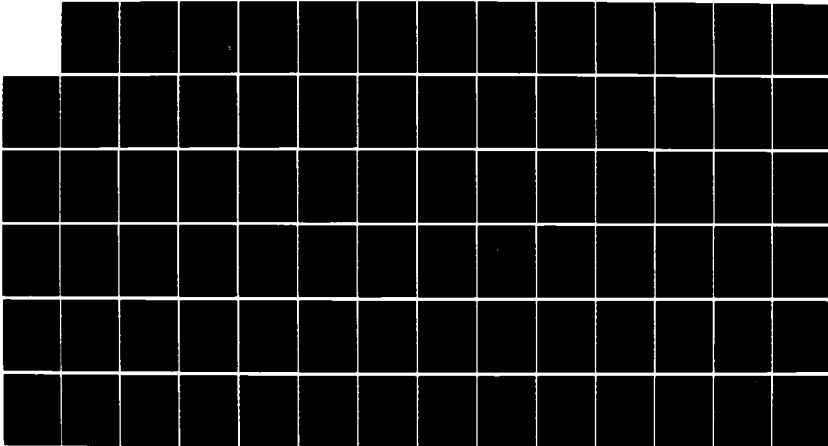AD-A164 019   STABILITY OF SPINNING ICBM (INTERCONTINENTAL BALLISTIC   1/ℓ
                 MISSILE) IN FIRST..(U) AIR FORCE INST OF TECH
                 WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..   R W BANDSTRA

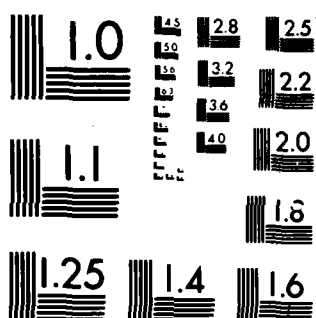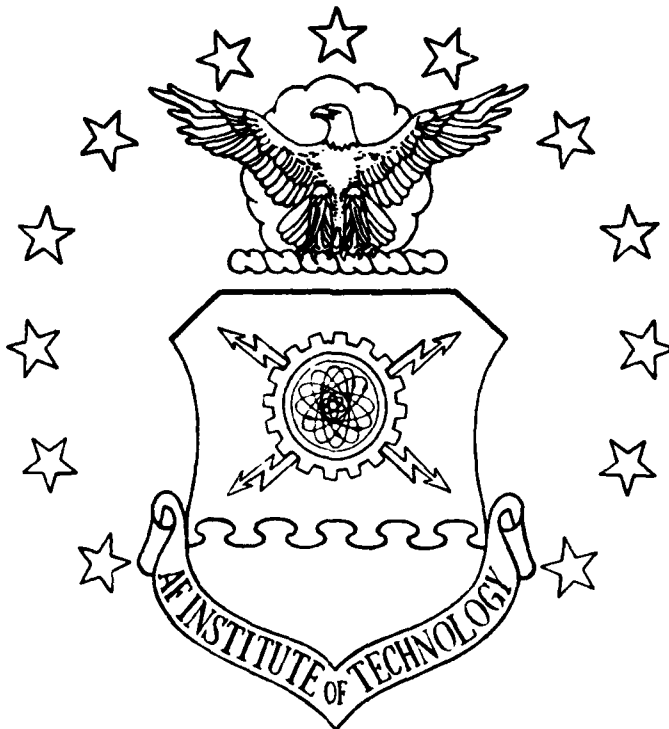UNCLASSIFIED   DEC 85 AFIT/GA/AA/85D-01   F/G 16/4.2   NL

END

MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

STABILITY OF SPINNING ICBM IN

FIRST STAGE BOOST PHASE

THESIS

Robert W. Bandstra
Captain, USAF

AFIT/GA/AA/85D-01

DTIC
ELECTE
FEB 1 2 1986
S D
B

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

86 2 12 061

AFIT/GA/AA/85D-01

STABILITY OF SPINNING ICBM IN

FIRST STAGE BOOST PHASE

THESIS

Robert W. Bandstra
Captain, USAF

AFIT/GA/AA/85D-01

DTIC
SELECTED
FEB 1 2 1986

B

AFIT/GA/AA/85D-01

STABILITY OF SPINNING ICBM IN FIRST STAGE BOOST PHASE

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

Robert W. Bandstra, B.S.

Captain, USAF

December 1985

Acknowledgements

I wish to thank Dr. William Wiesel for his knowledge and encouragement during this research effort. Dr. Wiesel's immense knowledge of astrodynamics and vehicle dynamics made completion of the thesis possible. His invaluable contributions as advisor and instructor cannot be overstated.

I wish to thank each member of the GA-85D section for their support and friendship which has made the past 18 months a memorable experience. I wish to specifically thank 2Lt Bill Berrier for his time, technical assistance, and friendship during the thesis effort.

I wish to thank my dear wife, Laura, for her loving encouragement and understanding throughout our AFIT tour.

Finally, all credit and praise belong to my faithful Lord and Saviour, Jesus Christ, whose guiding hand led me through this effort.

<div align="right">Robert W. Bandstra</div>

Table of Contents

List of Figures

List of Tables

AFIT/GA/AA/85D-01

Abstract

A computer program is developed to model a spinning
Intercontinental Ballistic Missile (ICBM) during the first
stage boost phase. The equations of motion are derived and
presented and a full rotation matrix is used to show the
relationship between a launch-centered, nonrotating earth,
inertial reference frame and the missile body reference frame.
The moments of inertia and aerodynamic forces are derived and
presented. A feedback controller is derived which proved to
be a necessary addition to the system in order to reduce the
angle of attack. The angle of attack of the missile produced
adverse effects on the burnout vector without the feedback
controller but the effects are reduced considerably with the
controller included. Problem areas include possibly excessive
nozzle gimbal rates caused by the feedback controller and the
need to change the initial kick angle if the missile is
spinning in order to achieve the same burnout conditions as a
nonspinning missile.

STABILITY OF SPINNING ICBM IN FIRST STAGE BOOST PHASE

Chapter One Introduction

Statement of the Problem

In the near future, Intercontinental Ballistic Missiles (ICBM) may be vulnerable to airborne and space-based lasers. The most vulnerable stage of the ICBM's flight is during the Stage 1 burn when it is moving slowly away from a well known position at the launch site. If the ICBM is to survive this stage of flight defensive measures will have to be taken. One of the ways to accomplish this would be to spin the missile about its symmetry axis to reduce the dwell time of a beam on a particular point on the missile. But, when this spin is introduced, other problems may arise. The missile's flight will describe a coning motion about its symmetry axis that may cause an excessive angle of attack. If the angle of attack remains small it is possible that the position and velocity errors at the end of the boost phase may be unacceptably large. That is the problem that this research effort will address.

Introduction

Since the first days of computers, modelling of missiles has been attempted at varying levels of complexity. A review of the technical literature available shows that everything from spinning mortar shells to sophisticated ICBMs have been

1

modelled. Even though other specific cases have been modelled
no one seems to have analyzed the case of a spinning ICBM.
The remainder of this chapter presents a summary of related
efforts.

Synopsis of Related Efforts

In 1976 the Aerophysics Laboratory conducted research on
statically stable missiles that were given a nominal roll rate
to average out lifting effects of configuration assymmetries
(1). They showed that even with a steady roll, lift
variations can cause dispersion and lift nonaveraging. This
study was done for artillery rounds spinning at about 40
rad/sec after thrust termination. Another study of small,
spinning missiles presents the effects of variable spin rate
and thrust misalignment on the natural frequencies and mode
shapes of the trajectory (2). The effects were significant
for small missiles but there is some question as to how this
applies to ICBMs.

The second category of related work presents trajectory
simulation programs for space vehicle launchers and ICBMs.
They all develop the equations used and present specific uses
for their programs but never consider the effects of steady
spin on the trajectory (3,4,5). These research works, though,
were good sources for programming techniques and missile
property computations.

The most beneficial reference materials are the textbooks

2

on rocket propulsion and spaceflight dynamics. The material covered is better explained than in research papers and some textbooks cover the case of spinning missiles. There are several good reference texts available (6,7), but the most complete presentation is given by Cornelisse and others in their book entitled "Rocket Propulsion and Spaceflight Dynamics" (8). This book is the main source of reference for this thesis.

Chapter Two   Derivation of Equations of Motion

Introduction to the Derivations

In order to describe the motion of a missile analyti-
cally, the equations of motion must be derived as a function
of time and numerically integrated. The equations of motion
can be set up as a set of ordinary differential equations and
integrated using a predictor-corrector algorithm to arrive at
a solution at each time step. The predictor-corrector algo-
rithm used in this research is of the fourth order, meaning
that it carries along the last four values of the state vector
and extrapolates these values to obtain the next value. It
then corrects the extrapolated value to find a new value for
the state vector. It was first used by Haming and bears his
name (9).

Once the integration method has been chosen the equations
of motion must be set up so they are compatible with the in-
tegrator. One of the problems to be solved is choice of
reference frames. Some of the forces and moments are more
easily expressed in a body reference frame and others in an
inertial reference frame. The solution to this problem is to
introduce a rotation matrix which will relate the reference
frames chosen. The two reference frames are shown in Figure
2.1. The inertial reference frame, $\underset{\sim}{E} = (X,Y,Z)$, has the
launch point as its origin and the vehicle reference frame,
$\underset{\sim}{E}_r = (x,y,z)$, has the vehicle center of mass, (COM), as its

origin and is fixed to the vehicle. The vehicle reference
frame will, in general, be rotating and translating with
respect to the inertial frame.

As stated before, Cornelisse's book, "Rocket Propulsion
and Spaceflight Dynamics" (8), will be referenced heavily in
this report. Any departures or additions to his derivations
will be noted but otherwise only the highlights of his
derivations will be presented here.
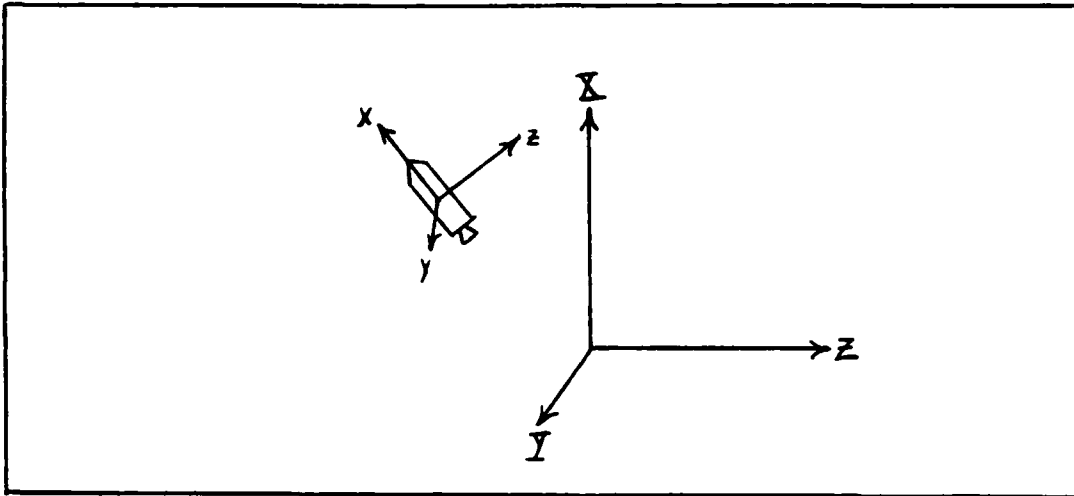
Figure 2.1 Reference Frames

The Dynamical Equations of Motion

The dynamical equations of motion describe how the forces
and their associated moments effect the velocity and rotation
of the center of mass of the vehicle. The force equation is
given by (8:78):

$$M(d\underline{V}_{cm}/dt) = \underline{T} + \underline{W} + \underline{F}_a \qquad (1)$$

5

where

M = instantaneous missile mass

$\underset{\sim}{V}_{cm}$ = velocity of the COM of the missile

$\underset{\sim}{T}$ = thrust of stage 1

$\underset{\sim}{W}$ = gravitational force

$\underset{\sim}{F}_a$ = aerodynamic forces

The moment equation is (8:78):

$$d(\underset{\sim}{I}\cdot\underset{\sim}{\Omega})/dt = -\dot{m}\underset{\sim}{r}_e \times (\underset{\sim}{\Omega} \times \underset{\sim}{r}_e) + \underset{\sim}{r}_e \times \underset{\sim}{F} + \underset{\sim}{M}_a \qquad (2)$$

where

$\underset{\sim}{I}$ = moment of inertia about the principal axes

$\underset{\sim}{\Omega}$ = angular velocity of vehicle about the body axes

$\dot{m}$ = mass flow rate

$\underset{\sim}{r}_e$ = position vector from the COM to the center of mass flow

$\underset{\sim}{M}_a$ = aerodynamic moments

Eqs (1) and (2) are vector equations of motion that can be separated into six scalar equations. The vectors occurring in Eqs (1) and (2) are resolved like this (8:79):

$\underset{\sim}{V}_{cm} = (u,v,w)\underset{\sim}{E}$ = velocity of the COM

$\underset{\sim}{\Omega} = (p,q,r)\underset{\sim}{E}_r$ = rotational velocity about the COM

$\underset{\sim}{T} = (T_x,T_y,T_z)\underset{\sim}{E}$ = thrust of stage 1

$\underset{\sim}{F}_a = (X_a,Y_a,Z_a)\underset{\sim}{E}$ = aerodynamic forces

$\underset{\sim}{M}_a = (L, M, N)\underset{\sim}{E}_r$ = aerodynamic moments

$\underset{\sim}{r}_e = (x_e,y_e,z_e)\underset{\sim}{E}_r$ = position vector from the COM to the center of mass flow

$\underset{\sim}{g} = (g_x,g_y,g_z)\underset{\sim}{E}$ = gravitational acceleration

6

$$\underset{\sim}{I} = (I_{xx}, I_{yy}, I_{zz})\underset{\sim}{E}_r = \text{the moments of inertia}$$

It should be noted that second-order terms involving y_e, z_e, T_y, and T_z are neglected because they are small compared to x_e and T_x, respectively. Substituting into Eqs (1) and (2) yields (8:79):

$$Mdu/dt = T_x + Mg_x + X_a \tag{3}$$

$$Mdv/dt = T_y + Mg_y + Y_a \tag{4}$$

$$Mdw/dt = T_z + Mg_z + Z_a \tag{5}$$

$$I_{xx}dp/dt = -pdI_{xx}/dt + rq(I_{yy}-I_{zz}) + \dot{m}x_e(y_eq+z_er)$$
$$+ L - m\rho l^2 p \tag{6}$$

$$I_{yy}dq/dt = -qdI_{yy}/dt + pr(I_{zz}-I_{xx}) - \dot{m}qx_e^2 - x_eF_z$$
$$+ z_eF_x + M \tag{7}$$

$$I_{zz}dr/dt = -rdI_{zz}/dt + pq(I_{xx}-I_{yy}) - \dot{m}rx_e^2 + x_eF_y$$
$$- y_eF_x + N \tag{8}$$

where ρl = offset distance of the COM flow from the missile's centerline

Eqs (3), (4), and (5) differ from Cornelisse's equations because he presents all of the vectors in the body frame and thus includes an additional term necessary to decompose the $\underset{\sim}{V}_{cm}$ into the body frame. That was avoided in this treatment by leaving all of the terms in Eqs (3), (4), and (5) in the inertial reference frame.

The last term in Eq (6) is not in Cornelisse's equation but it is included here because otherwise the spin of the missile is not properly modelled (7:225). As the propellant

7

is burned and exhausted, it takes with it a small amount of angular momentum. If this term is not included in Eq (6) the computer simulation would show that the missile spins faster and faster as the propellant is burned when this isn't necessarily true.

The Kinematical Equations of Motion

In the dynamical equations, the forces and the moments are dependent on the position and orientation of the missile. The kinematical equations are needed to relate the position and orientation of the missile to the translational and rotational velocity. The first three kinematical equations are represented by the vector equation (8:81):

$$d\underset{\sim}{R}_{cm}/dt = \underset{\sim}{V}_{cm} \qquad (9)$$

where $\underset{\sim}{R}_{cm} = (X,Y,Z)\underset{\sim}{E}$ = the position of the COM

Or, in component form:

$$dX/dt = u \qquad (10)$$

$$dY/dt = v \qquad (11)$$

$$dZ/dt = w \qquad (12)$$

Once again, by leaving the $\underset{\sim}{V}_{cm}$ vector in the inertial frame, a coordinate transformation is avoided that is included in Cornelisse's text.

