NAVAL POSTGRADUATE SCHOOL
Monterey , California

THESIS

SECOND AND THIRD ORDER MINIMUM TIME CONTROLLERS AND MISSILE ADJOINTS

by

Colin Roy Cooper

June 1991

Thesis Advisor: Hal A. Titus

Jnclassified

Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified		1b Restrictive Markings			
2a Security Classification Authority		3 Distribution Availability of Report Approved for public release; distribution is unlimited.			
2b Declassification/Downgrading Schedule					
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)			
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(If Applicable)* EC	7a Name of Monitoring Organization Naval Postgraduate School			
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000			
8a Name of Funding/Sponsoring Organization	8b Office Symbol *(If Applicable)*	9 Procurement Instrument Identification Number			
8c Address *(city, state, and ZIP code)*		10 Source of Funding Numbers			
		Program Element Number	Project No	Task No	Work Unit Accession No

11 Title *(Include Security Classification)* SECOND AND THIRD ORDER MINIMUM TIME CONTROLLERS AND MISSILE ADJOINTS

12 Personal Author(s) Colin R. Cooper

13a Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report *(year, month,day)* June 1991	15 Page Count 95

16 Supplementary Notation 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.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	

19 Abstract *(continue on reverse if necessary and identify by block number*

The optimal minimum time controller (i.e. bang-bang controller) is applied to the fast reaction missile defense problem. From Pontryagin, the optimal control was determined to be a function of the adjoint in the minimization of the Hamiltonian. The control may also be posed either as a function of time or as a function of the states. The state space can be partitioned into regions, surfaces and curves where the optimal control action is either its maximum plus or minus N.

In missile simulation problems, the method of adjoints is often used in parametric studies of errors and miss distance. This technique is developed both graphically and mathematically, and is used here to help one visualize the solution trajectory and families of optimal trajectories for all possible initial conditions.

20 Distribution/Availability of Abstract [✓ unclassified/unlimited] same as report DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Harold A. Titus	22b Telephone *(Include Area code)* (408) 646-2560	22c Office Symbol EC/Ts

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted — All other editions are obsolete

security classification of this page
Unclassified

Second and Third Order Minimum Time Controllers and Missile Adjoints

by

Colin Roy Cooper
B.S., University of California, San Diego, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1991

Michael M. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

The optimal minimum time control (i.e. bang-bang controller) is applied to the fast reaction missile defense problem. From Pontryagin, the optimal control was determined to be a function of the adjoint in the minimization of the Hamiltonian. The control may also be posed either as a function of time or as a function of the states. The state space can be partitioned into regions, surfaces and curves where the optimal control action is either its maximum plus or minus N.

In missile simulation problems, the method of adjoints is often used in parametric studies of errors and miss distance. This technique is developed both graphically and mathematically, and is used here to help one visualize the solution trajectory and families of optimal trajectories for all possible initial conditions.

TKO515
C7524
C.1

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

In appreciation for their time, effort, and patience, many thanks go to my instructors, advisors, and the staff and faculty of the Electrical & Computer Engineering Dept., NPS. Also, to my fellow students with whom I spent many long days, and longer nights, learning more than just what was taught in classes.

A special note of thanks goes to my friend and advisor Dr. Hal Titus; for his guidance, encouragement, and his special way of making the world a better place to live.

Most of all I want to thank my wife, whose unfaltering love and confidence were a constant source of strength and motivation. Thank you Cassandra.

I. INTRODUCTION

In this era of technological advancement, more and more effort is being devoted to automation to increase speed and efficiency. Everything is becoming faster, smaller, more efficient and more effective, from fuzzy logic controlled appliances to smart weapon systems. As our engineering and design of physical systems improves we are able to generate faster and more accurate mechanical devices, and the control systems for such devices must improve as well. In pushing to develop the fastest, most efficient controller for whatever our application, we introduce the bang-bang controller.

Many of our current defensive missile systems were designed to be used against threats that are no longer of greatest importance. A point defense missile may now be required to bring down a target that has a significant speed advantage, and be able to do it in such a way as to control the fallout after the impact.

Even if we redesign our systems, current economic conditions make it unlikely that we could build up an arsenal of new weapon systems. However, we could upgrade our currently existing arsenal by replacing the original control logic with a newer, more capable controller.

In this report we will develop second and third order minimum time optimal controllers and apply them to a fast reaction missile defense problem. While we are using the missile defense problem as our example, it should be noted that the controllers are not limited to this example and may be applied as generic controllers for a wide variety of systems.

II. SECOND ORDER CONTROLLER

A. MINIMUM TIME OPTIMAL CONTROL

Designing a control system is frequently a hit-and-miss process using a variety of design techniques to iteratively create a system that meets specific criteria. The performance of the system may be defined in the time and frequency domain in terms of overshoot, settling time, etc., or it may be defined by some externally measured criteria. For example, a satellite control system may be designed to minimize fuel consumption.

"The objective of optimal control theory is to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criterion." [1]

1. Problem Definition

The system and optimization problem is defined as

$$\dot{x} = Ax + Bu \tag{2.1}$$

with $x(0) = c$, and $x(t_f) = 0$, while minimizing

$$J = \int_0^{t_f} dt. \tag{2.2}$$

Let us consider the simple example with

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u. \tag{2.3}$$

From Pontryagin [1], we find we can minimize J by minimizing the Hamiltonian

$$H = 1 + p_1 x_2 + p_2 u. \tag{2.4}$$

This is minimum when u is operating at its maximum possible value and with opposite the sign of the adjoint p_2. Thus we have

$$u = -N \cdot \mathrm{sign}(p_2) \tag{2.5}$$

where

$$\dot{p} = -\frac{\partial H}{\partial x} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix} p. \tag{2.6}$$

This has a solution

$$p_1 = c_1 \tag{2.7}$$

$$p_2 = -c_1 t + c_2. \tag{2.8}$$

A typical solution would appear as seen in Fig. 2.1.

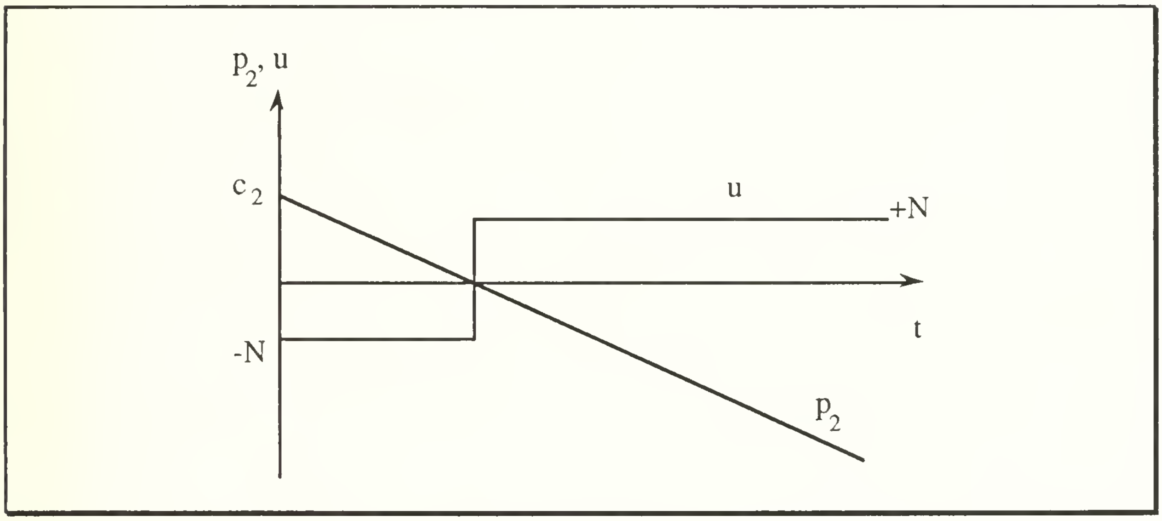

Figure 2.1 Solution to Minimized Hamiltonian

Note that from the adjoint solution, the control may change sign only once. We would like to solve this problem for all possible initial conditions and only one terminal condition ($x(t_f) = 0$). Hence it makes sense to look at this problem in negative time (adjoint), starting from the end point of motion. From the uniqueness theorem in ordinary differential equations, only two trajectories can emanate from the origin; one for u = -N and one for u = +N, as shown in Figure 2.2. In this second order example, our solution is constrained to the x_1, x_2 plane. These negative time trajectories divide the state space (here a plane) into two parts. Solution trajectories emanating from these curves constitute all possible solutions for all possible initial conditions.

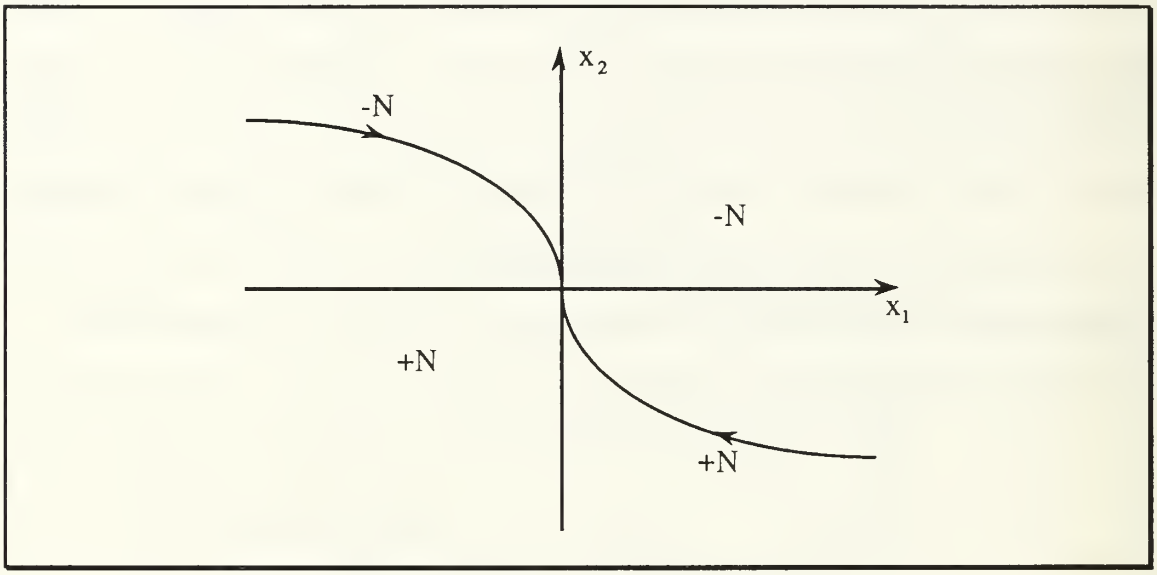

Figure 2.2 Zero Trajectory Curves

2. Minimum Time Trajectories, Second Order

Consider the system

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u . \tag{2.9}$$

This can be represented in flow diagram form as

$$\begin{tikzcd} u \arrow[r, "\frac{1}{s}"] & x_2 \arrow[r, "\frac{1}{s}"] & x_1 \end{tikzcd}$$

The control, u, may be defined for a bang-bang control system as ±N, where N is some constant. Given any starting values for the states, there are only two possible paths for the states to take; one corresponding to u = +N, and one corresponding to u = -N. If we desire to drive all states to zero, and we know that we can only apply u = ±N, Figure 2.3 shows the paths followed by the states given various starting values.

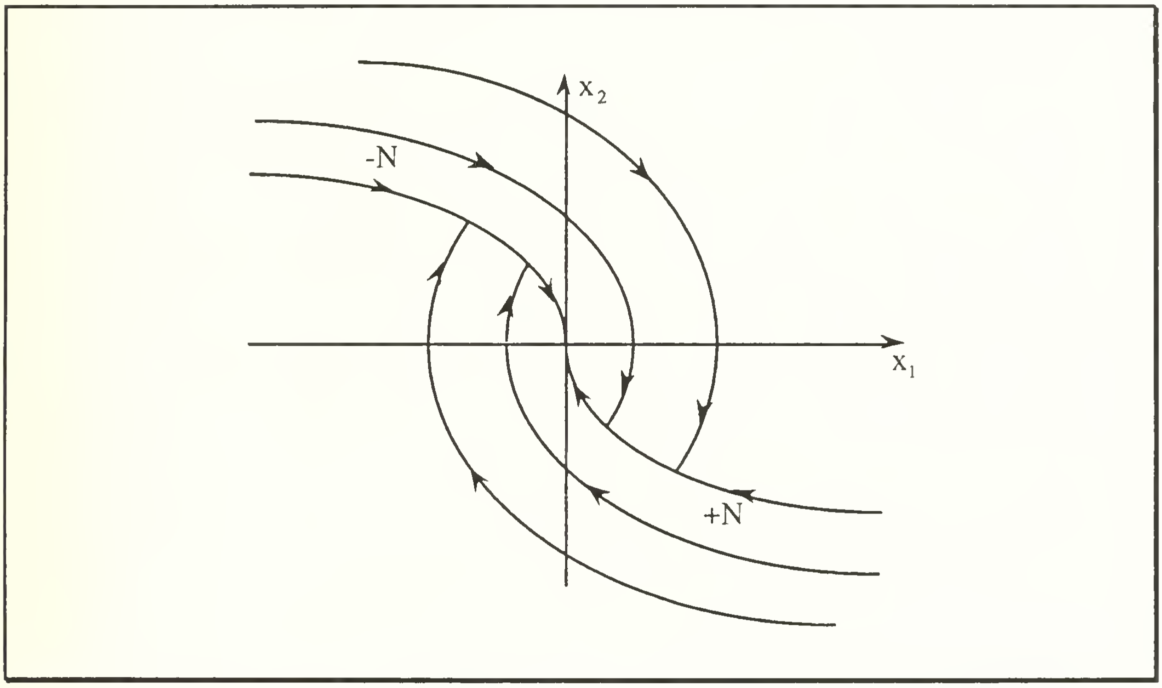

Figure 2.3 Minimum Time Trajectory Solution Curves

3. Second Order Solution Trajectories

This system has a solution

$$x(t) = \phi(t,0)x(0) + \Delta(t,0)u(0) \tag{2.10}$$

where

$$\phi = e^{At}, \tag{2.11}$$

and

$$\Delta = \int_0^t e^{At} B\,dt. \tag{2.12}$$

Expanding e^{At}, we get

$$e^{At} = \left[I + At + \frac{1}{2!} A^2 t^2 + \ldots \right]$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} t + \frac{1}{2!} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} t^2 + \ldots$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & t \\ 0 & 0 \end{bmatrix} + \frac{1}{2!} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} t^2 + \ldots = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}. \tag{2.13}$$

Since A^2 is the zero matrix, all higher orders of A will also be the zero matrix and we may drop them. Evaluating Δ (for u fixed) we find

$$\Delta = \int_0^t e^{A(t-\tau)} B\,d\tau = \int_0^t \begin{bmatrix} 1 & t-\tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} d\tau$$

$$= \int_0^t \begin{bmatrix} t-\tau \\ 1 \end{bmatrix} d\tau = \begin{bmatrix} t\tau - \frac{1}{2}\tau^2 \\ \tau \end{bmatrix}_0^t = \begin{bmatrix} \frac{1}{2}t^2 \\ t \end{bmatrix}. \tag{2.14}$$

The solution for this system is

$$x(t) = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} x(0) + \begin{bmatrix} \frac{1}{2}t^2 \\ t \end{bmatrix} u(0), \tag{2.15}$$

or in scalar equations

$$x_1(t) = x_1(0) + x_2(0)t + \tfrac{1}{2}u(0)t^2 \tag{2.16}$$

$$x_2(t) = x_2(0) + u(0)t. \tag{2.17}$$

These equations describe the states as a function of time given any initial conditions and the fixed control effort, u.

B. ANALYTIC SOLUTION FOR SECOND ORDER SWITCHING TIME

We may now treat this system as a boundary value problem and analytically solve for switching times of the control effort.

1. Solving Boundary Value Problems

Since the control is piecewise constant, ($\pm N$), we can separate the problem into two pieces and match boundary values at the point where u changes sign. In other words, if the system moves from point $x(t_0)$ through point $x(t_s)$ to point $x(t_f)$, we can solve the problem in two parts. Each of our boundary value problems can be stated in such a way as to supply simplifying boundary conditions, i.e., setting initial or final values to zero. Optimal control theory tells us that for a second order system there will be at most one switching time, the change from $u = +N$ to $u = -N$, or visa versa. Let us consider first the solution for our system with some initial condition $[x_1(0), x_2(0)]$, as shown in Figure 2.4.

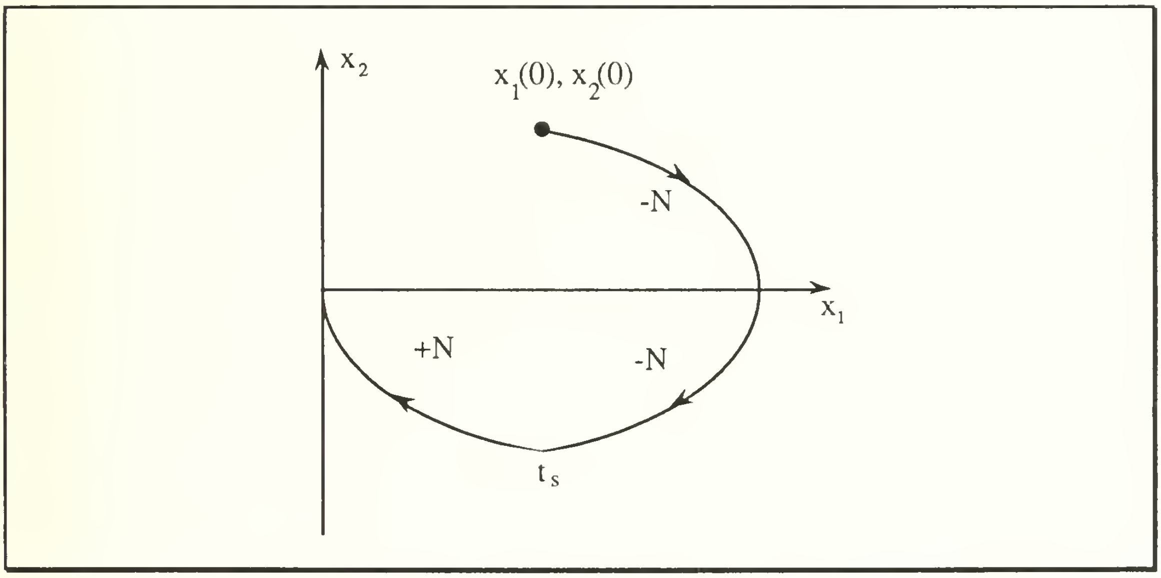

Figure 2.4 Minimum Time Trajectory From a Fixed Initial Condition

The time from t=0 to t_s is the length of time before the control effort changes sign. We cannot solve directly for t_s because we do not have enough information on the boundary values. Since we are solving this problem for arbitrary initial conditions, $x_1(0)$ and $x_2(0)$, $x_1(t_s)$ and $x_2(t_s)$ are unknown. We do know, however, that the final states, $x_1(t_f)$ and $x_2(t_f)$, will be zero, and from this we can determine the zero-trajectory curve in negative time from the origin. This curve, shown in Figure 2.5., is the only curve that will go through the origin with $u = +N$. Therefore the switching time will occur when the path of a $u = -N$ curve intersects this curve. There are an infinite number of $u = -N$ curves intersecting this curve, however, only one curve will go through any particular set of initial conditions.

To simplify the problem, we will start with initial conditions on the x_1 axis. Define $x_1(0) = c_1$, $x_2(0) = 0$, $x_1(t_f) = x_2(t_f) = 0$, and $u = -N$. This situation is described in Figure 2.6 where t_1 may be defined as the time from t_0 to t_s. Equation (2.16) may now be written

$$x_1(t) \quad = x_1(0) + x_2(0)t + \tfrac{1}{2}ut^2 = x_1(0) - \tfrac{1}{2}Nt^2 \tag{2.18}$$

Since the curves for $u = +N$ and $u = -N$ are symmetric, it can be noted that

$$x_1(t_1) = \tfrac{1}{2}x_1(0) \tag{2.19}$$

and therefore

$$\tfrac{1}{2}x_1(0) = x_1(0) - \tfrac{1}{2}Nt_1^{\,2} \tag{2.20}$$

and finally

$$t_1 = \sqrt{\frac{x_1(0)}{N}}\,. \tag{2.21}$$

This is valid for any initial conditions such that $x_1(0) > 0$, $x_2(0) = 0$ and $u = -N$.

