AIAA 98-4713
Multidisciplinary Design Optimization Techniques For Branching Trajectories

L. A. Ledsinger
J. R. Olds
Space Systems Design Lab
Georgia Institute of Technology
Atlanta, GA

7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization
September 2-4, 1998 / St. Louis, MO

Multidisciplinary Design Optimization Techniques for Branching Trajectories

Laura A. Ledsinger[†]
Dr. John R. Olds[††]
Space Systems Design Laboratory
School of Aerospace Engineering
Georgia Institute of Technology, Atlanta, GA 30332-0150

ABSTRACT

Fully reusable two-stage-to-orbit vehicle designs that incorporate 'branching' trajectories during their ascent are of current interest in the advanced launch vehicle design community. Unlike expendable vehicle designs, the booster of a reusable system must fly to a designated landing site after staging. Therefore, both the booster return branch and the orbital upper stage branch along with the lower ascent trajectory are of interest after the staging point and must be simultaneously optimized in order to achieve an overall system objective. Current and notable designs in this class include the U. S. Air Force Space Operations Vehicle designs with their 'pop-up' trajectories, the Kelly Astroliner, the Kistler K-1, one of the preliminary designs for NASA's Bantam-X study, and NASA's proposed liquid flyback booster designs (Space Shuttle solid booster upgrade).

The solution to this problem using an industry-standard trajectory optimization code (POST) typically requires at least two separate computer jobs — one for the orbital branch from the ground to orbit and one for the booster branch from the staging point to the landing site. In some cases, three computer jobs may be desired: one from launch to staging, one for the upper stage, and one for the booster flyback. However, these jobs are tightly coupled and their data requirements are interdependent. This paper expounds upon the research necessary to improve the accuracy, computational efficiency, and data consistency with which the branching trajectory problem can be solved. In particular, the proposed methods originate from the field of Multidisciplinary Design Optimization (MDO).

NOMENCLATURE

ADS	Automated Design Synthesis code
CO	Collaborative Optimization
ET	Space Shuttle external tank
FPI	Fixed Point Iteration
I_{sp}	specific impulse (sec.)
J_A	error between Ascent & optimizer variables
J_{PI}	error between POST I & optimizer variables
J_{PII}	error between POST II & optimizer variables
KSC	NASA Kennedy Space Center
LFBB	Liquid Flyback Booster
MDO	Multidisciplinary Design Optimization
NASA	National Aeronautics and Space Admin.
OBD	Optimization-Based Decomposition
$\mathbf{P_s}$	staging point vector
$\mathbf{P_s}'$	staging point vector guess
$\overline{\mathbf{P}}_s$	staging point vector target
POST	Program to Optimize Simulated Trajectories
RBCC	Rocket-Based Combined Cycle
RTLS	Return to Launch Site
SSTO	Single-Stage-to-Orbit
TSTO	Two-Stage-to-Orbit
w_{fb}	flyback fuel weight
w_{fb}'	flyback fuel weight guess
$\overline{w}_{fb}$	flyback fuel weight target
w_{us}	upper stage weight
w_{us}'	upper stage weight guess
$\overline{w}_{us}$	upper stage weight target

[†] - *Ph.D. Student, School of Aerospace Engineering, Student Member AIAA.*

[††] - *Assistant Professor, School of Aerospace Engineering, Senior Member AIAA.*

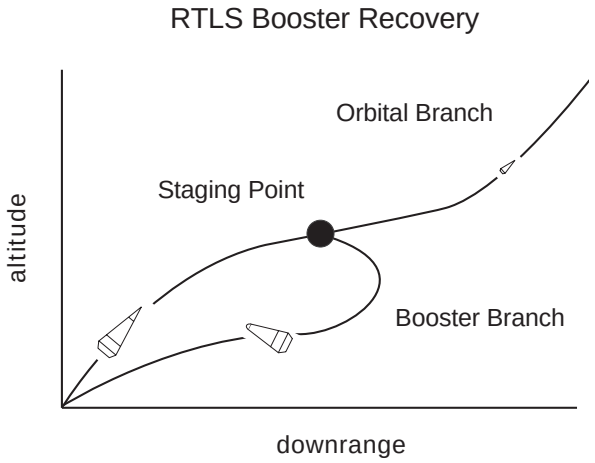

Figure 1: RTLS Branching Trajectory

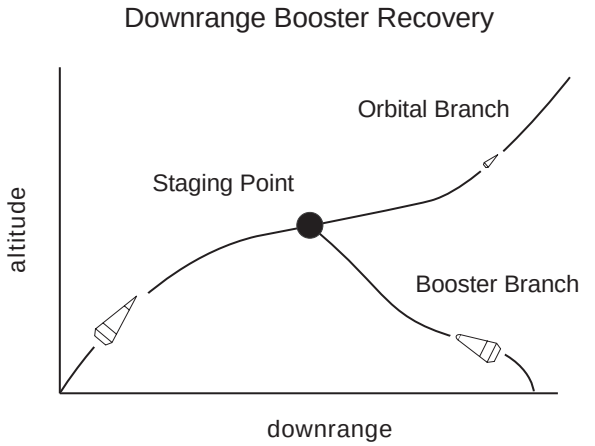

Figure 2: Downrange Branching Trajectory

INTRODUCTION

In an effort to lower costs, designers of advanced two-stage-to-orbit (TSTO) launch vehicles are beginning to consider launch systems in which the booster stage can be recovered, serviced, and reflown. Often the reusable booster is required to land at a predesignated recovery site either near the original launch site (RTLS-style trajectory, Figure 1) or downrange of the staging point (Figure 2). In these cases, the entire trajectory is composed of three parts. The ascent trajectory follows the vehicle from launch to staging. At this point, the trajectory is assumed to split into two 'branches.' One is the *orbital branch* beginning at staging and following the orbital upper stage all the way to orbit. The second branch, or *booster branch*,starts at either the staging point or after a coast phase following staging and follows the reusable booster to its landing site. Due to recovery distance or out-of-plane maneuvers required, the booster is often powered for its flight to the landing site. In simulations where the booster is jettisoned from an orbital vehicle, it may be convenient to combine the ascent trajectory and the orbital branch to create one computer job. The same may be said about a launch vehicle with an upper stage which is jettisoned; the ascent trajectory and the booster branch may be combined.

In general, both the orbital branch and the booster branch rely upon the ascent trajectory for their respective initial conditions. These initial conditions are vectors composed of geographical position, altitude, velocity, flight path angle, velocity azimuth, and staging weight. The ascent trajectory also depends on both branches. Assuming that the booster is powered, the amount of flyback fuel required by the booster influences the gross lift-off weight of the vehicle and thus the ascent path. The weight of the upper stage (which is dependent upon initial staging conditions) also affects the gross lift-off weight of the vehicle and thus the ascent path. Consequently, all the parts of the entire trajectory are tightly coupled or interdependent.

Traditional Solution Method

Unfortunately, the most common method currently used in industry for optimizing a problem of this type (henceforth the *Traditional Method*) while recognizing the coupling of the ascent trajectory and orbital branch ignores the flyback fuel coupling from the booster branch to the ascent trajectory. The ascent trajectory, orbital branch, and booster branch are treated as separate, but sequential optimization subproblems. A reasonable guess at flyback fuel and associated structure is made to establish an initial booster weight. Then, the ascent is optimized for maximum weight at staging (or some other similar criteria). The ascent trajectory will produce a staging state vector used to start the orbital branch and a vector used to initiate flyback, or booster branch. These vectors include: altitude, velocity, flight path angle, etc. The orbital branch will typically be optimized with respect to maximizing the upper stage burnout weight, while the

booster branch will typically be optimized with respect to minimizing the flyback fuel consumed.