The last set of equations necessary is the transformation matrix that converts vectors from the inertial frame to the

8

body frame. They are, in vector form (8:81):

$$d\underset{\sim}{A}_r/dt = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} \underset{\sim}{A}_r \qquad (13)$$

where $\underset{\sim}{A}_r$ is a 3X3 matrix of the direction cosines between the
body reference frame and the inertial reference frame. Eq
(13) can be expanded into the nine equations that it
represents:

$$dA_{r11}/dt = r\cos(y,X) - q\cos(z,X) \qquad (14)$$

$$dA_{r12}/dt = r\cos(y,Y) - q\cos(z,Y) \qquad (15)$$

$$dA_{r13}/dt = r\cos(y,Z) - q\cos(z,Z) \qquad (16)$$

$$dA_{r21}/dt = -r\cos(x,X) + p\cos(z,X) \qquad (17)$$

$$dA_{r22}/dt = -r\cos(x,Y) + p\cos(z,Y) \qquad (18)$$

$$dA_{r23}/dt = -r\cos(x,Z) + p\cos(z,Z) \qquad (19)$$

$$dA_{r31}/dt = q\cos(x,X) - p\cos(y,X) \qquad (20)$$

$$dA_{r32}/dt = q\cos(x,Y) - p\cos(y,Y) \qquad (21)$$

$$dA_{r33}/dt = q\cos(x,Z) - p\cos(y,Z) \qquad (22)$$

where $\cos(i,j)$ indicates the cosine of the angle between i and
j which are also elements of $\underset{\sim}{A}_r$

$\underset{\sim}{r} = (x,y,z)\underset{\sim}{E}_r$ = the axes of the body frame

$\underset{\sim}{R}_{cm} = (X,Y,Z)\underset{\sim}{E}$ = the axes of the inertial frame

The use of a full rotation matrix avoids the singularities
associated with any particular set of orientation angles. The
set of 18 equations of motion that fully describe the behavior
of the missile are Eqs (3)-(8), (10)-(12), and (14)-(22).

9

Chapter Three   Derivation of Missile Parameters and
Aerodynamic Forces

Introduction to Missile Data

All of the forces and moments in Eqs (3) - (8) vary with time, position, or orientation of the missile. In order to find relationships for each of these terms several assumptions were made. Some of the higher order terms were assumed to be small and thus ignored leaving linear relationships. The missile data used in all analyses in this report resemble that of a Minuteman III (MM III) ICBM but this is only for convenience since the data was available (10). Other approximations were made that will be presented throughout this chapter.

To avoid the added complication of launching from a vertical position and at some later time initiating a pitchover maneuver for a gravity turn, a cold launch system was assumed. This is a departure from current hardware configurations since MM III missiles can't be cold launched. The assumptions associated with the cold launch are that the stage 1 engine starts at 50 feet above mean earth radius with the center of mass moving at 50 ft/sec vertically. It is also assumed the desired initial angular velocities have been achieved when the thrust begins. To simulate a gravity turn trajectory an initial kick angle was input by misaligning the body axes with the inertial axes at the beginning of the simulation.

Gravity and Thrust Terms

The most important external forces acting on the missile are the gravitational forces and the thrust. The gravitational forces are dependent on the distance from the center of the Earth and can be represented by (8:75):

$$\underset{\sim}{g} = -GM \underset{\sim}{R}_{cm}/R_{cm}^3 \qquad (23)$$

where

$\underset{\sim}{g}$ = $(g_x, g_y, g_z)\underset{\sim}{E}$ = gravitational acceleration vector

GM = $1.407646882E+16$ ft^3/sec^2 = the gravitational parameter of the Earth

R_{cm} = the magnitude of the position vector from the Earth's center to the COM of the missile

$\underset{\sim}{R}_{cm}$ = $(X, Y, Z)\underset{\sim}{E}$ = position vector from the Earth's center to the COM of the missile

This vector can be resolved into its three components, g_x, g_y, and g_z, and multiplied by the instantaneous missile mass, M, to produce the respective terms in Eqs (3) - (5).

The thrust is assumed to be a constant throughout the duration of the first stage burn. This is never achievable in actual engine firings but the fraction of a second that it takes for the thrust curve to level off is not worth including in this study. In order to use the thrust in Eqs (3) - (5) though, it must be transformed from the body frame to the inertial frame. This is done by multiplying the transpose of $\underset{\sim}{A}_r$ by the body frame thrust, $\underset{\sim}{F}$:

$$\underset{\sim}{T} = \left[\underset{\sim}{A}_r\right]^T \underset{\sim}{F} \qquad (24)$$

11

where

$$\underset{\sim}{T} = (T_x, T_y, T_z) \underset{\sim}{E} = \text{thrust in the inertial frame}$$

$$\underset{\sim}{A}_r = \text{transformation matrix from inertial to body frame}$$

$$\underset{\sim}{F} = (F_x, F_y, F_z) \underset{\sim}{E}_r = \text{thrust in the body frame}$$

$\underset{\sim}{T}$ is then resolved into its three components for use in Eqs (3) - (5).

Moments of Inertia and Their Derivatives

The moments of inertia and their time derivatives are very important to this study because of their effects on rotational velocity. The best source of accurate data for this would be experimental results where the moments have been determined but this data is not readily available in the unclassified literature. Also, in order to use this computer program to simulate other missiles, it would be better if the program computed an estimate of the moments of inertia from a few standard inputs since the actual moments are hard to find. In order to calculate the moments and keep the calculations generic, several assumptions were made. The first assumption is that each stage is a solid, homogeneous cylinder so that the mass distribution is known. This, of course, is not the case in a real missile but the results are probably close enough for this study. The next assumption is that the fuel in the first stage motor burns uniformly from the inside out to within one inch of the outer radius. In the time derivatives of the moments of inertia, the burn rate is then

12

considered to be a constant given as:

$$db/dt = \dot{b} = b_f/burnt \qquad (25)$$

where

   b_f = radius of stage 1 minus one inch (for outer shell thickness)

   burnt = burn time of stage 1

The final assumption is that the missile is an axisymmetric, rigid body with the y and z body axes' moments of inertia the same. Their time derivatives are also considered to be the same.

There are several preliminary calculations that are necessary for the final moment of inertia equations to be easier to understand. The distances are defined in Figure 3.1.

The equations are:

$$m_1 = m_0 - \dot{m}t \qquad (26)$$

where

   m_1 = instantaneous mass of stage 1

   m_0 = initial mass of stage 1

   $\dot{m}$ = mass flow rate form stage 1

   t = elapsed time in seconds

$$m_{top} = m_2 + m_3 + m_4 \qquad (27)$$

where

   m_{top} = total mass of stages 2, 3, and 4

13

m_2 = mass of stage 2

m_3 = mass of stage 3

m_4 = mass of stage 4
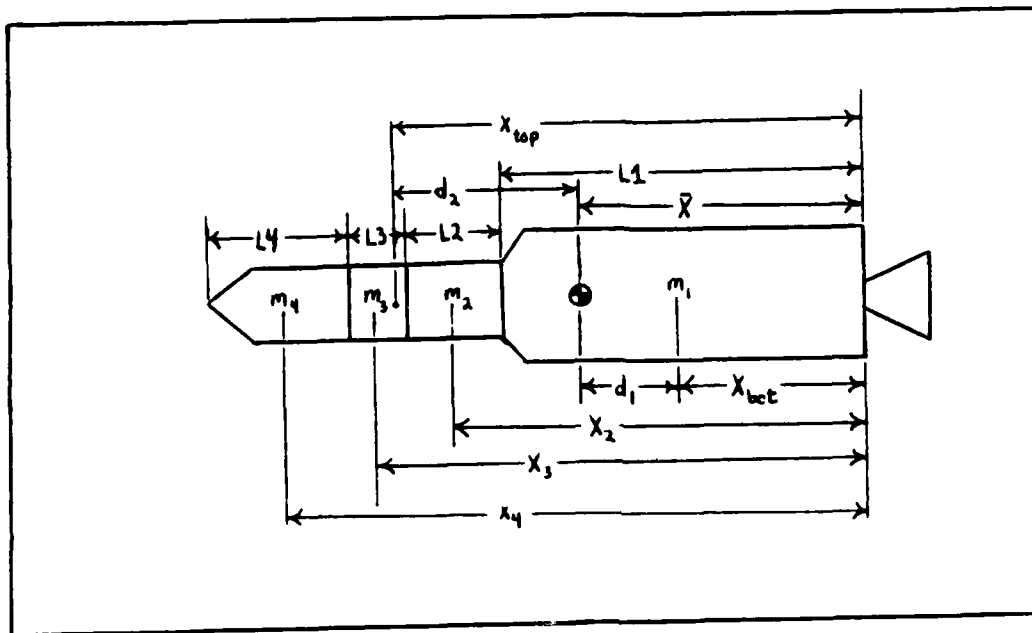
Figure 3.1   Missile Dimensions

$$x_{bot} = L1/2 \tag{28}$$

$$x_2 = L1 + L2/2 \tag{29}$$

$$x_3 = L1 + L2 + L3/2 \tag{30}$$

$$x_4 = L1 + L2 + L3 + L4/2 \tag{31}$$

where

x_{bot} = location of COM of stage 1 from bottom of missile

x_2 = location of COM of stage 2 from bottom of missile

x_3 = location of COM of stage 3 from bottom of missile

x_4 = location of COM of stage 4 from bottom of missile

14

L1 = length of stage 1

L2 = length of stage 2

L3 = length of stage 3

L4 = length of stage 4

$$x_{top} = (x_2 m_2 + x_3 m_3 + x_4 m_4)/m_{top} \tag{32}$$

$$\bar{x} = (x_{bot} m_1 + x_{top} m_{top})/(m_1 + m_{top}) \tag{33}$$

$$d_1 = \bar{x} - x_{bot} \tag{34}$$

$$d_2 = x_{top} - \bar{x} \tag{35}$$

where

x_{top} = location of COM of the top 3 stages from bottom of missile

$\bar{x}$ = distance between COM and bottom of missile

d_1 = moment arm distance for stage 1

d_2 = moment arm distance for top 3 stages

$$M = tmass - \dot{m}t \tag{36}$$

where

tmass = total mass of the missile at launch

M = instantaneous missile mass

The standard equations for the moments of inertia of a solid, homogeneous cylinder are:

$$I_{xx} = mass*radius^2/2 \tag{37}$$

$$I_{yy} = I_{zz} = mass(3*radius^2 + length^2)/12 \tag{38}$$

The moments of inertia for a four-stage missile are:

15

$$I_{xx} = (m_1(r_1^2 - \dot{b}^2 t^2) + m_2 r_2^2 + m_3 r_3^2$$
$$+ m_4 r_4^2)/2 \qquad (39)$$

$$I_{yy} = I_{zz} = (m_1(3r_1^2 + L1^2) + m_2(3r_2^2 + L2^2)$$
$$+ m_3(3r_3^2 + L3^2) + m_4(3r_4^2 + L4^2)$$
$$- 3m_1\dot{b}^2 t^2)/12 + m_{top}d_2^2 + m_1 d_1^2 \qquad (40)$$

where

r_1 = stage 1 radius

r_2 = stage 2 radius

r_3 = stage 3 radius

r_4 = stage 4 radius

I_{xx} = moment of inertia about the body frame x axis

I_{yy} = moment of inertia about the body frame y axis

I_{zz} = moment of inertia about the body frame z axis

The derivatives of the moments of inertia are given by:

$$dI_{xx}/dt = (-2m_o t\dot{b}^2 - \dot{m}r_1^2 + 3\dot{m}\dot{b}^2 t^2)/2 \qquad (41)$$

$$dI_{yy}/dt = dI_{zz}/dt = (-m_{top}d_2 d_1 + m_1 d_1^2) 2\dot{m}/M$$
$$- (3r_1^2 + L1^2)\dot{m}/12 - \dot{b}^2 m_o t/2$$
$$+ 3\dot{b}^2 \dot{m} t^2/4 - \dot{m}d_1^2 \qquad (42)$$

The moments of inertia and their derivatives appear in Eqs (6) - (8).

Aerodynamic Forces

The aerodynamic forces and their moments are extremely hard to model in a concise way without oversimplifying the terms. They depend on the velocity of the missile with

respect to the air, the local atmospheric pressure, density, and temperature as well as the missile's shape and angle of attack. The force equation will be presented first and each of the terms in the equation will be defined thereafter. The aerodynamic forces, in vector form, can be given by:

$$\underline{F}_a = -C_d \rho A |V_{cm}| \underline{V}_{cm}/2 \tag{43}$$

where

$\underline{F}_a = (X_a, Y_a, Z_a)\underline{E}$ = aerodynamic forces in the inertial frame

C_d = coefficient of drag of the missile

ρ = local atmospheric density

A = effective area of the missile

$\underline{V}_{cm}$ = velocity of the COM of the missile

The coefficient of drag varies with Mach number, Reynold's number, angle of attack, and shape of the missile, to name a few. The C_d used in this report is:

$$C_d = C_{d_0} + C_{d_\alpha} \alpha \tag{44}$$

where

C_d = drag coefficient

C_{d_0} = base drag coefficient

C_{d_α} = drag coefficient variance with angle of attack

α = angle of attack

Most references on the subject give good, accurate descriptions of the factors that influence C_d but few, if any, present the data required in this report. Reference 11 presents a graph (11:88) that shows that the base drag

17

coefficient varies with Mach number. A value of $C_{d_0} = 0.3$ was obtained from Reference 11 (11:88). There was no data available for C_{d_α} for this particular missile shape. There are, however, references to conical bodies in Krasnov (12) with $C_{d_\alpha} = 0.1$ so that is the value used for this study.

The variation of the density of the atmosphere with altitude was simulated using a simple atmospheric model (6:18-19). It uses the thermal lapse rate at varying heights above the Earth's surface to extrapolate between values in a lookup table.

The area of the missile effecting the aerodynamic forces is given by (9):

$$A = A_f \cos\alpha + A_s \sin\alpha \qquad (45)$$

where

A = effective area of the missile

A_f = frontal area of the missile

A_s = surface area of the missile

α = angle of attack

The frontal area is given by:

$$A_f = \pi r_1^2 \qquad (46)$$

where r_1 is the radius of the largest stage of the missile (stage 1). The surface area is given by:

$$A_s = \pi r_1^2 + 2\pi(r_1 L1 + r_2 L2 + r_3 L3 + r_4 L4) \qquad (47)$$

All of these variables have been defined earlier in this

18

chapter.

The angle of attack is defined as the angle between the velocity vector in the body frame and the x axis of the body frame. The velocity vector can be decomposed into its three components which produces the three components of the aerodynamic forces, namely, X_a, Y_a, and Z_a.

The aerodynamic moments are given by:

$$\underset{\sim}{M}_a = \underset{\sim}{d} \times \underset{\sim}{F}_a \tag{48}$$

where

$\underset{\sim}{M}_a = (L,M,N)\underset{\sim}{E}_r$ = aerodynamic moments in the body frame

$\underset{\sim}{d}$ = the moment arm which is given by:

$$\underset{\sim}{d} = -(\bar{x} - x_{cp})\hat{1} \tag{49}$$

where

$\bar{x}$ = distance between COM and bottom of missile

x_{cp} = distance between the center of pressure and the bottom of missile

The center of pressure is usually determined experimentally but since that is not possible in this case the data from Krasnov (12:442) will suffice. Krasnov reports that for a pointed body of revolution with a length of six to eight diameters, the center of pressure is approximately equal to 52% of the missile length, according to wind-tunnel tests. He chooses the top of the missile as his zero reference so 48% of the total missile length, 28.7 feet, will be used for this study. The length of this missile is about 11 diameters but

19

this difference should be insignificant. Since the center of pressure is behind the center of mass the missile has an inherent stability called "arrow stability".

Since the moment equations are in the body frame the aerodynamic moments must also be in the body frame. The transformation is made by use of the transformation matrix, $\underset{\sim}{A}_r$, and after the cross product is performed, Eq (48) expands to:

$$L = 0 \tag{50}$$

$$M = -(1/2)dC_d \rho A |V_{cm}| (A_{31}u + A_{32}v + A_{33}w) \tag{51}$$

$$N = (1/2)dC_d \rho A |V_{cm}| (A_{21}u + A_{22}v + A_{23}w) \tag{52}$$

All of the variables above have been defined earlier in this chapter.

Momentum Loss Term

As the burned propellant is exhausted out the nozzle it takes a small amount of angular momentum with it. A term must be included to account for the angular momentum imparted to the exhausted propellant or else the angular momentum will remain with the missile causing the spin rate to increase. This is not treated in Cornelisse's presentation but it is added to this analysis. Thomson shows that this angular momentum loss can be accounted for by (7:225):

$$M_x = -\dot{m} \rho 1^2 p \tag{53}$$

where

M_x = moment in the body frame x direction due to momentum loss

$\dot{m}$ = mass rate of flow

$\rho 1^2$ = the square of the offset distance of the COM flow and is estimated to be approximately $(1/2)r_1^2$

This term is then included in Eq (6) as shown. If this term is not included the spin rate rises to unacceptable rates and the system is not properly modelled.

