2008

Skip entry trajectory planning and guidance

Christopher William Brunner
Iowa State University

Skip entry trajectory planning and guidance

by

Christopher William Brunner

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Aerospace Engineering

Program of Study Committee:
Ping Lu, Major Professor
Bion Pierson
Bong Wie
Atul Kelkar
Steven Kawaler

Iowa State University

Ames, Iowa

2008

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

This work would not have been as successful without the superb guidance of Professor Ping Lu. His knowledge and motivational factors contributed greatly to the development of the skip guidance algorithm. Throughout the development process, the discussions held with him were of fundamental importance in resolving key issues. The computational resources provided greatly sped up extensive Monte Carlo simulations. Professor Lu made excellent suggestions in choosing coursework, that provided additional ideas for the development of the algorithm.

The author would also like to the program of study committee for their helpful ideas for improving this dissertation.

Many thanks to the bevy of graduate students who provided answers to questions, helpful suggestions, help with the LaTeXwriting, and general camaraderie. These students include Morgan Baldwin, Brian Griffin, Oscar Murillo, and Branden Rademacher.

One special acknowledgment must go to the original Apollo guidance designers. One would think they were working in the equivalent of the guidance stone age, but they made some remarkable engineering achievements considering the tools they had available to them. It is remarkable that the guidance system was successful to the extent that it was. While their work was not used in any direct fashion for this algorithm, it provided a useful basis for the development.

Thanks to my family and friends for their support with special thanks to my wife, Erin Myers Brunner for her love, appreciation, and continued confidence.

ABSTRACT

A numerical predictor-corrector (NPC) method for trajectory planning and closed-loop guidance of low lift-to-drag (L/D) ratio vehicles during the skip entry phase of a lunar-return mission is presented. The strategy calls for controlling the trajectory by modulation of the magnitude of the vehicle's bank angle. The magnitude of the bank angle used in the skip phase is determined by satisfying the downrange requirement to the landing site. The problem is formulated as a nonlinear univariate root-finding problem. Full three degree of freedom (3DOF) nonlinear trajectory dynamics are included to achieve high accuracy of the landing prediction. In addition, the proposed approach automatically yields a direct entry trajectory when the downrange is such that a skip entry is no longer necessary. The same algorithm repeatedly applied on-board in every guidance cycle realizes closed-loop guidance in the skip entry phase. A number of issues are identified and addressed that are critical in closed-loop implementations. Extensive 3DOF dispersion simulations are performed to evaluate the performance of the proposed approach, and the results demonstrate very reliable and robust performance of the algorithm in highly stressful dispersed conditions. Comparison is made between the proposed algorithm and an earlier skip algorithm developed for the Apollo space program. It is shown that the proposed algorithm is superior to the Apollo algorithm especially when used for entries with long downranges.

CHAPTER 1. INTRODUCTION

1.1 Atmospheric Reentry

With the development of space travel, significant emphasis has been placed on transporting astronauts into space. However, as of this writing, no permanent space colonies have been developed, thus requiring the return of the astronauts to Earth. Reentry is a dangerous part of spaceflight as recent developments have shown. Several Russian Soyuz missions (TMA-1, TMA-10, and TMA-11) missed their landing site by a large distance and potentially endangered the safety of the cosmonauts onboard those vehicles. A typical spacecraft reentry will attempt to use some of the vehicle's aerodynamic capability to lessen the loads the crew is subject to and also reduce the heat spike on the capsule. Aerodynamic lift provides critical maneuverability for the vehicle as well. These particular Soyuz missions used a backup form of reentry guidance whereby strictly a ballistic mode was used. The ballistic mode is designed to insure landing, but this does not occur at the same landing site as the aerodynamic mode would have used. Soyuz TMA-1 had some kind of computer problem [1] causing it to land over 500 km from the expected landing site. Soyuz TMA-10 had a frayed wire which caused the ballistic mode to activate [2] missing the landing site by 340 km. The latest failure in Soyuz TMA-11 is still being studied, but it appears that a problem with capsule separation caused the backup entry guidance mode to activate. This vehicle missed the landing target by 420 km[3]. The cosmonauts in this mission may have been subject to up to eight times the force of gravity for a period of time[3]. Fortunately in all these missions, the astronauts were not seriously injured, but they had to wait for ground personnel to arrive to assist with operations. Accurate and safe reentry guidance must be emphasized as a crucial component of space mission engineering and design.

1.2 Background

For entry vehicles with relatively low lift-to-drag (L/D) ratios, a known strategy since the Apollo era for achieving long downrange is to allow the vehicle to skip out of the atmosphere[4, 5, 6, 7, 8, 9, 10, 11]. Figure 1.1 shows an illustration of this concept with the first entry, skip, then reentry of the atmosphere and final entry phase to the landing site. The atmospheric exit condition of the skip phase, including downrange, velocity, and flight path angle, must be such that they lead to correct entry conditions for the final phase for successful landing. The two essential functions of the entry guidance system in the skip phase are to plan the skip trajectory and provide guidance commands to ensure the satisfaction of the required conditions at the atmospheric exit. Given the critical importance of the skip trajectory and the relatively limited maneuverability of such vehicles, early in the Apollo program it was already realized that the skip trajectory would need to be designed on-board based on the actual flight condition.

Due to severely limited on-board computation capability at the time, the Apollo skip entry trajectory planning algorithm had to rely on a number of approximations and empirical equations to arrive at analytical expressions that relate the bank angle magnitude to the exit condition of the skip trajectory[4, 5, 12]. The corresponding downrange in the skip phase is also obtained by approximate analytical equations. From a particular skip exit state, the range in the Kepler phase is computed analytically based on Keplerian orbit theory. The downrange in the final entry phase is estimated by using the adjoints to the linearized trajectory dynamics as the sensitivity coefficients. In this way, the total downrange is determined as an approximate analytical function of the bank angle magnitude in the skip phase. A secant iteration scheme is employed to find the required bank angle magnitude. Once such a reference trajectory is found, the Apollo guidance system commands the bank angle to track the reference velocity and rate of altitude profiles by a linear guidance law. The various approximations and linearization assumptions have been found to limit the accuracy of the guidance, especially in the case where the downrange is relative long in this work and others[9, 10]. This inaccuracy can have a significant adverse impact on the success of the mission. For instance, for the class of entry vehicles such as Apollo and Orion Crew Exploration Vehicle (CEV), the maximum variations

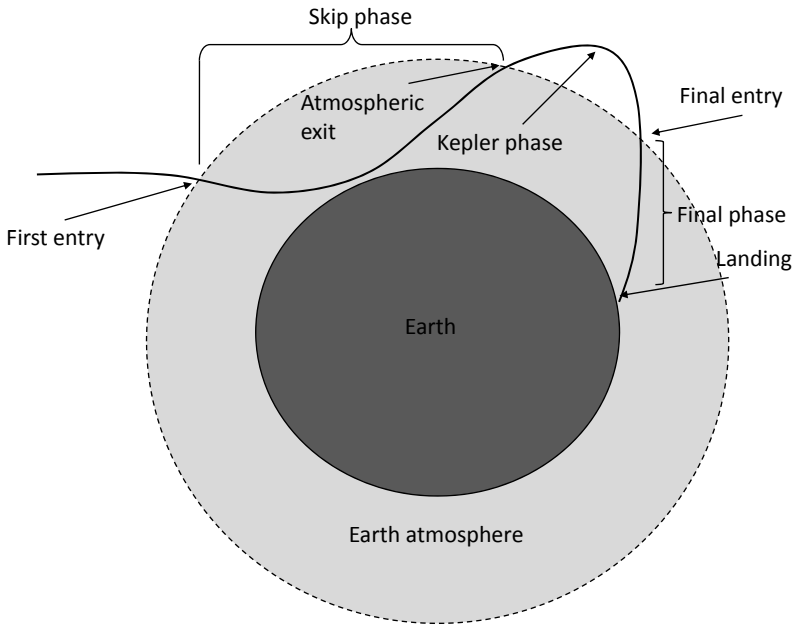

Figure 1.1 Skip entry trajectory for lunar return mission.

in the downrange for a nominal entry mission that the vehicle can accommodate in the final phase are on the order of ±370 km (200nm) due to the limited L/D ratio. Significant errors arising from incorrect atmospheric exit conditions in the skip phase could render the vehicle unable to satisfy the remaining range requirement in the final phase.

Improvement is sought in the guidance, and the literature provides many examples of guidance methodology. For high lifting vehicles, the Space Shuttle guidance provides a mature and well-tested method of tracking a pre-computed drag profile [13]. For low lifting vehicles, various other methods have been suggested for atmospheric guidance. For direct entry, one method is to extend the Space Shuttle entry guidance approach to track a pre-computed drag profile[14, 15, 16, 17, 18]. Modern computer capabilities permit generation of a feasible trajectory onboard the vehicle and use of a similar Shuttle-like tracking technique [27]. Analytic approaches using asymptotic expansions and optimal control techniques have been attempted for skip entry and aeroassist maneuvers[6, 8, 19, 20]. With the advent of flight computers,

onboard calculation of a trajectory has become viable through the use of numerical predictor-corrector methods[9, 10, 11, 20, 21, 22, 23, 24, 25, 26]. These methods are not limited by the validity of the simplifications, approximations and empirical assumptions necessary for any analytical treatment, thus they hold greater potential to be more accurate and adaptive. In this paper, a numerical predictor-corrector approach motivated by similar principles is developed for the skip entry problem.

The numerical method presented in this paper is for both purposes of trajectory planning and closed-loop guidance for a low L/D vehicle during the skip entry phase. In the trajectory planning, a linear bank angle magnitude profile is sought in skip phase and a specified constant bank angle (e.g. 70°) is flown in the final phase. The trajectory from the current condition in the skip phase to the end of the final phase is numerically simulated. The initial magnitude of the bank angle used in the skip phase is determined by satisfying the downrange requirement to the landing site. The search for this required initial bank angle magnitude is formulated as a nonlinear univariate root-finding problem. Unlike the Apollo skip-phase trajectory planning approach, this method does not rely on approximations and empirical equations. Therefore, full 3DOF nonlinear trajectory dynamics, including the effects of Earth rotation can be used in the trajectory planning without any difficulty for achieving high accuracy of the landing prediction. In addition, the proposed approach automatically yields a direct entry trajectory, without a skip out of the atmosphere by the vehicle, when the downrange is such that a skip entry is no longer necessary.

When the same algorithm is used in every guidance cycle update during the skip phase, the solution effectively provides a closed-loop bank angle magnitude command. No additional tracking guidance law is needed. The sign of the bank angle is determined by bank reversal logic that keeps the crossrange within an envelope and steers the vehicle to the landing site. Closed-loop guidance in the final phase is provided by another robust numerical predictor-corrector method specifically developed for final phase entry guidance.[24]

The performance and robustness of the proposed algorithm also hinge on several critical issues. In the planning algorithm and the guidance logic, the transition from one phase to

another must not cause difficulty in the next phase. Thus, the phase transition logic is carefully devised. The impact of the transient effects of bank reversals in the skip phase proves to be significant on the subsequent trajectory. Thus bank reversals should be executed in the direction of minimizing the time of reversals during the skip. It is found very helpful in achieving required high precision that first-order fading memory filters are used to continuously shape the lift and drag acceleration profiles in the planning and guidance algorithm. This is particularly so in the presence of large atmospheric and aerodynamic uncertainties. For long skip trajectories, Earth rotation tends to cause a final phase crossrange larger than the vehicle can recover from, mainly in the presence of large dispersions with high drag and low lift. Taking advantage of the planning algorithm, an automated targeting bias logic is developed for the skip phase to compensate for such an occurrence.

Extensive 3DOF dispersion simulations using a capsule vehicle model similar to that of the Orion CEV are conducted for several lunar-return mission scenarios with a wide spread of initial downranges. The dispersions include those in entry condition at the first entry interface due to de-orbit condition dispersions, aerodynamic coefficients, vehicle mass, and atmospheric density. The end-to-end performance of the algorithms in guiding the vehicle through skip entry to the end of the final phase entry flight is evaluated. In the 70,000 dispersed simulations, 99.857% of the trajectories end within 2.5 km to the landing site under highly stressful dispersed conditions. The test results establish the viability and promising potential of this skip entry trajectory planning and guidance approach.

The original Apollo skip guidance algorithm is used in a comparison study with the proposed algorithm. To assess whether the Apollo skip guidance or the Apollo final phase guidance is the most problematic in achieving high landing precision and worthy of improvement, the various guidance sections are inter-mixed. For instance, the Apollo skip guidance is tested with a numerical final phase and the proposed skip algorithm is tested with the Apollo final phase. Different downranges and landing sites are tested to determine which algorithm performs well.

1.3 Research Objectives and Dissertation Overview

This dissertation describes the development of a numerical predictor-corrector algorithm for skip reentry. It provides a comparison of the Apollo skip guidance with the newly developed numerical predictor-corrector algorithm. This work seeks to improve upon the Apollo algorithm. The primary objective of this work is to devise an algorithm that guides the vehicle to the parachute deployment condition above a landing site. Very high accuracy is sought under stressful dispersion conditions. Secondary objectives include: 1) maintain g-loads within safe limits for crew and vehicle 2) maintain heating loads within safe limits for vehicle heat shield 3) minimize fuel usage during vehicle maneuvering. Though these secondary objectives are not directly addressed in this particular work, some possibilities for their achievement are outlined in Chapter 9.

Chapter 2 discusses how a capsule is controlled and how the motion of a reentry vehicle is simulated in a numerical environment. Chapter 3 discusses the baseline skip guidance algorithm originally designed for the Apollo program. Chapter 4 discusses the new numerical predictor-corrector algorithm for skip guidance. Chapter 5 discusses some critical issues that arise in close-loop guidance using the numerical predictor-corrector that must be addressed and in place for accurate guidance. Chapter 6 discusses the method by which the algorithms are evaluated using specific vehicles and Monte-Carlo dispersion simulations. Chapter 7 gives extensive results of testing the proposed numerical-predictor algorithm. Chapter 8 compares the results of simulations run with the new algorithm compared with the Apollo skip guidance. It also merges the skip phase of the new algorithm with the Apollo final phase guidance and then the Apollo skip guidance with the numerical final phase guidance developed by Lu [24]. Chapter 9 gives some concluding remarks and potential future work for improvement.

CHAPTER 2. PROBLEM FORMULATION

2.1 Bank Angle Modulation

For a blunt body like a capsule, generation of lift is accomplished by offsetting the center of gravity (CG) from the center of pressure (CP). This is unlike a vehicle with wings, which generates lift by a difference in pressure between the top and bottom surface. The CP is in a fixed location dictated by the geometry of the vehicle. However, by adjustment of internal components the CG can be moved so that in a trim condition, there is no rotation about the CG in the longitudinal direction and we have the equilibrium condition, Eq. (2.1). The CG is fixed in position once the design of the vehicle is completed; no active control of the CG is performed during entry flight. The angle of attack is α. A schematic version of the lift generation appears in Fig. 2.1.

$$\Sigma M_G = nF_N - aF_A = 0 \tag{2.1}$$

To control the vehicle's downrange flight, the vehicle is rolled to cause lift out of plane as in Fig. 2.2. By altering the bank angle, the vertical component of lift is made to vary. A smaller bank angle produces a greater component of lift in the vertical plane and enables the vehicle to remain aloft a greater distance. This effect is similar to introducing a slip into the flight of an aircraft. A larger bank angle, even to the limit of pointing the lift vector towards the ground, will reduce the vertical lift and shorten the distance aloft.

A horizontal component is coupled to the motion in the vertical plane. As the total magnitude of lift remains constant, the lift generated out of the vertical plane is seen in the horizontal plane. This has the effect of altering the vehicle's heading angle and causing a turn. Theoretically, the vehicle could fly a constant bank angle, with the perfect amount of lift to carry it

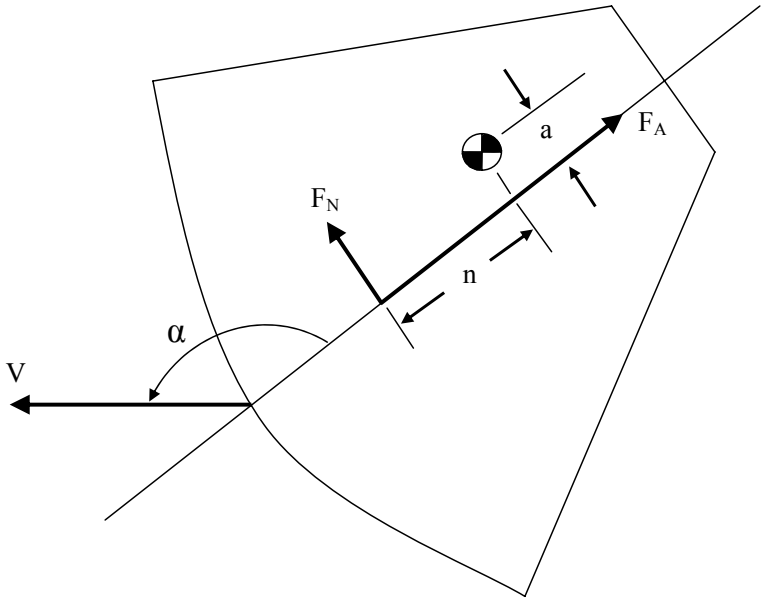

Figure 2.1 Generation of lift by CG offset from CP

to the landing site downrange of its entry point. However, the crossrange would not be zero. Thus, the vehicle occasionally needs to reverse the bank angle from positive to negative or vice versa to null out the cross range. During the finite time in changing the bank angle from one side to the other, some time is spent with the vehicle bank angle at a point that will not satisfy the downrange requirement. Thus, the vehicle's bank angle is continuously monitored and modified to ensure that it will satisfy all the both downrange and crossrange requirements.

2.2 Entry Vehicle Dynamics

To simulate a spacecraft's entry flight, the dimensionless equations of three-dimensional motion for a point mass about a rotating Earth are integrated. The equations are dimensionless for purposes of numerical conditioning. These equations are: [28]

$$\dot{r} = V \sin\gamma \tag{2.2}$$

$$\dot{\theta} = \frac{V \cos\gamma \sin\psi}{r \cos\phi} \tag{2.3}$$

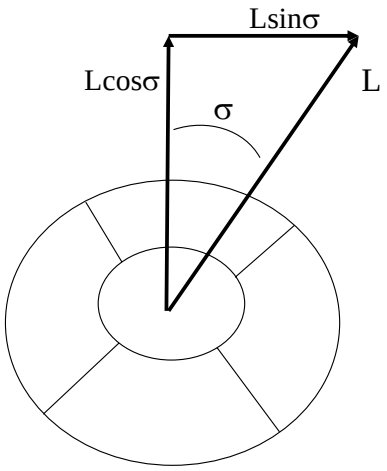

Figure 2.2 Modulation of bank angle by rolling

$$\dot{\phi} = \frac{V\cos\gamma\cos\psi}{r} \tag{2.4}$$

$$\dot{V} = -D - \frac{\sin\gamma}{r^2} + \Omega^2 r\cos\phi(\sin\gamma\cos\phi - \cos\gamma\sin\phi\cos\psi) \tag{2.5}$$

$$\begin{aligned} V\dot{\gamma} &= L\cos\sigma + (V^2 - \frac{1}{r})\frac{\cos\gamma}{r} + 2\Omega V\cos\phi\sin\psi \\ &+ \Omega^2 r\cos\phi(\cos\gamma\cos\phi + \sin\gamma\cos\psi\sin\phi) \end{aligned} \tag{2.6}$$

$$\begin{aligned} V\dot{\psi} &= \frac{L\sin\sigma}{\cos\gamma} + \frac{V^2\cos\gamma\sin\psi\tan\phi}{r} - 2\Omega V(\tan\gamma\cos\psi\cos\phi - \sin\phi) \\ &+ \frac{\Omega^2 r\sin\psi\sin\phi\cos\phi}{\cos\gamma} \end{aligned} \tag{2.7}$$

with

$$D = \rho(V_s V)^2 S_{ref} C_D/(2mg_0) \tag{2.8}$$

$$L = \rho(V_s V)^2 S_{ref} C_L/(2mg_0) \tag{2.9}$$

The differentiation is with respect to a dimensionless time.

$$\tau = t/\tau_s \tag{2.10}$$

with

$$\tau_s = \sqrt{R_0/g_0} \tag{2.11}$$

where, r is the distance from the center of the Earth, θ the longitude, ϕ the latitude, V the atmospheric-relative velocity, γ the flight path angle of the relative velocity, ψ the heading angle of the relative velocity vector measured from north, positive in a clockwise direction, and σ the bank angle measured positive to the right from the view inside the vehicle. The Earth rotation rate is Ω, S_{ref} the reference area, ρ the local atmospheric density, D the drag force, L the lift force, C_D the coefficient of drag, C_L the coefficient of lift, m the spacecraft mass, and g_0 the acceleration due to gravity at the Earth's surface = 0.00981 km/s^2. Note that r is scaled by the Earth's radius, $R_0 = 6378.135$ km, and V is scaled by the dimensionless velocity:

$$V_s = \sqrt{R_0 g_0} \tag{2.12}$$

Thus, the term V_s is included in Eqs. (2.8)-(2.9) to redimensionalize the velocity term. Then the division by the weight of the vehicle gives the respective drag and lift forces in multiples of gravitational acceleration, commonly known as g's. One g of acceleration is equal to the acceleration due to gravity at the Earth's surface.

In the planning of the trajectory, the 1976 U.S. Standard Atmosphere is used as the atmospheric model.[29] The equations of motion are integrated with a fourth order Runge-Kutta method.

Initial entry conditions are assumed to be those on a lunar return trajectory. The entry altitude is 121.9 km (400,000 ft). The velocity, flight path angle, heading, latitude, and longitude are assumed to have been targeted via earlier de-orbit maneuvers to bring the spacecraft to a point and heading that permit the guidance to ensure a safe entry. Specific cases are described further in Chapter 7. A pre-specified final velocity, V_{final}, terminates the simulation, which is targeted toward a landing site. For the purposes of the simulation, when the vehicle has reached 150 m/s (500 ft/s) relative velocity, parachute deployment will occur and the simulation for entry flight terminates.

To control the trajectory, the bank angle is varied to achieve pre-specified terminal constraints. The bank angle is tied into the flight path angle via Eq. (2.6) and the heading angle via Eq. (2.7). It is responsible for achieving the longitudinal and lateral constraints. The bank

Table 2.1 Entry variables

State	Symbol	Dimensions
Altitude	none	km
Radius	r	none
Longitude	θ	radians
Latitude	ϕ	radians
Relative Velocity	none	km/s
Relative Velocity	V	none
Flight Path Angle	γ	radians
Heading Angle	ψ	radians
Bank Angle	σ	radians
Drag	D	g's
Lift	L	g's
Density	ρ	kg/km^3
Reference Area	S_{ref}	km^2
Drag Coefficient	C_D	none
Lift Coefficient	C_L	none
Mass	m	kg
Time	t	seconds
Dimensionless Time	τ	none

angle is determined by the guidance algorithm, which is described in Chapter 4.

To summarize the variables and their dimensionality, see Table 2.1. The constants are in Table 2.2. For example, the inital altitude, typically given in kilometers, is converted to the Earth centered distance by adding the Earth radius to it. Then this value is divided by the Earth radius to give the dimensionless radius, r. Similarly, if velocity is given in kilometers per second, this value is divided by the scale velocity, V_s to give the dimensionless relative velocity, V. Also, the Earth rotation rate in radians per second is multiplied by the scale time, τ_s to give the dimensionless rotation rate, Ω.

Table 2.2 Entry constants

State	Symbol	Value	Dimensions
Earth Rotation Rate	none	7.2921151e-5	radians/s
Earth Rotation Rate	Ω	0.05879	none
Gravity	g_0	0.00981	km/s^2
Radius of Earth	R_0	6378.135	km
Scale Velocity	V_s	7.9100	km/s
Scale Time	τ_s	806.329	seconds

CHAPTER 3. APOLLO SKIP GUIDANCE

For the Apollo program, the reentry guidance sought to use a skip reentry. The stated reasons were for mission flexibility and for trajectory flexibility. Mission flexibility refers to adjusting to different reentry conditions and target locations. For example, depending on the relative orbital positions of the Earth and Moon, the vehicle might be on different approach angles and headings. Also, if the original landing site were unavailable due to weather or lack of recovery crew, the landing site could be changed. Trajectory flexibility refers to achievement of entry objectives in spite of atmospheric and vehicle uncertainties [4]. At the time of Apollo, engineers did not have significant heritage in entry vehicle performance. Similarly, knowledge of the atmosphere was limited.

3.1 Guidance Design

Fig. 3.1 will be referenced in the following simplified overview of the Apollo skip guidance. Fig. 3.2 gives a more detailed overview of the guidance in relation to the flight condition. The original design of the Apollo skip guidance appears in reference [12], though reference [4] contains a more recently updated and detailed version of the guidance algorithm. Reference [4] contains complete derivations and flowcharts for the entire guidance algorithm. However, certain gains and constants were unavailable in that document. Reference [9] lists these constants and gains, but they appear to be from the earlier version, reference [12] of the Apollo skip guidance. For this work, the gains and constants are used from a more recent document[30]. Reference[5] contains some information concerning the actual performance of the guidance system during the Apollo 10 mission. However, the skipping capabilities of the guidance were never utilized during that particular mission, nor any other Apollo Mission.

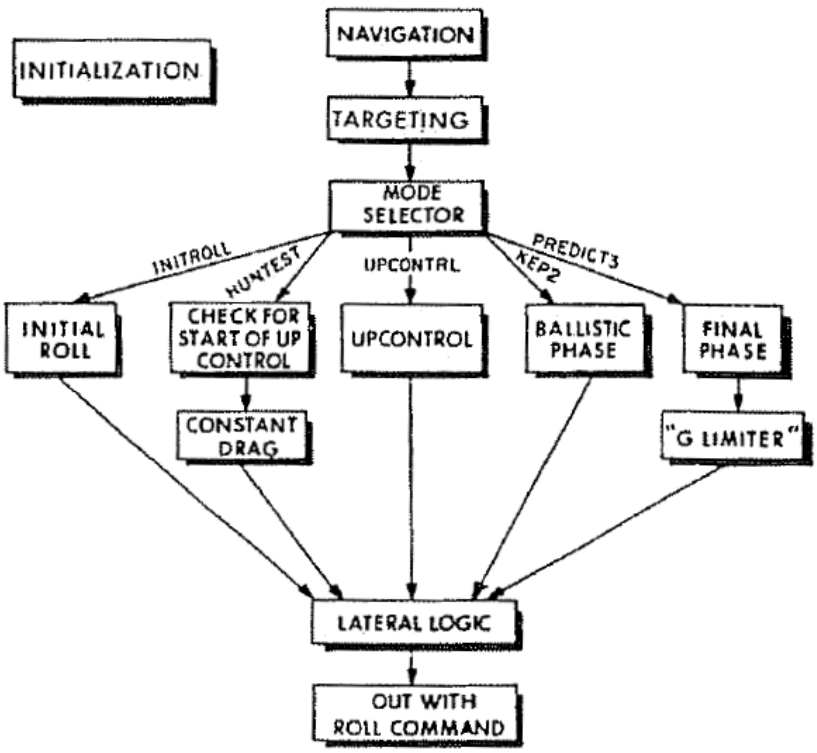

Figure 3.1 Guidance flowchart for Apollo skip algorithm[4].

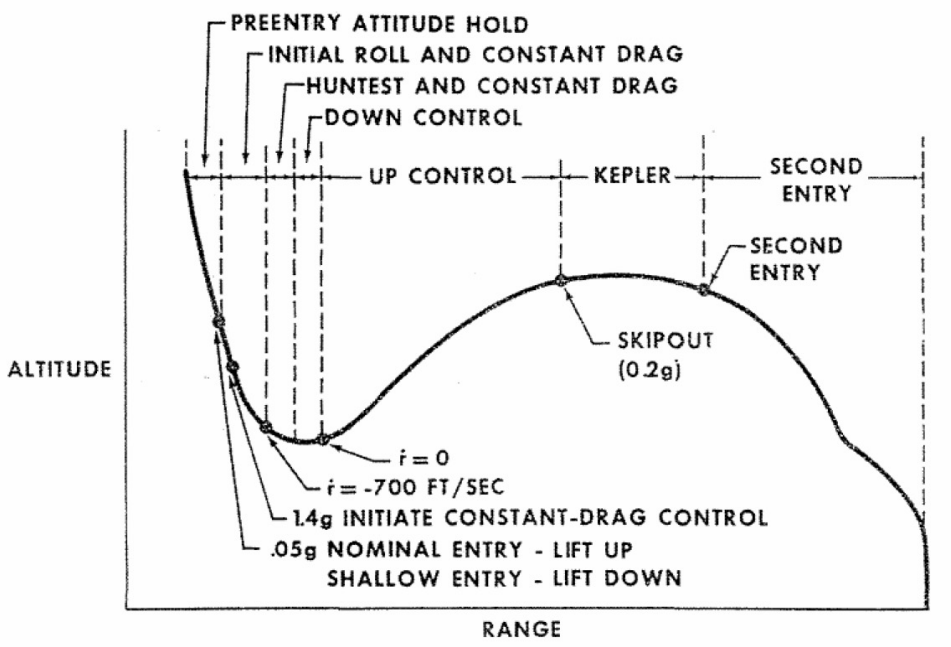

Figure 3.2 Reentry guidance phases for Apollo skip algorithm[5].

The Apollo skip guidance is designed in a modular fashion whereby one part of the code is executed until completion as shown in the figure. Certain sections such as navigation, targeting, mode selector, and the lateral logic are executed during every call to the guidance. The Apollo guidance accomplishes two objectives. First, a trajectory is designed that will reach the landing site. Second guidance control laws are used to fly the designed trajectory while not exceeding certain constraints on vehicle load.

When the guidance is called the first time, an initialization routine is run. This sets up constants for the guidance that will be used and initializes the switches. It is only called once. The navigation routine takes data from the onboard vehicle accelerometers. This data is used to propagate the vehicle's position and velocity vectors forward in time. The targeting routine is called to update the vectors from the vehicle to the target site.

Next, the mode selector is run to decide which phase of guidance the vehicle is in. There is a parameter called selector which is initialized during the initial roll subroutine. This parameter is retained in memory and updated as the trajectory progresses. This begins the process of

designing a trajectory to-be-flown and then actually flying it.

The initial roll subroutine calculates the reference drag level for the later constant drag phase. It also determines an initial orientation, either lift up or lift down, for the vehicle during the open-loop phase of flight. Once the initial pass has been made and the drag level passes a threshold value of $1.4g_0$ as seen in Fig. 3.2, the closed-loop constant drag phase is run. This phase is only run when the drag is high enough. Because the guidance relies on measuring the drag when the vehicle is at the top of the sensible atmosphere where the density is highly variable. The guidance might produce bank commands that are correspondingly highly variable. This continues until the vehicle has slowed sufficiently and the trajectory shallows out. Then, it will begin the Huntest phase of closed loop planning. This occurs once the altitude rate is greater than -700 ft/s as shown in Fig. 3.2.

The Huntest phase is complex depending on the trajectory of the vehicle through different branches. The main functions are: 1) calculate the vehicle state conditions at the pullout point (bottom of the skip), 2) calculate the exit conditions of the upcontrol phase, and 3) calculate the required L/D value during the upcontrol that will permit the vehicle to achieve those exit conditions. A reference trajectory is calculated taking the vehicle to the pullout point and the entire upcontrol phase. The complete derivation and equations are available [30]. During Huntest, if the calculation of the vehicle's exit velocity is greater than the circular satellite velocity, or if the total range to-be-flown on the reference trajectory is an overshoot, it will continue to run the constant drag segment. If there is an undershoot, the required L/D value is updated using a secant method. This process continues until the predicted range-to-be flown is within 25 nautical miles of the target.

The constant drag module calculates the L/D ratio of the vehicle required to track a reference drag level and altitude rate. This is accomplished with a linear control law. It is used once vehicle controllability is assured and during Huntest if the vehicle is expected to overshoot the targeted site.

The upcontrol phase produces vehicle L/D commands to achieve the conditions at the pullout point and the atmospheric exit. Typically the vehicle is still going down when the

Huntest phase has converged, so it flies a modified constant drag profile to reach the pullout point. This is labeled "Down Control" on Fig. 3.2. Once past the pullout, a control law is used to generate L/D commands so the vehicle flies to the atmospheric exit conditions. This continues until the drag decreases to be considered in a Kepler phase. As shown in Fig. 3.2, this occurs at $0.2g_0$. If a loft trajectory were to occur where the vehicle does not fully exit the atmosphere, the final phase guidance will begin.

During the ballistic or Kepler phase, the vehicle simply maintains a neutral position with zero bank angle or full lift up. This continues until the drag starts to increase again. With the drag greater than $0.2g_0$, the final phase begins at the second entry point.

The final phase guidance is based on a precomputed reference trajectory. A control law is used to track the reference trajectory and produce bank angle commands for the vehicle. The reference trajectory requires a table look-up and interpolation with vehicle velocity as the indenpendent variable to produce the correct values for the control law. During final phase, if the landing site is passed, full lift down is commanded.

During the final phase, large bank magnitudes can produce very high g-loads. A limitation on the g-load is introduced to prevent exceeding the maximum vehicle capability. If the load is less than half of the maximum permitted g-load, the computed bank angle is flown. When between half the maximum g-load and the maximum g-load, a further computation involved the altitude rate ($\dot{r}$) is done. Depending on this calculated value, the computed bank angle may be flown or full lift up may be commanded (0 deg bank). If the maximum g-load is exceeded, full lift up is commanded. In the Apollo guidance, the value of the maximum permitted g-load is 8 g's. However, it is stated that this value is chosen for safety, since the true maximum g-load is 10 g's.

Following computation of the required L/D value in the guidance phases where control is possible (Constant Drag, Huntest, Down Control, Up Control, and Final), the guidance computes the sign of the needed bank angle using the lateral logic. This computation requires the crossrange of the vehicle to lie within a corridor. Bank reversals are commanded whenever the crossrange exceeds this velocity-dependent corridor. The crossrange guidance also prevents

the guidance from flying a bank angle within 15° of full lift up or down. This is to prevent inadequate nulling of crossrange error in the event the guidance commands a bank angle near full lift up or down. There are also conditions that prohibit roll reversals entirely. Normally, the autopilot will roll in the shortest direction, but during high load portions of the upcontrol phase, the vehicle is prevented from rolling through lift down.

3.2 Modifications for Current Use

For testing and comparison, the Apollo skip guidance is implemented in modern software (MATLAB). The Apollo skip guidance required a number of modifications to the original Apollo code for use. The Apollo code is modular in that each phase is designed to work independently, but still needs to access to several variables from other modules. With repeated function calls to the Apollo guidance in the code, a number of variables are stored for use in future calls.

Next, the navigation subroutine is not used, since the simulation has direct access to the exact state of the vehicle. As discussed in Chapter 6, perfect knowledge of the state was assumed, whereas the navigation subroutine only assumes access to information from onboard accelerometers. With modern Global Positioning System (GPS) data, as discussed in [10], greater knowledge of vehicle state information is expected.

Next, the initialization and targeting subroutines perform a computation to determine the range to the target based on the estimated time-to-go and earth rotation angle. Since the vehicle velocity, V is already Earth-relative as in Eq. (2.6), the range-to-go is computed by computing the arc length as in Eq. (4.4).

Bairstow mentions an error in the lateral logic routine (called "310" in Moseley [4]), that is corrected [9]. The calculation for Y, (equivalent to χ_c in Eq. (4.8)) omits a conversion of the variable LATBIAS from nautical miles to radians.

Also, the digital autopilot (referred to as DAP in much of the Apollo literature) was not available, so the bank reversal was simulated with a separately written code to ensure bank reversal in the shortest direction. This is also used in the proposed algorithm with further

detail in Section 5.2.

3.3 Difficulties with Apollo

Throughout the Apollo skip guidance, numerous empirical relations and approximations are made. Extensive study was made in [30] to show their development. However, it has been shown in more recent work [9] and will be shown in this work that the Apollo guidance is not accurate under longer range entries, with a wider range of entry conditions, and with atmospheric, vehicle, and entry dispersions.

Specifically in Bairstow, two things were shown that cause inaccuracy at longer ranges [9]. First, it was shown that the upcontrol phase guidance law does not properly guide the vehicle to the calculated exit conditions. This was shown by examining the transition from the skip phase to the final phase. Bairstow defined an "energy bucket". The bucket gave a corridor through which the vehicle's combined range-to-go and total energy (a combination of altitude and velocity) must lie. The bucket was found by evaluating the vehicle's range-to-go and the potential range that could be covered using either full lift up for the remainder of the trajectory, or full lift down subject to a 10-g limit. Bairstow found that for longer ranges, the Apollo final phase transition occurred with too much range-to-go and too little energy. This led to studying the exit conditions for the upcontrol phase, since no control occurred during the Kepler phase.

