(Preprint) AAS 12-236

RECENT IMPROVEMENTS TO THE COPERNICUS TRAJECTORY DESIGN AND OPTIMIZATION SYSTEM

Jacob Williams [*], Juan S. Senent [†], Cesar Ocampo [‡], and David E. Lee [§]

Copernicus is a software tool for spacecraft trajectory design and optimization. The latest version (v3.0.1) was released in October 2011. It is available at no cost to NASA centers, government contractors, and organizations with a contractual affiliation with NASA. This paper is a brief overview of the recent development history of Copernicus. An overview of the evolution of the software and a discussion of significant new features and improvements is given, and how the tool is used to design spacecraft missions.

INTRODUCTION

Copernicus, a generalized spacecraft trajectory design and optimization system,[1,2,3] is capable of solving a wide range of trajectory problems such as planet or moon centered trajectories, libration point trajectories, planet-moon transfers and tours, and all types of interplanetary and asteroid/comet missions. Impulsive and finite burn (low to high thrust) propulsion systems based on chemical, solar electric, or nuclear powered engines can be modeled. The *segment*[1] is the fundamental building block of a multiple-shooting based trajectory design process in Copernicus. A mission can be defined by any number of segments, which can represent multiple spacecraft, multiple stages of a single spacecraft, or can simply be computational segments used to obtain information about selected states in the mission (for example to compute orbital elements or altitude). Segments can be independent, connected via inheritance in complicated ways, or initially disconnected and constrained to be connected during the optimization process. They can contain impulsive Δv maneuvers, finite burn maneuvers, and user-defined force models. Additionally, segment definition parameters such as times, initial mass, mass discontinuities due to staging, initial state, engine parameters, impulsive maneuver parameters and finite burn control law histories[2] can all be optimization variables. Each segment can have a different integration or propagation method and segments can be propagated either forward or backwards in time. The system includes a variety of parameter optimization and targeting methods.[3]

The Copernicus Project started at the University of Texas at Austin in August 2001. In June 2002, a grant from the NASA 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 (ISP) Program[4] and from the Flight Dynamics Vehicle Branch of Goddard Spaceflight

[*]ERC, Inc. (Engineering and Science Contract Group), Houston, Texas
[†]Odyssey Space Research, Houston, Texas.
[‡]Department of Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin, Austin, Texas.
[§]EG/Aeroscience and Flight Mechanics Division, NASA Johnson Space Center, Houston, TX 77058.

Center. The first operational version was completed in March 2006 (v1.0). 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 Flight Mechanics and Trajectory Design Branch of JSC.

Copernicus was used extensively at several NASA centers as part of mission design and analysis work for the Constellation Program.[5,6] During this period (2006-2010), the developers of Copernicus used the tool extensively at JSC for analysis of manned lunar missions, and many new features not available in the original tool were added in order to enable specific analysis tasks. These included: a batch processing capability to allow for many instances of Copernicus to run simultaneously in order to conduct large trade studies, the ability to use lower-fidelity models for trade studies, and the ability to easily transition from low-fidelity to high-fidelity models, more capability to interact with and exchange data with other software tools (input and output data), and various usability improvements (e.g., improving the GUI to make the tool easier to use, making it easier to manipulate segments, etc.). In addition, during this period, many internal code improvements were made in order to make the program easier to upgrade, the run-time efficiency of the program was improved, and a library of low-level routines known as the Copernicus Toolkit was created to allow the ability to build other tools using the same code-base.[7]

In 2010 and 2011, the NASA In-Space Propulsion Technology Program also funded specific Copernicus development tasks. These included new finite burn engine models, more flexible data output, gravity assist maneuvers, a Δv to finite burn conversion wizard, and new capabilities for defining halo orbits about libration points. The new finite burn models included a BB/DD model for engine efficiency vs. I_{sp}, a mass flow rate and thrust vs. power polynomial model, a custom solar power decay model, a non-propulsion systems reserved power model, and shadow models for SEP engines. These model enabled more realistic simulation of real-world in-space propulsion systems.

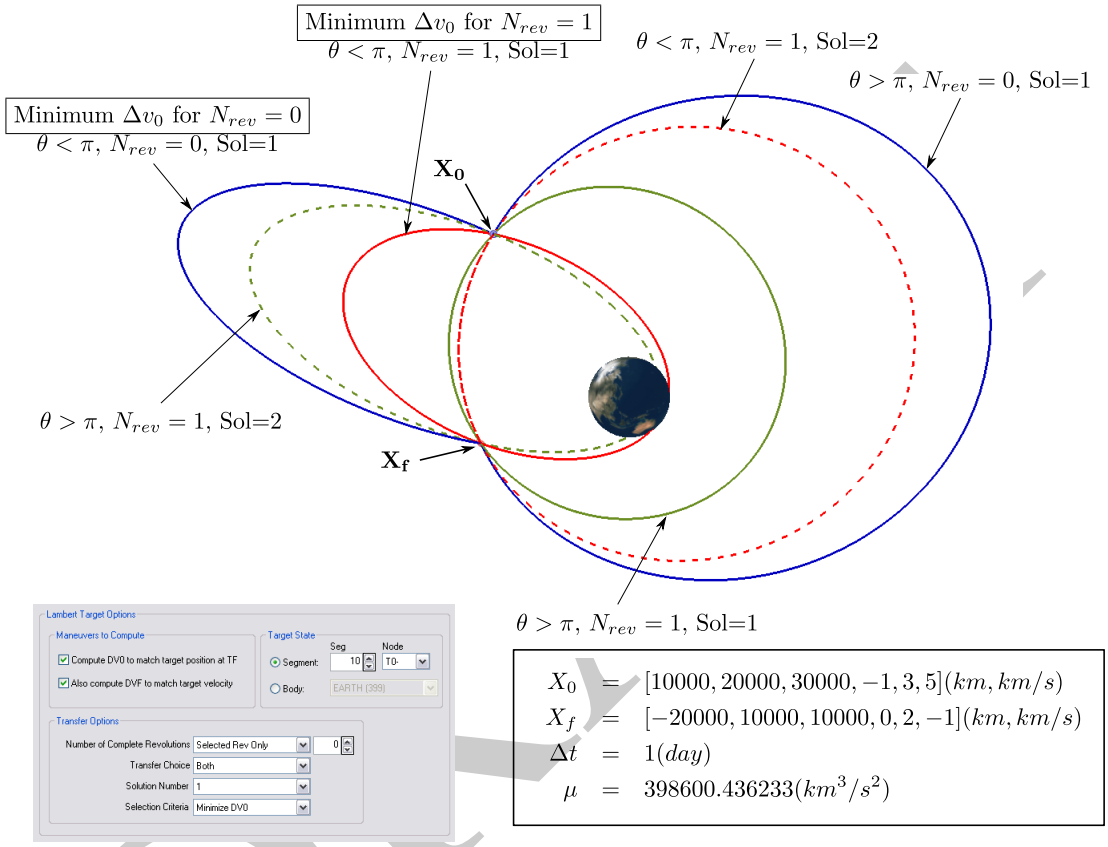
The following sections describe specific features and recent improvements to Copernicus.

IMPULSIVE MANEUVER UPGRADES

Several new options have been added to allow more flexibility in the use of Δv maneuvers in Copernicus. They are described below.

