Kill Zone Analysis for a Bank-to-Turn

Missile-Target Engagement

by

Venkatraman Renganathan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2016 by the
Graduate Supervisory Committee:

Armando A. Rodriguez, Chair
Panagiotis Artemiadis
Spring Melody Berman

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

With recent advances in missile and hypersonic vehicle technologies, the need for being able to accurately simulate missile-target engagements has never been greater. Within this research, we examine a fully integrated missile-target engagement environment. A MATLAB based application is developed with 3D animation capabilities to study missile-target engagement and visualize them. The high fidelity environment is used to validate miss distance analysis with the results presented in relevant GNC textbooks [51], [52] and to examine how the kill zone varies with critical engagement parameters; e.g. initial engagement altitude, missile Mach, and missile maximum acceleration. A ray-based binary search algorithm is used to estimate the kill zone region; i.e. the set of initial target starting conditions such that it will be "killed". The results show what is expected. The kill zone increases with larger initial missile Mach and maximum acceleration & decreases with higher engagement altitude and higher target Mach. The environment is based on (1) a 6DOF bank-to-turn (BTT) missile, (2) a full aerodynamic-stability derivative look up tables ranging over Mach number, angle of attack and sideslip angle (3) a standard atmosphere model, (4) actuator dynamics for each of the four cruciform fins, (5) seeker dynamics, (6) a nonlinear autopilot, (7) a guidance system with three guidance algorithms (i.e. PNG, optimal, differential game theory), (8) a 3DOF target model with three maneuverability models (i.e. constant speed, Shelton Turn & Climb, Riggs-Vergaz Turn & Dive). Each of the subsystems are described within the research. The environment contains linearization, model analysis and control design features. A gain scheduled nonlinear BTT missile autopilot is presented here. Autopilot got sluggish as missile altitude increased and got aggressive as missile mach increased. In short, the environment is shown to be a very powerful tool for conducting missile-target engagement research - a research that could address multiple missiles and advanced targets.

Dedicated to my parents and the loving memory of my brother Ravi

ACKNOWLEDGMENTS

I want to thank the almighty for his blessings. First, I would like to express my sincere gratitude to my MS thesis advisor Dr. Armando A. Rodriguez for showing confidence in my work, his continuous motivation and support for my research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me throughout the course of the research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Masters studies. Besides my thesis advisor, I would like to extend my gratitude to the rest of my thesis committee Dr. Spring Berman and Dr. Panagiotis Artemiadis. I am grateful to all of the faculty members who handled graduate courses for me at ASU.

I would also like to acknowledge the support of my fellow research mates Jesus Aldaco Lopez(Thanks for those daily yoghurts & discussions), Zhichao Li, Xianglong Lu, Nikhilesh Ravishankar, Kamalakannan Thammireddi Vajram, Dibyadeep Bose & Michael Thompson for standing through as my pillars of strength during all times. Thanks to David Phelps for sharing my burden. I take this opportunity to thank my friends Karan Puttannaiah, Kaustav Mondal, Ashfaque Bin Shafique, Rakesh Joshi, Parag Mitra, Sai Akshit Kumar Gampa, Madhurima Poore & Vignesh Narayanan for their tremendous support and stimulating discussions about my research which kept me motivated to complete my Masters thesis. Immense thanks to Shruti Anand for introducing me to LaTeX. Special thanks to Justin Echols for letting me use his printer as much as I wanted. I wish all of them great and exciting careers for their future. An heartfelt thanks to my past roommates Sudarsan, Narasimhan & Hari and my present roommates Pranav & Shishir for being there for me anytime I wanted. Finally, I would like to dedicate this work to my lovely parents Mr. V. Renganathan & Mrs. Brinda Renganathan for their love, encouragement, support and attention. Wherever you are my dear Ravi, this achievement is because of you.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

LIST OF SYMBOLS

[.]'	superscript ' denotes matrix transpose
$[.]^b$	superscipt b denotes body reference frame
$[.]^i$	superscipt i denotes inertial reference frame
$[.]^s$	superscipt s denotes seeker reference frame
$[.]^v$	superscipt v denotes vehicle reference frame
$[.]^w$	superscipt w denotes aerodynamic wind reference frame
a	Sonic velocity (speed-of-sound); varies with Temperature T
A$\times B$	Cross product between A and B
A_g	Gravitational acceleration; defined as $[0, 0, g]'^i$ or $[A_{g_x}, A_{g_y}, A_{g_z}]'^b$.
A_m	Inertial acceleration of CG_0; defined as $[A_{m_x}, A_{m_y}, A_{m_z}]'^b$.
A_t	Inertial acceleration of target; defined as $[A_{t_x}, A_{t_y}, A_{t_z}]'^i$.
A_{t_c}	Commanded target acceleration; defined as $[A_{t_{xc}}, A_{t_{yc}}, A_{t_{zc}}]^i$.
A_{mzL}	The limited pitch acceleration command generated by the autopilot.
$A_{mz_{max}}$	Max. value autopilot allows for commanded pitch acceleration.
A_{nt}	The desired target normal acceleration.
A_{yc}, A_{zc}	Commanded acceleration the autopilot receives from guidance.
C_D	Drag from pitch fin deflection δ_q
C_{DT}	Base drag due to Mach.
CG	Instantaneous center-of-gravity; moves relative to CG_0 as fuel burns; located by $[S_{c_x}, 0, 0]^b$.
CG_0	Initial center-of-gravity; Reference point for missile location & inertial dynamics; Co-origin for several non-inertial reference frames; located by $[S_{m_x}, S_{m_y}, S_{m_z}]^i$, $[0, 0, 0]^b$, $[0, 0, 0]^s$, $[0, 0, 0]^v$ & $[0, 0, 0]^w$.

C_{L_β}	Roll moment from sideslip.
C_{LP}	Roll damping moment from pitch rate.
C_M	Pitch moment aerodynamic coefficient.
C_{M_α}	Pitch Moment from Angle of Attack.
C_{N_α}	Lift due to Mach.
C_{N_β}	Yaw moment from Sideslip.
$C_{N_{\delta_r}}$	Yaw moment from yaw fin deflection.
C_{N_R}	Yaw Damping Moment - Yaw Rate.
C_{Y_β}	Side Force from Sideslip.
$C_{L_{\delta_q}}$	Roll Moment from Roll Fin Deflection.
$C_{M_{\delta_q}}$	Pitch Moment from Pitch Fin Deflection.
$C_{N_{\delta_q}}$	Lift from Pitch Fin Deflection.
$C_{N_{\delta_r}}$	Yaw Moment from Yaw Fin Deflection.
$C_{Y_{\delta_r}}$	Side Force from Yaw Fin Deflection.
C_X	Drag aerodynamic coefficient.
C_Y	Side force aerodynamic coefficient.
dm	Impulse change in mass (5.75 slug)
F_1,F_2,F_3,F_4	Deflection anlges of the missile's true steering fins, max = 20 deg
(F_x, F_y, F_z)	Aerodynamic (wind) force acting at CG_0; defined in the body frame. F_x = body frame drag, F_y = side force and F_z = lift.
F_g	Gravitational force acting at CG_0; defined as $[F_{g_x}, F_{g_y}, F_{g_z}]^b$.
F_{ie}	The difference between the actual and commanded fin positions.
F_{is}	The new commanded fin position, before the position filter.
F_m	Total external force acting at CG_0; defined as $[F_{m_x}, F_{m_y}, F_{m_z}]^b$.
F_{max}	Maximum angle allowed for fin actuators, (20 deg).
$\dot{F}_{max}$	Maximum Rate allowed for fin actuators, (600 $\frac{deg}{sec}$).

F_p	Propulsive force acting at CG_0; defined as $[-Thrust, 0, 0]^b$.
F_w	Aerodynamic (wind) force acting at CG_0; defined as $[D, C, L]^w$
g	Gravitational acceleration; decreases with inertial altitude h_i.
g_0	Gravitational acceleration at sea level, 45 degrees North latitude (32.174 $\frac{ft}{sec}$).
G_g	Gravitational moment acting about CG_0; defined as $[G_{g_x}, G_{g_y}, G_{g_z}]^b$.
G_m	Total external moment acting about CG_0; defined as $[L, M, N]^b$.
G_w	Aerodynamic (wind) moment acting about CG_0; defined as $[G_{w_x}, G_{w_y}, G_{w_z}]^b$.
h	Geopotential (constant-gravity) altitude above sea level; used to calculate air pressure, temperature T and air density ρ.
h_i	Inertial altitude above sea-level; equals $\lvert S_{m_z}\rvert$ when referring to the missile or $\lvert S_{t_z}\rvert$ when discussing the target; used to compute g
H_m	Total angular momentum about CG_0; defined as $[H_{m_x}, H_{m_y}, H_{m_z}]^b$.
ImpFrac	Fraction of Impulse accumulated at time t.
ImpNorm	Impulse described as a normalized linear function of time t.
Impulse	Time integral of Thrust; increases with time t.
I_{xx}	Moment of inertia about the body frame X^b-axis; decreases with time t.
I_{yy}	Moment of inertia about the body frame Y^b-axis; decreases with time t.
I_{zz}	Moment of inertia about the body frame Z^b-axis; decreases with time t.
I_{xxo}	Initial value of moment of inertia I_{xx}, ($0.34\ slug - ft^2$).
I_{yyo}	Initial value of moment of inertia I_{yy}, ($34.1\ slug - ft^2$).

I_{zzo}	Initial value of moment of inertia I_{zz}, (34.1 $slug - ft^2$).
L	Body frame roll moment, parallel to X^b-axis.
L_{ref}	Effective chord length of the missile airframe, (0.0625 ft).
m(t)	Instantaneous missile mass effectively located at CG; decreases with time t. Denoted 'Mass' in program.
m_f	Mass of expended fuel; increases with time t.
m_0	Initial value of missile mass m, (5.75 slug).
M	Body frame pitch moment, parallel to Y^b-axis.
Mach	Vehicle airspeed V_b normalized to local speed-of-sound SOS.
N	Body frame yaw moment, parallel to Z^b-axis.
P	Body frame roll angular velocity.
p_{g_1}	Nominal gain used in proportional guidance, is equal to 3.0.
p_{g_2}	Nominal gain used in proportional guidance, is equal to 3.0.
P_m	Total linear momentum of CG_0.
P_s	Projection onto linear subspace defined by S.
Q	Body frame pitch angular velocity.
Q_{dp}	Dynamic air pressure acting on a slow aircraft as it moves through the atmosphere at airspeed V_b.
Q_{sl}	Dynamic air pressure times the reference area times the refernce length.
R	Body frame yaw angular velocity.
R_0	Sea-level radius of earth, (20,903,264 ft).
Range	Magnitude of S_r or S_s.
SOS	Speed of Sound.
S_c	Displacement of CG from CG_0; increases with time; defined as $[S_{c_x}, 0, 0]^b$.

S_m	CG_0 Displacement from $[0,0,0]^i$; increases with time; defined $[S_{m_x}, S_{m_y}, S_{m_z}]^i$.
S_r	Target Displacement from $[0,0,0]^v$; increases with time; defined $[S_{r_x}, S_{r_y}, S_{r_z}]^v$.
S_{ref}	Effective cross-sectional area of the missile airframe, (0.307 ft^2).
S_s	Target Displacement from $[0,0,0]^s$; increases with time; defined $[S_{s_x}, S_{s_y}, S_{s_z}]^s$.
S_t	Target Displacement from $[0,0,0]^i$; increases with time; defined $[S_{t_x}, S_{t_y}, S_{t_z}]^i$.
t	Instantaneous time.
T_{change}	Half the time required to make a thrust transition, 0.025 sec.
Thrust	Magnitude of propulsive force ${F_p}^b$; modeled by Th_1 & Th_2.
Th_1	First stage missile thrust, (9250 lbs).
Th_2	Second stage missile thrust, (2140 lbs).
U	Body Frame inertial X^b-velocity.
V	Body Frame inertial Y^b-velocity.
V_b	Missile body velocity.
$(V_{m_x}, V_{m_y}, V_{m_z})$	Missile velocity in the inertial frame.
V_r	Missile target relative velocity, defined as $[V_{r_x}, V_{r_y}, V_{r_z}]^v$.
W	Body frame inertial Z^b-velocity.
X	Body frame drag.
Y	Body frame sideforce.
Z	Bpdy frame lift.
$\Delta Impulse$	Total change in Impulse after fuel is expended.
ΔI_{xx}	Total change in I_{xx} after fuel is expended.
ΔI_{yy}	Total change in I_{yy} after fuel is expended.

ΔI_{zz}	Total change in I_{zz} after fuel is expended.
Δm	Total change in mass m after fuel is expended.
ΔS_{c_x}	Total change in S_{c_x} after fuel is expended.
α	Angle of attack ; positive value locates V_m on $+Z^b$ side of body frame $(XY)^b$-plane.
β	Sideslip angle ; positive value locates V_m on $+Y^b$ side of body frame $(XZ)^b$-plane.
δ_{pc}	Effective roll fin deflection angle command, (aileron).
δ_{qc}	Effective pitch fin deflection angle command, (flapperon).
δ_{rc}	Effective yaw fin deflection angle command, (rudder).
δ_{sc}	Effective squeeze mode, ILAAT combining logic.
θ	Euler pitch angle; positive value locates body frame X^b-axis on -Z^v side of vehicle frame $(XY)^v$-plane.
θ_c	commanded seeker elevation angle.
θ_e	Measured seeker elevation error angle.
$\theta_{G_{max}}$	Maximum allowed seeker elevation angle, ($\pm$70 deg).
θ_G, θ_s	Seeker elevation gimbal angle; positive value locates body frame X^b-axis on +Z side of vehicle frame $(XY)^s$-plane.
$\dot{\theta}_{G_{max}}$	Maximum allowed rate for seeker servos, (75 $\frac{deg}{sec}$).
$\dot{\theta}_{G_{sat}}$	Limited seeker elevation rate.
ζ_f	Fin actuator damping ration, 0.30.
ζ_s	Seeker servo damping ration, 49.5.
ρ	Mass density of the atmosphere; decreases with geopotential altitude h.
σ_a	Vehicle frame azimuth LOS angle; positive values locates S_r on $+Y^v$ side of vehicle frame $(XZ)^v$-plane.

σ_e	Vehicle frame elevation LOS angle; positive values locates S_r on -Z^v side of vehicle frame $(XY)^v$-plane.
σ_{ep}	Seeker frame pitch LOS angle error.
σ_{ey}	Seeker frame yaw LOS angle error.
σ_p	Seeker frame pitch LOS angle; positive value locates S_s on -Z^s side of seeker frame $(XY)^s$-plane.
σ_y	Seeker frame yaw LOS angle; positive value locates S_s on $+Y^s$ side of seeker frame $(XZ)^s$-plane.
τ_p	Propulsion time-constant for exp. thrust transitions, 0.010 sec.
τ_t	Target response time constant.
ϕ	Euler roll angle; positive value locates body frame Y^b-axis on $+Z^v$ side of vehicle frame $(XY)^v$-plane.
ψ	Euler yaw angle; positive value locates body frame X^b-axis on $+Y^v$ side of vehicle frame $(XZ)^v$-plane.
ψ_c	Commanded seeker azimuth angle.
ψ_e	Measured seeker azimuth error angle.
$\psi_{G_{max}}$	Maximum allowed seeker azimuth angle, ($\pm$65 deg).
ψ_G, θ_s	Seeker azimuth gimbal angle; positive value locates body frame X^b-axis on -Y^s side of vehicle frame $(XZ)^s$-plane.
$\dot{\psi}_{G_{max}}$	Maximum allowed rate for seeker servos, (75 $\frac{deg}{sec}$).
$\dot{\psi}_{G_{sat}}$	Limited seeker azimuth rate.
ω^b	Angular velocity of body frame about its own axis relative to vehicle frame; defiend as $[P, Q, R]'^b$.
ω_f	Fin actuator undamped natural frequency, 195.0077 $\frac{rad}{sec}$.
ω_s	Seeker servo undamped natural frequency, 0.041 $\frac{rad}{sec}$.
${\Omega_m}^i$	Missile inertial angular velocity, $(\Omega_{mx}, \Omega_{my}, \Omega_{mz})^i$.

Chapter 1

INTRODUCTION & OVERVIEW OF WORK

1.1 Introduction and Motivation

A comprehensive procedure to ensure robust missile flight dynamics will include - defining mission requirements, wind tunnel testing, mathematical analysis, computer simulation and flight demonstration [55]. In this research, a MATLAB application has been developed to evaluate the performance of missile guidance and control system [1], [5], [15] and [17]. The application contains a complex dynamic simulation, displays missile-target intercept in 3D Animation with different viewpoints, provides a user friendly graphical user interface to input the initial flight condition and to view the post flight data plots. This research work includes miss distance analysis and kill zone (missile launch envelope) analysis with respect to different missile-target engagement parameters. Also, linear model of the missile is analyzed at different flight conditions and its dynamic flight modes are studied. A detailed comprehensive study of the Bank-to-Turn (BTT) missile gain scheduled nonlinear autopilot is presented.

The simulation consists of a six-degree-of-freedom Extended Medium Range Air-to-Air Technology (EMRAAT) missile (Range upto 200 miles) in pursuit of a three-degree-of-freedom evading target (e.g. enemy aircraft). Current Medium Range missiles have a range upto 3000 km. The simulation includes realistic missile and actuator dynamics, an autopilot, several missile guidance laws, seeker navigation model, various target models and several numerical integration methods. Missile dynamics include nonlinear features such as speed and altitude dependent aerodynamics, fuel consumption effects on mass and moments of inertia, nonlinear actuator and sensor dynamics with position and rate saturation.

This kill zone estimation problem arises mainly as a resource allocation problem. Imagine an enemy aircraft is spotted by military radar. Target has to be tracked down and destroyed before it damages any important resources of a country. Even if there are many missile launching centres, they have to be operated intelligently so that every missile launch turns out to be successful. So depending upon the need of the hour and position of the target, the results presented in this research shall quickly guide us through operating missiles intelligently. Using the program developed by [1] to simulate the missile to track and hit the target from any given starting position, this research tries to extend the work done by [1] to simulate the missile from different starting positions and estimate the kill zone for a given target. Thus, if the kill zone estimation for different flight conditions are known, missile launching centres can be operated with high success rate in tracking any enemy target aircrafts.

1.2 Literature Survey: Missile Guidance System - State of the Field

In an effort to shed light on the state of missile system modeling, control design, and post flight data analysis, the following topically organized literature survey is offered. An effort is made below to highlight what technical papers/works are most relevant to this thesis. All missile-target simulations are carried out using C program or MATLAB and the simulation data was analyzed using Matlab to come up with the results discussed in this thesis. In short, the following works are most relevant for the developments within this thesis:

- Traditionally, a computationally intensive simulation such as Missile-Target Engagement required working on a mainframe or workstation [18]. Nowadays even laptops can do very high end simulations at ease, given the hardware speed and improvement of software algorithms over the years.

- Initial attempt in missile-target simulation was carried out in a mainframe program by [4] and it offered very good speed but was lacking in clear visual aid to facilitate interpretation.

- Subsequent attempt was made by [2] where simulation was carried out using Visual Basic program on a personal computer but it suffered from speed and maintainability.

- Successful attempt of overcoming those difficulties was carried out by [1] where a C program was developed to simulate the missile-target engagement on a personal computer with very good visualization. It even successfully implemented graphical display of missile-target engagement using target maneuvers developed by [3] and [4]. Initial simulink version of above simulation was presented by [6], but it was still incomplete without good animation graphics to visualize the missile target engagement because it was not available at that time.

- The Aerodynamic coefficients used in missile dynamics are discussed in detail at [20]. Using polynomial fit to mathematically model the wind tunnel data about the missile aerodynamics should fasten the computation time of future missile guidance & control system simulation.

- The nonlinear autopilot used in this research work was originally designed by [4] with references from [25], [10] and [18]. The gain scheduling used in this research can be read in detail from [40] and [28]. The need for a nonlinear autopilot is clearly explained in [11].

- Robustness analysis is performed to evaluate the controller (autopilot) performance [29], [26] and the idea of studying the closed maps [21] at different loop breaking points is addressed in books [80], [78] and works done by [43], [44] and many others.

- The complex nonlinear differential equations governing the 6DOF missile dynamics need to be solved and numerical integration methods explained in [68] and [13] are used in this research. Engagement geometry analysis presented in this research helps us in selecting an optimal step size for the numerical integration used and the problem of actuators hitting their saturation levels frequently due to poor step size selection is addressed in [41], [38] [45] and [46].

- Miss distance analysis results from renowned GNC texts [51] and [52] motivates the miss distance analysis done in Chapter 8 of this research work. The high fidelity environment developed by [5] and [2] is used in this research to validate the miss distance profiles presented in the above mentioned GNC text books.

- The main challenge was coming up with an efficient search algorithm in 3D space to vary the missile starting position and see whether it hits the target starting from those starting positions. This is where ideas developed by [22], [72] were helpful in narrowing down the algorithm selection to Binary Search to come up with different missile starting positions intelligently.

- Entire search space is divided into rays starting from origin where missile is assumed to be located. Along each ray, binary search algorithm is used to find first hit position, first miss position, last hit position and last miss position. Then all the hit positions are joined together to form a boundary, which can be interpreted as a Kill Zone [14], [27], [37], [30], [24], [23] a closed space from where if the missile starts to track the target, it is assured to hit it with greater probability.

- Visualization of missile target engagement is the motivation factor for developing a MATLAB 3D Animation. Previous works in trying to simulate and animate aerospace vehicles were done by [9] and the steps to build the animation are available online [73].

An attempt is made below to provide relevant insightful technical details.

- **Missile Modeling.** Siouris's book [51] and Zarchan's book [52] address modeling for bank-to-turn missiles. Linearization of missile dynamics is addressed within [62]. Within this thesis, the focus is on guidance, navigation and control of bank-to-turn missiles.

- **Nonlinear Autopilot.** The need for a nonlinear autopilot for missile flight control system is addressed within the paper [10] and [11]. Within this text, it is shown that while the missile is inherently non-minimum phase in nature and a robust autopilot is needed to stabilize that and make the missile to operate across different flight conditions.

- **Classical Controls.** Classical control design fundamentals are addressed within the text [64]. Internal model principle ideas - critical for command following and disturbance attenuation - are presented within [64]. General PID (proportional plus integral plus derivative) control theory, design and tuning are addressed within the text [80]. Fundamental performance limitations are discussed within [78], [64].

- **Multivariable Control.** General multivariable feedback control system analysis and design is addressed within the text [65]. Linear quadric regulator (LQR) and LQ servo concepts are discussed within [79], [65].

- **Relevant Nonlinear Control.** In order to acheive adequate performance over the entire envelope of operating conditions, the autopilot of a modern air-to-air tactical missile must be *nonlinear* [10]. The nonlinearity arises either through the gain-scheduling of linear point designs or through the direct application of nonlinear control technique to the problem.

- **Multiple Loop Control.** It is interesting to ask the following question while studying about designing missile flight control system.

  > When do we need multiple control loops and
  > why a single feedback loop won't suffice?

  The time-scale separation experienced by missiles between "slow" *translational dynamics* and "fast" *rotational dynamics* calls for a two loop strategy implementation, as single unified (single loop) framework would become ineffective here [7]. Single loop strategy fails because of following reasons,

  - Control surface deflections directly respond to the translational error correction demands, which may lead to the instability of the rotational dynamics.
  - This is especially true for control surfaces located either at the front or at the tail of the missile (we have a tail controlled missile in our consideration here in this research), because deflections of these control surfaces can create only minor forces, whereas they create large moments due to long moment arm from the center-of-mass.
  - Consequently these control surfaces are ineffective in directly correcting translational errors, whereas they can be very effective in turning the flight

vehicle.

Therefore, for a successful flight control system, the design must exploit the time-scale separation that exists between translational and rotational motions of the center-of-mass.

- **Autopilot Innermost-Loop Control.** A nonlinear controller with its gains scheduled as function of different flight conditions is implemented here. Innermost loop is mainly for stabilizing the missile while helping the missile to follow the commanded angular rates by issuing proper fin commands to the actuators. Essentially innermost autopilot loop is meant for controlling angular rates here, referred sometimes as *Rate Loop*.

- **Autopilot Intermediate-Loop Control.** Intermediate loop is mainly for controlling the missile bank angle, angle of attack and sideslip angle while helping the missile to follow the commanded bank angle which is generated by the BTT Logic module.

- **Outer-Loop Guidance Control.** Within this thesis, various outer-loop guidance control laws are examined. Usually referred as the *Guidance Loop*, this will help the missile to steer towards the target (read it as *position control loop*). Essentially this is also proportional controlled loop with gains determined by the guidance laws.

1.3 Goals and Contributions

Miss distance of the target with respect to the missile was analyzed upon varying various parameters of missile and the results are presented in this work and they agree [1; 17; 16; 15; 33]. This research work will address and provide concrete answers to see if the Kill Zone Estimation done using binary search algorithm correlates well with the miss distance results presented in above mentioned papers. Missile parameters such as initial altitude, initial mach and maximum missile acceleration are varied in different sizes, one at a time and the variation of estimated kill zone is analyzed. Before pursuing the study, it is instructive to acknowledge some simple ideas and intuitions below which are answered in this research.

1.4 Contributions of Work: Questions to be Addressed

Within this thesis, the following fundamental questions are addressed. When taken collectively, the answers offered below, and details within the thesis, represent a useful contribution to researchers in the field.

Why should a hierarchical inner-outer loop control architecture be used? Hierarchical inner-outer loop controllers are found across many industrial/-commercial/military application areas (e.g. aircraft, spacecraft, robots, manufacturing processes, etc.) where it is natural for slower (outer-loop generated) high-level commands to be followed by a faster inner control loop that must deliver robust performance (e.g. low frequency reference command following, low-frequency disturbance attenuation and high-frequency sensor noise attenuation) in the presence of significant signal and system uncertainty. A well designed inner-loop can greatly simplify outer-loop design. An excellent example of inner-outer loop architectures are used in this missile-target application arena. Here, an autopilot (inner-loop)[1] follows commands generated from the guidance system (outer-loop). More substantively, inner-outer loop control structures are used to tradeoff properties at distinct loop breaking points (e.g. outputs/errors versus inputs/controls) [43], [44].

[1]Within an autopilot there is typically very critical lower-level actuator control inner-loops.

Inner-Loop Control What are typical inner-loop objectives? Typical inner-loop objectives can be speed control; i.e. requiring the design of a angular speed control system. Within this thesis, inner-loop control for our BTT missile specifically refers to nonlinear gain scheduled autopilot.

What is a suitable inner-loop control structure? When is a classical (decentralized) PI structure sufficient? When is a multivariable (centralized) structure essential? For many applications such as differential drive robotic vehicle, a simple PI/PID (decentralized) control law with high frequency roll-off and a command pre-filter suffices (see Chapters 3 and 6). Such an approach should work when the plant is not too coupled and the design specifications are not too aggressive relative to frequency dependent modeling uncertainty. A multivariable (centralized) structure becomes essential when the plant is highly coupled such as the missile control system considered within this thesis and the design specifications are very aggressive (e.g. high bandwidth relative to coupling/uncertainty)[65].

What are the limitations on the bandwidth of the missile flight control system? How does the presence of RHP zeros (nonminimum phase) and RHP poles affect the closed loop bandwidth? The pitch up instability phenomenon present in all tail controlled vehicles give rise to both RHP pole and RHP zeros in system. While the unstable pole demands a minimum bandwidth to stabilize the system, the nonminimum phase zero poses an upward limit on the maximum bandwidth of the system. Thus going by the thumb rule, the bandwidth of a system with RHP pole, 'p' and RHP zero, 'z' is given by following equation.

$$2\,|p| \leq Bandwidth \leq \frac{|z|}{2} \tag{1.1}$$

What is a suitable outer-loop control structure? When is a more complex structure needed? Suppose that an inner-loop speed control system has been designed. Suppose that it looks like $\frac{a}{s+a}$. It then follows that if position is concerned, then we have a system that looks like $\left[\frac{a}{s(s+a)}\right]$; i.e. there is an additional integrator present. Given this, classical control (root locus) concepts [64] can be used to motivate an outer-loop control structure $K_o = g(s+z)$. In an effort to attenuate the effect of high frequency sensor noise, one might introduce additional roll-off; e.g. $K_o = g(s+z)\left[\frac{b}{s+b}\right]^n$ where $n = 2$ or greater. (See work within Chapter 6)

1.5 Overview of Thesis

In this research, a MATLAB application is developed and used to evaluate the performance of missile guidance and control systems. The program simulates and uses MATLAB 3D Animation using VRML toolbox to display the missile-target air-to-air engagement. The endgame portion of the engagement is of particular interest whereby the target maneuvers causing the missile controls to saturate and possibly induce instability [76]. The missile controls may saturate in the thin air found at higher altitudes or when the actuator saturation limit is small. It would be desirablle to visualize this phenomenon and quantify it and use the techniques in [41], [39], [34], [45], [42] and [46] to prevent it. The simulation includes realistic missile and actuator dynamics, various guidance systems (proportional, optimal and differential game), a seeker navigation system model and various target models. The target represents a simplified version of a high performance enemy aircraft. The three-degree-of-freedom target is modeled with its acceleration limited to ± 9 Gs, values tolerable to human pilot.

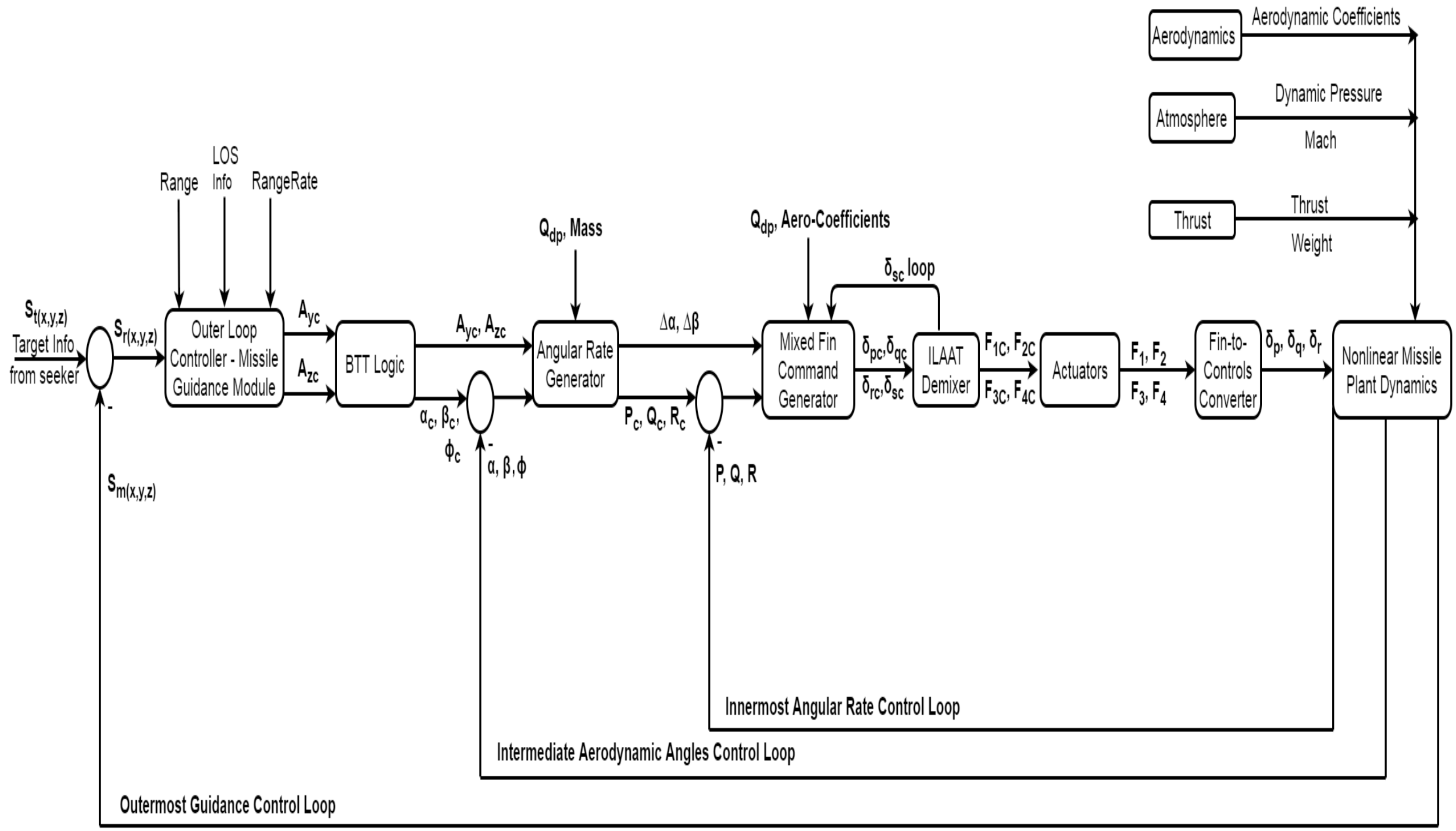

Figure 1.1: Information Flow for Missile-Target Engagement.

Figure 1.1 illustrates how the above systems interact with one another, Each subsystem is briefly discussed as follows:

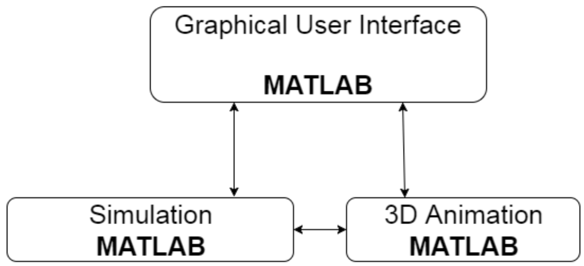

Figure 1.2: Organization of MATLAB Program: 3 Modules.

Missile Dynamics. A set of nonlinear ordinary differential equations capturing aerodynamic, atmospheric and variable mass effects are used to model an EMRAAT BTT missile. The model relates four controls (fin deflections - F_1, F_2, F_3, F_4) to the missile's coordinate velocities (V_{mx}, V_{my}, V_{mz}) and its roll, pitch and yaw angles (ϕ, θ, ψ).

Actuator Dynamics. Each of the four missile fins is controlled by a servo-based actuation system - modelled by a nonlinear underdamped system with position and rate saturations.

Autopilot. Because of the inherent instabilities associated with missiles, stability augmentation systems are essential. The autopilot provides the added stability and ensures that acceleration commands from the guidance system are properly followed. More precisely, the autopilot uses feedback to process the guidance commands and deliver appropriately coordinated fin commands to the actuators.

Guidance System. The purpose of the guidance system is to issue appropriate acceleration commands to the autopilot on the basis of target information obtained from the seeker/navigation (target sensing) system.

Seeker/Navigation System. The seeker/navigation (target sensing) system generates target line-of-sight (LOS) rate information which is used by the guidance system.

Target Dynamics. Different models are used to reflect the maneuverability and intelligence of the target. Each model has 3 degree of freedom.

The prime objective is to minimize the distance [1] between the missile and the target within a limited time.

1.6 Organization of Thesis

The remainder of the thesis is organized as follows.

- Chapter 2 (page 16) presents an overview for a general 6DOF missile equations of motions and 2^{nd} order dynamics governing 4 missile fin actuators.
- Chapter 3 (page 55) describes the linearization routine followed in linearizing the nonlinear missile plant. The ideas presented here include analysis of all dynamic flight modes of missile with respect to different flight parameters. This chapter also provides a foundation for the work in Chapter 6 where both autopilot and plant analysis is done together.
- Chapter 4 (page 114) presents seeker dynamics and the 6DOF missile guidance laws that helps the missile to intercept a maneuvering target. Three different guidance laws are described.

[1] Miss distance is defined as the final range between missile and target, after the missile has tried to intercept the target.

- Chapter 5 (page 129) describes 3DOF target modeling and its three different maneuvering modes are discussed in detail.

- Chapter 6 (page 135) describes modeling and control issues for a Bank-To-Turn (BTT) missile using a nonlinear autopilot. Linearization of missile autopilot is discussed and this chapter serves as the basis for main control design. This chapter contains the main work that was conducted in this research.

- Chapter 7 (page 184) describes the usage of different numerical integration techniques. The chapter serves as the basis for selection of optimal step size for numerical integration through engagement geometry analysis.

- Chapter 8 (page 196) describes the effect of different missile and target parameters on the final miss distance of a missile as described in [51] and [52]. The chapter serves as the basis for Chapter 9, which is just an extension of chapter 8 ideas in a different perspective.

- Chapter 9 (page 219) describes the effect of different missile and target parameters on the estimated Kill zone of a missile using binary search algorithm.

- Chapter 10 (page 235) describes modeling and animating the entire missile-target engagement using VRML toolbox of MATLAB. 3D animation results using VRML toolbox and initial graphical user interface development are discussed.

- Chapter 11 (page 248) summarizes the thesis and presents directions for future missile research. While much has been accomplished in this thesis, lots remains to be done.

- Appendix A (page 258) contains C program implementation of Binary Search algorithm to estimate kill zone. Also MATLAB files to plot the kill zone is included in this section.

- Appendix B (page 264) contains MATLAB ‘m’ files used in this thesis for plotting linearized plant and autopilot analysis plots.

1.7 Summary and Conclusions

In this chapter, we provided an overview of the work presented in this thesis and the major contributions. A central contribution of the thesis is an improved autopilot design with animation to visualize the missile target engagement and detailed Kill Zone & Miss Distance analysis to explore the complexities involved in missile-target engagement.

Chapter 2

MISSILE & ACTUATOR DYNAMICS

2.1 Introduction and Overview

In this chapter the six degree-of-freedom nonlinear missile dynamics are described. Also described are the nonlinear fin actuator dynamics. Section 2.2 describes the reference frames used to develop the missile dynamics. Section 2.3 describes the effect of fuel loss. Section 2.4 describes the aerodynamic relationships, i.e. the effects due to the static and dynamic fluid properties of the atmosphere - accounted for via the dynamic pressure, Mach number and stability derivatives. Section 2.5 contains the equations of motion for the missile and Section 2.6 describes the nonlinear actuator dynamics. Finally Section 2.7 summarizes the chapter and concludes the items explained in this chapter.

2.2 Inertial, Vehicle and Body Frames

In this section three reference frames are described. A perspective, or reference frames, can be selected so that the dynamics within them can be described by relatively simple equations. The overall system can then be described by simply transforming the equations from one reference frame to another. Reference frames used in missile dynamics analysis include: (1) Inertial Frame, (2) Vehicle Relative Frame and (3) the Body Frame. Introducing these reference frames significantly simplifies the equations of motion for the missile.

2.2.1 *Inertial Frame*

Inertial Frame is a stationary coordinate system used to describe the motion of all objects within it. Throughout the thesis and in the program, the origin of this frame $(0, 0, 0)^i$ [1] is located at sea level directly below the missile initial launch point as shown in the Figure 2.1. This assumption is valid and typical for short range missions. It is not valid, for example, in long range Inter-Continental Ballistic Missile (ICBM) applications [62]. Given the above convention, the missile's launch (time zero) center of gravity, denoted by CG_0 is located by the inertial point:

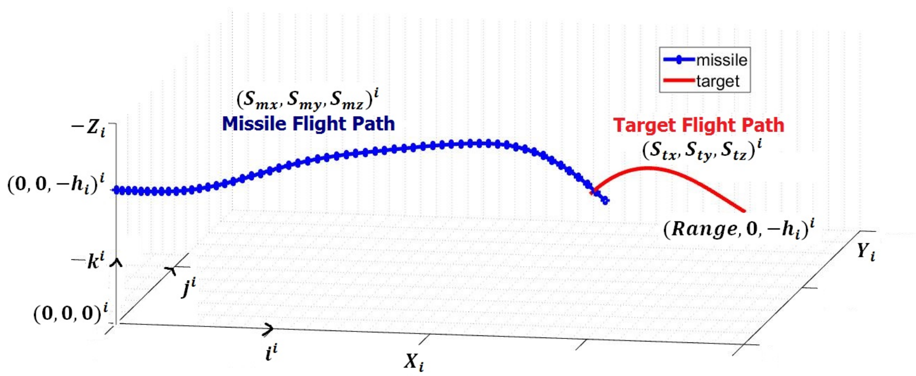

Figure 2.1: Local Inertial Frame with missile and target flight paths

$$S_m{}^i \stackrel{\text{def}}{=} (S_{mx}, S_{my}, S_{mz})^i \tag{2.1}$$

Its inertial velocity is denoted by

$$V_m{}^i \stackrel{\text{def}}{=} (V_{mx}, V_{my}, V_{mz})^i \tag{2.2}$$

The missile's inertial angular velocity is denoted by

[1] The superscript i will be used to denote a coordinate with respect to the inertial frame

$$\Omega_m{}^i \stackrel{\text{def}}{=} (\Omega_{mx}, \Omega_{my}, \Omega_{mz})^i \tag{2.3}$$

Similarly, the target is located by the inertial point:

$$S_t{}^i \stackrel{\text{def}}{=} (S_{tx}, S_{ty}, S_{tz})^i \tag{2.4}$$

The target's inertial velocity is denoted by

$$V_t^i \stackrel{\text{def}}{=} (V_{tx}, V_{ty}, V_{tz})^i \tag{2.5}$$

Also shown in Figure 2.1 are typical missile and target flight paths.

2.2.2 Vehicle Frame

Often it is convenient to use the missile's (time zero) center-of-gravity, CG_o as the origin and this motivates the so-called vehicle frame. This is a nonstationary coordinate system used to measure the relative distance between the missile and target, its origin is at the missile's (time zero) center-of-gravity, CG_o. This is a right-handed coordinate system centered at CG_o with axes denoted (X^v, Y^v, Z^v) which remain parallel to their inertial counterparts (X^i, Y^i, Z^i). The vehicle frame can be visualized as shown in Figure 2.2.

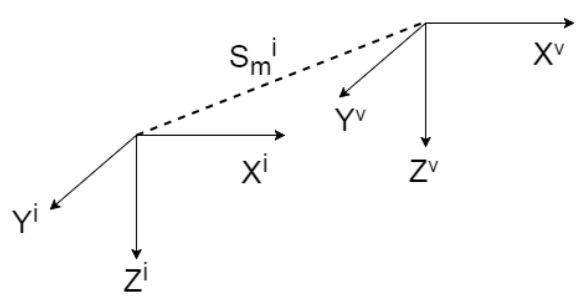

Figure 2.2: Visualization of Inertial and Vehicle Frames

2.2.3 Body Frame

A coordinate system is needed to conveniently define the missiles physical geometry as well as sum all forces and moments acting on the missile. This motivates the body frame. This is a right-handed coordinate system centered at missile's (time zero) center-of-gravity, CG_o. Its axes are denoted (X^b, Y^b, Z^b), where X^b emerges from the missile's nose is a forward axis and Y^b is a starboard axis. The body frame can be visualized as shown in Figure 2.3.

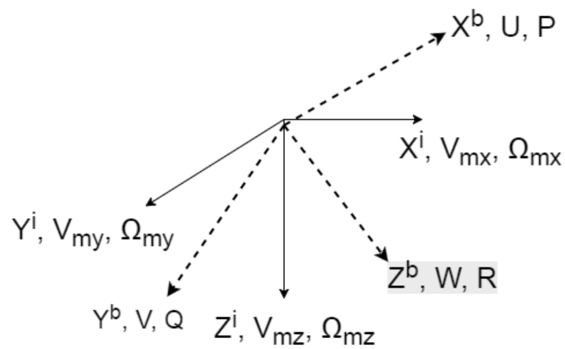

Figure 2.3: Visualization of Body Axes and Velocities

Body Axis Velocities.

We define the missile's body axis velocities (U, V, W) to be the components of the missile's inertial velocity ${V_m}^i$ along the body axes (X^b, Y^b, Z^b). The body axis velocities can be visualized as shown in Figure 2.3.

Body Axis Angular Velocities.

We define the missile's body axis angular velocities (P, Q, R) to be the components of the missile's inertial angular velocity ${\Omega_m}^i$ along the body axes (X^b, Y^b, Z^b). These body axis angular velocities can be visualized as shown in Figure 2.3.

Euler Angles and Missile Attitude.

To precisely specify the orientation(attitude) of the missile in inertial space, it is convenient (and convention) to introduce the so-called Euler angles (ϕ, θ, ψ). To precisely define these angles, consider the vehicle and body axes systems shown in Figure 2.4.

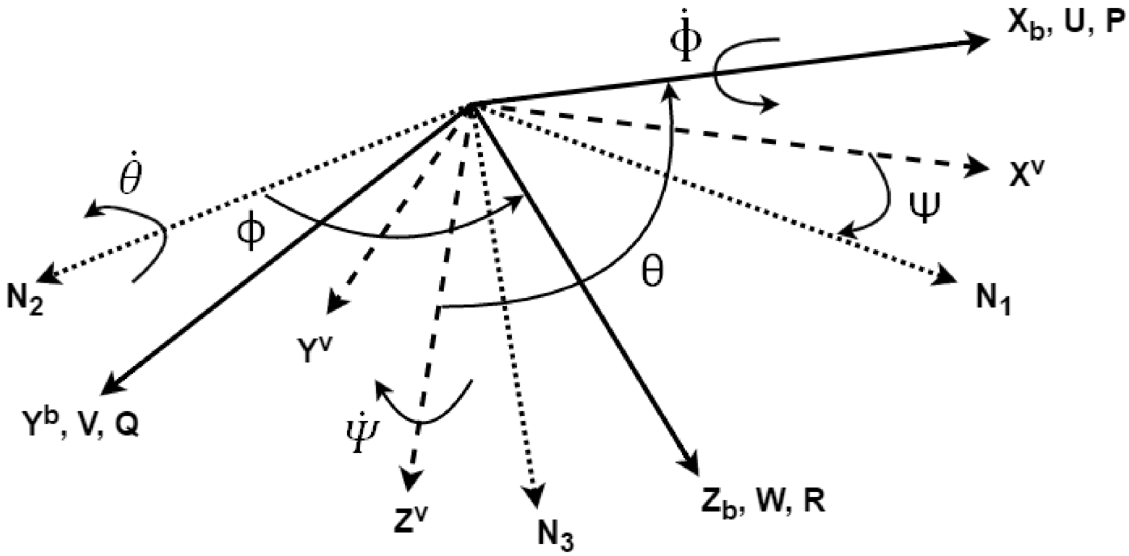

Figure 2.4: Visualization of Euler Angles

To define the Euler angles we proceed as follows. Let N_1 denote the projection of X^b onto the X^vY^v (horizontal) plane [2]:

$$N_1 \stackrel{\text{def}}{=} P_{X^vY^v} X^b \tag{2.6}$$

where $P_{X^vY^v}$ denotes a projection operator. The missile *pitch angle* or *pitch attitude* is then defined as the angle from N_1 to X^b measured in the vertical N_1X^b plane:

$$\theta \stackrel{\text{def}}{=} \angle N_1 X^b \tag{2.7}$$

[2]Recall that the X^vY^v plane is parallel to the X^iY^i plane. See Figure 2.2.

The missile *yaw angle* ψ is defined as the angle from X^v to N_1 measured in the horizontal X^vY^v plane:

$$\psi \stackrel{\text{def}}{=} \angle X^v N_1 \tag{2.8}$$

At this point, it would be useful to organize some geometric observations in the form of lemmas.

Lemma 2.2.1 (Orientation of N_1)

1. N_1 lies in the Z^vX^b plane
2. $N_1 \times X^b$ is parallel to $Z^v \times X^b$

Proof: To prove this, it suffices to show that N_1 lies in the Z^vX^b plane. This, however follows from the following algebraic manipulations.

$$\begin{aligned} N_1 &\stackrel{\text{def}}{=} P_{X^vY^v}X^b \\ &= \text{-}[X^b \text{ - } P_{X^vY^v}X^b] + P_{X^vY^v}X^b + [X^b \text{ - } P_{X^vY^v}X^b] \\ &= \text{-}P_{Z^v}X^b + X^b \end{aligned}$$

where $P_{Z^v}(.)$ denotes the projection of (.) onto the Z^v plane.

Now let N_2 denote the axis defined by the angular velocity $\dot{\theta}$ via the right-hand rule. By definition of θ and lemma 2.2.1, we see that N_2 is parallel to Z^v x X^b. For convenience we will write

$$N_2 \stackrel{\text{def}}{=} \text{Z}^v \text{ x } X^b$$

Given this, we make the following observation:

Lemma 2.2.2 (Orientation of N_2)

1. N_2 lies in the Y^bZ^b plane

Proof: If one lets $Z^v = \alpha_1 X^b + \alpha_2 Y^b + \alpha_3 Z^b$, then the result follows from the following equality:

$$\begin{aligned} N_2 = Z^v \text{ x } X^b &= [\alpha_1 X^b + \alpha_2 Y^b + \alpha_3 Z^b] \text{ x } X^b \\ &= \alpha_2 Y^b \text{ x } X^b + \alpha_3 Z^b \text{ x } X^b \\ &= \beta_1 Z^b + \beta_2 Y^b \end{aligned}$$

for some scalars β_1, β_2.

It should be noted that the Y^bZ^b plane, in general, need not be vertical. Given lemma 2.2.2, one should justifiably define the missile *roll angle* or *roll attitude* to be the angle from N_2 to Y^b measured in the Y^bZ^b plane:

$$\phi \stackrel{\text{def}}{=} \angle N_2 Y^b \tag{2.9}$$

To relate the Euler rates ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$) to the body rates (P, Q, R), it is convenient to define N_3 to be the projection of Z^v onto the Y^bZ^b plane:

$$N_3 \stackrel{\text{def}}{=} P_{Y^bZ^b} Z^v \tag{2.10}$$

Now we make the following important geometric observations.

Lemma 2.2.3 (Orientation of N_1, N_2 and N_3)

1. $N_1 \perp$N$_2$, $N_2 \perp N_3$, N_1 not $\perp N_3$

2. $\angle\ N_2 Y^b = \angle\ N_3 Z^b = \phi$

3. N_3 lies in the $Z^v X^b$ plane

4. $\angle\ Z^v N_3 = \angle\ N_1 X^b = \theta$

Proof:

(1) Since

$$
\begin{aligned}
N_2 = Z^v \text{ x } X^b &= Z^v \text{ x } [P_{X^vY^v}X^b + X^b \text{ - } P_{X^vY^v}X^b] \\
&= Z^v \text{ x } [N_1 + P_{Z^v}X^b] \\
&= Z^v \text{ x } N_1
\end{aligned}
$$

it follows that N_1 and N_2 are orthogonal. Similarly, since

$$
\begin{aligned}
N_2 = Z^v \text{ x } X^b &= [P_{Y^bZ^b}Z^v + Z^v \text{ - } P_{Y^bZ^b}Z^v] \text{ x } X^b \\
&= [N_3 + P_{X^b}Z^v] \text{ x } X^b \\
&= N_3 \text{ x } X^b
\end{aligned}
$$

It follows that N_2 and N_3 are orthogonal. Also, since $N_1 = P_{X^vY^v}X^b$ and $N_3 = P_{Y^bZ^b}Z^v$, it follows that N_1 and N_3 need not to be orthogonal.

(2) From lemma 2.2.2 N_2 lies in the Y^bZ^b plane. N_3 lies in this plane by the

definition. Since N_2 and N_3 are orthogonal and both lie in the Y^bZ^b plane, it follows from Figure 2.5 that

$$\angle\, N_2Y^b = \angle\, N_3Z^b = \phi$$

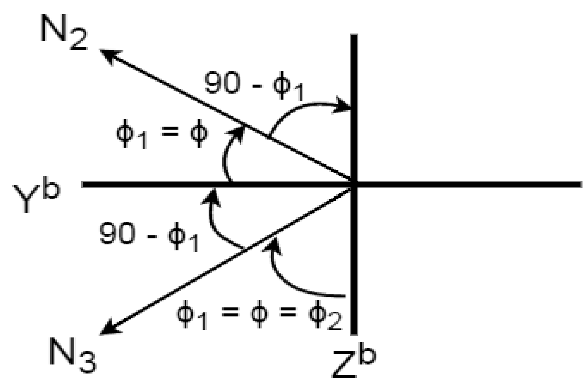

Figure 2.5: Visualization of N_2 and N_3 in Y^bZ^b plane.

(3) Since $N_3 = P_{Y^bZ^b}Z^v = Z^v$ - $[Z^v$ - $P_{Y^bZ^b}Z^v] = Z^v$ - $P_{X^b}Z^v$

it follows that N_3 lies in the Z^vX^b plane.

(4) Now recall from lemma 2.2.1 that N_1 lies in the Z^v x X^b plane. Since N_3 does also, it follows from Figure 2.6 that

$$\angle\, Z^vN_3 = \angle\, N_1X^b = \theta$$

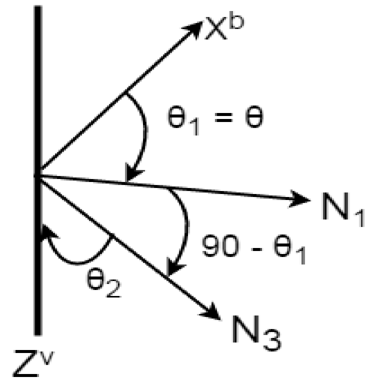

Figure 2.6: Visualization of N_1 and N_3 in X^b x Z^v plane.

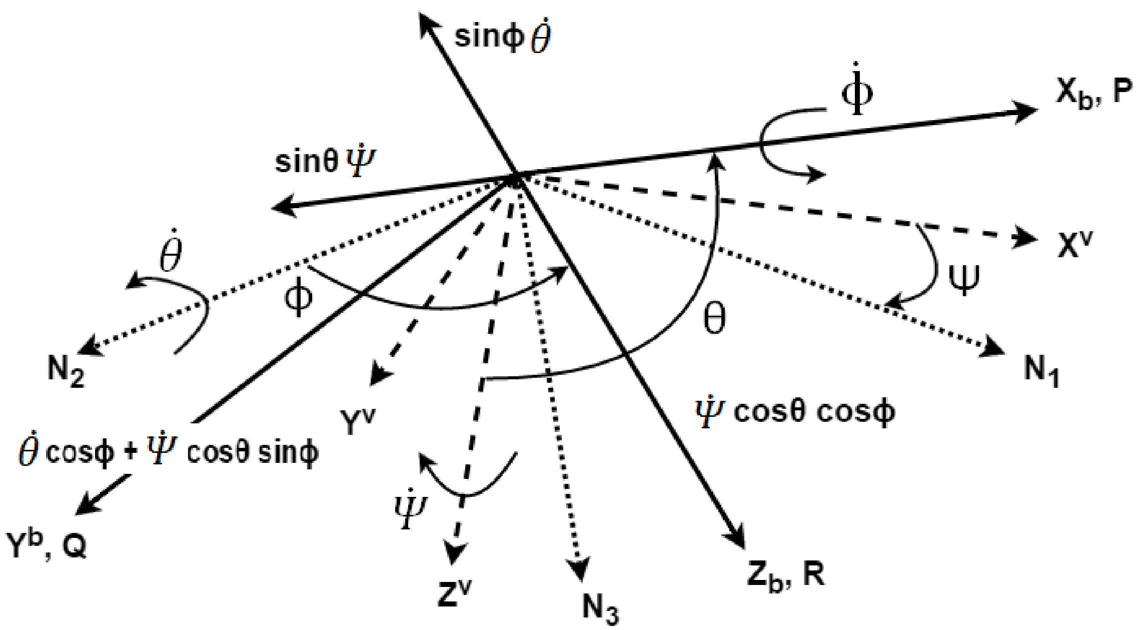

Figure 2.7: Relationship between Euler Angles and Body Rates.

Figure 2.7 which shows how to relate the Euler rates ($\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$) to the body rates (P, Q, R).

From Figure 2.7, one obtains the following coordinate transformation:

$$
\begin{bmatrix} P \\ Q \\ R \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.11}
$$

from which one obtains:

$$
\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} P \\ Q \\ R \end{bmatrix} \tag{2.12}
$$

2.3 Thrust Profile and Variable-Mass Dynamics

Loss of mass through fuel comsumption influence missile dynamics and must be accounted for in a realistic manner. Mass loss during flight causes the CG to move forward with respect to the CG_0, because the seeker is located in the forward part of the missile. The rate of mass loss will also vary with time due to the missile being modeled with variable thrust. A two-stage thrust profile is used in this simulation. A large initial thrust is needed to free the missile from the launching aircraft. A second level of thrust allows the missile to approach the target while remaining within its designed capabilities. The two thrust levels are as follows:

$$
Th_1 = 9250\, lbs \tag{2.13}
$$

$$
Th_2 = 2140\, lbs \tag{2.14}
$$

as shown in the Figure 2.8.

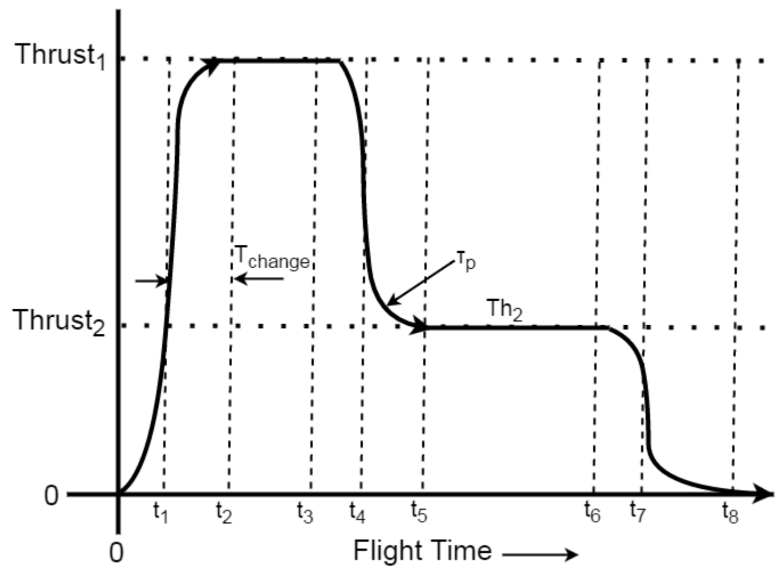

Figure 2.8: Missile Two-Stage Thrust Profile

The thrust profile is a piecewise linear function of time. All variable-mass effects are modeled as piece-wise linear time functions. Small time-constant exponential fucntions are used to smooth the piecewise connected thrust profile at the transition points. A more precise description of the thrust profile is given in Table 2.1. T_{change} = 0.025 sec, is equal to half the time required to make a thrust transition. It is chosen to be much smaller than the Th_1 time interval, which is equal to 0.6 sec. The time constant τ_p (equal to 0.01 sec) is chosen to be at least 5 times smaller than $2T_{change}$.

$$
\begin{aligned}
t_0 &= & &= 0.000 \text{ sec} \\
t_1 &= & T_{change} &= 0.025 \text{ sec} \\
t_2 &= & t_1 + T_{change} &= 0.050 \text{ sec} \\
t_3 &= & &= 0.600 \text{ sec} \\
t_4 &= & t_3 + T_{change} &= 0.625 \text{ sec} \\
t_5 &= & t_4 + T_{change} &= 0.650 \text{ sec} \\
t_6 &= & &= 6.090 \text{ sec} \\
t_7 &= & t_6 + T_{change} &= 6.115 \text{ sec} \\
t_8 &= & t_7 + T_{change} &= 6.140 \text{ sec}
\end{aligned}
$$

Modified Thrust. The caveit of using the above thrust profile is it may lead to missile being travelling at say Mach 7 at an altitude of 40 kft which is bad. The missile can't travel at such higher mach values given its fuel, aerodynamics and design. So to avoid the above confusion, it is suggested to have the following modified thrust profile, which is obtained through multiplying scaled air density component (ρ) in old thrust profile. Air gets thinner as we go up and if that is modelled along with this thrust profile as below, then missile will always stay within its specified mach value. This change is discussed in [6] but not present in simulation environments done by [2] & [5]. ρ_{sl} is a constant (air density at sea level) and defined as 0.0024 $\frac{slug}{ft^3}$.

$$Thrust_{new} = Th_i \frac{\rho}{\rho_{sl}} \tag{2.15}$$

where i = 1,2 respectively. By assumption, the time-derivative of the missile's mass and moments of inertia are directly proportional to the missile impulse time-derivative, where impulse is given as the time integral of thrust:

Time Interval	Thrust Value when τ_p = 0.010 sec
First Transition: to Th_1	$\frac{Th_1}{1+e^{-(t-t_1)/\tau_p}}$
Th_1	9250 lbs
Transition: Th_1 **to** Th_2	$Th_1 + \frac{(Th_2-Th_1)}{1+e^{-(t-t_4)/\tau_p}}$
Th_2	2140 lbs
Transition: Th_2 **to end**	Th_2 - $\frac{Th_2}{1+e^{-(t-t_7)/\tau_p}}$

Table 2.1: Thrust Profile Equations

$$Impulse \stackrel{\text{def}}{=} \int_{t_o}^{t} Thrustdt + Impulse(0) \approx Th_1(t_3 - t_2) + Th_2(t_6 - t_5) \tag{2.16}$$

where,

$t_0 \stackrel{\text{def}}{=} 0$ and Impulse(0) $\stackrel{\text{def}}{=} 0$

First we define three impulse-fraction constants:

$$ImpFrac_1 \stackrel{\text{def}}{=} \frac{Th_1}{Impulse} \tag{2.17}$$

$$ImpFrac_2 \stackrel{\text{def}}{=} \frac{Th_2}{Impulse} \tag{2.18}$$

$$ImpFrac_3 \stackrel{\text{def}}{=} (ImpFrac_1 - ImpFrac_2)(t_7 - t_0) \tag{2.19}$$

These constants allow the impulse to be re-described as a normalized linear function of time:

$$ImpNorm = \begin{cases} 0 & : t < t_2 \\ ImpFrac_1\ t & : t_2 < t < t_3 \\ ImpFrac_2\ t + ImpFrac_3 & : t_5 < t < t_6 \\ 1 & : t_6 < t \end{cases} \tag{2.20}$$

Missile mass can now be represented by a time function:

$$m(t) = m_0 - ImpNorm(t)\, dm \tag{2.21}$$

where m_0 = 5.75 slug is the missile's time-zero mass and dm = 2.2689 slug is defined so where, [ImpNorm(t_8 - t_0) dm] is the total mass lost from t_0 to t_8. The missile's principal moments of inertia are similarly defined:

$$I_{xx}(t) = I_{xxo} - ImpNorm(t)\, dI_{xx} \tag{2.22}$$

$$I_{yy}(t) = I_{yyo} - ImpNorm(t)\, dI_{yy} \tag{2.23}$$

$$I_{zz}(t) = I_{zzo} - ImpNorm(t)\, dI_{zz} \tag{2.24}$$

where time-zero moments and impulse change in moments of inertia are defined in Table 2.2.

The CG is located along the body frame positive X-axis using:

$$X_{CG}(t) = ImpNorm(t)\, dCG \tag{2.25}$$

Figure 2.9 shows the orientation of the CG to the missile body frame.

Time-Zero Missile Inertial Moment (slug-ft^2)	Impulse Change in Moment of Inertia (slug-ft^2)
$I_{xxo} = 0.34$	$dI_{xx} = 0.11$
$I_{yyo} = 34.1$	$dI_{yy} = 6.94$
$I_{zzo} = 34.1$	$dI_{zz} = 6.90$

Table 2.2: Missile's Time-Zero Mass and Moment of Inertia

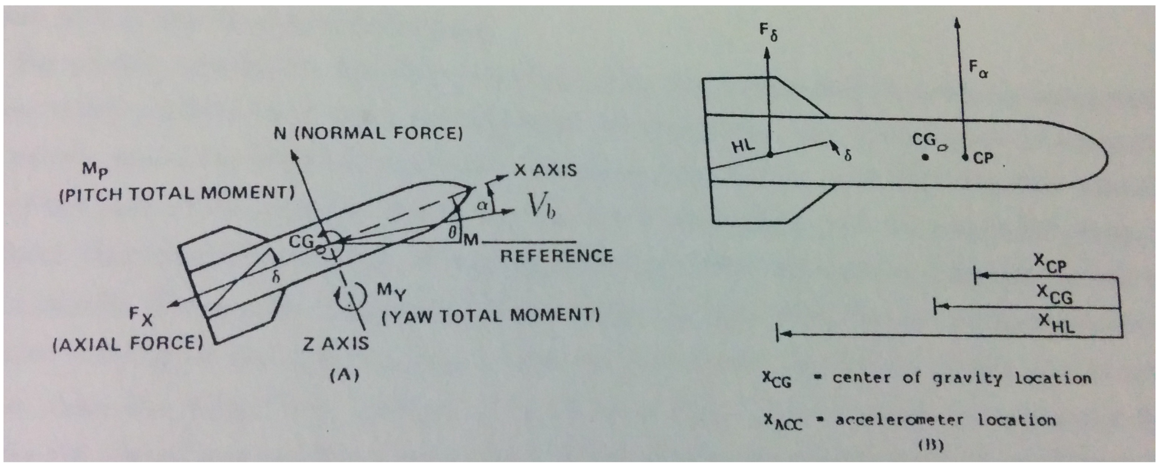

Figure 2.9: Visualization of CG_0 and V_b

I_{xx}, I_{yy}, I_{zz}, X_{CG}, their derivatives and mass denoted by m(t), are then included in the derivation of the missile dynamic equations. The time-derivative of Mass is accounted for in the missile thrust term.

Maximum Missile Acceleration Calculation. By Newton's law, F = ma, we understand that the acceleration varies inversely with respect to the mass. Thus missile will reach its maximum acceleration when maximum fuel mass is lost during its flight. Missile acceleration is measured in terms of G-force here. When the missile fligh time t > t_6, the ImpNorm would be 1 and missile will have the least mass. Going by the Equation 2.21, we calculate the maximum mass lost as follows.

$$
\begin{aligned}
Maximum\,Mass\,Lost &= ImpNorm(t)\,dm \\
&= 1 * 2.2689 \\
&= 2.2689\,slugs
\end{aligned}
$$

Thus the remaining missile mass is given by,

$$
\begin{aligned}
Remaining\,Mass &= m_0 - Maximum\,Mass\,Lost \\
&= 5.7500 - 2.2689 \\
&= 3.4811\,slugs = 50.8028\,kg = 498.205\,N
\end{aligned}
$$

Maximum Thrust is given as = 9250 lbs = 4195.729 kg = 41146.04591 N. The "g" force is calculated as follows,

$$
\begin{aligned}
G\,force &= \frac{Thrust}{Weight} \\
&= \frac{Thrust\,in\,N}{mass\,in\,kg * 9.8ms^{-2}} \\
&= \frac{Thrust\,in\,N}{Weight\,in\,N}
\end{aligned}
$$

Thus the maximum G force will occur when the thrust is max and weight is minimum.

$$
\begin{aligned}
Maximum\,G\,force &= \frac{Maximum\ Thrust}{Minimum\ Weight} \\
&= \frac{41146.04591\,N}{495.205\,N} \\
&= 82.58
\end{aligned}
$$

Thus the BTT missile in our considered can maximum pull up to 82.58 g's with its capabilities while trying to intercept a target. 82.58 g is really a considerable acceleration advantage over the target which is limited to maximum ± 9g, which a human pilot can endure.

2.4 Missile Aerodynamics

During the missile's flight through the atmosphere, it will experience aerodynamic forces, F_w and moments G_w [49]. The amount of lift generated and drag that must be overcome are greatly influenced by the missile's orientation with respect to its velocity vector, speed and the local dynamic air pressure.

If one defines the missile's body frame inertial speed, V_b as:

$$V_b \stackrel{\text{def}}{=} \sqrt{(U^2 + V^2 + W^2)} \tag{2.26}$$

then the *dynamic pressure* Q_{dp} is a function of the local air density ρ and is given by:

$$Q_{dp} = 0.5\rho V_b^2 \tag{2.27}$$

A critical parameter in this work is *Mach Number.* Mach number is defined as follows:

$$Mach \stackrel{\text{def}}{=} \frac{V_b}{SOS} \tag{2.28}$$

where SOS is the local speed-of-sound.

For missile velocity V_b less than the local SOS, the missile motion produces compression waves which radiate away from the missile in all directions. The wave motion in advance of the missile starts the local air molecules in motion, in a manner such that they flow smoothly out of the path of the missile. This is known as the sub-sonic flight and the missile lift, drag and sideforce characteristics depend on the smooth flow of the atmosphere over the surface area of the missile. For missile velocity V_b greater than the local SOS, the air molecules recieve no advance warning of the approaching missile and are forced rapidly out of the way at speeds greater than the local SOS, creating a shock wave [57]. This is known as supersonic flight. The lift, drag and sideforce properties of the missile are significantly changed from those properties at sub-sonic flight.

SOS and ρ are in turn functions of the missile's inertial altitude ${S_{mz}}^i$ and are modeled using equations fitted to a set of tabulated data [57], [58]. The tabulated data contains results of extensive wind-tunnel tests. SOS and ρ decrease with increasing altitude. More precisely, SOS varies as temperature and ρ varies as temperature (below 36088 ft) or altitude (above 36088 ft, geopotential height).

2.4.1 Stability and Control Derivatives

In order to express the external body frame forces (F_x, F_y, F_z) and body frame moments (L, M, N) in terms of the aerodynamic variables (α, β), aerodynamic parameter (Mach) and the controls (δ_p, δ_q, δ_r), it is convention to introduce the stability derivatives in Table 2.3 and the control derivatives in Table 2.4. These coefficients represent the partial derivatives of body frame forces (F_x, F_y, F_z) with respect to body linear velocities (U, V, W) and body frame moments (L, M, N) with respect to body frame angular velocities (P, Q, R). Stability derivatives are interpolated from aerodynamic coefficient arrays created using empirical values measured during the actual missile wind-tunnel tests. Tables are used becuase of the complex dependence on Mach number, angle of attack and sideslip angles. This simulation still uses 15 aerodynamic coefficients which are interpolated using Mach, $\alpha, \beta \& \delta_q$. Their parameter dependence is indicated in Table 2.3 and Table 2.4.

Aero Coefficients	Quantifies	Depends on
C_D	Drag from Pitch Fin Deflection	δ_q, Mach-1, α-1
C_{DT}	Base Drag due to Mach	Mach-2
C_{L_β}	Roll Moment from Sideslip	Mach-1, α-1
C_{L_P}	Roll Damping Moment - Pitch rate	Mach-1, α-3
C_{M_α}	Pitch Moment from Angle of Attack	Mach-1, α-4
C_{M_q}	Pitch Moment from Pitch Fin Deflection	Mach-2, α-3
C_{N_α}	Lift due to Mach	Mach-1
C_{N_β}	Yaw Moment from Sideslip	Mach-1, α-1
C_{N_R}	Yaw Damping Moment - Yaw Rate	Mach-2, β-2
C_{Y_β}	Side Force from Sideslip	Mach-1, α-1

Table 2.3: Stability Derivatives and Parameter Dependence

Aero Coefficients	Quantifies	Depends on
$C_{L_{\delta_p}}$	Roll Moment from Roll Fin Deflection	Mach-1, α-1
$C_{M_{\delta_q}}$	Pitch Moment from Pitch Fin Deflection	Mach-1, α-2
$C_{N_{\delta_q}}$	Lift from Pitch Fin Deflection	Mach-1, α-2
$C_{N_{\delta_r}}$	Yaw Moment from Yaw Fin Deflection	Mach-1, β-1
$C_{Y_{\delta_r}}$	Side Force from Yaw Fin Deflection	Mach-1, β-1

Table 2.4: Control Derivatives and Parameter Dependence

Polynomial Fit Data Models and Dynamic Implications

Stability & Control derivatives were studied for their complex dependencies on flight parameters. Polynomial fitting of these Stability & Control derivatives will cut down the computation time of simulation by several times as this will avoid matrix parsing of aerodynamic look up tables using interpolation.