Bairstow found that during the last part of upcontrol, the vehicle was flying only moderate bank angles leaving it with plenty of capability. Thus, the guidance was unaware of its predicament of having too much range-to-go with too little energy. The upcontrol control law tracks a reference altitude rate and a reference velocity. Bairstow found that although the velocity was being tracked closely, the altitude rate was not. Since in the equations of motion, specifically Eq. (2.2), the velocity was not in error, but the altitude rate was, this meant that the flight path angle was significantly in error. This error caused a significantly different Kepler phase to be flown for the longer trajectory, and prevented the final phase guidance from having any chance of succeeding.

In a further study, Bairstow analyzed whether the reference trajectory via the exit conditions were even valid. In the derivation of the range prediction equations for the Apollo skip guidance, a purely ballistic orbit is assumed for the Kepler phase. At the altitudes encountered, drag plays a small, but significant role in modifying the vehicle's path. Bairstow showed this by using numerical simulation of the trajectory through Kepler phase from the calculated upcontrol exit conditions. It was found that during longer range skips, accumulated drag slows the vehicle and causes the undershoot.

One additional point it that the sensitivity of the trajectory must have been realized by the original Apollo skip guidance designers because roll reversals are not allowed during upcontrol while the drag is greater than 4.3 g's.

CHAPTER 4. ALGORITHM DESCRIPTION

4.1 Trajectory Planning

4.1.1 A Root Finding Method

In the planning of the complete entry trajectory from skip to the end of the final phase, the magnitude of the bank angle in the skip phase is parameterized as a linear function of the range-to-go s_{to-go} until a specified threshold s_{thres} which is set equal to 2000 km. Below s_{thres}, a constant bank angle magnitude of $\sigma_f = 70^\circ$ is used. Figure 4.1 shows this bank angle profile which is continuous. Therefore the bank angle magnitude $|\sigma|$ at any range-to-go $s_{to-go} \geq s_{thres}$ is

$$|\sigma| = \sigma_f + (\sigma_0 - \sigma_f)\frac{s_{to-go} - s_{thres}}{s^0_{to-go} - s_{thres}}, \quad s^0_{to-go} \geq s_{to-go} \geq s_{thres} \tag{4.1}$$

where s^0_{to-go} is the current range-to-go. For a given bank angle magnitude, the sign is assigned by a bank reversal logic (more later). The corresponding skip trajectory is simulated on-board from the current condition to the end of the entry trajectory, defined by a specified velocity. Therefore, the skip trajectory planning problem is to determine the initial bank angle, σ_0, in the bank angle parameterization Eq. (4.1) that will meet the final condition on the miss distance to the landing site at the specified final velocity. Fig. 4.2 shows the planning transition logic.

The choice of s_{thres} in the bank angle parameterization is such that it represents typical downrange at the initiation of the final phase guidance. This value does not vary widely for low L/D vehicles. Thus the skip and Kepler phases take place from s^0_{to-go} to s_{thres}. Even though bank angle has no influence on the trajectory in the Kepler phase, one reason for the parameterization in Fig. 4.1 is to maintain continuity of the bank angle. This feature is

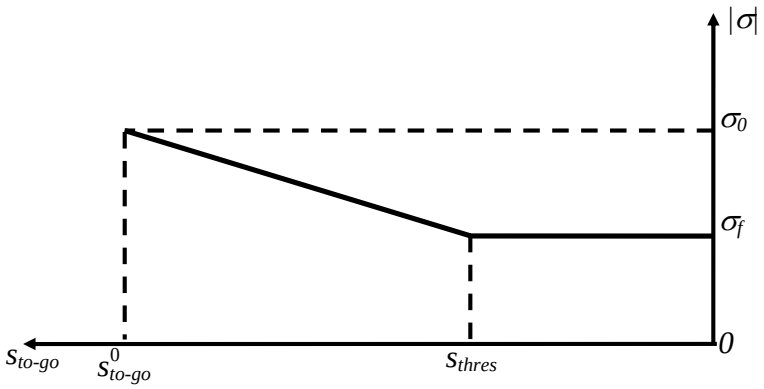

Figure 4.1 Bank angle parametrization with respect to range-to-go s_{to-go}

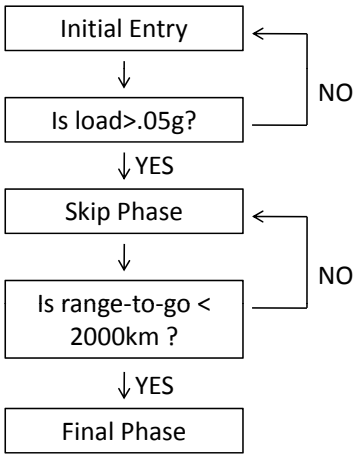

Figure 4.2 Phase transition logic for planning

important in ensuring the convergence of the planning algorithm when the initial downrange is short and the trajectory does not have full skip and Kepler phases. Another benefit of the linear profile in Fig. 4.1 is that the skip trajectory will deplete the excess energy early in skip phase (since the bank angle has larger magnitude). This tends to help alleviate the undesirable need to fly a very large bank angle toward the end of the skip phase, and increase the robustness of the guidance in the presence of severe atmospheric and aerodynamic uncertainties.

In the planning, the downrange covered is added as a seventh dimensionless equation of motion to integrate with the other variables as given before.

$$\dot{s} = V \cos\gamma / r \tag{4.2}$$

The planning algorithm attempts to find the initial required bank angle $|\sigma_0|$ so that the downrange flown along the complete entry trajectory is equal to the actual range-to-go to the landing site. Through the integration of the equations of motion (2.2)-(2.7) and (4.2) with $|\sigma|$ given by Eq. (4.1), the search for σ_0 becomes a root finding problem: at the current spacecraft state, find the σ_0 that will permit satisfying the downrange requirement:

$$s_{miss} = s^0_{to-go} - s_f(|\sigma_0|) = f(|\sigma_0|) = 0 \tag{4.3}$$

Through integration of the equations of motion, the range flown, s_f is found at the specified final velocity. The current range-to-go, s^0_{to-go} is found by computing the great circle distance from the current vehicle position to the landing site. The great-circle distance, s^0_{to-go} in radians between the current vehicle position (θ, ϕ) and the landing site (Θ^*, Φ^*) is:

$$\cos(s^0_{to-go}) = \sin\phi \sin\Phi^* + \cos\Phi^* \cos\phi \cos(\Theta^* - \theta) \tag{4.4}$$

The resulting distance, s_{miss} will be positive if the spacecraft undershoots the final position and negative if there is an overshoot. If strictly the range-to-go s^0_{to-go} computed at the final vehicle position in Eq. (4.4) is used, the resulting function in Eq. (4.3), will be discontinuous resulting in convergence difficulty. This stems from the fact the range-to-go at the end of the

integration may have a significant component resulting from non-zero crossrange. Thus the function in Eq. (4.3) is used.

A line search method is used to compute a search direction for the next bank angle in Eq. (4.5).[31]

$$x_{n+1} = x_n + \alpha_n p_n \tag{4.5}$$

The step length, α, is chosen to be one and the direction, p_n, is chosen with a secant method. This inexact line search is used to find the root or rather the exact bank angle that will permit the final position of the vehicle to be as close as possible to the desired final position. Due to the choice of the inexact step length, some difficulty is encountered in computing the update. This is described in more detail below. The search direction is found from Eq. (4.6).[31]

$$p_n = -B_n^{-1} \nabla f_n = -\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \tag{4.6}$$

Thus, the update equation becomes the well-known secant method which is used to iteratively solve Eq. (4.3). The iterates are generated by the Eq. (4.7).

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \tag{4.7}$$

where $x = \cos\sigma_0$. To start the secant iteration scheme appropriately for the first time, an initial guess of the bank angle is required that will enable the spacecraft to reach the final velocity following a skip trajectory. An arbitrary choice could result in the vehicle bouncing off or diving steeply into the atmosphere. To find this initial bank angle, the algorithm begins by assuming a bank angle of 0° in the numerical simulation of the trajectory. If the vehicle's altitude exceeds 300 km at any time, the spacecraft is assumed to have bounced out of the atmosphere. Then the bank angle magnitude is incremented by a fixed amount of 2.5°, and the process repeats until the corresponding trajectory reenters the atmosphere following the skip and ends in the final phase with an undershoot in the final range-to-go to the landing site.

The secant iteration stops when the miss distance to the parachute deployment condition is achieved, i.e., at specified final velocity of 150 m/s (500 ft/s), the miss is less than 25 km

(i.e., when $|s_{miss}| < 25km$). Note that to convert s_{miss} to kilometers, it must be multiplied by R_0 to give the correct units.

4.1.2 Bank Reversal

In order to actually fly toward the end conditions, the sign of the bank angle is determined by a lateral logic that is similar to the one used in the Apollo entry guidance.[4, 5] A crossrange threshold is defined as a linear function of the dimensionless velocity

$$\chi_c = c_1 V + c_0 \tag{4.8}$$

where c_0 and c_1 are constants set equal to 8.71e-5 rad and 5.21e-3 rad respectively for this work. Whenever the crossrange along the trajectory exceeds χ_c as defined above, the sign of the bank angle is reversed. In the trajectory planning, the change of the bank angle sign is achieved instantaneously without any rate or acceleration limits. If the rate and acceleration limits are enforced, the number of bank reversals could be different in two successive iterations of the secant search. This difference will result in a discontinuity in the miss-distance function f, which will in turn cause convergence difficulties in the secant method. Instantaneous bank reversal is done to avoid this possible problem. However, it should be stressed that the bank angle rate and acceleration limits are enforced in the guidance command generation. A sample plot of the crossrange variation over time is shown in Fig. 4.3. The reversal of the bank angle ensures the crossrange remaining with the necessary corridor. The corridor shrinks over time due to the decreasing velocity.

4.1.3 Issues in Continuity of Miss Function

The secant method requires that Eq. (4.3), be a continuous function. Several measures are undertaken to insure this is the case. Some of these are within the planning, others are in the root finding algorithm.

First, it was found that the phase transition between the skip phase and the final phase is especially important. Originally, the determination of final phase was via a complex sequence of events as shown in Fig. 4.4.

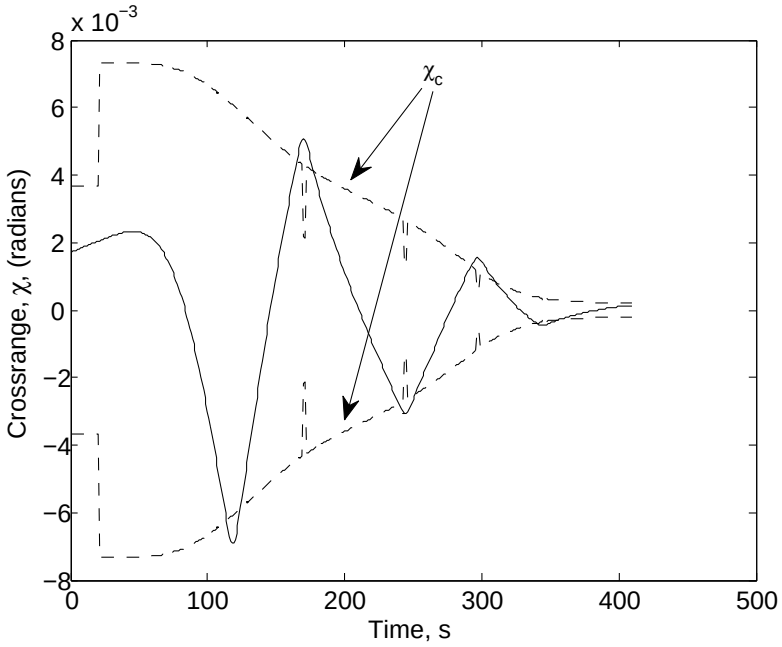

Figure 4.3 Crossrange variation

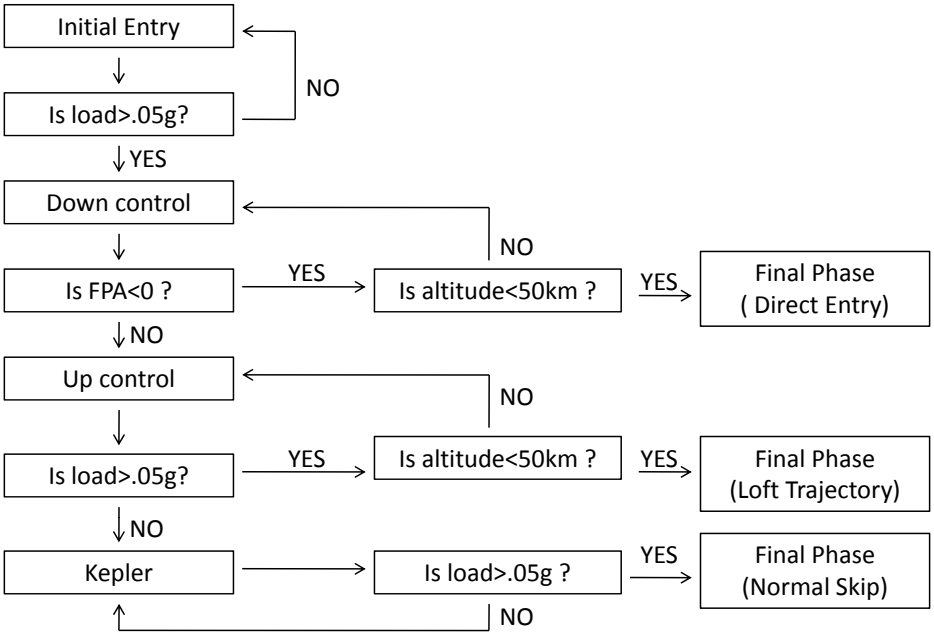

Figure 4.4 Phase transition logic with discontinuous planning

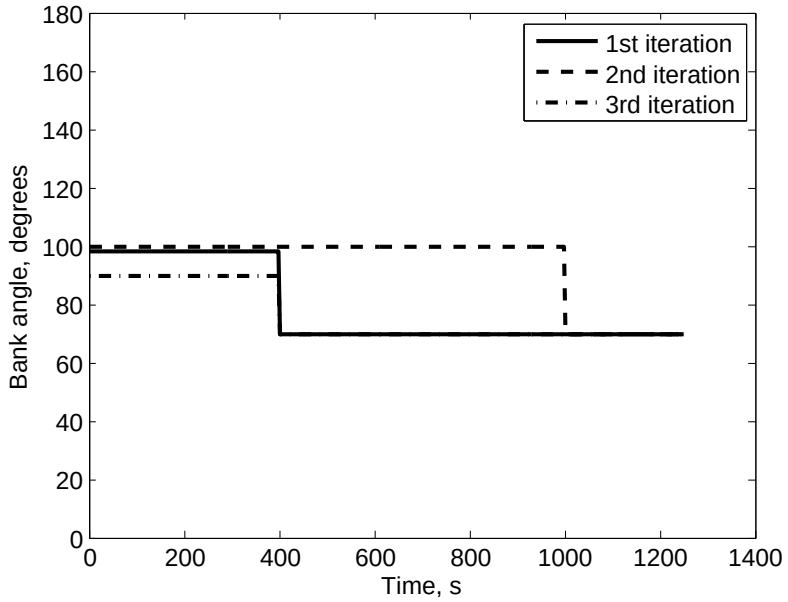

Figure 4.5 Bank angle magnitude histories with discontinuous planning

In a particular type of trajectory that is on the edge of having a Kepler phase (bordering the loft and normal skip paths) this type of transition logic is a problem. Depending on the bank angle that is planned during the skip phase, different paths will followed. Originally a constant bank angle was used during skip phase, but as this work developed, the linear profile was evolved. For this discussion, the constant bank angle method is reverted to. In an iteration of the secant method as shown in Fig. 4.5, a bank angle of about 100° is flown. This causes the profile of the trajectory to appear in Fig. 4.6.

Since the load decreases below 0.05g, the path of normal skip is followed and the final phase begins around 400s. Since the vehicle overshoots the target slightly (not shown in figures), the bank angle is increased via the secant method. The resulting trajectory follows that of Fig. 4.7. With this trajectory, the logic path followed is the loft trajectory whereby the final phase begins when the vehicle is below 50 km in altitude. The altitude level where the load is below 0.05g is approximate in the figure. With this path, the final phase begins significantly later than in the first iteration. During the early part of the second entry, the bank angle flown is much larger than it was with the first iteration. This affects the trajectory

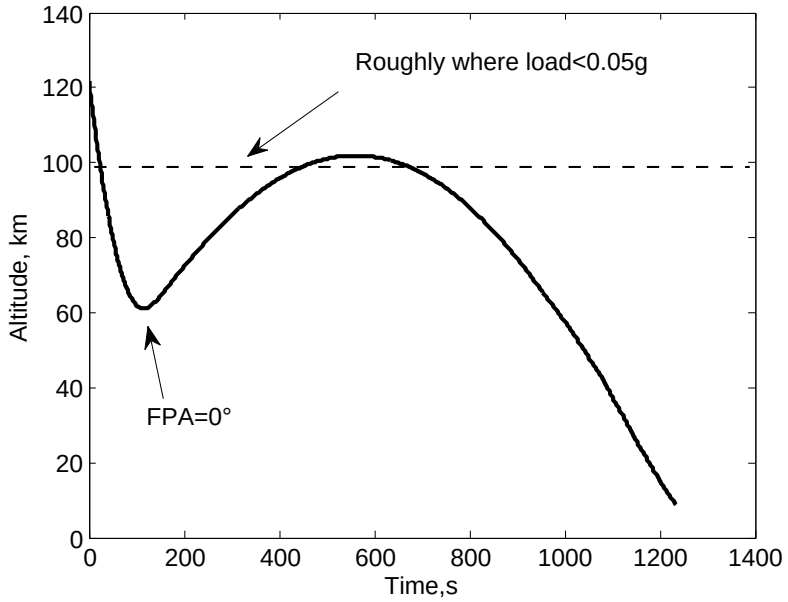

Figure 4.6 Discontinuous planning: time versus altitude-first iteration

so that it undershoots the landing site significantly more than it should have if the process was continuous. Thus, a third iteration follows with a bank angle significantly smaller than the first, creating a significant overshoot, and the problems continue as convergence is not possible. With the implementation of range-to-go as the variable to determine transition to final phase, these problems are eliminated. Since range-to-go is decreasing in a fashion that is common to all trajectory types, the final phase begins at nearly the same point.

One other idea was to use the vehicle energy to determine the start of final phase. The specific mechanical energy is calculated as in Eq. (4.9).

$$e = \frac{1}{r} - \frac{V^2}{2} \tag{4.9}$$

Using the dimensionless variables as before, the energy starts near zero and increases to one at the end. A value of the energy must be chosen to signify the transition to final phase during the planning. In using the planning logic above in Fig. 4.2 or Fig. 4.4, it is found that longer range trajectories transition to final phase with significantly more energy remaining than

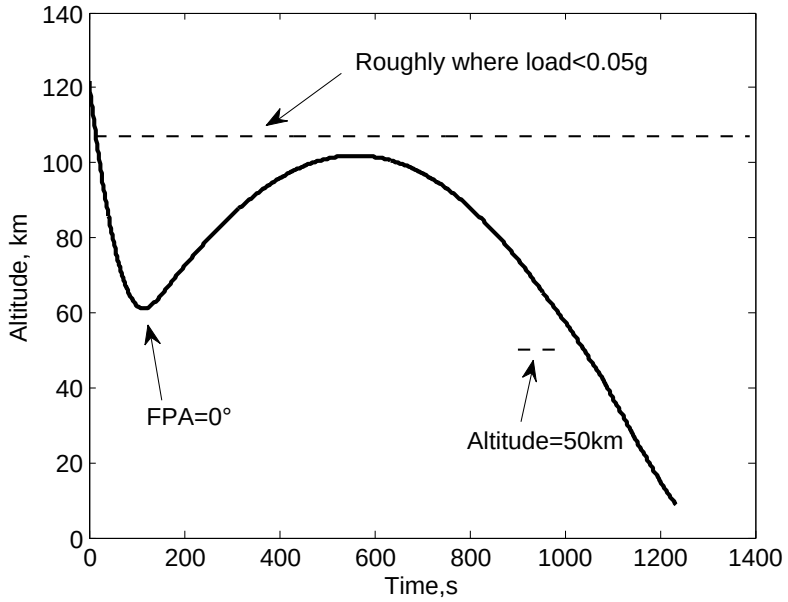

Figure 4.7 Discontinuous planning: time versus altitude-second iteration

shorter range. The longest trajectories should switch to final phase with energy about equal to 0.51. The short trajectories transition around 0.57. This is because the longer skip trajectories require less velocity reduction during the skip to reach the higher altitudes during the Kepler phase. Due to the large difference in needed threshold values, it was not possible to find one that satisfied all the different possible trajectories. If the transition value is set to 0.51, then the shorter range trajectories remain in skip phase too short a time (transition occurs early in the upcontrol phase). If the value is set to 0.57, then the longer range trajectories transition to final phase too late, until the altitude has decreased to less than 50km in the second entry. This does not leave enough margin for the final phase to reach the target. Fig. 4.8 gives the representation of the planning transition logic using energy.

Due to the sensitivity of the final condition, the normal integration step size, $h = 10s$ (or 0.01246 nondimensionalized) for the planning phase is adjusted so that the final velocity is met exactly to enable better comparison of the resulting ranges flown for the secant method. If $V + \dot{V}h$ (with $\dot{V}$ calculated from Eq. (2.5) with the current conditions) is less than or equal to V_{final}, then the step size is set equal to the value in Eq. (4.10). This procedure takes place

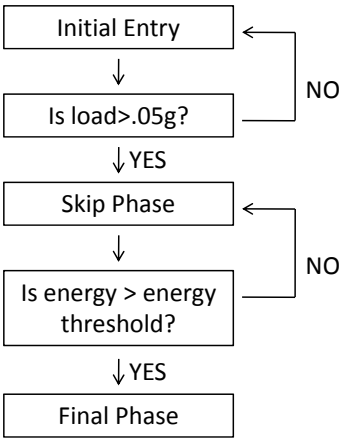

Figure 4.8 Phase transition logic with energy (discontinuous planning)

when $V <= 1.5V_{final}$.

$$h = \frac{V_{final} - V}{\dot{V}} \tag{4.10}$$

As discussed in Section 4.1.2, the bank reversals in the planning take place instantaneously. This could cause a problem due to the finite time spent in reversal and with the number of reversals occurring being different from one secant iteration to the next.

On occasion, the secant computation will calculate a bank angle that causes the ensuing trajectory to skip out. Thus calculation of the miss distance is not possible. To restart the computation, the current bank angle that skips out has 2.5° added to it and the trajectory is re-simulated. This continues in a similar fashion as finding the initial angle for the secant until the trajectory ends in an undershoot. Then the secant computation resumes.

Due to the inexact line search, the secant scheme will compute a value for $\cos\sigma > 1$. A check is performed to see if the previously converged bank angle is less than 15°. If it is, then it is assumed that the vehicle actually needs to fly a small bank angle. If not, then the inexact line search caused a problem, and the procedure described for leaving the ground is followed to restart the computation. A similar logic follows if $\cos\sigma < -1$ and the previously converged

bank angle is greater than 165°.

4.2 Closed-Loop Guidance

Once the process in the preceding section converges, a feasible skip trajectory has been found. Subsequent guidance commands in principle can be found by tracking this reference trajectory as done in the Apollo design. For a vehicle with low L/D value, however, achieving the kind of high precision required at the atmospheric exit (particularly in the flight path angle) by tracking a fixed reference trajectory is difficult. This is because there are multiple competing conditions in velocity, flight path angle, and range that need to be met, and the vehicle's limited maneuverability is inadequate to null all trajectory dispersions if they are not small.

The guidance approach adopted in this work is to use the same algorithm described in Section 4.1 repeatedly to generate a new skip trajectory using the current state as the initial condition in every guidance cycle. The magnitude of the commanded bank angle in the current cycle is then the one found in the solution. Since this solution, thus the bank angle, depends on the current state of the trajectory, such a guidance command is closed-loop in nature. The advantages of this approach over reference-tracking should be three-fold: (1) the vehicle should always fly on a feasible trajectory that satisfies the required condition on range-to-go to the landing site, provided that the algorithm converges; (2) there should be no need for separate tracking guidance law and the generation and tuning of the associated gains; (3) it might be possible to further incorporate the information from the vehicle health monitoring system in the guidance solution in the event of degradation of vehicle capability due to partial system failure. The risk, of course, is that the convergence is not achieved in one or several guidance cycles. This risk arises most likely only in the case of severely abnormal situations. If non-convergence occurs in just one guidance cycle, the previously obtained solution can always still be used. One of the objectives of evaluation is to assess the robustness and reliability of the algorithm through extensive testing.

On closed-loop guidance calls to the algorithm, the previously calculated bank angle is

used as an initial guess for the initial to-be-flown bank angle for the skip phase rather than beginning with zero bank angle. If this angle is still sufficient to bring the downrange error within 25 km, the secant method is skipped and the previous bank angle is flown. Otherwise, the secant method proceeds as above and determines a new bank angle magnitude.

During the finite time in a bank reversal in the actual flight, the above algorithm continues to run, with the current state as the initial condition. The transient effects of the bank reversal will cause the algorithm to yield a somewhat different bank angle once the reversal is completed. Further discussion concerning this effect appears in Section 5.2.

No closed-loop guidance is performed in the Kepler phase where aerodynamic control is ineffective.

In the final entry phase, closed-loop guidance is provided by another numerical predictor-corrector algorithm.[24] This algorithm searches for a one-parameter linear bank-angle profile that will result in the downrange flown by the vehicle be equal to the actual range-to-go to the landing site. The reader is referred to Ref.[24] for detailed description of the algorithm and performance evaluation based on Monte Carlo simulations of the final phase entry flight.

CHAPTER 5. CRITICAL ISSUES IN CLOSED-LOOP GUIDANCE

In this chapter several issues critical to the success and performance of the proposed algorithm when applied in closed-loop skip entry guidance are addressed.

5.1 Seamless Phase Transitions

The Apollo entry guidance[4] and a recent work[32] on Orion CEV have identified the need for improved phase transition logic in the skip entry guidance. In the Apollo entry guidance, a different strategy of guidance is used in each phase of the skip entry. Therefore, it is necessary to ensure that the transition from one phase to another will not cause difficulty in the next phase. In particular, the Apollo final phase guidance is based on tracking of a pre-stored reference trajectory. If the transition into final phase guidance does not takes place at an appropriate point, the guidance command could be severely saturated while attempting to reach the reference conditions. Another critical situation, where smooth transition between different phases of guidance is needed, occurs when a skip trajectory is not required, as in the case where the initial downrange is relatively short.

Since this current skip guidance algorithm generates a reference trajectory at every time step in the same manner and the guidance strategy is the same in each phase of the skip trajectory, a well-defined transition point is not necessary between two phases. The final phase guidance is of a very similar nature in generating the bank angle command, and it integrates with the skip guidance smoothly. The starting point for the final phase guidance is not constrained to a small neighborhood about a fixed reference trajectory because a feasible trajectory is generated onboard based on the actual state conditions. All these traits in this approach afford seamless and easy transition between phases of the guidance logic.

Figure 5.1 shows the different phases in the skip entry guidance logic. The initial phase of entry is an open-loop phase, flown with a 0° bank angle until the aerodynamic load reaches 0.05 g_0. Following this, the skip is flown with a bank angle computed as described in Chapter 4.1. In the normal skip entry case, the skip phase continues when the flight path angle (FPA) becomes positive. The skip trajectory is considered to have entered the Kepler phase when the load is below 0.05 g_0. The vehicle switches to the pre-planned bank angle of 70 degrees during Kepler phase. The final phase begins once range-to-go has decreased below 2000 km.

In the case where the entry condition renders a skip trajectory completely unnecessary, this algorithm will automatically adjust to the need or lack thereof for a skip. In the skip phase, if the vehicle never has a positive flight path angle and the range-to-go continues to decrease below 2000 km, the final phase guidance algorithm will take over. In such a case, a direct entry naturally takes place instead. The path of "Direct Entry" in Fig. 5.1 is for this scenario. This type of entry will also occur if the range-to-go is below 2000 km in the initial entry phase.

There are also cases where the skip is sufficiently short so that the vehicle does not leave the atmosphere (loft trajectory), thus a Kepler phase does not exist. In this case, the skip guidance continues to run with a positive flight path angle. Once the range-to-go is found to be less than 2000 km, the final phase guidance is initiated, as indicated by the path of "Loft Trajectory" in Fig. 5.1.

Figure 5.2 shows the typical altitude-versus-downrange profiles of the normal skip, loft and direct entry trajectories. Figure 5.3 plots the corresponding closed-loop bank angle histories for these trajectories. These and future plots show the downrange-to-go in kilometers, which has been converted from Eq. (4.3) (in radians) by multiplying by the Earth's radius, R_0. The present algorithm automatically adapts to all these different situations without a priori indication on which kind of trajectory a particular mission will take.

5.2 Direction of Bank Reversals

The problem of steering or crossrange reduction is determined by whether to assign a positive or negative value to the computed bank angle magnitude. The initial call to the bank

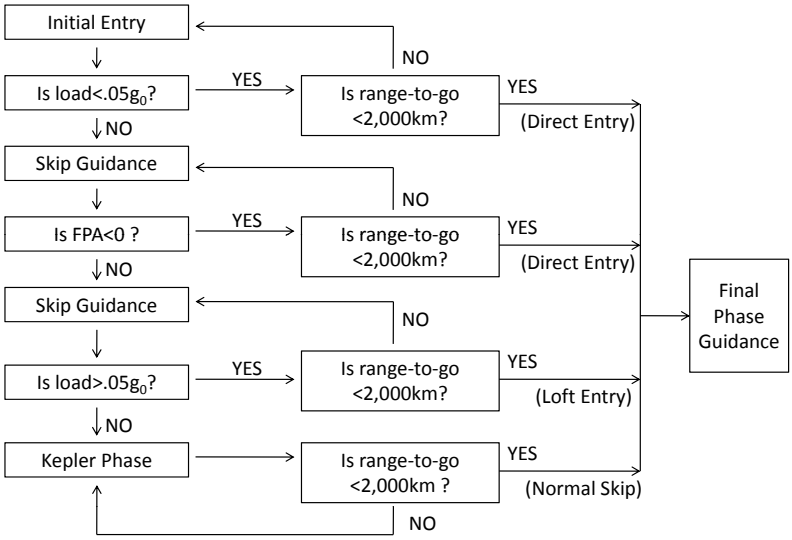

Figure 5.1 Phase transition logic for guidance

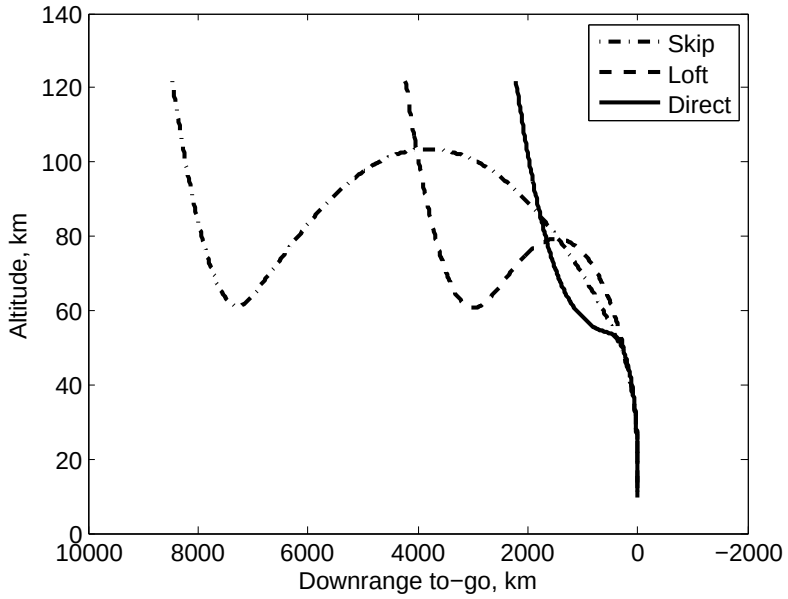

Figure 5.2 Possible entry trajectory types

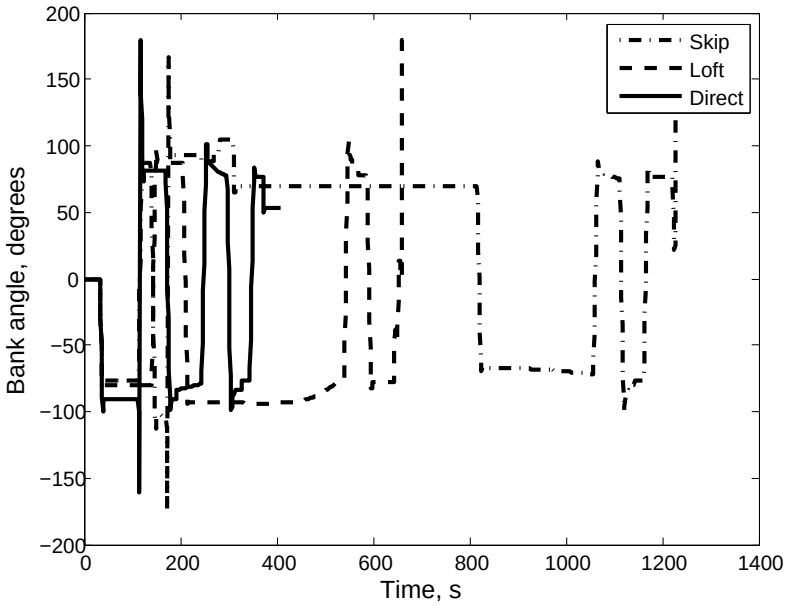

Figure 5.3 Bank angle histories for different trajectory types

reversal logic assigns the bank angle a sign that is opposite to that of a crossrange variable χ defined by

$$\chi = \sin^{-1}[\sin s_{to-go} \sin(\psi - \Psi)] \tag{5.1}$$

where s_{to-go} is the range-to-go, ψ the current heading angle, and Ψ the line-of-sight azimuth angle along a great circle to the landing site. All the variables have units of radians. On future calls, the sign of the bank angle will be reversed whenever the crossrange exceeds the velocity-dependant threshold defined in Eqn. (4.8).

In the actual flight, instantaneous bank angle reversals are not possible as they were assumed during the planning. The bank angle changes can occur at a metered rate that does not exceed the vehicle-dependent rate and acceleration limits. Through testing, it is found that the sensitivity of the trajectory with respect to the transient effects of the bank reversals in the skip phase is rather high, and the transition in bank reversals must be performed in an appropriate direction. If the pre-reversal bank angle has a magnitude greater than ninety degrees, the vehicle has excess energy and a bank reversal from below (through full lift down)

will help deplete some of the excess energy and cause a net effect of leaving the vehicle in the middle of the downrange envelope. However, a bank reversal from above (through full lift up) in this case will slow down the needed energy depletion. If such a "wrong cycle" repeats a few times, soon the current downrange will be too short for the vehicle even if the vehicle flies with full lift down in the skip phase. The situation is similar in nature (in the opposite sense) when the pre-reversal bank angle magnitude is less than ninety degrees. Thus, it appears critical for the guidance system to always command the roll in a skip phase bank reversal via the short way rather than always from above or from below. If the bank angle has a magnitude equal to 90°, the vehicle is on the path of zero lift to the target and has maximum margin with respect to the downrange targeting capability.

5.3 Automated Targeting Bias

In the recent work by Rea and Putnam[32] on Orion CEV and this work, it is found necessary to target a biased "landing site" for longer northbound and eastbound skip trajectories in the bank reversal logic during the skip flight. In these situations, the effects of Earth rotation tend to cause the vehicle to have a crossrange at the start of the final phase that is larger than the final phase guidance can compensate for, mainly in the presence of large dispersions with high drag and low lift. The target biasing is introduced to force the trajectory to veer more toward the desired direction during the skip so as to have acceptable crossrange for the final phase. The biased landing site is only targeted during the skip phase in the lateral logic (and the longitudinal logic still uses the true landing site).

Perhaps the most effective biasing effect is achieved by moving the targeted site from the true landing site along the direction of crossrange. The objectives to be achieved are to establish how much bias should be used, and for a specified bias, determine what the coordinates of the biased site should be. In this work the trajectory planning capability of the algorithm is used to automate this process.

During the first call to the trajectory planning in the skip phase, the true landing site coordinates are used. Once a converged solution is found, the value of the crossrange is

examined at the point in the planned trajectory where the final phase begins. If the crossrange is outside the velocity-dependent envelope in the lateral logic for bank reversal (cf. Eq. (4.8)), the bias logic is executed. This logic determines the bias necessary to cause the crossrange to have a sign opposite to that of the crossrange without bias. Otherwise the biasing logic is bypassed.

