Post-Flight Performance Analysis of Navigation and Advanced Guidance Algorithms on a Terrestrial Suborbital Rocket Flight

Matthew P. Fritz[1, *], Javier A. Doll[1, *], Kari C. Ward[1, *], Gavin Mendeck[2,†], Ronald Sostaric[2, ‡], Samuel Pedrotty[2, §], Behçet Açikmeşe[2,**], Christopher Kuhl[3, ††], Stefan Bieniawski[4,‡‡], Lloyd Strohl, III[4,§§], Andrew Berning Jr.[4, ***]

[1]*The Charles Stark Draper Laboratory*
[2]*NASA Johnson Space Center*
[3]*NASA Langley Research Center*
[4]*Blue Origin, LLC*

There is currently renewed interest in robotic and crewed landers for a return to the lunar surface. Advanced guidance and navigation algorithms are essential to accurately delivering cargo and crew safely to the moon successfully. This paper reports the overall performance of an integrated set of navigation and guidance algorithms flown on a terrestrial suborbital rocket up to an altitude of approximately 100km. The navigation algorithm consists of an onboard extended Kalman Filter (EKF) that ingests multiple sensor measurements, one of which is the output from a terrain relative navigation (TRN) algorithm that cross-references camera images to on-board satellite imagery to perform feature correlation within the camera image. The guidance algorithm solves for a 6-degree-of-freedom (DoF) optimal trajectory using a successive convexification method during powered descent. The altitude range as well as the landing dynamics experienced during this test flight are realistic for an extraterrestrial landing and provide an invaluable data set to gauge the current development of these landing algorithms in an effort to advance the overall software readiness levels (SRL). This paper will delve into different aspects of each algorithm and present an analysis of the in-flight performance of the algorithms. This flight was conducted under the National Aeronautics and Space Administration (NASA) Safe and Precise Landing Integrated Capabilities Evolution (SPLICE) project focused on technology advancement for landing applications.

Nomenclature

- Bold face variables represent vector and matrix quantities
- x (without any kind of accent) represents the truth value of a particular state or the known value of a state, such as the time difference between two inertial measurement unit (IMU) measurements
- $\hat{x}$ represents an EKF estimated state or quantity derived from an estimated state
- $x^c_{a/b}$ defines a variable x that is defined as reference frame A origin with respect to (w.r.t.) reference frame B origin provided in reference frame C
- Subscript "k" denote the current time step while "k-n" denotes the nth time step prior to the current time step

[*] Member Technical Staff
[†] SPLICE FSW Lead
[‡] SPLICE PM, AIAA Senior Member
[§] SPLICE Deputy PM
[**] SPLICE Guidance Lead
[††] SPLICE Chief Engineer
[‡‡] Tipping Point Project Lead
[§§] Blue Origin Guidance Lead
[***] GNC Engineer

- Prior notation (state prior to an update) is represented by a negative sign, x_{k^-}, while posterior notation (state after an update) is represented by a positive sign, x_{k^+}
- $\tilde{\boldsymbol{q}}$ the tilde denotes a dual quaternion
- $\bar{\boldsymbol{x}}$ the bar denotes that it is a reference variable based on a previous iteration (can apply to state, control, and time variables)

I. Introduction

The Safe and Precise Landing Integrated Capabilities Evolution (SPLICE) project is a continuation of various programs that have studied and worked to advance various lander technologies over the years [1]. The roots of the SPLICE project can be traced back to the Autonomous Landing Hazard Avoidance Technology (ALHAT) project which was eventually incorporated into and flown as part of the Morpheus project [2] [3]. After multiple successive Morpheus campaigns, additional lander technologies advancement efforts were targeted as part of the COBALT project. Additional technology advancement was identified after the Cooperative Blending of Autonomous Landing Technologies (COBALT) project which led to the initiation of the SPLICE project [4]. The SPLICE project has successfully demonstrated its baseline terrain relative navigation (TRN) technology on the Masten Xodiac vehicle up to an altitude of 500 meters [5]. All of these efforts to date have involved aircraft and terrestrial lander applications with limited altitude ranges not exceeding 500m. While these applications have allowed for valuable closed-loop testing and algorithm validation near landing, none of these applications have been able to fully test the algorithms over the entire operational range of a representative lunar descent trajectory. In order to expand this operational test envelope, the SPLICE project was able to secure a multi-flight test campaign as a payload on the Blue Origin New Shepard suborbital launch vehicle. This launch vehicle provides the capability to test technologies of interest up to an altitude of 100km, which corresponds to the altitude of a baseline lunar orbit prior to initiation of the deorbit insertion and descent of several proposed lunar landers.

Each of the aforementioned programs (ALHAT, Morpheus, COBALT) had a slightly different navigation algorithm implemented to support testing and performance analysis of the technologies being developed. The ALHAT navigation filter utilized hazard relative navigation, navigation Doppler LiDAR (NDL) measurements and a laser altimeter coupled with IMU measurements to provide relative state navigation [2]. The Morpheus project implemented an inertial navigation filter with integrated GPS capabilities that provided accurate state knowledge to ensure successful objective completion for the mission [3]. The COBALT navigation algorithm utilized TRN and NDL range and velocity measurements along with IMU measurements to provide a state estimate as part of its open-loop test campaigns on the Masten Xodiac vehicle [4]. The SPLICE TRN flight test on the Masten vehicle utilized an onboard Multi-State Constraint Kalman Filter utilizing IMU, barometer and TRN measurements to provide open-loop state estimates [5]. The navigation approach presented in this paper uses a similar navigation scheme as the COBALT project to emulate expected navigation algorithms that will be available for non-terrestrial landers. While this paper will delve into the navigation algorithm implemented for this test flight, expanding the operational test range for the targeted sensor technology and collecting the data was the primary objective of this test flight. This paper will provide insight into the performance of the navigation algorithm utilizing post flight recreation tooling to incorporate necessary corrections to onboard parameter settings that were unable to be incorporated for flight due to schedule restrictions.

For powered descent guidance, the first look into the capability naturally began with the Apollo program. The interest in spaceflight fueled new research and development, which eventually resulted in the first rocket-powered landing by Surveyor I on a celestial body [6]. This incredible feat has inspired future generations to continue pushing the boundaries of powered descent guidance. Although optimal guidance had already been studied during the Apollo program, Apollo guidance was not optimal [7]. It was also a 3-degree-of-freedom (DoF) translational guidance algorithm for which attitude commands were computed separately. The desire for optimal powered descent guidance became apparent for robotic Martian landers. Improvements to guidance optimality would mean efficient propellant usage and would directly correlate to allowing for a larger science payload. D'Souza was able to show an analytical guidance law that minimizes acceleration as a function of time-to-go [8].

The 6-DoF guidance problem has continued to evolve in complexity. In order to have the capability of reaching scientifically interesting locations, guidance algorithms need to be capable of enforcing complex constraints in order to precisely land in potentially hazardous terrain. An alternative solution to the guidance problem emerged that formulated the problem to leverage convex programming [9] [10] [11]. These contributions have continued to receive

attention and have evolved from 3-DoF implementations using Cartesian coordinates to 6-DoF implementations expressing state information in a dual quaternion form. It has been shown that expressing state information in a dual quaternion form can transform coupled translational and attitude state constraints into a convex form. In addition a Dual Quaternion Guidance (DQG) algorithm has been shown to also be capable of handling non-convex constraints through successive convexification techniques [12]. This capability of determining optimal guidance solutions that also maintain mission, vehicle, and trajectory constraints can become fruitful for future robotic and human landers.

