Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research 72 (2023) 5086–5101

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

Energy reference guidance for drag-modulated aerocapture

Samuel W. Albert [a,*], Ethan R. Burnett [a], Hanspeter Schaub [a], P. Daniel Burkhart [b] Alex Austin [b]

[a] *Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, 3775 Discovery Dr, Boulder, CO 80303, USA*
[b] *Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive Dr. M/S 198-228, Pasadena, CA 91109, USA*

Received 28 March 2023; received in revised form 6 August 2023; accepted 16 September 2023
Available online 22 September 2023

Abstract

Aerocapture is a method of achieving orbit insertion via a single pass through the atmosphere of the central body. Single-event jettison drag modulation is a simple way of achieving control during atmospheric flight by effecting a discrete change in the aerodynamics of the vehicle. A novel guidance algorithm, energy reference guidance, is developed and implemented for a reference scenario of a small satellite aerocapture demonstration at Earth, and is compared to the baseline numerical predictor–corrector solution. The new approach is shown to achieve equivalent apoapsis targeting performance as the baseline algorithm with significantly lower CPU demand during atmospheric flight, although more onboard memory is required in exchange. The relationship between targeting performance and required memory is quantitatively explored for the new algorithm; the selected configuration generates approximately 3.3 MB of data, which is expected to be well within the capability of relevant avionics systems.

Keywords: Aerocapture; Guidance; Small satellite missions; Numerical predictor–corrector; Flight mechanics

1. Introduction

Aerocapture is a technology that could enable shorter transit times and lower total expended mass for orbit insertion for a variety of interplanetary missions (Cruz, 1979; Hall et al., 2005; Spilker et al., 2019). To perform aerocapture, the spacecraft executes a single pass through the atmosphere of a planet or moon, dissipating enough energy to reach the desired target orbit upon exit from the atmosphere. During the subsequent pass through apoapsis the spacecraft performs a propulsive maneuver to raise periapsis out of the atmosphere, and performs other correction maneuvers as necessary. For missions to the ice giants Uranus and Neptune, aerocapture can potentially reduce cruise duration by 2–5 years while reducing mass for orbit insertion by some 40% (Dutta, 2021; Agrawal et al., 2014). Aerocapture also offers significant benefit to small satellites (SmallSats) launched via rideshare with a primary mission, enabling orbit insertion despite the lack of high-ΔV systems at SmallSat scale and reducing sensitivity to primary mission trajectory design (Austin, 2020; Austin et al., 2019). Although it has been proposed for a number of missions (Walberg et al., 1987; Powell, 2012; Cazaux et al., 2004), aerocapture has never been flown (Spilker et al., 2019).

Variability in the spacecraft state at atmospheric entry, atmospheric density, aerodynamic properties of the vehicle, and other day-of-flight dispersions require a spacecraft performing aerocapture to autonomously control its trajectory through the atmosphere. During this hypersonic flight phase, control is achieved by judiciously adjusting the aerodynamic forces acting on the vehicle, and control approaches thus fall into two main categories: lift modulation and drag modulation. Lift modulation

* Corresponding author.
E-mail address: samuel.albert@colorado.edu (S.W. Albert).

involves changing the attitude of the vehicle to reorient the lift vector, typically either by banking about a fixed trim angle (bank angle modulation) or by independently modulating angle of attack and side-slip angle (direct force control) (Vinh et al., 2000; Deshmukh et al., 2020). Note that direct force control also involves changes in the drag and side force components, but the primary control authority is obtained by modulating lift, and thus this technique is categorized with lift modulation for the purposes of this discussion. Lift modulation, particularly bank angle modulation, is well-studied in the literature (Rousseau et al., 2012; Lu et al., 2015; Deshmukh et al., 2020; Deshmukh, 2021), and has relevant flight heritage from guided hypersonic entry of blunt-body aeroshells including the Apollo (Moseley, 1969), Orion (Putnam et al., 2010), Mars Science Laboratory (Steltzner et al., 2014), and Mars 2020 (Nelessen et al., 2019) missions, all of which relied on some form of closed-loop lift modulation.

Recent work has studied drag modulation as a potentially simpler method of achieving control for aerocapture (Roelke, 2021; Putnam and Braun, 2014). Typically, a drag-modulated vehicle is assumed to be axisymmetric and to fly at zero angle of attack, thus generating no lift. The trajectory is influenced by adjusting the ratio of mass to effective drag area, termed ballistic coefficient; when this ratio is low, the vehicle rapidly dissipates energy through drag, and vice versa. This can take a variety of forms, including devices that achieve continuously-variable drag (Vinh et al., 1986), release of a trailing inflatable drag device (Rohrschneider and Braun, 2007), and jettison of one or more rigid drag skirts (Roelke et al., 2022). Single-event jettison, defined here as a single discrete change in ballistic coefficient caused by the jettison of a rigid drag skirt, is the control architecture that will be the focus of this work. This represents a limiting case, because after jettison the vehicle flies passively for the remainder of atmospheric flight and the vehicle lacks any out-of-plane control. However, for a sufficiently large change in ballistic coefficient, single-event jettison can achieve a total control authority comparable to lift modulation with heritage blunt-body aeroshells (Heidrich et al., 2020). Compared to lift modulation, single-event jettison drag modulation may be less complex because the vehicle can be passively spin-stabilized, rather than requiring a high-rate reaction control system that must operate during atmospheric flight (Powell and Braun, 2012). Moreover, ballast masses are not required to create an offset center of gravity, as is typically the case for lift-modulated axisymmetric vehicles (Steltzner et al., 2014).

A limited number of guidance algorithms for single-event jettison drag-modulated aerocapture exist in the literature. The simplest solution in terms of computational expense is to trigger jettison when the instantaneous value of a measured state exceeds some threshold, such as a velocity trigger (Falcone et al., 2019). To reduce error caused by noisy measurements, the observed state can be filtered and jettison can be triggered based on some polynomial function of the state. For example, the algorithm implemented in Johnson and Lyons (2004) triggers jettison when the total integrated ΔV exceeds a polynomial function of the filtered instantaneous or maximum sensed acceleration. The deceleration curve fit algorithm used for Mars Pathfinder parachute deploy (Braun et al., 1999) and applied to drag-modulated aerocapture in Werner and Braun (2019) also triggers based on deceleration measurements. In this case, two measurements are taken a set time apart, and a pre-computed curve fit between the second deceleration measurement and time until jettison is consulted to set a jettison timer. All of these approaches require only minimal onboard computation and memory, but each is also shown to have poor performance when relevant uncertainties are applied. The predictive trigger approach applied in Gulick et al. (2003) is more computationally-intensive; in this case, the energy of the spacecraft at atmospheric exit is predicted by numerically propagating the equations of motion, and jettison is commanded when the predicted final energy is less than or equal to the desired final energy. Machine learning-based guidance schemes have been successfully developed for entry and aerobraking problems (Cheng et al., 2021; Wang and Elgohary, 2020; Shi and Wang, 2021; Falcone and Putnam, 2022), but have yet to be applied to single-event jettison drag-modulated aerocapture other than for the purpose of atmospheric estimation (Wagner et al., 2011; Amato et al., 2020).

While the algorithms summarized above share the benefit of relatively low onboard computational burden, the current state of the art guidance for drag-modulated aerocapture is the numerical predictor–corrector (NPC) approach (Putnam and Braun, 2014). This algorithm also predicts the final state by numerically propagating the equations of motion, then takes the additional step of making a correction to the jettison time. This two-step procedure is applied iteratively, such that the algorithm should converge to an optimal jettison time each guidance call. NPC has two key differences with the predictive trigger. First, because NPC solves for jettison time rather than directly commanding jettison, the release timing can operate at significantly higher resolution; this is under the assumption that a simple controller releases the drag skirt when the jettison time is reached, operating at a higher rate than the guidance algorithm itself. Second, NPC is significantly more computationally expensive than the predictive trigger because multiple numerical propagations may be required in each step. In summary, NPC guidance is significantly more accurate in the presence of uncertainties than the other algorithms discussed here (Putnam and Braun, 2014; Falcone et al., 2019; Werner and Braun, 2019), but is also much more computationally demanding. A more detailed description of the NPC algorithm is given in Section 3.

This work investigates a new guidance algorithm for single-event jettison drag-modulated aerocapture, with the goal of achieving the same level of accuracy as the

NPC but with significantly less computational demand. The reference mission for this study is an Earth flight test of aerocapture with a SmallSat using a rigid deployable drag skirt; that is, the drag skirt is stowed during launch and deployed during cruise, but does not change its shape during atmospheric flight. Assumptions regarding modeling of dynamics and uncertainties are discussed, and key physical parameters defined. The baseline NPC algorithm is described in detail, including a novel approach to the correction step that improves computational efficiency, and targeting results under relevant uncertainties are estimated. The proposed algorithm is also described, and compared directly with NPC. A parameter study is presented that gives insight into optimal tuning and tradeoffs between memory and performance for the proposed algorithm. Finally, results are discussed along with a number of avenues for potential future work.

2. Methodology

2.1. Reference mission

Researchers from the NASA Jet Propulsion Laboratory (JPL), NASA Ames, and the University of Colorado Boulder have been studying drag-modulated aerocapture for small satellites (Austin et al., 2019; Werner et al., 2017), including concepts for an Earth flight test of the technology (Werner and Braun, 2019). This idea is supported by the 2022 Strategic Framework[1] released by the NASA Space Technology Mission Directorate, which states that "an Earth-based aerocapture demonstration will reduce perceived risk and mature guidance and control methods" for aerocapture at other planetary destinations. Motivated by these developments, single-event jettison drag-modulated aerocapture at Earth by a SmallSat is the reference mission considered in this work. As summarized in Fig. 1, the spacecraft is launched into a geosynchronous transfer orbit, then performs a maneuver to lower periapsis into the atmosphere, achieving the desired entry state. Based on the JPL reference mission, the spacecraft targets an apoapsis of 5000 km and performs a maneuver at the next pass through apoapsis to raise periapsis to 200km. Autonomously raising periapsis during the first pass through apoapsis in order to achieve a near-term stable orbit is a significant component of successful aerocapture; however, specific consideration of the on-orbit maneuver guidance and control is beyond the scope of this study.

The drag skirt in this study is modeled as the Adaptable Deployable Entry and Placement Technology (ADEPT), an umbrella-like deployable structure for entry probes currently under development at NASA Ames (Cassell et al., 2018). During launch, ADEPT is in the retracted configuration, significantly reducing fairing volume required for the spacecraft and enabling stowage in the standard ESPA envelope (Cassell et al., 2018; Wegner et al., 2001). The drag skirt is fully deployed between separation from the launch vehicle and atmospheric entry, and remains rigidly deployed until it is jettisoned by the guidance algorithm.