21

Chapter Four <u>Checkout of Equations</u>

Introduction to Checkout

In order to ensure that the computer model of the missile's trajectory is accurate, each term was checked individually as it was added to the equations of motion. In addition, there are a number of tests that can be performed to make sure the computer results match with hand calculations. The first tests involve setting up driver routines to make sure the numerical integrator works satisfactorily. This was done and it showed that the numerical integrator gave considerably better results when the inputs were double precision rather than single precision. For example, if an input error was made in the fourth significant digit the error was obvious in the final results with single precision but it wasn't quite as pronounced with double precision accuracy. This is to be expected, of course, since the algorithm used extrapolates and corrects thousands of times in a single test case. As a result, double precision variables were used throughout the program.

Torque-Free Axisymmetric Rigid Body

The first terms to be checked out were the terms which make up Euler's equations for a torque-free, axisymmetric, rigid body. They are:

$$dp/dt = (I_{yy} - I_{zz})qr/I_{xx} \tag{54}$$

22

$$dq/dt = (I_{zz} - I_{xx})pr/I_{yy} \qquad (55)$$

$$dr/dt = (I_{xx} - I_{yy})pq/I_{zz} \qquad (56)$$

These equations are readily identifiable as simplifications of
Eqs (6) - (8). Since $I_{yy} = I_{zz}$ for an axisymmetric body,
$dp/dt = 0$ or p = constant. This leaves a pair of constant
coefficient, linear differential equations whose solution is
known (13). The magnitude of the angular velocity vector must
remain constant and the period of oscillation can be calcu-
lated by hand and compared with computer results. They
compared favorably to the fourth significant digit. The angle
between the angular velocity vector and the missile's symmetry
axis must also remain constant and this also proved to be
true.

Gravity and Position-Velocity Relations

The next terms to be checked were the gravity terms and
the position-velocity relationships. The gravity terms are
those found in Eqs (3) - (5) and represent the first terms
that effect the translational velocity of the missile. Since
the velocity of the missile is the time derivative of its
position, Eqs (10) - (12) were included at the same time. The
checkout used for these terms was to set the missile's
velocity equal to the orbital speed at the altitude chosen and
see if the missile would orbit the earth in the same period
that hand calculations predicted. This checked out to be

23

true.

Transformation Matrix

The transformation equations were the next set of
equations to be installed and checked. They are given by Eqs
(14) - (22). In order to check the equations the missile was
forced to cone about its angular momentum vector. This was
done by choosing A_{31} = cosθ where θ is the angle between the
angular momentum vector and the symmetry axis of the missile.
The angular momentum vector in the body frame is given by
(13:16):

$$\underset{\sim}{H}_{bod} = I_{xx}p\hat{\imath} + I_{yy}q\hat{\jmath} + 0\hat{k} \qquad (57)$$

The $\hat{k}$ component is chosen to be zero. The transformation
matrix will be computed so that:

$$\underset{\sim}{H}_{in} = \left[\underset{\sim}{A}_r\right]^T \underset{\sim}{H}_{bod} = 0\hat{\imath}_g + 0\hat{\jmath}_g + |H|\hat{k}_g \qquad (58)$$

where

$\quad\underset{\sim}{H}_{in}$ = the angular momentum vector in the inertial
$\qquad\qquad$ frame

$\quad\hat{\imath}_g, \hat{\jmath}_g, \hat{k}_g$ = the unit vectors in the inertial frame

This represents a set of equations that can be solved
simultaneously with the standard equations which relate
elements of rows and columns of a transformation matrix,
namely, the sum of the squares of the elements of any row or
column must equal one. After solving for the A_{ij}'s and using
them in the program, the A_{31} element should not vary with

24

time because it is a constant in a coning motion. That is what the computer results showed so it checks.

Jet Damping Terms

The jet damping terms are caused by the exhaust jet in a spinning missile and have a damping effect on the angular velocities. In Eqs (6) - (8) they are the terms, respectively:

$$\dot{m}x_e(y_e q + z_e r), \quad -\dot{m}qx_e^2, \quad -\dot{m}rx_e^2$$

They result from the cross product of the angular velocity with the center of mass flow offset vector, $\underset{\sim}{r}_e$. The higher order terms have been dropped since they are negligibly small compared to the terms retained. The magnitudes of the jet damping terms are several orders of magnitude larger than the moment of inertia terms already in place so an approximate solution can be used to check the jet damping terms' effects. The approximation is:

$$I_{zz}dr/dt = Mqx_e^2 \tag{59}$$

$$\int dr/r = - \int (Mx_e^2/I_{zz})dt \tag{60}$$

$$r(t) = r(t_o)\exp(Mx_e^2(t-t_o)/I_{zz}) \tag{61}$$

where

$\quad\quad r(t)$ = the angular velocity about the z body axis at time t

$\quad\quad r(t_o)$ = the initial angular velocity about the z body axis at time $t = t_o = 0$

By substituting the values that the program used for I_{zz}, M,

25

x_e, and $r(t_o)$ the approximate solution can be compared to the program's results as shown in Figure 4.1. The results show that the moment of inertia terms are almost insignificant and that the program is producing the predicted results. Figure 4.2 shows that the jet damping terms do reduce the angular velocities with time. In other words, jet damping has the desirable result of reducing the coning motion.

Thrust

The addition of the thrust terms to Eqs (3) - (5) is the next step. The thrust is in the body axis x direction and must be transformed to the inertial coordinate system by using the transformation matrix, A_r. In the next chapter, thrust misalignment will be used to control the angle of attack but for now the thrust will be in the body axis x direction only. The thrust terms were tested by simulating a vertical flight and a gravity turn trajectory. The simulations worked as expected with the flight path angle remaining at zero for the vertical flight and growing steadily in the gravity turn simulation.

Aerodynamic Forces and Moments

The aerodynamic forces and moments are extremely difficult to model accurately and also very expensive in computer time consumption. They are included in this report with many simplifying assumptions so that the computational

26

time of the program could be kept lower and to provide a certain amount of completeness. The terms were all hand checked at various stages of a flight simulation and the atmospheric model was tested by checking the density the program output at various altitudes.

Moments of Inertia

Up to this point, the moments of inertia were estimated by calculating the moments of a solid cylinder with the radius of the first stage and the length of the missile as its dimensions. With the arrival of more extensive data (10), a more accurate model was constructed. Again, the program's output was checked against hand calculations at various stages of a flight simulation and the results were the same. The equations for the moments of inertia are presented in Chapter Three in Eqs (39) and (40). Table 4.1 shows the values that were used in all the simulations after these equations were implemented.

Table 4.1   Minuteman III Data

	Length,ft	Radius,ft	Initial Mass,slugs
Stage 1	25.3	2.75	1571.14
Stage 2	13.1	2.15	482.38
Stage 3	7.1	2.15	253.31
Stage 4	14.1	2.15	55.32
Totals	59.9	--	2362.15

The stage 1 thrust used was 202,000 lb$_f$ and the stage 1 propellant mass was 1424.44 slugs.

The derivatives of the moments of inertia are given in Eqs (41) and (42) and were checked in the same way. Since the moments of inertia are decreasing with time because of the decrease in mass, the derivatives of the moments are negative. Since they are subtracted from the other moments in Eqs (6) – (8) their net effect is positive or, they decrease the jet damping effect. This is also the prediction of Cornelisse (8:78).

Figure 4.1 z-axis Rotational Velocity (r) as a Function of Time

Computer results- jet damping and moment of inertia terms

Estimate of jet damping term only

r (z-axis rotational velocity in rad/sec)

Time, t (seconds)

29
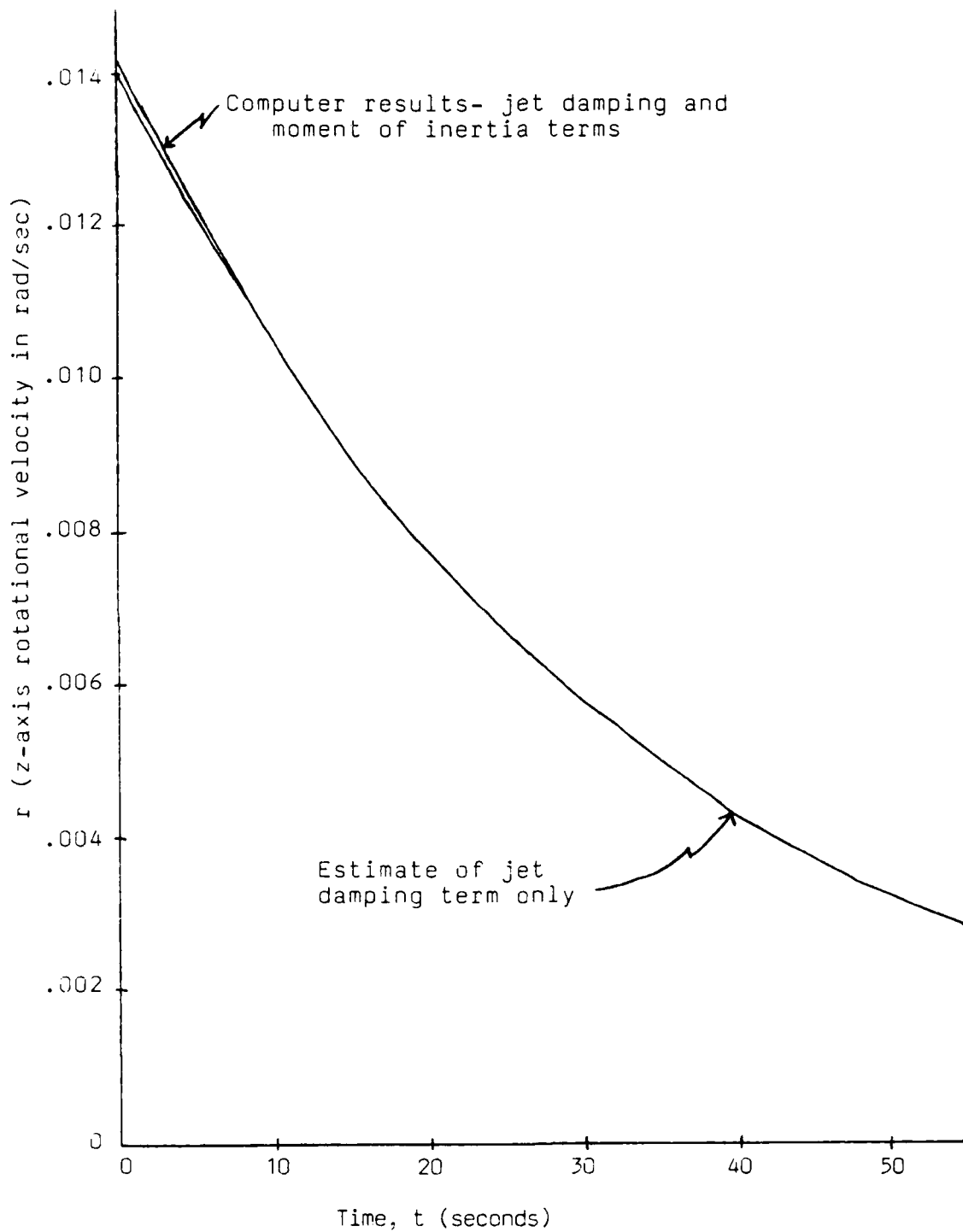
Figure 4.2   Coning Motion Decay Caused by the Jet Damping

r, angular velocity about the z-body axis (rad/sec)

q, angular velocity about the y-body axis (rad/sec)
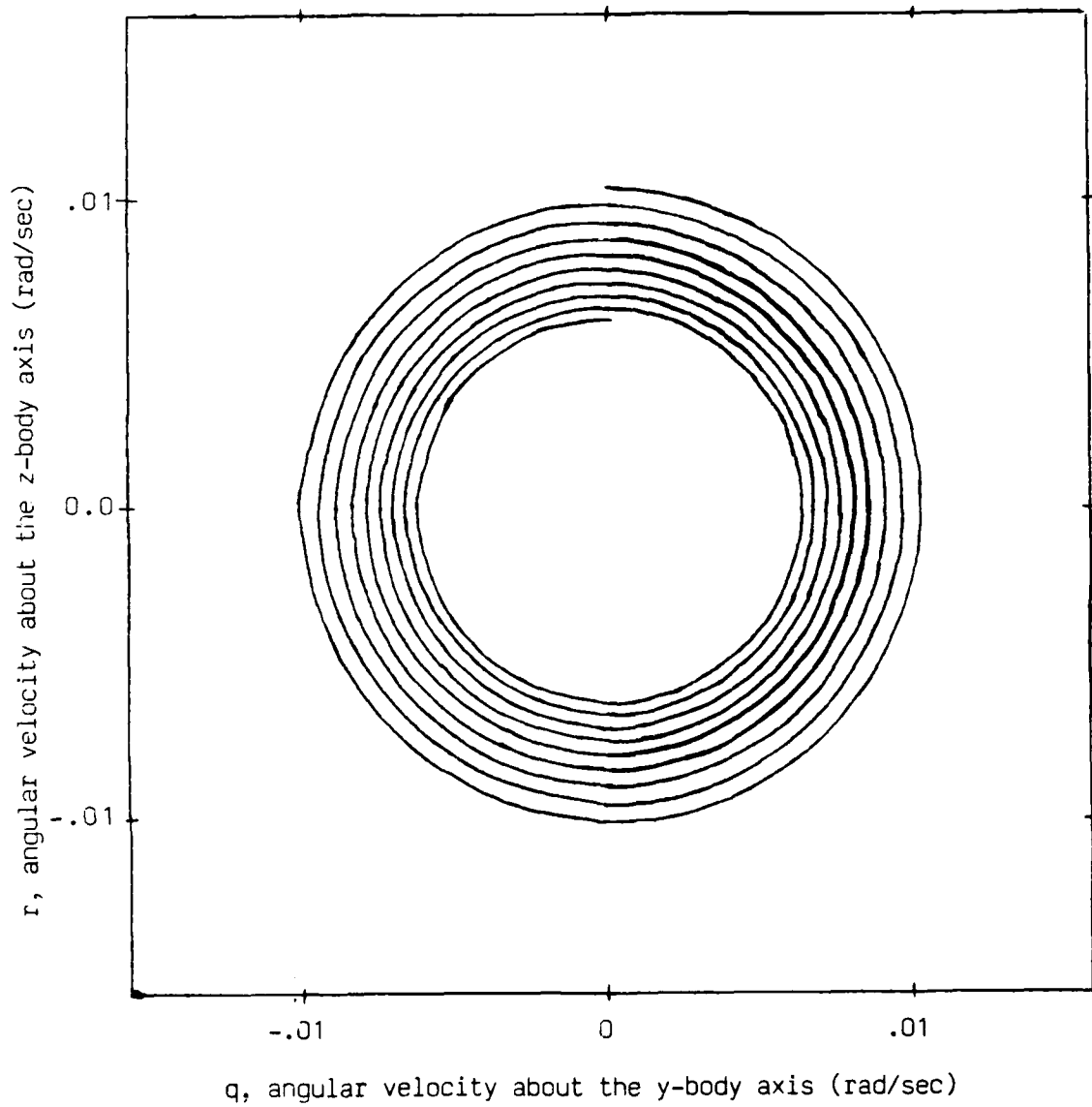
Chapter Five Feedback Control to Reduce Angle of Attack

Motivation for Feedback Controller

After all the equations had been included and all the terms checked individually, the set of equations was analyzed. Some of the same characteristics that were mentioned in Chapter Four must still be true for the complete system. Namely, the missile should fly a gravity turn trajectory and the angle of attack should drop off quickly and stay small. Figure 5.1 shows the flight path angle of the trajectory as a function of time. The flight path angle used throughout this report is defined as the angle between the inertial vertical axis, X, and the velocity vector in the inertial frame. Notice that the flight path angle rises to 0.59 radians or about 33° and stays there. This isn't normal. Figure 5.2 shows the angle of attack plotted as a function of time and that the missile flight simulation is flying the missile at a rather high angle of attack throughout the 61 seconds of flight. The angle of attack used throughout this report is defined as the angle between the missile's symmetry axis and the velocity vector in the body frame. The initial kick angle for the plots was 15° and the majority of the flight the missile is flying at an angle of attack greater then 0.0873 radians or 5°. Figures 5.1 and 5.2 were produced with only the thrust, gravity, and moment of inertia terms included in the simulation. The reason the angle of attack increases

31

during the flight is because the velocity vector falls away in a gravity turn but the nose of the missile doesn't follow it because it is coning about the angular momentum vector. So the feedback controller is implemented to reduce the angle of attack.

Derivation

The general idea behind the feedback controller is to compute the angle of attack about the y and z body frame axes and to feed back a restoring torque to the moment equations given in Eqs (7) and (8). These two angles of attack are related to the angle of attack that is being reduced through spherical trigonometry. The restoring torque is computed by first finding the missile's velocity in the body frame. The transformation matrix, $\underset{\sim}{A}_r$, is multiplied by the inertial velocity vector to find the components of the body frame velocity vector. Next, the angles of attack about the y and z body frame axes are computed by:

$$\theta_y = \arctan(v_k/v_i) \tag{62}$$

$$\theta_z = \arctan(v_j/v_i) \tag{63}$$

where

θ_y = the angle of attack about the y body frame axis

θ_z = the angle of attack about the z body frame axis

v_i = velocity component in the x body frame direction

v_j = velocity component in the y body frame direction

v_k = velocity component in the z body frame direction

32

Figure 5.3 shows how the angles of attack are related to the body frame velocity components.

