

An Architecture for a Generalized Spacecraft Trajectory Design and Optimization System

Cesar Ocampo

Assistant Professor,

Department of Aerospace Engineering and Engineering Mechanics,
The University of Texas at Austin, Austin, Texas, USA 78712

Abstract

The elements of a general high precision system for trajectory design and optimization for single or multiple spacecraft using one or more distinct propulsion systems, and operating in any gravitational environment within the solar system are discussed. The system architecture attempts to consolidate most all spacecraft trajectory design and optimization problems by using a single framework that requires solutions to either a system of nonlinear equations or a parameter optimization problem with general equality and/or inequality constraints. The use of multiple reference frames that generally translate, rotate, and pulsate between two arbitrary celestial bodies facilitates the analysis of multiple celestial body force field trajectories such as those associated with libration point missions, cycling trajectories between any set of celestial bodies, or any other type of trajectory or mission requiring the use of multiple celestial bodies. A basic trajectory building block, referred to as the *basic segment*, that can accommodate impulsive maneuvers, maneuver and non-maneuver based mass discontinuities, and finite burn or finite control acceleration maneuvers, is used to construct single or multiple spacecraft trajectories. The system architecture facilitates the modeling and optimization of a large range of problems ranging from single spacecraft trajectory design around a single celestial body to complex missions using multiple spacecraft, multiple propulsion systems, and operating in multiple celestial body force fields.

1 Introduction

Spacecraft trajectory optimization is a field that has received considerable attention over the last several decades. The field continues to evolve as a result of innovations in trajectory dynamics associated with spacecraft utilizing the simultaneous gravitational attraction of two or more celestial bodies where the contribution from any of the celestial bodies influences significantly the motion of the spacecraft (Ref.[1]). Further advancements have been made in both the analytical and numerical based solution methods required to solve these types of problems. The most recent developments in purely analytical solutions to low thrust orbit transfer problems have been given by Azimov and Bishop[2] and a recent numerical based solution method capable of solving complex low thrust multiple body gravity trajectories has recently been documented by Whiffen and Sims[3].

Fundamental results in trajectory optimization are based in part on original research due to Lawden[4]. Lawden introduced the Primer Vector and its use in the optimization of space trajectories is known as Primer Vector Theory. The Primer Vector and its applications form an integral part of the current system but the specific details associated with the optimization of both impulsive and finite burn maneuvers are not discussed in this article.

Associated with the theory of optimal space trajectories, are the numerical methods required to solve spacecraft trajectory problems in force fields where even closed form solutions of the uncontrolled trajectories are not available. A comprehensive survey on the computational issues applied specifically to the spacecraft trajectory optimization problem is given by Betts[5]. A clear exposition on the conversion of optimal control problems into sub-optimal parameter optimization problems whose solutions require nonlinear programming is given by Hull[6].

The focus of the current article is to describe a general approach that draws upon pertinent aspects of trajectory design and optimization theory. The effort is an attempt to present a single framework in which one or more spacecraft operating in a force field environment under the mutual attraction of one or more celestial bodies, and using one or more propulsion systems can be analyzed and solved efficiently. The framework for this architecture is currently implemented in a prototype trajectory design and optimization system called COPERNICUS that is under development at the University of Texas.

The term ‘architecture’ in the title of this article refers to the structure and methodology of the system. The architecture is defined by its basic components which include the force models, the coordinate frames, the numerical methods, and the method used to model the trajectories. The term ‘generalized’ implies that the system is designed to handle many classes of problems involving common gravitational force fields and acceleration models, different sets of boundary conditions, and various types of propulsion systems without resorting to specific procedures or algorithms to solve each type of problem separately with unique and distinct methods. Common in the spacecraft trajectory optimization literature is the description and implementation of specific methods and algorithms to solve specific problems. Though this is entirely valid, the current architecture attempts to unify the approach to these same problems under a single framework. The terms ‘trajectory design’ refer to the process of generating nominal and feasible solutions that satisfy a predetermined set of constraints and boundary conditions without considering the optimization of any aspect of such a solution. The term ‘optimization’ refers to the process of generating a trajectory design solution that extremizes some general scalar quantity of the solution, regardless of how complex it may be, provided that it can be determined or computed deterministically.

In a spacecraft trajectory optimization problem, the cost functions typically considered include, minimization of the total impulse required, minimization of the total transfer times, minimization of propellant used, or maximization of the final spacecraft mass. However, other allowable cost functions may include the minimization of the hyperbolic excess velocity relative to an arrival or flyby celestial body or the minimization of the value of the Jacobi constant of a spacecraft in a circular restricted three body system. Important here is that the cost function should be allowed to take on any value that is of interest to a mission, provided that it can be uniquely computed from the variables and system parameters used to model the problem.

The current prototype of the system uses explicit numerical integration for state propagation since the force fields encountered in these problems are in general nonlinear and non-autonomous. The solutions to the different classes of problems are obtained as solutions to either a system of nonlinear equa-

tions or to a nonlinear constrained parameter optimization problem. Efficient gradient based nonlinear root finding algorithms and sequential quadratic programming algorithms are used to obtain these solutions.

All spacecraft trajectory problems can be formulated as either multi-point boundary value problems or as nonlinear programming problems that explicitly extremize single or multi-objective cost functions. In the case of a multi-point boundary value problem formulation, two cases exist. First, feasible trajectory solutions with no implicit optimization satisfy only the boundary conditions associated with the kinematics or the physical variables of the problem such as the conditions related to the physical state of the spacecraft or the physical parameters of the force model. A simple example of this includes the central body, orbital two point boundary value problem that satisfies Lambert's theorem (Ref.[7]). Here, a nonlinear search is required for the three components of the velocity vector that connects two distinct position vectors in a given flight time. However, there exist other types of targeting involving more complex boundary conditions that are general functions of the state variables. An example of one of these is to find the launch injection conditions from any given Earth centered parking orbit so that the trajectory terminates in a captured orbit about the Moon. This is the classic ballistic lunar capture trajectory first examined by Belbruno[8]. For this problem, a common parameterization of the launch conditions is given by the hyperbolic excess velocity magnitude, v_∞ , and the right ascension and declination of the outgoing asymptote, α_∞ and δ_∞ , respectively. The flight time, Δt , is a free parameter, and one of the boundary conditions for a successful capture at the Moon is that the value of the Jacobi Constant of the spacecraft's state measured with respect to the Earth-Moon system be between the Jacobi Constant values associated with the interior and exterior libration points that are near the Moon. The other boundary condition is that the radial distance between the spacecraft and the moon be less than the radius of the Hill sphere around the Moon at the time of the capture. The problem is further complicated by the fact the gravitational attraction of the Sun is required for this solution to exist. This problem has four unknowns ($v_\infty, \alpha_\infty, \delta_\infty, \Delta t$) but three boundary conditions given as inequalities. It is a targeting problem, but much more complex than the orbital two point boundary value problem, that can still be solved as a system of nonlinear equations.

The second case of a multi-point boundary value problem formulation results if optimal control theory is used to formulate the multi-point boundary value conditions associated with a specific problem. Here the solutions produced satisfy both the kinematic boundary conditions and the natural boundary conditions, commonly referred to as the transversality conditions, that are part of the first order necessary conditions for a solution of the optimal control problem. This case implicitly extremizes a scalar cost function given in either the Mayer, Lagrange, or Bolza forms and is referred to as an indirect method.

The nonlinear constrained parameter optimization problem directly extremizes a scalar cost function based on a variable parameter vector that can include parameters of the model and/or all other variables defining the states and dynamics of the spacecraft. A hybrid formulation is also possible where the search variables associated with the optimal control formulation of the multi-point boundary value problem augment the parameter vector of the pa-

parameter optimization problem. This eliminates the necessity of deriving and implementing the transversality conditions as constraint conditions associated with the optimal control problem.

Having stated and described a class of solution methods that can solve these problems, significant importance is placed on the modeling of the problems, the proper choice of coordinate frames for state definition and targeting, and the proper identification of the independent and dependent variables. Though there are multiple ways in which a problem can be modelled, there are some that have better convergence properties than others for a given choice of solution method. A well designed system should facilitate, via experimentation if needed, the modelling of many types of problems and the generation of the solution procedures required to solve them. A solution procedure entails identifying a single stage or multi-stage procedure that drives an initial estimate of a solution to convergence.

It is noted that the trajectory problems solved by this system, whether optimal in some sense or not, are referred to as ‘open loop’ solutions in contrast to the guidance or stabilization problems encountered in many control problems. These solutions will eventually have to be controlled with closed loop feedback systems because in a real operational implementation of the solutions, unmodelled perturbations, uncertainties in the state of the spacecraft, and maneuver execution errors will be present.

Fundamental to the general methodology presented here is the use of a basic element referred to as the *basic segment* that is composed of several basic entities. The basic segment is the basic building block from which all trajectories are constructed. Fundamentally, the basic segment is an arc that connects two endpoint nodes. The segment can accommodate velocity and mass discontinuities at either or both of the endpoint nodes. The velocity discontinuities result from either an impulsive maneuver produced by an engine or from an approximation to a gravity assist flyby about a given celestial body. However, whether the gravity assist is approximated by an impulse change in the velocity vector or whether it is a fully integrated trajectory arc about a celestial body is a choice allowed in the definition of the segment. The mass discontinuities at either node result from the mass depletion associated with an impulsive maneuver, or the mass changes that result when part of a spacecraft is discarded or added. This results when an engine stage is discarded or when a spacecraft is captured after a rendezvous. The arc connecting the end point nodes can be either a ballistic arc or a controlled arc if a finite burn engine or any other device (such as a solar sail) is used to produce an independent acceleration to the spacecraft. A ballistic arc is the same as a coast arc. Each segment can be altered to form several segment types. For example, a segment can be defined to be an impulse followed by a coast period, or a finite burn arc without the endpoint impulses, or simply a node point defined with an epoch, a position, and a velocity.

Inherent to the system is the capability to work with disconnected segments. A segment is disconnected to any other segment if any of the state variables (position, velocity, or mass) at either node of one segment is discontinuous with respect to any of the state variables at the nodes of the other segment. Common sets of disconnected segments are those that have position discontinuities. This allows multiple spacecraft missions to be studied because additional spacecraft are modelled as segments that may be dynamically

uncoupled from the segments used to model the trajectories of the other spacecraft. If some of the spacecraft are required to interact with other spacecraft, the node points of the segments that represent them are coupled via suitable boundary conditions to examine intercept, rendezvous, constrained formation flight, or any other type of problem that requires constraining the dynamic state of any spacecraft with respect to other spacecraft.

A similar method of using disconnected segments was used by Byrnes and Bright[9] to examine complex impulsive maneuver based multiple body flyby trajectories to connect and optimize initially disconnected segments similar in concept to those described here. Though their definition of a segment and how it is propagated is different than the one presented here, the concept is similar in that a solution is achieved by simultaneously minimizing the total sum of the impulsive maneuvers and achieving continuity in position, and optionally the velocity vectors, at initially disconnected node points that they referred to as ‘breakpoints’. Their method was able to robustly produce solutions to the Galileo tour of the Jovian system and the heliocentric multi-planetary flyby trajectory for the Cassini mission.

The principal goal of current the system and architecture is to solve complex problems in a standard way without the need to develop specific models and algorithms for specific problems. If designed and implemented correctly, a general trajectory design and optimization system should be the model of choice for any specific problem, regardless of its complexity. Such a system should be capable of solving practical trajectory design and optimization problems using multiple propulsion systems, multiple spacecraft, and multiple celestial bodies and with any set of measurable perturbing accelerations in the force field. Examples of the type and scope of the problems that can be solved by the system include:

1. Trajectories about a single celestial body for orbit transfers, rendezvous, intercepts, arrival and capture, departure and escape;
2. Transfer and return trajectories between any pair of celestial bodies that orbit each other or that are in orbit about another celestial body;
3. Trajectories associated with the libration points of any two celestial body system, including transfer trajectories to and between libration points or libration point orbits; or libration point orbit trajectory design;
4. Sample return missions including descent and ascent at the target celestial body including any necessary rendezvous maneuvers;
5. Ballistic, low energy, or low thrust cycler trajectories between any pair of celestial bodies;
6. Multiple body gravity assisted trajectories in the Solar System or any central body with one or more natural satellites using any combination of impulsive and/or low thrust maneuvers;
7. Ballistic or controlled low energy capture and escape from any celestial body using the direct influence of other celestial bodies.

A secondary goal of this system is to efficiently produce an accurate solution to a problem with minimal effort. Minimal effort is defined to be a level

of effort required to generate a solution based on a predefined procedure to model and solve a particular problem. Producing a predefined procedure for a given problem, i.e., determining how many segments should be used, what are the independent and dependent variables, etc., can be a lengthy process. However, once this process is shown to achieve convergence from a wide range of initial estimates, then it becomes an automated process facilitating the solution to similar problems with different parameters. Complex problems may require a multi-stage approach, where sub-problems are solved independently and then combined in an overall solution. A well designed system should at least generate solutions to the subproblems efficiently and attempt to solve the complex problem.

A general approach to the problem facilitates the solution and optimization of trajectories for many types of missions in any force field encountered in *most* spacecraft trajectory problems. Though it is desired that this architecture solve *all* spacecraft trajectory problems foreseen in the next several decades, problems may be posed that are not solvable with the current architecture without further analysis, development, and generalization.

Common to all of these problems is the requirement to have available an initial estimate of a solution that leads to convergence. This initial estimate becomes increasingly difficult to produce as the problem increases in complexity. This initial estimate has often been referred to as the ‘first guess’, however, this term will be avoided, because it should not be a ‘guess’. There is a lot of information available in a problem that can be used to construct a first estimate. Without going into details, the analytical solutions or approximations for simple force models and the dynamical systems based analysis currently being developed for more complex models (Ref.[1]) serve as a basis for the construction of these estimates. However, convergence to a solution from even a well founded initial estimate is not guaranteed, given the complexity and scope of the problems that can be considered.

Several unresolved issues remain that need to be addressed eventually if the system described here is to achieve some of the goals stated. First, it will be necessary to explore simplifications in the construction of initial estimates to most or all of the problems attempted. Secondly, if a comprehensive spacecraft dynamics system is to be produced, it will be necessary to incorporate a six degree of freedom spacecraft model to account for the attitude reorientation maneuvers required to properly align the spacecraft to perform the needed maneuvers. The spacecraft model considered in the current system is still restricted to be a three degree of freedom model. Thirdly, if a detailed spacecraft operations system is to be produced, it will be necessary to include the observability and navigation accuracy of the trajectory solutions as part of the cost function. And fourthly, for close proximity operations between multiple spacecraft such as formation flight, it will be necessary to add a general relative motion frame model to examine multi-spacecraft trajectory problems about arbitrary trajectories, flying in any force field environment.

Notation: All scalar quantities are typeset as italicized uppercase or lowercase, i.e., a and B are scalars. Vectors are typeset as bold lower case, i.e., $\mathbf{a}$ and $\mathbf{b}$ are vectors. Matrices are typeset as bold uppercase, i.e., $\mathbf{A}$ and $\mathbf{B}$ are matrices. The definition and dimensions of these variables or constants are context dependent and appear in the text where appropriate. Vectors are column vectors, so that if $\mathbf{a}$ is an n vector and $\mathbf{b}$ is an m vector and $\mathbf{a}$ is a

function of $\mathbf{b}$, then $\partial \mathbf{a} / \partial \mathbf{b}$ is an $n \times m$ matrix. Dots above any quantity represent differentiation with respect to time. Superscript and subscript symbols are used to further distinguish the meaning of a given quantity; the definition of these are given in the text where appropriate.

2 Trajectory Design and Optimization Architecture

This section describes the basic elements of the system. This includes the formal definitions of the segments, trajectories, missions, force fields, and the coordinate systems.

2.1 Definition of the Basic Segment

The system makes use of what is termed the *basic segment*. It is a trajectory arc that connects two node points. The arc connecting the two node points is a solution to the equations of motion which are propagated by whatever means necessary. The force field models are generally smooth, highly nonlinear, and time dependent, thus requiring numerical based solutions of the equations of motion. A trajectory is composed of one or more segments. A complete mission is composed of one or more trajectories associated with one or more spacecraft. A spacecraft is any object in the model that is not a celestial body and does not influence the motion of any other object in the system such as another spacecraft or a celestial body. It is the object whose state at single or multiple times is being determined or controlled. A celestial body is any object that can influence the motion of a spacecraft via its gravitational potential or emitted radiation in the equations of motion. Celestial bodies also influence the motion of other celestial bodies in the system. This influence is directly available from a pre-computed ephemeris of the celestial bodies in the model. It is assumed that the motion of the celestial bodies is known, either from a realistic ephemeris or from an analytical approximation to a real ephemeris; i.e., the system only propagates the motion of one or all of the spacecraft in a mission.

The node points are tagged with an epoch that is referenced to a specified, but otherwise arbitrary, reference epoch, denoted as t_{epoch} . The epoch of the initial node point of any segment i is t_0^i and the epoch of the final node point is t_f^i . The superscript ‘ i ’ denotes the segment number. A mission can have any number of segments so that $i = 1, \dots, n$. There is no restriction on the values that t_0^i and t_f^i can have, i.e.,

$$t_0^i = t_f^i \quad \text{or} \quad t_0^i > t_f^i \quad \text{or} \quad t_0^i < t_f^i$$

However, the values of t_0^i and t_f^i can be constrained in any way necessary during the solution process, i.e., both quantities can be independent if desired or functionally dependent with respect to each other or the time epochs of the node endpoints of any other segment. For example, the time of flight for a segment i can be constrained to be less than or equal to a specified time of flight, or the time of flight of any other segment k ; i.e., $dt^i = t_f^i - t_0^i \leq dt^k$. Forward time or backward time propagations are handled in the same way and the temporal direction is only determined by the specific values of t_0^i and t_f^i . If $t_0 \neq t_f$, all segment propagations are from t_0 to t_f , regardless of their relative values. No propagation is made if $t_0 = t_f$.

The state of the spacecraft at either node point is given by its position, $\mathbf{r}_{(0,f)}^i$, velocity $\mathbf{v}_{(0,f)}^{i-}$, and mass, $m_{(0,f)}^{i-}$. The subscript $(0, f)$ on each of these quantities denotes that the quantity is referenced to either the t_0 node or the t_f node. The superscript ‘ $-$ ’ on $\mathbf{v}_{(0,f)}^{i-}$ and $m_{(0,f)}^{i-}$ states that the value of the velocity and the mass, respectively, is given prior to any possible discontinuities in their values.

There are two types of velocity impulses allowed at either node. The first type is a maneuver based impulse provided by an engine and the second type is a gravity assist impulse. The gravity assist impulse is used to approximate the change in velocity relative to a fixed external reference frame not attached to the celestial body providing the gravity assist. It is useful only when solving a problem where the flyby celestial body is treated as a zero-point mass. The components of the gravity assist impulse are constrained to satisfy the conservation of energy across the flyby and optionally, a minimum flyby radius relative to the central body. A fully integrated flyby of a celestial body without any associated discontinuity in the velocity is also allowed, but this flyby is modelled as a numerically integrated segment with a non-zero time of flight duration. Thus two ways to model a gravity assist are available, with one of them being an approximation. The approximation can be used for broad searches that may include multiple flybys and accuracy is not critical. The integrated flybys are used for more accurate trajectories.

Both types of velocity impulses are treated the same way, except for the mandatory constraint imposed on the gravity assist impulse and the fact that the gravity assist impulse does not have an associated mass depletion. To model a ‘powered’ gravity assist flyby where in addition to the gravity assist impulse there is an additional maneuver based impulse, two segments are used. One of the segments can be a simple node with an impulse representing the gravity assist, and the other segment has an impulse representing the maneuver. If the epochs for both segments are the same, the order of the segments is not important.

The impulse at either node can have zero magnitude. After the impulse, the velocity vector is

$$\mathbf{v}_{(0,f)}^{i+} = \mathbf{v}_{(0,f)}^{i-} + \Delta \mathbf{v}_{(0,f)} \quad (1)$$

where the superscript ‘ $+$ ’ specifies the value after the impulse.

The evolution of the mass value across a node depends on three allowable and distinct mass discontinuities. The first mass discontinuity is a non-maneuver mass discontinuity that can result from either a mass drop off or a mass add on, so it can be positive or negative and is labeled as $\Delta m_{(0,f)}^-$. In other words, a spacecraft component can be discarded prior to the impulse. Or, if the spacecraft has performed a rendezvous with another spacecraft and has captured it, a positive mass discontinuity represents that additional mass associated with this capture. The mass value after this first non-maneuver mass discontinuity is

$$m_{(0,f)}^{i-+} = m_{(0,f)}^{i-} + \Delta m_{(0,f)}^- \quad (2)$$

where $\Delta m_{(0,f)}^-$ is the value of the mass change. Proceeding with the impulse, the mass value after the maneuver is

$$m_{(0,f)}^{i+-} = m_{(0,f)}^{i-+} + \Delta m_{(0,f)}^i \quad (3)$$

where $\Delta m_{(0,f)}^i$ is the mass change that results from the instantaneous depletion of propellant that results from a maneuver based impulse. It is zero for a gravity assist impulse. The maneuver based mass change is directly related to the magnitude of the maneuver, $\Delta v_{(0,f)}^i$, and the exhaust velocity of the engine used to provide this maneuver, $c_{(0,f)}^i$. The maneuver mass discontinuity is obtained from a form of the rocket equation,

$$\Delta m_{(0,f)}^i = m_{(0,f)}^{i-+} (e^{-\Delta v/c} - 1) \quad (4)$$

where the superscripts and subscripts on Δv and c have been omitted for notational simplicity. The exhaust velocity by definition is related to the specific impulse of the propellant used and the reference gravity acceleration value at the Earth's surface, $c_{(0,f)}^i = \left(Isp_{(0,f)}^i \right) (g_{earth})$.

Following the impulse, another non-maneuver mass discontinuity with the same characteristics as the one prior to the impulsive maneuver is allowed. The mass value at the end of either node is then

$$m_{(0,f)}^{i+} = m_{(0,f)}^{i+-} + \Delta m_{(0,f)}^+ \quad (5)$$

where $\Delta m_{(0,f)}^{i+}$ is the post impulse, non-maneuver mass discontinuity. Typically this mass change results when the engine stage used to produce the impulsive maneuver is discarded. However, allowance is made so that a mass add on can occur at this point again for reasons associated with a spacecraft capture. The reason non-maneuver mass discontinuities are allowed on either side of the impulse at either node is because in the case of a spacecraft rendezvous and capture, the maneuver performing the rendezvous could occur at the initial node point of a given segment, or at the final node point of a previous segment to which the given segment is connected to.

Based on the evolution of the velocity and mass across a node point, the following distinct times labels that are equal in value but are used to distinguish the values of both the velocity and the mass are

- $t_{(0,f)}^-$: node initial time, and time prior to any velocity or mass discontinuities
- $t_{(0,f)}^{-+}$: time after first non-maneuver mass discontinuity
- $t_{(0,f)}^{+-}$: time after the velocity impulse
- $t_{(0,f)}^+$: node final time, and time after the second non-maneuver mass discontinuity

The arc that connects both node points can be either a ballistic arc with no independent control, or a controlled arc with thrust or acceleration controls. The control is provided by a thrust vector from an engine or a controlled acceleration that results from a non-mass depleting device such as a sail using radiation from a photon emitting celestial body such as the Sun or a star, or any other external momentum transfer device. Any system used to provide this control will be referred to as a propulsion system, though some may not use propellant such as in the case of a sail. The arc connecting the t_0 node to the t_f node of a segment i is defined by both the parameters of the propulsion system and the equations of motion. The control vector, $\mathbf{\Gamma}(t)$, will in general have the following functional dependence,

$$\mathbf{\Gamma}^i = \mathbf{\Gamma}^i(c(t), T(t), P(t), \varepsilon, \mathbf{u}(t), m(t), \mathbf{a}_p, t) \quad (6)$$

where c is the exhaust velocity, T is the thrust, P is the power, ε is the efficiency of the propulsion system, $\mathbf{u}$ is the control direction unit vector, m is the instantaneous mass, and $\mathbf{a}_p$ is a vector containing any other parameter that defines the propulsion system. For example, in the case of a sail, $\mathbf{a}_p$ will contain parameters such as sail area, surface reflectivity, and other parameters that uniquely define it.

