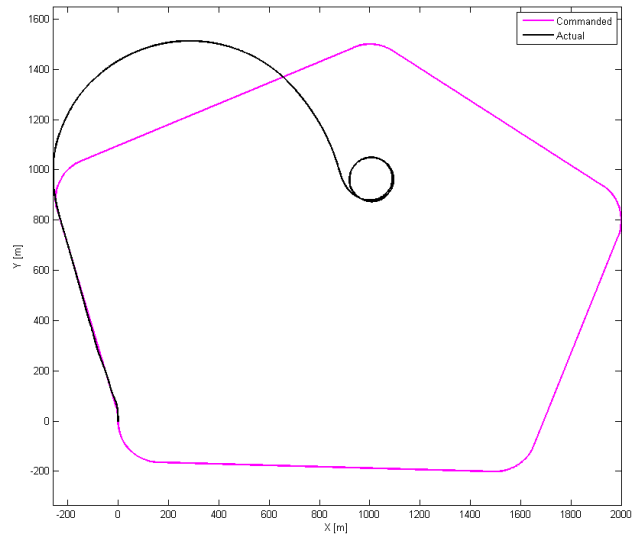

Figure C-124—Dubins Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for PPID Controller, 2D View

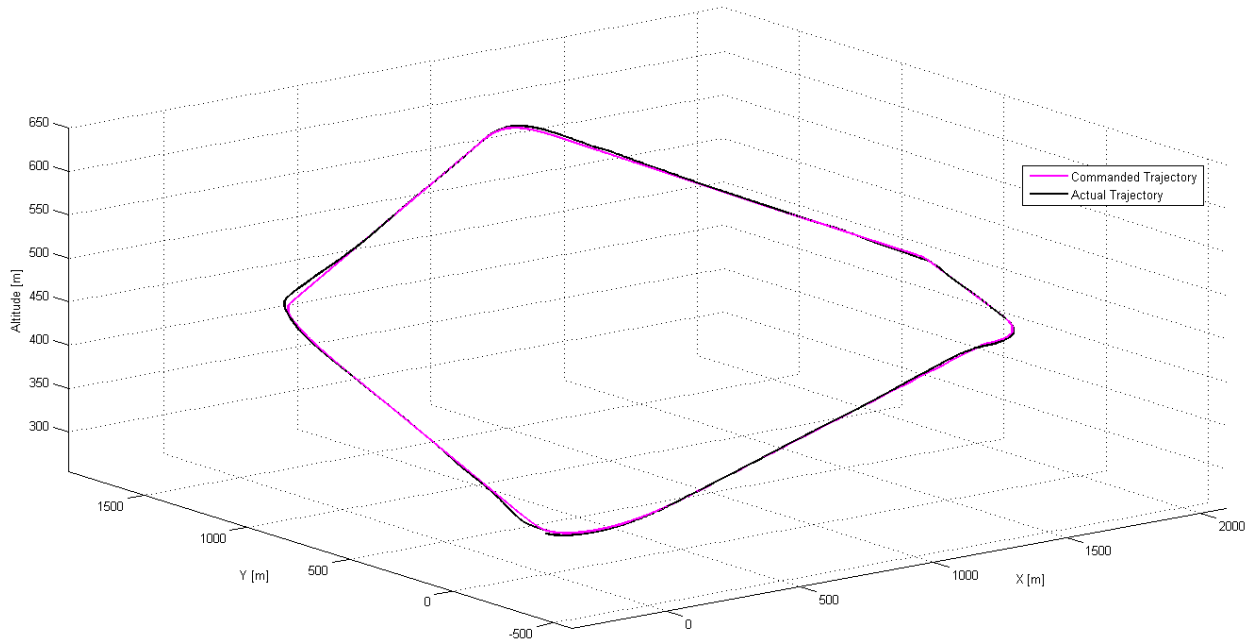

Figure C-125—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 3D View

