JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 3, May–June 2010

Onboard Autonomous Targeting for the Trans-Earth Phase of Orion

Belinda G. Marchand*
University of Texas at Austin, Austin, Texas 78712
Michael W. Weeks†
NASA Johnson Space Flight Center, Houston, Texas 77058
and
Chad W. Smith‡ and Sara Scarritt‡
University of Texas at Austin, Austin, Texas 78712

DOI: 10.2514/1.42384

The present investigation focuses on one aspect of the autonomous targeting process used onboard during the Orion trans-Earth injection phase, specifically, a fast and robust algorithm that identifies a feasible return trajectory, one that meets the entry constraints without exceeding the fuel available. Unlike earlier Apollo missions, Orion seeks to land near the polar regions of the moon. Thus, a substantial plane change maneuver is required before returning to Earth. To reduce the fuel expenditure associated with this plane change, a three-maneuver sequence is employed during the return phase. An autonomous onboard targeting process for precision entry (one that incorporates multiple coordinated trans-Earth maneuvers) is sought in the event of loss of communication with the ground. The latter scenario presents a very unique challenge: one never before required of any Apollo vehicle. The Apollo missions also benefited from flexible entry requirements in contrast to Orion. Precision targeting in multibody regimes has only been previously demonstrated in unmanned sample return missions such as Genesis. The formulation presented here ensures that the entry constraints are met without violating the available fuel budget.

Nomenclature

$A_{k,k-1}$	=	3×3 submatrix of $\Phi(t_k, t_{k-1})$, upper left corner
$\mathbf{a}_k^+$	=	outgoing inertial acceleration of the vehicle at patch point k
$\mathbf{a}_k^-$	=	incoming inertial acceleration of the vehicle at patch point k
$B_{k,k-1}$	=	3×3 submatrix of $\Phi(t_k, t_{k-1})$, upper right corner
C_k	=	partial derivative of $\mathbf{y}(t_k)$ relative to $\mathbf{X}(t_k)$
$C_{k,k-1}$	=	3×3 submatrix of $\Phi(t_k, t_{k-1})$, lower left corner
$D_{k,k-1}$	=	3×3 submatrix of $\Phi(t_k, t_{k-1})$, lower right corner
$\hat{\boldsymbol{e}}_N$	=	unit vector due east and normal to the line of longitude of the entry site
$\mathbf{f}(\mathbf{X})$	=	nonlinear vector state equations as function of the state vector, $\mathbf{X}$
$\mathbf{h}(\mathbf{X})$	=	nonlinear vector function that relates the path constraints to the state $\mathbf{X}$
h_N	=	vehicle's altitude at entry, associated with the Nth patch point
$m(t)$	=	spacecraft mass
$\dot{m}_p$	=	propellant mass flow rate, constant over a burn subarc
$\hat{\boldsymbol{n}}_N$	=	unit vector due north along the line of longitude of the entry site
$\mathbf{r}_k$	=	vehicle's inertial position vector, an element of $\mathbf{X}(t_k)$
t	=	time
$\mathbf{u}$	=	inertial thrust vector direction, constant over a burn subarc
$\mathbf{v}_k$	=	vehicle's inertial velocity vector, an element of $\mathbf{X}(t_k)$
$\mathbf{v}_k^+$	=	outgoing inertial velocity of the vehicle at patch point k
$\mathbf{v}_k^-$	=	incoming inertial velocity of the vehicle at patch point k
x, y, z	=	components of the vehicle's Earth-centered position vector in inertial coordinates
$\mathbf{X}$	=	vehicle's state vector at a given point in time
$\dot{x}, \dot{y}, \dot{z}$	=	components of the vehicle's Earth-centered inertial velocity in inertial coordinates
$\hat{\boldsymbol{x}}_E$	=	unit vector along the Earth's mean equator and aligned with the prime meridian
$\hat{\boldsymbol{x}}_G$	=	inertial unit vector along the equinox of J2000 and on the mean equator plane
$\hat{\boldsymbol{x}}_N$	=	unit vector directed radially from the center of the Earth to the entry site
$\mathbf{X}^*$	=	state vector along an unspecified neighboring trajectory at a given point in time
$\mathbf{X}_k^+$	=	vehicle's outgoing state vector at time t_k
$\mathbf{X}_k^-$	=	vehicle's incoming state vector at time t_k
$\mathbf{y}(t_k)$	=	constraint vector associated with the patch point at time t_k
$\hat{\boldsymbol{y}}_E$	=	unit vector normal $\hat{\boldsymbol{x}}_E$ and tangent to the Earth's mean equator
$\hat{\boldsymbol{y}}_G$	=	inertial unit vector on the mean equator plane and normal to the equinox of J2000
$\hat{\boldsymbol{z}}_E$	=	unit vector normal to the Earth's mean equator
$\hat{\boldsymbol{z}}_G$	=	inertial unit vector normal to the mean equator and equinox of J2000
γ_N	=	vehicle's inertial flight-path angle at entry, associated with the Nth patch point
$\delta\mathbf{X}$	=	implies a noncontemporaneous variation relative to the current path

Presented as Paper 7262 at the AIAA Guidance, Navigation, and Control Conference and Exhibit, Honolulu, HI, 18–21 August 2008; received 25 November 2008; revision received 23 September 2009; accepted for publication 3 February 2010.

*Assistant Professor, Aerospace Engineering and Engineering Mechanics, 210 East 24th Street. Member AIAA.
†Guidance, Navigation, and Control Engineer, Aeroscience and Flight Mechanics Division, 2101 NASA Parkway, Code EG-6. Member AIAA.
‡Graduate Student, Aerospace Engineering and Engineering Mechanics, 210 East 24th Street. Student Member AIAA.

$\delta\mathbf{X}'$ = implies a contemporaneous variation relative to the current path
δ_N = vehicle's inertial declination at entry, associated with the Nth patch point
$\delta\mathbf{y}(t_k)$ = noncontemporaneous variation of the constraint vector relative to the current path
λ_N = vehicle's geocentric longitude at entry, associated with the Nth patch point
$\Phi(t_k, t_{k-1})$ = state transition matrix associated with the forward flow from t_{k-1} to t_k
χ_N = vehicle's inertial flight-path azimuth at entry, associated with the Nth patch point

I. Introduction

THE onboard flight software for the Crew Exploration Vehicle (CEV) must provide the capability to autonomously target, at any time, a specified Earth entry state through a sequence of three trans-Earth injection (TEI) maneuvers. In doing so, the algorithm must ensure that the associated fuel requirements do not exceed the amount of fuel available onboard at that time. The three-maneuver sequence is graphically illustrated in Fig. 1.

The goal of the first two maneuvers is to minimize the final injection that places the spacecraft on a path to Earth for a given set of entry constraints. Specifically, the vehicle must meet up to five constraints at entry interface: flight-path angle (FPA), latitude, longitude, altitude, and azimuth. In contingency scenarios, some of these constraints may be relaxed, but the targeting algorithm must at least be able to ensure an entry altitude and flight-path angle at all times. Some constraints are functions of latitude set by the Earth–moon antipode. In earlier designs, the targeting of these final constraints is controlled mainly through the final TEI maneuver in the sequence. Although this approach does address some of the goals, it does not fully exploit the third-body effects that influence the path. Certainly, an end-to-end targeting process that simultaneously adjusts all three maneuvers in a fully perturbed model can add flexibility during the targeting process and ensure that the overall cost does not exceed the fuel budget.

Targeting, in this case, refers to the numerical process of identifying feasible solutions to a problem that is subject to both path and control input constraints. This is different from the goal of an optimizer, which seeks to minimize some cost index subject to a set of user-specified path and control input constraints. In an onboard scenario, optimality is not as important as feasibility. Certainly, an algorithm that offers significant improvement in computational speed is preferred. A two-level targeter [1] that allows for the incorporation of path constraints [2,3] offers many advantages in this respect. A constrained two-level targeter [3] is a generalized targeting algorithm for the analysis of constrained dynamical systems. One of the most appealing aspects of the methodology is its conceptual simplicity and numerical efficiency in contrast to trajectory optimizers.

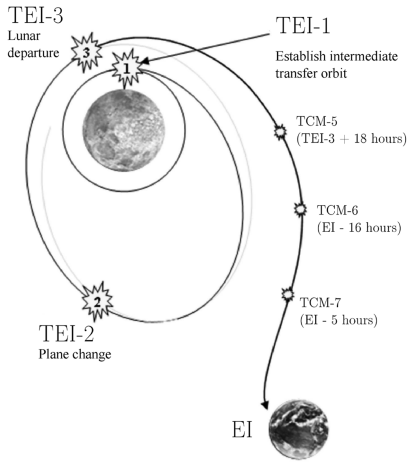

Fig. 1 Three-dimensional representation of the TEI sequence.

Of course, a constrained two-level targeting algorithm [3] is not designed to be self-starting or autonomous. Autonomy, in the present context, refers to the process of 1) identifying a suitable startup arc for the targeting algorithm and 2) reconverging the solution based on an updated set of departure and entry conditions. It also entails any numerical checks and balances necessary to ensure monotonic convergence throughout the process and how to address the possibility that a feasible solution does not exist. If a feasible solution is not available, the process must be able to identify suitable contingency scenarios to target instead. At all times, the targeting process must ensure that the solutions meet the specified set of entry constraints without exceeding the total fuel available onboard at the time. Each of these aspects of the autonomous targeting process is an area of study onto itself.

The onboard determination of the startup arc, for instance, is accomplished in one of two ways. The simplest and most common approach is to generate a database of optimal solutions over a time interval of interest, before departure, and use those as nominal scenarios that can be adjusted by a targeter as needed onboard [4,5]. More recent methods [6], specifically tailored around the Orion trans-Earth injection phase, consider the use of infeasible solutions (i.e. with state and time discontinuities) based on a series of two-body approximations. Generally speaking, both methods are suitable for the generation of an initial guess in this case. Even if a solution is infeasible initially, the two-level targeter can accommodate these types of discontinuities during the numerical process.

The present investigation is strictly focused on the mathematical developments and subsequent validation necessary to implement a two-level targeter during the Orion trans-Earth injection phase. Aside from the relevant entry constraint formulations, this study also presents the first successful implementation of a multi-patch-point constraint in the context of a two-level targeting process. Specifically, a total cost constraint is developed and validated. Here, the total cost is determined as the sum of the three TEI maneuvers in the sequence. This particular constraint formulation is the key to the success of the two-level targeter during an onboard determination. It ensures that all feasible solutions identified do not exceed the fuel available onboard. Aside from these mathematical preliminaries, the numerical studies performed serve to enhance the knowledge base on the underlying sensitivities associated with the three-maneuver sequence. This will be particularly useful in future studies that aim at addressing closed-loop computational autonomy.

