AAS 23-128

COPERNICUS-LINCOV (COPCOV) SOFTWARE INTEGRATION IN SUPPORT OF ROBUST TRAJECTORY OPTIMIZATION

Joshua K. Geiser*, David Woffinden†, and Matt Horstman‡

Robust trajectory optimization is the process of optimizing a trajectory while accounting for system uncertainty due to a variety of potential error sources. This work highlights the development and features of a novel tool known as CopCov to support robust trajectory optimization efforts. CopCov acts as an interface between Copernicus, a generalized trajectory design and optimization tool, and LinCov, a linear covariance analysis tool. By having a direct interface between these two software packages, Copernicus can receive covariance information from LinCov through a direct feedback loop, thus enabling optimization of a trajectory that is robust to trajectory dispersions and navigation errors. This paper details the architecture of CopCov and its flexibility to operate under varying configurations, including with both tools running locally or alternatively with the tools communicating via a remote connection. Additionally, the CopCov tool is demonstrated on a simple Hohmann transfer reference trajectory with varying numbers of Trajectory Correction Maneuvers (TCMs) and varying problem formulations. This example scenario is used to highlight how the inclusion of the CopCov interface affects burn placement of both major burns and minor burns (i.e., TCMs) in the optimized solution. Results are compared against analytical solutions and against a Genetic Algorithm (GA) optimizer for independent verification and validation.

INTRODUCTION

The trajectory design process seeks to find a flight profile that meets a set of specified mission objectives while ensuring that any trajectory constraints imposed on the vehicle are not violated. Most frequently, some form of numerical optimizer is used throughout the trajectory design process to generate a feasible trajectory that is optimal with respect to a specified performance index, such as propellant usage or total ΔV. This process informs the timing and placement of major burns (i.e., nominally non-zero ΔV) along the flight path as well as other trajectory events, such as Sphere of Influence (SOI) changes, eclipses, flybys, etc. The resulting trajectory is often used as a reference or baseline to support other subsystem analyses.

Another important aspect of the mission design process is the analysis of a spacecraft's Guidance, Navigation, and Control (GN&C) system. In reality, no GN&C system is capable of perfectly flying its intended reference trajectory. A variety of error sources are introduced during flight, including navigation errors, maneuver execution errors, unmodeled dynamics, and more. This reality necessitates the ability to characterize the performance of the GN&C system in the presence of these error sources through preflight integrated simulations. Traditionally, Monte Carlo[1] simulations and/or linear covariance techniques[2,3] are used to assess GN&C system performance. These analyses can,

*Aerospace Engineer, Flight Mechanics and Trajectory Design Branch, NASA Johnson Space Center
†Aerospace Engineer, GN&C Autonomous Flight Systems Branch, NASA Johnson Space Center
‡Aerospace Engineer, Barrios Technology

for example, help inform placement of minor correction burns (i.e. nominally zero ΔV) in order to correct dispersions from the reference trajectory. This inclusion of covariance analyses leads to the notion of robust trajectory optimization, where a trajectory is optimized while accounting for system uncertainty.[4–13]

The completion of the trajectory design process independent of the integrated GN&C analyses is not uncommon. For example, a generalized trajectory design and optimization tool (e.g., Copernicus[14]) is used to support trajectory design work, while the integrated GN&C performance analysis evaluating the proposed trajectory's feasibility is accomplished with a separate Monte Carlo or linear covariance (LinCov) analysis tool. While there are advantages to this workflow due to some natural delineation between these two fields of flight mechanics and GN&C system design, the process is not fully decoupled. Depending on the flight phase, they can actually be closely coupled. As such, this disconnect between analyses creates a few notable problems. First, if the constrained optimization problem fails to account for the proposed GN&C system and the inherent uncertainty, the optimal solution is typically situated on the boundaries of the design space defined by the constraints, leaving little to no margin for error. This eventual discovery, which is often later in the design process, forces a redesign manifesting in additional cost and schedule slip. Secondly, this iterative design process ultimately requires data products being passed back and forth between the trajectory design tool (e.g., Copernicus) and the GN&C system analysis tool (e.g., LinCov) to converge on a final end-to-end trajectory design. This integration gap between the two tools causes unnecessary and significant delays. Substantial time and effort is often required to develop the output data products from one tool, send those to a separate team, have that team rerun their analysis and send updates back, and so on. Lastly, if this capability is to be extended to real-time operational support for burn targeting and trajectory optimization, then these two tools need to be more tightly integrated so that they are able to communicate in a fast, automated, and reliable fashion.

This paper highlights the development and testing of the CopCov tool which seeks to address the above issues by filling the gap in integration between Copernicus and LinCov. First, a brief background is given into the Copernicus and LinCov tools as well as a discussion on how these tools have traditionally been used to support robust trajectory optimization work to illustrate the current challenges in the design process. The next section highlights the initial architecture of the CopCov tool and how this architecture supported its development and initial verification purposes. Next, details are provided on an end-to-end CopCov architecture that can allow more flexibility in communication between Copernicus and LinCov over a remote server. The following sections provide verification and validation of the tool through reference trajectory testing and comparison of results using multiple options, including a mission map, an integrated GN&C performance emulator, the full LinCov tool in-the-loop, and a Genetic Algorithm (GA) optimizer.[15] Last, a conclusion is provided to summarize the work completed on the CopCov tool thus far and highlight future work to extend its capabilities.

BACKGROUND

To demonstrate the challenges and dependencies between the flight mechanics trajectory design and the vehicle's GN&C system design, a brief overview of the Copernicus and LinCov tools is provided, as well as how they have traditionally been used to support robust trajectory design and optimization efforts.

Copernicus

Copernicus[16,17] is a generalized spacecraft trajectory design and optimization tool that serves as the backbone for many trajectory design efforts within NASA. To construct a trajectory, users specify different trajectory segments which may delineate various phases throughout the flight, such as finite burns, impulsive burns, coasts, sphere-of-influence (SOI) changes, gravity assists, and more. One of the most powerful features of Copernicus is the ability to setup constrained optimization problems and produce solutions in the form of reference trajectories. Copernicus comes with a variety of off-the-shelf numerical optimizers installed (e.g., SNOPT[18]) to support direct targeting and/or optimization efforts. A user can specify optimization variables (e.g., burn durations, burn directions, perilune pass altitude, etc.), constraints on the trajectory (e.g., entry interface conditions, state continuity between segments, etc.), and an objective function (e.g., total ΔV minimization). With these parameters specified and a reasonable initial guess for the trajectory, a user can "Iterate" with the selected optimizer to converge on an optimized trajectory solution.

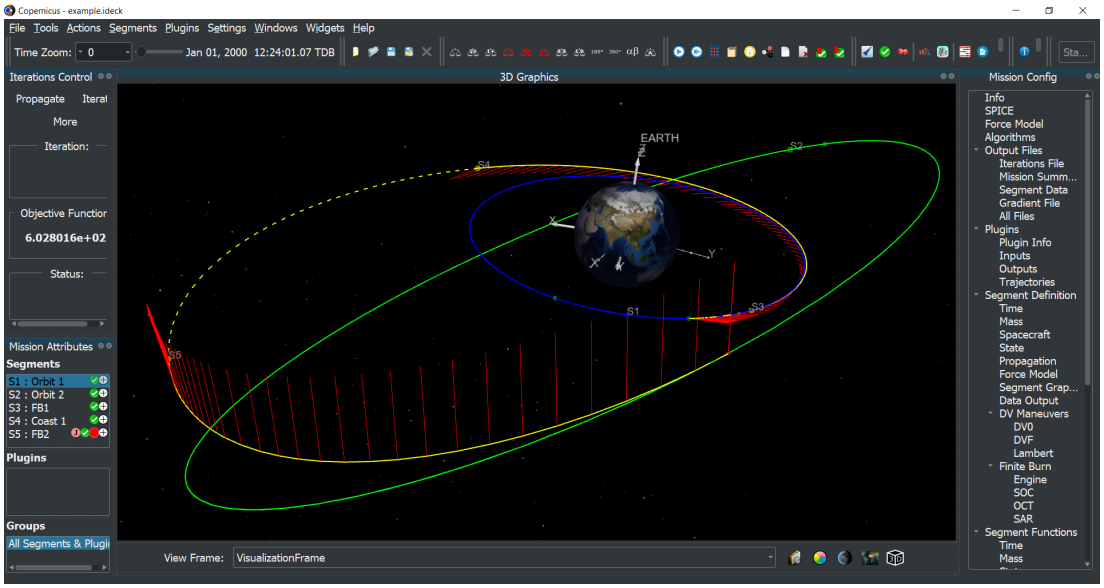


Figure 1. Copernicus GUI with an example mission. Note the (empty) *Plugins* pane on the bottom-left.

Copernicus also provides support for calling external *plugins* as a part of the propagation and optimization processes.[14] These plugins are user-developed scripts to provide extended capabilities not native to Copernicus or they can act as interfaces to other software packages. Copernicus contains a Python[19] Application Programming Interface (API) that allows it to efficiently call Python plugins by treating the plugin as a Python package that can be called from the already-running Python instance.[20] A user adds plugins from the Copernicus Graphical User Interface (GUI), and these plugins affect the optimization problem and its resulting trajectory solution.

