

NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



THESIS

**COMPUTER MODEL AND SIMULATION OF A
THEATER BALLISTIC MISSILE (TBM)
COUNTERFORCE PLAN INVOLVING A
LETHAL UNMANNED AIR VEHICLE (UAV)**

by

Richard W. Kammann, Jr.
September, 1995

Thesis Advisor:

Isaac I. Kaminer

Approved for public release; distribution is unlimited.

19960315 121

DTIC QUALITY INSPECTED 1

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 September 19, 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE COMPUTER MODEL AND SIMULATION OF A THEATER BALLISTIC MISSILE (TBM) COUNTERFORCE PLAN INVOLVING A LETHAL UNMANNED AIR VEHICLE (UAV)		5. FUNDING NUMBERS		
6. AUTHOR(S) Kammann, Richard W., Jr.				
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)		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.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) In cooperation with the Operations Research Department at the Naval Postgraduate School, this thesis introduces a computer model for simulating the integration of a lethal unmanned air vehicle (UAV) into a tactical ballistic missile (TBM) counterforce plan. The current capability of autonomous and precise UAV trajectory tracking utilizing an onboard Global Positioning System (GPS) integrated guidance, navigation, and control (GNC) suite lends itself exceptionally to this role. The counterforce concept implies destruction of the TBM transporter-erector-launcher (TEL) within enemy territory. The study of this concept has recently been directed towards emulating well established anti-submarine warfare (ASW) search and destroy methods. The scenario presented in this thesis leaves that premise intact by the implementation of unattended ground sensors (UGS) for the purpose of detecting, tracking and classifying the TEL vehicle, and cueing the lethal UAV for attack. The end result of this work is a viable simulation for use in various future analyses. The simulation design allows for easy modification and expansion, and will serve as a valuable tool in the exploration of TBM counterforce alternatives.				
14. SUBJECT TERMS Unmanned Air Vehicle (UAV), Lethal UAV, Theater Ballistic Missile (TBM) Defense			15. NUMBER OF PAGES 111	
			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 UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18 298-102

Approved for public release; distribution is unlimited.

**COMPUTER MODEL AND SIMULATION OF A THEATER BALLISTIC
MISSILE (TBM) COUNTERFORCE PLAN INVOLVING A LETHAL
UNMANNED AIR VEHICLE (UAV)**

Richard W. Kammann, Jr.
Lieutenant, United States Navy
B.S., United States Naval Academy, 1988

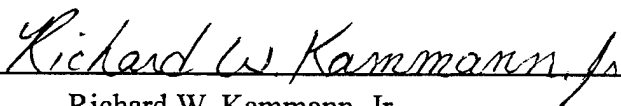
Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

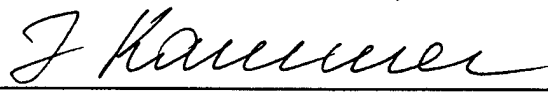
from the

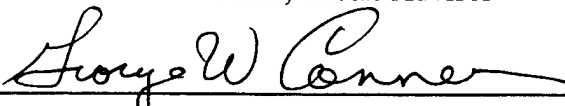
**NAVAL POSTGRADUATE SCHOOL
September, 1995**

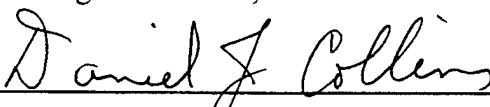
Author:


Richard W. Kammann, Jr.

Approved by:


Isaac I. Kaminer, Thesis Advisor


George W. Conner, Second Reader


Daniel J. Collins, Chairman
Department of Aeronautics and Astronautics

ABSTRACT

In cooperation with the Operations Research Department at the Naval Postgraduate School, this thesis introduces a computer model for simulating the integration of a lethal unmanned air vehicle (UAV) into a tactical ballistic missile (TBM) counterforce plan. The current capability of autonomous and precise UAV trajectory tracking utilizing an onboard Global Positioning System (GPS) integrated guidance, navigation, and control (GNC) suite lends itself exceptionally to this role. The counterforce concept implies destruction of the TBM transporter-erector-launcher (TEL) within enemy territory. The study of this concept has recently been directed towards emulating well established anti-submarine warfare (ASW) search and destroy methods. The scenario presented in this thesis leaves that premise intact by the implementation of unattended ground sensors (UGS) for the purpose of detecting, tracking and classifying the TEL vehicle, and cueing the lethal UAV for attack. The end result of this work is a viable simulation for use in various future analyses. The simulation design allows for easy modification and expansion, and will serve as a valuable tool in the exploration of TBM counterforce alternatives.

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	SCENARIO	7
	A. REAL WORLD PERSPECTIVE	7
	B. COMPUTER MODELING PERSPECTIVE	10
III.	UAV MODEL	13
	A. EQUATIONS OF MOTION	13
	1. Equations for Linear Motion	14
	2. Equations for Angular Rotation	14
	3. Equations for External Forces and Moments	15
	B. CONTROLLER DESIGN	19
	C. TRAJECTORY GENERATOR	25
	D. INTEGRATED INERTIAL NAVIGATION SYSTEM (INS)/GLOBAL POSITIONING SYSTEM (GPS)	29
	1. The INS Model	29
	2. The GPS Receiver and Pseudorange Models	33
	3. Kalman Filter Integration of INS and GPS	35
IV.	MODELING THE GROUND COMPONENTS	43
	A. TRANSPORTER-ERECTOR-LAUNCHER (TEL) AND ACOUSTIC SIGNATURE MODELS	43
	B. UNATTENDED GROUND SENSOR (UGS) MODEL	45
V.	UAV MISSILE MODEL	51
VI.	RUNNING THE SIMULATION	57
VII.	RECOMMENDATIONS AND CONCLUSIONS	65
	APPENDIX. SUPPORTING MATLAB CODE	69

LIST OF REFERENCES	93
INITIAL DISTRIBUTION LIST	95

LIST OF FIGURES

1.1	Comparison of Effects of Improving Attack on Incoming Missile and Tel From Ref. [3]	2
2.1	Big Picture of Counterforce Concept	9
2.2	Integration of Model Components	11
3.1	Equations of Motion Model for UAV	19
3.2	Standard LQR Feedback Configuration	20
3.3	Synthesis Model of Aircraft	23
3.4	Nonlinear UAV Model With Nonlinear Controller	24
3.5	Command Loop (left column) and Control Loop Bandwidths (right column) ..	25
3.6	Basic Trajectory Generator	26
3.7	UAV Trajectory Generator	27
3.8	Onboard Computer	28
3.9	Tangent Plane Navigation Model	31
3.10	Tangent Plane Navigation Model With Pseudorange Outputs	32
3.11	GPS Receiver Model	35
3.12	Kalman Filter Synthesis Model	37
3.13	Navigation Model and Kalman Filter Integration	39
3.14	Bode Gain Plots of Pseudorange Inputs to Pseudorange Outputs	40
3.15	Integral Kalman Filter Bias Rejections	41
4.1	TEL and Acoustic Signal Generator	46

4.2	Frequency Counter Component of UGS	48
4.3	Integrated Model of TEL, Acoustic Signal Generator, and UGS	50
5.1	Missile Model	52
5.2	Geometry Used for the LOS Controller	53
5.3	Missile PID Controller	54
6.1	Complete SIMULINK Diagram of TBM Counterforce Model	58
6.2	Simulation Display of Missile Track Prior to Launch From UAV (Looking Down Perspective)	60
6.3	Simulation Display of TEL Ground Track	60
6.4	Simulation Display of UGS Detection (of TEL) Signal	61
6.5	Simulation Display of Missile-to-TEL Slant Range	62
6.6	Simulation Display of Missile Altitude	62
6.7	3-D Plot of an Entire Simulation	63

LIST OF TABLES

3.1	Eigenvalues for Linearized UAV Model	32
3.2	Eigenvalues of UAV Model with Pseudorange Outputs	33
3.3	Eigenvalues of Filtered Navigation Model.	40

ACKNOWLEDGEMENTS

I would like to thank Professor Isaac I. Kaminer for sharing with me his vast knowledge and enthusiasm in the avionics and UAV fields during the past two years. Also, my gratitude goes out to CAPT George W. Conner, USN (Ret.) for his support and contribution to this thesis. I am grateful for the time and effort spent on portions of this project by my classmates and friends: LT John Green, USN, LT Steve Hope, USN, LT Mike Knowles, USN, LT Aaron Rondeau, USN and LT Scott Winfrey, USN. Finally, my deepest thanks to Caryn, Ryan and Daniel for their special support.

I. INTRODUCTION

The Persian Gulf War was testimony to how significant an effect an adversary can have on the overall scope of a conflict through the use of Tactical Ballistic Missiles (TBMs). Iraq's use of SCUD missiles drove much of the Allies' policy, tactics and mindset. Defense against that threat, in the forms of PATRIOT missiles and SCUD Combat Air Patrol (CAP) missions, met with little success. Post Desert Storm analyses have shown that war time battle damage assessment (BDA) was optimistically high. Revised numbers of SCUD missiles intercepted and mobile transporter-erector-launchers (TELs) destroyed indicate that the counter effort was not successful. [Ref. 1]

As with the use of PATRIOT missiles in defense of TBMs, the SCUD CAP tactic focuses on post-missile-launch action. CAP aircraft are cued simply by observation of a TBM launch, or "flaming datum," and subsequently pursue an attack on the site. There were zero confirmed TEL kills in Iraq by use of this highly reactive tactic [Ref. 2]. Because of this ineffectiveness during Desert Storm, emphasis has recently been placed on improving TBM counterforce strategy. TBM counterforce implies the detection and destruction of TELs. This is contrasted with active defense, which involves the detection and destruction of inbound missiles. The advantages of counterforce, although not validated through execution in Desert Storm, have been quantified [Ref. 3]. A sample of the results from that paper are given in Figure 1.1. It is shown that by improving attacks (increasing the probability of kill) on TELs, the reduction in enemy warheads that reach

their target is exponentially greater than that achieved by improving attacks on incoming missiles.

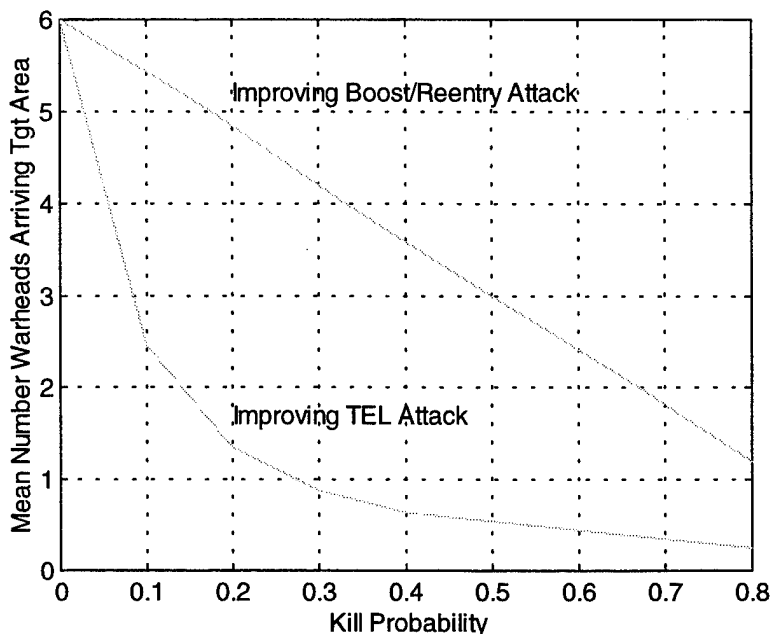


Figure 1.1 Comparison of Effects of Improving Attack on Incoming Missile and TEL. From Ref. [3].

The purpose of this thesis is to develop and introduce a TBM counterforce concept through computer modeling and simulation. The concept involves the marriage of two dynamically evolving technologies: a lethal unmanned air vehicle (UAV), and unattended ground sensors (UGS). Ideally, the system will be totally autonomous in its prosecution of TELs, and considerably less expensive and more effective than any alternative. This work serves as a starting point and tool for analysis and exploration of such a counterforce plan.

Using UAVs as attack platforms is uncommon, but not new. One of the earliest versions emerged in 1958. The QH-50, a remotely piloted helicopter (RPH), was capable of carrying a torpedo, a gun, bombs or rockets [Ref. 4]. In the 1970s, a variant of the jet powered and remotely piloted Firebee was designed to carry weapons such as Shrike and Maverick missiles [Ref. 5]. Although neither of these examples progressed beyond testing, both were seeds for today's growing interest in UAV applications. Current technology enables autonomous and precise trajectory tracking with relatively inexpensive, high endurance UAVs. Indeed, a low risk UAV has the ability to operate well beyond the forward line of own troops (FLOT) for long periods without putting humans in direct danger. These attributes render the UAV perfectly suited to a lethal role in support of TBM counterforce.

In this thesis, the UAV is used as a lethal weapon launcher. The other critical component of the counterforce scheme is the UGS system. Effective deployment of UGS is itself an issue worthy of extensive research. Much study has been dedicated to comparing TEL counterforce to the well established doctrine of anti-submarine warfare (ASW). The general principles of ASW have been learned and developed through years of experience, and apply directly to countering mobile missile launchers [Ref. 1]. Prior to the deployment of a UGS array, considerable effort must go into identifying and localizing the area of suspected TEL activity. This requires preliminary cueing by means of tactical intelligence, airborne sensors and space based sensors. With the UGS array successfully in place, it will detect, discriminate and track TELs, and uplink the data to

overhead lethal UAVs. All of this capability is realized due to the advances in ground sensor technology. Currently, second generation UGS may consist of sensor suites involving multiple types: seismic, video, infrared, magnetic, acoustic, chemical and nuclear. Additionally, by implementing neural networks, the UGS can be trained to classify and identify specific objects. Along with onboard GPS and battery preserving sleep modes, the latest UGS are far superior to first generation units used during the Vietnam war, and are a perfect device for the counterforce mission. [Ref. 2]

Combined, the lethal UAV and UGS models make up the autonomous TBM counterforce plan suggested in this thesis. To simulate the plan in action, the overall model also incorporates a mobile TEL. With the underlying objective of conceiving a low cost TBM counterforce alternative, the technologies represented in this work presently exist. This particular use of those technologies, however, is unique. Although some simplifications and assumptions had to be made because of CPU limitations, the overall model represents a viable option for TBM counterforce, and serves as a useful tool for demonstration, analysis and evolution of the concept.

To gradually introduce the reader to the complete model, the development of the major components are described in separate chapters. Prior to that, however, a general description of the overall scenario is presented in Chapter II. Chapter III is a discussion of the UAV design. The ground components of the model, including the UGS and the TEL with its associated acoustic signature, are covered in Chapter IV. Chapter V explains the design of the missile model. How to run and interpret the simulation is

discussed in Chapter VI. Finally, recommendations for future work on this project and conclusions are presented in Chapter VII. The Appendix contains a listing of the supporting computer code developed for the model.

II. SCENARIO

A. REAL WORLD PERSPECTIVE

Prior to describing the computer model in technical detail, the real world scenario that is represented in the simulation will be discussed. In doing so, many of the assumptions and simplifications will be addressed.

Presently, the stability and control data used for the UAV model is that of the *Bluebird*. *Bluebird* is a test bed UAV for navigation and guidance systems at the Naval Postgraduate School. Its use here is primarily due to the availability of its dynamic model. When considering other well suited UAVs for this mission, the lack of complete models consistently posed a problem. With its 20 lb. payload capacity, *Bluebird* is realistically undersized for this mission. However, the model code is written such that any set of data obtained for a particular aircraft may be input to the model at any time. The performance of the simulation should not change, as there are no significant maneuvering demands placed on the UAV in this scenario.

Figure 2.1 is an illustration of the overall concept represented by the simulation. It commences with the UAV, loaded with a small missile, loitering in a pre-programmed pattern, or trajectory, at a trimmed airspeed of 73 feet per second, and a selected altitude.

A road in the vicinity of the UAV is a portion of the localized area covered by a UGS array. On the road lies a string of second generation UGS incorporating onboard GPS. As a TEL approaches, a single UGS unit receives and recognizes the vehicle's acoustic signal. By measuring the Doppler shift of the acoustic input, the UGS

continuously computes the velocity of the TEL. The velocity is then integrated to yield the TELs position at any time.

In determining the TEL position, some simplifying assumptions are established. First, although the statements in Chapter I regarding the capabilities of second generation UGS systems serve to support the real world feasibility of this counterforce plan, a comprehensive computer model representing those technologies in full is well beyond the scope of this thesis. For simplicity, the UGS model in this scenario will actually resemble a first generation system. That is, the unit will exploit the frequency and amplitude characteristics of the acoustic signal generated by the moving TEL. Second, the road of interest is straight with a known heading. Whereas a real world network of UGS would compute the TEL position by means of triangulation, the computational demands imposed by a simulation of that nature would be immense. With the above assumptions in place, the UGS array can be represented by a single sensor model located on the road, thereby reducing the processing requirements of the simulation without compromising the validity of the concept. Third, the UGS unit lies directly on the road. The planning and execution involved in placing the sensor there are important issues, and should not be disregarded. However, these are extensive topics outside of the focus of this work. With this assumption, the single sensor model is further simplified. The amplitude of the acoustic signal peaks at the closest point of approach (CPA) between the TEL and UGS, which in this case occurs as the TEL moves directly over the sensor. Hence, the position

of the vehicle is known at that time, and is used as an initial condition in calculating subsequent positions.

As the TEL marks on top of the UGS, the lethal UAV is cued via radio data link. The UAV, now receiving TEL position updates from the UGS, disengages from its loiter pattern and flies towards the TEL at a constant altitude. When the weapon release parameter is met, the missile is launched from the UAV. The launch parameter consists of a maximum slant range between the UAV and TEL.

The missile model constantly receives the TEL position via data link from the UGS, and uses an onboard integrated GPS/Inertial Navigation System (INS) guidance, navigation and control (GNC) suite, similar to that of the UAV, to steer to the TELs position.

