

(Preprint) AAS 15-662

COMBINING SIMULATION TOOLS FOR END-TO-END TRAJECTORY OPTIMIZATION

**Ryan Whitley^{*}, Jeffrey Gutkowski[†], Scott Craig[‡], Tim Dawn[†], Jacob Williams[§],
William B. Stein[‡], Daniel Litton[¶], Rafael Lugo^{||}, and Min Qu^{||}**

Trajectory simulations with advanced optimization algorithms are invaluable tools in the process of designing spacecraft. Due to the need for complex models, simulations are often highly tailored to the needs of the particular program or mission. NASA's Orion and SLS programs are no exception. While independent analyses are valuable to assess individual spacecraft capabilities, a complete end-to-end trajectory from launch to splashdown maximizes potential performance and ensures a continuous solution. In order to obtain end-to-end capability, Orion's in-space tool (Copernicus) was made to interface directly with the SLS's ascent tool (POST2) and a new tool to optimize the full problem by operating both simulations simultaneously was born.

INTRODUCTION

Up to this point, Orion and SLS have maintained separate toolsets to conduct their optimization analyses. Orion primarily focuses on the in-space phase of flight and thus uses an advanced astrodynamics tool known as Copernicus¹ to design maneuvers and conduct trajectory optimization for a problem that often features multiple gravitational bodies. SLS deals primarily with ascent and Earth departure sequences thus requiring both advanced Earth models to predict gravitational forces and atmospheric properties as well as complex propulsion and structural system models. SLS uses the simulation tool POST2 (Program to Optimize Simulated Trajectories)² which contains built in models suited for ascent optimization.

As performance of the combined system has become critical, it is vital to construct a complete end-to-end trajectory from launch to splashdown in order to maximize overall performance and ensure a continuous solution. For instance, if the ascent is optimized to a fixed main engine cutoff (MECO) state independent of that state's effect on in-space cost, vital performance gains that could result from optimizing the MECO state for both the ascent and in-space phases could remain unrealized. Figure 1 displays an example of the independently optimized components of a sub-optimal mission trajectory. Unfortunately, the tools most equipped to do the individual optimization analyses cannot simulate the end-to-end trajectory. Copernicus is not currently equipped to model ascent while POST2 is not suited for modeling complex in-space maneuvers. Instead of embarking on a

^{*} Aerospace Engineer, Exploration Mission Planning Office, NASA JSC, Houston, TX, 77058

[†] Aerospace Engineer, Flight Mechanics and Trajectory Design Branch, NASA JSC, Houston, TX, 77058

[‡] Aerospace Engineer, ERC Inc./Jacobs ESSSA Group, Huntsville, AL, 35812

[§] Aerospace Engineer, ERC Inc. (JSC Engineering, Science, and Technology Contract), Houston, TX, 77058

[¶] Aerospace Engineer, Atmospheric Flight & Systems Branch, NASA LaRC, Hampton, VA, 23681.

^{||} Aerospace Engineer, Analytical Mechanics Associates, 21 Enterprise Parkway, Suite 300, Hampton, VA 23666.

new development program to create a single simulation tool that can handle optimizing both phases of flight in the manner required to obtain the best performance estimates, a path was pursued to allow Copernicus to interface directly with POST2 and become the full problem optimization engine to operate both simulations simultaneously.

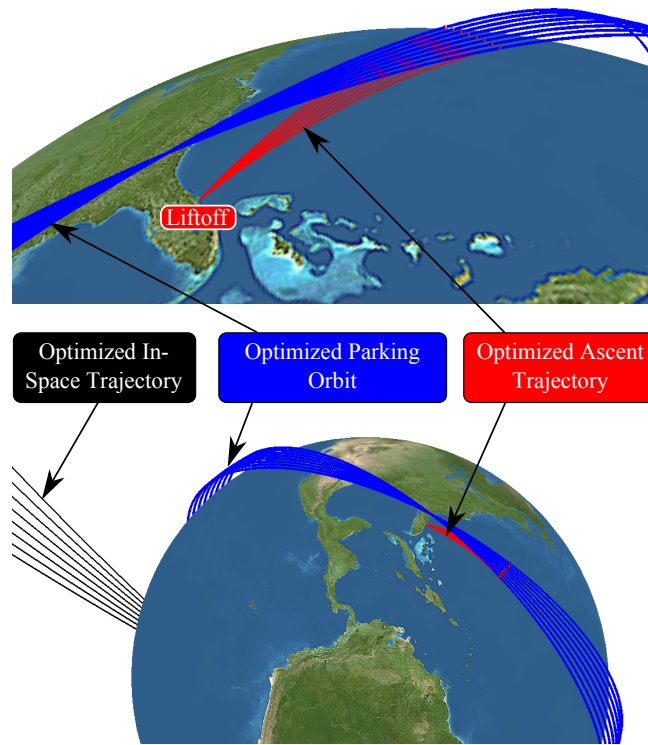


Figure 1: Example of independently optimized trajectory segments

In this paper, the ability for a new approach to end-to-end trajectory optimization will be evaluated. First, a decision matrix of options will be shown that gives an overview of the many ways existing tools could be used as a framework for end-to-end optimization. An overview of Copernicus and POST2 will then provide the context for how these tools can be merged into one overall optimization simulation. One of the options pursued required significant modifications to Copernicus with the addition of a generalized plugin feature. The plugin feature is described in detail in a separate paper.³ The plugin feature allows Copernicus to seamlessly interface not only with POST2 but any other simulation tool that has input and output capability. In addition, the Copernicus plugin enables other approaches to trajectory optimization including the use of curve-fitted datasets and other ascent or entry simulation tools. A second method for merging the simulations that utilizes a third party optimizer known as Isight will then be discussed. Preliminary results showing increased optimality in the production of actual SLS and Orion trajectories for both methods will be given, focusing on the relative performance improvement due to the use of an end-to-end optimizer. While the pros and cons of these two methods will not be compared directly, ample discussion regarding the operation challenges of each approach will provide insight into the different end-to-end optimization implementations including quantitative metrics associated with time spent and relative quality of the results. The performance gain showed in the results along with the challenges of implementing the optimizer will provide a benchmark for future work to improve the end-to-end

tool.