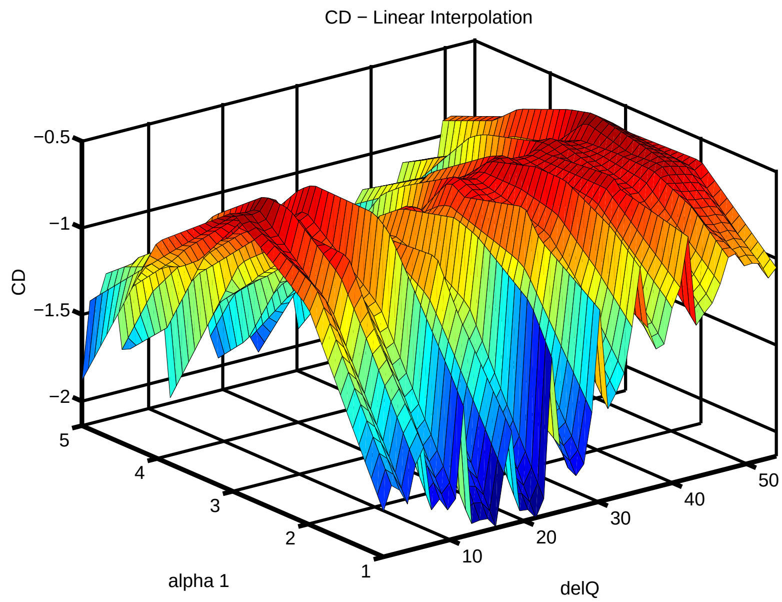

Figure 2.10: C_D - Drag from Pitch Fin Deflection - depends on δ_q, Mach-1, α-1

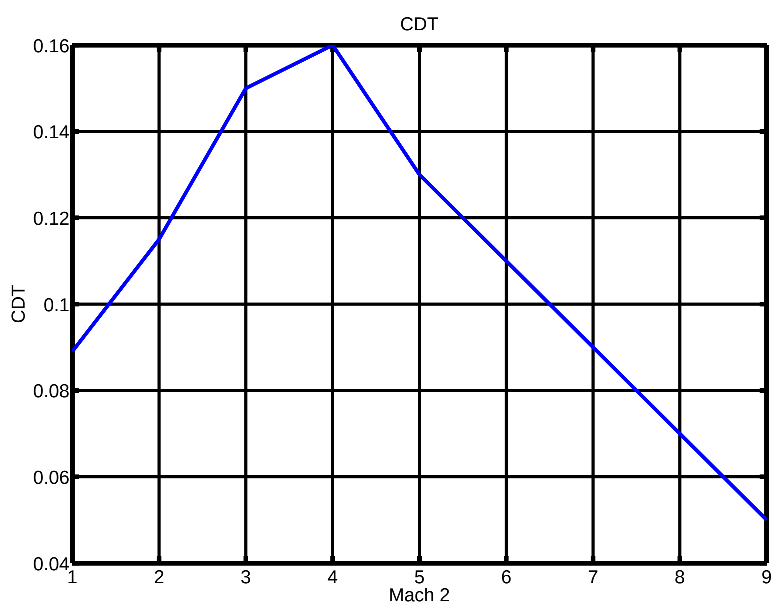

Figure 2.11: C_{DT} - Base Drag due to Mach-2

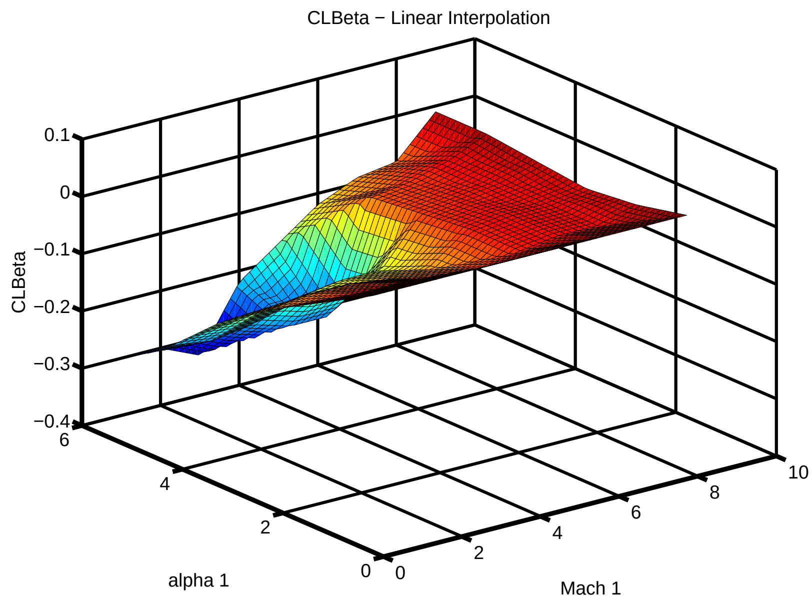

Figure 2.12: C_{L_β} - Roll Moment from Sideslip - depends on Mach-1, α-1

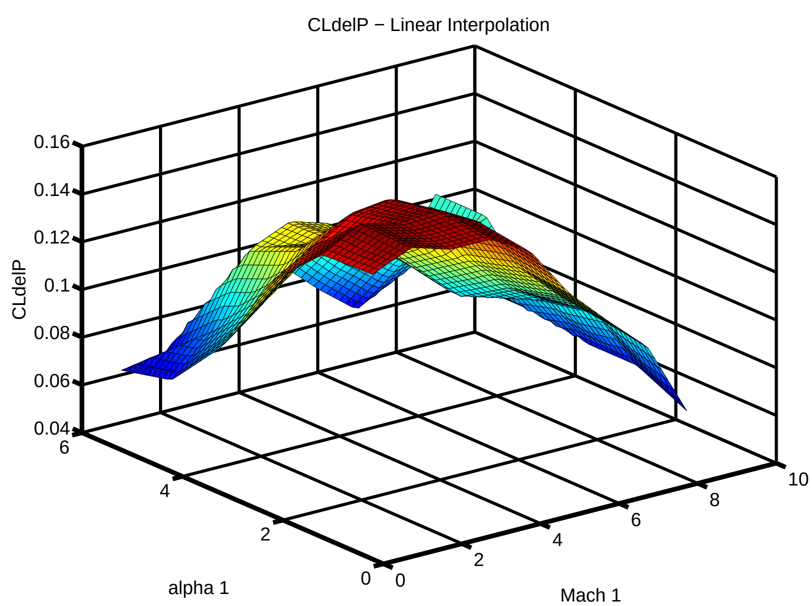

Figure 2.13: $C_{L_{\delta_p}}$ - Roll Moment from Roll Fin Deflection - depends on Mach-1, α-1

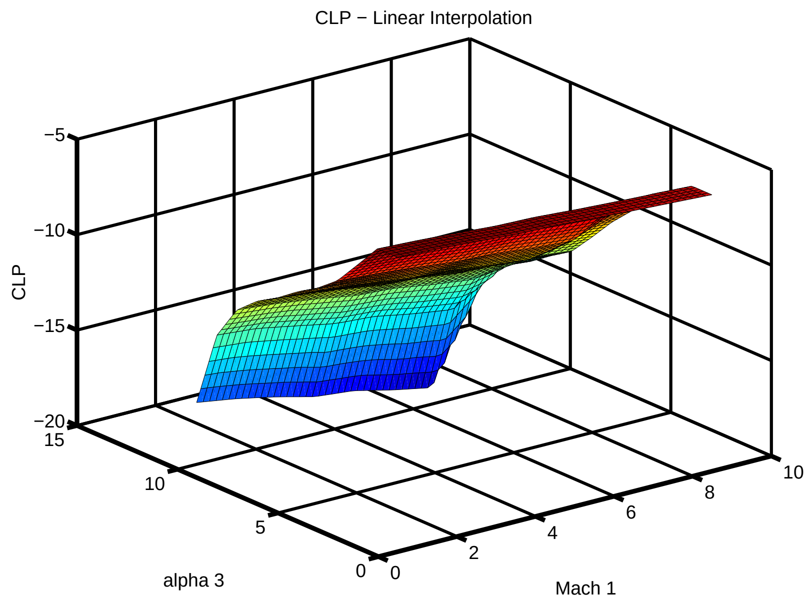

Figure 2.14: C_{L_P} - Roll Damping Moment - Pitch rate - depends on Mach-1, α-3

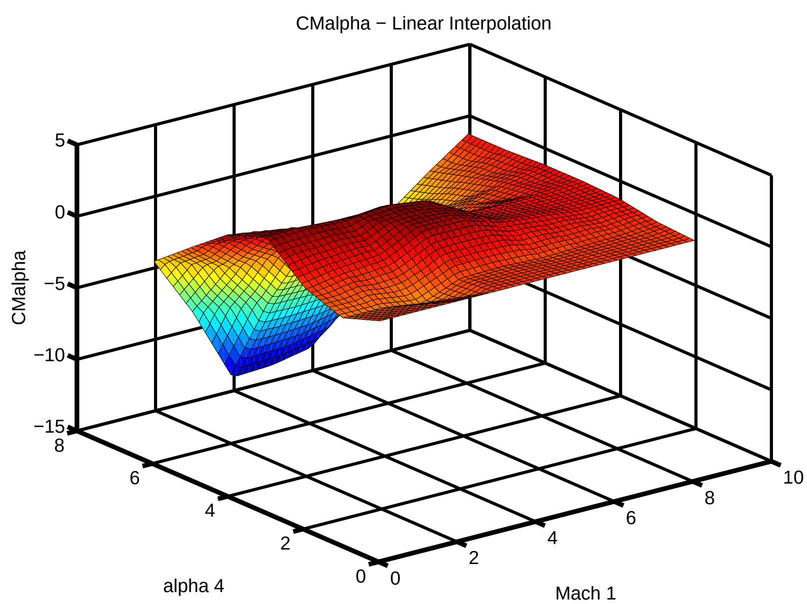

Figure 2.15: C_{M_α} - Pitch Moment from Angle of Attack - depends on Mach-1, α-4

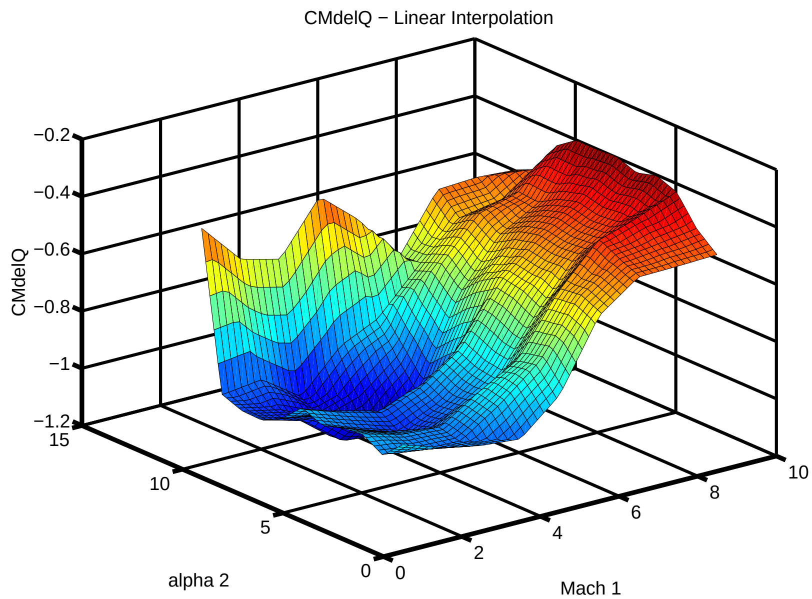

Figure 2.16: $C_{M_{\delta_q}}$ - Pitch Moment from Pitch Fin Deflection - depends on Mach-1, α-2

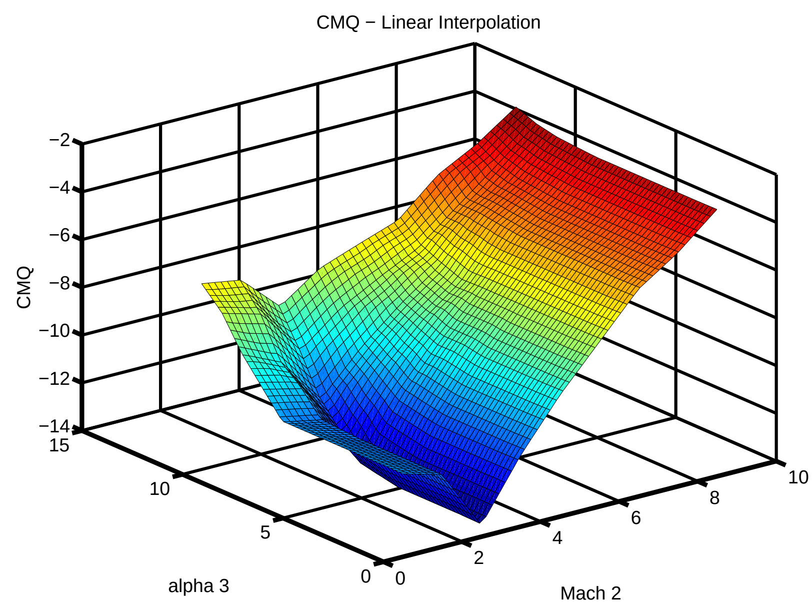

Figure 2.17: C_{M_q} - Pitch Moment from Pitch Fin Deflection - depends on Mach-2, α-3

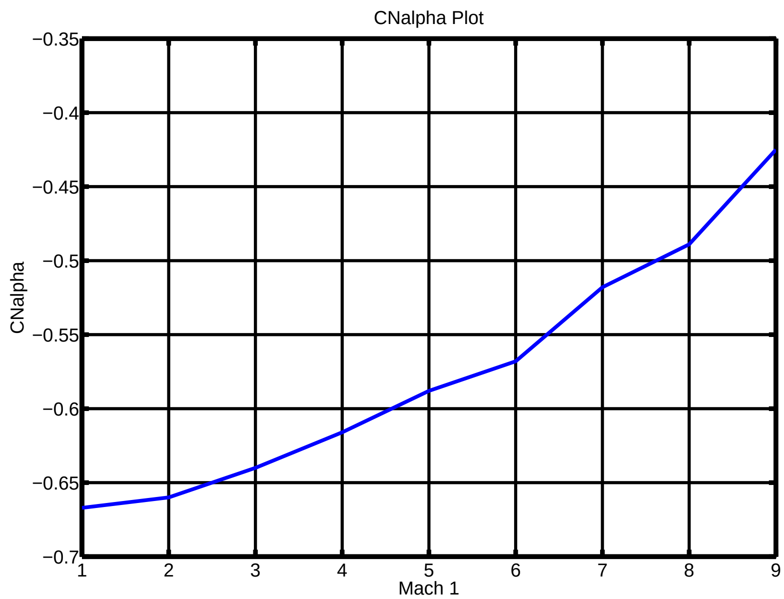

Figure 2.18: C_{N_α} - Lift due to Mach-1

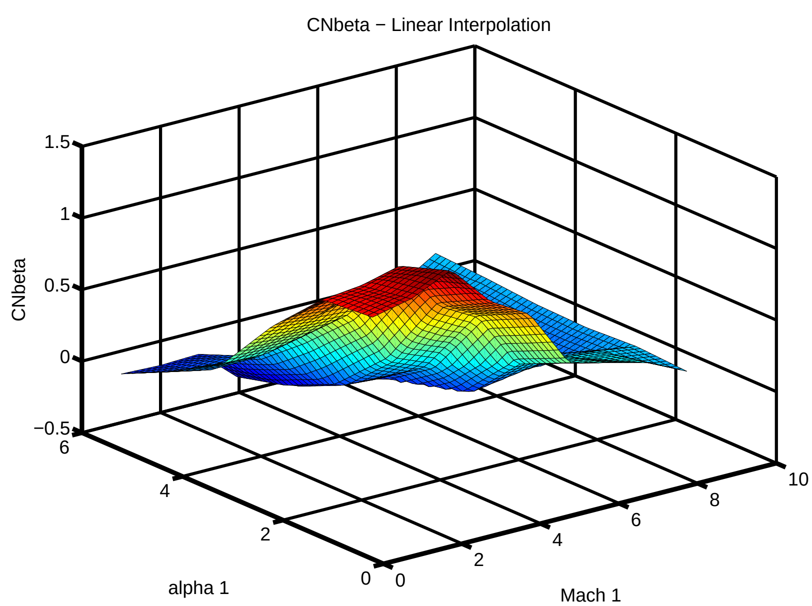

Figure 2.19: C_{N_β} - Yaw Moment from Sideslip - depends on Mach-1, α-1

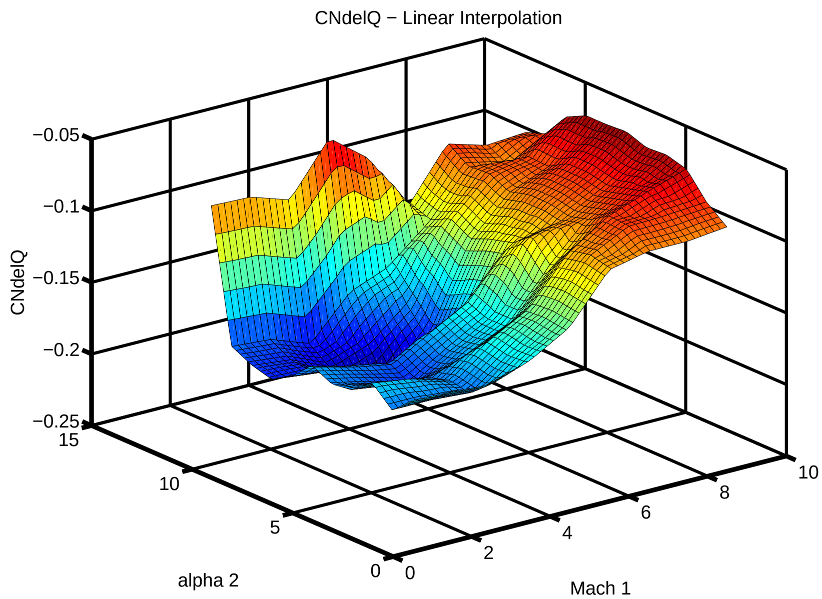

Figure 2.20: $C_{N_{\delta_q}}$ - Lift from Pitch Fin Deflection - depends on Mach-1, α-2

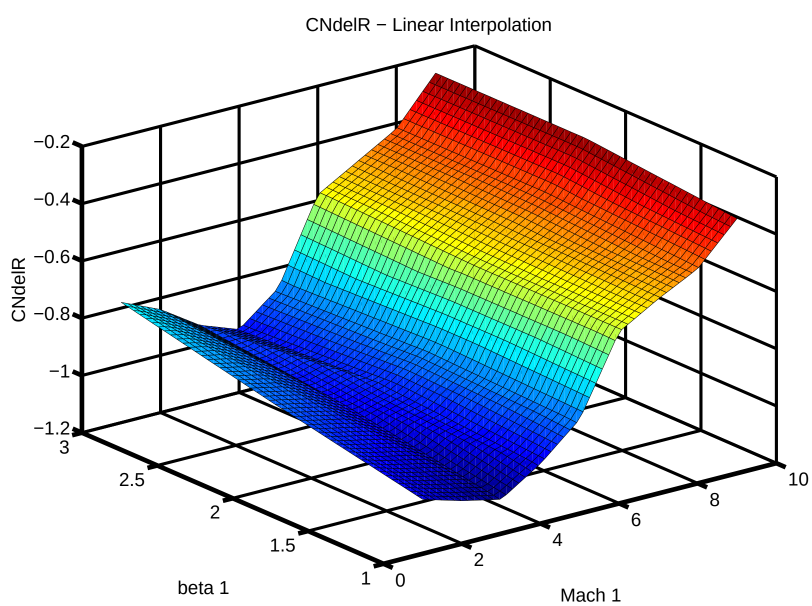

Figure 2.21: $C_{N_{\delta_r}}$ - Yaw Moment from Yaw Fin Deflection - depends on Mach-1, β-1

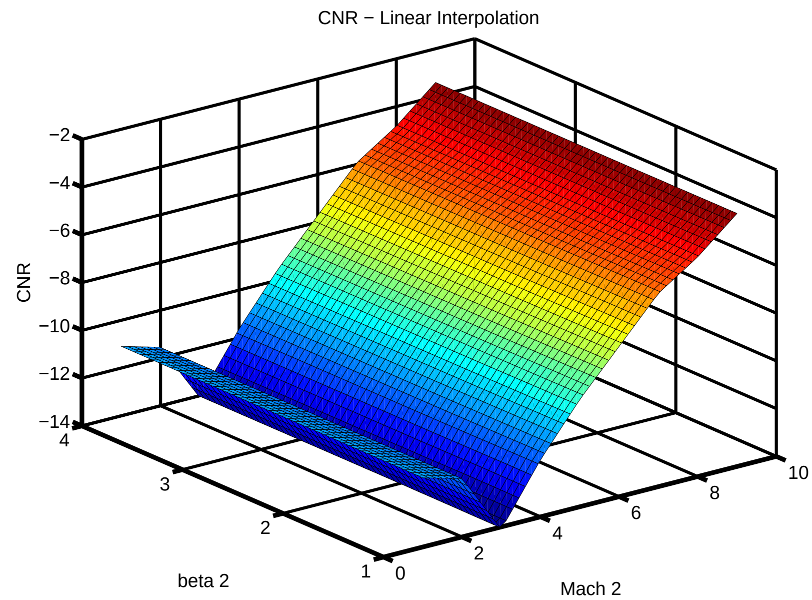

Figure 2.22: C_{N_R} - Yaw Damping Moment - Yaw Rate - depends on Mach-2, β-2

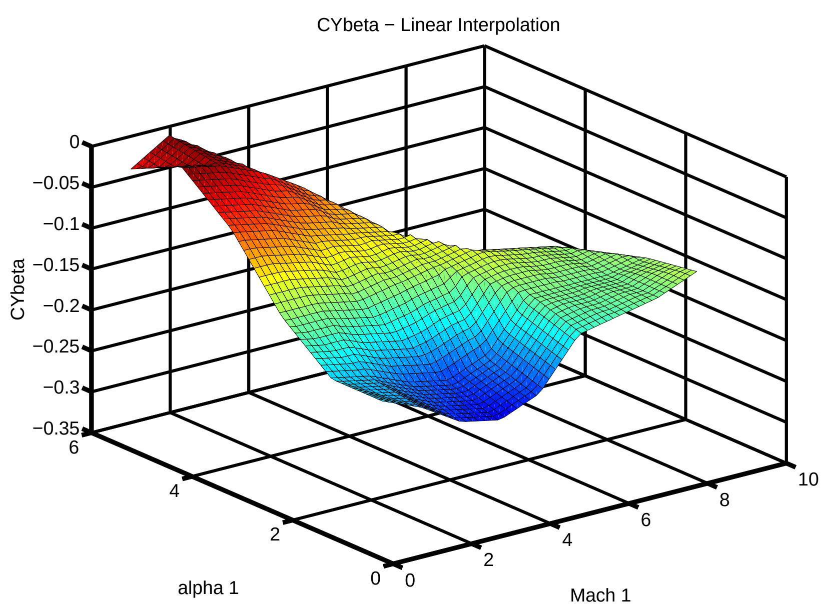

Figure 2.23: C_{Y_β} - Side Force from Sideslip - depends on Mach-1, α-1

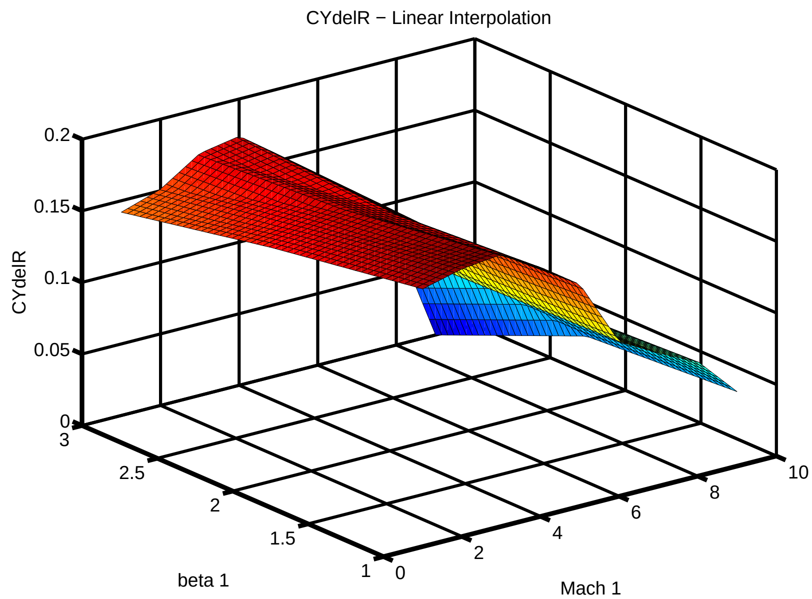

Figure 2.24: $C_{Y_{\delta_r}}$ - Side Force from Yaw Fin Deflection - depends on Mach-1, β-1

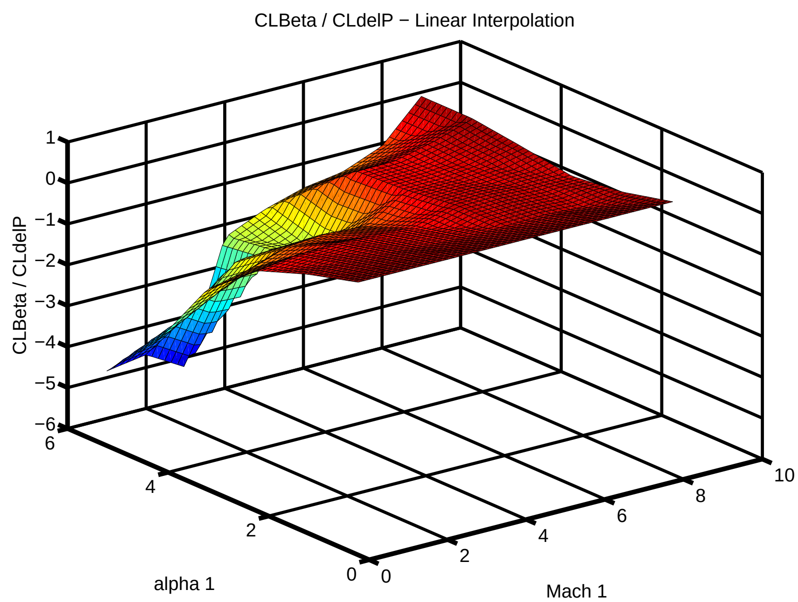

Figure 2.25: Scheduled Gain

2.4.2 Aerodynamic (Wind) Frame

The aerodynamic frame is defined so as to relate the missile body frame velocity and orientation to the external aerodynamic forces and moments. The stability and control derivatives are dependent on missile body rate information. The body frame linear velocities are transformed into the aerodynamic frame by the following equations.

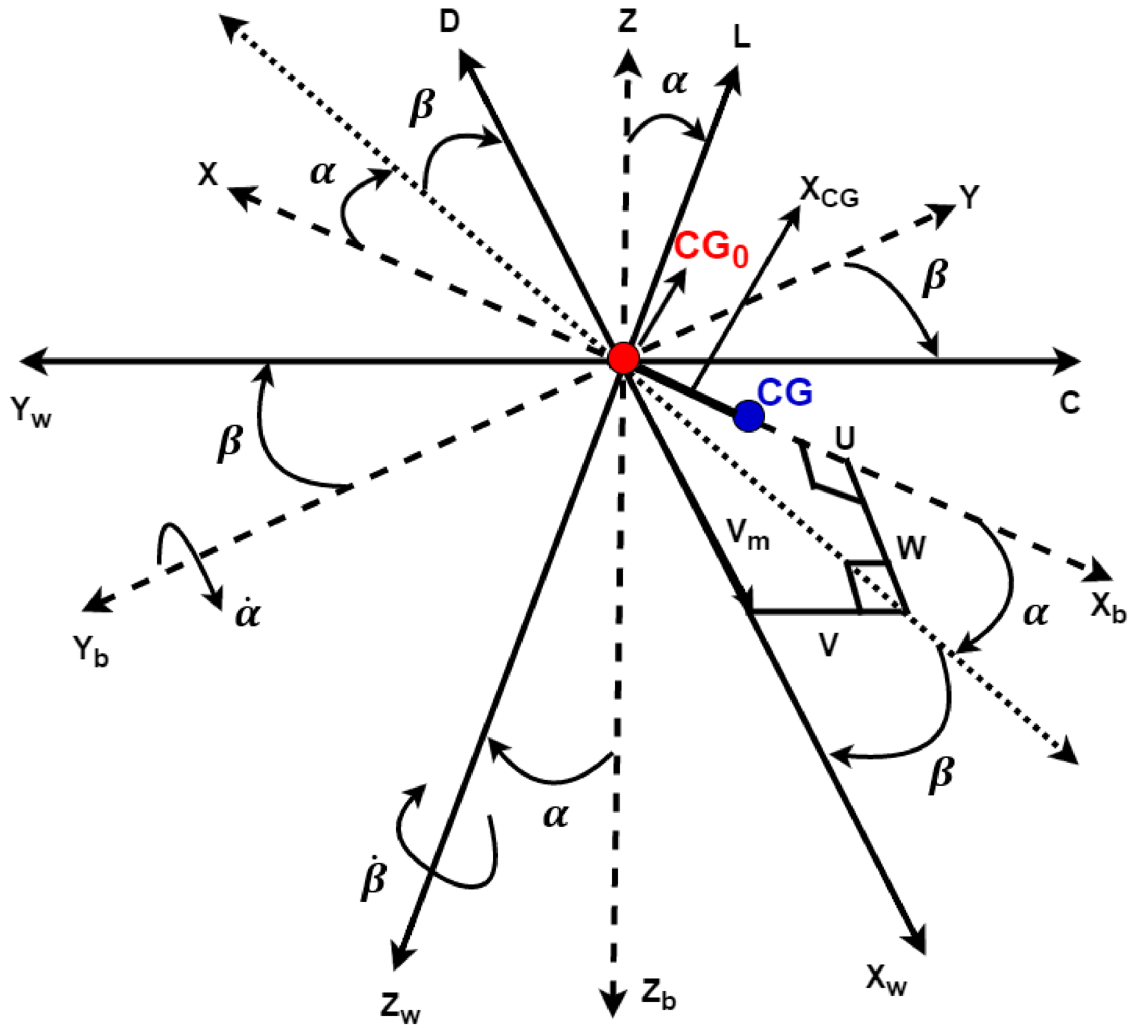

Figure 2.26: Aerodynamic Force, Body Velocity and Aerodynamic Angles

Figure 2.26 shows the aerodynamic force $F_w = (X, Y, Z)^w$, body velocity V_m and aerodynamic angles - α, β defined below. Velocity of the missile in wind frame is given by:

$$V_m{}^w = (V_b, 0, 0)^w \tag{2.29}$$

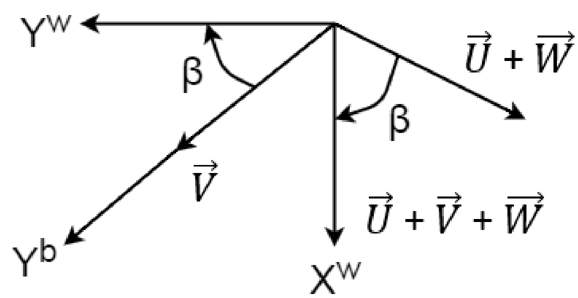

Figure 2.27: Visualization of Sideslip Angle, β

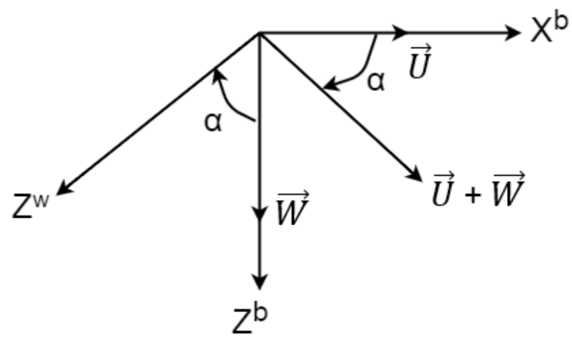

Figure 2.28: Visualization of Angle of Attack, α

The orientation of the wind frame will now be discussed. The wind frame has its origin at the missile center of gravity CG_0, with X^w in the plane defined by X^w and Y^b as shown in the Figure 2.28. Z^w is defined to be orthogonal to $\vec{U} + \vec{W}$ in the X^bZ^b plane as shown in Figure 2.27. We now define two key aerodynamic variables: (1) *Angle of Attack* α and (2) *Sideslip Angle* β. These quantities are defined in terms of the body axis velocities (U, V, W) as follows. The *Angle of Attack*, denoted by α is defined as shown in Figure 2.12 as the angle from Z^b to Z^w, i.e.

$$\alpha \stackrel{\text{def}}{=} \angle Z^b Z^w \tag{2.30}$$

From figure 2.28, it also follows that

$$\alpha \stackrel{\text{def}}{=} tan^{-1}\frac{W}{U} \tag{2.31}$$

The *Sideslip Angle*, denoted by β is defined as the angle from $\vec{U}$ + $\vec{W}$ to $\vec{U}$ + $\vec{V}$ + $\vec{W}$, measured in the plane formed by $\vec{U}$ + $\vec{W}$ and Y^b: The *Sideslip Angle*, denoted by β is defined as shown in Figure 2.27 as the angle from Y^b to Y^w, i.e.

$$\beta \stackrel{\text{def}}{=} \angle Y^b Y^w \tag{2.32}$$

From Figure 2.27, it also follows that

$$\beta \stackrel{\text{def}}{=} tan^{-1}\frac{V}{\sqrt{(U^2 + W^2)}} \tag{2.33}$$

2.4.3 Force and Moment Coefficients

Stability derivatives are multiplied with dynamic parameter values to form body frame force coefficients and body frame coefficients. Figure 2.29 [59] shows the aerodynamic forces (F_x, F_y, F_z), moments (L, M, N), body linear velocities (U, V, W) and body angular velocities (P, Q, R). Their notation is defined in the Table 2.5.

Direction/Rotation	Velocity	Forces & Moments	Distances
Forward	U	X = F_x	x
Side	V	Y = F_y	y
Vertical	W	Z = F_z	z
Roll	P	L	
Pitch	Q	M	
Yaw	R	N	

Table 2.5: Body Frame Force and Moment Notation

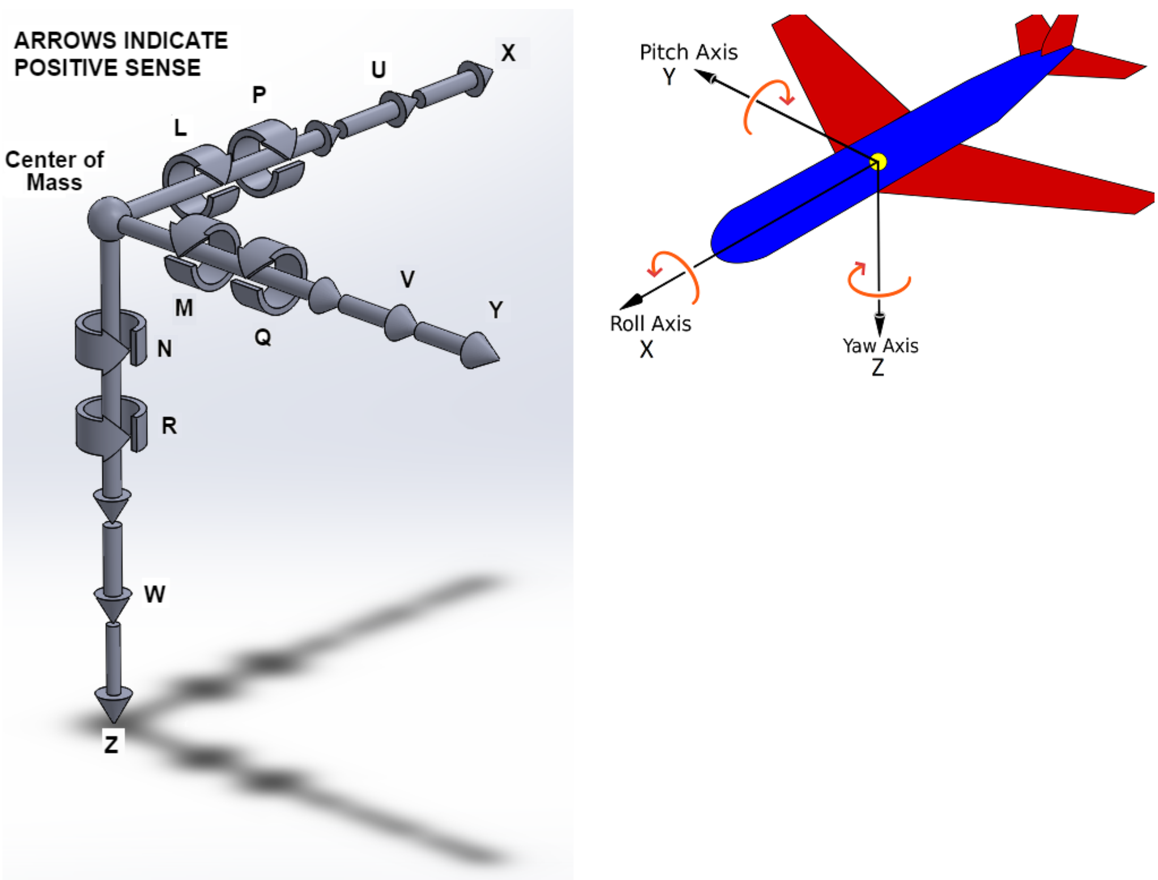

Figure 2.29: Body Frame Axis System and Notation

The symbols δ_p, δ_q, δ_r are equivalent to aileron, elevator and rudder deflections for an aircraft. They are related to the actual fin deflections via an Integrated Logic for Air-to-Air Technology (ILAAT) demixer [48].

The drag, side force and lift coefficients are:

$$C_X = C_D + C_{DT} \tag{2.34}$$

$$C_Y = C_{Y_\beta}\beta + C_{Y_{\delta_r}}\delta r \tag{2.35}$$

$$C_Z = C_{N_\alpha}\alpha + C_{N_{\delta_q}}\delta q \tag{2.36}$$

and the roll, pitch and yaw moment coefficients are

$$C_L = C_{L_{\delta_p}}\delta_p + C_{L_P} P L_{2V} + C_{L_\beta}\beta \tag{2.37}$$

$$C_M = C_{M_{\delta_q}}\delta_q + C_{M_Q} Q L_{2V} + C_{M_\alpha}\alpha \tag{2.38}$$

$$C_N = C_{N_{\delta_r}}\delta_r + C_{N_R} R L_{2V} + C_{N_\beta}\beta \tag{2.39}$$

where

$$L_{2V} \stackrel{\text{def}}{=} \frac{L_{ref}}{2} V_b \tag{2.40}$$

$L_{ref} = 0.625$ (ft) is an effective reference missile length used to describe moments about CG. C_{M_α} determines whether the airframe is statically stable [62]. A missile is statically stable if it returns to its equilibrium point after encountering a small disturbance [58].

2.4.4 Aerodynamic Forces (F_x, F_y, F_z) and Moments (L, M, N)

The aerodynamic force coefficients are converted into forces by multiplication with the local dynamic pressure, missile effective cross sectional area $S_{ref} = 0.307\ (ft^2)$ and missile mass denoted by m:

$$F_x = C_X Q_{dp} S_{ref} m \tag{2.41}$$

$$F_y = C_Y Q_{dp} S_{ref} m \tag{2.42}$$

$$F_z = C_Z Q_{dp} S_{ref} m \tag{2.43}$$

The moment coefficients are similarly converted into aerodynamic moments about the body frame XYZ axes by multiplying with Q_{dp}, S_{ref} and L_{ref}:

$$L = C_L Q_{dp} S_{ref} L_{ref} \tag{2.44}$$

$$M = C_M Q_{dp} S_{ref} L_{ref} \tag{2.45}$$

$$N = C_N Q_{dp} S_{ref} L_{ref} \tag{2.46}$$

2.4.5 Gravitational Forces and Moments

In this missile simulation program, the gravitational force of attraction is modeled as an external force $(F_{g_x}, F_{g_y}, F_{g_z})^b$ acting at the missile instantaneous center-of-gravity CG. For a CG displaced from the body frame CG_0, the external force $(F_{g_x}, F_{g_y}, F_{g_z})^b$ causes an external moment $(G_{g_x}, G_{g_y}, G_{g_z})^b$ about the CG_0.

Gravitational Acceleration, g

The acceleration **g** of the Earth's gravity decreases with altitude as a function of $\frac{1}{R^2}$, R $\stackrel{\text{def}}{=}$ radial distance from the center of the Earth and can be written as:

$$g = g_0 \sqrt{\frac{R_0}{R_0 + h_i}} \tag{2.47}$$

where

$h_i \stackrel{\text{def}}{=}$ the inertial altitude of missile $= (S_z)^i$

$R_0 \stackrel{\text{def}}{=}$ sea-level radius of Earth = 20,903,264 ft

$g_0 \stackrel{\text{def}}{=}$ sea-level value for gravity = 32.174 $\frac{ft}{sec^2}$

Gravitational Acceleration

For the missile-target engagement, g is modeled as an inertial acceleration $(A_{g_x}, A_{g_y}, A_{g_z})^i$ $= [0, 0, g]^i$ and when transformed into the body frame is denoted as $(A_{g_x}, A_{g_y}, A_{g_z})^b$.

Gravitational Force and Moment

The external gravitational force $(F_{g_x}, F_{g_y}, F_{g_z})^b$ acting on the missile CG is given by:

$$(F_{g_x}, F_{g_y}, F_{g_z})^b = m(A_{g_x}, A_{g_y}, A_{g_z})^b \tag{2.48}$$

and therefore the components are

$$F_{g_x} = mA_{g_x} \tag{2.49}$$

$$F_{g_y} = mA_{g_y} \tag{2.50}$$

$$F_{g_z} = mA_{g_z} \tag{2.51}$$

An external gravitational moment, $(G_{g_x}, G_{g_y}, G_{g_z})^b$ results because the force $(F_{g_x}, F_{g_y}, F_{g_z})^b$ is applied at the missile CG, which is displaced from body frame origin by a distance $(S_{c_x}, S_{c_y}, S_{c_z})^b$. The moment equation is given by:

$$(G_{g_x}, G_{g_y}, G_{g_z})^b = (S_{c_x}, S_{c_y}, S_{c_z})^b \times (F_{g_x}, F_{g_y}, F_{g_z})^b \tag{2.52}$$

The CG motion is confined along the body frame X-axis for this model, the last equation expands as:

$$G_{g_x} = 0 \tag{2.53}$$

$$G_{g_y} = -S_{c_x} F_{g_z} \tag{2.54}$$

$$G_{g_z} = S_{c_x} F_{g_y} \tag{2.55}$$

2.5 Equations of Motion for the Missile

A complete set of 6DOF nonlinear equations can be described by summing all external forces and moments, (external forces being defined as the aerodynamic and gravitational forces and moments), acting on the missile and setting them equal to the forces and moments due to the missile inertial acceleration. Inertial and external components are indicated by i and e subscripts respectively.

Translational Dynamics

Inertial accelerations expresses in body frame $(X, Y, Z)^b$ are:

$$A_{ix} = \dot{U} + QW - RV + \ddot{X}_{CG} - X_{CG}(Q^2 + R^2) \tag{2.56}$$

$$A_{iy} = \dot{V} + RU - RW + 2R\ddot{X}_{CG} - X_{CG}(PQ + \dot{R}) \tag{2.57}$$

$$A_{iz} = \dot{W} + PV - QU + 2Q\ddot{X}_{CG} - X_{CG}(PR - \dot{Q}) \tag{2.58}$$

where $X_{CG}(t)$ is given by the Equation (2.25).

The second derivative of X_{CG} is retained to model effects during rocket thrust transients. X_{CG} is defined as the distance the instantaneous missile CG is displaced from the body frame origin CG_0, $X_{CG} = S_{c_x}$.

These inertial accelerations (A_{ix}, A_{iy}, A_{iz}) are set equal to the accelerations caused by external forces (A_{ex}, A_{ey}, A_{ez}), which include: (1) the aerodynamic forces, (2) rocket thrust and (3) gravity transformed into the body frame:

$$A_{ex} = \frac{F_x}{Mass} + {F_{g_x}}^b + \frac{Thrust}{Mass} \tag{2.59}$$

$$A_{ey} = \frac{F_y}{Mass} + {F_{g_y}}^b \tag{2.60}$$

$$A_{ez} = \frac{F_z}{Mass} + {F_{g_z}}^b \tag{2.61}$$

where (F_x, F_y, F_z) are specified by equations (2.41) - (2.43), thrust is specified by equations (2.13) - (2.14), m is specified by the equation (2.21) and $(F_{g_x}, F_{g_y}, F_{g_z})^b$ are given in equations (2.49) - (2.51).

Rotational Dynamics

Inertial moments about the body frame XYZ axes are described by:

$$L_i = \dot{P} I_{xx} + P \dot{I}_{xx} + QR(I_{zz} - I_{yy}) \tag{2.62}$$

$$M_i = \dot{Q} I_{yy} + Q \dot{I}_{yy} + RP(I_{xx} - I_{zz}) \tag{2.63}$$

$$N_i = \dot{R} I_{zz} + R \dot{I}_{zz} + PQ(I_{yy} - I_{xx}) \tag{2.64}$$

where (I_{xx}, I_{yy}, I_{zz}) are specified by equations (2.22) - (2.24).

These inertial moments (L_i, M_i, N_i) are set equal to the sum of external moments (L_e, M_e, N_e) acting on the missile, which include: (1) aerodynamic moments and (2) the two moments due to gravity:

$$L_e = L \tag{2.65}$$

$$M_e = M - G_{g_z}{}^b \tag{2.66}$$

$$N_e = N + G_{g_y}{}^b \tag{2.67}$$

where (L, M, N) are specified by equations (2.44) - (2.46).

2.6 Actuator Dynamics

The missile under study is a tail controlled missile. Missile control is achieved by appropriately coordinating four fins. Fin commands $(F_{1c}, F_{2c}, F_{3c}, F_{4c})$ are generated by the autopilot, to be discussed in Chapter 6. The fin commands drive four nonlinear actuator servo mechanisms, whose outputs are the actual fin deflections (F_1, F_2, F_3, F_4). each actuator is modelled as shown in Figure 2.30.

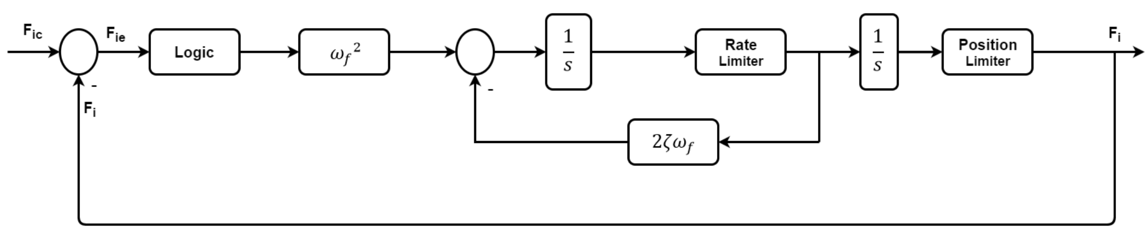

Figure 2.30: Model for Nonlinear Fin Actuators / Servomechanisms

Neglecting nonlinearities, each servo has a transfer function from F_{ic} to F_i given by:

$$H_i(s) = \left[\frac{\omega_f{}^2}{s^2 + 2\zeta_f\omega_f s + \omega_f{}^2}\right] \tag{2.68}$$

where, i = 1, 2, 3, 4 and

$\zeta_f \stackrel{\text{def}}{=}$ the damping ratio of the fin actuator and is equal to 0.3,

$\omega_f \stackrel{\text{def}}{=}$ the servo undamped natural frequency and is equal to 195.0077 $\frac{rad}{sec}$

Fin deflections are limited in position to

$$F_{max} = \pm 20 deg \tag{2.69}$$

and in rate to

$$\dot{F_{max}} = \pm 600 \frac{deg}{sec} \tag{2.70}$$

Also in the Logic block, the difference between the commanded fin deflection angle and the actual fin deflection angle is set to zero if it is less than 0.05 degrees.

2.7 Summary and Conclusions

In this chapter,the six degree-of-freedom nonlinear missile dynamics were described. The three reference frames, (1) Inertial frame, (2) Vehicle frame and (3) Body frame used to develop the equations of motion were introduced. The loss of mass through fuel consumption was mathematically described, because this mass loss will influence missile dynamics and must be accounted for in a realistic manner. The aerodynamic relationships were discussed. The gravitational model used in this simulation was described. The equations of motion for the missile were presented. The missile's fin actuator dynamics were described in the last section.

Chapter 3

LINEARIZED MISSILE MODEL ANALYSIS

3.1 Introduction and Overview

The governing nonlinear missile equations of motion were presented in Chapter 2. To use modern multivariable control theory or classical control theory to design autopilots requires that the missile equations of motion be in a linear time-invariant state-space form. Thus, it is necessary to linearize the nonlinear equations about trimmed flight conditions, or equilibrium points, to yield linear equations that accurately describe the missile's dynamic behavior. This appendix presents the derivation of the governing linear equations of motion, often called *perturbation equations*, for several trimmed flight conditions. In addition, the eigenvalues of the linear equations motion about selected equilibrium points are presented and the most significant factors that influence these modes are discussed.

The chapter is organized as follows. Section 3.2 presents the perturbation technique used to linearize the missile 6DOF equations and generate linear time-invariant state space systems which can be controlled using modern multivariable control theory or classical control theory. Section 3.3 throws light on selection of equilibrium points while linearizing the missile dynamics. Section 3.4 discusses the time scaling used to scale the linear system. Following the linear model generation, section 3.5 talks about decoupled longitudinal and lateral model and particular emphasis has been given on explaining the nonminimum phase and unstable pole dynamic behaviour in the decoupled models. After that section 3.6 discusses the static analysis of missile performed on trim elevator deflection. The causes for missile fin deflection saturation is explained in detail. Finally section 3.7 concludes the chapter.

3.2 Linear Equations of Motion

As discussed in the previous chapter, the nonlinear governing equations can be put into state space form described by the following compact notation:

$$\dot{x} = f(t, x, u) \tag{3.1}$$

By definition, (x^*, u^*) [1] is an equilibrium point of Equation 3.1 for all t $\geq$ 0.

$$f(t, x^*, u^*) = 0 \tag{3.2}$$

where x^* and u^* are the state and input (control) vectors respectively.

In the linearization of the nonlinear EOM we will make use of the Taylor series expansion of Equation 3.1. Taking the Taylor series expansion of Equation 3.1 and neglecting all 2nd order and higher terms yields:

$$\dot{x} = f(t, x^*, u^*) + \left.\frac{\partial f}{\partial x}\right|_{(x^*,u^*)} (x - x^*) + \left.\frac{\partial f}{\partial u}\right|_{(x^*,u^*)} (u - u^*) \tag{3.3}$$

We can express the states and the inputs as a linear combination of their respective equilibrium values and a perturbation value that represents their change due to a disturbance from their equilibrium values. Thus, we can write

$$\begin{aligned} \dot{x} &= \dot{x}^* + \Delta\dot{x} = f(t, x^*, u^*) + \Delta\dot{x} = \Delta\dot{x} \\ x &= x^* + \Delta x \\ u &= u^* + \Delta u \end{aligned} \tag{3.4}$$

[1]Throughout this chapter, all variables with supercript * correspond to their respective equilibrium values

Using Equation 3.4, we can rewrite Equation 3.3 as follows:

$$\Delta \dot{x} = \left.\frac{\partial f}{\partial x}\right|_{(x^*,u^*)} \Delta x + \left.\frac{\partial f}{\partial u}\right|_{(x^*,u^*)} \Delta u \tag{3.5}$$

Equation 3.5 describes the linear dynamic behavior of a nonlinear system *about an equilibrium point* under the assumption that the perturbations are “small”. Similarly, the nonlinear output equations (as of yet, unspecified) can be linearized using a Taylor series expansion and retaining only the first order terms as follows:

$$\Delta y = \left.\frac{\partial g}{\partial x}\right|_{(x^*,u^*)} \Delta x + \left.\frac{\partial g}{\partial u}\right|_{(x^*,u^*)} \Delta u \tag{3.6}$$

Employing the above linearization procedure we can write the linear state-space perturbation equations of the nonlinear equations of motion presented in Equations 2.56 - 2.58 and 2.62 - 2.62. The linear time-invariant state-space equations are given as follows:

$$\begin{aligned} \Delta \dot{x} &= A\Delta x + B\Delta u \\ \Delta y &= C\Delta x + D\Delta u \end{aligned} \tag{3.7}$$

where

$$\begin{aligned} A = \left.\frac{\partial f}{\partial x}\right|_{(x^*,u^*)}, B &= \left.\frac{\partial f}{\partial u}\right|_{(x^*,u^*)} \\ C = \left.\frac{\partial g}{\partial x}\right|_{(x^*,u^*)}, D &= \left.\frac{\partial g}{\partial u}\right|_{(x^*,u^*)} \end{aligned} \tag{3.8}$$

Before proceeding with the linearization of the nonlinear state and output equations using small perturbation theory, we will make the following simplifying assumptions.

Assumptions/Idealizations/Approximations used in Linearization of the EOM:

1. Changes in the local air density, ρ, are "small" relative to perturbations of the other variables about equilibrium points of interest. This assumption simplifies the expansion of the perturbed dynamic pressure, ΔQ_{dp}, into being only dependent on the missile's velocity perturbation, ΔV_b. Thus, $\Delta\ Q_{dp} = (\ \rho^* V_b{}^*)\ \Delta V_b$. This assumption is valid along as long as changes in the missiles altitude are "small" about equilibrium points.

2. The missile's mass properties m, X_{CG}, I_{xx}, I_{yy} and I_{zz} are dependent only upon time t, because the propulsive thrust is modeled as time scheduled thrust profile (e.g., versus a throttle-controlled thrust, $\delta_{throttle}$). The perturbations in these parameters can be effectively modeled as time dependent disturbances acting on each of the six EOM. Thus, in the linearization procedure that follows, we will ignore their time variation and the set their respective time derivatives to zero, and effectively treat them as constants. However, since the constant part of the time varying parameters does affect the characteristic modes of the missile, we will evaluate the linear EOM at different "snap-shots" in flight time with the corresponding values of the mass properties at this instant in flight time. We will assume, for simplicity only, that $X_{CG} = 0$, at all steady-flight conditions (equilibrium points).

3. Fin actuator dynamics will be ignored in the linearized state-space EOM. This is usually a valid assumption because the bandwidths of servo actuators are usually specified (designed) to be higher than that of the expected controller bandwidth such that the dominant dynamics are that of the plant and not that of the actuators.

The nonlinear BTT missile state equations from Chapter 2 are given below for convenience (where the fin actuator dynamics have been neglected under idealization (3) given above):

$$\dot{U} = \frac{X}{m} - QW + RV + X_{CG}(Q^2 + R^2) \tag{3.9}$$

$$\dot{V} = \frac{Y}{m} - RU + PW - 2R\ddot{X}_{CG} + X_{CG}(PQ + \dot{R}) \tag{3.10}$$

$$\dot{W} = \frac{Z}{m} - PV + QU - 2Q\ddot{X}_{CG} + X_{CG}(PR - \dot{Q}) \tag{3.11}$$

$$\dot{P} = \frac{L}{I_{xx}} - \frac{P\dot{I}_{xx}}{I_{xx}} - \frac{QR(I_{zz} - I_{yy})}{I_{xx}} \tag{3.12}$$

$$\dot{Q} = \frac{M}{I_{yy}} - \frac{Q\dot{I}_{yy}}{I_{yy}} - \frac{RP(I_{xx} - I_{zz})}{I_{yy}} \tag{3.13}$$

$$\dot{R} = \frac{N}{I_{zz}} - \frac{R\dot{I}_{zz}}{I_{zz}} - \frac{PQ(I_{yy} - I_{xx})}{I_{zz}} \tag{3.14}$$

where

$$\begin{aligned} X &= F_{X_{aero}} + F_{X_g} + T_X \\ Y &= F_{Y_{aero}} + F_{Y_g} \\ Z &= F_{Z_{aero}} + F_{Z_g} \\ L &= L_{aero} \\ M &= M_{aero} + M_g \\ N &= N_{aero} + N_g \end{aligned} \tag{3.15}$$

and

$$
\begin{aligned}
F_{X_{aero}} &= Q_{dp} S_{ref} C_X = Q_{dp} S_{ref} (C_D + C_{DT}) \\
F_{Y_{aero}} &= Q_{dp} S_{ref} C_Y = Q_{dp} S_{ref} (C_{Y_\beta} \beta + C_{Y_{\delta_r}} \delta_r) \\
F_{Z_{aero}} &= Q_{dp} S_{ref} C_Z = Q_{dp} S_{ref} (C_{N_\alpha} \alpha + C_{N_{\delta_q}} \delta_q) \\
L_{aero} &= Q_{dp} S_{ref} L_{ref} C_L = Q_{dp} S_{ref} L_{ref} (C_{L_{\delta_p}} \delta_p + C_{L_p} (L_{ref}/2V_b) P + C_{L_\beta} \beta) \\
M_{aero} &= Q_{dp} S_{ref} L_{ref} C_M = Q_{dp} S_{ref} L_{ref} (C_{M_{\delta_q}} \delta_q + C_{M_q} (L_{ref}/2V_b) Q + C_{M_\alpha} \alpha) \\
N_{aero} &= Q_{dp} S_{ref} L_{ref} C_N = Q_{dp} S_{ref} L_{ref} (C_{N_{\delta_r}} \delta_r + C_{N_r} (L_{ref}/2V_b) R + C_{N_\beta} \beta)
\end{aligned}
\tag{3.16}
$$

and

$$
\begin{aligned}
F_{X_g} &= -mg \sin(\theta) \\
F_{Y_g} &= mg \cos(\theta) \sin(\phi) \\
F_{Z_g} &= mg \cos(\theta) \cos(\phi) \\
L_g &= 0 \\
M_g &= -X_{CG}\, mg \cos(\theta) \cos(\phi) \\
N_g &= X_{CG}\, mg \cos(\theta) \sin(\phi)
\end{aligned}
\tag{3.17}
$$

The above equations can be put into the following compact notation, where the output equations are dependent on the available measurements and the variables to be controlled.

State equations:

$$\dot{x} = f(t, x, u) \tag{3.18}$$

Output equations:

$$y = g(t, x, u) \tag{3.19}$$

Under assumption (2), Equations 3.9 - 3.14 reduce to:

$$\dot{U} = \frac{X}{m} - QW + RV \tag{3.20}$$

$$\dot{V} = \frac{Y}{m} - RU + PW \tag{3.21}$$

$$\dot{W} = \frac{Z}{m} - PV + QU \tag{3.22}$$

$$\dot{P} = \frac{L}{I_{xx}} - \frac{QR(I_{zz} - I_{yy})}{I_{xx}} \tag{3.23}$$

$$\dot{Q} = \frac{M}{I_{yy}} - \frac{RP(I_{xx} - I_{zz})}{I_{yy}} \tag{3.24}$$

$$\dot{R} = \frac{N}{I_{zz}} - \frac{PQ(I_{yy} - I_{xx})}{I_{zz}} \tag{3.25}$$

For the linearized state-space system we will make the following steady flight condition assumptions.

Assumptions about Steady Flight Conditions:

1. The steady trimmed flight condition is one of uniform translational motion, i.e., where the equilibrium angular rates are zero. Thus P* = Q* = R* = 0, where all starred, "*", variables will indicate equilibrium values of the variables.

2. The sideslip angle, β, is taken to be zero. This is a valid assumption since one of the requirements of the BTT missile autopilot is to minimize the sideslip angle during flight. Thus, V* = 0.

3. The bank angle, ϕ and the yaw angle, ψ, are taken to be zero.

4. The steady, or equilibrium, thrust level will taken two be that of the second stage (2140 lbf) but corrected for altitude for all trimmed flight conditions. We assume this level of thrust because the missile probably will spend most of its flight time at this stage (the first stage being relatively short in duration). Also, we will assume that the level of thrust is constant even in perturbed flight about equilibrium points.

5. The missile's mass properties change with flight time, as discussed in Chapter 2; However, for simplicity we assume that they are constant about trimmed flight conditions. The eigenvalues of the linear EOM will be evaluated about the same trim conditions but at different flight times to gage the affects of the flight-time dependent mass properties.

$$\Delta\dot{U} = \frac{\Delta X}{m^*} - W^*\Delta Q \tag{3.26}$$

$$\Delta\dot{V} = \frac{\Delta Y}{m^*} - U^*\Delta R + W^*\Delta P \tag{3.27}$$

$$\Delta\dot{W} = \frac{\Delta Z}{m^*} - U^*\Delta Q \tag{3.28}$$

$$\Delta\dot{P} = \frac{\Delta L}{I_{xx}^*} \tag{3.29}$$

$$\Delta\dot{Q} = \frac{\Delta M}{I_{yy}^*} \tag{3.30}$$

$$\Delta\dot{R} = \frac{\Delta N}{I_{zz}^*} \tag{3.31}$$

Under the above steady flight condition assumptions, we can write the perturbation equations for the nonlinear Equations 3.20 - 3.25: where all starred, "*", variables indicate equilibrium values and where force and moment perturbations are

$$\begin{aligned}
\Delta X &= \Delta X_{aero} + \Delta F_{X_g} + \Delta T_X \\
\Delta Y &= \Delta Y_{aero} + \Delta F_{Y_g} \\
\Delta Z &= \Delta Z_{aero} + \Delta F_{Z_g} \\
\Delta L &= \Delta L_{aero} \\
\Delta M &= \Delta M_{aero} + \Delta M_g \\
\Delta N &= \Delta N_{aero} + \Delta N_g
\end{aligned} \tag{3.32}$$

However, under assumptions (2) and (iv), we have

$$\Delta T_X = \Delta M_g = \Delta N_g = 0$$

The gravitational force perturbations are as follows:

$$\begin{aligned}
\Delta F_{X_g} &= -mg\,\cos(\theta^*)\Delta\theta \\
\Delta F_{Y_g} &= mg\,\cos(\theta^*)\,\cos(\phi^*)\Delta\phi - mg\,\sin(\theta^*)\,\sin(\phi^*)\Delta\theta \\
\Delta F_{Z_g} &= -mg\,\cos(\theta^*)\,\sin(\phi^*)\Delta\phi - mg\,\sin(\theta^*)\,\cos(\phi^*)\Delta\theta
\end{aligned} \tag{3.33}$$

However, under flight condition assumption 3, i.e., $\phi^* = 0$, Equations 3.33 reduce to

$$\begin{aligned}
\Delta F_{X_g} &= -mg\,\cos(\theta^*)\Delta\theta \\
\Delta F_{Y_g} &= mg\,\cos(\theta^*)\Delta\phi \\
\Delta F_{Z_g} &= -mg\,\sin(\theta^*)\Delta\theta
\end{aligned} \tag{3.34}$$

The form of the aerodynamic forces and moments in Equations 3.16 (i.e., the stability derivative representation) gives us valuable information on their dependencies on the state and control variables. For example, lets consider the Taylor series expansion of the aerodynamic pitch moment, M:

$$\Delta M = Q_{dp}{}^* S_{ref} L_{ref} \Delta C_M + \rho^* V_b{}^* S_{ref} L_{ref} C_M{}^* \Delta V_b$$

NOTE: $C_M{}^*$ and the other trimmed aerodynamic moment coefficients are not necessarily zero because the missile's c.g. is not located at body fixed-frame (except at t = 0 because the body axis is fixed to the time-zero location of the c.g. and where we assume that all of the aerodynamic data is referenced from). However, under idealization (2), we will assume X_{CG} is zero, and thus, all trimmed *moment* aerodynamic coefficients are zero. This will be more apparent in Section 3.3, where we discuss the trim, or equilibrium, conditions of the missile.

From Equations 3.16, we can immediately see that

$$\Delta C_M = C_{M_{\delta_q}}{}^* \Delta\delta_q + C_{M_q}{}^*(L_{ref}/2V_b)\Delta Q + C_{M_\alpha}{}^* \Delta\alpha$$

which is of the form,

$$\Delta C_M = \left(\frac{\partial C_M}{\partial \delta_q}\right)^* \Delta\delta_q + \left(\frac{\partial C_M}{\partial (Q L_{ref}/2V_b)}\right)^* (L_{ref}/2V_b)\Delta Q + \left(\frac{\partial C_M}{\partial \alpha}\right)^* \Delta\alpha \quad (3.35)$$

In the work that follows, we will make use of the stability derivatives in the Taylor series expansions of C_X, C_Y, C_Z, C_L, C_M, and C_N. The stability derivatives, as discussed in Chapter 2, already tell us the important states and controls that they depend on and give their respective partial derivatives with respect to the states and controls.

For later work, we will need the Taylor series expansions of the aerodynamic variables α and β, thus, we will give them here (we will eventually write the linearized EOM with respect to the principal axis, which is sometimes used the for high speed missiles where inertial effects are important [53] and not the commonly used stability axis):

$$\Delta\alpha = -\frac{\sin(\alpha^*)}{V_b{}^*}\Delta U + \frac{\cos(\alpha^*)}{V_b{}^*}\Delta W \quad (3.36)$$

and under steady flight condition assumption (2), i.e. $V^* = 0$, we have

$$\Delta\beta = \left(\frac{1}{V_b{}^*}\right)\Delta V \quad (3.37)$$

Also we can write the pertubation of the resultant missile velocity (for $V^* = 0$), as:

$$\Delta\beta = \left(\frac{U^*}{V_b{}^*}\right)\Delta U + \left(\frac{W^*}{V_b{}^*}\right)\Delta W = (\alpha^*)\Delta U + \sin(\alpha^*)\Delta W \quad (3.38)$$

For notational conveniences, we will define the following compact forms of the partial derivatives of forces and moments:

Partial Derivatives of Forces:

$$X_u{}^* = \frac{1}{m^*}\left(\frac{\partial X}{\partial U}\right)^*$$

$$Z_w{}^* = \frac{1}{m^*}\left(\frac{\partial Z}{\partial W}\right)^*$$

and etc...

Partial Derivatives of Moments:

$$M_w{}^* = \frac{1}{I_{yy}{}^*}\left(\frac{\partial M}{\partial W}\right)^*$$

$$N_v{}^* = \frac{1}{I_{zz}{}^*}\left(\frac{\partial N}{\partial V}\right)^*$$

and etc...

By inspection of the right hand sides of Equations 3.16, i.e., the stability derivative representation of the aerodynamic forces and moments, and making use of Equations 3.36-3.38, we can write the following perturbation equations:

X-component of Translational Acceleration:

$$\Delta\dot{U} = X_u{}^*\Delta U + X_w{}^*\Delta W + X_q{}^*\Delta Q - (g^*\cos\theta^*)\Delta\theta \tag{3.39}$$

where

$$X_u{}^* = \left(\frac{Q_{dp}{}^* S_{ref}}{m^* V_b{}^*}\right) 2C_X{}^*\cos\alpha^*$$

$$X_w{}^* = \left(\frac{Q_{dp}{}^* S_{ref}}{m^* V_b{}^*}\right) 2C_X{}^*\sin\alpha^*$$

$$X_q{}^* = -W^*$$

Y-component of Translational Acceleration:

$$\Delta\dot{V} = Y_u{}^*\Delta U + Y_v{}^*\Delta V + Y_w{}^*\Delta W + Y_p{}^*\Delta P + Y_r{}^*\Delta R + Y_{\delta_r}{}^*\Delta\delta_r + (g^* cos\theta^*)\Delta\phi \quad (3.40)$$

where

$$Y_u{}^* = \left(\frac{Q_{dp}{}^* S_{ref}}{m^* V_b{}^*}\right) 2C_Y{}^* \cos\alpha^*$$
$$Y_v{}^* = \left(\frac{Q_{dp}{}^* S_{ref}}{m^* V_b{}^*}\right) C_{N_\beta}{}^*$$
$$Y_w{}^* = \left(\frac{Q_{dp}{}^* S_{ref}}{m^* V_b{}^*}\right) 2C_Y{}^* \sin\alpha^*$$
$$Y_p{}^* = W^*$$
$$Y_r{}^* = -U^*$$
$$Y_{\delta r}{}^* = \left(\frac{Q_{dp}{}^* S_{ref}}{m^* V_b{}^*}\right) C_{Y_{\delta_r}}{}^*$$

Z-component of Translational Acceleration:

$$\Delta\dot{W} = Z_u{}^*\Delta U + Z_w{}^*\Delta W + Z_q{}^*\Delta Q + Z_{\delta_q}{}^*\Delta\delta_q - (g^* \sin\theta^*)\Delta\theta \quad (3.41)$$

where

$$Z_u{}^* = Q_{dp}{}^* S_{ref}\left(\frac{2C_Z{}^* \cos\alpha^* - C_{N_\alpha}{}^* \sin\alpha^*}{m^* V_b{}^*}\right)$$
$$Z_w{}^* = Q_{dp}{}^* S_{ref}\left(\frac{2C_Z{}^* \sin\alpha^* + C_{N_\alpha}{}^* \cos\alpha^*}{m^* V_b{}^*}\right)$$
$$Z_q{}^* = U^*$$
$$Z_{\delta_q}{}^* = \left(\frac{Q_{dp}{}^* S_{ref}}{m^*}\right) C_{N_{\delta_q}}{}^*$$

X-component of Angular Acceleration:

$$\Delta \dot{P} = {L_p}^* \Delta P + {L_v}^* \Delta V + {L_{\delta_p}}^* \Delta \delta_p \tag{3.42}$$

where

$$\begin{aligned}
{L_p}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref} (L_{ref}/2{V_b}^*)}{{I_{xx}}^*} \right) {C_{L_p}}^* \\
{L_v}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref}}{{I_{xx}}^* {V_b}^*} \right) {C_{L_\beta}}^* \\
{L_{\delta_p}}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref}}{{I_{xx}}^*} \right) {C_{L_{\delta_p}}}^*
\end{aligned}$$

and where the fact that during trimmed flight (for $X_{CG} = 0$), ${C_L}^* = 0$ has been used in the stability derivatives of L_u and L_w, i.e. they are equal to zero and not included.

Y-component of Angular Acceleration:

$$\Delta \dot{Q} = {M_q}^* \Delta Q + {M_u}^* \Delta U + {M_w}^* \Delta W + {M_{\delta_q}}^* \Delta \delta_q \tag{3.43}$$

where

$$\begin{aligned}
{M_q}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref} (L_{ref}/2{V_b}^*)}{{I_{yy}}^*} \right) {C_{M_q}}^* \\
{M_u}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref}}{{I_{yy}}^* {V_b}^*} \right) {C_{M_\alpha}}^* \sin \alpha^* \\
{M_w}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref}}{{I_{yy}}^* {V_b}^*} \right) {C_{M_\alpha}}^* \cos \alpha^* \\
{M_{\delta_q}}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref}}{{I_{yy}}^*} \right) {C_{M_{\delta_q}}}^*
\end{aligned}$$

and where the fact that during trimmed flight (for $X_{CG} = 0$), ${C_M}^* = 0$ has been used in the stability derivatives of M_u and M_w, i.e. they are equal to zero and not included.

Z-component of Angular Acceleration:

$$\Delta\dot{R} = {N_r}^*\Delta R + {N_v}^*\Delta V + {N_{\delta_r}}^*\Delta\delta_r \tag{3.44}$$

where

$$\begin{aligned}
{N_r}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref} (L_{ref}/2{V_b}^*)}{{I_{zz}}^*}\right) {C_{N_r}}^* \\
{N_v}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref}}{{I_{zz}}^* {V_b}^*}\right) {C_{N_\beta}}^* \\
{N_{\delta_r}}^* &= \left(\frac{{Q_{dp}}^* S_{ref} L_{ref}}{{I_{zz}}^*}\right) {C_{N_{\delta_r}}}^*
\end{aligned}$$

and where the fact that during trimmed flight (for $X_{CG} = 0$), ${C_N}^* = 0$ has been used in the stability derivatives of N_u and N_w, i.e. they are equal to zero and not included.

Equations 3.39 - 3.44 can be put into the following compact state equation form:

$$\begin{pmatrix} \Delta\dot{U} \\ \Delta\dot{V} \\ \Delta\dot{W} \\ \Delta\dot{P} \\ \Delta\dot{Q} \\ \Delta\dot{R} \\ \Delta\dot{\phi} \\ \Delta\dot{\theta} \end{pmatrix} = \begin{pmatrix} {X_u}^* & 0 & {X_w}^* & 0 & {X_q}^* & 0 & 0 & -g\cos\theta^* \\ {Y_u}^* & {Y_v}^* & {Y_w}^* & {Y_p}^* & 0 & {Y_r}^* & g\cos\theta^* & 0 \\ {Z_u}^* & 0 & {Z_w}^* & 0 & {Z_q}^* & 0 & 0 & -g\sin\theta^* \\ 0 & {L_v}^* & 0 & {L_p}^* & 0 & 0 & 0 & 0 \\ {M_u}^* & 0 & {M_w}^* & 0 & {M_q}^* & 0 & 0 & 0 \\ 0 & {N_v}^* & 0 & 0 & 0 & {N_r}^* & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta U \\ \Delta V \\ \Delta W \\ \Delta P \\ \Delta Q \\ \Delta R \\ \Delta\phi \\ \Delta\theta \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & {Y_{\delta_r}}^* \\ 0 & {Z_{\delta_q}}^* & 0 \\ {L_{\delta_p}}^* & 0 & 0 \\ 0 & {M_{\delta_q}}^* & 0 \\ 0 & 0 & {N_{\delta_r}}^* \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta_p \\ \delta_q \\ \delta_r \end{pmatrix} \tag{3.45}$$

where in the Equations 3.45, we have used for small perturbations that the Euler angles (bank angle and attitude), only retaining 1st order terms, can be approximated as:

$$\Delta\dot{\phi} = \Delta P \tag{3.46}$$

$$\Delta\dot{\theta} = \Delta Q \tag{3.47}$$

Finally Equations 3.45 yield the linear state-space equations of the BTT missile (under the assumptions in the section).

3.3 Calculation of Equilibrium Points

From Section 3.2, (x^*, u^*) is an equilibrium point of Equation 3.1 for all t ≥ 0.

$$f(t, x^*, u^*) = 0 \tag{3.48}$$

Thus, equations 3.9-3.14, for the assumptions of zero angular and translational accelerations and the assumptions about ignoring the time rates of change of the mass, mass moments of inertia, and c.g. location yield

$$\begin{aligned}
X^* &= {F_{X_{aero}}}^* + {F_{X_g}}^* + {T_X}^* = 0 \\
Y^* &= {F_{Y_{aero}}}^* + {F_{Y_g}}^* = 0 \\
Z^* &= {F_{Z_{aero}}}^* + {F_{Z_g}}^* = 0 \\
L^* &= {L_{aero}}^* = 0 \\
M^* &= {M_{aero}}^* + {M_g}^* = 0 \\
N^* &= {N_{aero}}^* + {N_g}^* = 0
\end{aligned} \tag{3.49}$$

For now, we will assume X_{CG} is not zero only to see what effect our earlier idealization that $X_{CG} = 0$ has on our linear EOM. Substituting for the aerodynamic and gravitational forces and moments into Equation 3.49 using Equations 3.16 and 3.17 and rearranging yields and

$$
\begin{aligned}
X &: Q_{dp} S_{ref} C_X = mg\ \sin\theta - T_x \\
Y &: Q_{dp} S_{ref} C_Y = -mg\ \cos\theta \sin\phi \\
Z &: Q_{dp} S_{ref} C_Z = -mg\ \cos\theta \cos\phi \\
L &: Q_{dp} S_{ref} L_{ref} C_L = 0 \\
M &: Q_{dp} S_{ref} L_{ref} C_M = X_{CG}\, mg\ \cos\theta \cos\phi \\
N &: Q_{dp} S_{ref} L_{ref} C_N = -X_{CG}\, mg\ \cos\theta \sin\phi
\end{aligned}
\tag{3.50}
$$

If we assume some level of thrust T_x, we can solve for the trimmed aerodynamic coefficients C_X, C_Y, C_Z, C_L, C_M, and C_N. The values of the aerodynamic coefficients in trimmed flight are given as follows:

$$
\begin{aligned}
{C_X}^* &= \frac{mg\ \sin\theta^* - {T_x}^*}{{Q_{dp}}^* S_{ref}} \\
{C_Y}^* &= \frac{-mg\ \cos\theta^* \sin\phi^*}{{Q_{dp}}^* S_{ref}} \\
{C_Z}^* &= \frac{-mg\ \cos\theta^* \cos\phi^*}{{Q_{dp}}^* S_{ref}} \\
{C_L}^* &= 0 \\
{C_M}^* &= \frac{X_{CG} mg\ \cos\theta^* \cos\phi^*}{{Q_{dp}}^* S_{ref} L_{ref}} \\
{C_N}^* &= \frac{-X_{CG} mg\ \cos\theta^* \sin\phi^*}{{Q_{dp}}^* S_{ref} L_{ref}}
\end{aligned}
\tag{3.51}
$$

From idealization (2), i.e., $X_{CG} = 0$, ${C_M}^* = {C_N}^* = 0$. In addition, ${C_Y}^* = 0$ under the previous idealization that the missiles bank angle is zero ($\phi = 0$). These idealizations were accounted for in all linear equations presented in the previous section. From the above Equations 3.51, we can see that when we assume that the equilibrium value of the c.g. location is zero that the trimmed values of C_M and C_N are zero. Also from equations 3.51, we can see even if X_{CG} is non-zero but relatively "small" that at "high" missile velocities and low missile altitudes (this gives large Qdp) that the trimmed values of C_M and C_N are "small" and that the idealization that $X_{CG} = 0$ is valid. However, this might not be a valid idealization for a "slow" moving aircraft, such as one approaching for landing, or for very "large" c.g. displacements.

In Section 2.4, the BTT missile stability derivatives were presented and Table 2.3 summarized their dependence on other variables. Using Table 2.3 as reference, Equations 3.51 are rewritten to emphasize their dependence on the stability derivatives (also X_{CG} is assumed to be zero, as discussed previously):

The m-file "btt_linr.m" uses the above equations, given a user specified trim angle of attack, α^*, and altitude, to iterate for the corresponding Mach number and actuator deflection δ_q.

$$
\begin{aligned}
C_D({\delta_q}^*, M^*, \alpha^*) + C_{DT}(M^*) &= \frac{mg\ \sin\theta^* - {T_x}^*}{{Q_{dp}}^* S_{ref}} \\
C_{Y_\beta}(M^*, \alpha^*)\beta^* + C_{Y_{\delta_r}}(M^*, \beta^*){\delta_r}^* &= \frac{-mg\ \cos\theta^* \sin\phi^*}{{Q_{dp}}^* S_{ref}} \\
C_{N_\alpha}(M^*)\alpha^* + C_{N_{\delta_q}}(M^*, \alpha^*){\delta_q}^* &= \frac{-mg\ \cos\theta^* \cos\phi^*}{{Q_{dp}}^* S_{ref}} \\
C_{L_{\delta_p}}(M^*, \alpha^*){\delta_p}^* + C_{L_p}(M^*, \alpha^*)(L_{ref}/2V_b)P^* + C_{L_\beta}(M^*, \alpha^*)\beta^* &= 0 \\
C_{M_{\delta_q}}(M^*, \alpha^*){\delta_q}^* + C_{M_q}(M^*, \alpha^*)(L_{ref}/2V_b)Q^* + C_{M_\alpha}(M^*, \alpha^*)\alpha^* &= 0 \\
C_{N_{\delta_r}}(M^*, \beta^*){\delta_r}^* + C_{N_r}(M^*, \beta^*)(L_{ref}/2V_b)R^* + C_{N_\beta}(M^*, \alpha^*)\beta^* &= 0
\end{aligned}
\tag{3.52}
$$

Under Steady state flight assumptions (i) and (ii), equations 3.52 can be rewritten as follows:

$$
\begin{aligned}
C_D({\delta_q}^*, M^*, \alpha^*) + C_{DT}(M^*) &= \frac{mg\ \sin\theta^* - {T_x}^*}{{Q_{dp}}^* S_{ref}} \\
C_{Y_{\delta_r}}(M^*, \beta^*){\delta_r}^* &= \frac{-mg\ \cos\theta^* sin\phi^*}{{Q_{dp}}^* S_{ref}} = 0, (\because \phi = 0) \\
C_{N_\alpha}(M^*)\alpha^* + C_{N_{\delta_q}}(M^*, \alpha^*){\delta_q}^* &= \frac{-mg\ \cos\theta^*}{{Q_{dp}}^* S_{ref}} \\
{\delta_p}^* &= 0 \\
{\delta_q}^* &= \frac{-C_{M_\alpha}(M^*, \alpha^*)}{C_{M_{\delta_q}}(M^*, \alpha^*)}\alpha^*, (Solved\, for\, \delta_q) \\
{\delta_r}^* &= 0
\end{aligned}
\tag{3.53}
$$

The m-file “btt_linr.m” uses the above equations, given a user specified trim angle of attack, α^*, and altitude, to iterate for the corresponding Mach number and actuator deflection δ_q.

3.4 Scaled Linear BTT Missile State-Space System

In this section, the linear EOM derived in the previous section are dimensionally scaled. A dimensionally scaled state-space system is desirable for the following reasons:

1. Modal analysis of the state-space system to determine the systems natural tendencies is easier to interpret when all of the system equations have the same units. This makes comparisons between translational and rotational modes of the same (scaled) size.

2. Multivariable control theory such as the H_∞ design method essentially “shape” the systems transfer function matrix (TFM) singular value bode magnitude plots based upon some user supplied weightings on performance and/or robustness. It is well known that singular values are unit sensitive and thus it is

desirable that we have singular value loop shapes that have the same units such that we are comparing "apples to apples" and not "apples to oranges".

3. From a properly done dimensional analysis (i.e., a properly scaled system), a simple observation of the systems terms is all that is necessary to determine the relative importance of the dependent variables in the EOM. This is an invaluable tool during model reduction.

Non-dimensional State Equations:

In our dimensional analysis we will define the following non-dimensional quantities:

$$\begin{aligned}
\hat{u} &\equiv \frac{\Delta U}{{V_b}^*} \\
\hat{v} &\equiv \frac{\Delta V}{{V_b}^*} \\
\hat{w} &\equiv \frac{\Delta W}{{V_b}^*} \\
\hat{p} &\equiv \hat{t}\Delta P \\
\hat{q} &\equiv \hat{t}\Delta Q \\
\hat{r} &\equiv \hat{t}\Delta R
\end{aligned} \tag{3.54}$$

where

$$\hat{t} \equiv \frac{m{V_b}^*}{{Q_{dp}}^* S_{ref}}, (sec) \tag{3.55}$$

and where we define the non-dimensional aerodynamic time, τ, as

$$\tau \equiv \frac{t}{\hat{t}} \tag{3.56}$$

From Equation 3.56, we can see that the differentiation operator now becomes

$$\frac{d()}{dt} = \frac{1}{\hat{t}}\frac{d()}{d\tau} = \frac{{Q_{dp}}^* S_{ref}}{m{V_b}^*}\frac{d()}{d\tau} \tag{3.57}$$

dt = $\hat{t}$ dτ.

