

DEVELOPMENT OF A LQR CONTROLLER TO PROVIDE A
GUIDANCE GAIN SCHEDULE FOR MARS LANDER 2001

Brande Tuck

The George Washington University/Joint
Institute for the Advancement of Flight Sciences

August 28, 1998

Table of Contents

Abstract	iii
Nomenclature	iv
List of Figures	vi
Introduction	1
Linearization of Guidance Equations of Motion	2
Linear Quadratic Regulator	5
Gain Scheduling	7
Nonlinear Validation	9
Conclusions	11
References	21
Appendix A: Matlab Programs	22

Abstract

Unlike the Mars Pathfinder, the Mars 2001 Lander will follow a specific trajectory using course correction capabilities. To correct the path of the Lander, reaction control jets will roll the vehicle to rotate the lift vector and, subsequently, change the heading angle. A full feedback state space linearization was performed on the guidance equations of motion. The state and control matrices can only be assumed to be constant for a short time due to the large changes in altitude, velocity, and dynamic pressure from the point of atmospheric entry to the landing site. The trajectory has been divided into breakpoints to calculate different matrices for different flight regimes. To follow the landing trajectory, a linear quadratic regulator was used to find a gain matrix for each set point. These gain matrices were combined to form a gain scheduling matrix that provided dynamic stability for the Mars 2001 Lander along the guidance trajectory.

Nomenclature

A	state, or plant, matrix
B	control matrix
C_{D_α}	linear coefficient of drag
$C_{D_{\alpha^2}}$	quadratic coefficient of drag
C_{L_α}	coefficient of lift
D	drag
deg	degree
EDL	Entry, Descent, and Landing phase
H	altitude, height above surface of Mars
\dot{H}	derivative of altitude
km	kilometers
L	lift
LQR	Linear Quadratic Regulator
m	mass of Mars Lander
Q	state weighting matrix
q	dynamic pressure
R	control weighting matrix
Rm	radius of Mars
S	reference area on Lander
sec	second(s)
V	velocity

\dot{V}	derivative of velocity
x	state vector
\dot{x}	derivative of state vector
$'01$	2001
α	angle of attack
γ	flightpath angle
$\dot{\gamma}$	derivative of flightpath angle
μ	gravitational constant
ϕ	roll angle
ψ	heading angle
$\dot{\psi}$	derivative of heading angle

List of Figures

Figure 1: EDL Sequence of Events	12
Figure 2: Simulink Model of Control Law	13
Figure 3: Response of the Altitude State Variable to a Step Heading Angle Command	14
Figure 4: Response of the Velocity State Variable to a Step Heading Angle Command	15
Figure 5: Response of the Flightpath Angle State Variable to a Step Heading Angle Command	16
Figure 6: Response of the Heading Angle State Variable to a Step Heading Angle Command	17
Figure 7: Response of Angle of Attack Control Variable to a Step Heading Angle Command	18
Figure 8: Response of Roll Angle Control Variable to a Step Heading Angle Command	19
Figure 9: Comparison of EOM and Data for Height	20
Figure 10: Comparison of EOM and Data for Velocity	21
Figure 11: Comparison of EOM and Data for Flightpath Angle	22
Figure 12: Heading Angle found from Integration of EOM	23
Figure 13: Comparison of Nominal Trajectory and Gain Controlled EOM for Height	24
Figure 14: Comparison of Nominal Trajectory and Gain Controlled EOM for Velocity	25
Figure 15: Comparison of Nominal Trajectory and Gain Controlled EOM for Flightpath Angle	26
Figure 16: Comparison of Nominal Trajectory and Gain Controlled EOM for Heading Angle	27
Table 1: Gain Calculated from LQR for each Breakpoint	28


Introduction

The Mars 2001 Lander will follow a specific trajectory upon entering the atmosphere of Mars. Unlike the Mars Pathfinder mission which had a landing ellipse of 100 km by 200 km, the '01 Lander is required have a landing circle with a 50 km radius and a 10 km radius is desired. This small target area requires the Lander to follow a precise guidance trajectory. The entry, descent, and landing phase begins one day before the spacecraft arrives at Mars and ends when the Lander has touched down on the surface. The sequence of events for EDL is given in Figure 1. The vehicle enters the atmosphere at an altitude of 125 km and is protected by the aeroshell and the backshell. The aeroshell is between the surface and the Lander while the backshell is between the Lander and space. The Lander does not have traditional aerosurfaces, such as elevators or ailerons, to control the vehicle. Instead, it has control jets that roll the vehicle. To change the direction of the Lander, the jets roll the vehicle to rotate the lift vector and, subsequently, change the heading angle. By varying the control system, a predetermined attitude trajectory is followed. When the altitude is about 10 km and the Mach number is less than two, the parachute is deployed and the aeroshell is released. The touchdown engines are started at an altitude of 1.5 km and the parachute and backshell are released. The touchdown engines will provide the Lander with a soft controlled landing. [Ref. 1] For the Lander to touchdown within the designated area, the vehicle must be able to follow the guidance trajectory. Several universities and NASA centers are developing guidance algorithms. These algorithms include predictor-correctors, drag reference followers, and nominal trajectory followers. The best of these various methods will be chosen as the guidance system for the Mars 2001 Lander. This paper deals with one possible method to stabilize and guide the Lander along the guidance trajectory.

Linearization of Guidance Equations of Motion

The equations of motion for the Mars 2001 Lander describe the height, velocity, flightpath angle, and heading angle of the vehicle along the guidance trajectory. These four variables form the state variables while the angle of attack and the roll angle form the control variables. The equations of motion are:

$$\begin{aligned}
 \dot{H} &= V \sin \gamma \\
 \dot{V} &= -\frac{D}{m} - \frac{\mu \sin \gamma}{(R_e + H)^2} \\
 \dot{\gamma} &= \frac{L \cos \phi}{mV} + \frac{V \cos \gamma}{(R_e + H)} - \frac{\mu \cos \gamma}{V(R_e + H)^2} \\
 \dot{\psi} &= \frac{L \sin \phi}{mV \cos \gamma}
 \end{aligned}$$



Eq 1

$$L = \frac{1}{2} \rho V^2 S C_{L\alpha} \alpha$$

where:

$$D = \frac{1}{2} \rho V^2 S (C_{D_0} + C_{D_\alpha} \alpha + C_{D_{2\alpha}} \alpha^2)$$

Eq 2

$$\rho = \rho_0 e^{-bH}$$

[Ref 2 & 3] These equations are highly nonlinear so they were linearized with a state space realization. However, this realization could neither produce one constant state matrix nor one constant control matrix. The state and control matrices could only be assumed to be constant for a short time along the trajectory path due to the large changes in altitude, velocity, and dynamic pressure from the point of atmospheric entry to the landing site. The trajectory was divided into break points to allow for different matrices for the different flight regimes.

An analytic approach was used to linearize the equations of motion. Each equation of motion was expanded in a Taylor series for each state variable. The linearization was accomplished by disregarding the higher order terms. To form the A matrix corresponding to the state equation: $\dot{x} = Ax + Bu$, the partial derivative of each equation of motion was found with respect to each state variable. The B matrix was formed from the partial derivative of the equations of motion with respect to the control variables. For clarity, the derivation of the state and control matrix corresponding to the height equation of motion is shown here.

$$\frac{\partial \dot{H}}{\partial H} = 0 \quad \frac{\partial \dot{H}}{\partial V} = \sin \gamma \quad \frac{\partial \dot{H}}{\partial \gamma} = V \cos \gamma \quad \frac{\partial \dot{H}}{\partial \psi} = 0 \quad \frac{\partial \dot{H}}{\partial \alpha} = 0 \quad \frac{\partial \dot{H}}{\partial \phi} = 0$$

$$[\dot{H}] = [0 \quad \sin \gamma \quad V \cos \gamma \quad 0] \begin{bmatrix} H \\ V \\ \gamma \\ \psi \end{bmatrix} + [0 \quad 0] \begin{bmatrix} \alpha \\ \phi \end{bmatrix} \quad \text{Eq 3}$$

The complete A and B matrices were found in a similar manner for each equation of motion. The final state and control matrices are:

$$[A] = \begin{bmatrix} 0 & \sin \gamma & V \cos \gamma & 0 \\ \frac{D}{m} + \frac{2\mu \sin \gamma}{(Rm+H)^3} & \frac{-2D}{mV} & \frac{V \cos \gamma}{(Rm+H)^2} & 0 \\ \frac{-L \cos \phi}{mV} - \frac{V \cos \gamma}{(Rm+H)^2} + \frac{2\mu \cos \gamma}{(Rm+H)^3 V} & \frac{-L \cos \phi}{mV^2} + \frac{\cos \gamma}{(Rm+H)} + \frac{\mu \cos \gamma}{(Rm+H)^2 V^2} & \frac{-V \sin \gamma}{(Rm+H)} + \frac{\mu \sin \gamma}{(Rm+H)^2 V} & 0 \\ \frac{-L \sin \phi}{mV \cos \gamma} & \frac{L \sin \phi}{mV^2 \cos \gamma} & \frac{L \sin \phi \sin \gamma}{mV \cos^2 \gamma} & 0 \end{bmatrix} \quad \text{Eq 4}$$

$$[B] = \begin{bmatrix} 0 & 0 \\ -qS(C_{D_{\alpha}} + 2C_{D_{\alpha^2}} \alpha) & 0 \\ \frac{m}{L \cos \phi} & \frac{-L \sin \phi}{mV} \\ \frac{mV \alpha}{L \sin \phi} & \frac{L \cos \phi}{mV \cos \gamma} \\ \frac{mV \cos \gamma \alpha}{mV \cos \gamma} & \frac{mV}{L \cos \phi} \end{bmatrix} \quad \text{Eq 5}$$

where lift and drag are functions of angle of attack and dynamic pressure, which is a function of altitude and velocity.

Due to the nature of the equations, this was an improper realization. While the heading angle was one of the outputs being controlled, the actual angle itself did not appear in any of the equations of motion. This gave the zero column in the state matrix. This matrix provided neutral stability because it had eigenvalues on the imaginary axis. While the ill-conditioned plant suggests a possibly uncontrollable system, a controllability check of the A and B matrices show that the controllability matrix had full rank which guaranteed a controllable system. This linearization was used at breakpoints every 20 seconds to calculate the new plant and control matrices. Since the breakpoints were 20 seconds apart, a controller that would provide stability in less than 20 seconds was initially desired. However, this led to unrealistically high gains at the breakpoints that occurred between the Mach numbers of 32 and 18. The stability requirement was then relaxed to 100 seconds. Since this stability requirement was an artificial constraint, changing it did not adversely affect the design.

Linear Quadratic Regulator

To provide stability at each matrix along the guidance trajectory, a linear quadratic regulator was used to find the gain K . The objective of the regulator was to drive any initial condition error to zero, thus guaranteeing stability. This was achieved by selecting the control input to minimize a quadratic cost function of the type:

$$J = \frac{1}{2} \int_0^{\infty} (x^T Q x + u^T R u) dt \quad \text{Eq 6}$$

where Q was a symmetric positive semidefinite weighting matrix and R was a positive definite weighting matrix. The Q matrix weighted the state while the R matrix weighted the control. The relative magnitudes of Q and R could be selected to trade off requirements on the smallness of the state against requirements on the smallness of the input. To make the state go to zero more quickly, a larger Q was used. By using full state feedback, it is possible to guarantee the stability of the closed loop system using the optimal LQR state feedback gain. The theorem for this is presented here:

Theorem: Let H be any matrix so that $Q = H^T H$. Suppose that (H, A) is detectable. Then (A, B) is controllable if and only if:

- (a) There exists a unique positive semidefinite solution to the Riccati equation, and
- (b) The closed-loop system is asymptotically stable if the Kalman gain K is computed in terms of this positive semidefinite solution P .

The algebraic Riccati equation is: $0 = A^T P + PA + Q - PBR^{-1}B^T P$ Eq 7

and the Kalman Gain is: $K = R^{-1}B^T P$. Eq 8

This theorem stated that as long as (H,A) is observable and (A,B) is controllable, then the linear quadratic regulator did guarantee a stable closed loop system for the full state feedback gain. [Ref 3] This theorem led to a straightforward method to find a state-variable feedback gain that would stabilize any controllable plant.

Gain Scheduling

A LQR calculated the Kalman gain at each breakpoint to guarantee stability of the system. In addition to stability, the controller needed to return the states to their steady state value in 100 sec. The weighting matrices, Q and R, were varied for each breakpoint until the system met the design requirements. Figure 2 shows a Simulink model of the full state feedback control loop. This model was used to analyze each gain until a satisfactory K was found at any given breakpoint. The gain was designed to follow a simple step input as the heading angle command.

The Matlab programs generated to calculate the gains are included in Appendix A. The program Refdata.m provides reference data for the Mars 2001 Lander. Data.m calculates aerodynamic coefficients at each breakpoint. Similarly, AB.m calculates the linearized state and control matrices at each breakpoint. Finally, the Kalman gain for each breakpoint was found by varying Q and R in Marscon.m.

