NASA/TM–20230006507
NESC-RP-18-01313

Flight Mechanics Analysis Tools Interoperability and Component Sharing

Heather Koehler/NESC
Langley Research Center, Hampton, Virginia

Edwin Dove
Goddard Space Flight Center, Greenbelt, Maryland

April 2023

NASA/TM–20230006507
NESC-RP-18-01313

Flight Mechanics Analysis Tools Interoperability and Component Sharing

Heather Koehler/NESC
Langley Research Center, Hampton, Virginia

Edwin Dove
Goddard Space Flight Center, Greenbelt, Maryland

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

April 2023

Acknowledgements

The team wishes to acknowledge the support received from the different flight mechanics tool development teams and their supporting managers not represented in the Team Members table for assisting with tool updates and allowing the team members listed to participate in the assessment.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

NASA Engineering and Safety Center
Technical Assessment Report

Flight Mechanics Analysis Tools Interoperability and Component Sharing

TI-18-01313

NESC Lead – Heather Koehler

Technical Lead – Edwin Dove

April 11, 2023

Report Approval and Revision History

NOTE: This document was approved at the April 11, 2023, NRB.

Approved: TIMMY WILSON Digitally signed by TIMMY WILSON Date: 2023.04.24 17:24:06 -04'00'	
NESC Director	

Version	Description of Revision	Office of Primary Responsibility	Effective Date
1.0	Initial Release	Heather Koehler, NASA Technical Fellow for Flight Mechanics, MSFC	03/30/2023

Table of Contents

List of Figures

List of Tables

1.0 Notification and Authorization

The primary stakeholders for this assessment are the flight mechanics analysts, engineers, and tool developers, as well as the chief engineers and project managers for existing and future programs, projects, and/or tasks that involve mission analysis, navigation, trajectory optimization, and mission operations to any destination (e.g., Earth orbit, Sun, planet, comet, asteroid, halo orbit, Lagrange point, and distant retrograde orbits). Several key stakeholders include Sam Schreiber, Director of the Flight Dynamics Facility (FDF) at Goddard Space Flight Center (GSFC); Russell Carpenter, Space Science Missions Operations (SSMO) Deputy Project Manager/Technical Lead at GSFC; and Melissa McGuire, Deep Space Gateway (DSG) Trajectory Design Lead at Glenn Research Center.

2.0 Signatures

Submitted by: NESC Lead

Heather Koehler

Digitally signed by Heather Koehler
Date: 2023.04.24
09:28:41 -05'00'

Ms. Heather Koehler

Significant Contributors:

EDWIN DOVE

Digitally signed by EDWIN DOVE
Date: 2023.04.21
16:29:50 -04'00'

Mr. Edwin Dove

Signatories declare the findings, observations, and NESC recommendations compiled in the report are factually based on data extracted from program/project documents, contractor reports, and open literature, and/or generated from independently conducted tests, analyses, and inspections.

3.0 Team Members

Name	Discipline	Organization
Core Team		
Heather Koehler	NESC Lead	MSFC
Edwin Dove	Technical Lead	GSFC
Daniel Murri	Former NESC Lead	LaRC
Michael Shoemaker	Former Technical Lead	GSFC
Michael Mahoney	Former Technical Lead	GSFC
Steve Hughes	Former Technical Lead	GSFC
Darrel Conway	GMAT Developer	GSFC/Thinking Systems
John McGreevy	GMAT Developer	GSFC/Emergent Space Technologies LLC
Noble Hatten	GMAT Developer	GSFC
John McGreevy	GMAT Developer	GSFC/Emergent Space Technologies LLC
Alex Campbell	GMAT Developer	GSFC/Emergent Space Technologies LLC
Jairus Elarbee	GMAT Developer	GSFC/Emergent Space Technologies LLC
Steve Cooley	GMAT Developer	GSFC
Ravi Mathur	Copernicus Developer	JSC/Emergent Space Technologies LLC
Jerry Condon	Copernicus Developer	JSC
Jacob Williams	Copernicus Developer	NASA/Jacobs Technology
Anubhav Kamath	Copernicus Developer	NASA/Jacobs Technology
Randy Eckman	Copernicus Developer	JSC
Powtawche Valerino	MONTE/Copernicus Analyst	MSFC
Roby Wilson	MONTE Developer	JPL
Ted Drain	MONTE Developer	JPL
Scott Evans	MONTE Developer	JPL
Ricardo Restrepo	MONTE Analyst	JPL
Consultants		
Joel Parker	ODTBX Lead	GSFC
Business Management		
Linda Moore	Program Analyst	LaRC/MTSO
Assessment Support		
Missy J. Strickland	Project Coordinator	LaRC/AMA
Linda I. Burgess	Planning and Control Analyst	LaRC/AMA
Leanna S. Bullock	Technical Editor	LaRC/AMA

3.1 Acknowledgements

The team wishes to acknowledge the support received from the different flight mechanics tool development teams and their supporting managers not represented in the Team Members table for assisting with tool updates and allowing the team members listed to participate in the assessment.

4.0 Executive Summary

Several NASA centers have developed independent flight mechanics tools to meet the science needs of missions. This assessment sought to explore the ways to increase the interoperability of three specific tools: Copernicus from Johnson Space Center (JSC), the General Mission Analysis Tool (GMAT) from Goddard Spaceflight Flight Center (GSFC), and the Mission-Analysis Operations Navigation Toolkit Environment (MONTE) from the Jet Propulsion Laboratory (JPL). All of these tools are utilized in various spaceflight regimes and mission lifecycles (e.g., design, analysis, operations) to generate a variety of products (e.g., maneuver planning, orbit determination, error analysis, trade study, flight products). Each tool was built over the years with specific goals unique to each center, which were based on the science missions they supported, and naturally lead to variations in their capabilities. Before this assessment, these tools were not integrated and could not easily share data, models, or components. The goal of the assessment was to improve interoperability and component sharing of these flight mechanic tools to increase Agency efficiency and reduce cost.

To achieve the goal of the assessment it was important to analyze a framework that could establish a connection between the tools and allow users to leverage the strengths of each tool based on their needs. The underlying concept that arose was one of incorporating the tools into an enterprise System of Systems (SoS) that exposes components/functionality via interfaces like Application Programming Interfaces (APIs) and plugins. The common technology selected to enable the integration of data and components was the Python programming language. Efforts associated with this assessment integrated work to leverage Python in the flight mechanics tools as well as strengthen pre-existing development and expand it beyond its original scope.

The assessment was divided into three main high-level task development areas: 1) GMAT-MONTE interoperability, 2) GMAT-Copernicus interoperability, and 3) MONTE-Copernicus interoperability. Each of these tasks resulted in an improvement in the ability to interface/integrate with additional tools (e.g., Jupyter notebooks, Orbit Determination Toolbox (ODTBX), virtual reality (VR) headsets). These three tasks had different milestones that were adjusted because of the COVID-19 pandemic.

This assessment achieved its goal of improving interoperability between legacy flight mechanics tools at different NASA centers and helped pave a path forward for increased Agency-wide collaboration. The Copernicus three-dimensional (3D) graphics technology was successfully integrated into GMAT with advancements in that technology shared between the tools. It is recommended that the Python interfaces that the tools used to integrate data and components should be maintained, as well as the capability for the tools to leverage the Spacecraft Planet Instrument "C-matrix" Events (SPICE) toolkit to ingest and export trajectories. In addition, Copernicus, GMAT, and MONTE users should upgrade to the latest tool version to take advantage of the features that enabled the interoperability between the tools. Lastly, changes to the NASA software release process are recommended to enable broader distribution of source code and these tools which will enable wider use of the new features described in this report.

5.0 Assessment Plan

The first objective of this assessment was to develop an enterprise SoS that exposes, via APIs, unique capabilities available in GMAT, Copernicus, and MONTE for reuse among those tools and for use in other tools/systems/environments. The second objective was to complete the integration of the OpenFrames 3D graphics component, developed for Copernicus, into GMAT and deprecate GMAT's legacy 3D graphics component thereby eliminating some component duplication. The third objective was for the core flight mechanic tool development teams to document lessons learned for using NASA's flight dynamics systems in an SoS environment, and document policies and procedures to promote increased sharing and collaboration between centers.

The beneficial outcome of this effort to the Agency was to reduce duplication of effort via component reuse and sharing among NASA's enterprise tools, reduce cost, reduce risks, increase inter-center collaboration, and increase Agency technical capability.

Based on the objectives, the assessment work was divided into four technical tasks:

1. GMAT-MONTE interoperability development
2. GMAT-Copernicus interoperability development
3. MONTE-Copernicus interoperability development
4. Flight mechanic tool implementation lessons learned

Each task is described individually in later sections where a task overview and a summary of the activities performed are provided. The key deliverables are presented in Table 1 for all tasks and the SoS integration is visually represented in Figure 4 and supplemental deliverables are listed in Section 10. Software Usage Agreements (SUA) were in place at the onset of this assessment that allowed sharing of MONTE, GMAT, and Copernicus between the software development teams and tool users. Additionally, no special requirements were needed for facilities or tools to perform the work in this assessment.

The initial assessment plan was approved by the NASA Engineering and Safety Center (NESC) Review Board (NRB) on May 17, 2018, and consisted of a detailed schedule mapping out the work items necessary to produce the deliverables in Table 1 that are aligned with the aforementioned technical tasks. The original period of performance for the assessment plan was from the NRB approval date to the end of the third quarter of 2020, but due to the COVID-19 pandemic and assessment team staffing turnover the final assessment completion date was delayed until the end of the first quarter of 2023.

6.0 Problem Description and Background

6.1 Introduction

NASA centers utilize a variety of flight mechanics tools to support projects and analysis efforts that often are isolated to their own tool ecosystems. Despite the flight mechanics tool ecosystems being developed independently, there are many areas where unique attributes can be leveraged to allow for interoperability and component sharing between tools. This assessment explored the interoperability of NASA flight mechanics tools, including Copernicus, GMAT, and MONTE. At the onset of this assessment none of the tools were integrated or shared data, models, or components. This assessment involved the design, implementation, testing, and deployment of interfaces to support the interoperability of the NASA institutional flight mechanics tools and the potential for interoperability beyond those tools. Copernicus, GMAT, and MONTE tools are a subset of NASA's enterprise systems for navigation and mission design and support more than 30 projects for design, analysis, and operations. These systems support capabilities spanning the disciplines of orbit determination, trajectory optimization, error analysis, and operational flight product generation. Improving the interoperability and component sharing of these tools can lead to increased efficiency and reduce the cost of flight mechanics analyses Agencywide.

6.2 Background

6.2.1 MONTE

MONTE is a Python-based set of modules for trajectory design, navigation, and control that is suitable for all phases of non-atmospheric spaceflight [ref. 1]. MONTE is nominally designed for JPL's interplanetary missions but is suitable for most Earth-orbiting and lunar missions and is being used by other NASA centers. See Figure 1 for an example of a MONTE performance result. MONTE is delivered in two versions: the Project Edition and the Design Edition. The Project Edition is the full-featured version of MONTE utilized by JPL to perform trajectory design, orbit determination, and flight-path control for deep-space missions [ref. 1]. The Design Edition cannot perform statistical orbit determination. It can perform covariance analysis and trajectory design required for mission studies, but the processing of measurements required for orbit determination has been disabled [ref. 3]. The variation in the MONTE edition a user is allowed to obtain is due to export control restrictions.

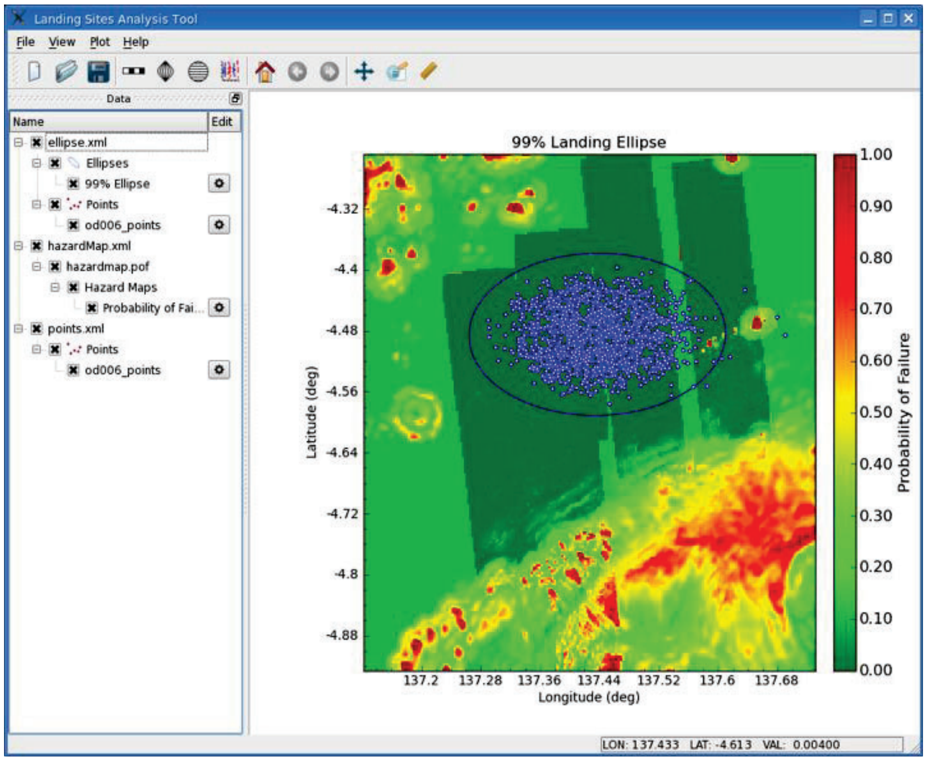


Figure 1. MONTE: Landing Site Statistical Hazard Avoidance

MONTE's design began in 1998 as a successor to JPL's Double Precision Trajectory and Orbit Determination Program (DPTRAJ/ODP) navigation tool that dated to the 1960s. In 2007, MONTE was utilized in its first operational mission and by 2012 powers all JPL navigation services [ref. 4]. MONTE is a primarily Python-based application that runs on Linux with its core code components utilizing C++. The source code for the C++ layer MONTE is not distributed [ref. 3]. MONTE is in active development and integrates new functionality based on stakeholder inputs.

