Entry, Descent and Landing Vehicle Design Space Exploration for Crewed Mars Missions

by

Zahra Khan

B. Eng. Aerospace Engineering
Carleton University, 2005

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MAY 2008

Signature of Author……..
Zahra Khan
Department of Aeronautics and Astronautics
May 2008

Certified by……
Jeffrey Hoffman
Professor of the Practice
Aeronautics and Astronautics
Thesis Supervisor

Accepted by……
Prof. David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

Entry, Descent and Landing Vehicle Design Space Exploration for Crewed Mars Missions

by

Zahra Khan

Submitted to the Department of Aeronautics and Astronautics
on May 23, 2008 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Aeronautics and Astronautics
At the Massachusetts Institute of Technology

Abstract

With the announcement of the Vision for Space Exploration in 2004, NASA has been preparing plans for a crewed mission to Mars in the next few decades. One challenge associated with crewed missions to the Martian surface is the comparatively large mass of planned surface elements which will require an increase of at least an order of magnitude in the landed mass performance of entry, descent and landing (EDL) vehicles compared to previous successful robotic missions and ones currently planned for the near future. The work presented in this thesis attempts to look at the effect of individual vehicle and mission design parameters on the overall EDL system landed mass performance. For this purpose, a planetary EDL simulation and sensitivity analysis tool has been created which allows for the variation of a number of relevant parameters and provides the user with performance data. To date, the tool has been validated with experimental and simulated data for Mars missions. The tool assumes the use of propulsive means for final descent. Using this tool, a number of sensitivity analyses for different parameters for both the entry phase and the entire EDL profile have been conducted. Major insights gleaned from these include the need for research into propulsive descent safety requirements to prevent over-conservatism in establishing the propulsive descent initiation (PDI) altitude since a high PDI altitude can significantly lower EDL vehicle landed mass performance. Another area of research with the potential to provide significant improvements in performance is structural and thermal technology that reduces the aeroshell mass of the vehicle. Finally, the landed mass performance of a reference EDL vehicle with a spherical forebody and a conical aftbody was also evaluated. For an entry mass of 62.5 mT (corresponding to a two Ares V launches), a moderate aeroshell mass fraction and a low-lying landing site, a significant payload of around 26 metric tons can be delivered to the Martian surface with two Ares V launches. This indicates that, from a landed mass perspective, a crewed Mars mission is feasible using the moderate lift reference vehicle, since the minimum landed mass requirements for such a mission are on the order of 20 to 30 metric tons.

Thesis Supervisor: Jeffrey Hoffman
Title: Professor of the Practice, Aeronautics and Astronautics

Acknowledgements

"In the name of God, Most Gracious, Most Merciful"

It's been a busy, rushed and amazing two years and I'd like to thank all those who helped me through my time at MIT. First, thank you to God and mom for listening and supporting me through numerous rough patches. To the folks at MIT Medical for trying so hard to keep me healthy and mostly succeeding. You Rock. To my advisors Prof. Crawley and Prof. Hoffman for their advice, understanding and support. To those who believed in me enough to fund me: MIT Department of Aeronautics and Astronautics, Natural Sciences and Engineering Research Council (NSERC) of Canada, J. Armand Bombardier Internationalist Foundation and Prof. Crawley/NASA. To Ms. Barbara Lechner for listening and advising me on everything from initiating student projects to finding a job. To Dr. Juan Cruz and Dr. Bandu Pamadi of NASA Langley for providing me information and help with some of the more challenging aspects of this work. To my labmate Wilfried Hofstetter for painstakingly proofreading this thesis and excellent technical advice throughout this work. To Phillip Cunio, labmate and friend, for great conversations and his ready willingness to join me in various quirky pursuits. And to the MIT community for providing a wonderful array of learning opportunities and distractions throughout my time here. Thank you all.

Table of Contents

List of Figures

List of Tables

Nomenclature

A	Vehicle frontal area (aerodynamic reference area)
AMF	Aeroshell Mass Fraction
C_d	Drag coefficient
D	Drag
DSMF	Descent stage Structural Mass Fraction
EDL	Entry, Descent and Landing
G	Universal Gravitational Constant
h	Altitude above mean planet radius
HEO	High Earth Orbit
L	Lift
L/D	Lift-to-Drag ratio
LEO	Low Earth Orbit
m_{entry}	Vehicle entry mass
M_{planet}	Mass of planet
MIT	Massachusetts Institute of Technology
mT	Metric tons
NASA	National Aeronautics and Space Administration
PDI	Propulsive Descent Initiation
r	Magnitude of vehicle position vector
R_{planet}	Mean planet radius
s	Vehicle range from entry point
TMI	Trans-Mars Injection
V	Vehicle velocity
V_x	x-component of vehicle velocity
V_y	y-component of vehicle velocity
x	x-component of vehicle position vector
y	y-component of vehicle position vector
γ	Flight path angle
θ	Angle subtended at center of planet by entry arc
ρ	Atmospheric density
ω_{planet}	Planet sidereal rotation speed

1.0 Introduction

With the announcement of the Vision for Space Exploration in 2004 (1), NASA has been preparing plans for a crewed mission to Mars in the next few decades. Although NASA has successfully landed various robotic craft on the surface of Mars, a crewed mission entails several new challenges. One of these is the size of the landing craft. The largest landers so far have been the Viking landers with a mass of about 600 kg (2). All spacecraft which have to date landed successfully on Mars have used a combination of blunt body aeroshell/heatshield shapes and parachutes for deceleration purposes, with propulsive means used only for final descent/landing. The 2000 kg MSL lander will push these technologies further in terms of landed mass performance (3). However, a crewed mission requires at least an order of magnitude increase in landed mass performance, given that current designs for a crewed mission to Mars include individual surface elements having masses estimated at 19 mT to 30 mT (4) (5) (6) [*]. It is unclear whether parachute technology can be improved to accommodate this new level of performance. Scaling of current parachute technology leads to very large diameter parachutes with a relatively low Mach number operating limit and long deployment times (3). For safety reasons, it is therefore preferable to deploy parachutes at higher Mach numbers. However, there appears to be a physical limit on the use of parachutes at higher Mach numbers in the Martian atmosphere i.e. at Mach numbers above about 3, parachutes are predicted to undergo violent unstable oscillations (7) (8).

This thesis aims to explore the relationship between various design factors and the landed mass performance of an entry, descent and landing vehicle for crewed Martian exploration using *robust technologies* for deceleration purposes. Factors considered consist of both vehicle and mission design parameters including aerodynamic characteristics, structural design factors, and landing site elevation. The design of the EDL system can have a major impact on the overall system architecture for a crewed Mars mission since EDL represents a crucial link in the transportation chain to the Martian surface. On one side of this link, constraints can be placed on the EDL system due to mass and volume capabilities of the vehicle delivering the EDL system into Martian orbit. On the other side, the landed mass performance of the EDL system may constrain the mass of the surface exploration elements. Therefore, with concept development underway for crewed Mars missions, it is essential to understand the performance limitations of the EDL system.

Much of the inspiration for this work derives from the NASA Concept Exploration and Refinement (CE&R) (9) study conducted jointly by Draper Laboratory and MIT, which explored mission architecture choices for a crewed Mars mission in parallel with the Exploration Systems Architecture Study (10). The CE&R study, along with other work by Dr. Robert Braun (of the Georgia Institute of Technology), one of the study participants, highlighted the challenges associated with landing crewed missions on Mars and provided some initial concepts for resolving these challenges.

The Draper/MIT CE&R study as well as other papers have presented parametric results for altitude reached by various entry bodies at specific Mach numbers. Using this data, the feasibility of

[*] This mass range is for elements that need to be delivered to the surface fully assembled e.g. surface habitat or a high-risk system such as the Mars Ascent Vehicle (MAV). Thus, they represent minimum requirements for landed mass performance on Mars. While other surface systems may have greater masses overall, there is a possibility to assemble them on surface. However, due to the risk associated with assembling e.g. the MAV system, it is highly preferable to land this fully assembled. The numbers presented have been taken from different studies as referenced.

using parachutes for further deceleration was explored, with the conclusion that parachute-only descent is infeasible for entry masses required for human missions due to the large parachute sizes required. Therefore, modeling of propulsive descent was also performed to investigate delta-v requirements and the effect of propulsive descent initiation (PDI) speed on these requirements. It was concluded that initiation of propulsive descent at higher Mach numbers results in greater gravity losses and thus, higher delta-v is required. Some effects of propulsive descent system design on the overall mass performance of the vehicle were also analyzed. Suggestions for improvement of performance include development of supersonic decelerators, large diameter aeroshells and supersonic propulsion solutions (3) (11). However, for the most part, essentially a single geometry was investigated with varying L/D and the effect of drag coefficient was not clearly investigated in the studies. Additionally, many of the results for the subsystems such as aerodynamic entry and propulsive descent were presented separately. This makes it difficult to judge the effect of the various parameters on the overall performance (11) (3).

The work presented in this thesis aims to bridge the gap between the effect of design parameters on EDL subsystems which has been investigated in previous works and the overall EDL system performance.

In order to conduct parametric analysis for the EDL system, numerical trajectory simulation is required due to the high variability of atmospheric and gravitational forces with altitude. Several EDL trajectory simulation codes are available in the aerospace community including POST (12) which is used by NASA and ASTOS (13) used by ESA. However, several issues exist with these codes. The first main issue is public availability and the second is complexity. The author had attempted to use the unrestricted ASTOS code at the beginning of this study with the realization that the amount of information provided by this tool is not necessary for a high-level design study and the programming effort required does not facilitate large scale sensitivity analyses.

Therefore, the first part of this thesis deals with developing an EDL trajectory simulation tool that allows for rapidly conducting high level sensitivity analyses and easy visualization of the results for an EDL system using the heatshield/aeroshell and propulsive means for deceleration. Apart from propulsion system sizing, automated optimization of the landing trajectory as well as event initiation points has not been attempted to date in this work. One reason for this is to allow for better graphical visualization of the contribution of each parameter to the overall performance rather than embedding important information in an optimization routine.

The second part of this thesis is devoted to using the tool developed to actually conduct sensitivity studies and analyze the results. The goal is to provide insights into the sensitivity of an EDL vehicle design to various vehicle and mission design parameters as well as to help define the boundaries of the design envelope for crewed EDL vehicles.

It is envisioned that the results of this effort will contribute to crewed Mars mission architecture analysis by possibly placing constraints on Martian surface element sizes as well as by identifying key technologies that need to be developed and key aspects of vehicle and mission design that need to be investigated to extend the capability of EDL vehicles.

2.0 Design Space Exploration Tool

The purpose of the EDL Design Space Exploration Tool developed for this project is to allow for easily conducting and visualizing performance and sensitivity analyses using various parameters including aerodynamic characteristics, vehicle mass properties, entry conditions and descent transition conditions. The tool, therefore, has two essential components: a trajectory simulation routine and a run setup and visualization routine. The latter uses the former to conduct sensitivity analyses. This section describes the physics embodied in the trajectory simulation routine as well as the overall features and limitations of the tool.

2.1 Model Development and Description

The trajectory is divided into two phases: aerodynamic entry and propulsive descent. The limitations of including only these two phases of flight are discussed in Section 2.2. Model development and description for each of the phases can be found in the following sections. Additionally, the propulsive descent part of the trajectory simulation also contains an internal optimization scheme to size the descent propulsion system.

2.1.1 Assumptions

Both the aerodynamic entry and propulsive descent portions of the trajectory were simulated using a simple force balance integration scheme. This model embodies several assumptions as follows:

- The vehicle enters from orbit rather than direct entry. Therefore, entry conditions are specified as the apoapsis and periapsis of the entering orbit.
- The vehicle maintains a constant angle of attack throughout the trajectory. The angle of attack of the vehicle is defined as the angle between the vehicle's longitudinal axis and the flight direction. A particular angle of attack is modeled by the combination of drag coefficient and lift over drag ratio achieved at that angle.
- The trajectory is modeled in a reference frame that is pinned to the center of the planet and rotates with the planet.

Additionally, for all analyses presented in this thesis, the vehicle enters into the atmosphere in the direction of the sidereal rotation of the planet. Entering with the rotation is advantageous because it reduces the relative velocity of the vehicle with respect to the planetary surface.

2.1.2 Inputs

This model allows the user to control several different parameters. These are requested as inputs at the beginning of the simulation. A listing is provided in Table 1. The user may choose two of these parameters as variables for conducting a sensitivity analysis. The user must also specify the direction in which the vehicle enters into the atmosphere i.e. in the direction or opposite to the direction of sidereal rotation of the planet.

Table 1: Simulation Parameters - Vehicle and Mission

Parameter Name	Abbreviation*	Unit
Drag Coefficient	Cd	None
Lift-to-Drag Ratio	L/D	None
Vehicle Base Diameter	Dia	m
Vehicle Entry Mass	Entry Mass	mT (metric tons)
Aeroshell Mass Fraction	AMF	None
Descent Propulsion Stage Structural Mass Fraction	DSMF	None
Landing Altitude	Landing Alt	km
Propulsive Descent Initiation (PDI) Altitude (Above	PDalt	km
Entry Orbit Apoapsis Altitude	Orbit Apo	km
Entry Orbit Periapsis Altitude	Orbit Peri	Km
Powered Descent Drag Coefficient (Drag Coefficient after Aeroshell Jettison)	Descent Cd	None
Powered Descent Lift-to-Drag Ratio	Descent L/D	None
Powered Descent Engine Isp	Engine Isp	S

*These abbreviations are used to indicate the parameter values on sensitivity analysis graphs produced using the tool described in this section.

Two terms in Table 1 need to be defined. These are the two mass fractions. The aeroshell mass fraction is defined by comparison of the aeroshell mass to the rest of the EDL vehicle. For example, an aeroshell mass fraction of 1 means that the aeroshell has the same mass as the rest of the EDL vehicle i.e. it is 50% of the total mass of the EDL vehicle.

$$\boldsymbol{Aeroshell\ Mass\ Fraction} = \frac{\boldsymbol{Aeroshell\ Mass}}{\boldsymbol{Total\ EDL\ Vehicle\ Mass - Aeroshell\ Mass}} \quad \textbf{Eq 1}$$

The descent propulsion stage structural mass fraction is defined by comparison of the propellant mass to the structural mass of the descent stage.

$$\boldsymbol{Descent\ Propulsion\ Stage\ Structural\ Mass\ Fraction} = \frac{\boldsymbol{Descent\ Stage\ Structural\ Mass}}{\boldsymbol{Propellant\ Mass}}$$

Eq 2

Another point that needs to be noted is that the Propulsive Descent Initiation Altitude is measured from the landing site or landing altitude i.e. it is Above Ground Level (AGL) whereas all other altitudes are measured from the mean planet radius.

Apart from the vehicle and mission parameters specified, the user must supply planet-specific information including constants listed in Table 2 as well as an atmospheric model. In the current version of the tool, the default is set for the planet Mars. However, the option for changing this is provided in

order to increase the tool's utility. The atmospheric model must be able to output density and temperature at the specified altitude.

Table 2: Planetary Constants used in Simulation

Planet Constant	Unit
Mass	kg
Mean Radius	km
Ratio of Specific Heats of Atmosphere	None
Gas Constant for Atmosphere	J/kg/K
Atmosphere Interface Altitude	km
Sidereal Rotation Period	hr
Parachute Operation Limiting Mach Number	None

2.1.3 Entry Conditions

Before beginning the EDL simulation, the entry conditions at the atmospheric interface are determined using the inputs provided and the planetary constants. The entry conditions are calculated as an entry velocity and entry angle using the following equations (14), where a is the semimajor axis of the entry orbit and H is the angular momentum.

$$a = \frac{r_{apoapsis} + r_{periapsis}}{2} \quad \text{Eq 3}$$

$$V_{entry} = \sqrt{G * M_{Planet} * \left(\frac{2}{r_{AtmosphericInterface}} - \frac{1}{a}\right)} \quad \text{Eq 4}$$

$$V_{apoapsis} = \sqrt{G * M_{Planet} * \left(\frac{2}{r_{apoapsis}} - \frac{1}{a}\right)} \quad \text{Eq 5}$$

$$H = V_{apoapsis} * r_{apoapsis} \quad \text{Eq 6}$$

$$\gamma_{entry} = -\cos^{-1}\left(\frac{H}{V_{entry} * r_{AtmosphericInterface}}\right) \quad \text{Eq 7}$$

2.1.4 Aerodynamic Entry

The aerodynamic entry part of the trajectory is simulated using a force balance integration. The reference frame and force balance diagram as well as the equations of motion are presented below. Table 3 provides a list of symbols and their definitions. These are also presented in the Nomenclature but are included here for quick reference.

Table 3: Definition of EDL simulation quantities

Symbol	Quantity	Unit
L	Lift	N
D	Drag	N
ρ	Atmospheric density	kg/m^3
h	Altitude above mean planet radius	m/km
V	Vehicle velocity	m/s
A	Vehicle frontal area (aerodynamic reference area)	m^2
C_d	Drag coefficient	None
L/D	Lift-to-Drag ratio	None
G	Universal Gravitational Constant	$m^3/(kg.s^2)$
M_{planet}	Mass of planet	kg
r	Magnitude of vehicle position vector	m
γ	Flight path angle	deg/rad
V_x	x-component of vehicle velocity	m/s
V_y	y-component of vehicle velocity	m/s
ω_{planet}	Planet sidereal rotation speed	rad/s
x	x-component of vehicle position vector	m
y	y-component of vehicle position vector	m
m_{entry}	Vehicle entry mass	kg
θ	Angle subtended by entry arc at planet center	deg/rad
R_{planet}	Mean planet radius	m/km
s	Vehicle range from entry point	m/km

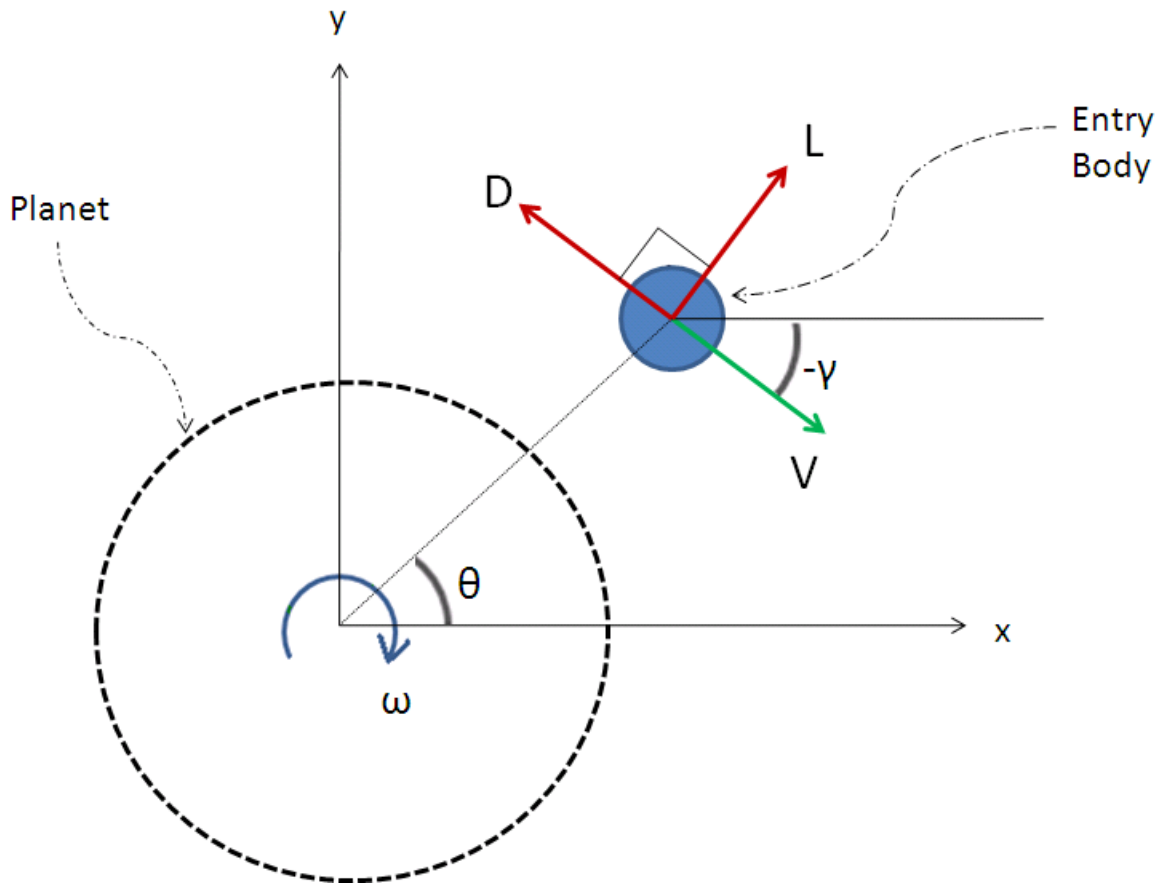

Figure 1: Entry Body Reference Frame and Force Balance

$$drag\ (t) = \frac{1}{2}\rho_{Planet}\big(h(t)\big)V(t)^2AC_d \quad \textbf{Eq 8}$$

$$lift\ (t) = \frac{1}{2}\rho_{Planet}\big(h(t)\big)V(t)^2AC_d(L/D) \quad \textbf{Eq 9}$$

$$gravity(t) = \frac{G*M_{Planet}}{r(t)^2} \quad \textbf{Eq 10}$$

$$\frac{dV_x}{dt} = lift(t) * \frac{\cos(\gamma(t)+{}^{\pi}/_{2})}{m_{entry}} + drag(t) * \frac{\cos(\gamma(t)+\pi)}{m_{entry}} - \frac{gravity(t)*x(t)}{r(t)} + 2 * \omega_{Planet} * V(t)_y + \omega_{Planet}{}^2 * x(t) \quad \textbf{Eq 11}$$

$$\frac{dV_y}{dt} = lift(t) * \frac{\sin(\gamma(t)+{}^{\pi}/_{2})}{m_{entry}} + drag(t) * \frac{\sin(\gamma(t)+\pi)}{m_{entry}} - \frac{gravity(t)*y(t)}{r(t)} - 2 * \omega_{Planet} * V(t)_x + \omega_{Planet}{}^2 * y(t) \quad \textbf{Eq 12}$$

$$V_x(t+1) = V_x(t) + \frac{dV_x}{dt} * \Delta t \quad \textbf{Eq 13}$$

$$V_y(t+1) = V_y(t) + \frac{dV_y}{dt} * \Delta t \quad \textbf{Eq 14}$$

$$V(t) = \sqrt{V_x(t)^2 + V_y(t)^2} \quad \textbf{Eq 15}$$

$$x(t+1) = x(t) + V_x * \Delta t \quad \textbf{Eq 16}$$

$$y(t+1) = y(t) + V_y * \Delta t \quad \textbf{Eq 17}$$

$$r(t) = \sqrt{x(t)^2 + y(t)^2} \quad \textbf{Eq 18}$$

$$h(t) = r(t) - R_{Planet} \quad \textbf{Eq 19}$$

$$\theta(t) = \arctan\left(\frac{y(t)}{x(t)}\right) \quad \textbf{Eq 20}$$

$$s(t) = R_{Planet} * \left(\pi/2 - \theta(t)\right) \quad \textbf{Eq 21}$$

2.1.5 Propulsive Descent

The basic equations for the propulsive descent part of the EDL trajectory are the same as for aerodynamic entry with the addition of thrust in the direction of drag; this results in a gravity turn trajectory. Additionally, the mass is adjusted at each time step to account for the propellant burn. The aerodynamic coefficients used for this part of the trajectory are those for a flat plate, as this is how the vehicle is "seen" by the flow in the descent configuration.

The initial conditions for this part of the trajectory are obtained using the transition logic described in Section 2.1.6. In addition to the state variables, the mass at the start of the descent is obtained by subtracting the aeroshell mass from the total EDL vehicle mass, thus simulating an aeroshell jettison. The user also has the option of specifying a deployment time for the propulsion system to model engine start time, etc. The code then simulates this as a coast period after aeroshell jettison i.e. the aerodynamics are those for the powered descent configuration but the thrust is zero.

The thrust is kept constant throughout the trajectory. However, the thrust is not a fixed value for all runs. The code incorporates an optimization scheme to size the descent propulsion system for each data point. The algorithm is described in Figure 2.

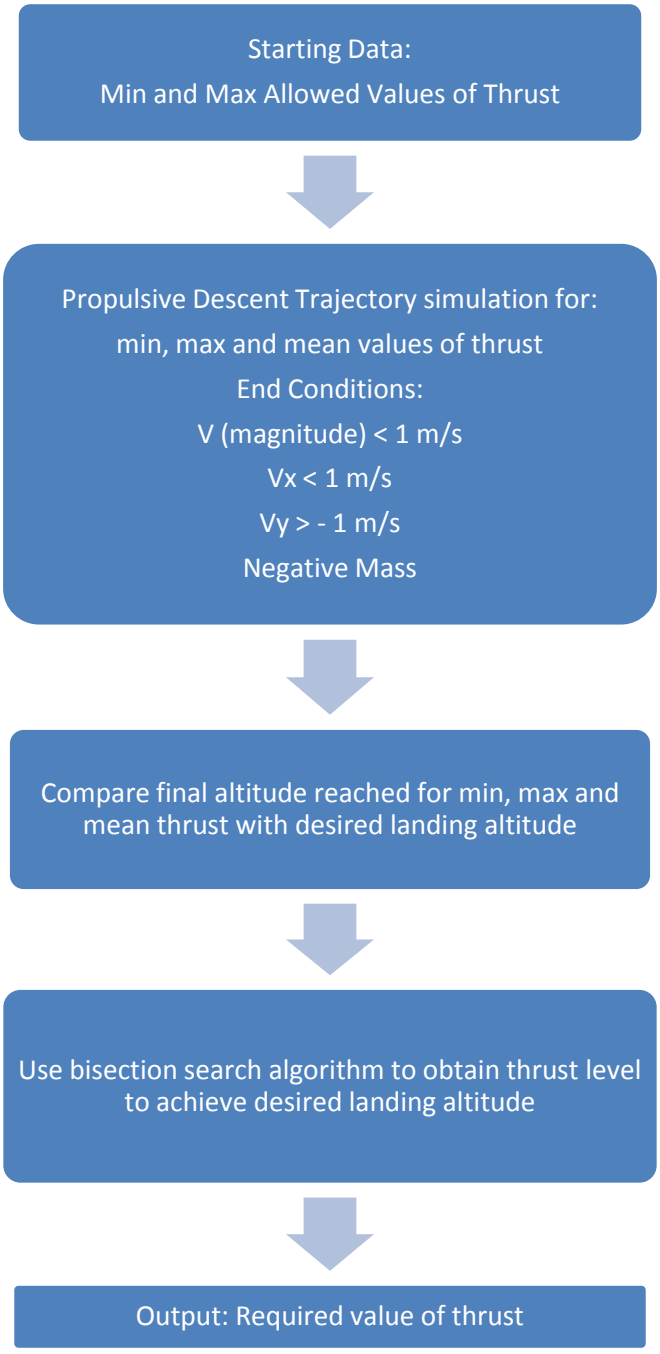

Figure 2: Descent Propulsion System Sizing Algorithm

The actual trajectory is then simulated and stored using the required value of thrust produced by the above algorithm. Note that the desired landing altitude is set by the user.

Two special cases can occur that might corrupt the optimization scheme. The first case is one where a Mach number is used to set the transition but the vehicle does not reach this Mach number

before hitting the surface. In this case, the program is designed to detect this condition and skip the data point. This is done by checking to see that the PDI altitude is above the desired landing altitude. The second case is one in which due to a high thrust level, the propellant burn may cause the mass of the vehicle to go to zero or become negative. If this condition occurs, the optimization iteration stops and returns a final altitude value to the main algorithm such that it biases the search towards lower thrust values. The reason for this is that a lower thrust value would result in lower propellant burn, thus potentially eliminating the negative mass condition. For the data points considered in this study, these fixes provide feasible results.

2.1.6 Descent Transition

One of the challenging aspects of the trajectory simulation part of the problem is the transition to propulsive descent from aerodynamic entry. Several different criteria can be used for determining the timing of the switch. Traditionally, Mach number has been one of the main transition criteria due to performance limitations of parachutes. Although, in this case, propulsive descent is being used, a drogue chute might be necessary for separation of part of the entry body as well as stabilization. For this reason, a parachute operating limit has been incorporated into the model (8). However, the actual transition condition set by the user is a transition altitude together with some associated constraints. The user is provided the option of specifying a maximum Mach number for transition initiation. The logic used to determine the actual transition point is as follows:

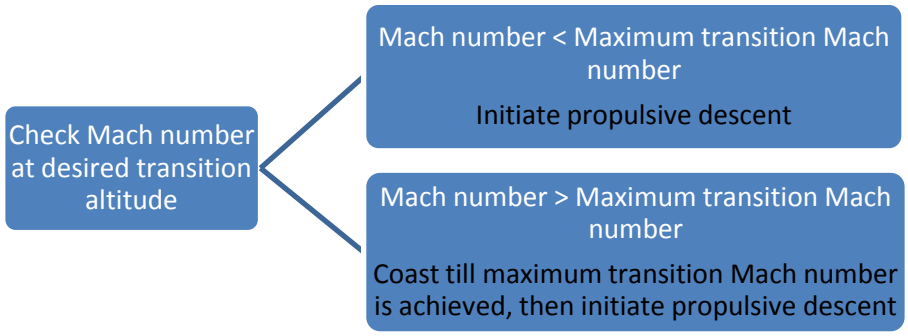

Figure 3: Propulsive Descent Transition Logic

Additionally, there is no limitation in the model for transition occurring at a Mach number higher than the parachute/drogue chute operating limit. In this case, it is assumed that the aeroshell/backshell is separated from the entry body by propulsive means rather than drogue chutes. A mass penalty is incorporated into the model for propulsive separation and can be adjusted by the user. This mass penalty is modeled in the same way as the aeroshell mass fraction (Eq 1).

2.1.7 Final Output Calculations

In addition to the entry and descent trajectory, the model provides other useful outputs such as the landed mass performance characterized by the payload mass fraction which is defined as:

$$\boldsymbol{Payload\ Mass\ Fraction} = \frac{\boldsymbol{Payload\ Mass}}{\boldsymbol{Total\ EDL\ Vehicle\ Mass}} \quad \textbf{Eq 22}$$

The payload mass in the above calculation is found as follows:

$$\boldsymbol{Payload\ mass} = \boldsymbol{Vehicle\ Descent\ Mass} - \boldsymbol{Propellant\ Mass} * (\mathbf{1} + \boldsymbol{Descent\ Propulsion\ Stage\ Structural\ Mass\ Fraction})$$

Eq 23

2.1.8 Information Flow

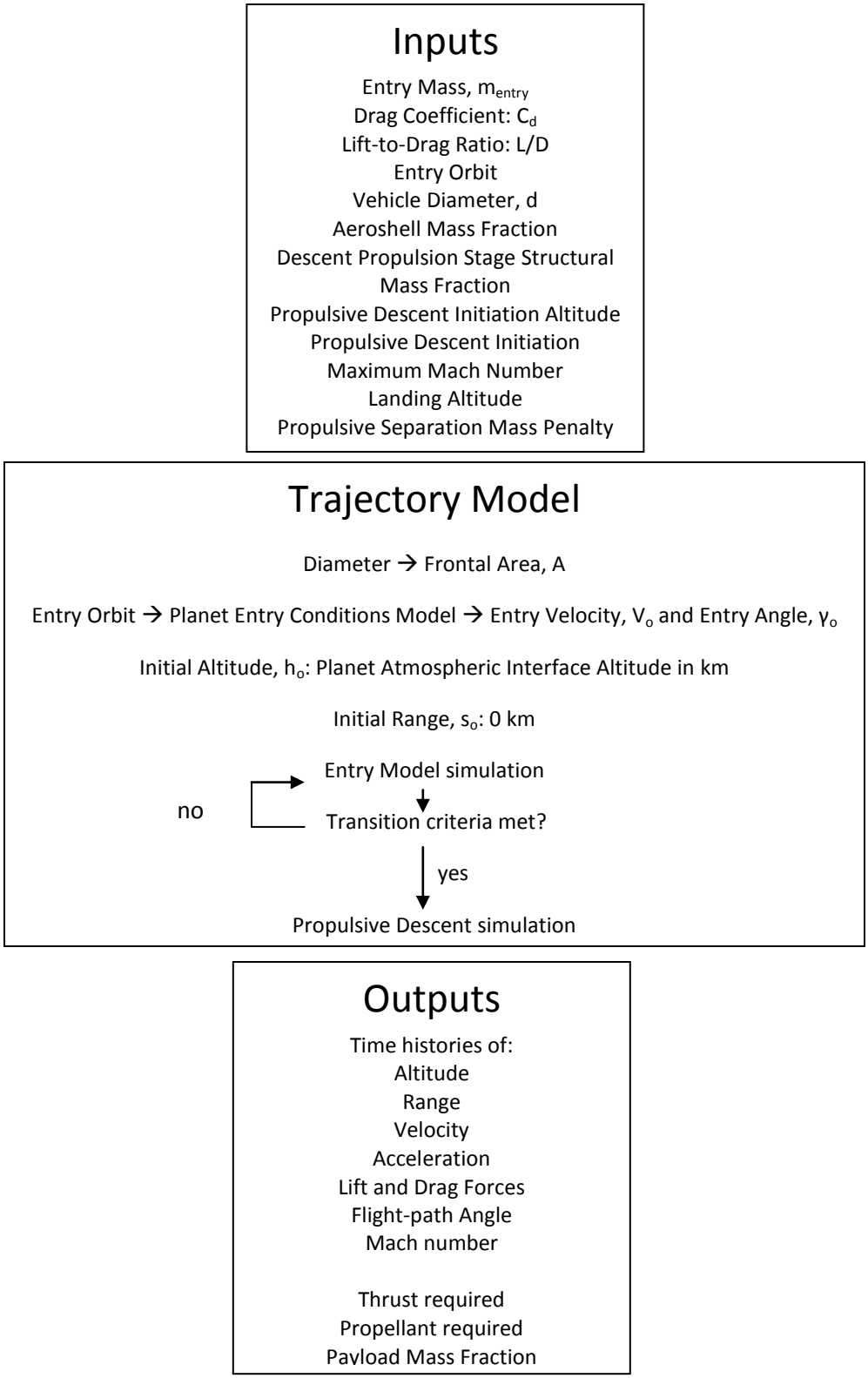

Figure 4: Information Flow Diagram for EDL Simulation

2.2 Features and Limitations

The main features of this model are user control over a variety of vehicle and mission design parameters including entry mass, vehicle aerodynamic characteristics, vehicle diameter, structural mass fractions, landing and propulsive descent initiation altitudes. Although simulation constants and atmospheric conditions have, by default, been set for Mars, it is possible for the user to easily specify these for other planetary bodies through text and function files that "plug-in" to the main program.

The model also has a number of specific limitations. One of the main limitations is that aerodynamic decelerators were not modeled. This is because the focus was on large payloads for which, aside from drogue chutes, aerodynamic decelerators such as parachutes cannot be used due to stability problems at high Mach numbers (8) (7). Additionally, no active optimal control law has been implemented to optimize the entry trajectory. This is deemed acceptable for initial architecture-level analyses, as the addition of a control law should improve the performance of the vehicle. Therefore, an uncontrolled entry essentially represents conservative lower performance limit.

Some limitations were introduced as a result of parametric sensitivity analyses. For example, the drag coefficient and lift-over-drag ratio were varied independently. This is not the case in reality as only discrete combinations of the two can be achieved for a particular shape. Additionally, the aerodynamic characteristics remain constant throughout the flight, which is not the case in reality, as these change with Mach number. Also, a structural scaling relationship was not included i.e. it was assumed the structural mass fractions remain constant regardless of the physical size of the vehicle.

2.3 Validation

Validation is an important step in the development of this tool in order to be able to trust the results obtained from it. Validation for this particular tool is difficult due to various differences between EDL trajectories such as separation events, technologies used and control laws. Therefore, the focus was on validation of the basic physics of entry. For this reason, the aerodynamic entry part of the EDL trajectory, which does not incorporate any active control, was chosen to be compared to other data. This is deemed acceptable as the equations for the powered descent phase are the same as the entry with the addition of retarding thrust.

Even with the simpler case of validation of the aerodynamic entry profile, the analysis can be difficult due to unavailability of good data for comparison. In many cases, the relevant entry conditions are not explicitly specified. Another issue is the different atmospheric models used by different simulations which can have a significant effect on the entry trajectory.

For this work, the simulation results were compared to both experimental results and other simulations, and this analysis is presented in the following subsections.

2.3.1 Comparison with other simulation results

Simulation data for comparison purposes was obtained from lecture notes for a course on Hypersonic EDL (15). The lecture notes detailed the atmospheric model used for the simulations as well as the necessary entry conditions, thus representing a complete data set for comparison. Figure 6 and Figure 7 show the results from the lecture notes for simulations for a range of ballistic coefficient values while Figure 5 and Figure 8 show the results from simulations conducted using the code developed for this study. As can be seen from these graphs, the results from the current work match the reference results very well.

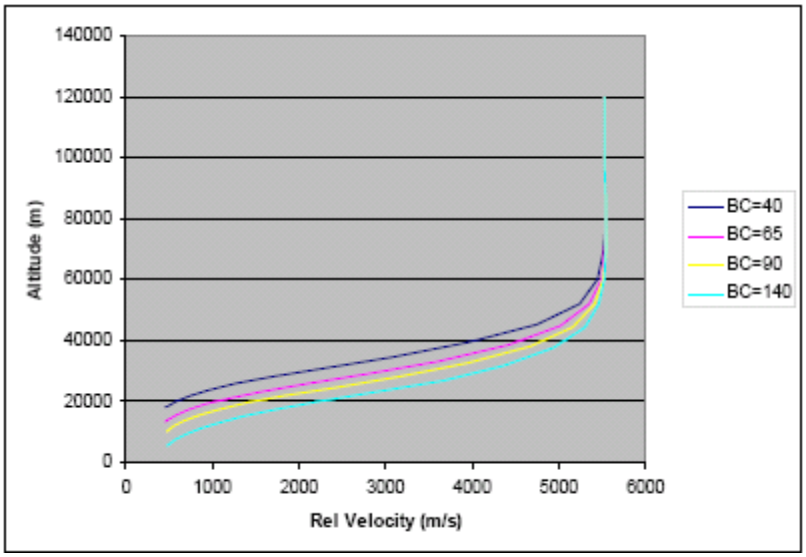

Figure 6: Altitude vs. Relative Velocity Reference Simulation Results (15) (BC = Ballistic Coefficient)

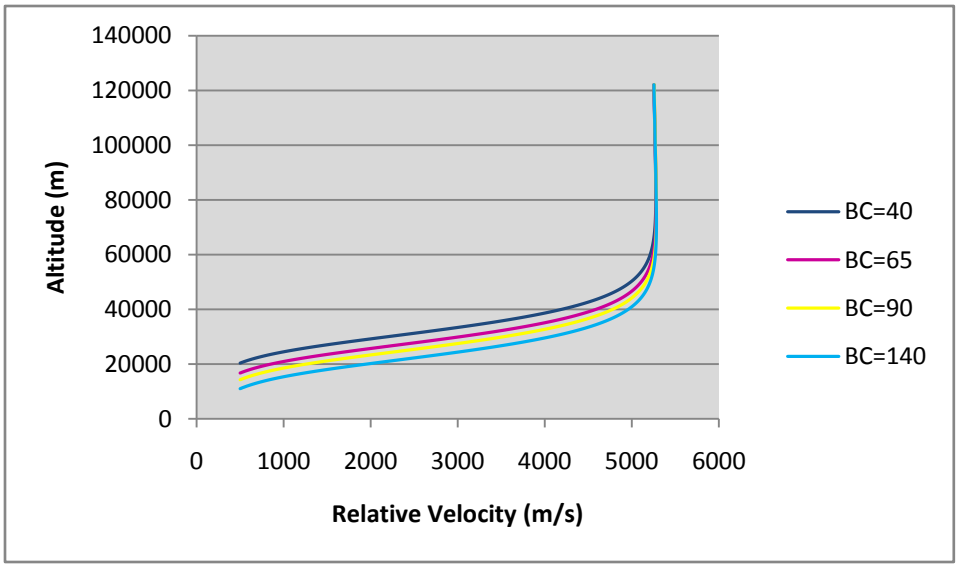

Figure 5: Altitude vs. Relative Velocity Results for Validation of Tool developed in this study (BC = Ballistic Coefficient)

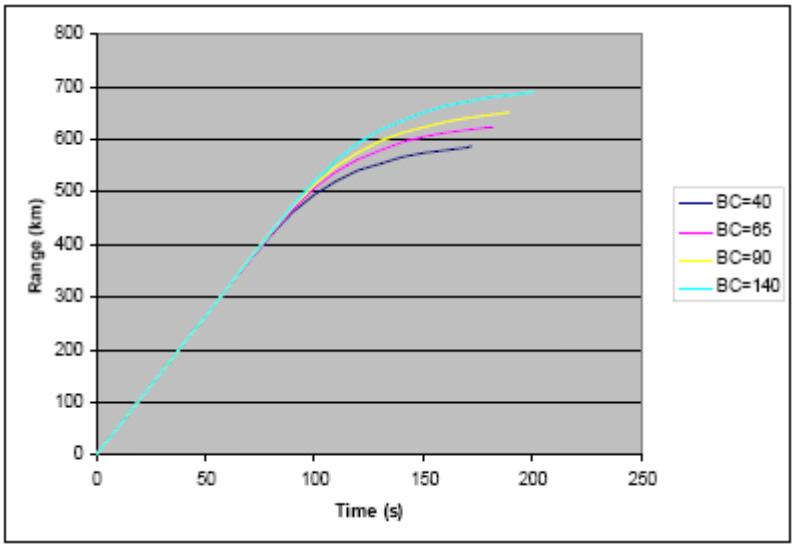

Figure 7: Range Time History Reference Simulation Results (15)
(BC = Ballistic Coefficient)

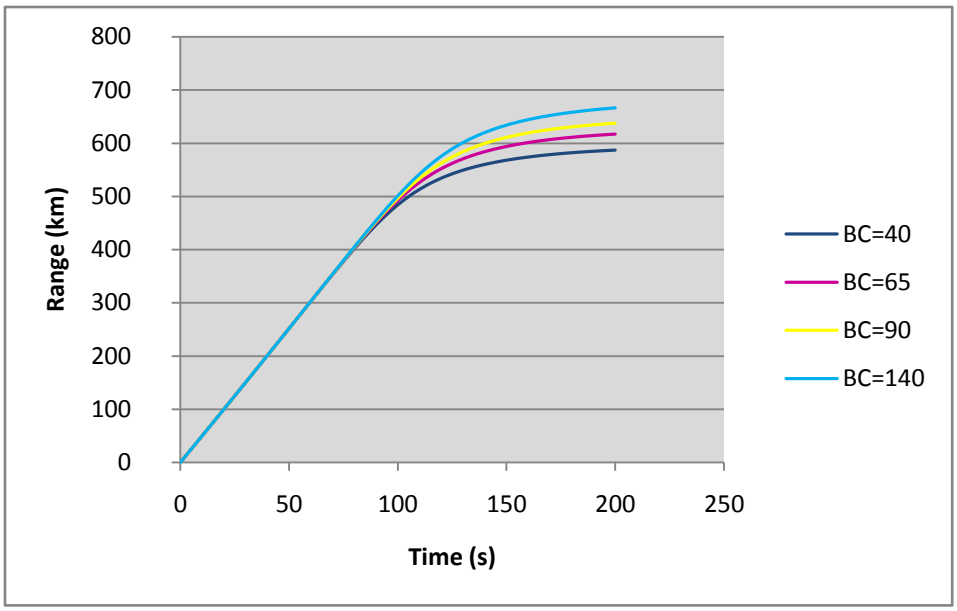

Figure 8: Range Time History Simulation Results for Validation of Tool developed for this study (BC = Ballistic Coefficient)

2.3.2 Comparison with experimental data

As mentioned previously, the atmospheric model used for simulation can have a significant effect on the entry trajectory. For this study, the atmospheric model used for Mars was obtained using data from the Pathfinder mission (16). Therefore, the aerodynamic entry profile from the Pathfinder mission represents an excellent source of validation data. Additionally, for both the aerodynamic entry phase of the Pathfinder mission and the entry simulations in this study, the entry vehicle is not actively controlled.