II. Constrained Two-Level Corrector

A constrained two-level targeter [3] is largely based on a linearized description of the dynamical model, though the process also exhibits some elements commonly seen in optimization. In fact, the standard two-level targeter [1] bears many similarities with earlier methods used for the analysis of optimal multiple flyby trajectories [7]. In implementing a two-level targeter, the trajectory is first divided into segments by introducing a series of intermediate targets known as patch states. A simple iterative targeter then introduces impulsive maneuvers at the interior patch states until position continuity across all segments is achieved; this is referred to as the level-I process [1]. The second stage, the level-II process, adjusts the shape of the trajectory by spatially and temporally relocating the patch states to drive the velocity discontinuities to zero [1]. For each level-II iteration, an iterative level-I targeting process is completed across each trajectory segment. The end result is a trajectory in the vicinity of the startup arc that is continuous in position and velocity. The constrained two-level targeter [3] incorporates path constraints into the targeting process. The linear corrections are identified as the minimum norm solution [8]. A minimum norm solution is specifically selected to identify the smallest changes in the control parameters which lead to a path that meets the specified constraints. This is an important element of the process, because the fundamental equations behind the targeting algorithm are linear.

An immediate advantage of the generalized constrained two-level targeter formulation [3] is that it is independent of the dynamical model or the reference frames selected. This allows for a simple implementation of additional perturbations into the model. In the present analysis, the bodies that influence the spacecraft motion are the Earth, the moon, and the sun. Although nongravitational perturbations are not considered here, nonspherical body effects, solar radiation pressure, atmospheric drag, and other perturbations are easily incorporated into the process without the need to change the targeting algorithm itself.

The Orion onboard targeting algorithm benefits from earlier studies [2,9] that develop and validate the altitude and flight-path-angle constraints. Though a formulation for the true latitude constraint, as defined relative to a nonspherical Earth, is not presently available, one is not yet necessary. At this point, the Earth is assumed spherical. Thus, the geocentric latitude is equivalent to the inertial declination previously developed [2,9]. The flight-path azimuth and geocentric longitude constraints, both measured in time-varying coordinate systems, are developed here.

The existing two-level targeter framework [2,3,9] already addresses constraints that depend on a single patch state. There is, however, no precedent for constraints that depend on multiple patch states. For instance, the cost constraint placed on the sum of the TEI maneuvers represents such a multi-patch-point constraint. This particular constraint introduces unprecedented flexibility during the convergence process, particularly in dynamically sensitive problems. The constraint allows the targeter to essentially decide how to best distribute fuel resources for trajectories that employ multiple maneuvers. The targeter solutions are compared against neighboring optimal solutions to specifically highlight the computational advantages of the two-level process.

A. Dynamical Model

Let $\mathbf{X} = [\,\mathbf{r}^T \quad \mathbf{v}^T\,]^T$, where $\mathbf{r} = \mathbf{r}(t)$ denotes the time-varying position vector and $\mathbf{v} = \mathbf{v}(t)$ is the corresponding velocity vector. Then the nonlinear differential equations that govern the evolution of $\mathbf{X}$ in time are generally represented as

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t) \tag{1}$$

Let $\mathbf{X}^* = \mathbf{X}^*(t)$ denote the state along a reference trajectory that satisfies Eq. (1) and exists in the vicinity of the current path, defined by $\mathbf{X}$. Then the difference (at time t_k) between the state along the current path and that along the reference trajectory is given by

$$\delta\mathbf{X}'_k = \mathbf{X}^*_k - \mathbf{X}_k \tag{2}$$

In the theory of calculus of variations, the quantity $\delta\mathbf{X}'_k$ is classically known as a contemporaneous variation because it represents the difference between two quantities that are evaluated at the exact same time. If $\mathbf{X}^*_k$ is in the immediate vicinity of $\mathbf{X}_k$, and δt_k represents a small variation in time, then the noncontemporaneous variation $\delta\mathbf{X}_k$ may be approximated (to first order) as

$$\delta\mathbf{X}_k = \mathbf{X}^*(t_k + \delta t_k) - \mathbf{X}(t_k) \approx \delta\mathbf{X}'(t_k) + \dot{\mathbf{X}}(t_k)\delta t_k \tag{3}$$

Thus, the linear system associated with Eq. (1), specifically linearized about the current path, admits a solution of the form

$$(\delta\mathbf{X}_k - \dot{\mathbf{X}}_k\delta t_k) = \Phi(t_k, t_{k-1})(\delta\mathbf{X}_{k-1} - \dot{\mathbf{X}}_{k-1}\delta t_{k-1}) \tag{4}$$

where $\Phi(t_k, t_{k-1})$ denotes the state transition matrix. When the elements of $\mathbf{X}$ are only the position and velocity of the spacecraft [3], the 6×6 matrix $\Phi(t_k, t_{k-1})$ can be subdivided into four 3×3 submatrices, such that

$$\Phi(t_k, t_{k-1}) = \begin{bmatrix} A_{k,k-1} & B_{k,k-1} \\ C_{k,k-1} & D_{k,k-1} \end{bmatrix} \tag{5}$$

The subscript pair $k, k-1$ denote the direction of the propagation. For example, the right subscript $k-1$ denotes the start time t_{k-1} of the propagation, and the left subscript k reflects the terminal time t_k.

Now consider two adjacent segments along a trajectory, with the first defined by $t_{k-1} \le t \le t_k$ and the next defined by $t_k \le t \le t_{k+1}$. Suppose that a velocity discontinuity in the path exists at t_k. This discontinuity represents an impulsive maneuver at that point. An impulsive maneuver can be mathematically represented as the difference between the incoming velocity $\mathbf{v}_k^-$ along the first of the two segments and the outgoing velocity $\mathbf{v}_k^+$ along the second: specifically, $\Delta\mathbf{v}_k = \mathbf{v}_k^+ - \mathbf{v}_k^-$. To allow for the possibility that such an impulsive maneuver exists at the kth patch state, the variational equations for each segment must be formulated separately. The variational equations associated with the first segment may then be expressed as

$$\begin{bmatrix} \delta\mathbf{r}_k - \mathbf{v}_k^-\delta t_k \\ \delta\mathbf{v}_k^- - \mathbf{a}_k^-\delta t_k \end{bmatrix} = \begin{bmatrix} A_{k,k-1} & B_{k,k-1} \\ C_{k,k-1} & D_{k,k-1} \end{bmatrix} \begin{bmatrix} \delta\mathbf{r}_{k-1} - \mathbf{v}_{k-1}^+\delta t_{k-1} \\ \delta\mathbf{v}_{k-1}^+ - \mathbf{a}_{k-1}^+\delta t_{k-1} \end{bmatrix} \tag{6}$$

if the integration proceeds forward from t_{k-1} to t_k. For the second segment, if the integration proceeds backward from t_{k+1} to t_k, the variational equations are given by

$$\begin{bmatrix} \delta\mathbf{r}_k - \mathbf{v}_k^+\delta t_k \\ \delta\mathbf{v}_k^+ - \mathbf{a}_k^+\delta t_k \end{bmatrix} \begin{bmatrix} A_{k,k+1} & B_{k,k+1} \\ C_{k,k+1} & D_{k,k+1} \end{bmatrix} \begin{bmatrix} \delta\mathbf{r}_{k+1} - \mathbf{v}_{k+1}^-\delta t_{k+1} \\ \delta\mathbf{v}_{k+1}^- - \mathbf{a}_{k+1}^-\delta t_{k+1} \end{bmatrix} \tag{7}$$

In general, the $+$ and $-$ superscript notations next to any vector or scalar variable are used here to identify the variable's value across a discontinuity. For example, an integration that proceeds backward from t_{k+1} to t_k requires an initial state vector $\mathbf{X}_{k+1}^- = [\,\mathbf{r}_{k+1}^T \quad \mathbf{v}_{k+1}^{-T}\,]^T$. In contrast, an integration that proceeds forward in time from t_{k-1} to t_k employs $\mathbf{X}_{k-1}^+ = [\,\mathbf{r}_{k-1^T} \quad \mathbf{v}_{k-1}^{+T}\,]^T$ as an initial state for the propagation. Note that the $+$ and $-$ are not applied to the position vector in this case, because it is assumed that position continuity exists across segments after a level-I process is applied.

B. Startup Arcs and Patch States

Targeters and optimizers are not self-starting processes. Both require a reasonably accurate initial guess to proceed. A startup arc may originate from a simple numerical integration or optimization process, even if it leads to a solution that does not initially meet some or all of the constraints. Alternatively, a startup solution may be identified from patched conic analysis. Even if propagation in the ephemeris model results in state discontinuities, neither the two-level targeter nor an optimizer require a feasible startup arc.

In the present study, two different sources are considered in identifying startup arcs. One approach relies on a sample block set of previously determined solutions [4,5]. From this set, a suitable startup arc (whether feasible or infeasible) is selected. The two-level targeter process is then used to identify neighboring solutions that meet the specified constraints. More recent methods [6] consider the on-demand identification of startup arcs through the use of two-body approximations. In this case, an Earth-centered initial state that meets some of the desired characteristics is numerically propagated backward toward the lunar sphere of influence. Then information about the Earth–moon antipode is used to compute (in a two-body model) the three maneuvers required to depart the vicinity of the moon en route toward the Earth. Of course, since the third-body effects in this case are not insignificant, this moon-centered propagation can lead to trajectories that do not come near the Earth segment. This leads to potentially significant position discontinuities at the interface near the lunar sphere of influence. Since the block-set method is not subject to these large position discontinuities, it is selected here as the source for startup arcs.

Once the startup trajectory is available, a series of representative patch states are selected along the path. For Orion applications, an average of 10 patch states is common. In selecting the position of the patch states, it is important to choose points that adequately capture the overall spatial and temporal features of the startup arc. However, it is most important to ensure that the points selected are not subject to extreme dynamical sensitivities. This can be assessed, for instance, by numerically propagating the patch state forward in time and evaluating the rate of state dispersion. Dynamical sensitivity implies that small changes in the initial state lead to large deviations as time

progresses. Therefore, although the analyst is generally free to specify any desired set of patch states, the convergence of the algorithm is enhanced by adequately spacing the points in time with respect to each other. It is also important that the temporal spacing is not too small between any two patch states. This introduces a certain degree of inertia into the algorithm. That is, since the process is based on linearized equations, and a minimum norm solution is further selected [8], the changes applied by the corrections process will already be the smallest possible. Thus, temporally spacing points too closely can slow down the convergence process by restricting the neighborhood of solutions explored. It also hinders the process of identifying other neighboring and potentially lower-cost solutions. Similarly, spacing the patch states too far apart in time can over-expand the search space outside the linear range, which can also introduce convergence difficulties. Although the overall patch state selection process is not presently automated, the sensitivity previously described was recognized during the course of this study. This can be used to automate the temporal spacing of patch states and, subsequently, the number and placement of patch states along a startup solution. This is a subject of ongoing study.

C. Level-I Process

The constrained two-level targeter [3] consists of two fundamental steps. The level-I process is analogous to (though not the same as) a two-body Lambert solver. Traditionally, a level-I process is a differential correction scheme [10] that seeks to identify the transfer arc between two position vectors by adjusting the departure velocity. The process differs from a Lambert targeter in that the algorithm is independent of the dynamical model. Thus, the user has the flexibility of employing any level of model complexity desired (i.e. perturbations). The level-I targeter also differs from a Lambert targeter in that the time of flight along the path is not necessarily fixed. In fact, the time of flight may be specified as an additional control variable to add flexibility in meeting the desired target. Of course, the notion of a level-I process is not limited to position targeting. The process can also be configured to target other constraints (altitude, flight-path angle, etc.) The only limitation is that the number of constraints cannot exceed the number of control variables available in the formulation. The following examples compare a traditional level-I process that targets the terminal position vector with a modified level-I scheme that targets generalized terminal constraints.

