(Preprint) AAS 23-312

TRAJECTORY REVERSE ENGINEERING: A GENERAL STRATEGY FOR TRANSFERRING TRAJECTORIES BETWEEN FLIGHT MECHANICS TOOLS

Ricardo L Restrepo*

Space exploration missions are usually a product of the collaboration between different teams, generally involving the use of different flight mechanics tools, such as trajectory design, orbit determination, or mission planning. Each tool has its own set of dynamical and numerical models. Slight differences between the models can cause differences between the shared trajectories, which may be unacceptable for comparison purposes. To overcome this issue, we propose a trajectory-sharing process via SPK kernels, which are agnostic to the models used. A judicious kernel scan is used to recover the original trajectory. Examples of transfer solutions between the NASA tools Monte and Copernicus are presented, including Earth-to-Moon transfers, interplanetary trajectories, Moon tours, and icy-worlds orbiter solutions.

INTRODUCTION

Space exploration missions are usually a product of the collaboration between different institutions and different teams within the same institution. Each team typically has a preference and utilizes a specific set of tools, making the transfer of data between teams and institutions an important part of the problem. In this paper we focus on bridging the gap specifically between flight mechanics tools, finding a seamless transfer of trajectory solutions between teams.

In general, sharing or transferring trajectory solutions between flight mechanics tools is a complex endeavor. Each tool has its own set of dynamical models and numerical algorithms; hence, propagating the same set of initial conditions in one tool will not yield the same trajectory outcome in another tool. To map a trajectory between different tools, the dynamical environment as well as the numerical algorithms used (e.g. order, step size, and tolerance of the integrator) need to be practically identical. Even when the task of replicating the models is carefully performed, realistic scenarios generally occur under complex dynamical models (e.g. multi-body dynamics and highly perturbed environments), where slight differences in the fidelity of the model, or between the numerical representation of the systems will make a trajectory diverge in one tool and converge in another. Thus, it is virtually impossible to exactly recover the original trajectory in an alternative tool given the same state initial conditions. An additional challenge of transferring trajectory solutions is that it requires the use of compatible state representation (e.g., cartesian, orbital elements, etc) and frames, which do not always map one-to-one between tools.

To overcome the challenge of transferring trajectories between flight mechanics tools, we propose *Trajectory Reverse Engineering*, a strategy that is independent of the trajectory design tool.

*Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109. Corresponding Author: Ricardo L Restrepo; *E-mail address*: ricardo.l.restrepo@jpl.nasa.gov.

The overall idea is to transfer the trajectory in the form of SPK (Spacecraft and Planet Kernel) from SPICE, developed by NAIF at the Jet Propulsion Laboratory* (JPL). An SPK file is an object that represents a trajectory as an invariant structure in phase-space (6D), 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 one can determine states 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 then 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 then be smoothed out by the implementation of a multiple shooting algorithm.[1] 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, like Copernicus, this implementation requires manual attention. With the method outlined above, a set of python scripts have been written to automatically implement this process, and recover the desired trajectory.

Trajectory Reverse Engineering emerged from the necessity to connect multiple NASA flight mechanics tools, in an attempt to reduce redundancy and work effort between different centers, and was sponsored by the Flight Mechanics Technical Fellow under the NASA Engineering & Safety Center (NESC), as part of the Flight Mechanics Analysis Tools Interoperability and Component Sharing project. The three main flight mechanics tools developed at NASA are Monte (developed and maintained at the Jet Propulsion Laboratory), Copernicus‡ (developed at maintained at Johnson Space Center), and GMAT§ (developed and maintained at Goddard Space Flight Center). Other private companies develop their own flight mechanics tools, depicted as "ToolX" in Figure 1. The traditional way of operating between tools has been to create specific transfer strategies between a pair of tools, as depicted in Figure 1(b). We propose instead a common transfer format, by making use of the SPK kernel (BSP), reducing the amount of script permutations to just one (1(a)). The strategy developed in this paper is intended to be general for any flight mechanics tool; however, as a proof of concept, the paper is focused on the interphase of solutions between Copernicus and Monte.

Copernicus is a medium-fidelity trajectory design tool that is particularly well-suited for optimization tasks. One of the key strengths of Copernicus is its excellent visualization capabilities, which allows users to easily understand, interpret and manipulate trajectory solutions. On the other hand, Monte has the capability to do high-fidelity trajectory design as well as navigation analysis, including orbit determination and flight path control. Monte has been used extensively for navigating numerous flown missions, Cassini[2] and Juno[3] for example. Even though Monte is incredibly powerful, it lacks in its visualization capabilities. Therefore, a common flow to design a trajectory is to use Copernicus to obtain an initial trajectory, and then transfer the solution to Monte to obtain a high-fidelity solution. Figure 2 summarizes the differences between Monte and Copernicus. All the examples outlined in this paper assume a flow from Copernicus to Monte, as a proof of concept of the usage of the proposed strategy.

*https://naif.jpl.nasa.gov/naif/index.html
†https://montepy.jpl.nasa.gov/
‡https://www.nasa.gov/centers/johnson/copernicus/index.html
§https://software.nasa.gov/software/GSC-17177-1

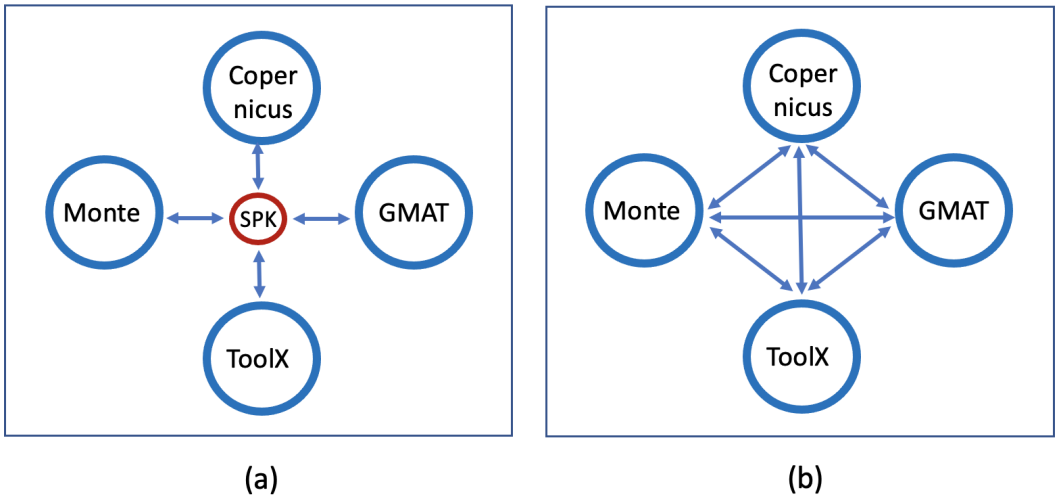


Figure 1. Interoperability between flight mechanics tools, (a) using a standarized trajectory structure and (b) specific tool-to-tool interphase design.