LinCov

LinCov is a MATLAB[21]-based linear covariance analysis tool used for integrated GN&C analysis. The tool utilizes both augmented state and onboard state covariance matrices to produce navigation errors and trajectory dispersions. Used extensively over the past several decades, LinCov can support numerous flight phases, including powered ascent, rendezvous, proximity operations, and

docking (RPOD), cis-lunar and interplanetary, aerocapture, attitude control, and entry, descent, and landing (EDL).[22–40] Whereas Copernicus is used to generate nominal reference trajectories, LinCov is able to assess how well a particular GN&C system can follow the desired reference profile given the complex system uncertainty connected with the trajectory design. This uncertainty quantification allows LinCov to support the identification of the optimal placement of translational burns that minimize not only the nominal ΔV, but rather the *total* ΔV. This total ΔV includes both the nominal ΔV (typically derived from the trajectory design tool such as Copernicus) and the expected 3σ ΔV dispersions.

A substantial benefit of using a LinCov tool is speed. Key performance metrics of a closed-loop GN&C system are reliably obtained in a single simulation run, as opposed to Monte Carlo techniques that traditionally take hundreds or thousands of runs to obtain similar statistical information. These metrics include true dispersions, navigation dispersions, true navigation error, and onboard navigation error. In particular, the metrics highlighted in this paper are the 3σ position dispersions (i.e., how precisely the GN&C system can follow the reference trajectory in position) and the 3σ ΔV dispersions (i.e., how much extra ΔV is expected due to the imperfect execution of maneuvers).

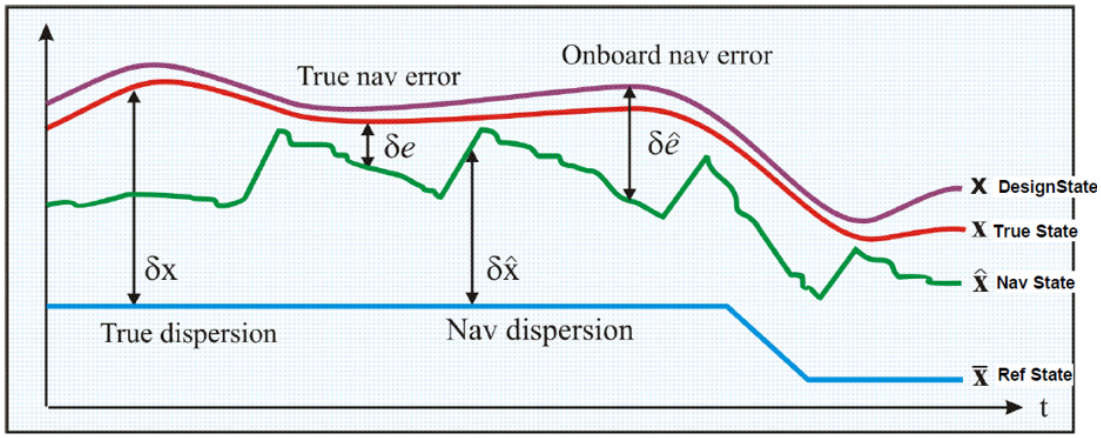


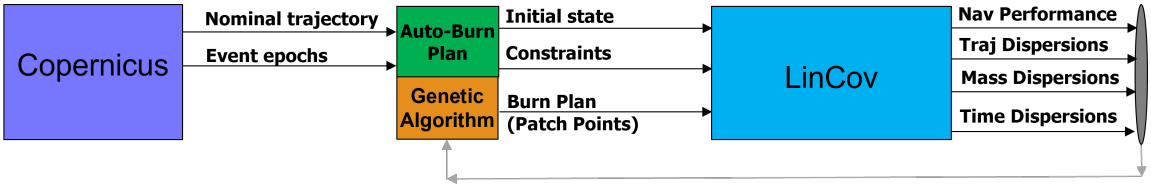
Figure 2. Definitions of some key performance metrics obtained using LinCov

Robust Trajectory Design Overview

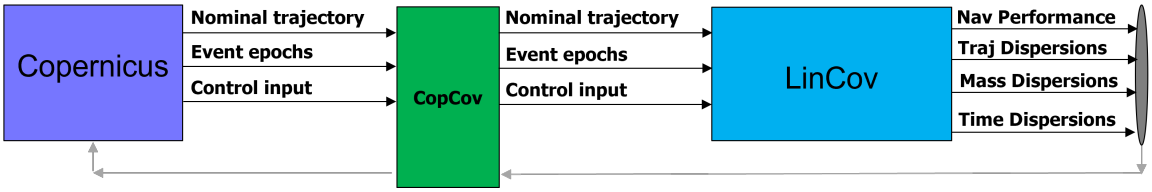
Process flow schematics help provide a visual model for the complex interface between Copernicus and LinCov. Figure 3 highlights both the legacy and newly developed workflows for integrating Copernicus and LinCov to support robust trajectory optimization efforts. Figure 3(a) provides a schematic overview of the legacy implementation. Copernicus produces a nominal profile with an optimal reference trajectory, including the placement of major translational burns. Key information from this trajectory is extracted using a tool known as Auto Burn Plan (ABP) to generate a burn plan representative of the mission that identifies the timing, magnitude, and direction of each burn and its corresponding targeting constraints. This burn plan is used by LinCov to generate the integrated GN&C performance metrics while running in-the-loop with a Genetic Algorithm (GA) to optimize the selected performance parameters, such as trajectory dispersions, total ΔV, or a combination of both.

The newly developed workflow, shown in figure 3(b), replaces the GA optimizer with Copernicus itself, leveraging its extensive optimization routines and heritage. For this architecture to function,

it is necessary to have an interface that allows data to transfer between Copernicus and LinCov in a fast, automated, and reliable manner. This key interface is referred to as CopCov, or the Copernicus-LinCov plugin tool. This tool provides the capability to perform end-to-end trajectory design and analysis during all phases of the design cycle, ranging from preliminary trajectory design to mission operations, all while allowing the trajectory design to be robust to the vehicle's GN&C system design.



(a) Schematic of the legacy robust trajectory optimization workflow



(b) Schematic of the newly developed robust trajectory optimization workflow

Figure 3. Robust Trajectory Design Architectures

DEVELOPMENT ARCHITECTURE

To develop, test, and verify the CopCov tool, two different software architectures are utilized. The complexity of LinCov introduces challenges in the development of interfacing software. First, repeated LinCov calls increase the computation time required to test the interface, slowing down development. Additionally, the correct processing of inputs and computation of outputs is difficult to verify. To mitigate these challenges, a simple emulator was developed that mimics the LinCov outputs for a given set of inputs. The initial development architecture of CopCov was then designed to allow for flexibility in calling either the emulator or the full LinCov tool, assuming they are available on a user's local machine. Due to constraints regarding general access to the LinCov tool, the end-to-end architecture (discussed further in the following sections) makes accomodations to incorporate the use of a remote connection to the full LinCov tool.

LinCov Emulator

The LinCov *Emulator* supports initial development and verification efforts and may also prove to be a long term solution that incorporates the concept of using pre-generated LinCov data to reduce run time. The LinCov Emulator contains the same interface as the actual LinCov tool but returns a fast and specified functional output. Due to its simplicity, the Emulator drastically reduces the runtime of individual calls to the CopCov plugin by Copernicus. The LinCov Emulator essentially treats the LinCov tool as a *black box* where a predetermined set of inputs $\vec{x}$ and outputs $\vec{y}$ are known. It then calculates the set of outputs $\vec{y} = f(\vec{x})$ for a set of inputs $\vec{x}$ from Copernicus. Consequently, the only difference between LinCov and the LinCov Emulator is the internal mapping function $f(\vec{x})$ which can be characterized to varying levels of fidelity.

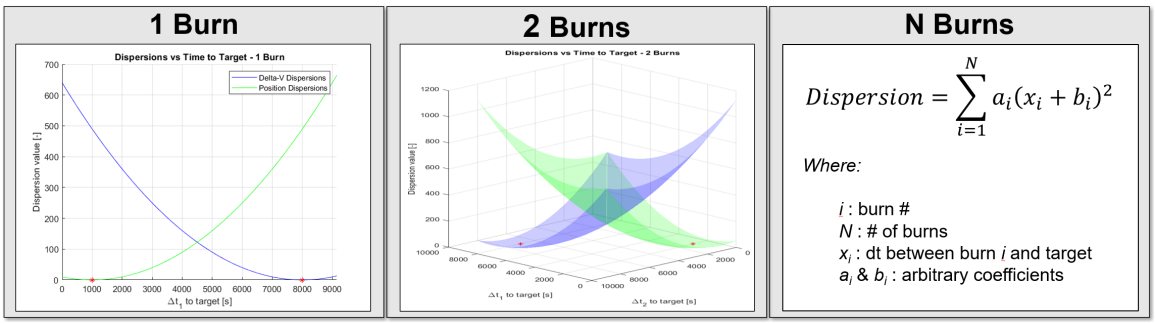


Figure 4. Illustration of the quadratic functions used by the LinCov Emulator for a variable number of burns

For early development and testing, quadratic functions were used as the primary mapping function of inputs to outputs in the LinCov Emulator, as illustrated in Figure 4. Due to their smoothness and convexity, they optimize well with the gradient-based optimization algorithms internal to Copernicus. Additionally, they provide an easy and intuitive way to scale to multivariate input spaces and place minima at specified locations. Figure 16 in the Appendix provides an example of how quadratic functions were initially used to strategically place the optimized solution given a multi-dimensional input space. Due to the flexibility of the Emulator, these quadratic functions are easily replaced with multivariate polynomial fits of more realistic dispersion functions, as discussed and illustrated later in the results.

