47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Confere
1 - 4 May 2006, Newport, Rhode Island

AIAA 2006-1717

Development, Structure, and Application of MAST: A Generic Mission Architecture Sizing Tool

Jarret M. Lafleur[*], Carolina I. Restrepo[*], and Michael J. Grant[*]
NASA Johnson Space Center, Houston, Texas 77058

Over two years ago, NASA began pursuit of a vision calling for human exploration of the Moon, Mars, and beyond. As a result, organizations at NASA and elsewhere have been evaluating the many diverse mission architectures possible for fulfilling these goals. Evaluating these architectures often requires sizing procedures specialized for each mission design and considerable iteration among mass estimation and trajectory engineers. MAST is a tool under development aimed at facilitating quick, top-level mass sizing of various mission architectures and streamlining the trajectory/mass sizing process. Written primarily within a Microsoft Excel/Visual Basic framework, a distinguishing feature of MAST is that it allows unlimited numbers of vehicles and maneuvers to be modeled with a minimum of inputs. MAST also incorporates empirical stage mass relationships, and a MATLAB code segment allows optimization of propellant distribution among multiple stages. To date, MAST has been able to size vehicles for moderately complex architectures, and it has been able to accurately compare mass trends among different mission architectures. Future work includes continued sizing of test cases, improvements in the software's ease of use, and further integration of Visual Basic and MATLAB code segments. It is also intended for MAST to achieve automated integration with the trajectory design process.

Nomenclature

A	=	empirical stage mass multiplier	m_f	=	final vehicle mass
a	=	empirical propellant mass fraction offset	m_i	=	initial vehicle mass
b	=	empirical propellant mass fraction constant	m_{PL}	=	payload mass
c	=	empirical propellant mass fraction constant	m_{stage}	=	stage mass
g_0	=	reference gravitational constant	m_{total}	=	total vehicle mass
GEO	=	Geosynchronous Earth Orbit	$TDRS$	=	Tracking and Data Relay Satellite
i	=	orbit inclination	V_c	=	circular orbit velocity
$IMLEO$	=	Initial Mass in Low Earth Orbit	V^+	=	final instantaneous velocity
I_{sp}	=	engine specific impulse	V^-	=	initial instantaneous velocity
IUS	=	Inertial Upper Stage	V_∞	=	hyperbolic excess velocity
LEO	=	Low Earth Orbit	ΔV	=	orbital velocity change magnitude
LPR	=	Lagrangian Point Rendezvous	ζ	=	stage propellant mass fraction
LSR	=	Lunar Surface Rendezvous	λ	=	stage structural mass fraction

I. Introduction

IN January 2004, President George W. Bush announced a new Vision for Space Exploration directing the National Aeronautics and Space Administration (NASA) to return humans to the Moon between 2015 and 2020 in preparation for human exploration of Mars and other destinations in the solar system. In support of this goal, organizations at NASA Johnson Space Center and elsewhere have undertaken the task of evaluating the many options available for returning to the Moon and sending humans to Mars.

Accompanying the typical choice of chemical propulsion for such missions is the need for mass minimization through extensive trajectory optimization. In this trajectory optimization, the total ΔV necessary to accomplish a mission is often used as the figure of merit. However, especially for missions with many architecture options, vehicle mass (and mission cost) may not be minimized for the trajectory associated with the smallest total ΔV. Thus,

[*]Co-op Student, NASA Johnson Space Center, Houston, TX 77058, Student Member AIAA.

initial vehicle mass in low Earth orbit (IMLEO) is a significantly more meaningful figure of merit. A mission architecture sizing tool (MAST) has been developed to interface with trajectory tools in order to perform automated iteration between mass sizing and trajectory design for any user-defined mission architecture. The development and testing of MAST is the focus of this report. The automated interfacing between MAST and trajectory tools is left as future work.

MAST in its present form is the product of approximately 11 months of work among three undergraduate co-op students in NASA Johnson Space Center's Flight Mechanics and Trajectory Design Branch. In Fall 2003, the bulk of the MAST code was written in Visual Basic within a Microsoft Excel framework. This formed the foundation of the code. Work in Summer 2004 focused on interface and performance improvements on this framework as well as the development of a separate MATLAB component of MAST to enable offline optimization of staging and propellant distribution. In Fall 2004, work focused on the improvement of existing and creation of new graphical user interfaces. Internal coding changes were made to allow greater efficiency and more versatility in mission modeling. Additionally, the MATLAB component of MAST was partially integrated into the main Visual Basic code. Currently, the MAST Visual Basic code contains approximately 5000 lines, with an additional 700 lines of code in the full MATLAB-based optimization segment.

II. Structure of MAST

The most unique aspect of MAST is the way in which the program is logically structured in order to accommodate mission architectures involving unlimited numbers of vehicles. It does so through a hierarchical structure of vehicles and maneuvers, applying the rocket equation successively to each vehicle stage. This tool also makes available specialized maneuvers and optimization procedures, and the Microsoft Excel/Visual Basic environment leaves room for further expansion of the tool to interface with external trajectory tools.

A. Data Storage Structure

A distinguishing feature of MAST is that its internal data storage structure allows unlimited numbers of vehicles and maneuvers to be modeled. This is enabled primarily by the fact that all mission architecture data is stored in a single array within the Visual Basic code, the organization of which is shown schematically in Fig. 1: The single Visual Basic array named "aVehicles" contains any given number of vehicles, each of which has associated information such as vehicle name and mass. Each vehicle also has a sub-array of maneuvers which it performs, and each maneuver in this sub-array contains basic information plus a sub-array of stages. Each element of the stage array contains information (primarily concerning ΔV imparted, specific impulse, and stage propellant mass fraction) pertinent to

