Summer 2016

# Detecting and Resolving Air Traffic Conflicts Using a Point of Closest Approach Method

Marc S. Easton
*Old Dominion University*

# DETECTING AND RESOLVING AIR TRAFFIC CONFLICTS USING A POINT OF CLOSEST APPROACH METHOD

by

Marc S. Easton
B.S., May 2013, Virginia Polytechnic Institute and State University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

AEROSPACE ENGINEERING

OLD DOMINION UNIVERSITY
August 2016

## ABSTRACT

DETECTING AND RESOLVING AIR TRAFFIC CONFLICTS USING A POINT OF CLOSEST APPROACH METHOD

Marc S. Easton
Old Dominion University, 2016
Director: Dr. Brett Newman

A geometrical point of closest approach method is used to solve air traffic conflict detection/resolution problems with a single conflict vehicle that is unaware or unable to aid in resolving the conflict. Nonlinear, three degree of freedom equations of motion for a point-mass vehicle are derived and formulated to allow commanded trajectories to steer the vehicle to a desired location. A dynamic model is developed to propagate the vehicle in three dimensions. A closed-loop model implementing a negative feedback controller using a Proportional-Integral-Derivative control scheme is used to drive the vehicle to match the commanded trajectories. A conflict detection/resolution algorithm implementing a point of closest approach method is developed to determine the point at which the simulated, target vehicle will be just on the edge of a safe sphere surrounding the conflict aircraft to simulate the Federal Aviation Administration's requirements for proper spacing between aircraft. A velocity vector is created to steer the target vehicle to this point to avoid any conflict. MathWorks' Simulink computational environment is used to simulate the target vehicle and conflict vehicle. Various trajectories for the target vehicle and the conflict vehicle are tested to evaluate the performance of the algorithm. The algorithm performed satisfactorily in detecting and steering the vehicle away from a conflict, always improving the relative spacing between the two vehicles. However, the algorithm was lacking in capability to precisely satisfy the separation requirement. In all cases the target vehicle mildly penetrated the safe region. Future research directions are discussed with the goal of improving the conflict detection/resolution algorithm performance so that the separation requirement can be reliably met.

## ACKNOWLEDGEMENTS

I would like to thank my academic advisor, Dr. Brett Newman, for providing me the opportunity to participate in this research as well as for guiding me through this process and for his constant pressure to always encourage me to be a better student and a better engineer. I would also like to thank Dr. Thomas Alberts for his teaching and review of my thesis and Dr. Khan Iftekharuddin for serving on my committee and reviewing my thesis. My sincerest thanks also go out to my parents, Jim and Cheryl Easton, and my better half, Jaymie Schioppa, for without them, none of this would be possible.  This work is supported by research grant NNL14AA11C-PR-4200517291, NASA Langley Research Center.

# TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

Figure Page

Figure Page

# CHAPTER 1

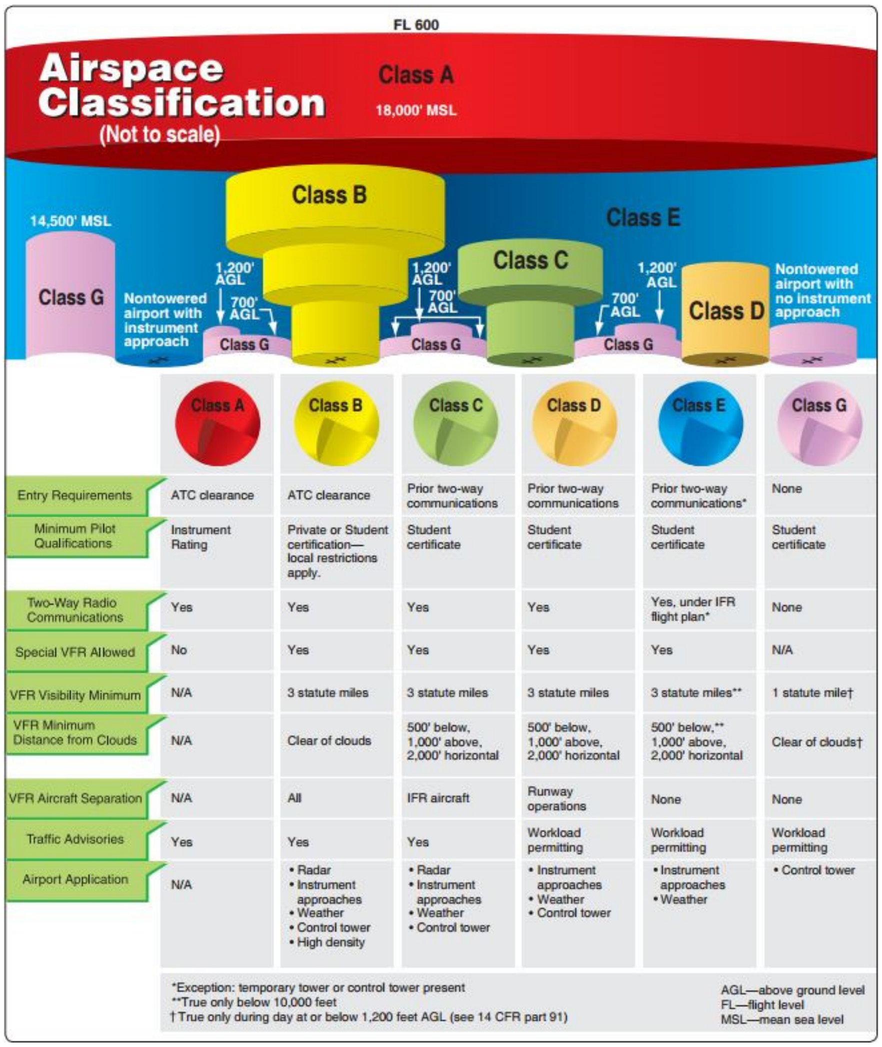
# INTRODUCTION

## 1.1 Overview

Unmanned Aerial Systems (UAS) are becoming more prevalent in today's society. Small unmanned aircraft are seeing increased use by both hobbyists and professionals. With the advancing capabilities, any barrier to adopting UAS by turnkey purchase or in-house design-build-fly efforts is quickly disappearing. More and more technology is being developed in order to remove the human pilot and completely automate these UAS. Autonomous Unmanned Aerial Systems (AUAS) pose a problem when attempting to integrate into the Federal Aviation Administration's (FAA) airspace. If an AUAS is on a trajectory that will cause some kind of conflict with another aircraft that will violate FAA regulations, the AUAS needs to have a way to be able to detect that conflict and come up with a resolution that will keep both vehicles from experiencing the conflict and simultaneously satisfying separation assurance requirements. Various methods exist for implementing a Conflict Detection/Resolution (CDR) algorithm with varying degrees of complexity. This function is also commonly referred to as Sense and Avoid (SAA) in the UAS community. This thesis focuses on development of a three-dimensional Point of Closest Approach (PCA) algorithm to perform the CDR autonomous flight control function and creating a nonlinear simulator to implement and test this algorithm. The simulator focuses on only one conflict vehicle, however it can be modified to deal with multiple conflict vehicles for simulations of larger and more crowded airspaces such as at an airport.

## 1.2 Literature Review

The content of this thesis focuses on the development and testing of a CDR algorithm. An important initial step is to review the history of conflict detection and resolution in aviation. For as long as society has been using aircraft for mass transport of goods and people, there have been conflicts and collisions between aircraft that results in loss of property and life. The FAA has strict regulations that dictate where airspace is controlled and monitored by the FAA and where it is uncontrolled.[1,2] Figure 1.1 illustrates the varying controlled and uncontrolled airspaces existing in the United States, depending on traffic density, flight operations, and current weather conditions. The incident that sparked the increased research and regulation of aircraft conflicts was the collision between Trans World Airlines Flight 2 and United Airlines Flight 718 over the Grand Canyon in Arizona on June 30, 1956. The collision between these two aircraft occurred in uncontrolled airspace and resulted in the loss of all passengers and crew. In uncontrolled airspace, there are no ground radar systems and air traffic controllers to direct and manage traffic. Both aircraft were visually impaired by cloud formations and collided.[3] A need was seen for expanded ground radar systems and on-board systems that could detect and prevent conflicts between two aircraft or an aircraft and the ground in both controlled and uncontrolled airspace. In the rapidly evolving and modern airspace environment called the Next Generation (NextGen) National Airspace System (NAS), Global Positioning System (GPS) navigation sensors and Automatic Dependent Surveillance Broadcast (ADS-B) transceivers expand this capability considerably.[4]



| | Class A | Class B | Class C | Class D | Class E | Class G |
|---|---|---|---|---|---|---|
| Entry Requirements | ATC clearance | ATC clearance | Prior two-way communications | Prior two-way communications | Prior two-way communications* | None |
| Minimum Pilot Qualifications | Instrument Rating | Private or Student certification—local restrictions apply. | Student certificate | Student certificate | Student certificate | Student certificate |
| Two-Way Radio Communications | Yes | Yes | Yes | Yes | Yes, under IFR flight plan* | None |
| Special VFR Allowed | No | Yes | Yes | Yes | Yes | N/A |
| VFR Visibility Minimum | N/A | 3 statute miles | 3 statute miles | 3 statute miles | 3 statute miles** | 1 statute mile† |
| VFR Minimum Distance from Clouds | N/A | Clear of clouds | 500' below, 1,000' above, 2,000' horizontal | 500' below, 1,000' above, 2,000' horizontal | 500' below,** 1,000' above, 2,000' horizontal | Clear of clouds† |
| VFR Aircraft Separation | N/A | All | IFR aircraft | Runway operations | None | None |
| Traffic Advisories | Yes | Yes | Yes | Workload permitting | Workload permitting | Workload permitting |
| Airport Application | N/A | • Radar • Instrument approaches • Weather • Control tower • High density | • Radar • Instrument approaches • Weather • Control tower | • Instrument approaches • Weather • Control tower | • Instrument approaches • Weather | • Control tower |

*Exception: temporary tower or control tower present
**True only below 10,000 feet
†True only during day at or below 1,200 feet AGL (see 14 CFR part 91)

AGL—above ground level
FL—flight level
MSL—mean sea level

**Figure 1.1 FAA National Airspace Classifications**[1]

While air traffic controllers with appropriate data can manage air traffic in controlled airspace effectively, it is good to have backup systems in the case of unforeseen circumstances. Additionally, in uncontrolled airspaces vehicles must rely on in situ sensors and visual information to avoid conflicts. While these systems can help pilots detect a conflict, there needs to be a system that also aids pilots in resolving this conflict while still maintaining a correct level of separation between the two aircraft as regulated by the FAA. Conflict detection and resolution has been a subject of research for quite some time. Mostly it has been focused on controlling traffic in airspace around high traffic areas, such as airports. Conflict detection and resolution algorithms were developed for the Terminal Radar Approach Control (TRACON) facilities that are integrated into most medium to large sized airports.[5] A similar system employing air traffic control CDR algorithms for high altitude air routes between airports is implemented by the Air Route Traffic Control Centers (ARTCC). An on-board system for increased safety and protection against commercial airliner collision accidents was developed and called the Traffic Collision Avoidance System (TCAS). The TCAS can detect potential conflicts and develop a trajectory that will be fed to the pilots to make a corrective course of action.[6] Figure 1.2 shows how TCAS behaves at various ranges to identify threats and conflicts. Development and mandatory implementation of TCAS originated from another midair collision involving a Piper Archer general aviation aircraft and a DC-9 commercial airliner near Cerritos, California, on August 31, 1986.[7] A similar on-board system to avoid ground collision is the Ground Collision Avoidance System (GCAS). Although midair collisions between manned aircraft are quite rare, they can and still do occur.[8]

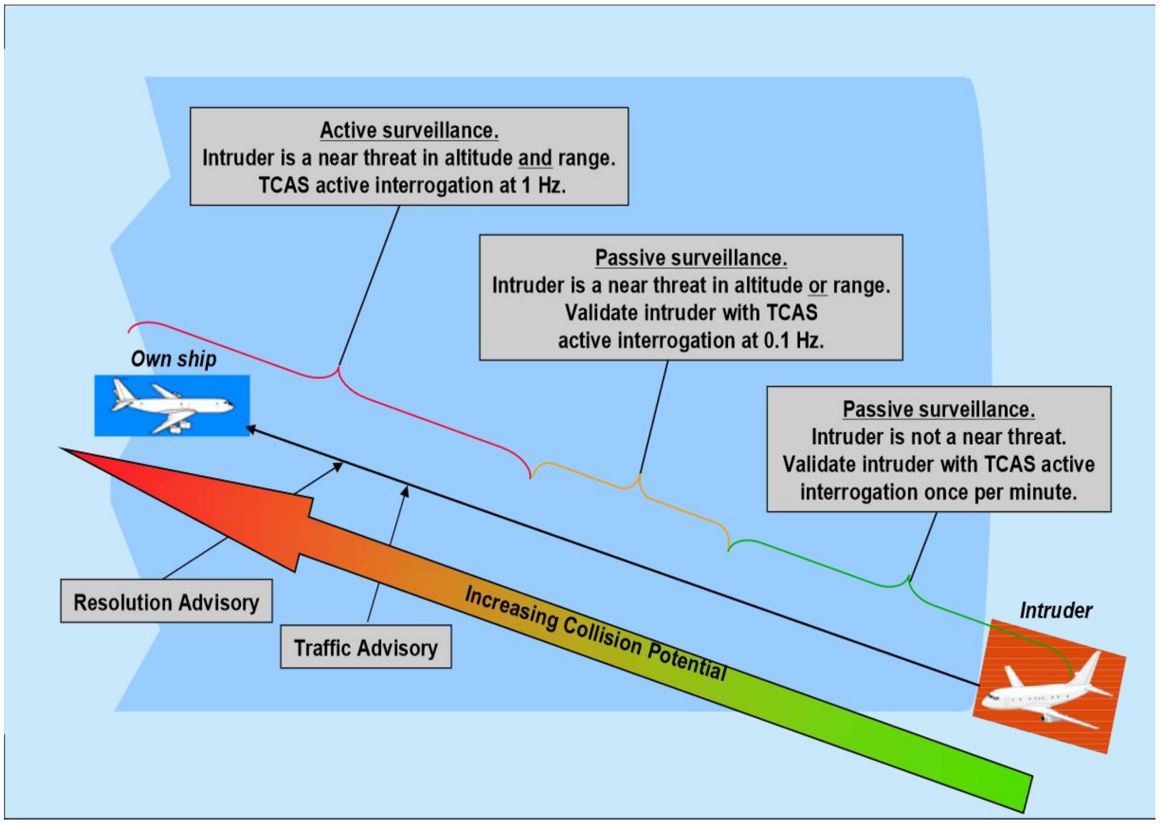

Active surveillance.
Intruder is a near threat in altitude and range.
TCAS active interrogation at 1 Hz.
Passive surveillance.
Intruder is a near threat in altitude or range.
Validate intruder with TCAS active interrogation at 0.1 Hz.
Passive surveillance.
Intruder is not a near threat.
Validate intruder with TCAS active interrogation once per minute.
Own ship
Intruder
Resolution Advisory
Traffic Advisory
Increasing Collision Potential

**Figure 1.2 TCAS Functionaliy[6]**

In the case of UAS and AUAS platforms, there is a need to develop systems like TCAS to integrate the aerial platforms into regulated airspace. TCAS requires a specific type of transponder to be installed on larger, higher-speed vehicles and can only detect other vehicles using this transponder.[6] This implementation presents a problem for vehicles that are smaller, slower, or are unable to equip this transponder. Many UAS are built with the purpose of being small, light, and low-cost so they are often limited in terms of what systems they are capable of mounting on-board the vehicle. UAS also have the benefit of being cost-effective alternatives to manned aircraft so finding low-cost ways for these vehicles to integrate into controlled airspace safely can help further research and utility ventures. Currently, research is being done to aid the development of a reliable system that will allow UAS pilots or AUAS systems to effectively detect and resolve conflicts. Recently, the FAA has published a final ruling on operating UAS in the NAS.[9] In this ruling, the FAA dictates that UAS must be piloted and must maintain Line of Sight (LOS) as well as give right of way to any other vehicles. The ruling makes no mention of AUAS, which is understandable as research into these vehicles is still ongoing and likely will not see certification for some time. The ruling makes comments on the size and weight constraints of UAS and possible solutions. One comment highlights the fact that larger aircraft could make use of the TCAS system while another comment suggests that for smaller aircraft, the large, heavy transponders could be placed near the remote pilots to remove the size and weight requirements that these transponders would need. These rules appear all acceptable for piloted vehicles operating within LOS, however future research will most likely be focused on AUAS operating outside of LOS.

To begin to develop the CDR algorithm for this thesis, it is important to review the literature on past CDR algorithms and use them to influence building the current tool. The very first item to review is the appropriate level of airframe dynamic modeling needed for CDR development. Erzberger provides a good starting point for developing Equations of Motion (EOM) for an aircraft.[10] While not fully consistent with the specific vehicle model this thesis will be using, the Erzberger model does touch on using a velocity frame instead of an inertial frame for the model formulation. His equations are focused on a rigid body aircraft moving in two dimensions while the vehicle used in this thesis is a point-mass particle moving in three dimensions. Erzberger also gives a description of the Runge-Kutta fourth order numerical integration method used in propagating the equations of motion, which is utilized in this thesis. A later paper by Slattery and Zhao[11] develops EOM for a Six Degree of Freedom (6DOF), rigid body aircraft. This thesis helps move the EOM into three dimensions from the information given in Erzberger. Texts by Vinh[12] and by Hull[13] provide a rigorous derivation of the Three Degree of Freedom (3DOF) point-mass dynamic aircraft model. The simplified aircraft model used in the thesis is based on this type of model along with an assumption of a high-authority, high-accuracy control system.

A few papers provide tools and ideas on building and implementing a conflict detection and resolution algorithm. A paper by Isaacson and Erzberger[5] on developing a CDR algorithm for the TRACON system discusses using four-dimensional (4D) trajectories as the basis for all calculations. This approach is incredibly useful as these data are easy to obtain from ground-based systems. These data can then easily be converted into the aircraft's state vector. Erzberger also discusses the efficiency required for an effective algorithm. Performing too many calculations at once can slow down the processing of the algorithm and the CDR tool may no longer represent a real-time environment. This factor is not as much of a concern for this thesis as computational processing power has increased considerably since the paper was written in 1997. This thesis also does not strive to create a simulation that is effective in real-time. In reality, processing power is always a concern and the number of calculations should be limited. However with current technology, there is less of a concern now. A paper by Merz[14] discusses a maximum-miss approach. While this thesis is more concerned with a minimum-miss strategy, Merz does offer many useful approaches to CDR. Merz makes heavy use of a relative formulation of the conflict vehicle with respect to the target vehicle. He also uses a simplified spherical envelope around the target vehicle instead of the more complicated cylindrical envelopes defined by the FAA. The basis for the CDR algorithm comes from a paper by Krozel.[15] In this paper, Krozel develops a geometric approach for solving for the miss vector and miss distance as well as the time to miss. Krozel uses a cylindrical safe zone instead of the spherical zone this thesis will be using. A paper by Park[16] provides a procedure for resolving the conflict detected by the process used by Krozel.

This procedure utilizes a geometric approach, much like the one this thesis focuses on. Park's approach makes use of two vehicles that are aware of each other and are capable of sharing information in such a way that each vehicle can assume part of the responsibility of resolving the conflict. Each vehicle is assigned a direction unit vector based on the current speed and heading of the vehicle. This unit vector is then used to plot a course that will steer each vehicle in such a way that the PCA no longer conflicts with any FAA guidelines.

## 1.3 Problem Statement

The purpose of this thesis is to develop a PCA-based CDR algorithm to prevent air traffic conflict events and evaluate the performance of the algorithm using a simulator. Some CDR algorithms that use a PCA approach assume that both vehicles are aware of each other, are both using the same algorithm, and are both willing to aid in the conflict resolution. Until communication and detection become standard between both piloted aircraft and AUAS, this assumption is unrealistic. This research assumes all of the CDR functioning is implemented in the target vehicle and removed from the conflict vehicle, although the conflict vehicle can still be maneuvering. Additionally, the ideal case where the target has complete and perfect knowledge of the conflict aircraft state vector is assumed. Another assumption being made in this thesis is that the vehicle being studied can only move with 3DOF in translation. While a 6DOF translational and rotational vehicle model is more realistic, it offers too many variables that must be accounted for at this early stage of CDR research. The six EOM are assumed to be first order, nonlinear ordinary differential equations (ODE) for the kinetics and kinematics of translation. However, velocity control assumptions lead to linear kinetic equations.

The objectives of the thesis are to:

1. develop a working CDR algorithm that can solve the three-dimensional conflict problem in a general setting with practical implementation,
2. develop a closed-loop aircraft simulation with traffic implementing the CDR algorithm, and
3. assess the effectiveness of the designed CDR algorithm in various scenarios to generally aid in research that will eventually integrate UAS and AUAS into FAA controlled airspace.

Developing a CDR algorithm that can be simulated and tested can be used in further research to develop guidance and control logic for not only UAS, but also to refine current logic in manned vehicles.[17] Developing a simulator to test and evaluate control logic increases the level of safety and confidence that researchers can attain prior to performing full-scale tests in congested airspace that can be both high-risk and high-cost, or sub-scale tests in intermediate development steps.[18] An important high level contribution of this thesis will be to provide methodology for evaluating overall CDR concepts to see if it is useful for future studies involving integrating AUAS into FAA airspace.

## 1.4 Thesis Outline

In Chapter 2, the equations of motion are derived and the reference frames that are used are presented. Aircraft state variables are also defined and the process of transitioning from the open-loop kinetic equations to an assumed form of the corresponding closed-loop equivalent is offered. In Chapter 3, the overview of the kinematic control scheme is given with the type of control being used and the format of the controller being presented. The CDR algorithm and the development of both the conflict detection and the conflict resolution strategies are described. The resolution method relies on solving three simultaneous equations for three unknowns and the use of a Newton-Raphson solver is implemented. A baseline 4D path following control system is also developed. Chapter 4 describes the simulator and both the open-loop and closed-loop simulators are presented. MathWorks Simulink is used to create the simulator and MathWorks MATLAB is used for pre- and post-processing the vehicle trajectories. Chapter 5 presents the completed and assembled simulator and results of various simulation runs to test the limits of the algorithm and simulator. Chapter 6 provides final conclusions and suggestions for future works.

# CHAPTER 2

# DEVELOPMENT OF EQUATIONS OF MOTION

## 2.1 Position and Velocity Descriptions

In this chapter, the equations of motion used to simulate the trajectory of the vehicle will be presented. The vehicle will be considered as a point-mass object with three degrees of translational freedom: $X, Y$, and $Z$ positions in the inertial frame. Figure 2.1 illustrates this setting where A denotes the aircraft and axes $XYZ$ indicate the inertial frame. Aircraft position vector $\vec{R}_A$ is given by

$$\vec{R}_A = X_A\hat{I} + Y_A\hat{J} + Z_A\hat{K} \tag{2.1}$$

where $\hat{I}, \hat{J}, \hat{K}$ denote unit vectors for the inertial frame. Note this inertial frame is somewhat non-standard with $Z$ pointing up away from the flat Earth model, lying in the $XY$ plane. Axes $X, Y$, and $Z$ here correspond to the North, West, and Up directions. Figure 2.1 also shows the aircraft velocity vector, $\vec{V}_A$. For convenience, consider the vehicle's velocity state in both an inertial North, West, Up frame $(\dot{X}, \dot{Y}, \dot{Z})$ and in a frame aligned with the direction of the velocity. This frame is known as the velocity frame because it is pinned to the vehicle with the $x_V$ axis always pointing along the velocity vector. The velocity frame is obtained from the inertial frame by a sequence of two rotations; first a rotation of angle $\chi$ about the $Z$ axis and then a second rotation of $\gamma$ about the $-y_V$ axis. The velocity frame is also shown in Figure 2.1 with axes $x_V, y_V, z_V$ where $y_V$ is always horizontal or parallel to the Earth's surface. Angle $\chi$ is the heading angle and $\gamma$ is the flight path or climb angle. The vehicle's velocity in this new frame is described by $V, \chi, \gamma$ where $V$ denotes the magnitude of the velocity. The velocity vector is expressed as

$$\vec{V}_A = \dot{X}_A \hat{I} + \dot{Y}_A \hat{J} + \dot{Z}_A \hat{K} \tag{2.2a}$$

$$\vec{V}_A = V \hat{\imath}_V \tag{2.2b}$$

where $\hat{\imath}_V$ is the unit vector along $x_V$.