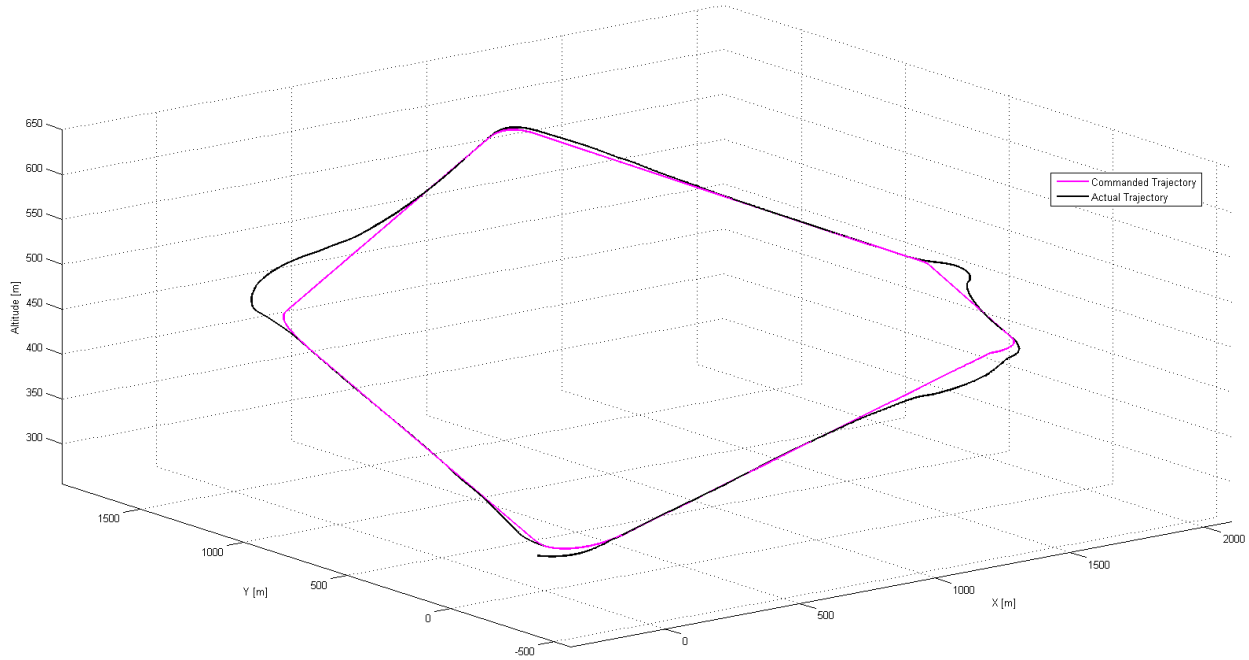

Figure C-126—Dubins Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 3D View

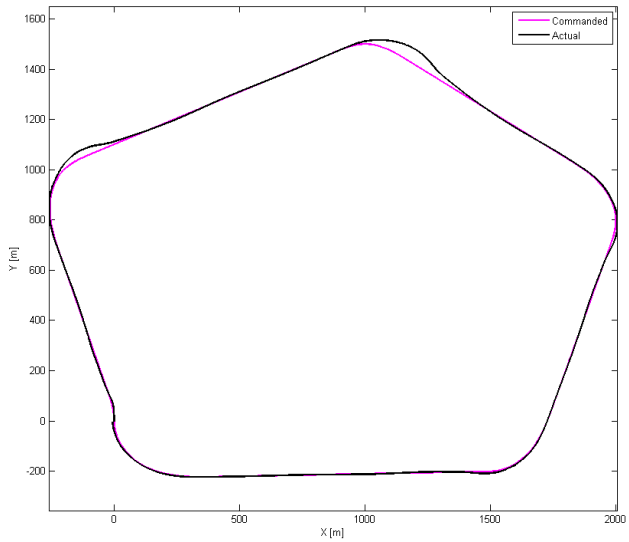

Figure C-127—Clothoid Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 2D View

