Multidisciplinary Optimization Techniques for Branching Trajectories

J. R. Olds
L. A. Ledsinger
Georgia Institute of Technology
Atlanta, GA

A Companion Paper to
an AIAA Young Members
Off-the-Drawing-Board Presentation

36th Aerospace Sciences Meeting & Exhibit
January 12 -15, 1998 / Reno, NV

Multidisciplinary Optimization Techniques for Branching Trajectories

Dr. John R. Olds†
Laura A. Ledsinger††
Aerospace 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 of the ascent trajectory are of interest after the staging point and must be simultaneously optimized to achieve an overall system objective. Two current and notable designs in this class include the U. S. Air Force's Military Spaceplane designs with their 'pop-up' trajectories and NASA's proposed liquid fly back booster designs (Space Shuttle solid booster upgrade).

The solution to this problem using an industry-standard trajectory optimization code (POST) typically requires 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. However, these two jobs are tightly coupled and their data requirements are interdependent. This paper introduces research to improve the accuracy, computational efficiency, and data consistency with which this twin job problem can be solved. In particular, the proposed methods originate from the field of Multidisciplinary Design Optimization (MDO). The planned research program is outlined, and preliminary results are reported.

---

† - *Assistant Professor, School of Aerospace Engineering, Senior member AIAA.*

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

NOMENCLATURE

ADS	Automated Design Synthesis code
CO	Collaborative Optimization
ET	Space Shuttle external tank
FPI	Fixed Point Iteration
I_{sp}	specific impulse (sec.)
KSC	NASA Kennedy Space Center
LFBB	Liquid Fly Back Booster
MDO	Multidisciplinary Design Optimization
MSP	Military Spaceplane
NASA	National Aeronautics and Space Admin.
OBD	Optimization-Based Decomposition
$\mathbf{P_s}$	staging point variable vector
$\mathbf{P_s}'$	staging point variable vector guess
POST	Program to Optimize Simulated Trajectories
RTLS	Return to Launch Site
SSTO	Single-Stage-to-Orbit
TSTO	Two-Stage-to-Orbit
w_b	booster staging weight
w_b'	booster staging weight guess

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 ascent trajectory is said to have two 'branches'. One is the *orbital branch* beginning at launch with the mated vehicle and following the orbital upper stage all the way the orbit. The second or *booster branch* starts at the staging point and follows the reusable booster to its landing site. Due to

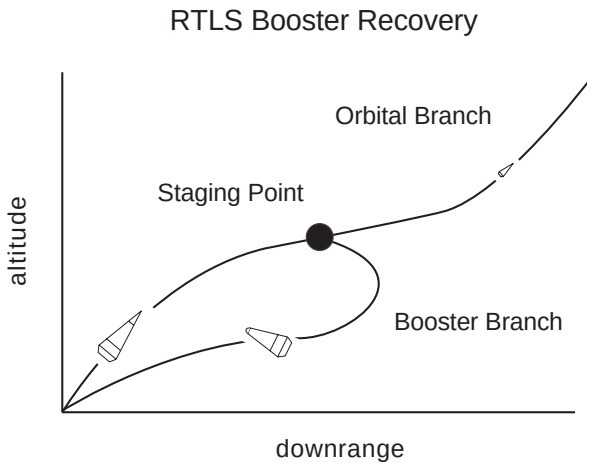

Figure 1: RTLS Branching Trajectory

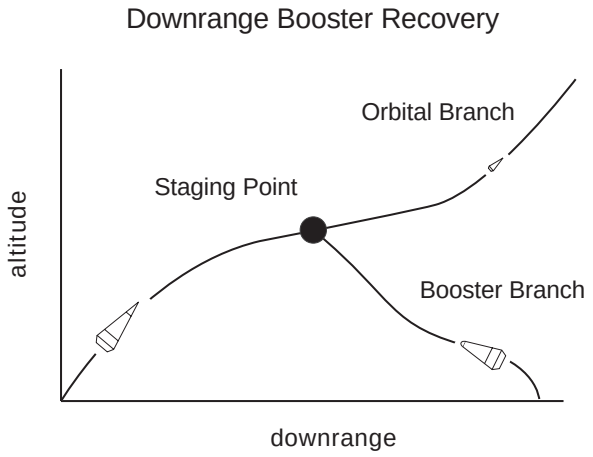

Figure 2: Downrange Branching Trajectory

recovery distance or out-of-plane maneuvers required, the booster is often powered for it's flight to the landing site.