Substituting for $\Delta U, \Delta V, \Delta W, \Delta P, \Delta Q$, and ΔR in equations 3.39 - 3.44 using equations 3.54 and also substituting for the differentiation operator using equation 3.57 and then dividing through the resulting equations by ${V_b}^*$ yields the following non-dimensional equations of motion:

$$\dot{\hat{u}} = {x_u}^*\hat{u} + {x_w}^*\hat{w} + {x_q}^*\hat{q} - \hat{g}\cos\theta^*\Delta\theta \tag{3.58}$$

$$\dot{\hat{v}} = {y_u}^*\hat{u} + {y_v}^*\hat{v} + {y_w}^*\hat{w} + {y_p}^*\hat{p} + {y_r}^*\hat{r} + \hat{g}\cos\theta^*\Delta\phi \tag{3.59}$$

$$\dot{\hat{w}} = {z_u}^*\hat{u} + {z_w}^*\hat{w} + {z_q}^*\hat{q} + {z_{\delta_q}}^*\Delta\delta_q - \hat{g}\sin\theta^*\Delta\theta \tag{3.60}$$

$$\dot{\hat{p}} = {l_p}^*\hat{p} + {l_v}^*\hat{v} + {l_{\delta_p}}^*\Delta\delta_p \tag{3.61}$$

$$\dot{\hat{q}} = {m_u}^*\hat{u} + {m_w}^*\hat{w} + {m_q}^*\hat{q} + {m_{\delta_q}}^*\Delta\delta_q \tag{3.62}$$

$$\dot{\hat{r}} = {n_v}^*\hat{v} + {n_r}^*\hat{r} + {n_{\delta_r}}^*\Delta\delta_r \tag{3.63}$$

where

$$\hat{g} \equiv \frac{m^* g}{{Q_{dp}}^* S_{ref}} \tag{3.64}$$

The lower case stability derivatives are dimensionless are related to the previously defined stability derivatives as follows:

$$
\begin{aligned}
{x_u}^* &= \left(\frac{m^*{V_b}^*}{{Q_{dp}}^*S_{ref}}\right){X_u}^* = 2{C_X}^*\cos\alpha^* \\
{x_w}^* &= \left(\frac{m^*{V_b}^*}{{Q_{dp}}^*S_{ref}}\right){X_w}^* = 2{C_X}^*\sin\alpha^* \\
{x_q}^* &= \left(\frac{1}{{V_b}^*}\right){X_q}^* = -\left(\frac{W^*}{{V_b}^*}\right) = \sin\alpha^* \\
{y_u}^* &= \left(\frac{m^*{V_b}^*}{{Q_{dp}}^*S_{ref}}\right){Y_u}^* = 2{C_Y}^*\cos\alpha^* \\
{y_v}^* &= \left(\frac{m^*{V_b}^*}{{Q_{dp}}^*S_{ref}}\right){Y_v}^* = {C_{N_\beta}}^*
\end{aligned}
$$

$$y_w{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right) Y_w{}^* = 2 C_Y{}^* \sin\alpha^*$$

$$y_p{}^* = \left(\frac{1}{V_b{}^*}\right) Y_p{}^* = \left(\frac{W^*}{V_b{}^*}\right) = \sin\alpha^*$$

$$y_r{}^* = \left(\frac{1}{V_b{}^*}\right) Y_r{}^* = -\left(\frac{U^*}{V_b{}^*}\right) = -\cos\alpha^*$$

$$y_{\delta_r}{}^* = \left(\frac{m^*}{Q_{dp}{}^* S_{ref}}\right) Y_{\delta_r}{}^* = C_{Y_{\delta_r}}{}^*$$

$$z_u{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right) Z_u{}^* = 2 C_Z{}^* \cos\alpha^* - C_{N_\alpha}{}^* \sin\alpha^*$$

$$z_w{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right) Z_w{}^* = 2 C_Z{}^* \sin\alpha^* + C_{N_\alpha}{}^* \cos\alpha^*$$

$$z_q{}^* = \left(\frac{1}{V_b{}^*}\right) Z_q{}^* = \left(\frac{U^*}{V_b{}^*}\right) = \cos\alpha^*$$

$$z_{\delta_q}{}^* = \left(\frac{m^*}{Q_{dp}{}^* S_{ref}}\right) Z_{\delta_q}{}^* = C_{N_{\delta_q}}{}^*$$

$$l_p{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right) L_p{}^* = \left(\frac{\frac{1}{2} m^* L_{ref}{}^2}{I_{xx}{}^*}\right) C_{L_p}{}^*$$

$$l_v{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right)^2 V_b{}^* L_v{}^* = \left(\frac{2 m^{*2} L_{ref}}{\rho^* S_{ref} I_{xx}{}^*}\right) C_{L_\beta}{}^*$$

$$l_{\delta_p}{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right)^2 L_{\delta_p}{}^* = \left(\frac{2 m^{*2} L_{ref}}{\rho^* S_{ref} I_{xx}{}^*}\right) C_{L_{\delta_p}}{}^*$$

$$m_u{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right)^2 V_b{}^* M_u{}^* = -\left(\frac{2 m^{*2} L_{ref}}{\rho^* S_{ref} I_{yy}{}^*}\right) C_{M_\alpha}{}^* \sin\alpha^*$$

$$m_w{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right)^2 V_b{}^* M_w{}^* = \left(\frac{2 m^{*2} L_{ref}}{\rho^* S_{ref} I_{yy}{}^*}\right) C_{M_\alpha}{}^* \cos\alpha^*$$

$$m_q{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right) M_q{}^* = \left(\frac{\frac{1}{2} m^* L_{ref}{}^2}{I_{yy}{}^*}\right) C_{M_q}{}^*$$

$$m_{\delta_q}{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right)^2 M_{\delta_q}{}^* = \left(\frac{2 m^{*2} L_{ref}}{\rho^* S_{ref} I_{yy}{}^*}\right) C_{M_{\delta_q}}{}^*$$

$$n_v{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right)^2 V_b{}^* N_v{}^* = \left(\frac{2 m^{*2} L_{ref}}{\rho^* S_{ref} I_{zz}{}^*}\right) C_{N_\beta}{}^*$$

$$n_r{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right) N_r{}^* = \left(\frac{\frac{1}{2} m^* L_{ref}{}^2}{I_{zz}{}^*}\right) C_{N_r}{}^*$$

$$n_{\delta_r}{}^* = \left(\frac{m^* V_b{}^*}{Q_{dp}{}^* S_{ref}}\right)^2 N_{\delta_r}{}^* = \left(\frac{2 m^{*2} L_{ref}}{\rho^* S_{ref} I_{zz}{}^*}\right) C_{N_{\delta_r}}{}^*$$

Using Equations 3.55 - 3.64 and the above defined non-dimensional stability derivatives, we can write the state equations in the following compact form:

$$\begin{pmatrix} \dot{\hat{u}} \\ \dot{\hat{v}} \\ \dot{\hat{w}} \\ \dot{\hat{p}} \\ \dot{\hat{q}} \\ \dot{\hat{r}} \\ \Delta\dot{\phi} \\ \Delta\dot{\theta} \end{pmatrix} = \begin{pmatrix} x_u{}^* & 0 & x_w{}^* & 0 & x_q{}^* & 0 & 0 & -\hat{g}\cos\theta^* \\ y_u{}^* & y_v{}^* & y_w{}^* & y_p{}^* & 0 & y_r{}^* & \hat{g}\cos\theta^* & 0 \\ z_u{}^* & 0 & z_w{}^* & 0 & z_q{}^* & 0 & 0 & -\hat{g}\sin\theta^* \\ 0 & l_v{}^* & 0 & l_p{}^* & 0 & 0 & 0 & 0 \\ m_u{}^* & 0 & m_w{}^* & 0 & m_q{}^* & 0 & 0 & 0 \\ 0 & n_v{}^* & 0 & 0 & 0 & n_r{}^* & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{u} \\ \hat{v} \\ \hat{w} \\ \hat{p} \\ \hat{q} \\ \hat{r} \\ \Delta\phi \\ \Delta\theta \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & y_{\delta_r}{}^* \\ 0 & z_{\delta_q}{}^* & 0 \\ l_{\delta_p}{}^* & 0 & 0 \\ 0 & m_{\delta_q}{}^* & 0 \\ 0 & 0 & n_{\delta_r}{}^* \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta_p \\ \delta_q \\ \delta_r \end{pmatrix} \tag{3.65}$$

where in Equation 3.65, we have made use of the fact that

$$\frac{1}{\hat{t}} \frac{d(\Delta\phi)}{d\tau} = \frac{1}{\hat{t}} \hat{p}$$

or simply

$$\frac{d(\Delta\phi)}{d\tau} = \hat{p}$$

and similarly,

$$\frac{d(\Delta\theta)}{d\tau} = \hat{q}$$

Note that $\hat{v}$ is the sideslip angle, $\Delta\beta$, under the assumption the equilibrium value of V is zero. However, $\hat{w}$ is only approximately equal to $\Delta\alpha$ for "small" equilibrium or reference values of α since we are using the principal body axis and not the stability axis for our linearized state-space system. Now that we have missile dynamics represented by its mathematical model, the following two questions is of particular interest to us.

1. How does the missile plant change when it travels at different velocities?

2. How does the missile plant change when it ascends up or descends down?

The following plots ranging from Figure 3.1 - 3.15 show how various missile plant outputs vary with respect to aileron, elevator and rudder inputs while the altitude is varied. The following plots ranging from Figure 3.16 - 3.30 show how various missile plant outputs vary with respect to aileron, elevator and rudder inputs while the mach is varied.

Missile I/P-O/P Transfer Function Frequency Responses - Altitude Varying

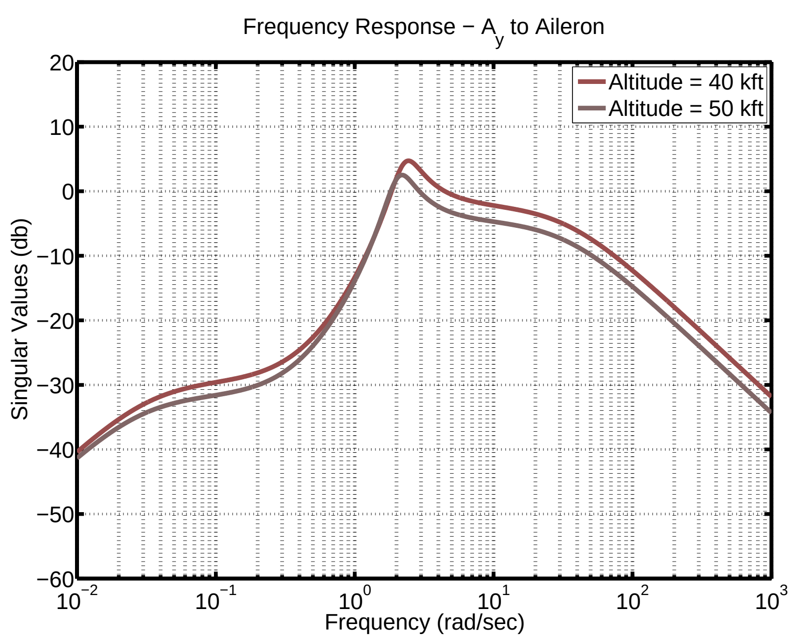

Figure 3.1: Frequency Response - A_y vs Aileron - Altitude Varying

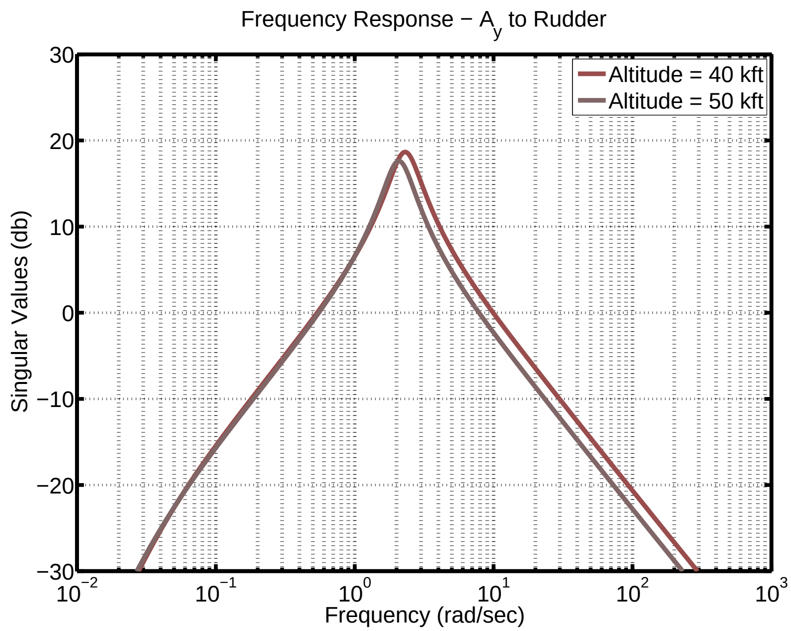

Figure 3.2: Frequency Response - A_y vs Rudder - Altitude Varying

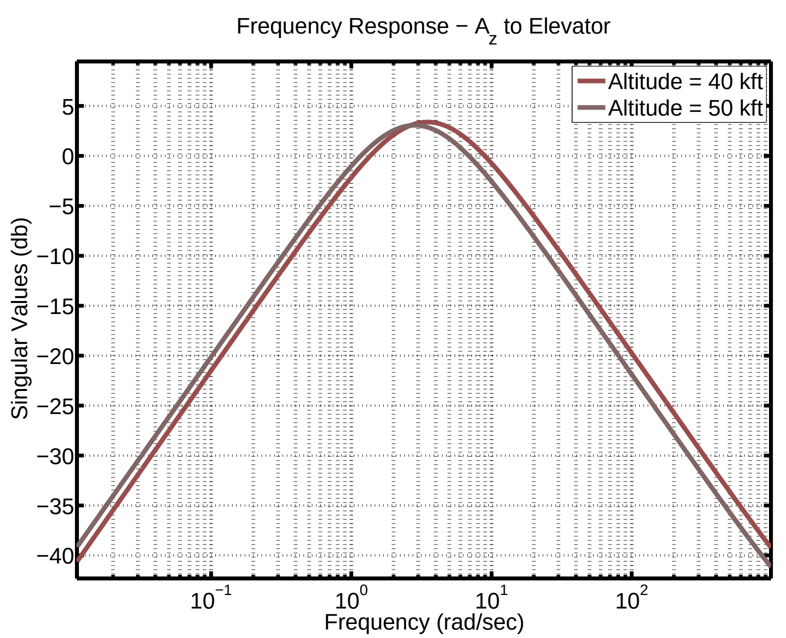

Figure 3.3: Frequency Response - A_z vs Elevator - Altitude Varying

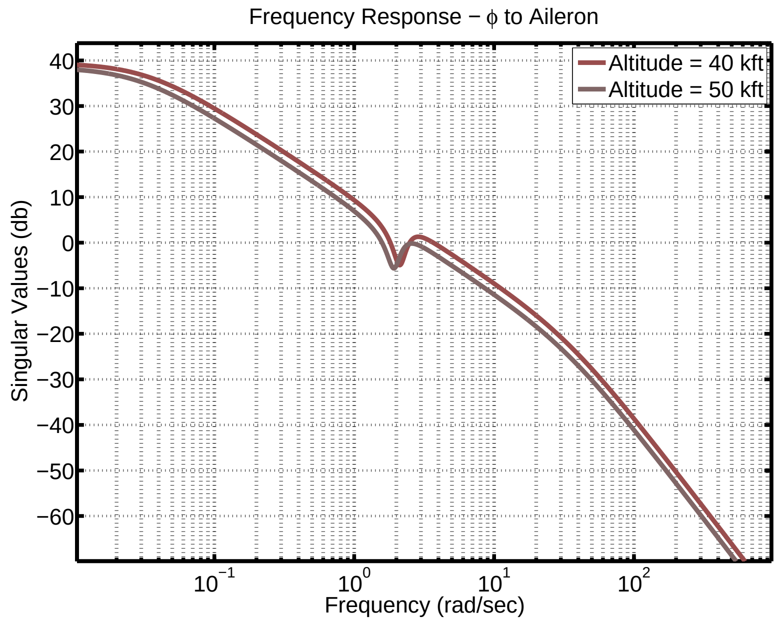

Figure 3.4: Frequency Response - ϕ vs Aileron - Altitude Varying

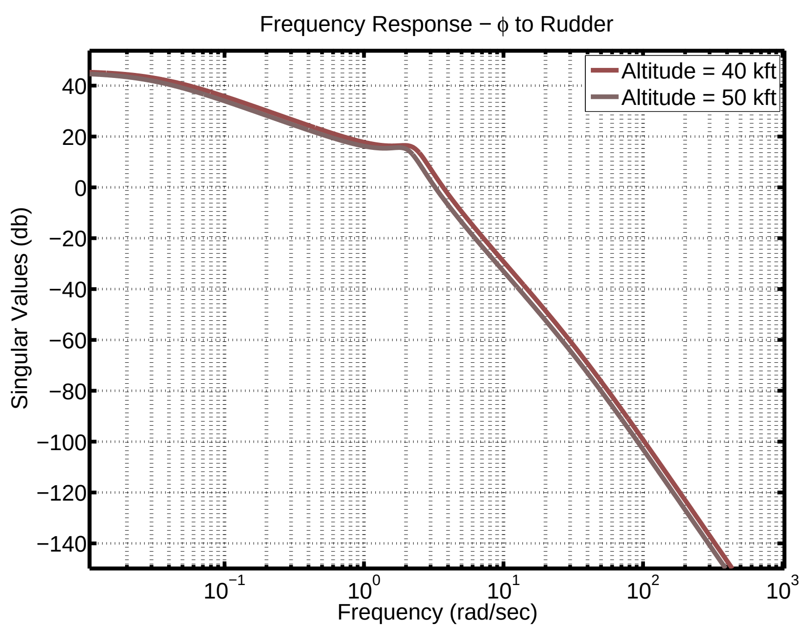

Figure 3.5: Frequency Response - ϕ vs Rudder - Altitude Varying

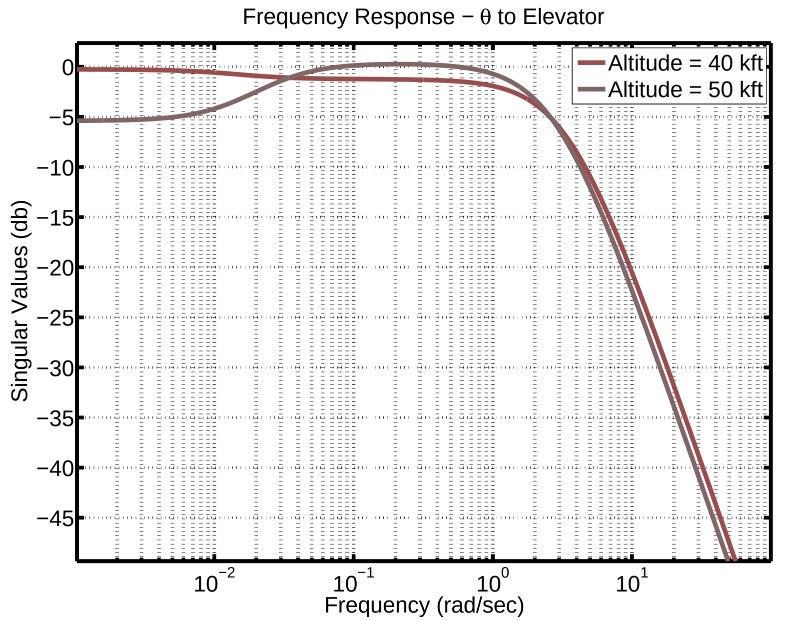

Figure 3.6: Frequency Response - θ vs Elevator - Altitude Varying

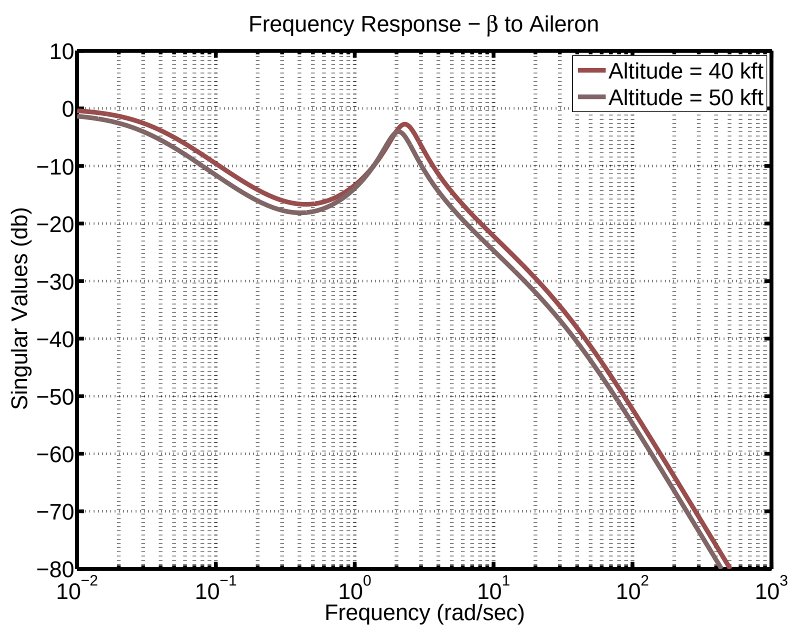

Figure 3.7: Frequency Response - β vs Aileron - Altitude Varying

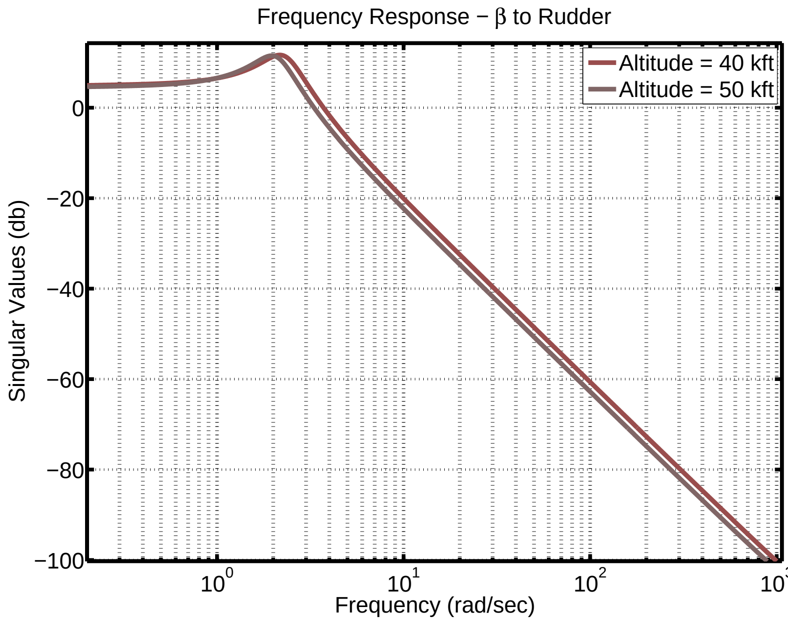

Figure 3.8: Frequency Response - β vs Rudder - Altitude Varying

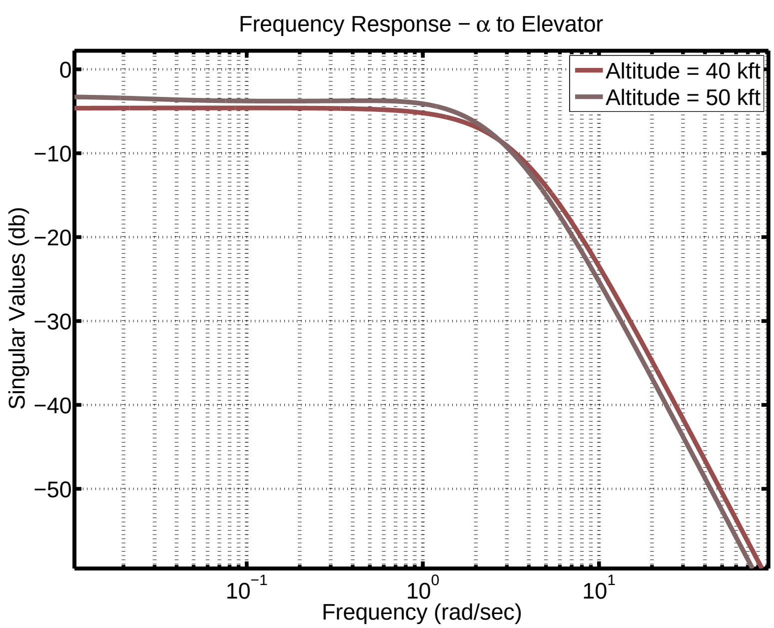

Figure 3.9: Frequency Response - α vs Elevator - Altitude Varying

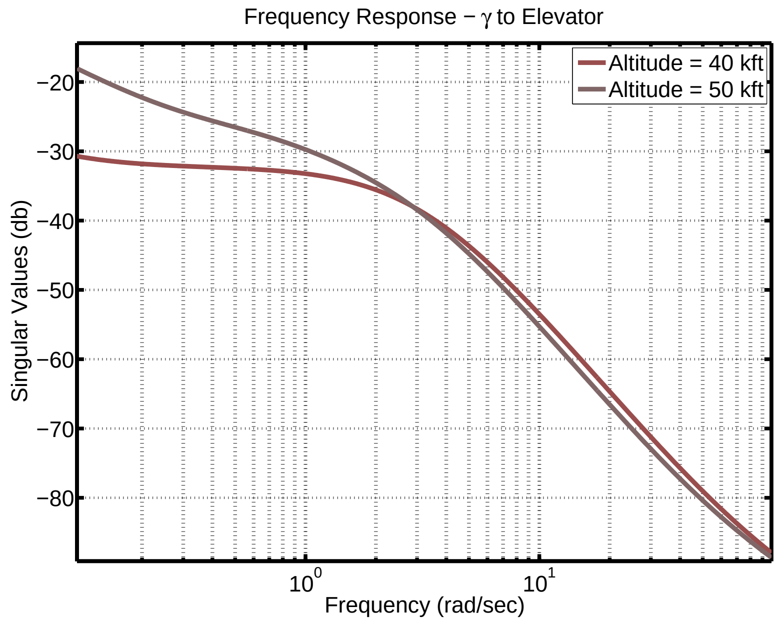

Figure 3.10: Frequency Response - γ vs Elevator - Altitude Varying

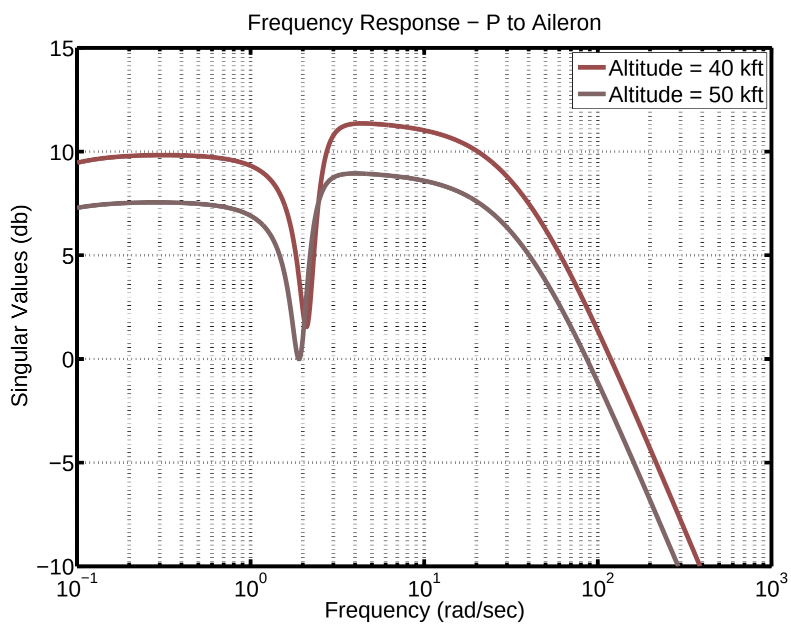

Figure 3.11: Frequency Response - P vs Aileron - Altitude Varying

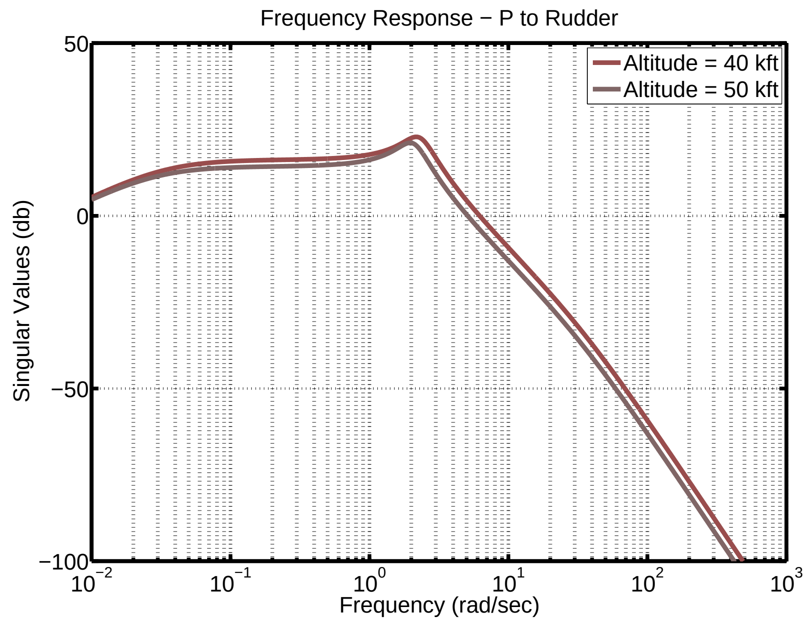

Figure 3.12: Frequency Response - P vs Rudder - Altitude Varying

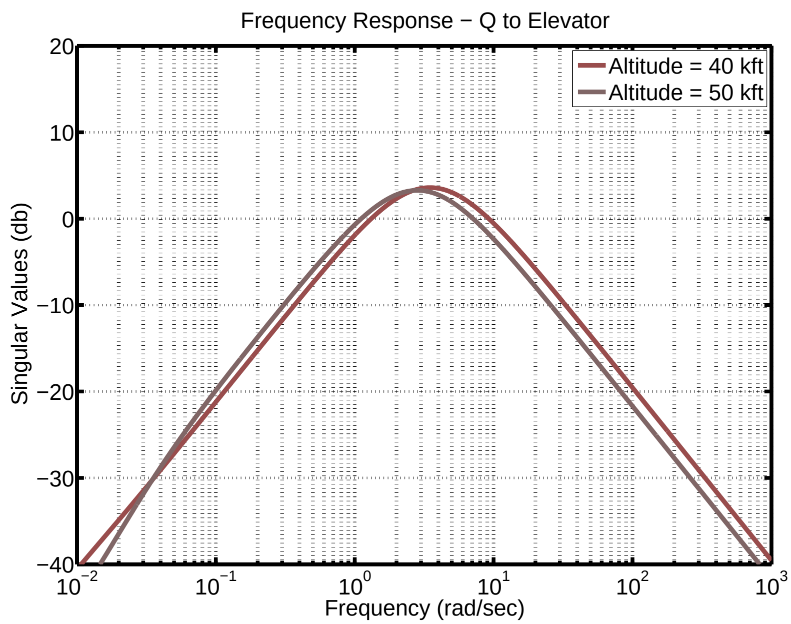

Figure 3.13: Frequency Response - Q vs Elevator - Altitude Varying

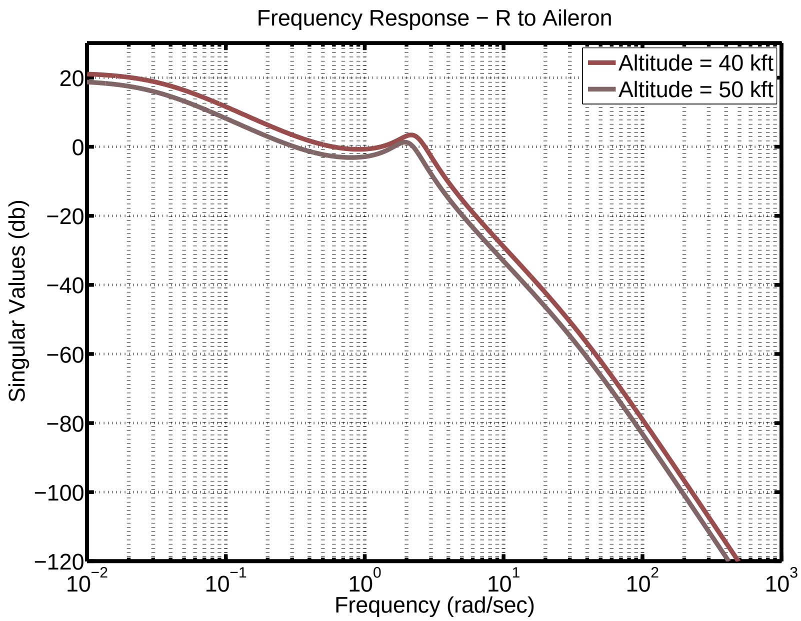

Figure 3.14: Frequency Response - R vs Aileron - Altitude Varying

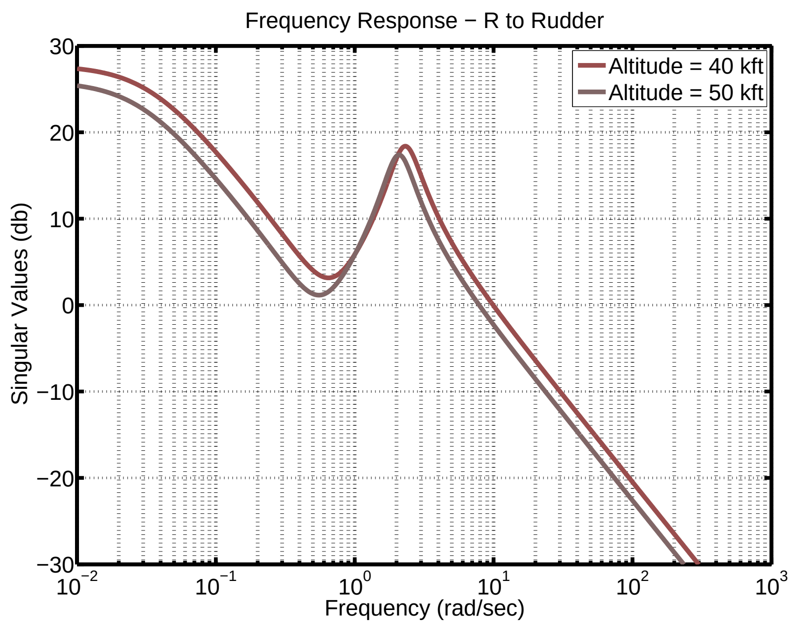

Figure 3.15: Frequency Response - R vs Rudder - Altitude Varying

Missile I/P-O/P Transfer Function Frequency Responses - Mach Varying

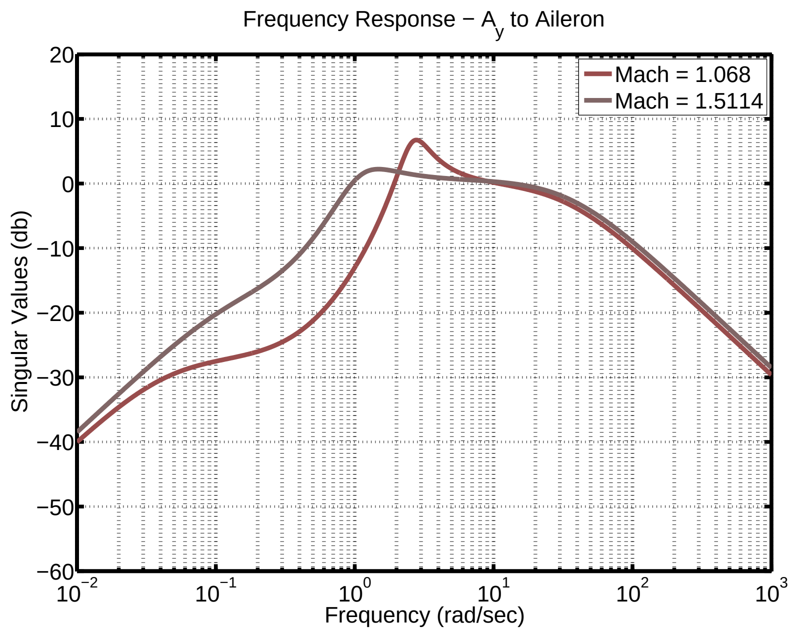

Figure 3.16: Frequency Response - A_y vs Aileron - Mach Varying

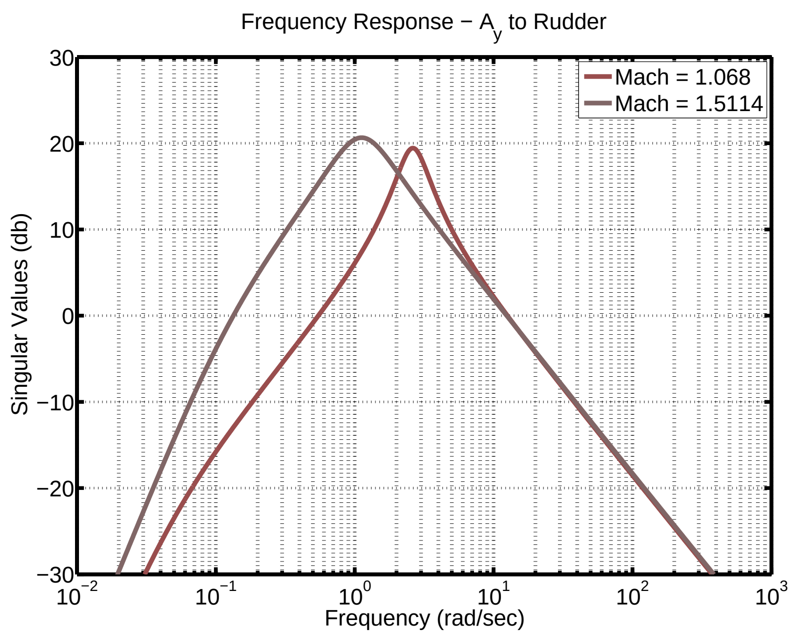

Figure 3.17: Frequency Response - A_y vs Rudder - Mach Varying

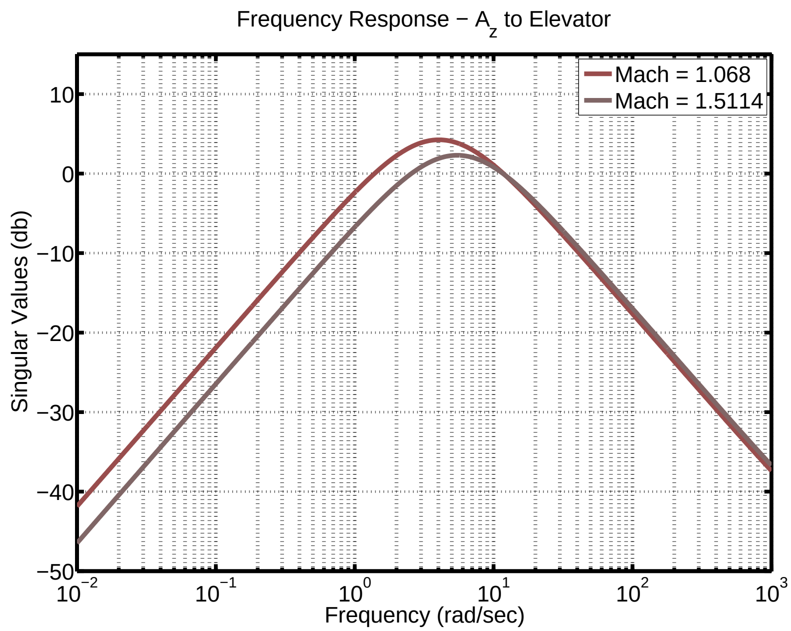

Figure 3.18: Frequency Response - A_z vs Elevator - Mach Varying

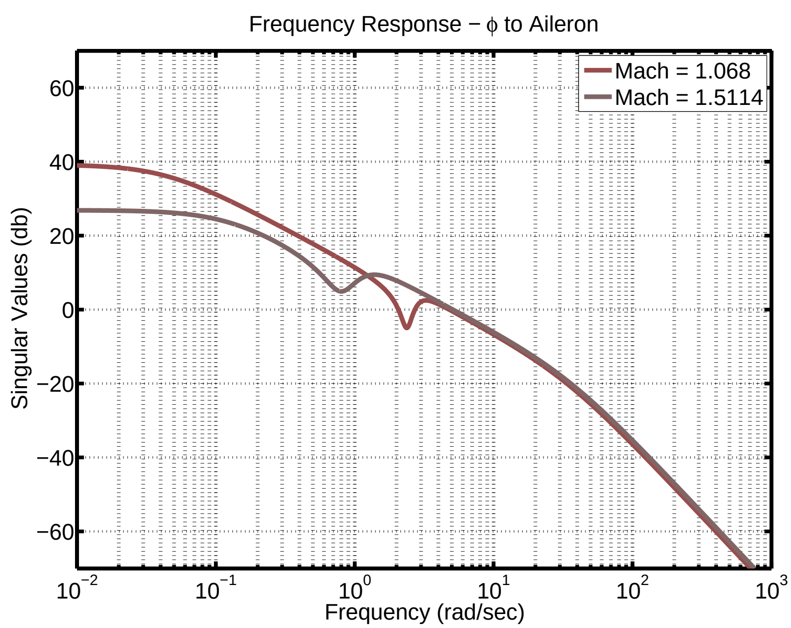

Figure 3.19: Frequency Response - ϕ vs Aileron - Mach Varying

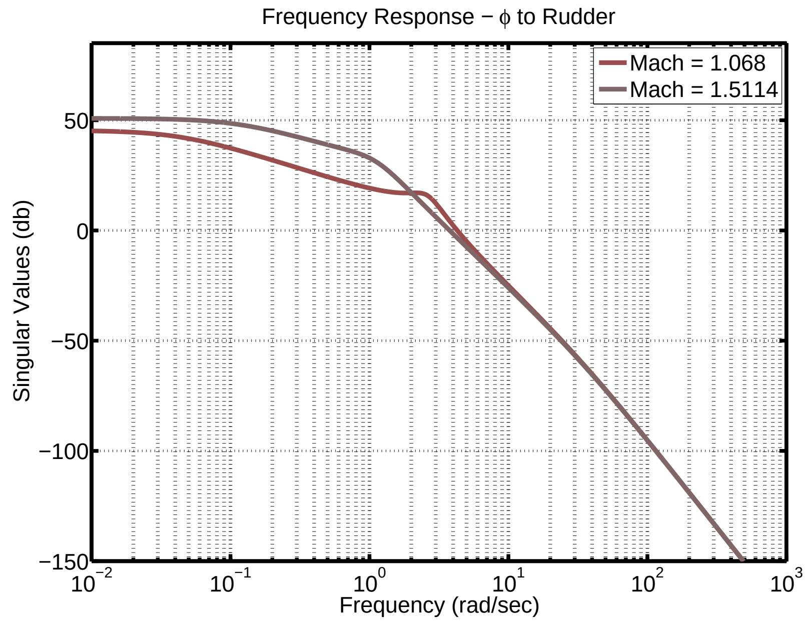

Figure 3.20: Frequency Response - ϕ vs Rudder - Mach Varying

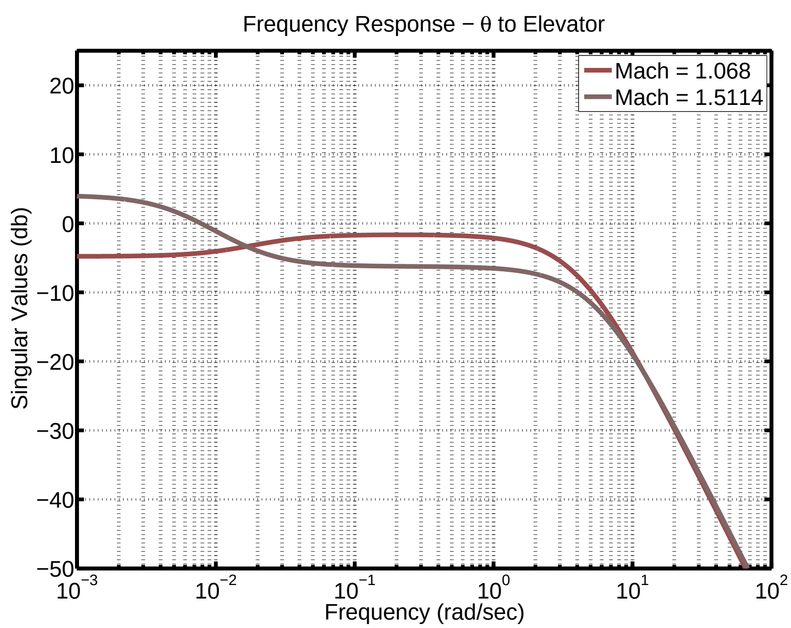

Figure 3.21: Frequency Response - θ vs Elevator - Mach Varying

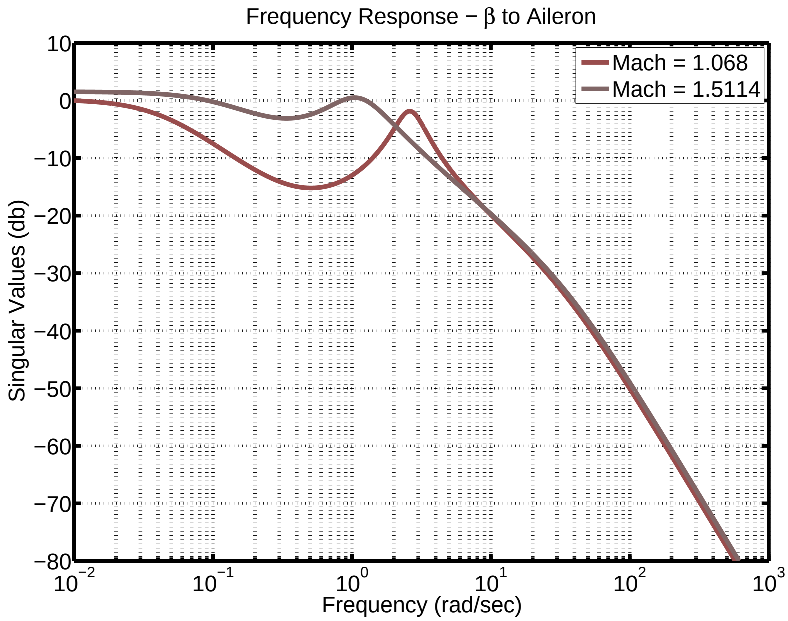

Figure 3.22: Frequency Response - β vs Aileron - Mach Varying

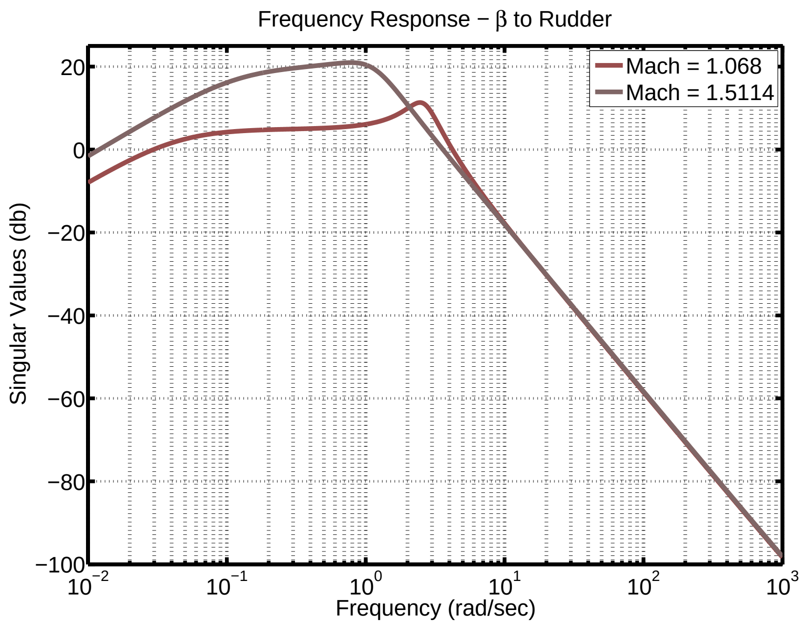

Figure 3.23: Frequency Response - β vs Rudder - Mach Varying

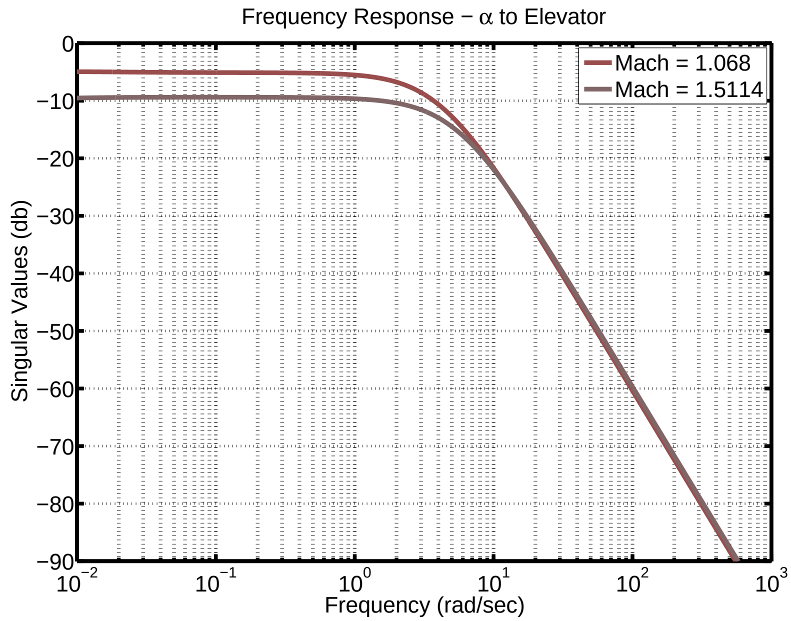

Figure 3.24: Frequency Response - α vs Elevator - Mach Varying

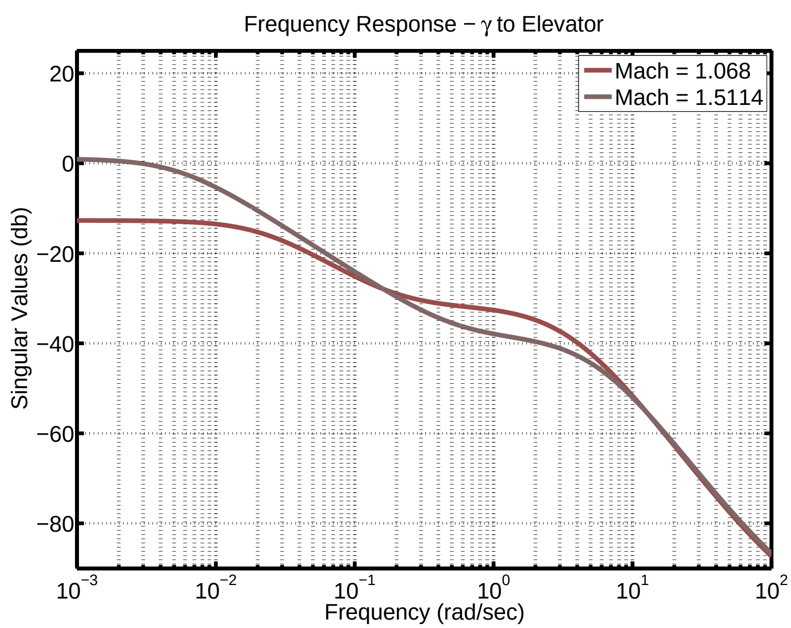

Figure 3.25: Frequency Response - γ vs Elevator - Mach Varying

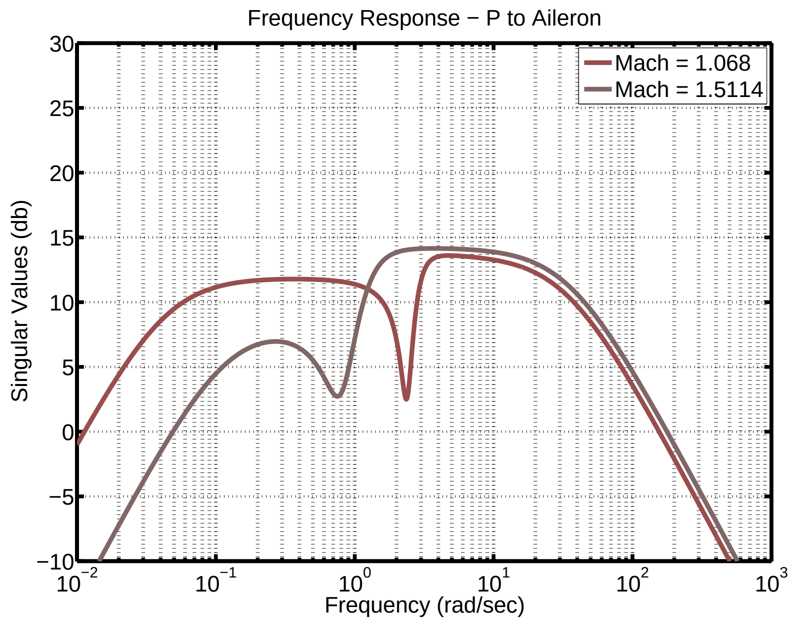

Figure 3.26: Frequency Response - P vs Aileron - Mach Varying

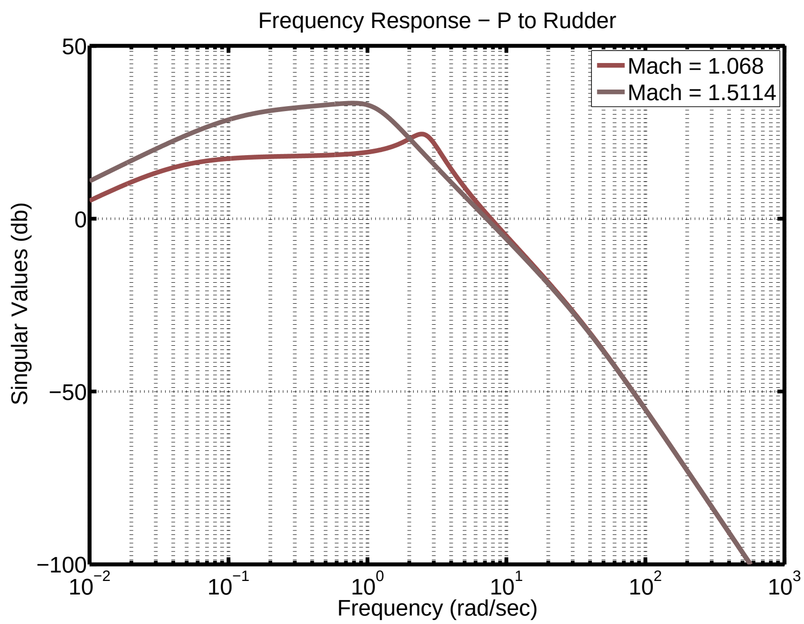

Figure 3.27: Frequency Response - P vs Rudder - Mach Varying

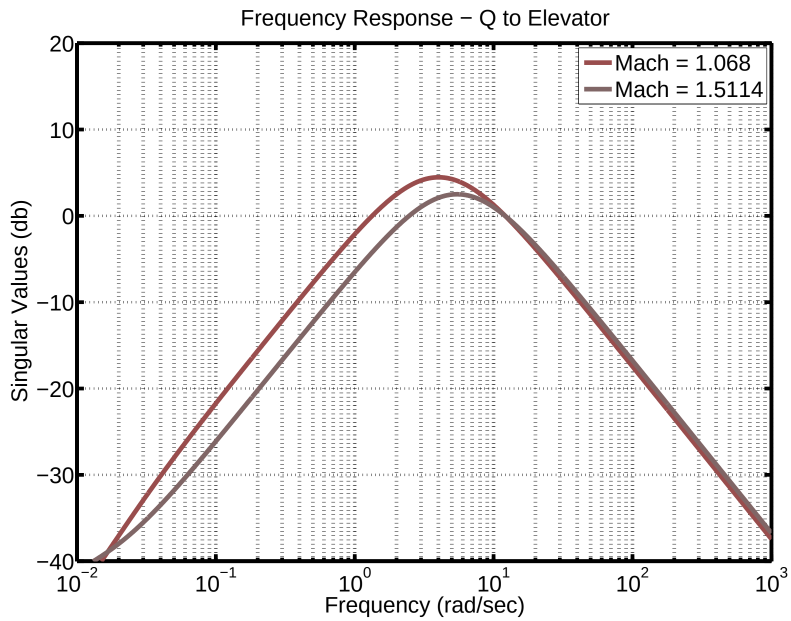

Figure 3.28: Frequency Response - Q vs Elevator - Mach Varying

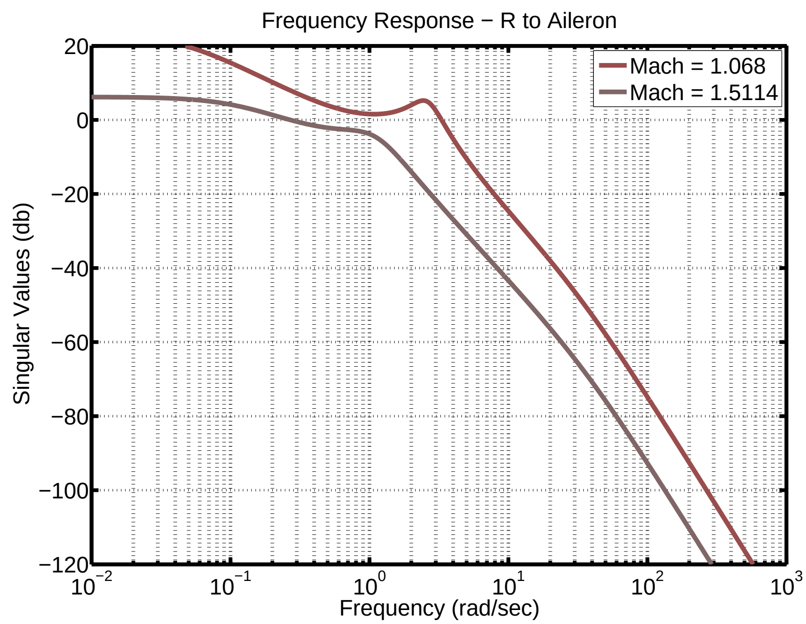

Figure 3.29: Frequency Response - R vs Aileron - Mach Varying

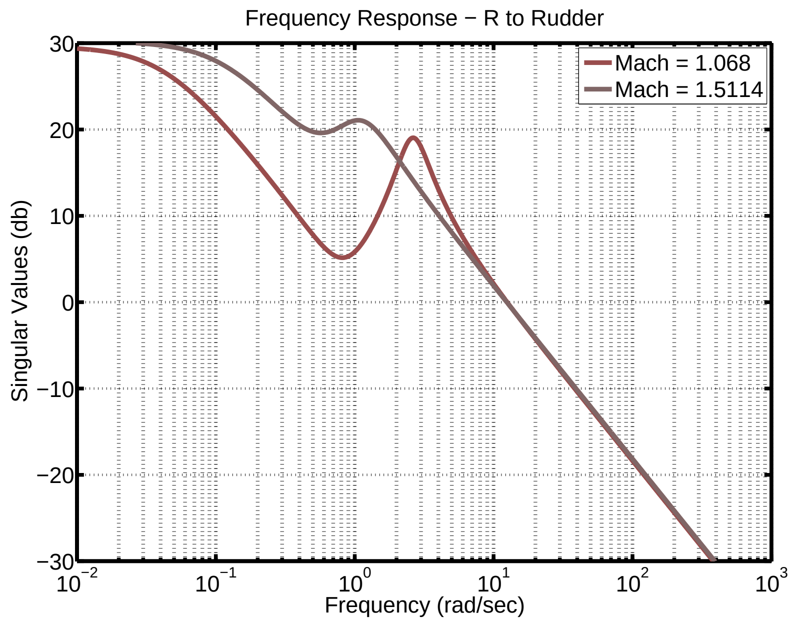

Figure 3.30: Frequency Response - R vs Rudder - Mach Varying

3.5 Discussion of BTT Missile Natural Modes (Eigenvalues)

Using the non-dimensional linear system of equations, equation 3.65, and the iterative trim procedure in MATLAB m-file "missile_plant_analysis.m", the characteristic modes of the BTT missile were investigated for the steady flight condition discussed in Section 3.2. The plant dynamics, or "A" matrix, of equation 3.65 was used in its presented form to find the characteristic modes of the missile. The condition number of the A-matrix in equation 3.65 was very large ($>1 \times 10^6$) due to the integration of p and q for ϕ and θ, respectively. After observing the relative sizes of the non-dimensional terms and the very weak longitudinal and lateral dynamic coupling, the following reduced systems are used for determining the characteristic modes:

States = [Axial Velocity, Vertical Velocity, Pitch Rate, Pitch Angle]

Controls = [Elevator Deflection]

Output of Interest = [Flight Path Angle γ]

$$\begin{bmatrix} \dot{U} \\ \dot{W} \\ \dot{Q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} X_u & X_w & X_q & X_\theta \\ Z_u & Z_w & Z_q & Z_\theta \\ M_u & M_w & M_q & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} U \\ W \\ Q \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \\ Z_{\delta_q} \\ M_{\delta_q} \\ 0 \end{bmatrix} \begin{bmatrix} \delta_q \end{bmatrix} \tag{3.66}$$

$$\begin{bmatrix} \gamma \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ W \\ Q \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} \delta_q \end{bmatrix} \tag{3.67}$$

Non-Minimum Phase Zero & Unstable Pole Dynamics Acceleration control of highly agile, aerodynamically-controlled missiles is a well-known non-minimum phase control problem [8]. Also to qualitatively understand this non-minimum phase behaviour consider the control problem of accelerating the missile upward. Typically a tail-controlled missile (i.e control surface aft of the center of gravity, G) is statically stable with $C_{m_\alpha} < 0$, $C_{z_\delta} < 0$ and $C_{m_\delta} < 0$. This means that a negative unit-step pitch deflection command initially induces a downward force on the missile causing the missile to accelerate downward. This downward force also induces a counter-clockwise pitching-moment about the center of gravity that tries to push the nose-up. But due to the inherent tendency of the missile to oppose any such change in angle of attack the missile continues to accelerate downward until an overall positive pitching moment about the center of gravity develops. Eventually the trim angle-of-attack and consequently the lift acting on the vehicle increase which together create an upward force about the fuselage; and thus the missile accelerates upward as desired. The above described non-minimum phase behaviour is a characteristic of several important tail controlled aerospace flight control problems such as control of Vertical Take-off and Landing (VTOL) aircraft, and Conventional Take-offand Landing (CTOL) aircraft. Pitch-up instability phenomenon occurs when center of pressure moves forward due to tip stall due to high angle of attack. Both the RHP Pole-Zero dynamics was captured here in linearization routine and their behaviour with different flight conditions are explained below. The decoupled longitunal system exhibits nonminimum phase behaviour with flight path angle dynamics. So naturally the below question arises in our mind.

> When does a Nonminimum phase system arise? What is the cause?
>
> Subtracting two systems where one has slow & weak dynamics and other has fast & strong dynamics will result in a nonminimum phase system.

Illustrative Example. Consider the following systems

$$\begin{aligned}\text{Slow/Weak Dynamics - Fast/Strong Dynamics} &= \frac{3}{s+1} - \frac{4}{s+2} \\ &= \frac{3(s+2) - 4(s+1)}{(s+1)(s+2)} \\ &= \frac{2-s}{(s+1)(s+2)}\end{aligned}$$

The nonminimum phase flight path angle with respect to the elevator deflection dynamics can be explained below

$$\begin{aligned}\frac{\lambda(s)}{\delta_q(s)} &= \text{Slow/Weak Dynamics - Fast/Strong Dynamics} \\ &= \frac{\alpha(s)}{\delta_q(s)} - \frac{\theta(s)}{\delta_q(s)}\end{aligned}$$

This holds true even if we try with a perfectly decoupled longitudinal system as both pitch and angle of attack parameters are longitudinal components. Similarly, the control problem of acceleration in upward direction with respect to the elevator deflection can be explained as follows

$$\begin{aligned} a_z &= \dot{w} - qu + pv - \text{cross coupling components} \\ &= \text{Slow/Weak Dynamics - Fast/Strong Dynamics} \end{aligned}$$

So, if we neglect the inertial cross coulping terms during linearization, we won't be able to capture the nonminimum phase behaviour. That is why, here in this research we get only minimum phase system here and this assumption is made to make the control design easy, when the nonlinear dynamic inversion technique is applied to get a nonlinear controller. This holds true even if we try with a perfectly decoupled system as cross coupling terms won't be present even there.

Thus, in general if two systems are combined such as $\frac{g_1}{s+p_1} - \frac{g_2}{s+p_2}$, the process will result in non-minimum phase behaviour if and only if

$$\frac{g_1}{p_1} - \frac{g_2}{p_2} > 0 \text{ and } g_1 - g_2 < 0.$$

Effect of Coupling on Zero-dynamics. When you have a tightly coupled system, the transmission zeros of the system are not the same as zeros in the plant input-output transfer functions. But when systems are decoupled (like in our case), then the transmission zeros of the system are the same as zeros in the plant input-output transfer functions.

Missile experiences higher dynamic pressure, "Q_{dp}" at lower altitudes, as a result of which the pitch up instability and nonminimum phase behaviour is very strong at those altitudes. And the magnitude of RHP pole and RHP zero decrease as altitude increases. The above said behaviour is captured well in Figures 3.31 & 3.35. While the effect of angle of attack on RHP pole-zero is opposite to that of altitude effects. As angle of attack increases, both the magnitude of RHP pole and RHP zero increased. This behaviour is captured well in Figures 3.33 & 3.37 respectively.

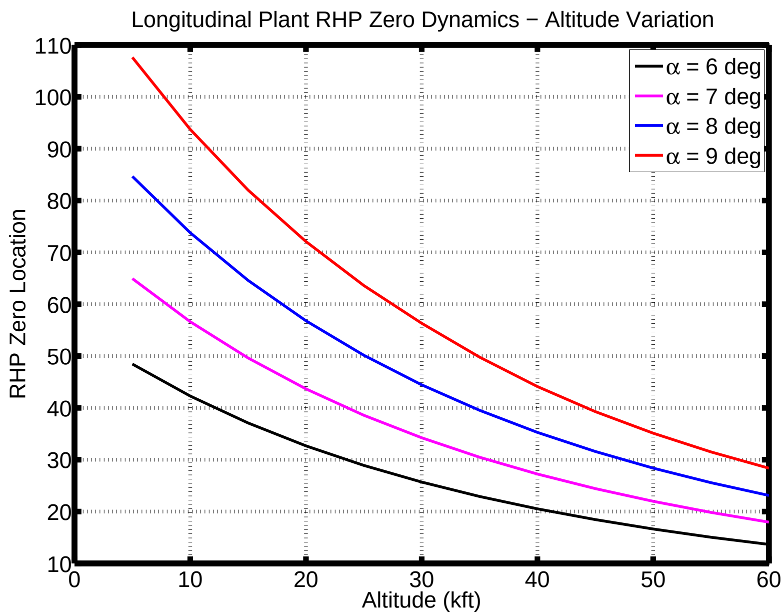

Figure 3.31: Longitunal Plant RHP Zero Dynamics - Altitude Varying

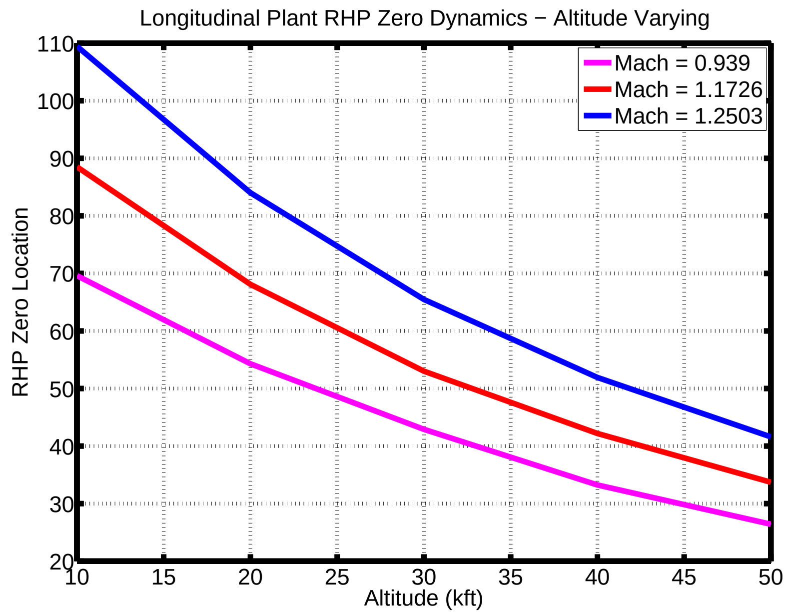

Figure 3.32: Longitunal Plant RHP Zero Dynamics - Altitude Varying With Mach

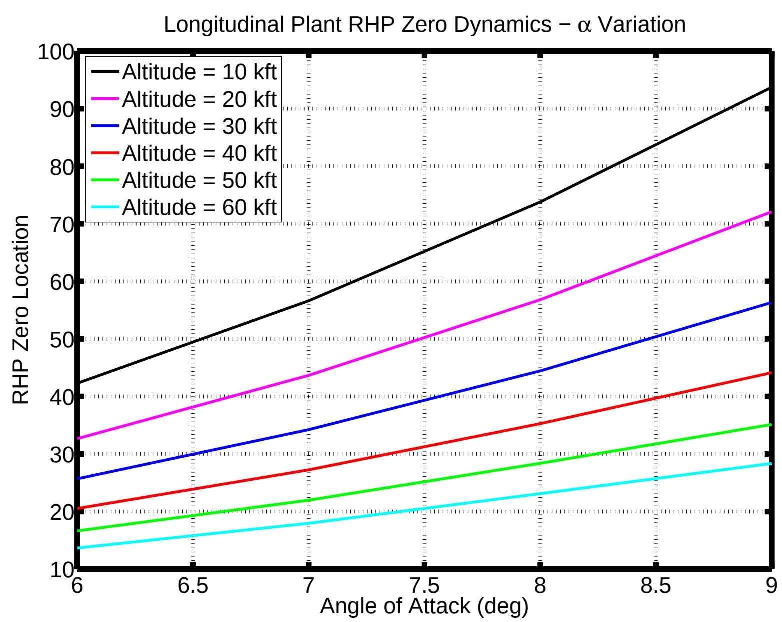

Figure 3.33: Longitunal Plant RHP Zero Dynamics - α Varying

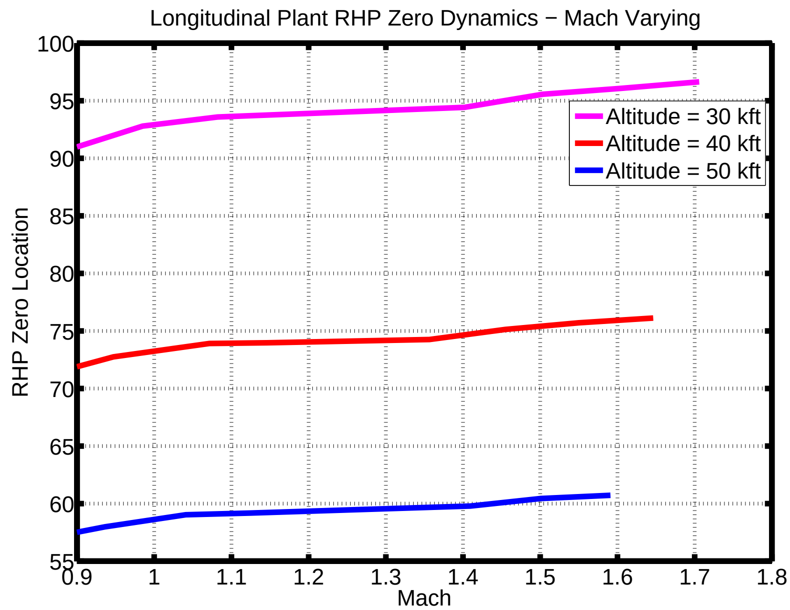

Figure 3.34: Longitunal Plant RHP Zero Dynamics - Mach Varying

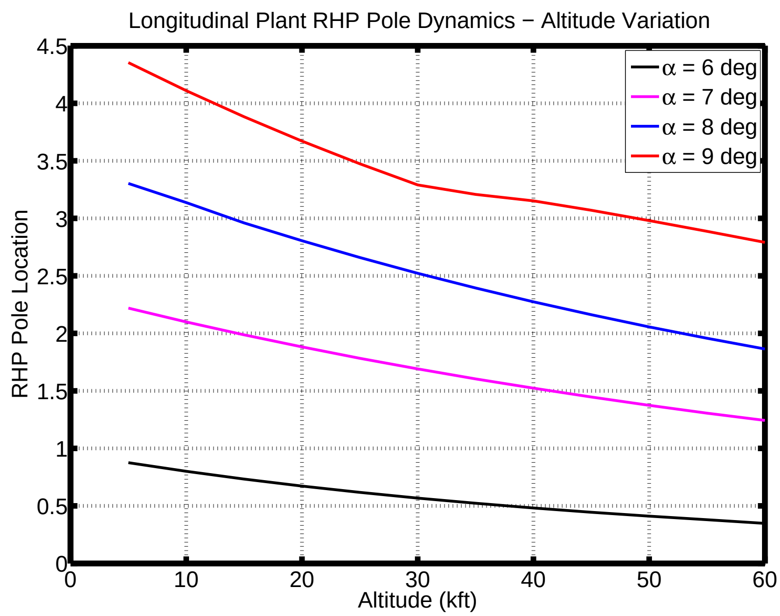

Figure 3.35: Longitunal Plant RHP Pole Dynamics - Altitude Varying

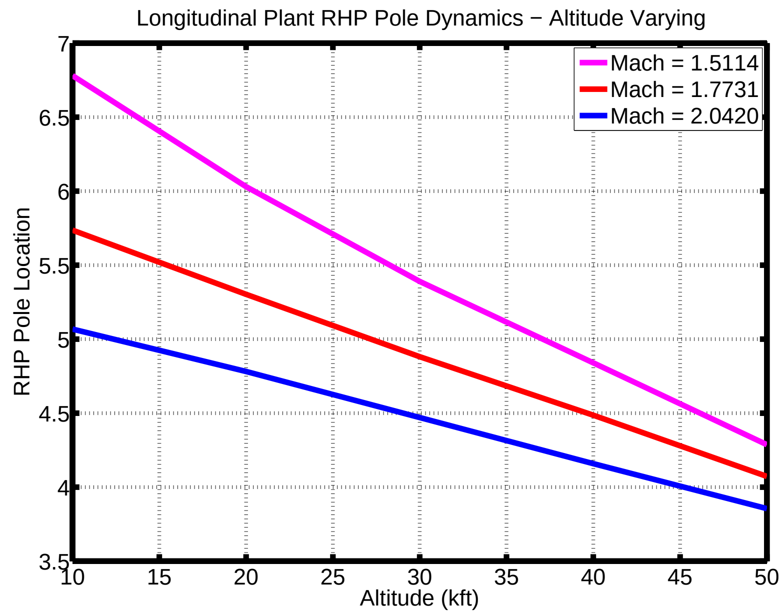

Figure 3.36: Longitunal Plant RHP Pole Dynamics - Altitude Varying With Mach

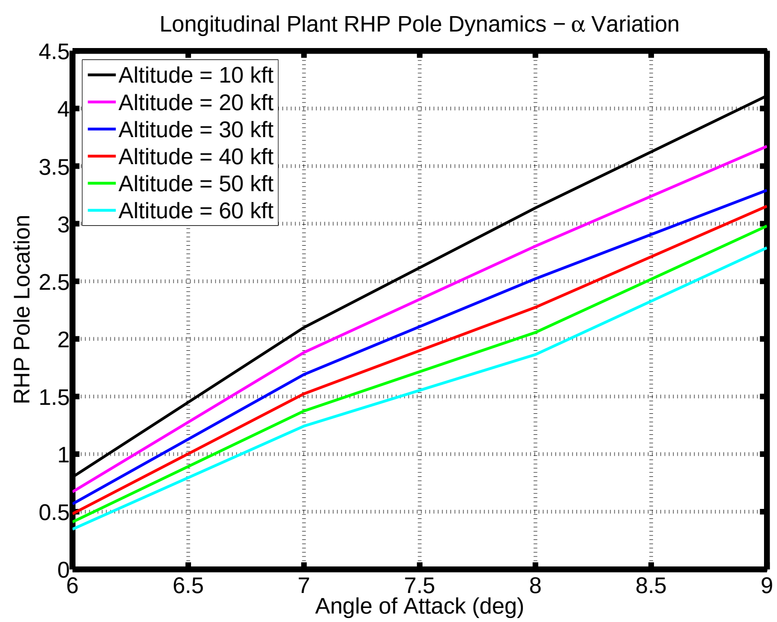

Figure 3.37: Longitunal Plant RHP Pole Dynamics - α Varying

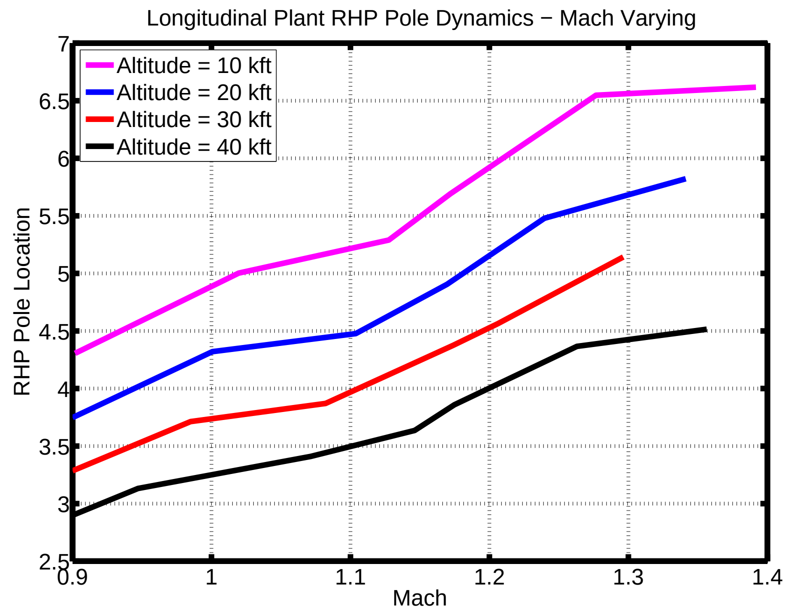

Figure 3.38: Longitunal Plant RHP Pole Dynamics - Mach Varying

3.5.2 Lateral Dynamics

States = [Lateral Velocity, Roll rate, Yaw Rate , Roll Angle]

Controls = [Aileron Deflection, Rudder Deflection]

Output of Interest = [Roll ϕ, Roll Rate P, Sideslip β, Yaw Rate R]

$$\begin{bmatrix} \dot{V} \\ \dot{P} \\ \dot{R} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} Y_v & Y_p & Y_r & Y_\phi \\ L_v & L_p & L_r & 0 \\ N_v & N_p & N_r & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V \\ P \\ R \\ \phi \end{bmatrix} + \begin{bmatrix} Y_{\delta_p} & Y_{\delta_r} \\ L_{\delta_p} & L_{\delta_r} \\ N_{\delta_p} & N_{\delta_r} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_p \\ \delta_r \end{bmatrix} \tag{3.68}$$

$$\begin{bmatrix} \beta \\ P \\ R \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ P \\ R \\ \phi \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_p \\ \delta_r \end{bmatrix} \tag{3.69}$$

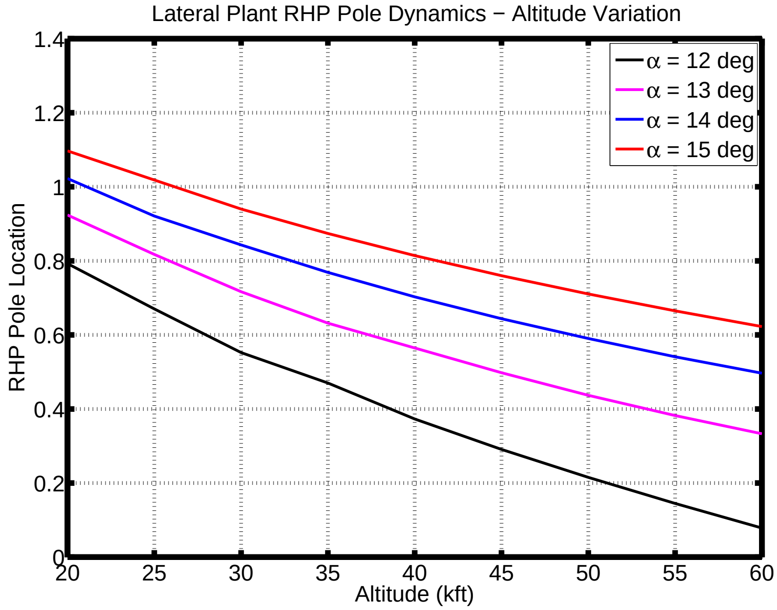

Figure 3.39: Lateral Plant RHP Pole Dynamics - Altitude Varying

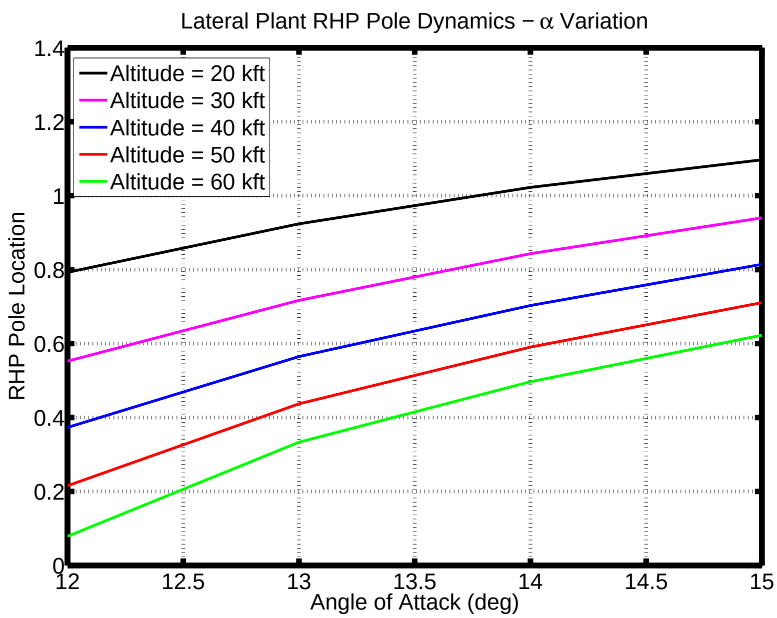

Figure 3.40: Lateral Plant RHP Pole Dynamics - α Varying

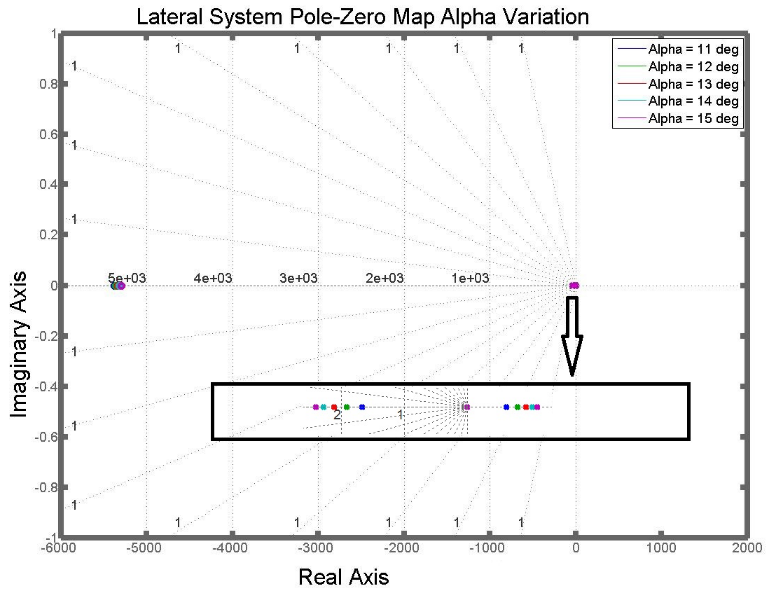

Figure 3.41: Lateral Plant Pole-Zero Map - α Varying

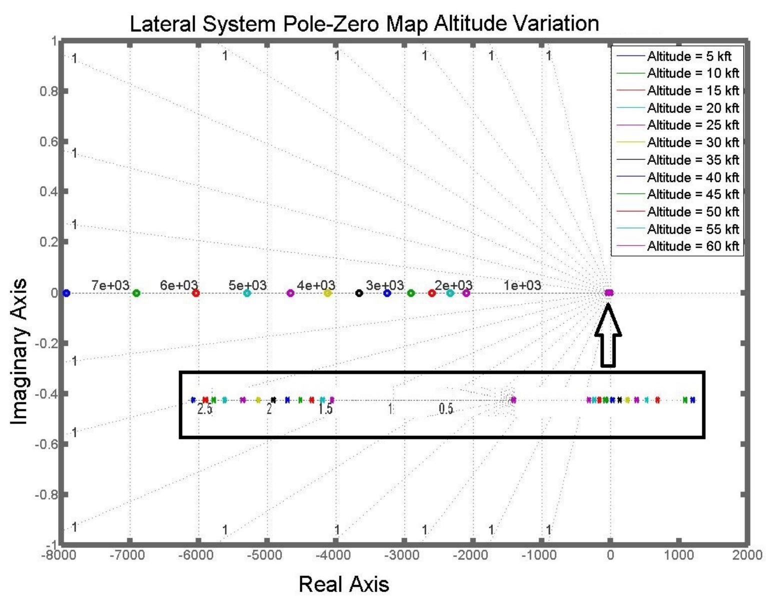

Figure 3.42: Lateral Plant Pole-Zero Map - Altitude Varying

Figures 3.39, 3.40, 3.41 & 3.42 show the calculated lateral eigenvalues for the above non-dimensional A-matrices. The unstable pole denotes the “spiral divergence mode”. This indicates a more sluggish response of the missile in the lateral direction at higher altitudes. Similar to the longitudinal dynamics, the unstable poles of the lateral dynamics move closer to the imaginary axis as altitude increases, while they move deeper into the RHP plane when angle of attack is increased.