When the biasing is determined to be needed, the targeted site is moved away from the true landing site by a small amount $\delta\chi$ in the crossrange direction. Taking differentials of Eq. (5.1), while keeping the range-to-go s_{to-go} unchanged and current vehicle heading angle ψ fixed, gives:

$$\delta\chi \cos\chi = \sin(s_{to-go}) \cos(\psi - \Psi)(-\delta\Psi) \tag{5.2}$$

The small increment $\delta\chi$ is selected for the target biasing (e.g., $|\delta\chi| = 0.15^\circ$ to begin with). The sign of $\delta\chi$ is opposite to that of the crossrange at the beginning of the final phase without target biasing. The above equation determines $\delta\Psi$ since everything else in the equation is now known. The great-circle distance between the current vehicle position (θ, ϕ) and the biased site (Θ, Φ) is:

$$\cos(s_{to-go}) = \sin\phi \sin\Phi + \cos\Phi \cos\phi \cos(\Theta - \theta) \tag{5.3}$$

The azimuth angle from the vehicle location to this biased landing site is given by:

$$\sin\Psi = \frac{\sin(\Theta - \theta)\cos\Phi}{\sin(s_{to-go})} \tag{5.4}$$

Take differentials of Eqs. (5.3)-(5.4), keeping in mind that θ, ϕ, and s_{to-go} are not changed by the biasing. The following system of linear equations is obtained as a result:

$$\begin{bmatrix} \frac{\cos(\Theta^* - \theta)\cos\Phi^*}{\sin(s_{to-go})} & -\frac{\sin(\Theta^* - \theta)\sin\Phi^*}{\sin(s_{to-go})} \\ -\cos\Phi^* \cos\phi \sin(\Theta^* - \theta) & \sin\phi\cos\Phi^* - \sin\Phi^* \cos\phi \cos(\Theta^* - \theta) \end{bmatrix} \begin{bmatrix} \delta\Theta \\ \delta\Phi \end{bmatrix} = \begin{bmatrix} \delta\Psi \cos\Psi \\ 0 \end{bmatrix} \tag{5.5}$$

where (Θ^*, Φ^*) are the coordinates of the true landing site. With $\delta\Psi$ already found from Eq. (5.2), $\delta\Theta$ and $\delta\Phi$ are found easily from system (5.5). The coordinates of the biased landing site are then computed by

$$\begin{aligned} \Theta &= \Theta^* + \delta\Theta \\ \Phi &= \Phi^* + \delta\Phi \end{aligned} \tag{5.6}$$

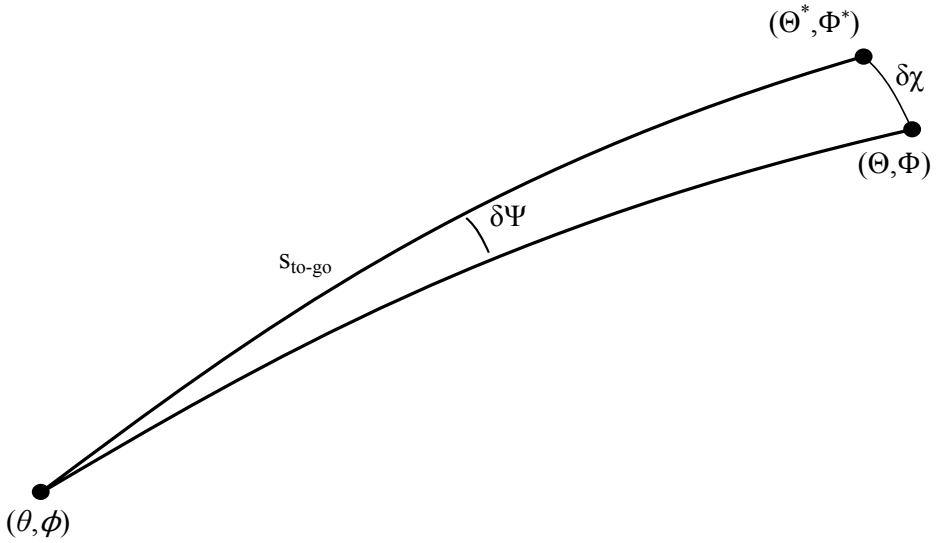

Figure 5.4 Crossrange biasing diagram

Note that since Eqs. (5.2) and (5.5) are linear in $\delta\chi$, $\delta\Psi$, $\delta\Theta$ and $\delta\Phi$, any change in $\delta\chi$ means the changes of the same proportion in $\delta\Theta$ and $\delta\Phi$. A new skip trajectory is planned using the biased target coordinates in Eq. (5.6). If the predicted crossrange condition at the beginning of the final phase still does not satisfy the above stated criterion for stopping the biasing, the bias $\delta\chi$ is doubled. This will lead to the increments of the same proportion in $\delta\Theta$ and $\delta\Phi$ (but system (5.5) does not need to be resolved again). The biasing process is repeated using the updated biased target until the stopping criterion is met.

Fig. 5.4 shows a schematic biasing.

Experience shows that the typical targeting bias for the long northbound entry is about 1.5° and the eastbound entry about 1.0°.

Fig. 5.5 shows the crossrange with respect to the true landing site and the crossrange with respect to the biased landing site. During the skip phase (before 300 s) of this particular trajectory, the bank reversals are with respect to the biasing landing site, thus that particular crossrange remains within the corridor. Following that is the Kepler phase. With the final phase starting around 700s, the crossrange with respect to the true landing site is inside the

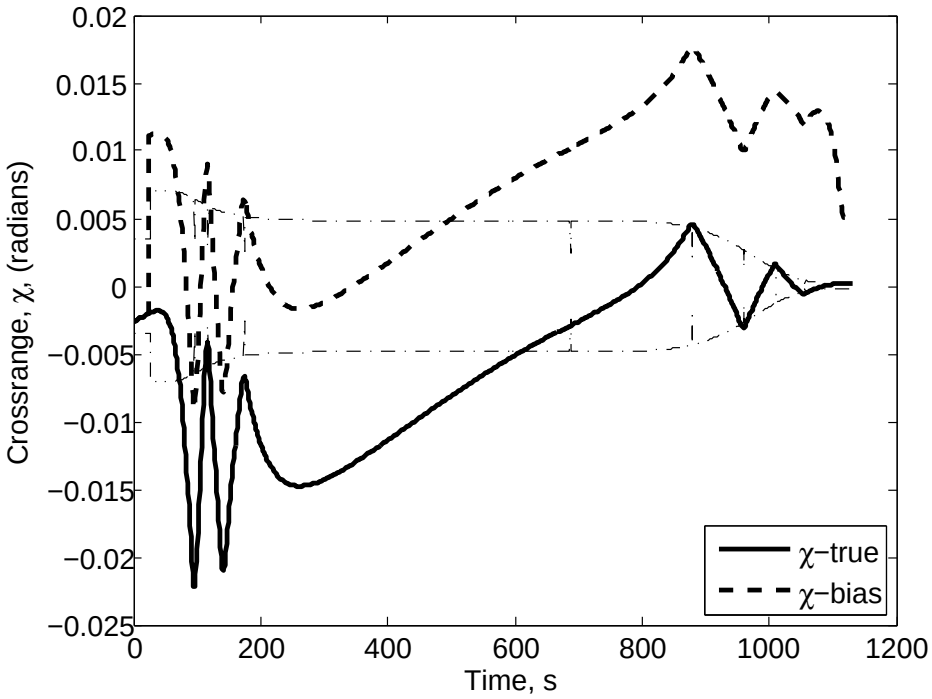

Figure 5.5 True crossrange and biased crossrange

corridor. This allows the final phase guidance to maintain the crossrange within the corridor until the target site is reached. This particular example is for an eastbound entry.

One note concerning the matrix in Eq. 5.5 is necessary. The matrix is not always invertible. If $\sin(s_{to-go})$ equals zero, the system obviously has no solution. Following removal of the $\sin(s_{to-go})$, applying a trigonometric identity, and factoring out the term $\cos\Phi^*$ which is zero if the landing latitude (Φ^*) is at either pole ($\pm 90°$), the determinant of the matrix equals zero in the following condition:

$$\cos(\Theta^* - \theta)\tan\phi = \tan\Phi^* \tag{5.7}$$

The conditions under which this equation hold true are not immediately obvious. A brute force search is run to find conditions under which the conditions would be true. For this search, the landing site is fixed at Edwards Air Force Base (EAFB) as described in Chapter 6.1. Then the position of the vehicle is varied to find the solution to Eq. (5.7). Fig. 5.6 shows

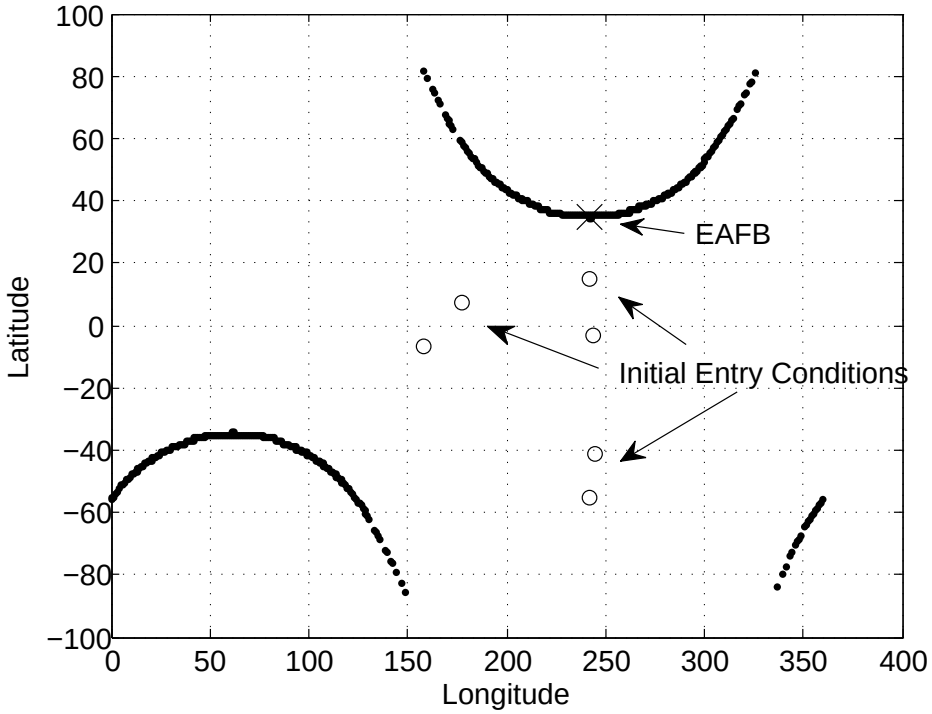

Figure 5.6 Crossrange biasing exclusion lines for initial entry condition with landing site at EAFB

the exclusionary lines for the vehicle position. If the vehicle position is along one of the arcs shown in the Fig. 5.6, then the determinant is near zero. The vehicle initial entry conditions are indicated by circles. Therefore, the trajectory enroute to EAFB will not encounter a condition where the determinant is zero. Similar checks were performed for the other two landing sites (near Hawaii and KSC). Neither landing site will encounter this problem.

5.4 Lift and Drag Filters

It is suggested in the literature and found to be true again in this work that appropriate use of estimated information on the aerodynamic and density biases is very helpful in achieving required high precision.[9, 21, 22, 23] This is particularly so in the presence of large atmospheric and aerodynamic uncertainties. The dispersions include those in entry condition at the first entry interface due to de-orbit condition dispersions, aerodynamics, vehicle mass, and

atmospheric density. All the dispersions, except for those in the entry condition, affect only the lift and drag accelerations. During the flight, the actual values of lift and drag accelerations can be obtained from navigation data and accelerometer outputs. The effect of one source of dispersion (e.g. density) cannot be distinguished from another (e.g. aerodynamic coefficients) without another measurement (e.g. an air data system). Yet it is realized that only the combined effect of these dispersions needs to be estimated. Toward this end, a first-order fading memory filter is used[33]

$$X_{n+1} = X_n + (1 - \beta)(X^* - X_n) \tag{5.8}$$

where, X^* is the current ratio of the measured variable (lift or drag acceleration) to its nominal value based on the nominal model via Eq. (2.8) or (2.9), X_n is the past filtered ratio, and $0 < \beta < 1$ is a gain, typically a value slightly less than one to emphasize past values over the most recent value. For this work, $\beta = 0.9$ is used. To initialize the filter, the first past filtered ratio, X_0 is set to be equal to one. In each skip trajectory guidance cycle, the output of this filter is used to multiply the nominal lift L (or drag D) profile in the integration/planning of the trajectory, so the corresponding trajectory is computed based on the modified L and D.

This type of filter would be able to capture the combined effect of constant biases in mass, density, and constant percentage dispersions in aerodynamic coefficients. However, in the testing of the algorithm significant non-zero altitude-dependent components in the atmospheric density dispersions are intentionally introduced via GRAM 2007 and the analytic model so the algorithm would not benefit unrealistically from this feature of the filter (see section 6.2). In the literature, the usefulness of such type of measures has been long recognized in atmospheric entry and aeroassist maneuvers.[9, 21, 22, 23] With the exception of Ref.[23], similar estimation has been applied to just drag or the L/D ratio. Experience in skip entry guidance shows that application of such estimation to shape both drag and lift acceleration profiles in the on-line trajectory planning and guidance has clear advantages over application only to D.

A second-order filter was also tested, but there was no visible improvement in results over the first-order filter. The form of the filter is as follows[33]

$$X_{n+1} = X_n + \dot{X}_n T_s + (1-\beta)(X^* - (X_n + \dot{X}_n T_s)) \tag{5.9}$$

$$\dot{X}_{n+1} = \dot{X}_n + (1-\beta^2)/T_s(X^* - (X_n + \dot{X}_n T_s)) \tag{5.10}$$

where, all the variables are the same as before with the additions of: $\dot{X}_n$ is the past-filtered-derivative-ratio, and T_s is the sampling time. To initialize the filter, the first past-filtered-derivative-ratio, $\dot{X}_0$ is set to be equal to zero. The sampling time used in this work was the same as the guidance cycle of one second. (or 1 Hz)

CHAPTER 6. VEHICLE AND DISPERSION MODELS FOR TESTING

6.1 Nominal Vehicle Model

The vehicle model used in the simulations in this paper is similar to that of the Orion Crew Exploration Vehicle. The vehicle has a capsule configuration with a base diameter of 5 meters and weighs 8382 kg (18,464 lb). The vehicle flies a Mach-dependent trim angle of attack profile, which is about 160.2° in the skip phase. The L/D value is 0.289 at this trim angle-of-attack. The maximum rate and acceleration imposed on the bank angle in the 3DOF simulations are 20°/s and 10°/s^2, respectively, similar to those used in the literature.[22] The mission conditions correspond to those from a lunar-return mission and landing at Edwards Air Force Base (EAFB). One mission is included that simulates a return to Kennedy Space Center (KSC).

6.2 Entry Condition Dispersion

The purpose of dispersion simulations is to evaluate the performance of the guidance algorithm in the presence of significant deviations in trajectory state, vehicle, and environment modeling uncertainties. Before discussing the entry condition dispersions, a slight deviation to discuss coordinate systems is necessary. The state variables used in most of this work $(r, \theta, \phi, V, \gamma, \psi)$ can be converted to an Earth-centered Earth-fixed (ECEF) coordinate system in the following manner. The position vector in the ECEF coordinate system can be represented as shown in Fig. 6.1.

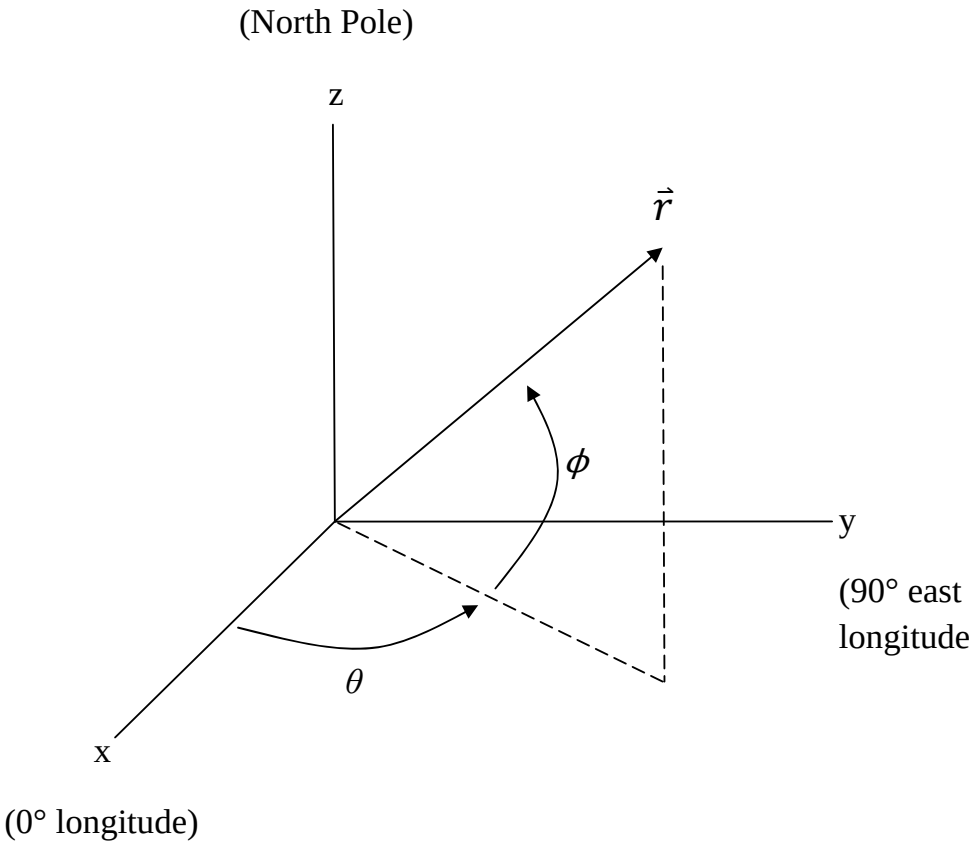

Figure 6.1 ECEF coordinate system

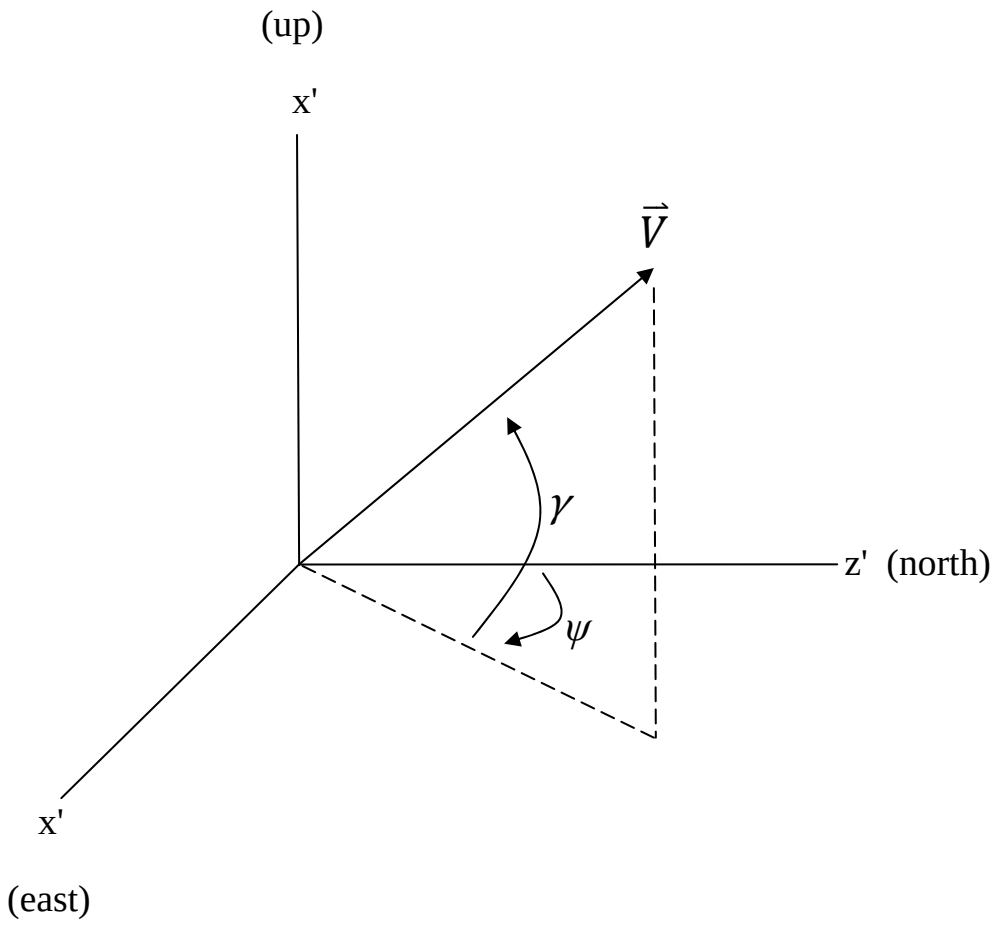

Figure 6.2 Velocity LVLH coordinate system

The conversion can be made as in Eq. (6.1),

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos\phi\cos\theta \\ r\cos\phi\sin\theta \\ r\sin\phi \end{bmatrix} \tag{6.1}$$

The velocity components are converted to a local-vertical-local-horizontal (LVLH) coordinate system as in Fig. 6.2,

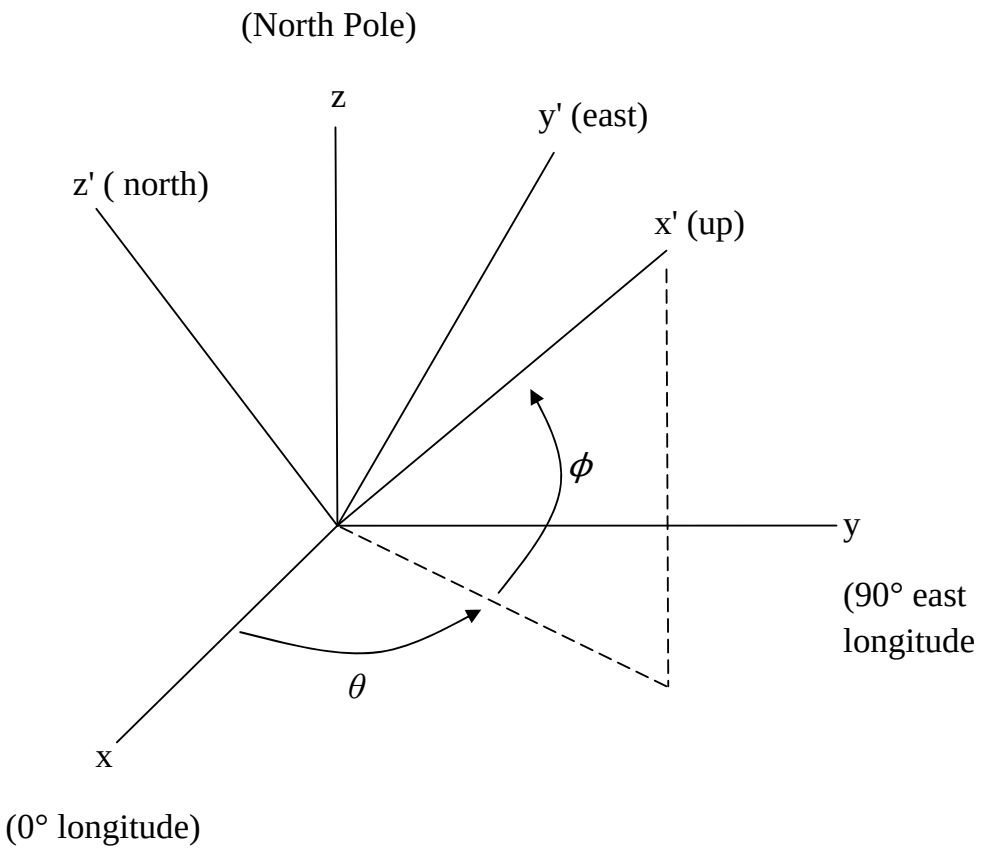

Figure 6.3 ECEF and LVLH coordinate transformation

$$\begin{bmatrix} V_{up} \\ V_{east} \\ V_{north} \end{bmatrix} = \begin{bmatrix} V\sin\gamma \\ V\cos\gamma\sin\psi \\ V\cos\gamma\cos\psi \end{bmatrix} \tag{6.2}$$

The velocity in LVLH coordinates can be transformed to ECEF via Eq. (6.3) using rotation matrices as in Fig. 6.3.

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} V_{up} \\ V_{east} \\ V_{north} \end{bmatrix} \tag{6.3}$$

Now with the position and velocity in ECEF coordinates, a discussion of entry condition dispersions can proceeed. The initial entry conditions except for altitude are perturbed via

a covariance matrix. The initial entry conditions can be written as a function of deorbit conditions in ECEF coordinates.

$$\bar{X}_{EI} = f(\bar{X}_{deorb}) \tag{6.4}$$

Taking a partial derivative of this equation results in

$$\frac{\partial \bar{X}_{EI}}{\partial \bar{X}_{deorb}} = \bar{C} \tag{6.5}$$

This matrix, $\bar{C}$, is a sensitivity matrix describing the relation between the dispersions at the last deorbit burn and those at entry interface. It is found by propagating the nominal entry interface conditions, $\bar{X}^*_{EI} = (\theta, \phi, V, \gamma, \psi)$ backward three hours in time where the last deorbit burn is expected to be performed. The resulting state vector is converted from the nominal state conditions to ECEF coordinates as described in Eqs. (6.1)-(6.3).

The first dimensionless position, x in ECEF coordinates, $\bar{X}^*_{deorb} = (x, y, z, V_x, V_y, V_z)$ is perturbed by a small value (δ = 1e-5). Then, the new state vector is reconverted to the usual state components via the inverse transformation of Eqs. (6.1)-(6.3), and propagated forward until the altitude reaches the entry interface at 121.92 km (400,000 ft) giving the perturbed state:

$$\bar{X}'_{EI} = \begin{bmatrix} \theta'_{EI} & \phi'_{EI} & V'_{EI} & \gamma'_{EI} & \psi'_{EI} \end{bmatrix}^{\mathrm{T}} \tag{6.6}$$

The first column of the sensitivity matrix can be computed as

$$\bar{C}_1 = \frac{\bar{X}'_{EI} - \bar{X}^*_{EI}}{\delta} \tag{6.7}$$

The second component of the sensitivity matrix can be computed in a similar fashion as the first, by perturbing y. Ultimately to produce the sensitivity (covariance) matrix each component must be perturbed.

$$\bar{C} = \left[\bar{C}_1 | \bar{C}_2 | \bar{C}_3 | \bar{C}_4 | \bar{C}_5 | \bar{C}_6\right]_{(5x6)} \tag{6.8}$$

The deorbit condition dispersions in position and inertial velocity components are modeled by zero-mean Gaussian dispersions ($N(\mu, \sigma^2)$) where μ is the mean, and σ^2 is the standard deviation. Each position component has a 3-sigma value of 135 m, and each inertial velocity

Table 6.1 Dispersions in entry interface state and other parameters

State/Parameter	Distribution	3-σ Value/Range (northern entry)	3-σ Value/Range (eastern entry)
Longitude (deg)	Zero-mean Gaussian	0.0749	0.2591
Latitude (deg)	Zero-mean Gaussian	0.3202	0.1790
Relative velocity (m/s)	Zero-mean Gaussian	12.9053	13.3611
Flight path angle (deg)	Zero-mean Gaussian	0.1484	0.1505
Heading angle (deg)	Zero-mean Gaussian	0.0973	0.0526
C_L	Zero-mean Gaussian	0.0778(20%)	0.0778(20%)
C_D	Zero-mean Gaussian	0.2696(20%)	0.2696(20%)
Mass (kg)	Uniform	$\pm$5%	$\pm$5%
Atmospheric Density(analytic)	Uniform	$\pm$40%	$\pm$40%
Atmospheric Density	GRAM 07 dispersed		

component has a 3-sigma value of 1.35 m/s. The entry condition dispersions are then obtained by multiplying the deorbit dispersions with the co-variance matrix.

$$\begin{bmatrix} \Delta\theta \\ \Delta\phi \\ \Delta V \\ \Delta\gamma \\ \Delta\psi \end{bmatrix} = \bar{C} \begin{bmatrix} N(0,\sigma_x^2) \\ N(0,\sigma_y^2) \\ N(0,\sigma_z^2) \\ N(0,\sigma_{Vx}^2) \\ N(0,\sigma_{Vy}^2) \\ N(0,\sigma_{Vz}^2) \end{bmatrix} \tag{6.9}$$

The resultant dispersions in entry conditions are summarized in Table 6.1 for both northbound and eastbound entry missions. These entry condition dispersions are comparable to those reported in the recent literature.[34]

6.3 Atmospheric Dispersions

For the dispersion in atmospheric density, it was originally chosen to use an analytic model. However, to enhance the validity of the results, the NASA Global Reference Atmospheric Model (GRAM) 2007 model was obtained. GRAM is a fairly large and detailed FORTRAN code that increased the simulation runtime significantly. The primary test results of the proposed algorithm are obtained with GRAM. However, in the comparison testing between the Apollo

skip guidance and the present algorithm, the analytic model is used to reduce code complexity. In figures below(6.4-6.5), the two perturbation models are compared to the US Standard Atmosphere. The figures show similar dispersion in the upper atmosphere. However, the analytic model shows greater dispersion in the lower atmosphere where the more significant (for controlling the vehicle's downrange) part of the skip trajectory actually takes place.

6.3.1 GRAM Atmosphere

For the atmospheric density dispersions, the 2007 version of the Earth Global Reference Atmosphere Model (GRAM07) is used without any adjustments for wind. GRAM07 is the most recent updated version of GRAM99[36], an industry standard for modeling atmospheric properties and dispersions. It provides complete geographic and temporal variations (though temporal variations are not studied in this work) rather than a standard atmosphere which only provides variations with respect to altitude. It also simulates position-dependent and altitude-dependent perturbations in the mean atmosphere. The strength of these factors is controlled through a parameter known as RPSCALE. This is a scale factor on the 1-sigma variation about the nominal atmospheric parameters. The nominal value of RPSCALE is 1.0, which is used throughout all simulations. One hundred dispersed profiles of the GRAM07 atmosphere are shown in Fig. 6.4 relative to the 1976 US Standard Atmosphere. Note the large magnitudes of dispersions in the upper atmosphere, but the dispersions near the ground are smaller.

6.3.2 Analytic Model

For the analytic atmospheric dispersion, based on qualitative Shuttle flight data,[35] it was decided to use a dispersion model with a constant bias and two sinusoidal variations, one with long period, high magnitude and one with short period, low magnitude. Similar density waves have been used in other work.[22] The density wave of the dispersed atmosphere used in the simulations is represented by:

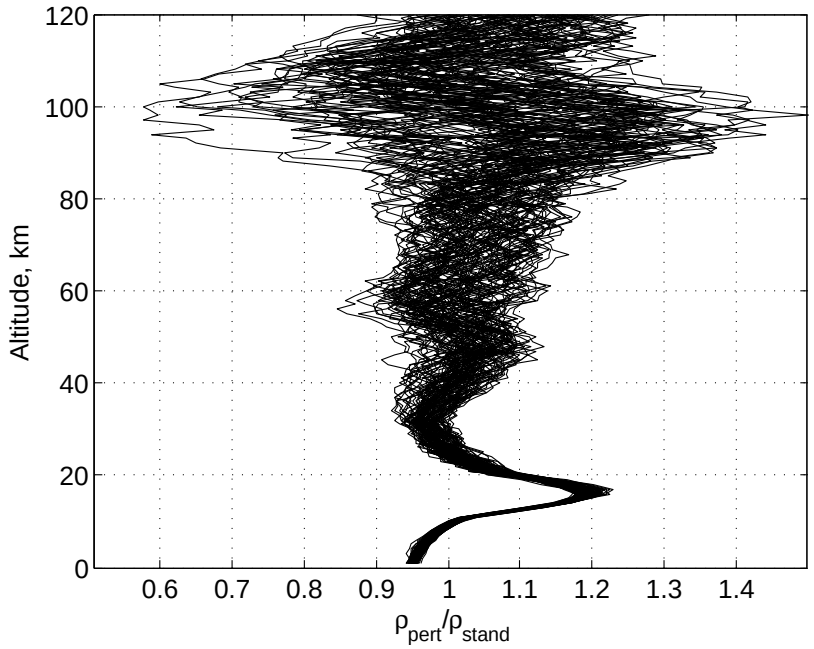

Figure 6.4 Ratio of GRAM 07 perturbed atmosphere to US 76 standard atmosphere as a function of altitude for 100 cases.

$$\rho_{true} = \rho_{nom}\{1 + B_{atm} + [M_1 + M_2 \sin(h\omega_2)] \sin(h\omega_1 + \lambda)\} \tag{6.10}$$

where ρ_{true} is the actual atmospheric density, ρ_{nom} is the density computed from the 1976 U.S. standard atmosphere, B_{atm} is a constant bias term, uniformly distributed with maximum values of ± 0.2. M_1 is the magnitude of the low frequency sinusoidal variation, uniformly distributed and constrained to between $\pm 0.1 - \pm 0.19$ so that the altitude-dependent sinusoid dispersion will have a significant presence. The effect of having both positive and negative values is to allow the atmosphere to be either thicker or thinner at any given altitude. The current altitude is h measured in kilometers. The frequency of the sinusoidal variation, ω_1, is uniformly distributed to cause between one-half and two periods over the altitude range of the vehicle (122 km to 0 km). The higher frequency sinusoid has similar parameters, M_2, and ω_2, the magnitude and frequency respectively. M_2 lies uniformly between 0 and 10% of M_1. This gives an approximate total atmospheric dispersion up to $\pm 40\%$. The higher frequency, ω_2, is another uniform parameter to give between zero and 50 periods over the altitude range of the vehicle. The phase of the low frequency sinusoid, λ, is chosen to minimize the dispersion at the ground in the following manner. By equating $\rho_{true} = \rho_{nom}$ at $h = 0$ in Eq. (6.10), this sets up the following constraint:

$$B_{atm} + M_1 \sin\lambda = 0 \tag{6.11}$$

The corresponding λ is solved from this equation, provided $|B_{atm}/M_1| \leq 1$. When this inequality condition is not satisfied, the phase parameter λ is chosen as $\lambda = \pi/2$ or $-\pi/2$, whichever minimizes $|B_{atm} + M_1 \sin\lambda|$. This way for each B_{atm} and M_1, λ is determined by these conditions (thus λ is different in each dispersed run). One hundred dispersed profiles of the atmosphere are shown in Fig. 6.5. Note the large variety of dispersions in the upper atmosphere, but due to the selection of the phase angle, the dispersion near the ground is relatively close to the nominal density.

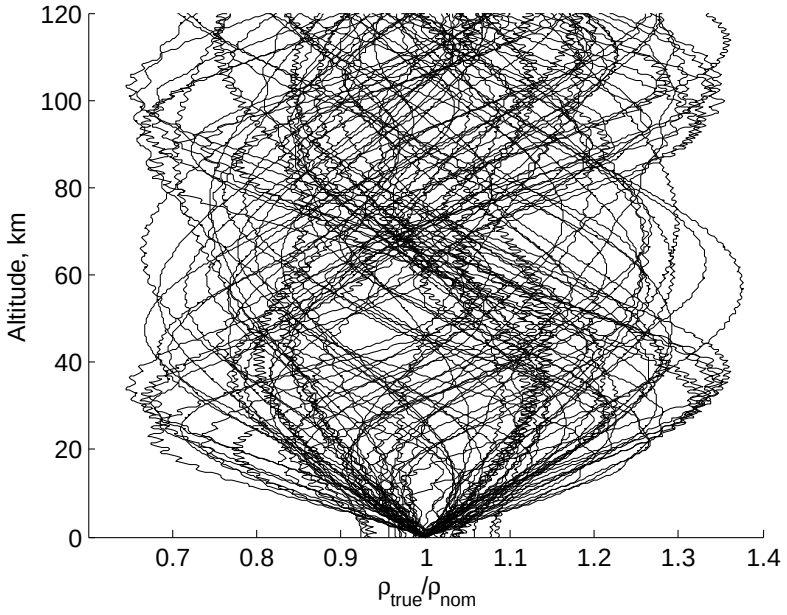

Figure 6.5 Density dispersion as a function of altitude for 100 cases.

6.4 Other Dispersions

The vehicle mass was perturbed uniformly up to 5%. With the nominal mass of 8382 kg, this gives a range of values of 7962.9 to 8801.1 kg.

For the lift and drag coefficients, existing work uses a 10% zero-mean normally distributed uncertainty in C_L or C_D.[22] To fully stress the algorithm developed in this paper, a 20% zero-mean normally distributed uncertainty from the C_L or C_D at high Mach numbers was used. This value is added to the nominally calculated C_L or C_D giving a constant bias at all Mach numbers. The effects of constant biases in the aerodynamic coefficients are actually more demanding as compared to Mach-dependent uncertainty, because their influence on L/D is persistently in one direction and there is no benefit of averaging effects. At high Mach, the nominal C_L is 0.3892 and the nominal C_D is 1.3479 giving a nominal L/D of 0.289. The L/D standard deviation is about 0.0276 giving a total range of about 0.2070 to 0.3728. However in simulation runs, (with 10,000 cases) the maximum L/D was about 0.43 with the minimum the same value.