Figure 5.3 Angles of Attack Diagram
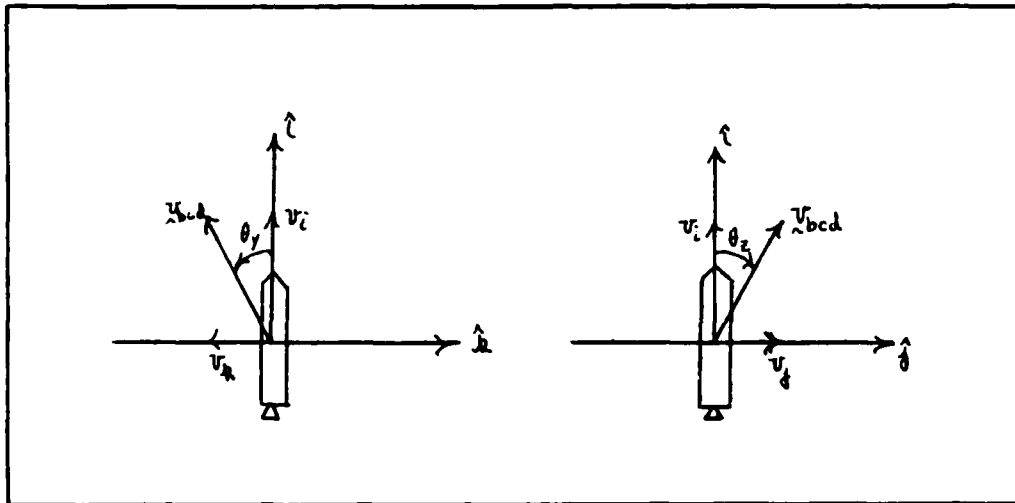
Next, the y and z components of the position vector from the missile's COM to the center of mass flow are computed. This is done by (14):

$$y_e = Kr + C\theta_z \tag{64}$$

$$z_e = Kq + C\theta_y \tag{65}$$

where

y_e = y axis component of the position vector to the COM flow

z_e = z axis component of the position vector to the COM flow

K = gain1

C = gain2

q = angular velocity about the y body frame axis

r = angular velocity about the z body frame axis

The gains, K and C, will be discussed a little later in this chapter. The three components of the body frame thrust can be computed from the components of the position vector to the COM flow using:

$$F_y = Ty_e / |r_e| \tag{66}$$

$$F_z = Tz_e / |r_e| \tag{67}$$

$$F_x = (T^2 - F_y^2 - F_z^2)^{1/2} \tag{68}$$

where

F_x, F_y, F_z = the three components of the body frame thrust

$|r_e|$ = magnitude of the position vector to the COM flow

T = thrust, a constant

The feedback terms added to Eqs (7) and (8) are:

$$MFEEDJ = F_x z_e \tag{69}$$

$$MFEEDK = -F_x y_e \tag{70}$$

where

MFEEDJ = feedback torque about the y body frame axis

MFEEDK = feedback torque about the z body frame axis

The gains, K and C, must be chosen so that the feedback torque will drive the angle of attack to an acceptably small value in a short period of time. They can not be too large, though, or else the gimbal limits of the missile's stage 1 nozzle will be exceeded. The gimbal limits for a MM III missile happen to be about 40°. By starting with Euler's

equations with a restoring torque term added, a simplified set
of equations can be solved to produce approximate values for
the gains. Euler's equations are:

$$I_{xx}\dot{p} + (I_{yy} - I_{zz})qr = 0 \tag{71}$$

$$I_{yy}\dot{q} + (I_{xx} - I_{zz})pr - F_x z_e = 0 \tag{72}$$

$$I_{zz}\dot{r} + (I_{yy} - I_{xx})pq - F_x y_e = 0 \tag{73}$$

Since $I_{yy} = I_{zz}$ for an axisymmetric body, Eq (71) reduces to
p = constant. Next, the assumption is made that the velocity
vector is a constant on the timescale of the coning motion so
that Eqs (72) and (73) become a set of two equations with two
unknowns. This assumption means that:

$$\dot{\theta}_y = q, \quad \ddot{\theta}_y = \dot{q}, \quad \dot{\theta}_z = r, \quad \ddot{\theta}_z = \dot{r}$$

Using this, along with Eqs (64) and (65), Eqs (72) and (73)
change to:

$$I_{yy}\ddot{\theta}_y + (I_{xx} - I_{yy})p\dot{\theta}_z - F_x(K\dot{\theta}_y + C\theta_y) = 0 \tag{74}$$

$$I_{yy}\ddot{\theta}_z + (I_{yy} - I_{xx})p\dot{\theta}_y - F_x(K\dot{\theta}_z + C\theta_z) = 0 \tag{75}$$

Putting Eqs (74) and (75) into matrix form:

$$\begin{bmatrix} I_{yy}/F_x & 0 \\ 0 & I_{yy}/F_x \end{bmatrix} \begin{pmatrix} \ddot{\theta}_y \\ \ddot{\theta}_z \end{pmatrix} + \begin{bmatrix} -K & p(I_{xx}-I_{yy})/F_x \\ p(I_{yy}-I_{xx})/F_x & -K \end{bmatrix} \begin{pmatrix} \dot{\theta}_y \\ \dot{\theta}_z \end{pmatrix}$$
$$+ \begin{bmatrix} -C & 0 \\ 0 & -C \end{bmatrix} \begin{pmatrix} \theta_y \\ \theta_z \end{pmatrix} = 0 \tag{76}$$

35

Substituting the following for (θ_y, θ_z):

$$\begin{pmatrix} \theta_y \\ \theta_z \end{pmatrix} = \underset{\sim}{\alpha} e^{\lambda t} \quad , \quad \begin{pmatrix} \dot{\theta}_y \\ \dot{\theta}_z \end{pmatrix} = \underset{\sim}{\alpha} \lambda e^{\lambda t} \quad , \quad \begin{pmatrix} \ddot{\theta}_y \\ \ddot{\theta}_z \end{pmatrix} = \underset{\sim}{\alpha} \lambda^2 e^{\lambda t}$$

yields:

$$\begin{bmatrix} I_{yy}\lambda^2/F_x - K\lambda - C & -p\lambda(I_{yy}-I_{xx})/F_x \\ p\lambda(I_{yy}-I_{xx})/F_x & I_{yy}\lambda^2/F_x - K\lambda - C \end{bmatrix} = 0 \qquad (77)$$

Which produces the following fourth order equation:

$$I_{yy}^2 \lambda^4/F_x^2 - 2KI_{yy}\lambda^3/F_x + (p^2(I_{yy}-I_{xx})^2/F_x^2$$
$$- 2CI_{yy}/F_x + K^2)\lambda^2 + 2KC\lambda + C^2 = 0 \qquad (78)$$

By substituting values for the known constants I_{xx}, I_{yy}, F_x, and p a fourth order equation can be set up in K and C. Routhe's stability criterion (15:185-188) can be used to show that in order for the system to be stable, $K < 0$. By means of trial-and-error it was found that for $K = -0.9$ and $C = -10$ the roots are $\lambda_{1,2} = -0.3019 \pm j3.139$, $\lambda_{3,4} = -0.1538 \pm j1.599$. These roots are all stable with the longest damping period being about six seconds. This result is satisfactory because for shorter damping periods the missile goes through wild oscillations to achieve the desired angle of attack causing the aerodynamic forces to take control. Figures 5.4 and 5.5 show the missile's response without the feedback controller and Figures 5.6 and 5.7 show the same test case run with the

36

feedback controller installed. Comparing Figures 5.4 and 5.6, notice that the angles of attack around the y and z body axes are reduced quicker with the controller installed than without it, which is the result desired. A comparison of Figures 5.5 and 5.7 shows that without the feedback controller the total angle of attack keeps oscillating even after a long period of time but with the controller installed, the angle of attack goes to zero in about 25 seconds. A decaying exponential curve is superimposed on the angle of attack plot in Figure 5.7. The exponential curve was generated by:

$$\alpha = \alpha_0 \exp(-\lambda t) \tag{79}$$

where

α = angle of attack

α_0 = initial kick angle

λ = the longest decay constant = 0.1538

Notice that the simulation stays well below the predicted angle of attack during the first 25 seconds. This is probably due to the jet damping terms which help damp out the angle of attack.

37

Figure 5.1   Flight Path Angle without Feedback Controller
              Moment of Inertia terms only

Time, t (seconds)

38
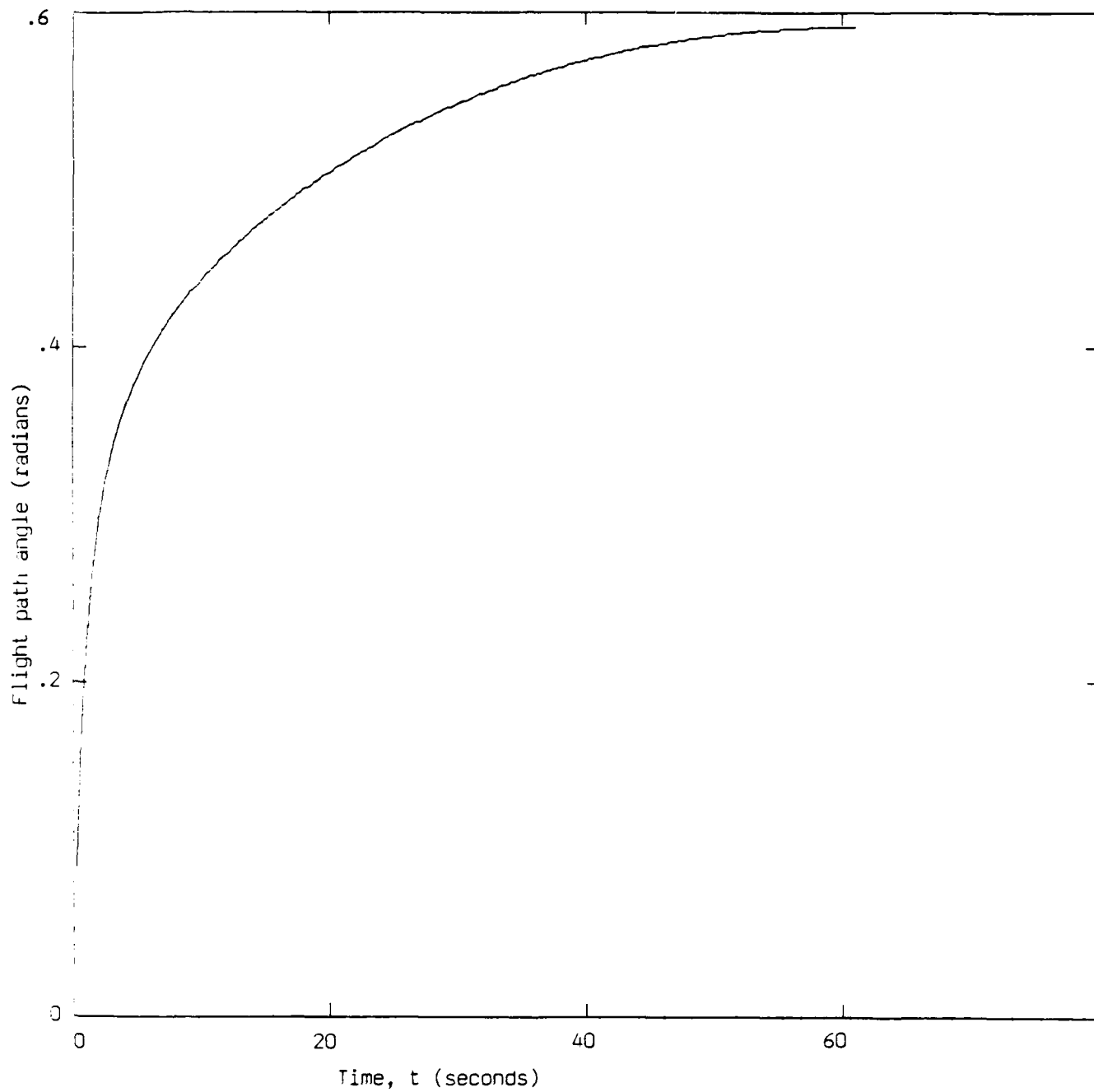
Figure 5.2 Angle of Attack Without Feedback Controller
Moment of Inertia Terms Only

39
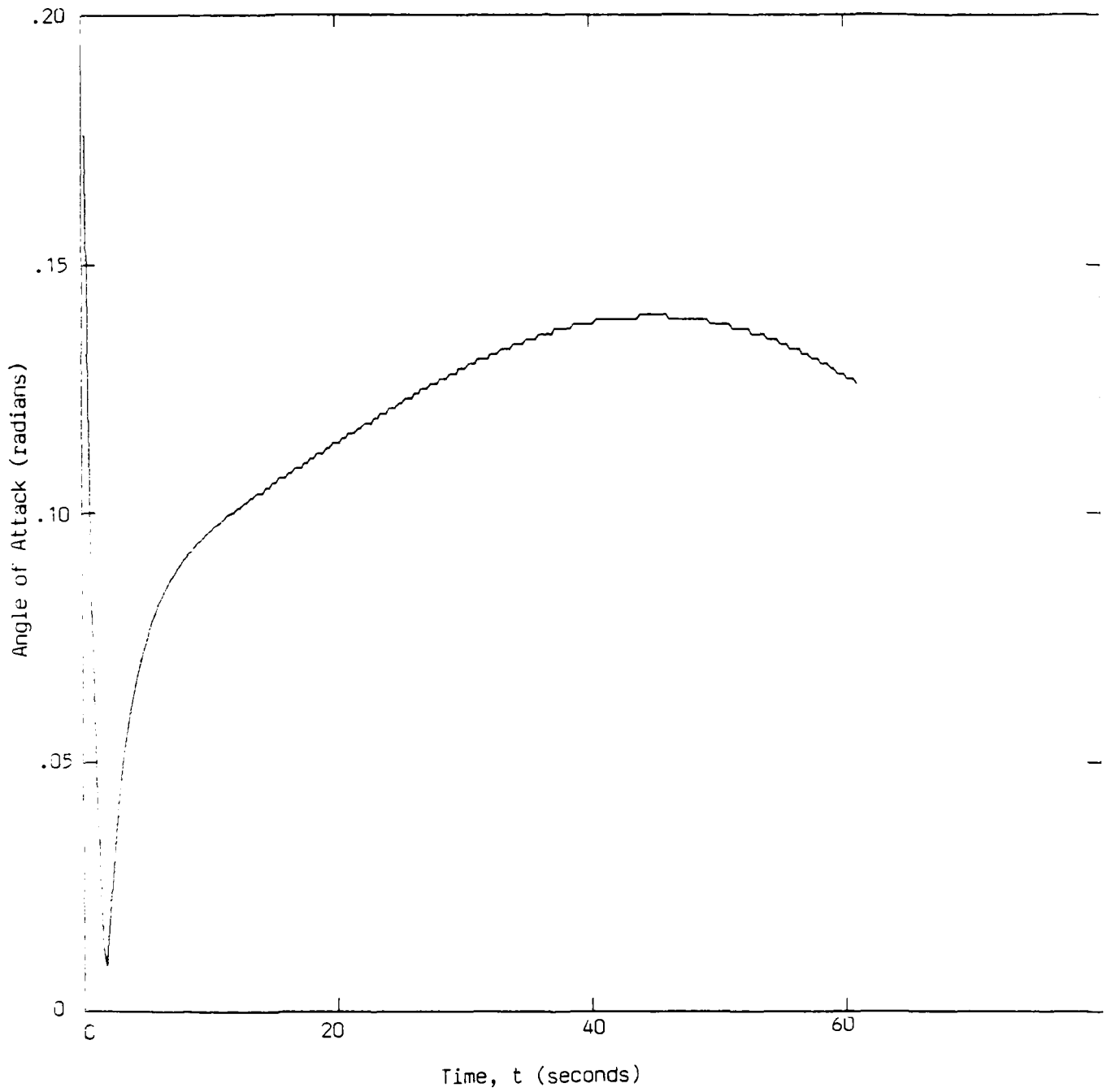
Figure 5.4   y-direction vs. z-direction Angles of Attack
             with All Terms Included Except Feedback Controller