6.1.1.1 Software Owner

MONTE is a product of the JPL Mission Design and Navigation Section with sponsorship from NASA's Multi-mission Ground System and Services/ Advanced Multi-Mission Operations System (MGSS/AMMOS) Program Office. MONTE is property of the California Institute of Technology [ref. 4].

6.1.1.2 Software Acquisition Process

MONTE can be licensed from JPL for government, academic, and commercial use. Users interested in obtaining the MONTE tool can formally request the software via JPL's Software Release website (https://download.jpl.nasa.gov/ops/request/request_introduction.cfm) [ref. 3].

6.2.2 GMAT

GMAT is designed to model, optimize, and estimate spacecraft trajectories in flight regimes ranging from low Earth orbit (LEO) to lunar applications, interplanetary trajectories, and other deep space missions [ref. 5]. GMAT is a feature-rich system containing high-fidelity space system models, optimization and targeting, built-in scripting and programming infrastructure, and customizable plots, reports and data products to enable flexible analysis and solutions for custom and unique applications. GMAT can be driven from a fully featured, interactive Graphical User Interface (GUI), from a custom script language, or via API [ref. 6]. See Figure 2 for an example of the GMAT tool.

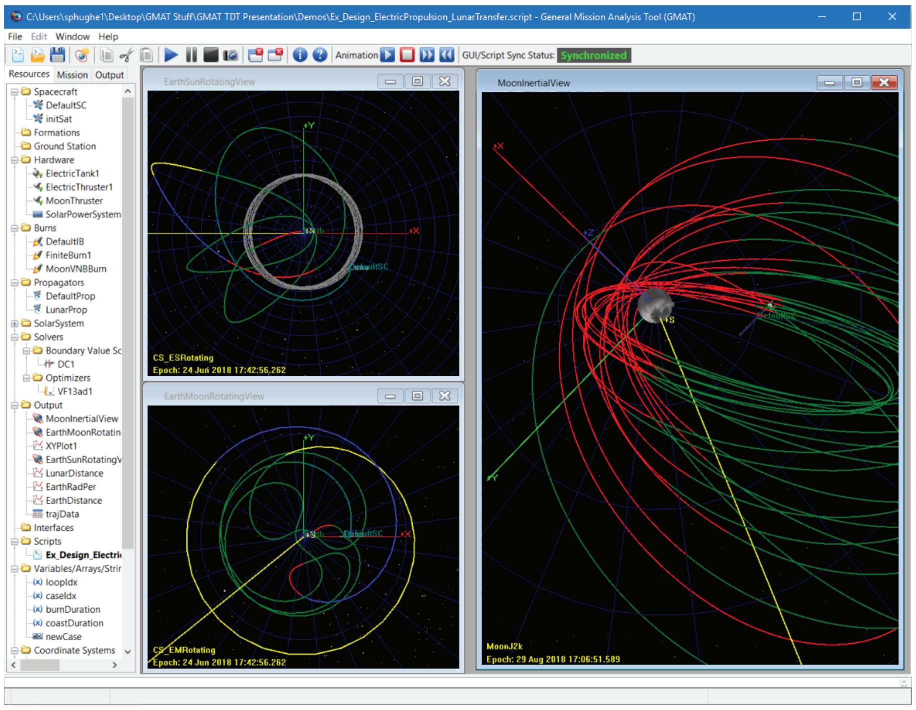


Figure 2. GMAT: Launch, through Lunar Flyby, to Lunar Capture using Solar Electric Propulsion (SEP)

Resources can be configured to meet the needs of specific applications and missions. GMAT contains an extensive set of available resources that can be separated into physical model resources and analysis model resources. Physical resources include spacecraft, thruster, tank, ground station, formation, impulsive burn, finite burn, planet, comet, asteroid, moon, barycenter, and libration point. Analysis model resources include differential corrector, propagator, optimizer, estimator, 3D graphics, x-y plot, report file, ephemeris file, user-defined variable, array, and string, coordinate system, custom subroutine, MATLAB function, and data [ref. 5].

GMAT design began in 2002 with the first public production release in 2013. GMAT's core codebase is written in C++ and is capable of interfacing with MATLAB, Python, and Java. GMAT is in active development and integrates new functionality based on stakeholder inputs.

6.2.2.1 Software Owner

The GSFC Navigation and Mission Design Branch and Ground Software Systems Branch work jointly to perform project management, design, implementation, and integration testing activities. External stakeholders contribute to design, implementation, testing, and documentation. The GMAT team uses a collaborative development model that enables innovation and actively involves the public and private sectors stakeholders.

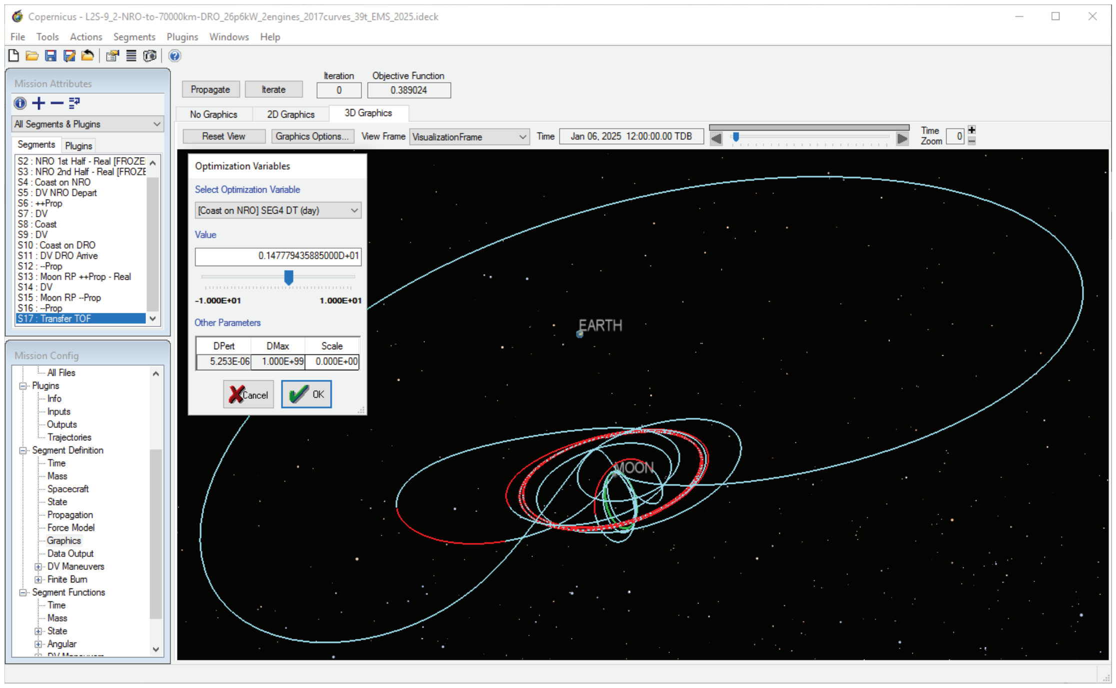
6.2.2.2 Software Acquisition Process

GMAT is an open-source tool that is licensed under Apache License 2.0 and is publicly available from https://sourceforge.net/projects/gmat/files/GMAT/.

6.2.3 Copernicus

Copernicus is a generalized spacecraft trajectory design and optimization system capable of solving a wide range of trajectory problems (e.g. planet or moon centered trajectories, libration point trajectories, planet-moon transfers and tours, and all types of interplanetary and asteroid/comet missions) [ref. 7]. Copernicus is capable of using multiple spacecraft and propulsion systems, utilizes integrated GUI and 3D graphics, includes flexible segment and plugin architecture, and allows for selectable mission fidelity from simple to complex. It is capable of supporting an extensive range of missions: impulsive/low high thrust, multi body, planet centered/inter planetary, and multi body transfers/trajectories [ref. 8]. See Figure 3 for an example of a Copernicus tool.

Copernicus started in 2001 as a Windows-only Fortran 77/90 and Compaq Visual Fortran application. Eventually it transitioned to being cross platform and is currently utilizing Intel Fortran 2019 and Python 3.8. Copernicus is in active development and continuously adding improvements and modernization. New features get added to the Fortran and Python components based on stakeholder feedback [ref. 8].



Copernicus - L2S-9_2-NRO-to-70000km-DRO_26p6kW_2engines_2017curves_39t_EMS_2025.ideck

S2 : NRO 1st Half - Real [FROZE
S3 : NRO 2nd Half - Real [FROZE
S4 : Coast on NRO
S5 : DV NRO Depart
S6 : ++Prop
S7 : DV
S8 : Coast
S9 : DV
S10 : Coast on DRO
S11 : DV DRO Arrive
S12 : --Prop
S13 : Moon RP ++Prop - Real
S14 : DV
S15 : Moon RP --Prop
S16 : --Prop
S17 : Transfer TOF

Iteration: 0
Objective Function: 0.389024
View Frame: VisualizationFrame
Time: Jan 06, 2025 12:00:00.00 TDB

Select Optimization Variable: [Coast on NRO] SEG4 DT (day)
Value: 0.147779435885000D+01
-1.000E+01 1.000E+01
Other Parameters

DPert	DMax	Scale
5.253E-06	1.000E+99	0.000E+00

Figure 3. Copernicus: Near Rectilinear Halo Orbit (NRHO) to Distant Retrograde Orbit (DRO) Transfer in the Earth Moon System

6.2.3.1 Software Owner

Copernicus started at the University of Texas at Austin in August 2001. In June 2002, a grant from the Johnson Space Center (JSC) was used to develop the first prototype which was completed in August 2004. In the interim, support was also received from NASA's In Space Propulsion Program and from the GSFC Flight Dynamics Vehicle Branch. The first operational version (v1.0) was completed in March 2006. The initial development team consisted of Dr. Cesar Ocampo and graduate students at the University of Texas at Austin Department of Aerospace Engineering and Engineering Mechanics. Since March 2007, primary development of Copernicus has been at the JSC Flight Mechanics and Trajectory Design Branch [ref. 7].

6.2.3.2 Software Acquisition Process

In accordance with NASA's obligations under mandating legislation, JSC makes Copernicus available free of charge to other NASA centers, government contractors, and universities, under the terms of a US government purpose license [ref. 7]. Individuals eligible can download the software from https://software.nasa.gov/software/MSC-26673-1 [ref. 9].

6.3 Proposed Solutions and Deliverables

Through the work of this assessment the MONTE, GMAT and Copernicus flight mechanics tools from the various NASA centers were enhanced to improve tool interoperability. Over the development history of each flight mechanics tool the science mission needs resulted in unique features being prominent. Copernicus' 3D graphics capabilities, MONTE's high fidelity dynamics modeling, and GMAT's detailed scripting environment tied to a GUI are a sliver of the strengths of each tool. This assessment sought to increase interoperability between the tools to allow the NASA community to better leverage the tool that meets their needs.