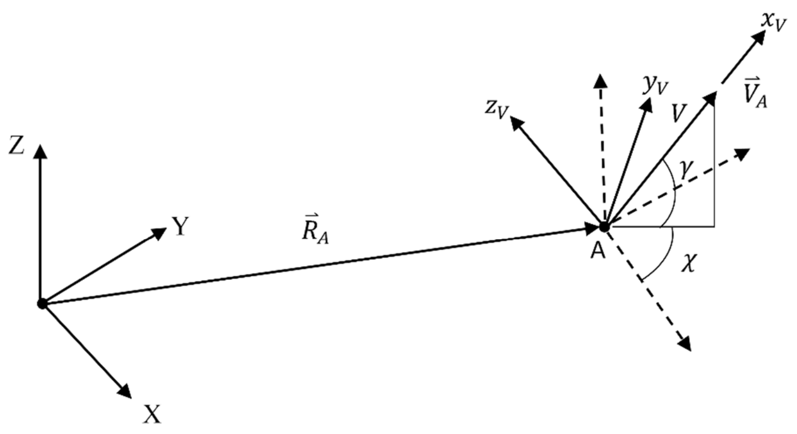


**Figure 2.1 Visualization of Inertial and Velocity Frames**

Inertial coordinates are useful when following a path given to the aircraft and the velocity frame is convenient because it is attached to the vehicle and oriented in the direction the vehicle is traveling. The transformation from the inertial frame to the velocity frame is done using an Euler angle rotation sequence with a Body 3-2-1 set.[19] A rotation from the inertial frame to an intermediate frame through the heading angle, χ, is shown by

$$\begin{bmatrix} \hat{\imath}_1 \\ \hat{\jmath}_1 \\ \hat{k}_1 \end{bmatrix} = \begin{bmatrix} C_\chi & S_\chi & 0 \\ -S_\chi & C_\chi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{I} \\ \hat{J} \\ \hat{K} \end{bmatrix} \tag{2.3a}$$

where $C_\chi$ is $\cos\chi$ , $S_\chi$ is $\sin\chi$, and $\hat{\imath}_1, \hat{\jmath}_1$, and $\hat{k}_1$ are the unit vectors in the intermediate frame. From there, another rotation through the climb angle, $\gamma$, is given by

$$\begin{bmatrix} \hat{\imath}_V \\ \hat{\jmath}_V \\ \hat{k}_V \end{bmatrix} = \begin{bmatrix} C_\gamma & 0 & S_\gamma \\ 0 & 1 & 0 \\ -S_\gamma & 0 & C_\gamma \end{bmatrix} \begin{bmatrix} \hat{\imath}_1 \\ \hat{\jmath}_1 \\ \hat{k}_1 \end{bmatrix} \tag{2.3b}$$

where $C_\gamma$ is $\cos\gamma$ , $S_\gamma$ is $\sin\gamma$ and $\hat{\imath}_V, \hat{\jmath}_V$, and $\hat{k}_V$ are the unit vectors in the velocity frame. Note this rotation is about the negative intermediate $y$ axis. The third rotation is 0 as there are only two angles to rotate through. An overall transformation matrix is created by multiplying both rotation matrices together to form

$$\begin{bmatrix} \hat{\imath}_V \\ \hat{\jmath}_V \\ \hat{k}_V \end{bmatrix} = \begin{bmatrix} C_\chi C_\gamma & S_\chi C_\gamma & S_\gamma \\ -S_\chi & C_\gamma & 0 \\ -C_\chi S_\gamma & -S_\chi S_\gamma & C_\gamma \end{bmatrix} \begin{bmatrix} \hat{I} \\ \hat{J} \\ \hat{K} \end{bmatrix} \tag{2.4}$$

This transformation matrix has the unique property where its inverse is equal to its transpose because the matrix is orthogonal with rows and columns of unit length. This property allows quick and efficient transformations between the inertial and the velocity frames, regardless of direction. This property becomes useful when a closed-loop control system is implemented.

## 2.2 Developing Kinetic Equations of Motion

Developing the equations of motion for the vehicle starts by analyzing the 3DOF model presented by Vinh.[12] Vinh's six EOM are the kinematic formulas defined in a spherical earth

$$\frac{dr}{dt} = V\sin\gamma$$

$$\frac{d\theta}{dt} = \frac{V\cos\gamma\cos\chi}{r\cos\phi} \tag{2.5}$$

$$\frac{d\phi}{dt} = \frac{V \cos\gamma \sin\chi}{r}$$

where $r$, $\theta$, and $\phi$ are polar position coordinates in an inertial frame, similar to the velocity frame described above, and the kinetic formulas

$$\frac{dV}{dt} = \frac{F_A}{m} - g\sin\gamma$$

$$V\frac{d\gamma}{dt} = \frac{F_N}{m}\cos\sigma - g\cos\gamma + \frac{V^2}{r}\cos\gamma \tag{2.6}$$

$$V\frac{d\chi}{dt} = \frac{F_N}{m}\frac{\sin\sigma}{\cos\gamma} - \frac{V^2}{r}\cos\gamma\cos\chi\tan\phi$$

where the axial force $F_A$ is $F_A = T\cos\epsilon - D$, the normal force $F_N$ is $F_N = T\sin\epsilon + L$, $T$ is the thrust of the aircraft, $D$ is the total aerodynamic drag on the aircraft, $L$ is the amount of lift, $\epsilon$ is the thrust pitch angle, $m$ is the aircraft's mass, $g$ is the acceleration due to gravity, and σ is the bank angle of the aircraft. Note that $D$ and $L$ are functions of $V$ and angle of attack, $\alpha$, while $T$ is a function of $V$ and throttle $\delta_{Th}$. For 3DOF models, the angle of attack, bank angle, and throttle serve as control inputs. Right away, a few assumptions about these equations can be made. The first assumption that Vinh makes is that the Earth can be represented as flat. This assumption results in a uniform gravitational field allowing for a constant $g$. Following the same simplification process that Vinh uses regarding flight speed being much less than the minimum circular orbital speed results in being able to neglect the $\frac{V^2}{r}$ terms. The kinetic equations can be restated as

$$\dot{V} = \frac{F_A}{m} - g\sin\gamma$$

$$V\dot{\gamma} = \frac{F_N}{m}\cos\sigma - g\cos\gamma \tag{2.7}$$

$$V\dot{\chi} = \frac{F_N}{m}\frac{\sin\sigma}{\cos\gamma}$$

## 2.3 Feedback Linearizing Kinetic Equations of Motion

To create a 4D path following outer closed-loop system, it is necessary to have a high-authority inner closed-loop system controlling the velocity states. In manual control, the human pilot would manipulate angle of attack, bank angle, and throttle to achieve this control. Here an automatic feedback linearizing controller is outlined to achieve this functionality with the added benefit of generating a linear closed-loop kinetics model of the velocity dynamics.

For this formulation, the equation for $\dot{V}$ will be analyzed first. To directly affect the forward velocity, the most efficient way is to change the amount of thrust coming from the vehicle. Assuming that the vehicle does not have a propulsion system capable of changing the thrust pitch angle, $\epsilon$, and that the control system is sufficiently powerful enough to compensate for the drag on the vehicle, the thrust term $T$ can be rewritten as a constant multiplied by a change in throttle.

$$T = C_V\delta_{Th} \tag{2.8}$$

Input $\delta_{Th}$ can be defined from a controls standpoint as the difference in the commanded velocity value $V_C$ and the current velocity value $V$ multiplied by a control gain and, with accurate enough sensors to detect the current flight path angle and variables needed to construct the drag value, a way to compensate for the gravity and drag terms, is given by

$$\delta_{Th} = \kappa_V(V_c - V) + \frac{m}{C_V}\frac{1}{\cos\epsilon}\left(\frac{D}{m} + g\sin\gamma\right) \tag{2.9}$$

Note drag is given by the equation $D = \frac{1}{2}\rho V^2 A C_D(\alpha, M, Re)$ where $\rho$ is the density of the air, $V$ is the velocity, $A$ is the area of the wing, and $C_D$ is the drag coefficient. $C_D$ is a complex term that is a function of the angle of attack $\alpha$, the Mach number $M$, and the Reynolds number $Re$. Inserting this term back into the $\dot{V}$ equation yields

$$\dot{V} = \frac{C_V\left(\kappa_V(V_c - V) + \frac{m}{C_V}\frac{1}{\cos\epsilon}\left(\frac{D}{m} + g\sin\gamma\right)\right)\cos\epsilon - D}{m} - g\sin\gamma \qquad (2.10)$$

leaving the closed-loop velocity equation of motion as

$$\dot{V} = \frac{C_V\kappa_V\cos\epsilon\,(V_c - V)}{m} \qquad (2.11)$$

In Equation (2.11), $C_V$ is a constant, $\kappa_V$ is a control gain that can be set by the engineer designing the system, $m$ is the mass of the vehicle, and $\epsilon$ is the thrust angle. These four constants can be collected into one constant that can represent the size and class of aircraft that is being studied. The term $\frac{C_V\kappa_V\cos\epsilon}{m}$ can be inverted and redefined as a time delay constant, $\tau_V$. As this time delay value gets larger, the dynamic velocity response becomes slower which is interpreted as approximating a larger, heavier aircraft that cannot maneuver very quickly. The final linear, inner closed-loop equation of motion for $\dot{V}$ is

$$\dot{V} = (V_c - V)\frac{1}{\tau_v} \qquad (2.12)$$

where $\tau_v$ is the specific value to describe the resistance of the vehicle to change its velocity. The $\chi$ and $\gamma$ have similar specific $\tau$ values.

Deriving the inner closed-loop derivatives for $\chi$ and $\gamma$ are not as straightforward as the velocity term is. The lift term present in $F_N$ is given by the equation $L = \frac{1}{2}\rho V^2 A C_L(\alpha, M, Re)$ where $C_L$ is the lift coefficient. $C_L$ is a complex term that is also a function of $\alpha, M,$ and $Re$. One way to change the flight path and heading angles is through affecting the angle of attack and the bank angle of the aircraft. To execute a change in either $\chi$ or $\gamma$, the required $\alpha$ and $\sigma$ need to be calculated.

The two equations for $\dot{\chi}$ and $\dot{\gamma}$ can be rewritten as

$$\dot{\chi} = \frac{T\sin\epsilon + \frac{1}{2}\rho V^2 A C_L(\alpha, M, Re)}{Vm}\frac{\sin\sigma}{\cos\gamma} \tag{2.13}$$

and

$$\dot{\gamma} = \frac{T\sin\epsilon + \frac{1}{2}\rho V^2 A C_L(\alpha, M, Re)}{Vm}\cos\sigma - \frac{g\cos\gamma}{V} \tag{2.14}$$

All terms besides $\alpha$ and $\sigma$ can be considered known values. Equations (2.13) and (2.14) require the use of a numerical algorithm to simultaneously solve for the control values. Focusing just on the $\dot{\chi}$ equation,

$$\dot{\chi} = \frac{T\sin\epsilon}{Vm}\frac{\sin\sigma}{\cos\gamma} + \frac{\frac{1}{2}\rho V^2 A C_L}{Vm}\frac{\sin\sigma}{\cos\gamma} \tag{2.15}$$

a new control input term can be developed that isolates the combined lift and bank angle term in a way done similarly to $\dot{V}$. Let $C_\chi = \frac{\frac{1}{2}\rho V A}{\cos\gamma}$, and define the new input term as $\delta_{C_L S_\sigma} = C_L \sin\sigma$ leading to a new form of the $\dot{\chi}$ equation, or

$$\dot{\chi} = \frac{T \sin \epsilon}{Vm} \frac{\sin \sigma}{\cos \gamma} + \frac{C_\chi}{m} \delta_{C_L S_\sigma} \tag{2.16}$$

The control law for $\delta_{C_L S_\sigma}$ is defined from a difference term involving the commanded and current heading values with a multiplier control gain $\kappa_\chi$ summed with a thrust term.

$$\delta_{C_L S_\sigma} = \kappa_\chi(\chi_c - \chi) - \frac{1}{C_\chi} \frac{T \sin \epsilon}{V} \frac{\sin \sigma}{\cos \gamma} \tag{2.17}$$

This control leads to a linear, inner closed-loop heading equation of motion as

$$\dot{\chi} = (\chi_C - \chi) \frac{1}{\tau_\chi} \tag{2.18}$$

where the time constant $\tau_\chi = \frac{m}{C_\chi \kappa_\chi}$. Manipulation of $\dot{\gamma}$ follows a similar process. Letting $C_\gamma = \frac{1}{2}\rho V A$ and defining the new control input $\delta_{C_L C_\sigma} = C_L \cos \sigma$, the new form of the $\dot{\gamma}$ equation is

$$\dot{\gamma} = \frac{T \sin \epsilon}{Vm} \cos \sigma + \frac{C_\gamma}{m} \delta_{C_L S_\sigma} - \frac{g \cos \gamma}{V} \tag{2.19}$$

The control logic for $\delta_{C_L S_\sigma}$ is chosen as

$$\delta_{C_L C_\sigma} = \kappa_\gamma(\gamma_c - \gamma) - \frac{T \sin \epsilon \cos \sigma}{C_\gamma V} + \frac{m}{C_\gamma} \frac{g \cos \gamma}{V} \tag{2.20}$$

where $\gamma_C$ is the commanded flight path angle and $\kappa_\gamma$ is a control gain. Substitution for $\delta_{C_L C_\sigma}$ yields the linear, inner closed-loop flight path equation of motion

$$\dot{\gamma} = (\gamma_c - \gamma) \frac{1}{\tau_\gamma} \tag{2.21}$$

In Equation (2.21), the time constant $\tau_\gamma$ is $\frac{m}{C_\gamma \kappa_\gamma}$.

To implement this multivariable inner loop control law, values for $\delta_{Th}, \alpha, \sigma$ must be solved for numerically from a set of simultaneous equations. The coupling arises from drag (or $\alpha$) appearing in the $\dot{V}$ expression and T (or $\delta_{Th}$) appearing in the $\dot{\chi}$ and $\dot{\gamma}$ expressions, along with $L$ (or $\alpha$) and $\sigma$ appearing in both the $\dot{\chi}$ and $\dot{\gamma}$ expressions. The three equations involving $\delta_{Th}, \alpha, \sigma$ are

$$\begin{gathered}\frac{T(\delta_{Th})\cos\epsilon - D(\alpha)}{m} - g\sin\gamma = \frac{1}{\tau_V}(V_C - V)\\ \frac{T(\delta_{Th})\sin\epsilon + L(\alpha)}{Vm}\frac{\sin\sigma}{\cos\gamma} = \frac{1}{\tau_\chi}(\chi_C - \chi)\\ \frac{T(\delta_{Th})\sin\epsilon + L(\alpha)}{Vm}\cos\sigma - \frac{g\cos\gamma}{V} = \frac{1}{\tau_\gamma}(\gamma_C - \gamma)\end{gathered} \tag{2.22}$$

In summary, use of a feedback linearizing controller for augmenting the velocity states leads to a set of linear, inner closed-loop equations of motion for the velocity kinetics. These equations are

$$\begin{gathered}\dot{V} = (V_c - V)\frac{1}{\tau_V}\\ \dot{\chi} = (\chi_c - \chi)\frac{1}{\tau_\chi}\\ \dot{\gamma} = (\gamma_c - \gamma)\frac{1}{\tau_\gamma}\end{gathered} \tag{2.23}$$

The research does not actually implement this inner loop control law in the simulation tool, but rather starts with use of Equation (2.23) for the velocity kinetics. This viewpoint was chosen primarily to emphasize the CDR development effort and to simplify the model used in that effort. The control law described in detail in this section could be implemented, and this places the velocity kinetics model in in Equation (2.23) on a solid foundation.

## 2.4 Transforming Kinematic Equations of Motion

Obtaining the inertial velocities $\dot{X}, \dot{Y}, \dot{Z}$ can be accomplished by one of two ways. Vinh's kinematic equations can be transformed from spherical Earth centered inertial polar coordinates to flat Earth surface centered inertial Cartesian coordinates. In this approach, $r$ in Equation (2.4) is expressed as the sum of Earth radius $r_E$ and altitude $Z$ using the flat Earth axes coordinate for up.

$$r = r_E + Z \tag{2.24}$$

Since the Earth radius is constant and $Z$ is assumed to be small compared to $r_E$ for aircraft flight, expressions for $\dot{r}, \dot{\theta}, \dot{\phi}$ yield

$$\begin{gathered} \dot{Z} = V \sin\gamma \\ r_E \cos\phi\, \dot{\theta} = V \cos\gamma \cos\chi \\ r_E \dot{\phi} = V \cos\gamma \sin\chi \end{gathered} \tag{2.25}$$

With time rates of change of the longitude angle $\theta$ and latitude angle $\phi$, the flat Earth velocities along North, West, and Up directions are

$$\begin{gathered} \dot{X} = r_E \dot{\phi} = V \cos\gamma \sin\chi \\ \dot{Y} = -r_E \cos\phi\, \dot{\theta} = -V \cos\gamma \cos\chi \\ \dot{Z} = V \sin\gamma \end{gathered} \tag{2.26}$$

Vinh uses a heading angle measured from the East axis while this thesis uses a heading angle measured from the North axis. Thus, after conversion of angles, the velocity expressions become

$$\begin{gathered} \dot{X} = V \cos\chi \cos\gamma \\ \dot{Y} = V \sin\chi \cos\gamma \\ \dot{Z} = V \sin\gamma \end{gathered} \tag{2.27}$$

In the second approach, a flat Earth environment is assumed from the outset. From Figure 2.1, all that is required is to transform the magnitude of the velocity along the velocity frame $x_V$ axis by projecting this magnitude onto the inertial frame through angles $\chi$ and $\gamma$. The appropriate transformation matrix was given in Equation (2.4). The property of orthogonal matrices can be exploited here to invert the transformation matrix by taking the transpose. This operation results in the matrix equation

$$\begin{bmatrix}\dot{X}\\ \dot{Y}\\ \dot{Z}\end{bmatrix} = \begin{bmatrix} C_\chi C_\gamma & -S_\chi & -C_\chi S_\gamma \\ S_\chi C_\gamma & C_\gamma & -S_\chi S_\gamma \\ S_\gamma & 0 & C_\gamma \end{bmatrix} \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix} \tag{2.28}$$

Carrying out the multiplication results in the three velocities in the inertial frame presented below.

$$\dot{\mathrm{X}} = \mathrm{V}\cos\chi\cos\gamma$$

$$\dot{\mathrm{Y}} = \mathrm{V}\sin\chi\cos\gamma \tag{2.29}$$

$$\dot{\mathrm{Z}} = \mathrm{V}\sin\gamma$$

Comparison with the Vinh approach confirms either way leads to the same set of nonlinear kinematic equations of motion for position.

An ideal block diagram of the equations of motion developed in this chapter is represented by Figure 2.2 below. This system is referred to as the open-loop model, even though it is actually a closed-loop system as noted by the feedback loop in Figure 2.2 which describes the effect of the feedback linearizing controller. Commands for velocity, heading, and flight path are inputs to the system. The linear first order kinetics equations are integrated yielding time dependent responses for the velocity, heading, and flight path. The nonlinear first order kinematics equations are then integrated yielding time dependent responses for the Cartesian positions, which serve as the model output. This processing will be implemented in a digital computer. Further, this open-loop model is used to build up a 4-Dimensional Path Following (4PF) control system that will control the position of the vehicle based on input trajectory commands.

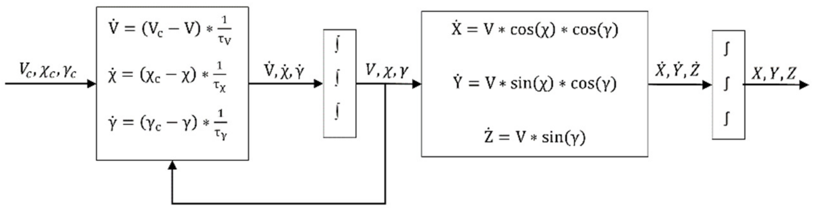

$V_c, \chi_c, \gamma_c$
$\dot{V} = (V_c - V) * \frac{1}{\tau_V}$
$\dot{\chi} = (\chi_c - \chi) * \frac{1}{\tau_\chi}$
$\dot{\gamma} = (\gamma_c - \gamma) * \frac{1}{\tau_\gamma}$
$\dot{V}, \dot{\chi}, \dot{\gamma}$
$\int$
$V, \chi, \gamma$
$\dot{X} = V * \cos(\chi) * \cos(\gamma)$
$\dot{Y} = V * \sin(\chi) * \cos(\gamma)$
$\dot{Z} = V * \sin(\gamma)$
$\dot{X}, \dot{Y}, \dot{Z}$
$X, Y, Z$

**Figure 2.2 Block Diagram of Open-Loop Model**

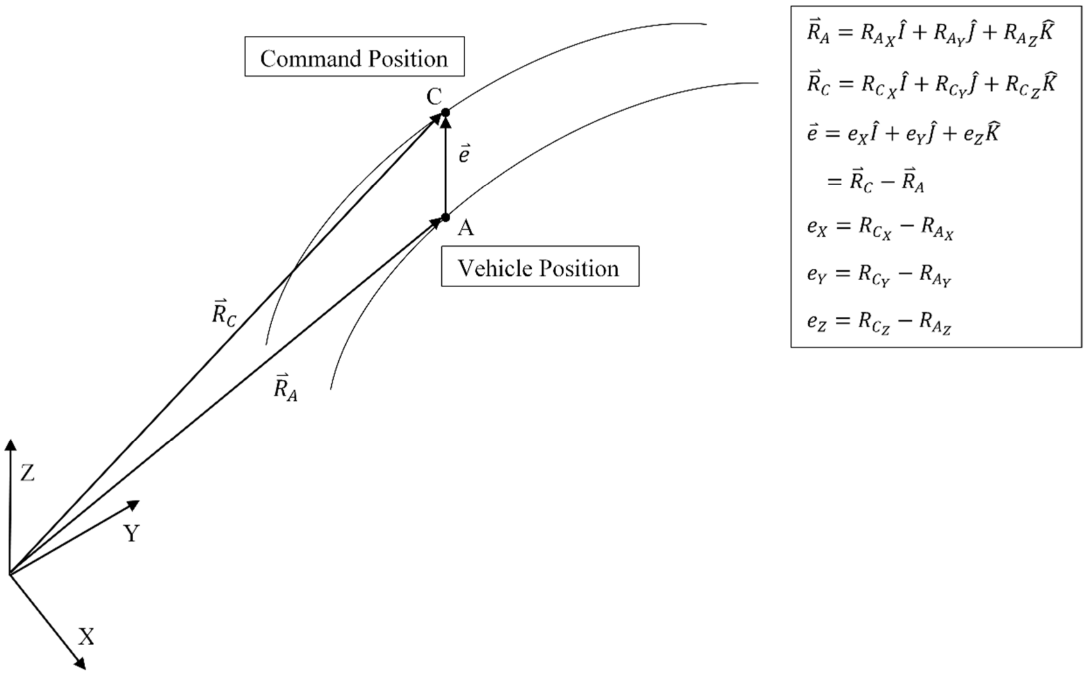
# CHAPTER 3

# OVERVIEW OF CONTROL SCHEME AND CDR ALGORITHM

## 3.1 Path Following Control Scheme

Development of a path following control system to achieve the nominal mission of the target vehicle is considered here. If the target vehicle was a manned vehicle or a remotely piloted UAS, the human pilot could provide this function. Here, the target vehicle is considered to be an AUAS where the mission is achieved autonomously by the path following control system. The method of control being used in this system will be negative feedback control. Negative feedback makes use of the difference between the command value and the current value to produce an error that can be acted on to drive the system to a steady state value that matches the command value.

Figure 3.1 shows how the three-dimensional position error signal is created and passed into the controller for processing. $\vec{R}_A$ is the position vector of the target vehicle in the $\hat{I}, \hat{J}, \hat{K}$ inertial frame (see Equation (2.1) and Figure 2.1), $\vec{R}_c$ is the commanded position vector of the vehicle, and $e_X, e_Y, e_Z$ are the values of the error between the commanded and actual position vectors in the respective inertial coordinates. Note the command positions are indexed by time, making this scheme a four-dimensional path following system.



Command Position
C
$\vec{e}$
A
Vehicle Position
$\vec{R}_C$
$\vec{R}_A$
Z
Y
X
$\vec{R}_A = R_{A_X}\hat{I} + R_{A_Y}\hat{J} + R_{A_Z}\hat{K}$
$\vec{R}_C = R_{C_X}\hat{I} + R_{C_Y}\hat{J} + R_{C_Z}\hat{K}$
$\vec{e} = e_X\hat{I} + e_Y\hat{J} + e_Z\hat{K}$
$= \vec{R}_C - \vec{R}_A$
$e_X = R_{C_X} - R_{A_X}$
$e_Y = R_{C_Y} - R_{A_Y}$
$e_Z = R_{C_Z} - R_{A_Z}$

**Figure 3.1 Visualization of Error between Commanded and Actual Positions**

Before being passed to the controller, it is more useful to transform the errors from the inertial frame into the velocity frame. To make this transformation, the matrix developed in Chapter 2 can be utilized and adapted to act on the error signals. The matrix equation for this transformation is given by

$$\begin{bmatrix} e_V \\ e_\chi \\ e_\gamma \end{bmatrix} = \begin{bmatrix} C_\chi C_\gamma & S_\chi C_\gamma & S_\gamma \\ -S_\chi & C_\gamma & 0 \\ -C_\chi S_\gamma & -S_\chi S_\gamma & C_\gamma \end{bmatrix} \begin{bmatrix} e_X \\ e_Y \\ e_Z \end{bmatrix} \tag{3.1}$$

where $e_V, e_\chi, e_\gamma$ are the errors in the $V, \chi, \gamma$ directions and $e_X, e_Y, e_Z$ are the errors in the $X, Y, Z$ directions.

The controller used will be Proportional-Integral-Derivative (PID) signal processing. PID control uses all three of the basic compensation strategies to provide an efficient, easy to use, and powerful controller that can augment the system in the desired way. Proportional control is a simple method that only uses a gain applied to the error to produce an output signal that is proportional to the error signal. Integral control works by multiplying a gain by the integral of the error signal. In a similar fashion, derivative control uses a gain multiplied by the derivative of the error to create an output signal. All three of these logics act on the error signal and allow the engineer to tune the response to the desired state. Changing the values for the proportional, integral, and derivative gains affect the system in different ways. Increasing the proportional and integral gains will help to decrease the rise time of the system while also increasing the amount that the system overshoots its desired value. To help counter the increase in overshoot, the derivative gain can be increased. However, increasing the derivative gain will increase the settling time and the rise time. Increasing the integral gain will eliminate the steady state error of the system.