It should be noted that navigation uncertainties are not considered here. However, it is well recognized that the uncertainties of state-of-the-art navigation systems are not primary factors affecting the performance of the entry guidance system.

CHAPTER 7. EVALUATION OF PREDICTOR CORRECTOR ALGORITHM

Simulations results are presented using the new predictor-corrector algorithm with GRAM 2007 as the atmospheric model.

Also, test results are shown with the analytic density dispersion model and four potential guidance scenarios. These are described in Chapter 8.

7.1 Mission Setup

Several different entry mission scenarios are tested. They correspond to different downrange distances and approach directions, resulting from different lunar return conditions. The initial conditions are given in Tables 7.1 and 7.2. The first mission in Table 7.1 has a short initial downrange of only 2200 km. There will be no skip trajectory for this mission. It is selected to test how the algorithm automatically adapts to such cases. In Table 7.2, the case for landing at KSC is considered the maximum range that a lunar return entry would endure based on the worst-case scenario for the Earth-Moon orbital conditions [37]. The initial conditions used in this case are chosen for that range. All the missions are run under exactly the same setting, without any mission-dependent adjustments of guidance parameters or logic.

For each mission, 10,000 dispersion runs are performed. The guidance cycle is set at 1 Hz. The Orion CEV program has a landing precision requirement of 5 km [32]. Therefore in testing, when the final position of the vehicle in a dispersed 3DOF simulation is within 2.5 km to the landing site, it is considered a successful run as the remaining 2.5 km is allocated for parachute drift. Any trajectory ending outside the 2.5 km radius, but within the 5 km radius is regarded as a 50% success. Any trajectory ending outside the 5 km radius from the landing

Table 7.1 Northbound initial conditions

State	Northbound Direct	Northbound Short Range	Northbound Medium Range	Northbound Long Range
Altitude, h (km)	121.92	121.92	121.92	121.92
Longitude, θ (deg)	242.00	244.00	244.83	242.00
Latitude, ϕ (deg)	15.00	-3.00	-41.13	-55.00
Relative velocity, V(km/s)	10.98	10.98	10.98	10.98
Flight path angle, γ (deg)	-5.576	-5.576	-5.576	-5.576
Heading angle, ψ (deg)	0.47	0.47	0.47	0.47
Downrange, s (km)	2200	4200	8400	10,000
Crossrange, χ (km)	7	204	298	42

Table 7.2 Eastbound initial conditions

State	Eastbound(EAFB) Medium Range	Eastbound(EAFB) Long Range	KSC Max Range
Altitude, h (km)	121.92	121.92	123.88
Longitude, θ (deg)	176.99	158.00	166.00
Latitude, ϕ (deg)	7.14	-7.00	-28.65
Relative velocity, V(km/s)	10.60	10.60	10.66
Flight path angle, γ (deg)	-5.988	-5.988	-6.23
Heading angle, ψ (deg)	54.63	54.63	74.00
Downrange, s (km)	7300	10,000	13,519
Crossrange, χ (km)	-17	-4.8	94.6

site is regarded as a failure.

7.2 Results

In Fig. 7.1, a plot of the nominal altitude versus downrange clearly shows the skip out of the atmosphere for a northbound medium range case (8400 km). The complete bank angle history is shown in Fig. 7.2 and only the skip phase in Fig. 7.3. The spike in bank angle appearing around 175 seconds in Fig. 7.2 is the spacecraft rolling from -100° through 180° to a positive bank angle, which can be seen more clearly in Fig. 7.3. By rolling through full lift down, the net effect is to maintain the maximum margin in bank angle capability. The resultant command by the time the bank reaches the opposite side is slightly smaller than it was at the beginning owing to the energy dissipated during the reversal. The ground track for the northbound skip entry trajectory is given in Fig. 7.4. The ground track for the eastbound skip entry also appears in Fig. 7.4. The altitude profile for a nominal eastbound medium range

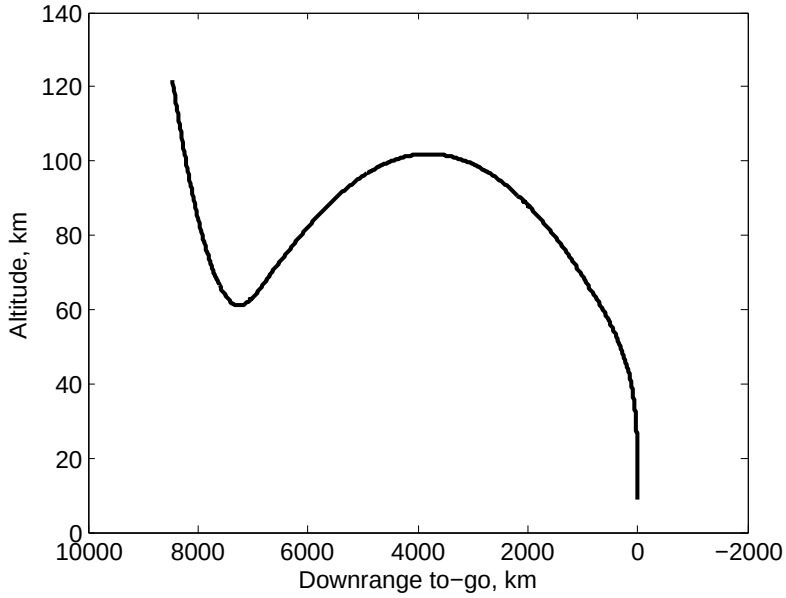

Figure 7.1 Altitude versus downrange-to-go for the nominal northbound medium range case (8400km).

(7300 km) skip entry is very similar to Fig. 7.1. The bank angle time history is very similar to Fig. 7.2.

In Fig. 7.5, the altitude versus downrange is shown for 100 dispersed trajectories for the northbound medium range skip entry. All the trajectories follow a similar path in the down-control phase, however they differ considerably during the up-control and Kepler phases. With the normal start of the final phase with 2000 km downrange-to-go, the final phase guidance has to respond to up to 13 km differences in altitude at this point. These kind of dispersions would likely severely stress the Apollo final phase guidance. But with the seamless integration of the current skip guidance and final phase guidance algorithm in Ref. [24], precision landing is still achieved reliably.

Similarly, the altitude histories of 100 dispersed trajectories for the short-range (2200 km) eastbound mission are plotted in Fig. 7.6. While the nominal trajectory for this mission is a direct entry trajectory, some of the dispersed trajectories are actually loft trajectories as can be seen from Fig. 7.6. The guidance algorithm automatically adapts to both direct entry

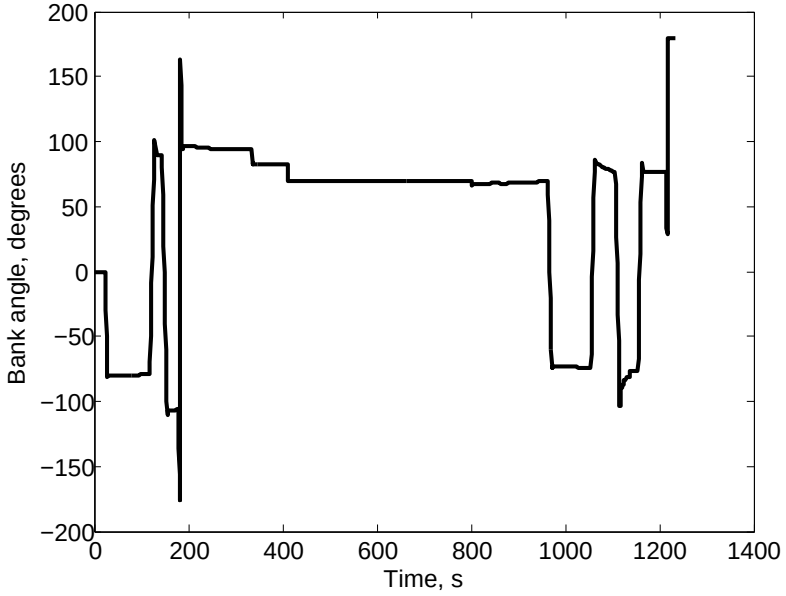

Figure 7.2 Bank angle profile for the nominal northbound medium range case (8400km).

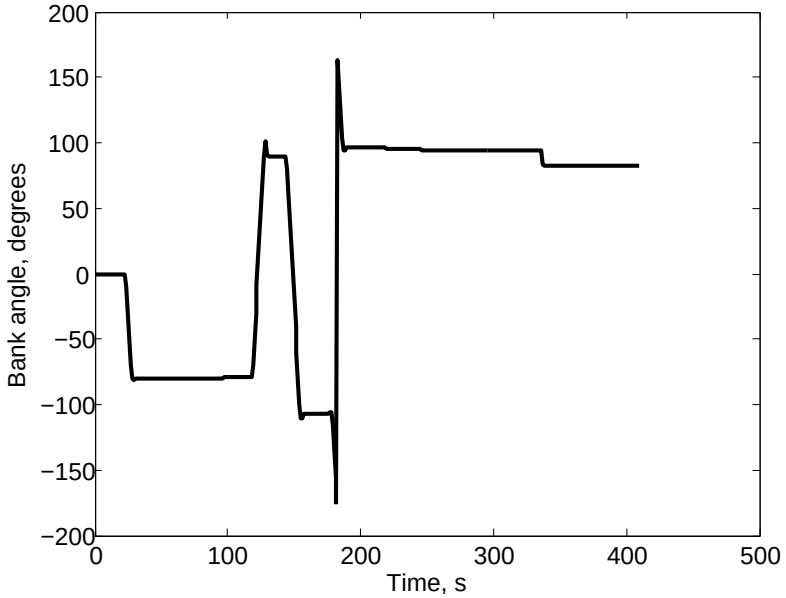

Figure 7.3 Close-up view of bank angle profile during skip phase for the nominal northbound medium range case(8400km).

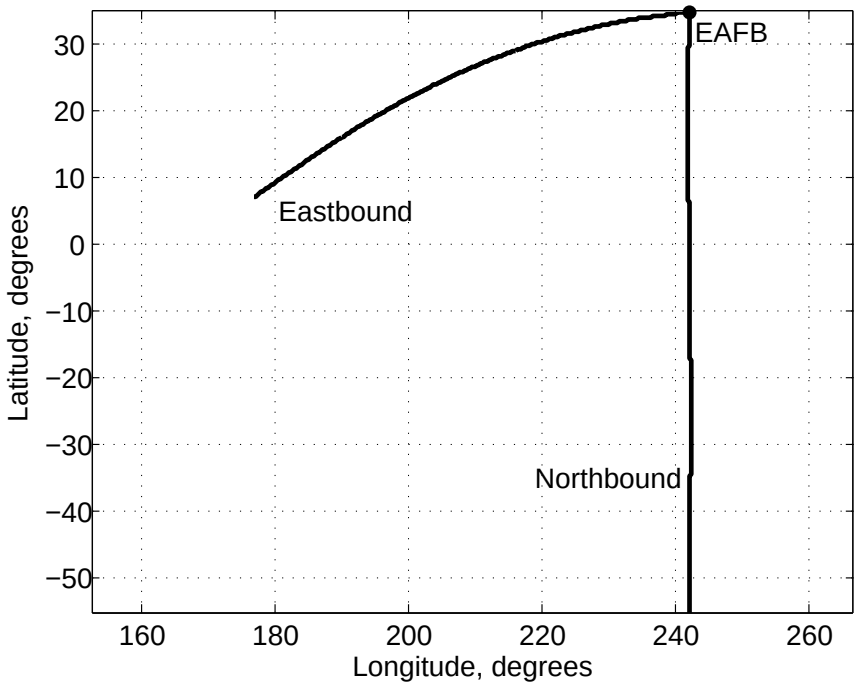

Figure 7.4 Ground track for the nominal northbound and eastbound cases.

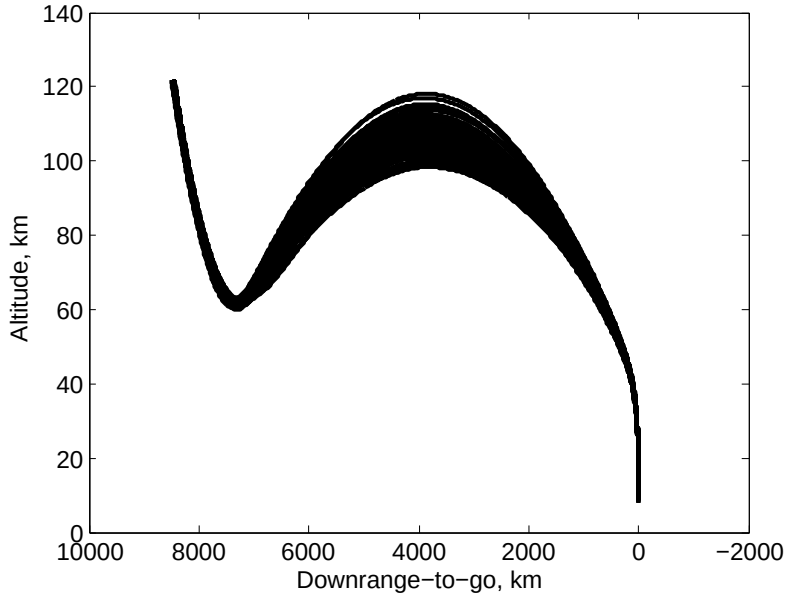

Figure 7.5 Altitude versus downrange-to-go for 100 dispersed trajectories (northbound, 8400 km).

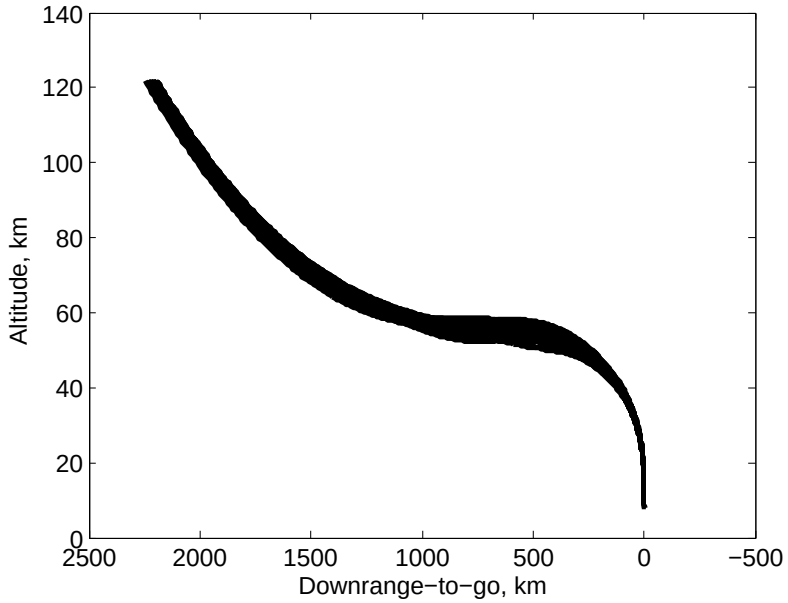

Figure 7.6 Altitude versus downrange-to-go for 100 dispersed trajectories (northbound, 2200 km).

and loft entry cases without difficulty. Also, the load histories for 100 dispersed northbound trajectories are shown in Fig. 7.7. The average peak g-load during the skip phase is about 4.2 g and about 5 g in the final phase. No load-relief maneuvers are taken. However, if necessary, a predictive load-relief strategy developed in Ref. [24] can be applied in the final phase.

In Figs. 7.8-7.9, the final locations of the 10,000 dispersed entry trajectories for two missions are shown. The other northbound missions have plots very similar to Fig. 7.8. In all cases except for one in the northbound long range (10,000 km) mission and a handful in the northbound maxium range (13,519 km), the vehicle's position is within 2.5 km of the landing site at the termination of entry guidance. The failure case for the 10,000 km will be discussed further shortly. The very longest entry case (13,519 km) is at the ultimate limit of this particular algorithm and has multiple failures all corresponding to overshoots.

Table 7.3 summarizes the final position statistics for 10,000 dispersed trajectories for each of the missions in Table 7.1. Table 7.4 summarizes the final position statistics for 10,000 dispersed trajectories for each of the missions in Table 7.2.

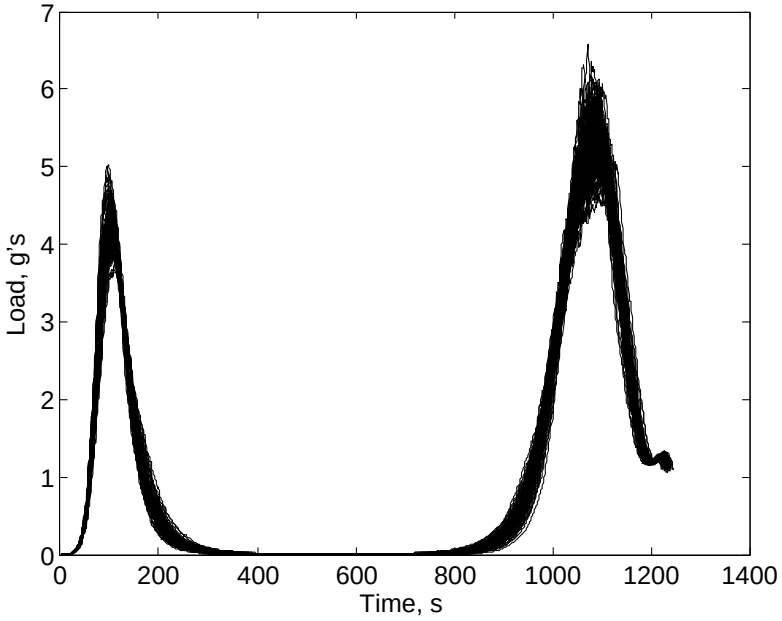

Figure 7.7 Time history of load for 100 dispersed trajectories (northbound, 8400 km).

Table 7.3 Statistics on final miss distances in 10,000 dispersed trajectories (*See discussion)

Miss distance (km)	Northbound Direct	Northbound Short Range	Northbound Medium Range	Northbound Long Range
Minimum	0.072	0.087	0.064	0.073
Maximum	1.584	1.736	1.594	2.335*
Average	0.941	0.890	0.968	0.930
Median	0.911	0.819	0.988	0.888
Standard Deviation	0.316	0.311	0.307	0.423
Number of Misses	0	0	0	1

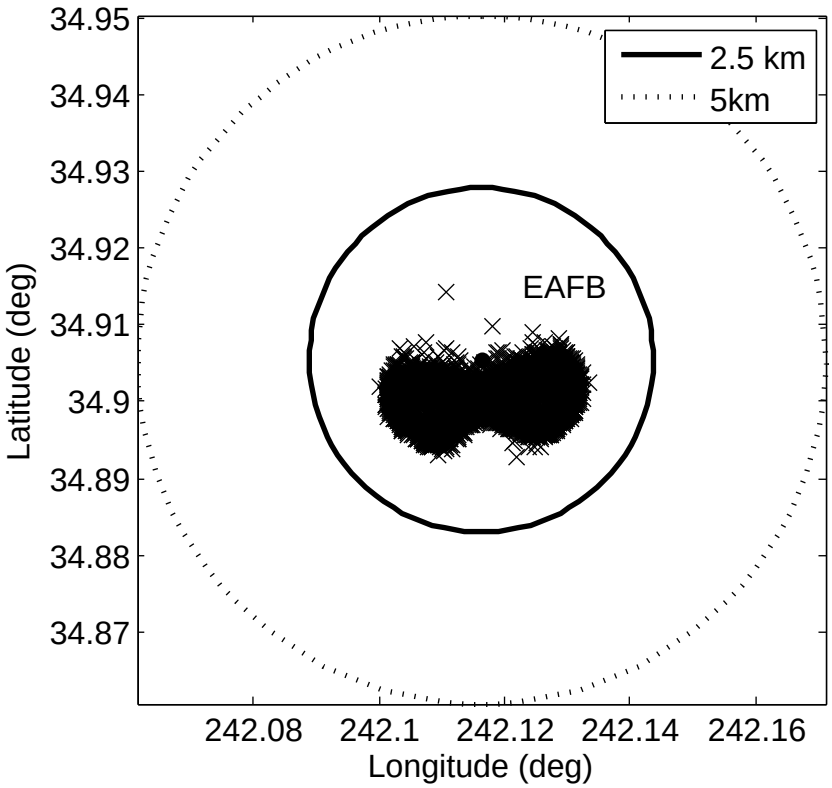

Figure 7.8 Final positions of 10,000 dispersed trajectories for the northbound direct case(2200km).

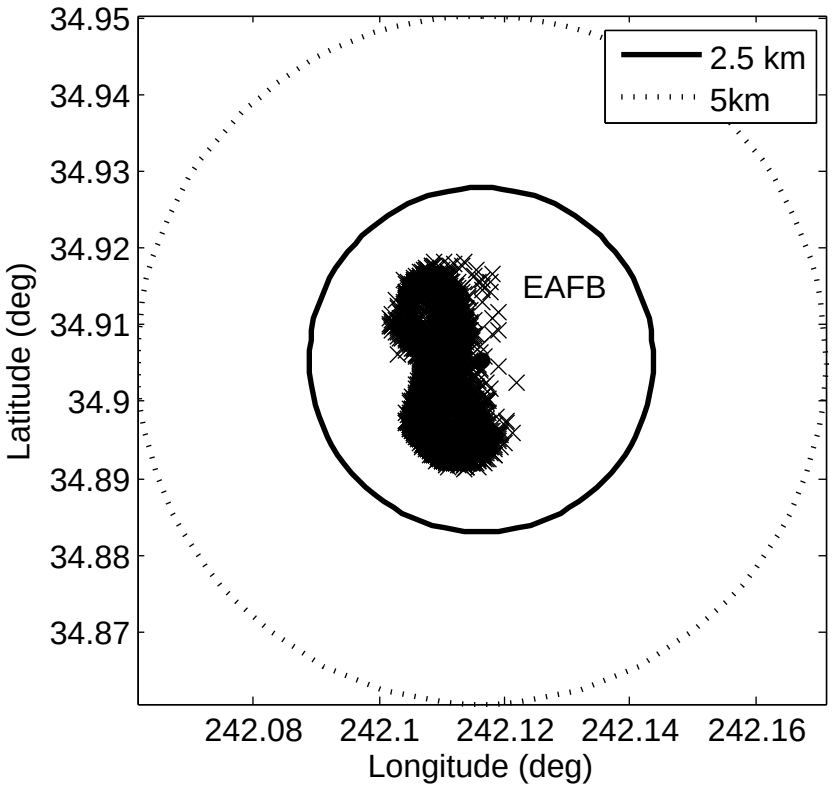

Figure 7.9 Final positions of 10,000 dispersed trajectories for the east-bound medium range case(7300km).

The long range (10,000 km) mission has one case which fails to terminate within 5 km to the landing site. In this case, the trajectory misses the landing site by 15 km. A closer examination reveals that the reasons for the failure are the simultaneous concurrence of a particular pattern of the relatively extreme dispersion in atmospheric density and an appreciable dispersion (about 20% reduction) in L/D. In Fig. 7.10, the ratio of the GRAM 07 dispersed density (ρ_{true}) to the US 76 standard atmosphere (ρ_{model}) is shown for the case. The dispersed atmosphere is over 40% thicker during the downward phase of the skip. The extreme thickness in atmospheric density relative to the modeled value early in the entry trajectory causes the vehicle to fly a relatively small bank angle as seen in Fig. 7.11. Subsequently, the atmospheric density rapidly thins, which, together with the reduction in L/D, requires the vehicle having to fly 180-deg bank angle in order to deplete enough energy during the remainder of the skip phase. However, this maneuver leaves no lateral control on the trajectory. Consequently, as the vehicle enters the final phase after the Kepler phase, the crossrange is larger than expected, despite the targeting bias. The rapid reduction in bank angle in Fig. 7.11 near the end of the trajectory is a reflection of the guidance detecting that there is still a distance to cover to the landing site as the velocity approaches the specified terminal value. This distance, however is due to crossrange error. As the vehicle passes by (laterally) the landing site, the guidance commands full lift-down to quickly terminate the entry trajectory, yielding a miss distance of 15 km.

If the atmospheric density profile could be known in advance to have the shape as in Fig. 7.10, the guidance could use this information so that in the down-control phase, a sufficiently large bank angle would be commanded, and this case would have been saved. Since the actual atmospheric density profile ahead would be impossible to predict closely, it seems unlikely that the predictor-corrector guidance could do more to alleviate such an extreme event. As it currently stands, the guidance achieves 99.998% success rate in this more difficult mission. A risk assessment to examine the likelihood of an event of the above particular nature and magnitudes in long range skip entry missions would help determine if further improvement is warranted.

The entry to KSC pushes the limit of the algorithm's capabilities with the stressful dis-

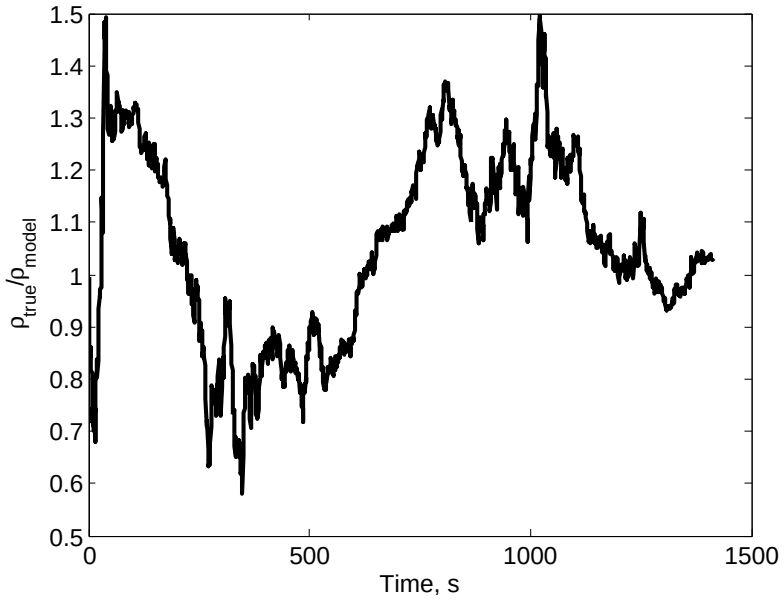

Figure 7.10 Ratio of GRAM 07 atmosphere (ρ_{true}) to US 76 standard atmosphere (ρ_{model}) for the failed northbound long range case (10,000km).

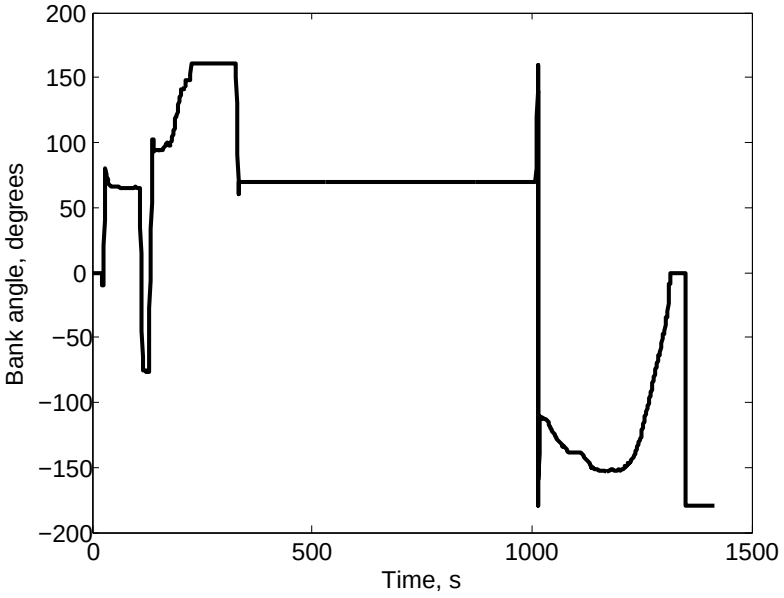

Figure 7.11 Bank angle history for the failed northbound long range case (10,000km).

Table 7.4 Statistics on final miss distances in 10,000 dispersed trajectories (**See discussion)

Miss distance (km)	Eastbound Medium Range	Eastbound Long Range	KSC Max Range
Minimum	0.064	0.113	0.147
Maximum	1.625	1.638	1076**
Average	0.949	0.839	1.694
Median	0.918	0.765	0.935
Standard Deviation	0.326	0.300	24.54
Number of misses	0	0	8

persions. It is remarkable that it still achieves 99.92% accuracy. The particular case that missed by 1076 km severely affects the statistics. The other misses in order of worst to best are 161,150,35,27,23,10, and 9 km. The reasons for these failures are extreme dispersions in nominal parameters.

Very short direct entry cases benefit from some algorithm adjustments. The bank angle actually flown during final phase is around 100°. During the skip planning phase, the guidance predicts an angle of 70°. Due to the small time in "skip" phase (it is a direct entry, not an actual skip), the algorithm cannot converge on an angle that will permit reaching the target. The final phase in planning is flown with too small an angle. However, the guidance is actually turned over to the final phase shortly thereafter, and the final phase guidance is able to reach the landing site. A bank history for one of these cases is shown in Fig. 7.12. An adjustment made is to change the range threshold to a smaller value around 500 km for the skip planning and guidance. Since the skip and final phase guidance is very similar in nature, the skip guidance is able to calculate a viable bank angle and not saturate as in Fig. 7.12. Also, this will not affect the longer range entries as the range-to-go is examined during the first call to the planning. If the range-to-go at this point is less than 3500 km, then the range threshold is adjusted in this manner.

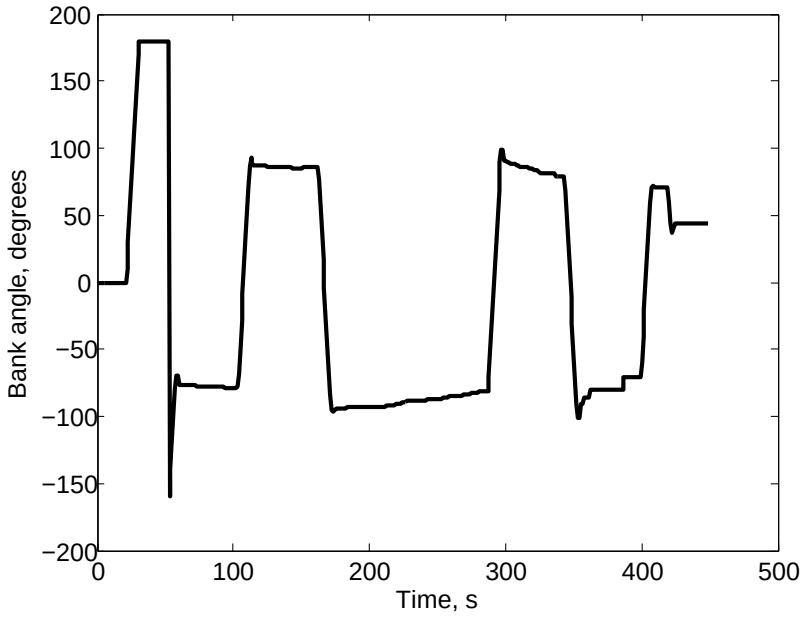

Figure 7.12 Bank angle history of entry case of northbound 2500 km where the skip phase bank angle saturates.

CHAPTER 8. COMPARISON WITH APOLLO SKIP GUIDANCE

8.1 Setup

Improvement is sought for the skip guidance problem. Thus, to show improvement over the Apollo skip guidance, comparison tests are run. This is made possible by the modularity of the new skip algorithm and the Apollo algorithm. The test matrix in Table 8.1 is devised.

Table 8.1 Test matrix

Scenario	Skip Algorithm	Final Phase Algorithm
1	Apollo Skip	Apollo Final
2	NPC Skip	NPC Final from Lu [24]
3	Apollo Skip	NPC Final from Lu [24]
4	NPC Skip	Apollo Final

With the Apollo skip algorithm in hand, a study is performed to see how the Apollo algorithm responds and prove the additional computation necessary in this algorithm is worthwhile. For this additional study, all dispersions are retained as described in Chapter 6.2. However, two changes are made.

First, given that the Apollo skip algorithm has a number of gains that are designed and optimized for the Apollo capsule, this vehicle model is used in all tests. The vehicle has a capsule configuration with a base diameter of 2 meters and weighs 5443 kg (12,000 lb). The vehicle flies a Mach-dependent trim angle of attack profile, which is about 155.7° in the skip phase. The L/D value is 0.3877 at this trim angle-of-attack. The maximum rate and acceleration imposed on the bank angle in the 3DOF simulations are 20°/s and 10°/s^2, respectively, the same as used for the CEV. The Apollo skip algorithm does not have a bank reversal algorithm as this was built into the autopilot, so the one from the earlier testing is

Table 8.2 Initial conditions for comparison tests

State	EAFB	Hawaii	Hawaii	EAFB	EAFB
Altitude, h (km)	121.92	121.92	121.92	121.92	191.92
Longitude, θ (deg)	242.00	154.00	144.40	244.83	242.00
Latitude, ϕ (deg)	12.00	10.00	1.50	-41.13	-55.00
Relative velocity, V(km/s)	10.98	10.63	10.63	10.98	10.98
Flight path angle, γ (deg)	-5.576	-6.5	-6.0	-5.576	-5.576
Heading angle, ψ (deg)	0.47	59.78	56.9	0.47	0.47
Downrange, s (km)	2500	3600	5000	8400	10,000
Crossrange, χ (km)	9.7	-82.4	-22.9	298	42

used.

Second, the analytic atmospheric dispersion is used, rather than GRAM 2007 due to code and time constraints. As shown in Chapter 6, the dispersions are similar.

Several different entry mission scenarios are tested. They correspond to different downrange distances and approach directions, resulting from different lunar return conditions. The initial conditions are given in Table 8.2. All the missions are run under exactly the same setting, without any mission-dependent adjustments of guidance parameters or logic.

The landing site varies slightly under these conditions. Three cases from the earlier skip testing were retained, with EAFB still the landing site. Two other cases with landing just north of the Hawaiian Islands at (184.613°E, 23.573°N) are used.

8.2 Results and Discussion

With the four potential guidance scenarios and five potential reentry scenarios, there are twenty different sets of statistics. The statistics for the different cases are outlined in Tables 8.3-8.7.

The simulations show that the complete Apollo algorithm works for shorter trajectories, but performs very poorly as the range increases. The NPC algorithms are successful at all ranges. When mixing the different guidance modules, the NPC-Apollo performs well in the middle ranges, but not as well for short and long ranges. The Apollo-NPC combination is adequate at short ranges, but not at longer ranges where the Apollo skip guidance does not

Table 8.3 Statistics on final miss distances in 500 dispersed trajectories for 2500 km EAFB case

Miss distance (km)	Apollo-Apollo	NPC-NPC	Apollo-NPC	NPC-Apollo
Minimum	0.922	0.195	0.089	N/A
Maximum	34.45	1.596	1.963	N/A
Average	1.813	0.952	0.909	N/A
Median	1.768	0.916	0.915	N/A
Standard Deviation	1.517	0.357	0.375	N/A
5km>Missed>2.5km	0	0	0	0
Missed>5km	1	0	0	500
Skipouts	0	0	0	500

Table 8.4 Statistics on final miss distances in 500 dispersed trajectories for 3600 km Hawaii case

Miss distance (km)	Apollo-Apollo	NPC-NPC	Apollo-NPC	NPC-Apollo
Minimum	0.263	0.255	0.053	0.428
Maximum	283.5	1.859	264.5	154.7
Average	7.168	0.908	4.102	16.01
Median	1.852	0.850	0.974	7.467
Standard Deviation	29.42	0.348	20.18	23.20
5km>Missed>2.5km	14	0	0	37
Missed>5km	31	0	19	288
Skipouts	0	0	0	0

Table 8.5 Statistics on final miss distances in 500 dispersed trajectories for 5000 km Hawaii case

Miss distance (km)	Apollo-Apollo	NPC-NPC	Apollo-NPC	NPC-Apollo
Minimum	0.617	0.255	0.166	0.921
Maximum	692.3	1.859	549.4	2.325
Average	204.9	0.908	94.76	1.744
Median	208.7	0.850	67.32	1.750
Standard Deviation	137.3	0.348	100.4	0.314
5km>Missed> 2.5km	3	0	7	0
Missed > 5km	466	0	351	0
Skipouts	0	0	0	0

Table 8.6 Statistics on final miss distances in 500 dispersed trajectories for 8400 km EAFB case

Miss distance (km)	Apollo-Apollo	NPC-NPC	Apollo-NPC	NPC-Apollo
Minimum	0.892	0.084	0.214	0.996
Maximum	1416	1.884	1398	32.22
Average	428.9	0.958	406.4	1.749
Median	434.9	0.930	413.2	1.665
Standard Deviation	239.8	0.355	236.9	1.412
5km> Missed> 2.5km	0	0	0	1
Missed >5km	466	0	466	2
Skipouts	0	0	0	0

Table 8.7 Statistics on final miss distances in 500 dispersed trajectories for 10,000 km EAFB case

Miss distance (km)	Apollo-Apollo	NPC-NPC	Apollo-NPC	NPC-Apollo
Minimum	0.872	0.195	0.109	0.912
Maximum	2394	2.283	2355	128.2
Average	1293	0.965	1264	1.988
Median	1503	0.935	1487	1.694
Standard Deviation	763.7	0.358	747.7	5.678
5km >Missed >2.5km	2	0	0	4
Missed> 5km	459	0	465	3
Skipouts	0	0	0	0

perform well. The reasons for these failures are outlined below, though some discussion has already been made in Chapter 3.

