A Generalized Guidance Approach to In-Space Solid-Propellant Vehicle Maneuvers

Jason M. Everett*

Exploration-class vehicles that require fully autonomous ascent and descent must employ robust, explicit path-adaptive guidance algorithms that can operate in a wide range of physical environments. Vehicle designs that employ solid-propellant rocket motors (SRMs) for maneuvers are attractive from a systems engineering perspective because of their simplicity and reliability, but may cause complications for both mission designers and GNC engineers when dealing with total impulse uncertainty, as well as proper energy management of a motor with an uncontrolled cutoff time. This paper presents a simplified guidance algorithm, named Simple Cross-Product Steering (SxS), that was derived during early studies of NASA's Mars Sample Return mission's Mars Ascent Vehicle. The algorithm takes roots in a flight-proven guidance algorithm commonly referred to as Cross-Product Steering. This algorithm has been shown to provide sufficient guidance accuracy for in-space SRM burns in a simulated Martian environment, and preliminary studies have been conducted to test the algorithm in a solid-propellant lunar braking scenario. A method for predicting proper motor ignition time during execution of the Cross-Product Steering algorithm is the primary contribution of this paper. Mechanization notes are also provided that were realized in early phases of MAV. Results are shown for an example ascent vehicle a simulated Mars environment.

INTRODUCTION

NASA's Mars Ascent Vehicle (MAV) is a two-stage solid-propellant ascent vehicle that plays one part of several in the Mars Sample Return (MSR) mission architecture. Its objective is to safely and autonomously lift martian samples from the surface of the planet to a nominal, stable orbit. A separate deep space vehicle will rendezvous with the MAV in low-Mars orbit, retrieve the samples on-board the vehicle, and return to Earth where the samples can be studied under more rigorous scrutiny. The design of an entirely autonomous ascent vehicle, with access to only limited atmospheric information at launch, proves a steep challenge for every subsystem of the vehicle, GNC included. The solid-propellant nature of both phases proved reliable from a systems perspective; however, achieving acceptable orbital insertion accuracy quickly became a pressing issue for the Guidance subsystem of this vehicle.

During early studies conducted for MAV, several methods were tested for handling both the ascent and orbital insertion maneuvers of the vehicle. While open-loop flight throughout ascent and insertion was easily implementable, the resulting orbital insertion errors observed from dispersed simulations were large enough to warrant a study on other closed-loop algorithms. The idea of modifying Powered-Explicit Guidance (PEG)[1] to handle a numeric integration of externally computed thrust integrals was also explored, but it was

* Aerospace Engineer, EV42/Guidance, Navigation, and Mission Analysis Branch, NASA Marshall Space Flight Center, Huntsville, AL

decided early on that the level of complexity required to complete development of the modified algorithm was out of scope for the current analysis cycle of MAV.

The idea of employing Cross-Product Steering (CPS)[3] was then explored. CPS is a simplified guidance algorithm that calculates a computed desired inertial velocity, $\vec{v}_{go}$, required to obtain a specified guidance target/orbit. Initial simulation studies of employing CPS for the orbital insertion maneuver at ballistic apoapsis, and an open-loop vs. altitude scheme for the ascent maneuver, were promising. During dispersed simulation studies, though, any mass/motor performance variations injected into the simulation during the first burn caused large discrepancies in final semi-major axis and eccentricity. This implied that the guidance architecture necessitated a more proper energy management scheme. The idea of implementing a tunable parameter, to affect the starting time/location of the orbital insertion maneuver, was explored. Maneuver ignition time then became a function of how strongly the ascent maneuver outperformed its original apoapsis target. While this method had seemed more feasible than other options explored, the tunable parameter described above was very sensitive to vehicle mass and configuration changes, as well as orbital insertion target changes.

Then, the idea was explored to monitor CPS required velocity, v_{go}, re-calculate it throughout the coast phase of the vehicle, and compare this velocity to the estimated delta-v capability (v_{cap}) on-board the vehicle. If at any time the v_{go} required from CPS to complete the burn matched the estimated v_{cap} on-board the vehicle (to some specified tolerance), then the vehicle would instantaneously initiate the burn at that time. This method worked well and proved stable, and further simulation showed the added potential benefit of centering the burn on the specified guidance target, rather than initiating the burn right at the start of the target. This required knowledge of when this optimal impulsive point, in the future, would occur. An iterative Newton-Raphson scheme was employed to detect when the burn should begin, at some future time. This led to a more acceptable semi-major axis dispersion in the orbit.

Thus the explicit, path-adaptive guidance algorithm referred to as SxS (Simple cross-product Steering) was born: a combination of modifications to the CPS algorithm with an ignition routine driven by estimated vehicle energy capacity. Using SxS for the orbital insertion maneuver, combined with an open-loop vs. altitude scheme for the ascent maneuver, provided desirable results that allowed the vehicle architecture to hit a target orbit within specifications, even under the influence of several vehicle and dynamics dispersions[4].

This paper is laid out as follows: the Algorithm Description section outlines the construction of the SxS target frame, derives the velocity-to-go routine chosen for SxS, and explores the algorithm's ignition predictor routine; the Performance section covers preliminary simulation results for SxS employed in a martian environment; the Notes on Mechanization section covers some caveats for implementation of SxS in simulation code (and flight software); and the Conclusion and Appendix contains provides closing notes and analytical derivations not covered in the main content of the paper, respectively.