The formula for a generic traditional PID controller is given by

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \dot{e}(t) \tag{3.2a}$$

in the time domain and

$$u(s) = \left(K_P + K_I \frac{1}{s} + K_D s\right) e(s) \tag{3.2b}$$

in the Laplace domain where $u$ is the input to the EOM, $e(t)$ and $e(\tau)$ are the instantaneous errors with $\tau$ being a dummy variable representing time, $s$ is the frequency, $K_P$ is the proportional control gain, $K_I$ is the integral control gain, and $K_D$ is the derivative control gain. It is more common to work with PID controllers in the Laplace domain and it can be shown that the equation for $u(s)$ can be rearranged to form

$$u(s) = \frac{K_D s^2 + K_P s + K_I}{s} e(s) \tag{3.3}$$

This transfer function structure presents a problem when realizing and implementing the controller into an actual system as the transfer function has more zeroes than it has poles. In a real system, the derivative is determined by using a finite difference method of subtracting the previous value from the current value and dividing by the step size between them. This process introduces lag into the derivative signal. In simulating a PID controller, a filter needs to be created that adds a pole into the system. The equation for the new PID controller is given by

$$u(s) = \left( K_P + \frac{K_I}{s} + K_D \frac{N}{1 + N\frac{1}{s}} \right) e(s) \tag{3.4}$$

where *N* is the filter coefficient specified by the engineer.

In the 4PF control system for the target vehicle, Equation (3.4) must be replicated three times for the three channels of the open-loop dynamic system. This expansion will introduce nine PID feedback gains and three filter coefficients that must be specified numerically. Equation (3.5) describes the governing control expressions for each channel.

$$V_C(s) = \left( K_{P_V} + \frac{K_{I_V}}{s} + K_{D_V} \frac{N_V}{1 + N_V\frac{1}{s}} \right) e_V(s)$$

$$\chi_C(s) = \left(K_{P_\chi} + \frac{K_{I_\chi}}{s} + K_{D_\chi}\frac{N_\chi}{1 + N_\chi \frac{1}{s}}\right) e_\chi(s) \tag{3.5}$$

$$\gamma_C(s) = \left(K_{P_\gamma} + \frac{K_{I_\gamma}}{s} + K_{D_\gamma}\frac{N_\gamma}{1 + N_\gamma \frac{1}{s}}\right) e_\gamma(s)$$

Note $V_C$ is used to primarily control position errors along the trajectory as $e_V$ represents the error component along the velocity frame $x_V$ axis which is always tangent to the trajectory. Also $\chi_C$ and $\gamma_C$ are used primarily to control errors normal to the trajectory (errors along the velocity frame $y_V$ and $z_V$ axes). Since axis $y_V$ is always horizontal, $\chi_C$ is used to eliminate azimuth positional errors, while $\gamma_C$ is used to eliminate elevation position errors.

Figure 3.2 shows a block diagram of the closed-loop system with the 4PF block integrated.

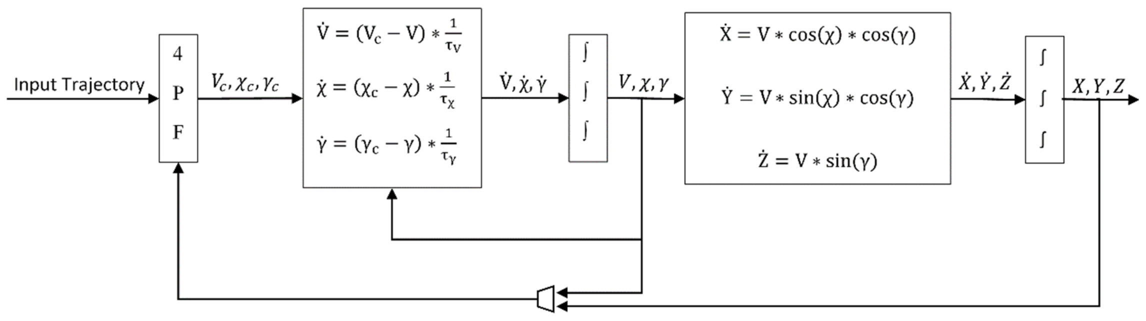

Input Trajectory
4 P F
$V_c, \chi_c, \gamma_c$
$\dot{V} = (V_c - V) * \frac{1}{\tau_V}$
$\dot{\chi} = (\chi_c - \chi) * \frac{1}{\tau_\chi}$
$\dot{\gamma} = (\gamma_c - \gamma) * \frac{1}{\tau_\gamma}$
$\dot{V}, \dot{\chi}, \dot{\gamma}$
∫ ∫ ∫
$V, \chi, \gamma$
$\dot{X} = V * \cos(\chi) * \cos(\gamma)$
$\dot{Y} = V * \sin(\chi) * \cos(\gamma)$
$\dot{Z} = V * \sin(\gamma)$
$\dot{X}, \dot{Y}, \dot{Z}$
∫ ∫ ∫
$X, Y, Z$

**Figure 3.2 Block Diagram of Closed-Loop Model**

## 3.2 Relative Equations of Motion Formulation

In the next section, the CDR control algorithm will be developed, which is based on a relative motion description between the target and conflict vehicles. Thus, this section will present the relative formulation of the EOM.[20] With respect to the inertial frame, the absolute equations for the position and velocity vectors of both the target vehicle and conflict vehicle are given by

$$\begin{aligned} \vec{R}_A &= X_A\hat{I} + Y_A\hat{J} + Z_A\hat{K} \\ \vec{R}_B &= X_B\hat{I} + Y_B\hat{J} + Z_B\hat{K} \\ \vec{V}_A &= \dot{X}_A\hat{I} + \dot{Y}_A\hat{J} + \dot{Z}_A\hat{K} \\ \vec{V}_B &= \dot{X}_B\hat{I} + \dot{Y}_B\hat{J} + \dot{Z}_B\hat{K} \end{aligned} \tag{3.6}$$

where A is the target vehicle and B is the conflict vehicle. $\vec{R}_A$ and $\vec{R}_B$ are the position vectors, and $\vec{V}_A$ and $\vec{V}_B$ is the velocity vector. $\hat{I}, \hat{J}, \hat{K}$ are the unit vectors in the $X, Y,$ and $Z$ directions respectively. The relative positions and velocities of the conflict vehicle with respect to the target vehicle are given by

$$\begin{aligned} \vec{r}_r &= \vec{R}_B - \vec{R}_A \\ \vec{V}_r &= \vec{V}_B - \vec{V}_A \end{aligned} \tag{3.7}$$

where $\vec{r}_r$ is the relative position vector and $\vec{V}_r$is the relative velocity vector. These vectors are shown in Figure 3.3 and describe what an observer in A would see regarding the location and movement of B.

## 3.3 Formulation of CDR Algorithm

This section addresses the thesis research core. Logic to perform conflict detection and conflict resolution based on the concept of point of closest approach is developed here. CDR logic is built around two important quantities: the miss distance and the time to miss. Using the relative motion state, these two quantities are computed and compared against critical values to detect conditions that could potentially lead to a collision, and if this is the case, to resolve new conditions that will prevent a collision. The conflict detection component is always active and is updated at every compute cycle. The conflict resolution component is only activated when collision potential is detected. When a potential collision does exist, a new commanded trajectory that eliminates the conflict is computed and used to steer the aircraft. After the collision is eliminated, the original mission command trajectory again steers the aircraft.

The miss distance vector $\vec{r}_m$ will be defined from a triple vector product as

$$\vec{r}_m = \hat{\epsilon}_{V_r} \times \left(\vec{r}_r \times \hat{\epsilon}_{V_r}\right) \tag{3.8}$$

In Equation (3.8), the unit vector $\hat{\epsilon}_{V_r}$ corresponds to the relative velocity vector $\vec{V}_r$. The miss distance vector locates the point of closest approach of the conflict aircraft (B) relative to the target aircraft (A), assuming both vehicles were to continue to fly along rectilinear trajectories based on their current state. Figure 3.3 shows the miss vector and PCA geometry and how they relate to the absolute and relative positions and velocities discussed in the previous section. Vector $\vec{r}_m$ determines where the two vehicles will be at their closest relative distance (i.e., the PCA). This distance is a very important quantity because it will alert the algorithm if there will be a conflict.

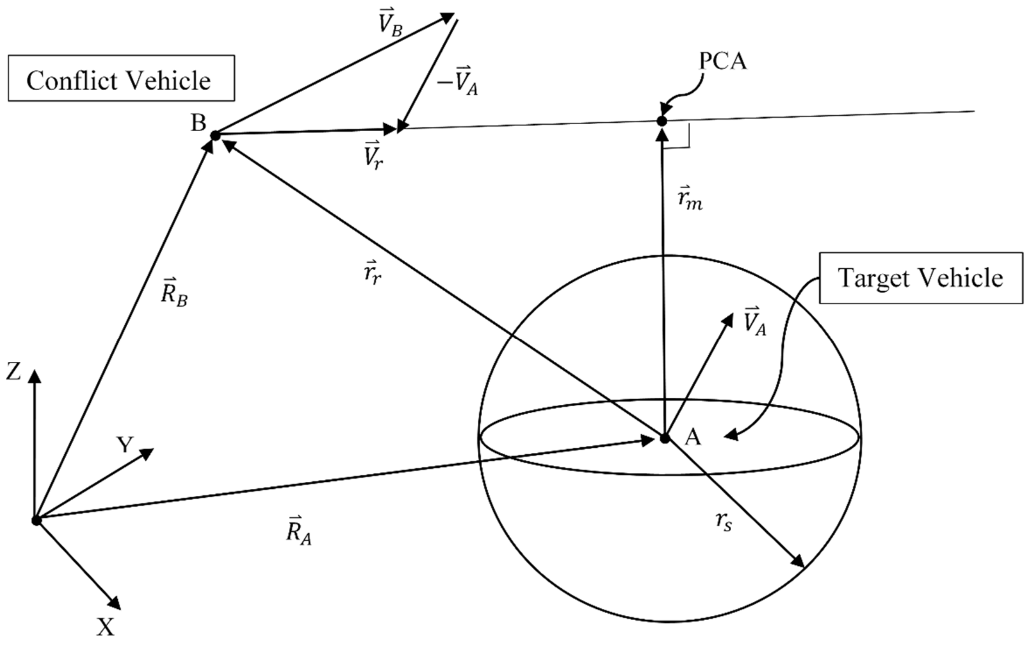


**Figure 3.3 Visualization of PCA**

The time at which this critical point will be reached is $\tau_m$. The equation for the miss time is given by a ratio of dot products, or

$$\tau_m = -\frac{\vec{r}_r \cdot \vec{V}_r}{\vec{V}_r \cdot \vec{V}_r} \tag{3.9}$$

This miss time can be thought of as a countdown of how long the vehicles have until the conflict point is reached. Again noting the geometry in Figure 3.3, the case where aircraft B has not yet reached the PCA corresponds to a positive value of $\tau_m$ (i.e. the PCA lies in the future). A negative value of $\tau_m$ means aircraft B has moved beyond the PCA (which lies in the past). Time to miss will generally decrease from positive values through zero to negative values, though not necessarily in a linear manner, as regular time advances. This time is also a very important quantity because it will alert the algorithm to a potential collision.

The CDR algorithm starts by computing $\tau_m$ and comparing this value with zero. If $\tau_m > 0$, then no action is taken since a conflict cannot exist. If $\tau_m < 0$, then a conflict may exist and further logic is processed. The CDR algorithm computes $\vec{r}_m$ and its magnitude, $r_m = \|\vec{r}_m\|$. A safety assurance standard or safe distance, $r_s$, is defined to be the radius of a sphere around the conflict vehicle, B, that can be set to approximate the required separation distance regulated by the FAA between aircraft of varying size and weight classes or any other distance that the researcher is interested in. The CDR algorithm then checks to see if a conflict actually exists indicated by the condition where $r_s$ is greater than $r_m$, or when the PCA lies inside the safe sphere. Figure 3.3 shows the case where no conflict exists ($r_s < r_m$) and Figure 3.4 shows the case when an actual conflict exists ($r_s > r_m$).

When a conflict has been detected, the CDR algorithm will turn on the conflict resolution portion to calculate and plot a course to resolve the conflict. Algorithm logic assumes that the conflict vehicle, vehicle B, is not aware of the conflict or is unwilling or unable to aid in resolving it. Vector $\hat{e}_{V_r'}$ is defined as the new relative velocity unit vector that points in the direction of the conflict resolution point. Ideally, this point will be just touching the safe sphere and in a somewhat similar direction to the current $\hat{e}_{V_r}$ to minimize control system maneuver input level. There are an infinite number of points on the sphere that will resolve the conflict, but there is only one that is most efficient for the vehicle to move to. To solve for the three components of $\hat{e}_{V_r'}$, a series of three equations were developed to describe the conflict resolution point (CRP). The first equation forces the new relative velocity vector to be orthogonal to the cross product of the current relative position vector and the current relative velocity vector and is given by

$$(\vec{r}_r \times \hat{e}_{V_r}) \cdot \hat{e}_{V_r'} = 0 \tag{3.10}$$

This equation requires the new velocity unit vector to lie in the plane containing $\vec{r}_r$ and $\vec{V}_r$. The second equation forces the new velocity unit vector to have a unit length, or

$$\left\|\hat{\epsilon}_{V_r'}\right\| = 1 \tag{3.11}$$

The third equation replicates the equation for the miss distance vector and forces the distance to be equivalent to the radius of the safe sphere and is shown to be

$$\left\|\hat{\epsilon}_{V_r'} \times \left(\vec{r}_r \times \hat{\epsilon}_{V_r'}\right)\right\| = r_s \tag{3.12}$$

This equation causes the CRP to lie on the surface of the safe sphere. The geometry behind these equations can be visualized with Figure 3.4.

After the new relative velocity unit vector is computed, it must be translated into a velocity vector and new command trajectory that the vehicle can follow. This process is achieved by freezing the moment in time at the current $\tau_m$ value, computing the new velocity vector $\vec{V}_R$ and constructing a trajectory from that. The equation used is

$$\vec{V}_r' = -\frac{\vec{r}_r \cdot \hat{\epsilon}_{V_r'}}{\tau_m}\hat{\epsilon}_{V_r'} \tag{3.13}$$

In Figure 3.4, vehicle B will now reach the CRP in $\tau_m$ units of time. This requires vehicle A to not only alter its heading and flight path, but also accelerate or decelerate to achieve the timing condition from Equation (3.13). Figure 3.4 shows how this new $\vec{V}_r'$ is used to create the modified $\vec{V}_A'$ that is used to steer and accelerate the target vehicle away from the conflict. In the relative perspective, the original path becomes the modified path which passes through the CRP.

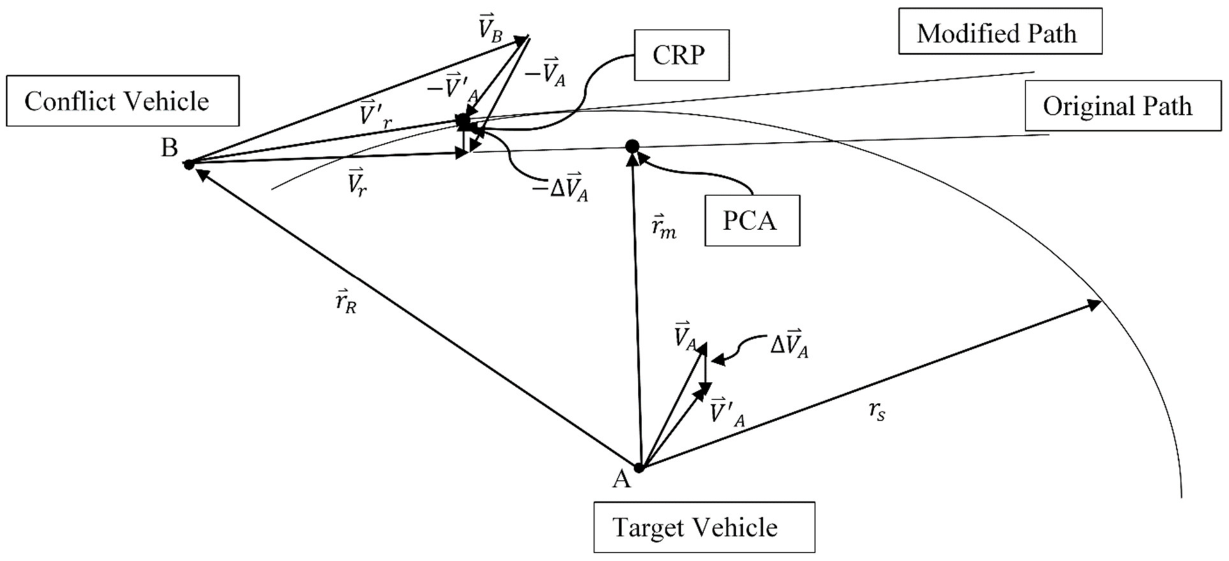


**Figure 3.4 Visualization of How CRP is Used to Steer Vehicle**

Rather than controlling aircraft A by velocity steering based on $\vec{V}_r'$ and $\vec{V}_A'$, $\vec{V}_r'$ is used to create a modified command trajectory that is fed into the existing 4PF control system from Section 3.1. A rectilinear path from B to the CRP is created with the boundary conditions that $\vec{r}_r$ locates B at time $t$ and $\vec{r}_r$ locates CRP at time $t + \tau_m$. Figure 3.5 shows the block diagram of the closed-loop model with CDR logic included. Note how the switch is used to command the nominal trajectory when no conflict is detected and to command the modified trajectory when a conflict is detected and requires resolution.

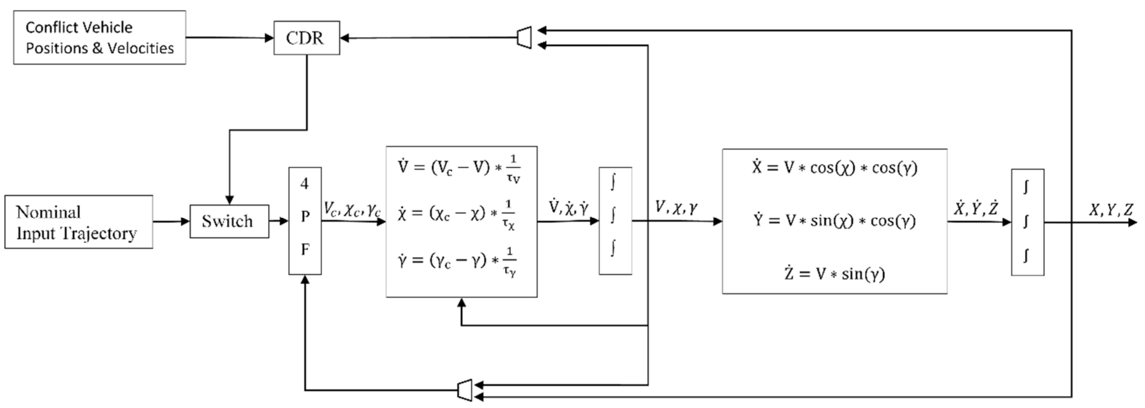

Conflict Vehicle Positions & Velocities
CDR
Nominal Input Trajectory
Switch
4 P F
$V_c, \chi_c, \gamma_c$
$\dot{V} = (V_c - V) * \frac{1}{\tau_V}$
$\dot{\chi} = (\chi_c - \chi) * \frac{1}{\tau_\chi}$
$\dot{\gamma} = (\gamma_c - \gamma) * \frac{1}{\tau_\gamma}$
$\dot{V}, \dot{\chi}, \dot{\gamma}$
$V, \chi, \gamma$
$\dot{X} = V * \cos(\chi) * \cos(\gamma)$
$\dot{Y} = V * \sin(\chi) * \cos(\gamma)$
$\dot{Z} = V * \sin(\gamma)$
$\dot{X}, \dot{Y}, \dot{Z}$
$X, Y, Z$

**Figure 3.5 Full Closed-Loop Model with CDR Implemented**

## 3.4 Newton-Raphson Solving Method

When the conflict resolution component of the CDR algorithm is active, three nonlinear simultaneous equations in three unknowns must be solved at every compute cycle. To solve these equations, a Newton-Raphson method is utilized. The Newton-Raphson algorithm requires any functions being solved to be equal to zero. Equations (3.11)-(3.12) are rearranged so that this structure is present, and then functions $f_1, f_1, f_3$ are defined as

$$f_1 = \left(\vec{r}_r \times \hat{e}_{V_r}\right) \cdot \hat{e}_{V_r'}$$

$$f_2 = \left\|\hat{e}_{V_r'}\right\|^2 - 1 \tag{3.14}$$

$$f_3 = \left\|\hat{e}_{V_r'} \times \left(\vec{r}_r \times \hat{e}_{V_r'}\right)\right\|^2 - r_s^2$$

Note Equations (3.10) and (3.11) are first squared to avoid square root operations in the numerical calculations. Newton-Raphson works by first creating a “guess” as to what the correct value will be. This guess will be iterated until it meets certain criteria to be accepted as the final, solved value.

It is often the case that a good guess will be the current value of $\hat{\epsilon}_{V_r}$. This type of initialization was implemented in all thesis research computations.

The Newton-Raphson algorithm equation is

$$x_{i+1} = x_i - [f'(x_i)]^{-1} f(x_i) \tag{3.15}$$

where $x_{i+1}$ is the next iteration value and $x_i$ is the past iteration value. Equation (3.15) is an algebraic expression with all terms defined below.

$$\begin{aligned} \hat{\epsilon}_{V_r'} &= \epsilon_{V_{r_X}'} \hat{I} + \epsilon_{V_{r_Y}'} \hat{J} + \epsilon_{V_{r_Z}'} \hat{K} \\ x &= \left[\epsilon_{V_{r_X}'}\ \epsilon_{V_{r_Y}'}\ \epsilon_{V_{r_Z}'}\right]^T \\ &= [x_1\ x_2\ x_3]^T \\ f(x_i) &= [f_1\ f_2\ f_3]^T \\ f'(x_i) &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix} \end{aligned} \tag{3.16}$$

The equations for vector $f(x_i)$ and matrix $f'(x_i)$ are given in Appendix A. The criteria for exiting the iteration is when the absolute value of all three components of the difference between the next iteration value and the current iteration value have met a certain tolerance, or $|x_{1_{i+1}} - x_{1_i}| < \delta$, $|x_{2_{i+1}} - x_{2_i}| < \delta$, $|x_{3_{i+1}} - x_{3_i}| < \delta$ where $\delta$ is the tolerance provided by the user. A smaller tolerance will result in a more accurate answer while also possibly increasing the run time and decreasing the likelihood that the solver will converge to a solution. A value of $\delta = 1 \times 10^{-5}$ was used in all thesis research computations.

# CHAPTER 4

# OVERVIEW AND EVALUTATION OF SIMULATOR

## 4.1 Open-Loop Simulator

The software used to simulate the target vehicle's trajectory will be MathWorks MATLAB and Simulink.[21] These commercial software products offer a variety of useful and powerful tools for engineering computing and simulation. MATLAB will be used to generate reference trajectories and initial conditions as well as plot results while Simulink will handle the simulation and time propagation of the model. The MATLAB *.m* file used for the open-loop model is located in Appendix B. Simulink has a wide variety of settings used to solve a set of ODEs. The solver used in this research is MATLAB's ode4 solver, implementing a Runge-Kutta 4th Order solver.

Values for positions $X, Y, Z$ are in units of distance, $d$, and the user can set the simulator to scale to a certain desired length. Time variable $t$ is similarly defined using a unit of time, $s$, defined by the user. Velocity is then in units of length divided by units of time, $\frac{d}{s}$. Heading and climb angles are measured in radians. All simulation runs in this thesis are computed with no specific units indicated as the simulations typically represent sub-scale cases consistent with research conducted in Reference 18. Parameters $\tau_V, \tau_\chi, \tau_\gamma$ are kept constant at 1 for all simulations performed in this thesis. The final simulation time is also kept constant for all runs at a constant of 10 units. The step size of the ode4 solver is kept constant at 0.001 units of time. All simulations will begin with initial positions of the target vehicle at $X = 0, Y = 0, Z = 500$ distance units.

The Simulink open-loop model is shown in Figure 4.1 and corresponds to the engineering block diagram in Figure 2.2.

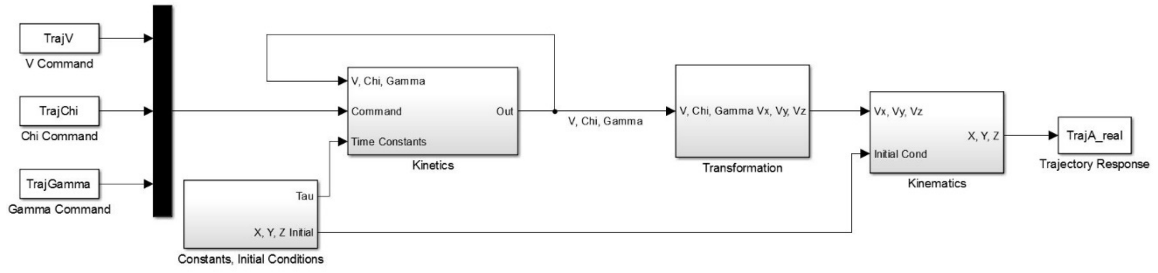


**Figure 4.1 Simulink Open-Loop Model**

The open-loop system accepts $V_c, \chi_c, \gamma_c$ as well as calculated initial values for $V, \chi, \gamma$ and $X, Y, Z$, and $\tau_V, \tau_\chi, \tau_\gamma$ values from the MATLAB workspace and propagates the vehicle kinetics and kinematics in both the inertial and velocity frames. A test case of the open-loop system using straight and level flight is shown below in Figure 4.2. This case has initial velocity of 100 $\frac{d}{s}$, and initial $\chi$ and $\gamma$ values of 0 rad. The vehicle is given a step input of $+10\ \frac{d}{s}$ after $2s$ and continues at this velocity for the duration of the simulation. Figure 4.3 shows the plot of $V$ vs. $t$ for both $V$ and $V_C$. The simulator responds well, albeit slowly, to the command of $+10\ \frac{d}{s}$. The simulated inertial trajectory extends past the final value of $1000\ d$, which is the expected final position of a vehicle traveling at a constant $100\ \frac{d}{s}$ for $10\ s$. The response in Figure 4.3 shows a response that could be quickened or slowed by changing the value of $\tau_V$, which this thesis does not explore.