For any segment i , the dynamic state of the spacecraft along the arc for times between t_0^{i+} and t_f^{i-} is defined as an augmented state vector comprised of its position, velocity, and mass $(\mathbf{r}^i(t) \quad \mathbf{v}^i(t) \quad m^i(t))^T$ and satisfies the first order vector equation of motion,

$$\begin{pmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{m} \end{pmatrix}^i = \begin{pmatrix} \mathbf{v}(t) \\ \mathbf{g}(\mathbf{r}, \mathbf{v}, m, t, \mathbf{a}_g) + \mathbf{\Gamma}(t) \\ -T(t)/c(t) + \dot{m}_p \end{pmatrix}^i \quad (7)$$

Here, $\mathbf{g}(\mathbf{r}, \mathbf{v}, m, t, \mathbf{a}_g)$ is the acceleration per unit mass resulting from control independent terms such as the acceleration due to the gravitational potential from any celestial body, radiation pressure from energy emitting celestial bodies, or nonconservative forces such as atmospheric drag. The symbol ‘ $\mathbf{g}$ ’ commonly has been used to represent the gravitational acceleration due to gravitating celestial bodies with a dependence only on position and time. Here it is generalized to also include a dependence on velocity, $\mathbf{v}$, mass, m , and an additional problem specific acceleration, $\mathbf{a}_g$. The acceleration vector $\mathbf{a}_g$ in the acceleration vector contains all of the constant or variable parameters of the force model such as the gravitational constant, the mass of the celestial bodies, the non-spherical gravitational potential field of the celestial body, the atmospheric parameters of the celestial body, the radiation parameters of any energy emitting celestial body, and any other term that may appear in the ballistic acceleration of the spacecraft. This acceleration may be time dependent. The time dependency of the force field results in part from the motion of the celestial bodies in the model or the rotation of a body about an internal axis if the non spherical gravitational potential of a celestial body is used. The mass rate, $\dot{m}^i$, is a result of fuel consumed during an engine burn and is related to the thrust, T , and exhaust velocity c , or any other arbitrary mass depletion that results from intentional or unintentional continuous venting of liquids or gasses, or mass accumulation during flight through resisting a medium such as dust clouds or atmospheres. The non-thrust contribution to $\dot{m}^i$ is given by the general term $\dot{m}_p$. In summary, the spacecraft equations of motion between node points at t_0^i and t_f^i are completely general and arbitrary, but known.

For any segment, the parameters for three propulsion systems need to be specified: one for the initial impulsive maneuver, one for the controlled arc, and one for the final maneuver. If the impulsive maneuvers are due to a gravity assist, then the propulsion system parameters for these are unimportant. The propulsion system parameters can all be defined for a single propulsion system, i.e., the exhaust velocity for the initial and final maneuver based impulses, and the controlled arc, assuming it is a constant exhaust velocity system, can be defined to be the same system by equating the exhaust velocity values of each segment node and the exhaust velocity of the controlled arc. But since these can be independent, multiple propulsion systems can be used in

one individual segment. If the segment is a pure coast or ballistic arc then all propulsion system parameters are set to zero. If a segment is initiated with a high thrust booster, whose maneuver can be approximated with an impulsive maneuver, and controlled arc of the segment uses an independent low thrust propulsion system, then the arc requires the definition of the initial node impulsive maneuver and the parameters defining the low thrust engine.

Another key entity of the basic segment is referred to as floating node point. It is similar to the two final endpoint nodes in that it has an associated time tag denoted as t_i that is required to lie between t_0 and t_f ,

$$t_0 \leq t_i \leq t_f \quad (8)$$

The main restriction of the floating node point is that it cannot have any velocity or mass discontinuities. It is used as a position, velocity, and mass reference measured along the segment. Since it is a node, the state vector associated with it can be constrained, or it can serve as a constraint for other segments. For example, a rendezvous between a spacecraft on segment i and a spacecraft on segment k can occur at the floating node point of segment k . The boundary conditions would be $\mathbf{r}_f^i = \mathbf{r}_i^k$ and $\mathbf{v}_f^i = \mathbf{v}_i^k$. Another example for the use of the floating endpoint is to find the location along a segment i that is at the periapsis point with respect to a celestial body. Here, t_i is a free parameter constrained by Eq. 8 and its value must be such that $\dot{r}^i(t_i) = 0$ and $\ddot{r}^i(t_i) > 0$, where $\dot{r}^i(t_i)$ is the radial velocity with respect to the celestial body, and $\ddot{r}^i(t_i)$ is the second time derivative of the position magnitude. Both of these conditions are sufficient for finding the location of periapsis. These types of constraints could be imposed without the need to define a floating node point because either endpoint nodes could be used to serve the same purpose. However, including the floating node point, simplifies the modelling process by removing the need to include an additional segment for some types of problems.

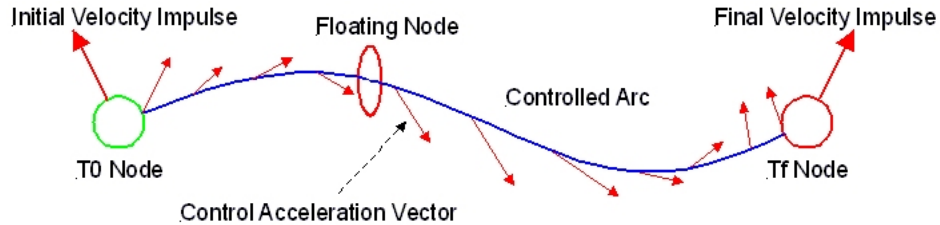


Figure 1: The basic segment Building Block. Velocity impulses can exist at either node and the arc connecting the nodes can be either a ballistic or an controlled arc with a time dependent variable control vector. Mass discontinuities can exist at either node that result from an impulsive maneuver or non-maneuver mass changes. A floating node point lies between the endpoint nodes.

A sketch of the basic segment is given in Figure 1. Figure. 2 shows the exploded time scale for the segment and the locations in time of the discontinuous and continuous states of the spacecraft. A subset of the parameters that define the basic segment are listed in Table 1. Within an individual segment, some of these quantities are independent and some are dependent. Outside of the segment, as in a trajectory or a mission, any of these quantities can be constrained and will thus be dependent in the solution process.

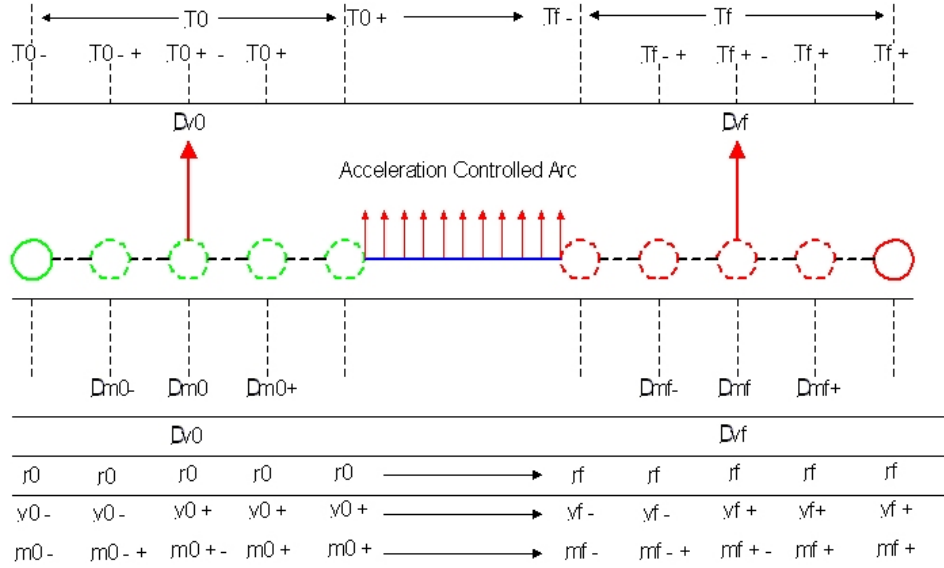


Figure 2: Exploded View Representation of the basic segment. Mass discontinuities can exist at each node point. These are either stage drop offs, mass additions, or impulsive maneuver mass discontinuities. The non maneuver mass discontinuities occur on either side of the velocity impulses. The velocity impulses are due to either an impulsive maneuver or an approximated gravitational assist. The velocity before an impulsive is $\mathbf{v}^-$, the velocity after an impulsive maneuver is $\mathbf{v}^+$. The maneuver based impulses have an associated mass discontinuity, Δm . The non-maneuver mass discontinuities are Δm^- if it occurs before the impulse, and Δm^+ if it occurs after the impulse. At either node point, the beginning mass value is m^- . The mass value prior to a velocity impulse is m^{-+} . The mass value after a velocity impulse is m^{+-} . The mass value after the second non-maneuver mass discontinuity is m^+ . The controlled accelerated arc is between t_0^+ and t_f^- . All state variables vary continuously along this arc. The simplest type of segment is a node point at t_0 with all parameters set to zero except for the node state variables, $\mathbf{r}_0, \mathbf{v}_0^-, m_0^-$ which are required to be defined.

The basic segment also includes a Lagrange multiplier vector, λ , adjoined to the physical state variables, $\mathbf{r}, \mathbf{v}, m$ at each time instant associated with each node and between t_0^+ and t_f^- . For $t_0^+ \rightarrow t_f^-$, the evolution of Lagrange multiplier vector is governed by the Euler-Lagrange differential equations associated with the optimal control problem. Other names given to this vector include the adjoint vector and the costate vector. This vector is composed of the position costate vector, λ_r , the velocity costate vector, λ_v , and the mass costate scalar, λ_m .

Additionally, because of the basic segment definition, the segments can be all independent and initially disconnected. The dependency a segment may have with respect to another segment is specified in the constraint definitions. Because of this, the architecture allows then the modeling of multiple spacecraft problems. It is not necessary to specify how many spacecraft are in the model. All that is required is to model as many segments as needed and that they be constrained in any way necessary to represent the number of spacecraft in the model. For example, a spacecraft rendezvous between two

spacecraft requires a minimum of two segments. A possible constraint is that the final endpoint nodes of both spacecraft have the same position and velocity at the final solution. If both spacecraft have maneuvering capability, these maneuvers are then defined and allowed to be adjusted for each segment. If one spacecraft remains passive, then its segment definition has its propulsion parameters nulled out.

Symbol	Variable Type/Name	Dim.	Dep.	Range or Constraint	Units
t_0	initial time	1	i	$-\infty < t_0 < \infty$	day
r_0	position	3	i	$0 \leq \mathbf{r}_0 < \infty$	km
v_0^-	velocity	3	i	$0 \leq \mathbf{v}_0^- < \infty$	km/s
m_0^-	mass	1	i	$0 \leq m_0^- < \infty$	kg
Δm_0^-	mass discontinuity	1	i	$-\infty < \Delta m_0^- < \infty$	kg
λ_0^-	Lagrange multiplier	7	i	$0 < \lambda_0^- < \infty$	—
Δv_0	velocity impulse	3	i	$0 \leq \Delta \mathbf{v}_0 < \infty$	km/s
Isp_0	specific impulse	1	i	$0 \leq Isp_0 < \infty$	sec
Δm_0	maneuver mass discontinuity	1	d	$0 \leq \Delta m_0 < \infty$	kg
Δm_0^+	mass discontinuity	1	i	$-\infty \leq \Delta m_0^+ < \infty$	kg
v_0^+	velocity	3	d	$0 \leq \mathbf{v}_0^+ < \infty$	km/s
m_0^+	mass	1	d	$0 \leq m_0^+ < \infty$	kg
λ_0^+	Lagrange multiplier	7	d	$-\infty < \lambda_0^+ < \infty$	none
t_i	intermediate time	1	i	$t_0 \leq t_i \leq t_f$	day
T	thrust	1	i	$0 \leq T \leq T_{\max} < \infty$	Newtons
Isp_i	specific impulse	1	i	$0 \leq Isp_i \leq Isp_{\max} < \infty$	sec
P	power	1	i	$0 \leq P \leq P_{\max} < \infty$	Watts
u	thrust direction unit vector	3	i	$ \mathbf{u} = 1$	unit vector
ε	engine efficiency	1	i	$0 \leq \varepsilon \leq 1$	—
t_f	final time	1	i	$\infty < t_f < \infty$	day
r_f	position	3	d	$0 \leq \mathbf{r}_f < \infty$	km
v_f^-	velocity	3	d	$0 \leq \mathbf{v}_f^- < \infty$	km/s
m_f^-	mass	1	d	$0 \leq m_f^- < \infty$	kg
Δm_f^-	mass discontinuity	1	i	$-\infty < \Delta m_f^- < \infty$	kg
λ_f^-	Lagrange multiplier	7	d	$-\infty \leq \lambda_f^- \leq \infty$	—
Δv_f	velocity impulse	3	i	$0 \leq \Delta \mathbf{v}_f \leq \infty$	km/s
Isp_f	specific impulse	1	i	$0 \leq Isp_f \leq \infty$	sec
Δm_f	maneuver mass discontinuity	1	d	$0 \leq \Delta m_f < \infty$	kg
Δm_f^+	mass discontinuity	1	i	$-\infty < \Delta m_f^+ < \infty$	kg
v_f^+	velocity	3	d	$0 \leq \mathbf{v}_f^+ < \infty$	km/s
m_f^+	mass	1	d	$0 \leq m_f^+ < \infty$	kg
λ_f^+	Lagrange multiplier	7	d	$-\infty < \lambda_f^+ < \infty$	—

Table 1: Basic set of Elements Defining the Basic Segment

Since the segments can be initially disconnected, the introduction of additional variables and possible constraints, though increases the number of variables, also decreases the sensitivity of the iterated solutions compared to strictly forward time or backward time propagated trajectories if the magnitude of the duration of a segment is longer than allowed for perturbations to be linearly valid in gradient based solution methods. However, if this is not an issue, any segment j can be forced apriori to be connected sequentially to any other segment i at any of the node points and distinct times of segment i . The idea of propagating disconnected segments that will eventually be connected or at least constrained in some way is a generalization of what is known as direct multiple shooting.

The definition of the basic segment described here can be altered to include more parameters. The working model described has been sufficient to examine a broad class of trajectory problems. The definition of the basic segment, however, is dynamic so that yet unforeseen trajectory problems can be accommodated. The simplest mission can be modelled with only one segment. More complex missions will require more than one segment.

2.1.1 Segment Types

The general definition of the segment allows for the modeling of different segment types. The type and number of segments used depends on the nature of the problem, the number of spacecraft in the model, the propulsion systems used in the model, and the complexity of the trajectory or mission. Based on the basic segment, the segment types that can be modelled, by properly specifying the values that define it, are listed in Table 2. Figure 3 shows the different segment types beginning with the most basic segment type and proceeding down to the simplest segment models which are simple node points or node points with impulsive maneuvers. Each segment type in Figure 3 is labeled with an integer identifier that corresponds to those listed in Table 2.

Segment Type	Segment Description
1	t_0 node, $\Delta\mathbf{v}_0$, controlled arc, t_f node, $\Delta\mathbf{v}_f$
2	t_0 node, $\Delta\mathbf{v}_0$, controlled arc, t_f node
3	t_0 node, controlled arc, t_f node, $\Delta\mathbf{v}_f$
4	t_0 node, controlled arc, t_f node
5	t_0 node, $\Delta\mathbf{v}_0$, ballistic arc, t_f node, $\Delta\mathbf{v}_f$
6	t_0 node, $\Delta\mathbf{v}_0$, ballistic arc, t_f node
7	t_0 node, ballistic arc, t_f node, $\Delta\mathbf{v}_f$
8	t_0 node, ballistic arc, t_f node
9	t_0 node, $\Delta\mathbf{v}_0$
10	t_0 node
11	t_f node, $\Delta\mathbf{v}_f$
12	t_f node

Table 2: Segment Types as Sub Types of the Basic Segment

Recall that a trajectory for a single spacecraft is comprised of one or more segments. Therefore, it is possible to model a trajectory for a single spacecraft that is equipped with one or more propulsion systems. Further, mass discontinuities resulting from either impulsive maneuvers, mass drop offs, and mass

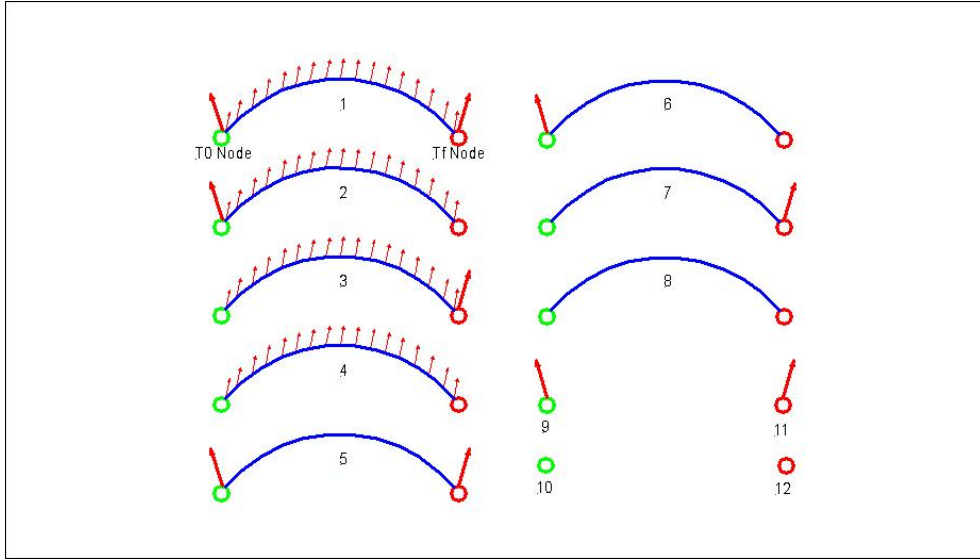


Figure 3: Segment Types Based on the basic segment. These are treated as building blocks to construct many types of trajectories for one or more spacecraft. The large vector arrows represent velocity impulses; the small vector arrows represent a finite acceleration control arc. Dynamically there is no difference between segment types 9 and 11 or 10 and 12 except that the time epochs are different since segment types 9 and 10 are the t_0 node and segment types 11 and 12 are the t_f node, with and without a velocity impulse as shown.

additions are accounted for by properly defining these quantities in the definition of the segment. A mission which is composed of one or more trajectories, which in turn is composed of one or more segments, will have as its basic building block any of the segment types shown in Figure 3 and tabulated in Table 2. The choice of which segments need to be connected to which other set of segments is problem dependent. In the case of multiple spacecraft problems, the connectivity of the segments is determined by the type of multiple-spacecraft mission being analyzed. For independent spacecraft that are not required to rendezvous or intercept, these may all be disconnected but constrained in any way necessary, such as maintaining their positions to satisfy some geometrical formation, for example. For intercept or rendezvous problems, clearly, some segments will need to be connected; i.e., for rendezvous problems, position and velocity are required to be continuous, or for intercept problems, only position needs to be continuous after the maneuver. A rendezvous between two spacecraft where one of the spacecraft assumes the additional mass of the other spacecraft will require the addition of a non-maneuver mass discontinuity which can be a fixed or variable quantity depending on whether mass of the captured spacecraft is fixed or variable.

Impulsive maneuvers, which are constrained to occur only at the endpoints, can be referenced to any allowable fixed-center frame or relative to the trajectory state associated with the respective node point allowing either velocity vector or radius vector referenced maneuvers. Finite acceleration maneuvers (finite engine burns, sails, etc.) can also be steered relative to the trajectory or remain fixed in any fixed-centered frame. In the case of finite burn maneuvers, both thrust constrained and power constrained thrust arcs can be modelled.

Algorithmically, no distinction is made between finite burn, high thrust or low thrust systems. Both are treated equally and the specified propulsion system parameters further defines the type of thrust arc used. The variation of the thrust magnitude, the exhaust velocity, and the power are constant or variable depends on how each of these are constrained.

2.2 A Conceptual Modelling Example

To illustrate the use of using the segment building blocks to construct a complex mission, consider the following hypothetical mission. It is desired to transfer a main spacecraft stationed at the vicinity of the interior libration point of the Earth-Moon system to the vicinity of the Jovian moon, Europa. A landing and ascent vehicle of fixed and known initial mass is attached to the main spacecraft. At Europa the lander separates from the main spacecraft, descends to the surface of Europa, performs the required objectives, and ascends to rendezvous with the main spacecraft. The main spacecraft returns to its starting point in the Earth-Moon system, but on its return route, it intercepts and flies by a comet. The main spacecraft is equipped with a nuclear powered continuous thrust, variable specific impulse engine. The lander/ascent vehicle is equipped with a high thrust constant specific impulse engine with fixed propellant mass. The mission performance objective is to minimize the initial mass of the main spacecraft while constraining the final mass (the dry mass) of the main spacecraft. The total mission duration is constrained to be no greater than a given value and the stay time for the lander at Europa is constrained to be no less than a given value. The mission can utilize any beneficial gravity assist maneuvers around celestial bodies that exist in the model. Clearly, there are many more details required to fully model this mission. However, the basic parameters stated are enough to conceptually illustrate the modeling of this type of mission. Figure 4 illustrates a possible set of segment blocks that can be used to construct the first iterate. The short dashed lines indicate the node points that need to be connected and required to be continuous at least in position and possibly velocity. In this sketch, the first assumption is that the mission can be modelled with ten segments. Other parts of the mission, such as the landing and ascent phases can be decomposed into further segments if necessary. To correctly design and optimize such a mission, it will be necessary to include the gravitational attraction from at least the principle celestial bodies involved which in this case include the Sun, Earth, Moon, Jupiter, and Europa. The comet can be assumed to be a non-gravitating celestial body and can be modelled as an independent segment if its state vector is known at some epoch, or it can be treated as a gravitating celestial body if its ephemeris is known. Further, it will be necessary to specify the initial state or orbit of the main spacecraft with respect to a reference frame in which the location of the interior libration point of the Earth-Moon system is known. At departure, the trajectory may take advantage of the complex force field that is dominated by the simultaneous attraction from the Sun, Earth, and Moon. At Jupiter, the trajectory may also exhibit a complex behavior since the force field will be dominated by a restricted four body model that includes the Sun, Jupiter, and Europa. Though a solution to this mission is not presented here, it is one example of a type of mission that can be modelled and solved with the system described

here.

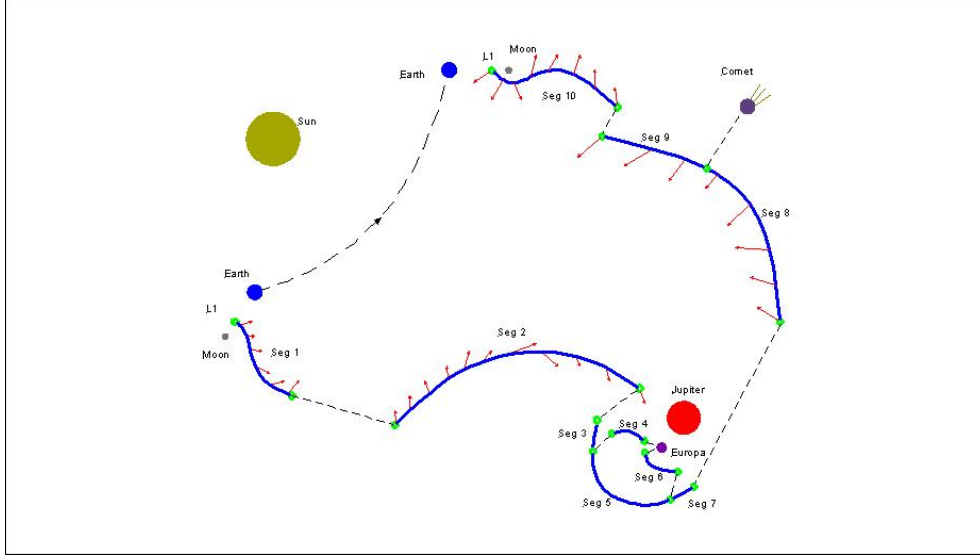


Figure 4: Modelling of a Complex Mission. Different segment types are used to construct a round trip mission beginning at the interior libration point of the Earth-Moon system, to the Jupiter-Europa System, with an intermediate flyby of a comet, before returning to the starting point. Some segments will be low thrust arcs, others will be ballistic arcs with impulsive maneuvers at their nodes. All segments are shown initially disconnected. Some of these could have been defined to sequentially connected to other segments initially. The dashed lines between some of the nodes imply that these nodes are required to be connected at the final solution.

2.3 The Equations of Motion and the Propagation Reference Frame

For a general system, the choice of reference frame in which to model the spacecraft's three degree of freedom point dynamics and the associated equations of motions is critical from several viewpoints. First, it is desired to have the simplest reference frame model allowable without compromising the validity of a solution and its accuracy. Second, in a comprehensive model, that may include multiple spacecraft, multiple propulsion systems, and multiple celestial bodies, it is desirable to work with a coordinate frame for state propagation in which transformations between the frames in which the states, targets, and maneuvers are referenced to are simple and computed efficiently. If the dynamics are modelled correctly, any reference frame used must yield the same result. On the other hand, some reference frames may include terms in the equations of motion which approach the limiting accuracy with which small terms are evaluated by the computer. Aware of the accuracy issues associated with the proper choice of reference frame for state propagation, the current system propagates the equations of motion in either a reference frame that is fixed (non-rotating) to any barycentered frame that can be defined or to any chosen celestial body. If it is attached to a specified celestial body, the frame translates but does not rotate. The celestial body selected to be the center of this frame depends on the nature of the problem being modelled.