ALGORITHM DESCRIPTION

SxS analytically solves a problem that can be framed in the following manner: Given a current estimated state, $\vec{X}_0 = [\vec{r}_0, \vec{v}_0]$, under an exo-atmospheric spherical-gravity environment, what is the velocity vector required, $\vec{v}_{go}$, that is sufficient and necessary to guide the vehicle to a some desired inertial target state, $\vec{X}_f = [\vec{r}_f, \vec{v}_f]$?

Target Frame

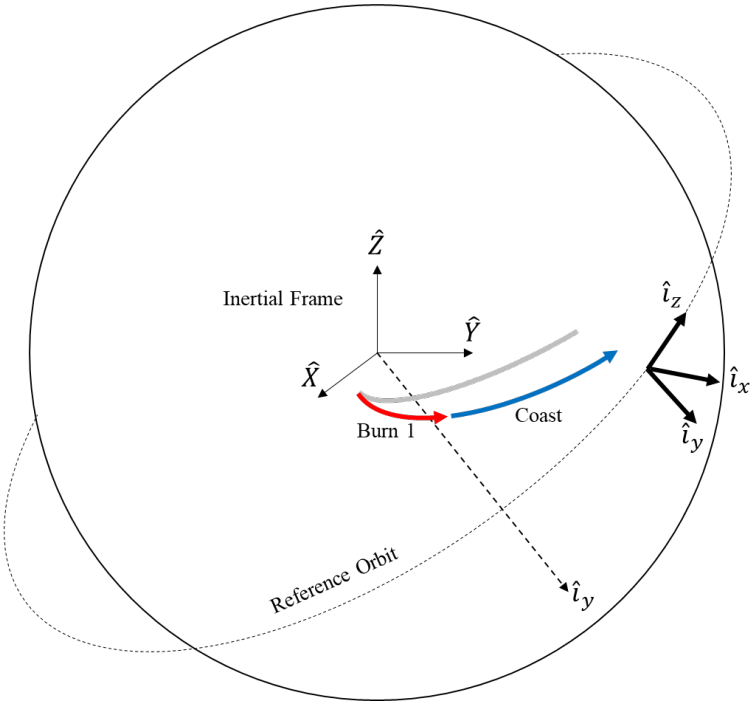


Figure 1. Guidance Target Frame Representation

As common in several modern path-adaptive guidance algorithms, the target orbit (and thus the target state of the algorithm) is not fully inertially constrained (but is constrained in-plane), in order to avoid algorithm infeasibilities[1]. SxS also assumes that the orbital insertion maneuver is near impulsive (i.e., the target orbital period is significantly longer than the total time of the maneuver). Instead of a standard inertial frame, a representative guidance target frame is constructed, the $\hat{\imath}_x\hat{\imath}_y\hat{\imath}_z$ frame, which shares similarities to Powered Explicit Guidance's (PEG) guidance target frame[1]. The guidance target frame is visualized in Figure 1, and the derivation of this target frame is discussed throughout this section. In order to construct this frame, SxS takes as input the following scalar values as target parameters, which are unchanging throughout the algorithm's employment:

- a_d – target semi-major axis
- Ω_d – target orbital right ascension of the ascending node (RAAN)
- i_d – target orbital inclination

First, the unit vector $\hat{\imath}_y$, which represents the out-of-plane component of the target frame, is constructed in the following fashion:

$$\hat{\imath}_y = \begin{bmatrix} -\sin(\Omega_d)\sin(i_d) \\ \cos(\Omega_d)\sin(i_d) \\ -\cos(i_d) \end{bmatrix} \tag{1}$$

This is synonymous to the anti-parallel direction to the angular momentum vector of the target orbit. The reason that the $\hat{\imath}_y$ unit vector was chosen to be anti-parallel rather than parallel to the angular momentum vector was to keep convention of previous algorithms that use similar manifestations of the $\hat{\imath}_x\hat{\imath}_y\hat{\imath}_z$ frame, such as PEG[1].

Next is the calculation of the radial component of the target frame $\hat{\imath}_x$. The radial component of the target frame is represented by the current position vector projected onto the target orbital plane. The reader may note that this is commensurate to the derivation of PEG, and this projection is required to maintain orthogonality. It can be shown that assigning $\hat{\imath}_x = \hat{r}$, instead of Equation (2), leads to the same performance of the algorithm, because of the fact that only $\hat{\imath}_z$ will be used for velocity-to-go computations described in the next section.

$$\hat{\imath}_x = \frac{\vec{r} - (\vec{r} \cdot \hat{\imath}_y)\hat{\imath}_y}{\left|\vec{r} - (\vec{r} \cdot \hat{\imath}_y)\hat{\imath}_y\right|} \tag{2}$$

Lastly, the final downrange component of the target frame, $\hat{\imath}_z$, can be calculated as follows:

$$\hat{\imath}_z = \frac{\hat{\imath}_x \times \hat{\imath}_y}{\left|\hat{\imath}_x \times \hat{\imath}_y\right|} \tag{3}$$

Thus the combination of Equations (1), (2), and (3) create the guidance target frame, or the $\hat{\imath}_x\hat{\imath}_y\hat{\imath}_z$ frame.