Data Flow Pipeline

All the overhead cost of opening Python and MATLAB instances is completed upon Copernicus startup and plugin initialization, speeding up the actual optimization process. Figure 5 illustrates how CopCov handles the flow of data for each iteration of Copernicus' optimization routine.

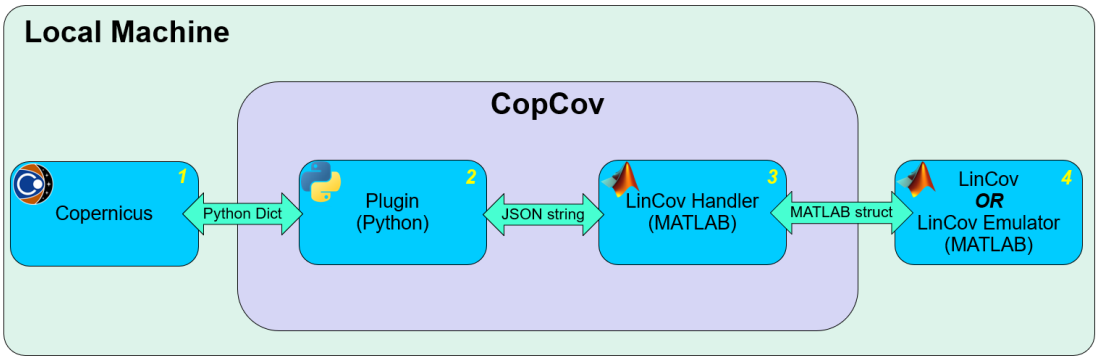


Figure 5. Initial software architecture for local data flow between Copernicus and the LinCov Emulator / LinCov

First, Copernicus passes raw segment data to the Python plugin (e.g., burn state vectors, burn TIGs, target state). The plugin parses this data, extracts pertinent metrics (e.g., ΔV vectors), and repackages it into a standardized JSON format before passing it off to the MATLAB-based LinCov Handler. The LinCov Handler repackages the input data into a more usable MATLAB-specific format and calls either the LinCov Emulator or LinCov itself, based on user-specified flags. Since both the Emulator and LinCov share the same set of expected inputs/outputs, they are easily inter-

changed. Once output dispersion values are calculated, these quantities are passed back through the pipeline from MATLAB to Python to Copernicus. Finally, Copernicus ingests this data into the optimization algorithm and continues iterating until convergence.

END-TO-END ARCHITECTURE

For increased flexibility and ease-of-access of the proprietary LinCov tool, the end goal was to develop a software architecture where CopCov users could make remote calls to LinCov without having read/write access to the underlying source code. This created some added challenges, as CopCov would need the ability to handle transfer of data packets across a remote connection. However, this would ultimately allow Copernicus to communicate directly with LinCov while both packages operate on different machines.

Connecting to a Remote Server

The Flight Sciences Laboratory (FSL) is a high performance remote computing cluster within the Aerosciences and Flight Mechanics Division at NASA JSC. As a computer system that can be connected to remotely via Secure Shell (SSH[41]) connection, the FSL made for a great environment for testing the feasiblity of Copernicus/LinCov communication via remote calls.

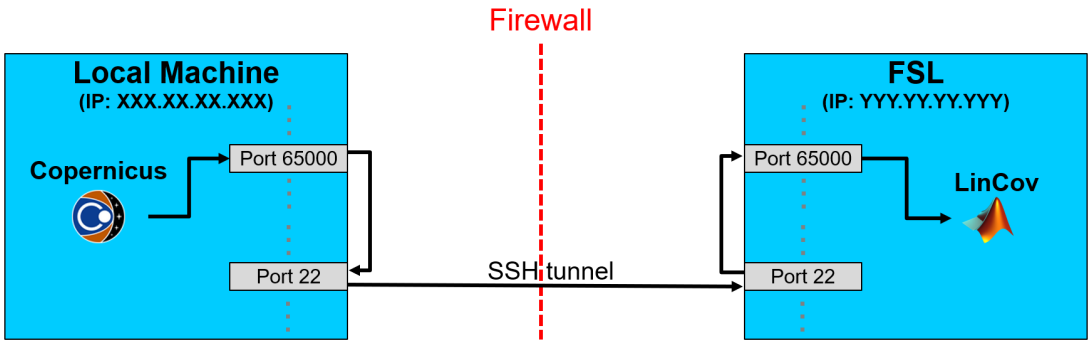


Figure 6. Connecting to LinCov (running remotely) while Copernicus (running locally) performs the optimization routine. An SSH tunnel is opened to authenticate user credentials and allow data exchange between the two softwares throughout the optimization process.

An SSH tunnel was chosen as the underlying method in which CopCov transports data across an encrypted SSH connection. Since the FSL is a secure computing environment protected by a firewall, SSH tunnels provide a means for users to connect to the remote environment and securely transfer data between local/remote machines. However, since Port 22 is reserved for SSH protocol, Copernicus and LinCov cannot send data packets directly to these ports. Instead, they transmit data to an unreserved high-level port (e.g. Port 65,000 in Figure 6) and use local port forwarding to transfer the data to Port 22 and subsequently across the SSH tunnel. Once LinCov receives a data packet from Copernicus and generates its associated output, that packet is then transmitted back through the SSH tunnel to Copernicus running on the user's local machine.

End-to-End Data Flow Pipeline

The end-to-end architecture consists of CopCov using a client/server model to establish a connection across machines. First, a user can manually startup the server on the remote machine (Block

4 in Figure 7), which will initiate MATLAB startup and LinCov initialization routines (Blocks 5 & 6), in addition to setting up the server model for interfacing with a local client. In the long term, this could also be a server that continuously runs in the background, based on available computing resources on the remote machine. This would simplify the startup process for an end user, as they would not need to worry about manually initiating any startup routines on the remote side.

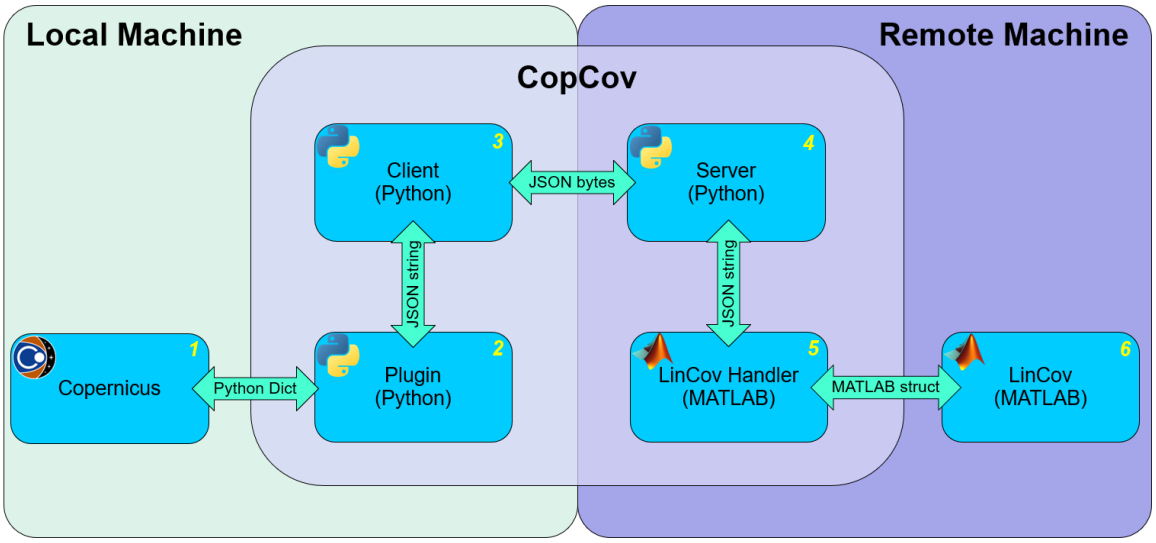


Figure 7. End-to-End architecture of Copernicus communication with LinCov through a remote connection. Data is transferred between Copernicus and LinCov on each iteration of the optimization process.

Once the remote server-side connection is up and running, a user can startup Copernicus (Block 1) and add the CopCov plugin to the desired ideck file (Block 2). CopCov will then automatically startup the client model (Block 3), which will ask the user to authenticate credentials, establish the SSH tunnel across machines (as in Figure 6), and finally connect to the server.

Blocks 1–4 have been demonstrated using the FSL, thus verifying the ability for Copernicus to transmit/receive data across a remote connection during individual iterations of the optimization process. The development architecture (Figure 5) was used to complete the analyses in the following sections while the full end-to-end architecture (Figure 7) continues to be developed and tested.

ROBUST TRAJECTORY OPTIMIZATION USING COPCOV – SETUP/OVERVIEW

This section highlights the nominal reference trajectory, various optimization problem formulations, and GN&C system algorithms and assumptions that were used to test and verify the robust trajectory optimization process using the CopCov tool against existing solution methods (i.e. Genetic Algorithm running in-the-loop with LinCov).

Nominal Reference Trajectory

Due to its simplicity and known optimal solution, the Hohmann transfer scenario was selected as the reference trajectory of choice to test and verify the CopCov tool.[42] The Hohmann transfer is an optimal two-impulse transfer between two circular coplanar orbits. The optimal solution consists of an impulsive burn (*M1* in Figure 8) to place the spacecraft on the transfer trajectory, followed by

a coast period with a 180-degree transfer angle, and finally a second impulsive maneuver (*M2*) to re-circularize the orbit.