While it is clear that the booster branch depends on the orbital branch for it's initial conditions or state vector at the staging point (geographical position, altitude, velocity, flight path angle, velocity azimuth, and staging weight), it is less obvious that the orbital branch also depends on the booster branch. Assuming that the booster is powered, the amount of fly-back fuel required by the booster influences it's total liftoff weight and therefore affects the orbital branch. Thus the two branches 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*) ignores the fly-back fuel coupling from the booster branch to the orbital branch. The branches are treated as two separate, but sequential optimization subproblems. A reasonable guess at fly-back fuel and associated structure is made to establish an initial booster weight. Then, the orbital branch is optimized for maximum payload (or some other similar criteria). This path will produce a staging state vector (altitude, velocity, flight path angle, etc.) at which the booster runs out of usable ascent propellant and is jettisoned. This staging point is then passed to a second trajectory optimization job and the booster's path is optimized to its recovery site from that fixed staging point. Typically, this second optimization attempts to minimize fly-back fuel.

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

At a more fundamental level however, the Traditional Method solution is inherently flawed. The objective functions of the two 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 should we expect an optimum solution from a method that first maximizes the payload to orbit for the orbital branch, then minimizes the fly-back fuel for the booster branch? Could not a compromise in the payload delivered be made such that the resultant change in the staging condition significantly reduces the fly-back fuel and thus decreases the booster weight? A proper solution to this problem requires simultaneous and coupled treatment of both 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 both 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) to the branching problem relies on two separate POST input decks — 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 two separate input decks (one job for each branch), but will attempt to eliminate any objective function conflict and lack of data consistency between them.

SAMPLE VEHICLES

To provide applicability to this research, 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. The preliminary results reported in this paper, however, pertain only to the liquid fly back booster (LFBB) design.

Military Spaceplane

The U.S. Air Force has recently expressed interest in a small reusable Military Spaceplane (MSP) to extend it's global reach capabilities from continental U.S. bases and provide access to space. Preferred MSP configurations (vertical vs. horizontal take off and landing, motive power, propellant, etc.) are still far from being established, but preliminary operational goals and mission requirements are already being discussed. In particular, the MSP is to be capable of operating as an SSTO launch vehicle with a relatively small payload of about 6 klb. to low earth orbit (ref. 2). Very aggressive turnaround times of less than 8 hours will require fast, simplified ground processing.

To boost delivery payload, a unique 'pop-up' mission has been envisioned (refs. 3 and 4). Coupled with an expendable upper stage, the MSP would serve a booster stage in a TSTO version of the system. The MSP would fly a suborbital trajectory, ejecting the upper stage and payload at some optimum staging point. The MSP booster would be recovered at some downrange landing site, and the upper stage would continue to orbit. In this TSTO configuration, delivered payload is estimated to be as high as 20 klb. to low earth orbit (ref. 2).

The MSP 'pop-up' trajectory is downrange branching trajectory (as in fig. 2). The upper stage branch and the booster branch must be simultaneously optimized to obtain the best system-level objective function. It is therefore of significant interest to the current researchers.

Liquid Fly Back Booster

To extend the life of the Space Shuttle and reduce launch costs, NASA is considering replacing the current solid rocket boosters with a single or twin reusable liquid booster(s) (figure 3). After staging, the

Figure 3: Liquid Fly Back Booster Concept

liquid fly back 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 concepts typically require deployable or fixed wings. Fly-back cruise is assumed to be at 10,000 ft. altitude.

The LFBB replaces only the solid rocket boosters. The orbiter and the external tank (ET) remain relatively unchanged by the substitution. Additional payload delivered is the result of extra performance built into the LFBB(s). In addition, the throttleability of the liquid engines promises to improve ascent abort options.

Like the MSP, the LFBB configuration and it's characteristics are far from final definition. However, it's trajectory will certainly be a branching problem (like fig. 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.

PROPOSED 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 booster fly-back 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 1 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 Method iterated between the two subproblems to ensure that the coupling variables are internally consistent between the branches (i.e. the initial gross booster weight reflects the fly-back 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 the two subproblems is not resolved. A brief discussion of each technique follows.

ANALYSIS

This study will utilize the Program to Optimize Simulated Trajectories (POST) (ref. 1) in order to simulate the branching trajectories. In the following

Table 1: 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

figures, the orbital branch will be designated as 'POST I.' The booster branch will be denoted as 'POST II.' 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. Otherwise a plain box will indicate that POST is simply being used to integrate along a given trajectory to determine end-point conditions.