Lambert Maneuvers

Optimization of impulsive Δv maneuvers was present in Copernicus from the beginning. A recent addition was the ability to also specify a Δv in order to target another state or body using Lambert's algorithm. The Gooding Lambert routines[8] are used for this. They allow multiple transfer options when computing a Lambert transfer, including multiple revolutions and the choice of transfers in the $> \pi$ or $< \pi$ directions. Figure 1 shows an example. The transfer options are all available in the Copernicus GUI. The "selection criteria" is a Copernicus feature that allows the automatic selection among a set of selected transfer solutions in order to minimize or maximize the individual maneuvers (or the sum of the two). For example: target a state in a single revolution or less, using either a $> \pi$ or $< \pi$ transfer, selecting the one which minimizes the initial maneuver. Lambert maneuver can also be combined with gravity assist parameters described below to design gravity tours (described further below).



$$
\begin{aligned}
X_0 &= [10000, 20000, 30000, -1, 3, 5](km, km/s) \\
X_f &= [-20000, 10000, 10000, 0, 2, -1](km, km/s) \\
\Delta t &= 1(day) \\
\mu &= 398600.436233(km^3/s^2)
\end{aligned}
$$

Figure 1. Multiple Solution in Lambert's Problem. In the Copernicus implementation, all the solutions are available, including multi-rev solutions. The user indicates which solution to use (and can specify to use the one which maximizes or minimizes the initial or final maneuver, or the sum of the two maneuvers.

Gravity Assist Δv Maneuver Parameterization

A Δv maneuver can be used to approximate a high-thrust maneuver by a spacecraft, but can also be used to approximate a gravity assist flyby of a planetary body (i.e., assuming a patched-conic approximation). In Copernicus, this feature was added by allowing an impulsive Δv maneuver to be parameterized as a zero-sphere-of-influence (ZSOI) planetary flyby maneuver. This allows a planetary flyby to be modeled as an instantaneous change in velocity at the body,[9] using either the Δv_0 or the Δv_f of a Copernicus segment. The new ZSOI parameters (r_p, ϕ, Δv_∞, and Δv_{r_p}) are described in Table 1. Maneuvers parallel to the velocity vector at periapsis (Δv_{R_p}) or infinity (Δv_∞) can also be included in the parameterization. These represent maneuvers performed by the spacecraft, and result in the incoming and outgoing $\mathbf{v}_\infty$ vectors having different magnitudes.

For the patched-conic approximation, the $\mathbf{v}_\infty$ vectors (before and after the flyby) are obtained from the Copernicus segment nodes t_0 or t_f (as shown in Figure 2). It is assumed that the node position is at the center of the flyby body. In general, this can be achieved as an initial condition (if the ZSOI maneuver is at t_0), imposed as a constraint, or as the result of a Lambert maneuver targeting the flyby body. The segment $\Delta\mathbf{v}$ is $\Delta\mathbf{v}_{ZSOI}$, the difference in the $\mathbf{v}_\infty$ vectors from the "Heliocentric" point of view:

$$\Delta\mathbf{v}_{ZSOI} = \mathbf{v}_\infty^+ - \mathbf{v}_\infty^- \tag{1}$$

For a pure flyby (unpowered), the specified clock angle (ϕ) and hyperbolic turning angle (δ) are used to rotate the $\mathbf{v}_\infty^-$ to produce the $\mathbf{v}_\infty^+$ vector, as shown in Figure 3. The turning angle is simply:

$$\delta = 2\sin^{-1}\left(\frac{1}{r_p v_\infty^2/\mu + 1}\right) \tag{2}$$

Where r_p is the periapsis of the hyperbola, μ is the gravitational parameter of the flyby body, and v_∞ is the magnitude of $\mathbf{v}_\infty^-$ or $\mathbf{v}_\infty^+$ (which are equal for a pure flyby).

Maneuvers parallel to the velocity vector at periapsis (Δv_{R_p}) or infinity (Δv_∞) can also be included in the parameterization. These represent maneuvers performed by the spacecraft, and result in the incoming and outgoing $\mathbf{v}_\infty$ vectors having different magnitudes. If using Δv_∞, this magnitude is simply added to the $\mathbf{v}_\infty^+$ after the rotation is performed (refer to Figure 3). If using Δv_{r_p}, then this velocity is added to the velocity at periapsis.

The direct version of the transformation takes $\mathbf{v}_\infty^-$, r_p, ϕ, (Δv_∞, or Δv_{r_p}) and computes $\mathbf{v}_\infty^+$. The inverse transformation takes $\mathbf{v}_\infty^-$ and $\mathbf{v}_\infty^+$ and computes r_p, ϕ, (Δv_∞, or Δv_{r_p}). The inverse transformation is used when the maneuver is also a Lambert maneuver (see below). In this case, $\Delta\mathbf{v}_{ZSOI}$ is determined by the Lambert algorithm, and the gravity assist parameters are not independent quantities.

Table 1. New ZSOI Δv Maneuver Parameters

Symbol	Description
r_p	Periapsis radius magnitude
ϕ	Clock angle (angle between $\hat{\mathbf{Z}}$ and $\mathbf{B}$ vectors)
Δv_∞	Velocity magnitude change at infinity (i.e., the sphere of influence)
Δv_{r_p}	Velocity magnitude change at periapsis of the hyperbola

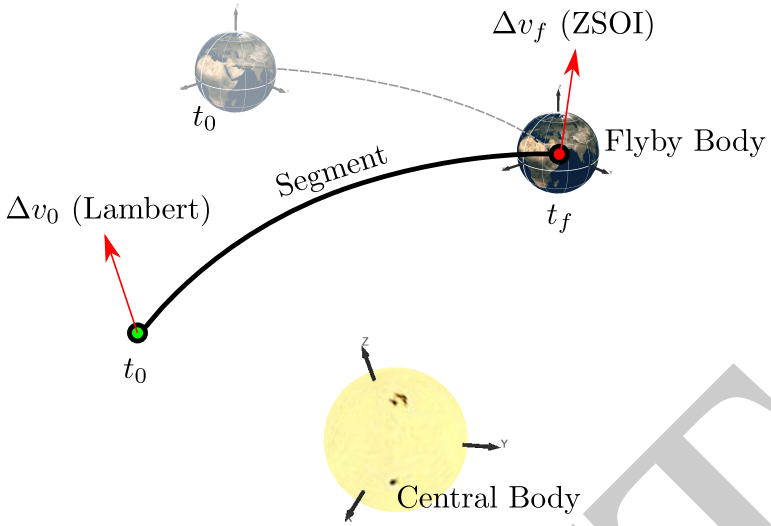


Figure 2. ZSOI Maneuver Parameterization (Heliocentric View). A Δv from a Copernicus segment which occurs at the center of a body (in this case, Earth), can be parameterized as a ZSOI flyby maneuver.

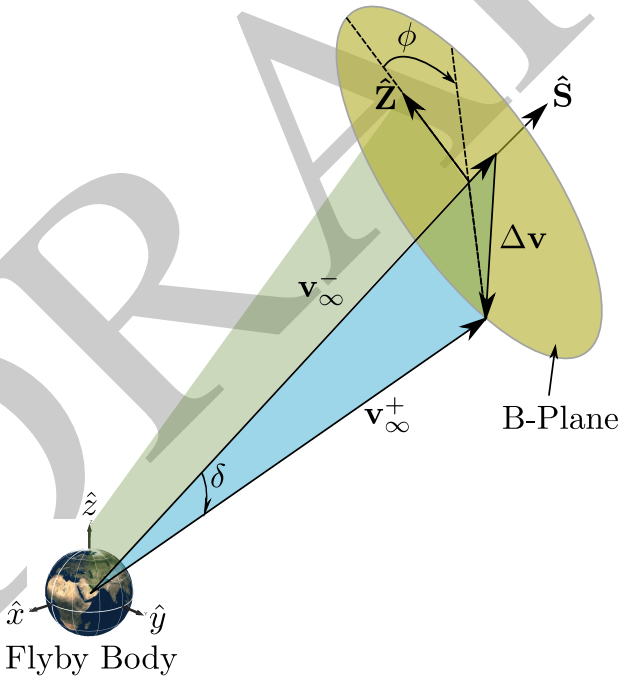


Figure 3. ZSOI Maneuver Parameterization (Planetocentric View). For an unpowered flyby ($\Delta v_{r_p} = 0$), the clock angle (ϕ) and the hyperbolic turning angle (δ) are used to rotate the $\mathbf{v}_\infty^-$ to produce the $\mathbf{v}_\infty^+$ vector. The difference between the two $\mathbf{v}_\infty$ vectors is transformed into the heliocentric frame, and becomes the segment Δv.