Tables 3.1 & 3.3 show the longitudinal and lateral eigenvalues for the above non-dimensional A matrices when they are calculated for the missile flying at 10 kft and 40 kft respectively, for several different angles of attack, and initial time mass properties (fully fuelled missile). Similarly Tables 3.2 & 3.4 show the longitudinal and lateral eigenvalues for same angles of attack and altitude at 10 kft and 40 kft respectively but for the “fuel-spent” mass properties of the missile (fuel depleted missile).
The system modes for the non-dimensional system are given by the following equation (i.e., only if all the system eigenvalues are distinct):

$$\vec{x}(\tau) = \sum_{i=1}^{n} (\vec{p_i}\vec{x_0}) e^{\lambda_i \tau} \vec{q_i} = \sum_{i=1}^{n} (\vec{p_i}\vec{x_0}) e^{\lambda_i \frac{t}{\hat{t}}} \vec{q_i} \tag{3.70}$$

where
$\vec{p_i} \stackrel{\text{def}}{=}$ Left eigenvector of A associated with λ_i
$\vec{q_i} \stackrel{\text{def}}{=}$ Right eigenvector of A associated with λ_i

From equation (3.70) we can see that the aerodynamic time scaling factor given by equation 3.71

$$\hat{t} \stackrel{\text{def}}{=} \frac{m{V_b}^*}{{Q_{dp}}^* S_{ref}} = \frac{2m}{\rho^* {V_b}^* S_{ref}} \tag{3.71}$$

scales the response time of each mode. Since this is the case, we can see that the missiles mass, altitude (ρ is dependent on altitude), velocity magnitude, and aerodynamic reference area are very important in determining missile responsiveness. This is evident from lateral eigenvalues from tables 3.1 & 3.3. The lateral eigenvalues are about the same magnitude for a majority of the angles of attack but the time-scaling factors at higher altitude from table 3.3 are on the order of three times as large as those at lower altitudes given by the table 3.1. This indicates a more "sluggish" response of the missile in the lateral direction at higher altitudes (even though the crresponding Mach numbers are relatively close).

$(\hat{t})$	α **(deg)**	**Mach**	$Q_{dp}(lbf/ft^2)$	**Longitudinal Poles**	**Lateral Poles**
6.11	2	3.24	1.7 $\times 10^4$	-0.080$\pm$0.796i	-20.24, -0.12$\pm$2.35i
6.26	5	3.16	1.019 $\times 10^4$	-0.085$\pm$1.541i	-23.63, -0.34$\pm$2.13i
9.29	10	2.13	4.628 $\times 10^3$	6.357, -6.650	-3.24, 2.06, -34.32
8.91	15	2.22	5.034 $\times 10^3$	4.549, -4.843	-3.34, 1.71, -40.37
8.63	20	2.30	5.372 $\times 10^3$	-0.142$\pm$6.205i	-2.87, -48.53, 0.87

Table 3.1: Time-Zero Mass Properties for Altitude = 10 kft

$(\hat{t})$	α **(deg)**	**Mach**	$Q_{dp}(lbf/ft^2)$	**Longitudinal Poles**	**Lateral Poles**
3.70	2	3.24	1.7 $\times 10^4$	-0.062$\pm$0.540i	-18.24, -0.08$\pm$1.59i
3.79	5	3.16	1.019 $\times 10^4$	-0.065$\pm$1.045i	-21.30, -0.22$\pm$1.44i
5.63	10	2.13	4.628 $\times 10^3$	4.303, -4.521	-2.16, 1.41, -31.00
5.39	15	2.22	5.034 $\times 10^3$	3.077, -3.294	-2.20, 1.18, -36.55
5.22	20	2.30	5.372 $\times 10^3$	-0.103$\pm$4.209i	-1.85, -43.96, 0.62

Table 3.2: Fuel Spent Mass Properties for Altitude = 10 kft

$(\hat{t})$	α **(deg)**	**Mach**	$Q_{dp}(lbf/ft^2)$	**Longitudinal Poles**	**Lateral Poles**
17.86	2	3.69	3.744×10^3	1.896, -2.042	-20.29, -0.17±3.41i
18.34	5	3.59	3.55×10^3	2.495, -2.664	-23.12, -0.68±3.40i
25.16	10	2.62	1.887×10^3	9.505, -9.810	-5.31, 2.57, -32.43
26.39	15	2.50	1.716×10^3	7.628, -8.801	-6.44, 2.22, -38.22
25.75	20	2.56	1.802×10^3	-0.205±7.841i	-6.30, -44.82, 0.92

Table 3.3: Time-Zero Mass Properties for Altitude = 40 kft

$(\hat{t})$	α **(deg)**	**Mach**	$Q_{dp}(lbf/ft^2)$	**Longitudinal Poles**	**Lateral Poles**
10.81	2	3.69	3.744×10^3	1.280, -1.391	-18.22, -0.11±2.31i
11.10	5	3.59	3.55×10^3	1.687, -1.812	-21.05, -0.42±2.30i
15.23	10	2.62	1.887×10^3	6.443, -6.660	-3.42, 1.78, -29.81
15.98	15	2.50	1.716×10^3	5.172, -5.434	-4.05, 1.56, -35.46
15.59	20	2.56	1.802×10^3	-0.139±5.319i	-3.83, 0.67, -41.74

Table 3.4: Fuel Spent Mass Properties for Altitude = 40 kft

$(\hat{t})$	α **(deg)**	**Mach**	$Q_{dp}(lbf/ft^2)$	**Longitudinal Poles**	**Lateral Poles**
26.36	2	2.0	1.72 $\times 10^3$	-0.1142±2.94i	-19.57, -0.25±5.07i
26.36	5	2.0	1.72 $\times 10^3$	-0.1324±4.64i	-19.57, -0.25±5.07i
26.36	10	2.0	1.72 $\times 10^3$	9.9, -10.23	2.84, -5.52, -33.32
26.36	15	2.0	1.72 $\times 10^3$	-7.63, -8.00	2.22, -6.43, -38.22
26.36	20	2.0	1.72 $\times 10^3$	-0.2124±8.49i	0.98, -6.51, -45.36

Table 3.5: α Variation for Alt. = 40 kft, Mach = 2.0

$(\hat{t})$	α **(deg)**	**Mach**	$Q_{dp}(lbf/ft^2)$	**Longitudinal Poles**	**Lateral Poles**
73.21	15	0.9	222.89	-17.01, 15.28	-2.43, -0.08±1.59i
43.93	15	1.5	619.16	-16.14, 15.32	1.33, -0.22±1.44i
32.94	15	2.0	1.101 $\times 10^3$	-8.62, 8.08	2.71, -8.09, -36.31
21.96	15	3.0	2.477 $\times 10^3$	-7.90, 7.61	1.43, -4.80, -38.20
16.47	15	4.0	4.403 $\times 10^3$	-9.19, 9.00	-40.66, -0.80±2.78i

Table 3.6: Mach Variation for Alt. = 40 kft, α = 15 deg

From the above tables, we can see that the longitudinal and lateral modes are very dependent on the angle of attack and Mach number for a given altitude. The affects of Mach number and angles of attack on two of the pertinent longitudinal aerodynamic stability derivatives are illustrated in Figures 2.15 and 2.17. Figure 2.15 shows a plot of C_{M_α} versus angle of attack and Mach number. The Figure 2.17 shows a plot of C_{M_Q} versus angle of attack and Mach number. Although the angle of attack and Mach number are very important factors that influence the modes of the missile, a quick comparison of Tables 3.1 & 3.3 show that they are not the only influential factors. Note the longitudinal eigenvalues for an angle of attack of 2 deg in Tables 3.1 & 3.3. Although the corresponding Mach numbers only differ by 12%, the eigenvalues of Table 3.1 for this case are a pair of stable complex poles while the corresponding eigenvalues of Table 3.3 consist of one stable and one unstable pole. The major difference between these two cases is that the dynamic pressure is much smaller for this case in Table 3.3. However, this should not be surprising if we inspect the non-dimensional stability derivatives of Section 3.4. In Section 3.4, we see that the non-dimensional stability derivative, m_w, is inversely proportional to the dynamic pressure. The dynamic pressure in Table 3.1 is about 4.5 times as large as the corresponding angle of attack in Table 3.3. This indicates that the non-dimensional stability derivative, m_w, of Table 3.3 is 4.5 times as large as that in Table 3.1 for this condition (i.e., angle of attack and approximately the same Mach number).

The trim conditions in Tables 3.1 through 3.4 let the missile Mach number vary (and thus dynamic pressure), that is calculated by solving the longitudinal part of trim equations (3.52). Since the missile is assumed not to have any throttle control, the missile can not be trimmed to a specified Mach number for a given angle of attack. However, it is of interest to see how the longitudinal and lateral eigenvalues vary as a function of only Mach number while holding angle of attack constant and vice versa. Simply substituting a Mach number, angle of attack, side-slip angle, etc., into trim equations (3.52) has inherent errors associated with it since, more likely than not, there does not exist a set of fin deflections which can be found to satisfy these equations. For example, considering only the longitudinal plane, we only have one independent control variable (pitch fin deflection angle). Thus, we can only specify one dependent variable to trim (if we had a propulsive throttle control we could trim Mach number and angle of attack simultaneously). In Tables 3.1 through 3.4 we chose angle of attack as the dependent trim variable and let Mach number and dynamic pressure vary. However, in Table 3.5 angle of attack is varied while holding altitude constant at 40kft and Mach constant at 2.0 (and thus dynamic pressure). In Table 3.6, the angle of attack is held constant at 15 degrees, altitude is held constant at 40kft, and the missile Mach number is varied between 0.9 and 4.0.

In short, the major factors that influence the modes of missile can be seen from the form of the non-dimensional stability derivatives of the previous section. We can see that the dynamic pressure, the missile mass properties, and missile aerodynamic reference areas and lengths scale the non-dimensional stability derivatives. In addition, the stability derivatives themselves, as can be seen in Figures 2.15 and 2.17, are very dependent on the angle of attack and Mach number. As a note, it should not now be surprising to the reader to learn that many missile and aircraft flight control systems are gain scheduled as a function of dynamic pressure, angle of attack, and Mach number.

3.6 Missile Static Analysis - Elevator & Throttle Trim

Static analysis is needed to study how missile controls vary to attain a commanded flight condition. Given the saturation limits on both missile fin actuators and fin rates, this static analysis will throw light on missile flight conditions which will result in fin actuator saturation. Thus in this linearization routine where a steady level flight for missile is considered, static analysis is performed on the trim elevator and trim throttle conditions and different flight parameters affecting that is studied.

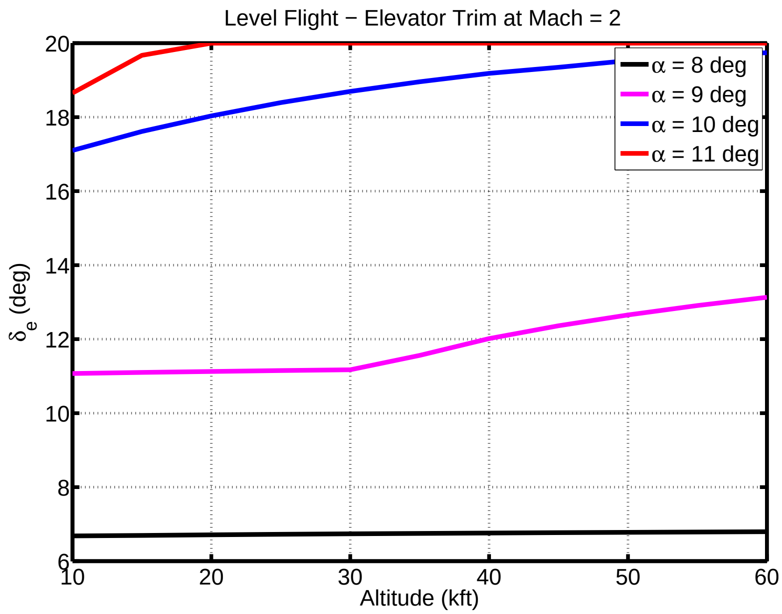

Figure 3.43: Level Flight - Elevator Trim for Altitude

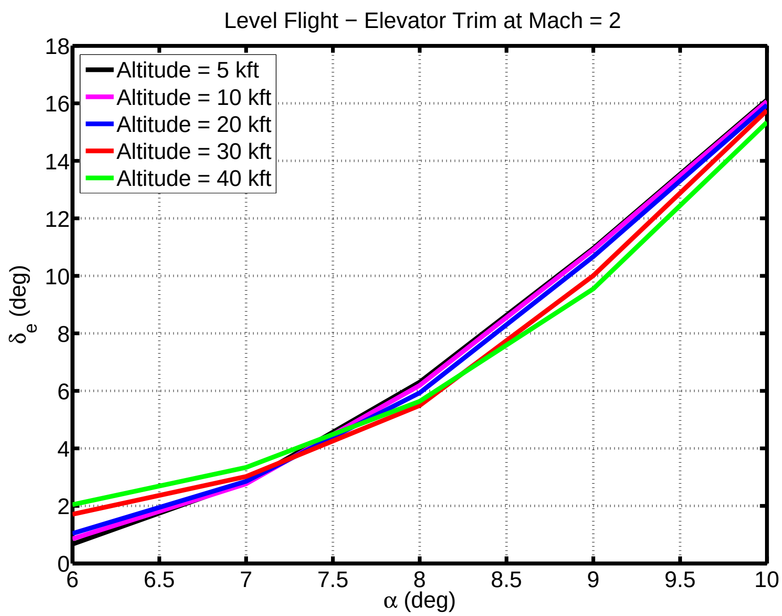

Figure 3.44: Level Flight - Elevator Trim for α

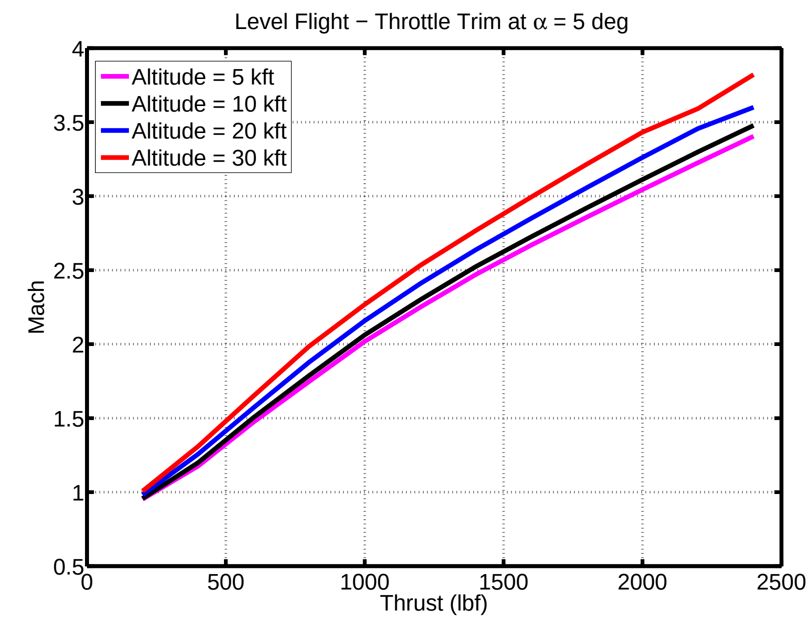

Figure 3.45: Level Flight - Throttle Trim

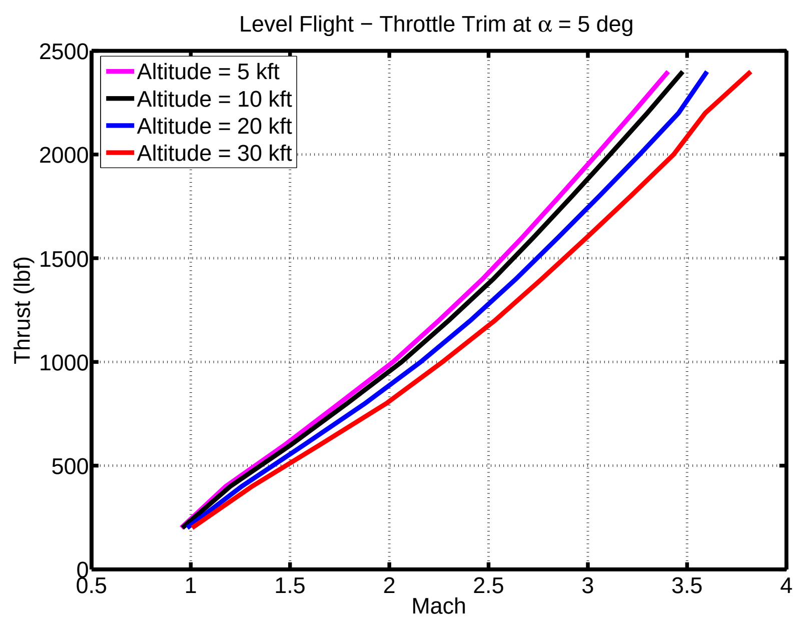

Figure 3.46: Level Flight - Throttle Trim for Mach

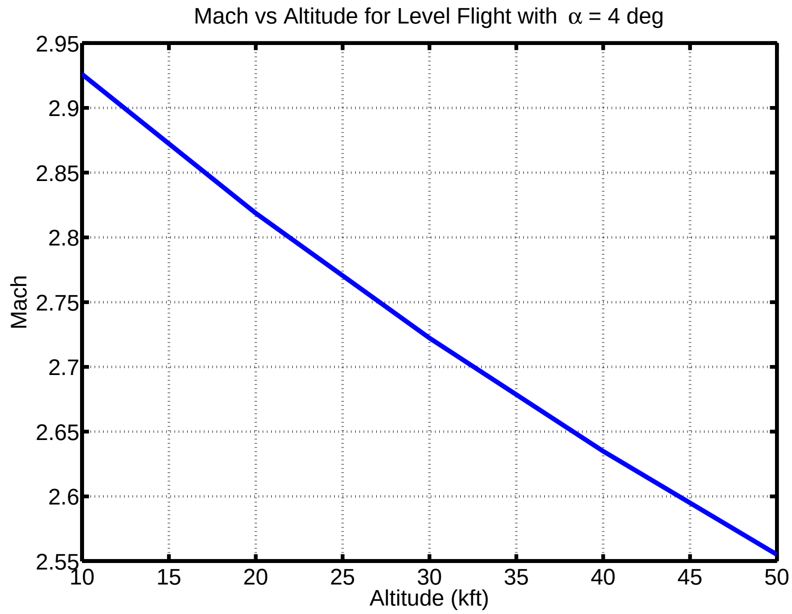

Figure 3.47: Level Flight - Mach Varying with Altitude

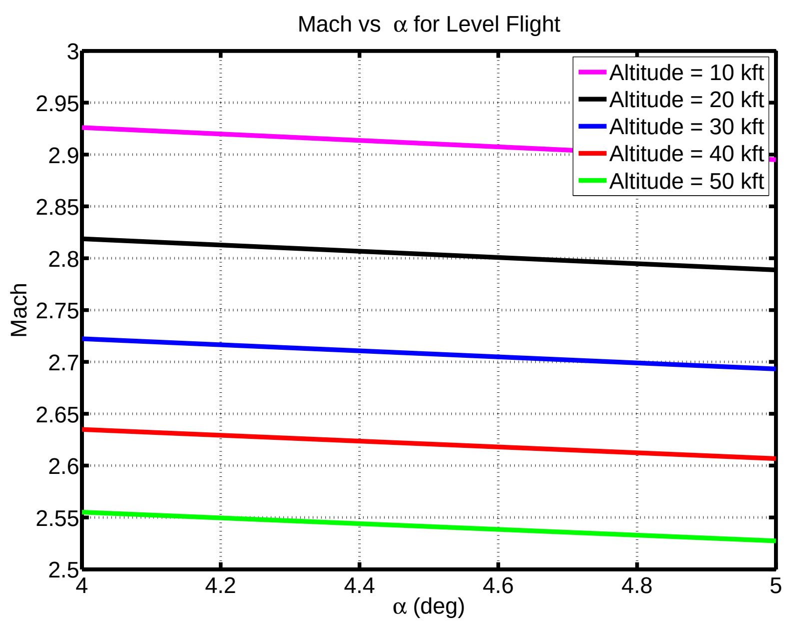

Figure 3.48: Level Flight - Mach Varying with α

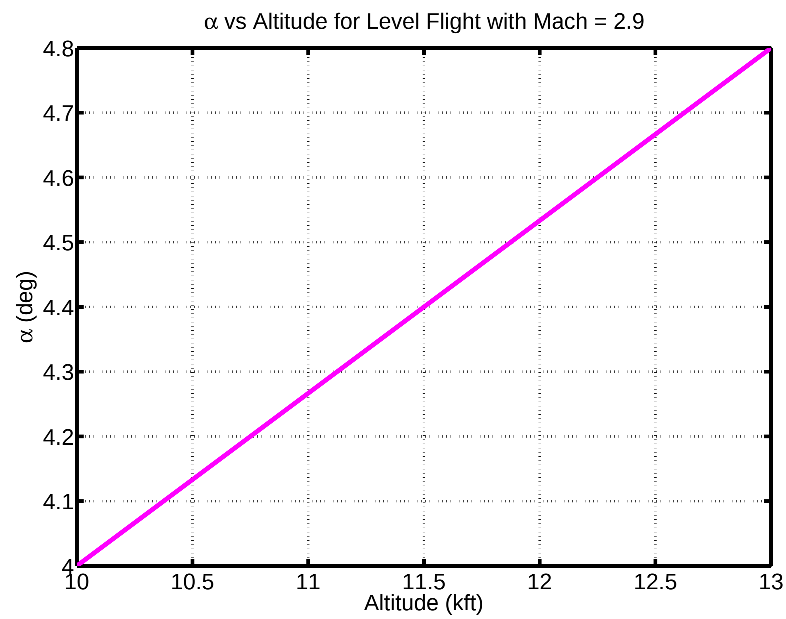

Figure 3.49: Level Flight - α Varying with Altitude

From figure 3.43, it is clear that when a rise in altitude is demanded, the elevator deflection increases. Also when positive angle of attack is commanded, the elevator fin deflection increases. This is expected because, elevator deflection is responsible for the missile to pitch up or down. While positive elevator deflection pitches up the missile to match the commanded angle of attack or altitude, negative deflection does the opposite. Interesting point to note here is the fin saturation level. If higher angle of attack or altitude is commanded, elevator deflection saturates. While a linear behaviour is exhibited by the fin deflection with respect to change in altitude and angle of attack, the same behaviour is lost and saturation occurs beyond certain commanded values. This is very much evident from the figure 3.43. Thus given this detailed analysis, one should not command more than some threshold angle of attack or altitude values as fin deflections will saturate beyong those threshold values.

From Figures 3.47, 3.48 & 3.49 respectively, the following concepts are very evident.

1. $Mach \propto \frac{1}{h}$

2. $Mach \propto \frac{1}{\alpha}$

3. $h \propto \alpha$

3.7 Summary and Conclusions

In this chapter, mathematical modeling of a BTT missile was discussed. Modeling included linearization routine using perturbation technique and time scaling. Both lateral and longitudinal models were presented in detail. The nonminimum phase zeros and unstable poles in both longitunal and lateral dynamics were analyzed in detail. Finally the missile static analysis was performed for elevator trim and throttle trim conditions and effect of various flight parameters on fin saturations was presented.

Chapter 4

MISSILE SEEKER / NAVIGATION & GUIDANCE

4.1 Introduction and Overview

This chapter describes the seeker/navigation system dynamics and the three guidance options available to the missile. Navigation is traditionally defined as knowing the location of a missile [62]. This is essential in long distance applications such as inter-Continental Ballistic Missiles (ICBMs). For EMRAAT missile being considered, navigation involves using range and range-rate information to determine where it is with respect to its target. Hence, in this document, the term navigation is used to refer to the missile determining its location with respect to the target.

A seeker is a range and angle sensing instrument which resides in the forward portion of the missile. It provides the guidance system with information about the evading target. The gimbals isolate the gyros from the missile's rotational environment, as explained in [60]. The seeker/navigation system can be visualized as shown in the Figure 4.1. It consists of a (1) Relative Range/Rate Generator, (2) LOS Angle Generator, (3) A/D Quantizing block, (4) Gimbal Angle/Rate Error Generator and a (5) Gimbal Rate Generator. Each subsystem is described in this chapter. The next section describes the seeker/navigation system in greater detail.

The missile guidance systen processes range and range-rate information from the seeker / navigation system and generates commanded horizontal and vertical accelerations to the autopilot. Three guidance laws are available to the missile.

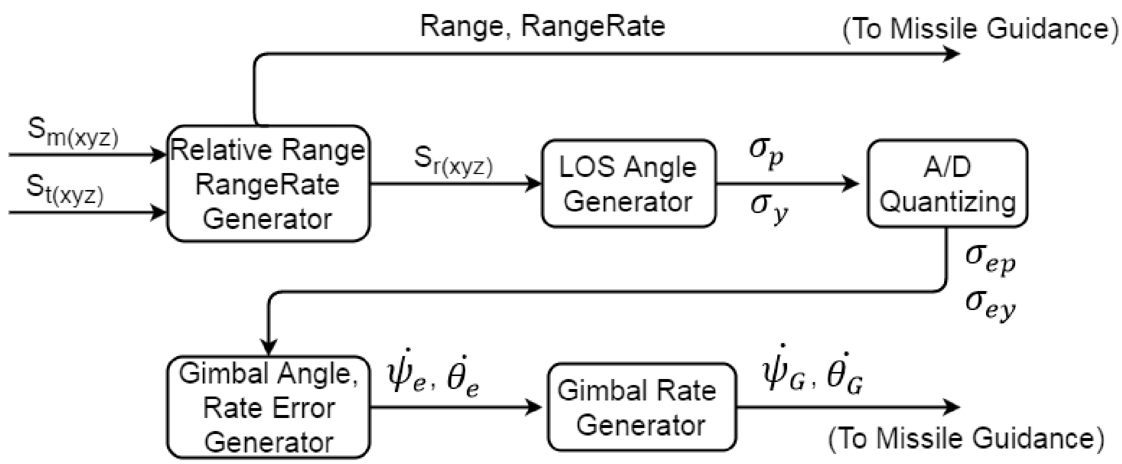

Figure 4.1: Block Diagram of Seeker/Navigation Model Algorithm

1. Proportional Navigation Guidance
2. Optimal Control Theory Navigation
3. Differential Game Theory Navigation

All three guidance laws are discussed in details in this chapter.

4.2 Seeker Frame

The missile tracks its target using a range and angle sensing system called the seeker. The seeker sits on a gimballed platform, affixed toward the nose of the missile. The seeker frame is a right handed coordinate system with its origin located at the time-zero missile's center of gravity CG_0. Although the seeker is located in the forward part of the missile, in this model the seeker frame origin is located at the missile CG_0 for mathematical convenience. At large distance and for large closing velocity, the error due to this misalignment is innocuous [2]. Its axes are denoted (X^s, Y^s, Z^s) and perfect tracking alignment is achieved when the seeker X^s positive axis passes through the target's position. Seeker gimbal angles $(\psi_s, \theta_s, 0)$ represent the measured azimuth and elevation of the seeker frame relative to X^bZ^b and X^bY^b planes of the body frame. This orientation of the seeker platform relative to the body frame is shown in Figure 4.2. The seeker frame is used to describe error between the actual flight path and the desired flight path. Vectors are transformed between the seeker and body frames by transformation matrices which use the seeker gimbal angles $(\psi_s, \theta_s, 0)^s$.

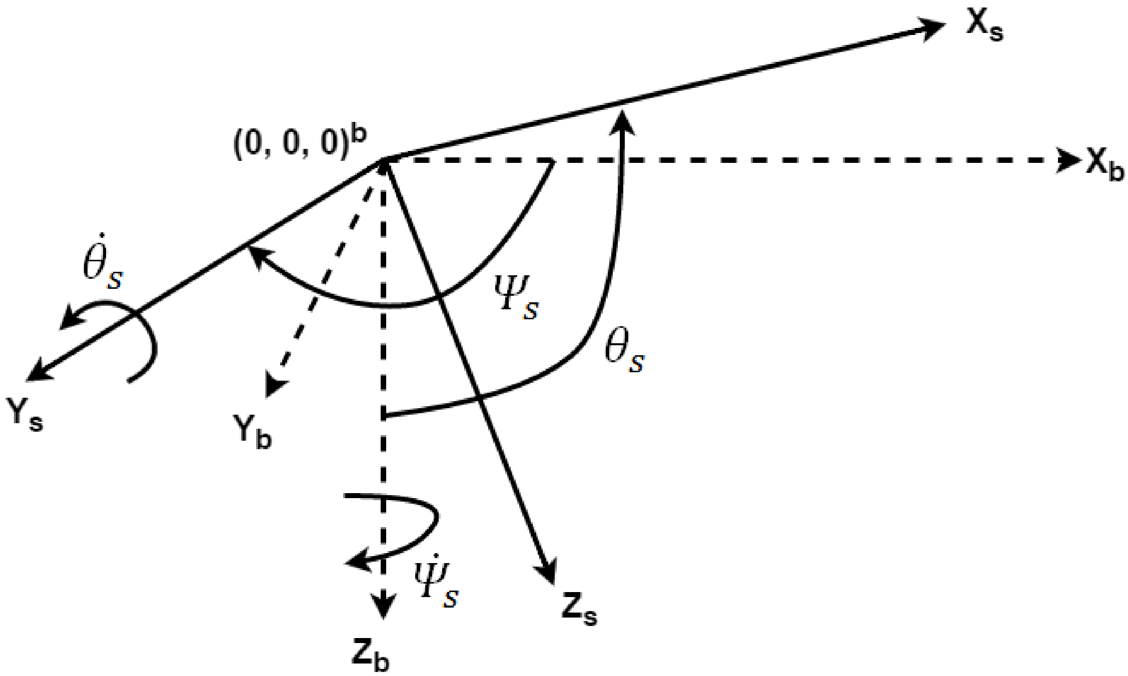

Figure 4.2: Seeker Frame orientation with respect to Seeker Gimbal Angles

A polar form of the target position in the seeker frame is given by components of radial distance Range and seeker line-of-sight angles (σ_y, σ_p) as shown in the Figure 4.3, where line-of-sight is defined as the distance from the missile center-of-gravity to the target center-of-gravity. σ_y corresponds t the azimuth angle and σ_p corresponds to the elevation angle. These angles are calculated as a function of the seeker frame

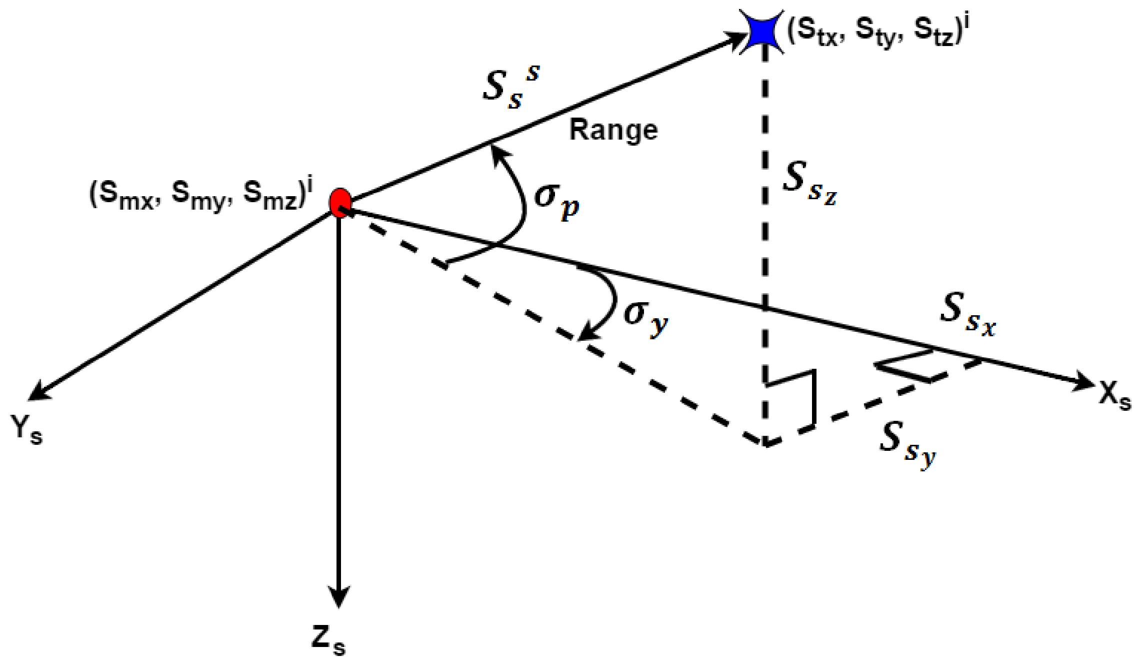

Figure 4.3: Seeker Frame Line-of-Sight Angles (σ_y, σ_p) and Range

representation of the vehicle relative displacement $S_r{}^v$. The vehicle realtive vector $S_r{}^v$ identified as $S_s{}^s$ in the seeker frame, is found by the following equation:

$$S_s{}^s = [T_{bs}][T_{vb}]S_r{}^v \tag{4.1}$$

The 3×3 vehicle-to-body transformation matrix, denoted by T_{vb} is given by the following equations. To transform a vector from body frame to the vehicle frame, the transposed matrix $T_{vb}{}'$ is used.

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}_{body} = T_{vb} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}_{vehicle} \tag{4.2}$$

$$T_{vb} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\theta)\cos(\psi)\sin(\phi) - \sin(\psi)\cos(\phi) & \sin(\theta)\sin(\psi)\sin(\phi) - \cos(\psi)\cos(\phi) & \cos(\theta)\sin(\phi) \\ \sin(\theta)\cos(\psi)\cos(\phi) + \sin(\psi)\sin(\phi) & \sin(\theta)\sin(\psi)\sin(\phi) - \cos(\psi)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{4.3}$$

The 3×3 body-to-seeker transformation matrix, denoted by T_{bs} is given by the following equations. To transform a vector from the seeker frame to the body frame, the transposed matrix $T_{bs}{}'$ is used.

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}_{seeker} = T_{bs} \begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix}_{body} \tag{4.4}$$

$$T_{bs} = \begin{bmatrix} \cos(\theta_s)\cos(\psi_s) & \cos(\theta_s)\sin(\psi_s) & \sin(\theta_s) \\ \sin(\psi_s) & \cos(\psi_s) & 0 \\ \sin(\theta_s)\cos(\psi_s) & \sin(\theta_s)\sin(\psi_s) & \cos(\theta_s) \end{bmatrix} \tag{4.5}$$

The seeker LOS angles are then found by:

$$\sigma_y = \tan^{-1}\left(\frac{S_{s_y}}{S_{s_x}}\right) \tag{4.6}$$

and:

$$\sigma_p = \tan^{-1}\left(\frac{-S_{s_z}}{\sqrt{{S_{s_x}}^2 + {S_{s_y}}^2}}\right) \tag{4.7}$$

4.3 Seeker Dynamics

The following section describes how the missile tracks its target. the seeker reference frame is used to describe error between desired and actual missile flight path.

4.3.1 Seeker Model Software Algorithm

This section describes how the seeker dynamics are modelled in the software. Each of the blocks in the Figure 4.1 are now described.

Relative Range Rate Generator

A definition is definitely needed to conveniently describe the distance between the missile and target. The vehicle separation is found as the difference between the inertial frame target position and the inertial frame missile position. The calculations in block one, see Figure 4.1 are described by the following equations. The relative separation is defned as follows:

$$S_r{}^v \stackrel{\text{def}}{=} S_t{}^i - S_m{}^i \tag{4.8}$$

with components

$$S_r{}^v = (S_{r_x}, S_{r_y}, S_{r_z}) \tag{4.9}$$

The relative velocity is defined as follows:

$$V_r{}^v \stackrel{\text{def}}{=} V_t{}^i - V_m{}^i \tag{4.10}$$

with components

$$V_r{}^v = (V_{r_x}, V_{r_y}, V_{r_z}) \tag{4.11}$$

The vehicle relative separation can be visualized as shown in the Figure 4.4.

LOS Angle Generator

To find the perfect Line-of-sight (LOS) angles, σ_y and σ_p, the relative target information is transformed first from the relative frame into the body frame and then into the seeker frame. The seeker LOS angles are then found by:

$$\sigma_y = tan^{-1}\left(\frac{S_{s_y}}{S_{s_x}}\right) \tag{4.12}$$

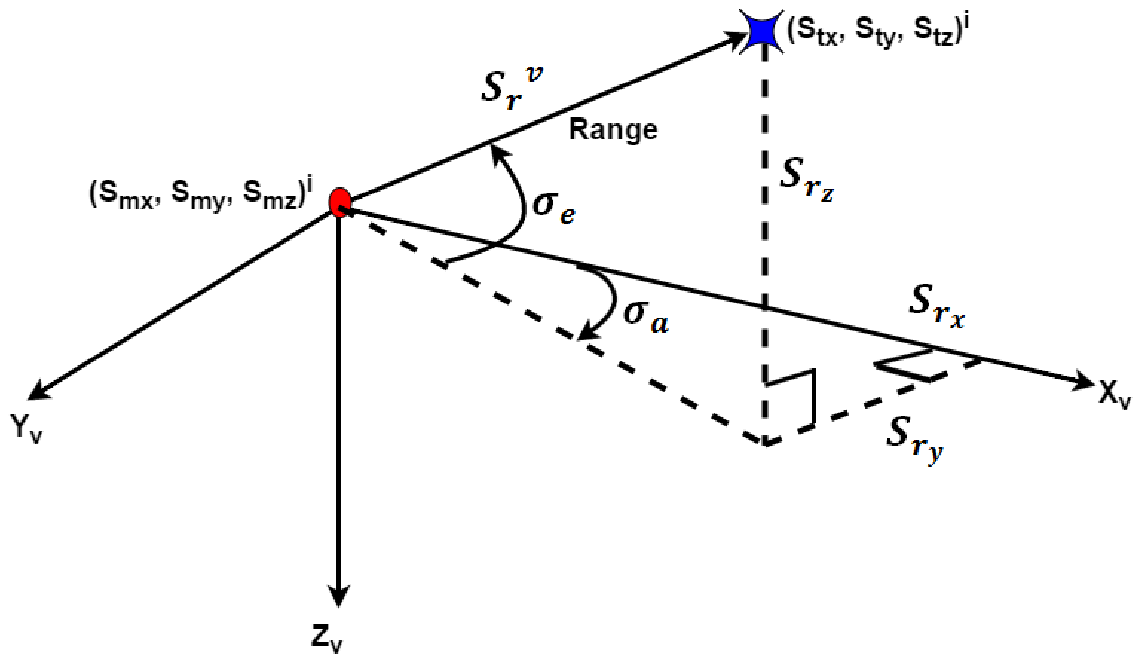

Figure 4.4: Visualization of Vehicle Relative Separation

and:

$$\sigma_p = tan^{-1}\left(\frac{-S_{s_z}}{\sqrt{{S_{s_x}}^2 + {S_{s_y}}^2}}\right) \tag{4.13}$$

A/D Quantizing

The perfect LOS angles (σ_y, σ_p) are then multiplied by 1000, truncated to three significant digits and divided by 1000 to simulate A/D quantizing error, forming $(\sigma_{ey}, \sigma_{ep})$. σ_{ey} is limited to $\pm$ 2 deg and σ_{ep} is limited to $\pm$ 4 deg.

Gimbal Angle Rate Error Generator

The measured error angles $(\sigma_{ey}, \sigma_{ep})$ are passed through a second order underdamped system described by the following equations:

$$\ddot{\psi}_e + 2\zeta_s\omega_s\dot{\psi}_e + \omega_s^2\psi_e = \omega_s^2\sigma_{ey} \tag{4.14}$$

$$\ddot{\theta}_e + 2\zeta_s\omega_s\dot{\theta}_e + \omega_s^2\theta_e = \omega_s^2\sigma_{ep} \tag{4.15}$$

where $\zeta_s \stackrel{\text{def}}{=}$ is the damping ratio of the seeker servos and is equal to 0.35
$\omega_s \stackrel{\text{def}}{=}$ is the servo natural frequency of oscillation and is equal to 49.5 $(\frac{rad}{sec})$

The seeker gimbal yaw and pitch error angles (ψ_e, θ_e) and their rates $(\dot{\psi}_e, \dot{\theta}_e)$ are taken as the output of the underdamped system. this is given in state space form by the following matrix equations. The state equations equivalent to the yaw axis equation is given by:

$$\begin{bmatrix} \dot{\psi}_e \\ \ddot{\psi}_e \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_s{}^2 & -2\zeta_s\omega_s \end{bmatrix} \begin{bmatrix} \psi_e \\ \dot{\psi}_e \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_s{}^2 \end{bmatrix} \begin{bmatrix} \sigma_{ey} \end{bmatrix} \tag{4.16}$$

The state equations equivalent to the yaw axis equation is given by:

$$\begin{bmatrix} \dot{\theta}_e \\ \ddot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_s{}^2 & -2\zeta_s\omega_s \end{bmatrix} \begin{bmatrix} \theta_e \\ \dot{\theta}_e \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_s{}^2 \end{bmatrix} \begin{bmatrix} \sigma_{ep} \end{bmatrix} \tag{4.17}$$

Gimbal Rate Generator

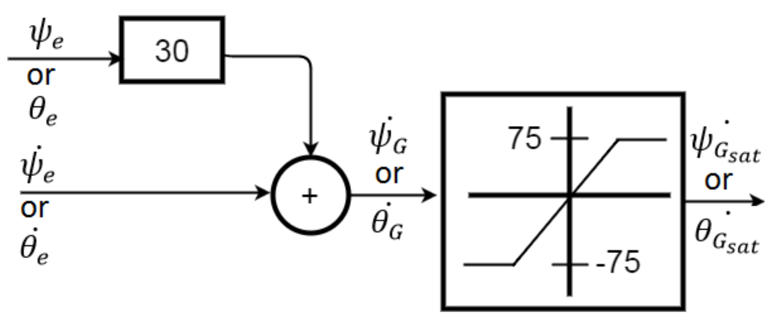

Figure 4.5: Commanded Gimbal Rate Generator

The gimbal error angles and rates are scaled using:

$$\dot{\psi}_G = \dot{\psi}_e + 30\psi_e \tag{4.18}$$

$$\dot{\theta}_G = \dot{\theta}_e + 30\theta_e \tag{4.19}$$

and limited by:

$$|\dot{\psi}_{G_{sat}}| < 75\frac{deg}{sec} = \dot{\psi}_{G_{max}} \tag{4.20}$$

$$|\dot{\theta}_{G_{sat}}| < 75\frac{deg}{sec} = \dot{\theta}_{G_{max}} \tag{4.21}$$

to form commanded gimbal rates $(\dot{\psi}_{G_{sat}}, \dot{\theta}_{G_{sat}})$. Figure 4.5 shows a block diagram of the scale and limit process.

Gimbal angles (ψ_G, θ_G) are found by the following equations:

$$\psi_G = \int \dot{\psi}_G - (P, Q, R)^s \tag{4.22}$$

$$\theta_G = \int \dot{\theta}_G - (P, Q, R)^s \tag{4.23}$$

where $(P, Q, R)^s$ represents the missile angular velocities transformed into the seeker frame.

The servo deflections are limited in position to:

$$\psi_{G_{max}} = \pm 65 deg \tag{4.24}$$

$$\theta_{G_{max}} = \pm 70 deg \tag{4.25}$$

The navigation seeker model simulates A/D quantization error as described above. It would also be desirable to introduce noise in the relative displacement calculations.

4.3.2 Seeker Dynamics Block Diagram

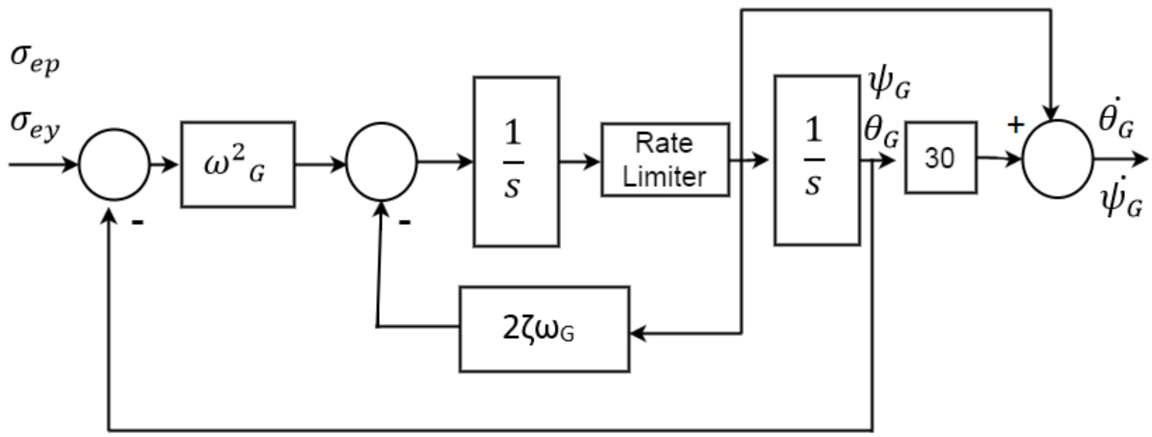

Figure 4.6: Block Diagram of Seeker Dynamics

Figure 4.6 shows the seeker dynamics in block diagram form. The block diagram only shows the seeker azimuth angle ψ_G, however the same block diagram is valid for the seeker elevation angle θ_G.

Neglecting nonlinearities, the seeker has a transfer function matrix given by:

$$H(s) = \left[\frac{{\omega_G}^2(s+30)}{s^2 + 2\zeta_G\omega_G s + {\omega_G}^2}\right] I_{2\times 2} \tag{4.26}$$

where the first channel governs the azimuth gimbal dynamics and the second channel the elevation gimbal dynamics. Minimum phase zero in the transfer function is due to the scaling operation explained above. Here

$\zeta_G \overset{\text{def}}{=}$ is the damping ratio of the seeker servos and is equal to 0.35
$\omega_G \overset{\text{def}}{=}$ is the servo natural frequency of oscillation and is equal to 49.5 $(\frac{rad}{sec})$

Servo deflections are limited in position [1].

4.4 Guidance Algorithms

The missile guidance makes corrections to keep the missile on course by sending appropriate acceleration commands to the autopilot [62], see Figure 1.1. Missile Commanded horizontal and vertical body-frame accelerations A_{yc} and A_{zc} are generated from measured relative target range and range-rate. Three guidance laws are available to the missile autopilot. They are,

1. Proportional Navigation Guidance
2. Optimal Control Theory Navigation
3. Differential Game Theory Navigation [3]

4.4.1 Proportional Navigation Guidance

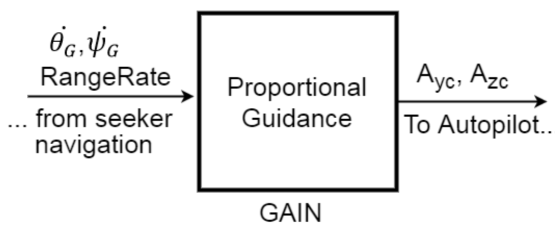

Figure 4.7: Proportional Navigation Guidance

The proportional guidance as shown in Figure 4.7 can be mathematically described by the following equations.

$$A_{yc} = -p_{g_1}\, RangeRate\, \dot{\psi}_G \tag{4.27}$$

$$A_{zc} = p_{g_2}\, RangeRate\, \dot{\theta}_G \tag{4.28}$$

[1]Not Shown in block diagram. See code in Appendix A.

where A_{yc} and A_{zc} are in the directions of the body frame axis Y^b and Z^b respectively. The proportional navigation gains are summarized in Table 4.1:

$p_{g_1} = 3.0$
$p_{g_2} = 3.0$

Table 4.1: Proportional Guidance Gains

For proportional guidance, the missile is commanded to turn at a rate proportional to the angular velocity of the line-of-sight. If the proportional guidance gains, p_{g_1} and p_{g_2} are small, the missile will respond slowly and it will not be able to catch the target. If the gains are too large the (outer) guidance loop will become unstable due to the high frequency seeker dynamics, see [62].

4.4.2 Optimal Control Theory Guidance

The optimal guidance shown in Figure 4.8 can be described using the following equations.

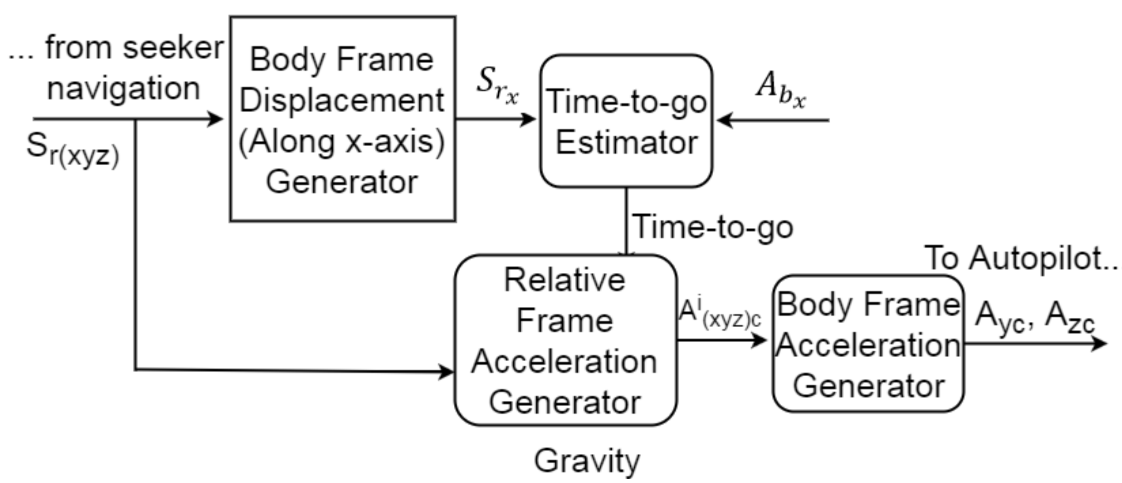

Figure 4.8: Optimal Control Theory Guidance

Body Frame Displacement Generator

The relative position and relative velocity along he missile X^v axis are transformed into the body frame using the vehicle reference to body transformation matrix, shown in Chapter 2.

$$S_x{}^b = T_{vb} S_x{}^v \tag{4.29}$$

$$V_x{}^b = T_{vb} V_x{}^v \tag{4.30}$$

Time-to-go Estimator

This outputs a guidance law time-to-go estimate that forces missile's axial acceleration command to be current acceleration.

$$A_{curr} = V_x{}^{b2} + A_x{}^b S_x{}^b \tag{4.31}$$

If the current acceleration is not zero, time-to-go, denoted as T_{go} is established as

$$T_{go} = \frac{2S_x{}^b}{\sqrt{A_{curr}} - V_x{}^b} \tag{4.32}$$

Otherwise, the time-to-go is

$$T_{go} = -\frac{S_x{}^b}{V_x{}^b} \tag{4.33}$$

Relative Frame Acceleration Generator

The inertial acceleration commands are computed from missile relative positions and velocities as follows

$$A_x{}^i = K_1 \frac{\left(\frac{S_{r_x}}{T_{go}} + V_{r_x}\right)}{T_{go}} \tag{4.34}$$

$$A_y{}^i = K_1 \frac{\left(\frac{S_{r_y}}{T_{go}} + V_{r_y}\right)}{T_{go}} \tag{4.35}$$

$$A_z{}^i = K_1 \frac{\left(\frac{S_{r_z}}{T_{go}} + V_{r_z}\right)}{T_{go}} - g \tag{4.36}$$

where, g is defined in Equation (2.47).

Body Frame Acceleration Generator

The vehicle reference to body transformation matrix is used to transform commanded inertial frame acceleration $(A_y{}^i, A_z{}^i)$ into missile body commands (A_{yc}, A_{zc}).

4.4.3 Differential Game Theory Guidance

In such formulations, a disturbance (e.g. Target Maneuver) "competes" with a control (e.g. missile acceleration command). The disturbance attempts to maximize a performace index (e.g. Miss Distance), while the control attempts to minimize the index [61]. Maneuver Index is a unit-less quantity which is used to quantify the degree of maneuverability of the target [19]. The differential game theory guidance is a variation of the optimal guidance. The commanded accelerations are first found from the optimal guidance algorithm, denote them (A_{yco}, A_{zco}). The differential game theory guidance commanded accelerations are then found using the following equations:

$$A_{yc} = \frac{A_{yco}}{1 - MI} \tag{4.37}$$

$$A_{zc} = \frac{A_{zco}}{1 - MI} \tag{4.38}$$

MI is defined as a Maneuver Index constant.

The missile autopilot receives the acceleration commands (A_{my}, A_{mz}) from the guidance system and converts them into fin deflection commands in order to steer the missile.

4.5 Summary and Conclusions

In this chapter, the seeker/navigation system dynamics were described. A block diagram of the seeker dynamics was given to show how the commanded seeker servo angles are generated. Also, the seeker model software algorithm was described. The missile guidance system, which processes range and range-rate infromation from the seeker/navigation system, was described. the guidance system generated commanded horizontal and vertical body accelerations to the autopilot. The three guidance algorithms available in this simulation were also discussed. Guidance algorithms included: proportional, optimal and differential game theory.

Chapter 5

TARGET MODELING

5.1 Introduction and Overview

This chapter gives a description of the three-degree-of-freedom target model and the three evasive maneuvers available to the target. For more on the target models used in this simulations, the reader is referred to [3] and [4]. For more on target modeling, in general, the reader is referred to [18], [52], [60].

A simple evasive three-degree-of-freedom target model is included in the program to test the missile's tracking and steering capabilities. The model used to describe the target dynamics is discussed in Section 5.2. The target can be made to maneuver with one of three methods. The target can fly straight with no evasive accelerations, use the Sheldon turn and climb methods [4] or use the Rigges Vergaz turn and dive method [3].

5.2 3DOF Target Dynamics

The target used in this study is modeled as a point mass with three degrees of freedom. The following vector differential equation is used to describe the target's response to acceleration commands:

$$\dot{A}_t = \frac{(A_{t_c} - A_t)}{\tau} \tag{5.1}$$

where

$A_t \stackrel{\text{def}}{=}$ actual target body acceleration, $(A_{t_x}, A_{t_y}, A_{t_z})^i$

$A_{t_c} \stackrel{\text{def}}{=}$ commanded acceleration, (A_{yc}, A_{zc})

$\tau \stackrel{\text{def}}{=}$ response time-constant and is $= 0.5$

The commanded acceleration, A_{t_c} is a function of: estimated time to go, and (sin, cos) of missile Euler (yaw, pitch, roll) angles. Commanded accelerations are restricted to a range representing the limits of a pilot's mental alertness, $\pm$9G's.

5.3 Straight Flight with No Maneuver.

The simplest option available to the target is to make it to fly in a straight path with constant velocity. No evasive maneuvers are generated to avoid the oncoming missile. The commanded target acceleration, $A_{t_c} = (A_{t_{xc}}, A_{t_{yc}}, A_{t_{zc}})^i$ is determined using the following algorithm:

Algorithm:

Compute the target inertial acceleration, A_{t_c} as:

$A_{t_{xc}} = 0$

$A_{t_{yc}} = 0$

$A_{t_{zc}} = 0$

5.4 Sheldon Turn & Climb Maneuver

The Sheldon Turn & Climb algorithm can be visualized as in Figure 5.1; viewed from the target toward the missile. For missile position in the right half of the target plane-of-view, the target will turn and climb right. If the missile lies in the left half of the target plane-of-view, the target will turn and climb left.

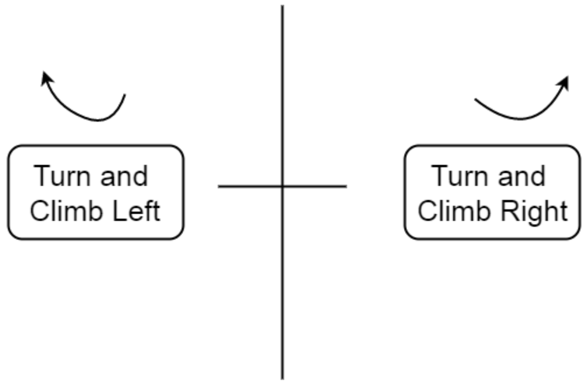

Figure 5.1: Sheldon Evasive Maneuver, Viewed from target-to-missile.

The commanded target inertial acceleration, A_{t_c} is calculated using time-to-go information from the missile autopilot, target Euler angles and the initial missile-target Aspect angle. The aspect angle measures angle from the inertial LOS vector to target velocity vector. The target euler angles are defined identically to the missile Euler angles in Chapter 2. For more on the Sheldon Turn & Climb algorithm, reader is referred to [4]. The commanded target acceleration A_{t_c} is determined using the following algorithm.

Algorithm:

1. Calculate sine and cosine of commanded target Euler roll angle ϕ based on estimated time-to-go and initial Aspect angle. Also, assign a value to A_{nt} the desired target normal acceleration. g_0 is defined in Chapter 2.

   If time-to-go > 1, then $\sin(\phi) = 0.707$ sign of $(\sin(Aspect))$, $\cos(\phi) = 0.707$ and $A_{nt} = 5\ g_0$

   Else $\sin(\phi) = 0$, $\cos(\phi) = 0.707$ and $A_{nt} = 9\ g_0$

2. Calculate target total body velocity as $V_t = \sqrt{[{V_{t_x}}^2 + {V_{t_y}}^2 + {V_{t_z}}^2]}$

3. Calculate target body velocity in X^iY^i plane as $V_{t_{xy}} = \sqrt{[{V_{t_x}}^2 + {V_{t_y}}^2]}$

4. Limit desired normal acceleration as a function of local air density (ρ) and V_t, so target angle of attack, α remains < 30 deg, i.e. $0.0 < A_t < 0.33\,\rho\,V_t$

5. Calculate sine and cosine target Euler pitch angle. $\sin(\theta) = \frac{-V_{t_z}}{V_t}$ and $\cos(\theta) = \frac{V_{t_{xy}}}{V_t}$

6. Calculate sine and cosine target Euler yaw angle. $\sin(\psi) = \frac{-V_{t_y}}{V_{t_{xy}}}$ and $\cos(\psi) = \frac{V_{t_x}}{V_{t_{xy}}}$

7. Compute target inertial acceleration, A_{t_c}, where g is as defined in Chapter 2.

   (a) $A_{t_{xc}} =$ -A_{nt} $(\cos(\phi)\ \sin(\theta)\ \cos(\psi) + \sin(\phi)\ \sin(\psi))$

   (b) $A_{t_{yc}} =$ -A_{nt} $(\cos(\phi)\ \sin(\theta)\ \sin(\psi) - \sin(\phi)\ \cos(\psi))$

   (c) $A_{t_{zc}} =$ -A_{nt} $(\cos(\phi)\ \cos(\theta))$ - g

5.5 Riggs Vergaz Turn & Dive Maneuver

The Riggs Vergaz Turn & Dive algorithm can be visualized as in Figure 5.2; viewed from target-to-missile. Missile may be spotted in one of the four quadrants. Missile positions in the bottom(top) two halves of the target plane-of-view result in the target turning and climbing(diving) right or left as indicated.

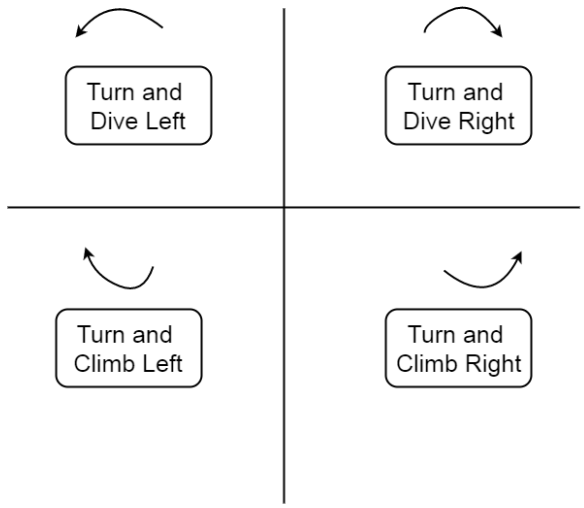

Figure 5.2: Riggs Vergaz Evasive Maneuver, Viewed from target-to-missile.

For more on the Riggs Vergaz Turn & Dive algorithm, reader is referred to [3]. The commanded target acceleration A_{t_c} is determined using the following algorithm.

Algorithm:

1. Calculate sine and cosine of commanded target Euler roll angle ϕ based on estimated time-to-go and initial Aspect angle. Also, assign a value to A_{nt} the desired target normal acceleration. g_0 is defined in Chapter 2.

   If time-to-go > 1, then $\sin(\phi) = 0.707$ sign of $(\sin(Aspect)), \cos(\phi) =$ -0.707 sign of $(\sin(Aspect))$ and $A_{nt} = 9\ g_0$

   Else $\sin(\phi) = 0$, $\cos(\phi) =$ -1 and $A_{nt} = 9\ g_0$

2. Calculate target total body velocity as $V_t = \sqrt{[{V_{t_x}}^2 + {V_{t_y}}^2 + {V_{t_z}}^2]}$

3. Calculate target body velocity in X^iY^i plane as $V_{t_{xy}} = \sqrt{[{V_{t_x}}^2 + {V_{t_y}}^2]}$

4. Limit desired normal acceleration as a function of local air density (ρ) and V_t, so target angle of attack, α remains < 30 deg, i.e. $0.0 < A_t < 0.33\,\rho\,V_t$

5. Calculate sine and cosine target Euler pitch angle, $\sin(\theta) = \frac{-V_{t_z}}{V_t}$ and $\cos(\theta) = \frac{V_{t_{xy}}}{V_t}$

6. Calculate sine and cosine target Euler yaw angle, $\sin(\psi) = \frac{-V_{t_y}}{V_{t_{xy}}}$ and $\cos(\psi) = \frac{V_{t_x}}{V_{t_{xy}}}$

7. Compute target inertial acceleration, A_{t_c}, where g is as defined in Chapter 2.

   (a) $A_{t_{xc}} =$ -A_{nt} $(\cos(\phi)\ \sin(\theta)\ \cos(\psi) + \sin(\phi)\ \sin(\psi))$

   (b) $A_{t_{yc}} =$ -A_{nt} $(\cos(\phi)\ \sin(\theta)\ \sin(\psi) - \sin(\phi)\ \cos(\psi))$

   (c) $A_{t_{zc}} =$ -A_{nt} $(\cos(\phi)\ \cos(\theta))$ - g

5.6 Summary and Conclusions

In this chapter, a 3 degree-of-freedom target model was described. The target model is included in the program to test the missile's tracking and steering capabilities. Three target maneuver algorithms were discussed. Each algorithm has been implemented in the simulation software. The target evasive maneuvers include: (1) Straight Flight with No Maneuver, (2) the Sheldon Turn and Climb maneuver and (3) the Riggs Vergaz Turn and Dive maneuver.

Chapter 6

BTT MISSILE AUTOPILOT

6.1 Introduction and Overview

In order to acheive adequate performance over the entire envelope of operating conditions, the autopilot of modern air-to-air tactical missile must be nonlinear [10]. Because of the inherent instabilities associated with missiles, stability augmentation systems are essential. The autopilot provides the added stability and ensures that accelerations from the guidance systems are properly followed. More precisely, the autopilot uses feedback to process the guidance commands and deliver appropriately coordinated fin commands to the actuators.

Throughout this research, a BTT missile has been considered. BTT missile control is made more difficult by the high roll rates required to achieve the short response time necessary for a high-performance missile. The high roll rates increase the aerodynamic coupling, which will be discussed here and can lead to inertial cross-coupling problems. The motivation for using BTT missile control is that the ramjet missile propulsion requires positive angles of attack and minimal sideslip angles, whch can be achieved by BTT missiles.

The bank-to-turn steering policy used in this simulation is sometimes referred to as *Preferred Orientation Control (POC)* [20]. In other words, it turns like an airplane. The EMRAAT missile is asymmetrical, see Figures 6.1 and 6.2, making the bank-to-turn steering policy particularly desirable. The propulsive performance of asymmetric missiles or missiles with off-axis air-breathing propulsion systems [20] may be adversely affected with certain angles of attack or sideslip. A bank-to-turn steering policy provides minimum sideslip angle.

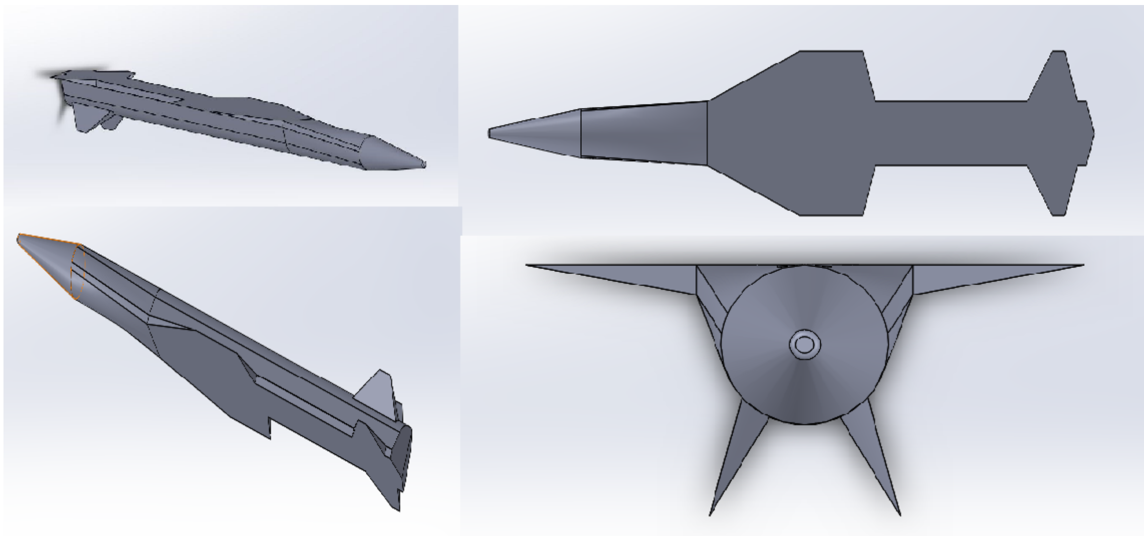

Figure 6.1: An Asymmetrical EMRAAT Missile

The guidance system sends horizontal and vertical acceleration commands to the autopilot. The commands are processed and converted into appropriately coordinated fin commands which are delivered to the actuators.

Body acceleration commands (A_{yc}, A_{zc}) generated by the guidance system are converted by the autopilot into commanded fin deflection angles F_{1c}, F_{2c}, F_{3c} and F_{4c}. The autopilot consists of following components:

1. Acceleration-Roll-Side-Slip Command Generator (BTT LOGIC)
2. Angular Rate Command Generator
3. Mixed Fin Command Generator: p-q-r-thrust/drag

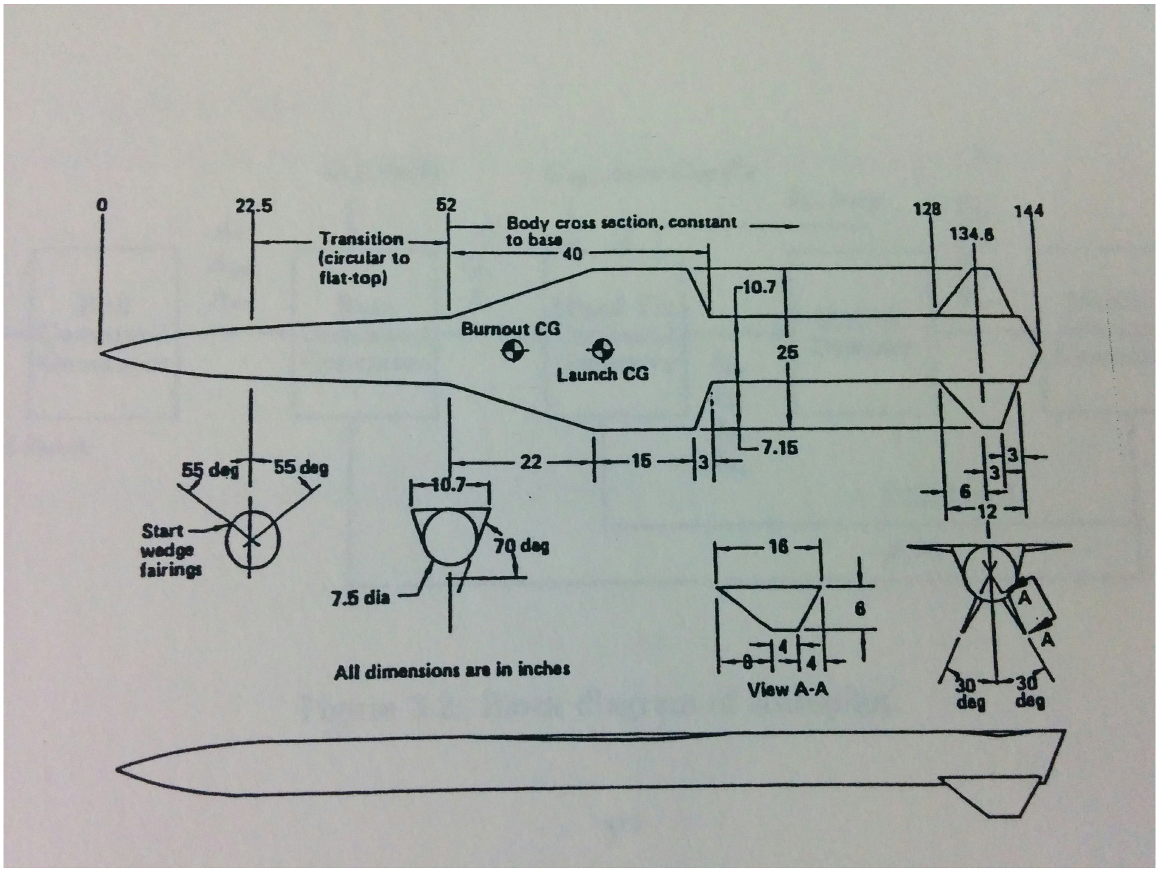

Figure 6.2: An Asymmetrical EMRAAT Missile Dimesions

4. ILAAT De-Mixer: Four Fin Force Commands to Actuators

5. ILAAT Mixer: Three Effective Aileron, Flapperon, Rudder Controls

Fundamentally, the autopilot is a nonlinear gain scheduled controller designed using classical control ideas. Two-loop autopilot structure is used here. Innermost loop is used to control the rate dynamics which are faster and outer loop controls the sideslip dynamics. BTT missiles ideally should have no Side-Slip. To achieve a desired orientation, the missile is rolled(banked) so that the plane of maximum aerodynamic normal force is oriented to the desired direction. Magnitude of the force is then controlled by adjusting the angle of attack in that plane. Figure 6.3 shows the information flow through the autopilot. Body acceleration commands (A_{yc}, A_{zc}) generated by the guidance system are converted by the autopilot into commanded fin deflection angles F_{1c}, F_{2c}, F_{3c} and F_{4c}. Four tail-mounted fins steer the missile. Effective roll, pitch and yaw deflection angles (δ_p, δ_q, δ_r) are algebraically related to the fin deflection angles. Each component of the autopilot is described in this chapter.

If we are primarily interested in controlling the missile
then when does a need for nonlinear controller arises?

So when we want our missile to operate over an entire envelope of flight conditions, the need for a nonlinear autopilot design arises. The gain of the controller should be scheduled as function of flight conditions for operating across the entire envelope of flight conditions. Thus the missile needs a nonlinear gain scheduled autopilot. There are several ways of obtaining a nonlinear controller and one best technique is to use *Incremental Nonlinear Dynamic Inversion*. The entire process is explained in detail in this chapter.

Remainder of this chapter is organized as follows: Section 6.2 discusses the nonlinear dynamic inversion using feedback linearization technique to obtain a nonlinear controller for the nonlinear missile plant. Also the design of controller gains as a function of flight condition(Gain Scheduling) is discussed in detail there. Then in Section 6.3, the BTT logic of designing for commanded bank angle for missile is discussed. Fair amount of information is also provided about the singularity problem that arises in the design and the ways to correct it. Section 6.4 explains how commanded angular rates are formed which serve as the reference for *Innermost Angular Rate Control Loop*. In the Section 6.5, the design for the commanded control deflections from the

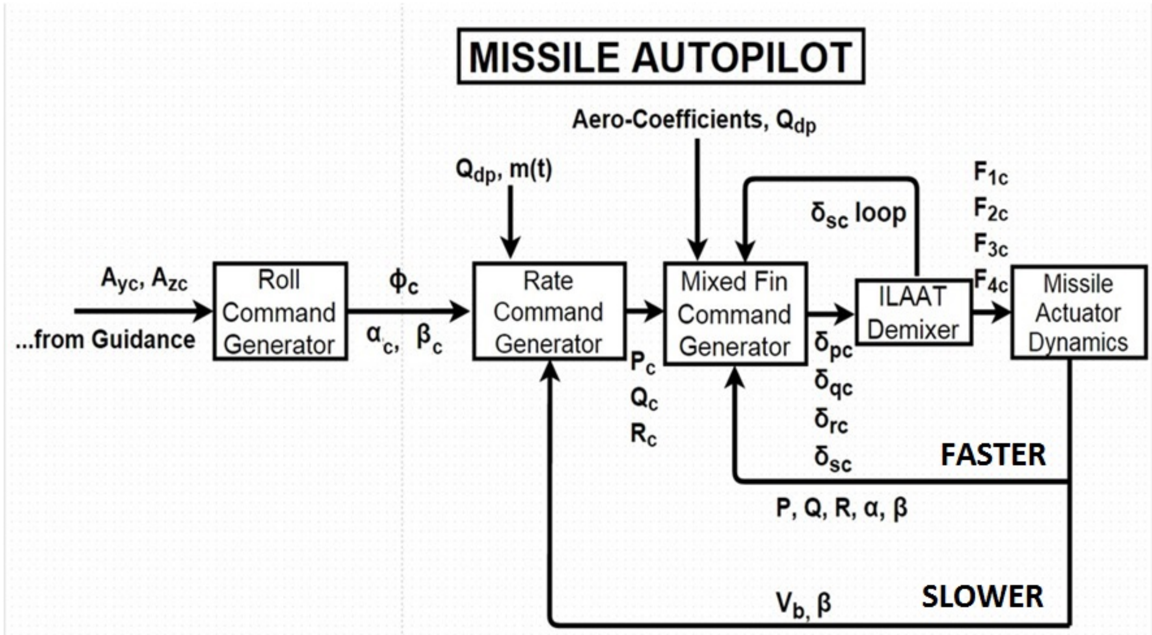

Figure 6.3: Block diagram of BTT Missile Autopilot

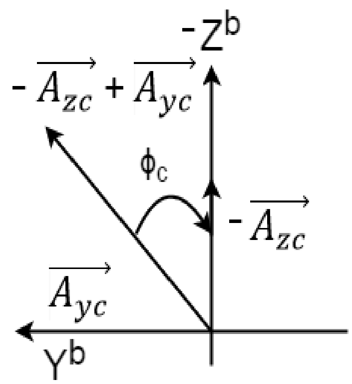

Figure 6.4: Determination of Commanded Roll Angle from A_{yc} & A_{zc}

angular rate information is discussed. Section 6.6 throws light on how the commanded control deflections combine together to form the commanded fin deflections. Section 6.7 explains how to realize the effective control deflections from fin deflections from the actuators. The simulation results using the nonlinear gain scheduled autopilot is presented in section 6.8. A comprehensive analysis of nonlinear autopilot is done in

Section 6.9 where the nonlinear autopilot is linearized and analyzed with the missile linear plant design obtained from Chapter 3. Autopilot is analyzed for its robustness and performance. Finally Section 6.10 concludes the work done in this chapter.