For simplicity, the exact results for only one breakpoint will be discussed. This breakpoint occurred at a Mach number of 32.5. The Kalman gain found from LQR was compared to a unity feedback gain. The system using the Kalman gain is referred to as the compensated system, while the system using only a unity feedback gain is referred to as the uncompensated system. The comparison for each state and control variable is given in Figures 3 through 8. The response of the height variable is shown on Figure 3, velocity is on Figure 4, flightpath angle is on Figure 5, and the heading angle is on Figure 6. The angle of attack and roll angle are shown on Figures 7 and 8, respectfully. For the system with the K gain, all of the steady state errors returned to their final values by 100 seconds. The eigenvalues for the compensated system were all negative, so the system is stable. One of the eigenvalues is very close to zero. This acts as a pole at the origin, so most of the steady state errors go to zero, approximately. The steady state errors of the

uncompensated system did not return to zero in 100 sec. The response of the uncompensated system is oscillatory and poorly damped. It takes 500 sec for the steady state errors to return to zero. The resulting gain calculated by the linear quadratic regulator for this Mach number was:

$$K = \begin{bmatrix} 9.99 & -0.98 & 5.68e^4 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.0e^2 \end{bmatrix} \quad \text{Eq 9}$$

In a similar manner, gains were found for each breakpoint. They were then combined into one matrix that provided gain scheduling along the trajectory path. For most of the breakpoints, it took more than 20 sec for the gain to provide stability. While it was possible to achieve stability in 20 seconds for each point, it often resulted in an impractical gain matrix. To have both stability and practicality, the 20 sec requirement was relaxed to 100 sec. All of the gain matrices provided stability by 100 sec. However, in the region corresponding to Mach numbers 32.5 to 18, the calculated gain matrices still possessed relatively high values for some components. This is shown in Eq 9, the component that relates the flightpath angle to the angle of attack is large. The complete gain scheduling matrix is given on Table 1.

Nonlinear Validation

In order to validate the gain scheduling matrix, a nonlinear simulation was constructed. The first part of this simulation was the creation of a nominal trajectory. The state and control variable values used in the linear simulation were created for the Pathfinder mission. The equations of motion needed to correspond to this data. The equations of motion were numerically integrated and compared to the data. The control variables, the angle of attack and the roll angle, were interpolated from a table of Pathfinder data. The comparison of the data and the integrated equation of motion are shown in Figures 9, 10, & 11. Figure 9 corresponds to the height, Figure 10 to the velocity, and Figure 11 to the flightpath angle, respectively. The height, velocity, and flightpath angle all agree with the Pathfinder data. This verifies that the equations of motion used to form the gain matrices accurately describe the spacecraft motion. Since no data exists for the heading angle, a comparison plot could not be made. However, the heading angle found from the integrated equations of motion is shown on Figure 12. The initial value of the heading angle was assumed to be 180 deg, which corresponds to both the Pathfinder yaw angle and the bank angle.

The next step was to integrate the equations of motion, using the gain scheduling matrix. To do this, the gain matrix at each breakpoint was used to calculate the control variables. These values were then used in the integration of the equations of motion. The initial values for the height, velocity, flightpath angle and heading angle were taken from the nominal trajectory. Figures 13-16 show the state variables found from this integration. These plots do not compare well with the nominal trajectory or the Pathfinder data. Initially, all states follow the nominal trajectory. When the time reaches 80 sec, the heading angle and the flightpath angle begin to diverge from their nominal trajectories. This affects the height and velocity, and they no longer follow their respective nominal trajectories. There could be several reasons for this divergence.

The states begin to diverge at a Mach number of 32.5. This is the beginning of the region where it was difficult to calculate reasonable gains. The linearization of the equations of motion might prevent the linear quadratic regulator from creating a usable gain matrix. Since the linearization neglected all of the higher order dynamic terms, this would be a valid reason for the deficiency of the gain schedule matrix. The modeling of the linear system might be another possible reason for the difference between the nominal trajectory and the trajectory that used the gain scheduling matrix. In this model, the heading command was a simple step input. The gain at each breakpoint was calculated to drive the steady state errors of the height, velocity, flightpath angle, roll angle and angle of attack to zero by 100 sec. The gain also forced the heading angle to achieve and to maintain a steady state of 1 deg. The actual heading angle given by the integration of the equations of motion was shown on Figure 12. The heading angle oscillates between 176 and 184 degrees. In the future, the simulation needs to be modified to command the heading angle trajectory given by the equations of motion.

Conclusions

The gain scheduling matrix provided stability for the Lander along the guidance trajectory path. By using a linear quadratic regulator to calculate the gain scheduling matrix, the matrix was guaranteed to provide stability. Varying the weighting matrices for each breakpoint created a gain matrix that provided a timely response. In the given example where the breakpoint corresponded to a Mach number of 32.5, the compensated system had nearly zero steady state errors by at least 100 sec. Some of the states reached a zero error in less than 30 sec. Using a linear quadratic regulator to form a gain scheduling matrix did create a dynamically stable system. The poles at each breakpoint were negative which guarantees dynamic stability. Although, the original design constraint was relaxed to calculate more realistic gains, this model did not use any real world constraints, e.g. fuel consumption. So while the linear quadratic regulator can theoretically be used to develop a stable controller that follows the guidance trajectory, it might not be practical in the real world. However, the purpose of this study was to linearize the guidance equations of motion and to develop a dynamically stable control law. While the gain scheduling matrix does provide the Mars 2001 Lander with dynamic stability, the matrix does not allow the Lander to accurately follow the nominal trajectory. In the future, the simulation should be altered to include the real heading angle command instead of a step command. This will allow the creation of a gain scheduling matrix that will provide stability and will follow the nominal trajectory.