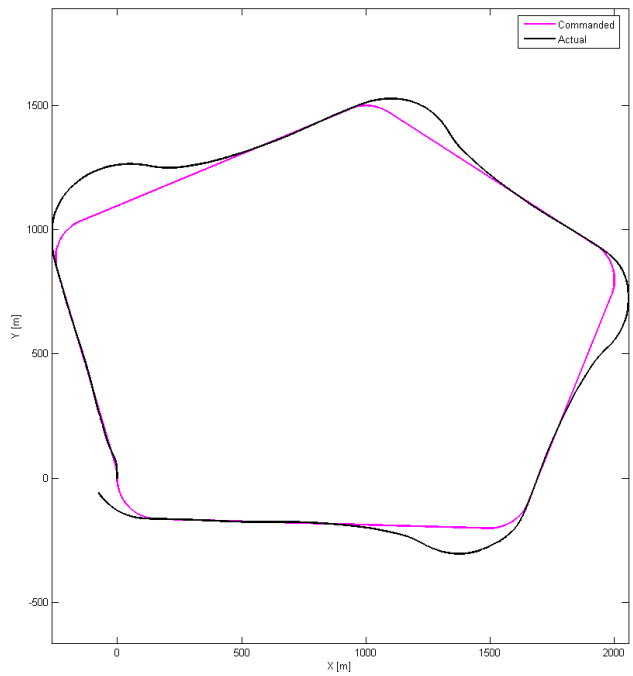

Figure C-128—Dubins Trajectory Tracking Results for Circling 3D Path with Left Rudder Stuck at 8° for ONLDI Controller, 2D View

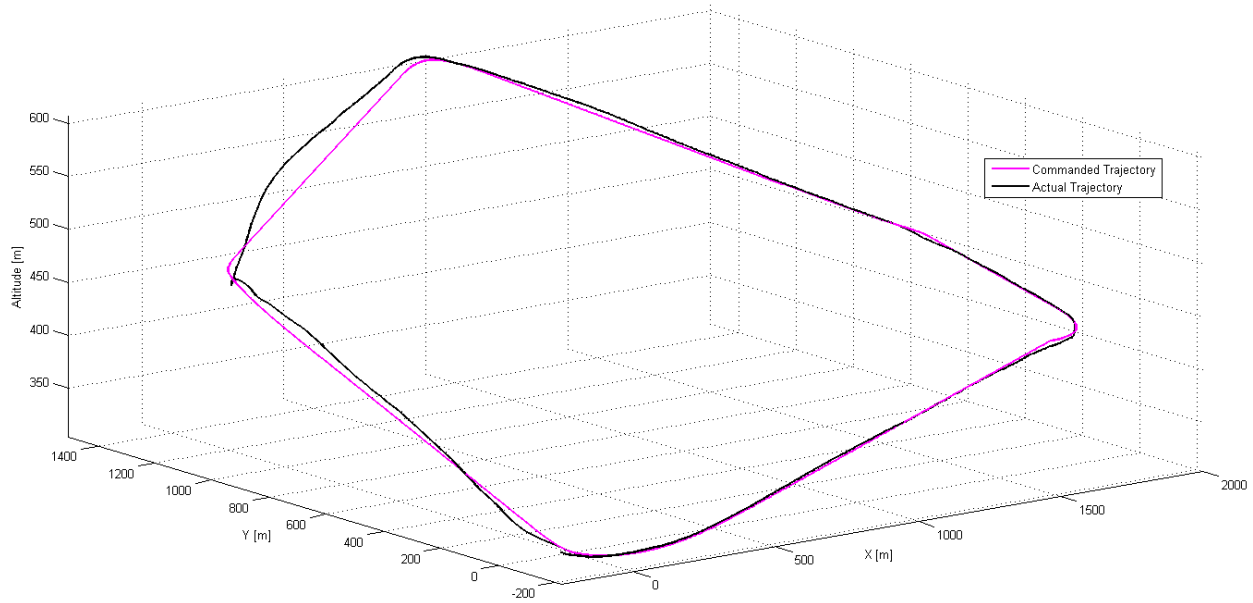

Figure C-129—Clothoid Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for PPID Controller, 3D View