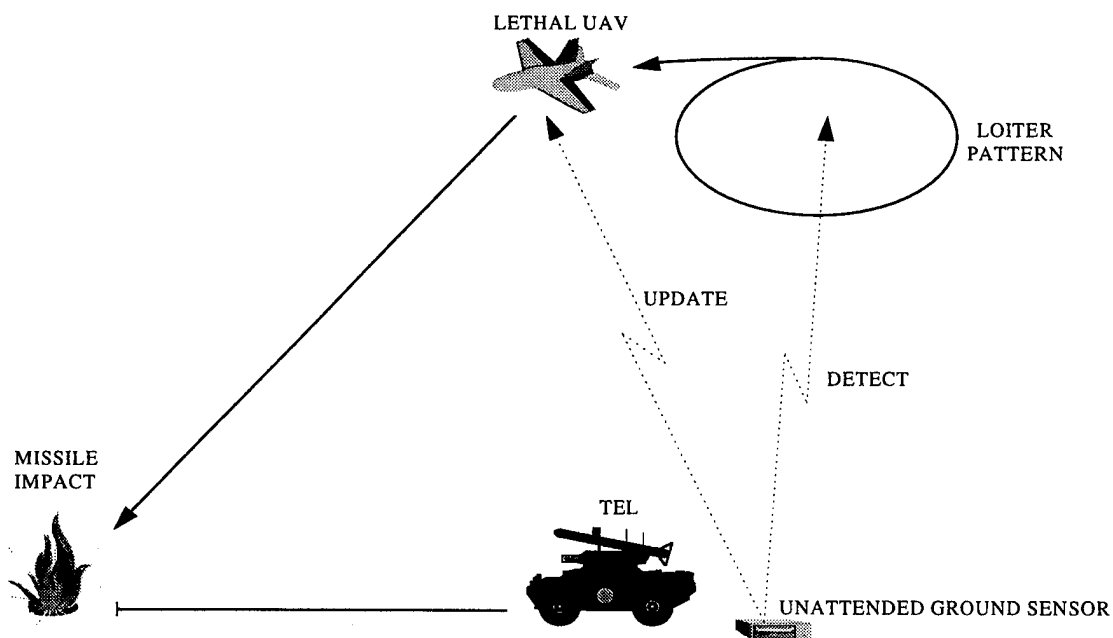


Figure 2.1. Big Picture of Counterforce Concept

B. COMPUTER MODELING PERSPECTIVE

The TBM counterforce computer model was designed using SIMULINK and MATLAB software by the *Math Works Corp.*. SIMULINK is a graphics environment that allows a user to model and simulate a system using standard block diagrams and control systems conventions. It is designed to work with MATLAB, a matrix manipulation program that has become an industry and academic standard for solving both linear and non-linear systems of equations.

The simulation of the scenario was developed on state-of-the-art *Silicon Graphics* workstations in the Department of Aeronautics and Astronautics at the Naval Postgraduate School. Nonetheless, the model's computational demands on processor resources proved to be a limiting consideration, and necessitated the application of the aforementioned simplifications and assumptions. In doing so, simulation run times have been reduced to a reasonable 20 minutes.

In order to successfully simulate the entire counterforce system, it was necessary to model several components individually. In a modular fashion, each segment was designed and tested singularly, and then added to previous components until a working model of the whole system was complete. The design process resulted in a very flexible model, allowing for future addition, removal and modification of new components as the plan evolves. The following is a list of the individual components that make up the entire lethal UAV model:

- 1) the UAV model, or aircraft equations of motion (EOM),

- 2) a dynamic controller to provide trajectory tracking for the UAV model,
- 3) a trajectory generator to provide x, y, and z position commands to the controller,
- 4) an INS system (for both the UAV and the missile),
- 5) a GPS receiver (for both the UAV and the missile), and satellites,
- 6) a Kalman filter (for both the UAV and the missile) to integrate the INS and GPS outputs,
- 7) a moving TEL and its associated acoustic signature,
- 8) a UGS model, and
- 9) a guided missile model.

Figure 2.2 is a simplified block diagram depicting all of the model components and their interaction.

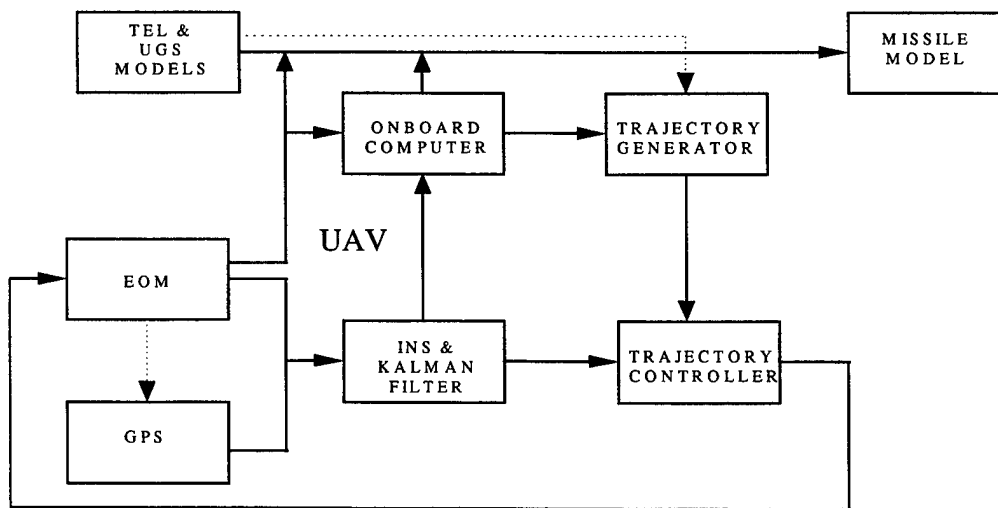


Figure 2.2. Integration of Model Components

In order to integrate the listed elements into a single model for coherent simulation of the counterforce scenario, numerous connectors and switches were implemented. These will be discussed, along with each part of the simulation, in the following sections.

III. UAV MODEL

A. EQUATIONS OF MOTION

A very detailed treatment of the development of rigid body equations of motion as they apply to the UAV is found in Reference [6]. The same procedure was carried out in the development of the UAV model for this project. This section contains highlights of that work.

The derivation of the equations of motion for this six degree of freedom aircraft model are based on the following coordinate systems, notation, and assumptions:

- 1) $\{U\}$ represents the inertial tangent plane coordinate system attached to Earth.
- 2) $\{B\}$ represents the body-fixed coordinate system.
- 3) The aircraft utilizes a strapdown IMU and, therefore, the output of the IMU sensors are resolved in $\{B\}$. Accordingly, the equations of motion are developed in $\{B\}$.
- 4) All sensors are located at the aircraft center-of-gravity (c.g.).
- 5) The mass of the aircraft remains constant.
- 6) The derivative of a vector, \mathbf{v} , with respect to $\{B\}$ is denoted as $d\mathbf{v}/dt$ and the derivative with respect to $\{U\}$ is denoted as $\dot{\mathbf{v}}$.

In general, the equations of motion for an aircraft may be divided into three parts. The first two parts consist of the equations of motion for any rigid body in space, based on both the linear and angular momentum of the body. The third part is the contribution of aerodynamic, gravitational, and thrust forces to the aircraft's motion.

1. Equations for Linear Motion

Newton's Law is first applied in the {B} frame:

$$\begin{aligned} {}^B F &= m {}^B a \\ &= m {}^B_I R^I \dot{v} \\ &= m {}^B \dot{v}. \end{aligned} \tag{3.1}$$

Coriolis' Theorem is then used to relate the inertial and body accelerations of the aircraft:

$${}^B \dot{v} = \frac{d}{dt} {}^B v + ({}^B \omega_B \times {}^B v). \tag{3.2}$$

Substituting Equation 3.2 into Equation 3.1, the final equation for the sum of the external inertial forces resolved in the {B} frame is:

$${}^B F = m \frac{d}{dt} {}^B v + m ({}^B \omega_B \times {}^B v). \tag{3.3}$$

2. Equations for Angular Rotation

Euler's Law for the conservation of angular momentum is first applied for the {B} frame:

$${}^B \dot{L} = {}^B N, \tag{3.4}$$

where ${}^B L$ is the angular momentum of the aircraft with respect to the {B} frame and ${}^B N$ is the total moment applied to the aircraft with respect to the {B} frame. Coriolis' Theorem is then used again to expand the derivative of ${}^B L$:

$${}^B \dot{L} = \frac{d}{dt} {}^B L + ({}^B \omega_B \times {}^B L). \tag{3.5}$$

${}^B L$ can be defined as the product of the aircraft's inertia tensor, J_B , and the aircraft's angular velocity, ${}^B \omega_B$. By substituting this definition into Equation 3.5, the result is:

$${}^B \dot{L} = \frac{d}{dt} (J_B {}^B \omega_B) + ({}^B \omega_B \times J_B {}^B \omega_B),$$

or,

$${}^B N = \frac{d}{dt} (J_B {}^B \omega_B) + ({}^B \omega_B \times J_B {}^B \omega_B). \quad (3.6)$$

3. Equations for External Forces and Moments

The external forces and moments acting on the aircraft can be grouped as:

$$\begin{bmatrix} {}^B F \\ {}^B N \end{bmatrix} = \begin{bmatrix} {}^B F_{gravity} + {}^B F_{propulsion} + {}^B F_{aerodynamic} \\ {}^B N_{propulsion} + {}^B N_{aerodynamic} \end{bmatrix} \quad (3.7)$$

The gravitational force is expressed as:

$${}^I F_{gravity} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix},$$

or,

$${}^B F_{gravity} = {}^B R^I F_{gravity}. \quad (3.8)$$

The moment due to propulsion is eliminated by assuming centerline thrust, such that

${}^B N_{prop} = 0$. Therefore, the propulsive force is expressed as:

$${}^B F_{prop} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} \delta_{thr}, \quad (3.9)$$

where δ_{thr} is the throttle control input.

In implementing the dynamic model in a state-space form suitable for numerical simulation, the aerodynamic forces and moments can be combined as:

$$\begin{bmatrix} {}^B F_{aero} \\ {}^B N_{aero} \end{bmatrix} = \bar{q} \bar{S} \begin{bmatrix} {}^B R & 0 \\ 0 & {}^B R \end{bmatrix} \left(C_{F_0} + \frac{\partial C}{\partial x'} M' x + \frac{\partial C}{\partial \dot{x}'} \dot{M}' \dot{x} + \frac{\partial C}{\partial \Delta} \Delta \right), \quad (3.10)$$

where, C_{F_0} , $\frac{\partial C}{\partial x'}$, and $\frac{\partial \dot{C}}{\partial \dot{x}'}$

are the stability derivative matrices, and,

$$\frac{\partial C}{\partial \Delta}$$

is the control derivative matrix, with the control inputs $\Delta = [\delta_e, \delta_r, \delta_a]$.

M' , \bar{q} , and \bar{S} are matrices used to dimensionalize the stability and control derivatives,

and convert the state vector, x , to x' :

$$x = [u \ v \ w \ p \ q \ r]^T,$$

$$\bar{q} = \text{dynamic pressure},$$

$$\bar{S} = \text{diag}[-S, S, -S, Sb, Sc, Sb],$$

$$x' = M'x,$$

$$M' = \text{diag}[1/V_T, 1/V_T, 1/V_T, b/2V_T, c/2V_T, b/2V_T],$$

and,

$$\dot{x}' = \dot{M}'\dot{x},$$

$$\dot{M}' = \text{diag}[0, b/2V_T, c/2V_T, 0, 0, 0].$$

Substituting the expressions of the forces and moments above, Equation 3.7 may be expanded using the Taylor series, and simplified by combining terms to yield:

$$\begin{aligned} \frac{d}{dt} \begin{bmatrix} {}^B v_{cg} \\ {}^B \omega_B \end{bmatrix} = \chi^{-1} \left\{ \left(\begin{bmatrix} -{}^B \omega_B \times & 0 \\ 0 & -{}^B J_b^{-1} ({}^B \omega_B \times {}^B J_B {}^B \omega_B) \end{bmatrix} + M_I^{-1} {}^B T {}^B \bar{q} \bar{S} \frac{\partial C_f}{\partial x'} M' \right) \begin{bmatrix} {}^B v_{cg} \\ {}^B \omega_B \end{bmatrix} + \right. \\ \left. M_I^{-1} \left(\begin{bmatrix} {}^B F_{grav} \\ 0 \end{bmatrix} + \begin{bmatrix} {}^B F_{prop} \\ 0 \end{bmatrix} \delta_{thr} + {}^B T {}^B \bar{q} \bar{S} \left(C_{F_0} + \frac{\partial C_f}{\partial \Delta} \Delta \right) \right) \right\}, \quad (3.11) \end{aligned}$$

where:

$${}^B T = \begin{bmatrix} {}^B R & 0 \\ 0 & {}^B R \end{bmatrix}$$

is the rotation matrix from the wind frame, {W}, to the {B} frame, and:

$$M_I = \begin{bmatrix} m & 0 \\ 0 & {}^B J_B \end{bmatrix}$$

and,

$$\chi = I_6 - M_I^{-1} {}^B T {}^B \bar{q} \bar{S} \frac{\partial C_F}{\partial \dot{x}'} \dot{M}'.$$

Equation 3.11 expresses the derivative of the first six states for the nonlinear model in matrix form. It is solved in the user defined MATLAB routine, *dstate2.m*. In this thesis, the physical aircraft parameters that are called from within the function *dstate2.m* are those specific to *Bluebird*, and are stored in a MATLAB file, *Bluedat.m*. Changing the model of the air vehicle simply requires changing the constants in *Bluedat.m*.

The Euler angles were added next as three additional states of the model, and were obtained from the equation:

$$\dot{\Lambda} = \begin{bmatrix} 1 & \sin \Phi \tan \Theta & \cos \Phi \tan \Theta \\ 0 & \cos \Phi & -\sin \Phi \\ 0 & \sin \Phi \sec \Theta & \cos \Phi \sec \Theta \end{bmatrix} {}^B \omega_B \quad (3.12)$$

where,

$$\Lambda = \begin{bmatrix} \Phi \\ \Theta \\ \Psi \end{bmatrix}$$

Equation 3.12 is integrated to obtain the time histories of the Euler angles.

Finally, the inertial position of the aircraft c.g. is computed from:

$${}^U \dot{P} = {}^U v = {}^U R {}^B v. \quad (3.13)$$

Equation 3.13 is integrated to obtain the time history of the position of the aircraft in the tangent plane .

The final nonlinear aircraft model contains twelve states: [u, v, w, p, q, r, ϕ , θ , ψ , x_{Pos} , y_{Pos} , z_{Pos}], where,

u, v, w = inertial velocity of the aircraft resolved in the body-fixed frame,

p, q, r = angular velocity of the aircraft wrt {U} resolved in {B},

ϕ , θ , ψ = Euler angles, and

x_{pos} , y_{pos} , z_{pos} = position of the aircraft.

The SIMULINK diagram of the nonlinear model is shown in Figure 3.1. The diagram includes an accelerometer block, which will be necessary for providing acceleration inputs to the IMU/INS system. The accelerations are calculated from the EOM model using the following equation:

$${}^B A = \frac{d}{dt} {}^B v + ({}^B \omega_B \times {}^B v) - {}^B R {}^U g \quad (3.14)$$

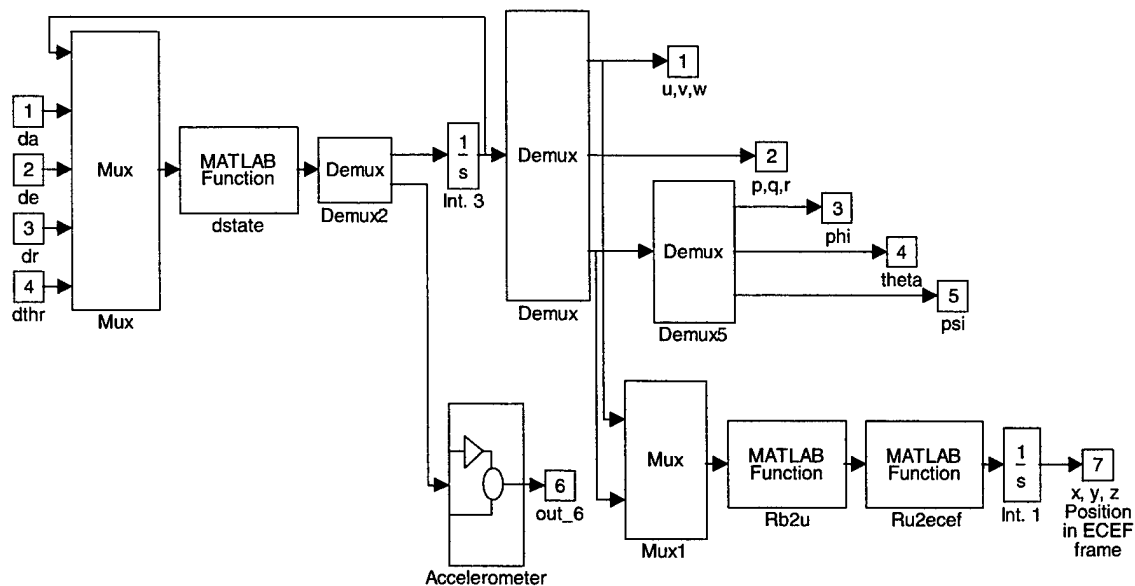


Figure 3.1. Equations of Motion Model for UAV

B. CONTROLLER DESIGN

The suggested TBM counterforce plan requires the lethal UAV to fly a specific pattern and target intercept. Therefore, a nonlinear dynamic controller is required to provide trajectory tracking for the model in the local tangent plane. The Linear Quadratic Regulator (LQR) method was selected for designing the controller. A very detailed treatment of the LQR design process for *Bluebird* is in Reference [6].