The initial epoch for simulation of this mission is defined as 10 min before nominal atmospheric entry, which is the time of the final orbit determination (OD) update to the spacecraft from ground control. From this point onward, the navigated states are based on propagation with only IMU data. The nominal entry state, defined at the atmospheric interface altitude of 125km, has a planet-relative velocity u of 9.9km/s and flight-path angle γ of -4.6°, where flight-path angle is the angle between the planet-relative velocity vector $\boldsymbol{u}$ and the local horizontal plane. The nominal entry point is at a geocentric latitude ϕ of -7.4° and longitude θ of 14.8° with a heading of 118.9°, where heading angle ψ is defined as the angle between the horizontal projection of the velocity vector and a due-North vector in that same plane. These definitions are illustrated in Fig. 2, where the unit vector bases $\left\{\hat{n}_1, \hat{n}_2, \widehat{K}\right\}, \left\{\widehat{I}, \widehat{J}, \widehat{K}\right\}$, and $\{\hat{e}_1, \hat{e}_2, \hat{e}_3\}$ define inertial, planet-fixed, and position frames, respectively. The vector from the central body to the vehicle is denoted $\boldsymbol{r}$, and $\hat{\boldsymbol{r}} = \boldsymbol{r}/r$ is the associated unit vector.

2.2. Problem dynamics

2.2.1. Simulation

In this work, the performance of each guidance algorithm is quantified through testing in a high-fidelity simulation environment implemented in the Dynamics Simulator for Entry, Descent, and Surface Landing (DSENDS) software developed by the DARTS lab at NASA JPL (Cameron et al., 2016). The gravity model includes point-mass and spherical harmonics of degree and order 8 for the Earth, as well as point-mass gravity from the Moon and the Sun. Atmospheric density is modeled using the Earth Global Reference Atmospheric Model (Earth-GRAM) 2010 (Leslie and Justus, 2011), such that the value of density depends on 3D position and time.

The vehicle shape is a 60-degree sphere-cone both with and without the drag skirt, such that the drag skirt extends the conical section at the same angle. The aerodynamics model used in simulation includes drag and aerodynamic moments. No lift is modeled; the vehicle is axisymmetric and passively-stabilized, such that the axis of symmetry remains approximately aligned with the freestream velocity vector. Thus, while the simulation is 6 degree-of-freedom, oscillations in vehicle attitude are small and have only a minor effect on the vehicle trajectory.

2.2.2. Predictor model

Both guidance algorithms presented in this work rely on numerical propagation of the relevant equations of motion to predict trajectories onboard. These equations constitute a simplified version of the dynamics modeled in the full

[1] https://techport.nasa.gov/framework.

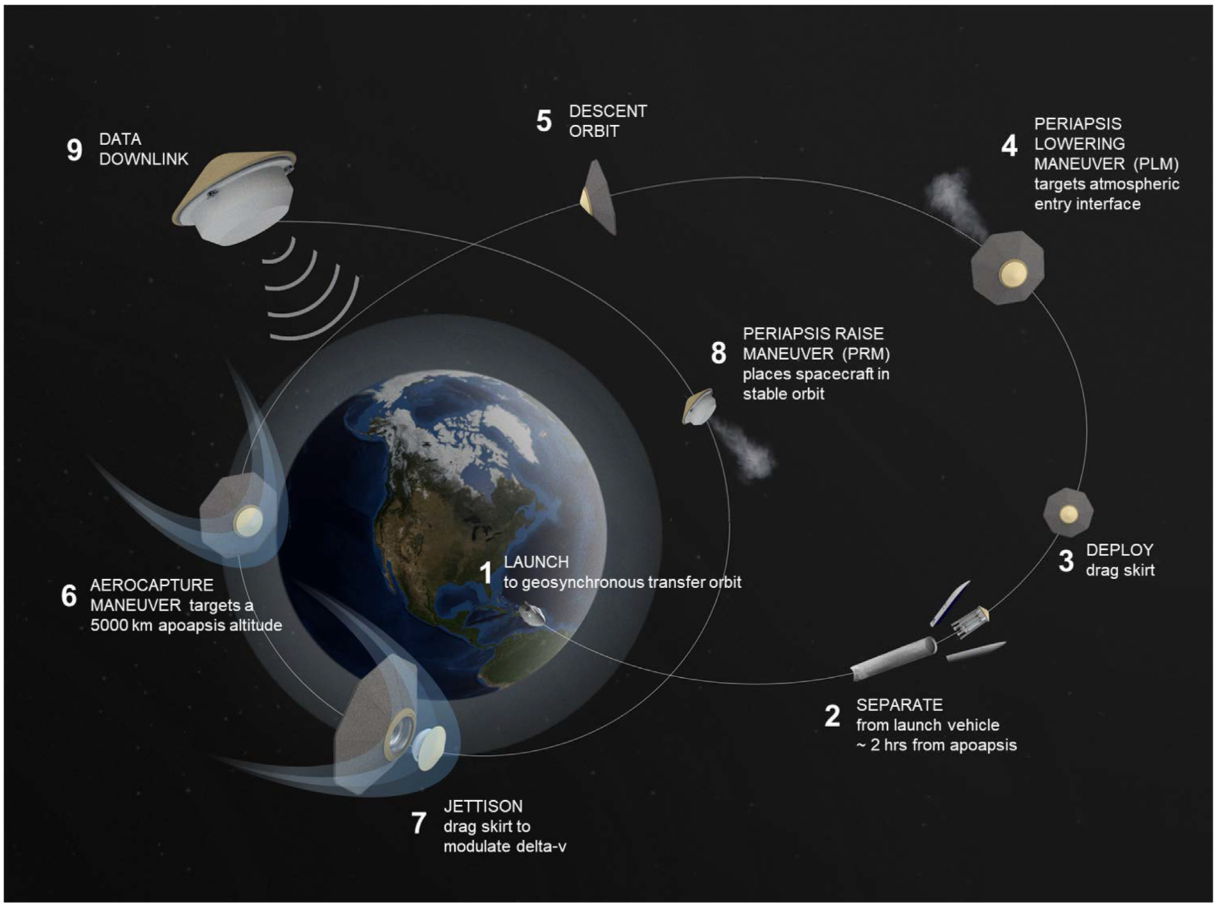
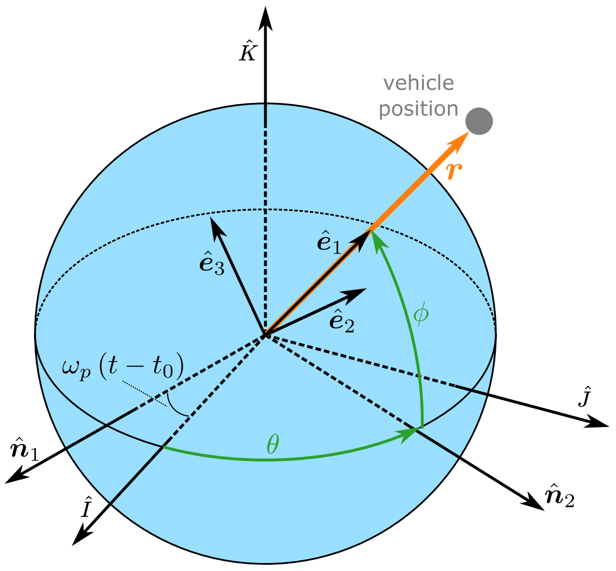
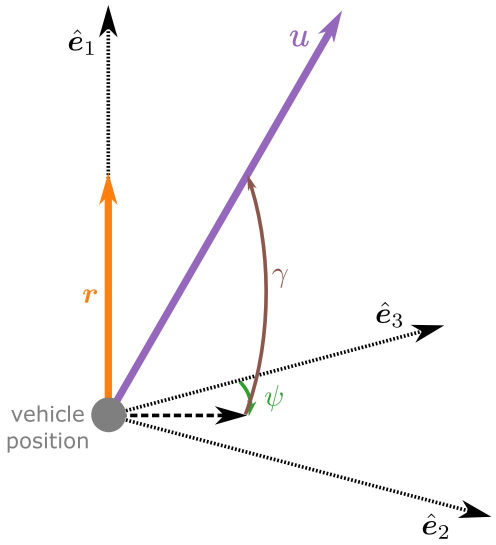

9 DATA DOWNLINK
5 DESCENT ORBIT
4 PERIAPSIS LOWERING MANEUVER (PLM) targets atmospheric entry interface
8 PERIAPSIS RAISE MANEUVER (PRM) places spacecraft in stable orbit
3 DEPLOY drag skirt
1 LAUNCH to geosynchronous transfer orbit
6 AEROCAPTURE MANEUVER targets a 5000 km apoapsis altitude
2 SEPARATE from launch vehicle ~ 2 hrs from apoapsis
7 JETTISON drag skirt to modulate delta-v

Fig. 1. Aerocapture earth flight test.



(a) Latitude ϕ, longitude θ, and position vector $\boldsymbol{r}$



(b) Flight-path angle γ, heading angle ψ, and flow velocity vector $\boldsymbol{u}$

Fig. 2. Frame definitions.