Combining Lambert and Gravity Assist

The Lambert and ZSOI gravity assist Δv features can be used together to specify a Lambert maneuver that is also a gravity assist. The gravity assist parameters are then computed from the Lambert Δv, for the specified flyby central body. An example use of this new capability is the ability to create a patched-conic trajectory where a powered planetary flyby targets a specified endpoint (e.g., another body). For example, in the mission concept shown in Figure 11b, the Δv_{ZSOI} could be a Lambert maneuver which targets the center of the destination body (Mars). When a segment Δv maneuver is both a ZSOI flyby and a Lambert maneuver, then the Lambert Δv (computed from the central body, the segment duration, and the target state) is transformed into the ZSOI parameterization. If the ZSOI parameterization uses Δv_{r_p}, then this represents the component of the gravity assist performed as a powered flyby maneuver at periapsis. During the optimization, this component can be constrained (e.g., an upper bound could be imposed, or it could be driven to zero to produce a pure gravity assist). If converting the maneuver to a hyperbolic segment (as in Figure 11c), the powered component is inserted as a maneuver at periapsis of the inserted segment. This feature is explained below.

Conversion of a Gravity Assist Δv Maneuver to a Hyperbola

A new feature was also added to automatically convert a Δv maneuver parameterized as a ZSOI gravity assist into a hyperbolic mission segment at the flyby body. This feature inserts two segments into the mission: one propagated forward in time and the other backwards in time. The initial state is a hyperbola centered at the flyby body, with a true anomaly of zero, and a periapsis radius and clock angle equal to the values from the ZSOI Δv parameterization. To complete the hyperbolic state, the v_∞ vector will be equal to the planet-relative velocity in the original segment. If a non-zero Δv_∞ or Δv_{r_p} is used, then this maneuver will also be added to the inserted segment. The new segments are then propagated to the sphere of influence of the flyby body.

This feature enables the user to optimize a trajectory using patched-conic Δv's, and then easily convert the Δv's to realistic planetary flybys. An example in shown in Figure 4. In this figure, the dashed black line is a ZSOI flyby trajectory with only the Sun in the force model. A Δv is applied at the center of Jupiter, and is parameterized as a ZSOI maneuver. The solid blue line is the corresponding hyperbolic segments (with Jupiter in the force model). A zoomed-out view is also shown of the sphere-of-influence radius. Once the flyby segments are inserted, the user can then connect them to the rest of the mission via an optimization or targeting process.

Δv to Finite Burn Conversion

One of the goals of Copernicus is to provide a user-friendly way to design complex trajectories. The use of "wizards" that can be used to automatically solve common sub-problems is one manifestation of this objective. The first such wizard added to Copernicus allows a user to easily convert an impulsive Δv maneuver into an equivalent finite burn maneuver, and then insert the finite burn segment into the mission (as shown in Figure 5). This process is accomplished by using the Δv maneuver as an initial guess for a finite burn maneuver,[10] which is then optimized to close the constraints and maximize the post-maneuver mass. The initial guess for the finite burn duration (Δt_{FB}) is computed using the rocket equation:

$$\Delta t_{FB} = \frac{m_0 c}{T}\left(1 - e^{-\Delta v/c}\right) \tag{3}$$

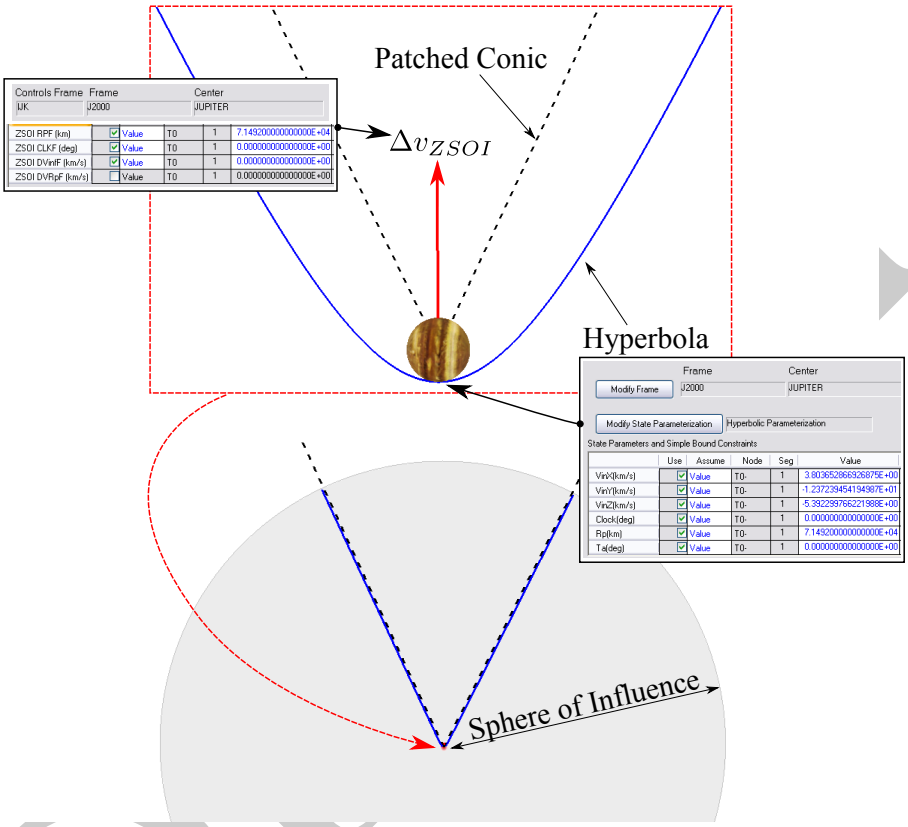


Figure 4. Conversion of a patched-conic trajectory into a hyperbola. The patched-conic trajectory includes the Sun only in the force model. An impulsive Δv is applied at the center of Jupiter. The hyperbolic trajectory includes Jupiter in the force model. The zoomed-out view shows the sphere-of-influence radius.

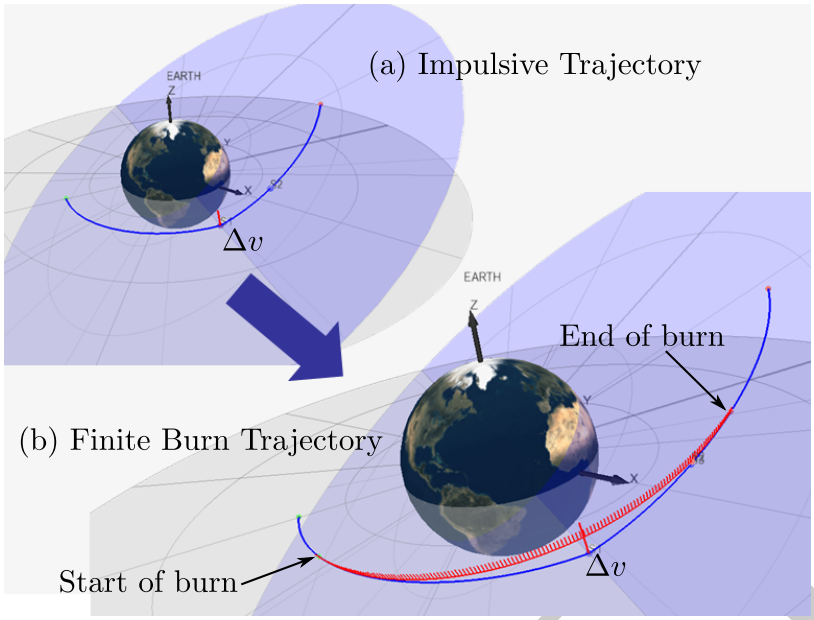


Figure 5. Conversion of a Δv to a Finite Burn. The finite burn trajectory begins and ends on the original impulsive trajectory.