The standard LQR problem is to find a controller such that the feedback system in Figure 3.2 is internally stable, and a defined cost function, J , is minimized.

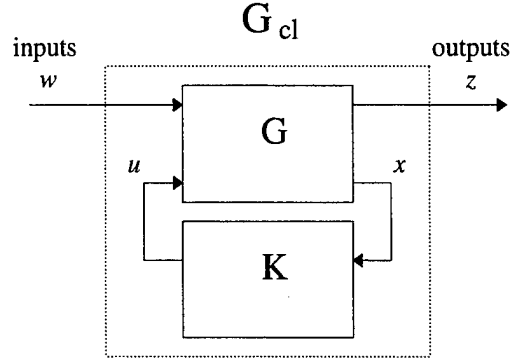


Figure 3.2: Standard LQR Feedback Configuration

The linear system, G , may be expressed as:

$$G = \begin{cases} \dot{x} = Ax + Bu \\ z = C_1x + D_1u \\ y = x \end{cases}, \quad (3.15)$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $z \in \mathbb{R}^p$.

The cost function is defined as:

$$J = \int (z^T z) dt = \int (x^T C_1^T C_1 x + u^T D_1^T D_1 u) dt, \quad (3.16)$$

where $C_1^T C_1 = Q \geq 0$, and $D_1^T D_1 = R > 0$. Q and R are called the weighting matrices.

The gain for the feedback system can be found from the equation:

$$K = -R^{-1}B^T P, \quad (3.17)$$

where P is the solution of the Algebraic Ricatti Equation:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0. \quad (3.18)$$

Using manipulations involving the system and the Hamiltonian matrix, H , it can be shown that stable eigenvalues of H are eigenvalues of the closed loop system and their unstable reflections about the imaginary axis, where:

$$H = \begin{bmatrix} A & -BR^{-1}B^T \\ Q & -A^T \end{bmatrix} \text{ [Ref. 7].}$$

Thus, the LQR controller exhibits asymptotic properties which may be useful in the design process. That is, by varying the weighting on Q and R , root locus analysis indicates that the command and control loop bandwidths may be manipulated to meet design requirements [Ref. 7].

The LQR controller in this case was required to meet the following design specifications:

- 1) Zero steady state errors for all three states, $[x_{pos}, y_{pos}, z_{pos}]$, in response to step inputs.
- 2) Command loop bandwidth in any channel should be no greater than 1 radian per second, and no less than 1/10 radian per second.
- 3) Control loop bandwidth should not exceed 12 radians per second for elevator and aileron actuators, and 5 radians per second for the throttle actuator.
- 4) The dominant closed loop eigenvalues should have a damping ratio of at least 0.7.

The first step in designing the linear controller was to construct a synthesis model as a means of interfacing between the control design and the LQR algorithm. A

linearized model of the aircraft was used in this model. It was necessary to place only a single integrator on each of the three output states to meet the zero steady-state error requirement in response to step commands.

Before the synthesis model could be used, the aircraft model had to be modified. As stated previously, the aircraft has four control inputs. However, the controller can use only three of these inputs to control the three states, $[x_{Pos}, y_{Pos}, z_{Pos}]$. The throttle and elevator are the only means to control the x and z position states, respectively. The y position state, however, can be controlled by either the rudder or the ailerons. It was decided to use the ailerons to bank the aircraft to turn, and the rudder for turn coordination only. Therefore, the rudder control was removed as an input to the controller and replaced with a yaw damper. The yaw damper consists of the yaw rate, r , fed back to the rudder input through a constant gain. The SIMULINK diagram of the synthesis model with the yaw damper incorporated is shown in Figure 3.3.

Designing the LQR controller involved the following iterative process:

1. Select target zeros and linearize the synthesis model to obtain "synthesized" A and B matrices.
2. Initially set the weighting matrices, Q and R, to I (Identity matrix).
3. Calculate the feedback gain matrix, K.
4. Evaluate command and control loop bandwidths and adjust the weighting matrices until the required bandwidths are obtained.

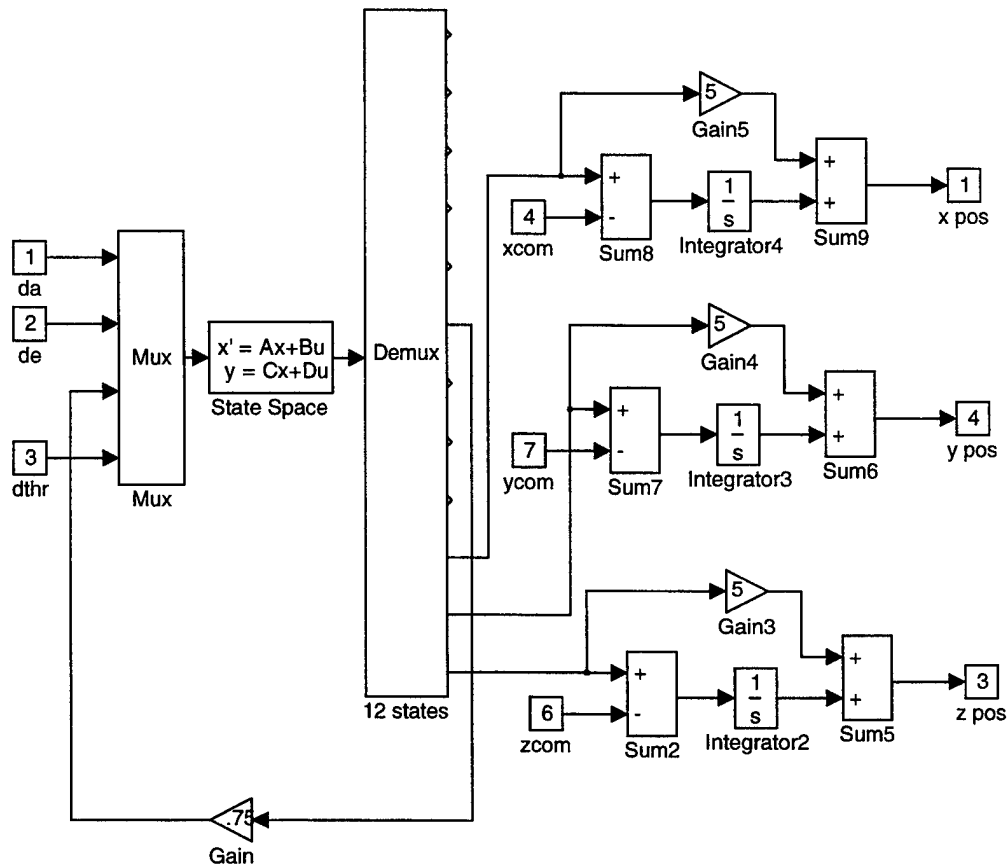


Figure 3.3. Synthesis Model of Aircraft

5. Connect the linear controller to the linear aircraft model and evaluate its performance.
6. Replace the linear controller with a nonlinear version using the same matrix gain, K , and confirm its performance.
7. Replace the linear aircraft model with the full nonlinear aircraft model and confirm that it meets design requirements.

If the bandwidth requirements can not be met by adjusting the Q and R matrices

alone, the target zeros are repositioned, and the entire process repeated. The SIMULINK diagram of the nonlinear model and nonlinear controller is shown in Figure 3.4.

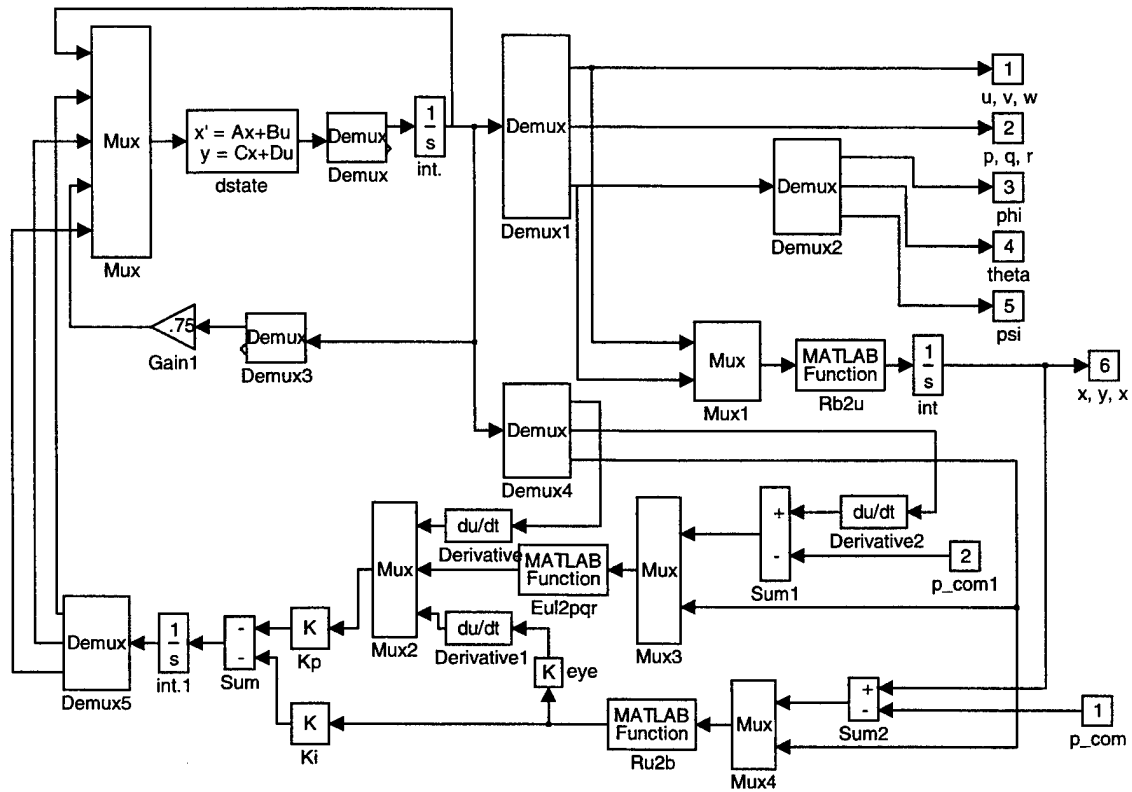


Figure 3.4: Nonlinear UAV Model with Nonlinear Controller

The Bode gain plots of the final command and control loop responses are shown in Figure 3.5, in the left and right columns respectively.

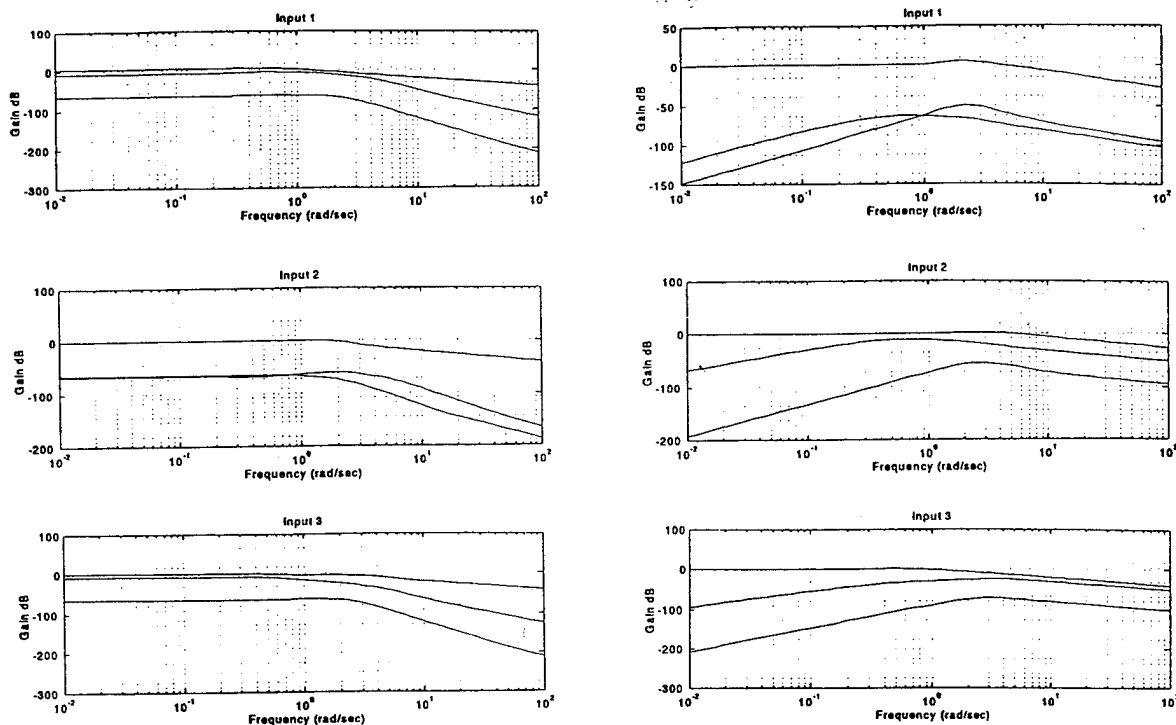


Figure 3.5. Command Loop (left column) and Control Loop Bandwidths (right column)

C. TRAJECTORY GENERATOR

The LQR controller developed in the previous section controls the position of the aircraft in the tangent plane. In essence, it drives the aircraft to commanded x , y , and z coordinates. Therefore, the next step is to design a trajectory generator which will provide x , y , and z position commands to the controller.

The trajectory generator consists of velocity (u), flight path angle (θ), and turn rate (r) in the $\{U\}$ frame. The turn rate is integrated to provide heading (ψ), and the resulting velocity vector is then rotated to the $\{U\}$ frame. This vector generates a series of points, each of which provides an x , y , and z position command to the controller. The

SIMULINK diagram of the basic trajectory generator is shown in Figure 3.6.

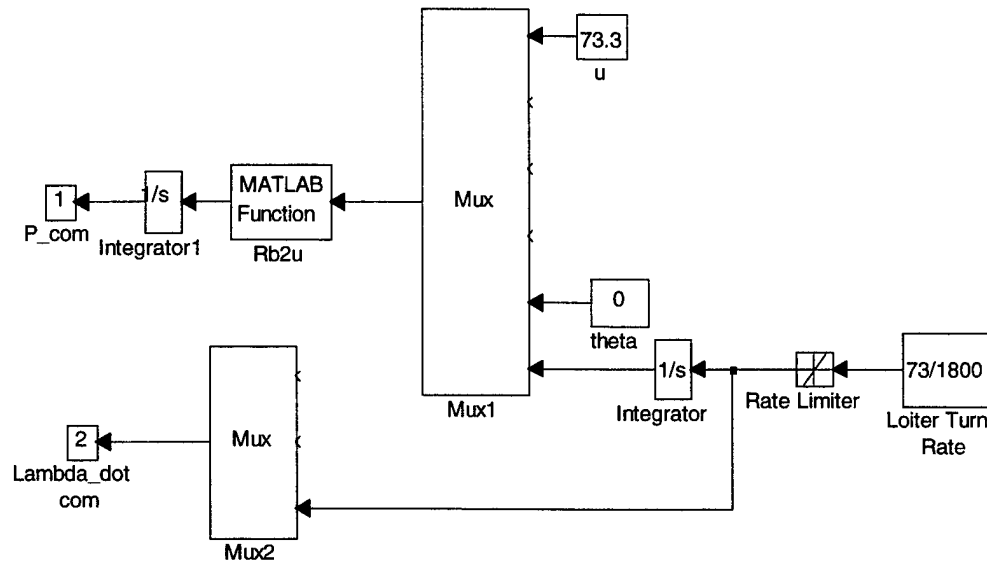


Figure 3.6. Basic Trajectory Generator

The trajectory generator for use in the lethal UAV, however, must be modified beyond the basic form in order to perform the TBM counterforce mission. The design is more complex, in that it must be capable of the following:

- 1) maintaining the aircraft in a circular orbit until a target is detected,
- 2) upon target detection, turn the aircraft at a rate of 0.1 radian/second until pointed at the target, and
- 3) fly the aircraft in a straight-line path towards the target until missile launch.

The SIMULINK diagram of this trajectory generator is shown in Figure 3.7.

the UAV heading, y . When the two are equal (within a set tolerance) the *On Hdg* switch is triggered. With this switch triggered, the turn rate is set to zero, and the target bearing (which is now relatively constant) is used as a heading command.

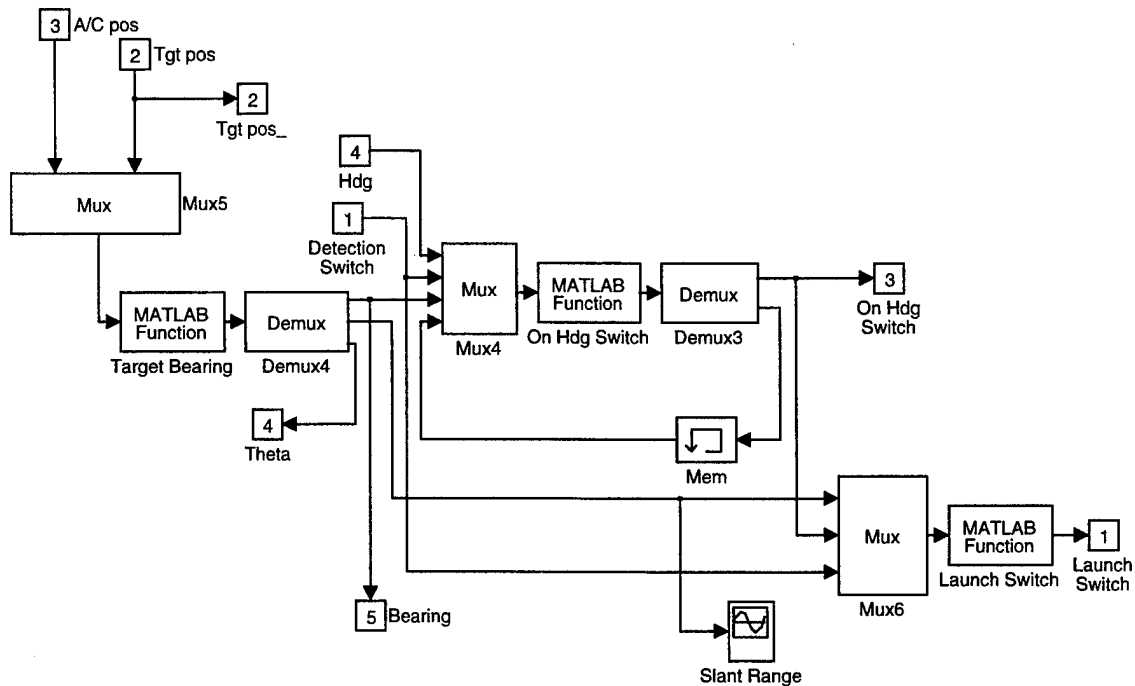


Figure 3.8. Onboard Computer

If necessary, the depression angle of the sensor may be used to point the UAV downward towards the target just prior to missile launch. In the setups simulated to date, the missile has been able to guide toward the target when launched from the aircraft in level flight. If the missile is launched from altitudes above 10,000 feet, a pre-launch dive command may be required. This could be accomplished by triggering a *dive* switch with

the same signal that triggers the *On Hdg* switch. It should be noted though, that dive angles of greater than 0.3 radians may cause the trajectory controller to become unstable due to the aircraft speed exceeding the controller limits.