Figure 10 and Figure 11 present the time histories of various state variables for the Pathfinder EDL mission. Note that this data has been reduced from acceleration data measured by the vehicle. Figure 9 and Figure 12 present the same data obtained using the aerodynamic entry model developed for this project. The cutoff time for these plots corresponds to the end of the "Aeroshell Phase" in the Pathfinder data plots.

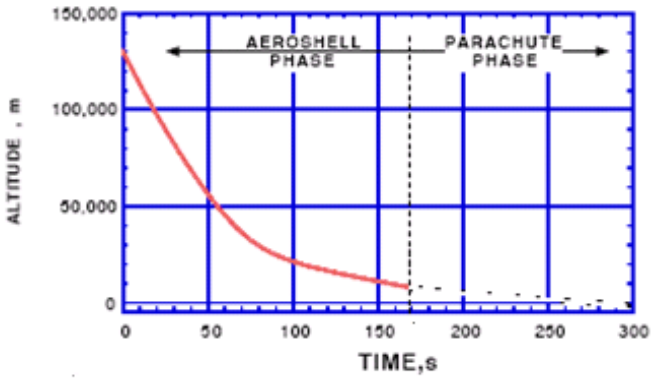

Figure 10: Pathfinder Altitude Time History (17)

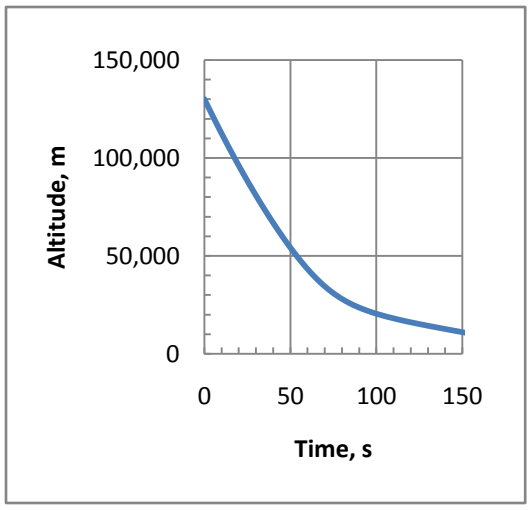

Figure 9: Pathfinder Altitude Simulated Time History

It can be seen from Figure 9, Figure 10, Figure 11 and Figure 12 that the altitude and velocity time histories for the actual and simulated trajectories match very well both in terms of the shapes of the curves as well as the endpoints. In case of the flight path angle, there is a small discrepancy both in the shape of the curve and the endpoint. This can be explained by anomalies in the real Pathfinder trajectory. The mission was designed to fly a ballistic entry with no lift. If this was indeed the case, the flight path angle increase seen in Figure 11 does not seem reasonable. However, the vehicle suffered from some stability problems in the early phase of its flight. Thus, an angle of attack was introduced causing lift and increasing the flight path angle (18). Since the simulated trajectory perfectly maintained a zero lift condition, the same trend was not seen. Also, the increase in flight path angle in the earlier phase of flight resulted in a high flight path angle at the end of 150 seconds as compared to the simulated value.

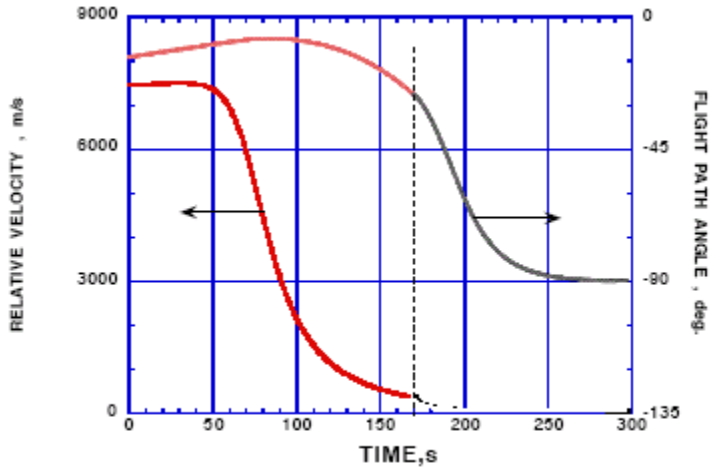

Figure 11: Pathfinder Velocity and Flight Path Angle Time History (17)

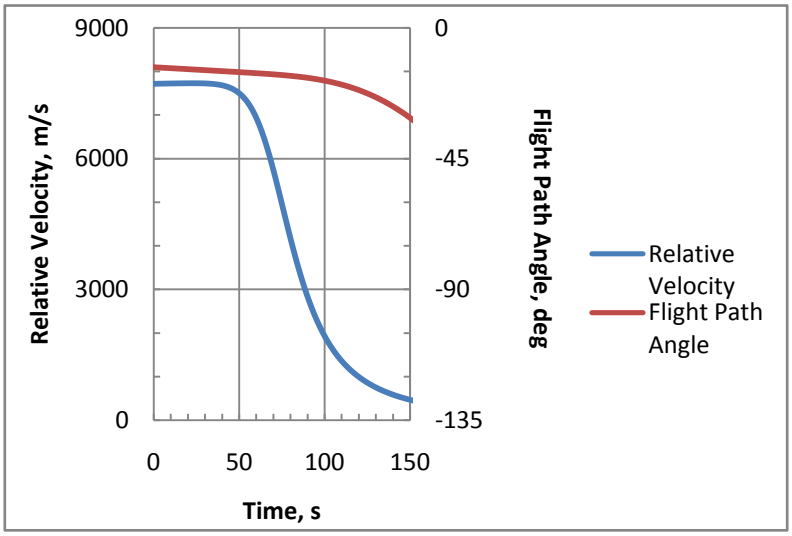

Figure 12: Pathfinder Simulated Velocity and Flight Path Angle Time History

The conclusion is that the comparison to experimental data and validates the accuracy of the simulation. Therefore, the results can be used with confidence for a first-order analysis.

3.0 Design Space Exploration Results

As described in the introduction, the goal of this study was to facilitate insights into the design space for planetary EDL vehicles. The results of this study, therefore, are a number of sensitivity analyses highlighting factors with the greatest impact on limiting the Mars EDL system design space. These are presented with comments in Sections 3.2 and 3.3. Additionally, some insights gained from studying the sensitivity analyses as a whole are used to formulate a promising design envelope for EDL vehicles for crewed missions to Mars, and these are presented in Section 3.4.

Each of the sensitivity analyses investigates the impact of a single factor, e.g. the effect of lift-over-drag ratio of the vehicle on its final landed mass performance as measured by the "payload mass fraction" which indicates what fraction of the overall entry mass is usable payload/cargo landed on the surface of Mars. However, as shown in the previous section, a large number of calculations need to be carried out in order to obtain this result. Therefore, the sensitivity analyses plots of a single factor vs. the landed mass performance have a lot of information embedded into them e.g. the actual start point for the propulsive descent. Since such top-level results can seem confusing or counter-intuitive without the knowledge of underlying details, the approach taken in this section is to first look at the basic results for a few data points in detail in order to build an understanding of the situation and then present the composite sensitivity analysis plots. For this approach, reference values are needed for the various parameters. Therefore, a reference entry vehicle is defined in Section 3.1 with characteristics which are used as default values for fixed parameters throughout the results section.

3.1 Reference Mars Entry Vehicle[†]

The reference Mars entry vehicle chosen for this study is similar to previous vehicles used for Mars missions. It is derived from the reference design used in the MIT/Draper CE&R study (9). The main considerations for this design were simplicity and good packaging properties due to the low sidewall angle of 20°.

The major specifications for this design are:

- Diameter: 12 m
- Length: 11.24 m
- Sidewall Angle: 20°
- Aeroshell Mass Fraction: 0.68 (See Eq 1 for definition)
- Nose radius: 11.66 m

[†] The reference Mars entry vehicle design work presented in this section was carried by the following individuals in the author's research group: Wilfried Hofstetter, Ryan McLinko and Emily Grosse.

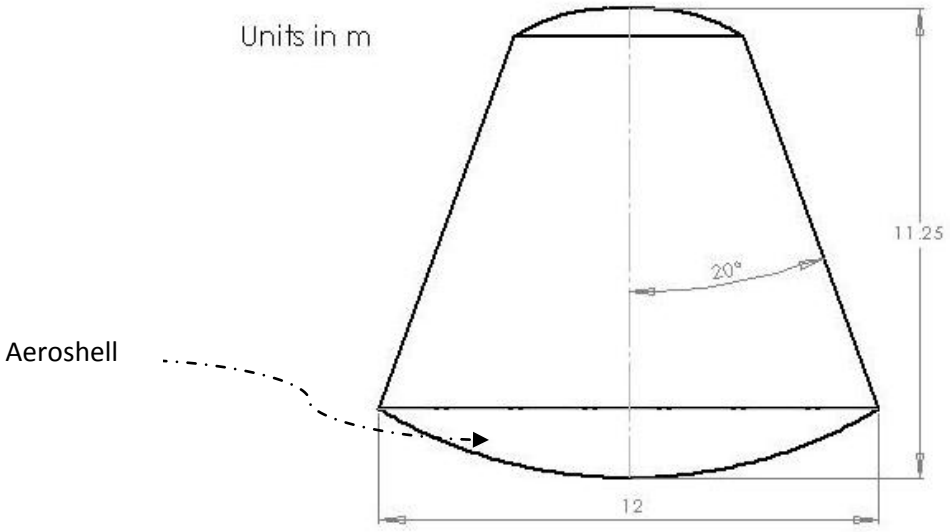

Figure 13: Reference Vehicle (Schematic drawn by Ryan McLinko)

In addition to the above attributes, the default value for the structural mass fraction of the descent propulsion system is set to 0.65 (See Eq 2 for definition).

3.1.1 Aerodynamic Characteristics

In order to conduct EDL analysis on this reference vehicle, knowledge of its aerodynamic characteristics was required. This was obtained using the Aerodynamic Preliminary Analysis System (APAS) software developed by NASA Langley, which conducts aerodynamic analyses "based on potential theory at subsonic/supersonic speeds and impact type finite element solutions at hypersonic conditions" (19) (20). It is capable of analyzing "three-dimensional configurations having multiple non-planar surfaces of arbitrary planform and bodies of non-circular contour" (20). For this study, the APAS results were validated for a simple flat plate case (See Appendix D: APAS Validation) before conducting runs for the reference vehicle geometry.

Aerodynamic characteristics of the reference vehicle geometry were obtained for several different freestream conditions; these were essentially the Mach number and atmospheric conditions at the point of maximum dynamic pressure during the EDL profile. Three different EDL profiles were simulated using the tool described in Section 2.0. The relevant inputs and outputs for these runs can be found in Table 4. Note that while the aerodynamic characteristics of the entry body were not known prior to conducing the APAS simulation, it was necessary to specify estimates for the EDL trajectory runs. The drag coefficient was estimated to be 1.68 and the lift-to-drag ratio was estimated to be 0.3.

Table 4: APAS Initial Conditions

Run #	Entry Mass (mT)	Drag Coefficient	Lift-to-Drag Ratio	Stagnation Pressure (Pa)	Stagnation Temperature (K)	Mach number
1	40	1.68	0.3	639098.5877	6958.6548	14.1539
2	80	1.68	0.3	1239728.1013	6980.2796	14.2868
3	120	1.68	0.3	1806607.2902	6992.8050	14.3631

Figure 14 shows the results from the APAS runs. It can be seen that the estimated L/D matches the value from APAS for an angle of attack of 17-18° but the drag coefficient has been overestimated. However, note that the aerodynamic characteristics seem insensitive to the stagnation pressure and temperature conditions. For better accuracy, the initial APAS run conditions were reset using drag coefficient and L/D values from the current APAS results in the EDL trajectory code to obtain new stagnation pressure, temperature and Mach number conditions. These can be found in Table 5. The pressure, temperature and Mach number conditions for runs 4 and 5 are bracketed by runs 1 to 3. Therefore, only the conditions from Run 6 were used for an additional APAS simulation. Results of this simulation can also be found in Figure 14.

In Figure 14, the drag coefficient and L/D values obtained for each run overlap for all angles of attack, thus all the lines for C_d and L/D lie on top of each other. This illustrates the insensitivity of the aerodynamic characteristics to the atmospheric conditions and Mach number. This is in line with expectations, since it is well known that aerodynamic characteristics at hypersonic speeds (Mach number > 5) remain almost constant.

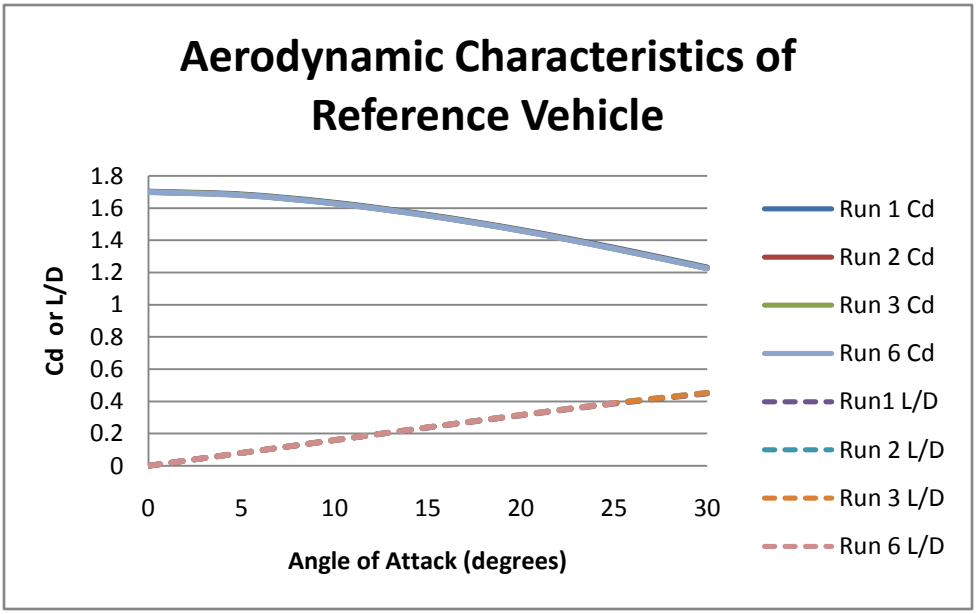

Figure 14: Aerodynamic Characteristics of Reference Vehicle

Table 5: Revised APAS Initial Conditions

Run #	Entry Mass (mT)	Drag Coefficient	Lift-to-Drag Ratio	Stagnation Pressure (Pa)	Stagnation Temperature (K)	Mach number
4	40	1.5	0.3	713434.2653	6962.2015	14.1759
5	80	1.5	0.3	1378258.2613	6983.7935	14.3082
6	120	1.5	0.3	2003823.3219	6996.2822	14.3841

An important point to note about the results in Figure 14 is that the calculations were carried for an atmosphere composed of Carbon Tetrafluoride. This is due to the limited set of gases available for simulation in APAS. The gas properties are used in the calculation of freestream conditions from the stagnation pressure and temperature specified. Freestream conditions calculated by APAS using Carbon Tetrafluoride most closely matched those for the Martian atmospheric model used in this study and hence, this gas was chosen for simulation of aerodynamic characteristics of the reference vehicle. The major effect of the gas composition on the aerodynamic characteristics is on skin friction drag.

For this study, a reference angle of attack of around 18° was chosen, which gave a drag coefficient of approximately 1.5 and a lift-to-drag ratio of around 0.3.

3.1.2 Sample Entry Trajectories

In order to better understand and visualize the sensitivity analyses results presented later on in this chapter, sample entry trajectories are presented in the following figures for vehicles with different characteristics as listed in the accompanying captions.

The interesting characteristics of the altitude vs. range plots are that they show the magnitude of vertical oscillations experienced by the vehicle. The altitude vs. velocity plots show the altitudes at which the vehicle reaches various Mach numbers. Mach number contours ranging from 2 to 5 are provided on each plot for reference.

Comparing the trajectories for vehicles with L/D=0 and L/D=0.3 (Figure 15 and Figure 16) shows that the vehicle with L/D=0.3 experiences a slight vertical oscillation near the end of its trajectory and reaches various Mach numbers at higher altitudes compared to the zero lift vehicle. The expectation would be that these trends would continue for even higher L/D values. However, the third plot (Figure 17) for L/D=2.0 reveals that the case is somewhat different. The vertical oscillations increase to the point that the vehicle skips out of the atmosphere on the first entry and then re-enters the atmosphere. The second entry then sets the initial conditions for its trajectory through the atmosphere. In the particular case of the vehicle with L/D of 2.0, the vehicle does not actually manage to cross the Mach number contours before reaching the surface. Again, the reason for this is the “re-entry” experienced by the vehicle with L/D of 2.0. One thing to note is that for a vehicle with such a high L/D, the lift vector direction is often actively controlled which may alleviate the skip-out problem and the vehicle may be

able to cross the Mach number contours before reaching the surface. However, in this study, active control of lift vector direction was not used. This fact should be kept in mind when considering the results presented here for very high L/D vehicles.

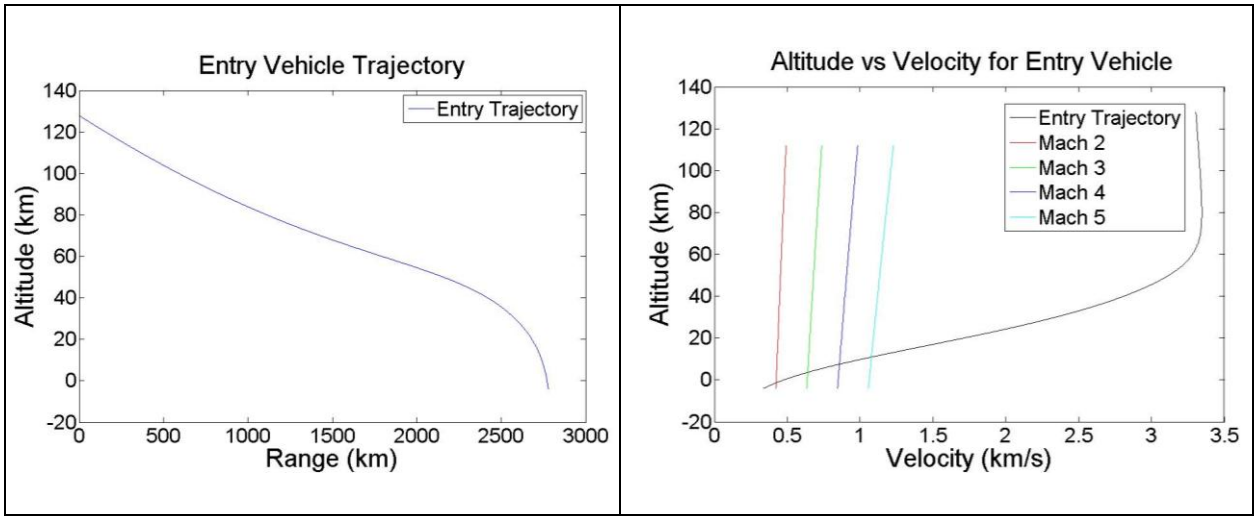

Figure 15: Entry Trajectory for Vehicle with Cd=1.5, L/D=0, Mass=40 mT

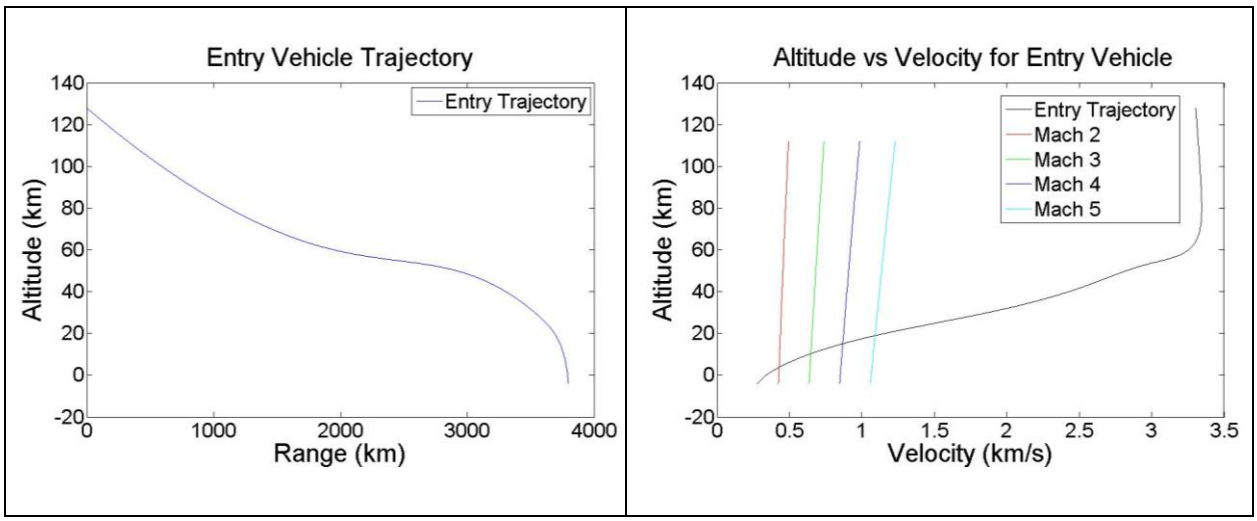

Figure 16: Entry Trajectory for Vehicle with Cd=1.5, L/D=0.3, Mass=40 mT

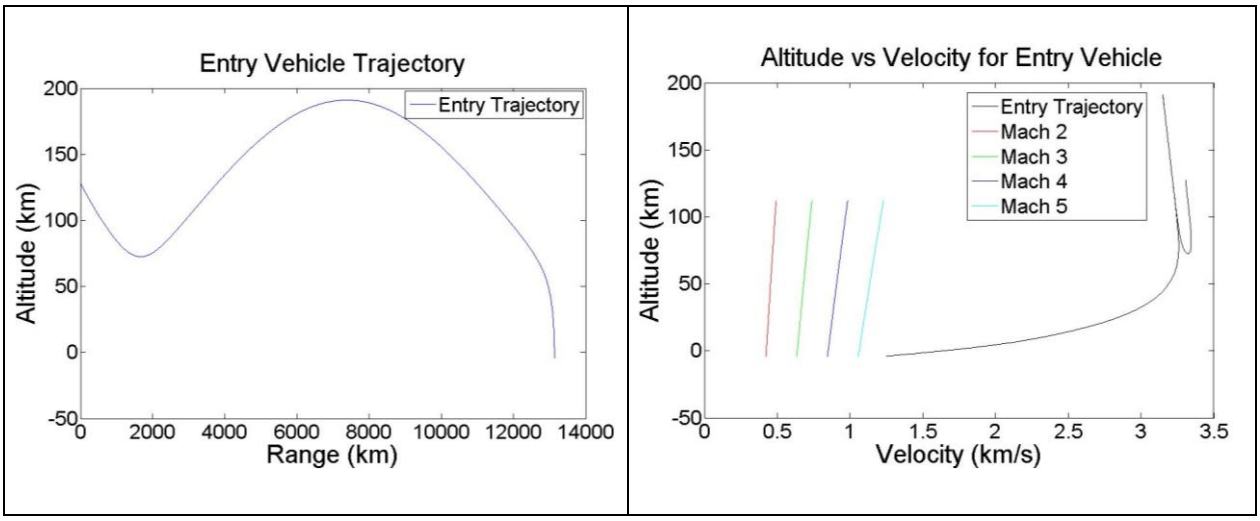

Figure 17: Entry Trajectory for Vehicle with Cd=1.5, L/D=2.0, Mass=40 mT

Since the vehicles presented in two of the cases above reach reasonably slow speeds near the surface, the next step is to add the propulsive descent to the trajectories. Assuming that drogue chutes would be utilized for separation of the aeroshell, this provides an upper limit to the propulsive descent start of Mach 3 as this is expected to be the operational limit for parachutes in the Martian atmosphere (8). This is specified as the propulsive descent transition criteria for each of the three cases and the results are as shown in the following figures. Associated payload mass fractions and thrust requirements are listed in Table 6.

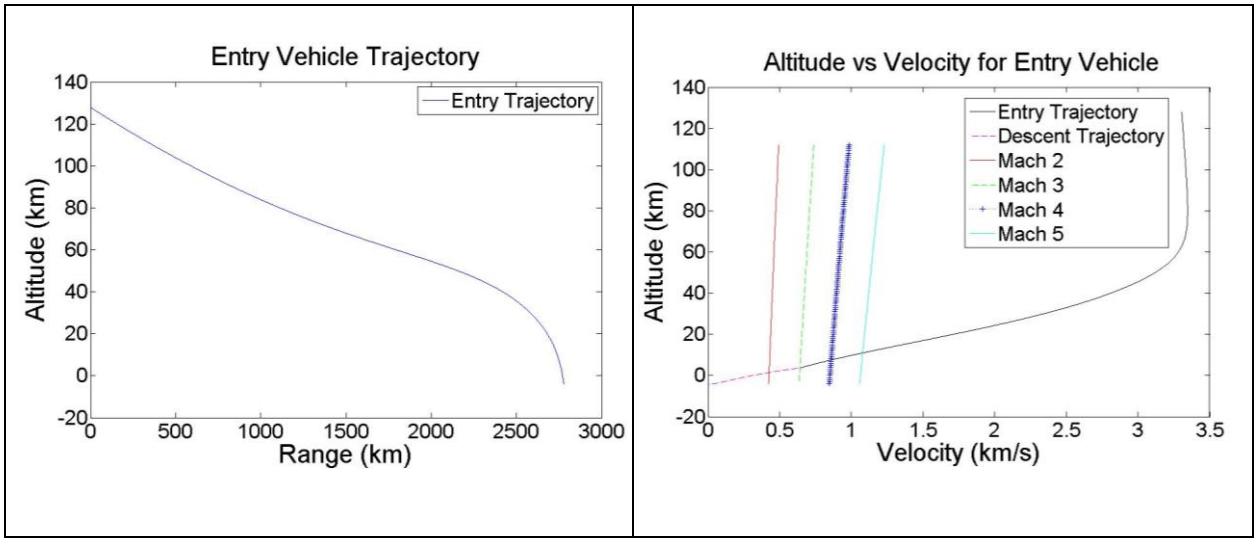

Figure 18: EDL Trajectory for Vehicle with Cd=1.5, L/D=0, Mass=40 mT

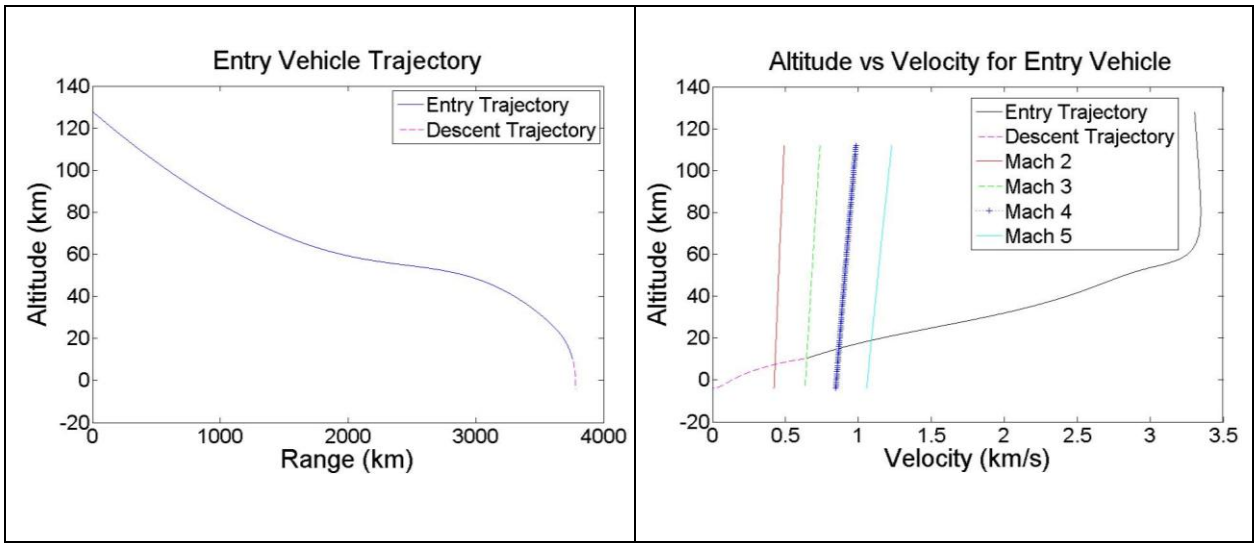

Figure 19: EDL Trajectory for Vehicle with Cd=1.5, L/D=0.3, Mass=40 mT

Table 6: Payload Mass and Thrust Results for Sample EDL Trajectories

Entry Vehicle Characteristics	Payload Mass Fraction	Thrust Required/10^7 (N)	Powered Descent Time (s)
Cd=1.5, L/D=0, Mass=40mT	0.4447	0.1460	72.30
Cd=1.5, L/D=0.3, Mass=40mT	0.4027	0.0994	144.30

3.2 Entry Parametric Analyses

Before looking at the sensitivity analyses for the full EDL trajectory, it is insightful to consider the sensitivity of only the entry trajectory to various parameters. The effect of most of the major parameters can be observed by plotting performance results while varying the ballistic coefficient and L/D. The vehicle performance is characterized by the altitude reached by the vehicle at a certain Mach number and the points considered are Mach 4, 3 and 2. The ballistic coefficient represents an efficient way to observe the effect of several variables by collapsing them together and is commonly used in aerospace literature. It is defined as:

$$\beta = \frac{m}{C_D A} \quad \textbf{Eq 24}$$

where m is entry vehicle mass, C_d is the drag coefficient and A is frontal area at zero angle-of-attack.

The entry parametric sensitivity results are presented in the following subsections. For clarity and to reduce simulation run-time, the L/D range has been subdivided into three parts. In each subsection, where needed, the full trajectory for a single data point is shown to clarify apparent anomalies in the parametric analysis results.

3.2.1 Results for L/D of 0 to 0.5

Figure 20, Figure 21 and Figure 22 show the final altitude reached by the entry body at a certain Mach number for combinations of ballistic coefficients and L/D values. An important point is that the Mars entry conditions for all data points were kept the same: an elliptical orbit of 500 km x 50 km.

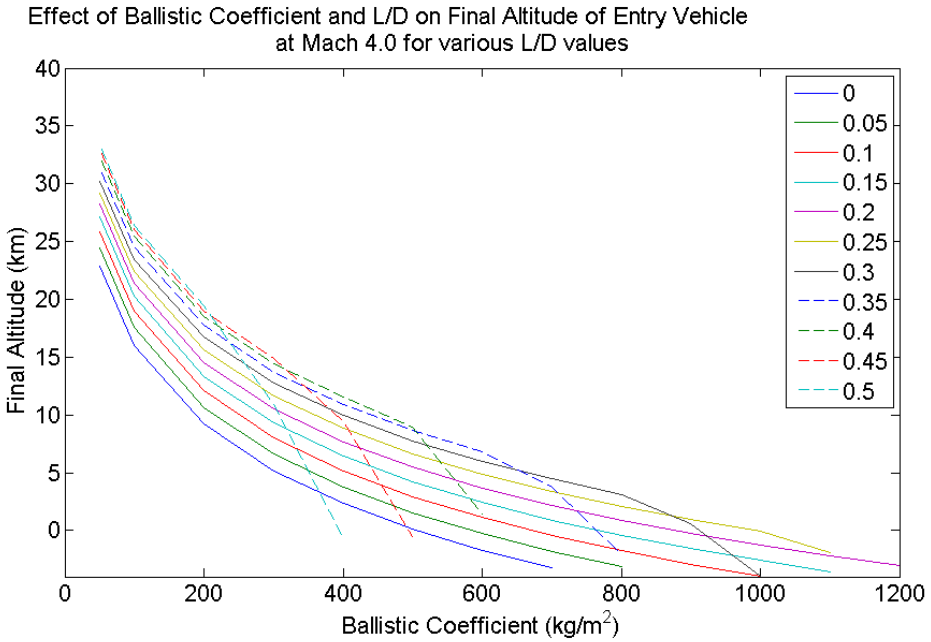

Figure 20: Sensitivity of Mach 4 Altitude to Ballistic Coefficient and L/D for L/D range of 0 to 0.5

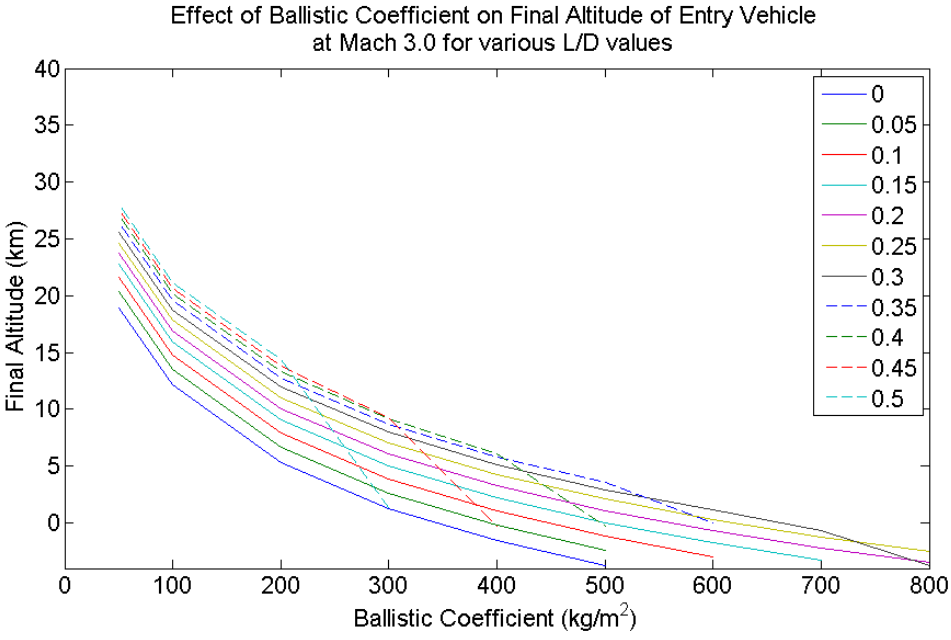

Figure 21: Sensitivity of Mach 3 Altitude to Ballistic Coefficient and L/D for L/D range of 0 to 0.5

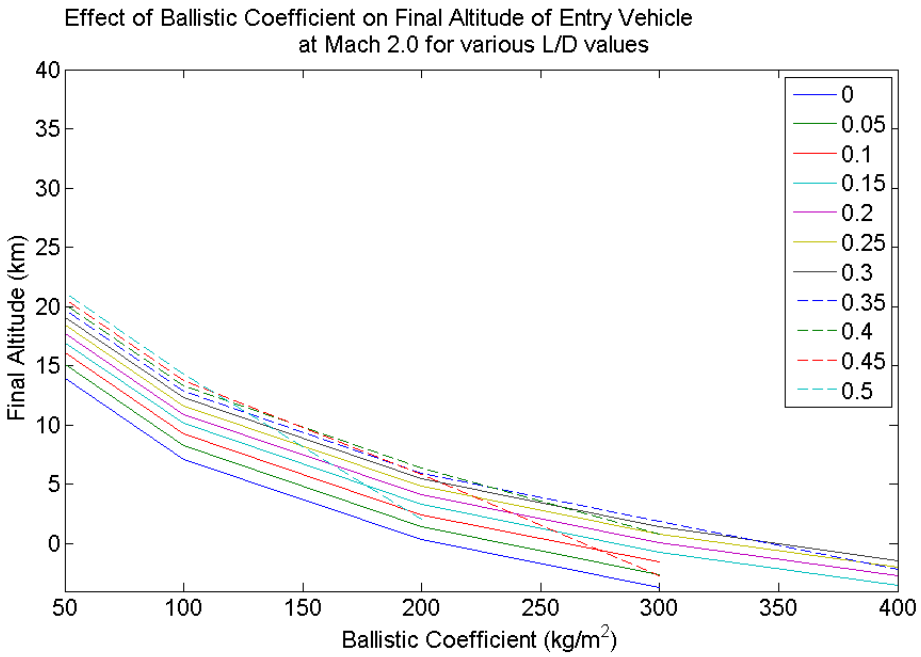

Figure 22: Sensitivity of Mach 2 Altitude to Ballistic Coefficient and L/D for L/D range of 0 to 0.5

As expected, for a particular L/D, for higher ballistic coefficients, the altitude reached for a certain Mach number is lower due to the body's greater inertia which makes it less sensitive to aerodynamic deceleration forces. Note that although the simulation was run for a ballistic coefficient range of 50 to 2400 kg/m^2, a limit is reached for a particular L/D value after which the vehicle does not slow to, say, a Mach number of 4 before reaching the surface. This limit decreases for a lower Mach number as can be seen in the plots above.

Additionally, as also expected, the altitude at a particular Mach number for a certain ballistic coefficient increases with L/D. A higher L/D causes the vehicle to stay in the upper atmosphere for a longer period of time thus, bleeding off the entry energy at higher altitudes. This means that it reaches slower speeds at higher altitudes compared to bodies with lower L/D values. From the graphs, it can be seen however, that there are anomalies in the data e.g. for L/D of 0.5. To investigate this, the full entry trajectories are presented for the following data points of interest: ballistic coefficients of 200 and 400 and L/D values of 0 and 0.5.

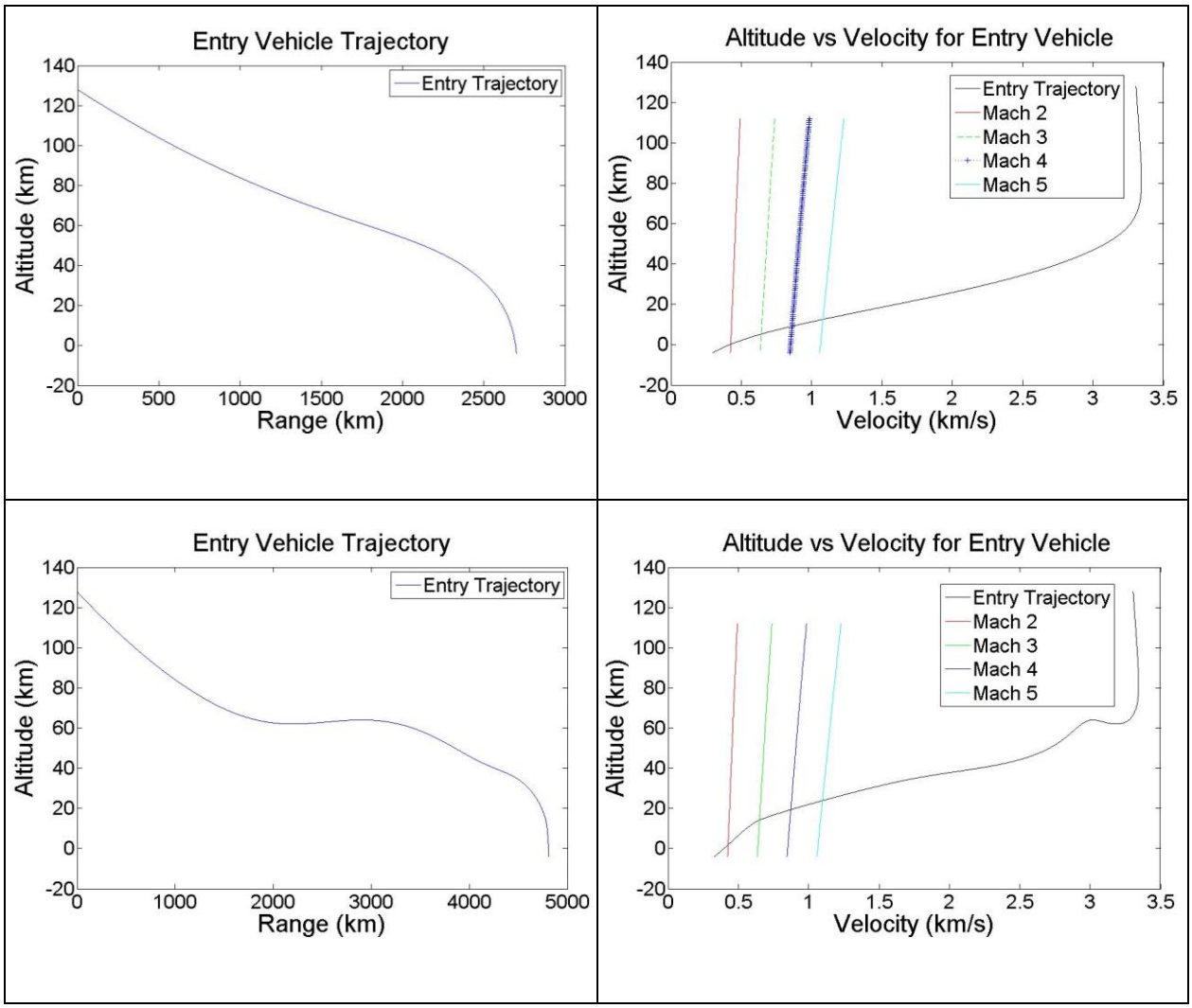

Figure 23: Entry Trajectories for Vehicle with Ballistic Coefficient = 200. Top: L/D = 0, Bottom: L/D = 0.5

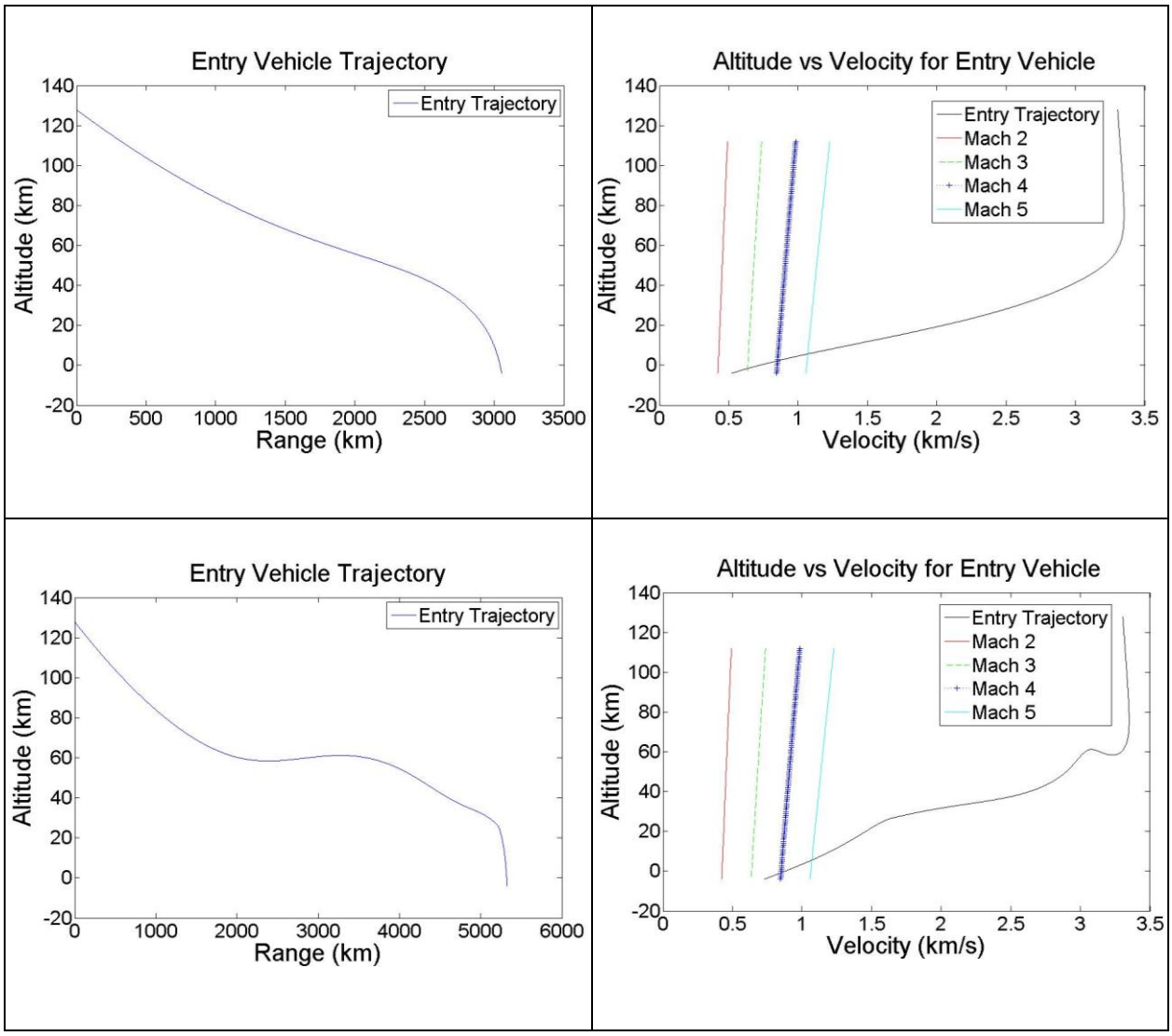

Figure 24: Entry Trajectories for Vehicle with Ballistic Coefficient = 400. Top: L/D = 0, Bottom: L/D = 0.5

From Figure 23 and Figure 24, two observations are clear. The first is that a vehicle with a lower ballistic coefficient slows down to a lower speed before reaching the surface. This is in keeping with what was seen in the sensitivity graphs. The second observation is that a vehicle with a sufficiently high L/D experiences vertical oscillations during its trajectory. This is expected due to the exponential nature of the atmospheric density profile. However, this oscillation has a significant effect on where the vehicle crosses the Mach number lines in the figures presented.