"truth" simulation. Specifically, the modeled forces include point-mass gravity, J2 oblateness, and drag, resulting in the following equation for inertial acceleration (Burnett and Schaub, 2022):

$$\begin{aligned}\ddot{\boldsymbol{r}} = &-\frac{\mu}{r^2}\hat{\boldsymbol{r}} \\ &-\frac{3\mu J_2 R^2}{2r^4}\left(\left(1-5\left(\hat{\boldsymbol{r}}\cdot\hat{\boldsymbol{K}}\right)^2\right)\hat{\boldsymbol{r}}+2\left(\hat{\boldsymbol{r}}\cdot\hat{\boldsymbol{K}}\right)\hat{\boldsymbol{K}}\right) \\ &-\frac{\rho u^2}{2\beta}\hat{\boldsymbol{u}}\end{aligned} \tag{1}$$

where $\boldsymbol{r}$ is the vector from the central body to the vehicle, μ is the gravitational parameter, J_2 is the oblateness coefficient, ρ is atmospheric density, R is the planetary equatorial radius, $\hat{\boldsymbol{K}}$ is the polar axis unit vector, and $\beta = m/(C_D A)$ is the ballistic coefficient of the vehicle. The quantities m, C_D, and A are the mass, drag coefficient, and reference area of the vehicle, respectively. The quantity $\boldsymbol{u}$ is the flow velocity, or the velocity of the spacecraft with respect to the planetary atmosphere, which is assumed to be rotating with the planet with angular velocity $\boldsymbol{\omega}_p$ between initial time t_0 and current time t,

$$\boldsymbol{u} = \dot{\boldsymbol{r}} - \boldsymbol{\omega}_p \times \boldsymbol{r}, \tag{2}$$

where $\dot{\boldsymbol{r}}$ is inertial velocity. The predictor models density by linearly interpolating from a table of density vs. altitude output by EarthGRAM that represents a nominal atmosphere profile. Note that the predictor thus assumes the same density is experienced in the descending and ascending portions of the aerocapture trajectory, other than as modified by the atmospheric scale factor as discussed later, whereas the DSENDS simulation incorporates dependence of density on latitude and longitude. A table of C_D vs. dynamic pressure is similarly used by the predictor to compute β. However, note that this latter step is likely higher-fidelity than necessary because C_D changes little in the relevant flight regime for this scenario; constant C_D would be a reasonable approximation. The values of μ, J_2, and R used in both the predictor and simulation are provided in Table 1. The average ballistic coefficient for each phase is also listed, where β_1 and β_2 are the values pre- and post-jettison, respectively. The predictor uses fourth-order Runge–Kutta integration to numerically propagate the equations of motion, with a fixed time step of 0.125 s.

Table 1
Nominal simulation parameters.

Parameter	Value
μ	3.9860×10^5 km^3/s^2
ω_p	7.2921×10^{-5} rad/s
J_2	0.0010826
R	6378.1 km
β_1	32 kg/m^2
β_2	137 kg/ m^2

2.3. Models of uncertainty

The variability of atmospheric density is modeled by EarthGRAM, which has a built-in Monte Carlo framework for generating realistic dispersions (Leslie and Justus, 2011). The vehicle aerodynamics are dispersed based on experience with blunt-body aeroshells Way et al. (2003), resulting in a standard deviation of about $\sigma = 0.015$ for C_D near peak dynamic pressure, where σ is standard deviation and the nominal value is 1.38. The entry state is dispersed according to a navigation assessment performed at JPL that was then scaled to match the project requirement of entry flight-path angle delivery error with a standard deviation value of $3\sigma = 0.2^\circ$ at the atmospheric interface altitude of 125 km. The time required for the drag skirt to fully separate from the capsule is assumed to be uniformly dispersed along a range from 0.05 s to 0.14 s. The vehicle mass and area are not dispersed, nor are gravitational parameters.

Importantly, the predictor does not operate on the true state of the spacecraft. Noisy measurements from an inertial measurement unit (IMU) are modeled and fed into a navigation filter, and the predictor operates on these filtered state estimates. The navigation filter uses the same dynamics model as the predictor, Eq. 1.

3. Numerical predictor–corrector guidance

NPC guidance is treated as the baseline solution in this work due to both its state-of-the-art targeting performance and its previous application as part of the JPL SmallSat aerocapture project (Putnam and Braun, 2014; Austin et al., 2019; Strauss et al., 2021). The implementation discussed here is similar to that presented in Putnam and Braun (2014), but with a more computationally-efficient correction method. The algorithm is summarized by Fig. 3 and outlined in detail in this section; performance results are given in Section 5.

IMU measurements are used to generate an estimate of sensed acceleration (or g-load), $\hat{g}$, and when this exceeds some threshold value g_t the guidance routine is initiated. In the subsequent step, nominal density at the navigation-estimated altitude is used with the navigation-estimated state to compute an estimate of the dynamic pressure:

$$q_{\text{est}} = \frac{1}{2}\rho_{\text{nom}}(r)u^2 \tag{3}$$

where the estimated dynamic pressure is used to obtain an estimate of the drag coefficient via interpolation of stored data of C_D vs. q. Next, the density is estimated from a rearranged expression for acceleration due to aerodynamic drag (which equals $\hat{g}$ since the vehicle is assumed to have no lift):

$$\rho_{\text{est}} = 2\frac{m_1\hat{g}}{A_1 C_{D,1} u^2} = 2\frac{\beta_1 \hat{g}}{u^2} \tag{4}$$

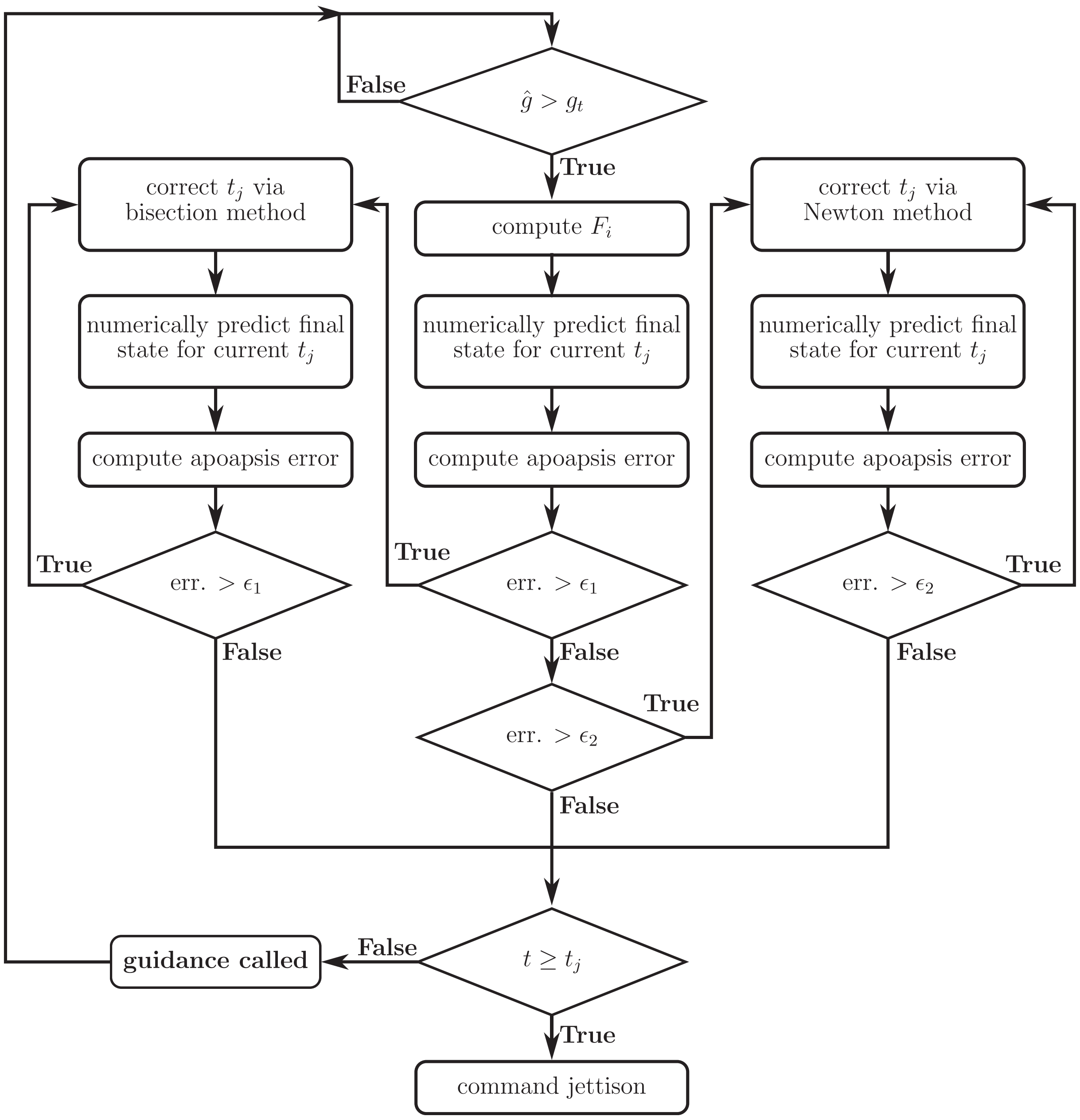


Fig. 3. NPC guidance diagram.

where $m_1, A_1, C_{D,1}$, and β_1 are the pre-jettison values of those variables. The density estimate is used to compute the *i*th density scale factor F_i:

$$F_i = \rho_{\text{est}}(t_i)/\rho_{\text{nom}}(r(t_i)) \tag{5}$$

This value is then filtered via a low-pass filter:

$$\overline{F}_i = (1-k)\overline{F}_{i-1} + kF_i \tag{6}$$

As the gain k is decreased, this filter will increasingly reject small disturbances. Sensible values of k depend on the frequency of density scale factor measurement updates. Alternatively, the density scale factor could also be filtered with a moving average filter, detailed below:

$$\overline{F}_n = \frac{1}{n}\sum_{i=1}^{n} F_i \tag{7}$$

where n is a memory parameter, and again the chosen value of n should be tuned based on the density scale factor update frequency. In this work, the low-pass filter is implemented with $k = 0.05$ for a guidance update rate of 8 Hz. The nominal density profile is then re-scaled by F_i for all subsequent numerical propagations within that guidance call, as follows:

$$\rho_{\text{pred}}(r) = \overline{F}_i \rho_{\text{nom}}(r) \tag{8}$$

This form of density re-scaling significantly improves targeting results compared to ignoring atmospheric estimation altogether (Perot and Rousseau, 2002), but is limited to linearly shifting the entire profile and thus fails to capture the more complex atmospheric perturbations that occur in reality. Other methods, such as exponentially correlating the scale factor, ensemble correlation filtering (Roelke et al., 2019), machine learning (Wagner et al., 2011; Amato et al., 2020), or modeling density as a Gaussian random field (Albert et al., 2021) may improve the atmospheric estimation component of NPC guidance.

Once the density scale factor is computed, the navigation-estimated state is numerically propagated until the altitude of the spacecraft either exceeds the atmospheric interface altitude or decreases below some minimum. This prediction uses the jettison time computed by the previous guidance call or, in the case of the first guidance call, a pre-defined initial guess, set to 700 s in this case. The radius of apoapsis is then computed from the final state using Keplerian relations, and error is computed as the difference between the predicted and desired apoapsis radii. In the case of an escape trajectory, apoapsis radius is poorly-defined and the error is set equal to positive infinity. In the case of an impact trajectory, in which the spacecraft reaches the surface instead of exiting the atmosphere, the Keplerian apoapsis is computed from the final state as normal; the value will badly undershoot the target and thus the guidance algorithm behaves as expected. As an aside, note that for certain, more extreme mission scenarios an edge case is possible in which the vehicle reaches the minimum altitude bound while still hyperbolic in terms of orbital energy, and care should be taken to correctly categorize these cases as undershoots, despite their hyperbolic Keplerian state.

