NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A STUDY INTO ADVANCED GUIDANCE LAWS USING COMPUTATIONAL METHODS

by

Daniel Perh

December 2011

Thesis Co-Advisors: Robert G. Hutchins
Oleg Yakimenko

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2011	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE A Study into Advanced Guidance Laws Using Computational Methods		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Daniel Perh			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

Effective guidance laws that are optimal for tactical air-to-air scenarios tend to improve the performance characteristics of the missile and increase the probability of a hit in combat. Proportional guidance is the current baseline algorithm for tactical missile guidance. Increases in computational capabilities now permit more complicated guidance laws to be implemented. This research focuses on two promising advanced guidance laws, comparing them to proportional navigation using simulation, with the kinematic boundary as the performance measure. Studies are also made of performance degradation in the presence of sensor noise.

The three guidance laws, Proportional Navigation (PN), Augmented Proportional Navigation (APN) and Differential Geometry (DG), were each simulated against a non-maneuvering target and a maneuvering target. The theoretical missile engagement envelope (the kinematic boundary) is utilized as a simple and intuitive visual aid in comparing the effectiveness of each guidance law.

Band-limited white noise is then introduced into the seeker system to determine the ability of the guidance law to deal with noise perturbations, in particular, to discover the level of noise tolerance for each guidance law.

This research used a simulation model previously developed here at the Naval Postgraduate School (NPS). This simplified six degree of freedom (6DOF) model was used in a slightly modified form to: 1) verify earlier results obtained at NPS, 2) investigate an additional guidance law, the DG law, and 3) study the effects of noise on the robustness of the various guidance laws.

14. SUBJECT TERMS Missile Guidance Laws, Proportional Navigation, Augmented Proportional Navigation, Differential Geometry Guidance, Kinematic Boundary Analysis			**15. NUMBER OF PAGES** 147
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A STUDY INTO ADVANCED GUIDANCE LAWS USING COMPUTATIONAL METHODS

Daniel Perh
Captain, Singapore Armed Forces
B.S.M.E, National University of Singapore, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2011

Author: Daniel Perh

Approved by: Robert G. Hutchins
Thesis Co-Advisor

Oleg Yakimenko
Thesis Co-Advisor

Knox T. Millsaps
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Effective guidance laws that are optimal for tactical air-to-air scenarios tend to improve the performance characteristics of the missile and increase the probability of a hit in combat. Proportional guidance is the current baseline algorithm for tactical missile guidance. Increases in computational capabilities now permit more complicated guidance laws to be implemented. This research focuses on two promising advanced guidance laws, comparing them to proportional navigation using simulation, with the kinematic boundary as the performance measure. Studies are also made of performance degradation in the presence of sensor noise.

The three guidance laws, Proportional Navigation (PN), Augmented Proportional Navigation (APN) and Differential Geometry (DG), were each simulated against a non-maneuvering target and a maneuvering target. The theoretical missile engagement envelope (the kinematic boundary) is utilized as a simple and intuitive visual aid in comparing the effectiveness of each guidance law.

Band-limited white noise is then introduced into the seeker system to determine the ability of the guidance law to deal with noise perturbations, in particular, to discover the level of noise tolerance for each guidance law.

This research used a simulation model previously developed here at the Naval Postgraduate School (NPS). This simplified six degree of freedom (6DOF) model was used in a slightly modified form to: 1) verify earlier results obtained at NPS, 2) investigate an additional guidance law, the DG law, and 3) study the effects of noise on the robustness of the various guidance laws.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

Acronyms	Definition
6DOF	six degrees of freedom
AAM	air to air missile
ABC	aircraft body centered
abg	alpha-beta-gamma
APN	augmented proportional navigation
c.g.	centre of gravity
c.p.	centre of pressure
DG	differential geometries
ECI	earth centered inertial
IMU	inertial measurement unit
IR	infra-red
LOS	line of sight
NED	north east down
NPS	naval postgraduate school
ode	ordinary differential equation
PN	proportional navigation
ZEM	zero effort miss

Symbol	Definition
$\mathbf{A_M}$	missile body frame acceleration vector $\begin{bmatrix} A_{Mx} & A_{My} & A_{Mz} \end{bmatrix}^T$
AR	wing aspect ratio
$\mathbf{A_T}$	target inertial acceleration vector $\begin{bmatrix} A_{Tx} & A_{Ty} & A_{Tz} \end{bmatrix}^T$
B_B	NED rotation matrix using quarternions $\begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$
c_{d_i}	induced drag coefficient
c_{d_0}	parasitic drag coefficient
C_N	normal force coefficient
D	drag force
e	wing efficiency relative to an elliptical planform
η_m, σ_L	missile look angle
η_t	target heading angle relative to LOS

$\mathbf{F_B}$ applied forces (ABC frame)

$$\begin{bmatrix} F_x & F_y & F_z \end{bmatrix}^T$$

F_N normal force

$\boldsymbol{g_0}$ gravitational constant vector (NED frame)

$$\begin{bmatrix} 0 & 0 & g_0 \end{bmatrix}$$

g_0 gravitational acceleration on earth surface ($9.804\ m/s^2$)

I_{3x3} 3 x 3 identity matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

J missile body inertial matrix

$$\begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}$$

κ_m missile trajectory curvature

κ_t target trajectory curvature

N' navigational constant

n_c guidance law commanded acceleration

$\boldsymbol{\omega_B}$ missile angular velocity vector (ABC frame)

$$\begin{bmatrix} P & Q & R \end{bmatrix}^T$$

ω_x angular velocity of the earth (7.292×10^{-15} rad/s)

Ω_B body rate cross product matrix

$$\begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}$$

Ω_E earth rate cross product matrix

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\omega_x \\ 0 & \omega_x & 0 \end{bmatrix}$$

Ω_q quaternion cross product matrix

$$\begin{bmatrix} 0 & P & Q & R \\ -P & 0 & -R & Q \\ -Q & R & 0 & -P \\ -R & -Q & P & 0 \end{bmatrix}$$

p missile position vector (NED or ECI frame)

$$\begin{bmatrix} x & y & z \end{bmatrix}^T$$

$\mathbf{p_r}$	relative position vector $\begin{bmatrix} x_r & y_r & z_r \end{bmatrix}^T$
$\mathbf{q}$	vector of missile quaternion's $\begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T$
ρ	atmospheric density
S_{ref}	missile cross sectional area (reference area)
σ	LOS angle
$\dot{\sigma}$	LOS angle rate
$\mathbf{T_B}$	vector of applied torques (ABC frame) $\begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T$
t_{go}	time to go
V	missile absolute speed
$\mathbf{v_B}$	vector of missile velocities (ABC frame) $\begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$
V_c	missile closing velocity
$\mathbf{v_r}$	relative velocity vector $\begin{bmatrix} v_{x_r} & v_{y_r} & v_{z_r} \end{bmatrix}^T$
0_{3x3}	3 x 3 zero matrix $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author wishes to thank Professors Robert G. Hutchins and Oleg Yakimenko for their guidance and assistance in his research. Thanks especially to Professor Hutchins, for his patience and extra lessons necessary for the author to complete this project. Special thanks to Robert D. Broadston, whose advice and help was deeply appreciated.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The history of manned flight is a relatively short one. In the span of one hundred years, we have seen technological innovations and courageous pioneers who have pushed the development of aeronautics to where we stand today. From the day the Wright brothers achieved powered flight in 1903, to the day Major Charles E. Yeager broke the sound barrier in 1947 [1], to the unmanned drones that fly over Afghanistan today, aviation history is full of stories of human courage and ingenuity: Courage to step into the unknown, risking life and limb for the advancement of science. Ingenuity to design and develop the technologies required for the wonders of modern day flight: planes that takeoff vertically, flying wings that look more like a child's boomerang than a menacing deep penetration bomber, missiles that can be launched at enemy targets even before the human eye can see them.

It was perhaps inevitable that with the advancement in fighter aircraft technology, something more than simple .50 caliber rounds that fire at a fixed point ahead of the aircraft was needed to shoot them down. Rockets have been known for hundreds of years, far longer than human flight, yet it was not until recent decades that developments in technology allowed for the evolution of the simple rocket into the sophisticated missiles of today.

In this thesis, the focus is on tactical missiles. Tactical missiles are used in scenarios where the ranges concerned are more limited and are usually guided by a seeker sensor. The seekers can be active, passive or even semi-active, utilizing electromagnetic waves from a radar, an infra-red (IR) sensor or a laser. This suite of sensing abilities allows the seeker to detect and identify the target and guide the missile to it. The ability to guide a missile to a detected target is the province of guidance laws. Perhaps the most intuitive and also one of the earliest guidance laws is the pursuit guidance law. Pursuit guidance basically states that as long as the missile is pointed at the target at all times, given enough kinetic energy, the missile will hit the target. While a simple guidance law to implement, in reality, it does not work so well because the kinetic energy available to

such missiles is limited. The rocket propellant boosts the missile up to maximum speed within seconds of launching before being consumed, and the missile than glides the rest of the distance on kinetic energy alone. Hence, in pursuit guidance, the missile is more apt to run out of kinetic energy before it can successfully close with the target. This is especially true when launched at a maneuvering target from its frontal hemisphere.

The solution to this problem is the proportional navigation (PN) guidance law. While not as intuitive as the pursuit guidance law, it is still a simple and robust concept. Basically, the concept of proportional navigation is to point the missile at a point ahead of the target so that the missile will lead the target, and this will reduce the amount of maneuvers necessary, thus conserving kinetic energy for the missile to make the intercept. The implementation is simple and the basic idea is to strive to maintain a constant line of sight (LOS) angle between the missile and the target, the premise being that a constant LOS angle would signify that the missile and the target are on a collision course. The acceleration commands are theoretically applied perpendicular to the LOS and are proportional to the LOS rate and closing velocity. This basic concept is so successful that most of the more successful guidance laws in use today tend to be extensions of the basic PN law.

In particular, the Augmented Proportional Navigation (APN) law [2], [3], and the Differential Geometry (DG) law [4], [5], are investigated in greater detail in this study as extensions of the basic PN law. Guidance laws based on Optimal Control Theory [6] are not investigated in this study due to time and scope limitations.

The remainder of this chapter highlights the literature review conducted in this field as well as the goals of this research. Chapter II lays out the simulation methodology and the selected guidance laws in detail. Chapter III describes the experimental procedures, results and analysis, while Chapter IV presents the research conclusions and suggestions for further research.

A. MISSILE GUIDANCE LAWS LITERATURE REVIEW

Simply stated, the goal of guidance is to reach the target [7]. In general, missile command guidance can occur in two basic forms. The first form is homing guidance where the missile relies on its onboard seeker to detect the target and compute the required command guidance through integral software logic circuits. The second form relies on an external source that detects the target and the missile, computes the required guidance to bring the missile towards the target and then transmits that information to the missile for flight control. Such command guidance control is advantageous, as the missile is not required to have a seeker onboard, which is a costly component. Such guidance tends to result in very good missile performance. However, command guidance is highly susceptible to tracking errors. The quality of the commanded guidance is only as good as the quality of the tracking data. Since the external source usually remains relatively stationary in space, the intercept between the missile and the target usually occurs far away from the command source. Hence measurement accuracy of the tracking data and guidance accuracy are limited.

Conversely, for homing guidance, having a seeker onboard means additional cost, but at the same time delivers improved guidance accuracy results since the seeker is continuously approaching the target as time progresses. It is thus apparent that some form of terminal homing guidance is highly desirable for Air to Air Missiles (AAM), which are usually engaging targets at some distance from the launching platform and are highly maneuverable at the same time.

In general, regardless of command or homing guidance, a guidance law ultimately acts as the determinant on how a particular set of commands for guidance is to be generated. In general, Goodstein [8] describes three guidance law general categories into which all other guidance laws can be categorized. There are many special cases that are modifications of these three basic guidance concepts (see Figure 1):

- LOS
- Pursuit
- Proportional

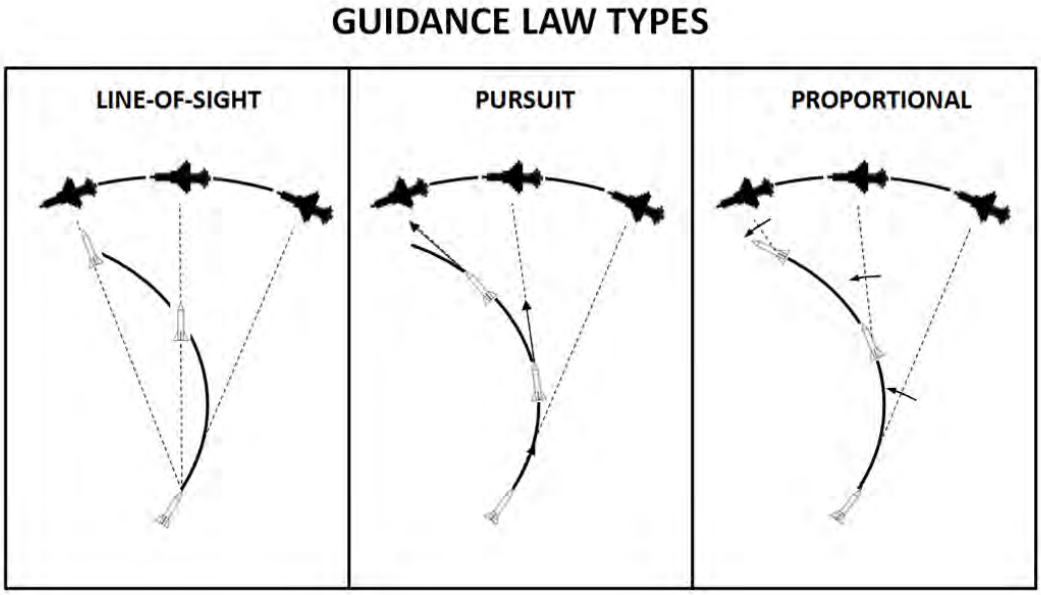


Figure 1. Graphical representation of three basic guidance laws. After [8].

For homing guidance implementations, only pursuit guidance and proportional guidance are applicable. LOS guidance by definition requires an external control vehicle to establish the LOS between the target and the vehicle along which the missile is then guided to travel. Hence, for this study into advanced guidance laws, the focus will remain on onboard seeker-capable implementations.

1. Pursuit Guidance

As described earlier, pursuit guidance works by aiming the missile directly at the target throughout the entire engagement. It is a simple implementation that is less sensitive to noise. However, it is not effective against moving targets like aircraft, as it usually ends up in an energy-consuming tail chase scenario. There are other applications where the speed advantage of the interceptor is very large (as in an air-to-surface engagement against a fixed target), in which case pursuit guidance is an effective guidance law.

2. Lead Pursuit / Lead Collision Guidance

Variations of the pursuit guidance law include lead pursuit and lead collision (see Figure 2). As the name implies, lead pursuit means that the missile is flying a course whereby it is in pursuit of a leading point just slightly ahead of the target. As this guidance law aims to predict slightly ahead of where the target's next position will be, it tends to be more effective than pure pursuit guidance and is usually able to engage targets earlier in the flight path. However, it still has essentially the same problems with pure pursuit guidance and is seldom used in systems that require intercept of high speed high maneuver targets.

Lead collision is a further extension of the lead pursuit guidance. It is also potentially the most efficient and optimal missile trajectory as it basically involves pointing the missile at the point ahead of the target where a collision would occur if the target continued in a straight line with no acceleration. This is an efficient trajectory, as it requires minimal control effort from the missile. The largest drawback is that it requires the target to fly a constant trajectory with minimal accelerations.

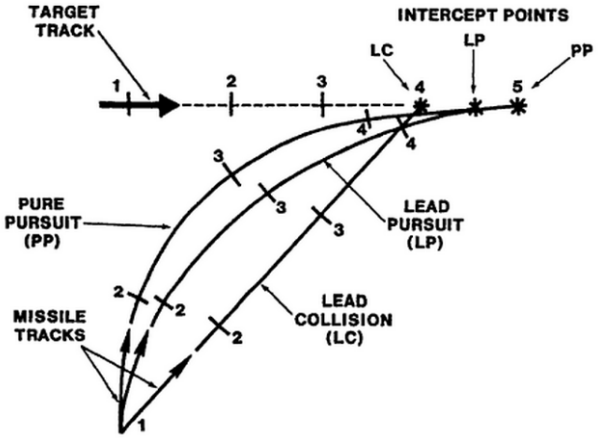


Figure 2. Pursuit guidance trajectories. From [9].

3. Proportional Navigation

As one of the most popular guidance laws, PN or some augmented form exists in many missile systems in the world today. It is based largely upon the instantaneous direction of the target relative to the missile and its first derivative with respect to time. There are two generic classes of PN, pure and true. Pure PN applies the commanded acceleration with reference to the velocity vector of the missile; whereas True PN applies the commanded accelerations with reference to the LOS (see Figure 3). PN has a highly nonlinear set of governing equations, and attempts to solve them tend to take the approach of true PN, which is more mathematically tractable as compared to pure PN. However, in practical application terms, pure PN is the more natural PN law, as implementing an acceleration vector perpendicular to the LOS as required by true PN is a physical challenge in practical applications.

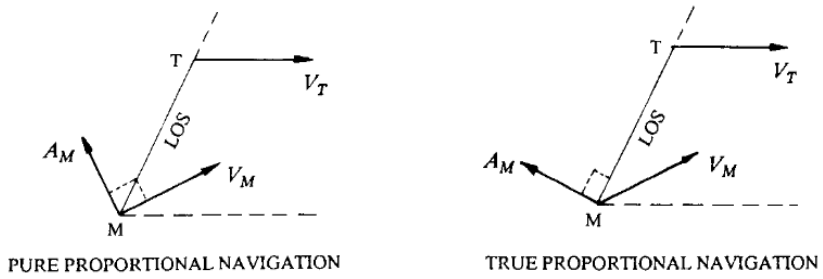


Figure 3. Geometry showing application of acceleration vector. From [10].

B. GOALS

This research thesis is focused on two areas: 1) to investigate the performance of advanced guidance laws (APN and DG) and compare them with baseline PN performance, 2) to investigate the effects of noise injection into the seeker and analyze the degradation in performance between the different compared guidance laws.

This will be achieved by modeling the system utilizing MATLAB® Simulink® and six degrees of freedom (6 DOF) models. The efficacies of the guidance laws will be compared using the kinematic boundary concept espoused by Broadston in an earlier paper [11]. The kinematic boundary is basically a visual representation of the engagement envelope with the target at the center, and the boundary represents the maximum range at which a missile can be launched at the target and be expected to hit. The missile is assumed to be launched directly at the target at all heading angles and it undergoes a short period of constant thrust to achieve maximum speed. The missile subsequently glides to the target while slowing down due to drag forces. The target is assumed to maintain a constant speed.

For the noise study, defined baseline noise is added to the seeker system as band limited white noise, and the scenarios are run over a hundred simulations to determine the percentage of hits at various factors of the baseline noise.

THIS PAGE INTENTIONALLY LEFT BLANK

II. SIMULATION METHODOLOGY

For the simulation model, Broadston's work on a simplified 6DOF model was utilized as the main simulation engine [11]. Part of this thesis investigates the assumptions that were utilized in creating the original model and the amendments that were made to update the model for use in this thesis paper.

A. SUMMARY OF SIMULATION MODEL

1. SIX DEGREES OF FREEDOM (6DOF) EQUATIONS

For a free body in space, it is possible to be translated and rotated along the three principal axes. These six freedoms of movement are the 6DOF that an unconstrained body in free space will be able to experience.

The general practice for an aircraft body-centered (ABC) coordinate frame is to define the origin at the center of gravity (c.g.) of the body with the x-axis pointing towards the nose, the y-axis pointing towards the right wing (looking at the aircraft from top down, nose pointing up) and the z-axis is 90° to both x and y axes pointing straight down.

For simulation of the air-to-air combat scenarios, the North-East-Down (NED) frame is used as the reference frame. The NED frame has its origin point placed at the earth's surface, with the x-axis pointing due north, y-axis pointing due east and the z-axis is pointing down towards the center of the earth.

For this simulation, flat earth approximations are used, as the ranges involved in air-to-air combat are relatively short as compared to tactical ballistic missiles, and a point mass model will be assumed for the flight dynamics. The following vector equations fully describe the motion dynamics of a free body in space [12], [13]:

$$
\begin{aligned}
\dot{\boldsymbol{v}}_{\boldsymbol{B}} &= -\Omega_B \boldsymbol{v}_{\boldsymbol{B}} + B_B \boldsymbol{g}_{\boldsymbol{0}}^{\boldsymbol{T}} + \frac{\boldsymbol{F}_{\boldsymbol{B}}}{m} && \text{(force - translational kinematics)} \\
\dot{\boldsymbol{\omega}}_{\boldsymbol{B}} &= -J^{-1}\Omega_B J\boldsymbol{\omega}_{\boldsymbol{B}} + J^{-1}\boldsymbol{T}_{\boldsymbol{B}} && \text{(moment - rotational dynamics)} \\
\dot{\boldsymbol{q}} &= -\frac{1}{2}\Omega_q \boldsymbol{q} && \text{(attitude - rotational kinematics)} \\
\dot{\boldsymbol{p}}_{\boldsymbol{NED}} &= B_B^T \boldsymbol{v}_{\boldsymbol{B}} && \text{(navigation - translational kinematics)}
\end{aligned}
\tag{2.1}
$$

Here $\dot{\mathbf{v}}_{\mathbf{B}}, \boldsymbol{\omega}_{\mathbf{B}}, \mathbf{g}_{\mathbf{0}}, \mathbf{p}_{\mathbf{NED}}, \mathbf{F}_{\mathbf{B}}, \mathbf{T}_{\mathbf{B}},$ are vectors and $\Omega_B, \Omega_q, J, B_B,$ are square matrices. The equations used in (2.1) are intended for flat earth approximations. Hence they do not include terms that transform the NED frame to an earth centered inertial (ECI) frame. Those terms would be needed for simulations that require modeling of the missile flight path over a spherical, rotating earth, such as simulations of ballistic missile trajectories.

It is important to note that the equations in (2.1) provide no direct correlations between the force and moment equations. In the physical world, a dynamically stable flight body would have its c.g. forward of its center of pressure (c.p.). This would result in the missile self-aligning itself to the relative wind direction during flight. Therefore, to simulate this stable dynamic behavior, a proportional-differential controller is designed to model the missile attitude such that it simulates actual physical behavior.

2. MISSILE MODEL

In order not to duplicate previous work, this thesis adopts Broadston's AIM-120 AMRAAM model [11] with its flight characteristics and dynamics. The missile model was created based on capabilities reported in open source literature and on engineering approximations. Hence it is not meant to be an exact replica of the actual missile capabilities. However, the simulation model is created modularly so the missile model characteristics can be easily modified and inserted into the simulation as required. The missile body dimensions used in this simulation is given in Table 1 and has been simplified to follow the models described in Blakelock [14] and Zarchan [3].

DESCRIPTION	SYMBOL USED IN CODE	VALUE
MISSILE BODY DIMENSIONS		
Missile mass	MASS	156.8 kg
Missile diameter	DIAM	0.1778 m
Missile length	LENGTH	3.657 m
Location of c.g. measured from the nose	XCG	1.8288 m
Length of the nose cone	LN	0.6769 m
MISSILE TAILPLANE DIMENSIONS		
Hinge line distance from nose	XHL	3.454 m
Tail root chord	CRT	0.4061 m
Tail tip chord	CTT	0.0676 m
Tail extension	TXT	0.0676 m
Tail height	HT	0.2286 m
MISSILE WINGPLANE DIMENSIONS		
Wing to radome tangency point distance from nose	XW	1.134 m
Wing root chord	CRW	0.3554 m
Wing tip chord	CTW	0 m
Wing extension	WXT	0 m
Wing height	HW	0.1778 m

Table 1. Summary of missile dimensions used in code. From [11].

a. Thrust Characteristics

The AMRAAM thrust profile is simulated as a constant 23,000 N thrust for the initial six seconds of the simulation that accelerates the missile up to a maximum speed of around 1,100 m/s, which is approximately Mach 3.5 at an altitude of 6,000 m.

b. Moments of Inertia

The missile is modeled as a thin rod for the y and z axes because the missile control surfaces are assumed to have minimal impact on the moment of inertias about the axes. The cylindrical model was selected for the x axis.

c. Drag Model

In general, the drag force is a combination of friction drag and drag caused when the integral of pressure over the whole surface area of the missile body is nonzero [13]. The components of total drag on the body can be broken down into three major components, parasitic drag (which includes friction drag and form drag), induced drag and wave drag. In reality, the three drag components are not independent and cannot be linearly added to derive total drag. However, for the assumptions of the simulation, a simplified drag model is derived from the first two components, parasitic and induced drag. The following equation describes the total drag force along the x-axis of the ABC frame [15]:

$$D = (C_{d0} + C_{di})\rho\frac{V^2}{2}S_{ref} \tag{2.2}$$

Parasitic drag, C_{d0}, is estimated using typical values from [12]. Figure 4 plots the value of C_{d0} at various Mach numbers. The plot shows the difference in parasitic drag values caused by the presence or absence of the rocket thrust plume during the boost phase and the gliding phase.

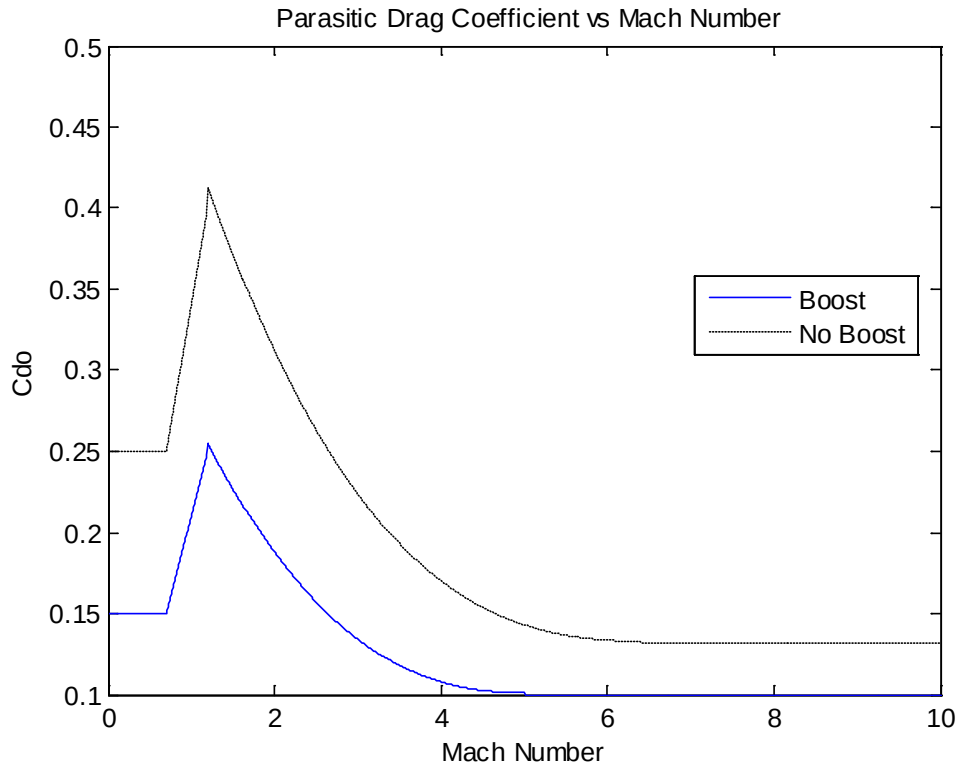


Figure 4. Parasitic drag coefficient variations with Mach no.

For induced drag, C_{di}, it is normally estimated as a function of angle of attack. In this simulation, the normal forces acting on the missile body are used instead. For subsonic flight, a crude approximation of C_{di} is made as the applied normal force times the maximum value of C_{d0} in subsonic flight. For supersonic flight, C_{di} is computed using the following [15]:

$$C_{di} = \frac{C_N^2}{\pi e(AR)} \tag{2.3}$$

Where the normal force coefficient is given by:

$$C_N = 2\frac{F_N}{\rho V^2 S_{ref}} \tag{2.4}$$

For the simulation, the normal forces and induced drag acting on the y- and z- axes are computed separately before being summed together as a vector to derive the total induced drag acting upon the system.

d. Drag Model Validation

As the figures used in generating the drag forces are estimates, a sanity check was done to ensure that the drag profile of the missile is not too far from reality. To do that, open source data for AIM-120 D claiming a max range of approximately 72 km was compared against the simulation model running the drag profile. In a tail chase scenario without target maneuvers, the missile was able to achieve a five meter radius impact on the target at a max range of 76.6 km. This represents an error of about six percent, which is acceptable in validating the parasitic drag coefficient numbers and induced drag model used in this simulation.

3. Target Dynamics Model

For the simulation, the target dynamics were modeled as a point mass model and implemented with the following vector equation [12]:

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\omega & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \omega & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ a_z \end{bmatrix} \tag{2.5}$$

In this target dynamic model, the target is modeled with lateral accelerations that are given as turn rate, ω. The vertical acceleration is given as a direct input, a_z, to the subsystem, which is equal to the gravitational acceleration.

4. Guidance Law Implementation

For the simulation, the guidance laws are invoked through MATLAB® function calls, which compute the required acceleration commands, n_y and n_z for the horizontal and vertical plane. To assist the functions in deriving the acceleration commands, the model provides the function with data from the seeker head and the inertial measurement

unit (IMU) processor. The seeker computes target range, range rate, vertical and horizontal azimuth, as well as azimuth angular rates, while the IMU processes the missile's own state vectors and orientation.

Some advance guidance laws require the entire target state vector as additional inputs, and a simple alpha-beta-gamma (abg) filter was implemented to estimate the target states for the guidance law inputs.

Section C of this chapter will discuss the three guidance laws that were chosen to be implemented in greater detail.

5. Navigation Model

The IMU provides navigational data that accelerometers and gyros would provide in real life. The missile orientation in the form of Euler angles, velocity, acceleration, angles of attack and body rotation rates are computed by the IMU. The IMU is assumed to provide this information as truth.

6. Noise Model

For the noise model, it was necessary to change the simulation solver from a variable step ordinary differential equation (ode) 45 (Dormand-Prince) solver to a fixed step size ode4 (Runge-Kutta) solver. This is to ensure that the noise input process is constant at each time step of the simulation. This results in slightly less accurate simulation runs as the fixed step size has to be set at a large enough value that ensures that each simulation run does not take too long while at the same time not so large that the simulation results become very inaccurate. For the noise simulations, the selected value for the fixed step size was 0.01s and it resulted in a difference in the results of around 1% when compared to the ode45 solver.

To model noise, band-limited white noise was added to the seeker measurement outputs of range, range rate, LOS, and LOS rate. Due to some issues with the abg-filter simulation results, it was decided that the noise for the filter outputs would be modeled in

a similar fashion to the seeker output, where the band-limited white noise was added directly to the measurement outputs of target position, velocity and acceleration.

The baseline noise model is defined in Table 2, where the accuracy of the radar systems in defining range was estimated at ±10 m, and range rate at ±1 m/s. The accuracy of LOS and LOS rate was defined as ±1 mrad for both measurements. To test for increased noise in the system, a common gain factor is applied to all power spectral density values as defined in the baseline noise model. The multiple of the gain factor then determines the amount of noise that the system is able to withstand.