The Apollo final phase guidance follows a reference trajectory based on range to the target. If the skip algorithm guides the vehicle to a reasonable final phase starting condition, then the Apollo final phase algorithm is generally successful. However, the Apollo skip algorithm is not successful with the longer ranges regardless of the final phase guidance algorithm that is used. This is a well-known limitation of the Apollo algorithm [10]. The Apollo algorithm calculates incorrect exit conditions for the upcontrol phase (the exit point of the skip). Also, it does not take into account the finite amount of drag that occurs during the Kepler phase for longer range skip entries. The Apollo skip algorithm was designed for shorter ranges, thus the long time spent in the Kepler phase adds a significant amount of unexpected drag to the trajectory. Extensive further analysis and details are available in the work by Bairstow [9] and were discussed in Chapter 3.

It is found that choice of entry flight path angle strongly affects the accuracy and whether or not a landing is made in the case of the Apollo algorithm. If the entry is too shallow, the vehicle can skip out of the atmosphere; too steep an entry will cause the g-load to exceed vehicle limits. It is found in testing, that depending on the combination of guidance algorithms, certain initial flight path angles are more conducive to accurate landings than others. For example, in the short entry case of 3600 km, with an entry flight path angle of -5.65° produces the statistics in Table 8.8. Compare these results with those in Table 8.4. Clearly, the Apollo algorithm is not as able to handle the change in flight path angle as well as the NPC algorithms are. The NPC algorithm still achieves 100% success, however the Apollo algorithm has a failure rate of 62% with the revised flight path angle as compared to the 7% failure rate with the earlier flight path angle.

The combination of Apollo-NPC and NPC-Apollo are not optimized for phase transition. This fact causes a significant number of failures. The NPC skip always transitions to final phase when the range-to-go is 2,000 km. The Apollo skip guidance transitions in a fashion depending on the trajectory type. For skip entries, where there is a Kepler phase, the final phase starts

Table 8.8 Statistics on final miss distances in 500 dispersed trajectories for 3600 km Hawaii case with revised FPA.

Miss distance (km)	Apollo-Apollo	NPC-NPC	Apollo-NPC	NPC-Apollo
Minimum	0.210	0.117	0.123	190.9
Maximum	912.6	1.653	739.6	2144
Average	237.6	0.936	3.135	758.8
Median	200.2	0.898	0.935	747.1
Standard Deviation	224.8	0.355	33.69	299.7
5km>Missed>2.5km	4	0	0	0
Missed>5km	375	0	9	500
Skipouts	0	0	1	0

when the drag is greater than 6.5 ft/s^2 (about $0.2g_0$). For loft entries, if the vehicle is going down and the velocity is less than a calculated amount (the predicted upcontrol exit velocity plus 500 ft/s), then final phase is started. For direct type entries, final phase is started if the predicted upcontrol exit velocity is less than 18,000 ft/s. These transitions typically occur when the range is much less as the Apollo final phase reference trajectory has a maximum starting range of about 1500 km. Thus the Apollo final phase guidance is not expecting to be handed the reins which that much range-to-go. This is a problem in the shortest cases, because the guidance is handed over to the Apollo final phase earlier than expected. The result is saturation of the guidance command. Initially, the guidance commands a small bank angle, but then quickly switches to full lift down and the vehicle skips out of the atmosphere. The bank history for this case is shown in Fig. 8.1.

Fig. 8.1 shows a brief "skip" phase where the vehicle flies the initial entry (0°). Once the range-to-go is less than 2000 km, the guidance switches to final phase. At this point, the vehicle still has a velocity of near its initial entry velocity of 11 km/s (36,000 ft/s) and flight path angle of -3.15°. The Apollo final phase reference trajectory was designed with the initial velocity expected to be around 7.16 km/s (23,500 ft/s) and flight path angle of -2.0°. Most other cases start with the velocity and flight path angle in the neighborhood of the reference condition as shown in Table 8.9. However, the large difference seen in the 2500 km case causes the bank command to saturate. It is initially very small, but then quickly change to a large value (near full lift down) suggesting the vehicle is on the edge of the reference trajectory.

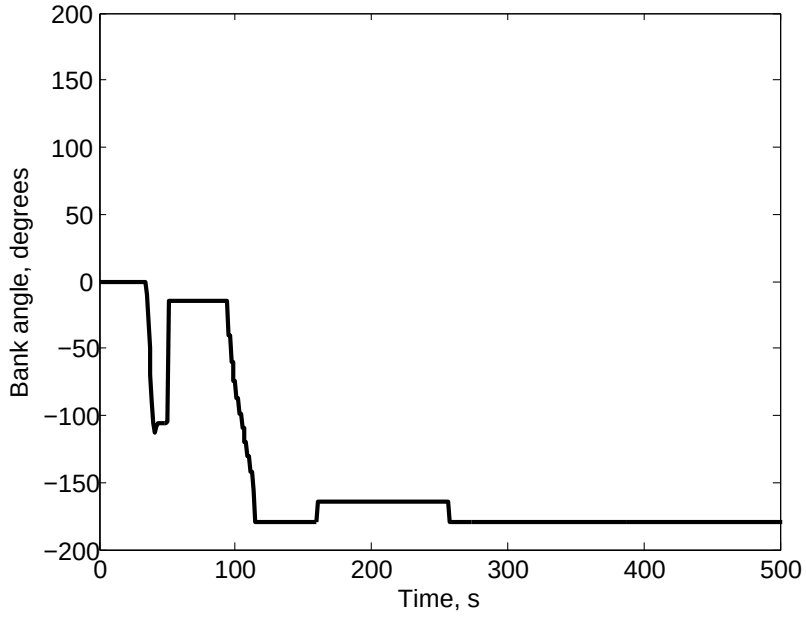

Figure 8.1 Bank angle history for NPC-Apollo 2500 km entry case.

Table 8.9 Starting conditions for Apollo final phase

	Ref. Cond.	2500km NPC-Apollo	2500 km Apollo-Apollo	5000km NPC-Apollo	5000km Apollo-Apollo
Velocity (km/s)	7.16	11.0	7.63	6.98	6.81
Flight Path Angle (deg)	-2.0	-3.15	-1.05	0.00	-1.21
Drag (g's)	0.186	0.304	3.89	0.10	0.19

Eventually the vehicle overshoots the target and skips out of the atmosphere. The remaining cases in Table 8.9 are successful in guiding the vehicle to the target conditions. The main difference in the 2500 km case appears to be due to different starting velocity in final phase.

The phase transition from Apollo skip guidance to NPC final phase is able to deal with the wider range of initial conditions and thus is more successful at guiding the vehicle in this case. However, in longer range cases, the Apollo skip guidance performs so poorly during skip phase, that even the numerical final phase is unable to reach the target.

In summary, the NPC-NPC algorithm is 100% successful for all entry ranges. It is clear that the Apollo skip guidance fails for longer range cases. Similarly, the Apoll final phase

guidance has difficulty in the case of extreme diversions from the reference trajectory. This is not a problem with the numerical final phase guidance, since it generates a reference trajectory at every guidance step.

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

A method of skip entry guidance for a vehicle with a low L/D ratio is developed. The approach combines both onboard skip trajectory planning and closed-loop guidance in one. Unlike the Apollo skip entry guidance, the algorithm computes the bank angle required to achieve the final range condition and the corresponding trajectory based on numerical solution of the full 3DOF dynamics, without further assumptions or simplifications. No particular patterns of the trajectory are assumed. As a result, the algorithm is able to transition seamlessly between different phases of the skip trajectory and reliably adapts to different trajectory patterns, regardless of whether the entry trajectory actually skips out of the atmosphere. Several implementation issues are found to be critical to the performance and robustness of the algorithm. They include appropriate direction for bank angle reversals, automated targeting bias for long skip trajectories, and proper utilization of lift and drag acceleration filters for enhanced robustness with respect to modeling uncertainties. The success of the algorithm as compared to the Apollo skip entry guidance in extensive, stressful 3DOF end-to-end dispersion simulations clearly demonstrates the high performance and robustness of the algorithm for a wide range of mission scenarios.

9.2 Future Work

The skip guidance algorithm is very successful at guiding the vehicle to the target. However, a number of measures and further studies are possible to improve the algorithm.

Also, the secondary objectives desired of the guidance include minimizing load and heat to the vehicle. One possibility for these during the final phase is the method discussed in Lu [24].

Generally during skip phase, the load is never too high. G-load limits are exceeded mainly in the final phase when the guidance commands large bank angles (with lift down). This occurs when the vehicle is in danger of overshooting the target. One possible load mitigation strategy for the skip phase could be adjustment of the angle flown during the initial entry and the Kepler phases. This only has a very small impact on the trajectory though.

Integrated heat load can be fairly large due to the length of time of the trajectory. For short range entries, the maximum heat spike will need to be examined. Measures to reduce these similar to the load reduction could be taken.

To minimize fuel usage, fewer bank reversals are necessary. This can be accomplished by widening the corridor of acceptable crossrange. However, this may cause the biasing logic to have greater difficulty in finding an appropriate biased target point.

Further study on dispersions can be performed. For instance, imprecise navigation data could be fed into the planning algorithm to generate uncertainty on the vehicle state. In a similar vein, a 6DOF simulation could be tested with the vehicle not able to perfectly fly the commanded bank angle. A small test was performed adding some Gaussian noise to the commanded bank angle. Due to the nature of the predictor-corrector algorithm, it was able to continually update the command to permit reaching the target successfully.

Generally, the C_L and C_D dispersions are dependent on the Mach number. For this study, a constant bias was used. As a Mach-dependent dispersion would receive benefit from averaging effects, the constant dispersion used in this study is more stressful on the guidance algorithm. Still, a study of this type would benefit from greater reality.

Using an exact line search may remove some of the issues encountered in converging on a solution. It may also converge faster in terms of the number of secant iterations required. Due to the greater computational expense, the cost in time of computation may not decrease since the number of skip trajectories that need to be planned will increase.

APPENDIX A. MATLAB SIMULATION ENVIRONMENT CODES

Secant Dispersion

This is the file that simulates one or more dispersed trajectories of the vehicle.

```
function secant_disp(ki,kf)
%This part is the "script" that selects the dispersed variables
%and calls the function that runs a simulation for one set of dispersed
%variables.
clear all;clc;close all;
if nargin==0
    clear all;
end
global bias_atm Mag_atm1 omega_atm1 theta_atm Mag_atm2 omega_atm2
global CL_bias CD_bias lon_site lat_site lon_site_bias lat_site_bias

vehicle=2;  %1=CEV, 2=Apollo
dispersions=1; %1 is on, 0 is off
if nargin==0
    ki=1; %range of seeds to use, ki if initial, kf is final
    kf=1;
end
filename='test'; %name of file for saving data
entrycase=5;     %which set of entry conditions and landing site
scenario=4;  %which guidance pathway
% Scenarios
% 1=Full Apollo Guidance
% 2=Full NPC Guidance
% 3=Apollo Skip, NPC Final
% 4=NPC Skip, Apollo Final

%Earth parameters
R0=6378136.3;                       %Earth Radius, m
g0=9.80665;                         %Earth gravity, m/s^2
Vscale=sqrt(R0*g0);                 %Scale Velocity = 7908.4m/s

%Conversion factors
m2ft=3.28083989501;                 %meters to feet
ft2m=1/m2ft;                       %feet to meters
d2r=0.01745329251994;          %degrees to radians = pi/180;
r2d=57.29577951308232;         %radians to degrees= 180/pi;

%default entry conditions
altft=400000;                    %feet
```

```
Lon_entry=244.826690629075;     %degrees
Lat_entry=-41.1330892463129;    %degrees
Vftpers=36013.7885790865;       %velocity
gammaentry=-5.57559182971711;   %flight path angle
headingentry=0.468530972644024; %heading angle (zero is north, pos CW)

%Default Landing site (Edwards AFB)
lon_site=242.1163;              %deg
lat_site=34.9055;               %deg

if entrycase==1     %Nominal Entry and landing condtions, 8400 km downrange
    altft=400000;
    Lon_entry=244.826690629075;
    Lat_entry=-41.1330892463129;
    Vftpers=36013.7885790865;
    gammaentry=-5.57559182971711;
    headingentry=0.468530972644024;
    %Entry condition disperson co-variance matrix, nominal entry
    stmEI =[ 0.3059   -0.4070    0.0956    5.9425    1.0608    1.1057;
        -0.1782   -2.9285    1.8762    8.5371  -20.2060   23.7063;
        0.0874   -0.2913    0.5757    0.9753   -0.7515    9.4148;
        -0.1418   -1.4342    0.8705    4.5581  -10.7264    9.4930;
        -0.5409    0.4266   -0.0646   -7.5863   -1.5741   -0.5982];
elseif entrycase==2    %Northbound entry for 4200 km downrange (loft)
    Lon_entry=244;
    Lat_entry=-3;
    stmEI =[0.2489   -0.2441    0.1514    4.3169    1.3704    1.4268;
        0.3246   -0.9266    3.0883    2.1815   -1.9421   32.1002;
        0.2876    0.2195    0.6113   -1.0413    4.9439    7.9926;
        0.0543   -0.5090    1.4626    1.8073   -2.8445   14.6425;
        -0.3331    0.1462    0.0634   -3.8809   -0.8470    0.8406];
elseif entrycase==3%Northbound entry, 2500km downrange, (small skip)
    Lon_entry=242;
    Lat_entry=12;
    stmEI =[0.2537   -0.2116    0.1586    4.3394    1.6033    1.4141;
        0.5168    0.0011    3.2164   -0.3803    5.9975   31.6737;
        0.3519    0.3919    0.5497   -1.5034    6.7635    6.4078;
        0.1381   -0.0661    1.5301    0.5419    0.8167   14.9921;
        -0.2503    0.1207    0.1334   -2.8442   -0.0490    1.3991];
elseif entrycase==4   %Northbound entry, 2200km downrange, (direct)
    Lon_entry=242;
    Lat_entry=15;
    stmEI =[0.2582   -0.2050    0.1591    4.3713    1.6909    1.4019;
        0.5582    0.1907    3.2161   -0.8792    7.5690   31.3289;
        0.3605    0.4253    0.5328   -1.6185    7.0680    6.0359;
        0.1576    0.0242    1.5312    0.3030    1.5627   14.9395;
        -0.2338    0.1175    0.1462   -2.6321    0.1173    1.4878];
elseif entrycase==5     %Northbound entry for 10,000km downrange
    Lon_entry=242;
    Lat_entry=-55;
    stmEI =[ 0.3657   -0.5397    0.0634    7.8211    0.9983    0.8793;
        -0.5225   -3.3630    1.1913    8.8254  -25.5364   17.7734;
        -0.0214   -0.4587    0.4951    1.5415   -2.9583    8.8957;
        -0.2837   -1.6223    0.5418    4.4870  -12.8322    6.4257;
        -0.6275    0.6506   -0.0612  -10.2017   -1.2937   -0.7876];
elseif entrycase==6    %Eastbound entry, 7300km downrange
```

```
    altft=400000;
    Lon_entry=176.988152;
    Lat_entry=7.135801517;
    Vftpers=34787.29993376976;
    gammaentry=-5.987650659018430;
    headingentry=54.628060344205217;
    stmEI =[ 0.1795   -2.9172    1.4579   16.6514  -16.9590   12.9336;
        0.2173   -1.7634    1.1279    9.5158   -8.7220   13.0738;
        0.4912   -0.7896    0.3258    9.3439   -0.8265    3.7542;
        0.0456   -1.5729    0.9019    7.5047  -10.4114    8.8236;
        0.0705   -0.2148    0.4570    1.5323   -0.4078    5.1000];

elseif entrycase==7    %Eastbound entry, 10,000km downrange
    altft=400000;
    Lon_entry=158;
    Lat_entry=-7;
    Vftpers=34787.29993376976;
    gammaentry=-5.987650659018430;
    headingentry=54.628060344205217;
    stmEI =[-1.1682   -2.7782    1.3092    7.2926  -22.6977   12.7784;
        -0.6887   -1.6684    1.0289    3.1898  -12.6107   12.9732;
        0.1005   -0.8839    0.3731    7.5692   -4.2979    5.1503;
        -0.7126   -1.4714    0.8084    1.7680  -13.1186    8.2080;
        0.1676    0.5184    0.0993   -1.7337    4.4869    1.8135];
elseif entrycase==8 %3600km Apollo special case
    %Nominal Entry and landing condtions
    altft=400000;
    Lon_entry=154;
    Lat_entry=10;
    Vftpers=34888.2526525112;
    headingentry=59.7846978972999;
    gammaentry=-6.5; %for testing (works well in scenario 4)
    %Landing site
    lon_site=184.613;
    lat_site=23.573;

    % Entry condition disperson co-variance matrix
    stmEI =[-0.9635   -3.0094    1.3440    9.8926  -23.9346   11.6823;
        -0.3732   -1.5413    0.8966    5.3189  -10.4169   10.8597;
        0.1624   -0.9360    0.2571    8.4896   -4.5655    2.7681;
        -0.5498   -1.5364    0.7938    3.3161  -13.0617    7.7529;
        -0.0572   -0.4368    0.5202    2.1350   -2.6031    5.6410];
elseif entrycase==9 %Apollo special, 5000 km
    %Nominal Entry and landing condtions
    altft=400000;
    Lon_entry=144.4;
    Lat_entry=1.5;
    Vftpers=34888.2526525112;
    gammaentry=-6;
    headingentry=56.9;

    %Landing site
    lon_site=184.613;
    lat_site=23.573;

    % Entry condition disperson co-variance matrix
```

```
    stmEI =[ -1.5500   -2.5711    1.3231    3.9620  -23.9557   12.1969;
        -0.8059   -1.4668    0.9859    2.1001  -12.0007   12.1276;
        -0.0333   -0.9260    0.3232    7.1721   -5.8820    4.0330;
        -0.8919   -1.3396    0.8137    0.0109  -13.2062    8.1124;
        0.0523    0.0516    0.3104    0.2178    1.0365    3.8263];
elseif entrycase==10 %Approach to KSC 13,519km downrange (7300nm)
    altft=4.0644e+005;
    Lon_entry=166.0000;
    Lat_entry=-28.6500;
    Vftpers=3.4976e+004;
    gammaentry=-6.2296;
%     headingentry=78.9862; %(from Tigges CEV paper, too far left)
    headingentry=74;
    stmEI =[-0.9738   -4.1202    0.4209   14.1195  -30.8349    6.8046;
   -0.3368   -0.8458    0.2575    0.8223   -7.5102    5.7583;
    0.1658   -0.9302    0.2528    7.8584   -4.4699    4.6700;
   -0.5159   -1.7535    0.1944    4.2470  -14.5926    2.4623;
    0.3272    2.1482    0.0486   -8.8792   15.2327   -0.0977];
    %Landing site (KSC)
    lon_site=280.651;
    lat_site=28.585;

end
%Vehicle Parameters
if vehicle==1
    mass_nom  = 18464.4*0.454;          % mass of the CEV (kg)
    Sref = 19.648;                      % reference area of the CEV (m^2)
elseif vehicle==2
    mass_nom = 5443.10844;                 %kilograms Apollo
    Sref  = 12.01707457427626;             %m^2 Apollo
end

lon_site_bias=lon_site;
lat_site_bias=lat_site;

%Non-dimensionalization of Entry/Initial Conditions
altitude=altft*ft2m;                   %m
r1=(R0+altitude)/R0;                   %dimensionless
theta0=Lon_entry*d2r;                 %initial longitude (radians)
phi0=Lat_entry*d2r;                   %initial latitude(radians)
V0=Vftpers*ft2m/Vscale;                %Velocity  -> dimensionless
gamma0=gammaentry*d2r;                %initial flight path angle (radians)
psi0=headingentry*d2r;                %initial heading wrt north (radians)

origstate=[r1,theta0,phi0,V0,gamma0,psi0,0];

% Entry condition disperson co-variance matrix
%            [stmEI]=icperturbation(origstate,mass_nom,Sref,vehicle)
%   here are the "inputs":
% deltaPosX        0. N 33.   # 1-sigma, m
% deltaPosY        0. N 33.   # 1-sigma, m
% deltaPosZ        0. N 33.   # 1-sigma, m
% deltaVelX        0. N .33   # 1-sigma, m/s
% deltaVelY        0. N .33   # 1-sigma, m/s
% deltaVelZ        0. N .33   # 1-sigma, m/s
```

```
% The outputs (of cov. Matrix times the delta vector) are the
% perturbations in terrestrial latitude (deg), terrestrial longitude
% (deg), relative speed (m/s), relative flight path (deg), relative
% heading (deg).

load -ascii seeds\r_Seeds.dat % 2000x3, zero-mean normal dist, 1-sigm= 10 m
load -ascii seeds\V_Seeds.dat % 2000x3, zero-mean normal dist,1-sigm=0.1m/s
load -ascii seeds\Seeds_mass.dat   % 2000, uniform in [0,1.0]
load -ascii seeds\Seeds_atmsphr.dat% 2000x2, uniform in [0,1.0]
load -ascii seeds\Seeds_aero_normal.dat % 1000x2, uni in normal dist[-1,1]

%=========================================================================
if ki-kf==0
    plots=1;
else
    plots=0;
    CL_error=zeros(1,kf-ki+1);
    CD_error=zeros(1,kf-ki+1);
    Mass_error=zeros(1,kf-ki+1);
    Bias_error=zeros(1,kf-ki+1);
    Mag_error1=zeros(1,kf-ki+1);
    Omega_freq1=zeros(1,kf-ki+1);
    Mag_error2=zeros(1,kf-ki+1);
    Omega_freq2=zeros(1,kf-ki+1);
    Total_mag=zeros(1,kf-ki+1);
    down_miss=zeros(1,kf-ki+1);
    final_long=zeros(1,kf-ki+1);
    final_lat=zeros(1,kf-ki+1);
    dlat=zeros(1,kf-ki+1);
    dlon=zeros(1,kf-ki+1);
    dV=zeros(1,kf-ki+1);
    dgamma=zeros(1,kf-ki+1);
    dpsi=zeros(1,kf-ki+1);
end
for j=1:kf-ki+1
    p=ki+j-1; %Obtains correct seed

    if plots==0
        p
    end
    %This part produces the initial entry condition dispersions
    DEOstate = [r_Seeds(p,:)/R0*4.5 V_Seeds(p,:)/Vscale*4.5]';
    DEIstate = stmEI*DEOstate;
    dlon(j)  = DEIstate(1);
    dlat(j)  = DEIstate(2);
    dV(j)    = DEIstate(3);
    dgamma(j)= DEIstate(4);
    dpsi(j)  = DEIstate(5);

    %If we don't want dispersions, zero out the delta's
    if dispersions==0
        dlon(j)=0;
        dlat(j)=0;
        dV(j)=0;
        dgamma(j)=0;
        dpsi(j)=0;
```

```
end

%Add dispersions to nominal state values to get perturbed initial cond
state(1) = origstate(1);
state(2) = origstate(2)+dlon(j);
state(3) = origstate(3)+dlat(j);
state(4) = origstate(4)+dV(j);
state(5) = origstate(5)+dgamma(j);
state(6) = origstate(6)+dpsi(j);
state(7) = origstate(7); %downrange flown

%Determine perturbed value of mass.
mass_seed = Seeds_mass(p,1)-0.5;
mass_pert = mass_nom*(1+(mass_seed*2)*0.05); %max +/-5%

% Add atmospheric uncertainty for analytic dispersion
seed1 = Seeds_atmsphr(p,1)-0.5;
seed2 = (Seeds_atmsphr(p,2)-0.5)*1.8;
seed3 = Seeds_atmsphr(p,3)*.75+.25;
seed4 = Seeds_atmsphr(p,4)-.5;
seed5 = Seeds_atmsphr(p,5);

if seed2>=0
    seed2=seed2+1;
elseif seed2<0
    seed2=seed2-1;
end
bias_atm  = 2*seed1*0.20;
Mag_atm1   = seed2*0.10;

% (pos values give thick atm in mid alt,
%       neg values give thin atm in mid alt,
%       zero values give no alt-dep dispersion)
%     omega_atm=  0.046889442590892;  % pi/67 is default
%     theta_atm= -2.180359080476499; %pi/2-80*pi/67 is default

max_periods1=2;
%Values should permit max 2 periods over 120km
omega_atm1 = seed3*(max_periods1*2*pi/121.9);
Mag_atm2= 2*seed4 * .05*Mag_atm1;
max_periods2=50;
%Values should permit max 100 periods over 120km
omega_atm2 = seed5*(max_periods2*2*pi/121.9);

%Calculates phase of sinusoid dispersion.  If possible, dispersion will
%be non-existent at surface, otherwise try to minimize dispersion at
%surface.
if abs(bias_atm/Mag_atm1)<=1
    theta_atm=asin(-bias_atm/Mag_atm1);
else
    theta_atm=pi/2;
    diff1=abs(bias_atm+Mag_atm1*sin(theta_atm));
    diff2=abs(bias_atm+Mag_atm1*sin(-theta_atm));
    if diff1>diff2
        theta_atm=-theta_atm;
    end
```

```
    end

    % Add aerodynamic uncertainty
    seed6=Seeds_aero_normal(p,1)/2;
    seed7=Seeds_aero_normal(p,2)/2;

    if vehicle==1 %CEV
        CL_nom=.3892;
        CD_nom=1.3479;
    elseif vehicle==2 %Apollo
        CL_nom = 0.4717;
        CD_nom = 1.2167;
    end

    CL_bias   = 2*seed6*CL_nom*.20;  %max CL_bias 20%, uniformly
    CD_bias   = 2*seed7*CD_nom*.20;  %max CD_bias 20%, uniformly

    %As before,if no dispersions needed,zero out everything and reset mass.
    if dispersions==0
        bias_atm=0;
        Mag_atm1=0;
        Mag_atm2=0;
        CL_bias=0;
        CD_bias=0;
        mass_pert = mass_nom*1;
    end

    %Record values for later plotting
    CL_error(j)=(CL_bias+CL_nom)/CL_nom*100;
    CD_error(j)=(CD_bias+CD_nom)/CD_nom*100;
    Mass_error(j)=(mass_pert-mass_nom)/mass_nom*100+100;
    Bias_error(j)=bias_atm*100+100;
    Mag_error1(j)=Mag_atm1*100+100;
    Omega_freq1(j)=seed3*max_periods1;
    Mag_error2(j)=Mag_atm2*100+100;
    Omega_freq2(j)=seed5*max_periods2;
    Total_mag(j)=(abs(Mag_atm1)+Mag_atm2)*100+100;

    %Simulate dispersed trajectory
    [down_miss(j),final_long(j),final_lat(j)]=simulate_entry...
        (state,mass_nom,mass_pert,Sref,vehicle,plots,scenario);

    [range234,Psi234] = sphereping(state(2),state(3),lon_site*pi/180,...
lat_site*pi/180);
    CR = asin(sin(range234)*sin(state(6) - Psi234));% crossrange in rad
    CR_km(j)=CR*R0/1000;
    Actual_CL(j)=CL_bias;
    Actual_CD(j)=CD_bias;
    Actual_mass(j)=mass_pert;
    Actual_bias(j)=bias_atm;
    Actual_mag(j)=Mag_atm1;
    Actual_freq(j)=omega_atm1;
%                                   down_miss(j)=p;
%                                   final_long(j)=0;
%                                   final_lat(j)=0;
    lon_site_bias=lon_site; %reset biased landing sites before next run
```

```
    lat_site_bias=lat_site;
    clear functions %reset persistent variables in ext func before next run
end

if plots==0
    indices=ki:kf;
    data=[indices;down_miss;CL_error;CD_error;Mass_error;Bias_error;...
Mag_error1;Omega_freq1;Mag_error2;Omega_freq2;Total_mag;...
final_long;final_lat];
    if ki==1
        save(filename,'data','-ascii')
    end
end

if kf-ki~=249 && ki-kf~=0
    figure
    plot(ki:kf,down_miss,'x')
    xlabel('index')
    ylabel('Miss distance (km)')

    figure
    subplot(3,3,1)
    plot(CL_error,down_miss,'x')
    xlabel('CL')
    ylabel('Miss Distance(km)')
    subplot(3,3,2)
    plot(CD_error,down_miss,'x')
    xlabel('CD')
    ylabel('Miss Distance(km)')
    subplot(3,3,3)
    plot(Mass_error,down_miss,'x')
    xlabel('mass')
    ylabel('Miss Distance(km)')
    subplot(3,3,4)
    plot(Bias_error,down_miss,'x')
    xlabel('Bias Dispersion')
    ylabel('Miss Distance(km)')
    subplot(3,3,5)
    plot(Mag_error1,down_miss,'x')
    xlabel('Mag1 Dispersion')
    ylabel('Miss Distance(km)')
    subplot(3,3,6)
    plot(Omega_freq1,down_miss,'x')
    xlabel('Number of Atm Periods1')
    ylabel('Miss Distance(km)')
    subplot(3,3,7)
    plot(Mag_error2,down_miss,'x')
    xlabel('Mag2 Dispersion')
    ylabel('Miss Distance(km)')
    subplot(3,3,8)
    plot(Omega_freq2,down_miss,'x')
    xlabel('Number of Atm Periods2')
    ylabel('Miss Distance(km)')
    subplot(3,3,9)
    plot(Total_mag,down_miss,'x')
    xlabel('Total Sin Magnitude')
```

```
    ylabel('Miss Distance(km)')

    figure
    earthradius = almanac('earth','radius','km');
    [latc,longc] = scircle1(lat_site,lon_site,2.5,[],earthradius);
    [lat5,long5] = scircle1(lat_site,lon_site,5,[],earthradius);
    plot(longc+360,latc,'r--',long5+360,lat5,'k:',final_long,final_lat,...
'x',lon_site,lat_site,'r*')
    xlabel('Longitude (deg)','Fontsize',12)
    ylabel('Latitude (deg)','Fontsize',12)
    legend('2.5 km','5km')
    set(gca,'Fontsize',12)
    axis tight square

    figure
    subplot(2,3,1)
    plot(ki:kf,Actual_CL,'x')
    ylabel('CL Bias')
    xlabel('index')

    subplot(2,3,2)
    plot(ki:kf,Actual_CD,'x')
    ylabel('CD Bias')
    xlabel('index')

    subplot(2,3,3)
    plot(ki:kf,Actual_mass,'x')
    ylabel('True Mass (kg)')
    xlabel('index')

    subplot(2,3,4)
    plot(ki:kf,Actual_bias,'x')
    ylabel('Bias Dispersion')
    xlabel('index')

    subplot(2,3,5)
    plot(ki:kf,Actual_mag,'x')
    ylabel('Mag Dispersion')
    xlabel('index')

    subplot(2,3,6)
    plot(ki:kf,Actual_freq,'x')
    ylabel('Actual Frequency (rad/km)')
    xlabel('index')

    std(dlon*r2d)*3
    std(dlat*r2d)*3
    std(dV*Vscale)*3
    std(dgamma*r2d)*3
    std(dpsi*r2d)*3
    figure
    subplot(2,3,1)
    plot(ki:kf,dlon*r2d,'x')
    xlabel('index')
    ylabel('Longitude Dispersion (deg)')
    subplot(2,3,2)
```

```
    plot(ki:kf,dlat*r2d,'x')
    xlabel('index')
    ylabel('Latitude Dispersion (deg)')
    subplot(2,3,3)
    plot(ki:kf,dV*Vscale,'x')
    xlabel('index')
    ylabel('Velocity Dispersion (m/s)')
    subplot(2,3,4)
    plot(ki:kf,dgamma*r2d,'x')
    xlabel('index')
    ylabel('Gamma Dispersion (deg)')
    subplot(2,3,5)
    plot(ki:kf,dpsi*r2d,'x')
    xlabel('index')
    ylabel('Heading Dispersion(deg)')
    subplot(2,3,6)
    plot(ki:kf,CR_km,'x')
    xlabel('index')
    ylabel('Initial Crossrange(km)')
end
```