The paper is outlined as follows. We begin by explaining the general strategy of *Trajectory Reverse Engineering* and the particular scripts implemented under this concept. We then proceed to show a series of examples, from simplest to more complex, implementing the reverse engineering trajectory recovering process. The first application is to use Copernicus to visualize any trajectory contained in an SPK kernel, regardless of the tool that was used to generate the trajectory. The next section proceeds to recover trajectories from SPK kernels using Monte, including the trajectory recovery from low to high fidelity dynamics of solutions steaming from Copernicus. Furthermore, a complex Enceladus moon tour example is used to show feature details of the strategy presented here, including the maneuver recovery process and the optimization set up. We finalize the paper with final conclusions and thoughts for future development.

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

Figure 2. Monte vs. Copernicus.

TRAJECTORY REVERSE ENGINEERING: GENERAL STRATEGY

The process of transferring trajectory solutions between different spaceflight mechanics tools involves the transfer of multiple data, like states along the trajectory, usually in a particular set of parameters (e.g., Cartesian, classical orbital elements, geodetic coordinate, etc), frame definitions, time coordinates, ephemerides of the celestial bodies included, numerical and dynamical models, and more. This transcription is usually accomplished through a manual process, through the use of tables with a particular data structure that is tool dependent. This process can be cumbersome, time-consuming, and prone to errors. Additionally, even when the data transfer is performed carefully, it may still result in a final discontinuous trajectory due to discrepancies in the numerical or dynamical models between the original and new tool, requiring further numerical procedures, such as differential correction, to smooth out these discontinuities.

We present here a general and robust method for transferring trajectory solutions between multiple astrodynamics tools using SPK kernels. SPK kernels were created for the design and implementation of the "SPICE" ancillary information system, and are commonly used within the astrodynamics community. These kernels provide an invariant representation of the trajectory that is agnostic to gravitational environments, fidelity models, and numerical representations of the system, and do not require the integration of the equations of motion. By performing a careful search over this invariant structure, it is possible to extract the necessary information to reconstruct an orbital path with minimal or no prior knowledge of the original trajectory, resembling a reverse engineering process.

Scanning SPK Kernels

SPK kernels are used to store ephemerides, i.e., the position and velocity evolution of natural and artificial bodies, in the form of binary files with extension *".bsp"*. The ephemerides are represented as inertial Cartesian coordinates and are saved as a set of coefficients of an interpolated predefined polynomial (e.g., Lagrange, Hermite, etc). Multiple information can be stored in a single SPK file, hence, an identification SPICE ID associated to each trajectory is required. Because the files are binary objects, it is not possible to simply load them in a text editor to look for information. NAIF provides a set of utilities in a variety of programming languages like C, Fortran, Python, or Java, that allows to extract information from these kernels. Additionally, a binary executable called *"brief"*[*] allows a user to quickly read basic information from the kernels, like the bodies included in the kernel, the SPICE ID of all bodies, as well as the central body the states are refered to, by simply typing *"brief myTrajectory.bsp"* in the command line. The SPICE toolkit can be used to read states at any point along the trajectory, and, with basic numerical libraries, allows to identify geometric places of interest, like periapsis or apoapsis with respect to a central body, impulsive maneuvers in the form of velocity discontinuities, or even finite burns.

Trajectory Recovery Process:

By scanning any SPK kernel as mentioned in the above subsection, it is possible to extract necessary information to reconstruct the original trajectory. The first step is to define the initial and final states at the beginning and the end of the trajectory leg. These initial and final states can be fixed, bounded, or allowed to move during the recovery process, depending on the nature of the problem. Control points can be placed along the trajectory to control or impose constraints on the trajectory path. During the scan, strategic locations for placing controls, such as the periapsis and apoapsis

[*] `https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/C/ug/brief.html`

of the trajectory with respect to a given central body can be identified. The natural bodies to which specific segments of the trajectory were originally defined can also 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 can be 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.

Periapsis and apoapsis are detected by scanning the trajectory for minimum and maximum ranges from the central body. Impulsive maneuvers along a spacecraft trajectory can be identified as velocity discontinuities. To detect such discontinuities, the trajectory is discretized into small segments. An impulse burn can be approximated as $\Delta v_i = v_i^+ - v_i^-$, where v_i^+ and v_i^- are the velocities at the beginning and end of any segment i, respectively. The time step size of the segments should be small enough to not account for velocity changes due to natural accelerations of the system. For example, the spacecraft trajectory vector may experience a fast rotation as well as a fast magnitude change in its close approach to gravitational bodies. However, choosing a time step size that is too small could lead to long computation times. By defining a minimum Δv detection value for the discontinuity search, an efficient time step size can be chosen that allows for fast searches while excluding velocity changes due to the accelerations of the system. In addition, finite burns can be identified along predefined segments by approximating the trajectory path as following Keplerian motion. In the Keplerian model, finite burns can be observed as variations in the two-body energy of the spacecraft with respect to the central body. More detail information on the recovery of finite burn maneuvers is explained in the last section of the paper.

By using an input JSON (JavaScript Object Notation) configuration file, a human friendly file format, a user can easily configure their specific problem by defining frames, coordinate systems, parameter sets (classical orbital elements, hyperboilic, cartesian, ...), and parameter constraints, and maneuver frames and bounds. In this configuration file a user can also define any desired bodies to be included, forces, specific optimizer, etc. User defined constraints are also possible, but its implementation will be tool-dependent.

Multiple Shooting Strategy to Solve the Trajectory Reverse Engineering Problem

The main idea behind "Trajectory Reverse Engineering" is to map points in the form of states along the path of the trajectory. During the scanning process strategic locations for controls points are added, such as periapsis and apoapsis of the trajectory, maneuvers, or any set of points along the trajectory that a user may define. When propagating the selected set of states at the desired control points, discontinuities along the trajectory will naturally occur due to discrepancies in the models between the original and new tool (see Figures 3 and 4). To get rid of these discontinuities a numerical procedure is implemented.

The multiple shooting algorithm is a powerful method for solving this particular trajectory design problem. 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 the optimization problem.

Figures 3 and 4 show a forward and forward-backward propagated scheme for a multiple shooting algorithm, respectively. In order to described the implementation procedure, the forward prop-

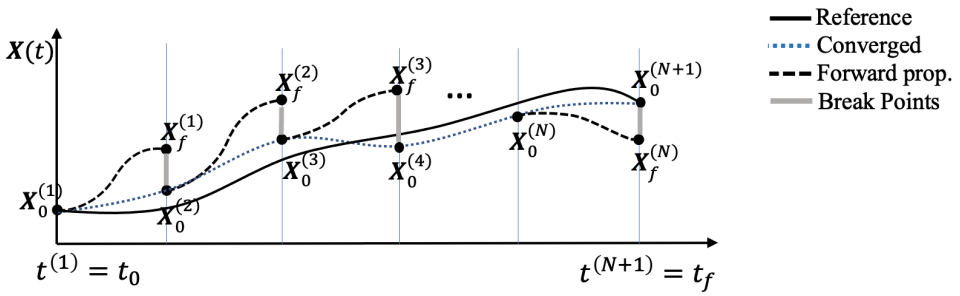


Figure 3. Multiple-shotting forward propagation scheme.