Noise Parameters	Range (m)	Range Rate (m/s)	LOS (rad)	LOS Rate (rad/s)	Target Position (m)	Target Velocity (m/s)	Target Acceleration (m/s^2)
Power Spectral Density	1	1e-2	1e-8	1e-8	1	1e-2	2e-2
Standard Deviation	9.917	0.992	9.9e-4	9.9e-4	9.917	0.992	1.402
Variance	98.34	0.983	9.8e-7	9.8e-7	98.34	0.983	1.967

Table 2. Applied baseline noise values for noise simulation.

B. MODEL ANALYSIS AND MODIFICATIONS

The simplified 6DOF model that was adopted for use in this thesis was modified from the original in order to solve some anomaly issues that were found while attempting to generate the kinematic boundary comparisons. In the initial implementation of the model, it was found that certain anomalies kept appearing in the kinematic boundary plots, an example of which can be seen in Figure 5.

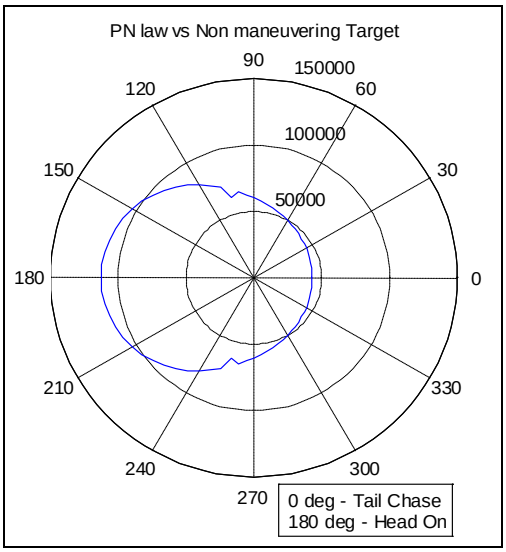


Figure 5. KB plot of PN vs. non-maneuvering target showing anomaly.

There was a predominance of anomalies happening at the broadsides and towards the rear quadrant. In the case of Figure 5, the anomaly occurs at 105° heading, with a sudden decrease in the maximum range. To investigate this problem, multiple runs were conducted at the 105° heading at 100 m range step increments and the minimum miss distance at each range was recorded and plotted in Figure 6.

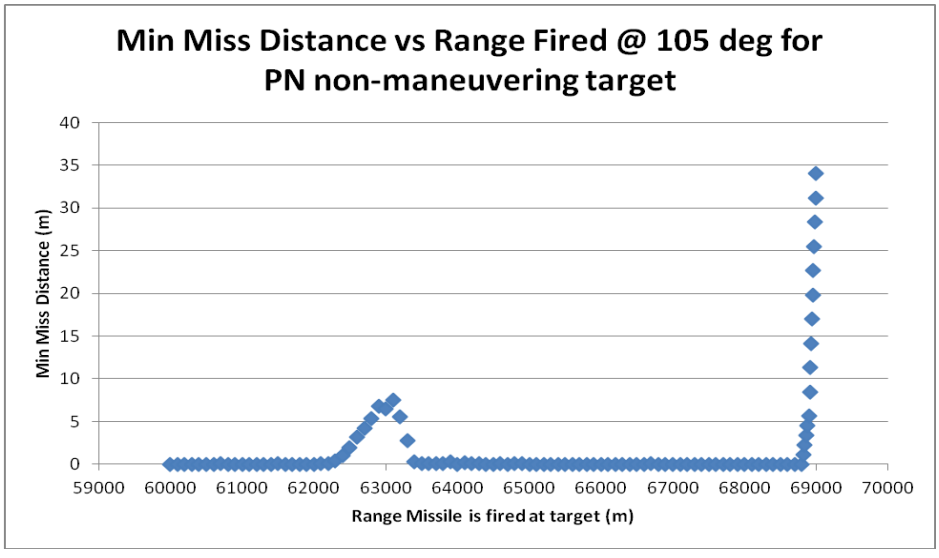


Figure 6. Minimum miss distances vs. range.

As is evident in the plot, there seems to be a specific range bin whereby the missile miss distances at end game increases suddenly before going back to the baseline miss distance. The sharp increase in miss distances at the far right side of the plot represents the maximum range of the missile.

Upon investigation, it was discovered that the region where the anomalies occur actually corresponds to the point where the missile is decelerating through the transonic region. This occurs at the range where the missile is fired such that the missile is at the end game near the target just as its velocity traverses the transonic region. Below Mach one, the previous drag model (see Figure 7) demonstrated a sudden decrease in drag coefficient, which translates to lower drag forces. The guidance law perceives this as a sudden acceleration on the part of the missile, and simulation studies indicate a miss (where a miss is defined as a minimum miss distance greater than 5 m). It can be seen that if the missile is launched even further away such that it passes through the Mach one boundary before entering end game, the missile is actually able to impact the target again.

1. Modified Parasitic Drag Curve

In a bid to reduce the transonic region's instability, the drag mode as proposed by Broadston [11] was modified slightly to allow for a gradual increase and decrease in drag coefficient through the transonic boundary instead of as a sudden step increase. The old and new parasitic drag models are shown in Figure 7 as a comparison.

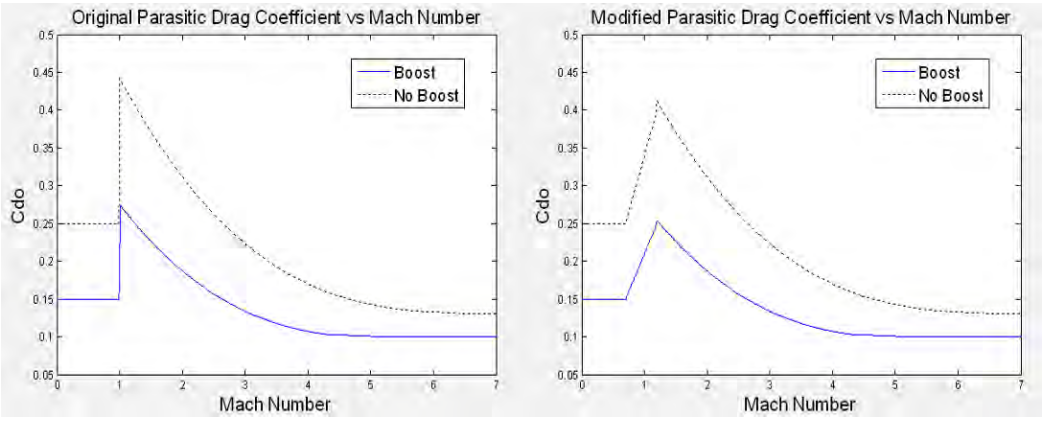


Figure 7. Original (left) and modified (right) induced drag models.

2. Additional F_x Saturation

The total maximum g-force the missile is assumed to be able to sustain is 52 times the normal gravitational acceleration. An attempt to limit the commanded acceleration on all three axes proportionally led to algebraic errors in the simulation runs. Therefore, a simplified method was adopted to limit the accelerations, which basically is a direct limitation placed on the accelerations in the individual axes not to exceed 30g each. Initially, only the commanded accelerations in the y- and z- axes were limited. However, in the simulation analysis and verification study, it was discovered that the x- axis acceleration must be limited as well to provide consistent and smooth results. Therefore the final model contains limitations on all three axes. The limiters for the x-, y- and z-axis are modeled as saturation blocks within the Simulink® model.

3. Additional Turn Limiter

In the simulation analysis stage of the project, it was discovered that using the Simulink® switch block alone to govern the moment when the target is expected to execute a constant g-turn maneuver towards the missile could lead to unexpected effects in certain simulation conditions. The switch block is basically given the time-to-go value (as derived from range-to-go and range-to-go rate) and once the time-to-go drops below the simulation determined constant value, it passes the constant turn command through to the target dynamics model. It was discovered that in certain end game simulations where the missile is in the transonic region of its flight, the sudden decrease in the missile's velocity caused the time-to-go value to increase back above the determined constant value. This causes the switch block to stop passing the turn command. To prevent such situations from occurring and introducing added anomalies to the results, a turn limiter was coded into the simulation. The turn limiter basically holds the turn command once it is passed through the switch block.

C. GUIDANCE LAWS

This section describes the three guidance laws that were selected for investigation in detail. The first two laws, PN and augmented proportional navigation (APN) were selected as the two best performing guidance law that Broadston [11] investigated. From the additional literature research, an additional guidance law that was not investigated previously using the kinematic boundary technique was selected to compare against the other two laws. This additional law is based on differential geometry (DG) [4], [5], [16], and [17], and basically generates commanded accelerations as a function of the missile flight geometry. The geometry of a typical engagement scenario is shown in Figure 8.

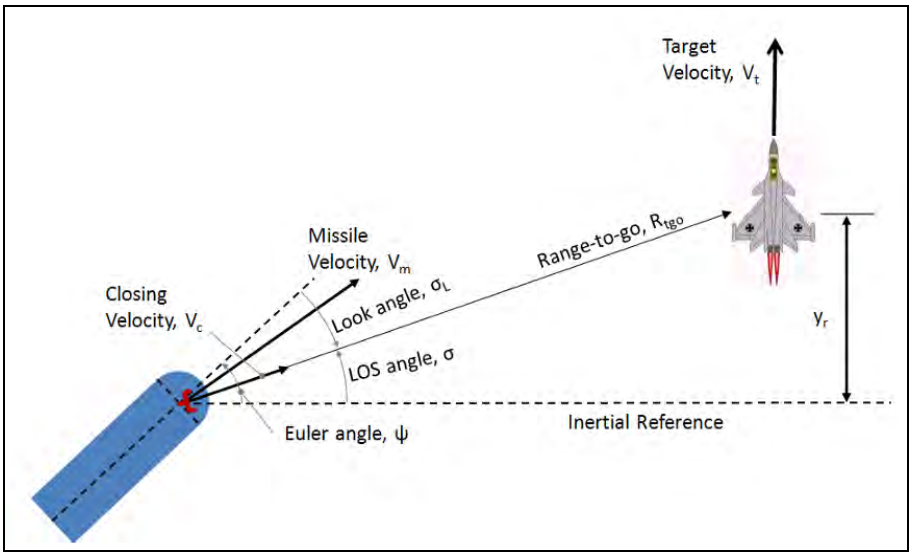


Figure 8. Missile engagement geometry. After [11].

1. Proportional Navigation (PN)

As described briefly in the literature review section of the introduction chapter, the implementation of the PN guidance law in this simulation is modeled after the True PN model where the commanded accelerations are applied perpendicularly to the LOS. The 2D commanded acceleration is given as follows [3]:

$$n_c = N' V_c \dot{\sigma} \tag{2.6}$$

While it is mathematically more tractable to compute the commanded acceleration perpendicular to the LOS, in the simulation, it is instead easier to apply the commanded acceleration perpendicular to the missile body axis instead. In order to achieve this, the commanded acceleration obtained in (2.6) is corrected to the missile y- axis as follows:

$$n_c = \frac{N' V_c \dot{\sigma}}{\cos(\sigma_L)} \tag{2.7}$$

The corrected missile acceleration is then fed directly to the missile dynamics block for simulation. The navigational constant, N' is selected to be five for the entire simulation for PN law. As described in Zarchan [3], setting the navigation ratio at five reduces the required missile acceleration advantage to 1.67, which increases the maximum distance that the guidance law will be able to hit the target.

2. Augmented Proportional Navigation (APN)

An advanced version of PN law, the APN law is basically PN augmented with a component that accounts for the maneuvering target dynamics. As described in [3], an alternative, mathematically equivalent, expression for (2.6) is as follows:

$$n_c = N' V_c \dot{\sigma} = \frac{N'(y_r + \dot{y}_r t_{go})}{t_{go}^2} \tag{2.8}$$

Where t_{go} is the time to go until intercept and y_r is defined in Figure 8. The terms in the parentheses are also referred to as the zero effort miss (ZEM) [3]. ZEM basically states the miss distance that would result if the missile and the target experienced no further accelerations and were allowed to keep traveling with the same velocity. Therefore, if we were to assume a target that maneuvers, than the ZEM equation must be augmented with an additional term that addresses the maneuver. The new equation for ZEM with target maneuver is as follows:

$$ZEM_{new} = y_r + \dot{y}_r t_{go} + \frac{1}{2} n_t t_{go}^2 \tag{2.9}$$

Where n_t is the target acceleration. Therefore, the augmented PN law becomes as follows:

$$n_c = N' V_c \dot{\sigma} + \frac{N' n_t}{2} = \frac{N'(y_r + \dot{y}_r t_{go} + \frac{1}{2} n_t t_{go}^2)}{t_{go}^2} \tag{2.10}$$

It can thus be seen that APN is basically PN law augmented with a constant proportion of the target acceleration. This increases the complexity of the system as a tracking filter that will estimate the target acceleration is now required.

The implementation of APN in the simulation is given by the vector equation from [18]:

$$\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \left(\frac{N'}{t_{go}^2} \right) \begin{bmatrix} I_{3x3} & t_{go} I_{3x3} & \frac{t_{go}^2}{2} I_{3x3} & 0_{3x3} \end{bmatrix} \begin{bmatrix} p_r \\ v_r \\ A_t \\ A_m \end{bmatrix} \tag{2.11}$$

According to [3], the optimal value for N' occurs when N' = 3. This would be evaluated subsequently in the scenario analysis. From this vector, only the values for y- and z- accelerations are used as x- is computed from thrust and drag components of the model.

3. Differential Geometry (DG)

DG guidance is a form of curvature control command where the command for missile guidance is developed based on the curvature concept in curve theory and the relative rotational motion of a pseudo-missile pointing velocity vector [16].

According to Chiou in [4], the Frenet-Serret formula for classical differential geometry curve theory was utilized in developing this control law. The fundamental theory lying behind PN and APN guidance laws is that a fixed model for the likely target maneuvers have to be assumed and is then used to derive the guidance law. This approach has certain drawbacks, not least of which is that the real world target seldom exhibit constant maneuver tactics. Therefore, the DG guidance law attempts to address this issue by ignoring specific target maneuver models in its formulation.

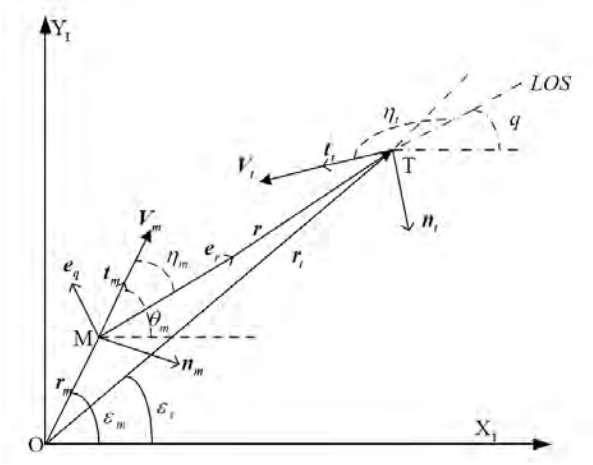


Figure 9. Geometry of engagement scenario. From [17].

A simple 2D application of the DG guidance law in the time domain is described in [17], where the engagement geometry is laid out as in Figure 9.

$$\kappa_m = N^2 \kappa_t \frac{\cos\eta_t}{\cos\eta_m} + N' \frac{\dot{r}\dot{q}}{v_m^2 \cos\eta_m} \tag{2.12}$$

In this formulation, κ_t and κ_m represents a measure of the curvature of the target and the missile's trajectories respectively. η_t and η_m represents the lead angles of the target and the missile, r' represents the closing speed of the system along the LOS and q' represents the angular rate of LOS. N' is the control gain and is not to be confused with N, which is defined as the ratio of the magnitude of the target and missile velocities, v_t and v_m.

$$N = \frac{\|v_t\|}{\|v_m\|} \tag{2.13}$$

The curvature of a circular track can be described simply as:

$$\kappa = \frac{\|a\|}{\|v\|^2} \tag{2.14}$$

Substituting equation (2.13) and (2.14) into (2.12):

$$\left(\frac{\|a_m\|}{\|v_m\|^2}\right) = \left(\frac{\|v_t\|}{\|v_m\|}\right)^2 \left(\frac{\|a_t\|}{\|v_t\|^2}\right) \frac{\cos\eta_t}{\cos\eta_m} + N' \frac{V_c \dot{\sigma}}{v_m^2 \cos\eta_m} \tag{2.15}$$

$$\|a_m\| = \|a_t\| \frac{\cos\eta_t}{\cos\eta_m} + N' \frac{V_c \dot{\sigma}}{\cos\eta_m} \tag{2.16}$$

From the respective definitions, $\cos(\eta_m) = \cos(\sigma_L)$, the DG law expressed in (2.16) can be seen to be the basic PN law augmented by a geometric term in front that requires target acceleration as well as target velocity heading information.

For implementation within the simulation, the acceleration requirements are computed respectively in the horizontal and the vertical plane to generate the required y- and z- axes acceleration.

III. SIMULATION SCENARIOS, COMPARISON AND ANALYSIS

To maintain similarity with Broadston's [11] work, the simulation scenario parameters were kept largely the same so that effective comparisons could be made. As before, the simulations are run at a fixed 6000 m altitude with the target and missile initial speed at Mach 0.83. The missile is always considered to be fired pointing directly at the target at all heading angles, and for the maneuvering scenarios, the target is always assumed to pull a 6g turn maneuver towards the missile at t_{go} = 3 s. Table 3 summarizes the various test scenarios.

Test No.	Title	Description	Simulation Settings
1A	APN Navigational Constant	To test for N′ = 3 and N′ = 5 for APN law and compare the results.	ode45 (Dormand-Prince) simulation. 6g maneuvering target 3 s prior to impact.
1B	DG Navigational Constant	To test for N′ = 3, N′ = 5 and N′ = 7 for DG law and compare the results	ode45 (Dormand-Prince) simulation. 6g maneuvering target 3 s prior to impact.
2A	Non-maneuvering target	Test for the effectiveness of each guidance law against constant velocity target.	ode45 (Dormand-Prince) simulation. No target maneuvers. N′ = 5 for all guidance laws.
2B	Maneuvering target	Test for the effectiveness of each guidance law against 6g maneuvering target.	ode45 (Dormand-Prince) simulation. 6g maneuvering target. N′ = 5 for all guidance laws.
3A	Non-maneuvering target with noise	Compare and contrast effectiveness of each guidance law upon addition of noise against constant velocity target	ode4 (Runge-Kutta) simulation. No target maneuvers. N′ = 5 for all laws. Noise factor 1x baseline noise.
3B	maneuvering target with noise	Compare and contrast effectiveness of each guidance law upon addition of noise against 6g maneuvering target	ode4 (Runge-Kutta) simulation. 6g maneuvering target. N′ = 5 for all laws. Noise factor 1x baseline noise.
4	Noise tolerance study	Compare noise tolerance levels of each guidance law at 45° and 135° heading with 100 simulation runs each. For both maneuvering and non-maneuvering scenarios.	ode4 (Runge-Kutta) simulation. N′ = 5 for all laws. 100 simulation runs. Noise factor increased until just before more than 30% misses. Tested at 90% of max range at each heading.

Table 3. Summary of simulation test scenarios.

For the kinematic boundary simulations, the scenarios were run with an initial 10000 m launch range for the missile and target at a tail chase scenario of 0° heading. The simulation then slowly increments the launch range with 1000 m steps, then 100 m steps, up to a resolution of 10 m steps to determine the max range whereby the missile

first misses the target by more than 5 m at the end game. Once the maximum range that the missile can hit the target is determined at that bearing, the heading is then incremented by 5° before the next simulation run to determine the maximum range at that bearing.

For each simulation run, there are three stopping criteria which would stop the simulation once any one of the three is met. The first stopping condition is a velocity stop and it stops the simulation once the missile speed falls below the target speed. The second condition is a range rate stop and it stops the simulation when the rate of change of the distance between the missile and the target becomes positive. The last condition is a range stop and it is triggered when the range between the missile and the target is less than 0.0001 m.

The simulation is run from 0° to 180° bearing and the maximum launch range results are all stored and finally plotted in a polar plot as the kinematic boundary of the simulation. The time taken for one such simulation run varies from as little as 2.5 hours for the simple PN guidance law with ode45 solver up to 8.5 hours for the more complicated DG law in a 6g target maneuvering scenario running the same variable step solver.

For the noise scenarios, using the ode4 fixed step solver, the simulation times range from 7.5 hours to 10 hours for a single kinematic boundary simulation.

When using the ode45 variable step solver, PN law running with a simulation tolerance of 1e-3 and minimum step size of 1e-3 and maximum step size at auto provides reasonably accurate results in a short time. However, for the more complicated APN law, the tolerance and minimum step size of the ode45 solver has to be tightened to 1e-5 and the maximum step size specified at 0.1 for reasonably accurate results.

For the DG law, even tighter simulation parameters are required. While the tolerance and minimum step size is held similar to APN, the maximum step size has to be decreased to 0.02 and the filter sample rate for the abg-filter set at 50 Hz for reasonable results to be generated. In contrast, the APN law was able to generate reasonable results with the abg-filter set to a 10 Hz sample rate with the maximum step size at 0.1.

For the noise scenarios, the fixed step size is set at 0.01 as it was empirically tested to provide the most accurate results for all 3 guidance laws within the least amount of simulation run time.

The simulation SIMULINK® models are shown in **Appendix A** while the MATLAB® codes are organized within **Appendix B**.

To demonstrate the validity of the simulation model, the engagement geometry and time history plots of certain parameters are shown in **Appendix C**.

A. TEST 1A - APN NAVIGATIONAL CONSTANT

In this test, the APN law was tested for the effects of the values chosen for the navigational constant. In [8], N' = 3 was derived mathematically as the optimal value for the guidance law. The results for this scenario are shown in Figure 10. The polar plot represents the kinematic boundary of the guidance law at the two navigational constant while the top right hand plot is the same plot plotted linearly to emphasize the difference. The bottom right hand plot directly plots the difference in max ranges between APN N'=3 and APN N'=5.

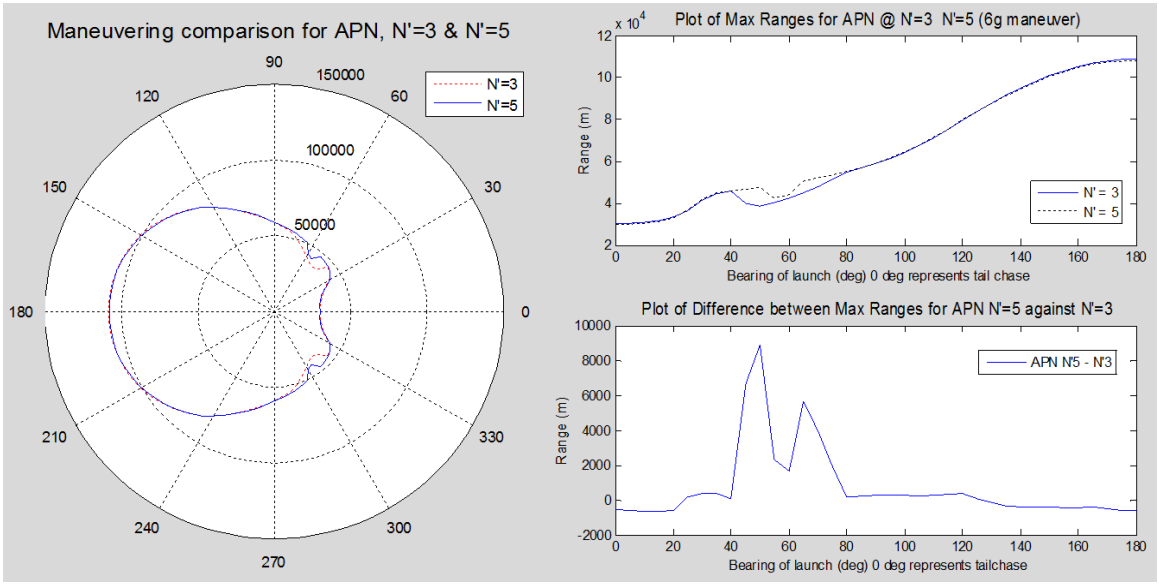


Figure 10. Comparison of APN with N'=3 and N'=5.

It can be seen from the difference plot that there is a significant improvement to the performance of the missile when N'=5 especially between 40° and 80° bearing where improvement of max range between 2000 m and 8000 m are seen. For the rest of the bearings, there is a minor advantage to N'=3 but the difference is insignificant.

It is clear that having the navigational constant set at N'=5 offers a general improvement to the performance of the missile hence for APN law, N' is fixed at N'=5 for the rest of the simulations.

B. TEST 1B - DG NAVIGATIONAL CONSTANT

As in test 1A, the DG law is tested at various navigational constant values for maneuvering target scenario. N' = 3, 5 and 7 were chosen for the test and the results are shown in Figure 11.

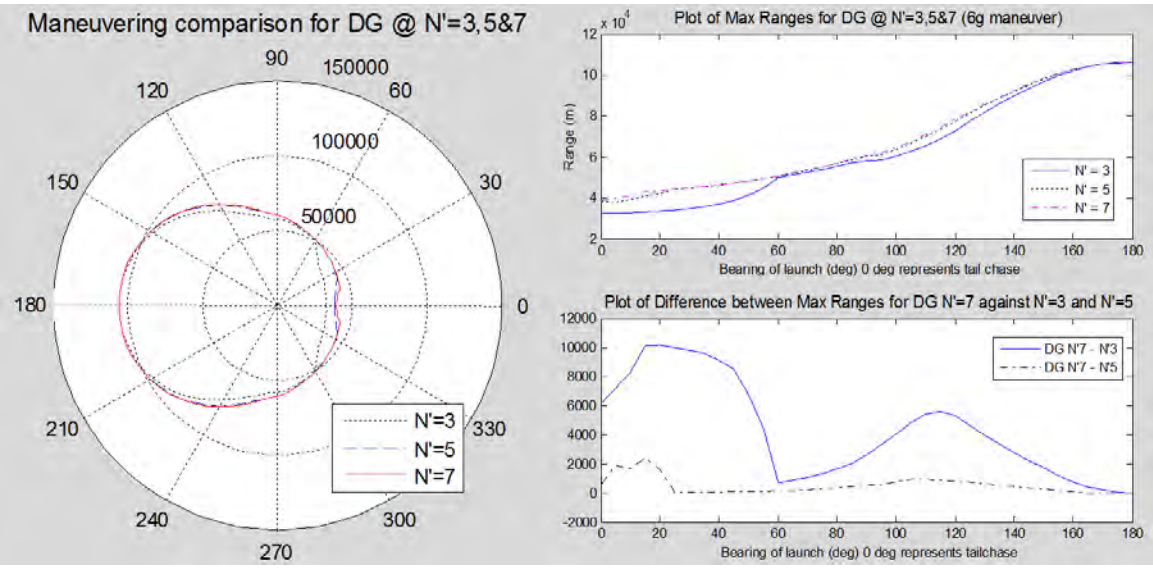


Figure 11. Comparison of DG with N' = 3, 5 and 7.

It can be clearly seen that N'=7 resulted in the best performance envelope of the missile in this maneuvering scenario. It showed significant improvements up to 10000 m over N'=3. However, it demonstrates only a slight improvement of up to 2000 m over N'=5 and only in a very narrow tail chase region of approximately 0° to 20°.

Since there is only a slight difference between N'=5 and N'=7, the navigational constant for DG is selected to be N'=5 so as to be consistent with the other two guidance laws that are being tested.

C. TEST 2A - NON-MANEUVERING TARGET

In this test, each of the three guidance laws was simulated against a constant 0.83 Mach velocity target. As both APN and DG laws are basically PN laws augmented with an additional control term that takes into account the target acceleration, it is predicted that all three guidance laws should perform similarly when the target exhibits constant velocity characteristics for the entire simulation. Figure 12 summarizes the results for all three guidance laws.

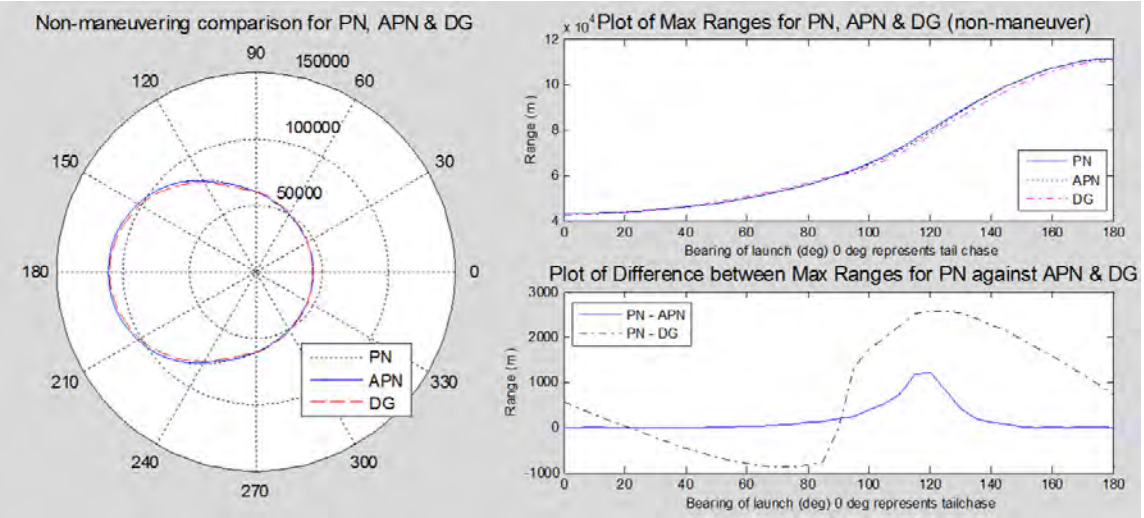


Figure 12. Non-maneuvering comparison of PN, APN and DG

It can be seen that the simulation results are similar to the predicted results. The kinematic boundary plot shows that all three laws are within 1% - 3% error ranges of each other and that the difference plot shows a difference of -1000 m to 3000 m for DG when compared to PN and a difference of up to 1000m for a narrow heading around 120° for APN when compared to PN.

This difference amounts are of little significance and it can be concluded that all three guidance laws perform similarly in a constant velocity target scenario.

D. TEST 2B - MANEUVERING TARGET

Now that the non-maneuvering results show that all three guidance laws are similar at the baseline non-maneuvering scenario, the target is modeled with constant 6g acceleration towards the missile when $t_{go} \leq 3$ s. Figure 13 summarizes the results.