Each segment in the mission can have a distinct reference frame for propagation if desired. The only restriction is that it be a non-rotating frame. For example, if interplanetary missions are being solved with respect to the Sun, and the other celestial bodies of interest are treated as non gravitating zero point masses, the reference frame should be centered at Sun or the solar system barycenter. For the same mission, if some of the segments are required to operate in the vicinity of a celestial body, such as in the case of a long duration spiral escape or capture from a given celestial body, then that segment should have as its propagation frame, a frame fixed to the celestial body of interest. In some problems, state definition and targeting can be specified in coordinates of a rotating frame defined between two celestial bodies. Though it is possible to integrate the equations in coordinates of the rotating frame, experimentation has so far determined that this does not provide any significant advantages. However, the use of rotating frame coordinates for state definition and targeting is a key component of the current architecture. The discussion of the rotating frames used for this purpose are reserved for a later section.

In a barycentered frame that is assumed to be inertial the second order equations of motion are

$$\ddot{\mathbf{r}} = \sum_{j=1}^{n_{cb}} GM_j \frac{\mathbf{r} - \mathbf{r}_j}{|\mathbf{r} - \mathbf{r}_j|^3} + \mathbf{a}_{pert} + \mathbf{\Gamma}(t) \quad (9)$$

where $\mathbf{r}$ is the relative position vector of the spacecraft with respect to barycentered frame, $\mathbf{r}_j$ is the relative position of celestial body j with respect to the barycentered frame, n_{cb} is the number of celestial bodies in the model, and GM_j is gravitational parameter for celestial body j . All additional non-control related terms are contained in $\mathbf{a}_{pert}$ and control related terms are contained in $\mathbf{\Gamma}$.

If the propagation reference frame is centered at a specified celestial body, denoted by CB_f , the equations of motion in this fixed (non-rotating) but translating frame is

$$\ddot{\mathbf{r}} = -\frac{GM_{CB_f}}{r^3} \mathbf{r} - \sum_{j=1}^{n_{cb}} GM_j \left[\frac{\mathbf{r} - \mathbf{r}_j}{|\mathbf{r} - \mathbf{r}_j|^3} + \frac{\mathbf{r}_j}{r_j^3} \right] + \mathbf{a}_{pert} + \mathbf{\Gamma}(t) \quad (10)$$

where GM_{CB_f} is the gravitational parameter of the celestial body to which the frame is fixed, $\mathbf{r}_j$ is the position vector of the other celestial bodies with respect to CB_f , and n_{cb} is the number of celestial bodies treated as third bodies.

The vector terms associated with the gravitational acceleration due to third bodies include both the direct acceleration vector term and the indirect acceleration term needed to account for the fact the CB_f fixed centered frame is not an inertial frame. Eqs. 9 or 10 are used to propagate all of the segments from $t_0^{i+} \rightarrow t_f^{i-}$ ($i = 1, \dots, n$) where n is the number of segments in the mission and noting that each segment can use either set of equations and have its own distinct choice for the center of the reference frame.

The position of the other celestial bodies is obtained from an explicit time dependent ephemeris and hence the time dependence of the force model. The ephemeris provides the positions, and possibly the velocity, of any of the celestial bodies with respect to any other celestial body and can be a highly

accurate ephemeris, such as the Jet Propulsion Laboratory’s set of planetary ephemerides, or any user defined ephemeris. The user defined ephemeris is useful when studying the dynamics in simplified force models such as the circular or elliptic restricted three body problem; or user defined model solar systems with one or more arbitrary celestial bodies where the motion of the celestial bodies relative to the frame center can be simple Keplerian type orbits or based on a precomputed ephemeris.

For example, consider the analysis of cycling trajectories between two planets orbiting the Sun. Such a study should begin by assuming that the planets of interest are in circular orbits about the Sun. This model provides the basic properties associated with the construction of these trajectories. Once these properties have been understood, then the model is modified to account for the real motion of the planets. The assumption being that the circular model solutions carry over to the real planetary model with an expected variation. However, it is important to be aware that the model that uses the real planetary model may yield solutions that do not exist in the simplified circular model. If so, then a better approximation to the real planetary model should be used to identify the basic properties of cycling trajectories between these celestial bodies, such as an elliptic model for the motion of the planets of interest.

Current and future missions will require targeting relative to one or more solar system minor bodies such as asteroids and comets. Assuming an ephemeris for these bodies is available, then these are treated as celestial bodies with an associated gravitational parameter. If only a state vector and epoch is known for any of these bodies then either an ephemeris is computed separately with the current system by treating it as a ballistic segment, or the body is treated as a ballistic segment in the solution process by assuming that its mass does not influence the motion of the other spacecraft segments.

2.4 Multi-Body Reference Frames

There are certain spacecraft trajectory design and optimization problems that are best solved in alternate reference frames. In the case where two or more celestial bodies are included in the force model and any two of these bodies are in bounded orbits relative to each other, the dynamics of a spacecraft is best analyzed and understood in a reference frame that can in general translate, rotate, and pulsate in such that the representation of the positions of the celestial bodies remain stationary. This frame is the well known rotating frame that has been used extensively in the study of trajectories in the Circular and Elliptic Restricted Three Body problems[10]. For example, consider the motion of a spacecraft in the Earth-Moon system. Analogous to the barycentered rotating frame of the restricted three body problem, a convenient frame in which both the Earth and Moon remain stationary is one that rotates with the instantaneous Earth-Moon line and pulsates with the varying distance between the Earth and Moon if the motion of the Moon about the Earth is based on a realistic ephemeris.

Another example of the use of a rotating frame is in the analysis of Earth-Mars trajectories where a suitable reference frame is one where both Earth and Mars remain stationary. Here, the fundamental axis is the instantaneous Earth-Mars line. The north pole axis of the Earth or the Sun can be used

to construct a normal vector to this line, from which the final right-handed coordinate frame is constructed. The advantage of this frame is clearly noted in the visualization and targeting of trajectories connecting both planets but more importantly, the capability then exists to specify initial state components and constraint functions directly in this frame greatly simplifying the targeting and optimization of trajectories relative to any set of moving bodies.

These two example reference frames can be examined under a single formulation. In the first example, two celestial bodies are in orbit about their common barycenter. In the other example two celestial bodies are orbiting a common reference center that is not their barycenter. The state, targeting, and maneuver definitions can be given in coordinates of this frame and appropriate transformations are needed to transform position, velocity, and acceleration between this frame and the segment propagation frame.

Consider the motion of two celestial bodies CB_i and CB_j . The time varying position vectors for these bodies are known relative to some other fixed frame that can be an inertial frame or a frame centered at another body; i.e., $\mathbf{r}_{CB_i}$ and $\mathbf{r}_{CB_j}$ are known vector functions of time. Define a unit vector along the relative position vector of CB_j with respect to CB_i

$$\hat{\mathbf{r}} \equiv (\mathbf{r}_{CB_j} - \mathbf{r}_{CB_i}) / |\mathbf{r}_{CB_j} - \mathbf{r}_{CB_i}| \quad (11)$$

Several options exist to define the remaining two basis vectors for the frame. If CB_j is in a closed and bounded orbit about CB_i such that its relative angular momentum vector with respect to CB_i remains nearly constant, then the remaining basis unit vectors are defined as

$$\hat{\mathbf{s}} \equiv \hat{\mathbf{t}} \times \hat{\mathbf{r}} \quad (12)$$

$$\hat{\mathbf{t}} \equiv (\mathbf{r} \times \mathbf{v}) / |\mathbf{r} \times \mathbf{v}| \quad (13)$$

where the $\hat{\mathbf{t}}$ unit vector is along the instantaneous specific angular momentum vector and $\mathbf{v}$ is the relative velocity vector of CB_j with respect to CB_i , $\mathbf{v} = \mathbf{v}_{CB_j} - \mathbf{v}_{CB_i}$. This basis can be used, for example if the two bodies are a planet and the Sun, or a planet and its moon. Figures 5 and 6 illustrate the CB_i - CB_j rotating frames in both a CB_k fixed frame and a rotating frame centered at one of the celestial bodies.

Alternatively, if the relative angular momentum vector between the bodies changes in way that the motion alternates between being retrograde to prograde (such as the motion of two celestial bodies around a central body that are not in orbit about each other) then it is convenient to use a nearly constant vector, such as a reference north pole axis to define the remaining basis unit vector. Let this reference unit vector be $\hat{\mathbf{z}}$. The remaining unit vectors then are

$$\hat{\mathbf{s}} \equiv (\hat{\mathbf{z}} \times \hat{\mathbf{r}}) / |\hat{\mathbf{z}} \times \hat{\mathbf{r}}| \quad (14)$$

$$\hat{\mathbf{t}} \equiv \hat{\mathbf{r}} \times \hat{\mathbf{s}} \quad (15)$$

This set of basis vectors is ideally suited for a rotating-pulsating frame where the Earth is CB_i and Mars is CB_j (or vice-versa). Figures 7 and 8 illustrate the CB_i - CB_j rotating-pulsating frames in both a CB_k fixed frame and a rotating-pulsating frame centered at one of the celestial bodies.

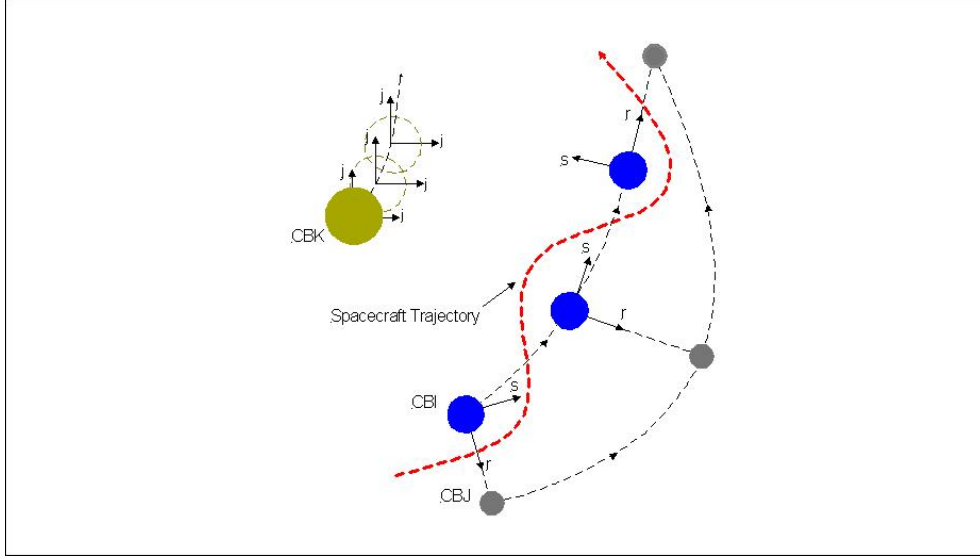


Figure 5: The CB_i - CB_j Rotating Frame with respect to a CB_k Fixed Centered Frame. The two celestial bodies, CB_i and CB_j , are in closed orbits about their common barycenter. In general, the distance between them varies.

The transformation of a position vector referenced in the rst frame to a fixed frame centered at CB_i with basis unit vectors $\mathbf{i}, \mathbf{j}, \mathbf{k}$ (referred to as the ijk frame) and scaled with respect to the instantaneous distance between CB_j and CB_i is

$$\mathbf{r}^{ijk} = \frac{r}{k} \mathbf{R} \mathbf{r}^{rst} \quad (16)$$

where $r = |\mathbf{r}|$, k is a positive scaling constant with the same units as r , and $\mathbf{R}$ is the transformation direction cosine matrix between the rst frame and the ijk frame,

$$\mathbf{R} = \begin{pmatrix} \mathbf{i} \cdot \hat{\mathbf{r}} & \mathbf{i} \cdot \hat{\mathbf{s}} & \mathbf{i} \cdot \hat{\mathbf{t}} \\ \mathbf{j} \cdot \hat{\mathbf{r}} & \mathbf{j} \cdot \hat{\mathbf{s}} & \mathbf{j} \cdot \hat{\mathbf{t}} \\ \mathbf{k} \cdot \hat{\mathbf{r}} & \mathbf{k} \cdot \hat{\mathbf{s}} & \mathbf{k} \cdot \hat{\mathbf{t}} \end{pmatrix} \quad (17)$$

The fixed frame velocity and acceleration vectors are obtained by successive time differentiation of Eq. 16,

$$\mathbf{v}^{ijk} = \frac{1}{k} \left[\left(\dot{r} \mathbf{R} + r \dot{\mathbf{R}} \right) \mathbf{r}^{rst} + r \mathbf{R} \mathbf{v}^{rst} \right] \quad (18)$$

$$\mathbf{a}^{ijk} = \frac{1}{k} \left[\left(\ddot{r} \mathbf{R} + 2\dot{r} \dot{\mathbf{R}} + r \ddot{\mathbf{R}} \right) \mathbf{r}^{rst} + \left(2\dot{r} \mathbf{R} + 2r \dot{\mathbf{R}} \right) \mathbf{v}^{rst} + r \mathbf{R} \mathbf{a}^{rst} \right] \quad (19)$$

where $\dot{r} = dr/dt$ and $\ddot{r} = d^2r/dt^2$. The inverse transformation that provides expressions for $\mathbf{r}^{rst}$, $\mathbf{v}^{rst}$, and $\mathbf{a}^{rst}$ in terms of $\mathbf{r}^{ijk}$, $\mathbf{v}^{ijk}$, and $\mathbf{a}^{ijk}$ is readily available. Note that the higher order time derivatives for r and $\mathbf{R}$ will require up to a first order time derivative in the relative acceleration vector between CB_j and CB_i . If this information is not available from the ephemeris, then it will need to be estimated with any finite difference approximation provided at least the time dependent position vectors are available from the ephemeris.

In either the rotating or the rotating-pulsating frame, it becomes possible to define initial state components for any segment and to target and treat as constraints the state components or functions of them at either node of any

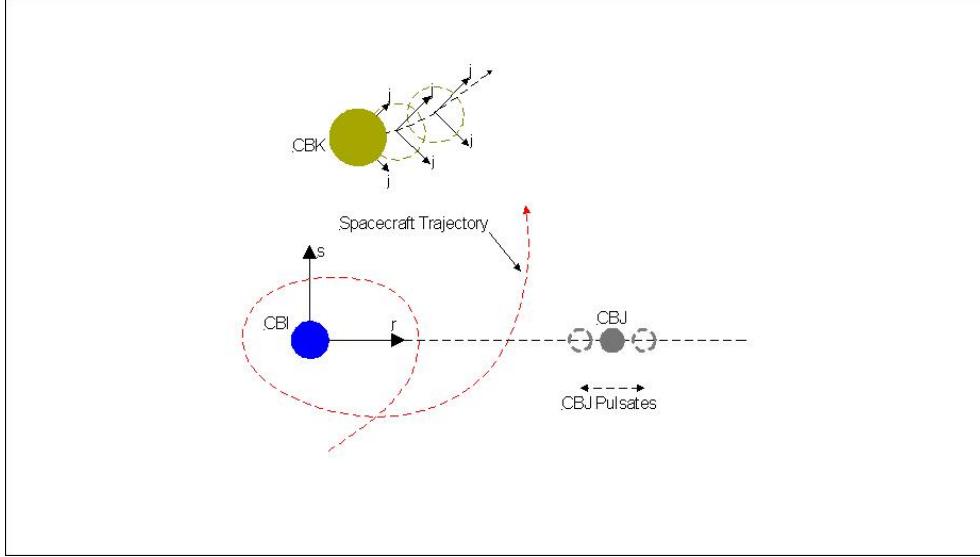


Figure 6: The CB_i - CB_j Rotating Frame with respect to a CB_i Fixed Centered Frame. CB_j pulsates along the line connecting CB_i and CB_j if their absolute motion is non-circular with respect to each other.

segment. For example, finding an equilibrium point in a rotating-pulsating frame centered at the Earth with the Moon along the $\hat{\mathbf{r}}$ direction requires a three dimensional search for the position components $\mathbf{r}^{rst}$ with the conditions that $\mathbf{v}^{rst} = \mathbf{a}^{rst} = \mathbf{0}$, the definition of an equilibrium point. If there are no external accelerations beyond those from the gravitational acceleration due to the Earth and the Moon, then this equilibrium point will remain fixed in the rotating-pulsating frame. If external accelerations are present, such as the gravitational acceleration due to the Sun, then this equilibrium point can be only defined for a particular epoch.

2.5 State Transformations

Recall that all segment propagations are made in either a barycentered or celestial body fixed frame. It is required that a segment's initial state vector and all other quantities that define it be referenced to the segment's fixed reference frame used in its propagation. It is convenient for certain problems to express these quantities in any other reference frame or coordinate set. If this is the case, a suitable transformation to the segment's propagation frame is necessary. Consider, for example, a segment i in a gravity field composed of the celestial bodies, CB_i and CB_j . Celestial body CB_i can be defined to be CB_f , the center of the integration frame, and the motion of CB_j is assumed known relative to CB_f . The initial state of the spacecraft can be referenced to any of the bodies. If it is referenced to CB_j , the required transformation is

$$\begin{pmatrix} (\mathbf{r}_0)_{CB_j} \\ (\mathbf{v}_0^-)_{CB_j} \end{pmatrix}^i \rightarrow \begin{pmatrix} (\mathbf{r}_0)_{CB_f} \\ (\mathbf{v}_0^-)_{CB_f} \end{pmatrix}^i \quad (20)$$

If there exists a more convenient vector or parametric representation of the initial state vector, say the classical Keplerian orbital elements of the spacecraft

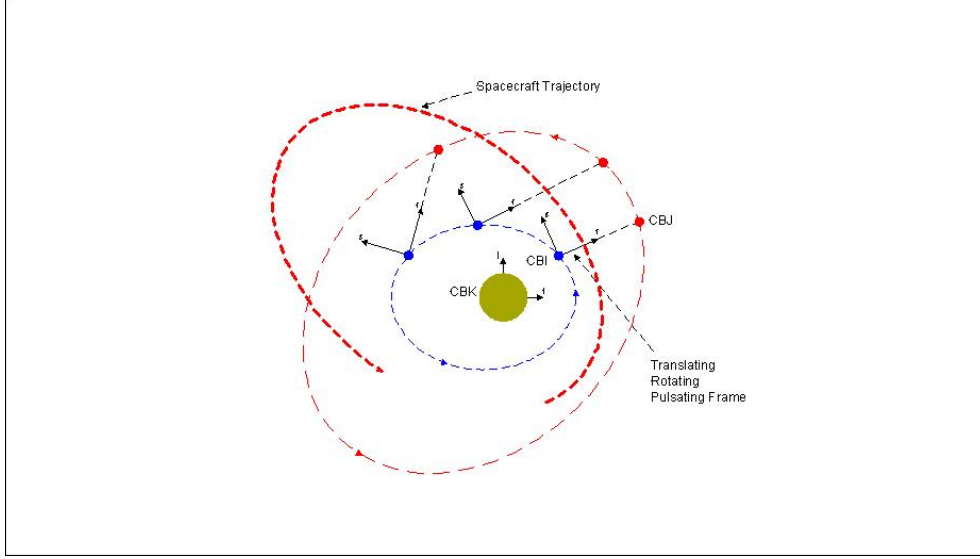


Figure 7: The CB_i - CB_j Rotating Pulsating Frame with respect to a CB_k Fixed Centered Frame. The two celestial bodies, CB_i and CB_j , do not orbit in closed orbits about their common barycenter.

with respect to CB_j at t_0^i , then the required transformation is

$$\begin{pmatrix} a \\ e \\ i \\ \Omega \\ \omega \\ \nu \end{pmatrix}_{CB_j}^i \rightarrow \begin{pmatrix} (\mathbf{r}_0)_{CB_j} \\ (\mathbf{v}_0^-)_{CB_j} \end{pmatrix}^i \rightarrow \begin{pmatrix} (\mathbf{r}_0)_{CB_f} \\ (\mathbf{v}_0^-)_{CB_f} \end{pmatrix}^i \quad (21)$$

where $(a, e, i, \Omega, \omega, \nu)_{CB_j}^{i\top}$ is a vector whose components are the semi-major axis, eccentricity, inclination, right ascension of the ascending node, argument of periapsis, and true anomaly of the spacecraft at t_0^i relative to CB_j . Generally, it can be assumed that an initial state vector, $\mathbf{y}(t_0^i)$, is comprised of any independent set of quantities that uniquely represent the initial state vector of the spacecraft. Ultimately, the transformation $\mathbf{y}(t_0^i)_{CB_j} \rightarrow (\mathbf{r}_o^i, \mathbf{v}_o^i)_{CB_f}$ will be needed for any independent state vector representation with respect to any celestial body CB_j . There are many possible parameterizations for $\mathbf{y}(t_0^i)_{CB_j}$ including those that contain departure, capture, or flyby parameters used in practice such as the hyperbolic excess velocity v_∞ or the square of its value referred to as twice the hyperbolic energy, C_3 , and the direction of the incoming or outgoing asymptote, $\alpha_{v_\infty}, \delta_{v_\infty}$.

2.6 Open Loop Solution of a Trajectory or a Mission

Given the number of segments and their defining parameters, the open loop solution of the system can be determined. The open loop solution is the first estimate or iterate which will generally be infeasible since the constraints or boundary conditions are not necessarily satisfied. The segments are all propagated independently unless a particular segment is required to be connected

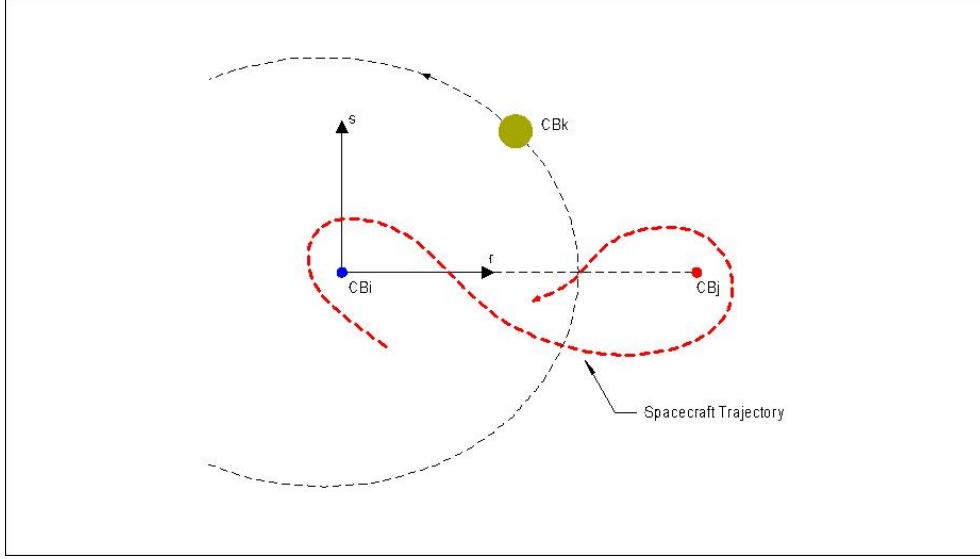


Figure 8: The CB_i - CB_j Rotating Pulsating Frame with respect to a CB_i Fixed Centered Frame. CB_i and CB_j remain fixed in this frame.

sequentially to any other segment in which case the segments that are to be continued with other segments are propagated after the segments to which they are connected to.

If the segments are initially disconnected, i.e., discontinuous in any of the state variables and possibly the time, and these segments form a trajectory for a single spacecraft, the segments will have to be connected by the solution process. Some of the parameters defining these segments will be variables that can be adjusted, and constraint conditions are imposed as continuity conditions for values of all or either the position, velocity, mass, and time. For a multiple spacecraft mission where several of the spacecraft need to rendezvous with other spacecraft, some of the segments will have to be connected either in position or velocity, or both, at the appropriate nodes.

If a segment k is forced to be connected sequentially to any other node of any other segment i beforehand, the connectivity is enforced at any of the discrete node times, $t_{(0,f)}^{i-}$, $t_{(0,f)}^{i-+}$, $t_{(0,f)}^{i+}$, $t_{(0,f)}^{i+}$, i.e., the continuity of the position, velocity, and mass of segment i can be made prior to or after the initial or final impulsive maneuvers or any of the mass discontinuities, if they exist. For example, if a purely ballistic segment k is required to be connected sequentially in all three state variables ($\mathbf{r}, \mathbf{v}, m$) to any other segment i in the model at the t_f node after all, if any, of the velocity and mass discontinuities, the state vector of the t_0 node of segment k is then

$$\begin{aligned} \mathbf{r}_0^k &= \mathbf{r}_{(0,f)}^i \\ \mathbf{v}_0^{k-} &= \mathbf{v}_{(0,f)}^{i+} \\ m_0^{k-} &= m_{(0,f)}^{i+} \end{aligned} \tag{22}$$

Here then, it is necessary that segment i be evaluated entirely before segment k can be evaluated. On the other hand, if the final solution requires that segments i and k be connected as in Eq. 22, these can be initially defined

independently and Eq. 22 is used as an equality constraint condition that must be satisfied by the final solution. Continuity is not allowed at any time between t_0^+ and t_f^- because only node points can be constrained. But this is not a restriction, because an additional segment is introduced such that one of its nodes can be used as the continuation to another segment.

