Performance Analysis of a Dual Quaternion Guidance Algorithm applicable during Lunar Approach with a Hazard Avoidance Maneuver

Javier A. Doll[1,*], Samuel T. Wagner[1,†], Matthew P. Fritz[1,‡], Jiann-woei Jang[1,§], Jeanette M. Harper[3,**], Kyle W. Smith[1,††], Behçet Açikmeşe[2,‡‡], Samuel M. Pedrotty[2,§§], Gavin F. Mendeck[2,***]

[1]*The Charles Stark Draper Laboratory*

[2]*NASA Johnson Space Center*

[3]*Odyssey Space Research*

There is currently a need for advanced guidance and targeting algorithms that can provide real-time trajectories in the presence of state and vehicle constraints during powered descent. The Safe & Precise Landing Integrated Capabilities Evolution (SPLICE) program aims to mature Hazard Detection (HD) technology along with advanced navigation and guidance algorithms. This paper will report the overall performance of the SPLICE Dual Quaternion Guidance (DQG) algorithm which is a 6-Degree-of-Freedom (6-DOF) convex optimization-based guidance algorithm that can enforce multiple vehicle and state constraints required for Hazard Detection Lidar (HDL) scans during the approach phase. DQG was tested in a closed-loop lunar descent simulation with realistic HDL sensing constraints in a Monte Carlo assessment of 1000 dispersed runs.

I. Introduction

The Safe & Precise Landing Integration Capability Evolution (SPLICE) program represents a groundbreaking NASA initiative aimed at revolutionizing our landing capabilities for space exploration. Through the fusion of advanced technologies and methodologies, SPLICE seeks to achieve precisely targeted landings, reduce mission risks, and maximize the scientific potential of future missions. By integrating cutting-edge navigation systems, hazard detection and avoidance technologies, high-resolution terrain mapping techniques, and real-time sensor feedback, SPLICE aims to enhance the reliability and performance of descent and landing systems [1].

Scientifically interesting landing sites are often surrounded by hazardous obstacles, motivating the need for the development of a descent and landing system capable of detecting hazards, identifying safe sites, and capable of re-targeting to these safe sites with precision. During the Apollo program, this was accomplished through the design of approach trajectories that allowed the Lunar Module Commander and Pilot to have visibility to the landing site leading up to about 5 seconds prior to terminal descent. This allowed the Commander to re-direct the Lunar Guidance

* Senior Member of Technical Staff
† Member of Technical Staff
‡ Principal Member of Technical Staff
§ Principal Member of Technical Staff
** Engineer
†† Senior Member of Technical Staff
‡‡ IPA from University of Washington, AIAA Associate Fellow
§§ SPLICE Chief Engineer
*** SPLICE Project Manager

Computer to land at a visually verified safe site [2]. To achieve safe and precise landing autonomously instead of manually, hazard detection technology and advanced guidance algorithms are required to reliably produce a trajectory that adheres to imposed constraints and can do so in a fuel-efficient manner. These constraints may vary to ensure that hazard detection sensor suites are effective, that the vehicle can fly the trajectory, and that the vehicle will reach its destination safely and accurately.

Research and development on convex optimization strategies applied to powered descent guidance, have been elegantly utilized to solve the optimal guidance problem in the presence of vehicle and state constraints. Lossless convexification techniques can be utilized to resolve nonconvexities in desired constraints and 3-Degree-of-Freedom (3DOF) algorithms have been designed using lossless convexification [3] [4] [5]. These results were successfully demonstrated on-board in a real-time terrestrial test flight [6]. Despite the warranted success of these algorithms, 3-DOF algorithms are limited in their ability to adequately handle hazard detection constraints. These hazard detection constraints feature a coupling between translational (position and velocity) and rotational (attitude and angular velocity) states. Successive convexification techniques were then utilized to support the development of 6-Degree-of-Freedom (6-DOF) convex optimization based guidance algorithms, which were also capable of enforcing state-triggered constraints to be active during specified flight envelopes [7] [8]. The development of state-triggered constraints has been significant for the hazard detection problem, because these critical yet restrictive constraints are typically only required for a specific phase of flight. The use of a dual quaternion state representation was then introduced in a 6-DOF convex optimization guidance algorithm, which allowed for the combination of translation and rotational state information to be represented in a single term [9]. This simplification proved to be beneficial for the derivation of Hazard Detection Lidar (HDL) sensor scanning constraints, and their use in a successive convexification algorithm.

The SPLICE program aims to mature the software readiness level of the Dual Quaternion Guidance (DQG) algorithm to solve the constrained powered descent guidance problem. Building off the research done at the University of Washington, a DQG algorithm was implemented in the Guidance, Navigation, and Control (GNC) system integrated in the SPLICE payload. This payload operated in an open-loop fashion on-board the Blue Origin New Shepard vehicle during a sub-orbital test flight, where DQG was able to generate several guidance trajectories in real-time [10]. Since then, focus has been placed towards the continued development and testing of the DQG algorithm in closed-loop lunar descent simulations. Closed-loop Monte-Carlo analysis of lunar descent profiles with randomly chosen divert locations were studied in simulation as part of a technology cooperative agreement between NASA and Blue Origin [11].

The work presented in this manuscript will largely cover the results from the SPLICE GNC Preliminary Design Review which was held earlier this year. The SPLICE DQG algorithm was tested in a closed-loop Trick simulation with perfect navigation, realistic hazard detection state constraints, and divert scenarios.

A. Acronyms

3-DOF Three Degrees-of-Freedom
6-DOF Six Degrees-of-Freedom
BSOCP Behcet Second Order Program
CDF Cumulative Distribution Function
CLPS Commercial Lunar Payload Services
CPRS Custom Parser
DEM Digital Elevation Map
DQG Dual Quaternion Guidance
ENU East North Up
GNC Guidance, Navigation, and Control
GTSG Guidance Target State Generator
HD Hazard Detection
HDL Hazard Detection Lidar
JEOD JSC Engineering Orbital Dynamics
NAV Navigation
RK4 Runge-Kutta method

SGB Splice Guidance Body
SOCP Second Order Cone Program
SPLICE Safe & Precise Landing Integration Capabilities Evolution

II. Lunar Hazard Detection & Avoidance Concept of Operations

Performance testing of the GNC algorithms has been prepared for supporting a lunar powered descent trajectory similarly modeled to a Commercial Lunar Payload Services (CLPS) mission application. SPLICE designed a generic lunar lander for the purposes of simulated testing. The vehicle mass properties, size, and available effectors were modeled after the average CLPS lunar lander. The propulsion system used consists of a single fixed-axis main engine (no thrust vector control system) along with a set of Reaction Control System (RCS) jets for the attitude control system.

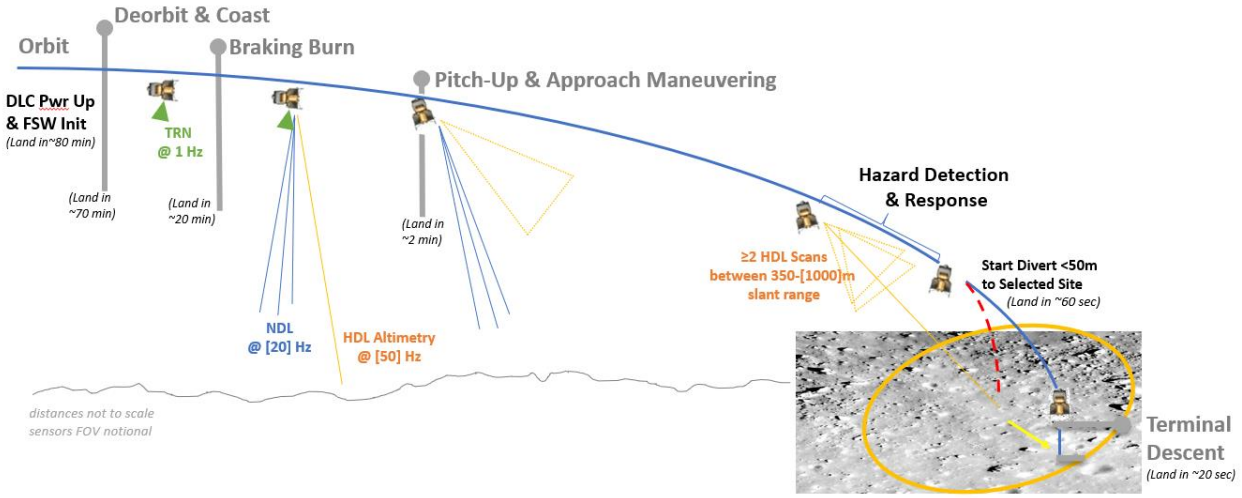


Fig. 1 SPLICE Concept of Operations

Fig. 1 shows the SPLICE lunar descent concept of operations. The performance analyzed in this manuscript begins shortly after the pitch-up and approach maneuver and the start of the approach phase. Prior to this point, the host vehicle guidance and controller system are used for the braking burn phase. During the approach phase, the DQG algorithm is first executed and provides a full 6-DOF trajectory along with main engine thrust and attitude control system torque commands with the final target being the initial terminal descent initiation point. The SPLICE controller then provides attitude control along the desired DQG generated reference trajectory.

As the vehicle approaches the initial terminal descent point, it will eventually be close enough to do a hazard detection scan of the intended landing site. The vehicle is modeled with a fixed hazard detection lidar boresight vector, that must be pointed as close as possible to the initial landing site. A set of state constraints are enforced by DQG to ensure a direct and visible scan of the landing site during the Hazard Detection and Response phase. The SPLICE HD algorithm will then analyze and identify a set of safe sites within the scan region, and the SPLICE Mode Commander application will have the ability to issue a divert command to a new site. If this occurs, the SPLICE navigation algorithms will provide state estimates relative to the new site and a divert command is issued to guidance [12]. DQG will then execute a second solve and provide a new trajectory, main engine thrust commands, and attitude control system torque commands to achieve the new terminal descent initiation point. In the current concept of operations, we are only accounting for a single divert command being issued if one is issued at all. The SPLICE attitude controller will then begin to track the new attitude profile until the vehicle reaches the terminal descent initiation altitude.

B. SPLICE GNC Architecture

DQG receives current navigated state estimates from the SPLICE navigation algorithms, which it then uses to produce a full 6-DOF state trajectory along with a nominal set of body thrust vector commands for the main engine, and body torque commands for the attitude control system. Fig. 2 is an architectural flow chart of the SPLICE GNC system. Ultimately, the DQG generated solution is to be provided to the SPLICE controller which operates at 50 Hz. However, DQG only executes when a new trajectory is required, and the provided trajectory is not a 50 Hz trajectory that can be used directly with the SPLICE controller. Therefore, the Guidance Target State Generator (GTSG) was designed to bridge the gap between DQG and the controller. GTSG can provide a target state based on the DQG trajectory and supply it to the controller at a 50 Hz rate.

The SPLICE controller used in this analysis as a stand-in that represents the host vehicle controller. It receives the desired attitude, angular rate, and torque commands from GTSG along with the current navigated state information from the navigation application to compute state errors. The GTSG-provided torque command then becomes modulated by a proportional derivative control law which is passed to the actuator allocator and the host vehicle. The controller does not currently track position and velocity errors, instead the nominal DQG main engine thrust commands are commanded directly to the host vehicle. In future applications, we aim to remedy this and have a fully closed loop controller modulating both the DQG provided thrust and torque commands to minimize position, velocity, attitude, and angular rate errors.

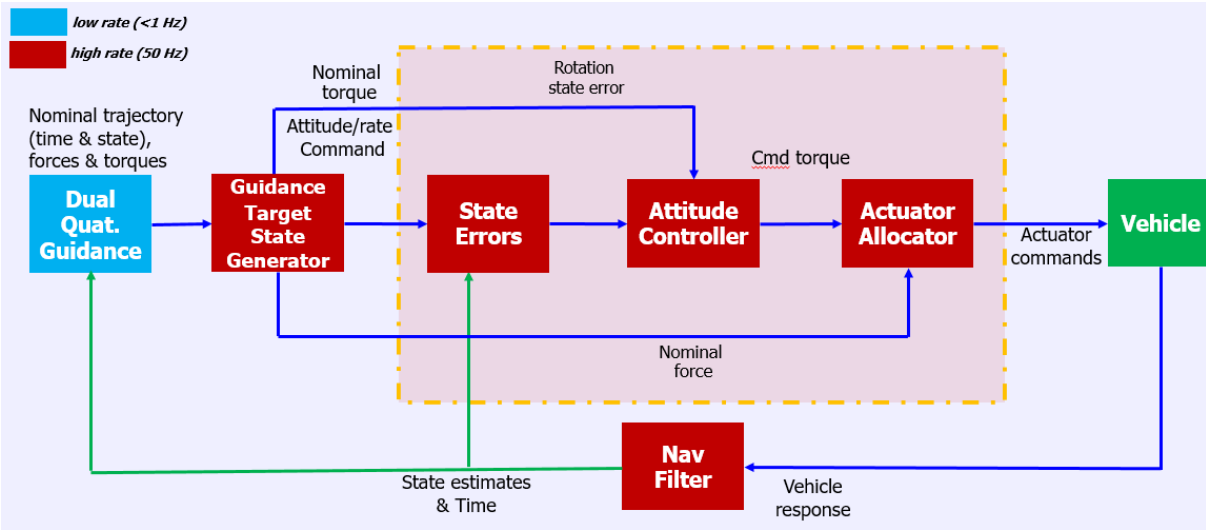


Fig. 2 SPLICE GNC Software Architecture

III. Dual Quaternion Guidance Algorithm

DQG is an algorithm that provides a fuel optimal trajectory solution that meets several vehicle constraints by formulating the dynamics and constraints as a Second Order Cone Convex Optimization Problem (SOCP). Once formulated into a SOCP structure, a solver can then optimize and determine a solution.