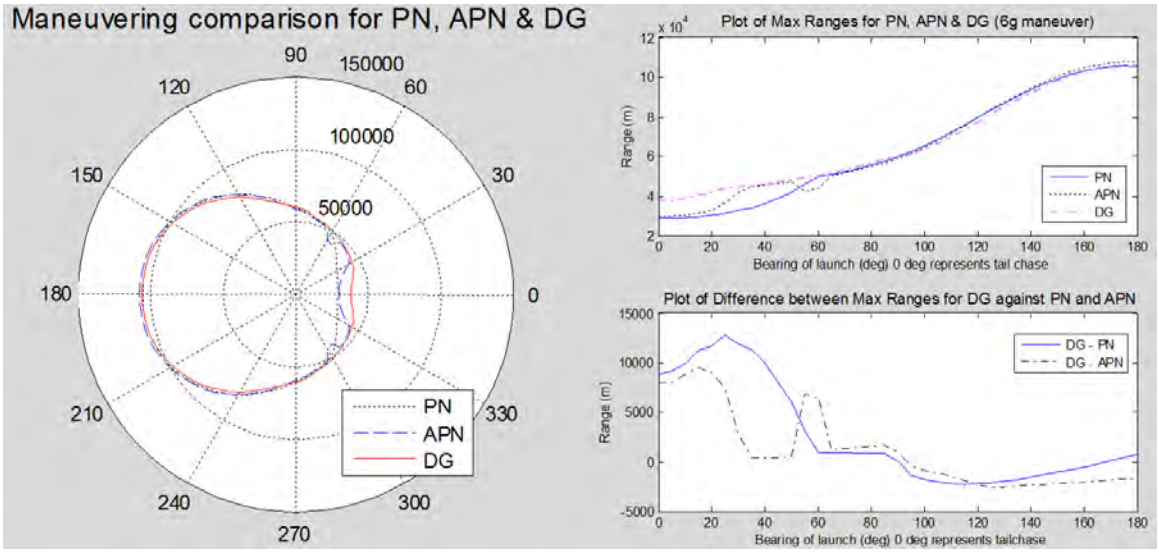


Figure 13. Maneuvering comparison of PN, APN and DG.

It can be seen from the results that all three guidance laws perform similarly in the forward and broadside quadrants of the kinematic boundary where the engagement geometry varies from 90° to head on with the target at 180°.

The rear quadrant, however, shows significant differences between the three guidance laws. It can be seen that PN is the worst performing guidance law in a tail chase maneuvering target scenario except for a narrow bandwidth around 60° heading where it performs slightly better than APN. APN in contrast behaves generally better than PN between 20° to 50° heading and, other than the slight inferior performance compared to PN, APN is essentially performing similarly to PN guidance.

The law that clearly performs better is DG where improvements of up to 12000 m in maximum range can be achieved in the rear quadrant. However, careful analysis shows that DG has a slight performance issue in the front quadrant when compared to the other two guidance laws even though the difference is insignificant.

E. TEST 3A - NON-MANUEVERING TARGET WITH NOISE

For this test, baseline noise was added to the system and a single simulation to derive the kinematic boundary was carried out. While the noise process is random and the kinematic boundary simulation will output a different result each time it is ran, the variations from one degree step to the next should even out across the entire 180° boundary. This hypothesis was put to the test and the simulation results for each of the three guidance laws in a non-maneuvering scenario was compared with their own results in a noiseless simulation and summarized in Figures 14, 15 and 16.

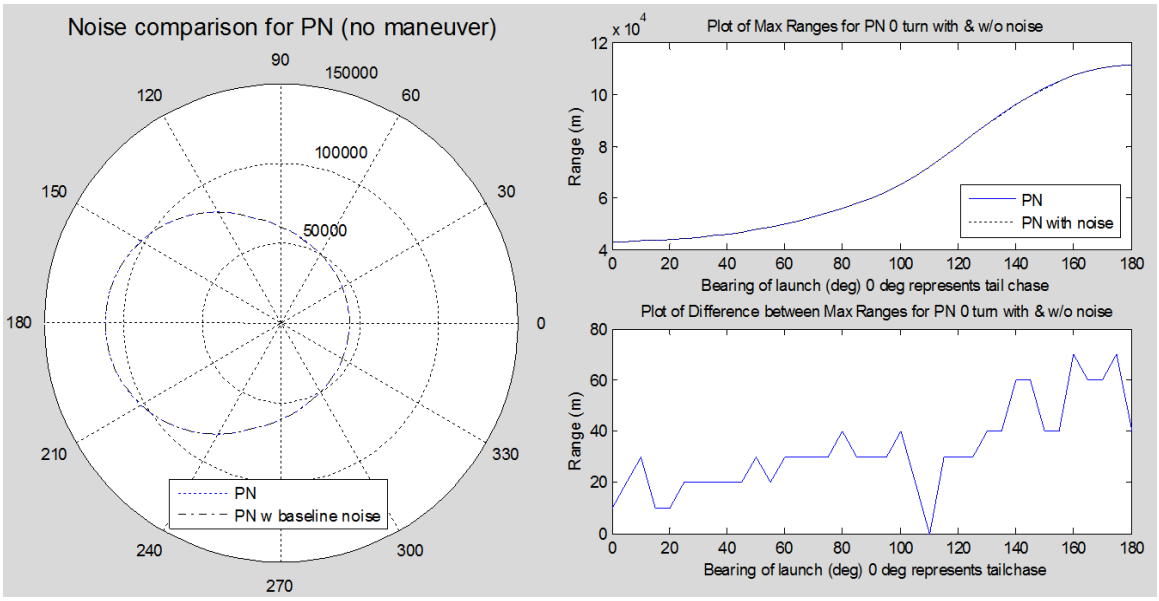


Figure 14. Noise comparison for PN (no maneuver)

It can be seen that the PN law under the influence of baseline noise is hardly affected by it and is able to perform just as well as the scenario where noise was not added.

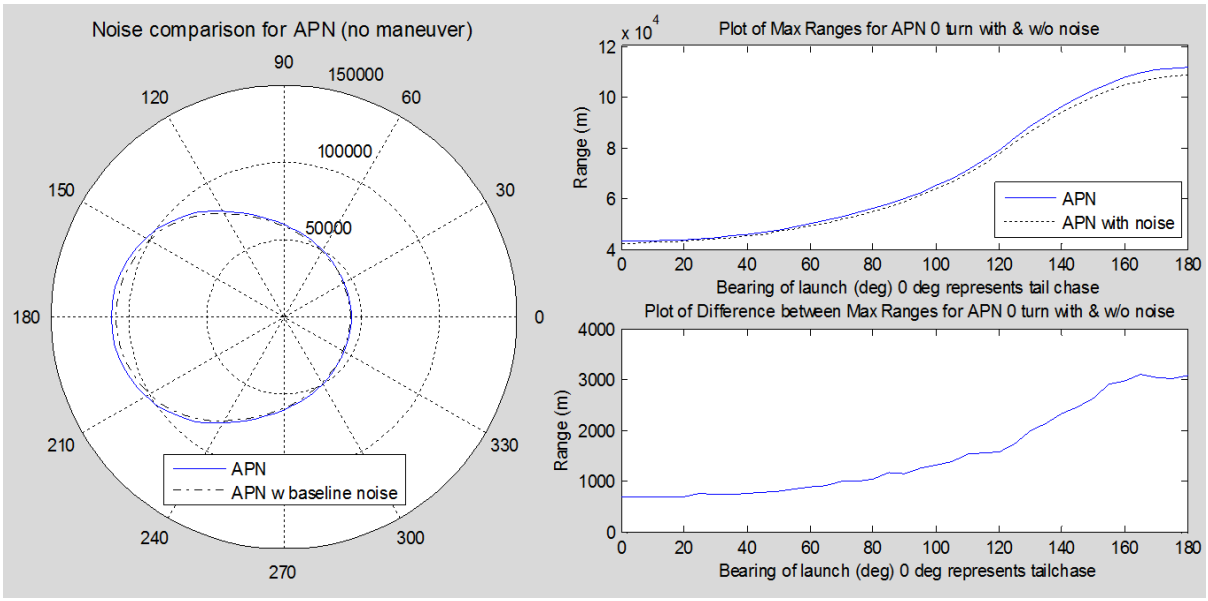


Figure 15. Noise comparison for APN (no maneuver)

For the APN law, it can be seen that it is more affected by baseline noise in the system as compared to PN. There is an increasing difference in maximum ranges, especially towards the front quadrant where the difference is as much as 3000 m.

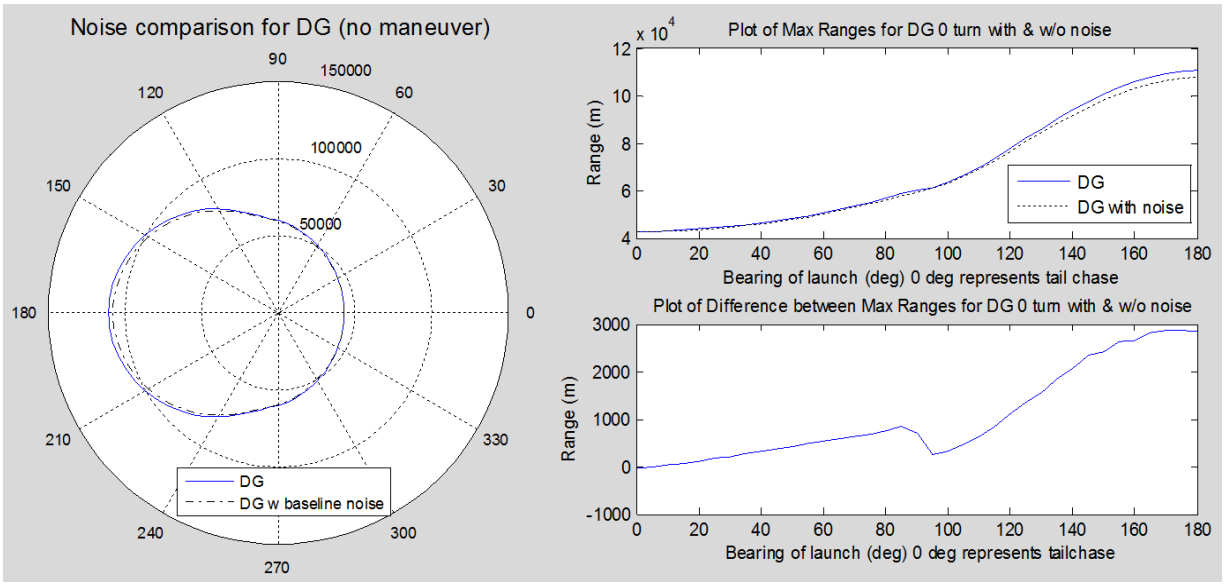


Figure 16. Noise comparison for DG (no maneuver)

It can be seen that DG suffers approximately the same level of degradation as compared to APN under the influence of baseline noise.

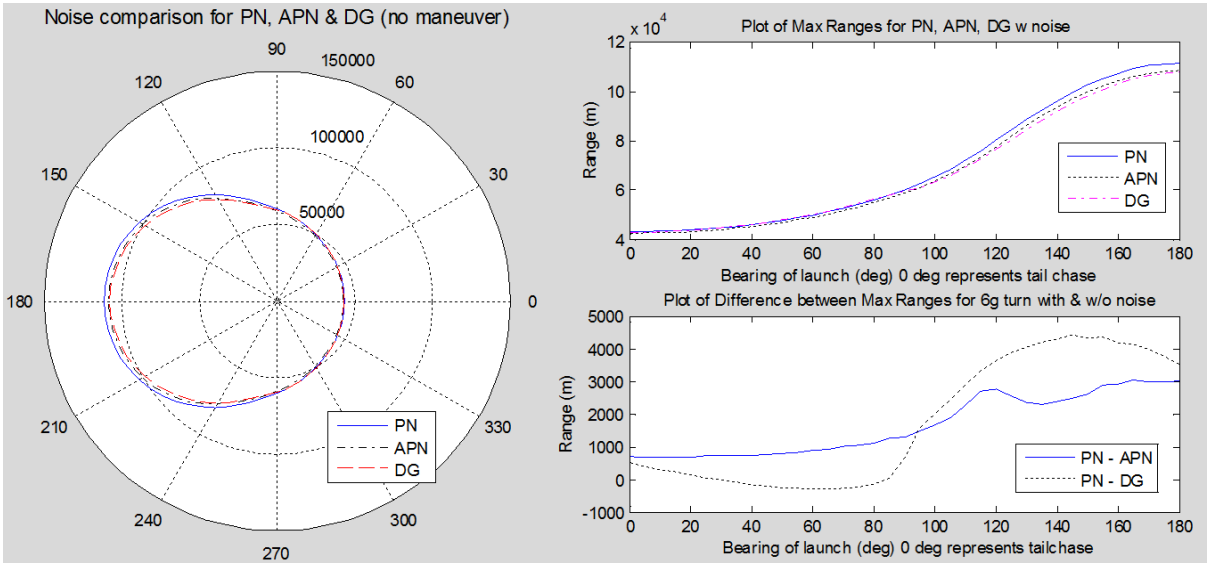


Figure 17. Noise comparison for PN, APN and DG (no maneuver)

When the three guidance laws are compared against each other under the influence of noise, it can be seen from Figure 17 that the difference between the two advanced guidance laws and PN has increased by about 2000 m when compared with the results in test 2A.

This is consistent with the expectation that the advanced guidance laws should suffer greater performance penalties due to the additional complexity of the guidance law and the added command acceleration terms.

F. TEST 3B - MANEUVERING TARGET WITH NOISE

Similarly to test 3A, the simulation is now carried out with the target maneuver scenario. The results for the individual guidance laws are shown in Figures 18, 19 and 20.

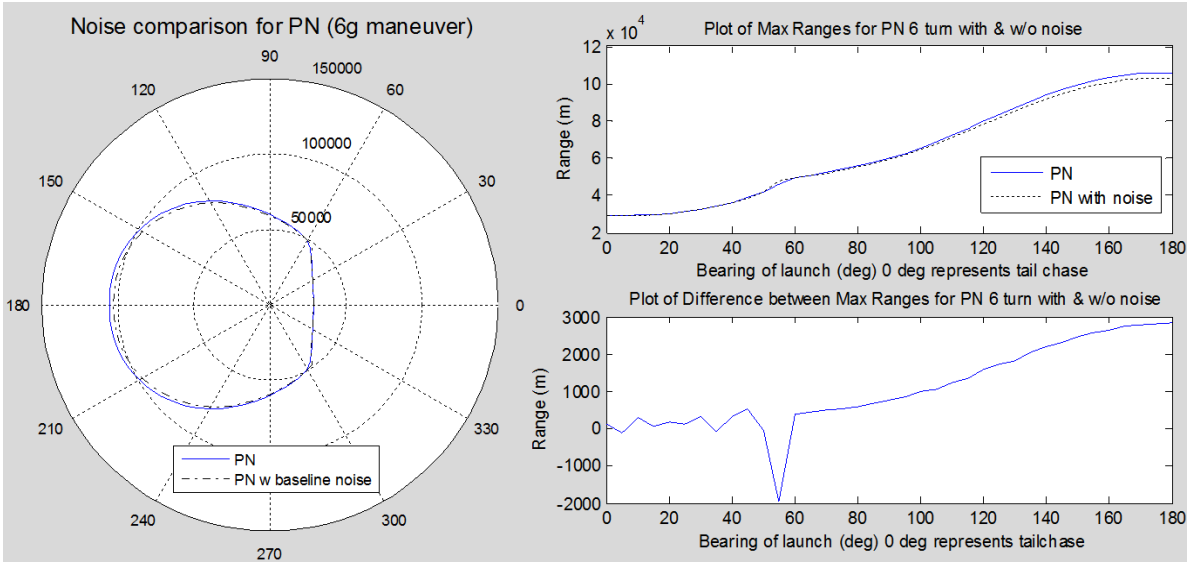


Figure 18. Noise comparison for PN (6g maneuver)

Under the no target maneuver noise scenario, PN was the only guidance law out of the 3 that did not suffer performance degradation under the effects of noise. Now that the target is maneuvering, PN is seen to suffer the same amount of performance penalty as APN and DG did under the no maneuver scenario. The spike at the 55° heading point that seems to show a region where the PN law performed better with noise then without noise is most probably an anomaly.

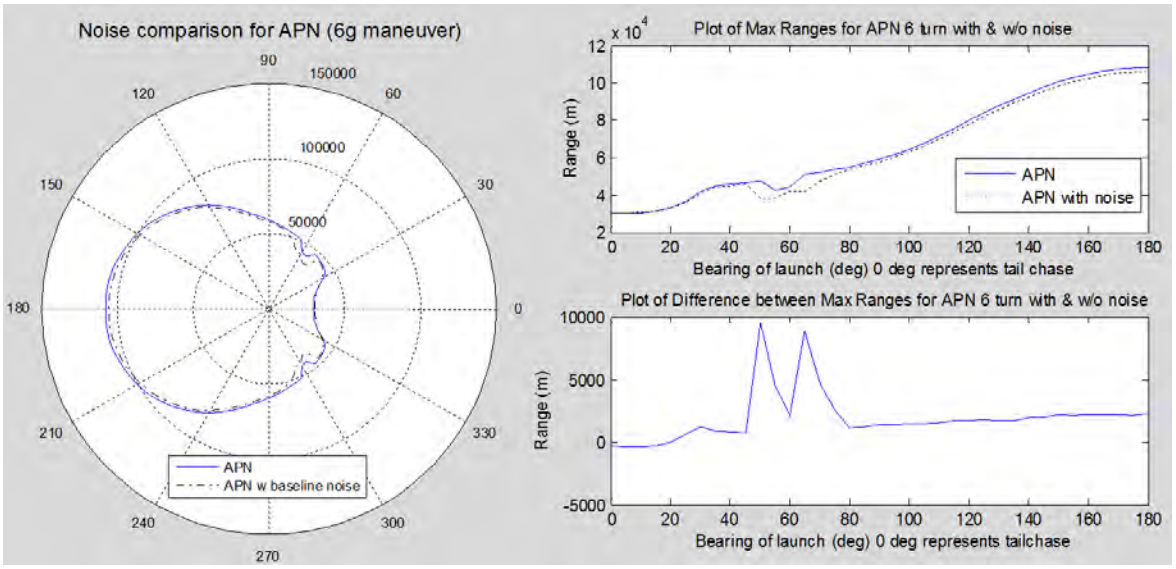


Figure 19. Noise comparison for APN (6g maneuver)

As with the results for PN, it can be seen that APN suffers much greater performance degradation around the 60° to 70° heading region. While PN 'suffered' a performance increase, APN suffers a performance decrease. The reason for this increased performance anomaly region is most likely attributable to the transonic region effect.

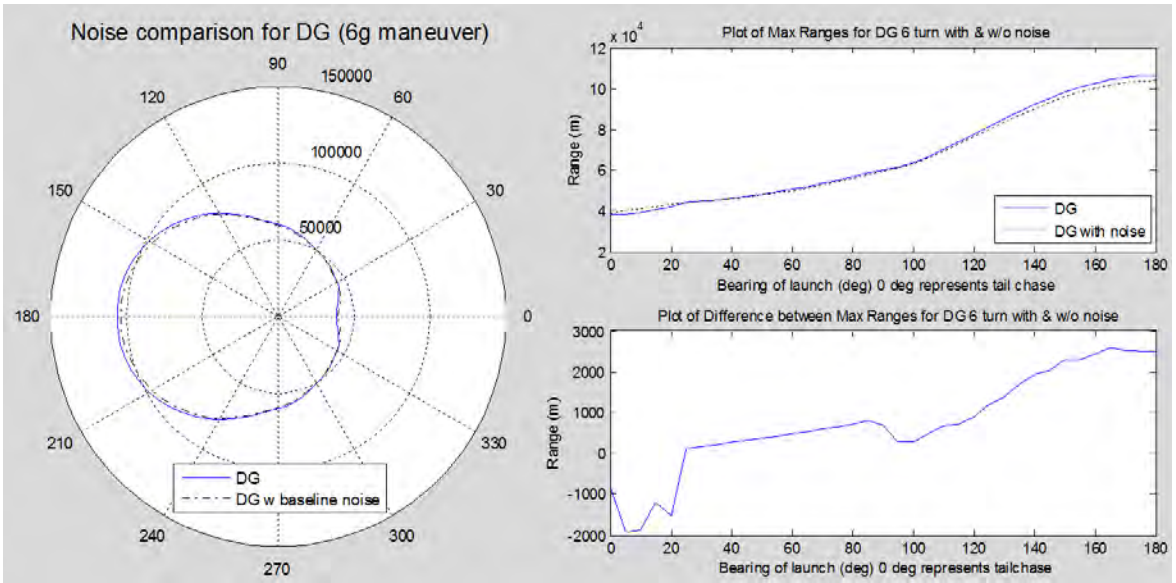


Figure 20. Noise comparison for DG (6g maneuver)

DG does not seem to suffer as much performance degradation due to the effects of noise as much as the other two guidance laws. It can be seen that DG suffers approximately the same amount of performance loss due to noise regardless of the target making or not making a maneuver in the scenario.

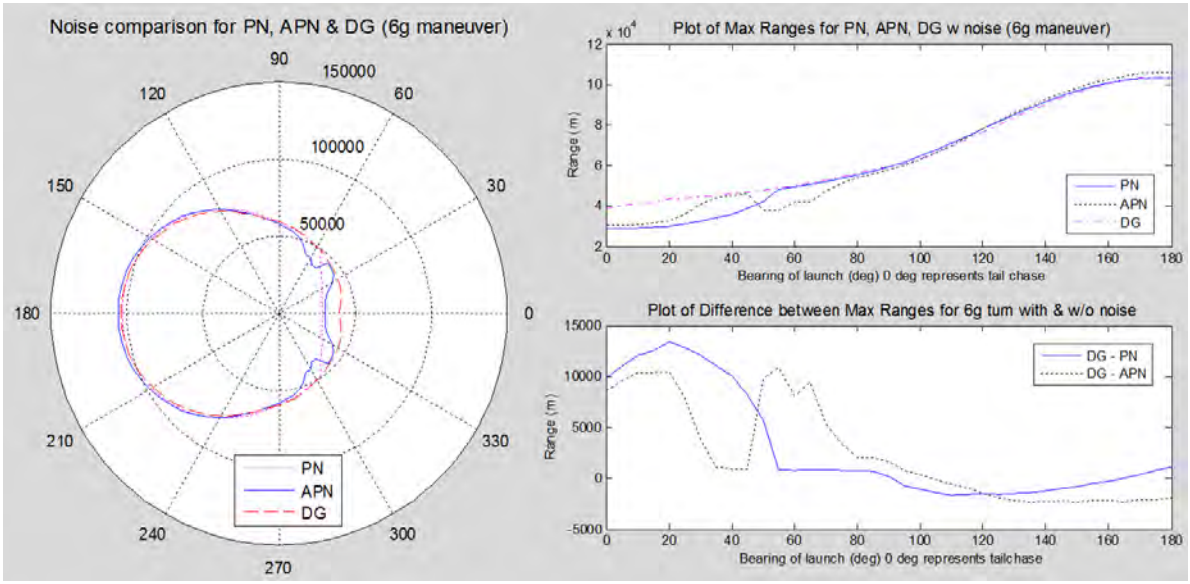


Figure 21. Noise comparison for PN, APN and DG (6g maneuver)

From Figure 21, it can be seen that the addition of noise in the simulation does not affect the relative performance of the DG law when compared to the other two laws. DG is still the best performing guidance law under target maneuvering scenarios.

G. TEST 4 - NOISE TOLERANCE STUDY

In the noise tolerance study section, each simulation data point was run for a total of 100 times with random white noise throughout. Due to the time taken to run each data point 100 times with a fixed step ode4 solver at 0.01 s step size and 100 Hz filter sample rate, the simulations were run only at the 45° and 135° heading.

For each of the three guidance laws, and for both maneuvering and non-maneuvering target scenarios, the baseline noiseless maximum range at the two bearings was first determined. A 10% reduction of the maximum range was taken and the simulations were then run at that range. The baseline noise in the system is then increased by a constant factor and the factor was increased until no less than 70% of the 100 runs were registered as successful hits on target (i.e. for the missile to come within 5 m radius of the target).

The results are summarized in Table 4.

Target Heading (deg)	Target Maneuver (g)	Guidance Law	Original Max Range (m)	10% Reduced Range (m)	Factor of Baseline Noise	Percentage Hit (%)
45	0	PN	46,800	42,120	17.7x	81
					17.8x	65
		APN	46,790	42,110	7.2x	76
					7.3x	60
		DG	47,220	42,500	16.2x	82
					16.3x	65
	6	PN	39,190	35,270	16.0x	75
					16.1x	69
		APN	46,680	42,010	7.3x	80
					7.4x	54
		DG	47,030	42,320	16.8x	75
					16.9x	55
135	0	PN	92,530	83,270	4.9x	97
					5.0x	69
		APN	92,320	83,090	5.3x	78
					5.4x	49
		DG	90,140	81,120	4.5x	82
					4.6x	38
	6	PN	88,490	79,640	6.1x	76
					6.2x	38
		APN	91,200	82,080	5.5x	94
					5.6x	66
		DG	88,780	79,900	4.7x	83
					4.8x	29

Table 4. Summary table for noise tolerance of guidance laws.

From the results, it can be seen that at the 45° heading, both PN and DG laws exhibit robust tolerance to noise and can take noise factors around 16x to 17x of baseline for both the maneuvering and non-maneuvering scenarios. However, it is significant to note that the maximum ranges for PN law is nearly 7 km shorter than DG and APN in the maneuvering scenario. In contrast, APN suffers larger performance degradation under the effects of noise and has a noise tolerance of only around 7x though at a maximum range similar to that of DG.

At the 135° heading, all three guidance laws exhibit similar noise tolerance characteristics of about 5x the baseline noise.

As can be seen in test 3B, the three guidance laws suffered similar levels of performance degradation at the forward quadrant of the engagement scenario. For the rear quadrant, it could be seen that PN and DG suffered similar performance degradation when compared to their respective noiseless scenario performances but APN suffered additional performance losses at the rear quadrant. That result is reflected in the noise tolerance study at the 45° heading where it can be seen that the greater the performance degradation the guidance law suffered, the lower the noise tolerance of that guidance law at that particular heading.

The full data and histogram plots for each of the scenarios described in Table 4, with a hit percentage greater than 70%, is presented in **Appendix D**.

THIS PAGE INTENTIONALLY LEFT BLANK

IV CONCLUSION

A. CONCLUSIONS

The developed simplified 6DOF simulation model proved to be a useful tool in the modeling, simulation and evaluation of the different guidance laws. Even with a set of simplifying assumptions about physical system performance characteristics, the simulation results matched expectations.

As a result of the simulations, it was proved that a navigational constant of N' = 5 is optimal for both the APN law as well as the DG law.

For the maneuvering target scenario, this study has shown that while the APN law does indeed perform better than the PN law except for a very narrow region, the new DG guidance law, which is based upon target motion geometries instead of expected target maneuvering models, performed the best out of the three guidance laws tested. With the DG guidance law, significant improvements to the missile performance in the rear quadrant, tail chase scenario were achieved.

For the non-maneuvering target scenarios, it was shown that while all three guidance laws performed similarly under no noise conditions, when noise was added to the system, both APN and DG suffered greater performance degradation as compared to PN. This is intuitive as the more complex guidance laws are expected to be affected by noise to a larger extent.

On the contrary, when noise was injected for the maneuvering scenarios, the DG law was shown to be the most robust performer instead of PN, while APN suffered the greatest performance degradations, especially around the 60° heading region. The DG guidance maintained its performance advantage even with the addition of noise and is shown to be an overall superior performing guidance law that can be relied upon in both maneuvering and non-maneuvering scenarios.

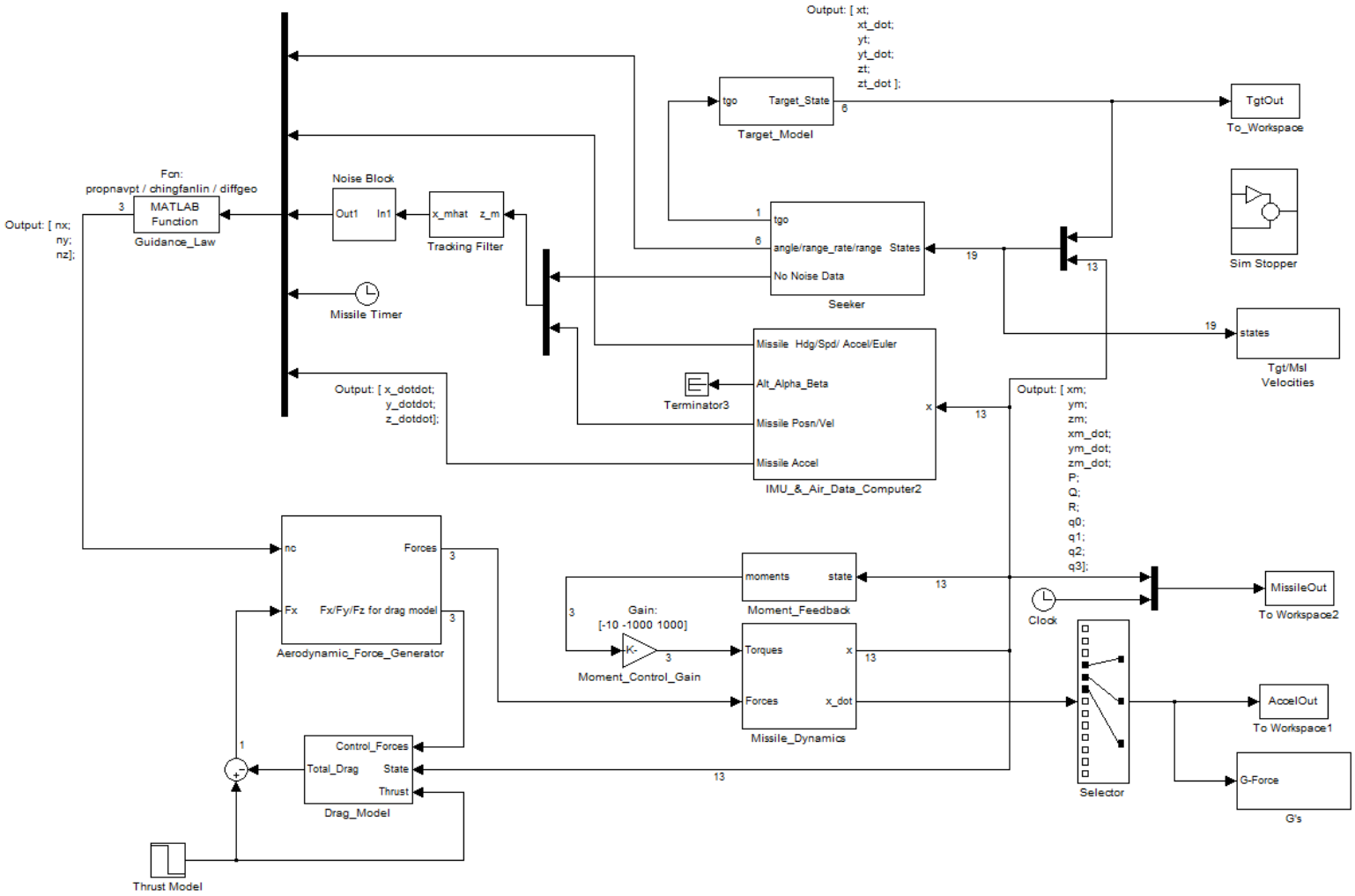
B. FUTURE RESEARCH

Encouraged by the results of the current study, future research can address the following:

- Increasing model fidelity with more accurate sensor and controller models.
- Expanding the kinematic boundary concept to three dimensional scenarios and considering a broader range of engagement scenarios.
- Comparing other guidance laws against the ones considered in this study.
- Having a more thorough study in noise implementation and specific filtering algorithms.