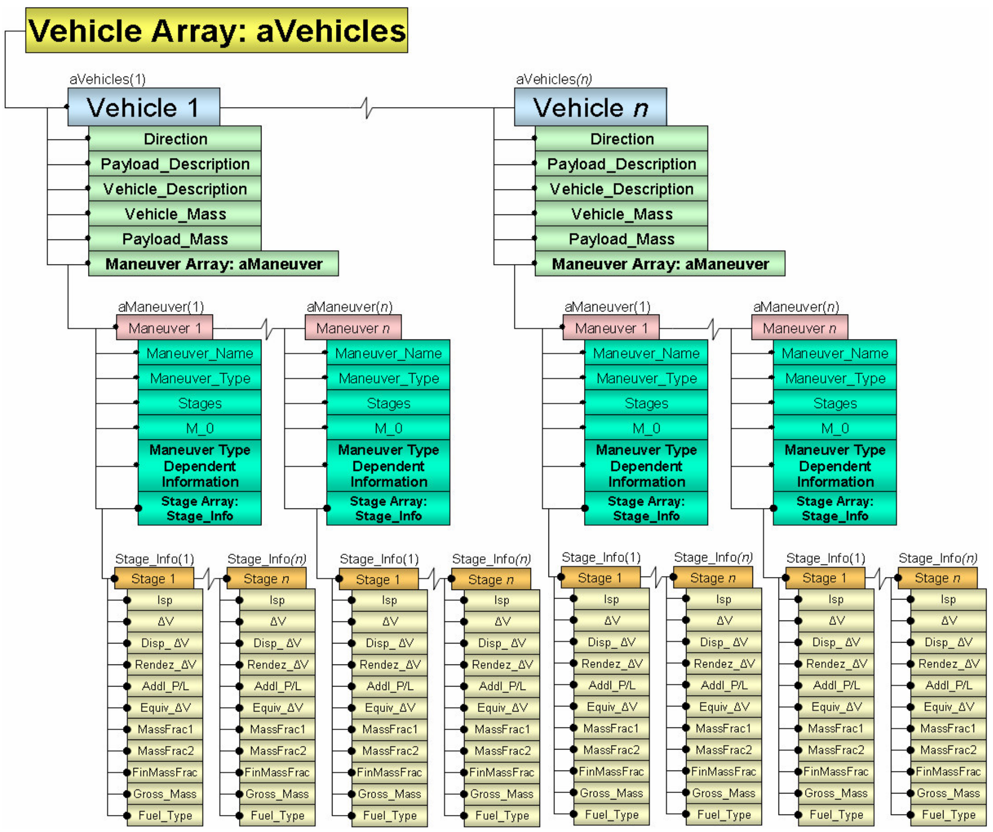

Figure 1. MAST Data Storage Structure.

the performance of the given stage. An additional feature of the structure is that a vehicle's payload can be defined as a previously created vehicle.

This data storage structure proves quite useful for coding and organizational purposes. Further, since the structure exists entirely within the Visual Basic code, the Excel worksheets contain no formulas and are used for display only. At the same time, the displays on the familiar Excel interface do allow the user some ease in comparing and editing different mission architectures.

B. Sizing Method

MAST utilizes the simple rocket equation (shown in Eq. (1)) in order to calculate stage masses in a basic mass sizing process.

$$\Delta V = g_0 I_{sp} \ln\left(\frac{m_i}{m_f}\right) \tag{1}$$

Typical user inputs are payload mass and a sequence of maneuvers (typically requiring inputs of propellant mass fraction, specific impulse, and ΔV imparted for each stage of each maneuver). The MAST code sizes a given vehicle as in Eq. (2) below. Starting with the last stage, total vehicle mass (i.e. gross stage mass plus payload mass) is calculated. This total vehicle mass then becomes the payload mass for the next stage in the stack, and through this iterative procedure the initial system mass is found.

$$m_{total} = m_{PL}\left(\frac{\zeta}{e^{-\frac{\Delta V}{g_0 I_{sp}}} - \lambda}\right) \tag{2}$$

The simplicity of this sizing process is an advantage; however it requires that the user know ΔV, stage mass fractions,[†] and rocket specific impulse in advance.[‡] ΔV is not an issue, since this would presumably come from a trajectory code, and specific impulse can be estimated based on a propellant selection (MAST contains a database of propellant combinations and associated specific impulses to aid in this); however, the optimum ΔV distribution among stages may not be known, just as stage mass fractions may not be readily known. MAST thus incorporates several special features to aid in these and other areas.

C. Special Features

As an aid to the user, MAST incorporates several special features, including the ability to define multiple maneuver types, to utilize historical propellant mass fraction relationships, to add propellant for a stage to perform maneuvers while separated from its main payload, and to optimize the distribution of propellant among stages.

1. Multiple Maneuver Types

In some situations, it is envisioned that it may be easier for a user to input initial and final orbit parameters rather than a direct ΔV for a maneuver. For situations such as these, options exist for the user to select an "Orbit Change Maneuver" or "V_∞ Maneuver" rather than the normal "ΔV Maneuver". The former computes the minimum-energy Hohmann transfer ΔV required to change from an initial to a final specified orbit, and the latter computes the minimum ΔV required to attain a given V_∞ from a specified initial orbit.

Additionally, delta-mass events can be modeled (and are classified as "maneuvers"). A delta-mass event is defined as any mass gain or loss between ΔV maneuvers (i.e. between burns). Thus, one use of this event is to model vehicle dockings and undockings. It should be noted that this event is not necessary for modeling the discarding of stages since stages are assumed to be discarded after their respective maneuvers.

[†] Here, ζ is stage propellant mass fraction (the ratio of stage propellant mass to stage gross mass) and λ is stage structural mass fraction (the ratio of stage structural mass to stage gross mass), so the sum of ζ and λ is one.
[‡] While MAST protects against such errors, the user should be aware that both numerator and denominator on the right side of Eq. (2) must be positive from physical considerations.

2. Auto Mass Fraction Option

For situations in which stage propellant mass fraction data is not readily available, MAST is programmed with a method for estimating this parameter. From work previously performed at Johnson Space Center,[1] a trendline based on historical stage data is programmed into MAST. The trendline is extrapolated from a database that relates gross mass and propellant mass fraction for conventional stages for both cryogenic and Earth-storable propellants. The equation of this line takes the form of Eq. (3), where a, b, c, and A are empirical constants and m_{stage} is stage gross mass:

$$\zeta = a + \log\left(A \cdot m_{stage}\right)\left(b - c \cdot \log\left(A \cdot m_{stage}\right)\right) \tag{3}$$

As expected, higher stage masses allow for higher propellant-to-structure ratios and thus higher propellant mass fractions. Since this trendline does require a stage gross mass in order to determine a mass fraction, the auto mass fraction determination function iterates to converge on the correct mass fraction.