1. *Example 1: Targeting the Terminal Position Vector*

A classical position targeter [10], i.e., a level-I process, is graphically illustrated in Fig. 2. In this case, the initial and terminal times along the segment are fixed such that $\delta t_k = \delta t_{k-1} = 0$. The control variables, in this case, are restricted to the components of $\Delta \mathbf{v}_{k-1}$, the impulsive maneuver applied at t_{k-1}. If the initial time is fixed, this is equivalent to $\delta t_k \neq 0$. For a fixed time transfer [3], where $\delta t_{k-1} = \delta t_k = 0$, each iterative step of the level-I process updates the outgoing velocity along the segment by

$$\delta \mathbf{v}_{k-1}^{+} = B_{k,k-1}^{-1} \delta \mathbf{r}_k \tag{8}$$

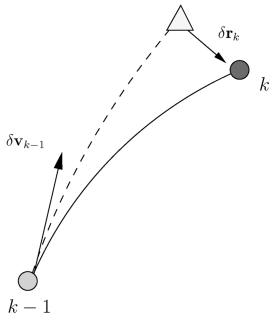

Fig. 2 Stylized representation of level-I corrections process.

Of course, it is not necessary to constrain the time of flight. In fact, added flexibility is achieved by treating the time of flight along the path as a control parameter. If the terminal segment time is free, $\delta t_k \neq 0$, a modified update is applicable:

$$\begin{bmatrix} \delta \mathbf{v}_{k-1}^{+} \\ \delta t_k \end{bmatrix} = [\, B_{k,k-1} \quad \mathbf{v}_k^{-} \,]^T ([\, B_{k,k-1} \quad \mathbf{v}_k^{-} \,][\, B_{k,k-1} \quad \mathbf{v}_k^{-} \,]^T)^{-1} \delta \mathbf{r}_k \tag{9}$$

Equation (9) represents the minimum norm solution [8] to the variational position equations when both the initial velocity and terminal time are treated as control variables.

2. *Example 2: Targeting Terminal Constraints*

Though the traditional level-I process previously described targets a terminal position vector, the process is easily tailored to target terminal constraints:

$$\mathbf{y} = \mathbf{h}(\mathbf{X}, t) \tag{10}$$

In Eq. (10), $\mathbf{y} = \mathbf{y}(t)$ represents a $p \times 1$ vector of specified constraints. The nonlinear vector function $\mathbf{h}(\mathbf{X}, t)$ indicates the relation between $\mathbf{y}$ and the state vector $\mathbf{X} = \mathbf{X}(t)$. Consider a segment defined by $t_{k-1} \leq t \leq t_k$ such that $\mathbf{y}_k = \mathbf{y}(t_k)$ represents the vector of terminal constraints. If one can identify the functional relationship between the terminal state vector $\mathbf{X}_k$ and the constraint $\mathbf{y}_k$, the linearized approximation of Eq. (10) is subsequently determined as

$$\delta \mathbf{y}_k = H_{\mathbf{X}}^{(k)} \delta \mathbf{X}_k + \mathbf{H}_t^{(k)} \delta t_k \tag{11}$$

where

$$H_{\mathbf{X}}^{(k)} = \left. \frac{\partial \mathbf{h}(\mathbf{X}, t)}{\partial \mathbf{X}} \right|_{t_k, \mathbf{X}_k} \tag{12}$$

$$\mathbf{H}_t^{(k)} = \left. \frac{\partial \mathbf{h}(\mathbf{X}, t)}{\partial t} \right|_{t_k, \mathbf{X}_k} \tag{13}$$

The partial derivatives in Eq. (11) are traditionally a part of the constrained level-II process [3], but they are employed here in a level-I scheme as a proof of concept. The goal now is to rewrite Eq. (11) such that the terminal constraint is expressed strictly as a function of the control parameters associated with the initial state. Then consider Eq. (6) when $\delta t_{k-1} = 0$ and $\delta \mathbf{r}_{k-1} = \mathbf{0}$:

$$\delta \mathbf{r}_k = B_{k,k-1} \delta \mathbf{v}_{k-1}^{+} + \mathbf{v}_k^{-} \delta t_k \tag{14}$$

$$\delta \mathbf{v}_k^{-} = D_{k,k-1} \delta \mathbf{v}_{k-1}^{+} + \mathbf{a}_k^{-} \delta t_k \tag{15}$$

Equations (14) and (15) may also be expressed in a more compact vector form:

$$\delta \mathbf{X}_k^{-} = \underbrace{\begin{bmatrix} B_{k,k-1} & \mathbf{v}_k^{-} \\ D_{k,k-1} & \mathbf{a}_k^{-} \end{bmatrix}}_{M_c} \underbrace{\begin{bmatrix} \delta \mathbf{v}_{k-1}^{+} \\ \delta t_k \end{bmatrix}}_{\mathbf{b}_c} \tag{16}$$

Substitution of Eq. (16) into Eq. (11) leads to

$$\delta \mathbf{y}_k = Q_c \mathbf{b}_c \tag{17}$$

where

$$Q_c = (H_{\mathbf{X}}^{(k)} M_c - [\, 0_{p \times 3} \quad \mathbf{H}_t^{(k)} \,]) \tag{18}$$

If $\dim\{\mathbf{y}_k\} < \dim\{\mathbf{b}_c\}$, an infinite number of solutions to Eq. (17) exist. Subsequently, a minimum norm solution [8] suggests the smallest corrections to the initial velocity and segment duration that meet the desired constraint error value $\delta \mathbf{y}_k$ at the terminal point:

$$\mathbf{b}_c = Q_c^T (Q_c Q_c^T)^{-1} \delta \mathbf{y}_k \tag{19}$$

Table 1 Patch points for level-I targeting example (ECI)

k	Time, s	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	96119.5218	334822.1786	191091.5896	44823.3558	1.3344	−0.7664	0.2778
2	397493.3471	−5895.7461	−1733.9958	−2117.5885	0.5824	−7.4621	8.0510

To illustrate the functionality of the above approach, a simple constraint-based level-I process is implemented in this example to the transfer from a sample TEI-3 point to Earth entry. The longitude constraint partials are detailed later in this paper. The altitude and latitude partials were developed in earlier investigations [2,3,9]. Since the Earth is assumed to be a spherical body, the latitude is simply determined as the inertial declination relative to the mean equatorial plane.

For the present example, TEI-3 occurs on 3 August 2018 at 19:58:05.811 UTC (Universal Time Coordinated). The TEI state and initial terminal state are given in Table 1; the DE405 ephemeris model is used for all examples. The target entry state corresponds to an altitude of 121.9 km, a flight-path angle of −6.03°, a longitude of −134.5456°, a latitude of −19.2041°, and a flight-path azimuth of 13.9960°. Table 2 summarizes the error in the entry state associated with the initial guess. The constraint-based level-I process is then employed to target the desired entry state. The results of the process are summarized in Table 3. The required maneuver, generally equivalent to TEI-3 for Orion, is listed. Clearly, a significant error exists, particularly in the first three quantities. Thus, this particular initial guess offers an excellent way to demonstrate the type of difficulties a level-I process can encounter when the quality of the initial guess is poor. At the same time, it also demonstrates that the process is surprisingly robust under some conditions.

However, convergence via a level-I process is not guaranteed over the entire lunar cycle or for all constraint combinations, particularly as the quality of the initial guess degrades. For instance, a case in which the altitude, longitude, and flight-path angle or azimuth are targeted via a level-I process, for the particular initial state employed in this example, fails to converge. This is attributed to the correlation between the specified entry constraints. The longitude, for instance, is a time-varying quantity, since it is measured in an Earth-fixed rotating frame. The flight-path azimuth is also specified relative to an Earth-fixed rotating frame, but one associated with the entry site. Thus, targeting these constraints simultaneously basically amounts to specifying that the vehicle must reach the desired target at the exact same approach angle, regardless of how the target's orientation (in this case the longitude) has changed. Since both the position vector attitude and the velocity vector orientation are time-sensitive, it is not surprising that an inadequate initial guess (such as that supplied here) can lead to convergence difficulties and sometimes nonconvergence. After all, a linear targeter searches for solutions in the vicinity of the startup arc that meet the desired criteria. If those solutions do not exist in that particular neighborhood, a linear targeter can fail. A level-I process is particularly sensitive to this. Difficulties also arise when a solution that meets the entry state exists in the neighborhood but at a significantly higher cost than the available fuel budget allows. For these reasons, a level-I process alone is not suitable for onboard targeting, though it may be suitable in simpler cases [11]. A two-level targeter, however, can easily remedy these deficiencies by introducing a larger number of control parameters and flexibility in the search for a solution.

Table 2 Initial constraint errors

Constraint	Error
Altitude, km	594.1471
FPA, deg	28.0569
Latitude, deg	56.9704
Longitude, deg	−1.3226
Azimuth, deg	0.1133

Table 3 Level-I constraint targeting results

Entry constraints	TEI-3 Δv, km/s	Iterations
Altitude, FPA	0.3627	12
Altitude, latitude	0.4465	29
Altitude, azimuth	0.3701	10
Altitude, latitude, longitude	0.6233	41
Altitude, FPA, azimuth	0.3709	10

D. Level-II Process

In a traditional level-II process [1], all velocity discontinuities are simultaneously driven to zero by adjusting the spatial and temporal properties of the patch states in a single calculation. This approach is useful when a natural solution is sought; that is, a solution that is continuous in both position and velocity. To accomplish this, a linear expression that relates the velocity discontinuity at each patch state to the positions and times of the neighboring patch points is identified [1]. Consider, for example, a trajectory characterized by N patch states (i.e. $k = 1, \ldots, N$), such that $\mathbf{X}_1$ and $\mathbf{X}_N$ denote the initial and terminal states along the path, respectively. The interior patch states are then identified by $\mathbf{X}_k$, for $k = 2, \ldots, N-1$. The velocity discontinuities introduced at these interior points, due to the level-I process, define $3(N-2)$ equations, one for each interior $\Delta\mathbf{v}_k$. These $3(N-2)$ expressions are assembled into an augmented system of equations:

$$\Delta\mathbf{v} = M\mathbf{b} \tag{20}$$

Since each patch state contributes four control parameters (time and three position elements), the matrix M is of dimension $3(N-2) \times 4N$. The number of patch states is purposefully selected to ensure that the system is always underdetermined [1]. In this formulation, each interior velocity discontinuity $\Delta\mathbf{v}_k$ contributes three rows to Eq. (20). The general form of these individual contributions [1] may be summarized as

$$\delta\Delta\mathbf{v}_k = \Delta\mathbf{v}_k^* - \Delta\mathbf{v}_k = M_k\mathbf{b}_k \tag{21}$$

Here,

$$M_k = \begin{bmatrix} \frac{\partial\Delta\mathbf{v}_k}{\partial\mathbf{r}_{k-1}} & \frac{\partial\Delta\mathbf{v}_k}{\partial t_{k-1}} & \frac{\partial\Delta\mathbf{v}_k}{\partial\mathbf{r}_k} & \frac{\partial\Delta\mathbf{v}_k}{\partial t_k} & \frac{\partial\Delta\mathbf{v}_k}{\partial\mathbf{r}_{k+1}} & \frac{\partial\Delta\mathbf{v}_k}{\partial t_{k+1}} \end{bmatrix} \tag{22}$$

is a block of the matrix M associated with $\Delta\mathbf{v}_k$: specifically, rows $3(k-2)+1$ through $3(k-2)+3$ and columns $4(k-2)+1$ through $4(k-2)+12$. The vector

$$\mathbf{b}_k = \begin{bmatrix} \delta\mathbf{r}_{k-1} \\ \delta t_{k-1} \\ \delta\mathbf{r}_k \\ \delta t_k \\ \delta\mathbf{r}_{k+1} \\ \delta t_{k+1} \end{bmatrix} \tag{23}$$

is comprised only of the elements of $\mathbf{b}$ that affect $\Delta\mathbf{v}_k$: specifically, elements $4(k-2)+1$ through $4(k-2)+12$. Note that since $\Delta\mathbf{v}_k$ depends only on the temporal and spatial coordinates of patch states associated with t_{k-1}, t_k, and t_{k+1}, the matrix M in Eq. (20) exhibits a generally sparse structure.