y-direction angle of attack (radians)
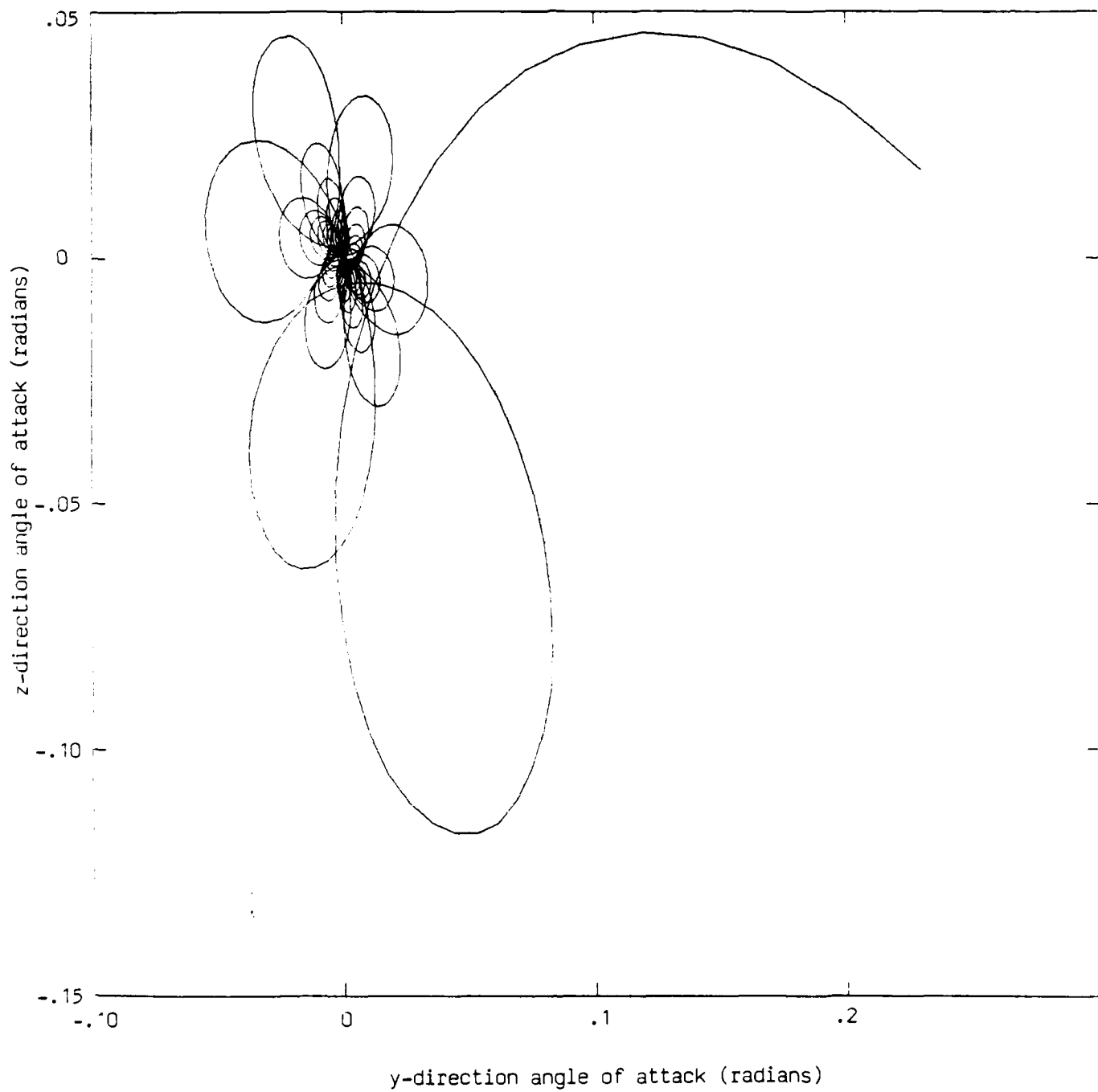
Figure 5.5 Angle of Attack with All Terms Included
          Except Feedback Controller

41
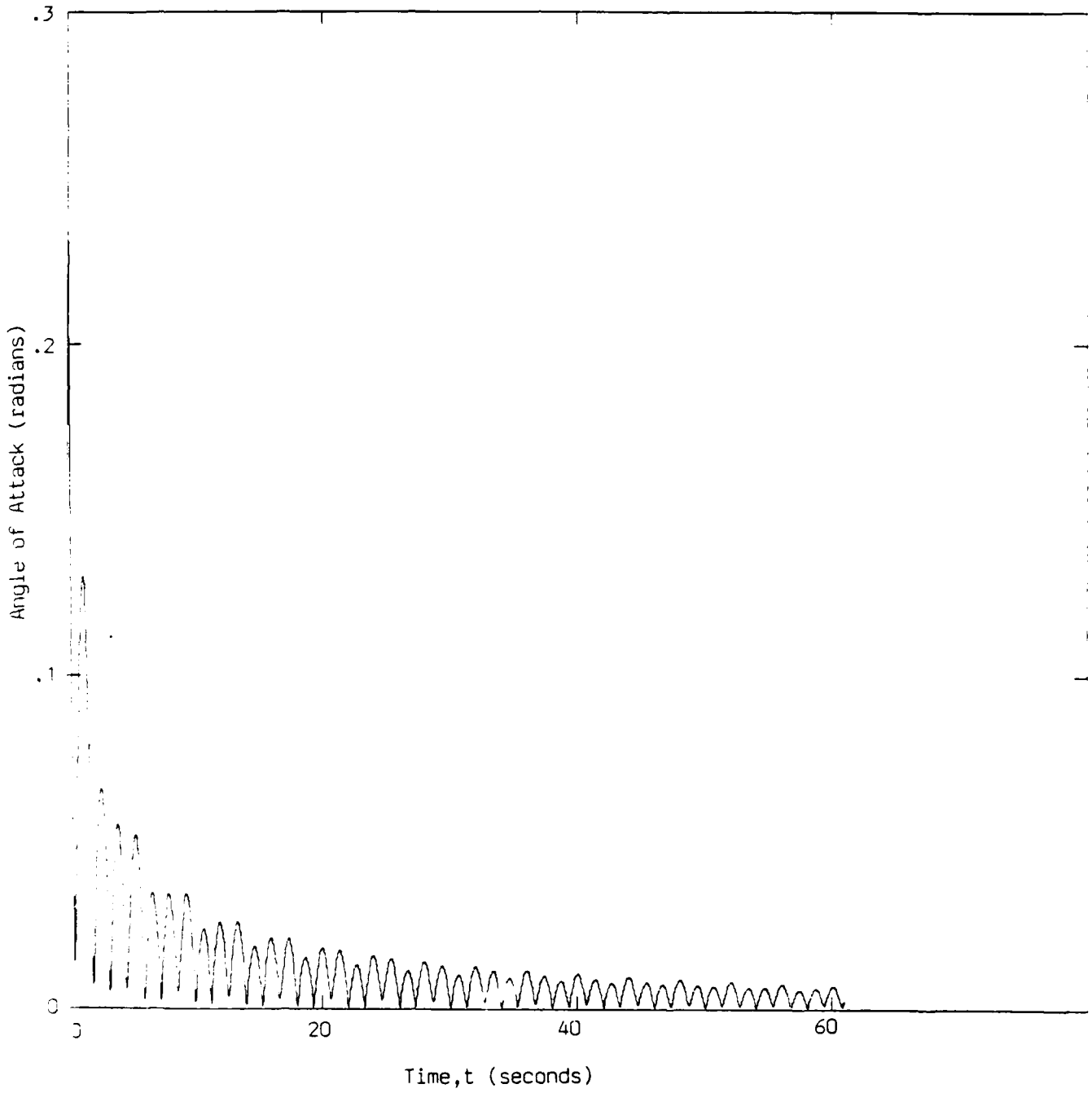
Figure 5.6 y-direction vs. z-direction Angles of Attack
with Controller

z-direction angle of attack (radians)

.05

0

-.05

-.10

-.15

-.1   0   .1   .2

y-direction angle of attack (radians)

42
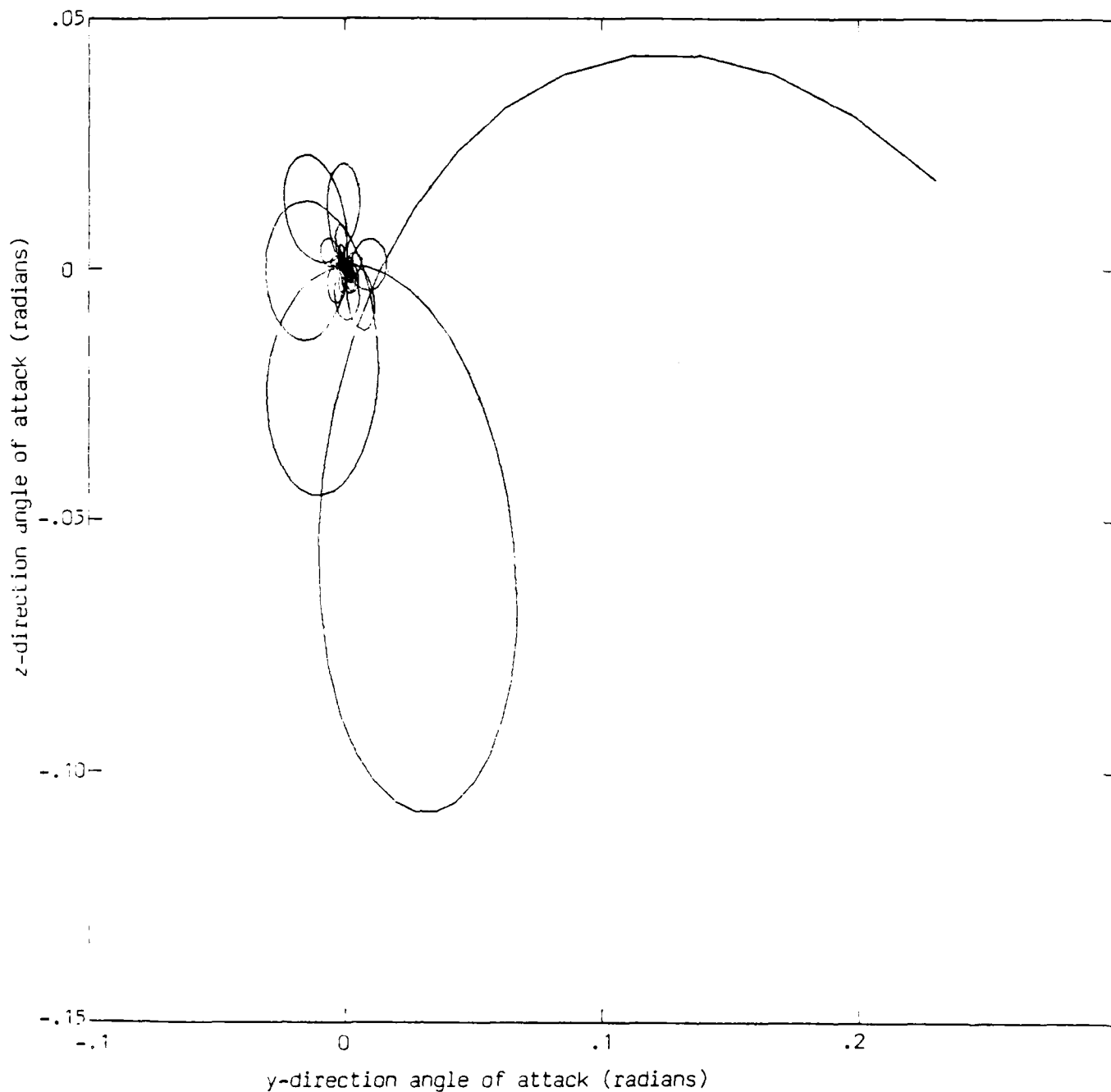
Figure 5.7 Angle of Attack with Controller

DECAYING EXPONENTIAL

Time, t (seconds)

Angle of Attack (radians)

43
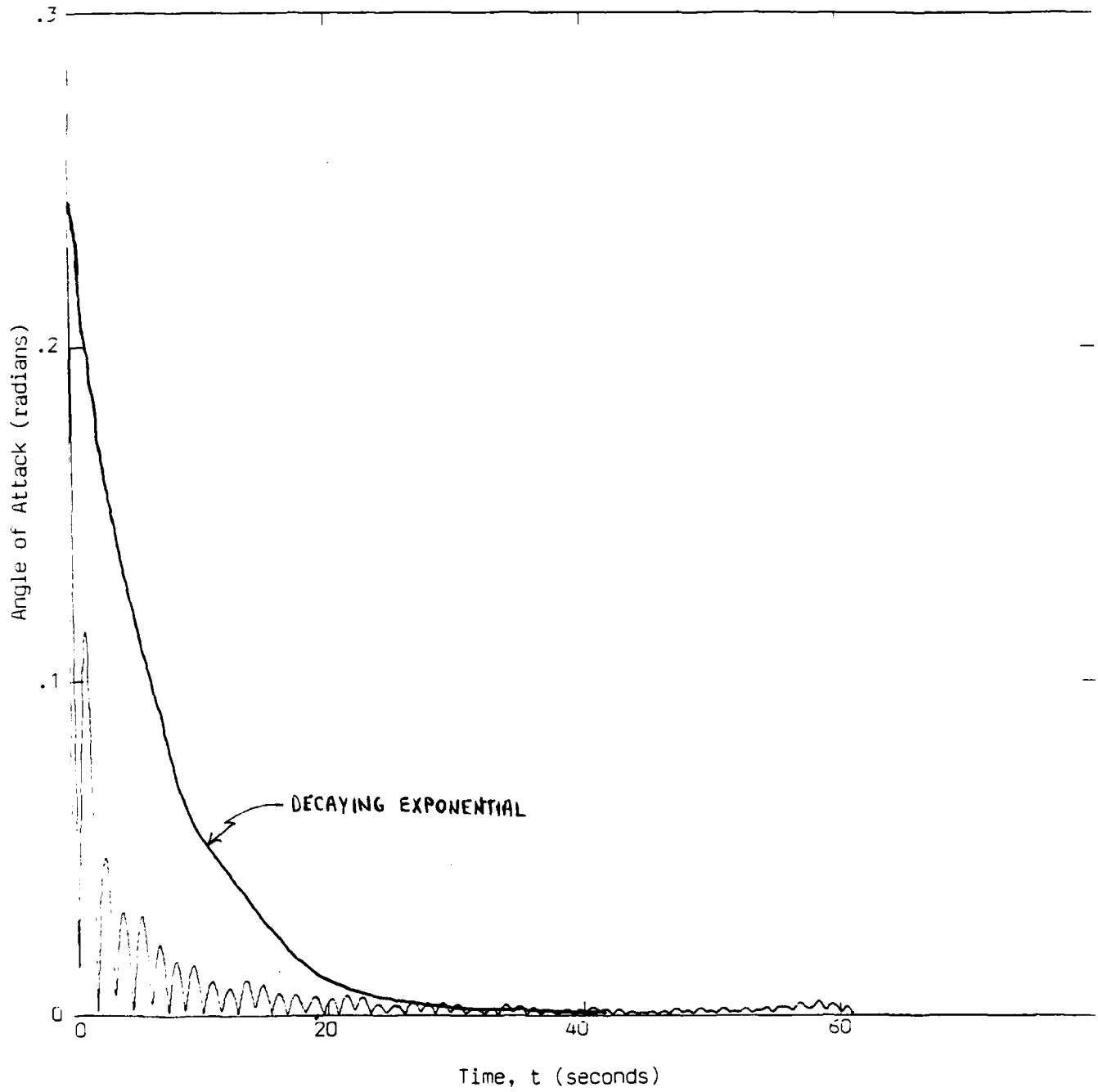
Chapter Six Results and Conclusions

Position - Velocity Errors

The first noticeable difference in the comparison of the
trajectories of a spinning and nonspinning missile is that the
stage 1 burnout state vectors are different. Table 6.1 lists
the burnout position and velocity, relative to the launch
point, for three cases, namely, 15° initial kick angle with
and without symmetry axis spin and 16.5° kick angle with
spin.

Table 6.1   Position - Velocity Comparisons

	15°, no spin	15°, spin	16.5°, spin
X, ft	131,680	134,650	131,760
Y, ft	0	10,418	11,072
Z, ft	-80,106	-72,713	-78,954
u, ft/sec	5,086	5,246	5,092
v, ft/sec	0	487	516
w, ft/sec	-3,723	-3,394	-3,673
$|V_{cm}|$, ft/sec	6,303	6,267	6,300

In the last two cases the spin rate was 1.571 rad/sec. The
altitude, represented by x in the table, differs by about 3000
ft between the spinning and nonspinning cases with the same
initial kick angle. By spinning the missile, the altitude
achieved from the same initial kick angle is greater but the

44

downrange distance is less. The differences between the y direction position and velocity is explained by the fact that the missile's flight path describes a coning motion in the spinning case which causes some dispersion in the y direction. This could probably be eliminated by introducing a small initial kick angle in the negative y direction. The V_{cm} row in the table represents the magnitude of the velocity vector for each case. Notice that the 16.5^O, spinning case and the 15°, nonspinning case are nearly the same. The x and z position components compare favorably for these two cases also, so it seems that spinning the missile at the relatively slow rate of 1.571 rad/sec forces the trajectory analyst to choose a slightly larger initial kick angle in order to achieve the same burnout vector as the nonspinning case. Figure 6.1 shows the relationship between stage 1 burnout altitude and initial kick angle in the same flight regime as that of Table 6.1. The altitude that the 15°, nonspinning case achieves corresponds to the 16.5^O, spinning case just as Table 6.1 shows. The conclusion is that there is almost no performance penalty due to spin.