Limitations on the usefulness of this auto mass fraction function are imposed by the historical data from which the associated trendlines were derived: Stage masses significantly above or below the data set may not produce accurate mass fraction results. Further, unconventional stages (particularly those with specialized structural requirements, such as a lunar landing stage) cannot be reasonably modeled using this conventional-stage (i.e. Saturn IB, Titan IV, Centaur, Shuttle Solid Rocket Booster stages, etc.) data.

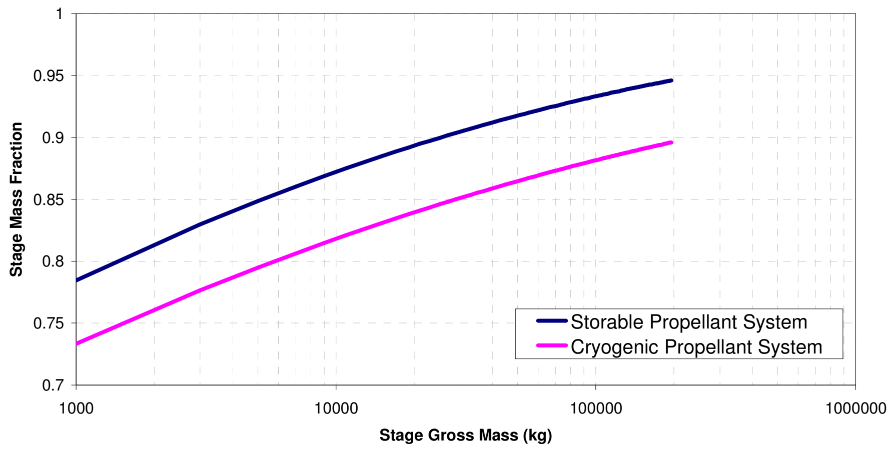

Figure 2. Historical Propellant Mass Fraction Relationship.

3. Disposal and Rendezvous Maneuvers

In the process of testing the MAST code, it was realized that no way existed for a stage to be sized to impart a given ΔV after having been discarded from the stack. Such a burn could be useful, for example, in propelling the stage into an entry or collision trajectory with a planetary body. Thus, the capability was created (by implementing an additional internal iteration loop) to model such stage disposal burns within MAST. The capability was also added for the stage to carry an additional payload mass through its final "disposal" burn (see Fig. 3). This could be useful in a mission architecture, for example, in which a main payload is propelled by a given stage and then a secondary

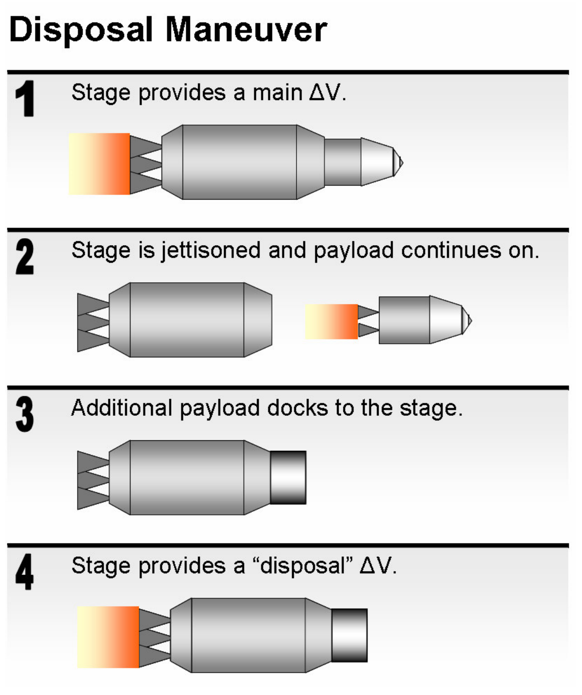
Disposal Maneuver
1
Stage provides a main ΔV.
2
Stage is jettisoned and payload continues on.
3
Additional payload docks to the stage.
4
Stage provides a "disposal" ΔV.

Figure 3. Disposal Maneuver Definition.

payload docks to the discarded stage and is propelled into a disposal or other trajectory. In the case of a "pure" disposal maneuver, this additional payload mass is simply zero.

A logical follow-on to enabling the modeling of a disposal maneuver is to enable the modeling of a rendezvous maneuver. In this procedure, a stage is sized to perform a burn (with or without some additional payload) prior to its docking with the main payload (see Fig. 4). The iteration required is nearly identical to that for the disposal maneuver case.

4. *ΔV Distribution Optimization*

Another issue that arose during the initial testing of MAST was that no way existed to decide how to split ΔV among stages for a multi-stage maneuver. A first step in this direction was made in the development of a standalone MATLAB code to evaluate various ΔV distribution options for any given number of stages and determine the option with the minimum system mass. For situations where stage propellant mass fraction for each stage was unknown, the addition of auto mass fraction functionality showed that the optimum ΔV distribution was one in which larger ΔVs were driven to earlier stages (since these stages had larger masses and could accommodate less structural mass per unit mass of propellant).

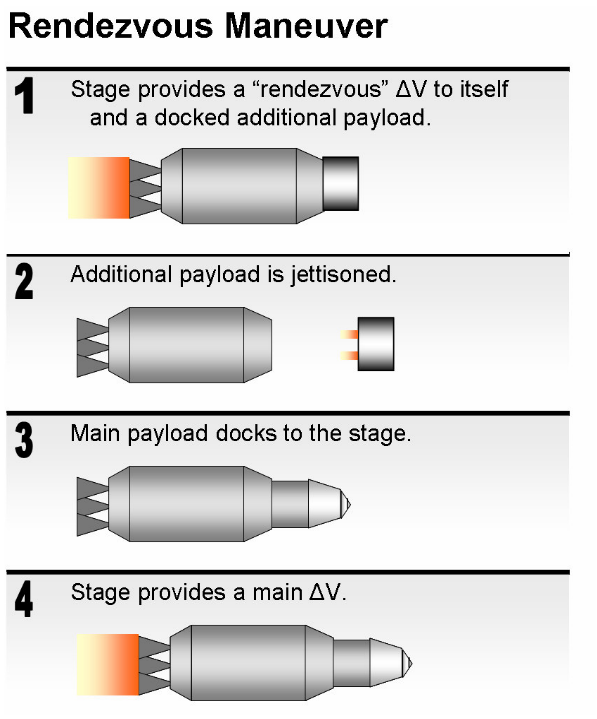
Rendezvous Maneuver
1
Stage provides a "rendezvous" ΔV to itself and a docked additional payload.
2
Additional payload is jettisoned.
3
Main payload docks to the stage.
4
Stage provides a main ΔV.