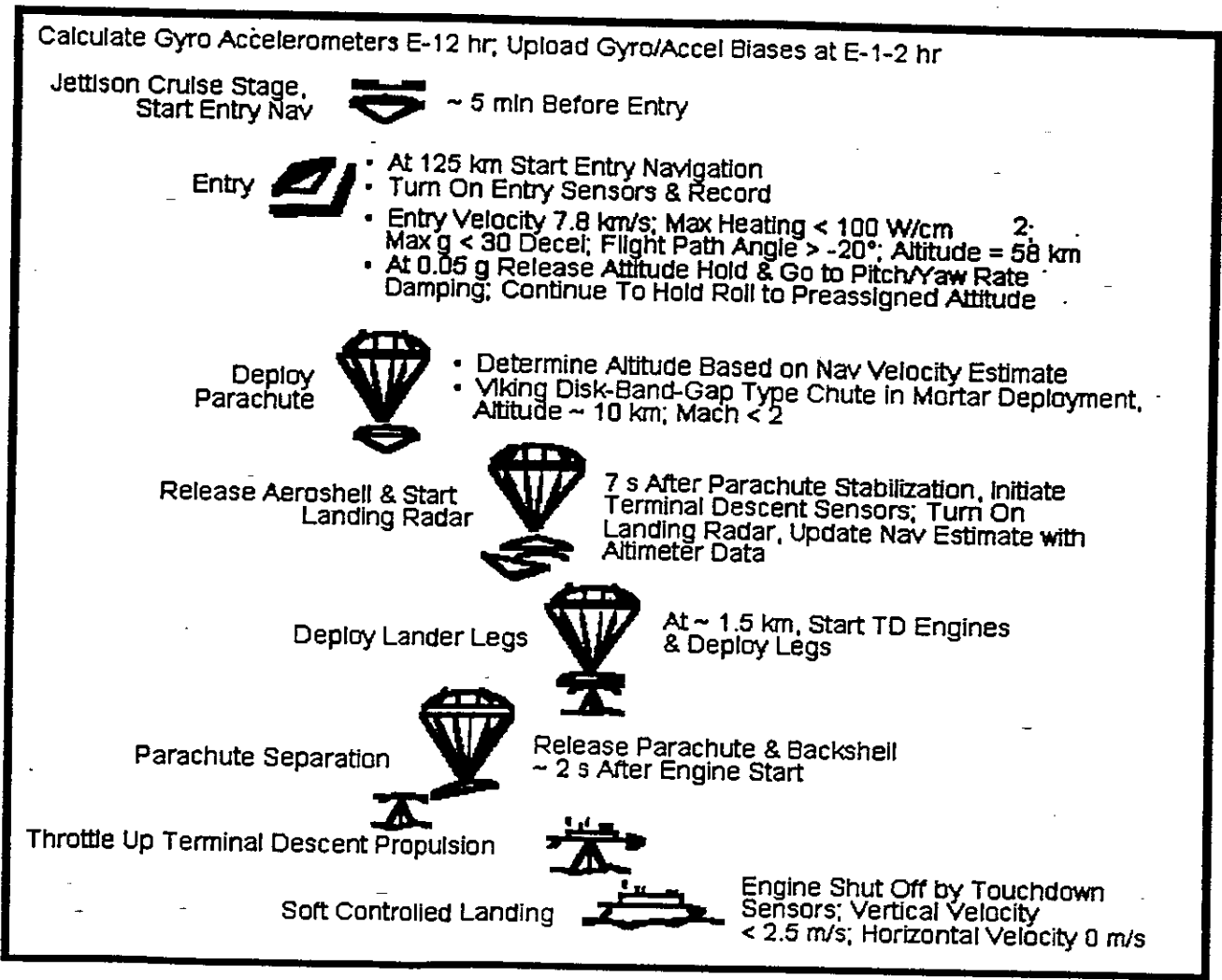


Figure 1: EDL Sequence of Events

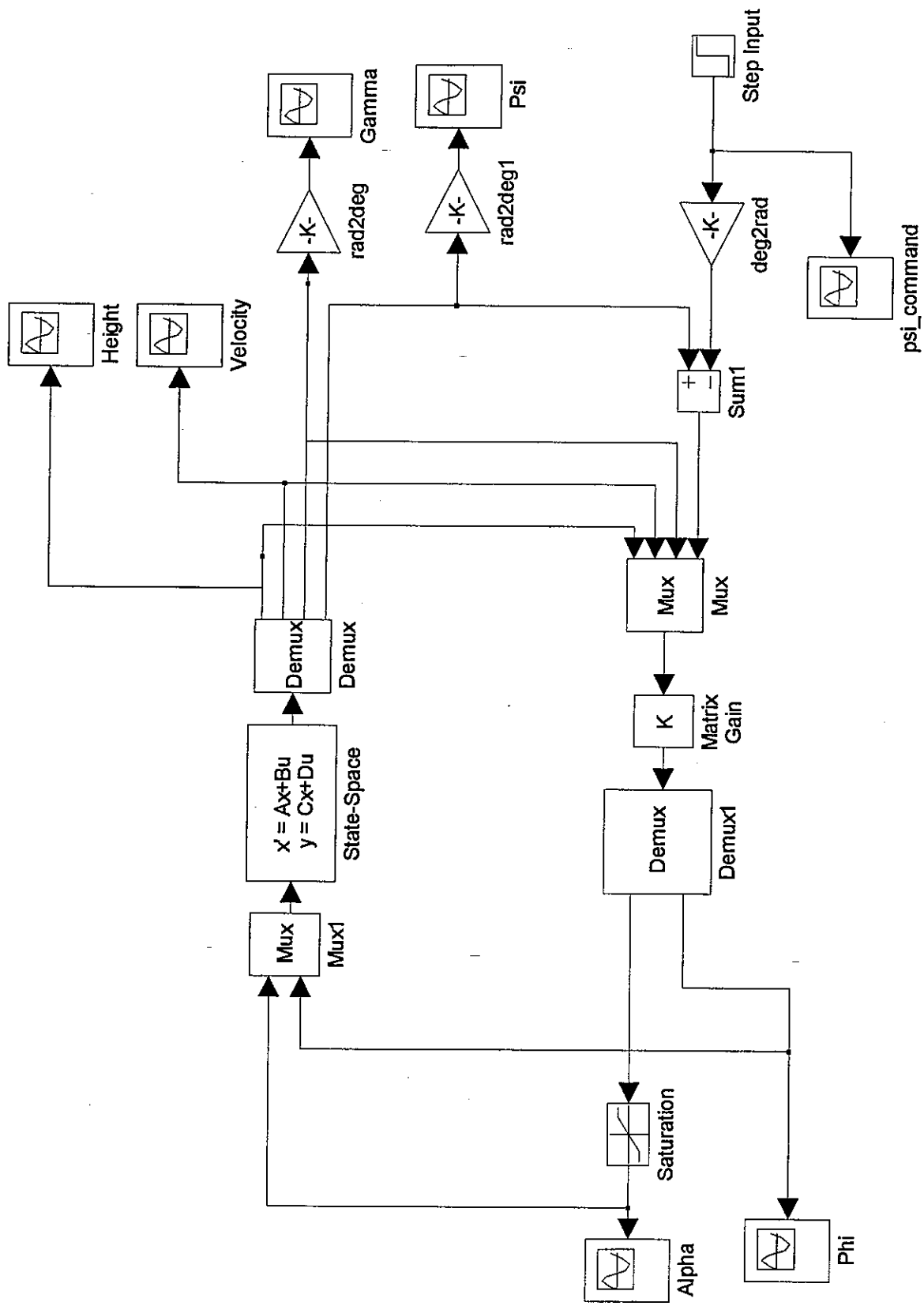


Figure 2: Simulink Model of Linearized System

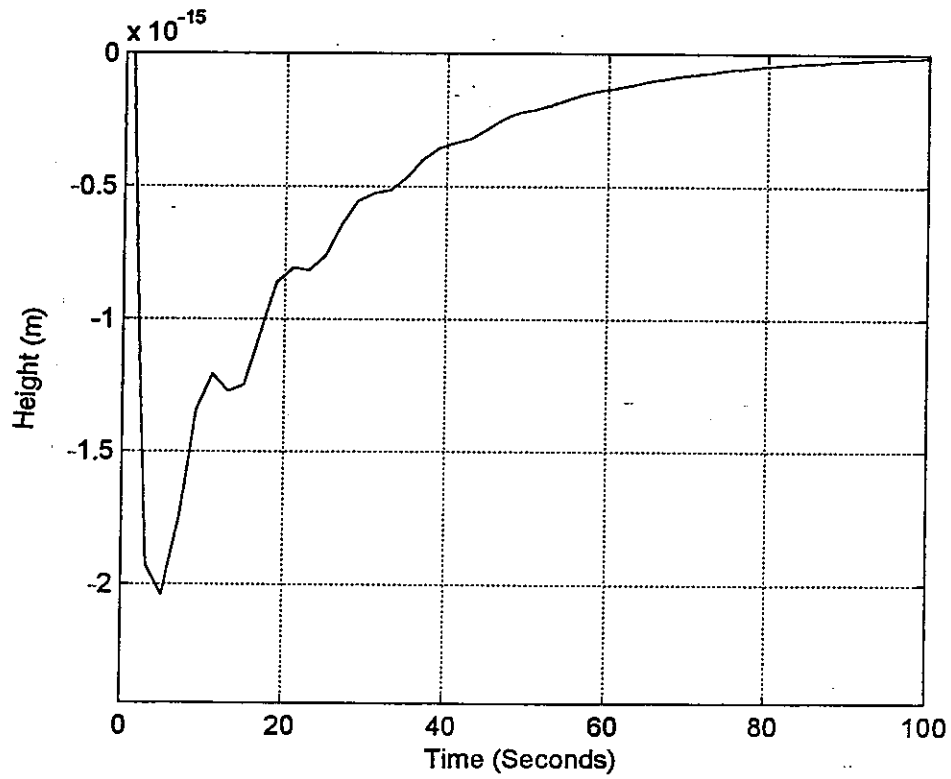
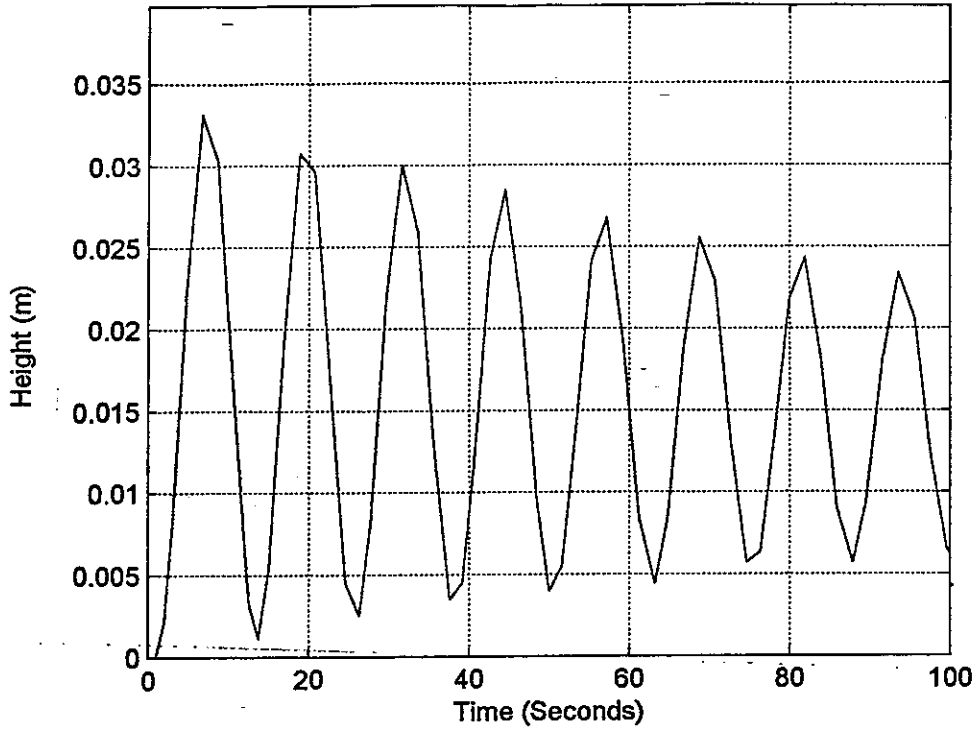


Figure 3: Response of the Altitude State Variable to a Step Heading Angle Command

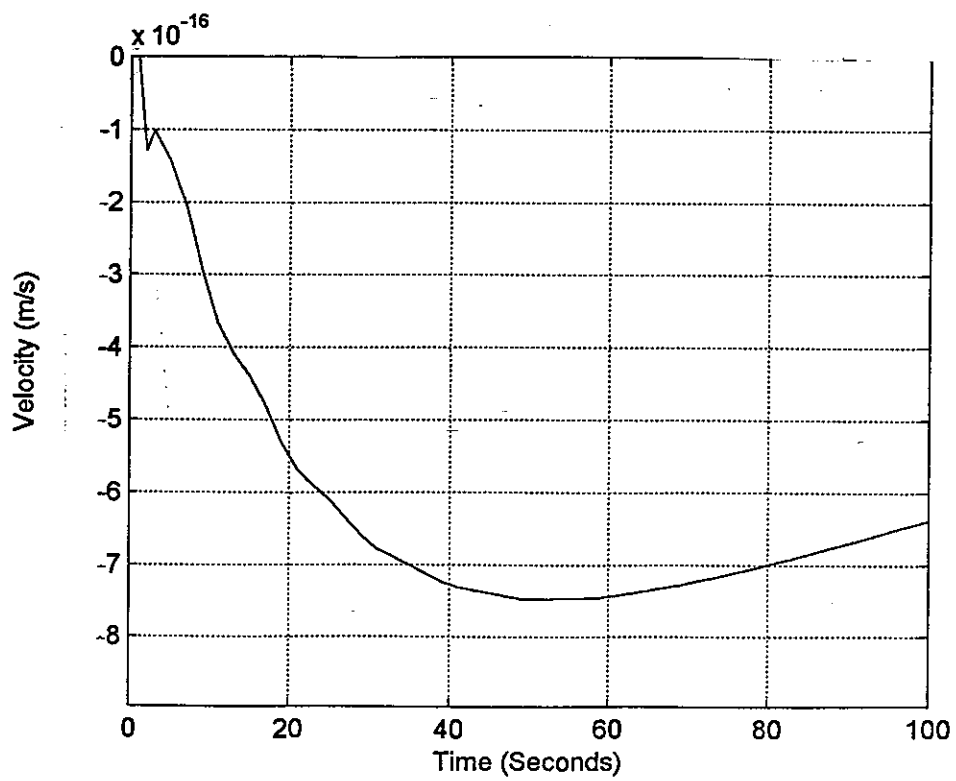
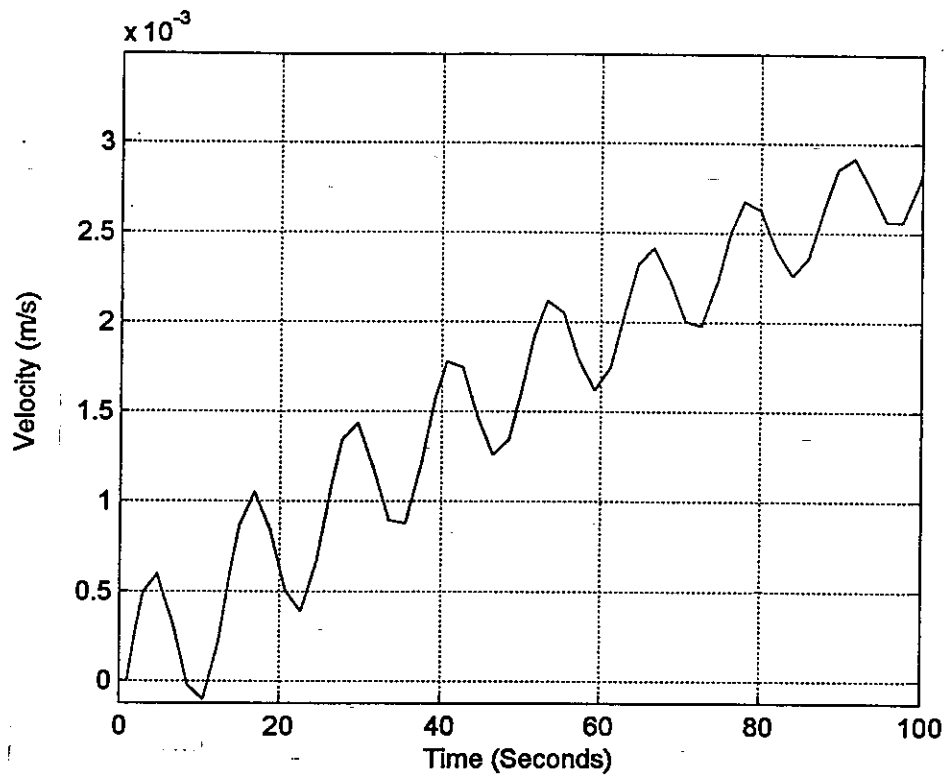


Figure 4: Response of the Velocity State Variable to a Step Heading Angle Command

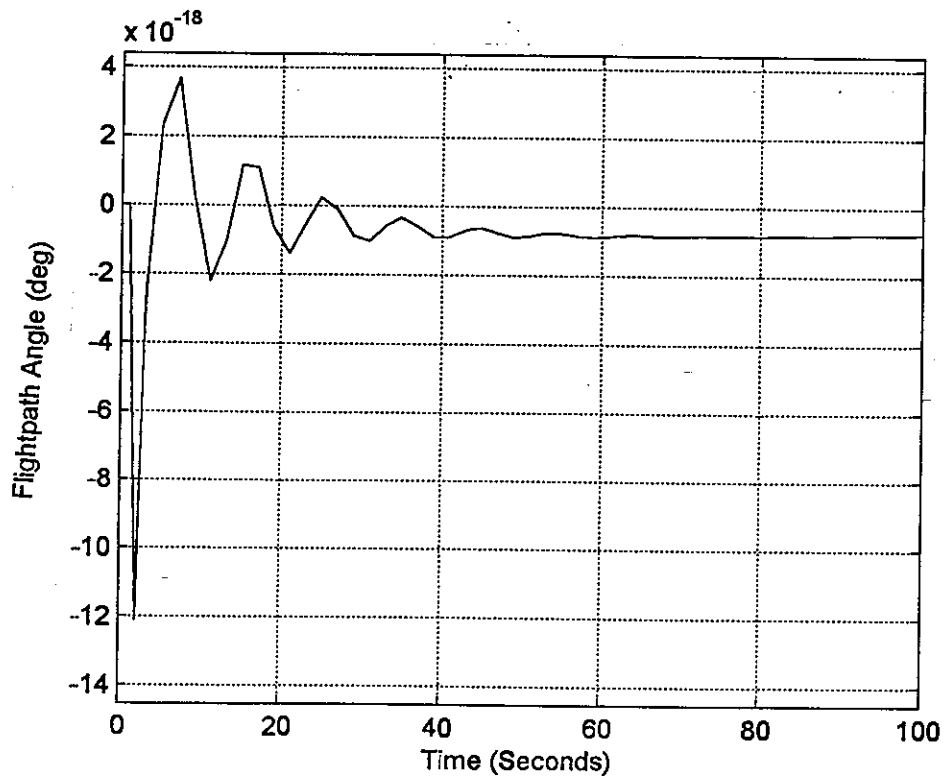
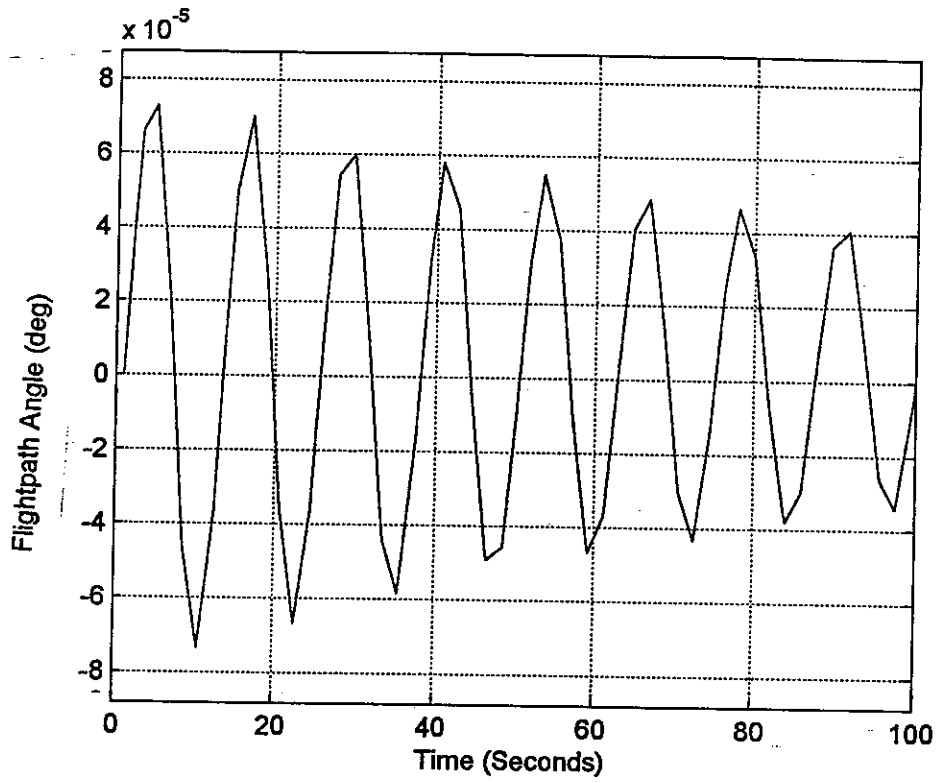