The 6-DOF rocket guidance problem is nonlinear, time-varying, and some constraints can be non-convex, all of which would prevent a SOCP from determining a reliable solution. DQG exercises linearization and discretization techniques to massage the dynamics into a set of discrete linear time invariant models. The vehicle state information is represented using dual quaternions to combine the attitude and position state information into a single term. The representation using dual quaternions allows certain constraints to become convex by combining coupled attitude and translation information, and in other cases it simplifies the convexification approximation. In most cases a first order Taylor series expansion of a non-convex constraint results in a convex constraint, and this technique is used to handle any remaining non-convex constraints.

To clean up all the approximations that take place during the linearization and convexification process, DQG uses a successive convexification strategy to iteratively apply these approximations and drive the solution closer and closer to the reference solution. The successive convexification loop, and an overview of the DQG algorithm phases are shown in Fig. 3.

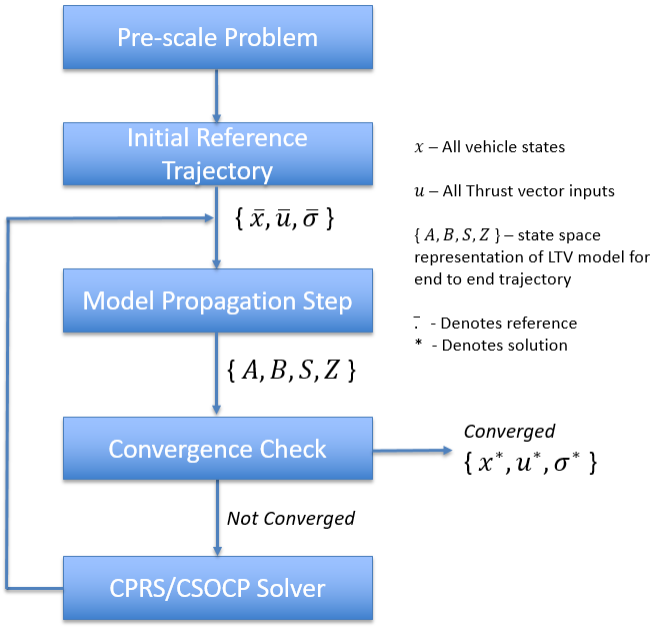


Fig. 3 DQG Algorithm Architecture

A. Pre-scaling

The entirety of computations within SPLICE's DQG algorithm are completed with a scaled state, vehicle, environment, and control parameters. Pre-scaling first occurs at application initialization, when all the internal parameters are scaled by predefined scale factors. Additionally, all dynamic inputs to the DQG application such as the current estimated state are also scaled up front prior to being utilized by the guidance algorithm. We utilized pre-scaling as a tuning parameter that has been shown to directly affect the optimization problem's conditioning. An ill-conditioned problem tends to require more solver iterations to converge than a well-conditioned problem, and thus proper selection of scale factors can help ensure that we keep the problem well-conditioned.

First, scale factors are chosen for the unit of length relative to the ENU landing site, unit of length relative to the vehicle guidance body frame origin, unit of time, and unit of mass. Utilizing these four building blocks, scale factors can then be computed for other kinematic and dynamic descriptions such as velocity, angular velocity, acceleration, force, torque, and inertia. We have found that it is important to utilize two scale factors for units of length relative to the ENU landing site and relative to the vehicle's guidance body frame due to the significant difference in scale size. If we were to just use a length scale factor relative to the ENU landing site, the resulting torque and inertia scale factors that are computed would be incredibly small and lead to truncation error in implementation resulting in artificial constraint violations.

B. Initial Trajectory Generation

The DQG algorithm requires an initial reference trajectory to linearize about and provide an initial guess to the solver. The quality of the initial reference trajectory can have a significant impact to the rate of convergence, since providing the algorithm with a dynamically feasible trajectory that already meets constraints should result in a reduction of successive convexification iterations. However, this implementation of the algorithm uses the simplest method of

initial reference trajectory generation which is the straight-line approach. Prior to starting the successive convexification loop, DQG generates an initial trajectory that interpolates uniformly distributed states starting from the initial state conditions and ending at the desired target conditions. All non-attitude state conditions are interpolated utilizing a simple linear interpolation method, and the quaternion attitude trajectory is interpolated using spherical linear interpolation. Control commands are also part of the initial reference trajectory, and so initial thrust and torque commands are also generated. Thrust commands are selected based on offsetting the force of gravity on the vehicle. The initial torque commands are produced by generating torque values that would offset any lever arm effects that the main engine initial thrust commands cause during reference trajectory generation.

C. Kinematics & Dynamics

The vehicle state information, kinematics, and dynamics are all represented using unit dual quaternion state variables. This state representation allows the combination of both vehicle position and attitude to be represented as a single unit dual quaternion variable $\tilde{\boldsymbol{q}}$. Eq. (1) shows how the dual quaternion position/attitude variable is constructed where $\mathbf{q}_{\mathrm{ENU}\to SGB}$ is the attitude quaternion that describes the transformation from the ENU frame to the SPLICE Guidance Body (SGB) frame, ε is the dual unit, and $\boldsymbol{r}_{\mathrm{ENU}}$ is a pure quaternion representing the vehicle's current ENU position. We can also define the angular velocity and velocity components as a dual quaternion. Eq. (2) shows this representation, where $\boldsymbol{\omega}_{\mathrm{SG}B}$ is a pure quaternion representing the angular velocity in the SGB frame, and $\boldsymbol{v}_{\mathrm{SG}B}$ is a pure quaternion representing the vehicle's velocity in the SGB frame.

$$\tilde{\boldsymbol{q}} = \boldsymbol{q}_{ENU\to SGB} + \frac{\varepsilon}{2}\boldsymbol{r}_{\mathrm{ENU}} \otimes \boldsymbol{q}_{ENU\to SGB} \tag{1}$$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_{\mathrm{SG}B} + \varepsilon \boldsymbol{v}_{\mathrm{SG}B} \tag{2}$$

Utilizing this state representation, the kinematic equation can be derived and is shown in Eq. (3). The mass kinematic equation is also shown in Eq. (4), where the scalar α is the inverse of the nozzle exit velocity, and $\boldsymbol{T_{SGB}}$ is the thrust command vector in the SGB frame. It is important to note here that mass depletion is only computed as a function of main engine thrust commands. Thus, RCS torque commands that are applied to vehicle do not deplete the available fuel and are therefore free in this implementation.

$$\dot{\tilde{\boldsymbol{q}}} = \frac{1}{2}\tilde{\boldsymbol{q}} \otimes \tilde{\boldsymbol{\omega}} \tag{3}$$

$$\dot{m} = -\alpha \|\boldsymbol{T_{SGB}}\|_2 \tag{4}$$

Eq. (5) shows how the vehicle dynamics are represented using dual quaternions, where $\tilde{\boldsymbol{J}}$ is an eight-by-eight inertia matrix shown in Eq. (6), $\tilde{\boldsymbol{F}}$ represents the total external forces and torques applied to the vehicle through the vehicle's control system and is shown in Eq. (7), and $\tilde{\boldsymbol{g}}_{SGB}$ represents the effects of gravity acceleration represented as a dual quaternion in Eq. (8) in the SGB frame. The derivations for the dual quaternion kinematics and dynamics are in [9].

$$\tilde{\boldsymbol{J}}\dot{\tilde{\boldsymbol{\omega}}} + \tilde{\boldsymbol{\omega}} \oslash \tilde{\boldsymbol{J}}\tilde{\boldsymbol{\omega}} = \tilde{\boldsymbol{F}} + \tilde{\boldsymbol{g}}_{SGB} \tag{5}$$

$$\tilde{\boldsymbol{J}} = \begin{bmatrix} \mathbf{0}_{4\times4} & m\boldsymbol{I}_3 & \mathbf{0}_{3\times1} \\ & \mathbf{0}_{3\times1} & 1 \\ \boldsymbol{J} & \mathbf{0}_{3\times1} & \\ \mathbf{0}_{1\times3} & 1 & \mathbf{0}_{4\times4} \end{bmatrix}_{8\times8} \tag{6}$$

$$\tilde{\boldsymbol{F}} = \begin{bmatrix} \boldsymbol{T}_{SGB} + m\boldsymbol{g}_{SGB} \\ \mathbf{0} \\ \boldsymbol{r}_u{}^{\times}\boldsymbol{T}_{SGB} + \boldsymbol{\tau}_{SGB} \\ \mathbf{0} \end{bmatrix}_{8\times1} \tag{7}$$

$$\widetilde{\boldsymbol{g}}_{SGB} = \begin{bmatrix} m\boldsymbol{g}_{SGB} \\ \boldsymbol{0}_{4\times 1} \end{bmatrix}_{8\times 1} \tag{8}$$

D. Linearization & Discretization

The vehicle dynamics are nonlinear and time varying, but to express the dynamic constraint for a convex problem we need to complete a linearization and discretization step. Eq. (9) describes the vehicle dynamics as a function of our state $\boldsymbol{x}(t)$, control $\boldsymbol{u}(t)$, and time t. The formulation of the state and control variables are also shown in Eq. (10) and Eq. (11), where $\boldsymbol{T}$ represents the main engine thrust vector, and $\boldsymbol{\tau}$ represents the additional torque vector supplied by an attitude control system.

$$\dot{\boldsymbol{x}}(t) = f(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\,), \qquad t \in [0, t_f) \tag{9}$$

$$\boldsymbol{x}(t) := \begin{bmatrix} m(t) \\ \widetilde{\boldsymbol{q}}(t) \\ \widetilde{\boldsymbol{\omega}}(t) \end{bmatrix} \tag{10}$$

$$u(t) := \begin{bmatrix} \boldsymbol{T}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} \tag{11}$$

Prior to linearization, we apply a time normalization to the dynamic equation which has become a standard practice in free final time successive convexification problems [7]. This normalization is done by using a time dilation factor σ, which becomes a new solution variable and equates to the trajectory time of flight. This transforms the dynamic equation to Eq. (12).

$$\dot{\boldsymbol{x}}(\tau) = F(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \sigma\,) \tag{12}$$

We can now complete the linearization step which is accomplished by taking the first order Taylor series expansion of the dynamic equation about a reference trajectory $\bar{\boldsymbol{x}}(\tau)$, reference control input $\bar{\boldsymbol{u}}(\tau)$, and reference trajectory time of flight $\bar{\sigma}$. The resulting continuous time equation is shown in Eq. (13).

$$\dot{\boldsymbol{x}}(\tau) = A(\tau)[\boldsymbol{x}(\tau) - \bar{\boldsymbol{x}}(\tau)] + B(\tau)[\boldsymbol{u}(\tau) - \bar{\boldsymbol{u}}(\tau)] + S(\tau)[\sigma - \bar{\sigma}] \tag{13}$$

Once the linear continuous time dynamics have been defined, we will need to perform a discretization step to define a set of linear time invariant models. These discrete points along the trajectory are uniformly distributed in time between [0,1] (due to the time normalization) and are typically called trajectory nodes. Thus, the entire trajectory is represented in the convex problem as a discrete set of nodes for which the dynamics and all state constraints will be enforced. To complete the discretization, the control inputs are modeled with a first order hold and a uniform fixed step RK4 integration method is utilized to propagate the nonlinear dynamics, compute a state transition matrix, and the input transition matrix simultaneously using a multiple shooting method. A complete derivation of this process is covered in [13]. A visual example of the shooting method is shown in Fig. 4. Each reference state node $\bar{\boldsymbol{x}}_k$, where k represents the kth node, is used as an initial condition for each RK4 propagated segment which propagates up to the next node. The state defect between the nonlinear dynamics and the reference trajectory state are recorded and referenced in Fig. 4 as Δ.