There are a number of deficiencies in the Traditional Method. First, the final solution is not 'internally consistent,' in other words, it is not guaranteed to be converged between the subproblems. What if the required flyback fuel differs significantly from the initial guess used to establish the booster lift-off weight? Alternately for an unpowered glide-back booster, what if the booster fails to reach the designated landing site from the given initial conditions?

At a more fundamental level, however, the Traditional Method solution is inherently flawed. The objective functions of the subproblems are not the same, so therefore they can be in conflict. If the system-level objective is to deliver a certain payload to orbit with a minimum weight booster, then why expect an optimum solution from a method that first maximizes the payload to orbit for the orbital branch, then minimizes the flyback fuel for the booster branch? Could not a compromise in the staging conditions be made such that it reduces the flyback fuel and thus decreases the booster weight? A proper solution to this problem requires simultaneous and coupled treatment of all branches of the trajectory, and the establishment of a single, consistent objective function between them (i.e. a system-level optimization).

Trajectory Optimization with POST

In defense of current practitioners, the industry-standard trajectory optimization code typically used in conceptual design is not capable of simultaneously treating and optimizing all parts of a branching trajectory. The Program to Optimize Simulated Trajectories — POST (Ref. 1) is a Lockheed Martin and NASA code that is widely used for trajectory optimization problems in advanced vehicle design. POST is a generalized event-oriented code that numerically integrates the equations of motion of a flight vehicle given definitions of aerodynamic coefficients, propulsion system characteristics, atmosphere tables, and gravitational models. Guidance algorithms used in each phase are user-defined. Numerical optimization is used to satisfy trajectory constraints and minimize a user-defined objective function by changing independent steering and propulsion variables along the flight path. POST runs in a batch execution mode and depends on an input file (or input deck) to define the initial trajectory, event structure, vehicle parameters, independent variables, constraints, and objective function. Multiple objective functions and simultaneous trajectory branches cannot currently be defined in POST.

The Traditional Method solution (discussed above) used for the branching problem relies on at most three separate POST input decks — one for the ascent trajectory subproblem, one for the orbital branch subproblem, and one for the booster branch subproblem. Each subproblem has it's own independent variables, constraints, and objective function. The current research will retain the POST code and the use of at most three separate input decks (one job for each part), but will attempt to eliminate any objective function conflict and lack of data consistency between them.

SAMPLE VEHICLES

To provide applicability to this research, the missions of two candidate TSTO launch vehicle designs were chosen to serve as reference missions. The overall research goal is to investigate various solution approaches for both vehicle designs.

Liquid Flyback Booster

To extend the life of the Space Shuttle and reduce launch costs, NASA is considering replacing the current solid rocket boosters with a reusable liquid booster(s) in a single or dual configuration (Figure 3). After staging, the liquid flyback booster(s) (LFBB) would return to KSC under powered flight. Power for the return flight would be provided by conventional turbofan or turbojet airbreathing engines. LFBB

Figure 3: Liquid Flyback Booster Concept

concepts typically require deployable or fixed wings. Flyback cruise is assumed to be at 10,000 ft. altitude, for this study.

The LFBB configuration and its characteristics are far from final definition. However, when including the orbiter, its trajectory will certainly be a branching problem (like Figure 1). The orbital branch (the Orbiter and the ET) and the booster branch of the ascent trajectory must be treated simultaneously to produce an overall system-level objective. Compromises in the orbital branch might significantly improve the booster branch and vice versa. The complete trajectory profile can be seen in Figure 5.

The reference orbital branch trajectory used in this study contains 12 independent variables (mostly pitch angles, booster throttle settings, and booster throttle bucket initiation time and duration). The orbital branch contains 8 constraints (Space Station transfer orbit targets, maximum dynamic pressure, and lift-off thrust-to-weight ratio). Nominally, the objective of the orbital branch is to maximize the unconsumed ET propellant weight (like maximizing excess payload) for a given set of propulsion characteristics, vehicle aerodynamics, Orbiter inert weight, ET propellant, LFBB ascent propellant, and LFBB inert weight.

The reference LFBB trajectory used in this study uses 13 independent variables (mostly bank angles and angles of attack) and 7 constraints (maximum loads, angle of attack limits, and termination requirements for a return trajectory to KSC). Given a set of jet engine propulsion characteristics, aerodynamics, and a staging point, the booster trajectory nominally tries to minimize flyback fuel weight. The required staging point data from the orbital branch includes time, altitude, flight path angle, latitude, longitude, velocity, velocity azimuth, and booster staging weight (per booster).

Bantam-X Stargazer

The second vehicle to be studied in this research effort is *Stargazer*, a proposed concept for NASA's Bantam-X study (Figure 4). The mission for the Bantam-X study is to deliver a 300 pound payload (of a university explorer class) to a 200 nautical mile circular low-earth orbit at 28.5˚ inclination. Additional programmatics details include a flightrate of 24 flights per year and a recurring cost goal of less than $1.5M per flight. *Stargazer* is the concept that was developed by the Georgia Tech Space Systems Design Lab for the Bantam-X study. It aims to meet these mission and programmatic requirements.

Stargazer is a TSTO vehicle with a reusable RBCC booster and an expendable, 'pop-up' upper stage with a Fastrac-derived engine. The four RBCC engines are ejector-scramjet engines with four modes consisting of ejector, ramjet, scramjet, and rocket. The vehicle is fully autonomous and uses advanced technologies and TPS. From its horizontal take-off at KSC, the booster with the upper stage would fly its ascent trajectory until it reaches Mach 15. After a brief coast, the upper stage would then be jettisoned to continue on its flight to a 200 nmi. circular orbit. The booster would then return to KSC under ramjet power at an altitude of approximately 65,000 ft. to an eventual horizontal landing. From the description given above, one can see that the *Stargazer* trajectory (Figure 6) certainly involves branching trajectories, specifically like that of Figure 1.

In the LFBB branching trajectory, the booster itself is jettisoned, in this case the upper stage is jettisoned at Mach 15, the branching point. As a result of this fact and the fact that the ramjet cruise flyback to the launch site is so difficult to model and costly in a time sense to run, the *Stargazer* trajectory is modeled by three different POST decks. These are the ascent trajectory, the upper stage branch, and the booster flyback branch.