The MDO techniques introduced in this paper have the potential to improve the results of the Traditional Method (and the Manual Iteration method). These proposed techniques are fixed point iteration (FPI), partial optimization-based decomposition (OBD), full optimization-based decomposition, and collaborative optimization (CO) methods. These methods have been used successfully by others for preliminary aircraft design (ref. 5) and launch vehicle design (ref. 6). The FPI method is a sequential execution technique which uses an overall system optimizer in place of subproblem-level (or *local*) optimizers. This method explicitly uses the feedforward/feedback loops linking the coupling 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 MDO methods and their use for the specific application in this paper follows.

The Fixed Point Iteration Method (Figure 4)

In the FPI method, the system optimizer has control of *all* trajectory variables and constraints (table 2). Each trail step from the system optimizer requires a series of iterations between POST I and II. Given its initial trajectory variables from the optimizer, POST I 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 II. With the $\mathbf{P}_s$ staging conditions as a starting point and it's subset of trajectory variables from the system optimizer, POST II then runs in non-optimizing mode from the staging point to the recovery site. The new booster staging weight, w_b, is calculated from the resultant fly-back fuel (plus a base booster inert weight and the structure required to contain the fly-back fuel) and is fed back into POST I. The iterations between POST I and POST II continue until the new booster staging weight (output from POST II) matches the weight which was previously input to POST I to a certain tolerance. After the iteration process is completed, the outputs from both POST analyses are fed back to the system optimizer to determine the objective function and system constraints. The trajectory variables are then changed in order to minimize booster staging weight and satisfy all of the constraints.

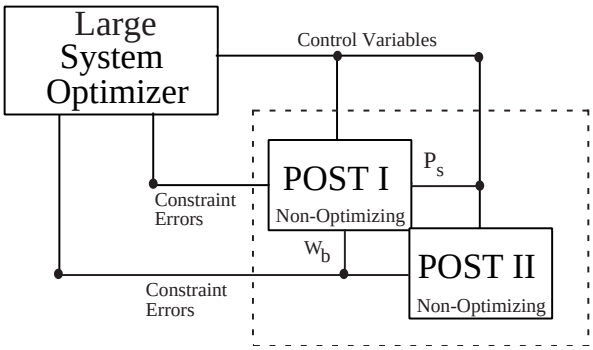

Figure 4: Fixed Point Iteration

Table 2: Size of System Optimizer for LFBB Cases

	Variables	Constraints
Manual Iteration	-	-
FPI	25	15
Partial OBD	26	16
Full OBD	34	24
Collaborative	9	2

The major advantage of the FPI method is that (unlike the Traditional Method) it will find the true system optimum without conflicting objectives from the subproblems. It has several disadvantages. The iteration between POST I and POST II at each trail step from the system-optimizer can be quite time consuming. Iteration can introduce numerical noise into a gradient calculation process. The disciplinary experts running POST don't 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, POST I and POST II must execute in sequence, consuming more real time than if they executed in parallel.

The Optimization-Based Decomposition Methods (Figures 5 & 6)

The optimization-based decomposition methods also require a system optimizer with an objective function to minimize booster staging weight. In the partial OBD method, the feedback loop from POST II to POST I that appears in the FPI method is broken. An additional design variable for POST I is now needed to replace the booster staging weight which was originally fed back. This is the guessed booster staging weight, w_b', which is controlled by the system-level optimizer. A compatibility constraint added to the system optimizer is used to ensure agreement between w_b' and the true weight output from POST II at the solution. As a result, *iteration is no longer required* between POST I and POST II to ensure data consistency. Significant execution time savings relative to FPI are expected. A diagram of this method can be seen in figure 5.

In the full OBD method (figure 6), both feedback and feedforward loops that can be seen in the FPI method diagram are broken. In addition to the new design variable and compatibility constraint from the broken *feedback* loop (defined in the previous paragraph), a set of 8 intermediate variables representing the 'expected' staging conditions is created at the system level ($\mathbf{P_s}$') to be provided directly to POST II for the LFBB cases. This effectively breaks the *feedforward* loop as well, and creates a *parallel* set of subproblems. Eight additional compatibility constraints are also added to the system optimizer to ensure that, at the final optimum, the 8 intermediate variables ($\mathbf{P_s}$') match the actual staging conditions ($\mathbf{P_s}$) produced by POST I.