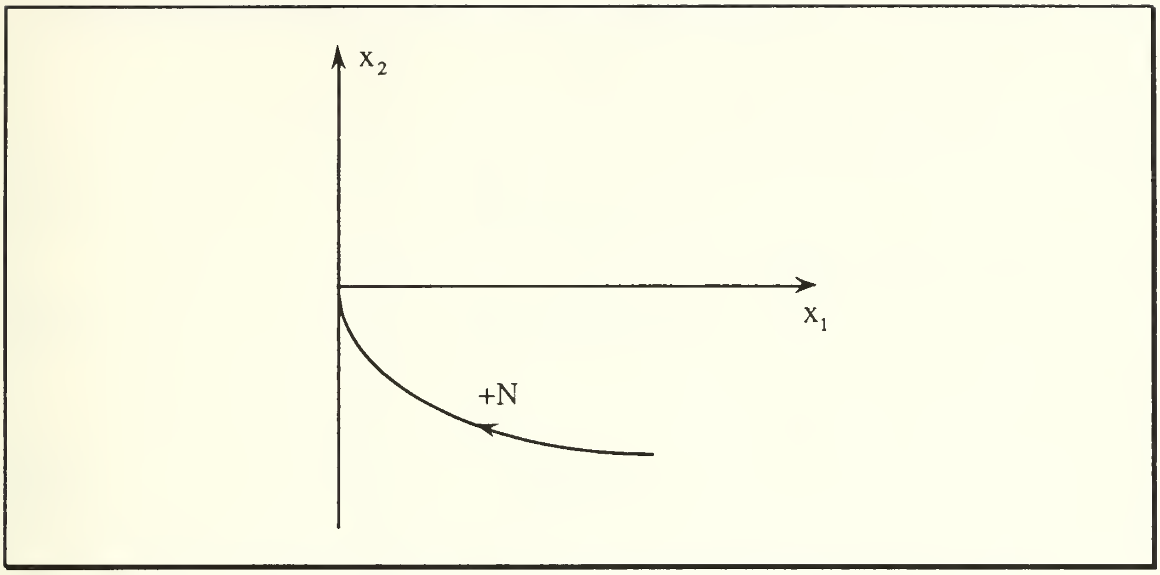

Figure 2.5 Simplified Boundary Value Problem #1

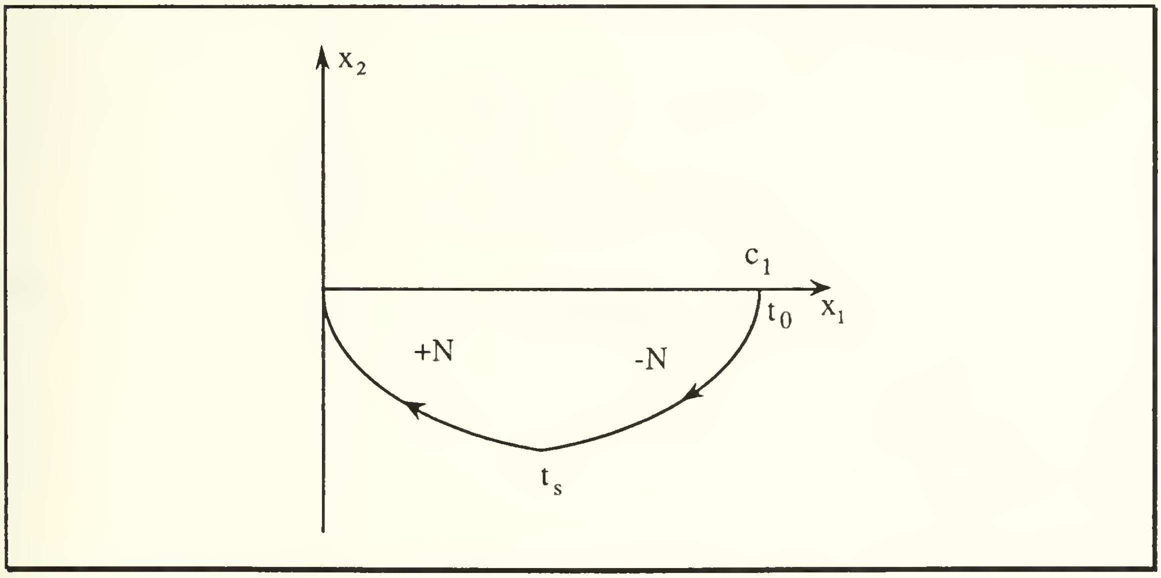

Figure 2.6 Simplified Boundary Value Problem #2

The next simplification involves getting from the positive x_2 axis down to the positive x_1 axis as shown in Figure 2.7. Given $x_1(0) = 0$, $x_2(0)$ = positive real, $x_1(t_f)$ = positive real, $x_2(t_f) = 0$, and $u = -N$.

Equation (2.17) may be written

$$x_2(t) = x_2(0) - Nt. \tag{2.22}$$

At $t = t_2$, $x_2(t_2) = 0$ giving

$$0 = x_2(0) - Nt_2 \tag{2.23}$$

and

$$t_2 = \frac{x_2(0)}{N}. \tag{2.24}$$

The final stage is to translate the initial condition off of the x_2 axis to some unknown initial condition. The time required for the states to get from some initial condition $x_2(0)$ to $x_2(t)$ is not based on the x_1 state, and is therefore always equal to t_2, as seen in Figure 2.8.

We solve (2.16) for $x_1(t_2)$

$$\begin{aligned} x_1(t_2) &= x_1(0) + x_2(0)t_2 - \tfrac{1}{2}Nt_2^2 \\ &= x_1(0) + x_2(0)\left(\frac{x_2(0)}{N}\right) - \tfrac{1}{2}N\left(\frac{x_2(0)}{N}\right)^2 \\ &= x_1(0) + \frac{1}{2N}x_2^2(0). \end{aligned} \tag{2.25}$$

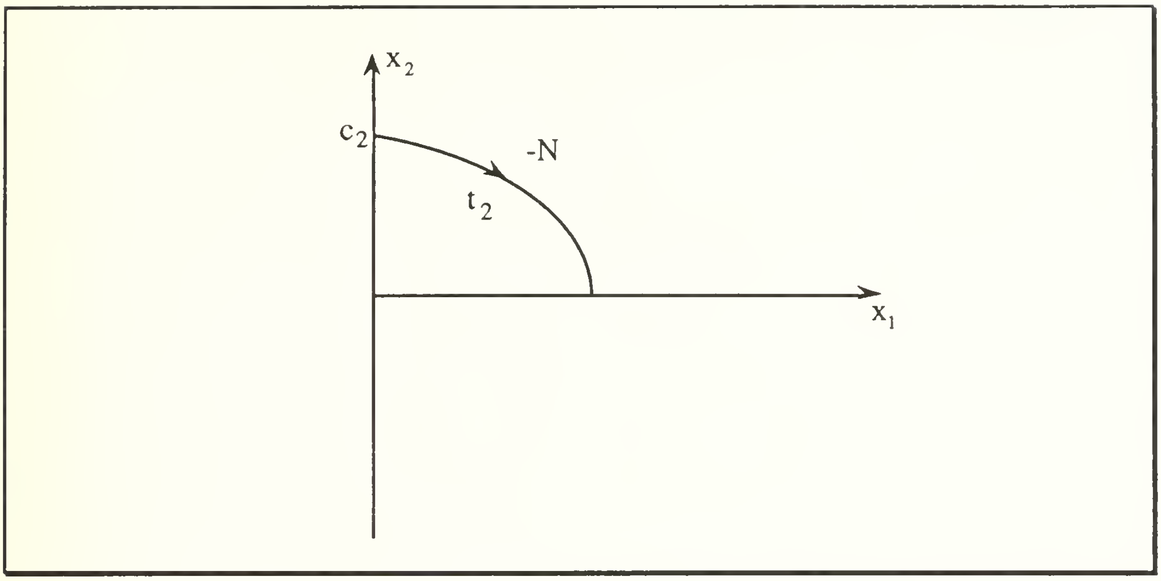

Figure 2.7 Simplified Boundary Value Problem #3

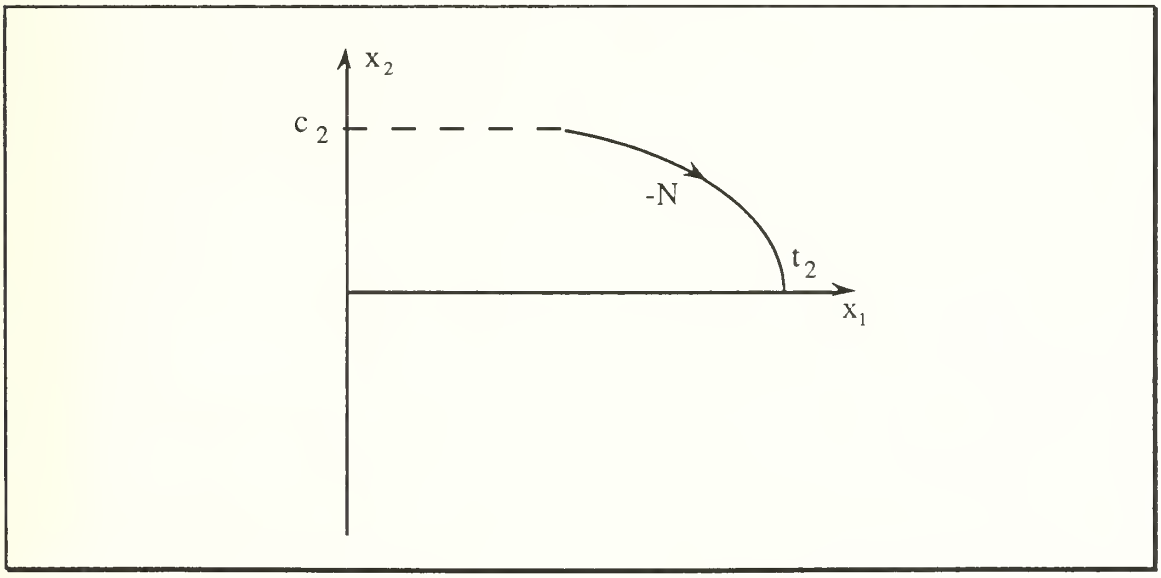

Figure 2.8 Simplified Boundary Value Problem #4

Finally we combine these solutions to determine the switching time, t_s, for any initial conditions such that $x_1(0) > 0$ and $x_2(0) > 0$ with $u = -N$. From Figure 2.3 we see that $x_1(0)$ for the t_1 solution is the same point as $x_1(t_2)$ from the t_2 solution so that

$$t_1 = \sqrt{\frac{x_1(0)}{N}} = \sqrt{\frac{x_1(t_2)}{N}}$$

$$= \sqrt{\frac{x_1(0) + \frac{1}{2N} x_2^2(0)}{N}}$$

$$= \sqrt{\frac{x_1(0)}{N} + \frac{1}{2}\left(\frac{x_2(0)}{N}\right)^2} \tag{2.26}$$

and

$$t_s = t_1 + t_2 = \sqrt{\frac{x_1(0)}{N} + \frac{1}{2}\left(\frac{x_2(0)}{N}\right)^2} + \frac{x_2(0)}{N}. \tag{2.27}$$

These equations were derived for specific initial conditions and are only valid where the initial conditions lie above the zero trajectory curves of Figure 2.2. In order to expand the capabilities of the control system for all initial conditions we re-derive the switching time equations for initial conditions $x_1(0) < 0$, $x_2(0) < 0$, and $u(0) = +N$ for the solution

$$t_s = \sqrt{\frac{-x_1(0)}{N} + \frac{1}{2}\left(\frac{x_2(0)}{N}\right)^2} - \frac{x_2(0)}{N}. \tag{2.28}$$

This equation is valid only when the initial values are below the zero trajectory curves of Figure 2.2. For this discussion we will limit ourselves to initial conditions above the zero trajectory curves and use the solution (2.27).

2. Simulation of Analytic Switching Time Controller

Using a computer simulation, we test the accuracy of the solution by choosing initial conditions and observing the response to our control effort. We simulated the example system of (2.9) with $x_1(0) > 0$, $x_2(0) > 0$, and $N = -1$. The control effort, u, changes sign to -N at the calculated switching time, t_s. The results show that the states pass through the origin of the state space and the error at the origin is associated with the discretization of the simulation, as shown in Figures 2.9 and 2.10. The switching time shown in Figure 2.10 is the calculated value. The control effort actually switches at the first sampling time following the caluculated switching time, t_s. If we increase the sample rate for the simulation we reduce the terminal miss error. Upon reaching the origin some additional control logic must be devised to maintain position.

While the switching time solution is a minimum time solution for driving the states to the origin, or any desired values, it must be shut off at t_f. The switching time solution cannot adapt to a time varying situation as the switching time is defined solely on the initial conditions.

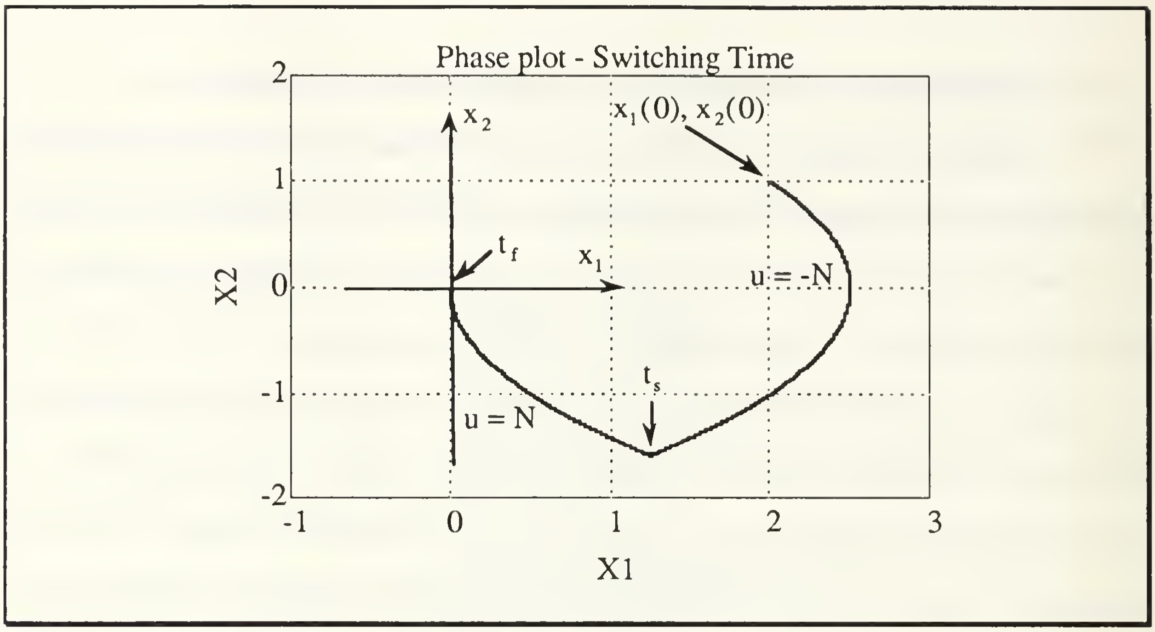

Figure 2.9 Simulation of Analytic Switching Time Solution

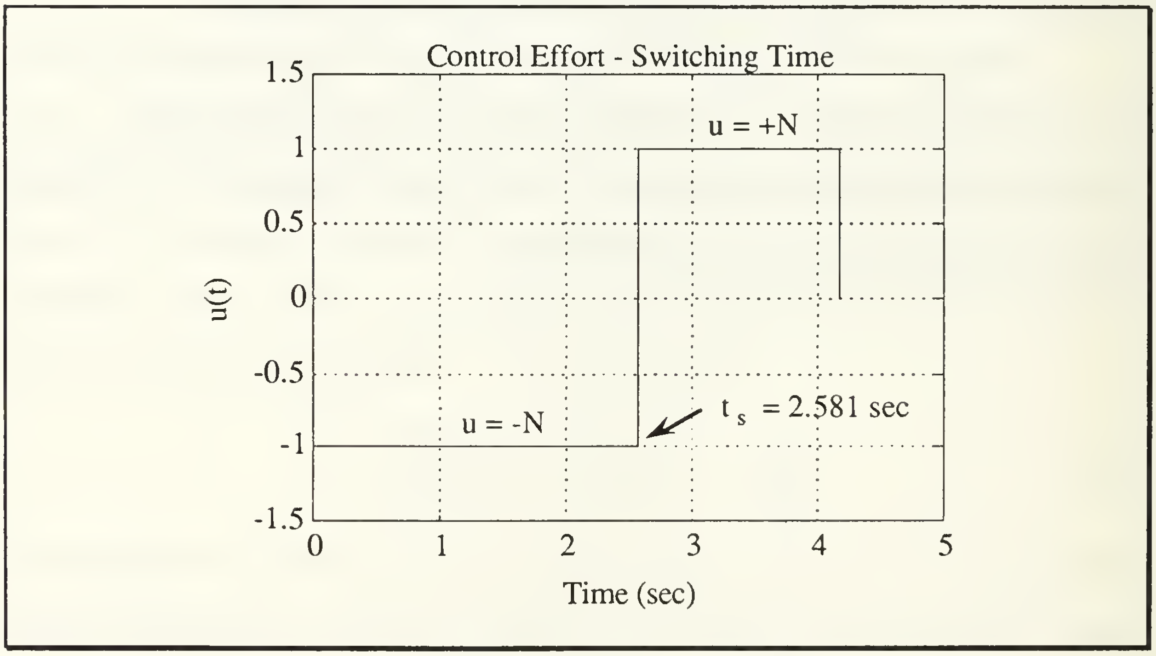

Figure 2.10 Control Effort for Analytic Switching Time Simulation

C. SOLUTION TO SECOND ORDER SWITCHING CURVES

An option for improving the capabilities of the second order controller is to remove the time dependancy of the control effort. The control problem may be posed as a function of the states instead of a function of time for some simple systems. By defining the control effort as only a function of the states the control effort is updated continuously and may change immediately in response to a change in the system parameters.

There are only two possible control efforts, -N and +N. Therefore, any position in space will have either +N or -N control effort to drive it along its unique path to the origin, and we can solve this system for the second order switching curves that divide the state space into two parts, see Figure 2.2.

1. Second Order Switching Laws

Given the system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{2.29}$$

we can integrate the state equations. Setting $u = \pm N$

$$x_2(t) = \int \dot{x}_2(t)dt = \int udt = \pm Nt + c_1 \tag{2.30}$$

$$x_1(t) = \int \dot{x}_1(t)dt = \int x_2 dt = \int \pm Nt + c_1 dt = \pm \tfrac{1}{2} Nt^2 + c_1 t + c_2 . \tag{2.31}$$

Choosing the boundary values so that $x_1(t_f) = x_2(t_f) = 0$ we may rewrite the equations as

$$x_1(t_f) = \pm \tfrac{1}{2} Nt_f^2 + x_2(0)t_f + x_1(0) \tag{2.32}$$

$$x_2(t_f) = \pm Nt_f + x_2(0) \tag{2.33}$$

or

$$0 = \pm \tfrac{1}{2} Nt_f^2 + x_2(0)t_f + x_1(0) \tag{2.34}$$

$$0 = \pm N t_f + x_2(0). \tag{2.35}$$

Solving for t_f

$$t_f = \mp \frac{x_2(0)}{N}. \tag{2.36}$$

Then for any value of t

$$t = \mp \frac{x_2(t)}{N}. \tag{2.37}$$

Substituting this into equation (2.34)

$$0 = x_1(t) + x_2(t)\left(\mp \frac{x_2(t)}{N}\right) \pm \tfrac{1}{2} N \left(\mp \frac{x_2(t)}{N}\right)^2 \tag{2.38}$$

$$0 = x_1(t) \mp \frac{x_2^2(t)}{N} \pm \tfrac{1}{2} \frac{x_2^2(t)}{N} \tag{2.39}$$

$$0 = x_1(t) \mp \tfrac{1}{2} \frac{x_2^2(t)}{N}. \tag{2.40}$$

We make the substitution

$$\mp x_2^2(t) = -x_2(t)\left|x_2(t)\right| \tag{2.41}$$

so that

$$0 = x_1(t) - \tfrac{1}{2N} x_2(t)\left|x_2(t)\right|. \tag{2.42}$$

Equation (2.42) describes the two parabolic trajectories referred to as the zero trajectory curves because any states on these curves will, with the appropriately signed control effort, be driven to the origin. These zero trajectory curves divide the state space into two regions of opposite control effort. If the states are above the curves then u = -N, but if they are below the curves then u = +N. Therefore our control effort, u, may be defined

$$u = -N \cdot \text{sign}\left(x_1(t) - \tfrac{1}{2N} x_2(t)\left|x_2(t)\right|\right). \tag{2.43}$$

With this switching law we drive the states from any initial conditions to the origin with at most one change of the control effort.

For example, let us define the initial conditions of the states to be above the zero trajectory curves so that $x_1(0) < 0$ and $x_2(0) > 0$, as shown in Figure 2.11. From (2.43) the control effort is $u = -N$, which drives the states along the parabolic trajectory shown in Figure 2.11.

When the states intersect the zero trajectory curve, the control effort changes, according to (2.43) to $u = +N$, and follows the zero trajectory curve into the origin.

2. Limit Cycles

The control effort, $u = +N$ will not only drive the states to the origin, it will, in fact, become confused there. It is a point of indecision and chatter or limit cycle motion ensues as in Figure 2.12. The magnitude of the limit cycle depends on the sampling rate or time delay for the control effort. If the sample rate is low, the states will penetrate far into the opposite control region before the control effort can change, and the limit cycle will swing widely around the origin. If the sample rate is high then the states will not travel far away from the origin before the control effort corrects the direction of travel.

Since a heavy chatter mode is not desirable for many real systems, control logic for reducing or removing the chatter mode may be developed, however, it will not be included in this paper.