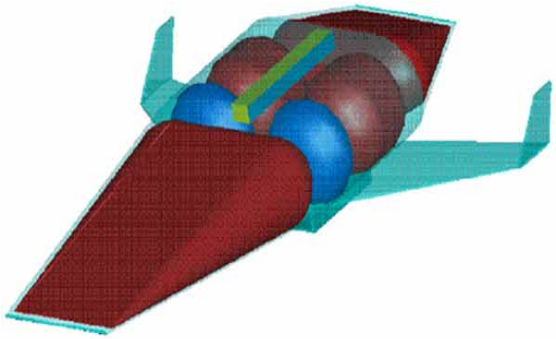

Figure 4: Stargazer *Bantam-X Concept*

The reference ascent trajectory (from launch to staging) contains 11 independent variables that control

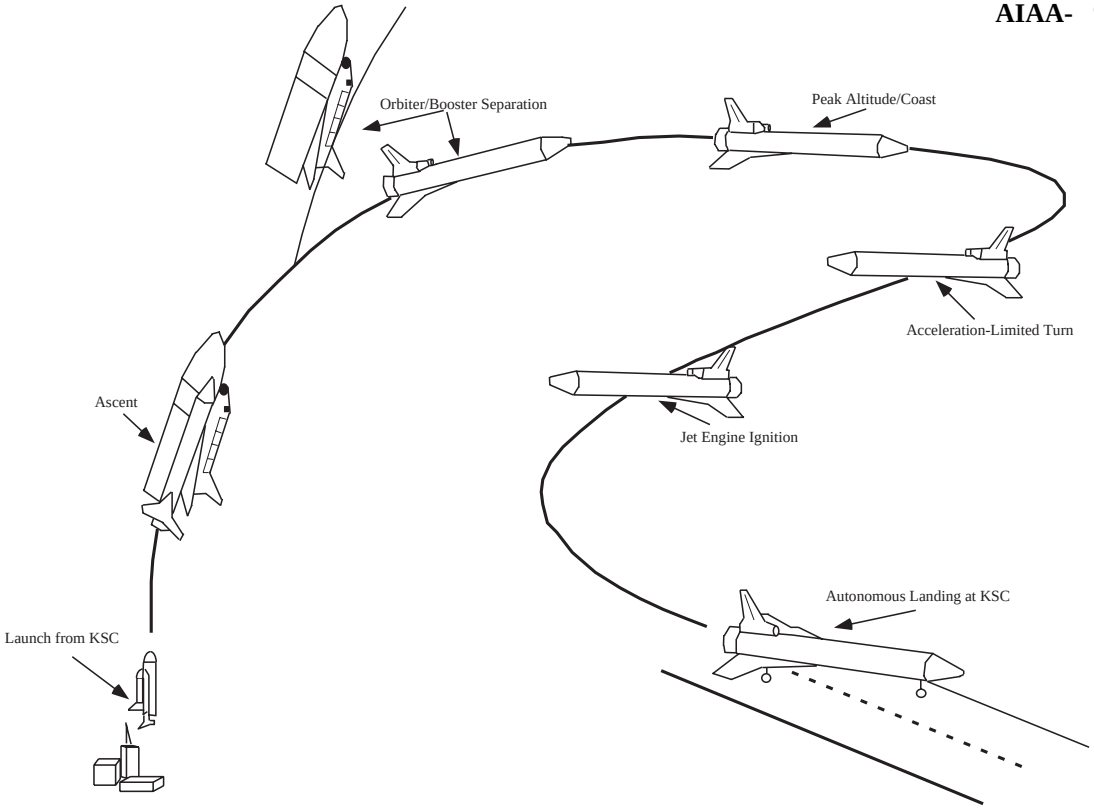

Figure 5: LFBB Mission Profile

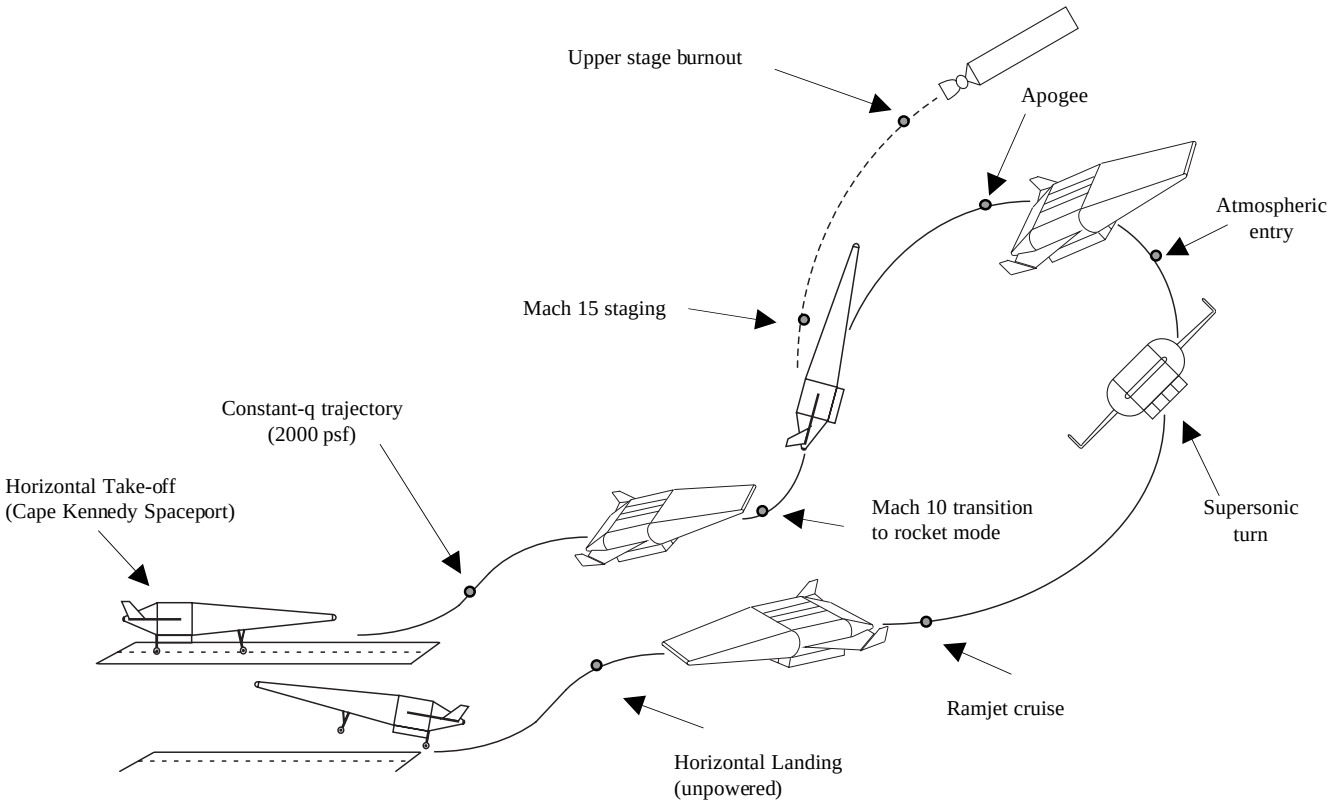

Figure 6: Stargazer *Mission Profile*

relative pitch angles throughout the flight. The trajectory is constrained by dynamic pressure boundaries which provide optimal RBCC performance and by changes in pitch rates which provide smooth ejector and rocket pull-ups. The staging vector at Mach 15 (weight, altitude, longitude, latitude, velocity, flight path angle, and azimuth velocity) must be supplied to the upper stage and flyback branches. The objective of the ascent trajectory is to maximize the weight at staging.

The reference upper stage trajectory (the orbital trajectory branch) is controlled by 7 independent variables, one of which is the time at which the upper stage engine should turn on and the others are inertial pitch angles. The trajectory is constrained by a smooth pull-up at rocket ignition (i.e., a transition from rocket 'off' to rocket 'on' in which the change in pitch angle is limited) and orbital termination criteria. The objective of the upper stage trajectory is to maximize the weight at the end of the trajectory.

The reference flyback trajectory (the booster trajectory branch) is controlled by 16 independent variables. Most of these variables are aerodynamic angles used for the turnaround to KSC, others include the altitude at which the turn begins, the heading coming out of the turn, and the time at which the ramjet is turned on. The trajectory is constrained by the termination conditions at KSC and the conditions at which the ramjet can be started. The objective of the flyback trajectory is to minimize the weight of the fuel consumed. Along with the flight path characteristics supplied from the ascent portion, data from the ramjet mode of the RBCC engine must also be supplied.