This generality facilitates the modeling of complex trajectories that may be otherwise difficult to obtain as a single forward time or backward time propagated trajectory and in the modeling of multi-spacecraft problems where the other spacecraft are moving along different trajectories.

3 Solution Methods

For any given problem, that in general can be complex and involve multiple constraints, it is possible that no solution exists. A common reason for this is that the constraints are inconsistent meaning the problem definition itself precludes the possibility of a solution. Therefore, a necessary requirement is to formulate a consistent problem where the variables of the problem are independent and the constraints are at least linearly independent. If it is then assumed that a solution should exist, there are two possible solution methods. These methods provide solutions to the majority of the spacecraft trajectory design and optimization problems considered by the current architecture. This section describes these methods which are used to solve trajectory problems that are based on the set of basic segments.

Let n be the number of segments in the trajectory or the mission. Let the segment vector, $\mathbf{s}^i$, be a vector whose components are all of the independent and dependent quantities that uniquely define segment i . Within an individual segment, some of these quantities will be either dependent or independent. For example, m_f^+ , the vehicle mass at t_f^+ , is a dependent quantity that depends on any impulsive maneuver at t_0 and/or t_f , any mass depleting finite maneuver between t_0^+ and t_f^- , and any non-maneuver mass discontinuity that may be present at either or both nodes. But m_0^- is an independent quantity (within the segment), though it may be constrained outside of the segment during the solution process. Depending on the target conditions required to be satisfied for a given mission, further information can be computed from $\mathbf{s}^i$ that can be used as constraints or as a cost function. For example, the state vector at any of the discrete times of the node point relative to any of the other celestial bodies that are present or any function of these can be computed provided the appropriate transformations are available. A common function is the two-body energy of a segment endpoint with respect to a celestial body.

Let $\mathbf{x}_p$ be a p vector containing all of the variable parameters that have been identified for any given problem. The parameter vector $\mathbf{x}_p$ can contain any independent element of any or all of the segments identified in $\mathbf{s}^i (i = 1, \dots, n)$. It can also contain any parameter that defines the force model. Let $\mathbf{c}$ be a q vector containing all of the equality and inequality constraint functions. The constraint vector $\mathbf{c}$ is in general a nonlinear function of any of the elements in $\mathbf{x}_p$. Let J be a general nonlinear scalar function of any of the elements in $\mathbf{x}_p$. The value of J is to be extremized. A description of the two solution methods and their implementation into the current architecture follow.

3.0.1 Case 1: Solution of a System of Nonlinear Equations, $\mathbf{c}(\mathbf{x}_p)=\mathbf{0}$.

If the number of variables in $\mathbf{x}_p$ is equal to the number of constraint equations $\mathbf{c}$ ($p = q$) and all of the elements in $\mathbf{c}$ are equality constraints then the problem to solve is a targeting problem with no explicit optimization, i.e., no explicit scalar function is extremized. However, this does not exclude the possibility of implicitly extremizing a cost function if the equality constraint functions include necessary conditions based on optimal control theory.

The targeting is completely general and the solution for $\mathbf{x}_p$ is a solution to a general nonlinear root finding problem for a system of nonlinear functions or equations with an equal number of unknown parameters. Functionally, the problem is to solve the set of equations $\mathbf{c}(\mathbf{x}_p) = \mathbf{0}$. It is assumed that the functions $\mathbf{c}(\mathbf{x}_p)$ be smooth, i.e., at least twice-continuously differentiable, in the independent variable vector $\mathbf{x}_p$. Though in general, these equations are not available explicitly, the functions given in $\mathbf{c}(\mathbf{x}_p)$ are determined from a sequence of operations that depend on the specific model and problem. These operations will involve numerical propagations of the segments, transformations between reference frames, and function evaluations of the elements in the constraint vector, $\mathbf{c}$.

The system can use any efficient and robust algorithm to solve the nonlinear root finding problem. The current prototype system uses a Newton-Raphson/Steepest descent correction to compute a search direction and is coupled with Broyden's method for computing and updating the Jacobian matrix of the system, $\partial\mathbf{c}/\partial\mathbf{x}_p$. The analytical and numerical issues associated with this and similar methods can be found in Dennis and Schnabel[11], Gill et. al[12], and Nocedal and Wright[13].

The two-body orbital boundary value problem, which is known as the Lambert Problem if the gravitational force model includes only one celestial body with no other perturbations is a subset of this problem. The orbital boundary value problem, for any general gravitational force field which can include more celestial bodies, or a complex potential gravity field for one or more celestial bodies, or other perturbations, is a two point boundary value problem which is also a subset of this case. Other specific problems that can be solved as a nonlinear root finding problem in orbital mechanics include periodic orbit searches, Earth-to-Moon trajectories, including free return trajectories, multi-body gravity assisted trajectories in the Solar System, etc.

The functional optimization or optimal control problem for either impulsive maneuvers, finite engine burn maneuvers, or non-mass depleting acceleration controlled maneuvers using optimal control theory is also a subset of this case. Here the time varying Lagrange multipliers associated with each of the physical state variables augment the $\mathbf{s}^i$ vector for each segment, and the first order necessary conditions and the transversality conditions lead to a well defined and in general multi-point boundary value problem. If solved as a nonlinear root finding problem, the problem is an implicit optimization. An extremal solution that satisfies the first order necessary conditions from optimal control can be produced by solving the appropriate nonlinear system of functions. This is the well known indirect method for trajectory optimization and has been well documented in the literature. A comprehensive treatment of the optimal control problem and its solution methods is given by Hull[14].

Since the system of equations is in general nonlinear, multiple solutions

can be expected. If the functional optimization problem is solved indirectly, the solution provided by the nonlinear search will satisfy first order necessary conditions for an optimal solution. If the general targeting problem is solved without the use of the functional optimization conditions, then the solution is only a feasible solution.

It will always be necessary to provide a reasonable estimate for the starting iterate given by $\mathbf{x}_p$. For simple problems, this starting guess can be estimated analytically. For more complex problems, it may be necessary to use results available from previous research in specific problems. For example, circular restricted three body trajectories are well understood though closed form solutions for these do not exist. For force models that are not too different from a simplified circular restricted three body force field model, the results from the simple circular restricted three body model can be used as starting iterates for more complex models.

If some of the elements in the constraint vector, $\mathbf{c}$, for a targeting problem are inequality constraints, these can be converted to equality constraints by adding one more variable to the parameter vector $\mathbf{x}_p$ known as a slack variable such that the final problem is an equality constrained problem and solved as such (Ref.[14]).

The underdetermined or overdetermined cases where there are more or less variables, respectively, than equality constraints is solved as a minimax problem (Ref. [12]) commonly used for nonlinear minimax data-fitting, where the constraint function with the maximum absolute value is minimized. The minimax solution to the system of nonlinear equations

$$\mathbf{c}(\mathbf{x}_p) = \mathbf{0}$$

is the solution that minimizes the function, F defined by

$$F(\mathbf{x}_p) = \max |c_i(\mathbf{x}_p)|$$

where c_i is the i -th element of the constraint vector $\mathbf{c}$.

This technique is robust and efficient for the class of nonlinear targeting problems considered here where there is not an explicit cost function and satisfaction of the constraint equations results in a feasible solution. In the underdetermined case, a solution to the minimax problem can be used as the starting estimate for the parameter optimization problem since all of the constraints are satisfied initially.

3.0.2 Case 2: The Constrained Parameter Optimization Problem

This is the general problem of nonlinear functional optimization with nonlinear equality and inequality constraints and whose solution methods have been well documented in the literature (see for example Gill *et al.*[12] and Nocedal and Wright[13]). The problem is to minimize or maximize the objective function

$$J = J(\mathbf{x}_p)$$

subject to both equality and inequality constraints,

$$\begin{aligned} \mathbf{c}_{eq}(\mathbf{x}_p) &= \mathbf{0} & \mathbf{c}_{eq} : q_{eq} \times 1 \\ \mathbf{c}_{ineq}(\mathbf{x}_p) &\geq \mathbf{0} & \mathbf{c}_{ineq} : q_{ineq} \times 1 \end{aligned}$$

where $\mathbf{c}_{eq}$ is the q_{eq} vector of equality constraints and $\mathbf{c}_{ineq}$ is the q_{ineq} vector of inequality constraints. The functions $\mathbf{c}_{eq}(\mathbf{x}_p)$ and $\mathbf{c}_{ineq}(\mathbf{x}_p)$ can represent lower and upper bounds on the individual elements in $\mathbf{x}_p$, linear constraints on two or more of the elements in $\mathbf{x}_p$, or more generally, nonlinear functions of the elements of $\mathbf{x}_p$. The algorithm used in the current system is a sequential quadratic programming (SQP) variable metric method described in Ref.[12] and Ref.[13].

With the given architecture, the formulation of many parameter optimization problems is straightforward. The objective function to extremize (minimum or maximum) can be taken to be a single scalar variable in $\mathbf{s}^i (i = 1, \dots, n)$ or any scalar function of these elements. For example, a trajectory that begins in an orbit about CB_i and terminates in an orbit about CB_j can be such that the initial mass of the spacecraft is a minimum if the final mass has been prescribed. For this case $J = \min(m_0^{i-})$ where i is first segment of the trajectory representing the spacecraft. The value of the initial mass required is a function of all the segments following it including all of the possible impulsive or finite burn maneuvers used to reach the final orbit, and any non maneuver mass discontinuities. The constraints associated with the initial orbit and the final orbit would be the nonlinear constraint functions in $\mathbf{c}$, and $\mathbf{x}_p$ would contain the parameters that can be estimated such as the time of the maneuvers and the parameters that define the maneuver and that are allowed to be adjusted.

4 General System Issues

Several main aspects important to a general trajectory design and optimization system are discussed in this section. This includes the benefits of using a system that is modular in all of its subcomponents, the importance of tuning the algorithms and automating the tuning process, and the use of integrated visualization as a key part of the solution process.

4.1 Modular Architecture

A general trajectory optimization system should be modular in the sense that the algorithms used to solve different parts of the overall trajectory problem are independent and can be readily modified or replaced. The basic components include algorithms for explicit numerical integration, solutions to systems of nonlinear equations, and solutions of the nonlinear constrained parameter optimization problem. The individual algorithms used for each of these functions should be the ‘best available’ and with the added flexibility of being updated or changed entirely as new and more robust and efficient algorithms are developed. Provisions should also be made to allow completely general force models to be used including the use of either realistic ephemeris models or user defined models. In summary, the assumption is to treat each algorithm to each sub-problem as ‘solved’. The continued development should be directed to the system architecture and all that it entails such as refinement of the definition of the basic segment, the maneuver models, and the coordinate frames for state definition, targeting, and maneuver parameterization.

4.2 Algorithm and System Tuning

Provided an estimate for a convergent solution is available for both the non-linear equation solvers or the parameter optimization algorithms based on gradient information, there exists a sequence of perturbations step sizes for the parameter vector, $\mathbf{x}_p$, that achieves a solution in a minimum number of function evaluations for the constraint vector, $\mathbf{c}$, or both $\mathbf{c}$ and the cost function, J . If the method of finite differences are being used to estimate gradients, typically the vector of perturbation step sizes, $\delta\mathbf{x}_p$, is set as a constant during the iteration process. Regardless of the method used to compute the finite difference based gradients, determining the ‘best’ value for the perturbation vector is a process referred to as tuning and which is problem, algorithm, and processor dependent. A given value of $\delta\mathbf{x}_p$ that provides accurate estimates for the gradients at the beginning of an iteration process may not be the best choice at other parts of the search space, therefore this vector should be allowed to change and recomputed periodically to produce accurate gradients at different points in the search space.

Any system should have in place an automatic tuning algorithm that can adjust the perturbation vector over the course of the iteration process to achieve convergence using the minimum amount of function and constraint vector evaluations. Included in the tuning process is the proper scaling of the parameter vector and the constraint functions if their values differ by large orders of magnitude. Also, a change to each of the elements in the parameter vector $\mathbf{x}_p$ per iteration should be bounded to avoid evaluating subsequent iterates that cause the solution to diverge. This is common in trajectory problems where the endpoints of a trajectory operating in a complex force field are highly sensitive to changes in the parameters of earlier parts of the trajectory. But for well behaved solutions, if this upper bound is too small, many iterations will be needed for convergence. The proper choice of $\delta\mathbf{x}_p$ per iteration, the scaling, and the maximum allowable changes to $\mathbf{x}_p$ per iteration, are three factors that influence the convergence rate of a solution and have to be considered carefully in an automated tuning process.

A procedure that leads to accurate derivative estimates is based on the calculation of the state transition matrix (Ref. [7]) along the ballistic or controlled accelerated arcs for all segments between t_0^+ and t_f^- and for the variables that are numerically integrated. The state transition matrix is the time varying fundamental matrix solution of the linearized variational equations evaluated along the trajectory arcs. If the state vector $\mathbf{x}(t)$ is composed only of the physical state variables $\mathbf{r}, \mathbf{v}, m$ the state transition matrix at t_f^- for a given statement provides the Jacobian matrix of the final state with respect to the initial state. It is a 7×7 matrix with the gradients $\partial\mathbf{x}(t_f^-)/\partial\mathbf{x}(t_0^+)$. If the Lagrange multiplier vector is part of the state vector, so that all of the variables $\mathbf{r}, \mathbf{v}, m, \lambda_r, \lambda_v, \lambda_m$ are numerically integrated along a trajectory arc, then the state transition matrix will be a 14×14 matrix. The state transition matrix is integrated along with the state vector thereby requiring the integration of $n + n^2$ first order equations where n is the size of the state vector. The information available in the state transition matrix represents some of the terms necessary to compute the required gradients.

Given a constraint vector, $\mathbf{c}$, or a scalar objective function J , that depends on an independent parameter vector, $\mathbf{x}_p$, there exists analytical expressions

for both the gradient vector, $\partial J/\partial \mathbf{x}_p$, and the Jacobian matrix, $\partial \mathbf{c}/\partial \mathbf{x}_p$, that are linearly valid near any nominal solution of a segment, trajectory, or mission. These expressions are based on state transition matrices along all of the segments, and the gradients across impulse points, discontinuous mass points, and any state and maneuver transformations used. Though numerical integration of the state vector and the state transition matrix is required to evaluate the quantities necessary to evaluate both $\partial J/\partial \mathbf{x}_p$ and $\partial \mathbf{c}/\partial \mathbf{x}_p$, these provide accurate values at $\mathbf{x}_p$ necessary for any gradient based solution method. The disadvantage of this process is that it is problem specific and requires the derivation of these expressions for each type of problem, a time consuming process if many classes of problems are to be considered. Nevertheless, these expressions can be evaluated and used as the actual gradients necessary in the solution methods. Alternatively, these expressions can be evaluated at discrete points in the iteration process, and used to tune the perturbation vector $\delta \mathbf{x}_p$ at these points. The perturbation vector is then used to estimate the derivatives with a finite difference approximation.

4.3 Integrated Visualization

A general trajectory design and optimization system should include interactive visualization capability not only for presentation of intermediate and final results but as a key part of the solution process. The system currently provides the capability to visually display a three dimensional graphics representation of the trajectory design and optimization process in real processor time; i.e. immediate visual feedback during the targeting or optimization process is available. The reference frame used to visualize the dynamics and solution process can be independent from the reference frames used in the trajectory problems. The information shown in the graphics representation typically includes the positions of each point in the trajectory. However, other phase space variables can be visualized as well if the visualization of their evolution provides better information.

Though visualization has generally not been accepted as a necessary capability in the solution process, it is used here as a critical component in the design and optimization process for several reasons. First, it is immediately possible to determine if there is an error in the input information thereby allowing for the termination of the process and correction of the error. Secondly, it is possible to determine whether the solution is converging, and if so, the rate of convergence. If convergence is slow, the speed of convergence can be increased by re-tuning the algorithms. Thirdly, an intuitive understanding of the dynamics associated with complex trajectories in multi-body gravity fields or rotating frames can be obtained, thereby facilitating future investigation of more difficult problems. All of these benefits can be obtained from a non-graphics based system in a post execution manner by examining the output data generated and plotting any desired information. But clearly, the time saved by not having to perform these post execution tasks is a reason that supports the use of integrated graphic visualization in the solution process.

5 Conclusions

The key aspects associated with a general trajectory optimization system have been presented. The continuing effort is an attempt to construct a system that can be used to analyze a large range of problems that are currently required in trajectory design and optimization. Future spacecraft missions will use innovative trajectory dynamics that take advantage of the natural and complex dynamics associated with multi-body gravity fields, multiple and hybrid propulsion systems, and multiple spacecraft. A main objective is the development of a system that can facilitate the solution to these problems using a general framework that is applicable to all of the sub-problems required for the overall solution. Such a system should produce results that can be used in research topics involving both trajectory optimization theory and numerical methods for the solution to these problems. The system should also be able to produce results that can be used in actual spacecraft operations by accounting for actual mission operation constraints; detailed propulsion system models; attitude control, constraints, and requirements; and some level of information regarding the observability of the solutions. A deterministically optimal solution based on propellant or mass performance only may not be the best solution given a measurement model to observe the state of the spacecraft. The capability to produce an integrated solution that offers an acceptable compromise between performance and the ability to accurately navigate it is desirable.

Caution must be exercised in not having a system so general that it solves many problems only superficially and with many restricting assumptions. There will always be a balance between how general a system is and how detailed and complex the solutions produced by such a system are for any given problem. A measure of the usefulness of such a system is the scope of the problems it can solve while solving them to the level of detail needed to be valid enough. A useful system should produce solutions of practical interest to spacecraft missions that will require only minor adjustments when and if the mission is actually flown.

No claim has been made concerning the superiority of the described architecture over existing architectures and systems. The only claim made is that the given architecture facilitates the modeling and optimization of many classes of trajectories. The system remains evolutionary, meaning that it can be changed and enhanced as needed to address problems whose requirements can not currently be met with the current system architecture. If this is the case, then it is postulated that the majority of the changes needed will be directed to the modification and enhancement of definition of the basic segment; and not so much in the solution methods.

References

- [1] Gómez, G., Koon, W.S., Lo M.W., Marsden, J.E., Masdemont, J., Ross, S.D., "Invariant Manifolds, The Spatial Three-Body Problem and Space Mission Design", *Proceedings of the First International Conference on Libration Point Orbits and Applications*, Parador d'Aiguablava, Girona, Spain, June 2002.

- [2] Azimov, D.M., Bishop, R.H. “Extremal Rocket Motion with Maximum Thrust in a Linear Central Field”, *Journal of Spacecraft and Rockets*, Vol. 38, No. 5, Sep-Oct 2001.
- [3] Whiffen, G.J., Sims, J.A., “Application of the SCD Optimal Control Algorithm to Low-Thrust Escape and Capture Trajectory Optimization”, *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, San Antonio, Texas, February 2001.
- [4] Lawden, D.E., *Optimal Trajectories for Space Navigation*, London Butterworths, 1963.
- [5] Betts, J.T., “Survey of Numerical Methods for Trajectory Optimization,” *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, March-April 1998.
- [6] Hull, D.G., “Conversion of Optimal Control Problems into Parameter Optimization Problems”, *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, Jan-Feb 1997.
- [7] Battin, R.H., *An Introduction to the Mathematics and Methods of Astrodynamics*, Revised Edition, AIAA Education Series, American Institute of Aeronautics and Astronautics, Reston, Virginia, 1999.
- [8] Belbruno, E.A., Miller, J.K., “Sun-Perturbed Earth-to-Moon Transfer with Ballistic Capture”, *Journal of Guidance, Control, and Dynamics*, Vol. 16, No.4, 1993.
- [9] Byrnes, D.V., Bright, L.E., “Design of High-Accuracy Multiple Flyby Trajectories Using Constrained Optimization”, *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, Halifax, Nova Scotia, Canada, 1995.
- [10] Szebehely, V., *Theory of Orbits*, Academic Press, 1967.
- [11] Dennis, J.E., Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983.
- [12] Gill, P.E., Murray, W., Saunders, M.A., Wright M.H., *Practical Optimization*, Cambridge University Press, 1998.
- [13] Nocedal, J., Wright, S.J., *Numerical Optimization*, Springer Series in Operations Research, Springer, 1999.
- [14] Hull, D.G., *Calculus of Differentials and Optimization*, Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, www.ae.utexas.edu/~dghull, 2002.

THEORETICAL FOUNDATION OF COPERNICUS: A UNIFIED SYSTEM FOR TRAJECTORY DESIGN AND OPTIMIZATION

Cesar Ocampo¹, Juan S. Senent², and Jacob Williams³

¹*Department of Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin, Austin, Texas*

²*Odyssey Space Research, Houston, Texas*

³*ERC, Inc. Engineering and Science Contract Group, Houston, Texas*

ABSTRACT

The fundamental methods are described for the general spacecraft trajectory design and optimization software system called Copernicus. The methods rely on a unified framework that is used to model, design, and optimize spacecraft trajectories that may operate in complex gravitational force fields, use multiple propulsion systems, and involve multiple spacecraft. The trajectory model, with its associated equations of motion and maneuver models, are discussed.

Key words: Copernicus, Spacecraft Trajectory Optimization Software.

1. INTRODUCTION

Copernicus [17] is a general trajectory design and optimization system intended to solve a wide range of spacecraft trajectory problems in a robust and efficient manner. It facilitates the design and optimization of simple to complex spacecraft trajectories. Some examples from recent Project Constellation studies include: trans-lunar [3], trans-Earth [16] and trans-lunar abort return trajectories [14]. Interplanetary mission applications include: design of a 10 year 32-asteroid tour using a low-thrust propulsion system, Mars sample return missions and trajectory design for the Lunar Crater Observation and Sensing Satellite (LCROSS).

The three main problems addressed by the system are:

1. **Modelling and Open Loop Simulation:** The system can simulate open loop trajectories in any force field, using one or more spacecraft, and including impulsive and/or finite burn maneuvers.
2. **General Targeting via Nonlinear Root Finding:** The system can search for the values of the independent variables required to satisfy a set of constraint functions. This is required for trajectory targeting problems, such as orbital boundary value problems. But

more generally, it is possible to formulate and solve general systems of nonlinear equations that depend on a preselected set of independent variables.

3. **Optimization:** The system can extremize the value of a function consistent with the constraints. Any computable function can be extremized. Typically, the optimization mode of the system is used for maneuver design and optimization. This includes determining the times and parameters that describe the maneuvers, which can be impulsive or finite burns.

A key component of the system is the *trajectory model* that describes how a complete trajectory for a single or set of spacecraft is modelled. The supporting components of the trajectory model include the equations of motion, the propulsion system, the control parametrization, the independent variables, and the constraint functions. This paper describes and summarizes some of the theoretical aspects of these components. Some of these along with the details of the system architecture have been documented previously as the development of the system has evolved over the years [8, 9, 10]. A discussion on future extensions and proposed formulations for the finite burn maneuver modeling is also given.

2. TRAJECTORY MODEL

The trajectory model uses a building block called a *segment*. A segment is a trajectory arc that can have velocity impulses and/or a finite burn maneuver. Depending on how the relationship between different segments is defined, single or multiple spacecraft trajectories that may interact can be modeled. The generality of how segments are related to one another facilitates the modeling of complex missions. The simplest mission may have only one segment. Complex missions may have tens or hundreds of segments.

The segment arc is bracketed by two node points. The node points are tagged with epochs that are referenced to a reference epoch, t_{epoch} . The epoch of the initial node is t_0 and the epoch of the final node is t_f . There is no

restriction on the values for t_0 and t_f ($t_0 = t_f$, $t_0 < t_f$, $t_0 > t_f$). The segment duration is $\Delta t = t_f - t_0$. The segment nodes are uniquely defined by specifying any of the pairs (t_0, t_f) , $(t_0, \Delta t)$, $(\Delta t, t_f)$.

The segment state vector $\mathbf{x}$ is

$$\mathbf{x} = \begin{pmatrix} \mathbf{r}^\top & \mathbf{v}^\top & m \end{pmatrix}^\top. \quad (1)$$

The initial position vector is $\mathbf{r}_0$, the initial velocity vector is $\mathbf{v}_0^-$, and the initial mass is m_0^{--} . Both $\mathbf{v}$ and m can have discontinuities at the initial node. For velocity, the discontinuity is an impulsive maneuver $\Delta \mathbf{v}_0$. For mass, the discontinuities are due to the propellant mass consumed by the impulsive maneuver, and independent mass discontinuities before and after the impulsive maneuver. The independent mass discontinuities can be used to represent acquisition or removal of other spacecraft, and/or stage masses. After these discontinuities are accounted for, the segment is propagated from t_0 to t_f if $t_0 \neq t_f$. At the t_f node, velocity and mass have the same type of discontinuity as at the t_0 node. The propagation direction is considered as forward if $t_0 < t_f$ and backward if $t_0 > t_f$. If $t_0 = t_f$, then the propagation direction is specified to properly account for how the velocity and mass discontinuities are used. Let $i_d = +1$ for a forward propagation and $i_d = -1$ for a backward propagation.