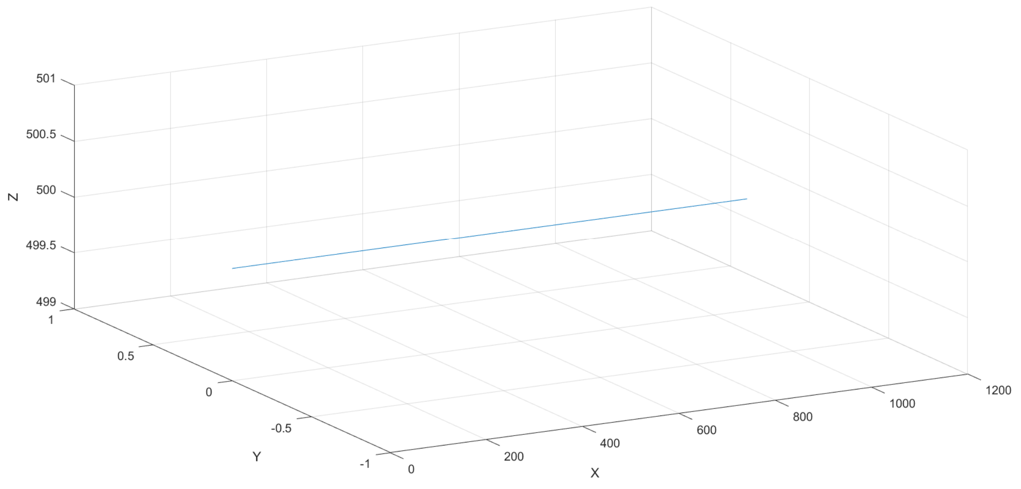


**Figure 4.2 Open-Loop, Straight and Level, $V_C$ of $+10\,\frac{d}{s}$**

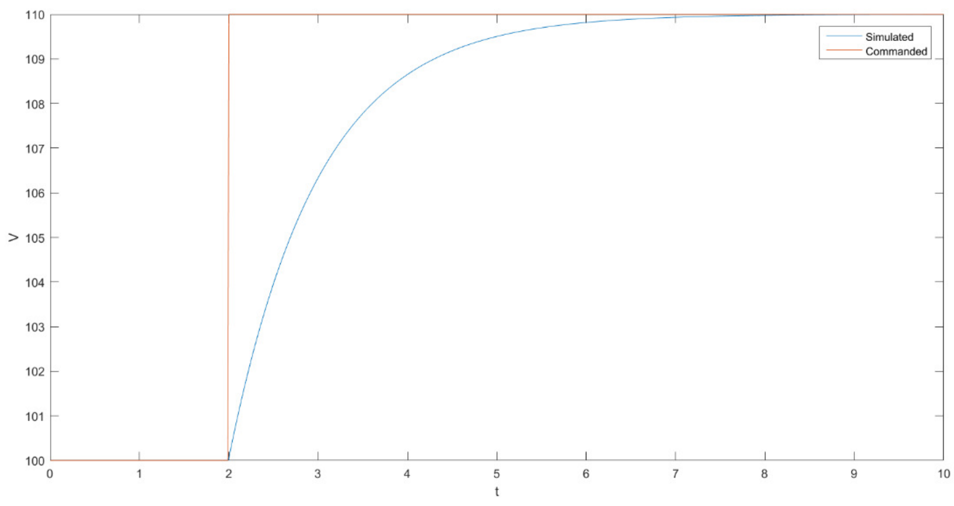


**Figure 4.3 $V\ vs\ t$, $V_C$ of $+10\ \frac{d}{s}$**

Figure 4.4 shows a case where velocity was kept constant at $100\ \frac{d}{s}$, $\gamma$ is kept at a constant $0\ rad$, and a step input of $\frac{\pi}{6}\ rad$ was given at $2\ s$, then a counter step of $-\frac{\pi}{6}\ rad$ was given after

$1\ s$, and after another $1\ s$, a command of $0\ rad$ is given to straighten the vehicle motion out. Figure 4.5 shows a plot of $\chi$ vs. $t$ for both $\chi$ and $\chi_C$.

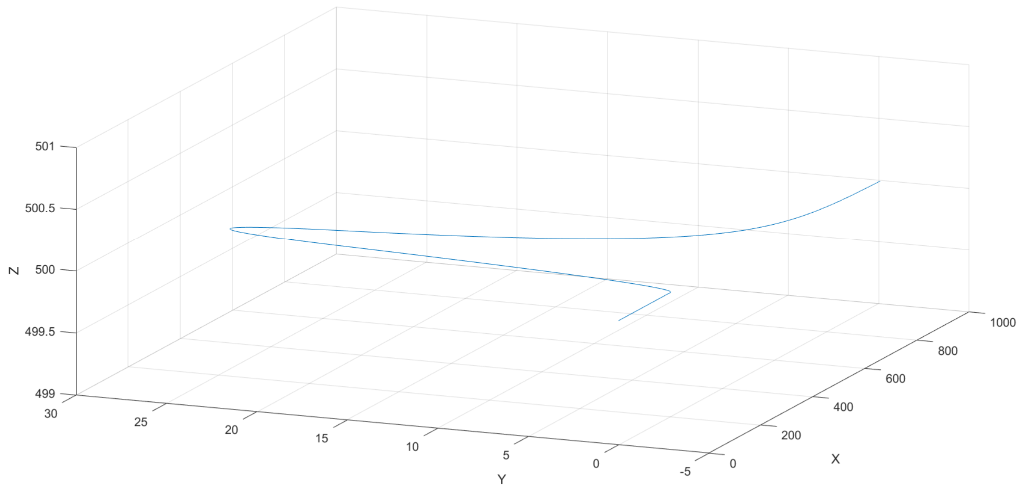


**Figure 4.4 Open-Loop, Level, $\chi_C$ of $\pm\frac{\pi}{6}\ rad$**

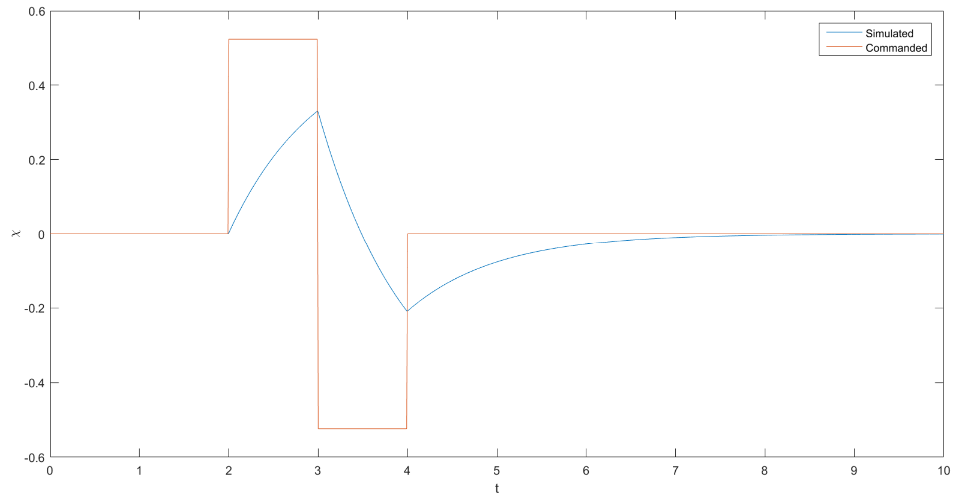


**Figure 4.5 $\chi$ vs. $t$, $\chi_C$ of $\pm\frac{\pi}{6}\ rad$**

A similar response to Figure 4.3 is shown in Figure 4.5. The simulated trajectory is smooth and even, which is a very desirable result. The commanded value of $\chi$ is unable to be met before the next step is given, however this is likely due to the time delay constant. Figure 4.6 shows a case where $V$ and $\chi$ are kept constant and $\gamma$ is given a similar step input to the case above except with a $\pm\frac{\pi}{4}\ rad$ magnitude.

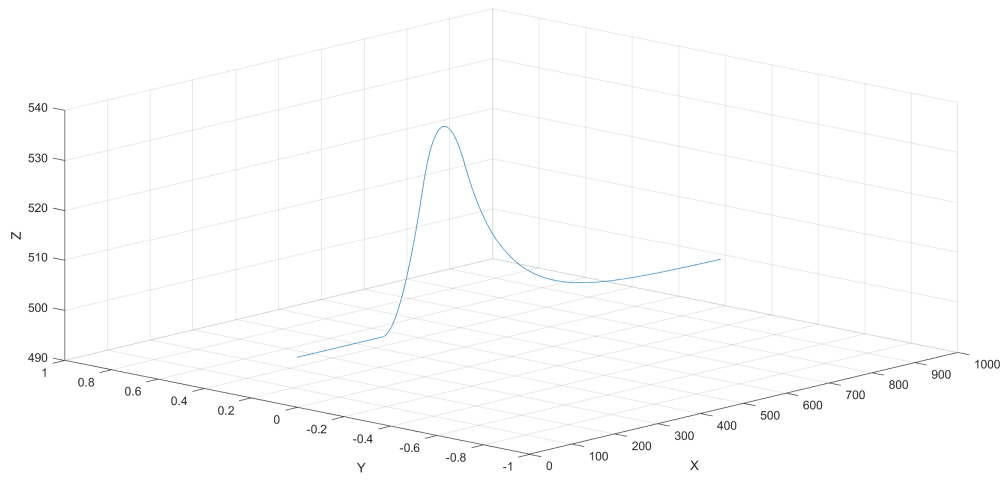


**Figure 4.6 Open-Loop $\gamma_C$ of $\pm\frac{\pi}{4} rad$**

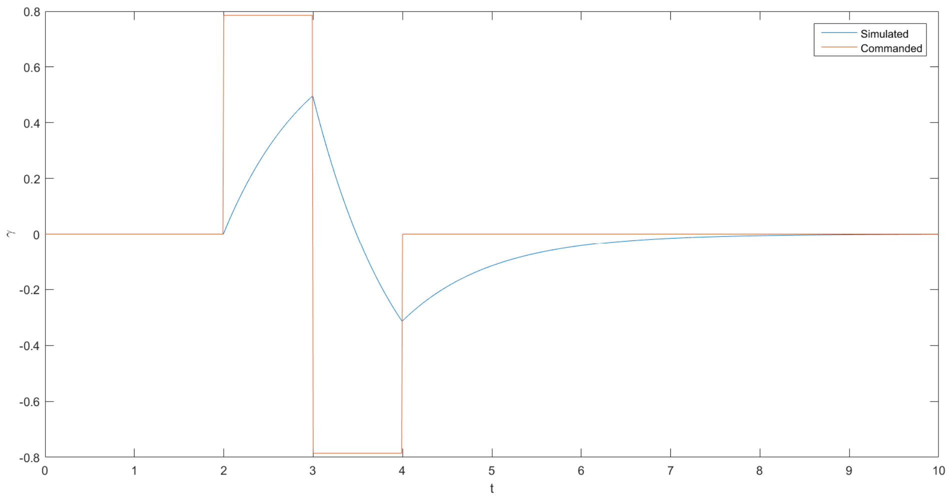


**Figure 4.7 $\gamma$ vs. $t$, $\gamma_C$ of $\pm\frac{\pi}{4}\ rad$**

Nearly identical responses to Figures 4.4 and 4.5 are seen in this simulation. These test cases indicate that the kinetic and kinematic relationships are working as expected.

## 4.2 Closed-Loop Simulator (4PF)

After installing the 4PF control logic from Section 3.1 and closing the loop, the simulator can take inertial $X, Y, Z$ command trajectory coordinates as well as initial conditions and parameter values from the MATLAB workspace and propagate the vehicle in three-dimensional space. Linear trajectories are built using MATLAB's *linspace* command that linearly spaces values between an initial and final value for a specified number of points. Initial trajectory coordinates are kept constant from the open-loop model above and final trajectory coordinates are kept constant for level flight cases at $X = 1000, Y = 500, Z = 500$. To create an instantaneous command jump, a value that can be specified by the engineer was added a certain period of time for the duration of the run. To create a curved trajectory, a *linspace* command from 0 to $\pi$ was placed inside of a sin function and added to the previous linear trajectory created. Appendix C shows details of the PID control implementation for the "4PF Control Logic" block, and Figure 4.8 shows the Simulink closed-loop model with 4PF capability. This computer implementation corresponds to the engineering block diagram presented in Figure 3.2. The MATLAB *.m* file for the closed-loop model is located in Appendix B.

The 4PF control block accepts $X_C$, $Y_C$, and $Z_C$ as well as feedback signals $X, Y, Z$ and $V, \chi, \gamma$ as inputs, and outputs velocity commands $V_C, \chi_C, \gamma_C$ which drives the aircraft dynamics. Table 4.1 below presents the values for the controller gains while Table 4.2 presents the filter coefficients for the PID controller. These values are kept constant for all simulations. The controller gains and filter coefficients were chosen through trial and error to produce a stable response. A series of tests showing the simulator's ability to follow various trajectories are presented next.

**Table 4.1 PID Controller Gains**

| | $V$ | $\chi$ | $\gamma$ |
|---|---|---|---|
| $K_P$ | 1 | 0.01 | 0.01 |
| $K_I$ | 1 | 0.01 | 0.01 |
| $K_D$ | 1 | 0.005 | 0.005 |

**Table 4.2 Filter Coefficient Values**

|   | $V$ | $\chi$ | $\gamma$ |
|---|---|---|---|
| N | 100 | 10 | 10 |

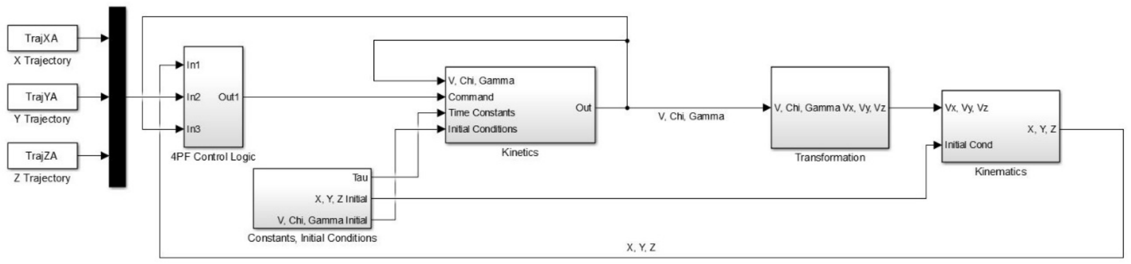


**Figure 4.8 Simulink Closed-Loop Model**

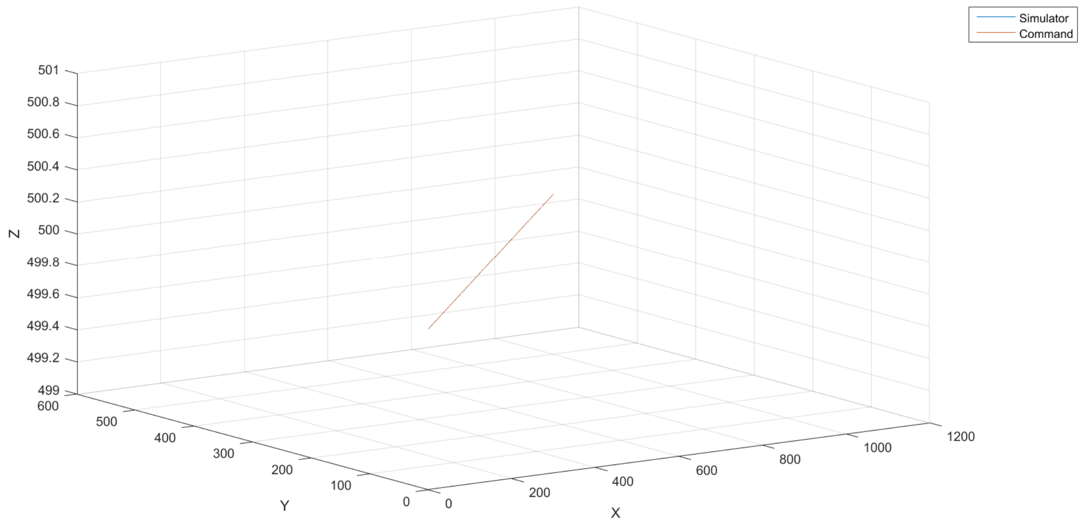


**Figure 4.9 Closed-Loop Test Case, Straight and Level Flight**

Figure 4.9 shows the results of a straight and level, equilibrium test case. This test case is quite unremarkable as the simulated and command response lines lie on top of each other, demonstrating the closed-loop system maintains the flight condition equilibrium. A simple change in the commanded final $Z$ coordinate of $+100\ d$ (referred to in this thesis as a 3D trajectory) along with a new initial condition for $\gamma$ results in a straight, climbing equilibrium test case. This case, shown in Figure 4.10, shows a slight deviation from the commanded trajectory. The presence of the blue simulator line on top of the red commanded line indicates that there is a discrepancy between the simulated and commanded trajectories. The difference is not large enough to cause any alarm about the effectiveness of the simulator. In fact, it shows that even if there is a discrepancy then the controller is working properly to get the vehicle back on track, as evidenced by the simulator line disappearing back behind the command line.

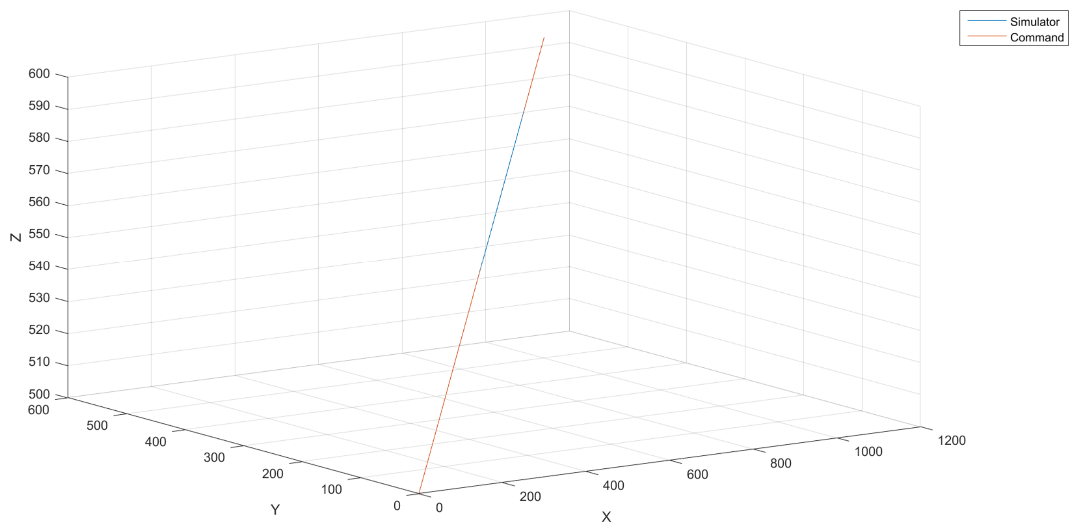


**Figure 4.10 Closed-Loop Test Case, 3D Trajectory**

A case where an instantaneous jump in the $X$ direction is applied is shown in Figures 4.11 and 4.12. A jump of $+100\ d$ was given at $2\ s$. The vehicle has some initial overshoot but returns to the commanded value after a relatively short time. A 2D plot of just the $Y$ vs. $X$ data shows that the vehicle is approaching back to the command value.

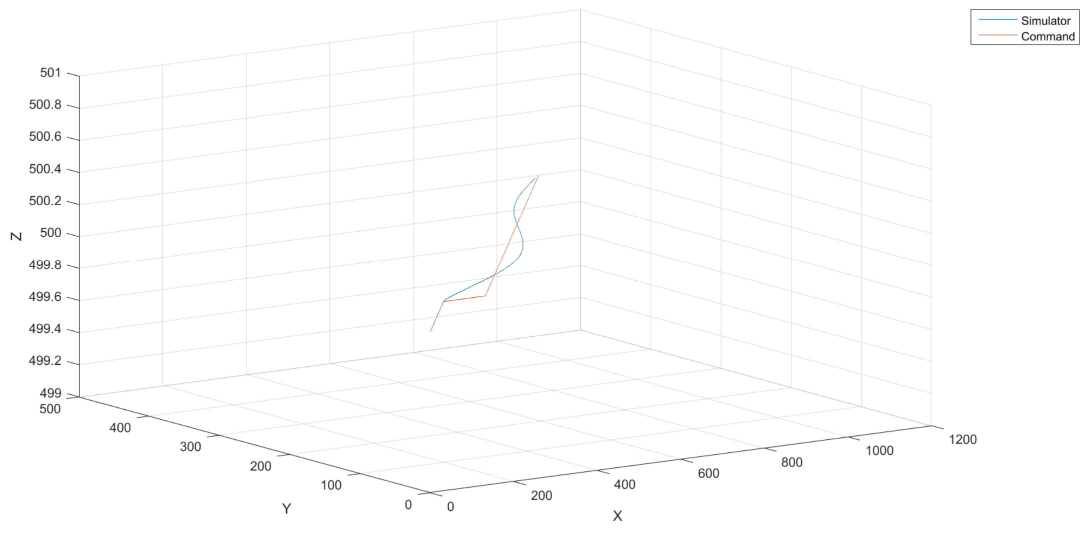


**Figure 4.11 3D View of Trajectory with Instantaneous Jump in *X***

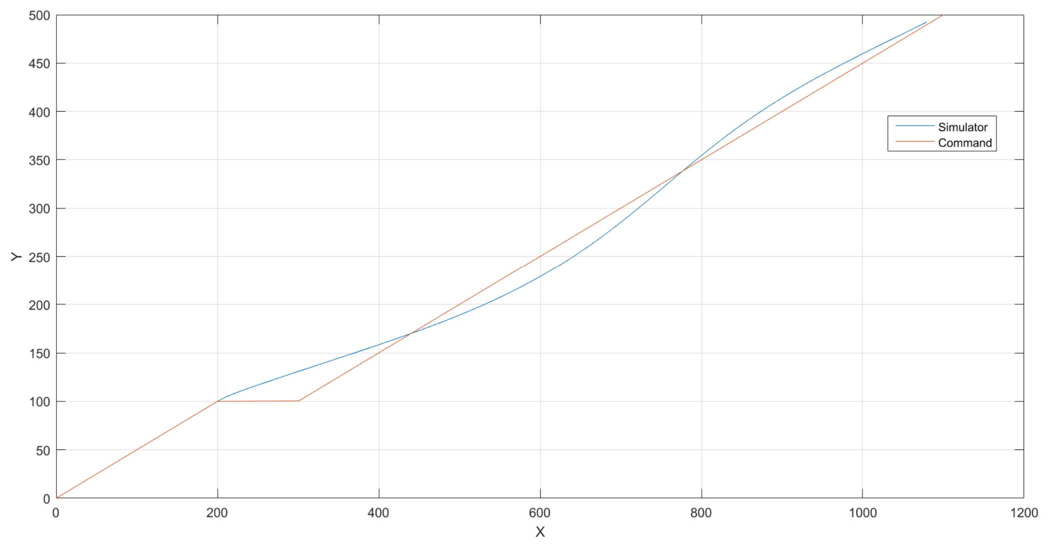


**Figure 4.12 2D View of Trajectory with Instantaneous Jump in *X***

Figures 4.13, 4.14, and 4.15 below show the ability of the simulator to respond to a jump in all $X, Y, Z$ directions. The scale of the plot creates the illusion that the simulator is unable to respond as quickly to a jump in the $Z$ direction, however an overshoot of only 4 $d$ may or may not be concerning to the researcher. A vertical overshoot of 4 miles could be significantly more concerning than one of 4 feet. If the overshoot proves to be an issue, adjusting the $\kappa_{P_\gamma}, \kappa_{I_\gamma}, \kappa_{D_\gamma}$ gains or even a change in the $\tau_\gamma$ value could produce more desirable results.