D. INTEGRATED INERTIAL NAVIGATION SYSTEM (INS)/GLOBAL POSITIONING SYSTEM (GPS)

The INS/GPS navigation system is designed to accept both GPS pseudoranges and INS accelerometer signals as inputs, and produce estimates of the aircraft's position in the North-East-up (NEU) tangent plane as outputs. The navigation system is divided into three distinct subsystems:

- 1) the INS model in the tangent plane coordinate system,
- 2) the GPS receiver model, and the four pseudoranges it uses to obtain a position, and
- 3) the Kalman filter with bias rejection for integrating the INS and GPS subsystems.

1. The INS Model

Three fundamental components are identifiable in the design process for the INS model. First is a SIMULINK model of the tangent plane navigation system. Second are the accelerometer inputs to be fed into the model. The final component consists of the outputs of both the air vehicle's position in the tangent plane (NEU), and the equivalent satellite pseudoranges which are compared with GPS pseudoranges by the Kalman filter.

A tangent plane navigation system is one that can be used for flight within a limited region. It is assumed that within the tangent plane Newton's laws apply, and the vector from the center of the earth to the aircraft is essentially the same in magnitude as the vector from the center of the earth to the origin of the tangent plane. For this model the earth is assumed to be spherical. The tangent plane navigation system is the same as a space-stable configuration. In this configuration, the inertial platform maintains a constant orientation with respect to the inertia space, which, in this case, is the tangent plane. Moreover, in this type of system the platform frame coincides with the inertial frame. [Ref. 7]

The primary inputs into the space-stable navigation system are accelerations sensed in the inertial reference coordinates. The acceleration due to vehicle motion with respect to inertial space is integrated twice in inertial reference coordinates. Therefore, the output of the first integrator is the inertial relative velocity, and that of the second integrator is the position relative to the center of the earth in Cartesian coordinates. The torquing for this tangent plane, space-stable mechanization is such that the platform maintains a constant orientation with respect to the tangent plane.

The following are the governing equations for the tangent plane system:

$$\frac{d^2 \bar{R}}{dt^2} = \bar{A} + Gravity, \quad (3.19)$$

$$Gravity = \frac{\mu}{norm(R)^2} \frac{(-R)}{norm(R)}, \quad (3.20)$$

$$R = R_0 + R_t. \quad (3.21)$$

Note that R is the position vector from the center of the earth to the vehicle, R_0 is the position vector from the center of the earth to the center of the tangent plane, and R_t is the position vector from the center of the tangent plane to the UAV.

From these equations, the tangent plane model was developed, and is shown in Figure 3.9.

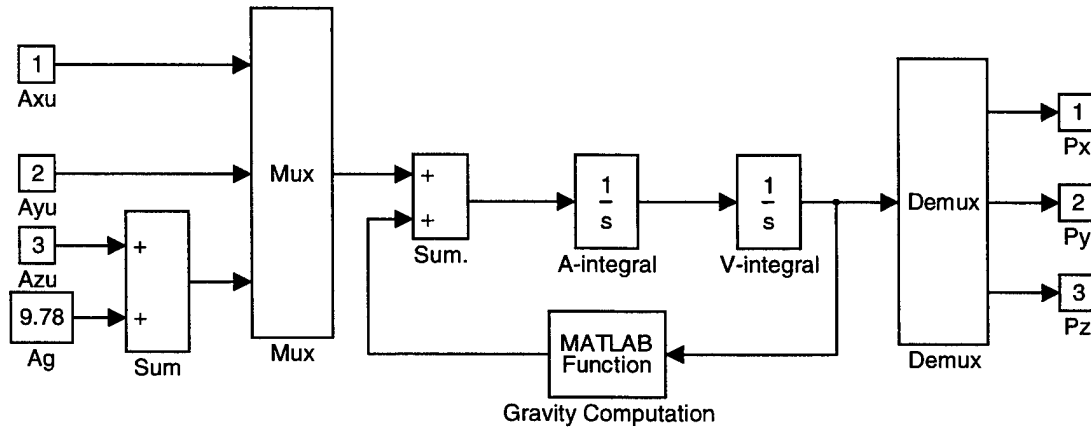


Figure 3.9. Tangent Plane Navigation Model

When linearized, the model produced the eigenvalues listed in Table 3.1. The eigenvalues indicate that four of the poles are on the imaginary axis, and will give rise to a sinusoidal motion at the Schuler frequency. One of the other poles is in the right half plane, and will cause instability in the vertical channel.

The MATLAB routine, *ranges.m*, is used to convert position in the tangent plane to GPS ranges. These ranges are converted to pseudoranges by adding the clock noise bias and drift. The outputs of the navigation system then are the four GPS pseudoranges. The eigenvalues of this model are the same as those of the previous one, with additional poles at zero due to the bias and drift integrals for the clock noise error. The eigenvalues are listed in Table 3.2.

Eigenvalues	Damping	Frequency (rad/sec)
$0 + 0.0012i$	0.0000	0.0012
$0 - 0.0012i$	0.0000	0.0012
0.0018	-1.0000	0.0018
-0.0018	1.0000	0.0018
$0 + 0.0012i$	0.0000	0.0012
$0 - 0.0012i$	0.0000	0.0012
0	-1.0000	0
0	-1.0000	0

Table 3.2. Eigenvalues of UAV Model with Pseudorange Outputs

2. The GPS Receiver and Pseudorange Models

GPS ranges in the system are modeled in the MATLAB function *range.m*. This function takes the actual positions of the vehicle and the four satellites in the ECEF

coordinate frame and calculates an exact range. This exact range is then corrupted by two sources or error: selective availability (SA) noise and clock error, both of which are generated using white noise sources. The result is not an exact range, but a "pseudorange" which is sent to the GPS receiver, and calculated using Equation 3.20.

$$R = \sqrt{(X_{sat} - X_{rec})^2 + (Y_{sat} - Y_{rec})^2 + (Z_{sat} - Z_{rec})^2} + c \Delta t_{total} \quad (3.20)$$

The SA noise runs through a low pass filter with a time constant of 180 seconds, with each satellite's noise source having its own seed to assure independence. The resultant nominal SA noise for the four satellites is then run through a gain block to give an average range error of 20 meters to each of the four satellites.

The clock error is divided into bias and drift, both of which are generated from the same noise sources. As with the SA noise, each satellite has its own noise source with different seeds to ensure independence. One set of noise signals enters a gain block directly from an integrator which increases the average magnitude of the signal to 0.3 meters. This bias quantity was arrived at by multiplying the speed of light by 1 nanosecond, with 1 nanosecond representing the order of a real system's clock error. The other noise signals also pass through two integrators. These integrators convert the noise from a constant to a drift. The gain of 0.003 on the drift represents an average drift of 0.01 nanosecond per second, or 36 nanoseconds per hour.

Since the vehicle position is not stationary one, two integrators were used to establish the its velocity and position. One integrator was used to model the satellite positions. Figure 3.11 is a SIMULINK block diagram of the GPS receiver model.

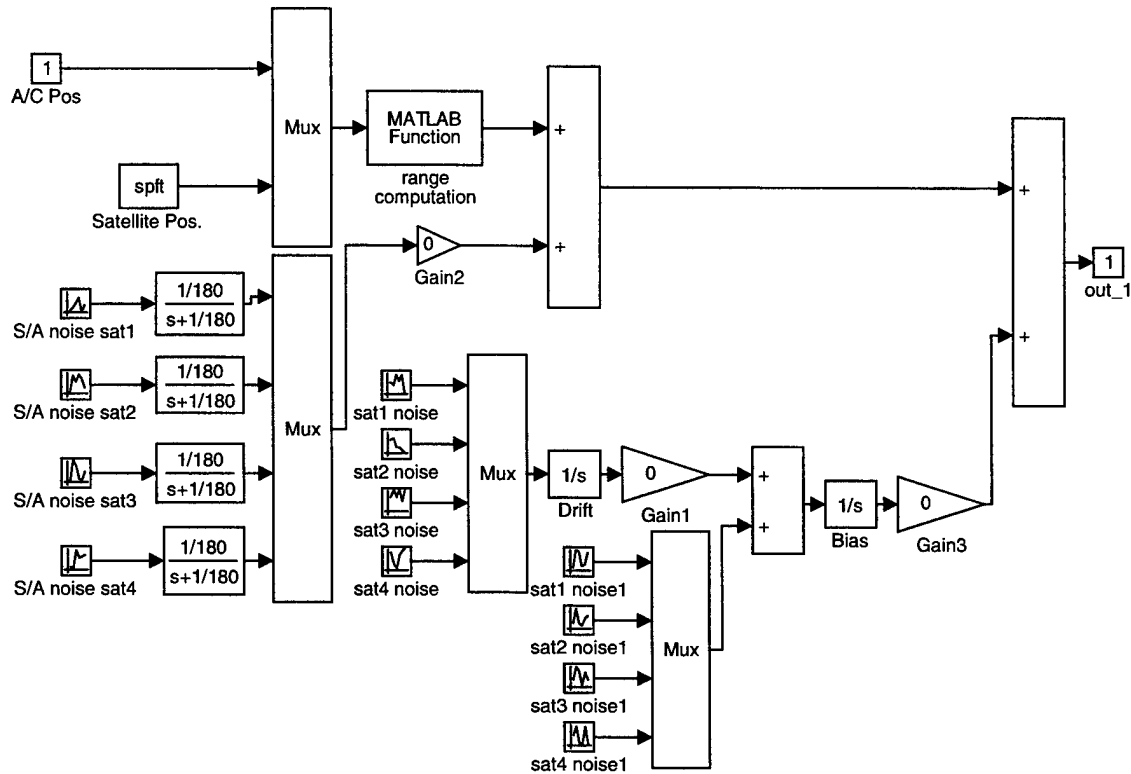


Figure 3.11. GPS Receiver Model

3. Kalman Filter Integration of INS and GPS

Kalman filtering is used along the linear and angular position channels to provide optimal estimates of the states for the controller. Two primary requirements exist for the Kalman filter integration of the INS/GPS. First, the filter should use pseudorange estimates from both the INS and GPS models to compute estimates of the vehicle's

position in the tangent plane. Second, the filter should reject biases in accelerometers, and high frequency noise in the GPS receiver.

The Kalman filter was designed using the following steps:

- 1) The applicable navigation equations were obtained:

$$\begin{aligned}\dot{x} &= f(x) + u \\ y &= [X_t, Y_t, Z_t].\end{aligned}$$

- 2) The navigation equations were linearized to obtain the error model:

$$\begin{aligned}\delta \dot{x} &= A \delta x + B \delta u \\ \delta y &= \delta [X_t, Y_t, Z_t].\end{aligned}$$

- 3) The actual Kalman Filter was designed using linear quadratic estimator (LQE) techniques.

- 4) The filter was implemented on the nonlinear equations.

- 5) The implementation was verified by linearizing the filter design.

The actual Kalman Filter design technique, alluded to in step three, is the dual process to the LQR design method described previously. For a linear system described by:

$$\begin{aligned}\dot{x} &= Ax + Bu + Gw \\ z &= Cx + v,\end{aligned}$$

where, v and w are zero mean white noise with respective power spectral densities of V and W , the Kalman filter that produces an optimal estimate of x is:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(z - C\hat{x}). \quad (3.21)$$

The Kalman gain, L , is:

$$L = YC^T V^{-1}, \quad (3.22)$$

where Y solves Equation 3.18, the Algebraic Riccati Equation. [Ref. 7]

The LQE process exhibits asymptotic properties, which are the dual of those involved in the LQR method. Therefore, by appending integrators, and associated target zeros, the filter performance may be adjusted to meet design requirements.

In proceeding with the Kalman filter design, a synthesis model was developed. This model, shown in Figure 3.12, uses the linearized error equations for the tangent plane navigation model in Figure 3.10.

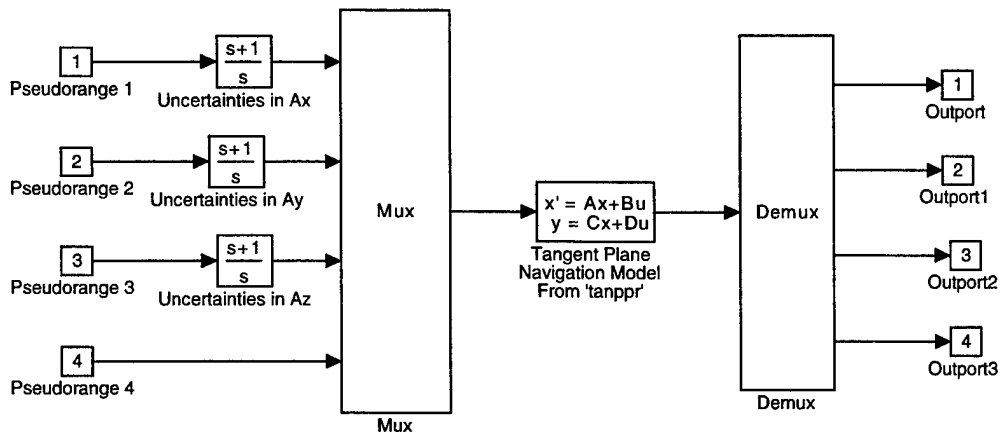


Figure 3.12. Kalman Filter Synthesis Model

Bias uncertainties in the x , y , z accelerometers are represented as integrators. The numerators of the integrators represent zero locations chosen to provide the desired bandwidth for operation. It was determined that the optimum bandwidth, from

pseudorange input to estimated pseudorange output, should be one rad/sec. This would ensure that GPS inputs will be used only at frequencies up to one rad/sec (the maximum update rate of most GPS receivers is one second). Additionally, high frequency noise will be eliminated by the filter.

After linearizing the synthesis model, the resulting A and B matrices were used as inputs for the MATLAB function, *lqe*. Additionally, the weightings of the V and W matrices were adjusted and input to produce the desired bandwidth and performance. Thus, the Kalman filter gains were determined, and incorporated into the model as shown in Figure 3.13.

As expected, after linearizing the SIMULINK model in Figure 3.13, the resulting eigenvalues matched those obtained from the expression $(A_{\text{synth}} - L \cdot C_{\text{synth}})$, and are listed in Table 3.3. Also as expected, the estimator gains that were found with the MATLAB function, *lqe*, appear in the B matrix of the system in Figure 3.13.

Bode gain plots of the pseudorange inputs to estimated pseudorange outputs are shown in Figure 3.14. It is seen that each plot exhibits the desired cutoff frequency of approximately one rad/sec.

The final test of the integrated INS/GPS system was a time response analysis comprised of two simulation runs. This analysis confirms that the system operates as designed, and the Kalman filter rejects biases in the accelerometers. Figure 3.15 shows

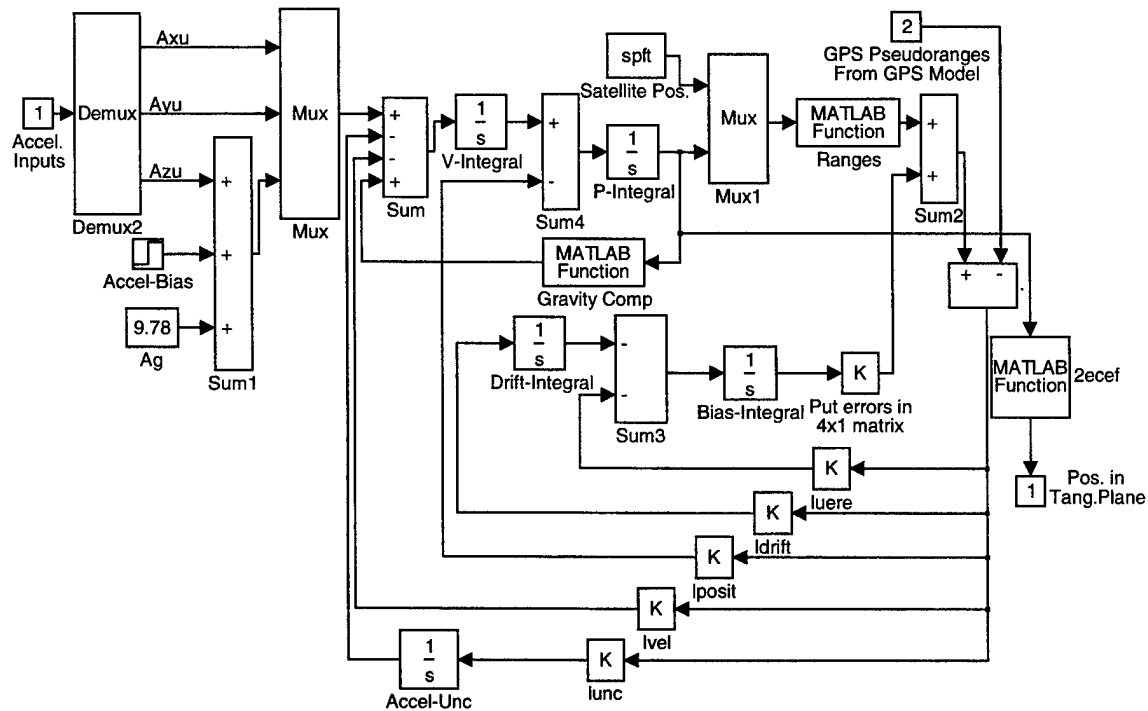


Figure 3.13. Navigation Model and Kalman Filter Integration

the bias rejection for both runs. The two simulations induce Y axis and Z axis acceleration biases of 5 m/s^2 at 5 seconds and 15 seconds respectively. Additionally, in simulation 2, the velocity components $V_y = 25 \text{ m/s}$ and $V_z = 50 \text{ m/s}$ are present. It is seen that high frequency pseudorange errors are rejected as expected.