Velocity-to-Go

One of the baseline assumptions of SxS is that the algorithm targets a final state that lies directly on the apse-line of the final target orbit, even if the vehicle does not currently align on the apse-line of its current orbit. The reader may note that this assumption is not commensurate to optimality; however, that is by design, and is a critical component of the energy management strategy employed by SxS. This applied apse-line shift can also be viewed as nulling the r-dot ($\dot{r}$) component of velocity, and this velocity nulling technique requires energy. The energy management scheme employed for this algorithm is realized by purposefully performing a non-optimal maneuver (expelling energy into an apse-line shift). This apse-line assumption, paired with the impulsivity assumption discussed in the last section, is what allows a target state to be defined with only three scalar inputs: semi-major axis, inclination, and RAAN.

The following outlines the creation of the velocity-to-go vector, or $\vec{v}_{go}$ vector, which is made analytically possible by the assumptions outlined above. Firstly, because the burn is assumed to be impulsive, the magnitude of the desired velocity after the maneuver has been completed (v_d) can be calculated using the Vis-Viva equation:

$$v_d = \sqrt{\mu\left(\frac{2}{r} - \frac{1}{a_d}\right)} \tag{4}$$

Where μ is the planetary gravitational parameter, r is the vehicle's current radius magnitude, and a_d is the desired targeted semi-major axis. This magnitude can then be used to construct the desired velocity vector, $\vec{v}_d$:

$$\vec{v}_d = v_d\hat{\imath}_z \tag{5}$$

Where $\hat{\imath}_z$ is the third axis of the guidance target frame expressed in the previous section. Note that the desired velocity vector is only in the downrange ($\hat{\imath}_z$) direction, and does not have any components in the out-of-plane ($\hat{\imath}_y$) or radial ($\hat{\imath}_x$) direction. This aligns with the assumption that the final target state will lie on the apse-line of the target orbit, i.e. there will be no $\dot{r}$ velocity term at maneuver cutoff. Therefore, $\vec{v}_d$ can be described in words as the desired inertial velocity vector that lies in the target orbit plane, at current position, whose SMA matches the SMA from the guidance target. Thus, the velocity to go, or $\vec{v}_{go}$, can be defined as the difference between the current vehicle inertial velocity and the desired final inertial velocity:

$$\vec{v}_{go} = \vec{v}_d - \vec{v} \tag{6}$$

Thus the closed-loop commanded velocity-to-go equation has been fully constructed.

Iterative-Based Ignition Prediction

The need for awareness of the proper time to ignite the second stage has been discussed in the introduction of this paper. This section discusses the iterative scheme that is used to predict when the burn should be initiated. During coast phase of flight, at some time t_{ig} after the current mission elapsed time t, the second stage burn should ignite at $t_b = t + t_{ig}$. This time should be such that the velocity required to complete the burn, $v_{go} = |\vec{v}_{go}|$, is equal to estimated vehicle capability of the second stage, v_{cap}, which is calculated by the rocket equation. There is no direct closed-form solution to solve for t_{ig}, but it can be iterated on instead in a Newton-Raphson fashion. Because of the coasting nature of the vehicle, and the simplicity of the guidance algorithm's velocity targeting scheme, this is a well-behaved root-finding problem. Before the reader goes forth in this section, it is beneficial to take a look at the behavior of $|\vec{v}_{go}(t)|$ over time for this algorithm, as the vehicle coasts to apoapsis on its ballistic trajectory. Figure 2 displays a typical v_{go} curve for three different vehicle configurations – a nominal case, an underperforming first stage case (less impulse than expected was provided over the course of the open-loop ascent burn), and an outperforming first stage case (more impulse than expected was provided over the course of the open-loop ascent burn). It is noted that Figure 1 is not to scale.

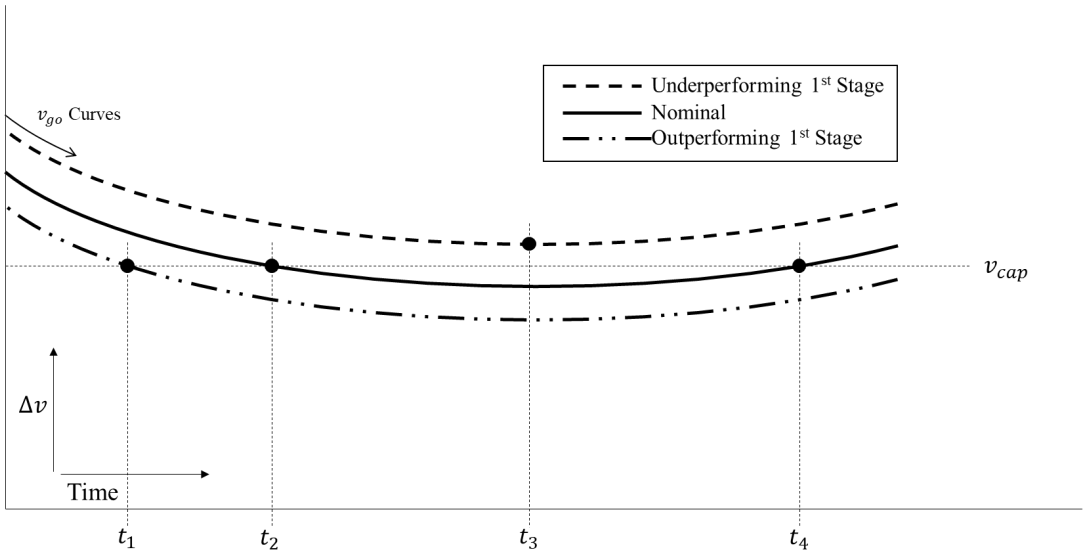


Figure 2. Typical v_{go} Curve Behaviors

It can be easily shown that the lowest-energy point to perform this maneuver is at apoapsis, which is when the v_{go} curve achieves a minimum, represented as time t_3 in Figure 2. To ensure that there is a crossing of v_{go} and v_{cap}, the vehicle and corresponding nominal trajectory must be designed such that even with