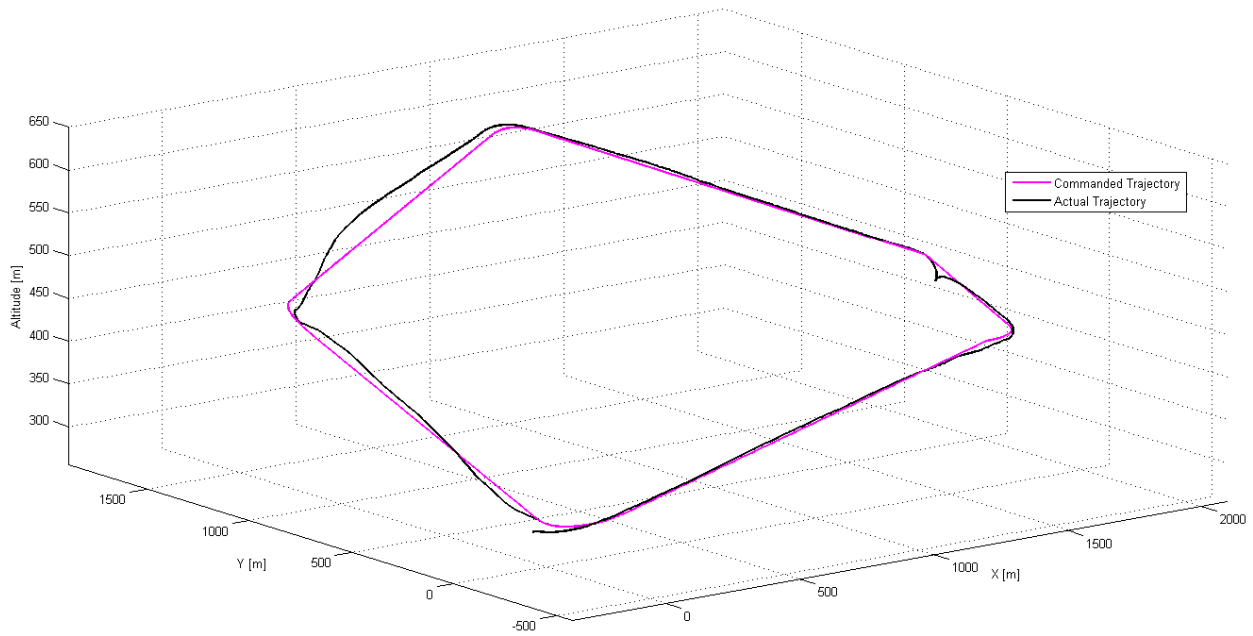

Figure C-130—Dubins Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for PPID Controller, 3D View

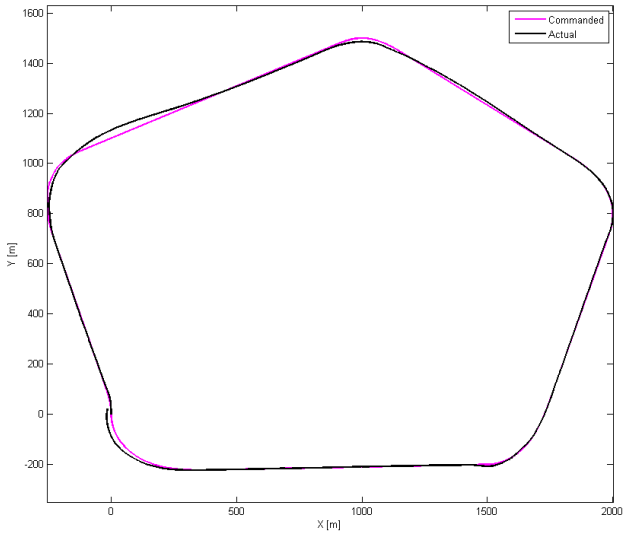

Figure C-131—Clothoid Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for PPID Controller, 2D View