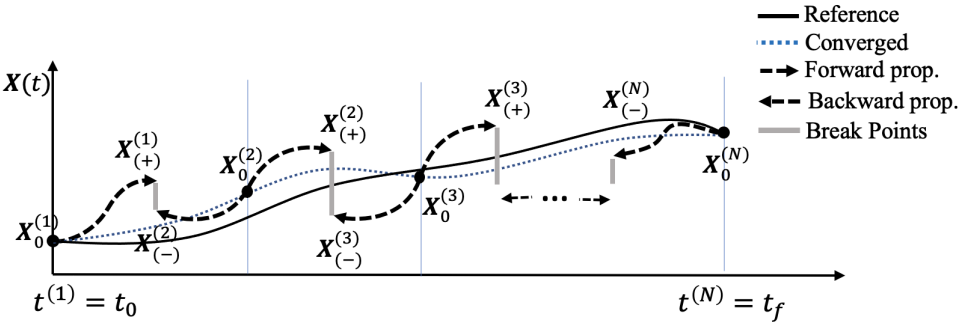


Figure 4. Multiple-shotting forward-backward propagation scheme.

agation scheme is used. Here, the trajectory to be solved for has been discretized into N control points, where the goal is to remove the state discontinuities between two adjacent segments $X^{(i)}$ and $X^{(i+1)}$. A differential corrector algorithm is then set up to get rid of the whole set of discontinuities as defined in Eq. (1). The initial and final points can be fixed or free, depending on the nature of the mission design problem. If the problem requires the extreme points to be fixed, an additional equality constraint is added (Eq. (2) to the differential corrector problem. Additionally, bounds on the parameter of the states can be imposed by adding inequality constraints, defined in Eq. (3), where q_i represents any parameter of a given control point. For example, the minimum altitude of a flyby at closest approach can be constrained, to keep the trajectory collision free. If maneuvers are included in the trajectory recovery process, then a cost function can be included to minimize the sum of all the ΔVs as shown in Eq. (4). In this case, the problem becomes an optimization problem, where the objective is to remove discontinuities while minimizing the objective function.

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}_f^{(1)} - \mathbf{X}_0^{(2)} \\ \vdots \\ \mathbf{X}_f^{(N-1)} - \mathbf{X}_0^{(N)} \end{bmatrix} = \mathbf{0} \tag{1}$$

$$\mathbf{\Theta} = \begin{bmatrix} \mathbf{X}_0^{(1)} - \mathbf{X}_0^{*} \\ \mathbf{X}_f^{(N)} - \mathbf{X}_f^{*} \end{bmatrix} = \mathbf{0} \tag{2}$$

$$\mathbf{\Psi} = \begin{bmatrix} q_1 - q_1^* \\ \vdots \\ q_p - q_p^* \end{bmatrix} \leq \mathbf{0} \tag{3}$$

$$J = \sum_{i=1}^{n} |\Delta V_i| \tag{4}$$

COPERNICUS AS A TRAJECTORY VISUALIZATION CAPABILITY

As mentioned in the Introduction, Copernicus is a flight mechanics tool that stands out for its powerful visualization. In this section we describe one of the simplest applications of *Trajectory Reverse Engineering*, which is to take any trajectory in the form of SPK file and visualize it in Copernicus.

As part of the NESC tool interoperability project, releases of Copernicus 5.0 and above include a python interphase, that allows any user to create a full solution from a python script.[4] Making use of this python Copernicus interface, a python script called *bsp2visualCop.py* has been developed as part of *Trajectory Reverse Engineering* to visualize any trajectory in Copernicus. The foundation of the script follows the trajectory recovery process outlined in the previous section. Here, the trajectory contained in the SPK kernel is loaded into the Copernicus SPICE tool and simply scanned to create a "static point trajectory" for visualization purposes. The python function call requires as input the SPK kernel (*.bsp*) file. Optional argument inputs are the spacecraft ID, the central body, the frame to visualize the trajectory in, the amount of days, a specific set of bodies to visualize, and the color. Several examples of the function call are shown below.

```
a) >> bsp2visualCop.py lrorg_2009169_2010001_v01.bsp -sc -85
b) >> bsp2visualCop.py lrorg_2009169_2010001_v01.bsp -sc -85 -tl 10 -c Moon -f iau_body_fixed
c) >> bsp2visualCop.py leo_to_nrho.bsp -sc -30100 -c Moon -f iau_body_fixed
d) >> bsp2visualCop.py Voyager_2_m05016u_merged.bsp -sc -32 -c Sun -bl Sun Earth Mars Jupiter Saturn
      Uranus Neptune -tl 4500 -co cyan
e) >> bsp2visualCop.py 170509AP_SK_17118_17258.bsp -sc -82 -c Ssaturn -bl Saturn Titan -co cyan
f) >> bsp2visualCop.py 21F31_MEGA_L241010_A300411_LP01_V5_pad_scpse.bsp -sc -159 -c Jupiter  -bl
      Jupiter Europa Io Ganymede -co blue -f j2000

Input Options =>  -sc: spacecraft ID, -c: Center, -f: Frame, -tl: Timeline (days), -bl: Body List, -co: color
```

Figure 5 shows six trajectory examples of past, current, and future missions visualized in Copernicus by using the function calls shown above. The SPK files for these trajectories have been obtained from the NAIF website*. Figure 5(a) and (b) show Lunar Reconnaissance Orbiter's (LRO)† trajectory in inertial and body-fixed frame, respectively. Figure 5(c) shows a transfer from low-Earth orbit (LEO) to a near rectilinear halo orbit (NRHO) at the Moon, similar to Gateway's mission. Figure 5(d) shows Voyager's 2‡ trajectory, (e) Cassini's Grand Finale§, and (f) Europa Clipper's moon tour¶.

For more complex trajectory visualizations, an optional JSON input file can be passed to the function call to generate a more personalized output with specific frames, central body, etc. Figure

*https://naif.jpl.nasa.gov/naif/data_operational.html
†https://lunar.gsfc.nasa.gov/
‡https://solarsystem.nasa.gov/missions/voyager-2/in-depth/
§https://www.nasa.gov/mission_pages/cassini/main/index.html
¶https://europa.nasa.gov/