intrinsic solid motor performance variation, the vehicle is capable of meeting the target orbit requirements. Therefore, the ideal vehicle sizing should nominally have energy in excess of that required to hit the target orbit under nominal conditions; leading to two crossings of v_{go} and v_{cap}, one on either side of apoapsis (see times t_2 and t_4 in Figure 1). If the first stage significantly underperforms, the vehicle may never have enough energy to meet v_{go} requirements, even at apoapsis (see time t_3 in Figure 2). If the first stage significantly outperforms its target, this may push the burn initialization to be too early (see time t_1 in Figure 2) and may breach other restrictions (i.e. minimum burn altitude or minimum coast time). These issues all pertain to vehicle design and are mission specific; therefore they are not discussed further in this paper.

First, the root equation $f(t)$ is configured:

$$f(t_{ig}) = v_{go}(t_{ig}) - v_{cap} \tag{7}$$

Where the ideal solution is when $f(t) = 0$. Using the Newton-Raphson scheme, t_{ig} can be iterated on in the following fashion:

$$t_{ig_{k+1}} = t_{ig_k} - \frac{f(t_{ig})}{f'(t_{ig})} = t_{ig_k} - \frac{v_{go}(t_{ig}) - v_{cap}}{\frac{\delta v_{go}}{\delta t_{ig}}} \tag{8}$$

Through simulation, and because of the well-behaved manner of the velocity targeting routine, it has been shown to be sufficient to initialize t_{ig} to 0 on the first iteration. Note that v_{cap} does not change in time. Then, once $v_{go}(t_{ig} = 0)$ is calculated, the partial $\frac{\delta v_{go}}{\delta t}$ is calculated and a new t_{ig} value is calculated using Equation 8. On the next iteration, $v_{go}(t_{ig})$ is once again calculated but this time using the new position $\vec{r}(t_{ig})$ and velocity $\vec{v}(t_{ig})$, and the new partial is calculated at this predicted future state as well. Either an on-board propagator, or Keplerian equations (i.e. calculating position and velocity in Lagrangian functions[5]), can be used to solve for the future state $\vec{X}(t_{ig}) = [\vec{r}(t_{ig}), \vec{v}(t_{ig})]$.

It is crucial to note that the partial $\frac{\delta v_{go}}{\delta t}$ is in reality the partial of the magnitude of the vector $\vec{v}_{go}$ with respect to time.

$$\frac{\delta v_{go}}{\delta t} = \frac{\delta |\vec{v}_{go}|}{\delta t} = \hat{v}_{go} \cdot \frac{\delta \vec{v}_{go}}{\delta t} \tag{9}$$

The partial of the vector $\vec{v}_{go}$ with respect to time is then clearly essential for this magnitude partial, and is fully derived in the appendix. The final form of this partial is arranged as such:

$$\frac{\delta \vec{v}_{go}}{\delta t} = -\frac{\mu}{v_d r^2}[\hat{r} \cdot \vec{v}]\hat{\imath}_z - v_d \frac{\vec{v} \cdot \hat{\imath}_z}{\vec{r} \cdot \hat{\imath}_x}\hat{\imath}_x + \frac{\mu}{r^3}\vec{r} \tag{10}$$

When present in the form of Equation (10), this partial is separated into three unique components, which can be intuitively spoken to in the following points:

- Radial Distance Effect – $\vec{v}_{go}$ will decrease in time (in the negative $\hat{\imath}_z$ direction) as an inverse function of radial distance from the planet, and as a function of how much of the velocity vector is currently in the radial direction ($\hat{r} \cdot \vec{v}$). This is why v_{go} incurs a minimum at apoapsis.
- Target Frame Rotation – The downrange ($\hat{\imath}_z$) direction of the target frame will change over time in the negative radial direction ($\hat{\imath}_x$), which makes sense intuitively. This change is a function of the current velocity in the downrange direction ($\vec{v} \cdot \hat{\imath}_z$) and the inverse of how far out of plane the vehicle currently is ($\vec{r} \cdot \hat{\imath}_x$). Studies have not been conducted, and convergence is not guaranteed, for maneuvers that are initialized significantly far out of plane.

- Gravity – $\vec{v}_{go}$ will change over time proportional to how $\vec{v}$ changes over time, which, when in coast phase, is modelled as a simple spherical gravitational model. This partial could be modified in higher-fidelity implementations to include spherical harmonics partials.