II. Navigation Algorithm

The navigation algorithm is based on a discretized extended Kalman filter (EKF) formulation with the *UDU* decomposition of the error state used for filter stability and computational benefits. The filter design is based on an inertial state filter approach. Using this approach, the translational states are represented in an Earch Centered Inertial (ECI) reference frame with the rotational states defined relative to the inertial reference frame. The filter implementation consists of 10 dynamic states (describing states governed by equations of motion) and 31 parameter states (unknown sensor specific states whose evolution is controlled through a set of parameters). These states are stacked into a single state vector to give a 41 state filter, as shown in Eq. (II-1) – Eq. (II-3) where $\boldsymbol{r_i}$ is the ECI position, $\boldsymbol{v_i}$ is the ECI velocity, and $\boldsymbol{q_i^{imu}}$ is the ECI to IMU case attitude quaternion. For the bias parameter states in Eq. (II-2), the $\boldsymbol{b_{V\Delta}}$ is the IMU delta velocity bias, $\boldsymbol{b_{\Delta\theta}}$ is the IMU delta angle bias, $\boldsymbol{b_{\rho_{ndl}}}$ and $\boldsymbol{b_{v_{ndl}}}$ are the NDL range and velocity biases (three for each beam), $\boldsymbol{\phi_{\frac{cam}{imu}}}$ is the TRN camera case reference frame to IMU case reference frame misalignment, b_{f_x}, b_{f_y}, b_{p_x}, and b_{p_y} are the camera biases for the X / Y focal length and X / Y principal point respectively, and $\boldsymbol{r_{f_k}^{ecef}}$ is the three-dimensional world coordinates of the k[th] TRN feature..

$$\boldsymbol{x_d} = \begin{bmatrix} \boldsymbol{r_i} & \boldsymbol{v_i} & \boldsymbol{q_i^{imu}} \end{bmatrix} \tag{II-1}$$

$$\boldsymbol{x_p} = \begin{bmatrix} \boldsymbol{b_{V\Delta}} & \boldsymbol{b_{\Delta\theta}} & \boldsymbol{b_{\rho_{ndl}}} & \boldsymbol{b_{v_{ndl}}} & \boldsymbol{\phi_{\frac{cam}{imu}}} & b_{f_x} & b_{f_y} & b_{p_x} & b_{p_y} & \boldsymbol{r_{f_1}^{ecef}} & \boldsymbol{r_{f_2}^{ecef}} & \boldsymbol{r_{f_3}^{ecef}} & \boldsymbol{r_{f_4}^{ecef}} \end{bmatrix} \tag{II-2}$$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_d} & \boldsymbol{x_p} \end{bmatrix} \tag{II-3}$$

The error states represent the onboard uncertainty of the navigation state. There is an almost one-to-one mapping between filter states and filter error states except for the attitude state. While represented as a four-component quaternion in the filter state, the attitude error is represented by a three-component modified Rodrigues parameter in the filter error state. As a result, the current implementation consists of 40 filter error states, as shown in Eq. (II-4) – Eq. (II-6).

$$\boldsymbol{\delta x_d} = \begin{bmatrix} \boldsymbol{\delta r_i} & \boldsymbol{\delta v_i} & \boldsymbol{\delta \theta_i^{imu}} \end{bmatrix} \tag{II-4}$$

$$\boldsymbol{\delta x_p} = \begin{bmatrix} \boldsymbol{\delta b_{\Delta V}} & \boldsymbol{\delta b_{\Delta\theta}} & \boldsymbol{\delta b_{\rho_{ndl}}} & \boldsymbol{\delta b_{v_{ndl}}} & \boldsymbol{\delta \phi_{\frac{cam}{imu}}} & \delta b_{f_x} & \delta b_{f_y} & \delta b_{p_x} & \delta b_{p_y} & \boldsymbol{0_{12\times 1}} \end{bmatrix} \tag{II-5}$$

$$\boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{\delta x_d} & \boldsymbol{\delta x_p} \end{bmatrix} \tag{II-6}$$

The navigation filter is broken into two flight software (FSW) applications scheduled at different rates: a high-rate propagation application and a low-rate state and error state update application. The propagation application uses IMU measurements coupled with onboard dynamics to propagate the vehicle state. The state update application updates the state correction vector and error states using measurements from sensors, correcting these states based on the residual between the sensor measurement and an onboard prediction of the measurement.

A. Propagation Application

The propagation application is scheduled at 50Hz and its primary purpose is to propagate the vehicle state by an elapsed time based on the time difference between two successive IMU measurements as received from the IMU interface application. In addition, the propagation application also performs the following:

- Vehicle state initialization based on table parameters loaded prior to flight (I-load parameters)
- Estimated state correction based on error state correction vector from the update application
- IMU error state compensation
- IMU measurement buffering required by the update application

- Navigation state outputs customized for downstream application use

The state estimate is propagated using an onboard linear dynamics approximation while using model replacement techniques to substitute IMU measurements for onboard models of non-gravitational accelerations and rotational rates of the vehicle. While IMU measurements are used to replace the non-gravitational and rotational disturbances in the onboard dynamics model, the gravitational forces must still be approximated using an onboard model. The initial state of the suborbital launch vehicle is well known and is stored as an I-load parameter. The I-load parameters are relative to the WGS-84 ellipsoid for the position data and relative to the Earth Centered Earth Fixed (ECEF) rotating reference frame for the attitude data. The state is constantly reinitialized at a 0.2Hz rate within the propagation application based on the I-load parameters opposed to performing a ground state update through the EKF.

The estimated state correction is calculated at a 5Hz rate by the update application but applied within one of the 50Hz propagation calls. Eq. (II-7) is a simplification of the state update logic as some attitude state updates are multiplicative and not additive. This approach is necessary given that the propagation and update logic are executed at different rates. The incoming state correction vector is valid at time t_{k-1}, while the current state vector is valid at time t_k. As a result, the state transition matrix is used to propagate the state correction vector to the current time, at which point it is used to update the current estimated state in the propagation application. As for the error state, it cannot be reset to zero as traditionally done but instead must be updated as shown in Eq (II-8) [13].

$$\hat{\boldsymbol{x}}_{\boldsymbol{k+}} = \hat{\boldsymbol{x}}_{\boldsymbol{k-}} + \hat{\boldsymbol{x}}\left(\boldsymbol{\delta x}_{(\boldsymbol{k-1})+}\right) \tag{II-7}$$

$$\boldsymbol{\delta x}_{\boldsymbol{k-}} \leftarrow \boldsymbol{\delta x}_{\boldsymbol{k-}} - \boldsymbol{\delta x}_{(\boldsymbol{k-1})+} \tag{II-8}$$

B. Update Application

The update application executes at 5Hz with the primary role of providing a state correction vector based on the processing of sensor measurements. The state correction vector is computed using a typical EKF approach that is reliant on an error state covariance. The update application is responsible for the following activities.