In this method, the optimizer again controls all the local trajectory variables (angles, throttles, etc.) and constraints for each trajectory branch plus the new intermediate variables, creating a much larger optimization problem than FPI and even larger than the partial OBD method (table 2). The appropriate values are simultaneously input to POST I and POST II. POST I and POST II use these values and run in non-optimizing mode. The results (including w_b and $\mathbf{P_s}$) are returned from both POST jobs to the system optimizer. The optimizer makes adjustments to the

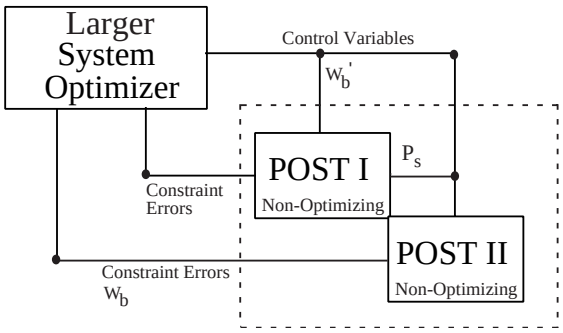

Figure 5: Partial OBD

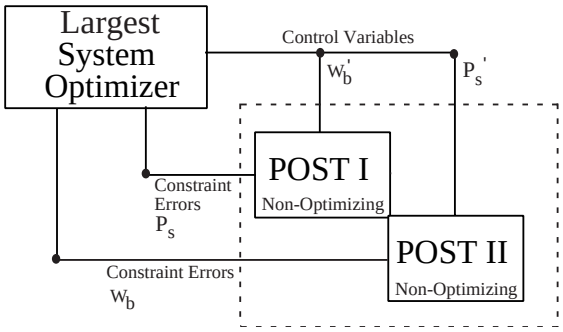

Figure 6: Full OBD

variables to minimize booster staging weight while satisfying all trajectory and compatibility constraints.

An advantage of this parallel, full ODB scheme is that the two 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 running have little say in terms of optimality in their respective branches.

The Collaborative Optimization Method (Fig. 7)

In the collaborative optimization technique (ref. 6), a system optimizer is incorporated with an objective function of minimizing booster staging weight. However, all of the trajectory variables and constraints are kept in their respective subproblems for manipulation. The system optimizer does retain control over all of the coupling variables and chooses a *target* staging point ($\mathbf{P_s}$ target) and a *target* booster staging weight (w_b target) which are to be used in POST I and POST II. POST I and POST II then run in *optimizing* mode. Each tries to satisfy it's own local constraints (orbital or landing) with its own trajectory variables while minimizing the error between its local versions of the staging vector

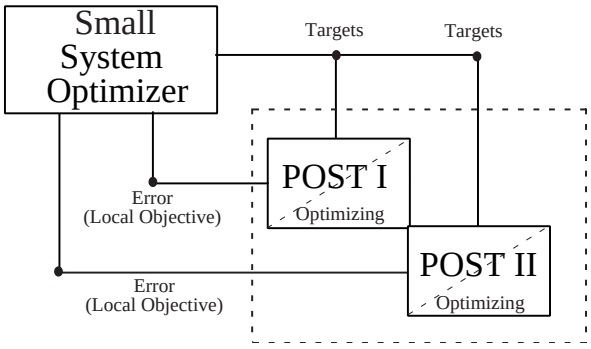

Figure 7: Collaborative Optimization

variables and w_b and the system optimizer's targets. For example, given a target booster staging velocity from the system optimizer, the local booster branch optimizer will try to meet its landing constraints. If this is impossible, it will slowly adjust the staging velocity (and other target variables) until it can meet its constraints. For every target variable that cannot be met, an error is calculated. Minimizing the sum of these local errors becomes an additional constraint for the system optimizer. The system optimizer changes the new target $\mathbf{P}_s$ and w_b until the local errors are zero and the booster staging weight is minimized. This is a multi-level optimization scheme.

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 target variable errors) and constraints are met. The system optimizer is relatively small (containing only the 9 target variables, 2 system-level constraints, and the overall objective function for the LFBB case) and POST I and POST II 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 POST I and POST II are in optimization mode. This will also contribute to a greater execution time at the local level. Previous researchers have also noted unusual convergence patterns for collaborative optimization methods (ref. 6). Future research will investigate these concerns.