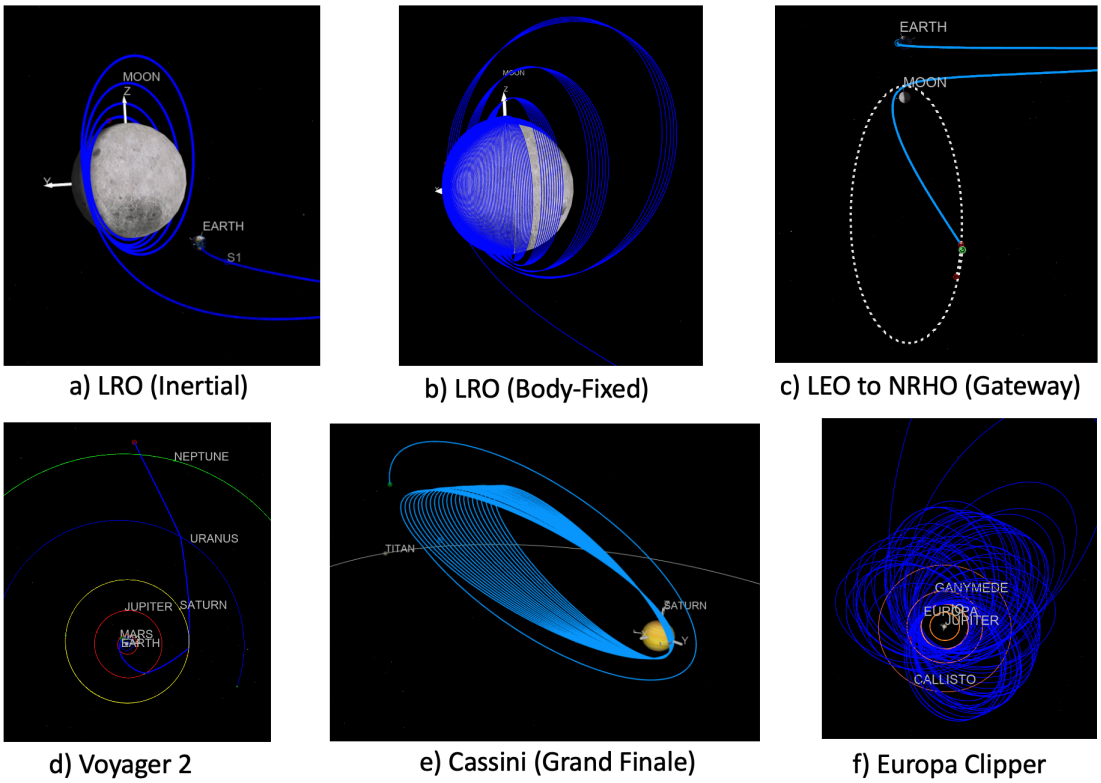


Figure 5. Trajectory visualization of SPK kernel files in Copernicus from several past, current, and future NASA missions.

6(a) shows a Europa Clipper trajectory, where the phases of the mission are color coded. Figure 6(b) shows the JSON input file that was created to generate the trajectory visualization. The function call in this case is the same as above, with an additional input argument,

```
>> bsp2visualCop.py file.bsp -sc ID -c Jupiter -j file.JSON
```

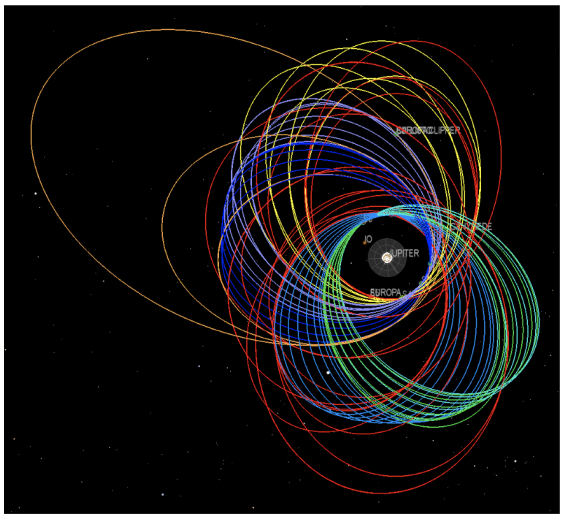
Another complex trajectory visualization is shown in Figure 7, with the JSON input file shown in (c).

TRAJECTORY RECONSTRUCTION IN MONTE: EXAMPLES

In the next section, several examples of trajectory reconstruction in Monte are presented. The trajectories are an orbiter around Enceladus for jet exploration, a moon tour, and a lunar missions.

Trajectory Recovery From Low to High Fidelity Dynamics

The following example demonstrates the use of the *Trajectory Reverse Engineering* strategy for an NRHO trajectory around Enceladus generated using the Circular Restricted Three Body Problem (CR3BP) in Copernicus. The Copernicus solutions serves as initial condition for reconstruction in Monte under a higher fidelity dynamical model. Enceladus is a celestial body known for its complex gravitational environment, and designing trajectories under this environment is particularly

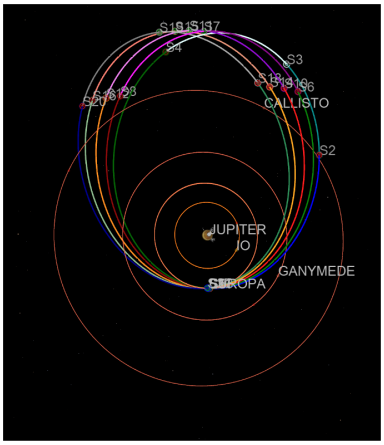


a) Europa Clipper Moon Tour (Jupiter - j2000)

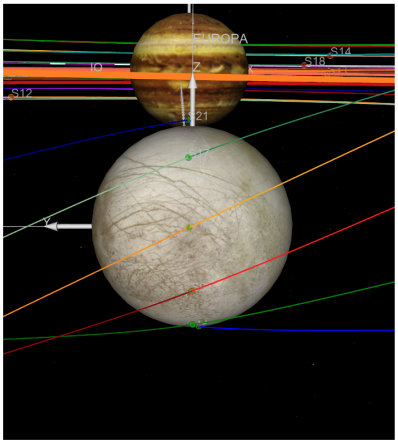
```
{
    "Defaults": {
        "Frame": "j2000",
        "Center": "Jupiter",
        "Ephems" :[
          "../support_files/generic_kernels/spk/satellites/jup310.bsp",
          "../support_files/generic_kernels/spk/planets/de430.bsp",
        ]
    },
    "visCop": {
        "Frame": "iau_body_fixed",
        "Center": "Europa",
        "AuxBody": "Jupiter",
        "BPframe": "EMO2000",
        "scId" : "-159",
        "numSegs" : 1,
        "IniOffset" : 0,
        "timlineInterv" : 0,
        "timeStep" : 1000,
    },
    "Segments": [
          ["Pump-Down",     "10-APR-2030 17:25:42.71",   "orange" ],
          ["COT-1",         "27-MAY-2031 12:58:43.15",   "blue" ],
          ["COT-2",         "23-OCT-2031 21:44:28.82",   "blue" ],
          ["Petal",         "25-APR-2032 23:55:52.67",   "red" ],
          ["TEC2",          "01-SEP-2032 07:18:14.08",   "yellow"],
          ["LE",            "27-MAY-2033 16:03:36.34",   "blue"],
          ["COT-3",         "13-AUG-2033 11:41:59.26",   "blue"],
          ["COT-4",         "15-DEC-2033 07:40:40.21",   "blue"],
          ["COT-5",         "25-APR-2034 16:05:41.28",   "cyan"],
          ["Disposal",      "03-SEP-2034 05:04:32.41" ,  "red"],
          ["Impact",        "12-OCT-2034 01:23:22.13",   "gray"]
    ]
}
```

b) User inputs to search events

Figure 6. Advanced Copernicus visualization of Europa Clipper, color coded by mission phases.



a) Europa Clipper (Jupiter - j2000)



b) Europa Clipper (Europa - body-fixed)

```
{
    "dvSearch": [
        {"centerBody" : "Jupiter",
         "minDV" : 10.0,
         "timeStep" :  100.0,
         "searchType" : "dvDisc",
         "searchFrame" : "j2000",
        }
    ],
    "searchBodies" : [
        {"bodyName" : "Europa",
         "maxAlt" : 10000.0,
         "searchType" : "Peri",
         "searchFrame" : "j2000",
        },
        {"bodyName" : "Jupiter",
         "minAlt" : 10000.0,
         "searchType" : "Apo",
         "searchFrame" : "j2000",
        }
    ]
}
```

c) User inputs to search events

Figure 7. Advanced Copernicus visualization of Europa Clipper, highlighting several latitude flybys in different colors.

challenging. Two cases are considered, the first is an untargeted NRHO, and the second one is an NRHO that targets specific jets at the south pole of the icy moon.