Figure 24 shows that for the higher L/D vehicle, the second oscillation results in a steeper entry thus making it fall short of the Mach 3 line, while the vehicle with L/D = 0 crosses this line before reaching the surface. Therefore, these oscillations for vehicles with higher L/D cause the anomalies in the sensitivity graphs presented earlier.

3.2.2 Results for L/D of 0.5 to 1.0

The final altitude data for L/D values of 0.5 to 1.0 presents a more complicated picture than the lower L/D range. At Mach 4, the expected trend of higher altitude for higher L/D is exhibited only for high ballistic coefficient values. At lower ballistic coefficient values, the trend for all Mach number graphs seems to be reversed. Also, there seem to be a lot of infeasible data points in the middle ballistic coefficient range.

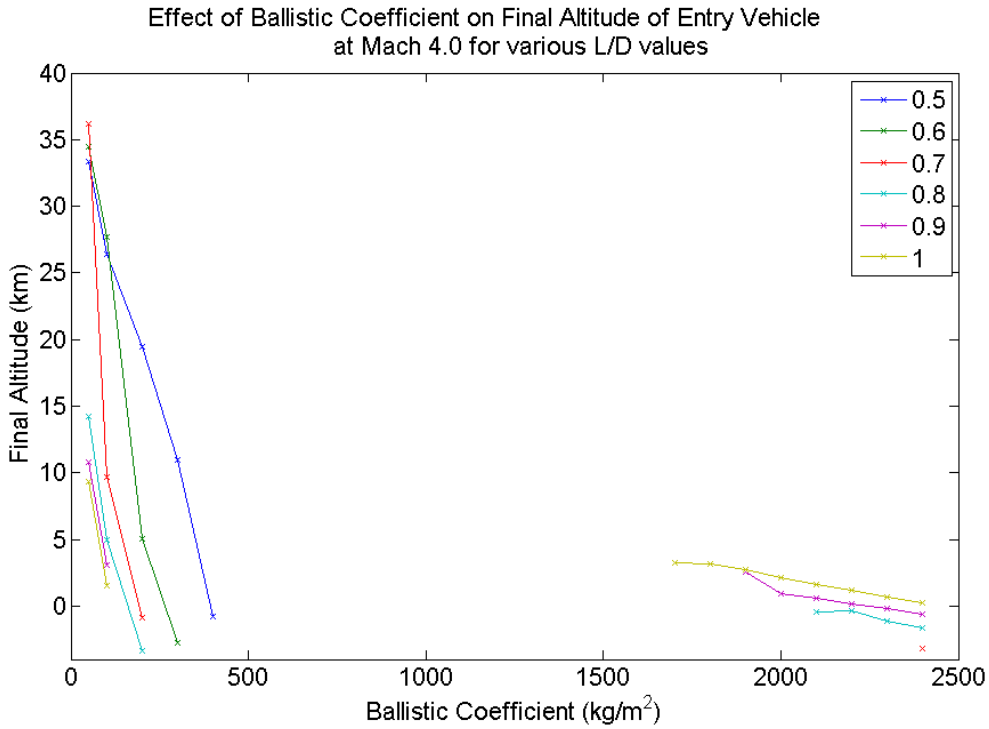

Figure 25: Sensitivity of Mach 4 Altitude to Ballistic Coefficient and L/D for L/D range of 0.5 to 1.0

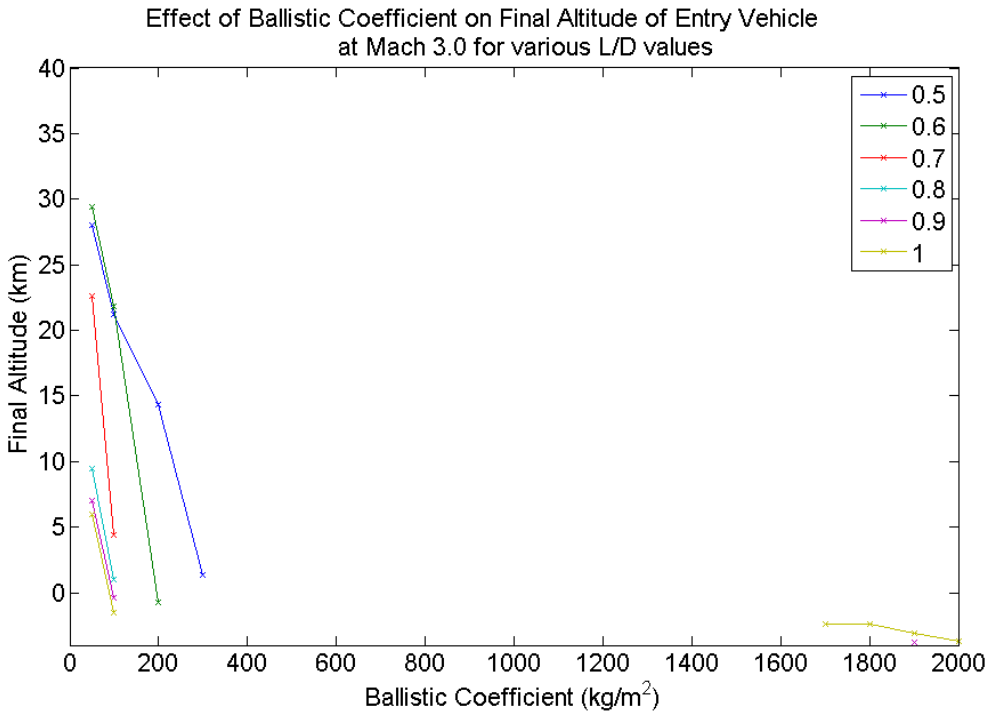

Figure 26: Sensitivity of Mach 3 Altitude to Ballistic Coefficient and L/D for L/D range of 0.5 to 1.0

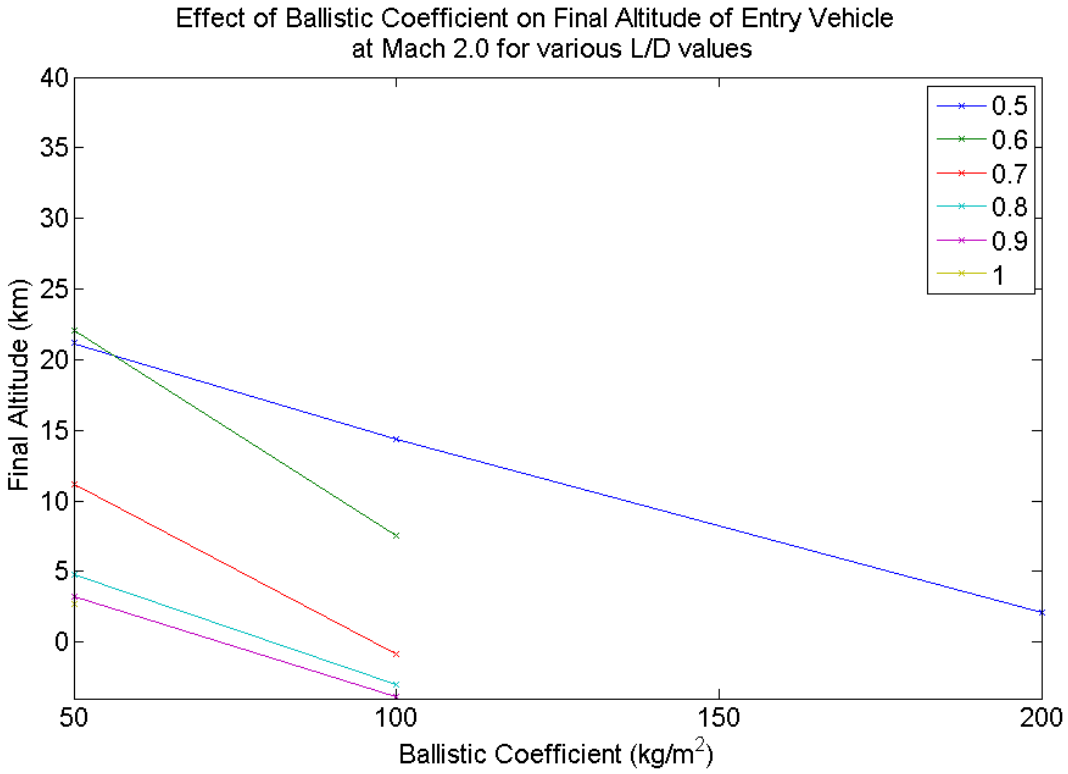

Figure 27: Sensitivity of Mach 2 Altitude to Ballistic Coefficient and L/D for L/D range of 0.5 to 1.0

Let us investigate the unexpected qualities of this data set one by one. The L/D trend reversal is investigated by plotting individual trajectories for a ballistic coefficient of 100. Figure 28, Figure 29 and Figure 30 show the trajectories for L/D values of 0.5, 0.7 and 1.0.

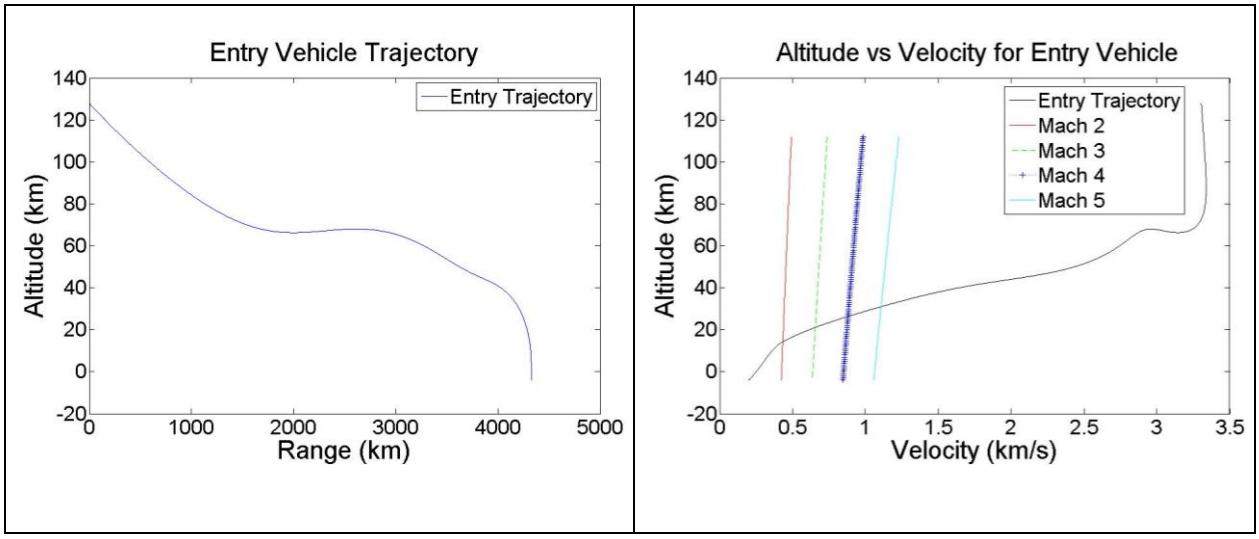

Figure 28: Entry Trajectory for Vehicle with Ballistic Coefficient = 100, L/D = 0.5

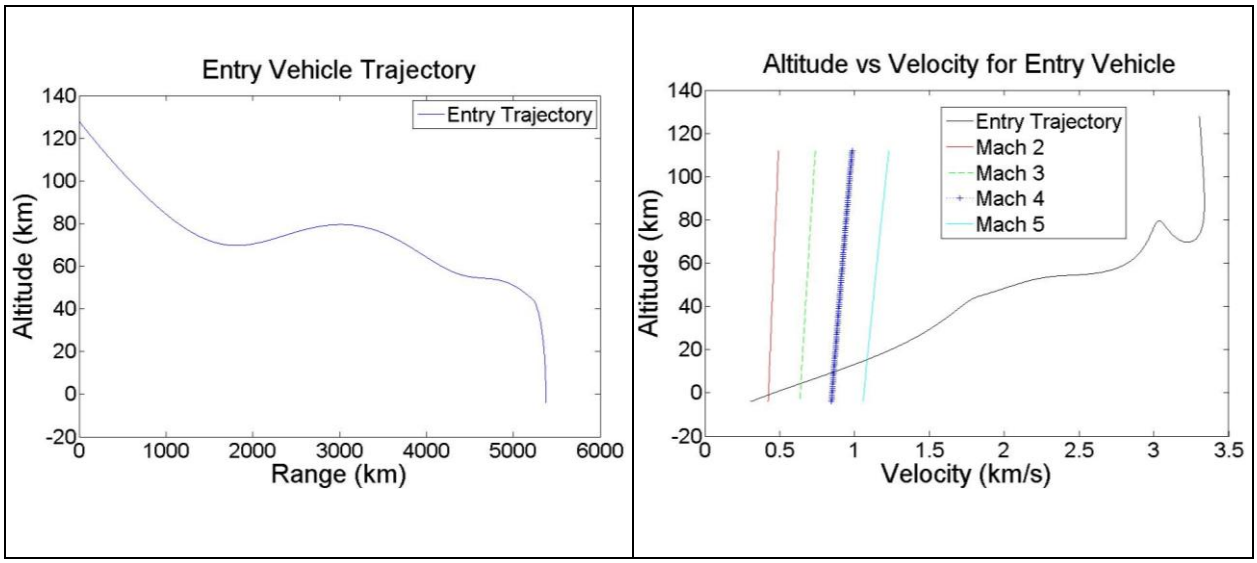

Figure 29: Entry Trajectory for Vehicle with Ballistic Coefficient = 100, L/D = 0.7

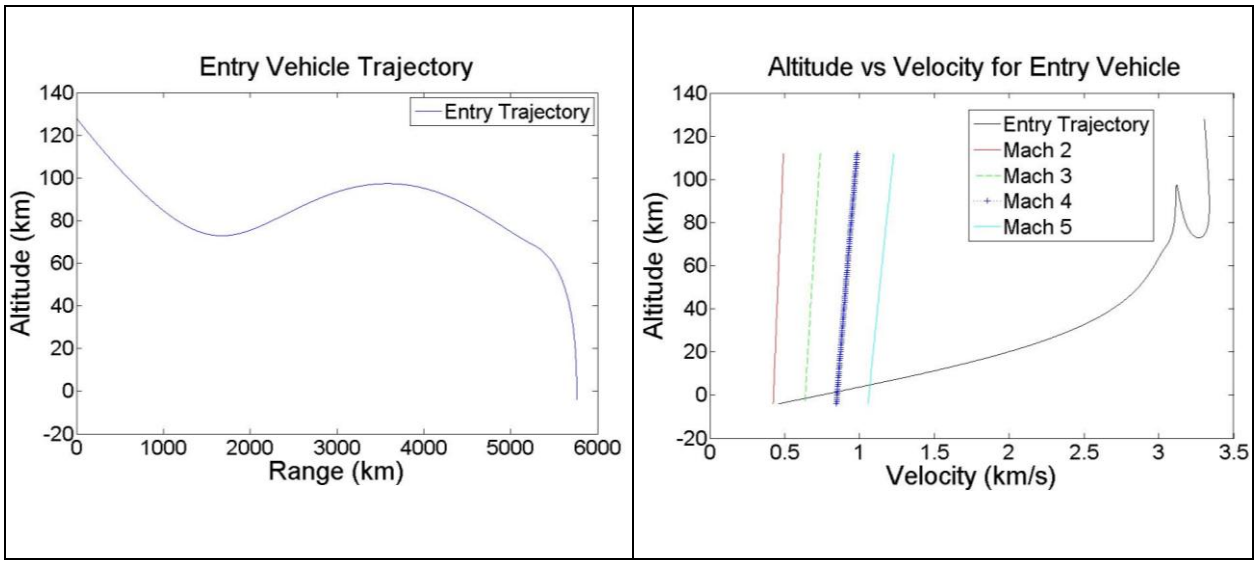

Figure 30: Entry Trajectory for Vehicle with Ballistic Coefficient = 100, L/D = 1.0

As can be seen from the Figure 28, Figure 29 and Figure 30, upon first entering the atmosphere, the vehicle experiences a vertical oscillation due to its high lift properties. The magnitude of this oscillation increases with increasing L/D as expected. The higher lift creates a stronger upward force on the body thus pushing it to a higher altitude. At the end of the oscillation, the vehicles with a higher L/D seem to acquire a steeper trajectory slope thus resulting in a lower altitude for a particular Mach number compared to a vehicle with lower L/D. It should be stressed here that for the trajectories presented in this study, the vehicle was not actively controlled. If a technique such as lift vector direction control is used, the “skip” behavior described above can be mitigated.

The second issue of the large number of infeasible data points was tackled in a similar way i.e. by investigating the characteristics of individual trajectories in the infeasible as well as feasible regions.

As mentioned above, for a high L/D, the vehicle experiences a significant vertical oscillation. For a low ballistic coefficient case (see Figure 30), after the oscillation, the vehicle slows down enough to cross the Mach number contour lines and results in a feasible data point on the sensitivity analysis plots. As the ballistic coefficient gets higher, the vehicle is unable to slow down enough to appear on the plots, which is in line with expectations (see Figure 31). However, as seen in Figure 25, Figure 26 and Figure 27, an interesting behavior is that very high ballistic coefficient points seem to be feasible when the mid-range ballistic coefficient points are not. Figure 32 presents the trajectory for a high ballistic coefficient case with L/D of 1.0. As can be seen, for this high ballistic coefficient and high L/D case, the vehicle experiences a skip-out upon first entering the atmosphere. Upon "re-entry", it also experiences a minor oscillation in the upper atmosphere. Looking at the range vs. altitude history of the vehicle, it is clear that the vehicle spends a considerable amount of time in the upper atmosphere which results in a significant amount of deceleration. Due to this, the vehicle is able to reach slower speeds compared to the moderate ballistic coefficient vehicle.

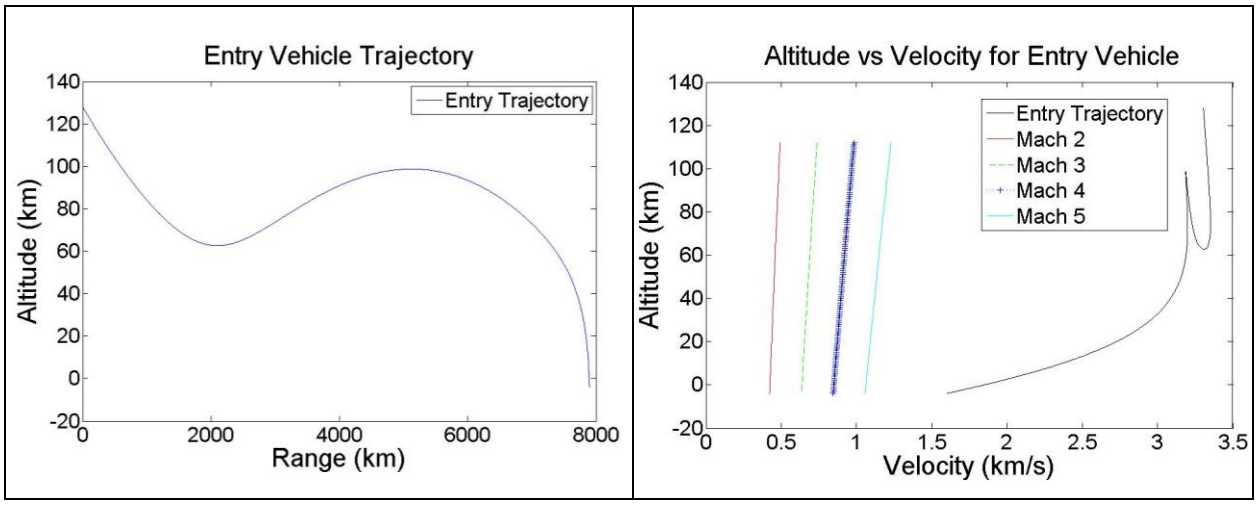

Figure 31: Entry Trajectory for Vehicle with Ballistic Coefficient = 500, L/D = 1.0

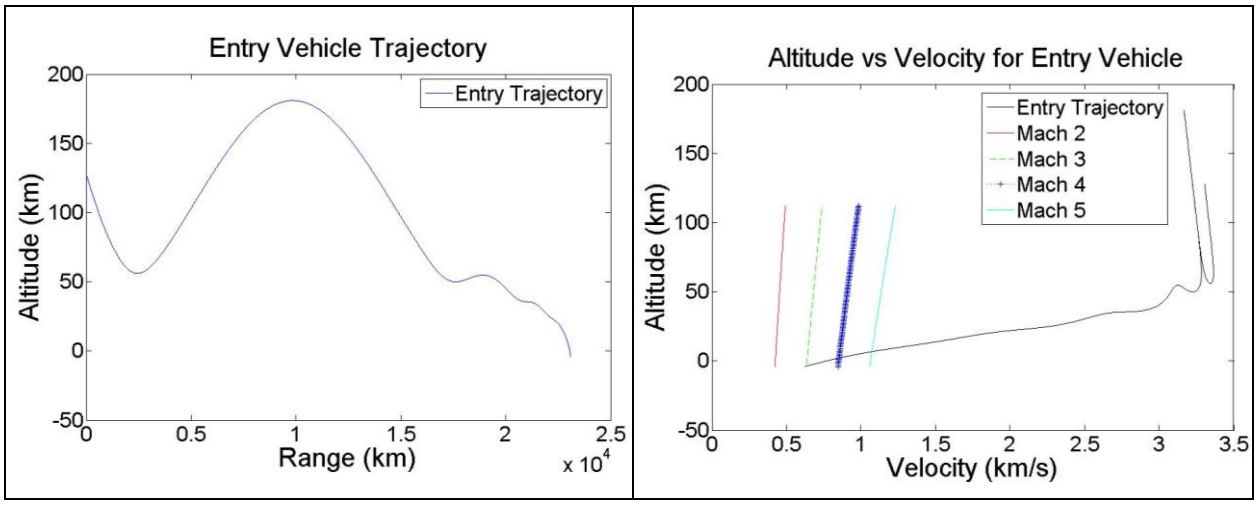

Figure 32: Entry Trajectory for Vehicle with Ballistic Coefficient = 2000, L/D = 1.0

Another observation from Figure 32 is that upon “re-entry”, the flight path angle of the vehicle is much steeper than before and the vehicle does not skip out a second time. This indicates that the entry flight path angle may play a role in the amount of deceleration achieved by the vehicle. This in turn impacts whether or not it appears as a “feasible” point on the sensitivity plots presented above. Therefore, a second set of sensitivity plots was created in which the entry conditions were varied to examine the sensitivity of the trajectories to entry angle. In this study, the entry angle is varied by varying the entry orbit. For reference, the entry conditions obtained using this method are presented in Table 7. The sensitivity plots are presented in Figure 33, Figure 34 and Figure 35. For clarity, the data is also presented in table form in Table 8, Table 9 and Table 10. Additionally, individual trajectories for a ballistic coefficient of 500 showing in detail the effect of entry angle are presented in Figure 36.

Table 7: Mars Entry Conditions for Varying Entry Orbit

Entry Orbit Periapsis (km)	Entry Orbit Apoapsis (km)	Entry Velocity (km/s)	Entry Angle (deg)
50	500	3.5548	-2.6610
50	550	3.5659	-2.8159
50	600	3.5768	-2.9591
50	650	3.5875	-3.0924
50	700	3.5981	-3.2170
50	750	3.6085	-3.3342
50	800	3.6187	-3.4446
50	850	3.6287	-3.5491
50	900	3.6386	-3.6483
50	950	3.6483	-3.7426
50	1000	3.6579	-3.8325

Table 8: Tabulated Data for Sensitivity of Mach 4 Altitude to Ballistic Coefficient and Entry Conditions, L/D = 1.0

	Mach 4 Final Altitude (km)					
Ballistic Coefficient (kg/m^2)	500	600	700	800	900	1000
Entry Orbit Apoapsis (km)						
550						
600						
650					-2.80	8.41
700				12.63	10.09	9.13
750			13.81	11.35	10.18	9.08
800		13.04	12.69	11.36	10.13	9.01
850	3.75	14.14	12.71	11.32	10.06	8.92
900	15.32	14.27	12.68	11.26	9.98	-1.32
950	16.08	14.25	12.63	11.18	7.55	
1000	16.10	14.22	12.55	11.09		

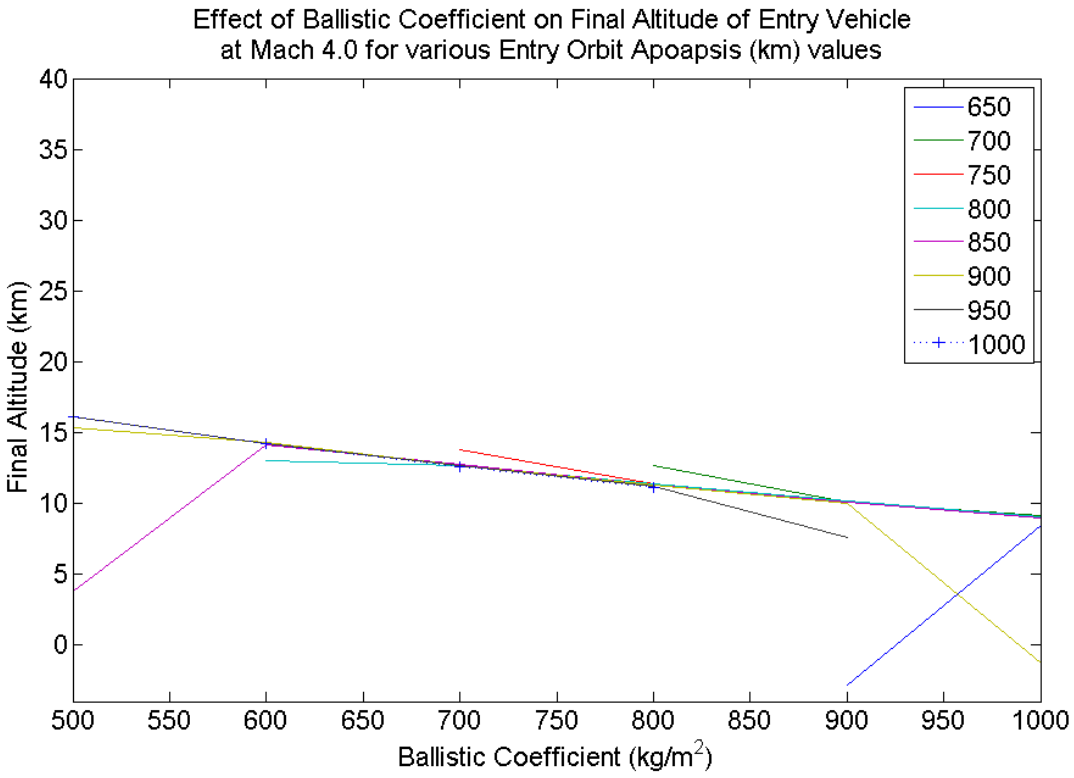

Figure 33: Sensitivity of Mach 4 Altitude to Ballistic Coefficient and Entry Conditions, L/D = 1.0

Table 9: Tabulated Data for Sensitivity of Mach 3 Altitude to Ballistic Coefficient and Entry Conditions, L/D = 1.0

	Mach 3 Final Altitude (km)					
Ballistic Coefficient (kg/m^2)	500	600	700	800	900	1000
Entry Orbit Apoapsis (km)						
550						
600						
650						3.39
700				4.44	4.49	3.24
750			6.91	5.59	4.27	3.18
800		7.29	6.92	5.44	4.22	3.15
850	2.29	8.53	6.77	5.39	4.19	3.14
900	10.34	8.34	6.72	5.35	4.18	
950	10.24	8.28	6.68	5.34		
1000	10.14	8.23	6.66	5.34		

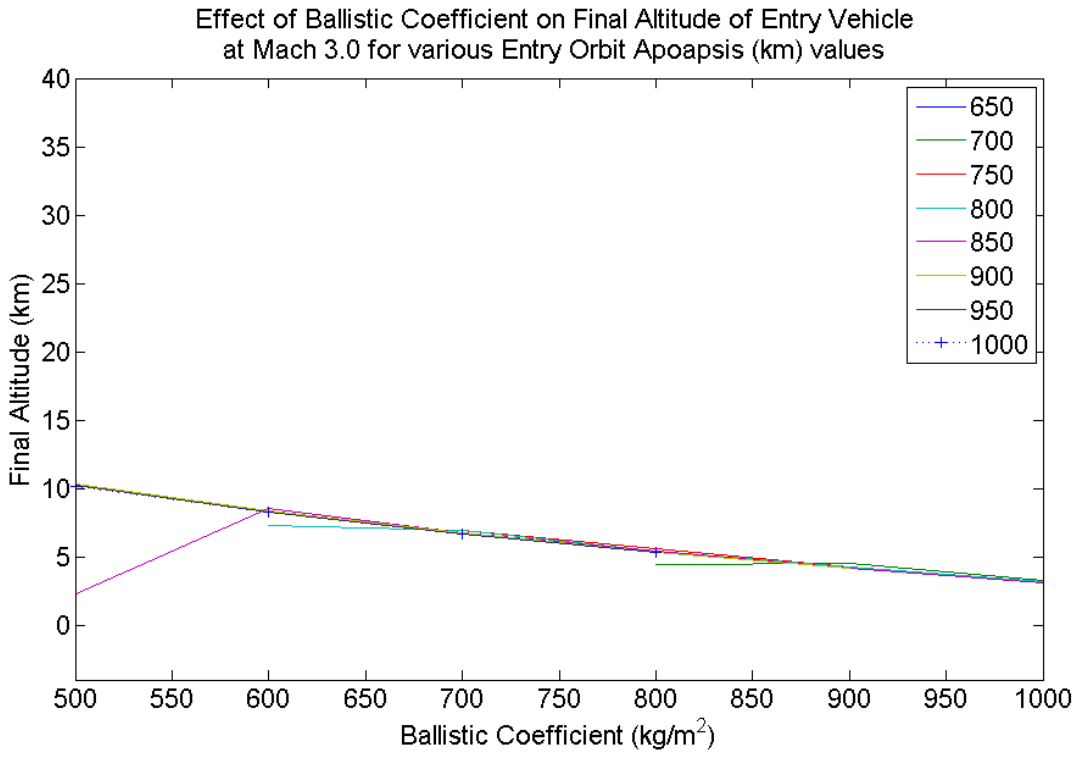

Figure 34: Sensitivity of Mach 3 Altitude to Ballistic Coefficient and Entry Conditions, L/D = 1.0

Table 10: Tabulated Data for Sensitivity of Mach 2 Altitude to Ballistic Coefficient and Entry Conditions, L/D = 1.0

	Mach 2 Final Altitude (km)					
Ballistic Coefficient (kg/m^2)	500	600	700	800	900	1000
Entry Orbit Apoapsis (km)						
550						
600						
650						
700				-2.39	-3.43	
750			-1.28	-2.27	-3.40	
800		1.15	-0.95	-2.23	-3.37	
850	0.67	0.58	-0.92	-2.21	-3.34	
900	2.58	0.60	-0.89	-2.18	-3.32	
950	2.38	0.62	-0.87	-2.15		
1000	2.40	0.64	-0.84			

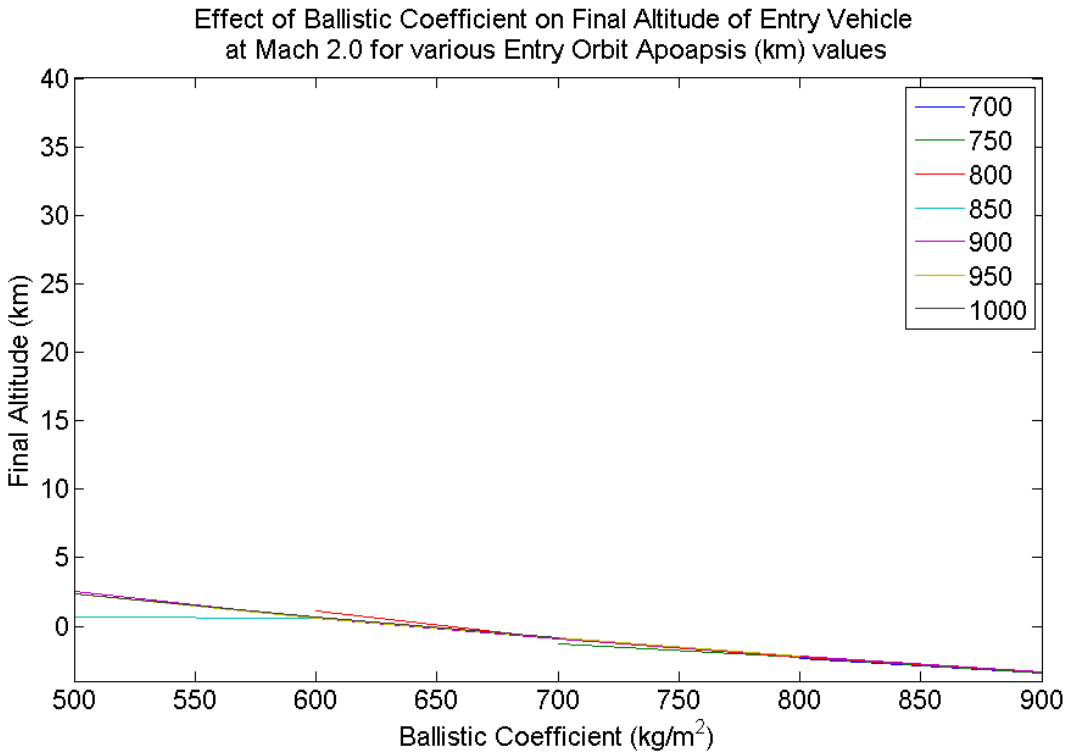

Figure 35: Sensitivity of Mach 2 Altitude to Ballistic Coefficient and Entry Conditions, L/D = 1.0

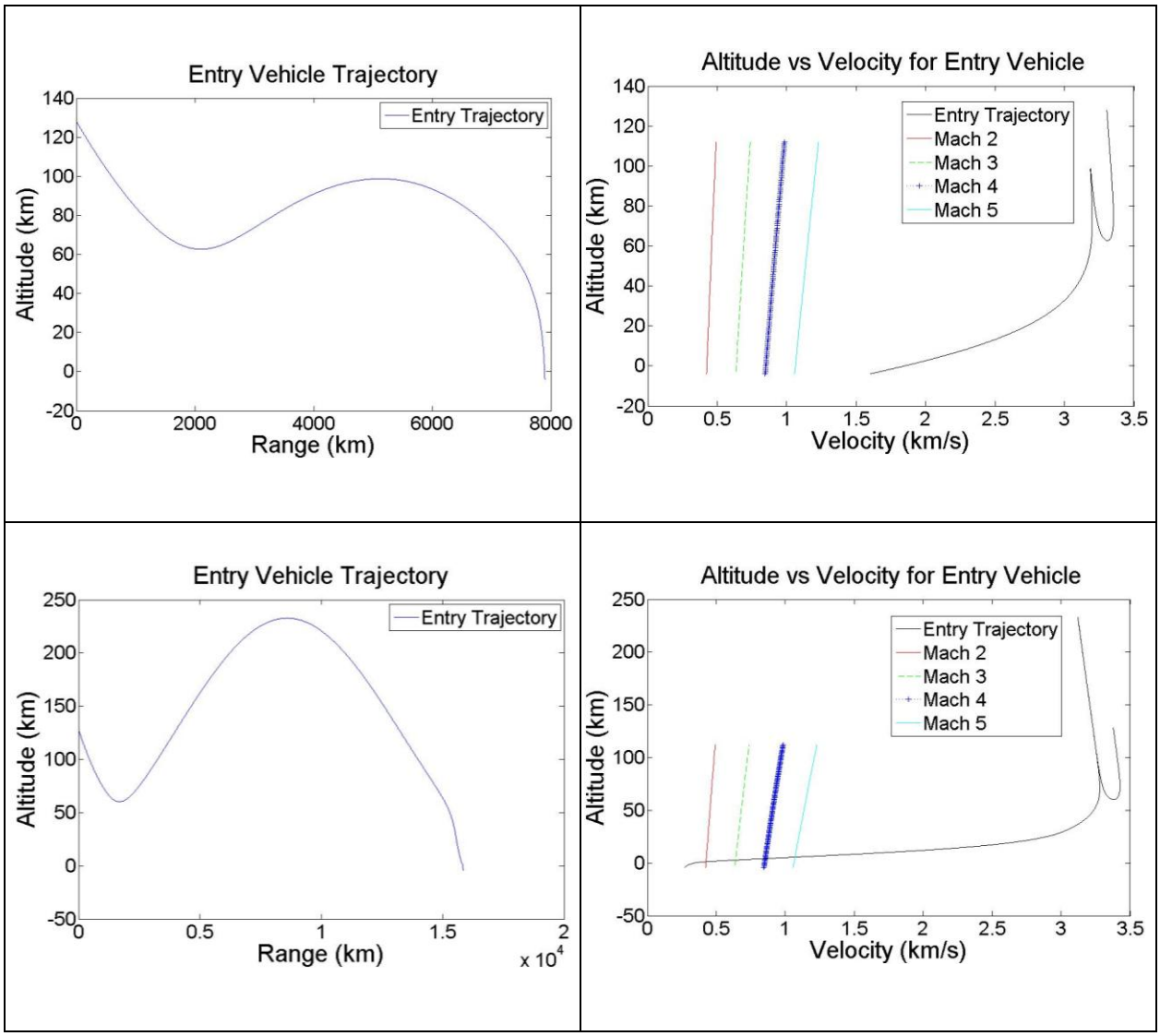

Figure 36: Entry Trajectory for Vehicle with Ballistic Coefficient = 500, L/D = 1.0. Top: Entry Apoapsis Altitude = 500 km, Bottom: Entry Apoapsis Altitude = 850 km

It can be seen from Figure 33, Figure 34, Figure 35 and Figure 36 that vehicles with mid-range ballistic coefficients are extremely sensitive to changes in the entry angle. This implies that the entry angle must be carefully chosen for high L/D vehicles or active control of the lift vector direction must be used in order to avoid skip-out. This can prove to be problematic for parametric sensitivity analyses since the L/D may not be varied independently. However, for the rest of this study, the L/D range will be close to the reference vehicle L/D value of 0.3. Thus, it will be low enough to avoid this problem.

3.2.3 Results for L/D of 1.0 to 1.5

For completeness, the final altitude results for very high L/D values are presented in this section. The trends are the same as for the L/D range of 0.5 to 1.0 and the anomalies can be attributed to the same causes.