Across either the t_0 or the t_f node, position remains constant. The velocity at the end of the node is

$$\mathbf{v}^+ = \mathbf{v}^- + (i_d) \Delta \mathbf{v} \quad (2)$$

The mass across the node evolves as

$$m^{--} \rightarrow m^{-+} \rightarrow m^{+-} \rightarrow m^{++}$$

Any one of these is specified, and the remaining three are determined by knowing the propagation direction, the independent mass discontinuity Δm^- before the impulse, the independent mass discontinuity Δm^+ after the impulse, the magnitude of the impulsive maneuver, and the exhaust velocity c of the engine doing the impulsive maneuver. Define the factor

$$f = e^{-\Delta v/c} \quad (3)$$

There are two main cases:

1. m^{--} or m^{-+} is specified; depending on which one is specified, the other is solved by using

$$m^{-+} = m^{--} + \Delta m^- \quad (4)$$

then

$$m^{+-} = (m^{-+}) (f)^{i_d} \quad (5)$$

$$m^{++} = m^{+-} + \Delta m^+ \quad (6)$$

2. m^{+-} or m^{++} is specified; depending on which one is specified, the other is solved by using

$$m^{++} = m^{+-} + \Delta m^+ \quad (7)$$

then

$$m^{-+} = (m^{+-}) (f)^{-i_d} \quad (8)$$

$$m^{--} = m^{-+} - \Delta m^- \quad (9)$$

As stated before, the segment initial state vector is

$$\mathbf{x}_0^- = \begin{pmatrix} \mathbf{r}_0^\top & \mathbf{v}_0^{-\top} & m_0^{--} \end{pmatrix}^\top. \quad (10)$$

After accounting for the potential velocity and mass discontinuities at the t_0 node, the initial condition for the state vector that is propagated from t_0 to t_f (if $t_0 \neq t_f$) is

$$\mathbf{x}_0^+ = \begin{pmatrix} \mathbf{r}_0^\top & \mathbf{v}_0^{+\top} & m_0^{++} \end{pmatrix}^\top. \quad (11)$$

The state vector after the propagation to t_f is

$$\mathbf{x}_f^- = \begin{pmatrix} \mathbf{r}_f^\top & \mathbf{v}_f^{-\top} & m_0^{--} \end{pmatrix}^\top. \quad (12)$$

The final value of the state vector after all possible velocity and mass discontinuities are accounted for at the t_f node is

$$\mathbf{x}_f^+ = \begin{pmatrix} \mathbf{r}_f^\top & \mathbf{v}_f^{+\top} & m_f^{++} \end{pmatrix}^\top. \quad (13)$$

Sets of segments are used to construct complete trajectories for one or more spacecraft. Segments can be connected to other segments by forcing the complete state or a subset of it to inherit the complete state or a subset of it from either node of another segment. Segments can also be completely disconnected to model independent trajectories representing other spacecraft, for example. These segments can later be constrained to be connected via continuity constraints to other segments if needed. A mission consists of the set of all segments.

A mission has a *base frame* which is required to be non-rotating and centered at a celestial body. All state transformations use the base frame as a hub through which all state, maneuver, and state function transformations are made. Each segment has a *state input frame*, a *propagation frame*, and a *function output frame*. The propagation frame is the frame used for the integration of the equations of motion. It is required to be non-rotating and centered at a celestial body. The initial position and velocity state data $(\mathbf{r}_0, \mathbf{v}_0^-)$ is given in the state input frame. The function data for a segment includes all functions that can be computed from the distinct segment states $(\mathbf{r}_0, \mathbf{r}_f, \mathbf{v}_0^-, \mathbf{v}_0^+, \mathbf{v}_f^-, \mathbf{v}_f^+)$, and so on) and are computed in the function output frame. The state input and function output frames are required to be centered at a celestial body and can be either fixed or rotating. Each segment is allowed to have its own state input, propagation, and function output frame. A simple mission may have all of these frames be the same.

The individual segment impulsive maneuvers are each referenced to an *impulsive maneuver frame*. The impulsive maneuver frame can be either fixed (non-rotating) or osculating along the instantaneous trajectory path. In

either case, the impulsive maneuver frame is referenced to a frame centered at a celestial body. A fixed impulsive maneuver frame is typically the same as the segment propagation frame. Alternatively, an example of a common osculating impulsive maneuver frame is one that uses the instantaneous velocity vector as a basis vector. The remaining two basis vectors can be constructed, for example, by choosing the instantaneous angular momentum vector, if it exists, as one of the basis vectors.

A segment vector $\mathbf{s}^i$ is defined for each segment i to be a mixed variable type vector containing at least all of the independent and dependent variables and functions associated with the segment. Additionally it contains all the frame definitions, how and what data is inherited from other segments, what functions need to be evaluated from the state vector at the distinct time tags, which variables are search variables, and so on. This list of information is large. As an example, assume that a segment is purely ballistic with impulsive maneuvers at both nodes. A subset of the segment vector $\mathbf{s}^i$ is

$$\mathbf{s}^i = \begin{pmatrix} t_0, \mathbf{r}_0^\top, \mathbf{v}_0^\top, m_0^-, \Delta m_0^-, \Delta \mathbf{v}_0^\top, \Delta m_0^+, \\ t_f, \Delta m_f^-, \Delta \mathbf{v}_0^\top, \Delta m_f^+, \mathbf{r}_f, \mathbf{v}_f^{++}, m_f^{++}, \dots \end{pmatrix}^\top \quad (14)$$

The complete mission with n segments can be computed uniquely in open loop (simulated) by processing all of the information contained in $\mathbf{s}^i$ ($i = 1, \dots, n$).

Every segment i has a variable vector $\mathbf{s}_x^i$ and a function vector $\mathbf{s}_f^i$. $\mathbf{s}_x^i$ is a subset of $\mathbf{s}^i$ and contains only the variables that have been identified to be independent search variables to be determined. $\mathbf{s}_f^i$ is a subset of $\mathbf{s}^i$ and contains only the functions that have been identified to be constrained or added to an objective function.

The state vector for any segment is governed by the first-order vector equation of motion

$$\frac{d}{dt} \begin{pmatrix} \mathbf{r} \\ \mathbf{v} \\ m \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ \mathbf{g}(\mathbf{r}, \mathbf{v}, m, t, \mathbf{p}) + \frac{T(t)}{m} \mathbf{u}(t) \\ -\frac{T(t)}{c(t)} \end{pmatrix}^i \quad (15)$$

where $\mathbf{g}$ is the ballistic acceleration per unit mass, $\mathbf{p}$ is a vector of non-state or non-control parameters such as force model and vehicle parameters, T is the finite engine thrust force, $\mathbf{u}$ is the unit thrust direction, and c is the engine exhaust velocity. T , c , and $\mathbf{u}$ are possible control functions. The information needed to evaluate these functions are also contained in $\mathbf{s}^i$. These equations are solved as an initial value problem from t_0 and t_f . The only allowable discontinuities along the integrated solution are associated with the control terms and only if an optimal control formulation is used to determine the values of the control functions.

The natural ballistic acceleration vector $\mathbf{g}$ is expressed in the segment propagation frame which is centered on a main celestial body c_b and is non rotating. An example

expression for $\mathbf{g}$ is

$$\mathbf{g}(\mathbf{r}, t) = -\frac{Gm_{c_b}}{r^3} \mathbf{r} - G \sum_{j=1}^{n_b} m_j \left(\frac{\mathbf{r} - \mathbf{r}_j(t)}{|\mathbf{r} - \mathbf{r}_j(t)|^3} + \frac{\mathbf{r}_j(t)}{r_j^3(t)} \right) + \mathbf{a}_{drag} + \mathbf{a}_{srp} + \mathbf{a}_{non-spherical} \quad (16)$$

where G is the universal constant of gravitation, m_{c_b} is the mass of the main celestial body, n_b is the number of additional celestial bodies, and m_j is the mass of celestial body j . It is assumed that the positions of the n_b celestial bodies is known as an explicit function of time with respect to c_b . Depending on the problem, additional terms accounting for other common accelerations, such as atmospheric drag ($\mathbf{a}_{drag}$), solar radiation pressure ($\mathbf{a}_{srp}$), and non-spherical celestial bodies ($\mathbf{a}_{non-spherical}$), are added to the vector function $\mathbf{g}$. The equations of motion are integrated numerically with an explicit numerical integration algorithm. The trajectory arc that connects the t_0 node to the t_f node is either ballistic or it is a finite burn arc. If it is a finite burn arc, one of two main finite burn models are used. These are described below.

3. FINITE BURN MANEUVER MODELS

The modelling of a finite burn arc is based on one of two possible models. The first model is the parameter model that uses a finite set of parameters to describe the time evolution of the control variables. This finite set of parameters form part of $\mathbf{s}_x^i$. The second model is the optimal control model which requires augmenting the state vector to include the costate vector which is the vector of Lagrange multipliers adjoined to the state. The initial conditions of the costate vector, or the initial values of a transformation vector used to uniquely determine the initial costate vector, form part of $\mathbf{s}_x^i$.

3.1. Simple Engine Model

The following discussion refers to the trajectory arc for a particular segment between t_0 and t_f ($t_0 \leq t \leq t_f$). The finite burn acceleration is given by the thrust acceleration term

$$\mathbf{a}_{thrust} = \frac{T(t)}{m(t)} \mathbf{u}(t) \quad (17)$$

The mass rate is

$$\dot{m} = \frac{-T(t)}{c(t)} \quad (18)$$

and the minimal set of control constraints are

$$T_{\min} \leq T(t) \leq T_{\max} \quad (19)$$

$$|\mathbf{u}(t)| = 1 \quad (20)$$

$$c(t) > 0 \quad (21)$$

The type of engine being modelled determines which of these are the controls. A constant exhaust velocity engine that can only be turned on and off with $T = T_{\max}$ when it is on has $c(t) = c$ and T is piecewise constant (0 or $T_{\max}$). If the engine can be throttled then $T(t)$ needs to be determined consistent with its lower and upper bounds. A simple model assumes that $T(t) = T$, a constant, with $0 \leq T \leq T_{\max}$.

An engine that is limited by the input power constrains $T(t)$ and $c(t)$ to satisfy

$$P(t) = \frac{1}{2}T(t)c(t) \quad (22)$$

with

$$P_{\min} \leq P(t) \leq P_{\max} \quad (23)$$

along with bound constraints on either $T(t)$

$$T_{\min} \leq T(t) \leq T_{\max} \quad (24)$$

or $c(t)$

$$c_{\min} \leq c(t) \leq c_{\max} \quad (25)$$

In this model, any pair $(T(t), P(t))$, $(T(t), c(t))$, $(c(t), P(t))$ are the control variables. The power P can be obtained by an internal source or by external source such as a solar electric system and will depend in part on the radial distance from the Sun. Optimal control theory has been used to determine the values of the controls that affect directly the thrust acceleration magnitude, namely T , c , P , for this system[9].

3.2. Detailed Engine Model

For the detailed engine model the controls can be Thrust (T), specific impulse (I_{sp}), exhaust velocity (c), Power (P), mass flow rate ($\dot{m}$), Thrust acceleration (T_{acc}), or Thrust over weight (T/W). In addition, the efficiency (η) of the engine can be modeled which relates the input power (P_{input}) to the power available to the engine. Any valid combination of the engine controls can be selected (e.g., Thrust and I_{sp} , Power and c , etc.) Any of the control parameters can be specified as constants, or can be modeled using selected engine control laws. The selected engine controls must satisfy the equations:

$$\eta = \frac{P}{P_{input}} \quad (26)$$

$$P = \frac{1}{2} \cdot T \cdot c \quad (27)$$

$$\dot{m} = -T/c \quad (28)$$

$$c = I_{sp} \cdot g_0 \quad (29)$$

$$T_{acc} = T/m \quad (30)$$

$$T/W = T/(m \cdot g_0) \quad (31)$$

The efficiency of an electric propulsion systems can be modeled as a function of specific impulse using the equation:

$$\eta = \frac{BB \times I_{sp}^2}{I_{sp}^2 + DD^2} \quad (32)$$

where BB and DD are specified coefficients. Alternately, polynomial expressions for mass flow rate and thrust as a function of power can be used:

$$\dot{m} = c_f \cdot [c_m(1) + c_m(2) \cdot P + c_m(2) \cdot P^2 + c_m(4) \cdot P^3 + c_m(5) \cdot P^4] \quad (33)$$

$$T = c_f \cdot [c_t(1) + c_t(2) \cdot P + c_t(2) \cdot P^2 + c_t(4) \cdot P^3 + c_t(5) \cdot P^4] \quad (34)$$

Where c_f and c_t are coefficient arrays for modeling LILT effects and solar panel degradation, respectively; c_f is the duty cycle fraction.

The engine can use an internal power source, or be modeled as a solar electric engine. For a solar electric engine, a custom solar panel power model can be used:

$$P = \phi \cdot P_0 \cdot \left[\frac{g(1) + g(2)/r + g(3)/r^2}{1 + g(4) \cdot r + g(5) \cdot r^2} \right] \cdot \frac{1}{r^2} \cdot [t(1) + t(2) \cdot e^{t(3) \cdot \Delta t} + t(4) \cdot \Delta t] \quad (35)$$

Where g and t are coefficient arrays, Δt is the accumulated time the vehicle has been in space, P_0 is the reference power at 1 AU at the start of the mission, r is the distance from the Sun, and ϕ is the fraction of the Sun that is visible. A variety of eclipse models can be used, from a simple cylindrical model to a complex model that includes the umbra, penumbra, and antumbra. The accumulated time in space can be tracked across multiple segments, allowing for multiple coast and thrust arcs in the mission.

The remaining discussion is limited to discussing the computation of the thrust direction unit vector $\mathbf{u}(t)$.

3.3. Parameter Steering Model

The thrust direction unit vector is parameterized by two spherical angles referred to the *finite burn control frame*. This frame can be the same as the segment propagation frame, but it can also be an osculating frame such as one that uses the instantaneous velocity vector as a basis direction. In terms of the spherical angles,

$$\mathbf{u}(t) = \begin{pmatrix} \cos \alpha(t) \cos \beta(t) \\ \sin \alpha(t) \cos \beta(t) \\ \sin \beta(t) \end{pmatrix} \quad (36)$$

The angles are time functions of the form

$$\alpha(t) = \alpha_0 + \dot{\alpha}_0(t - t_0) + \frac{\ddot{\alpha}_0(t - t_0)^2}{2} + a_\alpha \sin(\omega_\alpha(t - t_0) + \phi_\alpha) \quad (37)$$

$$\beta(t) = \beta_0 + \dot{\beta}_0(t - t_0) + \frac{\ddot{\beta}_0(t - t_0)^2}{2} + a_\beta \sin(\omega_\beta(t - t_0) + \phi_\beta) \quad (38)$$

where $(t_0 \leq t \leq t_f)$. Each of the constants in Eqs. 37-38 can be part of $\mathbf{s}_x^i$. As defined, the functions for the

spherical angles α and β admit constant, linear, quadratic, and sinusoidal terms. For most practical applications, the constant and linear terms are sufficient. For longer finite burn durations, both the quadratic and the sinusoidal terms may be needed.

A more general model that attempts to emulate the potential complexity that the steering of a finite burn can exhibit is referred to as the Single Axis Rotation (SAR) Model. In this model a reference unit vector $\mathbf{u}_{ref}$ is rotated about a rotation axis $\mathbf{n}$ through a time varying angle γ . The reference unit vector $\mathbf{u}_{ref}$ is referenced to a control frame that in turn is defined by basis vectors defined in a frame centered at a given celestial body c_b . The spherical angles that define the rotation vector $\mathbf{n}$ and the rotation angle γ are functions of time similar to the time functions given by Eqs. 37-38. A special case exists if $\mathbf{u}(t)$ is confined to a plane normal to the rotation axis $\mathbf{n}$; here the new reference unit vector $\mathbf{u}'_{ref}$ that gets rotated is

$$\mathbf{u}'_{ref} = \frac{(\mathbf{n} \times \mathbf{u}_{ref}) \times \mathbf{n}}{|(\mathbf{n} \times \mathbf{u}_{ref}) \times \mathbf{n}|}$$

Depending on the parameters used to define $\mathbf{u}_{ref}$, $\mathbf{n}$ and γ , $\mathbf{u}(t)$ can be constant, or it can rotate at a constant rate in plane normal to $\mathbf{n}$, or it can precess and nutate relative to a cone whose central axis is the rotation axis $\mathbf{n}$. In general, the unit vector thrust direction is calculated:

$$\begin{aligned} \mathbf{u}(t) = & \mathbf{u}_{ref}(t)^\top \mathbf{n}(t) (1 - \cos \gamma(t)) \mathbf{n}(t) \\ & + \cos \gamma(t) \mathbf{u}_{ref}(t) + \sin \gamma(t) (\mathbf{n}(t) \times \mathbf{u}_{ref}(t)) \end{aligned}$$

3.4. Optimal Control Steering Model

In the Mayer form of the optimal control problem[1, 4] where the cost function to maximize is of the form

$$J = \phi(t_0, \mathbf{x}_0, t_f, \mathbf{x}_f) \quad (39)$$

the optimal control Hamiltonian is

$$H = \boldsymbol{\lambda}^\top \mathbf{f} \quad (40)$$

where $\boldsymbol{\lambda}$ is the costate vector of Lagrange multipliers adjoined to the state vector and $\mathbf{f}$ are the state equations,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t, \mathbf{u}_c) \quad (41)$$

and $\mathbf{u}_c$ is a vector of control variables. A first-order necessary condition is the vector differential equation for $\boldsymbol{\lambda}$

$$\dot{\boldsymbol{\lambda}} = - \left(\frac{\partial H}{\partial \mathbf{x}} \right)^\top \quad (42)$$

Additionally, the control vector $\mathbf{u}_c$ is chosen to extremize H at all points on a solution consistent with the control constraints if they are present. The explicit costate vector is

$$\boldsymbol{\lambda} = (\boldsymbol{\lambda}_r^\top \quad \boldsymbol{\lambda}_v^\top \quad \lambda_m)^\top. \quad (43)$$

For the force model considered,

$$\mathbf{f}(\mathbf{x}, t, \mathbf{u}_c) = \begin{pmatrix} \mathbf{g}(\mathbf{r}, \mathbf{v}, m, t, \mathbf{p}) + \frac{T(t)}{m} \mathbf{u}(t) \\ -\frac{T(t)}{c(t)} \end{pmatrix} \quad (44)$$

the Hamiltonian is

$$H = \boldsymbol{\lambda}_r^\top \mathbf{v} + \boldsymbol{\lambda}_v^\top \left(\mathbf{g}(\mathbf{r}, \mathbf{v}, m, t, \mathbf{p}) + \frac{T}{m} \mathbf{u} \right) + \lambda_m \left(-\frac{T}{c} \right) \quad (45)$$

and the costate vector equations are

$$\dot{\boldsymbol{\lambda}}_r = - \left(\frac{\partial H}{\partial \mathbf{r}} \right)^\top = - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{r}} \right)^\top \boldsymbol{\lambda}_v \quad (46)$$

$$\dot{\boldsymbol{\lambda}}_v = - \left(\frac{\partial H}{\partial \mathbf{v}} \right)^\top = - \boldsymbol{\lambda}_r - \left(\frac{\partial \mathbf{g}}{\partial \mathbf{v}} \right)^\top \boldsymbol{\lambda}_v \quad (47)$$

$$\dot{\lambda}_m = - \frac{\partial H}{\partial m} = - \boldsymbol{\lambda}_v^\top \left(\frac{\partial \mathbf{g}}{\partial m} - \frac{T}{m^2} \mathbf{u} \right) \quad (48)$$

Extremization of H with respect to the thrust unit vector results in the well known thrust unit vector steering law[5, 6].

$$\mathbf{u}(t) = \frac{\boldsymbol{\lambda}_v(t)}{\lambda_v(t)} \quad (49)$$

and is referred to as the Primer Vector.

The solution requires knowing the costate vector $\boldsymbol{\lambda}(t_0)$; and it forms part of $\mathbf{s}_x^i$. An estimate for $\boldsymbol{\lambda}(t_0)$ is obtained by using an adjoint-control transformation[2]. If an estimate for $\mathbf{u}_0$ is available,

$$\boldsymbol{\lambda}_{v_0} = \lambda_{v_0} \mathbf{u}_0 \quad (50)$$

where $\mathbf{u}_0$

$$\mathbf{u}_0 = \begin{pmatrix} \cos \alpha_0 \cos \beta_0 \\ \sin \alpha_0 \cos \beta_0 \\ \sin \beta_0 \end{pmatrix} \quad (51)$$

The time rate of change of $\boldsymbol{\lambda}_{v_0}$ is

$$\dot{\boldsymbol{\lambda}}_{v_0} = \dot{\lambda}_{v_0} \mathbf{u}_0 + \lambda_{v_0} \dot{\mathbf{u}}_0 \quad (52)$$

where

$$\dot{\mathbf{u}}_0 = \begin{pmatrix} -\dot{\alpha}_0 \sin \alpha_0 \cos \beta_0 - \dot{\beta}_0 \cos \alpha_0 \sin \beta_0 \\ \dot{\alpha}_0 \cos \alpha_0 \cos \beta_0 - \dot{\beta}_0 \sin \alpha_0 \sin \beta_0 \\ \dot{\beta}_0 \cos \beta_0 \end{pmatrix} \quad (53)$$

Using Eq. 47

$$\boldsymbol{\lambda}_{r_0} = - \left[\dot{\lambda}_{v_0} + \lambda_{v_0} \left(\frac{\partial \mathbf{g}}{\partial \mathbf{v}} \right)_{t_0}^\top \right] \mathbf{u}_0 - \lambda_{v_0} \dot{\mathbf{u}}_0. \quad (54)$$

The basic transformation is summarized as

$$\begin{pmatrix} \lambda_{v_0} \\ \dot{\lambda}_{v_0} \\ \alpha_0, \beta_0 \\ \dot{\alpha}_0, \dot{\beta}_0 \\ \lambda_{m_0} \end{pmatrix} \Rightarrow \begin{pmatrix} \lambda_{v_0} \\ \dot{\lambda}_{v_0} \\ \mathbf{u}_0 \\ \dot{\mathbf{u}}_0 \\ \lambda_{m_0} \end{pmatrix} \Rightarrow \begin{pmatrix} \boldsymbol{\lambda}_{v_0} \\ \boldsymbol{\lambda}_{r_0} \\ \lambda_{m_0} \end{pmatrix}. \quad (55)$$

The necessity of explicitly specifying all of the components of the costate vector has been removed in favor of specifying the four control related variables $(\alpha, \beta, \dot{\alpha}, \dot{\beta})$ and three costate related variables $(\lambda_v, \dot{\lambda}_v, \lambda_m)$. Without any prior knowledge of the solution for $\mathbf{u}(t)$, obtaining an estimate for these remaining quantities is possible, but is problem dependent and requires further knowledge associated with the additional transversality conditions associated with the optimal control problem[9]. If a parameter model solution exists, a method to obtain these for a constant exhaust velocity engine is discussed in the last section of this paper. The use of the transformation replaces the explicit costate vector in $\mathbf{s}_x^i$ with the adjoint control variables.

It is important to note that even though an optimal control formulation leads to a well defined multi-point boundary value problem with functions that include transversality conditions on the states and costates at distinct times, these conditions are not used in the system. This is intentional. Instead, the method used is best described as a hybrid method where some of the unknown variables, which include the initial values of the costates or functions of them, form part of the variable vector and the cost function is optimized directly with an optimization algorithm. The advantage being that the control follows the control optimality condition, without explicitly deriving and enforcing the transversality conditions.