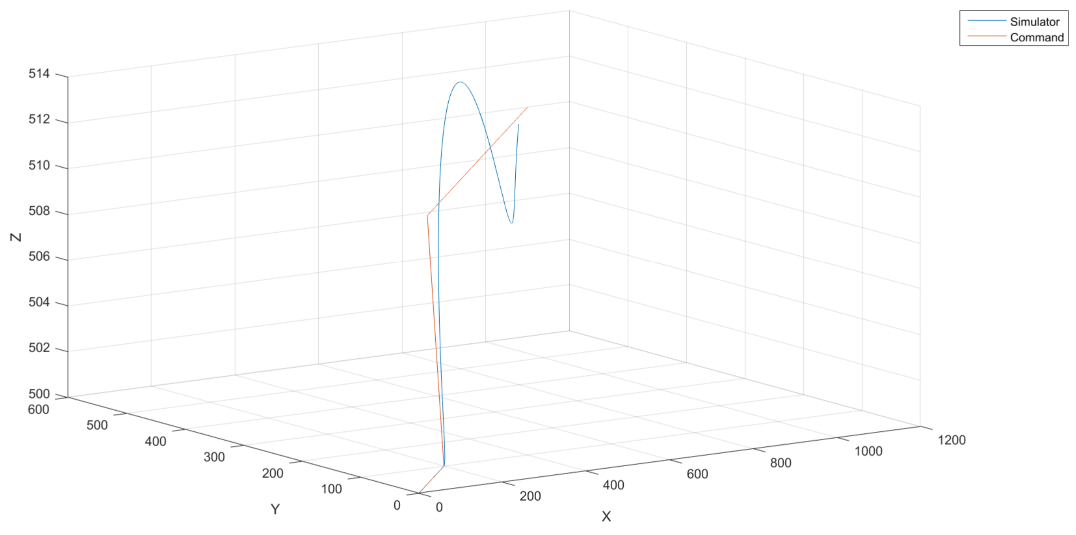


**Figure 4.13 3D View of Trajectory with Instantaneous Jump in $X, Y, Z$**

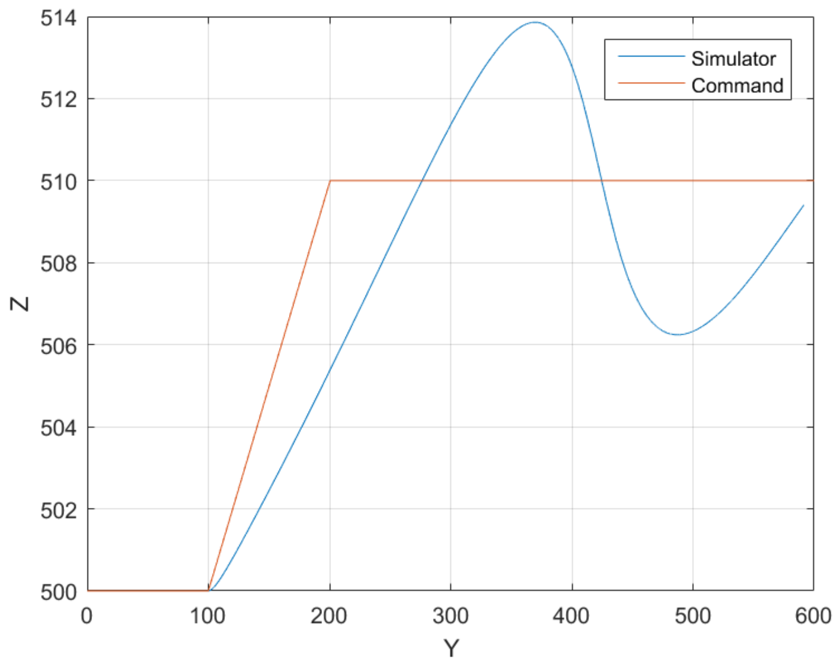


**Figure 4.14 2D View of Trajectory with Instantaneous Jump in *Y*, *Z***

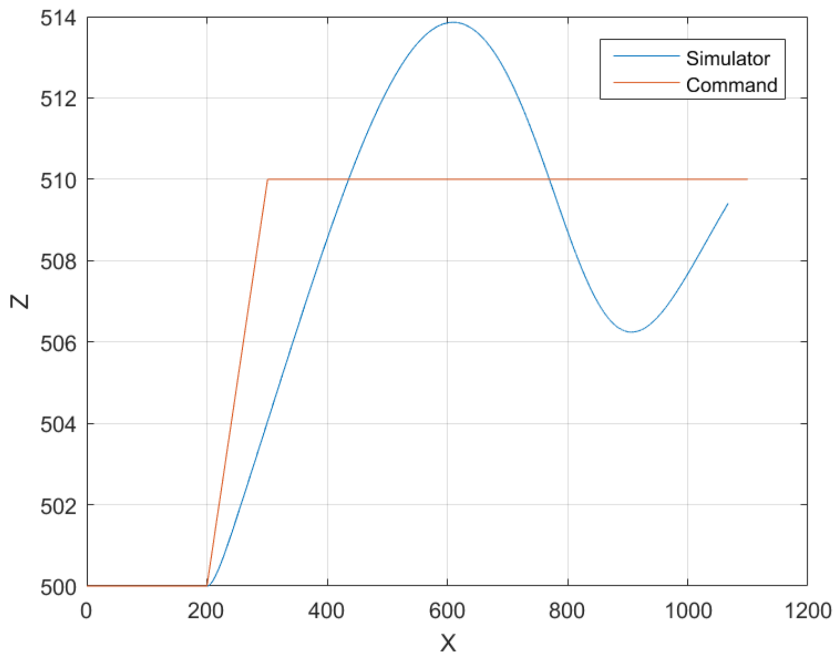


**Figure 4.15 2D View of Trajectory with Instantaneous Jump in *X*, *Z***

A curved trajectory was also used to test the simulator. A 3D trajectory implementing a curve in the $X$ direction is shown in Figure 4.16. The simulator is able to respond very well and quickly to the commanded trajectory with very acceptable results. A more rigorous test of the simulator is for it to implement a 3D trajectory with a curve in both $X$ and $Y$ directions. Figure 4.17 demonstrates the ability of the simulator to accurately follow this trajectory. Figure 4.18 shows the 2D plot of $Y$ vs. $X$ that shows the effectiveness in the $X$ and $Y$ directions.

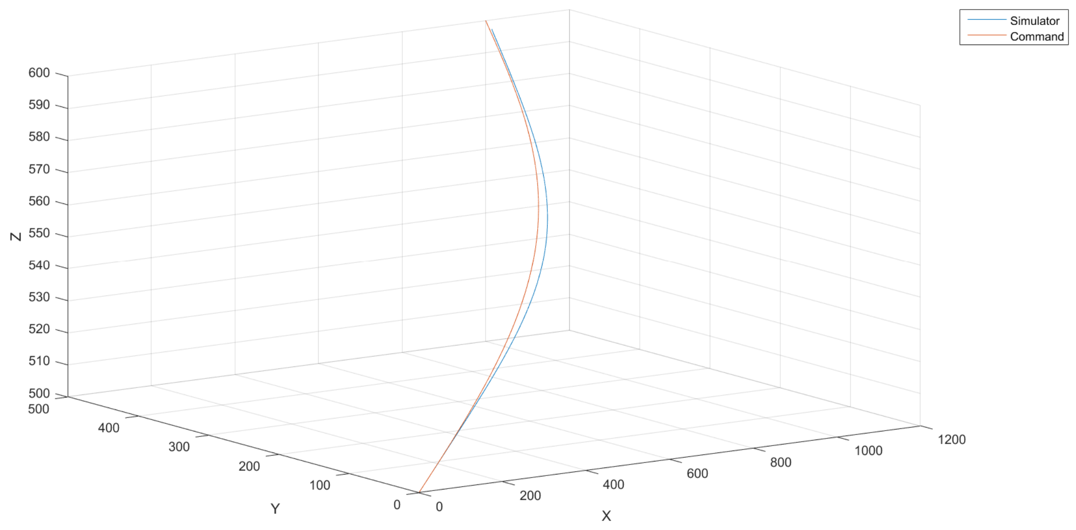


**Figure 4.16 3D View of 3D Trajectory; Curve in *X***

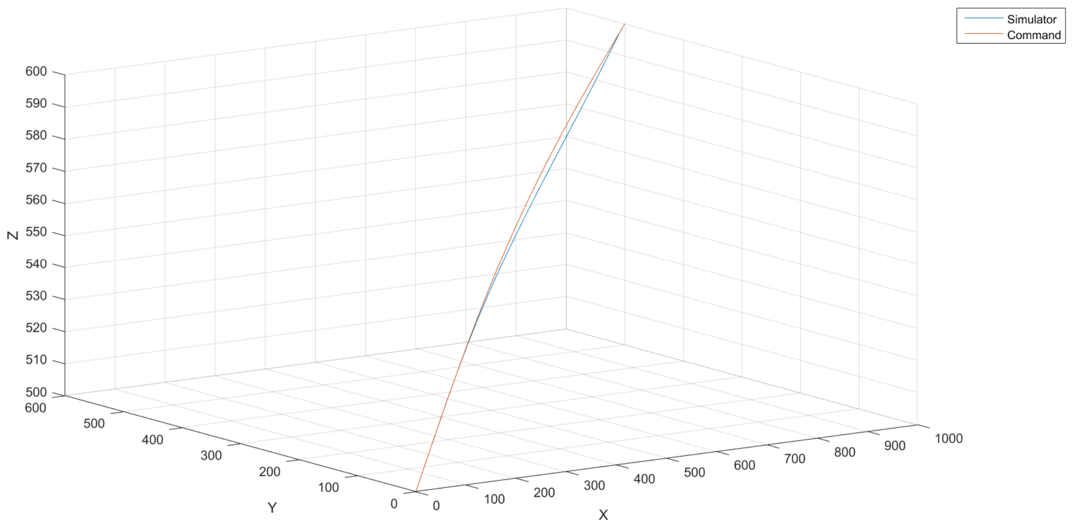


**Figure 4.17  3D View of 3D Trajectory; Curve in *X* and *Y***

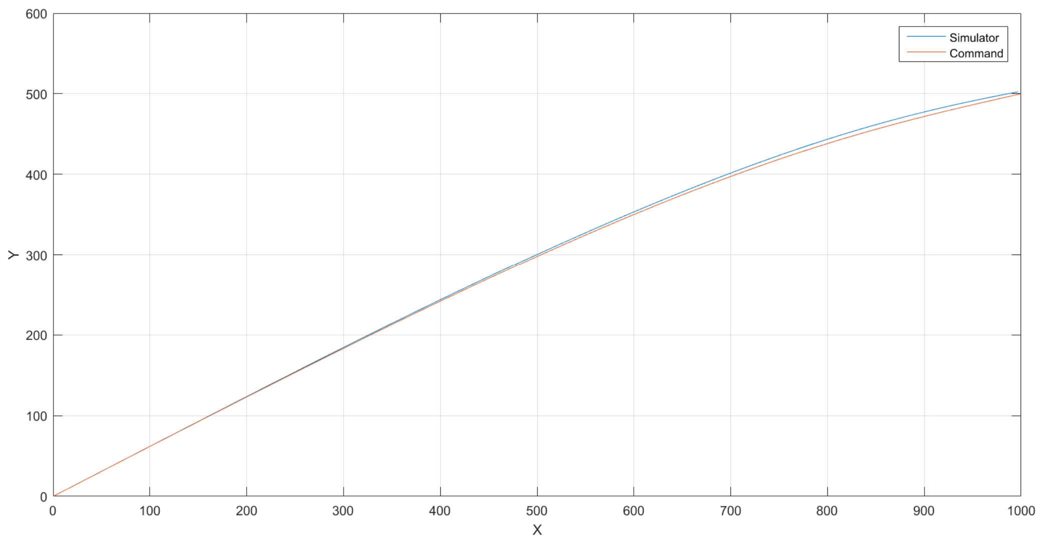


**Figure 4.18 2D View of 3D Trajectory; Curve in X and Y**

It can be seen that the simulator utilizing the 4PF controller is quite capable of tracking a wide variety of command trajectory types.

## 4.3 Closed-Loop Simulator (4PF + CDR)

The full Simulink closed-loop model used to simulate a CDR scenario is shown in Figure 4.19 which corresponds to the engineering block diagram shown in Figure 3.5. Internals of the subsystems that don't pertain to the CDR algorithm can be seen in Appendix C. Calculation of the miss vector and miss time is performed in the separate block labeled "PCA Logic." This block inputs all position and velocity states of both aircraft A and aircraft B and outputs $r_m$ and $\tau_m$. Details of this calculation are shown in Figure 4.20. Conflict resolution and conflict detection logic is then processed in the separate block labeled "CDR Logic." This block inputs $r_m, \tau_m, \vec{R}_A, \vec{V}_A, \vec{R}_B, \vec{V}_B$ and outputs the CDR activation switch signal and the modified trajectory command.

Simulink has no readily available template block that solves simultaneous equations, so code has to be written in a function block to implement the Newton-Raphson method previously discussed. The internal processing of the CDR Logic subsystem is the conflict resolution algorithm function block and its inputs and outputs are shown in Figure 4.21. The MATLAB function used to compute the $\hat{e}_{V_r'}$ vector and decision making on whether a conflict exists or not, and construction of the new commanded trajectory, is located in Appendix B. This assembled simulator will be used to analyze several CDR scenarios and test the effectiveness of the CDR algorithm in solving these problems.

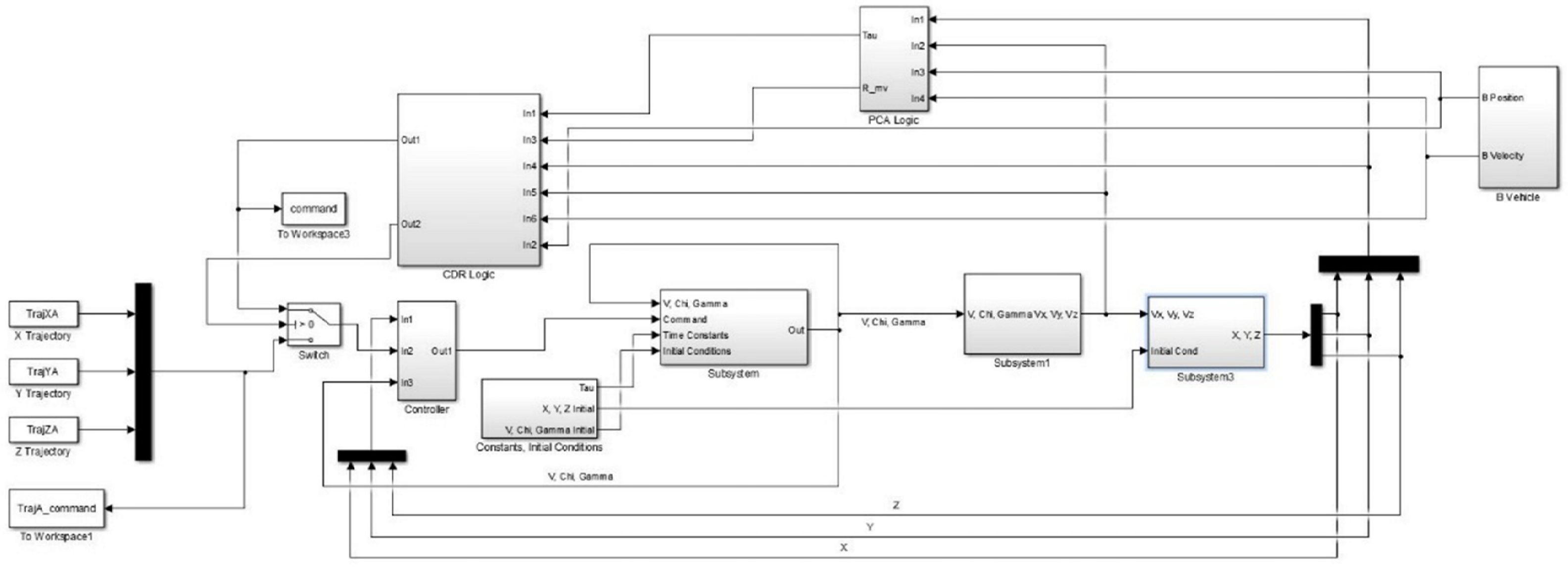


Figure 4.19 Simulink Closed-Loop Model with CDR Algorithm

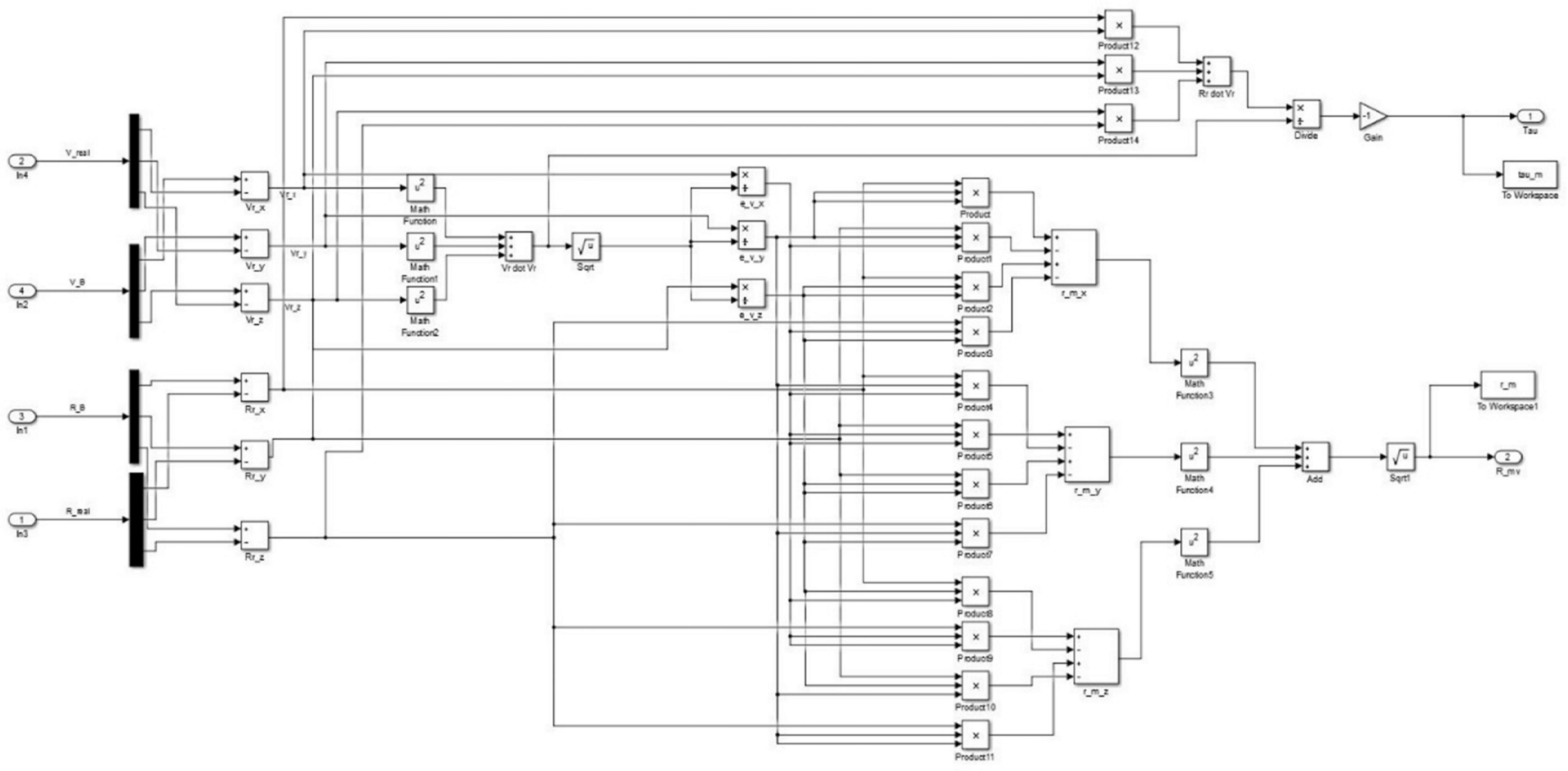


**Figure 4.20 Simulink Wiring for Conflict Detection Portion of CDR Algorithm**

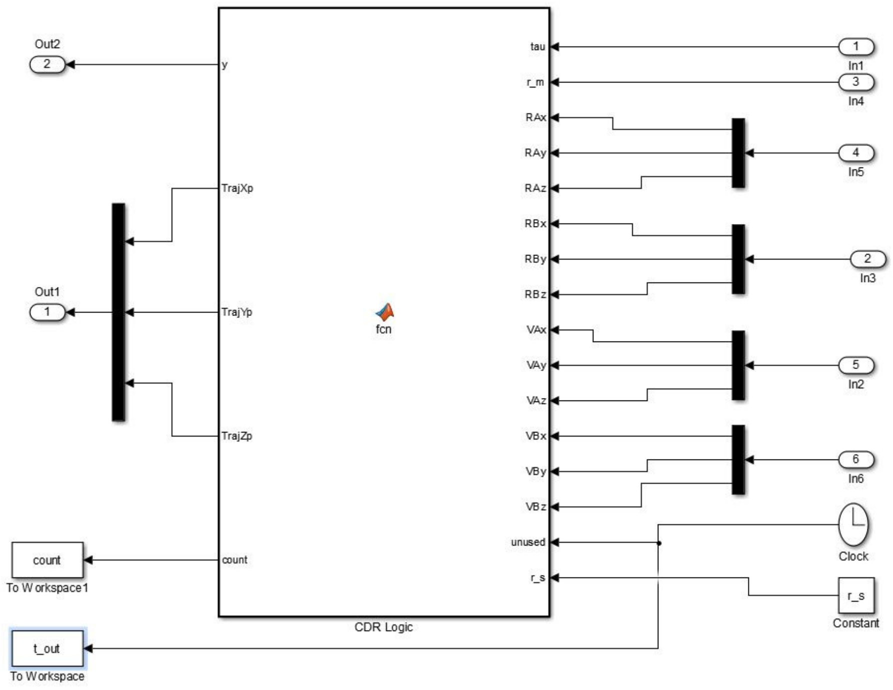


**Figure 4.21 Simulink Wiring for Conflict Resolution and Trajectory Synthesis**

# CHAPTER 5

# RESULTS

## 5.1 Nominal Trajectory

This chapter covers the results obtained by using the CDR algorithm and MathWorks MATLAB and Simulink computer implementation described in the previous chapters to analyze the effectiveness of the logic in avoiding air traffic conflicts. In this section, a nominal, 2D test case will be presented that further simulation runs will be compared against. Each simulation run presented will have plots of the trajectory in 3D space, the trajectory projection in the $X$-$Y$ plane, $r_m$ vs. $t$, and $\tau_m$ vs. $t$. The reason for showing both the 3D view and the 2D $X$-$Y$ view is that the majority of the movement ends up being in the $X$-$Y$ plane so a top-down view of the run is often more effective in visualizing the geometry.

Velocities of the conflict vehicle are calculated using a finite difference method described by the equation

$$\dot{x}_i = \frac{x_{i+1} - x_i}{h} \tag{5.1}$$

where $\dot{x}_i$ is the approximated time derivative, $x_{i+1}$ is the value one time-step in the future, $x_i$ is the value at the current time, and $h$ is the value of the time-step. This method is valid when $x_{i+1}$ is known. At the final time, when $i = n$, the value of $x_{n+1}$ is undefined so the velocity $\dot{x}_n$ is approximated to be the same as the value for $\dot{x}_{n-1}$. The value of $h$ used is 0.01.

All values from the previous chapter remain constant and the safe sphere radius $r_s$ is at a constant 275. Target vehicle initial coordinates are kept constant at $X = 0, Y = 0, Z = 500$ and final $X$ and $Y$ coordinates are constant at $X = 1000, Y = 500$. Final target vehicle altitude coordinates are determined by whether or not the case being run is a climbing 3D case. Conflict vehicle initial and final coordinates are varied in different cases; however, the altitude is kept constant at $Z = 550$. The direction of movement along a trajectory is denoted by a black circle for the starting point, and a green cross for the ending point. The PCA is denoted on the target and conflict vehicle trajectories by a red X. In these simulation runs, the safe sphere is placed around the conflict vehicle at the time of the PCA. Throughout this thesis, the safe sphere has been presented as being drawn over the target vehicle. It is easier to visualize the performance of the CDR logic in these simulations if it is shown to be maneuvering around a sphere centered on the conflict vehicle. Regardless of where the sphere is centered, the radius between the vehicles remains the same.

Figure 5.1 demonstrates a $\tau_m$ plot with no CDR active. The plot is linear with a constant decreasing slope and is consistent with what should be expected. The $\tau_m$ vs. $t$ plots of later simulation runs will be compared against this nominal case to assess the effectiveness of the CDR logic to avoid the presented conflict.

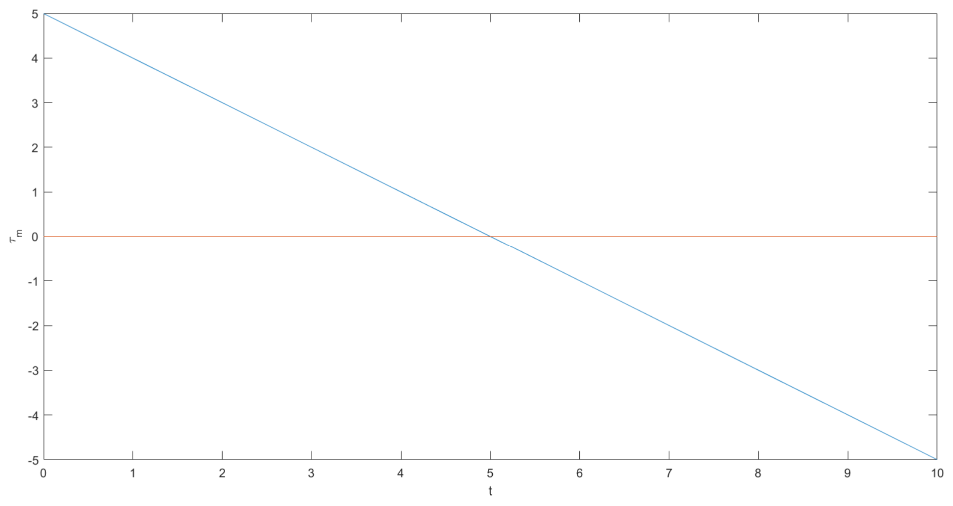


**Figure 5.1 Plot of $\tau_m$ vs. *t* with No Conflict**

Figures 5.2 and 5.3 below show the nominal, 2D case. The commanded target vehicle's trajectory is straight and level and the initial and final conflict vehicle planar coordinates are $X = 500, Y = 500, Z = 550$ and $X = 1050, Y = 0, Z = 550$. In this nominal run, it is shown that the CDR algorithm is ineffective at properly reaching the desired $r_s$. However, it was effective at recognizing that a conflict existed and attempting to steer the vehicle away from the conflict. This note is important because this nominal case presents the closest that the algorithm can come to properly meeting this requirement. The following runs will be used to analyze how the simulator and CDR algorithm that was developed responds to various trajectories.