- Initialization of error state covariance
- Fault detection and residual editing checks for accepting/rejecting NDL and TRN sensor measurements
- Error state covariance propagation
- State correction vector reset
- State propagation to time of measurement
- State correction calculation, error state covariance update based on NDL and TRN sensor measurements
- Measurement underweighting utilizing Lear's method [14]
- Output construction for use by propagation application

The error state implementation used throughout the update application is consistent with the UDU implementation for the error state covariance [15]. Using this approach, the traditional covariance can be represented as an upper diagonal matrix combined with a diagonal matrix using the Cholesky decomposition. As a result, the covariance $\boldsymbol{P_k}$ is represented in terms of $\boldsymbol{U_k}$ and $\boldsymbol{D_k}$ as shown in Eq. (II-9).

$$\boldsymbol{P_k} = \boldsymbol{U_k D_k U_k^T} \tag{II-9}$$

The UDU implementation of the covariance is advantageous in that it is a more stable form of the covariance compared to other alternatives and when implemented correctly, is more computationally efficient as the user can take advantage of the diagonal property and upper triangular formulation of the matrices [15]. The error state covariance propagation is handled in the update application opposed to the propagation application to optimize execution speed since the covariance is only used within the update application. The error state covariance propagation can be broken into the following four different processes: 1) IMU buffer processing, 2) state matrix definition, 3) error state covariance propagation and 4) state correction vector propagate and reset.

Covariance propagation is handled via a combined modified Gram Schmidt algorithm and an Agee-Turner Rank One update algorithm derived from various factorization methods [16] [17]. The state transition matrix is used to propagate the error state vector with the reset handled by Eqn (II-8). Covariance propagation and state resetting are precursors to the primary functionality of the update application: state correction based on processed measurements. Both the error state covariance as well as the error state are updated with each measurement processed. Once the measurement

equation and subsequent sensitivity matrix have been derived, the resultant equations are plugged into the measurement update logic, which can be broken into the following steps: 1) compute the measurement sensitivity matrix using the current estimated state, 2) perform measurement underweighting if necessary, 3) compute the Kalman gain based on the measurement sensitivity, error state covariance and measurement noise matrices, 4) compute the predicted measurement based on the current estimated state, 5) compute the measurement residual, 6) perform residual editing, and 7) compute the updated error state covariance and state correction vector using the Kalman gain.

For the results presented in this paper, the filter is configured to process measurements from the NDL and an onboard TRN algorithm. The NDL sensor provides range and velocities along three beams similar to the package flown on the COBALT test flights [4]. The TRN algorithm processes real-time imagery from an onboard camera and provides pixel location in the image corresponding to features that were matched with features in an onboard database [5]. These measurements are then fused together to provide an update to the state [18]. For each measurement, the state estimate, error state and covariance are propagated to the time of the measurements prior to the above steps being completed. Once the steps are completed, these quantities are then propagated back to the time corresponding to the latest IMU time in the buffer received by the update application.

III. Dual Quaternion Guidance Algorithm

The guidance algorithm used for this application is based on previous work completed on a DQG algorithm at the University of Washington [19] [20]. In order to support the overall goal of Precision Landing and Hazard Avoidance (PL&HA), the implemented guidance algorithm must be able to reliably produce a trajectory solution that does not violate imposed constraints. These constraints may vary from safe landing trajectory considerations, to sensor considerations and constraints imposed by the available hardware. The DQG algorithm aims to formulate the vehicle dynamics along with these constraints into a convex form, which allows a Second Order Cone Program (SOCP) to optimize on the problem.

It is common for some vehicle constraints to be coupled in both attitude and translation, which results in constraint equations that are non-convex. To derive a convex form of these constraints, the attitude and translational position of the vehicle can be expressed in a dual quaternion form. The velocity and attitude rate of the vehicle can also be expressed this way. This allows for both translational and rotational state information to be expressed by a single term, which transforms some of these non-convex constraint equations into convex equations.

Eq. (III-1) describes the typical formulation of a dual quaternion, where $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ are ordinary quaternions and ε is the dual unit, which satisfies $\varepsilon^2 = 0$. This creates the eight-dimensional term $\tilde{\boldsymbol{q}}$. The dual quaternion can be used to express both translational and rotational information. This is derived by defining a position quaternion $\boldsymbol{r_I} = [x\ y\ z\ 0]^T$, which is a pure quaternion representing the position vector of the origin of the SPLICE guidance body frame with respect to the landing site origin in the East, North, Up (ENU) frame. The final result of the position quaternion is shown in Eq. (III-2), where $\boldsymbol{q}$ is the ordinary attitude unit quaternion and $\otimes$ denotes the quaternion product. The velocity components can also be represented by a dual quaternion by allowing the angular velocity and velocity components to be represented by pure quaternions. The dual velocity quaternion is shown in Eq. (III-3), where $\boldsymbol{\omega_B}$ is the pure quaternion defining the angular velocity vector in the body frame, and $\boldsymbol{v_B}$ is the pure quaternion defining the velocity in the body frame.

$$\tilde{\boldsymbol{q}} = \boldsymbol{q_1} + \boldsymbol{\varepsilon q_2} \tag{III-1}$$

$$\tilde{\boldsymbol{q}} = \boldsymbol{q} + \frac{\boldsymbol{\varepsilon}}{\boldsymbol{2}} \boldsymbol{r_I} \otimes \boldsymbol{q} \tag{III-2}$$

$$\tilde{\boldsymbol{\omega}} = \boldsymbol{\omega_B} + \boldsymbol{\varepsilon v_B} \tag{III-3}$$

The kinematics are computed by taking the derivative of Eq. (III-2), for which the result is shown in Eq. (III-4). The rigid body dynamics are formulated in Cartesian form for a 6-DoF rocket problem utilizing a single gimbaled main engine in Cartesian form. The equations are shown below as Eq. (III-5) – Eq. (III-9), where m is the vehicle mass, $\boldsymbol{u_B}$ is the engine thrust vector in the body frame, $\boldsymbol{\tau_B}$ is the external torque commands in the body frame, $\boldsymbol{J_{veh}}$ is the moment of inertia matrix, and $\boldsymbol{F}$, $\boldsymbol{T}$ are the total force and torque imparted on the vehicle respectively.

$$\dot{\tilde{q}} = \frac{1}{2}\tilde{q} \otimes \tilde{\omega} \tag{III-4}$$

$$\frac{dm}{dt} = -\alpha \|\boldsymbol{u_B}\|_2 \tag{III-5}$$

$$\frac{d}{dt}(m\boldsymbol{v_B}) = m\dot{\boldsymbol{v}}_B + \boldsymbol{\omega_B}^{\times}(m\boldsymbol{v_B}) = \boldsymbol{F} \tag{III-6}$$

$$\frac{d}{dt}(\boldsymbol{J\omega_B}) = \boldsymbol{J_{veh}}\dot{\boldsymbol{\omega}}_B + \boldsymbol{\omega_B}^{\times}(\boldsymbol{J_{veh}\omega_B}) = \boldsymbol{T} \tag{III-7}$$

$$\boldsymbol{T} = \boldsymbol{r_u}^{\times}\boldsymbol{u_B} + \boldsymbol{\tau_B} \tag{III-8}$$

$$\boldsymbol{F} = \boldsymbol{u_B} + m\boldsymbol{g_B} \tag{III-9}$$

Eq. (III-5) – Eq. (III-9) can then be formulated using dual quaternions instead of Cartesian vectors. This formulation is shown in Eq. (III-10) where $\boldsymbol{J}$ is shown in Eq. (III-11), $\widetilde{\boldsymbol{F}}$ is shown in Eq. (III-12), $\oslash$ is the dual quaternion cross product, and $\widetilde{\boldsymbol{g}}_{\boldsymbol{B}}$ is shown in Eq. (III-13). Eq. (III-14) describes $\boldsymbol{g_B}$ as the acceleration quaternion due to gravity in the body frame, which is derived from the pure quaternion $\boldsymbol{g_I}$ that represents the inertial gravitational acceleration vector [20].