Eigenvalue	Damping	Frequency (rad/sec)
$-1.0677 + 1.1409i$	0.6833	1.5625
$-1.0677 - 1.1409i$	0.6833	1.5625
$-0.7749 + 0.9878i$	0.6172	1.2555
$-0.7749 - 0.9878i$	0.6172	1.2555
$-0.4524 + 0.6220i$	0.5882	0.7691
$-0.4524 - 0.6220i$	0.5954	0.5856
$-0.3487 - 0.4704i$	0.5954	0.5856
-0.7261	1.0000	0.7261
-0.8763	1.0000	0.8763
-0.5868	1.0000	0.5868

Table 3.3. Eigenvalues of Filtered Navigation Model

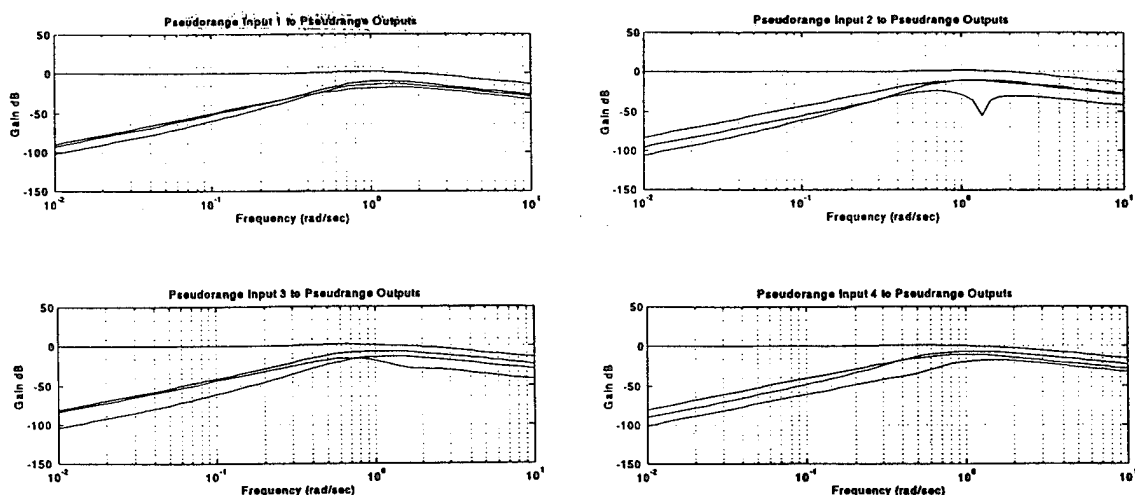


Figure 3.14. Bode Gain Plots of Pseudorange Inputs to Pseudorange Outputs

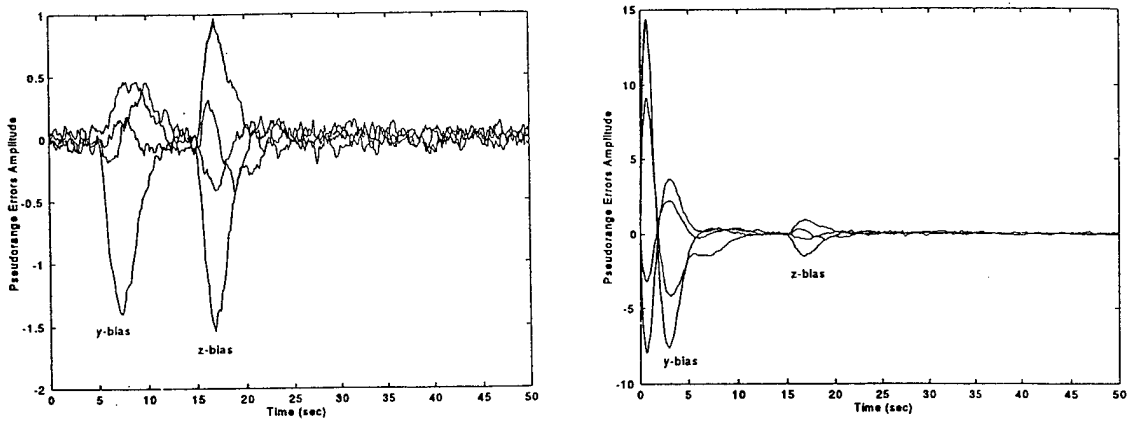


Figure 3.15. Integral Kalman Filter Bias Rejections

IV. MODELING THE GROUND COMPONENTS

The ground components included in the overall TBM counterforce scenario are the transporter-erector-launcher (TEL) and the unattended ground sensors (UGS). Descriptions of each are included in separate sections of this chapter, however, some overlap in the modeling and discussion is inherent.

A. TRANSPORTER-ERECTOR-LAUNCHER (TEL) AND ACOUSTIC SIGNATURE MODELS

In developing the SIMULINK model of the TEL and its acoustic signature, two of the simplifying assumptions mentioned in Chapter II arise. Specifically, it is assumed that the moving TEL emits a known, constant, acoustic frequency; and it travels at a constant heading and speed.

The MATLAB files *count_cycles.m* and *freq.m* (see Appendix), which are used by the UGS model to process the TEL acoustic signal, are capable of handling a variable TEL speed. However, in the interest of reducing processor load, the TEL speed was kept constant. This reduces the necessary MATLAB code, and avoids the requirement of using a smaller simulation step size. These two factors save considerable simulation time.

With the velocity vector of the TEL remaining constant, the vehicle is easily treated as a two-dimensional point mass. The governing equations for this model are:

$$\bar{v} = \int \bar{a} dt \quad (4.1)$$

and,

$$\bar{d} = \int \bar{v} dt, \quad (4.2)$$

where, \bar{a} = vehicle acceleration, \bar{v} = vehicle velocity, and \bar{d} = vehicle distance from an initial position.

Equations 4.1 and 4.2 are implemented in the TEL model as two integrators with initial conditions as shown in Figure 4.1.

The constant frequency acoustic signature generated by the TEL is modeled with a sine wave. Again, the simulation step size influenced the design. A realistic frequency for the TEL acoustic output would require a simulation step size on the order of millisecond. To be consistent with the other model components, a frequency compatible with the desired step size of approximately 0.1 seconds had to be used. In order to produce an adequately smooth sine wave for simulation at this step size, a baseline frequency of 1 Hz was selected. While this value is lower than realistic, the principle of operation remains the same, and does not affect the overall integrity of the simulation. The baseline power of the acoustic signature amplitude is set at 10 Watts. This ensures an intelligible signal at medium ranges.

Having established a baseline amplitude and frequency for the TEL, a method for representing its movement relative to a stationary UGS could be developed. In doing so, a Doppler shift due to the relative motion will be modeled, as well as changes in the amplitude of the acoustic signal as the TEL either approaches or recesses from the UGS. Essentially, either of the two ground sensors may be considered as “listening” points; and the generated TEL signal must accurately exhibit the acoustical characteristics of a vehicle moving relative to the points. To accomplish this, the MATLAB routines

truck_data1.m and *truck_data2.m* use the velocity and position of the TEL in the tangent plane, and output appropriate values for the frequency and amplitude of the acoustic signal at the location of each of the two ground sensors. The velocity information, $V_{relative}$, is used in Equation 4.3 to compute the frequency of the acoustic signature relative to each sensor as follows:

$$\Delta f = 2 \frac{V_{relative}}{\lambda}. \quad (4.3)$$

The TEL position information is used along with the ground sensor position to determine the TEL's range from each sensor. With the TEL range, R , the acoustic signal amplitude measured at the ground sensor may be computed using Equation 4.4, which is based upon the dissipation of a transmitted wave in air:

$$amplitude = \frac{10000}{4\pi R^2}. \quad (4.4)$$

These calculations are repeated continuously so that the Doppler shift and changing amplitude arriving at each UGS represent the moving TEL. The SIMULINK model of the TEL and its acoustic signal generator is shown in Figure 4.1.

B. UNATTENDED GROUND SENSOR (UGS) MODEL

The TBM counterforce scenario consists of two UGS units. Only one, however, is necessarily involved in the short mission demonstrated by the simulation. Although the additional UGS also constantly processes the TEL position, it does not transmit data to

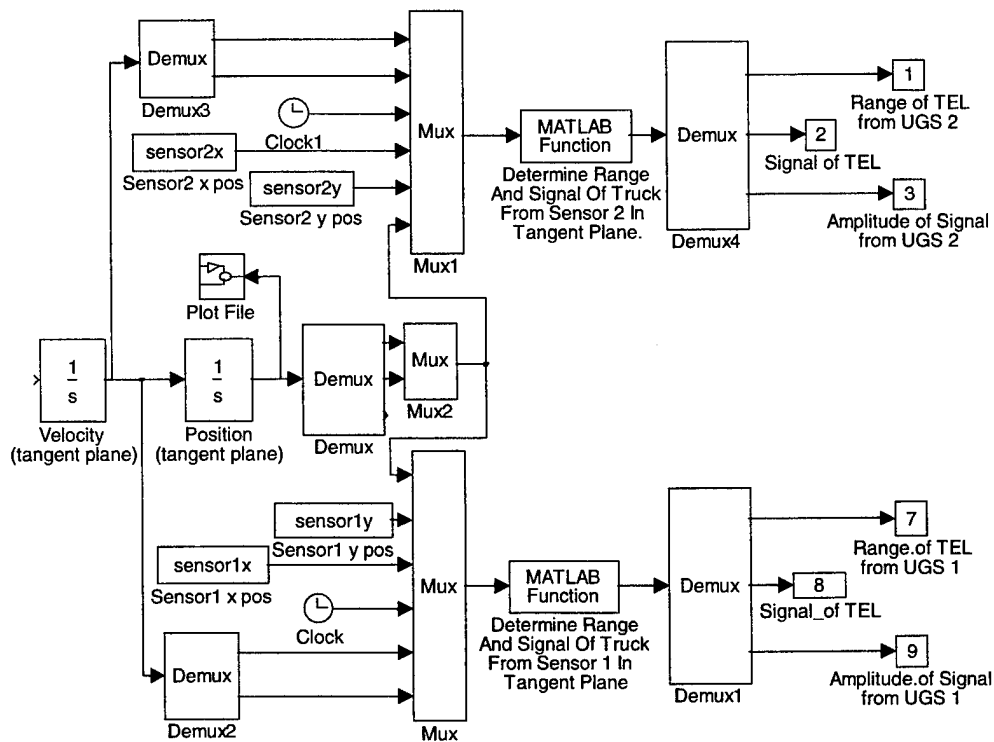


Figure 4.1. TEL and Acoustic Signal Generator

the UAV. It is merely present to indicate that an entire string of sensors would be in place along the road in a real world situation. Their spacing would be such that an acoustic signature from a TEL traveling along the string would remain in range of any of the sensors on the network long enough for the prosecution sequence to complete. For the actual model, it is assumed that the TEL will not exceed the range of the single transmitting UGS in the time required for the UAV to receive a TEL detection signal from the UGS, and subsequently destroy the target. In a common simulation run geometry, the UAV loiters at an altitude of 3,000 feet, and is offset 10,000 feet from the

UGS on the road. With this setup, and a TEL speed of 30 miles per hour, missile impact occurs at the target when it is 5,000 feet beyond the transmitting UGS. Therefore, even with the single UGS simplification, the scenario is not unreasonable. While reducing the processing load on the computer by eliminating the need to model a working UGS network, this simplification does not compromise the validity of the counterforce model.

As discussed in previous chapters, the capabilities of second generation UGS systems are extensive, and would lend themselves well for use in this TBM counterforce plan. However, the design of the UGS model described here resembles that of older systems widely used in the Vietnam War [Ref. 2]. It emulates the basic operating principles of sonobouys, which are used for anti-submarine warfare (ASW). Sonobouys exploit frequency Doppler shifts and amplitude changes to locate and track submarines. Furthermore, they rely on the fact that each class of submarine has a distinct and identifiable acoustic signature.

In the same manner, the UGS is designed to detect a TEL, continuously compute its changing position, and transmit the information to the UAV and missile trajectory controllers. The method for producing the frequency Doppler shift and amplitude variations of the TEL has been described. A discussion of how the UGS processes this information follows.

In each UGS, the frequency of the incoming acoustic wave is determined by a frequency counter as shown in Figure 4.2. The MATLAB routine, *count_cycles.m* (see Appendix), counts the number of zero-crossings, or cycles, of the wave during a specific

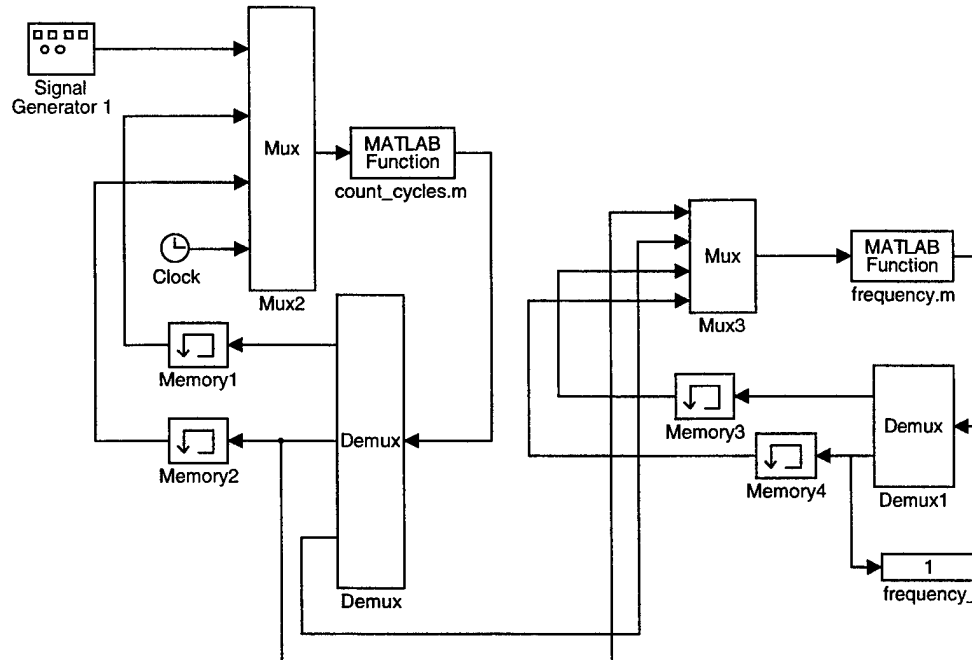


Figure 4.2. Frequency Counter Component of UGS

duration of time. The routine, *frequency.m* (see Appendix), then uses the cycle count to compute the frequency.

As the changing frequency is computed, it is used in Equation 4.3 above, the Doppler equation, to determine the TEL velocity. In addition, received amplitude is used to determine the time when the TEL marks on top of the UGS; which, in this case, is the closest point of approach (CPA) of the TEL to the UGS. The mark-on-top is detected by the UGS 2 when the amplitude of the TEL acoustic signal peaks in magnitude. At this moment the position of the TEL is known to be the same as that of the UGS, and is used

in subsequent position computations. Additionally, the amplitude peak causes the UGS to transmit a detection signal to the orbiting lethal UAV. From that point on, the TEL position solutions will also be transmitted to the UAV and missile for guidance.

The MATLAB routine, *trigger.m* (see Appendix), detects the mark-on-top of the UGS, and calculates the TEL velocity. Since the direction of the TEL is known to be that of the road, the velocity can then be resolved into x and y components by the MATLAB routine, *velocity2d.m* (see Appendix). These velocities are integrated repeatedly, and added to the x and y positions of UGS 2 in the tangent plane. In this way the position of the TEL is tracked.

Figure 4.3 is the SIMULINK block diagram of the integrated TEL, acoustic signal generator, and UGS model. The TEL and acoustic signal generator for simulating its motion are contained within the *Truck Model* block. The amplitude detector is also located in the *Truck Model* block. The frequency counters are contained within the *UGS 1* and *UGS 2* blocks.