The deliverables associated with this assessment primarily come in the form of released software and are documented in Table 1. Each tool uses an existing SUA provided by the controlling center's Software Release Authority (SRA). Note that the core tools are routinely released under existing SUAs. and each team is experienced with managing releases of their system in alignment with NASA software release requirements. The SoS integration is visually represented in Figure 4**.**

Table 1. Assessment Deliverables

Deliverable	**Deliverable Content**	**Recipient/Availability**
MONTE Plugin	Software Application/Executable, Documentation, Supplemental Scripts	Government/Contractors, Commercial Use, Academia
Copernicus	Software Application/Executable, Documentation	Government/Contractors, Academia
GMAT API and plugin	Software Application/Executable, Documentation, Source Code, Supplemental Scripts	Open Source (Apache 2.0 License)
OpenFrames Shared Library	Software Application/Executable, Documentation, Source Code	Open Source (Apache 2.0 License)
NESC Assessment Report (NASA Technical Memorandum (TM))	Final Report/Lessons Learned	NASA projects

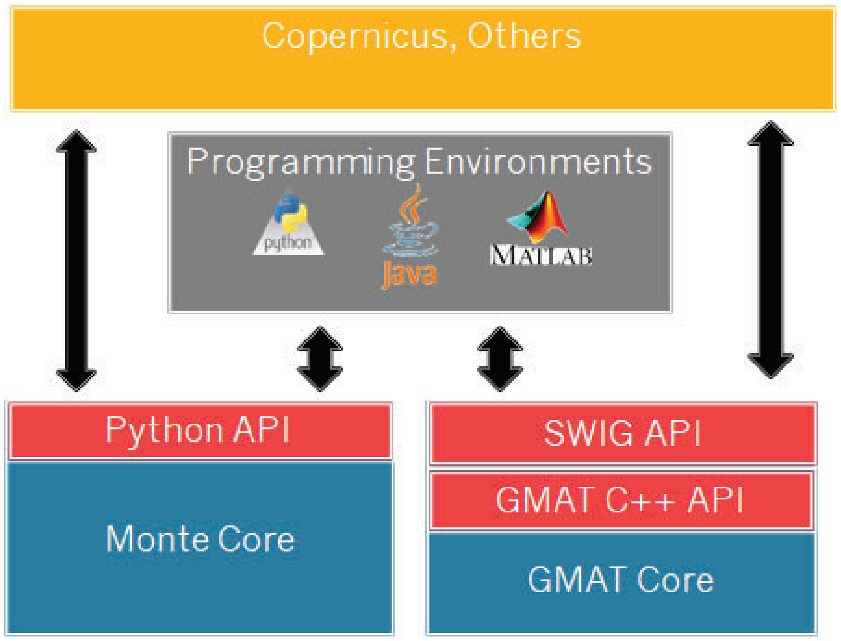


Figure 4. SoS Integration

7.0 Assessment Summary

7.1 GMAT-MONTE Interoperability

To achieve interoperability between GMAT and MONTE, as presented in Figure 4, the interfaces between the tools needed to be examined and expanded upon, when necessary. MONTE's dynamics modeling capabilities was known to be a strength and GMAT's scripting interface paired with its GUI was a strength. While GMAT could also do high-fidelity dynamics modeling for several orbit regimes and use cases there were some aspects where MONTE had more capabilities (e.g., more complicated structural modeling). Based on these strengths, the functionality desired between the tools resulted in the following tasks:

- Data integration – sharing key data between systems (e.g., ephemerides, covariance, maneuvers)
- Component integration – integrating core components (e.g., dynamics, propagation) between tools
- Shadow operations – showcasing the tools can work together with mission data

GMAT was originally designed to utilize a script-based method to capture inputs or a GUI that translates user inputs into a script. This method alone hinders external applications from fully interacting with the tool. External applications could alter script contents prior to runtime but could not interact during runtime with the exception of exporting output to a file. As GMAT's development progressed, additional interfaces were included that allowed function calls to MATLAB and Python during execution. The additional interface for function calls to robust programming languages were a good start but this was limited to GMAT mostly fulfilling the role of a driving application where a limited number of its activities were available to external applications during runtime. An API was required to realize the full potential of an external application being able to drive the various GMAT capabilities and supplement additional features from that external application. MONTE's interface since inception was utilizing a Python interface and was well suited to pass information GMAT needed to support the interoperability efforts of this assessment.

The shadow operations task would involve GMAT utilizing MONTE's dynamics modeling that provided increased customization of dynamics models, versus GMAT's dynamics modeling and performing shadow maneuvers for an in-operation mission. During the shadow operations, users would observe how the GMAT-MONTE results compared with the primary flight mechanics' operations tool.

7.1.1 Data & Component Integration

The GMAT API design focused on users being able to access GMAT components in a user-friendly manner from external applications [refs. 10 and 11]. This was achieved by utilizing the Simplified Wrapper and Interface Generator (SWIG). SWIG is an interface compiler that connects programs written in C and C++ with scripting languages (e.g., Perl, Python, Ruby, and Tcl). It works by taking the declarations found in C/C++ header files and using them to generate the wrapper code that scripting languages need to access the underlying C/C++ code [ref. 12].

The initial GMAT API development utilized the following milestones:

1. User-needs survey: Identify API components that are broadly applicable to users, while selecting two key components for exposure and use case development that demonstrate the interface functionality.

2. Initial API design and core code updates: Design API interfaces, helper functions, and style guide. Update the key GMAT architectural code interfaces to adhere to design and style.
3. API test system development: Build test system structures and components based on the existing GMAT test system that exercise the features exposed to users through the API.
4. Component exposure 1: Expose first set of key component(s) identified in the user-needs survey.
5. Component exposure 2: Expose second set of key component(s) identified in the user-needs survey.
6. Demonstration and documentation: Provide a demonstration of the ready to use API showcasing the added functionality, supporting documentation, and stakeholder recommendations for future updates.

Preliminary prototyping by the GMAT team used the SWIG library to expose core GMAT functionality to Python and Java, and through those interfaces MATLAB could also be used from/by GMAT. Feedback from users and stakeholders for these interfaces, along with the SWIG prototyping, led to the GMAT API component stack seen in Figure 5. GMAT's code base classes are exposed through this interface using language-specific wrappers that allow users to interact with the GMAT classes.

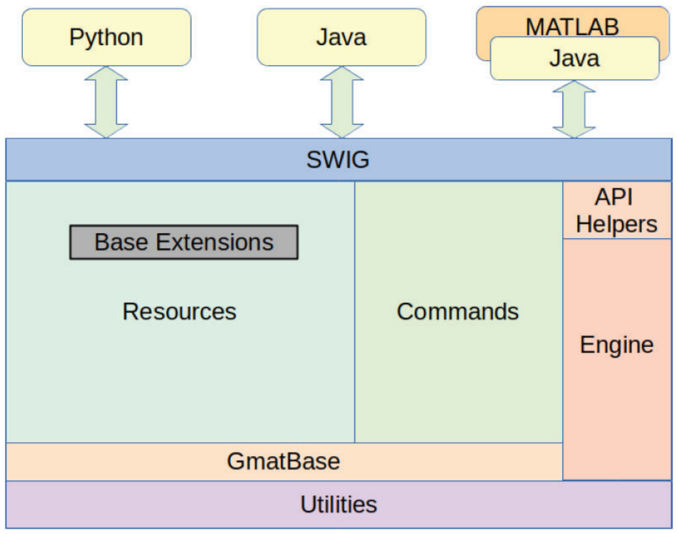


Figure 5. GMAT API Stack

Producing an API system comparable in quality to the other GMAT production-ready systems required a rigorous set of system tests to exercise the system features. Developers leveraged the preexistent MATLAB-based GMAT test system that runs nightly and performs thousands of tests for the application. Test system structures and components based on the core GMAT test system were built by the API development team to exercise the features exposed to users through the API. This required expanding the test system to incorporate MATLAB-Python and MATLAB-Java interfaces. Utilizing this approach of reusing pre-existing test architecture avoided the costly expense of developing separate test harnesses for the three target languages (i.e., MATLAB, Java, and Python). During the course of the assessment, and as additional functionality was added to the GMAT API, new tests were added, and regression testing was performed in the nightly builds with the other GMAT functionality.

The initial GMAT API development allow users to perform the following GMAT actions:

- Component access: state conversions, dynamics models, propagation, and measurement modeling
- Scripting controls: load scripts, change settings and values, and run scripts

Some examples of GMAT API's potential over the standard scripting functionality are:

- External application providing real-time tracking data to provide the user with real-time GMAT-generated measurement model output to display on a user-generated website that allow for advanced capabilities like real-time maneuver evaluation and tracking data anomaly detection.
- External application driver providing a variety of flight mechanic tool capabilities to a user in a larger system where GMAT is one of the many flight mechanic tools used.
- Performing a coverage analysis tool trade study by evaluating over 100 configurations.
- Performing an upper stage launch vehicle dispersions trade study utilizing a flexible assortment of hardware configurations.
- Jupyter notebook executing a GMAT script and displaying results live to the output.

A set of helper functions were woven into the design of the GMAT API to alleviate the need of a user to understand details of GMAT class structure implementation by encapsulating backend calls in a simplified manner. Most users prefer a less detailed level than a class by class or object by object interaction. The design of these helper functions came to fruition by the developers identifying two groups of users of the GMAT API:

1. Those familiar with GMAT and want to use the API to run GMAT scripts, making API calls to adapt their scripts along the way
2. Those that want to use capabilities provided by GMAT inside of models that they are running in a tool like MATLAB or Python, or in a compiled application written in a language like Java.

These API helpers are exposed through the SWIG interface layer for use by API users. The incorporation of class and object-level helper functions for classes that are identified as "API ready" help to address inconsistencies of user expected naming conventions to how the GMAT base systems are labeled. A beta version of the GMAT API was included in the GMAT 2020a public release. Feedback from this release of the GMAT API prompted additional helper functions.

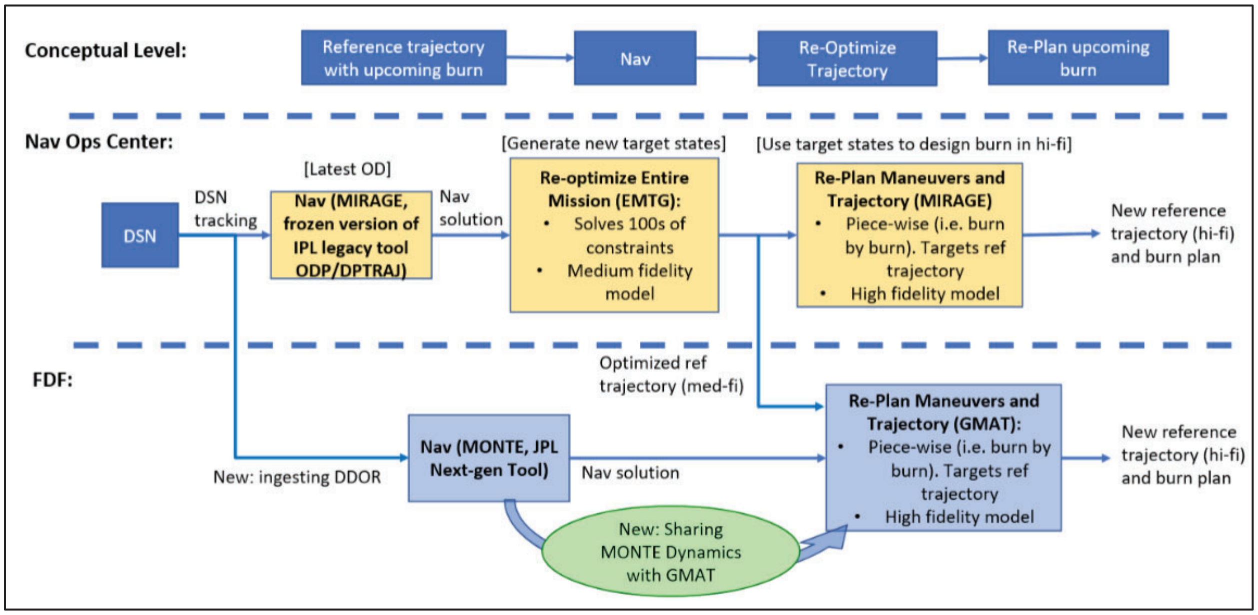
The GMAT 2020a release contained the GMAT API functionality associated with data sharing with external applications including ephemerides, covariance, and maneuvers. In March 2022, design work to allow GMAT to have the ability to ingest dynamics modeling to support the shadowing effort began. The GMAT 2022a release contained additional enhancements to the GMAT API and an external force model plugin to expose components that would enable the ingestion of external dynamics modeling during propagation [ref. 13]. The production version of the GMAT API without the beta designation was included in the GMAT 2022a public release.

To fulfill the task associated with integrating MONTE dynamics into GMAT for the shadow operations, an external-force model plugin was needed. The integration of MONTE dynamics into GMAT could not be accomplished through the API and would have required making changes to GMAT. Instead of making an interface for this specific to GMAT-MONTE, the more general approach of calling any force model in Python was used. The plugin route enabling dynamics sharing between MONTE and GMAT is built on a more general-purpose implementation, allowing propagation of any Python-based dynamics model in GMAT. MONTE dynamics sharing is a specific case of a general capability, implemented in GMAT, of dynamics sharing. MONTE's dynamics are exposed in Python, so they can be used via this generic channel. The GMAT 2022a release contains an alpha version of the GMAT external force model plugin that added the necessary hooks to fulfill the SoS integration, seen in Figure 4, to share key data and components between tools.

7.1.2 Mission Shadow Operations

The original in-operation shadowing effort to exercise the interoperability between GMAT and MONTE was planned for 2020, but the COVID-19 pandemic hampered development schedules and alternative plans were made. Part of the pivot made by the development teams was to switch the mission of the in-operation shadowing to the Lucy spacecraft that launched in October 2021 [ref. 14]. The agreement for the shadowing effort was that it would be a "do no harm" activity where none of the actions/outcomes would impact nominal spacecraft operations. The FDF analysts, Lucy operations staff, MONTE developers, and GMAT developers worked closely for this shadowing effort. The Deep Space Maneuvers (DSM) that target an Earth Gravity Assist (EGA) event was selected for the shadow operations. A GMAT 2022a equivalent build was used for the shadowing effort and MONTE version 152.