Figure 4. Rendezvous Maneuver Definition.

Later, a version of this code specialized to optimizing ΔV distribution for the two-stage case was incorporated into the main Microsoft Excel/Visual Basic portion of MAST. During the design of a two-stage ΔV maneuver, the user has the option of inputting the total ΔV required with MAST distributing it optimally between stages. For the user's reference, in each situation the program outputs a graph similar to that in Fig. 5 showing a relation between total vehicle mass and the amount of ΔV in the first stage. Thus, along with knowing the optimum distribution, the sensitivity of vehicle mass due to deviations from the optimum can be seen graphically. At this stage in MAST's development, if the ΔV for a maneuver needs to be divided among more than two stages, the user must use the MATLAB code segment for insight.

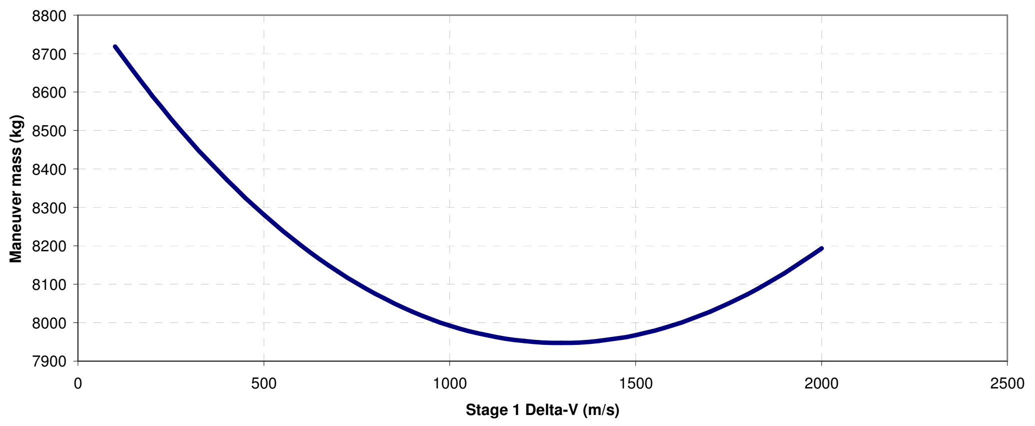

Figure 5. Sample Two-Stage ΔV Optimization Graph.

D. Integration with Trajectory Tool

While computing the masses of each vehicle stage, MAST also computes a mass partial associated with each maneuver. These mass partials relate the sensitivity in IMLEO to a change in ΔV for each maneuver. The mass partial for a given stage is computed by incrementing the maneuver by 1 m/s and calculating the change in IMLEO. For this small perturbation from the design condition, system mass is assumed to exhibit a linear relationship with respect to ΔV. These mass partials could then be used by trajectory tools to redistribute the ΔV within the mission architecture in order to optimize IMLEO. However, since the linear mass partial relationship holds only for small changes in the distribution of ΔV in the mission architecture, iterations must occur between MAST and the trajectory program until the mission architecture converges to an optimal solution. The Flight Mechanics and Trajectory Design Branch at NASA Johnson Space Center (the group in which MAST was developed) utilizes SolSysTr (*Solar System Trajectory*), an in-house trajectory program developed to assist in interplanetary mission design. SolSysTr is largely based on historical interplanetary trajectory tools, INTRPLAN and BEST1WAY and is also in the Microsoft Excel environment.[2] This would allow easy integration of MAST with SolSysTr or other trajectory programs in a setup such as shown in Fig. 6. The integration of MAST with SolSysTr or other trajectory tools has not yet been performed and is left as future work.

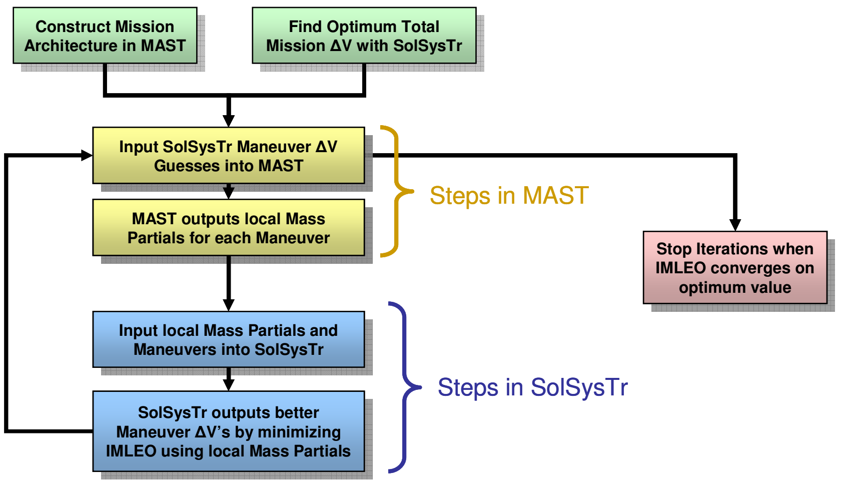
Construct Mission Architecture in MAST
Find Optimum Total Mission ΔV with SolSysTr
Input SolSysTr Maneuver ΔV Guesses into MAST
MAST outputs local Mass Partials for each Maneuver
Steps in MAST
Stop Iterations when IMLEO converges on optimum value
Input local Mass Partials and Maneuvers into SolSysTr
SolSysTr outputs better Maneuver ΔV's by minimizing IMLEO using local Mass Partials
Steps in SolSysTr

Figure 6. Basic MAST/SolSysTr Mass Optimization Flowchart.