END-TO-END SIMULATION OPTIONS

Simulation tools currently available for trajectory design come in various forms and meet varying analysis needs. There are many commercial available software tools as well as tools built in-house for NASA or privately for use solely by independent aerospace contractors. Some tools feature complex models while some focus on optimization capability with less complex subroutines. With so many tools available to use, it can be a daunting task to choose the best simulators to use to build an end-to-end tool with combined optimization capability. To aid in this process from the onset, the number of tools was reduced to focus on those familiar to the team or which were readily available to NASA. In the end, it was this familiarity that was the driving force behind the choices made for the optimizer. Those were: 1) Copernicus,¹ 2) POST,² 3) OTIS,⁴ 4) SORT,⁵ and 5) MALTO.⁶ There are many ways to combine these sims, so to simplify the selection process the methods were downselected a priori to the 10 combinations shown in Figure 1.




























#	Ascent Sim	Orbit Sim	Primary (ETE) Optimizer	Platform(s)	# of Optimizers	Type of Optimizers	Third Party Code	Primary Coders
1				Linux or Windows	3	E.G. Newton's Method	Matlab	Either MSFC or JSC
2				Linux or Windows	1	SNOPT (other options...)	None	GRC
3				Linux	1 – 3	Several Options	Isight	NESC (w/ MSFC/JSC support)
4				Linux or Windows	2	Copernicus Optimizers	None	JSC (some MSFC)
5				Linux or Windows	1 – 2	Copernicus Optimizers	None	JSC, MSFC, and LaRC
6				Windows	1	Copernicus Optimizers	None	JSC
7			Human in the Loop	Linux or Windows	3	Human in the Loop	None	Individual Analyst
8				Windows	2	Copernicus Optimizers	None	JSC/MSFC
9				Linux	1	NPSOL and others	None	LaRC
10		MALTO	MALTO	Linux	2	Several Options	None	JSC with JPL Support

Table 1: Initial Set of ETE Options Considered

From these 10, a number of metrics were evaluated to determine which option to select including a) Effort to Implement, b) Quality of Solution, c) Ease of Use and d) Analyst Time. While all 10 options were viable, the emphasis on effort to implement narrowed the choices significantly. For example, the ideal end-to-end optimizer would have all phases of flight in the same simulation, but this was not viable without creating either an ascent or in-space sim in another tool. The only tool

that could do both from the list was OTIS and OTIS was not ready to do either the SLS or Orion analyses. Thus, it is no surprise that in the end methods to combine POST2 and Copernicus were selected listed as options 5 and 3 respectively in Figure 1. Option 5 has been constructed by the SLS and Orion teams as part of the overall program core analysis capability, referred in this paper as the *End-to-End Plugin Optimizer (ETE Plugin Optimizer)*. In addition, Option 3 has been adopted by the NASA Engineering & Safety Center (NESC) for use in independent comparative analysis and is referred to as the *ETE Isight Optimizer* in this paper.

COPERNICUS AND POST2 OVERVIEW

To get a sense of the challenges of merging two separate and quite disparate simulation tools, a brief discussion of each base simulation is given below. Copernicus is primarily an in-space trajectory tool and POST2 is primarily an atmospheric ascent and entry tool. A discussion of these tools will demonstrate how the type of problem has shaped the architecture of each simulation and why merging the optimization processes together is challenging.

Copernicus

Copernicus is a generalized spacecraft trajectory design and optimization application.^{1,7} The original prototype was developed at the University of Texas at Austin, and subsequent releases have been developed and maintained at the NASA Johnson Space Center (JSC). The most recent release was version 4.2 (July 2015). Copernicus is actively developed, and has become one of the main software tools at JSC for advanced mission design for future manned missions.^{8,9}

Copernicus is capable of solving a wide range of 3-DOF trajectory design and optimization problems. These include trajectories centered about any planet or moon in the solar system, trajectories influenced by two or more celestial bodies such as libration point trajectories (halo orbits), distant retrograde orbits or other trajectories that exist only in at least a restricted three body model, Earth-Moon and interplanetary transfers, asteroid and comet missions, and more.

A core element of the program is the “segment”, which is the fundamental building block of mission design in Copernicus.¹⁰ Copernicus assumes that all trajectory problems can be modeled using a combination of segments, which can include a set of optimization variables that can be varied, along with an associated set of constraints to be achieved, and a cost function that can be minimized or maximized. Any number of segments can be defined in a mission, and can represent multiple spacecraft or multiple stages of a single spacecraft. In support of the studies described in this report, a recent release also introduced a new “plugin” mission attribute in order to enable the incorporation of external tools and user-created algorithms into the optimization problem.³

Program to Optimize Simulated Trajectories II (POST2)

POST2² is a trajectory optimization program initially developed by Lockheed Martin and maintained at the Langley Research Center. Originally POST2 was two programs, POST3D for 3-DOF (Degree of Freedom) and POST6D for 6-DOF, but the release of a new version of POST, appropriately called POST2, successfully combined the two variants into one tool. POST2 shapes the trajectory through a list of independent variables, the combination of which was known as a u-vector in POST3D. In normal operations, POST2 will optimize a given parameter by changing the independent variables, while meeting one or more constraints. However, for most of the Copernicus

plugin related analysis, the POST2 optimizer is disabled and the simulation is run as a trajectory propagator.

POST2 has seen heavy use in the SLS programs as the primary 3-DOF optimization tool for ascent trajectories. Its strength lies in the fidelity that can be applied to the launch vehicle. At the time of this paper, the SLS Block 1 vehicle is at the CDR level of maturity, which requires fairly detailed modeling of the subsystems. For example, the Core propellant tanks require repressurization, the thrust vector control system requires power, and design constraints may limit the acceleration of the vehicle through certain regimes of flight. All of these constraints require advanced models that exist within the POST2 framework. Without these details, the trajectory may not be feasible and result in inaccurate predictions of performance, launch time, or some other flight critical parameter such as the i-loads for the guidance system. Before the development of the end-to-end tools, POST would be run only to a given ascent target that was defined by the program. The ascent target can now be optimized to get the most performance out of the launch vehicle for a given flight.

END-TO-END PLUGIN OPTIMIZATION METHOD

By selecting the two simulations that the teams were already using in detailed Orion and SLS analyses, there was no need to develop advanced new spacecraft models to directly match the current trajectories and resultant performance. Instead the primary challenge of creating a new optimizer was in allowing the tools to talk to each other and the ability to solve a much more complex optimization problem. For the *ETE Plugin Optimizer* the complexity was minimized by focusing on the two tools and their respective suite of optimizers. The goal is to maximize the types of problems that can be solved by giving the advanced optimization tools within Copernicus direct access to the data coming out of POST2, the ascent simulation. To this end, Copernicus was modified to be able to call any tool directly through the use of an interface with established function calling and a fixed data file format.³ The solution flow is given in Figure 2.