The Lucy team's primary means of producing a spacecraft burn plan utilizes Orbit Determination (OD) products generated from the Multiple Interferometric Ranging Analysis using GPS Ensemble (MIRAGE) navigation tool in combination with the Evolutionary Mission Trajectory Generator (EMTG) tool. Deep Space Network (DSN) tracking data gets fed into EMTG to generate an initial medium-fidelity burn plan then that plan is fed back into MIRAGE to generate the final high-fidelity burn plan that is used to perform maneuvers. The GMAT-MONTE shadowing effort was designed to utilize the same DSN tracking data fed through MIRAGE to generate OD products. The DSN tracking data used for primary operations is fed into MONTE then coupled with a reference trajectory ephemeris from the primary operations medium-fidelity EMTG solution to feed into GMAT maneuver scripts. These GMAT scripts utilize the tracking data to establish where the spacecraft is and reference trajectory to target the maneuver along with MONTE dynamics modeling through propagations to generate a high-fidelity burn plan. The Lucy primary burn plan approach with the GMAT-MONTE shadowing effort is illustrated in Figure 6.



Conceptual Level:
Reference trajectory with upcoming burn
Nav
Re-Optimize Trajectory
Re-Plan upcoming burn
Nav Ops Center:
DSN
DSN tracking
[Latest OD]
Nav (MIRAGE, frozen version of IPL legacy tool ODP/DPTRAJ)
Nav solution
[Generate new target states]
Re-optimize Entire Mission (EMTG): • Solves 100s of constraints • Medium fidelity model
[Use target states to design burn in hi-fi]
Re-Plan Maneuvers and Trajectory (MIRAGE) • Piece-wise (i.e. burn by burn). Targets ref trajectory • High fidelity model
New reference trajectory (hi-fi) and burn plan
FDF:
Optimized ref trajectory (med-fi)
New: ingesting DDOR
Nav (MONTE, JPL Next-gen Tool)
Nav solution
Re-Plan Maneuvers and Trajectory (GMAT): • Piece-wise (i.e. burn by burn). Targets ref trajectory • High fidelity model
New reference trajectory (hi-fi) and burn plan
New: Sharing MONTE Dynamics with GMAT

Figure 6. Lucy GMAT-MONTE Shadow Operations Plan

For the shadowing effort to work, the following activities needed to be complete:

1. Set up MONTE navigation
2. Integrate GMAT with MONTE
3. Generate scripting to retarget the maneuvers
4. Utilize operations data to perform shadow operations

The FDF analysts created custom scripting on their systems to be able to process the Lucy Delta-Differential One-Way Ranging (DDOR) data coming from DSN and transform it into a TRK-2-34 format that can be ingested by several navigation tools. The OD configuration for MONTE was set to be comparable with the fidelity of Lucy's primary navigation tool.

Pre-release builds of GMAT were utilized as necessary to test the capabilities of its external force model plugin dynamics model sharing with MONTE since the GMAT public release with the desired features would not happen until after the Lucy maneuver that targets the first EGA. The custom Lucy GMAT script points to a MONTE Python script to use high-fidelity modeling of Lucy dynamics, tuned to replicate the forces imparted on the spacecraft similar to the operations OD tool, at each propagation step. When the GMAT external-force model plugin was using MONTE's dynamics models, GMAT's internal dynamics models were disabled. This meant that GMAT scripting and visualizations were used while being taking advantage of MONTE's dynamics modeling strengths.

Three different Lucy EGA burn plan results were compared for this assessment:

1. Primary MIRAGE configuration utilizing six-plate spacecraft structural model
2. GMAT-MONTE configuration utilizing six-plate spacecraft structural model
3. GMAT configuration using a cannonball structural model of the spacecraft

The results of these configurations are summarized in Figure 7. DSM1 occurred on June 7, 2022, and targeted EGA1 took place on October 16, 2022. The scripts for both tools performed targeting for EGA1 and EGA2, which is scheduled to occur in December 2024. Figure 8 shows a position comparison of the trajectory, in radial, in-track, cross-track, and range representations, generated by the GMAT-MONTE shadow solution to the primary MIRAGE solution. Since the targeting sequence focused on a target of the EGA events, it was noticed both solutions were near zero difference at those points. All other times the dynamics were slightly different for each tool. This was unavoidable due to there being slight differences in the tools no matter how closely it was attempted to synchronize the settings between the tools. Utilizing a simplified spacecraft cannonball model can produce close enough solutions when high accuracy is not needed but for a highly sensitive problem more complicated structural models are needed. The dynamics modeling needed for the DSMs are sensitive enough to necessitate the higher fidelity structural modeling for consistent results. For the first DSM, the GMAT cannonball configuration diverged from the MIRAGE solution, but the GMAT-MONTE configuration was within about a 1% difference of the MIRAGE solution. These GMAT-MONTE results successfully showcased interoperability between the tools during shadow operations is possible. During the shadowing efforts there were several lessons learned that are presented in Section 11.

DSM1	Magnitude (m/s)	Magnitude Percent Difference	Angle Difference (deg)
MIRAGE	4.2020		
GMAT-MONTE	4.2372	0.837	0.0304
GMAT	5.0768	20.82	0.0731

DSM2	Magnitude (m/s)	Magnitude Percent Difference	Angle Difference (deg)
MIRAGE	912.102		
GMAT-MONTE	912.955	0.094	0.0221
GMAT	912.202	0.011	0.0008

Figure 7. Lucy GMAT-MONTE EGA Shadowing Results

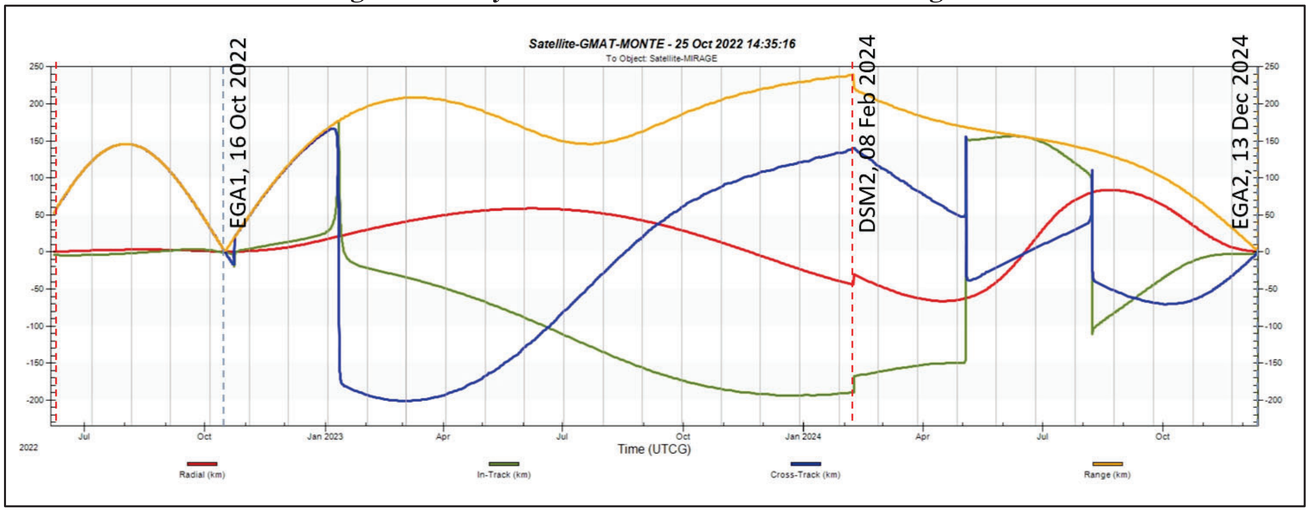


Figure 8. Lucy GMAT-MONTE EGA Trajectory Compared to Reference MIRAGE Trajectory

7.2 GMAT-Copernicus Interoperability

To achieve interoperability between GMAT and Copernicus, as presented in Figure 4, the interfaces between the tools required examination and expansion, when necessary. Copernicus' 3D graphics capabilities were mature and a strength and GMAT's scripting paired with its GUI and high-fidelity modeling were a strength. Based on these strengths, the functionality desired between the tools resulted in the task to utilize a common 3D graphics engine.

Prior to this assessment, GMAT and Copernicus used independent 3D graphics interfaces for their respective tool's code base. GMAT's original 3D graphics interface (OrbitView) was not optimized to function in tandem with the high-fidelity dynamics calculations it was performing and any attempts to upgrade using the existent codebase would have been a prohibitive effort. Figure 9 shows the original OrbitView 3D graphics interface. Copernicus' 3D graphic interface utilized OpenFrames and OpenSceneGraph for over a decade. OpenFrames is an API that provides the ability to add interactive 3D graphics to any application and OpenSceneGraph is a toolkit that provides cross-platform scene graph management, which is a concept that groups objects with similar characteristics together. OpenSceneGraph is used by application developers in fields (e.g., as visual simulation, games, VR, scientific visualization, and modelling). These graphics components greatly reduce the burden on the developers to be as familiar with the intricacies of writing complex 3D graphics code. OpenFrames and OpenSceneGraph were developed over several Small Business Innovation Research (SBIR) initiatives and released to the public under the Apache 2.0 Open Source License [refs. 15, 16, 17]. OpenFrames and OpenSceneGraph being open source lent themselves to be great candidates to replace GMAT's OrbitView.

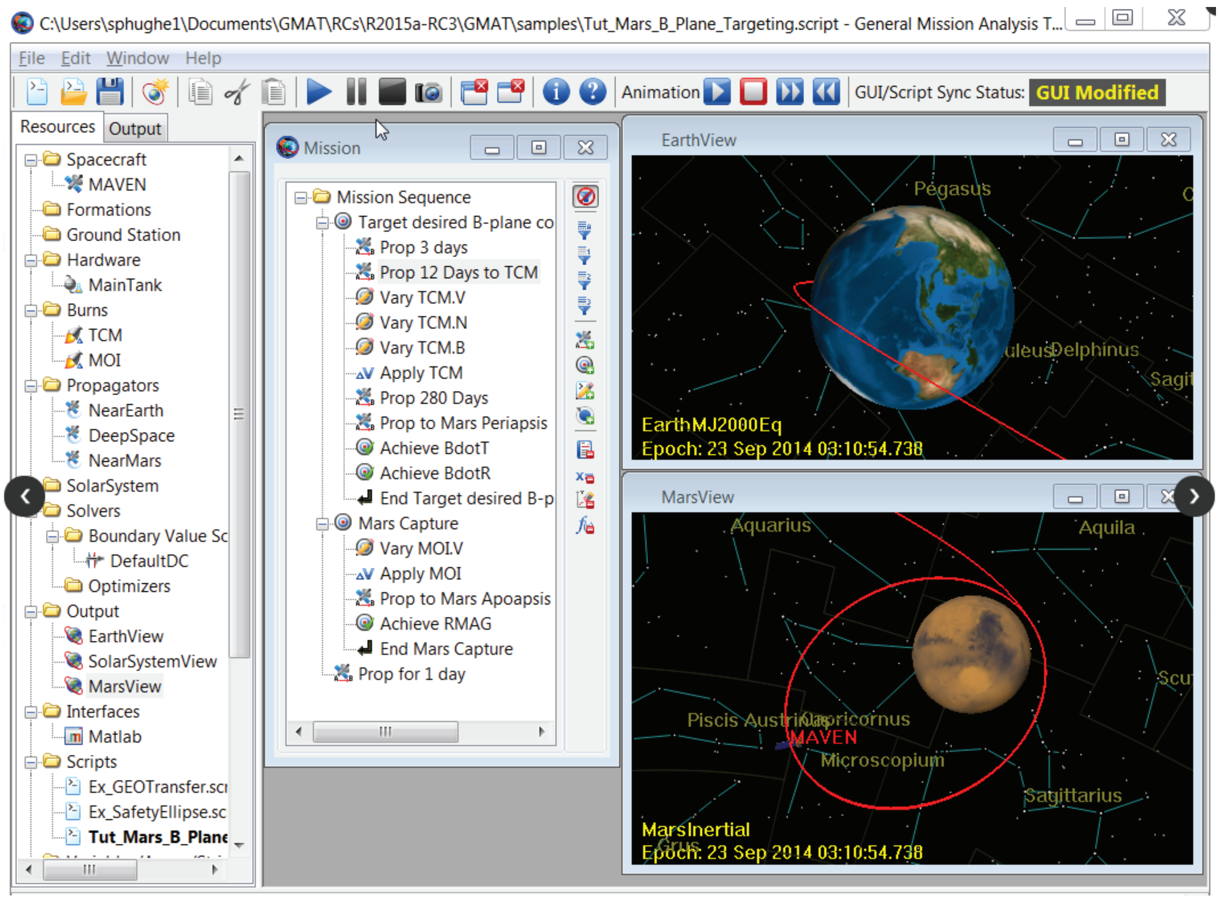


Figure 9. Previous GMAT 3D Graphics Interface (OrbitView)

GMAT being able to leverage the same 3D graphics engine as Copernicus would allow GMAT to utilize existing robust graphics capabilities and extend the OpenFrames flight mechanics graphics API for trajectory visualization. Copernicus would benefit from improvements invested in the common graphics components by receiving the same updates that provide new functionality. Bug fixes made to the common 3D graphics interface would benefit both tools.