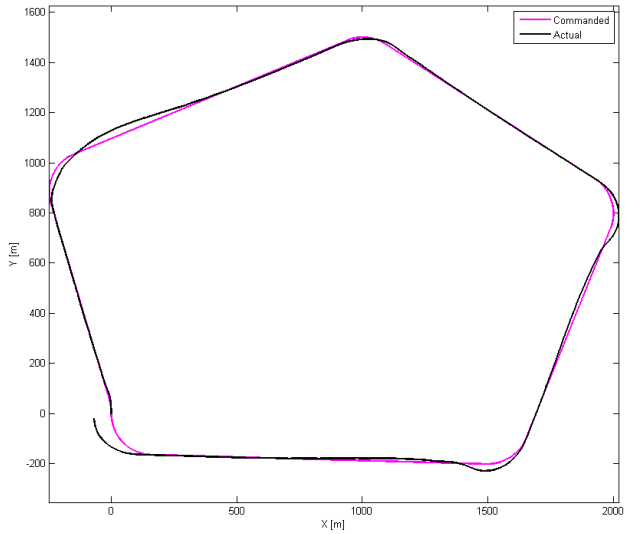

Figure C-132—Dubins Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for PPID Controller, 2D View

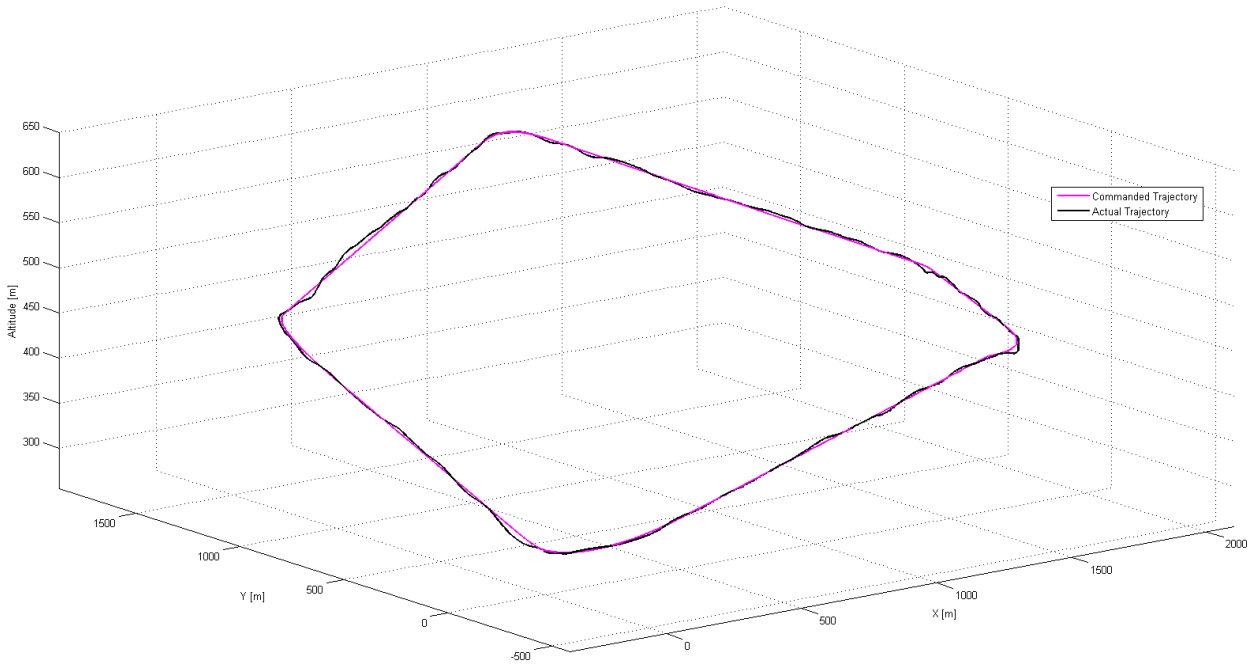

Figure C-133—Clothoid Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for ONLDI Controller, 3D View