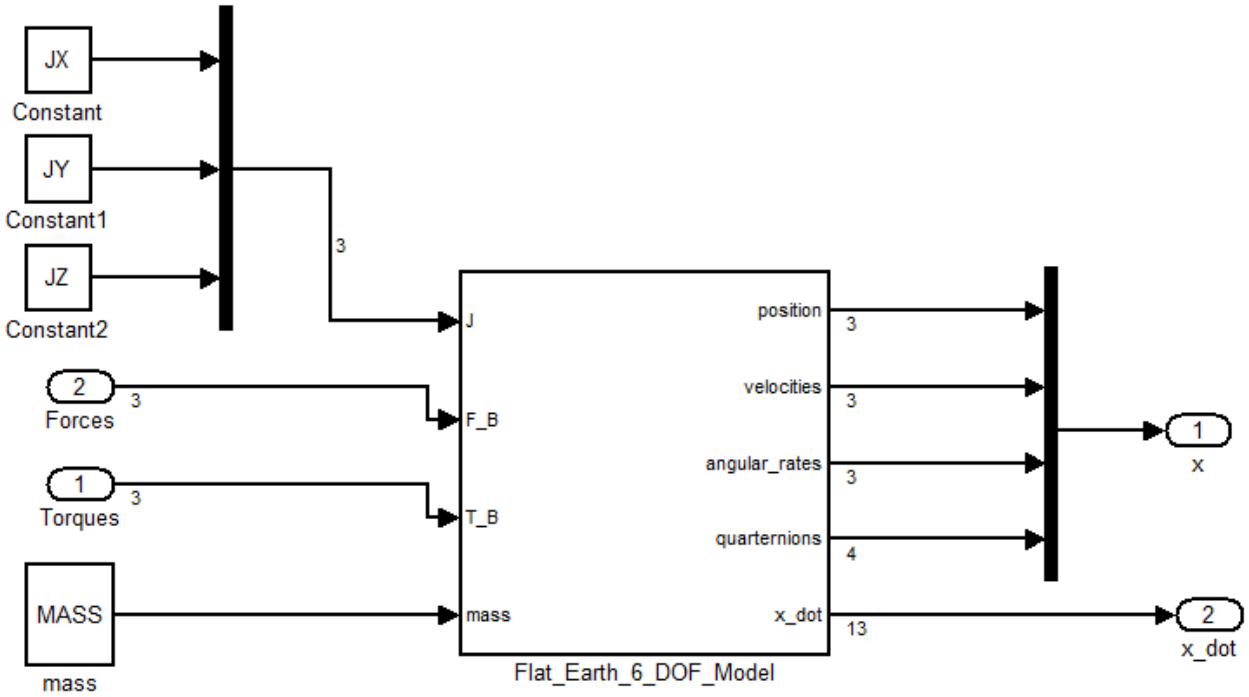
APPENDIX A. SIMULINK® MODELS

The SIMULINK® block diagrams used in the simulation are presented in this appendix. The simulation is created as a single model and functions that are not required can be modified or switched off as required.



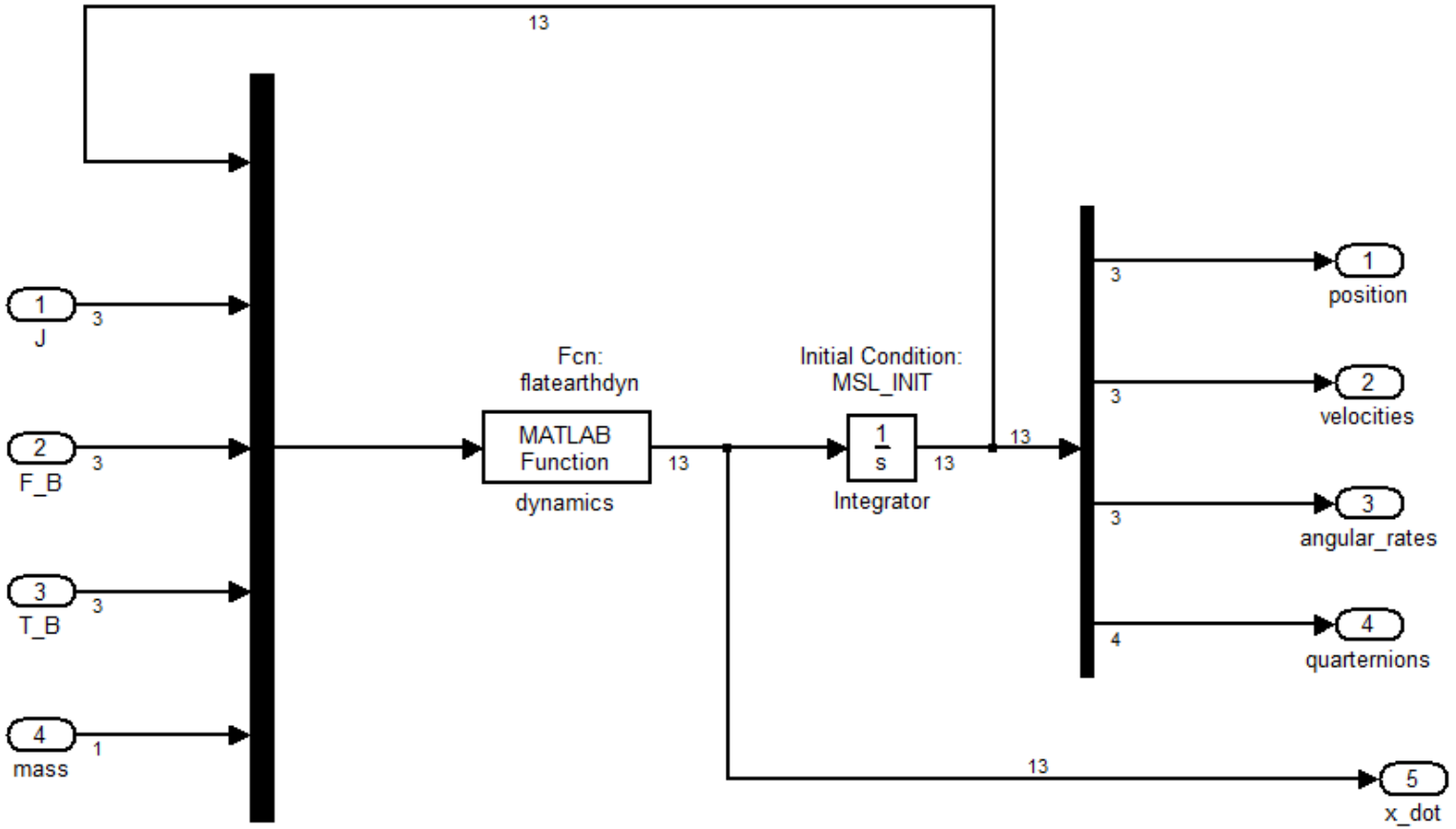
SIMPLIFIED 6DOF SIMULATION

>MODEL

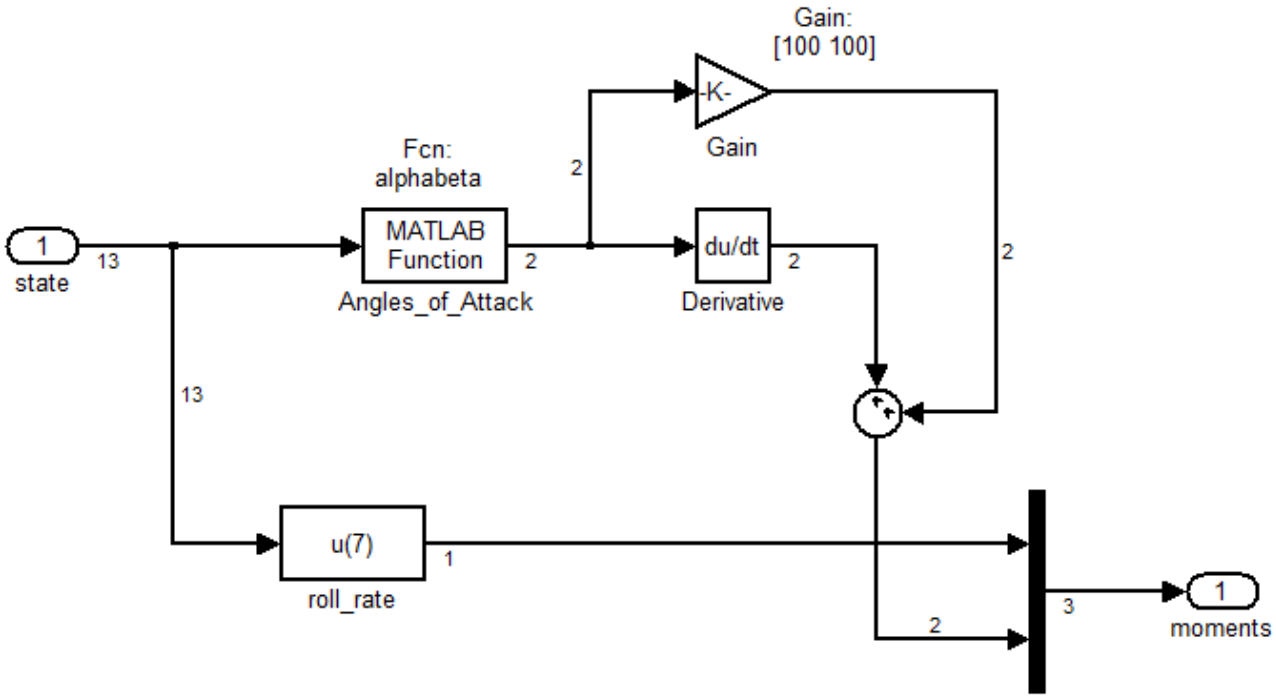


Flat Earth 6DOF Dynamics from Stevens & Lewis "Aircraft Control & Simulation"

>MODEL/Missile_Dynamics

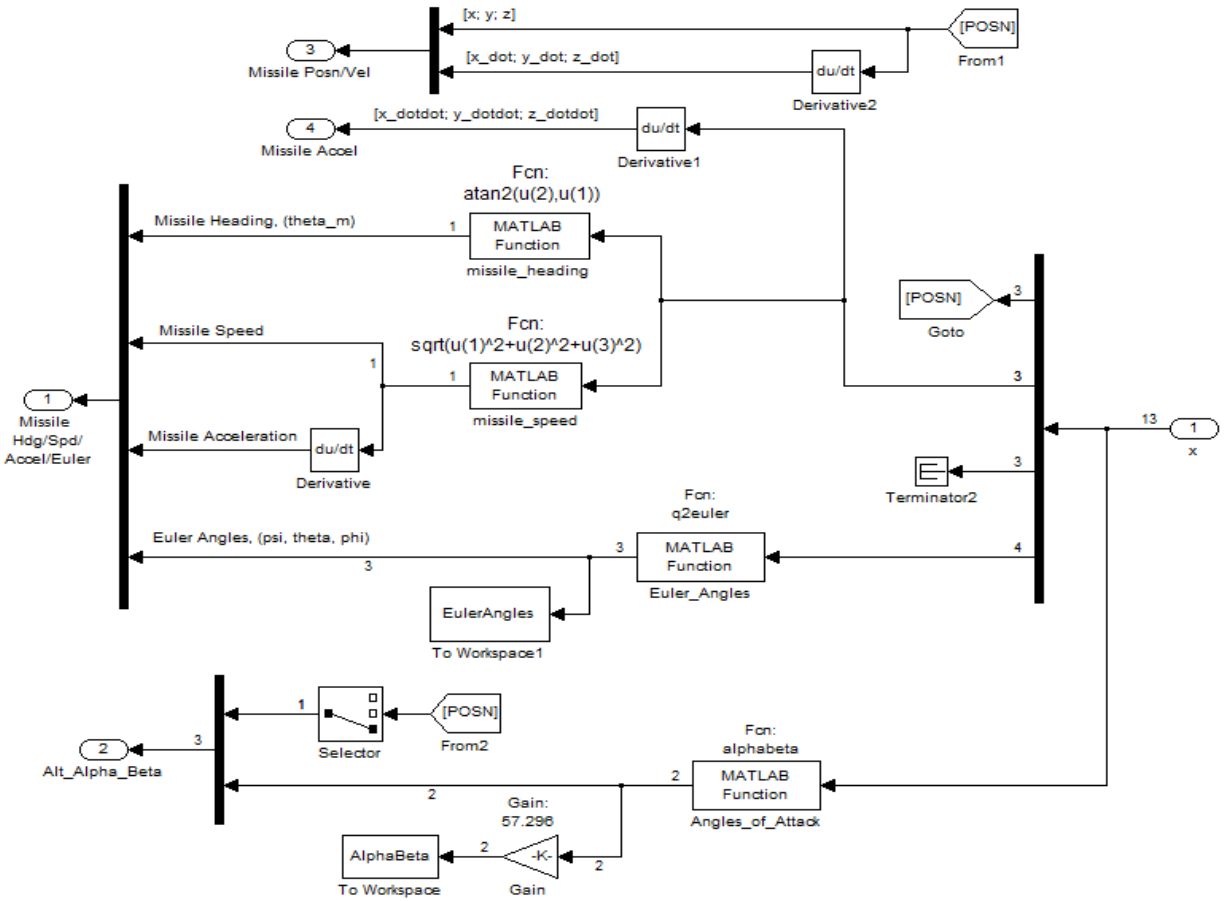


>MODEL/Missile_Dynamics/Flat_Earth_6_DOF_Model



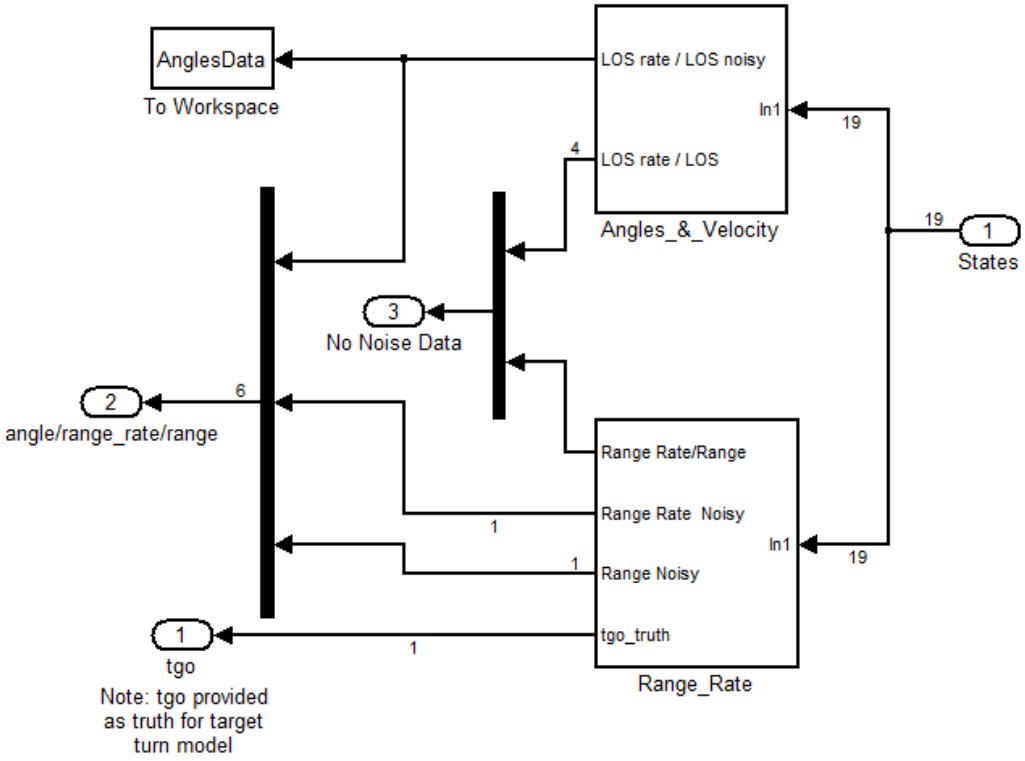
PD controller required to model missile weather vaning

>MODEL/Moment_Feedback



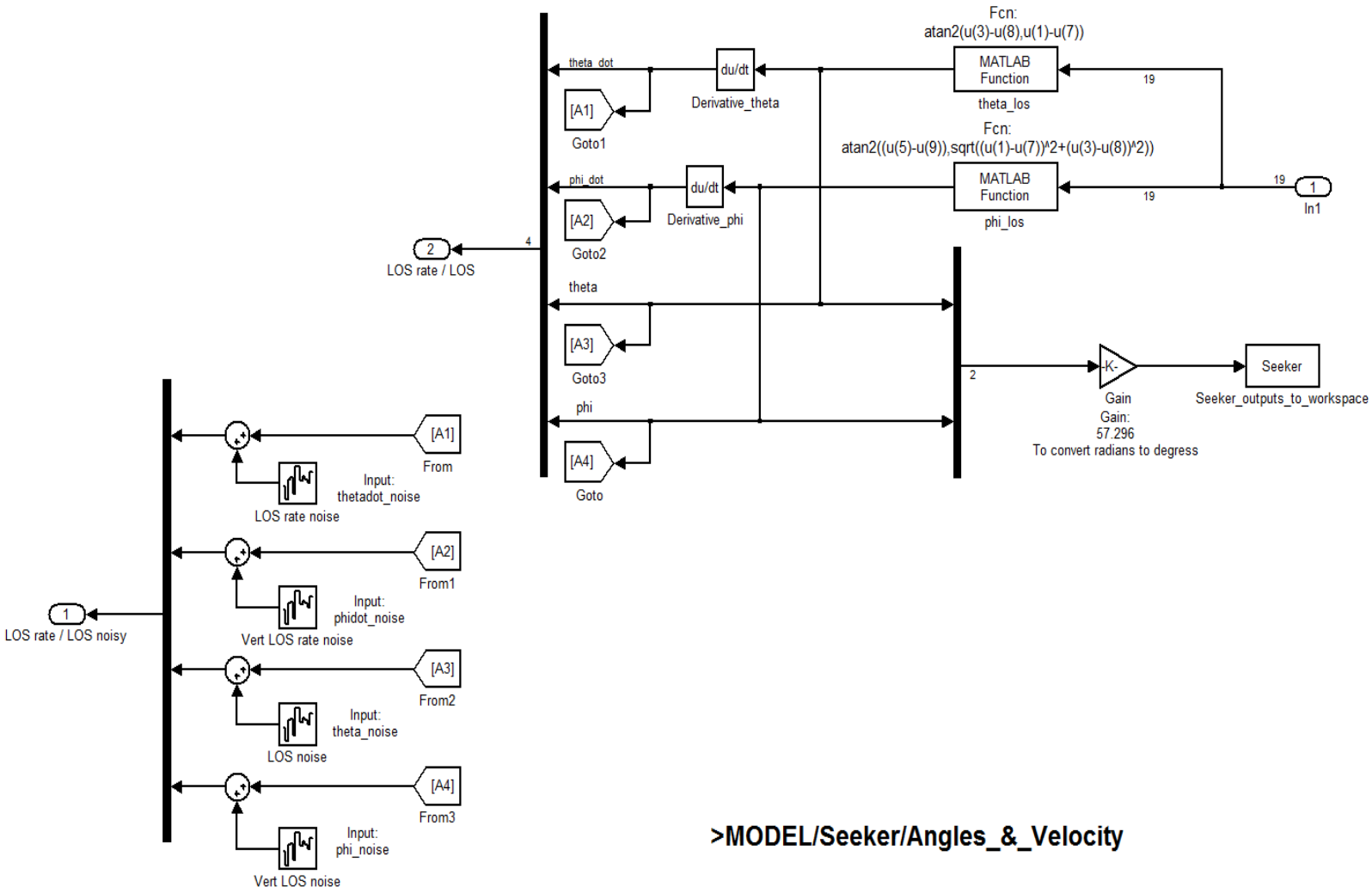
Simulation of Missile IMU and Air Data Computer

>MODEL/IMU_&_Air_Data_Computer2

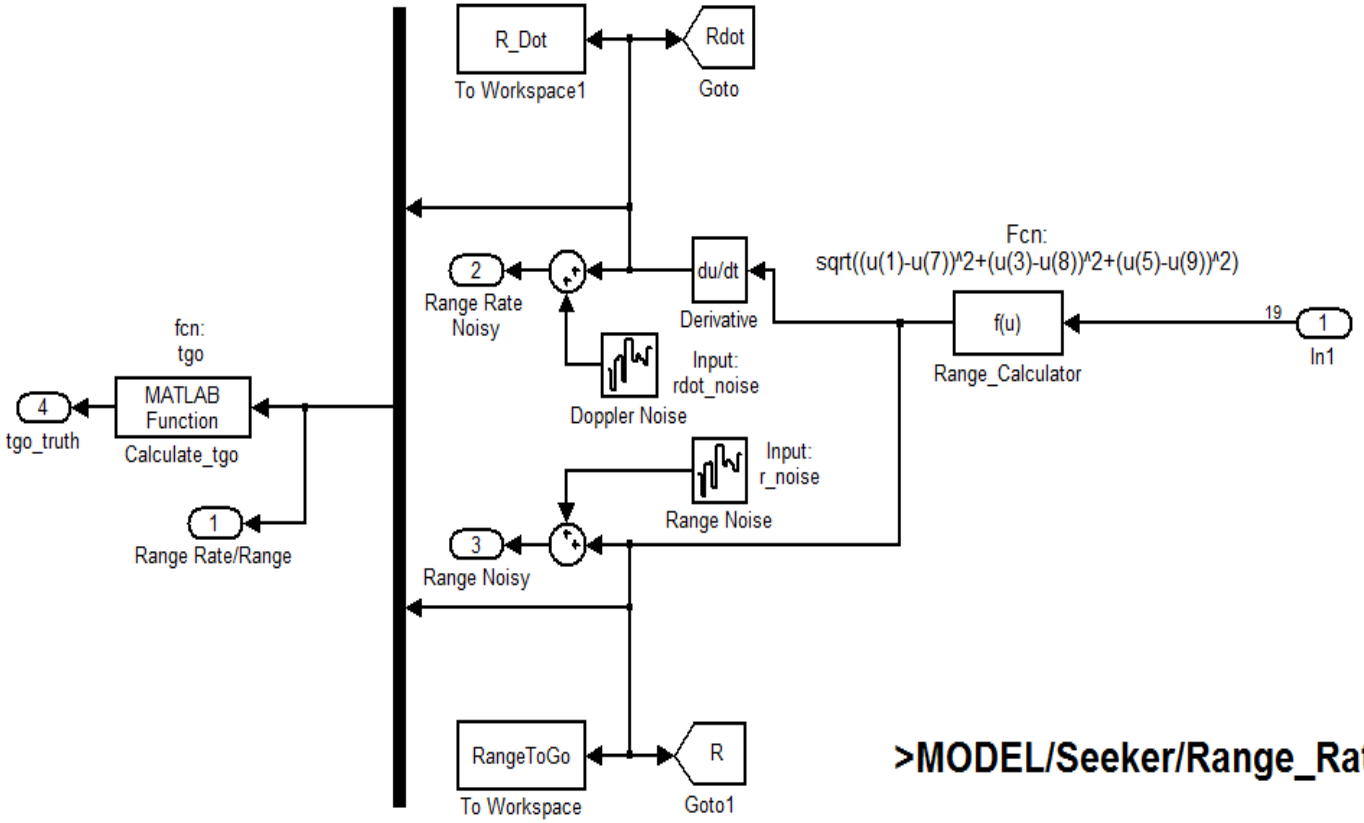


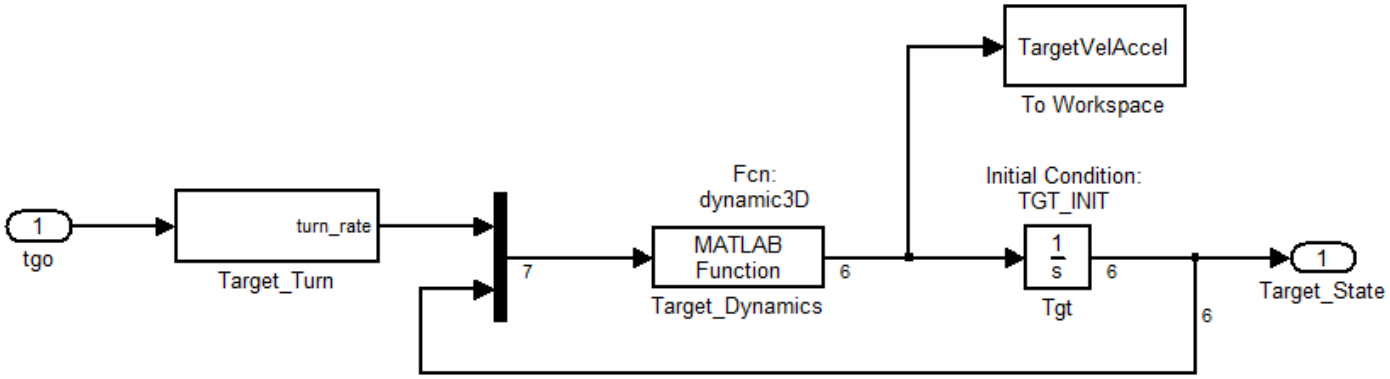
Simulator Seeker Head All Data Available to Guidance Law
User can select which to use

>MODEL/Seeker



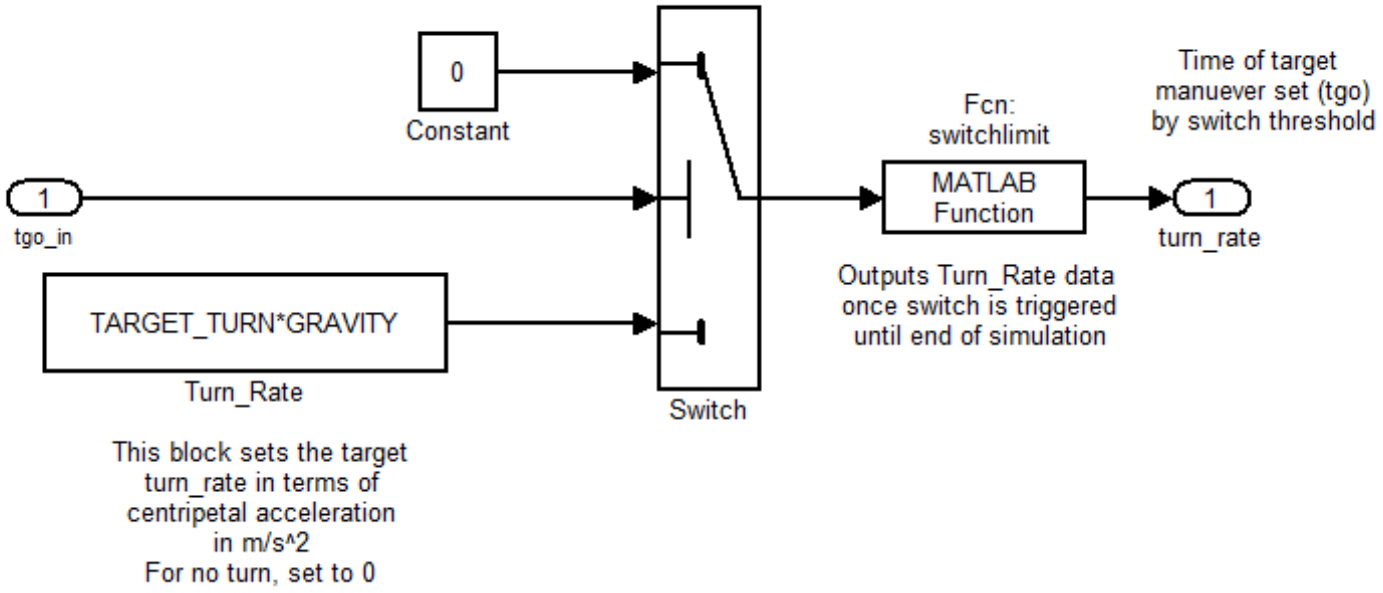
>MODEL/Seeker/Angles_&_Velocity


R_Dot
To Workspace1
Rdot
Goto
Fcn:
sqrt((u(1)-u(7))^2+(u(3)-u(8))^2+(u(5)-u(9))^2
2
Range Rate
Noisy
du/dt
Derivative
f(u)
Range_Calculator
19
1
In1
Input:
rdot_noise
Doppler Noise
fcn:
tgo
4
tgo_truth
MATLAB
Function
Calculate_tgo
Input:
r_noise
Range Noise
1
Range Rate/Range
3
Range Noisy
RangeToGo
To Workspace
R
Goto1
>MODEL/Seeker/Range_Rate