Example: Enceladus NRHO for South Pole Jet Reconnaissance An NRHO is a great candidate as a trajectory to place a polar orbiter around Enceladus with the objective to perform jet reconnaissance, or to target specific jets.[5] These orbits are not trivially designed in full Enceladus dynamics, due to the high sensitivity environment that surrounds this small icy moon, where the non-spherical contributions of both Saturn and the satellite have significant effects on the dynamics. To start the design process, initial conditions for the NRHO with periapsis altitude of 10 km are obtained from

a CR3BP public database*. These initial conditions are loaded in Copernicus, where the Saturn-Enceladus CR3BP dynamical model is set up. The orbit is propagated for 30 days (equivalent to 60 NRHO revs) as shown in Figure 8(a). This trajectory is then saved as an SPK kernel.

To reconstruct this orbit in a high fidelity model, the SPK kernel is loaded into Monte using a full ephemeris model that includes ephemerides from JPL's Horizons system, point-mass gravity from the four Saturnian satellites Titan, Rhea, Dione, and Tethys, as well as Saturn zonal harmonics up to degree 6 and Enceladus zonal harmonics up to degree 3. Notice that if one attempts to reproduce the orbit in the high fidelity model just by re-propagating the initial state of the orbit, the result will be a spacecraft crashing with the icy moon's surface in no longer than a couple of days. The SPK kernel is scanned to identify the osculating apsis of the orbit. Control points and maneuvers are placed at all the osculating apoapsis in order to help with the orbit reconstruction. At the periapsis of the orbit, located near the south pole, control points defined with classical orbital elements are added, and bounds to set up a minimum periapsis altitude $r_{p_{min}} \geq 3$ km are included. The periapsis bounds are included as inequality constraints, as shown in Eq. (5), and together with Eqs. 1–4 are used to define the optimization problem, which is solved under the mutliple shooting forward-backward propagation scheme. The commercial optimizer by preference used in Monte is SNOPT,[6] a general-purpose Sequential Quadratic Programming (SQP) system for constrained optimization involving many variables and constraints. The results of the optimization procedure is shown in Table 1.

$$\mathbf{\Psi} = \begin{bmatrix} r_{p_1} - r_{p_{min}} \\ \vdots \\ r_{p_p} - r_{p_{min}} \end{bmatrix} \geq \mathbf{0} \tag{5}$$

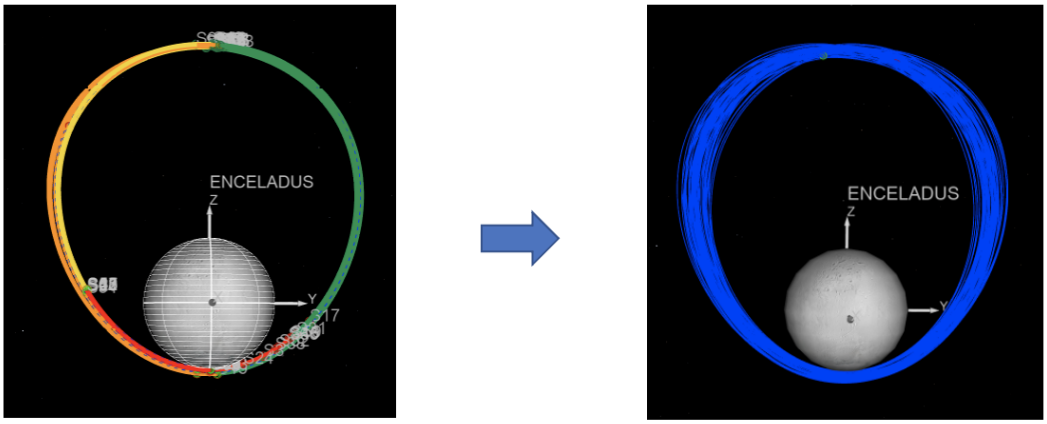
The optimization procedure results in a 30 day controlled trajectory, with a total ΔV$= 15$ m/s. The resulting trajectory is shown in Figure 8 (b). The effect of the full dynamics is observed as a distortion of the periodicity of the orbit, looking more like a quasi-periodic orbit. The effect of the non periodicity of the orbit is also reflected in an increased coverage of the ground track over the south pole region. Figure 9(a) shows the ground track for a 10 days segment of this orbit.

Example: Enceladus Jet Targeting Orbiter An alternative problem to the "free" Enceladus NRHO orbiter, is to design the NRHO to target specific jets from the Tiger Striped region around the south pole. This type of orbit can be useful for sample collection. The general problem is similar to the untargeted case, but additional constraints are imposed to ensure that the orbit flies over the selected jet. Using data from Ref. 7, we have selected a jet called Baghdad043 as the target. Baghdad043 is located close to the natural path of the NRHO (see Figure 9 (b)), with geodetic coordinates of latitude 88.3° and longitude 125.2°. Because the longitude coordinate changes rapidly in the proximity of the south pole, the optimization process can become slow or problematic to converge. To address this issue, we transform the geodetic coordinates into cartesian elements and impose equality constraints with a 400 m tolerance on the x and y coordinates of the jet location, and the altitude of the orbit at this location is fixed to 3 km. To avoid over constraining the problem, an additional maneuver is placed a few hours after each close approach.

The commercial optimizer used The results of the optimization procedure is shown in Table 1.

The *Trajectory Reverse Engineering* process used to reconstruct the two Enceladus NRHO trajectories is outlined in the following diagram:

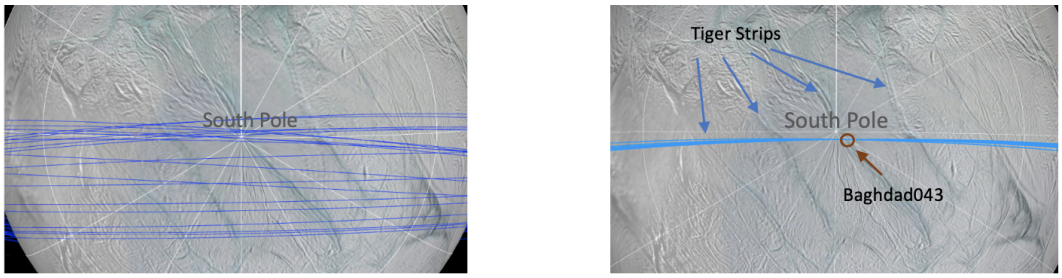
*https://ssd.jpl.nasa.gov/tools/periodic_orbits.html



(a) NRHO: CR3BP, 60-Revs (30 days prop.) - Copernicus

(b) NRHO: High Fidelity, 60-Revs (30 days prop.) -Monte

Figure 8. Reconstruction from low to high fidelity dynamics of an Enceladus NRHO.