$$\boldsymbol{J}\dot{\tilde{\boldsymbol{\omega}}} + \tilde{\boldsymbol{\omega}} \oslash \boldsymbol{J}\tilde{\boldsymbol{\omega}} = \widetilde{\boldsymbol{F}} + \widetilde{\boldsymbol{g}}_{\boldsymbol{B}} \tag{III-10}$$

$$\boldsymbol{J} = \begin{bmatrix} 0 & \boldsymbol{0_{1\times3}} & m\boldsymbol{I_{3\times3}} & \boldsymbol{0_{3\times1}} \\ \boldsymbol{0_{3\times1}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{1\times3}} & 1 \\ \boldsymbol{J_{veh}} & \boldsymbol{0_{3\times1}} & \boldsymbol{0_{3\times3}} & \boldsymbol{0_{3\times1}} \\ \boldsymbol{0_{1\times3}} & 1 & \boldsymbol{0_{1\times3}} & 0 \end{bmatrix}_{\boldsymbol{8\times8}} \tag{III-11}$$

$$\widetilde{\boldsymbol{F}} = \begin{bmatrix} \boldsymbol{F} \\ 0 \\ \boldsymbol{T} \\ 0 \end{bmatrix}_{\boldsymbol{8\times1}} \tag{III-12}$$

$$\widetilde{\boldsymbol{g}}_{\boldsymbol{B}} = \begin{bmatrix} m\boldsymbol{g_B} \\ \boldsymbol{0}_{4\times1} \end{bmatrix}_{\boldsymbol{8\times1}} \tag{III-13}$$

$$\boldsymbol{g_B} = \boldsymbol{q}^{*} \otimes \boldsymbol{g_I} \otimes \boldsymbol{q} \tag{III-14}$$

A. Dual Quaternion Guidance Architecture

The DQG application executes at 0.2 Hz and receives the vehicle state data from the navigation propagation application. Given this state update, the DQG application iteratively constructs the guidance problem, including vehicle and trajectory constraints, into a convex form to be solved by a Custom SOCP (CSOCP). CSOCP is a convex optimization solver algorithm that was generated offline by the Behçet SOCP (BSOCP) which was provided by the University of Washington [21]. A high-level operational flow of the application is highlighted in Figure 1.

B. Enforced Constraints

Multiple vehicle and trajectory constraints are implemented within the DQG application. These constraints were programmed with the provided Custom Parser (CPRS) from the University of Washington [19], which allows for intuitive programming of these constraints without the need to parse and program them by hand. These constraints were all enforced by the algorithm in flight and during playback analysis.

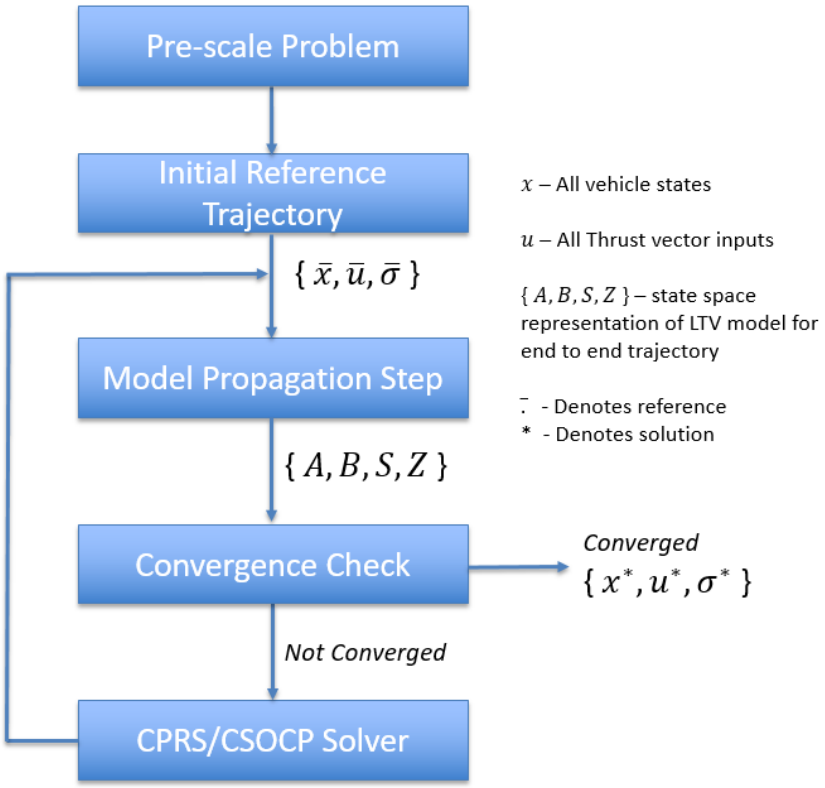


Figure 1: Dual Quaternion Guidance Application Architecture

1. Maximum Engine Thrust Magnitude

This constraint defines the maximum thrust magnitude that the vehicle's main engine can provide and is constructed into a convex form shown in Eq. (III-15) [19].

$$\|\boldsymbol{u_k}\|_2 \leq u_{max} \tag{III-15}$$

2. Minimum Engine Thrust Magnitude

This constraint defines the minimum thrust magnitude that the vehicle's main engine can provide once it has been ignited. This constraint is naturally non-convex, but a first order Taylor series approximation is applied to convexify the constraint. This convex constraint is shown in Eq. (III-16) [19].

$$u_{min} - \frac{\bar{\boldsymbol{u}}^{\boldsymbol{T}}{}_{\boldsymbol{k}}}{\|\bar{\boldsymbol{u}}_{\boldsymbol{k}}\|_2}\boldsymbol{u_k} \leq 0 \tag{III-16}$$

3. Maximum Torque Capability

This constraint defines the maximum external torque capability that the vehicle's attitude control system can provide. The convex constraints can be shown in Eq. (III-17). This constraint is applied on a per-axis basis with each body frame axis constrained by the same torque magnitude limitation.

$$\begin{aligned} -\tau_{max} \leq \tau_{x_k} \leq \tau_{max} \\ -\tau_{max} \leq \tau_{y_k} \leq \tau_{max} \\ -\tau_{max} \leq \tau_{z_k} \leq \tau_{max} \end{aligned} \tag{III-17}$$

4. Maximum Engine Thrust Rate

This constraint defines the maximum thrust magnitude rate that the engine can provide. This convex constraint is shown in Eq. (III-18) and Eq. (III-19), where u_{z_k} represents the thrust in the z-axis direction for the kth node where a node is a discretized point along the DQG solution, N represents the total number of nodes, σ is the burn time, and $\dot{u}_{max}$ represents the maximum engine thrust rate.