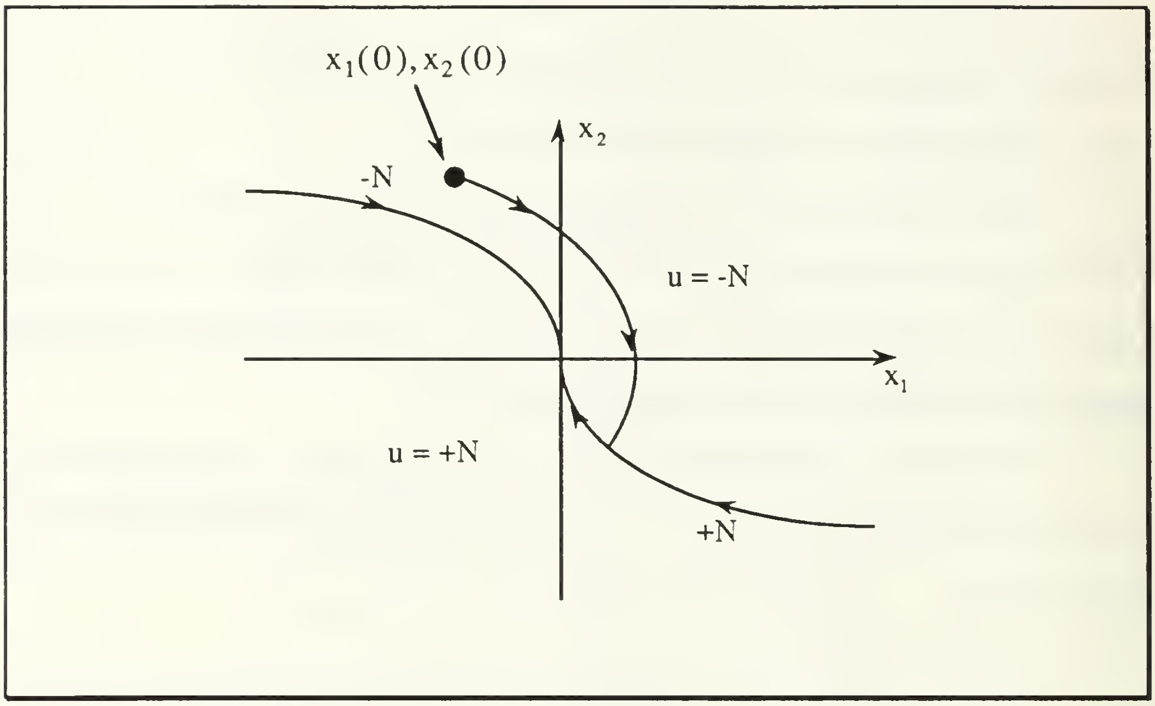

Figure 2.11 Solution Trajectory From Fixed Initial Condition

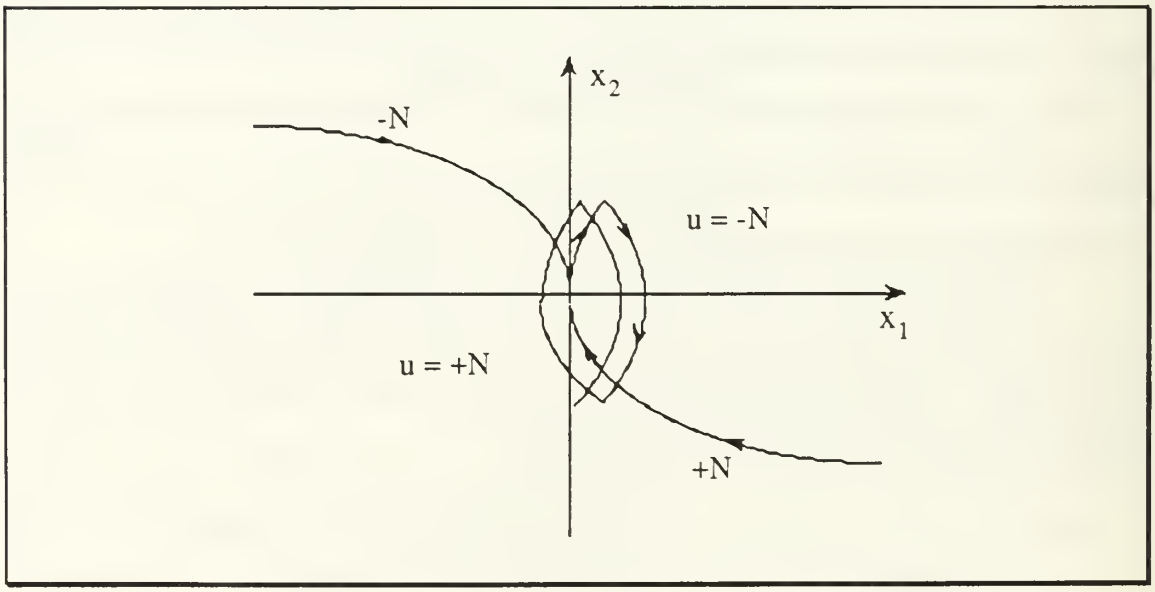

Figure 2.12 Limit Cycle on Second Order Solution Trajectory

3. Simulation of the Switching Law Controller

To test the switching law (2.43), we simulate the system of (2.29) using a maximum control effort of N = 1 and initial conditions $x_1(0) > 0$, $x_2(0) > 0$. The output of the simulation, shown in Figure 2.13, demonstrates the control effort driving the states first to the zero trajectory curve, then along the zero trajectory curve to the origin. Once at the origin, the control effort goes into limit cycle as shown in Figure 2.14.

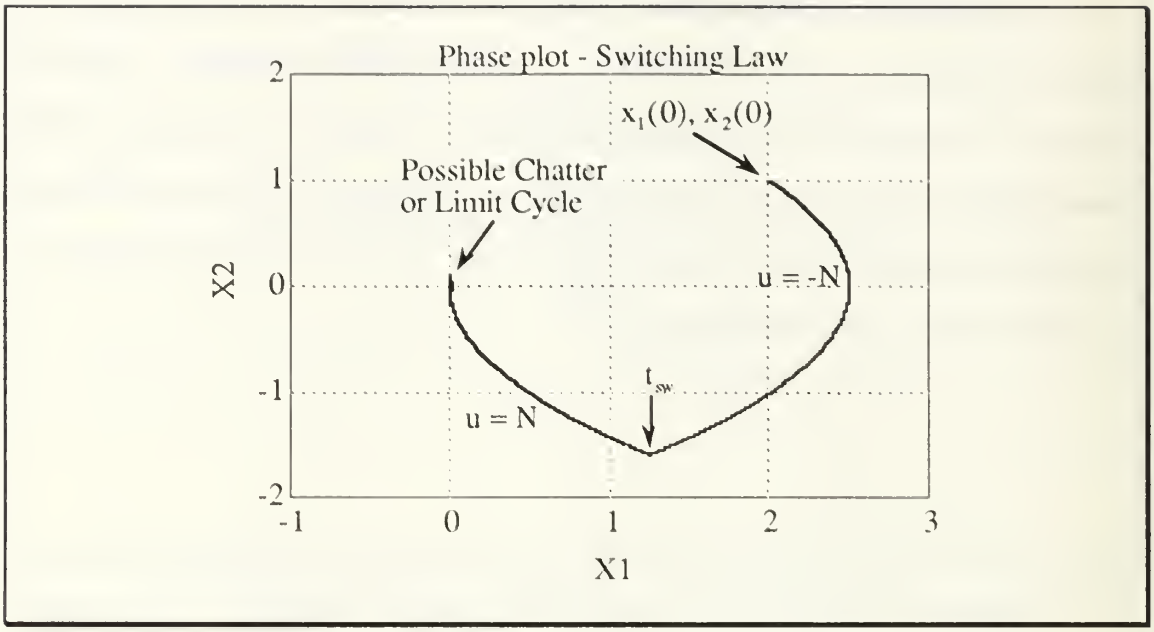

Figure 2.13 Simulation Using Second Order Switching Law

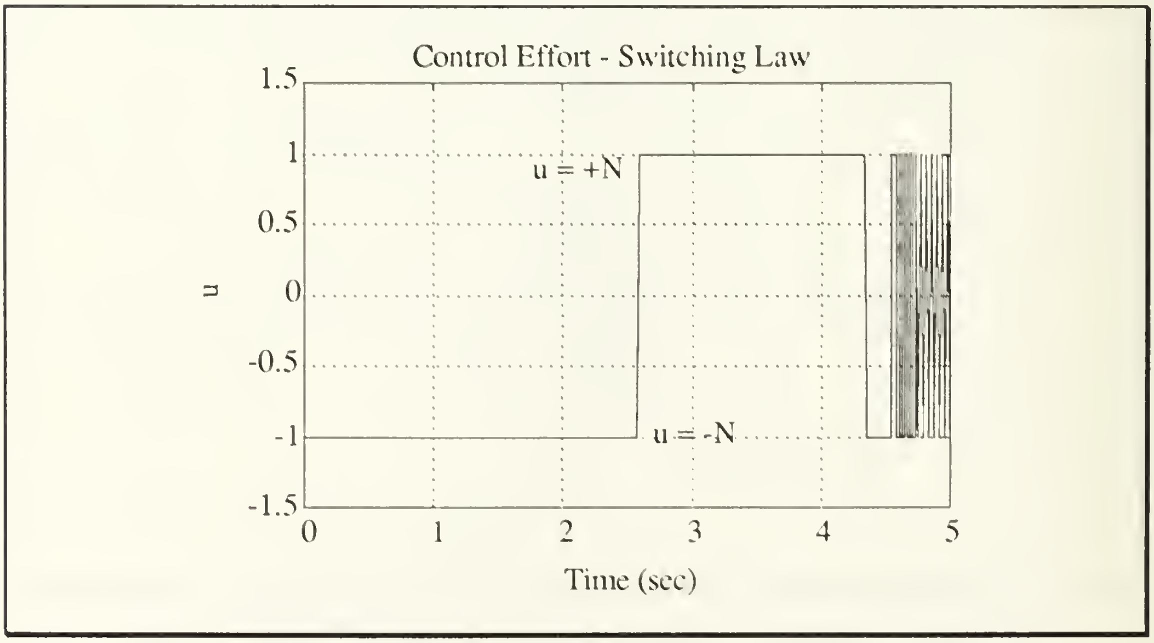

Figure 2.14 Control Effort for Second Order Switching Law Solution

III. APPLICATION OF A SECOND ORDER CONTROLLER

A. MISSILE/TARGET INTERCEPT MODEL

An application for Minimum Time Optimal Control is an anti-missile defense system, where the incoming target has a speed advantage over the defensive missile. In this case the missile control system must operate in saturation mode. The bang-bang controller, where control effort is either maximum-positive or maximum-negative, has a faster response capability than the standard Proportional Navigation guidance system. A simple model of a missile to target engagement can be described by Figure 3.1 where σ is the line of sight (LOS) angle from missile to target, γ_m is the angle of the missile velocity, γ_t is the angle of the target velocity, and all angles are relative to an inertial reference.

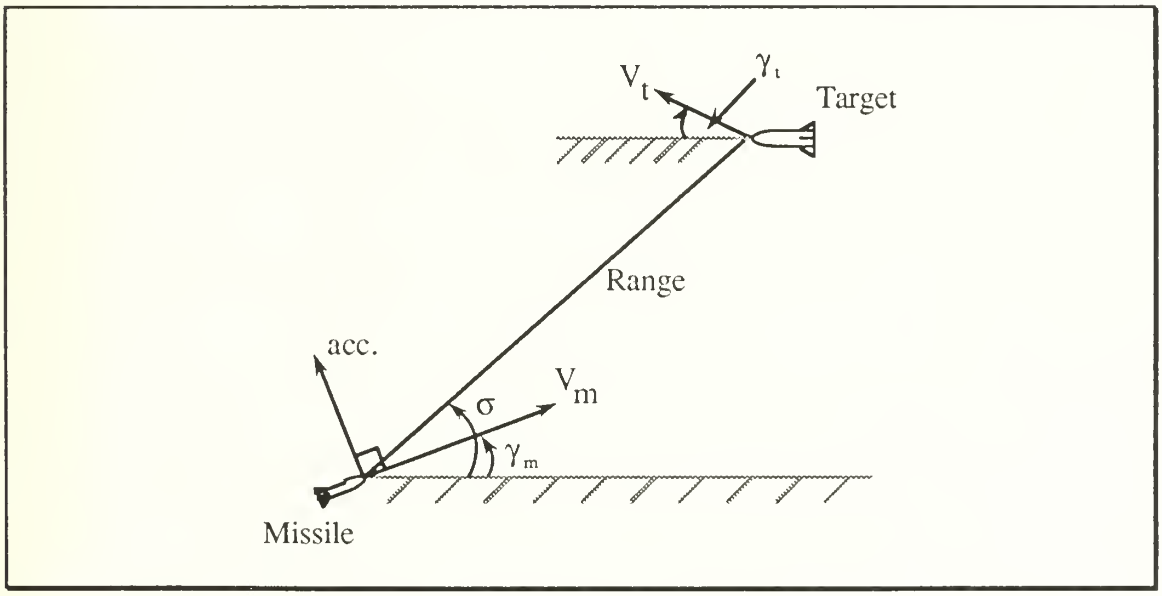

Figure 3.1 Missile/Target Geometry

It is assumed that the target is on the final leg of it's flight and is now on a straight, non-maneuvering trajectory. The control is to drive the line of sight rate ($\dot{\sigma}$) and it's derivative ($\ddot{\sigma}$) to zero in minimum time.

For our models we will be using only two dimensional scenarios. The system dynamics use second order models for each dimension. The target dynamics are non-maneuvering so that the acceleration, a_t, is zero. The missile acceleration, a_m, is assumed perpendicular to the velocity vector γ_m. The system dynamics are shown in the signal flow graph in Figure 3.2.

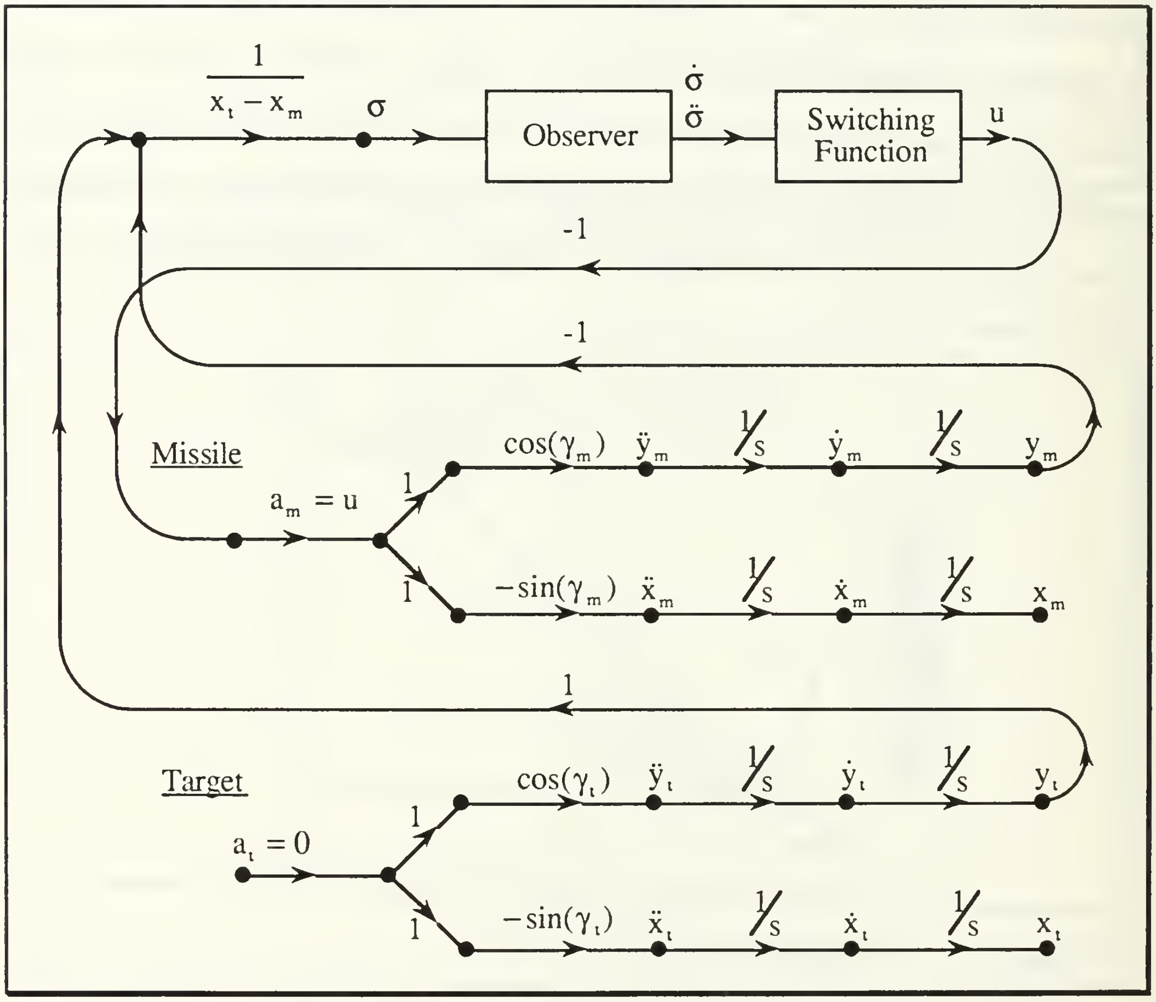

Figure 3.2 System Dynamics of the Intercept Model

The LOS angle, σ, and it's derivatives, $\dot{\sigma}$ and $\ddot{\sigma}$, may be derived analytically from the available states. We define the geometry of the system as in Figure 3.3.

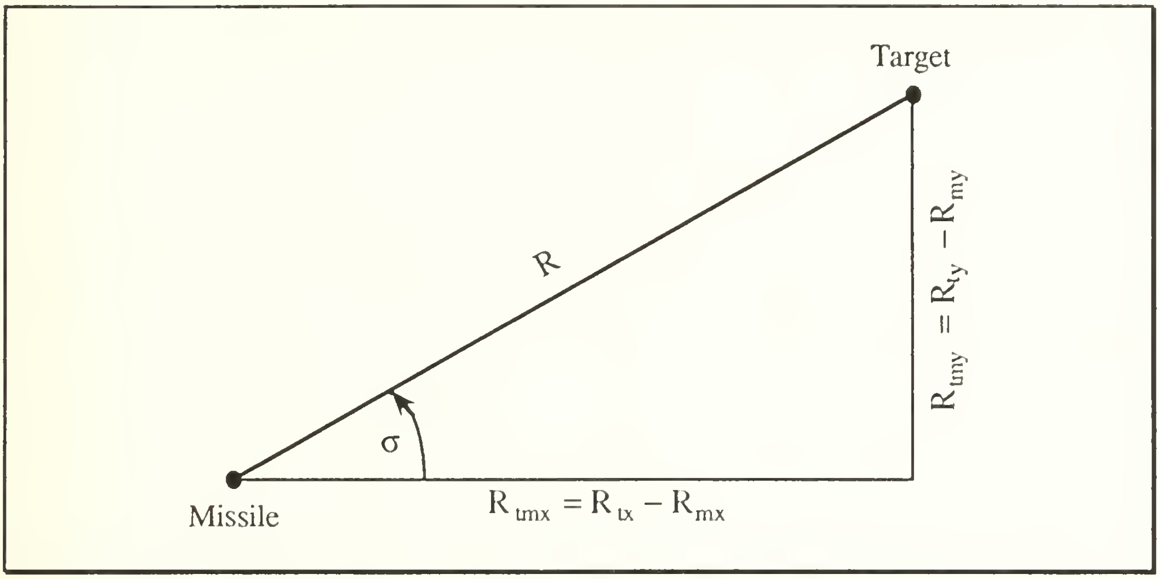

Figure 3.3 Geometry for Angle Definitions

with

R_t = Range of the Target.

R_m = Range of the Missile.

V_t = Velocity of the Target.

V_m = Velocity of the Missile.

a_t = Acceleration of the Target.

a_m = Acceleration of the Missile.

V_c = Closing Velocity.

so that

$$\sigma = \tan^{-1}\left(\frac{R_{tmy}}{R_{tmx}}\right) \tag{3.1}$$

$$\dot{\sigma} = \frac{R_{tmx} \cdot V_{tmy} - R_{tmy} \cdot V_{tmx}}{R^2} \tag{3.2}$$

and

$$\ddot{\sigma} = \frac{R_{tmx} \cdot a_{tmy} + 2V_{tmx} \cdot V_{tmy} - R_{tmy} \cdot a_{tmx}}{R^2} + \frac{2V_c\left(R_{tmx} \cdot V_{tmy} - R_{tmy} \cdot V_{tmx}\right)}{R^3} \tag{3.3}$$

where

$$V_c = -\dot{R} \tag{3.4}$$

In an actual application Kalman or Luenberger Observers may be used to obtain estimates of $\dot{\sigma}$ and $\ddot{\sigma}$.

B. APPLICATION OF THE SWITCHING LAW

The switching law from (2.43) is applied to our scenario where $\dot{\sigma}$ and $\ddot{\sigma}$ form the state space of this simulation, and is implemented as

$$u = N \cdot \text{sign}\left(\dot{\sigma} + \frac{\ddot{\sigma} \cdot |\ddot{\sigma}|}{2N}\right). \tag{3.5}$$

The sign convention of (2.43) was changed to conform to the geometry of our scenario.

We set the simulation with the initial conditions

$R_{mx}(0) = 0$ ft	$R_{my}(0) = 0$ ft
$V_{mx}(0) = 2000$ ft/sec	$V_{my}(0) = 0$ ft/sec
$a_{mx}(0) = 0$ ft/sec^2	$a_{my}(0) = 0$ ft/sec^2
$R_{tx}(0) = 10000$ ft	$R_{ty}(0) = 1000$ ft
$V_{tx}(0) = 2000$ ft/sec	$V_{ty}(0) = 0$ ft/sec
$a_{tx}(0) = 0$ ft/sec^2	$a_{ty}(0) = 0$ ft/sec^2
$t_{final} = 2.25$ sec	$dt = 0.01$ sec

Running the simulation we find that the controller does drive $\dot{\sigma}$ and $\ddot{\sigma}$ to zero, and maintains them about the origin until intercept is reached (Figure 3.4). Once the states reach the origin, the system controller goes into chatter mode, see Figure 3.5, and the trajectory chatters back and forth about the desired intercept path.