where c is the engine exhaust velocity ($I_{sp}g_0$), T is the engine thrust, and m_0 is the initial mass. The start time guess for the finite burn is $t_0 = t_{\Delta v} - \Delta t_{FB}/2$. If $\mathbf{u}$ is the finite burn thrust direction unit vector at $t_{\Delta v}$ (the time of the Δv maneuver), then the initial guesses for $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$ are:

$$\mathbf{u} = \Delta\hat{\mathbf{v}} \tag{4}$$

$$\dot{\mathbf{u}} = \mathbf{0} \tag{5}$$

$$\ddot{\mathbf{u}} = (3\mu/r^5)\left[(\mathbf{r}\cdot\mathbf{u})\mathbf{r} - (\mathbf{r}\cdot\mathbf{u})^2\mathbf{u}\right] \tag{6}$$

where $\mathbf{r}$ is the position vector and μ is the gravitational constant of the central body. The equation for $\ddot{\mathbf{u}}$ assumes a central body force field only. The finite burn control law is assumed to be:

$$\alpha(t) = \alpha_0 + \dot{\alpha}_0(t - t_0) + \frac{1}{2}\ddot{\alpha}_0(t - t_0)^2 \tag{7}$$

$$\beta(t) = \beta_0 + \dot{\beta}_0(t - t_0) + \frac{1}{2}\ddot{\beta}_0(t - t_0)^2 \tag{8}$$

where $\alpha(t)$ and $\beta(t)$ are the thrust direction right ascension and declination control history (α_0, $\dot{\alpha}_0$, $\ddot{\alpha}_0$, β_0, $\dot{\beta}_0$, $\ddot{\beta}_0$ are constant parameters to be optimized). The vectors $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$ (defined at $t_{\Delta v}$) are converted to $\alpha(t_{\Delta v})$, $\dot{\alpha}(t_{\Delta v})$, $\ddot{\alpha}(t_{\Delta v})$, $\beta(t_{\Delta v})$, $\dot{\beta}(t_{\Delta v})$, and $\ddot{\beta}(t_{\Delta v})$, then propagated to t_0 (the start of the finite burn) to provide an initial guess for α_0, $\dot{\alpha}_0$, $\ddot{\alpha}_0$, β_0, $\dot{\beta}_0$, and $\ddot{\beta}_0$.

This feature provides an easy way to convert between a impulsive Δv and a feasible finite burn, using any of the engine models available in Copernicus. This enables a faster lower-fidelity solution to be computed, and then converted into feasible solution using realistic engine models. Once this is done, the trajectory can then be further refined if desired (i.e., optimization of the thrust history and/or engine parameters). This is another feature which facilitates a design process of producing high-fidelity trajectories by starting with simple models which can be converged quickly.

EXPANDED STATE PARAMETERIZATIONS

A state parameterization is a set of (up to six) variables, called state parameters, that can be used to represent the state of a spacecraft in a specified reference frame at a specified epoch. The most common state parameterizations are cartesian (position and velocity), and classical orbital elements. However, there are many different ways of parameterizing any state vector.[11] In a generalized trajectory design and optimization system, it is convenient to have multiple ways to specify, inherit, optimize, and constrain states, in order to provide maximum flexibility in the design process. New state parameterizations have been added to Copernicus in recent revisions, including the Gooding universal element set[12] (which exhibits very low loss of precision during transformations to and from position and velocity), and modified equinoctial elements[13,14] (which eliminates the singularities present in the classical orbital element set).

In addition, thorough unit testing of the state transformation subroutines enables the accuracy of each of the parameterizations to be studied, and the best algorithms identified for performing the transformations. Rigorous testing also enables the identification of numerical issues and singularities. For example, for the computation of geodetic altitude and latitude in the geographic parameterization, the original iterative algorithm was replaced with the closed-form Heikkinen algorithm,[15] which was found to exhibit better numerical accuracy.

In general, the state parameterizations in Copernicus can be used to represent any state vector. Two of the specialized parameters are described below, which do not apply to all states, but have specific applications to certain classes of states.

Hyperbolic State Parameterization

The hyperbolic state parameterization, only valid for hyperbolic states, contains the most parameters of any state parameter set in Copernicus. This parameterization has many uses when dealing with hyperbolic orbits[6] (for example, to guarantee that a state is always hyperbolic, while still allowing some of the orbital parameters to vary during optimization). A recent revision expanded this parameterization to include many new state parameters, including B-Plane parameters.[16,17,18] For a hyperbolic flyby, the B-plane is the plane perpendicular to the $\mathbf{v}_\infty$ vector, and centered at the flyby body (as shown in Figure 6). In addition, to be completely general, Copernicus includes two B-plane definitions (the standard one, and one referenced to the v_∞^+ vector which can be used for backward integration). The new hyperbolic state parameters (ϕ, θ, B, $\mathbf{B}\cdot\mathbf{T}$, $\mathbf{B}\cdot\mathbf{R}$, δ, and $r^\pm$) are described in Table 2, where:

Table 2. New Hyperbolic State Parameters

Symbol	Description
ϕ	Clock angle (angle between $\hat{\mathbf{Z}}$ and $\mathbf{B}$ vectors)
θ	Aim point orientation (angle between $\hat{\mathbf{T}}$ and $\mathbf{B}$ vectors)
B	Magnitude of the $\mathbf{B}$ vector
$\mathbf{B}\cdot\mathbf{T}$	Dot product of $\mathbf{B}$ and $\hat{\mathbf{T}}$ vectors
$\mathbf{B}\cdot\mathbf{R}$	Dot product of $\mathbf{B}$ and $\hat{\mathbf{R}}$ vectors
δ	Hyperbolic turning angle (angle between $\mathbf{v}_\infty^-$ and $\mathbf{v}_\infty^+$ vectors)
$r^\pm$	Radius magnitude (> 0 past periapsis, and < 0 before periapsis).

- $\mathbf{B}$ is the vector from the planet to the point where the asymptote of the hyperbolic flyby trajectory pierces the B-plane.
- $\hat{\mathbf{S}}$ is a vector parallel to the $\mathbf{v}_\infty$ vector (and perpendicular to the B-plane).
- $\hat{\mathbf{Z}}$ is the projection of the z-axis of the selected coordinate frame into the B-plane. In Copernicus, any frame can be used for this purpose (for example, the ecliptic frame or the body-fixed frame of the flyby body).
- $\hat{\mathbf{R}}$ is the negative of $\hat{\mathbf{Z}}$.
- $\hat{\mathbf{T}}$ is the cross product of $\hat{\mathbf{S}}$ and $\hat{\mathbf{Z}}$.

An example use of the new parameters is shown in Figure 7, which uses the $\mathbf{v}_\infty^-$, ϕ, δ, and $r^\pm$ state parameters to define an Earth-centered hyperbolic state. Also note that the hyperbolic state parameterization works together with the ZSOI Δv maneuver parameterization, enabling the conversion of a ZSOI Δv maneuver into a hyperbolic segment.