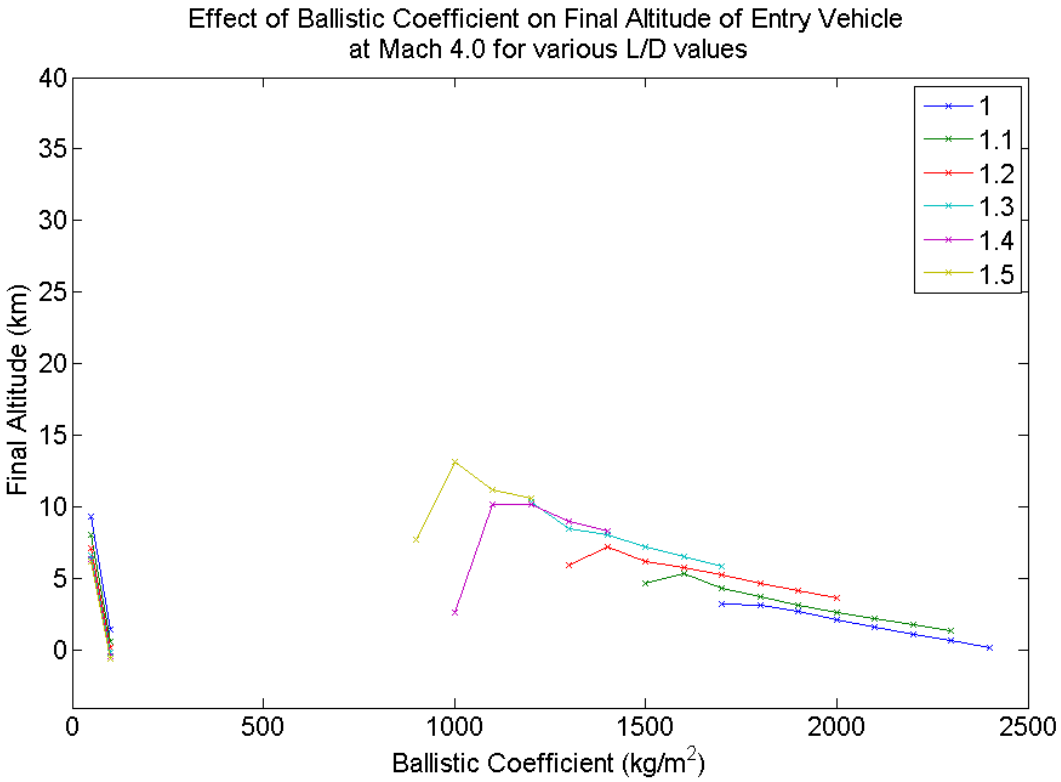

Figure 37: Sensitivity of Mach 4 Altitude to Ballistic Coefficient and L/D for L/D range of 1.0 to 1.5

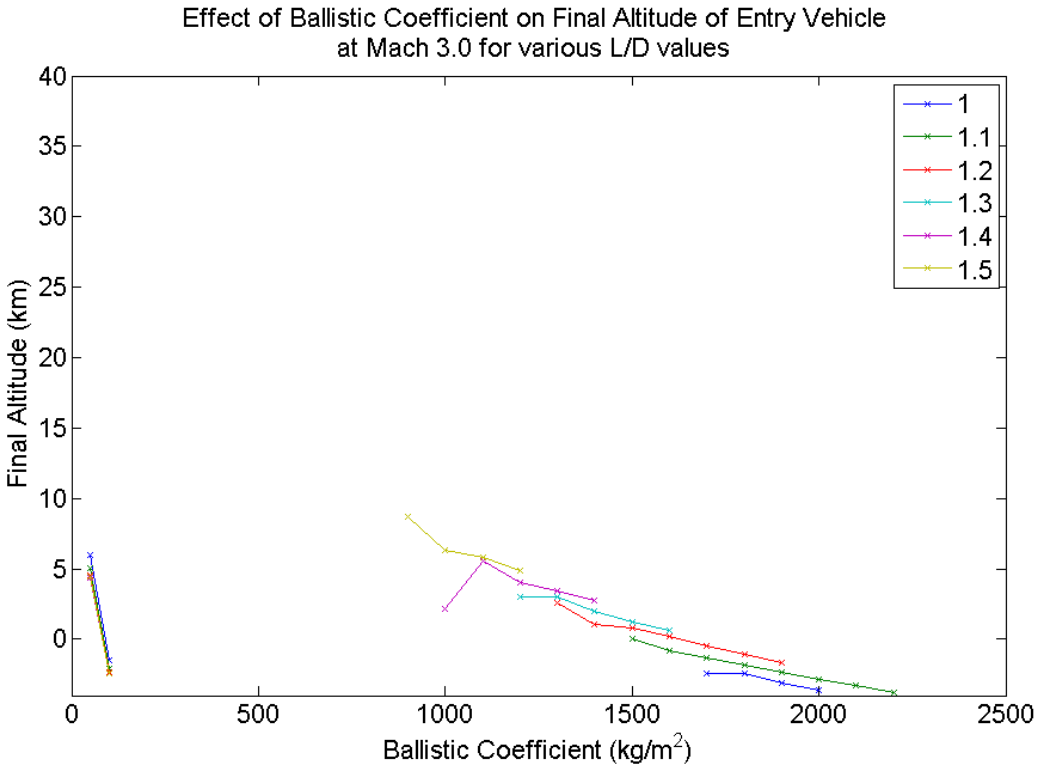

Figure 38: Sensitivity of Mach 3 Altitude to Ballistic Coefficient and L/D for L/D range of 1.0 to 1.5

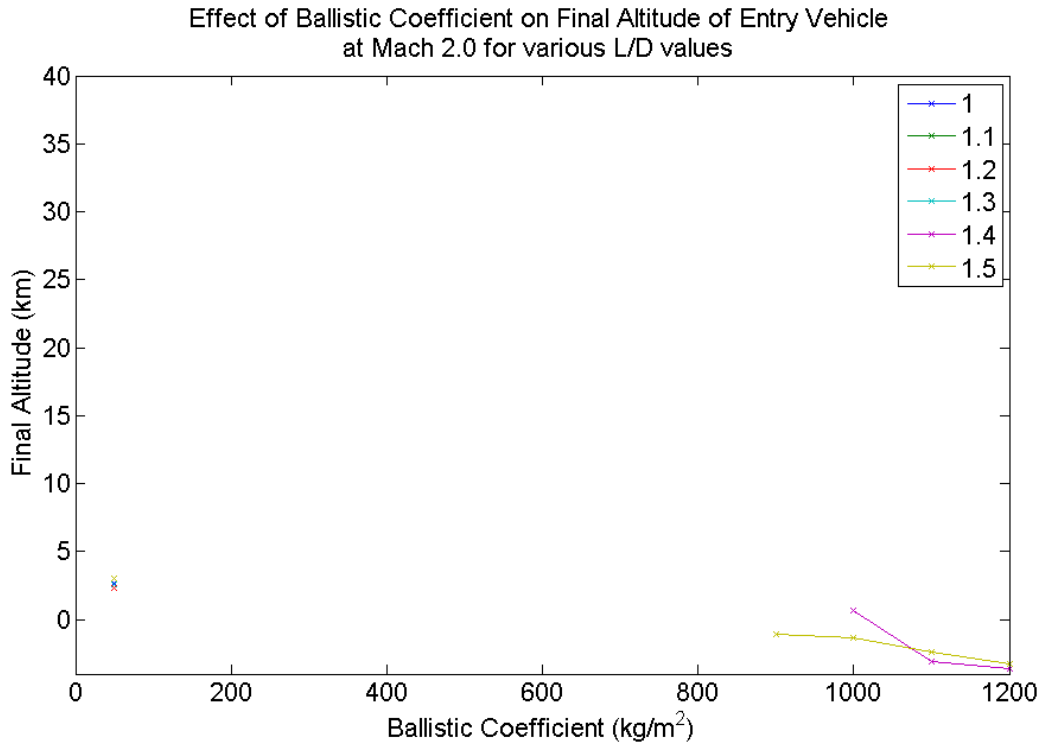

Figure 39: Sensitivity of Mach 2 Altitude to Ballistic Coefficient and L/D for L/D range of 1.0 to 1.5

3.3 EDL Parametric Analyses

With insight gained into the speeds of vehicles with certain characteristics close to the surface, we can begin adding in the last part of the vehicle's trajectory: propulsive descent. Although the tool developed for this study is capable of handling any combination of two factors to study their effect on landed mass performance, to allow for easier comparison between different analyses, one factor was kept as the vehicle's entry mass in each case. Note that the parameters that are fixed for a particular analysis are printed on the results graph with abbreviated labels. A list of the abbreviations has been provided in Table 1.

In all analyses, the range used for entry mass was 40 – 120 metric tons. The lower end of the range represents the maximum mass that the Ares V launch vehicle being developed for lunar and Martian exploration is capable of inserting into Martian orbit (21). The mass at the upper end of the range would be achieved with multiple stages boosting the vehicle towards Mars. Using the Ares V capability to determine the range of the graphs serves to illustrate the landed mass performance that can be achieved by this vehicle which is being developed for human exploration missions.

3.3.1 Trend with Lift-to-Drag Ratio Variation

One of the first factors to consider with regards to the landed mass performance of an EDL vehicle is its lift-to-drag ratio. Several studies have proposed the use of biconic vehicles for entry purposes due to good maneuverability characteristics that are desirable for precision landing (22) (23) (24). The good maneuverability of biconic vehicles can be attributed to high L/D values (22), which may also enhance the vehicle performance. The reason for this is that a vehicle with higher L/D stays in the upper atmosphere for a longer time period, thus bleeding off more energy at higher altitudes and having reduced speed at lower altitudes as seen in Section 3.2. This can allow the use of more lightweight descent devices and hence improve landed mass performance.

In Figure 40, all vehicle characteristics are kept the same as the reference vehicle, except for L/D, which is varied for a range of entry masses. The relevant parameter values are listed on the figure. This simulation was carried out such that the propulsive descent part of the trajectory started at Mach 3 for each case. The reason for choosing Mach 3 as the descent start criteria is that the aeroshell must be separated from the entry vehicle in order to ignite the engines and start propulsive descent. This separation is often achieved using a drogue chute, and the operating limit for drogue chutes in the Martian atmosphere is around Mach 3 (8). In Figure 40, the propulsive descent initiation altitude is shown as being set to 70 km. This is done in order to ensure that the vehicle attains a speed of Mach 3 below this altitude in order to prevent the descent transition logic described in Section 2.1.6 from adjusting the transition point.

Note also that the L/D values used for this analysis represent the lower range of L/D values considered in the entry sensitivity analyses. The reason for this is that within this L/D range, vehicles represented by most data points reach a speed of Mach 3 before hitting the surface for the given entry orbit conditions. This ensures that most data points are feasible for the propulsive descent transition of Mach 3. The same analysis can be applied to higher L/D values. However, the entry orbit conditions would have to be adjusted as discussed in Section 3.2.2.

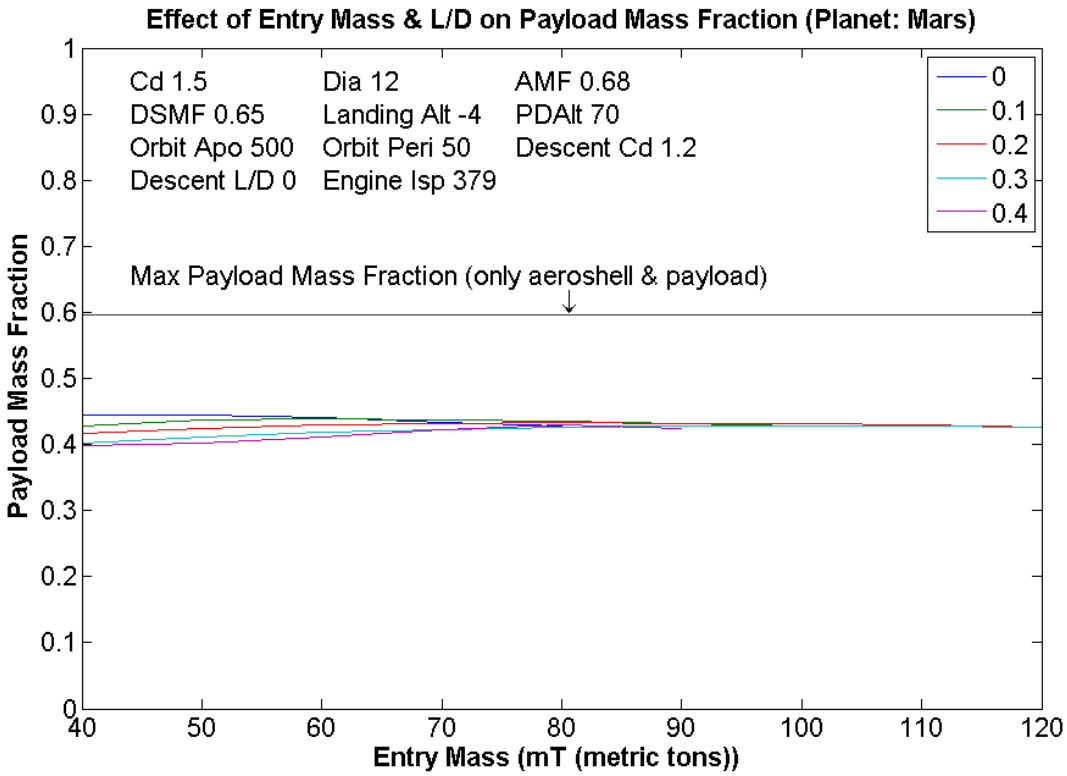

Figure 40: Effect of L/D on EDL vehicle performance for Mach 3 Propulsive Descent Initiation

From the figure, it seems that higher L/D results in a lower payload mass fraction, which is counterintuitive. Upon investigation, it was discovered that due to the higher altitude of the burn start, engine thrust was lowered and burn time was greatly increased, resulting in high propellant burn and thus, lower payload mass fraction (see Table 11 and Figure 41). In actuality, this can be remedied by optimizing the descent start time, but it provides interesting insights in that it highlights the need for a tradeoff between choosing a safe altitude for descent initiation and propellant conservation by later ignition.

Table 11: Detailed Results for Effect of L/D on EDL Vehicle Performance

Entry Mass (kg)	C_d	L/D	Payload Mass Fraction	Thrust/10^7 (N)	Descent Time (s)	PDI Altitude (km)
50000	1.5	0	0.4454	0.2704	48.5000	1.4961
50000	1.5	0.1	0.4365	0.1849	76.3500	4.1022
50000	1.5	0.2	0.4238	0.1538	101.0500	6.2986
50000	1.5	0.3	0.4115	0.1383	122.5000	8.1640
50000	1.5	0.4	0.4026	0.1305	137.4500	9.4064

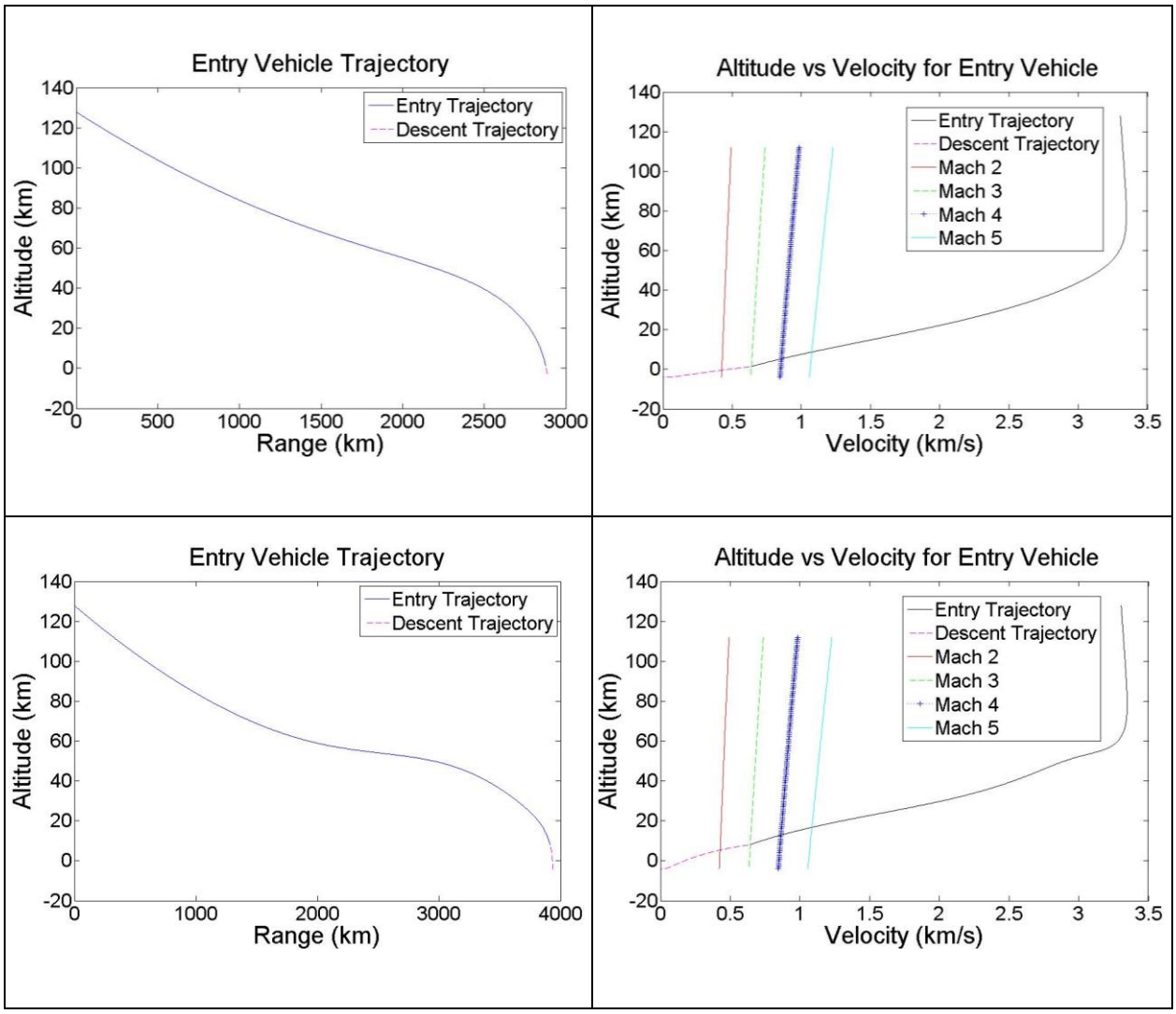

Figure 41: EDL Trajectories for Vehicle with Mass=50mT, Cd=1.5, Mach 3 Propulsive Descent Initiation. Top: L/D=0, Bottom: L/D=0.3

In order to further investigate this, the sensitivity analysis described above is repeated. This time, the propulsive descent initiation altitude is set to 5 km above the landing altitude. No Mach number limit is applied to aeroshell separation, and there is no mass penalty attached with propulsive separation. Figure 42 shows the results of this analysis & Figure 43 presents sample trajectories.

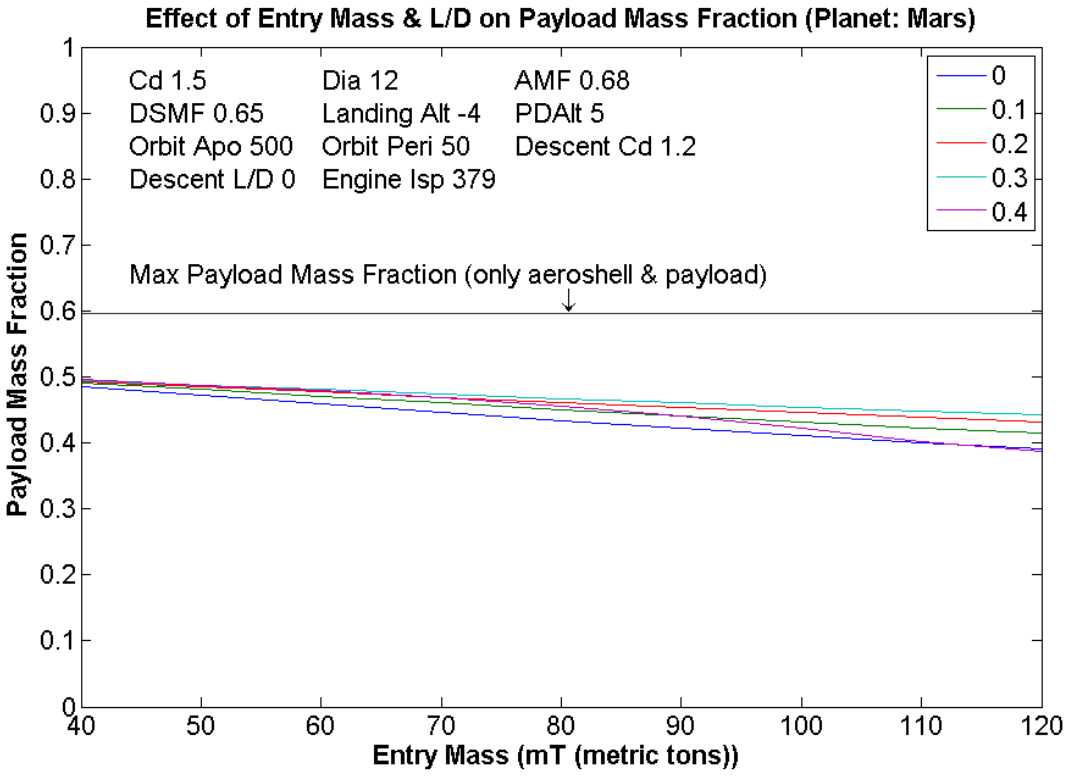

Figure 42: Effect of L/D on EDL Vehicle Performance for 5 km AGL Propulsive Descent Initiation Altitude, no mass penalty for propulsive aeroshell separation

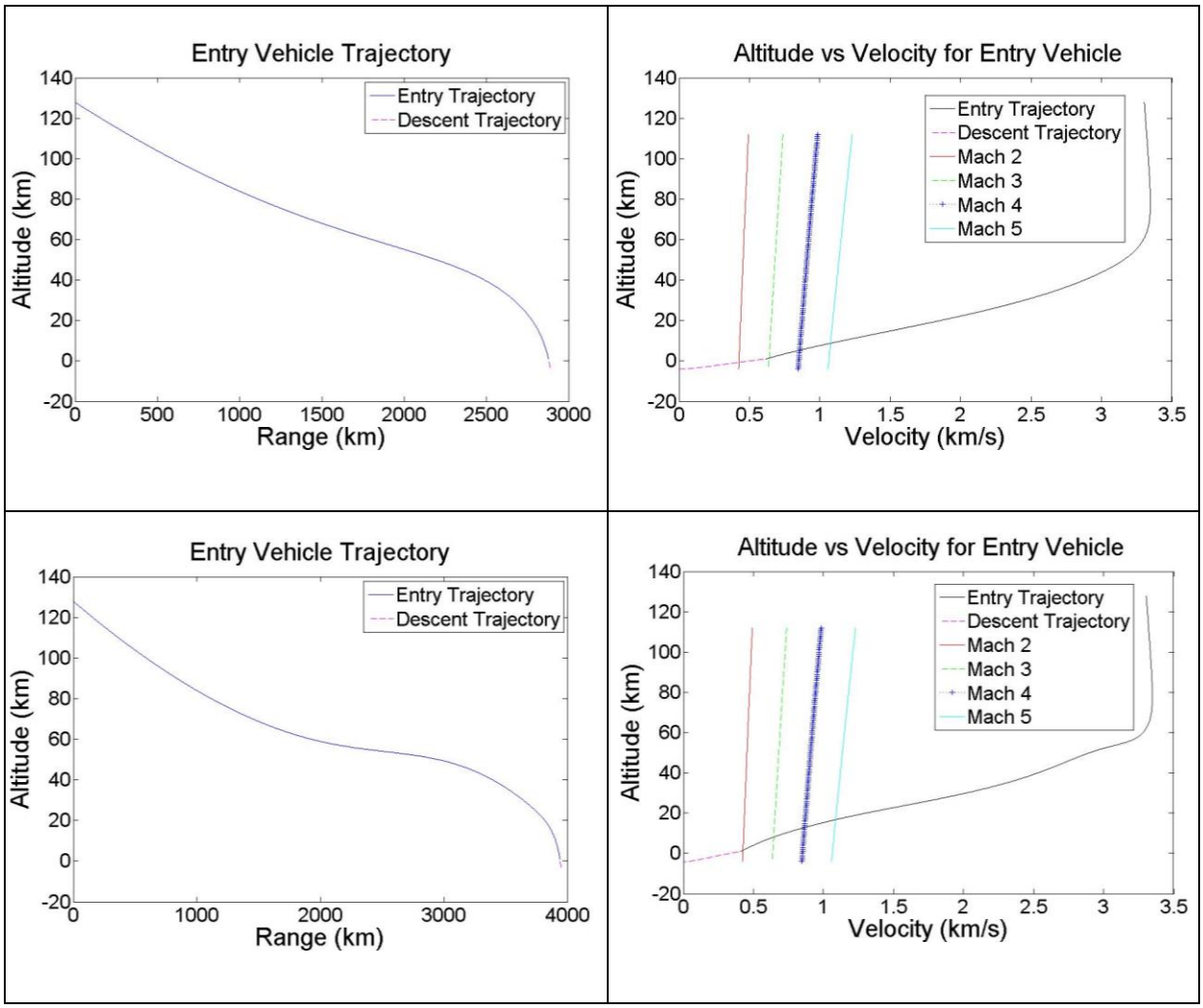

Figure 43: EDL Trajectories for Vehicle with Mass=50mT, Cd=1.5, 5 km AGL Propulsive Descent Initiation Altitude. Top: L/D=0, Bottom: L/D=0.3

From Figure 42, it can be seen that once the PDI altitude is fixed to a certain value, the higher L/D vehicles indeed perform better, as expected. This is due to the fact that a higher L/D vehicle bleeds off most of its energy in the upper atmosphere; therefore, it possesses lower velocity at the PDI altitude as compared to a lower L/D vehicle. Thus, it requires less propellant for the final descent phase and hence, has better landed mass performance. This suggests that a safe altitude rather than Mach number be used for PDI in order to improve performance. Note that the above analysis placed no limits on the Mach number at which descent began and also placed no mass penalty for propulsive aeroshell separation. Since the lower L/D vehicles are faster than the higher L/D vehicles, it is likely that vehicles traveling faster than Mach 3 at the specified PDI are the lower L/D ones. If a mass penalty is applied for propulsive aeroshell separation, this will only further reduce the performance of the low L/D vehicles. In that case, the generality of the above results will be maintained. To confirm this, Figure 44 shows the results of the same analysis but with a propulsive aeroshell separation mass fraction penalty of 0.1.

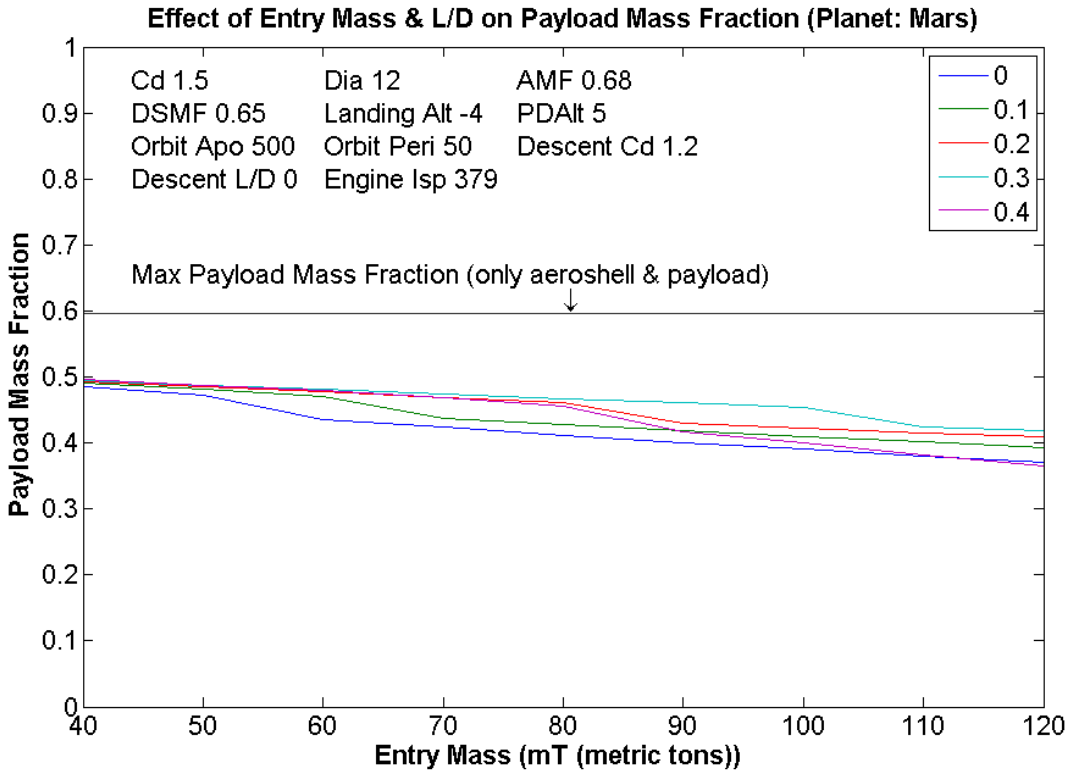

Figure 44: Effect of L/D on EDL Vehicle Performance for 5 km AGL Propulsive Descent Initiation Altitude, 10% mass penalty for propulsive aeroshell separation

3.3.2 Trend with Drag Coefficient Variation

The motivation behind this analysis comes from the fact that since the Viking era, the same general vehicle shape has been used for aerodynamic entry at Mars. Therefore, there is a need to investigate vehicles with different aerodynamic characteristics, including different drag coefficients. Several studies have proposed the use of biconic vehicles for entry purposes due to good maneuverability characteristics that are desirable for precision landing (22) (23) (24). The good maneuverability of biconic vehicles can be attributed to high L/D values (22), which also enhance the vehicle performance as seen in the last section. However, a survey of biconic shape aerodynamics suggests (See Appendix E: Biconics Overview) that many of them have low drag coefficient values from 0.5 to just above 1. It is unclear whether such a low drag coefficient is sufficient for effective landed mass performance at Mars since, as seen in Section 3.2, the corresponding high ballistic coefficient can result in a significantly lower deceleration achieved at the chosen PDI altitude.

Figure 45 shows the results of the analysis of landed mass performance sensitivity to drag coefficient. All other parameters are kept the same as the reference vehicle and the PDI altitude is set to 5 km above the landing altitude. No Mach number limit is applied to aeroshell separation and there is no mass penalty attached with propulsive separation. (Note: In Figure 46 and Figure 45, Descent Cd refers to the drag coefficient of the vehicle in descent configuration i.e. without the aeroshell.)

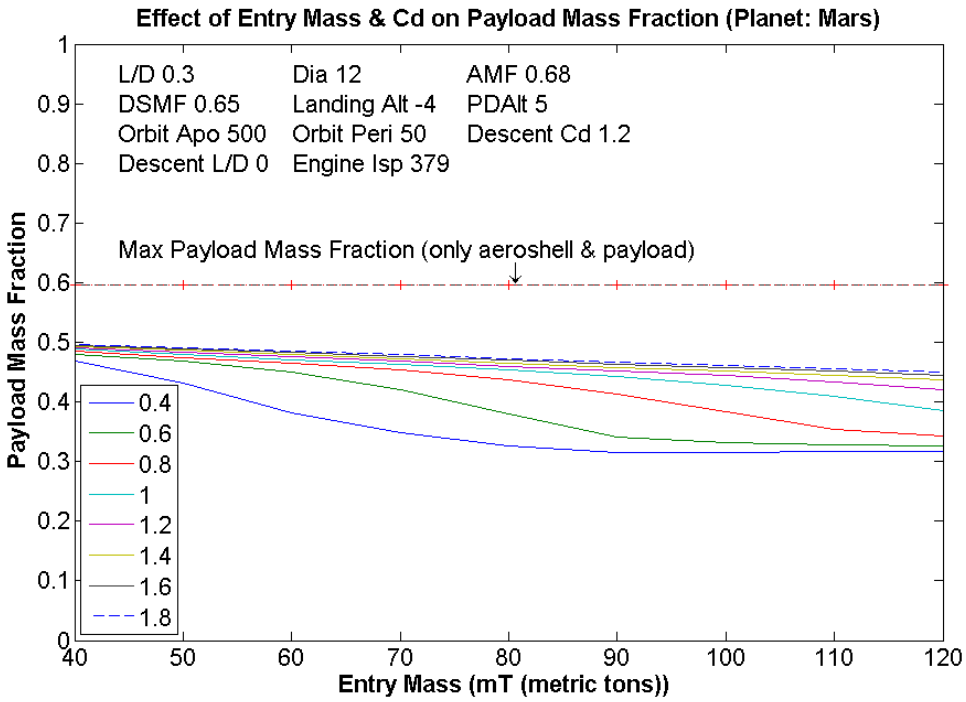

Figure 45: Effect of Drag Coefficient on EDL Vehicle Performance for 5 km AGL PDI Altitude, no mass penalty for propulsive aeroshell separation

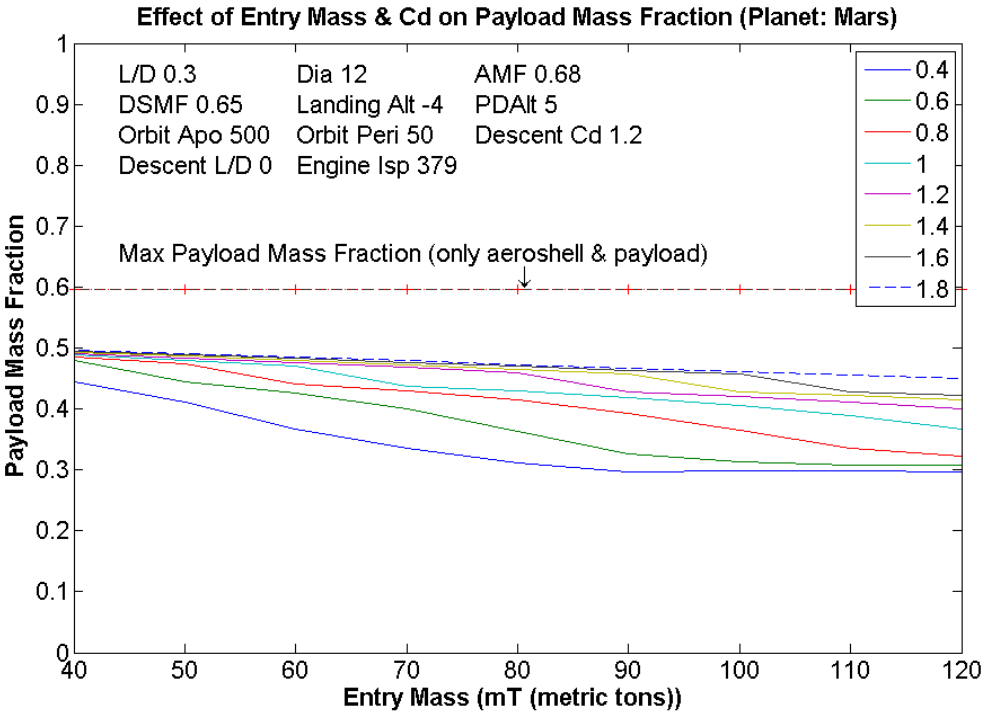

Figure 46: Effect of Drag Coefficient on EDL Vehicle Performance for 5 km AGL Propulsive Descent Initiation Altitude, 10% mass penalty for propulsive aeroshell separation

As expected, the vehicles with higher drag coefficient perform better and there seems to be a greater advantage at higher entry masses. Again note that for this analysis, no penalty was applied for propulsive aeroshell separation. Vehicles with lower drag coefficients are expected to require propulsive separation due to higher speeds at PDI altitude compared to high drag vehicles. Thus, their performance would be even lower (compared to high C_d vehicles) if a propulsive aeroshell separation mass penalty was applied and the generality of the above conclusion would hold. Figure 46 which shows the results of the drag coefficient sensitivity analysis with a propulsive aeroshell separation penalty of 0.1 confirms this.

At lower entry masses, more detailed analysis might be needed to gauge the marginal benefit gained by increasing drag coefficient. However, at higher entry masses, a high drag coefficient seems to provide significant benefits.

3.3.3 Trend with Propulsive Descent Initiation Altitude

As mentioned in Section 3.3.1, Propulsive Descent Initiation (PDI) altitude is an important factor in determining the performance of an EDL vehicle. In this section, results are presented for a case with fixed aerodynamics but varying PDI altitude. In this study, in general, altitude is measured from mean planet radius. But PDI altitude is referenced from Landing Altitude i.e. it represents the height above ground level (AGL).

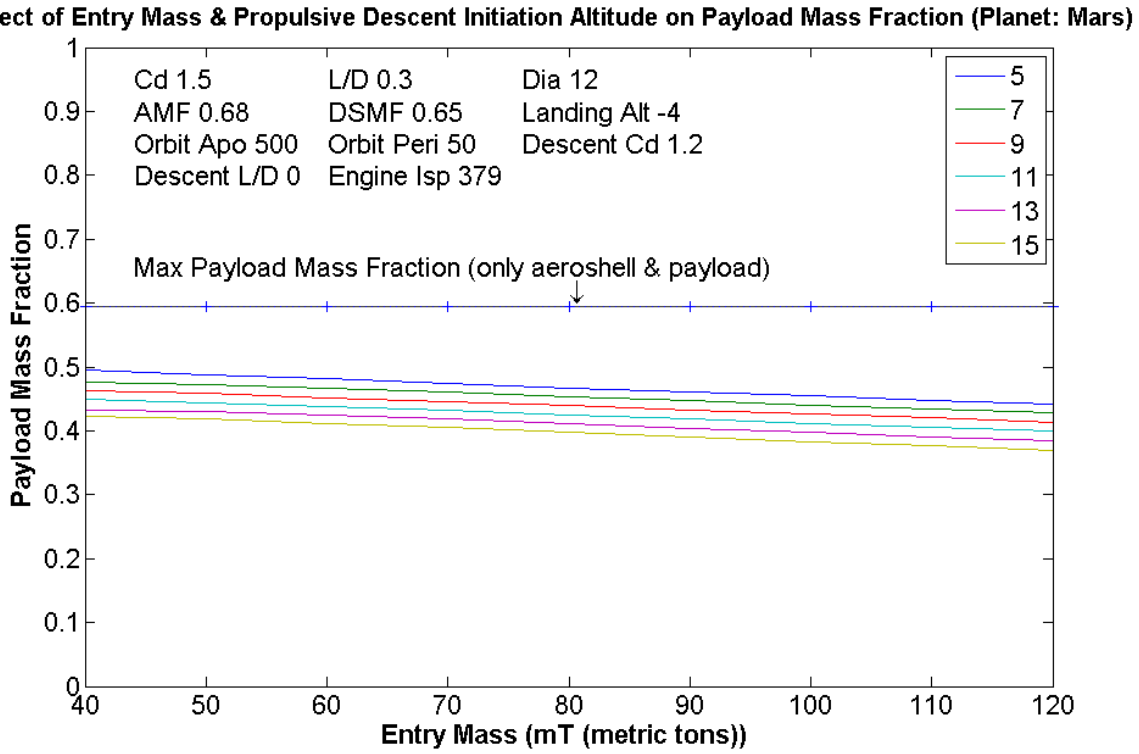

Figure 47: Effect of Propulsive Descent Initiation Altitude on EDL Vehicle Performance, no mass penalty for propulsive aeroshell separation

Figure 47 and Table 12 show that the earlier conclusion of higher PDI altitude resulting in lower landed mass performance was indeed correct. Note that this analysis did not apply a Mach number limit to PDI and there was no penalty attached with propulsive aeroshell separation. Figure 48 shows the results of the analysis repeated with a propulsive separation mass penalty of 0.1. The conclusion is the same. For an EDL vehicle using propulsive means for final descent, PDI altitude is an important factor in determining the landed mass performance of the vehicle. The reason for this is that a higher PDI altitude results in significant gravity loss due to the greater engine burn time and lower thrust needed for the gravity turn maneuver for final descent. This implies that a safe PDI altitude or engine burn time that may be required for obstacle avoidance, etc. needs to be traded off against the landed mass performance of the vehicle.

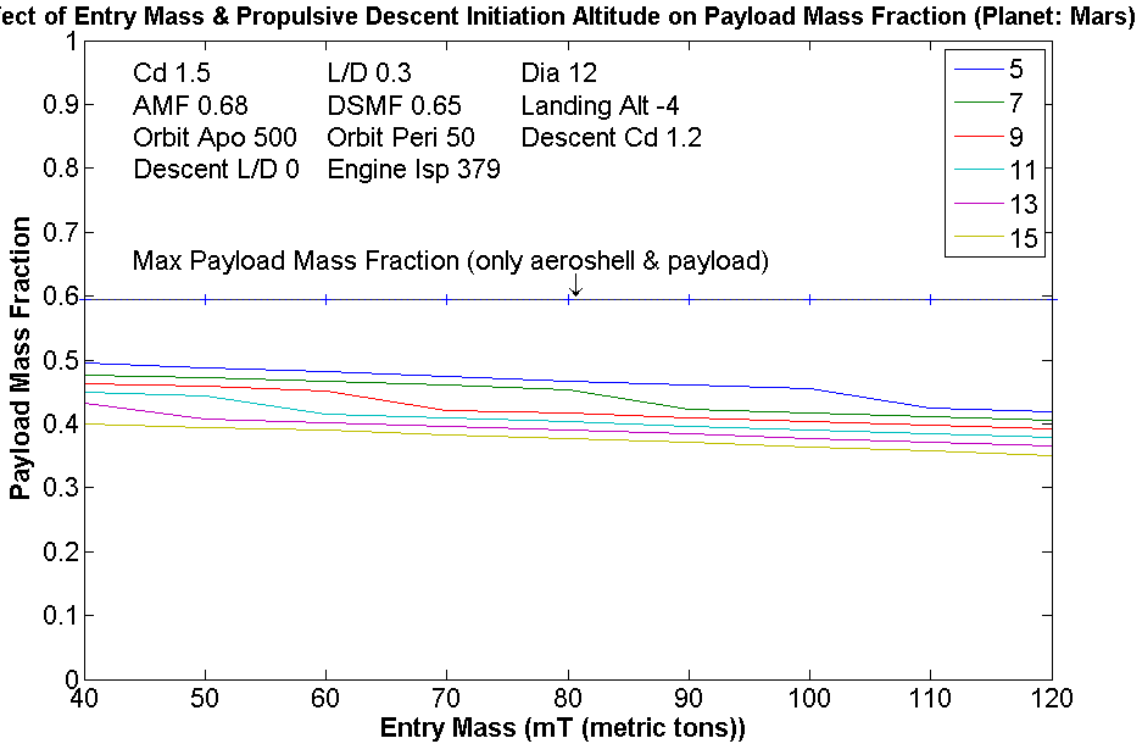

Figure 48: Effect of Propulsive Descent Initiation Altitude on EDL Vehicle Performance, 10% mass penalty for propulsive aeroshell separation

Table 12: Detailed Results for Effect of Propulsive Descent Initiation Altitude on EDL Vehicle Performance

PDI Altitude	Entry Mass	Cd	L_D	Payload Mass Fraction	Thrust/1000000 (N)	Descent Time (s)
15	50000	1.5	0.3	0.4191	0.1460	135.9000
13	50000	1.5	0.3	0.4295	0.1460	127.9000
11	50000	1.5	0.3	0.4431	0.1538	111.4500
9	50000	1.5	0.3	0.4588	0.1694	90.8000
7	50000	1.5	0.3	0.4727	0.1849	74.6500
5	50000	1.5	0.3	0.4873	0.2160	56.3000

3.3.4 Trend with Diameter

In Section 3.2, it was seen that the ballistic coefficient of a vehicle had a significant effect on the altitude reached for a particular Mach number. As the vehicle base diameter is a significant part of the ballistic coefficient, its effect on the landed mass performance for the EDL profile must be studied. Figure 49 shows the results of a sensitivity analysis for the vehicle base diameter. The PDI altitude was set to 5 km and no mass penalty was applied for propulsive aeroshell separation. The diameter range for this analysis was set using information about the Ares V launch vehicle. Initially, the Ares V Earth Departure Stage diameter was 8 m so this was chosen as the lower end of the range for sensitivity analysis (10). The upper end is 16 m, as final stage to fairing diameter ratios are not usually above 1.5 (25) and the current Ares V stage diameter is 10 m (26).