(a) NRHO – High Fidelity 20-Revs 10 days prop. 3-50 km Alt. Untargeted (ΔV_total = 5 m/s)

(b) NRHO – High Fidelity 20-Revs 10 days prop. 3 km Alt Jet Targeting (ΔV_total = 20 m/s)

Figure 9. Groundtracks for an south pole NRHO at Enceladus, (a) unargeted and (b) targeted cases.

Trajectory Recovery Process for an Enceladus NRHO orbit:

1: Load I.C. of NRHO into Copernicus
2: Propagate 60 Revs in CR3BP model
3: Create SPK kernel
4: Scan SPK
5: Detect periapsis and apoapsis times
6: Create control points at periapsis. Add ΔV's at apoapsis
7: Optional: Add user defined constraints to the trajectory path (e.g. bounds to the controls)
8: Optional: Add clean up maneuvers a couple of hours after periapsis
9: Solve multiple-shooting optimization procedure

Enceladus Moon Tour

Reaching the icy moons of our Solar System is an expensive task. However, they are currently the hot spot in the search for extraterrestrial life. Implementing a moon tour is in general the more

Case	ΔV_{Total} (m/s)	TOF(days)	Constraints
Copernicus (CR3BP)	0.0	30	N/A
Monte (Untargeted)	15.1	30	3 km $\leq r_{p_i} \leq$ 50km
Monte (Jet targeting)	60.8	30	Fly over Baghdad043 at 3 km alt.

Table 1. ΔVresults for the Enceladus NRHO orbit reconstruction.

fuel-efficient way to place an orbiter around one these satellites. A moon tour is a sequence of flybys used to pump down the energy of the spacecraft until a low ΔVorbit insertion maneuver is possible. To reach Enceladus, several moon tours have been proposed.[8,9] We use here as a reference a moon tour that is designed in Star,[10] a third party flight dynamics tool, in a medium fidelity model.

The *Trajectory Reverse Engineering* strategy was employed to recreate a high fidelity model of a portion of the moon tour using Monte. Specifically, the final phase of the tour, comprising the Enceladus flyby resonant sequence prior to Enceladus Orbit Insertion (EOI), following the Tethys flyby sequence is presented in this study. The Enceladus *M:N* resonance set is [7:6, 15:13, 8:7, 17:15, 9:8, 10:9, 11:10], where *M* and *N* represent the number of revolutions of Enceladus and the spacecraft around Saturn, respectively. The entire Enceladus phase, which includes seven Enceladus flybys (E1, E2, ..., E7) ending with an Enceladus orbit insertion interface (E8) and spans 105.5 days over 68 revolutions around Saturn, is depicted in Figure 10. The SPK reference kernel, provided by the same author who generated the medium fidelity reference tour, was used and reconstructed in Monte without the need of any additional information.

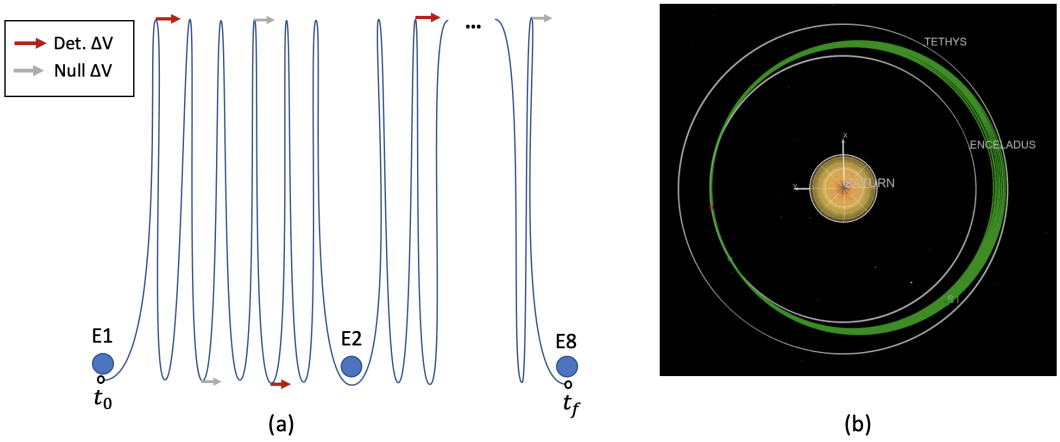


Figure 10. Enceladus moon tour (a) trajectory diagram showing flybys and maneuvers used to reconstruct the trajectory and (b) trajectory visualization in Copernicus.

The reconstruction of the Enceladus tour in Monte starts by loading and scanning the SPK kernel. The first step after loading the kernel is to read initial and final states of the trajectory. Two cases were run, one leaving the bounding states fixed, and a second leaving the final state fixed but free to move in time. Then, any close approaches around Enceladus are detected, where control points are placed in order to have control of the flybys. The state of the close approaches are represented as hyperbolic orbital elements and bounds to the minimum altitude of the flybys are added. The SPK kernel is then scanned in search of impulsive burns in the form of velocity discontinuities. Nine burns were found and added to the reconstruction process. The dynamical environment is set to

include the Sun, Saturn, and the major satellites of the Saturnian system. Saturn and Enceladus are modeled with zonal harmonics up to degree 6 and 3, respectively. Once the defined control states are propagated in the new dynamics, discontinuities between control points (or flybys), ranging from 25 to 6,000 km show up. The forward and backward multiple-shooting algorithm is implemented to close the discontinuities. To help with the convergence of the problem, additional "null" maneuvers are placed every few periapsis and apoapsis of the orbits, as shown in Figure 10(a). A cost function to minimize the ΔV is added to the multiple-shooting algorithm in order to complete the optimization procedure.

Choosing the appropriate step size when searching for velocity discontinuities is key. As was mentioned above, the time step size of the segments should be small enough to not account for velocity changes due to natural accelerations of the system. However, choosing a time step size that is too small could lead to long computation times. By defining a minimum ΔV detection value (ΔV_{min}) for the discontinuity search, an efficient time step size can be chosen that allows for fast searches while excluding velocity changes due to the accelerations of the system. Figure 11 shows the velocity discontinuities for the Enceladus moon tour with a step size of 10 seconds and $\Delta V_{min} = 10$ m/s. In this case, five velocity discontinuities, i.e. impulsive maneuvers, are found. On the other hand, if the step size is reduced to 1 second, together with a $\Delta V_{min} = 3$ m/s, eight impulsive maneuvers are found, as shown in Figure 12. An additional maneuver of approx. 1.8 m/s can be detected by using $\Delta V_{min} = 2$ m/s.