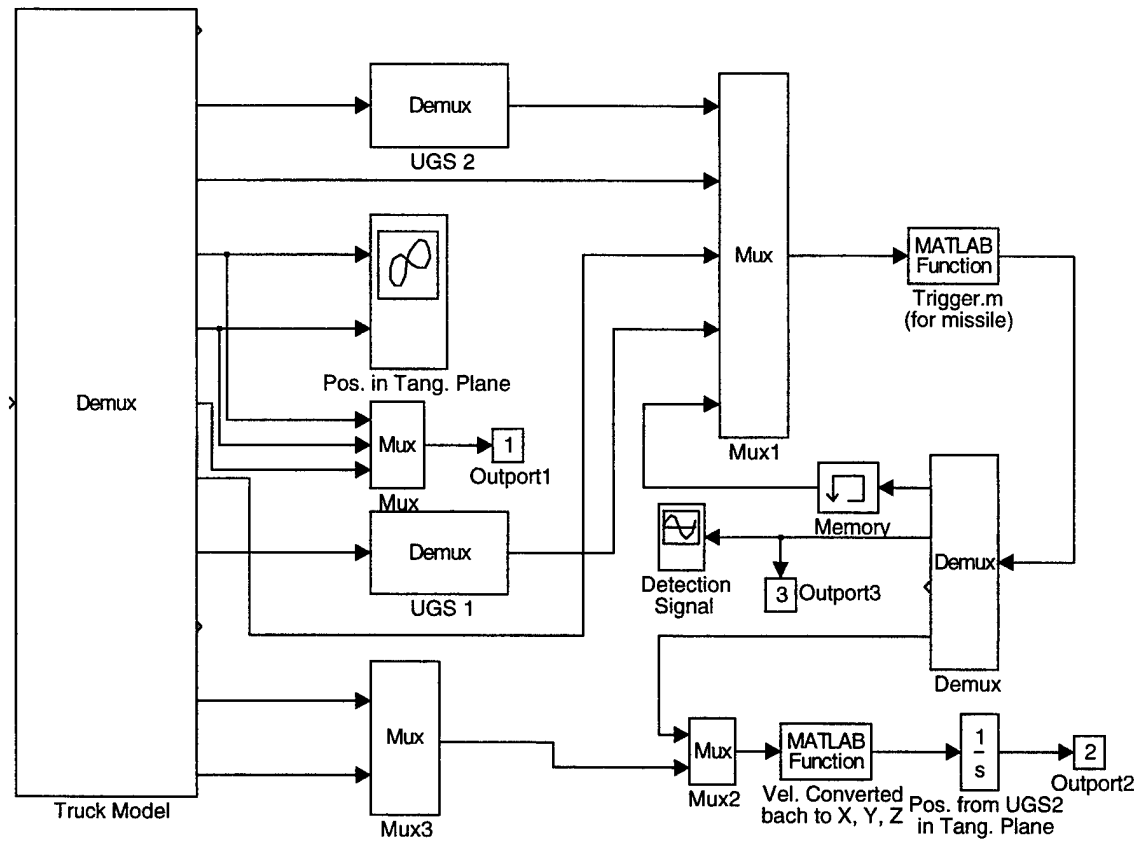


Figure 4.3. Integrated Model of TEL, Acoustic Signal Generator, and UGS

V. UAV MISSILE MODEL

The UAV missile is modeled as a point mass, and is governed by Newton's Second Law, $F = ma$. The forces acting on the missile are to thrust, drag, gravity, and control inputs. The guidance system for the missile consists of a proportional-plus-integral-plus-derivative (PID) line-of-sight (LOS) controller utilizing missile position derived from an onboard GPS, and TEL position received from the transmitting UGS.

The missile's rocket motor has a 10 second burn time represented by a *timer* variable in the MATLAB routine, *missile_eom.m* (see Appendix). When missile launch is commanded, the thrust is set to 2500 pounds, and the timer started. When the timer reaches a 10 seconds, the thrust value is returned to zero.

The main equation computed by *missile_eom.m* is:

$${}^B_a = {}^B\dot{v} = \frac{\text{thrust} + {}^B_R m g + \text{drag} + \text{inputs}}{m}, \quad (5.1)$$

where the *drag* force is a function of velocity; the force of gravity, g , is computed using Equation 3.8; and the control *inputs* are forces are proportional to velocity, and act in the missile y and z body axis directions.

Equation 5.1 is integrated twice and rotated to obtain missile position in the {U} frame. The SIMULINK diagram of the missile model is shown in Figure 5.1.

Prior to the missile launch command, the missile is strapped to the aircraft and must follow the aircraft trajectory. To accommodate this, the aircraft states u, v, w, ϕ, θ ,

as in the UAV model, the missile GPS position and accelerometer outputs are fed through a Kalman filter. The output of the filter is the missile's position in the $\{U\}$ frame. The missile's LOS to the target may then be determined easily using geometry.

After launch, the basic functions of the LOS controller are to compute the missile flight path angles (θ and ψ) and line-of-sight (to the TEL) angles (α and β), and to drive the error between them to zero. The angle errors are referenced to the body axis of the missile. Therefore, when they are maintained at zero the missile is guiding directly towards the target's position. Figure 5.2 illustrates the geometry used for the LOS controller.

The controller design is shown in Figure 5.3. The input commands for the z and y axis channels are computed separately from two sets of gains contained within the block labeled *PID LOS Controller*. The gains are set to achieve a very fast rise time (natural

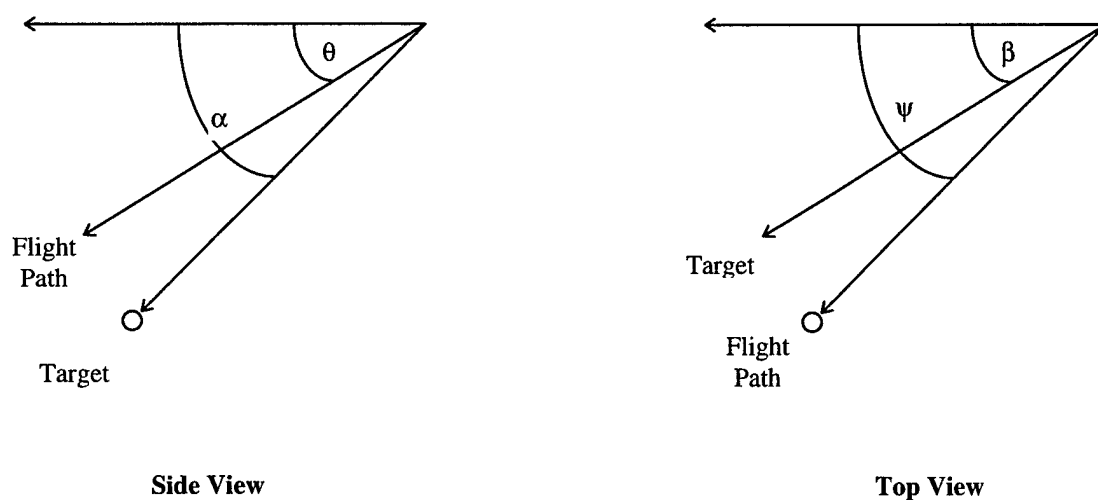


Figure 5.2: Geometry Used for the LOS Controller

frequency, ω_n , of 5.0 rad/sec) with very little overshoot (damping ratio, ξ , of 1.0). Since control inputs can not act on the missile until launch, switches were incorporated to allow only zero inputs to the integrators before the launch command. Without the switches, the integrators receive constantly increasing errors signals; and the controller is unable to react. This results in excessive control inputs when the missile is eventually launched.

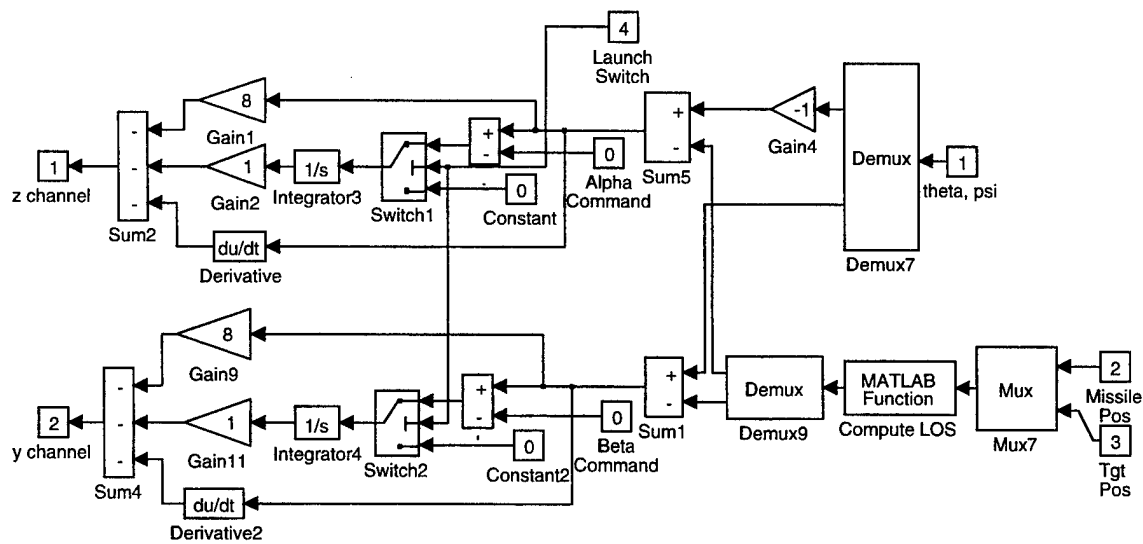


Figure 5.3. Missile PID Controller

The missile model includes a *Stop Simulation* block which terminates the simulation run when the missile impacts. This is accomplished using the MATLAB routine, *missdist.m*, which sends a signal to the *Stop Simulation* block when the missile

and target z coordinates are nearly equal. At that time, the routine also calculates the miss distance by comparing the actual x and y coordinates of the missile and target.

VI. RUNNING THE SIMULATION

The complete SIMULINK model of the lethal UAV TBM counterforce plan is shown in Figure 6.1. It was obtained by connecting the previously described components together; and insuring that correct and compatible data was being transferred between them.

A method for running the simulation has been developed to accommodate the use by a person unfamiliar with the details of the system and its software. The method makes use of the MATLAB routine, *lethal_UAV.m* (see Appendix), which briefly describes the model scenario, prompts the user for the necessary setup inputs, and begins the simulation automatically. The following is a list and description of the steps necessary to begin a simulation:

- 1) Log on to an applicable workstation; and start MATLAB.
- 2) Change to the *TBM* directory.
- 3) Type "lethal_uav" in the MATLAB command window.
- 4) Type in the requested information at the prompts. This will include desired UAV altitude, TEL speed, TEL heading, TEL starting position in two dimensional Cartesian coordinates, and elapsed time until TEL is detected by the UGS. Sample inputs are provided.
- 5) Upon completion of the simulation, a three dimensional plot of the entire run may be displayed by typing "sim_results" in the command window.

During the simulation five displays appear, in the form of MATLAB plots, that allow the user to observe the progress of the run. The information presented in the plots was chosen to best provide the user with an intuitive perspective of the proceeding scenario. Such a perspective is difficult to attain without the benefit of converting the output into picturesque animation. Nonetheless, the plots provide the necessary awareness, and are described below.

Figure 6.2 shows a snapshot in time of the displayed missile track plot. The perspective of the track is from above, looking down. Before the missile launch, the plot also represents the UAV track. At the instant captured in Figure 6.2, the UAV has begun its counterclockwise loiter pattern at the origin in the XY plane; and is awaiting a detection signal from the UGS. At detection, the track will change as the UAV begins a two dimensional intercept (constant altitude) to the TEL. Missile launch will be indicated on this plot by a sudden and significant increase in speed of the intercept.

Figure 6.3 shows a snapshot in time of the TEL ground track plot. In this case, the starting coordinates were input as $(-15000,0)$.

Figure 6.4 shows the simulation display of the detection signal transmitted by the UGS to the lethal UAV. The signal changes from a constant zero to a constant one when the TEL marks on top of the UGS position. This is the cue for the UAV to detach from its loiter pattern and fly a two dimensional (constant altitude) intercept to the TEL. The missile will be launched from the UAV when its slant range to the TEL decreases to 12000 feet.

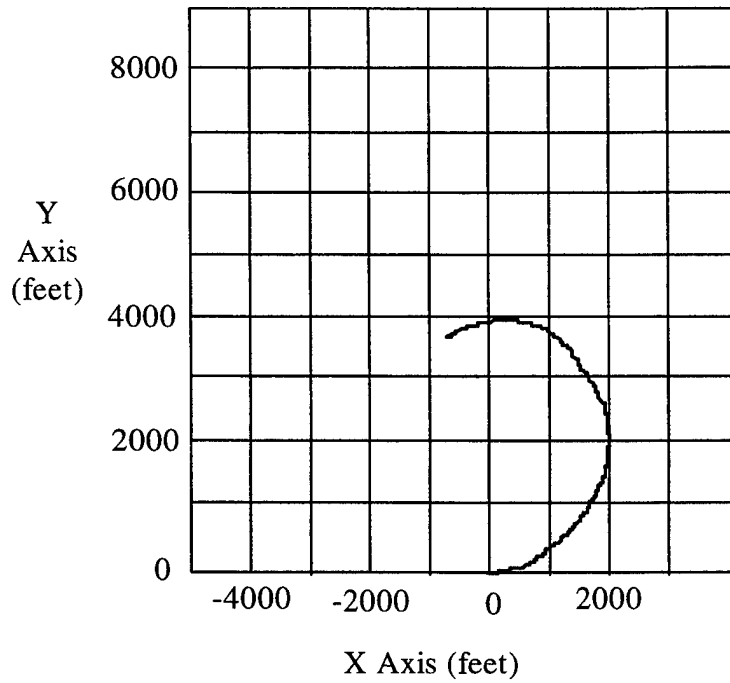


Figure 6.2. Simulation Display of Missile Track Prior to Launch From UAV (Looking Down Perspective)

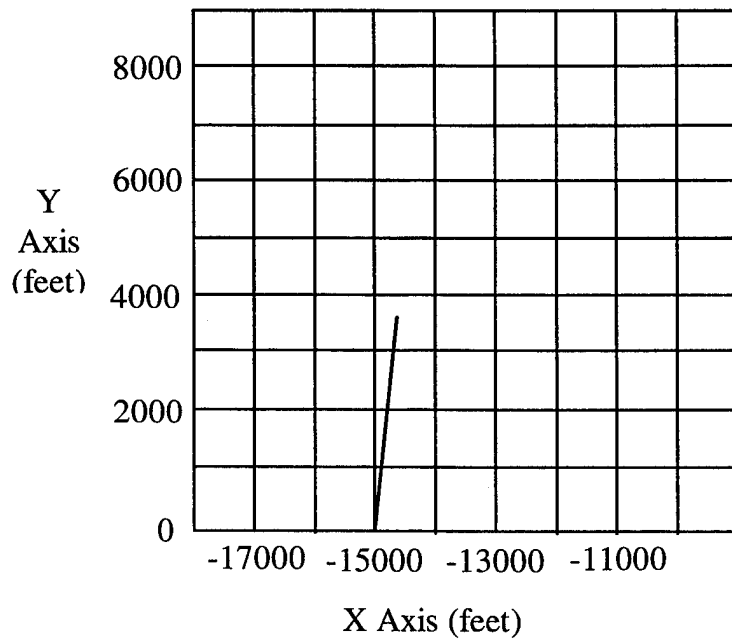


Figure 6.3. Simulation Display of TEL Ground Track

Figure 6.5 is a snapshot in time of the missile-to-TEL slant range plot. This is useful for anticipating the missile launch. In this case, at 190 seconds of elapsed simulation time, the slant range is nearly 12500 feet, indicating missile launch in approximately five seconds. At launch, the slant range plot will suddenly begin a rapid decrease towards zero feet.

Figure 6.6 is a snapshot in time of the missile altitude plot. Missile launch is indicated at the moment the altitude plot shows a rapid descent from the input UAV altitude.

Perhaps the most useful graph is the three dimensional plot of the entire simulation, which may be obtained after the run is completed. Running the MATLAB

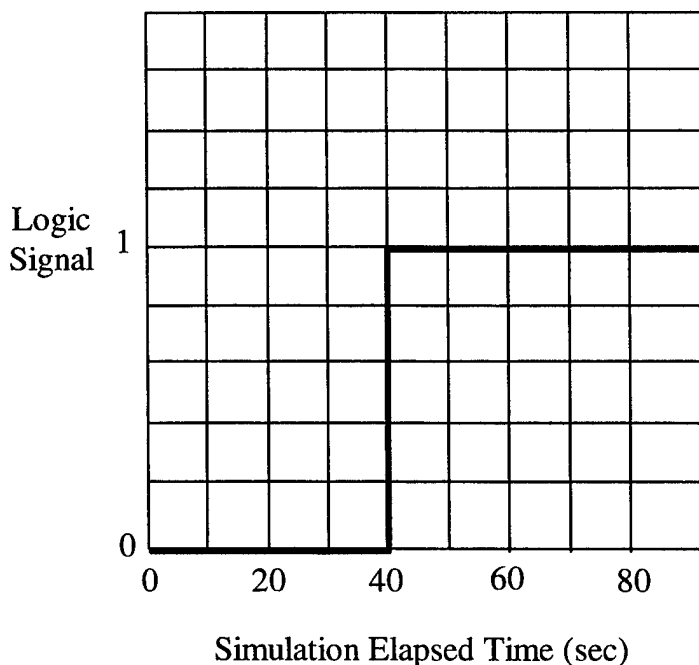


Figure 6.4. Simulation Display of UGS Detection (of TEL) Signal

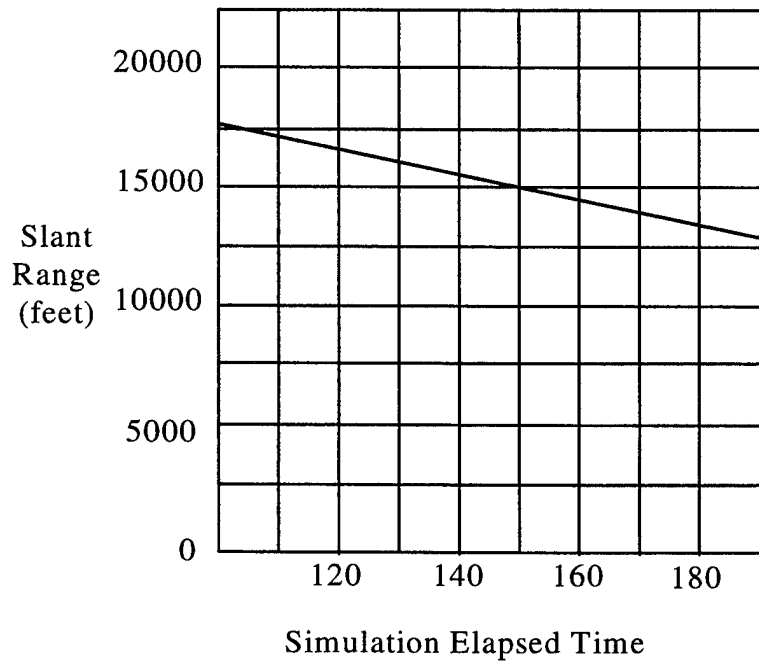


Figure 6.5. Simulation Display of Missile-to-TEL Slant Range

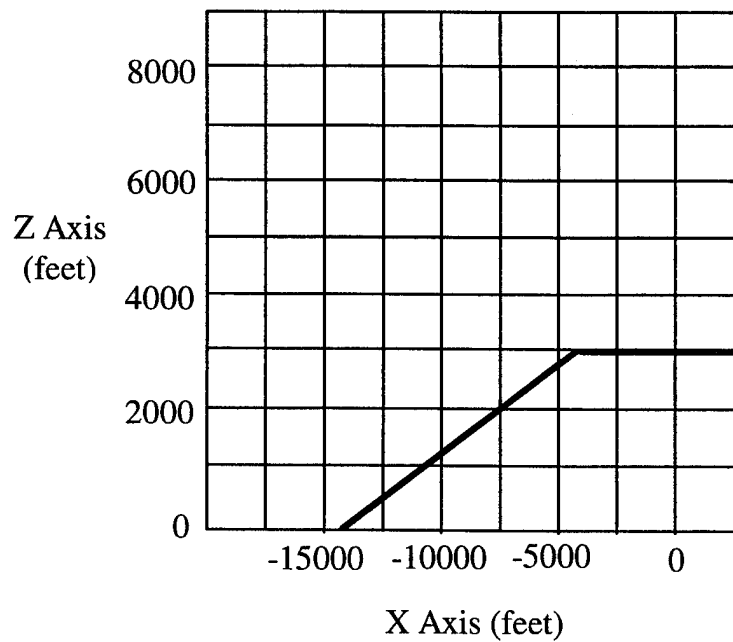


Figure 6.6. Simulation Display of Missile Altitude

routine, *sim_results.m*, will produce a plot similar to that shown in Figure 6.7. Originally, this plot was designed to be displayed and produced in real-time during the simulation. However, the increased processing demand slowed the entire evolution down dramatically. A user would not benefit from observing the simulation in three dimensions if it takes hours to complete.