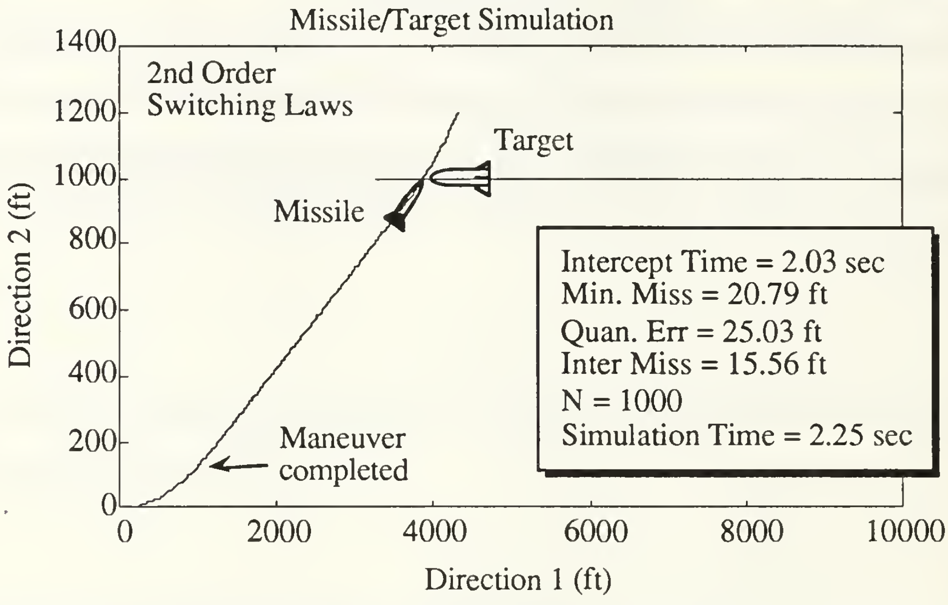

Figure 3.4 **Missile/Target Intercept Simulation Using Second Order Controller**

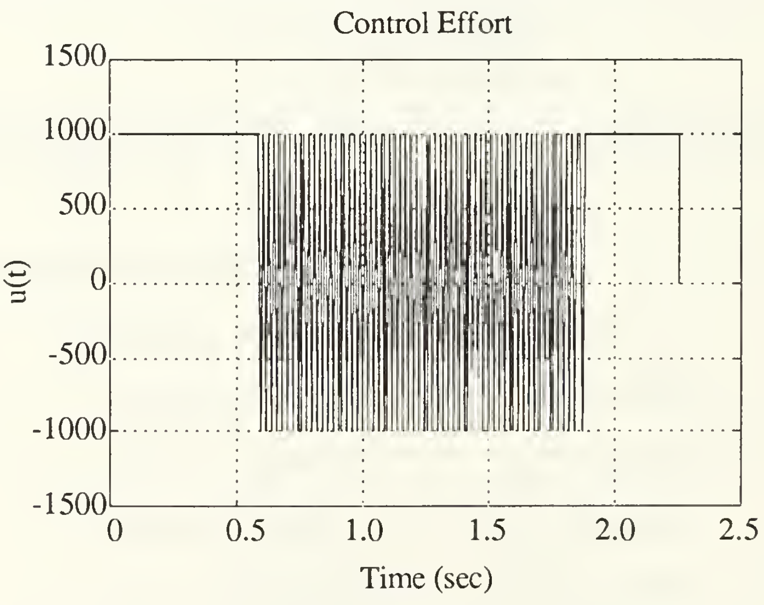

Figure 3.5 **Control Effort for Second Order Missile/Target Simulation**

C. COMPARISON WITH PROPORTIONAL NAVIGATION CONTROLLER

The system dynamics for the proportional navigation controller simulation is basically the same as that in the previous section, except that the control effort , u, is proportional to $\dot{\sigma}$, which is obtained with an estimator, shown in Figure 3.6, where $\dot{\beta} = \hat{\dot{\sigma}}$. We limit the acceleration so that u is bounded by $\pm N$. With only the change in the controller we ran the simulation for the same initial conditions with the results shown in Figures 3.7 and 3.8.

This Proportional Navigation system is effective at intercepting the target only when the missile/target geometry and kinematics are sufficient that the missile has time to maneuver. The delay in saturation of the control effort caused the missile to turn too slowly so that the Proportional Navigation Controlled system was unable to maneuver quickly enough to intercept the faster, although non-maneuvering, target.

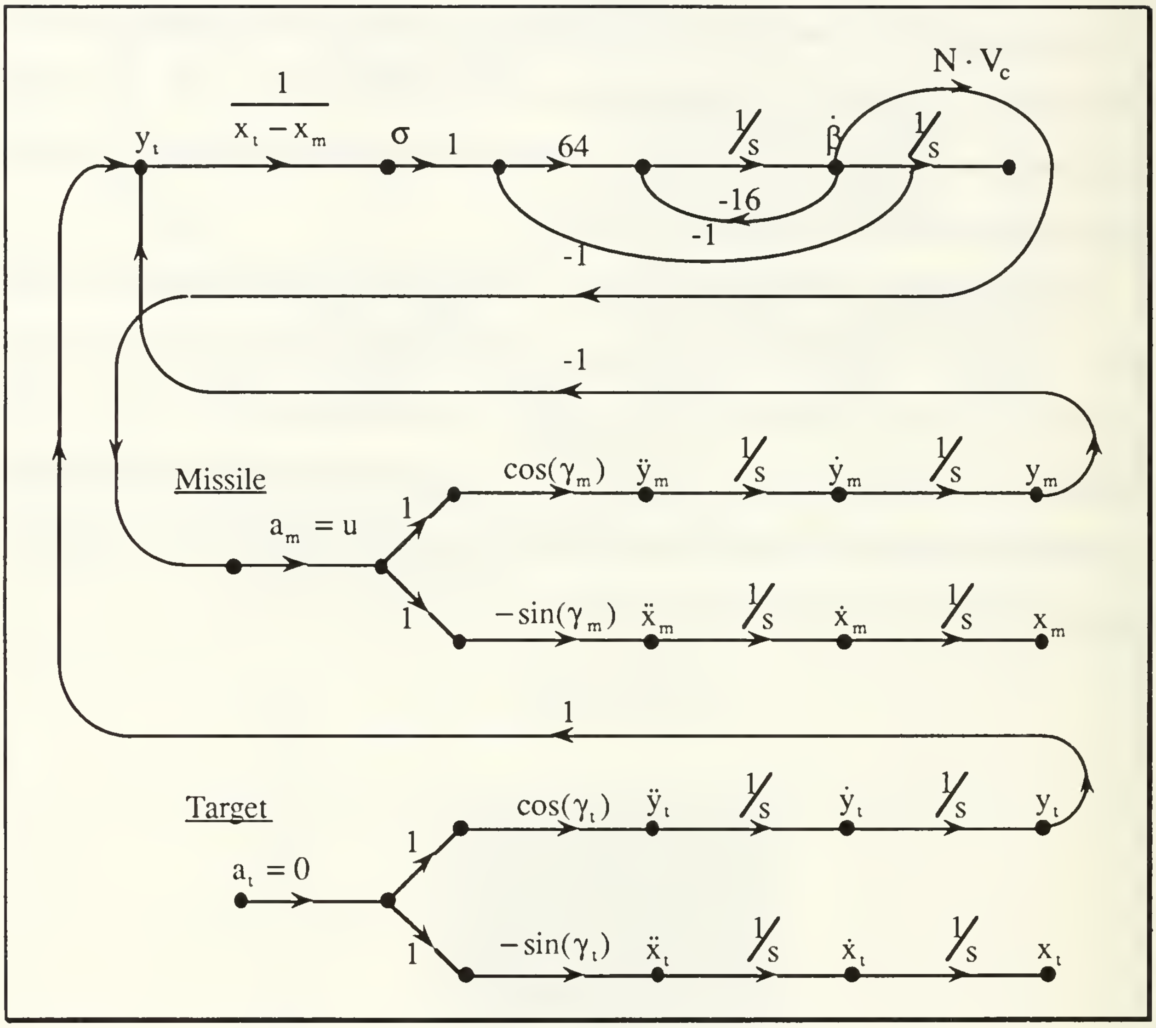

Figure 3.6 System Dynamics of Proportional Navigation Missile/Target Model

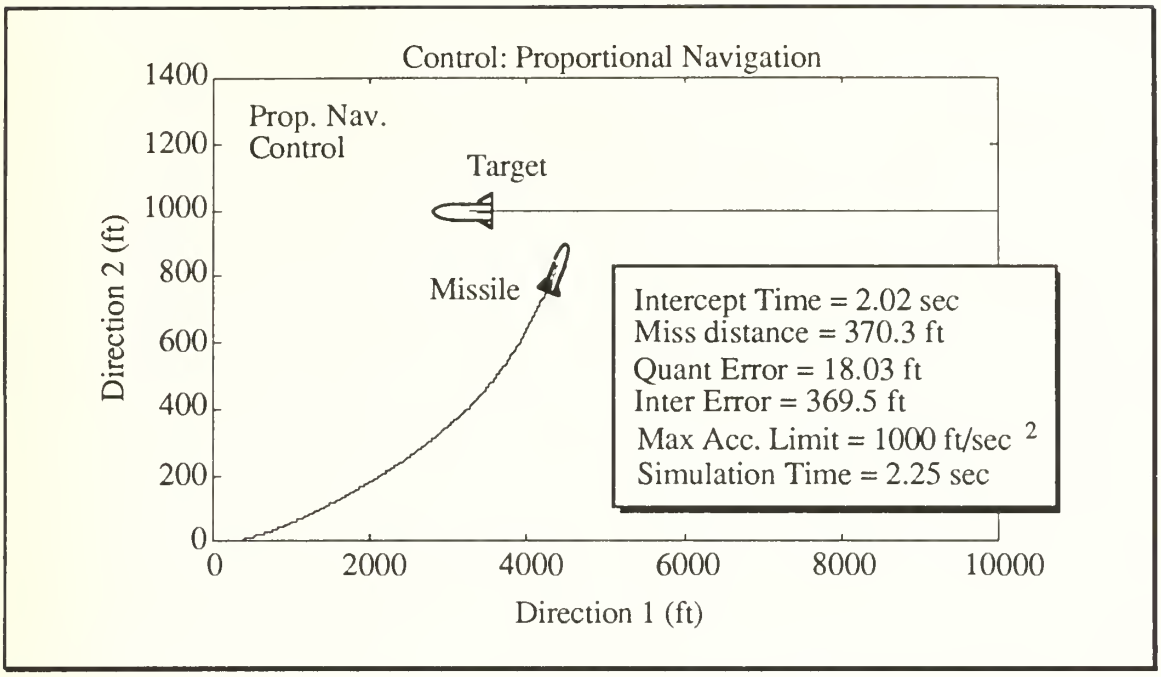

Figure 3.7 **Proportional Navigation Missile/Target Simulation**

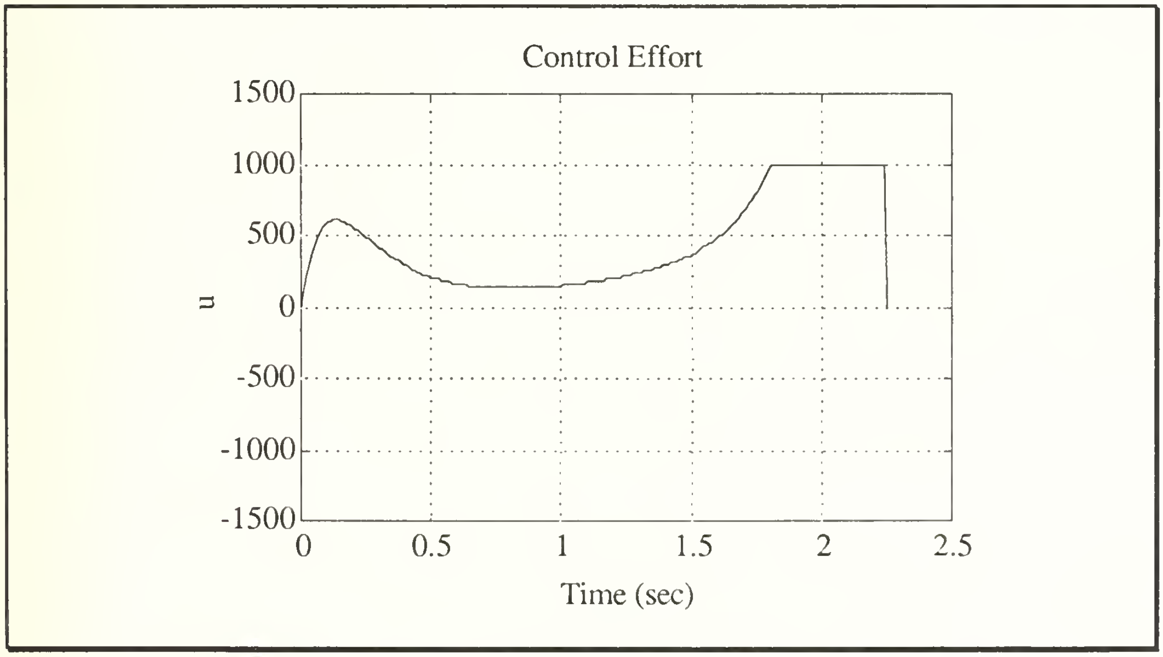

Figure 3.8 **Control Effort for Proportional Navigation Missile/Target Simulation**

IV. THIRD ORDER CONTROLLER

A. FORWARD TIME SYSTEM

The second order controller is effective at intercepting a target that has speed advantage. Previously we have controlled the system by driving $\dot{\sigma}$ and $\ddot{\sigma}$ to zero, and thereby maintaining a constant LOS angle, σ, until impact. But what about controlling the LOS angle itself? There are situations in which we would like to be able to attack a target from a particular angle, or perhaps have multiple missiles, each with it's own pre-defined attack angle.

Consider a point defence system in which the missile, launched from some point away from the target's final trajectory, would try to position itself in minimum time onto a "head-on" collision course with the target, as shown in Figure 4.1. In such a situation we would use a third order minimum time controller to drive σ, $\dot{\sigma}$, and $\ddot{\sigma}$ to zero.

1. System Definition

Our third order example is

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u. \tag{4.1}$$

Minimizing the Hamiltonian and solving the system we find

$$H = 1 + p_1 x_2 + p_2 x_3 + p_3 u \tag{4.2}$$

and

$$u = -N \cdot \operatorname{sign}(p_3) \tag{4.3}$$

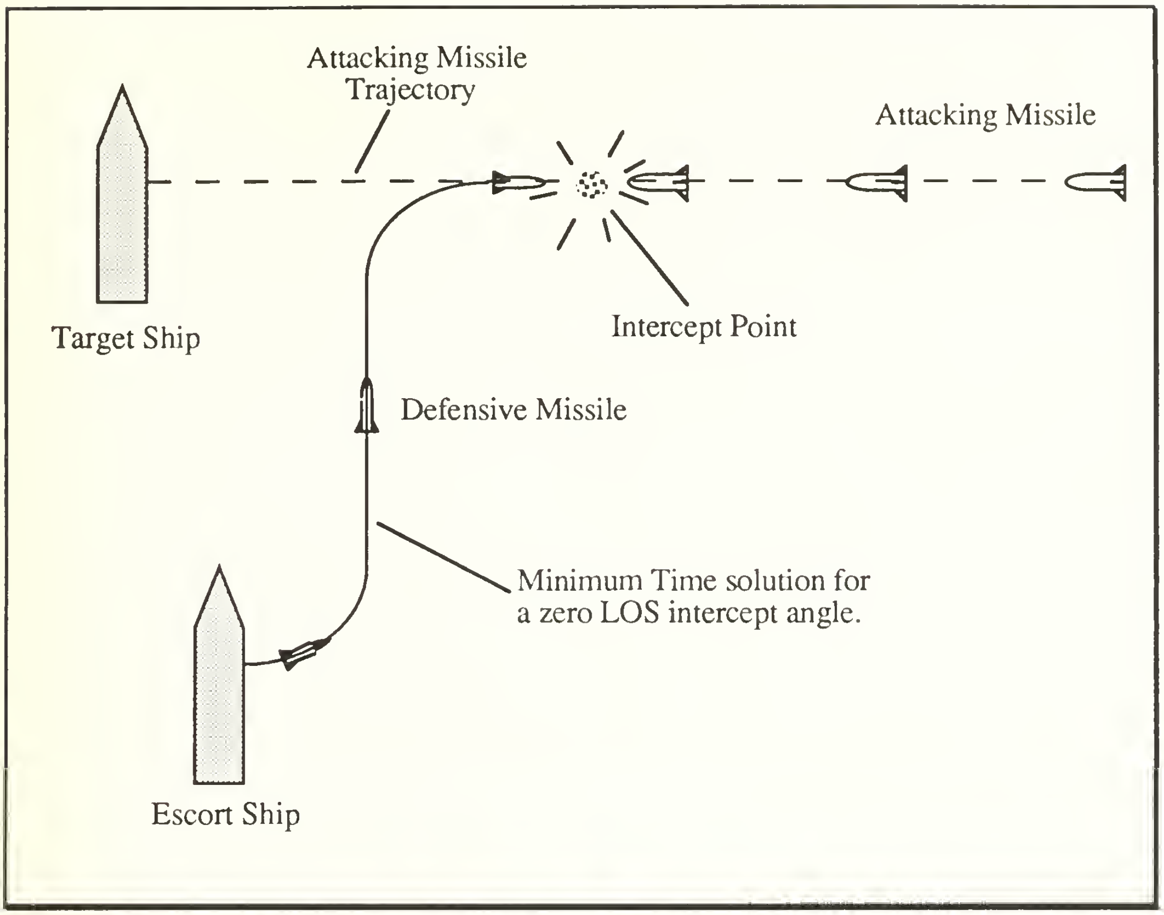
Attacking Missile Trajectory
Attacking Missile
Target Ship
Intercept Point
Defensive Missile
Minimum Time solution for a zero LOS intercept angle.
Escort Ship

Figure 4.1 Application of Third Order Minimum Time Controller

where

$$p = -\frac{\partial H}{\partial x} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} p \tag{4.4}$$

so that

$$p_1 = c_1 \tag{4.5}$$

$$p_2 = -c_1 t + c_2 \tag{4.6}$$

$$p_3 = \tfrac{1}{2} c_1 t^2 - c_2 t + c_3 . \tag{4.7}$$

From the adjoint solution the control may switch sign no more than twice. Again tracing this problem in negative time we may follow the zero trajectory curves out from the origin with control efforts of ±N. These curves are now in three dimensional space residing in their own plane which intersects the origin. Intersecting these zero trajectory curves are an infinite number of curves making a surface, and leading off from this surface the infinite number of trajectories lead to the initial conditions. Therefore in forward time we may start with an initial condition such that u = +N will drive the system to intersect with the surface at t_{sw1} as shown in Figure 4.2. Switching the control effort to u = -N will drive the system along the surface to intersect with the zero trajectory curve at t_{sw2} where the control effort switches again. Finally u = +N will drive the system to the origin.

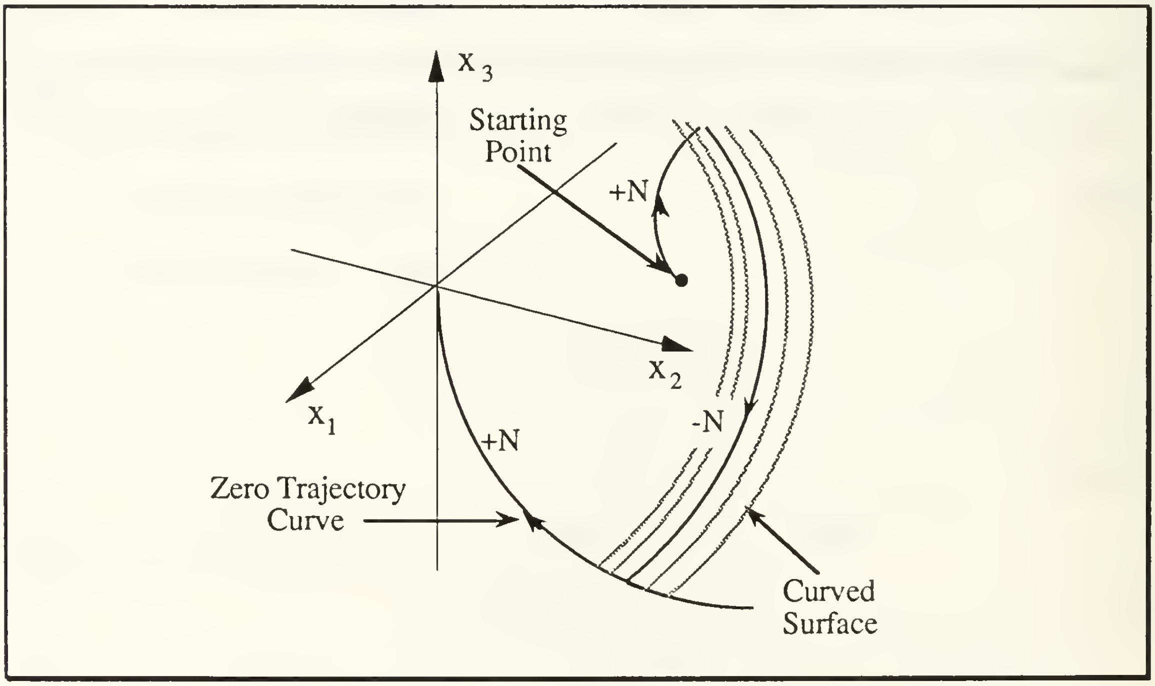

Figure 4.2 Three Dimensional Switching Curves

2. Third Order Switching Curves

We have determined from the second order system that solving for the control switching times is not as widely applicable as defining the control as a function of the states; so we now move on to defining the third order switching curves.