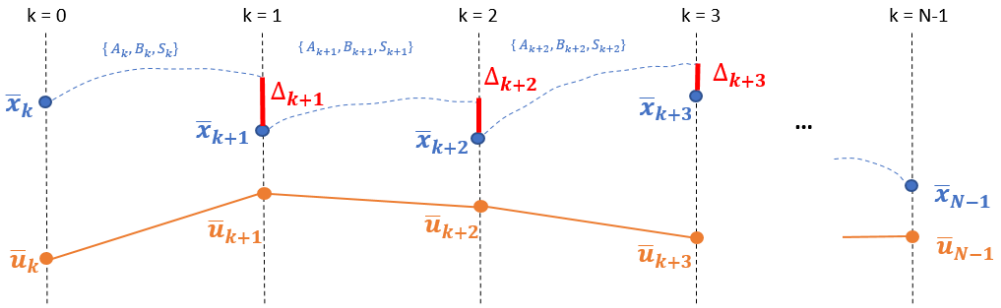


Fig. 4 RK4 Multiple Shooting Method

The SPLICE DQG application also uses an implementation of the stitching condition in the dynamic equation, which allows for the dynamic equation to factor in the state defects caused by the nonlinear dynamics [14]. The final resulting discrete dynamic equation is shown in Eq. (14).

$$\boldsymbol{x}_{k+1} - \bar{\boldsymbol{x}}_{k+1} = \boldsymbol{A}_k(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k) + \boldsymbol{B}_k(\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k) + \boldsymbol{B}_{k+1}(\boldsymbol{u}_{k+1} - \bar{\boldsymbol{u}}_{k+1}) + \boldsymbol{S}_k(\sigma - \bar{\sigma}) + \Delta_{k+1} \tag{14}$$

E. Convergence Criteria

Upon completion of the model propagation step, we now have everything we need to setup the convex optimization problem and execute the solve step. However, prior to executing the solver we can first verify whether the reference trajectory, control input, and time of flight meet the convergence criteria. The convergence criteria checks are skipped on the first DQG successive convexification iteration because it is obvious that the initial line guess that is currently being used would result in a dynamically infeasible solution. Convergence is thus assessed at all subsequent successive convexification iterations and are evaluated based on three criteria: dynamic feasibility, solution optimality, and remaining propellant optimality.

1. Dynamic Feasibility

Section III.D described the state defect term which is used when implementing the stitching condition in the dynamic constraint and shown in Eq. (14). The state defect term is also used as an indication of dynamic feasibility. When the state defects between the propagated nonlinear dynamics and the reference state node are small enough, then this indicates that our reference trajectory and control solution is dynamically feasible. Thus, tolerances for each state variable are defined as parameters, and for each successive convexification iteration the state defect terms are evaluated against the desired tolerances. When the state defects meet the desired tolerances, then the solution passes the first criteria for convergence. The dynamic feasibility criteria must always be met to achieve a converged solution.

2. Solution Optimality

If dynamic feasibility has been met, the next criterion evaluates if an optimal solution has been found. This is accomplished by quantifying the difference between the current solution and the reference solution, which is shown in Eq. (15). A small difference between the latest solution and the reference solution indicates that the successive convexification process is no longer producing significantly unique solutions and has converged.

$$\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_\infty \leq \varepsilon_{tol} \tag{15}$$

3. Propellant Optimality

The final criterion is checked as an additional protection to prevent DQG from over executing iterations for marginal gain. In certain cases, it was observed that although dynamic feasibility was accomplished the algorithm could

potentially fail to meet the solution optimality tolerance criteria. This resulted in continuous successive convexification iterations that provided little to no benefit to the final solution metrics. To prevent this, an additional check is accomplished by evaluating the difference in the vehicle's remaining mass between consecutive dynamically feasible trajectories. This criterion is shown in Eq. (16), which offered additional protection in cases were there could be large enough changes in the trajectory solution to fail to meet the solution optimality tolerance but result in marginal changes to the vehicle's final mass.

$$|m_F - \bar{m}_F| \leq m_{tol} \tag{16}$$

F. Constraint Enforcement

1. Initial Condition Constraints

This classification of constraints defines the vehicle's initial state conditions and are only applied at the first node of the trajectory. Eq. (17) enforces the initial total vehicle mass which includes propellant mass. Mass properties information is provided directly from the host vehicle. Eq. (18)-(20) all define the initial vehicle attitude, velocity, and angular velocity states provided by SPLICE's navigation algorithm. All these constraints are equality constraints.

$$m_0 = m_{IC} \tag{17}$$

$$\boldsymbol{q}_0 = \boldsymbol{q}_{IC} \tag{18}$$

$$\boldsymbol{v}_{B_0} = \boldsymbol{v}_{IC} \tag{19}$$

$$\boldsymbol{\omega}_{B_0} = \boldsymbol{\omega}_{IC} \tag{20}$$

The initial vehicle position quaternion is also enforced as an equality constraint, however an additional solution variable $\boldsymbol{\delta}_r$ is utilized to provide some slack in the initial state position. This position slack term is set as a pure quaternion that defines a position bias in the East and North direction. A zero value is enforced in the Up direction of the $\boldsymbol{\delta}_r$ pure quaternion and is shown in Eq. (22). Although this positional bias is applied to the first node, the true purpose of this bias is to allow some horizontal position miss at terminal descent initiation. During post-processing of the trajectory solution, the entire position trajectory is then offset by the bias value. This bias is applied as an initial condition rather than a final condition because the initial attitude is defined directly as the vehicle's current attitude, while the terminal attitude is a solution variable and is not fixed. Since the position quaternion is a function of the attitude quaternion, applying this bias at the terminal condition would result in a non-convex constraint. Eq. (21) outlines this initial position constraint as a function of the initial vehicle attitude $\boldsymbol{q}_{IC}$, initial position quaternion $\boldsymbol{q}_{\boldsymbol{r}_0}$ and the terminal position bias factor $\boldsymbol{\delta}_r$. Naturally, the purpose of any targeting or guidance algorithm is to minimize any terminal position miss distance, thus $\boldsymbol{\delta}_r$ is added as an additional term in the minimization cost function.

$$\boldsymbol{q}_{\boldsymbol{r}_0} + \frac{1}{2}[\boldsymbol{q}_{IC}]^*_{\otimes}\boldsymbol{\delta}_r = \frac{1}{2}\boldsymbol{r}_{ENU_{IC}} \otimes \boldsymbol{q}_{IC} \tag{21}$$

$$\boldsymbol{\delta}_U = 0 \tag{22}$$

2. Final Condition Constraints

This classification of constraints defines the desired state conditions of the vehicle at the at the start of the vertical descent phase. Since the total mass of the vehicle is treated as a state solution variable, Eq. (23) enforces a limitation on how much propellant can be depleted from the approach phase to the vertical descent phase, where m_N is the mass of the vehicle at the final node and m_{TDI} is the minimum total vehicle mass required to complete the vertical descent phase.

$$m_N \geq m_{TDI} \tag{23}$$

Since there are no restrictions on the roll axis attitude orientation at the start of vertical descent, a maximum final tilt angle constraint is utilized at the final node of the trajectory. Eq. (24) describes this constraint, where $\widetilde{\boldsymbol{q}}_N$ is the dual quaternion defining position and attitude at the final node, $\boldsymbol{M}_T$ is the tilt angle coefficient matrix defined in Eq. (25), and $\theta_F$is the maximum terminal tilt angle.

$$\widetilde{\boldsymbol{q}}^{\boldsymbol{T}}{}_N \boldsymbol{M}_T \widetilde{\boldsymbol{q}}_N + \cos(\theta_F) - 1 \leq 0 \tag{24}$$

$$\boldsymbol{M}_T = \begin{bmatrix} [\boldsymbol{z_I}]_{\otimes}[\boldsymbol{z_B}]^*_{\otimes} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} \end{bmatrix} \tag{25}$$

The maximum final tilt angle constraint allows the final attitude orientation to remain a solution variable with some limitation on tilt. Due to the nature of dual quaternions, the final position quaternion becomes a function of the final attitude quaternion $\boldsymbol{q}_N$ and the parameter $\boldsymbol{r}_{I_F}$, which defines the targeted final position represented as a pure quaternion. Eq. (26) defines this equality constraint.

$$\boldsymbol{q}_{\boldsymbol{r}_N} = \frac{1}{2}\boldsymbol{r}_{ENU_F} \otimes \boldsymbol{q}_N \tag{26}$$

The terminal velocity constraints of the approach phase are separated into a vertical and horizontal velocity constraint. The horizontal velocity constraint shown in Eq. (27), limits the magnitude of the vehicle's velocity in the East and North directions at the final node, $\boldsymbol{v}_{EN_N}$. While the vertical velocity at the final node, v_{U_N}, is constrained by two linear constraints shown in Eq. (28).

$$\left\|\boldsymbol{v}_{EN_N}\right\|_2 \leq v_{EN\,max} \tag{27}$$

$$-v_{U\,max} \leq v_{U_N} \leq 0 \tag{28}$$

Similarly, to the final velocity constraints, the final angular velocity constraints are also separated into a final tilt rate constraint and a final roll rate constraint. Eq. (29) defines the tilt rate constraint which limits the magnitude of the vehicle pitch and yaw rate state variables of the final node, $\boldsymbol{\omega}_{XY_N}$. The final roll rate ω_{Z_N} is then sandwiched between two linear inequalities shown in Eq. (30).

$$\left\|\boldsymbol{\omega}_{XY_N}\right\|_2 \leq \omega_{XY_F} \tag{29}$$

$$-\omega_{Z_F} \leq \omega_{Z_N} \leq \omega_{Z_F} \tag{30}$$

3. Control Constraints

The vehicle control constraints ensure that commanded main engine thrust, and torque commands remain within the hardware limitations, and thus allows for the solver to generate a trajectory that is feasible with the available hardware. The vehicle used in this analysis has a fixed main engine, and thus Eq. (31) enforces a minimum and maximum thrust command along the body Z axis. Eq. (32) constrains the other body axis directions to zero since the engine is fixed.

$$T_{min} \leq T_{Z_k} \leq T_{max} \tag{31}$$

$$T_{X_k} = 0, \qquad T_{Y_k} = 0 \tag{32}$$

There is also a limitation on the engine's thrust rate, which is shown in Eq. (33) where $\dot{T}_{max}$ is the maximum thrust rate capability, N is the total number of nodes that the trajectory is discretized by, and σ is the total trajectory time solution.

$$-\dot{T}_{max}\sigma \leq \left(T_{Z_k} - T_{Z_{k-1}}\right)(N-1) \leq \dot{T}_{max}\sigma \tag{33}$$

Since we have a fixed main engine, DQG does solve for and command a torque and can do so in all three body axis directions. The maximum amount of torque applied is constrained in Eq. (34)-(36). Since each of these constraints are independent of each other, individual torque limitations can be enforced at each axis.

$$-\tau_{X_{max}} \leq \tau_{X_k} \leq \tau_{X_{max}} \tag{34}$$

$$-\tau_{Y_{max}} \leq \tau_{Y_k} \leq \tau_{Y_{max}} \tag{35}$$

$$-\tau_{Z_{max}} \leq \tau_{Z_k} \leq \tau_{Z_{max}} \tag{36}$$

4. Global State Constraints

The constraint classifications prior to this all described the typical set of guidance constraints such as target conditions and vehicle hardware limitations. The remaining constraints are a set of globally applied state constraints enforced at all nodes of the trajectory. Eq. (37) defines a minimum altitude constraint, which prevents DQG from converging on a solution that may fly hazardously close to the lunar surface. This constraint was derived by starting with the glide slope constraint shown in Eq. (53), represented as a second order cone constraint described with dual quaternions. The cone vertex was then offset by a desired minimum altitude value r_{alt} and the maximum glide slope angle was set to 90 degrees. Setting the maximum angle to 90 degrees eliminates a term in the constraint and defines a plane above the surface for which the vehicle must never drop below. This constraint was then linearized about the reference trajectory position and attitude quaternion defined as $\bar{\bar{\boldsymbol{q}}}_k$, which transforms it into a linear half-space constraint.

$$-\bar{\bar{\boldsymbol{q}}}_k{}^T \boldsymbol{M}_G \bar{\bar{\boldsymbol{q}}}_k + r_{alt} + \left[-2\bar{\bar{\boldsymbol{q}}}_k{}^T \boldsymbol{M}_G\right]\left(\tilde{\boldsymbol{q}}_k - \bar{\bar{\boldsymbol{q}}}_k\right) \leq 0 \tag{37}$$

$$\boldsymbol{M}_G = \begin{bmatrix} 0_{4x4} & [\boldsymbol{z}_I]_{\otimes}^T \\ [\boldsymbol{z}_I]_{\otimes} & 0_{4x4} \end{bmatrix} \tag{38}$$

In addition to having a terminal tilt angle constraint, there is also a global version of the constraint shown in Eq. (39). The only difference here is that the global maximum tilt angle parameter θ_{max} is used instead of the terminal maximum tilt angle.

$$\tilde{\boldsymbol{q}}^{\boldsymbol{T}}{}_k \boldsymbol{M}_T \tilde{\boldsymbol{q}}_k + \cos(\theta_{max}) - 1 \leq 0 \tag{39}$$

The final set of global state constraints defines a box constraint sandwiching the vehicle's angular velocity in each individual body axis. Eq. (40)-(42) define how the angular velocity in each body axis is constrained using a set of scalar infinity norm constraints.

$$-\omega_{X_{max}} \leq \omega_{X_k} \leq \omega_{X_{max}} \tag{40}$$

$$-\omega_{Y_{max}} \leq \omega_{Y_k} \leq \omega_{Y_{max}} \tag{41}$$

$$-\omega_{Z_{max}} \leq \omega_{Z_k} \leq \omega_{Z_{max}} \tag{42}$$

5. State-Triggered Hazard Detection Constraints

This purpose of State-Triggered HD constraints is to ensure that the Hazard Detection system can adequately perform its function in scanning the targeted landing site and identifying a safe landing location. These constraints are relatively restrictive and would be difficult to enforce during the entire trajectory. Additionally, these constraints are only required during a specific flight envelope of the trajectory for which the surface scans are expected to occur. Thus, defining the hazard detection constraints as a set of state triggered constraints makes the most sense. State triggered constraints allow for specific constraints to be activated or deactivated based on specific state conditions along the trajectory. This is done by defining trigger functions for the activation or deactivation of these constraints and then applying them multiplicatively to the original state constraint. Eq. (43) shows an example of a generic state-triggered constraint, where a constraint f_c is being multiplied by multiple trigger functions s_k. The actual values of these trigger functions only influence the constraint enforcement when they are set to zero, which effectively deactivates the constraint. A more complete derivation of the use of state triggered constraints can be found in [8] and [9]. For this application of Hazard Detection, two slant ranged triggers are used to activate and then deactivate the enforcement of these constraints during the expected scan region. Eq. (44) defines the slant range trigger functions, where $\boldsymbol{q}_{r_k}$ is the position quaternion at the kth node, d_{min} is the minimum slant range parameter for completion of the scan, and d_{max} is the maximum slant range parameter that triggers the activation of the constraints. Due to the multiplicative trigger functions that are each a function of a solution variable, the resulting constraint inequalities are typically non-convex.