The error magnitude is then compared against two tolerance values, ϵ_1 and ϵ_2, where $\epsilon_1 > \epsilon_2$. The purpose of the dual tolerances is to direct the algorithm to an appropriate root-finding subroutine for the correction step. If the error exceeds both tolerances, bisection method is selected; if the error is between the two tolerance values, Newton's method is selected; finally, if the error is below both tolerances, no updates to jettison time are required and the algorithm skips the correction step entirely. In this work, tolerances were defined as $\epsilon_1 = 500\ \text{km}$ and $\epsilon_2 = 25\ \text{km}$, selected based on a trial-and-error process in order to achieve a good balance between accuracy and speed. These tolerances would need adjustment for a significantly different apoapsis target or central body.

The bisection method subroutine begins with lower and upper bounds on the optimal jettison time, selected *a priori* without any dependence on the solution from the previous guidance call. These values should span the duration of the longest atmosphere pass that is expected based on dispersions and are strongly scenario-dependent. For this work, bounds of 600 and 900 s are selected, noting that $t = 0$ is defined as 10 min prior to atmospheric entry. The jettison time is then set equal to the midpoint of these bounds, and the predictor numerically propagates to the final state and computes an apoapsis error. If the magnitude of this error is below the tolerance ϵ_1, the algorithm exits the bisection subroutine with a converged solution. Otherwise, the bounds on jettison time are updated based on the sign of the error. In an overshoot case with positive error, the upper bound is set equal to the current value of the jettison time; in the undershoot case, the lower bound is similarly updated. The subroutine then repeats, using the updated midpoint as the new jettison time, and continues until either the error magnitude is below the tolerance ϵ_1 or a maximum number of iterations is reached. The subroutine also includes logic to recognize cases in which the jettison time converges against the original upper or lower bound. This can occur in cases where, due to dispersions, the target state is unreachable and the best-case scenario is to jettison as early or as late as possible.

Newton's method begins by perturbing an initial guess for the jettison time by some pre-determined amount; in this work, a perturbation of $\delta t_j = 0.5\ \text{s}$ is used and the initial guess is set to 700 s. For numerical consistency, the perturbation should be a multiple of the time step used by the predictor for fixed-time step integration. The apoapsis radius corresponding to this perturbed jettison time is then numerically predicted; note that this propagation is not explicitly represented in Fig. 3. The derivative of the objective function, in this case the slope of apoapsis radius as a function of jettison time $r'_a(t_j)$, is then approximated via first-order finite differencing as shown in Eq. 9. The updated jettison time is then computed via Eq. 10, which finds the x-intercept of the tangent line. The apoapsis radius resulting from the updated jettison time $t_{j,i+1}$ is numerically predicted, and the error is computed and checked against the tolerance ϵ_2. For a sufficiently accurate linearization and a nonzero slope of $r_a(t_j)$, the error should decrease each step. The subroutine repeats until either converging within the tolerance ϵ_2 or reaching a maximum number of iterations.

$$r'_a(t_j) \approx \frac{r_a(t_j + \delta t_j) - r_a(t_j)}{\delta t_j} \tag{9}$$

$$t_{j,i+1} = t_{j,i} - \frac{r_a(t_j)}{r'_a(t_j)} \tag{10}$$

The advantage of combining these two root-finding methods in a single guidance algorithm is that bisection method is robust but relatively slow, whereas Newton's method tends to converge more efficiently but requires a sufficiently-accurate initial guess. In particular, for a more typical aerocapture scenario in which the initial orbit is hyperbolic, escape cases that are still hyperbolic after exiting the atmosphere can be frequently encountered and may exist near the optimal solution for a high-energy target orbit. In these cases apoapsis radius is poorly-defined and the elliptical Keplerian equations would yield a negative value. Because the error no longer varies smoothly, the gradient is poorly-behaved and Newton's method fails to accurately converge to the solution. Bisection method, on the other hand, can handle errors of $\pm\infty$ and thus behaves as desired when escape cases are simply assigned an error of ∞. Once converged to a solution, however, the optimal jettison time (as predicted based on the navigation-estimated states) tends to require only small corrections in subsequent guidance calls. Because the initial guess is good, Newton's method can more efficiently compute these minor adjustments as long as the perturbation step is tuned appropriately. Note that a possible alternative implementation of the NPC would, during a single guidance update, call the Newton's method subroutine after the bisection method reduces the error to be between the two tolerance values. However, bisection method is typically only used in either edge cases where the solution is unreachable or at times far from the optimal jettison time, and therefore this modification would add computational expense with a negligible impact on performance.

The output of this prediction-correction loop is a jettison time t_j. In Fig. 3 the logic to command jettison once this time is reached or exceeded is portrayed as part of the guidance algorithm. However, note that this command is not necessarily limited to the update frequency of the guidance algorithm. Instead, t_j can be output by the guidance and a separate jettison controller can check the current time and command drag skirt jettison when t_j is reached. This controller is simple and can run at a higher rate than the guidance algorithm, enabling higher-resolution commanding of jettison and a corresponding improvement in targeting accuracy. Finally, the "guidance called" delay block in Fig. 3 reflects the fact that this process is called at a fixed rate rather than constantly iterating.

A significant drawback of the NPC guidance algorithm is that the number of iterations required to converge is indeterminate. That is, while an upper limit on the number of iterations can be enforced, there is no guarantee on the resulting error magnitude once this limit is reached. Each guidance call requires a minimum of one numerical propagation, used to determine whether or not the current t_j results in apoapsis error within the tolerances. The bisection subroutine requires one additional propagation per iteration, and Newton's method, while more efficient, requires two propagations per iteration (one perturbed, one corrected). The end result is that the NPC is not only computationally expensive due to the requirement of onboard propagation, but the number of operations required for convergence is in general **unknown**. In practice the number of propagations required for convergence can be approximately bounded through analysis with expected dispersions, as shown in Section 5, but the lack of a theoretical guarantee can make validation of the NPC approach difficult.

4. Energy reference guidance

The energy reference guidance (ERG) algorithm proposed in this work[2] seeks to achieve comparable performance to the NPC while reducing computational requirements. The algorithm is summarized in Fig. 4 and outlined in detail in this section, with results provided in Section 5. ERG is divided into two phases: a pre-compute phase that is executed after the final OD update to the spacecraft is received, and an algorithm that is executed each time guidance is called during the atmospheric flight phase.

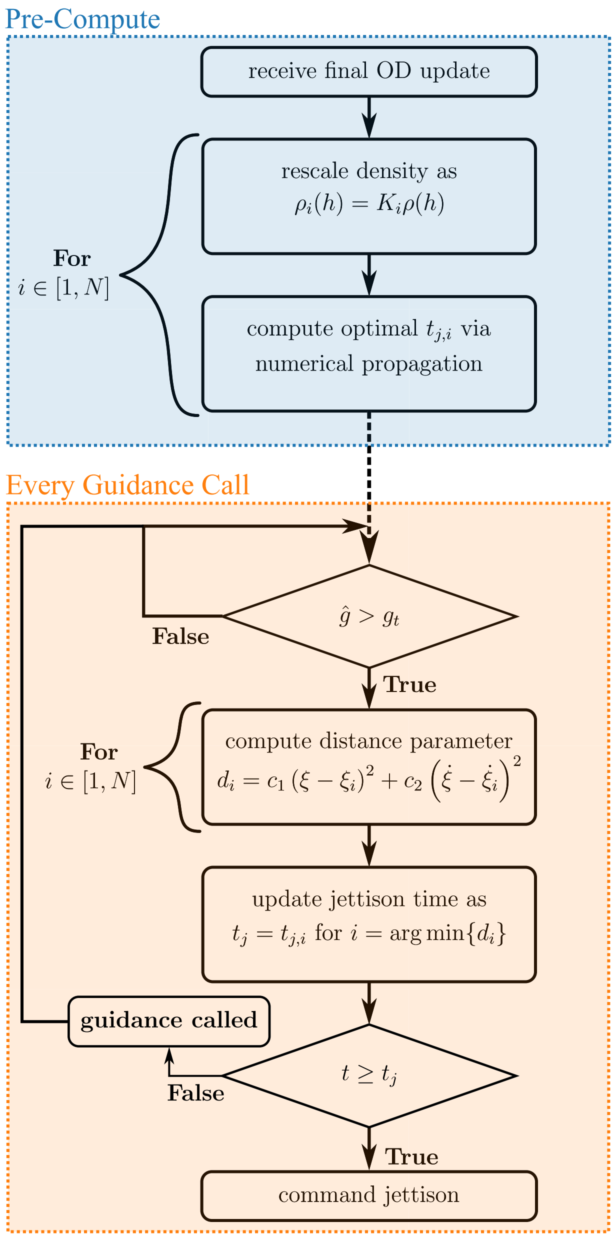

Pre-Compute
receive final OD update
For $i \in [1, N]$
rescale density as $\rho_i(h) = K_i\rho(h)$
compute optimal $t_{j,i}$ via numerical propagation
Every Guidance Call
$\hat{g} > g_t$
False
True
For $i \in [1, N]$
compute distance parameter $d_i = c_1(\xi - \xi_i)^2 + c_2\left(\dot{\xi} - \dot{\xi}_i\right)^2$
update jettison time as $t_j = t_{j,i}$ for $i = \arg\min\{d_i\}$
guidance called
$t \geq t_j$
False
True
command jettison

Fig. 4. Energy reference guidance diagram.

During the pre-compute phase, a smoothly-varying family of reference trajectories is generated and stored for later use. The ith reference trajectory is computed by linearly rescaling the nominal density profile by some factor K_i, then solving for the optimal jettison time $t_{j,i}$ through an iterative prediction-correction procedure. This jettison time optimization is equivalent to the Newton's method subroutine

[2] ERG is equivalent to the simplified form of the QIC algorithm proposed by the authors in Burnett et al. (2022).