Since the resulting linear system of equations is underdetermined, a minimum norm solution [8] is selected to minimize the requested changes in the positions and times of the patch states [1]:

$$\mathbf{b} = M^T(MM^T)^{-1}\delta\Delta\mathbf{v} \tag{24}$$

This ensures that the corrections process seeks out solutions in the immediate vicinity of the current path. Equation (24) suggests the smallest changes in the temporal and spatial coordinates of the patch states that meet the desired constraints in the linear system. Subsequently, the positions and times of all N patch states are updated according to these suggested changes. Then an iterative level-I process is applied to achieve position continuity. The overall process is repeated in the nonlinear system until all velocity discontinuities have been reduced to some specified tolerance.

Later studies by Marchand et al. [3] present a generalized formulation of the impulsive two-level targeter that allows for the incorporation of algebraic constraints at any patch state along the trajectory. In this formulation, a constraint imposed on the kth patch state must be formulated strictly as a function of t_k, $\mathbf{X}_k^+$, and/or $\mathbf{X}_k^-$. Once the constraint vector variational equation is identified, the generalized impulsive level-II process [3] follows a similar structure to that presented by Howell and Pernicka [1]. In this case, however, the minimum norm solution [8] sought corresponds to a system of the form

$$\begin{bmatrix} \Delta\mathbf{v} \\ \delta\alpha \end{bmatrix} = \begin{Bmatrix} M \\ M_\alpha \end{Bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{b}_\alpha \end{bmatrix} \tag{25}$$

The elements of M_α are the partial derivatives of the constraints with respect to the corresponding patch states and times [3]. This particular structure is generally acceptable for the analysis presented here, with one exception. Earlier implementations and formulations of the two-level targeter, including that presented by Marchand et al. [3], do not consider constraints that depend on multiple patch states. This capability is of critical importance for Orion entry targeting. Targeting in this study entails more than achieving a prespecified entry state at Earth arrival. It is most important that the targeter identify solutions that do not exceed the fuel available onboard at the time the maneuver sequence is initiated. Thus, constraining the total mission cost is fundamental to the work presented here. Of course, the total cost depends on the $\Delta\mathbf{v}$ applied at TEI-1, TEI-2, and TEI-3. Although constraining the magnitude of individual $\Delta\mathbf{v}$ is possible within the existing targeter framework [3], this earlier formulation does not directly accommodate constraints, such as total $\Delta\mathbf{v}$ spent, that depend on the state at multiple patch states. A total cost constraint is thus formulated and tested here for applications to Orion entry targeting.

III. Trajectory Constraints

At entry interface, Orion seeks to target up to five quantities: altitude, flight-path angle, latitude, longitude, and flight-path azimuth. Furthermore, the total Δv may be constrained to satisfy the available fuel budget. Other constraints may designate, for instance, that TEI-2 and TEI-3 take place at an apse, where $\mathbf{r}\cdot\mathbf{v}=0$. The primary entry constraints of interest here, associated with the Nth patch state, are summarized in Eqs. (26–30):

$$\alpha_{N_1} = \lambda_N - \lambda_{\text{des}} \tag{26}$$

$$\alpha_{N_2} = \sin\delta_N - \sin\delta_{\text{des}} \tag{27}$$

$$\alpha_{N_3} = \sin\gamma_N - \sin\gamma_{\text{des}} \tag{28}$$

$$\alpha_{N_4} = \chi_N - \chi_{\text{des}} \tag{29}$$

$$\alpha_{N_5} = h_N - h_{\text{des}} \tag{30}$$

Naturally, these equations represent one of many ways to formulate the same constraint. Some formulations are more numerically well-behaved than others. Angular constraints can be particularly cumbersome because of quadrant ambiguities and potential singularities. However, the set selected above is suitable for the current application. The following sections focus on the determination of the analytic partial derivatives for the entry constraints and the total mission cost constraint.

A. Coordinate Frames and Entry Angles

The illustration in Fig. 3 depicts the relation between the Earth rotating frame, defined by unit vectors $\hat{\boldsymbol{x}}_E$–$\hat{\boldsymbol{y}}_E$–$\hat{\boldsymbol{z}}_E$; the entry site frame, or up–east–north frame, defined by unit vectors $\hat{\boldsymbol{r}}_N$–$\hat{\boldsymbol{e}}_N$–$\hat{\boldsymbol{n}}_N$; and the Earth-centered inertial unit vectors $\hat{\boldsymbol{x}}_G$–$\hat{\boldsymbol{y}}_G$–$\hat{\boldsymbol{z}}_G$.

The entry interface state is characterized in terms of its geocentric longitude and latitude, the vehicle altitude, and the flight-path angle and azimuth of the velocity vector. Earlier studies [2] consider the altitude and flight-path-angle constraints. Thus, the following discussion focuses on the remaining entry targets; the geocentric latitude and longitude, and the flight-path azimuth. The geocentric latitude is equivalent to the inertial declination δ_N, illustrated in Fig. 3. This particular constraint and the associated partial derivatives are also available in the literature [2]. Thus, it is not necessary to formulate new equations for the geocentric latitude constraint. At the entry interface point, only formulations for the longitude and flight-path azimuth constraints are then necessary.

In addressing the geocentric longitude constraint, identifying an expression for the inertial right ascension θ_N as a function of the entry state is useful. In this case, θ_N (measured relative to the inertial $\hat{\boldsymbol{x}}_G$ axis) is determined as

$$\tan\theta_N = \frac{\mathbf{r}_N^T\hat{\boldsymbol{y}}_G}{\mathbf{r}_N^T\hat{\boldsymbol{x}}_G} \tag{31}$$

The spherical-Earth assumption implies that the geocentric longitude is related to the right ascension as follows:

$$\theta_N = \theta_G + \lambda_N = (\theta_{g0} + \omega_e(t_N - t_0)) + \lambda_N \tag{32}$$

where θ_G represents the location of the prime meridian relative to the Earth-centered inertial x axis $\hat{\boldsymbol{x}}_G$. The time t_0 is associated with the zeroth hour (midnight) on the day of entry, and θ_{g0} represents the right ascension of the prime meridian at that time. The algorithm employed in approximating θ_{g0} is that listed by Vallado [12].

The rotational rate of the Earth is denoted by ω_e. Thus, the spacecraft longitude at entry is determined to be

$$\lambda_N = \tan^{-1}\left(\frac{\mathbf{r}_N^T\hat{\boldsymbol{y}}_G}{\mathbf{r}_N^T\hat{\boldsymbol{x}}_G}\right) - \theta_{g0} - \omega_e(t_N - t_0) \tag{33}$$

Other entry constraints of interest include the flight-path angle γ_N [2,9] and the flight-path azimuth χ_N. The formal definitions are pictorially represented in Fig. 4.

To identify an expression for the flight-path azimuth, consider an alternate representation of the spacecraft velocity in terms of the up–east–north frame. This requires a mathematical representation for $\hat{\boldsymbol{r}}_N$, $\hat{\boldsymbol{e}}_N$, and $\hat{\boldsymbol{n}}_N$ in terms of the spacecraft state at entry:

$$\hat{\boldsymbol{r}}_N = \hat{\boldsymbol{r}}_N(\mathbf{r}_N) = \frac{\mathbf{r}_N}{|\mathbf{r}_N|} \tag{34}$$

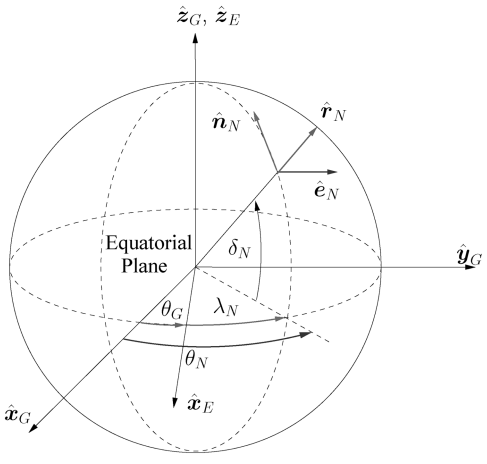

Fig. 3 Flight-path angle and azimuth.