$$s_1(\boldsymbol{x})s_2(\boldsymbol{x}) \dots s_k(\boldsymbol{x})f_c(\boldsymbol{x},\boldsymbol{u},\sigma) \leq 0 \tag{43}$$

$$s_1(\boldsymbol{q}_r) = max\left(2\left\|\boldsymbol{q}_{r_k}\right\|_2 - d_{min}, 0\right), s_2(\boldsymbol{q}_r) = max\left(d_{max} - 2\left\|\boldsymbol{q}_{r_k}\right\|_2, 0\right) \tag{44}$$

Eq. (45) shows one of the hazard detection constraints which requires that the vehicle's velocity must remain below a specific magnitude during the scan, where $s_2(\boldsymbol{q}_r)$ is the activation trigger function, $\boldsymbol{v}_k$ is the vehicle velocity vector at the kth node, and $v_{HD\ max}$ is the maximum speed parameter during the hazard detection scan. The deactivation trigger function $s_1(\boldsymbol{q}_r)$ is not applied to this constraint, because the focus of the lunar powered descent problem is to slow down the vehicle enough to reach the desired target constraints. Allowing the vehicle to slow down for the scan and then speed up afterwards is not desired when the goal is to also conserve propellant. Thus, this constraint is always enforced once the vehicle's position is within the activation slant range.

$$s_2(\boldsymbol{q}_r)\|\boldsymbol{v}_k\|_2 \leq v_{HD\ max} \tag{45}$$

To deal with the non-convex constraint, Eq. (45) is linearized about the reference trajectory and is shown in Eq. (46), where $\bar{\boldsymbol{q}}_{\boldsymbol{r_k}}$ is the reference position quaternion of the kth node, $\bar{\boldsymbol{v}}_k$ is the reference velocity vector of the kth node, $\tilde{\boldsymbol{q}}_k$ is the dual quaternion solution variable that defines both position and attitude at the kth node, and $\tilde{\boldsymbol{\omega}}_k$ is the dual quaternion solution variable that defines both velocity and angular velocity at the kth node. The resulting inequality constraint is a linear half-space constraint.

$$s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})(\|\bar{\boldsymbol{v}}_k\|_2 - v_{HD\ max}) + s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})(\|\bar{\boldsymbol{v}}_k\|_2 - v_{HD\ max})(\tilde{\boldsymbol{q}}_k - \bar{\tilde{\boldsymbol{q}}}_k) + \dot{s}_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\left(\frac{\bar{\boldsymbol{v}}_k}{\|\bar{\boldsymbol{v}}_k\|_2}\right)(\tilde{\boldsymbol{\omega}}_k - \bar{\tilde{\boldsymbol{\omega}}}_k) \leq 0 \tag{46}$$

The vehicle's angular velocity is also constrained more strictly during the scan to ensure that the HDL is steadily pointing at the targeted landing site. Since this constraint enforces hard restrictions on the vehicle, it is only applied during the scan and is relaxed afterward unlike the velocity constraint. Following the same linearization procedure as the state-triggered velocity constraint, results in the constraint described in Eq. (47). Where $\bar{\boldsymbol{\omega}}_k$ is the reference angular velocity vector of the kth node, $\omega_{HD\ max}$ is the maximum angular velocity allowed during the scan, $\tilde{\boldsymbol{\omega}}_k$ is the dual quaternion solution variable that defines both angular and translational velocity, and $\bar{\tilde{\boldsymbol{\omega}}}_k$ is the reference dual quaternion defining both angular and translational velocity.

$$\begin{aligned} s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})(\|\bar{\boldsymbol{\omega}}_k\|_2 - \omega_{HD\ max}) + \left[\left(\dot{s}_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}}) + s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\dot{s}_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\right)(\|\bar{\boldsymbol{\omega}}_k\|_2 - \omega_{HD\ max})\right](\tilde{\boldsymbol{q}}_k - \bar{\tilde{\boldsymbol{q}}}_k) \\ + \left[s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\frac{\bar{\boldsymbol{\omega}}_k}{\|\bar{\boldsymbol{\omega}}_k\|_2}\right](\tilde{\boldsymbol{\omega}}_k - \bar{\tilde{\boldsymbol{\omega}}}_k) \leq 0 \end{aligned} \tag{47}$$

The HDL sensor line of sight to the desired landing site is a critical constraint to ensure that the HDL system can get an accurate scan of the landing site and the surrounding area. It is a difficult constraint to meet and is a function of both vehicle attitude and position. Prior to applying the activation and deactivation trigger functions, the line-of-sight constraint is already non-convex. The same linearization procedure is applied to this constraint which is shown in Eq. (48), where $f_{\varphi}(\bar{\bar{\boldsymbol{q}}}_k)$ is the left-hand side of the line-of-sight constraint as a function of the reference trajectory and is defined in Eq. (49). Additionally, Eq. (51) defines the coefficient matrix $\boldsymbol{M}_L$, where $\boldsymbol{p}_B$ is a pure quaternion that defines the HDL boresight vector represented in the guidance body frame. The derivation for the state-triggered line of sight constraint can be found in [15].

$$s_1(\bar{\boldsymbol{q}}_{r_k})s_2(\bar{\boldsymbol{q}}_{r_k})f_{\varphi}(\bar{\bar{\boldsymbol{q}}}_k) + \left[s_2(\bar{\boldsymbol{q}}_{r_k})\dot{s}_1(\bar{\boldsymbol{q}}_{r_k})f_{\varphi}(\bar{\bar{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{r_k})\dot{s}_2(\bar{\boldsymbol{q}}_{r_k})\, f_{\varphi}(\bar{\bar{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{r_k})s_2(\bar{\boldsymbol{q}}_{r_k})\frac{\partial f_{\varphi}(\bar{\bar{\boldsymbol{q}}}_k)}{\partial \bar{\bar{\boldsymbol{q}}}_k}\right](\tilde{\boldsymbol{q}}_k - \bar{\bar{\boldsymbol{q}}}_k) \leq 0 \tag{48}$$

$$f_{\varphi}(\bar{\bar{\boldsymbol{q}}}_k) = \bar{\bar{\boldsymbol{q}}}_k{}^T \boldsymbol{M}_L \bar{\bar{\boldsymbol{q}}}_k + \left\|2\bar{\boldsymbol{q}}_{r_k}\right\|_2 \cos(\varphi_{HD\,max}) \tag{49}$$

$$\frac{\partial f_{\varphi}(\bar{\bar{\boldsymbol{q}}}_k)}{\partial \bar{\bar{\boldsymbol{q}}}_k} = 2\bar{\bar{\boldsymbol{q}}}_k{}^T \boldsymbol{M}_L + \frac{2\bar{\boldsymbol{q}}_{r_k}}{\left\|\bar{\boldsymbol{q}}_{r_k}\right\|_2}\cos(\varphi_{HD\,max}) \tag{50}$$

$$\boldsymbol{M}_L = \begin{bmatrix} 0_{4x4} & [\boldsymbol{p}_B]^*_{\otimes}{}^T \\ [\boldsymbol{p}_B]^*_{\otimes} & 0_{4x4} \end{bmatrix} \tag{51}$$

The final state triggered constraint being enforced is a glide slope constraint active during the scan which is being utilized to prevent a scan of the landing site that is too oblique. If scans are completed with an oblique view, there is the possibility that some information about the landing site is lost due to obstacles, such as boulders, obscuring the surface behind them. The glide slope constraint ensures that the vehicle's position remains within a cone aligned with the Up direction and its vertex at the desired landing site. The half angle that defines the cone is $\gamma_{HD\,max}$. This constraint becomes non-convex after adding in the activation and deactivation trigger functions. Therefore, the linearization procedure is applied here and shown in Eq. (52)-(54), where $f_{\gamma}(\bar{\bar{\boldsymbol{q}}}_k)$ is the original left-hand side of the glide slope constraint as a function of the reference trajectory, and $\boldsymbol{M}_G$ is a coefficient matrix defined previously in Eq. (38).

$$s_1(\bar{\boldsymbol{q}}_{r_k})s_2(\bar{\boldsymbol{q}}_{r_k})f_{\gamma}(\bar{\bar{\boldsymbol{q}}}_k) + \left[s_2(\bar{\boldsymbol{q}}_{r_k})\dot{s}_1(\bar{\boldsymbol{q}}_{r_k})f_{\gamma}(\bar{\bar{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{r_k})\dot{s}_2(\bar{\boldsymbol{q}}_{r_k})\, f_{\gamma}(\bar{\bar{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{r_k})s_2(\bar{\boldsymbol{q}}_{r_k})\frac{\partial f_{\gamma}(\bar{\bar{\boldsymbol{q}}}_k)}{\partial \bar{\bar{\boldsymbol{q}}}_k}\right](\tilde{\boldsymbol{q}}_k - \bar{\bar{\boldsymbol{q}}}_k) \leq 0 \tag{52}$$

$$f_{\gamma}(\bar{\bar{\boldsymbol{q}}}_k) = -\bar{\bar{\boldsymbol{q}}}_k{}^T \boldsymbol{M}_G \bar{\bar{\boldsymbol{q}}}_k + \left\|2\bar{\boldsymbol{q}}_{r_k}\right\|_2 \cos(\gamma_{HD\,max}) \tag{53}$$

$$\frac{\partial f_{\gamma}(\bar{\bar{\boldsymbol{q}}}_k)}{\partial \bar{\bar{\boldsymbol{q}}}_k} = -2\boldsymbol{M}_G \bar{\bar{\boldsymbol{q}}}_k + \frac{2\bar{\boldsymbol{q}}_{r_k}}{\left\|\bar{\boldsymbol{q}}_{r_k}\right\|_2}\cos(\gamma_{HD\,max}) \tag{54}$$

G. Virtual State Variables

In previous applications of DQG there has always been a method to introduce some slack into the optimization problem that is penalized in the cost function. Slack variables aid convergence using sequential convexification methods and improve robustness of the guidance algorithm. In some cases, the enforced constraints may not be dynamically feasible given a set of initial and final conditions and a set of hardware limitations. In such cases, we would rather have the guidance algorithm converge on to a dynamically feasible trajectory solution that attempts to enforce all the desired constraints than to allow the algorithm to remain dynamically infeasible. In previous publications, DQG leaned heavily on the use of virtual control solution variables which were applied directly to the dynamic constraint equation shown in Eq. (14), as an additional term [9]. This allowed the solver to apply external dynamic slack, and the virtual control term was then heavily penalized in the cost function to keep it as small as possible. However whenever virtual control was applied, the slack of that virtual control trickled into the next sequential convexification iteration via the newly created reference trajectory. Thus, small adjustments made by the solver via virtual control application made dynamic changes to the reference trajectory which could influence evaluation of dynamic feasibility during the linearization and discretization process outlined in section III. D.

In recent publications, the use of a virtual state strategy was outlined which does not cause a negative effect to the linearization and discretization of subsequent reference trajectories [14]. The SPLICE DQG algorithm has adopted

this new strategy for implementing slack into the optimization problem. A virtual state solution variable $\boldsymbol{v}$ is created, that is of the same size and structure as the actual state variable $\boldsymbol{x}$. However, the virtual state variable is only applied at the vehicle state constraints. The only difference here is that the virtual state variables are not being utilized on the initial condition constraints and the final condition mass constraint shown in Eq. (23), but they are utilized on all other state constraints including the other final condition constraints. It is also important to note that there are no virtual slack variables are applied to control constraints, since we do not want to allow the solver to believe we have more control authority than physically possible. The remaining propellant constraint at the end of the trajectory is classified as a control constraint rather than a state constraint, and so it also refrains from using any virtual state slack variables.

H. DQG Cost Function

The SPLICE DQG algorithm utilizes a linear cost function built by several individual terms which are summed and minimized, shown in Eq. (55). Each term has an individual gain applied to them, and they each serve a unique purpose in the optimization problem. All the k coefficient gains applied are tunable parameters that can be adjusted to improve solution convergence robustness and speed. Each of the following sub-sections will review each term outlined in Eq. (55) in more detail.

$$J = -k_m m_N + k_\eta[(\boldsymbol{x}-\bar{\boldsymbol{x}})^2 + (\boldsymbol{u}-\bar{\boldsymbol{u}})^2 + (\boldsymbol{v}-\bar{\boldsymbol{x}})^2] + k_\sigma(\sigma-\bar{\sigma})^2 + k_s(\boldsymbol{v}_{1:N-1}-\boldsymbol{x}_{1:N-1})^2 + k_B(\boldsymbol{v}_N-\boldsymbol{x}_N)^2 + k_{miss}\boldsymbol{\delta}_r \tag{55}$$

1. Final Mass Term

DQG aims to solve for an ideal trajectory that meets all the desired state and hardware constraints and does so while using as little propellant as possible. The final mass term ensures that we minimize the latter, by minimizing the negative value of the final mass we are essentially stating that we would like to maximize the mass of the vehicle at the final node m_N.