Figure 5: Response of the Flightpath Angle State Variable to a Step Heading Angle Command

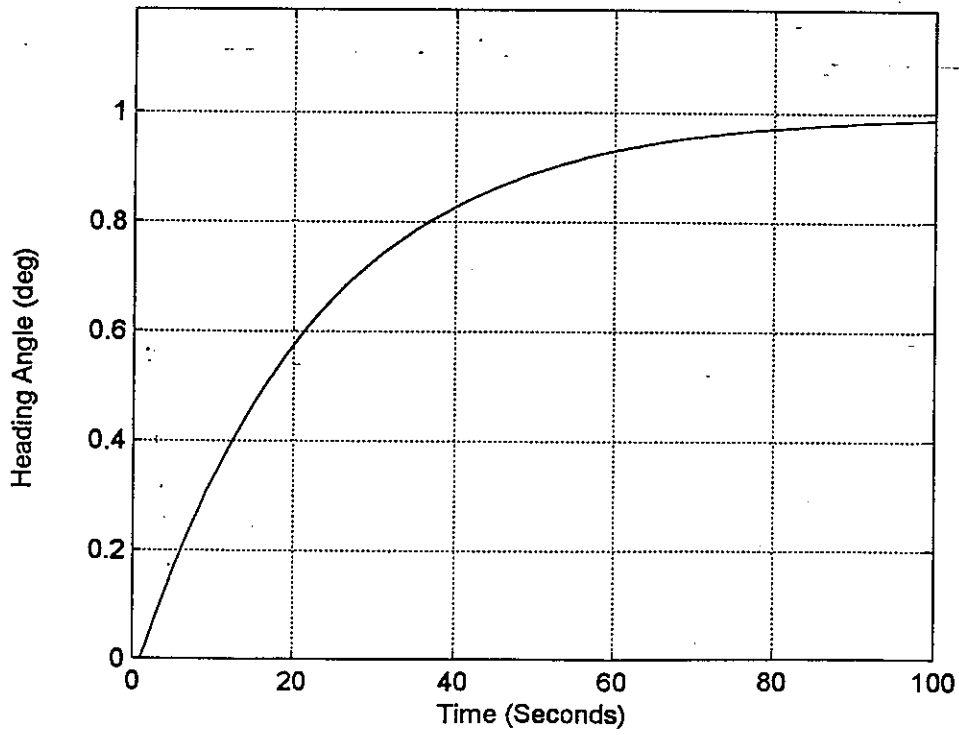
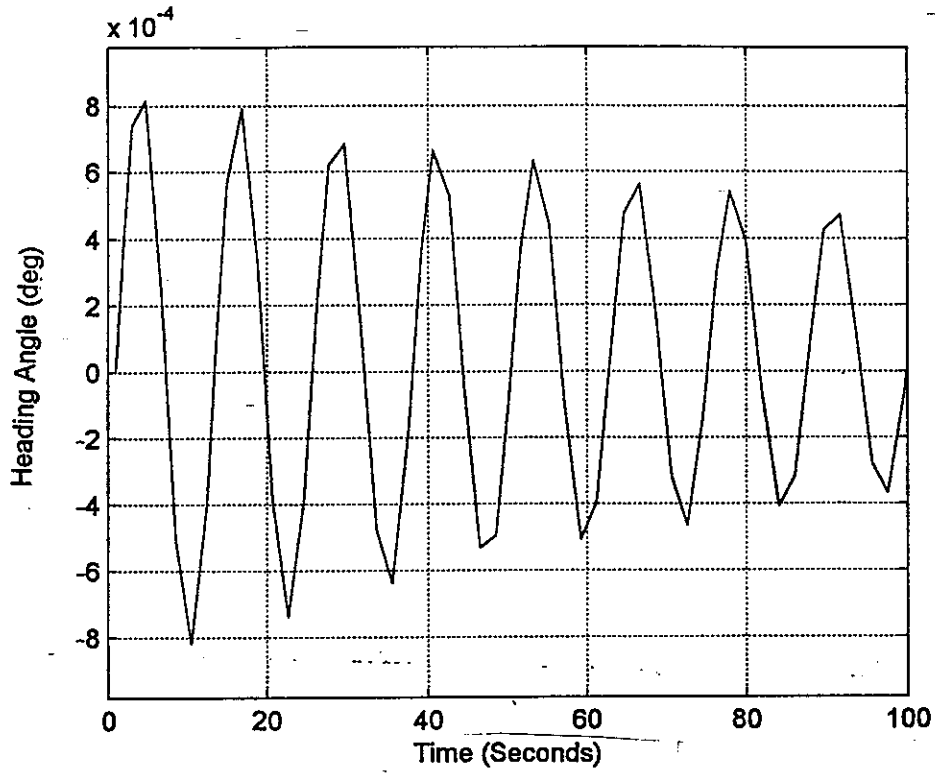


Figure 6: Response of the Heading Angle State Variable to a Step Heading Angle Command

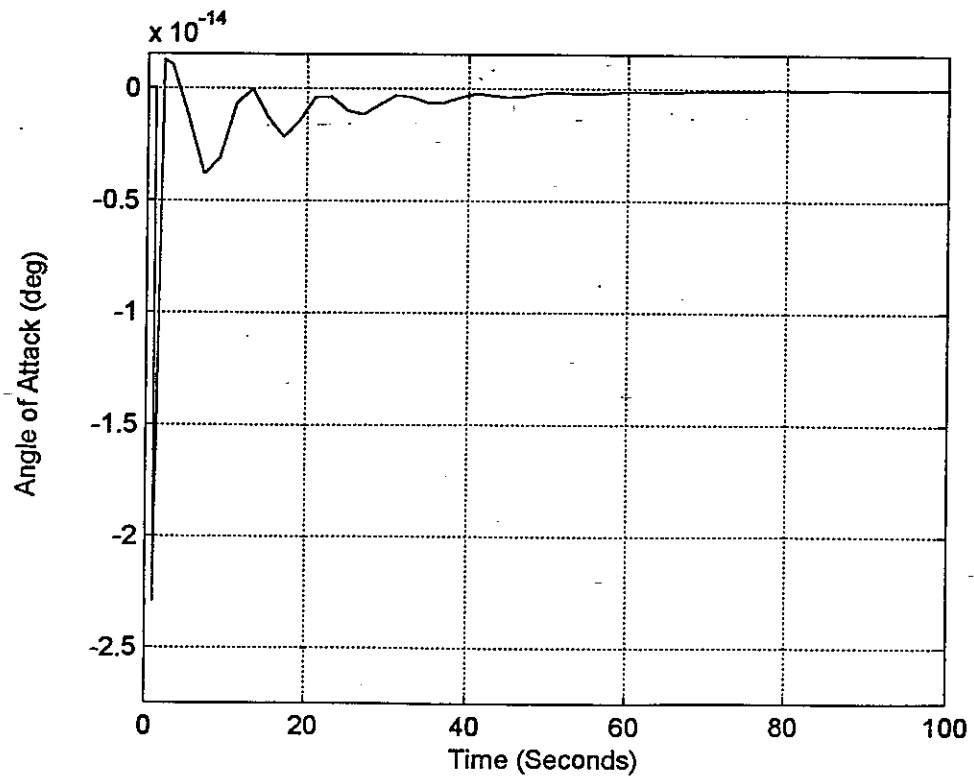
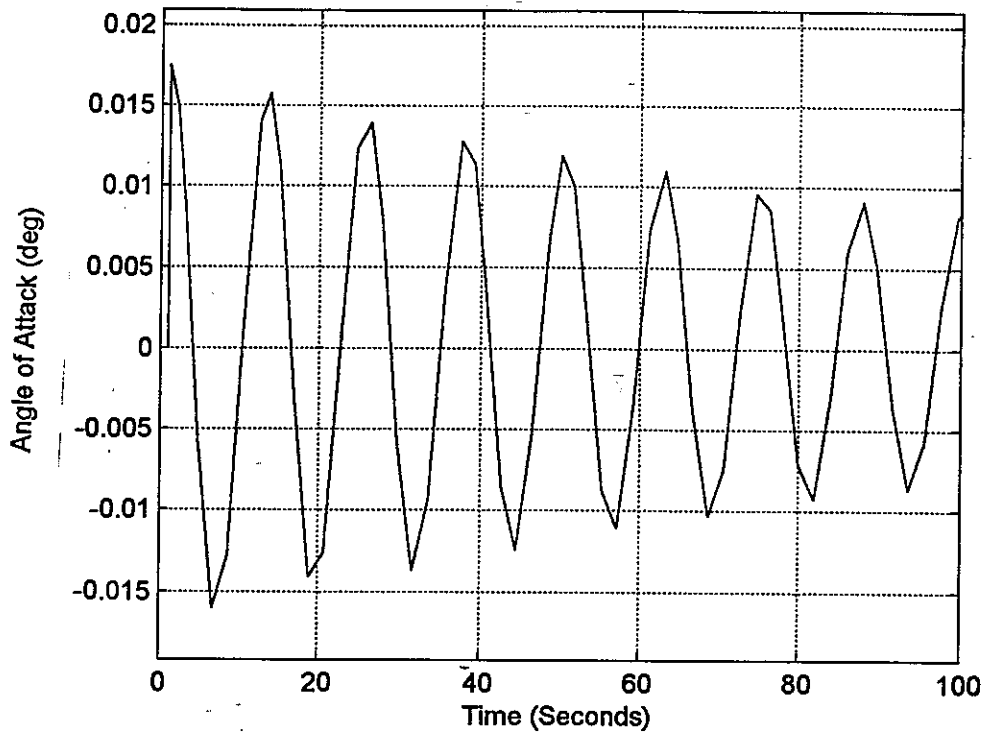


Figure 7: Response of Angle of Attack Control Variable to a Step Heading Angle Command