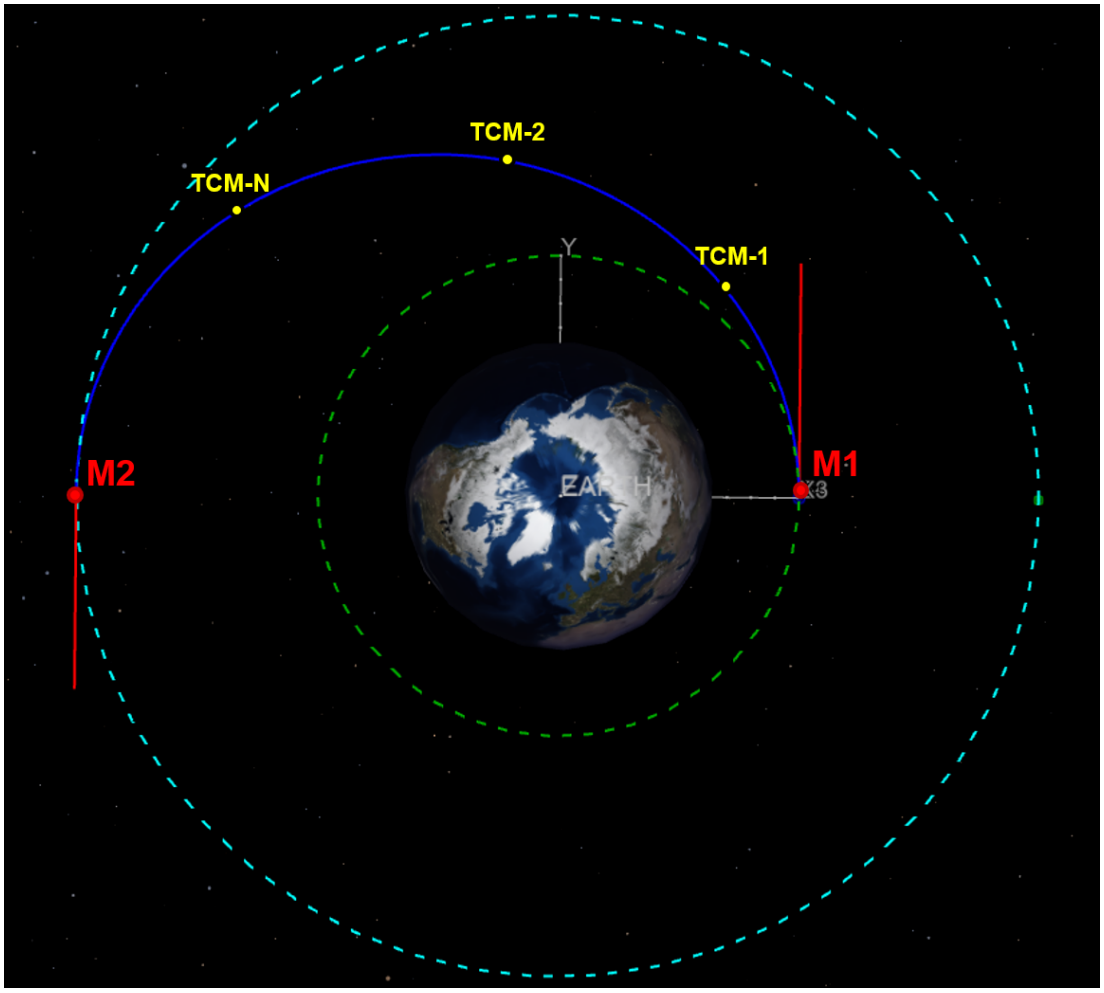


Figure 8. Reference Hohmann transfer trajectory with variable number of TCMs used for testing and verification of the CopCov tool

The Hohmann transfer is ΔV optimal from one circular orbit to another assuming an ideal nominal trajectory. However, depending on the selected targeting algorithms, navigation system, vehicle disturbance accelerations, and initial orbit insertion errors, the *optimal* transfer trajectory may look different when the integrated GN&C performance is considered. To illustrate, a spacecraft is chosen to transfer from a $10,000$ km circular orbit to a $20,000$ km circular orbit at a specific point (M2) and specific final time. If burn M1 occurs with a 180-deg transfer angle, as is the case with a Hohmann transfer, the corresponding transfer time would be ~ 2.54 hours. However, what is the optimal transfer time (or location) to perform the M1 burn in the presence of trajectory dispersions and navigation errors? Initially, no additional trajectory correction maneuvers are assumed. However, as Figures 8 and 16 illustrate, a variable number of TCMs can be considered due to the generality of the CopCov interface. From a nominal trajectory perspective, the number and placement of the TCMs is inconsequential since they are nominally zero. However, what is the optimal number and placement of these correction burns if they were added? The emphasis of the remaining sections of this paper

is to highlight the utility of CopCov in answering these rather simple yet relevant questions using robust trajectory design techniques.

Problem Formulations

To answer the conditions of optimality, the definition of optimal must first be quantified. Four different formulations are proposed and evaluated, each having different objective functions and constraints as summarized in Table 1. The initial problem formulation, Problem #0, attempts to only minimize the nominal ΔV, neglecting any resulting trajectory or ΔV dispersions. Under these conditions, the expected optimal solution should converge to the 180-degree Hohmann transfer profile. Problem #1 minimizes the total ΔV (i.e. nominal ΔV plus 3σ ΔV dispersions) while constraining the final position dispersions (at point *M2* in Figure 8) to be below some user-specified threshold (i.e. 3 km in Table 1), 3σ RSS. Problem #2 is the inverse, where the objective function minimizes the final position dispersions at M2 while constraining the total ΔV to be below a specified threshold of 3.0 km/s. Lastly, Problem #3 minimizes a weighted sum of final position dispersions and total ΔV where each parameter is weighted equally.

Table 1. Optimization Problem Formulations

Formulation	Objective Function (minimization)	Constraint
Problem #0	Nominal ΔV	—
Problem #1	Total ΔV (Nominal + 3σ Disp)	Final RSS 3σ Position Disp $<$ 3.0 km
Problem #2	Final 3σ Position Disp	Total ΔV (Nom + 3σ Disp) $<$ 3.0 km/s
Problem #3	Weighted Sum ($w_{\Delta v} = 0.5$, $w_r = 0.5$) of Final 3σ Position Disp & Total ΔV	—

Once the integrated GN&C system performance is considered as part of the optimization problem, the optimal solution will likely no longer be the traditional 180-deg transfer. With the problem formulations established, the answers to the previous optimality questions can now be determined, once details regarding the selected GN&C system are established.

GN&C Algorithms and Performance Specifications

Table 2. GN&C Performance Specifications

Uncertainty Parameter	3σ	Units
Initial Position Dispersions (per axis)	100	km
Initial Velocity Dispersions (per axis)	11	m/s
Initial Position Nav Error (per axis)	50	m
Initial Velocity Nav Error (per axis)	5	cm/s
Maneuver Execution Error (per axis)	75	mm/s
Disturbance Accelerations (per axis)	1	mm/s/$\sqrt{s}$
GPS Position Measurements (per axis, once every 30 sec)	200	m

For the notional Hohmann transfer scenario, it is assumed the vehicle's GN&C system utilizes a Lambert targeting algorithm[43] to compute the required ΔV to transfer to the desired final M2

position at the fixed final time. Although other targeting techniques exist for this type of circular orbit transfer, the selection of this particular algorithm (which has known limitations for multiples of 180-deg transfer angles[44]) emphasizes how the actual GN&C system can impact the trajectory design. This is particularly pronounced when uncertainty is included.

A summary of the GN&C performance parameters is provided in Table 2. The navigation system utilizes the Global Positioning System (GPS) with absolute position measurements processed every 30 seconds at an accuracy of 200 m (3σ, per axis). The disturbance acceleration acting on the spacecraft is 1 mm/s/$\sqrt{s}$ (3σ, per axis). The maneuver execution error is 75 mm/s (3σ, per axis). The initial navigation error in position is 50 m (3σ, per axis) with a velocity navigation error of 5 cm/s (3σ, per axis). The initial trajectory position dispersions are 100 km (3σ, per axis) and initial velocity dispersions are 11 m/s (3σ, per axis).

ROBUST TRAJECTORY OPTIMIZATION USING COPCOV

To motivate, demonstrate, and validate the concept of robust trajectory optimization using the CopCov tool, this section will solve the various optimization problem formulations (Problems #0-#3) for the Hohmann transfer scenario using four different techniques: 1) Mission Maps, 2) CopCov with a Polynomial-Fit LinCov Emulator, 3) CopCov with LinCov In-the-Loop, and 4) Genetic Algorithm with LinCov In-the-Loop. Detailed results are presented for the case with only major burns (i.e. no TCMs).

LinCov Pre-Generated Mission Map

A mission map provides a graphical depiction of the sensitivity of the optimization variable to the various optimization performance metrics such that an operator or engineer can draw immediate intuition to the problem and determine the optimal solution graphically. For simple problems like the one posed, this is feasible and provides valuable insight, intuition, and confirmation of numerical solutions derived subsequently. For the Hohmann transfer scenario, Figure 9 provides a pre-generated mission map where a scan of possible transfer times are evaluated as inputs along the x-axis. The corresponding performance metrics, provided on the y-axes, include the final 3σ position dispersions at M2 (blue line), the nominal ΔV (magenta line), the total ΔV (orange line), and the weighted sum of the final 3σ position dispersions and total ΔV (green line).