PERFORMANCE

Figure 3 below shows the results of employing SxS in a standard martian environment in three different scenarios. In the simulated cases shown in Figure 3, the I_{sp} of the first stage engine was varied by $\pm 0.05\%$ in both directions, yet the vehicle flies the same open-loop ascent table in all three simulations. As seen in the figure, the case with the outperforming first stage significantly overshoots both its apoapsis and periapsis targets at the end of the ascent burn. Because of this, SxS predicts a necessity to initiate the second stage burn earlier than usual to expel more energy into $\dot{r}$ nulling. As stated before, and shown in Figure 2, even the nominal case still requires $\dot{r}$ nulling. This is an intentional effect to ensure that the nominal vehicle has room for dispersion in both the underperforming and the outperforming directions.

Similarly, for the underperforming case, SxS necessitates a shift in stage two ignition time to compensate for energy differences from the first stage. In the underperforming case, stage two fires later (closer to apoapsis) than the nominal case; this is because more of the stage two energy is required directly for the orbital maneuver and less energy is expelled into an $\dot{r}$ nulling. If approximate vehicle dispersions are known early in the design phase, the trajectory engineer can plan ahead and ensure that the most underperforming expected case still allows the vehicle to hit a nominal orbit by employing SxS.

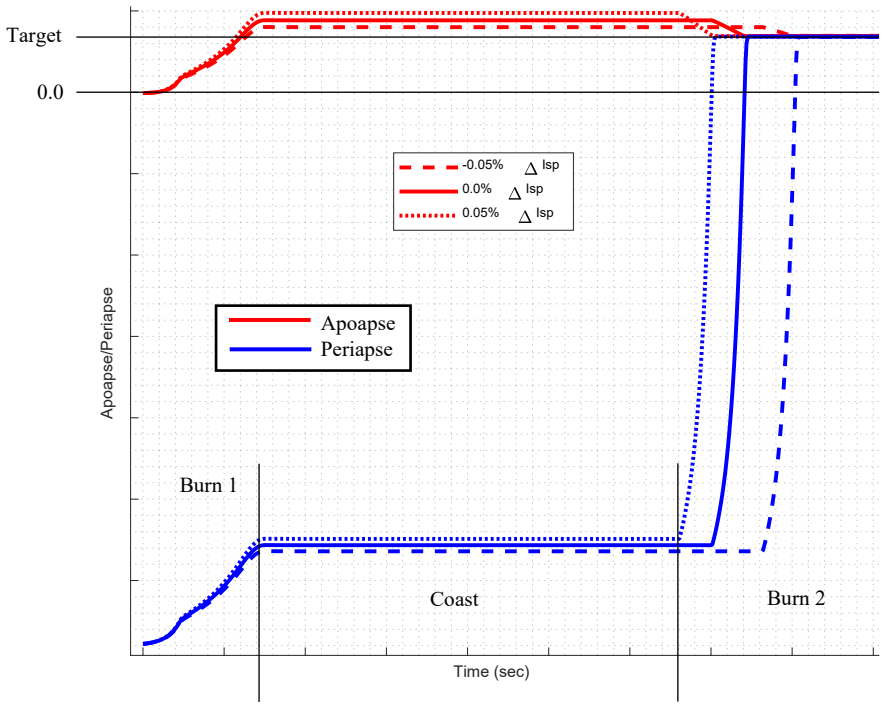


Figure 3. Apoapsis & Periapsis Height Over Time, Varying Stage 1 I_{sp}

NOTES ON MECHANIZATION

For the iteration framework described in the previous section, initializing $t_{ig} = 0$ has been shown in simulation to ensure that the Newton-Raphson scheme converges on the first root. As noted above and as shown in Figure 2, in cases where the first stage significantly underperforms, the vehicle may never have enough energy to achieve the target orbit. To check for this case, Keplerian equations (or propagation) should be used to check the energy requirement at apoapsis; if this energy still exceeds that which is estimated to be available on-board the vehicle, then the iteration scheme can be skipped and the burn should occur at apoapsis. Even if this check is not performed for underperforming cases, the iteration scheme will not converge if a root is not findable within the local vicinity of this initial guess of $t_{ig} = 0$; therefore, burning at apoapsis should be enabled if the iterative scheme does not converge.

The near impulsive nature of the orbital insertion maneuver theoretically requires that the maneuver be initiated instantaneously at the moment of nodal crossings of the target orbit and the current coasting trajectory. Due to dispersions in the vehicle performance of the ascent maneuver, and the specialized ignition scheme described above, this may lead to SxS providing a solution that requires orbital insertion ignition to occur off of this nodal crossing. It has been shown in simulation that this difference leads to mainly a larger dispersion in orbital insertion RAAN when launching near the equator, whereas orbital insertion inclination remains at a very tight accuracy. This dispersion is heavily dependent on vehicle performance and should be left to the engineer implementing this algorithm to determine if the dispersion is within mission tolerances.

Also, because of inherent motor and dynamics dispersions, it may still be seen that the Δv capacity of the orbital insertion stage of the vehicle may exceed the required velocity of the maneuver ($|\vec{v}_{go}|$) calculated by SxS, even with this energy management ignition routine. If the vehicle has the capacity to overshoot this desired orbital target, and does so, this will correspond to an instantaneous directional switch of $\vec{v}_{go}$**.** This may lead to undesired vehicle behavior. This dilemma can be mitigated in the algorithm's mechanization by supplementing the algorithm with a conditional statement, expressed as follows: if the magnitude of $\vec{v}_{go}$ ever dips below a specified threshold, then the vehicle's attitude is held constant through the rest of the maneuver. This threshold can be set relatively low (on the order of meters per second), but is dependent on the vehicle controllability characteristics. However, the reader may note that this fail-safe may lead to orbital dispersions at the end of the maneuver. A simple first-order analysis can be performed (and is left as an exercise for the reader) to show the effect of solid motor dispersions on final orbital parameters (i.e., semi-major axis dispersions as a function of I_{sp} changes).