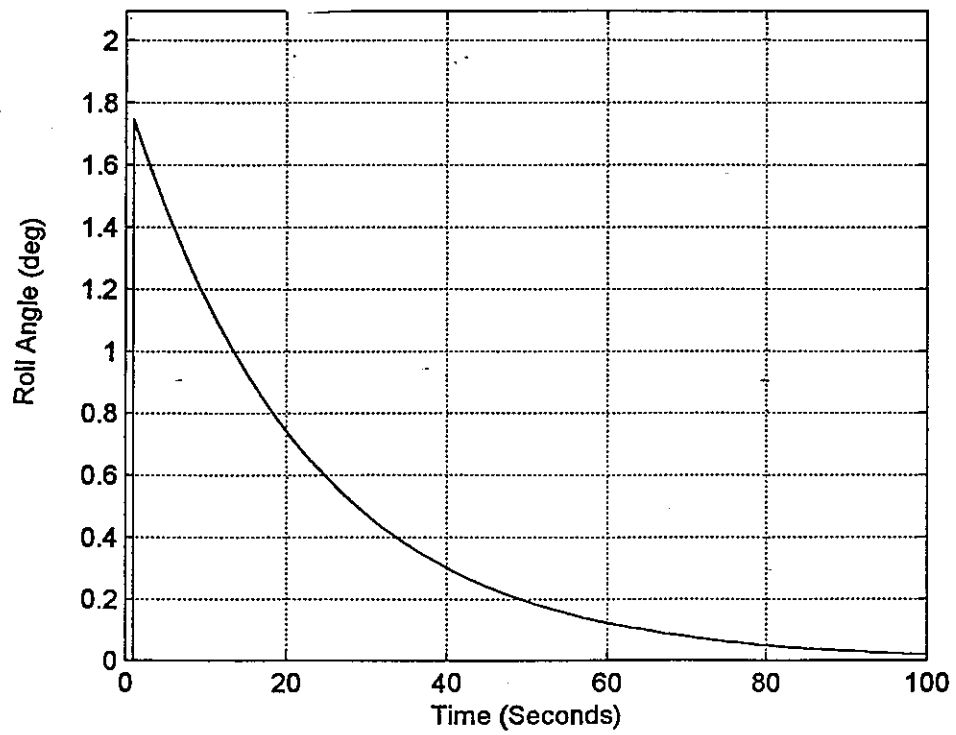
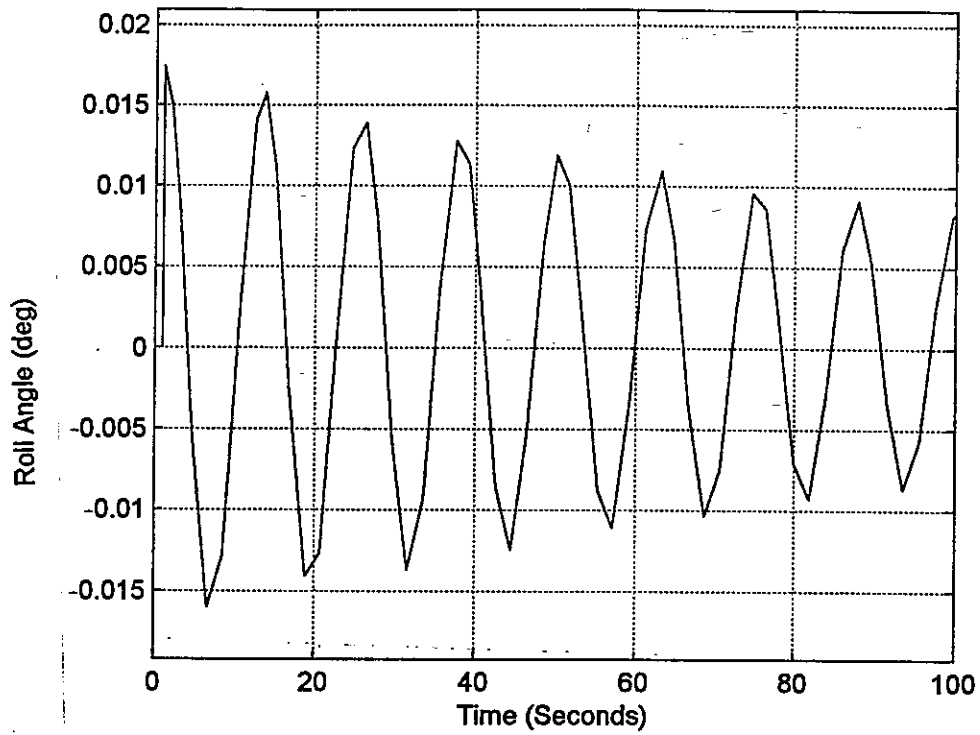


Figure 8: Response of Roll Angle Control Variable to a Step Heading Angle Command