Halo Orbit State Parameterization

In the latest revision of Copernicus, a new state parameterization was added that can be used to easily define halo orbits. First, the user needs to select a two-body rotating frame (Earth-Moon, Sun-Earth, etc.) with its center at one of the collinear libration points: L_1, L_2 or L_3. Second, the amplitude of the halo orbit has to be defined. The user can specify the amplitude in the z or x-axis. This amplitude is just a 1^{st} order approximation of the actual amplitude of the halo orbit. When defining a halo orbit, the selected amplitude (A_x or A_z) has to satisfy certain constraints.[19] If the specified amplitude does not generate a halo orbit, a pop-up error dialog will be generated. In this dialog, the minimum required amplitude will be shown. Third, the halo orbit family: North or South (see Figure 9) should be selected and finally the location of the initial state on the halo orbit is defined by the phase parameter τ (see Figure **??**). τ increases in the direction of motion.

With the previous information, Copernicus can now calculate the state on the halo orbit. The user has two options for the accuracy of this initial state: Richardson's 3^{rd} order approximation[19,20] and differentially corrected with 1 or 2 revolutions of accuracy. Since the Richardson's approximation requires less computational time, it can be used for quick scans where not much accuracy is required (this approximation is valid for solutions with accuracy less than half a revolution). If more accuracy is needed (e.g. the halo orbit has to be propagated for 1 or 2 revolutions), a differentially corrected with 1 or 2 revolutions of accuracy can be used. The Richardson's solution is used as an initial guess and then a differentially corrected solution is obtained.[21] The accuracy of these solutions is 1 km after one or two revolutions of the orbit. If there is no convergence in the differential correction method, a warning message is generated in the console window. This message will also contain some information about the accuracy of the obtained solution (e.g. the number of half-revs. the solution satisfy the accuracy criteria). The computational time associated with the differential correction method is higher than one of the Richardson's approximation. Thefore, this option might slow down the solution of the whole optimization problem that Copernicus is trying to solve.

Although this new feature makes it simple to define states on halo orbits, the differential correction method might not converge for all the feasible amplitudes. For those cases, the user can define a differential correction scheme on his own by using the Copernicus optimization variables and constraints.

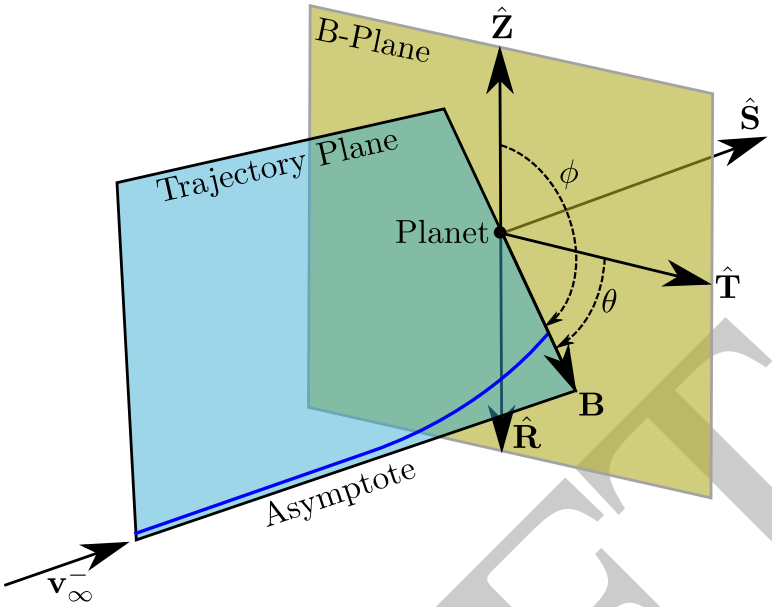


Figure 6. B-Plane Parameters. The B-plane is perpendicular to the $\mathbf{v}_\infty$ vector, and centered at the flyby body. The clock angle ϕ is the angle between $\hat{\mathbf{Z}}$ and B, and the aim point orientation (θ) is the angle between $\hat{\mathbf{T}}$ and B.

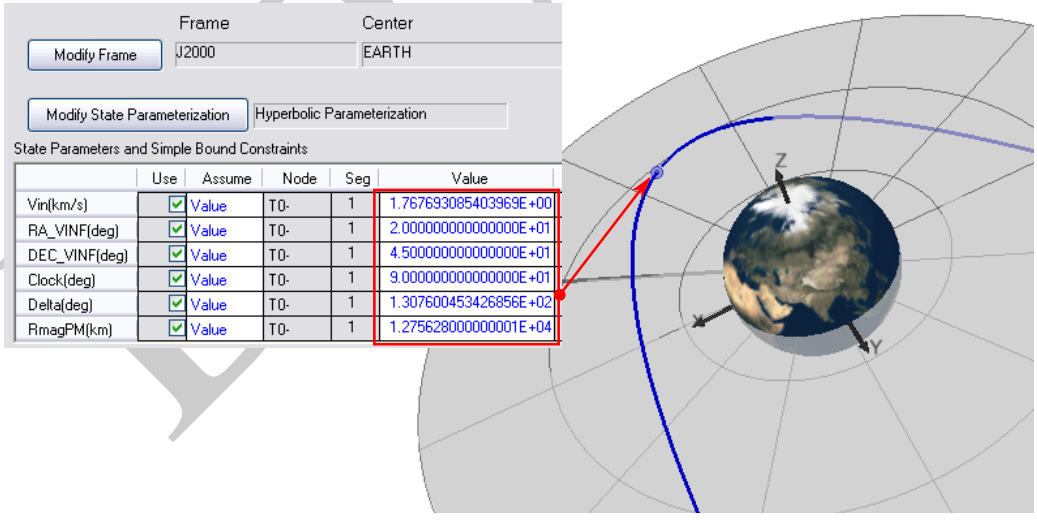


Figure 7. Example Hyperbolic State. This example uses the $\mathbf{v}_\infty^-$, ϕ, δ, and $r^\pm$ state parameters to define an Earth-centered hyperbolic state.