In the current configuration, run times require approximately 20 minutes, depending on the input setup geometry, TEL speed, and time-to-detection of the TEL.

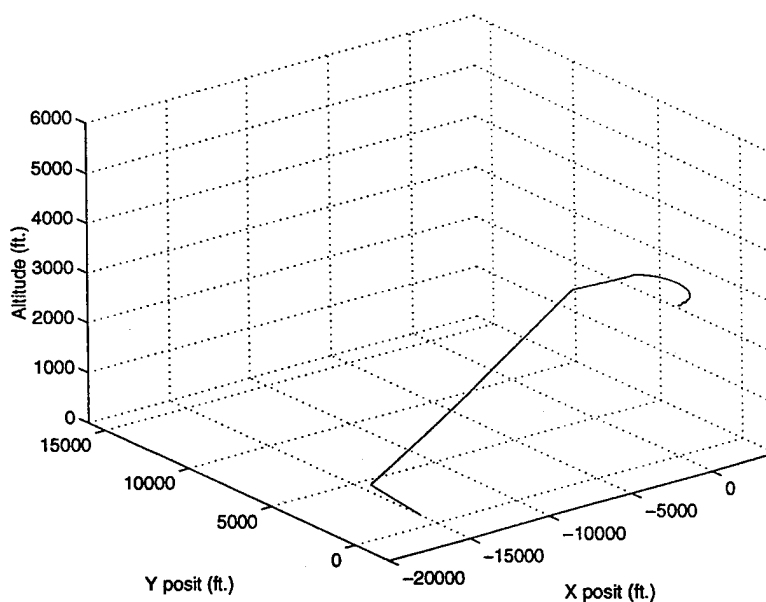


Figure 6.7. 3-D Plot of an Entire Simulation

VII. RECOMMENDATIONS AND CONCLUSIONS

The small number of simulations performed to date have been primarily run to test the proper operation of the model. It has been noticed that missile miss distances consistently range from 20 to 60 feet, depending on the setup. Miss distances are expected because the missile guidance system, as described in Chapter IV, is subject to the normal selective availability and receiver clock errors associated with GPS. Although the distances experienced are acceptable for air navigation, they would likely be too large for a small missile to effectively destroy a TEL.

The missile model greatly resembles the lethal UAV model in regards to trajectory track control. As a method of guidance, this is well suited to the initial and midcourse phases. However, a more precise method of target homing is required for the terminal phase of the missile's flight. This is especially the case for a small missile with a necessarily small warhead. In keeping with the objective of presenting a low cost TBM counterforce alternative, the lethal UAV must be small and simplistic. Therefore, the size of the missile and warhead will also be limited. Future work on this project should include researching a viable, compact, and effective terminal homing option for the missile component; and, implementing a SIMULINK/MATLAB model for it.

Future efforts should also include striving for consistent zero miss distances by the elimination of all sources of error and noise. In doing so, the model may serve as a

“truth model,” representing a perfect and ideal situation. Such a tool would provide much use in Monte Carlo and error analysis.

It has been stated repeatedly that the conservation of processing demand was a limiting consideration throughout the design of the counterforce model. AUTO CODE software has been requested for use in the Department of Aeronautics and Astronautics, which will greatly accelerate the processing speed of SIMULINK/MATLAB applications on the *Silicon Graphics* workstations. Follow-on work should include pursuing the AUTO CODE acquisition, installation, and implementation into the counterforce model. This will afford the designer much more latitude in producing accurate, complex model components, and provide for a less generalized TBM counterforce scenario.

Another recommendation for continued work on this project is incorporating into the UAV model an modern sensor suite capable of detecting, identifying and tracking a TEL. This modification would greatly alter the presented counterforce concept by eliminating the use of a UGS system. However, it would also represent a more costly plan. The modified scenario could be used to compare and affirm the performance of the lower cost counterforce plan represented by the current model.

The lethal UAV in the new plan would be much more complex than a simple missile platform. Modern reconnaissance and targeting sensors have been designed in a small and lightweight manner explicitly for use aboard UAVs. These include electro-optic (EO) sensors, infrared (IR) sensors, and synthetic aperture radar (SAR) [Ref. 8].

With the current attention on UAV technology, the potential capabilities are immense.

However, the price of these capabilities can also be significant.

Another significant result of replacing the ground sensors with airborne sensors would be the lack of true autonomous operation. The use of airborne sensors requires a human-in-the-loop. The technology has not yet been realized to automatically identify an object based on its reproduced image obtained from any of an infinite number of aspects. Whereas second generation UGS systems have the advantage of sensing seismic, acoustic and magnetic signatures which can be learned and stored in libraries within the sensor, airborne sensors must rely on near-real time data link of imagery to a ground control station for target verification by a human.

Finally, effort should be made to improve the presentation of the simulation runs. Specifically, output from the model could be used by animation software such as *Designer's workbench*. This program has recently been incorporated into other UAV projects in the Department of Aeronautics and Astronautics at the Naval Postgraduate School.

The goal of this thesis, as established in Chapter I, was to develop and introduce a computer model for simulating an autonomous TBM counterforce plan involving a lethal UAV. The model demonstrated the feasibility of a low cost, yet effective, concept, and is a useful and flexible tool for a variety of analyses in the future. Having realized this goal; the next steps are to make use of and tailor the model, and demonstrate the possibilities for lethal UAV applications.

APPENDIX . SUPPORTING MATLAB CODE

The following is a list of MATLAB routines written in support of the SIMULINK portion of the model.

Bluedat.m

% This is the dynamic model data file for Bluebird. The data for any aircraft may be
% substituted.

function [u0, w0, rho, s, b, c, m, To, Ib, Cfx, Cfo, Cfi, Cfxdot] = bluedat

u0 = 73.3;

w0 = 0;

rho = .0023769;

s = 22.38;

b = 12.42;

c = 1.802;

m = 1.7095;

To = [15 0 0 0 0 0]';

Ib = [10 0 0; 0 16.12 0; 0 0 7.97];

Cfx = [0 0 .188 0 0 0;

0 -.31 0 0 0 .0973;

0 0 4.22 0 3.94 0;

0 -.0597 0 -.363 0 .1;

0 0 -1.163 0 -11.77 0;

0 .0487 0 -.0481 0 -.0452];

Cfo = [.03 0 .385 0 0 0]';

```

Cfi = [.065 0 0;
0 .0697 0;
.472 .0147 0;
0 .0028 .265;
-1.41 0 0;
0 -.0329 -.0347];
Cfxdot = [0 0 0 0 0 0;
0 0 0 0 0 0;
0 0 1.32 0 0 0;
0 0 0 0 0 0;
0 0 -4.7 0 0 0;
0 0 0 0 0 0];

```

count_cycles.m

```

function cycles = count_cycles(a)

% This function inputs a 4x1 matrix consisting of the amplitude
% of the acoustic signature of the received signal, the previous amplitude
% of the signal, the cycle count, and the system time.
% The function then counts the number of cycles by determining sign
% changes from positive to negative.
% The output is the amplitude of the signal, the running count of cycles,
% and the system time.

% input data

amplitude = a(1,1);
prev_amp = a(2,1);
cycles = a(3,1);
time = a(4,1);

% Test if current amplitude is positive or negative

if amplitude > 0
    t = 1;
else
    t = 2;
end

% Test if previous amplitude is positive or negative

```

```

if prev_amp > 0
    s = 1;
else
    s = 2;
end

% Calculate a test variable for logic step

r = t + s;

% If the two amplitudes are different signs, a zero crossing occurred and a
% half a cycle has occurred. The one is added to account for the simulation
% start point being other than zero.

if r == 3
    cycles = cycles + 1;
end

% output data

cycles = [amplitude;cycles;time];

```

dstate2.m

```

function statedot = dstate2(x)

% This function computes udot, vdot, wdot, pdot, qdot, rdot, phidot, thetadot,
% and psidot of an aircraft which are then fed through an integrator to obtain
% u, v, w, p, q, r, phi, theta, and psi. The aircraft accelerations in the {U}
% frame are also computed which are then fed to the accelerometer model.

% Get constants

[u0, w0, rho, s, b, c, m, To, Ib, Cfx, Cfo, Cfi, Cfxdot] = bluedat;

% Assign incoming vector, x

u = x(1);
v = x(2);

```

```

w = x(3);
p = x(4); q = x(5); r = x(6);
phi = x(7);
theta = x(8);
psi = x(9);
da = x(10); de = x(11); dr = x(12); dthr = x(13);
states = [u v w p q r]';
states1 = [u-u0, v, w-w0, p, q, r]';
lamda = [phi theta psi];
inputs = [de dr da]';

Vt = (u^2 + v^2 + w^2)^.5;

qbar = .5*rho*(Vt^2);

M1 = diag([1/Vt, 1/Vt, 1/Vt, (.5*b)/Vt, (.5*c)/Vt, (.5*b)/Vt]);

M2 = diag([0, (.5*b)/(Vt^2), (.5*c)/(Vt^2), 0, 0, 0]);

Sbar = diag([-1, 1, -1, b, c, b]*s);

M = inv([eye(3)*m zeros(3); zeros(3) Ib]);

% Calculate Tw2b

alfa = w/Vt;
beta = v/Vt;

T1 = [cos(alfa)*cos(beta) -cos(alfa)*sin(beta) -sin(alfa);
      sin(beta) cos(beta) 0;
      sin(alfa)*cos(beta) -sin(alfa)*sin(beta) cos(alfa)];

Tw2b = [T1, zeros(3); zeros(3), T1];

% Calculate chi

chi = eye(6) - M*Tw2b*qbar*Sbar*Cfxdot*M2;

% Gravity force rotated to {B} frame

g = 32.174;

uFg = m*g;

```

```

Ru2b = [-sin(theta);
        sin(phi)*cos(theta);
        cos(theta)*cos(phi);
        0;0;0];

bFg = uFg*Ru2b;

% Calculate inputs
thrust = To*dthr;

ctrl = qbar*(Tw2b*(Sbar*(Cfo + (Cfi*inputs))));

xdoti = (M*(ctrl+thrust+bFg));

% Calculate kinematic contributions

omegax = [0 -r q;
          r 0 -p;
          -q p 0];

wxIb = (-inv(Ib))*(omegax*Ib);

xdot = [-omegax zeros(3); zeros(3) wxIb]*states;

% Calculate aerodynamic forces

xdotcfx = qbar*(M*(Tw2b*(Sbar*(Cfx*(M1*states1)))));

% Combine all factors

kinedot = inv(chi)*(xdot + xdotcfx + xdoti);

lamdot = [1 sin(phi)*tan(theta) cos(phi)*tan(theta);
          0 cos(phi) -sin(phi);
          0 sin(phi)*sec(theta) cos(phi)*sec(theta)]*[p q r]';

% Calculate accelerations in the {U} frame

wxv = omegax*[u v w]';

Rot = [1 0 -sin(theta);
       0 cos(phi) cos(theta)*sin(phi);

```



```

0 -sin(phi) cos(theta)*cos(phi)];

uA = Rot'*(kinedot(1:3) + wxv - bFg(1:3));

statedot = [kinedot; lamdot; uA/m];

```

freq.m

```

function f = freq(b)

% This file inputs a 4x1 matrix. This matrix consists of the cycle count,
% the system time, the previous cycle count, and the previously calculated
% frequency. The function then computes the frequency of the signal. The
% output is a 2x1 matrix that is the cycle count and the current frequency.

% input data

count = b(1,1);
time = b(2,1);
pcnt = b(3,1);
phz = b(4,1);

% If the current cycle count is greater than the previous count
% a variable is set accordingly for the next logic step.

if count > pcnt
    z = 1;
else
    z = 2;
end

% With a new cycle present, indicated by z = 1, the frequency is calculated
% otherwise the old frequency is passed. The frequency is determined by
% dividing the number of cycles by the system time. By doing this as frequency
% would change the file would be able to always determine the new frequency.

if z == 1
    hz = ((count - 1)/2)/time;
else
    hz = phz;
end

```

% output data

f = [count;hz];

gpsrange.m

function [r]=gpsrange(z)

% This function calculates the range from each of four satellites to the vehicle.
% The input is a 15x1 vector. The first three rows are the exact position
% of the vehicle in ECEF coordinates. The next twelve rows are the
% positions of the 4 satellites. The out put is a 4x1 vector of the
% calculated ranges from each satellite to the vehicle.

vx = z(3);
vy = z(2);
vz = z(1);
s1x = z(4);
s1y = z(5);
s1z = z(6);
s2x = z(7);
s2y = z(8);
s2z = z(9);
s3x = z(10);
s3y = z(11);
s3z = z(12);
s4x = z(13);
s4y = z(14);
s4z = z(15);

% Compute Ranges

r(1) = sqrt((s1x-vx)^2 + (s1y-vy)^2 + (s1z-vz)^2);
r(2) = sqrt((s2x-vx)^2 + (s2y-vy)^2 + (s2z-vz)^2);
r(3) = sqrt((s3x-vx)^2 + (s3y-vy)^2 + (s3z-vz)^2);
r(4) = sqrt((s4x-vx)^2 + (s4y-vy)^2 + (s4z-vz)^2);

gravity.m

```
function g = gravity(r)

% This function computes the gravity vector in the tangent plane.
% 'r' is the vector of the vehicle's position in tangent plane.

R0 = 2.093e7;
mu = 32.174*R0^2;
R = (r(1)^2 + r(2)^2 + (r(3) + R0)^2)^.5;
gx = -(mu/R^3)*r(1);
gy = -(mu/R^3)*r(2);
gz = -(r(3) + R0)*mu/R^3;
g = [gx; gy; gz];
```

launchswitch.m

```
function launch = launchswitch(x)

% This function produces the launch signal which is sent to the missile
% model. Launch is triggered with the following conditions met: 1) Detection
% of the target, 2) UAV on heading towards tgt, and 3) slant range to
% target < 12000 feet.

slanrange = x(1);
onhdg = x(2);
detect = x(3);
if slanrange < 12000 & onhdg == 1 & detect == 1;

    launch = 1;

else

    launch = 0;

end;
```