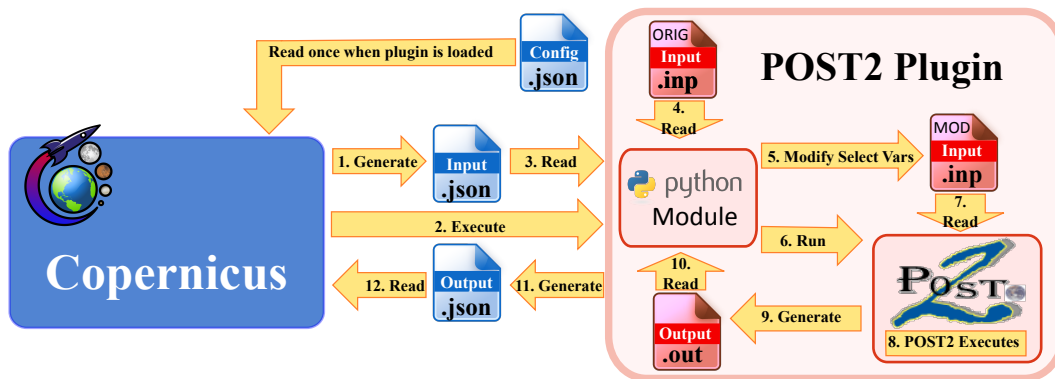


Figure 2: End-to-End Plugin Optimization Method Solution Flow

Essentially Copernicus runs the overall optimization problem by generating data files from which an executable script manages the flow of data to POST2 including its execution and output data repackaging. Copernicus will call the POST2 Plugin as frequently as needed to get the required gradients and trajectory data. As a result, there are two modes currently available to run the tool, 1) Single-level Optimization mode (S-OPT) where POST2 is simply a propagator and 2) Bi-level Optimization (B-OPT) mode where POST2 completes its own optimization cycle and feed backs

data into the subsequent combined problem. Both methods are iterative and have pros and cons as discussed below.

The two main features of the end-to-end simulation are thus the Copernicus plugin and ETE Python scripts. The structure of the Copernicus plugin feature is documented in detail in a companion paper.³ To provide needed context for the operation of the end-to-end simulation, an overview of Copernicus and the relevant plugin capabilities are given below followed by a discussion of the python scripts that established the input/output data connections.

Copernicus Plugin Overview

The Copernicus 4.1 release (March 2015) included a new capability for user-created plugins, which can be used to extend the capabilities of the tool.³ The Copernicus-Plugin interface allows for the transfer of variables in both directions (from Copernicus to the plugin, and from the plugin back to Copernicus). Various types of plugins are allowed. Plugins can be used to enable Copernicus to call an external tool (in this case, POST) as part of an overall optimization problem. For the application described in this study, the plugin is a Python script, and data is exchanged by reading and writing JSON¹¹ data files. Plugins allow Copernicus to be used to handle an overall optimization problem that can include both Copernicus-native segments, as well as external mission phases produced by other means. The emphETE Plugin Optimizer is one example of this capability, with Copernicus calling a plugin (which in turn calls POST). The POST run is a “black box” to Copernicus, which is only concerned with the input and outputs to and from the plugin.

Python Bridge Interface

The *ETE Plugin Optimizer* utilizes a python module which serves as a communication interface between both Copernicus and POST and forms the core of the ETE plugin. This module consists of four separate functions, breaking down the Python Module box in Figure 2:

1. *cop_to_post()*: Loads JSON file provided by Copernicus and writes a new POST input file using a template POST input file and the JSON information.
2. *runPOST()*: Executes POST.
3. *post_to_cop()*: Reads copernicus.out and reformats state data and dependent variables into a JSON format for reading by copernicus.
4. *load_post_data()*: Loads time history data of the POST trajectory for trajectory visualization.

Copernicus loads the python module using the plugin interface and executes it as a script; executing the functions *cop_to_post()*, *runPOST()*, and *post_to_cop()* in order. Copernicus initially writes a JSON file to the working directory, which *cop_to_post()* loads along with a template POST input deck. The function then updates the POST template file text with the new independent and dependent variables provided in the JSON file, writing a new input deck for use during PSOT execution. POST then executes through the function *runPOST()*, resulting in both a copernicus.out file as well as CSV file containing the time histories of given variables throughout the simulation. The python script finally executes *post_to_cop()* in order to parse the copernicus.out file into a JSON format from which copernicus reads in data and continues the simulation.

The plugin allows for POST execution using POST milestone (binary) files as well, which facilitates faster file I/O. The plugin also can execute POST in two different modes, in a single run/propagation mode which allows Copernicus and SNOPT¹² to provide the optimization, or in an targeting/optimization mode which utilizes the native optimizer included in POST. As described previously, the first mode is referred to as Single-level Optimization mode (*S-OPT*) and the second mode as Bi-level Optimization mode (*B-OPT*).

Using this plugin structure for Copernicus also provides future flexibility for integrating other analysis and trajectory simulations using a similar set of functions which interface with Copernicus using the JSON format.

END-TO-END ISIGHT OPTIMIZATION METHOD

Another way to implement an end-to-end optimizer is through a third party tool. To that end, another portion of the team used Isight,¹³ a commercial software package capable of combining multi-disciplinary models and applications in a process flow, in this case trajectories from POST2 and Copernicus. In fact, while the Copernicus based tool in the first implementation has only successfully combined two phases, ascent and on-orbit, the Isight implementation (referred to as *ETE Isight Optimizer* in this paper) has taken it a step further by connecting an entry trajectory also modeled in POST2. In this method, each segment is its own optimization problem, with Isight doing a combined optimization problem on a set of global optimization variables. Figure 3 shows the iterative process flow.