INITIAL RESULTS FOR LIQUID FLY BACK BOOSTER

The vehicle which is used extensively in this study is the proposed Space Shuttle upgrade which utilizes a Liquid Fly Back 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 3.

The ascent engines for this particular LFBB configuration will be five Pratt & Whitney LOX/kerosene RD-180's per booster. The corresponding engine data is given in table 4. The aerodynamic data for the Space Shuttle LFBB upgrade is taken from the all-rocket SSTO in NASA's Access to Space Study (ref. 7). The booster has a reference wing area of 2,522 square feet. For the purpose of this study, 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.

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 burnout/staging weight given fixed ascent propellant amounts and base inert weights for both the

Table 3: 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 4: LFBB RD-180 Engine Characteristics

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

Orbiter and the LFBB. These values were obtained for the reference NASA LFBB configuration. The LFBB staging weight is calculated as a base inert weight plus the required fly-back propellant plus an additional 38% of the fly-back propellant to account for additional tankage, structure, etc. to support the required fly-back 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 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, terminate return trajectory at KSC). Given a set of jet engine propulsion characteristics, aerodynamics, and a staging point, the booster trajectory nominally tries to minimize fly-back fuel weight. The required staging point coupling data from the orbital branch includes 8 variables — altitude, flight path angle, latitude, longitude, velocity, velocity azimuth, staging time, and booster staging weight (per booster). At staging, all aerodynamic angles are assumed to be zero (angle-of-attack, bank angle, and sideslip).

Manual Iteration Method Results

This research is currently underway. To date, initial results have only been created for the comparison Manual Iteration method (see table 1). 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. Recall that the Manual Iteration method uses two subproblem optimizers and no system-level optimizer. Execution is sequential and iterative between POST I and POST II. This result will be used as a comparison case in the MDO method assessment currently being conducted.

Booster weight and execution time results are shown in table 5. 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 8 -12.

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 — the Air Force's Military Spaceplane and NASA's Liquid Fly Back Booster.

The traditional methods of solving branching problems using two jobs with POST were shown to

Table 5: Manual Iteration Method Results

	Manual Iteration Method
Base LFBB Inert Weight (**Does not** include fuel and fuel structure.)	237,731.7 lbs
Booster LFBB Staging Weight (**Does** include fuel and fuel structure.)	266,460.9 lbs
Iterations Between POST I & II	12 manual
Total Computational Time	40.15 minutes

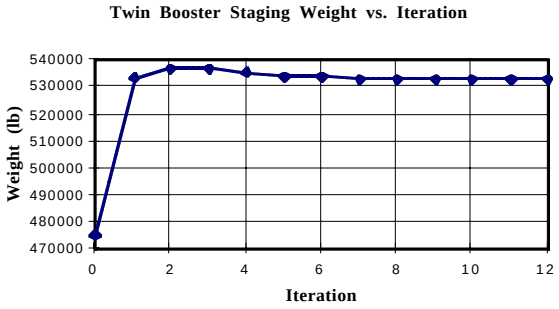

Figure 8: LFBB Staging Weight vs. Iteration

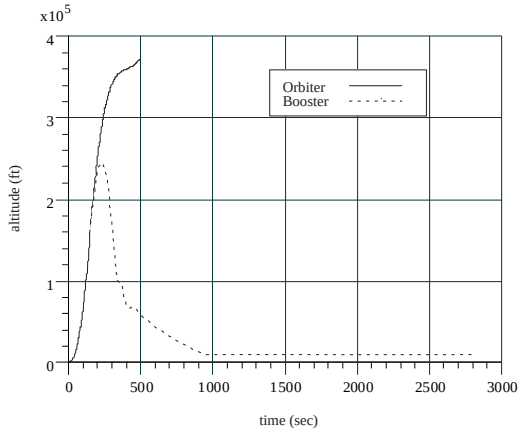

Figure 9: Altitudes vs. Time (Manual Iter.)

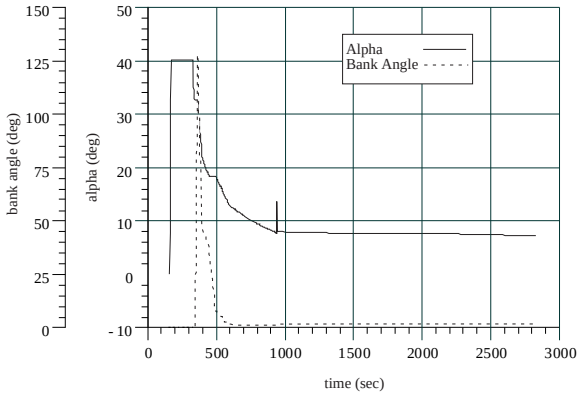

Figure 10: Angle of Attack and Bank Angle vs. Time for the LFBB Branch (Manual Iteration)

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 of each of the three proposed techniques was given, along with 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.