SOLUTION APPROACH

The goal of the current research is to retain the current analysis tool (POST) while producing a solution that results in internally consistent data (the true booster flyback fuel is reflected in the initial gross weight, etc.) and a single system-level objective function (without conflicting objective functions for each subproblem). In addition, the solution should be reasonably fast, robust, and efficient.

Since it consists of two highly coupled subproblems, the branching trajectory problem resembles more common multidisciplinary problems such as the coupling between structures and aerodynamics (even though, in this case the subproblems are actually the same discipline!). For that reason, solution techniques from the field of Multidisciplinary Design Optimization (MDO) can be advantageously applied to its solution. Table I lists the characteristics of the four proposed MDO solution techniques — fixed point iteration (FPI), two variations of optimization-based decomposition (OBD), and collaborative optimization (CO). In addition, an entry labeled 'Manual Iteration' is included for comparison. Manual Iteration is simply the traditional (sequential) method iterated to ensure that the coupling variables are internally consistent between the two branches (i.e. the gross booster weight reflects the flyback fuel and any additional structure required to contain it). The Manual Iteration method is not a preferred solution however, since the conflict between the competing objective functions of

Table I: Proposed Solution Techniques to Branching Problems

Method	Internally Consistent Data	Iteration Between Branches	Conflicting Objective Functions	System-level Optimizer	Branch Execution	Optimizer Strategy
Manual Iteration	Yes	Yes	Yes	No	Sequential	Distributed
Fixed Point Iteration (FPI)	Yes	Yes	No	Yes	Sequential	System Level (large)
Partial OBD	Yes	No	No	Yes	Sequential	System Level (very large)
Full OBD	Yes	No	No	Yes	Parallel	System Level (extremely large)
Collaborative	Yes	No	No	Yes	Parallel	Distributed

the two subproblems is not resolved. A brief discussion of each technique follows in the Analysis section.

Note that there are many ways to optimize the trajectories of both the upper stage and the booster. Should the booster ascent propellant be resized if necessary? Should all inert weights remain fixed? Should the system-level objective be maximum orbital payload or minimum booster weight?

In this research, all of the proposed methods analyzed for the LFBB will have a system-level objective of minimizing booster staging weight (prior to flyback phase) given fixed ascent propellant amounts and base inert weights for both the Orbiter and the LFBB. These values were established for an initial concept at NASA - Marshall Space Flight Center. The LFBB staging weight is rather simply calculated as a base inert weight plus the required flyback propellant plus an additional 38% of the flyback propellant to account for additional tankage, structure, etc. to support the required flyback fuel. The excess ET propellant at the end of the orbital branch is required to be zero or positive. In addition, orbital targets for the orbital stage and landing targets for the booster stage must both be satisfied.

The methods of analysis for the Bantam-X concept vehicle, *Stargazer*, will all have a system-level objective of minimizing the dry weight of the booster. In order to run performance analyses on *Stargazer*, aerodynamics, weight, sizing, and engine information must be available. The models for *Stargazer* weights, aerodynamics, and propulsion were supplied by the Space Systems Design Lab at Georgia Tech. Depending on how accurate the initial guess is for the flyback fuel for the booster, the vehicle will probably need to be resized. This must occur at every iteration step, in addition to running three POST decks.

ANALYSIS

This study will utilize the Program to Optimize Simulated Trajectories (POST) in order to simulate the branching trajectories. In the following figures, the orbital branch will be designated as 'POST I.' The booster branch will be denoted as 'POST II.' The ascent trajectory will be labeled 'Ascent.' Since in the LFBB case the Orbiter itself continues to orbit, the ascent portion and the orbital branch are simulated by one POST deck and, for that vehicle, compose POST I. When the internal optimization capability in POST is enabled, a box with a diagonal line will indicate that POST is being used for trajectory analysis *plus* local optimization. A plain box will indicate that POST is simply being used to integrate along a given trajectory. The analysis in this paper corresponds to the *Stargazer* branching trajectories. Note that the sizing and propulsion aspects are important in running the *Stargazer* POST decks. Figure 7 shows the design structure matrix for *Stargazer*. The disciplines in the dashed box represent those which are integral in resizing the vehicle. This study, however, is meant to focus on the performance discipline which is emphasized in the figure and following analysis.

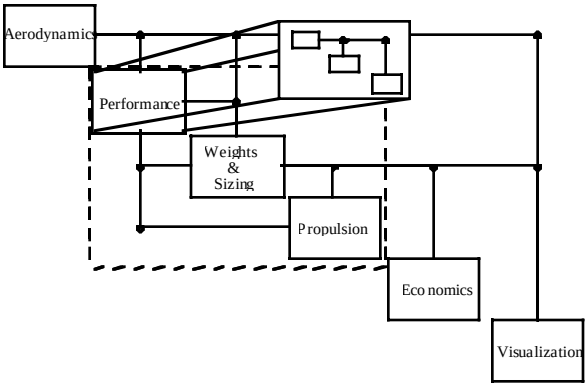

Figure 7: Stargazer *DSM*

The MDO techniques introduced in this paper will improve the results of the Traditional Method (and the Manual Iteration method). These proposed techniques are fixed point iteration (FPI), partial parallel optimization-based decomposition (OBD), full parallel optimization-based decomposition, and collaborative optimization (CO) methods. These methods have been used successfully by others for preliminary aircraft design (Ref. 2) and launch vehicle design (Ref. 3). The FPI method is a serial execution technique which uses an overall system optimizer. This method explicitly uses the coupled feedforward/feedback loops linking the variables of the subproblems. The collaborative and parallel optimization methods are decomposition algorithms in that they break feedback/feedforward loops between the subproblems and incorporate an overall system optimizer. In addition, collaborative

optimization is a multi-level optimization scheme. The respective advantages and disadvantages of these three methods and their use for the branching trajectory problem follows.

The Fixed Point Iteration Method (Figure 8)

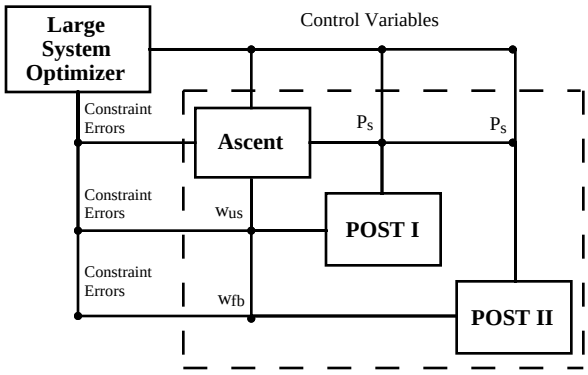

Figure 8: Fixed Point Iteration

In the FPI method, the system optimizer has control of *all* trajectory variables and constraints (Table II). Each trial step from the system optimizer requires a series of iterations between Ascent, POST I , and POST II. Given its initial trajectory variables from the optimizer, Ascent runs in non-optimizing mode (i.e. it simply integrates the equations of motion along the given trajectory and returns the results). The resultant staging conditions ($\mathbf{P}_s$) are then input to POST I and POST II. With these initial conditions as a starting point and their subset of trajectory variables from the system optimizer, POST I runs in non-optimizing mode, followed by POST II. The new upper stage weight, w_{us}, and the new flyback weight, w_{fb}, are fed back to Ascent through the resizing process. The iterations between Ascent, POST I, and POST II continue until the booster dry weight matches the weight which was previously input to Ascent, within a certain tolerance. After the iteration process is completed, the outputs from each POST analysis is fed back to the system optimizer to determine the objective function and system constraints. The control variables are then changed in order to minimize booster dry weight.