Lastly, it is not required that the iterative scheme presented in the last section be called each high-rate frame in the guidance subsystem. A wrapper could be implemented that performs this iterative calculation at a slower rate, and steps down the value of t_{ig} using time step counters on the higher rate frame calls. Then, the previous value of t_{ig} can be used to prime the next iteration on the next major low-rate frame. Figure 4 below shows a typical iteration count of an example Mars ascent coast scenario for calling this iteration scheme at 10 second intervals. As seen in Figure 4, the first time the predictor is called requires 6 iterations to converge, because of the inaccuracy of the initial guess for t_{ig}. After, only 3 iterations are required during the coast phase. For this example, a propagator using the Super-G algorithm[6] was used rather than Lagrangian or Keplerian functions for calculation of $\vec{r}(t_{ig})$ and $\vec{v}(t_{ig})$. Future work would be to test if a specific propagation technique leads to a lower iteration count.

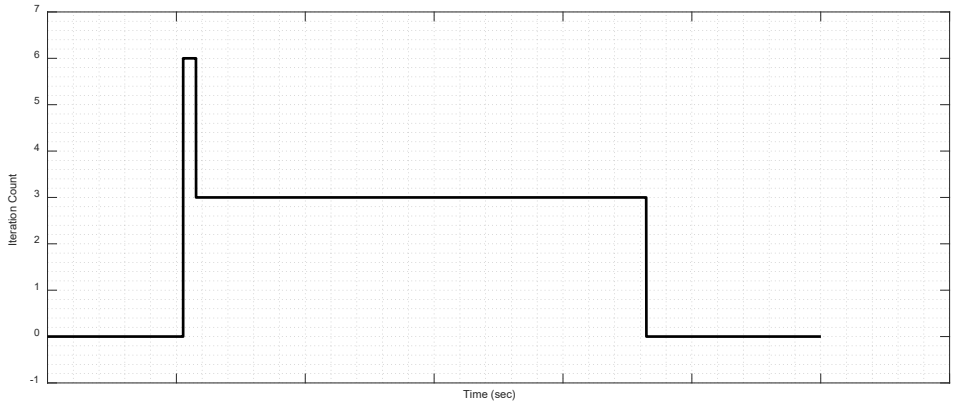


Figure 4. Iteration Count of Predictor

CONCLUSION

A simplified guidance algorithm for in-space solid-propellant maneuvers is presented. The algorithm is dependent on a specified set of targeting parameters, and the allowance of a variable starting ignition time. The algorithm takes roots in Cross-Product Steering, however the velocity targeting routine was modified to incorporate an $\dot{r}$ nulling effect for energy management. The realization of the guidance target frame has been presented, as well as the new velocity targeting routine derivation. The employed method for predicting motor ignition starting time has been derived fully in the appendix.

As work continues on NASA's MAV MSR, this algorithm may undergo further modifications to include higher-order effects. This algorithm had also been briefly tested during NASA's internal VIPER lander development[7], and initial results showed that this algorithm had promising potential for a solid-propellant lunar braking scenario as well. The author plans to continue development and implementation testing of this algorithm throughout future analysis cycles of MAV MSR. Readers (and engineers who attempt to implement this algorithm) are encouraged to contact the author with questions about implementation or recommendations for future work.

ACKNOWLEDGEMENTS

The reader would like to acknowledge the following engineers at NASA's Marshall Spaceflight Center: Greg Dukeman (EV42), for providing insurmountable access to heritage algorithm development and knowledge; Dane Erickson (EV42), for his excellent systems engineering perspective by working with the author to brainstorm energy management techniques for this algorithm, as well as providing a high-accuracy 6-DOF simulation as a testbed for the algorithm; Evan Anzalone (EV42), for helping with paper development and providing mentorship and support through the development process; and Naeem Ahmad (EV42), for acting as the author's "math checker" through some of the trickier portions of derivation.

REFERENCES

[1] R. L. McHenry. "Space Shuttle Ascent Guidance, Navigation, and Control", *The Journal of the Astronautical Sciences*, Vol. XXVII, No. 1, pp. 1-38, 1979

[2] Si-Yuan Chen, Qun-Li Xia. "A Multiconstrained Ascent Guidance Method for Solid Rocket-Powered Launch Vehicles", *International Journal of Aerospace Engineering,* Volume 2016

[3] Richard H. Battin. "An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition", *American Institute of Aeronautics & Astronautics,* 1999

[4] Anzalone, Erickson, Everett, Powers. "Guidance and Navigation Challenges for a Mars Ascent Vehicle", *To Be Published in IEEE Aerospace*

[5] Bate, Mueller, White. "Fundamentals of Astrodynamics"

[6] Pinson, Von der Porten, Ahmad. "Space Launch System Guidance Description for Artemis I", NASA Marshall Space Flight Center, 2019