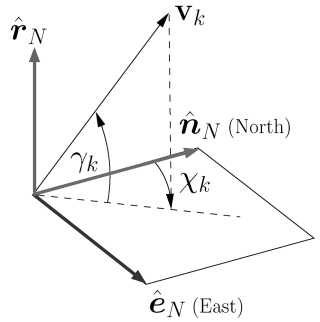

Fig. 4 Definitions of flight-path azimuth and flight-path angle.

$$\hat{\boldsymbol{e}}_N = \hat{\boldsymbol{e}}_N(\mathbf{r}_N) = \left(\frac{\hat{z}_G \times \hat{\boldsymbol{r}}_N}{|\hat{z}_G \times \hat{\boldsymbol{r}}_N|}\right) = \left(\frac{\hat{z}_G \times \mathbf{r}_N}{|\hat{z}_G \times \mathbf{r}_N|}\right) \tag{35}$$

$$\hat{\boldsymbol{n}}_N = \hat{\boldsymbol{n}}_N(\mathbf{r}_N) = \hat{\boldsymbol{r}}_N \times \left(\frac{\hat{z}_G \times \mathbf{r}_N}{|\hat{z}_G \times \mathbf{r}_N|}\right)$$
$$= \frac{1}{|\mathbf{r}_N||\hat{z}_G \times \mathbf{r}_N|}(\mathbf{r}_N \times (\hat{z}_G \times \mathbf{r}_N)) \tag{36}$$

Note that although these unit vectors are clearly time-varying, they are not explicit functions of time. They are also independent of the spacecraft velocity. The only explicit dependence apparent is on the position vector $\mathbf{r}_N$. Given these definitions, the velocity of the spacecraft (at the kth patch state) may be alternatively represented as follows:

$$\mathbf{v}_N^- = (\mathbf{v}_N^- \cdot \hat{\boldsymbol{r}}_N)\hat{\boldsymbol{r}}_N + (\mathbf{v}_N^- \cdot \hat{\boldsymbol{e}}_N)\hat{\boldsymbol{e}}_N + (\mathbf{v}_N^- \cdot \hat{\boldsymbol{n}}_N)\hat{\boldsymbol{n}}_N \tag{37}$$

Using Eq. (37) and Fig. 4, the flight-path azimuth χ_N may be expressed as

$$\chi_N = \tan^{-1}\left(\frac{\mathbf{v}_N^- \cdot \hat{\boldsymbol{e}}_N}{\mathbf{v}_N^- \cdot \hat{\boldsymbol{n}}_N}\right) = \tan^{-1}\left(\frac{\mathbf{v}_N^{-T}\hat{\boldsymbol{e}}_N}{\mathbf{v}_N^{-T}\hat{\boldsymbol{n}}_N}\right) \tag{38}$$

B. Flight-Path Azimuth

Consider the flight-path azimuth constraint α_{N_4}. From Eqs. (34–36) and (38), it is apparent that the flight-path azimuth depends explicitly on the velocity and on the position vector associated with the patch state, but not on the time at entry. Of course, the state vector in general is a function of time. Thus, as the entry state is adjusted during the corrections process, the coordinate frame in which the azimuth is evaluated changes. Therefore, in some sense, there is a dependence on time, even if not an explicit one. For notational simplicity, let

$$v_{k_{en}}^- = \sqrt{(\mathbf{v}_N^{-T}\hat{\boldsymbol{e}}_N)^2 + (\mathbf{v}_N^{-T}\hat{\boldsymbol{n}}_N)^2} \tag{39}$$

$$v_{k_e}^- = \mathbf{v}_N^{-T}\hat{\boldsymbol{e}}_N \tag{40}$$

$$v_{k_n}^- = \mathbf{v}_N^{-T}\hat{\boldsymbol{n}}_N \tag{41}$$

Recall that the partial derivative of the arctangent of a function of any variable x with respect to x is given by

$$\frac{\partial}{\partial x}[\tan^{-1} f(x)] = \left(\frac{1}{1 + f(x)^2}\right)\frac{\mathrm{d}f}{\mathrm{d}x} \tag{42}$$

Thus, the partial derivative of the azimuth with respect to the terminal position vector may be expressed as follows:

$$\frac{\partial \chi_N}{\partial \mathbf{r}_N} = \frac{1}{v_{k_{en}}^2}\left[v_{k_n}^- \mathbf{v}_N^{-T}\frac{\partial \hat{\boldsymbol{e}}_N}{\partial \mathbf{r}_N} - v_{k_e}^- \mathbf{v}_N^{-T}\frac{\partial \hat{\boldsymbol{n}}_N}{\partial \mathbf{r}_N}\right] \tag{43}$$

Similarly, the partial derivative with respect to the velocity vector is determined as

$$\frac{\partial \chi_N}{\partial \mathbf{v}_N^-} = \frac{1}{v_{k_{en}}^2}(v_{k_n}^- \hat{\boldsymbol{e}}_N^T - v_{k_e}^- \hat{\boldsymbol{n}}_N^T) \tag{44}$$

These are the only nonzero partial derivatives of the constraints in Eq. (29). Note that the partial derivative in Eq. (43) is dependent on the availability of the partial derivatives of Eqs. (35) and (36) with respect to $\mathbf{r}_N$. In this case, it is helpful to rewrite these expressions in a form convenient for differentiation. For instance, if

$$Z_\times = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{45}$$

then

$$\hat{z}_G \times \mathbf{r}_N = Z_\times \mathbf{r}_N \tag{46}$$

Equation (36) is further simplified through direct application of the vector triple-product identity:

$$\mathbf{r}_N \times (\hat{z}_G \times \mathbf{r}_N) = \hat{z}_G(\mathbf{r}_N^T\mathbf{r}_N) - \mathbf{r}_N(\mathbf{r}_N^T\hat{z}_G) \tag{47}$$

Subsequently, and after extensive algebraic manipulations, the partial derivatives of $\hat{\boldsymbol{e}}_N$ and $\hat{\boldsymbol{n}}_N$ may be summarized as

$$\frac{\partial \hat{\boldsymbol{e}}_N}{\partial \mathbf{r}_N} = \left(I - \frac{1}{|Z_\times \mathbf{r}_N|^2}(Z_\times \mathbf{r}_N)(Z_\times \mathbf{r}_N)^T\right)\frac{Z_\times}{|Z_\times \mathbf{r}|} \tag{48}$$

$$\frac{\partial \hat{\boldsymbol{n}}_N}{\partial \mathbf{r}_N} = \boldsymbol{N}_1 + \boldsymbol{N}_2\boldsymbol{N}_3 \tag{49}$$

where

$$\boldsymbol{N}_1 = \frac{1}{|\mathbf{r}_N||Z_\times \mathbf{r}_N|}[2\hat{z}_G\mathbf{r}_N^T - \mathbf{r}_N\hat{z}_G^T - (\mathbf{r}_N^T\hat{z}_G)I] \tag{50}$$

$$\boldsymbol{N}_2 = \frac{-1}{|\mathbf{r}_N|^2|Z_\times \mathbf{r}_N|^2}[\hat{z}_G(\mathbf{r}_N^T\mathbf{r}_N) - \mathbf{r}_N(\mathbf{r}_N^T\hat{z}_G)] \tag{51}$$

$$\boldsymbol{N}_3 = \left[\left(\frac{|Z_\times \mathbf{r}_N|}{|\mathbf{r}_N|}\right)\mathbf{r}_N^T + \frac{|\mathbf{r}_N|}{|Z_\times \mathbf{r}_N|}(Z_\times \mathbf{r}_N)^T Z_\times\right] \tag{52}$$

Equations (48) and (49) may subsequently be substituted into Eq. (43). The partial derivatives in Eqs. (43) and (44) are then employed in constructing the variational constraint equation presented by Marchand et al. [3]. The resulting variational equation is added to the underdetermined linear system of equations previously identified in Eq. (25), which is subsequently solved during the level-II process.

C. Longitude

The longitude constraint defined by Eqs. (26) and (33) reveals an explicit dependence on the position vector and time associated with the Nth patch state. Thus, the partial derivatives of the longitude with respect to position and time are required before the variational constraint equation is evaluated. Once again, multiple formulations are available. Here, the longitude angle is constrained directly. Alternatively, the sine or cosine or some other function of the longitude may be constrained instead. The selected formulation affects the explicit partials. As such, only one formulation is presented here, though multiple formulations have been tested to date.

It is important to note that angular constraints can introduce convergence difficulties if the $2n\pi$ ambiguity is not properly handled during the convergence process. It is wise, in general, to choose formulations that avoid or adequately handle scenarios involving

singularities or quadrant ambiguities. With that in mind, consider the longitude constraint formulation. For notational simplicity, let

$$r_{k_{xy}} = \sqrt{(\mathbf{r}_N^T \hat{\mathbf{x}}_N)^2 + (\mathbf{r}_N^T \hat{\mathbf{y}}_N)^2} \tag{53}$$

$$r_{k_x} = \mathbf{r}_N^T \hat{\mathbf{x}}_N \tag{54}$$

$$r_{k_y} = \mathbf{r}_N^T \hat{\mathbf{y}}_N \tag{55}$$

Then from Eq. (33) and using Eq. (42), the constraint partials are determined as

$$\frac{\partial \lambda_N}{\partial \mathbf{r}_N} = \frac{1}{r_{k_{xy}}^2}(-r_{k_y}\hat{\mathbf{x}}_N^T + r_{k_x}\hat{\mathbf{y}}_N^T) \tag{56}$$

$$\frac{\partial \lambda_N}{\partial t_N} = -\omega_e \tag{57}$$

D. Total Mission Cost Constraint

A two-level corrector traditionally seeks to drive the interior velocity discontinuities to zero [1–3,9]. However, a series of nonzero maneuvers are naturally required to achieve the entry constraints in a nonnatural arc. In a constrained two-level corrector, the analyst can specify constraints on individual maneuvers [2,3,9]. A maneuver can be individually constrained as follows:

$$|\mathbf{v}_k^+ - \mathbf{v}_k^-| - \Delta v_{\text{des}} \le 0 \tag{58}$$

where

$$|\mathbf{v}_k^+ - \mathbf{v}_k^-| = |\Delta \mathbf{v}_k| = \Delta v_k = \sqrt{\Delta \mathbf{v}_k^T \Delta \mathbf{v}_k} \tag{59}$$

Note that the constraint in Eq. (58) depends explicitly on both the incoming and outgoing velocity vectors associated with the kth patch state; no dependence on position or time is present. Thus, the partial derivatives of the constraint relative to the velocity before and after the discontinuity at patch point k are simply given by

$$\frac{\partial \Delta v_k}{\partial \mathbf{v}_k^+} = \frac{1}{2}(\Delta \mathbf{v}_k^T \Delta \mathbf{v}_k)^{-\frac{1}{2}}(2\Delta \mathbf{v}_k^T)\frac{\partial \Delta \mathbf{v}_k}{\partial \mathbf{v}_k^+} = \frac{\Delta \mathbf{v}_k^T}{|\Delta \mathbf{v}_k|} \tag{60}$$

$$\frac{\partial \Delta v_k}{\partial \mathbf{v}_k^-} = \frac{1}{2}(\Delta \mathbf{v}_k^T \Delta \mathbf{v}_k)^{-\frac{1}{2}}(2\Delta \mathbf{v}_k^T)\frac{\partial \Delta \mathbf{v}_k}{\partial \mathbf{v}_k^-} = -\frac{\Delta \mathbf{v}_k^T}{|\Delta \mathbf{v}_k|} \tag{61}$$

If an equality constraint is applicable, the constraint is active at all times. If an inequality constraint is desired, however, the user need only remove the appropriate rows of the state relationship matrix before performing the level-II correction. If at a later point the constraint becomes active again, then the rows associated with the above partial derivatives can be introduced back into the state relationship matrix. Thus, the dimension of the state relationship matrix may fluctuate throughout the two-level process as various constraints become active and inactive.

In the case of Orion's three-TEI-maneuver strategy, constraining the magnitude of individual maneuvers is not as advantageous as it is to constrain the sum of the maneuvers. The goal of this particular approach is to ensure that the entry constraints are met while the total propulsive cost does not exceed the available Δv. This approach gives the two-level process much needed flexibility to place the maneuvers wherever the dynamics dictate it is most advantageous, as long as the sum does not exceed the allowable budget. In this case, the constraint is of the form

$$\Delta v_{\text{TEI}_1} + \Delta v_{\text{TEI}_2} + \Delta v_{\text{TEI}_3} - \Delta v_{\text{des}} \le 0 \tag{62}$$

Note that the variation of the total Δv may be expressed as the sum of the variations:

$$\delta \sum_{j=1}^{3} \Delta v_{\text{TEI}_j} = \sum_{j=1}^{3} \delta \Delta v_{\text{TEI}_j} \tag{63}$$

Since each maneuver is associated with a different vector variational equation, summing these variational equations yields an expression for the sum of the variations in Eq. (63). The resulting expression may subsequently be used to constrain the sum of all maneuvers without constraining individual maneuvers. Since each patch state contributes 12 control variables (three positions and three time elements), the total cost constraint for a three-burn TEI sequence can depend on up to 36 control variables, a far more desirable and successful approach than that in Eq. (58). This particular constraint is useful in extending the two-level corrector into an autonomous algorithm, because it is less constrained and subsequently less susceptible to the quality of the initial guess.