The major advantage of the FPI method is that (unlike the one the Manual Iteration Method) it will find the true system optimum without conflicting objectives from the subproblems. It has several disadvantages. The disciplinary experts running POST do not have much say in the optimization process. The system optimizer can become large due to the fact that it controls *all* trajectory variables and constraints. Also, Ascent, POST I, and POST II must execute in series, consuming more real time than if they executed in parallel.

Table II: Size of System Optimizer for Stargazer *Cases*

	Variables	Constraints
Manual Iteration	-	-
FPI	34	13
Partial OBD	36	15
Full OBD	43	22
Collaborative	9	3

The Optimization-Based Decomposition Methods (Figures 9 & 10)

The optimization-based decomposition methods also require a system optimizer with an objective function to minimize booster dry weight. In the partial OBD method, the feedback loops from POST I and POST II to Ascent in the FPI method are broken. Two additional design variables for Ascent are now needed to replace the weights which were originally fed back. These are the guessed upper stage weight, w_{us} ', and the guessed flyback fuel weight, w_{fb} ', which are controlled by the system-level optimizer. Two compatibility constraints added to the system optimizer are used to ensure agreement between the guessed weights and the true weights output from POST I and POST II at the solution. As a result, iteration is no longer required between Ascent, POST I , and POST II to ensure data consistency. This will reduce time between optimizer executions, but may increase iterations on a system level. A diagram of this method can be seen in Figure 9. Note that the resizing event will occur before Ascent is run and during optimizer execution.

In the full OBD method (Figure 10), both the feedback and the feedforward loops that can be seen in the FPI method diagram are broken. In addition to the new design variables and compatibility constraints from the broken *feedback* loops (defined in the previous paragraph), a set of intermediate variables representing the 'expected' staging conditions is created

at the system level ($\mathbf{P_s}$') to be provided directly to POST I and POST II. This effectively breaks the *feedforward* loops as well, and creates a parallel set of subproblems. Compatibility constraints are also added to the system optimizer to ensure that, at the final optimum, the intermediate variables ($\mathbf{P_s}$') match the actual conditions ($\mathbf{P_s}$) produced by Ascent.

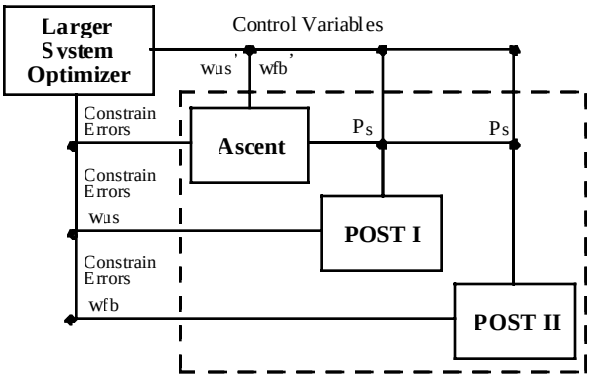

Figure 9: Partial Parallel OBD Algorithm

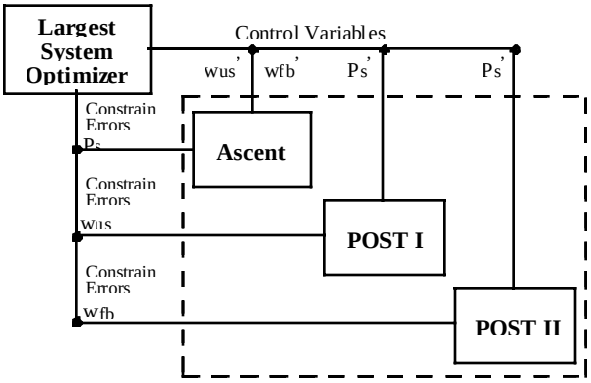

Figure 10: Full Parallel OBD Algorithm

In this method, the optimizer again controls all the local trajectory variables (angles, throttles, etc.) and constraints for each trajectory branch. In addition, the optimizer also controls the new intermediate variables, creating a much larger optimization problem than FPI and even larger than the partial OBD method. The appropriate values are simultaneously input to Ascent, POST I, and POST II which are all run in non-optimizing mode. The results (including w_{us}, w_{fb}, and $\mathbf{P_s}$) are returned from all POST jobs to the system optimizer. The optimizer makes adjustments to the variables to minimize booster dry weight while satisfying all trajectory and compatibility constraints. Note again that the resizing event will occur at the same places as in the partial OBD method.

An advantage of this parallel scheme is that all POST decks can be run simultaneously, reducing the execution time even more. Disadvantages are that the size of the optimizer can become quite large and system experts have little say in terms of optimality in their respective branches.

The Collaborative Optimization Method (Fig. 11)

In the collaborative optimization technique (Ref. 3), a system optimizer is incorporated with an objective function of minimizing booster dry weight. The system optimizer chooses target initial condition vectors and weights ($\mathbf{P_s}$, w_{us}, and w_{fb} targets) which are to be used in Ascent, POST I and POST II which are all run in *optimizing* mode. Each tries to satisfy it's own (local) constraints (ascent, orbital, or landing) with its own trajectory variables while minimizing the error between its local versions of the staging variables and the system optimizer's targets. The sum of the squares of the local errors (J_A, J_{PI}, and J_{PII}) become additional constraints for the system optimizer. The system optimizer changes the new target staging conditions until the errors are zero and the booster dry weight is minimized. In this method, the resizing event will occur during the system optimizer execution. This is a multi-level optimization scheme.

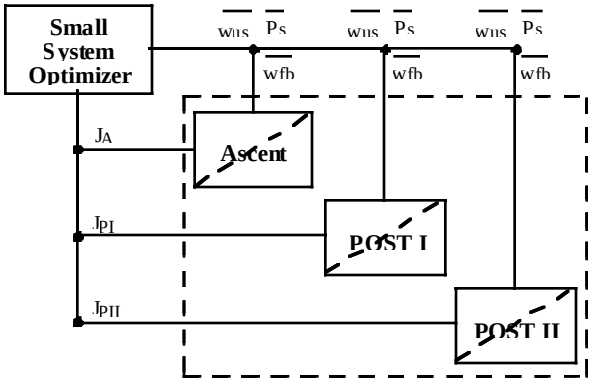

Figure 11: Collaborative Optimization Algorithm

Some advantages of collaborative optimization are that no optimization conflicts occur at the system level and that disciplinary experts can control their own POST programs so that the local objective functions (minimizing the errors) and constraints are

met. The system optimizer is relatively small (containing only the target variables, system-level constraints, and the overall objective function) and all POST decks can run at the same time, conserving time. This method, too, finds the true system-level optimum.

This method has some disadvantages as well. There may be coding or robustness problems given that the POST decks are in optimization mode. This will also contribute to a greater execution time at the local level. Future research will investigate these concerns.

INITIAL RESULTS AND SPECIFICATIONS FOR THE LIQUID FLYBACK BOOSTER

A vehicle which is used extensively in this study is the proposed Space Shuttle upgrade which utilizes a Liquid Flyback Booster. The Space Shuttle Orbiter (with the exception of payload) and the ET weights are not changed as a result of the upgrade. Reference simulations and data were obtained from NASA's Marshall Space Flight Center. The Space Shuttle main engine data is given in Table III.