6.2 Control Law Formulation

The control law used in this research is obtained through Incremental Nonlinear Dynamic Inversion (INDI) using feedback linearization technique [69]. To apply NDI technique, it is required to know the full state of the system. If the state is not known, they can be approximated using nonlinear observer or stochastic state estimator as required. Also the system model has to be known completely to cancel the nonlinearities. If the system model is partially known, system identification process has to be done to get a full model knowledge. However system possessing RHP zeros (nonminimum phase systems) are not a good candidate for the application of NDI technique to obtain a nonlinear controller. The RHP zero-dynamics will result in a unstable controller while being inverted to cancel the nonlinearities. Missile acceleration control is a nonminimum phase problem. But when we assume symmetrical airframe by neglecting the inertial cross-coupling elements, it results in a minimum phase system and ready for NDI technique to be applied. The design of innermost autopilot rate control is done as following.

Let us recall the rotational dynamics of missile involving inertial and rate components as below

$$\begin{pmatrix} L \\ M \\ N \end{pmatrix}_{com} = J\dot{\omega} + \omega \times J\omega \tag{6.1}$$

where,

$$J\dot{\omega} + \omega \times J\omega = Q_{dp} S_{ref} L_{ref}(\xi + \chi u) + G \tag{6.2}$$

$$\omega = \begin{pmatrix} P \\ Q \\ R \end{pmatrix}, \mathrm{J} = \begin{pmatrix} \mathrm{I}_{xx} & -\mathrm{I}_{xy} & -\mathrm{I}_{xz} \\ -\mathrm{I}_{xy} & \mathrm{I}_{yy} & -\mathrm{I}_{yz} \\ -\mathrm{I}_{xz} & -\mathrm{I}_{yz} & \mathrm{I}_{zz} \end{pmatrix}, \mathrm{u} = \begin{pmatrix} \Delta\delta_{p_c} \\ \Delta\delta_{q_c} \\ \Delta\delta_{r_c} \end{pmatrix}, \chi = \begin{pmatrix} \mathrm{C}_{L_{\delta_p}} & 0 & 0 \\ 0 & \mathrm{C}_{M_{\delta_q}} & 0 \\ 0 & 0 & \mathrm{C}_{N_{\delta_r}} \end{pmatrix}$$

$\xi = \begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{act} = \begin{pmatrix} \mathrm{C}_{L_\beta}\beta + C_{L_p}\ \mathrm{P} \\ \mathrm{C}_{M_\alpha}\alpha + C_{M_q}\ \mathrm{Q} \\ \mathrm{C}_{N_\beta}\beta + C_{N_r}\ \mathrm{R} \end{pmatrix}, \mathrm{G} = \begin{pmatrix} 0 \\ G_{gy} \\ G_{gz} \end{pmatrix}$, assuming off-diagonal elements to be 0 in J matrix (due to axis symmetry), $\mathrm{J} = \begin{pmatrix} \mathrm{I}_{xx} & 0 & 0 \\ 0 & \mathrm{I}_{yy} & 0 \\ 0 & 0 & \mathrm{I}_{zz} \end{pmatrix}$. Also we know, $Q_{sl} = Q_{dp} S_{ref} L_{ref}$. Performing below INDI algebraic operations below,

$$\begin{pmatrix} L \\ M \\ N \end{pmatrix}_{com} = Q_{sl}(\xi + \chi u) + G \tag{6.3}$$

$$\begin{pmatrix} L \\ M \\ N \end{pmatrix}_{com} - G = Q_{sl}(\xi + \chi u) \tag{6.4}$$

$$\begin{pmatrix} \frac{L}{Q_{sl}} \\ \frac{M}{Q_{sl}} \\ \frac{N}{Q_{sl}} \end{pmatrix}_{com} - \begin{pmatrix} 0 \\ \frac{G_{gy}}{Q_{sl}} \\ \frac{G_{gz}}{Q_{sl}} \end{pmatrix} = \begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{com} - \begin{pmatrix} 0 \\ -\frac{S_{cx}F_{gz}}{Q_{sl}} \\ \frac{S_{cx}F_{gy}}{Q_{sl}} \end{pmatrix} = \xi + \chi u \tag{6.5}$$

$$\begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{com} - \begin{pmatrix} 0 \\ -\frac{S_{cx}A_{gz}Mass}{Q_{sl}} \\ \frac{S_{cx}A_{gy}Mass}{Q_{sl}} \end{pmatrix} = \xi + \chi u \tag{6.6}$$

$$\begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{com} - \begin{pmatrix} 0 \\ -\frac{S_{cx}A_{gz}Mass}{Q_{sl}} \\ \frac{S_{cx}A_{gy}Mass}{Q_{sl}} \end{pmatrix} - \xi = \begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{com} - \begin{pmatrix} 0 \\ -\frac{S_{cx}A_{gz}Mass}{Q_{sl}} \\ \frac{S_{cx}A_{gy}Mass}{Q_{sl}} \end{pmatrix} - \begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{act} = \chi u \tag{6.7}$$

$$u = \chi^{-1} \left(\begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{com} - \begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{act} - \begin{pmatrix} 0 \\ -\frac{S_{cx}A_{gz}Mass}{Q_{sl}} \\ \frac{S_{cx}A_{gy}Mass}{Q_{sl}} \end{pmatrix} \right) \tag{6.8}$$

$$\begin{pmatrix} \Delta\delta_{p_c} \\ \Delta\delta_{q_c} \\ \Delta\delta_{r_c} \end{pmatrix} = \begin{pmatrix} C_{L_{\delta_p}} & 0 & 0 \\ 0 & C_{M_{\delta_q}} & 0 \\ 0 & 0 & C_{N_{\delta_r}} \end{pmatrix}^{-1} \left(\begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{com} - \begin{pmatrix} C_L \\ C_M \\ C_N \end{pmatrix}_{act} - \begin{pmatrix} 0 \\ -\frac{S_{cx}A_{gz}Mass}{Q_{sl}} \\ \frac{S_{cx}A_{gy}Mass}{Q_{sl}} \end{pmatrix} \right) \tag{6.9}$$

The main idea of nonlinear dynamic inversion is to cancel the nonlinearities in the system and use classical control theory ideas to control the resulting linear system.

So, the below 2 questions are very intuitive to ask.

1. How does the resulting linear system look like?

2. Is it still dependent upon flight conditions?

The following calculation will answer the above questions. Considering the 1^{st} channel and referring to the equation 3.29, the following analysis can be made. Similar procedures can be utilized for understanding the other 2 channels.

$$\dot{P} = \frac{L}{I_{xx}^*} = \frac{C_L Q_{dp} S_{ref} L_{ref}}{I_{xx}^*} = \frac{Q_{dp} S_{ref} L_{ref}}{I_{xx}^*} (C_{L_{\delta_p}} \delta_p + C_{L_P} L_{2V} P + C_{L_\beta} \beta) \tag{6.10}$$

Rearranging above terms and writing as below,

$$\dot{P} = \frac{Q_{dp} S_{ref} L_{ref}}{I_{xx}^*} \left(C_{L_P} L_{2V} P + C_{L_\beta} \beta \right) + \frac{Q_{dp} S_{ref} L_{ref}}{I_{xx}^*} \left(C_{L_{\delta_p}} \delta_p \right) \tag{6.11}$$

This now resembles standard nonlinear state equation as shown below,

$$\dot{P} = f(P, \beta) + g(x)\delta_p, \text{ which looks like, } \dot{x} = f(x) + g(x)u \tag{6.12}$$

Substituting the control law obtained above from nonlinear dynamic inversion, we get

$$\dot{P} = \frac{Q_{dp} S_{ref} L_{ref}}{I_{xx}^*} \left(C_{L_{\delta_p}} \left(\frac{K_4 (P_c - P)}{Q_{dp}} + \frac{C_{L_\beta}}{C_{L_{\delta_p}}} (\beta_c - \beta) \right) + C_{L_P} L_{2V} P + C_{L_\beta} \beta \right) \tag{6.13}$$

Rearranging above terms writing in terms of varying coefficients & remembering that β_c is always set to zero, we get,

$$\dot{P} = -g_1 \, P + g_2 \, P_c \tag{6.14}$$

This looks like the following equation, with states, x = [P] and reference, r = [P_c]

$$\dot{x} = -A(\sigma)\,x + B(\sigma)\,r \tag{6.15}$$

6.2.1 Gain Scheduling of Linear Parameter Varying System

The equation 6.14 looks like a linear parameter varying system, where the "A" & "B" matrices depend upon the flight conditions such as α, β, Mach & Q_{dp} which are collectively represented by σ scheduling variable. In designing feedback controllers for dynamical systems, the controllers are often designed at various operating points using linearized models of the system dynamics and are scheduled as a function of a parameter or parameters for operation at intermediate conditions [74]. It is an approach for the control of nonlinear systems that uses a family of linear controllers, each of which provides satisfactory control for a different operating point of the system. One or more observable variables, called the scheduling variables, are used to determine the current operating region of the system and to enable the appropriate linear controller. Here in case of BTT missile control, a set of controllers are designed at different gridded locations of corresponding parameters such as α, β, Mach & δ_q. In brief, gain scheduling is a control design approach that constructs a nonlinear controller for a nonlinear plant by patching together a collection of linear controllers. These linear controllers are blended in real-time via interpolation in our case through the use of look up tables. Though the stability is not guaranteed at operating conditions other than the design points, it is a very efficient technique where the parameter dependency of controllers are large due to increased operating envelopes with more demanding performance requirements.

Thus referring to 6.14, the system matrix depends upon α, Mach & Q_{dp}. Essentially we are looking upon the following pole caused by the A().

$$\frac{1}{(s + \frac{Q_{sl} K_4}{I_{xx} Q_{dp}} - \frac{C_{L_P} L_{ref} V_b}{2})} = \frac{1}{(s + \lambda)}$$

- As Mach increases, system becomes bigger as RHP pole & RHP zero increase in magnitude. This can be seen easily by inspecting the λ parameter. Thus autopilot gets aggressive as mach increases to stabilize the big unstable pole.

- As altitude increases, system becomes smaller as RHP pole & RHP zero decrease in magnitude. Thus autopilot gets sluggish as altitude increases to stabilize the small unstable pole.

6.3 BTT Logic

The guidance system acceleration commands, (A_{yc}, A_{zc}) are initially used to form a commanded bank angle (ϕ_c), commanded angle of attack (α_c) [25] and commanded sideslip angle (β_c) as follows:

$$\phi_c = \tan^{-1}\left(\frac{A_{yc}}{-A_{zc}}\right) \tag{6.16}$$

$$\alpha_c = \frac{\left(\|a_c\| \frac{M}{Q_{dp} S_{ref}}\right) - |(C_z - C_{z_\alpha} \alpha)|}{|C_{z_\alpha}|} \tag{6.17}$$

$$\beta_c = 0 \tag{6.18}$$

where $a_c = [A_{yc}\ A_{zc}]$ and M is the mass.

Singularity Problem.

For $|A_{yc}| < 35$ and $|A_{zc}| < 40$, ϕ_c is set equal to zero. This provides a noise threshold to prevent roll commands whenever commanded body accelerations are too small. If $A_{zc} = 0$, ϕ_c is set $\pm\frac{\Pi}{2}$ depending on the sign of the actual body roll rate P. This is set to avoid the singularity problem that arises if A_{zc} in equation 6.16 goes to 0 and thus arctangent function becomes infinity in both directions.

6.4 Angular Rate Command Generator

Rotation rate commands (P_c, Q_c, R_c) are formed from A_{yc}, A_{zc}, ϕ_c, β, dynamic pressure, missile velocity and missile mass using below equations. P_c and R_c are selected to be proportional to ϕ and β respectively. A_{zc} is limited, denoted as A_{zcL}, so that the magnitude of pitch acceleration command (Q_c) has a maximum value near $\alpha = 28$ degrees. The maximum acceleration command, denoted by $A_{mz_{max}}$ is calculated form the current dynamic pressure Q_{dp} and missile mass, m(t) as follows:

$$A_{mz_{max}} = 5.25 \frac{Q_{dp}}{m(t)} \tag{6.19}$$

If $|A_{zc}| \quad > A_{mz_{max}}$, then A_{zcL} is set to $A_{mz_{max}}$, else it is left equal to A_{zc}. Below equations compute the rotation rate commands. The autopilot gains are summarized in Table 6.1.

$$P_c = K_1(\phi_c - \phi) \tag{6.20}$$

$$Q_c = K_2 \frac{(A_{zcL} - {A_{mz}}^b)}{Q_{dp}} - \frac{A_{zcL}}{V_b} \tag{6.21}$$

$$R_c = K_3(\beta_c - \beta) \tag{6.22}$$

It is to be noted here that, ${A_{mz}}^b = Q_{sm}\, C_z = Q_{sm}\, (C_{N_\alpha}(\alpha_c - \alpha) + C_{N_{\delta_q}} \delta_q)$

K_1	7
K_2	-10
K_3	0.5
K_4	500
K_5	-1.75
K_6	-1500
K_7	-5000

Table 6.1: Autopilot Gains

6.5 Mixed Fin Command Generator: p-q-r-thrust/drag

The commanded effective fin deflections (δ_{p_c}, δ_{q_c}, δ_{r_c}) model an ideal set of physical fin deflections which cause the missile to roll, pitch and yaw about its body axes [20]. The effective squeeze mode δ_s represents a *squeeze or speed-brake mode*, which is used to minimize the axial drag, i.e. no preferred roll, pitch or yaw is induced [20].

After generation of the rate commands (P_c, Q_c, R_c), these are used along with the true body rotation rates (P, Q, R), α, β, Q_{dp}, missile mass as well as a few of the aerodynamic coefficients to generate effective aileron, elevator and roll commands (δ_{p_c}, δ_{q_c}, δ_{r_c}) [1] via the following nonlinear control law.

$$\delta_{p_c} = \frac{K_4(P_c - P)}{Q_{dp}} + \frac{C_{L_\beta}}{C_{L_{\delta_p}}}(\beta_c - \beta) \tag{6.23}$$

$$\delta_{q_c} = (K_5 + \frac{K_6}{Q_{dp}})(Q_c - Q) + \frac{C_{M_\alpha}}{C_{M_{\delta_q}}}(\alpha_c - \alpha) + \frac{A_{gz}S_{cx}Mass}{Q_{sl}C_{M_{\delta_q}}} \tag{6.24}$$

$$\delta_{r_c} = \frac{K_7(R_c - R)}{Q_{dp}} + \frac{C_{N_\beta}}{C_{N_{\delta_r}}}(\beta_c - \beta) + \frac{30PQ - A_{gy}S_{cx}Mass}{Q_{sl}C_{N_{\delta_r}}} \tag{6.25}$$

[1] As we would do for an aircraft

where,

$$Q_{sl} = Q_{dp} S_{ref} L_{ref} \tag{6.26}$$

F_{g_y} and F_{g_z} are gravitational accelerations in the body frame and Q_{dp}, S_{ref} and L_{ref} are as defined in the Chapter 2. The gains K_4, K_5, K_6 and K_7 are given in Table 6.1. Also produced is a squeeze mode command δ_{s_c} formed by taking above linear combination of previous values of (F_{1C}, F_{2C}, F_{3C} and F_{4C}) using [2]:

$$\delta_{s_c} = 0.25(F_{1C} - F_{2C} - F_{3C} + F_{4C}) \tag{6.27}$$

6.6 ILAAT De-Mixer: Four Fin Force Commands to Actuators

Finally, effective fin deflection commands (δ_{p_c}, δ_{q_c}, δ_{r_c}) and the effective squeeze mode δ_s are transformed algebraically into true fin deflection commands (F_1, F_2, F_3, F_4) using below ILAAT (*Integrated Logic for Air-to-Air Technology*) combination logic [48] below. The BTT missile used here uses the "×" delta configuration ILAAT mixing logic as below,

$$\begin{bmatrix} F_{1C} \\ F_{2C} \\ F_{3C} \\ F_{4C} \end{bmatrix} = \begin{bmatrix} -1 & +1 & -1 & +1 \\ -1 & +1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 \end{bmatrix} \begin{bmatrix} \delta_{p_c} \\ \delta_{q_c} \\ \delta_{r_c} \\ \delta_{s_c} \end{bmatrix} \tag{6.28}$$

6.7 ILAAT Mixer: 3 Effective Aileron, Flapperon, Rudder Controls

Finally, the effective control deflections (δ_p, δ_q, δ_r) i.e. aileron, flapperon and rudder can be realized using ILAAT mixing combination logic as follows,

[2] Here, the current value of δ_{s_c} is found by the previous values of (F_{1C} - F_{2C} - F_{3C} + F_{4C})

$$\begin{pmatrix} \Delta\delta_p \\ \Delta\delta_q \\ \Delta\delta_r \\ \Delta\delta_s \end{pmatrix} = 0.25 \begin{pmatrix} -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \Delta F_1 \\ \Delta F_2 \\ \Delta F_3 \\ \Delta F_4 \end{pmatrix} \tag{6.29}$$

The above matrix is the inverse of the matrix in the equation ()6.28.

6.8 Nonlinear Autopilot Simulation Results

Table 6.2 shows the flight conditions considered for evaluating the performance of the new improved nonlinear autopilot design.

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Missile Guidance	Optimal Control
Target Maneuver	Sheldon	Aspect Angle	0 deg

Table 6.2: Flight Conditions for Nonlinear Autopilot Simulations

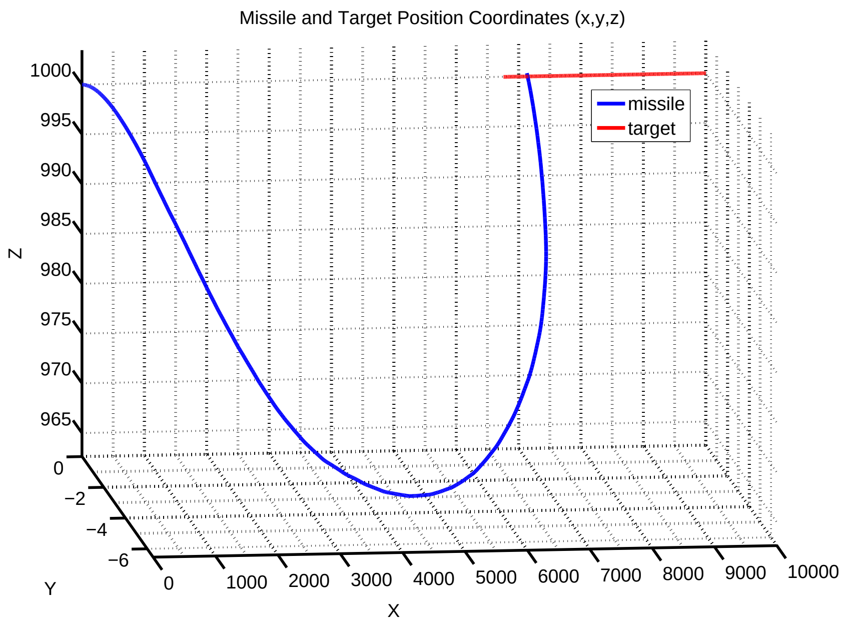

Figure 6.5: Post Flight Analysis - Missile Target Engagement

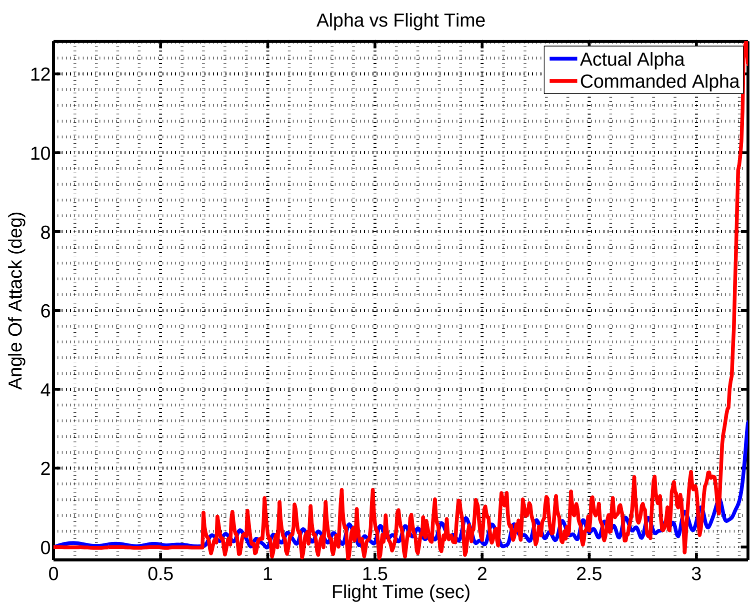

Figure 6.6: Post Flight Analysis - α Profile

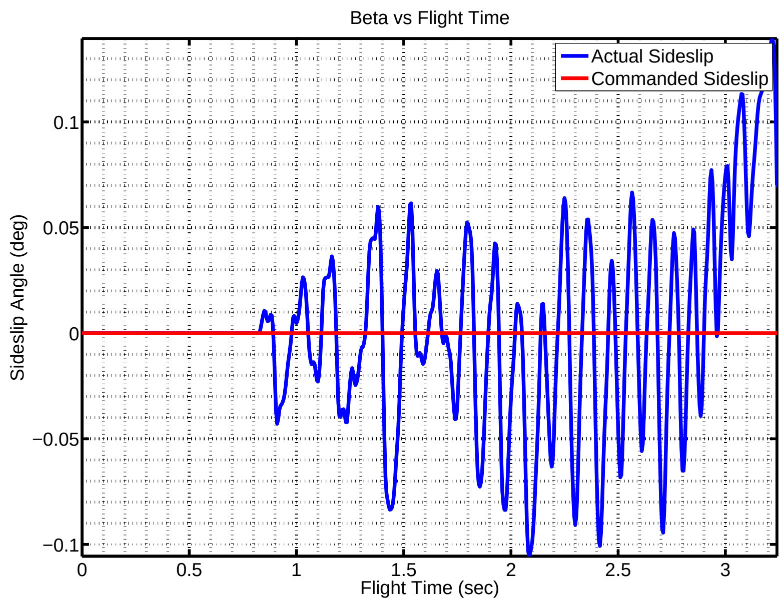

Figure 6.7: Post Flight Analysis - β Profile

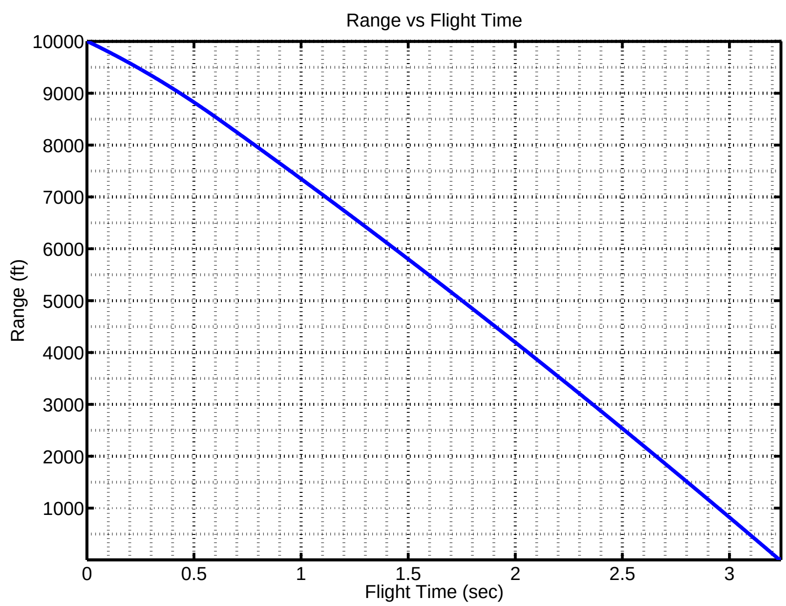

Figure 6.8: Post Flight Analysis - Range Profile

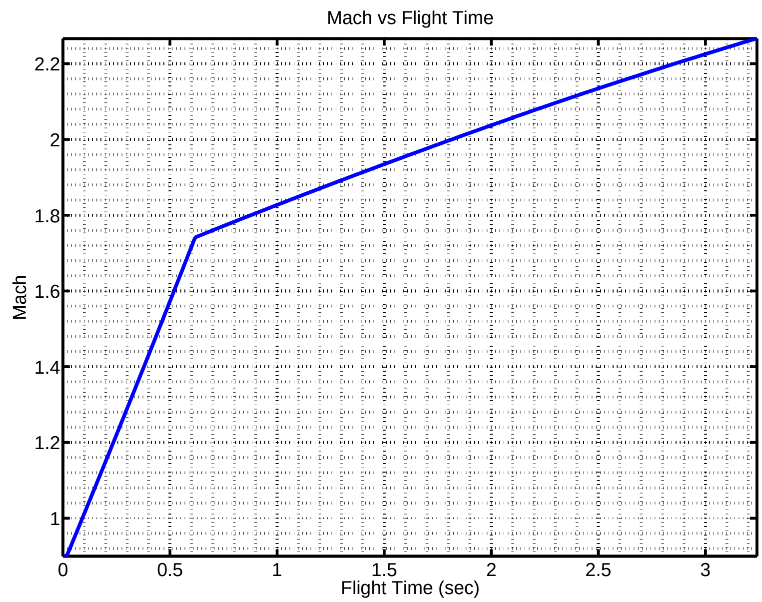

Figure 6.9: Post Flight Analysis - Mach Profile

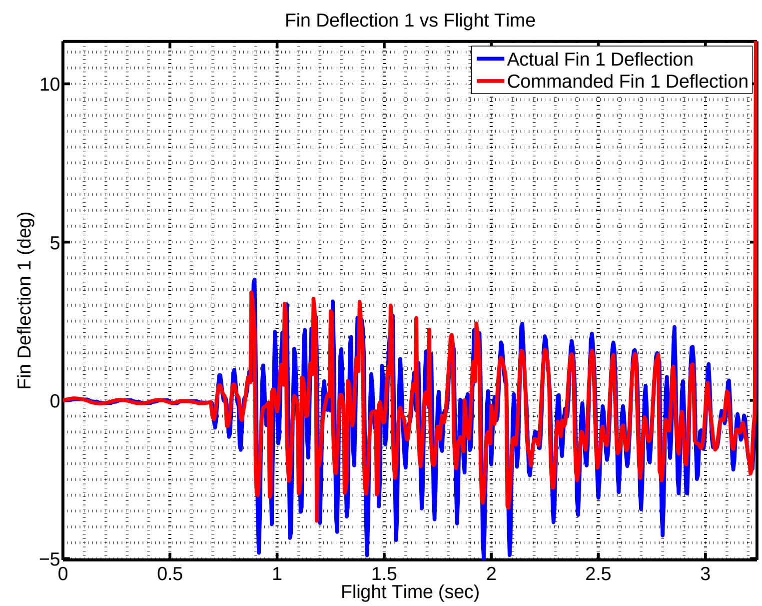

Figure 6.10: Post Flight Analysis - Fin 1 Deflection Profile

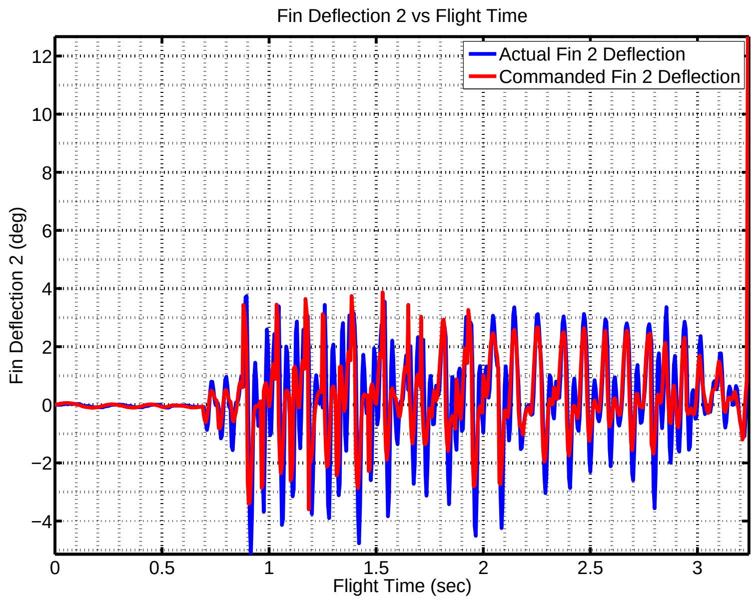

Figure 6.11: Post Flight Analysis - Fin 2 Deflection Profile

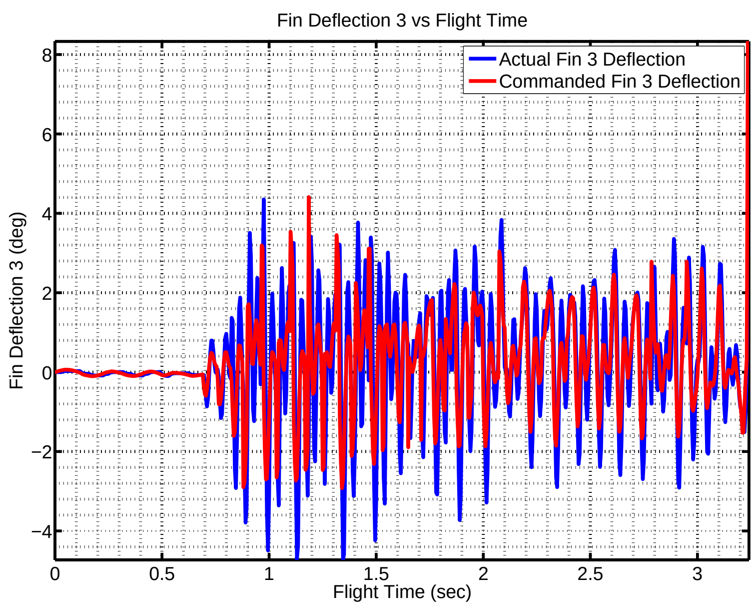

Figure 6.12: Post Flight Analysis - Fin 3 Deflection Profile

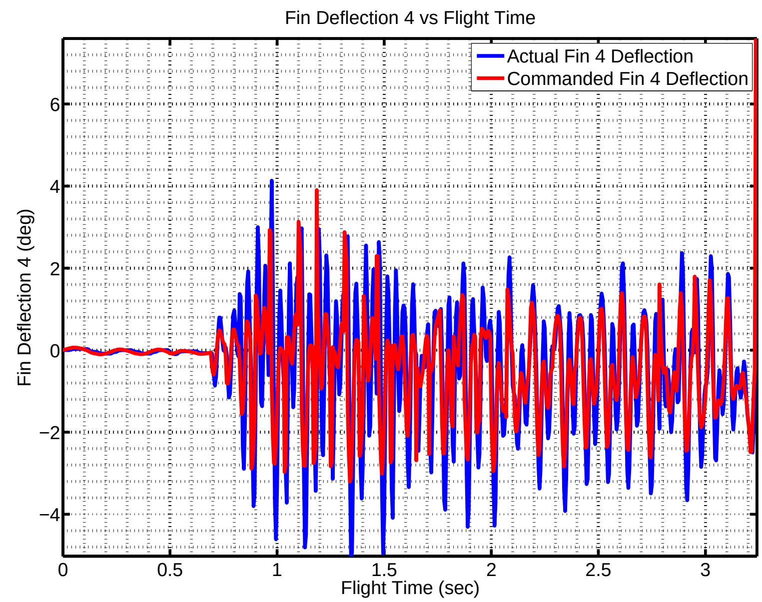

Figure 6.13: Post Flight Analysis - Fin 4 Deflection Profile

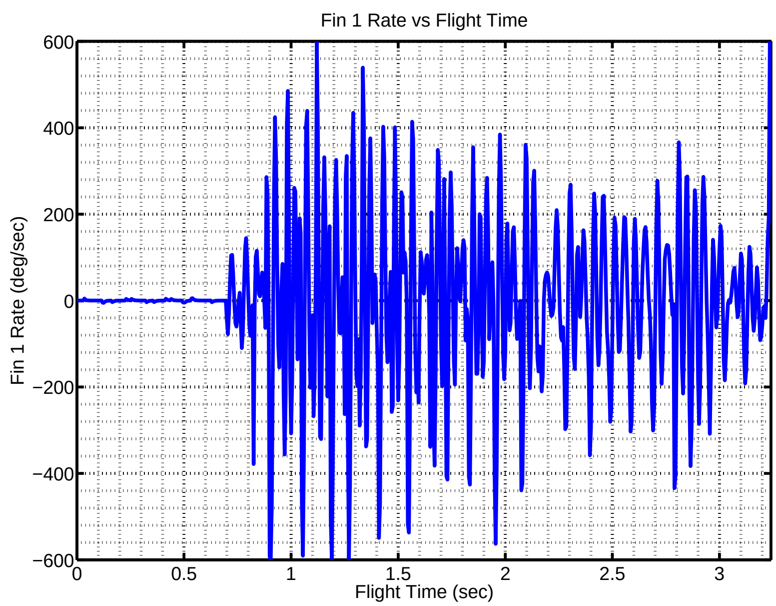

Figure 6.14: Post Flight Analysis - Fin 1 Rate Profile

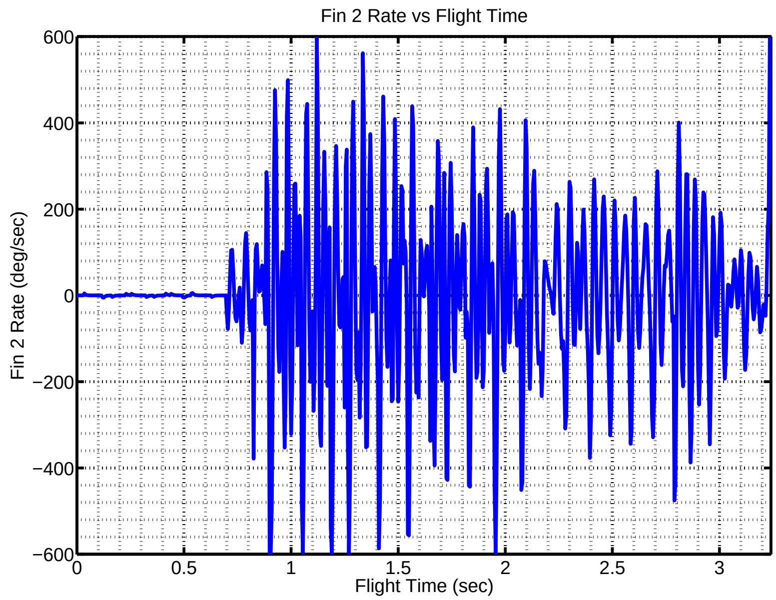

Figure 6.15: Post Flight Analysis - Fin 2 Rate Profile

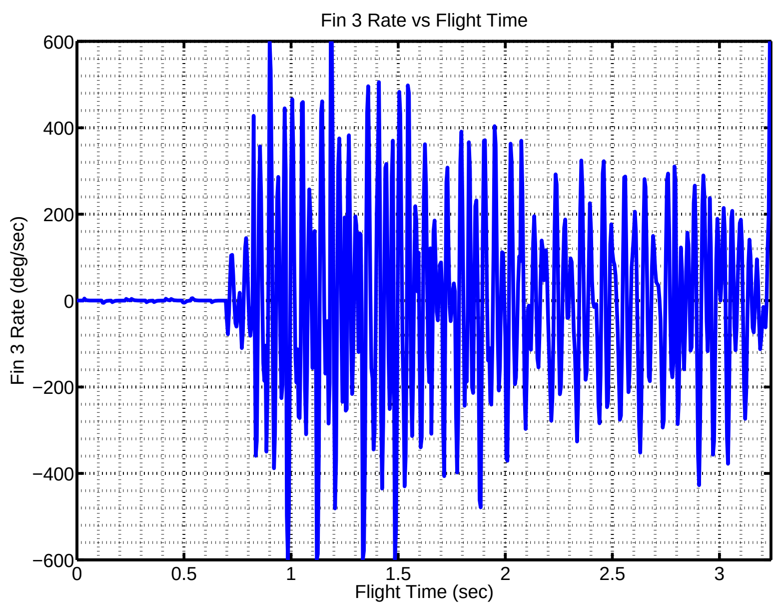

Figure 6.16: Post Flight Analysis - Fin 3 Rate Profile

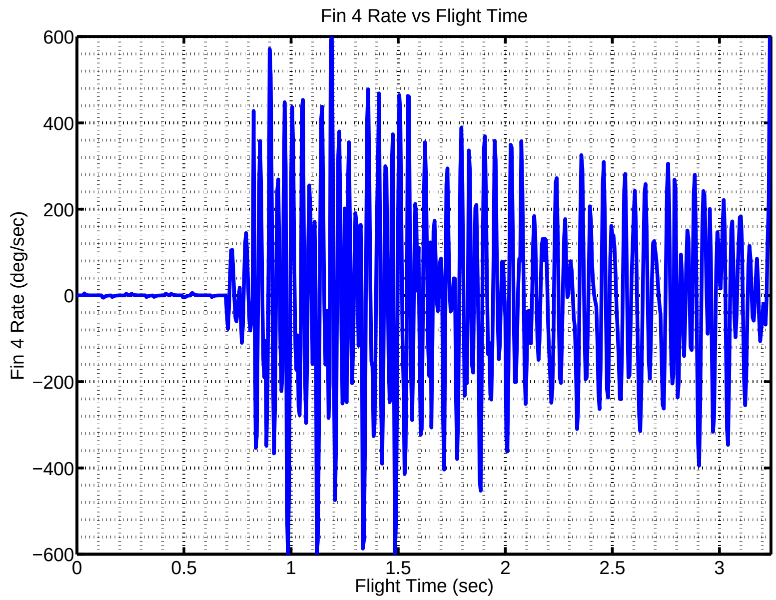

Figure 6.17: Post Flight Analysis - Fin 4 Rate Profile

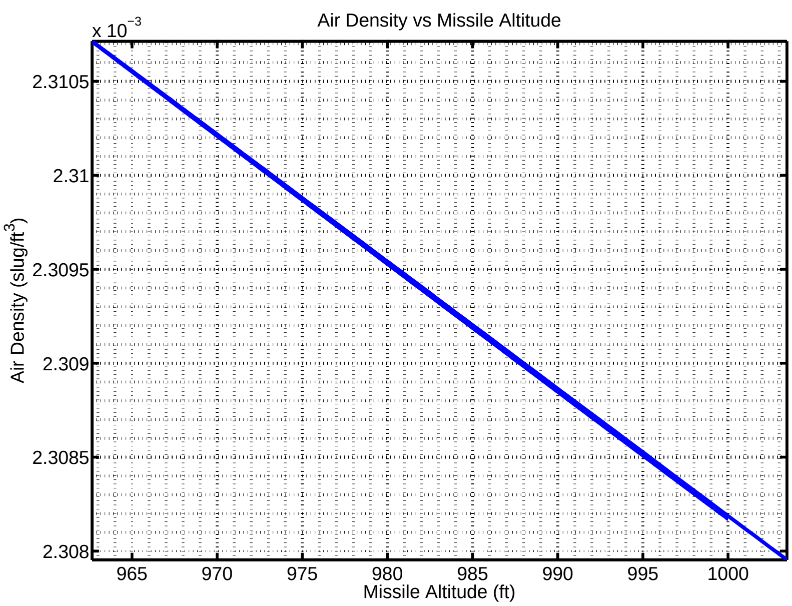

Figure 6.18: Post Flight Analysis - Air Density Profile

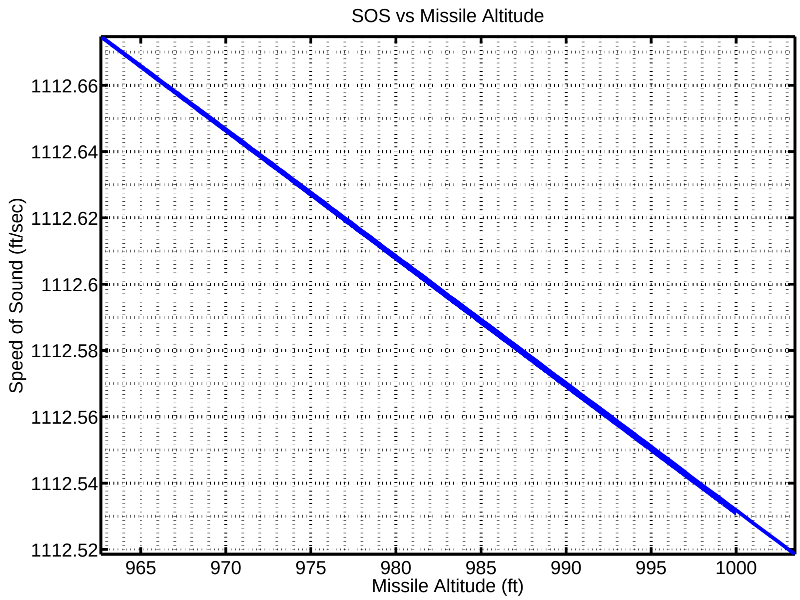

Figure 6.19: Post Flight Analysis - SOS Profile

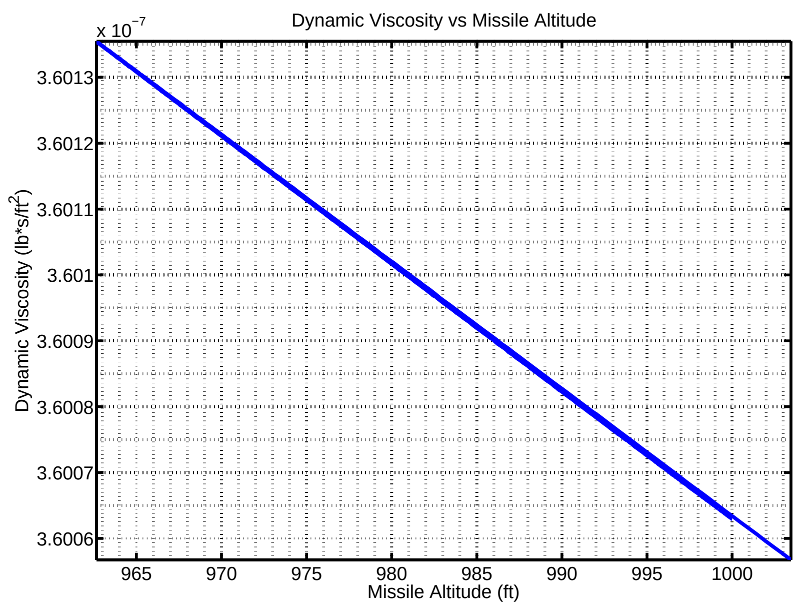

Figure 6.20: Post Flight Analysis - Dynamic Viscosity Profile

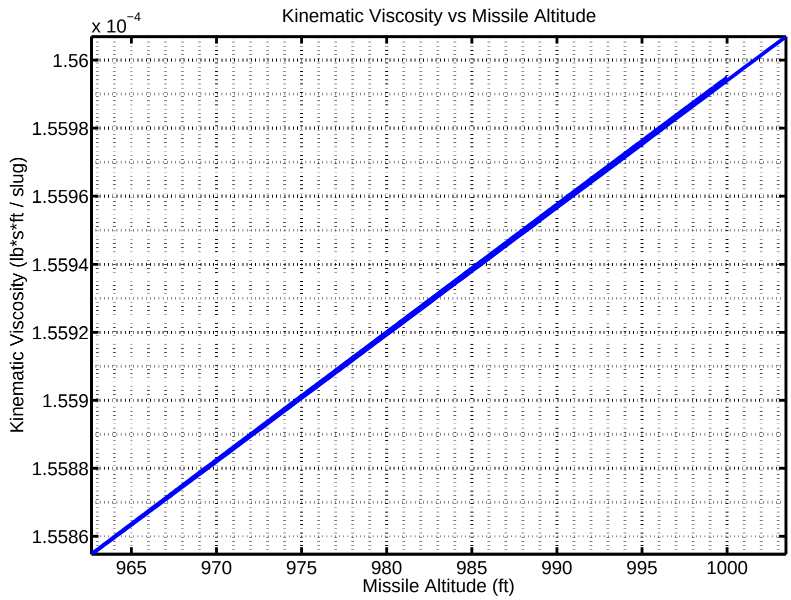

Figure 6.21: Post Flight Analysis - Kinematic Viscosity Profile

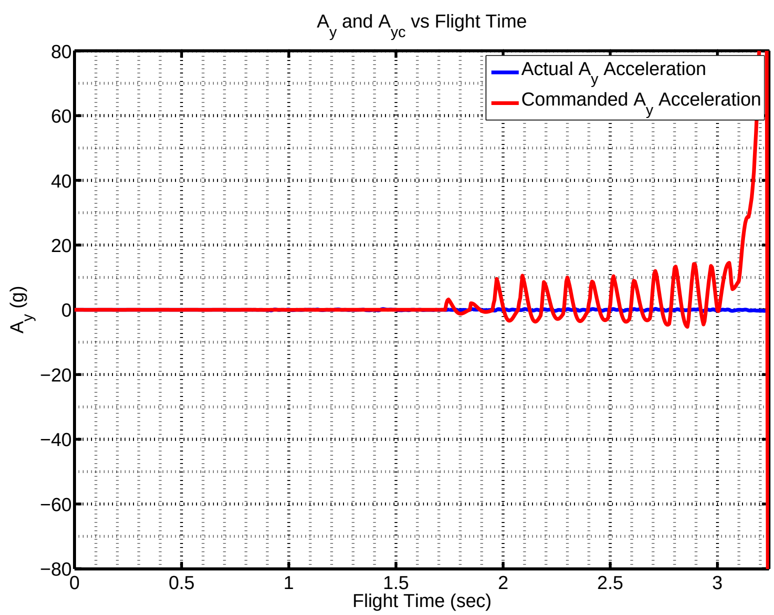

Figure 6.22: Post Flight Analysis - Acceleration in Y Direction Profile

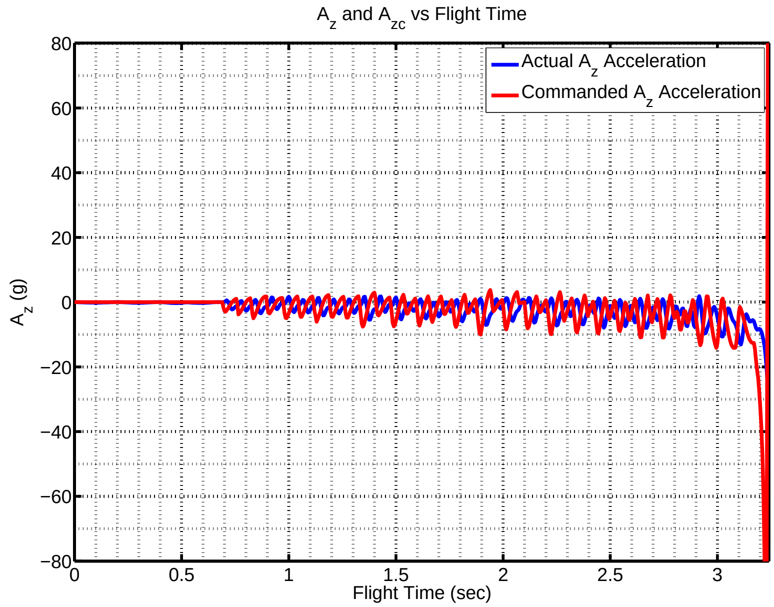

Figure 6.23: Post Flight Analysis - Acceleration in Z Direction Profile

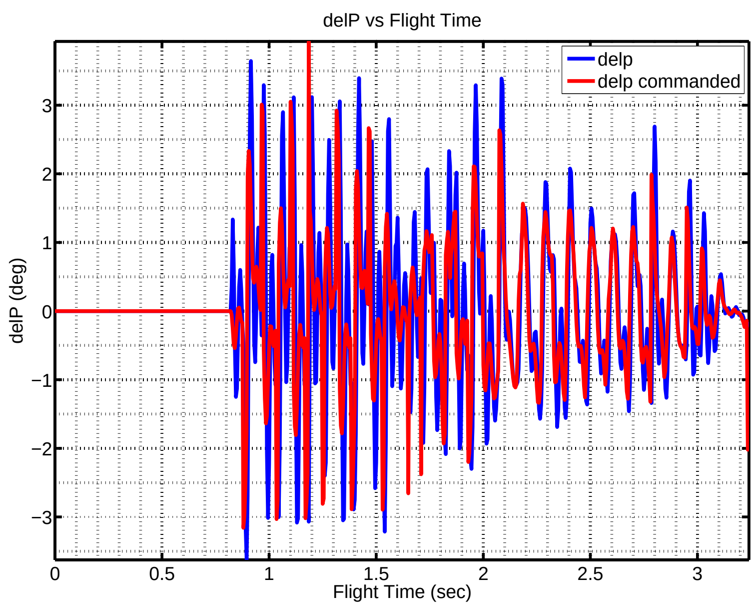

Figure 6.24: Post Flight Analysis - Aileron Profile

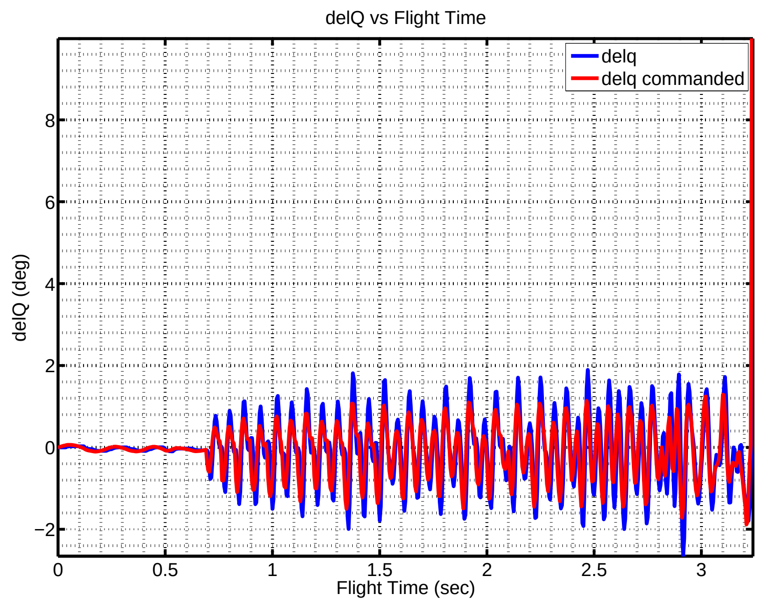

Figure 6.25: Post Flight Analysis - Elevator Profile

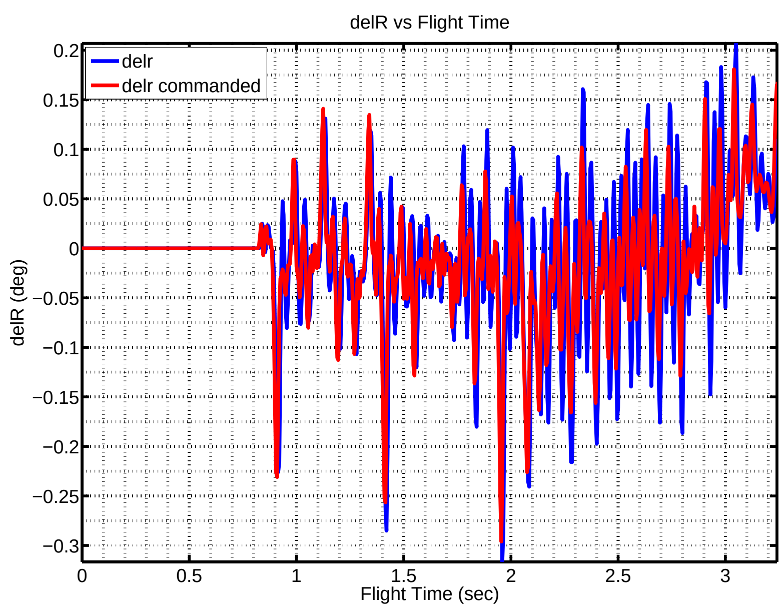

Figure 6.26: Post Flight Analysis - Rudder Profile

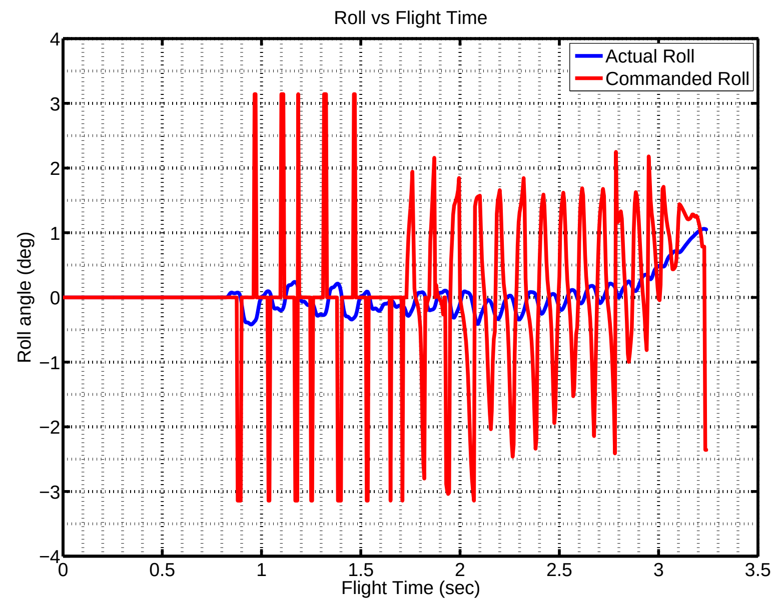

Figure 6.27: Post Flight Analysis - Roll Angle Profile

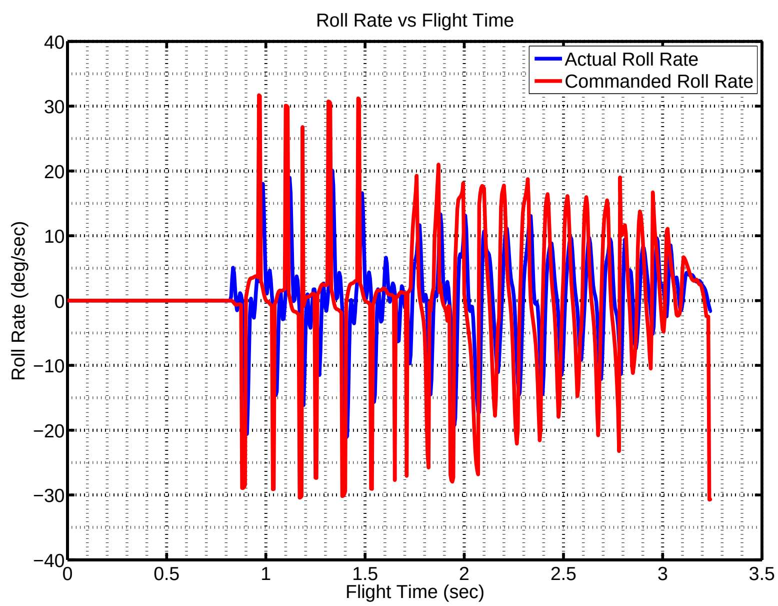

Figure 6.28: Post Flight Analysis - Role Rate Profile

6.9 Autopilot Linearization

Linearizing the autopilot around a flight condition will definitely give an idea about the range where the linear autopilot can approximate the nonlinear autopilot.

6.9.1 Assumptions about Steady Flight Conditions

The following assumptions are made for linearizing the autopilot routines.

1. The steady trimmed flight condition is one of uniform translational motion, i.e., where the equilibrium angular rates are zero. Thus $P^* = Q^* = R^* = 0$.

2. One of the requirements of the BTT missile autopilot is to minimize the sideslip angle during flight. Thus, $V^* = 0$.

3. The bank angle, ϕ and the yaw angle, ψ are taken to be zero.

6.9.2 Innermost Loop

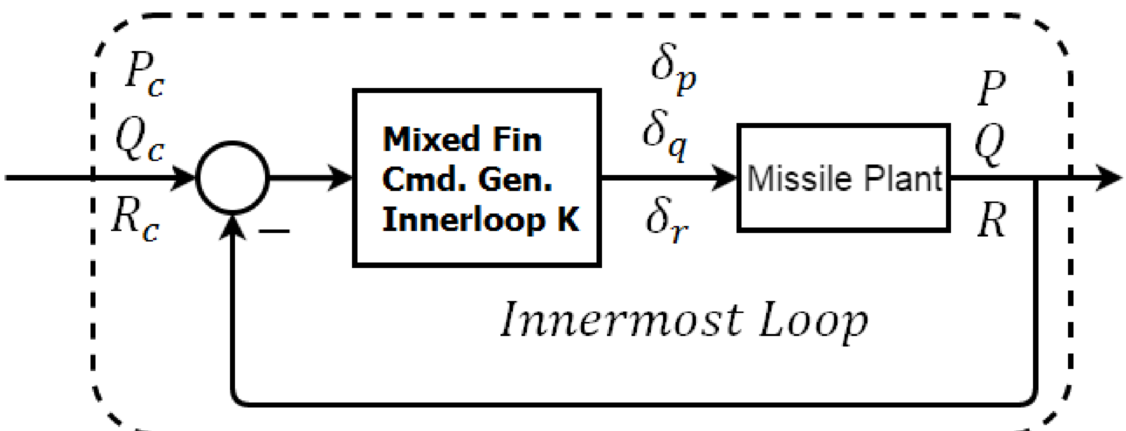

Figure 6.29: Block Diagram of Autopilot Innermost Loop

Recalling the rate control loop equations,

$$\delta_{p_c} = \frac{K_4(P_c - P)}{Q_{dp}} + \frac{C_{L_\beta}}{C_{L_{\delta_p}}}(\beta_c - \beta) \tag{6.30}$$

$$\delta_{q_c} = (K_5 + \frac{K_6}{Q_{dp}})(Q_c - Q) + \frac{C_{M_\alpha}}{C_{M_{\delta_q}}}(\alpha_c - \alpha) + \frac{A_{gz}S_{cx}Mass}{Q_{sl}C_{M_{\delta_q}}} \quad (6.31)$$

$$\delta_{r_c} = \frac{K_7(R_c - R)}{Q_{dp}} + \frac{C_{N_\beta}}{C_{N_{\delta_r}}}(\beta_c - \beta) + \frac{30PQ - A_{gy}S_{cx}Mass}{Q_{sl}C_{N_{\delta_r}}} \quad (6.32)$$

$$\delta_{s_c} = 0.25(F_{1C} - F_{2C} - F_{3C} + F_{4C}) \quad (6.33)$$

The above equations (6.30 - 6.32) can be written in terms of perturbed small scall error signals as follows

$$\Delta\delta_{p_c} = a_1\Delta e_p + a_2\Delta e_\beta \quad (6.34)$$

$$\Delta\delta_{q_c} = a_3\Delta e_q + a_4\Delta e_\alpha \quad (6.35)$$

$$\Delta\delta_{r_c} = a_5\Delta e_r + a_6\Delta e_\beta \quad (6.36)$$

$$\Delta\delta_{s_c} = 0 \quad (6.37)$$

where δ_{s_c} is a constant and so its perturned value $\Delta\delta_{s_c}$ vanishes. Also $\Delta e_p \stackrel{\text{def}}{=} P_c - P$, $\Delta e_q \stackrel{\text{def}}{=} Q_c - Q$ and $\Delta e_r \stackrel{\text{def}}{=} R_c - R$ and $a_1 = \frac{K_4}{Q_{dp}}$, $a_2 = \frac{C_{L_\beta}}{C_{L_{\delta_p}}}$, $a_3 = \left(K_5 + \frac{K_6}{Q_{dp}}\right)$, $a_4 = \frac{C_{M_\alpha}}{C_{M_{\delta_q}}}$, $a_5 = \frac{K_7}{Q_{dp}}$, $a_6 = \frac{C_{N_\beta}}{C_{N_{\delta_r}}}$

Writing above equations in matrix form.

$$\begin{bmatrix} \Delta\delta_{p_c} \\ \Delta\delta_{q_c} \\ \Delta\delta_{r_c} \\ \Delta\delta_{s_c} \end{bmatrix} = \begin{bmatrix} a_1 & 0 & 0 & 0 & a_2 \\ 0 & a_3 & 0 & a_4 & 0 \\ 0 & 0 & a_5 & 0 & a_6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta e_p \\ \Delta e_q \\ \Delta e_r \\ \Delta e_\alpha \\ \Delta e_\beta \end{bmatrix} \quad (6.38)$$

combining this with the ILAAT de-mixer, we get fin commands as follows

$$\begin{bmatrix} \Delta F_{1C} \\ \Delta F_{2C} \\ \Delta F_{3C} \\ \Delta F_{4C} \end{bmatrix} = \begin{bmatrix} -1 & +1 & -1 & +1 \\ -1 & +1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 \end{bmatrix} \begin{bmatrix} \Delta\delta_{p_c} \\ \Delta\delta_{q_c} \\ \Delta\delta_{r_c} \\ \Delta\delta_{s_c} \end{bmatrix} \tag{6.39}$$

Using equation (6.38) in equation (6.39), we get

$$\begin{bmatrix} \Delta F_{1C} \\ \Delta F_{2C} \\ \Delta F_{3C} \\ \Delta F_{4C} \end{bmatrix} = \begin{bmatrix} -1 & +1 & -1 & +1 \\ -1 & +1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 \end{bmatrix} \begin{bmatrix} a_1 & 0 & 0 & 0 & a_2 \\ 0 & a_3 & 0 & a_4 & 0 \\ 0 & 0 & a_5 & 0 & a_6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta e_p \\ \Delta e_q \\ \Delta e_r \\ \Delta e_\alpha \\ \Delta e_\beta \end{bmatrix} \tag{6.40}$$

Taking $c_1 = {\omega_f}^2$ and $c_2 = -2\zeta_f\omega_f$, the actuator dynamics explained in section 2.6 can be re-written in matrix format with perturbed states as below,

$$\begin{bmatrix} \Delta\dot{F}_i \\ \Delta\ddot{F}_i \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -c_1 & c_2 \end{bmatrix} \begin{bmatrix} \Delta F_i \\ \Delta\dot{F}_i \end{bmatrix} + \begin{bmatrix} 0 \\ c_1 \end{bmatrix} \begin{bmatrix} \Delta F_{ic} \end{bmatrix} \tag{6.41}$$

where i = 1...4. Now expanding above equation for all 4 fins, we get

$$\begin{bmatrix} \Delta\dot{F}_1 \\ \Delta\ddot{F}_1 \\ \Delta\dot{F}_2 \\ \Delta\ddot{F}_2 \\ \Delta\dot{F}_3 \\ \Delta\ddot{F}_3 \\ \Delta\dot{F}_4 \\ \Delta\ddot{F}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -c_1 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -c_1 & c_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -c_1 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -c_1 & c_2 \end{bmatrix} \begin{bmatrix} \Delta F_1 \\ \Delta\dot{F}_1 \\ \Delta F_2 \\ \Delta\dot{F}_2 \\ \Delta F_3 \\ \Delta\dot{F}_3 \\ \Delta F_4 \\ \Delta\dot{F}_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ c_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 \end{bmatrix} \begin{bmatrix} \Delta F_{1c} \\ \Delta F_{2c} \\ \Delta F_{3c} \\ \Delta F_{4c} \end{bmatrix} \tag{6.42}$$

Representing above matrices with symbols below such as

$$B_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ c_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_1 \end{bmatrix}, B_2 = \begin{bmatrix} -1 & +1 & -1 & +1 \\ -1 & +1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 \end{bmatrix}, B_3 = \begin{bmatrix} a_1 & 0 & 0 & 0 & a_2 \\ 0 & a_3 & 0 & a_4 & 0 \\ 0 & 0 & a_5 & 0 & a_6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A_{con} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -c_1 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -c_1 & c_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -c_1 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -c_1 & c_2 \end{bmatrix}, \Delta x_{con} = \begin{bmatrix} \Delta F_1 \\ \Delta \dot{F}_1 \\ \Delta F_2 \\ \Delta \dot{F}_2 \\ \Delta F_3 \\ \Delta \dot{F}_3 \\ \Delta F_4 \\ \Delta \dot{F}_4 \end{bmatrix}, \Delta u_{con} = \begin{bmatrix} \Delta e_p \\ \Delta e_q \\ \Delta e_r \\ \Delta e_\alpha \\ \Delta e_\beta \end{bmatrix}$$

Using $B_{con} = B_1 \ B_2 \ B_3$, ignoring Δ for notational convenience and substituting equation (6.40) in equation (6.42) we get,

$$\dot{x}_{con} = A_{con} x_{con} + B_{con} u_{con} \tag{6.43}$$

We need the four fin deflections as output and they are available as states. Thus output equation can be formed as follows

$$y_{fin} = C_{final} x_{con} + D_{final} u_{con} \tag{6.44}$$

where $Y_{fin} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$, $C_{final} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$ and $D_{final} = \text{zeros}(4,5)$

Using ILAAT mixing logic explained in section 6.7, the effective aileron, elevator and rudder deflections can be retrieved using below operation.

$$\begin{pmatrix} \delta_p \\ \delta_q \\ \delta_r \\ \delta_s \end{pmatrix} = 0.25 \begin{pmatrix} -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \Delta F_1 \\ \Delta F_2 \\ \Delta F_3 \\ \Delta F_4 \end{pmatrix} \tag{6.45}$$

Taking $\Gamma = 0.25 \begin{pmatrix} -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix} = {B_2}^{-1}$, $y_{con} = \begin{pmatrix} \delta_p \\ \delta_q \\ \delta_r \\ \delta_s \end{pmatrix}$, equation (6.45) becomes

$$y_{con} = \Gamma y_{fin} \tag{6.46}$$

Using equation (6.46) in equation (6.44), and $C_{con} = \Gamma C_{final}$, $D_{con} = \Gamma D_{final}$ we get the final innermost rate controller state space as follows,

$$\boxed{\begin{aligned} \dot{x}_{con} &= A_{con} x_{con} + B_{con} u_{con} \\ y_{con} &= C_{con} x_{con} + D_{con} u_{con} \end{aligned} \qquad (6.47)}$$

6.9.3 Intermediate Loop

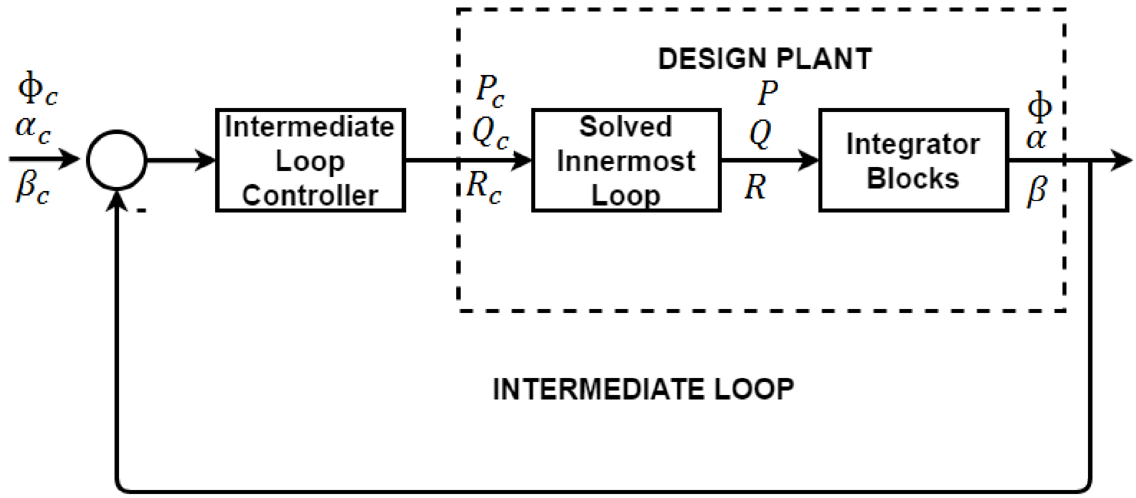

Figure 6.30: Block Diagram of Autopilot Intermediate Loop

Recalling the rate command generator equations which is the intermediate loop controller in this case. The error in α & β signals have to be passed on to the innermost loop.

$$\begin{aligned} P_c &= K_1(\phi_c - \phi) \\ Q_c &= K_2\frac{(A_{zcL} - A_{mz})}{Q_{dp}} - \frac{A_{zcL}}{V_b} \\ R_c &= K_3(\beta_c - \beta) \end{aligned} \tag{6.48}$$

Rewriting above equation interms of error signals and taking $K_{11} = \left(\frac{-K_2\, S_{ref}\, C_{N\alpha}}{Mass}\right)$, defining error signals $e_\phi = \phi_c$ - ϕ, $e_\alpha = \alpha_c$ - α and $e_\beta = \beta_c$ - β.

$$\begin{aligned} P_c &= K_1 e_\phi \\ Q_c &= K_{11} e_\alpha \\ R_c &= K_3 e_\beta \end{aligned} \tag{6.49}$$

Writing above equations in matrix form we get,

$$\begin{pmatrix} P_c \\ Q_c \\ R_c \\ e_\alpha \\ e_\beta \end{pmatrix} = \begin{pmatrix} K_1 & 0 & 0 \\ 0 & K_{11} & 0 \\ 0 & 0 & K_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} e_\phi \\ e_\alpha \\ e_\beta \end{pmatrix} \tag{6.50}$$

Thus the final state space equation of intermediate controller can be written as follows

$$Y_{inter} = D_{inter} U_{inter} \tag{6.51}$$

where $Y_{inter} = \begin{pmatrix} P_c \\ Q_c \\ R_c \\ e_\alpha \\ e_\beta \end{pmatrix}$, $D_{inter} = \begin{pmatrix} K_1 & 0 & 0 \\ 0 & K_{11} & 0 \\ 0 & 0 & K_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $U_{inter} = \begin{pmatrix} e_\phi \\ e_\alpha \\ e_\beta \end{pmatrix}$.