Figure 6.1   Burnout Altitude vs Initial Kick Angle
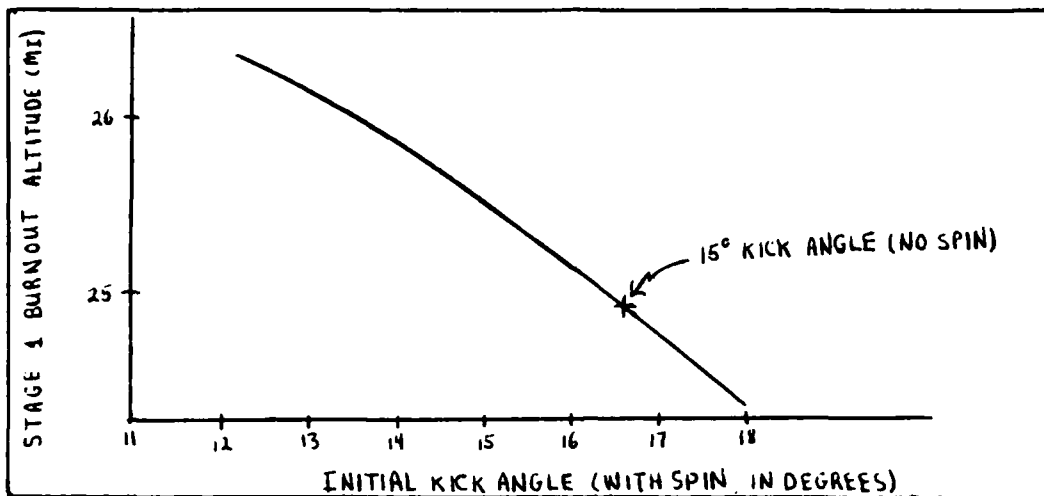
Gimbal Angles

The stage 1 engine's expansion nozzle on most missiles is
gimballed so that minor corrections can be made to the flight
path during the boost phase. As stated before, the MM III
gimbal angle limit is about 40° so this is a hardware
limitation that must be considered in this study. Figure 6.2
shows a "worst case" plot of gimbal angle variation with time.
The feedback controller produces the largest restoring moments
during the first few seconds of flight when the angle of
attack is the largest due to the initial kick angle. A high
spin rate, 20 rad/sec, was chosen with an initial kick angle
of 15° to see if the spin rate produced gimbal angles that
were too large. The plot shows that even at high spin rates

46

the gimbal angle only changes about 11° at most. This is well within the hardware limits of the missile system. The rate at which the gimbal angle changes may be of concern though. The gimbal angle changes about 11° in 0.4 seconds. This may be a concern if the gimbal actuator can not respond at this rate.

Conclusions

The model tested and discussed in this report was subjected to varying spin rates and compared to nonspinning cases. It was necessary to design a feedback controller to minimize the angle of attack and with this modification included it seems to be possible to spin a missile without adverse effects to the trajectory. Some adjustments in the trajectory must be made to produce the same stage 1 burnout state vector as the nonspinning model but the variations are predictable so the burnout vector is also predictable.

Figure 6.2 Nozzle Gimbal Angle as a Function of Time
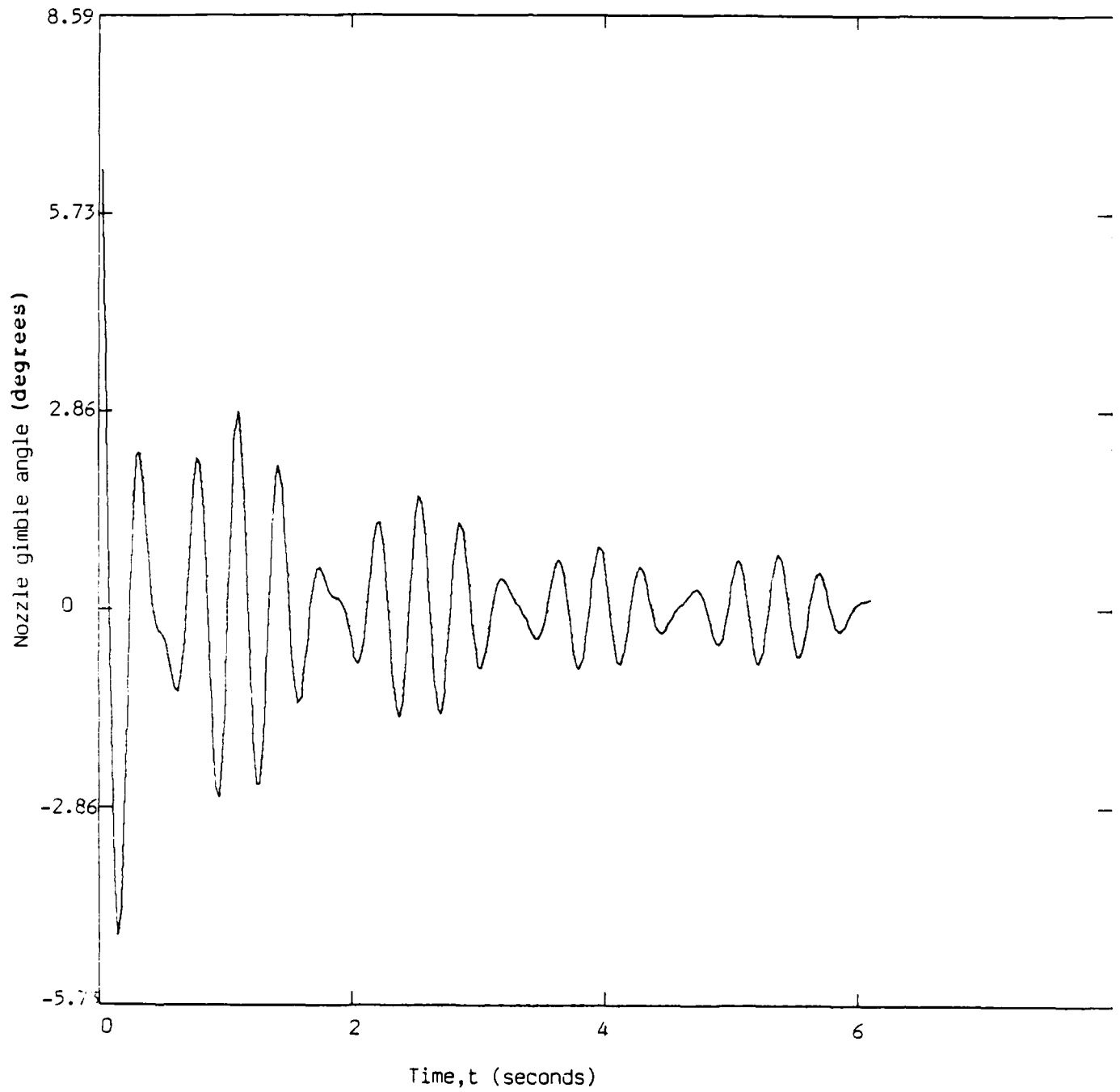
Appendix A

A listing of the program which was used for this study is shown below. It was run with the input file shown in Appendix B on a VAX 11/785 computer with a VMS operating system.

```
c**************************************************************
c
      program misspin
c
c**************************************************************
c     program misspin is a simple dynamics propagator
c     it takes input state vector y and propagates
c     it from to to tf.
c written by: Capt Robert W. Bandstra  AFIT/GA-85D
c
c yy(i) is an array to read the a matrix into.
c t is the time
c to is the initial time
c tf is the final time
c y(i,j) is the array that contains the state vectors
c n is the number of equations
c h is the timestep
c mode is not currently used
c n1 is the number of times the outside loop is performed
c n2 is the number of times the inside loop is performed
c nstp is used as an input to vary the step size
c
      common /ham/ t,y(18,4),f(18,4),err(18),n,h,mode
      double precision t,y,f,err,h,tf
      common /miscon/ l1,l2,l3,l4,m0,m2,m3,m4,rad,r2,r3,r4,
     1 bdot,mtop,xtop,xbot,alfa,flpa
      double precision l1,l2,l3,l4,m0,m2,m3,m4,rad,r2,r3,r4,
     1 bdot,mtop,xtop,xbot,alfa,flpa
      common /count/ j
      common /feedbk/ mfeedj,mfeedk,xe,ye,ze,fx,fy,fz,
     1 thetaj,thetak
      double precision mfeedj,mfeedk,xe,ye,ze,fx,fy,fz,
     1 thetaj,thetak
      real*8 yy(18)
c
      j = 0
c j is a switch that makes sure a section of atmos is only
c   executed once.
c
c   open input file
c
```

```fortran
        open(2,file='input.dat2',status='old')
c
c       open output files
c
        open(3,file='output.dat2',status='old')
        open(4,file='plot.dat3',status='old')
        open(7,file='plot.dat4',status='old')
c
c       read input
c
        read (2,10)to,tf
        read (2,20)(y(i,1),i=1,3)
        read (2,20)(y(i,1),i=4,6)
        read (2,20)(y(i,1),i=7,9)
        read (2,20)(yy(i),i=10,12)
        read (2,20)(yy(i),i=13,15)
        read (2,20)(yy(i),i=16,18)
   10   format(2x,2e20.13)
   20   format(2x,3e20.13)
c
c  dir. cos must be converted from y (angles) to cos(y)
c
        do 25 i = 10,18
        y(i,1) = dcosd(yy(i))
   25   continue
        y(12,1) = -y(12,1)
c
c  read mode, number of steps for numerical integration,
c       and plot flag
c
        read (2,30)mode,nstp,iplot
   30   format(2x,i5,i6,i5)
c
c       print ics
c
   50   format(//,2x,'dynamics propagator',//,2x,'to, tf:',
     1      2e20.13,//,2x,'initial state vector',
     2      /,2x,3e20.13,/,2x,3e20.13,/,2x,3e20.13,/,
     3      /,2x,3e20.13,/,2x,3e20.13,/,2x,3e20.13,/,
     4      2x,'mode, stps:',2i6,/)
        write(3,50)to,tf,(y(i,1),i=1,18),mode,nstp
        call strtup
c
c       setup rest of haming initialization
c
c       number of odes:
        n = 18
        t = to
c
c  timestep calculation
c  nl sets the number of points sent to the output files
c
```

```fortran
      nl = min0(244,nstp)
      n2 = 1 + nstp/nl
      h = (tf - to)/(nl*n2)
c
c     initialize haming
c
      nxt = 0
c  haming initialization flag
      call haming(nxt)
      if(nxt.eq.0) stop 99
c
c     numerical integration loop - one timestep per call
c
      do 1000 int = 1,nl
      do 900 nit2 = 1,n2
      call haming(nxt)
  900 continue
      write(3,61) t,alfa,flpa,thetaj,thetak
   61 format(2x,'t =',e16.9,/,2x,
     2 'alfa =',e20.13,' flpa =',e20.13,/,2x,
     3 'thetaj =',e16.9,' thetak =',e16.9)
c     write(3,60)t,(y(i,nxt),i=1,18),alfa,flpa,thetaj,thetak
   60 format(/,2x,'t =',e20.13,/,2x,'y =',3e20.13,/,2x,
     1 3e20.13,/,2x,3e20.13,/,2x,3e20.13,/,2x,3e20.13,/,2x,
     2 3e20.13,/,2x,'alfa =',e20.13,' flpa =',e20.13,/,2x,
     3 'thetaj =',e16.9,' thetak =',e16.9)
      write(4,95) t,alfa
      write(7,95) t,flpa
   95 format(2x,2e10.3)
      if(iplot.ge.0) write(4,90) y(8,nxt),y(9,nxt)
   90 format(2x,2e20.13)
 1000 continue
c
c     print final position
c
      write(3,70)(y(i,nxt),i=1,18)
   70 format(/,2x,'at time tf:',/,2x,'state vector is',/,
     1      2x,e20.13,/,2x,e20.13,/,2x,e20.13,/
     2      2x,e20.13,/,2x,e20.13,/,2x,e20.13,/
     3      2x,e20.13,/,2x,e20.13,/,2x,e20.13,/
     4      2x,e20.13,/,2x,e20.13,/,2x,e20.13,/
     5      2x,e20.13,/,2x,e20.13,/,2x,e20.13,/
     6     ,2x,e20.13,/,2x,e20.13,/,2x,e20.13,/)
      close(unit=2)
      close(unit=3)
      close(unit=4)
      close(unit=7)
      stop
      end
c*****************************************************************
c
      subroutine strtup
```

```
c
c****************************************************************
c   set up missile parameters at beginning of flight
c
c   tmass is the total mass of the missile at launch in slugs
c   rad is the radius cf stage i in ft.
c   r2 is the radius of stage ii in ft.
c   r3 is the radius of stage iii in ft.
c   r4 is the radius of stage iv in ft.
c   l1 is the length of stage i in ft.
c   l2 is the length of stage ii in ft.
c   l3 is the length of stage iii in ft.
c   l4 is the length of stage iv in ft.
c   tlen is the total length of missile at launch in ft.
c   gm is the grav. parameter of the earth in ft**3/sec**2
c   thrust is the thrust of stage i in lbf.
c   m0 is the initial mass of stage i in slugs
c   m2 is the mass of stage ii in slugs
c   m3 is the mass of stage iii in slugs
c   m4 is the mass of stage iv in slugs
c   mtop is the sum of m2, m3, & m4 in slugs
c   xtop is the location of the com of the top 3 stages in ft
c   maspro is the init. mass of the stage 1 prop. in slugs
c   mdot is the mass flow rate of stage i in slug/sec
c   burnt is the stage 1 engine burn time in sec.
c   bdot is the rate of change of inner radius
c           of stage 1 propellant in ft/sec
c   bf is the final inner rad. of stage 1 at end of burn in ft
c   rho2sq is the square of the offset distance of the center
c           of mass flow in ft**2
c
      common /misdat/ gm,thrust,mdot,tmass,burnt,maspro,mass,
     1 xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,gain1,
     2 gain2,gain3,gain4
      double precision gm,thrust,mdot,tmass,burnt,maspro,
     1 mass,xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,
     2 gain1,gain2,gain3,gain4
      common /miscon/ l1,l2,l3,l4,m0,m2,m3,m4,rad,r2,r3,r4,
     1 bdot,mtop,xtop,xbot,alfa,flpa
      double precision l1,l2,l3,l4,m0,m2,m3,m4,rad,r2,r3,r4,
     1 bdot,mtop,xtop,xbot,alfa,flpa
      double precision tlen,bf,x2,x3,x4
c
c   read missile parameters
c
      read(2,10)  tmass,rad,tlen
      read(2,10)  thrust,mdot,burnt
      read(2,10)  l1,l2,l3
      read(2,10)  l4,m0,m2
      read(2,10)  m3,m4,maspro
      read(2,10)  r2,r3,r4
      read(2,11)  gain1,gain2
```

52

```
      read(2,11) gain3,gain4
   10 format(2x,3e20.13)
   11 format(2x,2e20.13)
c
c  print missile parameters
c
      write(3,20) tmass,rad,tlen,thrust,mdot,burnt,l1,l2,l3,
     1 l4,m0,m2,m3,m4,maspro,r2,r3,r4,gain1,gain2,gain3,gain4
   20 format(/,2x,'tmass =',e16.9,' rad =',e16.9,/,2x,
     1'tlen =',e16.9,' thrust =',e16.9,/,2x,'mdot =',e16.9,
     2' burnt =',e16.9,/,2x,'l1 =',e16.9,' l2 =',e16.9,/,2x,
     3'l3 =',e16.9,' l4 =',e16.9,/,2x,'m0 =',e16.9,' m2 =',
     4 e16.9,/,2x,'m3 =',e16.9,' m4 =',e16.9,/,2x,'maspro =',
     5 e16.9,' r2 =',e16.9,/,2x,'r3 =',e16.9,' r4 =',e16.9,
     6 /,2x,'gain1 =',e16.9,' gain2 =',e16.9,
     7 /,2x,'gain3 =',e16.9,' gain4 =',e16.9)
      gm =    1.407646882d+16
      bf = rad - 0.08d+00
      bdot = bf/burnt
      mtop = m2 + m3 + m4
      xbot = l1/2.0d+00
      x2 = l1 + l2/2.0d+00
      x3 = l1 + l2 + l3/2.0d+00
      x4 = l1 + l2 + l3 + l4/2.0d+00
      xtop = (x2*m2+x3*m3+x4*m4)/mtop
      rho2sq = rad**2.d+00/2.d+00
      write(3,30) bf,bdot,mtop,xbot,xtop,rho2sq
   30 format(2x,'bf =',e20.13,' bdot =',e20.13,/,2x,'mtop ='
     1 ,e20.13,' xbot =',e20.13,/,2x,'xtop =',e20.13,
     2 ' rho2sq =',e20.13)
      return
      end
c  ***********************************************************
c
      subroutine haming(nxt)
c
c***********************************************************
c
c      haming is an ordinary differential equations integrator
c      it is a fourth order predictor-corrector algorithm
c      which means that it carries along the last four
c      values of the state vector, and extrapolates these
c      values to obtain the next value (the prediction part)
c      and then corrects the extrapolated value to find a
c      new value for the state vector.
c
c      the value nxt in the call specifies which of the 4
c      values of the state vector is the 'next' one.
c      nxt is updated by haming automatically, and is zero on
c      the first call
c
c      the user supplies an external routine rhs(nxt) which
```