III. Application of MAST

To date, MAST has been demonstrated able to size vehicles for moderately complex architectures, and it has been able to accurately compare different mission profiles in terms of total system mass. Below are two examples of MAST capabilities. The first is a detailed application of MAST towards the sizing of the boosters for a Tracking and Data Relay Satellite (TDRS) to demonstrate first-order numerical accuracy. The second demonstrates MAST's ability to compare mass trends to ΔV trends across complex architectures.

A. TDRS Sizing Example

In order to demonstrate the mass sizing capability of MAST, the architecture for insertion of TDRS-D from low Earth orbit (LEO) to geosynchronous Earth orbit (GEO) was sized[3] (see Fig. 7). An inertial upper stage (IUS) was used to transfer TDRS from the payload bay of the Space Shuttle Orbiter *Discovery* on STS-29 at an altitude of 341 km to GEO.[3,4] The mass of TDRS-D was assumed to be 2108 kg based on Ref. 5. The IUS was assumed to have a constant delivered specific impulse (I_{sp}) of 295.5 s.[6] The IUS consists of two stages. The first stage was assumed to place the vehicle into a Hohmann transfer orbit from LEO to GEO with an altitude of approximately 35743 km.[5] Therefore, the second stage would be used to perform the circularization maneuver and plane-change maneuver (to change the orbital inclination from 28.5° due to the initial Orbiter's orbit[4] to 0° to achieve GEO).

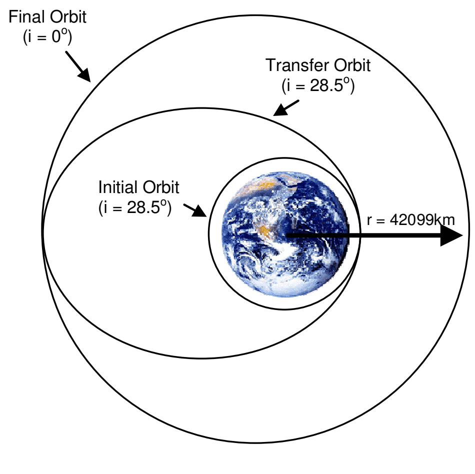

Figure 7. TDRS-D Initial, Final, and Assumed Transfer Orbits.

In order to provide a conservative and ideal mission ΔV estimate with which to size the TDRS-IUS vehicle, the second stage was modeled in two separate ways. In the first scenario, the circularization and plane-change maneuvers were assumed to be performed independently (see Table 1 for individual maneuver ΔVs). In this scenario, the second stage was used to perform the circularization ΔV at the apoapsis of the transfer orbit. Immediately following, a plane-change maneuver (see Fig. 8) was performed to reduce the inclination from 28.5^o to 0^o. The plane-change maneuver to an equatorial orbit may be found in Eq. (4), where V_c represents the circular orbital velocity and i represents the initial orbital inclination.

$$\Delta V = 2V_c \sin\left(\frac{i}{2}\right) \tag{4}$$

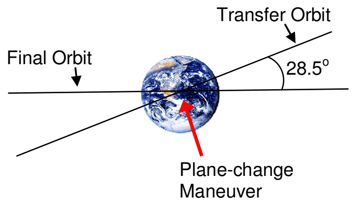

Figure 8. Assumed Plane Change Maneuver.

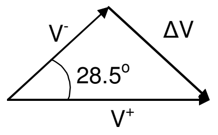

Figure 9. Plane Change Maneuver Vector Diagram.

Table 1. Maneuver Profile for Individual Maneuvers.

Maneuver	ΔV (km/s)	
Transfer Orbit Injection Maneuver	2.419	Stage 1
Circularization Maneuver	1.465	Stage 2
Plane Change Maneuver	1.515	

Table 2. Maneuver Profile for Combined Maneuvers.

Maneuver	ΔV (km/s)	
Transfer Orbit Injection Maneuver	2.419	Stage 1
Combined Maneuver	1.828	Stage 2

The scenario described above provided a conservatively high ΔV estimate of the second stage and, consequently, provided a conservatively high IUS mass.

In the second scenario, the circularization and plane-change maneuvers were assumed to be implemented simultaneously. This scenario is only possible if the apoapsis of the transfer orbit coincided with a node and would provide an ideal ΔV for the second stage. This scenario is more realistic since the solid second stage would not be capable of performing these maneuvers separately. Hence, this would most likely represent the ideal mission profile of the IUS.

Using MAST, the initial mass of the IUS and payload was calculated to be 23,400 kg for the scenario in which the circularization and plane-change maneuvers were performed separately. This is significantly larger than the actual 17,070 kg deployed mass of TDRS and the kick stage (obtained from mission data)[7] and reflects the conservatively high ΔV imparted by the second stage. The mission ΔV profile for the second scenario in which the maneuvers are combined may be seen in Table 2. The corresponding vector diagram may be seen in Fig. 9 in which V^- represents the velocity of the IUS at apoapsis of the transfer ellipse and V^+ represents the final GEO orbital velocity. The combined maneuver may be easily calculated using the law of cosines. MAST was used once again to size the initial mass of the IUS and payload. MAST calculated a new initial mass of 13,960 kg, which is less than the actual deployed TDRS-IUS mass of 17,070 kg. The conservative and ideal cases demonstrate that MAST is capable of bounding the true initial IUS mass. Several sources of uncertainty result in the error of MAST calculating the deployed mass. In both cases the vehicle mass ratio was computed using the auto mass fraction option in MAST. The auto mass fraction option utilizes historical data from a wide range of vehicles and may have resulted in an error in assumed mass fractions for each stage. Also, a minimum ΔV (Hohmann) transfer was assumed to define the transfer orbit. Constraints or other mission specific