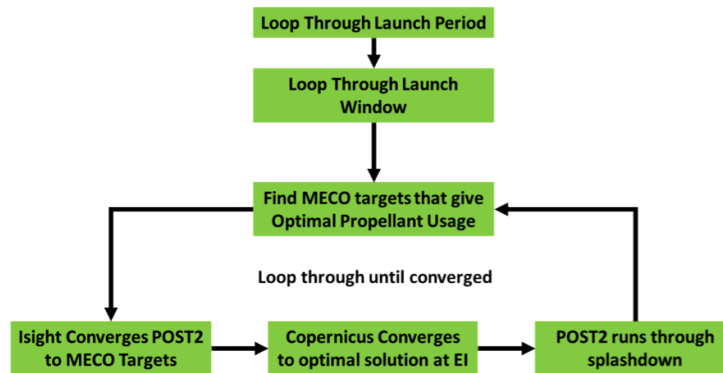


Figure 3: Overview of Isight Architecture

The set of global optimization variables depend on the type of problem being solved, but generally speaking the target state between the ascent and the on-orbit trajectory is included to give the optimizer the most possible flexibility. The primary constraint between on-orbit and entry is the Entry Interface (EI) target line that constrains flight path angle, azimuth and longitude based on a given latitude.

ETE PLUGIN OPTIMIZER RESULTS

As both new ETE optimizers were built to maximize the combined Orion and SLS performance, various planning analyses used to assess the feasibility of upcoming NASA Exploration Missions (EM), in particular EM-1 and EM-2, are serving as the testbed for the different methods. The

data in this paper conveys relative performance increases but is not intended to provide final results representing the official NASA programs.

SLS Block 1 Launch Window Study

The *ETE Plugin Optimizer* was first validated by reproducing a launch window analysis for SLS. By combining ascent and on-orbit trajectories into one optimization problem, the construction of a launch window that optimizes the orbit altitude and inclination of the Earth Parking Orbit (EPO) is enabled. While an actual mission might fix the parking orbit to simplify the guidance targeting is sufficient performance margin was available, the ETE optimized the parking orbit targets at each launch time to show the relative gain if the problem is fully optimized. Figure 4 shows a net gain in propellant margin (propellant above what is required for the mission) when comparing the maximums of a given launch window to the original analysis completed.

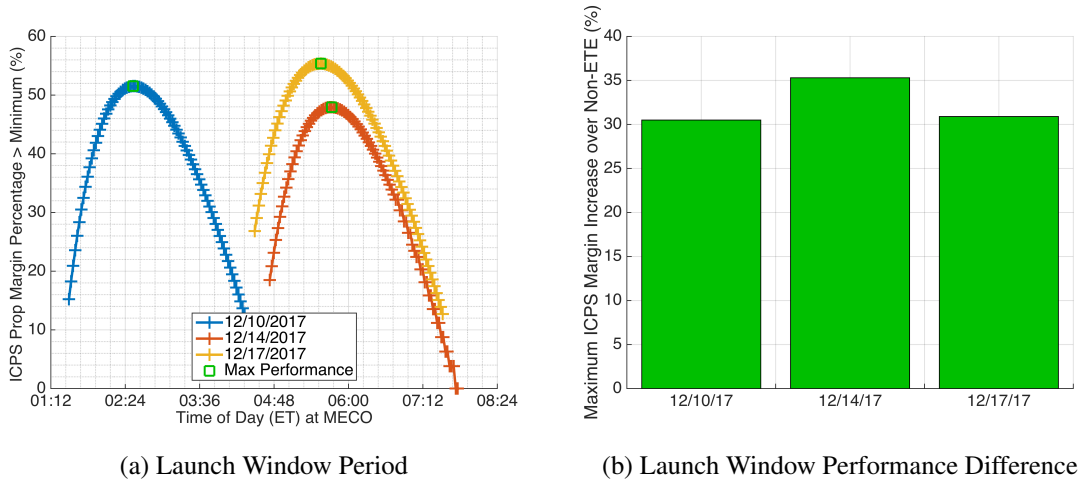


Figure 4: ETE Results for EM-1 Launch Window Analysis

SLS Block 1 TLI Performance Study

The *ETE Plugin Optimizer* was then used to produce performance analysis for SLS and Orion’s first two missions: Exploration Mission 1 (EM-1) and Exploration Mission 2 (EM-2). This performance analysis assessed the maximum payload capability through the Trans-Lunar Injection (TLI) burn of the SLS Block 1 configuration over a monthly launch period. For both missions, the SLS Block 1 configuration is required to send Orion on a lunar intercept trajectory. Orion then performs the remaining burns to complete the desired mission. Payload was maximized through TLI and Orion propellant usage was minimized in order to complete the mission. Any mass above the mass of Orion would be considered additional payload. It was conservatively assumed that the respective EM-1 or EM-2 control mass of Orion was used for the post-TLI portion of each trajectory. By tying the ascent and on-orbit phases of the mission the tool affords analysts the ability to optimize the ascent and on-orbit phases at the same time. This produces a valid optimal mission that can be flown end to end.

The reference trajectory for EM-1 assumes that after Main Engine Cut-Off (MECO) of the SLS core stage, the Interim Cryogenic Propulsion Stage (ICPS) and Orion are placed into a 28.5° inclined, 22 x 975 nmi altitude orbit. For EM-2 the post MECO orbit is also inclined at 28.5° but with

a larger relative altitude at 19×1175 nmi. The combined spacecraft stack then propagates up to apogee and ICPS performs the Perigee Raise Maneuver (PRM) to increase the perigee of the orbit to 100 nmi placing the stack into a closed orbit above the atmosphere which is referred to as the Earth Parking Orbit (EPO). Then TLI is performed near the second perigee passage by the ICPS to place Orion on a trajectory towards the moon. The mission profiles post-TLI differ for EM-1 and EM-2; however the maximum payload study was primarily focused on the common EPO portion of the two missions.