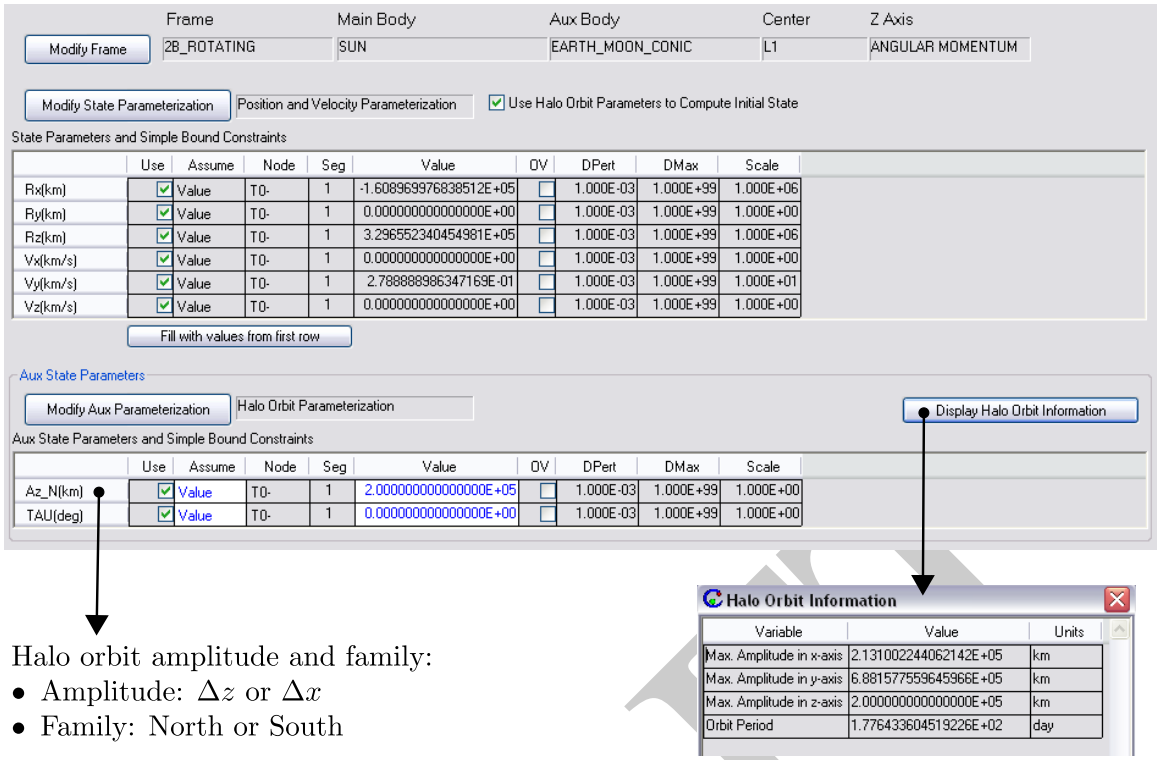


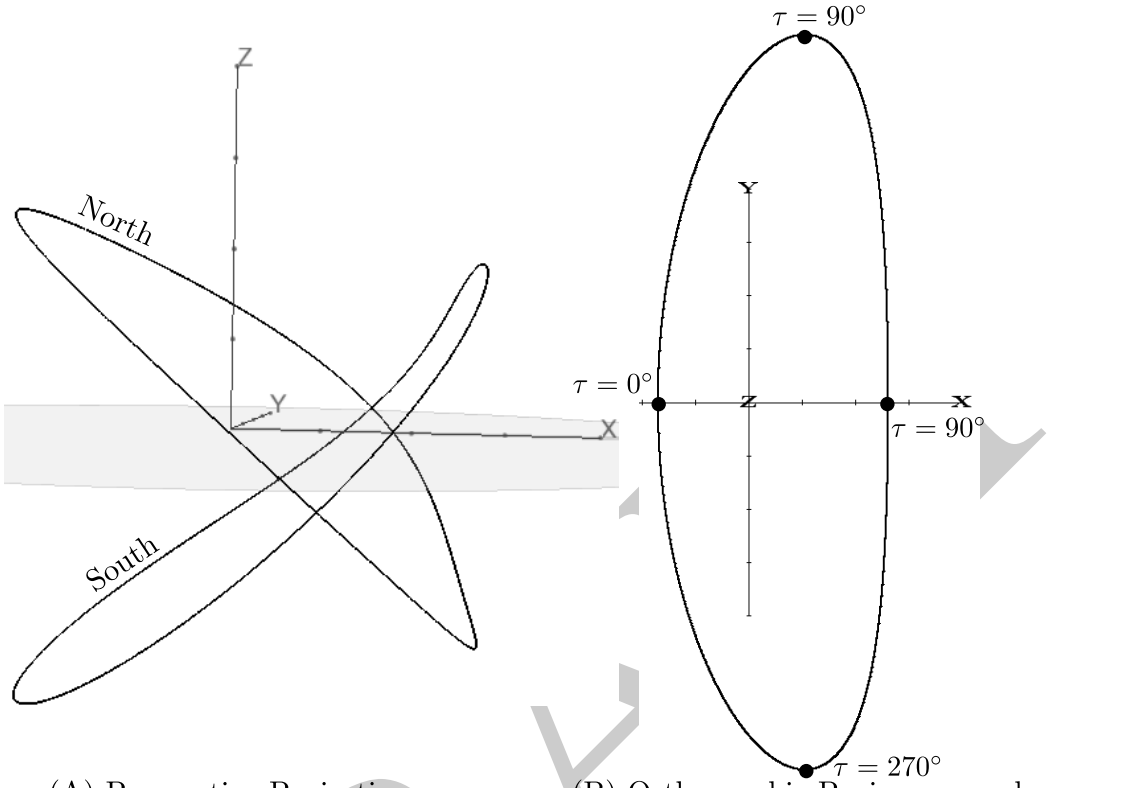
Figure 8. Graphical user interface components used to define halo orbits.

PROPAGATION METHODS

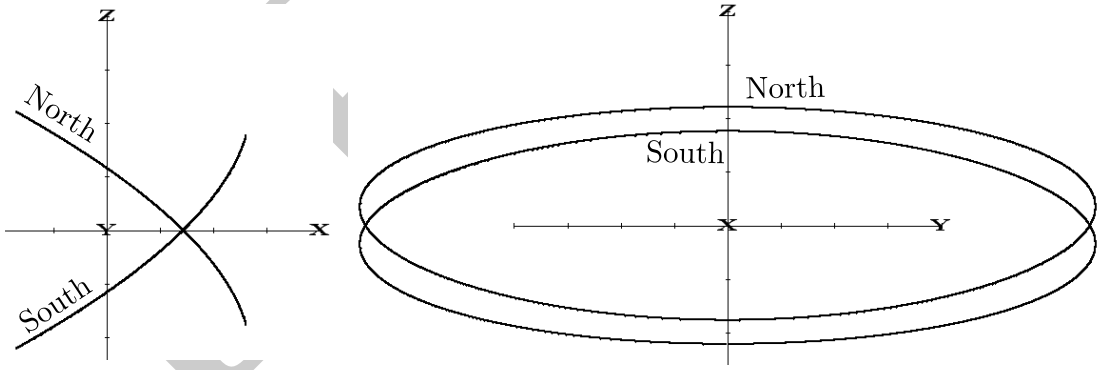
Each segment in Copernicus can have its own propagation method (or they can all use the same one). One recent upgrade to Copernicus was the inclusion of Kepler propagation methods (early versions only included explicit integration). Kepler methods available in Copernicus include the Battin,[22] Goodyear,[23] and Shepperd[24] methods. Kepler propagation methods can be used for central-body patched conic trajectories. Can also switch to high-fidelity integration. Copernicus also includes a suite of both fixed and variable-step explicit integration methods. These include Runge-Kutta, Adams, Encke, Nystrom, and extrapolation methods.

DIFFERENTIAL EVOLUTION

Differential evolution,[25] a population based stochastic optimization method, has been added to Copernicus. DE has proven useful in the design of interplanetary missions.[26] The inclusion of DE into Copernicus further expands the ability of the tool to solve complicated trajectory optimization problem within a single software tool. Within Copernicus, the user also has the ability to switch between DE and gradient-based methods (e.g., SQP). DE can be used to search the solution space to produce a set of population members which can then be refined using another method. DE is particularly useful for situations with discontinuities which would cause gradient-based optimization methods to fail (for example, switching between a single and multi-revolution Lambert solution).



(A) Perspective Projection (B) Orthographic Proj. on x-y plane



(C) Orthographic Proj. on x-z plane (D) Orthographic Proj. on y-z plane

Figure 9. Different views of the north and south halo orbit families.

EXAMPLE MISSION DESIGN

Copernicus contains a set of building blocks with which the user can use to solve a wide range of trajectory design and optimization problems. The flexibility of the Copernicus approach facilitates the modeling of many types of trajectory optimization problems as well as the generation of a methodology required to solve them. Though there are multiple ways in which any trajectory optimization problem can be modeled, there are some that will have better convergence properties than others for a given choice of solution method. Copernicus is not a "black box" to solve problems with little or no input from the user, nor is it simply an algorithm, but a tool to assist the user in designing and solving the problem. Using Copernicus is a creative process, requiring the user to decide how many segments are needed to model the trajectory, how to parameterize the states and maneuvers, which algorithms to use, etc. The integrated graphical user interface (GUI) and 3D graphics also provide continuous feedback during solution process.