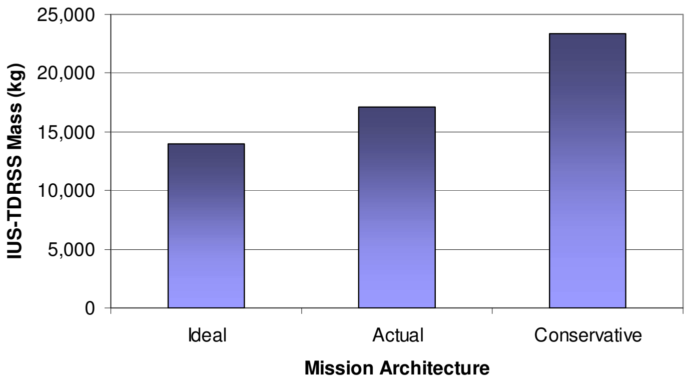

Figure 10. Mass Sizing of TDRS-IUS Architectures

factors may result in a non-Hohmann transfer requiring a higher ΔV. For a graphical representation of the mass sizing for the various scenarios explained above, see Fig. 10. The ideal scenario corresponds to the case in which both the circularization and plane-change maneuvers were combined into one maneuver. The conservative scenario corresponds to the case in which the circularization and plane-change maneuvers were assumed to be performed separately with the circularization maneuver immediately following the plane-change maneuver. The actual scenario corresponds to actual mission data.

B. Mass/ΔV Trends

MAST has the capability to quickly size complex architectures. In this section, various complex lunar mission architectures are compared including lunar orbit rendezvous, Lagrangian point rendezvous (LPR), and lunar surface rendezvous (LSR). In lunar orbit rendezvous (see Fig. 11), the command module and service module remain in lunar orbit while the lander is placed on the lunar surface. In the Lagrangian point rendezvous (see Fig. 12), the command module and service module remain at L1 as the lander is placed on the lunar surface. The architectures of these missions are similar, but are dramatically different from the lunar surface rendezvous architecture. In lunar surface rendezvous (see Fig. 13), the entire vehicle is placed on the lunar surface. MAST was able to quickly size the various complex mission architectures. In addition to the various complex mission architectures, MAST was also able to quickly size various lunar orbit rendezvous missions.

For a lunar orbit rendezvous mission architecture, various landing site latitude missions from two different initial parking orbits were investigated. As can be seen in Fig. 14, landing site latitudes far from the initial parking orbit inclinations about the Moon required higher mission ΔVs. This is largely due to the expensive plane change maneuvers that must be performed. In all of these scenarios, the mission architecture remained constant, and it is clear that IMLEO correlates with ΔV (i.e. IMLEO increases or decreases when ΔV increases or decreases, respectively). However, this scaling does not necessarily hold when comparing different mission architectures. As can be seen in Fig. 14, the Lagrangian orbit rendezvous and lunar surface rendezvous architectures do not have an IMLEO that scales with ΔV. In fact, the lunar surface rendezvous requires the least ΔV but requires the most IMLEO. The rather large IMLEO for the lunar surface rendezvous is due to the large payloads that each stage must propel. These large payloads clearly dominate IMLEO and, hence, simply minimizing ΔV does not produce the optimal mission design. Hence, MAST provided the capability to gain insight in the sensitivity of IMLEO for various complicated mission architectures. Such a capability is ultimately required for the optimization of IMLEO.

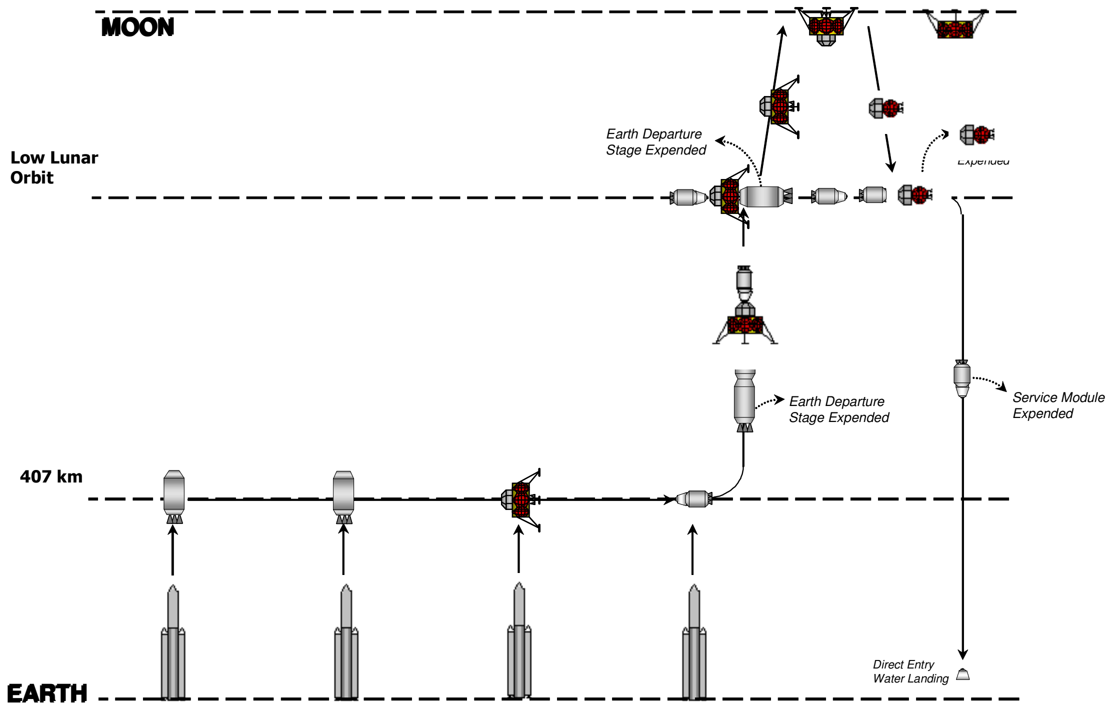
MOON
Low Lunar Orbit
Earth Departure Stage Expended
Earth Departure Stage Expended
Service Module Expended
407 km
Direct Entry Water Landing
EARTH

Figure 11. Lunar Orbit Rendezvous Mission Profile.