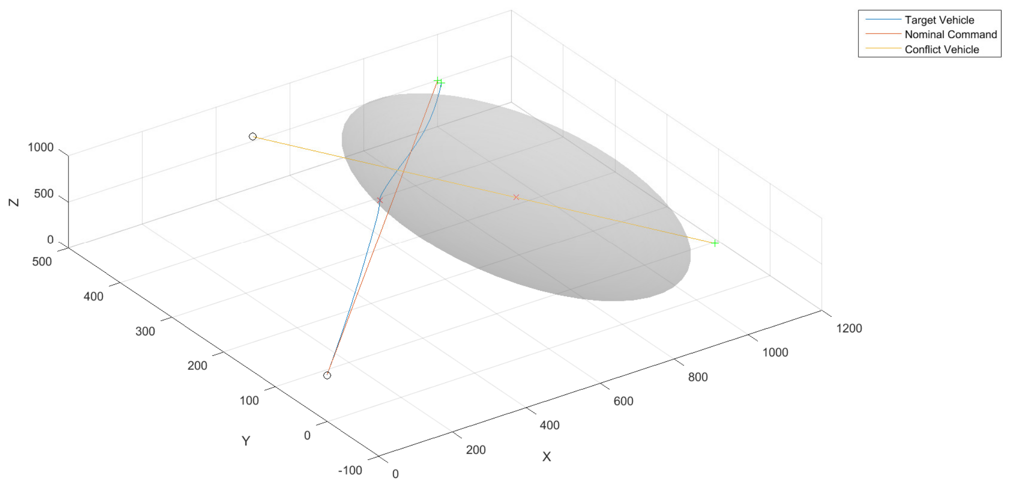


**Figure 5.2 3D View of Nominal, 2D Trajectory**

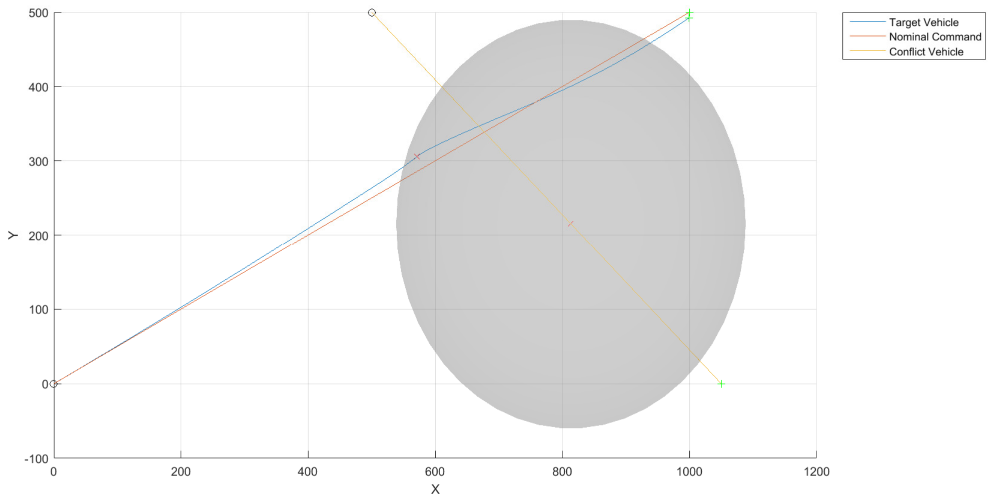


**Figure 5.3 2D View of Nominal, 2D Trajectory**

Figures 5.4 and 5.5 show the nominal $\tau_m$ and $r_m$. When compared to the plot of the $\tau_m$ in Figure 5.1, without any CDR active, there is a discrepancy around the time where $\tau_m = 0$. This difference is because at this point, the simulated vehicle is getting very close to the PCA. Since $\tau_m$ is positive before the PCA and negative after the PCA, any time where the simulator and target vehicle are operating around the PCA, $\tau_m$ will alternate between positive and negative because of how the CDR logic is built. The CDR logic is designed so that at a $\tau_m < 0$, the controller is commanded to follow the original trajectory dictated by the user. This commanded trajectory is already located inside of the safe sphere so on the next simulation run, $\tau_m$ is no longer negative and the CDR activates again to push the vehicle outside of the safe sphere. This process repeats until the vehicle is sufficiently clear of the PCA where the user-commanded trajectory is again used to steer the vehicle. This phenomenon, known in this thesis as chattering, will be seen in later $\tau_m$ vs. $t$ plots.

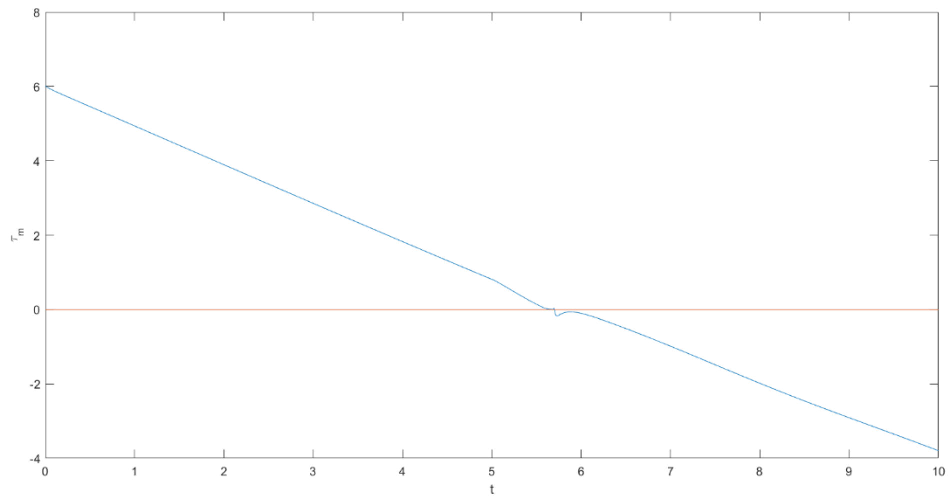


**Figure 5.4 Plot of $\tau_m$ vs. $t$ for Nominal, 2D Trajectory**

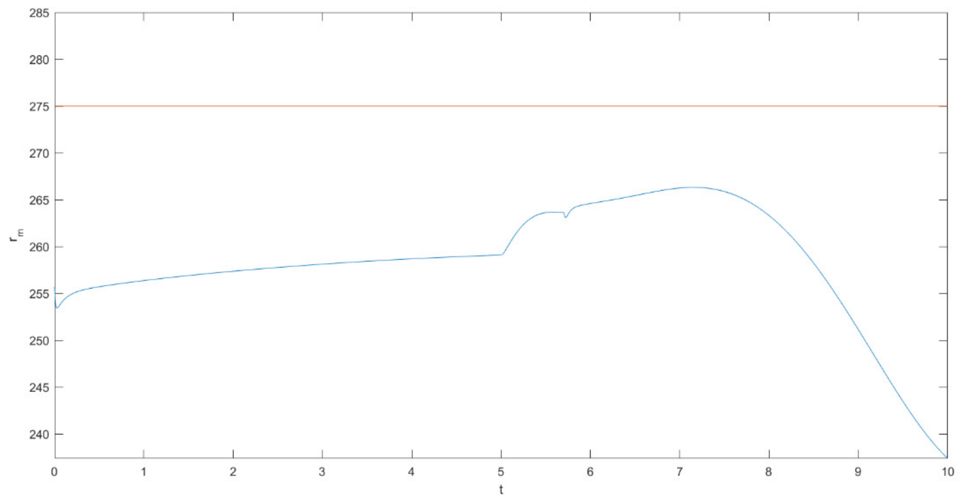


**Figure 5.5 Plot of $r_m$ vs. $t$ for Nominal, 2D Trajectory**

In Figure 5.5, observe that the aircraft are in conflict at the initial condition with $r_m = 255 < 275 = r_s$. The CDR control system immediately maneuvers the vehicle and $r_m$ begins to increase. The plot of $r_m$ vs. $t$ in Figure 5.5 shows that the $r_m$ never reaches the required $r_s$. The $r_m$ at the PCA is 263.63 which results in a 4.13% error from the required $r_m$ of 275. The shape of the $r_m$ curve is also highly irregular. When the CDR algorithm is working properly, the PCA should be pushed almost immediately to the required value. This, along with the $r_m$ value at the PCA, is evidence that the developed CDR algorithm falls short of properly meeting the $r_s$ requirement for this case. The slight increase in $r_m$ in Figure 5.5 between 0 and 7 $s$ does indicate that the logic was attempting to steer the vehicle to avoid the conflict.

Figures 5.6 and 5.7 show a 3D case using the nominal 2D trajectory but with a final target altitude of $Z = 600$. Once again, the simulator is unable to push the PCA to the required point on the edge of the safe sphere. This case does highlight the ability of the controller to return to the user-commanded trajectory after the PCA has been cleared.

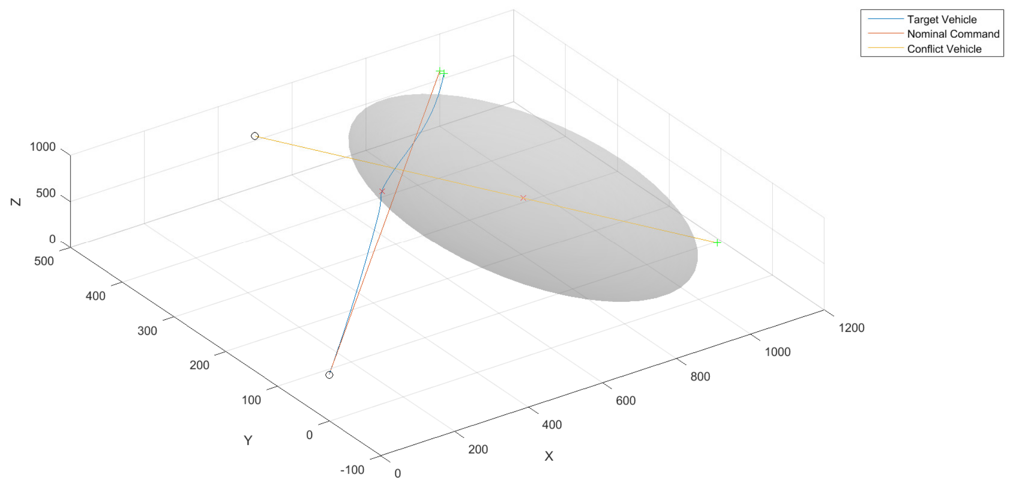


**Figure 5.6 3D View of Linear, 3D Trajectory**

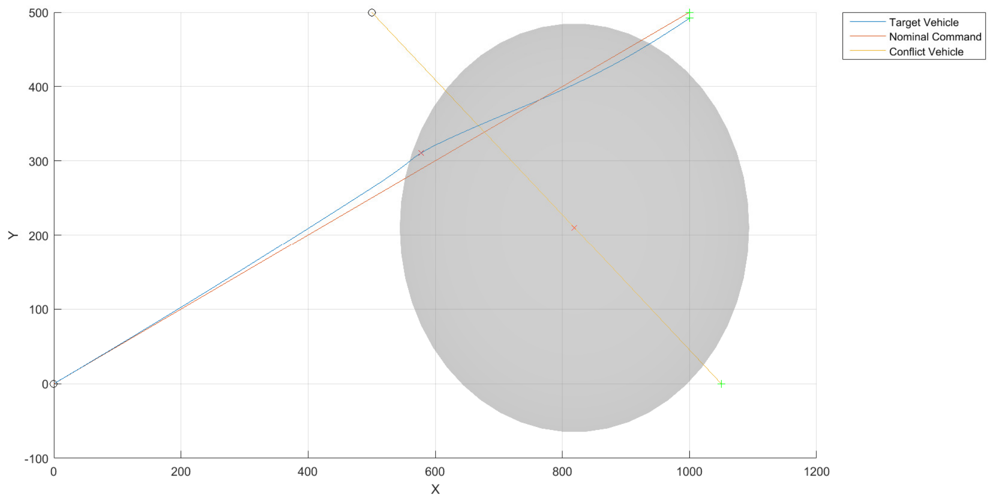


**Figure 5.7 2D View of Linear, 3D Trajectory**

Figures 5.8 and 5.9 show the corresponding result for the $\tau_m$ and $r_m$ responses as seen in the nominal case. The $\tau_m$ vs. $t$ plot in Figure 5.8 does see the first indication of $\tau_m$ being affected by the chattering phenomenon discussed above. The $r_m$ value at the PCA for this case is 261.53 which results in a 4.90% error from the required value.

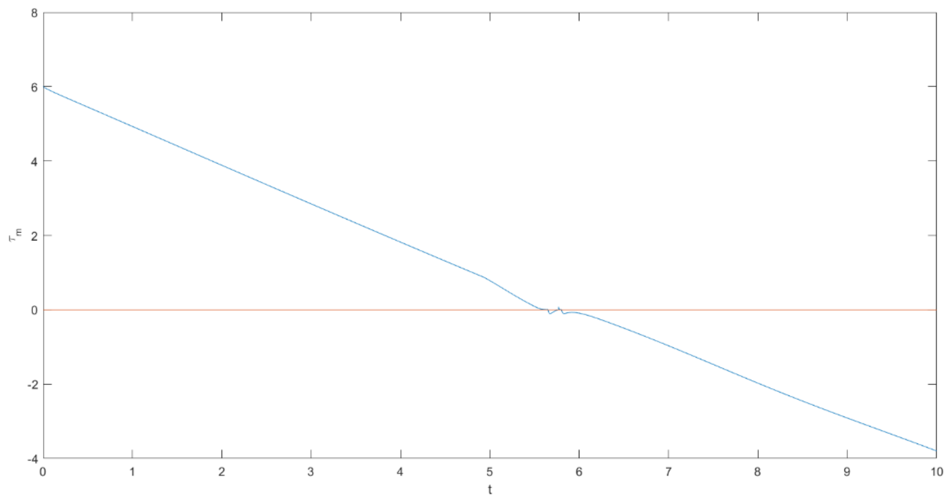


**Figure 5.8 Plot of $\tau_m$ vs. *t* for Linear, 3D Trajectory**

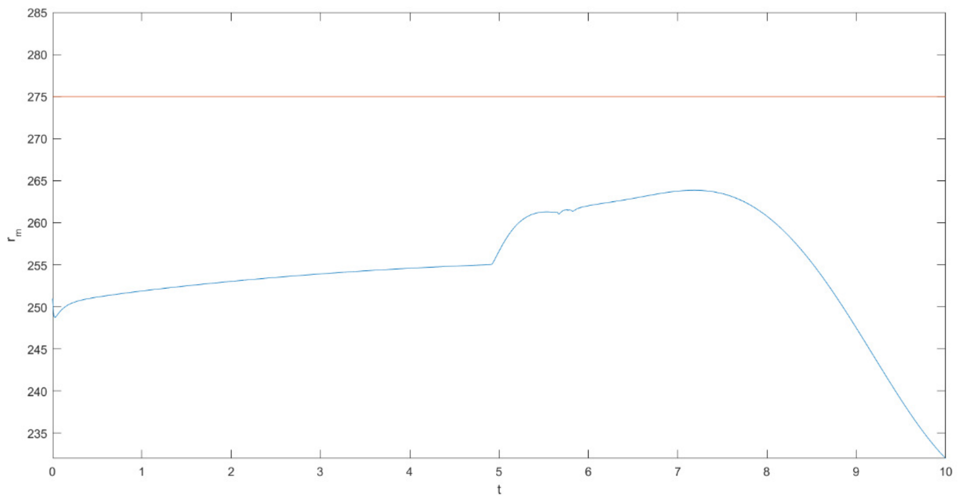


**Figure 5.9 Plot of $r_m$ vs. *t* for Linear, 3D Trajectory**

## 5.2 Altered Conflict Vehicle Trajectory

The next case studied will change the trajectory of the conflict vehicle. The conflict vehicle's initial positions were changed to be $X = 1050, Y = 500$ and the final positions are $X = 550, Y = 0$. This change places the conflict vehicle on more of a head-on trajectory relative to the target vehicle that is slightly skewed away from the target vehicle. These values for the conflict vehicle's trajectory are kept constant for the rest of the simulation runs. This scenario is the first case where an interesting phenomenon appears where the simulated vehicle seems to be forced to adjust rather quickly to try to meet the PCA requirement. Figures 5.10 and 5.11 show the 3D and 2D views of this case. Not only does the vehicle's normal trajectory clearly put it on a path that does not meet the PCA requirement, but it also seems that the CDR algorithm suddenly aggressively engages in an effort to meet the requirement, only to partially succeed and return back to the user-commanded trajectory.

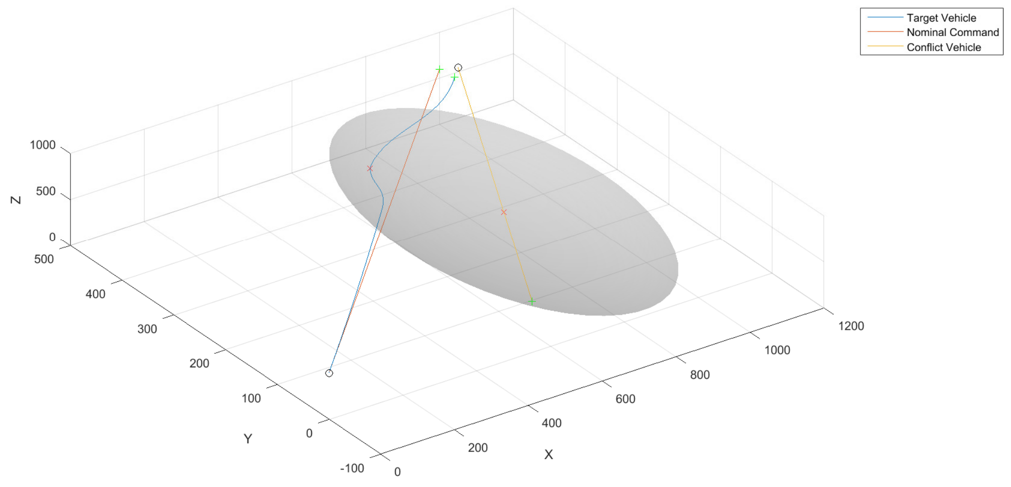


**Figure 5.10 3D View of Linear, 3D Trajectory with Alternate Conflict Vehicle Trajectory**

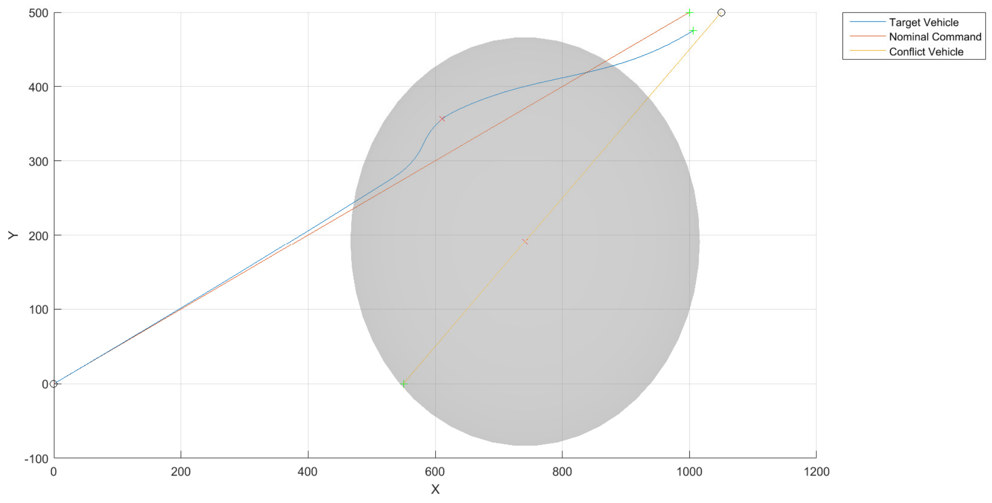


**Figure 5.11 2D View of Linear, 3D Trajectory with Alternate Conflict Vehicle Trajectory**

The $\tau_m$ and $r_m$ vs. $t$ plots for this case shown in Figures 5.12 and 5.13 can help to explain a bit of what is happening in this case. The $\tau_m$ vs. $t$ graph shows more extreme chattering than seen previously. The $r_m$ vs. $t$ graph shows the same shape as the previous graphs and is more evidence that the CDR algorithm is not properly pushing the PCA point to the required value. The value of $r_m$ at the PCA is 211.34 which results in an error of 23.15% from the required value.

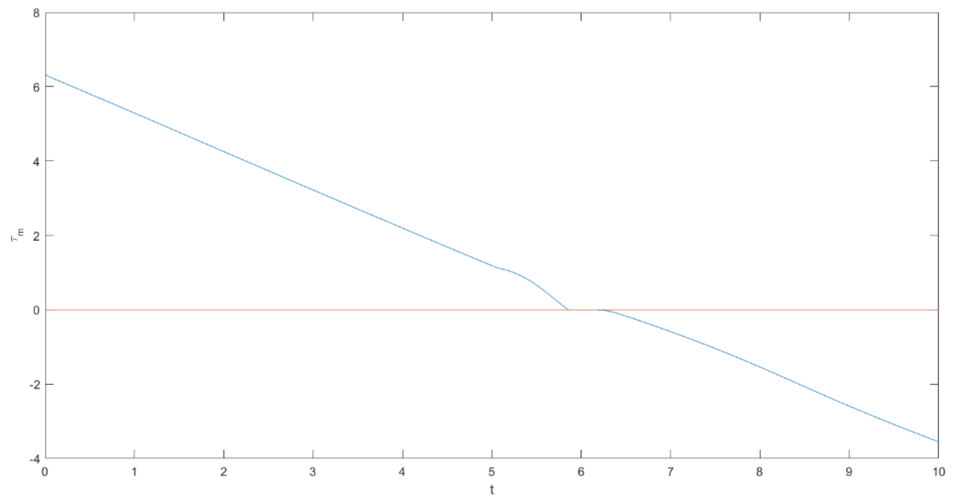


**Figure 5.12 Plot of $\tau_m$ vs. *t*; Altered Conflict Vehicle Trajectory**

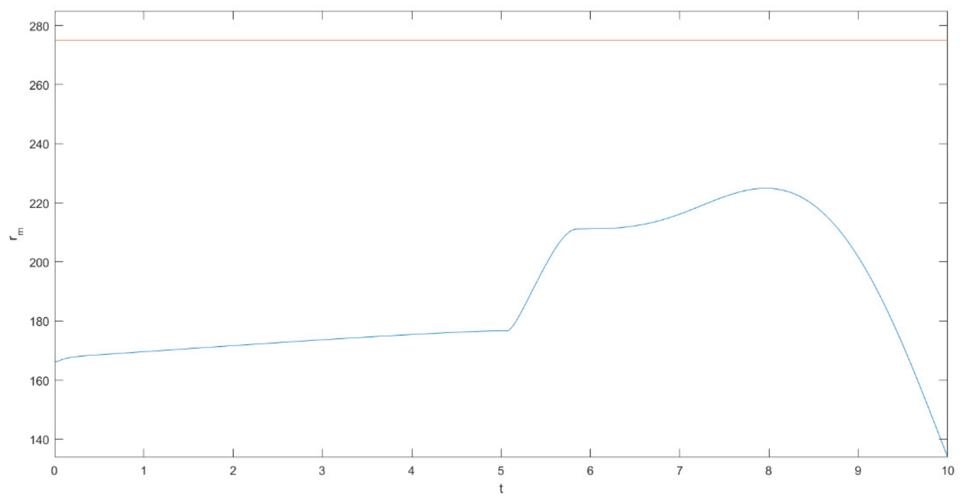


**Figure 5.13 Plot of $r_m$ vs. *t*; Altered Conflict Vehicle Trajectory**

## 5.3 Altered Target Vehicle Trajectory

The next simulation run to be considered is a case where a command for an instantaneous jump in $X$ was given to the simulated target vehicle. A jump of $+100\ d$ was given at $2\ s$. The purpose of this case is to show that the CDR algorithm has complete control over the commanded trajectory vehicle until after the PCA has been passed. Figures 5.14 and 5.15 show the 3D and 2D views of this run.

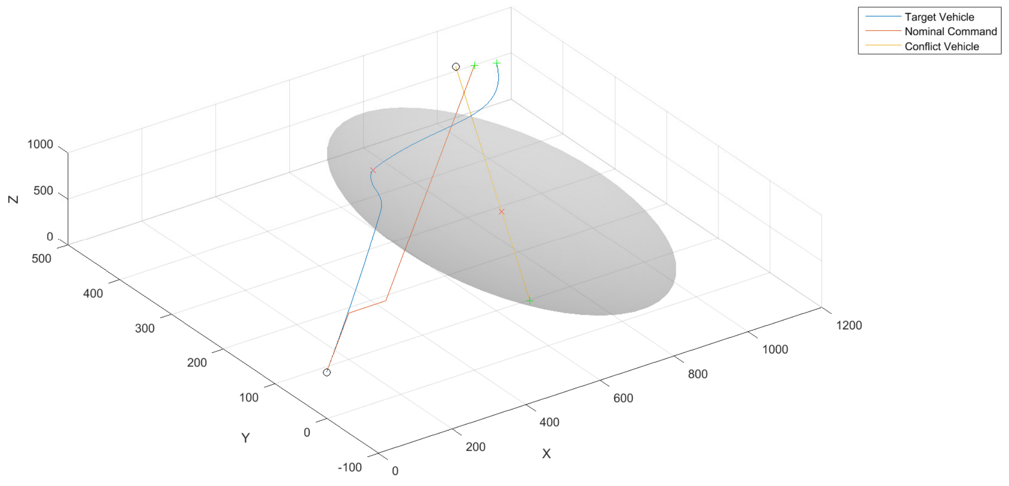


**Figure 5.14 3D View; Instantaneous Jump in *X***

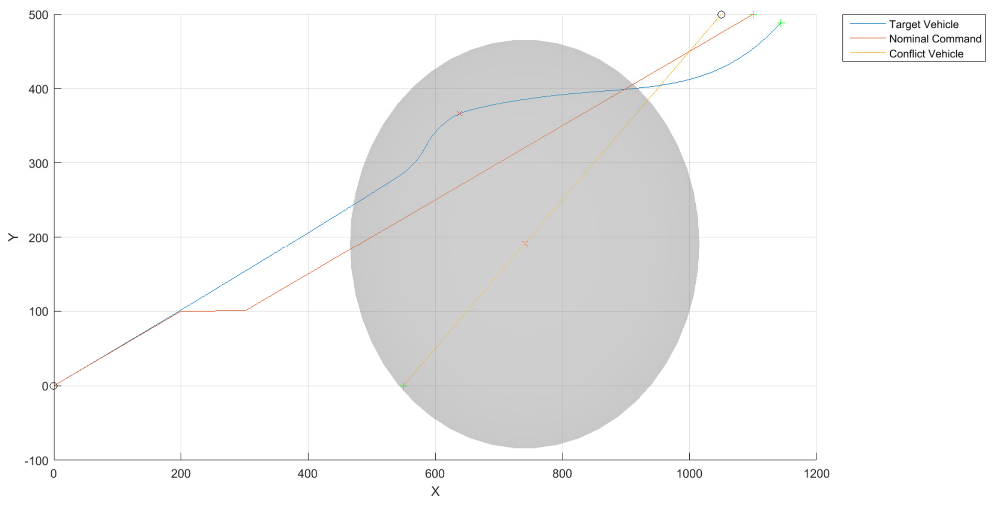


**Figure 5.15 2D View; Instantaneous Jump in *X***

Even after the instantaneous jump command has been given, the vehicle continues on the trajectory provided by the CDR algorithm and ignores and user-submitted commands. This case shows that the simulator is properly recognizing that a conflict exists and is ignoring the user-commanded trajectory. The $\tau_m$ and $r_m$ vs. $t$ graphs presented in Figures 5.16 and 5.17 show similar results to those seen above. The $r_m$ value at the PCA is 215.95 with an error of 21.47% from the required value.