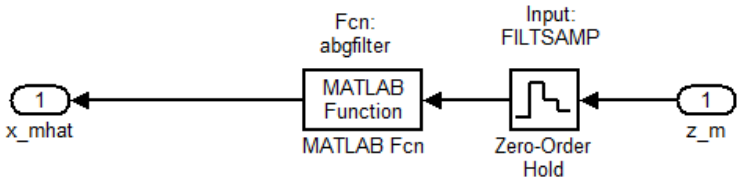


Missile Guidance Simulator Target Model
Six Dimensional State Vector

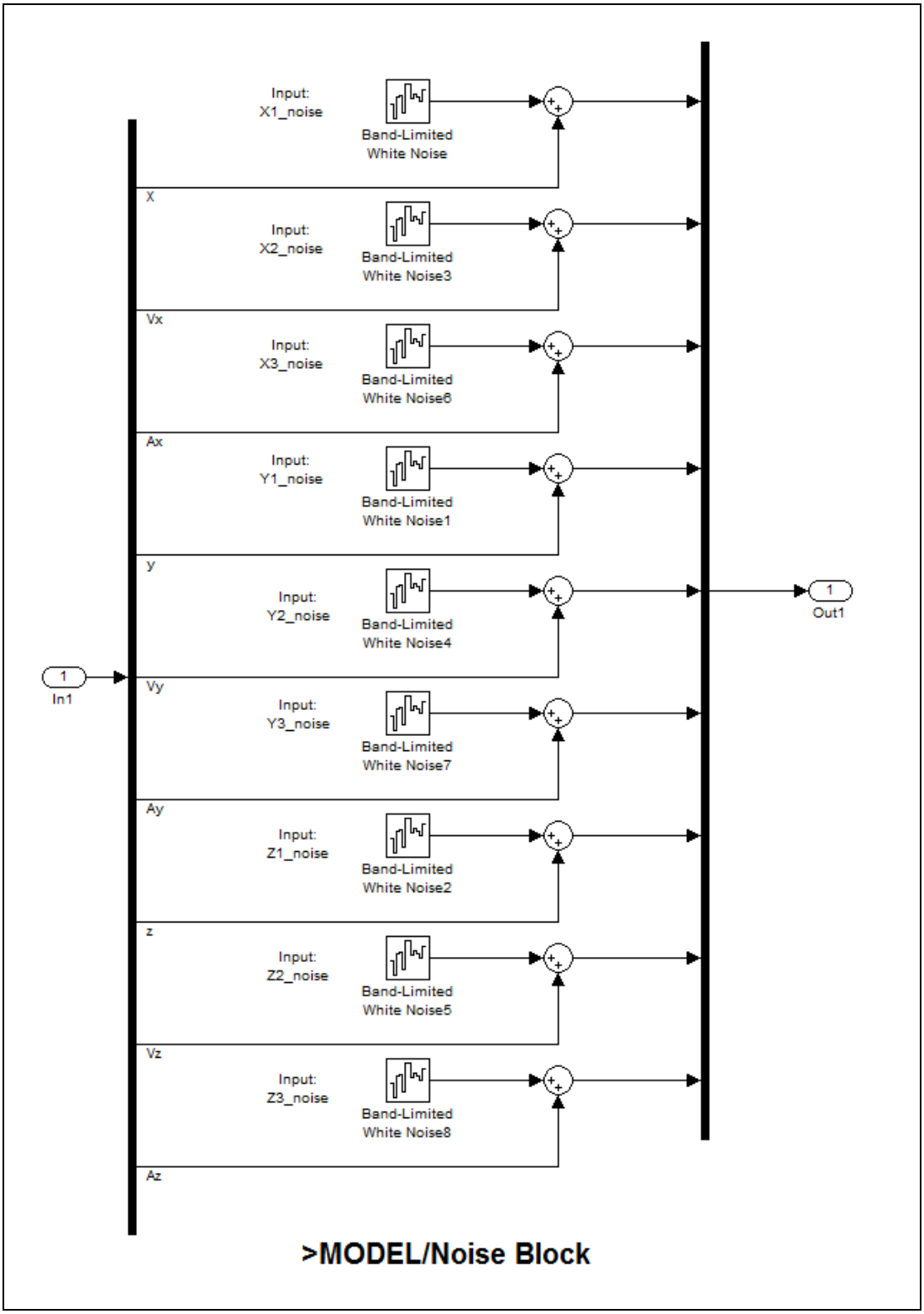
>MODEL/Target_Model



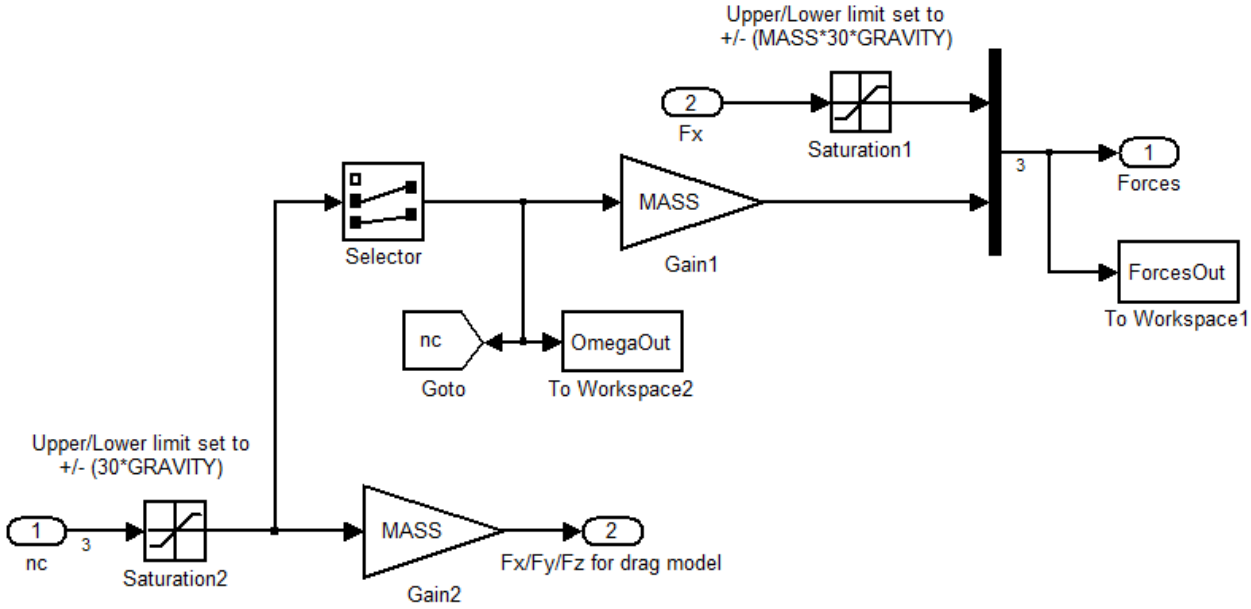
>MODEL/Target_Model/Target_Turn



>MODEL/Tracking Filter

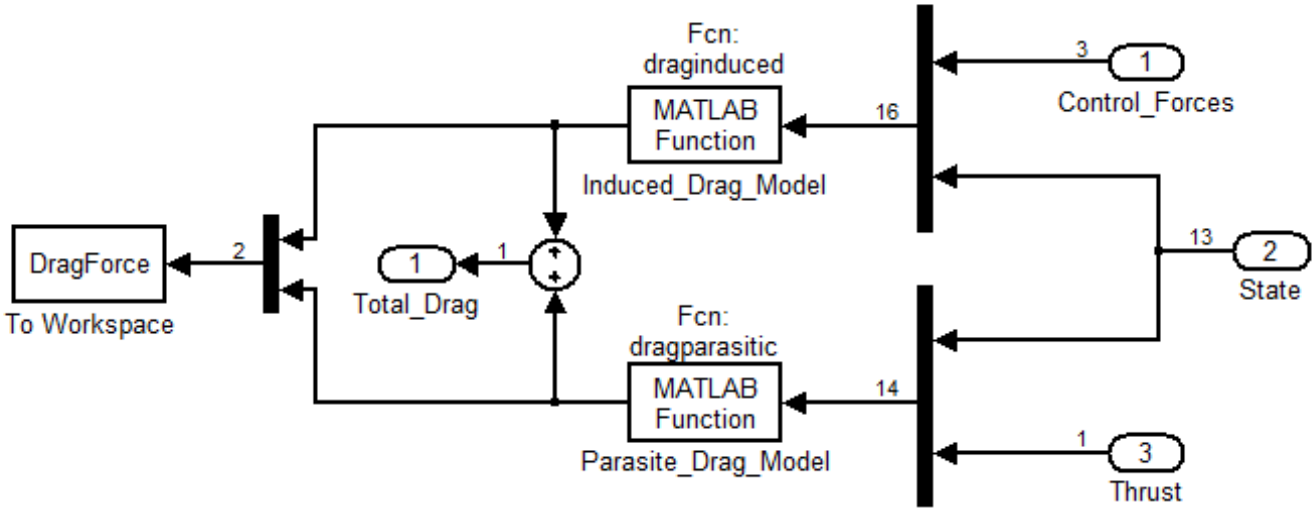


>MODEL/Noise Block



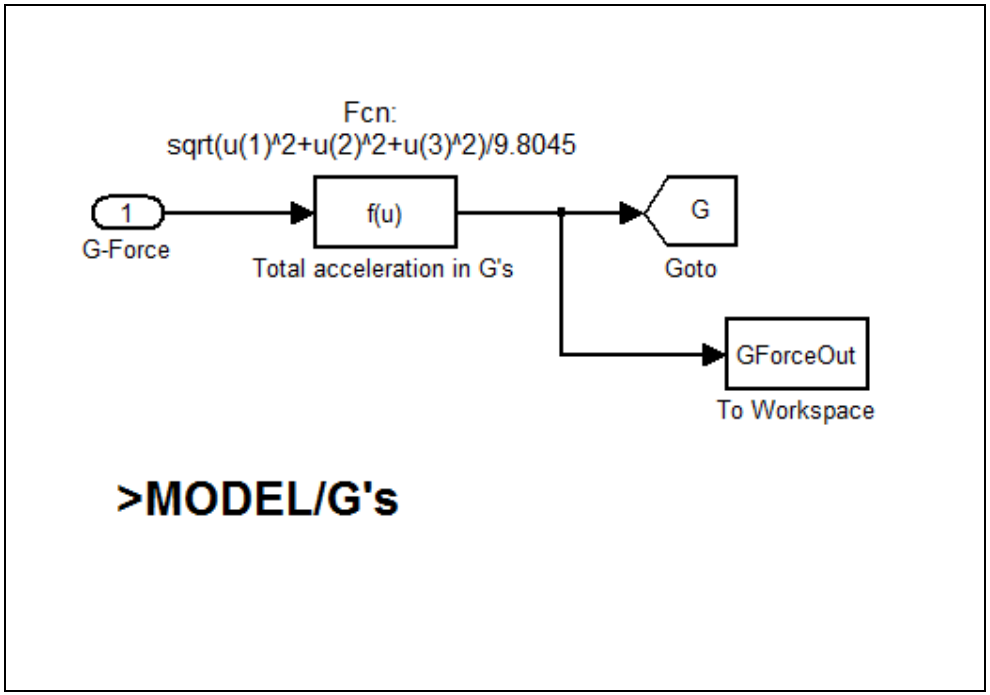
Simplified Aerodynamic Force Generator Converts Commanded Acceleration to Force

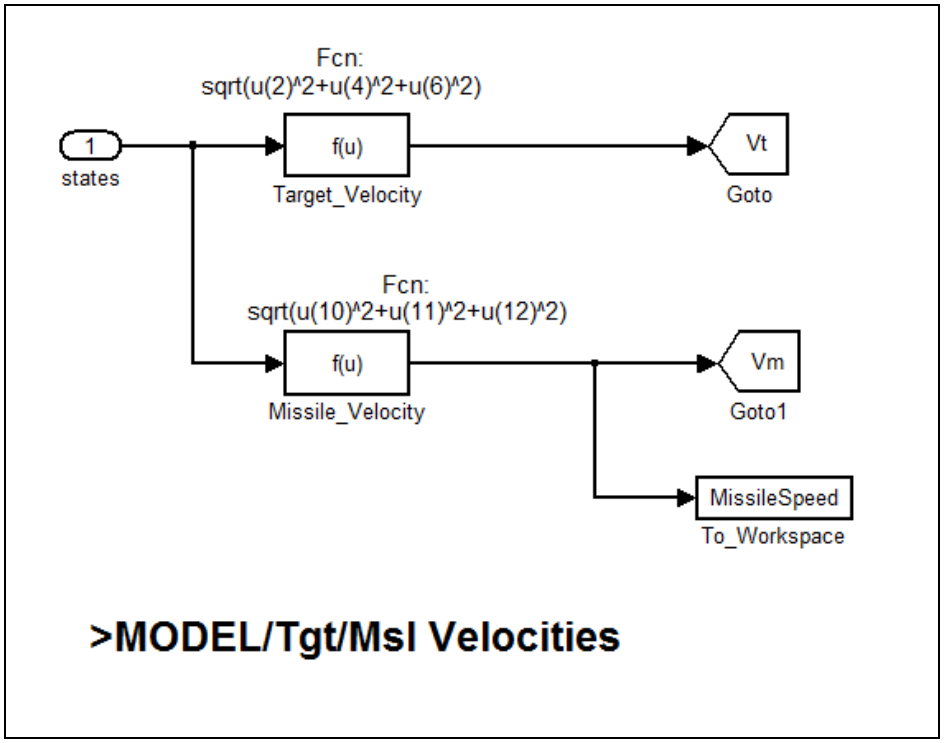
>MODEL/Aerodynamic_Force_Generator

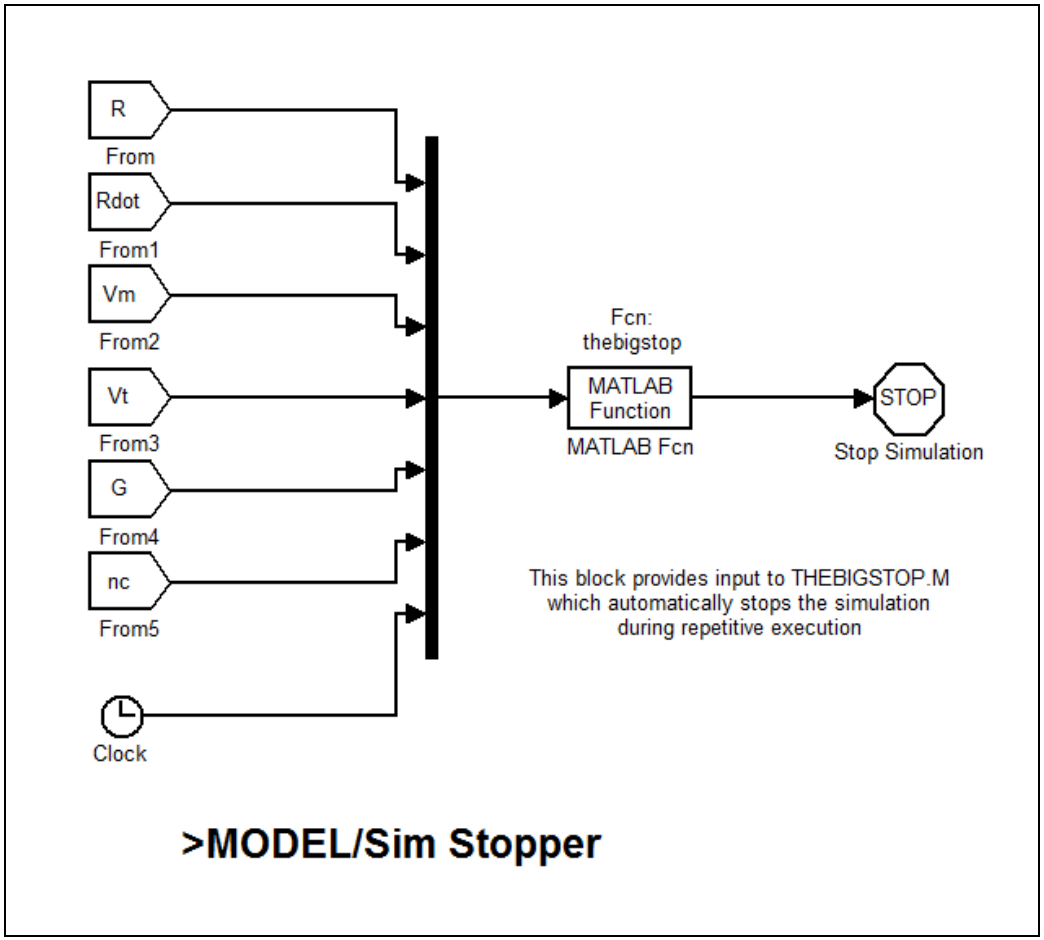


Drag Force Model for Simplified 6DOF Flight Model

>MODEL/Drag_Model


Fcn:
sqrt(u(1)^2+u(2)^2+u(3)^2)/9.8045
1
G-Force
f(u)
Total acceleration in G's
G
Goto
GForceOut
To Workspace
>MODEL/G's


Fcn:
sqrt(u(2)^2+u(4)^2+u(6)^2)
1
states
f(u)
Target_Velocity
Vt
Goto
Fcn:
sqrt(u(10)^2+u(11)^2+u(12)^2)
f(u)
Missile_Velocity
Vm
Goto1
MissileSpeed
To_Workspace
>MODEL/Tgt/Msl Velocities


R
From
Rdot
From1
Vm
From2
Vt
From3
G
From4
nc
From5
Clock
Fcn:
thebigstop
MATLAB
Function
MATLAB Fcn
STOP
Stop Simulation
This block provides input to THEBIGSTOP.M
which automatically stops the simulation
during repetitive execution
>MODEL/Sim Stopper

APPENDIX B. MATLAB® CODE

This appendix contains the list of script files and function codes that were used in the simulation. Table5 summarizes the code names and their purpose.

FILENAME	PURPOSE
A. Simulation run script files	
Kbouter3.m	Runs and computes the kinematic boundary for a particular guidance law
Noise_Study.m	Runs the noise simulation for a 100 repetitions
Single_run.m	Runs a single simulation for a particular scenario
B. Simulation initialization script files	
Missile_data.m	Loads the pre-defined missile constants for the AMRAAM model
Thesis_init.m	Loads the simulation initialization constants and variables
C. Simulation guidance law function files	
chingfanlin.m	Implements APN guidance law
diffgeo.m	Implements DG guidance law
propnavpt.m	Implements PN guidance law
D. Simulation function files	
abgfilter.m	Calls the alpha-beta-gamma filter model
alphabeta.m	Calculates the AOA
cdvmach.m	Polyfits the data curve for C_{di}
draginduced.m	Calculates the induced drag force
dragparasitic.m	Calculates the parasitic drag force
dynamic3D.m	Runs 3D target dynamics
flatearthdyn.m	Runs the flat earth 6DOF dynamics
formdrag.m	Calculates the form drag of the missile
machvalt.m	Calculates the speed of Mach one at particular altitude
q2euler.m	Calculates the euler angles from quarternions
quarternion.m	Calculates the quarternions from euler angles
quat2b.m	Calculates the B rotation matrix from quarternions
rhovalt.m	Calculates the atmospheric density at particular altitude
switchlimit.m	Triggers target turn rate and prevents switch back to zero
tgo.m	Calculates the time to go for the missile to intercept target
thebigstop.m	Stops the simulation run when conditions are met

Table 5. Summary of MATLAB® files and functions.

A. SIMULATION RUN SCRIPT FILES

```
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%   File:           Kbouter3.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Automatically computes a kinematic boundary using 6
%                   DOF simulator with tracking filter.
%                   - Streamlined search loops
%                   - Status indicator
%                   - Saves most recent data to disk
%   Inputs:         none
%   Outputs:        one figure of kinematic boundary
%   Process:        streamlined brute force search algorithm
%   Assumptions:
%   Comments:
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
clear
clc

%------ define globals ------
global SWITCHFLAG TURNFLAG

%------ define constants ------
Thesis_init;
STOPFLAG = 0;       % (1) enable display of simulation stop conditions
                    % (0) disable display of sim stop conditions

%------ define input vector ------
% initialize noise variables
% Set factor to 0 if noiseless simulation is required, else set to any
% positive integer to specify noise level desired. Factor will be
% mulitiplied directly into the power spectral density values of the
% white noise block
factor = 1;
r_noise = 1*factor;                 % Range Noise
rdot_noise = 1e-2*factor;           % Range Rate Noise
theta_noise = 1e-8*factor;          % Horizontal LOS angle
thetadot_noise = 1e-8*factor;       % Horizontal LOS angle rate
phi_noise = 0*factor;               % Vertical LOS angle
phidot_noise = 0*factor;            % Vertical LOS angle rate

% set noise for the abg filter
abgfactor = factor;
X1_noise = 1*abgfactor;             % Target X posn noise
X2_noise = 1e-2*abgfactor;          % Target X velocity noise
X3_noise = 2e-2*abgfactor;          % Target X accel noise
Y1_noise = 1*abgfactor;             % Target Y posn noise
Y2_noise = 1e-2*abgfactor;          % Target Y velocity noise
Y3_noise = 2e-2*abgfactor;          % Target Y accel noise
```

```
Z1_noise = 0*abgfactor;            % Target Z posn noise
Z2_noise = 0*abgfactor;            % Target Z velocity noise
Z3_noise = 0*abgfactor;            % Target Z accel noise

%------ initialize variables ------
TARGET_TURN = 6     % set target turn rate, in g's,
                    % default = 0, max allowable = 9.
MIN_RNG = 10000;    % set min engagement range (10000m default)
DEGSTEP = 5;        % set heading increment

START_TIME = tic;
MaxHit = zeros(1000,5); % Initializing 1000x5 matrix to hold MaxHit data
load CURRENT

%------ functions ------
% Start in tail chase step to head on by <DEGSTEP> degree increments
for HEADING = 0:DEGSTEP:180
    tic
    plotcount = 1;
    runplot = zeros(100,5); % Initialize 100x5 matrix to hold runplot
                            % data
    runplot(:,1) = 10;      % Preload MIN_RNG column with any value
                            % greater than 5 so as to allow for correct
                            % index search later
    rangemax = [0 0];       % Initialize 1x2 matrix to hold rangemax
                            % data
    rangemax(1,1) = MIN_RNG;% rangemax is a 1X2 matrix to store rangemax
                            % and min Range to go info
    disp(['Heading ',num2str(HEADING),' deg'])  % show heading counter

    % compute target speed components
    XSPD = TGT_SPD*cos(HEADING*pi/180);
    YSPD = TGT_SPD*sin(HEADING*pi/180);

    % first range loop step by 10 km
    for TGT_RNG = rangemax(1,1) : 10000 : 150000

        disp(['*** ',num2str(TGT_RNG),', 10km step size***'])
        % set initial target state
        TGT_INIT = [TGT_RNG;XSPD;0;YSPD;-ALT;0];
        TURNFLAG = 0;
        SWITCHFLAG = 0;
        XLAST = [TGT_RNG;0;0;0;0;    % store data for abgfilter.m use
                 0;-ALT;0;0];

        % call simulation
        sim('MODEL')

        % save run data
        disp(['>>> ',num2str(min(RangeToGo)),', min Range<<<'])
        runplot(plotcount,:)=[min(RangeToGo),0,0,0,TGT_RNG];
        % score run
        if (min(RangeToGo)>5)
            break
```

```
    end
    plotcount = plotcount+1;
end

INDEX = find(runplot(:,1)<=5);

if(INDEX)
   rangemax(1,1) = runplot(max(INDEX),5); % Store rangemax data
   rangemax(1,2) = runplot(max(INDEX),1); % Store min range2go data
end

runplot = zeros(100,5);
runplot(:,1) = 10;
plotcount = 1;

% main search loop 1km step size
for TGT_RNG = rangemax(1,1)+1000 : 1000 : rangemax(1,1)+9000

    disp(['*** ',num2str(TGT_RNG),', 1km step size***'])
    % set initial target state
    TGT_INIT = [TGT_RNG;XSPD;0;YSPD;-ALT;0];
    TURNFLAG = 0;
    SWITCHFLAG = 0;
    XLAST = [TGT_RNG;0;0;0;0;    % store data for abgfilter.m use
     0;-ALT;0;0];

    % call simulation
    sim('MODEL')

    % save run data
    disp(['>>> ',num2str(min(RangeToGo)),', min Range<<<'])
    runplot(plotcount,:)=[min(RangeToGo),0,0,0,TGT_RNG];
    % score run
    if (min(RangeToGo)>5)
        break
    end
    plotcount = plotcount+1;
end

INDEX = find(runplot(:,1)<=5);

if(INDEX)
   rangemax(1,1) = runplot(max(INDEX),5); % Store rangemax data
   rangemax(1,2) = runplot(max(INDEX),1); % Store min range2go data
end

runplot = zeros(100,5);
runplot(:,1) = 10;
plotcount = 1;

% main search loop 100m
for TGT_RNG = rangemax(1,1)+100 : 100 : rangemax(1,1)+900
```

```
    disp(['*** ',num2str(TGT_RNG),', 100m step size ***'])
    %set initial target state
    TGT_INIT = [TGT_RNG;XSPD;0;YSPD;-ALT;0];
    TURNFLAG = 0;
    SWITCHFLAG = 0;
    XLAST = [TGT_RNG;0;0;0;0;    % store data for abgfilter.m use
     0;-ALT;0;0];

    % call simulation
    sim('MODEL')

    % save run data
    disp(['>>> ',num2str(min(RangeToGo)),', min Range<<<'])
    runplot(plotcount,:)=[min(RangeToGo),0,0,0,TGT_RNG];
    % score run
    if (min(RangeToGo)>5)
        break
    end
    plotcount = plotcount+1;
end

INDEX = find(runplot(:,1)<=5);

if(INDEX)
   rangemax(1,1) = runplot(max(INDEX),5); % Store rangemax data
   rangemax(1,2) = runplot(max(INDEX),1); % Store min range2go data
end

runplot = zeros(100,5);
runplot(:,1) = 10;
plotcount = 1;

% main search loop 10m. Note, now it is computing the full output
% vector for each run. Starts calculation at rangemax so as to
% determine full output vector for previous rangemax.
for TGT_RNG = rangemax(1,1) : 10 :rangemax(1,1)+90

    disp(['*** ',num2str(TGT_RNG),', 10m step size***'])
    %set initial target state
    TGT_INIT = [TGT_RNG;XSPD;0;YSPD;-ALT;0];
    TURNFLAG = 0;
    SWITCHFLAG = 0;
    XLAST = [TGT_RNG;0;0;0;0;    % store data for abgfilter.m use
     0;-ALT;0;0];

    % call simulation
    sim('MODEL')

        % analyze data from current run
        TOUT = MissileOut(:,14);
        INDX = find(RangeToGo==min(RangeToGo));
        IP = TOUT(INDX(1));
```

```
            % compute cost function J=20*e(tf)^2+integ(u^2)/200 and
            % missile divert
            u2 = (OmegaOut(:,1).^2+OmegaOut(:,2).^2);
            integral = 0;
            for ii=2:INDX
                integral = integral+(TOUT(ii)-TOUT(ii-1))*u2(ii-1);
            end
            J = 20*min(RangeToGo)^2+integral/1000;

        % save run data [miss dist, cost, divert, time, max range]
        disp(['>>> ',num2str(min(RangeToGo)),', min Range<<<'])
        runplot(plotcount,:) = [min(RangeToGo),J,integral,IP,TGT_RNG];

        if (min(RangeToGo)>5)
            break
        end
        plotcount = plotcount+1;
    end

    INDEX = find(runplot(:,1)<=5);

    if(INDEX)
       rangemax(1,1) = runplot(max(INDEX),5); % Store rangemax data
       rangemax(1,2) = runplot(max(INDEX),1); % Store min range2go data
    end

    if (isempty(INDEX))
        MaxHit(HEADING+1,:) = [rangemax(1,2),0,0,0,rangemax(1,1)];
    else
        MaxHit(HEADING+1,:) = runplot(max(INDEX),:);
    end

    % save data to disk
    save CURRENT MaxHit
    toc
    % note for some guidance laws, the down step here
    % must be 2 or more---------------------|
     MIN_RNG = 10000*(floor(rangemax(1,1)/10000)-1);
     if (MIN_RNG <= 10000)
        MIN_RNG = 10000;
     end
    % MIN_RNG = 10000;
end
toc(START_TIME)
END_TIME = toc(START_TIME);

%%
% plot the graph
% 0 deg represents tail chase scenario
% 180 deg represents head on scenario
rho1 = MaxHit(1:DEGSTEP:181,5);
rho1 = [rho1;flipud(rho1)];
theta = 0:DEGSTEP:180;
theta = pi/180*theta;
```

```
theta = [theta,-1*fliplr(theta)]';
figure(5)
polar (theta,rho1)
```

```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           Noise_Study.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Automatically runs 100 simulation runs at a
particular
%                   point with random noise input to determine noise
%                   effects on guidance laws.
%   Inputs:         none
%   Outputs:        plot of abg filter data on TGT posn for the last run
%   Process:
%   Assumptions:
%   Comments:       NOTE: Must switch simulation solver to a fixed step
%                   solver before running this script file.
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
clear
clc

%------ define globals ------
global SWITCHFLAG TURNFLAG

%------ define constants ------
% initialize simulation
Thesis_init;
STOPFLAG = 0;       % (1) enable display of simulation stop conditions
                    % (0) disable display of sim stop conditions

% initialize variables to hold data
holdrange=zeros(1000,1);
holdpos=zeros(1000,3);

%------ define input vector ------
% initialize noise variables
% Set factor to any positive integer to specify noise level desired.
% Factor will be mulitiplied directly into the power spectral density
% values of the white noise block
factor = 4.7
r_noise = 1*factor;                 % Range Noise
rdot_noise = 1e-2*factor;           % Range Rate Noise
theta_noise = 1e-8*factor;          % Horizontal LOS angle
thetadot_noise = 1e-8*factor;       % Horizontal LOS angle rate
phi_noise = 0*factor;               % Vertical LOS angle
phidot_noise = 0*factor;            % Vertical LOS angle rate

% set noise for the abg filter
abgfactor = factor;
X1_noise = 1*abgfactor;             % Target X posn noise
X2_noise = 1e-2*abgfactor;          % Target X velocity noise
X3_noise = 2e-2*abgfactor;          % Target X accel noise
Y1_noise = 1*abgfactor;             % Target Y posn noise
```

```
Y2_noise = 1e-2*abgfactor;          % Target Y velocity noise
Y3_noise = 2e-2*abgfactor;          % Target Y accel noise
Z1_noise = 0*abgfactor;             % Target Z posn noise
Z2_noise = 0*abgfactor;             % Target Z velocity noise
Z3_noise = 0*abgfactor;             % Target Z accel noise

%------ initialize variables ------
% initialize target
TGT_RNG = 79900;
HEADING = 135;
TARGET_TURN = 6;

XSPD = TGT_SPD*cos(HEADING*pi/180);
YSPD = TGT_SPD*sin(HEADING*pi/180);

tic
%------ functions ------
% 100 realizations
for numloops=1:100
   disp(numloops)
   TGT_INIT = [TGT_RNG;XSPD;0;YSPD;-ALT;0];
   TURNFLAG = 0;
   SWITCHFLAG = 0;
   XLAST = [TGT_RNG;0;0;0;0;    % store data for abgfilter.m use. Must
            0;-ALT;0;0];        % be reset to baseline value before
                                % each simulation call
   sim('MODEL')
   % analyze data from current run
   disp(min(RangeToGo))
   holdrange(numloops,:)=min(RangeToGo);
   idx=find(RangeToGo==min(RangeToGo));
   holdpos(numloops,:)=MissileOut(idx,1:3)-TgtOut(idx,1:2:5);
   save NOISY holdrange holdpos
end

% Calculate and display mean miss distance and the standard deviation
missdistance=mean(holdrange)
sigmadistance=std(holdrange)
toc

% Plot target position based on abg filter results for the last run
figure(2)
plot(XLAST_Data(:,1),XLAST_Data(:,4))
title('Tgt Posn based on abg filter')
xlabel('X Posn')
ylabel('Y Posn')
```

```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           Single_run.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Runs a single iteration of a particular engagement
%                   scenario with user specified inputs
%   Inputs:         Target Parameters (Alt, Turn rate, Mach, Hdg, Rng)
%   Outputs:        Min Range (min miss dist)
%                   Graphs of (LOS Range to Go vs Time)
%                             (Engagement Geometry)
%                             (Msl & Tgt Speed vs Time)
%                             (Msl & Tgt Accel (g) vs Time)
%                             (Guidance command output vs Time)
%                             (Msl Forces vs Time)
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
clear;
clc;

%------ define globals ------
global STOPFLAG SWITCHFLAG TURNFLAG

%------ define constants ------
Thesis_init;
STOPFLAG = 0;       % (1) enable display of simulation stop conditions
                    % (0) disable display of sim stop conditions

%------ define input vector ------
% initialize noise variables
% Set factor to 0 if noiseless simulation is required, else set to any
% positive integer to specify noise level desired. Factor will be
% mulitiplied directly into the power spectral density values of the
% white noise block
factor = 0
r_noise = 1*factor;                 % Range Noise
rdot_noise = 1e-2*factor;           % Range Rate Noise
theta_noise = 1e-8*factor;          % Horizontal LOS angle
thetadot_noise = 1e-8*factor;       % Horizontal LOS angle rate
phi_noise = 0*factor;               % Vertical LOS angle
phidot_noise = 0*factor;            % Vertical LOS angle rate

% set noise for the abg filter
abgfactor = factor;
X1_noise = 1*abgfactor;             % Target X posn noise
X2_noise = 1e-2*abgfactor;          % Target X velocity noise
X3_noise = 2e-2*abgfactor;          % Target X accel noise
Y1_noise = 1*abgfactor;             % Target Y posn noise
Y2_noise = 1e-2*abgfactor;          % Target Y velocity noise
Y3_noise = 2e-2*abgfactor;          % Target Y accel noise
```

```
Z1_noise = 0*abgfactor;            % Target Z posn noise
Z2_noise = 0*abgfactor;            % Target Z velocity noise
Z3_noise = 0*abgfactor;            % Target Z accel noise

%------ initialize variables ------

% set target parameters
ALT = 6000;                    % default co-altitude in metres
TARGET_TURN = 6;               % set target turn rate, in g's,
                               % default = 0, max allowable = 9.

% set target speed
TGT_MACH = 0.83;                        % user sets Mach # for target
TGT_SPD = TGT_MACH*machvalt(ALT);       % machine computes speed

% define scenario variables
TGT_HDG = 40;           % heading of 0 represents tail chase,
                        % heading of 180 represents head on geometry
TGT_RNG = 20000;        % set target range

%------ functions ------
tic

disp(['Heading = ',num2str(TGT_HDG),' degrees'])
disp(['Target Range = ',num2str(TGT_RNG/1000),' km'])

% compute target speed components
XSPD = TGT_SPD*cos(TGT_HDG*pi/180);
YSPD = TGT_SPD*sin(TGT_HDG*pi/180);

TGT_INIT = [TGT_RNG;XSPD;0;YSPD;-ALT;0];
TURNFLAG = 0;
SWITCHFLAG = 0;
XLAST = [TGT_RNG;0;0;0;0;    % store data for abgfilter.m use
         0;-ALT;0;0];

sim('MODEL2')
toc

disp(['Min Range ',num2str(min(RangeToGo)),' m'])

%%
%----- plot graphs ------
time = rem(now,1);
hr = floor(time*24);
mins = floor(rem(time*24,1)*60);
timestr = [' ',num2str(hr),':',num2str(mins)];


% Missile to target distance (Range-to-go vs Time)
range = RangeToGo;
t = MissileOut(:,14);
t_disc = 0:FILTSAMP:max(t);
```

```
index = find(range==min(range));
ip = t(index(1));

figure(1)
subplot(2,1,2)
plot(t,range)
title('LOS Range to Go vs Time')
xlabel(['time (seconds)          ',date,timestr])
ylabel('LOS Range (meters)')

% engagement geometry
subplot(2,1,1)
plot(TgtOut(:,1),TgtOut(:,3),':',MissileOut(:,1),MissileOut(:,2))
axis equal
outtext1 = ['time: ',num2str(ip),' seconds'];
outtext2 = ['range: ',num2str(min(range)),' meters'];
text(100,10000,'Intercept at:')
text(100,7000,outtext1)
text(100,4000,outtext2)

title('Engagement Geometry')
xlabel('x (meters)')
ylabel('y (meters)')
legend('Target','Missile','Location','Best')

%%
% Missile and Target Velocities
Target_Spd = sqrt(TgtOut(:,2).^2+TgtOut(:,4).^2+TgtOut(:,6).^2);
figure(2)
plot(t,MissileSpeed,'b-',t,Target_Spd,'k:')
title('Missile/Target Speed vs Time')
ylabel('Speed (m/s)')
xlabel('Time (s)')
legend('Missile','Target','Location','Best')

%%
% Missile & Target Accelerations in g's
gforce = sqrt(AccelOut(:,1).^2+AccelOut(:,2).^2 ...
         +AccelOut(:,3).^2)./9.8045;
figure(3)
subplot(2,1,1)
plot(t,gforce)
title('Missile Accelerations (g) vs Time')

ylabel('Missile Acceleration (g)')
axis([0 round(max(t)) 0 50])
% compute cost function J=20*e(tf)^2+integ(u^2)/200
u2 = (OmegaOut(:,1).^2+OmegaOut(:,2).^2);
integral = 0;
for ii = 2:index
    integral = integral+(t(ii)-t(ii-1))*u2(ii-1);
end
J = 20*min(range)^2+integral/1000;
```

```
outtxt = ['Time (s) / Missile divert:  ',num2str(integral)];
xlabel(outtxt)

Tgtgforce = (sqrt(TargetVelAccel(:,2).^2+TargetVelAccel(:,4).^2+...
            TargetVelAccel(:,6).^2))/9.8045;
subplot(2,1,2)
plot(t,Tgtgforce)
title('Target Accelerations (g) vs Time')
xlabel('Time (s)')
ylabel('Target Acceleration (g)')

%%
% guidance command
figure(4)
plot(t,OmegaOut(:,1),t,OmegaOut(:,2),':')
title('Guidance law command output vs Time')
outtxt = ['Cost J: ',num2str(J),'   time (seconds)'];
xlabel([outtxt,'      ',date,timestr])
ylabel('n_c (m/sec^2)')
axis([0 round(max(t)) -200 50])
legend('n_c y','n_c z','Location','Best')

%%
% Missile Force vs Time
figure(5)
plot(t,ForcesOut(:,1),'b-',t,ForcesOut(:,2),'k:')
title('Missile Forces Fx/Fy vs Time')
ylabel('Force')
xlabel('Time (s)')
legend('Fx','Fy','Location','Best')
```