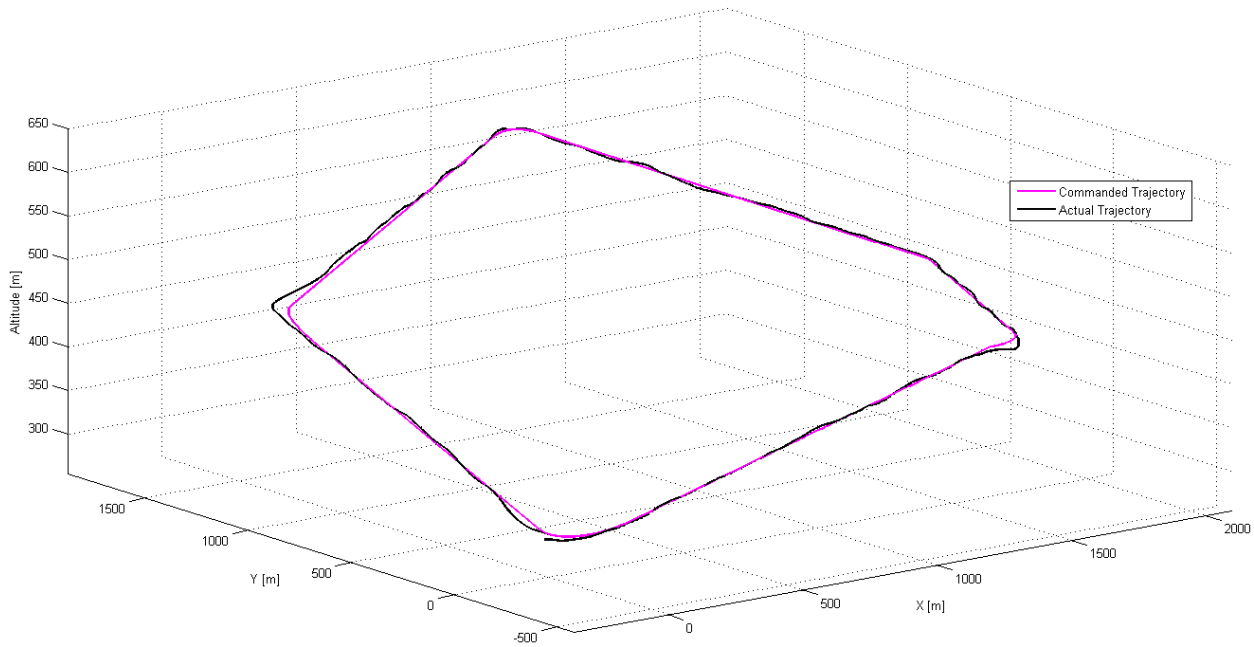

Figure C-134—Dubins Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for ONLDI Controller, 3D View