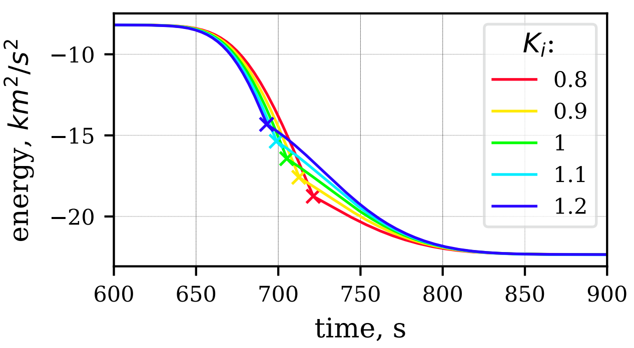


Fig. 5. Orbital energy vs. time for family of reference trajectories, where X marks optimal jettison time.

from the NPC algorithm, and similarly relies on numerical propagation from the navigation-estimated states. Fig. 5 shows an example set of reference trajectories, where the trajectories with earlier optimal jettison times correspond to denser atmospheres (larger K_i values).

In this work a range of $K_i \in [0.8, 1.2]$ is used based on trial and error; this range depends on the expected dispersions, atmospheric and otherwise, and is pre-defined on the ground. The smallest and largest K_i values correspond to the worst-expected overshoot and undershoot cases, respectively, based on both expected dispersions (aleatory uncertainty) and a potential lack of data at other planetary destinations (epistimic uncertainty), and can be conservative. The tradeoff for conservatism in these values is an incremental increase in memory and CPU requirements, but this has marginal effect on the CPU demand during the atmospheric flight phase. The number of K_i values, N, and the resolution at which reference trajectory data are saved are treated as tuning parameters and discussed in Section 5. It is important to note that this method of linearly re-scaling density is not meant to be a good model of how density dispersions behave in real atmospheres, in which dispersions vary with position and time. Additionally, note that it would be possible to implement ERG with other methods of modifying density to generate a family of reference trajectories, such as varying atmospheric scale height of an exponential model. In SubSection 5.2 ERG is tested against the higher-fidelity density dispersions provided by GRAM as described in SubSection 2.3.

During the atmospheric flight phase, guidance is called periodically and is active while sensed deceleration is above a threshold value, just like in the NPC guidance. Once a guidance call is initiated, the algorithm determines the reference trajectory that most closely matches the vehicle trajectory at the current time. This is accomplished via a heuristic distance parameter d:

$$d_i = c_1(\xi - \xi_i)^2 + c_2\left(\dot{\xi} - \dot{\xi}_i\right)^2 \tag{11}$$

where ξ and $\dot{\xi}$ are the energy and energy rate computed from the current navigation-estimated state, respectively, ξ_i and $\dot{\xi}_i$ are the energy and energy rate along the ith reference trajectory at the current time, and c_1 and c_2 are tuning parameters. Energy is specific orbital energy,

$$\xi = \frac{|\dot{\boldsymbol{r}}|^2}{2} - \frac{\mu}{|\boldsymbol{r}|}, \tag{12}$$

and energy rate is computed by differencing the current energy with the energy computed from a prior state estimate. The values along the reference trajectory are approximated for the current time by using the values at the time step immediately prior to the current time. See Section 5 for a discussion of why this method is chosen as opposed to interpolation, and for a discussion of the values of c_1 and c_2. The motivation for this choice of distance parameter is that the target orbit is associated with a particular energy value and, since the vehicle lacks any out-of-plane control authority, the guidance objective can be posed as an energy-targeting problem without loss of generality. The current energy of a trajectory gives information about the remaining energy that must be dissipated, and the current energy rate of that trajectory gives information about whether the vehicle is on track to reach the desired energy upon atmospheric exit as compared to pre-optimized reference trajectories.

Once d_i is computed for each reference trajectory, the reference with the smallest distance parameter is selected as the nearest match. Then, the algorithm simply updates the jettison time t_j to equal the jettison time that was computed for that nearest reference trajectory, $t_{j,i}$. Like NPC guidance, the algorithm outputs a jettison time that is monitored by a jettison controller that is potentially running at a higher rate.

To summarize, in ERG a family of optimized reference trajectories is generated during a pre-compute step. Then, during atmospheric flight updates, the nearest reference is selected based on a heuristic distance parameter and the commanded jettison time is updated to equal the jettison time associated with that reference. ERG has a number of things in common with the NPC guidance algorithm. Namely, both algorithms rely on onboard numerical propagation from a navigation-estimated state, and both solve for optimal jettison time in a root-finding procedure that requires an indeterminate number of iterations to converge. The key difference, however, is that in ERG the numerical propagations occur in a pre-compute phase that occurs before atmospheric entry, and is thus significantly less time-constrained. That is, whereas the NPC requires the prediction-correction procedure to converge during a single guidance call (0.125 s in this case), ERG only requires that the procedure converge for each reference in the time between OD cutoff and atmospheric entry. In fact, if the link budget and timing of the mission design allow, the pre-compute step could be performed on the ground and the relevant data could be uplinked along with the final OD update. Moreover, OD cutoff could be shifted earlier if necessary to allow a longer time for the pre-compute phase. An earlier OD cutoff does result in higher navigation error at entry, though, so this creates a tradeoff between accuracy and onboard computation requirements.

Quantitatively comparing the computational demand of these two algorithms would require hardware-in-the-loop simulation of a flight software-like implementation of each algorithm, which is beyond the scope of this study. While logged CPU time on a research computer is sometimes used as a basis of comparison in the literature, this approach can result in misleading data. The implementations of these two algorithms are developed as proofs-of-concept, not designed to emulate a flight software implementation and optimized for efficiency; additionally, other processes can draw from the same computing resource and affect the CPU time required. Nevertheless, the ERG algorithm has two clear advantages over the NPC in terms of CPU demand. During the atmospheric flight phase of ERG, no numerical propagation or iterative root-finding is required; the algorithm simply evaluates a mathematical expression for the distance parameter associated with each reference trajectory, then selects the minimum from among these values. It is clear that, when the algorithms are tuned for comparable performance, ERG requires significantly fewer computer operations per guidance call than the NPC and is less demanding of CPU capacity as a result. A second important feature of the ERG algorithm is that it is computationally well-posed, in that the number of individual operations required per guidance call can be predicted exactly. In contrast, the NPC requires an indeterminate number of numerical propagations to reach a given convergence tolerance as part of its root-finding procedure during each guidance call. Reduced algorithmic complexity and an ability to closely theoretically constrain CPU demand are significant advantages of the ERG over the NPC when it comes to verification and validation of flight software, especially in the case of radiation–hardened avionics with limited capacity.

Although ERG is less demanding of CPU capability, this is traded-off by a higher memory requirement compared to NPC. The time, energy, and energy rate at each point along each reference trajectory must be stored in memory and remain accessible to the guidance algorithm. Thus, the total memory required is a product of the number of reference trajectories, the number of datapoints per trajectory, and the memory required per value (e.g. 64 bits for double-precision numbers). In the following section, the relationship between targeting performance and required memory is quantitatively explored.

5. Results

5.1. NPC performance

Fig. 6 shows the histogram of apoapsis altitudes achieved using the NPC guidance in a 5001-trial Monte Carlo analysis, modeling the scenario and uncertainties as described in Section 2. The mean and standard deviation apoapsis altitude achieved by NPC are 5057 km and 357 km, respectively; recall that the target is 5000 km. The data are approximately Gaussian, with the exception of a small right skew due to a small number of high-apoapsis outliers. Note that one cause of these outliers is that the dispersions assumed in this work sometimes exceed the total control authority of the vehicle. For example, in cases where the atmospheric density is below nominal and simultaneously navigation errors result in delivery with a shallower entry flight-path angle than desired, the vehicle may overshoot the target orbit even if the drag skirt is never jettisoned.

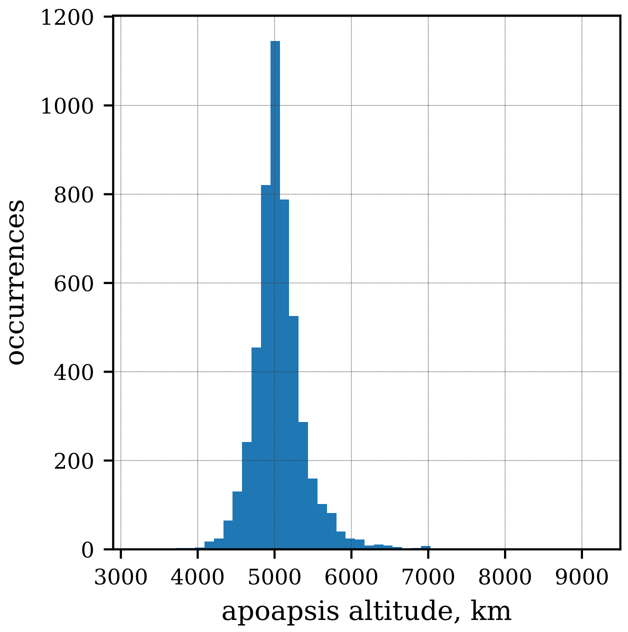


Fig. 6. Targeting results for NPC, 5001-trial Monte Carlo analysis.

In order to roughly assess the computational demand of the NPC algorithm, the number of propagations required per guidance call is counted and the maximum of this value is recorded for each trial; denote this maximum p_{max} for convenience. In 88% of cases $p_{max} = 7$, and in all but 2 of the 5001 trials $p_{max} \leqslant 7$; the maximum observed value was 11. Because the NPC lacks guarantees on the number of iterations required for convergence, this type of numerical analysis would be required to bound the required computational capacity. The statistics of p_{max} are affected by the incoming trajectory, target orbit, assumed dispersions, tuning of the guidance algorithm, and a number of other implementation details.

5.2. Baseline ERG performance

The targeting performance for ERG under the same circumstances is shown in Fig. 7, where $N = 17$ reference trajectories are generated. In this case the mean and standard deviation apoapsis altitude are 5009 km and 355 km, respectively, as summarized in Table 2. Statistically speaking, these targeting results are approximately equivalent;

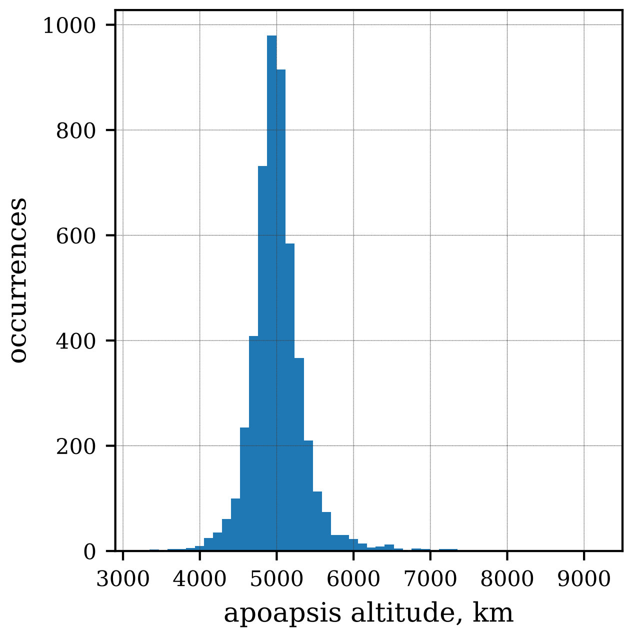


Fig. 7. Targeting results for ERG, 5001-trial Monte Carlo analysis.