$$(u_{z_k} - u_{z_{k-1}})(N-1) \leq \dot{u}_{max}\sigma \tag{III-18}$$

$$(u_{z_k} - u_{z_{k-1}})(N-1) \geq -\dot{u}_{max}\sigma \tag{III-19}$$

5. Maximum Engine Gimbal Angle Deflection

This constraint defines the maximum deflection angle that the main engine gimbal thrust vector control (TVC) system can accomplish. This convex constraint is shown in Eq. (III-20), where $\boldsymbol{z_B}$ represents the unit vector pointing in the vehicle body z-axis for which the TVC is deflecting, $\boldsymbol{u_{xy_k}}$ represents the two dimensional thrust vector in the body frame x and y axis of the kth node, and δ_{max} is the maximum gimbal angle deflection.

$$\left\|\boldsymbol{u_{xy_k}}\right\|_2 \leq \tan(\delta_{max})\, \boldsymbol{z_B} \cdot \boldsymbol{u_k} \tag{III-20}$$

6. Maximum Engine Gimbal Angular Rate Deflection

This constraint defines the maximum deflection rate that the main engine gimbal TVC system is capable of. When the time of flight variable is a solution variable, the constraint is non-convex. Thus, a first order Taylor series approximation about the reference solution is used to convexify the constraint. This convexified constraint is shown in Eq. (III-21) where $\dot{\delta}_{max}$ is the maximum gimbal angular rate magnitude.

$$\left\|(\boldsymbol{u_{xy_k}} - \boldsymbol{u_{xy_{k-1}}})\right\|_2 (N-1) \leq \dot{\delta}_{max}\left[\bar{\sigma}u_{z_k} + \bar{u}_{z_k}(\sigma - \bar{\sigma})\right] \tag{III-21}$$

7. Maximum Vehicle Angular Rate

This constraint defines the maximum angular rate that the vehicle can experience during flight. This convex constraint is constructed in Eq. (III-22) [12], where ω_{max} represents the maximum angular rate magnitude. This constraint is applied on a per-axis basis with each body frame axis constrained by the same angular rate limitation.

$$\begin{aligned} -\omega_{max} \leq \omega_{x_k} \leq \omega_{max} \\ -\omega_{max} \leq \omega_{y_k} \leq \omega_{max} \\ -\omega_{max} \leq \omega_{z_k} \leq \omega_{max} \end{aligned} \tag{III-22}$$

8. Maximum Vehicle Tilt Angle

This constraint defines the maximum vertical tilt angle that the vehicle can experience during flight. This convex constraint is constructed in Eq. (III-23) and Eq. (III-24) where θ_{max} is the maximum tilt angle and $\mathbf{z_l}$ is the unit vector representing the vertical axis of the target landing site adhering the quaternion products outlined by Reynolds [12].

$$\tilde{\boldsymbol{q}}^{\boldsymbol{T}} M_T \tilde{\boldsymbol{q}} + \cos(\theta_{max}) - 1 \leq 0 \tag{III-23}$$

$$M_T = \begin{bmatrix} [\boldsymbol{z_I}]_{\otimes}[\boldsymbol{z_B}]^*_{\otimes} & 0_{4\times4} \\ 0_{4\times4} & 0_{4\times4} \end{bmatrix} \tag{III-24}$$

9. Maximum Glide Slope

This constraint defines the maximum angle of the vehicle approach cone to the landing site. This angle is measured between the vertical axis of the landing site and the vehicle's position vector. This constraint is non-convex, so a first order Taylor series approximation is used to convexify the constraint. The constructed convex equation is shown in Eq. (III-25) through Eq. (III-28) [19], when $\boldsymbol{q_k}$ is the dual quaternion for position and attitude, γ_{max} is maximum glide slope and $\boldsymbol{z_I}$ is the Up-axis of the ENU reference frame.

$$-\tilde{\boldsymbol{q}}_k{}^T M_G \tilde{\boldsymbol{q}}_k + \|2E_d \tilde{\boldsymbol{q}}_k\| \cos(\gamma_{\max}) + \left.\frac{\partial f_G}{\partial \tilde{\boldsymbol{q}}}\right|_{\bar{\tilde{\boldsymbol{q}}}} \left(\tilde{\boldsymbol{q}}_k - \bar{\tilde{\boldsymbol{q}}}_k\right) \leq 0 \tag{III-25}$$

$$\left.\frac{\partial f_G}{\partial \tilde{\boldsymbol{q}}}\right|_{\bar{\tilde{\boldsymbol{q}}}} = -2M_G \bar{\tilde{\boldsymbol{q}}}_k + 2\frac{E_d \bar{\tilde{\boldsymbol{q}}}_k}{\left\|E_d \bar{\tilde{\boldsymbol{q}}}_k\right\|} \cos(\gamma_{max}) \tag{III-26}$$

$$M_G = \begin{bmatrix} 0_{4\times4} & [\boldsymbol{z_I}]_{\otimes}^T \\ [\boldsymbol{z_I}]_{\otimes} & 0_{4\times4} \end{bmatrix} \tag{III-27}$$

$$E_d = \begin{bmatrix} 0_{4\times4} & 0_{4\times4} \\ 0_{4\times4} & I_{4\times4} \end{bmatrix} \tag{III-28}$$

10. Initial Condition Constraints

This set of constraints define the initial vehicle state conditions at the start of each DQG FSW application cycle. The initial vehicle mass is set to the initial mass of the vehicle in Eq. (III-29). The initial attitude of the vehicle is fixed to the initial attitude provided by navigation, shown in Eq. (III-30). The initial position quaternion is free within a bounded magnitude in the East and North plane, shown in Eq. (III-31) and (III-32). The miss distance is represented by $\boldsymbol{\delta_r}$, which is the position relative to the initial vehicle position expressed as a pure quaternion. This is done to allow for the creation of a final miss distance tolerance. Incorporating a free final position bounded in the East and North plane as a terminal condition is not convex. However, a free initial position bound can be implemented with an offset applied to the position after a solution has been found. When applying this tolerance as an initial condition, it becomes convex since the initial attitude is fixed. The velocity and angular rate constraints are set to basic equality constraints, shown in Eq. (III-33) and (III-34).

$$m_0 = m_{IC} \tag{III-29}$$

$$\boldsymbol{q_0} = \boldsymbol{q_{IC}} \tag{III-30}$$

$$\boldsymbol{q_{r_0}} + \frac{1}{2}[\boldsymbol{q_{IC}}]_{\otimes}^{*} \boldsymbol{\delta_r} = \frac{1}{2}\boldsymbol{r_{I_{IC}}} \otimes \boldsymbol{q_{IC}} \tag{III-31}$$

$$\|\boldsymbol{\delta_{EN}}\|_2 \leq \Delta \boldsymbol{r}, \qquad \boldsymbol{\delta_U} = 0 \tag{III-32}$$

$$\boldsymbol{v_{B_0}} = \boldsymbol{v_{IC}} \tag{III-33}$$

$$\boldsymbol{\omega_{B_0}} = \boldsymbol{\omega_{IC}} \tag{III-34}$$

11. Final Condition Constraints

This set of constraints defines the desired vehicle target conditions. The final vehicle mass is constrained to be larger than the vehicle dry mass, m_{dry}, shown in Eq. (III-35). The final attitude of the vehicle is governed by a vehicle tilt angle constraint set by θ_F and shown in Eq. (III-36). The final position quaternion is set to an equality constraint in Eq. (III-37), where the right hand side represents a predetermined final position target as a function of the final vehicle attitude. The final velocity magnitude is constrained to be less than or equal to the set final velocity limit, shown in Eq. (III-38). Finally, Eq. (III-39) defines the final vehicle angular rate as an equality constraint.