Missile Linear Autopilot Frequency Responses - Altitude Varying

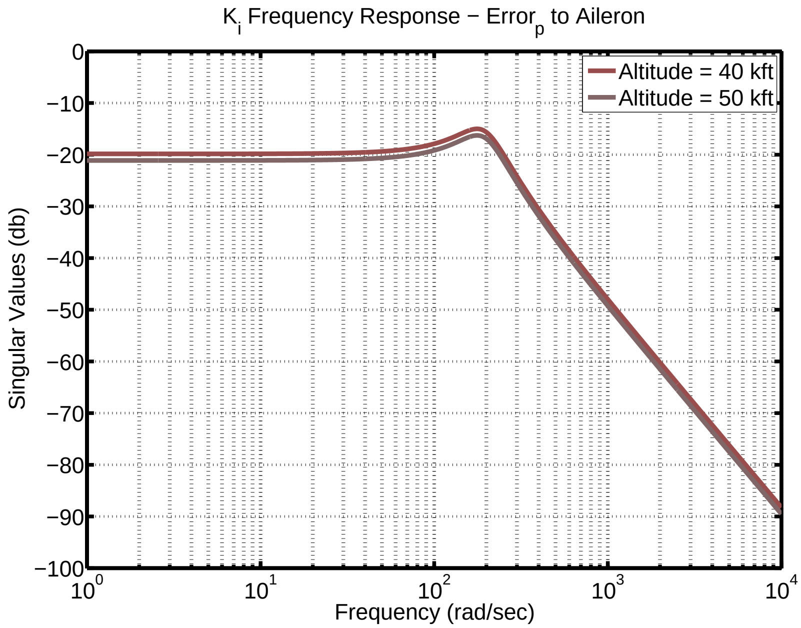

Figure 6.31: $K_i - 1^{st}$ Channel Frequency Response - Altitude Varying

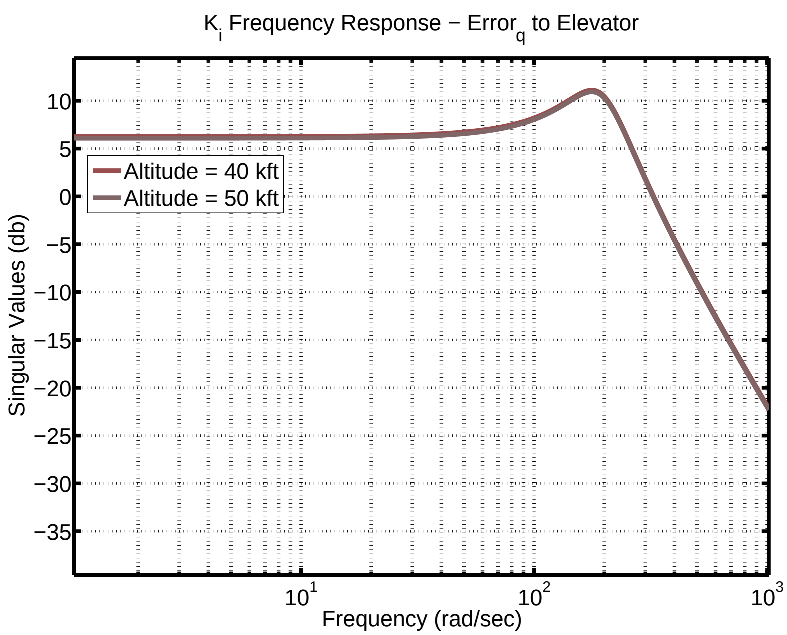

Figure 6.32: $K_i - 2^{nd}$ Channel Frequency Response - Altitude Varying

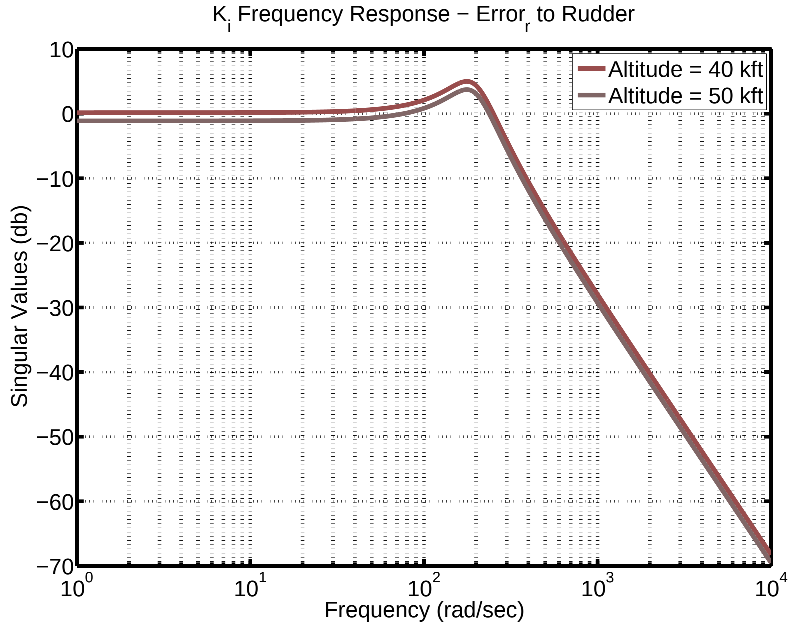

Figure 6.33: $K_i - 3^{rd}$ Channel Frequency Response - Altitude Varying

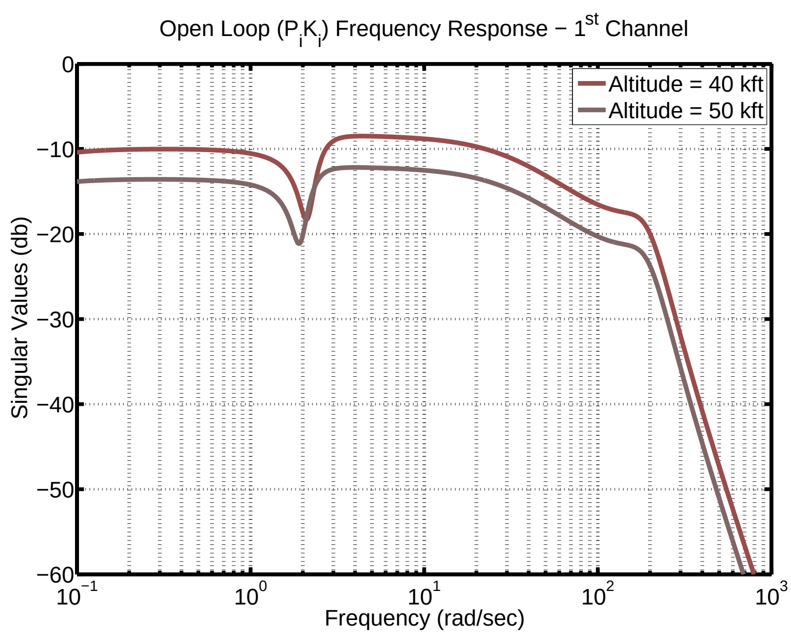

Figure 6.34: Open Loop Channel 1 Frequency Response - Altitude Varying

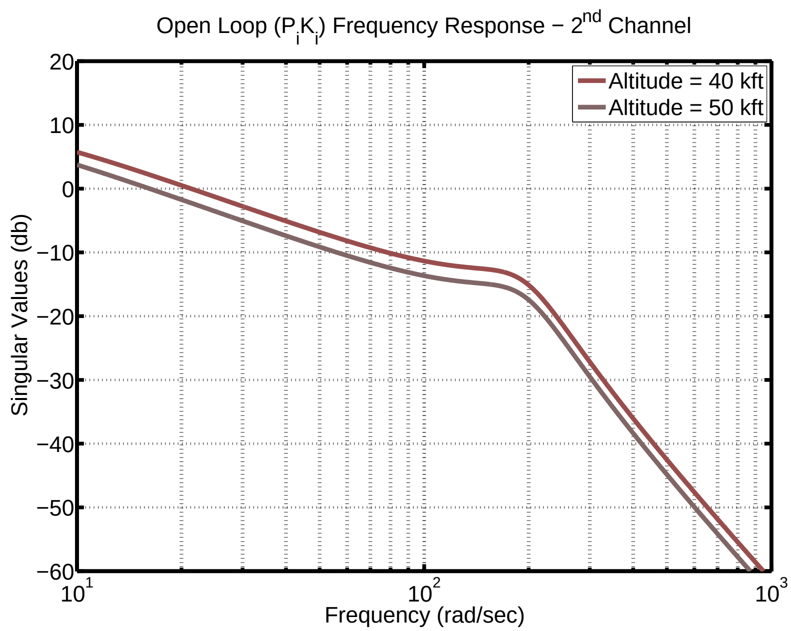

Figure 6.35: Open Loop Channel 2 Frequency Response - Altitude Varying

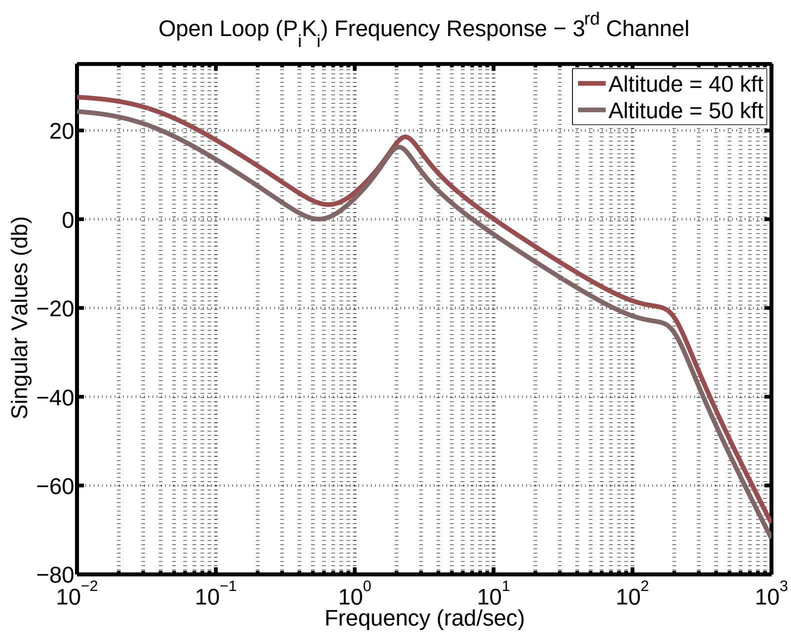

Figure 6.36: Open Loop Channel 3 Frequency Response - Altitude Varying

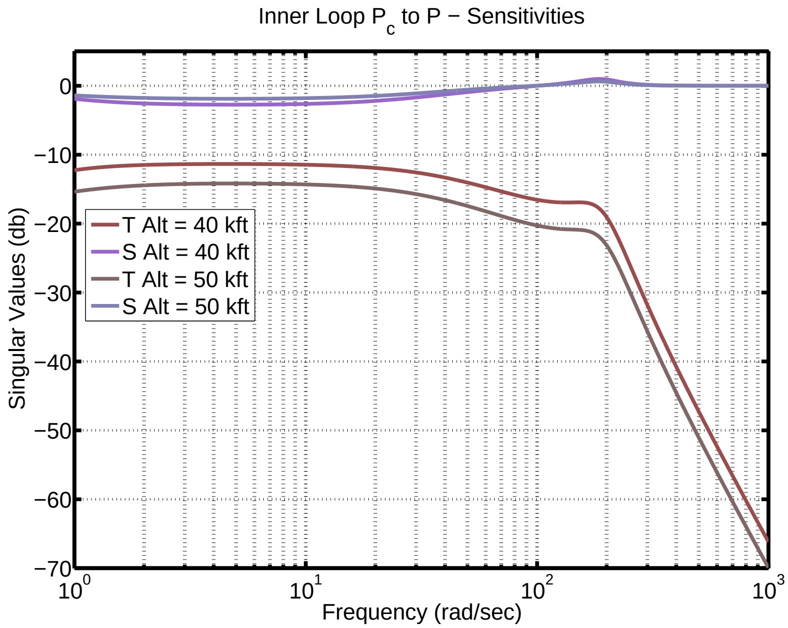

Figure 6.37: Inner Loop Complementary Sensitivity P_c vs P - Altitude Varying

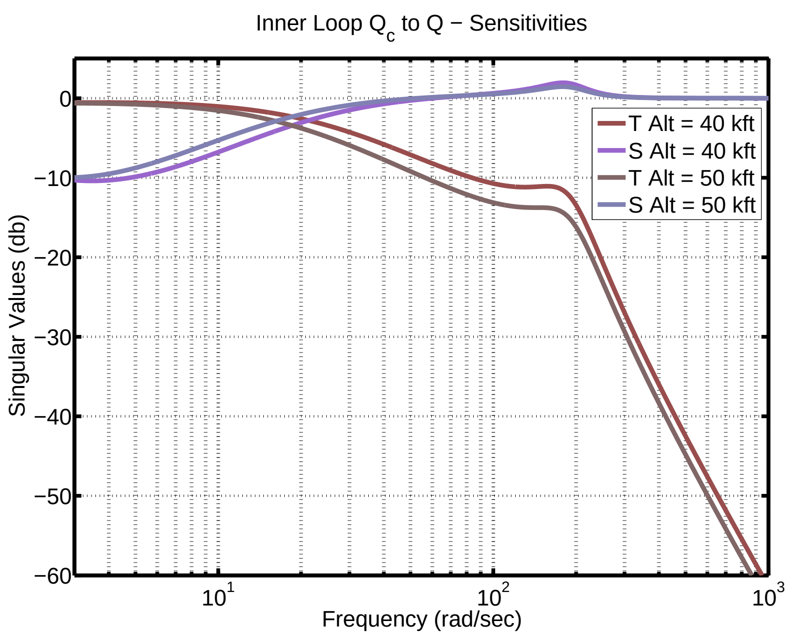

Figure 6.38: Inner Loop Complementary Sensitivity Q_c vs Q - Altitude Varying

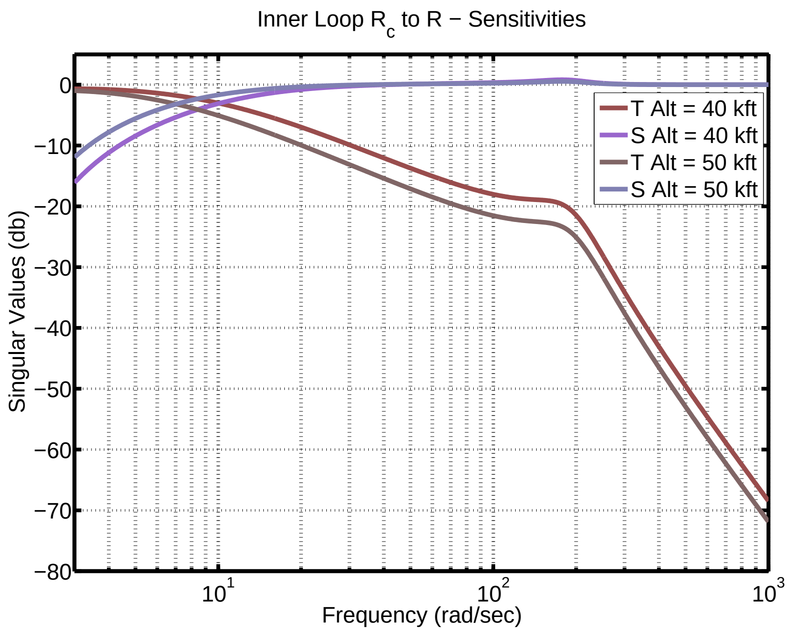

Figure 6.39: Inner Loop Complementary Sensitivity R_c vs R - Altitude Varying

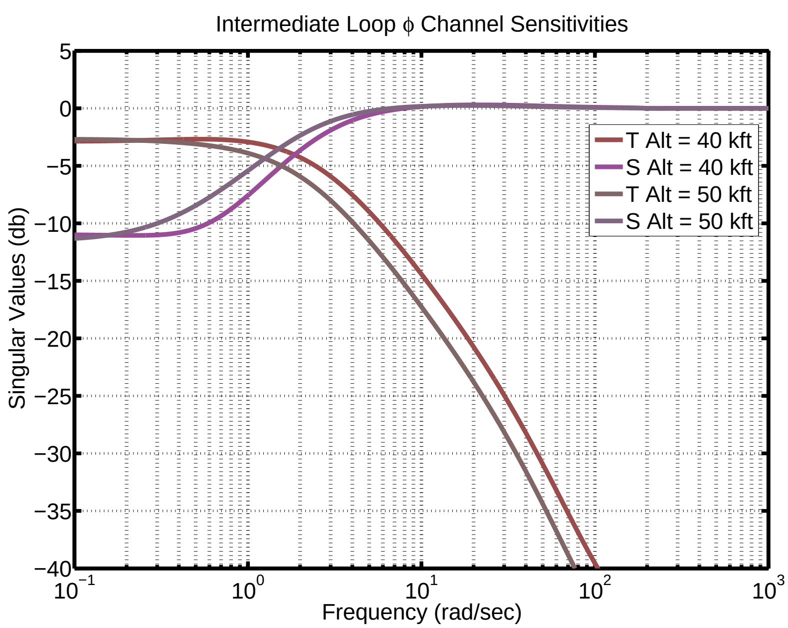

Figure 6.40: Intermediate Loop ϕ Channel Sensitivities - Altitude Varying

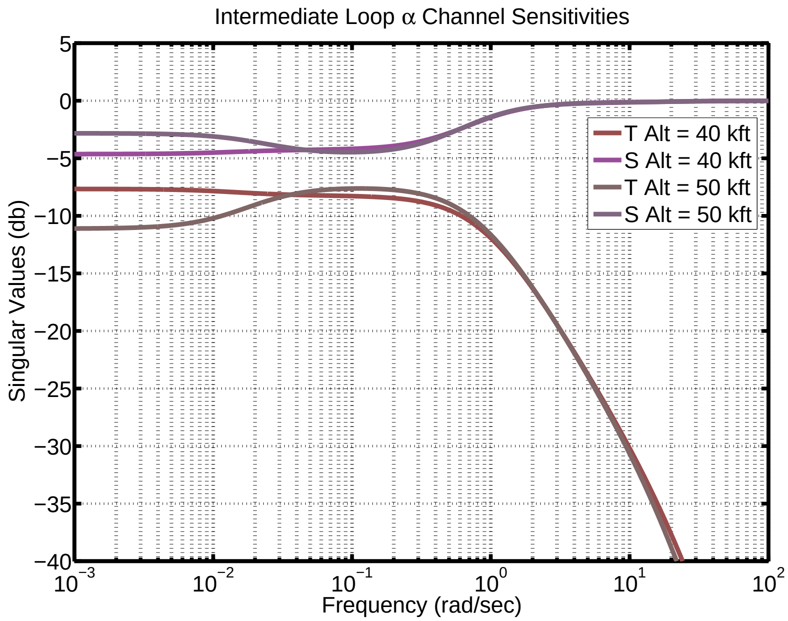

Figure 6.41: Intermediate Loop α Channel Sensitivities - Altitude Varying

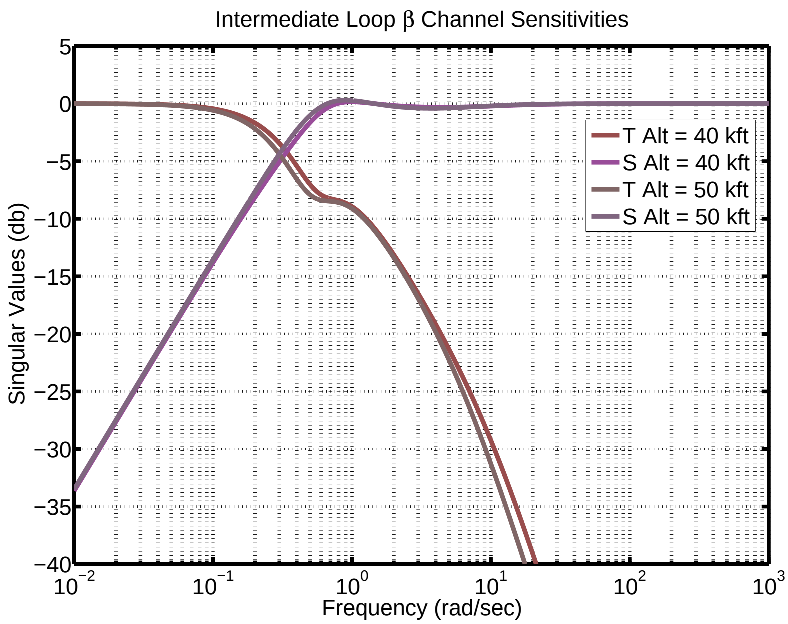

Figure 6.42: Intermediate Loop β Channel Sensitivities - Altitude Varying

All the above figures, 6.31 - 6.42, exhibit the following behaviour and the reason is explained below.

- As we go up, the air gets thinner.

- Missile fins can't operate efficiently at higher altitude because of the aerodynamic properties there.

- Thus to pitch up or down, more than the elevator fin deflection, it is the angle of attack that is more responsible for creating the required lift at such higher altitudes.

- The missile system as a whole becomes smaller as RHP pole & RHP zero decrease in magnitude as altitude increases because of lower dynamic pressure. Thus less bandwidth is required to to stabilize the missile. This the reason, why BTT missiles (equivalent to passenger aircrafts) operate at cruise control mode at higher altitude.

- Thus, it makes sense to have the autopilot to operate less aggressive as the altitude increases.

- Figures 6.37-6.39 corresponding to innermost loop sensitivities and Figures 6.40-6.42 corresponding to intermediate loop sensitivities show that their bandwidths decrease (becomes less aggressive, i.e. sluggish) as altitude increases.

- Similar behaviour is exhibited by the controller and open loop (both pertaining to innermost loop) frequency responses. The reader is referred to the figures 6.31 - 6.36 for observing the above said behaviour.

- It is very important to note here that, while the innermost rate control loop operates at a bandwidth of about 10 $\frac{rad}{sec}$ (on all 3 channels), the intermediate control loop operates at an bandwidth of about 1, 0.1 & 0.3 $\frac{rad}{sec}$ on ϕ, α &

β channels respectively which is about one decades slower than the innermost loop. Innermost loop faster than the intermediate loop ensures that the overall system is stable.

- α channel shows very less bandwidth, probably that is the reason why α_c design was omitted in the earlier designs as it does not bring in significant contribution to the overall performance. Research was conducted which shows that missile performance with the new α_c design and without it (old) design, showed no significant improvements. The new α_c was included because it is very important from a BTT missile point of view as BTT missile maneuvers by banking to desired orientation first and then angle of attack is varied in that normal plane to achieve the desired orientation while stabilizing the roll missile. The author feels this should need more investigation as to why this behaviour is exhibited.

Missile Linear Autopilot Frequency Responses - Mach Varying

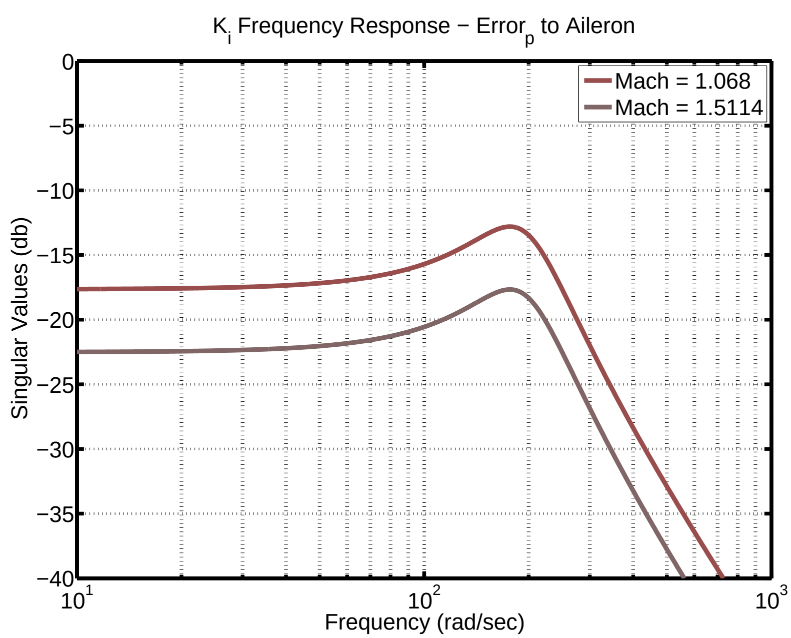

Figure 6.43: $K_i - 1^{st}$ Channel Frequency Response - Mach Varying

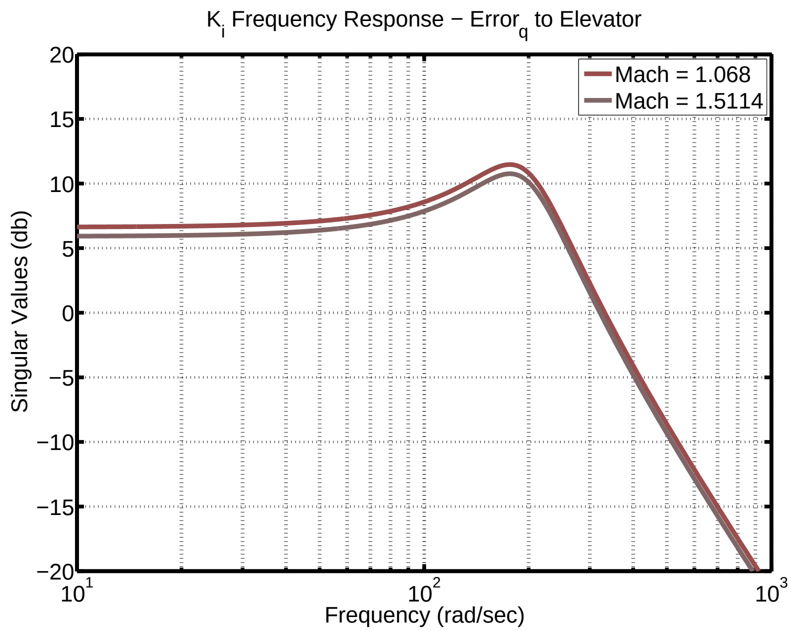

Figure 6.44: $K_i - 2^{nd}$ Channel Frequency Response - Mach Varying

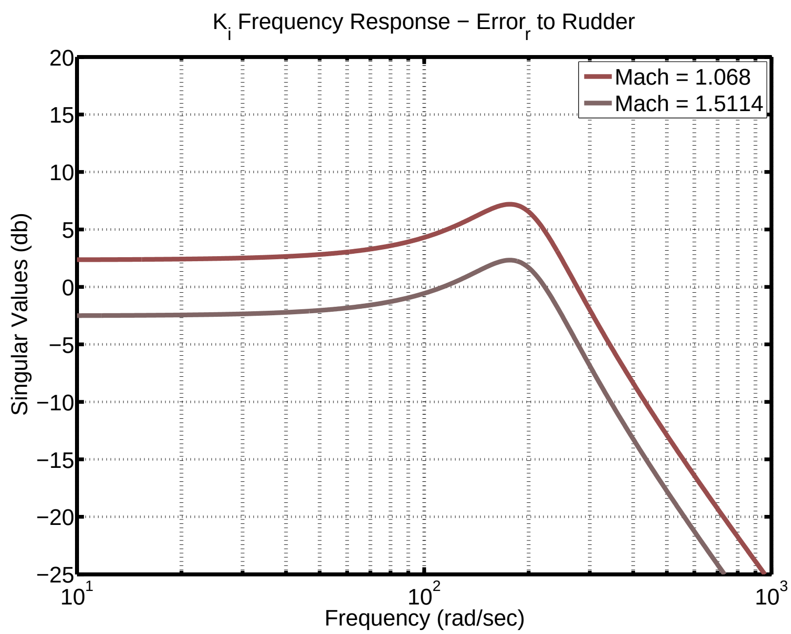

Figure 6.45: $K_i - 3^{rd}$ Channel Frequency Response - Mach Varying

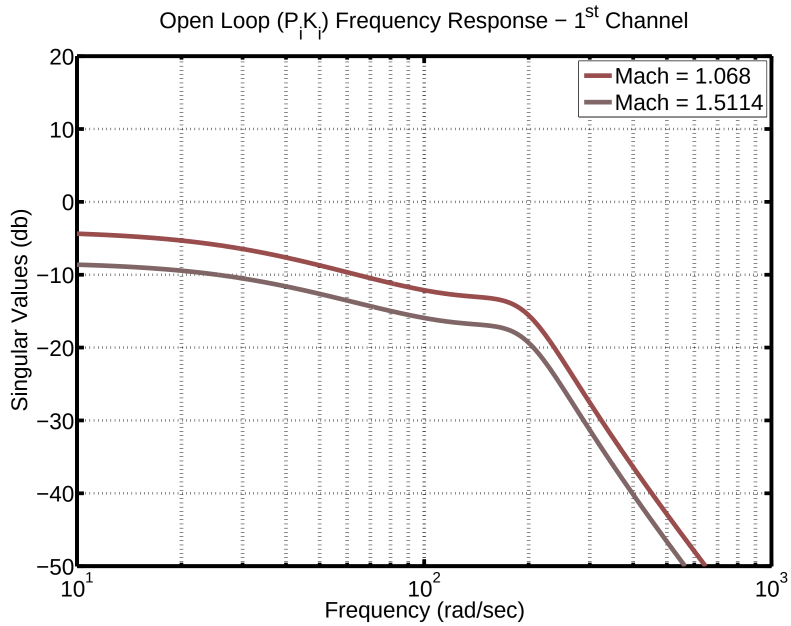

Figure 6.46: Open Loop Channel 1 Frequency Response - Mach Varying

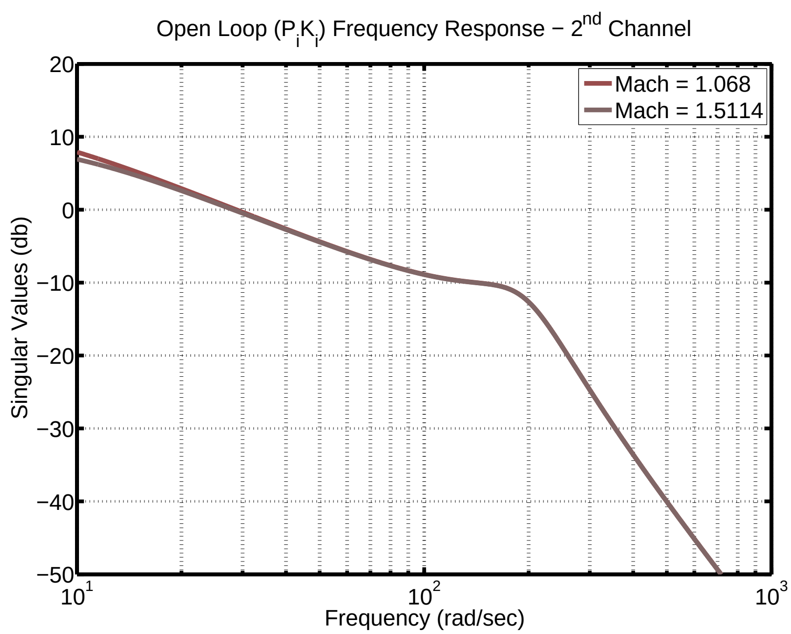

Figure 6.47: Open Loop Channel 2 Frequency Response - Mach Varying

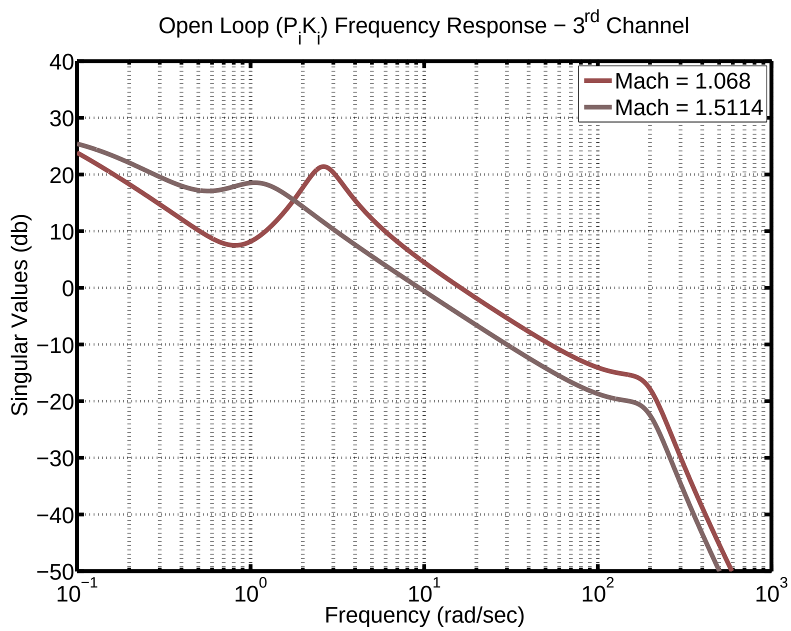

Figure 6.48: Open Loop Channel 3 Frequency Response - Mach Varying

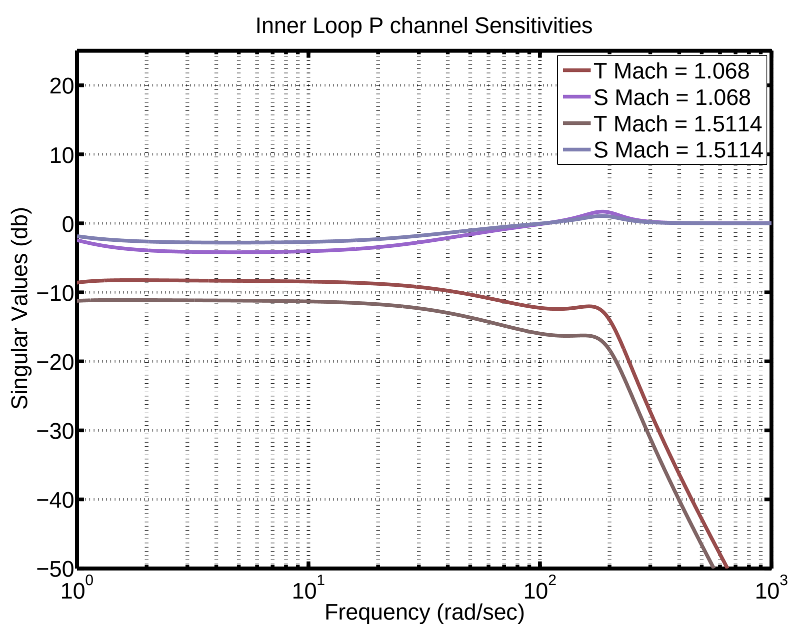

Figure 6.49: Inner Loop Complementary Sensitivity P_c vs P - Mach Varying

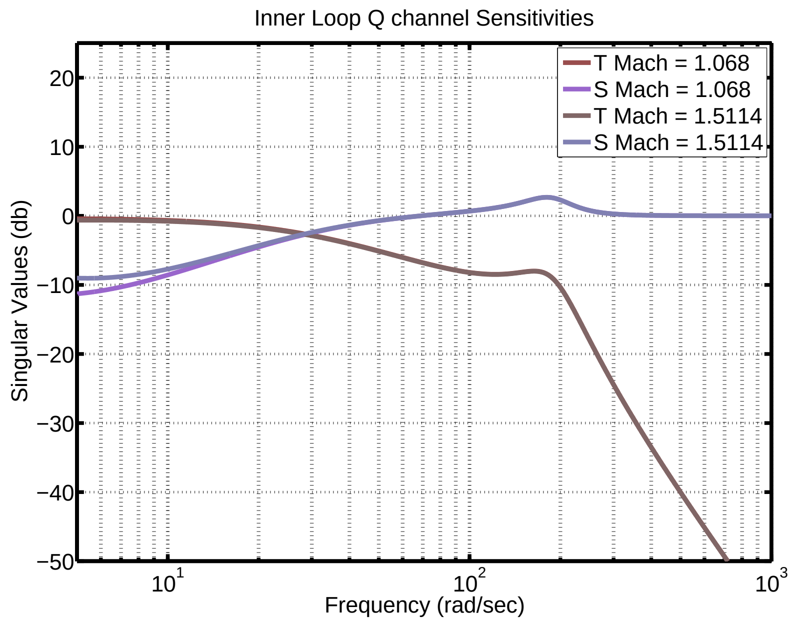

Figure 6.50: Inner Loop Complementary Sensitivity Q_c vs Q - Mach Varying

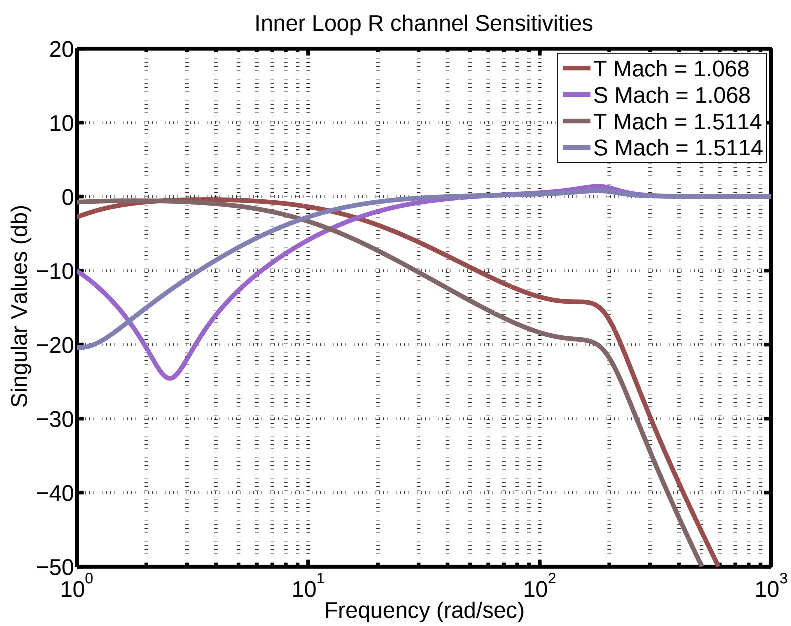

Figure 6.51: Inner Loop Complementary Sensitivity R_c vs R - Mach Varying

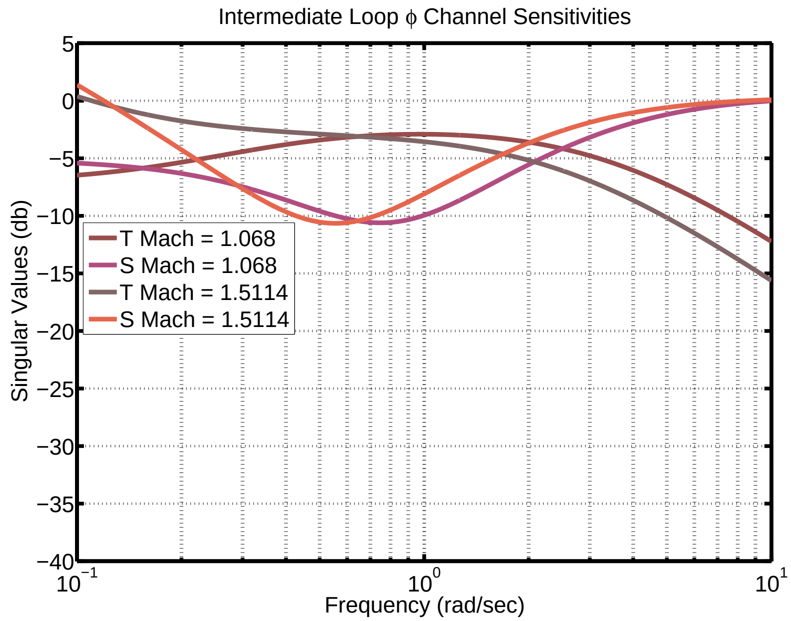

Figure 6.52: Intermediate Loop ϕ Channel Sensitivities - Mach Varying

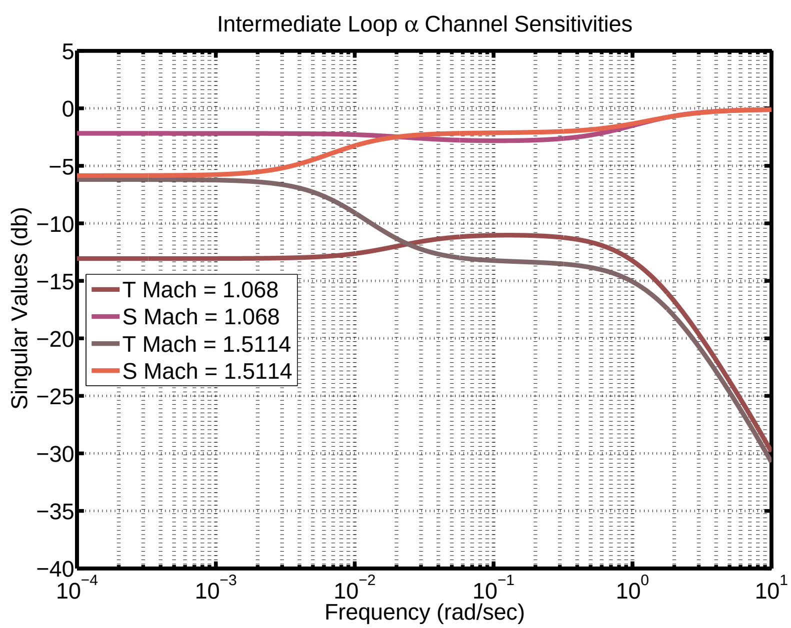

Figure 6.53: Intermediate Loop α Channel Sensitivities - Mach Varying

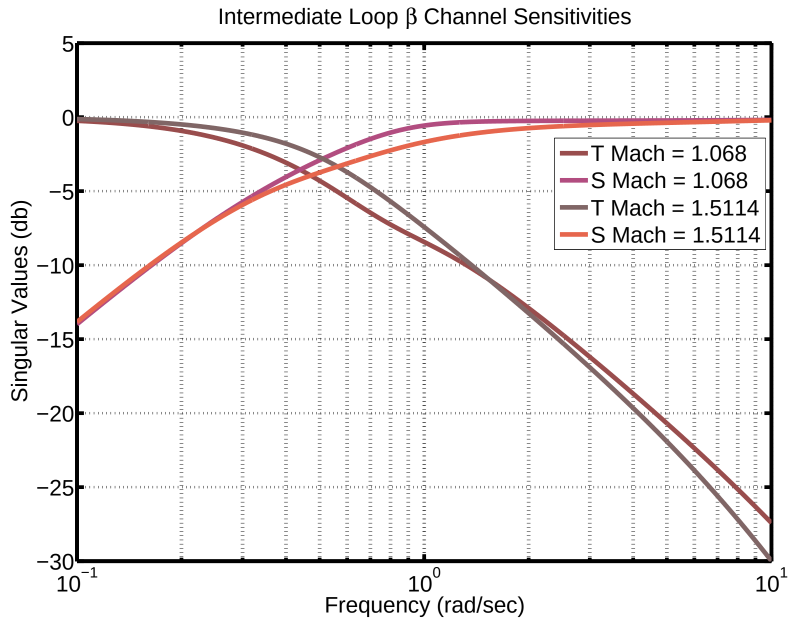

Figure 6.54: Intermediate Loop β Channel Sensitivities - Mach Varying

All the above figures, 6.43 - 6.54, exhibit the following behaviour and the reason is explained below.

- We know from figure 3.47, that $Mach \propto \frac{1}{h}$.
- The missile system as a whole becomes bigger as RHP pole & RHP zero increase in magnitude as Mach increases because of higher dynamic pressure. Thus more bandwidth is required to to stabilize the missile.
- Thus, it makes sense to have the autopilot to operate more aggressive as the Mach increases, mainly because the unstable pole grows in magnitude.
- Figures 6.49-6.51 corresponding to innermost loop sensitivities and Figures 6.52-6.54 corresponding to intermediate loop sensitivities corresponding to the longitudinal variables show that their bandwidths increase (becomes more aggressive, i.e. faster) as Mach increases.

- Similar behaviour is exhibited by the controller and open loop (both pertaining to innermost loop) frequency responses corresponding to the longitudinal variables. The reader is referred to the figures 6.43 - 6.48 for observing the above said behaviour.

- It is very important to note here that, while the innermost rate control loop operates at a bandwidth of about 10 $\frac{rad}{sec}$ (on all 3 channels), the intermediate control loop operates at an bandwidth of about 0.5, 0.01 & 0.5 $\frac{rad}{sec}$ on ϕ, α & β channels respectively which is about roughly one decades slower than the innermost loop. Innermost loop faster than the intermediate loop ensures that the overall system is stable.

6.10 Summary and Conclusions

This chapter has provided a comprehensive case study for our BTT Missile Autopilot. After the brief explanation of control law formulation using Incremental Nonlinear Dynamic Inversion technique, the nonlinear autopilot was explained, followed by its linearization and its analysis. The analysis show that the autopilot is very robust and properly follows the signal commands issued.

Chapter 7

NUMERICAL INTEGRATION

7.1 Introduction and Overview

Within this chapter, we address obtaining approximate solutions for the differential equations governing missile dynamics using numerical integration methods. Differential equations of first order can be solved using variety of mathematical tools. But for solving the equations using different initial conditions and real time inputs, we need a computer generated approximate solution. This is where numerical integration techniques, in particular Runge-Kutta methods come handy. Motivational examples from [68] are examined. Expecting a miss distance within the blast radius of the missile [70], nominal step size selection for a desired level of accuracy is demonstrated using missile target engagement geometry simulations.

Choosing an ideal step size for simulation is really important. Smaller the step size, more frequent the decisions are made about the next move. Given a small step size, missile moves a very small distance between each step towards the target. Similarly the larger step size involves less frequent decision making and missile moves to a big distance between each step towards the target. Going by the intuition, we normally prefer a smaller step size as we need a higher level of accuracy. Accuracy in this context refers to the final miss distance between missile and target. Conventional medium range missiles carrying high explosive warhead have blast radius of about 20ft [70]. This gives us an excellent information about what final miss distance we are looking for from our simulation. Smaller step size enjoys another benefit of not loading the actuators to perform till their saturation level continuously. This is evident from both the fin actuator and fin rate responses provided in this chapter.

To avoid making baby steps towards the target, we try to increase the step size and see where it starts to behave bad. The highest value of step size that gives us minimum miss distance without loading the actuators much is the ideal one. Trying an higher step size might even make a missile to miss the target initially and try hard enough to intercept it later. Given this, during such an awkward situation caused by larger step size, the autopilot is forced to make the actuators to work in the saturated level contantly. Thus engagement geometry of the missile target engagement is not smooth, making the life of the missile hard. The key goal of this chapter is to explain about the trade off involved in selection of ideal step size for integration.

Remainder of this chapter is organized as follows: In Section 7.2 all four Runge-Kutta methods are explained with an example and results are tabulated. Then step size selection through engagement geometry is explained in Section 7.3. Finally Section 7.4 summarizes and concludes the work explained in this chapter.

7.2 Runge-Kutta(RK) Integration Methods

Lets consider an initial value problem,

$$f(t,y) = \frac{dy}{dt} = y - t^2 + 1 \tag{7.1}$$

where $t_0 = 0$ and y($\mathrm{t_0}$) =$\mathrm{y_0}$ = 0.5 are the initial conditions considered. We need to solve for y between 0 $\leq$ t $\leq$ 2. Let "h" be the step size. Analytically solving this problem we get,

$$y(t) = t^2 + 2t + 1 - \frac{e^t}{2} \tag{7.2}$$

y(t=2) = 5.305471950534675 as the exact solution. In normal integration with end-points, we just use the end points of interval, and we dont know how the system behaves in between. Here, integration is carried out in small step size, which captures the behavior of system exactly over the entire time interval. This inherently tells us to keep the step size as minimum as possible to get a better solution. But decreasing the step size will increase the computational effort. Thus, a trade-off has to be observed between the two in order to get a desired level of accurate solution. Depending upon the importance given to the slope of function at different points in the interval, there are different types of methods available. Methods discussed below are Runge Kutta - $1^{st}, 2^{nd}, 4^{th} \& Fehlberg$.

7.2.1 Runge-Kutta 1^{st} Method

Also called as the Eulers method of integration, solution is given by equation 7.3 given below,

$$y(t+h) = y(t) + h\frac{dy(t,y)}{dt} \tag{7.3}$$

The next value is found out using value of function at that instant of time and derivative at that instant of time. The error between actual solution and approximated solution at all instances is relatively high in this method. Solving above example problem with this method we get 5.3001 as the final solution. Also, the error value is 5.3055 - 5.3001 = 0.0054. This is a high value of error given the accuracy of computers today. So this approximation is acceptable to certain extent.

7.2.2 Runge-Kutta 2^{nd} Method

Demanding need for more accuracy, we go for 2nd method, where slope at mid-point of the interval is considered for better approximation. The solution is given by equation 7.4 given below,

$$
\begin{aligned}
y(t+h) &= y(t) + k_2 \\
k_1 &= h\frac{dy(t,y)}{dt} \\
k_2 &= h\frac{dy(t+0.5h, y+k_1)}{dt}
\end{aligned}
\tag{7.4}
$$

Solving above example problem with this method, we get w = 5.3196 as the final solution. The error value between approximate solution and true sollution is still high. So this approximation is also acceptable only to a certain extent.

7.2.3 Runge-Kutta 4^{th} Method

This is also called classical Runge-Kutta method. This takes into account the slope of function at beginning, at the midpoint and at the end of interval to approximate the solution. Taking "h" to be the step size such that $t_i = t_0 + ih$, the solution is given by equation 7.5 given below,

$$
\begin{aligned}
w_i &\approx y(t_i), where \\
w_{i+1} &= w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
k_1 &= hf(t_i, w_i) \\
k_2 &= hf\left(t_i + \frac{h}{2}, w_i + \frac{k_1}{2}\right) \\
k_3 &= hf\left(t_i + \frac{h}{2}, w_i + \frac{k_2}{2}\right) \\
k_4 &= hf(t_i + h, w_i + k_3)
\end{aligned}
\tag{7.5}
$$

Solving above example problem with this method, we get 5.3055 as the final solution, error value between approximated solution and exact solution is negligible. We now know that this method is very good, only drawback being going through same step size for each and every iteration before settling down.

The error is compared with a threshold value at every step. If it is less than(more than) the threshold, we increase(decrease) the step size and re-do the current step again. This way, instead of going through same step size throughout the interval, we move forward intelligently adapting the step size. The solution is given by equation 7.6 given below,

$$
\begin{aligned}
R &= \frac{1}{h}|\tilde{w}_{i+1} - w_{i+1}| \\
w_{i+1} &= w_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \\
\tilde{w}_{i+1} &= w_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6 \\
k_1 &= hf(t_i, w_i) \\
k_2 &= hf\left(t_i + \frac{h}{4}, w_i + \frac{k_1}{4}\right) \\
k_3 &= hf\left(t_i + \frac{3h}{8}, w_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \\
k_4 &= hf\left(t_i + \frac{12h}{13}, w_i + \frac{1932}{2197}k_1 + \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \\
k_5 &= hf\left(t_i + h, w_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 + \frac{845}{4104}k_4\right) \\
k_6 &= hf\left(t_i + \frac{h}{2}, w_i - \frac{8}{27}k_1 + \ k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right) \\
\delta &= 0.84\left(\frac{\varepsilon}{R}\right)^{\frac{1}{4}}
\end{aligned}
\tag{7.6}
$$

if R $\leq \varepsilon$	Keep w as the current step solution and move to the next step with the step size δh
if R $> \varepsilon$	recalculate the current step with the step size δh

Solving above example problem with this method, we get 5.3055 as the final solution, which was obtained in very less amount of steps. The error is as usual very negligible like RK-4 method since internally this method used RK4 for approximation.

Integration Method	**Error between True and Approximate Solutions**	**Computational Effort**	**No. of Iterations**
RK-1	High	Very Less	More
RK-2	Considerably Low	Less	More
RK-4	Very Negligible	High	More
RK-Fehlberg	Very negligible	Very high	Very Less

Table 7.1: Comparison of Runge-Kutta Integration Methods

7.3 Nominal Step Size Selection using Engagement Geometry Analysis

Optimal step size will enable a smooth flight for missile without loading the actuators heavily and it will enable the missile to intercept the target with excellent accuracy. Here in this research, the target maneuvered using the sheldon mode while the missile tried intercepting it using optimal guidance and for the same initial flight conditions given by Table 7.2, the step size was varied to see where the simulation started to fail. This will give us an upper bound on the step size. See Figure 7.1. Similarly even smaller step sizes were tried and they gave us satisfactory results. See Figure 7.2. But they were having longer flight time because the missile was making baby steps towards the target. So to fasten the decision process and that too with expected accuracy, the optimal step size was selected which resulted in both fast and accurate simulations.

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Azimuth Angle	0 deg	Aspect Angle	0 deg

Table 7.2: Flight Conditions for Miss Distance vs Integration Step Size

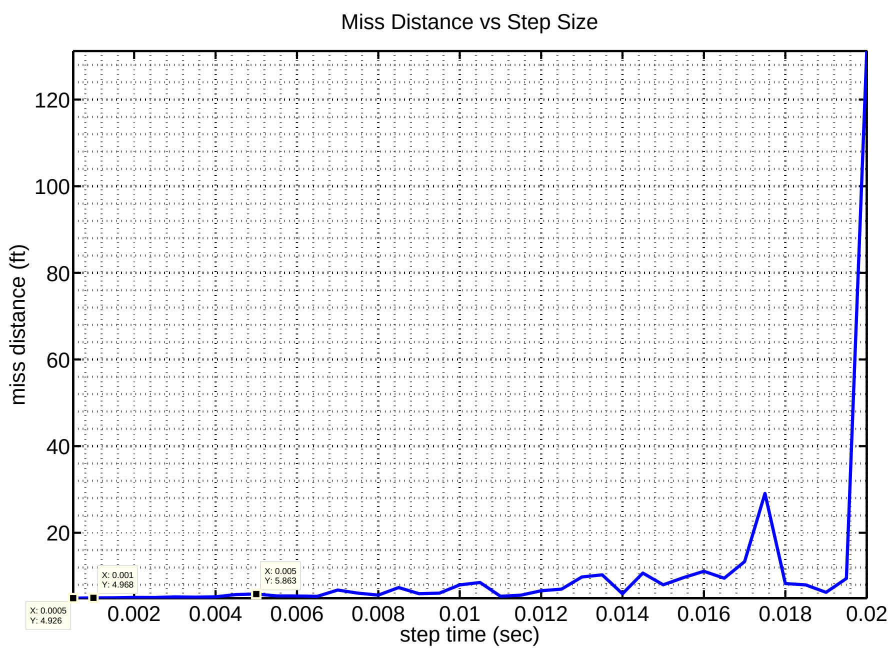

Figure 7.1: Miss Distance vs Integration Step Size

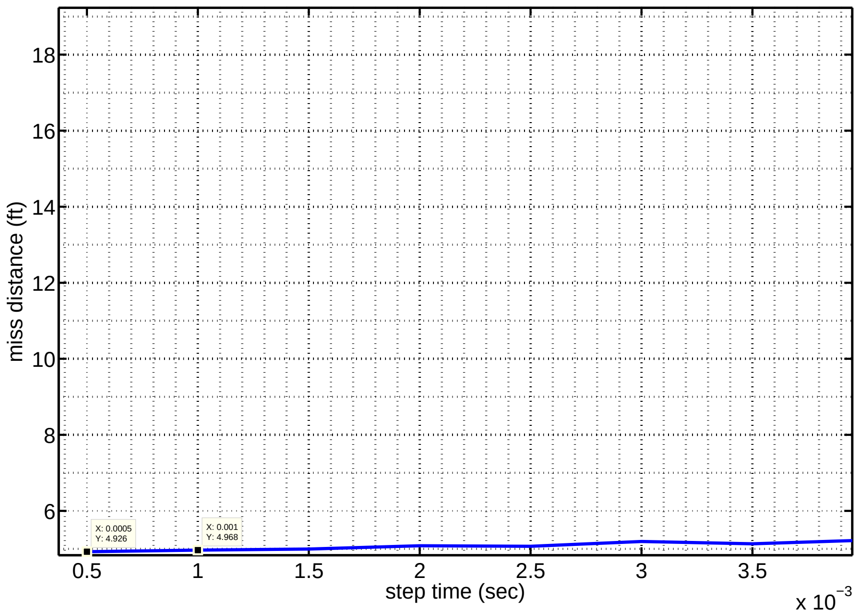

Figure 7.2: Zoomed in Figure 7.1

Engagement Geometry. Referring to Figures 7.3 and 7.5, it is evident that smaller step sizes gave a smooth engagement geometry, while larger step sizes made the life of missile difficult. By careful observation of Figures 7.3 and 7.4, it can be easily seen that as the step size grows larger, the missile starts to miss the target resulting in a bad simulation. It is important to emphasize here that a bad step size will result only in a bad simulation resulting in the missile missing the target, while it does not imply that the missile does not have the capability to hit the target. While operated with a big step size, the missile tries to the best of its abilities to make sharp turns to intercept the target even if it misses the target at initial ranges. While doing sharp turns, the missile fin actuators hit their saturation levels frequently, which is obviously not a good condition for fin actuators. The reader is referred to the Figures 7.7 and 7.8 to visualize the phenomenon explained above.

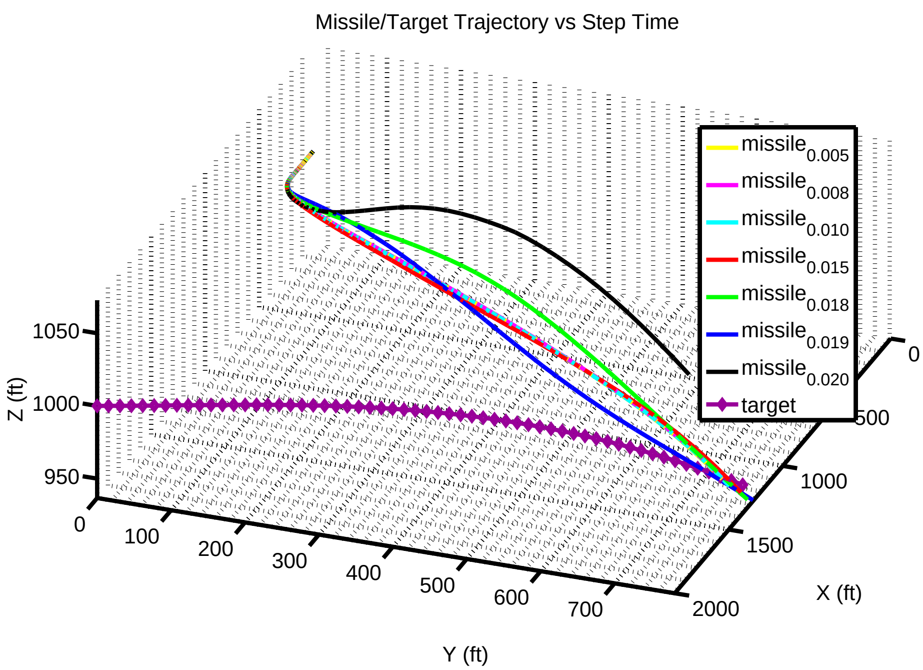

Figure 7.3: Engagement Geometry 3D Plot for different step sizes

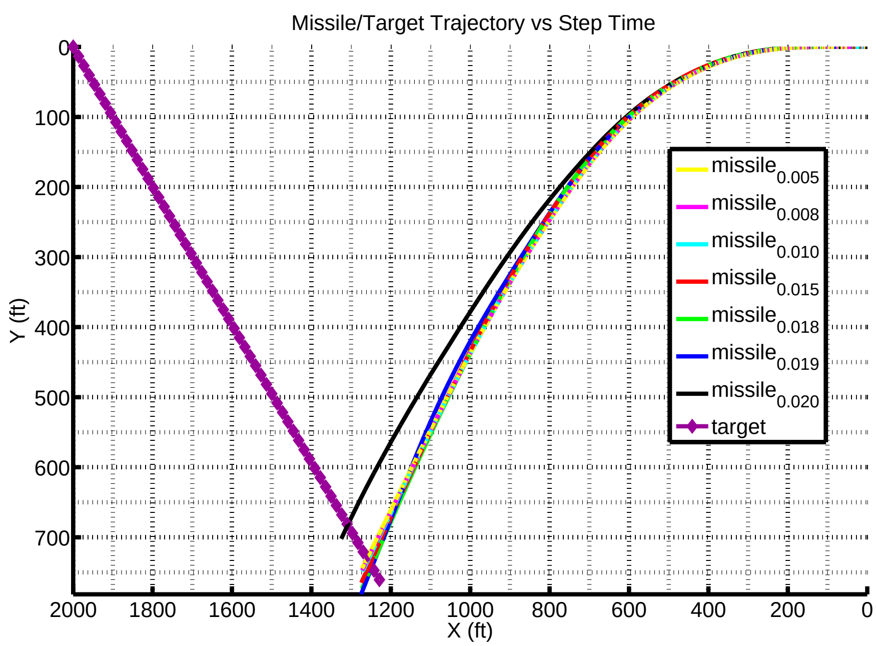

Figure 7.4: Engagement Geometry 2D Plot for different step sizes

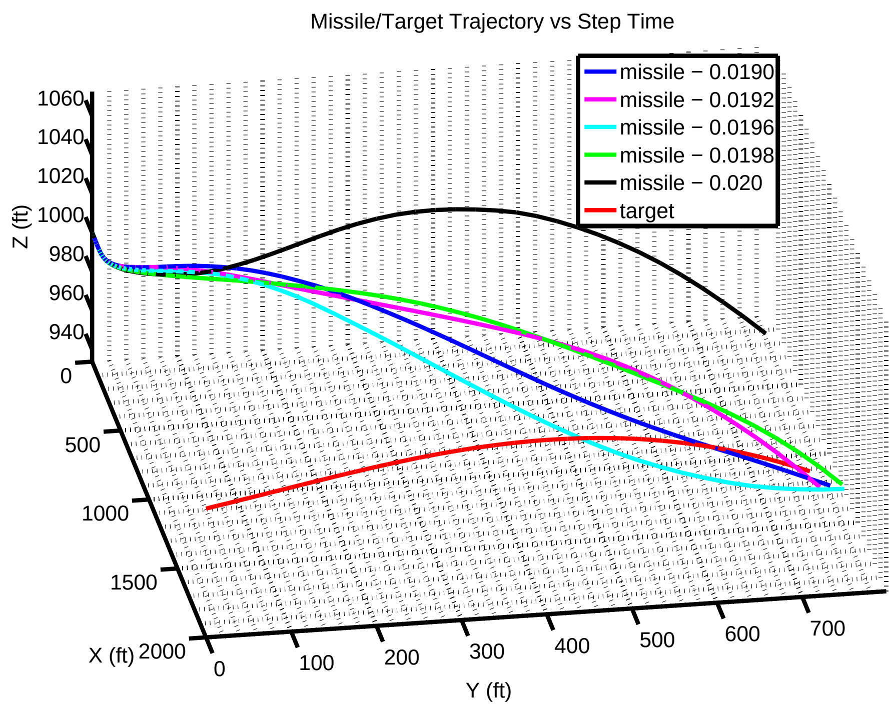

Figure 7.5: Engagement Geometry 3D Plot showing Step Size Failure

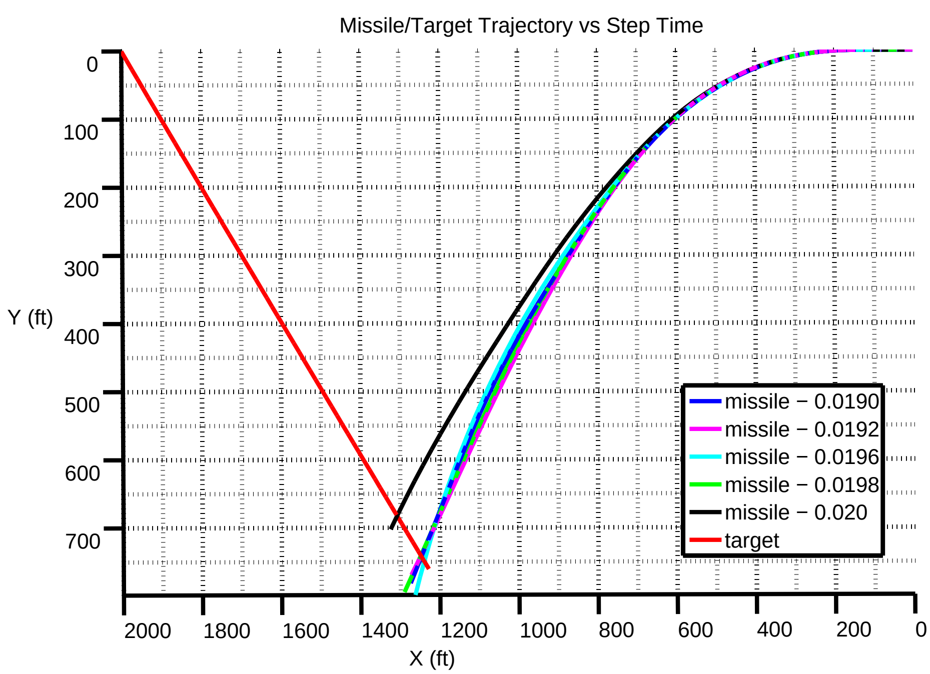

Figure 7.6: Engagement Geometry 2D Plot showing Step Size Failure

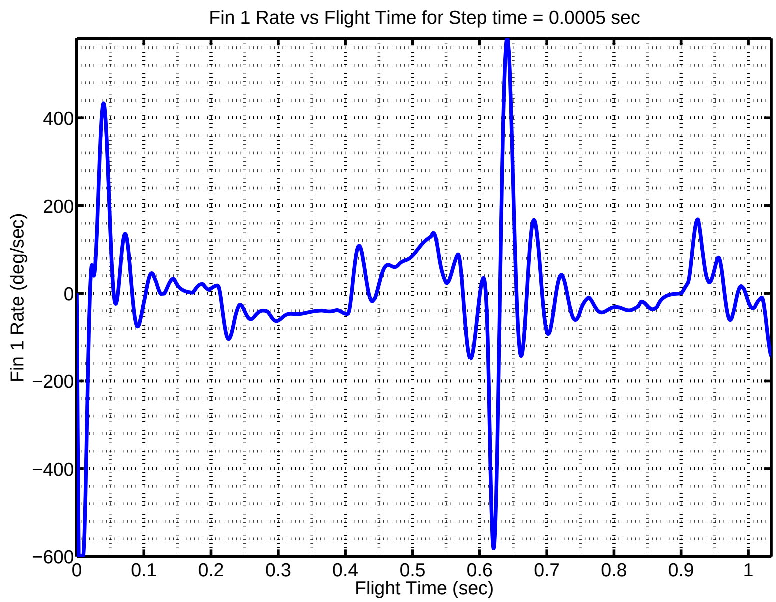

Figure 7.7: Fin Deflection Rate for Smaller Step Size

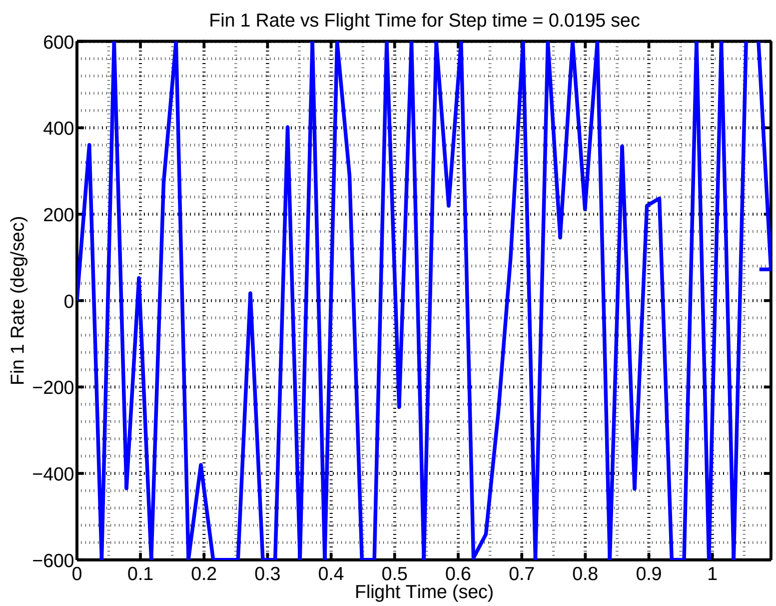

Figure 7.8: Fin Deflection Rate for Bigger Step Size

For the above initial flight conditions, step size of 0.005 would be very optimal which can be seen through the Figure 7.1. This optimal step size is expected to vary for different range and other flight conditions.

7.4 Summary and Conclusions

This chapter gives a brief idea about the proper usage of numerical integration in complex simulation like missile guidance control systems. The four different Runge-Kutta methods were explained using a mathematical example and their merits and demerits were tabulated. Then the procedure to select the optimal step size for numerical integration was explained using the engagement geometry analysis. Effect of bad step size selection on actuators hitting their saturation levels were clearly explained. Thus the purpose of the chapter was to provide a solid foundation on the numerical integration methods used to numerically approximate the complex, nonlinear missile and target differential equations during the missile target engagement.

Chapter 8

MISS DISTANCE ANALYSIS

8.1 Introduction and Overview

The purpose of this chapter is to illustrate the hunting capabilities of the BTT missile considered in this research. Given a thrust profile and fixed initial conditions, the analysis made in this chapter will answer how good a missile will be in intercepting a target within its killing range. The high fidelity environment used throughout the simulation used in this research is employed to study the miss distance profile with respect to different missile/target engagement parameters as described in the relevant GNC textbooks [51] and [52]. Also the work done in this chapter will lay a basic foundation and serve as a perfect motivation factor for kill zone estimation, which is explained in brief in the Chapter 9. Conventional warheads carried by the missile have a circular blast radius of about 20 ft [70]. Thus any simulation resulting in a final miss distance less than 20 feet is taken granted as a hit and miss distance profile is obtained as per this logic. Each section in this chapter will have information about the flight conditions considered, the result and its inference. The chapter is organized as follows: Section 8.2 will briefly discuss the miss distance profile change when the proportional gain is varied. Here the missile is assumed to possess Proportional Navigation guidance law to intercept the target. Section 8.3 analyses the effect of altitude variation on the miss distance profile. Section 8.4 throws light on effect of varying the maximum acceleration capability of the missile over the final miss distance achieved. Section 8.5 discusses the effect of initial missile speed on the final miss distance profile. Section 8.6 establishes a brief idea about how the miss distance profile varies as the target is made to maneuver more and more. Missile is assumed to possess Differential Game Theory guidance to intercept the target here. Section 8.7 shows how the miss distance varies when the target's orientation with respect to the missile measured in terms of *Aspect*. Section 8.8 elaborates how the miss distance varies when the initial range is varied. This motivates the work done in the entire Chapter 9. Finally Section 8.9 summarizes and concludes the work done in this chapter and gives a rough idea about estimating the missile's capabilities using above analyses.

8.2 Miss Distance Dependence on Proportional Gain

Throughout the simulation conducted here in this section, the missile is made to possess proportional navigation guidance and all the three target maneuvers are tested by varying the proportional gain. The initial flight conditions considered are shown in the Table 8.1.

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Integration Method	RK-4
Guidance Law	Proportional	Aspect Angle	135 deg

Table 8.1: Flight Conditions for Miss Distance vs Proportional Gain

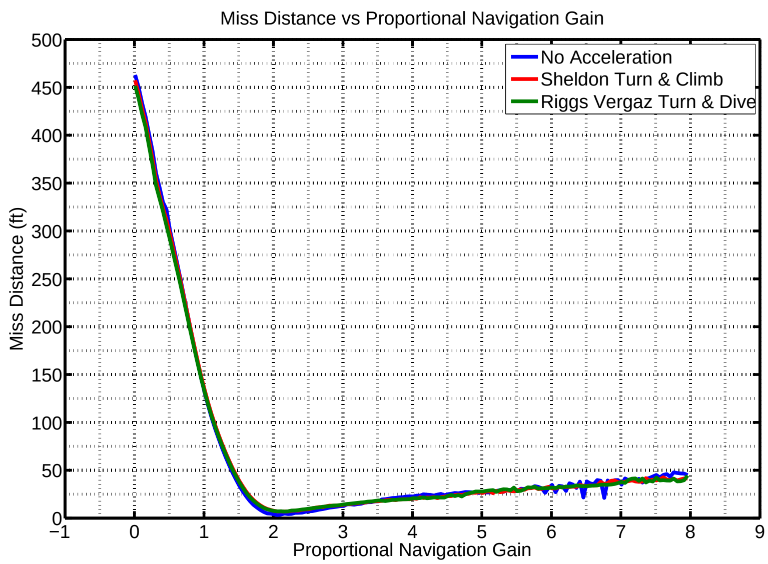

Figure 8.1: Miss Distance vs Proportional Gain

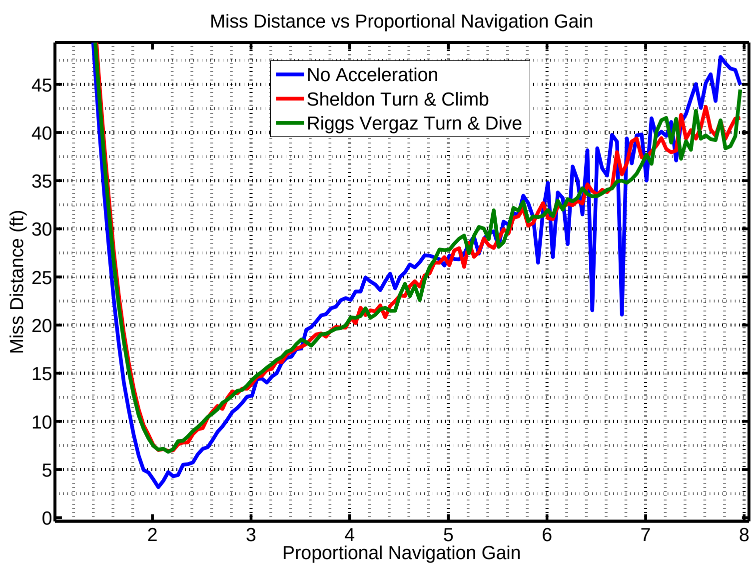

Figure 8.2: Zoomed in Figure 8.1

Inferences.

- From Figures 8.1 and 8.2, it is clearly evident that the miss distance is higher when the proportional gain is too small and the miss distance reaches the minimum value which is unique for different flight conditions.

- Beyond the minimum, the miss distance increases slowly as we increase the gain and this behaviour persists irrespective of the target maneuver.

- It is intuitive that if the proportional gain is very small, the missile will respond slowly and will not be able to catch the target and similarly if the gain is big, the outer guidance loop will become unstable due to the high frequency seeker dynamics.

- It is also observed that these changes are observed only when the initial altitude is small. At higher altitudes, the miss distance essentially becomes independent of the gain.

- The reader is referred to the [17] for further insight and information about this section.

8.3 Miss Distance Dependence on Initial Engagement Altitude

Throughout the simulation conducted here in this section, the missile is made to possess different guidance laws and all the three target maneuvers are tested by varying the initial engagement altitude. The initial flight conditions considered are shown in the Table 8.2.

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Maneuver Index	0.25
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Intgration Method	RK-4
Proportional Gain	2.1	Aspect Angle	135 deg

Table 8.2: Flight Conditions for Miss Distance vs Engagement Altitude

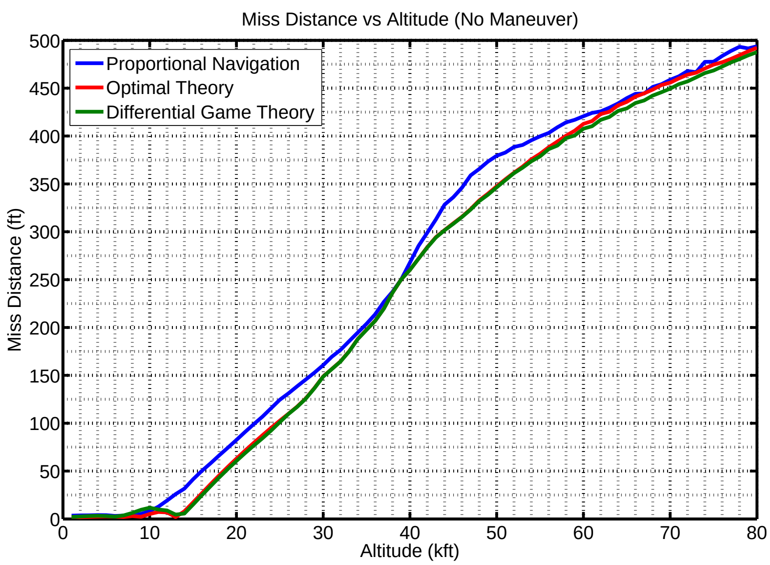

Figure 8.3: Miss Distance vs Engagement Altitude - No Maneuver

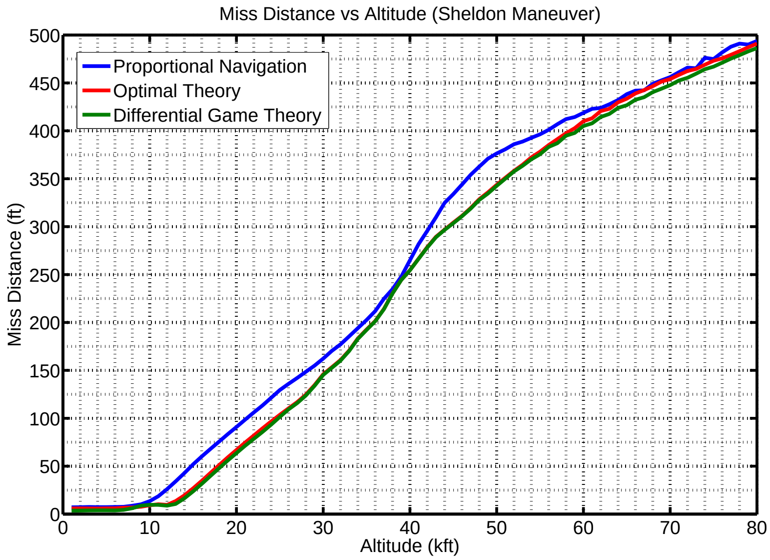

Figure 8.4: Miss Distance vs Engagement Altitude - Sheldon Maneuver

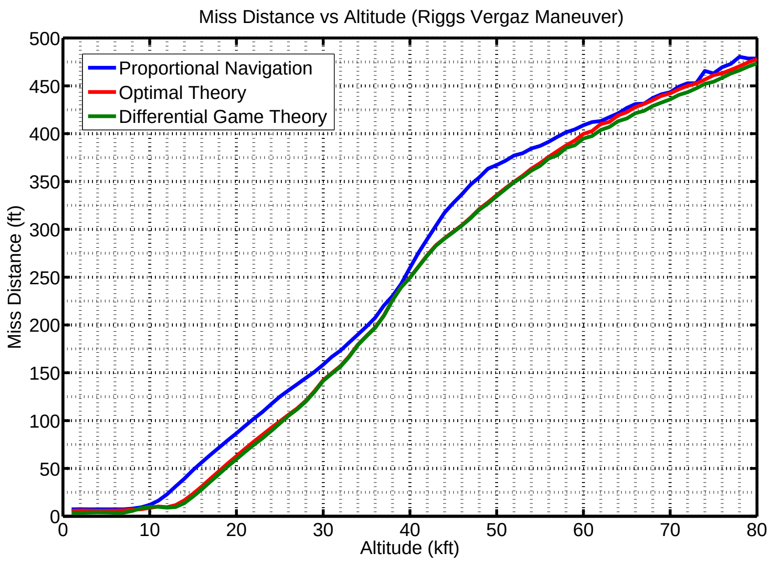

Figure 8.5: Miss Distance vs Engagement Altitude - Riggs Vergaz Maneuver

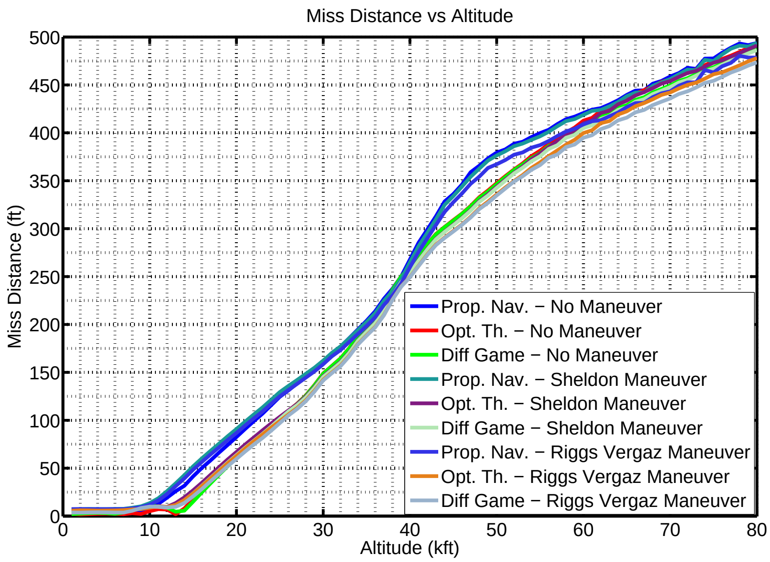

Figure 8.6: Miss Distance vs Engagement Altitude - All Maneuvers

Inferences.

- As Altitude increases, the miss distance increases.
- It is intuitive that the air density decreases with increasing altitude, one might expect that the missile fins lose their aerodynamic effectiveness at higher altitudes.
- Figures 8.3 - 8.6 supports our intuitive inference about the inability of the fins to control the missile in the thin air of the upper atmosphere.
- The reader is referred to the [17] for further insight and information about this section.

8.4 Miss Distance Dependence on Missile Maximum Acceleration

Throughout the simulation conducted here in this section, the missile is made to possess different guidance laws and all the three target maneuvers are tested by varying the initial maximum missile acceleration. The initial flight conditions considered are shown in the Table 8.3.

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Proportional Gain	2.1	Maneuver Index	0.25
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Altitude	-1000 ft	Aspect Angle	135 deg

Table 8.3: Flight Conditions for Miss Distance vs Missile Maximum Acceleration

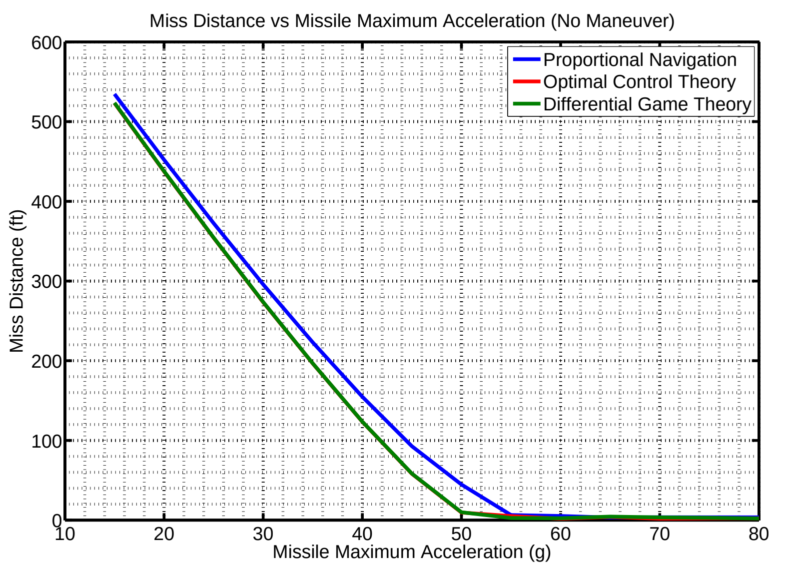

Figure 8.7: Miss Distance vs Missile Max. Acceleration - No Maneuver

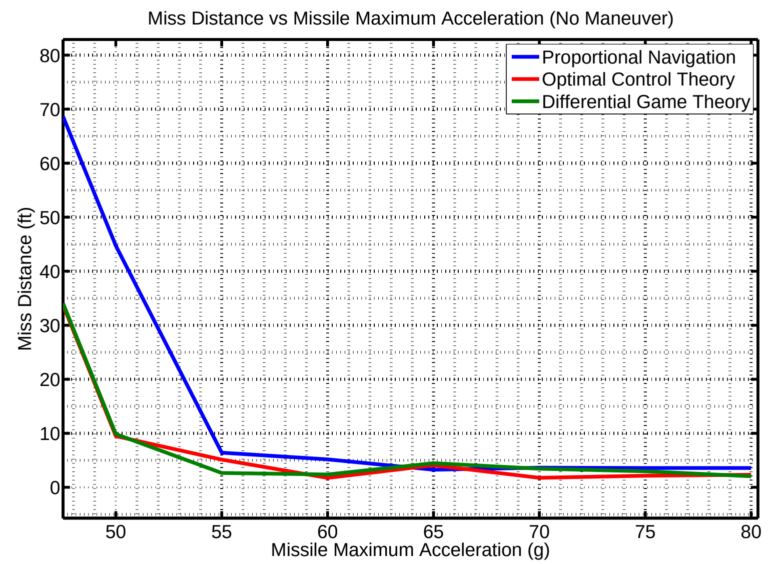

Figure 8.8: Zoomed in Figure 8.7

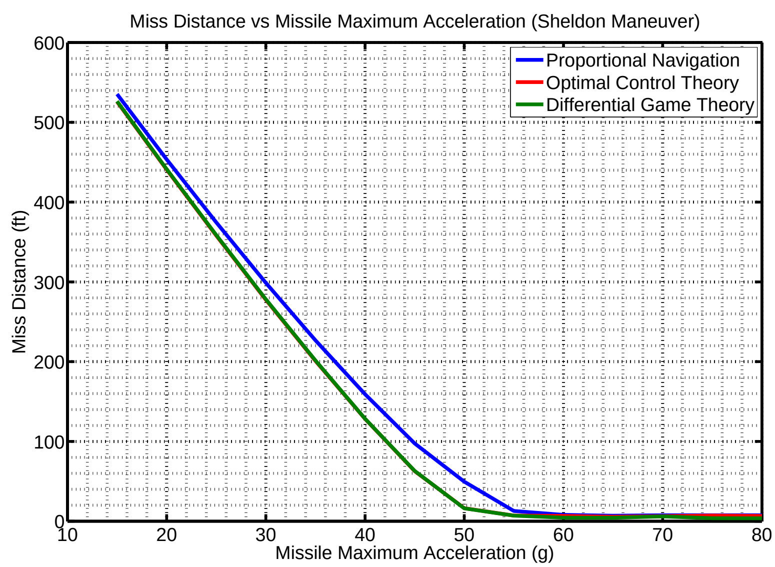

Figure 8.9: Miss Distance vs Missile Max. Acceleration - Sheldon Maneuver

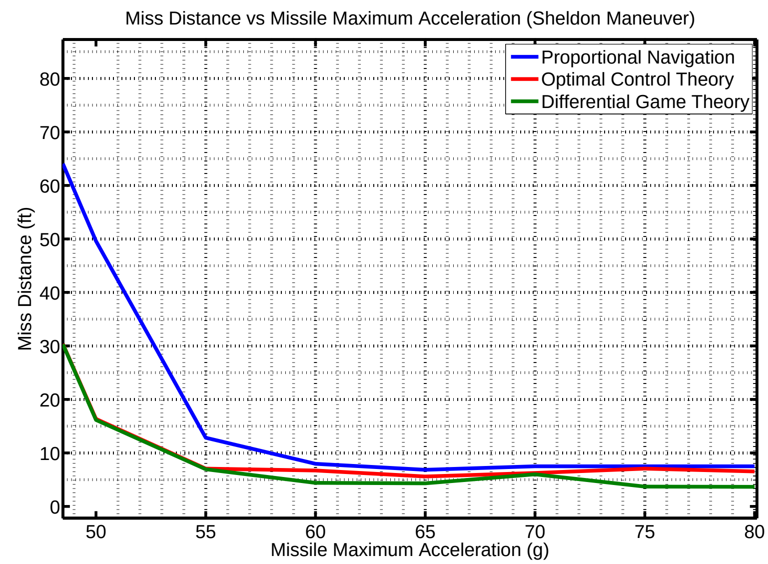

Figure 8.10: Zoomed in Figure 8.9

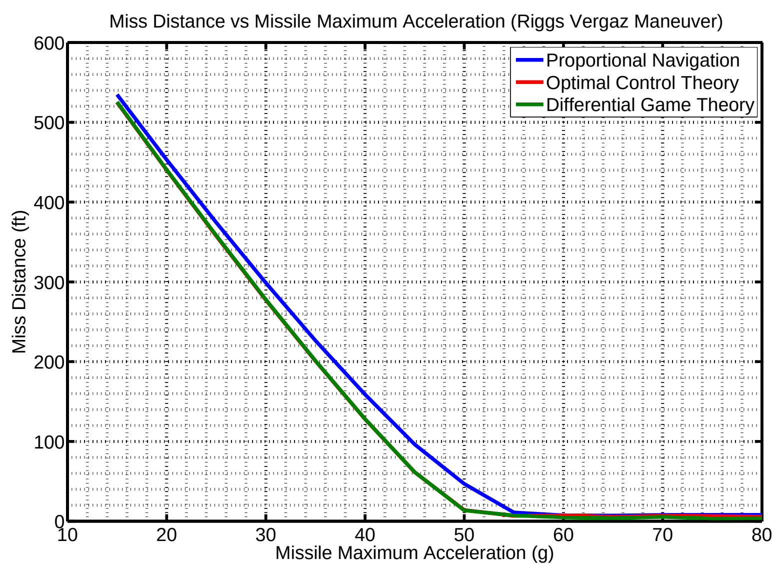

Figure 8.11: Miss Distance vs Missile Max. Acceleration - Riggs Vergaz Maneuver

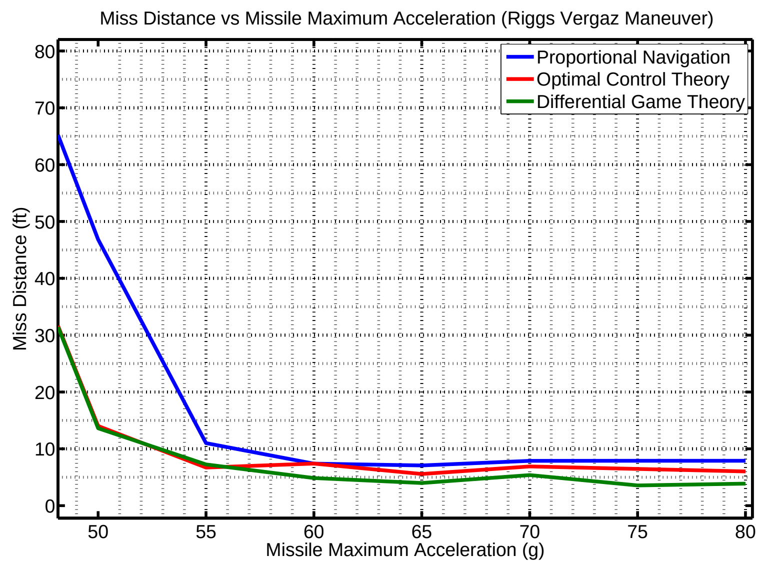

Figure 8.12: Zoomed in Figure 8.11

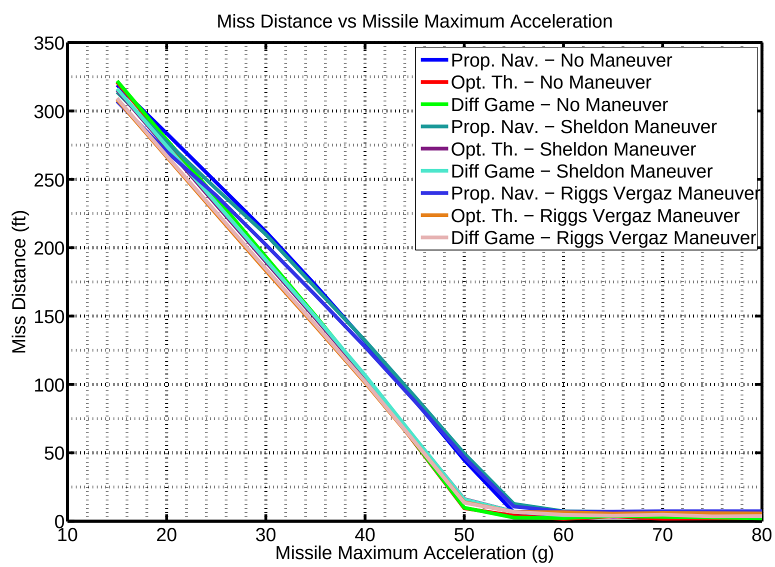

Figure 8.13: Miss Distance vs Missile Max. Acceleration - All Maneuvers

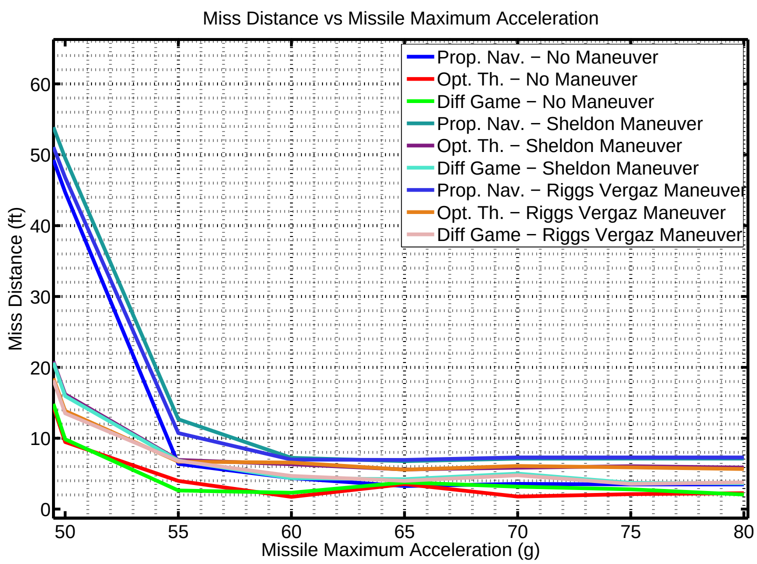

Figure 8.14: Zoomed in Figure 8.13

Inferences.