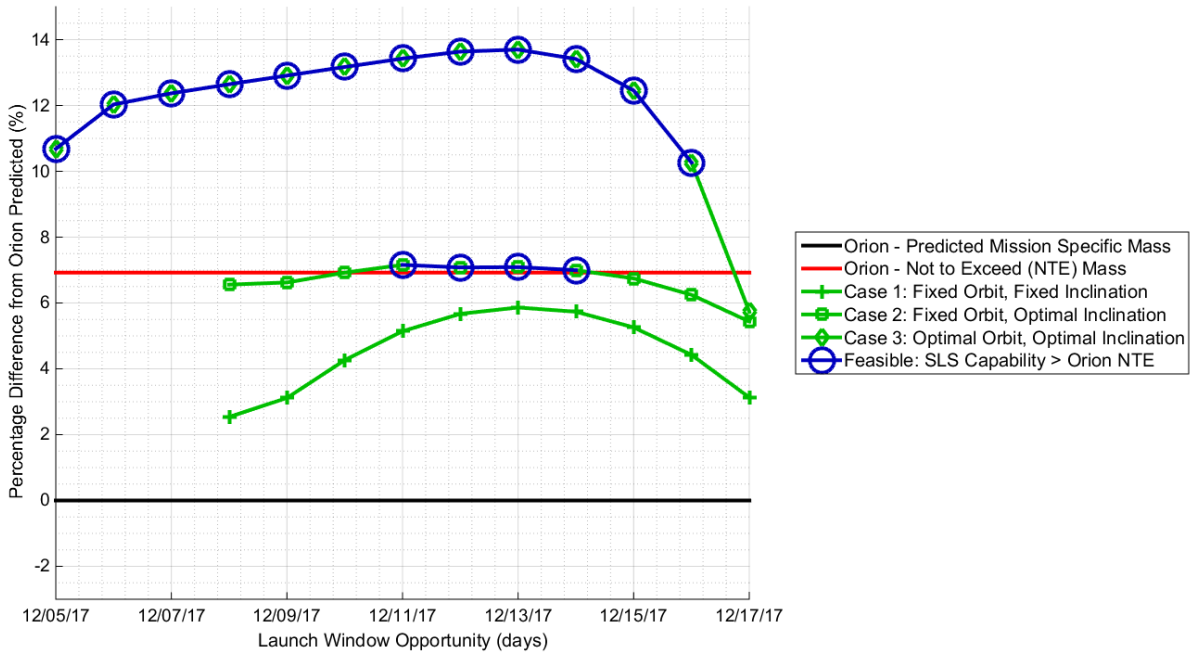
Figure 5 shows the results of various cases run with the *ETE Plugin Optimizer* for EM-1 and EM-2 respectively. Each point on the plots that are above the lightest Orion mass (horizontal lines) indicates a valid mission that allows a particular mass of Orion to perform its mission and return back to Earth via an acceptable Entry Interface (EI) condition. All the points shown are above the lowest Orion mass line, however the optimizer could be run on more launch days on either side of the current launch dates with decreasing payload performance through TLI. The points shown on the plots were sufficient for the purposes of these studies. Each study discussed below assessed three different cases:

- Case 1: Assumes a fixed orbit altitude and inclination.
- Case 2: Assumes a fixed orbit altitude and an optimized inclination.
- Case 3: Assumes an optimized orbit altitude and inclination.

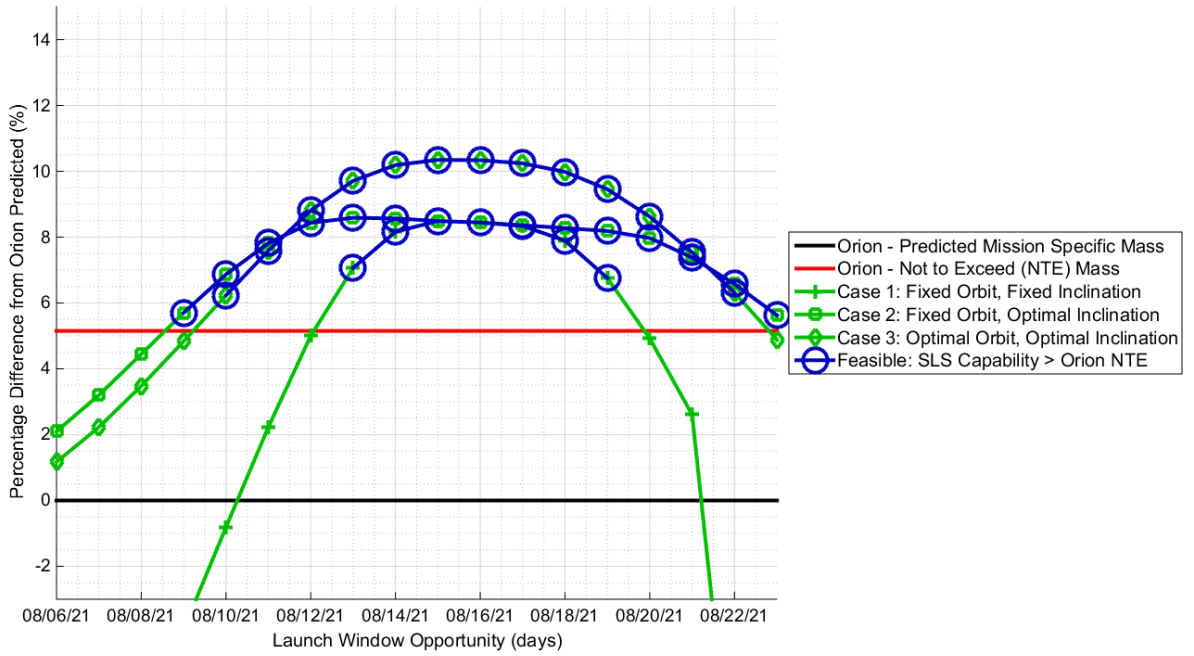
Case 1 for both missions show a variation in the performance through TLI even though the orbit parameters are fixed. This is due to the change in Earth-Moon geometry throughout the lunar month. All of the EPOs are elliptical and therefore the Moon has to be in the southern hemisphere in order to do a TLI burn near perigee instead of at apogee. This is because perigee will always be in the northern hemisphere since the primary launch site, Kennedy Space Center, is in the northern hemisphere. This effectively cuts out at least half of the lunar month to perform a mission that is within reasonable performance requirements.

Case 2 has a different effect for both EM-1 and EM-2. This is due to the lunar declination difference between launching in 2017 and 2021. The optimal inclination for EM-1 in 2017 is closer to 32° instead of 28.5° . However, for EM-2 in 2021 the optimal inclination is 28.5° , which is the inclination for the reference EPO for both EM-1 and EM-2. Therefore in the middle of the monthly opportunity window there is minimal difference in performance through TLI. The edges of the window do see an increase in performance relative to Case 1 because a higher inclination is more optimal. The edges of the opportunity window have inclinations close to 45° . This might be too high of an inclination for the SLS to fly due to launch pad constraints, but for the purposes of this study the middle of the window is the most important comparison.

Case 3 is where the full power of the optimizer can be realized. By allowing the EPO (both the perigee and apogee) to be optimized as well as the inclination the launch day specific optimal orbit can be found. This allows the full performance margin of the SLS core stage to be applied to the ICPS and through TLI by adjusting the EPO into an optimal orientation relative to the Moon's geometry on that day. It is true that this performance gain could be derived by using POST2 and Copernicus separately, but a large amount of manual iteration would be required on the part of the analyst. Using the optimizer also ensures that the optimal trajectory found in Copernicus also corresponds to a flyable SLS ascent trajectory and disposal conditions for both the core stage and ICPS.