$$m_N \geq m_{dry} \tag{III-35}$$

$$\tilde{\boldsymbol{q}}^T{}_N \boldsymbol{M_T} \tilde{\boldsymbol{q}}_N + \cos(\theta_F) - 1 \leq 0 \tag{III-36}$$

$$\boldsymbol{q_{r_N}} = \frac{1}{2}\boldsymbol{r_{I_F}} \otimes \boldsymbol{q_N} \tag{III-37}$$

$$\|\boldsymbol{v_N}\|_2 \leq \boldsymbol{v_F} \tag{III-38}$$

$$\boldsymbol{\omega}_{\mathrm{N}} = \boldsymbol{\omega_F} \tag{III-39}$$

IV. Suborbital Flight Hardware

The SPLICE project provided hardware for this test flight included the following: a descent and landing computer (DLC) engineering development unit (EDU), an commercial-of-the-shelf (COTS) IMU, an NDL developed by NASA Langley and a low-cost COTS monochrome camera used for TRN. The navigation and guidance algorithms are executed on the DLC with the sensor measurements passed through a field programmable gate array (FPGA) interface to provide accurate time stamping of the measurements. The DLC EDU surrogate includes the Xilinx UltraScale+ Multi-Processor System on a Chip (MPSoC) [22] [23]. A rendered image of the DLC EDU is shown in Figure 2.

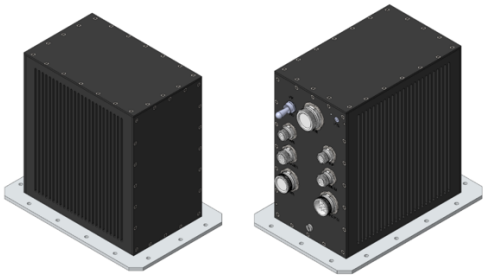
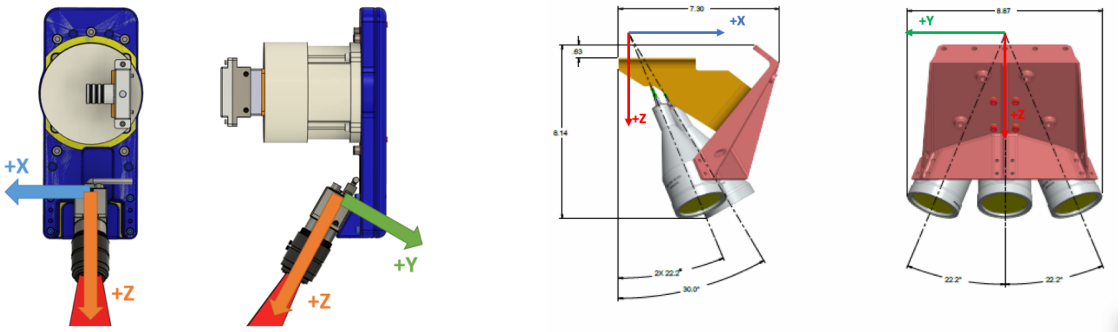


Figure 2: SPLICE DLC Surrogate Flight Computer

The sensor hardware was mounted separately from the DLC EDU surrogate for the test flight as part of the line of sight requirements of the TRN camera and NDL optical heads. To consolidate the SPLICE related IMU and TRN camera in a single location, a mounting bracket (shown in Figure 3a) was designed and fabricated to ensure viewing constraints were met. The NDL electronics chassis was mounted in a separate location on the propulsion module while the optical heads were mounted directly below the SPLICE camera and IMU bracket shown in Figure 3b to satisfy line of sight constraints.



a) SPLICE IMU, TRN Camera Mounting Bracket b) NDL Optical Head Mounting Bracket

Figure 3: SPLICE Sensor Mounting Brackets

V. Flight Operations Overview

The SPLICE hardware was externally mounted to the propulsion module of a Blue Origin New Shepard launch vehicle and launched from the Blue Origin Launch Site One near Van Horn, Texas [24]. Utilizing the New Shepard flight profile (shown in Figure 4 and Figure 5) allows for analysis of navigation performance over an ascent and of more interest, a descent trajectory while the dynamic landing of the booster on a landing pad using the rocket engine provides an ideal use case for the DQG algorithm emulating a powered descent trajectory. Vertical motion dominates the trajectory therefore only the vertical channels are shown. As can be seen in Figure 4, the trajectory surpasses 100km in altitude while horizontally, the maximum motion is less than 5km total.

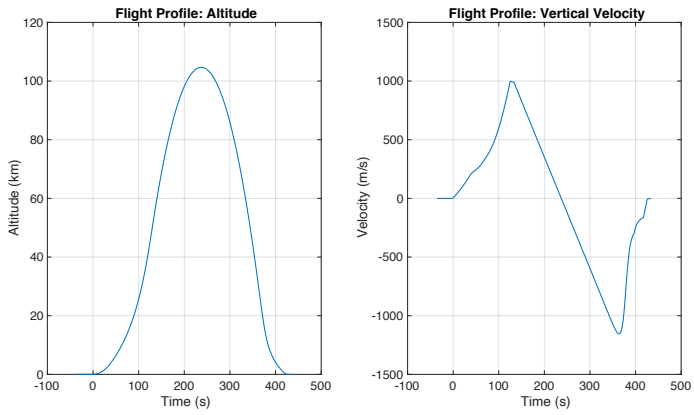


Figure 4: Vertical Flight Profile

The roll attitude of the booster is shown in Figure 5. The roll attitude remains fairly constant during ascent before beginning to roll shortly after main engine cutoff (MECO). This roll persists until it is arrested prior to the powered descent portion of flight. Due to this, the TRN algorithm exercised the majority of its onboard database and the flight test assessed the robustness of the algorithm to various lighting conditions, terrain relief, feature motion across successive images, and other operational constraints.

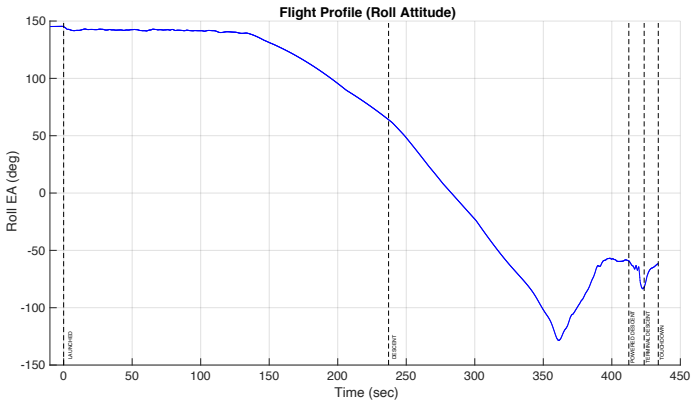


Figure 5: Roll Attitude Flight Profile

Day of launch operations had initial checkouts of the SPLICE hardware components and guidance, navigation and TRN (GNT) algorithms occurring approximately five hours prior to launch. Upon initial bootup, all sensor hardware and GNT algorithms passed their automated checks. All SPLICE components were then powered down and remained unpowered until two hours prior to flight. Around two hours prior to launch, the SPLICE hardware was rebooted in preparation for the flight test. While all SPLICE software components continued to operate nominally, NDL anomalies were identified. The anomalies were investigated prior to flight with multiple attempts to clear the anomalies executed without success. Ultimately, it was decided to proceed with the launch without an operational NDL. While NDL remained powered throughout flight, the sensor did not output valid navigation measurements and was therefore rejected by the navigation algorithm via its built-in fault protection logic. As a result, the only active sensors onboard for the flight test were the IMU and TRN camera.