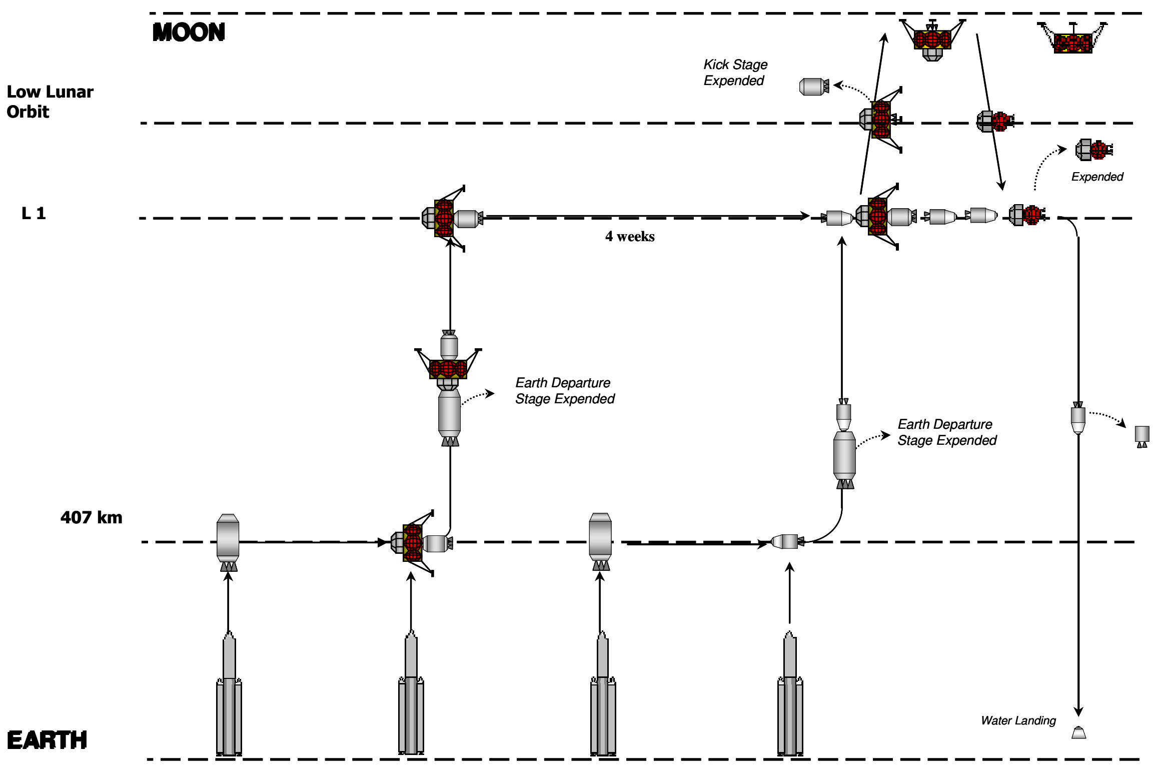

Figure 12. Lagrangian Point Rendezvous Mission Profile.

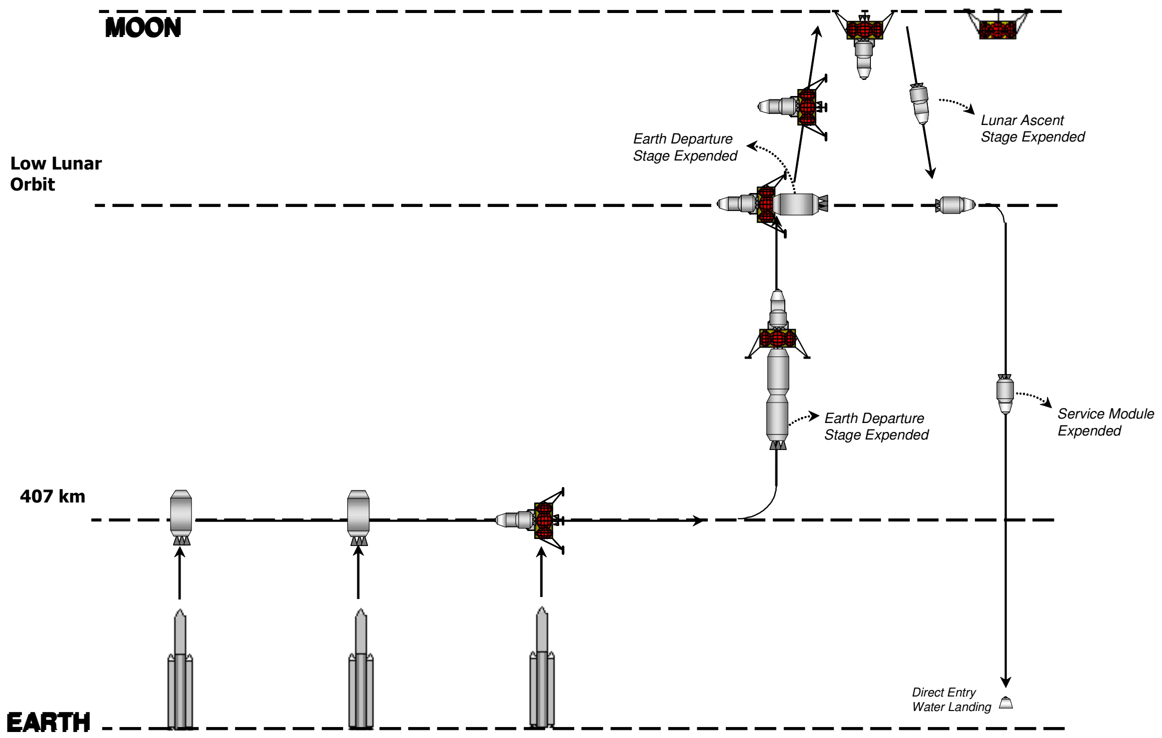
MOON
Earth Departure Stage Expended
Lunar Ascent Stage Expended
Low Lunar Orbit
Earth Departure Stage Expended
Service Module Expended
407 km
Direct Entry Water Landing
EARTH

Figure 13. Lunar Surface Rendezvous Mission Profile.