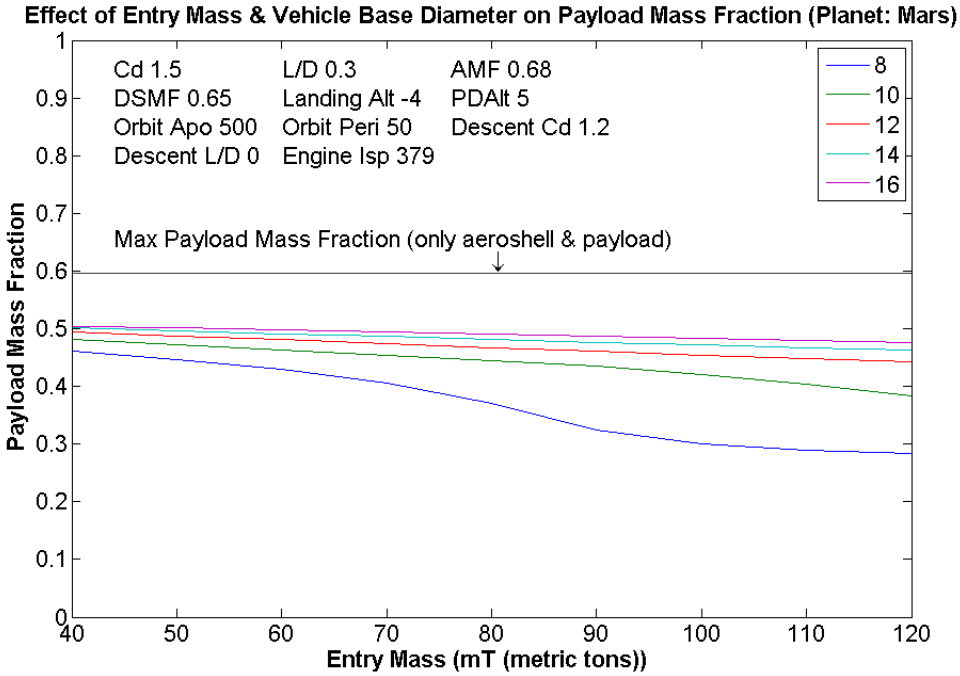

Figure 49: Effect of Vehicle Base Diameter on EDL Vehicle Performance for 5 km AGL PDI, no mass penalty for propulsive aeroshell separation

As expected, the diameter has a very significant impact on landed mass performance, especially for higher entry masses. From Figure 49, the benefit gained from a larger diameter seems to diminish at the higher values of the diameter. This is because the vehicle diameter actually affects the drag as a term in the frontal area calculation. Comparison of the frontal area for one data point with the next one shows that with increasing diameter, the percentage change in area decreases e.g. $10^2/8^2 = 1.56$ and $12^2/10^2 = 1.44$[‡]. Therefore, at larger diameters, the marginal benefit also diminishes. Still, in general, the sensitivity analysis results presented above show that increasing the vehicle base diameter has the potential to contribute significant benefits in terms of landed mass performance. However, this parameter is also one that will be subject to significant constraints due to launch vehicle shroud size. In this regard, the development of deployable aeroshells could enable improvements in landed mass performance.

3.3.5 Trend with Aeroshell Mass Fraction

For completeness, a sensitivity analysis with varying aeroshell mass fraction is also conducted. From Eq 22, it can be seen that the aeroshell mass fraction is used to calculate final performance of the vehicle. It is also used to set the vehicle mass at the start of propulsive descent. The graph presented in Figure 50 illustrates its importance as a factor in the landed mass performance of a Mars EDL vehicle. Creating a lightweight structure will have an immense impact on the landed mass performance at Mars and represents an important area for research and development efforts.

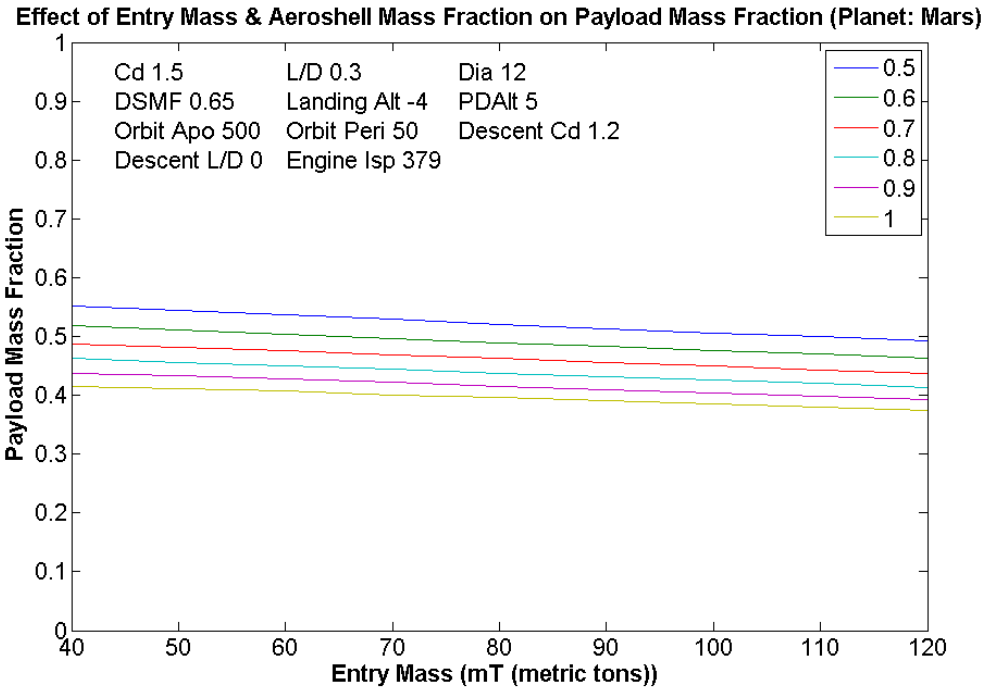

Figure 50: Effect of Aeroshell Mass Fraction on EDL Vehicle Performance, no mass penalty for propulsive aeroshell separation

‡ The formula for area using diameter is: $\pi*D^2/4$. In the example, only the D^2 terms are shown since the constant terms cancel out in the ratio.

3.3.6 Trend with Landing Site Elevation

Finally, the capability of a Mars EDL vehicle to reach a specific landing site needs to be assessed. There are interesting sites at higher elevations on Mars (27), and there exists concern about the capability to reach these compared to low-lying sites. Figure 51 shows the elevation distribution of the surface area of Mars and the landing site elevations for past Mars missions. It can be seen that each mission to date has extended the capability of accessing the Martian surface through landing at a higher elevation than previous missions. Since the landing site for a crewed mission has yet to be decided, a sensitivity analysis on the effect of landing site elevation on the vehicle performance may provide insights to aid this choice.

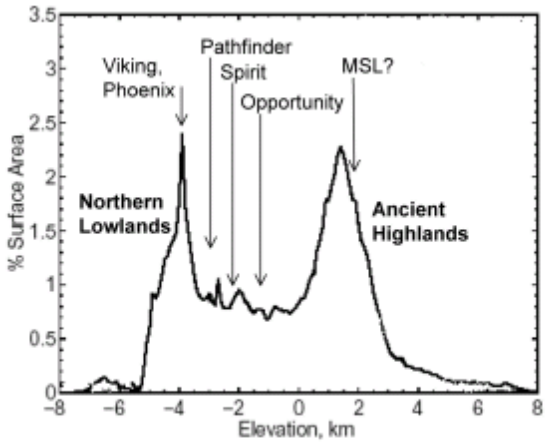

Fig. 4 Mars elevation area distribution.

Figure 51: Mars Elevation Area Distribution (29)

Figure 52 shows the results of this analysis with a PDI altitude of 5 km AGL. It is evident that the landing site elevation has a significant impact on the landed mass performance especially at higher masses. As expected, a low-lying landing site results in increased landed mass performance. The reason for this is that raising the landing site elevation raises the PDI altitude in the planet reference frame. At a higher PDI altitude, the vehicle is faster at the initiation of the propulsive stage and thus requires a greater amount of propellant to slow down.

Figure 53 also shows the results of the landing site elevation sensitivity analysis, this time with a penalty applied for PDI above Mach 3. This further lowers the landed mass performance for higher landing site elevations.

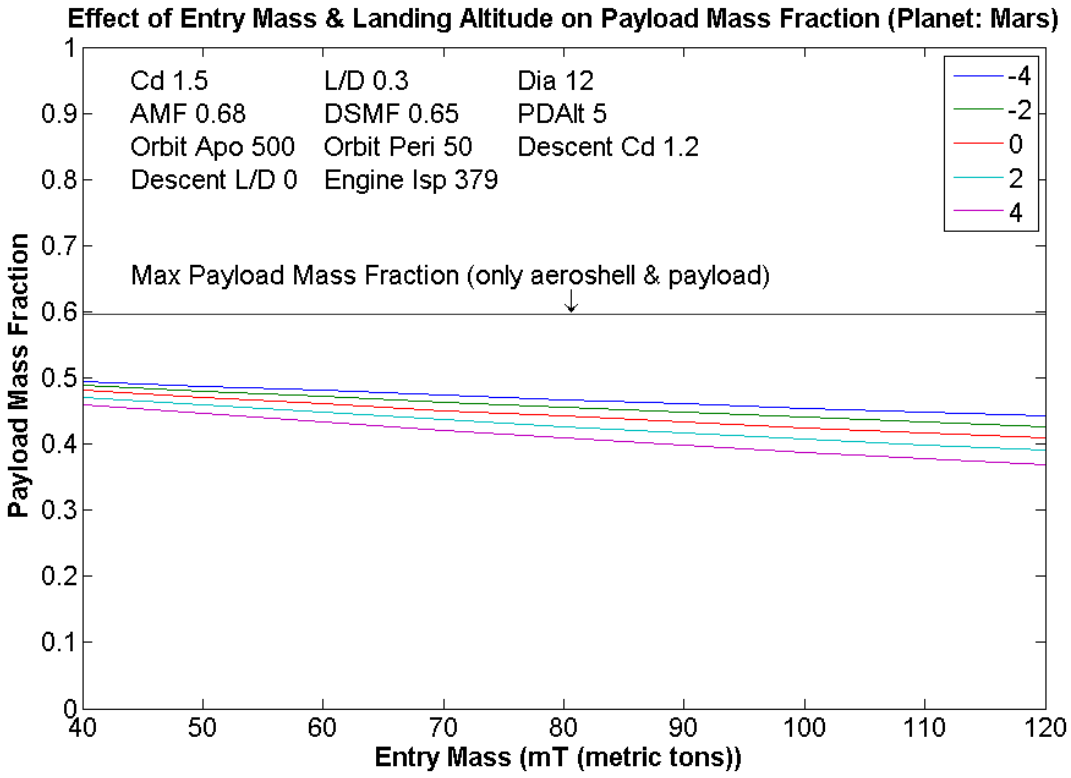

Figure 52: Effect of Landing Site Elevation on EDL Vehicle Performance, no mass penalty for propulsive aeroshell separation

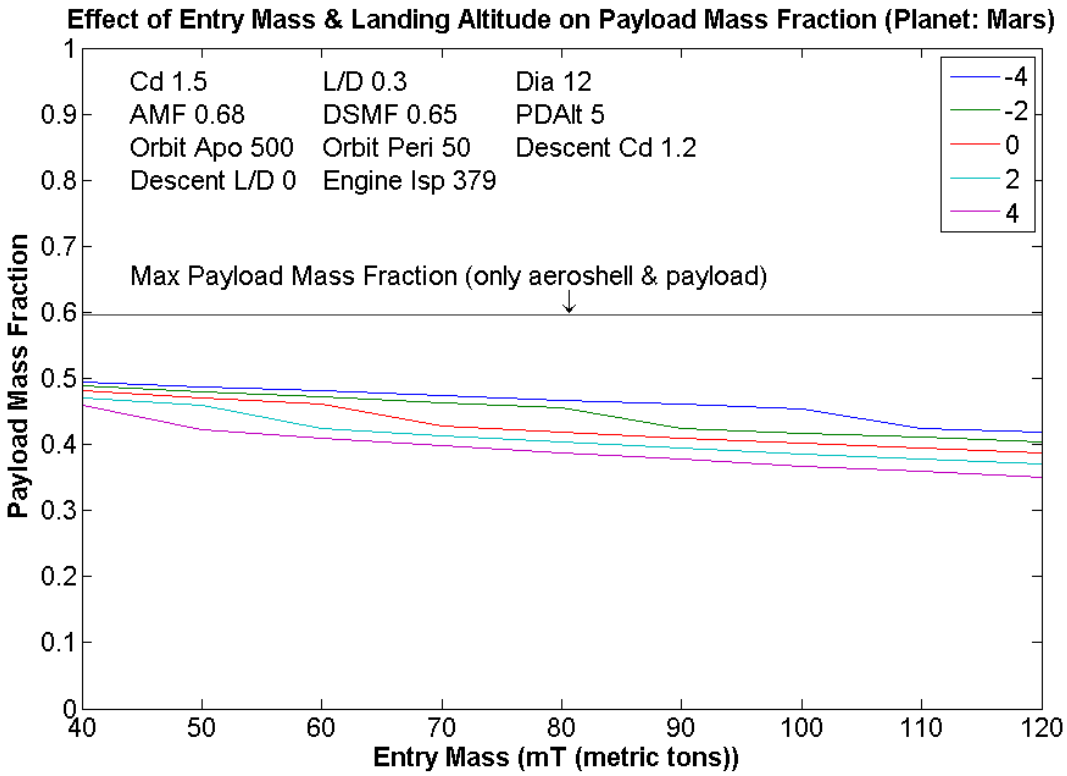

Figure 53: Effect of Landing Site Elevation on EDL Vehicle Performance, 10% mass penalty for propulsive aeroshell separation

This analysis implies that the choice of landing site for a crewed mission will have to be traded against the size of surface elements that need to be used at that particular site and thus, may limit the choice of landing sites to relatively low-lying ones. However, this might not be a show-stopper, since a large portion of the Martian surface is relatively low-lying, as is evident from Figure 51.

3.4 EDL Vehicle Design Insights

From the sensitivity analyses presented in the previous section, a number of design insights have been gleaned for a Mars EDL vehicle. These are summarized below:

- Before beginning the study, it was expected that improving the vehicle aerodynamic characteristics would be extremely important in improving the landed mass performance of the vehicle. However, although increasing drag coefficient and L/D does improve performance, optimizing the PDI altitude appears to hold more promise for increasing the landed mass performance.
- By far, the greatest impact on landed mass performance is that of the aeroshell mass fraction. Lightening the vehicle structure has the greatest potential to improve vehicle performance.
- Choosing a large diameter aeroshell within the limits of the launch shroud or using an inflatable aeroshell should also result in a significant performance improvement.
- Choosing a low-lying landing site can also greatly improve performance. Therefore, the choice of the landing site may in part be dictated by the mass of the mission elements that are needed for surface operations and the performance of the EDL system.
- After considering the above factors, increasing the drag coefficient and L/D would help further increase performance.
- For high L/D vehicles, the choice of entry angle is critical to ensure effective deceleration in the atmosphere.

One point to note is that for this study, active control of lift vector direction was not used. If this technique is employed, it may further improve the performance of high L/D vehicles.

4.0 Detailed Design Example

With an understanding of the sensitivity of EDL vehicle landed mass performance to a number of vehicle and mission design parameters, the reference vehicle shape described in Section 3.1 is investigated in detail for application to a crewed Mars mission. In particular, its landed mass performance is assessed.

For landed mass performance analyses, a number of design parameters are required. The values used for these and the rationale for choosing them is as follows:

- *Aerodynamic Characteristics*
  The drag coefficient and L/D values used in the analyses are 1.5 and 0.3, respectively as obtained in Section 3.1.
- *Mass*
  From research done in the author's research group at MIT (4), it is estimated that it is possible to inject 65 metric tons into a trans-Mars trajectory using two Ares V launches. If three launches are used, this mass increases to 105 metric tons[§]. Upon reaching Mars, the vehicle uses aerocapture to achieve orbit. Some of the vehicle mass is allocated to the propulsion system required for circularization of the aerocapture trajectory and subsequent deorbit for EDL. Therefore, the two entry masses used for analysis in this section are 62.5 and 100 metric tons.
- *Vehicle Base Diameter*
  The vehicle base diameter is set to 12 m. Since the current Ares V Earth Departure Stage diameter is set to 10 m (26), it is envisioned that a launch shroud capable of accommodating a 12 m diameter EDL vehicle is feasible.
- *PDI altitude*
  From Table 12, it can be seen that the descent time for a PDI altitude of 5 km is less than one minute for a 50 mT vehicle with moderate lift. In order to provide sufficient time for aeroshell separation and engine start, a PDI altitude of 15 km is chosen for this analysis for both entry masses in question. It should be noted however that this parameter has a significant effect on performance, as seen earlier, and must be optimized for the case in question.

Two major parameters now left are the landing site elevation and the aeroshell mass fraction and these will be varied for sensitivity analysis. The reason for this is that the landing site for a crewed Mars mission has yet to be chosen and the size of the payload that the EDL system can deliver to the surface may limit this choice. Therefore, the current analysis may help inform this choice. As for the aeroshell mass fraction, there is relatively little knowledge about how much the large structure required for crewed missions will weigh and how it will scale as compared to the much smaller structures for robotic missions. In this case, therefore, the best approach is a sensitivity analysis.

Figure 54 and Figure 55 show the sensitivity of landed mass performance for the two masses under consideration and Table 13 shows the detailed results for data points of interest. There is no Mach number limit applied to any of the sensitivity analyses and a mass penalty of 10% for propulsive separation is applied for aeroshell separation above Mach 3.

---

§ For details on the source of these numbers, see Appendix G. Note also that these numbers are different than the performance numbers cited in Section 3.3 which were based on an earlier study.

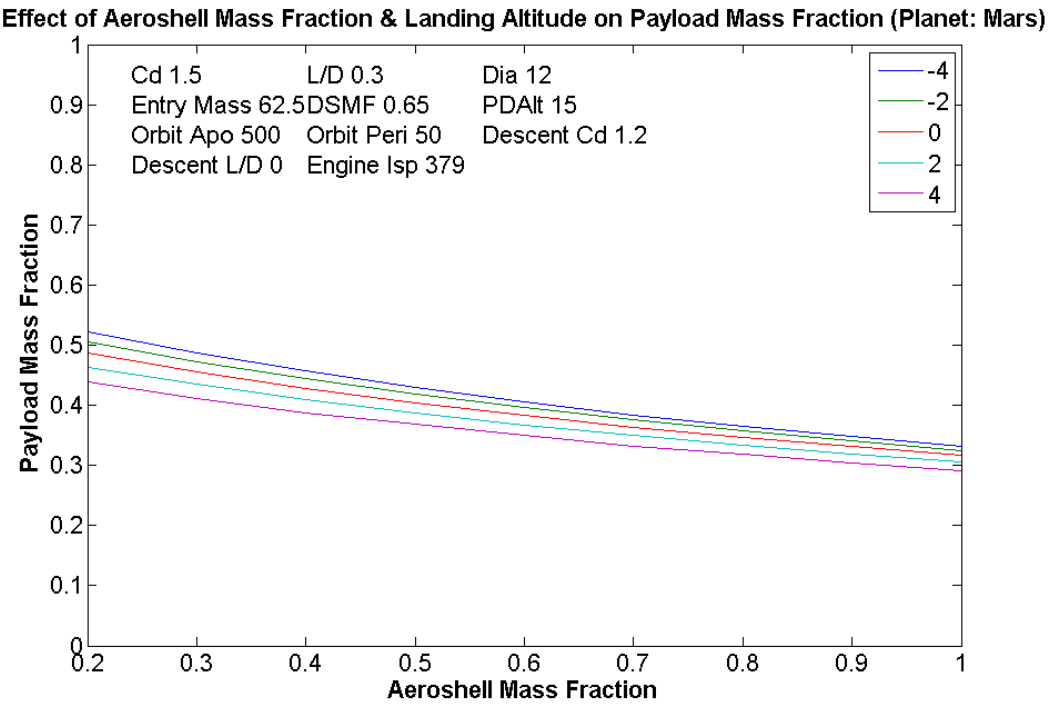

Figure 54: Effect of Aeroshell Mass Fraction and Landing Site Elevation on Reference Vehicle Payload Mass Fraction, Mass = 62.5 mT, 10% mass penalty for propulsive aeroshell separation

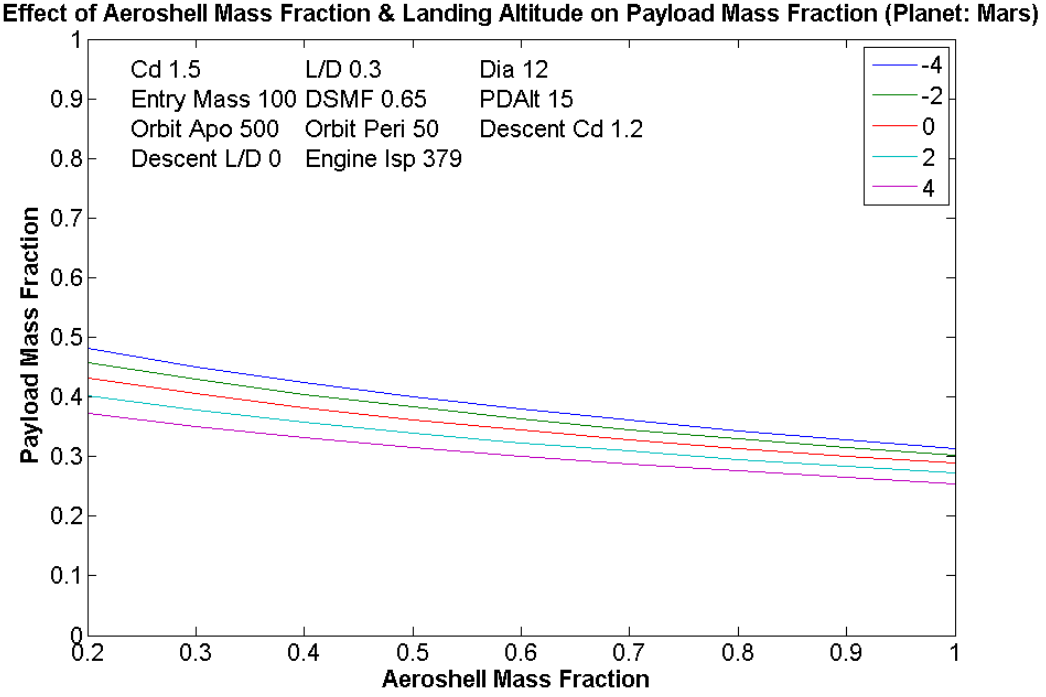

Figure 55: Effect of Aeroshell Mass Fraction and Landing Site Elevation on Reference Vehicle Payload Mass Fraction, Mass = 100 mT, 10% mass penalty for propulsive aeroshell separation

Table 13: Reference Vehicle Shape Performance Results

Entry Mass (mT)	Aeroshell Mass Fraction	Landing Site Elevation (km)	Payload Mass Fraction	Landed Mass (mT)	Marginal Benefit
62.5	1.0	4	0.2919	18.24	Datum
62.5	1.0	2	0.3058	19.11	4.77 %
62.5	1.0	-4	0.3325	20.78	13.93 %
62.5	0.5	4	0.3679	22.99	26.04 %
62.5	0.5	2	0.3874	24.21	32.73 %
62.5	0.5	-4	0.4295	26.84	47.15 %
100	1.0	4	0.2548	25.48	Datum
100	1.0	2	0.2728	27.28	7.06 %
100	1.0	-4	0.3142	31.42	23.31 %
100	0.5	4	0.3155	31.55	23.82 %
100	0.5	2	0.3397	33.97	33.32 %
100	0.5	-4	0.4002	40.02	57.06 %

As expected, both the landing altitude and the aeroshell mass fraction have a very significant effect on landed mass performance. From work done in the author's research group (4), a first estimate of the aeroshell mass has been obtained and corresponds to an aeroshell mass fraction of 0.5. Using this number and assuming that the vehicle lands at an elevation of 4 km, a payload mass fraction of 0.37 is obtained for a 62.5 metric ton vehicle and 0.3155 for a 100 metric ton vehicle even when a mass penalty for propulsive separation is applied. This translates to landed masses of 22.99 and 31.55 metric tons respectively.

However, since the aeroshell mass fraction of 0.5 is a first estimate, it is necessary to look at landed mass performance for other values of the aeroshell mass fraction. For the 62.5 metric ton case, the worst performance achieved during this analysis for the combination of the highest landing site elevation and aeroshell mass fraction is a payload mass fraction of approximately 0.3. This translates to a landed mass of 18.24 metric tons. For the 100 metric ton case, the worst performance is a payload mass fraction of 0.2548 which translates to a landed mass of 25.48 metric tons. While these numbers seem very low, the aeroshell mass fraction of 1.0 used is assumed to be very conservative.

Therefore, using the reference vehicle shape and a very conservative estimate for aeroshell mass, it is possible to deliver a significant payload of at least 18.24 metric tons of payload even to the highest landing site elevation of 4 km with two Ares V launches. This indicates that, from a landed mass perspective, a crewed Mars mission is feasible using the moderate lift reference vehicle, since the minimum landed mass requirements for such a mission are on the order of 20 to 30 metric tons as discussed in Section 1.0.

In addition to the high landing site elevation of 4 km, two other landing site elevations are of great interest: 2 km and – 4 km. These represent the peaks in the surface area distribution plot shown in Figure 51. From Table 13, it can be seen that the landed mass performance can be increased significantly by choosing one of the lower landing site elevations. For the aeroshell mass fraction of 1.0, an increase of upto 13% is possible by changing the landing site from + 4 km to – 4km for the 62.5 metric ton case and of upto 23% for the 100 metric ton case.

5.0 Conclusions

For this study, a planetary EDL vehicle trajectory simulation tool was created which is capable of conducting both individual trajectory simulations and sensitivity analyses for a large number of vehicle and mission design parameters. To date, this tool has been validated with experimental and simulated data for Mars missions. The relationships used to model the EDL vehicle trajectory are suitable for an architecture-level study. Several improvements can be made to this tool, and a list of recommendations in the next section highlights important ones.

Several sensitivity analyses relevant to EDL for crewed Mars missions were conducted using this tool. From the analyses for only the entry phase of flight as well as the full EDL profile, several insights were gleaned and are listed in Section 3.4. Some specific recommendations with the potential to greatly improve landed mass performance are as follows:

- Low propulsive descent initiation altitude is extremely important in increasing the performance of the vehicle. However, this needs to be traded against the safe altitude and a long enough descent time for obstacle avoidance as well as other crew safety considerations. Research needs to be done to determine the actual requirements for safety, since a PDI altitude that is too conservative may significantly lower the landed mass performance.
- The aeroshell mass fraction has a very significant impact on performance. However, a literature review[**] indicates that only limited analysis has been performed in this area, in particular for blunt bodies significantly different in shape and size compared to the Viking entry body. Detailed design analysis needs to be conducted in this area to establish a reference value for the aeroshell mass fraction based on current structural and thermal technology. Additionally, structural and thermal technology that reduces the aeroshell mass fraction can provide a highly significant increase in performance. Therefore, this area should also receive increased attention for R&D activities.
- Landing site elevation has a significant effect on landed mass performance especially for higher entry masses. Therefore, the choice of a low-lying landing site affords the ability to land more massive elements. Although certain landing sites are more interesting from the scientific point of view, EDL considerations related to the mass of mission surface elements might limit the choices.

Apart from the above measures, choosing an EDL vehicle shape with both high drag coefficient and high L/D would further increase the landed mass performance of the vehicle. However, the impact is less pronounced as compared to the factors listed above.

In addition to the sensitivity analyses conducted over a wide range of design parameters, the performance of a reference EDL vehicle shape with a spherical forebody and a conical aftbody was analyzed in detail for application to a crewed Mars mission. For an entry mass of 62.5 mT (corresponding to two Ares V launches) and a very conservative aeroshell mass fraction of 1.0, the reference vehicle shape is capable of delivering a mass of approximately 18.24 mT to a landing site with a high elevation of 4 km. Using a more moderate aeroshell mass fraction of 0.5, this mass increases to

[**] Conducted by Wilfried Hofstetter.

23 mT. Finally, by choosing a low-lying landing site of – 4 km elevation, the landed mass performance of the reference EDL vehicle can be further increased to approximately 26 mT.

Therefore, by achieving a moderate aeroshell mass fraction and choosing a low-lying landing site, a significant payload can be delivered to the Martian surface with two Ares V launches. This indicates that, from a landed mass perspective, a crewed Mars mission is feasible using the moderate lift reference vehicle, since the minimum landed mass requirements for such a mission are on the order of 20 to 30 metric tons as discussed in Section 1.0.

6.0 Recommendations for Future Work

This section presents recommendations for future work related to expanding and improving the work presented in this thesis, as well as highlighting EDL mission and vehicle design research directions with the greatest potential to enhance EDL vehicle performance for crewed Mars missions.

The work done for this thesis may be expanded upon both in terms of further development of the EDL simulation tool as well as the analyses conducted using this tool. In terms of tool improvement, recommendations for important features are listed below:

- Aerocapture modeling should be added to find aerodynamic loading on the structure due to aerodynamic deceleration, as that may place more stringent requirements on the vehicle structure.
- The propulsion system sizing algorithm should be improved for more robustness and speed.
- The entire code must be evaluated to find ways to decrease runtime.
- Currently, the aeroshell mass fraction of the EDL vehicle is set to the same value regardless of vehicle size (i.e. mass or diameter). Therefore, an aeroshell structural scaling model needs to be incorporated into the model to improve model fidelity.
- Active control modeling should be added to the code for better results with high L/D vehicles.
- The code should be validated for other planetary bodies.

With improvements to the tool, the sensitivity analyses presented in this document should be repeated to see the effects of active control, etc. New variables such as different atmospheric models should also be used to determine their effect on vehicle performance.

Additionally, there is a need for research into propulsive descent safety requirements for crewed Mars missions to prevent over-conservatism in establishing the propulsive descent initiation (PDI) altitude for EDL vehicles since a high PDI altitude can significantly lower EDL vehicle landed mass performance. Another area of research with the potential to provide significant improvements in performance is structural and thermal technology that reduces the aeroshell mass of the vehicle.

7.0 References

1. **Bush, President G. W.** *President Bush Announces New Vision for Space Exploration Program.* Washington, D.C. : White House, 2004.

2. **NASA - National Space Science Data Center.** Viking 2 Lander. [Online] [Cited: May 04, 2008.] http://nssdc.gsfc.nasa.gov/nmc/masterCatalog.do?sc=1975-083C.

3. **Braun, Robert D. and Manning, Robert M.** *Mars Exploration Entry, Descent and Landing Challenges.* Big Sky, Montana : 2006 IEEE Aerospace Conference, March 2006. IEEEAC paper #0076.

4. **Crawley, E, et al.** *MIT Mars Architecture Update Presentation to NASA.* December 2007.

5. **Hoffman, S and Kaplan, D (editors).** *The Reference Mission of the NASA Mars Exploration Study Team.* Houston, TX : Johnson Space Center, 1997. NASA SP-6017.

6. **Drake, B. G. (editor).** *Reference Mission Version 3: Addendum to the Human.* Houston, TX : Johnson Space Center, 1998. NASA SP-6017-ADD.

7. **Cruz, Juan R and Lingard, J Stephen.** *Aerodynamic Decelerators for Planetary Exploration: Past, Present and Future.* AIAA Guidance, Navigation, and Control Conference and Exhibit, 2006. AIAA-2006-6792.

8. *Private communication with Dr. Juan Cruz of NASA Langley.*

9. **Draper-MIT Team.** *Concept Exploration and Refinement Study Final Report.* Cambridge, MA : 2005.

10. **NASA.** *NASA's Exploration Systems Architecture Study Final Report.* 2005. NASA-TM-2005-214062.

11. **Wells, G., et al.** *Entry, Descent and Landing Challenges of Human Mars Exploration.* Breckenridge, Colorado : 29th Anuual AAS Guidance and Control Conference Proceedings, 2006. AAS 06-072.

12. **Sierra Engineering Inc.** Program to Optimize Simulated Trajectories (POST). [Online] [Cited: May 04, 2008.] http://www.sierraengineering.com/Post3d/post3d.html.

13. **Astos Solutions.** ASTOS AeroSpace Trajectory Optimization Software. [Online] [Cited: May 04, 2008.] http://www.astos.de.

14. *Private communication with Wilfried Hofstetter.*

15. **Braun, Robert D.** Flight Mechanics. *Hypersonic Educational Initiative Lecture Notes.* National Institute of Aerospace, 2008.

16. **Planetary Atmospheres Node of the Planetary Data System (PDS).** PDS Atmospheres Data Set Catalog. [Online] [Cited: May 04, 2008.] http://pds-atmospheres.nmsu.edu/data_and_services/atmospheres_data/catalog.htm#Mars.

17. **Spencer, David A, et al.** *Mars Pathfinder Atmospheric Entry Reconstruction.* American Astronautical Society, 1998. AAS 98-146.

18. **Wright, Michael J.** Hypersonic Aerodynamics. *Hypersonic Educational Initiative Lecture Notes.* National Institute of Aerospace, 2008.

19. **Bonner, E. and Clever, W. and Dunn, K.** *Aerodynamic Preliminary Analysis System II Part I - Theory.* NASA, 1991. Contractor Report 182076.

20. **Sova, G. and Divan, P.** *Aerodynamic Preliminary Analysis System II Part II - User's Manual.* NASA, 1991. Contractor Report 182077.

21. **Wooster, Paul D.** *Strategies for Affordable Human Moon and Mars Exploration.* Massachusetts Institute of Technology, 2007. S.M. Thesis.

22. **General Electric Co. - Re-entry Systems Division.** *Generic aerocapture atmospheric entry study - Final Report.* Philadelphia, PA : 1980.

23. **Lawson, Shelby J.** *Mars Rover Sample Return Aaerocapture Configuration Design.* Reno, NV : 27th Aerospace Sciences Meeting, 1989. AIAA-1989-631 .

24. **Pennsylvania State University.** *Mars Sample Return Mission: Two Alternate Scenarios - Final Report.* 1992. NASA-CR-189970.

25. **Isakowitz, Steven J and Hopkins, Joshua and Hopkins Jr., Joseph P.** *International Reference Guide to Space Launch Systems.* Reston, VA : AIAA, 2004. ISBN: 978-1-56347-591-7.

26. **Sumrall, Phil.** Ares V Overview. *3rd Space Exploration Conference.* Denver, Colorado : 2008.

27. **NASA.** Landing Site List from Workshop, First Landing Site Workshop, 2006. *Mars Exploration Program Landing Sites.* [Online] [Cited: 05 15, 2008.] http://marsoweb.nas.nasa.gov/landingsites/msl/workshops/1st_workshop/docs/MSL_workshop_site_list.pdf.

28. **Jits, Roman and Wright, Michael and Chen, Y.-K.** *Closed-Loop Trajectory Simulation for Thermal Protection System Design for Neptune Aerocapture.* Journal of Spacecraft and Rockets, Vol. 42, No. 6, 2005. AIAA-13428-390.

29. **Braun, Robert D and Manning, Robert M.** *Mars Exploration Entry, Descent and Landing Challenges.* Journal of Spacecraft and Rockets, Vol. 44, No.2, 2007.

Appendix A: EDL Code User Manual

This section describes how to use the tool developed in this study. Before starting, the user should make sure that all files in the folder entitled "Matlab tool" on the accompanying CD-ROM have been copied into the Matlab working directory.

There are four separate versions of the tool developed in this study. The first two simulate single trajectories, one simulates only the entry portion of the flight and the second one simulates both the entry and propulsive descent. The third version simulates the effect of ballistic coefficient, L/D and entry conditions on the final altitude reached by the entry body at Mach numbers 4, 3 and 2. The final version conducts sensitivity analysis combining both the entry and propulsive descent trajectories and shows the effect of two variables on the performance of the vehicle.

The interface for all the programs is a command-line input system implemented in Matlab. The user needs to open the main file in Matlab and click the run button. Once this happens, the user simply answers the prompts in the command window. If the user wishes to see more details of what is happening inside the program, the code is viewable in Matlab and is commented. The manipulation of each of the programs is fairly straightforward. Therefore, in this appendix, for each program, only some of the more confusing aspects of the program are identified and explained.

For All Programs

Two major inputs are relevant to all versions of the program. The first one is the direction that the body is entering into the planet's atmosphere relative to the planet's sidereal rotational direction. This is simply specified through answering the first prompt that appears on screen after the program is run.

The second major input is the planetary data. Three files need to be provided to the program with planetary data in order for the program to run. The first is a text file containing values of planetary constants, one value on each line. ***There should be no extra white space at the end of the last value.*** The planetary constant values need to be specified in this order:

```
1-Mass of Planet in kg
2-Mean Planet Radius in km
3-Ratio of Specific Heats of Planetary Atmosphere (no units)
4-Gas Constant for Planetary Atmosphere in J/kg/K
5-Altitude of Atmosphere/Space Interface of Planet in km
6-Sidereal Rotation Period of Planet in hr
7-Maximum Mach number for Operation of Parachutes in Planetary
Atmosphere (no units)
```

Note that the extension of the text file also needs to be provided to Matlab at the input prompt. The other two files must be Matlab functions files (*.m) and must contain a model of the planet's atmosphere. The input for each file must be the altitude above the planet's surface in meters. One file must output the atmospheric density in kg/m^3 and the other must output the atmospheric temperature in Kelvin. In this case, only the function names must be supplied; a file extension is not needed.

If running simulations for Mars, this data has been included in the package of Matlab files. The planetary constants data file is named: marsconstants.txt and the atmospheric profile functions are: Marsdensityfitted and Marstempfitted.

Single Entry Trajectory Tool

Filename: SingleEntryTrajectorysim.m

Most of the inputs for this tool are very straightforward. The only one that requires explanation is the "trajectory calculation stop altitude" which is labeled "StopAlt" in the user interface. This allows the user to terminate the trajectory calculation at a certain altitude above the landing site. A very important point is that this altitude is the height "above ground level" i.e. specifying 5 km means 5 km above the landing site. If the landing site altitude is – 4 km, then the trajectory stop altitude in absolute terms is 1 km.

Single EDL Trajectory Tool

Filename: SingleEDLTrajectorysim.m

The initial command prompts are self-explanatory. After the choice of simulation variables, the next prompts need some explanation:

- The user is asked if there is an upper Mach number limit for propulsive descent start. If the answer is yes, the user is asked to specify this limit. This Mach number limit might be applied for any reason. If it is above the parachute operating limit for that particular planet, the program displays the mass penalty for propulsive aeroshell separation since a drogue chute cannot be used for aeroshell separation. The user has the option to alter this mass penalty. If the user does not wish to apply a mass penalty for propulsive aeroshell separation, this number should be set to zero.
- Note that if the Mach number limit is applied, the program might adjust the PDI altitude set by the user according to the transition logic described in Section 2.1.6.
- The last input required from the user is a deployment time for the propulsion system. This can be thought of as the time required to jettison aeroshell or start the engine. In the simulation, the vehicle coasts in the descent configuration during this time. Therefore, the mass and aerodynamics numbers for the descent configuration are used.
- Note also that the parameters listed as "Descent Cd" and "Descent L/D" are the aerodynamic coefficient values for the descent configuration i.e. for the vehicle without the aeroshell.

Entry Sensitivity Tool

Filename: EntryParametricAnalysisTool.m

This tool is essentially a Matlab script file that runs the entire analysis. The only two inputs are the ranges for two simulation variables: ballistic coefficient and lift-to-drag ratio or ballistic coefficient

and entry orbit apoapsis. The user has the option of selecting the combination of variables at the start of the program.

One issue with this tool is that for certain data points, the vehicle does not slow down to the required Mach numbers and hence those points are left out from the graphs. This can cause discontinuities in a data series for a particular lift-to-drag ratio for which only some of the ballistic coefficient values are infeasible but ones lower or higher are. At present, the program does not have the capability to detect this and plot the two parts of the series separately. The user might notice this problem if two consecutive data point markers are very far apart. In this case, the data will need to be properly plotted by the user.

EDL Sensitivity Tool

Filename: EDLtool.m

This tool conducts EDL sensitivity analyses for a large number of parameters. However, only two parameters can be varied for a particular run. The initial command prompts are self-explanatory. After the choice of simulation variables, the next prompts need some explanation:

- The user is asked if there is an upper Mach number limit for propulsive descent start. If the answer is yes, the user is asked to specify this limit. This Mach number limit might be applied for any reason. If it is above the parachute operating limit for that particular planet, the program displays the mass penalty for propulsive aeroshell separation since a drogue chute cannot be used for aeroshell separation. The user has the option to alter this mass penalty. If the user does not wish to apply a mass penalty for propulsive aeroshell separation, this number should be set to zero.
- Note that if the Mach number limit is applied, the program might adjust the PDI altitude set by the user according to the transition logic described in Section 2.1.6.
- The last input required from the user is a deployment time for the propulsion system. This can be thought of as the time required to jettison aeroshell or start the engine. In the simulation, the vehicle coasts in the descent configuration during this time. Therefore, the mass and aerodynamics numbers for the descent configuration are used.
- Note also that the parameters listed as “Descent Cd” and “Descent L/D” are the aerodynamic coefficient values for the descent configuration i.e. for the vehicle without the aeroshell.

Note that this tool only produces two plots showing the effect of the two variables on payload mass fraction and required thrust. However, other data is available in workspace and can be manipulated by the user.

Appendix B: Matlab Code Listing

This section contains the code for all versions of the program described in Appendix A: EDL Code User Manual. This code can also be found on the attached CD-ROM in the folder entitled “Matlab files”. The following is a list of all the files included here (in alphabetical order). A number of them are sub-functions that are called from the main programs described in Appendix A: EDL Code User Manual.

- addblankline.m
- EDLsim.m
- EDLtool.m
- EntryParametricAnalysisPlotGeneratorLD.m
- EntryParametricAnalysisPlotGeneratorOrbit.m
- EntryParametricAnalysisTool.m
- Entrysim.m
- Entrysimballcoefffunc.m
- Entrysimonly.m
- legnum2.m
- num2cellstr.m
- PlanetEntryConditions.m
- PropulsiveDescentSim.m
- PropulsiveDescentSimFunc.m
- SingleEDLTrajectorysim.m
- SingleEntryTrajectorysim.m

For completeness, the code for functions describing the Martian atmosphere used in this study has also been included at the end of the above set of files. This can be used as a model for creating functions describing other atmospheres. The filenames for these functions are:

- Marsdensityfitted.m
- Marstempfitted.m

For clarity, the start and end of code for each file has been clearly marked on the following pages.