4. TARGETING AND OPTIMIZATION

Let $\mathbf{x}_p$ be the $n_p \times 1$ vector that contains the set of variables to be determined. $\mathbf{x}_p$ is an augmented vector that contains all of the segment variable vectors $\mathbf{s}_x^i (i = 1..n)$. The function vector is the $n_c \times 1$ function vector $\mathbf{c}$ that contains all of the segment function vectors $\mathbf{s}_f^i (i = 1..n)$. $\mathbf{c}$ is functionally dependent on $\mathbf{x}_p$,

$$\mathbf{c} = \mathbf{c}(\mathbf{x}_p) \quad (56)$$

The constraints on $\mathbf{c}$ are

$$\mathbf{c}^l \leq \mathbf{c}(\mathbf{x}_p) \leq \mathbf{c}^u \quad (57)$$

where $\mathbf{c}^l$ and $\mathbf{c}^u$ are constant vectors of lower and upper bounds, respectively. An equality constraint on an element j of $\mathbf{c}$ is defined by setting

$$\mathbf{c}_j^l = \mathbf{c}_j^u = \mathbf{c}_j^* \quad (58)$$

where $\mathbf{c}_j^*$ is the required target value. Simple one sided inequality constraints $\mathbf{c}_j$, are set by choosing

$$-\infty \leq \mathbf{c}_j \leq \mathbf{c}_j^u \text{ or } \mathbf{c}_j^l \leq \mathbf{c}_j \leq +\infty \quad (59)$$

If $n_p = n_c$ and all of the constraints on $\mathbf{c}$ are equality constraints, then the problem requires solving a nonlinear system of equations. If $n_p < n_c$ and all of the constraints on $\mathbf{c}$ are equality constraints, then a minimax solution is

needed where it is necessary to find the local minimum of the function

$$\max \left| [\mathbf{c}(\mathbf{x}_p)]_j \right|, j = 1..n_c. \quad (60)$$

If $n_p > n_c$, the system is underdetermined, the minimum of the function in Eq. 60 zero, and a solution is the minimum norm solution from the initial value of $\mathbf{x}_p$. For an optimization problem, let the element k of $\mathbf{c}$, c_k , be the objective function with the condition that $c_k^l \neq c_k^u$. The nonlinear constrained optimization problem is to minimize or maximize c_k subject to the constraints given by Eq. 57. The problem has been cast in a standard form currently used by state of the art nonlinear constrained optimization codes such as SNOPT[7, 15].

Regardless of whether a targeting or optimization problem is required to be solved, it is desirable to at least estimate as accurately as possible the $n_p \times n_c$ Jacobian matrix $\partial \mathbf{c} / \partial \mathbf{x}_p$. Numerical finite difference methods are the most common methods for this purpose, even though they have inherit difficulties especially for the functions in $\mathbf{c}$ that are highly sensitive to the elements in $\mathbf{x}_p$. For example, the (i, j) element of $\partial \mathbf{c} / \partial \mathbf{x}_p$ can be approximated using a central difference approximation

$$\frac{\partial c_i}{\partial x_{p_j}} \approx \frac{c_i(x_{p_j} + \Delta x_{p_j}) - c_i(x_{p_j} - \Delta x_{p_j})}{2\Delta x_{p_j}} \quad (61)$$

where Δx_{p_j} is the positive perturbation stepsize for the x_{p_j} element of $\mathbf{x}_p$. A very accurate method to compute $\partial \mathbf{c} / \partial \mathbf{x}_p$ uses state transition matrices and analytical gradient expressions for all of the functions and the states required to compute those functions[18, 11, 12]; however, this method is not practical for a generalized system because of the overhead required to derive all of the required relationships between the large number of search variables and constraint functions.

Examples that describe how to cast mission design and trajectory optimization problems into a targeting or optimization solution method for both impulsive and finite burn maneuvers are described in[13].

5. FUTURE CONSIDERATIONS

One of the areas currently being examined is an alternate formulation for the finite burn thrust steering model. For the parameter basic steering model, assuming only at most a quadratic function for the spherical angles used to represent $\mathbf{u}(t)$, an augmented state vector that includes the parameters that define the thrust unit vector $\mathbf{u}(t)$ and its rate $\dot{\mathbf{u}}(t)$ is being considered,

$$\mathbf{x} = \left(\mathbf{r}^\top \quad \mathbf{v}^\top \quad m \quad \alpha \quad \beta \quad \dot{\alpha} \quad \dot{\beta} \right)^\top \quad (62)$$

with

$$\dot{\mathbf{x}}(t, \mathbf{x}) = \begin{pmatrix} \mathbf{v} \\ \mathbf{g}(\mathbf{r}, \mathbf{v}, m, t, \mathbf{p}) + \frac{T(t)}{m} \mathbf{u}(t) \\ -\frac{T(t)}{c} \\ \dot{\alpha} \\ \dot{\beta} \\ \ddot{\alpha} \\ \ddot{\beta} \end{pmatrix}^i \quad (63)$$

In this model, the initial conditions for the spherical angles and their rates form part of the segment variable vector. With this model, a procedure is being investigated to automate the conversion of a parameter model finite burn solution to a finite burn solution that uses the primer vector for the thrust unit vector. As an example, assuming that $\mathbf{g}$ only depends on $\mathbf{r}$ and t , it can be shown that an augmented state vector of the form

$$\mathbf{x} = (\mathbf{r}^\top \mathbf{v}^\top m \alpha \beta \dot{\alpha} \dot{\beta} \lambda_v \dot{\lambda}_v \lambda_m)^\top \quad (64)$$

has the following first order equation of motion

$$\dot{\mathbf{x}}(t, \mathbf{x}) = \begin{pmatrix} \mathbf{v} \\ \mathbf{g}(\mathbf{r}, \mathbf{v}, m, t, \mathbf{p}) + \frac{T}{m} \mathbf{u} \\ -\frac{T}{c} \\ \dot{\alpha} \\ \dot{\beta} \\ \ddot{\alpha} \\ \ddot{\beta} \\ \dot{\lambda}_v \\ \dot{\lambda}_v \\ \frac{T}{m^2} \lambda_v \end{pmatrix} \quad (65)$$

where $\dot{\alpha}, \dot{\beta}$ are determined uniquely from $\mathbf{u}$ and $\dot{\mathbf{u}}$; and $\ddot{\alpha}, \ddot{\beta}$ are determined uniquely from $\mathbf{u}, \dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$; and

$$\ddot{\mathbf{u}} = \mathbf{G}\mathbf{u} - \{[(\mathbf{u}^\top \mathbf{G})(\mathbf{u})] + \dot{\mathbf{u}}^\top \dot{\mathbf{u}}\} \mathbf{u} - 2 \frac{\dot{\lambda}_v}{\lambda_v} \dot{\mathbf{u}} \quad (66)$$

$$\ddot{\lambda}_v = \lambda_v [(\mathbf{u}^\top \mathbf{G})(\mathbf{u}) + \dot{\mathbf{u}}^\top \dot{\mathbf{u}}] \quad (67)$$

where

$$\mathbf{G} = \frac{\partial \mathbf{g}(\mathbf{r}, t)}{\partial \mathbf{r}} \quad (68)$$

With this formulation, an automated procedure is being developed that transforms a parameter model finite burn solution to an approximation of the nearby optimal control finite burn solution. For a constant exhaust velocity engine use is made of the associated switching function S that determines the value for T ,

$$S = \frac{\lambda_v}{m} - \frac{\lambda_m}{c} \quad (69)$$

The cost function is minimization of the propellant consumed by the finite burn. With this, $S(t_0) = 0$ and t_0 is forced to be the switching time from zero thrust to $T_{\max}$. The scale for the costates is set by choosing

$$\lambda_v(t_0) = 1 \quad (70)$$

Using $S(t_0) = 0$,

$$\lambda_m(t_0) = c \frac{\lambda_v(t_0)}{m(t_0)} \quad (71)$$

The time history of $\lambda_v(t)$ is assumed to be quadratic with negative curvature ($\ddot{\lambda}_v < 0$) and with a maximum value near the middle of the burn. If t_m is the time of the midpoint of the burn then as an approximation set $\dot{\lambda}_v(t_m) = 0$. Having a parameter model finite burn solution, Eq. 67 is used at t_m to get an approximation for $\ddot{\lambda}_v$. $\ddot{\lambda}_v$ is assumed to be constant; this along with the conditions $\lambda_v(t_0) = 1, \dot{\lambda}_v(t_m) = 0$ the expression for $\dot{\lambda}_v(t_0)$ is

$$\dot{\lambda}_v(t_0) = -\lambda_v(t_m) [(\mathbf{u}^\top \mathbf{G})(\mathbf{u}) + \dot{\mathbf{u}}^\top \dot{\mathbf{u}}]_{t_m} (t_m - t_0) \quad (72)$$

where $\lambda_v(t_m)$ is the value of λ_v at t_m and is given by,

$$\lambda_v(t_m) = \frac{1}{1 + \frac{1}{2} [(\mathbf{u}^\top \mathbf{G})(\mathbf{u}) + \dot{\mathbf{u}}^\top \dot{\mathbf{u}}]_{t_m} (t_m - t_0)^2} \quad (73)$$

This procedure provides the initial estimate for all of the adjoint control variables needed in the augmented state vector, Eq. 64, at t_0 .

Another area being considered for possible inclusion in Copernicus is a general architecture for guidance algorithms and control laws. Current finite burn models in Copernicus can be complemented with guidance algorithms for specific applications. Finding a general method for the implementation of such algorithms is still under research. The implementation of a general relative motion capability between segments is also under consideration. This will allow a more flexible modeling and optimization of spacecraft rendezvous and formation flying problems. Finally, a more complete vehicle model within the segment that incorporates the ability to model a spacecraft as a system of multiple masses, e.g.: tanks, consumables, dry mass, stages, propellant, etc. is being designed.

REFERENCES

- [1] Bryson, A. E. Jr., and Ho, Y. C. (1975) *Applied Optimal Control*, Hemisphere Publishing Corporation, Revised Printing, New York, NY.
- [2] Dixon, L. C., and Bartholomew-Biggs, M. C. (1981) Adjoint Control Transformations for Solving Practical Optimal Control Problems, *Optimal Control Applications and Methods*, 2, 365–381.
- [3] Garn M., Qu M., Chrono J., Su P., Karlgaard C. NASA's Planned Return to the Moon: Global Access and Anytime Return Requirement Implications on the Lunar Orbit Insertion Burns, *Proceedings of the AIAA Guidance, Navigation and Control conference*, August, 2008.
- [4] Hull, D. G. (2003) *Optimal Control Theory for Applications*, Springer-Verlag, New York, NY.

- [5] Lawden, D. E. (1963) *Optimal Trajectories for Space Navigation*, Butterworths, London.
- [6] Marec, J. P. (1979) *Optimal Space Trajectories*, Elsevier Scientific Publishing Company, New York, NY.
- [7] Gill, P. E. et. al., (1998) *Practical Optimization*, Academic Press, San Diego, CA.
- [8] Ocampo, C. (2003) An Architecture for a Generalized Trajectory Design and Optimization System, in *Libration Point Orbits and Applications*, Edited by Gomez, G., Lo, M., and Masdemont, J. World Scientific Publishing Co., Singapore.
- [9] Ocampo, C. (2004) Finite Burn Maneuver Modeling for a Generalized Spacecraft Trajectory Design and Optimization System, *Astrodynamics, Space Missions, and Chaos, Annals of the New York Academy of Sciences*, **1017**, 210–233.
- [10] Ocampo C “Elements of a Software System for Spacecraft Trajectory Optimization,” *Spacecraft Trajectory Optimization*, Ed. Bruce Conway, Cambridge University Press 2010
- [11] Ocampo C. and Munoz JP. Variational Equations for a Generalized Trajectory Model, AAS-09-255, *Proceedings of the 19th Space Flight Mechanics Meeting*, Savannah, GA, 2009.
- [12] Ocampo C. and Munoz JP. Variational Model for the Optimization of Constrained Finite-Burn Escape Sequences AAS 09-381, *2009 AAS/AIAA Astrodynamics Specialist Conference Proceedings*, Pittsburgh, PA, 2009.
- [13] Ocampo, C. and Byrnes, D. “Mission Design and Trajectory Optimization,” chapter to appear in *Encyclopedia of Aerospace Engineering*, Wiley Publishing, Dec 2010
- [14] Senent J. An Optimal Initial Guess Generator for Entry Interface Targeters AAS 09-426, *2009 AAS/AIAA Astrodynamics Specialist Conference Proceedings*, Pittsburgh, PA, 2009.
- [15] Stanford Business Software http://www.sbsi-sol-optimize.com/asp/sol_product.snopt.html
- [16] Williams J., Davis E., Lee D., Condon G., Dawn T., Qu M. Global Performance Characterization of the three burn trans-Earth injection maneuver sequence over the lunar nodal cycle, *2009 AAS/AIAA Astrodynamics Specialist Conference Proceedings*, Pittsburgh, PA, 2009.
- [17] Williams J., Senent, J., Ocampo C., Mathur R., Davis E. Overview and Software Architecture of the Copernicus Trajectory Design and Optimization System, in *4th International Conference on Astrodynamics Tools and Techniques*, Madrid, May, 2010
- [18] Zimmer, S., and Ocampo, C. (2005) Use of Analytical Gradients to Calculate Optimal Gravity-Assist Trajectories, *Journal of Guidance, Control, and Dynamics*, **28**, No. 2, 324–332.

A HISTORY OF COPERNICUS: THE ORIGIN, DEVELOPMENT, AND EVOLUTION OF JSC'S SPACECRAFT TRAJECTORY DESIGN AND OPTIMIZATION SYSTEM

Jacob Williams ^{*}, Juan S. Senent [†], Ravishankar Mathur [‡], Shaun M. Stewart [§]

This paper describes the history of the Copernicus spacecraft trajectory design and optimization system, a software tool originally developed at The University of Texas at Austin by Dr. Cesar Ocampo, and then later at the NASA Johnson Space Center. For twenty years, Copernicus has been a workhorse tool for advanced mission design at JSC. During this time, the tool has been under constant development and has evolved significantly. The origin, development, and use of Copernicus is detailed, and some lessons learned and future possibilities are also presented.

INTRODUCTION

Advanced spacecraft trajectory design and optimization tools are a critical component for enabling human and science missions at NASA. In an earlier paper,¹ we outlined the history of on-orbit trajectory design for the Orion spacecraft at the NASA Johnson Space Center (JSC), culminating in the Artemis I mission flown in 2022. Copernicus,²⁻⁶ the software tool used to design that trajectory, is a comprehensive and generalized spacecraft trajectory design and optimization system. Developed at The University of Texas at Austin (UT) and JSC, it forms part of the suite of tools used by NASA, industry, and academia to study, design, and execute spacecraft missions.

The Copernicus software application and environment includes a GUI with interactive 3D visualization, as well as scripting, API, and parallelization components. It facilitates the design and optimization of spacecraft trajectory problems using complex force fields, multiple types of propulsion systems (impulsive → finite burn → low to high thrust), multiple spacecraft, contingency scenarios and multiple reference frames. It is applicable to celestial body centered (planets, moon, asteroids, comets), Earth-Moon, Earth-Moon-Sun (libration points and orbits), interplanetary (asteroids, comets, cyclers), and satellite tours. Copernicus, which was the winner of the 2021 NASA Software of the Year (SOTY) award, is intended to be a tool that adapts and evolves to state-of-the-art approaches and requirements in the field of spacecraft trajectory simulation, design, optimization, and operation. The earlier paper touched on the development of Copernicus during this period, and the current paper provides more detail, telling the story of the history and development of the software from its inception. The unique architecture of Copernicus was the brainchild of its original creator, Dr. Cesar Ocampo (May 1967 – September 2024) and this paper also serves as a tribute to him and the tool he created which has had a profound influence on spacecraft trajectory design and optimization.

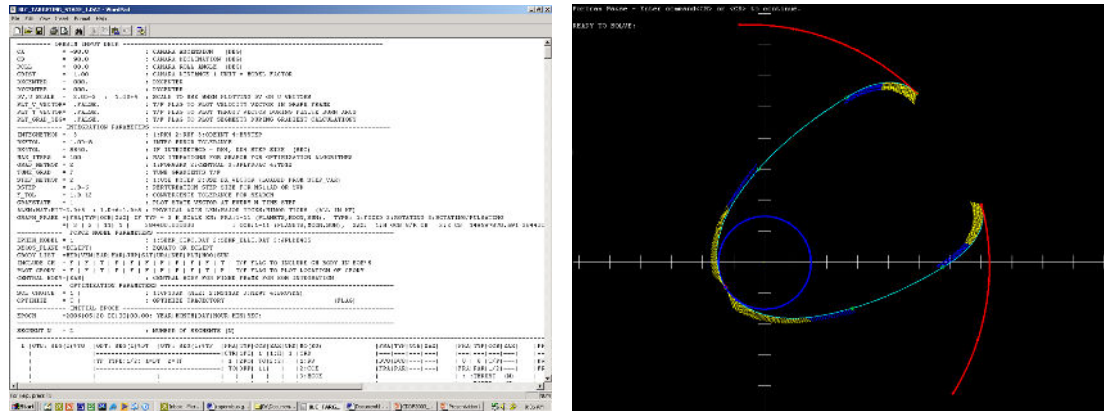
^{*}EG/Aeroscience and Flight Mechanics, NASA Johnson Space Center, Houston, TX 77058

[†]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

[‡]Intuitive Machines, Houston, TX, 77058

[§]Intuitive Machines, Houston, TX, 77058

ORIGINS OF COPERNICUS (2000 – 2002)



(a) Original Input File Format of Copernicus. The user would edit this file manually and then execute the program, which would then open another window and display the iterations. (b) Example Trajectory from the Original Copernicus Prototype. The graphics were based on code Cesar had written as an undergraduate student.⁷

Figure 1: Original Copernicus Prototype (pre-GUI, circa 2002). This was a command-line tool with a 2D graphics display of the trajectory. Cesar Ocampo used these early versions of Copernicus in his courses at UT, as well as demonstrations to NASA.

Copernicus* and its underlying concept was invented and developed as a prototype by Professor Cesar Ocampo at The University of Texas at Austin beginning in August 2001. Ocampo joined the faculty there in Fall 2000 as an assistant professor in the Department of Aerospace Engineering & Engineering Mechanics. Up to that time he had written many programs to solve specific problems: designing a minimum energy Earth trailing solar orbit for the Spitzer telescope (named SIRTf at the time),⁸ codes for computing how to use the Moon's gravity to extend the lifetime of a stranded Geosynchronous Orbit (GEO) satellite via a plane change from a high-inclination Geosynchronous Transfer Orbit (GTO),⁹ tools to design low-thrust transfers to an Earth-Sun Distant Retrograde Orbit (DRO)[†],¹⁰ a commercial tool called Xenon Ion Propulsion System Trajectory Optimization Program (XIPSTOP) for Hughes Space and Communications Company (HSC), used to move high-powered, xenon ion propulsion system based geostationary bound communications satellites from GTO to the required geostationary slot,¹¹ and many others. Ocampo conceived of Copernicus as the sum of all his prior work: a software tool that could solve all trajectory design and optimization problems, one that would be used for decades and never be finished nor replaced when new tools become available, but rather evolve by incorporating new algorithms and approaches over time.

Ocampo developed the initial working prototype of Copernicus single-handedly between late-2001 and early-2002 (see Figure 1). This prototype, written in Fortran 77/90[‡] using Compaq Visual Fortran (CVF) on Windows, was used to demonstrate the functionality and benefit to potential spon-

*Although sometimes written as “COPERNICUS” in the early days, it is not an acronym.

†The DRO acronym was coined by Ocampo.

‡Fortran was chosen for its mature numerical performance, portability, and deep heritage in the scientific computing community – especially for astrodynamics applications. This choice allowed Copernicus to leverage existing libraries and legacy trajectory tools while delivering high performance across platforms.

sors and also as a teaching tool in courses taught at the University of Texas.* This initial prototype could solve interesting problems quickly, but was limited to ten trajectory segments, had no GUI, used a simple text-file[†] based input (see Figure 1a), a 2D projection graphics model (see Figure 1b),⁷ and included basic numerical integrators and root solvers based on Numerical Recipes.¹² A key feature of Copernicus was the use of the VF13AD optimizer from the Harwell Subroutine Library.[‡] Ocampo presented this prototype to NASA JSC, and it was immediately recognized by Jerry Condon (a trajectory and mission designer there) as something innovative which had great potential to replace many of the legacy single-purpose tools that were used at that time at the center.

UT AND JSC COLLABORATION (2002 – 2006)

In June 2002 a grant from NASA JSC, which included JSC Center Director Discretionary Funding (CDDF) and internal Aerospace and Flight Mechanics Division funding, was used to begin production-level development. Support was also received from the NASA Goddard Space Flight Center (GSFC) and NASA’s In-Space Propulsion (ISP) Program Low-Thrust Trajectory Tools (LTTT) activity, which was a NASA-wide effort to develop a suite of low-thrust interplanetary trajectory tools.¹³ The top level goals for Copernicus, as outlined during the LTTT activity in 2004 were:

- Efficient and intuitive data input and modification.
- Interactive running, testing, editing, and result feedback.
- Real time integrated visualization.
- Automatic bookkeeping and updating of results (results re-populate the GUI fields; update necessary support files).
- All components must be modular (integrators, optimizers, etc.).
- System must be evolutionary with growth capability. *By design the system will never be finished.*
- System should be multi-platform; use libraries and toolsets that have multi-platform capability.

During this period (2002 – 2006) Ocampo lead the small development team from UT, and coordinated with Jerry Condon at JSC who would become the overall NASA lead of the project (and remain so until his retirement in 2025). The initial prototype was expanded and refactored, and development of a GUI and integrated 3D graphics was initiated. From the outset, graphical interaction with the trajectory problem was a key design requirement. The GUI was intended not just as a convenience, but as a central component for intuitive problem setup, while the integrated graphics window provided real-time visualization of the solution. This visual workflow enabled users to quickly understand the structure of the trajectory, assess convergence behavior, and iteratively refine the setup – all without relying solely on textual input or post-processing tools.

*Ocampo also created the “Mars Intercept Game”, which was a modified version of Copernicus that accepted keyboard input to change the thrust vector control of a spacecraft, with the goal of intercepting Mars. The game illustrated the basic problem of spacecraft trajectory optimization, and was used by Ocampo in his classes at UT and in other presentations to schools and students.

[†]The file (which used the .DAT extension) initially followed a fixed format, with each row and column serving a specific purpose. Later, it adopted a namelist format and became known as the “input deck” – a term rooted in the historical use of punched card decks for data input.

[‡]HSL Archive – <https://www.hsl.rl.ac.uk/archive/>

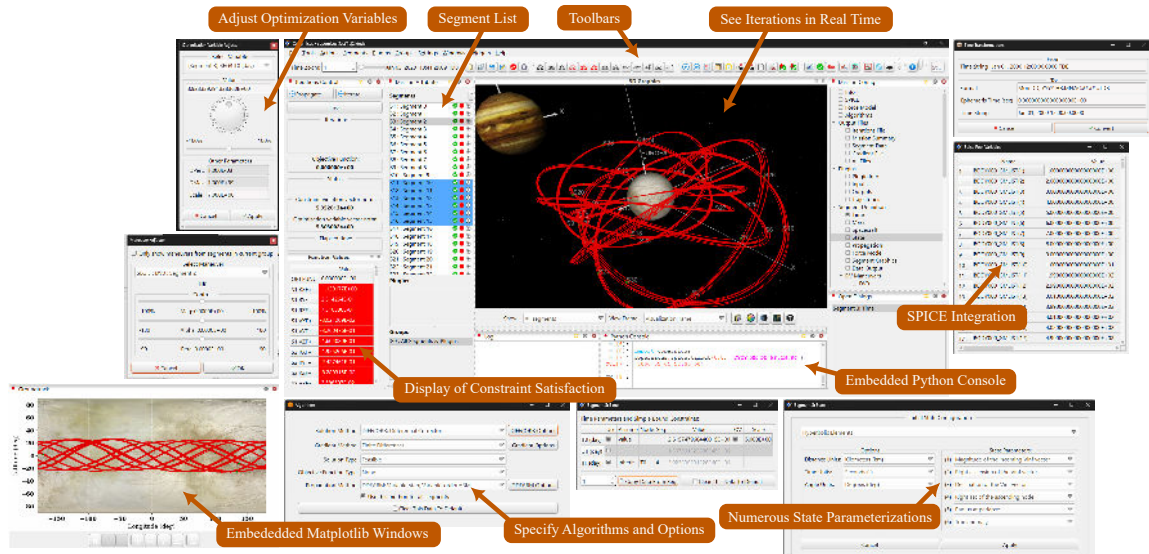


Figure 2: Copernicus GUI Dialogs. The GUI is a key feature of Copernicus, allowing the user to set up the trajectory design problem, adjust the settings, and receive text and graphical feedback. It has evolved significantly over the years, but remains a core and distinctive component of the solution process.

Having a user-friendly GUI (see Figure 2) was one of Ocampo’s top priorities, and indeed was one of the things that separated Copernicus from previous codes he had written and other NASA/JSC trajectory tools at the time. The GUI was always envisioned as the main interface to Copernicus, and a lot of effort was devoted to its development and integration as the original prototype was evolved into the first production release. Ocampo selected the Winteracter GUI toolkit* to build the Copernicus GUI for three main reasons: 1) it was a full-featured GUI toolkit with 3D graphics capabilities available for Fortran-based applications, 2) it supported Windows, Linux, and macOS, and 3) the built-in grid widget, which fit nicely with the concept that each variable would have many attributes that the user could edit (e.g., optimization variable settings such as scale factors and bounds). See Figure 6 for a comparison of various trajectory tool GUIs. The three phases[†] in the development of the original Copernicus GUI architecture at UT are shown in Table 1. The first version that included a working GUI was completed in August 2004 (see Figure 3).