The data in the next paragraph is that which was used to obtain results for this study. Currently, the ascent engines for the LFBB have yet to be chosen. NASA's Marshall Space Flight Center, along with contractors Lockheed-Martin and Boeing North American, is studying three candidate main engines for the LFBB (Ref. 5). These candidate engines are the Aerojet AJ-800, the Pratt & Whitney RD 180S, and the Rocketdyne RS-76. The first two are existing engines while the latter one is a new design.

The ascent engines for this LFBB study are five Pratt & Whitney RD180's; the corresponding engine data is given in Table IV. The aerodynamic data for the Space Shuttle LFBB upgrade is taken from the all-rocket SSTO in NASA's Access to Space Study (Ref. 4). The booster has a reference wing area of 2,522 square feet. The goal for the Orbiter branch is a target orbit of 43 nautical miles x 153.5 nautical miles x 51.6° inclination, a space station transfer orbit. The inert weight of the Orbiter/ET combination at injection is just under 343,000 pounds.

Table III: Space Shuttle Main Engine Characteristics

Vacuum Isp	455.2 sec
Sea-Level Thrust	375,000 lb
Expansion Area Ratio	77.5:1
Exit Area	44.879 ft^2

Table IV: LFBB Engine Characteristics

Vacuum Isp	338 sec
Sea-Level Thrust	880,400 lb
Exit Area	24.849 ft^2

Traditional Method

The solution for the Traditional Method for the Liquid Flyback Booster is outlined in Figure 12. The result does not account for the coupled nature of the orbital and booster branches. In this case, after, the booster staging weight is approximately 240,000 lbs. This weight does not reflect any addition of flyback fuel or structure.

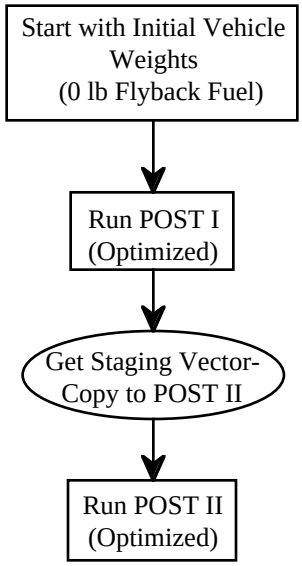

Figure 12: LFBB Traditional Method Flowchart

Manual Iteration Method

This research is currently underway. Initial results have been created for the comparison Manual Iteration method (see Table V & Figure 13). For this case, iteration was used between the two basic subproblems to ensure data consistency (unlike the Traditional Method), however the conflicting objective functions were not addressed. This result will be used as a comparison case in the MDO method assessment currently being conducted.

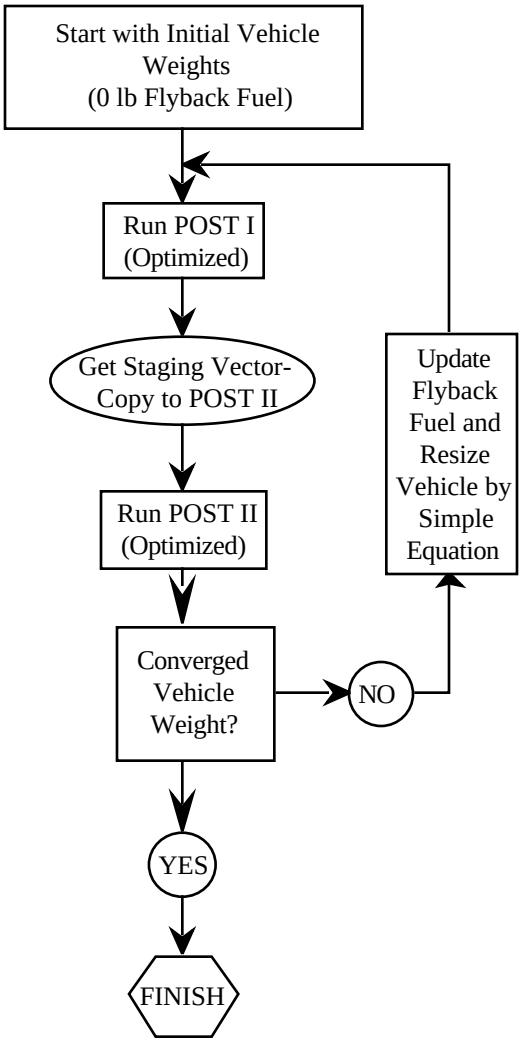

Figure 13: LFBB Manual Iteration Method Flowchart

The Manual Iteration method uses two subproblem optimizers and no system-level optimizer. Execution is sequential and iterative between POST I (launch to orbit) and POST II (staging to return). Booster weight and execution time results are shown in Table V. Note that the LFBB staging weight is intended to be the system-level optimization variable used in subsequent research, but in this case, it is simply an output of the two separate branch optimization processes. Additional trajectory data is shown in Figures 14 -17.

Table V: Manual Iteration Method Results

	Manual Iteration Method
Initial LFBB Staging Weight	237,731.7 lbs
Final LFBB Staging Weight	266,460.9 lbs
Iterations Between POST I & II	12 manual
Total Computational Time	40.15 minutes

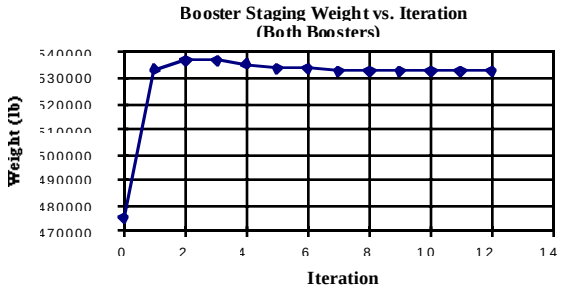

Figure 14: LFBB Staging Weight vs. Iteration

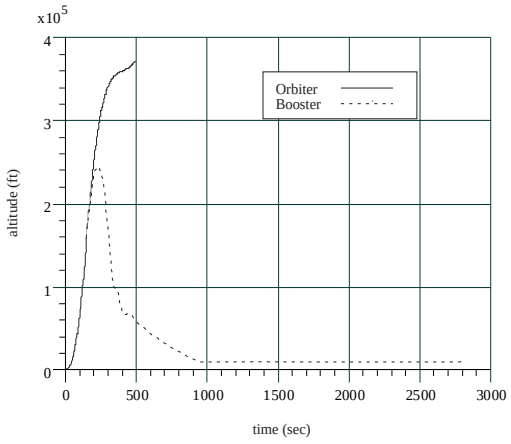

Figure 15: Altitudes vs. Time (Manual Iter.)