Starting with the state equations (4.1) we discretize and expand the equations so that

$$\phi = e^{At} = \left[I + At + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \dots \right]$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} t + \frac{1}{2!} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} t^2 + \dots$$
$$= \begin{bmatrix} 1 & t & \frac{1}{2}t^2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.8}$$

A^3 is the zero matrix so it and all higher terms of A are dropped.

$$\Delta = \int_0^t e^{A(t-\tau)} B \, d\tau = \int_0^t \begin{bmatrix} 1 & t-\tau & \frac{1}{2}(t-\tau)^2 \\ 0 & 1 & t-\tau \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} d\tau$$
$$= \int_0^t \begin{bmatrix} \frac{1}{2}(t-\tau)^2 \\ t-\tau \\ 1 \end{bmatrix} d\tau = \left. \begin{bmatrix} \frac{1}{2}t^2\tau - \frac{1}{2}t\tau^2 + \frac{1}{6}\tau^3 \\ t\tau - \frac{1}{2}\tau^2 \\ \tau \end{bmatrix} \right|_0^t = \begin{bmatrix} \frac{1}{6}t^3 \\ \frac{1}{2}t^2 \\ t \end{bmatrix}. \tag{4.9}$$

Combining these we obtain

$$x(t) = \begin{bmatrix} 1 & t & \frac{1}{2}t^2 \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} x(0) + \begin{bmatrix} \frac{1}{6}t^3 \\ \frac{1}{2}t^2 \\ t \end{bmatrix} u(0) \tag{4.10}$$

or in scalar equations

$$x_1(t) = x_1(0) + tx_2(0) + \tfrac{1}{2}t^2 x_3(0) + \tfrac{1}{6}t^3 u(0) \tag{4.11}$$

$$x_2(t) = x_2(0) + tx_3(0) + \tfrac{1}{2}t^2 u(0) \tag{4.12}$$

$$x_3(t) = x_3(0) + tu(0). \tag{4.13}$$

B. NEGATIVE TIME SYSTEM

The third order system, being piecewise continuous, is easily broken into several simple boundary value problems. In order to solve the systems of curves we must determine some boundary values for the intersections of these families of curves. We start at the origin and run the system in negative time to determine our other boundary conditions.

In negative time we have

$$\dot{x} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} u. \tag{4.14}$$

Discretizing we find

$$\phi = e^{At} = \begin{bmatrix} 1 & -t & \frac{1}{2}t^2 \\ 0 & 1 & -t \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

and

$$\Delta = \int_0^t e^{A(t-\tau)} B\, d\tau = \begin{bmatrix} -\frac{1}{6}t^3 \\ \frac{1}{2}t^2 \\ -t \end{bmatrix} \tag{4.16}$$

so that

$$x(t) = \begin{bmatrix} 1 & -t & \frac{1}{2}t^2 \\ 0 & 1 & -t \\ 0 & 0 & 1 \end{bmatrix} x(0) + \begin{bmatrix} -\frac{1}{6}t^3 \\ \frac{1}{2}t^2 \\ -t \end{bmatrix} u(0), \tag{4.17}$$

or in scalar equations

$$x_1(t) = x_1(0) - tx_2(0) + \tfrac{1}{2}t^2 x_3(0) - \tfrac{1}{6}t^3 u(0) \tag{4.18}$$

$$x_2(t) = x_2(0) - tx_3(0) + \tfrac{1}{2}t^2 u(0) \tag{4.19}$$

$$x_3(t) = x_3(0) - tu(0). \tag{4.20}$$

1. Solving for Negative Time Boundary Points

To develop a complete solution we solve the equations for several different points along the zero trajectory curve. Setting u = -N and traveling out along the zero trajectory curve for 1 second we find

$$x_1(1) = -\tfrac{1}{6}ut^3 = -\tfrac{1}{6}(-N) = \tfrac{1}{6}N \tag{4.21}$$

$$x_2(1) = \tfrac{1}{2}ut^2 = \tfrac{1}{2}(-N) = -\tfrac{1}{2}N \tag{4.22}$$

$$x_3(1) = -ut = -(-N) = N. \tag{4.23}$$

Similarly we run the system in negative time for 2 and 3 seconds to obtain other boundary points as shown in Figure 4.3, and listed in TABLE 1.

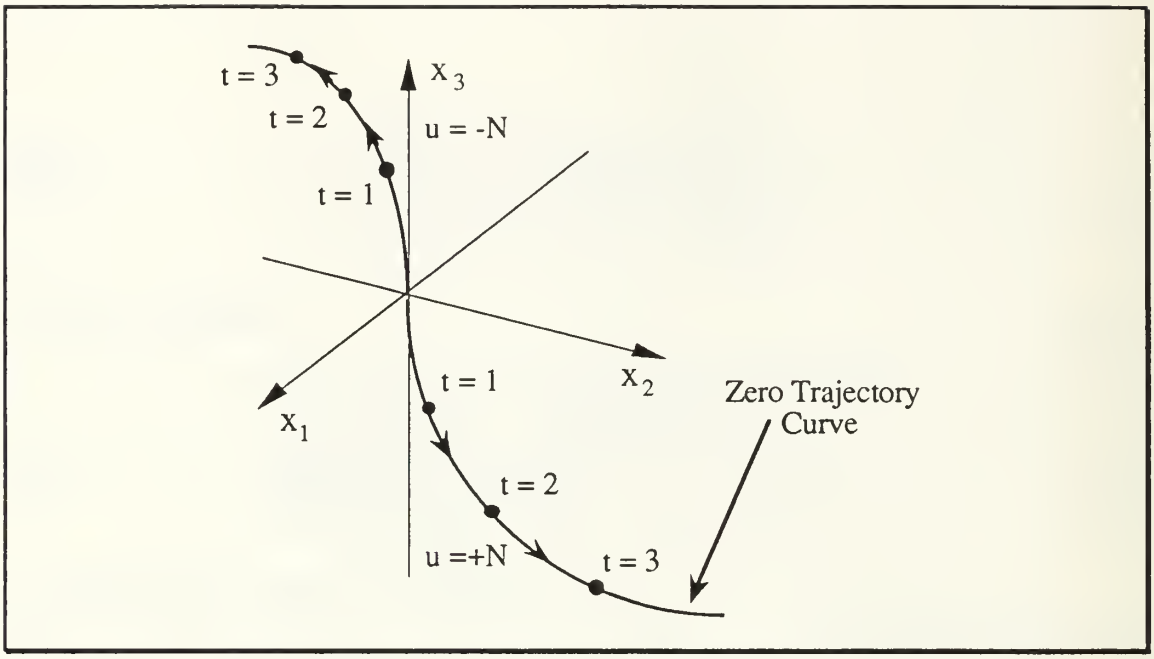

Figure 4.3 Boundary Values in Negative Time Solution

NEGATIVE TIME BOUNDARY CONDITIONS, u = -N

u = -N	t = 2	t = 3
$x_1(t)$	$\frac{4}{3}N$	$\frac{9}{2}N$
$x_2(t)$	-2N	$-\frac{9}{2}N$
$x_3(t)$	2N	3N

TABLE 1.

The control effort may also be u = +N, traveling away from the origin on the other zero trajectory curve giving the boundary conditions in TABLE 2.

NEGATIVE TIME BOUNDARY CONDITIONS, u = +N

u = +N	t = 1	t = 2	t = 3
$x_1(t)$	$-\frac{1}{6}N$	$\frac{4}{3}N$	$\frac{9}{2}N$
$x_2(t)$	$\frac{1}{2}N$	-2N	$-\frac{9}{2}N$
$x_3(t)$	-N	2N	3N

TABLE 2.

We may now solve the equations for the family of curves that intersect the zero trajectory curves. Specifically, we solve for the equation of the one curve that travels from some point $x_1(0)$, $x_2(0)$, $x_3(0)$, to the point $x_1(t) = \frac{1}{6}N$, $x_2(t) = \frac{1}{2}N$, $x_3(t) = N$. Since the control effort on the zero trajectory curve for this intersection point is u = -N, the control effort for the curve we are solving for must be u = +N, and the forward time equations may be written as

$$x_1(t) = \tfrac{1}{6}N = x_1(0) + tx_2(0) + \tfrac{1}{2}t^2x_3(0) + \tfrac{1}{6}t^3N \tag{4.24}$$

$$x_2(t) = -\tfrac{1}{2}N = x_2(0) + tx_3(0) + \tfrac{1}{2}t^2N \tag{4.25}$$

$$x_3(t) = N = x_3(0) + tN \tag{4.26}$$

Solving (4.26) for t we find

$$t = \frac{x_3(0)}{N} + 1. \tag{4.27}$$

Substituting (4.27) into (4.24) and (4.25), and after a bit of algebra we obtain

$$-N = x2(0) - \frac{x_3^{\,2}(0)}{2N} \tag{4.28}$$

$$0 = x_1(0) + x_2(0) - \frac{x_2(0)x_3(0)}{N} - \frac{x_3^{\,2}(0)}{2N} + \frac{x_3^{\,3}(0)}{3N^2}. \tag{4.29}$$

Using the other boundary values from TABLE 1 and TABLE 2 as well, we may generate a family of equations representing both sides of the zero trajectory curves, and different points of intersection along the curves (see TABLE 3). The control effort, u, in TABLE 3. is the control effort for the zero trajectory curve that is intercepted. The time, t, is the time out from the origin to the intercept point for the negative time system.

To combine all the equations from TABLE 3 into one solution we define

$$w = \operatorname{sign}\left(x_2 + \frac{x_3 \cdot |x_3|}{2N} \right) \tag{4.30}$$

and

$$f = x_2 + w \cdot \frac{x_3^2}{2N} \tag{4.31}$$

so that our family of curves is defined as

$$0 = x_1(0) + \frac{x_3{}^2(0)}{3N^2} + w \cdot \frac{x_2(0) \cdot x_3(0)}{N} + f \cdot \sqrt{\frac{f}{N}}. \tag{4.32}$$

Equation (4.30) determines which signs are to be applied based on which direction the zero trajectory curve is on; Equation (4.31) adjusts the magnitudes of the equation depending on the distance of the intersection of the zero trajectory curve from the origin. Each curve intersecting the zero trajectory curve will have a different value of the function f, and f will remain constant all along that curve.

FAMILY OF SOLUTION CURVES

u = -N
t = 1
	$-N = x2(0) - \frac{x_3^2(0)}{2N}$
	$0 = x_1(0) + x_2(0) - \frac{x_2(0)x_3(0)}{N} - \frac{x_3^2(0)}{2N} + \frac{x_3^3(0)}{3N^2}$

u = -N
t = 2
	$-4N = x2(0) - \frac{x_3^2(0)}{2N}$
	$0 = x_1(0) + 2x_2(0) - \frac{x_2(0)x_3(0)}{N} - \frac{x_3^2(0)}{N} + \frac{x_3^3(0)}{3N^2}$

u = -N
t = 3
	$-9N = x2(0) - \frac{x_3^2(0)}{2N}$
	$0 = x_1(0) + 3x_2(0) - \frac{x_2(0)x_3(0)}{N} - \frac{3x_3^2(0)}{2N} + \frac{x_3^3(0)}{3N^2}$

u = +N
t = 1
	$N = x2(0) + \frac{x_3^2(0)}{2N}$
	$0 = x_1(0) + x_2(0) + \frac{x_2(0)x_3(0)}{N} + \frac{x_3^2(0)}{2N} + \frac{x_3^3(0)}{3N^2}$

u = +N
t = 2
	$4N = x2(0) + \frac{x_3^2(0)}{2N}$
	$0 = x_1(0) + 2x_2(0) + \frac{x_2(0)x_3(0)}{N} + \frac{x_3^2(0)}{N} + \frac{x_3^3(0)}{3N^2}$

u = +N
t = 3
	$9N = x2(0) + \frac{x_3^2(0)}{2N}$
	$0 = x_1(0) + 3x_2(0) + \frac{x_2(0)x_3(0)}{N} + \frac{3x_3^2(0)}{2N} + \frac{x_3^3(0)}{3N^2}$

TABLE 3.

C. THIRD ORDER SWITCHING LAW

Using (4.30), (4.31) and (4.32) we can break up space into areas of opposite control effort. The control effort is defined by which of these volumes contain the states. Therefore, the third order switching law is defined as

$$u = -N \cdot \mathrm{sign}\left(x_1(0) + \frac{x_3{}^2(0)}{3N^2} + w \cdot \frac{x_2(0) \cdot x_3(0)}{N} + f \cdot \sqrt{\frac{f}{N}} \right). \tag{4.33}$$

D. THIRD ORDER CONTROLLER SIMULATION

A simulation of this third order model shows that the third order switching law, (4.33), drives the states to the origin with only 2 changes is the control effort. In application some means of shutting off the control effort will be required for the system to avoid entering chatter mode upon reaching the origin (see Figure 4.4).

An examination of the functions (4.30), (4.31), and (4.32) in Figure 4.5 shows that f is a smoothly increasing curve until the states intercept the switching surface. Once the states are following the surface, f remains a constant value. When the states reach the zero trajectory curve, f becomes approximately zero.

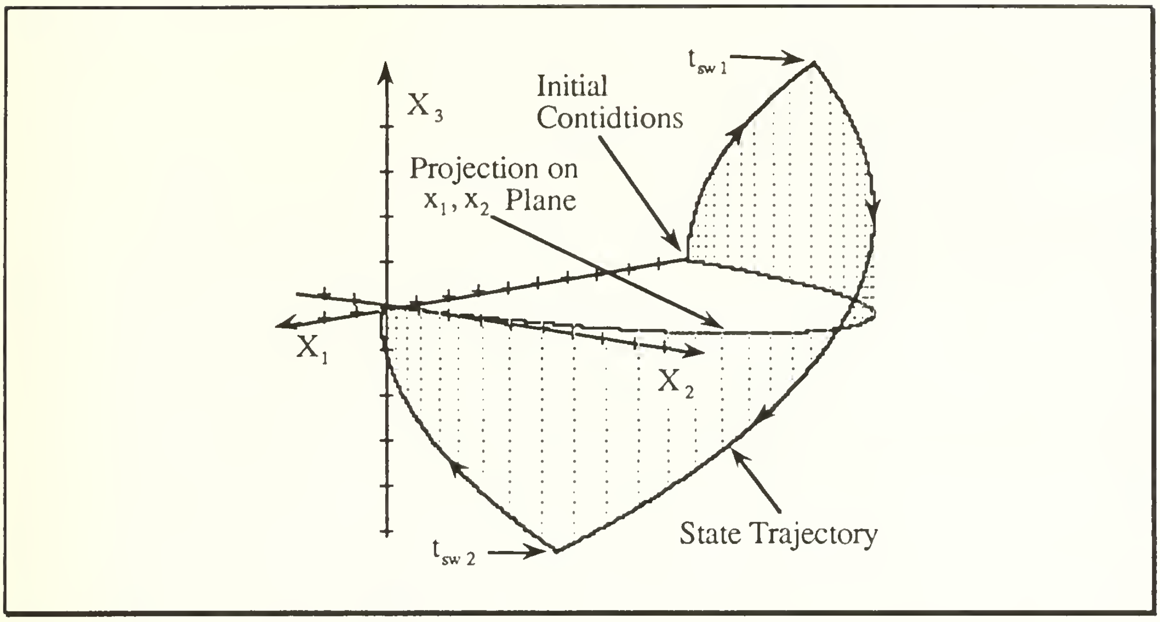

Figure 4.4 Simulation of a Third Order Minimum Time Controller From a Fixed Initial Condition

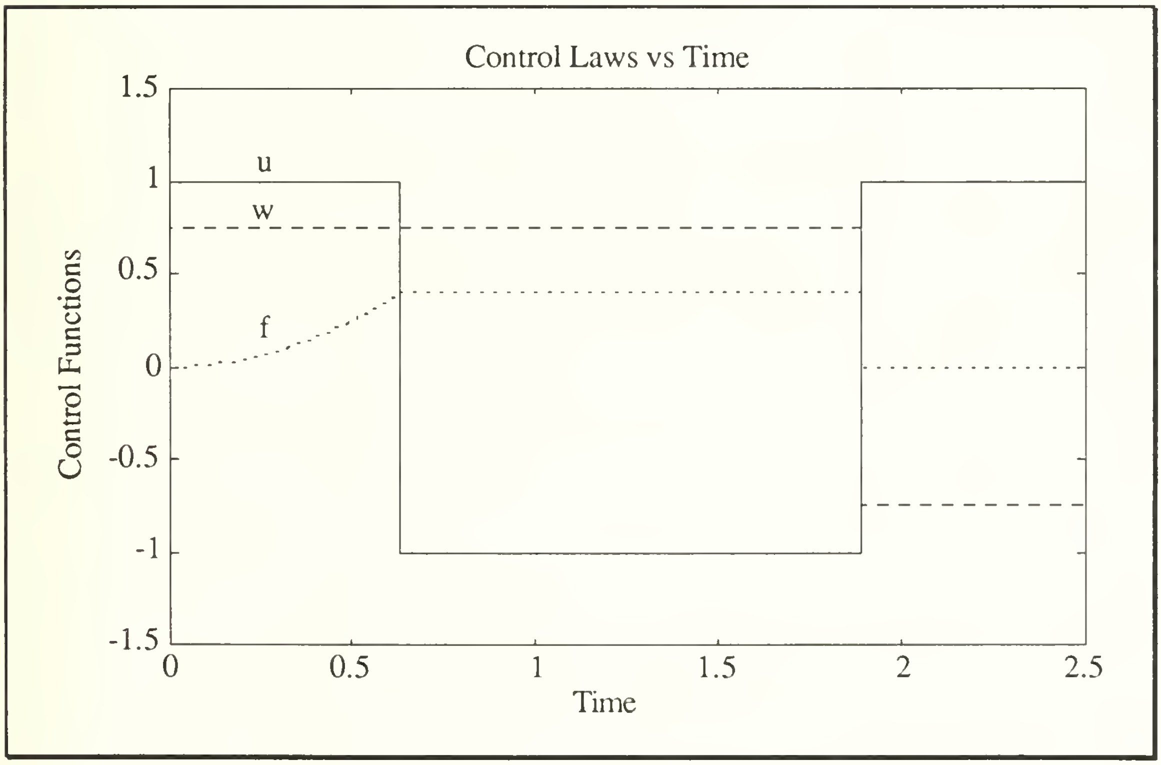

Figure 4.5 Control Laws For Third Order Solution

V. MISSILE ADJOINTS

Given a time varying system

$$\dot{x} = A(t)x + B(t)u \tag{5.1}$$

$$y = C(t)x. \tag{5.2}$$

it can be shown that the impulse response of the adjoint system

$$\dot{p} = A(t_f - t)^T p + C(t_f - t)^T r \tag{5.3}$$

$$y = B(t_f - t)^T p \tag{5.4}$$

is the response of the original system, at time t, to an impulse applied at time $t_f - t$ before t. [2] For a time invariant system the transfer function for the original system is identical to the transfer function of the adjoint system, i.e., they are self-adjoint.

A. CONSTRUCTION OF AN ADJOINT

Given the mathematics of the adjoint method, we now need to define some rules to create and understand the adjoint system with real systems and block diagrams. [3]

1. Rule 1: Convert All System Inputs to Impulses

In constructing the adjoint it is necessary that all system inputs be impulsive. Since this may not be the case with the block diagram, all system inputs and initial conditions must be converted to impulsive inputs via block manipulations and extensive use of integrators.. Figure 5.1 shows that step inputs and initial conditions are equivalent to the output of impulse driven integrators.

2. Rule 2: Replace t With t_f - t in the Arguments of All Time Varying Coefficients

In many linear systems it is possible to express a gain as a function of time. The adjoint system operates in negative time and Figure 5.2 shows the conversion of functions of time into the adjoint domain.