Filename: addblankline.m

***** Start of Code *****

```
spacestring=strvcat(' ');
disp(spacestring)
```

***** End of Code *****

Filename: EDLsim.m

***** Start of Code *****

```
function
[V_finalprop,h_finalprop,s_finalprop,v_initialprop,h_initialprop,s_initialpro
p,mass_final,fuelmass,payloadmass,payloadmassfraction,descenttime,thrustactua
l,Maxgtotal]=EDLsim(Cd,Dia,L_D,vehicleentrymass,aeroshellmassfraction,strucma
ssfraction,descentalt,landalt,entry_apo_alt,entry_peri_alt,Cdprop,L_Dprop,Isp
)



% global constant reading
global G Mplanet Rplanet kplanet Rgasplanet atmos_interface_planet
periodplanet angularrotplanet machparaoplimitplanet
global jettisonlimitcheck machjettisonlimit propdeploytime
propseparationmassfraction

global AtmDensityPlanetfunc AtmTempPlanetfunc


% adjustments
vehiclefrontalarea=pi()*(Dia^2)/4.0;
vehicleentrymass=vehicleentrymass*1000.0; %kg
landalt=landalt*1000;
descentalt=descentalt*1000;
entry_apo_alt=entry_apo_alt*1000;
entry_peri_alt=entry_peri_alt*1000;

% Aerodynamic Entry Simulation

Entrysim

%Vehicle Propulsive Descent parameters

propstartindex=i_final; % time at propulsive descent start


% If Propulsive Descent Initiation altitude is below altitude, do not run
% simulation & return infeasible values

if h(propstartindex) <= landalt

    V_finalprop = -9999;
    h_finalprop = -99999;
    s_finalprop = -99999;
    mass_final = -10;
    fuelmass = -10;
    payloadmass = -10;
```

```
    payloadmassfraction = -10;
    descenttime = -10;
    thrustactual = -10;
    Maxgtotal = -10;
    v_initialprop=-9999;
    h_initialprop=-99999;
    s_initialprop=-99999;


else


    if mach(propstartindex) <= machparaoplimitplanet % check if parachutes
can be used

        vehicledescentmass=vehicleentrymass/(1+aeroshellmassfraction); %mass
at start of propulsive maneuver
    else % mass penalty because of propulsive separation

        vehicledescentmass=vehicleentrymass/(1+aeroshellmassfraction+propsepa
        rationmassfraction); %mass at start of propulsive maneuver
    end


    % Descent Propulsion System Sizing Routine: using bisection search
algorithm
    % Objective: try to get vehicle as close as possible to desired landing
    % altitude with 1 m/s velocity
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A=0.01; % bisection search lower bound
    B=2.0; % bisection search upper bound
    C=(A+B)/2.0;
    count=1;
    tolerance=0.01;

    while abs(A-B)>tolerance
        count;

        fA=PropulsiveDescentSimFunc(A,vehicledescentmass,Cdprop,L_Dprop,Dia,I
        sp,Vx(propstartindex),Vy(propstartindex),h(propstartindex),s(propstar
        tindex),theta(propstartindex),x(propstartindex),y(propstartindex));

        fB=PropulsiveDescentSimFunc(B,vehicledescentmass,Cdprop,L_Dprop,Dia,I
        sp,Vx(propstartindex),Vy(propstartindex),h(propstartindex),s(propstar
        tindex),theta(propstartindex),x(propstartindex),y(propstartindex));

        fC=PropulsiveDescentSimFunc(C,vehicledescentmass,Cdprop,L_Dprop,Dia,I
        sp,Vx(propstartindex),Vy(propstartindex),h(propstartindex),s(propstar
        tindex),theta(propstartindex),x(propstartindex),y(propstartindex));

        if fC > landalt
            B = C;
            A = A;
```

```
        else
            A = C;
            B = B;
        end
        C=(A+B)/2.0;

        count=count+1;
    end

    throttleratiodesired=(A+B)/2.0;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % get required thrust from above routine

    % Propulsive Descent Simulation
    PropulsiveDescentSim


    % find maximum g's experienced by vehicle
    accel=sqrt(dvx_dt.^2+dvy_dt.^2);
    accelprop=sqrt(dvx_dtprop.^2+dvy_dtprop.^2);
    earthg=accel/9.80665;
    earthgprop=accelprop/9.8065;
    Maxg=max(earthg);
    Maxgprop=max(earthgprop);

    Maxgtotal=max([Maxg Maxgprop]);
end
```

***** End of Code *****

Filename: EDLtool.m

***** Start of Code *****

```
clear all
close all
clc


Planetname=input('Enter planet name for entry, descent and landing
simulation: ','s');

% planet constants

global G Mplanet Rplanet kplanet Rgasplanet atmos_interface_planet
periodplanet angularrotplanet machparaoplimitplanet

G=6.673e-11; % Universal Gravitational Constant

% Reading planet constants from file

planetconstantsfile=input('Enter name of text file (including extension)
containing the list of planet constants: \n','s');
planetconstants=dlmread(planetconstantsfile,'\n');

Mplanet=planetconstants(1); %Mass of Planet (kg)
Rplanet=planetconstants(2); % Mean Planet Radius (km)
kplanet=planetconstants(3); %Ratio of Specific Heats for Atmosphere
Rgasplanet=planetconstants(4);% Gas Constant for Atmosphere (J/kg/K)
atmos_interface_planet=planetconstants(5); %Atmospheric Interface Radius (km)
periodplanet=planetconstants(6); %Sidereal Rotation Period (hr)
machparaoplimitplanet=planetconstants(7); %Operational Limit for Parachutes
in Planetary Atmosphere



%planet constant adjustments & related calculations
Rplanet=Rplanet*1000.0;
periodplanet=periodplanet*60*60; %Sidereal Rotation Period (s)
rotdirection=input('Enter 1 for entering in direction of planet sidereal
rotation \nEnter -1 for entering in direction opposite to planet sidereal
rotation \n');
angularrotplanet=-rotdirection*2*pi()/periodplanet; %Sidereal Rotational
Speed (rad/s)


global jettisonlimitcheck machjettisonlimit propdeploytime
propseparationmassfraction

% Atmospheric data source specification
```

```
global AtmDensityPlanetfunc AtmTempPlanetfunc



AtmDensityPlanet=input('Enter the name for the Atmospheric Density Profile
function file (input=altitude in m)) \n','s');
AtmTempPlanet=input('Enter the name for the Atmospheric Temperature Profile
function file (input=altitude in m)) \n','s');

AtmDensityPlanetfunc=str2func(AtmDensityPlanet);
AtmTempPlanetfunc=str2func(AtmTempPlanet);

% Variable Declaration for Simulation Status
datapointcount=0;



%Default Variable Values
Cd=1.5; %Drag Coefficient
L_D=0.3; % Lift-over-Drag Ratio
Dia=12; %Vehicle diameter (m)
vehicleentrymass=50; %metric tons
aeroshellmassfraction=0.68;
strucmassfraction=0.65; % Descent Propulsion System Structural Mass Fraction
landalt=-4.0; % Desired Landing Site Elevation (measured from Mean Planet
Radius)(km)
descentalt=5; % Desired Propulsive Descent Initiation Altitude Above Landing
Site(km)
entry_apo_alt=500.0; %  Entry Orbit Apoapsis  Altitude (km)
entry_peri_alt=50.0; % Entry Orbit Periapsis Altitude (km)

propdeploytime=5.0; % Time between aeroshell jettison and engine start
deploytimecheck='n'; % Boolean  variable to allow user to change deploy time
propseparationmassfraction=0.1; % Mass penalty fraction for propulsive
separation of aeroshell
propseparationcheck='n'; % Boolean variable to allow user to change mass
penalty fraction

Cdprop=1.2; % Descent Configuration Drag Coefficient
L_Dprop=0.0; % Descent Configuration Lift over Drag Ratio
Isp=379; % Descent Engine Isp

changedefault='n'; % Boolean variable to allow user to change default values
of parameters



%Variable name coding

nvariables=13;
variables=cell(nvariables,3);
variableindex=1:1:nvariables;
```

```
variablenames={'Drag Coefficient (Cd)'; 'L/D'; 'Diameter (m)';'Entry Mass
(metric tons)';...
'Aeroshell Mass Fraction'; 'Descent Stage Structural Mass Fraction';'Landing
Altitude (m)';...
'Propulsive Descent Start Altitude (km)'; 'Entry Orbit Apoapsis Altitude
(km)';...
'Entry Orbit Periapsis Altitude (km)'; 'Descent Drag Coefficient'; 'Descent
L/D'; 'Engine Isp'};
variablenamesshort={'Cd'; 'L/D'; 'Dia';'Entry Mass';'AMF'; 'DSMF';'Landing
Alt';'PDAlt'; 'Orbit Apo';'Orbit Peri'; 'Descent Cd'; 'Descent L/D'; 'Engine
Isp'};
units={'none'; 'none'; 'm'; 'mT (metric tons)'; 'none'; 'none'; 'km'; 'km';
'km'; 'km'; 'none'; 'none'; 's'};
for i=1:nvariables
 variables{i,1}=variableindex(i);
 variables{i,2}=variablenamesshort{i};
 variables{i,4}=units{i};
end

k=ones(nvariables,1);
tableheadings={'#' 'Parameter Name' 'Value' 'Unit'};

variables{1,3}=Cd;
variables{2,3}=L_D;
variables{3,3}=Dia;
variables{4,3}= vehicleentrymass;
variables{5,3}=aeroshellmassfraction;
variables{6,3}=strucmassfraction;
variables{7,3}=landalt;
variables{8,3}=descentalt;
variables{9,3}=entry_apo_alt;
variables{10,3}=entry_peri_alt;
variables{11,3}=Cdprop;
variables{12,3}=L_Dprop;
variables{13,3}=Isp;
disp(tableheadings)
disp(variables)
addblankline

defaultparameters=variables; % setting default values of parameters

% User Input Block
xvariableindex=input('Choose x-variable (by number):');
yvariableindex=input('Choose y-variable (by number):');
addblankline
variables{xvariableindex,3}=input('Enter x-variable range in the form -
minimum:stepsize:maximum \n');
variables{yvariableindex,3}=input('Enter y-variable range in the form -
minimum:stepsize:maximum \n');
addblankline

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Changing Default Parameter Values Block

disp('The current default parameter values are:')
disp(defaultparameters)
changedefault=input('Would you like to change the default value of any of the
fixed parameters (y/n) ? \nNote: this choice will not affect the parameters
you have chosen as variables above. \n','s');
addblankline

while changedefault=='y'
    disp('Your variables for this simulation are:')
    disp(variables{xvariableindex,2})
    disp(variables{yvariableindex,2})
    addblankline
    changeparameterindex=input('Choose the number of the parameter to
change:');
    addblankline
    disp('You have chosen to change the default value for:');
    disp(variables{changeparameterindex,2})
    addblankline
    variables{changeparameterindex,3}=input('Please enter the new value of
this parameter:');
    addblankline

defaultparameters{changeparameterindex,3}=variables{changeparameterindex,3};
    disp('The current default parameter values are:')
    disp(tableheadings)
    disp(defaultparameters)

    changedefault=input('Would you like to change the default value of any of
the fixed parameters (y/n) ? \n','s');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Aeroshell Jettison Limits Check and Input

jettisonlimitcheck=input('Is there an upper mach number limit for release of
aeroshell/backshell (y/n) ?\n','s');
if jettisonlimitcheck=='y'
    machjettisonlimit=input('What is this limit? \n');
    addblankline
    if machjettisonlimit>machparaoplimitplanet
        disp('Separation may take place at speeds faster than the operational
envelope of parachutes.')
        disp('Therefore, propulsive means might be used.')
        addblankline
        disp('The current mass overhead fraction for propulsive separation
is:')
        disp(propseparationmassfraction)
        propseparationcheck=input('Would you like to change this overhead
(y/n) ? \n','s');
        if propseparationcheck=='y'
```

```
            propseparationmassfraction=input('Enter the new mass overhead
fraction for propulsive separation:\n');
        end
    end
elseif jettisonlimitcheck=='n'
    disp('Separation may take place at speeds faster than the operational
envelope of parachutes.')
    disp('Therefore, propulsive means might be used')
    disp('The current mass overhead fraction for propulsive separation is:')
    disp(propseparationmassfraction)
    addblankline
    propseparationcheck=input('Would you like to change this overhead (y/n) ?
\n','s');
    addblankline
    if propseparationcheck=='y'
        propseparationmassfraction=input('Enter the new mass overhead
fraction for propulsive separation:\n');
    end
end
addblankline

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% time to engine start from aeroshell jettison setting block

deploytimechecktext=['Would you like to specify a propulsion system
deployment time (y/n)? Default is ' num2str(propdeploytime) ' seconds. \n'];
deploytimecheck=input(deploytimechecktext,'s');
if deploytimecheck=='y'
    propdeploytime=input('Please specify the propulsion system deployment
time in seconds: \n');
end
addblankline

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Passing data to trajectory calculation routine

disp('Calculating. Please be patient...')
for j=1:length(variables{yvariableindex,3})
    countpayloadmassfraction=1;
    countthrustreq=1;

    for i=1:length(variables{xvariableindex,3})
     datapointcount=datapointcount+1;

      i;
      j;
      if

            i*j==length(variables{xvariableindex,3})*length(variables{yvariab
            leindex,3})
          disp('Almost done...')
      else
```

```
          disp(['Now Calculating Data Point ' num2str(datapointcount,'%3.0f')
' of '
num2str(length(variables{xvariableindex,3})*length(variables{yvariableindex,3
}),'%3.0f')]);
      end

      k(xvariableindex)=i;
      k(yvariableindex)=j;

[V_final(i,j),h_final(i,j),s_final(i,j),V_descent(i,j),h_descent(i,j),s_desce
nt(i,j),mass_final(i,j),fuelmass(i,j),payloadmass(i,j),payloadmassfraction(i,
j),descenttime(i,j),thrustreq(i,j),Maxg(i,j)]=EDLsim(variables{1,3}(k(1)),var
iables{3,3}(k(3)),variables{2,3}(k(2)),variables{4,3}(k(4)),variables{5,3}(k(
5)),variables{6,3}(k(6)),variables{8,3}(k(8)),variables{7,3}(k(7)),variables{
9,3}(k(9)),variables{10,3}(k(10)),variables{11,3}(k(11)),variables{12,3}(k(12
)), variables{13,3}(k(13)));

    end
    for p=1:length(variables{xvariableindex,3})
        if payloadmassfraction(p,j)~=-10 && thrustreq(p,j)~=-10

            plotpayloadmassfraction{j}(1,countpayloadmassfraction)=variables{
            xvariableindex,3}(p);

            plotpayloadmassfraction{j}(2,countpayloadmassfraction)=(payloadma
            ssfraction(p,j));
            countpayloadmassfraction=countpayloadmassfraction+1;
        end

    end
    for p=1:length(variables{xvariableindex,3})
        if payloadmassfraction(p,j)~=-10 && thrustreq(p,j)~=-10

plotthrustreq{j}(1,countthrustreq)=variables{xvariableindex,3}(p);
            plotthrustreq{j}(2,countthrustreq)=(thrustreq(p,j));
            countthrustreq=countthrustreq+1;
        end

    end

end

%  Plotting Payload Mass Fraction Data
figure
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
figure1handle=gca;
set(figure1handle,'FontSize',16)
if length(plotpayloadmassfraction)>1

for m=1:(length(plotpayloadmassfraction)-1)
    plot(plotpayloadmassfraction{m}(1,:),plotpayloadmassfraction{m}(2,:))
    hold all
end
end
```

```
plot(plotpayloadmassfraction{length(plotpayloadmassfraction)}(1,:),plotpayloa
dmassfraction{length(plotpayloadmassfraction)}(2,:))

% Adding line for maximum payload mass fraction

if xvariableindex ~= 5 && yvariableindex~=5
    maxpayloadmassfraction=1/(1+defaultparameters{5,3})
    hold all

    plot(variables{xvariableindex,3},maxpayloadmassfraction*ones(length(variab
    les{xvariableindex,3})))
end

ylim([0 1])
xlabel([variables{xvariableindex,2} ' (' variables{xvariableindex,4} ')
'],'FontSize',16,'FontWeight','bold')
ylabel('Payload Mass Fraction','FontSize',16,'FontWeight','bold')
legnum2(variables{yvariableindex,3}) % Numeric Legend

if xvariableindex ~= 5 && yvariableindex~=5

text(min(variables{xvariableindex,3})*1.1,maxpayloadmassfraction+0.06,'Max
Payload Mass Fraction (only aeroshell &
payload)','FontSize',16,'FontWeight','light')

text(mean(variables{xvariableindex,3}),maxpayloadmassfraction+0.02,'\downarro
w','FontSize',16,'FontWeight','light')
end

graphtitle1=['Effect of ' variables{xvariableindex,2} ' & '
variables{yvariableindex,2} ' on Payload Mass Fraction' ' (Planet: '
Planetname ')'];
title(graphtitle1,'FontSize',16,'FontWeight','bold')

% Adding text listing parameter values

xcoordtext=0.05;
ycoordtext=0.95;
count=0;
for b=1:length(variables)
    if b ~= xvariableindex && b ~= yvariableindex
        count=count+1;
        l=text(xcoordtext,ycoordtext,[variables{b,2} ' '
num2str(variables{b,3})],'Units','normalized','FontUnits','points','FontSize'
,16,'FontWeight','normal');
        currentextent=get(l,'Extent');

        xcoordtext=xcoordtext+0.2;
        if mod(count,3)==0
            xcoordtext=0.05;
            ycoordtext=ycoordtext-0.05;
        end
    end
end
```

```
% Plotting Required Thrust Data
figure
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
figure2handle=gca;
set(figure2handle,'FontSize',16)
if length(plotthrustreq)>1
for m=1:(length(plotthrustreq)-1)
    plot(plotthrustreq{m}(1,:),plotthrustreq{m}(2,:)./1000000)
    hold all
end
end
plot(plotthrustreq{length(plotthrustreq)}(1,:),plotthrustreq{length(plotthrus
treq)}(2,:)./1000000)

ylim([0 2])
xlabel([variables{xvariableindex,2} ' (' variables{xvariableindex,4} ')
'],'FontSize',16,'FontWeight','bold')
ylabel('Thrust Required * 1000000','FontSize',16,'FontWeight','bold')
legnum2(variables{yvariableindex,3}) % Numeric Legend
graphtitle2=['Effect of ' variables{xvariableindex,2} ' & '
variables{yvariableindex,2} ' on Thrust Required' ' (Planet: ' Planetname
')'];
title(graphtitle2,'FontSize',16,'FontWeight','bold')

% Adding text listing parameter values

xcoordtext=0.05;
ycoordtext=0.95;
count=0;
for b=1:length(variables)
    if b ~= xvariableindex && b ~= yvariableindex
        count=count+1;
        l=text(xcoordtext,ycoordtext,[variables{b,2} ' '
num2str(variables{b,3})],'Units','normalized','FontUnits','points','FontSize'
,16,'FontWeight','normal');
        currentextent=get(l,'Extent');

        xcoordtext=xcoordtext+0.2;
        if mod(count,3)==0
            xcoordtext=0.05;
            ycoordtext=ycoordtext-0.05;
        end
    end
end

% Playing sound at simulation completion

simulationcomplete = wavread('simulationcomplete.wav');
sound(simulationcomplete,22500)
```

***** End of Code *****

Filename: EntryParametricAnalysisPlotGeneratorLD.m

***** Start of Code *****

```
%Input Block

L_D=input('Enter L/D range in the form - minimum:stepsize:maximum \n');

ballisticcoeff=input('Enter Ballistic Coefficient range in the form -
minimum:stepsize:maximum \n');


entry_apo_alt=input('Enter Entry Orbit Apoapsis Altitude (km):');
entry_apo_alt=entry_apo_alt*1000; % m

entry_peri_alt=input('Enter Entry Orbit Periapsis Altitude (km):');
entry_peri_alt=entry_peri_alt*1000; % m

landalt=input('Enter Landing Site Elevation (km):');

landalt=landalt*1000;

count=1;

%Sensitivity Calculations

for j=1:length(L_D)
counthMach4=1;
counthMach3=1;
counthMach2=1;
    for i=1:length(ballisticcoeff)
        display('Calculating...')
        display('Ballistic Coefficient')
        display(ballisticcoeff(i))

        %Passing variables to simulation routine
        [v_final{j}(i), hMach4{j}(i), hMach3{j}(i), hMach2{j}(i)] =
Entrysimballcoefffunc(L_D(j),ballisticcoeff(i),entry_apo_alt,entry_peri_alt,l
andalt);

    end

    %Marking infeasible data points

    for k=1:length(hMach4{j})
        if hMach4{j}(k) >= 128*1000.0
            hMach4{j}(k)=-10000.0;
        end
```

```
    end

    for k=1:length(hMach3{j})
        if hMach3{j}(k) >= 128*1000.0
            hMach3{j}(k)=-10000.0;
        end

    end

    for k=1:length(hMach2{j})
        if hMach2{j}(k) >= 128*1000.0
            hMach2{j}(k)=-10000.0;
        end

    end

    %Removing infeasible data points

    for k=1:length(ballisticcoeff)
        if hMach4{j}(k)~=0.0 && hMach4{j}(k)~=-10000.0
            plothMach4{j}(1,counthMach4)=ballisticcoeff(k);
            plothMach4{j}(2,counthMach4)=(hMach4{j}(k))./1000.0;
            counthMach4=counthMach4+1;
        end

    end

    for k=1:length(ballisticcoeff)
        if hMach3{j}(k)~=0.0 && hMach3{j}(k)~=-10000.0
            plothMach3{j}(1,counthMach3)=ballisticcoeff(k);
            plothMach3{j}(2,counthMach3)=(hMach3{j}(k))./1000.0;
            counthMach3=counthMach3+1;
        end

    end

    for k=1:length(ballisticcoeff)
        if hMach2{j}(k)~=0.0 && hMach2{j}(k)~=-10000.0
            plothMach2{j}(1,counthMach2)=ballisticcoeff(k);
            plothMach2{j}(2,counthMach2)=(hMach2{j}(k))./1000.0;
            counthMach2=counthMach2+1;
        end

    end


end


%Plotting final altitude at Mach 4

figure
```

```
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
for j=1:(length(plothMach4)-1)
    plot(plothMach4{j}(1,:),plothMach4{j}(2,:))
    hold all
end
plot(plothMach4{length(plothMach4)}(1,:),plothMach4{length(plothMach4)}(2,:))



legnum2(L_D(1:length(plothMach4)))
ylim([-4 128])
xlabel('Ballistic Coefficient (kg/m^2)')
ylabel('Final Altitude (km)')
title('Effect of Ballistic Coefficient on Final Altitude of Reentry Vehicle
at Mach 4.0 for various L/D values')



%Plotting final altitude at Mach 3

figure
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
for j=1:(length(plothMach3)-1)
    plot(plothMach3{j}(1,:),plothMach3{j}(2,:))
    hold all
end
plot(plothMach3{length(plothMach3)}(1,:),plothMach3{length(plothMach3)}(2,:))




legnum2(L_D(1:length(plothMach3)))
ylim([-4 128])
xlabel('Ballistic Coefficient (kg/m^2)')
ylabel('Final Altitude (km)')
title('Effect of Ballistic Coefficient on Final Altitude of Reentry Vehicle
at Mach 3.0 for various L/D values')

%Plotting final altitude at Mach 2

figure
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
for j=1:(length(plothMach2)-1)
    plot(plothMach2{j}(1,:),plothMach2{j}(2,:))
    hold all
end
plot(plothMach2{length(plothMach2)}(1,:),plothMach2{length(plothMach2)}(2,:))



legnum2(L_D(1:length(plothMach2)))
```

```
ylim([-4 128])
xlabel('Ballistic Coefficient (kg/m^2)')
ylabel('Final Altitude (km)')
title('Effect of Ballistic Coefficient on Final Altitude of Reentry Vehicle
at Mach 2.0 for various L/D values')
```

***** End of Code *****

Filename: EntryParametricAnalysisPlotGeneratorOrbit.m

***** Start of Code *****

```
%Input Block


ballisticcoeff=input('Enter Ballistic Coefficient range in the form -
minimum:stepsize:maximum \n');



entry_apo_alt=input('Enter Entry Orbit Apoapsis altitude (km) range in the
form - minimum:stepsize:maximum \n'); % m
entry_apo_alt=entry_apo_alt.*1000.0;


entry_peri_alt=input('Enter Entry Orbit Periapsis Altitude (km):');
entry_peri_alt=entry_peri_alt*1000; % m

L_D=input('Enter Lift-to-Drag Ratio');

landalt=input('Enter Landing Site Elevation (km):');

landalt=landalt*1000;

count=1;

%Sensitvity Calculations

for j=1:length(entry_apo_alt)
counthMach4=1;
counthMach3=1;
counthMach2=1;
    for i=1:length(ballisticcoeff)
        display('Calculating...')
        display('Ballistic Coefficient')
        display(ballisticcoeff(i))

        %Passing variables to simulation routine
        [v_final{j}(i), hMach4{j}(i), hMach3{j}(i), hMach2{j}(i)] =
Entrysimballcoefffunc(L_D,ballisticcoeff(i),entry_apo_alt(j),entry_peri_alt,l
andalt);

    end

    %Marking infeasible data points

    for k=1:length(hMach4{j})
```

```
        if hMach4{j}(k) >= 128*1000.0
            hMach4{j}(k)=-10000.0;
        end

    end

    for k=1:length(hMach3{j})
        if hMach3{j}(k) >= 128*1000.0
            hMach3{j}(k)=-10000.0;
        end

    end

    for k=1:length(hMach2{j})
        if hMach2{j}(k) >= 128*1000.0
            hMach2{j}(k)=-10000.0;
        end

    end


    %Removing infeasible data points

    for k=1:length(ballisticcoeff)
        if hMach4{j}(k)~=0.0 && hMach4{j}(k)~=-10000.0
            plothMach4{j}(1,counthMach4)=ballisticcoeff(k);
            plothMach4{j}(2,counthMach4)=(hMach4{j}(k))./1000.0;
            counthMach4=counthMach4+1;
        end

    end

    for k=1:length(ballisticcoeff)
        if hMach3{j}(k)~=0.0 && hMach3{j}(k)~=-10000.0
            plothMach3{j}(1,counthMach3)=ballisticcoeff(k);
            plothMach3{j}(2,counthMach3)=(hMach3{j}(k))./1000.0;
            counthMach3=counthMach3+1;
        end

    end

    for k=1:length(ballisticcoeff)
        if hMach2{j}(k)~=0.0 && hMach2{j}(k)~=-10000.0
            plothMach2{j}(1,counthMach2)=ballisticcoeff(k);
            plothMach2{j}(2,counthMach2)=(hMach2{j}(k))./1000.0;
            counthMach2=counthMach2+1;
        end

    end


end
```

```
%Plotting final altitude at Mach 4

figure
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
for j=1:(length(plothMach4)-1)
    plot(plothMach4{j}(1,:),plothMach4{j}(2,:))
    hold all
end
plot(plothMach4{length(plothMach4)}(1,:),plothMach4{length(plothMach4)}(2,:))



legnum2(entry_apo_alt(1:length(plothMach4)))
ylim([-4 128])
xlabel('Ballistic Coefficient (kg/m^2)')
ylabel('Final Altitude (km)')
title('Effect of Ballistic Coefficient on Final Altitude of Reentry Vehicle
at Mach 4.0 for various Entry Orbit Apoapsis (km) values')



%Plotting final altitude at Mach 3

figure
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
for j=1:(length(plothMach3)-1)
    plot(plothMach3{j}(1,:),plothMach3{j}(2,:))
    hold all
end
plot(plothMach3{length(plothMach3)}(1,:),plothMach3{length(plothMach3)}(2,:))




legnum2(entry_apo_alt(1:length(plothMach3)))
ylim([-4 128])
xlabel('Ballistic Coefficient (kg/m^2)')
ylabel('Final Altitude (km)')
title('Effect of Ballistic Coefficient on Final Altitude of Reentry Vehicle
at Mach 3.0 for various Entry Orbit Apoapsis (km) values')

%Plotting final altitude at Mach 2

figure
set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})
for j=1:(length(plothMach2)-1)
    plot(plothMach2{j}(1,:),plothMach2{j}(2,:))
    hold all
end
```

```
plot(plothMach2{length(plothMach2)}(1,:),plothMach2{length(plothMach2)}(2,:))



legnum2(entry_apo_alt(1:length(plothMach2)))
ylim([-4 128])
xlabel('Ballistic Coefficient (kg/m^2)')
ylabel('Final Altitude (km)')
title('Effect of Ballistic Coefficient on Final Altitude of Reentry Vehicle
at Mach 2.0 for various Entry Orbit Apoapsis (km) values')
```

***** End of Code *****

Filename: EntryParametricAnalysisTool.m

***** Start of Code *****

```
clear all
close all
clc

%Select type of parametric sensitivity analysis to run

disp('Welcome to the Entry Parametric Analysis Tool')
addblankline
analysisselection=input('If you would like to vary Ballistic Coefficient and
L/D, enter 1 \nIf you would like to vary Ballistic Coefficient and Entry
Orbit Apoapsis Altitude, enter 2 \n');
addblankline


Planetname=input('Enter planet name for entry, descent and landing
simulation: ','s');

% planet constants

global G Mplanet Rplanet kplanet Rgasplanet atmos_interface_planet
periodplanet angularrotplanet machparaoplimitplanet

G=6.673e-11; % Universal Gravitational Constant

% Reading planet constants from file

planetconstantsfile=input('Enter name of text file (including extension)
containing the list of planet constants: \n','s');
planetconstants=dlmread(planetconstantsfile,'\n');

Mplanet=planetconstants(1); %Mass of Planet (kg)
Rplanet=planetconstants(2); % Mean Planet Radius (km)
kplanet=planetconstants(3); %Ratio of Specific Heats for Atmosphere
Rgasplanet=planetconstants(4);% Gas Constant for Atmosphere (J/kg/K)
atmos_interface_planet=planetconstants(5); %Atmospheric Interface Radius (km)
periodplanet=planetconstants(6); %Sidereal Rotation Period (hr)
machparaoplimitplanet=planetconstants(7); %Operational Limit for Parachutes
in Planetary Atmosphere

%planet constant adjustments & related calculations
Rplanet=Rplanet*1000.0;
periodplanet=periodplanet*60*60; %Sidereal Rotation Period (s)
rotdirection=input('Enter 1 for entering in direction of planet sidereal
rotation \nEnter -1 for entering in direction opposite to planet sidereal
rotation \n');
```

```
angularrotplanet=-rotdirection*2*pi()/periodplanet; %Sidereal Rotational
Speed (rad/s)


% Atmospheric data source specification

global AtmDensityPlanetfunc AtmTempPlanetfunc


AtmDensityPlanet=input('Enter the name for the Atmospheric Density Profile
function file (input=altitude in m)) \n','s');
AtmTempPlanet=input('Enter the name for the Atmospheric Temperature Profile
function file (input=altitude in m)) \n','s');

AtmDensityPlanetfunc=str2func(AtmDensityPlanet);
AtmTempPlanetfunc=str2func(AtmTempPlanet);

addblankline


%Run selected type of analysis

if analysisselection==1
    EntryParametricAnalysisPlotGeneratorLD

else
    EntryParametricAnalyisPlotGeneratorOrbit
end
```

***** End of Code *****

Filename: Entrysim.m

***** Start of Code *****

```
% entry conditions
atmos_interface_planet_r=atmos_interface_planet*1000+Rplanet;
[v_interface, gamma_interface]=PlanetEntryConditions(entry_apo_alt,
entry_peri_alt); % get velocity and flight path angle at entry interface
given entry orbit
vx_interface = v_interface*cos(gamma_interface); % Vehicle x-velocity at
atmospheric interface (m/s)
vy_interface=v_interface*sin(gamma_interface); % Vehicle y-velocity at
atmospheric interface (m/s)
vrot_interface= angularrotplanet*atmos_interface_planet_r; %  v=omega*r,
atmospheric velocity at interface (m/s)
vx_initial=vx_interface+vrot_interface; % Entry x-velocity (m/s) (relative to
planet frame)
vy_initial=vy_interface; % Entry y-velocity (m/s) (relative to planet frame)
v_initial=sqrt(vx_initial^2+vy_interface^2); % Entry velocity (m/s) (relative
to planet frame)
gamma_initial=atan(vy_initial/vx_initial);% Entry flight path angle (radians)
(relative to planet frame)
h_initial=atmos_interface_planet*1000; %Entry altitude (m)
s_initial=0*1000; %Entry range (m)


% All quantities are relative to planet frame

%initialize variables

delta_t=0.1; % time step
total_time=10000.0;
time=[0.0:delta_t:total_time];
h=zeros(size(time)); % altitude (m)
h(1)=h_initial;
r=Rplanet+h; % distance from planet center (m)
s=zeros(size(time)); % range from entry (m)
s(1)=s_initial;
V=zeros(size(time)); % Velocity (m/s)
V(1)=v_initial;
Vx=zeros(size(time)); % X-component of velocity (m/s)
Vx(1)=vx_initial;
Vy=zeros(size(time)); % Y-Component of velocity (m/s)
Vy(1)=vy_initial;
gamma=zeros(size(time)); % Flight Path Angle (radians)
gamma(1)= gamma_initial;
```

```
theta=zeros(size(time));% Angle subtended by entry arc at planet center,
used for figuring out local vertical for gravity force
theta(1)= pi()/2.0;
mach=zeros(size(time));
mach(1)=V(1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(1)));

dvx_dt=zeros(size(time)-1); %X-component of acceleration (m/s^2)
dvy_dt=zeros(size(time)-1); %Y-component of acceleration (m/s^2)

% forces
drag=zeros(size(time)-1);
lift=zeros(size(time)-1);
gravity =zeros(size(time)-1);

%Absolute positions in planet reference frame
y=zeros(size(time));
y(1)=r(1);

x=zeros(size(time));
x(1)=s(1);

nrevolutions=0; % number of "revolutions" around planet during the entry

%Time step routine
for i=1:(length(time)-1)
    drag(i)=0.5*AtmDensityPlanetfunc(h(i))*(V(i)^2)*vehiclefrontalarea*Cd;

lift(i)=0.5*AtmDensityPlanetfunc(h(i))*(V(i)^2)*vehiclefrontalarea*Cd*L_D;
    gravity(i)=G*Mplanet/(r(i)^2);

    %to keep lift pointing upward
    if Vx(i)>0.0

    dvx_dt(i)=
lift(i)*cos(gamma(i)+pi()/2.0)/vehicleentrymass+drag(i)*cos(pi()+gamma(i))/ve
hicleentrymass - gravity(i)*x(i)/r(i) + 2*angularrotplanet*Vy(i) +
(angularrotplanet^2)*x(i);
    dvy_dt(i)=
lift(i)*sin(gamma(i)+pi()/2.0)/vehicleentrymass+drag(i)*sin(pi()+gamma(i))/ve
hicleentrymass - gravity(i)*y(i)/r(i) - 2*angularrotplanet*Vx(i) +
(angularrotplanet^2)*y(i);

    else

    dvx_dt(i)= lift(i)*cos(gamma(i)-
pi()/2.0)/vehicleentrymass+drag(i)*cos(pi()+gamma(i))/vehicleentrymass -
gravity(i)*x(i)/r(i) + 2*angularrotplanet*Vy(i) + (angularrotplanet^2)*x(i);
    dvy_dt(i)= lift(i)*sin(gamma(i)-
pi()/2.0)/vehicleentrymass+drag(i)*sin(pi()+gamma(i))/vehicleentrymass -
gravity(i)*y(i)/r(i) - 2*angularrotplanet*Vx(i) + (angularrotplanet^2)*y(i);

    end
```

```
Vx(i+1)=Vx(i)+dvx_dt(i)*delta_t;
Vy(i+1)=Vy(i)+dvy_dt(i)*delta_t;


V(i+1)=sqrt(Vx(i+1)^2+ Vy(i+1)^2);

%to get correct value of angle in III'rd quadrant
if  Vx(i+1)<0.0
    gamma(i+1)=atan(Vy(i+1)/Vx(i+1)) + pi();
else
    gamma(i+1)=atan(Vy(i+1)/Vx(i+1));
end

y(i+1)=y(i)+Vy(i)*delta_t;
x(i+1)=x(i)+Vx(i)*delta_t;

%to get correct value of angle in III'rd quadrant
if x(i+1)<0.0
    theta(i+1)=atan(y(i+1)/x(i+1)) + pi();
else
    theta(i+1)=atan(y(i+1)/x(i+1));
end

r(i+1)=sqrt(y(i+1)^2+x(i+1)^2);
h(i+1)=r(i+1)-Rplanet;

% setting range based on subtended angle, theta

    if x(i+1)>=0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+pi()/2.0-theta(i+1));
    elseif x(i+1)<0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+5*pi()/2.0-theta(i+1));
    end
    if abs(s(i+1)-s(i))>0.9*2*pi()*Rplanet
        nrevolutions=nrevolutions+1;
    end

    if x(i+1)>=0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+pi()/2.0-theta(i+1));
    elseif x(i+1)<0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+5*pi()/2.0-theta(i+1));
    end


% Mach number calculation
if h(i+1)<=atmos_interface_planet*1000

    mach(i+1)=V(i+1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(i+1)));
else
    mach(i+1)=0.0;
end
```

```
    % Simulation stop logic:

    if r(i+1)<=Rplanet+(descentalt+landalt)
    % if desired propulsive descent initiation altitude has been reached,
    % check to see if there is an aeroshell jettison mach number limit
        if jettisonlimitcheck=='y'
            % if there is an aeroshell jettison mach number limit, check to
            % see that current mach number is below the limit and stop
            % simulation; if not, coast till below limit and stop
            % simulation
            if mach(i+1)<machjettisonlimit

                for j=i+2:length(s)
                    s(j)=s(i+1);
                end
                V_final=V(i+1);
                i_final=i;
                break;
            end
        else
            % if there is no Mach number limit, stop simulation
            for j=i+2:length(s)
                s(j)=s(i+1);
            end
            V_final=V(i+1);
            i_final=i;
            break;
        end
    end
    i_final=i;
    V_final=V(i);
end

%Mach number contour calculation
for i=1:117
    htest(i)=(i-5)*1000.0;
    mach2v(i)=2*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
    mach3v(i)=3*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
    mach4v(i)=4*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
    mach5v(i)=5*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
end
```