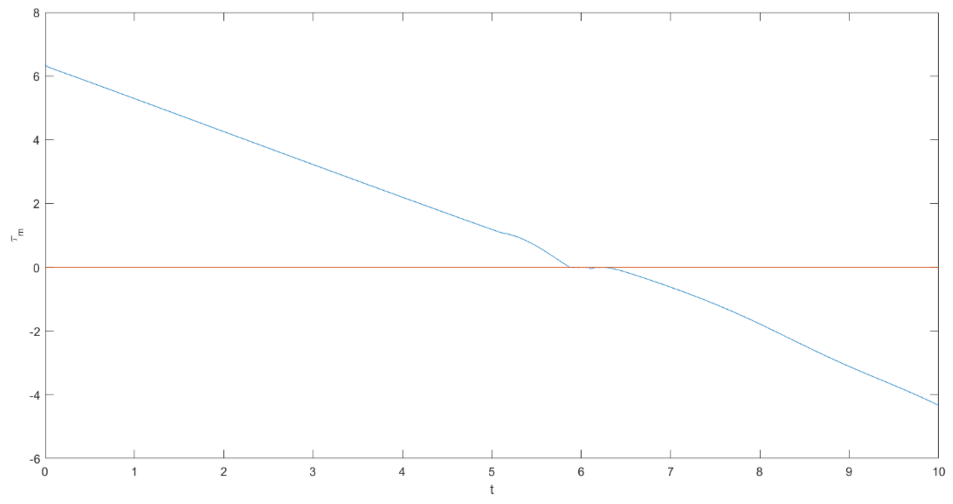


**Figure 5.16 Plot of $\tau_m$ vs. $t$; Instantaneous Jump in $X$**

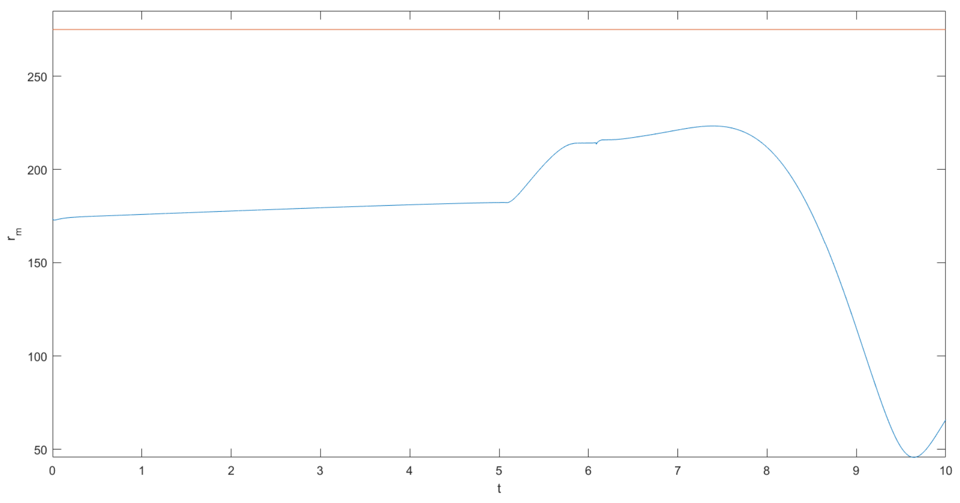


**Figure 5.17 Plot of $r_m$ vs. $t$; Instantaneous Jump in $X$**

The next case presented will be using a curved 3D trajectory in the X direction. This case will evaluate the simulator's ability to respond to a traffic conflict with a curved commanded trajectory. The curvilinear behavior is generated with $sin$ function as discussed in Section 4.2.

Figures 5.18 and 5.19 show the 3D and 2D views of the simulation. The simulated vehicle continues to follow the curved command trajectory while later exhibiting the same behavior described in the trajectory above. The algorithm is unable to catch up to the desired PCA value and attempts to quickly correct it.

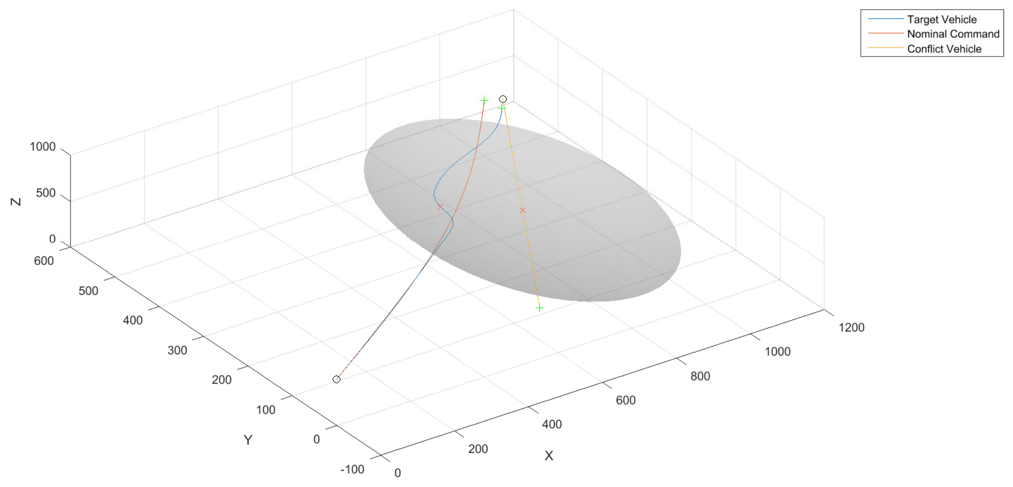


**Figure 5.18 3D View; Curve in *X***

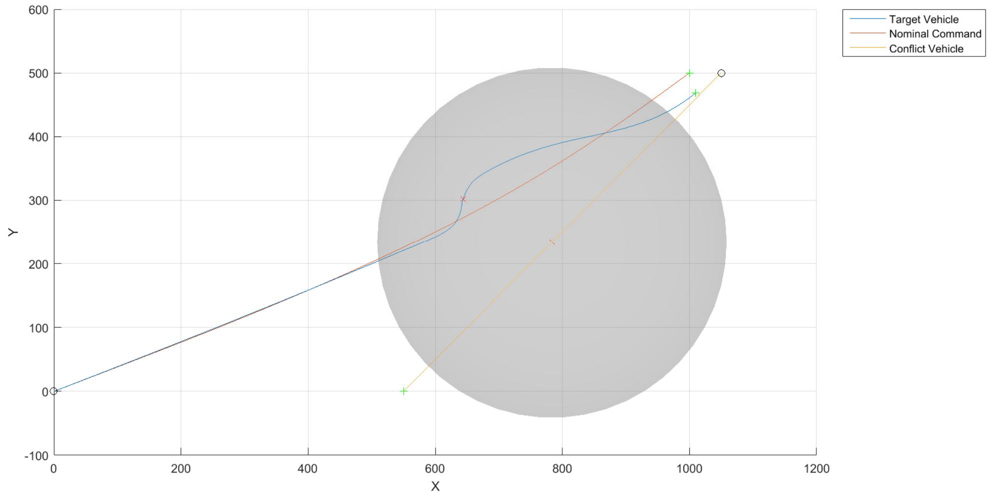


**Figure 5.19 2D View; Curve in *X***

Figures 5.20 and 5.21 below show the $\tau_m$ and $r_m$ vs. $t$ graphs. It is interesting to note that the irregularity in the $\tau_m$ graph at around the 4 $s$ mark matches up with the extreme jump seen in the $r_m$ graph. These events most likely coincide with the extreme behavior noted in the 3D and 2D views of the simulation. A similar behavior was noted in previous figures, however this case is the most extreme instance of it. $\tau_m$ still exhibits the chattering effect as it passes the PCA. The $r_m$ value at the PCA is 155.49 and results in an error of 43.46% from the required PCA value.

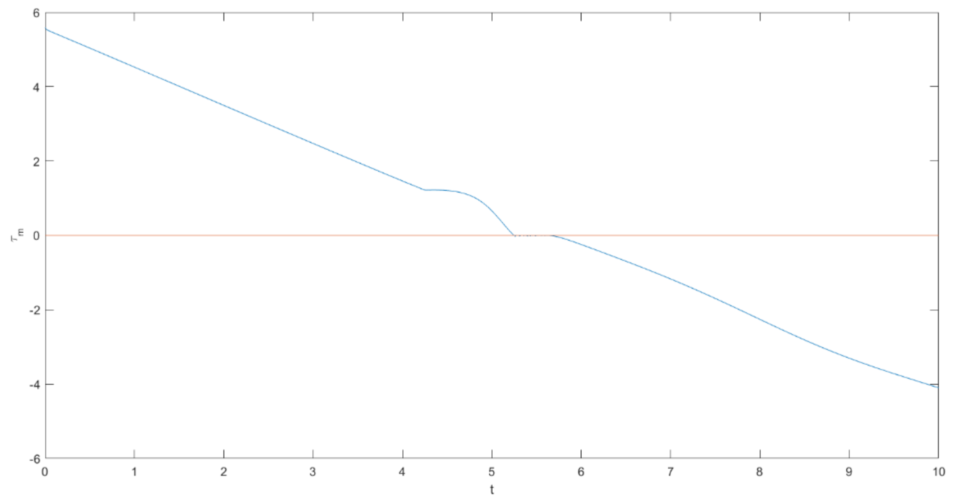


**Figure 5.20 Plot of $\tau_m$ vs. *t* for 3D Trajectory with Curve in *X***

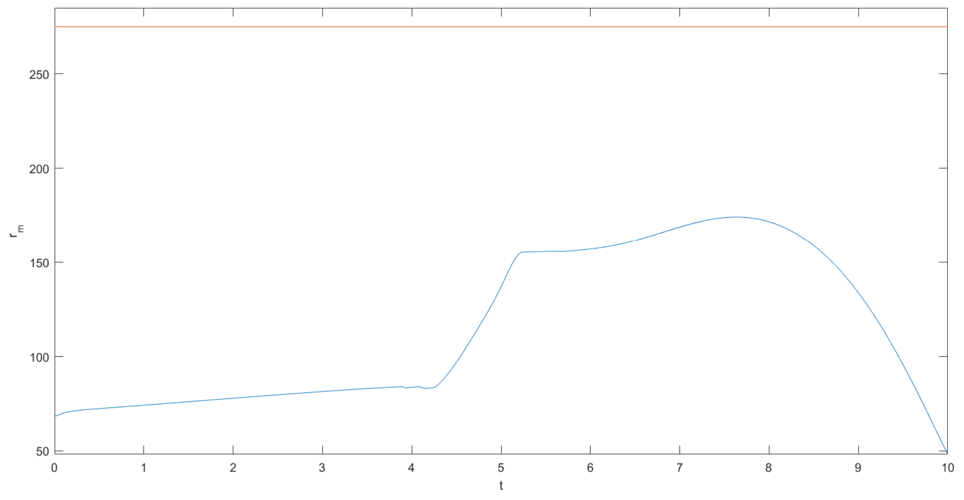


**Figure 5.21 Plot of $r_m$ vs. *t* for 3D Trajectory with Curve in *X***

The next case to be studied will be a simulation run using a 3D trajectory with both a curve in X and a curve in Y. Figures 5.22 and 5.23 show the 3D and 2D views of the run. This run is more promising as it does push the PCA closer towards the required value. A similar behavior to that above is noted where the vehicle experiences an extreme jump in the command given from the CDR algorithm. The increase in the ability of the CDR algorithm to meet the PCA requirement is likely due to the fact that the initial commanded trajectory is already headed towards the direction that the vehicle needs to go to meet the PCA requirement.

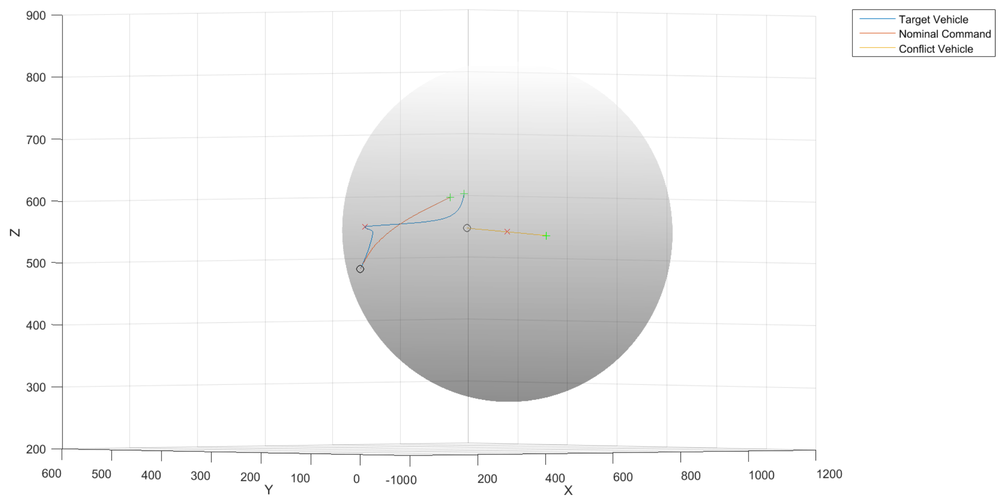


**Figure 5.22 3D View of 3D Trajectory with Curve in *X* and *Y***

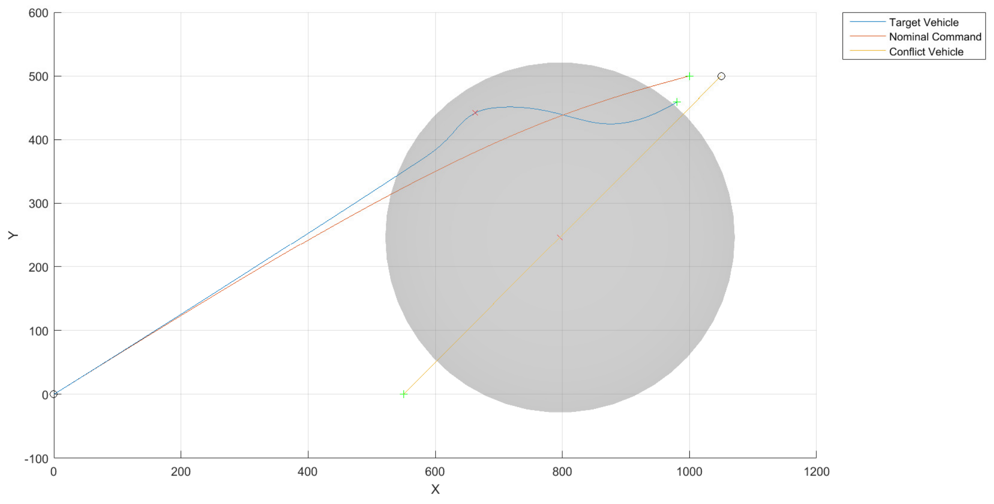


**Figure 5.23 2D View of 3D Trajectory with Curve in *X* and *Y***

Figures 5.24 and 5.25 show the $\tau_m$ and $r_m$ vs. $t$ plots for this simulation. This scenario is an interesting case to study because it shows a $\tau_m$ graph that is linear with a constant decreasing slope while $\tau_m$ is positive and after $\tau_m$ becomes negative, the graph is no longer linear. It is important to point out that this is not an issue because anything after $\tau_m$ becomes negative is post-PCA and not of high interest. The vehicle has already passed the PCA and would have to go "back in time" in order for behavior this to have any effect on the CDR. In fact, after $\tau_m$ becomes negative the CDR has turned off. This action likely means that the irregular shape of the $\tau_m$ graph after it becomes negative is due to the controller attempting to bring the vehicle back to the user-submitted command trajectory. In this case, the command trajectory is a curve and this likely is the cause of the curved shape of the second part of the $\tau_m$ graph. The $r_m$ value at the PCA is 236.62 which results in an error of 13.96% from the required value.

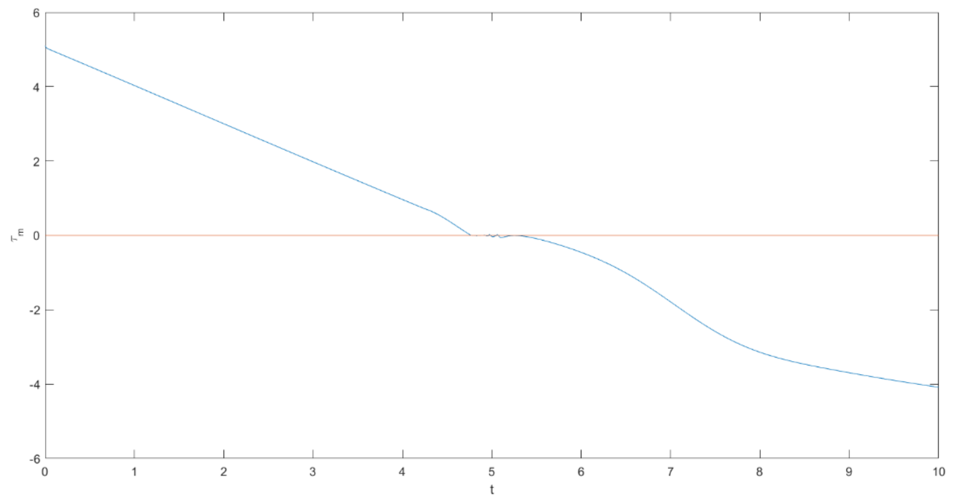


**Figure 5.24 Plot of $\tau_m$ vs. *t* for 3D Trajectory with Curve in *X* and *Y***

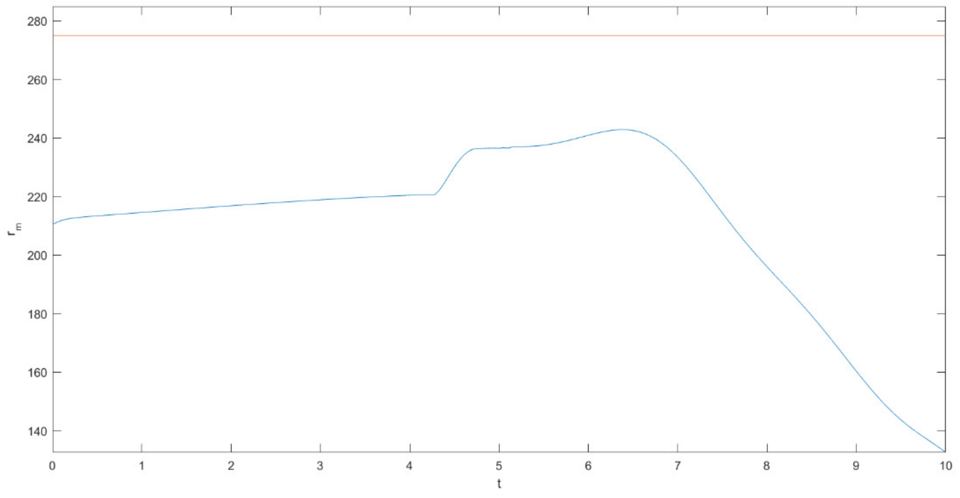


**Figure 5.25 Plot of $r_m$ vs. *t* for 3D Trajectory with Curve in *X* and *Y***

# CHAPTER 6

# CONCLUSIONS AND RECOMMENDATIONS

The purpose of this thesis was to develop and assess a PCA algorithm that was used to analyze air traffic CDR problems. Research into detecting and resolving air traffic conflicts before collisions or near misses occur is currently a very active area of research. Previous work has been done to develop on-board and ground-based systems on manned aircraft to detect and resolve air traffic conflicts both in controlled and uncontrolled airspace. The FAA has created guidelines for operating UAS in controlled airspaces but has put strict regulations on operating them. With the rise in popularity of UAS and increasing research into AUAS, it is desired to develop CDR methods in order to aid the integration of these aerial systems into the NAS. Previous works focused on CDR algorithms incorporating a PCA method that required both the target vehicle and the conflict vehicle to be willing to share resolution power. A key assumption in this thesis is that the conflict vehicle is unwilling or unable to try to resolve the conflict. A different set of geometrical equations were developed to solve for the new PCA that resolves the conflict. To solve this set of simultaneous equations, a Newton-Raphson iteration method was used. A 3DOF point-mass vehicle model was developed in order to simulate air traffic problems to test the effectiveness of the CDR algorithm developed. MathWorks' Simulink was used to simulate this model.

First, a rotation matrix was developed to transform the vehicle's Cartesian velocity, $\dot{X}, \dot{Y}, \dot{Z}$, in the inertial frame into a more useable velocity frame spherical description, $V, \chi, \gamma$, that is centric to the vehicle perspective. The 3DOF simulated vehicle is able to take commands in this frame and integrate and transform these commands into the inertial frame to more accurately describe the vehicle's position to any ground-based or on-board system. Next, an open-loop simulator was created in Simulink to test the ability of the 3DOF model to accurately propagate the trajectory of the vehicle forward in time. The open-loop model is quite capable of following $V, \chi, \gamma$ commands and is only limited by the values for the time delay constants used. This model validates the feedback linearization of the nonlinear 3DOF ODE kinetic EOM.

Next, a closed-loop model was developed to simulate the vehicle's trajectory under the influence of closed-loop tracking control. A negative feedback model implementing PID signal processing was developed. PID control was chosen because it is an efficient and proven method where changes to controller gains produce responses that are relatively easy to predict. The closed-loop model accepts inertial trajectory commands, forms trajectory errors, converts them to velocity frame commands, and then propagates the vehicle in the simulator. The errors in the inertial frame are transformed into the velocity frame before being acted on by the PID controller. The closed-loop model was sufficiently capable of simulating linear 2D and 3D trajectories, curved trajectories, and trajectories that included instantaneous jumps as well as combinations of any of these inputs. This model was deemed capable enough to work alongside the CDR algorithm.

Finally, the CDR algorithm was incorporated into the closed-loop simulator. Different combinations of both target and conflict vehicle trajectories were tested to assess the effectiveness of the CDR algorithm. Both 2D and 3D trajectories were tested. The algorithm had an issue from the first trajectory tested where it was unable to properly push the PCA far enough out to satisfy the required safe distance. The first test resulted in a small error that could be deemed acceptable, however further tests were conducted. With successive tests, the error increased. These errors proved to be less acceptable for use in more extensive and complicated tests. Each trajectory tested also exhibited some form of a phenomenon where initially the CDR algorithm was not commanding a trajectory that could properly meet the PCA requirements. This behavior resulted in a period of time before the PCA was reached where the algorithm had to catch up by executing an extreme maneuver in an attempt to avoid the conflict. This behavior could be a cause of the algorithm's inability to command a trajectory that properly satisfies the PCA requirements. It is believed that the CDR algorithm is solving for the correct PCA, however the commanded trajectory is not effectively steering the vehicle to this calculated point. An attempt to adjust the PID control gains could result in the vehicle being able to more precisely follow the path being commanded to the vehicle. Another solution could be to artificially inflate the $r_s$ value being used to solve for the conflict resolution point to potentially avoid any conflict.

Future work into developing a CDR algorithm using this PCA method should focus on minimizing the error between the actual and the required PCA. The commanded trajectory coming from the algorithm should also be examined to determine if it is properly steering the vehicle towards a PCA that meets the requirements. After this is completed, the power of this method can be properly explored. Using this PCA method allows for more than just one conflict vehicle to be put into the airspace. The relative miss distances and miss times can be compared and the one that will cause a conflict first can be avoided. The algorithm still suffers from only being able to solve for the PCA to resolve the conflict with a single vehicle. Research should be conducted into adapting this algorithm to be able to resolve conflicts with a large amount of traffic. Other areas for enhancement include introduction of maneuver constraints originating from physical limits or software imposed limits, conflict vehicle state uncertainty, and implementation of the true point-mass dynamics with feedback linearizing controller.

## REFERENCES

1. *Pilot's Handbook of Aeronautical Knowledge*, United States Department of Transportation, Federal Aviation Administration, Airman Testing Standards Branch, Oklahoma City, Oklahoma, 2016.

2. *General Operating and Flight Rules*, 14 CFR Part 91, 2013.