The test flight lasted approximately 7 minutes from liftoff to touchdown. The navigation algorithm was operational prior to and throughout the entirety of the flight. The DQG algorithm was activated shortly before the start of the powered descent phase and solved for a solution once every 5 seconds. This flight was meant to serve as an operational test of the integrated GNT software in an open-loop configuration, therefore the vehicle did not respond to any of the generated guidance commands. Post-flight, all data and imagery was downloaded for post flight analysis.

VI. Results

This section will cover the post flight analysis performed based on data collected during the flight. As previously stated in Section V, the NDL provided invalid navigation measurements during flight. For the post flight analysis performed, the team was able to simulate NDL measurements using a realistic simulation model of the NDL sensor during flight playback analysis. The results shown in this section include this simulated NDL sensor with a maximum operational slant range of 4.5km.

Navigation errors computed in this section are a result of a comparison of the SPLICE navigation solution with the launch vehicle navigation state. The launch vehicle used a GPS-aided navigation filter to generate their vehicle state estimate. Since a best-estimate trajectory from the launch provider was unavailable when transcribing this paper, the launch vehicle navigation state was accepted to be truth for the purpose of this analysis. Post flight data smoothing was performed on the launch vehicle navigation state to remove additional noise in the solution for the trajectory truth generation from the host vehicle navigation state. It is important to recognize that the launch vehicle navigation solution used an IMU that was independent from the IMU used by the SPLICE navigation system therefore noise characteristics between the navigation states varied.

A. Navigation Performance

The SPLICE navigation algorithm performed as expected based on the flight playback analysis shown in this section. All figures within this section include colored regions that indicate various sensor measurements that were incorporated in the navigation solution for the provided windows with an '*r*' prefix indicating flight recorded measurements and '*s*' prefix indicating simulated measurements.

- Yellow: *r*IMU + *s*NDL
- Green: *r*IMU + *s*NDL + *r*TRN
- Blue: *r*IMU + *r*TRN

Figure 6 provides the inertial position error along with the onboard 3σ standard deviation computed by the SPLICE navigation algorithm. Figure 7 provides the inertial velocity error along with the onboard 3σ standard deviation computed by the SPLICE navigation algorithm. Even though TRN measurements are received throughout flight, the growth seen up through descent is expected as the accuracy of TRN measurements decrease as altitude increases due to the resolution of the onboard maps and imagery at higher altitude. Once apogee is achieved (and the FSW mode switches to DESCENT around L+240 seconds), the velocity standard deviation begins to slowly decrease. The position and velocity errors begin to more sharply decrease once the vehicle begins to encounter the atmosphere where aerobraking is performed during descent around L+300 seconds. As descent continues, errors continue to decrease until the simulated NDL is activated providing more insight into the position and velocity states.

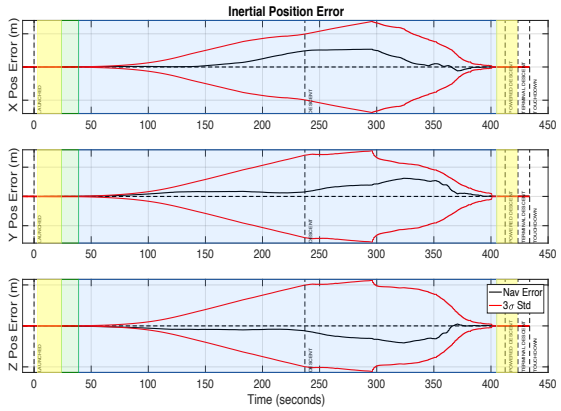


Figure 6: Inertial Position Navigation Error

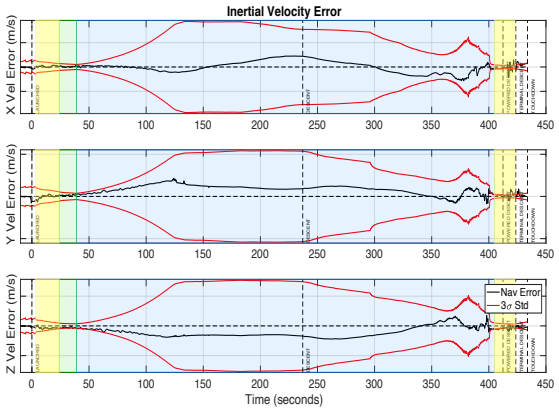


Figure 7: Inertial Velocity Navigation Error

The body attitude error is shown in Figure 8. Note the excursion of the error and general convergence back to zero near launch for the pitch and yaw channels. This is driven by uncertainty in the navigation state of the launch vehicle and not the SPLICE navigation algorithm. Secondly, note the apparent noisiness and error excursions of the signal

near landing. A highly dynamic flight phase coupled with differing IMU noise characteristics and data smoothing of the truth trajectory contributes to the calculated error. The attitude error remains bounded by the 3σ standard deviation outside of these two regions. Both the NDL and TRN measurements strongly affect the attitude estimate made evident by the tighter standard deviation bounds in Figure 8 compared to Figure 6 and Figure 7.

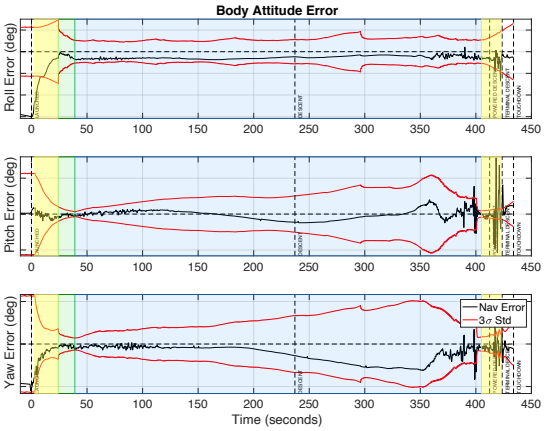


Figure 8: Body Attitude Navigation Error

Figure 9 provides insight into the TRN measurement update logic in the update application. The measurement from the TRN algorithm is in the form of a feature location in the 2-D pixel space of the image. The top two plots show the filter-computed measurement based on the navigation state compared to the actual measurement returned by the TRN algorithm. A 1440x1080 pixel camera was used for this flight, hence the varying y-axis limits and measurements when comparing the two plots. The top two plots represent one of four features that was processed at any given time by the navigation algorithm. These four features change throughout flight dependent on the correlated features returned by the TRN algorithm. Therefore, large jumps between successive measurements is indicative of a new feature being tracked while small, linear motion indicates the same feature being tracked across successive images for multiple frames. The bottom plots in Figure 9 show the residual as computed by the filter. The residuals start off a little larger before converging indicating state corrections have appropriately adjusted the navigation state based on the TRN measurements. Near the end of flight, the residuals trend larger due to a combination of a highly dynamic phase of flight coupled with foreign object debris on the lens due to environmental conditions present on the day of the flight. The constant values seen near landing on all the plots in Figure 9 are due to the TRN processing logic not executing and therefore the last valid measurement is reported in telemetry.