2. State, Control, and Virtual State Trust Region Term

DQG handles all problem nonlinearities by performing a linearization step about a reference trajectory and utilizing this approximation to model the dynamic constraint. The approximation for this linearization technique breaks down when the vehicle state and control trajectory stray too far from the reference state and control trajectory. The trust region term in the cost function aims to penalize the solver from using solution variables that stray too far from the reference trajectory. Thus, the difference between the solution state $\boldsymbol{x}$ and the reference trajectory state $\bar{\boldsymbol{x}}$, the difference between the control solution $\boldsymbol{u}$ and the reference control command $\bar{\boldsymbol{u}}$, and the difference between the virtual state $\boldsymbol{v}$ and the reference trajectory state $\bar{\boldsymbol{x}}$ are all penalized and amplified by the gain k_η. Technically, the virtual state variable does not necessarily need to be close to the reference trajectory. However, later terms in the cost function focus on minimizing the difference between the state solution and the virtual state, and so limiting the virtual state to be near the reference trajectory state helps to prevent two terms from fighting each other during the optimization process.

3. Time Trust Region Term

Similarly, to the state, control, and virtual state trust region term, this term is a trust region term applied to the total trajectory time. The total trajectory time is a solution variable, and our linearization approximations are computed utilizing a reference burn time guess $\bar{\sigma}$. Therefore, the same logic applies here for the time trust region, we penalize the difference between the burn time solution variable σ and the reference burn time variable $\bar{\sigma}$, and amplify it by the gain k_σ.

4. Trajectory Virtual State Difference Term

The solution state variable $\boldsymbol{x}$ is only applied at the initial state conditions, the dynamic equality constraint, and the final mass condition constraint. The virtual state variable $\boldsymbol{v}$ is only applied at all the other trajectory constraints including the final condition constraints. This means that the resulting state variable $\boldsymbol{x}$ will result in a dynamically feasible trajectory defined by our dynamic model, however it may not necessarily meet all the other state constraints.

Meanwhile, the virtual state variable $\boldsymbol{v}$ will meet all the other trajectory state constraints but may not necessarily result in a dynamically feasible trajectory that obeys our dynamic model. The trajectory virtual state difference term aims to minimize the difference between the trajectory solution variable and the virtual state. When both the trajectory solution variable and virtual state variables are equivalent, then that means that the solution is both dynamically feasible with respect to our model and meets all the desired state constraints.

This specific trajectory virtual state difference term focuses on the minimization of the trajectory solution variable and the virtual state variable for all state constraints, except for the final condition constraints. This is denoted by the difference between $\boldsymbol{v}_{1:N-1}$ and $\boldsymbol{x}_{1:N-1}$. The coefficient gain k_s is designed to specifically penalize the virtual state difference terms for all state constraint except for the final boundary constraints. This allows for some versatility in gain selection.

5. Final Condition Virtual State Difference Term

The final condition virtual state difference term aims to minimize the difference between the final virtual state variable $\boldsymbol{v}_N$ and the final trajectory solution variable $\boldsymbol{x}_N$. When both of those state terms are equivalent to each other, it ensures that the projected final vehicle state is going to meet all the terminal condition constraints that have been enforced. From a mission operations perspective, these final state constraints carry more significance to mission success than the other state constraints that are enforced throughout the trajectory, since they ensure a clean transition from the approach phase to the vertical descent phase of the mission. Having the final condition virtual state difference term separated from the other terms allows for another unique coefficient gain k_B that can be designed to have a heavier weighting than the other virtual difference terms and the selection of this gain is verified during tuning and testing.

6. Final Horizontal Miss Distance Term

The SPLICE DQG algorithm does not enforce a hard equality constraint on the vehicle's final position in the approach phase, instead there is some slack in the problem which allows for some deviation of final horizontal position (in this case, position in the East and North direction). This slack term $\boldsymbol{\delta}_r$, which was first introduced in the initial position constraint Eq. (21), is then added on to the cost function with an associated coefficient gain k_{miss}. Thus, even though we allow for some position miss, the solver will make every attempt to minimize that miss distance. We can then select an appropriate coefficient gain to tune the algorithm's performance in testing.

I. Full Optimization Problem Overview

$$\text{minimize} \quad -k_m x_{m_N} + k_\eta[(\boldsymbol{x}-\bar{\boldsymbol{x}})^2 + (\boldsymbol{u}-\bar{\boldsymbol{u}})^2 + (\boldsymbol{v}-\bar{\boldsymbol{x}})^2] + k_\sigma(\sigma-\bar{\sigma})^2 + k_s(\boldsymbol{v}_{1:N-1} - \boldsymbol{x}_{1:N-1})^2 + k_B(\boldsymbol{v}_N - \boldsymbol{x}_N)^2 + k_{miss}\boldsymbol{\delta}_r$$

$$\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}, \sigma, \boldsymbol{\delta}_r$$

subject to

Initial State

$$\begin{aligned} x_{m_0} &= m_{IC} \\ \boldsymbol{x}_{\boldsymbol{q_0}} &= \boldsymbol{q}_{IC} \\ \boldsymbol{x}_{\boldsymbol{v_0}} &= \boldsymbol{v}_{IC} \\ \boldsymbol{x}_{\boldsymbol{\omega_0}} &= \boldsymbol{\omega}_{IC} \end{aligned} \tag{56}$$

Initial Position with slack

$$\begin{aligned} \boldsymbol{x}_{\boldsymbol{q}_{r_0}} + \frac{1}{2}[\boldsymbol{q}_{IC}]^*_\otimes \boldsymbol{\delta}_r &= \frac{1}{2}\boldsymbol{r}_{ENU_{IC}} \otimes \boldsymbol{q}_{IC} \\ \delta_U &= 0 \end{aligned} \tag{57}$$

Dynamics

$$\boldsymbol{x}_{k+1} - \bar{\boldsymbol{x}}_{k+1} = \boldsymbol{A}_k(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_k) + \boldsymbol{B}_k(\boldsymbol{u}_k - \bar{\boldsymbol{u}}_k) + \boldsymbol{B}_{k+1}(\boldsymbol{u}_{k+1} - \bar{\boldsymbol{u}}_{k+1}) + \boldsymbol{S}_k(\sigma - \bar{\sigma}) + \Delta_{k+1} \tag{58}$$

Thrust

$$\begin{aligned} T_{min} \le u_{T_{Z_k}} &\le T_{max} \\ u_{T_{X_k}} = 0, \quad u_{T_{Y_k}} &= 0 \end{aligned} \tag{59}$$

Thrust Rate

$$-\dot{T}_{max}\sigma \le \left(u_{T_{Z_k}} - u_{T_{Z_{k-1}}}\right)(N-1) \le \dot{T}_{max}\sigma \tag{60}$$

Torque

$$\begin{aligned} -\tau_{X_{max}} \le u_{\tau_{X_k}} &\le \tau_{X_{max}} \\ -\tau_{Y_{max}} \le u_{\tau_{Y_k}} &\le \tau_{Y_{max}} \\ -\tau_{Z_{max}} \le u_{\tau_{Z_k}} &\le \tau_{Z_{max}} \end{aligned} \tag{61}$$

Altitude

$$-\bar{\tilde{\boldsymbol{q}}}_k{}^T \boldsymbol{M}_G \bar{\tilde{\boldsymbol{q}}}_k + r_{alt} + \left[-2\bar{\tilde{\boldsymbol{q}}}_k{}^T \boldsymbol{M}_G\right](\boldsymbol{v}_{\tilde{\boldsymbol{q}}_k} - \bar{\tilde{\boldsymbol{q}}}_k) \le 0 \tag{62}$$

Global Tilt Angle

$$\boldsymbol{v}_{\tilde{\boldsymbol{q}}_k}^T \boldsymbol{M}_T \boldsymbol{v}_{\tilde{\boldsymbol{q}}_k} + \cos(\theta_{max}) - 1 \le 0 \tag{63}$$

Global Angular Rate

$$\begin{aligned} -\omega_{X_{max}} \le v_{\omega_{X_k}} &\le \omega_{X_{max}} \\ -\omega_{Y_{max}} \le v_{\omega_{Y_k}} &\le \omega_{Y_{max}} \\ -\omega_{Z_{max}} \le v_{\omega_{Z_k}} &\le \omega_{Z_{max}} \end{aligned} \tag{64}$$

State-Triggered Velocity

$$s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})(\|\bar{\boldsymbol{v}}_k\|_2 - v_{HD\,max}) + s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})(\|\bar{\boldsymbol{v}}_k\|_2 - v_{HD\,max})(\boldsymbol{v}_{\tilde{\boldsymbol{q}}_k} - \bar{\tilde{\boldsymbol{q}}}_k) + \dot{s}_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\left(\frac{\bar{\boldsymbol{v}}_k}{\|\bar{\boldsymbol{v}}_k\|_2}\right)(\boldsymbol{v}_{\tilde{\boldsymbol{\omega}}_k} - \bar{\tilde{\boldsymbol{\omega}}}_k) \le 0 \tag{65}$$

State-Triggered Angular Rate

$$\begin{aligned} s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})(\|\bar{\boldsymbol{\omega}}_k\|_2 - \omega_{HD\,max}) + \left[\left(\dot{s}_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}}) + s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\dot{s}_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\right)(\|\bar{\boldsymbol{\omega}}_k\|_2 - \omega_{HD\,max})\right](\boldsymbol{v}_{\tilde{\boldsymbol{q}}_k} - \bar{\tilde{\boldsymbol{q}}}_k) \\ + \left[s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\frac{\bar{\boldsymbol{\omega}}_k}{\|\bar{\boldsymbol{\omega}}_k\|_2}\right](\boldsymbol{v}_{\tilde{\boldsymbol{\omega}}_k} - \bar{\tilde{\boldsymbol{\omega}}}_k) \le 0 \end{aligned} \tag{66}$$

State-Triggered Line of Sight

$$\begin{aligned} s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})f_\varphi(\bar{\tilde{\boldsymbol{q}}}_k) \\ + \left[s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\dot{s}_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})f_\varphi(\bar{\tilde{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\dot{s}_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\,f_\varphi(\bar{\tilde{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\frac{\partial f_\varphi(\bar{\tilde{\boldsymbol{q}}}_k)}{\partial \bar{\tilde{\boldsymbol{q}}}_k}\right](\boldsymbol{v}_{\tilde{\boldsymbol{q}}_k} \\ - \bar{\tilde{\boldsymbol{q}}}_k) \le 0 \end{aligned} \tag{67}$$

State-Triggered Glide Slope

$$\begin{aligned} s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})f_\gamma(\bar{\tilde{\boldsymbol{q}}}_k) \\ + \left[s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\dot{s}_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})f_\gamma(\bar{\tilde{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\dot{s}_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\,f_\gamma(\bar{\tilde{\boldsymbol{q}}}_k) + s_1(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})s_2(\bar{\boldsymbol{q}}_{\boldsymbol{r_k}})\frac{\partial f_\gamma(\bar{\tilde{\boldsymbol{q}}}_k)}{\partial \bar{\tilde{\boldsymbol{q}}}_k}\right](\boldsymbol{v}_{\tilde{\boldsymbol{q}}_k} \\ - \bar{\tilde{\boldsymbol{q}}}_k) \le 0 \end{aligned} \tag{68}$$

Final Mass

$$x_{m_N} \ge m_{TDI} \tag{69}$$

Final Tilt Angle

$$\boldsymbol{v}_{\tilde{\boldsymbol{q}}_N}^T \boldsymbol{M}_T \boldsymbol{v}_{\tilde{\boldsymbol{q}}_N} + \cos(\theta_F) - 1 \le 0 \tag{70}$$

Final Position

$$\boldsymbol{x}_{\boldsymbol{q}_{r_N}} = \frac{1}{2}\boldsymbol{r}_{ENU_F} \otimes \boldsymbol{v}_{\boldsymbol{q_N}} \tag{71}$$

Final Horizontal Velocity

$$\left\|\boldsymbol{v}_{\boldsymbol{v}_{EN_N}}\right\|_2 \le v_{EN\,max} \tag{72}$$

Final Vertical Velocity

$$-v_{U\,max} \le v_{v_{U_N}} \le 0 \tag{73}$$

Final Tilt Rate

$$\left\|\boldsymbol{v}_{\boldsymbol{\omega}_{XY_N}}\right\|_2 \le \omega_{XY_F} \tag{74}$$

Final Roll Rate

$$-\omega_{Z_F} \le v_{\omega_{Z_N}} \le \omega_{Z_F} \tag{75}$$

J. Behcet Second Order Cone Program

The SPLICE DQG application utilizes the Behcet Second Order Cone Program (BSOCP) which has been provided by the University of Washington to NASA. BSOCP is a software package that contains a Custom Parser (CPRS) and a generic Second Order Cone Program (SOCP) which are both implemented in C++. Like most available SOCPs, the optimization problem must be provided in a generic form. However, expressing the 6-DOF rocket guidance problem in a generic form requires a tedious parsing process to transform the entire problem (shown in Section I) into a generic set of inputs. The Custom Parser piece of BSOCP allows for this parsing process to be automated. This allows developers to rapidly prototype multiple desired constraints and express each of them individually, rather than combining them into a generic form.