[7] Anzalone, Braden, Ahmad, Everett, Miller. "Guidance and Navigation Trades for the Lunar Pallet Lander", AAS 19-091, 2019

APPENDIX: SOLVING FOR VELOCITY-TO-GO 1ST ORDER PARTIAL

Note, the derivative of a magnitude of a vector $\vec{x}$ with respect to time is as follows:

$$\frac{\delta x}{\delta t} = \hat{x} \cdot \frac{\delta \vec{x}}{\delta t} \tag{A1}$$

For SxS, the required partial for the iterative ignition scheme is $\frac{\delta v_{go}}{\delta t}$, which can be presented using Equation (A1) as:

$$\frac{\delta v_{go}}{\delta t} = \hat{v}_{go} \cdot \frac{\delta \vec{v}_{go}}{\delta t} \tag{A2}$$

Note from Equations (6) and (5) that:

$$\vec{v}_{go} = \vec{v}_d - \vec{v} = v_d \hat{\imath}_z - \vec{v} \tag{A3}$$

Using the product rule for derivation, the derivative of Equation (A3) can be represented as follows:

$$\frac{\delta \vec{v}_{go}}{\delta t} = \frac{\delta v_d}{\delta t} \hat{\imath}_z + v_d \frac{\delta \hat{\imath}_z}{\delta t} - \frac{\delta \vec{v}}{\delta t} \tag{A4}$$

Now, the derivation of the individual components of Equation (A4) above is required. Starting with the simplest term, assuming a spherical gravity model:

$$\frac{\delta \vec{v}}{\delta t} = \vec{a} = -\frac{\mu}{r^3} \vec{r} \tag{A5}$$

Plugging Equation (A5) back into Equation (A4):

$$\frac{\delta \vec{v}_{go}}{\delta t} = \frac{\delta v_d}{\delta t} \hat{\imath}_z + v_d \frac{\delta \hat{\imath}_z}{\delta t} + \frac{\mu}{r^3} \vec{r} \tag{A6}$$

Next, to solve for the scalar $\frac{\delta v_d}{\delta t}$, it is advantageous to split this into two separate partials using the chain rule:

$$\frac{\delta v_d}{\delta t} = \frac{\delta v_d}{\delta r} \frac{\delta r}{\delta t} = \frac{\delta v_d}{\delta r} \left[\hat{r} \cdot \frac{\delta \vec{r}}{\delta t} \right] = \frac{\delta v_d}{\delta r} [\hat{r} \cdot \vec{v}] \tag{A7}$$

Now, solving for $\frac{\delta v_d}{\delta r}$, starting with Equation (4):

$$\frac{\delta v_d}{\delta r} = \frac{1}{2 v_d} \left(-\frac{2\mu}{r^2} \right) = -\frac{\mu}{v_d r^2} \tag{A8}$$

Plugging Equation (A8) back into Equation (A7), and then back into Equation (A6):

$$\frac{\delta \vec{v}_{go}}{\delta t} = -\frac{\mu}{v_d r^2} [\hat{r} \cdot \vec{v}] \ \hat{\imath}_z + v_d \frac{\delta \hat{\imath}_z}{\delta t} + \frac{\mu}{r^3} \vec{r} \tag{A9}$$

The last piece needed to be solved for, and the most tedious, is $\frac{\delta \hat{\imath}_z}{\delta t}$. Before solving, it is important to note that $\hat{\imath}_z$ can be simplified from Equation (3) as the following:

$$\hat{\imath}_z = \frac{\left[\vec{r} - (\vec{r} \cdot \hat{\imath}_y)\hat{\imath}_y\right] \times \hat{\imath}_y}{\left|\left[\vec{r} - (\vec{r} \cdot \hat{\imath}_y)\hat{\imath}_y\right] \times \hat{\imath}_y\right|} = \frac{\vec{r} \times \hat{\imath}_y}{\left|\vec{r} \times \hat{\imath}_y\right|} \tag{A10}$$

The derivative of the quotient and dividend, used later, are:

$$\frac{\delta}{\delta t}\left[\vec{r} \times \hat{\imath}_y\right] = \frac{\delta \vec{r}}{\delta t} \times \hat{\imath}_y + \vec{r} \times \frac{\delta \hat{\imath}_y}{\delta t} = \frac{\delta \vec{r}}{\delta t} \times \hat{\imath}_y = \vec{v} \times \hat{\imath}_y \tag{A11}$$

$$\frac{\delta}{\delta t}\left[\left|\vec{r} \times \hat{\imath}_y\right|\right] = \frac{\vec{r} \times \hat{\imath}_y}{\left|\vec{r} \times \hat{\imath}_y\right|} \cdot \left[\vec{v} \times \hat{\imath}_y\right] \tag{A12}$$

Now, solving for $\frac{\delta \hat{\imath}_z}{\delta t}$, by taking the derivative of Equation (A10), using the results of Equations (A11) and (A12):

$$\frac{\delta \hat{\imath}_z}{\delta t} = \frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} - \frac{\vec{r} \times \hat{\imath}_y}{\left|\vec{r} \times \hat{\imath}_y\right|^2}\left[\frac{\vec{r} \times \hat{\imath}_y}{\left|\vec{r} \times \hat{\imath}_y\right|} \cdot \left[\vec{v} \times \hat{\imath}_y\right]\right] \tag{A13}$$