The original prototype included trajectory graphics (See Figure 1b) in a separate stand-alone window that would launch when the program executed. While often referred to as “2D graphics” (it was called that up to the 4.6 release when it was removed), the original prototype rendered 3D trajectories using a basic hidden-line removal algorithm. The graphics code was partially based on code Ocampo had written as an undergraduate student,⁷ converted to use the CVF native graphics API. Although the graphics were limited, they did display optimization iterations in real time as

*Winteracter was a graphics and GUI toolset for Fortran 90 or later. A commercial product, it was sold from 1997–2024, when it was coincidentally discontinued immediately after Cesar’s passing in September 2024. GINOMENU, another commercial product for Fortran GUIs, was also considered at the time but not ultimately selected.

[†]The 4th phase of the Copernicus architecture would be the 5.0 refactor at NASA,⁶ where Copernicus was rearchitected into a separate core library that is called from a Python/Qt GUI (rewritten from scratch, mirroring in many ways the original Winteracter GUI, but much improved).

Table 1: The three phases (from 2001 – 2006) in the development of the original Copernicus architecture

Phase 1	No GUI. Ocampo’s original version written at UT. This was a command line executable that read an input file. There was no GUI, although it did use the CVP graphics routines which opened in a separate window to show the optimization problem iterations. This version was entirely written by Ocampo (see Figure 1).
Phase 2	Prototype GUI. An initial prototype GUI that was a separate program that existed only to create the input file that the command line Copernicus would read. It would launch the core executable to iterate.
Phase 3	Unified GUI. Only one program that contained the GUI and core code. The core code was called directly from the GUI. This version had integrated graphics (originally just 2D graphics, which were converted to Winteracter graphics calls) and ultimately OpenGL 3D graphics. ¹⁴ This was the state of the program in the 1.0 release in 2006, which was delivered to NASA, and it remained this way for many years afterwards.

they occurred – a feature uncommon in many tools at the time. Ocampo’s vision was to provide immediate visual feedback throughout the process: from the initial guess, to each optimization step, to the final converged solution. This allowed users to quickly assess whether the solution was converging in the right direction or becoming stuck, without relying on external post-processing tools. Users could interrupt the process at any point, adjust settings such as algorithm parameters or segment configurations, and rerun the problem – enabling an interactive and iterative workflow that significantly improved usability and insight.

While Ocampo’s 2D graphics were novel, he recognized several limitations. They were not extensible to new visualization components (e.g., spacecraft, general celestial bodies, etc.), they significantly slowed down the trajectory computations, they did not provide end-user interactivity, and they did not leverage the computer’s graphics hardware. Having always been an avid champion of advanced 3D graphics, in 2003 Ocampo’s team began design, development, and integration of modern 3D graphics. The result was a fully standalone 3D graphics API, called OpenFrames*, that allowed any aerospace software to easily integrate high-performance interactive 3D visualizations via the industry-standard OpenGL and OpenSceneGraph† graphics interfaces.¹⁴ OpenFrames provided the ability to display celestial bodies, trajectories, spacecraft models, and maneuver vectors. It enabled interactive use (pan, zoom, rotate, animate) and utilized multithreading to leverage the burgeoning trend of multi-core computers to eliminate the slowdown on Copernicus’ propagators and optimizers. While Copernicus was the first software to utilize OpenFrames for its graphics, Ocampo’s eager support of its development led to it eventually being incorporated as the primary visualization tool for various commercial and NASA tools, including the General Mission Analysis Tool (GMAT) at GSFC.

At the core of Ocampo’s architecture was the *segment*² – the fundamental building block of all

*OpenFrames: Realtime interactive scientific visualization API. <https://github.com/ravidavi/OpenFrames>

†OpenSceneGraph is an open source high performance 3D graphics toolkit. <https://openscenegraph.github.io/openscenegraph.io/>

trajectories in Copernicus* (see Figure 4). A segment represents an arc connecting two endpoint nodes, t_0 and t_f , each of which may include velocity (Δv) and mass (Δm) discontinuities. The arc can be ballistic or include a finite-burn model. Segments are highly flexible: they can be connected or disconnected in time, state, or mass; inherit properties from other segments; and encapsulate optimization variables, constraints, and objective functions for use by nonlinear solvers or optimizers. They can be propagated both forward ($\Delta t > 0$) and backwards ($\Delta t < 0$), including with finite burns (where a backwards-propagated finite burn segment gains mass from $t_0 \rightarrow t_f$). Segments can model individual or multiple spacecraft and can be combined to construct custom transcriptions tailored to specific mission needs. This modular and flexible design enables Copernicus to handle a wide range of trajectory types, including impulsive, low-thrust, and high-thrust transfers, in both planetary and interplanetary, single-body or multi-body contexts. The segment framework also supports the simultaneous optimization of trajectories across multiple alternative scenarios. For example, a nominal trajectory can be optimized alongside several abort scenarios within a single unified formulation. The cost function can be customized to reflect complex mission objectives, such as minimizing the worst-case outcome across all scenarios, enabling robust and resilient trajectory design. Importantly, segments do not need to represent an actual spacecraft trajectory. They can also be used as computational constructs – for example, to evaluate a function or constraint that must be satisfied by another segment containing the physical trajectory. This abstraction adds further flexibility to the architecture, allowing complex dependencies and auxiliary calculations to be integrated seamlessly into the optimization problem.

In its original implementation by Ocampo, Copernicus used the JPL planetary ephemeris to model the positions of planets and satellites. However, it was quickly recognized that integrating the SPICE toolkit – a companion library developed by JPL[†] – would greatly enhance Copernicus’s capabilities. SPICE provided robust support for incorporating new reference frames, converting trajectory states between different frames and parameterizations, handling multiple time systems and conversions, and generating trajectory files in SPK format. These features significantly expanded the tool’s flexibility and interoperability with other mission design and analysis systems, and SPICE remains a key component of Copernicus up to the present day. Alongside these architectural improvements, Copernicus expanded its modeling capabilities to support high-fidelity, high-order gravity models, enabling accurate simulation of orbits in complex gravitational environments such as low lunar or low Earth orbit.

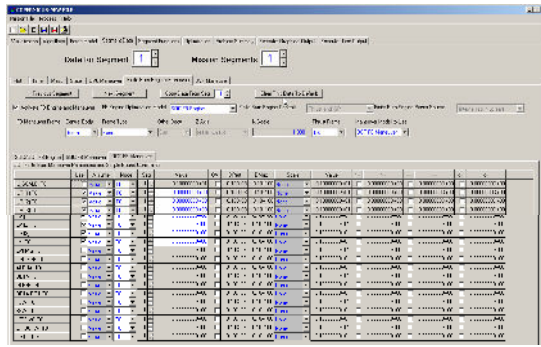
During the initial development of Copernicus, the U.S. human spaceflight program would be profoundly altered by the Space Shuttle Columbia accident in 2003 and the subsequent announcement of the Vision for Space Exploration (VSE) in 2004. Copernicus would be available at just the right time to be a major contributor to planning the next phase of human spaceflight at NASA, with its flexible architecture being exactly what was needed to design and study the required Earth-Moon trajectories for the Constellation Program and follow-on activities to return humans to the Moon and beyond.¹

Copernicus 1.0 (2006)

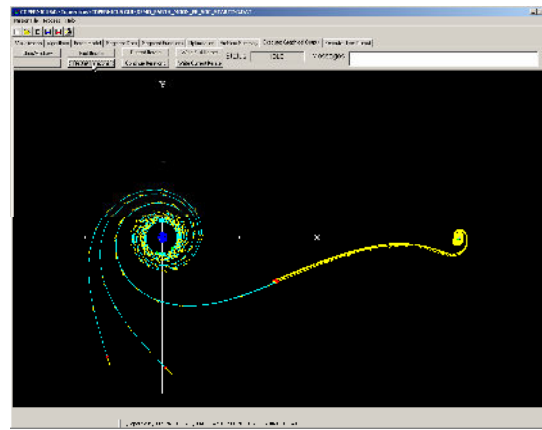
The first operational version of Copernicus was released to NASA in March 2006 (v1.0) at the conclusion of the LTTT activity (see Figure 5). This release included a full working GUI with

*In the 4.1 release in 2015, *plugins* were added as an additional basic building block. Plugins are companions of segments, and used to introduce user-defined code or other tools into the problem.¹⁵

[†]The SPICE Toolkit: <https://naif.jpl.nasa.gov/naif/toolkit.html>



(a) At this point, the GUI was taking shape, which each dialog continued within a tabbed layout. The grids are almost fully realized here, a key part of Copernicus, and the main reason for the selection of Winteracter as the GUI toolkit.



(b) The Graphics dialog was where the user would iterate the trajectory and watch the iterations as they progressed. Notably missing were the 3D graphics (Ocampo's original 2D graphics are the only available option).

Figure 3: Pre-1.0 Version of the Copernicus GUI (demoed at the 2004 Low Thrust TIM). Many of the features present in the V1.0 GUI of 2006 were already in place at this point.

integrated 3D OpenFrames-based graphics,¹⁴ finite burn and impulsive maneuvers, SPICE integration, and various propagation methods and solvers. While the tool has evolved and become much more advanced over time, all of the fundamental concepts (and some of the original code) from this original version remain in the very latest release.

JSC DEVELOPMENT (2007 – PRESENT)

Starting circa March 2007, primary development and maintenance of Copernicus was transferred to the Flight Mechanics and Trajectory Design Branch of JSC.* Copernicus has been in continuous use at JSC (and other NASA centers and commercial partners) since that time,¹ and many key capabilities in the tool as it exists today were developed during this period. Ocampo's collaboration continued, first from UT, and then later as a contractor at JSC, serving as an advisor and chief "power user". During this time, Copernicus was brought up to production level, with formal releases and documentation. As the tool began to be used more, new features were added to support current studies (some were natural extensions of the original architecture, and others were new ideas not originally envisioned). Refactoring and modernization of the codebase also continued at JSC.

A distinctive aspect of Copernicus's early development was that, for the most part, the developers were also its primary users. There was little separation between development and user teams – the tool was actively used by its creators to perform mission studies as it evolved. As a result, usability was a central focus and a major driver of development. New capabilities and features were identified organically through day-to-day use and were often implemented immediately, without a formalized development process (although this would be added later). This close integration between usage and

*By this point, the team consisted of Ocampo and two graduate students at UT, and Jerry Condon and two developers/analysts at JSC.

development fostered rapid iteration and ensured that the tool remained practical and responsive to immediate needs.

Another characteristic of Copernicus's development was its experimental and adaptive nature. Many features were added to support specific studies or one-off mission needs, but later removed when they proved too specialized or lacked broader applicability. This trial-and-error approach allowed the tool to remain focused, avoiding unnecessary complexity while still enabling rapid prototyping. It reflected the pragmatic philosophy behind Copernicus: to prioritize usability and flexibility for immediate mission needs, while evolving the tool based on actual usage rather than abstract requirements.

A major usability improvement implemented during this period was the ability to insert, delete, and move segments directly within the GUI. This was complemented by automatic detection and resolution of the correct propagation order for interconnected segments, eliminating the need for users to manually specify execution sequences. In the original prototype, segments were propagated in simple numerical order – a method that could produce incorrect results when segments inherited information (e.g., state or mass) from one another. Cesar Ocampo's original paper² emphasized the importance of propagating segments in a logically consistent order, but this feature was not implemented until after the 1.0 release. Another key enhancement was the ability to detect circular dependencies between segments – an issue that went undetected in earlier versions and could lead to invalid or misleading solutions. The eventual introduction of a dependency graph to manage segment relationships and identify such cycles significantly improved both reliability and transparency of the propagation logic. See Figure 7 for a representation of the segments and plugins in a reasonably-complex Copernicus mission.

As the overall use cases of Copernicus evolved and expanded over time, the tool was kept up-to-date, as it was being used for a wide range of analysis at JSC and other organizations. Other notable improvements and new features added to Copernicus over these years include:

- Allowing for unlimited number of trajectory segments, optimization variables, and constraints. The original prototype had various hard-coded limits to these parameters.
- GUI for working with SPICE kernels (loading and creating kernels, viewing data, etc.)
- Selectable gravity models, integrators, and force models for each segment. This was not part of the original prototype, but fit naturally into the architecture.
- The addition of other types of propagation methods for segments (e.g., Kepler, Encke, and modified equinoctial elements).
- The addition of many new state parameterizations, including a halo orbit parameterization.
- The plugin architecture,¹⁵ introduced in response to frequent user requests for greater flexibility. While the GUI made it easy to construct and solve complex trajectory optimization problems, users often needed to implement mission-specific features that were too specialized to be incorporated directly into the core Copernicus system. The plugin framework provided a solution by allowing user-defined code to be injected into the optimization process. This enabled analysts to incorporate custom models, constraints, or objective functions tailored to their unique mission scenarios, without altering the main codebase. Additionally, the plugin interface allowed Copernicus to interact with external tools, making it a more extensible and interoperable platform.
- Various widgets and tools such as the Δv to finite burn wizard, the optimization variable and maneuver adjusters (for modifying variables and seeing in real time how they affect the

trajectory), and the randomize optimization variables widget (which allows for perturbing all optimization variables by some specified percentage).*

- The addition of cross-platform functionality. Originally, Copernicus ran only on Windows. One of the earliest projects after the transition to JSC was to port the code from the then-discontinued Windows CVF compiler to the cross-platform Intel Fortran Compiler.[†] This meant that Linux and Mac releases would be possible, but it was several years before those were realized (a command-line only Linux build in 2009 and an X11-based Mac GUI prototype in 2018).[‡]
- General speed and stability improvements, and code refactoring as modern Fortran features became available in newer compilers.¹⁸
- Various visualization updates including celestial body shadowing, trajectory intermediate point markers, and stability improvements.
- More flexible data output. For example: being able to specify which variables to output (type, frame, parameterization, etc.) and the output file format (CSV, JSON, HDF5, or SPK).

A key to the success of Copernicus was regular releases through the NASA Technology Transfer Program.[§] This was a somewhat novel approach for mission design tools at JSC at the time, which historically had mostly consisted of internal tools not widely distributed, or proprietary tools built and controlled by specific contractors with no motivation to freely distribute them. Providing Copernicus freely to all NASA centers, other government entities, and government contractors, meant a growing pool of users on a variety of different projects. This not only meant less “reinventing the wheel” for trajectory design tools, but also that funds could be more readily available for support and upgrades since it was being widely used on projects that had specific needs. Some major upgrades to Copernicus would ultimately have their origin and funding source from outside of JSC.

Command Line and Scripting

As Copernicus matured and began to be used in production, it became increasingly clear that mission design workflows demanded more than interactive GUI-based problem setup. Trade studies, by their nature, require the evaluation of many configurations, scenarios, and parameter sweeps. Performing these studies manually – by editing input decks and rerunning optimizations one at a time – was impractical, particularly as the number of combinations grew exponentially with problem complexity. To address this challenge, Copernicus began evolving toward script-driven automation.

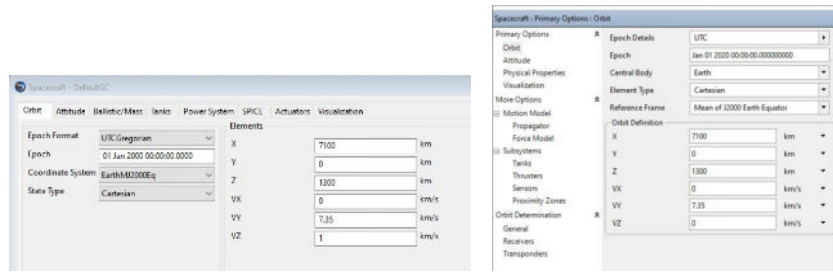
The first major step was the release of a Linux-compatible, command line only build of Copernicus, which enabled the tool to be executed on high-performance computing clusters. This version made it possible to launch large batches of runs in parallel, a critical capability for trade studies at NASA centers like JSC. Another key milestone was the redesign of the input deck, since the

*This feature has been used to develop a monotonic basin hopping wrapper to Copernicus.¹⁶

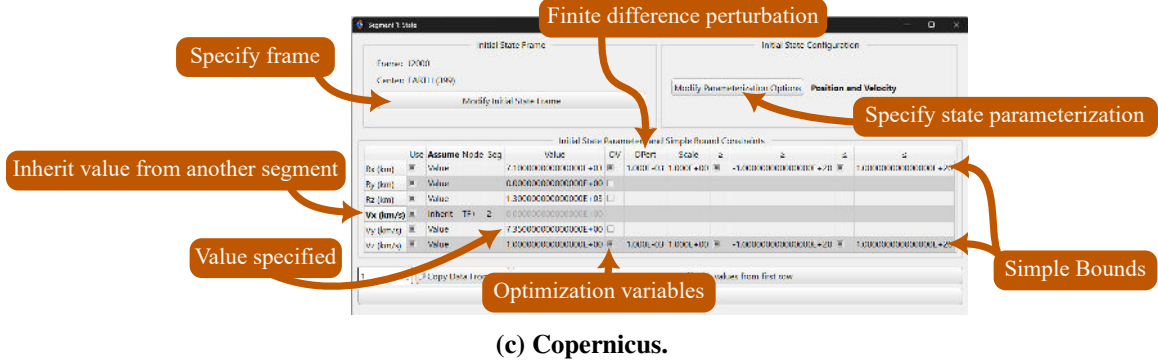
[†]CVF was the original Fortran compiler used for Copernicus. CVF was the successor to the DEC Visual Fortran compiler, after Compaq purchased DEC in 1999. In Aug. 2001, Compaq sold their Fortran business to Intel. In Dec. 2003, Intel released the successor product: Intel Visual Fortran 8.¹⁷

[‡]A fully feature-complete cross-platform Copernicus would not exist until the 5.0 release in 2020 with a rewrite of the GUI in Python.

[§]Copernicus Trajectory Design and Optimization System (Version 5.x) (MSC-26673-1). Copernicus is released under a “government use” license, and is freely available to any U.S. government entity or government contractor. Request here: <https://software.nasa.gov/software/MSC-26673-1>

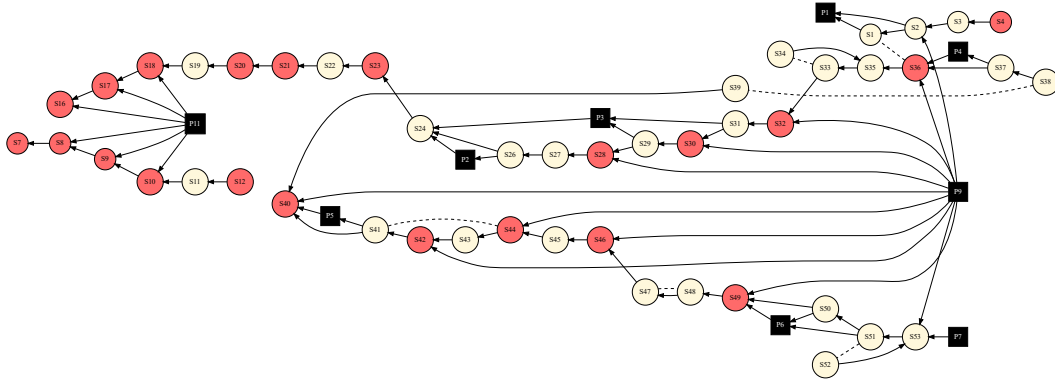


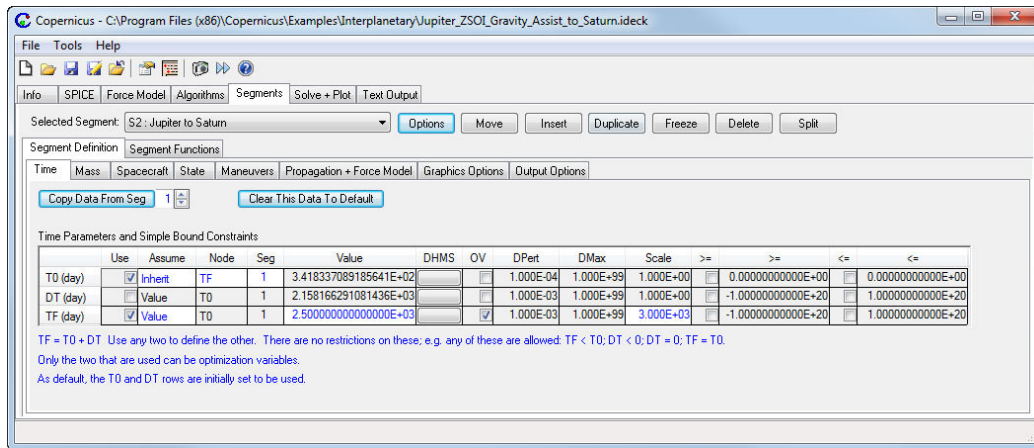
(a) GMAT (an open-source tool for space mission design and navigation, developed at GSFC). (b) FreeFlyer (a commercial astrodynamics software package).



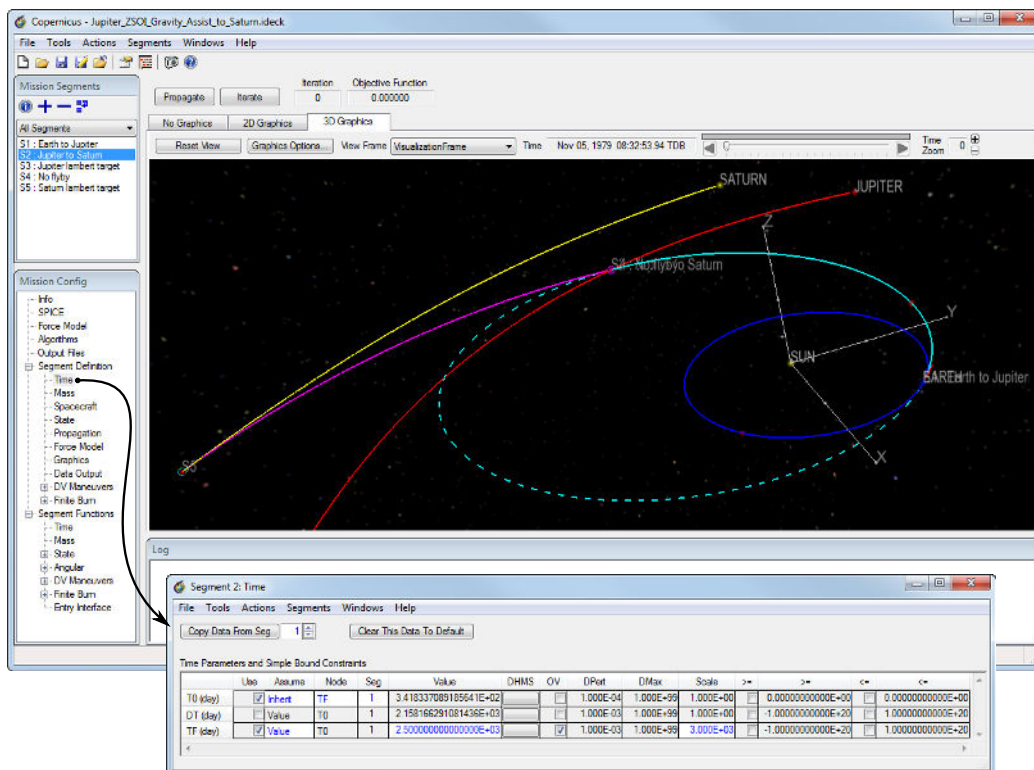
(c) Copernicus.

Figure 6: Comparison of Trajectory Tool State Dialogs (GMAT, FreeFlyer, and Copernicus). Copernicus uses a unique architecture where the trajectory segment is the fundamental mission unit, rather than a spacecraft or event sequence. Differing from the others, the distinctive grid interface in Copernicus allows for easily specifying variable attributes (such as optimization scales and bounds all in the same place in the GUI). No other tool is designed quite the same way (with both optimization and GUI interaction at the forefront of development).





(a) Copernicus GUI (v3.1). This was the last iteration of the deeply-nested tabbed GUI.



(b) Copernicus GUI (v4.0). All dialogs are accessed as popup windows.

Figure 8: Copernicus GUI Evolution (2012 – 2014). The original Copernicus GUI (from 1.0 – 3.1) consisted of one main window with a deeply-nested tab structure. In the 4.0 release, the old “Solve + Plot” tab (which included the 2D and 3D graphics displays) was made the main window, with two panels on the left with the segment list and a treeview field for accessing the mission configuration dialogs (which were now separate popup windows). This upgrade allowed multiple dialogs to be viewed at the same time.

environment but also a powerful engine for large-scale trajectory analysis.