Simulate Entry

This is the file that simulates one dispersed trajectory of the vehicle.

```
function [down_miss,final_long,final_lat]=simulate_entry(state,...
    mass_nom,mass_pert,Sref,vehicle,plots,scenario)

global bias_atm Mag_atm1 omega_atm1 theta_atm Mag_atm2 omega_atm2
global CL_bias CD_bias
global lon_site lat_site
global lon_site_bias lat_site_bias
% Inputs
% state, a 7 element vector consisting of the following elements
% r     =  radius, dimensionless
% theta =  longitude, radians
% phi   =  latitude, radians
% V     =  Relative Velocity, dimensionless
% gamma =  Flight Path angle, radians
% psi   =  Heading angle, radians, zero north, positive clockwise
% s     =  downrange flown
% mass_nom= Spacecraft nominal mass, kilograms
% mass_pert=Spacecraft perturbed mass, kilograms
% Sref   =  Spacecraft Reference area, square meters
% vehicle = number corresponding to vehicle type 1-Apollo, 2-CEV, 3-Shuttle
% plots = 1 or 0 flag to display plots for this simulation

% Outputs
% arc = miss distance, nautical miles

%Earth parameters
R0=6378136.3;                       %m
g0=9.80665;                        %m/s^2
Vscale=sqrt(R0*g0);                   %7908.4m/s
```

```
tscale=sqrt(R0/g0);                 %806.329s
%omega=7.2921151e-5*tscale;    %converts rad/s to dimensionless

%Conversion factors
m2ft=3.28083989501;                %meters to feet
%ft2m=1/m2ft;                      %feet to meters
d2r=0.01745329251994;         %degrees to radians = pi/180;
r2d=57.29577951308232;        %radians to degrees= 180/pi;

currentbank=0;          %initial vehicle bank

PHIBKRATEMAXDEG=20;     %vehicle maximum roll rate (deg/s)
PHIBKACCELMAXDEG=10;    %vehicle maximum acceleration rate (deg/s^2)

altF=7620;  %meters, 25000 feet
VF=150; %m/s

%Integration setup
tend=2500;                  %ending time for integration (seconds)
y=zeros(tend,7);        %ending time for integration (seconds)
y(1,:)=state;       %set first row of state matrix with ini. cond.
Vfinal=500/Vscale/m2ft; %final velocity non-dimensionlized (500 fps,152m/s)

hinput=1/tscale;        %step size  hinput=.0012399
p=1;                    %initialize counting variable

%For Matlab, initialize all the values to be recorded as zero
t=zeros(1,tend);            %time
alt=zeros(1,tend);          %altitude
load=zeros(1,tend);         %g-load
secantbank=zeros(1,tend);   %commanded bank
sigma=zeros(1,tend);        %actual flown bank
iterations=zeros(1,tend);   %number of iterations to converge
selector=zeros(1,tend);     %phase of guidance
D=zeros(1,tend);            %actual drag
L=zeros(1,tend);            %actual lift
D_nom=zeros(1,tend);        %nominally expected drag
L_nom=zeros(1,tend);        %nominally expected lift
d_density=zeros(1,tend);    %ratio of true density to dispersed
rho_true=zeros(1,tend);     %true density
rho_nom=zeros(1,tend);      %nominal density based on US 76 model
Dpsi=zeros(1,tend);         %crossrange envelope
CR=zeros(1,tend);           %crossrange with respect to true site
CR_bias=zeros(1,tend);      %crossrange with respect to biased site
K_drag_est=zeros(1,tend);   %drag multiplier
K_lift_est=zeros(1,tend);   %lift multiplier

K_drag_est(1)=1; %initial drag multiplier is one
K_lift_est(1)=1; %initial lift multiplier is one
selector(1)=1; %phase of guidance is initially one
NCall=[]; %initalize roll reversal flag

%range-to-go part along great circle distance
[arc, azimuth] = sphereping(y(p,2),y(p,3),lon_site*d2r,lat_site*d2r);
rangetogolanding_1st=arc*R0; %result in meters
while 1
```

```
    %True load is calculated in this section since it is assumed that
    %accelerometers will be able to measure the actual g load.
    alt(p)=(y(p,1)-1)*R0;       %calculate altitude (meters)
    [rho_nom(p),rho_a] = density(alt(p)/1000); %calculate density (kg/m^3)
    [Vs]= Vsound(alt(p)/1000);  %calculate speed of sound (m/s)
    Vs=Vs/Vscale;           %non-dimensionalize speed of sound
    Mach=y(p,4)/Vs;         %calculate mach number

    %Find trim angle of attack (rad.) and lift and drag coefficients for
    %particular vehicle
    if vehicle==1
        alpha = trim_alpha_CEV(Mach);
        [CL,CD]=CLCD_CEV(Mach, alpha);
    elseif vehicle==2
        alpha = trim_alpha_Apollo(Mach);
        [CL,CD]=CLCD_Apollo(Mach, alpha);
    else
        error('vehicle unselected')
    end

    CL  = CL+CL_bias;       %Perturb lift coefficient
    CD  = CD+CD_bias;       %Perturb drag coefficient


    %Use analytic density dispersion with alt (in meters to kilometers)
    d_density(p)=(1+bias_atm+(Mag_atm1+Mag_atm2*sin(omega_atm2*alt(p)/...
1000))*sin(theta_atm + omega_atm1*alt(p)/1000));
    rho_true(p) = d_density(p)*rho_nom(p);

    %Find true drag and lift (g's)
    D(p)=rho_true(p)*(Vscale*y(p,4))^2*Sref*CD*.5/(mass_pert*g0);
    L(p)=rho_true(p)*(Vscale*y(p,4))^2*Sref*CL*.5/(mass_pert*g0);

    %Find nominal drag and lift (g's)
    D_nom(p)=rho_nom(p)*(Vscale*y(p,4))^2*Sref*(CD-CD_bias)*.5/...
(mass_nom*g0);
    L_nom(p)=rho_nom(p)*(Vscale*y(p,4))^2*Sref*(CL-CL_bias)*.5/...
(mass_nom*g0);

    load(p)=sqrt(L(p)^2+D(p)^2);
    sigma(p)=currentbank;

    %Compute arc length and azimuth to true landing site
    [arc, azimuth] = sphereping(y(p,2),y(p,3),lon_site*pi/180,...
lat_site*pi/180);
    %Find crossrange to true landing site (rad)
    CR(p) = asin(sin(arc)*sin(y(p,6) - azimuth));
    Y = 0.25/24/2*y(p,4)+0.3*1852/R0;
    if abs(currentbank)<15
        Y   = Y/2;
    end
    Dpsi(p) = Y; %Record crossrange envelope

    %Compute arc length, azimuth, and crossrange to biased landing site
    [arc, azimuth] = sphereping(y(p,2),y(p,3),lon_site_bias*d2r,...
```

```
lat_site_bias*d2r);
    CR_bias(p) = asin(sin(arc)*sin(y(p,6) - azimuth));

    %Run guidance for particular scenario and phase
    if scenario==1
        %Full Apollo Guidance
        if mod(p+1,2)==0 %only runs every 2 seconds
            [currentbank,secantbank(p),selector(p)]=Apollo_guidance(p,...
y(p,:),load(p),currentbank);
        else
            selector(p)=selector(p-1);
            secantbank(p)=secantbank(p-1);
        end
    elseif  scenario==2
        %NPC Skip guidance
        if selector(p)~=5
            %Call lift and drag filters
            K_lift_est(p)=filter_LD_ratios(y(p,1),y(p,4),L(p),mass_nom,...
Sref,1,vehicle);
            K_drag_est(p)=filter_LD_ratios(y(p,1),y(p,4),D(p),mass_nom,...
Sref,0,vehicle);

            [currentbank,selector(p),iterations(p),biased_counts,...
secantbank(p)]=Secant_guidance_disp(p,[y(p,1),y(p,2),...
y(p,3),y(p,4),y(p,5),y(p,6)],load(p),PHIBKRATEMAXDEG,...
PHIBKACCELMAXDEG,mass_nom,Sref,vehicle,K_drag_est(p),...
K_lift_est(p));
        end
        %Ping Lu NPC Final Phase guidance.
        if selector (p)==5
            if isempty(NCall)
                NCall=1;
            else
                NCall=2;
            end
            %Everything sent to this function is nominal and with units
            [currentbank,t_togo,iterations(p),secantbank(p),...
K_lift_est(p),K_drag_est(p)]=EntryGuidance_disp(lon_site,...
lat_site,alt(p),y(p,2)*r2d,y(p,3)*r2d,y(p,4)*Vscale,...
y(p,5)*r2d,y(p,6)*r2d,altF,VF,currentbank,L(p),D(p),...
PHIBKRATEMAXDEG,PHIBKACCELMAXDEG,mass_nom,Sref,1,NCall,...
vehicle);
            selector(p+1)=5;
        end
    elseif scenario==3
        %Apollo Skip Guidance only
        if selector(p)~=5
            if mod(p+1,2)==0
                [currentbank,secantbank(p),selector(p)]=...
Apollo_guidance(p,y(p,:),load(p),currentbank);
            else
                selector(p)=selector(p-1);
                secantbank(p)=secantbank(p-1);
            end
        end
        %Ping Lu NPC Final Phase guidance.
```

```
        if selector (p)==5
            if isempty(NCall)
                NCall=1;
            else
                NCall=2;
            end
            %Everything sent to this function is nominal and with units
            [currentbank,t_togo,iterations(p),secantbank(p),...
K_lift_est(p),K_drag_est(p)]=EntryGuidance_disp(lon_site,...
lat_site,alt(p),y(p,2)*r2d,y(p,3)*r2d,y(p,4)*Vscale,...
y(p,5)*r2d,y(p,6)*r2d,altF,VF,currentbank,L(p),D(p),...
PHIBKRATEMAXDEG,PHIBKACCELMAXDEG,mass_nom,Sref,1,NCall,...
vehicle);
            selector(p+1)=5;
        end
    elseif scenario==4
        %NPC Skip guidance
        if selector(p)~=5
            K_lift_est(p)=filter_LD_ratios(y(p,1),y(p,4),L(p),mass_nom,...
Sref,1,vehicle);
            K_drag_est(p)=filter_LD_ratios(y(p,1),y(p,4),D(p),mass_nom,...
Sref,0,vehicle);

            [currentbank,selector(p),iterations(p),biased_counts,...
secantbank(p)]=Secant_guidance_disp(p,[y(p,1),y(p,2),...
y(p,3),y(p,4),y(p,5),y(p,6)],load(p),PHIBKRATEMAXDEG,...
PHIBKACCELMAXDEG,mass_nom,Sref,vehicle,K_drag_est(p),...
K_lift_est(p));
        end
        %Apollo Final phase Guidance
        if selector(p)==5
            if mod(p+1,2)==0
                [currentbank,secantbank(p+1),selector(p+1)]=...
                    Apollo_guidance_final(p,y(p,:),load(p),currentbank);
            else
                selector(p+1)=selector(p);
                secantbank(p+1)=secantbank(p);
            end
        end
    else
        error('scenario not selected')
    end

    %End integration of trajectory if velocity below Vfinal
    if  y(p,4)<Vfinal
        if (rangetogolanding_1st-y(p,7)*R0)/1000>5
            'Undershoot (>5km)'
        elseif (arc*R0)/1000<-5
            'Overshoot (>5km)'
        end
        break
    %End integration if time runs out
    elseif round(t(p)*tscale)==tend
        'time ran out'
        break
    %End integration if we're overshot by 5000km or more.
```

```
    elseif (rangetogolanding_1st-y(p,7)*R0)/1000<-5000
        'Overshoot (>5000km)'
        break
    end

    p=p+1;  %increment counting variable

    %Call next integration time step
    y(p,:)=myrungekutta_entry(@eqnofmotion_disp,t(p-1),y(p-1,:),...
        hinput,mass_pert,Sref,currentbank,vehicle);
    %Increase time.
    t(p)=t(p-1)+hinput;
end

[arc, azimuth] = sphereping(y(p,2),y(p,3),lon_site*d2r,lat_site*d2r);
cross_miss=asin(sin(arc)*sin(y(p,6)-azimuth))*R0/1000;
down_miss=arc*R0/1000; % miss distance in kilometers

final_long=y(p,2)*r2d;
final_lat=y(p,3)*r2d;
if plots
    %create a time vector that fits with the one used and is rescaled
    p=p-1;
    t=t(1:p)*tscale;
    rangetogolanding=rangetogolanding_1st-s*R0;
    r=y(1:p,1);
    theta=y(1:p,2);
    phi=y(1:p,3);
    V=y(1:p,4);
    gamma=y(1:p,5);
    psi=y(1:p,6);
    s=y(1:p,7);
    Dpsi=Dpsi(1:p);
    CR=CR(1:p);
    Dpsi(2,:)=-Dpsi;
    CR_bias=CR_bias(1:p);

    sigma=sigma(1:p);
    selector=selector(1:p);
    iterations=iterations(1:p);
    load=load(1:p);
    secantbank=secantbank(1:p);
    K_drag_est=K_drag_est(1:p);
    K_lift_est=K_lift_est(1:p);

    D=D(1:p);
    L=L(1:p);
    D_nom=D_nom(1:p);
    L_nom=L_nom(1:p);
    d_density=d_density(1:p);
    rho_true=rho_true(1:p);
    rho_nom=rho_nom(1:p);
    Dist_to_target_kilometers=down_miss

    figure
    subplot(2,2,1)
```

```
plot(t,((r*R0)-R0)/1000)
xlabel('Time, s')
ylabel('Altitude, km')
subplot(2,2,2)
plot(t,V*Vscale/1000)
xlabel('Time, s')
ylabel('Velocity, km/s')
subplot(2,2,3)
plot(V*Vscale/1000,((r*R0)-R0)/1000)
xlabel('Velocity, km/s')
ylabel('Altitude, km')
subplot(2,2,4)
plot(rangetogolanding/1000,(r*R0-R0)/1000,0,0,'r*')
set(gca,'XDir','reverse')
xlabel('Downrange to-go, km')
ylabel('Altitude, km')

figure
subplot(2,2,1)
earthradius = almanac('earth','radius','km');
[latc,longc] = scircle1(lat_site,lon_site,2.5,[],earthradius);
plot(theta*r2d,phi*r2d,longc+360,latc,'r',final_long,final_lat,'x',...
    lon_site,lat_site,'r*',lon_site_bias,lat_site_bias,'g*')
xlabel('Longitude, degrees')
ylabel('Latitude, degrees')
axis equal
grid on
subplot(2,2,2)
plot(t,gamma*r2d)
xlabel('Time, s')
ylabel('Flight Path Angle, degrees')
subplot(2,2,3)
plot(t,load)
xlabel('Time, s')
ylabel('Total Load, g''s')
subplot(2,2,4)
plot(t,psi*r2d)
xlabel('Time, s')
ylabel('Heading, degrees')

figure
plot(t,sigma,'',t,abs(secantbank),'r--')
xlabel('Time, s')
ylabel('Bank angle, degrees')
legend('Flown Bank','Commanded Bank')

figure
subplot(2,1,1)
plot(t,selector)
text(0,1,'iniroll','FontSize',8)
text(0,2,'down','FontSize',8)
text(0,3,'upcontrol','FontSize',8)
text(0,4,'kepler','FontSize',8)
text(0,5,'final','FontSize',8)

subplot(2,1,2)
```

```
    plot(t,iterations)
    xlabel('Time,s')
    ylabel('Iterations')
    Plottime

    figure
    subplot(2,2,1)
    plot(t,D,t,D_nom)
    xlabel('Time, s','Fontsize',12)
    ylabel('Drag g''s','Fontsize',12)
    legend('True Drag','Nominal Drag')
    set(gca,'Fontsize',12)
    subplot(2,2,2)
    plot(t,K_drag_est)
    xlabel('Time, s','Fontsize',12)
    ylabel('Drag Factor','Fontsize',12)
    set(gca,'Fontsize',12)
    subplot(2,2,3)
    plot(t,L,t,L_nom)
    xlabel('Time, s','Fontsize',12)
    ylabel('Gain value or g''s','Fontsize',12)
    legend('True Lift','Nominal Lift')
    set(gca,'Fontsize',12)
    subplot(2,2,4)
    plot(t,K_lift_est)
    xlabel('Time, s','Fontsize',12)
    ylabel('Lift Factor','Fontsize',12)
    set(gca,'Fontsize',12)
    Plottime4

    decx=[0.5 7 30 100 1000];
    decy=[11.5 10 8 4 4];
    nomx=[0.5 5 300 1000];
    nomy=[19 16 7.5 5];
    autox=[0.5 300 1000];
    autoy=[40 7.5 5];
    x = 0:.01:10.1;
    y=load;
    n_elements = histc(y,x);
    c_elements = cumsum(fliplr(n_elements));
    figure
    loglog(autox,autoy,'r','LineWidth',2)
    hold on
    loglog(nomx,nomy,'b--','LineWidth',2)
    loglog(decx,decy,'g--','LineWidth',2)
    loglog(fliplr(c_elements),x)
    legend('Max for automated crew abort/escape',...
'Nominal Entry (conditioned crew)',...
'Deconditioned,ill, or injured crew')
    axis([0.1 1000 1 100])
    ylabel('g-load')
    xlabel('time')
    grid on
    hold off

    figure
```

```
    subplot(2,2,1)
    plot(t,d_density)
    xlabel('Time, s')
    ylabel('\rho_{true}/\rho_{nom}')
    subplot(2,2,2)
    plot(d_density,(r*R0-R0)/1000)
    xlabel('\rho_{true}/\rho_{nom}')
    ylabel('altitude (km)')
    subplot(2,2,3)
    semilogy(t,rho_true,t,rho_nom)
    xlabel('Time, s')
    ylabel('Density (kg/m^3)')
    legend('True','Nominal')
    subplot(2,2,4)
    semilogx(rho_true,(r*R0-R0)/1000,rho_nom,(r*R0-R0)/1000)
    xlabel('Density (kg/m^3)')
    ylabel('altitude (km)')
    legend('True','Nominal')

    figure
    plot(t,CR,'b-',t,CR_bias,'r--',t,Dpsi,'k:',t,-Dpsi,'k:')
    legend('CR','CR-bias')
    xlabel('Time, s ','FontSize',12)
    ylabel('Crossrange, radians','FontSize',12)
    set(gca,'Fontsize',12)
    text(10,0.005,'right of target','FontSize',8)
    text(10,-0.005,'left of target','FontSize',8)

end
```

Runge-Kutta Entry

```
function x2=myrungekutta_entry(f,t,x,h,mass,Sref,bank,vehicle)

%Implements a 4th order Runge Kutta integration method
%f is the function in the equation x'=f(t,x)
%t is the current time, scalar
%x are the initial conditions which should be a row vector
%h is the step size, scalar
%x2 is the new conditions, as a row vector
%Christopher Brunner

p1=(h*f(t,x,mass,Sref,bank,vehicle))';
p2=(h*f(t+.5*h,x+.5*p1,mass,Sref,bank,vehicle))';
p3=(h*f(t+.5*h,x+.5*p2,mass,Sref,bank,vehicle))';
p4=(h*f(t+h,x+p3,mass,Sref,bank,vehicle))';

x2=x+ (p1 + 2*p2 + 2*p3 + p4)/6;
```

Dispersed Equations of Motion

```
function dydt=eqnofmotion_disp(t,y,mass,Sref,bank,vehicle)
```

```
% Christopher Brunner
% Evaluate the 3-DOF equations of motion over a rotating spherical Earth;
% All state variables and time are dimensionless
% Use 1976 US Standard Atmosphere
% Use time as independent variable
% Inputs are:
% t, dimensionless time
% y, a vector consisting of 7 elements,
% r, the dimensionless position
% theta, longitude, in radians
% phi, latitude in radians
% V, dimensionless velocity
% gamma, flight path angle, negative down, in radians
% psi, heading angle, in radians
% s, range flown, scaled by Earth's radius, dimensionless
% mass, Vehicle's true mass in kilograms
% Sref, Vehicle area in square meters
% bank, current bank angle in degrees
% vehicle, a number corresponding to the vehicle being flown
% 1- Crew Exploration Vehicle
% 2- Apollo Command Capsule
%
% Output is a 7 element column vector consisting of the time derivatives
% of the input vector.
%
% Global predefined dispersion variables should be the atmospheric bias and
% magnitude, along with constant dispersions in lift and drag coefficient.
%
% April 2007
%

global bias_atm Mag_atm1 omega_atm1 theta_atm Mag_atm2 omega_atm2
global CL_bias CD_bias

R0=6378136.3;                       %m
g0=9.80665;                        %m/s^2
Vscale=sqrt(R0*g0);                   %7908.4m/s
tscale=sqrt(R0/g0);                   %806.329s
omega=7.2921151e-5*tscale;    %converts rad/s to dimensionless
d2r=0.01745329251994;        %degrees to radians = pi/180;

r=y(1);
%theta=y(2);    %commented since not needed below
phi=y(3);
V=y(4);
gamma=y(5);
psi=y(6);
%s=y(7);        %commented since not needed below

%convert s/c radius from dimensionless to altitude in kilometers
alt=(r-1)*R0/1000;

%Compute density from US 76 (rho in kg/m^3)
[rho,rho_a] = density(alt);

%Takes an input in kilometers of altitude and outputs in meters/second
```

```
[Vs]= Vsound(alt);

%convert from m/s to dimensionless
Vs=Vs/Vscale;

Mach=V/Vs;

if vehicle==1
    %takes an input of mach number and outputs angle of attack, alpha in
    %radians
    alpha = trim_alpha_CEV(Mach);
    % Compute CL and CD for the Apollo-like vehicle as functions of
    % Mach number and angle of attack in radians
    [Cl,Cd]=CLCD_CEV(Mach, alpha);
elseif vehicle==2
    %takes an input of mach number and outputs angle of attack, alpha in
    %radians
    alpha = trim_alpha_Apollo(Mach);
    % Compute CL and CD for the Apollo-like vehicle as functions of
    % Mach number and angle of attack in radians
    [Cl,Cd]=CLCD_Apollo(Mach, alpha);
end

Cl  = Cl+CL_bias;
Cd  = Cd+CD_bias;
rho = (1+bias_atm+(Mag_atm1+Mag_atm2*sin(omega_atm2*alt))*...
sin(theta_atm + omega_atm1*alt))*rho;

%Lift and drag
D=rho*(Vscale*V)^2*Sref*Cd*.5/(mass*g0);
L=rho*(Vscale*V)^2*Sref*Cl*.5/(mass*g0);

sigma=bank*d2r;
rdot=V*sin(gamma);
thetadot=V*cos(gamma)*sin(psi)/(r*cos(phi));
phidot=V*cos(gamma)*cos(psi)/r;
Vdot=-D-sin(gamma)/r^2+omega^2*r*cos(phi)*(sin(gamma)*cos(phi)-...
    cos(gamma)*sin(phi)*cos(psi));
gammadot=1/V*(L*cos(sigma) + (V^2-1/r)*cos(gamma)/r + 2*omega*V*...
    cos(phi)*sin(psi) + omega^2*r*cos(phi)*(cos(gamma)*cos(phi)+...
    sin(gamma)*cos(psi)*sin(phi)));
psidot=1/V*(L*sin(sigma)/cos(gamma) + V^2*cos(gamma)*sin(psi)*tan(phi)...
    /r - 2*omega*V*(tan(gamma)*cos(psi)*cos(phi)-sin(phi)) + ...
    omega^2*r*sin(psi)*sin(phi)*cos(phi)/cos(gamma));
sdot=V*cos(gamma)/r;
dydt=[rdot thetadot phidot Vdot gammadot psidot sdot]';
```

Density function

This function computes the atmospheric density based on the 1976 US standard atmosphere.

```
function [rho,rho_a] = density(alt)
```

```
%   Calculate the atmospehric density and gradient of density
%   with respect to altitude by 1976 US standard atmosphere.
%   alt in km (0<alt <130)
%   rho=density in kg/m^3
%   rho_a= d(rho)/d(alt)  unit (kg/m^3/km)

rho_0 = 1.225;

a= -0.009334500409893852;
b= -0.07713480853494097;
c= -0.00382948508292597;
d= 5.242803244759748E-05;
e= 6.454862591920205E-07;
f= -2.031133609734722E-08;
g= 1.568378909033718E-10;
h= -3.928350728483702E-13;

x= alt;
if alt >130.0
    x = 130.;
end

y    = a+x*(b+x*(c+x*(d+x*(e+x*(f+x*(g+x*h))))));
rho  = rho_0*exp(y);
dy   = (b+x*(2.*c+x*(3.*d+x*(4.*e+x*(5.*f+x*(6.*g+7.*x*h))))));
rho_a= rho*dy;
return
```

Sound Velocity

This code calculates the speed of sound at a particular altitude.

```
function [V_sound]= Vsound(altitude)

%
%  Calculate the speed of sound as a function of altitude
%     Curve-fit to the 1976 US Standard Atmosphere
%
%  Input:
%  altitude -altitude in km (0=< alt =< 85 km)
%  (For 85 km and above, the speed of sound is
%  not well defined. This function truncates
%  any input altitude greater than 85 km at 85 km
%  so the speed of sound returned is constant
%  for alt > 85 km)
%
%  Output:
%  Vsound  -speed of sound (m/sec)
%
%   by Ping Lu
%  Iowa State University
%  May 20, 2004
%
```

```
alt = altitude;

if(alt > 85.0)
    alt = 85.0;
end

if(alt < 0.0)
        alt = 0.0;
end
a   =  340.29112;
b   =   -0.87245462;
c   = -300.10255;
d   =    0.3077688;
e   =  106.65518;
f   =   -0.052091446;
g   =  -18.273173;
c_h =    0.004143208;
c_i =    1.485658;
c_j =   -0.00012124848;
c_k =   -0.045201417;
h_s = sqrt(alt);
dum1 = a + c*h_s + e*alt + g*h_s*h_s*h_s + c_i*alt*alt + c_k*power(h_s,5.0);
dum2 = 1.0+b*h_s + d*alt + f*h_s*h_s*h_s + c_h*alt*alt + c_j*power(h_s,5.0);
V_sound=dum1/dum2;
```

CLCD Apollo

This function computes the lift and drag coefficients for the Apollo Command Module.

```
function [CL,CD]=CLCD_Apollo(Mach,alpha)

% Calculate CL and CD of Apollo Command Module
% near alpha= 155 deg (trim)
% Input
% Mach  - Mach number (0.7<=Mach)
% alpha - angle of attack in rad (near 155 deg)

mach= Mach;
y   = alpha;

if  Mach >10
    mach = 10;
elseif Mach <0.7
    mach=0.7;
end

if y> 160/57.296
    y = 160/57.296;
elseif y< 154.5/57.296
    y= 155.0/57.296;
end

x   =1.0/mach;
```

```
z1  =1.041491644520038+x*(-0.4111523830941650+x*(1.514954425110922...
    +x*(-1.487270049440492+x*(0.0676930935963811+x*0.2381479131194706))));
z2  =y*(0.4520310716015592+y*(-0.2398166259168704));
CL  =z1+z2;

z1  =3.715227798263816+x*(-1.931052031762790+x*(7.930451805087150...
    +x*(-11.24925503607193+x*(6.056043034756084+x*(-1.065359571892614)))));
z2  =y*(-2.650664383420442+y*0.6539634146341463);
CD  =z1+z2;
```

CLCD CEV

This function computes the lift and drag coefficients for the Crew Exploration Vehicle.

```
function [CL,CD]=CLCD_CEV(Mach,alpha)

% Calculate CL and CD of the Crew Exploration Vehicle
% CEV 602A (April 2006)
% Most accurate when alpha is near the trim (Mach dependent)
% Input
% Mach  - Mach number (0.5<=Mach<=33.7)
% alpha - angle of attack in rad (in the range of 150-170 deg )
% April 17, 2006
% By P. Lu

d2r=0.01745329251994;        %degrees to radians = pi/180;


mach= Mach;
y   = alpha;

if  Mach >33.7
    mach = 33.7;
elseif Mach <0.5
    mach=0.5;
end

if y> 170*d2r
    y = 170*d2r;
elseif y< 150*d2r
    y= 150.0*d2r;
end

x   = 1.0/mach;
z1  = 2.523457051642765+x*(-0.4349807599157984+...
    x*(2.387583341651182+x*(-3.754388446751747+...
    x*(2.197426333443260+x*-0.4333221758967753))));
z2  = y*-0.7594052924642753;
CL  = z1+z2;

z1 =-7.972686129072812+x*(-1.823592134486534+...
    x*(8.052361019744082+x*(-12.06958431525050+...
    x*(6.977094607191365+x*-1.382017708945732))));
z2 =y*(5.901190368747020+y*-0.9122809306975412);
```

```
CD = z1+z2;

return
```

trim alpha Apollo

This code calculates the trim angle of attack for the Apollo Command Module.

```
function alpha=trim_alpha_Apollo(Mach)

% Compute the trim alpha (rad) for the Apollo Command Module
% 0.4<Mach < 10

x = Mach;
if x>10
    x=10;
elseif x<0.4
    x=0.4;
end
x=x*x;
alpha=(2.874898382776241+x*(-4.214191310719779+x*(3.803336249031923...
    +x*(-0.5584426652219696+x*0.02987059265081175))))/...
    (1.0+x*(-1.499344405009540+x*(1.397512630553629+...
    x*(-0.2055419628490292+x*0.01099359208645952))));

return
```

trim alpha CEV

This code calculates the trim angle of attack for the Crew Exploration Vehicle.

```
function alpha=trim_alpha_CEV(Mach)

% Compute the trim alpha (rad) for the Crew Exploration Vehicle
% CEV 602A (April 2006)
% 0.5<Mach < 33.7
% alpha in rad
% April 17, 2006, by P. Lu

x = Mach;
if x>33.7
    x=33.7;
elseif x<0.45
    x=0.45;
end


 alpha=(168.0236229048111+x*(-250.0300295195276+x*(140.5100679500302...
     +x*(-10.27112015846153+x*0.6120736978892660))))/...
    (1.0+x*(-1.496351219250425+x*(0.8595875622553195+...
    x*(-0.06035589947504183+x*(0.003613162469840939+x*...
```

```
3.254315960973067E-06)))));
 alpha=alpha/57.2958;

return
```

APPENDIX B. SKIP GUIDANCE CODES

Secant Guidance

The following code is the main guidance algorithm that runs the secant iteration. It requires several additional functions to run as given below.

```
function [varargout]=Secant_guidance_disp(p,state,load,PHIBKRATEMAXDEG,...
    PHIBKACCELMAXDEG,mass,Sref,vehicle,K_drag,K_lift)

% Christopher Brunner
% Code started October 17,2006
% Version as of April 8, 2008
% Guidance function to compute skip bank angle and output rate-limited
% value.
% INPUTS
% p, current time in seconds
% state, a 6 element vector consisting of:
% r, dimensionless spacecraft radius
% theta, longitude radians
% phi, latitude radians
% Vin, dimensionless velocity
% gamma, flight path angle, radians
% psi, heading angle, radians
% load, spacecraft acceleration, g's
% PHIBKRATEMAXDEG, max bank rate in deg/s
% PHIBKACCELMAXDEG, max bank acceleration in deg/s^2
% mass, vehicle mass in kg
% Sref, vehicle area in m^2
% vehicle, corresponding to the vehicle being used 1=CEV, 2=Apollo
% K_drag, scaling factor on drag, dimensionless
% K_lift, scaling factor on lift, dimensionless
%
% OUTPUTS
% currentbank = output actual bank angle of vehicle in degrees
% SELECTOR = phase number
% iterations = number of iterations to compute skip angle
% biased_counts =  number of iterations for biasing
% SECANTBANK = converged skip bank angle in degrees
%
% Functions Required:
%   skipranger.m
%   sphereping.m
%   crossbias.m
```

```
%   Bank_sign_Apollo.m
%   shortest_bank_limiter.m
%=========================================================================
global lon_site_bias lat_site_bias
global lon_site lat_site

persistent SELECTOR a SECANTBANK Ncall bias_run biased_counts

secant_error=25000;              % in m (25km=13.5nm)

R0=6378136.3;                   % Earth Radius, m
% g0=9.80665;                   % Earth gravity, m/s^2
% Vscale=sqrt(R0*g0);           % Velocity scaling factor
d2r=0.01745329251994;         % degrees to radians = pi/180;
r2d=57.29577951308232;        % radians to degrees= 180/pi;

% Phases for SELECTOR
% Initial roll=1
% Down control=2
% Up control=3
% Kepler=4
% Final=5

kepbank=70; % in degrees, expected bank angle to fly in kepler phase
finalbank=70; % in degrees, expected bank angle to fly in final phase
g_thres=0.05; % load threshold to be inside atmosphere
range_thres=2000000; % range threshold to start final phase (2000 km), meters

%INITIALIZATION
if isempty(SELECTOR)
    Ncall=1;
    SELECTOR=1;
    bias_run=1;
    biased_counts=0;
    SECANTBANK=0;

    % final initial skip bank angle that causes trajectory to undershoot
    % landing site
    [predrangetogo_m,ground,crosscheck]=skipranger(0,state,SECANTBANK,...
    kepbank,finalbank,1,mass,Sref,vehicle,K_drag,K_lift);
    while predrangetogo_m<0 && SECANTBANK<180
        SECANTBANK=SECANTBANK+2.5;
        [predrangetogo_m,ground,crosscheck]=skipranger(0,state,SECANTBANK,...
        kepbank,finalbank,1,mass,Sref,vehicle,K_drag,K_lift);
    end
    % Warnings for too small a bank and too large a bank
    if SECANTBANK==0 && predrangetogo_m>0
        warning('vehicle has inadequate energy to reach landing site')
    elseif SECANTBANK==180
        warning('vehicle has too much energy')
    end
    a=cos(SECANTBANK*d2r);
end
iterations=1;
ground_error=0;
% Begin computation of bank angle for skip phase
```

```
if SELECTOR<=3 % Only run computation if we're before kepler phase
    % Next line makes initial computation of range based on previously
    % computed solution
    [fa,grounda,crosscheck]=skipranger(p,state,acos(a)*r2d,kepbank,...
    finalbank,SELECTOR,mass,Sref,vehicle,K_drag,K_lift);
    if abs(fa)>secant_error     % secant only done if range error is large
        % Perturb bank angle parameter and get new range
        b=a-1e-3;
        [fb,groundb,crosscheck]=skipranger(p,state,acos(b)*r2d,kepbank,...
        finalbank,SELECTOR,mass,Sref,vehicle,K_drag,K_lift);
        while 1
%error check to insure we don't divide by zero if ranges were equal
            if fb==fa
                % arning('divide by zero, prev bank angle used')
                a=cos(SECANTBANK*d2r);
                iterations=-3;
                break
            end

            % Guts of secant method, make correction in bank angle
            s=(b-a)/(fb-fa);
            b=a;
            fb=fa;
            a=a-fa*s;

            % If secant computes a cosine of bank angle that is greater than
            % -1, sets equal to -1, but this occurs only if SECANTBANK
            % (previous angle) agrees with the assessment, otherwise a
            % numerical problem with the perturbation size is assumed and we
            % go into ground correction with a different perturbation step
            % size
            if a>1
                if SECANTBANK<15
                    % warning('a>1')
                    a=.999;
                    iterations=-1;
                    break
                else
                    % warning('a>1, but SECANTBANK is larger than 15')
                    a=1;% Setting a=1, forces the next call to
                    % rangepredictor to be off the ground and starts
                    % the ground fix loop
                    ground_error=1;
                end
            elseif a<-1
                if SECANTBANK>165
                    % warning ('a<-1')
                    a=-.999;
                    iterations=-2;
                    break
                else
                    % warning('a<-1, but SECANTBANK is smaller than 165')
                    a=1; % Setting a=1, forces the next call to
                    % rangepredictor to be off the ground and starts
                    % the ground fix loop
                    ground_error=1;
```

```
                end
            end
            % Compute range with new bank angle.
            [fa,grounda,crosscheck]=skipranger(p,state,acos(a)*r2d,...
kepbank,finalbank,SELECTOR,mass,Sref,vehicle,K_drag,K_lift);

            % Next statements are if the trajectory leaves the ground.
            if grounda==0 || ground_error==1
                ground_error=0;
                warning('trajectory left ground...fixing')
                bankfix=acos(a)*r2d;
                fa=-1;
                while fa < 0
                    bankfix=bankfix+2.5;
                    [fa,grounda,crosscheck]=skipranger(p,state,bankfix,...
                    kepbank,finalbank,SELECTOR,mass,Sref,vehicle,K_drag,...
K_lift);
                    if bankfix>180 && grounda==0
                        bankfix=180
                        warning('Bank angle=180, trajectory skips out')
                        break
                    elseif bankfix>180 && grounda==1
                        bankfix=180
                        warning('Bank angle=180')
                        break
                    end
                end
                if abs(bankfix-180) <1e-4
a=-0.999
break
end
a=cos(bankfix*d2r);
                b=a-1e-6;
                [fb,groundb,crosscheck]=skipranger(p,state,acos(b)*r2d,...
kepbank,finalbank,SELECTOR,mass,Sref,vehicle,K_drag,...
K_lift);
            end
            % If range is small than permitted error, converged!
            if abs(fa)<secant_error  % 46.3km=25nm
                break
            elseif iterations==10
                %warning('Max iterations reached, prev. bank used,')
                a=cos(SECANTBANK*d2r);
                break
            end
            iterations=iterations+1;
        end
    end
end
%Compute skip phase bank angle.
SECANTBANK=acos(a)*r2d;
%Compute range for purposes of
[arc, azimuth] = sphereping(state(2),state(3),lon_site_bias*d2r,...
lat_site_bias*d2r);
rangetogolanding=arc*R0; %result in meters
```

```
if bias_run==1 && biased_counts==0 && abs(crosscheck)>1 && iterations<10...
&& SELECTOR>=2
    % see if 1)Have we done biasing (bias_run==1)?
    %       2)Does it need doing? (abs(crosscheck)>1)
    %       3)Did we have a converged solution (iterations<10)
    %       4)Are we far enough into trajectory to do it? (SELECTOR>=2)
    bias_run=0;
    biased_counts=1; % initialize the number of biases
    reversed=crosscheck; % store the first direction to bias (+2 or -2)
    crossbias(state,reversed/2,biased_counts); % bias the first time
    while 1
        [predrangetogo_m,ground,crosscheck]=skipranger(p,...
            state,acos(a)*r2d,kepbank,finalbank,SELECTOR,mass,Sref,...
            vehicle,K_drag,K_lift); % run biasing with new biased site.
        if reversed==2 && crosscheck<0
            break % quit if originally we were too far right, now we're left
        elseif reversed==-2 && crosscheck>0
            break % quit if originally we were too far left, now we're right
        elseif biased_counts>12
            warning('crosscheck exceeds limits')
            break % quit if we've done biasing 12 times.
        else
            biased_counts=biased_counts+1; % increase count
            crossbias(state,crosscheck,biased_counts); % run biasing
        end
    end
elseif bias_run==1 && abs(crosscheck)<=1 && iterations<10 && SELECTOR>=2
    bias_run=0; % never do biasing as long as other criteria met.
end


if SELECTOR==1 && load<g_thres % Initial roll
    ROLLC=0;
    if rangetogolanding<range_thres % change to final phase
        warning('Direct Entry')
        SELECTOR=5;
        SECANTBANK=finalbank;
        ROLLC=SECANTBANK;
    end
elseif SELECTOR==1 && load>g_thres % Transition to down control
    SELECTOR=2;
    ROLLC=SECANTBANK;
elseif SELECTOR==2 && state(5)<0 % Downcontrol
    ROLLC=SECANTBANK;
    if rangetogolanding<range_thres % change to final phase
        warning('Direct Entry')
        SELECTOR=5;
        SECANTBANK=finalbank;
        ROLLC=SECANTBANK;
    end
elseif SELECTOR==2 && state(5)>0 % Transition to upcontrol
    SELECTOR=3;
    ROLLC=SECANTBANK;
elseif SELECTOR==3 && load>g_thres % Upcontrol
    ROLLC=SECANTBANK;
    if rangetogolanding<range_thres %change to final phase
```

```
        warning('Loft entry')
        SELECTOR=5;
        SECANTBANK=finalbank;
        ROLLC=SECANTBANK;
    end
elseif SELECTOR==3 && load<g_thres % transition to Kepler
    SELECTOR=4;
    SECANTBANK=kepbank;
    ROLLC=SECANTBANK;
elseif SELECTOR==4 && load<g_thres % Kepler
    SECANTBANK=kepbank;
    ROLLC=SECANTBANK;
    if rangetogolanding<range_thres % change to final phase
        SELECTOR=5;
        SECANTBANK=finalbank;
        ROLLC=SECANTBANK;
    end
elseif SELECTOR==4 && load>g_thres % transition to final
    SECANTBANK=kepbank;
    ROLLC=SECANTBANK;
    if rangetogolanding<range_thres % change to final phase
        SELECTOR=5;
        SECANTBANK=finalbank;
        ROLLC=SECANTBANK;
    end
end

% With roll command from secant and transition logic, determine correct sign
% of bank angle and give to simulation the actual bank angle flown subject
% to rate and acceleration limits.  For first call, Ncall is uninitialized.
if Ncall==1
    sigma_sign=Bank_sign_Apollo(state,lon_site_bias*d2r,lat_site_bias*d2r,...
Ncall,SECANTBANK*d2r);
    ROLLC=abs(ROLLC)*sigma_sign;
    currentbank=shortest_bank_limiter(ROLLC*d2r,PHIBKRATEMAXDEG,...
PHIBKACCELMAXDEG,0)*r2d;
    Ncall=2;
elseif SELECTOR<=3
    sigma_sign=Bank_sign_Apollo(state,lon_site_bias*d2r,lat_site_bias*d2r,...
Ncall,SECANTBANK*d2r);
    ROLLC=abs(ROLLC)*sigma_sign;
    currentbank=shortest_bank_limiter(ROLLC*d2r,PHIBKRATEMAXDEG,...
PHIBKACCELMAXDEG,1)*r2d;
elseif SELECTOR>=4
    sigma_sign=Bank_sign_Apollo(state,lon_site*d2r,lat_site*d2r,...
Ncall,SECANTBANK*d2r);
    ROLLC=abs(ROLLC)*sigma_sign;
    currentbank=shortest_bank_limiter(ROLLC*d2r,PHIBKRATEMAXDEG,...
PHIBKACCELMAXDEG,1)*r2d;
end
varargout={currentbank,SELECTOR,iterations,biased_counts,SECANTBANK};
return
```