IV. Results

A. Example 1: Targeting Altitude and Flight-Path Angle Over the Lunar Cycle

The two-level targeter is applied here for various startup arcs over the lunar cycle, beginning on 1 February 2024 at 00:00:00 UTC. Tables 4–13 give the initial patch-point sets used for this example. The target entry constraints are 121.9 km altitude and −5.86° flight-path angle. Table 14 shows the total Δv cost associated with the startup arc before the corrections process is initiated as well as the initial error in each constraint for selected days over the lunar cycle. These initial guesses are employed in a full two-level targeter, both with and without a total Δv constraint. Figure 5 presents a comparison between the total Δv associated with the initial guess and the total Δv identified by the two-level targeter with and without the total cost constraint. The initial guess results correspond to the diamond-shaped markers that exhibit the smallest costs: between 1.4 and 1.5 km/s over the lunar cycle.

For the targeter solutions, samples were selected at three- to four-day intervals along the lunar cycle. The circles correspond to the targeter solution without the total cost constraint. The crosses are associated with the targeter solutions that are subject to a total cost constraint of 1.5 km/s. Clearly, the cost constraint has a significant impact on the solutions identified. If left unconstrained, the total Δv would quickly exceed the specified limits as the targeter searches the neighborhood of the initial guess for a feasible solution that meets the entry constraints. Constraining individual maneuvers is not consistently successful and can lead to nonconvergence. It is only by constraining the sum of the maneuvers instead that one is able to identify feasible solutions, as illustrated in Fig. 5, over the entire lunar cycle for the particular entry configuration. The individual TEI maneuvers for both the cost unconstrained and constrained two-level targeters are listed in Tables 15 and 16, respectively.

Of course, it is not reasonable to expect a linear targeting process to converge on a solution when the initial guess exhibits characteristics that are excessively dissimilar to those along the desired path. Currently, the block-set approach is preferred as the source of startup arcs for onboard determination [4,5]. However, two-body approximations [6] may be feasible if the associated discontinuities at the interface point are not excessive and if the methods used to bridge those discontinuities do not themselves require an unreasonable number of iterations. *Unreasonable* is defined in the present investigation as anything in excess of 20 iterations.

The most desirable feature of an onboard targeting algorithm for applications to Orion is one that is both numerically robust and minimizes computational overhead. Robustness is achieved by incorporating checks and balances that ensure both fast and monotonic convergence of the process. During the course of acquiring the trajectories for the present example it is observed that the largest source of overhead in a level-II process is the intermediate iterative level-I processes performed over each segment. Thus, identifying ways of speeding up and enhancing the convergence

Table 4 Patch points for lunar-cycle example, day 1 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.0800	1811.7587	195.0894	−235.6049	0.2603	−0.6192	1.4890
2	0.0939	1131.3087	−515.7179	1352.8625	−1.2847	−0.4509	0.9025
3	0.3186	−14025.7177	1927.5216	−6280.2732	−0.3177	0.1572	−0.4112
4	0.9791	−17191.7336	7312.5906	−19842.2331	0.1021	0.0379	−0.0924
5	0.9930	−17066.2790	7356.7247	−19949.6089	0.1069	0.0356	−0.0866
6	1.2046	−9351.0629	1556.4313	−14038.6710	0.4850	−0.3357	0.4116
7	1.4023	976.2354	−3258.8246	−2346.6280	0.6946	0.1290	1.3460
8	1.4161	1731.7989	−2909.4684	−624.9496	0.5363	0.4787	1.5064
9	1.8745	−7631.4693	35017.6465	30878.2008	−0.2610	0.8707	0.6288
10	4.3982	−68176.3260	211519.7031	63882.4925	−0.0228	0.8257	0.3968
11	5.6600	−6492.4424	331187.5816	178399.6460	−3.4414	−1.6887	10.5794

Table 5 Patch points for lunar-cycle example, day 3 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.0843	1808.3660	−42.3341	322.5813	−0.2783	−0.6419	1.4761
2	0.0982	599.1122	−652.7208	1609.6766	−1.5422	−0.2770	0.4617
3	0.3230	−11128.2055	3698.8282	−10101.5831	−0.1561	0.1947	−0.4808
4	0.9834	−9058.5751	9244.4824	−23530.4062	0.1395	0.0193	−0.0459
5	0.9973	−8889.6073	9265.8232	−23581.1814	0.1421	0.0163	−0.0387
6	1.2098	−2797.5015	3375.4149	−15598.5634	0.3588	−0.3526	0.5462
7	1.4084	2404.2421	−2174.2885	66.8395	−0.3267	0.2024	1.6804
8	1.4223	1762.4891	−1701.3856	1999.7540	−0.7280	0.5758	1.4787
9	1.8784	−31505.2873	26332.2563	28434.0208	−0.7656	0.6567	0.5435
10	4.3995	−173760.0319	158396.9655	36482.3868	−0.4494	0.6939	0.3268
11	5.6600	−181748.3946	277178.0908	154541.5980	−3.2206	−4.3371	9.7804

Table 6 Patch points for lunar-cycle example, day 6 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.0861	1748.1314	−141.2503	547.8324	−0.4968	−0.6307	1.4226
2	0.1000	354.8423	−689.4176	1665.7671	−1.6008	−0.1946	0.2605
3	0.3248	−9533.4536	4333.1892	−11380.9432	−0.0816	0.2049	−0.4937
4	0.9852	−5183.2054	9853.0910	−24131.4075	0.1532	0.0123	−0.0189
5	0.9991	−4998.3517	9865.8938	−24149.4498	0.1549	0.0090	−0.0112
6	1.2117	713.1056	5701.0557	−14614.7564	0.3160	−0.2733	0.6350
7	1.4104	2152.0911	−952.3354	2005.8320	−1.2197	−0.3889	1.2195
8	1.4243	504.8435	−1299.1310	3200.0587	−1.4641	−0.1912	0.7644
9	1.8802	−47750.6101	−1409.9080	12333.5821	−1.0937	0.0215	0.1620
10	4.4001	−245843.4912	−42627.8549	38221.0258	−0.8456	0.1536	0.1856
11	5.6600	−349455.9999	77757.5019	52025.1962	−1.5528	4.0371	−10.2007

of the level-I process is the key to further improving the overall computational efficiency of the two-level targeter. In spite of that, a two-level targeter is by nature more computationally efficient than an optimizer, because it only seeks feasible solutions in the neighborhood of the startup arc, rather than optimal solutions. Thus, although improvements in speed are always sought, the native speed of the algorithm is already adequate for onboard determination.

B. Example 2: Targeting Altitude, Latitude, Longitude, and Flight-Path Angle

One way to assess the success of the two-level targeting process for onboard determination is to compare its performance with that of an optimizer. To that end, a sequential quadratic programming algorithm [13–15] is selected here for comparison purposes. In this example, a three-burn TEI transfer is presented, subject to altitude,

Table 7 Patch points for lunar-cycle example, day 10 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.0926	1264.9187	−464.7081	1249.0292	−1.1822	−0.4929	1.0138
2	0.1065	−553.5780	−709.9509	1601.7088	−1.5556	0.1237	−0.4828
3	0.3313	−2530.9056	5861.9835	−14085.8017	0.1842	0.2073	−0.4637
4	0.9917	8028.4844	10212.7825	−23136.9288	0.1501	−0.0181	0.0532
5	1.0056	8207.1566	10189.0496	−23068.6637	0.1477	−0.0215	0.0606
6	1.2323	7342.8158	8139.5217	−11552.4736	−0.0959	−0.1666	0.6902
7	1.4451	31.4721	−330.8582	3115.5105	−1.2740	−1.1851	0.0165
8	1.4590	−1450.9126	−1672.7051	2814.0091	−1.1643	−1.0260	−0.4780
9	3.1418	−56026.8814	−127020.3316	−34394.5031	−0.4214	−0.8230	−0.2566
10	4.4009	−112061.4976	−204518.5425	−59686.8804	−0.6789	−0.5739	−0.2113
11	5.6600	−275896.0981	−231377.9901	−116889.2677	−1.0276	−0.2496	−11.2250

Table 8 Patch points for lunar-cycle example, day 13 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.1101	−1008.9920	−645.1486	1393.4239	−1.3629	0.2908	−0.8523
2	0.1240	−1829.5781	−25.0520	−166.6472	0.1275	0.6427	−1.4964
3	0.3488	14295.4678	2950.5652	−5172.8982	0.5397	−0.0011	0.0702
4	1.0092	27686.4726	610.6210	2390.6729	0.0255	−0.0530	0.1434
5	1.0231	27712.7559	547.0105	2562.4190	0.0183	−0.0530	0.1428
6	1.2501	17675.3814	3587.5500	3941.6792	−0.6108	0.1426	0.0529
7	1.4633	2749.0400	4655.2567	3513.0918	−1.1333	−0.2042	−0.2348
8	1.4772	1354.5350	4340.4663	3179.1459	−1.1886	−0.3284	−0.3271
9	1.9001	−798.5047	−31101.8060	−21962.1362	0.1394	−0.8250	−0.5670
10	3.1534	33319.8743	−131620.3865	−37122.6207	0.1975	−0.8889	−0.3178
11	4.4067	39648.2384	−226304.1089	−74508.5354	−0.1446	−0.8816	−0.4026
12	5.6600	−49527.5585	−345067.8043	−184830.5590	2.9658	0.7985	−10.8586

Table 9 Patch points for lunar-cycle example, day 16 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.1251	−1811.5619	34.6015	−304.7882	0.2611	0.6423	−1.4791
2	0.1389	−617.5457	649.3123	−1604.0504	1.5365	0.2832	−0.4769
3	0.3637	11241.1289	−3645.1334	9993.3322	0.1614	−0.1936	0.4793
4	1.0241	9277.4425	−9157.9284	23429.3105	−0.1404	−0.0189	0.0466
5	1.0380	9107.3016	−9178.7702	23480.8095	−0.1432	−0.0159	0.0393
6	1.2563	2910.7369	−3226.4340	15660.7124	−0.3572	0.3474	−0.5287
7	1.4607	−2399.1281	2229.3488	−257.6442	0.3423	−0.2594	−1.6510
8	1.4746	−1753.2367	1697.0191	−2141.7328	0.7176	−0.6144	−1.4330
9	1.9172	30557.8035	−26702.9250	−28822.8324	0.7709	−0.6892	−0.5735
10	3.1648	109867.2940	−98028.9783	−14839.0406	0.7427	−0.6315	−0.1943
11	4.4124	178875.1209	−173032.7415	−43878.0330	0.5081	−0.8075	−0.3910
12	5.6600	202052.8349	−306812.7946	−171394.3043	2.8040	4.7618	−9.6949