One quick trend that is observed from the mission map is that shorter transfer times reduce the final trajectory dispersions whereas longer transfer times typically reduce total ΔV. Another key observation is that the transfer time for the 180-deg transfer, indicated with the dashed vertical black line at 152.4 minutes (2.54 hours), corresponds with the minimum nominal ΔV as expected. However, due to limitations with the selected Lambert targeting algorithm, it also represents the transfer time that has the largest total ΔV when trajectory dispersions and navigation errors are considered.

Given these performance sensitivities from the mission map, optimal solutions for the various problem formulations can be identified using a manual approach with the graphical data, as illustrated in Figure 10. For example, the solution to Problem #0 can quickly be seen on the graph (when zoomed in closely) to be 152.359 minutes, which is the lowest point on the dashed magenta-line. Similarly, for Problem #1 the solution which minimizes the total ΔV subject to a constraint on final position dispersions is deduced from the plot to be 142.600 minutes. Notice, the transfer time that minimizes the total ΔV is actually 144.9 minutes, but the corresponding final 3σ trajectory disper-

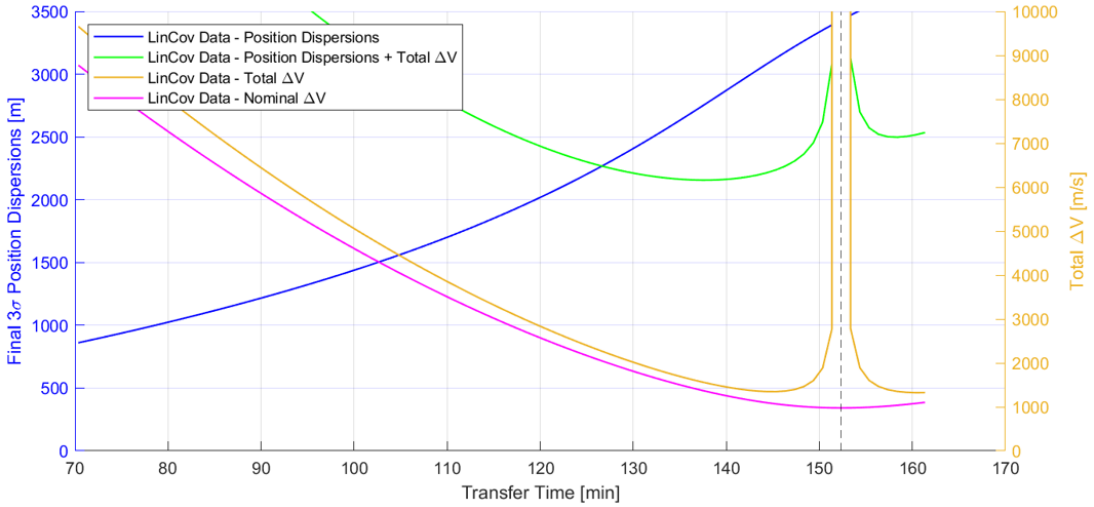


Figure 9. Mission Map for a Hohmann transfer problem with no Trajectory Correction Maneuvers

sions are greater than the 3 km constraint. For the Problem #2 objective function, minimizing the final 3σ position dispersions using less than 3 km/s occurs with a transfer time of 118.243 minutes based on the mission map plots. Lastly, for Problem #3, the transfer time that minimizes the weighted sum of the final position dispersions and the total ΔV is 137.358 minutes as derived from a simple visual inspection of the plotted data.

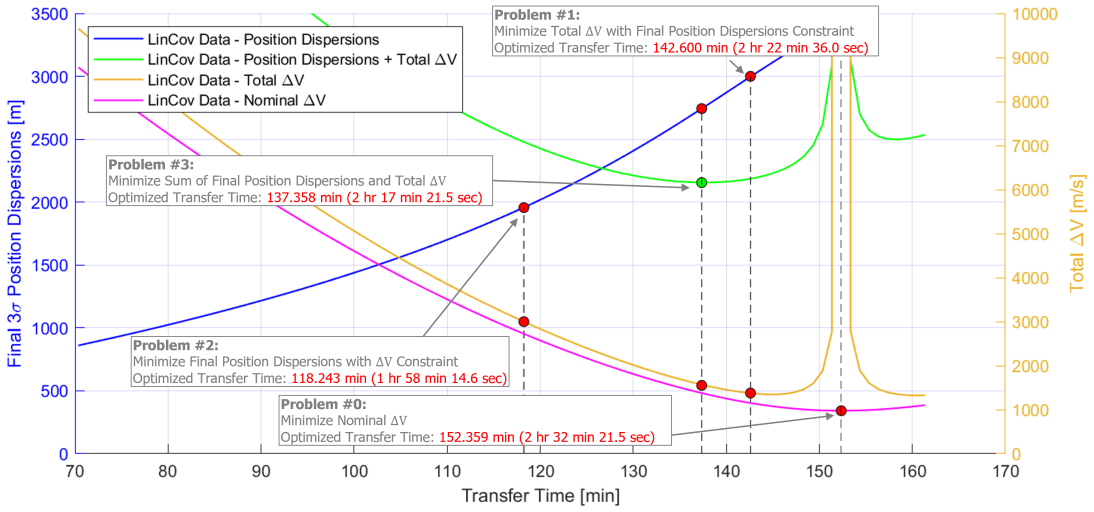
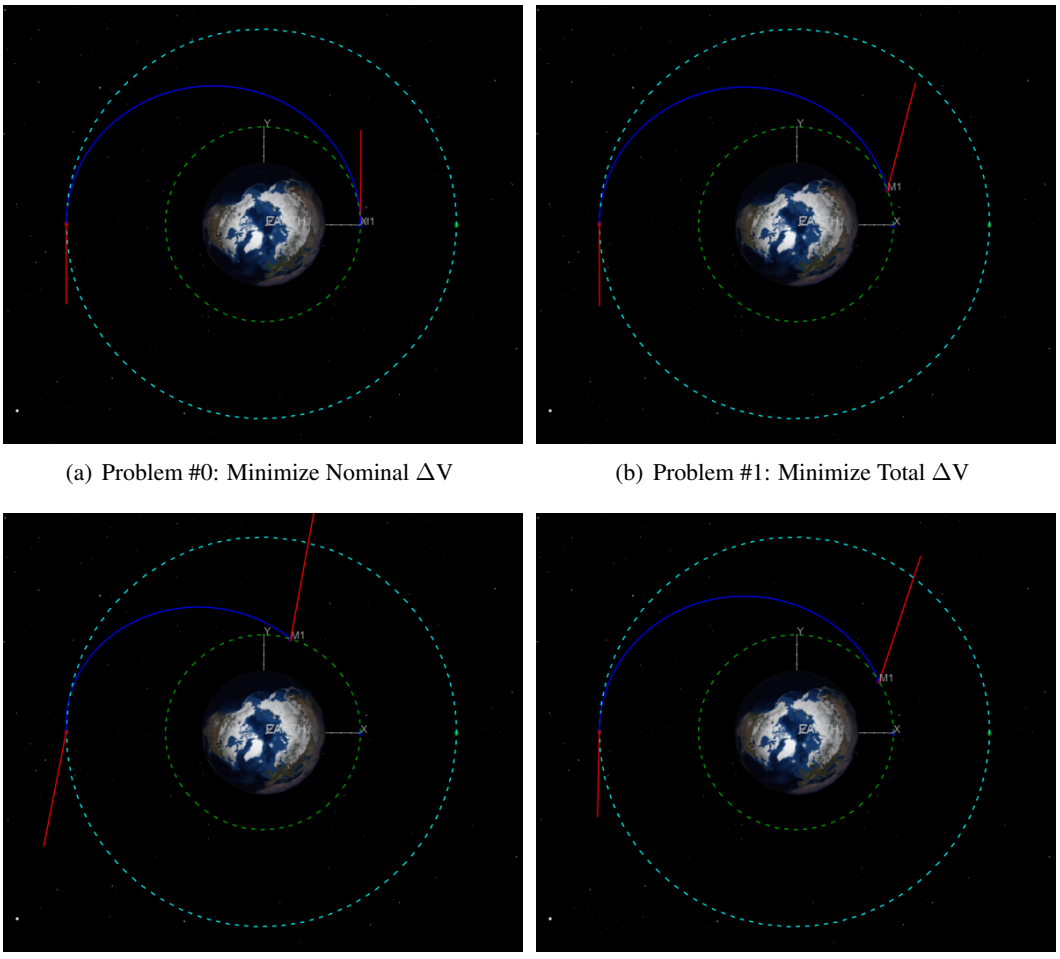


Figure 10. Mission Map Optimization Results

Geometrically, Figure 11 depicts the optimized trajectory and corresponding transfer burn placements for each different problem formulation. Figure 11(a) shows the burn location and resulting trajectory profile for Problem #0 that minimizes the nominal ΔV, Figure 11(b) summarizes the optimized profile for Problem #1 that minimizes the total ΔV given a final position dispersion constraint of 3 km, Figure 11(c) shows the Problem #2 optimized trajectory that reduces final position dispersions given a maximum total ΔV of 3 km/s, and Figure 11(d) depicts the optimized profile

for Problem #3 that reduces the weighted sum of the final position dispersions and the total ΔV. Although each solution is different, each is optimal for the given optimization criteria. Tables 3-5 provide comparisons of these visually-derived results against the numerical optimization algorithms described in the following subsections.