(a) EM-1 Results



(b) EM-2 Results

Figure 5: ETE EM-1 and EM-2 Launch Period Analyses

Fully optimizing the EPO with Case 3 provides the most payload through TLI and extends high payload capability farther through the opportunity window in addition to lengthening the opportunity window as well. From an operations/real-time perspective having a different EPO each day only requires a new set of i-loads to be uploaded to the vehicles prior to that day's launch attempt. The apogee altitude of the EPO increased to nearly 1300 nmi for both EM-1 and EM-2, but the optimizer maxed out there. Also, an ICPS vehicle propellant penalty was applied to all Case 3 results to protect for other vehicle considerations due to the variation in EPO orbit altitude. This causes some of the Case 3 TLI payload results to be slightly below the Case 2 payload results. This can be seen directly in Figure 5b.

ISIGHT ETE OPTIMIZER RESULTS

Launch Period Analysis

Similar to the results described in detail in the *ETE Plugin Optimizer* results section, the *ETE Isight Optimizer* also analyzed EM-1 cases using the Block 1 SLS. The first analysis consisted of determining the availability of daily opportunities within a given launch period. The optimization problem consisted of determining the optimal MECO time for a given day within a given launch period and was solved using the MMFD¹⁴ solver. The control variables included launch azimuth and apogee with the overall optimization objective to minimize TLI propellant. In Figure 6 launch days 4-7 demonstrate the best TLI performance, with over 25 higher TLI propellant remaining compared to the rest of the 11 day period shown. These particular days maintain enough of an optimal alignment within the confines of the Earth-moon geometry that allows maximizing the performance by aligning the TLI burn closely with the perigee of the intermediate parking orbit.

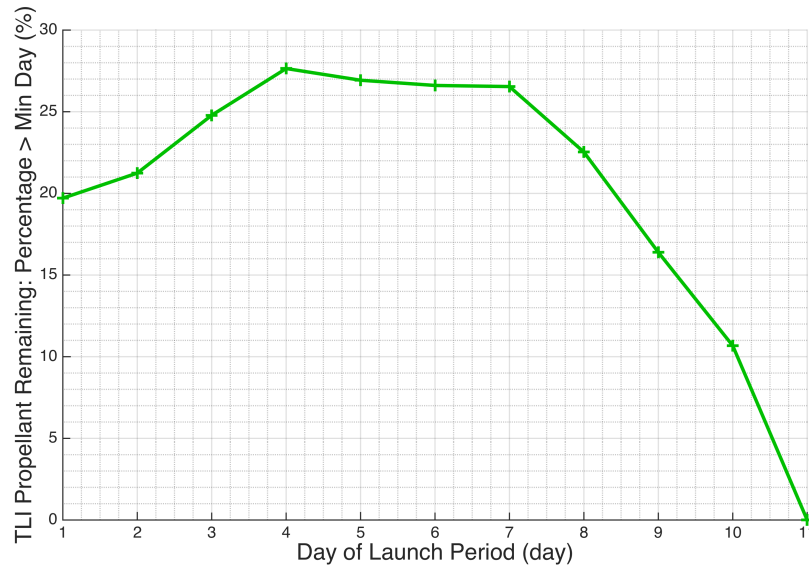


Figure 6: Isight Results for EM-1 Launch Period Analysis

Launch Window Analysis

While the launch period analysis produced macro results, a more detailed launch window analysis in which the optimal value was refined within a given time of day, required a more suitable optimizer.

Thus, for the launch window analysis, in which the ability to maintain a two-hour launch window was to be determined, the MISQP¹⁵ optimizer was used. Once again the control variables were launch azimuth and apogee and the objective was to minimize total TLI propellant with a fixed SM propellant load. As the earth-moon geometry comes out of perfect alignment the TLI cost increases substantially and thus the window peaks in the middle of the period, as shown in Figure 7a.

In order to compare the effect of different control variables, the analysis was rerun, but with a fixed apogee target while azimuth remained free. As would be expected, fixing a control variable reduced the performance, but not necessarily significantly. In Figure 7b it can be seen that the performance gain with a free apogee maxes out at 60 kg. Sensitivities to individual variables such as these are important to understand to maximize performance for missions where multiple independent parameters can contribute to a combined net increase in performance.

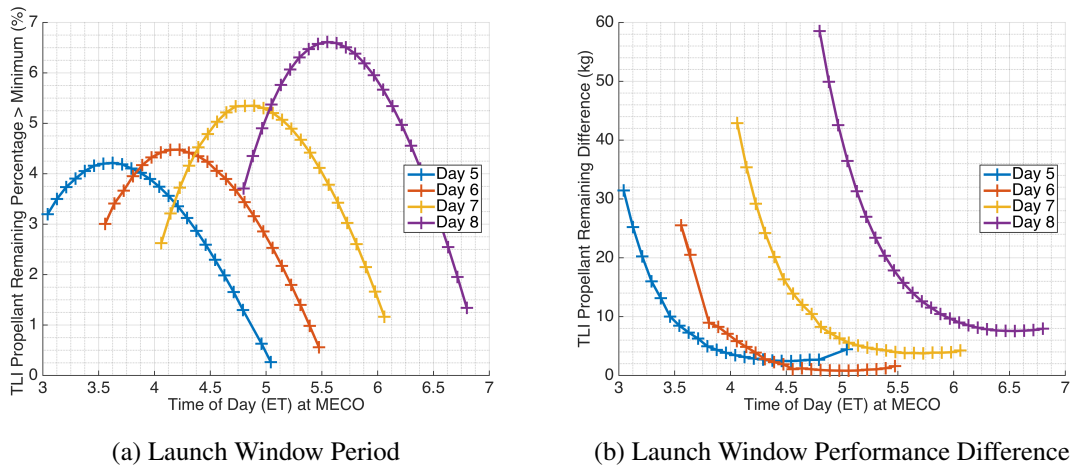


Figure 7: Isight Results for EM-1 Launch Window Analysis

ETE OPERATION & CHALLENGES

Each method implemented provided unique challenges for the analysts to tackle. The ability to achieve convergence still varies from case to case and the techniques to overcome the convergence challenges are detailed in the following sections for each optimization method. Lessons learned from these early test cases will aid significantly in solving future optimization problems.