B. SIMULATION INITIALIZATION SCRIPT FILES

```
%% AMRAAM %%
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           Missile_data.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Missile data for AMRAAM. Establishes missile
%                   dimensions for use in computing aerodynamic forces
%                   and moments. Except where noted, all dimensions in
%                         MKS system.
%   Inputs:
%   Outputs:        various
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

% Missile Name: PSEUDO AMRAAM

%------ define globals ------
global MASS DIAM LENGTH XCG XCPN XCPB XHL
global ST SW SPLAN SREF         % Areas
global JX JY JZ                 % Rotational Inertia

%------ define constants ------
%--- missile body dimensions ------------------------------
MASS = 156.8;           % mass, may be time varying
DIAM = 0.1778;          % diameter
LENGTH = 3.657;         % length
XCG = 1.8288;           % initial c.g., may be time varying
LN = 0.6769;            % length of nose cone
%--- missile tailplane dimensions -------------------------
XHL = 3.454;            % hinge line arm
CRT = 0.4061;           % tail root chord
CTT = 0.0676;           % tail tip chord
TXT = 0.0676;           % tail extension
HT = 0.2286;            % tail height
%--- missile wing dimensions ------------------------------
XW = 1.134;             % wing to radome tangency point
CRW = 0.3554;           % wing root chord
CTW = 0;                % wing tip chord
WXT = 0;                % wing extension
HW = 0.1778;            % wing height

%------ define input vector ------

%------ initialize variables ------
```

```
%------ functions ------
%--- Centers of pressure ----------------------------------
XCPN = 0.67*LN;                                      % nose CP
XCPW = LN+XW+0.7*CRW-0.2*CTW;                        % wing CP
AN = 0.67*LN*DIAM;                                   % plan area of nose
AB = (LENGTH-LN)*DIAM;                               % plan area of body
XCPB = (0.67*AN*LN+AB*(LN+0.5*(LENGTH-LN)))/(AN+AB); % body CP
%--- Area computations ------------------------------------
SW = 0.5*HW*(CTW+CRW)+CRW*WXT;              % wing area
ST = 0.5*HT*(CTT+CRT)+CRT*TXT;              % tail area
SPLAN = (LENGTH-LN)*DIAM+0.67*LENGTH*DIAM;  % body and nose plan area
SREF = pi*DIAM^2/4;                         % missile cross sectional
area
%--- Computing the inertial matrix ------------------------
JX = MASS*((DIAM/2)^2)/2;
JY = MASS*((LENGTH^2)/12+((DIAM/2)^2)/4)+MASS*((LENGTH/2)-XCG)^2;
JZ = JY;
```

```
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%   File:           Thesis_init.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    This script file initializes thesis work missile
%                   simulation
%   Inputs:
%   Outputs:
%   Process:
%   Assumptions:
%   Comments:
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%------ define globals ------
global STOPFLAG SATFLAG SWITCHFLAG TURNFLAG
global TARGET_TURN XLAST FILTSAMP

%------ define constants ------
% physical constants
OMEGA_X = 7.292115e-5;          % earth's rotation rate
GM_E = 3.9860014e14;            % G*mass of earth
R_E = 6.378164e6;               % radius of earth
F = 1/298.257;                  % ellipsoidal squash factor
OMEGA_E = [OMEGA_X;0;0];        % earth's rotational velocity vector
GRAVITY = 9.8045;               % gravitational acceleration constant

%------ define input vector ------
% DEFAULT INPUTS REQUIRED (Data defined here can be superseded by the
%                          data within the individual simulation script
%                          files)
ALT = 6000;                     % set default co-altitude in metres
TARGET_TURN = 0;                % set target turn rate, in g's,
                                % default = 0, max allowable = 9.
                                % Simulation applies tgt turn rate 8 s
                                % prior to impact
TGT_MACH = 0.83;                % user sets Mach # for target
TGT_SPD = TGT_MACH*machvalt(ALT);
MSL_MACH = 0.83;                % user sers Mach # for missile at launch
MSL_SPD = MSL_MACH*machvalt(ALT);
TGT_HDG = 70;                   % target heading in degrees. 0
                                % represents tail chase and 180
                                % represents head on geometry
TGT_RNG = 40000;                % target distance from initial launch
                                % point

% REQUIRED GLOBAL VALUES
STOPFLAG = 0;                 % (1) enable display of simulation stop
                              % conditions
TURNFLAG = 0;                 % store turn rate data, always set to (0)
SWITCHFLAG = 0;               % prevent turn rate from switching back to
                              % zero, always set to (0)
```

```
TMAX = 200;                  % set simulation max run time

XLAST = [TGT_RNG;0;0;0;0;    % store data for abgfilter.m use
         0;-ALT;0;0];

FILTSAMP = 0.01;             % set filter sample interval for
                             % abgfliter.m use. Also sets the Sampling
                             % time for the white noise blocks as well
                             % as the step size interval when selecting
                             % fixed step size simulation solver. Do not
                             % set larger than 0.02 for diffgeo guidance
                             % law.

%------ initialize variables ------
% missile physical parameters
Missile_data;

VB = [MSL_SPD;0;0];          % Missile initial velocity vector
POSN = [0;0;-ALT];           % initial missile position vector, note
                             % altitude is negative in NED coord

% compute Euler angles for missile
PSI = 0*pi/180;              % Varying PSI will determine the angle at
                             % which the missile is pointing at the
                             % target. 0 deg will be pointing straight
                             % at the target.
THETA = 0*pi/180;
PHI = 0*pi/180;

%------ functions ------
Q_0 = quarternion(PHI,THETA,PSI);
Q_0 = Q_0/sqrt(Q_0(1)^2+Q_0(2)^2+Q_0(3)^2+Q_0(4)^2);

B = quat2b(Q_0);

P = 0*pi/180;
Q = 0*pi/180;
R = 0*pi/180;

OMEGA_B = [P;Q;R];

% missile initial state vector
MSL_INIT = [POSN;VB;OMEGA_B;Q_0];

% compute target speed components
XSPD = TGT_SPD*cos(TGT_HDG*pi/180);
YSPD = TGT_SPD*sin(TGT_HDG*pi/180);

% target initial state vector [x;x_dot;y;y_dot;z;z_dot]
TGT_INIT = [TGT_RNG;XSPD;0;YSPD;-ALT;0];
```

C. SIMULATION GUIDANCE LAW FUNCTION FILES

```
function [ y ] = chingfanlin ( u )
% CHINGFANLIN
% Computes the optimal guidance law derived by Ching Fan Lin pg 475 with
% drag force inputs for point mass simulation

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           chingfanlin.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    computes APN guidance law from Ching Fan Lin
%   Inputs:         seeker output, filter output, accelerometer, missile
%                   timer
%   Outputs:        command accelerations, y & z forces for drag
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------
global MASS SATFLAG

%------ define constants ------
Nprime = 5;
Nprimez = 5;

%------ define input vector ------
thetadot =  u(1);
phidot =    u(2);
los =       u(3);
philos =    u(4);
Vc =       -u(5);
R =         u(6);
heading =   u(7);
Vm =        u(8);
Vmdot =     u(9);
phi =       u(10);
theta =     u(11);
psi =       u(12);

tgt_state = u(13:21);
time =      u(22);

accel_in =  u(23:25);

%------ initialize variables ------

%------ functions ------
```

```
if (Vc==0)
    tgo = 1e6;
else
    tgo = R/Vc;
end

% Compute relative state estimate
xhat = [R*cos(los);
        R*sin(los);
        R*sin(philos);
        tgt_state(2)-Vm*cos(psi);
        tgt_state(5)-Vm*sin(psi);
        tgt_state(8)-Vm*sin(theta);
        tgt_state(3);
        tgt_state(6);
        tgt_state(9);
        accel_in(1);
        accel_in(2);
        accel_in(3)];
if time<2.0
    ny = Nprime*Vc*(thetadot)/cos(heading-los);
    nz = Nprimez*Vc*(phidot)-9.8045;
else
    uc = (5/tgo^2)*[eye(3),tgo*eye(3), tgo^2/2*eye(3), zeros(3)]*xhat;
    ny = uc(2);
    nz = uc(3) - 9.8045;
end

% Define acceleration in x-axis as 0 as it will be determined from the
% thrust and drag model blocks
nx = 0;

% Saturation of forces at 30 g's will be done in the Aerodynamic Force
% Generator Block
y = [nx;ny;nz];

end
```

```
function [ y ] = diffgeo ( u )
% DIFFGEO
% Computes the differential geometric guidance law derived by Chaoyong
Li

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           diffgeo.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    computes Differential Geometry guidance law from
%                   Chaoyong Li
%   Inputs:         seeker output, filter output, accelerometer, missile
%                   timer
%   Outputs:        command accelerations, y & z forces for drag
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------
global MASS SATFLAG

%------ define constants ------
Nprime = 5;
Nprimez = 5;

%------ define input vector ------
thetadot =  u(1);
phidot =    u(2);
los =       u(3);
philos =    u(4);
Vc =       -u(5);
R =         u(6);
heading =   u(7);
Vm =        u(8);
Vmdot =     u(9);
phi =       u(10);
theta =     u(11);
psi =       u(12);

tgt_state = u(13:21);
time =      u(22);

accel_in =  u(23:25);

%------ initialize variables ------

%------ functions ------
tgt_accel = [u(15);u(18)];          % Target acceleration vector in X-Y
                                    % plane
```

```
eta_t = atan2(u(17),u(14)) - los;   % Angle between velocity vector of
                                    % target and the los vector in
                                    % horizontal plane
eta_tz = atan2(u(21),norm(tgt_accel)); % angle eta in vertical plane
eta_m = psi - los;                  % Look angle of the missile


% Diff Geometric guidance law
ny = norm(tgt_accel)*cos(eta_t)/cos(eta_m) +...
     Nprime*Vc*(thetadot)/cos(eta_m);
% vertical acceleration must account for gravity
nz = u(21)*cos(eta_tz)/cos(theta-philos) +...
     Nprimez*Vc*(phidot)/cos(theta-philos) - 9.8045;

% Define acceleration in x-axis as 0 as it will be determined from the
% thrust and drag model blocks
nx = 0;

% Saturation of forces at 30 g's will be done in the Aerodynamic Force
% Generator Block
y = [nx;ny;nz];

end
```

```
function [ y ] = propnavpt ( u )
% PROPNAVPT
% Computes the exact proportional navigation with drag force inputs for
% point mass simulation

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           propnavpt.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Prop nav guidance law for 6 DOF flight model.
Computes
%                   the applied forces for use by induced drag model.
%                   Required to eliminate algebraic loops in the
simulation
%   Inputs:         [seeker data, IMU data, timer]
%   Outputs:        [command accelerations. applied forces]
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------
global MASS SATFLAG

%------ define constants ------
Nprime = 5;
Nprimez = 5;

%------ define input vector ------
thetadot =  u(1);
phidot =    u(2);
los =       u(3);
philos =    u(4);
Vc =       -u(5);
theta =     u(11);
psi =       u(12);


%------ initialize variables ------

%------ functions ------
% classic PN guidance law
ny = Nprime*Vc*(thetadot)/cos(psi-los);
% vertical acceleration must account for gravity
nz = Nprimez*Vc*(phidot)/cos(theta-philos)-9.8045;

% Define acceleration in x-axis as 0 as it will be determined from the
% thrust and drag model blocks
nx = 0;
```

```
% Saturation of forces at 30 g's will be done in the Aerodynamic Force
% Generator Block
y = [nx;ny;nz];

end
```

D. SIMULATION FUNCTION FILES

```
function [ y ] = abgfilter ( u )
% ABGFILTER
% Implements an alpha-beta-gamma filter as outlined in Bar-Shalom & Li
% "Estimation and Tracking"

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           abgfilter.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Implements a 9-dimensional state vector
%                   alpha-beta-gamma tracking filter for use with
%                   missile guidance laws requiring tracking filters
%                   (Note: Uses global XLAST to preserve state between
%                   iterations)
%   Inputs:         measurements (los,los_dot,R,R_dot),
%                   missile posn (x,y,z)
%   Outputs:        9-dimensional estimate of target state
%                   (x,vx,ax,y,vy,ay,z,vz,az)
%   Assumptions:
%   Comments:       May require up to 20 samples to stabilize from
%                   initialization
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------
global FILTSAMP XLAST

%------ define constants ------

%------ define input vector ------
losdot = u(1);
phidot = u(2);
los = u(3);
phi = u(4);
rdot = u(5);
R = u(6);
xm = u(7);
ym = u(8);
zm = u(9);

%------ initialize variables ------
% compute target cartesian coordinates
xt = R*cos(los)+xm;
yt = R*sin(los)+ym;
zt = R*sin(phi)+zm;

z = [xt;yt;zt];
```

```
% set noise parameters
sigmav = 1;
sigmaw = 1;
lamda = sigmav*(FILTSAMP^2)/sigmaw;

% set filter parameters from Bar-Shalom & Li (Assumed Numbers)
falpha = .9;
fbeta = .9;
fgamma = .9;

% filter matrices
F = [1 FILTSAMP FILTSAMP^2/2 zeros(1,6);
     0 1 FILTSAMP zeros(1,6);
     0 0 1 zeros(1,6);
     0 0 0 1 FILTSAMP FILTSAMP^2/2 zeros(1,3);
     0 0 0 0 1 FILTSAMP zeros(1,3);
     0 0 0 0 0 1 zeros(1,3);
     0 0 0 0 0 0 1 FILTSAMP FILTSAMP^2/2;
     0 0 0 0 0 0 0 1 FILTSAMP;
     0 0 0 0 0 0 0 0 1];

 H = [1 0 0 0 0 0 0 0 0;
      0 0 0 1 0 0 0 0 0;
      0 0 0 0 0 0 1 0 0];

 % compute steady state gains
 W = [falpha;fbeta/FILTSAMP;fgamma/(2*FILTSAMP^2)];

 % build gain matrix
 P = [W zeros(3,2);
     zeros(3,1) W zeros(3,1);
     zeros(3,2) W];

 %------ functions ------
 % run filter
 xhat = F*XLAST;
 xhat1 = xhat + P*(z-H*xhat);

 XLAST = xhat1;

 y = xhat1;

end
```

```
function [ y ] = alphabeta( u )
% ALPHABETA
% Computes angles of attack in both vertical and horizontal planes

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           alphabeta.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes angles of attack using ATAN formulation in
%                   Bryson "Control of Spacecraft and Aircraft"
%   Inputs:         Missile state
%   Outputs:        Angles of attack (alpha, beta)
%   Process:        ATAN formulation of Bryson
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------
v = [u(4);u(5);u(6)];

%------ initialize variables ------

%------ functions ------
% (Equations developed in Bryson)
% using beta1 for sideslip angle to avoid problems with built in matlab
% function 'beta'

alpha = atan2(v(3),sqrt(v(1)^2+v(2)^2));
beta1 = atan2(v(2),v(1));

y = [alpha;beta1];

end
```

```
function [ y ] = cdvmach (mach, boost)
% CDVMACH
% Computes approximation of zero lift drag coefficient vs. mach number

%------------------------------------------------------------------------
%------------------------------------------------------------------------
%   File:           cdvmach.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes polynomial fit for cd0 vs Mach number
%   Inputs:         mach # and boost status
%   Outputs:        cd0
%   Process:        Fit on data from Hutchins EC4330 notes
%   Assumptions:
%   Comments:
%------------------------------------------------------------------------
%------------------------------------------------------------------------
%------ define globals ------
%------ define constants ------
NoBoost = [-0.0014 0.0299 -0.2110 0.6256];
Boost = [-0.0012 0.0243 -0.1521 0.4044];
%------ define input vector ------
%------ initialize variables ------
%------ functions ------
if (boost & (mach<0.7))
    y=0.15;
end
if (~boost & (mach<0.7))
    y=0.25;
end
if (boost & (mach>=0.7) & (mach<1.2))
    y=(mach-0.7)*0.2 + 0.15;
end
if (~boost & (mach>=0.7) & (mach<1.2))
    y=(mach-0.7)*0.3 + 0.25;
end
if ((mach>=1.2) & (boost~=0))
    y=polyval(Boost, mach);
end
if ((mach>=1.2) & (boost==0))
    y=polyval(NoBoost, mach);
end
if ((mach>5 & boost))
    y=0.10;
end
if ((mach>6.4) & ~boost)
    y=0.132;
end
end
```

```
function [ y ] =  draginduced( u )
% DRAGINDUCED
% Computes the induced aerodynamic drag force

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           draginduced.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    computes induced drag for simplified 6DOF
%   Inputs:         force output of guidance law, state
%   Outputs:        drag force
%   Process:        work backwards to CN from forces
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------
global MASS SREF

%------ define constants ------
eAR = 1.5;            % elliptical eff & AR

%------ define input vector ------
Fy = u(2);
Fz = u(3);
v2 = u(7)^2+u(8)^2+u(9)^2;          % missile velocity
alt = u(6);                         % missile alt

%------ initialize variables ------
rho = rhovalt(abs(alt));            % atmospheric density
mach = sqrt(v2)/machvalt(alt);
Q = rho*v2/2;                       % dynamic pressure

%------ functions ------
if (Q==0)
    Cny = 0;
    Cnz = 0;
else
    Cny = Fy/(Q*SREF);              % y normal coefficient
    Cnz = Fz/(Q*SREF);              % z normal coefficient
end
Cdi = (Cny^2+Cnz^2)/(pi*eAR);       % induced drag coefficient

if (mach<1)
    Cdi = 0.25*sqrt(Fy^2+Fz^2)/(MASS*9.8045);  % subsonic drag equal to
                                               % max Cd0*applied G force
end
y = Cdi*Q*SREF;                     % drag force

end
```

```
function [ y ] = dragparasitic ( u )
% DRAGPARASITIC
% Computes parasitic aerodynamic drag force

%------------------------------------------------------------------------
%------------------------------------------------------------------------
%   File:           dragparasitic.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes parasitic drag after breaking apart state
%                   vector
%   Inputs:         state vector, boost status
%   Outputs:        parasitic drag force
%   Process:
%   Assumptions:
%   Comments:
%------------------------------------------------------------------------
%------------------------------------------------------------------------

%------ define globals ------
global  SREF

%------ define constants ------

%------ define input vector ------
vel2 = u(4)^2+u(5)^2+u(6)^2;
alt = u(3);

boost = u(14);

%------ initialize variables ------

%------ functions ------
y = formdrag(SREF, alt, vel2, boost);

end
```

```
function [ y ] = dynamic3D ( u )
% DYNAMIC3D
% Computes the motion dynamics for a body (target) in three dimensions

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           dynamic3d.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    target motion dynamics
%   Inputs:         target state, turn rate input
%   Outputs:        derivative of target state
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------
Ac = u(1);              % Centripetal Acceleration
x = u(2);
xdot = u(3);
y = u(4);
ydot = u(5);
z = u(6);
zdot = u(7);

%------ initialize variables ------

%------ functions ------
TgtSpd = sqrt(xdot^2+ydot^2+zdot^2);
omega = Ac/TgtSpd;                  % where Ac = TgtSpd^2 / r
                                    % & omega = TgtSpd / r


y = [xdot;
     -omega*ydot;
     ydot;
     omega*xdot;
     zdot;
     0];

end
```

```
function [ y ] = flatearthdyn ( u )
% FLATEARTHDYN
% Computes motion dynamics for 6 DOF flat earth model

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%   File:           flatearthdyn.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes 6 DOF dynamics for flat earth using
%                   quarternion formulation
%   Inputs:
%   Outputs:        derivative of state vector
%   Process:        Steven & Lewis
%   Assumptions:
%   Comments:
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------
p = [u(1);u(2);u(3)];
v_b = [u(4);u(5);u(6)];
omega_b = [u(7);u(8);u(9)];
P = u(7); Q = u(8); R = u(9);

q = [u(10);u(11);u(12);u(13)];

magq = sqrt(q(1)^2+q(2)^2+q(3)^2+q(4)^2);
q = q/magq;

x = [p;v_b;omega_b;q];

J = [u(14)   0     0;              % inertial matrix
       0   u(15)   0;
       0     0   u(16)];

F_B = [u(17);u(18);u(19)];         % Forces

T_B = [u(20);u(21);u(22)];         % Torques

g = [0; 0; 9.8045];                % Not using external gravity model

m = u(23);                         % Mass

%------ initialize variables ------
% Compute rotation matrices
B = quat2b(q);
```

```
OMEGA_B = [0 -R  Q;
           R  0 -P;
          -Q  P  0];

OMEGA_q = [0  P  Q  R;
          -P  0 -R  Q;
          -Q  R  0 -P;
          -R -Q  P  0];

%------ functions ------
y = [zeros(3)         B'         zeros(3)            zeros(3,4);
     zeros(3)    -OMEGA_B        zeros(3)            zeros(3,4);
     zeros(3)     zeros(3)    -1*inv(J)*OMEGA_B*J    zeros(3,4);
     zeros(4,3)   zeros(4,3)     zeros(4,3)       -(1/2)*OMEGA_q];

y = y*x;

y = y+[zeros(3,1);
       B*g+(1/m)*F_B;
       inv(J)*T_B;
       zeros(4,1)];

end
```

```
function [ y ] = formdrag (A, alt, vel2, boost )
% FORMDRAG
% Computes form drag for a missile with frontal area A in a standard
% atmosphere. Uses MACHVALT, CDVMACH, RHOVALT

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           formdrag.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes the form drag for a missile with frontal
%                   area A in a standard atmosphere
%   Inputs:         area, altitude. V^2, boost on/off
%   Outputs:        parasitic drag force
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------

%------ initialize variables ------
rho = rhovalt(alt);
mach = (vel2)^(1/2)/machvalt(alt);

%------ functions ------
if (mach>100)
    mach = 0.83;
end

Cd = cdvmach(mach,boost);

y = rho*vel2*Cd*A/2;

end
```

```
function [ y ] = machvalt ( alt )
% MACHVALT
% Computes the linear approximation for a given altitude in meters/sec
% based on  standard ICAO atmosphere

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           machvalt.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes the linear approximation to Mach 1 for
%                   standard ICAO atmosphere
%   Inputs:         altitude
%   Outputs:        Mach 1
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------
Mach1 = [-0.0041 340.3];
Mach2 = 295.1;
Mach3 = [0.00067 281.7];

%------ define input vector ------

%------ initialize variables ------
alt = abs(alt);             % account for NED coords

%------ functions ------
if (alt<11000)
    y = polyval(Mach1,alt);
else
    if (alt>20000)
        y = polyval(Mach3,alt);
    else
        y = Mach2;
    end
end

end
```

```
function [ y ] =  q2euler( u )
% Q2EULER
% Computes the Euler angles from quarternions

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           q2euler.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes the euler angles from the quarternions
%   Inputs:         Quarternions
%   Outputs:        Euler Angles
%   Process:        Kuiper
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------
q0 = u(1);
q1 = u(2);
q2 = u(3);
q3 = u(4);

%------ initialize variables ------

%------ functions ------
% convert quarternions to euler angles
m11 = 2*q0^2+2*q1^2-1;
m12 = 2*q1*q2+2*q0*q3;
m13 = 2*q1*q3-2*q0*q2;
m23 = 2*q2*q3+2*q0*q1;
m33 = 2*q0^2+2*q3^2-1;

psi = atan2(m12,m11);
theta = asin(-m13);
% correct for singularity in pitch
if (isreal(theta))
    theta = theta;
else
    theta = sign(-m13)*pi/2;
end

phi = atan2(m23,m33);
y = [phi, theta, psi];

end
```

```
function [ y ] = quarternion ( phi,theta,psi )
% QUARTERNION
% Computes the quarternions from Euler angles in radians

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           quarternion.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes the quarternions from euler angles
%   Inputs:         Euler angles in radians
%   Outputs:        Quarternions
%   Process:        Kuiper
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------

%------ initialize variables ------

%------ functions ------
% Quarternion equations
q0 = cos(phi/2)*cos(theta/2)*cos(psi/2)...
        +sin(phi/2)*sin(theta/2)*sin(psi/2);
q1 = sin(phi/2)*cos(theta/2)*cos(psi/2)...
        -cos(phi/2)*sin(theta/2)*sin(psi/2);
q2 = cos(phi/2)*sin(theta/2)*cos(psi/2)...
        +sin(phi/2)*cos(theta/2)*sin(psi/2);
q3 = cos(phi/2)*cos(theta/2)*sin(psi/2)...
        -sin(phi/2)*sin(theta/2)*cos(psi/2);

y = [q0; q1; q2; q3];

end
```

```
function [ y ] = quat2b ( u )
% QUAT2B
% Computes rotation matrix from quarternions

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           quat2b.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes rotation matrix from quarternions
%   Inputs:         Quarternion
%   Outputs:        Rotation Matrix B
%   Process:        Kuiper
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------
q0 = u(1);
q1 = u(2);
q2 = u(3);
q3 = u(4);

%------ initialize variables ------

%------ functions ------
y = [q0^2+q1^2-q2^2-q3^2    2*(q1*q2+q0*q3)      2*(q1*q3-q0*q2);
       2*(q1*q2-q0*q3)    q0^2-q1^2+q2^2-q3^2    2*(q2*q3+q0*q1);
       2*(q1*q3+q0*q2)       2*(q2*q3-q0*q1)    q0^2-q1^2-q2^2+q3^2];

end
```

```
function [ y ] = rhovalt ( alt )
% RHOVALT
% Computes the atmospheric density vs altitude for ICAO standard
atmosphere

%------------------------------------------------------------------------
%------------------------------------------------------------------------
%   File:           rhovalt.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes the atmospheric density from ICAO standard
%                   atmosphere. Exponential model
%   Inputs:         Altitude
%   Outputs:        rho
%   Process:
%   Assumptions:
%   Comments:
%------------------------------------------------------------------------
%------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------

%------ initialize variables ------
alt = abs(alt);                 % account for NED coord

%------ functions ------
if alt>9144
    y = 1.75228763*exp(-alt/6705.6);
else
    y = 1.22557*exp(-alt/9144);
end

end
```

```
function [ y ] = switchlimit ( u )
% SWITCHLIMIT
% Governs the output of the switch in activating target turn maneuvers.

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%   File:           switchlimit.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    In certain scenarios, after Time-to-Go falls below
%                   specified threshold and the switch activates to
%                   start Target evasive maneuvers, the Time-to-Go may
%                   increase back above the specified threshold. When
%                   this happens, the switch no longer outputs the
%                   target turn rate and the target stops turning into
%                   the missile for evasive maneuvers until Time-to-Go
%                   falls below threshold again.
%
%                   Therefore, this function is implemented to output
%                   the turn rate value from the time where the Time-to-
%                   Go falls below the specified threshold for the first
%                   time and keeps outputting the turn rate value
%                   regardless if the Time-to-Go subsequently increases
%                   above threshold value until the end of the
%                   simulation.
%
%                   NOTE: the global variables SWITCHFLAG and TURNFLAG
%                   must always be reset to 0 prior to calling for the
%                   next model simulation.
%   Inputs:         Target Turn Rate
%   Outputs:        Target Turn Rate
%   Process:
%   Assumptions:
%   Comments:
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%------ define globals ------
global SWITCHFLAG TURNFLAG
%------ define constants ------
%------ define input vector ------
%------ initialize variables ------
%------ functions ------

if ((u==0) & (SWITCHFLAG==0))
    y=u;
else if (u~=0 & (SWITCHFLAG==0))
    SWITCHFLAG = 1;
    y = u;
    TURNFLAG = u;
    else if (u~=0 & (SWITCHFLAG==1))
        y=u;
        else if (u==0 &(SWITCHFLAG==1))
```

```
            y = TURNFLAG;
            end
        end
    end
end

end
```

```
function [ y ] = tgo ( u )
% TGO
% Computes time to go from Range and Range Rate

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   File:           tgo.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Computes Time-to-Go (tgo)
%   Inputs:         range, range rate
%   Outputs:        tgo
%   Process:
%   Assumptions:
%   Comments:
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

%------ define globals ------

%------ define constants ------

%------ define input vector ------
range = u(2);
rate = u(1);

%------ initialize variables ------

%------ functions ------
if (rate==0)
    y = 100;
else
    y = abs(range/rate);
end

end
```

```
function [ y ] = thebigstop ( u )
% THEBIGSTOP
% Consolidated simulation stop function

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%   File:           thebigstop.m
%   Name:           CPT Daniel Perh
%   Compiler:       MatLab v7.11.0.584 (R2010b)
%                   32-bit (win 32)
%   Date:           08 July 2011
%   Description:    Stops simulation under a variety of conditions
%   Inputs:         see below
%   Outputs:        stop flag
%   Process:
%   Assumptions:
%   Comments:
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%------ define globals ------
global STOPFLAG

%------ define constants ------

%------ define input vector ------
R = u(1);
Rdot = u(2);
Vm = u(3);
Vt = u(4);
G = u(5);
Ny = u(6);
Nz = u(7);
time = u(8);

%------ initialize variables ------
stop = [];
y = 0;

%------ functions ------

    if ((time>2.0)&(Vm<Vt))
        y = 111;
        stop = 'V stop';
    end

    if ((time>2.0)&(Rdot>0))
         y = 111;
        stop = 'Rdot stop';
    end

    if (R<0.000001)
        y = 111;
        stop = 'R stop';
```

```
    end

    if ((STOPFLAG==1)&(y==111))
        disp(['***  ',stop,'  ***'])
    end

end
```