(a) Problem #0: Minimize Nominal ΔV

(b) Problem #1: Minimize Total ΔV

(c) Problem #2: Minimize Final Position Disp

(d) Problem #3: Minimize Total ΔV & Final Pos Disp

Figure 11. Optimized solutions for each problem formulation using Mission Maps

LinCov Emulator with CopCov

Given a priori information generated from LinCov (such as the previously derived mission maps), the LinCov Emulator can replicate comparable performance metric values using a polynomial fit of the resulting output dispersion data. This is demonstrated in Figure 12 where polynomials were fit to each curve in the mission map. These continuous and smooth polynomial functions can then be used by a gradient-based optimizer such as SNOPT to converge on an optimal solution. For this specific test case, a 5th degree polynomial was used to model the position dispersions and a 20th degree polynomial was found to sufficiently represent the total ΔV trends. Additionally, this process provides the capability to scale pre-generated mission maps to multivariate input spaces due to the flexibility of polynomial curves to be fit to input spaces of arbitrary dimension.

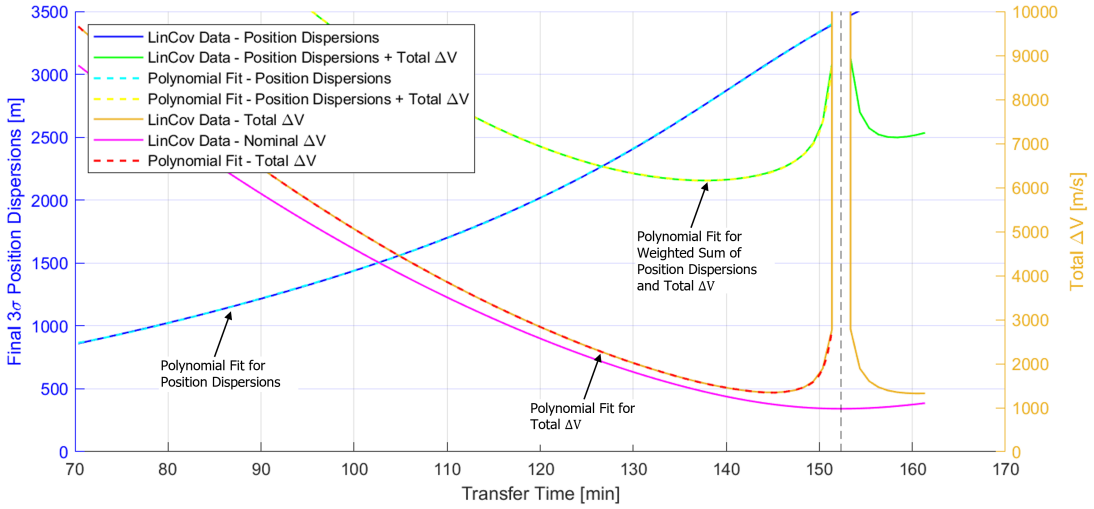


Figure 12. Polynomial fits of pre-generated LinCov mission map data across varying transfer times

With these polynomial curves embedded into the LinCov Emulator, Copernicus could quickly converge trajectories to realistic solutions at a fraction of the runtime as having the full LinCov tool in-the-loop. Figure 13 summarizes the converged solutions obtained using the polynomial-fitted LinCov Emulator. The results for the different objective functions in Problems #0-3 are very similar to those derived from the mission map and differ on the order of less then several seconds. Comparison tables of these results with the remaining cases are included in Tables 3-5.

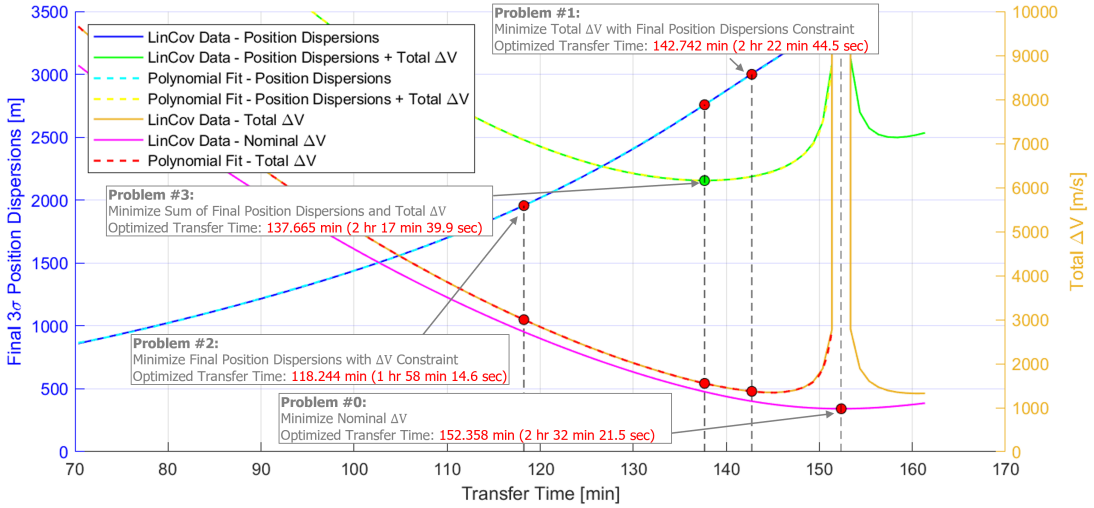


Figure 13. Optimized burn placement times using the LinCov Emulator with CopCov

As can be readily seen, with dispersion information being fed back to Copernicus using the objective functions in Problems #1-3, the optimized solution no longer converges to the standard 180-degree Hohmann transfer solution. Instead, in each case, it is suggested that the time of the first burn should be delayed such that the ΔV dispersions and final position dispersions are sufficiently reduced. This trend is consistent with the LinCov data in Figure 12 and the mission map results

discussed previously. As total ΔV dispersion values peak around a transfer time of $\sim$152 minutes, an adjusted transfer time is necessary to reduce the total potential propellant usage.

LinCov with CopCov

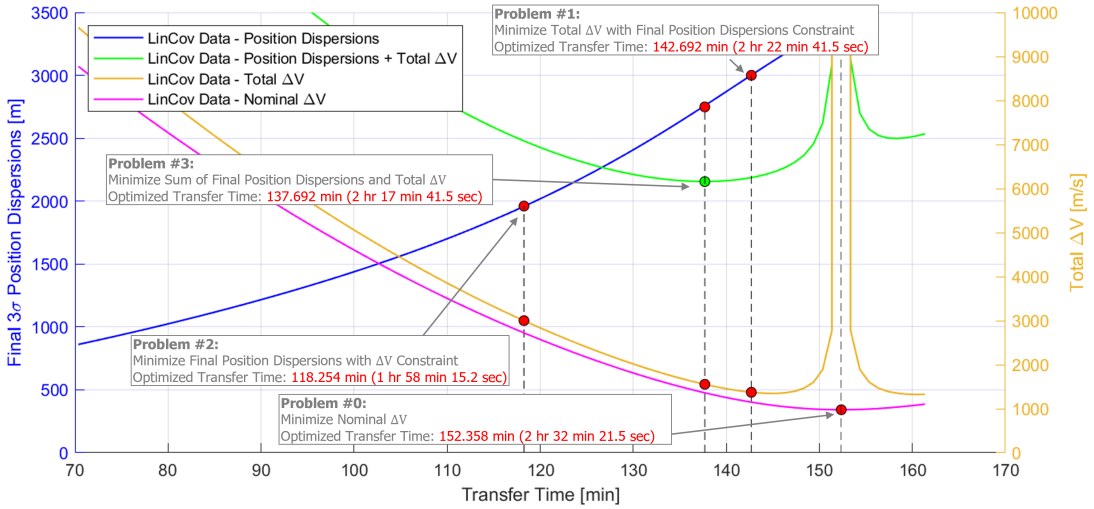


Figure 14. Optimized burn placement times using LinCov with CopCov

Given the insight and intuition from the mission maps and the LinCov Emulator, this section summarizes the results when CopCov connects Copernicus directly with the LinCov simulation tool to complete the optimization loop as outlined in Figure 3(b). This represents the first time this complete process has been demonstrated. The corresponding results are summarized in Figure 14, which are consistent with the solutions derived previously. The actual optimization time of Copernicus with LinCov in-the-loop was slower (increased from a few seconds with the Emulator to several minutes with the actual LinCov tool). Comparison tables of these results with the other cases are included in Tables 3-5.

LinCov with Genetic Algorithm

As mentioned previously and displayed in Figure 3(a), many of the robust trajectory optimization applications utilized a Genetic Algorithm (GA) in conjunction with the LinCov tool. This initial and near-term work flow has provided valuable insights into an assortment of space flight problems ranging from rendezvous and docking, lunar landing, aerocapture, and cis-lunar flight phases. To demonstrate that this earlier optimization approach also provides consistent results as Copernicus with LinCov in-the-loop, the GA solutions are provided in this section. This serves to confirm both processes *work*, each having their strengths and limitations. Using a population size of 250 candidates and 5 iterations, the solutions produced using a GA are provided in Figure 15.

It is important to note that the GA optimizer converged on the same solution as Copernicus to within several seconds of transfer time. The computational burden required for this strategy was noticeable. Several minutes were required to converge to a solution using multiple computational cores in parallel. Comparison data for these cases is included below in Tables 3-5.