Figure 9: Comparison of EOM and Data for Height

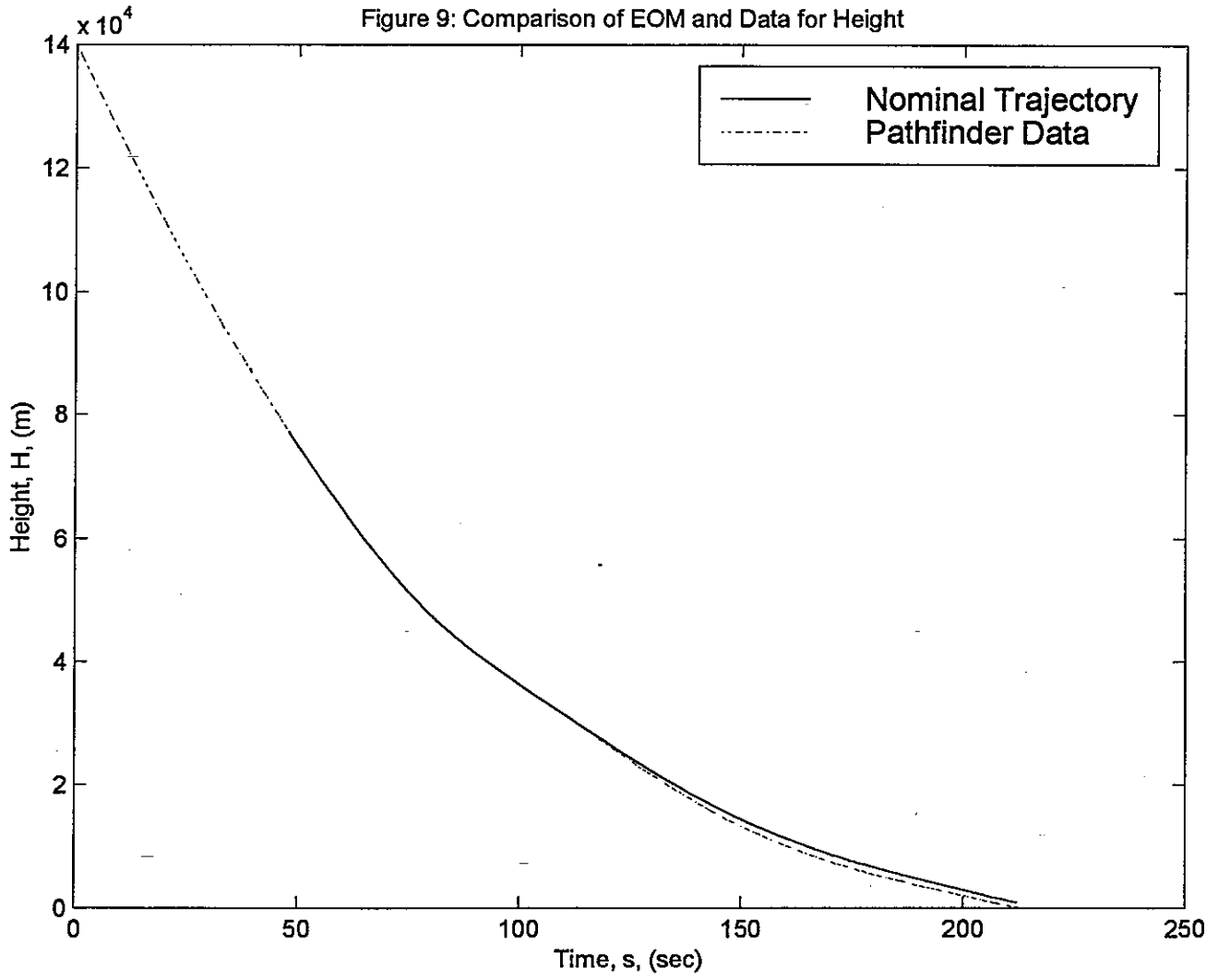


Figure 10: Comparison of EOM and Data for Velocity

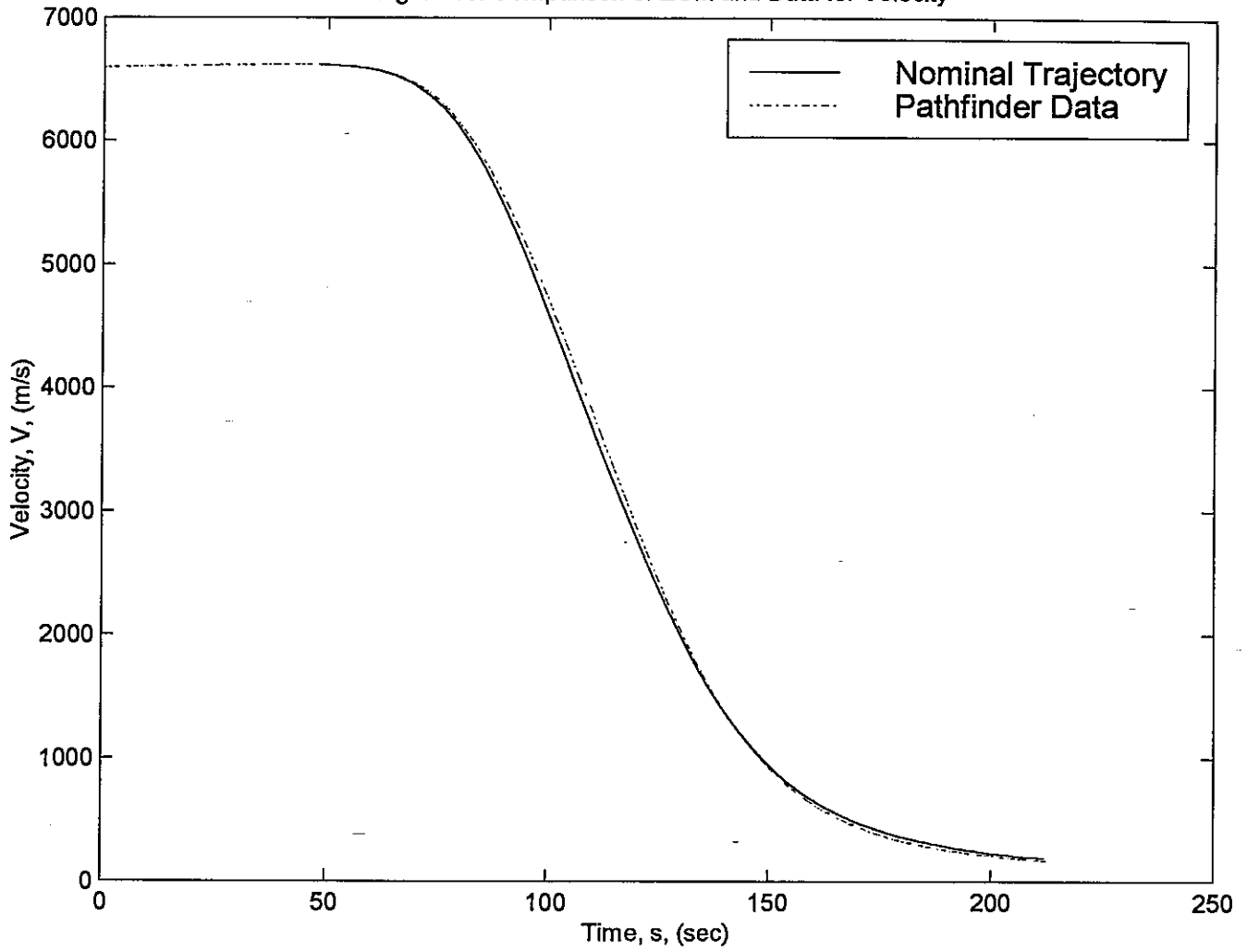


Figure 11: Comparison of EOM and Data for Flightpath Angle

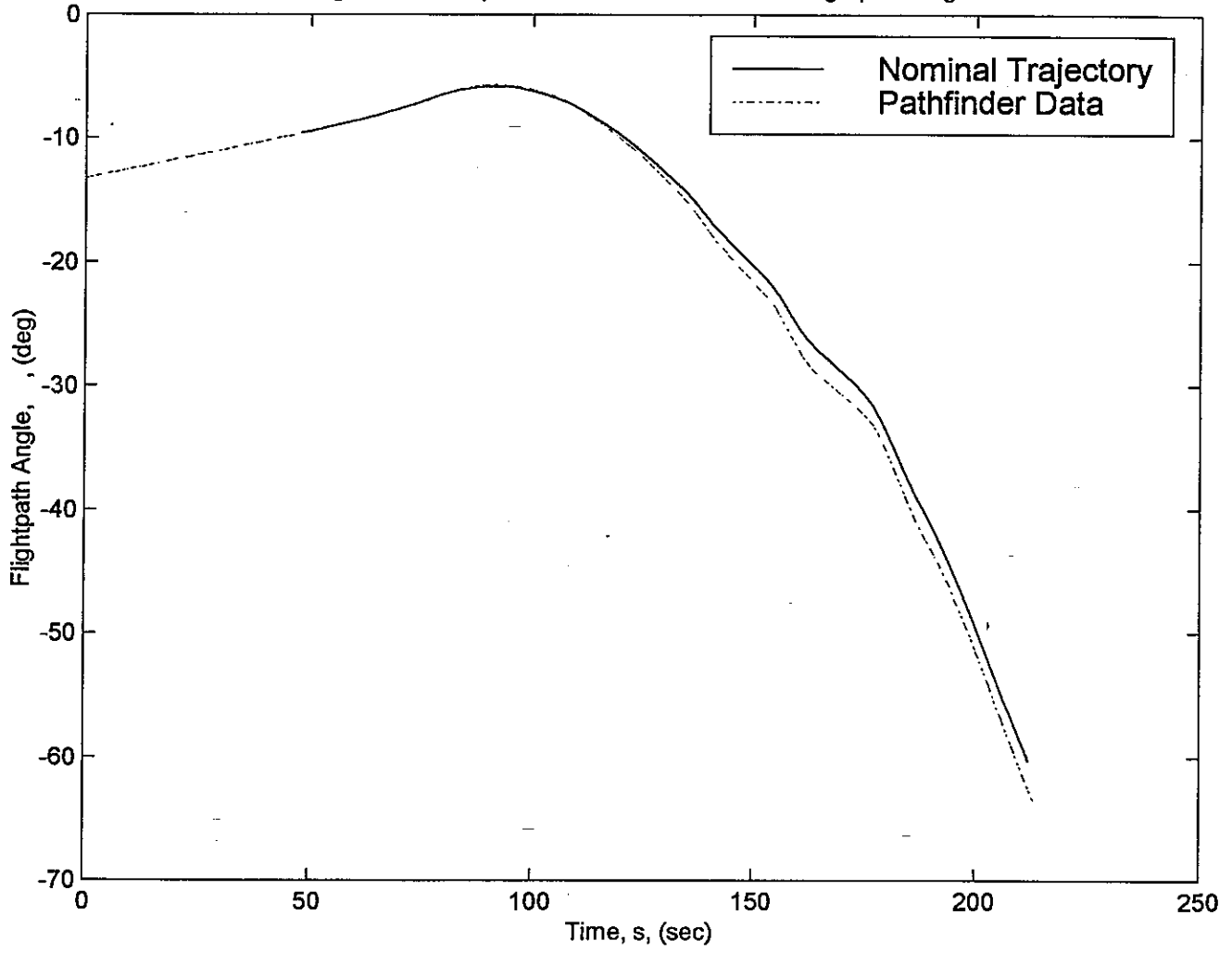


Figure 12: Heading Angle found from Integration of EOM

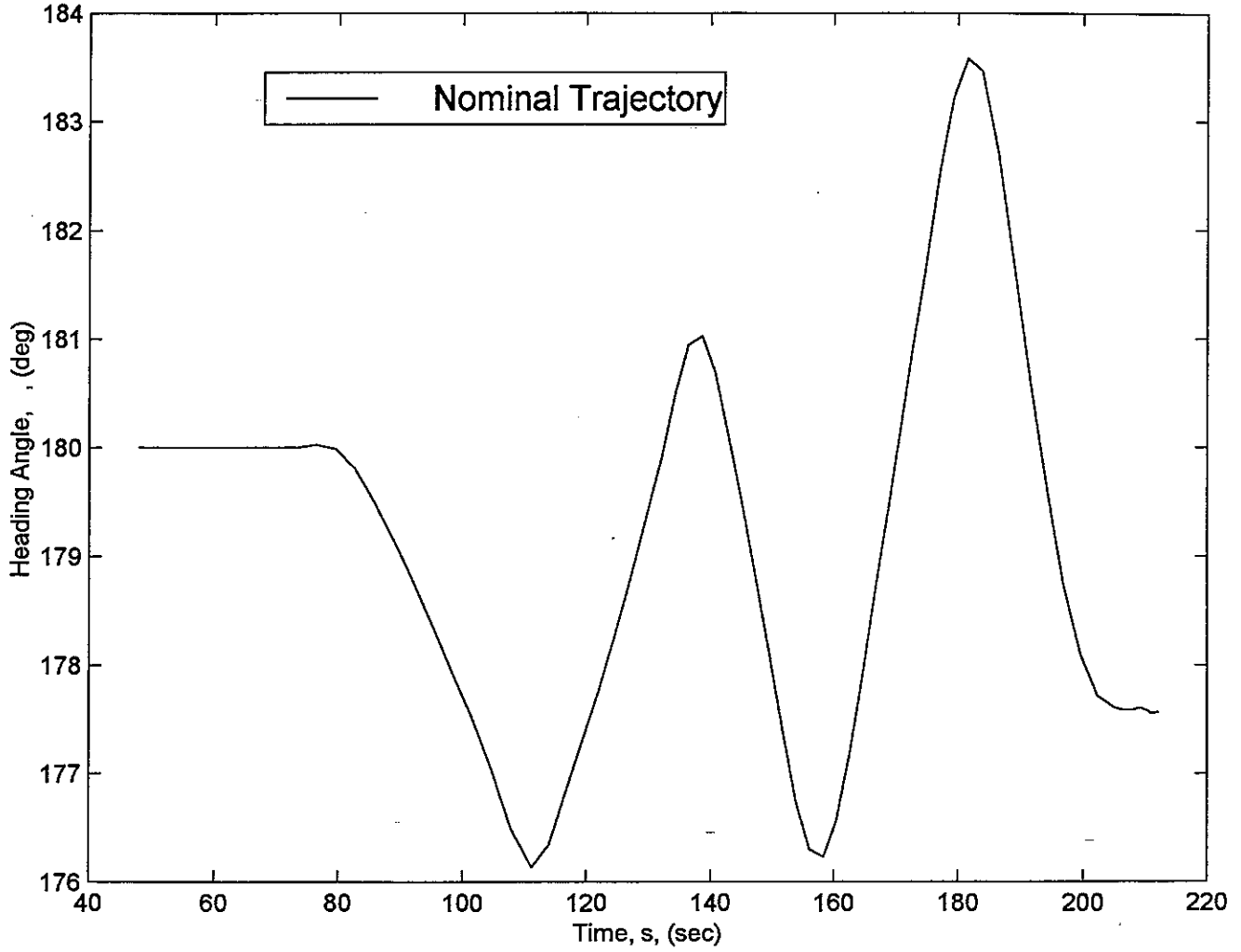


Figure 13: Comparison of Nominal Trajectory and Gain Controlled EOM for Height

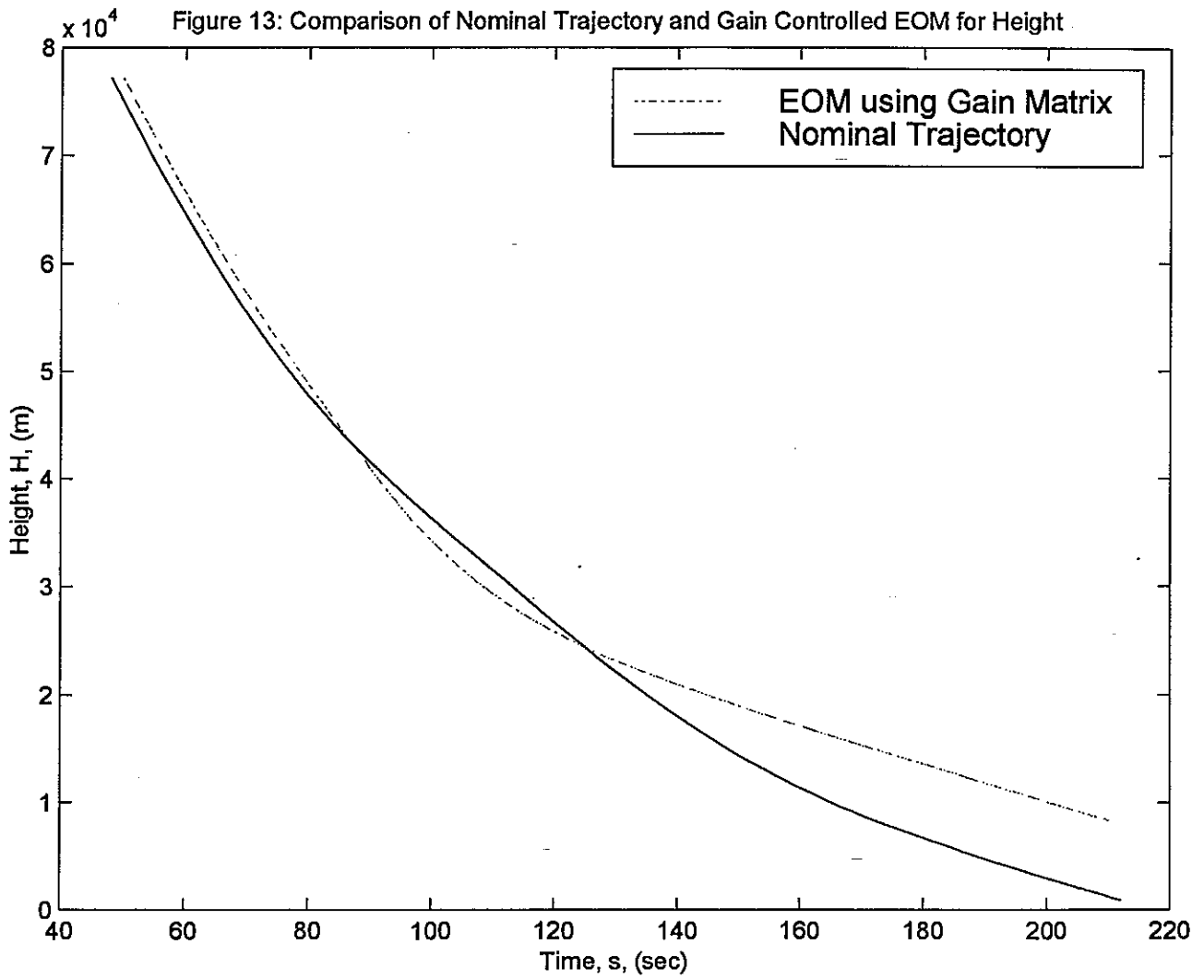


Figure 14: Comparison of Nominal Trajectory and Gain Controlled EOM for Velocity

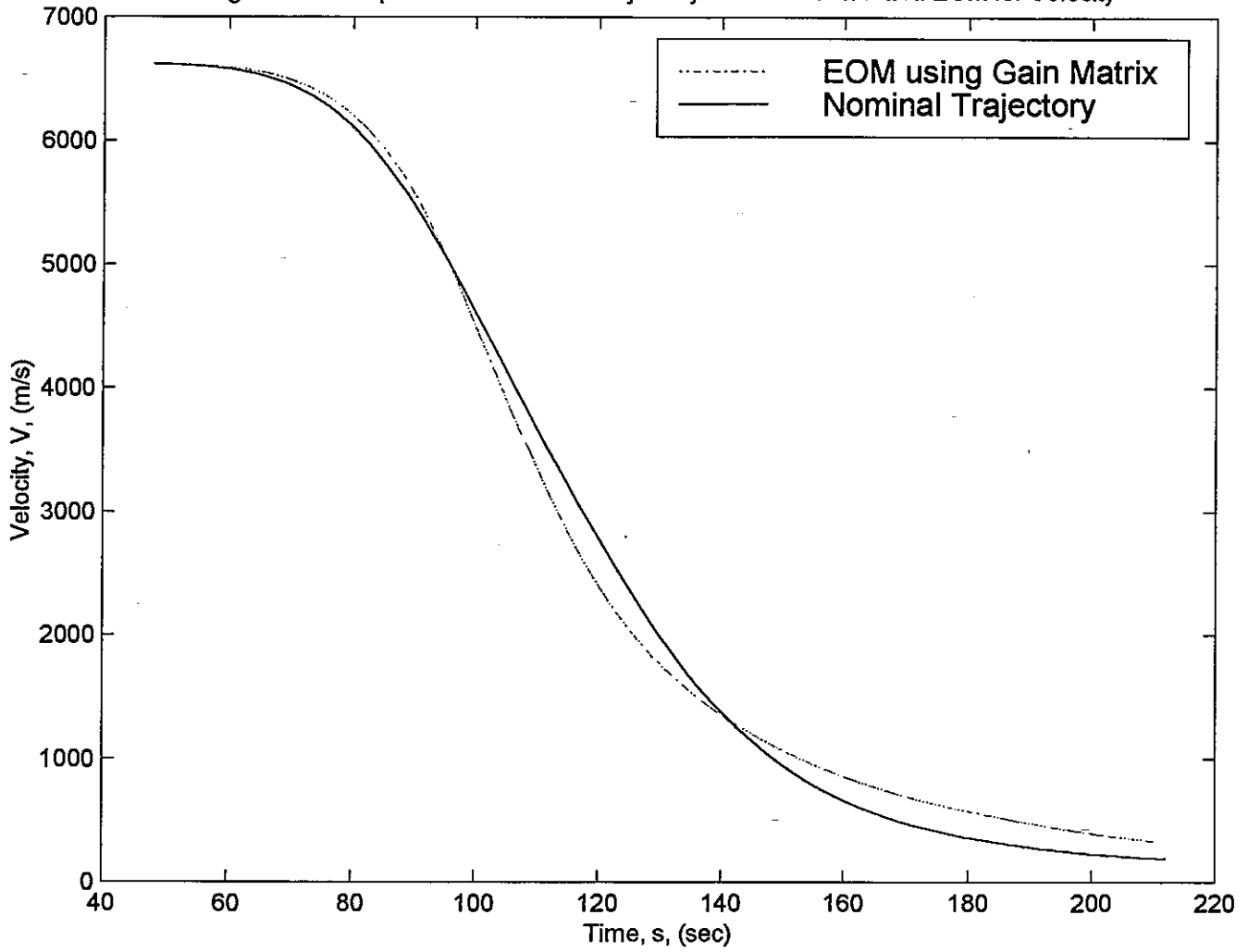


Figure 15: Comparison of Nominal Trajectory and Gain Controlled EOM for Flightpath Angle