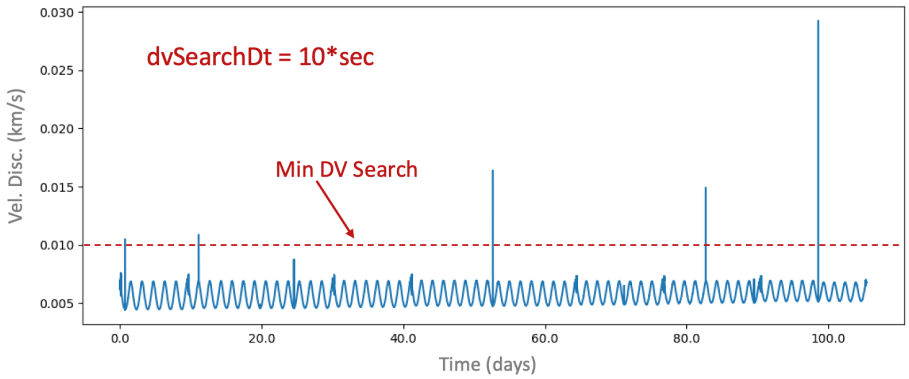


Figure 11. Velocity discontinuities for the Enceladus Moon tour, with δV search step $= 10$ sec.

The final optimization set up to reconstruct the Enceladus moon tour leg consists of seven control points, located at each of the Enceladus flyby, as well as a control point at the Enceladus orbit insertion interface (E8). Hyperbolic orbital elements were used for state parametrization. For each of the intermediate Enceladus encounters (E2, ... , E7), only the minimum altitude of the flyby were constrained. The state and epoch for the beginning of the tour leg (E1) were fixed. The state at E8 was also fixed but two cases where run, one where tf was fixed, and one where it was allows to move. Additionally, nine detected maneuvers, plus fourteen "null" maneuvers were added. The results of the optimization procedure, is shown in table 2.

Low Lunar Orbit to NRHO Gateway Transfer

During the Artemis III mission, one of the phases will involve returning the astronauts from the south pole of the Moon to the Orion capsule at the NRHO Gateway orbit.[11] A representative low-fidelity trajectory was designed in Copernicus, assuming only impulsive burns, and is depicted in

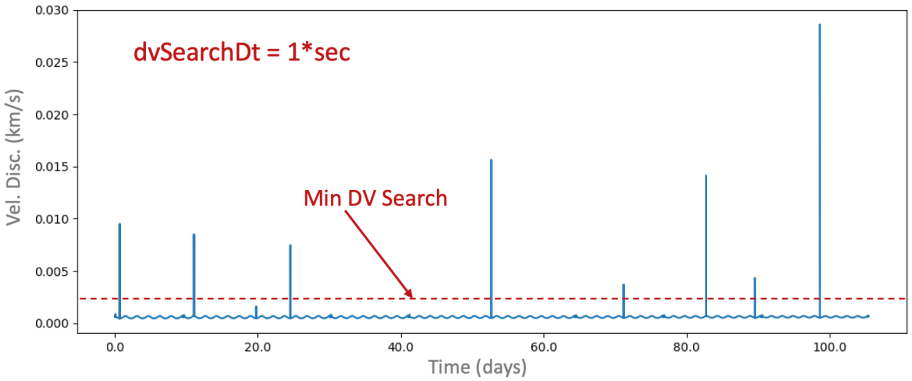


Figure 12. Velocity discontinuities for the Enceladus Moon tour, with δV search step $= 1$ sec.

Case	TOF (days)	EOI V_∞ (m/s)	EOI Alt. (km)	ΔV_{Total} (m/s)
Star (Original)	105.5	386	100	94.3
Monte (Fixed t_0, tf)	105.5	391	107	88.7
Monte (Fixed t_0, Free tf)	105.6	391	107	87.0

Table 2. Optimization results for the Enceladus moon tour reconstruction.

Figure 13. The trajectory begins at the south pole on a low elliptical orbit (post-ascent segment), and a circularizing maneuver is applied at the north pole at apoapsis. After several revolutions in this circular orbit, a maneuver is performed to initiate the transfer to the Lunar NRHO. This transfer segment lasts approximately half a day and culminates in a rendezvous with the Orion capsule at the NRHO orbit. The reference trajectory is saved as a SPK file and transferred to Monte, where a high-fidelity reconstruction is performed without any assumed information from the original Copernicus solution.

In Monte, the trajectory is loaded as an SPK kernel and scanned to identify the locations of the maneuvers. A δv search step of 10 seconds is used, and the velocity discontinuity over time is shown in Figure 14. The constant changes in velocity of approximately 15 m/s during the low lunar orbit segments (the first 7 hours of the orbit) are due to the rapid rotation of the low lunar orbit. Hence, a minimum ΔV detection of 20 m/s is used. The three maneuvers of the total transfer are evident in the velocity discontinuity plot. These discontinuities are used to locate and compute impulsive burns for the transfer. Control points are inserted at each location of the maneuvers, and the orbit is set to start with a fixed state at t_0 and a fixed target state at apoapsis of the NRHO. In total, 5 control points with 4 discontinuity constraints and 3 maneuvers are included, and the trajectory is optimized to obtain a minimum fuel continuous transfer. Table 3 shows the results of the optimization procedure, and the values of each maneuver for both the original and the recovered orbit.

FINITE BURN DETECTION AND TRANSCRIPTION INTO IMPULSIVE BURNS

Finite burns can be approximated as impulse burns by calculating the energy of the orbit before and after thrusting. This can be done by approximating the energy of the orbit using its osculating Keplerian semi-major axis with respect to a central body. Using this approximation, the finite burn

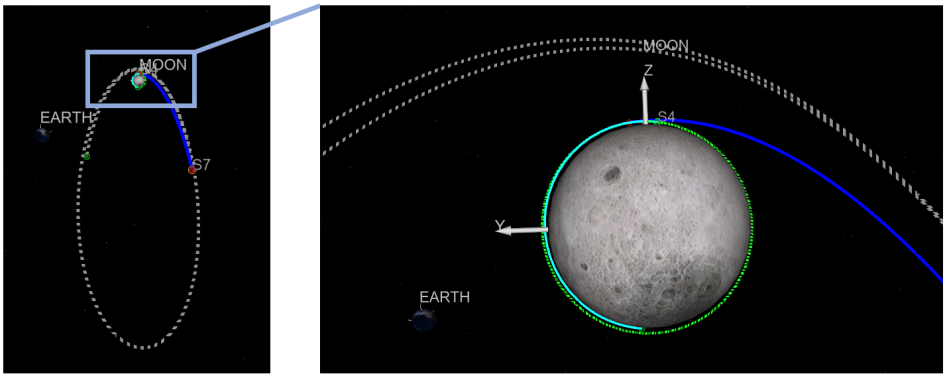


Figure 13. LLO to NRHO Gateway trasnfer trajectory.