Using the ETE Plugin Optimizer

While the optimization algorithms for the *ETE Plugin Optimizer* are fairly established, it is the tuning of the decks that remains a challenge and the lengthy run times that are currently measured in hours. Up to this point, two types of problems have been solved using the *ETE Plugin Optimizer* with varying degrees of success. The first features the Block 1 configuration of the SLS using the Interim Cryogenic Propulsion Module (ICPS) upper stage and the second features Block 1B of the SLS using the Exploration Upper Stage (EUS). The Block 1 results have been successfully generated while Block 1B results are still being developed.

SLS Block 1: ICPS Configuration The implementation used the Copernicus v4.1 program, the POST2 DAC3 executable, and the Python Module in *S-OPT* mode. Missions analyzed were the

EM-1 Distant Retrograde Orbit (DRO) mission and the EM-2 High Lunar Orbit (HLO) mission. For both missions, POST2 is used to model the ascent from launch to MECO with Copernicus driving the optimization. For the in-space portion of EM-1, Copernicus models the complete DRO mission from MECO to EI while minimizing the Orion total ΔV and maximizing the post-TLI mass. For EM-2, Copernicus models the first phase of the HLO mission, which is a free return flyby around the Moon to EI, while maximizing the post-TLI mass. For both cases, EI is a point along a targetline that ensures a water landing off the coast of southern California.

Depending on the particular problem being solved for Block 1 SLS, Copernicus was programmed to optimize between 30 and 60 optimization variables along with 25 to 55 constraints and between 1 to 4 variables for the overall objective function. Of these optimization variables, Copernicus passes 13 independent variables and 12 dependent variables (for the constraints) as input to the POST2 sim via the plugin module scripts. POST2 runs and sends back a state vector (with 6 geographic parameters) and mass at MECO + 20 seconds. These values are then pushed into the initial segment of the in-space trajectory simulation. This cycle is repeated until an optimal solution is found. Solution convergence is dependent on the type of problem, how many controls and constraints there are, as well as the solution tolerance. For example, as the number of modeled maneuvers decreases, the optimizer will converge faster to a solution. Also, reducing the number of variations to the orbits (i.e, the orbital elements) can also help the optimizer converge faster. As the tolerance decreases, the optimizer tends to work harder and longer to converge to a solution.

The EM-1 mission is more complicated than the first phase of the EM-2 mission. To meet this challenge, one must constrain the problem to help the *ETE Plugin Optimizer* find a non-optimal solution, then release the constraints to find an optimal solution. Two ways to constrain the problem is to *freeze* the POST2 plugin and the other is to *freeze* the DRO insertion state and epoch. When the POST2 plugin is frozen, only the in-space trajectory is optimized. When the DRO is frozen, the outbound trajectory has a consistent target and usually the remaining part of the mission from DRO insertion to EI does not change much. Since SNOPT is used as the optimizer, one must ensure that the optimization variables are scaled properly. Most of the time, 1:1 scaling is used so that all variables will be tweaked to find a feasible and optimal solution. Also, loosening the feasibility or optimality tolerances has been beneficial in finding an initial solution, and then tightening it back to get the real solution. Thus, optimizing a run requires quite a bit of user intervention, especially stopping and restarting a run that is drifting away from a solution. The total run time can vary from around 30 minutes to 2 hours. In comparison, doing this manually (running each tool separately) will take several days in order to arrive at a reasonable solution.

SLS Block 1B: EUS Configuration While the SLS Block 1B configuration is not as mature as the Block 1, there is sufficient data to start modeling the newest configuration in the optimizer. The Block 1B Core and Solid Rocket Boosters (SRBs) are identical to the Block 1 and the only difference is the upper stage. The Block 1B is planned to use a larger 4-engine stage Exploration Upper Stage (EUS) that will perform a significant portion of the ascent to the parking orbit as well as the TLI burn. This is in contrast to the Block 1 where the upper stage only performs a short burn to raise perigee to at least 100 nmi.

In modeling the Block 1B, the initial approach was similar to the Block 1; fly the Core and SRB's in POST and then model the EUS in Copernicus. This was possible because the EUS is at a sufficient altitude at its first ignition that atmospheric effects are negligible. Even though the EUS is only approaching the PDR level, parser plugins in Copernicus were required to adequately model the propellant boiloff and total fuel consumption. Another challenge was calculating the Flight

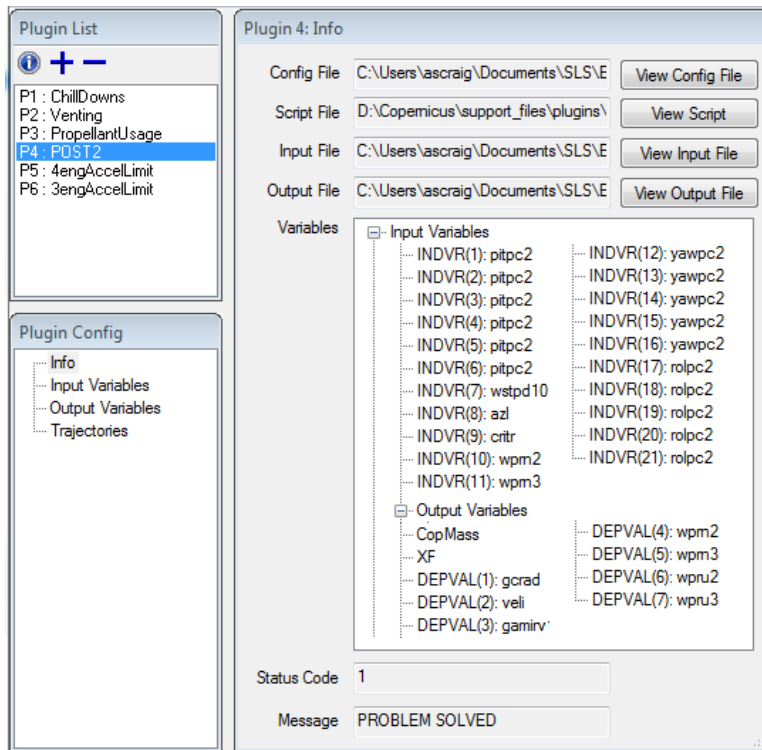


Figure 8: Plugin List for the SLS Block 1B

Propellant Reserve (FPR), which is typically calculated as a percentage of mission ideal delta-V for preliminary analysis. All of this was accomplished through the use of the parser plugins and linking those plugins through the Copernicus GUI. An example of this can be seen in Figure 8 with the POST2 plugin highlighted to show the inputs to POST and outputs back to Copernicus. To date, only a handful of point cases have been produced to show that the Block 1B can also be optimized utilizing the POST-Copernicus combo.