- As missile maximum acceleration increases, the miss distance decreases.
- This goes well with our intuition that given an higher acceleration advantage for the missile over the target, it is easier for the missile to track down the target.
- Figures 8.7 - 8.14 support our inferences.
- It is also seen that irrespective of the different target maneuver and different missile guidance laws, the above said conjecture seems to hold true.

8.5 Miss Distance Dependence on Initial Missile Mach

Throughout the simulation conducted here in this section, the missile is made to possess proportional guidance law and the target has no maneuver and this scenario is tested by varying the initial missile Mach. The initial flight conditions considered are shown in the Table 8.4.

Flight Conditions Considered:

Flight Parameter	Value	Flight Parameter	Value
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Integration Method	RK-4	Target Range	10000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Target Mode	Const. Velocity	Aspect Angle	135 deg

Table 8.4: Flight Conditions for Miss Distance vs Missile Mach

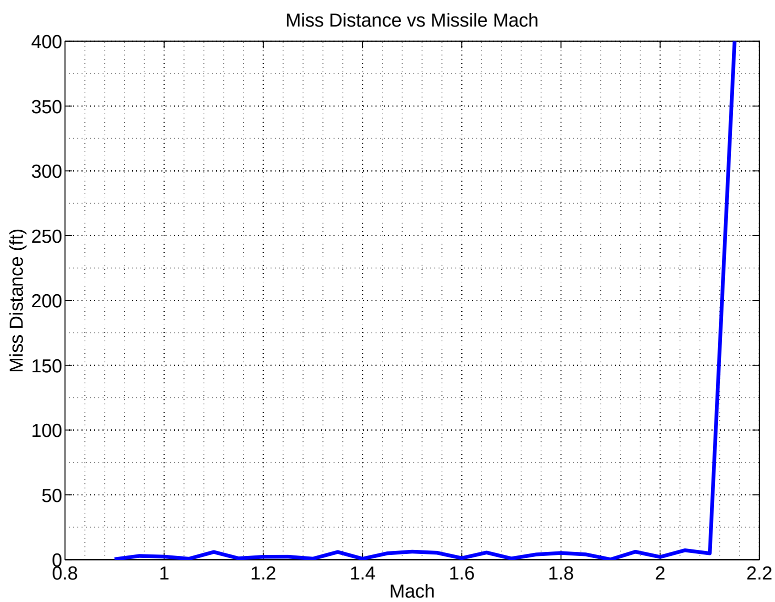

Figure 8.15: Miss Distance vs Initial Missile Mach

Inferences.

- It is intuitive that the missile will track its target if it is given an higher initial velocity.

- But it is also a point of interest to note here that, if the missile initial velocity is very big, e.g. here in our case if it is bigger than 2.15 for the above flight condition considered, the missile permanenetly misses the target because it had travelled probably in the wrong direction initially with higher velocity.
- Missile realizes that the it cannot track down its target as the range keeps on increasing.
- That is not captured here in Figure 8.15 as miss distance is a very big number in those cases.
- This scenario can be thought analogous to a condition where our integration step size is big.
- Reader is referred to the Section 7.3 in Chapter 7.

8.6 Miss Distance Dependence on Target Maneuver

Throughout the simulation conducted here in this section, the missile is made to possess differential game theory guidance and all the three target maneuvers are tested by varying the proportional gain. The initial flight conditions considered are shown in the Table 8.5.

Flight Conditions Considered:

Flight Parameter	Value	Flight Parameter	Value
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Integration Method	RK-4	Aspect Angle	135 deg

Table 8.5: Flight Conditions for Miss Distance vs Target Maneuver

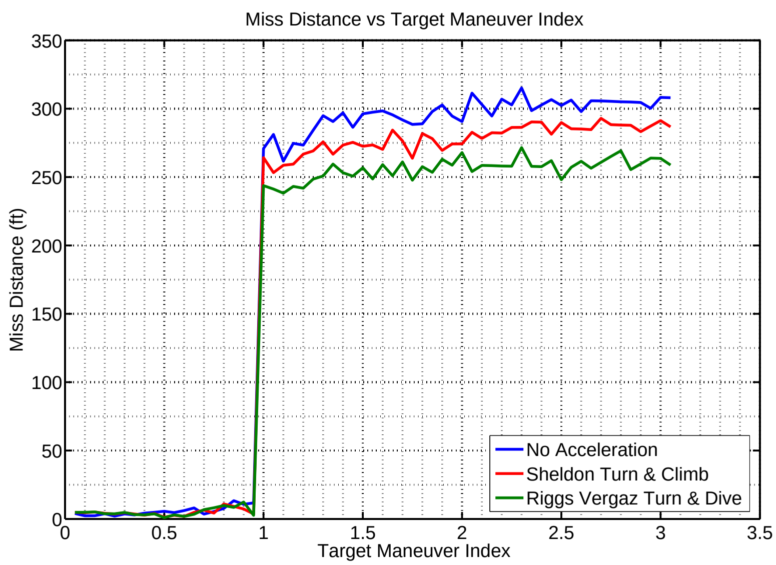

Figure 8.16: Miss Distance vs Target Maneuver

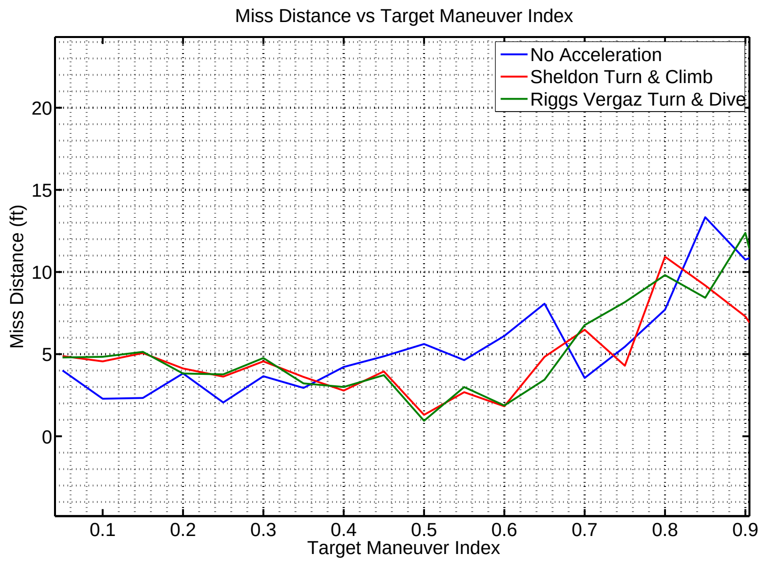

Figure 8.17: Zoomed in Figure 8.16

Inferences.

- From Figure 8.16 & 8.17, it is clear that as the target maneuvers more, it is difficult for the missile to intercept it.
- The degree of target maneuverability is measured using an unitless quantity called *"Target Maneuver Index"* or simply *"Maneuver Index(MI)"*. We know that from equation 4.37.
- It is clear that as long as Maneuver Index is small, the differential game thoery guidance is going to behave well as smaller MI indicates target maneuvering very less.
- But as MI approaches 1, its singular point, the miss distance starts to increase.
- Thus, for all values of MI > 1, which indicates the target maneuvering more, the miss distance is bad (i.e. the missile misses the target) irrespective of the target's intelligence.

- This behaviour is excellently captured in the Figure 8.16.
- For more information on this concept, the reader is referred to the relevant GNC texts [51] and [52].

8.7 Miss Distance Dependence on Target Aspect

Throughout the simulation conducted here in this section, the missile is made to possess proportional guidance and the target doesn't maneuver. This flight condition is tested by varying the initial target aspect with respect to the missile. The initial flight conditions considered are shown in the Table 8.6.

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Target Range	1-10 kft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Target Mode	Const. Velocity	Integration Method	RK-4

Table 8.6: Flight Conditions for Miss Distance vs Target Aspect

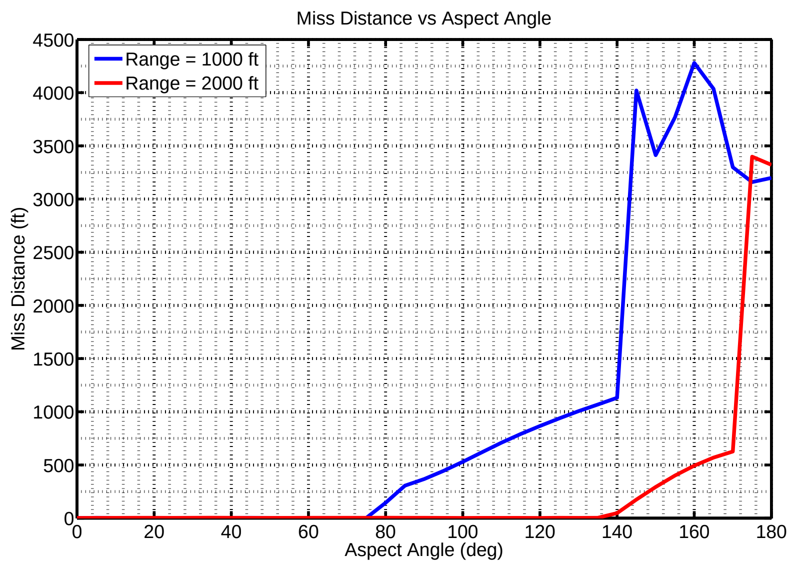

Figure 8.18: Miss Distance vs Target Aspect - Range = 1 kft, 2 kft

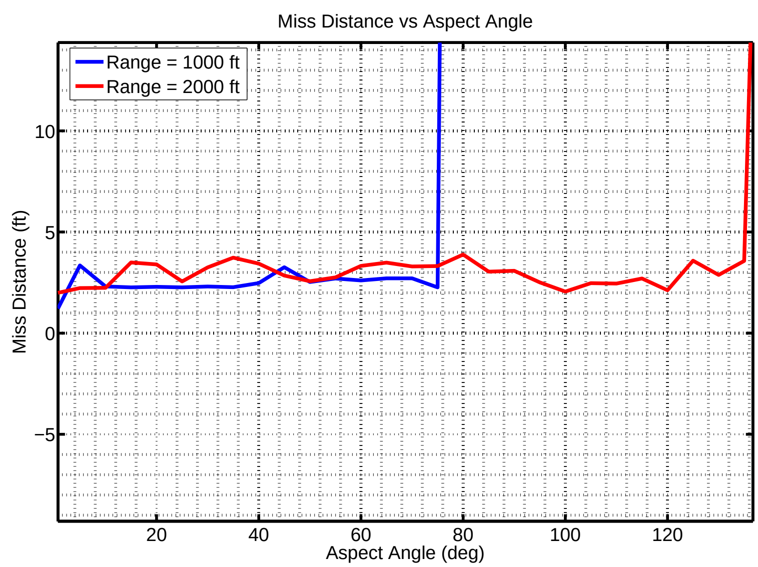

Figure 8.19: Zoomed in Figure 8.18

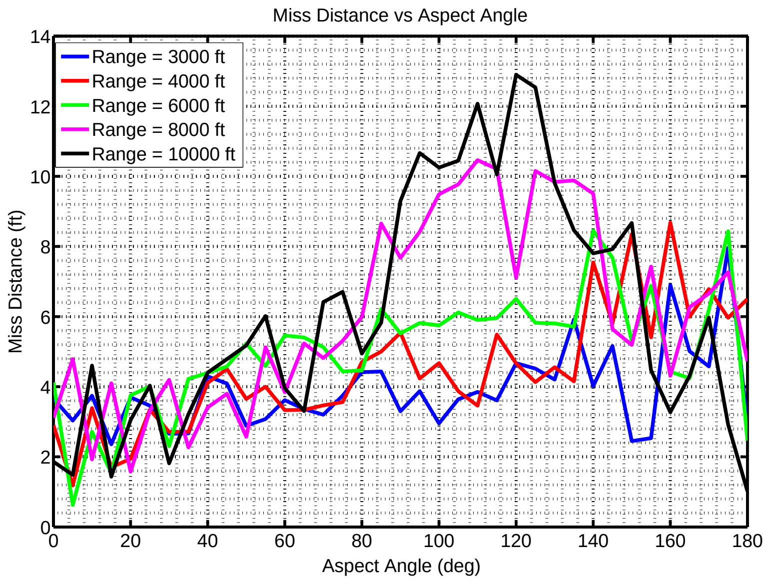

Figure 8.20: Miss Distance vs Target Aspect - Range = 3 kft - 10 kft

Inferences.

- For a given initial target range, a certain range of aspect angles would be favorable for a missile.
- Consider a missile having a target coming towards it at an Aspect of 180 deg which is the most favorable target orientation for the missile to hit it exactly.
- This favorable aspect varies with the range.
- A target which is at a closer range will not result in a hit if it is oriented at as aspect of 180, as missile might miss it at the very initial stage of its flight.
- Instead, 0 degree aspect in this case would result in an hit.
- Similarly a farther target if it is oriented at 180 Aspect (read it as "Head-on collision" case or coming towards the missile to get killed !) will result in a hit

condition, while 0 deg aspect would clearly result in an miss since missile is not guaranteed to succeed in a tail end chase with target being very far.

- This phenomenon is clearly captured in Figures 8.18 - 8.20.
- As we increase the range, all aspect angles from 0 to 180 degree is expected to give an hit.
- But going by the basic Physics, if we go on increase the range, there will be a point where missile will start to miss the target.
- This is explained in below Section 8.8 and this also motivates the work done in the Chapter 9.
- Thus Aspect is a very important parameter in missile-target engagement and it depends upon initial target range.

8.8 Miss Distance Dependence on Initial Target Range

Throughout the simulation conducted here in this section, the missile is made to possess proportional guidance with proportional navigation gain of about 2.3 for both the channels and the target doesn't maneuver. This flight condition is tested by varying the initial target range with respect to the missile. The initial flight conditions considered are shown in the Table 8.7. Aspect of 135 degrees is considered here.

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Guidance Law	Proportional
Initial Target Mach	0.8999	Time Constant	0.5 sec
Target Mode	Const. Velocity	Aspect Angle	135 deg

Table 8.7: Flight Conditions for Miss Distance vs Target Range

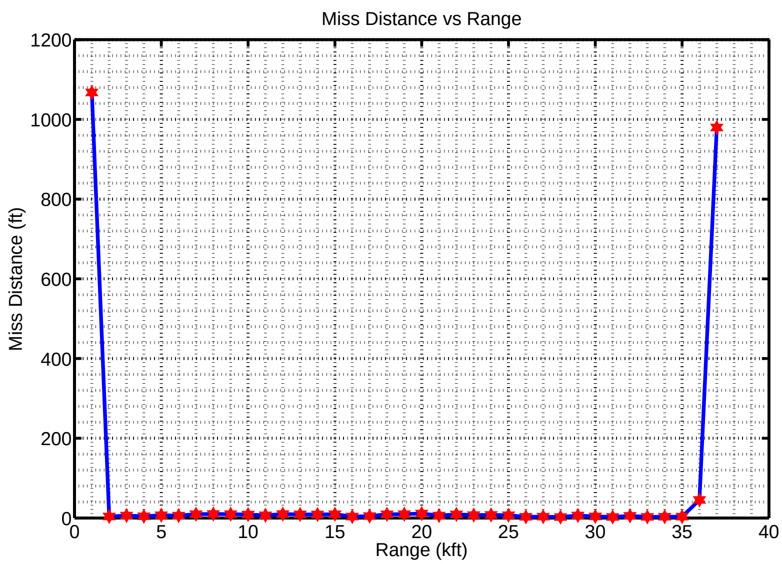

Figure 8.21: Miss Distance vs Initial Target Range

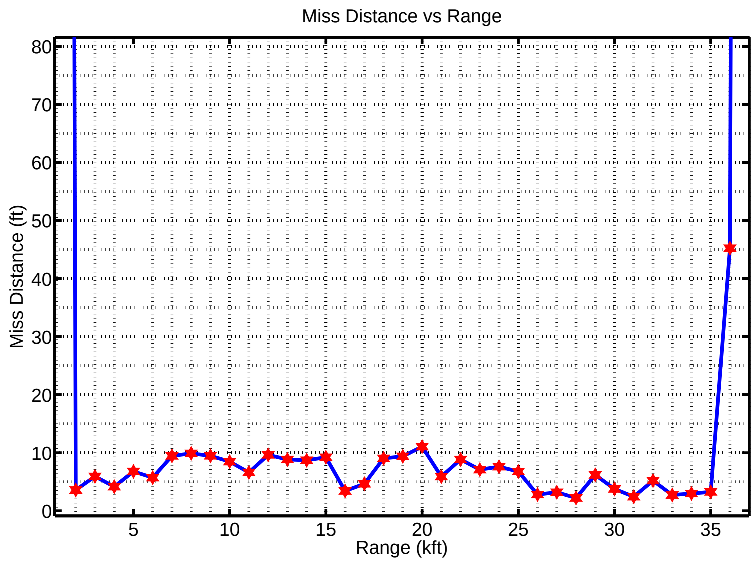

Figure 8.22: Zoomed in Figure 8.21

Inferences.

- For the given aspect, from Figures 8.21 & 8.22 reveal that smaller ranges end up with high miss distance as missile might initially move in wrong direction and miss the target completely.
- As range increases, missile can catch up with target's range and its orientation and thus our miss distance in those range is very small.
- Finally as the target is far away, missile will start to run out of fuel while catching up with the target and misses it.
- This clearly motivates the kill estimation work done in Chapter 9.
- Referring to the previous Section 8.7, for different target aspect, the hitting ranges including from closest hit till the farthest hit will vary.

8.9 Summary and Conclusions

In this chapter, the miss distance profiles with respect to different missile/target engagement parameters were discussed. This will give us a fair idea about how the miss distance varies as we vary different missile-target engagement parameters. This shall definitely help us in estimating the capability of a BTT missile.

Chapter 9

KILL ZONE COMPUTATION & ANALYSIS

9.1 Introduction and Overview

Launching missiles effectively with high success rate is a complex resource allocation problem. The cost of manufacturing and operating each missile is realtively very high and so they have to be launched only when their success is guaranteed. If the hunting area of the missile with its full capability is known, then any target spotted within the hunting area can be successfully intercepted by launching the missile. The purpose of this chapter is to illustrate the hunting zone of the BTT missile considered in this research. Given a thrust profile and fixed initial conditions for both missile and target, the analysis made in this chapter will explain about the zone of kill where the missile will successfully intercept the target. *Kill Zone* is a closed area on the space which includes all possible target's starting position, which will result in the missile intercepting the target. Estimating such a big area in 2D space will need a powerful estimation algorithm for faster computation and binary search algorithm [71] & [72] is used here. The research here has been restricted to 2D space by assuming both the missile and target initially start at the same altitude with respect to each other. The high fidelity environment used throughout the simulation in this research is employed to study the Kill Zone profile with respect to different missile/target engagement parameters using ideas described in the relevant GNC textbooks [51] and [52]. Also the work done in this chapter takes its motivation from the Section 8.8 in the Chapter 8.

Conventional warheads carried by the missile have a circular blast radius of about

20 ft [70]. Thus any simulation resulting in a final miss distance less than 20 feet is taken granted as a hit and kill zone estimation is developed as per this logic. Each section in this chapter will have information about the flight conditions considered, the result and its inference. The chapter is organized as follows: Section 9.2 will give an brief overview about the binary search algorithm and its usage here in our simulation. Section 9.3 analyses the effect of altitude variation on the estimated kill zone. Section 9.4 throws light on effect of varying the maximum acceleration capability of the missile over on the estimated kill zone. Section 9.5 discusses the effect of initial missile speed on the estimated kill zone. Section 9.6 discusses the effect of initial target speed on the estimated kill zone. Section 9.7 shows how the estimated kill zone varies when the target's orientation with respect to the missile measured in terms of *Aspect*. Section 9.8 will briefly discuss the estimated kill zone change when the proportional gain is varied. Here the missile is assumed to possess Proportional Navigation guidance law to intercept the target. Finally Section 9.9 summarizes and concludes the work done in this chapter and gives a rough idea about estimating the missile's area of kill using above analyses.

9.2 Binary Search Algorithm

When we have an infinite 2D space, it is very important to chose a proper algorithm for finding out the kill zone area in a shorter span of time. Naturally binary search will eliminate half of the unwanted space in each and every iteration and help us to converge faster towards the solution. As per the current simulation used here, the time required to obtain a kill zone for one flight condition is approximately 1 minute. For knowing more about the binary search algorithm, the reader is referred to [22], [71] and [72]. Since we are assuming both the missile and target to start at the same altitude, the search space reduces to 2D. Now the 2D space is divided radially into 360 rays. The challenge here was to find the first hit point and last hit point along each ray. The following algorithm was used.

Algorithm Steps.

1. Initially the algorithm is started by placing the target to be 100 ft away from missile along the 180 degree ray.

2. It is assumed that below range of 100 ft, it is pointless to launch a missile against a target.

3. Now if it is a hit, we double the range and search for a miss or if it is a miss, we take the average between current hit range and miss range.

4. Thus along a ray, we would find the first hit position.

5. Then the algorithm is restarted with twice the current hit range looking for final hit range.

6. After averaging and converging to an hit range which differs from miss range by just 100 ft, we stop the algorithm.

7. This idea is repeated for all the rays. Each ray can be incremented in steps of 2 or 5 degrees to suit the degree of accuracy needed.

8. To speed up the operation, the previous ray final hit range is used in the current ray final hit range estimation using the motivation from the continuity idea.

9. Finally only the hit positions data are stored. While post processing the data, we prepare an array of initial hit positions and final hit positions along each ray and finally plot them using the *"boundary"* command in MATLAB.

10. The above process is repeated by varying one of the flight condition parameter and the estimated kill zone is plotted for that parameter variation.

9.3 Kill Zone Dependence on Initial Engagement Altitude Variation

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-10000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Azimuth Angle	0 deg	Aspect Angle	0 deg

Table 9.1: Flight Conditions for Kill Zone vs Engagement Altitude

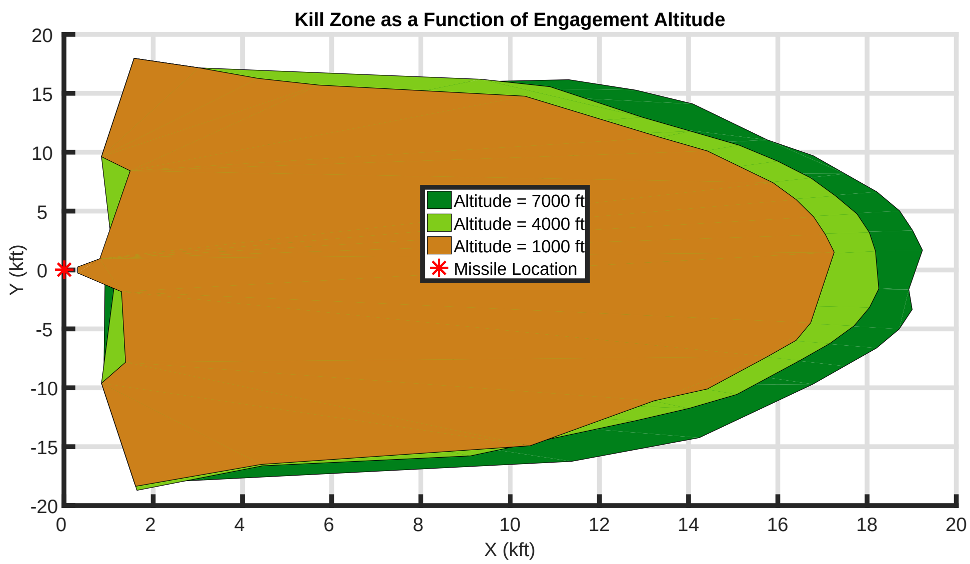

Figure 9.1: Kill Zone vs Engagement Altitude (Lower Altitudes)

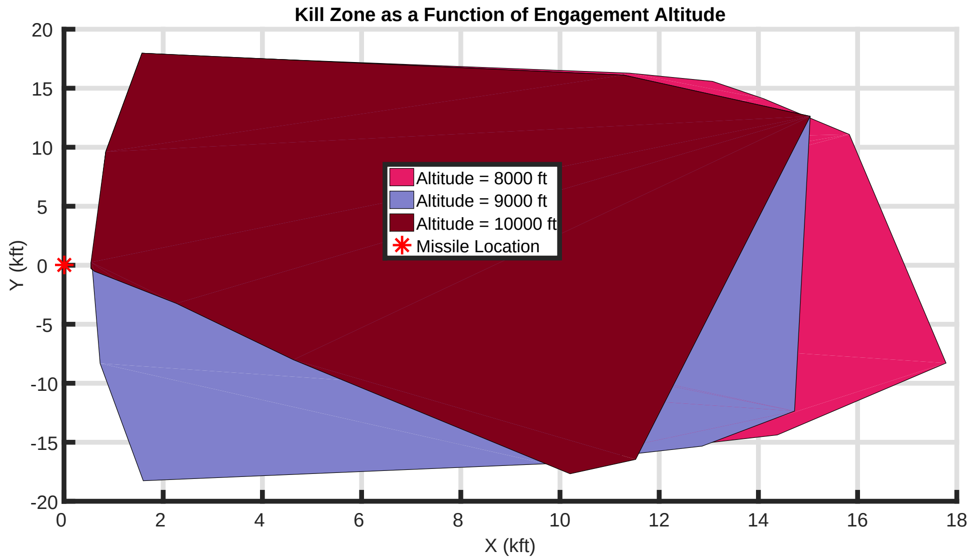

Figure 9.2: Kill Zone vs Engagement Altitude (Higher Altitudes)

Inferences

- From the idea obtained through referring Figure 8.6, it is generally observed that as the altitude increases, the miss distance increases and hence the kill zone should become narrower and eventually smaller.

- Results shown in 9.2 correlates well with the results published in [17]. Because air-density decreases with increasing altitude, it is expected that the missile lose their aerodynamic effectiveness at higher altitudes because of the inability of the fins to control the missile in the thin air of upper atmosphere.

- The same idea motivates that missile should perform well in lower altitudes and that facct is supported by Figure 9.1.

- Above Figures 9.1 & 9.2 exhibit similar pattern as shown in Figure 8.6, where below 10kft the missile has good chance and as we start increasing altitude from 10kft, the missile chances of hitting target becomes bad.

- Thus as the engagement altitude increases, the final miss distance increases [17] and hence the kill zone area decreases.

9.4 Kill Zone Dependence on Missile Maximum Acceleration Variation

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-10000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Azimuth Angle	0 deg	Aspect Angle	0 deg

Table 9.2: Flight Conditions for Kill Zone vs Missile Maximum Acceleration

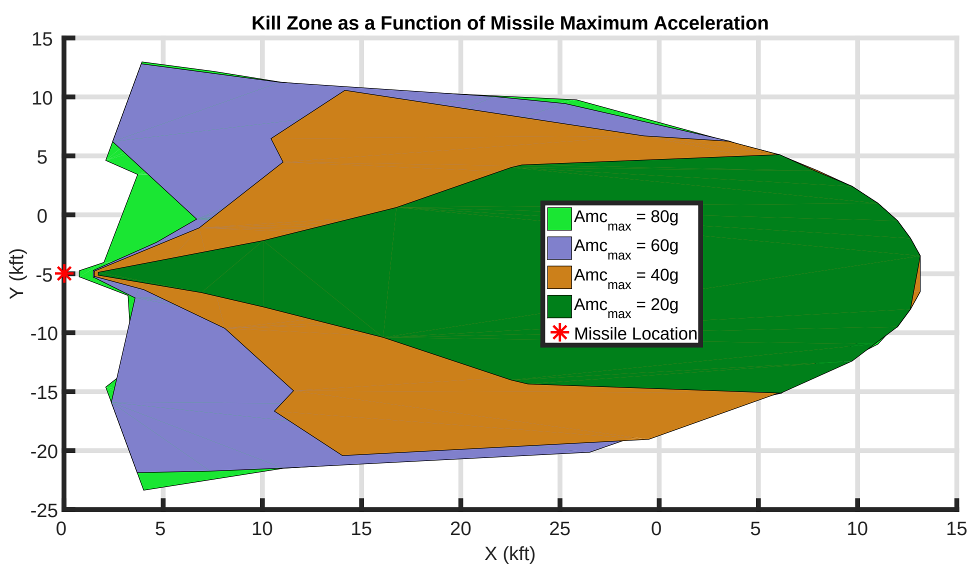

Figure 9.3: Kill Zone vs Missile Maximum Acceleration

Inferences

- Ideally giving a higher acceleration advantage of missile over the target will help missile intercept the target easily.
- The above intuition is well supported by results shown in Figure 9.3.
- The kill zone seems to grow as maximum missile acceleration is increased.

9.5 Kill Zone Dependence on Initial Missile Mach Variation

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-10000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Azimuth Angle	0 deg	Aspect Angle	0 deg

Table 9.3: Flight Conditions for Kill Zone vs Missile Mach

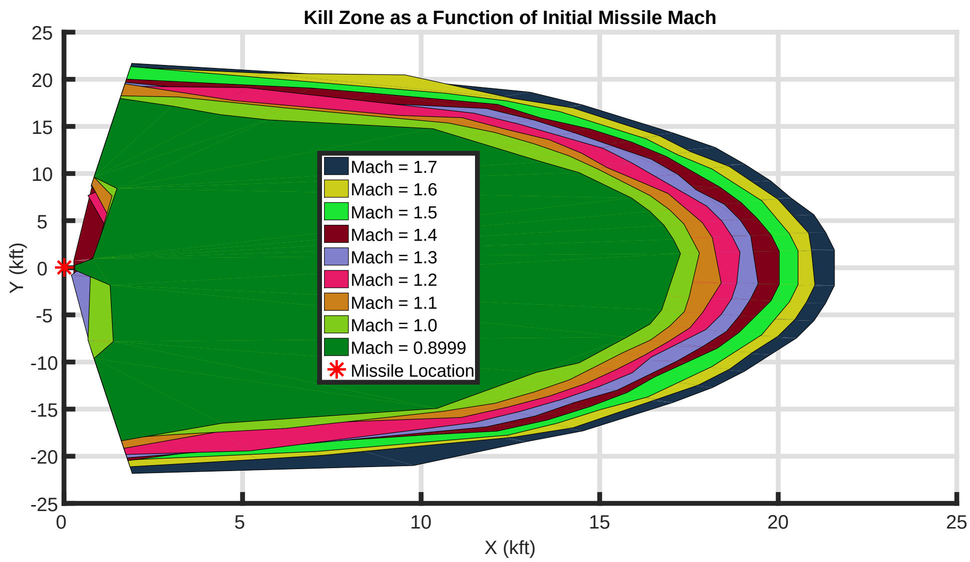

Figure 9.4: Kill Zone vs Initial Missile Mach

Inferences

- It is observed from Figure 9.4 that as the initial mach of the missile increases, the kill zone grows.
- This is because the missile is able to travel faster and so it can intercept the target quickly.
- As the speed of missile increases, the missile flight time decreases and hence with a greater mach, kill zone area increases.

9.6 Kill Zone Dependence on Initial Target Mach Variation

Flight Conditions Considered:

Flight Parameter	Value	Flight Parameter	Value
Missile Max. Accel.	80 g	Initial Height	-10000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Azimuth Angle	0 deg	Aspect Angle	0 deg

Table 9.4: Flight Conditions for Kill Zone vs Target Mach

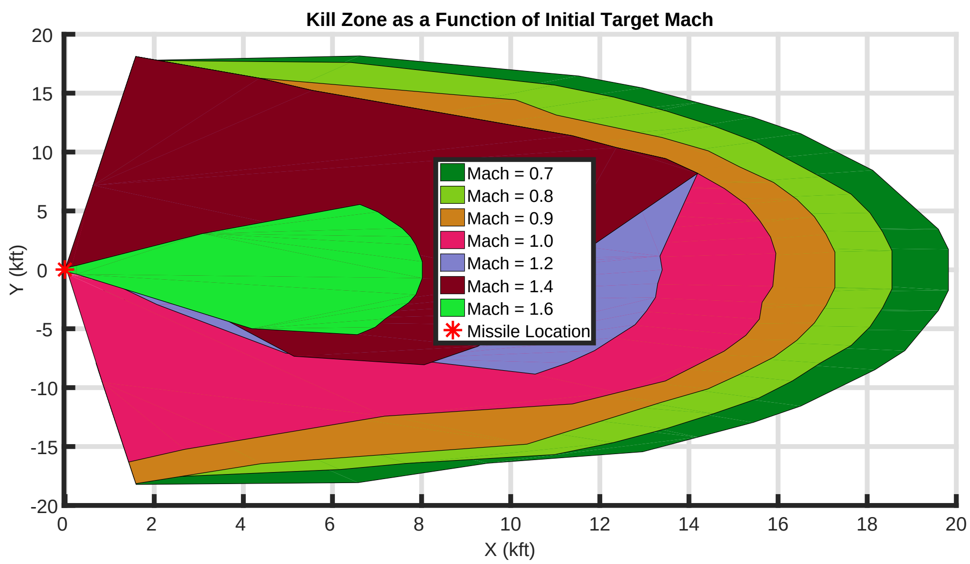

Figure 9.5: Kill Zone vs Target Mach

Inferences

- It is observed from Figure 9.5 that as the initial mach of the target increases, the kill zone decays.
- This is because the target is able to travel faster and so it can evade the missile quickly.
- As the initial speed of target increases and even with the missile initial as-

pect being correct, there are higher chances that the target evading the missile because of its higher velocity and thus kill zone area decreases.

9.7 Kill Zone Dependence on Initial Aspect Variation

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-10000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Time Constant	0.5 sec
Azimuth Angle	0 deg	Aspect Angle	0 deg

Table 9.5: Flight Conditions for Kill Zone vs Target Aspect

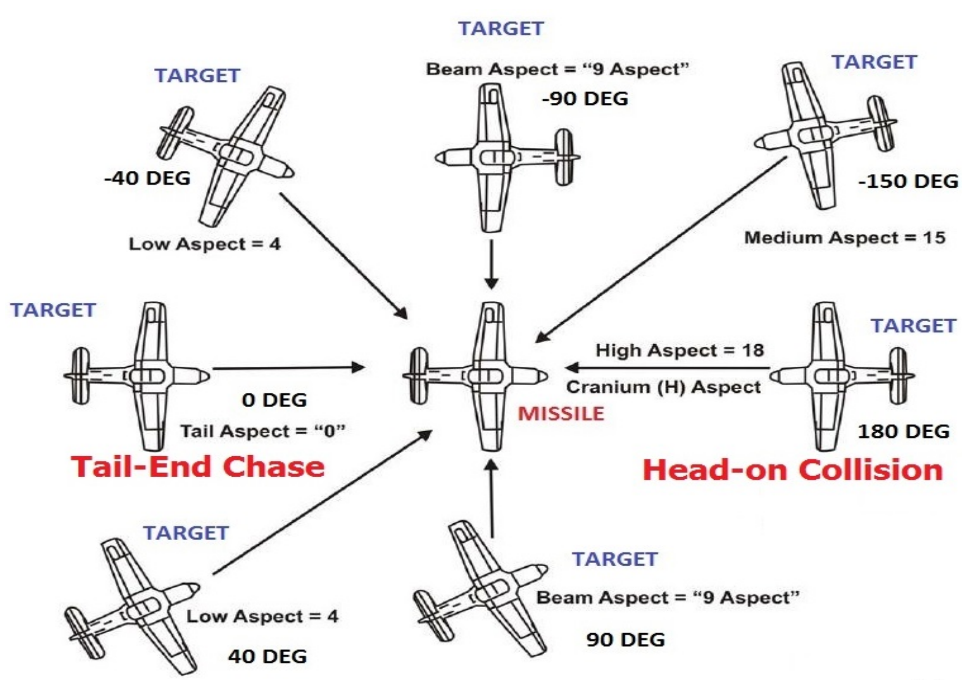

Figure 9.6: Target Aspect Orientation With Respect To Missile

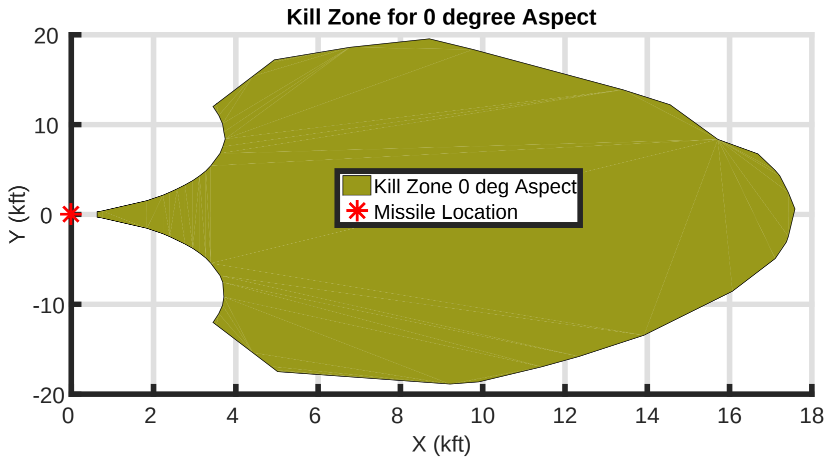

Figure 9.7: Kill Zone For 0 Aspect (Tail-End Chase)

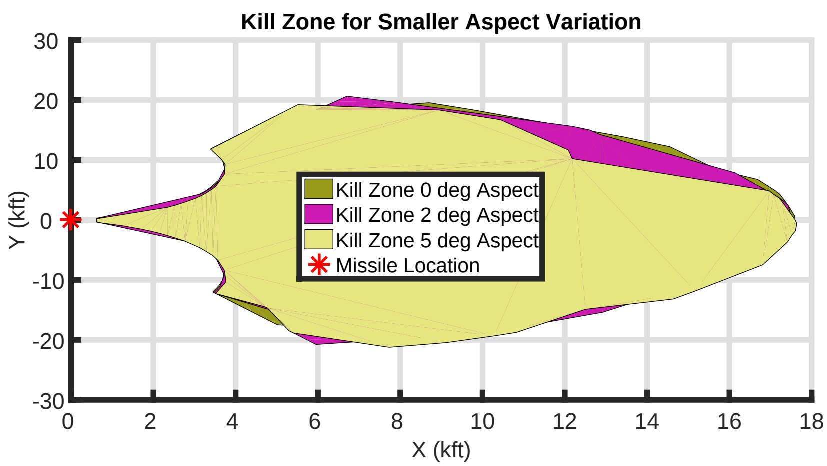

Figure 9.8: Kill Zone For Small Target Aspect Variation

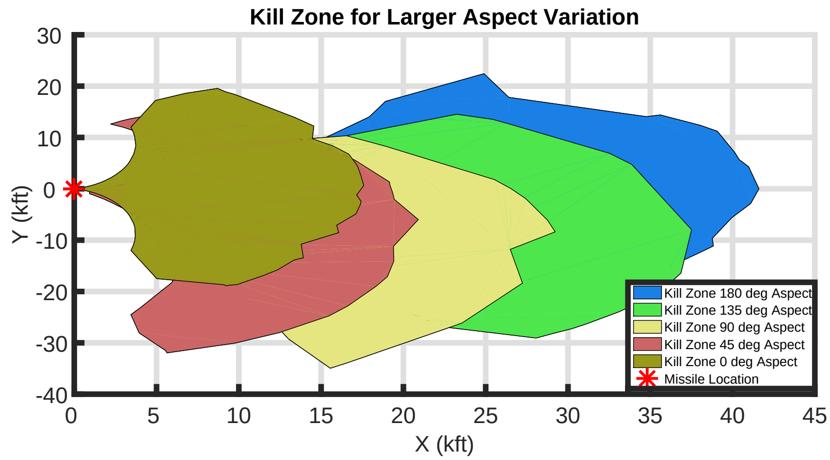

Figure 9.9: Kill Zone - Tail-End Chase to Head-on Collision

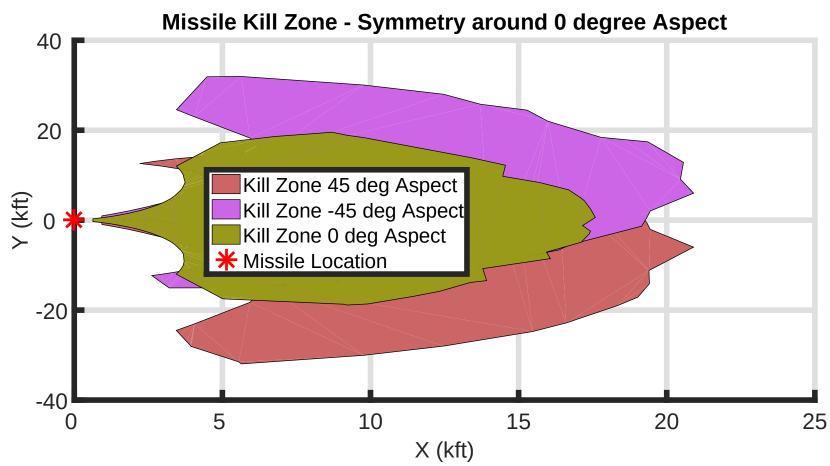

Figure 9.10: Kill Zone 45 Degree Symmetry Aspects

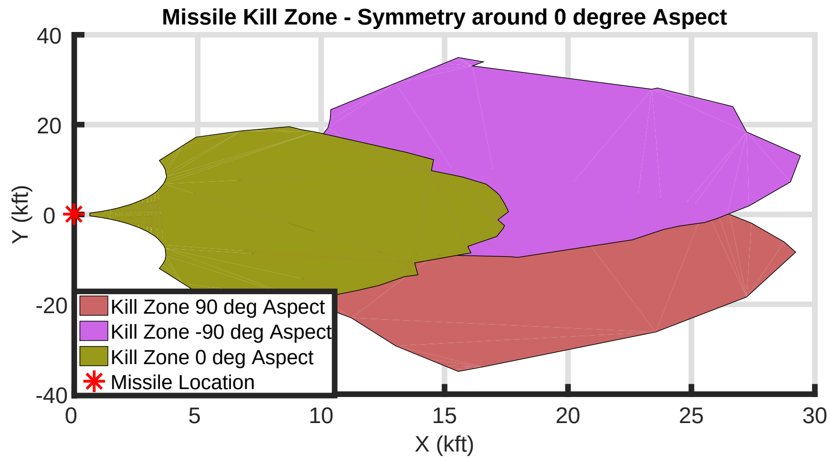

Figure 9.11: Kill Zone 90 Degree Symmetry Aspects

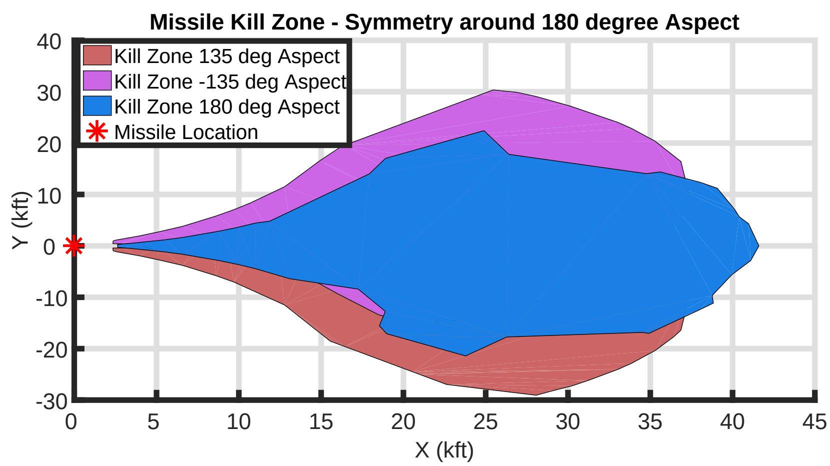

Figure 9.12: Kill Zone 135 Degree Symmetry Aspects

Inferences

- Target orientation with respect to the missile, referred here as the "Aspect Angle" is the most influential factor that governs the shape and size of the kill zone.

- The reader is referred to Figure 9.6 to see how target orientation varies from 0 degree to 180 degrees with respect to the missile.

- Consider the following scenarios which explains the declaration made just above.
  - 0 degree Aspect - Called as the "Tail-End Chase" orientation, has 2 types namely missile tailgating the target and vice-versa. Refer Figure 9.7.
  - 180 degree Aspect - Called as the "Head-On Collision" orientation, has 2 types namely missile coming opposite towards the target and vice-versa
  - In above cases, even though the missile is equipped with best possible Mach and Maximum Acceleration around 80g, it can miss the target by huge range if the target aspect is not favorable.
- It is possible for the shorter ranges to be missed (with unfavorable aspect) and longer ranges to be hit successfully (with favorable) and this behaviour can be easily seen through the Figure 9.9 which shows how the kill zone grows as we move from tail-end aspect to head-on collision aspect.
- Figure 9.8 shows how aspect angle changes the kill zone in smaller steps.
- Another important phenomenon to observe is the existence of "Symmetry" around the 0 degree aspect and 180 degree aspect.
- Presence of Symmetry as shown in Figures 9.10, 9.11 & 9.12 shows that missile will treat the target as same with the target being oriented with respect to it at 45 degrees or -45 degrees.
- With unfavored aspect, best missile capabilities can go in vain and with correct (proper) aspect, even tougher ranges could be covered and hence the "aspect" is an important factor in determining the missile's success (kill zone).

9.8 Kill Zone Dependence on Proportional Gain Variation

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Target Mode	No Maneuver
Initial Target Mach	0.8999	Missile Guidance	Prop. Nav.
Azimuth Angle	0 deg	Aspect Angle	0 deg

Table 9.6: Flight Conditions for Kill Zone vs Proportional Gain

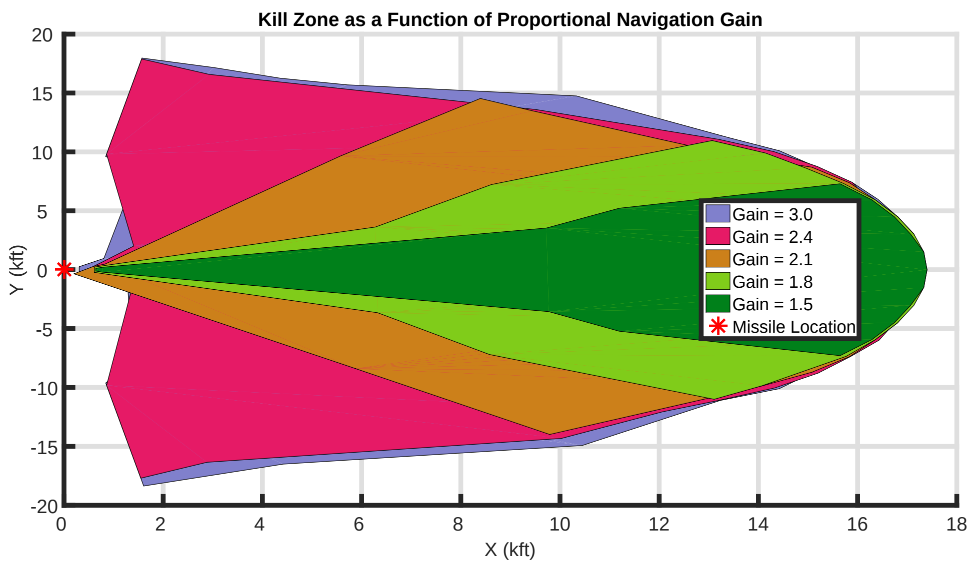

Figure 9.13: Kill Zone vs Proportional Gain

Inferences

- It is observed from Figure 9.13 that as the proportional navigation gain of the missile increases, the kill zone grows.
- This is because higher guidance gain will ensure the error between missile & target's position (interpret it as "range") sgoes to zero quickly as appropriate commanded variables are generated proportional to the error in position.

- The result presented above actually agrees well with the ideas explained in section 8.2 contained in the chapter 8.

- However, there is a caveit here that the proportional gain cannot be increased arbitrarily as a higher guidance gain will destabilize the guidance loop. This phenomenon is captured in Figure 8.1 and the same behaviour is expected here in kill zone estimation too.

9.9 Summary and Conclusions

In this chapter, the estimation of Kill Zone with respect to different missile/target engagement parameters were discussed. This will give us a fair idea about when to & when not to launch the missile when its initial conditions are known. Future research will involve searching in 3D space with same or different algorithms. Also, a complex target like 6DOF can be used instead of 3DOF to study the variation. This also opens a new area of research where missile tracking multiple targets based on their lethality and need of the hour. Thus the work done in this chapter has an excellent scope for future research.

Chapter 10

MISSILE-TARGET 3D ANIMATION USING MATLAB

10.1 Introduction and Overview

The purpose of this chapter is to illustrate the design of 3D animation using VRML toolbox in MATLAB. Earlier work done by [5] had just 2D simulation results. In order to see how a real missile would intercept its target, the need for 3D animation arises there. MATLAB offers 3D simulation using VRML toolbox. The objective was to connect the simulation part in MATLAB with the animation module as described in [9]. The simulation part was coded in MATLAB with object oriented programming design methodology and simulation updated the animation at each and every iteration. Different viewpoints showed how the objects would move in real time. This animation enables the visualization of missile-target engagement in real time scenario. A key goal of the chapter is justifying the fact if the missile simulation is going to work fine in this 3D animation, it is expected to work in real time scenario just like it behaved in the computer simulation. As such, the chapter illustrates how to input the inital conditions of missile and target in an interactive Graphical User Interface(GUI) and view the real time 3D animation with all the simulation running in the background in MATLAB.

Remainder of this chapter is organized as follows. Section 10.2 will show how to prepare GUI, simulate the initial flight conditions. Section 10.3 will discuss the process of updating the animation using the simulation details. Switiching between different viewpoints is also explained here. Section 10.4 will discuss the results and animation obtained by running the MATLAB application. Finally Section 10.5 will summarize and concludes the work explained in this chapter.

10.2 Interactive GUI Developement

The Graphical User Interface Design Environment (GUIDE) toolkit in the MATLAB can be used to develop several interactive GUIs with MATLAB script code running in the background. The idea is to build an interactive GUI for the Missile-target engagement application through which the initial flight conditions can be entered by the user.

Figure 10.1: Missile-Target Engagement - MATLAB GUI

The GUI developed for this application looks the Figure 10.1, shown above. Initial Flight conditions for the missile-target engagement includes the following parameters given in table 10.1:

Missile Guidance	Aspect	Integration Method	Range
Missile Max. Accel.	Step Size	Proportional Gain	Maneuver Index
Target Mach	Elevation	Target Maneuver	Azimuth
Target Altitude	Missile Mach	Missile Altitude	Target Tau

Table 10.1: GUI Flight Conditions Selection for Missile-Target Engagement

- Once the above parameters are selected, the *Load Initial Conditions* button is hit.
- Internally MATLAB will create Missile & Target Class objects and loads the user entered initial flight conditions.
- Then the *Run Animation* button is hit which will start the Missile-Target Engagement simulation and each and every step of the simulation is updated using an 3D animation which was developed using VRML toolbox in MATLAB.
- Once the animation, i.e., the simuation gets over, final statistics are displayed and the post flight data can be analyzed by clicking *Plot Data* button.
- Thus in one single GUI screen, the user will be able to enter their desired initial flight conditions, see the animation to get a virtual feel of how the missile would intercept the target in real time and conclude by seeing all the final statistics and the post flight data in the same screen.

10.3 3D Animation using MATLAB VRML Toolbox

VRML (Virtual Reality Modeling Language) toolbox in MATLAB can be used to make different interactive animations. Here in this research, missile-target engagement can be visualized using the features offered by the MATLAB. Since the entire MATLAB application has been coded in an object oriented architecture, the same program can be easily extended to multiple missile-target engagement just by using new object for Missile and Target class. Thus the entire simulation data has to be communicated to animating world in a way that it understands. Once that is achieved, then whatever happens in simulation can be seen in real time 3D animation as animation is just updating the simulation flow. The motivation for going for a 3D animation is to visualize how the missile-target engagement would happen in a real world scenario (which would be difficult for us to see in real time). And given a better modeling and design environment, it can be believed that real missile would exactly behave and intercept the target like it does in 3D animation. Several aerospace companies spend billions of dollar in modeling the environment so that things if they work in simulation well are expected to work almost the same way in real world.

Thus to prepare an interative 3D animation we require the following,

1. Nice and fancy 3D Background
2. Missile Object
3. Aircraft Object
4. Different Viewpoints
5. Proper interface between VRML editor and MATLAB - Could be either through MATLAB or SIMULINK. MATLAB interface is used in this research.

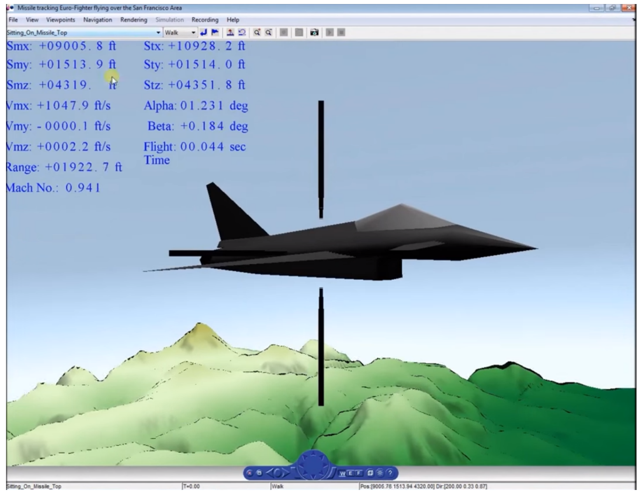

Figure 10.2: Missile-Target Engagement - 3D Animation

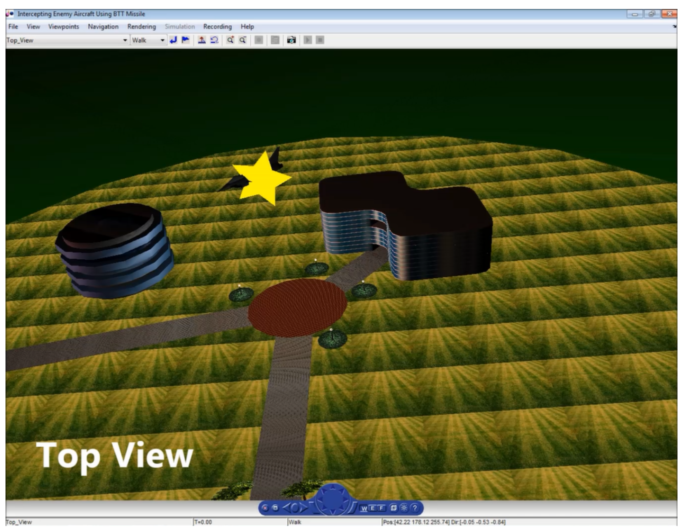

Figure 10.3: Missile-Target Engagement - 3D Animation Top View

3D World Editor - VRML Editor which comes as a part of MATLAB VRML toolbox was used to develop the 3D animation environment. There are other commercial VRML editors available in the market for cheaper costs like V-Realm Builder, 3DStudio, Blender etc... 3D World Editor was good enough to prepare the animation in this research. VRML files have ".wrl" format which is a short form for "world". Initially the 3D background was developed, then the missile and target objects were properly placed in the 3D background in such a way as to mimick the initial conditions given through the interactive MATLAB GUI. The VRML toolbox in MATLAB already has different aerospace objects like aircraft, missile, helicopters, rockets etc. One such missile and target aircraft from that repository is being used in this research. Then the different viewpoints can be made according to the user requirements. A missile cockpit viewpoint along with other 4 viewpoints were developed for this research. For example, refer figure 10.3 for a top viewpoint. Cockpit viewpoint as shown in Figure 10.2 will give a real-time feel as if we were sitting inside the missile and riding it(Although never done in real life!). Different flight parameters of both missile and target can be tracked as the animation progresses. This gives a real-time feel like traveling in a fighter aircraft being a pilot. Given the power of GPUs nowadays, this missile-target engagement simulation can be made much faster, while the current research is done without the usage of GPUs. Also, if the MATLAB is made aware of intelligently using the GPUs, then this research can get really interesting. There are other better softwares like Blender, Maya, 3DS Max which is capable of creating content rich 3D object files in different format. As of now, good resource files in ".wrl" format are really less available. MATLAB recently extended the 3D animation capability to ".x3d" format too. Similarly there are ways to import 3D object files from the 3D authoring worlds like AutoCAD, CATIA, Solidworks and so into the MATLAB and create animations with them.

10.4 Simulation Results & Analysis

Post flight analysis from simulating the conditions from table 10.2 are plotted and comparison of MATLAB results with C program [5] is presented. The plots ranging from Figure 10.4 - 10.14 show that both C and MATLAB simulations are really close to each other, depicting that MATLAB program is as accurate as C program, eventhough it is written with different programming style and interpolating techniques for calculating aerodynamic coefficients.

Flight Conditions Considered:

Flight Parameter	**Value**	**Flight Parameter**	**Value**
Missile Max. Accel.	80 g	Initial Height	-1000 ft
Initial Missile Mach	0.8999	Target Range	2000 ft
Initial Target Mach	0.8999	Missile Guidance	Optimal Control
Target Maneuver	Sheldon	Aspect Angle	0 deg

Table 10.2: Flight Conditions for MATLAB & C Simulations

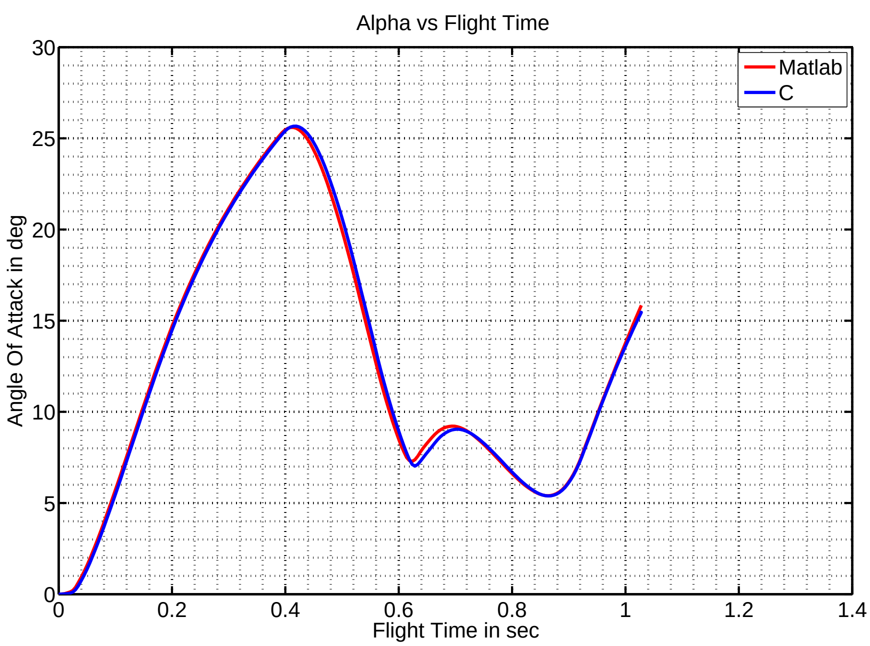

Figure 10.4: Alpha Profile - MATLAB & C Simulations

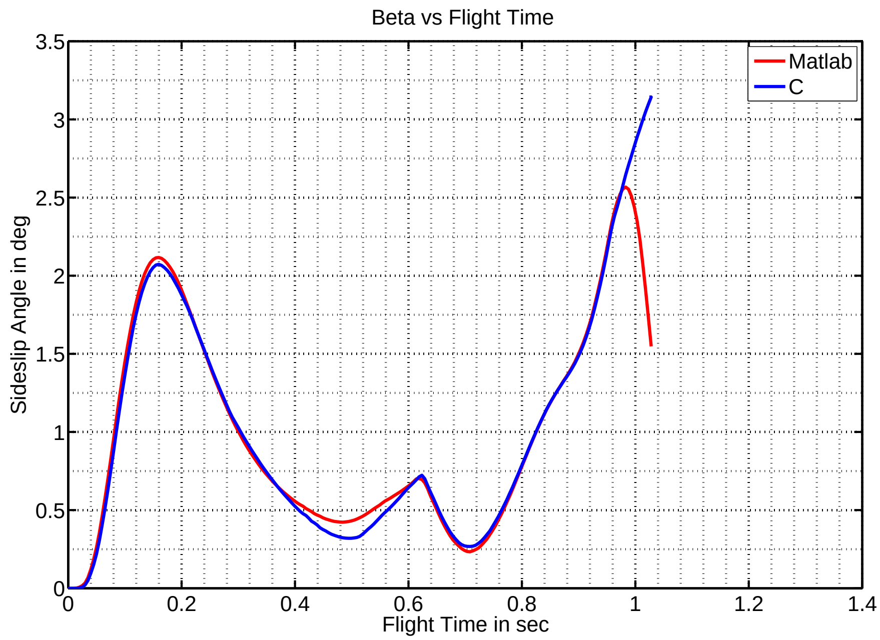

Figure 10.5: Profile - MATLAB & C Simulations

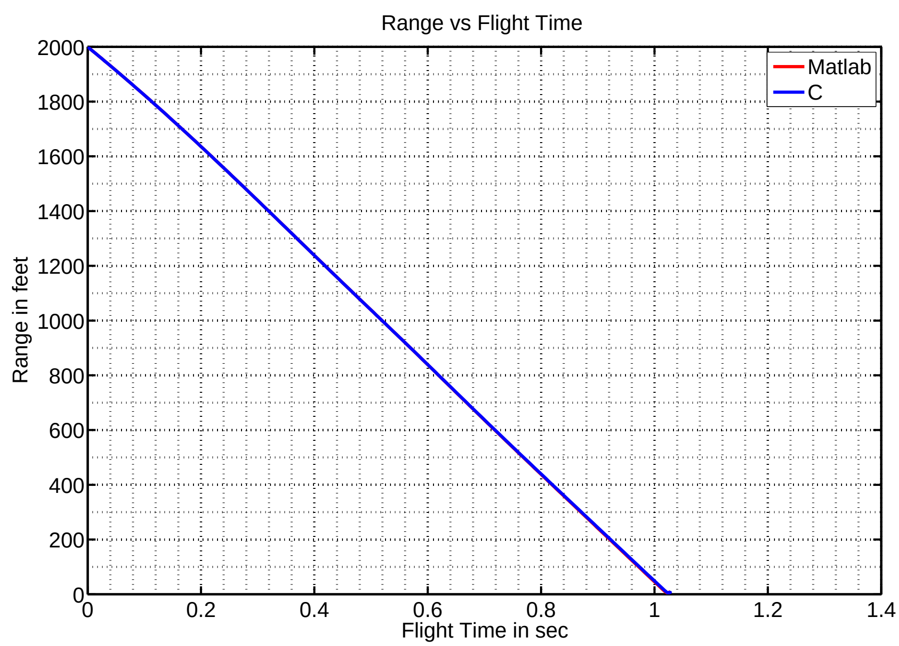

Figure 10.6: Profile - MATLAB & C Simulations

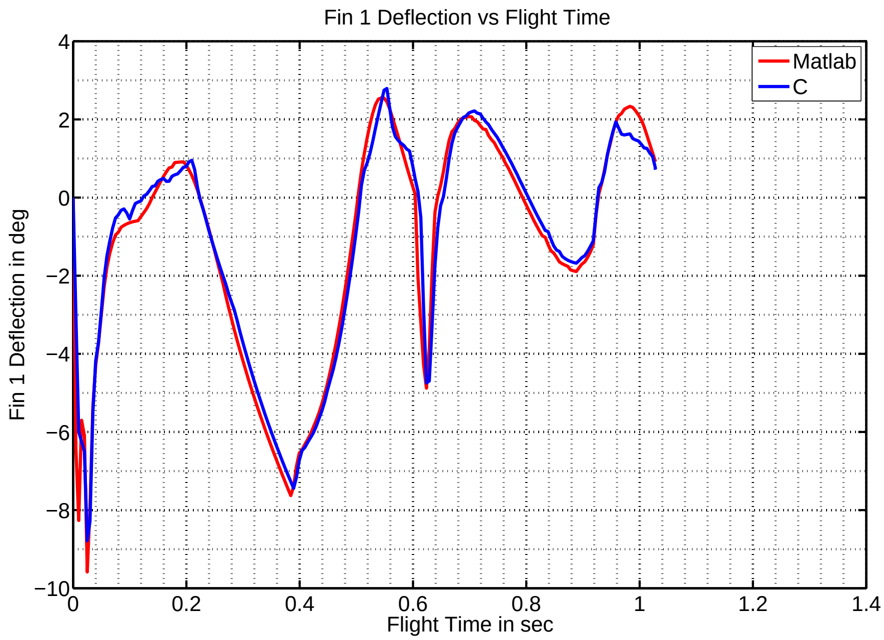

Figure 10.7: Fin 1 Profile - MATLAB & C Simulations

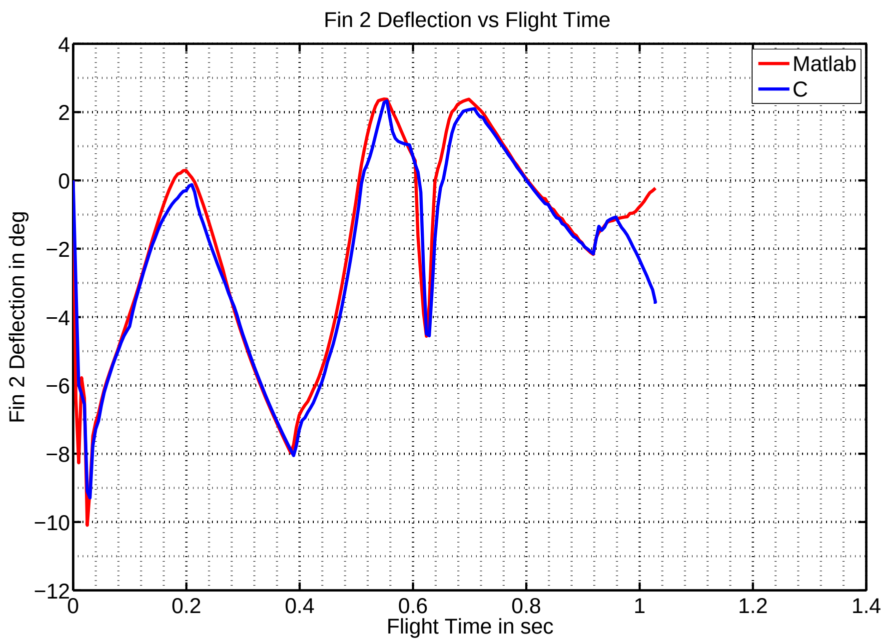

Figure 10.8: Fin 2 Profile - MATLAB & C Simulations

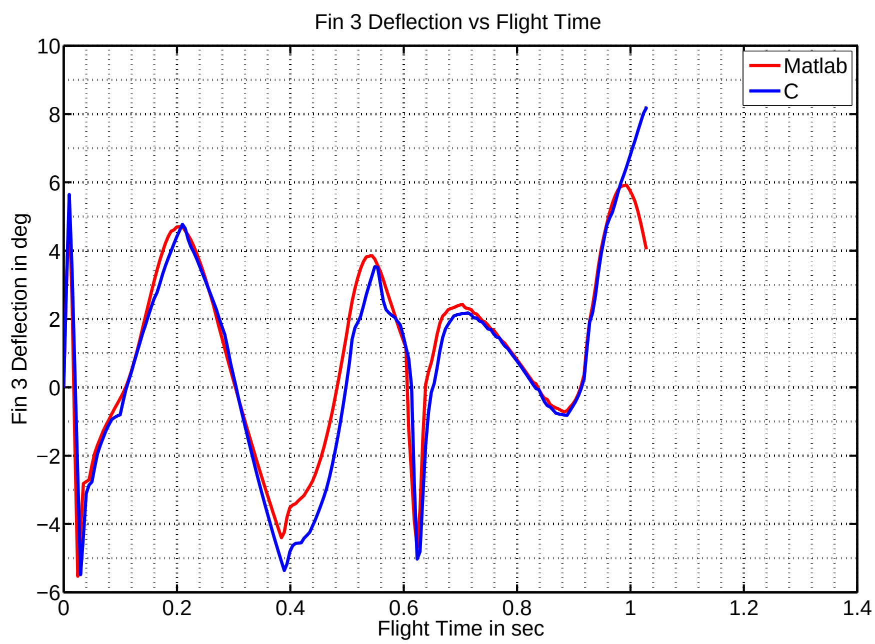

Figure 10.9: Fin 3 Profile - MATLAB & C Simulations

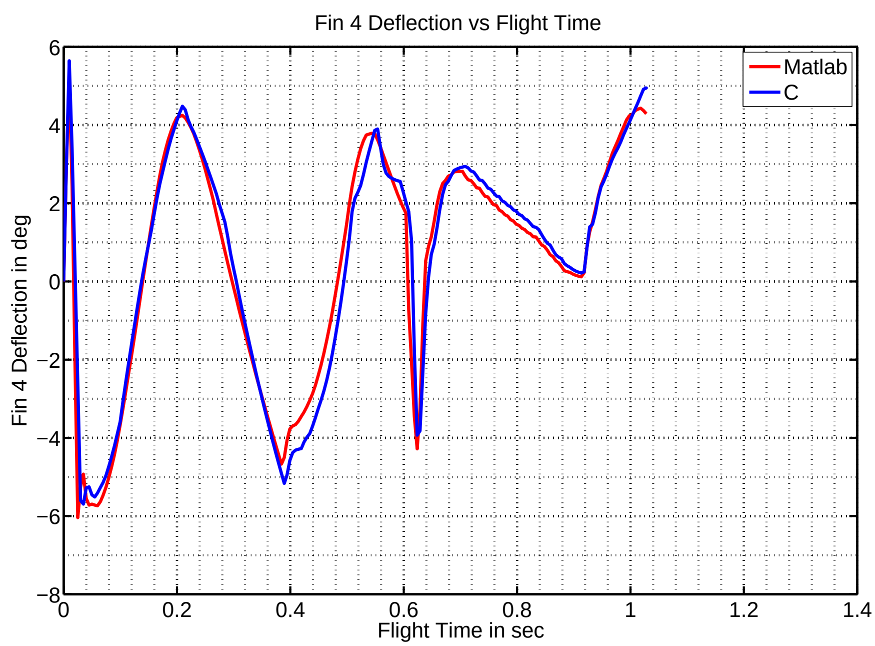

Figure 10.10: Fin 4 Profile - MATLAB & C Simulations

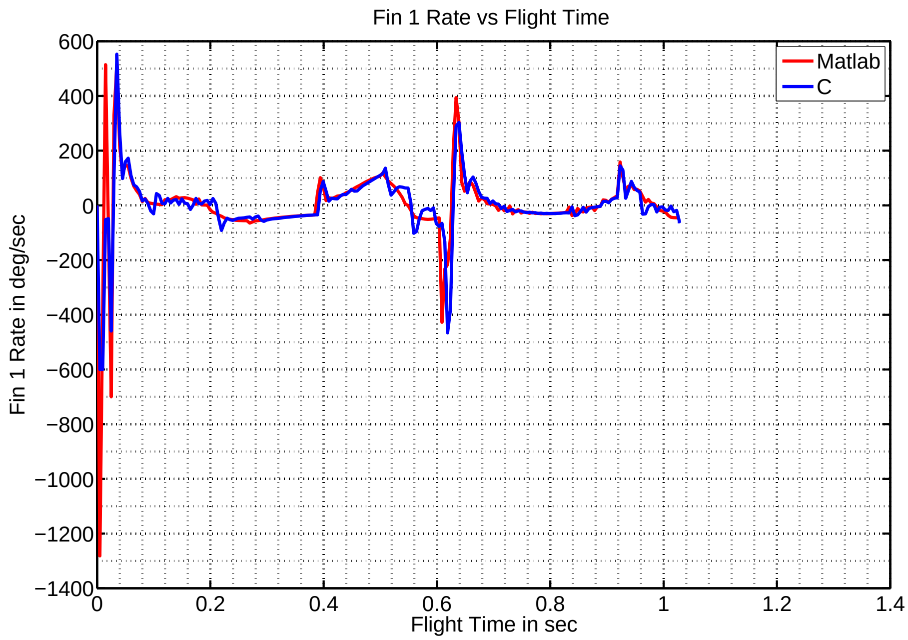

Figure 10.11: Fin 1 Rate Profile - MATLAB & C Simulations

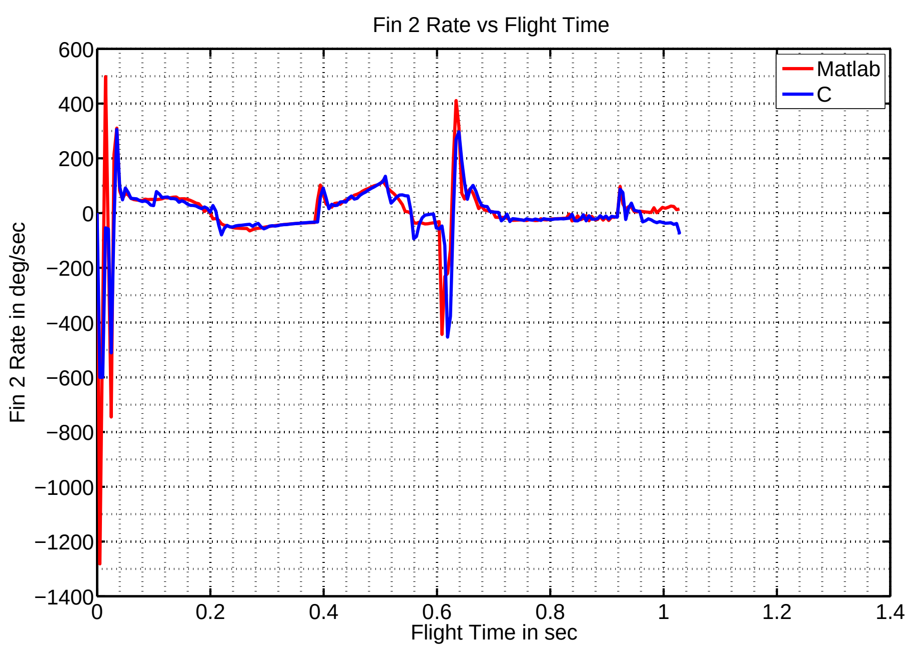

Figure 10.12: Fin 2 Rate Profile - MATLAB & C Simulations

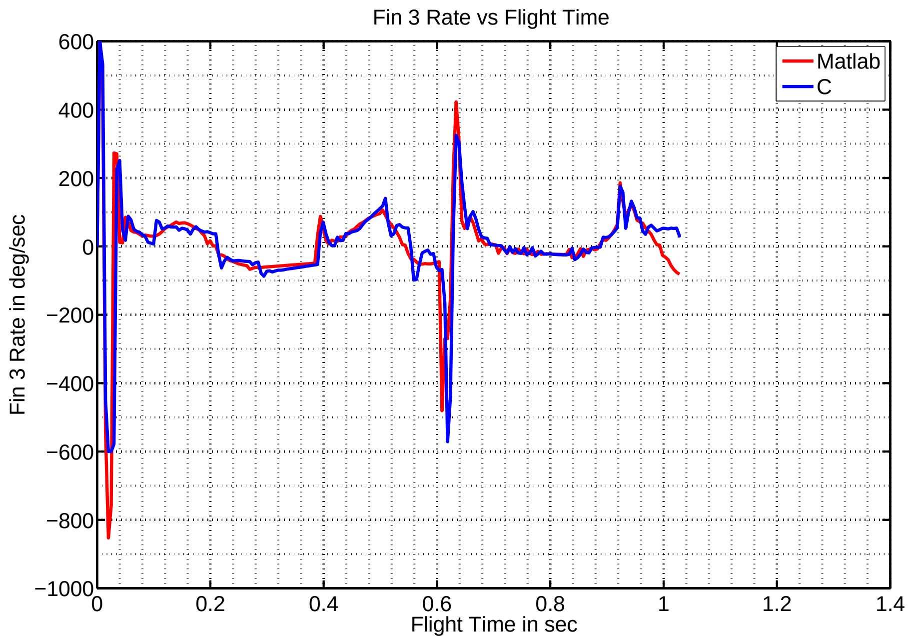

Figure 10.13: Fin 3 Rate Profile - MATLAB & C Simulations

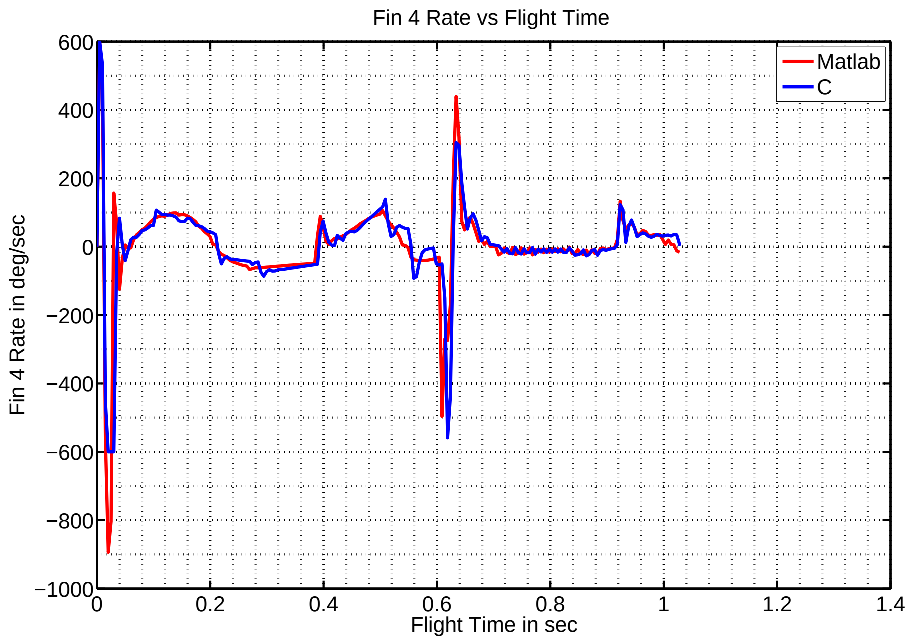

Figure 10.14: Fin 4 Rate Profile - MATLAB & C Simulations

10.5 Summary and Conclusions

In this chapter, developement of 3D animation using MATLAB VRML toolbox is explained in detail. Also development of interactive GUI for entering the initial flight conditions is explained. Visualization of missile-target engagement using MATLAB will enable us to explore future research, behaviour of both missile and target can be studied thoroughly. Finally the MATLAB simulation results are compared with C program [5] results and accuracy of MATLAB simulation is ascertained.