Prior to this assessment, integration of the GMAT specific OpenFrames/OpenSceneGraph graphics components was started under a SBIR effort and rated at a Technology Readiness Level (TRL) of 6, which means some part but not all of the system was utilized in an operational environment. This assessment allowed for the final testing and documentation to bring the GMAT OpenFrames/OpenSceneGraph components to a TRL 9, which means the entire system is proven through operations. This TRL 9 achievement resulted in the deprecation of GMAT's legacy 3D graphics (OrbitView) component by November 30, 2019. The graphics updates were available to the public in the GMAT 2020a release in April 2020 [ref. 18].

Copernicus and GMAT are utilizing common graphics components based on OpenFrames interface and the OpenSceneGraph library. Figure 10 depicts the shared graphics components between Copernicus and GMAT. Even though 3D graphics components are shared between the tools, each tool can utilize these components in different ways. Figure 11 shows Copernicus' 3D graphics implementation with OpenFrames, and Figure 12 shows GMAT's 3D graphics implementation with OpenFrames. Some examples of shared capabilities the two tools could leverage are the hooks into the OpenFrames VR framework, lighting source features to showcase day/night cycles of planets and eclipse shadowing, cross-platform compatibility, and multi-core support.

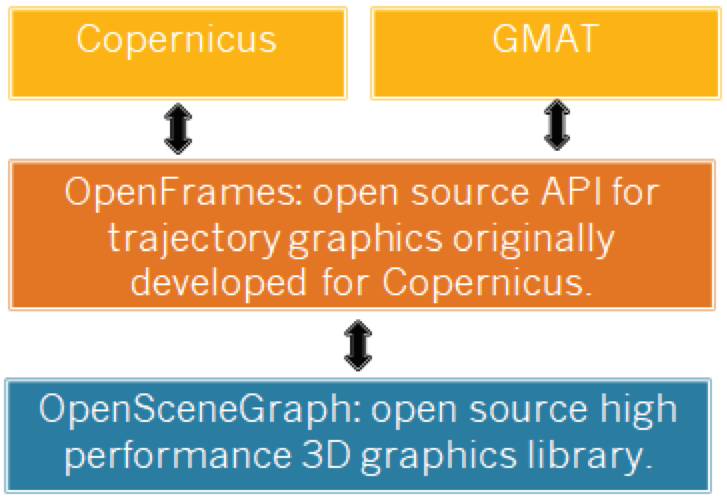
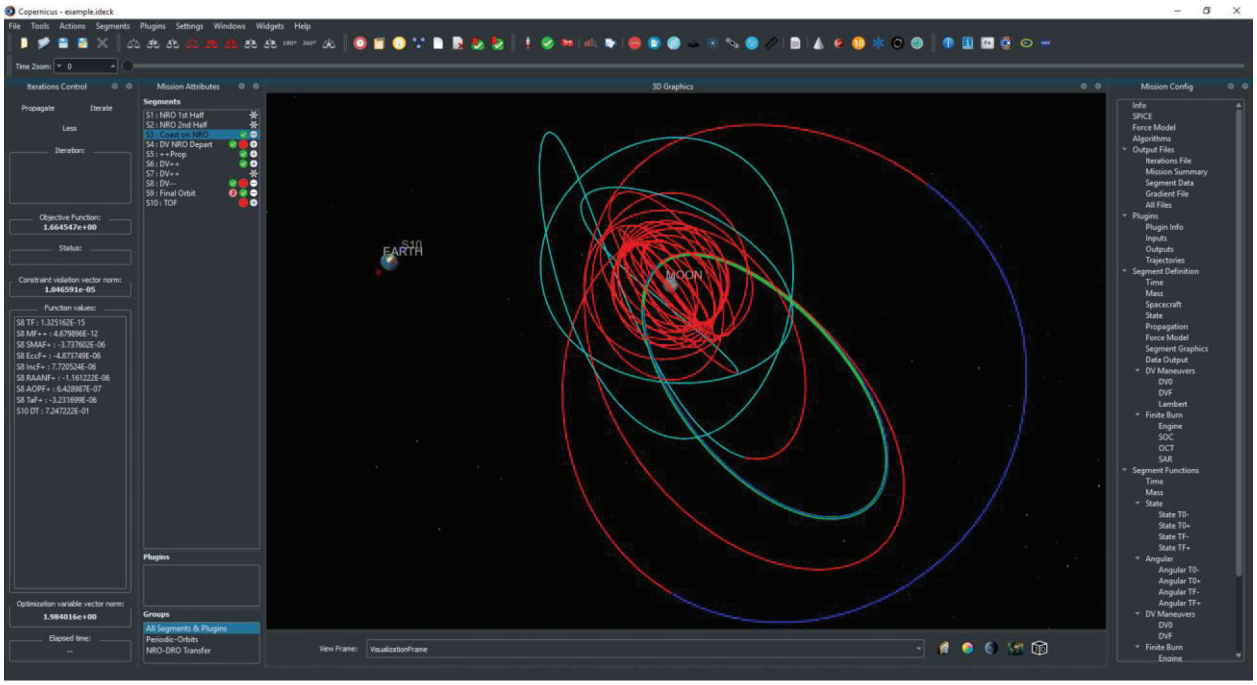

Copernicus
GMAT
OpenFrames: open source API for trajectory graphics originally developed for Copernicus.
OpenSceneGraph: open source high performance 3D graphics library.

Figure 10. GMAT <=> Copernicus Shared Graphics Components



Iteration:

Objective Function: 1.664547e+00

Status:

Constraint violation vector norm: 1.046591e-05

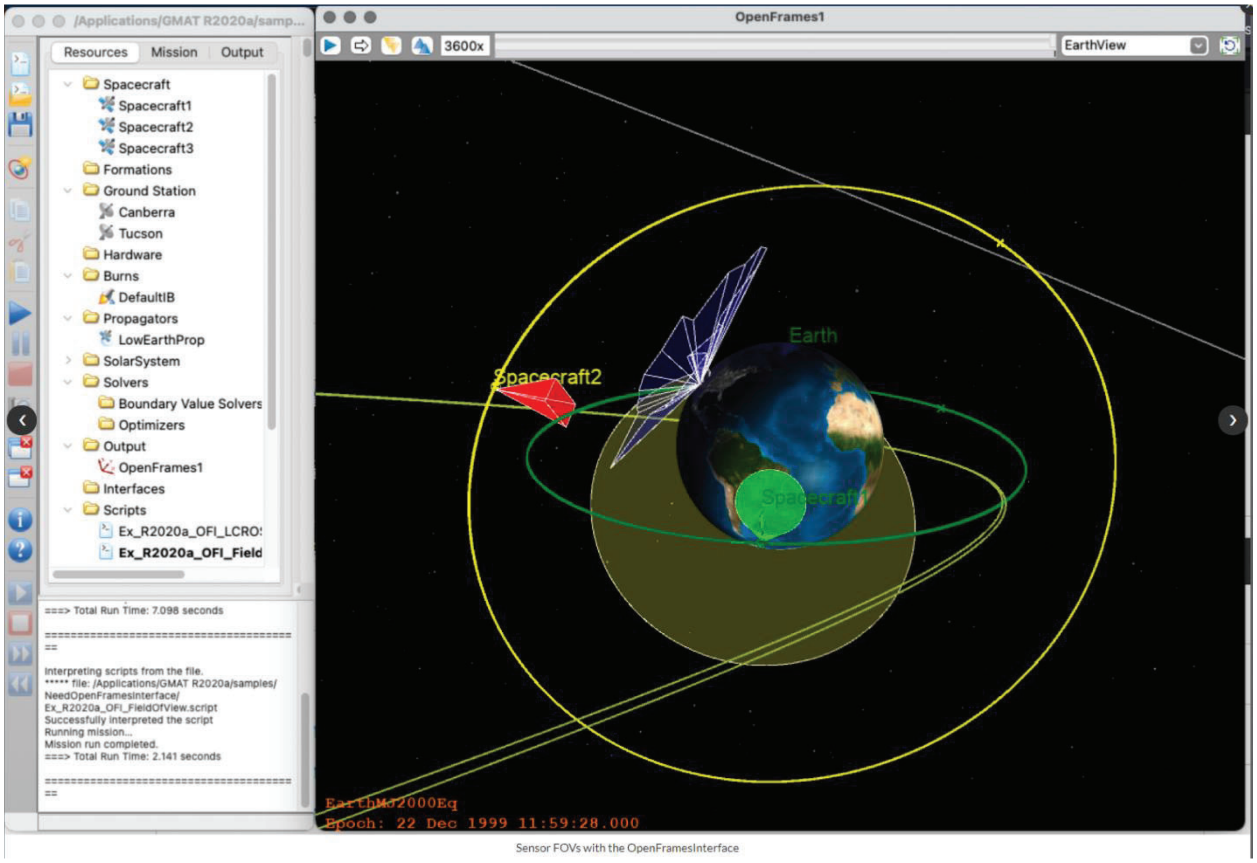
Function values:

```
S8 TF : 1.325162E-15
S8 MF++ : 4.679896E-12
S8 SMAF+ : -3.737602E-06
S8 EccF+ : -4.873749E-06
S8 IncF+ : 7.720524E-06
S8 RAANF+ : -1.161222E-06
S8 AOPF+ : 6.428987E-07
S8 TaF+ : -3.231699E-06
S10 DT : 7.247222E-01
```

Optimization variable vector norm: 1.984016e+00

Elapsed time: --

Figure 11. Copernicus OpenFrames 3D Graphics Implementation



```
===> Total Run Time: 7.098 seconds

========================================
==

Interpreting scripts from the file.
***** file: /Applications/GMAT R2020a/samples/
NeedOpenFramesInterface/
Ex_R2020a_OFI_FieldOfView.script
Successfully interpreted the script
Running mission...
Mission run completed.
===> Total Run Time: 2.141 seconds

========================================
==
```

Figure 12. GMAT OpenFrames 3D Graphics Implementation

7.3 MONTE-Copernicus Interoperability

To achieve interoperability between MONTE and Copernicus, as presented in Figure 4, the interfaces between the tools needed examination and expansion. Copernicus' ability to display spacecraft trajectories in 3D graphics was a strength and MONTE's high-fidelity dynamics modeling to obtain accurate solutions for complicated trajectories was a strength. During the design phase of this MONTE-Copernicus effort, the functionality desired between the tools to leverage their strength resulted in the following tasks:

- Interface development and documentation
- Tool updates for interface compliance
- Use cases leveraging interface
- Publication highlighting interface

Prior to this assessment, Copernicus versions before 4.5 used a Fortran GUI toolkit that was highly coupled with the Fortran core code, but as the capabilities of Copernicus grew and a desire to interface with more external applications arose a different paradigm was needed. The paradigm shift changes that were relevant to tool interoperability in this assessment were:

- the separation of the Copernicus GUI from the core code into a shared library
- implementing a Python GUI for easier access and usability
- utilizing a Python scripting interface that leverages Fortran callbacks
- leveraging a Fortran to C++ interface that can use a shared 3D graphics capability [refs. 8 and 19].

These changes are illustrated in Figure 13. Initially when the Copernicus Python interface was created (CopPy) that interface could only alter existing items in the input deck (ideck) container file Copernicus uses to store all the information for a particular mission run. CopPy could not be used to alter missions (or individual trajectory segments) from scratch, or add new elements (e.g., finite burns) that were not in the original file. Some workarounds existed for this interface but to satisfy the interoperability goals of this assessment it was necessary to create RoboCopPy, which is an object-oriented Python approach to the problem with a mapping of every option and field available in each GUI. RoboCopPy allows construction of complete idecks by adding class instances to various collections, allows modification of idecks, and satisfies the interoperability goal of this assessment [ref. 19].