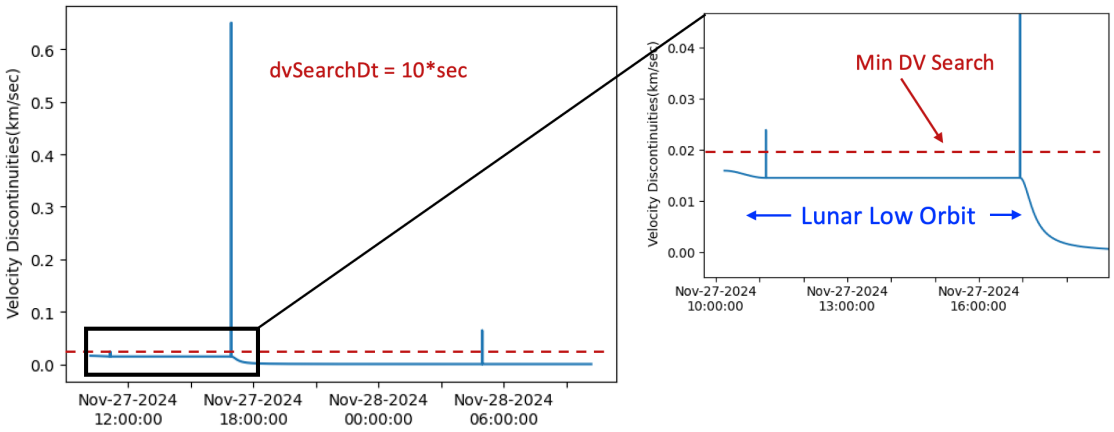


Figure 14. Velocity discontinuities for the LLO to NRHO Gateway trasnfer, with a ΔV step $=$ 10 sec, and minimum ΔV search of 10 m$/$s.

can be expressed in terms of an impulse burn, which is a good approximation for reproducing the original trajectory.

The energy of the orbit can be approximated by using the vis-via equation and the osculating orbital elements of the orbit before and after the burn. The vis-viva equation is a mathematical expression that relates the kinetic and potential energy of a celestial body in its orbit around another body. It is given by:

$$\frac{v^2}{2} - \frac{\mu}{r} = \frac{\mu}{2a} \tag{6}$$

where v is the inertial velocity of the spacecraft with respect to the central body, r is the distance of

Tool	ΔV1 (m/s)	ΔV2 (m/s)	ΔV3 (m/s)	ΔV_{Total} (m/s)
Copernicus (original)	19.1	650.6	64.4	734.1
Monte (reconstructed)	18.8	650.4	63.9	733.2

Table 3. Optimization results of the LLO to NRHO transfer.

the spacecraft, μ is the gravitational parameter of the central body, and a is the semi-major axis of the orbit.

The finite burn approximation into an impulsive burn is governed by

$$|\Delta V| \cong \sqrt{\mu\left(\frac{2}{r(t_i)} - \frac{1}{a_{\text{-pre}}}\right)} - \sqrt{\mu\left(\frac{2}{r(t_i)} - \frac{1}{a_{\text{-post}}}\right)} \tag{7}$$

where $a_{\text{-pre}}$ and $a_{\text{-post}}$ are the semi-major axis before and after the burn, and t_i is the epoch of the burn at the half-way point. As a first approximation, the scanned finite burn can be implemented as an impulsive burn. For future work we plan to implement a detailed example of a trajectory recovery process which contains several finite burns.

CONCLUSIONS

Trajectory Reverse Engineering is proposed here as a means to recover any desired trajectory that has been designed in a completely different flight dynamics tool. We propose a trajectory-sharing process via SPK kernels, which are agnostic to the models used. A judicious kernel scan is simply used to extract necessary information to recover the original trajectory, while a multiple-shooting implementation is used to close discontinuities along the trajectory, emerging from numerical and dynamical discrepancies between the original and recovery tool.

Examples of transfer solutions between the NASA tools Monte and Copernicus are presented. Copernicus is particularly powerful for visualization purposes and preliminary design. Monte on the other hand is incredibly powerful for high-fidelity design. We show how to use Copernicus to visualize any trajectory contained in an SPK kernel, regardless of the tool that was used to generate the trajectory, or obtained directly from the JPL Horizon database. We also show how to recover trajectories from SPK kernels using Monte, including a complex Enceladus moon tour example, a transfer from a Lunar Low Orbit to the NRHO Gateway, and the reconstruction in a high fidelity model of an Eceladus NRHO, originated from a CR3BP model.

The strategy presented, provides the foundations for a global flight mechanics tools interoperability system.

ACKNOWLEDGMENT

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, and sponsored by the Flight Mechanics Technical Fellow under the NASA Engineering & Safety Center (NESC), as part of the Flight Mechanics Analysis Tools Interoperability and Component Sharing project. © 2022. California Institute of Technology. Government sponsorship acknowledged.

REFERENCES

[1] A. V. Rao, "Trajectory optimization: a survey," *Optimization and optimal control in automotive systems*, pp. 3–21, Springer, 2014.

[2] S. Hernandez, "Cassini Maneuver Experience Through the Last Icy Satellite Targeted Flybys of the Mission," AAS/AIAA *Space Flight Mechanics Meeting, AAS 16-243*, Aug. 2016.

[3] S. Ardalan, J. Bordi, N. Bradley, D. Farnocchia, Y. Takahashi, and P. Thompson, "Juno Orbit Determination Experience During First Year at Jupiter," AAS/AIAA *Astrodynamics Specialist Conf., AAS 17-595*, 08 2017.

[4] *Improvements to the Copernicus Trajectory Design and Optimization System for Complex Space Trajectories*, Tech. Rep. NASA/TM-2019-220247, National Aeronautics and Space Administration, Jan. 2019.

[5] J. Blanchard, M. W. Lo, B. Anderson, R. L. Restrepo, and D. Landau, "Using NRHO Invariant Funnels to Target Enceladus South Pole," AAS/AIAA *Space Flight Mechanics Meeting, AAS 23-124*, Austin, TX, January 2023.

[6] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: an SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006, 10.1137/S1052623499350013.

[7] P. Helfenstein and C. C. Porco, "ENCELADUS' GEYSERS: RELATION TO GEOLOGICAL FEATURES," *The Astronomical Journal*, Vol. 150, aug 2015, p. 96, 10.1088/0004-6256/150/3/96.

[8] S. Campagnola, N. J. Strange, and R. P. Russell, "A fast tour design method using non-tangent v-infinity leveraging transfer," *Celestial Mechanics and Dynamical Astronomy*, Vol. 108, No. 12, 2010, pp. 165–186.

[9] N. J. Strange, S. Campagnola, and R. P. Russell, "A fast tour design method using non-tangent v-infinity leveraging transfer," *Advances in the Astronautical Sciences*, Vol. 135, No. 3, 2009, p. 2207–2225.

[10] D. Landau, "Efficient Maneuver Placement for Automated Trajectory Design," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 7, 2018, pp. 1531–1541, 10.2514/1.G003172.

[11] G. L. Condon, C. A. Ocampo, L. Burke, C. C. Esty, C. Berry, B. Mahajan, and S. P. Downs, *Mission and Trajectory Design Considerations for a Human Lunar Mission Originating from a Near Rectilinear Halo Orbit*, 10.2514/6.2020-1921.