THIS PAGE INTENTIONALLY LEFT BLANK

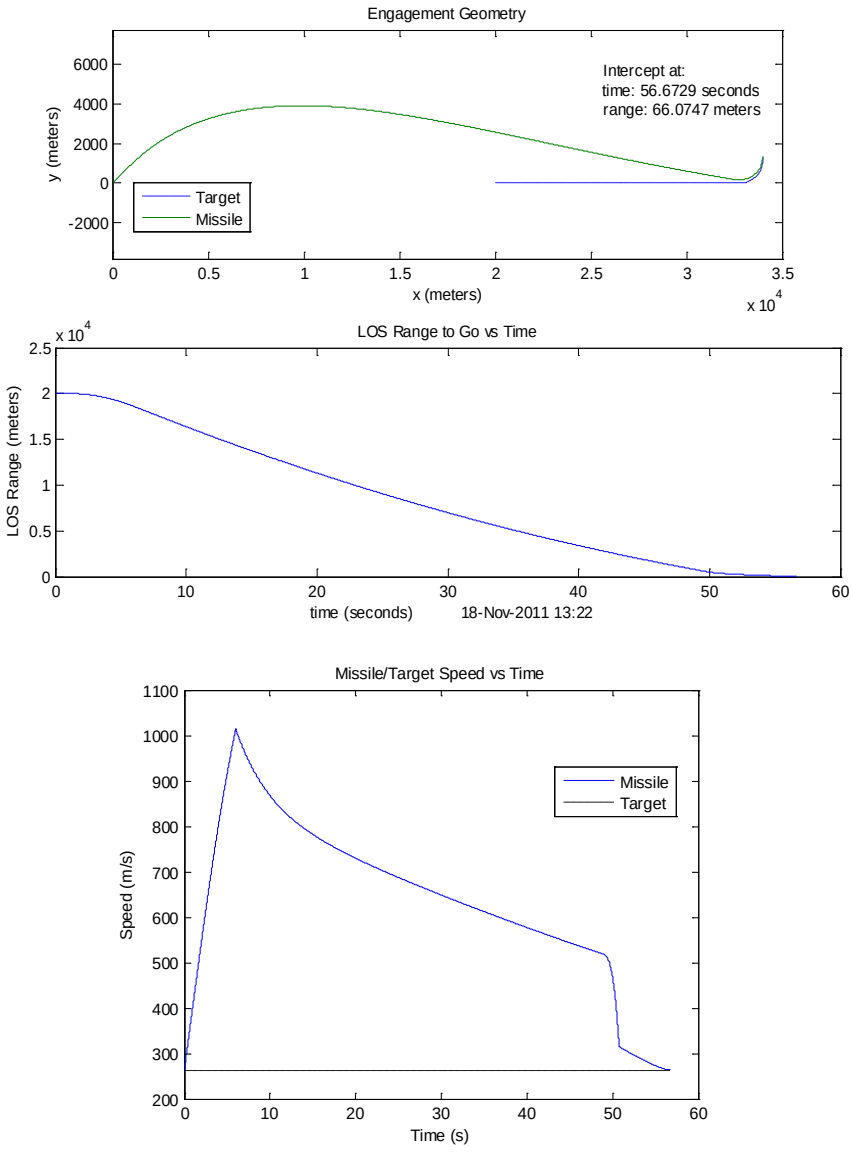
APPENDIX C. ADDITIONAL SIMULATION SCENARIOS

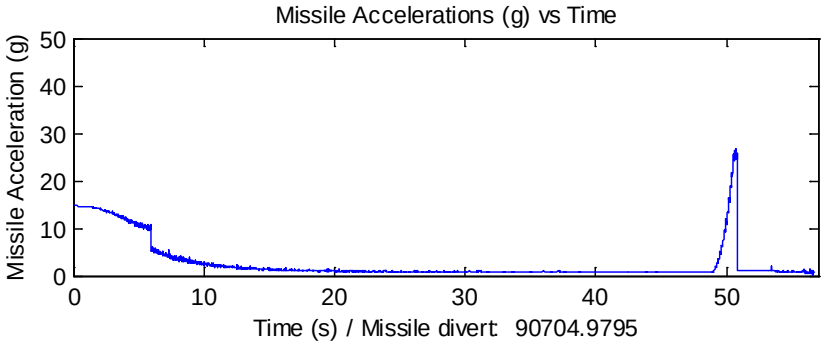
A sample of the single simulation run outputs are arranged in this appendix for each of the guidance law in a noise included scenario. For each law, a different target and missile heading is chosen with different simulation parameters to demonstrate the capability of the model and the guidance laws. The following plots are then used to present the results:

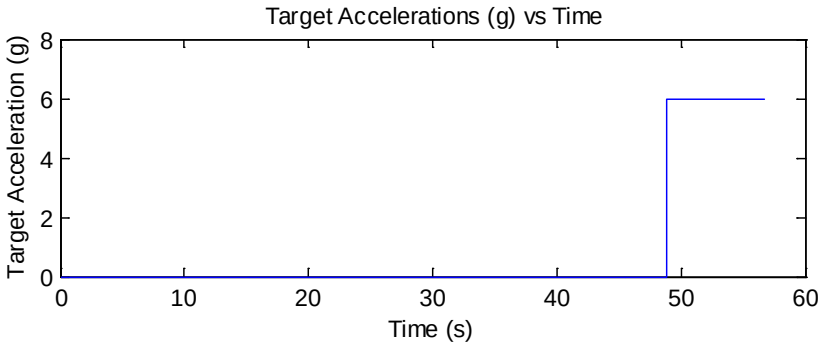
- Engagement geometry
- LOS range to go vs. time
- Missile and target speed vs. time
- Missile and target total acceleration vs. time
- Missile forces (F_x/F_y) vs. time

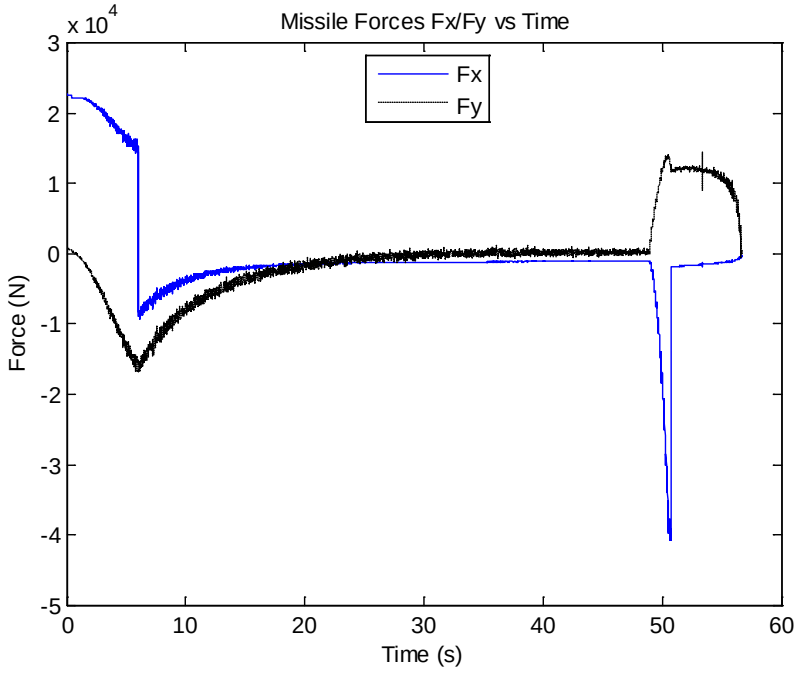
A. PN GUIDANCE LAW

The first scenario has an initial range of 20 km, direct tail chase at 0° azimuth but with the missile initially being launched pointing 45° away from the target. Co-altitude at 6000 m, with target 6g maneuver towards the missile at three seconds t_{go}.



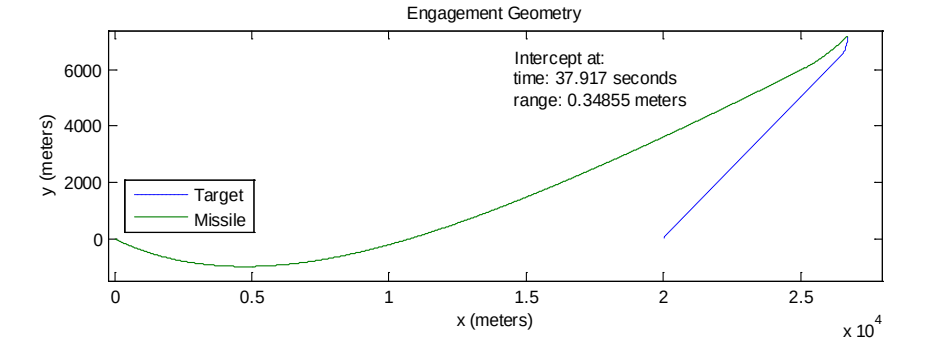

Missile Accelerations (g) vs Time
Missile Acceleration (g)
Time (s) / Missile divert: 90704.9795

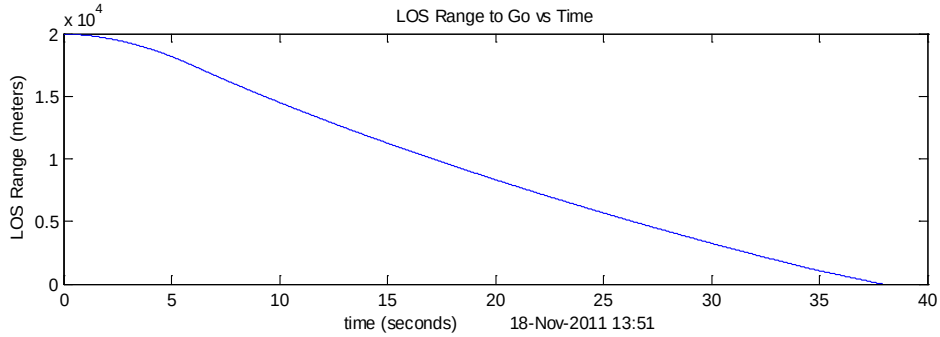

Target Accelerations (g) vs Time
Target Acceleration (g)
Time (s)

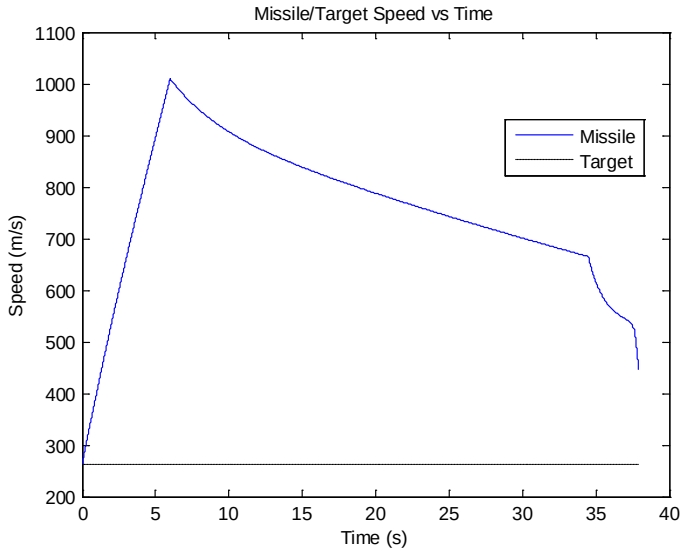

x 10^4
Missile Forces Fx/Fy vs Time
Fx
Fy
Force (N)
Time (s)

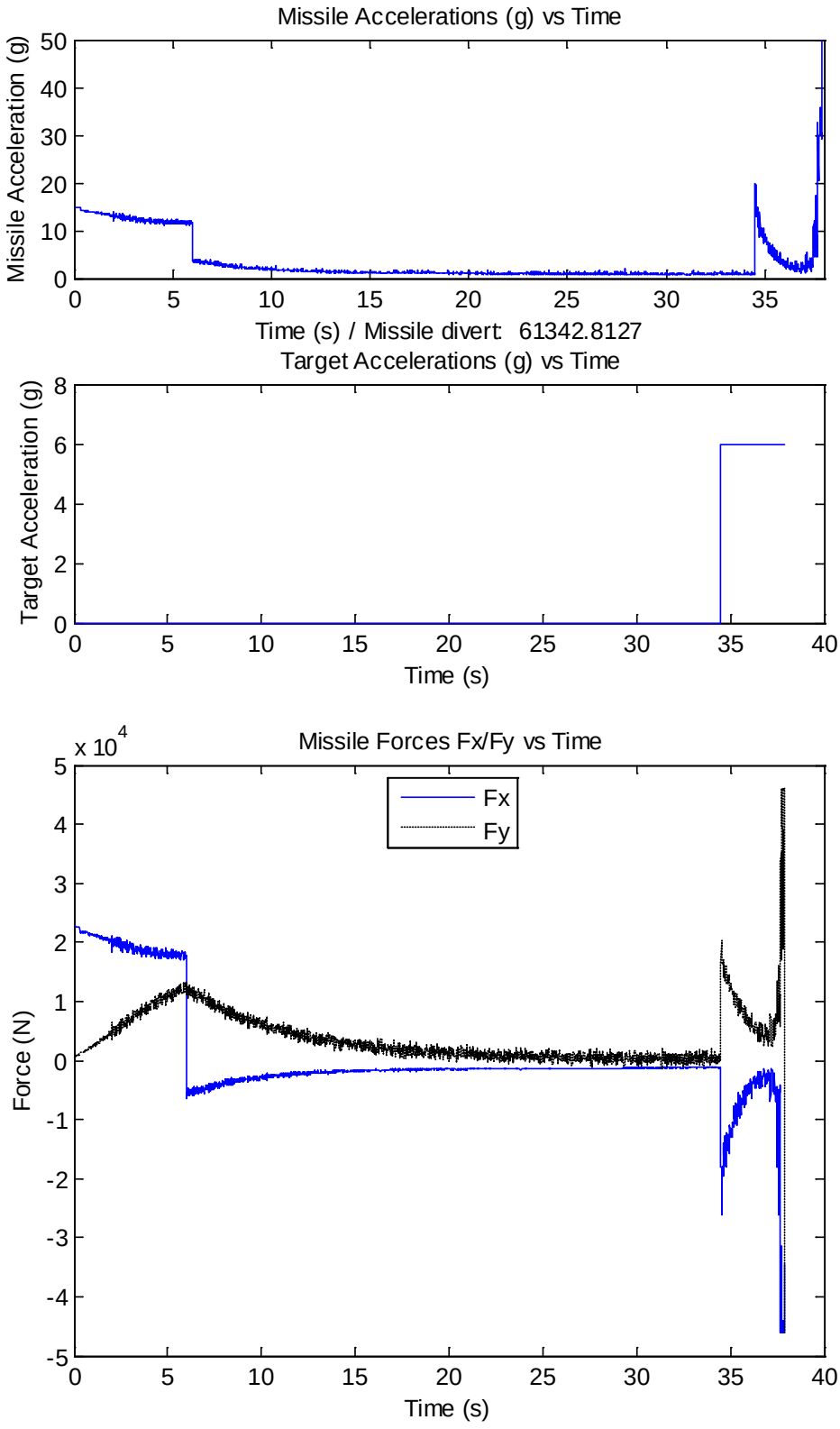
B. APN GUIDANCE LAW

This second scenario has an initial range of 20 km, at 45° azimuth but with the missile initially being launched pointing -25° away from the target. Co-altitude at 6000 m, with target 6g maneuver towards the missile at three seconds t_{go}.



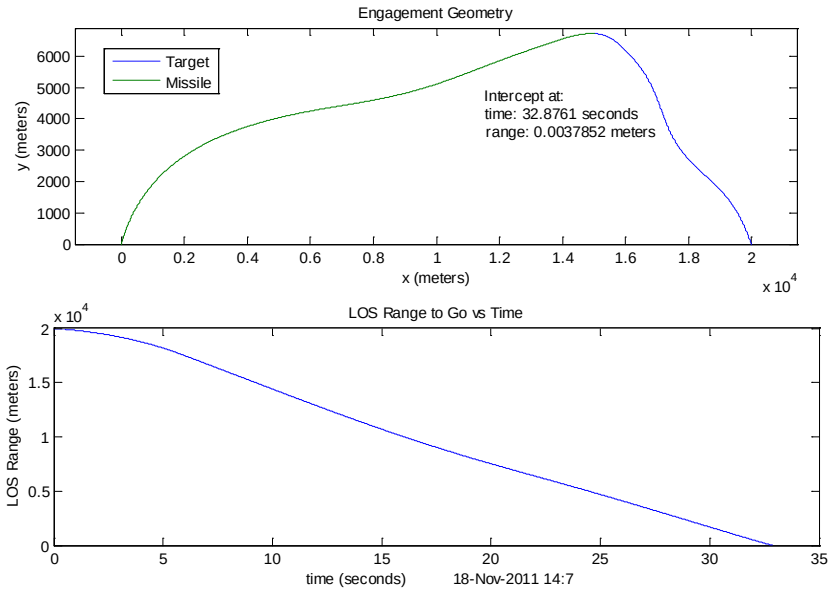


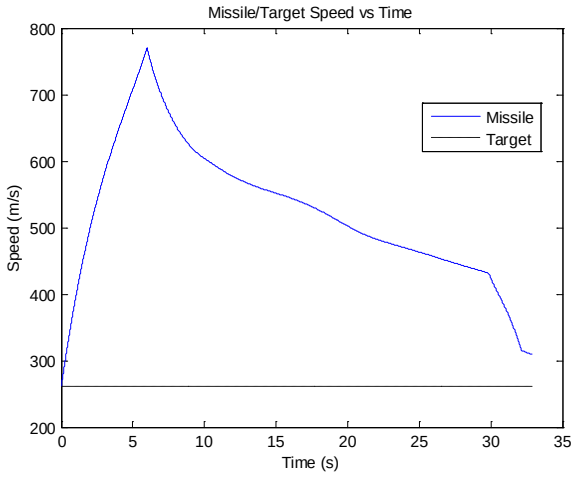


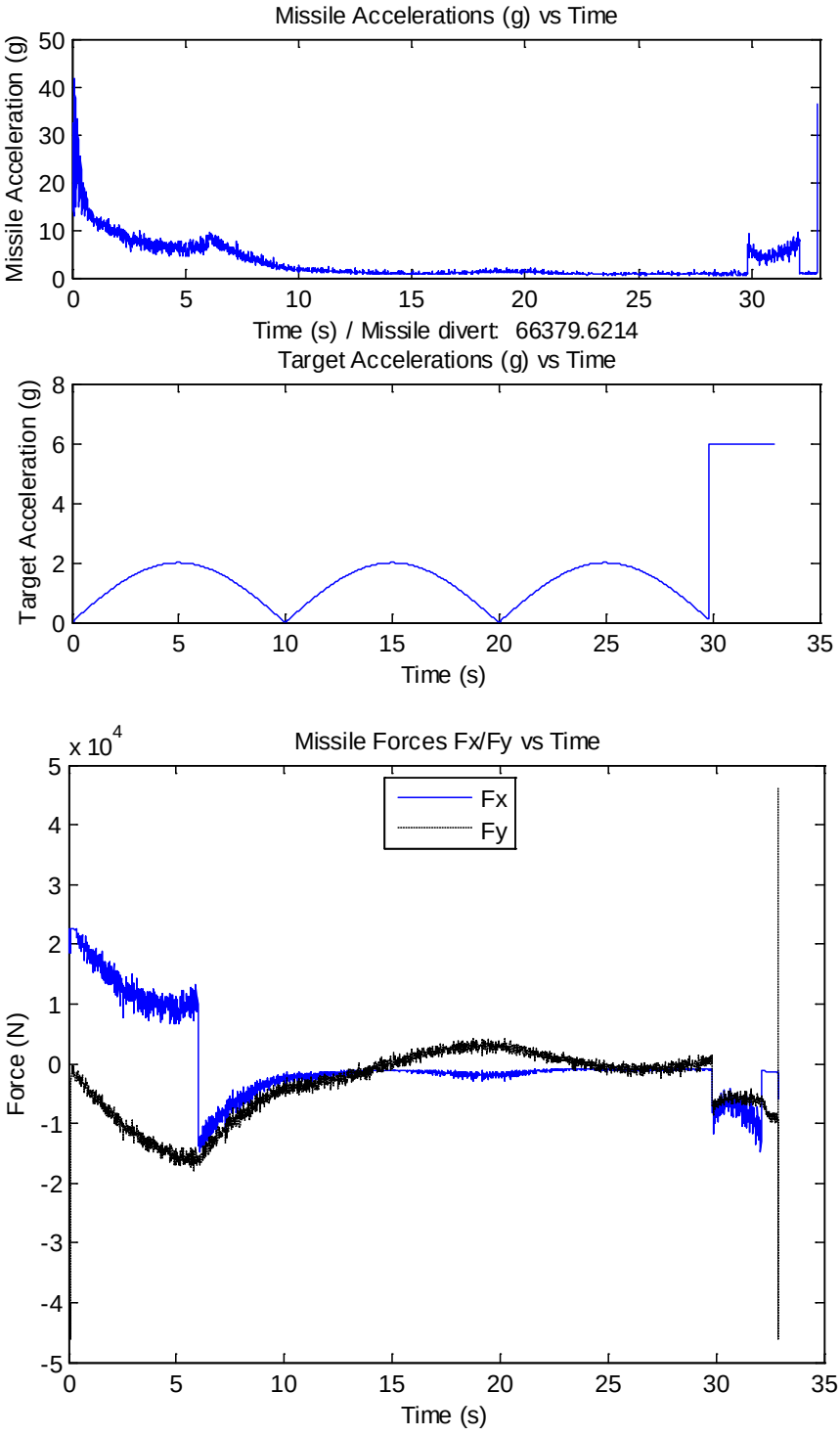

Missile Accelerations (g) vs Time
Missile Acceleration (g)
Time (s) / Missile divert: 61342.8127
Target Accelerations (g) vs Time
Target Acceleration (g)
Time (s)
x 10^4
Missile Forces Fx/Fy vs Time
Fx
Fy
Force (N)
Time (s)

C. DG GUIDANCE LAW

This third scenario has an initial range of 20 km, at 110° azimuth but with the missile initially being launched pointing 80° away from the target. Co-altitude at 6000 m. In this scenario, the target is modeled as moving with a constant 2g amplitude sinusoidal acceleration with a frequency of 0.1π rad/sec with a final 6g maneuver towards the missile at three seconds t_{go}.





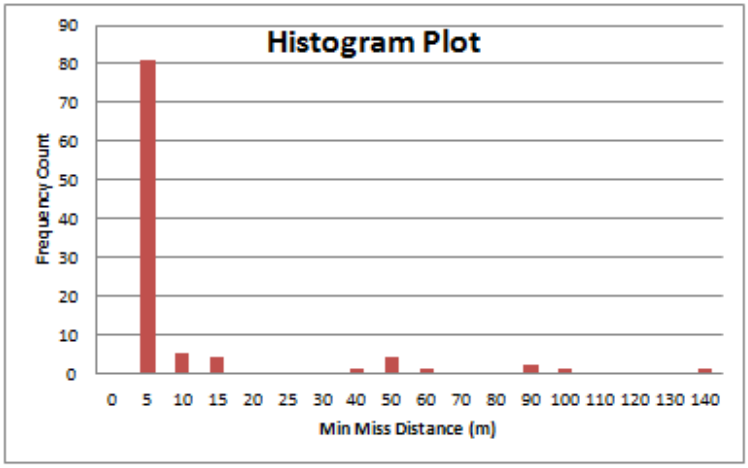

Missile Accelerations (g) vs Time
Missile Acceleration (g)
Time (s) / Missile divert: 66379.6214
Target Accelerations (g) vs Time
Target Acceleration (g)
Time (s)
Missile Forces Fx/Fy vs Time
x 10^4
Fx
Fy
Force (N)
Time (s)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. NOISE SIMULATION RESULTS DATA

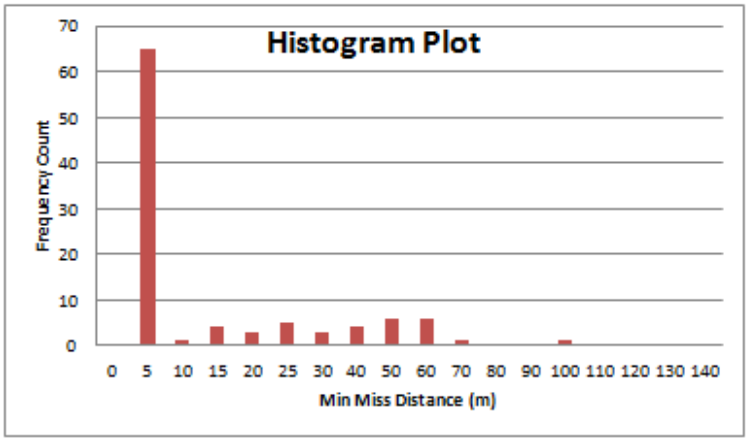
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	45 deg	0 turn	42120 m	17.7x	4929 s

Min	0.0003
Max	132.4260
Mean	7.5901
Median	0.0526
Standard Deviation	21.9122
Variance	480.1443
Miss Distance w/o noise (same sim conditions)	0.82335 m
% Hits	**81%**



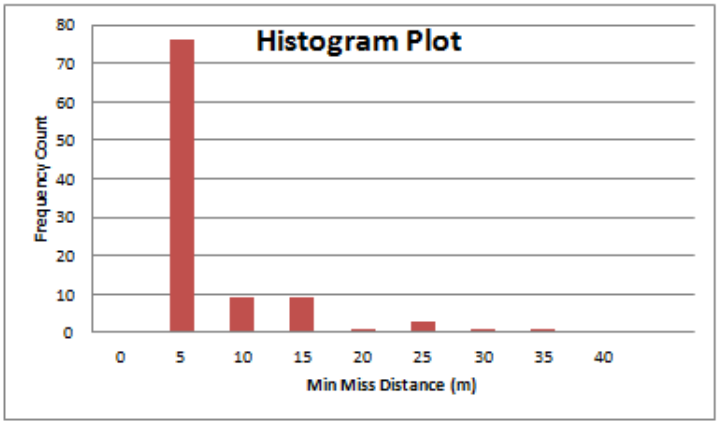
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	45 deg	0 turn	42120 m	17.8x	4666 s

Min	0.0034
Max	232.8992
Mean	14.5629
Median	0.0662
Standard Deviation	30.0578
Variance	903.4687
Miss Distance w/o noise (same sim conditions)	0.82335 m
% Hits	**65%**



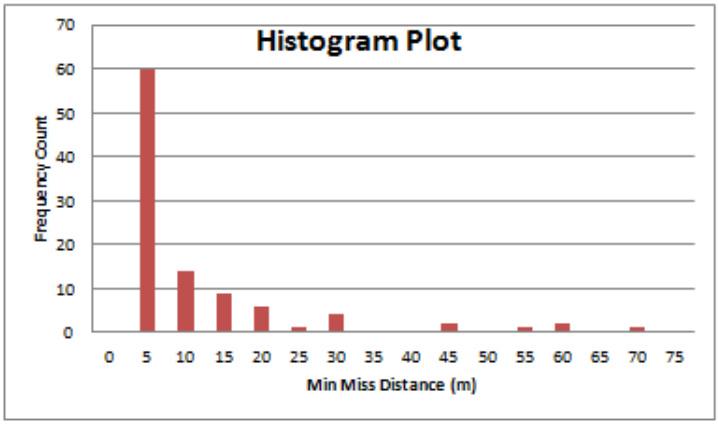
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	45 deg	0 turn	42110 m	7.2x	4306 s

Min	0.0682
Max	30.6004
Mean	4.4470
Median	2.0601
Standard Deviation	6.2311
Variance	38.8270
Miss Distance w/o noise (same sim conditions)	2.6854m
% Hits	**76%**