Lethal_uav.m

```
% This file is run by the user to begin a simulation of the lethal UAV TBM counterforce  
% scenario. The user is simply required to be in the TBM directory in the  
% MATLAB command window, and type "lethal_uav."
```

```
type welcome;          % Welcome.m contains introductory text that appears to the user
```

```
load lethal;
```

```
lethal_alt = input('Input altitude for the lethal uav in feet (ex: 3000)')  
trkspmph = input('Input the speed of the TEL in mph (ex: 30)')  
trkspeed = (trkspmph)*(5280)*(1/3600);  
trkhdg = input('Input the heading of the TEL in degrees (ex: 080)')  
trkxpos = input('Input X position of TEL in feet (ex: -15000)')  
trkypos = input('Input Y position of TEL in feet (ex: 1000)')  
time2det = input('Input the time in seconds until the UGS detects the TEL (ex: 20)')
```

```
senspos;               % Senspos.m computes the position of the UGS for simulation  
                      % continuity.
```

```
RK45('uavs'4000);    % This command starts the simulation.
```

lineofsight.m

```
function angles = lineofsight(x);
```

```
% This function computes the missile Line-of-Sight (LOS) angles, alfa and  
% beta, to the target.
```

```
acx = x(1);  
acy = x(2);  
acz = x(3);  
tgtx = x(4);  
tgty = x(5);  
tgtz = x(6);
```

```
dx = abs(acx - tgtx);  
dy = abs(acy - tgty);  
dz = abs(acz - tgtz);
```

```
slanrange = (dx^2 + dy^2 + dz^2)^.5;  
range = (dx^2 + dy^2)^.5;
```

```
alfa = asin(dz/slanrange);
```

```
beta1 = acos(dx/range);
```

```
% Place beta in the correct quadrant
```

```
if tgtx > acx & tgty > acy;
```

```
    beta = beta1;
```

```
elseif tgtx < acx & tgty > acy;
```

```
    beta = pi - beta1;
```

```
elseif tgtx > acx & tgty < acy;
```

```
    beta = 2*pi - beta1;
```

```
else
```

```
    beta = pi + beta1;
```

```
end;
```

```
angles = [alfa; beta];
```

missdist.m

```
function CEP = missdist(x)
```

```
% This function stops the simulation when the missile reaches a point 50 feet  
% above the ground and then calculates the miss distance of the missile using  
% the x and y coordinates of the missile and the target.
```

```
mslx = x(1);
```

```
msly = x(2);
```

```
mslz = x(3);
```

```
tgtx = x(4);
```

```

tgty = x(5);
tgtz = x(6);

if mslz >= tgtz-50; %% This is to eliminate the miss associated with high msl
    %% dive angle. Change the "50" if other than .05 step size.
    halt = 1;
else
    halt = 0;
end;

if halt == 1

    CEP = ((tgtx-mslx)^2 + (tgty-msly)^2)^.5

else

    CEP = 0;

end;

CEP = [halt; CEP];

```

missileom.m

```

function vdot = missileom(x);

% This function calculates udot, vdot, and wdot which are then fed through an
% integrator to obtain u, v, and w. Thrust is a 10 second impulse of a 2500 lb
% force which is controlled by the variable "timer". Gravity and control
% inputs are not allowed to act on the missile until the launch command is
% received. Drag and control inputs are proportional to velocity.

timer = x(1);

theta = x(2);

shoot = x(3);

yi = x(4);

zi = x(5);

```

```

u = x(6);
v = x(7);
w = x(8);

Vt = (u^2 + v^2 + w^2)^.5;

% Missile mass

m = 1.554;

% Calculate gravity force in {B} frame

Ru2b = [-sin(theta); 0; cos(theta)];

% Conditions for thrust, gravity, controls, and drag based on timer and launch

if shoot == 1;

    timer = timer + 1;
end;

if shoot == 1 & timer <= 25; % Change this "25" when step size other than .05 sec

    thrust = 2500;

else;

    thrust = 0;

end;

if shoot == 1

    g = 32.176;
    inputs = Vt*[0; yi; zi];
    drag = Vt* [.05; 0; 0];
else

    g = 0;
    inputs = [0; 0; 0];
    drag = [0; 0; 0];
end;

```

```
v_dot = (m*g*Ru2b + [thrust; 0; 0] + drag + inputs)/m;
```

```
vdot = [timer; v_dot];
```

onhdg.m

```
function swout = onhdg(x)
```

```
% This function produces the "On heading" signal which causes the aircraft to  
% stop its max rate turn and fly towards the target. The signal occurs when  
% the aircraft heading is equal to the target bearing (within a small  
% tolerance). The counter is used to ensure that the signal is not changed  
% (which would result in the aircraft returning to a max rate turn) if the  
% tolerance is exceeded at some later time.
```

```
%  
hdg = x(1);
```

```
detect = x(2);
```

```
brng = x(3);
```

```
cntr = x(4);
```

```
tol = .02;
```

```
% If the aircraft has completed a full orbit, the heading value must be  
% returned to a value,  $0 < \text{hdg} < 2\pi$ .
```

```
if hdg  $\geq 2\pi$ ;
```

```
    for i = 1:20
```

```
        if hdg  $\geq 2\pi$ ;
```

```
            hdg = hdg - ( $2\pi$ );
```

```
        end;
```

```
    end;
```

```
end;
```

```
% The aircraft will fly towards the target only if detection has occurred
```

```
if detect == 1;
```



```

        if hdg + tol >= brng & hdg - tol <= brng;

            cntr = cntr + 1;
            onhdg = 1;

        elseif cntr >= 1;

            onhdg = 1;
        else;
            onhdg = 0;
        end;
    else;
        onhdg = 0;
    end;

    swout = [onhdg; cntr];

```

plotter.m

% This function produces a 3D summary plot of the simulation run.

```

function plotter(msl,trk)

load msl
load trk

plot3(msl(2,:),msl(3,:),msl(4,:))
hold
plot3(trk(2,:),trk(3,:),trk(4,:),'c')
axis([-20000 4000 -2000 16000 0 6000])
grid
title('3D summary Plot of Simulation')
xlabel('X posit (ft.)')
ylabel('Y posit (ft.)')
zlabel('Altitude (ft.)')

```

range.m

```
sfunction [y] = range(z)

% This routine represents the truth model for
% the GPS ranges for Kalman filter estimates.
% The input is the vehicle's position in the tangent plane, and the
% output is the vector of exact ranges y = [y1;y2;y3;y4].
% Note here the origin of the tangent plane is chosen to be at
% the equator, on the Greenwich meridian:

s1 = z(1:3); s2 = z(4:6); s3 = z(7:9); s4 = z(10:12);

% position of body origin in tangent plane
Pbou = z(13:15);

% radius of earth

R0 = 6.38e6;

% position of tangent plane origin
Puoe = [R0;0;0];

% transformation matrix from ecef to tangent plane

Rue = [0 0 1;0 1 0;1 0 0];

% position in tangent plane

Pe = Rue*Pbou + Puoe;

% output data

y = [norm(Pe - s1); norm(Pe - s2); norm(Pe - s3); norm(Pe - s4)];
```

results.m

% This routine is called by sim_results.m, and the words are presented to the user.

The missile-to-TEL miss distance (in feet) is

sim_results.m

% This file produces a 3-D summary plot of the simulation,
% and displays the missile/TEL miss distance.

```
type results
cep;
posit=size(cep);
cepft=cep(posit(1,1),1);
cepft
plotter
```

tgtbearing.m

function brng = tgtbearing(x)

% This function computes the bearing of the target from the aircraft (in the
% {U} frame). It also computes the slant range to the target for use in
% determining missile launch time.

```
acx = x(1);
acy = x(2);
acz = x(3);
```

```
tgtx = x(4);
tgty = x(5);
tgtz = x(6);
```

```
dx = acx-tgtx;
dy = acy-tgty;
dz = acz-tgtz;
```

```
range = (dx^2 + dy^2)^.5;
```

```
slanrange = (dx^2 + dy^2 + dz^2)^.5;
```

```
theta = -atan(abs(dz)/range);
```

```

brng1 = acos(abs(acx-tgtx)/range);

% Target bearing is placed in the correct quadrant
if tgtx > acx & tgty > acy;

    brng = brng1;

elseif tgtx < acx & tgty > acy;

    brng = pi - brng1;

elseif tgtx > acx & tgty < acy;

    brng = 2*pi - brng1;
else
    brng = pi + brng1;
end;

brng = [brng; slantrange; theta];

```

theta_psi.m

```

function angles = theta_psi(x);

% This function computes the flight path angles, theta and psi, of the missile
% based on its x, y, and z axis velocities

mslv_x = x(1);
mslv_y = x(2);
mslv_z = x(3);

Vt = (mslv_x^2 + mslv_y^2 + mslv_z^2)^.5;

theta_vel = -asin(mslv_z/Vt);

psi_vel = asin(mslv_y/Vt);

% Psi is placed in the correct quadrant

if mslv_x < 0 & mslv_y <= 0

```

```

        psi = pi - psi_vel;

elseif mslv_x < 0 & mslv_y > 0

        psi = pi - psi_vel;

elseif mslv_x > 0 & mslv_y < 0

        psi = 2*pi + psi_vel;

else
        psi = psi_vel;

end;

angles = [theta_vel; psi];

```

trigger.m

```

function trgr = trigger(m)

% This file calculates the data necessary to trigger detection. A 5x1 matrix
% is the input. This matrix is made up of the frequency from UGS 2, the
% amplitude from UGS 2, the range to UGS 1, the frequency from UGS 1
% and the previous amplitude from UGS 2. The file uses this data to
% determine detection of the target by mark on top of sensor 2, and the
% velocity of the target. It also calculates the time to sensor 1. The
% output of this file is the matrix consisting of the current amplitude from
% from sensor 2, a detection flag, the time to sensor 1, and the targets
% velocity.

% input data

freq2 = m(1,1);
amp2 = m(2,1);
range1 = m(3,1);
freq1 = m(4,1);
prvamp2 = m(5,1);

% Stored data. Freqref is the known constant frequency. 1026.7 is the speed

```

```
% of sound in air in feet per second (700 mph). Corrn is added to account for
% too great a step size causing the sine wave generated by the truck to not
% be very smooth. This can be adjusted as necessary to obtain the truth model.
% 0.4 is for a simulation step size of 0.05. It can be set to zero and errors
% in actual and calculated velocity will be very small. Test is used to
% determine mark on top of sensor 2. This will occur when the amplitude
% peaks.
```

```
freqref = 1;
lambda = 1026.7/freqref;
corn = 0.4;
test = amp2 - prvamp2;
```

```
% Test if amplitude peaked. If previous amplitude is greater the signal
% peaked and is decreasing. If mark on top occurred send a detection flag
% to the uav. Time to sensor 1 is calculated for the scenario when destruction
% of the target will occur at sensor 1. Frequencies from sensors 1 and 2
% are present to be able to switch from 2 to 1 at a certain range when the
% signal becomes too small in amplitude. This may be necessary when noise is
% added to the system. It may also be necessary to input the amplitude from
% sensor 1. For the flight time of the uav now the range is fine. If flight
% time is increased it may be necessary to make the above changes. Velocity
% is passed so that it can be integrated to determine target position.
```

```
if test >= 0
    detect = 0;
    time2s1 = 0;
    velocity = 0;
else
    detect = 1;
    velocity = ((lambda*(freq1 - freqref))/2) + corn;
    time2s1 = range1/velocity;
end
```

```
trgr = [amp2;detect;time2s1;velocity];
```

truck_data1.m

```
function data = truck_data1(c)
```

```
% This file computes range and signal characteristics of the truck's acoustic
% signature from UGS1 located at a position in the tangent plane. This
% UGS has to be the sensor furthest from the target, i.e. the target would
% arrive at this sensor second. The file inputs a 5x1 matrix of the x and y
% position of the truck in the tangent plane, the system time, and the x and y
% velocity of the truck. It uses this data to generate the acoustic signature
% of the truck in amplitude and frequency form sensor 1. The range to sensor 1
% is also output.
```

```
% input data
```

```
x_posit = c(1,1);
y_posit = c(2,1);
time = c(3,1);
x_vel = c(4,1);
y_vel = c(5,1);
```

```
% sensor 1 location in tangent plane
```

```
sensor_x1 = -20000;
sensor_y1 = 20000;
```

```
% reference data for doppler equation. freqref is the known constant frequency
% source frequency. 1026.7 is the speed of sound in air in fps. (700 mph).
```

```
freqref = 1;
lambda = 1026.7/freqref;
```

```
% calculate data
```

```
range = (((x_posit - sensor_x1)^2) + ((y_posit - sensor_y1)^2))^(1/2);
```

```
velocity = ((x_vel)^2 + (y_vel)^2)^(1/2);
```

```
amp = 10000/(4*pi*(range^2));
```

```
% doppler equation
```

```
freq = freqref + ((2*velocity)/(lambda));
```

```
% acoustic signature
```

```
signal = amp*(sin(2*pi*freq*time));
```

```
% Output data
```

```
data = [range;signal;amp];
```

truck_data2.m

```
function data = truck_data2(c)
```

```
% This file computes range and acoustic signature signal characteristics  
% of the truck from sensor 2 located in the tangent plane. Sensor 2 must  
% be before sensor 1 in range from the truck. i.e. sensor 2 is closest to  
% the truck. The input is a 5x1 matrix of the x and y position and velocity  
% of the truck and the system time. This data is used to develop the  
% acoustic signature of the truck at sensor 2. The output is the range  
% to sensor 2, the acoustic signature signal, and the amplitude.
```

```
% Input data
```

```
x_vel = c(1,1);  
y_vel = c(2,1);  
time = c(3,1);  
x_posit = c(4,1);  
y_posit = c(5,1);
```

```
% Sensor location in tangent plane. Must be closest to truck.
```

```
sensor_x1 = -21500;  
sensor_y1 = 21500;
```

```
% Reference data. Freqref is the known constant frequency source frequency.  
% 1026.7 is the speed of sound in air in fps. (700 mph).
```

```
freqref = 1;  
lambda = 1026.7/freqref;
```

```
range = (((x_posit - sensor_x1)^2) + ((y_posit - sensor_y1)^2))^(1/2);
```



```
velocity = ((x_vel)^2 + (y_vel)^2)^(1/2);
```

```
amp = 10000/(4*pi*(range^2));
```

```
% Doppler equation
```

```
freq = freqref + ((2*velocity)/(lambda));
```

```
% Acoustic signature signal
```

```
signal = amp*(sin(2*pi*freq*time));
```

```
% Output data
```

```
data = [range;signal;amp];
```

velocity2d.m

```
function v2d = velocity2d(k)
```

```
% This file inputs the speed of the target and calculates the speed in the  
% x and y directions. This can be done since we know the heading of the road  
% and the truck is constrained to it. The speeds are calculated using the  
% Pythagorean theorem on a triangle whose angles are 45,45,90. There  
% the two sides are of equal length, hence one side = hypotenuse/sqrt(2).
```

```
% INPUT DATA
```

```
v1d = k(1,1);
```

```
% CALCULATE VX VY VZ
```

```
vx = (v1d)/((2)^(1/2));
```

```
vy = -vx;
```

```
vz = 0;
```

```
% OUTPUT DATA
```

```
v2d = [vx;vy;vz];
```

welcome.m

% This file is called by *lethal_uav.m*; and the text appears on the screen for the user.

This dynamic model serves as a Master's Thesis project by LT Dick Kammann,
with invaluable contributions by:

LT John Green	LT Mike Knowles
LT Steve Hope	LT Scott Winfrey
LT Aaron Rondeau	

of the Naval Postgraduate School Aerospace Engineering Department. Questions
and maintenance calls should be directed to Professor Isaac Kaminer @ 656-2491.

This simulation runs the SIMULINK system, *lethal_uav.m*. The system consists
of a moving transporter-erector-launcher (TEL) vehicle which is detected,
identified and tracked by an unmanned ground sensor (UGS), which directs an
overhead missile-carrying UAV to a firing solution. The missile is fired and tracks
the TEL position until impact.

Upon completion of this simulation the M-file, "sim_results," can be run to
produce the missile miss distance and a 3-D plot of the simulation.

For a complete discussion and description of this model, please read the
September, 1995 thesis (of the title shown above) by LT Richard W. Kammann.

LIST OF REFERENCES

1. Conner, G. W., Ehlers, M. A., Marshall, K. T., *Countering Short Range Ballistic Missiles*, Naval Postgraduate School, Monterey, CA, 1993.
2. Berhow, T., Buckingham, B., Day, J., Davis, S., Hill, J., Nessler, S., Wolfe, J., *A Theatre Ballistic Missile (TBM) Counterforce Concept*, Naval Postgraduate School, Monterey, CA, 1993.
3. Marshall, K. T., *The Roles of Counterforce and Active Defense in Countering Theater Ballistic Missiles*, Naval Postgraduate School, Monterey, CA, 1994.
4. Siuru, B., *Planes Without Pilots - Advances in Unmanned Flight*, TAB AERO, 1991.
5. Wagner, W. and Sloan W. P., *Fireflies and Other UAVs (Unmanned Aerial Vehicles)*, Aerofax, Inc., 1992.
6. Hallberg, E. N., *Design of a GPS Aided Guidance , Navigation, and Control System for Trajectory Control of an Air Vehicle*, Master's Thesis, Naval Postgraduate School, Monterey, CA, 1994.
7. Herrington, J. B., *Uniform Framework for GPS/IMU Integration Using Kalman Filtering*, Master's Thesis, Naval Postgraduate School, Monterey, CA, June, 1995.
8. Fulghum, D. A., Morrocco, J. D., "U.S. Military to Boost Tactical Recon in '95," *Aviation Week & Space Technology*, pp.22-23, January 9, 1995.
9. Soutter, P. A., *On the Use of Unmanned Aerial Vehicles to Search for Tactical Ballistic Missile Transporter-Erector-Launchers*, Master's Thesis, Naval Postgraduate School, Monterey, CA, 1994.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
Cameron Station
Alexandria, Virginia 22304-6145
2. Library, Code 52 2
Naval Postgraduate School
Monterey, California 93943-5101
3. Chairman 2
Department of Aeronautics and Astronautics
Naval Postgraduate School
Monterey, California 93943-5002
4. Professor Isaac I. Kaminer, Code AA/Ka 2
Department of Aeronautics and Astronautics
Naval Postgraduate School
Monterey, California 93943-5002
5. CAPT George W. Conner, USN (Ret.), Code OR/Co 2
Department of Operations Research
Naval Postgraduate School
Monterey, California 93943-5002

6. Professor Kneale T. Marshall, Code OR/Mt2
Department of Operations Research
Naval Postgraduate School
Monterey, California 93943-5002
7. U.S. Naval Test Pilot School 4
ATTN: LT Scott Winfrey, USN
BLDG 2168, Mail Stop 10
22783 Cedar Point Road
Patuxent River, Maryland 20670-5304
8. LT Andrew L. Caldera, USN 1
6206 Littlethorpe Lane
Alexandria, Virginia 22310
9. LT Richard W. Kammann, Jr., USN2
101 Santomera Lane
Wilmington, Delaware 19807
10. Mr. Michael Harper 1
Teledyne Ryan Aeronautical
2111 Wilson Boulevard, Suite 1100
Arlington, Virginia 22201-3058