***** End of Code *****

Filename: Entrysimballcoefffunc.m

***** Start of Code *****

```
function [V_final, hMach4, hMach3, hMach2] =
Entrysimballcoefffunc(L_D,ballisticcoeff,entry_apo_alt,entry_peri_alt,landalt
)

% author: Zahra Khan

% constants

global G Mplanet Rplanet kplanet Rgasplanet atmos_interface_planet
periodplanet angularrotplanet machparaoplimitplanet
global AtmDensityPlanetfunc AtmTempPlanetfunc




% entry conditions
atmos_interface_planet_r=atmos_interface_planet*1000+Rplanet;
[v_interface, gamma_interface]=PlanetEntryConditions(entry_apo_alt,
entry_peri_alt); % get velocity and flight path angle at entry interface
given entry orbit
vx_interface = v_interface*cos(gamma_interface); % Vehicle x-velocity at
atmospheric interface (m/s)
vy_interface=v_interface*sin(gamma_interface); % Vehicle y-velocity at
atmospheric interface (m/s)
vrot_interface= angularrotplanet*atmos_interface_planet_r; %  v=omega*r,
atmospheric velocity at interface (m/s)
vx_initial=vx_interface+vrot_interface; % Entry x-velocity (m/s) (relative to
planet frame
vy_initial=vy_interface; % Entry y-velocity (m/s) (relative to planet frame
v_initial=sqrt(vx_initial^2+vy_interface^2); % Entry velocity (m/s) (relative
to planet frame
gamma_initial=atan(vy_initial/vx_initial);% Entry flight path angle (radians)
(relative to planet frame
h_initial=atmos_interface_planet*1000; %Entry altitude (m)
s_initial=0*1000; %Entry range (m)


% All quantities are relative to planet frame

%initialize variables

delta_t=0.1; % time step
total_time=10000.0;
time=[0.0:delta_t:total_time];
h=zeros(size(time)); % altitude (m)
h(1)=h_initial;
```

```
r=Rplanet+h; % distance from planet center (m)
s=zeros(size(time)); % range from entry (m)
s(1)=s_initial;
V=zeros(size(time)); % Velocity (m/s)
V(1)=v_initial;
Vx=zeros(size(time)); % X-component of velocity (m/s)
Vx(1)=vx_initial;
Vy=zeros(size(time)); % Y-Component of velocity (m/s)
Vy(1)=vy_initial;
gamma=zeros(size(time)); % Flight Path Angle (radians)
gamma(1)= gamma_initial;
theta=zeros(size(time));% Angle subtended by entry arc at planet center,
used for figuring out local vertical for gravity force
theta(1)= pi()/2.0;
mach=zeros(size(time));
mach(1)=V(1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(1)));

dvx_dt=zeros(size(time)-1); %X-component of acceleration (m/s^2)
dvy_dt=zeros(size(time)-1); %Y-component of acceleration (m/s^2)


% forces
dynp=zeros(size(time)-1);
gravity =zeros(size(time)-1);



%Absolute positions in planet reference frame
y=zeros(size(time));
y(1)=r(1);

x=zeros(size(time));
x(1)=s(1);

nrevolutions=0; % number of "revolutions" around planet during the entry



%Time step routine
for i=1:(length(time)-1)

    gravity(i)=G*Mplanet/(r(i)^2);
    dynp(i)=0.5*AtmDensityPlanetfunc(h(i))*(V(i)^2);

    %to keep lift pointing upward

    if Vx(i)>0.0

    dvx_dt(i)=
(dynp(i)*L_D)*cos(gamma(i)+pi()/2.0)/ballisticcoeff+(dynp(i))*cos(pi()+gamma(
i))/ballisticcoeff - gravity(i)*x(i)/r(i) + 2*angularrotplanet*Vy(i) +
(angularrotplanet^2)*x(i);
```

```
    dvy_dt(i)=
(dynp(i)*L_D)*sin(gamma(i)+pi()/2.0)/ballisticcoeff+(dynp(i))*sin(pi()+gamma(
i))/ballisticcoeff - gravity(i)*y(i)/r(i) - 2*angularrotplanet*Vx(i) +
(angularrotplanet^2)*y(i);

    else

    dvx_dt(i)= (dynp(i)*L_D)*cos(gamma(i)-
pi()/2.0)/ballisticcoeff+(dynp(i))*cos(pi()+gamma(i))/ballisticcoeff -
gravity(i)*x(i)/r(i) + 2*angularrotplanet*Vy(i) + (angularrotplanet^2)*x(i);
    dvy_dt(i)= (dynp(i)*L_D)*sin(gamma(i)-
pi()/2.0)/ballisticcoeff+(dynp(i))*sin(pi()+gamma(i))/ballisticcoeff -
gravity(i)*y(i)/r(i) - 2*angularrotplanet*Vx(i) + (angularrotplanet^2)*y(i);

    end

    Vx(i+1)=Vx(i)+dvx_dt(i)*delta_t;
    Vy(i+1)=Vy(i)+dvy_dt(i)*delta_t;

    V(i+1)=sqrt(Vx(i+1)^2+ Vy(i+1)^2);



    %to get correct value of angle in III'rd quadrant
    if  Vx(i+1)<0.0
        gamma(i+1)=atan(Vy(i+1)/Vx(i+1)) + pi();
    else
        gamma(i+1)=atan(Vy(i+1)/Vx(i+1));
    end



    y(i+1)=y(i)+Vy(i)*delta_t;
    x(i+1)=x(i)+Vx(i)*delta_t;

    %to get correct value of angle in III'rd quadrant
    if x(i+1)<0.0
        theta(i+1)=atan(y(i+1)/x(i+1)) + pi();
    else
        theta(i+1)=atan(y(i+1)/x(i+1));
    end



    r(i+1)=sqrt(y(i+1)^2+x(i+1)^2);
    h(i+1)=r(i+1)-Rplanet;



    % setting range based on subtended angle, theta

        if x(i+1)>=0.0
            s(i+1)=Rplanet*(nrevolutions*2*pi()+pi()/2.0-theta(i+1));
        elseif x(i+1)<0.0
            s(i+1)=Rplanet*(nrevolutions*2*pi()+5*pi()/2.0-theta(i+1));
        end
```

```
        if abs(s(i+1)-s(i))>0.9*2*pi()*Rplanet
            nrevolutions=nrevolutions+1;
        end

        if x(i+1)>=0.0
            s(i+1)=Rplanet*(nrevolutions*2*pi()+pi()/2.0-theta(i+1));
        elseif x(i+1)<0.0
            s(i+1)=Rplanet*(nrevolutions*2*pi()+5*pi()/2.0-theta(i+1));
        end



    % Mach number calculation
    if h(i+1)<=atmos_interface_planet*1000

        mach(i+1)=V(i+1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(i+1)));
    else
        mach(i+1)=0.0;
    end

    % Simulation stop logic:

    if r(i+1)<=Rplanet+landalt

        for j=i+2:length(s)
            s(j)=s(i+1);
        end
        V_final=V(i+1);
        i_final=i;
        break;
    end
    i_final=i;
    V_final=V(i);
end



V_final=V_final/1000.0;

% Mach number altitude calculation

for i=1:length(time)
    if h(i)<45*1000 % to avoid counting skip part of the trajectory in the
Mach number altitude calculation
        hformachcalc=i;
        break
    end
end

machcalc=mach(hformachcalc:length(time));

for i=1:length(machcalc)-1
    if machcalc(i+1)<2.0 && machcalc(i)>2.0
```

```
        mach2index=i;
        break
    end
    mach2index=-9999; % if Mach number not attained before reaching the
surface
end
for i=1:length(machcalc)-1
    if machcalc(i+1)<3.0 && machcalc(i)>3.0
        mach3index=i;
        break
    end
    mach3index=-9999; % if Mach number not attained before reaching the
surface
end
for i=1:length(machcalc)-1
    if machcalc(i+1)<4.0 && machcalc(i)>4.0
        mach4index=i;
        break
    end
    mach4index=-9999; % if Mach number not attained before reaching the
surface
end

if mach4index==-9999
    hMach4=0.0;
else
    hMach4=h(hformachcalc+mach4index)+ (4.0-
machcalc(mach4index))*(h(hformachcalc+mach4index+1)-
h(hformachcalc+mach4index))/(machcalc(mach4index+1)-machcalc(mach4index));
end

if mach3index==-9999
    hMach3=0.0;
else
    hMach3=h(hformachcalc+mach3index)+ (3.0-
machcalc(mach3index))*(h(hformachcalc+mach3index+1)-
h(hformachcalc+mach3index))/(machcalc(mach3index+1)-machcalc(mach3index));
end

if mach2index==-9999
    hMach2=0.0;
else
    hMach2=h(hformachcalc+mach2index)+ (2.0-
machcalc(mach2index))*(h(hformachcalc+mach2index+1)-
h(hformachcalc+mach2index))/(machcalc(mach2index+1)-machcalc(mach2index));
end
```

***** End of Code *****

Filename: Entrysimonly.m

***** Start of Code *****

```
% entry conditions
atmos_interface_planet_r=atmos_interface_planet*1000+Rplanet;
[v_interface, gamma_interface]=PlanetEntryConditions(entry_apo_alt,
entry_peri_alt); % get velocity and flight path angle at entry interface
given entry orbit
vx_interface = v_interface*cos(gamma_interface); % Vehicle x-velocity at
atmospheric interface (m/s)
vy_interface=v_interface*sin(gamma_interface); % Vehicle y-velocity at
atmospheric interface (m/s)
vrot_interface= angularrotplanet*atmos_interface_planet_r; %  v=omega*r,
atmospheric velocity at interface (m/s)
vx_initial=vx_interface+vrot_interface; % Entry x-velocity (m/s) (relative to
planet frame)
vy_initial=vy_interface; % Entry y-velocity (m/s) (relative to planet frame)
v_initial=sqrt(vx_initial^2+vy_interface^2); % Entry velocity (m/s) (relative
to planet frame)
gamma_initial=atan(vy_initial/vx_initial);% Entry flight path angle (radians)
(relative to planet frame)
h_initial=atmos_interface_planet*1000; %Entry altitude (m)
s_initial=0*1000; %Entry range (m)


% All quantities are relative to planet frame

%initialize variables

delta_t=0.1; %time step
total_time=10000.0;
time=[0.0:delta_t:total_time];
h=zeros(size(time)); % altitude (m)
h(1)=h_initial;
r=Rplanet+h; % distance from planet center (m)
s=zeros(size(time)); % range from entry (m)
s(1)=s_initial;
V=zeros(size(time)); % Velocity (m/s)
V(1)=v_initial;
Vx=zeros(size(time)); % X-component of velocity (m/s)
Vx(1)=vx_initial;
Vy=zeros(size(time)); % Y-Component of velocity (m/s)
Vy(1)=vy_initial;
dynp=zeros(size(time));
dynp(1)=0.5*AtmDensityPlanetfunc(h(1))*(V(1)^2);
gamma=zeros(size(time)); % Flight Path Angle (radians)
```

```
gamma(1)= gamma_initial;
theta=zeros(size(time));% Angle subtended by entry arc at planet center,
used for figuring out local vertical for gravity force
theta(1)= pi()/2.0;
mach=zeros(size(time));
mach(1)=V(1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(1)));

dvx_dt=zeros(size(time)-1); %X-component of acceleration (m/s^2)
dvy_dt=zeros(size(time)-1); %Y-component of acceleration (m/s^2)

% forces
drag=zeros(size(time)-1);
lift=zeros(size(time)-1);
gravity =zeros(size(time)-1);

%Absolute positions in planet reference frame
y=zeros(size(time));
y(1)=r(1);

x=zeros(size(time));
x(1)=s(1);

nrevolutions=0; % number of "revolutions" around planet during the entry

%Time step routine
for i=1:(length(time)-1)
    drag(i)=0.5*AtmDensityPlanetfunc(h(i))*(V(i)^2)*vehiclefrontalarea*Cd;

lift(i)=0.5*AtmDensityPlanetfunc(h(i))*(V(i)^2)*vehiclefrontalarea*Cd*L_D;
    gravity(i)=G*Mplanet/(r(i)^2);

    %to keep lift pointing upward
    if Vx(i)>0.0

    dvx_dt(i)=
lift(i)*cos(gamma(i)+pi()/2.0)/vehicleentrymass+drag(i)*cos(pi()+gamma(i))/ve
hicleentrymass - gravity(i)*x(i)/r(i) + 2*angularrotplanet*Vy(i) +
(angularrotplanet^2)*x(i);
    dvy_dt(i)=
lift(i)*sin(gamma(i)+pi()/2.0)/vehicleentrymass+drag(i)*sin(pi()+gamma(i))/ve
hicleentrymass - gravity(i)*y(i)/r(i) - 2*angularrotplanet*Vx(i) +
(angularrotplanet^2)*y(i);

    else

    dvx_dt(i)= lift(i)*cos(gamma(i)-
pi()/2.0)/vehicleentrymass+drag(i)*cos(pi()+gamma(i))/vehicleentrymass -
gravity(i)*x(i)/r(i) + 2*angularrotplanet*Vy(i) + (angularrotplanet^2)*x(i);
    dvy_dt(i)= lift(i)*sin(gamma(i)-
pi()/2.0)/vehicleentrymass+drag(i)*sin(pi()+gamma(i))/vehicleentrymass -
gravity(i)*y(i)/r(i) - 2*angularrotplanet*Vx(i) + (angularrotplanet^2)*y(i);

    end
```

```
Vx(i+1)=Vx(i)+dvx_dt(i)*delta_t;
Vy(i+1)=Vy(i)+dvy_dt(i)*delta_t;


V(i+1)=sqrt(Vx(i+1)^2+ Vy(i+1)^2);
dynp(i+1)=0.5*AtmDensityPlanetfunc(h(i+1))*(V(i+1)^2);

%to get correct value of angle in III'rd quadrant
if  Vx(i+1)<0.0
    gamma(i+1)=atan(Vy(i+1)/Vx(i+1)) + pi();
else
    gamma(i+1)=atan(Vy(i+1)/Vx(i+1));
end

y(i+1)=y(i)+Vy(i)*delta_t;
x(i+1)=x(i)+Vx(i)*delta_t;

%to get correct value of angle in III'rd quadrant
if x(i+1)<0.0
    theta(i+1)=atan(y(i+1)/x(i+1)) + pi();
else
    theta(i+1)=atan(y(i+1)/x(i+1));
end

r(i+1)=sqrt(y(i+1)^2+x(i+1)^2);
h(i+1)=r(i+1)-Rplanet;

% setting range based on subtended angle, theta

    if x(i+1)>=0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+pi()/2.0-theta(i+1));
    elseif x(i+1)<0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+5*pi()/2.0-theta(i+1));
    end
    if abs(s(i+1)-s(i))>0.9*2*pi()*Rplanet
        nrevolutions=nrevolutions+1;
    end

    if x(i+1)>=0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+pi()/2.0-theta(i+1));
    elseif x(i+1)<0.0
        s(i+1)=Rplanet*(nrevolutions*2*pi()+5*pi()/2.0-theta(i+1));
    end


% Mach number calculation
if h(i+1)<=atmos_interface_planet*1000

    mach(i+1)=V(i+1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(i+1)));
else
    mach(i+1)=0.0;
end
```

```
    % Simulation stop logic:

    if r(i+1)<=Rplanet+(descentalt+landalt)
    % if desired propulsive descent initiation altitude has been reached,
    % check to see if there is an aeroshell jettison mach number limit
        if jettisonlimitcheck=='y'
            % if there is an aeroshell jettison mach number limit, check to
            % see that current mach number is below the limit and stop
            % simulation; if not, coast till below limit and stop
            % simulation
            if mach(i+1)<machjettisonlimit

                for j=i+2:length(s)
                    s(j)=s(i+1);
                end
                V_final=V(i+1);
                i_final=i;
                break;
            end
        else
            % if there is no Mach number limit, stop simulation
            for j=i+2:length(s)
                s(j)=s(i+1);
            end
            V_final=V(i+1);
            i_final=i;
            break;
        end
    end
    i_final=i;
    V_final=V(i);
end

%Mach number contour calculation
for i=1:117
    htest(i)=(i-5)*1000.0;
    mach2v(i)=2*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
    mach3v(i)=3*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
    mach4v(i)=4*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
    mach5v(i)=5*sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(htest(i)));
end

% Calculation for when entry body reaches certain Mach numbers
for i=1:length(time)
    if h(i)<45*1000 % to avoid counting skip part of the trajectory in the
Mach number altitude calculation
        hformachcalc=i;
        break
    end
end

machcalc=mach(hformachcalc:length(time));
```

```
for i=1:length(machcalc)-1
    if machcalc(i+1)<2.0 && machcalc(i)>2.0
        mach2index=i;
        break
    end
    mach2index=-9999; % if Mach number not attained before reaching the
surface
end
for i=1:length(machcalc)-1
    if machcalc(i+1)<3.0 && machcalc(i)>3.0
        mach3index=i;
        break
    end
    mach3index=-9999; % if Mach number not attained before reaching the
surface
end
for i=1:length(machcalc)-1
    if machcalc(i+1)<4.0 && machcalc(i)>4.0
        mach4index=i;
        break
    end
    mach4index=-9999; % if Mach number not attained before reaching the
surface
end
if mach4index==-9999
    hMach4=0.0;
else
    hMach4=h(hformachcalc+mach4index)+ (4.0-
machcalc(mach4index))*(h(hformachcalc+mach4index+1)-
h(hformachcalc+mach4index))/(machcalc(mach4index+1)-machcalc(mach4index));
end

if mach3index==-9999
    hMach3=0.0;
else
    hMach3=h(hformachcalc+mach3index)+ (3.0-
machcalc(mach3index))*(h(hformachcalc+mach3index+1)-
h(hformachcalc+mach3index))/(machcalc(mach3index+1)-machcalc(mach3index));
end

if mach2index==-9999
    hMach2=0.0;
else
    hMach2=h(hformachcalc+mach2index)+ (2.0-
machcalc(mach2index))*(h(hformachcalc+mach2index+1)-
h(hformachcalc+mach2index))/(machcalc(mach2index+1)-machcalc(mach2index));
end

hMach4=hMach4/1000.0
hMach3=hMach3/1000.0
hMach2=hMach2/1000.0
```

***** End of Code *****

Filename: legnum2.m
Note: This code was obtained from the Matlab Central website (http://www.matlabcentral.com) and was authored and posted by Alex Barnett.

***** Start of Code *****

```
%
% LEGNUM Legend current figure using array of numbers.
%    LEGNUM(X) adds a legend to current figure using string
%    representations of the numbers in X. If X is a two- or multi-dimensional
%    array, it will be flattened and all elements will be included.
%
%    LEGNUM(X, P) is the same but uses precision P, where P is an integer.
%
%    LEGNUM(X, P, S) same as above but includes a prefix string to
%    each legend label.
%
% Examples
%    legnum(logspace(-5,-4,7), 6);
%    Adds a legend with logarithmically-spaced number labels, with
%    6 significant digit precision
%
%    legnum(logspace(-5,-4,7), 6, 'x = ');
%    Same but labels are of the form 'x = 1e-5', etc.
%
% See also NUM2CELLSTR
%
%    Alex Barnett 12/5/02

function legnum2(a, prec, prefix)

if nargin==1
  legend(num2cellstr(a));
elseif nargin==2
  legend(num2cellstr(a, prec));
elseif nargin==3
  legend(num2cellstr(a, prec, prefix));
else
  error('too many arguments to legnum.')
end
```

***** End of Code *****

Filename: num2cellstr.m
Note: This code was obtained from the Matlab Central website (http://www.matlabcentral.com) and was authored and posted by Alex Barnett.

***** Start of Code *****

```
% NUM2CELLSTR convert array of floating-point numbers to cell array of
strings
%    NUM2CELLSTR(X) converts array X to cell array of strings.
%    If X is a two- or multi-dimensional array, it will be
%    flattened (all elements will still be included).
%
%    NUM2CELLSTR(X, P) is the same but uses precision P, where P is an
integer.
%
%    NUM2CELLSTR(X, P, S) same as above but includes a prefix string to
%    each cell.
%
%    This clumsy routine would be unnecessary if Matlab provided something
%    like python's string.strip() function.
%
% See also SPRINTF, CELLSTR
%    Alex Barnett 12/5/02

function [c] = num2cellstr(a, prec, prefix)

if nargin==1
  prec = 4;      % default precision
else
  if prec<1
    error('precision must be at least 1.')
  end
  if prec>16
    error('precision cannot exceed 16.')
  end
end
if nargin<3
  prefix = ''; % default prefix
end

l = 25;          % max number of characters for representing a number
n = numel(a);

% build printf format string
f = sprintf('%%-%d.%dg', l, round(prec));

c = cellstr([repmat(prefix, [n 1]) reshape(sprintf(f, a),[l, n])']);
```

***** End of Code *****

Filename: PlanetEntryConditions.m
Note: Equations for this code were provided by Wilfried Hofstetter.

***** Start of Code *****

```
function [v_entry, gamma_entry] = PlanetEntryConditions(h_apo,h_peri)

%Planet constants
global G Mplanet Rplanet atmos_interface_planet

atmos_interface_planet_r=atmos_interface_planet*1000+Rplanet; %Radius at Mars
atmosphere interface, unit:m

% Apoapsis and Periapsis distance calculations
r_apo=h_apo+Rplanet;
r_peri=h_peri+Rplanet;

% Semimajor axis
a=(r_apo+r_peri)/2.0;


%Entry Conditions Calculations

v_entry=sqrt(G*Mplanet*(2.0/atmos_interface_planet_r - 1.0/a)); % Velocity at
planet atmospheric interface
v_apo=sqrt(G*Mplanet*(2.0/r_apo - 1.0/a)); % Velocity at orbit apoapsis
H=v_apo*r_apo; % Angular momentum of body in orbit
gamma_entry=-acos(H/(v_entry*atmos_interface_planet_r)); % Angle at planet
atmospheric interface (negative sign is added since there are two solutions
but the negative angle results in entry)
```

***** End of Code *****

Filename: PropulsiveDescentSim.m

***** Start of Code *****

```
% note: subscript "prop" indicates propulsive descent to distinguish the
variables from  those for entry trajectory calculation
%engine parameters
thrustinitial= 1000000; %N
throttleratio=throttleratiodesired;
thrustactual=thrustinitial*throttleratio;

massflowrate=thrustactual/(Isp*9.81);

%simulation paramters
total_timeprop=1000.0;
delta_tprop=0.05; % time step


%inputs
vx_initialprop=Vx(propstartindex); %m/s
vy_initialprop=Vy(propstartindex); %m/s
v_initialprop=sqrt(vx_initialprop^2+vy_initialprop^2);
s_initialprop=s(propstartindex); %m
h_initialprop=h(propstartindex); %m
gamma_initialprop=atan(vy_initialprop/vx_initialprop);

% All quantities are relative to planet frame

%initialize variables
timeprop=[0.0:delta_tprop:total_timeprop];
hprop=zeros(size(timeprop));
hprop(1)=h_initialprop;
rprop=zeros(size(timeprop));
rprop(1)=r(propstartindex);
sprop=zeros(size(timeprop));
sprop(1)=s_initialprop;
Vprop=zeros(size(timeprop));
Vprop(1)=v_initialprop;
Vxprop=zeros(size(timeprop));
Vxprop(1)=vx_initialprop;
Vyprop=zeros(size(timeprop));
Vyprop(1)=vy_initialprop;
gammaprop=zeros(size(timeprop));
gammaprop(1)= gamma_initialprop;
thetaprop=zeros(size(timeprop));
thetaprop(1)=theta(propstartindex);
machprop=zeros(size(timeprop));
machprop(1)=Vprop(1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(hprop(1)));
dvx_dtprop=zeros(size(timeprop)-1);
```

```
dvy_dtprop=zeros(size(timeprop)-1);

%Absolute positions in planet reference frame

yprop=zeros(size(timeprop));
yprop(1)=y(propstartindex);

xprop=zeros(size(timeprop));
xprop(1)=x(propstartindex);

% forces

dragprop=zeros(size(timeprop)-1);
liftprop=zeros(size(timeprop)-1);
gravityprop=zeros(size(timeprop)-1);
thrust =zeros(size(timeprop)-1);

vehiclepropmass=zeros(size(timeprop));
vehiclepropmass(1)=vehicledescentmass; % vehicle mass at the beginning of
propulsive descent

% Time Step Routine
for i=1:(length(timeprop)-1)

dragprop(i)=0.5*AtmDensityPlanetfunc(hprop(i))*(Vprop(i)^2)*vehiclefrontalare
a*Cdprop;

liftprop(i)=0.5*AtmDensityPlanetfunc(hprop(i))*(Vprop(i)^2)*vehiclefrontalare
a*Cdprop*L_Dprop;
    gravityprop(i)=G*Mplanet/(rprop(i)^2);

    % no thrust before engine start
    if timeprop(i)<propdeploytime
        thrust(i)=0;
    else
        thrust(i)=thrustactual;
    end

     % to make sure lift keeps pointing upward
    if Vxprop(i)>0.0

    dvx_dtprop(i)=
liftprop(i)*cos(gammaprop(i)+pi()/2.0)/vehiclepropmass(i)+dragprop(i)*cos(pi(
)+gammaprop(i))/vehiclepropmass(i)
+thrust(i)*cos(pi()+gammaprop(i))/vehiclepropmass(i) -
gravityprop(i)*xprop(i)/rprop(i) + 2*angularrotplanet*Vyprop(i) +
(angularrotplanet^2)*xprop(i);
    dvy_dtprop(i)=
liftprop(i)*sin(gammaprop(i)+pi()/2.0)/vehiclepropmass(i)+dragprop(i)*sin(pi(
)+gammaprop(i))/vehiclepropmass(i)
+thrust(i)*sin(pi()+gammaprop(i))/vehiclepropmass(i) -
gravityprop(i)*yprop(i)/rprop(i) - 2*angularrotplanet*Vxprop(i) +
(angularrotplanet^2)*yprop(i);
```

```
    else
    dvx_dtprop(i)= liftprop(i)*cos(gammaprop(i)-
pi()/2.0)/vehiclepropmass(i)+dragprop(i)*cos(pi()+gammaprop(i))/vehiclepropma
ss(i) +thrust(i)*cos(pi()+gammaprop(i))/vehiclepropmass(i) -
gravityprop(i)*xprop(i)/rprop(i) + 2*angularrotplanet*Vyprop(i) +
(angularrotplanet^2)*xprop(i);
    dvy_dtprop(i)= liftprop(i)*sin(gammaprop(i)-
pi()/2.0)/vehiclepropmass(i)+dragprop(i)*sin(pi()+gammaprop(i))/vehiclepropma
ss(i) +thrust(i)*sin(pi()+gammaprop(i))/vehiclepropmass(i) -
gravityprop(i)*yprop(i)/rprop(i) - 2*angularrotplanet*Vxprop(i) +
(angularrotplanet^2)*yprop(i);

    end

    % no propellant burn before engine start
    if timeprop(i)<propdeploytime
        vehiclepropmass(i+1)=vehiclepropmass(i);
    else
        vehiclepropmass(i+1)=vehiclepropmass(i)-massflowrate*delta_tprop;
    end




    Vxprop(i+1)=Vxprop(i)+dvx_dtprop(i)*delta_tprop;
    Vyprop(i+1)=Vyprop(i)+dvy_dtprop(i)*delta_tprop;



    Vprop(i+1)=sqrt(Vxprop(i+1)^2+ Vyprop(i+1)^2);

      % correcting angle calculation in 3rd quadrant
    if Vxprop(i+1)<0.0
        gammaprop(i+1)=atan(Vyprop(i+1)/Vxprop(i+1)) + pi();
    else
        gammaprop(i+1)=atan(Vyprop(i+1)/Vxprop(i+1));
    end

    yprop(i+1)=yprop(i)+Vyprop(i)*delta_tprop;
    xprop(i+1)=xprop(i)+Vxprop(i)*delta_tprop;

    % correcting angle calculation in 3rd quadrant
    if xprop(i+1)<0.0
        thetaprop(i+1)=atan(yprop(i+1)/xprop(i+1)) + pi();
    else
        thetaprop(i+1)=atan(yprop(i+1)/xprop(i+1));
    end

    rprop(i+1)=sqrt(yprop(i+1)^2+xprop(i+1)^2);
    hprop(i+1)=rprop(i+1)-Rplanet;

     % setting range based on subtended angle, theta
    if xprop(i+1)>=0.0
        sprop(i+1)=Rplanet*(pi()/2.0-thetaprop(i+1));
```

```
    else
        sprop(i+1)=Rplanet*(5*pi()/2.0-thetaprop(i+1));
    end

    % Mach number calculation
    if hprop(i+1)<=atmos_interface_planet*1000

machprop(i+1)=Vprop(i+1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(hprop(i+1)
));
    else
        machprop(i+1)=0.0;
    end

     % stop calculation for velocity magnitude less than 1 m/s, x-velocity
     % less than 1 m/s and y-velocity greater than - 1 m/s (sink rate < 1
     % m/s)
    if (Vxprop(i+1)<=1.0 && Vyprop(i+1)>=-1.0) && Vprop(i+1)<=1.0

        for j=i+2:length(sprop)
            sprop(j)=sprop(i+1);
        end
        V_finalprop=Vprop(i+1);
        i_finalprop=i;
        break;
    end
    i_finalprop=i;
    V_finalprop=Vprop(i);
end

%results
V_finalprop;
h_finalprop=hprop(i_finalprop);
s_finalprop=sprop(i_finalprop);
mass_final=vehiclepropmass(i_finalprop);
fuelmass=vehicledescentmass-mass_final;
payloadmass=vehicledescentmass-fuelmass*(1+strucmassfraction);
payloadmassfraction=payloadmass/vehicleentrymass;
descenttime=timeprop(i_finalprop);
total_deltav=Isp*9.81*log(vehicledescentmass/mass_final);
```

***** End of Code *****

Filename: PropulsiveDescentSimFunc.m

***** Start of Code *****

```
function
h_final=PropulsiveDescentSimFunc(throttleratio,vehicledescentmass,Cd,L_D,Dia,
Isp,vx_initial,vy_initial,h_initial,s_initial,theta_initial,x_initial,y_initi
al)

%see comments for: PropulsiveDescentSim


% constants

global G Mplanet Rplanet kplanet Rgasplanet atmos_interface_planet
periodplanet angularrotplanet machparaoplimitplanet
global jettisonlimitcheck machjettisonlimit propdeploytime
global AtmDensityPlanetfunc AtmTempPlanetfunc




vehiclefrontalarea=pi()*(Dia^2)/4.0;

%engine parameters
thrustinitial= 1000000; %N
thrustactual=thrustinitial*throttleratio;


massflowrate=thrustactual/(Isp*9.81);

%simulation paramters
total_time=1000.0;
delta_t=0.05; % time step

% All quantities are relative to planet frame

%inputs
v_initial=sqrt(vx_initial^2+vy_initial^2);
gamma_initial=atan(vy_initial/vx_initial);

%initialize variables
time=[0.0:delta_t:total_time];
h=zeros(size(time));
h(1)=h_initial;
r=Rplanet+h;
s=zeros(size(time));
s(1)=s_initial;
V=zeros(size(time));
```

```
V(1)=v_initial;
Vx=zeros(size(time));
Vx(1)=vx_initial;
Vy=zeros(size(time));
Vy(1)=vy_initial;
gamma=zeros(size(time));
gamma(1)= gamma_initial;
theta=zeros(size(time));
theta(1)=theta_initial;
mach=zeros(size(time));
 mach(1)=V(1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(1)));
dvx_dt=zeros(size(time)-1);
dvy_dt=zeros(size(time)-1);

%Absolute positions in planet reference frame
y=zeros(size(time));
y(1)=y_initial;

x=zeros(size(time));
x(1)=x_initial;

%Forces
drag=zeros(size(time)-1);
lift=zeros(size(time)-1);
gravity =zeros(size(time)-1);
thrust =zeros(size(time)-1);
vehiclepropmass=zeros(size(time));
vehiclepropmass(1)=vehicledescentmass;

%Assume case is feasible i.e. the given thrust value is adequate to achieve
%the required velocity and altitude combination
isfeasible='y';

for i=1:(length(time)-1)
    drag(i)=0.5*AtmDensityPlanetfunc(h(i))*(V(i)^2)*vehiclefrontalarea*Cd;

lift(i)=0.5*AtmDensityPlanetfunc(h(i))*(V(i)^2)*vehiclefrontalarea*Cd*L_D;
    gravity(i)=G*Mplanet/(r(i)^2);

    % no thrust before engine start
    if time(i)<propdeploytime
        thrust(i)=0;
    else
        thrust(i)=thrustactual;
    end
```

```
    % to make sure lift keeps pointing upward
    if Vx(i)>0.0

        dvx_dt(i)=
lift(i)*cos(gamma(i)+pi()/2.0)/vehiclepropmass(i)+drag(i)*cos(pi()+gamma(i))/
vehiclepropmass(i) +thrust(i)*cos(pi()+gamma(i))/vehiclepropmass(i) -
gravity(i)*x(i)/r(i) + 2*angularrotplanet*Vy(i) + (angularrotplanet^2)*x(i);
        dvy_dt(i)=
lift(i)*sin(gamma(i)+pi()/2.0)/vehiclepropmass(i)+drag(i)*sin(pi()+gamma(i))/
vehiclepropmass(i) +thrust(i)*sin(pi()+gamma(i))/vehiclepropmass(i) -
gravity(i)*y(i)/r(i) - 2*angularrotplanet*Vx(i) + (angularrotplanet^2)*y(i);

    else

        dvx_dt(i)= lift(i)*cos(gamma(i)-
pi()/2.0)/vehiclepropmass(i)+drag(i)*cos(pi()+gamma(i))/vehiclepropmass(i)
+thrust(i)*cos(pi()+gamma(i))/vehiclepropmass(i) - gravity(i)*x(i)/r(i) +
2*angularrotplanet*Vy(i) + (angularrotplanet^2)*x(i);
        dvy_dt(i)= lift(i)*sin(gamma(i)-
pi()/2.0)/vehiclepropmass(i)+drag(i)*sin(pi()+gamma(i))/vehiclepropmass(i)
+thrust(i)*sin(pi()+gamma(i))/vehiclepropmass(i) - gravity(i)*y(i)/r(i) -
2*angularrotplanet*Vx(i) + (angularrotplanet^2)*y(i);

    end

    % no propellant burn before engine start
    if time(i)<propdeploytime
        vehiclepropmass(i+1)=vehiclepropmass(i);
    else
        vehiclepropmass(i+1)=vehiclepropmass(i)-massflowrate*delta_t;
    end

    Vx(i+1)=Vx(i)+dvx_dt(i)*delta_t;
    Vy(i+1)=Vy(i)+dvy_dt(i)*delta_t;

    V(i+1)=sqrt(Vx(i+1)^2+ Vy(i+1)^2);

     % correcting angle calculation in 3rd quadrant
    if  Vx(i+1)<0.0
        gamma(i+1)=atan(Vy(i+1)/Vx(i+1)) + pi();
    else
        gamma(i+1)=atan(Vy(i+1)/Vx(i+1));
    end

    y(i+1)=y(i)+Vy(i)*delta_t;
    x(i+1)=x(i)+Vx(i)*delta_t;

     % correcting angle calculation in 3rd quadrant
    if x(i+1)<0.0
        theta(i+1)=atan(y(i+1)/x(i+1)) + pi();
    else
        theta(i+1)=atan(y(i+1)/x(i+1));
    end
```

```
    r(i+1)=sqrt(y(i+1)^2+x(i+1)^2);
    h(i+1)=r(i+1)-Rplanet;

    % setting range based on subtended angle, theta
    if x(i+1)>=0.0
        s(i+1)=Rplanet*(pi()/2.0-theta(i+1));
    else
        s(i+1)=Rplanet*(5*pi()/2.0-theta(i+1));
    end

    % Mach number calculation
    if h(i+1)<=atmos_interface_planet*1000
         mach(i+1)=V(i+1)/sqrt(kplanet*Rgasplanet*AtmTempPlanetfunc(h(i+1)));
    else
        mach(i+1)=0.0;
    end

    % If mass becomes negative, stop simulation and output infeasible
    % condition
     if vehiclepropmass(i+1)<=0.0
         isfeasible='n';
         break;
     end

     % stop calculation for velocity magnitude less than 1 m/s, x-velocity
     % less than 1 m/s and y-velocity greater than - 1 m/s (sink rate < 1
     % m/s)
    if (Vx(i+1)<=1.0 && Vy(i+1)>=-1.0) && V(i+1) <=1.0
        for j=i+2:length(s)
            s(j)=s(i+1);
        end
        V_final=V(i+1);
        i_final=i;
        break;
    end
    i_final=i;
    V_final=V(i);
end

%results

%For infeasible case, output high final altitude value. Since bisection
search is
%trying to get thrust value that will get vehicle closest to surface, the
%search will move away from this thrust value
if isfeasible=='n'
    h_final=70000.0;
end
h_final=h(i_final);
```

***** End of Code *****

Filename: SingleEDLTrajectorysim.m

***** Start of Code *****

```
clear all
close all
clc

% global constant reading

Planetname=input('Enter planet name for entry, descent and landing
simulation:','s');


% planet constants

global G Mplanet Rplanet kplanet Rgasplanet atmos_interface_planet
periodplanet angularrotplanet machparaoplimitplanet

G=6.673e-11; % Universal Gravitational Constant

% Reading planet constants from file

planetconstantsfile=input('Enter name of text file (including extension)
containing the list of planet constants: \n','s');
planetconstants=dlmread(planetconstantsfile,'\n');

Mplanet=planetconstants(1); %Mass of Planet (kg)
Rplanet=planetconstants(2); % Mean Planet Radius (km)
kplanet=planetconstants(3); %Ratio of Specific Heats for Atmosphere
Rgasplanet=planetconstants(4);% Gas Constant for Atmosphere (J/kg/K)
atmos_interface_planet=planetconstants(5); %Atmospheric Interface Radius (km)
periodplanet=planetconstants(6); %Sidereal Rotation Period (hr)
machparaoplimitplanet=planetconstants(7); %Operational Limit for Parachutes
in Planetary Atmosphere



%planet constant adjustments & related calculations
Rplanet=Rplanet*1000.0;
periodplanet=periodplanet*60*60; %Sidereal Rotation Period (s)
rotdirection=input('Enter 1 for entering in direction of planet sidereal
rotation \nEnter -1 for entering in direction opposite to planet sidereal
rotation \n');
angularrotplanet=-rotdirection*2*pi()/periodplanet; %Sidereal Rotational
Speed (rad/s)


global jettisonlimitcheck machjettisonlimit propdeploytime
propseparationmassfraction
```

```
% Atmospheric data source specification

global AtmDensityPlanetfunc AtmTempPlanetfunc



AtmDensityPlanet=input('Enter the name for the Atmospheric Density Profile
function file (input=altitude in m)) \n','s');
AtmTempPlanet=input('Enter the name for the Atmospheric Temperature Profile
function file (input=altitude in m)) \n','s');

AtmDensityPlanetfunc=str2func(AtmDensityPlanet);
AtmTempPlanetfunc=str2func(AtmTempPlanet);



%Default Variable Values
Cd=1.5; %Drag Coefficient
L_D=0.3; % Lift-over-Drag Ratio
Dia=12; %Vehicle diameter (m)
vehicleentrymass=50; %metric tons
aeroshellmassfraction=0.68;
strucmassfraction=0.65; % Descent Propulsion System Structural Mass Fraction
landalt=-4.0; % Desired Landing Site Elevation (measured from Mean Planet
Radius)(km)
descentalt=5; % Desired Propulsive Descent Initiation Altitude Above Landing
Site(km)
entry_apo_alt=500.0; %  Entry Orbit Apoapsis  Altitude (km)
entry_peri_alt=50.0; % Entry Orbit Periapsis Altitude (km)
propdeploytime=5.0; % Time between aeroshell jettison and engine start
deploytimecheck='n'; % Boolean  variable to allow user to change deploy time
propseparationmassfraction=0.1; % Mass penalty fraction for propulsive
separation of aeroshell
propseparationcheck='n'; % Boolean variable to allow user to change mass
penalty fraction

Cdprop=1.2; % Descent Configuration Drag Coefficient
L_Dprop=0.0; % Descent Configuration Lift over Drag Ratio
Isp=379; % Descent Engine Isp

changedefault='n'; % Boolean variable to allow user to change default values
of parameters



%Variable name coding
ndefaultparameters=13;
defaultparameters=cell(ndefaultparameters,3);
parameterindex=1:1:ndefaultparameters;
parameternames={'Drag Coefficient (Cd)'; 'L/D'; 'Diameter (m)';'Entry Mass
(metric tons)';...
```

```
'Aeroshell Mass Fraction'; 'Descent Propulsion System Structural Mass
Fraction';'Landing Altitude (m)';...
'Propulsive Descent Start Altitude (km)'; 'Entry Orbit Apoapsis Altitude
(km)';...
'Entry Orbit Periapsis Altitude (km)'; 'Descent Drag Coefficient'; 'Descent
L/D'; 'Engine Isp'};
parameternamesshort={'Cd'; 'L/D'; 'Dia';'Entry Mass';'AMF'; 'DSMF';'Landing
Alt';'PDAlt'; 'Orbit Apo';'Orbit Peri'; 'Descent Cd'; 'Descent L/D'; 'Engine
Isp'};
units={'none'; 'none'; 'm'; 'mT (metric tons)'; 'none'; 'none'; 'km'; 'km';
'km'; 'km'; 'none'; 'none'; 's'};
for i=1:ndefaultparameters
 defaultparameters{i,1}=parameterindex(i);
 defaultparameters{i,2}=parameternamesshort{i};
 defaultparameters{i,4}=units{i};
end

k=ones(ndefaultparameters,1);
tableheadings={'#' 'Parameter Name' 'Value' 'Unit'};

defaultparameters{1,3}=Cd;
defaultparameters{2,3}=L_D;
defaultparameters{3,3}=Dia;
defaultparameters{4,3}= vehicleentrymass;
defaultparameters{5,3}=aeroshellmassfraction;
defaultparameters{6,3}=strucmassfraction;
defaultparameters{7,3}=landalt;
defaultparameters{8,3}=descentalt;
defaultparameters{9,3}=entry_apo_alt;
defaultparameters{10,3}=entry_peri_alt;
defaultparameters{11,3}=Cdprop;
defaultparameters{12,3}=L_Dprop;
defaultparameters{13,3}=Isp;


% User Input Block


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Changing Default Parameter Values Block

disp('The current default parameter values are:')
disp(tableheadings)
disp(defaultparameters)
changedefault=input('Would you like to change the default value of any of the
fixed parameters (y/n) ? \nNote: this choice will not affect the parameters
you have chosen as variables above. \n','s');
addblankline

while changedefault=='y'
```

```
    changeparameterindex=input('Choose the number of the parameter to
change:');
    addblankline
    disp('You have chosen to change the default value for:');
    disp(defaultparameters{changeparameterindex,2})
    addblankline
    defaultparameters{changeparameterindex,3}=input('Please enter the new
value of this parameter:');
    addblankline

    disp('The current default parameter values are:')
    disp(tableheadings)
    disp(defaultparameters)

    changedefault=input('Would you like to change the default value of any of
the fixed parameters (y/n) ? \n','s');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Aeroshell Jettison Limits Check and Input

jettisonlimitcheck=input('Is there an upper mach number limit for release of
aeroshell/backshell (y/n) ?\n','s');
if jettisonlimitcheck=='y'
    machjettisonlimit=input('What is this limit? \n');
    addblankline
    if machjettisonlimit>machparaoplimitplanet
        disp('Separation may take place at speeds faster than the operational
envelope of parachutes.')
        disp('Therefore, propulsive means might be used.')
        addblankline
        disp('The current mass overhead fraction for propulsive separation
is:')
        disp(propseparationmassfraction)
        propseparationcheck=input('Would you like to change this overhead
(y/n) ? \n','s');
        if propseparationcheck=='y'
            propseparationmassfraction=input('Enter the new mass overhead
fraction for propulsive separation:');
        end
    end
elseif jettisonlimitcheck=='n'
    disp('Separation may take place at speeds faster than the operational
envelope of parachutes.')
    disp('Therefore, propulsive means might be used')
    disp('The current mass overhead fraction for propulsive separation is:')
    disp(propseparationmassfraction)
    addblankline
    propseparationcheck=input('Would you like to change this overhead (y/n) ?
\n','s');
    addblankline
    if propseparationcheck=='y'
        propseparationmassfraction=input('Enter the new mass overhead
fraction for propulsive separation:');
```

```
    end
end
addblankline

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% time to engine start from aeroshell jettison setting block

deploytimecheck=input('Would you like to specify a propulsion system
deployment time (y/n)? Default is 5 seconds. \n','s');
if deploytimecheck=='y'
    propdeploytime=input('Please specify the propulsion system deployment
time in seconds: \n');
end
addblankline

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Cd=defaultparameters{1,3};
L_D=defaultparameters{2,3};
Dia=defaultparameters{3,3};
vehicleentrymass=defaultparameters{4,3};
aeroshellmassfraction=defaultparameters{5,3};
strucmassfraction=defaultparameters{6,3};
landalt=defaultparameters{7,3};
descentalt=defaultparameters{8,3};
entry_apo_alt=defaultparameters{9,3};
entry_peri_alt=defaultparameters{10,3};
Cdprop=defaultparameters{11,3};
L_Dprop=defaultparameters{12,3};
Isp=defaultparameters{13,3};



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




% adjustments

vehiclefrontalarea=pi()*(Dia^2)/4.0;
vehicleentrymass=vehicleentrymass*1000.0; %kg
landalt=landalt*1000;
descentalt=descentalt*1000;
entry_apo_alt=entry_apo_alt*1000;
entry_peri_alt=entry_peri_alt*1000;

% Aerodynamic Entry Simulation

Entrysim
```

```
%Vehicle Propuslive Descent parameters

propstartindex=i_final; % time at propulsive descent start


% If Propulsive Descent Initiation altitude is below altitude, do not run
% simulation
if h(propstartindex) <= landalt

    disp('Case is infeasible')
    disp('Propulsive Descent Initiation Altitude is below Desired Landing
Altitude')


else


    if mach(propstartindex) <= machparaoplimitplanet % check if parachutes
can be used

        vehicledescentmass=vehicleentrymass/(1+aeroshellmassfraction); %mass
at start of propulsive maneouvre
    else % mass penalty because of propulsive separation

        vehicledescentmass=vehicleentrymass/(1+aeroshellmassfraction+propsepa
        rationmassfraction); %mass at start of propulsive maneouvre
    end


    % Descent Propulsion System Sizing Routine: using bisection search
algorithm
    % Objective: try to get vehicle as close as possible to desired landing
    % altitude with 1 m/s velocity
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    A=0.01; % bisection search lower bound
    B=2.0; % bisection search upper bound
    C=(A+B)/2.0;
    count=1;
    tolerance=0.01;

    while abs(A-B)>tolerance
        count;

        fA=PropulsiveDescentSimFunc(A,vehicledescentmass,Cdprop,L_Dprop,Dia,I
        sp,Vx(propstartindex),Vy(propstartindex),h(propstartindex),s(propstar
        tindex),theta(propstartindex),x(propstartindex),y(propstartindex));

        fB=PropulsiveDescentSimFunc(B,vehicledescentmass,Cdprop,L_Dprop,Dia,I
        sp,Vx(propstartindex),Vy(propstartindex),h(propstartindex),s(propstar
        tindex),theta(propstartindex),x(propstartindex),y(propstartindex));

        fC=PropulsiveDescentSimFunc(C,vehicledescentmass,Cdprop,L_Dprop,Dia,I
```

```
        sp,Vx(propstartindex),Vy(propstartindex),h(propstartindex),s(propstar
        tindex),theta(propstartindex),x(propstartindex),y(propstartindex));

        if fC > landalt
            B = C;
            A = A;

        else
            A = C;
            B = B;
        end
        C=(A+B)/2.0;

        count=count+1;
    end

    throttleratiodesired=(A+B)/2.0;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % get required thrust from above routine

    % Propulsive Descent Simulation
    PropulsiveDescentSim

    % Plot Trajectory

    figure
    set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})

     plot(s(1:propstartindex)./1000.0,h(1:propstartindex)./1000.0,sprop(1:i_fi
     nalprop)./1000.0,hprop(1:i_finalprop)./1000.0,'--m')
    figure1handle=gca;
    set(figure1handle,'FontSize',24)
    xlabel('Range (km)','FontSize',30)
    ylabel('Altitude (km)','FontSize',30)
    title('Entry Vehicle Trajectory','FontSize',30)
    legend('Entry Trajectory','Descent Trajectory')

    %Plot Velocity vs. Altitude

    figure
    set(0,'DefaultAxesLineStyleOrder',{'-','--',':+'})

     plot(V(1:propstartindex)./1000.0,h(1:propstartindex)./1000.0,'k',Vprop(1:
     i_finalprop)./1000.0,hprop(1:i_finalprop)./1000.0,'m')
    hold on

    plot(mach2v./1000.0,htest./1000.0,'r',mach3v./1000.0,htest./1000.0,'g',mac
    h4v./1000.0,htest./1000.0,'b',mach5v./1000.0,htest./1000.0,'c')
    figure2handle=gca;
    set(figure2handle,'FontSize',24)
    legend('Entry Trajectory','Descent Trajectory','Mach 2', 'Mach 3', 'Mach
4','Mach 5')
    xlabel('Velocity (km/s)','FontSize',30)
    ylabel('Altitude (km)','FontSize',30)
```

```
    title(' Altitude vs Velocity for Entry Vehicle','FontSize',30)

    % find maximum g's experienced by vehicle
    accel=sqrt(dvx_dt.^2+dvy_dt.^2);
    accelprop=sqrt(dvx_dtprop.^2+dvy_dtprop.^2);
    earthg=accel/9.80665;
    earthgprop=accelprop/9.8065;
    Maxg=max(earthg);
    Maxgprop=max(earthgprop);
    Maxgtotal=max([Maxg Maxgprop]);
    payloadmassfraction
    throttleratiodesired
    V_finalprop;
    h_finalprop=hprop(i_finalprop)
    s_finalprop=sprop(i_finalprop);
    descenttime=timeprop(i_finalprop)

end

%Play sound at simulation completion

simulationcomplete = wavread('simulationcomplete.wav');
sound(simulationcomplete,22500)
```