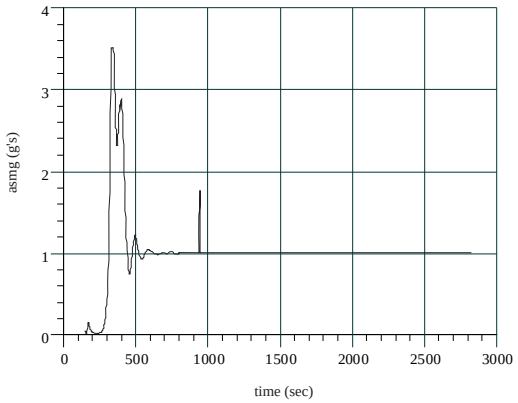

Figure 11: Sensed Acceleration (g's) vs. Time for the LFBB Branch (Manual Iteration)

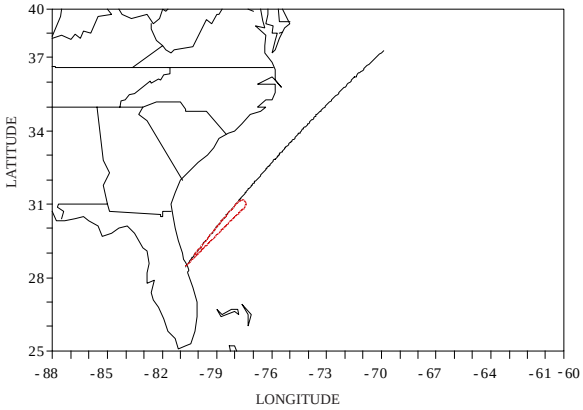

Figure 12: Resultant Ground Track for Both Branches (Manual Iteration)

This research is currently underway. Preliminary results for an iterative solution (Manual Iteration) have been created and were presented. 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 the MDO methods discussed in the Analysis section. Results will be compared based on solution speed, efficiency, and robustness. Research conclusions and recommended solutions will be published in subsequent papers. In addition, all of the same MDO techniques that have been used for the

Liquid Fly Back Booster will be used to solve the pop-up trajectories of a spaceplane concept similar to that of the Air Force's Military Space Plane.

ACKNOWLEDGMENTS

The authors would like to thank the members of the spacecraft research group of the Georgia Institute of Technology Aerospace System Design Laboratory (ASDL) for their support. For their assistance with the POST decks and pertinent data associated with the Liquid Fly Back Booster, the authors are grateful to Terri Schmitt and Jim Hays of NASA's Marshall Space Flight Center. Thanks also to Bob O'Leary at W. J. Schafer Associates for help with preliminary Military Spaceplane concepts.

This work was partially sponsored by NASA via the Georgia Space Grant Consortium under grant number NGT5-40023.

REFERENCES

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

2. Gaubatz, William A, and Zust, Eric L. "The Military Spaceplane Needs and Concept Development." Invited Presentation, AIAA Defense and Space Programs and Technologies Conference and Exhibit, Huntsville, AL, September, 1997.

3. Nielsen, Edward, and O'Leary, Robert. "The Potential Value of Employing a RLV-Based 'Pop-Up' Trajectory Approach for Space Access." In the Proceedings of the 1997 Space Technology and Applications International Forum (STAIF), American Institute of Physics and the University of New Mexico, Albuquerque, NM, January, 1997.

4. Eckmann, James; Cotta, Roy; Matuszak, Leo, and Perkins, David. "Upper Stage Options for Reusable Launch Vehicle 'Pop-up' Missions." In the Proceedings of the 1997 Space Technology and Applications International Forum (STAIF), American Institute of Physics and the University of New Mexico, Albuquerque, NM, January, 1997.

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

6. Braun, R. D., Powell, R. W., Lepsch, R. A., Stanley, D. O., and Kroo, I. M., "Multidisciplinary Optimization Strategies for Launch Vehicle Design," 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary, Analysis, and Optimization. Panama City Beach, FL, September 7-9, 1994.

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