After the preliminary setup, an initial trajectory for a DRO type mission took approximately 5 hours to converge with a significant amount of user intervention required. At times, the gradients can be very flat and the problem may go beyond the vicinity of the local optimal so the analyst must push it back closer to a feasible solution. This can either be done mathematically or visually depending on what constraints are being violated. For example, time and mass continuity can typically be fixed mathematically, but states are easier to match up by using the Copernicus GUI and adjusting the maneuvers manually until the solution is close.

The launch date was then moved 1 day later and the solution converged in just over an hour with only limited user intervention. Both trajectories met all constraints and were stated as optimal by SNOPT. One downside to this optimization and similar to what was seen with the Block 1 trajectories was the inability to fully optimize the launch time of day. To work around this, the POST plugin was initially frozen and all other segments were optimized to include the launch time of day. Once that converged to a solution, the POST2 plugin was reactivated and the launch time was frozen to optimize from liftoff to Earth entry interface.

Future work on the Block 1B includes a full weekly launch period and daily launch window

analysis for EM-2 and trading running POST open-loop vs closed-loop in the optimization. The daily launch window analysis will determine if the launch time was truly optimal or if the launch time would need to be shifted to take advantage of more better Earth-Moon alignment.

Using the ETE Isight Optimizer

Through the course of setting up a new tool it always takes some time to work out the best set of optimization algorithms, processes and other problem specific characteristics to achieve convergence. To that end, each phase of the *ETE Isight Optimizer* had independent optimization algorithms. For the ascent problem while POST2 offers internal optimizers, the Large Scale Generalized Reduced Gradient (LSGRG) optimizer within Isight was selected due to its rapid convergence to the optimal MECO state. For the on-orbit phase the internal Sparse Nonlinear OPTimizer (SNOPT)¹² was used to minimize the amount of TLI propellant required. For the entry phase, the internal POST2 Projected Gradient Method (PGM) optimizer was used to converge on a landing location. At this point the entry simulation does not feedback into the global optimization problem, so Copernicus fed data to Isight for the overall optimization loop directly to minimize the total propellant. For the overall optimization problem two different optimizers were employed: the Modified Method of Feasible Dimensions (MMFD)¹⁴ and the Mixed Integer Sequential Quadratic Programming (MISQP)¹⁵ optimizer.

The ETE Isight Optimizer process is designed to be easily modified to handle changing program requirements and trade studies during design cycles. One of the key aspects of this process was for it to be automatic, as the hallmark of an end-to-end sim should make it easier to use. In order to do this, it was necessary to use previous Copernicus solutions as seeds for the next iteration. While Copernicus has global search capabilities, an on-orbit problem should have an initial guess in the vicinity of the desired local extrema, as gradient or quadratic based methods hone in on these local points. The branching capability inherent to Isight was useful in making this possible.

CONCLUSION

An end-to-end optimization capability combining simulations designed to run independently has been demonstrated. Concurrent optimization with a large number of independent variables and constraints can achieve convergence when the optimization problem is well defined and an initial guess is in the vicinity of the optimal. The biggest challenge for these large optimization problems is establishing good initial guesses; to that end a process for building initial guesses is beginning to form for the problems described. Future analyses will continue to explore the capability of an end-to-end tool as well as an expansion of the problem to include other phases of flight.

REFERENCES

- [1] J. Williams, *Copernicus Version 4.2 User Guide*. NASA Johnson Space Center, July 2015. JETS-JE23-15-AFGNC-DOC-0052.
- [2] S. Striepe, *Program to Optimize Simulated Trajectories (POST2), Vol. 2: Utilization Manual, Ver 3.0*. NESC, NASA Langley Research Center, May 2014.
- [3] J. Williams, "A New Plugin Architecture for the Copernicus Spacecraft Trajectory Optimization Program," *AAS/AIAA Astrodynamics Specialist Conference*, August 2015. AAS 15-606.
- [4] C. Hargraves and S. Paris, "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, 1987, pp. 338 – 342.
- [5] *Simulation and Optimization of Rocket Trajectories, Version 9.0*. Houston, TX: Lockheed Engineering and Sciences Company, 2008. Contract NAS 9-17900.

- [6] J. Sims, P. Finlayson, E. Rinderle, M. Vavrina, and T. Kowalkowski, "Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design," *AAS/AIAA Astrodynamics Specialist Conference*, August 2006. AIAA 2006-6746.
- [7] J. Williams, J. S. Senent, and D. E. Lee., "Recent Improvements to the Copernicus Trajectory Design and Optimization System," *Advances in the Astronautical Sciences*, Vol. 143, January 2012. AAS 12-236.
- [8] J. P. Gutkowski, T. F. Dawn, and R. M. Jedrey, "Trajectory Design Analysis over the Lunar Nodal Cycle for the Multi-Purpose Crew Vehicle (MPCV) Exploration Mission 2 (EM-2)," *Advances in the Astronautical Sciences: Guidance, Navigation and Control*, Vol. 151, 2014. AAS 14-096.
- [9] J. Williams and G. L. Condon, "Contingency Trajectory Planning for the Asteroid Redirect Crewed Mission," *AIAA SpaceOps 2014*, May 2014. AIAA 2014-1697.
- [10] C. Ocampo, "An Architecture for a Generalized Trajectory Design and Optimization System," *Proceedings of the Conference: Libration Point Orbits and Applications* (G. Gómez, M. W. Lo, and J. J. Masdemont, eds.), World Scientific Publishing Company, June 2003, pp. 529–572. Aiguablava, Spain.
- [11] *The JSON Data Interchange Format*. ECMA International, October 2013.
- [12] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scaled Constrained Optimization," *SIAM Review, Society for Industrial and Applied Mathematics*, Vol. 47, No. 1, 2005, pp. 91–131.
- [13] *Isight 5.0*. Providence, RI: Dassault Systèmes Simulia Corp., 2014. Release 5.9-1.
- [14] G. N. Vanderplaats, "An Efficient Feasible Directions Algorithm for Design Synthesis," *AIAA Journal*, Vol. 22, No. 11, 1984, pp. 1633–1640.
- [15] O. Exler, T. Lehmann, and K. Schittkowski, "A Comparative Study of SQP-Type Algorithms for Non-linear and Nonconvex Mixed-Integer Optimization," *Mathematical Programming Computation*, Vol. 4, No. 4, 2012, pp. 383–412.