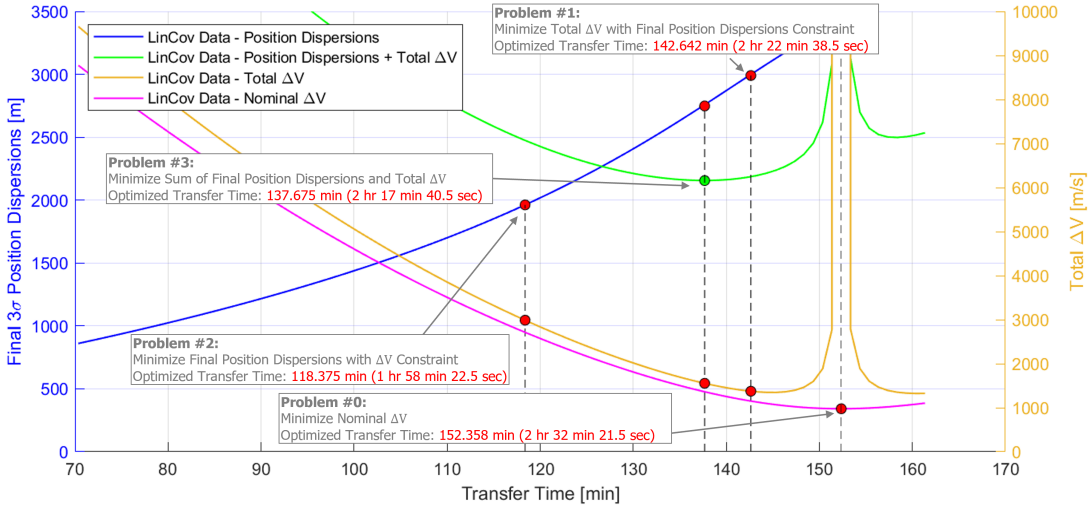


Figure 15. Optimized burn placement times using LinCov with GA optimizer

Comparison of Results

As can be seen from Tables 3-5, each of the different optimization methodologies converges to similar solutions, often to within only a few seconds difference in transfer time. This provides strong evidence that each of the newly proposed optimization techniques (i.e. Mission Maps, CopCov + Emulator, CopCov + LinCov) provides consistent results that align well with the heritage method of using a Genetic Algorithm + LinCov in-the-loop.

Table 3. Comparison of Results – Problem #1

Case	Transfer Time (H:M:S)	Total ΔV (m/s)	Position Disp (m)
LinCov Mission Map	2 hr 22 min 36.0 sec	1379.0	3000.0
LinCov Emulator + CopCov	2 hr 22 min 44.5 sec	1372.2	3000.0
LinCov + CopCov	2 hr 22 min 41.5 sec	1372.7	3000.0
LinCov + GA	2 hr 22 min 38.5 sec	1374.0	2990.4

Table 4. Comparison of Results – Problem #2

Case	Transfer Time (H:M:S)	Total ΔV (m/s)	Position Disp (m)
LinCov Mission Map	1 hr 58 min 14.6 sec	3000.0	1956.2
LinCov Emulator + CopCov	1 hr 58 min 14.6 sec	3000.0	1956.2
LinCov + CopCov	1 hr 58 min 15.2 sec	3000.0	1960.8
LinCov + GA	1 hr 58 min 22.5 sec	2987.6	1961.9

CONCLUSIONS AND FUTURE WORK

Robust trajectory optimization techniques were demonstrated on a Hohmann-like transfer trajectory using the novel CopCov tool. By providing a direct interface between Copernicus and LinCov,

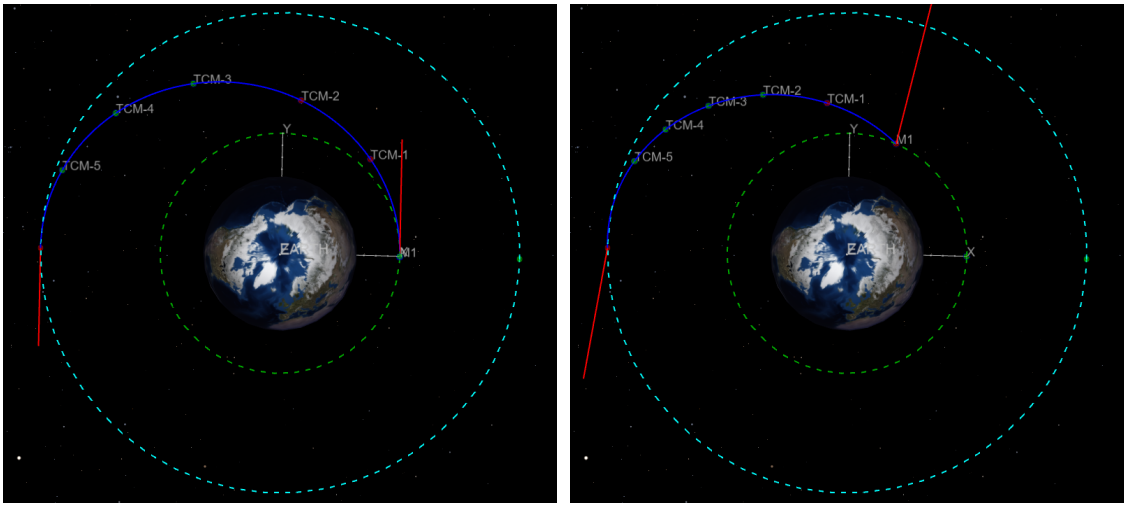
Table 5. Comparison of Results – Problem #3

Case	Transfer Time (H:M:S)	Total ΔV (m/s)	Position Disp (m)
LinCov Mission Map	2 hr 17 min 21.5 sec	1552.3	2743.2
LinCov Emulator + CopCov	2 hr 17 min 39.9 sec	1552.3	2758.2
LinCov + CopCov	2 hr 17 min 41.5 sec	1554.1	2748.8
LinCov + GA	2 hr 17 min 40.5 sec	1554.8	2748.6

dispersion information from LinCov could be fed back into the optimization routine and affect the optimized solution. A variety of different optimization problem formulations and optimization methodologies were proposed and tested. Results aligned closely for each case, indicating that each of the new methodologies (Mission Maps, CopCov + Emulator, CopCov + LinCov) has validity and potential for providing fast solutions to the robust trajectory optimization problem. Mission maps provide an intuitive graphical approach for easily visualizing the optimal solution assuming that LinCov data can be produced offline. Use of the Emulator with polynomial fits of LinCov data extended this capability to arbitrarily-sized input dimensions for multi-burn and multi-TCM cases. Lastly, CopCov operating in-the-loop with LinCov provided the full capability for Copernicus to optimize trajectories that are robust to trajectory dispersions and navigation errors from imperfect GN&C system performance.

Future work revolves around further maturing the CopCov software package to handle a wide variety of generalized trajectories. While impulsive burns were used in the Hohmann reference trajectories here, more work needs to be done in order for CopCov to properly handle finite burn maneuvers. Additionally, the end-to-end architecture is still yet to be fully tested and verified, which would prove that the two software packages could communicate across machines via remote connection. Lastly, if the maneuver partials (i.e. sensitivity of dispersions to maneuver timing) could be pre-calculated between Copernicus and CopCov, then LinCov could use this information directly rather than calling its own targeting algorithm to produce the partials. These features would provide additional capability that would allow these robust trajectory optimization techniques to be applied to real mission profiles and spacecraft GN&C systems.

APPENDIX



(a) Initial Guess (b) Converged Solution

Figure 16. Example 5-TCM Hohmann transfer case with LinCov Emulator running in-the-loop

To demonstrate the capability of Copernicus running end-to-end with a MATLAB-based tool in-the-loop, CopCov was first demonstrated using the quadratics-based LinCov Emulator on a 5-TCM case with Problem Formulation #3. Figure 16 illustrates both the initial guess trajectory and the optimized solution.

The initial guess was set to a 180-degree Hohmann transfer, with TCM initial guesses occuring at arbitrary times. Note that this is the solution that Copernicus would have converged to in the absence of the CopCov plugin, since the Hohmann transfer is optimal assuming no dispersions. Additionally, the TCMs would not have affected the objective function since they nominally have $\Delta V = 0$, thus they also would have converged to arbitrary locations.

However, in the optimization process, dispersion information from the LinCov Emulator is fed back to Copernicus which informs placement of both TCMs and major burns. The optimized solution suggests that with dispersions accounted for, a delayed TIG of the first major burn is optimal in terms of minimizing total ΔV and final position dispersions. Additionally, each of the TCMs is optimized to a specific location, each occuring exactly 1000 seconds apart as specified (arbitrarily) by the a_i and b_i coefficients of the quadratic functions within the LinCov Emulator. It is important to note that while these results are arbitrarily defined, they demonstrate the flexibility and capability of Copernicus to adjust its solution for a variety of generalized trajectories based on numerical feedback received through the CopCov interface.

REFERENCES

[1] N. Metropolis and S. Ulam, "The Monte Carlo method," *J. Am. Stat. Assoc.*, Vol. 44, 1949, p. 335.

[2] P. S. Maybeck, *Stochastic models, estimation, and control*, Vol. 1. New York: Academic Press, 1979.

[3] D. K. Geller, "Linear Covariance Techniques for Orbital Rendezvous Analysis and Autonomous Onboard Mission Planning," *Journal of Guidance, Control, and Dynamics*, Vol. 29, November-December 2006, pp. 1404–1414.

[4] N. Stastny and D. K. Geller, "Understanding Optimal Two-Burn Relative Trajectory Transfers with Maneuver Errors and Dispersion Requirements," *AAS 20-601, AIAA Astrodynamics Specialists Conference, Lake Tahoe, NV*, August 2020.