Table 10 Patch points for lunar-cycle example, day 19 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.1315	−1458.6933	371.4254	−1053.6583	0.9901	0.5518	−1.1762
2	0.1453	270.8822	722.8898	−1667.3461	1.6137	−0.0227	0.2523
3	0.3701	4896.0745	−5518.8358	13593.6064	−0.1024	−0.2120	0.4869
4	1.0305	−3861.9829	−10396.1004	24210.9196	−0.1519	0.0070	−0.0277
5	1.0444	−4043.5668	−10385.6517	24173.0226	−0.1507	0.0104	−0.0354
6	1.2483	−5790.2186	−5342.4929	13529.3366	−0.0641	0.3302	−0.7104
7	1.4383	−1808.0871	1245.7800	−2202.9813	1.4711	0.2359	−0.9621
8	1.4522	95.5490	1378.6353	−3067.9724	1.6381	−0.0088	−0.4729
9	1.9296	49289.9464	−6908.5525	509.3774	1.0412	−0.2032	0.1232
10	3.1731	146239.2488	8197.9877	−44201.0597	0.9807	−0.0034	−0.1890
11	4.4165	249252.0901	−6919.0600	−67598.9686	0.9514	−0.3469	−0.2620
12	5.6600	374034.9476	−143494.1636	−87995.2489	1.9669	−4.0490	10.1518

latitude, longitude, and flight-path-angle constraints. The initial set of patch points used is given in Table 17. The entry target values are 1) geodetic altitude of 121.92 km, 2) latitude of −19.2041 deg, 3) longitude of −134.5456 deg, and 4) geocentric flight-path angle of −6.03 deg

Results for both the optimization process and the targeter are given in Table 18. The optimal transfer identified, based on the initial guess, requires 1.0992 km/s of impulsive ΔV. The targeting algorithm identifies a solution that meets the same set of constraints at a total cost of 1.2700 km/s, per the imposed constraint. Figure 6 illustrates a

Table 11 Patch points for lunar-cycle example, day 22 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.1324	−1372.7642	416.1739	−1148.1596	1.0829	0.5261	−1.1041
2	0.1463	403.7304	719.1693	−1641.8973	1.5916	−0.0699	0.3607
3	0.3711	3807.9695	−5699.6526	13865.8204	−0.1409	−0.2108	0.4776
4	1.0315	−5804.0102	−10410.4652	23838.0931	−0.1513	0.0104	−0.0375
5	1.0454	−5984.4827	−10395.8924	23788.6012	−0.1495	0.0138	−0.0450
6	1.2540	−6486.6708	−7362.3226	12369.9484	0.0134	0.2225	−0.7339
7	1.4487	−1343.7399	47.4532	−2790.3132	1.3866	0.9574	−0.5431
8	1.4625	411.5817	1147.8427	−3098.3589	1.4817	0.8469	0.0188
9	1.9282	42774.2210	20266.0156	15307.5534	0.9216	0.3952	0.4504
10	4.4161	215687.5004	114836.2119	2609.5821	1.0142	0.2054	0.0438
11	5.6600	400920.3268	75545.6746	29789.7197	2.5928	−1.0520	10.9082

Table 12 Patch points for lunar-cycle example, day 25 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.1365	−930.3655	578.3179	−1475.1075	1.4064	0.3864	−0.7356
2	0.1504	936.0004	659.8702	−1436.8493	1.4034	−0.2636	0.7932
3	0.3752	−992.6047	−6084.1095	14185.9253	−0.2949	−0.1904	0.4114
4	1.0356	−13691.6517	−9564.2791	20920.7058	−0.1387	0.0272	−0.0764
5	1.0495	−13855.6951	−9529.8543	20825.0577	−0.1347	0.0302	−0.0830
6	1.2606	−9545.5014	−8153.0693	10076.5928	0.3025	0.1300	−0.6710
7	1.4579	−227.9130	−1337.3101	−2929.2129	1.1238	1.2728	−0.3223
8	1.4718	1094.9830	279.8446	−2978.7441	1.0397	1.3733	0.2450
9	1.9312	22181.3547	37064.2744	24920.0390	0.4456	0.7988	0.6692
10	4.4171	110471.2662	200681.6722	54605.5021	0.6789	0.6752	0.3105
11	5.6600	270356.3564	262953.8859	134978.4749	−0.3476	0.5289	11.2915

Table 13 Patch points for lunar-cycle example, day 28 (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.0785	1766.1394	273.7999	−426.4110	0.4436	−0.5952	1.4551
2	0.0924	1289.7995	−454.0580	1227.3176	−1.1607	−0.5006	1.0346
3	0.3171	−14683.0470	1252.2944	−4770.0449	−0.3663	0.1398	−0.3761
4	0.9776	−19575.8699	6424.7692	−18002.0623	0.0870	0.0430	−0.1052
5	0.9915	−19468.2004	6475.1207	−18125.1993	0.0924	0.0409	−0.1000
6	1.2031	−11110.1643	985.3447	−13030.7577	0.5286	−0.3146	0.3576
7	1.4009	418.9199	−3437.3750	−2815.4530	0.8540	0.1052	1.1791
8	1.4148	1405.0032	−3146.6234	−1289.1986	0.7665	0.4038	1.3594
9	1.8730	−4139.8111	35103.6917	29366.2119	−0.1933	0.8735	0.6036
10	4.3977	−45938.1764	218099.6459	67455.5807	0.0647	0.8437	0.4108
11	5.6600	27805.4616	337384.0539	182113.0290	−3.3113	−1.3542	10.6807

Table 14 Initial constraint errors over lunar cycle

Cycle day	Initial guess Δv	Alt. error, km	FPA error, deg
1	1.4617	0.0195	5.7828
3	1.4441	0.0206	5.8093
6	1.4282	0.0207	5.8574
10	1.4528	0.0183	5.8593
13	1.4958	0.0165	5.8301
16	1.4524	0.0157	5.8212
19	1.4213	0.0156	5.8598
22	1.4314	0.0159	5.8589
25	1.4583	0.0168	5.8598
28	1.4644	0.0187	5.7854

superposition of the resulting optimal and targeter solutions in the moon-centered inertial (MCI) frame.

The two-level targeter and optimization algorithms are executed on the same computing platform. In the present example, the optimal solution identified by the optimization process is acquired in 42.8 min. In contrast, the two-level targeter solution requires approximately 3.3 min: a 92% improvement. Of course, the optimizer's performance can be influenced through internal variable rescaling. However, that in itself is a disadvantage of the process for autonomous onboard determination, since the two-level targeter does not require any such variable scaling or tuning of scaling parameters.

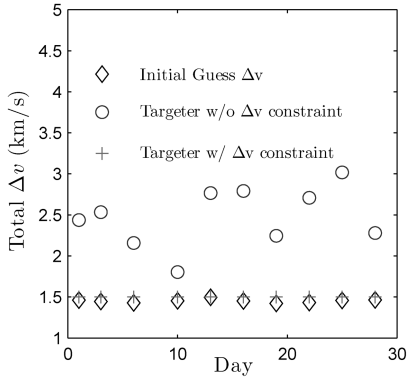

Fig. 5 Δv values over the lunar cycle.

Table 15 Δv values for unconstrained two-level targeting algorithm (km/s)

Cycle day	TEI-1, km/s	TEI-2, km/s	TEI-3, km/s	Total, km/s	Iterations
1	0.6028	0.7720	1.0618	2.4366	6
3	0.6041	0.7431	1.1866	2.5338	7
6	0.6018	0.4876	1.0694	2.1588	6
10	0.6022	0.5432	0.6592	1.8046	6
13	1.1483	0.6549	0.9635	2.7667	12
16	0.6039	0.6008	1.5882	2.7929	10
19	0.6015	0.4940	1.1499	2.2454	6
22	0.6014	0.4463	1.6605	2.7082	11
25	0.6021	0.5002	1.9166	3.0189	7
28	0.6025	0.7408	0.9373	2.2806	8

Table 16 Δv values for constrained two-level targeting algorithm (km/s)

Cycle day	TEI-1, km/s	TEI-2, km/s	TEI-3, km/s	Total, km/s	Iterations
1	0.6008	0.4658	0.4334	1.5000	7
3	0.6751	0.4364	0.3885	1.5000	11
6	0.6074	0.5072	0.3854	1.5000	14
10	0.6019	0.5638	0.3343	1.5000	7
13	0.5993	0.3635	0.5372	1.5000	11
16	0.6498	0.3360	0.5142	1.5000	13
19	0.5997	0.5684	0.3319	1.5000	16
22	0.7137	0.3784	0.4079	1.5000	24
25	0.6010	0.6059	0.2931	1.5000	11
28	0.5980	0.4819	0.4200	1.5000	7

Table 17 Patch points for optimization comparison example I (ECI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.0000	374424.0544	109403.2294	10334.5847	−0.4059	1.6939	1.7656
2	0.0978	372758.8348	117714.1237	14850.5366	1.1654	1.2035	0.8903
3	0.1115	374190.4086	118649.1328	15046.7190	1.3855	0.1351	−0.7889
4	0.7163	339456.3808	156186.1589	21607.7668	−0.5598	0.9526	0.5584
5	0.7448	338107.0137	158546.0446	23012.7546	−0.5064	1.0653	0.4862
6	1.0335	330759.9496	186065.0798	38924.5933	0.1188	1.0787	0.8472
7	1.1007	333536.6231	191196.5008	44155.1246	1.0605	0.3068	0.8138
8	1.1138	334822.1776	191091.5913	44823.3565	1.3344	−0.7664	0.2778
9	1.2445	333009.0784	184593.9650	41018.5355	−0.4244	−0.3317	−0.3013
10	3.8058	146319.5441	104098.9551	−10122.3614	−1.4378	−0.7164	−0.2172
11	4.6138	−5895.7461	−1733.9958	−2117.5885	0.5824	−7.4621	8.0510

Table 18 Targeting vs optimal results (example I)

	TEI-1	TEI-2	TEI-3	Total Δv, km/s	Iterations	Comp. time, s
Targeter	0.5680	0.1564	0.5457	1.2700	11	198.9
Optimizer	0.5711	0.1554	0.3727	1.0992	48	2567.6

C. Example 3: Targeting Altitude, Latitude, Longitude, Flight-Path Azimuth and Flight-Path Angle

In this example, all five entry constraints are targeted: altitude, latitude, longitude, flight-path angle, and flight-path azimuth. The first four entry constraints are the same as in the previous example, and the target flight-path azimuth value is 13.9960 deg. The initial patch-point file is listed in Table 19. Results are listed in Table 20. The total cost constraint imposed on the targeter solution is 1.35 km/s. With this constraint, the targeter requires 15 iterations and 524.5 s of computation time. The total budget available for the return phase is 1.5 km/s. Thus, 1.35 km/s for the primary maneuvers in these examples is still a reasonable cost.