3. Knight, C., Knight, K. S., *Plane Crash! The Mysteries of Major Air Disasters and How They Were Solved*, Greenberg Publishing, New York, New York, 1958.

4. Nolan, M.S., *Fundamentals of Air Traffic Control*, Delmar Cengage Learning, Clifton Park, New York, 2009.

5. Isaacson, D. R. and Erzberger, H., *Design of a Conflict Detection Algorithm for the Center/TRACON Automation System*, Proceedings of the IEEE-AIAA 16th Digital Avionics Systems Conference, New York, New York, October, 1997.

6. *Introduction to TCAS II Version 7.1*, Federal Aviation Administration, February 2011.

7. *Collision of Aeronaves De Mexico, S.A., McDonnel Douglas DC-9-32, XA-JED and Piper PA-28-181, N4891F*, Cerritos, California, August 31, 1986, Aircraft Accident Report NTSB/AAR-87/07, National Transportation Safety Board, Washington, DC, July, 1987.

8. *Midair Collision Over Hudson River, Piper PA-32R-300, N71MC and Eurocopter AS350BA, N401LH Near Hoboken, New Jersey, August 8, 2009*, Aircraft Accident Summary Report NTSB/AAR-10/05. National Transportation Safety Board, Washington, DC, September 2010.

9. *Operation and Certification of Small Unmanned Aircraft Systems*, 81 FR 42063, 2016.

10. Erzberger, H. and Tobias, L., *A Time-Based Concept for Terminal-Area Traffic Management*, NASA-TM-88243, NASA-Ames Research Center, Moffett Field, California, April, 1986.

11. Slattery, R. and Zhao, Y., *Trajectory Synthesis for Air Traffic Automation*, Journal of Guidance, Control, and Dynamics, Vol. 20, No. 2, March-April, 1997, pp. 232-238.

12. Vinh, N., *Flight Mechanics of High-Performance Aircraft*, Cambridge University Press, Cambridge, England, 1993.

13. Hull, D., *Fundamentals of Airplane Flight Mechanics*, Springer-Verlag, Berlin, Germany, 2007.

14. Merz, A. W., *Maximum-Miss Aircraft Collision Avoidance*, Dynamics and Control, vol. 1, No. 1, March, 1991, pp. 25-34.

15. Krozel, J. and Peters, M., *Strategic Conflict Detection and Resolution for Free Flight*, Proceedings of the 36th IEEE Conference on Decision and Control, San Diego, California, December 1997.

16. Park, J.-W. Oh, H-D. and Tahk, M.-J., *UAV Collision Avoidance Based on Geometric Approach*. Proceeding of the 2008 SICE Annual Conference, Tokyo, Japan, august, 2008 pp. 2122-2126.

17. Hughes, D., *New Age Anti-Collision, UAS in Civil Airspace: Problem or Opportunity,* Aviation Week & Space Technology, Vol. 169, No. 2, 14 July, 2008.

18. Papelis, Y., Newman, B., Iftekharuddin, K., Quach, P., and Ballin, M., *On The Feasibility of Using Sub-Scale Aerial Vehicles For Testing Full-Scale NAS Concepts*, AUVSI Unmanned System,s 2015 Conference, Atlanta, Georgia, May 2015.

19. Stevens, B.  L. and Lewis, F.  L., *Aircraft Control and Simulation*, John Wiley & Sons, New York, New York, 1992.

20. Meriam, J. L., *Dynamics*, *Engineering Mechanics*, John Wiley & Sons, New York, New York, 1978.

21. The MathWorks, Inc., *Siumulink User's Guide*, Natick, Massachusetts, 2016.

# APPENDICES

## APPENDIX A

## NEWTON-RAPHSON FUNCTION EQUATIONS

$$f_1 = \left(\vec{r}_r \times \hat{\epsilon}_{V_r}\right) \cdot \hat{\epsilon}_{V_r'}$$

$$= \left(r_{r_Y}\epsilon_{V_{rZ}} - r_{r_Z}\epsilon_{V_{rY}}\right)\epsilon'_{V_{rX}} + \left(r_{r_Z}\epsilon_{V_{rX}} - r_{r_X}\epsilon_{V_{rZ}}\right)\epsilon_{V_{rY}'} + \left(r_{r_X}\epsilon_{V_{rY}} - r_{r_Y}\epsilon_{V_{rX}}\right)\epsilon_{V'_{r_Z}}$$

$$f_2 = \left\|\hat{\epsilon}_{V_r'}\right\|^2 - 1$$

$$= \epsilon^2_{V'_{r_X}} + \epsilon^2_{V'_{r_Y}} + \epsilon^2_{V'_{r_Z}} - 1$$

$$f_3 = \left\|\hat{\epsilon}_{V_r'} \times \left(\vec{r}_r \times \hat{\epsilon}_{V_r'}\right)\right\|^2 - r_s^2$$

$$= \left(r_{r_X}\epsilon^2_{V'_{r_Y}} - r_{r_Y}\epsilon_{V'_{r_X}}\epsilon_{V'_{r_Y}} + r_{r_X}\epsilon^2_{V'_{r_Z}} - r_{r_Z}\epsilon_{V'_{r_X}}\epsilon_{V'_{r_Z}}\right)^2 + \left(r_{r_Y}\epsilon^2_{V'_{r_X}} - r_{r_X}\epsilon_{V'_{r_X}}\epsilon_{V'_{r_Y}} + r_{r_Y}\epsilon^2_{V'_{r_Z}} - r_{r_Z}\epsilon_{V'_{r_Y}}\epsilon_{V'_{r_Z}}\right)^2 + \left(r_{r_Z}\epsilon^2_{V'_{r_X}} - r_{r_X}\epsilon_{V'_{r_Z}}\epsilon_{V'_{r_X}} + r_{r_Z}\epsilon^2_{V'_{r_Y}} - r_{r_Y}\epsilon_{V'_{r_Y}}\epsilon_{V'_{r_Z}}\right)^2 - r_s^2$$

$$\frac{\partial f_1}{\partial x_1} = r_{r_Y}\epsilon_{V_{rZ}} - r_{r_Z}\epsilon_{V_{rY}}$$

$$\frac{\partial f_1}{\partial x_2} = r_{r_Z}\epsilon_{V_{rX}} - r_{r_X}\epsilon_{V_{rZ}}$$

$$\frac{\partial f_1}{\partial x_3} = r_{r_X}\epsilon_{V_{rY}} - r_{r_Y}\epsilon_{V_{rX}}$$

$$\frac{\partial f_2}{\partial x_1} = 2\epsilon_{V'_{r_X}}$$

$$\frac{\partial f_2}{\partial x_2} = 2\epsilon_{V'_{r_Y}}$$

$$\frac{\partial f_2}{\partial x_3} = 2\epsilon'_{V_{rZ}}$$

$$f_{3_{n_1}} = r_{r_X}\epsilon^2_{V'_{r_Y}} - r_{r_Y}\epsilon_{V'_{r_X}}\epsilon_{V'_{r_Y}} + r_{r_X}\epsilon^2_{V'_{r_Z}} - r_{r_Z}\epsilon_{V'_{r_X}}\epsilon_{V'_{r_Z}}$$

$$f_{3_{n_2}} = r_{r_Y}\epsilon^2_{V'_{r_X}} - r_{r_X}\epsilon_{V'_{r_X}}\epsilon_{V'_{r_Y}} + r_{r_Y}\epsilon^2_{V'_{r_Z}} - r_{r_Z}\epsilon_{V'_{r_Y}}\epsilon_{V'_{r_Z}}$$

$$f_{3_{n_3}} = r_{r_Z}\epsilon^2_{V'_{r_X}} - r_{r_X}\epsilon_{V'_{r_Z}}\epsilon_{V'_{r_X}} + r_{r_Z}\epsilon^2_{V'_{r_Y}} - r_{r_Y}\epsilon_{V'_{r_Y}}\epsilon_{V'_{r_Z}}$$

$$\frac{\partial f_3}{\partial x_1} = 2\left[\left(-r_{r_Y}\epsilon_{V'_{r_Y}} - r_{r_Z}\epsilon_{V'_{r_Z}}\right)f_{3_{n_1}} + \left(2r_{r_Y}\epsilon_{V'_{r_X}} - r_{r_X}\epsilon_{V'_{r_Y}}\right)f_{3_{n_2}} + \left(2r_{r_Z}\epsilon_{V'_{r_X}} - r_{r_X}\epsilon_{V'_{r_Z}}\right)\right]$$

$$\frac{\partial f_3}{\partial x_2} = 2\left[\left(2r_{r_X}\epsilon_{V'_{r_Y}} - r_{r_Y}\epsilon_{V'_{r_X}}\right)f_{3_{n_1}} + \left(-r_{r_X}\epsilon_{V'_{r_X}} - r_{r_Z}\epsilon_{V'_{r_Z}}\right)f_{3_{n_2}} + \left(2r_{r_Z}\epsilon_{V'_{r_Y}} - r_{r_Y}\epsilon_{V'_{r_Z}}\right)\right]$$

$$\frac{\partial f_3}{\partial x_3} = 2\left[\left(2r_{r_X}\epsilon_{V'_{r_Z}} - r_{r_Z}\epsilon_{V'_{r_X}}\right)f_{3_{n_1}} + \left(2r_{r_Y}\epsilon_{V'_{r_Z}} - r_{r_Z}\epsilon_{V'_{r_Y}}\right)f_{3_{n_2}} + \left(-r_{r_X}\epsilon_{V'_{r_X}} - r_{r_Y}\epsilon_{V'_{r_Y}}\right)\right]$$

# APPENDIX B

# MATLAB CODE

## Open-Loop

```
%Building Trajectories
close all %Closing any figures still open

%Defining three tau constants
tau_v     = 1;
tau_chi   = 1;
tau_gamma = 1;
t = 10; %Setting length of simulation
h = 1001;%Number of points in trajectory
hh = t/h; %Creating time-step
time = linspace(0,t,h); %Forming a time vector
ht = length(time);

%Creating V, chi, gamma command trajectories
v_comm = 100*ones(1,1001) + 0*[zeros(1,200) 10*ones(1,801)];
chi_comm = zeros(1,1001) + 0*[zeros(1,200) pi/6*ones(1,100) -pi/6*ones(1,100) zeros(1,601)];
gamma_comm = zeros(1,1001) + 1*[zeros(1,200) pi/4*ones(1,100) -pi/4*ones(1,100) zeros(1,601)];

V = v_comm(1);
chi_c = chi_comm(1);
gamma_c = gamma_comm(1);

%Simulink takes trajectories in the form of a timeseries
%Forming timeseries to be fed to simulink
TrajV = timeseries(v_comm,time);
TrajChi = timeseries(chi_comm,time);
TrajGamma = timeseries(gamma_comm,time);
```

## Closed-Loop

```
%Clearing any previous trajectory data
clear TrajXA TrajYA TrajZA TrajVAX TrajVAY TrajVAZ TrajXB TrajYB TrajZB TrajVBX TrajVBY TrajVBZ
%%Building Trajectories
close all %Closing any figures still open

%Defining three tau constants
tau_v     = 0.1;
tau_chi   = 0.1;
tau_gamma = 0.1;
t = 10; %Setting length of simulation
h = 1001; %Number of points in trajectory
hh = t/h; %Creating time-step
time = linspace(0,t,h); %Forming a time vector
ht = length(time);
r_s = 275; %Setting radius of safe-sphere

VAX = NaN(1,ht); %Pre-allocating memory for velocity data
VAY = VAX; VAZ = VAX;
VBX = VAX; VBY = VAY; VBZ = VAZ;
%Building trajectory for traget, A, vehicle
Xf = 1000;
Yf = 500;
Zi = 500;
Zf = 500;
XA = linspace(0,1000,ht) + 0*[zeros(1,201) 100*ones(1,800)] + 0*(100*sin(linspace(0,pi,1001)));
     %Overall length       Creates instantaneous jump         Creates curve
YA = linspace(0,500,ht)  + 0*[zeros(1,401) 100*ones(1,600)] + 0*(100*sin(linspace(0,pi,1001)));
ZA = linspace(Zi,Zf,ht)  + 0*[zeros(1,251) 10*ones(1,750)] + 0*(100*sin(linspace(0,pi,1001)));
%Simulink takes trajectories in the form of a timeseries
TrajXA = timeseries(XA,time);
TrajYA = timeseries(YA,time);
TrajZA = timeseries(ZA,time);
%Building trajectory for conflict, B, vehicle
XB = linspace(0,1000,ht);
YB = linspace(500,0,ht);
ZB = linspace(Zi+50,Zi+50,ht);

TrajXB = timeseries(XB,time);
TrajYB = timeseries(YB,time);
TrajZB = timeseries(ZB,time);

%Creating velocities using finite difference method
for i = 1:ht-1
    VAX(i) = (XA(i+1)-XA(i))/hh;
    VAY(i) = (YA(i+1)-YA(i))/hh;
    VAZ(i) = (ZA(i+1)-ZA(i))/hh;
    VBX(i) = (XB(i+1)-XB(i))/hh;
    VBY(i) = (YB(i+1)-YB(i))/hh;
    VBZ(i) = (ZB(i+1)-ZB(i))/hh;
end
```

```
%Assuming velocity at n+1 is same as velocity at n
VAX(i+1) = VAX(i);
VAY(i+1) = VAY(i);
VAZ(i+1) = VAZ(i);
VBX(i+1) = VBX(i);
VBY(i+1) = VBY(i);
VBZ(i+1) = VBZ(i);

%Forming the timeseries for velocity data to be fed to simulink
TrajVAX = timeseries(VAX,time);
TrajVAY = timeseries(VAY,time);
TrajVAZ = timeseries(VAZ,time);
TrajVBX = timeseries(VBX,time);
TrajVBY = timeseries(VBY,time);
TrajVBZ = timeseries(VBZ,time);

%Calculating initial V, chi, gamma values
V = sqrt(VAX(1)^2 + VAY(1)^2 + VAZ(1)^2);
chi_c = atan2(VAY(1),VAX(1));
gamma_c = atan2(VAZ(1),sqrt(VAX(1)^2 + VAY(1)^2));
```

## CDR Algorithm

```
%Clearing any previous trajectory data
clear TrajXA TrajYA TrajZA TrajVAX TrajVAY TrajVAZ TrajXB TrajYB TrajZB TrajVBX TrajVBY TrajVBZ
%%Building Trajectories
close all %Closing any figures still open

%Defining three tau constants
tau_v     = 0.1;
tau_chi   = 0.1;
tau_gamma = 0.1;
t = 10; %Setting length of simulation
h = 1001; %Number of points in trajectory
hh = t/h; %Creating time-step
time = linspace(0,t,h); %Forming a time vector
ht = length(time);
r_s = 275; %Setting radius of safe-sphere

VAX = NaN(1,ht); %Pre-allocating memory for velocity data
VAY = VAX; VAZ = VAX;
VBX = VAX; VBY = VAY; VBZ = VAZ;
%Building trajectory for traget, A, vehicle
Xf = 1000;
Yf = 500;
Zi = 500;
Zf = 500;
XA = linspace(0,1000,ht) + 0*[zeros(1,201) 100*ones(1,800)] + 0*(100*sin(linspace(0,pi,1001)));
```

```
     %Overall length       Creates instantaneous jump         Creates curve
YA = linspace(0,500,ht)  + 0*[zeros(1,401) 100*ones(1,600)] + 0*(100*sin(linspace(0,pi,1001)));
ZA = linspace(Zi,Zf,ht)  + 0*[zeros(1,251) 10*ones(1,750)] + 0*(100*sin(linspace(0,pi,1001)));
%Simulink takes trajectories in the form of a timeseries
TrajXA = timeseries(XA,time);
TrajYA = timeseries(YA,time);
TrajZA = timeseries(ZA,time);
%Building trajectory for conflict, B, vehicle
XB = linspace(0,1000,ht);
YB = linspace(500,0,ht);
ZB = linspace(Zi+50,Zi+50,ht);

TrajXB = timeseries(XB,time);
TrajYB = timeseries(YB,time);
TrajZB = timeseries(ZB,time);

%Creating velocities using finite difference method
for i = 1:ht-1
    VAX(i) = (XA(i+1)-XA(i))/hh;
    VAY(i) = (YA(i+1)-YA(i))/hh;
    VAZ(i) = (ZA(i+1)-ZA(i))/hh;
    VBX(i) = (XB(i+1)-XB(i))/hh;
    VBY(i) = (YB(i+1)-YB(i))/hh;
    VBZ(i) = (ZB(i+1)-ZB(i))/hh;
end
%Assuming velocity at n+1 is same as velocity at n
VAX(i+1) = VAX(i);
VAY(i+1) = VAY(i);
VAZ(i+1) = VAZ(i);
VBX(i+1) = VBX(i);
VBY(i+1) = VBY(i);
VBZ(i+1) = VBZ(i);

%Forming the timeseries for velocity data to be fed to simulink
TrajVAX = timeseries(VAX,time);
TrajVAY = timeseries(VAY,time);
TrajVAZ = timeseries(VAZ,time);
TrajVBX = timeseries(VBX,time);
TrajVBY = timeseries(VBY,time);
TrajVBZ = timeseries(VBZ,time);

%Calculating initial V, chi, gamma values
V = sqrt(VAX(1)^2 + VAY(1)^2 + VAZ(1)^2);
chi_c = atan2(VAY(1),VAX(1));
gamma_c = atan2(VAZ(1),sqrt(VAX(1)^2 + VAY(1)^2));
```

# APPENDIX C

## SUBSYSTEM FIGURES

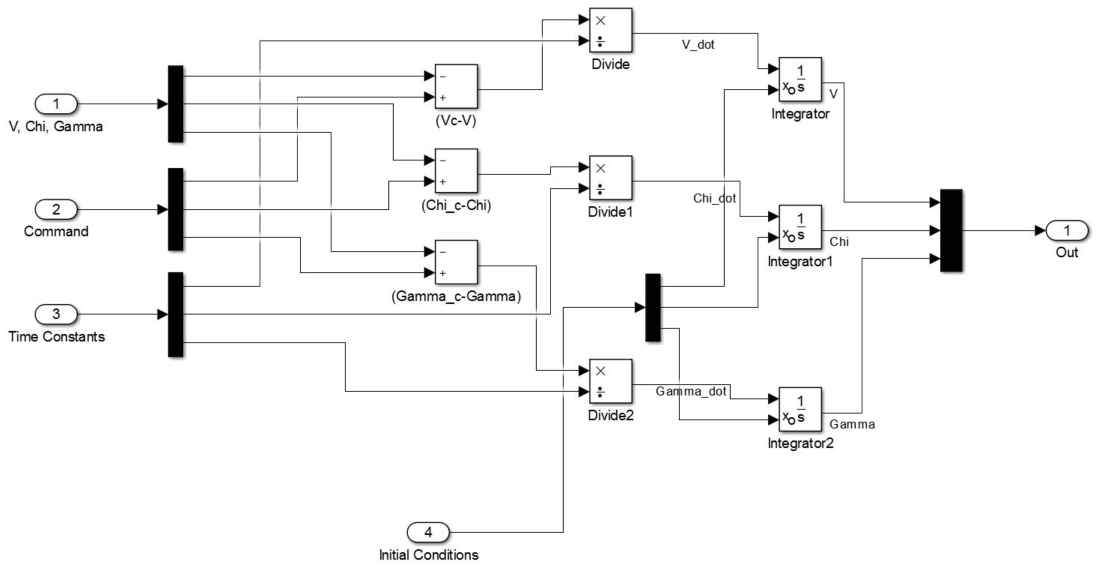


Figure C.1 Wiring for EOM

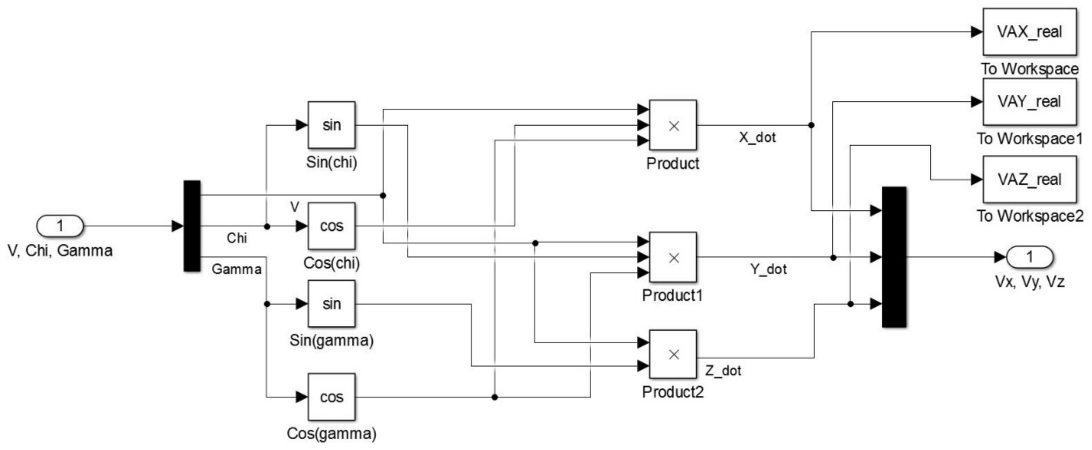


Figure C.2 Transforming $V, \chi, \gamma$ to form $\dot{X}, \dot{Y}, \dot{Z}$

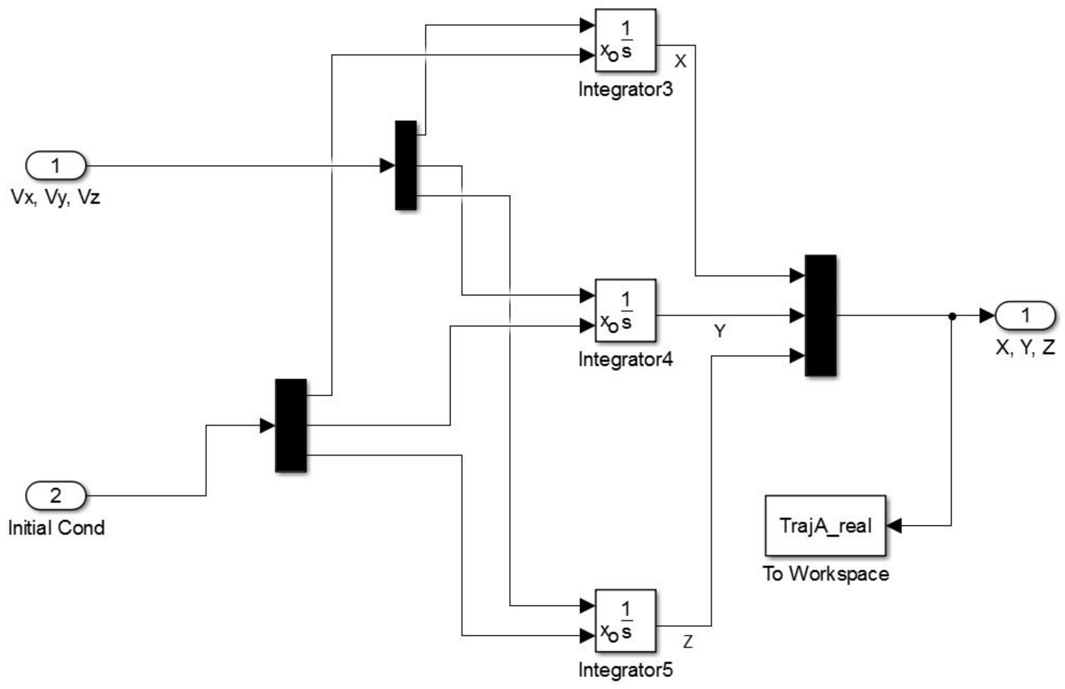


**Figure C.3 Integrating $\dot{X}, \dot{Y}, \dot{Z}$ to form $X, Y, Z$**

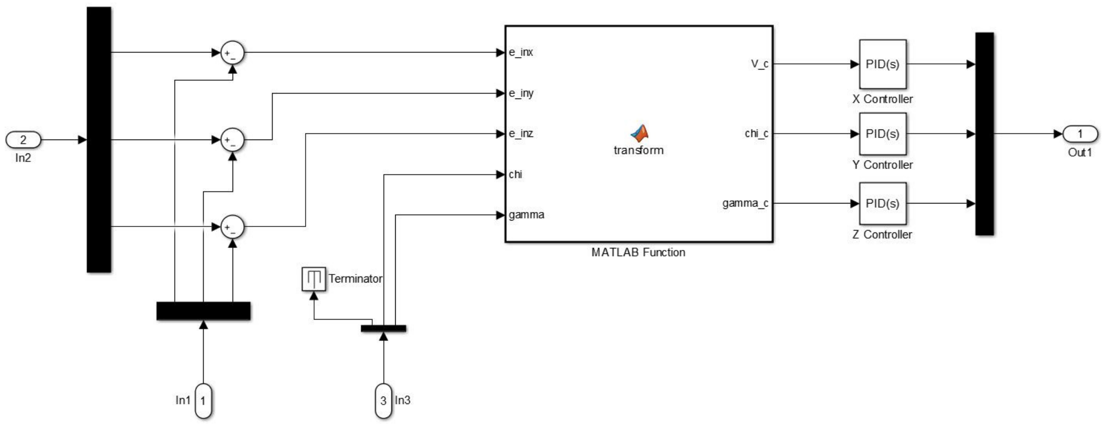


**Figure C.4 Wiring for PID Controller**

# VITA

Marc S. Easton received his Bachelor of Science in Aerospace Engineering from Virginia Polytechnic Institute and State University in May, 2013. While there, he was an active member of the Virginia Tech Corps of Cadets and served in the regimental marching band, the Highty Tighties. He also served in numerous leadership roles and led the band in exemplary performances on the national stage in various events. He will complete his Master of Science at Old Dominion University in August 2016 with a concentration in control systems engineering. He will be starting employment with Naval Surface Warfare Center Dahlgren Division as a research engineer following graduation.