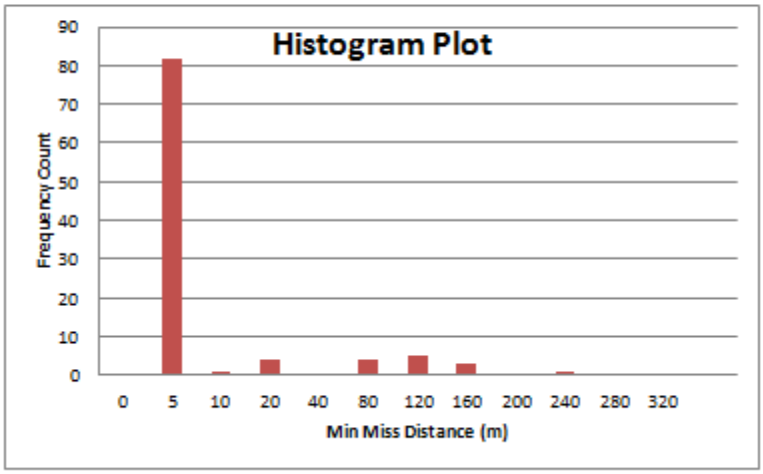
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	45 deg	0 turn	42110 m	7.3x	4468 s

Min	0.0292
Max	67.3485
Mean	8.6013
Median	3.0416
Standard Deviation	13.2316
Variance	175.0759
Miss Distance w/o noise (same sim conditions)	2.6854m
% Hits	**60%**



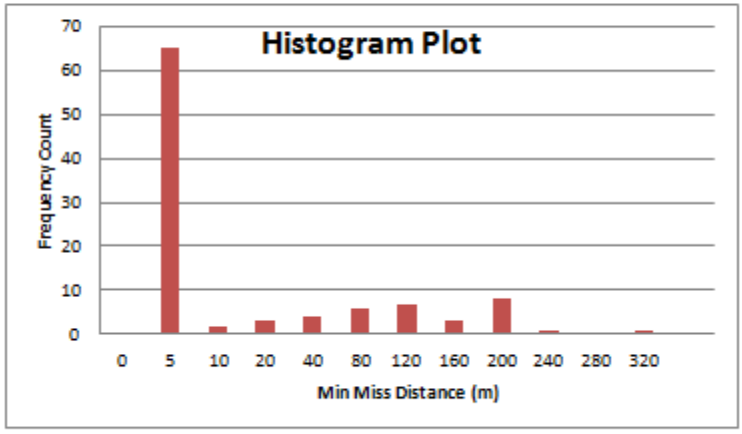
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	45 deg	0 turn	42500 m	16.2x	5214 s

Min	0.0002
Max	237.1886
Mean	14.5129
Median	0.0741
Standard Deviation	40.3892
Variance	1631.2854
Miss Distance w/o noise (same sim conditions)	0.62104m
% Hits	**82%**



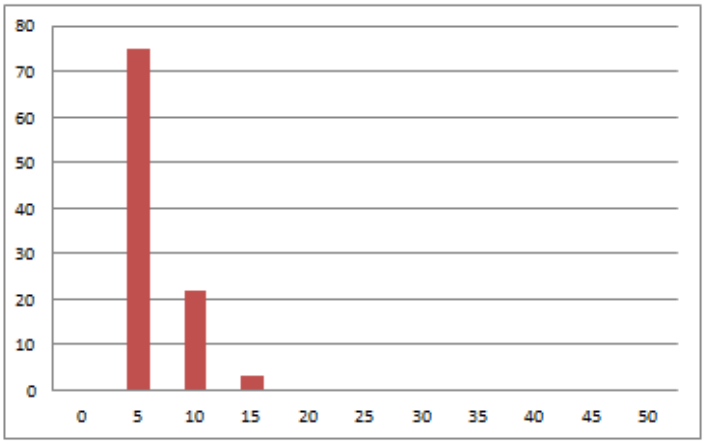
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	45 deg	0 turn	42500 m	16.3x	5206 s

Min	0.0004
Max	284.3137
Mean	35.9978
Median	0.1101
Standard Deviation	63.9568
Variance	4090.4730
Miss Distance w/o noise (same sim conditions)	0.62104m
% Hits	**65%**



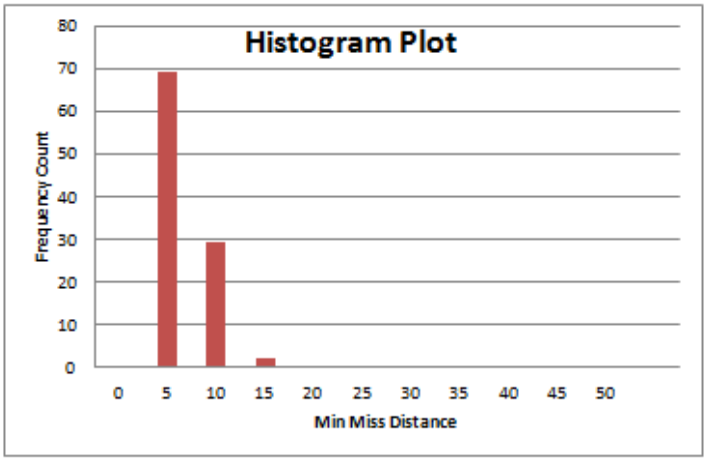
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	45 deg	6 turn	35270 m	16x	2797 s

Min	0.0019
Max	11.3017
Mean	3.3956
Median	3.1703
Standard Deviation	2.6143
Variance	6.8347
Miss Distance w/o noise (same sim conditions)	0.18861m
% Hits	**75%**



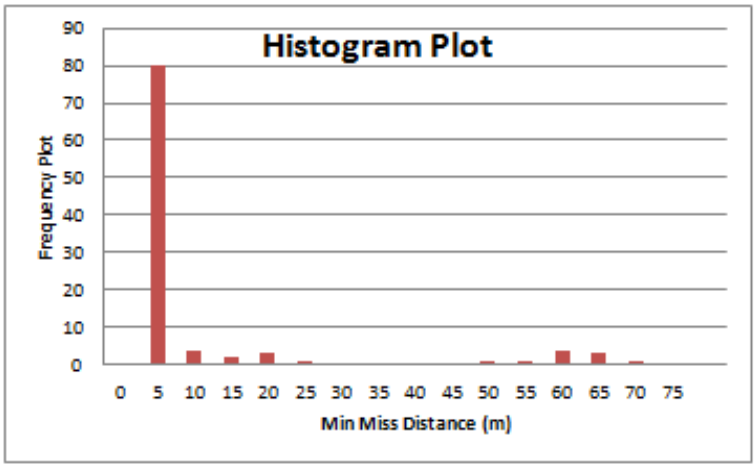
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	45 deg	6 turn	35270 m	16.1x	2798 s

Min	0.0026
Max	11.9976
Mean	3.6613
Median	3.3864
Standard Deviation	2.6794
Variance	7.1791
Miss Distance w/o noise (same sim conditions)	0.18861m
% Hits	**69%**



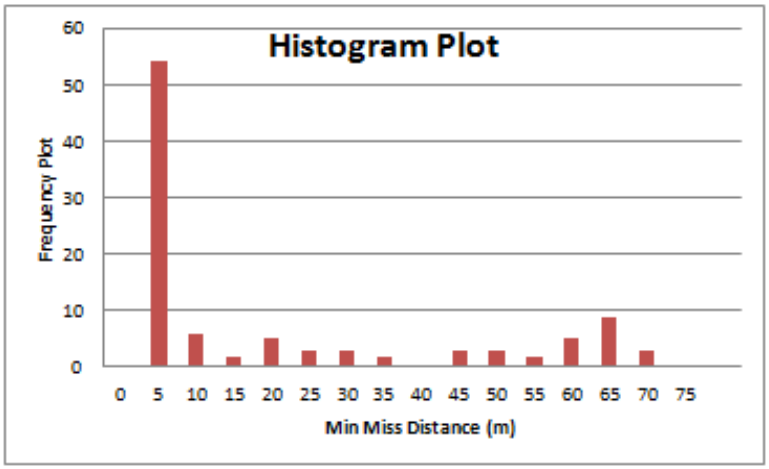
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	45 deg	6 turn	42010 m	7.3x	4244 s

Min	0.1924
Max	65.3089
Mean	8.3948
Median	1.9247
Standard Deviation	17.1722
Variance	294.8857
Miss Distance w/o noise (same sim conditions)	0.57701m
% Hits	**80%**



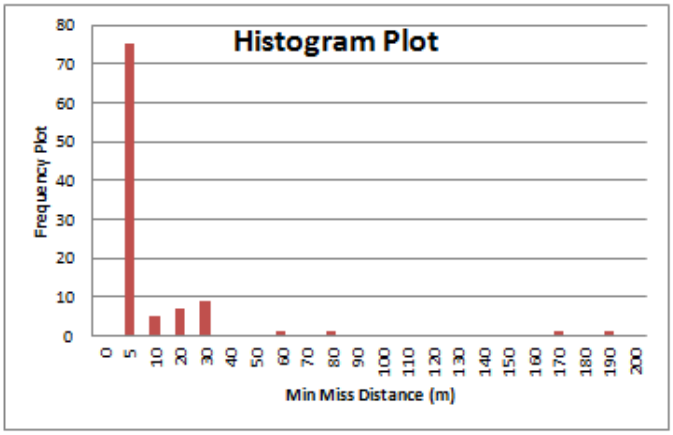
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	45 deg	6 turn	42010 m	7.4x	4244 s

Min	0.2164
Max	69.1988
Mean	19.1916
Median	3.8352
Standard Deviation	23.5596
Variance	555.0534
Miss Distance w/o noise (same sim conditions)	0.57701m
% Hits	**54%**



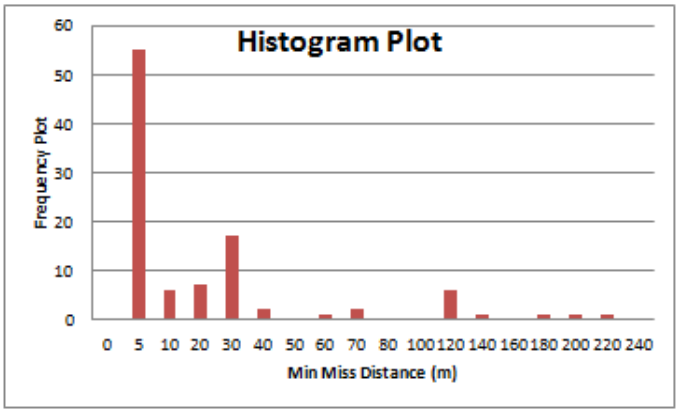
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	45 deg	6 turn	47030 m	16.8x	4886 s

Min	0.0002
Max	188.6313
Mean	8.6715
Median	0.1817
Standard Deviation	26.5695
Variance	705.9400
Miss Distance w/o noise (same sim conditions)	0.035274m
% Hits	**75%**



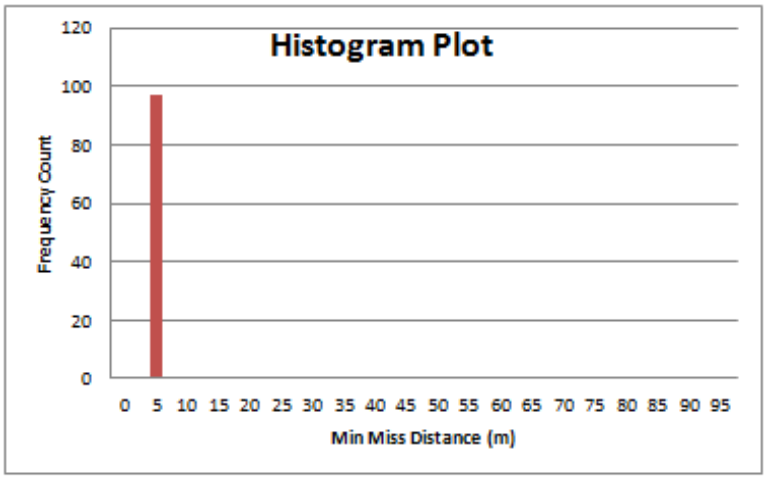
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	45 deg	6 turn	47030 m	16.9x	4874 s

Min	0.0021
Max	208.2754
Mean	22.3385
Median	1.2432
Standard Deviation	42.5965
Variance	1814.4610
Miss Distance w/o noise (same sim conditions)	0.035274m
% Hits	**55%**



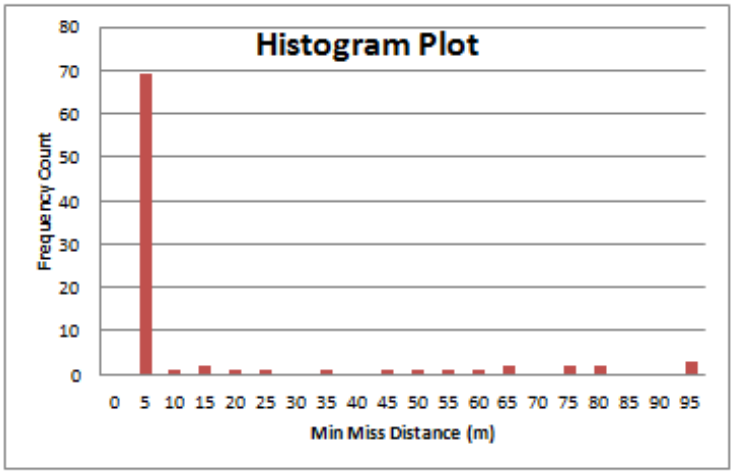
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	135 deg	0 turn	83270 m	4.9x	4806 s

Min	0.0194
Max	76.1915
Mean	2.5258
Median	0.8874
Standard Deviation	10.7369
Variance	115.2809
Miss Distance w/o noise (same sim conditions)	2.3919 m
% Hits	**97%**



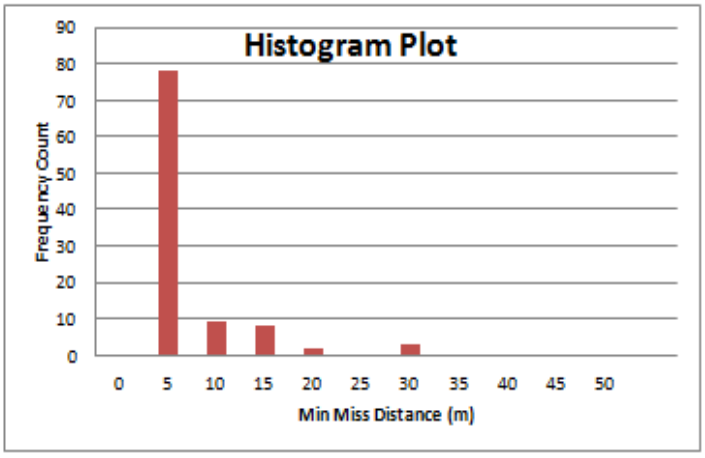
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	135 deg	0 turn	83270 m	5x	4758 s

Min	0.0013
Max	174.0091
Mean	25.3632
Median	1.1047
Standard Deviation	43.7803
Variance	1916.7121
Miss Distance w/o noise (same sim conditions)	2.3919 m
% Hits	**69%**



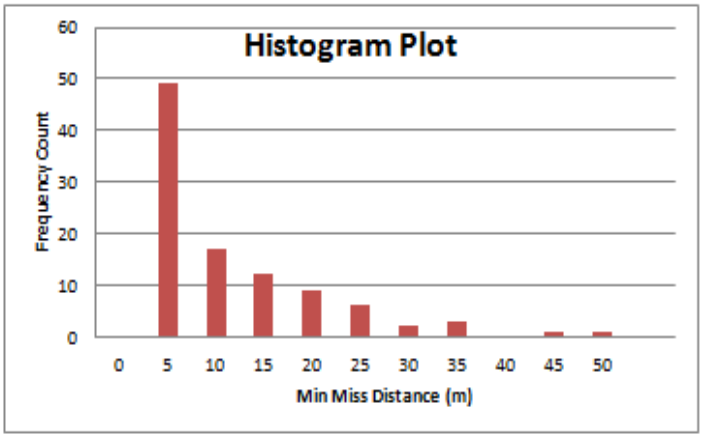
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	135 deg	0 turn	83090 m	5.3x	4936 s

Min	0.0205
Max	28.8667
Mean	3.6803
Median	1.1109
Standard Deviation	5.9450
Variance	35.3430
Miss Distance w/o noise (same sim conditions)	1.0216m
% Hits	**78%**



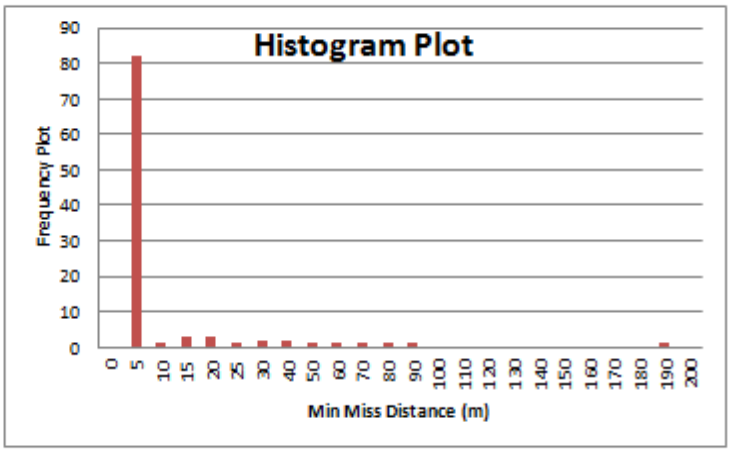
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	135 deg	0 turn	83090 m	5.4x	4795 s

Min	0.1590
Max	49.1520
Mean	8.7913
Median	5.8286
Standard Deviation	9.9349
Variance	98.7024
Miss Distance w/o noise (same sim conditions)	1.0216m
% Hits	**49%**



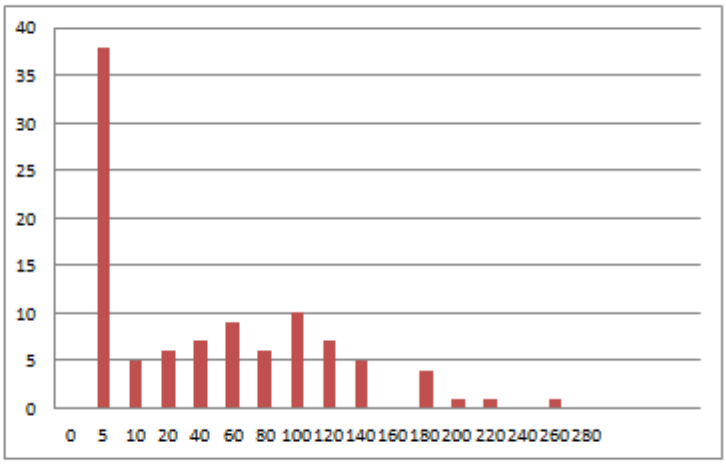
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	135 deg	0 turn	81120 m	4.5x	4929 s

Min	0.0515
Max	187.8093
Mean	8.0243
Median	0.7488
Standard Deviation	23.6310
Variance	558.4258
Miss Distance w/o noise (same sim conditions)	1.7183m
% Hits	**82%**



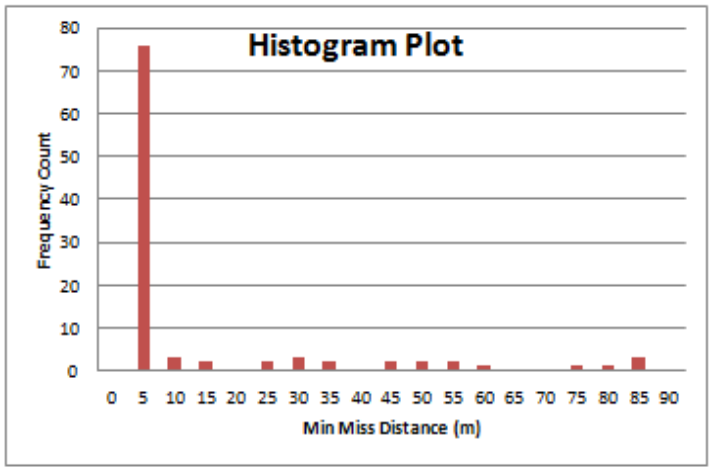
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	135 deg	0 turn	81120 m	4.6x	5102 s

Min	0.1094
Max	243.3163
Mean	48.7813
Median	27.1343
Standard Deviation	57.6461
Variance	3323.0703
Miss Distance w/o noise (same sim conditions)	1.7183m
% Hits	**38%**



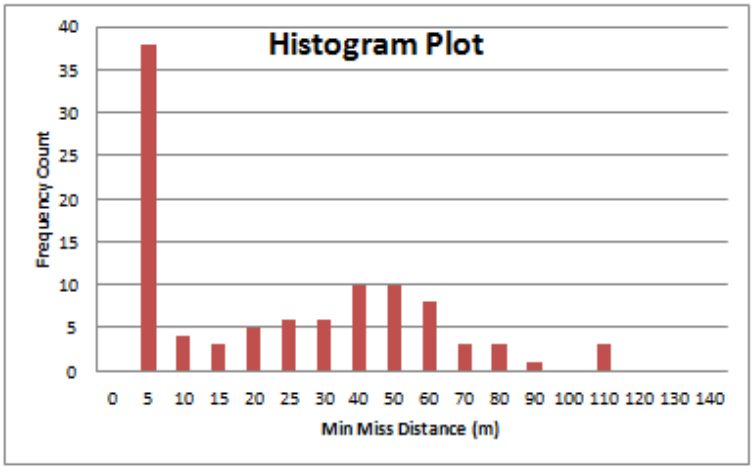
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	135 deg	6 turn	79640 m	6.1x	4750 s

Min	0.0067
Max	83.8048
Mean	10.7393
Median	1.7782
Standard Deviation	20.5951
Variance	424.1596
Miss Distance w/o noise (same sim conditions)	1.2905m
% Hits	**76%**



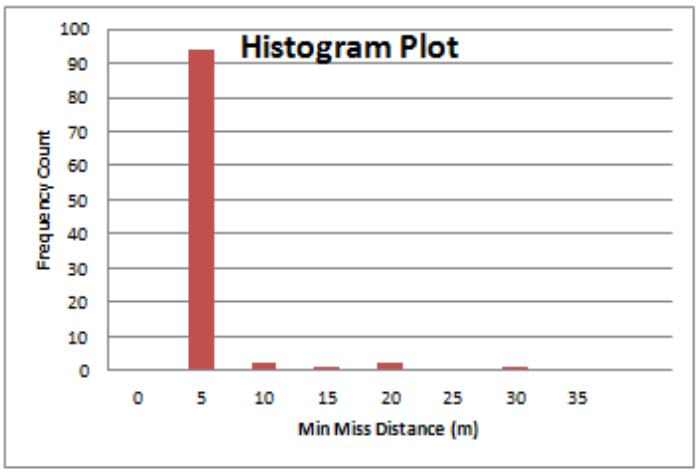
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
PN	135 deg	6 turn	79640 m	6.2x	4369 s

Min	0.2698
Max	109.9653
Mean	25.7535
Median	20.2508
Standard Deviation	27.0179
Variance	729.9655
Miss Distance w/o noise (same sim conditions)	1.2905m
% Hits	**38%**



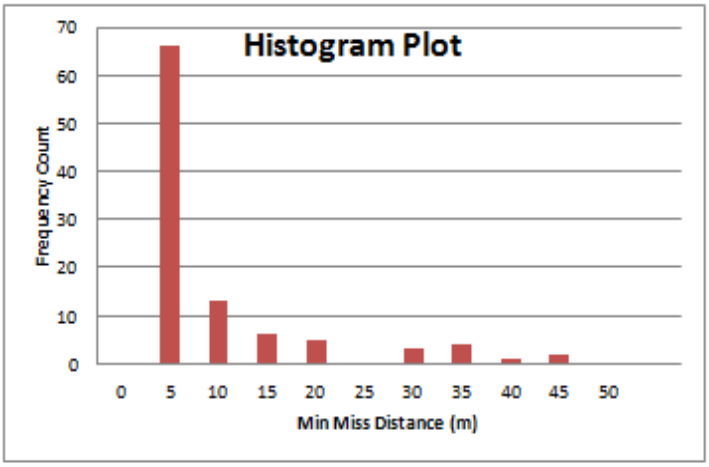
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	135 deg	6 turn	82080m	5.5x	4353 s

Min	0.0352
Max	26.7813
Mean	2.0190
Median	1.2028
Standard Deviation	3.6137
Variance	13.0591
Miss Distance w/o noise (same sim conditions)	1.9015m
% Hits	**94%**



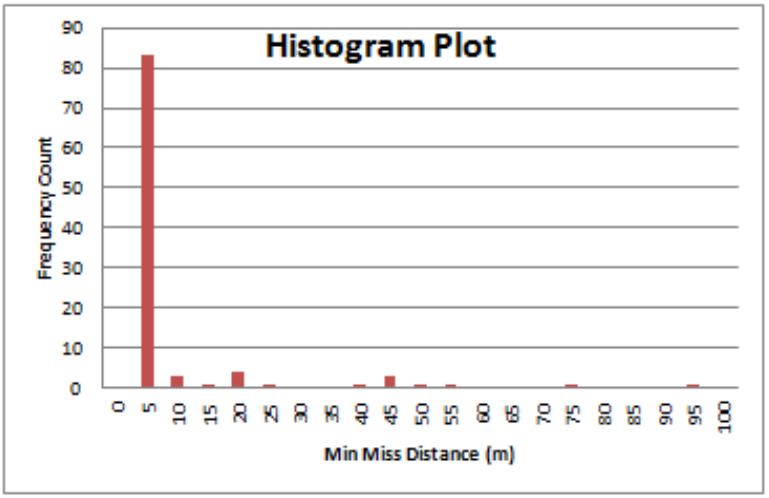
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
APN	135 deg	6 turn	82080m	5.6x	4539 s

Min	0.0731
Max	43.6102
Mean	6.9663
Median	1.8493
Standard Deviation	10.3143
Variance	106.3846
Miss Distance w/o noise (same sim conditions)	1.9015m
% Hits	**66%**



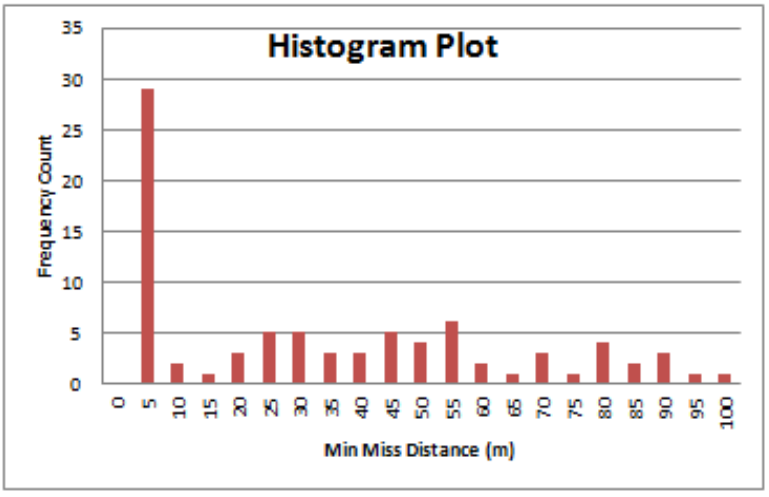
Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	135 deg	6 turn	79900 m	4.7x	4811 s

Min	0.0015
Max	94.1398
Mean	6.5540
Median	1.5178
Standard Deviation	15.4116
Variance	237.5164
Miss Distance w/o noise (same sim conditions)	2.637m
% Hits	**83%**



Guidance Law	Heading	Target Maneuver	Max Range	Factor of Baseline Noise	Simulation Run Time
DG	135 deg	6 turn	79900 m	4.8x	4669 s

Min	0.0171
Max	177.5913
Mean	48.1942
Median	39.0509
Standard Deviation	47.1788
Variance	2225.8394
Miss Distance w/o noise (same sim conditions)	2.637m
% Hits	**29%**



LIST OF REFERENCES

[1] C. Yeager and L. Janos, *Yeager: An Autobiography.* New York: Bantam Books, 1985.

[2] N. A. Shneydor, *Missile Guidance and Pursuit: Kinematics, Dynamics and Control.* Cambridge, UK: Woodhead Publishing Limited, 2008.

[3] P. Zarchan, *Tactical and Strategic Missile Guidance*, 4th ed. Reston, VA: American Institute of Aeronautics and Astronautics Inc., 2002.

[4] Y. C. Chiou, and C. Y. Kuo, "Geometric approach to three-dimensional missile guidance problem," *Journal of Guidance, Control, and Dynamics,* vol 21, no. 2, pp. 335–341, 1998.

[5] J. A. Lukacs, IV, and O. Yakimenko, "Trajectory-shape-varying missile guidance interception of ballistic missiles during the boost phase." *AIAA Guidance, Navigation and Control Conference and Exhibit.* Hilton Head, SC: Aug 2007.

[6] J. Z. Ben-Asher, and I. Yaesh, *Advances in Missile Guidance Theory.* Vol. 180, P. Zarchan, ed. Reston, VA: American Institute of Aeronautics and Astronautics, 1998.

[7] R. Yanushevsky, *Modern Missile Guidance.* Boca Raton, FL: CRC Press, 2008.

[8] R. Goodstein, "Guidance law applicability for missile closing," *Guidance and Control of Tactical Missiles*, AGARD Lecture Series, no. 52, May 1972.

[9] R. L. Shaw, *Fighter Combat: Tactics and Maneuvering.* Annapolis, MD: Naval Institute Press, 1988.

[10] U. S. Shukla and P. R. Mahapatra, "The proportional navigation dilemma—pure or true?" *IEEE Trans. on Aerospace and Electronics Systems*, vol. 26, no. 2, pp. 382–392, March 1990.

[11] R. D. Broadston, "A method of increasing the kinematic boundary of air-to-air missiles using an optimal control approach," M.S. thesis, Naval Postgraduate School, Monterey, CA, 2000.

[12] R. G. Hutchins, "Navigation, Missile, and Avionics Systems," class notes for EC 4340, Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, Spring 2011.

[13] B. L. Stevens, and F. L. Lewis, *Aircraft Control and Simulation*, 2nd ed. Hoboken, NJ: John Wiley & Sons, Inc, 2003.

[14] J. H. Blakelock, *Automatic Control of Aircraft and Missile,* 2nd ed. New York: John Wiley & Sons, 1991

[15] J. D. Anderson, *Fundamentals of Aerodynamic,* 2nd ed. Los Angeles, CA: McGraw-Hill, Inc., 1991.

[16] C. Y. Kuo, and Y. C. Chiou, "Geometric analysis of missile guidance command," *IEEE Proceedings: Control Theory and Appliations.* vol 147, no. 2, pp. 205–211, 2000.

[17] C. Y. Li, W. X. Jing, H. Wang, and Z. G. Qi, "Application of 2D differential geometric guidance to tactical missile interception," *IEEE Aerospace Conference, 25th Chinese Control Conference,* 2006.

[18] C. F. Lin, *Modern Navigation, Guidance, and Control Processing.* Englewood Cliffs, NJ: Prentice-Hall Inc, 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code ME
   Department of Mechanical and Aerospace Engineering
   Naval Postgraduate School
   Monterey, California

4. Associate Professor Robert G. Hutchins, Code EC/Hu
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor Oleg Yakimenko, Code SE/Yk
   Department of Systems Engineering &
   Department of Mechanical and Aerospace Engineering
   Naval Postgraduate School
   Monterey, California

6. Professor Yeo Tat Soon
   Temasek Defense Systems Institute
   National University of Singapore
   Singapore

7. Ms Tan Lai Poh
   Temasek Defense Systems Institute
   National University of Singapore
   Singapore