Equations (A14) through (A25) below show the simplification process of Equation (A13), to prep the equation form for further readability. Note, at this point forward, Equation (A13) could be plugged back into equation (A9) and the partial would be complete for implementation; however, what's presented below is a simplification that puts this portion of the partial into a much more intuitive form.

$$\frac{\delta \hat{\imath}_z}{\delta t} = \frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} - \frac{\vec{r} \times \hat{\imath}_y}{\left|\vec{r} \times \hat{\imath}_y\right|}\left[\frac{\vec{r} \times \hat{\imath}_y}{\left|\vec{r} \times \hat{\imath}_y\right|^2} \cdot \left[\vec{v} \times \hat{\imath}_y\right]\right] \tag{A14}$$

$$\frac{\delta \hat{\imath}_z}{\delta t} = \frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} - \left[\frac{\left(\vec{r} \times \hat{\imath}_y\right) \cdot \left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|^2}\right]\hat{\imath}_z \tag{A15}$$

$$\frac{\delta \hat{\imath}_z}{\delta t} = \frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} - \left[\frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} \cdot \frac{\left(\vec{r} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|}\right]\hat{\imath}_z \tag{A16}$$

$$\frac{\delta \hat{\imath}_z}{\delta t} = \frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} - \left[\frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} \cdot \hat{\imath}_z\right]\hat{\imath}_z \tag{A17}$$

Further simplifying the term within the brackets in Equation (A17):

$$\frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} \cdot \hat{\imath}_z = \frac{1}{\left|\vec{r} \times \hat{\imath}_y\right|}\left(\vec{v} \times \hat{\imath}_y\right) \cdot \hat{\imath}_z \tag{A18}$$

Using the scalar triple product identity:

$$\hat{\imath}_z \cdot \left(\vec{v} \times \hat{\imath}_y\right) = \vec{v} \cdot \left(\hat{\imath}_y \times \hat{\imath}_z\right) = \vec{v} \cdot \hat{\imath}_x \tag{A19}$$

Plugging Equation (A19) back into Equation (A18), and then Equation (A18) into Equation (A17):

$$\frac{\delta \hat{\imath}_z}{\delta t} = \frac{\left(\vec{v} \times \hat{\imath}_y\right)}{\left|\vec{r} \times \hat{\imath}_y\right|} - \frac{\vec{v} \cdot \hat{\imath}_x}{\left|\vec{r} \times \hat{\imath}_y\right|}\hat{\imath}_z \tag{A20}$$

Now, for further simplification of the term on the left-hand side of Equation (A20) using the vector triple product identity:

$$\vec{v} \times \hat{\imath}_y = \vec{v} \times (\hat{\imath}_z \times \hat{\imath}_x) = (\vec{v} \cdot \hat{\imath}_x)\hat{\imath}_z - (\vec{v} \cdot \hat{\imath}_z)\hat{\imath}_x \tag{A21}$$

Plugging Equation (A21) back into equation (A20):

$$\frac{\delta \hat{\imath}_z}{\delta t} = \frac{\vec{v} \cdot \hat{\imath}_x}{\left|\vec{r} \times \hat{\imath}_y\right|}\hat{\imath}_z - \frac{\vec{v} \cdot \hat{\imath}_z}{\left|\vec{r} \times \hat{\imath}_y\right|}\hat{\imath}_x - \frac{\vec{v} \cdot \hat{\imath}_x}{\left|\vec{r} \times \hat{\imath}_y\right|}\hat{\imath}_z \tag{A22}$$

After crossing out like terms, this simplifies to the more elegant form of:

$$\frac{\delta \hat{\imath}_z}{\delta t} = -\frac{\vec{v} \cdot \hat{\imath}_z}{\left|\vec{r} \times \hat{\imath}_y\right|}\hat{\imath}_x \tag{A23}$$

There is one last simplification that can be performed. For eliminating the absolute value sign from this term entirely, the vector triple product rule is used once again:

$$\vec{r} \times \hat{\imath}_y = \vec{r} \times (\hat{\imath}_z \times \hat{\imath}_x) = (\vec{r} \cdot \hat{\imath}_x)\hat{\imath}_z - (\vec{r} \cdot \hat{\imath}_z)\hat{\imath}_x = (\vec{r} \cdot \hat{\imath}_x)\hat{\imath}_z \tag{A24}$$

Because $\hat{\imath}_z$ is of unit length, this term and the absolute value terms in Equation (A23) negate. Therefore we are left with the most intuitive form:

$$\frac{\delta \hat{\imath}_z}{\delta t} = -\frac{\vec{v} \cdot \hat{\imath}_z}{\vec{r} \cdot \hat{\imath}_x}\hat{\imath}_x \tag{A25}$$

Finally, plugging Equation (A25) back into Equation (A9):

$$\frac{\delta \vec{v}_{go}}{\delta t} = -\frac{\mu}{v_d r^2}[\hat{r} \cdot \vec{v}]\hat{\imath}_z - v_d \frac{\vec{v} \cdot \hat{\imath}_z}{\vec{r} \cdot \hat{\imath}_x}\hat{\imath}_x + \frac{\mu}{r^3}\vec{r} \tag{A26}$$

Alas, the partial $\frac{\delta \vec{v}_{go}}{\delta t}$ has been derived and simplified, representing how the velocity targeting routine changes as a function of time.