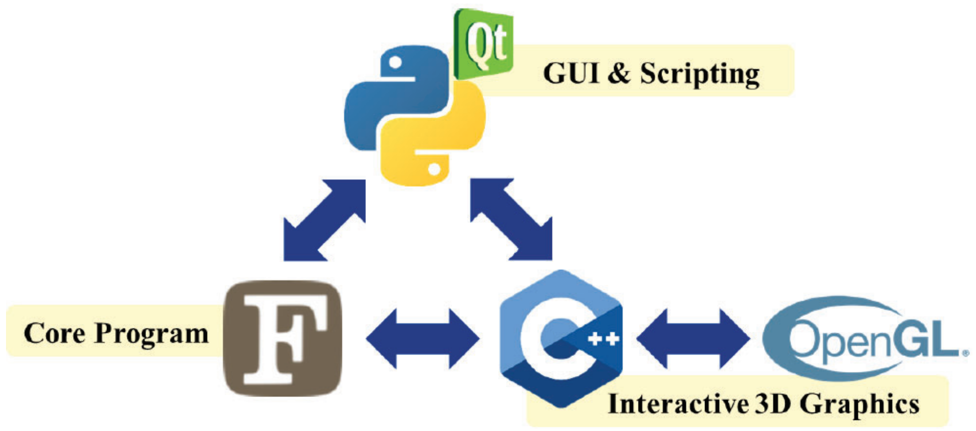


Figure 13. Copernicus Updates for Interoperability (Fortran, Python, and C++)

Once the Copernicus updates were made, the latest version of MONTE was ready to utilize the features for the MONTE-Copernicus interoperability tasks mentioned earlier in this section. See Figure 14 for a high-level comparison of the differences of the two tools. Some compelling benefits of this interoperability between MONTE and Copernicus are as followed:

- MONTE's mostly scripting tool to leverage the 3D graphics capabilities of Copernicus
- Copernicus to utilize higher-fidelity modeling

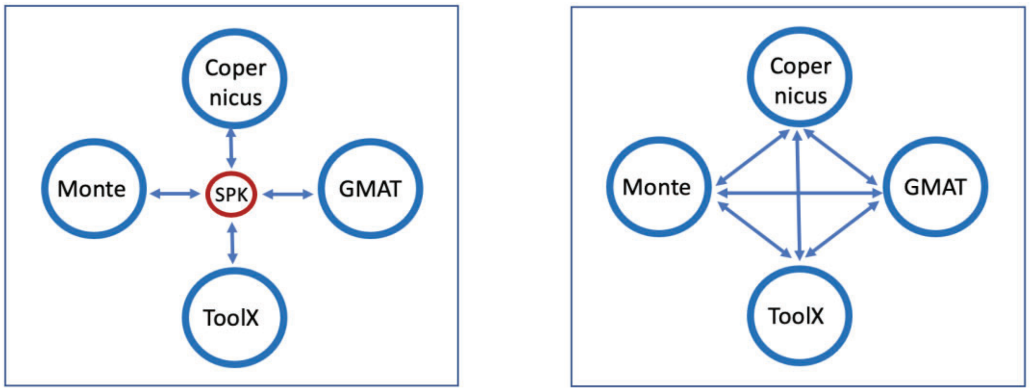
	MONTE	Copernicus
Managing Organization	JPL	JSC
Fidelity	High	Medium
Capabilities	Mission Design + Navigation	Mission Design
User Interface	Scripting	Graphical
Visualization	Simple	Extensive (GUI)
Optimizer	COSMIC*	Itself
Optimization Structure	COSMIC timeline (control/break points)	Segments
Input/Output	*.py, Boa	ideck (*.py)

(*) Note, Computer Optimization System for Multiple Independent Courses (COSMIC) is a software module of MONTE.

Figure 14. MONTE-Copernicus Differences

An initial MONTE-Copernicus interoperability strategy was to establish a one-to-one mapping of one tool's settings to another so its generated trajectory could be transferred into the other tool. Through initial prototyping and testing, this approach proved particularly challenging due to the many ways each tool can be configured. Subtle differences could cause deviations (e.g., how the tool's codebase deals with lower-level mathematical functions, the way a GUI interprets strings/numbers, the way a file is loaded). The permeations of differences for these scenarios were endless so a different process was needed to transfer trajectory information between tools.

In early 2022, Trajectory Reverse Engineering was developed that allowed a user to share trajectory information generated by one tool with another without being locked to any one tool. Instead of focusing on the transferring of trajectory information between Copernicus and MONTE through a meticulously detailed one-to-one mapping process this new process would work with almost any flight mechanics' tool. See Figure 15 for an illustration of the initial strategy for sharing trajectory information between tools and the Trajectory Reverse Engineering process.



a) using a standardized trajectory structure. ***b) specific tool-to-tool interphase design.***

Figure 15. Tool Interoperability Illustration

7.3.1 Trajectory Reverse Engineering

The overall idea of Trajectory Reverse Engineering is to transfer the trajectory from one tool to the next in the form of an SPK (Spacecraft and Planet Kernel) with a *.bsp extension from SPICE, which was developed by the JPL Navigation and Ancillary Information Facility (NAIF). NAIF has several different types of kernels on their website, but for simplicity the bsp kernel will be referred to as an SPK. An SPK file is a container object that represents a trajectory as an invariant structure in phase-space (6D) in the form of ephemerides, agnostic to gravitational environments, fidelity models, or numerical representation of the system, and does not require integration of the equations of motion. It can be thought of as a frozen image of the propagated trajectory. From an SPK the states can be determined at predetermined time intervals or strategic points along the trajectory (e.g., periapsis or apoapsis), maneuvers in the form of velocity discontinuities, and natural central bodies (bodies at which the states are defined). The trajectory can be propagated forward-in-time using the selected set of states. Due to the discrepancy between tool models, small or large discontinuities might appear between the integrated legs, which can be smoothed by the implementation of a multiple shooting algorithm. Some of the existing flight mechanics tools can handle the multiple-shooting implementation automatically (e.g. COSMIC, the trajectory design and optimization module, from MONTE), which makes the script translation a more manageable task. For other tools (e.g., Copernicus) this implementation requires manual attention. With the method outlined, a set of Python scripts were written to automatically implement this process and recover the desired trajectory [ref. 20]. Utilizing the trajectory reverse engineering process to transfer a trajectory from one tool to the next will not result in exact matches down to the machine level (e.g., "lossless") but the solutions are viable based on the strategically targeted control points.

Once the SPK is generated by one tool, a careful scan over the SPK can occur to extract the necessary information to reconstruct (reverse engineer) an orbital path by the receiving tool with minimal or no prior knowledge of the original trajectory, hence, the reverse engineering process name. The scanning process for this MONTE-Copernicus interoperability effort is part of a series of Python scripts. During the scan, strategic locations for placing controls on the trajectory (e.g., periapsis and apoapsis) with respect to a given central body can be identified. The natural bodies to which specific trajectory segments were originally defined can be identified from the SPK file and can be automatically included in the ephemerides model of the recovery trajectory. During the scan process, impulse maneuvers are detected as velocity discontinuities. Additional maneuvers can be added by the user to exert control over the recovery process, which is implemented as an optimization procedure. An optional functionality added to the scanning of the SPK was to utilize an input JavaScript Object Notation (JSON) configuration file, a human-friendly file format, where a user can easily configure their specific problem by defining frames, coordinate systems, parameter sets (e.g., classical orbital elements, hyperbolic, cartesian), parameter constraints, and maneuver constraints. In this configuration file, a user can define desired bodies to be included and dynamics and numerical models (e.g., forces, specific optimizer, etc.). User-defined constraints are also possible, but its implementation would be tool-dependent [ref. 20].

After the scanning process, a timeline for the trajectory to be recovered is created through a set of control points along its path (and optionally impulsive maneuvers), where a multiple-shooting technique is performed to combine discontinuities detected in the ephemerides. See Figure 16 for an illustration of the multiple-shooting technique. This algorithm works by dividing the time period over which the trajectory is to be optimized into a number of discrete intervals. These intervals are defined by every two consecutive control points. For each interval, the algorithm solves for the optimal control inputs that will steer the system from one end of the interval to the other, subject to the constraints and objectives of an optimization problem [ref. 20]. Reference 20 goes into the specifics of the optimization problem used to remove the discontinuities.

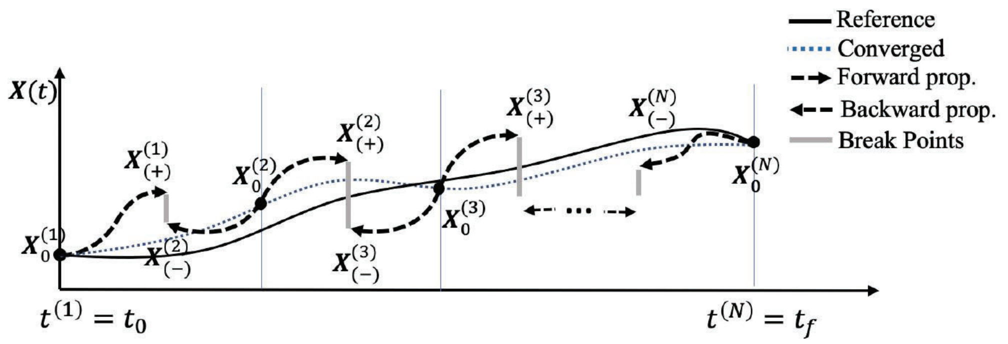


Figure 16. Multiple-Shooting Forward-Backward Propagation Scheme

At this point, a tool-specific script is needed to make use of the data collected to generate a file/input for the receiving tool to use to continue the trajectory reverse engineering process. For this MONTE-Copernicus interoperability effort the following scenarios dealt with transferring data from one tool to another and are expanded on in Sections 7.3.2 through 7.3.6:

1. Generic SPK to Copernicus visualization
2. Copernicus to MONTE trajectory transfer
3. MONTE to Copernicus trajectory transfer

Part of the strength of the trajectory reverse engineering process is in its applicability to any flight mechanics' tool. If that tool and supplemental software can do the following, then it can utilize the trajectory reverse engineering process:

- Ingest a trajectory from an SPK
    - Requires: Implementation of the SPICE toolkit
- Script to process the SPK contents and export into a format the receiving tool can ingest
- Export a trajectory to an SPK
    - Requires: Implementation of the SPICE toolkit

- Requires: Ability to tag each trajectory in the SPK with a specific SPICE Identification (ID)
- Optional: Adjust the step-size of the trajectory stored in the SPK to a sufficient resolution to extract the time maneuvers occur, reduce discontinuities between data points, capture desired perturbations, and other factors a user does not want to lose to subsequent interpolation

During the 33rd American Astronautical Society/American Institute of Aeronautics and Astronautics (AAS/AIAA) 2023 Conference in Austin, Texas the Trajectory Reverse Engineering processes was presented [refs. 20 and 21]. Several inquiries were made during the conference on how to obtain access to the scripts mentioned for the Trajectory Reverse Engineering process, and the flight mechanics tools mentioned in this assessment. Mission and navigation engineers from JPL expressed how this tool would eliminate their current SPK read-in processes for support spacecraft operations and analysis. NASA's software release process was prohibitive when attempting a public release of the trajectory reverse engineering scripts.

For this assessment, the target machine setup for the MONTE-Copernicus interoperability work was for a Windows machine to run Copernicus 5.2.0 and Docker to run MONTE version 149. The custom Python scripts used for the Trajectory Reverse Engineering process are bundled together with the delivery of this assessment final report. All of the use cases mentioned in Sections 7.3.2 through 7.3.6 have supplemental input files that have been bundled with the delivery of this assessment final report and mentioned in Section 10.0. In these supplemental input files, there is a brief description of the use case, all input files needed, and the expected results (see README.txt files). A MONTE-Copernicus interface document is also included to assist users with machine setup to make use of the supplemental input files.

7.3.2 NRHO/Gateway Use Case

This use case represents the Copernicus to MONTE data transfer scenario of a Low Lunar Orbit (LLO) to a NRHO trajectory (see Figure 17). The steps taken for this use case are to 1) ingest the initial Copernicus solution, 2) identify all the relevant control points, events (e.g., apoapsis/periapsis), and identify maneuvers (impulsive burns), 3) Create a COSMIC timeline and optimize (multiple-shooting) to produce a smooth continues optimized trajectory in MONTE, and 4) Save new solution in different formats: *.bsp , *.py, *boa. The boa (Binary Object Archive) file is the standard MONTE format to save a trajectory, and contains ephemeris, trajectory path, dynamical models and partials, and is the final product for Navigation analysis.