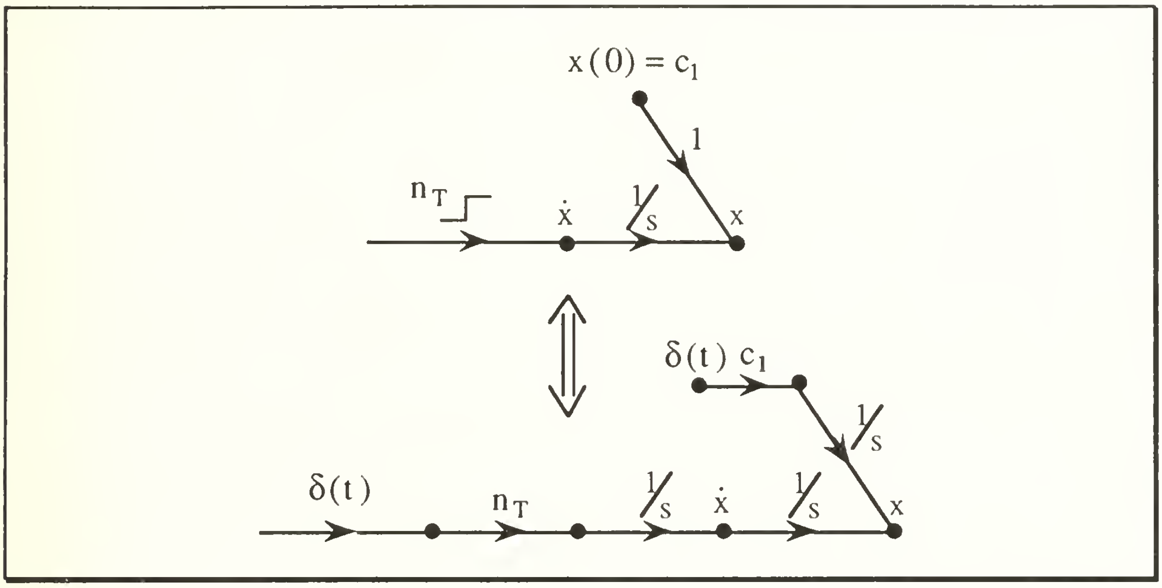

Figure 5.1 Conversion to Impulsive Inputs

	Original System	Adjoint System
Time Varying Gain	$K(t) = at + b$	$K(t - t_f) = a(t - t_f) + b$
	$K(t) = \dfrac{1}{a(t - t_f) + b}$	$K(t - t_f) = \dfrac{1}{at + b}$

Figure 5.2 Convert Functions of Time to the Adjoint Domain

3. Rule 3: Reverse the Direction of all Signal Flow

The direction of all signal flow must be reversed, redefining nodes as summing junctions and visa versa. Notice that all system outputs become inputs and all system inputs become outputs. This last rule allows for the simple graphical creation of the adjoint system by first drawing the block, or flow, diagram of the original system, then redrawing it with the arrows reversed. Figure 5.3 shows some examples f converting nodes to summing junctions and visa versa.

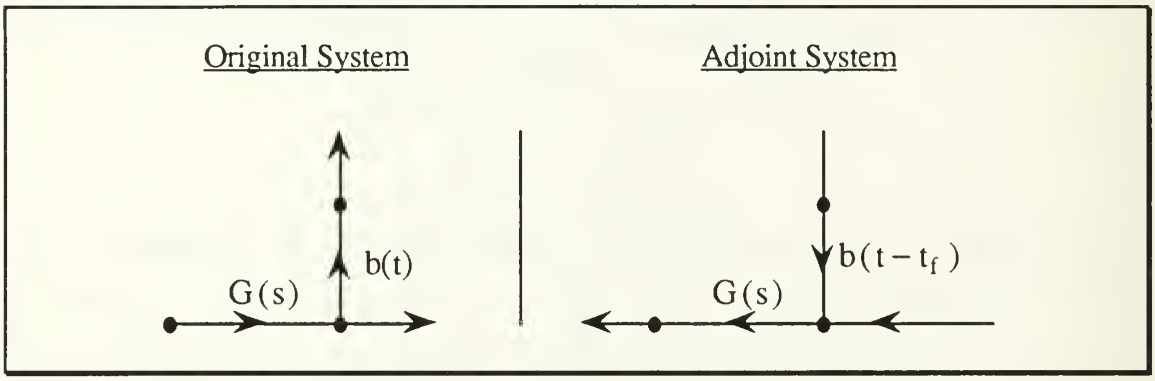

Figure 5.3 Converting Nodes to Summing Junctions

B. DEVELOPMENT OF A SECOND ORDER ADJOINT SYSTEM

This adjoint formulation lends itself well to analyzing some optimization problems, those where t_f is free and the terminal state is constrained to a point, curve or surface. From Pontryagin the control is a function of the homogeneous adjoint (see (4.2) and (4.3)). The missile adjoint form gives also a particular solution of the adjoint (i.e. system impulse response).

The adjoint allows one to generate optimum solutions (i.e. switching surfaces) for all possible initial conditions. The forcing impulses passed through an integrator gives our saturation type of optimal control ($\pm$N).

As an example for the application of the method of adjoints we will apply our adjoint rules to our second order controller. The system described by (2.9) has a time dependent control input where

$$u(t) = \begin{cases} -N & t < t_s \\ +N & t > t_s \end{cases} \tag{5.5}$$

and, from our second order example,

$$t_s = \sqrt{\frac{x(0)}{N} + \frac{1}{2}\left(\frac{\dot{x}(0)}{N}\right)^2} + \frac{\dot{x}(0)}{N}. \tag{5.6}$$

1. Apply Rule 1 to the Second Order Example

For this example we will define N = 1. We first draw out the original signal flow diagram, showing all the inputs and initial conditions. In order to apply the impulsive inputs we expand the second order system to the third order state equations

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \tag{5.7}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x. \tag{5.8}$$

Converting all the inputs to impulsive inputs and initial conditions we get the flow diagram in Figure 5.4.

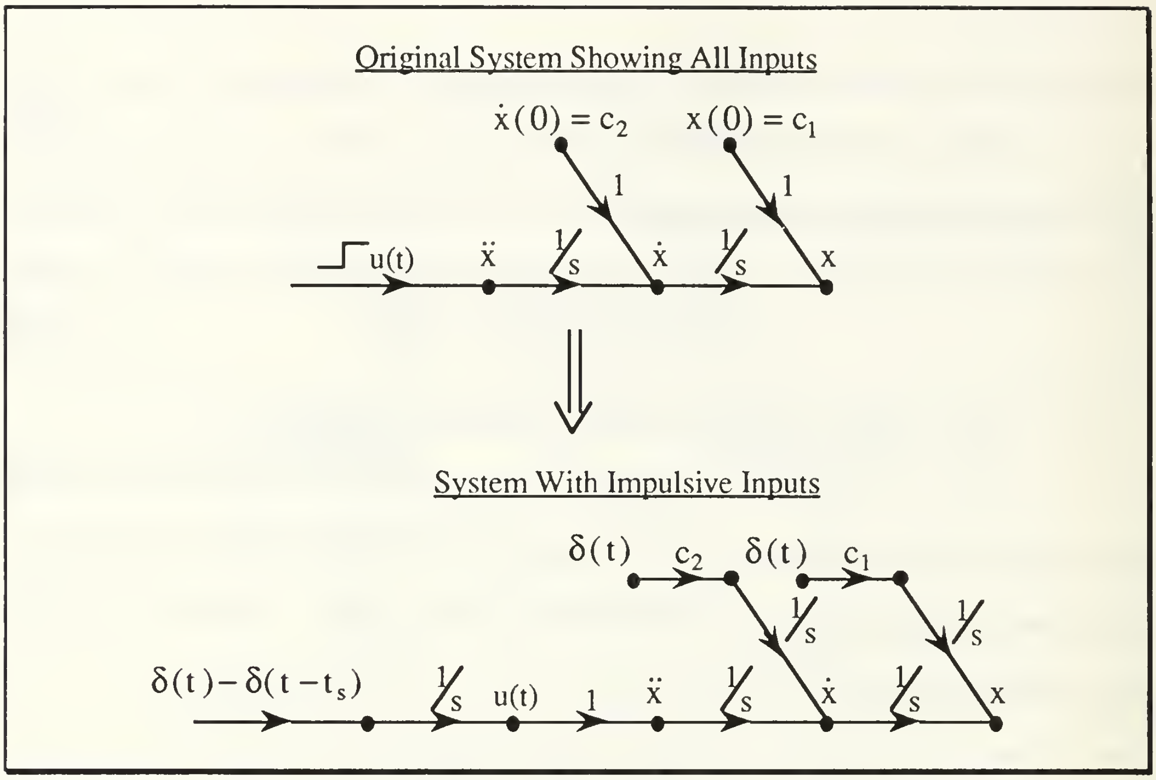

Figure 5.4 Application of Rule 1 on Second Order Example

2. Apply Rule 2 to the Second Order Example

We replace t with t - t_f in the arguments of all time varying coefficients to obtain the flow diagram in Figure 5.5.

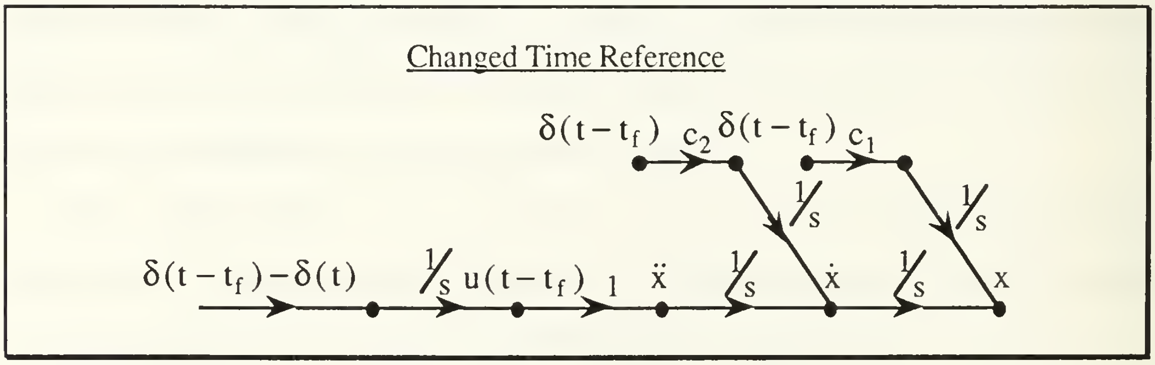

Figure 5.5 Application of Rule 2 on Second Order Example

3. Apply Rule 3 to the Second Order Example

Finally we reverse the direction of the signal flow, redefining nodes as summing junctions and visa versa, thereby changing the system inputs to outputs and the system outputs to inputs as shown in Figure 5.6.

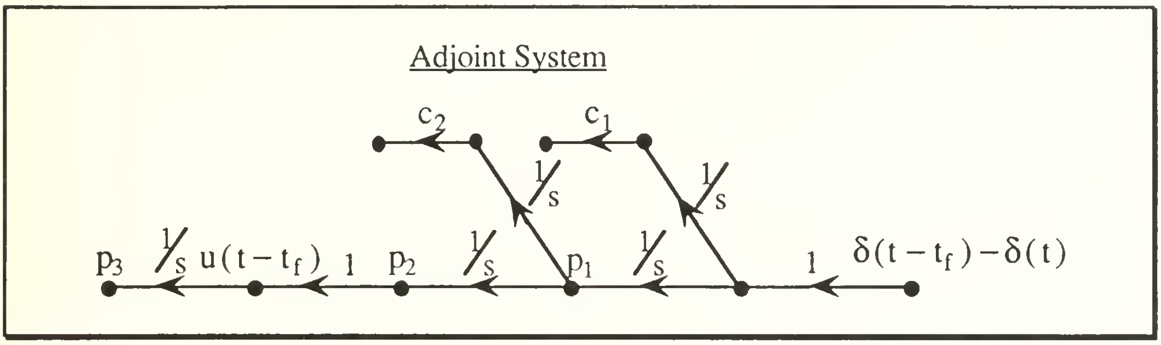

Figure 5.6 Application of Rule 3 on Second Order Example

The mathematical definition of the Adjoint System is

$$\dot{p} = A^T p + C^T u \tag{5.9}$$

$$y = B^T p. \tag{5.10}$$

Using the A, B, and C matrix from (5.7) and (5.8) we obtain

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \\ \dot{p}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(t-t_f) \tag{5.11}$$

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} p \tag{5.12}$$

which corresponds to Figure 5.6.

C. SIMULATION OF THE SECOND ORDER ADJOINT

1. Forward Time Second Order Simulation

In the forward time second order simulation we set N = 1, and the initial conditions $x(0) = 2$ and $\dot{x}(0) = 0$. Using impulses through integrators we apply the initial control effort at t = 0. A second impulse, opposite in sign and twice in magnitude, is applied at the switching time, as defined in (5.6), driving the states through the origin.

The simulation length is 4 seconds with a sample step of .05 seconds. The results are shown in Figure 5.7. Starting at the initial conditions the system is driven through the origin with a minimum miss distance of 0.003344 at a time of 2.83 sec.

2. Adjoint Solution for Second Order System

In the adjoint domain the system will travel in negative time starting at the origin and traveling outward to the initial conditions for the forward time system. Because this is a time invariant system the trajectory for the adjoint solution will be the same as the forward time system but in the opposite direction. From our second order example, the optimum switching in negative time from the terminal state at the origin is

$$t_f - t_s = \frac{|x_2(t_s)|}{N}. \tag{5.13}$$

The trajectory constraint yields

$$x_1(t_s) = -\frac{x_2(t_s)|x_2(t_s)|}{2N}. \tag{5.14}$$

A reverse impulse of twice the magnitude is applied and the trajectory proceeds out to the desired initial conditions given. The negative time impulse response is thus prescribed by using the switching times

$$t_f - t_s = \frac{1}{N}\sqrt{N \cdot x_1(0) + \tfrac{1}{2}x_2(0)^2} \tag{5.15}$$

$$t_{sw} - t_0 = \frac{1}{N}\sqrt{N \cdot x_1(0) + \tfrac{1}{2} x_2(0)^2} + \frac{x_2(0)}{N}. \qquad (5.16)$$

The parameters for the adjoint solution are the same as for the forward time, but the initial values for the states are at the origin. The output of the adjoint system, according to (5.12), is p_3, so the phase plot of Figure (5.8) plots $-p_2$ and p_3. The adjoint solution traces almost the same path as the forward time solution, switching at $t_s = 1.582$ sec., and missing the point (2,1) by 0.002733 at $t_f = 2.83$ seconds. By changing the initial sign of the control effort, and adjusting the switching time, we could drive the adjoint system to any desired point in the phase plane, and therefore know the optimal solution for the forward time system.

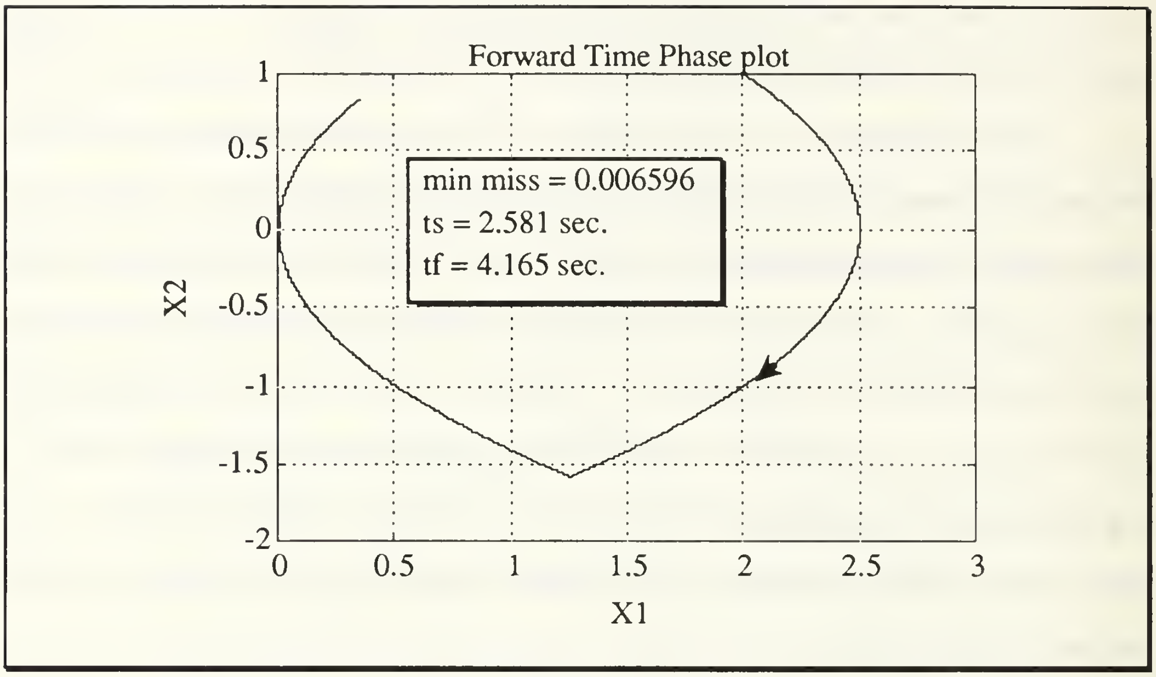

Figure 5.7 Simulation of Forward Time System

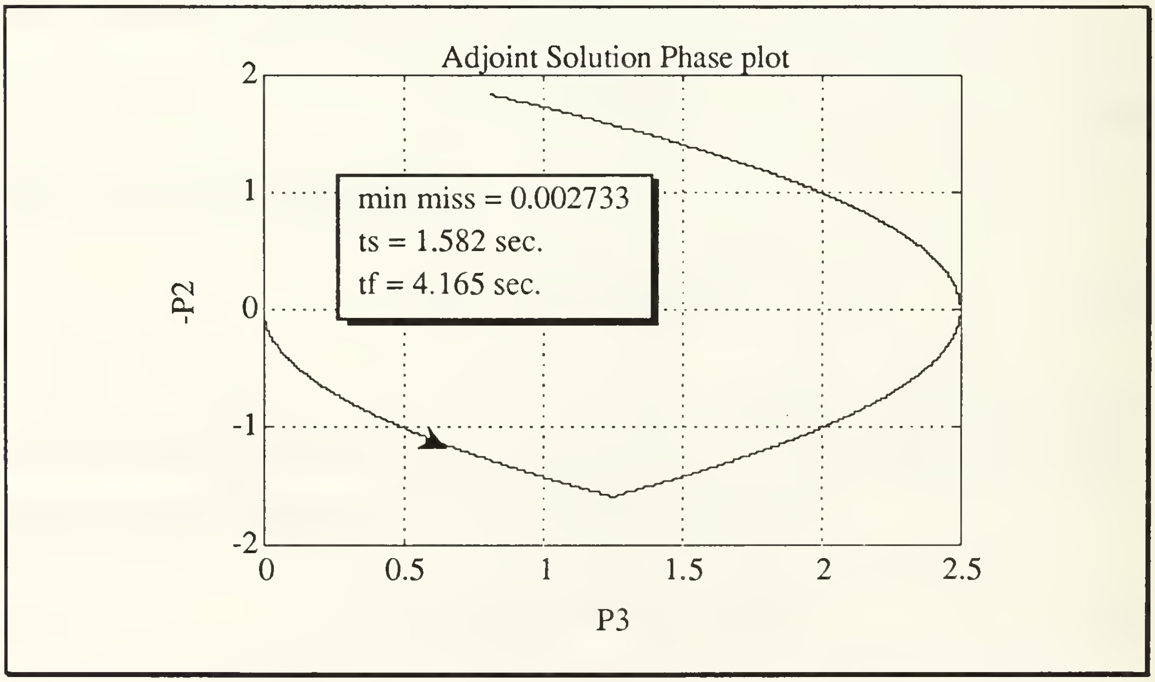

Figure 5.8 Simulation of the Adjoint Solution

D. MISSILE SIMULATION WITH ADJOINTS

The method of adjoints can be extremely useful when dealing with time varying systems. By using the adjoint solution we are able to see how the forward time system behaves for all times.

1. Missile/Target Model

We will simplify our scenario of missile/target engagement in order to present an uncluttered example of the method. The target will have zero x-velocity and constant y-velocity, and will start on the x-axis at a distance, R, from the origin (see Figure 5.9). The missile will start at the origin with a constant x-velocity by zero initial y-velocity.

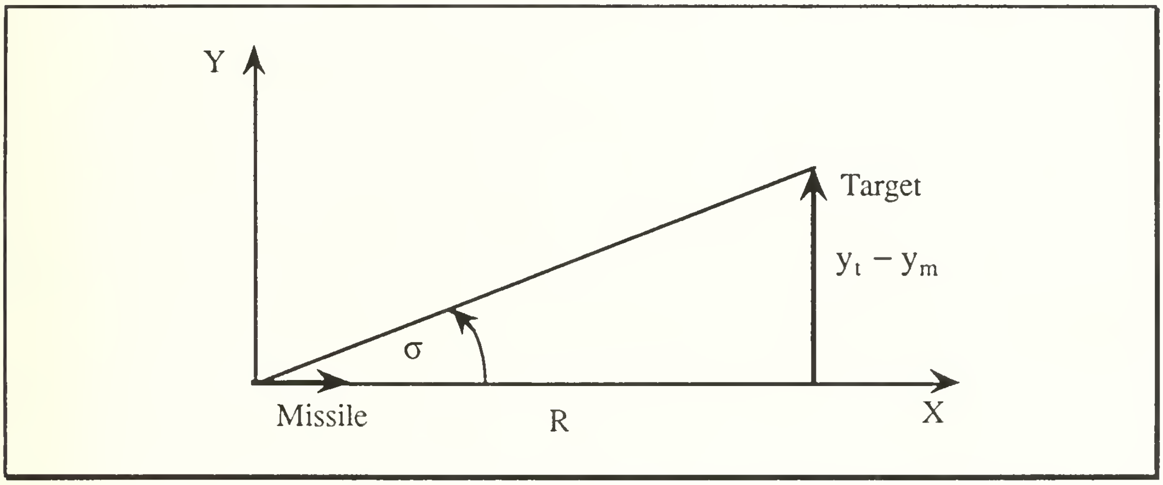

Figure 5.9 Geometry of Missile/Target Adjoint Solution

Since R is large and y_t - y_m is small we use the small angle approximation and define

$$\sigma = \tan^{-1}\left(\frac{y_t - y_m}{R}\right) \cong \frac{y_t - y_m}{R}. \tag{5.17}$$

The closing velocity is constant so we note that

$$R = V_c\left(t_f - t\right) \tag{5.18}$$

where t_f is the length of the simulation.