Both the Custom Parser and generic SOCP utilize dynamic memory allocation to allow for flexibility in parsing and solving problems. The use of dynamic memory allocation is not permitted for flight software. Thus, the parsing process only occurs once at application initialization and does not execute any re-parses during flight. Execution of the parser at initialization not only reduces the risk associated with dynamic memory allocation, but it also improves the DQG computation time since parsing is only completed once. This improvement is also shown in [16], where a real-time DQG algorithm benefitted from a customized pre-parsed problem. Additionally, the generic SOCP contains a feature to generate a customized version of the SOCP in C. This customized version does not utilize dynamic memory allocation, and typically results in a faster algorithm [17]. The results and performance shown in this publication all utilized this customized version of the SOCP.

BSOCP is executed on each DQG successive convexification iteration. When BSOCP determines a solution to the optimization problem, that solution becomes the new reference trajectory which then gets utilized to determine the updated dynamic models and constraint parameters.

IV. Guidance Target State Generator

Guidance Target State Generator (GTSG) is a software application needed for the practical reality of interfacing the DQG algorithm with the host vehicle control. It is necessary to bridge the gap between a high-rate controller and the comparatively low-rate execution of DQG. At each execution, GTSG will interpolate the DQG trajectory into a target state to be used downstream by the stand-in Host Vehicle controller.

DQG is expected to provide GTSG with an output trajectory and a reference time based on the time kept by the SPLICE navigation application (NAV). GTSG receives DQG trajectories as they become available as well as a time reference from NAV. It will interpolate the DQG trajectory to the current time reference and make this output available for the control system.

The actual interpolation is done using a linear interpolation and spherical linear interpolation algorithm for cartesian and quaternion members of the output trajectory respectively. Below are the equations for when the current time t is between two points in the trajectory $t_n < t < t_{n+1}$.

Linear Fraction

$$f(t) = \frac{t - t_n}{t_{n+1} - t_n} \tag{76}$$

Linear Interpolation

$$x(t) = f(t) * x_{n+1} + \big(1 - f(t)\big) * x_n \tag{77}$$

SLERP

$$ang = dot(q_{n+1}, q_n)$$

$$scl_{n+1} = \frac{\sin\left(f(t) * ang\right)}{\sin(ang)}, scl_n = \frac{\sin\left((1 - f(t)) * ang\right)}{\sin(ang)} \tag{78}$$

$$q(t) = scl_{n+1} * q_{n+1} + scl_n * q_n$$

V.Trick Simulation

SPLICE has developed a six degree of freedom, Monte Carlo capable lunar landing simulation for testing GNC algorithm performance. To simulate the vehicle, the vehicle's sensors, and the lunar environment, the simulation uses NASA Trick with the Johnson Space Center Engineering Orbital Dynamics (JEOD) simulation package. NASA Trick is a simulation development framework developed at Johnson Space Center (JSC) with the purpose of providing a common set of simulation capabilities. JEOD provides tools for simulating vehicles in an orbital environment.

For lunar gravity, the sim uses JEOD's gravity model with LP150Q spherical harmonic gravitational coefficient data. Degree and order are both set to 8. Earth gravity and Sun gravity are also included in the Lunar sim. Both are assumed to be spherical. The simulation does not model solar radiation pressure, propellant slosh, or vehicle flex.

The simulation begins with the vehicle in a near-circular orbit around the Moon, just prior to deorbit. It ends when the vehicle is 30 m above the Lunar surface. It does not continue all the way to touchdown, because SPLICE is baselined to hand off terminal descent to the host vehicle.

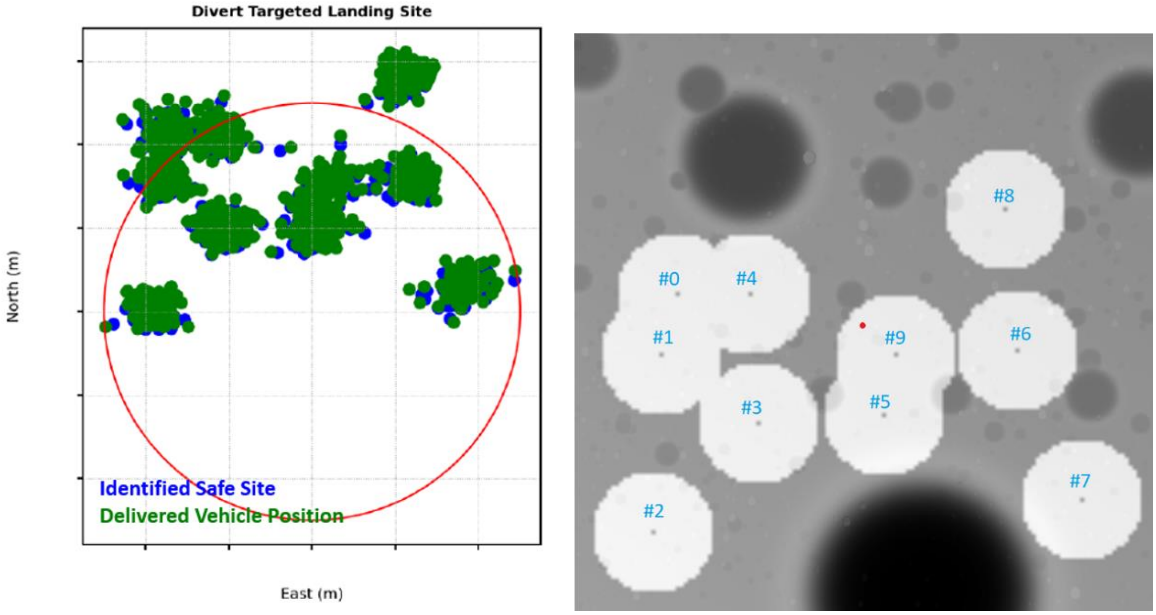
The simulation uses two different sets of GNC algorithms, referred to as Host Vehicle GNC and SPLICE GNC. SPLICE GNC uses the algorithms from the SPLICE Flight Software, including DQG and GTSG. Host Vehicle GNC uses a different set of algorithms, which are meant to represent the host vehicle's GNC. Both sets of algorithms run throughout the entire duration of the simulation, but only one set at a time is used to command the vehicle's effector models. When the simulation begins, Host Vehicle GNC has control of the vehicle. Once the vehicle enters powered descent mode, SPLICE GNC takes over until the end of the simulation.

The SPLICE GNC controller provides commands to the vehicle in the form of a force vector and an array of on-times. The force vector is used to set the main engine's thrust magnitude, and the on-times are used to actuate the reaction control thrusters. Thruster valves are assumed to open and close instantaneously, and thruster failures are not modeled.

VI. Results

A Monte Carlo analysis of 1000 runs was completed using the lunar Trick simulation, which began during the braking phase. State dispersions were initialized with 3-σ limits of 100 meters per axis in initial position, 1.5 meters per second per-axis in initial velocity, and 0.5 degrees per-axis in vehicle attitude. These were all completed using perfect navigation. A single hazard detection lidar scan is attempted during the approach phase, which is completed using a single Digital Elevation Map (DEM). For each sim run, the scan occurs at the same predetermined time which was selected so that it occurs during the targeted slant range window. The 100 meters by 100 meters DEM is projected onto the lunar surface, centered on the point where the HD boresight vector intersects with the lunar surface, and a random safe site from the map is selected for each run.

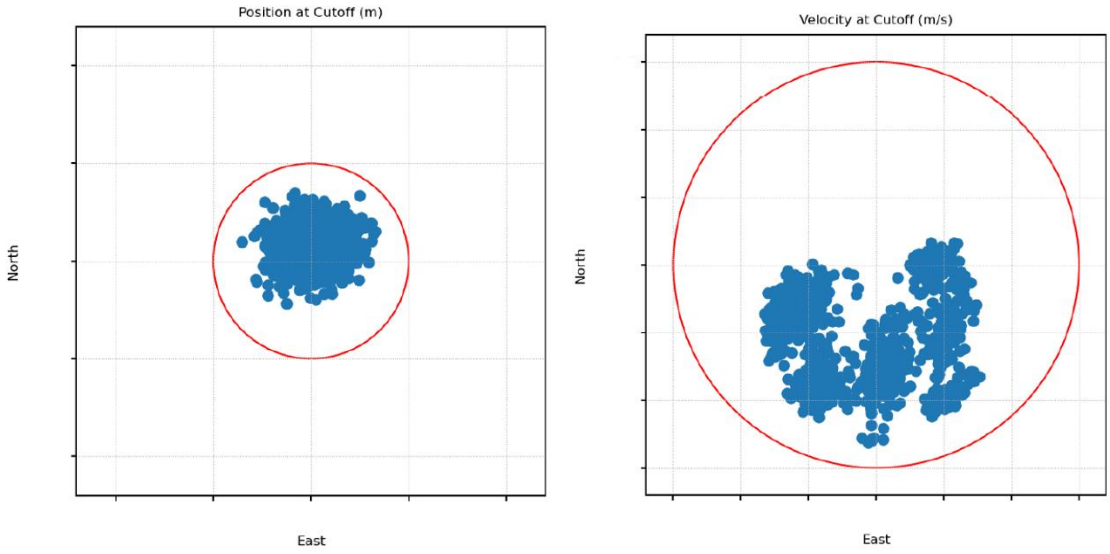
A. Terminal Target Performance



(a) (b)

Fig. 5 (a) Target Performance relative to Intended Landing Site, (b) DEM Map with available safe sites

A map of the identified safe sites along with the actual delivered vehicle positions at the terminal descent altitude relative to the original intending landing site is shown in Fig. 5 (a). The red circle in this map indicates the region where the expected identified safe sites should exist if the HDL boresight vector were pointed exactly at the original landing site during the scan. The map of the available safe sites in the DEM is shown in Fig. 5 (b). We can observe that the identified safe sites cluster in a similar pattern to where these locations exist within the DEM map. However, there does appear to be some northern offset since there should be more identified safe sites in the southern half of the map shown in Fig. 5 (a). This offset could be due to an HDL pointing error that is large enough to cause a significant positional offset that prevents the HDL boresight vector from pointing exactly at the intended landing site. In our current configuration, this offset pointing error is not accounted for and thus identified safe sites are being biased by this offset. As the SPLICE GNC system matures this bias will be accounted for and identified safe sites will target the true safe sites in the DEM. We will provide a review of the pointing error constraint later in this section to help show why this northerly offset in the identified landing sites.



(a) (b)

Fig. 6 (a) Vehicle position at sim cutoff (terminal descent initiation, (b)Vehicle velocity at sim cutoff

The North and East position at the terminal descent altitude relative to the identified safe sites are shown in Fig. 6 (a). The red circle defines our position accuracy requirement at the terminal descent altitude relative to the identified safe site. This plot shows that we meet this final position requirement for all 1000 dispersed simulations. Similarly, Fig. 6 (b) displays the terminal horizontal velocity at the terminal descent altitude, where the red circle indicates the maximum horizontal velocity requirement. This performance metric is also adequately enforced for all 1000 simulations.

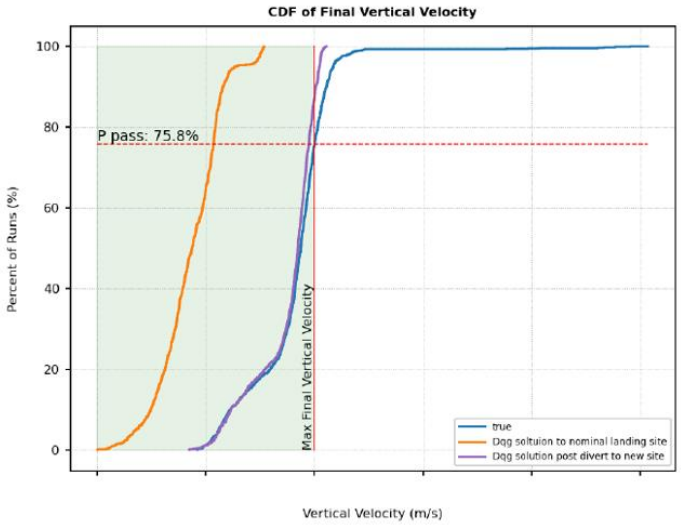


Fig. 7 Final Vertical Velocity CDF

A Cumulative Distribution Function (CDF) plot describing the performance of the vertical velocity constraint at the terminal descent altitude is shown in Fig. 7. The trajectories generated by DQG at the start of the approach phase and targeting the intended landing site, all properly enforced the final vertical velocity constraint. After the divert was

commanded and DQG provided a new trajectory, approximately 15% of them failed to meet the terminal vertical velocity constraint. These failures may be occurring because some of the identified safe sites were further away from the desired divert region and thus more difficult to reach. The identified safe sites chosen during each run were also chosen at random and did not consider the vehicle's current state. When observing the true vertical velocity state at the target, approximately 25% of the runs failed to meet the constraint. The additional constraint failures here are most likely due to the absence of translational control, and some controller tracking errors on attitude. For future analysis and mission applications, DQG should enforce a more restrictive constraint internally to provide some margin on constraint exceedance. Additionally, resolving closed-loop translational control should also play a significant impact in the reduction of constraint failures.