Chapter 11

SUMMARY & DIRECTIONS FOR FUTURE RESEARCH

11.1 Summary of Work

This thesis addressed about the analysis, and control issues that are critical about the BTT missiles. The following summarizes key themes within the thesis.

1. **Literature Survey.** A fairly comprehensive literature survey of relevant work was presented.

2. **Modeling.** A nonlinear dynamical model for the BTT missile was presented and linearization analysis was performed to understand the full utility of each model.

3. **Control.** Both inner-loop and outer-loop control designs were discussed in the context of an overall hierarchical control inner-outer loop framework. This framework lends itself to accommodate multiple phase of missile flight; The need for an nonlinear gain scheduled autopilot was explored and a sample nonlinear autopilot was obtained using incremental nonlinear dynamic inversion technique for the innermost rate control loop design. Comprehensive inner-loop trade studies were conducted for the BTT missile. A great deal of effort was spent on discussion fundamental performance limitations. Attention was spent on numerical integration step size limitations as well as dynamic (bandwidth) limitations.

4. **Miss Distance Analysis.** Set of missile-target engagement simulations were carried out by varying various missile flight conditions and their effects on final miss distance was analyzed and tabulated.

5. **Kill Zone Analysis.** Using Binary Search algorithm, a closed area in 2D space where the probability of missile hitting the target being high was estimated. The estimated result is analyzed for various flight parameter variations and tabulated.

6. **Animation Demonstrations.** Many animation demonstrations were conducted - with animation corroborating the simulation data results.

11.2 Directions for Future Research

Complicated research topic like missile control always presents great deal of future topics to explore. Remember uncertainity modeling in plant dynamics is not addressed here. Things get interesting when we try to include uncertainity model in our control design and we would like to see how they affect our robustness properties at different loop breaking points. Thus looking forward from the research conducted here, following points will throw some light on future topics to explore.

1. Integrated Guidance Navigation & Control (GNC) design where guidance loop is designed as a part of autopilot and the new design can be studied for its robustness.

2. Studying missile-target engagement with a 6DOF target and learn when would we need such a complicated target over a simple 3DOF target which is used in this research. Remember when target is 6DOF, it will have its own autopilot.

3. Trying out different target intelligence algorithms and learn how an increase in target intelligence would affect the missile's tracking ability.

4. Extending current 2D kill zone search to 3D search space, where both missile and target can start at any altitude. Also analyzing the same 3D kill zone with respect to different missile-target engagement parameters and comparitive studies can be done with 2D kill zone results presented in this thesis.

5. Optimal Control missile guidance law suffers from poor Time-To-Go estimate problem. This can be addressed using an Extended Kalman Filter (EKF) algorithm.

6. Extending one-on-one missile target engagement to multiple missile-target engagement. Multiple missiles can be made to chose their target dynamically on the run-time based on some state of emergency (need of the hour) or lethal nature of target. This is a very interesting resource allocation problem and interessting solutions can be achieved using game theory techniques developed for pursuit evasion problems.

REFERENCES

[1] Sonne M., Rodriguez A. A., "A Viable PC Tool for the Design and Evaluation of Missile Guidance and Control Systems," *Proceedings of the American Control Conference*, Baltimore, MD, June 29-July 1, 1994, pp. 2726-2730.

[2] Wallis M., Feeley J., "BTT Missile-Target Simulations on a Desktop Computer," *The Soc. for Comp. Sim. Int.*, 1989, pp. 79-84.

[3] Riggs T., Vergaz B., "Adv. Air-to-Air Missile Gui. Using Opt. Control and Est," *AFATL-TR-81-56 AFAL, Eglin AFB*, FL.

[4] Sheldon L. C., "6DOF BTT Missile Simulations of the ILAAT Missile with Lookup tables for the Aerodynamic Coefficients," *Air force Armament Laboratory*, Eglin AFB, FL, 1986.

[5] Sonne M. L., *Use of PC's in Evaluation and Design of Missile Guidance and Control Systems*, Arizona State University, MS Thesis, 1994.

[6] Jenniges A., *Simulink Based Evaluation and Design of Missile Guidance and Control Systems*, Arizona State University, Independent Study Report, 1998.

[7] *Çimen* T., "A Generic Approach to Missile Autopilot Design using State-Dependent Nonlinear Control," *Preprints of the* 18^{th} *IFAC World Congress*, Milano, Italy, August 28 - September 2, 2011, pp. 9587 - 9600.

[8] Siddarth A. N., Peter F., Holzapfel F., Valasek J., "Autopilot for a Nonlinear Non-Minimum Phase Tail-Controlled Missile," *AIAA Guidance, Navigation, and Control Conference*, 13-17 January 2014, National Harbor, Maryland.

[9] Wang G., Mei W., Liu H., Xiao Y., "The Design of Virtual Reality Flight Simulation Platform Based on Rigid Body Kinematics," *International Journal of Digital Content Technology and its Applications(JDCTA)*, Vol. 7, No. 7, April 2013.

[10] Cloutier J. R., Stansbery D. T., "Nonlinear, Hybrid Bank-to-Turn/Skid-to-Turn Missile Autopilot Design," *Proceedings of AIAA Guidance, Navigation & Control Conference*, Montreal, Canada, August 2001.

[11] Cloutier J. R., "State-Dependent Riccati Equation Techniques: An Overview," *Proceeding of American Control Conference*, Albuquerque, New Mexico, June 1997.

[12] Hawley P. A., Blauwkamp R. A., "Six-Degree-of-Freedom Digital Simulations for Missile Guidance, Navigation and Control," *John Hopkins APL Technical Digest*, Vol. 29, No. 1, 2010.

[13] Jackson P. B., "Overview of Missile Flight Control Systems," *John Hopkins APL Technical Digest*, Vol. 29, No. 1, 2010.

[14] Kaplan J. A., Chappell A. R., Mcmanus J. W., "The Analysis of a Generic Air-to-Air Missile Simulation Model," *NASA Technical Memorandum 109057*, June 1994.

[15] Rodriguez A. A., Sonne M., "The PC as a Viable Tool in the Design and Evaluation Process of Guidance and Control Systems for Missiles," *Proceedings of the 1994 International Conference on Simulation in Engineering Education*, Tempe, AZ, January 24-26 1994, pp. 329-334.

[16] Rodriguez A. A., Aguilar R., "Graphical Visualization of Missile-Target Air-to-Air Engagements: An Educational Tool for Designing and Evaluating Missile Guidance and Control Systems," *Journal of Computer Applications in Engineering Education*, Vol. 3, No. 1, 1995, pp. 5-20.

[17] Rodriguez A. A., Sonne M. L., "Evaluation of Missile Guidance and Control Systems Simulation," *The Journal of the Society for Computer Simulation*, Vol. 68, No. 6, June, 1997, pp. 363-376.

[18] Cloutier J. R., Evers J. H., Feeley J. J., "Assesment of Air-to-Air Missile Guidance and Control Technology," *IEEE Control Systems Magazine*, October 1989, pp. 27-34.

[19] Silbert M., Sarkani S., Mazzuchi T., "Comparing the State Estimates of a Kalman filter to a Perfect IMM Against a Maneuvering Target," 14^{th} *International Conference on Information Fusion*, Chicago, Illinois, July 5-8, 2011, pp. 1144-1148.

[20] Cronvich L. L., "Aerodynamic Considerations for Autopilot Design," *Progress in Astronautics and Aeronautics Tactical Missile Aerodynamics*, Vol. 141, 1992, p. 37.

[21] Williams D. E., Friedland B., "Modern Control Theory for Design of Autopilots for Bank-to-Turn Missiles," *Journal Guidance Control*, Vol. 10, No. 4, July-August 1987, pp. 378-386.

[22] Das P., Khilar P. M., "A Randomized Searching Algorithm and its Performance analysis with Binary Search and Linear Search Algorithms," *The International Journal of Computer Science and Applications (TIJCSA)*, Volume 1, No. 11, Jan 2013, ISSN - 2278-1080.

[23] Hui Y., Nan Y., Chen S., Ding Q., Wu S., "Dynamic Attack Zone of Air-to-Air missile after being launched in random wind field," *Chinese Journal of Aeronautics*, 2015, pp. 1519 - 1528.

[24] Shinar J., Gazit R., "Optimal 'No Escape' Firing Envelopes of Guided Missiles," *AIAA Guidance Navigation & Control Conference*, AIAA-paper No. 85-1960, Snowmass, Colorado, August 1985.

[25] Smith R. L., "An Autopilot Design Methodology for Bank-to-Turn Missiles," *Interim Report AFATL-TR-89-49* March 1988 - April 1989 Eglin AFB, FL.

[26] Kramer F. S., "Multivariable Autopilot Design and Implementation for Tactical Missiles," *American Institute of Aeronautics and Astronautics*, 1998.

[27] Venkatesan R. H., Sinha N. K., "A New Guidance Law for the Defense Missile of Nonmaneuverable Aircraft," *IEEE Transactions on Control Systems Technology*, Vol. 23, No. 6, November 2015.

[28] Cloutier J. R., Shamma J. S., "Gain-Scheduled Missile Autopilot Design Using Linear Parameter Varying Transformations," *Journal of Guidance Control and Dynamics*, Vol. 16, No. 2, March-April 1993.

[29] Bossi J. A., Langehough M. A., "Multivariable Autopilot Designs for a Bank-to-Turn Missile," *Boeing Aerospace*, Seattle.

[30] Birkmire B., "Weapon Engagement Zone Maximum Launch Range Approximation Using a Multilayer Perceptron," *MS Thesis at B.S. Wright State University*, 2008.

[31] Chen D. Z., "Efficient Parallel Binary Search on Sorted Arrays," *(1990) Computer Science Technical Reports*, Paper 11.

[32] Corzo C., DeHerrera M. F., Rodriguez A. A., "Missile Target Engagements: A Tool for Evaluating Missile Autopilots," *Presented at the 1996 MAES National Symposium and Career Fair*, Lake Buena Vista, FL, January 10-13, 1996.

[33] Aguilar R., Rodriguez A. A., "Graphical Visualization of Missile-Target Air-to-Air Engagements: A Tool for Designing and Evaluating Missile Guidance and Control Systems," *Presented at the 3rd NSF National Conference on Diversity in the Scientific and Technological Workforce*, October 1994.

[34] Rodriguez A. A., Wang Y., "Saturation Prevention Strategies for an Unstable Bank-to-Turn (BTT) Missile: Full Information," *13th IFAC Symposium: Automatic Control in Aerospace Engineeering - Aerospace Control 1994*, Editors: Schaechter D. B., Lorell K. R., Palo Alto, CA, September 12-16, 1994, pp. 149-154.

[35] Wu F., Packard A., Balas G., "LPV Control Design for Pitch- Axis Missile Autopilots," *Proceedings of the 34th Conference on Decision and Control*, New Orleans, LA, December, 1995.

[36] Balas G. J., Packard A., "Design of Robust, Time-Varying Controllers for Missile Autopilots," *Proc. 1st IEEE Conf. Control Applications*, 1992.

[37] Abell D. C., Caraway W. D., "A Method for Determination of Target Aspect Angle with Respect to an Unleveled Radar," *Army aviation and missile command redstone arsenal al missile research development and engineering center*, July 1998.

[38] Rodriguez A. A., Cloutier J. R., "Performance Enhancement for a Missile in the Presence of Saturating Actuators," *AIAA Journal of Guidance, Control and Dynamics*, January-February, 1996, Vol 19, pp. 38-46.

[39] Rodriguez A. A., Wang Y., "Performance Enhancement for Unstable Bank-to-Turn (BTT) Missiles with Saturating Actuators," *International Journal of Control*, Vol. 63, No. 4, 1996, pp. 641-678.

[40] Barker J. M., Balas G. J., "Comparing Linear Parameter-Varying Gain-Scheduled Control Techniques for Active Flutter Suppression," *Journal of Guidance, Control and Dynamics*, 23(5):948-955, September-October 2000.

[41] Rodriguez A. A., Wang Y., "Saturation Prevention Strategies for Unstable Bank-to-Turn (BTT) Missiles: Partial Information," *Proceedings of the American Control Conference*, Seattle, WA, June 21-23, 1995, pp. 2143-2147.

[42] Rodriguez A. A., Spillman M., Ridgely D. B., "Control of a Missile in the Presence of Saturating Actuators," *Presented and Distributed at the American Control Conference*, Seattle, WA, June 21-23, 1995.

[43] Puttannaiah K, Echols J, and Rodriguez A, "A Generalized $\mathcal{H}^\infty$ Control Design Framework for Stable Multivariable Plants subject to Simultaneous Output and Input Loop Breaking Specifications," *American Control Conf. (ACC), IEEE*, 2015.

[44] Puttannaiah K, Echols J, Mondal K and Rodriguez A, "Analysis and Use of Several Generalized H-Infinity Mixed Sensitivity Framework for Stable Multivariable Plants Subject to Simultaneous Output and Input Loop Breaking Specifications," *Accepted for publication in Conf. on Decision and Control (CDC), IEEE*, 2015.

[45] Rodriguez A. A., "A Methodology for Missile Autopilot Performance Enhancement in the Presence of Multiple Hard Nonlinearities," *AFOSR SFRP Report*, WL/MNAG, Eglin Air Force Base, FL, Sept, 1993.

[46] Rodriguez A. A., "Development of Control Design Methodologies for Flexible (High Order) Missile Systems with Multiple Hard Nonlinearities," *AFOSR SFRP Report*, WL/MNAG, Eglin Air Force Base, FL, September, 1992, pp. 47-1-47-20.

[47] Johnson and Reiss, "Numerical Analysis," Chapters 7.2.4 - 7.2.5.

[48] Arrow A., "An Analysis of Aerodynamic Requirements of Coordinated Bank-to-Turn Missiles," *NASA CR 3544*, 1982.

[49] Chin S., "Missile Configuration Design," *McGraw Hill*, 1961, p17.

[50] Cloutier J. et.al, "Assessment of Air-to-Air Missile Guidance and Control Technology," *IEEE C. Sys.*, Oct 1989, pp. 27-34.

[51] Siouris G. M., "Missile Guidance and Control Systems," *2004 Springer-Verlag New York, Inc.*

[52] Zarchan P., "Tactical and Strategic Missile Guidance," *AIAA Inc.*, 1990.

[53] Babister A. W., "Aircraft Stability and Control," *Oxford: Pergamon Press*, 1961.

[54] Shastry S., Bodson M., "Adaptive Control: Stability, Convergence, and Robustness," *Prentice Hall*, 1989.

[55] Etkin B., "Dynamics of Flight Stability and Control," *John Wiley and Sons, Inc.*, 1982, pp. 3.

[56] Reidel F. W., "Bank-To-Turn Control Technology Survey for Homing Missiles," *NASA CR 3325*, 1980.

[57] McCormick B., "Aerodynamics, Aeronautics and Flight Mechanics," *John Wiley and Sons, Inc.*, 1979.

[58] Robert N., "Flight Stability and Automatic Control," *McGraw Hill*, 1989, p. 6.

[59] McRuer, Duane et al., "Aircraft Dynamics and Automatic Control," *Princeton University Press*, Princeton, New Jersey, 1973, p. 208.

[60] Lin, Ching-Fang, "Modern Navigation Guidance and Control Processing," *Prentice Hall*, 1991, pp. 14, 184.

[61] Rodriguez A. A., "Missile Guidance," *Wiley Encyclopedia of Electrical and Electronics Engineering*, 15 June 2015.

[62] Blakelock J. H., "Automatic Control of Aircraft and Missiles," *John Wiley and Sons, Inc.*, Second Edition.

[63] Lofberg J., "Yalmip: A toolbox for modeling and optimization in matlab in Computer Aided Control Systems Design," *2004 IEEE International Symposium*, pp. 284-289.

[64] Rodriguez A. A., *Analysis and Design of Feedback Control Systems*, Control3D,L.L.C., Tempe, AZ, 2002.

[65] Rodriguez A. A., *Analysis and Design of Multivariable Feedback Control Systems*, Control3D, L.L.C., Tempe, AZ, 2002.

[66] Rodriguez A. A., *Linear Systems: Analysis and Design*, Control3D,L.L.C., Tempe, AZ, 2002.

[67] Rodriguez A. A., EEE481: Computer Control Systems, course notes, 2014.

[68] Olver P., "Notes on Nonlinear Ordinary Differential Equations," *https://www.math.umn.edu/~olver/am_/odz.pdf*

[69] Hespanha J. P., "Topics in Undergraduate Control Systems Design".

[70] "Conventional Missiles Warheads and their Blast Radii," *http://kitsune.addr.com/Rifts/Rifts-Missiles/convent.htm*

[71] "Wikipedia Link for Studying Binary Search Algorithm," *https://en.wikipedia.org/wiki/Binary_search_algorithm*

[72] "C Program Code for Binary Search Algorithm," *http://www.programmingsimplified.com/c/source-code/c-program-binary-search*

[73] "Simulink 3D Animation - User's Guide," *http://www.mathworks.com/help/sl3d/index.html?s_cid=doc_ftr*

[74] "Control of Linear Parameter Varying Systems - Wikipedia," *https://en.wikipedia.org/wiki/Linear_parameter-varying_control*

[75] Gerard Leng, "MDTS Guidance, Aerodynamics & Control Course Website," *http://dynlab.mpe.nus.edu.sg/mpelsb/mdts/index.html*

[76] Warnick S. C., Rodriguez A. A., "A Systematic Anti-windup Strategy and the Longitudinal Control of a Platoon of Vehicles with Control Saturations," *IEEE Transactions on Vehicular Technology*, Vol. 49, No. 3, May 2000, pp. 1006-101

[77] Hedrick J. K., Girard A., *Control of Nonlinear Dynamic Systems: Theory and Applications*, 2005.

[78] Stein G., "Respect the Unstable," *IEEE Control Systems Magazine*, 2003.

[79] Skogestad S., Postlethwaite I., *Multivariable Feedback Control: Analysis and Design*, Wiley, 1996.

[80] Morari M., Zafiriou E., *Robust Process Control*, Prentice Hall.

APPENDIX A

C CODE - BINARY SEARCH ALGORITHM

```
//
// VENKATRAMAN RENGANATHAN
// ASU ID: 1206395992
// MS EE Fall 2013 - Summer 2016
// Ph. No - 4806289124
// Thesis on Missile Guidance Control System
//
//BELOW C CODE CAN BE MODIFIED FOR MISS DISTANCE ANALYSIS TOO.
//BINARY SEARCH KILL ZONE

void main()
{
int i = 0, up_ray_finish, hit_reach, miss_reach, hit_counter = 0;
int previous_ray_hit_count = 0,half_search_complete = 0;
int restart_on = 0,miss_threshold = 0, NAN_check = 0;
float initial_range = 0, miss_range = 0,  final_180_hit_range = 0;
float previous_final_hit_range = 0, hit_range = 0, hit_array[100];
float target_hit_y_positions[100], target_hit_x_positions[100];

altitude_array[0] = -1000;
altitude_array[1] = -2000;
altitude_array[2] = -5000;
altitude_array[3] = -8000;
altitude_array[4] = -10000;

max_accel_array[0] = 15;
max_accel_array[1] = 30;
max_accel_array[2] = 45;
max_accel_array[3] = 60;
max_accel_array[4] = 80;

/*mach_array[0] = 0.8999;
mach_array[1] = 1.0;
mach_array[2] = 1.1;
mach_array[3] = 1.2;
mach_array[4] = 1.3;*/

mach_array[0] = 1.4;
mach_array[1] = 1.5;
mach_array[2] = 1.6;
mach_array[3] = 1.7;
mach_array[4] = 1.8;

flight_condition_count = 5;
for (i = 0; i<100; i++)
{
// Completely clear the arrays and make them ready for new ray
hit_array[i] = 0;
target_hit_x_positions[i] = 0;
target_hit_y_positions[i] = 0;
}

for(mach_id = 0; mach_id <5; mach_id++)
{
alti_id = 0; //
max_acc_id = 4; // Max accel = 80g
flight_condition_count = flight_condition_count + 1;
Intial_Conditions_Counter = 0; // Reset for every flight condition
ray_angle = 180;
half_search_complete = 0; // reset the flag for next iteration.
// Kill ZONE for 1 Flight Condition
while(ray_angle < 360 && ray_angle > 0)
{
if (hit_counter != 0)
{
OpenOut();
SaveData(hit_array, target_hit_x_positions, target_hit_y_positions);
for (i = 0; i<100; i++)
{
// Completely clear the arrays and make them ready for new ray
hit_array[i] = 0;
target_hit_x_positions[i] = 0;
target_hit_y_positions[i] = 0;
}
Fileclose();
}
// ray search to the far end
// store last ray hit counts for stopping the search.
previous_ray_hit_count = hit_counter;
initial_range = 100;
Range = initial_range;
hit_range = 0;
miss_range = 0;
up_ray_finish = 0;
miss_threshold = 0;
miss_reach = 0;
```

```
hit_counter = 0;
hit_reach = 0;
restart_on = 0;

// search along 1 ray
while(up_ray_finish == 0)
{
slope = tan((180-ray_angle)*Deg2Rad);
for(i=0;i<36;i++)
{
X[i] = 0;
Xdot[i] = 0;
}
Launch();
Flight(X,Xdot);    //fly missile,with initialized states
NAN_check = ((Range != Range) || (Smx != Smx));
if(restart_on == 0) // normal search is happening
{
if((Range <= 20 && Range >= 0) && miss_reach != 1)
{
// hit condition before 1st miss along ray
hit_reach = 1;
miss_threshold = 0;
hit_range = initial_range;
hit_array[hit_counter] = hit_range;
target_hit_x_positions[hit_counter] = target_initial_x;
target_hit_y_positions[hit_counter] = target_initial_y;
hit_counter = hit_counter + 1;
initial_range = 2 * hit_range;
Range = initial_range;
}
else if ((Range <= 20 && Range >= 0) && miss_reach == 1)
{
// hit condition after 1st miss along ray
hit_reach = 1;
miss_threshold = 0;
hit_range = initial_range;
hit_array[hit_counter] = hit_range;
target_hit_x_positions[hit_counter] = target_initial_x;
target_hit_y_positions[hit_counter] = target_initial_y;
hit_counter = hit_counter + 1;
initial_range = (hit_range + miss_range)/2;
Range = initial_range;
}
else if (((fabs(Range) > 20) || (NAN_check == 1)) && (hit_reach != 1))
{
// miss condition before 1st hit
miss_reach = 1;
miss_threshold = miss_threshold + 1;
miss_range = initial_range;
initial_range = 2* miss_range;
Range = initial_range;
}
else if (((fabs(Range) > 20) || (NAN_check == 1)) && (hit_reach == 1))
{
// miss condition after 1st hit
miss_reach = 1;
miss_threshold = miss_threshold + 1;
miss_range = initial_range;
initial_range = (hit_range + miss_range) / 2;
Range = initial_range;
}
}
else // restart is happening
{
if((Range <= 20 && Range >= 0) && miss_reach != 1)
{
// hit condition while querying range using
// previous_ray_final_hit_range
hit_reach = 1;
miss_threshold = 0;
hit_range = initial_range;
hit_array[hit_counter] = hit_range;
target_hit_x_positions[hit_counter] = target_initial_x;
target_hit_y_positions[hit_counter] = target_initial_y;
hit_counter = hit_counter + 1;
initial_range = 2 * hit_range;
Range = initial_range;
}
else if ((Range <= 20 && Range >= 0) && miss_reach == 1)
{
// hit condition after 1st miss along ray
hit_reach = 1;
miss_threshold = 0;
hit_range = initial_range;
hit_array[hit_counter] = hit_range;
target_hit_x_positions[hit_counter] = target_initial_x;
target_hit_y_positions[hit_counter] = target_initial_y;
```

```
hit_counter = hit_counter + 1;
initial_range = (hit_range + miss_range)/2;
Range = initial_range;
}
else if ((fabs(Range) > 20) || (NAN_check == 1))
{
// miss condition while querying range using
// previous_ray_final_hit_range
miss_reach = 1;
miss_threshold = miss_threshold + 1;
miss_range = initial_range;
initial_range = (hit_range + miss_range) / 2;
Range = initial_range;
}
} // restart module completed
if ((miss_threshold >= 10) && (hit_reach == 0))
{
// FINAL TERMINATION CRITERION
up_ray_finish = 1; // ray search over
}
if((fabs(hit_range - miss_range) < 100) && (up_ray_finish == 0))
{// |hit-miss|<100 || range>20
if (miss_range > hit_range)
{
// FINAL TERMINATION CRITERION
up_ray_finish = 1; // ray search over
}
else
{
// Initial Hit_Range found.
// Restart the algorithm to find the final hit_range
// FORCE RESTART
if (ray_angle == 180)
{
initial_range = 2 * hit_array[0];
}
else
{
// search current ray's final hit position with
//idea from previous ray's final hit position
initial_range = previous_final_hit_range;
}
hit_range = hit_array[0];
hit_reach = 0; // reset hit_reach flag
miss_reach = 0; // reset miss_reach flag
up_ray_finish = 0; // ray search not over
Range = initial_range;
restart_on = 1;
}
}
// CHECK FOR TERMINATION CRITERION FOR BOTTOM AND TOP SEARCH
if(up_ray_finish == 1)
{
if (hit_counter == 0 && previous_ray_hit_count != 0
&& half_search_complete == 0)
{
// FINAL TERMINATION CRITERION for BOTTOM SEARCH
// hit_counter == 0 ---> current ray is a complete missing ray
// previous_ray_hit_count != 0 ---> previous ray had atleast 1 hit
// half_search_complete == 0 --> bottom search is happenning
// Previous ray had atleast 1 hit and current ray has no hits.
// Stop searching along ray which continuously gives a miss
// force it to start searching from 178 deg in the top direction
// set the bottom search complete flag to 1
up_ray_finish = 1;
ray_angle = 180;
previous_final_hit_range = final_180_hit_range;
half_search_complete = 1;
}
else if (hit_counter == 0 && previous_ray_hit_count != 0
&& half_search_complete == 1)
{
// FINAL TERMINATION CRITERION for TOP SEARCH
// hit_counter == 0 ---> current ray is a complete missing ray
// previous_ray_hit_count != 0 ---> previous ray had atleast 1 hit
// half_search_complete == 1 --> top search is happenning
// Previous ray had atleast 1 hit and current ray has no hits.
up_ray_finish = 1;
// Stop searching along ray which continuously gives a miss
ray_angle = 500;
// Stop Kill Zone Search - big number to get out of both the loops
}
}
} // ray search gets over here
```

```
// DECIDING HOW TO PROCEED TO NEXT RAY
if (half_search_complete == 0)
{
// increment bottom search ray angle by 10 degree
if(ray_angle == 180)
{
final_180_hit_range = hit_array[hit_counter - 1];
previous_final_hit_range = final_180_hit_range;
}
else
{
previous_final_hit_range = hit_array[hit_counter - 1];
}
ray_angle = ray_angle + 5;
}
else // half_search_complete == 1
{
// decrement top search ray angle by 10 degree
if(ray_angle != 180)
{
previous_final_hit_range = hit_array[hit_counter - 1];
}
ray_angle = ray_angle - 5;
}
}
} // end of FOR LOOP
return;    /* ...and return */
}
```

```
%% DATA_PREPARE_SIMPLE.M
%% PREPARE KILL ZONE DAT FILES FOR PLOTTING
range_filename = 'out_range.dat';
stx_file_name = 'out_stx.dat';
sty_file_name = 'out_sty.dat';
file_name_1 = 'Simulation_Results/Flight_Cdtn_';
for i = 6:10
%     i = 1;
    flight_number_path = strcat(file_name_1, num2str(i));
    cd(flight_number_path);
    filestruct = dir;
    numdirectories(i) = sum([filestruct.isdir]) - 2;
    cd ..
    cd ..
end

for k = 6:10
% for each and every ray - each ray is an initial condition
for i = 1:numdirectories(k)
sim_number = num2str(i);
flt_cdtn_number = num2str(k);
flight_number_path = strcat(file_name_1, flt_cdtn_number);
file_name_2 = '/Simulated_IC_';
file_name_3 = '_Results';
folder_name = strcat(flight_number_path, file_name_2, ...
sim_number, file_name_3);
cd(folder_name);

fileID = fopen(range_filename, 'r+b');
temp_hit_range_array = fread(fileID, 50000, '*float');
fclose(fileID);

fileID = fopen(stx_file_name, 'r+b');
temp_hit_stx_array = fread(fileID, 50000, '*float');
fclose(fileID);

fileID = fopen(sty_file_name, 'r+b');
temp_hit_sty_array = fread(fileID, 50000, '*float');
fclose(fileID);
cd ..
cd ..
cd ..

% Prepare exact array from big array which has lot of zeros
for j = 1:length(temp_hit_stx_array)
    if(temp_hit_range_array(j) > 0)
        hit_range_array(j) = temp_hit_range_array(j);
        hit_stx_array(j) = temp_hit_stx_array(j);
        hit_sty_array(j) = temp_hit_sty_array(j);
    end
end

[min_range, min_index] = min(hit_range_array);
[max_range, max_index] = max(hit_range_array);
initial_hit_x(i) = hit_stx_array(min_index);
```

```
initial_hit_y(i) = hit_sty_array(min_index);
final_hit_x(i) = hit_stx_array(max_index);
final_hit_y(i) = hit_sty_array(max_index);

clear temp_hit_range_array;
clear temp_hit_stx_array;
clear temp_hit_sty_array;
clear hit_range_array;
clear hit_stx_array;
clear hit_sty_array;

end

hit_x = [initial_hit_x'; final_hit_x'];
hit_y = [initial_hit_y'; final_hit_y'];
dat_file_name = strcat('kill_zone_', flt_cdtn_number,'_data.mat');
cd('Kill Zone Dat Files');
save(dat_file_name);
cd ..
end
```

```
%% PLOT_KILL_ZONE.M
clear all; clc;
for l = 9:-1:1
    name_1 = 'kill_zone_';
    name_2 = '_data.mat';
    data_num = num2str(l);
    file_name = strcat(name_1, data_num, name_2);
    load(file_name);
    A = double(hit_x);
    B = double(hit_y);
    k = boundary(A,B);
    switch(l)
        case 1
            color_vector = [0 .5 .1];
        case 2
            color_vector = [.5 .8 .1];
        case 3
            color_vector = [.8 .5 .1];
        case 4
            color_vector = [.9 .1 .4];
        case 5
            color_vector = [.5 .5 .8];
        case 6
            color_vector = [.5 0 .1];
        case 7
            color_vector = [0.1 0.9 0.2];
        case 8
            color_vector = [0.8 0.8 0.1];
        case 9
            color_vector = [0.1 0.2 0.3];
        case 10
            color_vector = [0.9 0.8 0.7];
    end
    patch(A(k),B(k), color_vector)
    hold on;
end

plot(0,0, 'r*', 'MarkerSize', 20)
hold on;


grid on;
title('Kill Zone as a Function of Initial Missile Mach', 'fontsize',24)
legend({'Mach = 1.7', 'Mach = 1.6', 'Mach = 1.5', 'Mach = 1.4',...
'Mach = 1.3', 'Mach = 1.2', 'Mach = 1.1', 'Mach = 1.0',...
'Mach = 0.8999','Missile Location'},'Location','Best');
set(gcf, 'PaperPositionMode', 'auto');
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 3);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 6);
ax = gca;
x_vector = 0:5:25;
y_vector = -25:5:25;
set(ax,'XTickLabel',{x_vector})
set(ax,'YTickLabel',{y_vector})
set(a, 'FontSize', 24);
hold off
xlabel('X (kft)', 'fontsize',24);
ylabel('Y (kft)', 'fontsize',24);
```

APPENDIX B

MATLAB CODE - MISSILE PLANT & AUTOPILOT ANALYSIS

```
%==========================================================================+
% M-file "btt_linr.m" SOLVES FOR THE NON-DIMENSIONAL STABILITY
% DERIVATIVES OF THE NON-DIMENSIONAL (i.e., SCALED) STATE-SPACE SYSTEM.
% THIS M-FILE ALSO FORMS THE A, B, C & D STATE-SPACE MATRICES OF
% LINEAR MODEL.
%
%   Written by:  Venkatraman Renganathan
%   -----------  (480)628-9124 (Mobile Number)          %
%==========================================================================+

%**********************************************************************
% Reference (trim values) Inputs to the Linerization Procedure:
%**********************************************************************
mach_array = [1.068 1.5114 2.0420];
thrust_array = [600 1400 2000];
mach_length = length(mach_array);

for jj=1:2

altit_ref = 30000.00; % Missile Geometric Altitude Reference Value [ft]
alpha_ref = 14;       % Missile Angle of Attack Reference Value    [deg]
beta_ref  = 0.0;      % Missile Side-slip Reference Value          [deg]
delP_ref  = 0.0;      % "Roll" Fin Deflection Reference Value      [deg]
delR_ref  = 0.0;      % "Yaw" Fin Deflection Reference Value       [deg]
P_ref     = 0.0;      % Roll Rate Reference Value                [rad/s]
Q_ref     = 0.0;      % Pitch Rate Reference Value               [rad/s]
R_ref     = 0.0;      % Yaw Rate Reference Value                 [rad/s]
Phi_ref   = 0.0;      % Bank Angle Reference Value                 [deg]
Theta_ref = 0.0;      % Attitude Angle                             [deg]
Psi_ref   = 0.0;      % Heading Angle                              [deg]
ThrustX   = thrust_array(jj);     % Sea Level 2nd Stage Thrust Force in
                      % the Body X-direction                       [lbf]

%**********************************************************************
% Actuator Dynamics (parameters):
%**********************************************************************
   KdelP    = 1.0;   % Effective "Roll" actuator closed-loop gain
   KdelR    = 1.0;   % Effective "Yaw" actuator closed-loop gain
   KdelQ    = 1.0;
   tau_delP = .005;   % Effective "Roll" Actuator time constant [sec]
   tau_delR = .005;   % Effective "Yaw" Actuator time constant [sec]
   tau_delQ = .005;   % Effective "Pitch" Actuator time constant [sec]

%**********************************************************************
%  Set Aerodynamic Coefficient Iteration Loop Absolute Error Criteria
%**********************************************************************
   err_crit  = 0.005;

%**********************************************************************
% Other Aerodynamic, Mass, and Inertia Parameters:
%**********************************************************************
   Lref = 0.625;   % Aerodynamic Reference Length [ft]
   Sref = 0.307;   % Aerodynamic Reference Area [ft^2]
   mass = 5.75;    % Missile Mass [slug]
   Ixx  = 0.34;    % Missile Body Frame X-Comp. of Inertia (Fuel Spent):
   Iyy  = 34.10;   % Missile Body Frame X-Comp. of Mass Moment [slug/ft^2]
   Izz  = 34.10;   % Missile Body Frame X-Comp. of Mass Moment [slug/ft^2]
   xcg  = 0.0;     %.525 Final Location of Center of Mass [ft]

%**********************************************************************
% Calculate Atmospheric Properties:
%**********************************************************************
 % [rho,SOS,Patm,Tatm,gravity,drho_dz, dSOS_dz] = atmos(abs(altit_ref));
[rho,SOS,gravity] = Compute_Altitude_Parameters(altit_ref);

%**********************************************************************
% Interate for Mach Number, delQ_ref, and Aerodynamic Coefficients:
%**********************************************************************

%=====================================================
% Correct Sea Level Thrust for Altitude (air density):
%=====================================================
  rho_sea = 0.0024;                 % Sea Level Air Density [slug/ft3]
  ThrustX = ThrustX*(rho/rho_sea);  % Corrected Propulsive Thrust [lbf]

%======================================================
% Load Aerodynamic Tables (execute m-file "aerodat.m"):
%======================================================
  aerodat

%==============================
% Guess Mach Number and delQ_ref
%==============================
   Mach_ref = mach_array(jj);
   delQ_ref = 1.0;  % [deg]
```

```matlab
    error     = 1.0;
    icount    = 0;

%========================
% Begin Iteration Loop
%========================
 while error > err_crit

   Vb     = SOS*Mach_ref;
   Vb_old = Vb;

% Use Absolute Values of Alpha_ref and Beta_ref for most Interpolations:
    absAlp   = abs(alpha_ref);
    absBet   = abs(beta_ref);

% Use "Pitch" Fin Deflection to Determine Sign of delQAlp:
    if delQ_ref >= 0.0
      delQAlp = abs(alpha_ref);
    else
      delQAlp = -1.0*abs(alpha_ref);
    end

% Use "Yaw" Fin Deflection to Determince Sign of delRBet:
    if delR_ref >= 0.0
      delRBet = abs(beta_ref);
    else
      delRBet = -1.0*abs(beta_ref);
    end

%=============================================================================
% Interpolate for Drag Coefficient CD=CD(alpha,delQ,M):
% CD is a three dimensional array in actuality; however, MATLAB only
% supports interpolation of 2-D Tables. Thus, we will carry out 2-D
% interpolation between a family of 2-D tables in the x and y directions
% (alpha and delQ, respectively) and then linearly interpolate between
% these two values for the final z-direction(Mach number):
%=============================================================================
    if Mach_ref <= 1.0
      Mach_lo = 0.9;
      Mach_hi = 1.0;
      CDlo = interp2(TdelQ',Talpha1,TCD1,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD2,delQ_ref,absAlp,'bilinear');

    elseif Mach_ref <= 1.1
      Mach_lo = 1.0;
      Mach_hi = 1.1;
      CDlo = interp2(TdelQ',Talpha1,TCD2,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD3,delQ_ref,absAlp,'bilinear');

    elseif Mach_ref <= 1.3
      Mach_lo = 1.1;
      Mach_hi = 1.3;
      CDlo = interp2(TdelQ',Talpha1,TCD3,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD4,delQ_ref,absAlp,'bilinear');

    elseif Mach_ref <= 1.5
      Mach_lo = 1.3;
      Mach_hi = 1.5;
      CDlo = interp2(TdelQ',Talpha1,TCD4,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD5,delQ_ref,absAlp,'bilinear');

    elseif Mach_ref <= 2.0
      Mach_lo = 1.5;
      Mach_hi = 2.0;
      CDlo = interp2(TdelQ',Talpha1,TCD5,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD6,delQ_ref,absAlp,'bilinear');

    elseif Mach_ref <= 2.5
      Mach_lo = 2.0;
      Mach_hi = 2.5;
      CDlo = interp2(TdelQ',Talpha1,TCD6,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD7,delQ_ref,absAlp,'bilinear');

    elseif Mach_ref <= 3.0
      Mach_lo = 2.5;
      Mach_hi = 3.0;
      CDlo = interp2(TdelQ',Talpha1,TCD7,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD8,delQ_ref,absAlp,'bilinear');

    elseif Mach_ref <= 4.0
      Mach_lo = 3.0;
      Mach_hi = 4.0;
      CDlo = interp2(TdelQ',Talpha1,TCD8,delQ_ref,absAlp,'bilinear');
      CDhi = interp2(TdelQ',Talpha1,TCD9,delQ_ref,absAlp,'bilinear');
    end

%  Interploate in the z-direction(Mach): Linearly interpolate between
%  the two interpolated tablular values CDlo and CDhi.
    vv = (Mach_ref - Mach_lo)/(Mach_hi - Mach_lo);
    CD = (1.0 - vv)*CDlo + vv*CDhi;
```

```
%==============================================================
% Interpolate for the Other Aerodynamic Coefficients:
% (NOTE: Interpolation functions require that reference variables
%  [e.g.,Mach_ref, alpha_ref, etc.] lie within the tabular values -
%  an error will occur if this is not the case)
%==============================================================

%================================
% CDT = CDT(M):
%================================
  CDT = interp1(Tmach2,TCDT,Mach_ref);

%================================
% CLbeta = CLbeta(alpha,M):
%================================
  CLbeta = interp2(Tmach1',Talpha1,TCLbeta,Mach_ref,absAlp,'bilinear');

%================================
% CLdelP = CLdelP(alpha,M):
%================================
  CLdelP = interp2(Tmach1',Talpha1,TCLdelP,Mach_ref,absAlp,'bilinear');

%================================
% CLP = CLP(alpha,M):
%================================
  CLP = interp2(Tmach1',Talpha3,TCLP,Mach_ref,absAlp,'bilinear');

%==================================
% CMalpha = CMalpha(alpha,M):
%==================================
  CMalpha = interp2(Tmach1',Talpha4,TCMalpha,Mach_ref,absAlp,'bilinear');

%===================================
% CMQ = CMQ(alpha,M):
%===================================
  CMQ = interp2(Tmach2',Talpha3,TCMQ,Mach_ref,absAlp,'bilinear');

%===================================
% CMdelQ = CMdelQ(alpha,M):
%===================================
  CMdelQ = interp2(Tmach1',Talpha2,TCMdelQ,Mach_ref,delQAlp,'bilinear');

%===================================
% CNalpha = CNalpha(M):
%===================================
  CNalpha = interp1(Tmach1,TCNalpha,Mach_ref);

%===================================
% CNbeta = CNbeta(alpha,M):
%===================================
  CNbeta = interp2(Tmach1',Talpha1,TCNbeta,Mach_ref,absAlp,'bilinear');

%===================================
% CNdelR = CNdelR(beta,M):
%===================================
  CNdelR = interp2(Tmach1',Tbeta1,TCNdelR,Mach_ref,delRBet,'bilinear');

%===================================
% CNdelQ = CNdelQ(alpha,M):
%===================================
  CNdelQ = interp2(Tmach1',Talpha2,TCNdelQ,Mach_ref,delQAlp,'bilinear');

%===================================
% CNR = CNR(beta,M):
%===================================
  CNR = interp2(Tmach2',Tbeta2,TCNR,Mach_ref,absBet,'bilinear');

%===================================
% CYbeta = CYbeta(alpha,M):
%===================================
  CYbeta = interp2(Tmach1',Talpha1,TCYbeta,Mach_ref,absAlp,'bilinear');

%===================================
% CYdelR = CYdelR(beta,M):
%===================================
  CYdelR = interp2(Tmach1',Tbeta1,TCYdelR,Mach_ref,delRBet,'bilinear');

%=================================================================
% Correct signs of CLbeta and CMalpha to agree with sign of alpha_ref:
%=================================================================
    if alpha_ref < 0.0
      CLbeta  = -1.0*CLbeta;
      CMalpha = -1.0*CMalpha;
    end
```

```

  Theta_rad = Theta_ref*pi/180.;
  Phi_rad   = Phi_ref*pi/180.;

  Vbtemp   = -2.*(mass*gravity*sin(Theta_rad) + ThrustX)/...
      (rho*Sref*(CD+CDT));
  Vb = sqrt(Vbtemp);
  Mach_ref = Vb/SOS;
  if Mach_ref > 4.0
    Mach_ref = 4.0;
  elseif Mach_ref < 0.9
    Mach_ref = 0.9;
  end

%==========================================================================
% Calculate Effective Elevator Deflection Trim Values:
%==========================================================================
  delQ_ref = -(CMalpha/CMdelQ)*alpha_ref;

  if delQ_ref > 20
    delQ_ref = 20;
  elseif delQ_ref < -20
    delQ_ref = -20;
  end

  Vb_new = Vb;
  error  = abs( ((Vb_new - Vb_old)/Vb_new) );

  if icount >= 20    % Exit "while" statement after 20 iterations
    error = 0;
  end
  icount = icount + 1;

 end

% Finally set the missile velocity with Mach value satisfying the trim.
 Vb = SOS*Mach_ref;
%*********************************************************************
%  End of Iteration Loop
%*********************************************************************

% Clear intermediate variables:
  clear Mach_lo;
  clear Mach_hi;
  clear CDlo;
  clear CDhi;
  clear vv;
  clear Theta_rad;
  clear Phi_rad;
  clear Vb_old;
  clear Vb_new;
  clear icount;
  clear error;
  clear err_crit;

% Clear Aerodynamic Tables to Free Memory:
  clear Talpha1;
  clear Talpha2;
  clear Talpha3;
  clear Talpha4;
  clear Tbeta1;
  clear Tbeta2;
  clear TdelQ;
  clear Tmach1;
  clear Tmach2;
  clear TCD1;
  clear TCD2;
  clear TCD3;
  clear TCD4;
  clear TCD5;
  clear TCD6;
  clear TCD7;
  clear TCD8;
  clear TCD9;
  clear TCDT;
  clear TCLbeta;
  clear TCLdelP;
  clear TCLP;
  clear TCMalpha;
  clear TCMdelQ;
  clear TCMQ;
  clear TCNalpha;
  clear TCNbeta;
  clear TCNdelR;
  clear TCNdelQ;
  clear TCNR;
  clear TCYbeta;
  clear TCYdelR;

%*********************************************************************
%  Calculate Missile's Velocity Magnitude and Dynamic Pressure:
%*********************************************************************
   Vb  = SOS*Mach_ref;
```

```
    Qdp = 0.5*rho*Vb*Vb;
    Qsl = Qdp*Sref*Lref;

%**********************************************************************
%  Calculate Trim Values of Aerodynamic Forces and Moment Coefficients:
%      CX = -(Fgx + ThrustX)/(Qdp*Sref)
%      CY = -( xcg*mass*N/Izz + Fgy/(Qdp*Sref) )
%      CZ = -( -xcg*mass*M/Iyy + Fgz/(Qdp*Sref) )
%      CL = 0
%      CM = -Mg/(Qdp*Sref*Lref)
%      CN = -Ng/(Qdp*Sref*Lref)
%**********************************************************************
    theta = Theta_ref/57.2958;
    phi   = Phi_ref/57.2958;
    CL = 0;
    CM = xcg*mass*gravity*cos(theta)*cos(phi)/Qsl;
    CN = -xcg*mass*gravity*cos(theta)*sin(phi)/Qsl;
    CX = (mass*gravity*sin(theta) - ThrustX)/(Qdp*Sref);
    CY = -(mass*gravity*cos(theta)*sin(phi)/(Qdp*Sref) + ...
           mass*xcg*Lref*CN/Izz + ...
           xcg*xcg*mass*mass*gravity*cos(theta)*cos(phi)/(Qdp*Sref*Izz) );
    CZ = -(mass*gravity*cos(theta)*cos(phi)/(Qdp*Sref) - ...
           mass*xcg*Lref*CM/Iyy + ...
           xcg*xcg*mass*mass*gravity*cos(theta)*sin(phi)/(Qdp*Sref*Iyy) );


%**********************************************************************
% Calculate Stability Derivatives:
% --------------------------------
%  The aerodynamic coefficients interpolated above are not all
%  dimensionless. Some have dimensions of [deg^-1]. They will be
%  made dimesionless below by the proper conversion of degrees
%  to radians (i.e., 57.2958 [deg/rad]).
%**********************************************************************
  tau_time = (mass*Vb/(Qdp*Sref));        % Time scale factor [sec]
  g_hat    = (mass*gravity)/(Qdp*Sref);   % Non-dimensional gravity
  alpha    = alpha_ref*pi/180.;           % put alpha_ref in radians
  deg2rad  = pi/180.;

%==============================================
% X-Component of Acceleration (Principal Axis):
%==============================================
  x_u    = 2*CX*cos(alpha);
  x_v    = 0.0;
  x_w    = 2*CX*sin(alpha);
  x_p    = 0.0;
  x_q    = -sin(alpha);
  x_r    = 0.0;
  x_phi  = 0.0;
  x_thet =-g_hat*cos(Theta_ref*deg2rad);
  x_delP = 0.0;
  x_delQ = 0.0;
  x_delR = 0.0;

%==============================================
% Y-Component of Acceleration (Pricipal Axis):
%==============================================
  y_u    = 2*CY*cos(alpha);
  y_v    = CNbeta*deg2rad;            % Most Stability derivatives are in
  y_w    = 2*CY*sin(alpha);           % units of [1/deg] and we need to
  y_p    = sin(alpha);                % convert them into radians.
  y_q    = 0.0;
  y_r    =-cos(alpha);
  y_phi  = g_hat*cos(Theta_ref*deg2rad);
  y_thet = 0.0;
  y_delP = 0.0;
  y_delQ = 0.0;
  y_delR = CYdelR*deg2rad;

%==============================================
% Z-Component of Acceleration (Principal Axis):
%==============================================
  z_u    = 2*CZ*cos(alpha) - CNalpha*sin(alpha)*deg2rad;
  z_v    = 0.0;
  z_w    = 2*CZ*sin(alpha) + CNalpha*cos(alpha)*deg2rad;
  z_p    = 0.0;
  z_q    = cos(alpha);
  z_r    = 0.0;
  z_phi  = 0.0;
  z_thet =-g_hat*sin(Theta_ref/57.2985);
  z_delP = 0.0;
  z_delQ = CNdelQ*deg2rad;
  z_delR = 0.0;

%======================================================
% X-Component of Angular Acceleration (Principal Axis):
%======================================================
```

```
 l_u     = 0.0;
 l_v     = (2*mass*mass*Lref/(rho*Sref*Ixx))*CLbeta*deg2rad;
 l_w     = 0.0;
 l_p     = (0.5*mass*Lref*Lref/Ixx)*CLP;
 l_q     = 0.0;
 l_r     = 0.0;
 l_phi   = 0.0;
 l_thet  = 0.0;
 l_delP  = (2*mass*mass*Lref/(rho*Sref*Ixx))*CLdelP*deg2rad;
 l_delQ  = 0.0;
 l_delR  = 0.0;

%==============================================================
% Y-Component of Angular Acceleration (Principal Axis):
%==============================================================
 m_u     =-(2*mass*mass*Lref/(rho*Sref*Iyy))*CMalpha*sin(alpha)*deg2rad;
 m_v     = 0.0;
 m_w     = (2*mass*mass*Lref/(rho*Sref*Iyy))*CMalpha*cos(alpha)*deg2rad;
 m_p     = 0.0;
 m_q     = (0.5*mass*Lref*Lref/Iyy)*CMQ;
 m_r     = 0.0;
 m_phi   = 0.0;
 m_thet  = 0.0;
 m_delP  = 0.0;
 m_delQ  = (2*mass*mass*Lref/(rho*Sref*Iyy))*CMdelQ*deg2rad;
 m_delR  = 0.0;

%==============================================================
% Z-Component of Angular Acceleration (Principal Axis):
%==============================================================
 n_u     = 0.0;
 n_v     = (2*mass*mass*Lref/(rho*Sref*Izz))*CNbeta*deg2rad;
 n_w     = 0.0;
 n_p     = 0.0;
 n_q     = 0.0;
 n_r     = (0.5*mass*Lref*Lref/Izz)*CNR;
 n_phi   = 0.0;
 n_thet  = 0.0;
 n_delP  = 0.0;
 n_delQ  = 0.0;
 n_delR  = (2*mass*mass*Lref/(rho*Sref*Izz))*CNdelR*deg2rad;

%==============================================================
% Constants needed for controller state space
%==============================================================
K_1 = 7;
K_2 = -10;
K_3 = 0.5;
K_4 = 500;
K_5 = -1.75;
K_6 = -1500;
K_7 = -5000;
K_10 = K_2*((1/Qdp)-(1/Vb));
K_11 = -K_2*Sref*CNalpha/mass;
a_zeq = 2573.12;
a_yeq = 0.5;
a_1 = K_4/Qdp;
a_2 = CLbeta/CLdelP;
a_3 = K_5 + K_6/Qdp;
a_4 = CMalpha/CMdelQ;
a_5 = K_7/Qdp;
a_6 = CNbeta/CNdelR;
a_7 = -38028.00305929; % -omega^2
a_8 = -117.00462; % -2*zeta*omega
a_9 = 38028.00305929; % omega^2

b_1 = -sin(alpha_ref);
b_2 = cos(alpha_ref);


%% ==================================================================
% Define Linear State-Space System (i.e., A, B, and C matrices):
% ( The state vector, for reference, is x=[u v w p q r phi theta]'
%   and the control vector is u=[delP delQ delR]' )
%==================================================================

 A  = [x_u     x_v    x_w     x_p     x_q    x_r    x_phi    x_thet;
       y_u     y_v    y_w     y_p     y_q    y_r    y_phi    y_thet;
       z_u     z_v    z_w     z_p     z_q    z_r    z_phi    z_thet;
       l_u     l_v    l_w     l_p     l_q    l_r    l_phi    l_thet;
       m_u     m_v    m_w     m_p     m_q    m_r    m_phi    m_thet;
       n_u     n_v    n_w     n_p     n_q    n_r    n_phi    n_thet;
       0.0     0.0    0.0     1.0     0.0    0.0    0.0      0.0;
       0.0     0.0    0.0     0.0     1.0    0.0    0.0      0.0    ];

 B  = [0.0       0.0       0.0;
       0.0       0.0       y_delR;
       0.0       z_delQ    0.0;
       l_delP    0.0       0.0;
       0.0       m_delQ    0.0;
```

```
        0.0        0.0       n_delR;
        0.0        0.0       0.0;
        0.0        0.0       0.0];

 C = [y_u        y_v      y_w    y_p   y_q   y_r    y_phi     0; % a_y
      z_u        z_v      z_w    z_p   z_q   z_r    z_phi     z_thet; % a_z
      0.0        0.0      0.0    0.0   0.0   0.0    1.0       0.0; % phi
      0.0        0.0      0.0    0.0   0.0   0.0    0.0       1.0; % theta
      0.0        1.0      0.0    0.0   0.0   0.0    0.0       0.0; % beta
      b_1        0.0      b_2    0.0   0.0   0.0    0.0       0.0; % alpha
      0.0        0.0      -1.0   0.0   0.0   0.0    0.0       1.0; % Gamma
      0.0        0.0      0.0    1.0   0.0   0.0    0.0       0.0; % p
      0.0        0.0      0.0    0.0   1.0   0.0    0.0       0.0; % q
      0.0        0.0      0.0    0.0   0.0   1.0    0.0       0.0]; % r

 D = [0.0        0.0      y_delR;
      0.0        z_delQ   0.0;
      0.0        0.0      0.0;
      0.0        0.0      0.0;
      0.0        0.0      0.0;
      0.0        0.0      0.0;
      0.0        0.0      0.0;
      0.0        0.0      0.0;
      0.0        0.0      0.0;
      0.0        0.0      0.0];


%% Following Linear system is used for analysis OF LONGITUDINAL DYNAMICS
%
%   States - {Axial Velocity, Vertical Velocity, Pitch Rate, Pitch}
%   Controls - {Flapperon Deflection}
%   Outputs - {Flight Path Angle, Pitch}
%
% d/dt [dU  =   [X_u X_w X_q -gcos(theta)  * [dU      + [0      * del_q
%       dW        Z_u Z_w Z_q -gsin(theta)     dW          Z_delq
%       dQ        M_u M_w M_q 0                dQ          M_delq
%       dTheta]   0    0    1    0];           dTheta]     0]
%
% y1 = [1 0 0 0] *[dU dW dQ dTheta]' +[0] * del_q -> for U(s)/del_q(s)
% y2 = [0 0 0 1] *[dU dW dQ dTheta]' +[0] * del_q -> for theta(s)/del_q(s)
% Outputs are axial velocity and pitch
%

A_longitudinal = [x_u x_w x_q x_thet
                  z_u z_w z_q z_thet
                  m_u m_w m_q m_thet
                  0   0   1   0];

B_longitudinal = [0
                  z_delQ
                  m_delQ
                  0];

C_longitudinal = [0 -1 0 1]; % theta - alpha = gamma

D_longitudinal = 0;

%% Following Linear System Analysis is made to study LATERAL DYNAMICS
%
%   States - {Lateral Velocity, Roll rate, Yaw Rate, Roll Angle}
%   Controls - {Aileron Deflection, Rudder Deflection}
%   Outputs - {Roll, Roll Rate, Sideslip, Yaw Rate}
%
% d/dt [dV   = [Y_v Y_p Y_r Y_phi  * [dV    + [0       y_delR  * [delP
%       dP       L_v L_p L_r 0          dP       l_delP  0           delR]
%       dR       N_v 0   N_r 0          dR       0       n_delR
%       dphi]    0   1   0   0]         dphi]    0       0]
%
%
%
%
% When we want Sideslip angle and Roll rate as output
% Remember dV is sideslip ange under assuming equilb. value of V* = 0
%
% sideslip angle, [y3   = [1 0 0 0  * [dV   +  [0 0 * [delP
% Roll Rate,       y4]     0 1 0 0]    dP        0 0]   delR]
%                                      dR
%                                      dphi]

A_lateral = [y_v    y_p     y_r     y_phi
             l_v    l_p     l_r     0
             n_v    n_p     n_r     0
             0      1       0       0];

B_lateral = [y_delP   y_delR
             l_delP   0
             0        n_delR
```

```
            0        0];

%C_lateral = [1 0 0 0]; % Sideslip angle
%C_lateral = [0 1 0 0]; % Roll rate
%C_lateral = [0 0 1 0]; % Yaw rate
C_lateral = [0 0 0 1   % Roll Angle
             1 0 0 0]; % Sideslip angle

D_lateral = zeros(2);
%==========================================================================
% The following reduced lateral and longitudinal dynamics were
% used to investigate the BTT missile modes (Appendix E):
%==========================================================================
 Ar_lat2 = [y_v     y_p      y_r;
           l_v      l_p      l_r;
           n_v      n_p      n_r];

Ar_long2 = [z_w     z_q;
            m_w     m_q];

nondim_time = mass*Vb/(Qdp*Sref);
fname = 'plots';

%============================================================================
% The following reduced lateral and longitudinal dynamics were
% used to investigate the BTT missile modes
%============================================================================
reduced_longitudinal_poles = eig(Ar_long2);
reduced_lateral_poles = eig(Ar_lat2);
all_poles = eig(A);
all_zeros = tzero(ss(A, B, C, D));
lateral_system_poles = eig(A_lateral);
longitudinal_system_poles = eig(A_longitudinal);

%% AUTOPILOT CONTROLLER STATE SPACE
matrix_1 = [a_1 0   0   0   a_2
            0   a_3 0   a_4 0
            0   0   a_5 0   a_6
            0   0   0   0   0];
matrix_2 = [-1  1   -1  1
            -1  1   1   -1
            1   1   -1  -1
            1   1   1   1];
matrix_3 = matrix_2 * matrix_1;

A_controller = [0   1   0   0   0   0   0   0
                a_7 a_8 0   0   0   0   0   0
                0   0   0   1   0   0   0   0
                0   0   a_7 a_8 0   0   0   0
                0   0   0   0   0   1   0   0
                0   0   0   0   a_7 a_8 0   0
                0   0   0   0   0   0   0   1
                0   0   0   0   0   0   a_7 a_8];
B_1 = [0    0   0   0
       a_9  0   0   0
       0    0   0   0
       0    a_9 0   0
       0    0   0   0
       0    0   a_9 0
       0    0   0   0
       0    0   0   a_9];

B_controller = B_1 * matrix_3;

Gamma_matrix = 0.25 * [-1   -1  1   1
                       1    1   1   1
                       -1   1   -1  1
                       1    -1  -1  1];

C_hat = [1  0   0   0   0   0   0   0
         0  0   1   0   0   0   0   0
         0  0   0   0   1   0   0   0
         0  0   0   0   0   0   1   0];

C_controller = Gamma_matrix * C_hat;
D_controller = zeros(4,5);

inner_loop_controller_state_space = ss(A_controller, B_controller,...
    C_controller, D_controller);


%% Plant  Analysis
P = ss(A, B, C, D);
plant_zeros = tzero(ss(A, B, C(7,:), D(7,:))); % with Gamma as output
long_plant_zeros = tzero(ss(A_longitudinal, B_longitudinal,...
    C_longitudinal, D_longitudinal));
lateral_zero = tzero(ss(A_lateral, B_lateral, C_lateral, D_lateral));
K = ss(A_controller, B_controller, C_controller, D_controller);
s = tf('s');
```

```
sI = s*eye(8);
sI_minus_A = sI - A;
sI_minus_A_inverse = sI_minus_A\eye(8);
Plant_transfer_function_matrices = minreal(zpk(C * ...
    sI_minus_A_inverse * B + D));
Plant_transfer_function_matrices.u = {'aileron', 'elevator', 'rudder'};
Plant_transfer_function_matrices.y = {'A_y', 'A_z', '\phi', ...
    '\theta', '\beta', '\alpha', '\gamma', 'P', 'Q', 'R'};
[plant_rows, plant_cols] = size(Plant_transfer_function_matrices);

%% INNERMOST RATE CONTROL LOOP

sI_minus_A_controller = sI - A_controller;
sI_minus_A_controller_inverse = sI_minus_A_controller\eye(8);
Controller_tfm = zpk(minreal(C_controller(1:3,:) * ...
    sI_minus_A_controller_inverse * B_controller(:, 1:3) + ...
    D_controller(1:3,1:3)));
[controller_rows, controller_cols] = size(Controller_tfm);
Controller_tfm.u = {'error_p', 'error_q', 'error_r'};
Controller_tfm.y = {'aileron', 'elevator', 'rudder'};

Interested_plant_tf_matrices = Plant_transfer_function_matrices(8:10,:);
open_loop = Interested_plant_tf_matrices * Controller_tfm;
[L_rows, L_cols] = size(open_loop);
open_loop.u = {'error_p', 'error_q', 'error_r'};
open_loop.y = {'P', 'Q', 'R'};
sensitivity = minreal(feedback(eye(L_rows, L_cols), open_loop));
complementary_sensitivity = minreal(feedback(open_loop, ...
    eye(L_rows, L_cols)));
complementary_sensitivity.u = {'P_c', 'Q_c', 'R_c'};
complementary_sensitivity.y = {'P', 'Q', 'R'};
pc_to_p = complementary_sensitivity(1,1);
qc_to_q = complementary_sensitivity(2,2);
rc_to_r = complementary_sensitivity(3,3);
sens_p = sensitivity(1,1);
sens_q = sensitivity(2,2);
sens_r = sensitivity(3,3);

%% Intermediate Loop (alpha, beta, phi control loop)
% p = d(phi)/dt,
% q = d(theta)/dt, where if flight path angle is small then, theta = alpha
% r = d(psi)/dt, where psi = -beta. Reference - Babister Book.
integrator = tf(1,[1 0]);
Integrator_Matrix(1,1) = integrator;
Integrator_Matrix(2,2) = integrator;
Integrator_Matrix(3,3) = -integrator; % because psi = -beta

design_plant_matrix = series(complementary_sensitivity, Integrator_Matrix);


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NEW IMPLEMENTATION WITH JESUS HELP
% P_C = k_1 * error_phi
% Q_C = k_11 * error_alpha, where k_11 = -k_2*C_N_alpha*S_ref / Mass
% R_C = k_3 * error_beta,
% error_beta = error_beta
% error_alpha = error_alpha
% Y = DU, where U = [error_phi error_alpha error_beta]
% Y = [Pc Qc Rc E_alpha E_beta]
intermediate_controller_tf = [K_1   0     0
                              0     K_11  0
                              0     0     K_3
                              0     1     0
                              0     0     1];
% Pc,Qc & Rc
interested_intermediate_controller_tf = intermediate_controller_tf(1:3,:);
% L = PK, where P = 3*3, K = 3*3
intermediate_control_open_loop = design_plant_matrix * ...
    interested_intermediate_controller_tf;

phi_com_vs_phi_tf = zpk(minreal(...
    feedback(intermediate_control_open_loop(1,1), 1)));
sens_phi_channel = zpk(minreal(1 - phi_com_vs_phi_tf));
phi_ps = zpk(minreal(feedback(design_plant_matrix(1,1),...
interested_intermediate_controller_tf(1,1))));
phi_ks = zpk(minreal(feedback(...
interested_intermediate_controller_tf(1,1), design_plant_matrix(1,1))));
phi_com_vs_phi_tf.u = {'\phi_{commanded}'};
phi_com_vs_phi_tf.y = {'\phi_{actual}'};

alpha_com_vs_alpha_tf = zpk(minreal(feedback(...
    intermediate_control_open_loop(2,2), 1)));
sens_alpha_channel = zpk(minreal(1 - alpha_com_vs_alpha_tf));
alpha_ps = zpk(minreal(feedback(design_plant_matrix(2,2), ...
    interested_intermediate_controller_tf(2,2))));
```

```
alpha_ks = zpk(minreal(feedback(...
interested_intermediate_controller_tf(2,2), design_plant_matrix(2,2))));
alpha_com_vs_alpha_tf.u = {'\alpha_{commanded}'};
alpha_com_vs_alpha_tf.y = {'\alpha_{actual}'};

beta_com_vs_beta_tf = zpk(minreal(feedback(...
    intermediate_control_open_loop(3,3), 1)));
sens_beta_channel = zpk(minreal(1 - beta_com_vs_beta_tf));
beta_ps = zpk(minreal(feedback(design_plant_matrix(3,3), ...
    interested_intermediate_controller_tf(3,3))));
beta_ks = zpk(minreal(feedback(...
interested_intermediate_controller_tf(3,3), design_plant_matrix(3,3))));
beta_com_vs_beta_tf.u = {'\beta_{commanded}'};
beta_com_vs_beta_tf.y = {'\beta_{actual}'};

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(1,1),w);
figure(100)
semilogx(w,20*log10(tf_mag(1,:)),'Color',...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - A_{y} to Aileron', 'FontSize', 24);
grid on;
axis([0.01, 1000, -60, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(1,3),w);
figure(200)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color',...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - A_{y} to Rudder', 'FontSize', 24);
grid on;
axis([0.01, 1000, -30, 30])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(2,2),w);
figure(300)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color',...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - A_{z} to Elevator', 'FontSize', 24);
grid on;
axis([0.01, 1000, -50, 15])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(3,1),w);
figure(400)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - \phi to Aileron', 'FontSize', 24);
grid on;
axis([0.01, 1000, -70, 70])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
```

```
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(3,3),w);
figure(500)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color',...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - \phi to Rudder', 'FontSize', 24);
grid on;
axis([0.01, 1000, -150, 85])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-3,2, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(4,2),w);
figure(600)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - \theta to Elevator', 'FontSize', 24);
grid on;
axis([0.001, 100, -50, 25])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(5,1),w);
figure(700)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - \beta to Aileron', 'FontSize', 24);
grid on;
axis([0.01, 1000, -80, 10])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(5,3),w);
figure(800)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color',...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - \beta to Rudder', 'FontSize', 24);
grid on;
axis([0.01, 1000, -100, 25])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(6,2),w);
figure(900)
```

```
semilogx(w, 20*log10(tf_mag(1,:)), 'Color',...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - \alpha to Elevator', 'FontSize', 24);
grid on;
axis([0.01, 1000, -90, 0])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');


w  = logspace(-3,2, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(7,2),w);
figure(1000)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - \gamma to Elevator', 'FontSize', 24);
grid on;
axis([0.001, 100, -90, 25])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(8,1),w);
figure(1100)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - P to Aileron', 'FontSize', 24);
grid on;
axis([0.01, 1000, -10, 30])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(8,3),w);
figure(1200)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - P to Rudder', 'FontSize', 24);
grid on;
axis([0.01, 1000, -100, 50])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(9,2),w);
figure(1300)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - Q to Elevator', 'FontSize', 24);
grid on;
axis([0.01, 1000, -40, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
```

```
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(10,1),w);
figure(1400)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
     [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - R to Aileron', 'FontSize', 24);
grid on;
axis([0.01, 1000, -120, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

w  = logspace(-2,3, 2000);
[tf_mag, tf_phase] = bode(Plant_transfer_function_matrices(10,3),w);
figure(1500)
semilogx(w, 20*log10(tf_mag(1,:)), 'Color', ...
     [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('Frequency Response - R to Rudder', 'FontSize', 24);
grid on;
axis([0.01, 1000, -30, 30])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');

%% AUTOPILOT PLOTS
w  = logspace(0,4, 2000);
[Controller_tfm_mag, Controller_tfm_phase] = bode(Controller_tfm(1,1),w);
figure(1600)
semilogx(w, 20*log10(Controller_tfm_mag(1,:)), 'Color', ...
     [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('K_{i} Frequency Response - Error_{p} to Aileron', 'FontSize', 24);
grid on;
axis([10, 1000, -40, 0])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');


[Controller_tfm_mag, Controller_tfm_phase] = bode(Controller_tfm(2,2),w);
figure(16000)
semilogx(w, 20*log10(Controller_tfm_mag(1,:)), 'Color', ...
     [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('K_{i} Frequency Response - Error_{q} to Elevator', 'FontSize', 24);
grid on;
axis([10, 1000, -20, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');
```

```
[Controller_tfm_mag, Controller_tfm_phase] = bode(Controller_tfm(3,3),w);
figure(160000)
semilogx(w, 20*log10(Controller_tfm_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
title('K_{i} Frequency Response - Error_{r} to Rudder', 'FontSize', 24);
grid on;
axis([10, 1000, -25, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');


w  = logspace(0,3, 2000);
[open_loop_mag, open_loop_phase] = bode(open_loop(1,1),w);
figure(1700)
semilogx(w, 20*log10(open_loop_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
title('Open Loop (P_{i}K_{i}) Frequency Response - 1^{st} Channel',...
    'FontSize', 24);
grid on;
axis([10, 1000, -50, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');
hold on;

[open_loop_mag, open_loop_phase] = bode(open_loop(2,2),w);
figure(17000)
semilogx(w, 20*log10(open_loop_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
title('Open Loop (P_{i}K_{i}) Frequency Response - 2^{nd} Channel', ...
    'FontSize', 24);
grid on;
axis([10, 1000, -50, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');
hold on;

w  = logspace(-1,3, 2000);
[open_loop_mag, open_loop_phase] = bode(open_loop(3,3),w);
figure(170000)
semilogx(w, 20*log10(open_loop_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
title('Open Loop (P_{i}K_{i}) Frequency Response - 3^{rd} Channel', ...
    'FontSize', 24);
grid on;
axis([0.1, 1000, -50, 40])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('Mach = 1.068', 'Mach = 1.5114', 'Mach = 2.0420');
hold on;


w  = logspace(0,3, 2000);
[pc_to_p_mag, pc_to_p_phase] = bode(pc_to_p,w);
[sens_p_mag, sens_p_phase] = bode(sens_p,w);
figure(1800)
semilogx(w, 20*log10(pc_to_p_mag(1,:)), 'Color', ...
```

```
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
semilogx(w, 20*log10(sens_p_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.3+0.1*jj 0.9-0.1*jj])
hold on;
title('Inner Loop P channel Sensitivities', 'FontSize', 24);
grid on;
axis([1, 1000, -50, 25])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 4);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('T Mach = 1.068', 'S Mach = 1.068', 'T Mach = 1.5114',...
    'S Mach = 1.5114', 'T Mach = 2.0420', 'S Mach = 2.0420');
hold on;


w = logspace(0,3, 2000);
[qc_to_q_mag, qc_to_q_phase] = bode(qc_to_q,w);
[sens_q_mag, sens_q_phase] = bode(sens_q,w);
figure(1900)
semilogx(w, 20*log10(qc_to_q_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
semilogx(w, 20*log10(sens_q_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.3+0.1*jj 0.9-0.1*jj])
hold on;
title('Inner Loop Q channel Sensitivities', 'FontSize', 24);
grid on;
axis([5, 1000, -50, 25])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('T Mach = 1.068', 'S Mach = 1.068', 'T Mach = 1.5114', ...
    'S Mach = 1.5114', 'T Mach = 2.0420', 'S Mach = 2.0420');
hold on;


w = logspace(0,3, 2000);
[rc_to_r_mag, rc_to_r_phase] = bode(rc_to_r,w);
[sens_r_mag, sens_r_phase] = bode(sens_r,w);
figure(2000)
semilogx(w, 20*log10(rc_to_r_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
semilogx(w, 20*log10(sens_r_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.3+0.1*jj 0.9-0.1*jj])
hold on;
title('Inner Loop R channel Sensitivities', 'FontSize', 24);
grid on;
axis([1, 1000, -50, 20])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('T Mach = 1.068', 'S Mach = 1.068', 'T Mach = 1.5114', ...
    'S Mach = 1.5114', 'T Mach = 2.0420', 'S Mach = 2.0420');
hold on;


w = logspace(-1,1, 2000);
[phi_com_vs_phi_mag, phi_com_vs_phi_phase] = bode(phi_com_vs_phi_tf,w);
[sens_phi_mag, sens_phi_phase] = bode(sens_phi_channel,w);
figure(2100)
semilogx(w, 20*log10(phi_com_vs_phi_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
semilogx(w, 20*log10(sens_phi_mag(1,:)), 'Color', ...
    [0.5+0.2*jj 0.2+0.1*jj 0.7-0.2*jj])
hold on;
title('Intermediate Loop \phi Channel Sensitivities', 'FontSize', 24);
grid on;
axis([0.1, 10, -40, 5])
```

```
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('T Mach = 1.068', 'S Mach = 1.068', 'T Mach = 1.5114', ...
    'S Mach = 1.5114', 'T Mach = 2.0420', 'S Mach = 2.0420');
hold on;


w = logspace(-4,1, 2000);
[alpha_com_vs_alpha_mag, alpha_com_vs_alpha_phase] = bode(...
    alpha_com_vs_alpha_tf,w);
[sens_alpha_mag, sens_alpha_phase] = bode(sens_alpha_channel,w);
figure(2200)
semilogx(w, 20*log10(alpha_com_vs_alpha_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
semilogx(w, 20*log10(sens_alpha_mag(1,:)), 'Color', ...
    [0.5+0.2*jj 0.2+0.1*jj 0.7-0.2*jj])
hold on;
title('Intermediate Loop \alpha Channel Sensitivities', 'FontSize', 24);
grid on;
axis([0.0001, 10, -40, 5])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('T Mach = 1.068', 'S Mach = 1.068', 'T Mach = 1.5114', ...
    'S Mach = 1.5114', 'T Mach = 2.0420', 'S Mach = 2.0420');
hold on;


w = logspace(-1,1, 2000);
[beta_com_vs_beta_mag, beta_com_vs_beta_phase] = ...
    bode(beta_com_vs_beta_tf,w);
[sens_beta_mag, sens_beta_phase] = bode(sens_beta_channel,w);
figure(2300)
semilogx(w, 20*log10(beta_com_vs_beta_mag(1,:)), 'Color', ...
    [0.7-0.1*jj 0.2+0.1*jj 0.2+0.1*jj])
hold on;
semilogx(w, 20*log10(sens_beta_mag(1,:)), 'Color', ...
    [0.5+0.2*jj 0.2+0.1*jj 0.7-0.2*jj])
hold on;
title('Intermediate Loop \beta Channel Sensitivities', 'FontSize', 24);
grid on;
axis([0.1, 10, -30, 5])
set( findobj(gca,'type','line'), 'LineWidth', 2);
h = findobj(gcf, 'type', 'line');
set(h, 'LineWidth', 5);
a = findobj(gcf, 'type', 'axes');
set(a, 'linewidth', 4);
set(a, 'FontSize', 24);
xlabel('Frequency (rad/sec)', 'FontSize', 24);
ylabel('Singular Values (db)', 'FontSize', 24);
legend('T Mach = 1.068', 'S Mach = 1.068', 'T Mach = 1.5114', ...
    'S Mach = 1.5114', 'T Mach = 2.0420', 'S Mach = 2.0420');
hold on;

end % END OF FOR LOOP
```