All of these building blocks can be combined to design very complicated missions. The same problem can be solved using different levels of complexity and fidelity as needed within the same program. The user can start with simplified models and can gradually build a more complex and realistic solution to the problem. Some examples of this include:

- Solving a trajectory problem using simplified force models (e.g., pointmass gravity), and then activating a higher-fidelity force model (e.g., atmospheric drag, high-order gravity, or solar radiation pressure) and resolving.
- A Δv maneuver can first be estimated using Lambert targeting and then added to the optimization problem.
- Gravity assists can be modeled using a zero sphere of influence patched conic model and then converted to actual planetary flybys.
- A complex trajectory problem can be solved using optimized Δv maneuvers, which are then converted to finite burn maneuvers.
- Differential Evolution and patched conic trajectories can be be used to quickly search for favorable solutions, which can then be refined using higher-fidelity models and gradient-based optimization methods.

Copernicus allows a gradual mission design approach where simplified models can be used for initial scans or trade studies and then, using the same tool, the user can start adding more realistic features to the optimization problem or new phases to the mission, etc (Figure 11 shows an example of this process).

Show some example missions and how they are designed using Copernicus.

Example: Flyby sequence using lambert maneuvers and ZSOI parameterization.

Using the Copernicus Toolkit to build other programs. Parametric studies. GTOC5 solution: Toolkit + Copernicus.

See Figure 11 for a mission design concept that includes the use of the new Δv to finite burn and gravity assist to hyperbola conversion features. These new features are additional building blocks that can be used as part of a trajectory design and optimization process that begins with an

infeasible initial guess using simplified models and ends with a converged high-fidelity solution. A description of this kind of step-by-step mission design (from simple to complex models) is given in References.[3,27]

In our GTOC-5[28] solution (shown in Figure 10), a combinatorial search program was developed using the Toolkit. This allowed the same models to be used for the search as the final trajectory. The final trajectory with finite burns was optimized in Copernicus, using the impulsive solution as an initial guess.

SOFTWARE ARCHITECTURE

Upgrades using object-oriented features of Fortran 2003/2008. Would like to talk some about the software architecture

Just some notes...

Copernicus is written in the Fortran programming language.

CATO (Computer Algorithm for Trajectory Optimization) a Fortran 90 trajectory optimization program from the 1990s that included object-based programming concepts.[29,30] At the time, a full set of object-oriented language features was not available in Fortran. The original version of Copernicus included some Fortran 77 code, along with more modern Fortran 90/95 (for example, a Copernicus segment was implemented as a Fortran `TYPE`). With the advent of Fortran 2003 (and later Fortran 2008), more object-oriented features became available. The first object-oriented feature in Copernicus was the upgraded data output feature in 2010, which was implemented as a polymorphic array.

FUTURE WORK

Planned features, and/or stuff that would be nice to have... the logical evolution of the tool that we have now...

Collocation segments. Any segment would have the ability to be a collocation segment. The collocation variables and constraints would automatically be added to the optimization problem. Segments could be switched back and forth between collocation and explicit integration, etc.

Plug-in architecture for User-defined guidance/control laws and constraints. Relative motion, rendezvous, and formation flying.

Additional state parameterizations for other types of periodic and quasi-periodic orbits (such as Lypunov and Lissajous orbits).

More GUI “wizards” to automate complicated tasks.

More object-oriented internal upgrades. New features will follow object-oriented design principles using Fortran 2003/2008. Older code will be upgraded as necessary.

ACKNOWLEDGMENT

This work was funded by the NASA Constellation and Orion projects, and the NASA In-Space Propulsion Technology Program. The authors would like to acknowledge Jerry Condon (NASA/JSC) and John Dankanich (Gray Research Inc.) for their support of Copernicus development.

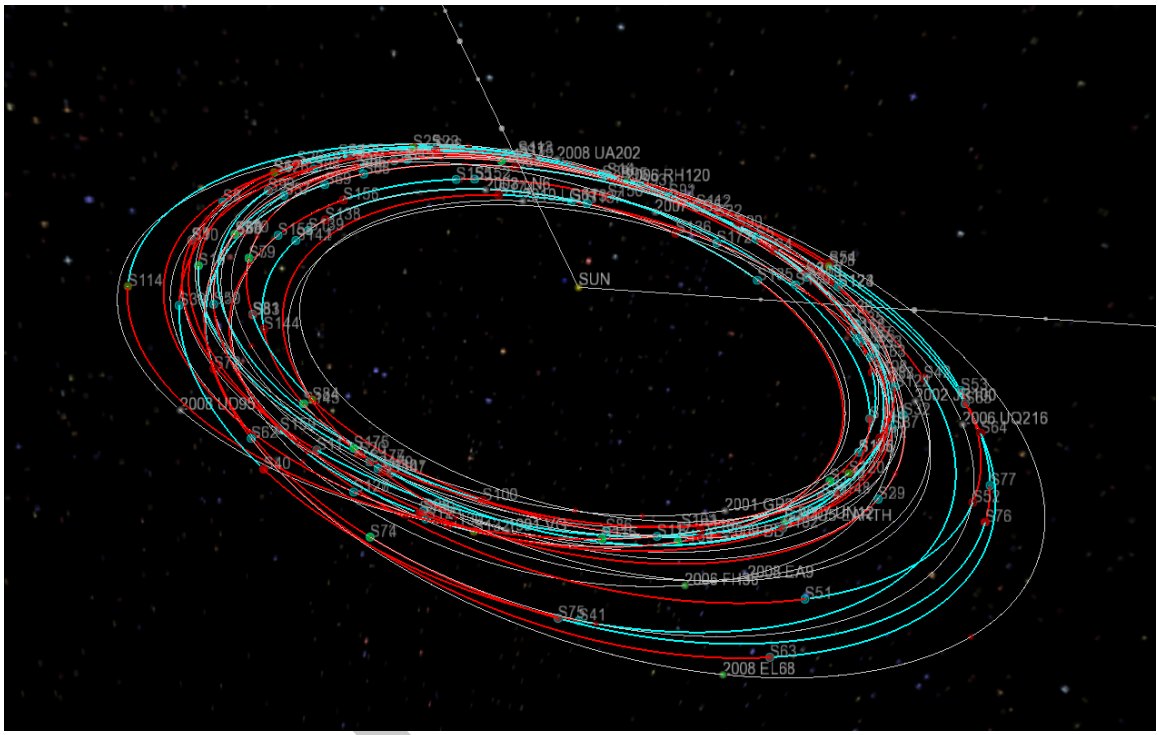
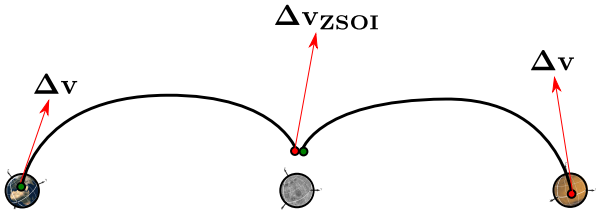
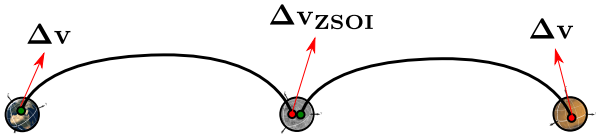


Figure 10. Copernicus Screenshot of a 15-Asteroid Solution to the GTOC-5 Problem. The initial impulsive guess was generated using a program built using the Copernicus Toolkit. The finite burn solution was optimized in Copernicus. In this figure, the cyan arcs are coast periods and the red arcs are thrust periods.