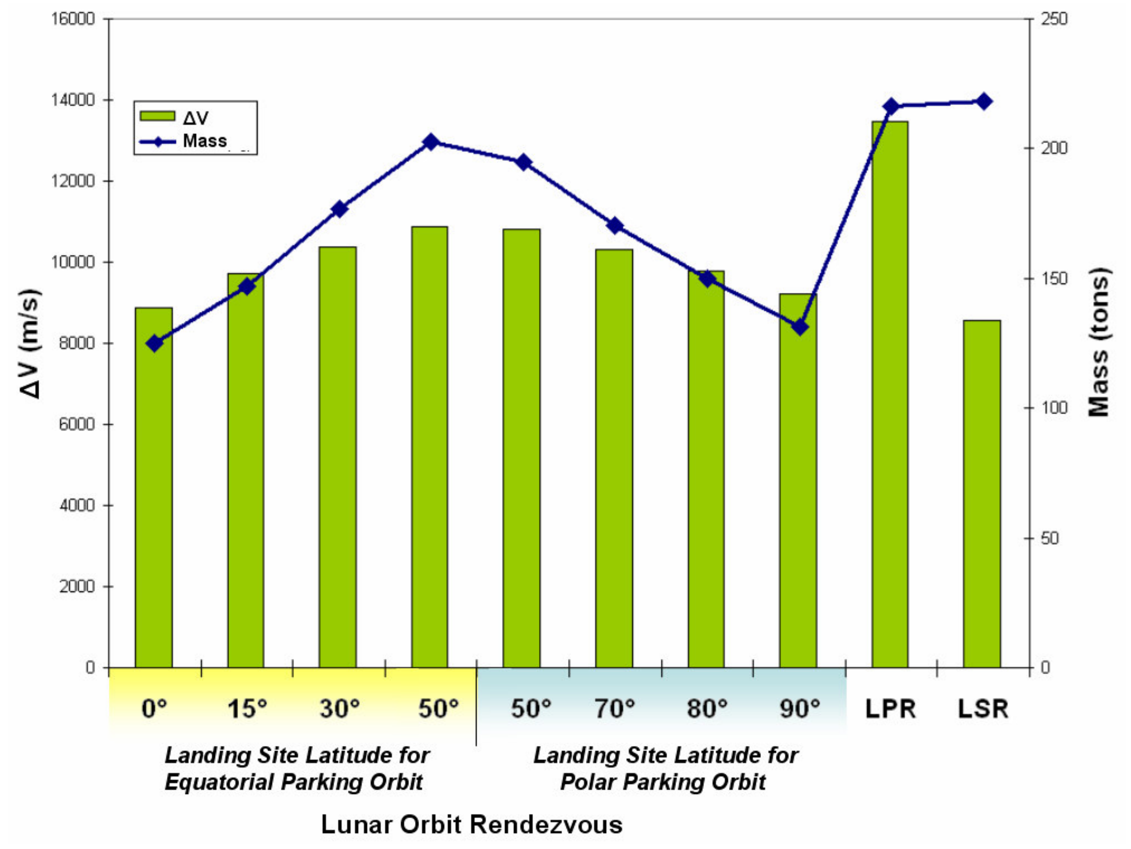

Figure 14. Mass and ΔV Requirements for Selected Lunar Architectures.

IV. Conclusions

To a large extent, MAST has achieved flexibility in modeling mission architectures. The tool is a functional mass sizer capable of modeling unlimited numbers of vehicles and maneuvers. Further, special features allow the modeling of multiple maneuver types (including disposal and rendezvous maneuvers), optimization of ΔV distribution among stages, and automatic selection of stage propellant mass fraction. Current uses of the tool include first-order estimation of system mass requirements and comparison of fundamentally different architectures in terms of vehicle mass rather than total required ΔV.

Future long-term work on MAST includes continued sizing of test cases, improvements in the software's ease of use, and further integration of Visual Basic and MATLAB code segments. It is also desirable to continue expanding MAST to achieve its second goal of integration with the trajectory development process. It is recognized that, for a given mission objective, mission optimization is achieved when the minimum vehicle mass is found. MAST is designed to permit study of overall vehicle mass sensitivities from various mission ΔV events. These sensitivities, or mass partial derivatives information, may then be used by trajectory programs to iteratively optimize mission design.

The combination of a generic and flexible vehicle and maneuver framework, built-in historical stage sizing option, simple Microsoft Excel interface, and an eventual mechanism for integration with trajectory development programs makes MAST unique among sizing tools. MAST holds promise for the mission architecture analysis that will be required as NASA ventures beyond low earth orbit and into the cosmos.

Acknowledgments

The authors would like to thank Jerry Condon for supporting this paper and providing guidance as a mentor. Thanks also to Chris Cerimele whose support for this paper and conference presentation has been essential. Thanks also go to the Flight Mechanics and Trajectory Design Branch, EG5, of NASA Johnson Space Center for providing insight on mission architecture sizing, lessons on orbital mechanics, and discussion of trajectory tools.

References

[1]Heineman, W., “Design Mass Properties II: Mass Estimating and Forecasting for Aerospace Vehicles Based on Historical Data,” NASA JSC-26098, Nov. 1994.

[2]Schroeder, C., and Dawn, T.F., “Interplanetary Mission Design Process (IMDP) Document,” NASA JSC Interoffice Memo, Nov. 1990.

[3]Isakowitz, S. J., *International Reference Guide to Space Launch Systems*, 2nd ed., AIAA, Washington, DC, 1995.

[4]NASA Kennedy Space Center, *1987-1992 Space Shuttle Launch Archives* [online database], URL: http://www.nasa.gov/centers/kennedy/shuttleoperations/archives/1987-1992.html [cited 2 Mar. 2006].

[5]NASA Goddard Space Flight Center, *National Space Science Data Center Spacecraft Query* [online database], URL: http://nssdc.gsfc.nasa.gov/database/sc-query.html [cited 2 Mar. 2006].

[6]Brown, C. D., *Spacecraft Propulsion*, AIAA Education Series, AIAA, Washington, DC, 1996, pp. 164-165.

[7]NASA Johnson Space Center, *Shuttle Operational Data Book* [online database], URL: http://www.spaceflight.nasa.gov/shuttle/reference/green/ [cited 2 Mar. 2006].