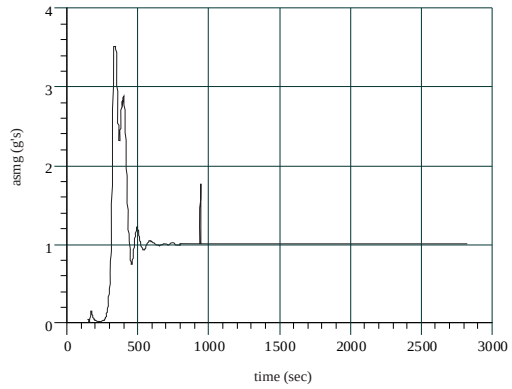

Figure 16: Sensed Acceleration (g's) vs. Time for the LFBB Branch (Manual Iteration)

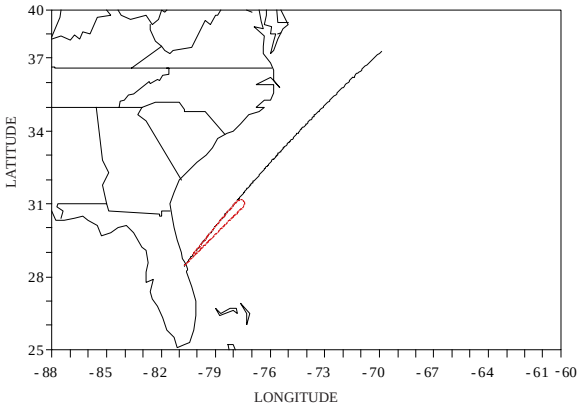

Figure 17: Resultant Ground Track for Both Branches (LFBB--Manual Iteration)

INITIAL RESULTS AND SPECIFICATIONS FOR *STARGAZER*

The other vehicle used in this study is the Bantam-X concept vehicle, *Stargazer*. In the solution approach section, it was mentioned that *Stargazer* must be resized at every iteration of the branching trajectories (Figure 7). Resultant mass ratios and needed mixture ratios for the ascent, upper stage, and the flyback are input to the weights and sizing Excel spreadsheet. The vehicle is then resized. A new thrust required and a new capture area for the RBCC engines are obtained and input SCCREAM (Simulated Combined Cycle Rocket Engine Analysis Module), the RBCC analysis code (Ref. 6). The output from SCCREAM is a POST engine deck. With that engine deck and the new information from the weights spreadsheet, branching trajectory iterations can begin again with POST. Note that the POST aerodynamics deck, which was produced using APAS (Aerodynamic Preliminary Analysis Software, Ref. 7 and Ref. 8), does not need to be recreated because photographic scaling was incorporated into the design.

Traditional Method

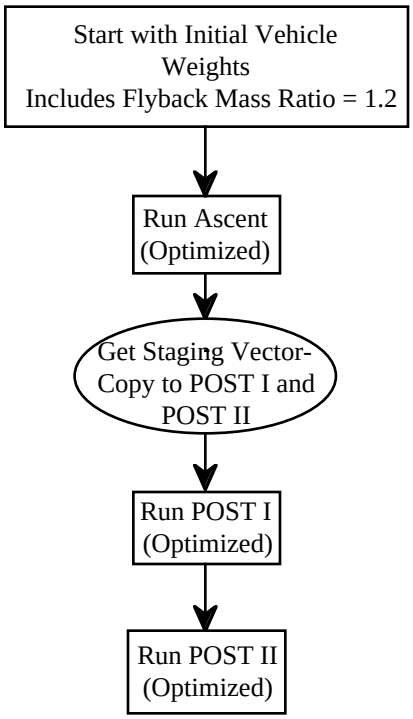

Figure 18: Stargazer *Traditional Method Flowchart*

The solution for the traditional method *Stargazer* is outlined in Figure 18. This result does not account for the coupled nature of the ascent and orbital and booster branches. In this case, the booster dry weight is approximately 35,000 lbs. Note that this does not reflect the required amount of flyback fuel and subsequent structure weight.

Manual Iteration Method

This research is also currently underway. Initial results have been created for the comparison Manual Iteration method (see Table VI & Figure 19). For this case, the Manual Iteration method uses three subproblem optimizers and no system-level optimizer. Execution is sequential and iterative between Ascent,

POST I, and POST II. Booster weight and execution time results are shown in Table VI. Note that the booster dry weight is intended to be the system-level optimization variable used in subsequent research, but in this case, it is simply an output of the two separate branch optimization processes. Additional trajectory data is shown in Figures 20-24.

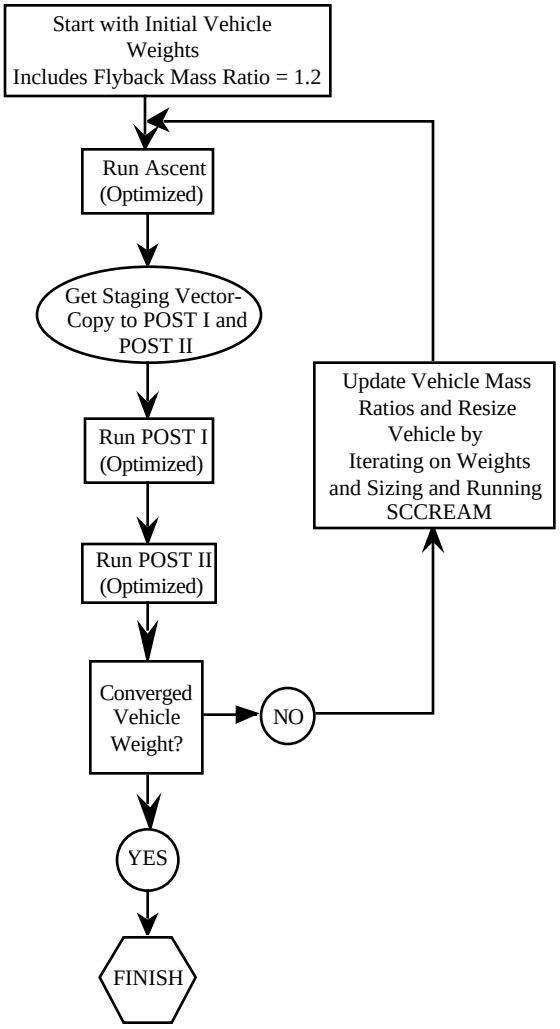

Figure 19: Stargazer *Manual Iteration Method Flowchart*

Table VI: Manual Iteration Method Results

	Manual Iteration Method
Initial *Stargazer* Dry Weight	35387 lbs
Final *Stargazer* Dry Weight	41386.3 lbs
Iterations: Ascent, POST I, & II	5 manual
Total Computational Time	2 hrs, 6 min