Skip Ranger

This is the file that simulates the planned trajectory given the skip bank angle. It also includes a sub-function, to determine when to do bank reversals.

```
function [range_m,ground,crosscheck]=skipranger(thetime,state,skipbank,...
    kepbank,finalbank,selector_in,mass,Sref,vehicle,K_drag,K_lift)

% Simulates skip phase with constant bank angle.
% Downrange is computed with respect to true landing site
% Crossrange bank reversals are done with respect to biased landing site.
%
% Outputs:
%   range_m     = range-to-go in meters, positive for undershoot, negative
%                   for overshoot
%   ground      = a flag;  ground=0 vehicle skipped out (alt>200km)
%                          ground=1 vehicle hit ground w/o skip out
%   crosscheck  = a flag;  =1 vehicle is left inside envelope
%                          =2 vehicle is left outside envelope
%                          =-1 vehicle is right inside envelope
%                          =-2 vehicle is right outside envelope
% Inputs:
%   thetime     = current time within outer simulation
%   state       = current state(r lon lat V gamma psi), dimensionless
%   skipbank    = skip phase bank angle 0<= skipbank <=180 (deg)
%   finalbank   = bank angle to fly for final phase (deg)
%   selector    = current phase 1=iniroll, 2=down, 3=up, 4=kep, 5=final
%   mass        = vehicle mass (kg)
%   S_ref       = vehicle reference area (m^2)
%   vehicle     = a flag;   1=CEV, 2=Apollo, 3=shuttle
%   K_drag      = Drag Density Gain (dimensionless)
%   K_lift      = Lift Density Gain(dimensionless)
% Functions Required:
%
%   sphereping.m
%   trim_alpha_CEV.m (or trim_alpha_Apollo.m)
%   CLCD_CEV.m (or CLCD_Apollo.m)
%   density.m
%   Vsound.m
%   myrungekutta_entry_bias.m
%   eqnofmotion_bias.m (required by myrungekutta_entry_bias.m)
%   Bank_sign_Apollo2.m (included function)
%   Matlab function "zeros"
%
%=========================================================================
global lon_site_bias lat_site_bias lon_site lat_site
persistent range_thres

%Earth parameters
R0=6378136.3;               %Earth Radius, m
g0=9.80665;                 %Earth surface gravity, m/s^2
Vscale=sqrt(R0*g0);         %velocity scaling factor, =7908.4m/s
tscale=sqrt(R0/g0);         %time scaling factor = 806.329s
omega=7.2921151e-5*tscale;  %converts rad/s to dimensionless
```

```
%Conversion factors
m2ft=3.28083989501;         %meters to feet
d2r=0.01745329251994;       %degrees to radians = pi/180;

%Integration setup
tend=2500;                  %total integration time, seconds
y=zeros(tend,7);
t=zeros(1,tend);
load=zeros(1,tend);
sigma=zeros(1,tend);
CR=zeros(1,tend);
Dpsi=zeros(1,tend);
CR_bias=zeros(1,tend);
selector_plot=zeros(1,tend);

y(1,:)=[state,0];           %initial state
Vfinal=500/Vscale/m2ft;     %final velocity dimensionless (500 ft/s)
endflag=0;
finalfirst=1;
hinput=10/tscale;           %step size, 10s=.0012399
p=1;                        %initialize counting variable
g_thres=0.05;               % load threshold to be inside atmosphere
crosscheck=0;           %initialize crosscheck variable
selector=selector_in;   %set initial phase selector to input value
if selector==1
    rangebank_cmd=0;
else
    rangebank_cmd=skipbank;
end
%range-to-go part along great circle distance
[arc, azimuth] = sphereping(y(p,2),y(p,3),lon_site_bias*d2r,...
    lat_site_bias*d2r);
rangetogolanding=arc*R0; %result in meters
if isempty(range_thres)
    % range threshold to begin transition to final phase (2000 km), meters
    range_thres=2000000;
    if rangetogolanding<3500000
        %reduce final phase threshold if range is short
        range_thres=500000;

    end
end

while 1
    %     r(p)=y(p,1);
    %     theta(p)=y(p,2);
    %     phi(p)=y(p,3);
    %     V(p)=y(p,4);
    %     gamma(p)=y(p,5);
    %     psi(p)=y(p,6);
    %     range convered=y(,p,7);

    alt=(y(p,1)-1)*R0; %result in meters
    [rho,rho_a] = density(alt/1000); %compute density
    Vs= Vsound(alt/1000);       %compute sound velocity
```

```
Vs=Vs/Vscale;               %non-dimensionalize sound velocity
Mach=y(p,4)/Vs;             %computer mach number
if vehicle==1
    alpha = trim_alpha_CEV(Mach);   %compute angle of attack
    [Cl,Cd]=CLCD_CEV(Mach,alpha);   %compute lift and drag coeff
elseif vehicle==2
    alpha = trim_alpha_Apollo(Mach);%compute angle of attack
    [Cl,Cd]=CLCD_Apollo(Mach,alpha);%compute lift and drag coeff
end
D=K_drag*rho*(Vscale*y(p,4))^2*Sref*Cd/(2*mass*g0); %Drag, g's
L=K_lift*rho*(Vscale*y(p,4))^2*Sref*Cl/(2*mass*g0); %Lift, g's

%Loading (results in g's)
load(p)=sqrt(L^2+D^2);
selector_plot(p)=selector;

%Crossrange with respect to true landing site (only used for plotting)
[arc, azimuth] = sphereping(y(p,2),y(p,3),lon_site*d2r,lat_site*d2r);
CR(p) = asin(sin(arc)*sin(y(p,6) - azimuth));% crossrange in rad
Y = 0.25/24/2*y(p,4)+0.3*1852/R0;
if abs(rangebank_cmd)<15
    Y   = Y/2;
end
Dpsi(p) = Y;

%Crossrange with respect to biased landing site
[arc, azimuth] = sphereping(y(p,2),y(p,3),lon_site_bias*d2r,...
    lat_site_bias*d2r);
CR_bias(p) = asin(sin(arc)*sin(y(p,6) - azimuth));% crossrange in rad

if (selector==1 && load(p)<g_thres)   %initial roll
    rangebank_cmd=0;
elseif (selector==1 && load(p)>g_thres)  %transition to skip
    selector=2;
    rangebank_cmd=finalbank + (skipbank-finalbank) * (arc*R0-...
        range_thres)/(rangetogolanding-range_thres);
elseif (selector>=2 && selector<=4 && (arc*R0)>range_thres) %skip
    rangebank_cmd=finalbank + (skipbank-finalbank) * (arc*R0-...
        range_thres)/(rangetogolanding-range_thres);
    if (selector==2 && y(p,5)>0) %selector changed only for plotting
        selector=3;
    elseif (selector==3 && load(p)<g_thres) %only for plotting
        selector=4;
    end
elseif (selector>=2 && selector<=4 && (arc*R0)<range_thres)
    %transition to final phase
    selector=5;
    rangebank_cmd=finalbank;
elseif (selector==5)  %final phase
    rangebank_cmd=finalbank;
else
    error('error in skipranger')
end

if finalfirst && selector==5
    if CR(p)>Dpsi(p) %vehicle is heading too far right, bias to left
```

```
        crosscheck=2; %outside envelope
    elseif CR(p)>0
        crosscheck=1;
    elseif CR(p)<-Dpsi(p)%vehicle is heading too left, bias to right
        crosscheck=-2;
    elseif CR(p)<0
        crosscheck=-1;
    else
        crosscheck=0;
    end
    finalfirst=0;
end

%Bank reversal is done with respect to biased landing site.
if p == 1    % initialization of bank_reversal
    sigma_sign=Bank_sign_Apollo2(y(p,:),lon_site_bias*d2r,...
        lat_site_bias*d2r,p,rangebank_cmd*d2r);
    rangebank_cmd=abs(rangebank_cmd)*sigma_sign;
    rangebank=rangebank_cmd;
else          % Normal bank reversal execution
    sigma_sign=Bank_sign_Apollo2(y(p,:),lon_site_bias*d2r,...
        lat_site_bias*d2r,p,rangebank_cmd*d2r);
    rangebank_cmd=abs(rangebank_cmd)*sigma_sign;
    rangebank=rangebank_cmd;
end

%record bank angle for plotting
sigma(p)=rangebank;

if endflag || alt<0
    ground=1;
    break
elseif alt>300000 %skipout
    ground=0;
    crosscheck=0;
    break
end

if y(p,4)<Vfinal*1.5 %Insures that final time step causes integration
    r=y(p,1);        %to stop at as close to Vfinal as possible
    phi=y(p,3);
    V=y(p,4);
    gamma=y(p,5);
    psi=y(p,6);
    Vdot=-D-sin(gamma)/r^2+omega^2*r*cos(phi)*(sin(gamma)*cos(phi)-...
        cos(gamma)*sin(phi)*cos(psi));
    if V+Vdot*hinput<=Vfinal
        hinput=(Vfinal-V)/Vdot;
        endflag=1;
    end
end

p=p+1;

y(p,:)=myrungekutta_entry_bias(@eqnofmotion_bias,t(p-1),y(p-1,:),...
    hinput,mass,Sref,rangebank,vehicle,K_drag,K_lift);
```

```
    t(p)=t(p-1)+hinput;
end

range_m=rangetogolanding-y(p,7)*R0;
%Make sure range is negative (overshoot), for skipouts
if (ground==0 && range_m > 0)
    range_m=-range_m;
end
return
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

function sigma_sign=Bank_sign_Apollo2(x,lonHAC,latHAC,NCall,bank)

% This function determine the sign of the current bank angle command
% (i.e., when to perform a bank reversal) according Apollo lateral logic
% Inputs:
%   x       - Array of the current 6 states of 3DOF motion, dimensionless
%   lonHAC  - longitude of the  HAC (target point), rad
%   latHAC  - latitude of the HAC (target point), rad
%   NCall   - Counter, integer (NCall=1 is the first call)
%   bank    - Commanded bank angle in rad
%
% Output:
%   sigma_sign  - The sign of the current bank command, 1 or -1

%=====================================================================
persistent Nrev2

R0      = 6378135.;     % m
r2d     = 57.29577951308232;
lon     = x(2);         % longitude, rad
lat     = x(3);         % latitude, rad
V       = x(4);         % relative velocity,dimless
psi     = x(6);         % heading angle, rad

if lonHAC <0
    lonHAC = 2*pi+lonHAC;
end
[range,Psi] = sphereping(lon,lat,lonHAC,latHAC);

Dpsi    = psi - Psi;
CR      = asin(sin(range)*sin(Dpsi));% crossrange in rad
%CR      = CR*R0/1000/1.852 ;     % crossrange in nm
KLat    = 0.25/24*0.5;
Latbias = 0.3*1852/R0;           % rad


%VSQ     = V^2;
Y       = KLat*V+Latbias;

%if V <0.1
% Y = 1*1852/R0;
%end

%KLat    = 0.3/24;
%Y       = KLat*VSQ+Latbias;
```

```
if abs(bank*r2d)<15
    Y   = Y*0.5;
end
Dpsi = Y;

% Determine initial sign of the bank angle
if NCall == 1
    if Psi >= psi
        Nrev2  = 1;
    else
        Nrev2  =-1;
    end
    sigma_sign = Nrev2;
    return
end


% Determine if a bank reversal is needed
if abs(CR) >= Dpsi
    if CR >= Dpsi && Nrev2 == 1
        Nrev2 =-1;     % reversal from positve to negative
    elseif CR <= -Dpsi && Nrev2 == -1
        Nrev2 = 1;     % reversal from negative to positive
    end
end

sigma_sign  = Nrev2;

return
```

Biasing Subroutine

This function modifies the landing site to a biased location.

```
function []=crossbias(state,crosscheck,multiplier)

% Christopher Brunner
% Latest version April 8,2008
% Computes a biased 'landing site' based on current state, landing site,
% direction to bias, and distance to bias.
% INPUTS
% state, a 6 element vector consisting of:
% r, dimensionless spacecraft radius
% theta, longitude radians
% phi, latitude radians
% Vin, dimensionless velocity
% gamma, flight path angle, radians
% psi, heading angle, radians
% crosscheck - gives direction to bias
% multiplier - gives multiple of biases to make
% OUTPUTS
% None (file updates lon_site_bias and lat_site_bias)
%=========================================================================
global lon_site_bias lat_site_bias lon_site lat_site
```

```
persistent delta_chi delta_site

lon=state(2);
lat=state(3);
psi=state(6);
d2r=0.017453292519943;        %degrees to radians = pi/180;
r2d=57.295779513082323;       %radians to degrees= 180/pi;

[arc, az] = sphereping(lon,lat,lon_site*d2r,lat_site*d2r);
if isempty(delta_chi)
    delta_chi=crosscheck*.15*d2r;
    chi=asin(sin(arc)*sin(psi-az));
    delta_Psi=-delta_chi * cos(chi) / (sin(arc)*cos(psi-az));
    a11=cos(lon_site*d2r-lon)*cos(lat_site*d2r)/sin(arc);
    a12=-sin(lon_site*d2r-lon)*sin(lat_site*d2r)/sin(arc);
    a21=-cos(lat_site*d2r)*cos(lat)*sin(lon_site*d2r-lon);
    a22=sin(lat)*cos(lat_site*d2r)-sin(lat_site*d2r)*cos(lat)*...
        cos(lon_site*d2r-lon);
    b1=cos(az)*delta_Psi;
    b2=0;
    det=a11*a22-a21*a12;
    if (abs(tan(lat)-sin(lat_site*d2r)*cos(lon_site*d2r-lon))< 1e-5  ...
            || lat_site==90)
        warning(' ill conditioned matrix')
    end
    delta_site(1)=(a22*b1-a12*b2)/det;
    delta_site(2)=(-b1*a21+a11*b2)/det;
end

lon_site_bias=lon_site+delta_site(1)*r2d*multiplier;
lat_site_bias=lat_site+delta_site(2)*r2d*multiplier;

return
```

Biased Equations of Motion

This function evaluates the right hand side of the equations of motion, producing the derivatives at the current state.

```
function dydt=eqnofmotion_bias(t,y,mass,Sref,bank,vehicle,K_drag,K_lift)

% Christopher Brunner
% Evaluate the 3-DOF equations of motion over a rotating spherical Earth;
% All state variables and time are dimensionless
% Use 1976 US Standard Atmosphere
% Use time as independent variable
% Inputs are:
% t, dimensionless time
% y, a vector consisting of 7 elements,
% r, the dimensionless position
% theta, longitude, in radians
% phi, latitude in radians
% V, dimensionless velocity
% gamma, flight path angle, negative down, in radians
% psi, heading angle, in radians
```

```
% s, range flown, scaled by Earth's radius, dimensionless
% mass, Vehicle mass in kilograms
% Sref, Vehicle area in square meters
% bank, current bank angle in degrees
% vehicle, a number corresponding to the vehicle being flown
% 1- Crew Exploration Vehicle
% 2- Apollo Command Capsule
% 3- Space Shuttle
% K_drag, a constant relating the nominal drag to the true drag
% K_lift, a constant relating the nominal lift to the true lift
%
% Output is a 7 element column vector consisting of the time derivatives
% of the input vector.
% April 2007

R0=6378136.3;                       %m
g0=9.80665;                        %m/s^2
Vscale=sqrt(R0*g0);                   %7908.4m/s
tscale=sqrt(R0/g0);                   %806.329s
omega=7.2921151e-5*tscale;    %converts rad/s to dimensionless
d2r=0.01745329251994;        %degrees to radians = pi/180;

r=y(1);
%theta=y(2);  %commented since not needed below
phi=y(3);
V=y(4);
gamma=y(5);
psi=y(6);
%s=y(7);    %commented since not needed below

%convert s/c radius from dimensionless to altitude in kilometers
alt=(r-1)*R0/1000;

%   Calculate the atmospheric density and gradient of density
%   with respect to altitude by 1976 US standard atmosphere.
%   alt in km (0<alt <130)
%   rho=density in kg/m^3
%   rho_a= d(rho)/d(alt)  unit (kg/m^3/km)
[rho,rho_a] = density(alt);

%Takes an input in kilometers of altitude and outputs in meters/second
Vs= Vsound(alt);

%convert from m/s to dimensionless
Vs=Vs/Vscale;

Mach=V/Vs;

if vehicle==1
    %takes an input of mach number and outputs angle of attack, alpha in
    %radians
    alpha = trim_alpha_CEV(Mach);
    %Compute CL and CD for the Apollo-like vehicle as functions of
    %Mach number and angle of attack in radians
    [Cl,Cd]=CLCD_CEV(Mach, alpha);
elseif vehicle==2
```

```
    %takes an input of mach number and outputs angle of attack, alpha in
    %radians
    alpha = trim_alpha_Apollo(Mach);
    %Compute CL and CD for the Apollo-like vehicle as functions of
    %Mach number and angle of attack in radians
    [Cl,Cd]=CLCD_Apollo(Mach, alpha);
end
%Lift and drag
D=K_drag*rho*(Vscale*V)^2*Sref*Cd/(2*mass*g0);
L=K_lift*rho*(Vscale*V)^2*Sref*Cl/(2*mass*g0);

sigma=bank*d2r;
rdot=V*sin(gamma);
thetadot=V*cos(gamma)*sin(psi)/(r*cos(phi));
phidot=V*cos(gamma)*cos(psi)/r;
Vdot=-D-sin(gamma)/r^2+omega^2*r*cos(phi)*(sin(gamma)*cos(phi)-cos(gamma)*...
sin(phi)*cos(psi));
gammadot=1/V*(L*cos(sigma) + (V^2-1/r)*cos(gamma)/r + 2*omega*V*...
cos(phi)*sin(psi) + omega^2*r*cos(phi)*(cos(gamma)*cos(phi)+...
sin(gamma)*cos(psi)*sin(phi)));
psidot=1/V*(L*sin(sigma)/cos(gamma) + V^2*cos(gamma)*sin(psi)*tan(phi)/r-...
    2*omega*V*(tan(gamma)*cos(psi)*cos(phi)-sin(phi)) + ...
    omega^2*r*sin(psi)*sin(phi)*cos(phi)/cos(gamma));
sdot=V*cos(gamma)/r;
dydt=[rdot thetadot phidot Vdot gammadot psidot sdot]';
```

Runge Kutta Bias

This is the 4th order Runge-Kutta integration function. It calls the right hand side of the equations of motion.

```
function x2=myrungekutta_entry_bias(f,t,x,h,mass,Sref,bank,vehicle,K_drag,...
K_lift)

%Implements a 4th order Runge Kutta integration method
%f is the function in the equation x'=f(t,x)
%t is the current time, scalar
%x are the initial conditions which should be a row vector
%h is the step size, scalar
%x2 is the new conditions, as a row vector
%Christopher Brunner

p1=(h*f(t,x,mass,Sref,bank,vehicle,K_drag,K_lift))';
p2=(h*f(t+.5*h,x+.5*p1,mass,Sref,bank,vehicle,K_drag,K_lift))';
p3=(h*f(t+.5*h,x+.5*p2,mass,Sref,bank,vehicle,K_drag,K_lift))';
p4=(h*f(t+h,x+p3,mass,Sref,bank,vehicle,K_drag,K_lift))';

x2=x+ (p1 + 2* p2 + 2* p3 +p4)/6;
```

Lift and Drag Ratio Filter

This describes the code for the filter on lift and drag.

```
function [X_filter]=filter_LD_ratios(r,V,X_meas,mass_nom,Sref,...
```

```
lift_or_drag,vehicle)

%drag or lift filter
%This function is a firs-order filter that slowly corrects, on the basis of
%measured lift or drag acceleration, the modeling mismatch (due mainly to
%atmospheric density and/or aerodynamic coefficient)
%Inputs
%   r:              Postion radius (Non-dimensional)
%   V               Velocity (Non-dimensional)
%   X_meas          Measured lift or drag (g's)
%   mass_nom        Nominal expected mass (kilograms)
%   Sref            Spacecraft area (square meters)
%   lift_or_drag    flag to indicate lift or drag is being filtered
%                   1= used as a lift filter
%                   0= used as a drag filter
%   vehicle         1= CEV
%                   2= Apollo
%Outputs
%   X_filter        Gain to be applied to nominal lift or drag (non-dim)
%
%Christopher Brunner
%June 7, 2007

persistent lift_past drag_past

if isempty(lift_past)&& isempty(drag_past)
    lift_past=1;
    drag_past=1;
end

%Earth parameters
R0=6378136.3;                   %m
g0=9.80665;                     %m/s^2
Vscale=sqrt(R0*g0);             %7908.4m/s

alt=(r-1)*R0; %resultant altitude in meters
[rho_nom,rho_a] = density(alt/1000);
[Vs]= Vsound(alt/1000);
Vs=Vs/Vscale;
Mach=V/Vs;
if vehicle==1
    alpha = trim_alpha_CEV(Mach);
    [CL,CD]=CLCD_CEV(Mach, alpha);
elseif vehicle==2
    alpha = trim_alpha_Apollo(Mach);
    [CL,CD]=CLCD_Apollo(Mach, alpha);
end
if lift_or_drag==1 %Lift filter

    K_lift_gain=0.9;
    L_nom=rho_nom*(Vscale*V)^2*Sref*CL*0.5/(mass_nom*g0);
    X_filter=K_lift_gain* lift_past + (1-K_lift_gain)*X_meas/L_nom ;
    lift_past=X_filter;

elseif lift_or_drag==0 %Drag filter
```

```
    K_drag_gain=0.9;
    D_nom=rho_nom*(Vscale*V)^2*Sref*CD*0.5/(mass_nom*g0);
    X_filter=K_drag_gain*drag_past + (1-K_drag_gain)*X_meas/D_nom ;
    drag_past=X_filter;

else
    error('Must specify lift or drag filter')
end
```

Arc Length Function - Sphereping

This computes the are length and azimuth between two points.

```
function [arc, azimuth] = sphereping(theta1,phi1,theta2,phi2)
% Everyting is in radian
% Evaluate the arc length  and azimuth angle between two points:
% P1(theta1,phi1)  and P2(tehta2,phi2) on the surface of a sphere
% theta = longitude, phi = latitude
% azimuth (measured clockwise from North) of the arc joining two point
%  -180 (deg) < azimuth < 180 deg
% Applicable for 0< arc < 3.1415927 (PI).
% The dimensional length of the arc will be radius * arc.
%
% Ping Lu
% Jan. 12, 2001

Dlon=theta2-theta1;
Dlat=phi2-phi1;
if abs(Dlon) <1.0e-8 && abs(Dlat) < 1.0e-8
    arc  = 0.0;
    azimuth = 0.0;
    'Warning: Two points coincide when calling sphere.m'
    return
else
    sn1 = sin(phi1);
    sn2 = sin(phi2);
    cn1 = cos(phi1);
    cn2 = cos(phi2);

    arc0= sn2*sn1+cn2*cn1*cos(Dlon);
    arc = acos(arc0);

    sn  = sin(Dlon)*cn2/sin(arc);   % sin(azimuth)
    cs  = (sn2*cn1-cn2*sn1*cos(Dlon))/sin(arc); % cos(azimuth)


    if abs(sn)>1
        sn = sn/abs(sn);
    end

    if abs(cs)>1
        cs = cs/abs(cs);
    end
```

```
    % Determine the correct quadrant for azimuth

    if cs >=0.0 && sn >=0.0        % 0  (deg) < azimuth < 90 (deg)
        azimuth = asin(sn);
    elseif cs<=0.0 && sn >=0.0 % 90 (deg) < azimuth < 180 (deg)
        azimuth = acos(cs);
    elseif cs >=0. && sn <=0.0 % -90(deg) < azimuth < 0 (deg)
        azimuth = asin(sn);
    elseif cs <=0.0 && sn <=0.0 % -180 (deg)< azimuth < -90 (deg)
        azimuth = -acos(cs);
    end
end
```

Shortest Bank Limiter

The following code determines the vehicle bank angle based on limited rate and acceleration. It will roll in the shortest direction to the commanded angle.

```
function bankCMD = shortest_bank_limiter(sigma_CMD,PHIBKRATEMAXDEG,...
    PHIBKACCELMAXDEG,DT_GUID)
% This function limits the rate and acceleration of bank angle command
% Modified by Chris Brunner
% This will decide change the bank angle in the shortest possible direction
% Inputs:
% sigma_CMD         - Current bank command before passing the limiter, rad,
% PHIBKRATEMAXDEG   - Maximum rate limit, deg/sec
% PHIBKACCELMAXDEG  - Maximum acceleration, deg/sec^2
% DT_GUID           - Time int between current time and previous call, sec.
%                     This input needs to be set to zero in the first call
% Output:
% bankCMD           - Bank angle command subject to the rate and
%                     acceleration limits, rad
% Note: Since this function stores past values of bank angle and
%  bank angle rate, it can only be used for bank angle in one calling
%  function.
%=========================================================================

if nargin==0
    shortest_tester
    return
end

persistent Past RatePast  % used as static variables

d2r=0.01745329251994;        %degrees to radians = pi/180;
r2d=57.29577951308232;       %radians to degrees= 180/pi;

if abs(DT_GUID)<1.0e-6
    bankCMD         = sigma_CMD;
    RatePast= 0;
```

```
    Past = sigma_CMD*r2d;
    return
end

Cmd = sigma_CMD*r2d;

%No ambiguity with definition of quadrant, if we have angle =0,=90
if Past>=0 && Past<=90
    PastQ=1;
elseif Past<=0 && Past>= -90
    PastQ=2;
elseif Past<=-90 && Past>=-180
    PastQ=3;
elseif Past>=90 && Past<=180
    PastQ=4;
elseif (Past==0)
    PastQ=5;
elseif (Past==90)
    PastQ=6;
elseif (Past==-90)
    PastQ=7;
elseif (Past==180)
    PastQ=8;
end

if Cmd>=0 && Cmd<=90
    CmdQ=1;
elseif Cmd<=0 && Cmd>=-90
    CmdQ=2;
elseif Cmd<= -90 && Cmd>= -180
    CmdQ=3;
elseif Cmd>=90 && Cmd<=180
    CmdQ=4;
elseif (Cmd==0)
    CmdQ=5;
elseif (Cmd==90)
    CmdQ=6;
elseif (Cmd==-90)
    CmdQ=7;
elseif (Cmd==180)
    CmdQ=8;
end

switch(PastQ)
    case(1)
        switch(CmdQ)
            case(1)
                %direct, do nothing
            case(2)
                %ok, do nothing
            case(3)
                %tricky,
                if Past+abs(Cmd)>180
                    Cmd=Cmd+360;
                end
                %otherwise leave default
```

```
        case(4)
            %ok, do nothing
        case(5) %Command is zero
        case(6) %Command is +90
        case(7) %Command is -90
        case(8) %Command is 180


    end
case (2)
    switch(CmdQ)
        case(1)
            %ok, do nothing
        case(2)
            %direct, do nothing
        case(3)
            %ok, do nothing
        case(4)
            %tricky
            if abs(Past)+Cmd>180
                Past=Past+360;
            end
        case(5) %Command is zero
        case(6) %Command is +90
        case(7) %Command is -90
        case(8) %Command is 180
            Cmd=-180;

    end
case(3)
    switch(CmdQ)
        case(1)
            %tricky
            if abs(Past)+Cmd >180
                Past=Past+360;
            end
        case(2)
            %ok, do nothing
        case(3)
            %direct, do nothing
        case(4)
            Past=Past+360;
            %tricky
        case(5) %Command is zero
        case(6) %Command is +90
        case(7) %Command is -90
            %Past=Past+360
        case(8) %Command is 180
            Cmd=-180;
    end
case(4)
    switch(CmdQ)
        case(1)
            %ok, do nothing
        case(2)
            %tricky
```

```
            if Past+abs(Cmd)>180
                Cmd=Cmd+360;
            end
        case(3)
            Cmd=Cmd+360;
            %tricky
        case(4)
            %direct, do nothing
        case(5) %Command is zero
        case(6) %Command is +90
        case(7) %Command is -90
            Cmd=Cmd+360;
        case(8) %Command is 180
    end
case(5) %Past flown angle is zero
    switch(CmdQ)
        case(1)
        case(2)
        case(3)
        case(4)
        case(5) %Command is zero
        case(6) %Command is +90
        case(7) %Command is -90
        case(8) %Command is 180
    end
case(6) %past is +90
    switch(CmdQ)
        case(1)
        case(2)
        case(3) %tricky,
            Cmd=Cmd+360;
        case(4)
        case(5) %Command is zero
        case(6) %Command is +90
        case(7) %Command is -90
        case(8) %Command is 180
    end
case(7) %past is -90
    switch(CmdQ)
        case(1)
        case(2)
        case(3)
        case(4) %tricky
            Past=270;
        case(5) %Command is zero
        case(6) %Command is +90
        case(7) %Command is -90
        case(8) %Command is 180
            Cmd=-180;
    end
case(8) %Past is 180
    switch(CmdQ)
        case(1)
        case(2)
            Past=-180;
        case(3)
```

```
                Past=-180;
            case(4)
            case(5) %Command is zero
            case(6) %Command is +90
            case(7) %Command is -90
                Cmd=270;
            case(8) %Command is 180
        end
end

rateDesired = (Cmd - Past )/DT_GUID;
if  rateDesired < -PHIBKRATEMAXDEG
    rateDesired = -PHIBKRATEMAXDEG;
elseif rateDesired > PHIBKRATEMAXDEG
    rateDesired = PHIBKRATEMAXDEG;
end

accelDesired = (rateDesired - RatePast)/DT_GUID;
if  accelDesired < -PHIBKACCELMAXDEG
    accelActual = -PHIBKACCELMAXDEG;
elseif accelDesired > PHIBKACCELMAXDEG
    accelActual = PHIBKACCELMAXDEG;
else
    accelActual = accelDesired;
end

rateActual = RatePast + accelActual*DT_GUID;
RatePast  = rateActual;
Cmd       = Past + rateActual*DT_GUID;
Past   = Cmd;

if Cmd>180
    Cmd=Cmd-360;
elseif Cmd< -180
    Cmd=Cmd+360;
end
if Past>180
    Past=Past-360;
elseif Past< -180
    Past=Past+360;
end

bankCMD = Cmd*d2r;
return
```