In the interest of identifying similar solutions, the terminal entry time is constrained to match the entry time previously identified by the two-level targeter. Thus, some disparity is expected between the two trajectories, since the optimizer is constrained to a fixed final time and the targeter is allowed a free final time. The optimal total cost is over 0.35 km/s lower than the targeter solution in this case. The optimizer in this example, while still using a sequential quadratic programming method, is now implemented in FORTRAN. Note that, in general, compiled languages offer improved executable performance over programs that employ interpreted languages. Thus, a FORTRAN implementation of an optimizer will generally execute faster than the same optimizer executed within the MATLAB interpreter. This is an important point to remember when comparing targeter results identified in the MATLAB interpreter against optimizer results obtained from a FORTRAN compiled executable. Furthermore, though the homogeneity of the implementation is not preserved in this comparison, the example demonstrates the computational benefits of the two-level process over an optimization algorithm. Here, the optimizer execution requires 22.2 min. In contrast, the two-level targeter only requires 8.7 min to identify a feasible solution. This is roughly a 61% improvement over the optimization process, despite the discrepancy in programming language efficiency.

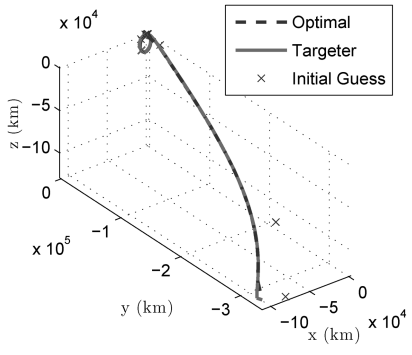

Fig. 6 Comparison 1: targeting vs optimal (MCI).

V. Optimal Trans-Earth Injection

The optimal trans-Earth injection phase has been extensively studied with various implementations of nonlinear programming algorithms. Though these numerical methods are not suitable for onboard determination, the insight gained from this analysis is useful in the development of a robust and numerically efficient autonomous targeting algorithm. Successfully targeting a specific entry state requires careful consideration of the timing and location of the maneuvers, particularly for constraints such as longitude and flight-path azimuth.

If one considers the trans-Earth injection phase purely in the context of a two-body problem, then TEI-1 raises apolune, TEI-2 executes a plane change, and TEI-3 places the spacecraft on a path to the desired entry interface point. Of course, this is all in an idealized two-body problem. In reality, TEI-3 alone cannot guarantee that all the desired constraints are met within the available fuel budget unless all perturbations are considered in the determination of the maneuver sequence. One advantage of the two-level targeter formulation presented here is that it allows all three maneuvers to be simultaneously adjusted to meet the entry constraints without exceeding the fuel budget. Since the targeter adjusts the spatial and temporal coordinates of the patch states associated with each maneuver, the targeter is also better able to exploit multibody effects in identifying a feasible solution that meets the cost and entry constraints.

Of course, a nonlinear programming algorithm can accomplish the same goals as a targeter while minimizing the cost. It is a well-known fact that the solutions identified by both a targeter and an optimizer are susceptible to the initial guess provided. Both methods search for solutions strictly in the neighborhood of the initial guess. The goal of the present analysis is to assess the impact of the initial guess on the optimal solution identified. To that end, consider Figs. 7 and 8. These figures illustrate the minimum Δv identified by the optimization process as a function of the initial argument of latitude and the right ascension of the ascending node associated with the initial low lunar orbit, respectively. It is evident from these figures that many, and sometimes significantly different, TEI maneuver combinations are identified by the optimizer, depending on the initial guess supplied. This indicates that the timing and geometry of the Earth return trajectory can significantly, and adversely, impact the total mission cost.

It is interesting to note, after properly converting the displayed units, that the first peak in Fig. 7, leading to a 2.7 km/s Δv, is roughly the same cost as the worst case identified by the two-level targeting process without a total cost constraint, illustrated in Fig. 5, of about 2.4 km/s. This supports the fact that both algorithms search for

Table 19 Patch points for optimization comparison example II (MCI)

k	Time, days	x, km	y, km	z, km	v_x, km/s	v_y, km/s	v_z, km/s
1	0.0000	1838.1000	0.0000	0.0000	0.0000	−1.6268	0.1443
2	0.0167	521.3234	−1755.7255	155.7605	−1.5661	−0.4614	0.0409
3	0.0431	−1812.8211	302.6071	−26.8462	0.2699	1.6044	−0.1423
4	0.0705	947.0087	3616.5019	−320.8405	1.3256	0.7315	−0.0649
5	1.0431	28715.9952	−4792.0104	426.1871	−0.0234	−0.1389	0.0123
6	1.9077	7803.1085	−8006.5727	−2089.8700	−0.7039	0.2221	0.0240
7	2.0431	−1807.2035	337.5984	−14.9796	0.3630	2.0918	0.7135
8	2.3991	23441.6464	24857.2847	−8256.7980	0.7339	0.5962	−0.2057
9	3.7541	101608.4866	85878.1689	−29520.4370	0.6309	0.4619	−0.1679
10	5.1930	188264.9646	134294.0899	−23362.9683	0.7167	0.2647	0.0919
11	6.8377	313360.6185	124543.2208	−3919.3448	1.3432	−0.7010	0.2886
12	7.2478	391669.3145	75008.2270	35040.0025	3.2994	0.6737	10.6875

Table 20 Targeting vs optimal results (example II)

	Total Δv, km/s	Iterations	Comp. time, s
Targeter	1.350	15	524.5
Optimizer	0.9799	N/A	1331.0

solutions in the same neighborhood: one searching for a feasible arc and the other searching for an optimal arc. It is thus evident, though not surprising, that an inadequate initial guess has adverse effects on both an optimization process and a linear targeting process. Figure 8 further indicates that given the proper alignment the optimal solutions do not typically exceed a combined mission cost of 1.2 km/s. This is an important fact that is useful in setting reasonable limits to the targeter's total cost constraint, provided that the startup arc is consistent with this observation. In general, insight into the optimal solution space can be of significant use in refining the block set [4,5] of startup arcs available to the targeting process onboard.

VI. Conclusions

In a loss-of-communications scenario, the Orion Crew Exploration Vehicle must be capable of autonomously targeting one of several possible entry sites on Earth. The present investigation focuses on one of the many elements of the autonomous targeting process: the targeting algorithm. An impulsive constrained two-level targeter is selected as the main algorithm for onboard precision entry targeting.

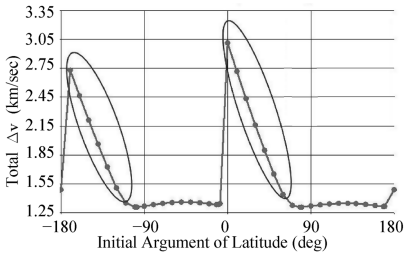

Fig. 7 TEI Δv (ft/s) vs argument of latitude, deg.

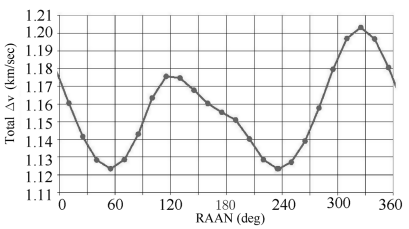

Fig. 8 Δv (ft/s) sensitivity to initial RAAN, deg.

At the entry interface point, the vehicle must achieve a prespecified set of constraints: altitude, latitude, flight-path angle, longitude, and azimuth. The first three of these constraints are formulated and validated in earlier studies. The present investigation first focuses on the development and validation of the longitude and azimuth constraints. Of course, in an onboard targeting scenario, it is imperative that the solutions identified by the targeting algorithm do not exceed the available fuel budget. Thus, a total cost constraint is formulated and validated in the context of a two-level targeter. The constraint formulations add unprecedented flexibility and robustness to the targeting process, particularly for problems susceptible to multibody effects such as the Orion trans-Earth injection phase. The robustness of the targeting process is assessed through numerical investigations over various departure and entry configurations.

Naturally, both an optimizer and a linear targeting process explore the same neighborhood of solutions, as supported by the results of this study. However, a two-level targeter seeks only feasible solutions, which significantly reduces the computational overhead in comparison with an optimizer. Aside from the reduced computational requirements, the combination of a two-level targeter with the total cost constraint developed here offers a generalized formulation that is both flexible and robust for onboard determination.

References

[1] Howell, K., and Pernicka, H., "Numerical Determination of Lissajous Trajectories in the Restricted Three-Body Problem," *Celestial Mechanics*, Vol. 41, Nos. 1–4, Mar. 1987, pp. 107–124. doi:10.1007/BF01238756

[2] Wilson, R., Barden, B., Howell, K., and Marchand, B., "Summer Launch Options for the Genesis Mission," *Advances in the Astronautical Sciences*, Vol. 109, 2001, Paper AAS 01-306.

[3] Marchand, B., Howell, K., and Wilson, R., "Improved Corrections Process for Constrained Trajectory Design in the n-Body Problem," *Journal of Spacecraft and Rockets*, Vol. 44, No. 4, 2007, pp. 884–897. doi:10.2514/1.27205

[4] Hyde, C. T., Foggat, C. E., and Weber, B. D., "Apollo Experience Report—Abort Planning," NASA, TR NASA-TN-D-6847, Houston, TX, June 1972.

[5] Yencharis, J. D., Wiley, R. F., Davis, R. S., Holmes, Q. A., and Zeiler, K. T., "Apollo Experience Report—Development of Guidance and Targeting Techniques for the Command Module and Launch Vehicle," NASA, TN-D-6848, Houston, TX, June 1972.

[6] Ocampo, C. A., and Saudemont, R. R., "Initial Trajectory Model for a Multi-Maneuver Moon to Earth Abort Sequence," AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society Paper 09-195, Savannah, GA, Feb. 2009.

[7] D'Amario, L. A., Byrnes, D. V., and Stanford, R. H., "A New Method for Optimizing Multiple-Flyby Trajectories," *Journal of Guidance and Control*, Vol. 4, No. 5, 1981, pp. 591–596. doi:10.2514/3.56115

[8] Corless, M., and Frazho, A., *Linear Systems and Control: An Operator Perspective*, CRC, Boca Raton, FL, 2003, pp. 313–314.

[9] Wilson, R., and Howell, K., "Trajectory Design in the Sun-Earth-Moon System Using Lunar Gravity Assists," *Journal of Spacecraft and Rockets*, Vol. 35, No. 2, 1998, pp. 191–198. doi:10.2514/2.3309

[10] Bate, R., Mueller, D., and White, J., *Fundamentals of Astrodynamics*, 1st ed., Dover, New York, 1971, pp. 122–131.
[11] Adamo, D., "Apollo 13 Trajectory Reconstruction Via State Transition Matrices," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 6, 2008, pp. 1772–1781. doi:10.2514/1.34977
[12] Vallado, D., *Fundamentals of Astrodynamics and Applications*, McGraw–Hill, New York, 1997, pp. 189–195.
[13] Fletcher, R., *Practical Methods of Optimization*, Wiley, New York, 1987.
[14] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, London, 1981.
[15] Powell, M. J. D., *Variable Metric Methods for Constrained Optimization*, Springer–Verlag, New York, 1983.