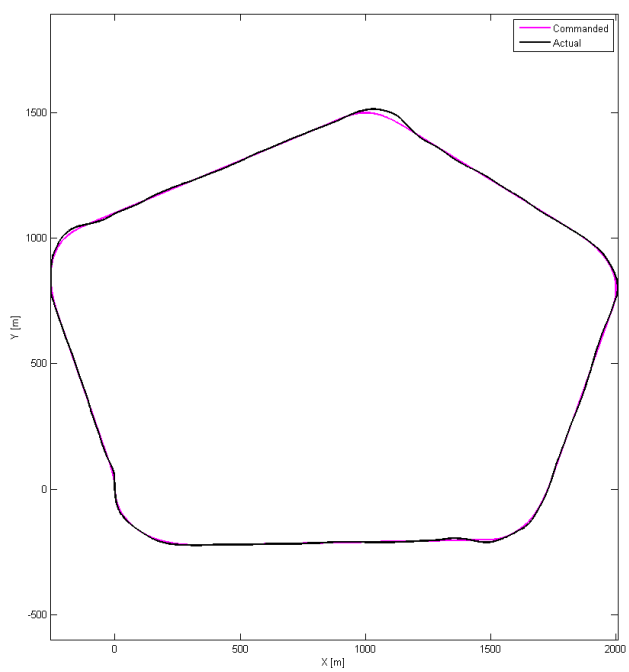

Figure C-135—Clothoid Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for ONLDI Controller, 2D View

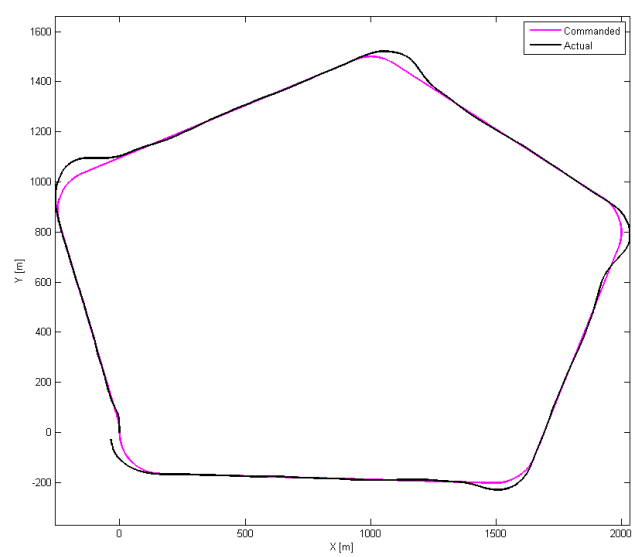

Figure C-136—Dubins Trajectory Tracking Results for Circling 3D Path with High Wind Turbulence for ONLDI Controller, 2D View

D RELATED PUBLICATIONS

1. **Wilburn J.**, Perhinschi M. G., Wilburn B., "Implementation of Composite Clothoid Paths for Continuous Curvature Trajectory Generation for UAVs", *AIAA Guidance, Navigation, and Control Conference*, Boston, MA, August 2013

2. **Wilburn J.**, Perhinschi M. G., Wilburn B., "Implementation of a 3-Dimensional Dubins-Based UAV Path Generation Algorithm", *AIAA Guidance, Navigation, and Control Conference*, Boston, MA, August 2013

3. **Wilburn, J.**, Perhinschi, M. G., Wilburn, B., "Enhanced Modified Voronoi Algorithm for UAV Path Planning and Obstacle Avoidance," *International Review of Aerospace Engineering (IREASE)*, April 2013.

4. Wilburn B., Perhinschi M. G., Moncayo H., Karas O., **Wilburn J.**, "Unmanned Aerial Vehicle Trajectory Tracking Algorithm Comparison", *International Journal of Intelligent Unmanned Systems*, Vol. 3, No. 3, pp. 276-302, 2013

5. **Wilburn, J.**, Perhinschi, M. G., Wilburn, B., "UAV Clothoid Path Generation Algorithm with Shortest Path Criterion," submitted to the *AIAA Journal of Guidance, Control, and Dynamics* in March 2013

6. **Wilburn J.**, Perhinschi M. G., Wilburn B., Karas O., "Development of a Modified Voronoi Algorithm for UAV Path Planning and Obstacle Avoidance", *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Minneapolis, MN, August 2012

7. **Wilburn J.**, Cole J., Perhinschi M. G., Wilburn B., "Comparison of a Fuzzy Logic Controller to a Potential Field Controller for Real-Time UAV Navigation", *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Minneapolis, MN, August 2012

8. Perhinschi M. G., Moncayo H., **Davis J.**, Wilburn B., Karas O., Wathen M., "Development of a Simulation Environment for Autonomous Flight Control Algorithms", *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, Portland, OR, August 2011