***** End of Code *****

Filename: SingleEntryTrajectorysim.m

***** Start of Code *****

```
clear all
close all
clc

% global constant reading

Planetname=input('Enter planet name for entry, descent and landing
simulation:','s');

% planet constants

global G Mplanet Rplanet kplanet Rgasplanet atmos_interface_planet
periodplanet angularrotplanet machparaoplimitplanet

G=6.673e-11; % Universal Gravitational Constant

% Reading planet constants from file

planetconstantsfile=input('Enter name of text file (including extension)
containing the list of planet constants: \n','s');
planetconstants=dlmread(planetconstantsfile,'\n');

Mplanet=planetconstants(1); %Mass of Planet (kg)
Rplanet=planetconstants(2); % Mean Planet Radius (km)
kplanet=planetconstants(3); %Ratio of Specific Heats for Atmosphere
Rgasplanet=planetconstants(4);% Gas Constant for Atmosphere (J/kg/K)
atmos_interface_planet=planetconstants(5); %Atmospheric Interface Radius (km)
periodplanet=planetconstants(6); %Sidereal Rotation Period (hr)
machparaoplimitplanet=planetconstants(7); %Operational Limit for Parachutes
in Planetary Atmosphere


%planet constant adjustments & related calculations

Rplanet=Rplanet*1000.0;
periodplanet=periodplanet*60*60; %Sidereal Rotation Period (s)
rotdirection=input('Enter 1 for entering in direction of planet sidereal
rotation \nEnter -1 for entering in direction opposite to planet sidereal
rotation \n');
angularrotplanet=-rotdirection*2*pi()/periodplanet; %Sidereal Rotational
Speed (rad/s)


global jettisonlimitcheck machjettisonlimit propdeploytime
propseparationmassfraction
```

```
% Atmospheric data source specification

global AtmDensityPlanetfunc AtmTempPlanetfunc


AtmDensityPlanet=input('Enter the name for the Atmospheric Density Profile
function file (input=altitude in m)) \n','s');
AtmTempPlanet=input('Enter the name for the Atmospheric Temperature Profile
function file (input=altitude in m)) \n','s');

AtmDensityPlanetfunc=str2func(AtmDensityPlanet);
AtmTempPlanetfunc=str2func(AtmTempPlanet);



%Default Variable Values
Cd=1.5; %Drag Coefficient
L_D=1.0; % Lift-over-Drag Ratio
Dia=12; %Vehicle diameter (m)
vehicleentrymass=100; %metric tons
landalt=-4.0; % Desired Landing Site Elevation (measured from Mean Planet
Radius)(km)
descentalt=0; % Desired Trajectory Calculation Stop Altitude Above Landing
Site(km)
entry_apo_alt=500.0; %  Entry Orbit Apoapsis  Altitude (km)
entry_peri_alt=50.0; % Entry Orbit Periapsis Altitude (km)


changedefault='n'; % Boolean variable to allow user to change default values
of parameters


%Variable name coding

ndefaultparameters=8;
defaultparameters=cell(ndefaultparameters,3);
parameterindex=1:1:ndefaultparameters;
parameternames={'Drag Coefficient (Cd)'; 'L/D'; 'Diameter (m)';'Entry Mass
(metric tons)';...
'Landing Altitude (m)';...
'Trajectory Calculation Stop Altitude (km)'; 'Entry Orbit Apoapsis Altitude
(km)';...
'Entry Orbit Periapsis Altitude (km)'};
parameternamesshort={'Cd'; 'L/D'; 'Dia';'Entry Mass';'Landing Alt';'StopAlt';
'Orbit Apo';'Orbit Peri'};
units={'none'; 'none'; 'm'; 'mT (metric tons)'; 'km'; 'km'; 'km'; 'km'};
for i=1:ndefaultparameters
 defaultparameters{i,1}=parameterindex(i);
 defaultparameters{i,2}=parameternamesshort{i};
 defaultparameters{i,4}=units{i};
end

k=ones(ndefaultparameters,1);
```

```
tableheadings={'#' 'Parameter Name' 'Value' 'Unit'};

defaultparameters{1,3}=Cd;
defaultparameters{2,3}=L_D;
defaultparameters{3,3}=Dia;
defaultparameters{4,3}= vehicleentrymass;
defaultparameters{5,3}=landalt;
defaultparameters{6,3}=descentalt;
defaultparameters{7,3}=entry_apo_alt;
defaultparameters{8,3}=entry_peri_alt;


% User Input Block


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Changing Default Parameter Values Block

disp('The current default parameter values are:')
disp(tableheadings)
disp(defaultparameters)
changedefault=input('Would you like to change the default value of any of the
fixed parameters (y/n) ? \nNote: this choice will not affect the parameters
you have chosen as variables above. \n','s');
addblankline

while changedefault=='y'

    changeparameterindex=input('Choose the number of the parameter to
change:');
    addblankline
    disp('You have chosen to change the default value for:');
    disp(defaultparameters{changeparameterindex,2})
    addblankline
    defaultparameters{changeparameterindex,3}=input('Please enter the new
value of this parameter:');
    addblankline

    disp('The current default parameter values are:')
    disp(tableheadings)
    disp(defaultparameters)

    changedefault=input('Would you like to change the default value of any of
the fixed parameters (y/n) ? \n','s');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Cd=defaultparameters{1,3};
L_D=defaultparameters{2,3};
Dia=defaultparameters{3,3};
vehicleentrymass=defaultparameters{4,3};
landalt=defaultparameters{5,3};
descentalt=defaultparameters{6,3};
entry_apo_alt=defaultparameters{7,3};
entry_peri_alt=defaultparameters{8,3};




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




% adjustments

vehiclefrontalarea=pi()*(Dia^2)/4.0;
vehicleentrymass=vehicleentrymass*1000.0; %kg
landalt=landalt*1000;
descentalt=descentalt*1000;
entry_apo_alt=entry_apo_alt*1000;
entry_peri_alt=entry_peri_alt*1000;

% Aerodynamic Entry Simulation

Entrysimonly

%Trajectory Calculation Stop

stopindex=i_final; % End of entry simulation

%Plot Trajectory

figure
plot(s(1:stopindex)./1000.0,h(1:stopindex)./1000.0)
figure1handle=gca;
set(figure1handle,'FontSize',24)
xlabel('Range (km)','FontSize',30)
ylabel('Altitude (km)','FontSize',30)
title('Entry Vehicle Trajectory','FontSize',30)
legend('Entry Trajectory')

%Plot Velocity vs. Altitude

figure
plot(V(1:stopindex)./1000.0,h(1:stopindex)./1000.0,'k')
hold on
plot(mach2v./1000.0,htest./1000.0,'r',mach3v./1000.0,htest./1000.0,'g',mach4v
./1000.0,htest./1000.0,'b',mach5v./1000.0,htest./1000.0,'c')
```

```
figure2handle=gca;
set(figure2handle,'FontSize',24)
legend('Entry Trajectory','Mach 2', 'Mach 3', 'Mach 4','Mach 5')
xlabel('Velocity (km/s)','FontSize',30)
ylabel('Altitude (km)','FontSize',30)
title(' Altitude vs Velocity for Entry Vehicle','FontSize',30)

% find maximum g's experienced by vehicle
accel=sqrt(dvx_dt.^2+dvy_dt.^2);

earthg=accel/9.80665;

Maxg=max(earthg);

%Find freestream conditions at maximum dynamic pressure

[qmax,indexqmax]=max(dynp)
machqmax=mach(indexqmax)
hqmax=h(indexqmax)
densityqmax=Marsdensityfitted(hqmax)
tempqmax=Marstempfitted(hqmax)
pressureqmax=Marspressurefitted(hqmax)

%Find stagnation conditions at maximum dynamic pressure

stagtempqmax= tempqmax*(1 + (kplanet-1)*(machqmax^2)/2)
stagpressureqmax=pressureqmax*(1 + (kplanet-
1)*(machqmax^2)/2)^(kplanet/(kplanet-1))

%Play sound at simulation completion

simulationcomplete = wavread('simulationcomplete.wav');
sound(simulationcomplete,22500)
```

***** End of Code *****

Filename: Marsdensityfitted.m

***** Start of Code *****

```
function [density] =Marsdensityfitted(h)

%input: Altitude in m
% function: calculates density of the Martian atmosphere at a certain
% altitude using a curve fit obtained from Mars Pathfinder EDL data
%(See Appendix F of thesis)

h=h/1000.0;

density = 0.02045*exp(-0.1038*h);
```

***** End of Code *****

Filename: Marstempfitted.m

***** Start of Code *****

```
function [temperature] =Marstempfitted(h)


%input: Altitude in m
% function: calculates temperature of the Martian atmosphere at a certain
% altitude using a curve fit obtained from Mars Pathfinder EDL data
%(See Appendix F of thesis)

Rgasmars=188.92;% J/kg/K gas constant for Mars atmosphere
h=h/1000.0;
density = 0.02045*exp(-0.1038*h);
pressure = 714.3*exp(-0.1012*h);
temperature = pressure/(density*Rgasmars);
```

***** End of Code *****

Appendix C: Issues Encountered during EDL Modeling

One of the main issues encountered during Matlab modeling of EDL was the angle calculations using the inverse tangent function.

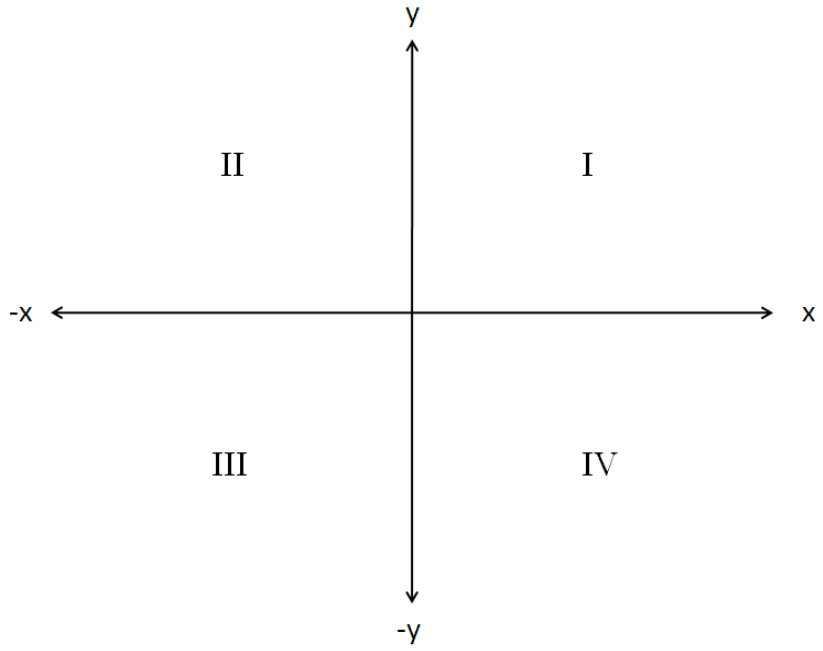

Figure C-1: Rectangular Axes Diagram with Quadrants Marked

If the arctan or atan (in Matlab) function is provided with a positive input, it gives an angle in the first quadrant and if given a negative input, provides a negative acute output i.e. the angle lies in the fourth quadrant (See Figure C-1). Therefore, for data points that lie in the second or third quadrants, the output produced is erroneous. After empirical investigation, it was discovered that the following fix works:

Check to see if data point is in second or third quadrant using the x value (of position or velocity as the case demands) i.e. if the x value is negative:
Add 180° to the angle produced by the atan function.

This technique was used to correct both γ and θ. The one issue with the θ calculation is that it is measured counterclockwise from the positive x axis. However, the range calculation which depends on θ needs the clockwise angle from the positive y axis. Therefore, the following formula was implemented for the calculation of the range based on the θ fix described above.

```
if x(i+1)>=0.0
    s(i+1)=Rplanet*(pi()/2.0-theta(i+1)); % formula for I & IV
quadrant
else
    s(i+1)=Rplanet*(5*pi()/2.0-theta(i+1)); % formula for II & III
quadrant
end
```

Appendix D: APAS Validation

Before conducting aerodynamic analysis for the reference vehicle shape used in this study, the simulation software APAS was used to run the simple case of a flat plate in order to verify functionality. One reason for this was that APAS is commonly used to find aerodynamic coefficients but the functionality to find pressure coefficients is not often used. In this work, the pressure coefficient data was also needed by another member of the research group and the flat plate provided a simple shape to evaluate the pressure coefficient results. Following is the screenshot of a flat plate of 1 m^2 area that was used for the simulation. The figure shows the panels created for the simulation.

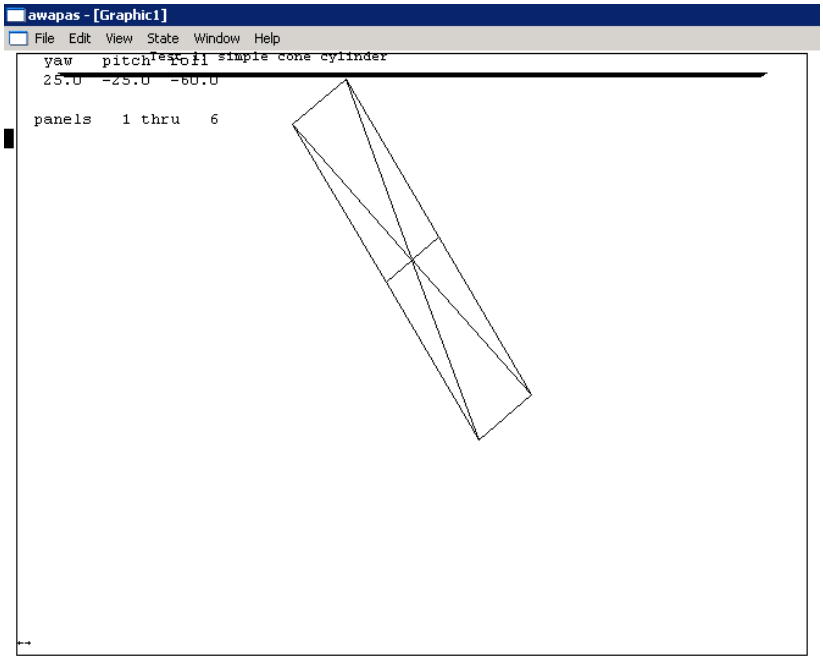

Figure D-1: Flat Plate (Area: 1 m^2) Geometry for APAS Validation including Panels Created for the Simulation

Run conditions were set to a Mach number of 5 at an Earth altitude of 20000 meters which set the pressure and temperature conditions. The drag coefficient c calculated for a zero angle of attack was 0.039168. The following are the results calculated for panels on the face of the plate.

```
flat plate test blun  0.00  0.00  5.00   20000.

ll   xc       yc       zc       area      cp     tw-r   q-b/sfs l/t
h-b/hrsfr

1  -0.0033  -1.0938   0.2706   1.3320  -0.0391    78.   2.185     0.0
17.889

2  -0.0033  -2.1875   0.8175   2.6910  -0.0391    78.   2.185     0.0
17.889
```

```
3  -0.0033  -1.0938   1.3644   1.3590  -0.0391    78.   2.185     0.0
17.889
```

Note that the pressure coefficient, cp values are the same as the drag coefficient. This is because the drag is obtained by integrating pressure forces over the body and in this case, the area was 1 so the values of pressure and drag coefficients are the same as was expected.

This verified that the program was providing correct pressure coefficient and drag values.

Appendix E: Biconics Overview

Within the aerospace literature, biconics are considered to be an important class of shapes for use as entry vehicles for planetary exploration. In order to gain an idea of their aerodynamic characteristics and their application to Mars EDL, a thorough literature search was conducted. Many references contain data about the lift-to-drag ratio of biconics. Relatively little data is available about their drag properties, however. This appendix presents data from papers that contained both the drag coefficients and lift-to-drag data. It is intended as a quick reference for biconic aerodynamic characteristics. Each of the subsections provides data from a specific reference work.

General Biconic Aerodynamics Range

Reference: "Generic aerocapture atmospheric entry study", Final Report General Electric Co., Philadelphia, PA. Re-Entry Systems Div., 1980.

The following figure presents an aggregate of data available about biconics.

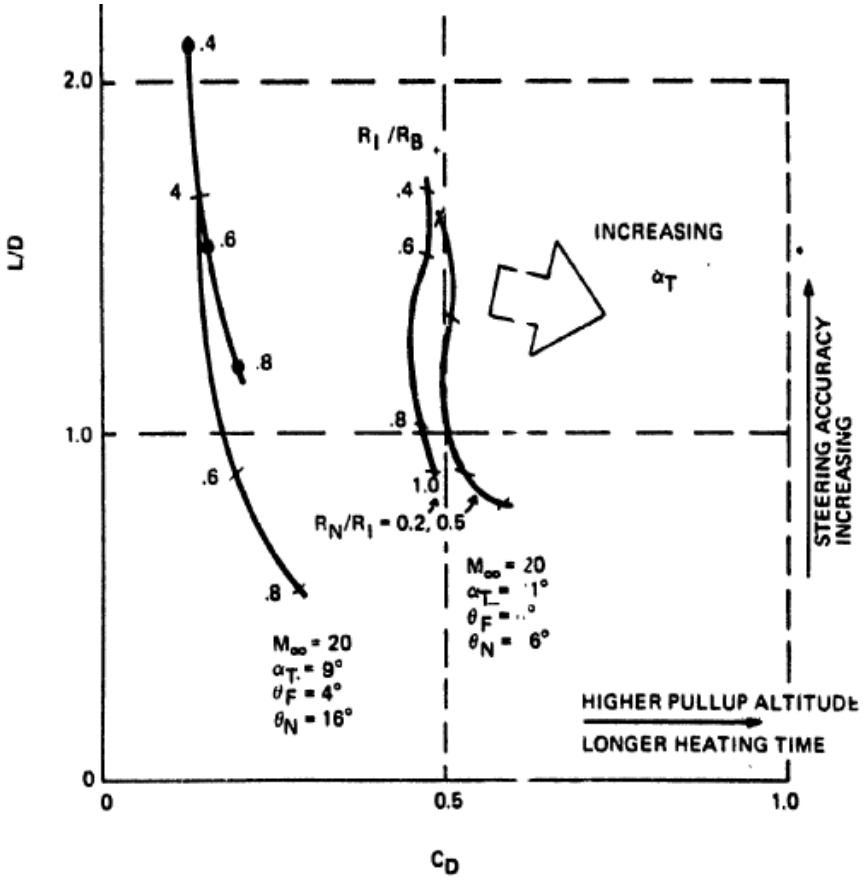

Figure E-1: Existing Biconic Database

Comparison of 5° cone and 10°/5° biconic

Reference: Stetson, Lewis. “Aerodynamic comparison of a conical and biconic reentry vehicle”, AIAA Guidance and Control Conference 1977 A82-13979 A77-43187

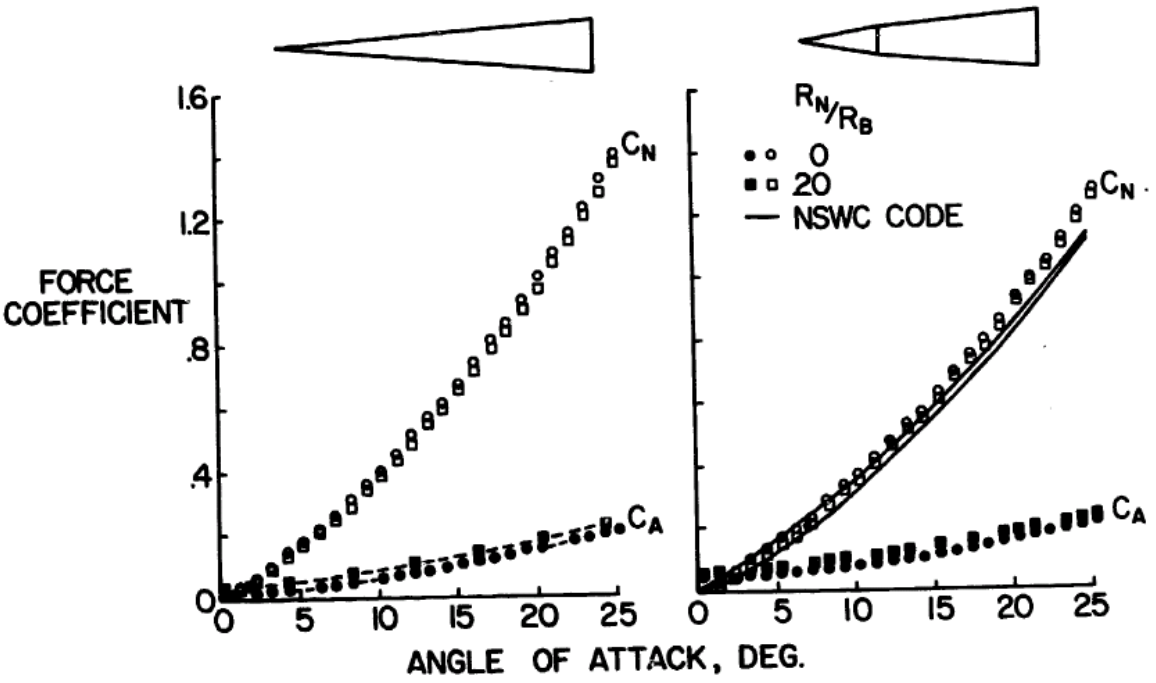

Figure E-2: Normal and Axial Force Coefficients versus Angle of Attack

A biconic with high drag: 20°/4°

Reference: Jits, Roman; Wright, Michael and Chen, Y.-K., "Closed-Loop Trajectory Simulation for Thermal Protection System Design for Neptune Aerocapture", Journal of Spacecraft and Rockets, Vol. 42, No. 6, November–December 2005, AIAA-13428-390.

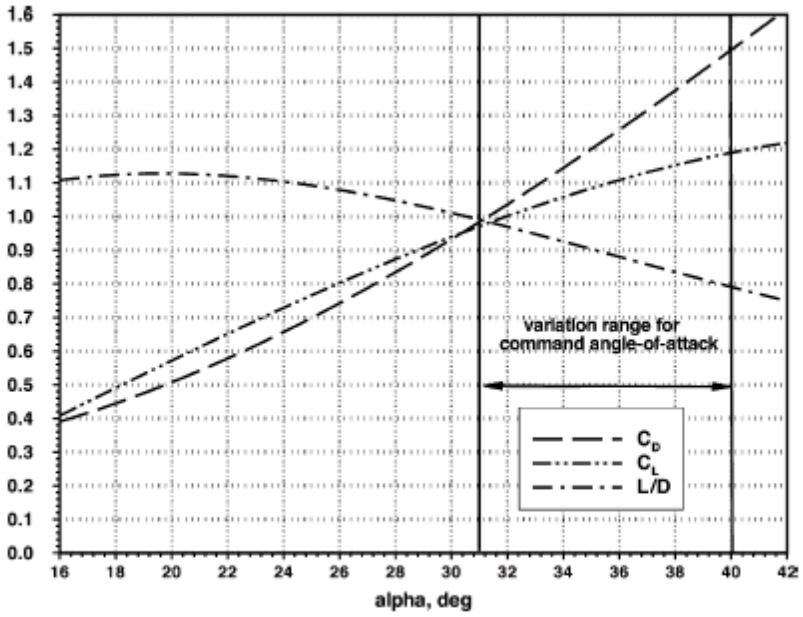

Figure E-3: Real-gas Aerodynamic Data for 20°/4° Biconic Vehicle

Appendix F: Martian Atmosphere Data

The Martian atmospheric model used in this study was created using data from the EDL trajectory of the Pathfinder mission, obtained from the Planetary Atmospheres Node of NASA's Planetary Data System[††].Plots of the data with curve fits can be found below for pressure and density. The temperature was found from density and pressure data using the ideal gas law. The source data can be found on the attached CD.

The equations for the curve fit are as follows:

```
density = 0.02045*exp(-0.1038*h)
pressure = 714.3*exp(-0.1012*h)
```

where h is altitude.

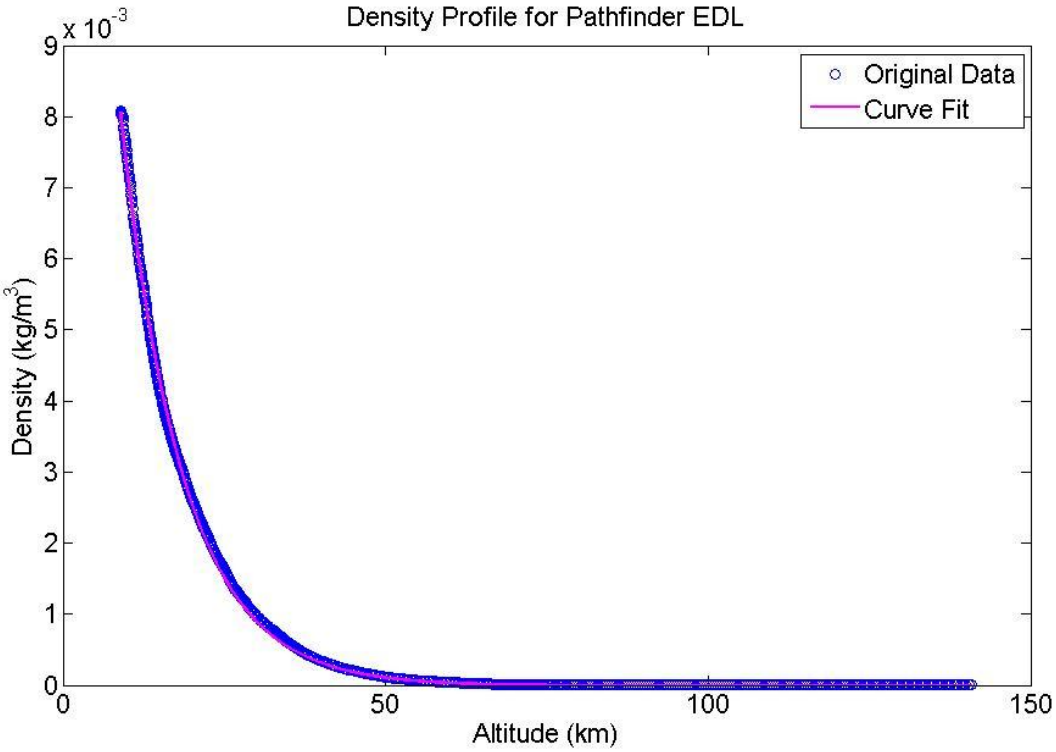

Figure F-1: Curve Fit for Mars Atmsopheric Density Profile using Pathfinder EDL data

[††] http://pds-atmospheres.nmsu.edu/cgi-bin/getdir.pl?dir=index&volume=mpam_0001

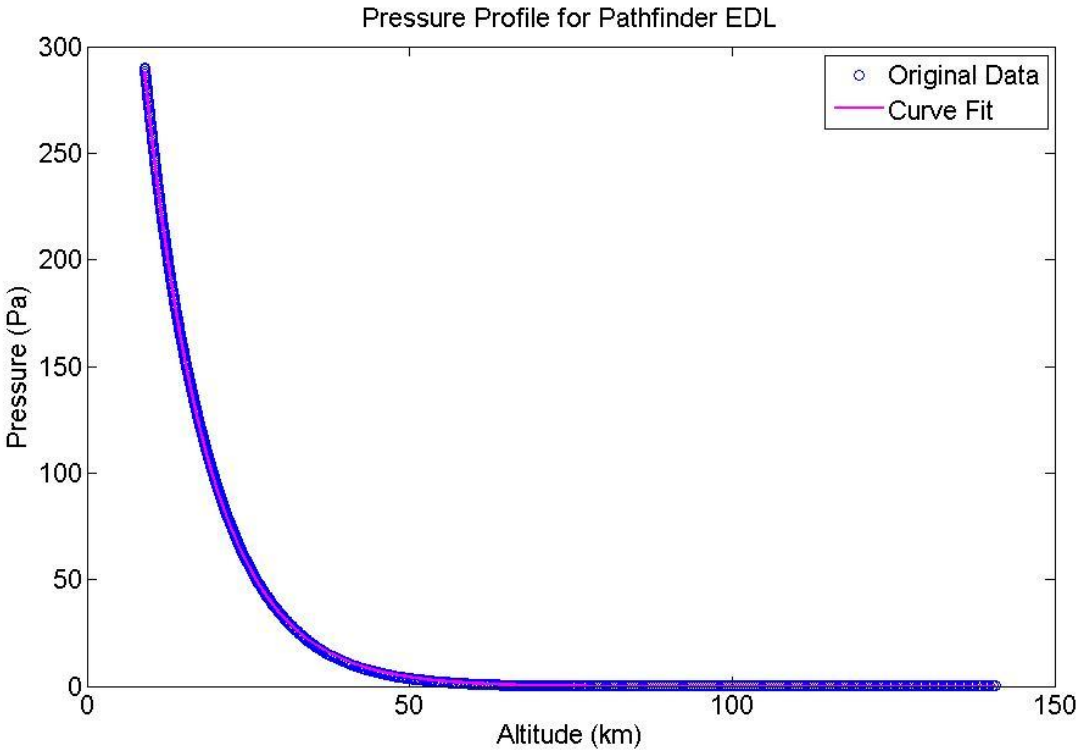

Figure F-2: Curve Fit for Mars Atmsopheric Pressure Profile using Pathfinder EDL data

Appendix G: Ares V Performance Details

(Note: all work presented in this section has been adapted from (4))

The earth departure strategy for crewed Mars missions using the Ares V launch vehicle has yet to be decided. This strategy consists of determining the propulsion stages required and the appropriate staging scenarios and locations. In the current configuration of the Ares V, the upper stage is the Earth Departure Stage (EDS).

A trade study for different earth departure options was conducted in the author's research group. Promising options that emerged from this trade study are as follows:

- Option 1: 3-launch departure using custom Trans-Mars Injection (TMI) stage
    - This option requires three Ares V launches which deliver various elements into earth orbit
        - Payload is launched first into Low Earth Orbit (LEO)
        - TMI-1 is launched second, later used for TMI
        - TMI-2 is launched third, docks with and boosts payload and TMI-1 into High Earth Orbit (HEO)
- Option 3: 2-launch departure using EDS + custom TMI stage
    - This option requires two Ares V launches which deliver various elements into earth orbit
        - Payload is launched first into LEO with custom TMI stage
        - Earth Departure Stage is launched second, remaining propellant is used to boost payload and TMI stage into HEO
    - Option 3.1: TMI uses LOX / LH_2 propellants

The figures below show a graphical representation of the options described and the graph shows the TMI performance of the different options.

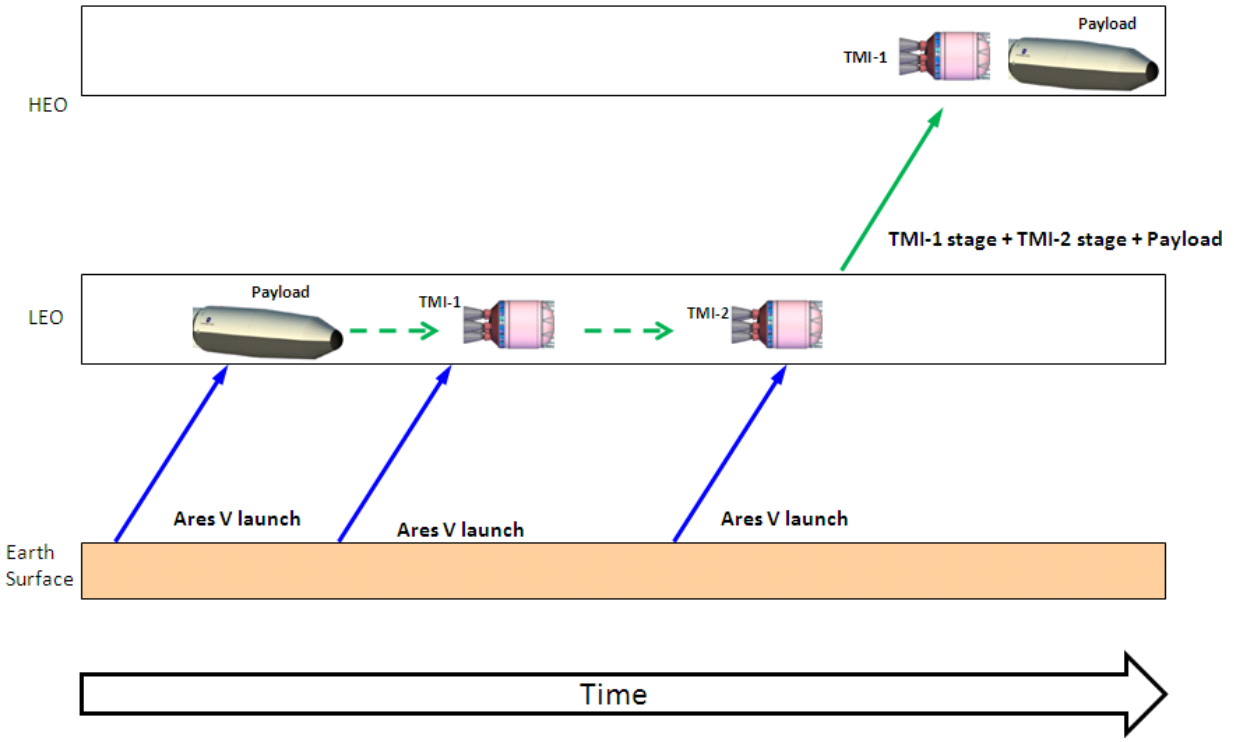

Figure G-1: Graphical representation of earth departure strategy Option 1

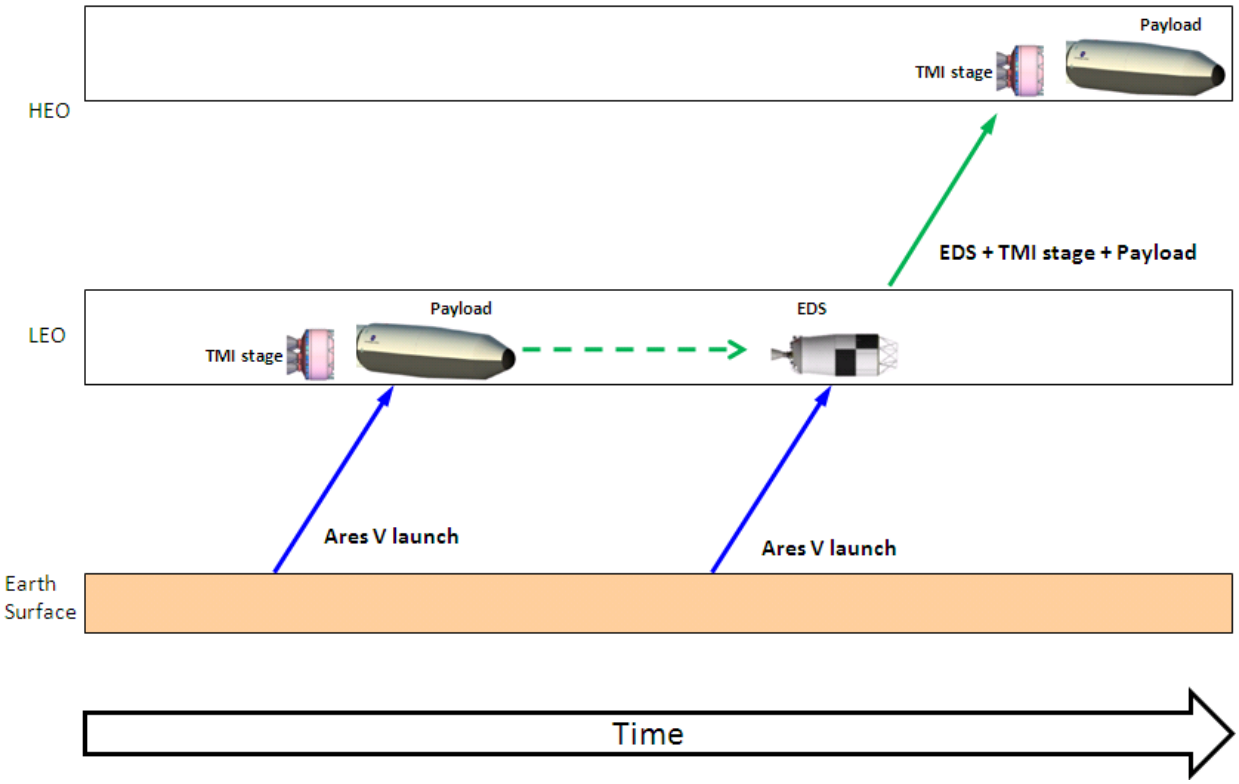

Figure G-2: Graphical representation of earth departure strategy Option 1

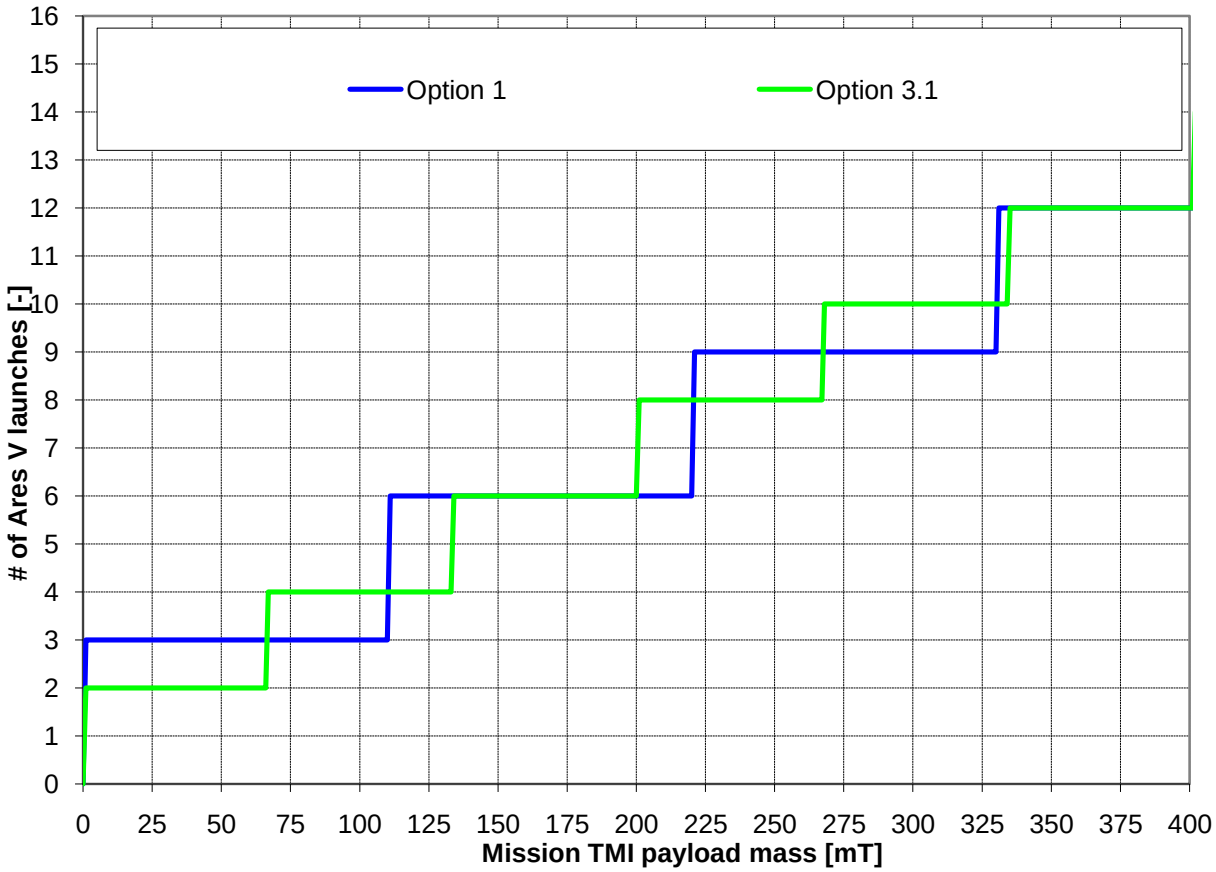

Figure G-3: Number of Ares V launches vs. TMI payload mass for different earth departure strategy options