2. Signal Flow Diagram

Because the x-velocities are constant we need only to model the y-dimension. We develop the flow diagram in Figure 5.10 from the geometry of Figure 5.9. We use proportional navigation control and impulsive inputs are used for initial conditions.

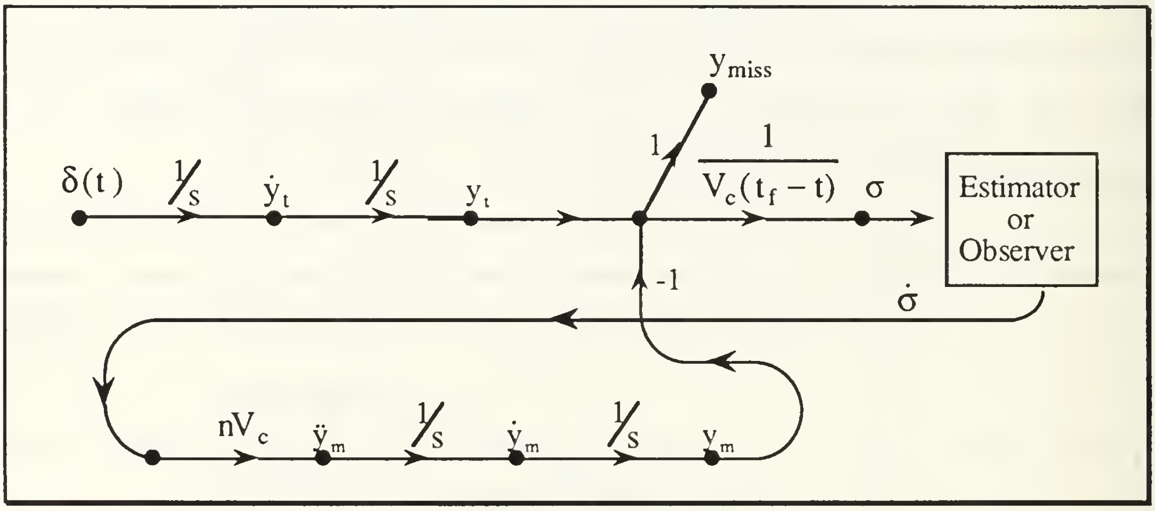

Figure 5.10 Signal Flow Diagram for Missile/Target Model

The estimator used was developed from the mechanics and dynamics of a seeker head system, however, it can be shown to be equivalent to a Kalman Estimator or Luenberger Observer. This estimator in Figure 5.11 has a time constant of 0.1 seconds. The primary interest in this model is to determine the miss distance at the final time, $t = t_f$, so the output of the system is y_{miss}.

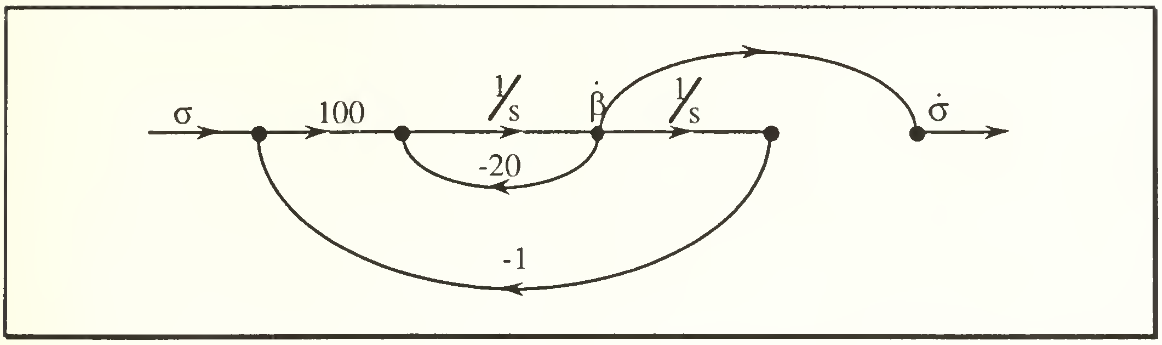

Figure 5.11 Signal Flow Diagram of the Estimator

3. Forward Time Simulation

We define our states to be

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} y_m \\ \dot{y}_m \\ \beta \\ \dot{\beta} \\ y_t \\ \dot{y}_t \end{bmatrix} \tag{5.19}$$

and the system state equations are then

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & nV_c & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \dfrac{-100}{t_f - t} & 0 & -100 & -20 & \dfrac{100}{t_f - t} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot u \tag{5.20}$$

$$y = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot x. \tag{5.21}$$

Since the x-velocities are constant we know approximately when the point of closest approach will occur, and therefore about how long to run the simulation. The LOS angle, σ, is a function of t and t_f, so that different values of t_f will result in changes to the control effort and system response. Because this state matrix, A, is time dependent , it must be redefined at each time step of the simulation. We define our proportionality constant, n, to 4, the closing velocity, V_c, to 5000 ft/sec., so that with $t_f = 4$ sec. the initial range is 20,000 ft. The output of the simulation is shown in Figure 5.12.

To study the effect of different values of t_f, or to locate the optimal value, we may have to run the forward time simulation many times, which could be a very tedious process, especially since we are only interested in the final value of $t = t_f$.

4. Adjoint Solution

An alternative to running the forward time system over and over again would be to run the adjoint solution once to generate the final values of the family of forward time solutions. This time we generate the adjoint system using matrix algebra instead of the graphical method. The solutions to the adjoint system are, from (5.9) and (5.10)

$$\dot{p} = \begin{bmatrix} 0 & 0 & 0 & \dfrac{-100}{t_f - t} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -100 & 0 & 0 \\ 0 & nV_c & 1 & -20 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{100}{t_f - t} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot p + \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \cdot u \qquad (5.22)$$

$$y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot p. \qquad (5.23)$$

This system generates a curve whose value at any time, t_a, is the final value of the forward time system where $t_f = t_a$. Figure 5.13 shows four curves from the forward time solution corresponding to t_f = 1,2,3, and 4 sec. Each curve end exactly on the adjoint solution curve at the corresponding adjoint time.

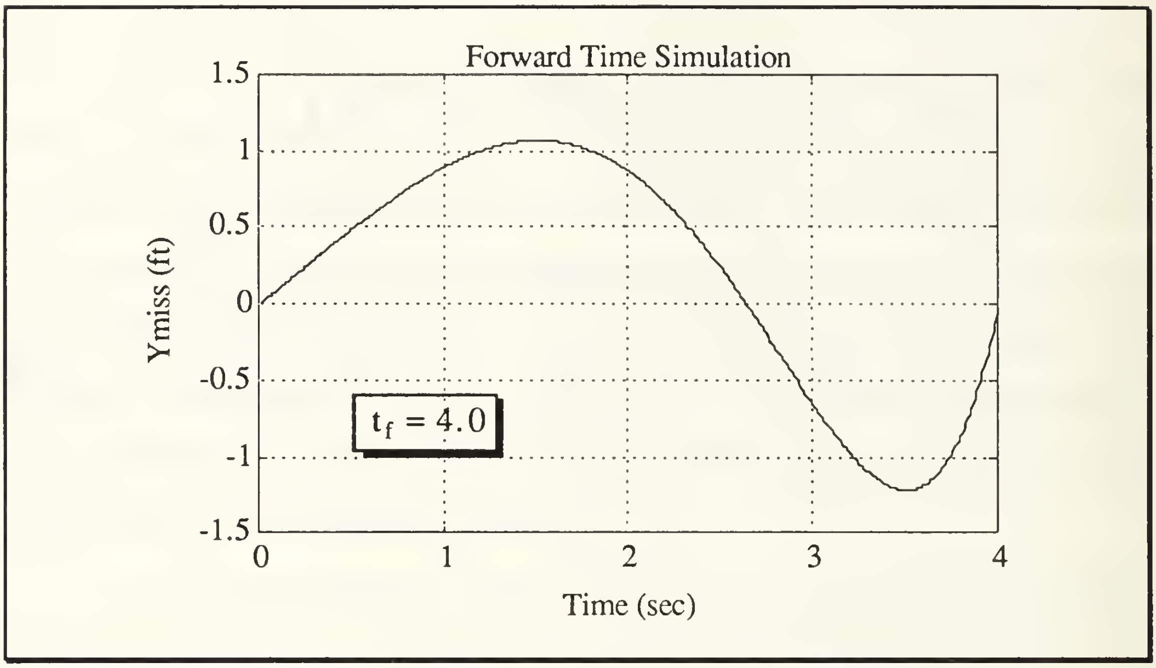

Figure 5.12 Forward Time Simulation

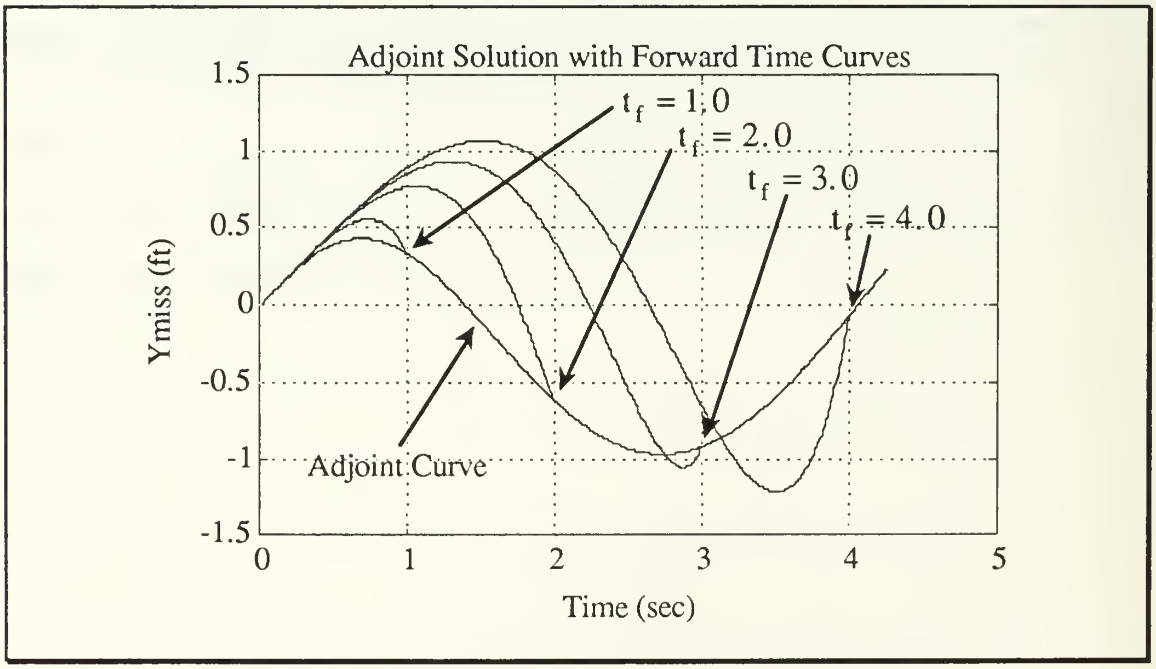

Figure 5.13 Adjoint Solution Curve

VI. CONCLUSIONS AND COMMENTS

We have developed the second order minimum time optimal controller and applied it to the fast reaction missile defense problem with both the closed form analytic solution, and the open form switching curves solution. In comparing our open form solution with a standard proportional navigation controller solution we demonstrate an increased maneuverability enabling our missiles to defend against faster, more capable threats. The accuracy at intercept is a function of the control logic used to shut off the control effort when desired conditions are met. This is a subject that should be explored in future projects.

We introduced a third order minimum time controller which promises to not only improve reaction time and maneuverability, but also presents us with the ability to control and define a desired attack angle for an improved destructive potential. A model for a practical third order controller should be developed and evaluated.

Having introduced the method of adjoints and shown some of its functions, we hope to stimulate some more practical applications of this technique. We are excited at the possibilities of using the method of adjoints in determining optimal, closed form solutions to forward time problems at speeds fast enough for practical implementation.

APPENDIX A. PROGRAM CODE

All of the simulations for this project were run on both IBM-AT[1] class and Macintosh II[2] computers using the matrix manipulation language MATLAB[3]. For IBM-AT compatibles MATLAB, version 3.5f was used, and on the Macintosh II computers version MAC II-MATLAB, version 1.1b. This appendix contains the source code for all of the simulations and functions written in support of this project.

Only a limited background in programming is required for understanding these files. While MATLAB is similar to FORTRAN, MATLAB's control structures are much less complex, and with matrix manipulation built into the system, vector definition and storage are greatly simplified. Comments are started by the percent sign (%) and continue to the end of the line. Ellipsis (...) at the end of a line indicate the continuance of the logical line onto the next line of code.

Each file will begin on a new page to assist those who are interested in examining or reproducing the code. Although an analysis of these files is not necessary to understand this report, readers are encouraged to examine them closely for further information.

The code is presented in the order of usage in the main text with all supporting functions grouped with the main program of interest.

---

[1] IBM and IBM-AT are registered trademarks of IBM.

[2] MAC II is a registered trademark of APPLE.

[3] MATLAB is a registered trademark of The MathWorks, Inc. [4]

1. BB2NDST.M

```
% BB2NDST.M                                                    11 Mar. 1991
% BB2NDST.M is a simulation of the 2nd order bang-bang controller with an analytic
% solution for the Switching Time of the control effort.  The signs of the control effort
% must be matched with the initial conditions to have convergence.

% written by Colin R. Cooper

% Define the State Equations.
A = [0 1;0 0];
B = [0 1]';
C = [1 0];
x10 = 2;                                  % x1 initial condition.
x20 = 1;                                  % x2 initial condition.
N = 1;  % Maximum control effort.
aN = abs(N);
Tf = 4.17;                                % Length of simulation.
dt = .005;                                % Time increment for simulation.
[Phi,Del] = c2d(A,B,dt);                  % Discretized System.

% Create the storage vectors.
kmax = Tf/dt + 1;
x = zeros(2,kmax);
y = zeros(1,kmax);
time = zeros(1,kmax);
x(:,1) = [x10;x20];                       % Set initial conditions for x.

% Define the Switching Time for the control effort.
tsf = abs(x20)/aN + sqrt(abs(x10 + x20*abs(x20)/(2*aN))/aN);

% Begin simulation loop.
```

```
for i = 1:kmax - 1
    if time(i)<tsf
        u = -N;
    else
        u = N;
    end
    x(:,i+1) = Phi*x(:,i) + Del*u;
    y(1,i+1) = C*x(:,i+1);
    time(i+1) = time(i)+dt;
end

%  Plot the output of the simulation.
clg,plot(x(1,:),x(2,:),'-w'),grid
title('Phase plot - Switching Time')
xlabel('X1'),ylabel('X2')
```

2. BB2NDSL.M

```
% BB2NDSL.M                                                   11 Mar. 1991
% is a simulation of the 2nd order bang-bang controller using a Switching Law for the
% control effort.

% written by Colin R. Cooper

A = [0 1;0 0];%                         Define the State Equations.
B = [0 1]';
C = [1 0];
x10 = 2;                                % x1 initial condition.
x20 = 1;                                % x2 initial condition.
N = 1; % Maximum control effort.
Tf = 4.5;                               % Length of simulation.
dt = .01;                               % Time increment for simulation.
[Phi,Del] = c2d(A,B,dt);                % Discretize the System.

% Create storage vectors.
kmax = Tf/dt + 1;
x = zeros(2,kmax);
y = zeros(1,kmax);
u = zeros(1,kmax);
x(:,1) = [x10;x20];                     % Set initial conditions for x.

% Begin simulation loop.
for i = 1:kmax - 1
    u(i) = -N*sign(x(1,i) + .5*x(2,i)*abs(x(2,i))/N);
    x(:,i+1) = Phi*x(:,i) + Del*u(i);
    (1,i+1) = C*x(:,i+1);
end
```

```
%  Plot the output of the simulation.
clg,plot(x(1,:),x(2,:)),grid
title('Phase plot - Switching Law')
xlabel('X1'),ylabel('X2')
```

3. SIM2SL.M

```
%  SIM2SL.M            2nd Order Switching Laws Control               08 Apr. 1991

%  Simulation of the missile/target simulation.
%
%  -> Uses analytic values for sigma-dot,sigma-ddot for calcultion of the control effort
%       using the 2nd order switching curves.
%  -> The Target makes no evasive maneuvers.
%  -> Calculates the Quantization Error based on the average velocities for the crossover
%       endpoints.
%  -> Calls INTERP.M function which takes the states at the crossover endpoints, creates
%       100 point straight lines to connect the points, and determine the minimum miss
%       distance.
%  -> Allows delay time for missile control.
%  -> Target is now on level flight with Beta = 0.

%  written by Colin R. Cooper


%  Define states.
N = 1000;                                      % Maximum control effort.
Am = [0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];        % Missile State Equations.
Bm = [0 0;1 0;0 0;0 1];
At = [0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];        % Target State Equations.
Bt = [0 0;1 0;0 0; 0 1];
Tf = 2.25;                                     % Total time of simulation.
dt = .01;                                      % Sample step size.

%  Descretize the states.
[Phim,Delm] = c2d(Am,Bm,dt);                   % Discrete Missile System
[Phit,Delt] = c2d(At,Bt,dt);                   % Discrete Target System
```

```
% Define storage vectors.
kmax = Tf/dt + 1;
xm = zeros(4,kmax);                         % xm = [ y yd x xd ]'
xt = zeros(4,kmax);                         % xt = [ y yd x xd ]'
time = zeros(1,kmax);
sig = zeros(3,kmax);
am = zeros(1,kmax);
um = zeros(2,kmax);
Rtm = zeros(1,kmax);
xm(:,1) = [0;0;0;2000];                     % Initial Conditions for Missile.
xt(:,1) = [1000;0;10000;-3000];             % Initial Conditions for Target.
Rtm(1) = sqrt((xt(1,1)-xm(1,1))^2 + (xt(3,1)-xm(3,1))^2);   % First value for Range.
acm = 0.0;                                  % Initial acceleration for the missile.
act = 0.0;                                  % Initial acceleration for the Target.

% Begin the Simulation Loop.
for i = 1:kmax-1
%   Define angles.
    beta = atan2(xt(2,i), xm(4,i));         % Velocity angle for the Target.
    gam = atan2(xm(2,i), xm(4,i));          % Velocity angle for the Missile.
    sig(1,i) = atan2(xt(1,i)-xm(1,i), xt(3,i)-xm(3,i));    % LOS angle.
    sig(2,i) = ((xt(3,i)-xm(3,i))*(xt(2,i)-xm(2,i)) - (xt(1,i)-xm(1,i))...
        *(xt(4,i)-xm(4,i)))/Rtm(i)^2;
    sig(3,i) = ((xt(3,i)-xm(3,i))*(act*cos(beta)-acm*cos(gam))-...
        (xt(1,i)-xm(1,i))*(act*sin(beta)+acm*sin(gam))+...
        2*(xt(4,i)-xm(4,i))*(xt(2,i)-xm(2,i)))/Rtm(i)^2;
    sig(3,i) = sig(3,i)+2*(xt(4,i)-xm(4,i)+xt(2,i)-xm(2,i))*sig(2,i)/Rtm(i);

%   Acceleration = Max value perpendicular to gamma.
    am(i) = N*sign(sig(2,i) + sig(3,i)*abs(sig(3,i))/(2*N));
    um(:,i) = [am(i)*cos(gam); -am(i)*sin(gam)];
    xm(:,i+1) = Phim*xm(:,i) + Delm*um(:,i);
    xt(:,i+1) = Phit*xt(:,i) + Delt*[0;0];
    time(i+1) = time(i) + dt;
    Rtm(i+1) = sqrt((xt(1,i+1)-xm(1,i+1))^2 + (xt(3,i+1)-xm(3,i+1))^2);
    acm = am(i);
end

% Evaluation of Miss distance.
iflag = 0;
rm = find(Rtm==min(Rtm));                    % Index of the minimum range value.
if rm == kmax                                % Determine whether the crossover occurs
    iflag = 1;                               %    before or after the min range value.
    It = rm - 1;
elseif Rtm(rm+1)<Rtm(rm-1)
    It = rm;
else
    It = rm - 1;
end

% Average Velocities in Intercept Area.
Vm = .5*(sqrt(xm(2,It)^2+xm(4,It)^2) + sqrt(xm(2,It+1)^2+xm(4,It+1)^2));
Vt = .5*(sqrt(xt(2,It)^2+xt(4,It)^2) + sqrt(xt(2,It+1)^2+xt(4,It+1)^2));
QE = .5*dt*(Vm + Vt);                        % Quantization Error.
if iflag == 0
    r = interp(xm(:,It:It+1),xt(:,It:It+1));    % Interpolated minimum miss distance.
    rstr = ['Inter Miss = ' num2str(r) ' ft'];
else
    rstr = ['Crossover never reached'];
end

%  Plot the output of the simulation with results.
plot(xm(3,:),xm(1,:),'-w',xt(3,:),xt(1,:),'-w')
text(.15,.85,['Intercept Time = ' num2str(time(rm)) ' sec'],'sc');
text(.15,.81,['Min. Miss = ' num2str(Rtm(rm)) ' ft'],'sc');
text(.15,.77,['Quan. Err = ' num2str(QE) ' ft'],'sc');
text(.15,.73,rstr,'sc');
text(.15,.69,['N = ',num2str(N)],'sc');
title('Control: 2nd Order Switching Laws (sig, sigd)')
xlabel('Direction 1 (ft)'),ylabel('Direction 2 (ft)')
```