53

```
c        evaluates the equations of motion
         common /ham/ x,y(18,4),f(18,4),errest(18),n,h,mode
         double precision x,y,f,errest,h,hh,xo
c
c        all of the good stuff is in this common block.
c        x is the independent variable ( time )
c        y(18,4) is the state vector- 4 copies of it, with nxt
c                    pointing at the next one
c        f(18,4) are the equations of motion, again four copies
c                    a call to rhs(nxt) updates an entry in f
c        errest is an estimate of the truncation error
c                    - normally not used
c        n is the number of equations being integrated
c        h is the time step
c        mode is 0 for just eom, 1 for both eom and eov
c
         tol = 0.0000000001
c        switch on starting algorithm or normal propagation
         if(nxt) 190,10,200
c
c        this is hamings starting algorithm....a predictor -
c        corrector needs 4 values of the state vector, and you
c        only have one - the initial conditions.
c        haming uses a picard iteration (slow and painfull) to
c        get the other three.
c        if it fails, nxt will still be zero upon exit,
c        otherwise nxt will be 1, and you are all set to go
c
      10 xo = x
         hh = h/2.0d+00
         call rhs(1)
         do 40 l = 2,4
         x = x + hh
         do 20 i = 1,n
      20 y(i,l) = y(i,l-1) + hh*f(i,l-1)
         call rhs(1)
         x = x + hh
         do 30 i = 1,n
      30 y(i,l) = y(i,l-1) + h*f(i,l)
      40 call rhs(1)
         jsw = -10
      50 isw = 1
         do 120 i = 1,n
         hh = y(i,1) + h*( 9.0d+00*f(i,1) + 19.0d+00*f(i,2)
        1     - 5.0d+00*f(i,3) + f(i,4) ) / 24.0d+00
         if( dabs( hh - y(i,2)) .lt. tol ) go to 70
         isw = 0
      70 y(i,2) = hh
         hh = y(i,1) + h*(f(i,1)+4.0d+00*f(i,2)+f(i,3))/3.0d+00
         if(dabs(hh-y(i,3)) .lt. tol) go to 90
         isw = 0
      90 y(i,3) = hh
```

```fortran
      hh = y(i,1) + h*( 3.0d+00*f(i,1) + 9.0d+00*f(i,2)
     1      + 9.0d+00*f(i,3) + 3.0d+00*f(i,4) ) / 8.0d+00
      if( dabs(hh-y(i,4)) .lt. tol ) go to 110
      isw = 0
  110 y(i,4) = hh
  120 continue
      x = xo
      do 130 l = 2,4
      x = x + h
  130 call rhs(l)
      if(isw) 140,140,150
  140 jsw = jsw + 1
      if(jsw) 50,280,280
  150 x = xo
      isw = 1
      jsw = 1
      do 160 i = 1,n
  160 errest(i) = 0.0
      nxt = 1
      go to 280
  190 jsw = 2
      nxt = iabs(nxt)
c
c     this is hamings normal propagation loop -
c
  200 x = x + h
      npl = mod(nxt,4) + 1
      go to (210,230),isw
c     permute the index nxt modulo 4
  210 go to (270,270,270,220),nxt
  220 isw = 2
  230 nm2 = mod(npl,4) + 1
      nml = mod(nm2,4) + 1
      npo = mod(nml,4) + 1
c
c     this is the predictor part
c
      do 240 i = 1,n
      f(i,nm2) = y(i,npl) + 4.0d+00*h*( 2.0d+00*f(i,npo)
     1      - f(i,nml) + 2.0d+00*f(i,nm2) ) / 3.0d+00
  240 y(i,npl) = f(i,nm2) - 0.925619835*errest(i)
c
c     now the corrector - fix up the extrapolated state
c     based on the better value of the equations of motion
c
      call rhs(npl)
      do 250 i = 1,n
      y(i,npl) = (9.0d+00*y(i,npo) - y(i,nm2) + 3.0d+00*h*
     1      (f(i,npl) + 2.0d+00*f(i,npo) - f(i,nml)))/8.0d+00
      errest(i) = f(i,nm2) - y(i,npl)
  250 y(i,npl) = y(i,npl) + 0.0743801653 * errest(i)
      go to (260,270),jsw
```

```fortran
      260 call rhs(npl)
      270 nxt = npl
      280 return
          end
c****************************************************************
c
          subroutine rhs(nxt)
c
c****************************************************************
c
c   rhs calculates the equations of motion of a spinning
c   missile in flight.
c   y(1-3,nxt) are the x,y,z comp. of the position vector
c   y(4-6,nxt) are the x,y,z comp. of the trans. vel. vector
c   y(7-9,nxt) are the x,y,z comp. of the rot. vel. vector
c   rcm is the mag. of the COM position vector in ft
c   y(10-18,nxt) are the dir. cos relationships between the
c      body frame and the inertial, earth-centered frame.
c   thrx is the inertial frame x-dir. thrust in lbf.
c   thry is the inertial frame y-dir. thrust in lbf.
c   thrz is the inertial frame z-dir. thrust in lbf.
c
          common /ham/ t,y(18,4),f(18,4),errest(18),n,h,mode
          double precision t,y,f,errest,h
          common /misdat/ gm,thrust,mdot,tmass,burnt,maspro,mass,
         1 xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,gainl,
         2 gain2,gain3,gain4
          double precision gm,thrust,mdot,tmass,burnt,maspro,
         1 mass,xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,
         2 gainl,gain2,gain3,gain4
          common /aero/ xa,ya,za,l,m,nl
          double precision xa,ya,za,l,m,nl
          common /feedbk/ mfeedj,mfeedk,xe,ye,ze,fx,fy,fz,
         1 thetaj,thetak
          double precision mfeedj,mfeedk,xe,ye,ze,fx,fy,fz,
         1 thetaj,thetak
          double precision rcm,thrx,thry,thrz
          call mominr(nxt)
          call feedbk(nxt)
          thrx = y(10,nxt)*fx + y(13,nxt)*fy + y(16,nxt)*fz
          thry = y(11,nxt)*fx + y(14,nxt)*fy + y(17,nxt)*fz
          thrz = y(12,nxt)*fx + y(15,nxt)*fy + y(18,nxt)*fz
          rcm = dsqrt(y(1,nxt)**2.d+00 + y(2,nxt)**2.d+00
         1 + y(3,nxt)**2.d+00)
          f(1,nxt) = y(4,nxt)
          f(2,nxt) = y(5,nxt)
          f(3,nxt) = y(6,nxt)
          f(4,nxt) = -gm*y(1,nxt)/rcm**3.0d+00 + thrx/mass
         1 + xa/mass
          f(5,nxt) = -gm*y(2,nxt)/rcm**3.0d+00 + thry/mass
         1 + ya/mass
          f(6,nxt) = -gm*y(3,nxt)/rcm**3.0d+00 + thrz/mass
```

```
1 + za/mass
 f(7,nxt) = ((yyi-zzi)/xxi)*y(8,nxt)*y(9,nxt)
1 + (mdot*xe*(ye*y(8,nxt)+ze*y(9,nxt)))/xxi
2 - y(7,nxt)*dixxdt/xxi
3 + 1/xxi - mdot*rho2sq*y(7,nxt)/xxi
 f(8,nxt) = ((zzi-xxi)/yyi)*y(7,nxt)*y(9,nxt)
1 - (mdot*y(8,nxt)*xe**2.0d+00)/yyi
2 - y(8,nxt)*diyydt/yyi
3 + m/yyi
4 + mfeedj/yyi
 f(9,nxt) = ((xxi-yyi)/zzi)*y(7,nxt)*y(8,nxt)
1 - (mdot*y(9,nxt)*xe**2.0d+00)/zzi
2 - y(9,nxt)*dizzdt/zzi
3 + nl/zzi
4 + mfeedk/zzi
 f(10,nxt) = y(9,nxt)*y(13,nxt) - y(8,nxt)*y(16,nxt)
 f(11,nxt) = y(9,nxt)*y(14,nxt) - y(8,nxt)*y(17,nxt)
 f(12,nxt) = y(9,nxt)*y(15,nxt) - y(8,nxt)*y(18,nxt)
 f(13,nxt) = -y(9,nxt)*y(10,nxt) + y(7,nxt)*y(16,nxt)
 f(14,nxt) = -y(9,nxt)*y(11,nxt) + y(7,nxt)*y(17,nxt)
 f(15,nxt) = -y(9,nxt)*y(12,nxt) + y(7,nxt)*y(18,nxt)
 f(16,nxt) = y(8,nxt)*y(10,nxt) - y(7,nxt)*y(13,nxt)
 f(17,nxt) = y(8,nxt)*y(11,nxt) - y(7,nxt)*y(14,nxt)
 f(18,nxt) = y(8,nxt)*y(12,nxt) - y(7,nxt)*y(15,nxt)
 return
 end
c*************************************************************
c
 subroutine mominr(nxt)
c
c*************************************************************
c
c mominr computes the moments of inertia and their first
c derivatives as a function of time.
c xxi = mom. of iner. about the body x axis in slug-ft**2
c yyi = mom. of iner. about the body y axis in slug-ft**2
c zzi = mom. of iner. about the body z axis in slug-ft**2
c dixxdt is the der. of xxi w.r.t. time in slug-ft**2/sec
c diyydt is the der. of yyi w.r.t. time in slug-ft**2/sec
c dizzdt is the der. of zzi w.r.t. time in slug-ft**2/sec
c ml is the instantaneous stage i mass in slugs
c xbar is the distance between the com and the bottom of
c      the missile in ft.
c dl is the moment arm distance for stage i in ft.
c d2 = moment arm dist. for the rest of the missile in ft.
c cd0 is the base drag coefficient (nondim)
c cdalfa is the drag coeff. variance with angle of attack
c vcm is the speed of the com in ft/sec
c flpa is the flight path angle of the missile in rad.
c alfa is the angle of attack in rad.
c cd is the drag coefficient (nondim)
c pi is not something you eat
```

57

```
c  af is the frontal area in ft**2
c  as is the missile surface area in ft*2
c  area is the effective aerodynamic area in ft**2
c  rho is the air density in slug/ft**3
c  aaa is a convenient grouping of terms used later
c  xcp is the distance of the center of pressure from the
c      base of the missile in ft.
c  d is the moment arm for the aero. moment in ft
c  xa is the aero. force in the inertial frame x-dir in lbf
c  ya is the aero. force in the inertial frame y-dir in lbf
c  za is the aero. force in the inertial frame z-dir in lbf
c  l is the aero moment around the body fr. x-axis in lbf-ft
c  m is the aero moment around the body fr. y-axis in lbf-ft
c  nl = aero. moment around the body fr. z-axis in lbf-ft
c
      common /ham/ t,y(18,4),f(18,4),errest(18),n,h,mode
      double precision t,y,f,errest,h
      common /misdat/ gm,thrust,mdot,tmass,burnt,maspro,mass,
     1 xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,gain1,
     2 gain2,gain3,gain4
      double precision gm,thrust,mdot,tmass,burnt,maspro,
     1 mass,xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,
     2 gain1,gain2,gain3,gain4
      common /miscon/ l1,l2,l3,l4,m0,m2,m3,m4,rad,r2,r3,r4,
     1 bdot,mtop,xtop,xbot,alfa,flpa
      double precision l1,l2,l3,l4,m0,m2,m3,m4,rad,r2,r3,r4,
     1 bdot,mtop,xtop,xbot,alfa,flpa
      common /aero/ xa,ya,za,l,m,nl
      double precision xa,ya,za,l,m,nl
      double precision ml,d1,d2,cd0,cdalfa,vcm,cd,
     1 pi,af,as,area,rho,aaa,xcp,d,alj
      mass = tmass-mdot*t
      if(t.gt.burnt) mass = tmass-maspro
      ml = m0 - mdot*t
      xbar = (xbot*ml + xtop*mtop)/(ml + mtop)
      d1 = xbar - xbot
      d2 = xtop - xbar
      xxi = (ml*(rad**2.d+00-bdot**2.d+00*t**2.d+00) +
     1 m2*r2**2.d+00 + m3*r3**2.d+00 + m4*r4**2.d+00)/2.d+00
      dixxdt = (-2.d+00*m0*t*bdot**2.d+00 - mdot*rad**2.d+00
     1 + 3.d+00*mdot*bdot**2.d+00*t**2.d+00)/2.d+00
      yyi = (ml*(3.d+00*rad**2.d+00 + l1**2.d+00) + m2*
     1 (3.d+00*r2**2.d+00+l2**2.d+00) + m3*(3.d+00*r3**2.d+00
     2 + l3**2.d+00) + m4*(3.d+00*r4**2.d+00 + l4**2.d+00) -
     3 3.d+00*ml*bdot**2.d+00*t**2.d+00)/12.d+00
     4 + mtop*d2**2.d+00 + ml*d1**2.d+00
      diyydt = (2.d+00*mdot/mass)*(mtop*d2*(-d1)
     1 + ml*d1**2.d+00) - (mdot/12.d+00)*(3.d+00*rad**2.d+00
     2 + l1**2.d+00) - bdot**2.d+00*m0*t/2.d+00
     3 + .75d+00*bdot**2.d+00*mdot*t**2.d+00-mdot*d1**2.d+00
      zzi = yyi
      dizzdt = diyydt
```

58

```
c
c     next, calculate the aerodynamic forces and moments
c
      cd0 = 0.3d+00
      cdalfa = 0.1d+00
      vcm = dsqrt(y(4,nxt)**2.d+00 + y(5,nxt)**2.d+00 +
     1 y(6,nxt)**2.d+00)
      flpa = dacos(y(4,nxt)/vcm)
      alj = dsqrt(y(10,nxt)**2.d+00 + y(11,nxt)**2.d+00 +
     1 y(12,nxt)**2.d+00)
      alfa = dacos((y(4,nxt)*y(10,nxt) + y(5,nxt)*y(11,nxt)
     1 + y(6,nxt)*y(12,nxt))/(vcm*alj))
      cd = cd0 + cdalfa*alfa
      pi = 3.1415926535898d+00
      af = pi*rad**2.d+00
      as = pi*(2.d+00*(rad*l1 + r2*l2 + r3*l3 + r4*l4)
     1 + rad**2.d+00)
      area = af*dcos(alfa) + as*dsin(alfa)
      zin = sngl(y(1,nxt)) - 2.0925673e+07
      call atmos(zin,dens)
      rho = dble(dens)
      aaa = -0.5d+00*cd*rho*area*vcm
      xa = aaa*y(4,nxt)
      ya = aaa*y(5,nxt)
      za = aaa*y(6,nxt)
      xcp = 28.704d+00
      d = xbar - xcp
      l = 0.0d+00
      m = d*aaa*(y(16,nxt)*y(4,nxt) + y(17,nxt)*y(5,nxt) +
     1 y(18,nxt)*y(6,nxt))
      nl = -d*aaa*(y(13,nxt)*y(4,nxt) + y(14,nxt)*y(5,nxt) +
     1 y(15,nxt)*y(6,nxt))
      return
      end
c***************************************************************
c
      subroutine atmos(zin,dout)
c
c***************************************************************
c   this subroutine is an atmospheric model that calculates
c   pressure, density, and temp at altitudes up to 300km.
c   input is altitude, zin(meters), in call statement.
c   subroutine parameters are gas const, r(joules/kg-deg-k),
c   and the S.L. values of the grav. const, g0(m/sec-sq),
c   atmospheric pressure at sea level, p0(n/m-sq),
c   and the density, d0(kg/m-cu).
c   the first order gravitational constant is b(1/m).
c   the following subscripted variables are used:
c   z(i) is the altitude (km)
c   t(i) is the molecular temperature (deg-k)
c   p(i) is the pressure (n/m-sq)
c   d(i) is the density (kg/m-cu)
```

```fortran
c   l(i) is the thermal lapse rate (deg-k/km)
      common /count/ j
      dimension z(22),t(22),p(22),d(22),l(22)
      if(j.eq.1) goto 200
      j = 1
      r = 287.0
      g0 = 9.806
      p0 = 1.01325e+05
      d0 = 1.225
      b = 3.139e-07
c   units change form ft to km
      zin = zin/3.281e+03
      data (z(i),i=1,22)/
     1   00.00,     11.019,    20.063,    32.162,    47.350,
     2   52.43,     61.590,    80.00,     90.00,     100.00,
     3  110.00,    120.00,    150.00,    160.00,    170.00,
     4  190.00,    230.00,    300.00,    400.00,    500.00,
     5  600.00,    700.00/
      data (t(i),i=1,22)/
     1   288.10,    216.65,    216.65,    228.65,    270.65,
     2   270.65,    252.65,    180.65,    180.65,    210.65,
     3   260.65,    360.65,    960.65,    1110.65,   1210.65,
     4  1350.65,   1550.65,   1830.65,   2160.65,   2420.65,
     5  2590.65,   2700.65/
      data (p(i),i=1,22)/
     1 1.000e+00,2.284e-01,5.462e-02,8.567e-03,1.095e-03,
     2 5.823e-04,1.797e-04,1.024e-05,1.622e-06,2.980e-07,
     3 7.220e-08,2.488e-08,5.000e-09,3.640e-09,2.756e-09,
     4 1.660e-09,6.869e-10,1.860e-10,3.977e-11,1.080e-11,
     5 3.400e-12, 1.176e-12/
      do 100 i = 1,22
      z(i) = z(i) * 1000
      p(i) = p(i) * p0
      d(i) = p(i)/(r*t(i))
100   continue
      do 200 i = 1,21
      l(i) = (t(i+1)-t(i))/(z(i+1)-z(i))
200   continue
      do 300 i = 1,21
      if(zin.ge.z(i+1)) go to 300
      if(abs(l(i)).lt.1.0e-05) go to 400
      q1 = 1+b*((t(i)/l(i))-z(i))
      q2 = (q1*g0)/(r*l(i))
      tout = t(i)+l(i)*(zin-z(i))
      q3 = tout/t(i)
      q4 = q3**(-q2)
      q5 = exp((b*g0*(zin-z(i)))/(r*l(i)))
      q6 = q4*q5
      pout = p(i)*q6
      q7 = q2 + 1
      dout = d(i)*(q3**(-q7))*q5/515.4
      go to 500
```