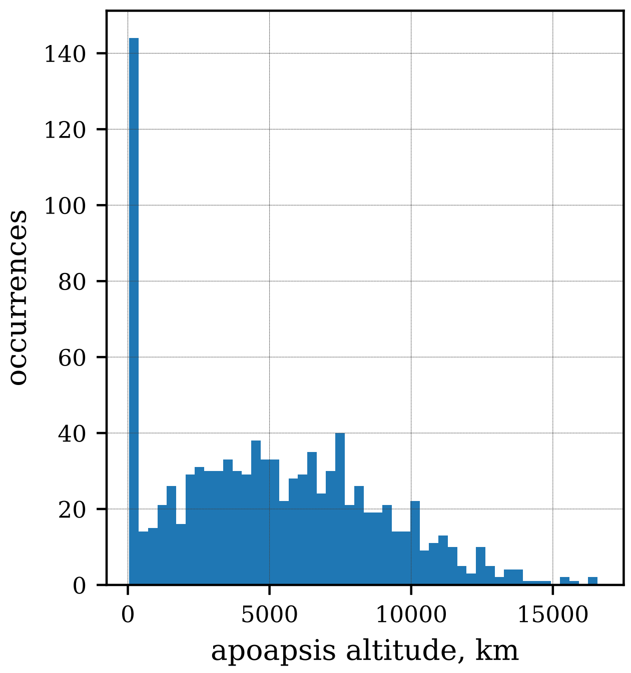


Fig. 8. Targeting results for fixed-time jettison optimized *a priori*, 1001-trial Monte Carlo analysis.

Table 2
Apoapsis altitude statistics for baseline NPC and ERG.

Algorithm	Mean, km	σ, km
NPC	5057	357
ERG	5009	355

the ERG algorithm achieves targeting performance almost identical to that of the baseline NPC algorithm. Though the mean apoapsis altitude of the ERG has lower error than that of the NPC, this difference is insignificant in the context of a 5000 km target apoapsis and standard deviation of more than 350 km. This is remarkably good performance considering that ERG can only choose from a set of 17 options for jettison time, whereas the NPC guidance refines jettison time to within a small tolerance.

Figs. 8 and 9 provide a comparison that gives some insight into how ERG is able to accurately target a final orbit. In both cases a single jettison time is chosen before atmospheric entry and used in every trial. In Fig. 8, t_j is optimized *a priori* based on the nominal scenario, whereas in Fig. 9 t_j is optimized using simulations beginning from the navigation-estimated state after OD cutoff 10 min prior to entry. Put differently, the former case is open-loop control and the latter case is equivalent to ERG with only a single reference trajectory.

In the open-loop case shown in Fig. 8, targeting performance is very poor. A significant number of cases either impact the planet or have apoapsis altitudes so low that the vehicle is doomed to re-enter before having a chance to maneuver, with 13.3% of cases reaching an apoapsis below 200 km. There is also a high number of overshoot cases and a wide spread to the data. The case in Fig. 9,

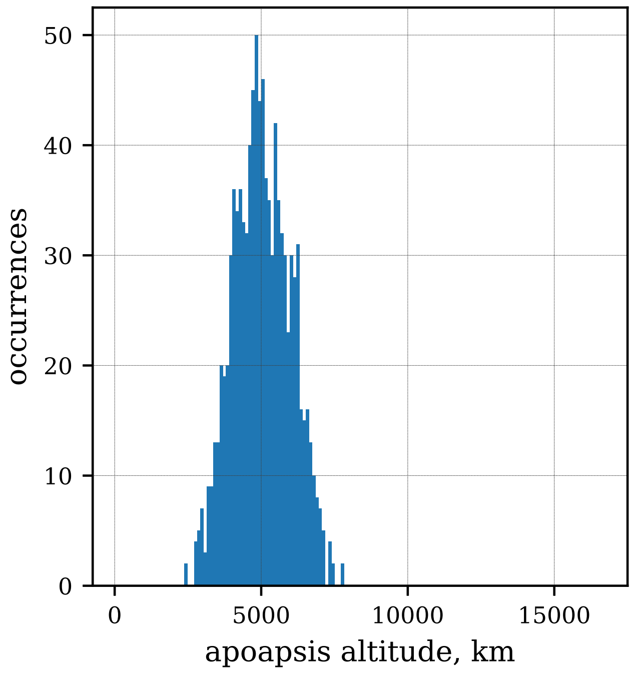


Fig. 9. Targeting results for fixed-time jettison optimized after OD cutoff, 1001-trial Monte Carlo analysis.

shown with the same x-axis scaling, stands in sharp contrast. With a standard deviation of 971 km it is significantly worse than the case with 17 reference trajectories shown in Fig. 7, but performs far better than the case in Fig. 8, avoiding any impact cases or any apoapsis altitudes above 8000 km.

This comparison serves to illustrate the following point. The state of the vehicle at atmospheric entry is subject to two distinct types of dispersions: delivery error and navigation error. The former is the difference between the pre-planned nominal entry state and true state, whereas the latter is the difference between the onboard best-estimate of the state, based on filtered navigation data, and the true state. Under the assumptions for this mission scenario, delivery error generally exceeds navigation error; that is, the spacecraft is delivered to entry with limited accuracy, but navigation filters produce a fairly accurate state estimate by the time of OD cutoff. The results in Fig. 8 use a jettison time based on the nominal entry state and are thus subject to both delivery and navigation errors. The Fig. 9 results, in contrast, use a jettison time based on the navigated state at OD cutoff, which effectively removes most of the delivery error. Therefore, it is clear that much of the benefit from the ERG algorithm is simply a result of re-computing a reference trajectory (in this case, a jettison time) onboard the spacecraft using an updated state estimate.

5.3. ERG tuning

Recall that the ERG algorithm can be tuned by adjusting the values of c_1 and c_2 in the distance parameter, Eq. 11. A parametric study was carried out to find values of these parameters that offer reasonable performance, with results shown in Table 3. In order to eliminate other factors, these cases used 401 reference trajectories with 8000 datapoints per trajectory. A tuning of $c_1 = 1, c_2 = 10$ is selected based on its minimum standard deviation result, and is used for all following results as well as for the baseline case in Fig. 7. It is interesting to note that the minimum-variance case occurs when c_1 and c_2 are of similar magnitude, and that when either parameter is set to zero performance degrades significantly. This highlights the fact that energy and energy rate are both necessary for the best match with a reference trajectory.

5.4. Memory vs. performance trade-Offs

Although the ERG algorithm is significantly less demanding of CPU capability, it is significantly *more* demanding of memory space accessible to the guidance algorithm. It is therefore of interest to quantify trade-offs between memory and performance for the ERG algorithm. The storage required is estimated as the product $3 \times N \times M \times D$ where N is the number of reference trajectories, M is the number of datapoints per reference trajectory, and D is the required memory per datapoint, and where 3 is pre-multiplied because each reference trajectory requires storing time, energy, and energy rate at each datapoint.

In Fig. 10, the number of reference trajectories is varied from 1 to 401 and the apoapsis altitude results are compared, with a 1001-trial Monte Carlo analysis performed in each case. $M = 8000$ datapoints are recorded for each reference trajectory. The mean and standard deviation of apoapsis altitude for these same trials are listed in Table 4. From these results, it is clear that increasing the number of reference trajectories above 81 makes no discernable difference in performance. From 81 to 17 there is a small increase in standard deviation, then from 17 to 9 a larger increase in variability and the first noticeable change in the histogram. For fewer than 9 reference trajectories, performance significantly degrades. Note that the mean remains centered for all cases, as overshoot and undershoot cases increase at approximately the same rate as the number of reference trajectories is decreased. Based on this analysis, a reasonable balance between memory and performance seems to be $N = 17$ reference trajectories. Note that this inflection point may change for differing mission scenarios.

A similar analysis is presented in Fig. 11 and Table 5, where in this case the number of datapoints per trajectory is varied from 8000 to 500 while holding the number of ref-

Table 3
Apoapsis altitude statistics for varying distance parameter tuning.

c_1	c_2	Mean, km	σ, km
1	0	6034	601
100	1	5036	389
10	1	5033	382
1	1	5027	361
1	10	5019	338
1	100	5022	340
1	1000	5053	416
0	1	5121	578

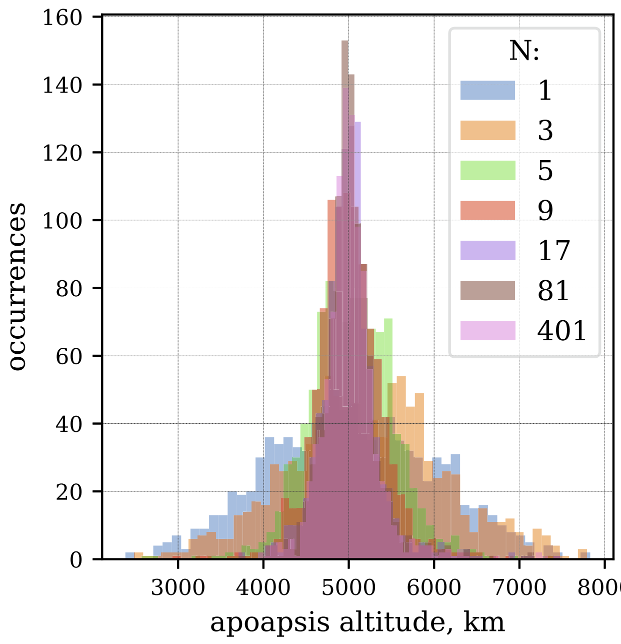


Fig. 10. Performance comparison for varying number of reference trajectories, 1001-trial Monte Carlo analysis.

Table 4
Apoapsis altitude statistics for varying number of reference trajectories.