General GUI Upgrades

The Copernicus GUI was extensively updated and improved at JSC (see Figure 8). The original GUI of one main window with a deeply-nested tab structure was eliminated in the 4.0 release in 2014. The updated 4.0 GUI layout had two panels on the left of the main window (Mission Segments and Mission Config), as well as a display of the log file at the bottom of the window.* The main part of the window was the old “Solve+Plot” tab, which contained the graphics tabs and the buttons used to propagate and iterate. The treeview field in the Mission Config dialog was used to access all the dialogs that were formerly in tabs (e.g., Algorithms, Segment Time, Segment State, etc.). Double-clicking an item in the treeview would open the dialog. With this upgrade, multiple dialogs could now be open at once (which was not possible before). The segment dialogs were populated based on the selected segment in the Mission Segments panel.

Models

The availability of multiple models and algorithms for different domains is a key feature of Copernicus, and adding new models and options has been a key part of Copernicus development. The models are intended to be selected by the user, who has total freedom to define the problem in a way best suited to the application. The comprehensive set of models in Copernicus enables a wide range of fidelity options – from very low to very high – to be used within the same tool. Users can easily transition from lower- to higher-fidelity simulations with just a few operations in the GUI. For example, a trajectory can be quickly converged using point-mass gravity models and then seamlessly upgraded to a high-fidelity force model for final analysis. This capability reflects a core philosophy of Copernicus: to serve as a modular and flexible tool in which the user plays an active role in the solution process. By allowing users to tailor problem setups creatively and precisely, Copernicus supports both rapid prototyping and detailed, mission-level analysis within a unified environment.

Various models and algorithms in Copernicus include: integrators/propagators (many different flavors including Adams, Runge-Kutta, and Nyström), optimal control theory, parameter optimization (including industry-standard solvers such as SNOPT, SLSQP, and IPOPT), numerical differentiation (2–9 point finite difference methods, with tuning), planetary ephemerides (selectable via SPICE), reference frames (many different versions of inertial, rotating, rotating-pulsating, and quasi-inertial frames, from SPICE and user-supplied), finite burn engine models (low thrust, high thrust, SEP, etc.), finite burn maneuver models (including SOC, OCT, and SAR), impulsive maneuvers, Lambert targeting (using the impulsive Δv maneuvers to target states such as other segments or celestial bodies, including gravity assists), state parameterizations (including Cartesian, classical orbital elements, hyperbolic, equinoctial, Delaunay, etc.), maneuver parameterizations (including spherical and cartesian), gravity assists, halo orbits, atmosphere and SRP, and gravity models (pointmass, spherical harmonic, and polyhedral).

Copernicus 5.0 (2020)

Starting in 2017, the development team began a significant activity to rewrite the Copernicus GUI. This transition was motivated by several key factors. There was growing concern about relying on a commercial toolkit for the GUI, both in terms of long-term sustainability and accessibility

*These panels were all resizable, but not undockable or movable (although this was a frequent user request, it was not possible to implement with the Winteracter toolkit).

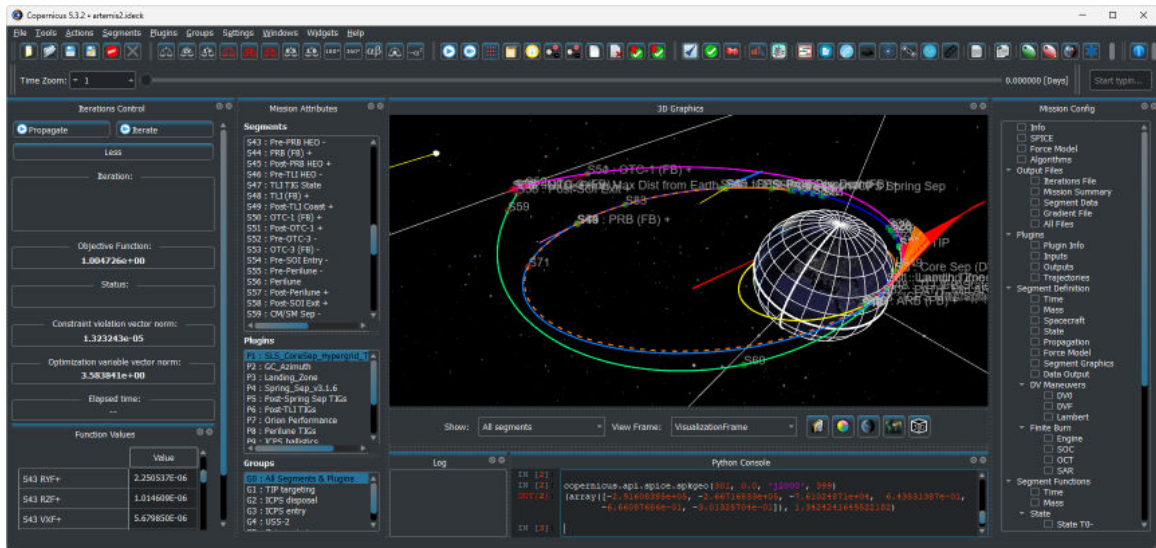
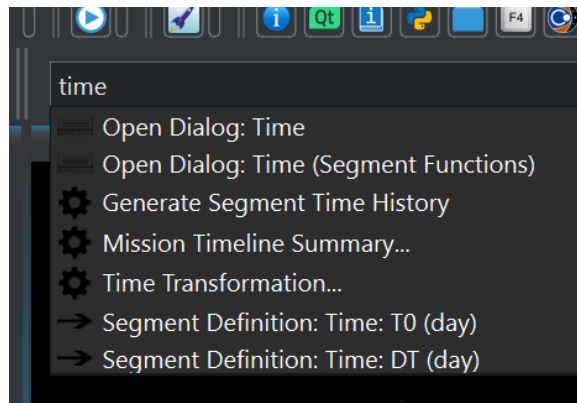


Figure 10: Copernicus 5.3.2 (2024). The latest release of Copernicus. The rewrite of the GUI in Python for the 5.0 release was a significant update, enabling many new features, usability enhancements, and cross-platform use.

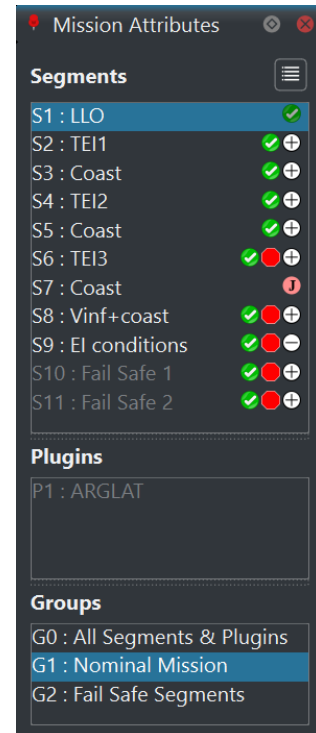


(a) A command palette allows for accessing common tasks using a text field, where the user can start typing and be presented with a list of selectable actions.

4	S1 Rz	SEG1 Rz (km)	-∞;+∞	2.0989120193247493E-01	6.000000000
5	S1 Vx	SEG1 Vx (km/s)	-∞;+∞	2.7430232846850955E-01	2.000000000
6	S1 Vy	SEG1 Vy (km/s)	-∞;+∞	2.000000000	2.000000000
7	S1 Vz	SEG1 Vz (km/s)	-∞;+∞	2.000000000	2.000000000
8	S2 T0	SEG2 T0 (day)	-∞;+∞	6.5407832277611777E-01	2.000000000

The optimization variable "S1 Rz" is:
 • Well scaled ✓
 • Within bounds ✓

(c) GUI tooltips provide help messages and useful information to the user.



(b) Widgets can be docked and reconfigured.

Figure 11: Selected GUI Features. The new Python/Qt GUI has enabled the use of modern UI features and a more dynamic and configurable system than was possible before. Being user-friendly remains a core focus for Copernicus.

for potential new developers. As Copernicus’s use cases expanded and expectations for user experience evolved, it became increasingly important to offer a more modern, user-friendly, and flexible interface. Additionally, the original GUI – built on the Winteracter framework – began to show its limitations. Certain features required by users could not be implemented within its constraints*, and support for stable and full-featured cross-platform development was lacking.† These limitations ultimately drove the need to explore more modern solutions better aligned with the future direction of the tool. Over the years, the Fortran programming language had also incorporated new features, such as built-in interoperability with C, which meant that a Fortran-only GUI solution was no longer really required, and the Python ecosystem began to achieve very wide use, meaning there were a lot of other options available. The developer and user experiences with the Python scripting interface also indicated the desirability of a Python-based solution.

The 5.0 release, completed in 2020, included some of the most significant changes to the program since the first release.⁶ The original Winteracter-based GUI was completely rewritten in Python and the open source Qt GUI toolkit, and the core Fortran code was rearchitected as a shared library that the GUI loaded (rather than having the GUI and core code intertwined). This new architecture allowed for possibilities not previously possible, such as: an API that exposed some of the underlying core routines to the user that can be used for other purposes (e.g., importing Copernicus as a generic Python library for orbital mechanics applications), Python plugins that called custom code written by the user and incorporated into the optimization problem (which can also use the API), an embedded Python console in the GUI, the ability to augment the GUI and add custom features to it, and the ability to build custom tools on top of Copernicus for other applications such as operations.[‡]

The new GUI is much more configurable than the original (e.g., users can dock/undock widgets and configure them however they want). It includes various refinements to make the tool easier to use and accessible (see Figures 10 – 11). Font size can be increased, tooltips provide useful information, variables can be clicked and the dialog automatically opened where they are defined, and there are numerous themes (possibly a unique feature among NASA’s trajectory tool suite).

As part of these upgrades to the tool, the development environment was also updated. The developers switched from Subversion to Git for version control and GitLab for issue tracking.[§] An automated Continuous Integration (CI) system was set up using GitLab, where each commit is unit tested, replacing the manual running of unit tests up to that point. CMake was used to enable cross-platform building (on Windows, this auto-generates the Visual Studio project automatically, replacing the hand-modified one that had been used since 2007; on Linux and macOS, CMake-generated make files are used[¶]). Code coverage and other metrics are generated, and developer documentation is auto generated and deployed. The end result is a much more efficient developer workflow.

*For example, Winteracter-based GUIs were limited to one graphics window, which precluded the ability to show multiple 3D scenes simultaneously.

†A Mac build of Copernicus (released with v4.5) using the Winteracter/X11 OpenMotif-based library was developed, but was found to have many stability and usability issues.

‡The internal architecture of Copernicus separates the core computational engine as a standalone library. This library has never been publicly released, but has been used internally at JSC. The Python API is an evolution of that basic concept in a modern form that is potentially more user-friendly for trajectory analysts.

§Originally the JSC development team used Trac, and then later Redmine, for issue tracking.

¶Visual Studio (with Intel Fortran integrations) had always been used to compile Copernicus on Windows since the migration from CVF, while on Linux the SCons tool was used originally and then later the Foray tool.

RECENT WORK

At JSC, after many years of studies, planning, and analysis, Orion mission design has entered a period of pre-flight data product generation and real-time operations support, and many Copernicus users and developers have shifted to this mode as well. Design and optimization of the Orion trajectories (including nominal missions and off-nominal contingencies) remains a core function of Copernicus, but post-processing, data product generation (reports), pre-flight scans, and operations support are also major components. To prepare for this, a great deal of work has been done on the Copernicus Python API. This API provides a foundation upon which to build scripts and other tools that can perform these other functions, using the core capabilities of Copernicus. The flexibility of the Python ecosystem has been an enabler of many new use cases that were not possible with the old statically-compiled GUI. For example, user-created Python plugins can now leverage any Python library for computation, and custom displays can be added to the GUI for greater insight into a user's specific use case.

In 2024, Intel discontinued its “classic” Fortran compiler (ifort) and replaced it with a new LLVM-based version (ifx). On Windows and Linux, the next release of Copernicus (5.4) will be compiled with this new compiler (some minor updates were required in the code base). On macOS, work has proceeded on making an Apple Silicon native build. Since Intel did not port ifort or ifx to the new processor architecture, recent releases for Mac have had to run in Apple's x86 Rosetta 2 emulation mode. A native build requires switching to a native compiler, and the GNU Gfortran compiler was selected. This port required slightly more effort since Copernicus was using some nonstandard Fortran idioms and internal libraries that were supported by CVF/ifort/ifx that are not available in GFortran, so replacements had to be implemented.

MISSIONS

Copernicus has been used on numerous projects over the years and has become one of NASA's main tools for advanced spacecraft trajectory optimization. It is also the core tool used at JSC to answer the “what if” type questions that feed into the design of the overall human spaceflight architecture, from Constellation to Artemis to Human Landing System (HLS). Its use (2006–present) from the early days of the Orion project was described in an earlier paper.¹ The LCROSS mission (2009) was the first flight mission that used Copernicus, which was one of the key tools used to understand the lunar impact trajectory for different launch dates and geometries.²⁰ On November 14, 2022 the CAPSTONE spacecraft entered into a lunar NRHO designed using Copernicus.²¹ The CAPSTONE ballistic lunar transfer trajectory was also designed using Copernicus.²² On November 16, 2022, the Artemis I mission launched, as the first flight test of the SLS launch vehicle and Orion spacecraft.¹ The trajectory was a lunar DRO mission, designed in Copernicus. It was the most complicated trajectory flown by a crew-rated spacecraft in decades, and the first JSC-designed Copernicus trajectory to fly. The mission splashed down on December 11, 2022. For the first time, Copernicus was also used to support JSC Mission Control Center (MCC) for flight operations during the mission.²³

Another first for Copernicus was on February 22, 2024, when the Intuitive Machines IM-1 “Odyssey” spacecraft landed on the lunar surface. IM-1 flew a trajectory designed exclusively in Copernicus, making it the trajectory design tool used for mankind's first commercial lunar lander mission. Subsequently, Copernicus was used to design the IM-2 “Athena” spacecraft's trajectory which landed on the Moon on March 6, 2025, carrying onboard a strip from Cesar Ocampo's iconic

hat and a Copernicus logo (see Figure 12).^{*} Copernicus is also being used to design trajectories and support operations for IM’s IM-3 and IM-4 missions as part of NASA’s Commercial Lunar Payload Services (CLPS) initiative, as well as IM’s Lunar Data Network constellation which will be used by programs including NASA’s Near Space Network Services (NSNS).

At NASA JSC and other centers, Copernicus is being used to design the Artemis II, III, IV trajectories and beyond. It is being used for designing the complex HLS trajectories to and from the lunar surface, and the deployment of the Gateway station to a lunar NRHO. This work continues, with the ultimate goal to return humans to the Moon and eventually to Mars.

FUTURE WORK

The 5.4 release of Copernicus is currently under development.[†] Copernicus continues to evolve and new features contemplated, including: graphics updates such as multiple 3D graphics windows (made possible by the new Qt-based GUI), high-fidelity celestial body terrain, and direct manipulation of the trajectory in the graphics;²⁴ more internal changes to allow for more flexible use cases, including a new input deck format; consolidation of the Python scripting interface and API; event finding (e.g., stopping the propagation of a segment on an event, rather than just time); expanding the API for accessing/modifying data during propagation of a segment; continuing to expand the parallelization engine;¹⁹ handling spacecraft attitude; upgraded two-body relative motion frames; and even more GUI refinements and usability enhancements.

CONCLUSIONS

The history of the first quarter-century of Copernicus, as well as the impact of its creator Dr. Cesar Ocampo, has been presented. Copernicus is an example of a NASA software tool (developed in partnership with academia, government, and commercial partners) that has been actively maintained and developed, growing and evolving as new software technologies and mission design and optimization methods become available. It has become a workhorse tool at JSC and in commercial space for advanced mission design, and is a vital resource for the U.S. space program in the quest to develop a sustained human presence beyond Earth orbit.

ACKNOWLEDGMENTS

Special thanks to Cesar A. Ocampo (1967 – 2024), the original creator of Copernicus, a teacher, colleague, mentor, and friend who had a profound impact on trajectory design at NASA, and on the lives of many. Some of the biographical and historical material in this paper is based on unpublished material by Cesar written in 2020. The authors also wish to thank all those who have contributed to Copernicus over the years, especially: Açmae El Yacoubi, Anu Kamath, Brian Killeen, Christopher Foster, David Lee, Elizabeth Williams, Fady Morcos, Jerry Condon, Juan Senent, Keven Bokelmann, Leonard Kramer, Mark Jesick, Matthew Ruschmann, Max Widner, Nadège Pie, Quentin

^{*}Cesar Ocampo passed away on September 16, 2024 in Houston, Texas, but not before learning that the Copernicus logo would be included on the lander as a tribute by his former students and colleagues. We would like to acknowledge his family for their donation of his hat. On October 25, 2024 the flag at the JSC MCC was also flown at half staff in his honor.

[†]“Copernicus has and will continue to revolutionize the way we and future analysts think about and design current and future spacecraft missions. In one single system we can solve a broad range of trajectory problems to the levels of accuracy needed. I have used the system to teach students of all ages, solve real problems, and explore new problems. A good system like this will never be completed . . . it will evolve indefinitely.” – Cesar Ocampo



(a) Closeup of the IM-2 Lander.



(b) Copernicus logos through the years.



(c) Cesar Ocampo at JSC in 2023.

Figure 12: A strip of Cesar's iconic hat (zip-tied to cables) was installed on the Nova-C IM-2 lander, along with a modified Copernicus logo patch. It landed on the Moon on March 6, 2025.

Moore, Randy Eckman, Rob Falck, Robert Harpold, Ryan Whitley, Sarah Faron, Sean McArdle, Trent Couse, and Virginia Martin.

TERMINOLOGY

API	Application Programming Interface	LCROSS	Lunar Crater Observation and Sensing Satellite
CAPSTONE	Cislunar Autonomous Positioning System Technology Operations and Navigation Experiment	LLVM	Low Level Virtual Machine
CDDF	Center Director Discretionary Funding	LTTT	Low-Thrust Trajectory Tools
CLPS	Commercial Lunar Payload Services	MCC	Mission Control Center
CI	Continuous Integration	NASA	National Aeronautics and Space Administration
CVF	Compaq Visual Fortran	NRHO	Near Rectilinear Halo Orbit
DEC	Digital Equipment Corporation	NSNS	Near Space Network Services
DRO	Distant Retrograde Orbit	OCT	Optimal Control Theory
EG	Aeroscience and Flight Mechanics Division	SAR	Single Axis Rotation
GEO	Geosynchronous Orbit	SEP	Solar Electric Propulsion
GMAT	General Mission Analysis Tool	SIRTF	Space Infrared Telescope Facility
GSFC	Goddard Space Flight Center	SLS	Space Launch System
GTO	Geosynchronous Transfer Orbit	SOC	Sub-Optimal Control
GUI	Graphical User Interface	SOTY	Software of the Year
HLS	Human Landing System	SPICE	Spacecraft, Planet, Instruments, C-matrix and Events
HPC	High Performance Computing	SRP	Solar Radiation Pressure
HSC	Hughes Space and Communications Company	TIM	Technical Interchange Meeting
IM	Intuitive Machines	UT	University of Texas at Austin
ISP	In-Space Propulsion	UI	User Interface
JPL	Jet Propulsion Laboratory	VSE	Vision for Space Exploration
JSC	Johnson Space Center	XIPSTOP	Xenon Ion Propulsion System Trajectory Optimization Program
JSON	JavaScript Object Notation		

REFERENCES

- [1] J. Williams, T. F. Dawn, and A. L. Batcha, "A History of Orion Mission Design, Copernicus Software Development, and the Artemis I Trajectory," AAS/AIAA Astrodynamics Specialist Conference, Aug. 2023. AAS 23-241.
- [2] C. Ocampo, "An Architecture for a Generalized Trajectory Design and Optimization System," *Proceedings of the Conference: Libration Point Orbits and Applications* (G. Gómez, M. W. Lo, and J. J. Masdemont, eds.), World Scientific Publishing Company, June 2003, pp. 529–572. Aiguablava, Spain.
- [3] 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.
- [4] C. Ocampo, J. S. Senent, and J. Williams, "Theoretical Foundation of Copernicus: A Unified System for Trajectory Design and Optimization," *4th International Conference on Astrodynamics Tools and Techniques. Madrid, Spain.*, May 2010.
- [5] J. Williams, J. S. Senent, and D. E. Lee., "Recent Improvements to the Copernicus Trajectory Design and Optimization System," *Advances in the Astronautical Sciences*, Vol. 143, 2012. AAS 12-236.
- [6] J. Williams, A. Kamath, R. Eckman, G. Condon, R. Mathur, and D. Davis, "Copernicus 5.0: Latest Advances in JSC's Spacecraft Trajectory Optimization and Design System," AAS/AIAA Astrodynamics Specialist Conference, Portland, ME, Aug. 2019. AAS 19-719.
- [7] C. Ocampo, "Computer Graphics Applied to the Visualization of Two-Body Orbital Mechanics Problems," 28th Aerospace Sciences Meeting, Jan. 1990. AIAA 90-0075.
- [8] C. Ocampo, "Solar Orbit Options for Space Infrared Telescope Facility (SIRTF)," Jet Propulsion Laboratory Inter-Office Memorandum, Aug. 1991. 312/91.2-1684.
- [9] C. Ocampo, "Trajectory Analysis for the Lunar Flyby Rescue of AsiaSat-3/HGS-1," *Annals of the New York Academy of Sciences*, Vol. 1065, No. 1, 2005, pp. 232–253.
- [10] C. A. Ocampo and G. W. Rosborough, "Transfer Trajectories for Distant Retrograde Orbiters of the Earth," AIAA Spaceflight Mechanics Meeting, Feb. 1993.
- [11] C. A. Ocampo, "Method of Simultaneously Reducing Inclination and Eccentricity for Geostationary Orbit Transfer," Jan. 2002. US Patent 6,341,749.
- [12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in Fortran 77: The Art of Scientific Computing*. Cambridge University Press, 2nd ed., 1992.
- [13] 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.
- [14] R. Mathur and C. A. Ocampo, "An Architecture for Incorporating Interactive Visualizations into Scientific Simulations," *Advances in the Astronautical Sciences*, Vol. 127, 2007. AAS 07-157.
- [15] J. Williams, "A New Architecture for Extending the Capabilities of the Copernicus Trajectory Optimization Program," *Advances in the Astronautical Sciences: Astrodynamics 2015*, Vol. 156, 2016. AAS 15-606.
- [16] S. L. McCarty and M. McGuire, "Parallel Monotonic Basin Hopping for Low Thrust Trajectory Optimization," *28th AIAA/AAS Space Flight Mechanics Meeting*, Jan. 2018. AIAA 2018-1452.
- [17] S. Lionel, "Has it really been 35 years?," <https://stevelionel.com/drfortran/2013/10/02/has-it-really-been-35-years/>, Oct. 2013.
- [18] J. Williams, R. D. Falck, and I. B. Beekman, "Application of Modern Fortran to Spacecraft Trajectory Design and Optimization," *2018 Space Flight Mechanics Meeting, AIAA SciTech Forum*, 2018. AIAA 2018-1451.
- [19] Q. Moore, J. Williams, and B. J. Killeen, "A New Architecture for Parallelization of Complex Spacecraft Trajectory Optimization Scans," AAS/AIAA Astrodynamics Specialist Conference, Aug. 2023. AAS 23-252.
- [20] D. S. Cooley, K. F. Galal, K. Berry, L. Janes, G. Marr, J. Carrico, and C. Ocampo, "Mission Design for the Lunar CRater Observation and Sensing Satellite (LCROSS)," AIAA/AAS Astrodynamics Specialist Conference, Toronto, Ontario Canada, Aug. 2010. AIAA 2010-8386.
- [21] J. Williams, D. E. Lee, R. L. Whitley, K. A. Bokelmann, D. C. Davis, and C. F. Berry, "Targeting Cislunar Near Rectilinear Halo Orbits for Human Space Exploration," AAS/AIAA Space Flight Mechanics Meeting, Feb. 2017. AAS 17-267.
- [22] N. L. Parrish, E. Kayser, S. Udupa, J. S. Parker, B. W. Cheetham, and D. C. Davis, "Survey of Ballistic Lunar Transfers to Near Rectilinear Halo Orbit," AAS/AIAA Astrodynamics Specialist Conference, Portland, ME, Aug. 2019. AAS 19-740.
- [23] R. A. Eckman, C. P. Barrett, B. J. Killeen, and A. L. Batcha, "Trajectory Operations of the Artemis I Mission," AAS/AIAA Astrodynamics Specialist Conference, Aug. 2023. AAS 23-363.
- [24] R. Mathur, "Visual Interactive Trajectory Design," *29th AAS/AIAA Space Flight Mechanics Meeting*, Jan. 2019. AAS 19-449.