[5] K. Jin, D. K. Geller, and J. Luo, "Robust Trajectory Design for Rendezvous and Proximity Operations with Uncertainties," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 4, 2020, pp. 741–753.

[6] D. K. Geller, S. Shuster, D. Woffinden, and S. Bieniawski, "Robust Cislunar Trajectory Optimization via Midcourse Correction and Optical Navigation Scheduling," *44th Annual AAS Guidance, Navigaiton and Control Conference*, Breckenridge, CO, AAS 22-065, 4-9 February 2022 2022.

[7] D. Woffinden, S. Shuster, and S. Geller, David Kand Bieniawski, "Robust Trajectory Optimization and GN&C Performance Analysis For NRHO Rendezvous," *2022 AAS/AIAA Astrodynamics Specialist Conference*, Charlotte, North Carolina, 22-564, 7-11 August 2022 2022.

[8] G. Calkins, D. Woffinden, and Z. Putnam, "Robust Trajectory Optimization for Guided Powered Descent and Landing," *2022 AAS/AIAA Astrodynamics Specialist Conference*, Charlotte, NC, AAS 22-660, 7-11 August 2022 2022.

[9] J. Joshi, D. Woffinden, and Z. Putnam, "End-to-End Mars Aerocapture Analysis Using Linear Covariance Techniques and Robust Trajectory Optimization," *2022 AAS/AIAA Astrodynamics Specialist Conference*, Charlotte, NC, AAS 22-678, 7-11 August 2022 2022.

[10] D. Woffinden and B. Barton, "Optimized Trajectory Correction Burn Placement for NRHO Orbit Maintenance," *33rd AAS/AIAA Space Flight Mechanics Meeting*, Austin, TX, AAS 23-364, 15 - 19 January 2023.

[11] D. Geller, D. Woffinden, and S. Bieniawski, "Sensitivity of Optimal Midcourse Correction Scheduling for Robust Cislunar Trajectory Design," *45th Rocky Mountain AAS GN&C Conference*, Breckenridge, CO, AAS 23-061, 2 Feb - 8 Feb 2023.

[12] T. Goulet, D. Woffinden, N. Collins, and B. Andrews, "Robust Trajectory Design for Rendezvous in a Near Rectilinear Halo Orbit," *45th Rocky Mountain AAS GN&C Conference*, Breckenridge, CO, AAS 23-066, 2 Feb - 8 Feb 2023.

[13] D. Woffinden, R. Eckman, and S. Robinson, "Optimized Trajectory Correction Burn Placement for the NASA Artemis II Mission," Breckenridge, CO, AAS 23-062, 1 Feb - 6 Feb 2023.

[14] J. Williams, *Copernicus Version 5.2.0 User Guide*. NASA Johnson Space Center, Aeroscience and Flight Mechanics Division, Flight Mechanics and Trajectory Design Branch (EG5), 2022.

[15] A. S. Fraser, "Simulation of genetic systems by automatic digital computers II. effects of linkage on rates of advance under selection," *Australian Journal of Biological Sciences*, Vol. 10, No. 4, 1957, pp. 492–500.

[16] C. A. Ocampo, "An Architecture for a Generalized Trajectory Design and Optimization System," *Proceedings of the International Conference on Libration Points and Missions*, June 2002.

[17] C. A. Ocampo, J. S. Senent, and J. Williams, "Theoretical Foundation of Copernicus: A Unified System for Trajectory Design and Optimization," *4th International Conference on Astrodynamics Tools and Techniques*, May 2010.

[18] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," Vol. 47, Jan. 2005, pp. 99–131, 10.1137/S0036144504446096.

[19] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[20] J. Williams, A. H. Kamath, R. A. Eckman, G. L. Condon, R. Mathur, and D. C. Davis, "Copernicus 5.0: Latest Advances in JSC's Spacecraft Trajectory Optimization and Design System," *Advances in the Astronautical Sciences*, Vol. 171, August 2019.

[21] T. M. Inc., "MATLAB version: 9.11.0.2022996 (R2021b) Update 4," 2021.

[22] N. B. Stastny and D. K. Geller, "Autonomous Optical Navigation at Jupiter: A Linear Covariance Analysis," *Journal of Spacecraft and Rockets*, Vol. 45, Mar-Apr 2008, pp. 974–981.

[23] D. Geller, "Analysis of the relative attitude estimation and control problem for satellite inspection and orbital rendezvous," *The Journal of the Astronautical Sciences*, Vol. 55, June 2007, pp. 195–214.

[24] C. D'Souza and e. al., "Orion Rendezvous, Proximity Operations, and Docking Design and Analysis," *AIAA 2007-6683*, 2007.

[25] D. C. Woffinden and D. K. Geller, "Relative Angles-Only Navigation and Pose Estimation for Autonomous Orbital Rendezvous," *Journal of Guidance, Control, and Dynamics*, Vol. 30, September-October 2007, pp. 1455–1469.

[26] N. Stastny and e. al., "LinCov Analysis of an Automated Celstial Inertial Navigation Approach for GEO Satellites," *AIAA 2008-6756*, 2008.

[27] T. J. Moesser and D. K. Geller, "Guidance and Navigation Linear Covariance Analysis for Lunar Powered Descent," *AAS/AIAA Astrodynamics Specialist Conference*, Mackinac Island, Michigan, AAS 07-313, 19-23 August 2007.

[28] C. D'Souza and F. Clark, "Linear Covariance Analysis Techniques Applied to Orion Cislunar Operations," *AIAA and AAS Space Flight Mechanics Conference*, Galveston, TX, AAS 08-100, 27-31 January 2008.

[29] D. Geller and D. Christensen, "Linear Covariance Analysis for Powered Lunar Descent and Landing," *The Journal of Spacecraft and Rockets*, Vol. 46, Nov-Dec 2009, pp. 1231–1248.

[30] R. Zanetti, "Autonomous Midcourse Navigation for Lunar Return," *Journal of Spacecraft and Rockets*, Vol. 46, July-Aug 2009, pp. 865–873.

[31] M. B. Rose and D. K. Geller, "Linear Covariance Techniques for Powered Ascent," *AIAA 2010-8175*, 2010.

[32] P. Miotto, L. Breger, I. Mitchell, B. Keller, and B. Rishikof, "Designing and Validating Proximity Operations Rendezvous and Approach Trajectories for the Cygnus Mission," *AIAA 2010-8446*, 2010.

[33] D. Woffinden, L. Epstein, G. Stafford, T. Mosher, J. Curry, and Z. Krevor, "Dream Chaser On-Orbit Operations: Preliminary Trajectory Design and Analysis," *AIAA Guidance, Navigation, and Control (GNC) Conference*, Portand, OR, AIAA 2011-6654, 8-11 August 2011.

[34] R. Zanetti, D. C. Woffinden, and A. Sievers, "Multiple Event Triggers in Linear Covariance Analysis for Spacecraft Rendezvous," *Journal of Guidance, Control, and Dynamics*, Vol. 35, March-April 2012, pp. 353–366.

[35] R. S. Christensen and D. Geller, "Linear Covariance Techniques for Closed-Loop Guidance Navigation and Control System Design and Analysis.," *Journal of Aerospace Engineering*, Vol. 228, 2012, pp. 44–65.

[36] D. Christensen and D. Geller, "Terrain-Relative and Beacon-Relative Navigation for Lunar Powered Descent and Landing.," *The Journal of the Astronautical Sciences*, Vol. 58, Jan-Mar 2011, pp. 121–151.

[37] C. D'Souza and R. Zanetti, "Navigation and Dispersion Analysis of the First Orion Exploration Mission," *AAS/AIAA Astrodynamics Specialist Conference*, Vail, CO, AAS 15-768, 9-13 Aug 2015.

[38] K. Jin, D. Geller, and J. Luo, "Linear Covariance Techniques for Atmospheric Entry Guidance Analysis," *Journal of Spacecraft and Rockets*, Vol. 56, May–June 2019.

[39] D. Woffinden, S. Robinson, J. Williams, and Z. Putnam, "Linear Covariance Analysis Techniques to Generate Navigation and Sensor Requirements for the Safe and Precise Landing - Integrated Capabilities Evolution (SPLICE) Project," *AIAA Scitech 2019 Forum*, San Diego, CA, AIAA 2019-0662, 7-11 January 2019 2019.

[40] J. W. Williams, W. E. Brandenburg, D. C. Woffinden, and Z. R. Putnam, "Validation of Linear Covariance Techniques for Mars Entry, Descent, and Landing Guidance and Navigaiton Performance Analysis," *AIAA Scitech 2022 Forum*, 2022, 10.2514/6.2020-0597.

[41] T. Ylonen, "SSH - Secure Login Connections over the Internet," *Proceedings of the 6th USENIX Security Symposium*, 1996, pp. 37–42.

[42] W. Hohmann, *The Attainability of Heavenly Bodies*. Washington: NASA Technical Translation F-44, 1925.

[43] R. B. E.R. Lancaster, "A Unified Form of Lambert's Theorem," tech. rep., Goddard Space Flight Center, Greenbelt, Md., September 1969. NASA Technical Note, NASA TN D-5368.

[44] B. F. Thompson, "Enhancing Lambert Targeting Methods to Accommodate 180-Degree Orbital Transfers," *Journal of Guidance, Control, and Dynamics*, Vol. 34, November-December 2011, pp. 1925–1928.