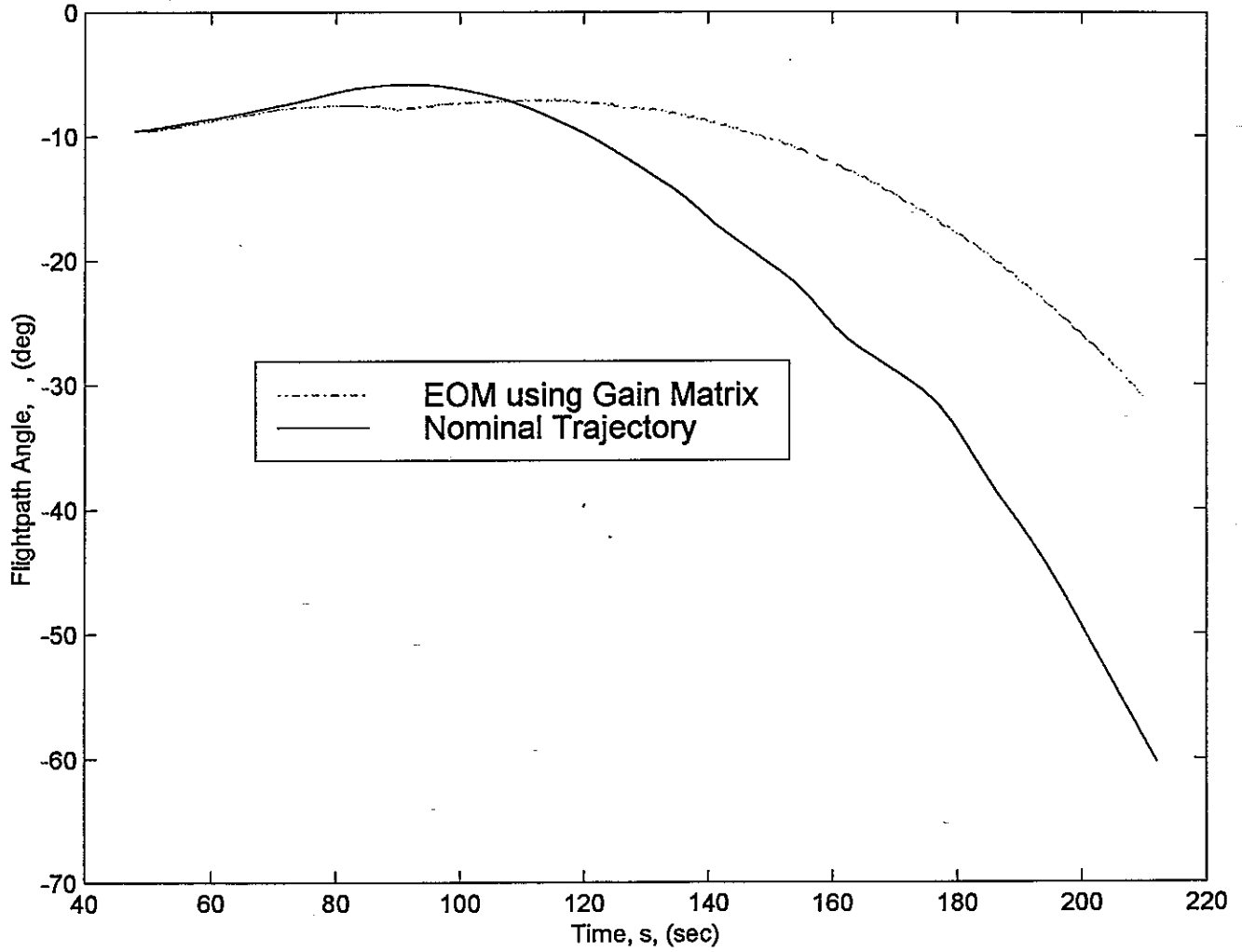
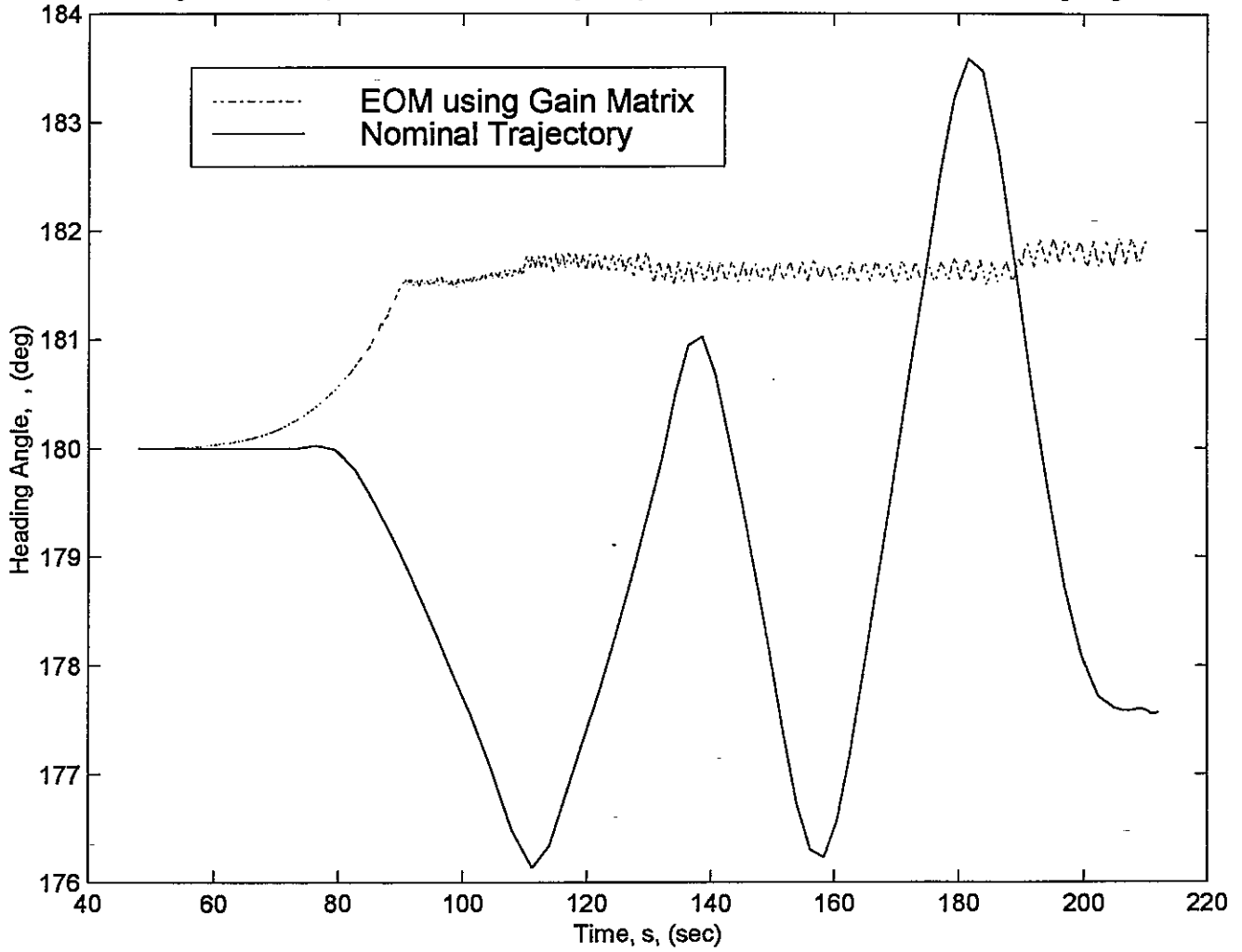


Figure 16: Comparison of Nominal Trajectory and Gain Controlled EOM for Heading Angle



Mach #	H	V	Gamma	Psi	
34.7	100.01 0.00	-3.00 0.00	624040.35 0.00	0.00 3162.30	Alpha Phi
32.6	9.99 0.00	-0.98 0.00	56768.01 0.00	0.00 100.00	Alpha Phi
27.2	-0.01 1.00	0.67 0.07	-40.04 2376.10	-30.82 7.08	Alpha Phi
18.3	-0.11 1.00	0.26 0.09	-312.55 1738.30	-99.91 -4.21	Alpha Phi
10.1	-0.15 -0.99	0.41 -0.01	-485.62 -1040.30	314.71 -30.91	Alpha Phi
4.7	-0.10 0.99	0.34 -0.16	-223.89 575.88	-315.07 -27.007	Alpha Phi
2.2	0.02 -1.00	0.15 0.42	29.45 -369.07	99.97 2.31	Alpha Phi
1.3	-0.01 1.00	0.01 -0.81	-18.50 255.73	-99.99 -0.92	Alpha Phi
0.9	-0.99 0.00	4.61 0.00	-633.94 0.00	0.00 -3.16	Alpha Phi

Table 1: Gain Calculated from LQR for each Breakpoint

References

- [1] Goodall, K., "2001 Lander Detailed Document," 2001 Mars Surveyor. 1998
<http://www.mars.sgi.com/2001/lander/pdf-lander.html>, (24, Mar 1998), pp. 9.

- [2] Queen, E. M., and Thompson, A., "Effect of Payload on Risk of Vehicle Loss Due to Engine Failure," *Journal of Spacecraft and Rockets*, Vol. 32, No 3, Nov. 1994, pp. 566-568.

- [3] Duke, E. L., Antoniewicz, R. F., Krambeer, K.D., "Derivation and Definition of a Linear Aircraft Model," NASA Reference Publication 1207, Aug. 1988, pp. 3-2.

- [4] Stevens, B.L., and Lewis, F.L., "Linear Quadratic Design with Full State Feedback," *Aircraft Control and Simulation*, 1st ed., John Wiley & Sons, Inc., New York, 1992, pp.437-444.

Appendix A: Matlab Programs

```

%
% marscon.m
%
% this does optimal control design for many design points.
%
% Eric M. Queen 2/9/95
% modified for Mars 4/11/96
% modified for '01 Lander 7-97 to 8-98 BAT

%global i del marscv

load mtqdp
load marscv
load gainm2
refdata;

[ntpts, flub]=size(mtqdp);
machv=mtqdp(:,1)';
timev=mtqdp(:,2)';
dynpv=mtqdp(:,3)';

for i=1:ntpts
    disp(['mach # ' num2str(machv(i))])
    [A,B,Ab,Bb,C,D]=ab(marscv,i);
    % q=mtqdp(i,4:7);
    % r=mtqdp(i,8:9);
    % [k,s,e]=lqr(A,B,diag(q),diag(r));
    % disp(' [k,s,e]=lqr(A,B,diag(q),diag(r));')
    for j=1:4
        k(1,j)=gainm2(2*i-1,j);
        k(2,j)=gainm2(2*i,j);
    end

    keyboard
    % mtqdp(i,4:7)=q;
    % mtqdp(i,8:9)=r;
    % for j=1:4
    %     gainm(2*i-1,j)=k(1,j);
    %     gainm(2*i,j)=k(2,j);
    % end
    % save mtqdp mtqdp
    % save gainm gainm
end;

```

```

%
% refdata.m
% this provides reference data for the Mars 2001 entry vehicle
%
global h0 U0 scale munum re S

bdiam=2.65;    % base diameter (m)
cbar=bdiam;
b=bdiam;
S=pi*(bdiam/2)^2;    % ref area (sq m)
%
munum=4.28282868534e+13;    % gravitational constant at Mars
re=3.393940e+06;    % Mars equatorial radius (m)
rp=3.376780e+06;    % Mars polar radius (m)
g0=munum/(re*re);
h0=15000;    % height for normilation (m)
U0=1.1339e+03;    % velocity for normilation (m/s)
U0=6.6210e+003;
%
scale=7635.8;    % scale height
%
xrp=0;    % reference point is at CG
yrp=0;
zrp=0;
%
% Coordinate system is completely different from usual x-out-nose
xcg=-1.291763;
ycg=-0.005622091;
zcg=-0.009761381;

% Coordinate system is completely different
% rearranged to make Ixz the largest abs val.
Ix=1.964899e6/10000;    % convert from kg*cm^2 to kg*m^2
Iy=1.633986e6/10000;
Iz=1.374171e6/10000;
Ixz=11325.13/10000;
Iyz=17753.79/10000;
Ixy=6360.292/10000;

%fifteen percent margin
Ix=Ix*1.15;
Iy=Iy*1.15;
Iz=Iz*1.15;
Ixz=Ixz*1.15;
Iyz=Iyz*1.15;
Ixy=Ixy*1.15;