B. Control Constraint Enforcement

Overall DQG did an exceptional job of ensuring all the control constraints always remain enforced during the 1000 simulations. This is to be expected as there are no slack variables applied to these control constraints and utilize a first order hold discretization strategy. This means that all the control commands are guaranteed to meet the enforced constraints. Fig. 8 (a) displays the main engine thrust magnitude commands, where the red horizontal threshold limits are shown for the minimum and maximum limits. Fig. 8 (b) shows the resulting thrust rate and a similar set of red horizontal thresholds for the positive and negative limitations placed on thrust rate. Finally, Fig. 9 shows the attitude control system torque commands per body-axis. The torque command limits are also shown in the horizontal red threshold limits. There are a few runs where the commanded pitch torque slightly exceeded the upper limit. This is because the attitude controller in use does not enforce the torque limitation, and it desired more torque capability to account for some error that it was experiencing. The actual DQG control trajectory did not command the torque exceedance itself.

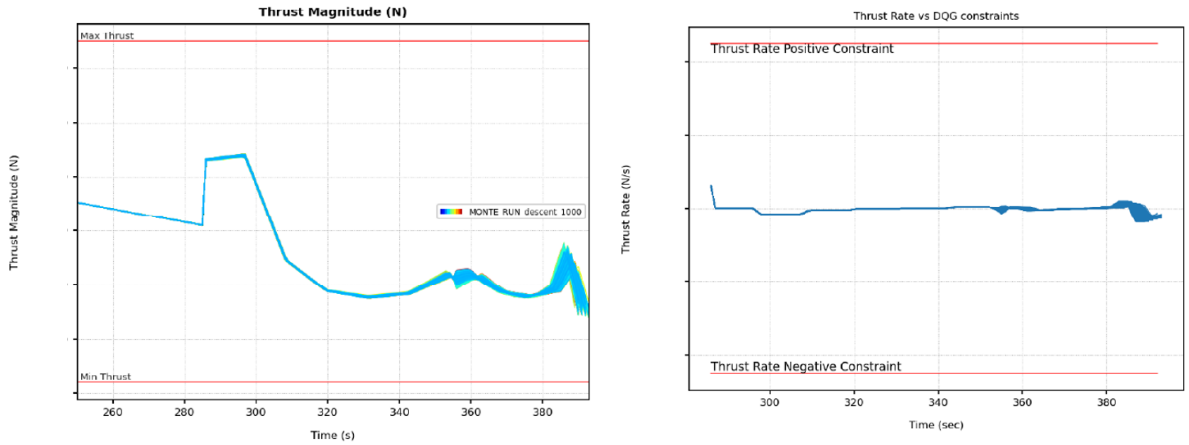


Fig. 8 (a) Main Engine Thrust Magnitude, (b) Main Engine Thrust Rate

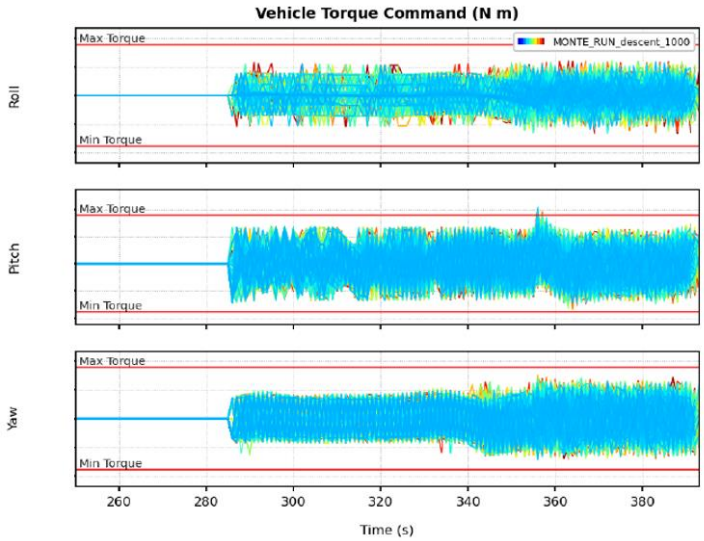


Fig. 9 Vehicle Torque Commands per body axis

C. Global State Constraint Enforcement

The vehicle tilt angle is plotted versus simulation time for all runs in Fig. 10 (a). The red horizontal threshold limit is the global maximum tilt angle, while the dashed red horizontal threshold limit is the maximum terminal tilt angle constraint at the terminal descent altitude. For all simulated runs the vehicle never exceeds the global maximum tilt angle, and the effects of the divert command after 350 seconds are also observed. A zoomed in plot of the terminal tilt angle is shown in Fig. 10 (b), which shows that about half of the runs do fail to meet the terminal tilt angle constraint.

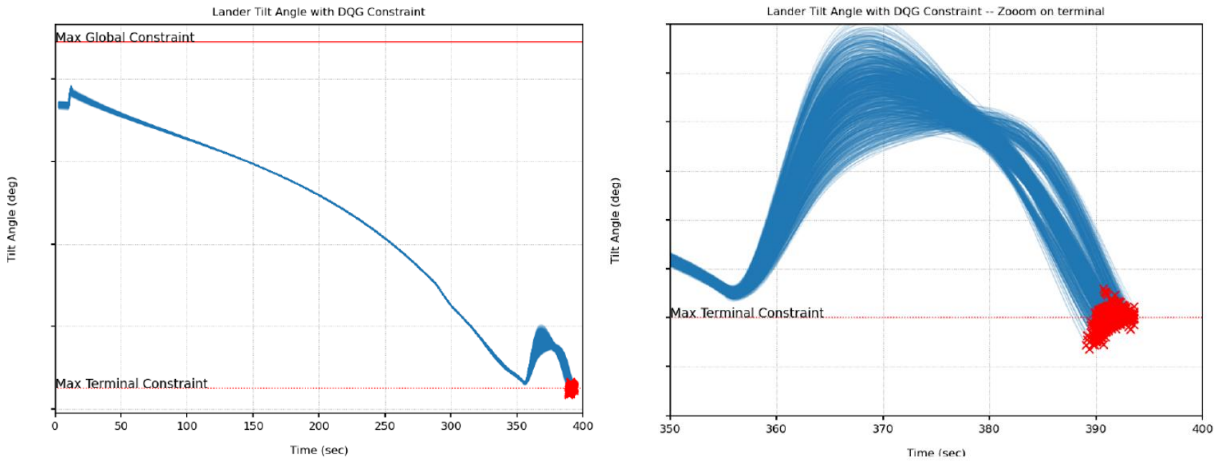


Fig. 10 (a) Vehicle Tilt Angle, (b) Zoomed in Final Vehicle Tilt Angle

A CDF plot is also provided in Fig. 11, which displays the performance of constraint enforcement from both the DQG generated solution and the actual vehicle performance in simulation. The trajectory solution that was computed at the start of the approach phase always met the final tilt angle constraint. However, when the divert was commanded, about 15% of the provided trajectories did fail the final tilt constraint. These failures probably occurred for identified safe sites that were more difficult to reach given the current vehicle state. Finally, the true terminal vehicle state did not perform to the standards that DQG requested, approximately 48.2% of runs failed to meet the terminal tilt constraint.

The large increase in failures on the true vehicle state indicates that there are still lingering issues in the vehicle tracking of the DQG trajectory. Although the constraint being violated is an attitude constraint, the failures here may also be directly related to the lack of a translational controller. This is because the simulation terminates whenever the vehicle's true altitude reaches the terminal descent initiation altitude. Since the current stand-in controller does not provide translational control, it was observed that a set of runs had significant altitude errors near the end of the trajectory that signified the true vehicle altitude was lower than the DQG desired altitude. This behavior would force the simulation to terminate earlier than DQG anticipated, and thus there may be a second or two remaining in the DQG trajectory that would have reduced the tilt angle prior to reaching the desired target. For future applications, the DQG enforced limit should be set to a value less than the actual desired terminal tilt constraint to account for any controller errors.

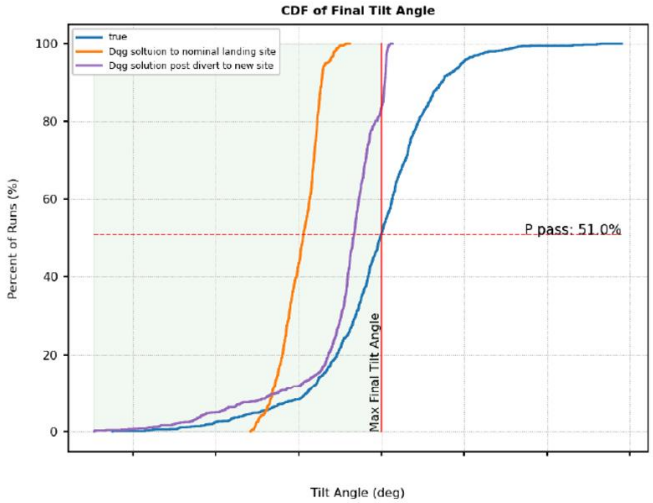
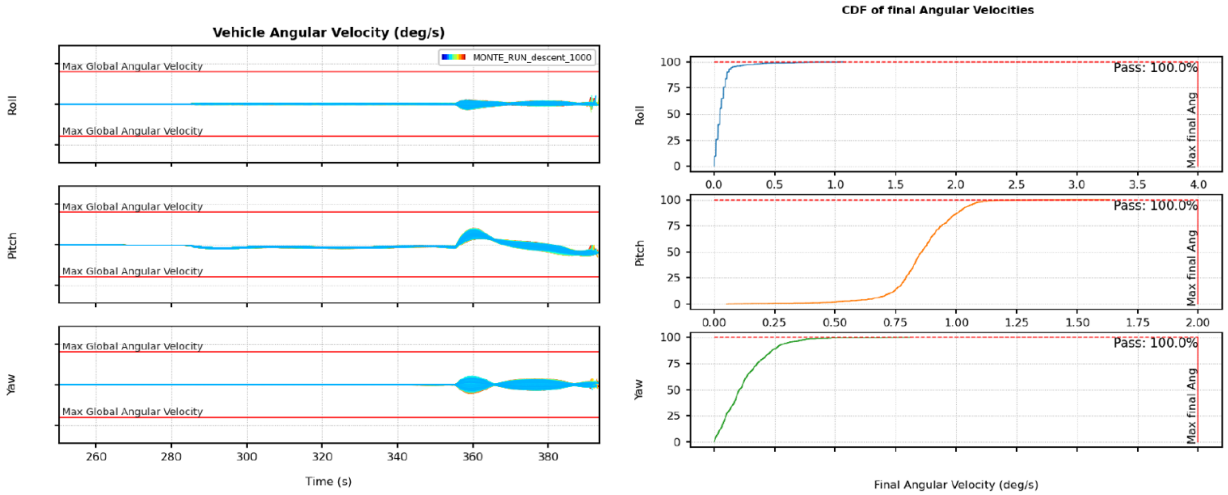


Fig. 11 Vehicle Final Tilt Angle CDF

The vehicle angular rate per body axis as a function of simulated time for all runs is shown in Fig. 12 (b). Each body axis direction has the same minimum and maximum global angular velocity constraint which is labeled in these plots as one set of red horizontal limits. Additionally, Fig. 12 (b) shows a CDF plot for the enforcement of the terminal angular rate constraints in each axis. For all runs, the angular velocity remained within the enforced limits for both the global and the final angular velocity constraints.