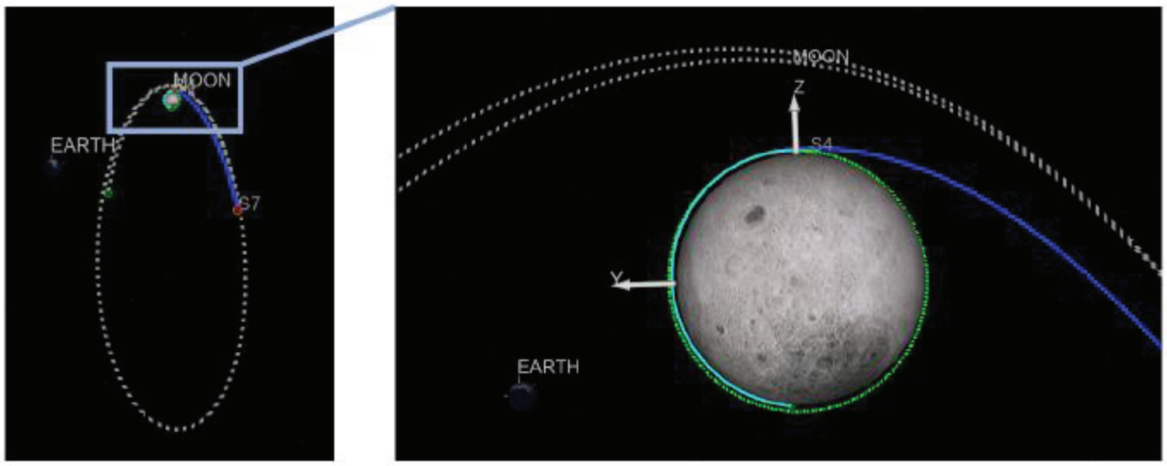


Figure 17. LLO to NRHO Trajectory

7.3.3 Europa Clipper Moon Flybys Visualization Use Case

This use case represents the SPK to Copernicus data transfer scenario where trajectories developed in MONTE or any other tool, are visualized in Copernicus. The Python script used to generate trajectory visualizations in Copernicus from SPK kernels is called bsp2visualCop.py. The script follows the trajectory recovery process outlined in Section 7.3.2. Here, the trajectory contained in the SPK kernel is loaded into Copernicus as an ephemeris file and a segment attached to it as a "static point trajectory" is used for visualization purposes (i.e., no optimization is applied). The Python function requires as input the SPK kernel (.bsp) file. Optional argument inputs (e.g., the spacecraft ID, the central body, the frame) to visualize the trajectory are possible. For more complex trajectory visualizations, an optional JSON input file can be passed to the function call to generate a tailored output (e.g., specific frames, central body, number of segments, colors, etc). The steps taken for this use case are to 1) ingest the MONTE generated SPK, 2) update the supplement JSON config to setup visualization (optional), 3) scan the file an generate ideck, and 4) ingest the ideck file and visualize the trajectory as desired. Figure 18 a shows a Europa Clipper trajectory, where the mission phases are color coded.

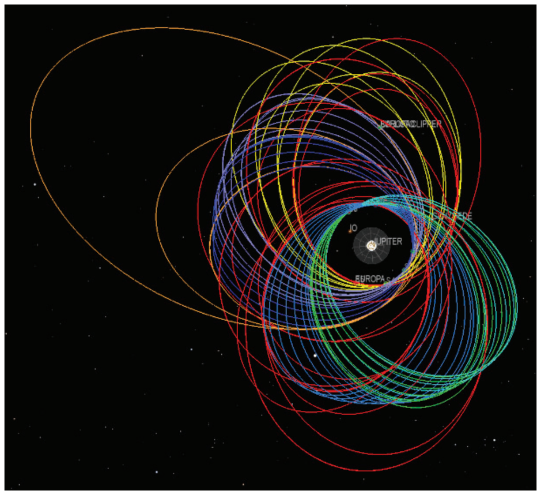


Figure 18. MONTE to Copernicus Europa Clipper Visualization

The steps highlighted in this use case should work with any application that can generate a SPK file and is not limited to the Europa Clipper trajectory displayed in Figure 18.

7.3.4 Icy Moons Multiple Shooting Use Case

This use case represents the Copernicus to MONTE data transfer scenario of an Icy moons Quasi-Periodic Orbits (QPO). In particular, a QPO around Enceladus is considered (See Figure 19). The QPOs often exist in low-fidelity CR3BP solutions created in Copernicus but breakdown when transferred to high-fidelity dynamics modeling in MONTE. The script used to perform the transcription is called bsp2cosmic.py. The steps taken for this use case are to 1) Design the QPO in Copernicus and export the trajectory to SPK, 2) convert the trajectory into a COSMIC timeline (MONTE trajectory), 3) discretize the ballistic trajectory into N number of segments, and 4) utilize MONTE to perform multiple-shooting method techniques, as seen in Figure 16, to reconverge on a solution that is along the phase space of the original low-fidelity frozen trajectory but implemented with the new dynamics.

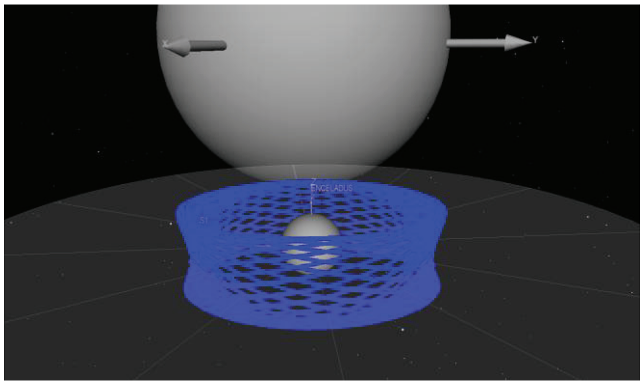


Figure 19. Icy Moons QPO Visualization

7.3.5 Enceladus Orbiter Use Case

This use case represents the Copernicus to MONTE data transfer scenario of a trajectory with complex mission phases (e.g., Moon tour, capture, mapping orbit) that tends to break down with high-fidelity modeling. The steps taken for this use case are to 1) Design the initial solution with low-fidelity modeling in Copernicus for a phase of the trajectory and export that trajectory to an SPK, 2) identify all the relevant control points (e.g., maneuvers) and events (e.g., apoapsis/ periapsis), 3) convert the trajectory into a MONTE file, 4) utilize MONTE to perform multiple shooting method techniques, as seen in Figure 16, to reconverge on a solution that is along the phase space of the original low-fidelity trajectory, and 5) repeat the process to the desired final solution for the entire trajectory. The final solution showcases a way of incorporating high-fidelity dynamics early in the design process, coupled with Copernicus' intuitive GUI, as iterations between Copernicus and MONTE are performed. Figure 20 depicts a trajectory that is dynamically sensitive to being transferred from low- to high-fidelity modeling.

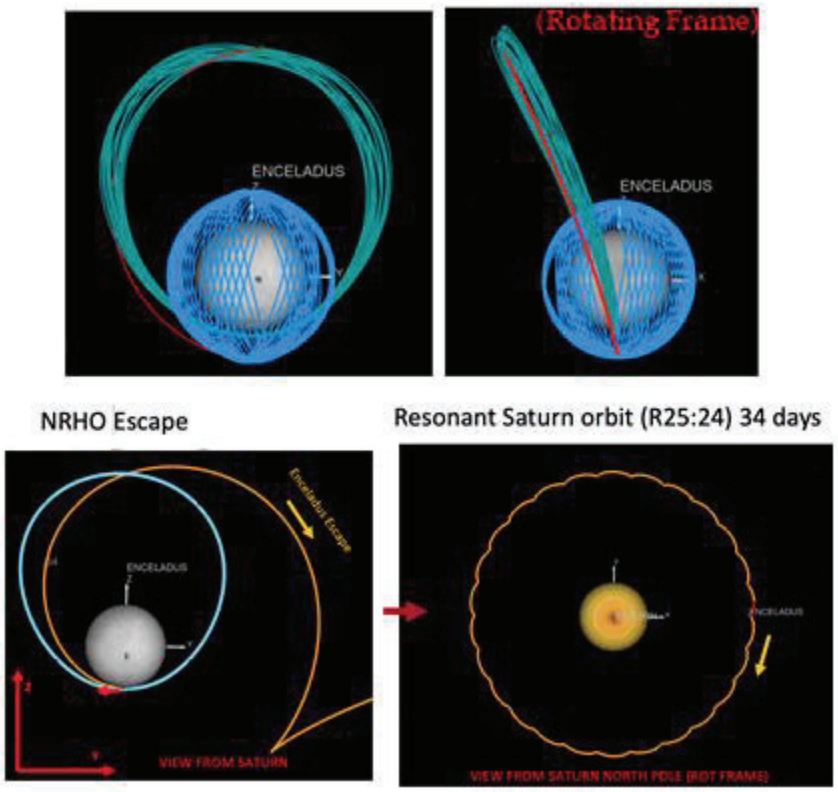
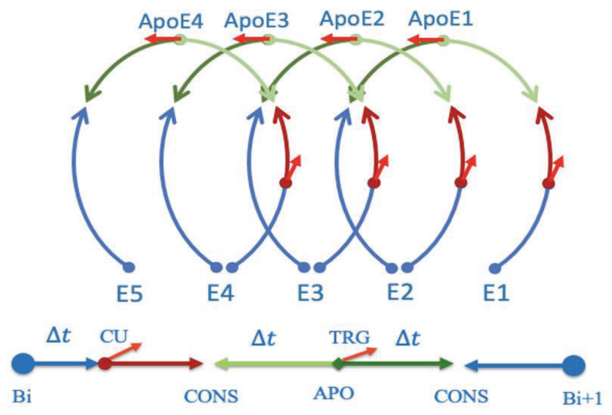
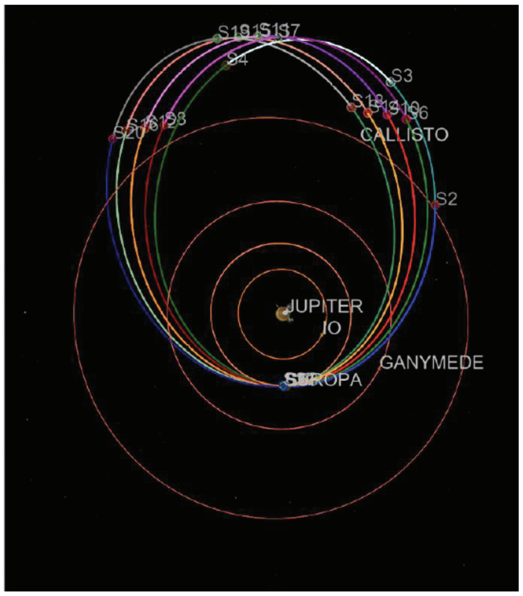


Figure 20. Dynamical Sensitive Trajectory in Low-Fidelity to High-Fidelity Transfer

7.3.6 Flyby Use Case

This use case represents the MONTE to Copernicus data transfer scenario of a Europa Clipper flyby trajectory design. Figure 21 depicts the Europa Clipper multiple-flyby architecture. The script used is bsp2cop.py. The steps taken for this use case are to 1) Design the trajectory in MONTE and export to an SPK, 2) identify all the relevant control points (e.g., maneuvers) and events (e.g., apoapsis/periapsis), 3) convert the trajectory into a Copernicus file, and 4) utilize Copernicus' GUI to adjust the segments and/or constraints/functions. To generate an ideck from an SPK kernel, a similar process to the one implemented for bsp2cosmic.py is performed, but the multiple-shooting strategy is manually set up on the Python script by creating segments that are propagated forward and backward in time while imposing continuity true state constraints.





a) Segment structure implemented in Copernicus. *b) Converged trajectory.*

Figure 21. Europa Clipper Multiple-Flyby Architecture

8.0 Findings, Observations, and NESC Recommendations

8.1 Findings

The following findings were identified:

F-1. The Copernicus, GMAT, and MONTE software release processes are different based on the NASA center the development team is associated with and each flight mechanics tool has its own restrictions on who can obtain the software and associated source code. Source code for the Copernicus and MONTE core system could not be distributed to the assessment team members that were not part of the tool's development team.

F-2. GMAT 2022a's generalized implementation of the external force model plug-in allowed for more tools to make use of it but significant computational performance improvements could be made for a specialized MONTE implementation where less files could be exchanged per integration step (e.g., boa).

8.2 Observations

The following observations were identified:

O-1. Flight mechanics tools incorporating SPK functionality would enable tool interoperability when paired with the Trajectory Reverse Engineering process.

O-2. The supplemental Python scripts to support the MONTE-Copernicus use cases are not accessible to users outside of the NESC Board.

O-3. MONTE, GMAT, and Copernicus interoperability features mentioned in this assessment require a functional interface with Python.

O-4. The following are specific versions of software that enabled the flight mechanic tools interoperability functionality mentioned in this assessment:

- Starting with Copernicus 5.0 and the inclusion of the RoboCopPy Python interface, Copernicus and MONTE were able to exchange needed data to support the identified use cases.
- The GMAT 2022a public release contains the external force model plugin (i.e., alpha) that enables the ability for GMAT to make use of MONTE dynamics.,
- The GMAT 2022a public release contains the GMAT API functionality to allow external tools to make use of components in environments external to GMAT's native interfaces.
- Starting with the GMAT 2020a release, this tool uses the same 3D graphics technology as Copernicus (i.e., OpenFrames/OpenSceneGraph) where both tools benefit from updates and bug fixes.
- MONTE version 149 was utilized in the generation of data for the MONTE-Copernicus use cases that make use of the Trajectory Reverse Engineering process.
- MONTE version 152 and a GMAT 2022a equivalent build was utilized in the Lucy spacecraft EGA shadow operations MONTE-GMAT interoperability effort.
- Copernicus version 5.0.0 decoupling the GUI from the base Fortran code and switching the GUI to be in Python allowed the GUI source code to be released allowing the user community to independently modify or replace the GUI.