N	Mean, km	σ, km
401	5019	338
81	5020	338
17	5010	361
9	5008	415
5	5036	511
3	5210	914
1	5026	971

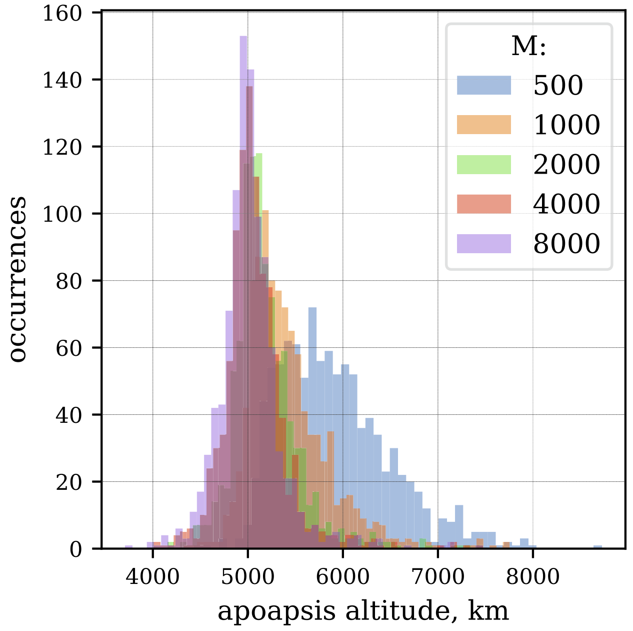


Fig. 11. Performance comparison for varying reference trajectory resolution, 1001-trial Monte Carlo analysis.

Table 5
Apoapsis altitude statistics for varying number of datapoints per reference trajectory.

M	Mean, km	σ, km
8000	5020	338
4000	5076	345
2000	5190	365
1000	5425	433
500	5912	588

erence trajectories constant at 81. Note that the numerical propagation always occurs with a timestep of 0.125 s, meaning that for $M = 4000$ a datapoint is recorded every other step, for $M = 2000$ every 4 steps, etc., assuming propagation for 1000 s total.

Whereas the data in Fig. 10 remain centered while the spread increases, in this case there is a shift to the right combined with an increased spread each time that M is decreased. That is, recording fewer datapoints results in a bias toward overshoot cases as well as increasing variability. Moreover, in these results targeting performance begins to degrade immediately, without a clear inflection point.

To understand these trends, recall that during the atmospheric flight phase the reference energy and energy rate values are approximated by using data from the time step immediately prior to the current time. Thus, for a resolution of 500 datapoints, the values used to compute the distance parameters are associated with a point on the trajectory up to two seconds earlier than the current time. This effectively inflates the energy of every reference point, and the result is that the matched trajectory has a higher density scale factor than it otherwise would, leading to an earlier jettison time and ultimately the skew toward overshoot cases observed in Fig. 11. It may seem as though using interpolation to compute energy and energy rate of the reference trajectory at the current time would address this issue. However, energy rate changes as a step function at the moment of drag skirt jettison. In short, interpolating across this discontinuity disrupts the ability of the algorithm to successfully match with the reference trajectory that would actually yield optimal performance. Therefore, in this work values from the previous time are used and the requirement for high-resolution reference trajectory data is accepted; a value of $M = 8000$ is taken to be the baseline configuration shown in Fig. 4. As a point of reference, if double-precision values of 64 bits each are assumed for this baseline configuration, a total of about 3.3 MB of memory would be required. In comparison, the Sphinx avionics platform, which was developed at JPL for SmallSat missions and now has flight heritage from the Lunar Flashlight spacecraft (Rizvi et al., 2022), includes 256 MB of synchronous dynamic RAM (Imken et al., 2017) This suggests that the 3.3 MB requirement is well within reason. Thus, while the tradeoff of reduced CPU demand for the ERG is increased memory requirement, this increase is not likely to represent a significant detriment to the overall design.

6. Discussion

A notable limitation of both algorithms presented in this work is that path constraints, such as peak heat flux and peak g-load, are not incorporated into the onboard logic. While other work does provide a method to account for these constraints in NPC guidance for aerocapture and entry (Lu et al., 2015; Lu, 2014), there is currently no equivalent approach for ERG. The impact of this limitation strongly depends on the mission scenario of interest. For the small satellite demonstration mission studied here, the vehicle design is expected to have significant margin compared to the expected heating and g-loads, and thus it is likely unnecessary for the onboard guidance to directly incorporate the associated constraints. In more stressing cases for which the nominal scenario is near the limits of heating and g-loads, an additional outer loop could be added to the ERG algorithm to prohibit executing jettison times that are predicted to have an unacceptably high likelihood of resulting in path constraint violation.

The most likely barriers to implementation of this algorithm are the computation time required to generate the reference trajectories and the memory required to store the associated data. Therefore, it would be of interest to extend the approach presented in this work to achieve the same performance with fewer reference trajectories, or else improve performance with the same number.

A potential approach would be to interpolate between the reference trajectories in some way, such that the commanded jettison time does not necessarily equal one of the reference jettison times. Because the current vehicle state will generally not equal the state at that time along even the nearest reference trajectory, the difference between the current and reference state could inform a correction to the jettison time of that reference trajectory. One could accomplish this by computing linear sensitivities of jettison time with respect to each relevant state component, then computing the correction term as the product of this sensitivity and the state difference. The altitude, velocity magnitude, and flight-path angle could be considered a sufficient set of state components since the primary concern is planar motion. However, there are two significant issues with that approach. First, this would require computing and storing sensitivity values at each time along each reference trajectory, resulting in a major increase in CPU demand and, assuming three state sensitivities, doubling the amount of memory required. Second, even setting aside the computational challenges, the dynamics are nonlinear and the true state tends to diverge significantly from any of the reference trajectories over time, leading to inaccurate linearization.

One possible workaround is the incorporation of quasi-initial conditions. These fully represent the current state by back-propagating through a nominal model, effectively defining a nonlinear coordinate transformation. Quasi-initial conditions have been shown to be a more linear state representation than the state at a given time for aerocapture (Grace et al., 2022). This state representation also removes the requirement of computing sensitivities at each time, since they need only be computed once in quasi-initial condition space, although a single back-propagation per guidance call is then required during atmospheric flight. Preliminary work by the authors incorporates quasi-initial conditions into an extension of the ERG algorithm presented here (Burnett et al., 2022). While early results are promising, it is difficult to guarantee reliable and accurate linearization in the presence of dispersions, whereas the simpler approach presented here performs well. Furthermore, note that while the computational burden of the quasi-initial condition approach is far less than a linearization based on the current state, it does still require numerically computing three sensitivity values for each reference trajectory, meaning that the number of numerical propagations during the pre-compute phase increases by roughly a factor of four.

Another interesting avenue for future work is some method of nonlinear corrections to the reference jettison time. This could be combined with the previous concept, such that some nonlinear interpolation surface is generated in quasi-initial condition space during the pre-compute step and then used to guide corrections during the atmospheric flight phase. This could potentially alleviate issues related to inaccurate linearization, although it would likely require a commensurate increase in computational cost.

7. Conclusions

It is worth returning here to the single-event jettison concept itself. This control architecture inherently sacrifices performance in pursuit of simplicity. By relying on the jettison of a single rigid drag skirt, the vehicle lacks any out-of-plane control authority, forgoes continuous control and, perhaps most importantly, is coasting without any control authority for the remainder of atmospheric flight once the drag skirt is jettisoned. A range of other approaches address one or more of these shortcomings, including continuously-variable drag modulation (Vinh et al., 1986), jettison of multiple drag skirts (Roelke et al., 2022), and lift modulation (Vinh et al., 2000; Deshmukh et al., 2020). However, each of these architectures adds complexity in terms of flight hardware and, in most cases, flight software. The motivation to use single-event jettison drag-modulation is not to achieve orbit insertion as accurately as possible; rather, the goal is to reliably reach the target orbit within some reasonable error bounds while keeping the aerocapture subsystem as simple as possible. This is appropriate either for missions that can tolerate a range of apoapsis altitudes or for cases where the spacecraft has sufficient propellant to clean up the expected targeting errors.

This broader motivation should inform the choice of guidance algorithm and the interpretation of results. In this work a novel guidance algorithm, ERG, is presented that achieves equivalent targeting performance to the baseline NPC. Both algorithms have a standard deviation of about 355km and in some outlier cases reach an apoapsis several thousand kilometers higher than the target. However, the choice of an inherently limited control architecture limits the ability of any guidance algorithm to accurately target a final orbit. The fact that the two distinct algorithms achieve nearly-identical results could suggest that both are operating near the ceiling of performance for this scenario. The ERG algorithm achieves this result with significantly reduced CPU demand, albeit with an increased demand for accessible memory. The simplicity of the atmospheric flight phase of the ERG algorithm aligns well with the broader motivation to reduce complexity for this type of mission scenario.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank Aaron Schutte and Bill Strauss for supporting work performed at NASA JPL for this paper. Additionally, the DSENDS environment in which these simulations were performed was developed and supported by Abhi Jain and others in the JPL DARTS Lab. This work was supported by a NASA Space Technology Research Fellowship. A portion of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

Agrawal, P., Allen, G.A., Sklyanskiy, E.B., et al., 2014. Atmospheric entry studies for Uranus. In: 2014 IEEE Aerospace Conference. IEEE, Big Sky, MT, USA, pp. 1–19. https://doi.org/10.1109/AERO.2014.6836417.

Albert, S.W., Doostan, A., Schaub, H., 2021. Finite-dimensional density representation for aerocapture uncertainty quantification. In: AIAA Scitech 2021 Forum AIAA 2021-0932, Virtual event, pp. 1–17. https://doi.org/10.2514/6.2021-0932.

Amato, D., Hume, S., Grace, B. et al., 2020. Robustifying mars descent guidance through neural networks. In: AAS Guidance, Navigation, and Control Conference AAS 20-073, pp. 1–13.

Austin, A., 2020. White Paper: Enabling and Enhancing Science Exploration Across the Solar System: Aerocapture Technology for SmallSat to Flagship Missions. Technical Report NASA JPL/CalTech.

Austin, A., Nelessen, A., Strauss, B. et al., 2019. Smallsat aerocapture to enable a new paradigm of planetary missions. In: 2019 IEEE Aerospace Conference, pp. 1–20. https://doi.org/10.1109/AERO.2019.8742220.

Braun, R.D., Spencer, D.A., Kallemeyn, P.H., et al., 1999. Mars pathfinder atmospheric entry navigation operations. J. Spacecraft Rock. 36 (3), 348–356. https://doi.org/10.2514/2.3477.

Burnett, E.R., Albert, S.W., Schaub, H., 2022. A new guidance technique for discrete-event drag modulation for aerocapture missions. In: AAS Guidance, Navigation, and Control Conference AAS 22-102, pp. 1–15.