4. SIMPRNV.M

```
% SIMPRNV.M simulation of the missile/target simulation.
% Proportional Navigation Guidance.

% written by Colin R. Cooper


% Define states.
As = [0 1; -64 -16];                       % Estimator for sigma-dot.
Bs = [0; 64];
Am = [0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];    % Missile State Equations.
Bm = [0 0;1 0;0 0;0 1];
At = [0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];    % Target State Equations.
Bt = [0 0;1 0;0 0; 0 1];
Tf = 2.25;                                 % Simulation Time.
dt = .01;                                  % Sample step size.
maxac = 1000.0;

% Descretize the states.
[Phis,Dels] = c2d(As,Bs,dt);               % Estimator System.
[Phim,Delm] = c2d(Am,Bm,dt);               % Missile System.
[Phit,Delt] = c2d(At,Bt,dt);               % Target System.

% Create strage vectors.
kmax = Tf/dt + 1;
xm = zeros(4,kmax);                        % xm = [ y yd x xd ]'
xt = zeros(4,kmax);                        % xt = [ y yd x xd ]'
um=zeros(2,kmax);
time = zeros(1,kmax);
Rtm = zeros(1,kmax);
b = zeros(2,kmax);
```

```
xm(:,1) = [0;0;0;2000];                      % Missile Initial Conditions.
xt(:,1) = [1000;0;10000;-3000];              % Target Initial Conditions.
Rtm(1) = sqrt((xt(1,1)-xm(1,1))^2 + (xt(3,1)-xm(3,1))^2);  % First value for Range.

% Begin simulation loop.
for i = 1:kmax-1
    Vm = sqrt(xm(2,i)^2 + xm(4,i)^2);
    ub(i) = atan2(xt(1,i)-xm(1,i), xt(3,i)-xm(3,i));
    b(:,i+1) = Phis*b(:,i) + Dels*ub(i);
    gam = atan2(xm(2,i), xm(4,i));
    if Vm*b(2,i)<=maxac                      % Apply Thrust limitation.
        am = Vm*b(2,i);
    else
        am = maxac;
    end
    um(:,i) = [am*cos(gam); -am*sin(gam)];   % Acceleration is perpendicular to gamma.
    xm(:,i+1) = Phim*xm(:,i) + Delm*um(:,i);
    xt(:,i+1) = Phit*xt(:,i) + Delt*[0;0];
    time(i+1) = time(i) + dt;
    Rtm(i+1) = sqrt((xt(1,i+1)-xm(1,i+1))^2 + (xt(3,i+1)-xm(3,i+1))^2);
end
rm = find(Rtm==min(Rtm));                    % Index of minimum range value.
if Rtm(rm+1)<Rtm(rm-1)                       % Determin whether crossover occurs
    It = rm;                                 %   before or after the min range value.
else
    It = rm - 1;
end
r = interp(xm(:,It:It+1),xt(:,It:It+1));     % Obtain the Interpolated min miss distance
Vm = sqrt(xm(2,It)^2 + xm(4,It)^2);
Vt = sqrt(xt(2,It)^2 + xt(4,It)^2);
QE = .5*dt*sqrt(Vm^2 + Vt^2);                % Quantization Error.

% Plot the output of the simulation.
axis([0 10000 0 1400]);                      % Same scale as the Optimal Control Sim.
```

```
plot(xm(3,:),xm(1,:),'-w',xt(3,:),xt(1,:),'-w')
text(.15,.85,['Intercept Time = ' num2str(time(rm)) ' sec'],'sc')
text(.15,.81,['Miss distance = ' num2str(Rtm(rm)) ' ft'],'sc');
text(.15,.77,['Quant Error = ' num2str(QE) ' ft'],'sc');
text(.15,.73,['Inter Error = ' num2str(r) ' ft'],'sc');
text(.15,.69,['Sigma(It) = ' num2str(ub(It)*180/pi) ' deg'],'sc');
text(.15,.65,['Max Acc. Limit = ' num2str(maxac) ' ft/sec^2'],'sc');
title('Control: Proportional Navigation')
xlabel('Direction 1 (ft)'),ylabel('Direction 2 (ft)');
```

5. BB3RDORD.M

```
% BB3RDORD.M                                                    3/7/91
% This is a simulation of a 3rd order system, forward time.
% This version allows varied N values.

%  written by Colin R. Cooper



x10=-.5;                                % Define initial conditions
x20=0;
x30=0;
N=1;   % Set the magnitude of the control effort.
Tf=2.5;                                 % Set maximum time of simulation.
dt=.002;                                % Set the simulation step size.

A=[0 1 0;0 0 1;0 0 0];                  %  Define the State Equations
B=[0 0 1]';
C=[1 0 0];

[Phi,Del]=c2d(A,B,dt);                  % Discretize the system.
kmax=Tf/dt +1;                          % Max integer value for the simulation.

% Prepare storage vectors.
u=zeros(1,kmax);
x=zeros(3,kmax);
y=zeros(1,kmax);
w=zeros(1,kmax);
f=zeros(1,kmax);
time=zeros(1,kmax);
```

```
x(:,1)=[x10;x20;x30];                         % Define initial conditions in state vectors

% Begin loop for simulation.
for (i=1:kmax-1)
    w(i)=sign(x(2,i)+x(3,i)*abs(x(3,i))/(2*N));      % Defining the switching law.
    f(i)=x(2,i)+w(i)*(x(3,i)^2)/(2*N);
    u(1,i)=-N*sign(x(1,i) + (x(3,i)^3)/(3*N^2) + w(i)*x(3,i)*x(2,i)/N +...
        f(i)*abs(f(i)/N)^.5);
    x(:,i+1) = Phi*x(:,i) +Del*u(1,i);          % Calculate the state values.
    y(1,i+1) = C*x(:,i+1);
    time(i+1)= time(i) + dt;                    % Store time vector.
end;

% Plot the switching law and its components.
clg, axis([0 2.5 -1.5 1.5]);
plot(time,u,time,.75*w,time,f)                  % w is scaled to distinquish it from u.
title('Control Laws vs Time')
pause

% Plot the 3-Dimensional view of the simulation from 45 deg. azimuth
%   and 45 deg. elevation angle. (Pos. x1 vector is out of the screen
%   and towards the left, Pos. x2 is out of the screen and towards right.
clg, axis;
plot3d(x(1,:),x(2,:),x(3,:),45,45);
```

6. ADJOINT.M

```
% ADJOINT.M  is a simulation of a controlled system using formard time simulation
%       and reverse time or adjoint solution.

% written by Colin R. Cooper                              11 Mar. 1991

A = [0 1 0;0 0 1;0 0 0];                          % Define Forward Time State Equations
B = [0 0 1]';
C = [1 0 0];
x10 = 2;                                          % x1 initial condition.
x20 = 0;                                          % x2 initial condition.
N = 1; % Maximum control effort.
Tf = 4;                                           % Length of simulation.
dt = .005;                                        % Time increment for simulation.
tsf = x20/N + sqrt(x10/N + .5*(x20/N)^2);         % Forward Time Switching Time
tsa = sqrt(x10/N + .5*(x20/N)^2);                 % Adjoint System Switching Time
kmax = Tf/dt + 1;                                 % Max length of storage vectors.

% Define the Storage Vectors
x = zeros(3,kmax);                                % States of the system
y = zeros(1,kmax);                                % Output state vector
time = zeros(1,kmax);
imp = zeros(1,kmax);                              % Impulse vector for control times.
imp(1) = -N/dt;                                   % First pulse at t = 0
imp(round(tsf/dt)+1) = 2*N/dt;                    % Second pulse at t = tsf
x(1:2,1) = [x10;x20];                             % Set initial conditions for x.
[Phi,del]=c2d(A,B,dt);                            % Discretize the state equations

for i = 1:kmax - 1                                % Begin the simulation loop
    x(:,i+1) = Phi*x(:,i) + del*imp(i);
    y(1,i+1) = C*x(:,i+1);
    time(i+1) = time(i)+dt;
end
miss = min(sqrt(x(1,:).^2 + x(2,:).^2));            % Find the min. miss distance
tff = time(find(sqrt(x(1,:).^2 + x(2,:).^2)== ...       % Find the time of the min. miss
        min(sqrt(x(1,:).^2 + x(2,:).^2))));
clg,plot(x(1,:),x(2,:),'-w'),grid                   % Plot the Phase Plane
title('Forward Time Phase plot')                    % Lable the graph and display the desired
xlabel('X1'),ylabel('X2')                           %    information
text(.6,.80,['min miss = ' num2str(miss)],'sc')
text(.6,.77,['tsf = ' num2str(tsf) ' sec.'],'sc')
text(.6,.74,['tff = ' num2str(tff) ' sec.'],'sc')
pause

% Now for the Adjoint System.
xa = zeros(3,kmax);                                 % Define storage vectors
time = zeros(1,kmax);
impa = zeros(1,kmax);
impa(1) = N/dt;                                     % First impulse
impa(round(tsa/dt)+1) = -2*N/dt;                    % Second impulse
[Phi,del]=c2d(A',C',dt);                            % Discretize the Adjoint System

for i = 1:kmax - 1                                  % Begin the simulation loop
    xa(:,i+1) = Phi*xa(:,i) + del*impa(i);
    ya(1,i+1) = B'*xa(:,i+1);
    time(i+1) = time(i) + dt;
end;
missa = min(sqrt((xa(3,:)-x10).^2 + (xa(2,:)-x20).^2));     % Find the min. miss distance
                                                            %    from the initial conditions
tfa = time(find(sqrt((xa(3,:)-x10).^2 + (xa(2,:)-x20).^2)== ...     % Find the time of the
        min(sqrt((xa(3,:)-x10).^2 + (xa(2,:)-x20).^2))));           %    min miss distance

plot(xa(3,:),-xa(2,:),'-w'),grid                    % Plot the Phase Plane
title('Adjoint Solution Phase plot')                % Label the graph and display the desired
xlabel('X3'),ylabel('X2')                           %    information.
```

```
text(.6,.8,['min miss = ' num2str(missa)],'sc')
text(.6,.77,['tsa = ' num2str(tsa) ' sec.'],'sc')
text(.6,.74,['tfa = ' num2str(tfa) ' sec.'],'sc')
```

7. ADJSIM.M

```
% ADJSIM.M  Adjoint solution to missile intercept problem.

% written by Colin R. Cooper                    11 Mar. 1991


n = 4;                                  % Proportionality Constant
v = 5000;                               % Closing Velocity
B = [0 0 0 0 0 1]';                     % Define State Matrices
C = [-1 0 0 0 1 0];
Tf = 1;                                 % Set first value for incremented times
dt = .01;

% This outer loop runs the complete forward time system for each value of Tf, storing
%       the output into vectors at the end of the loop.
for j = 1:4
    kmax = Tf/dt + 1;                   % Maximum index for storage vectors
    imp = zeros(1,kmax);                % Create vector for impulsive input
    imp(1) = 1/dt;                      % Define the pulse at t = 0
    x = zeros(6,kmax);                  % Create storage vectors for states
    y = zeros(1,kmax);
    time = zeros(1,kmax);

% Forward Time Simulation.
    count = 0;                          % Counter to indicate the computer is busy
    for i = 1:kmax - 1                  % Begin simulation loop
        count = count + 1;
        ki = 1/(v*(Tf - time(i) + 1e-12));   % 1/Range
        A = [0 1 0 0 0 0                % Define the time varying A matrix
        0 0 n*v 0 0 0
        0 0 0 1 0 0
        -100*ki 0 -100 -20 100*ki 0
        0 0 0 0 0 1
        0 0 0 0 0 0];
        Phi = eye(6) + A*dt + .5*A^2*dt^2;   % Discretize the matrix with 2nd order
                                             %    expansion - drop higher terms
        x(:,i+1) = Phi*x(:,i) + B*dt*imp(i);
        y(:,i+1) = C*x(:,i+1);
        time(i+1) = time(i) + dt;
        if count == 50                       % Counter indicates computer is busy
            disp('working');
            count = 0;
        end
    end                                      % End simulation loop

    Tend(j) = Tf;                            % Record final time
    yend(j) = y(length(y));                  % Record corresponding final value ymiss
    eval(['y',num2str(j),' = y;']);          % Save output vector
    eval(['t',num2str(j),' = time;']);1      % Save corresponding time vector
    Tf = Tf + 1                              % Increment to next Tf
end                                          % End outer forward time loop

% Adjoint Simulation.
Tf = 4.25;                                   % Set a simulation time
kmax = Tf/dt + 1;                            % Maximum index for vectors
imp = zeros(1,kmax);                         % Create vector for impulsive input
imp(1) = 1/dt;                               % Define the impulse at t = 0
xa = zeros(6,kmax);                          % Create storage vectors
ya = zeros(1,kmax);
time = zeros(1,kmax);
count = 0;
for i = 1:kmax - 1                           % Begin simulation loop
    count = count + 1;
    ki = 1/(v*(time(i) + 1e-12));            % 1/Range
    A = [0 1 0 0 0 0                         % Define the time varying A matrix
    0 0 n*v 0 0 0
    0 0 0 1 0 0
    -100*ki 0 -100 -20 100*ki 0
    0 0 0 0 0 1
    0 0 0 0 0 0];
    Phi = eye(6) + A'*dt + .5*A'^2*dt^2;          % Discretize the matrix
    xa(:,i+1) = Phi*xa(:,i) + C'*dt*imp(i);
    ya(:,i+1) = B'*xa(:,i+1);
    time(i+1) = time(i) + dt;
    if count == 50
        disp('working');
        count = 0;
    end
end                                               % End adjoint simulation loop

plot(t1,y1,'-w',t2,y2,'-w',t3,y3,'-w',t4,y4,'-w',time,ya,'-w')      % plot output
xlabel('Time (sec)'),ylabel('Ymiss (ft)'),grid
```

8. INTERP.M

```
function r = interp(xm,xt)
% INTERP.M will return an interpolated value for the min.miss distance given the states
%       for the two intercept values.

% Written by Colin R. Cooper                                    26 Mar 1991

% Increase the sample rate by a factor of 100 in crossover region.
dax = (xm(3,2)-xm(3,1))/100;
day = (xm(1,2)-xm(1,1))/100;
dbx = (xt(3,2)-xt(3,1))/100;
dby = (xt(1,2)-xt(1,1))/100;

% Define the storage vectors for 100 data points.
ax = zeros(1,100);
ay = zeros(1,100);
bx = zeros(1,100);
by = zeros(1,100);

% Set initial values for each vector.
ax(1) = xm(3,1);
ay(1) = xm(1,1);
bx(1) = xt(3,1);
by(1) = xt(1,1);

% Assuming a straight line trajectory with constant velocity in the crossover region, create
%       the interpolation data sets.
for i = 1:99
    ax(i+1) = ax(i) + dax;
    ay(i+1) = ay(i) + day;
    bx(i+1) = bx(i) + dbx;
    by(i+1) = by(i) + dby;
end

%  Find the closest point of approach of the two line segments.
r = min(sqrt((ax - bx).^2 + (ay - by).^2));
```

9. PLOT3D.M

```
% PLOT3D is a 3d plotting function allowing rotation and elevation adjustments. The plot
% shows a 3-D curve and its projection onto the X-Y Plane.
%       X,Y,and Z data must be passed, and if no azimuth or elevation values are passed
% they default to AZ = 45°, EL = 30°.  The Azimuth is the angle of rotation of the view
% angle about the Z-Axis.  AZ = 0° is looking straight down the X-Axis at the Y-Z Plane.
% The elevation is the angle from which the plot is viewed. EL = 90° is looking down the
% Z-Axis at the X-Y Plane. The elevation angle can vary from -90° to 90°.
%       The tick marks on the axis will default to 10 marks per axis and the values will be
% displayed on the screen.  To define the values of the tick marks a three element vector
% must be passed containing [dx dy dz];
%       Axis values will be calculated and fixed unless an axis vector is passed to the
% program: [xmin xmax ymin ymax].
%       The Transformed 2-D vectors V and Vs are returned, where V is the 3-D curve and
% Vs is the Projection on the X-Y Plane.
%
% Example :                    [V,Vs] = plot3d(x,y,z,-45,30,dx,Ax)

% written by Colin Cooper                                                   4/26/91

function [V,Vs]=plot3d(x,y,z,az,el,dx,Ax)

if nargin < 5, el = 30; end
if nargin < 4, az = 45; end

az=-az*pi/180; el=el*pi/180;
alpha = 45*pi/180; beta=30*pi/180;            % Angles for mapping onto 2D
                                              % alpha = rot. Beta = elv.
S1=[sin(az) cos(az) 0
    -sin(el)*cos(az) sin(el)*sin(az) cos(el)];
S2=[sin(az) cos(az) 0
    -sin(el)*cos(az) sin(el)*sin(az) 0]; % Trans for Projection.
```

```
[r,c]=size(x);
if r > 1
    x = x'; y = y'; z = z';
end

V = S1*[x;y;z];
Vs = S2*[x;y;z];

if nargin < 6
    dx(1) = (max(x)-min(x))/10;
    dx(2) = (max(y)-min(y))/10;
    dx(3) = (max(z)-min(z))/10;
end

ax1 = [min(x) max(x); 0 0; 0 0];
ax2 = [0 0; min(y) max(y); 0 0];
ax3 = [0 0; 0 0; min(z) max(z)];

ax1t = [fliplr(0:-dx(1):min(x)) 0:dx(1):max(x)
    zeros([0:-dx(1):min(x) 0:dx(1):max(x)])
    zeros([0:-dx(1):min(x) 0:dx(1):max(x)])];
ax2t = [zeros([0:-dx(2):min(y) 0:dx(2):max(y)])
    fliplr(0:-dx(2):min(y)) 0:dx(2):max(y)
    zeros([0:-dx(2):min(y) 0:dx(2):max(y)])];
ax3t = [zeros([0:-dx(3):min(z) 0:dx(3):max(z)])
    zeros([0:-dx(3):min(z) 0:dx(3):max(z)])
    fliplr(0:-dx(3):min(z)) 0:dx(3):max(z)];

ax1 = S1*ax1;
ax2 = S1*ax2;
ax3 = S1*ax3;
ax1t = S1*ax1t;
ax2t = S1*ax2t;
```

```
ax3t = S1*ax3t;

minx = min([V(1,:) Vs(1,:) ax1(1,:) ax2(1,:) ax3(1,:)]);
if minx == 0, minx = -1; end
maxx = max([V(1,:) Vs(1,:) ax1(1,:) ax2(1,:) ax3(1,:)]);
if maxx == 0, maxx = 1; end
miny = min([V(2,:) Vs(2,:) ax1(2,:) ax2(2,:) ax3(2,:)]);
if miny == 0, miny = -1; end
maxy = max([V(2,:) Vs(2,:) ax1(2,:) ax2(2,:) ax3(2,:)]);
if maxy == 0, maxy = 1; end

clg
axis('square');
if nargin < 7,
  axis([minx-.3*abs(minx) maxx+.3*abs(maxx) miny-.3*abs(miny) ...
      maxy+.3*abs(maxy)]);
else
  axis(Ax);
end
b = axis;
hold on
plot(ax1(1,:),ax1(2,:),'-w',ax1t(1,:),ax1t(2,:),'+w',...
     ax2(1,:),ax2(2,:),'-w',ax2t(1,:),ax2t(2,:),'+w',...
     ax3(1,:),ax3(2,:),'-w',ax3t(1,:),ax3t(2,:),'+w')
lv = length(V);
plot(V(1,:),V(2,:),'-w',Vs(1,:),Vs(2,:),'-w',...
     [V(1,1:15:lv);Vs(1,1:15:lv)],[V(2,1:15:lv);Vs(2,1:15:lv)],':w')
plot([b(1) b(2) b(2) b(1) b(1)],[b(3) b(3) b(4) b(4) b(3)],'-w');
hold off
text(.22,.08,['AZ = ' num2str(-az*180/pi) '°'],'sc');
text(.72,.08,['EL = ' num2str(el*180/pi) '°'],'sc');
text(.22,.88,['dx = ' num2str(dx(1)) ],'sc');
text(.50,.88,['dy = ' num2str(dx(2)) ],'sc');
text(.72,.88,['dz = ' num2str(dx(3)) ],'sc');
```

REFERENCES

[1] Kirk, D.E., *Optimal Control Theory, an Introduction,* ch. 5.4, pp. 240-259, Prentice-Hall Inc, 1970, .

[2] Ford Aerospace & Communications Corporation, Newport Beach, California, Technical Data Release No. RADAMS-005, *Miss Distance Computation Using the Method of Adjoint Differential Equations*, J.B. Burl, May 1983.

[3] Zarchan, P., *Tactical and Strategic Missile Guidance, (Progress in Astronautics and Aeronautics;v.124)*, American Institute of Aeronautics and Astronuatics, 1990.

[4] *PC-MATLAB, version 3.5f*, and *Mac II MATLAB, version 1.1b*, The MathWorks, Inc., Feb. 3, 1989

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Chairman, Code EC Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Hal A. Titus, Code EC/Ts Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	6
5.	Professor Jeffrey B. Burl, Code EC/Bl Department of Electrical & Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Professor A. W. Cooper, Code PH/Cr Department of Physics Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Mr. Colin R. Cooper 402 Grant Ave. Monterey, CA 93940	2

Thesis
C7524 Cooper
c.1 Second and third order minimum time controllers and missile adjoints.