8.3 NESC Recommendations

The following NESC recommendations were identified and directed towards the Copernicus, GMAT, and MONTE development teams to maintain interoperability between flight mechanics tools:

R-1. Maintain the SPICE toolkit functionality to export trajectories to an SPK and apply SPICE ID tags into flight mechanics tools for future Copernicus, GMAT, and MONTE software releases. ***(O-1)***

R-2. Maintain the Python interface functionality to pass data through to another flight mechanics tool for future Copernicus, GMAT, and MONTE software releases. ***(O-3)***

R-3. Distribute the interoperability functionality mentioned in this assessment as part of the Copernicus, GMAT, and MONTE software releases. ***(O-1, O-2, O-3)***

R-4. Provide institutional funding to Copernicus, GMAT, and MONTE development teams for on-going maintenance of the interoperability functionality mentioned in this assessment. ***(O-3)***

The following NESC recommendations were identified and directed towards the Copernicus users seeking to improve interoperability between flight mechanics tools:

R-5. Upgrade to Copernicus 5.0.0 or later version [ref. 23] to make use of the features described in this assessment. ***(O-2, O-3)***

The following NESC recommendations were identified and directed towards the GMAT users seeking to improve interoperability between flight mechanics tools:

R-6. Upgrade to GMAT 2022a to make use of the features described in this assessment. ***(O-1, O-3, O-4)***

The following NESC recommendations were identified and directed towards the MONTE development team seeking to improve interoperability between flight mechanics tools:

R-7. Bundle the trajectory reverse engineering python scripts and associated data in future MONTE software releases and maintain the scripts with each subsequent release. ***(O-2)***

The following NESC recommendations were identified and directed towards NASA STMD which governs NPR 2210, to improve interoperability between flight mechanics tools:

R-8. Update NASA software release policies (e.g., SUAs, NPR 2210.1) to make the distribution of flight mechanics software executables, source code, and supplemental files/documentation between NASA centers easier and consistent to boost collaboration. ***(F-1, O-2)***

R-9. Release the source code of MONTE and Copernicus to the NASA community. ***(F-1)***

9.0 Alternate Technical Opinion(s)

No alternate technical opinions were identified during the course of this assessment.

10.0 Other Deliverables

The following artifacts were included with this assessment upon delivery to the NESC Board in addition to the deliverables listed in Table 1:

- GMAT API user needs survey feedback
- 2020 Flight Mechanics TDT Face-to-Face presentation on GMAT R2020 API
- LUCY EGA Shadow Operations Presentation and scripts
- MONTE-Copernicus Trajectory Reverse Engineering Interface Document
- MONTE-Copernicus Trajectory Reverse Engineering scripts and use case examples
- Copernicus, GMAT, and MONTE user documentation

11.0 Lessons Learned

Under this task, the Copernicus, MONTE, and GMAT development teams collected, and documented lessons learned on the efforts to utilize the software in an SoS environment. Additionally, the teams documented policies and procedures that promoted increased software sharing and collaboration between centers.

<u>Sharing Interfaces</u>

- Python served as a medium to transfer data and component information between flight mechanics tools. This common bridge between the different tools greatly enabled the work presented in this report.
- Updates made to Copernicus in 2018 to 2019 via the OpenFrames interface were mutually beneficial to GMAT in the following ways: support for advanced user interfaces embedded in the 3D scene, realistic lighting on celestial bodies and spacecraft, hyper-realistic celestial body models that increase resolution as the viewer approaches the surface, sensor visualization, and viewing a scene in consumer-grade VR hardware (e.g., Oculus Rift or HTC1 Vive).

Flight Mechanics Tool 1-to-1 Mission Sequence Matching

- In theory, the concept of aligning all settings between two flight mechanics tools for a spacecraft mission sequence of events to produce the same results seems viable but in practice there are numerous areas that cause slight differences or require different implementations to ensure the output matches. It is not so much that the two tools could not be tweaked here and there to better match one another, but rather the number of resources needed to do this synchronization would be non-trivial and not easily generalized especially as the mission sequence increases in complexity.

GMAT API Development

- The initial API design provided lessons about how to proceed with a publicly available API for the GMAT code base. Users of the SWIG interfaces as they existed in the initial implementation found that the component settings and interdependencies for objects in the GMAT code are not always obvious, and that there is a fair amount of detailed configuration needed in order to access GMAT features. Once the user survey was distributed, the API development team collected the lessons learned from the initial implementation and added more functionality to increase accessibility to GMAT features.

GMAT External-Force Model Plugin

- Lucy Ops Staff: Several of the function and parameter names were confusing or unclear when using the external force model plugin. Two alternatives were suggested, which both sides agreed would improve readability and ease of use. These changes are:
    - Change ExcludeOtherForces to ExcludeGmatForces
    - Change GetForces to GetAcceleration
- Lucy Ops Staff: Include more input options.
    - At each integration step MONTE must pass in a boa. Loading this is a slow process and is understood to be one of the main factors hindering the performance for the MONTE-GMAT integration. The Lucy team asked to be able to pass the boa in once and have GMAT store the information to save computation time. This specific request to include capabilities in GMAT to specifically ingest boa files is unlikely to be met, since GMAT source code is release open source, while MONTE is under ITAR.
- Dev Team: Support other derivative types beyond cartesian state.
    - The plugin currently only supports derivatives of the cartesian state. This covers the majority of use cases and was the only type needed by the Lucy team, so it was the first to be implemented. Orbit matrix and orbit state transitions matrix are both supported by GMAT but still need to be implemented for the external force model.
- Dev Team: Implement support for Torque modeling.
    - GMAT force models use a function called GetTorquesForSpacecraft() to calculate the torques generated by the model on the object. Torques were not required for the Lucy demonstration, so it is not currently supported by the external force model. Adding this capability is necessary for the forces in the plugin to have the same capabilities as the GMAT internal force models and bring the feature out of alpha.
- Dev Team: Replace custom array math code with a Python package (e.g., NumPy).

    - Initial attempts were made to utilize some Python packages (e.g., NumPy) for array math and other common math functions but there were issues with the package import. In the interest of time, custom scripting was utilized for some math functions. The custom code should be replaced by well vetted Python packages that can perform array math calculations. Importation of packages and libraries is a core part of Python, which users expect to have fully functional. Elevating the external force model plug-in out of alpha involves the inclusion of the fully functional capability to import packages.
- Dev Team: Create ability to set location of the force models Python script.
    - Currently the Python script must be in the "gmat/userfunctions/python" folder. This should be changed to bring the feature more in line with other GMAT capabilities which are location agnostic.
- Dev Team: Clean-up warning messages.
    - When running the plugin, GMAT records two warning messages to the log and message window. These messages do not contain relevant information and should be resolved/removed to not confuse the user.

12.0 Recommendations for NASA Standards and Specifications

No recommendations for NASA standards and/or specifications were identified as a result of this assessment.

13.0 Definition of Terms

Finding	A relevant factual conclusion and/or issue that is within the assessment scope and that the team has rigorously based on data from their independent analyses, tests, inspections, and/or reviews of technical documentation.
Lessons Learned	Knowledge, understanding, or conclusive insight gained by experience that may benefit other current or future NASA programs and projects. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure.
Observation	A noteworthy fact, issue, and/or risk, which may not be directly within the assessment scope, but could generate a separate issue or concern if not addressed. Alternatively, an observation can be a positive acknowledgement of a Center/Program/Project/Organization's operational structure, tools, and/or support provided.
Recommendation	A proposed measurable stakeholder action directly supported by specific Finding(s) and/or Observation(s) that will correct or mitigate an identified issue or risk.

14.0 Acronyms and Nomenclature

2D	Two-dimensional
3D	Three-dimensional
6D	Six-dimensional
AAS	American Astronautical Society

AIAA	American Institute of Aeronautics and Astronautics
API	Application Programming Interface
boa	Binary Object Archive
Copy	Copernicus Python
COSMIC	Computer Optimization System for Multiple Independent Courses
CR3BP	Circular Restricted Three Body Problem
DDOR	Delta-Differential One-Way Ranging
DSG	Deep Space Gateway
DPTRAJ/ODP	Double Precision Trajectory and Orbit Determination Program
DRO	Distant Retrograde Orbit
DSM	Deep Space Maneuver
DSN	Deep Space Network
EGA	Earth Gravity Assist
EM-1	Exploration Mission 1
EMTG	Evolutionary Mission Trajectory Generator
FDF	Flight Dynamics Facility
GMAT	General Mission Analysis Tool
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
hi-fi	High Fidelity
ID	Identification
ideck	Input Deck
JPL	Jet Propulsion Laboratory
JSC	Johnson Space Center
JSON	JavaScript Object Notation
LaRC	Langley Research Center
LEO	Low Earth Orbit
LLO	Low Lunar Orbit
MGSS/AMMOS	Multi-mission Ground System and Services/ Advanced Multi-Mission Operations System
MIRAGE	Multiple Interferometric Ranging Analysis using GPS Ensemble
MONTE	Mission-Analysis Operations Navigation Toolkit Environment
NAIF	Navigation and Ancillary Information Facility
NESC	NASA Engineering and Safety Center
NRB	NESC Review Board
NRHO	Near Rectilinear Halo Orbit
OD	Orbit Determination
ODTBX	Orbit Determination Toolbox
QPO	Quasi-Periodic Orbits
SBIR	Small Business Innovation Research
SEP	Solar Electric Propulsion
SoS	System of Systems
SPICE	Spacecraft Planet Instrument "C-matrix" Events
SPK	Spacecraft and Planetary Kernel
SRA	Software Release Authority
SSMO	Space Science Missions Operations

SUA	Software Usage Agreement
SWIG	Simplified Wrapper and Interface Generator
TDT	Technical Discipline Team
TM	Technical Memorandum
TRL	Technology Readiness Level
VR	Virtual Reality

15.0 References

1. MONTE: the next generation of mission design and navigation software https://trs.jpl.nasa.gov/handle/2014/46063
2. MONTE python for deep space navigation https://trs.jpl.nasa.gov/handle/2014/46155
3. MONTE distribution overview document
4. MONTE brochure https://montepy.jpl.nasa.gov/static/document/brochure.pdf
5. GMAT overview https://gmat.atlassian.net/wiki/spaces/GW/overview
6. GMAT User's Guide https://gmat.sourceforge.net/docs/R2022a/html/index.html
7. Copernicus Overview https://www.nasa.gov/centers/johnson/copernicus/index.html
8. Copernicus Spacecraft Trajectory Design and Optimization Program, FortranCon 2020 https://ntrs.nasa.gov/citations/20205003826
9. Copernicus Software Request https://software.nasa.gov/software/MSC-26673-1
10. GMAT API Users Guide https://sourceforge.net/p/gmat/git/ci/GMAT-R2022a/tree/application/docs/GMAT_API_UsersGuide.pdf
11. GMAT Download https://sourceforge.net/projects/gmat/files/GMAT/\
12. What is SWIG? https://www.swig.org/exec.html
13. GMAT Release Notes https://gmat.sourceforge.net/docs/R2022a/help.html#ReleaseNotesR2022a
14. LUCY Shadow Maneuver Planning and Navigation Operations presentation on October 25, 2022
15. OpenFrames API repository https://github.com/ravidavi/OpenFrames
16. OpenSceneGraph Website http://www.openscenegraph.com/
17. OpenFramesInterface Wiki https://gitlab.com/EmergentSpaceTechnologies/OpenFramesInterface/-/wikis/home
18. GMAT API Presentation to the April 2020 Flight Mechanics TDT Face-to-Face
19. NESC-RP-15-01097 "Improvements to the Copernicus Trajectory Design and Optimization System for Complex Space Trajectories" November 15, 2018, Daniel G. Murri, et. Al
20. AAS 23-312 "Trajectory Reverse Engineering: A General Strategy for Transferring Trajectories Between Flight Mechanics Tools" January 2023 Ricardo L. Restrepo
21. 33rd AAS/AIAA Space Flight Mechanics Meeting, Austin, Texas, January 15-19, 2023, https://www.space-flight.org/docs/2023_winter/2023_winter.html
22. AAS 20-580 "Using the GMAT Application Programmer's Interface" August 2020 Darrel J. Conway, John McGreevy
23. Copernicus 5.0.0 Release Notes

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 04/27/2023	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Flight Mechanics Analysis Tools Interoperability and Component Sharing	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Koehler, Heather; Dove, Edwin	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER 869021.01.23.01.01
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-2199	**8. PERFORMING ORGANIZATION REPORT NUMBER** NESC-RP-18-01313
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/TM-20230006507

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category Space Transportation and Safety
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Several NASA centers have developed independent flight mechanics tools to meet the science needs of missions. This NASA Engineering and Safety Center (NESC) assessment sought to explore the ways to increase the interoperability of three specific tools: Copernicus from Johnson Space Center (JSC), the General Mission Analysis Tool (GMAT) from Goddard Spaceflight Flight Center (GSFC), and the Mission-Analysis Operations Navigation Toolkit Environment (MONTE) from the Jet Propulsion Laboratory (JPL). Before this assessment, these tools were not integrated and could not easily share data, models, or components. This report contains the outcome of the NESC assessment.

15. SUBJECT TERMS
General Mission Analysis Tool; NASA Engineering and Safety Center; Mission-Analysis Operations Navigation Toolkit Environment; Copernicus; Flight Mechanics Analysis Tools

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	43	(443) 757-5802