Burnett, E.R., Schaub, H., 2022. Approximating orbits in a rotating gravity field with oblateness and ellipticity perturbations. Celestial Mech. Dyn. Astron. 134 (1), 5. https://doi.org/10.1007/s10569-022-10061-z.

Cameron, J.M., Jain, A., Burkhart, P.D. et al., 2016. Dsends: Multi-mission flight dynamics simulator for nasa missions. In: AIAA SPACE 2016, pp. 1–18. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2016-5421.

Cassell, A., Smith, B., Wercinski, P. et al., 2018. ADEPT, A Mechanically deployable re-entry vehicle system, enabling interplanetary cubesat and small satellite missions. In: Proceedings of the Small Satellite Conference, Session 3: Advanced Technologies I, pp. 1–8. URL: https://digitalcommons.usu.edu/smallsat/2018/all2018/265/.

Cazaux, C., Naderi, F., Whetsel, C., et al., 2004. The NASA/CNES Mars sample return— a status report. Acta Astronaut. 54 (8), 601–617. https://doi.org/10.1016/j.actaastro.2003.07.001.

Cheng, L., Jiang, F., Wang, Z., et al., 2021. Multiconstrained real-time entry guidance using deep neural networks. IEEE Trans. Aerosp. Electron. Syst. 57 (1), 325–340. https://doi.org/10.1109/TAES.2020.3015321.

Cruz, M.I., 1979. The aerocapture vehicle mission design concept. In: Conference on Advanced Technology for Future Space Systems. AIAA, pp. 195–201. https://doi.org/10.2514/6.1979-893.

Deshmukh, R., 2021. System Analysis of a Numerical Predictor-Corrector Aerocapture Guidance Architecture. Ph.D. thesis Purdue University.

Deshmukh, R.G., Spencer, D.A., Dutta, S., 2020. Investigation of direct force control for aerocapture at neptune. Acta Astronaut. 175, 375–386. https://doi.org/10.1016/j.actaastro.2020.05.047.

Dutta, S., 2021. Aerocapture as an enhancing option for ice giants missions. Bull. AAS 53 (4), URL: https://baas.aas.org/pub/2021n4i046.

Falcone, G., Putnam, Z.R., 2022. Autonomous decision-making for aerobraking via parallel randomized deep reinforcement learning. IEEE Trans. Aerospace Electronic Syst., 1–18 https://doi.org/10.1109/TAES.2022.3221697.

Falcone, G., Williams, J.W., Putnam, Z.R., 2019. Assessment of aerocapture for orbit insertion of small satellites at mars. J. Spacecraft Rock. 56 (6), 1689–1703. https://doi.org/10.2514/1.A34444.

Grace, M.J., Burnett, E.R., McMahon, J.W., 2022. Quasi-Initial Conditions as a State Representation for Aerocapture. In: AIAA SciTech Forum, pp. 1–16.

Gulick, D., Lewis, J., Miller, K. et al., 2003. Trailing ballute aerocapture: System definition. In: 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, pp. 1–12. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2003-4655.

Hall, J.L., Noca, A.N., Bailey, R.W., 2005. Cost-benefit analysis of the aerocapture mission set. J. Spacecraft Rock. 42 (2), 309–320.

Heidrich, C., Roelke, E.E., Albert, S.W., et al., 2020. Comparative study of lift- and drag-modulation control strategies for aerocapture. In: 43rd Annual AAS Guidance, Navigation and Control Conference, Breckenridge, CO, pp. 1–16.

Imken, T., Castillo-Rogez, J., He, Y. et al., 2017. Cubesat flight system development for enabling deep space science. In: 2017 IEEE Aerospace Conference, pp. 1–14. https://doi.org/10.1109/AERO.2017.7943885.

Johnson, W., Lyons, D., 2004. Titan ballute aerocapture using a perturbed titangram model. In: AIAA Atmospheric Flight Mechanics Conference and Exhibit, pp. 1–12. https://doi.org/10.2514/6.2004-5280.

Leslie, F.W., Justus, C.G., 2011. The NASA Marshall Space Flight Center Earth Global Reference Atmospheric Model - 2010 Version. Technical Report NASA/TM-2011-216467 NASA.

Lu, P., 2014. Entry guidance: A unified method. J. Guidance Control Dyn. 37 (3), 713–728. https://doi.org/10.2514/1.62605.

Lu, P., Cerimele, C.J., Tigges, M.A., et al., 2015. Optimal aerocapture guidance. J. Guidance Control Dyn. 38 (4), 553–565. https://doi.org/10.2514/1.G000713.

Moseley, P., 1969. The apollo entry guidance: A review of the mathematical development and its operational characteristics. Task MSC/TRW A-220.

Nelessen, A., Sackier, C., Clark, I. et al., 2019. Mars 2020 entry, descent, and landing system overview. In: 2019 IEEE Aerospace Conference, pp. 1–20. https://doi.org/10.1109/AERO.2019.8742167.

Perot, E., Rousseau, S., 2002. Importance of an on-board estimation of the density scale height for various aerocapture guidance algorithms. In: AIAA/AAS Astrodynamics Specialist Conference and Exhibit, pp. 1–11. https://doi.org/10.2514/6.2002-4734.

Powell, R., 2012. Numerical roll reversal predictor corrector aerocapture and precision landing guidance algorithms for the mars surveyor program 2001 missions. In: 23rd Atmospheric Flight Mechanics Conference, pp. 1–9. URL: https://arc.aiaa.org/doi/abs/10.2514/6.1998-4574.

Powell, R., Braun, R., 2012. Six-degree-of-freedom guidance and control analysis of mars aerocapture. J. Guidance Control Dyn. 16 (6), 1038–1044. https://doi.org/10.2514/6.1992-736.

Putnam, Z.R., Braun, R.D., 2014. Drag-modulation flight-control system options for planetary aerocapture. J. Spacecraft Rock. 51 (1), 139–150. https://doi.org/10.2514/1.A32589.

Putnam, Z.R., Neave, M.D., Barton, G.H., 2010. Predguid entry guidance for orion return from low earth orbit. In: 2010 IEEE Aerospace Conference, pp. 1–13. https://doi.org/10.1109/AERO.2010.5447010.

Rizvi, A., Ortega, K.F., He, Y., 2022. Developing lunar flashlight and near-earth asteroid scout flight software concurrently using open-source f prime flight software framework. In: Small Satellite Conference. URL: https://digitalcommons.usu.edu/smallsat/2022/all2022/104/.

Roelke, E., 2021. Guidance Algorithm and Vehicle Architecture Enhancements for Improved Discrete-Event, Drag-Modulated Aerocapture. Ph.D. thesis University of Colorado Boulder.

Roelke, E., Hattis, P., Braun, R., 2019. Improved atmospheric estimation for aerocapture guidance. In: AAS/AIAA Astrodynamics Specialist Conference, pp. 1–16.

Roelke, E., McMahon, J.W., Braun, R.D., et al., 2022. Multi-event jettison guidance approaches for drag-modulation aerocapture. J. Spacecraft Rock. 59 (1), 190–202. https://doi.org/10.2514/1.A35059.

Rohrschneider, R.R., Braun, R.D., 2007. Survey of ballute technology for aerocapture. J. Spacecraft Rock 44 (1), 10–23. https://doi.org/10.2514/1.19288.

Rousseau, S., Perot, E., Graves, C. et al., 2012. Aerocapture guidance algorithm comparison campaign. In: AIAA/AAS Astrodynamics Specialist Conference and Exhibit, pp. 1–11. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2002-4822.

Shi, Y., Wang, Z., 2021. Onboard generation of optimal trajectories for hypersonic vehicles using deep learning. J. Spacecraft Rock. 58 (2), 400–414. https://doi.org/10.2514/1.A34670.

Spilker, T.R., Adler, M., Arora, N., et al., 2019. Qualitative assessment of aerocapture and applications to future missions. J. Spacecraft Rock. 56 (2), 536–545.

Steltzner, A.D., Miguel San Martin, A., Rivellini, T.P., et al., 2014. Mars science laboratory entry, descent, and landing system development challenges. J. Spacecraft Rock. 51 (4), 994–1003. https://doi.org/10.2514/1.A32866.

Strauss, W.D., Austin, A., Nelessen, A. et al., 2021. Aerocapture trajectories for earth orbit technology demonstration and orbiter science missions at venus, earth, mars, and neptune. In: AAS/AIAA Space Flight Mechanics Meeting AAS 21-229, pp. 1–20.

Vinh, N., Johnson, W., Longuski, J., 2000. Mars aerocapture using bank modulation. In: Astrodynamics Specialist Conference, p. 4424.

Vinh, N.X., Johannesen, J.R., Mease, K.D., et al., 1986. Explicit guidance of drag-modulated aeroassisted transfer between elliptical orbits. J. Guidance Control Dyn. 9 (3), 274–280. https://doi.org/10.2514/3.20103.

Wagner, J., Wilhite, A., Stanley, D. et al., 2011. An adaptive real time atmospheric prediction algorithm for entry vehicles. In: 3rd AIAA Atmospheric Space Environments Conference, pp. 1–27. https://doi.org/10.2514/6.2011-3200.

Walberg, G., Siemers, P., Calloway, R. et al., 1987. The aeroassist flight experiment. In: Brighton International Astronautical Federation Congress, pp. 1–11.

Wang, H., Elgohary, T.A., 2020. A simple and accurate apollo-trained neural network controller for mars atmospheric entry. Int. J. Aerospace Eng. 2020, 1–15. https://doi.org/10.1155/2020/3793740, Publisher: Hindawi.

Way, D., Powell, R., Masciarelli, J. et al., 2003. Aerocapture simulation and performance for the titan explorer mission. In: 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2003-4951.

Wegner, P., Ganley, J., Maly, J., 2001. Eelv secondary payload adapter (espa): providing increased access to space. In: 2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542), vol. 5. pp. 2563–2568, https://doi.org/10.1109/AERO.2001.931218.

Werner, M., Woollard, B., Tadanki, A. et al., 2017. Development of an earth smallsat flight test to demonstrate viability of mars aerocapture. In: 55th AIAA Aerospace Sciences Meeting, pp. 1–16. https://doi.org/10.2514/6.2017-0164.

Werner, M.S., Braun, R.D., 2019. Mission design and performance analysis of a smallsat aerocapture flight test. J. Spacecraft Rock. 56 (6), 1704–1713. https://doi.org/10.2514/1.A33997.