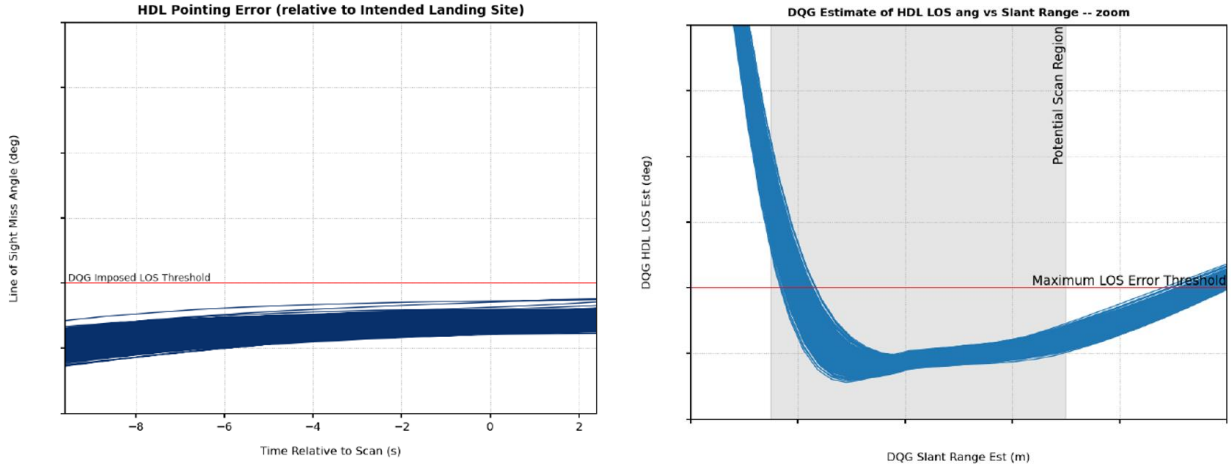


(a) (b)

Fig. 12 (a) Vehicle Angular Velocity per body axis, (b) Final Angular Velocity CDF

D. Hazard Detection Scan Constraint Enforcement

The hazard detection scan constraints are enforced to ensure that a healthy scan of the intended landing site can be completed, and a divert can be commanded to a safe site. The performance is shown as a function of time relative to the scan. After the scan occurs, these constraints are deactivated because in all runs a new safe site is identified and the divert command is issued. Following the divert command, there are no additional scans. The HDL pointing error to the landing site as a function of time relative to the scan is shown in Fig. 13 (a). DQG does attempt to compute a trajectory with no pointing error during the scan window, however in simulation we do observe some pointing error that is below our initial set threshold which is highlighted by the red horizontal limit. Despite being lower than the desired pointing error threshold, it does seem to cause a positional offset which was observed in Fig. 5 (a). There are several reasons for why this pointing error exists. Firstly, an assumption was made during the constraint derivation which assumed that the HDL boresight sensor is located at the vehicle center of gravity. In reality, and in simulation, this is not true and may cause a small error contribution. The line of sight pointing constraint is also a coupled constraint in position and attitude, however we only currently utilize an attitude controller to track the desired DQG trajectory. Thus, additional tracking errors may also play a role in the pointing error being observed. Lastly, the DQG line of sight constraint does utilize virtual state slack variables and is not guaranteed to be enforced but the constraint exceedance is minimized by the cost function in equation (*55*). Fig. 13 (b) shows the HDL pointing error of the trajectory generated by DQG at the start of the approach phase. A zero-degree pointing error was set as the line-of-sight constraint boundary, and DQG was allowed to minimize the exceedance of this constraint for nodes within the scan window highlighted in grey. The resulting trajectory that was computed however, did need to exceed this zero-degree threshold. Thus, a significant contribution to the pointing error comes from the optimized solution computed by DQG, but there is also some evidence that additional error is associated with other factors such as controller tracking performance and constraint assumptions. This evidence is shown in Fig. 14, which shows that the true HDL pointing error is larger than the DQG projected pointing error when the scan occurs.



(a) (b)

Fig. 13 (a) HDL Pointing Error relative to the scan time, (b) DQG projected HDL pointing error relative to projected slant range

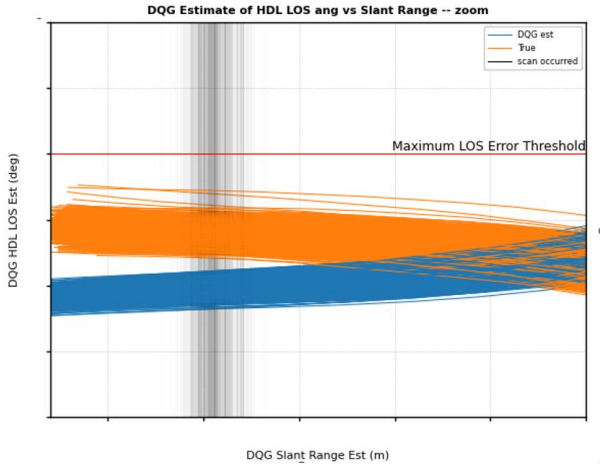


Fig. 14 True HDL Pointing Error and DQG Projected HDL Pointing Error as a function of slant range

The maximum velocity is limited during the scan, and this constraint is shown in Fig. 15. The vehicle's velocity magnitude remains well below the imposed constraint limitation which is highlighted by the red horizontal threshold.

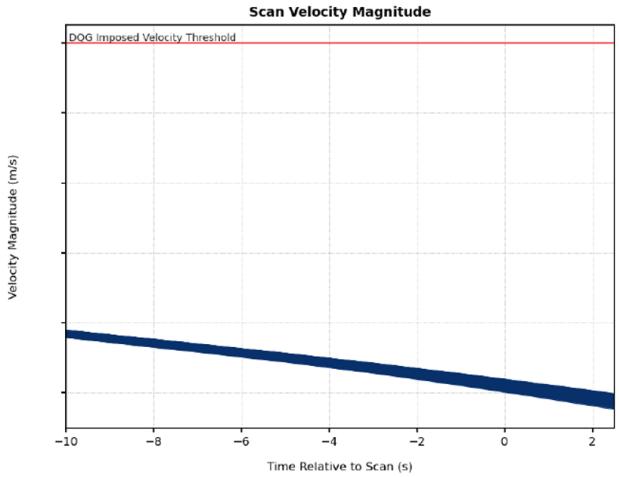
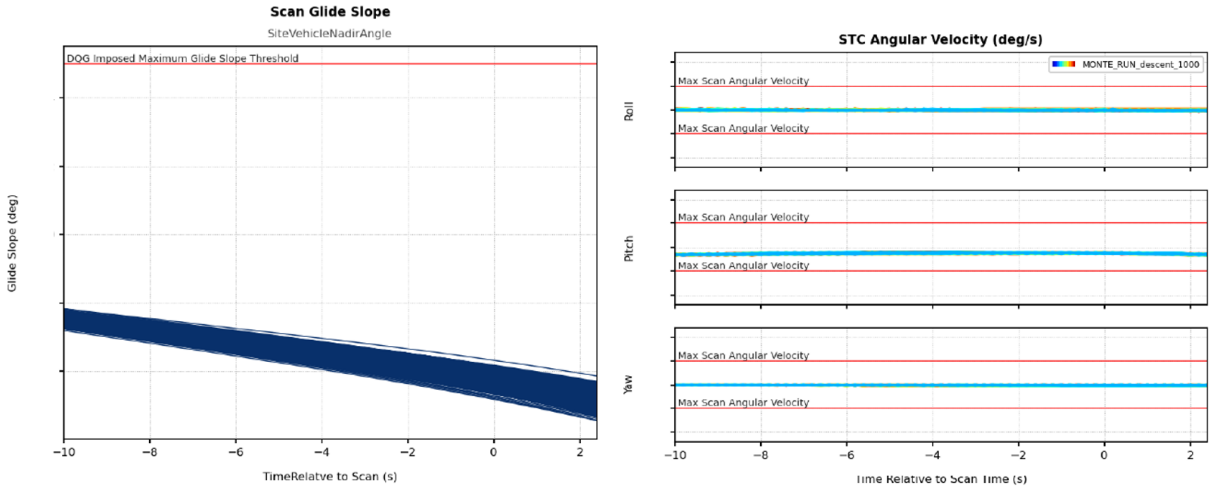


Fig. 15 Vehicle Velocity Magnitude relative to the scan time

The vehicle glide slope measured as the angle between the Up axis of the ENU frame at the intended landing site to the vehicle's position vector in the same frame. To ensure the image is not too oblique, the glide slope is constrained during the scan and is shown in Fig. 16 (a). As seen in this plot, the glide slope remains below the maximum threshold prior to the scan. Additionally, the vehicle's angular velocity is also constrained such that it is even more restrictive than the global and terminal thresholds for angular velocity. This angular velocity scan region constraint is shown in Fig. 16 (b) and shows that we can meet the constraint during the scan.



(a) (b)

Fig. 16 (a) Vehicle Glide Slope relative to the scan time, (b) Vehicle Angular Velocity relative to the scan time

VII. Conclusion

The results show that DQG can produce approach phase trajectories that adheres to all state and vehicle constraints and is capable of safely diverting to an identified safe site with great accuracy. Despite this success, there is still room for improvement regarding some of the final target constraints and most importantly the HDL line of sight constraint. Some of the constraint exceedance shown for final vertical velocity, final vehicle tilt angle, and the HDL pointing error may be cleared up by making improvements to the control algorithm to allow for translational tracking and improved attitude tracking. These final state constraints should also be re-designed to provide safe margin on the constraint boundaries that would account for any potential controller tracking errors. It would also be beneficial to reduce the HDL pointing error as much as possible. Further updates to the cost function and the virtual state variables associated with the HDL line of sight constraint may allow for a greater representation of this constraint error in the cost function to allow for error minimization. Finally, we are also interested in completing this analysis with a customized Proportional-Integral Projected Gradient (PIPG) solver, which is a first order solver that we expect will provide significant benefit for real-time applications [14] [18]. In addition to faster solution times, the use of a custom pre-conditioner algorithm shown in [14] may also improve the HDL line of sight constraint exceedance that may be underrepresented in our current cost function.

Acknowledgements

We would like to thank Gregory Barton and Breanna Johnson who also provided discussion and review support of the DQG performance results shown here. This work was authored by employees of Draper under Contract No. 80JSC021DA005 with the National Aeronautics and Space Administration. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, or allow others to do so, for United States Government purposes.

References

[1] Sostaric, R. R., Pedrotty, S., Carson, J. M., Estes, J. N., Amzajerdian, F., Dwyer-Cianciolo, A. M., and Blair, J. B., "The SPLICE Project: Safe and Precise Landing Technology Development and Testing," AIAA SciTech 2021 Forum, 2021.

[2] Klumpp, A. R., "Apollo Lunar-Descent Guidance," Charles Stark Draper Laboratory, Cambridge, MA, 1971.

[3] Açıkmeşe, B., Carson, J. M., and Blackmore, L., "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem," IEEE Transactions on Control Systems Technology, Vol. 21, No. 6,2013, pp. 2104–2113.

[4] Acikmese, B., Casoliva, J., Carson, J., and Blackmore, L., "G-FOLD: A Real-Time Implementable Fuel Optimal Large Divert Guidance Algorithm for Planetary Pinpoint Landing," Concepts and Approaches for Mars Exploration, Vol. 1679, 2012, p.4193.

[5] Berning, A. W., Strohl, L., and Bieniawski, S. R., "Lossless convex guidance for lunar powered descent," AIAA SciTech 2023 Forum, 2023.

[6] Acikmese, B., Aung, M., Casoliva, J., Mohan, S., Johnson, A., Scharf, D., Masten, D., Scotkin, J., Wolf, A., and Regehr, M. W., "Flight Testing of Trajectories Computed by G-FOLD: Fuel Optimal Large Divert Guidance Algorithm for Planetary Landing," AAS/AIAA spaceflight mechanics meeting, 2013.

[7] Szmuk, M., and Acikmese, B., "Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time," 2018 AIAA Guidance, Navigation, and Control Conference, 2018.

[8] Szmuk, M., Reynolds, T. P., and Açıkmeşe, B., "Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints," Journal of Guidance, Control, and Dynamics, vol. 43, 2020, pp. 1399–1413.

[9] Reynolds, T. P., Szmuk, M., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Dual Quaternion-Based Powered Descent Guidance with State-Triggered Constraints," Journal of Guidance, Control, and Dynamics, vol. 43, 2020, pp. 1584–1599.

[10] Fritz, M., Doll, J., Ward, K. C., Mendeck, G., Sostaric, R. R., Pedrotty, S., Kuhl, C., Acikmese, B., Bieniawski, S. R., Strohl, L., and Berning, A. W., "Post-Flight Performance Analysis of Navigation and Advanced Guidance Algorithms on a Terrestrial Suborbital Rocket Flight," AIAA SCITECH 2022 Forum, 2022.

[11] Strohl, L., Doll, J., Fritz, M., Berning, A. W., White, S., Bieniawski, S. R., Carson, J. M., and Acikmese, B., "Implementation of a Six Degree of Freedom Precision Lunar Landing Algorithm Using Dual Quaternion Representation," AIAA SciTech 2022 Forum, 2022.

[12] Ward, K., Smith, K., Rowe, I. Harper, J., Adams, D., Pedrotty, S., Mendeck, G., "Lidar-Based Safe Site Relative Navigation," AIAA SciTech 2024 Forum, 2024.

[13] Malyuta, D., Reynolds, T., Szmuk, M., Mesbahi, M., Acikmese, B., and Carson, J. M., "Discretization Performance and Accuracy Analysis for the Rocket Powered Descent Guidance Problem," AIAA SciTech 2019 Forum, 2019.

[14] Kamath, A. G., Elango, P., Mceowen, S., Yu, Y., Carson, J. M., Mesbahi, M., and Acikmese, B., "Customized Real-Time First-Order Methods for Onboard Dual Quaternion-Based 6-DOF Powered-Descent Guidance," AIAA SCITECH 2023 Forum, 2023.

[15] Reynolds, T., Szmuk, M., Malyuta, D., Mesbahi, M., Acikmese, B., and Carson, J. M., "A State-Triggered Line of Sight Constraint for 6-DOF Powered Descent Guidance Problems," AIAA SciTech 2019 Forum, 2019.

[16] Reynolds, T., Malyuta, D., Mesbahi, M., Acikmese, B., and Carson, J. M., "A Real-Time Algorithm for Non-Convex Powered Descent Guidance," AIAA Scitech 2020 Forum, 2020.

[17] Dueri, D., Zhang, J., and Açikmese, B., "Automated Custom Code Generation for Embedded, Real-time Second Order Cone Programming," IFAC Proceedings Volumes, vol. 47, 2014, pp. 1605–1612.

[18] Yu, Y., Elango, P., Topcu, U., and Açıkmeşe, B., "Proportional–integral projected gradient method for conic optimization," Automatica, vol. 142, 2022, p. 110359.