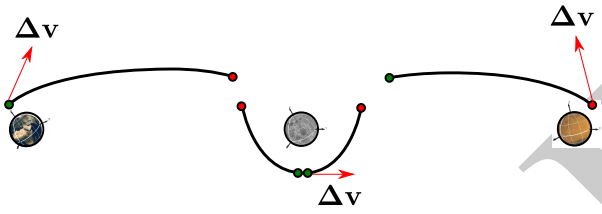


(a) Impulsive, infeasible, ZSOI force model

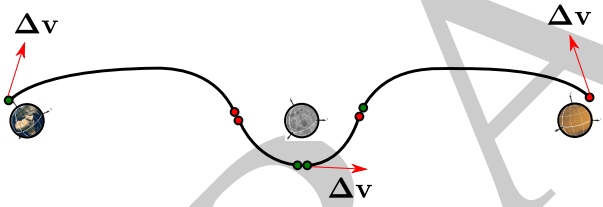


(b) Impulsive, optimal, ZSOI force model

Δv_{ZSOI} to Hyperbola

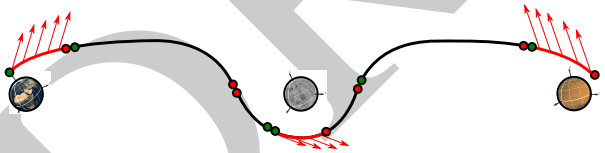


(c) Impulsive, infeasible, high-fidelity force model

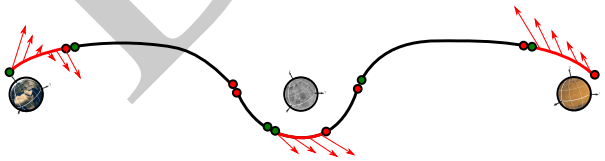


(d) Impulsive, optimal, high-fidelity force model

Δv to Finite Burn



(e) Finite burn, feasible, high-fidelity force model



(f) Finite burn, optimal, high-fidelity force model

Figure 11. Copernicus Mission Design (adapted from[27]). This example shows a mission that escapes one body, performs a powered flyby of a second body, and ends up captured by a third body. The process begins with an infeasible initial guess using simplified models and ends with a converged high-fidelity solution.

REFERENCES

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

[2] C. Ocampo, "Finite Burn Maneuver Modeling for a Generalized Spacecraft Trajectory Design and Optimization System," *Annals of the New York Academy of Science*, Vol. 1017, May 2004, pp. 210–233.

[3] J. Williams, J. S. Senent, C. Ocampo, R. Mathur, and E. C. Davis, "Overview and Software Architecture of the Copernicus Trajectory Design and Optimization System," *4th International Conference on Astrodynamics Tools and Techniques*, May 2010. Madrid, Spain.

[4] L. D. Kos, T. P. Polsgrove, R. C. Hopkins, D. Thomas, and J. A. Sims, "Overview of the Development for a Suite of Low-Thrust Trajectory Analysis Tools," *AIAA/AAS Astrodynamics Specialist Conference*, August 21-24 2006. AIAA 2006-6743.

[5] M. Garn, M. Qu, J. Chrone, P. Su, and C. Karlgaard, "NASA's Planned Return to the Moon: Global Access and Anytime Return Requirement Implications on the Lunar Orbit Insertion Burns," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, August 2008.

[6] J. Williams, E. C. Davis, D. E. Lee, G. L. Condon, T. F. Dawn, and M. Qu, "Global Performance Characterization of the Three Burn Trans-Earth Injection Maneuver Sequence over the Lunar Nodal Cycle," *AAS/AIAA Astrodynamics Specialist Conference*, August 2009.

[7] J. Williams, S. M. Stewart, G. L. Condon, D. E. Lee, E. C. Davis, J. S. Senent, and T. F. Dawn, "The Mission Assessment Post Processor (MAPP): A New Tool for Performance Evaluation of Human Lunar Missions," *Proceedings of the 20th AAS/AIAA Space Flight Mechanics Meeting*, February 14-17 2010. AAS 10-191.

[8] R. Gooding, "A Procedure for the Solution of Lambert's Orbital Boundry Value Problem," *Celestial Mechanics and Dynamical Astronomy*, Vol. 48, 1990, pp. 145–165.

[9] J. E. Prussing and B. A. Conway, *Orbital Mechanics*. Oxford University Press, 1993.

[10] C. A. Ocampo, "Exact Impulsive to Time Optimal Finite Burn Trajectory Automation," Unpublished Manuscript, The University of Texas at Austin.

[11] G. R. Hintz, "Survey of Orbit Element Sets," *Journal of Guidance, Control, and Dynamics*, Vol. 31, May-June 2008.

[12] R. H. Gooding, "On Universal Elements, and Conversion Procedures to and from Position and Velocity," *Celestial Mechanics*, Vol. 44, 1988.

[13] P. J. Cefola, "Equinoctial Orbit Elements - Application to Artificial Satellite Orbits," *AIAA/AAS Astrodynamics Specialist Conference*, September 1972.

[14] M. J. H. Walker, B. Ireland, and J. Owens, "A Set of Modified Equinoctial Orbit Elements," *Celestial Mechanics*, Vol. 36, August 1985.

[15] M. Heikkinen, "Geschlossene formeln zur berechnung raumlicher geodatischer koordinaten aus rechtwinkligen koordinaten," *Z. Ermess.*, Vol. 107, 1982.

[16] W. Kizner, "A Method of Describing Miss Distances for Lunar and Interplanetary Trajectories," External Publication No. 674, Jet Propulsion Laboratory, August 1959.

[17] W. Kizner, "Some Orbital Elements Useful in Space Trajectory Calculations," Technical Report No. 34-84, Jet Propulsion Laboratory, July 1960.

[18] A. B. Sergeyevsky, G. C. Snyder, and R. A. Cunniff, "Interplanetary Mission Design Handbook, Volume 1, Part 2: Earth to Mars Ballistic Opportunities, 1990-2005," JPL Publication 82-43, NASA Jet Propulsion Laboratory, September 1983.

[19] D. L. Richardson, "Analytic Construction of Periodic Orbits About the Collinear Points," *Celestial Mechanics*, Vol. 22, 1980, pp. 241–253.

[20] D. L. Richardson, "Halo Orbit Formulation for the ISEE-3 Mission," *Journal of Guidance and Control*, Vol. 6, 1980, pp. 543–548.

[21] R. Thurman and P. A. Worfolk, "The Geometry of Halo Orbits in the Circular Restricted Three-Body Problem," tech. rep., The Geometry Center, 1996.

[22] B. Uzzell, "Derivation and Documentation of the New Battin Conic Subroutines," JSC Internal Note No. 77-FM-3, NASA Johnson Space Center, Jan 1977.

[23] A. Klumpp, "Performance Comparison of Lambert and Kepler Algorithms," Interoffice Memorandum 314.1-0426-ARK, NASA Jet Propulsion Laboratory, January 1991.

[24] S. W. Shepperd, "Universal Keplerian State Transition Matrix," *Celestial Mechanics*, Vol. 35, 1985.

[25] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[26] A. D. Olds, C. A. Kluever, and M. L. Cupples, "Interplanetary Mission Design Using Differential Evolution," *Journal of Spacecraft and Rockets*, Vol. 44, September - October 2007.

[27] C. A. Ocampo and D. V. Byrnes, *Mission Design and Trajectory Optimization*. John Wiley & Sons, Ltd, 2010.

[28] "5th Global Trajectory Optimisation Competition," `http://mech.math.msu.su/gtoc5/`.

[29] D. V. Byrnes and L. E. Bright, "Design of High-Accuracy Multiple Flyby Trajectories Using Constrained Optimization," *AIAA/AAS Astrodynamics Specialist Conference*, August 1995.

[30] J. N. Hatfield, "CATO (Computer Algorithm for Trajectory Optimization): an implementation of Fortran 95 object-based programming," *SIGPLAN Fortran Forum*, Vol. 22, April 2003, pp. 2–7.