60

```
  400    tout = t(i)
         q8 = (-1.)*g0*(zin-z(i))*(1.-b/2.)*(zin+z(i))/(r*t(i))
         pout = exp(q8)*p(i)
c  units change from kg/m**3 to slug/ft**3 (density)
         dout = exp(q8)*d(i)/515.4
         go to 500
  300    continue
  500    return
         end
c**************************************************************
c
         subroutine feedbk(nxt)
c
c**************************************************************
c
c  feedbk calculates the missile's angle of attack in the
c          body frame and returns a restoring moment to rhs
c  vi = missile vel. in the body frame, x-dir. in ft/sec
c  vj = missile vel. in the body frame, y-dir. in ft/sec
c  vk = missile vel. in the body frame, z-dir. in ft/sec
c  thetaj is the angle of attack about the body frame y axis
c  thetak is the angle of attack about the body frame z axis
c  xe is the offset of mass flow from the body x axis in ft
c  ye is the offset of mass flow from the body y axis in ft
c  ze is the offset of mass flow from the body z axis in ft
c  fx is the thrust in the body frame x-dir. in lbf
c  fy is the thrust in the body frame y-dir. in lbf
c  fz is the thrust in the body frame z-dir. in lbf
c  mfeedj is the feedback mom. required to null out the AOA
c          about the body fr. y axis
c  mfeedk is the feedback mom. required to null out the AOA
c          about the body fr. z axis
c  gain1 is the damping constant for ye
c  gain2 is the gain coefficient for ye
c  gain3 is the damping constant for ze
c  gain4 is the gain coefficient for ze
c
         common /ham/ t,y(18,4),f(18,4),errest(18),n,h,mode
         double precision t,y,f,errest,h
         common /misdat/ gm,thrust,mdot,tmass,burnt,maspro,
        1 mass,xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,
        2 gain1,gain2,gain3,gain4
         double precision gm,thrust,mdot,tmass,burnt,maspro,
        1 mass,xxi,yyi,zzi,dixxdt,diyydt,dizzdt,rho2sq,xbar,
        2 gain1,gain2,gain3,gain4
         common /feedbk/ mfeedj,mfeedk,xe,ye,ze,fx,fy,fz,
        1 thetaj,thetak
         double precision mfeedj,mfeedk,xe,ye,ze,fx,fy,fz,
        1 thetaj,thetak
       · double precision vi,vj,vk
c
         xe = -xbar
```

61

```fortran
      vi = y(10,nxt)*y(4,nxt) + y(11,nxt)*y(5,nxt)
     1 + y(12,nxt)*y(6,nxt)
      vj = y(13,nxt)*y(4,nxt) + y(14,nxt)*y(5,nxt)
     1 + y(15,nxt)*y(6,nxt)
      vk = y(16,nxt)*y(4,nxt) + y(17,nxt)*y(5,nxt)
     1 + y(18,nxt)*y(6,nxt)
      thetaj = datan(vk/vi)
      thetak = datan(vj/vi)
      ye = gain1*y(9,nxt) + gain2*thetak
      ze = gain3*y(8,nxt) + gain4*thetaj
      if(dabs(ye/xe).ge.0.8391d+00)ye = dsign(17.619d+00,ye)
      if(dabs(ze/xe).ge.0.8391d+00)ze = dsign(17.619d+00,ze)
      fy = thrust*ye/dsqrt(xe**2.d+00+ye**2.d+00+ze**2.d+00)
      fz = thrust*ze/dsqrt(xe**2.d+00+ye**2.d+00+ze**2.d+00)
      fx = dsqrt(thrust**2.d+00 - fy**2.d+00 - fz**2.d+00)
      mfeedj = ze*fx
      mfeedk = -ye*fx
      return
      end
```

62

Appendix B

A sample of the input file used to support program misspin is listed below. This file is referred to in line 42 of misspin as 'input.dat2' and contains all the inputs required to run the program.

```
          t0                    tf
   0.0000000000000d+00 0.6100000000000d+02


          x                     y                     z
   2.0925722570000d+07 0.0000000000000d+00 0.0000000000000d+00


          u                     v                     w
   5.0000000000000d+01 0.0000000000000d+00 0.0000000000000d+00


          p                     q                     r
   1.5707963705063d+00 0.0100000000000d+00 0.0100000000000d+00


         a11                   a12                   a13
   0.1500000000000d+02 0.9000000000000d+02 0.7500000000000d+02


         a21                   a22                   a23
   0.9000000000000d+02 0.0000000000000d+00 0.9000000000000d+02


         a31                   a32                   a33
   0.7500000000000d+02 0.9000000000000d+02 0.1500000000000d+02

 mode nstp  iplot
 +0000+15000-    1

        tmass                   rad                  tlen
   2.3621558000000d+03 2.7500000000000d+00 5.9900000000000d+01


       thrust                  mdot                  burnt
   2.0200000000000d+05 2.3351510000000d+01 6.1000000000000d+01


         l1                    l2                    l3
   2.5300000000000d+01 1.3100000000000d+01 7.1000000000000d+00


         l4                    m0                    m2
   1.4400000000000d+01 1.5711444000000d+03 4.8237707000000d+02


         m3                    m4                  maspro
   2.5331013000000d+02 5.5324175000000d+01 1.4244421000000d+03
```

63

```
         r2                      r3                      r4
    2.15000000000000d+00 2.15000000000000d+00 2.15000000000000d+00

         gain1                   gain2
    -0.90000000000000d+00-1.00000000000000d+01

         gain3                   gain4
    -0.90000000000000d+00-1.00000000000000d+01
```

Appendix C

A sample of the output file produced by program misspin
is listed below. This file is referred to in line 46 of
misspin as 'output.dat2' and contains the output from a
standard run of the program. A large number of iterations has
been deleted between t = 2.0 and t = 60.0.

Dynamics Propagator

to, tf: 0.0000000000000e+00 0.6100000000000e+02

Initial State Vector
 0.2092572257000e+08 0.0000000000000e+00 0.0000000000000e+00
 0.5000000000000e+02 0.0000000000000e+00 0.0000000000000e+00
 0.1570796370506e+01 0.1000000000000e-01 0.1000000000000e-01

 0.9659258262891e+00 0.0000000000000e+00-0.2588190451025e+00
 0.0000000000000e+00 0.1000000000000e+01 0.0000000000000e+00
 0.2588190451025e+00 0.0000000000000e+00 0.9659258262891e+00
mode, stps: 0 15000

tmass = 0.236215580e+04 rad = 0.275000000e+01
tlen = 0.599000000e+02 thrust = 0.202000000e+06
mdot = 0.233515100e+02 burnt = 0.610000000e+02

```
l1 = 0.253000000e+02 l2 = 0.131000000e+02

l3 = 0.710000000e+01 l4 = 0.144000000e+02

m0 = 0.157114440e+04 m2 = 0.482377070e+03

m3 = 0.253310130e+03 m4 = 0.553241750e+02

maspro = 0.142444210e+04 r2 = 0.215000000e+01

r3 = 0.215000000e+01 r4 = 0.215000000e+01

gain1 =-0.900000000e+00 gain2 =-0.100000000e+02

gain3 =-0.900000000e+00 gain4 =-0.100000000e+02

bf = 0.2670000000000e+01 bdot = 0.4377049180328e-01

mtop = 0.7910113750000e+03 xbot = 0.1265000000000e+02

xtop = 0.3654265231710e+02 rho2sq = 0.3781250000000e+01

t = 0.250000000e+00

alfa = 0.1183984109802e+00 flpa = 0.1135910610128e+00

thetaj = 0.110650524e+00 thetak = 0.424722783e-01

t = 0.500000000e+00

alfa = 0.1349027162445e-01 flpa = 0.1576384415400e+00

thetaj = 0.134869061e-01 thetak =-0.301353252e-03

t = 0.750000000e+00

alfa = 0.6806496272313e-01 flpa = 0.1633206205755e+00

thetaj =-0.949918648e-02 thetak =-0.674028930e-01

t = 0.100000000e+01

alfa = 0.1080837271381e+00 flpa = 0.1492277787722e+00

thetaj = 0.193839957e-01 thetak =-0.106357830e+00

t = 0.125000000e+01

alfa = 0.1119415232809e+00 flpa = 0.1282593230884e+00

thetaj = 0.563485640e-01 thetak =-0.969292474e-01
```

```
t = 0.150000000e+01

alfa = 0.8545213815704e-01 flpa = 0.1105389365479e+00

thetaj = 0.657428036e-01 thetak =-0.547463508e-01

t = 0.175000000e+01

alfa = 0.4292624643032e-01 flpa = 0.1021869574191e+00

thetaj = 0.407005270e-01 thetak =-0.136580155e-01

t = 0.200000000e+01

alfa = 0.1116573412999e-02 flpa = 0.1040975924051e+00

thetaj = 0.827320623e-03 thetak = 0.749851508e-03

*** portion of output deleted for the sake of brevity ***

t = 0.600000000e+02

alfa = 0.1183470727665e-02 flpa = 0.5760804791962e+00

thetaj =-0.487526051e-03 thetak =-0.107838847e-02

t = 0.602500000e+02

alfa = 0.1627084792226e-02 flpa = 0.5767944135272e+00

thetaj =-0.149348123e-02 thetak = 0.645693414e-03

t = 0.605000000e+02

alfa = 0.2458572367868e-02 flpa = 0.5774826214918e+00

thetaj =-0.170839995e-02 thetak = 0.1768038441e-02

t = 0.607500000e+02

alfa = 0.1542471668112e-02 flpa = 0.5781562255459e+00

thetaj =-0.816599337e-03 thetak = 0.130858163e-02

t = 0.610000000e+02

alfa = 0.7942141257230e-03 flpa = 0.5788422181625e+00

thetaj = 0.727413518e-03 thetak =-0.318819283e-03
```

67

at time tf:

state vector is

 0.2106036754946e+08

 0.1041772883904e+05

-0.7271338505948e+05

 0.5246253808848e+04

 0.4870780460679e+03

-0.3393730020379e+04

 0.1516615939657e+01

 0.7507226603557e-02

 0.9808649397234e-02

 0.8366785173822e+00

 0.7759364221237e-01

-0.5421699781792e+00

-0.3489327010687e+00

 0.8385295590313e+00

-0.4184664248041e+00

 0.4221552186846e+00

 0.5393027028031e+00

 0.7286546273920e+00

BIBLIOGRAPHY

1. Aerophysics Laboratory. Dispersion of Spinning Missiles Due to Lift Nonaveraging. Report SAMSO-TR-76-135. Laboratory Operations, The Aerospace Corporation, El Segundo CA, June 1976 (AD-A027 582).

2. Hill, Dr. James L. Missile Response Analysis. TR-RL-CR-82-4. U.S. Army Missile Command, Redstone Arsenal, AL, April 1982 (AD-B065 755).

3. Reilly, M.H. Equations of Powered Rocket Ascent and Orbit Trajectory. NRL Report 8237. Communications Sciences Division, Naval Research Laboratory, Washington, D.C. May 1979 (AD-A069 296).

4. Green, G.S. and R.N.A. Plimmer. A Description of the Mathematical Modelling of the ELDO Vehicle (Revised). TR 74070. Royal Aircraft Establishment, May 1974 (AD-B000 566).

5. Powell, F. RAKOT 3D/6D Trajectory Simulation Program. SAMSO-TR-73-234. Engineering Science Operations, The Aerospace Corporation, El Segundo CA, August 1973 (AD 912 637).

6. Regan, Frank J. Re-Entry Vehicle Dynamics. New York: AIAA, Inc, 1984.

7. Thomson, William T. Introduction to Space Dynamics. New York: John Wiley & Sons, Inc, 1963.

8. Cornelisse, J.W. and others. Rocket Propulsion and Spaceflight Dynamics. London: Pitman, 1979.

9. Wiesel, William E. Jr, Lecture Materials Distributed in MC 6.36, School of Engineering, Air Force Institute of Technology, Wright Patterson AFB, OH, 1985.

10. Kreifels, T. 2Lt, Ballistic Missile Trajectory Engineer. Personal Correspondence. 544 SIW/DIJ, SAC HQ, Offutt AFB NE, 1 Sept 1985.

11. Miele, Angelo, Flight Mechanics Volume 1 Theory of Flight Paths. Reading, Mass.:Addison-Wesley Publishing Company, Inc, 1962.

12. Krasnov, N.F. Aerodynamics of Bodies of Revolution. New York: American Elsevier Publishing Company, Inc, 1970.

13. Wiesel, William E. Jr, Lecture Materials Distributed in MC 5.33, School of Engineering, Air Force Institute of Technology, Wright Patterson AFB, OH, 1985.

14. Wiesel, William E. Jr, Professor Air Force Institute of Technology, Wright Patterson AFB, OH   Interview 20 Sept 1985.

15. D'Azzo, John J. and Constantine H. Houpis. Linear Control Systems Analysis and Design (Second Edition). New York: McGraw-Hill Book Company, 1981.

VITA

Captain Robert W. Bandstra was born on April 15, 1956 in Slayton, Minnesota. He graduated from Southwest Minnesota Christian High School in Edgerton, Minn. in 1974. He enlisted in the Air Force in June of 1976. While enlisted he worked as an Automatic Flight Control Systems Specialist at Dover AFB, DE. In December, 1980 he received his Bachelor of Science degree from Texas A&M University in Aerospace Engineering through the Airmen's Education and Commissioning Program. He received his commission from Officer's Training School in March of 1981. He then worked as a Ballistic Missile Trajectory Engineer at Offutt AFB, NE until entering the School of Engineering, Air Force Institute of Technology, in June of 1984.

<div style="text-align: center">

Permanent address: Box 192

Chandler, Minnesota 56122

</div>

_AD-A164 077_

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GA/AA/85D-01	5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENY	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433		7b. ADDRESS (City, State and ZIP Code)
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 NOS.			
	PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19				

12. PERSONAL AUTHOR(S)
Robert W. Bandstra, Capt, USAF

13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 December	15. PAGE COUNT 78

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	SPINNING (MOTION), ASCENT TRAJECTORIES, ROCKET TRAJECTORIES
16	02		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: STABILITY OF SPINNING ICBM IN FIRST STAGE BOOST PHASE

Thesis Chairman: Dr. William E. Wiesel, Jr.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. William E. Wiesel, Jr., Professor	22b. TELEPHONE NUMBER (Include Area Code) 513-255-3517	22c. OFFICE SYMBOL AFIT/ENY

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

A computer program is developed to model a spinning Intercontinental Ballistic Missile (ICBM) during the first stage boost phase. The equations of motion are derived and presented and a full rotation matrix is used to show the relationship between a launch-centered, nonrotating earth, inertial reference frame and the missile body reference frame. The moments of inertia and aerodynamic forces are derived and presented. A feedback controller is derived which proved to be a necessary addition to the system in order to reduce the angle of attack. The angle of attack of the missile produced adverse effects on the burnout vector without the feedback controller but the effects are reduced considerably with the controller included. Problem areas include possibly excessive nozzle gimbal rates caused by the feedback controller and the need to change the initial kick angle if the missile is spinning in order to achieve the same burnout conditions as a nonspinning missile.

END

FILMED

3-86

DTIC