```

```

%
% data.m
%
% this evaluates nonlinear data
% to provide aero coefficients at given breakpoints
%
%*****%
%
% read in tabular data
load nom_mf.mat
%

idx=[50:20:215];
timestations=[timem[idx]];
nstation=length(timestations);
%
% loop to create file of variables necessary to calculate
% A,B,C,D matrices at each breakpoint
%
refdata;
alpha=zeros(nstation,1);
marscv=zeros(nstation,25);

for i=1:nstation
    time_ic=timem(idx[i]);
    mass=weightm(idx(i))/9.8;
    reh=re+gdaltm(idx(i));
    sgam=sin(pi/180*gamarm(idx(i)));
    cgam=cos(pi/180*gamarm(idx(i)));
    sph=sin(pi/180*rolrm(idx(i)));
    cphi=cos(pi/180*rolrm(idx(i)));

    calculate rho0 for each breakpoint from the density equation:
    rho=rho0*e^(-1/scale height * height)
    rho0=densm(idx(i))/exp(-1/scale*gdaltm(idx(i)));

    alpha(i)=alphan[idx(i)];
    bastage=[velrm(idx(i)) machm(idx(i)) 0 betam(idx(i))];
    save input1 bastage -ascii;
    eval(['!main < input1']);
% pathaero contains:
% cn ca cy cm cw cll cmq cnr cyb cllb cw
load pathaero.dat;
cn(1)=pathaero(1);
ca(1)=pathaero(2);

    bastage=[velrm(idx(i)) machm(idx(i)) alpha(i)/2 betam(idx(i))];
    save input1 bastage -ascii;
    eval(['!main < input1']);
% pathaero contains:
% cn ca cy cm cw cll cmq cnr cyb cllb cw
load pathaero.dat;
cn(2)=pathaero(1);
ca(2)=pathaero(2);
al=alpha(i)/2;

    bastage=[velrm(idx(i)) machm(idx(i)) alpha(i) betam(idx(i))];
    save input1 bastage -ascii;
    eval(['!main < input1']);
% pathaero contains:
% cn ca cy cm cw cll cmq cnr cyb cllb cw
load pathaero.dat;
cn(3)=pathaero(1);
ca(3)=pathaero(2);
a2=alpha(i);

```

```

% Calculate parabolic drag coefficients
Cd0=-ca(1)*cos(0)-cn(1)*sin(0);
Cd1=-ca(2)*cos(a1*pi/180)-cn(2)*sin(a1*pi/180);
Cd2=-ca(3)*cos(a2*pi/180)-cn(3)*sin(a2*pi/180);

% Calculate Cdalpha & Cdalpha2
Cda=((Cd2-Cd0)*a1^2+(Cd0-Cd1)*a2^2)/(a1^2*a2-a2^2*a1);
Cda2=((Cd0-Cd2)*a1+(Cd1-Cd0)*a2)/(a1^2*a2+a2^2*a1);

% Calculate Clalpha
Cl=ca(3)*sin(a2*pi/180)-cn(3)*cos(a2*pi/180);
Cla=Cl/a2;

% Calculate Drag
drag=-(Cd0+Cda*alpha(i)+Cda2*alpha(i)^2)*S*dynpm(idx(i));

% Calculate Lift
lift=-Cla*alpha(i)*S*dynpm(idx(i));

% marslong(i,:)= [sgam h0 velrm(idx(i)) cgam mu reh U0 ...
%               Cd0 Cda Cda2 rho0 S mass Cla alpha(i) ...
%               sphi cphi scale gdaltm(idx(i)) dragwm(idx(i)) ...
%               drag liftm(idx(i)) lift time_ic gammarm(idx(i))];

% vartab(i,:)= [time_ic rolrm(idx(i)) Cla Cd0 Cda Cda2 rho0];

mtqdp2(i,:)= [machm(idx(i)) time_ic dynpm(idx(i)) ones(1,6)];

end;

%**** save all the important data to an appropriately named file
%save marslong marslong
ave mtqdp2 mtqdp2

```

```
function [A,B,Ab,Bb,C,D]=ab(marscv,i)
```

```
% Computes the A & B state matrices
```

```
% marscv(i,:)= [sgam h0 velrm(idx(i)) cgam mu reh U0 Cd0 Cda ...  
                Cda2 rho S mass Cla alphas(idx(i)) sphi cphi scale alt];
```

```
sgam=marscv(i,1);  
h0=marscv(i,2);  
U=marscv(i,3);  
cgam=marscv(i,4);  
mu=marscv(i,5);  
reh=marscv(i,6);  
U0=marscv(i,7);  
Cd0=marscv(i,8);  
Cda=marscv(i,9);  
Cda2=marscv(i,10);  
rho0=marscv(i,11);  
S=marscv(i,12);  
mass=marscv(i,13);  
Cla=marscv(i,14);  
alpha=marscv(i,15);  
sphi=marscv(i,16);  
cphi=marscv(i,17);  
scaleh=marscv(i,18);  
h=marscv(i,19);
```

```
s=1/scaleh;  
rho=rho0*exp(-s*h);  
q=1/2*rho*U^2;
```

```
L=-q*S*Cla*alpha;  
L=-q*S*(Cd0+Cda*alpha+Cda2*alpha^2);
```

```
Ab=[0 sgam U*cgam 0;  
    D*s/(mass)+2*mu*sgam/(reh^3) -2*D/(mass*U) -mu*cgam/(reh^2) 0;  
    -s*L*cphi/(mass*U)-U*cgam/reh^2+2*mu*cgam/(U*reh^3) L*cphi/(mass*U^2)+...  
    cgam/reh+mu*cgam/(U^2*reh^2) -U*sgam/reh+mu*sgam/(U*reh^2) 0;  
    -s*L*sphi/(mass*U*cgam) L*sphi/(mass*U^2*cgam) L*sphi*sgam/(mass*U*cgam^2) 0];
```

```
Bb=[0 0;  
    -q*S*(Cda+2*Cda2*alpha)/(mass) 0;  
    L*cphi/(mass*U*alpha) -L*sphi/(mass*U);  
    L*sphi/(mass*U*cgam*alpha) L*cphi/(mass*U*cgam)];
```

```
C=eye(4);
```

```
D=zeros(4,2);
```

```

%
% nominal.m
%
% this creates a nominal trajectory from the EOM
%
8-98 BAT

refdata;
load nom_mf.mat
load vartab

global S munum scale re

idx=[50:1:215];
timestations=[timem(idx)];
nstation=length(timestations);

i=1;

x0(1)=gdaltm(idx(i));
x0(2)=velrm(idx(i));
x0(3)=gammarm(idx(i))*pi/180;
x0(4)=bnkangm(idx(i))*pi/180;
x0=x0';

tspan=[48 212]
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4]);
[ttraj,nomtraj]=ode45('eom',tspan,x0,options,vartab);

%plot results
figure(1)
clf
plot(ttraj,nomtraj(:,1),'k-',timem,gdaltm,'k-.')
legend('Nominal Trajectory','Pathfinder Data')
ylabel('Height, H, (m)')
xlabel('Time, s, (sec)')
title('Figure 9: Comparison of EOM and Data for Height')

figure(2)
clf
plot(ttraj,nomtraj(:,2),'k-',timem,velrm,'k-.')
legend('Nominal Trajectory','Pathfinder Data')
ylabel('Velocity, V, (m/s)')
xlabel('Time, s, (sec)')
title('Figure 10: Comparison of EOM and Data for Velocity')

figure(3)
clf
plot(ttraj,nomtraj(:,3)*180/pi,'k-',timem,gammarm,'k-.')
legend('Nominal Trajectory','Pathfinder Data')
ylabel('Flightpath Angle, (deg)')
xlabel('Time, s, (sec)')
title('Figure 11: Comparison of EOM and Data for Flightpath Angle')

figure(4)
clf
plot(ttraj,nomtraj(:,4)*180/pi,'k-')
legend('Nominal Trajectory')
ylabel('Heading Angle, (deg)')
xlabel('Time, s, (sec)')
title('Figure 12: Heading Angle found from Integration of EOM')

save nomtraj nomtraj
save ttraj ttraj

```

```

% Brande Tuck
% eom.m
% this program defines the equations of motion for the lander

function a=eom(t,x,options,vartab,nomtraj2)
    global S scale munum re U0 h0

% table of variables(t): t,alpha,phi,Cla,Cd0,Cda,Cda2,rho0,mass
time=vartab(:,1);
table=[vartab(:,2) vartab(:,3) vartab(:,4) vartab(:,5) vartab(:,6) vartab(:,7) vartab(:,8)
vartab(:,9)];
tab=interp1(time,table,t,'cubic');

L=-0.5*tab(7)*exp(-x(1)/scale)*x(2)^2*S*tab(3)*tab(1);
D=0.5*tab(7)*exp(-x(1)/scale)*x(2)^2*S*(tab(4)+tab(5)*tab(1)+tab(6)*tab(1)^2);

a(1)=x(2)*sin(x(3));
a(2)=D/(tab(8)) - munum*sin(x(3))/(re+x(1))^2;
a(3)=L*cos(tab(2)*pi/180)/(tab(8)*x(2)) + x(2)*cos(x(3))/(x(1)+re) -
munum*cos(x(3))/(x(2)*(re+x(1))^2);
a(4)=L*sin(tab(2)*pi/180)/(tab(8)*x(2)*cos(x(3)));

a=a';

```

```

%
% nonlin.m
%
% uses the gain schedule matrix to calculate the control matrix
% when integrating the EOM
% 8-98 BAT

refdata;
load nom_mf.mat
load vartab
load nomtraj
load ttraj
load gainm2

idx=[50:20:215];
timestations=[timem(idx)];
nstation=length(timestations);

% Initial value
x0=[nomtraj(1,1)
    nomtraj(1,2)
    nomtraj(1,3)
    nomtraj(1,4)];

for i=1:nstation-1

    tspan=[idx(i) idx(i+1)];

    for j=1:4
        k(1,j)=gainm2(2*i-1,j);
        k(2,j)=gainm2(2*i,j);
    end

    if i==1
        options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4]);
        [t,x]=ode45('eomnon',tspan,x0,options,vartab,k);
        [tr,tc]=size(t);
        timevec=t;
        xvec=x;
    else
        x0=[x(tr)
            x(tr)
            x(tr)
            x(tr)];
        [t,x]=ode45('eomnon',tspan,x0,options,vartab,k);
        [tr,tc]=size(t);
        timevec=[timevec;t];
        xvec=[xvec;x];
    end
end

%plot results
figure(1)
clf
plot(timevec,xvec(:,1),'b-.',ttraj,nomtraj(:,1),'b-')
title('Figure 13: Comparison of Nominal Trajectory and Gain Controlled EOM for Height')
legend('EOM using Gain Matrix','Nominal Trajectory')
xlabel('Height, H, (m)')
ylabel('Time, s, (sec)')

figure(2)
clf
plot(timevec,xvec(:,2),'b-.',ttraj,nomtraj(:,2),'b-')
title('Figure 14: Comparison of Nominal Trajectory and Gain Controlled EOM for Velocity')

```

```
legend('EOM using Gain Matrix','Nominal Trajectory')
xlabel('Velocity, V, (m/s)')
ylabel('Time, s, (sec)')
```

```
figure(3)
```

```
clf
plot(timevec,xvec(:,3)*180/pi,'b-.',ttraj,nomtraj(:,3)*180/pi,'b-')
title('Figure 15: Comparison of Nominal Trajectory and Gain Controlled EOM for Flightpath Angle')
legend('EOM using Gain Matrix','Nominal Trajectory')
xlabel('Flightpath Angle, , (deg)')
ylabel('Time, s, (sec)')
```

```
figure(4)
```

```
clf
plot(timevec,xvec(:,4)*180/pi,'b-.',ttraj,nomtraj(:,4)*180/pi,'b-')
title('Figure 16: Comparison of Nominal Trajectory and Gain Controlled EOM for Heading Angle')
legend('EOM using Gain Matrix','Nominal Trajectory')
xlabel('Heading Angle, , (deg)')
ylabel('Time, s, (sec)')
```

```

% Brande Tuck

% eomnon.m
% this program defines the equations of motion for the lander
% calculates alpha & phi from gain matrix, k

function a=eomnon(t,x,options,vartab,k)

global scale munum re S

% table of variables(t): t,alpha**,phi**,Cla,Cd0,Cda,Cda2,rho0,mass **not used
time=vartab(:,1);
table=[vartab(:,2) vartab(:,3) vartab(:,4) vartab(:,5) vartab(:,6) vartab(:,7) vartab(:,8)
vartab(:,9)];
tab=interp1(time,table,t,'linear');

Cla=tab(3); Cd0=tab(4); Cda=tab(5); Cda2=tab(6); rho0=tab(7); mass=tab(8);

% control variables calculated from gain matrix
del=k*x;

alpha=del(1)*pi/180;
phi=del(2)*pi/180;

cphi=cos(phi);
sphi=sin(phi);
cgam=cos(x(3));
sgam=sin(x(3));
reh=re+x(1);

% saturation limit on alpha
if alpha > 15.0
    alpha=15.0;
elseif alpha < -15.0
    alpha = -15.0;
end;

% Lift and Drag equations
L=-0.5*rho0*exp(-x(1)/scale)*x(2)^2*S*Cla*alpha;
D=-0.5*rho0*exp(-x(1)/scale)*x(2)^2*S*(Cd0+Cda*alpha+Cda2*alpha^2);

% EOM
a(1)=x(2)*sgam;
a(2)=-D/(mass) - munum*sgam/(reh^2);
a(3)=L*cphi/(mass*x(2)) + x(2)*cgam/(reh) - munum*cgam/(x(2)*reh^2);
a(4)=L*sphi/(mass*x(2)*cgam);

a=a';

```