APPENDIX C. APOLLO GUIDANCE CODES

Apollo Guidance

This is the Apollo guidance in its modified versions as described in Chapter 3.

```
function [varargout]=Apollo_guidance(p,state,load,bank)

%This is the Apollo Skip Entry Guidance modified slightly to enable running
%the function in Matlab.
%Code started May 31,2006
%By Christopher Brunner
%
%INPUTS
%p, current time (integer - not used)
%state is a vector of 6 states compromised of
%r, dimensionless spacecraft radius
%long, longitude (radians)
%lat, latitude (radians)
%Vin, dimensionless velocity
%gamma, flight path angle, (radians)
%psi, heading angle, (radians)
%load, spacecraft acceleration, (g's)
%bank, current bank angle, (degrees)
%
%OUTPUTS (see final line of code, vargargout=variable arguments out)
%flownbank - rate limited bank angle  actually flown by the vehicle(deg)
%ROLLC*r2d - bank angle commanded by guidance (deg)
%selectorout - phase of guidance
%               1-initial roll
%               2-huntest/constant drag
%               3-upcontrol
%               4-Kepler
%               5-final (predict3)
%
%References:
%Morth, R., "Reentry Guidance for Apollo," MIT/IL R-532 Vol. I, 1966.
%
%Moseley, P. E., "The Apollo Entry Guidance: A Review of the Mathematical
%Development and its Operational Characteristics," TRW Note No.
%69-FMT-791, Dec. 1969, Houston,TX.
%
%Levine, G.M., "Apollo Guidance, Navigation, and Control," MIT-Charles
%Stark Draper Laboratory. R-577, Guidance System Operations Plan for
%Manned CM Earth Orbital and Lunar Missions using Program Colossus 3,
```

```
%Section 5, Guidance Equations (Rev 15) January 1972, Cambridge,
%Massachusetts.
%
%Bairstow, S. H., "Reentry Guidance with Extended Range Capability for Low
% L/D Spacecraft," S. M. Thesis, Department of Aeronautics and
% Astronautics, MIT, Feb. 2006.
%
%Of the above, the 1st ref, is the earliest form of the guidance, the
%second is an update, and the 3rd contained many of the gains and constants
%necessary for the code to run.  The last contained a mention of some
%errors found in the earlier references.

if nargin==0
    secant_disp;
    return
end

global lon_site lat_site  % global variables for landing site (deg).

persistent GONEPAST HIND P05GSW INRLSW NOSWITCH SELECTOR Q7...
    FACTOR LOD LAD KLAT LOVERDCMINR LOVERD DIFFOLD DLEWD Q2 KA K2ROLL...
    D0 LEWD V1 A0 FACT1 FACT2 ALP VS1 AHOOK DHOOK VL Ncall%LATSW

%Define constants
R0=6378136.3;                       %m
g0=9.80665;                         %m/s^2
m2ft=3.28083989501;                 %meters to feet
Vscale=sqrt(g0*R0);                 %7908.4m/s
d2r=0.01745329251994;        %degrees to radians = pi/180;
r2d=57.29577951308232;       %radians to degrees= 180/pi;

PHIBKRATEMAXDEG=20;         %max roll rate (deg/s)
PHIBKACCELMAXDEG=10;        %max acceleration rate (deg/s^2)

% the following are all dimensionless
% rin=state(1);              %input vehicle geocentric position
long=state(2);              %input vehicle longitude
lat=state(3);               %input vehicle latitude
Vin=state(4);               %input vehicle relative velocity
gamma=state(5);             %input vehicle flight path angle
psi=state(6);               %input vehicle heading agnle

%conversion of inputs
% R=rin*R0*m2ft;            %Vehicle geocentric position(ft)
V=Vin*Vscale*m2ft;          %Vehicle relative velocity (ft/s)
D=load*g0*m2ft;             %Vehicle load (ft/s^2)

%constants and gains
ATK=3437.7468;          %Earth Radius (nm)
GS=32.2;                %Gravity (fpss)
HS=28500;               %Atm Scale height (ft)
% KWE=1546.70168;         %Equatorial Earth Rate (fps)
RE=21202900;            %Earth Radius (ft)
% REO=20925738.2;        %Earth equatorial radius (ft)
VSAT=25766.1973;        %Satellite Velocity at RE (fps)
%WIE=0.0000729211505;   %Earth rate (rad/sec)
```

```
C1=1.25;                %Factor in ALP computation (none)
C16=0.01;               %CONSTD gain on drag (none)
C17=0.001;              %CONSTD gain on RDOT (none)
C18=500;                %Bias velocity for final phase start (ft/s)
C20=210;                %Max drag for down-lift (ft/s^2)
C21=140;                %Drag for no lateral switch (ft/s^2)
CHOOK=0.25;             %factor in AHOOK computation (none)
CH1=1.0;                %factor in GAMMA1 computation (none)
COS15=0.965;            %cos (15 deg.) (none)
DLEWD0=-0.05;           %initial variation in LEWD (none)
%DT=2;                  %computation cycle time interval (sec)
GMAX=257.6;             %Maximum acceleration (ft/s^2)
KA1=1.3;                %factor in KA calculation (g's)
KA2=0.2;                %factor in KA calculation (g's)
KA3=90;                 %factor in D0 calculation (ft/s^2)
KA4=40;                 %factor in D0 calculation (ft/s^2)
KB1=0.5;                %optimized upcontrol gain (none)
KB2=0.0025;             %optimized upcontrol gain (none)
KDMIN=0.5;              %increment in Q7F, detects end of Kep phase(ft/s^2)
%KTETA=1000;            %time of flight constant (none)
KLAT1=1/24;             %factor in KLAT calculation (none)
K44=19749550;           %gain used in initial roll section (ft/s)
LATBIAS=0.41252961;     %lateral switch bias term (nm)
LEWD1=0.15;             %nominal upcontrol L/D (none)
POINT1=0.1;             %factor to reduce upcontrol gain (none)
Q3=0.07;                %final phase drange/D V (nm / ft/s)
Q5=7050;                %final phase drange/D gamma (nm/rad)
Q6=0.0349;              %final phase initial flight path angle (rad)
Q7F=6;                  %min drag for upcontrol (ft/s^2)
Q7MIN=805;              %min value for Q7 in FACTOR calculation (ft/s^2)
Q19=0.5;                %factor in GAMMAL1 calculation (none)
% VFINAL1=27000;        %velocity to start final phase on ini entry (ft/s)
VFINAL=26600;           %factor in initial up-down calculation (ft/s)
VLMIN=18000;            %minimum VL [exit velocity for upcontrol] (ft/s)
% VMIN=VSAT/2;          %velocity to switch to relative vel (ft/s)
VRCONTRL=700;           %RDOT to start into HUNTEST (ft/s)
VQUIT=1000;             %velocity to stop steering (ft/s)

% LADPAD=0.3;           %nominal vehicle L/D stated in documents
LADPAD=0.387700900791527; %Vehicle max L/D for vehicle I'm using.
% LODPAD=0.18;          %final phase L/D stated in documents
LODPAD=0.21;            %final phase L/D for vehicle I'm using
HEADSUP=1;              %indicator for initial roll

%This section is the initialization routine (called just once)
if isempty(INRLSW)
    GONEPAST=1;         %indicates overshoot of target (switch 1 or 0)
    %RELVELSW=0;        %relative velocity switch (I always use relative)
    %EGSW=0;            %final phase switch
    HIND=0;             %inidicates iteration in huntest
    %     LATSW=1;      %switch prevents rolling under during upcontrol
    %     P05GSW=0;     %indicates drag greater than .05 times  gravity
    INRLSW=0;           %indicates initial roll started
    NOSWITCH=0;         %inhibit lateral switch
```

```
    %This is section is a modification of the Apollo targeting to measure
    %the great circle distance
    [THETA, azimuth] = sphereping(long,lat,lon_site*d2r,lat_site*d2r);
    LATANG=asin(sin(THETA)*sin(psi-azimuth));
    K2ROLL=-sign(LATANG);

    Q7=Q7F;                          %minimum drag for UPCONTROL
    SELECTOR='INITIALROLL';          %SELECTOR is the general "mode"
    GOTO='INITIALROLL';              %GOTO specifies which module to call
    FACTOR=1;
    LOD=LODPAD;                      %final phase L/D
    LAD=LADPAD;                      %nominal L/D
    KLAT=KLAT1*LAD;                  %lateral switch gain
    LOVERDCMINR=LAD*COS15;           %minimum commandable L/D (within 15deg)
    LOVERD=-LAD*sign(HEADSUP);       %choose an initial bank angle
    DIFFOLD=0;                       %initialize for HUNTEST
    DLEWD=DLEWD0;                    %initialize for HUNTEST
    LEWD=LEWD1;                      %initialize for HUNTEST
    Q2=-1992+3500*LAD;               %final phase range (21600 nm)
else %This is the targeting routine that is called everytime except first
    GOTO=SELECTOR;
    [THETA, azimuth] = sphereping(long,lat,lon_site*d2r,lat_site*d2r);
    LATANG=asin(sin(THETA)*sin(psi-azimuth));
end

%TARGETING
VSQ=V^2/VSAT^2;     %normalized square of velocity (none)
LEQ=(VSQ-1)*GS;     %gravity plus centripetal acceleration
RDOT=V*sin(gamma);  %altitude rate (ft/s)

%This is an adaptation to determine if vehicle has past landing site
if abs(azimuth-psi)>pi/2 %Landing site is passed
    GONEBY=1;
    %     RTOGO=-THETA;
else %not yet passed
    GONEBY=0;
    %     RTOGO=THETA;
end

%Checking for drag to be bigger than .05 times gravity.
if D-.05*GS<0
    P05GSW=1;
else
    P05GSW=0;
end

runguidance=1;

while runguidance
    switch(GOTO)
        case ('INITIALROLL')
            if INRLSW==0; %this run on the first pass only
                if P05GSW==1; %start guidance
                    INRLSW=1;
                    GONEPAST=0;
                    KA=(KA1*(LEQ/GS)^3+KA2)*GS;
```

```
            if KA-1.5*GS>0 %test if drag is high enough for linear
                           %theory to be applicable
                KA=1.5*GS;
            end
            if V-VFINAL<0
                SELECTOR='KEPLER';  %set for ballistic mode
                GOTO='310';         %but execute lateral logic
            else
                D0=KA3*LEQ/GS+KA4;  %compute ref drag level
                %determine orientation of lift vector
                if V-VFINAL+K44*(RDOT/V)^3>0
                    LOVERD=-LAD; %lift down
                else
                    LOVERD=LAD; %lift up
                end
                GOTO='310';
            end
        else
            GOTO='310';
        end
    else
        if RDOT+VRCONTRL<0 %if rdot is smaller than -700 ft/s,
                           %wait for planning
            if D-KA>0
                GOTO='CONSTD'; %drag high enough for closed loop
                               %drag tracking
            else
                GOTO='310'; %drag too low, maintain lateral logic
            end
        else %otherwise starting planning
            SELECTOR='HUNTEST';
            GOTO='HUNTEST';
        end
    end
case ('HUNTEST') %
    if RDOT<0
        V1=V+RDOT/LAD; %compute initial velocity for upcontrol
        A0=(V1/V)^2*(D+RDOT^2/(2*C1*HS*LAD)); %compute initial
                                              %drag for upcontrol
        A1=A0;
    else
        V1=V+RDOT/LEWD; %compute initial velocity for upcontrol
        A0=(V1/V)^2*(D+RDOT^2/(2*C1*HS*LEWD)); %compute initial
                                               %drag for upcontrol
        A1=D;
    end
    if LOVERD<0
        V1=V1-VQUIT;
    end
    Q7=Q7F;
    ALP=2*C1*A0*HS/(LEWD*V1^2);
    FACT1=V1/(1-ALP);
    FACT2=ALP*(ALP-1)/A0;
    VL=FACT1*(1-sqrt(FACT2*Q7+ALP)); %upcontrol exit velocity
    if VL-VLMIN<0 %if velocity is too small, start final phase
        SELECTOR='PREDICT3';
```

```
        %                 EGSW=1;
        GOTO='PREDICT3';
    else
        if VL-VSAT>0 %is exit velocity bigger than circular vel?
            SELECTOR='HUNTEST'; %if so, burn off more.
            GOTO='CONSTD';
        else
            if V1-VSAT>0%is ini upcontrol vel bigger than circular?
                VS1=VSAT;
            else
                VS1=V1;
            end
            DVL=VS1-VL; %initial vel for upcontrol minus exit vel
            DHOOK=((1-VS1/FACT1)^2-ALP)/FACT2;
            AHOOK=CHOOK*(DHOOK/Q7-1)/DVL;
            GAMMAL1=LEWD*(V1-VL)/VL;
            %flight path angle at upcontrol exit
            GAMMAL=GAMMAL1-CH1*GS*DVL^2*(1+AHOOK*DVL)/(DHOOK*VL^2);
%if that angle is negative, replan since cannot reach that condition
            if GAMMAL<0
                VL=VL+GAMMAL*VL/(LEWD-(3*AHOOK*DVL^2+2*DVL)*...
                    (CH1*GS/(DHOOK*VL)));
                %acceleration at exit of upcontrol
                Q7=((1-VL/FACT1)^2-ALP)/FACT2;
                GAMMAL=0;
            end
            GAMMAL1=GAMMAL1*(1-Q19)+Q19*GAMMAL;
            GOTO='RANGEPREDICTION';
        end
    end

 case ('RANGEPREDICTION')
    VBARS=VL^2/VSAT^2;
    COSG=1-GAMMAL^2/2;
    E=sqrt(1+(VBARS-2)*COSG^2*VBARS);
    ASKEP=2*ATK*asin(VBARS*COSG*GAMMAL/E); %ballistic range
    ASP1=Q2+Q3*VL; %final phase range
    ASPUP=ATK/RE*(HS/GAMMAL1)*log(A0*VL^2/(Q7*V1^2));%upphase range
    ASP3=Q5*(Q6-GAMMAL);    %gamma correction
    ASPDWN=-RDOT*V*ATK/(A0*LAD*RE); %range to pullout
    ASP=ASKEP+ASP1+ASPUP+ASP3+ASPDWN; %total range
    DIFF=ATK*THETA-ASP;
    if abs(DIFF)-25<0;
        SELECTOR='UPCONTROL'; %plan is ok, so fly it now!
        GOTO='UPCONTROL';
    else
        if HIND==0 && DIFF<0
            DIFFOLD=DIFF; %planned range too long, fly const drag
            Q7=Q7F;
            GOTO='CONSTD';
        else %planned range is undershoot
            DLEWD=DLEWD*DIFF/(DIFFOLD-DIFF);
            if LEWD+DLEWD<0         %adjust upcontrol reference L/D
                DLEWD=-LEWD/2;
            end
            LEWD=LEWD+DLEWD;
```

```
                HIND=1;
                DIFFOLD=DIFF;
                Q7=Q7F;
                GOTO='HUNTEST';
            end
        end

    case ('CONSTD')
        %control law to maintain constant drag
        LOVERD=-LEQ/D0+C16*(D-D0)-C17*(RDOT+2*HS*D0/V);
        GOTO='NEGTEST';
    case ('NEGTEST')
        if D-C20>0 %test for maximum drag for down lift.
            %                 LATSW=0;
            if LOVERD<0
                LOVERD=0;
            end
        end
        GOTO='310'; %lateral logic

    case ('UPCONTROL')
        if D-C21>0
            NOSWITCH=1; %high drag so roll reversals are prohibited
        end
        if V-V1>0 %Have not yet hit the pullout condition
            RDTR=LAD*(V1-V);
            DR=(V/V1)^2*(A0-RDTR^2/(2*C1*HS*LAD));
            LOVERD=LAD+C16*(D-DR)-C17*(RDOT-RDTR); %control law
            GOTO='NEGTEST';
        else
            if D-Q7>0 %is drag greater than minimum?
                if RDOT<0 && V-VL-C18<0 %going back down and
                    SELECTOR='PREDICT3'; %start final phase
                    GOTO='PREDICT3';
                    %                           EGSW=1;
                else %continue to fly the reference trajectory to exit
                    if A0-D<0%drag greater than start drag,full lift up
                        LOVERD=LAD;
                        GOTO='310';
                    else %remainder of the computation figures out the
%                          %needed L/D to fly the upcontrol trajectory.
                         %final phase ref velocity
                        VREF=FACT1*(1-sqrt(FACT2*D+ALP));
                        if VREF-VS1>0
                            RDOTREF=LEWD*(V1-VREF);
                        else
                            RDOTREF=LEWD*(V1-VREF)-CH1*GS*(VS1-...
                                VREF)^2*(1+AHOOK*(VS1-VREF))/...
                                (DHOOK*VREF);
                        end
                        if D-Q7MIN>0
                            FACTOR=(D-Q7)/(A1-Q7);
                        end
                        TEM1B=-KB2*FACTOR*(KB1*FACTOR*(RDOT-RDOTREF)...
                            +V-VREF);
                        if abs(TEM1B)-POINT1>0
```

```
                        TEM1B=(POINT1+POINT1*(abs(TEM1B)-POINT1)...
                            )*sign(TEM1B);
                    end
                    LOVERD=LEWD+TEM1B;
                    GOTO='NEGTEST';
                end
            end
        else %drag has decreased, switch to kepler
            SELECTOR='KEPLER';
            GOTO='KEPLER';
        end
    end
case ('KEPLER')
    if P05GSW==0 %load is small, so neutral bank commanded
        ROLLC=0;
    end
    if D-(Q7F+KDMIN)>0
        %                  EGSW=1;
        SELECTOR='PREDICT3'; %end ballistic phase
        GOTO='PREDICT3';
    else
        ROLLC=bank*pi/180; %maintain previous bank
        GOTO='380';
    end
case ('PREDICT3')
    if V-VQUIT>0 %not yet to end of guidance
        %this matrix is the reference trajectory
        inter=[994 -690.9 41.15 .002507 -.03446  3.7    6.10*2;...
            2103 -719   60    .003582 -.05551 10.4   10.91*2;...
            3922 -694   81.5  .007039 -.09034 23.6   21.64*2;...
            6295 -609   93.9  .01446  -.1410  46.3   48.35*2;...
            8531 -493   98.5  .02479  -.1978  75.4   93.72*2;...
            10101 -416  102.3  .03391  -.2372  99.9  141.1*2;...
            14014 -352  118.7  .06139  -.3305 170.9  329.4;...
            15951 -416  125.2  .07683  -.7500 210.3  465.5;...
            18357 -566  120.4  .09982  -1.000 266.8  682.7;...
            20829 -781   95.4  .1335   -1.000 344.3  980.5;...
            23090 -927   28.1  .2175   -2.021 504.8 1385;...
            23500 -820    6.4  .3046   -3.354 643.0 1508;...
            35000 -820    6.4  .3046   -3.354 794.3 1508];

        %this is the interpolation scheme on the reference traj.
        for k=1:12 %index runs until 12 since there are 13
            if inter(k,1) <=V && V<=inter(k+1,1)
                break
            end
        end
        grad=(V-inter(k,1))/(inter(k+1,1)-inter(k,1));
        RDOTREFV=inter(k,2)+grad*(inter(k+1,2)-inter(k,2));
        DREFRV=inter(k,3)+grad*(inter(k+1,3)-inter(k,3));
        F2V=inter(k,4)+grad*(inter(k+1,4)-inter(k,4));
        F1V=inter(k,5)+grad*(inter(k+1,5)-inter(k,5));
        RTOGOV=inter(k,6)+grad*(inter(k+1,6)-inter(k,6));
        F3V=inter(k,7)+grad*(inter(k+1,7)-inter(k,7));

        %compute ranging potential
```

```
        PREDANGL=RTOGOV+F2V*(RDOT-RDOTREFV)+F1V*(D-DREFRV);
        if GONEPAST==0
            if GONEBY==0
                %DNRNGERR=PREDANGL-THETA*ATK; %a display quantity
                %compute vehicle L/D to fly to target
                LOVERD=LOD+4*(ATK*THETA-PREDANGL)/F3V;
            else
                GONEPAST=1;
                %DNRNGERR=MAXRNG;
                LOVERD=-LAD; %landing site passed, full lift down
            end
        else
            LOVERD=-LAD; %landing site prev passed, full lift down
        end
        GOTO='GLIMITER';
    else %velocity is small, cmd neutral bank for end of guidance
        ROLLC=0;
        GOTO='310';
    end

case ('GLIMITER')
    if GMAX/2-D>0 %g-load is less than half of max, fly regular cmd
        GOTO='310';
    else
        if GMAX-D>0 %g-load is between one half and max g-load
            X=sqrt(2*HS*(GMAX-D)*(LEQ/GMAX+LAD)+(2*HS*GMAX/V)^2);
            if RDOT+X>0 %predicted g-load will be ok, fly cmd
                GOTO='310';
            else %g-load in danger of being exceeded, fly lift up
                'g load nearly exceeded';
                LOVERD=LAD;
                GOTO='310';
            end
        else %g-load is exceeded, fly lift up.
            'g-load definitely exceeded';
            LOVERD=LAD;
            GOTO='310';
        end
    end
case ('310') %lateral logic
    LOVERD1=LOVERD;
    if GONEPAST==0;
        %an error is noted by Bairstow here in the references above
        Y=KLAT*VSQ+LATBIAS/ATK;
        if abs(LOVERD)-LOVERDCMINR<0
            %determine if crossrange is outside envelope
            if K2ROLL*LATANG-Y>0
                if NOSWITCH==1
                    %prohibit roll reversals if drag is large
                    %during upcontrol
                else
                    K2ROLL=-K2ROLL; %reversal bank direction
                end
            end
        else
            %commanded bank angle is within 15deg of lift up,
```

```
                    %reduce envelope size
                    Y=Y/2;
                    if K2ROLL*LATANG>0
                        if K2ROLL*LATANG-Y>0
                            if NOSWITCH==1
                                %as before, no reversal if drag is large
                                %during upcontrol
                            else
                                K2ROLL=-K2ROLL;
                            end
                        end
                    else
                        %limits bank to within 15 deg of full lift up or
                        %down
                        LOVERD1=LOVERDCMINR*sign(LOVERD);
                    end
                end
            end
            LOVERD1OVERLAD=LOVERD1/LAD; %compute actual vehicle L/D
            %if L/D is bigger than one, make sure it's between +/- 1
            if abs(LOVERD1OVERLAD)-1>0
                LOVERD1OVERLAD=sign(LOVERD);
            end
            %compute roll command for autopilot (rad)
            ROLLC=K2ROLL*acos(LOVERD1OVERLAD);
            NOSWITCH=0; %reset upcontrol reversal prohibition
            GOTO='380'; %end guidance cycle.

        case ('380')
            runguidance=0;
    end
end

%Determine which phase the guidance is in
if strcmp(SELECTOR,'INITIALROLL')
    selectorout=1;
elseif strcmp(SELECTOR,'HUNTEST')
    selectorout=2;
elseif strcmp(SELECTOR,'UPCONTROL')
    selectorout=3;
elseif strcmp(SELECTOR,'KEPLER')
    selectorout=4;
elseif strcmp(SELECTOR,'PREDICT3')
    selectorout=5;
else
    selectorout=0;
end

%Call "autopilot" which rate limits bank angle to reverse in the shortest
%direction
if isempty(Ncall) %first call to limiter.
    Ncall=1;
    flownbank=shortest_bank_limiter(ROLLC,PHIBKRATEMAXDEG,...
        PHIBKACCELMAXDEG,0)*r2d;
else %future calls to bank limiter.
    flownbank=shortest_bank_limiter(ROLLC,PHIBKRATEMAXDEG,...
```

```
        PHIBKACCELMAXDEG,2)*r2d;
end

%outputs
varargout={flownbank,ROLLC*r2d,selectorout};
```

Apollo Final Guidance

This is strictly the Apollo guidance for the final phase only with everything that is unnecessary removed.

```
function [varargout]=Apollo_guidance_final(p,state,load,bank)

%NOTE: This final is extracted from the regular Apollo skip guidance and
%contains only the final phase of guidance with minimal gains, etc.
%It also contains the g-limiter and lateral logic functions.
%INPUTS
%p, current time (integer - not used)
%state is a vector of 6 states compromised of
%r, dimensionless spacecraft radius
%long, longitude (radians)
%lat, latitude (radians)
%Vin, dimensionless velocity
%gamma, flight path angle, (radians)
%psi, heading angle, (radians)
%load, spacecraft acceleration, (g's)
%bank, current bank angle, (degrees)
%
%OUTPUTS (see final line of code, vargargout=variable arguments out)
%flownbank - rate limited bank angle  actually flown by the vehicle(deg)
%ROLLC*r2d - bank angle commanded by guidance (deg)
%selectorout - phase of guidance
%               1-initial roll
%               2-huntest/constant drag
%               3-upcontrol
%               4-Kepler
%               5-final (predict3)
%
%Code started May 31,2006
%By Christopher Brunner
%
%References:
%Morth, R., "Reentry Guidance for Apollo," MIT/IL R-532 Vol. I, 1966.
%
%Moseley, P. E., "The Apollo Entry Guidance: A Review of the Mathematical
%Development and its Operational Characteristics," TRW Note No.
%69-FMT-791, Dec. 1969, Houston,TX.
%
%Levine, G.M., "Apollo Guidance, Navigation, and Control," MIT-Charles
%Stark Draper Laboratory. R-577, Guidance System Operations Plan for
%Manned CM Earth Orbital and Lunar Missions using Program Colossus 3,
%Section 5, Guidance Equations (Rev 15) January 1972, Cambridge,
%Massachusetts.
```

```
%
%Bairstow, S. H., "Reentry Guidance with Extended Range Capability for Low
% L/D Spacecraft," S. M. Thesis, Department of Aeronautics and
% Astronautics, MIT, Feb. 2006.
%
%Of the above, the 1st ref, is the earliest form of the guidance, the
%second is an update, and the 3rd contained many of the gains and constants
%necessary for the code to run.  The last contained a mention of some
%errors found in the earlier references.

if nargin==0 %this provided a way in Matlab of me "running" from this file.
    secant_disp;
    return
end

global lon_site lat_site %bring global variables for landing site (deg).

persistent GONEPAST INRLSW NOSWITCH SELECTOR LOD LAD KLAT...
            LOVERDCMINR LOVERD K2ROLL Ncall

%Define constants
R0=6378136.3;                       %m
g0=9.80665;                         %m/s^2
m2ft=3.28083989501;                 %meters to feet
Vscale=sqrt(g0*R0);                 %7908.4m/s
d2r=0.01745329251994;         %degrees to radians = pi/180;
r2d=57.29577951308232;        %radians to degrees= 180/pi;

PHIBKRATEMAXDEG=20;          %max roll rate (deg/s)
PHIBKACCELMAXDEG=10;         %max acceleration rate (deg/s^2)

% the following are all dimensionless
% rin=state(1);               %input vehicle geocentric position
long=state(2);               %input vehicle longitude
lat=state(3);                %input vehicle latitude
Vin=state(4);                %input vehicle relative velocity
gamma=state(5);              %input vehicle flight path angle
psi=state(6);                %input vehicle heading agnle

%conversion of inputs
% R=rin*R0*m2ft;             %Vehicle geocentric position(ft)
V=Vin*Vscale*m2ft;           %Vehicle relative velocity (ft/s)
D=load*g0*m2ft;              %Vehicle load (ft/s^2)

%constants and gains
ATK=3437.7468;          %Earth Radius (nm)
GS=32.2;                %Gravity (fpss)
HS=28500;               %Atm Scale height (ft)
VSAT=25766.1973;        %Satellite Velocity at RE (fps)

COS15=0.965;%cos (15 deg.) (none)
GMAX=257.6;%Maximum acceleration (ft/s^2)
KLAT1=1/24;%factor in KLAT calculation (none)
LATBIAS=0.41252961;%lateral switch bias term (nm)
VQUIT=1000;%velocity to stop steering (ft/s)
```

```
% LADPAD=0.3;           %nominal vehicle L/D stated in documents
LADPAD=0.387700900791527; %Vehicle max L/D for vehicle I'm using.
% LODPAD=0.18;          %final phase L/D stated in documents
LODPAD=0.21;            %final phase L/D for vehicle I'm using
HEADSUP=1;              %indicator for initial roll

%This section is the initialization routine (called just once)
if isempty(INRLSW)
    GONEPAST=0;                     %indicates overshoot of target
    INRLSW=1;                       %indicates initial roll started
    NOSWITCH=0;                     %inhibit lateral switch
 %This is section is a modification of the Apollo targeting to measure
    %the great circle distance
    [THETA, azimuth] = sphereping(long,lat,lon_site*d2r,lat_site*d2r);
    LATANG=asin(sin(THETA)*sin(psi-azimuth));
    K2ROLL=-sign(LATANG);
    SELECTOR='PREDICT3';            %SELECTOR is the general mode we're in
    GOTO='PREDICT3';                %GOTO specifies which module to call.
    LOD=LODPAD;                     %final phase L/D
    LAD=LADPAD;                     %nominal L/D
    KLAT=KLAT1*LAD;                 %lateral switch gain
    LOVERDCMINR=LAD*COS15;          %minimum commandable L/D
    LOVERD=-LAD*sign(HEADSUP);      %choose an initial bank angle
else %This is the targeting routine that is called everytime except first
    GOTO=SELECTOR;
    [THETA, azimuth] = sphereping(long,lat,lon_site*d2r,lat_site*d2r);
    LATANG=asin(sin(THETA)*sin(psi-azimuth));
end

%TARGETING
VSQ=V^2/VSAT^2;     %normalized square of velocity (none)
LEQ=(VSQ-1)*GS;     %gravity plus centripetal acceleration
RDOT=V*sin(gamma);  %altitude rate (ft/s)


%This is an adaptation to determine if vehicle has past landing site
if abs(azimuth-psi)>pi/2 %Landing site is passed
    GONEBY=1;
    %     RTOGO=-THETA;
else %not yet passed
    GONEBY=0;
    %     RTOGO=THETA;
end

runguidance=1;

while runguidance
    switch(GOTO)
        case ('PREDICT3')
            if V-VQUIT>0%not yet to end of guidance
                %this matrix is the reference trajectory
                inter=[994 -690.9 41.15 .002507 -.03446  3.7    6.10*2;...
                    2103 -719   60    .003582 -.05551 10.4   10.91*2;...
                    3922 -694   81.5  .007039 -.09034 23.6   21.64*2;...
                    6295 -609   93.9  .01446  -.1410  46.3   48.35*2;...
                    8531 -493   98.5  .02479  -.1978  75.4   93.72*2;...
```

```
            10101 -416  102.3  .03391  -.2372  99.9  141.1*2;...
            14014 -352  118.7  .06139  -.3305 170.9  329.4;...
            15951 -416  125.2  .07683  -.7500 210.3  465.5;...
            18357 -566  120.4  .09982  -1.000 266.8  682.7;...
            20829 -781   95.4  .1335   -1.000 344.3  980.5;...
            23090 -927   28.1  .2175   -2.021 504.8 1385;...
            23500 -820    6.4  .3046   -3.354 643.0 1508;...
            35000 -820    6.4  .3046   -3.354 794.3 1508];
    %this is the interpolation scheme on the reference traj.
        for k=1:12 %index runs until 12 since there are 13
            if inter(k,1) <=V && V<=inter(k+1,1)
                break
            end
        end
        grad=(V-inter(k,1))/(inter(k+1,1)-inter(k,1));
        RDOTREFV=inter(k,2)+grad*(inter(k+1,2)-inter(k,2));
        DREFRV=inter(k,3)+grad*(inter(k+1,3)-inter(k,3));
        F2V=inter(k,4)+grad*(inter(k+1,4)-inter(k,4));
        F1V=inter(k,5)+grad*(inter(k+1,5)-inter(k,5));
        RTOGOV=inter(k,6)+grad*(inter(k+1,6)-inter(k,6));
        F3V=inter(k,7)+grad*(inter(k+1,7)-inter(k,7));
        %compute ranging potential
         PREDANGL=RTOGOV+F2V*(RDOT-RDOTREFV)+F1V*(D-DREFRV);
        if GONEPAST==0
            if GONEBY==0
                %DNRNGERR=PREDANGL-THETA*ATK; %a display quantity
                %compute vehicle L/D to fly to target
                LOVERD=LOD+4*(ATK*THETA-PREDANGL)/F3V;
            else
                GONEPAST=1;
                %DNRNGERR=MAXRNG;
                LOVERD=-LAD; %landing site passed, full lift down
            end
        else
            LOVERD=-LAD;%landing site already passed,full lift down
        end
        GOTO='GLIMITER';
    %velocity is small, command neutral bank for end of
    %guidance
    else
        ROLLC=0;
        GOTO='310';
    end


case ('GLIMITER')
    if GMAX/2-D>0 %g-load is less than half of max, fly command
        GOTO='310';
    else
        if GMAX-D>0 %g-load is between one half and max g-load
            X=sqrt(2*HS*(GMAX-D)*(LEQ/GMAX+LAD)+(2*HS*GMAX/V)^2);
            if RDOT+X>0 %predicted g-load will be ok, fly command
                GOTO='310';
            else %g-load in danger of being exceeded, fly lift up
                'g load nearly exceeded';
                LOVERD=LAD;
```

```
                GOTO='310';
            end
        else %g-load is exceeded, fly lift up.
            'g-load definitely exceeded';
            LOVERD=LAD;
            GOTO='310';
        end
    end
 case ('310') %lateral logic
    LOVERD1=LOVERD;
    if GONEPAST==0;
        %an error is noted by Bairstow here in the references above
        Y=KLAT*VSQ+LATBIAS/ATK;
        if abs(LOVERD)-LOVERDCMINR<0
            %determine if crossrange is outside envelope
            if K2ROLL*LATANG-Y>0

                if NOSWITCH==1
                    %prohibit roll reversals if drag is large
                    %during upcontrol
                else
                    K2ROLL=-K2ROLL; %reversal bank direction
                end
            end
        else
            %commanded bank angle is within 15deg of lift up,
            %reduce envelope size
            Y=Y/2;
            if K2ROLL*LATANG>0
                if K2ROLL*LATANG-Y>0
                    if NOSWITCH==1
                        %as before, no reversal if drag is large
                        %during upcontrol
                    else
                        K2ROLL=-K2ROLL;
                    end
                end
            else
                %limits bank to within 15 deg of full lift up or
                %down
                LOVERD1=LOVERDCMINR*sign(LOVERD);
            end
        end
    end
    LOVERD1OVERLAD=LOVERD1/LAD; %compute actual vehicle L/D
    %if L/D is bigger than one, make sure it's between +/- 1
    if abs(LOVERD1OVERLAD)-1>0
        LOVERD1OVERLAD=sign(LOVERD);
    end
    %compute roll command for autopilot (rad)
    ROLLC=K2ROLL*acos(LOVERD1OVERLAD);
    NOSWITCH=0; %reset upcontrol reversal prohibition
    GOTO='380'; %end guidance cycle.


case ('380')
```

```
            runguidance=0;
    end
end

selectorout=5;
%Call "autopilot" which rate limits bank angle to reverse in the shortest
%direction
if isempty(Ncall) %first call to limiter.
    Ncall=1;
    flownbank=shortest_bank_limiter(ROLLC,PHIBKRATEMAXDEG,...
        PHIBKACCELMAXDEG,0)*r2d;
else %future calls to bank limiter.
    flownbank=shortest_bank_limiter(ROLLC,PHIBKRATEMAXDEG,...
        PHIBKACCELMAXDEG,2)*r2d;
end
%outputs
varargout={flownbank,ROLLC*r2d,selectorout};
```

BIBLIOGRAPHY

[1] Saradzhyan, Simon, "Space.com – Soyuz Data Recorders Indicate Human Error Not to Blame," URL: http://www.space.com/missionlaunches/exp6_soyuz_030508.html [cited April 28, 2008].

[2] Morring, Frank, Jr. "Aviation Week – NASA Urges Caution on Soyuz Reports," URL: http://www.aviationweek.com/aw/generic/story_channel.jsp?channel=space&id=news/wait042308.xml [cited April 28, 2008].

[3] Malik, Tariq. "'Dramatic' Landing Capped Challenging Spaceflight, Astronaut Says," URL: http://www.space.com/missionlaunches/080423-whitson-landing-comments.html [cited May 22,2008].

[4] Moseley, P. E., "The Apollo Entry Guidance: A Review of the Mathematical Development and its Operational Characteristics," TRW Note No. 69-FMT-791, Dec. 1969, Houston, TX.

[5] Graves, C. A., and Harpold, J.C., "Apollo Experience Report - Mission Planning for Apollo Entry," NASA TN D-6725, Mar. 1972, Houston, TX.

[6] Kuo, Z., and Vinh, N. X., "Improved Matched Asymptotic Solutions for Three-Dimensional Atmospheric Skip Trajectories," *Journal of Spacecraft and Rockets*, Vol. 34, No.4, 1997, pp. 496-502.

[7] Marinsecu, A., and Ilin, S., "Minimum Heat Input Optimal Skip Entry into Venus Atmosphere," *Journal of Spacecraft and Rockets*, Vol. 35, No. 6, 1998, pp 774-777.

[8] Istratie, V., "Optimal Skip Entry into Atmosphere with Minimum Heat and Constraints," AIAA Paper 2000-3993, Aug. 2000.

[9] Bairstow, S. H., "Reentry Guidance with Extended Range Capability for Low L/D Spacecraft," S. M. Thesis, Department of Aeronautics and Astronautics, MIT, Feb. 2006.

[10] Putnam, Z. R., Braun, R. D., Bairstow, S. H., and Barton, G. H., "Improving Lunar Return Entry Footprint Using Enhanced Skip Trajectory Guidance," AIAA Paper 2006-7438, Sept. 2006.

[11] Tigges, M. A., Crull, T., and Rea, J. R., "Numerical Skip-Entry Guidance," AAS Paper 07-076, Feb. 2007

[12] Morth, R., "Reentry Guidance for Apollo," MIT/IL R-532 Vol. I, 1966.

[13] Harpold, J.C., Graves, C. A., "Shuttle Entry Guidance," *The Journal of the Astronautical Sciences*, Vol. 28, No. 3, 1979, pp 239-268.

[14] Carman, G. L., Ives, D. G. and Geller, D. K., "Apollo-Derived Mars Precision Lander Guidance," AIAA Paper 98-4570, Aug. 1998.

[15] Bryant, L.E., Tigges, M. A., and Ives, D. G., "Analytic Drag Control for Precision Landing and Aerocapture," AIAA Paper 98-4572, Aug. 1998.

[16] Lu, W.-M., and Bayard, D. S., "Guidance and Control for Mars Atmospheric Entry: Adaptivity and Robustness," 14th International Federation of Automatic Control Congress, Beijing, China, Jul 5-9, 1999.

[17] Tu, K.-Y., Munir, M. S., Mease, K. D., and Bayard, D. S., "Drag-Based Predictive Tracking Guidance for Mars Precision Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 4, 2000, pp. 620-628.

[18] Roenneke, A.J., Well, K.H., "Nonlinear drag-tracking control applied to optimal low-lift reentry guidance," AIAA Paper 96-3698, July 1996.

[19] Hanson, J. M., "Combining Propulsive and Aerodynamic Maneuvers to Achieve Optimal Orbital Transfer," *Journal of Guidance*, Vol. 12, No. 5, 1988, pp. 732-738.

[20] Schöttle, U. M., Burkhardt, J., and Zimmermann, F., "Optimal Flight Control of a Reentry Capsule with Consideration of Mission Constraints," AIAA Paper 97-3659, 1997.

[21] Gamble, J. D., Cerimele, C. J., Moore, T. E., and Higgins, J., "Atmospheric Guidance Concepts for an Aeroassisted Flight Experiment," *The Journal of the Astronautical Sciences*, Vol. 36, No. 1, 1988, pp. 45-71.

[22] Powell, R. W., and Braun, R. D., "Six-Degree-of-Freedom Guidance and Control Analysis of Mars Aerocapture," *Journal of Guidance Control and Dynamics*, Vol. 16, No. 6, 1993, pp. 1038-1044.

[23] Evans, S. W., and Dukeman, G. A., "Examination of a Practical Aerobraking Guidance Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 3, 1995, pp. 471-477.

[24] Lu, P., "Predictor-Corrector Entry Guidance for Low Lifting Vehicles," *Journal of Guidance, Control, and Dynamics*, accepted for publication, Jan 2008.

[25] Joshi, A., Sivan, K., and Amma, S. S., "Predictor-Corrector Reentry Guidance Algorithm with Path Constraints for Atmospheric Entry Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1307-1318.

[26] Kaluzhskikh, Y.N., Sikharulidze, Y.G., "A Control Algorithm for Reentry of a Rescue Space Vehicle into the Earth's Atmosphere," *Cosmic Research*, Vol. 38, No. 3, 2000, pp. 262-269.

[27] Leavitt, J.A., Mease, K.D., "Feasible Trajectory Generation for Atmospheric Entry Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 2, 2007, pp. 473-481.

[28] Vinh, N. X., *Optimal Trajectories in Atmospheric Flight*, Elsevier Scientific Publishing Company, New York, 1981, pp. 58, 60.

[29] US Standard Atmosphere, US Government Printing Office, Washington DC, 1976.

[30] Levine, G.M., "Apollo Guidance, Navigation, and Control," MIT-Charles Stark Draper Laboratory. R-577, Guidance System Operations Plan for Manned CM Earth Orbital and Lunar Missions using Program Colossus 3, Section 5, Guidance Equations (Rev 15) January 1972, Cambridge, Massachusetts.

[31] Nocedal, J., and Wright, S.J., *Numerical Optimization*, 2nd ed., Springer, New York, 2006, pp. 30, 46.

[32] Rea, J. R., and Putnam, Z. R., "A Comparison of Two Orion Skip Entry Guidance Algorithms," AIAA Paper 2007-6424, Aug. 2007.

[33] Zarchan, P., and Musoff, H., *Fundamentals of Kalman Filtering: A Practical Approach*, 2nd ed., AIAA Progress in Astronautics and Aeronautics, Vol. 208, Reston, Virginia, 2005, p. 647.

[34] D'Souza, C., Crain, T., Clark, F. D., and Getchius, J., "Orion Cislunar Guidance and Navigation," AIAA Paper 2007-6681, Aug. 2007.

[35] Findlay, J.T., Kelly, G.M., McConnell, J.G., and Compton, H.R., "Shuttle 'Challenger' Aerodynamic Performance from Flight Data Comparisons with Predicted Values and 'Columbia' Experience," AIAA Paper 84-0485, AIAA 22nd Aerospace Sciences Meeting, Reno, Nevada, January 1984.

[36] Justus, C. G., and Johnson, J. L. "The NASA/MSFC Global Reference Atmospheric Model – 1999 Version (GRAM-99)," NASA/TM-1999-209630, 1999.

[37] Tigges, M.A., Crull, T., Rea, J., Johnson, W., "Numerical Skip-Entry Guidance," AAS 07-076, 2007.