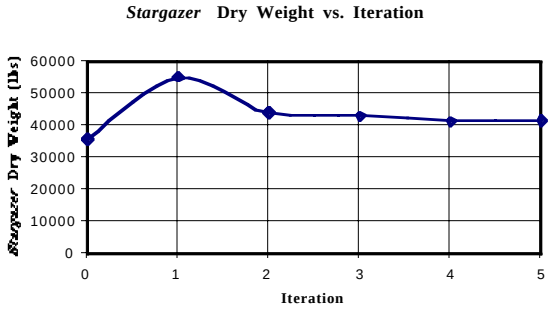

Figure 20: Stargazer *Dry Weight vs. Iteration*

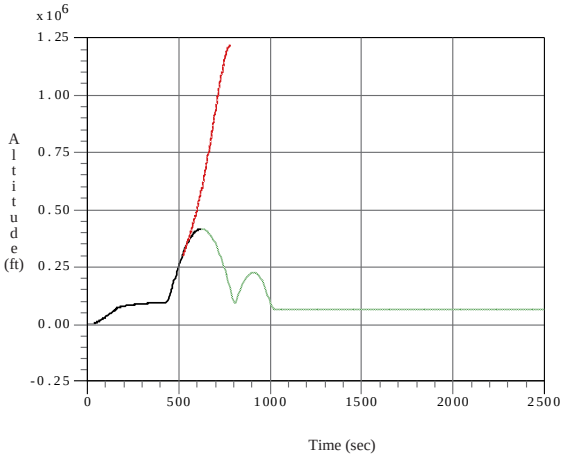

Figure 21: Altitude vs. Time for Stargazer *(Manual Iteration)*

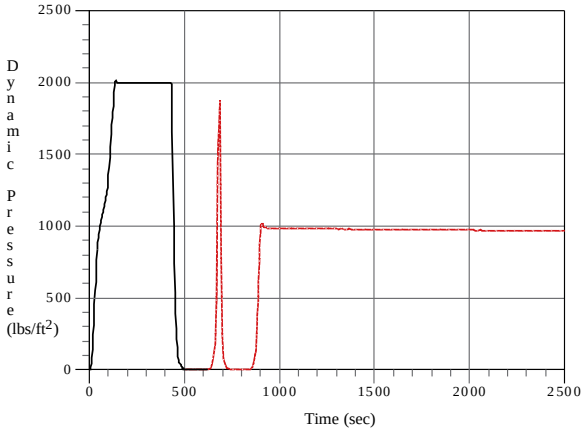

Figure 22: Dynamic Pressure vs. Time for Stargazer*—Ascent and Flyback (Manual Iteration)*

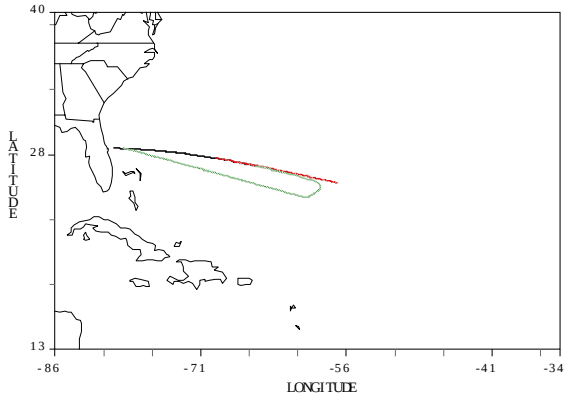

*Figure 23: Resultant Ground Track for All Branches (*Stargazer--*Manual Iteration)*

DISCUSSION

The previous section has shown results for the Traditional and Manual Iteration Methods for the branching trajectories of both the LFBB and *Stargazer*. As expected, the additional flyback fuel/structure weight consideration has increased the staging weight of the LFBB and the dry weight of *Stargazer*. The staging weight of the LFBB increased by 21% of its initial value while the dry weight of *Stargazer* increased by 17% of its initial value. The question remains as to the effect of Multidisciplinary Design Optimization techniques on these branching trajectories. While this question will be answered in the near future, some expectations may be stated at this point.

In general, simulation of the MDO techniques introduced in the Analysis section, when compared to the Manual Iteration Method, will have increased setup times and may have increased iterations and iteration run times due to the complexity added by the system optimizer. However, these disadvantages are outweighed by the benefit that all coupling variables are internally converged (consistent) and the problem is optimized with respect to a single overall objective function. An additional motivating factor for using MDO is the expectation of a smaller weight growth when compared to the percentages stated in the first paragraph of this section.

The use of MDO in this problem will give a new basis for comparison of collaborative and non-collaborative techniques as well. It will show whether it is more accurate and faster to use a large system-level optimizer, as in the FPI, Partial OBD, and Full OBD Methods, or to take advantage of distributed multi-level optimization, as in the CO Method. Use of MDO for this problem will also test the limits of the system optimizer. Will the Full OBD Method create so many design variables (Table II) such that the system optimizer has difficulty finding a valid solution?

The expectations and motivation for using Multidisciplinary Design Optimization techniques listed above validate the use of MDO for this problem. Not only will the use of MDO techniques aid in the solution of the branching trajectory problem, but it will also contribute to the field of MDO research.

SUMMARY

This paper has provided an introduction to trajectory optimization problems having branching trajectories. Two launch vehicles of current interest in this class were identified and discussed — NASA's Liquid Flyback Booster and the Bantam-X concept, *Stargazer*.

The traditional methods of solving branching problems using three POST jobs were shown to be deficient in a number of areas. Notably, they often result in solutions in which the coupling data is internally inconsistent or unconverged. In addition, the objective functions of the two trajectory branches (orbital and booster) are often in conflict.

A set of solution approaches based on MDO techniques was proposed to address these issues. A brief introduction to each of the three proposed techniques was given, along with the advantages and disadvantages of each. The techniques were Fixed Point Iteration with a single system-level optimizer, Optimization-Based Decomposition to eliminate iteration between the branches (two different formulations), and Collaborative Optimization to enable parallel subproblem execution with distributed, coordinated optimizers.

This research is currently underway. Preliminary results for an iterative solution (Manual Iteration) have been created and were presented for both the LFBB and

Stargazer vehicles. While this approach does not solve the problem completely, it does result in internally consistent (converged) coupling variables. The data from the Manual Iteration method will be used comparatively in future research.

FUTURE WORK

Future work for this research will include a full investigation of the MDO methods published in the final product of this research. Results will be compared based on solution accuracy, efficiency, and robustness. Research conclusions and recommended solutions will be published in the final copy of this paper.

Currently, the automated process which runs ADS with the POST decks has been programmed successfully for the LFBB case. The authors are working to make the ADS optimizer solve the branching trajectory problem by changing types of optimizers, scaling factors, convergence factors, and move limits. Future work may include programming a new, problem-specific optimizer, if this current effort is not successful.

ACKNOWLEDGEMENTS

The authors would like to thank the members of the spacecraft research group of the Georgia Institute of Technology Space System Design Laboratory (SSDL) for their support. For their assistance with the POST decks and pertinent data associated with the Liquid Flyback Booster, the authors are grateful to Terri Schmitt and Jim Hays of NASA's Marshall Space Flight Center (MSFC). Thanks also to D.R. Komar of MSFC for his support of the *Stargazer* Bantam-X concept study.

This work was partially sponsored by NASA via the Georgia Space Grant Consortium.

REFERENCES

1. Brauer, G. L., D. E. Cornick, and R. Stevenson, "Capabilities and Applications of the Program to Optimize Simulated Trajectories." NASA CR-2770, Feb. 1977.

2. Kroo, I., S. Altus, R. Braun, P. Gage, and I. Sobieski, "Multidisciplinary Optimization Methods for Aircraft Preliminary Design," AIAA-94-4325-CP, 1994.

3. Braun, R. D., R. W. Powell, R. A. Lepsch, D. O. Stanley, and I. M. Kroo, "Multidisciplinary Optimization Strategies for Launch Vehicle Design," AIAA Paper 94-4341, September, 1994.

4. Access to Space Study, NASA Advanced Technology Team, Volume III: Final Report-Design Databook, July, 1993.

5. Ryan, Richard M., W. J. Rothschild, and D. L Christensen, "Booster Main Engine Selection Criteria for the Liquid Fly-Back Booster," JANNAF JPM, July, 1998.

6. Bradford, J. and J. R. Olds, "Improvements and Enhancements to SCCREAM, A Conceptual RBCC Engine Analysis Tool," AIAA Paper 98-3775, July, 1998.

7. Bonner, E., W. Clever, and K. Dunn, "Aerodynamic Preliminary Analysis Software II: Part I--Theory," NASA CR 182076, Los Angeles, CA, April, 1991.

8. Divan, P., and G. Sova, "Aerodynamic Preliminary Analysis Software II: Part II—User Guide," NASA CR 182077, Los Angeles, CA, April, 1991.