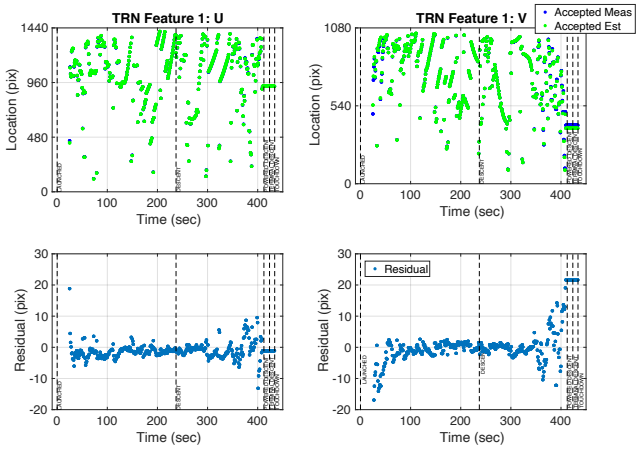


Figure 9: TRN Measurement Processing

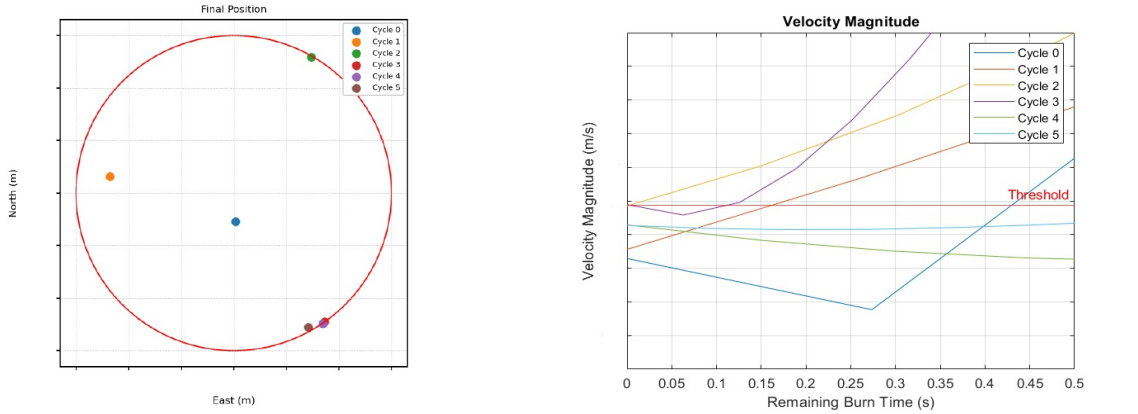
B. Dual Quaternion Guidance Performance

During the flight, DQG successfully operated and converged on guidance solutions within 3 seconds. However, it was only executed for three guidance cycles, so playback sensor data was then used to allow for additional testing post-flight. This allowed for minor parameter modifications to extend the operational range of DQG to occur earlier in flight.

For this application, DQG was designed to operate in two flight modes: POWERED DESCENT and TERMINAL DESCENT modes. The same vehicle constraints were enforced for both modes. However, during the POWERED DESCENT mode, the algorithm targeted a point above the landing pad. This target shifted to the landing pad itself once the vehicle transitioned into TERMINAL DESCENT. Thus, cycles 0 through 3 in the plots presented in this section are solutions found during POWERED DESCENT, while cycles 4 and 5 are solutions found during TERMINAL DESCENT.

Once the POWERED DESCENT mode began, the DQG FSW executed at a 0.2 Hz rate. The navigation application provided all of the vehicle state information as input to DQG, including a mass estimate of the vehicle. The DQG algorithm proceeded with computing a new solution every DQG FSW application cycle. This was completed in an open-loop fashion, thus there were no applications receiving input from the DQG application. Since all cycles were solved within 3 seconds, the DQG application successfully completed its computation within the budgeted 5 second window. To meet the timing and accuracy requirements, the DQG algorithm operated with 10 discrete trajectory nodes. Thus, the algorithm discretized the solution into 10 nodes and enforced the dynamics, control commands, and constraints at each node. A first order hold was utilized for commands, and the state information in between nodes was generated using Runge-Kutta propagation techniques.

Figure 10a depicts the final target position of the vehicle in the East North plane along with the final velocity magnitude. The final position was constrained to be within the red circle that represents the miss distance magnitude limit with many solutions terminating near this limit. It is important to note that although DQG is targeting the origin of this plane, it has freedom to move the final position anywhere within a threshold in order to minimize the cost function. The final velocity of the vehicle was constrained to be less than a specified limit to meet landing requirements. Figure 10b also shows that by the end of the DQG-generated trajectory, the vehicle's final velocity remains below this constraint.



a) Touchdown Position

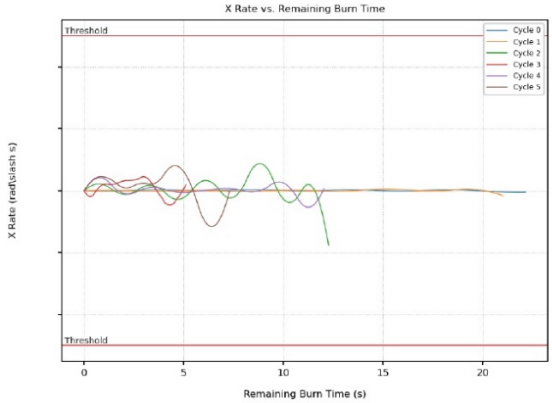
a) Velocity Magnitude vs. Remaining Burn Time

Figure 10: DQG Trajectory Solutions & Constrained Final Position

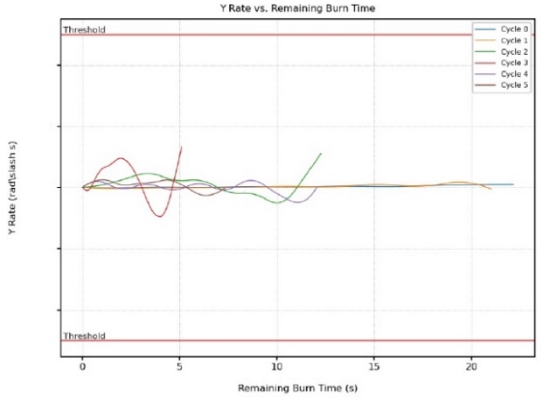
The resulting vehicle angular rates for the body yaw, pitch, and roll axis are shown in Figure 11. Each axis was constrained to be within the positive and negative angular rate maximum. The final boundary constraint was also set such that all angular rates were to be nulled by the time the vehicle reached the desired target. The plots show that these constraints were properly enforced during all DQG cycles.

Additional trajectory constraints such as maximum vehicle tilt angle and glide slope angle are shown in Figure 12 with respect to remaining burn time. The tilt angle constraint is enforced both as a total maximum limit as well as a

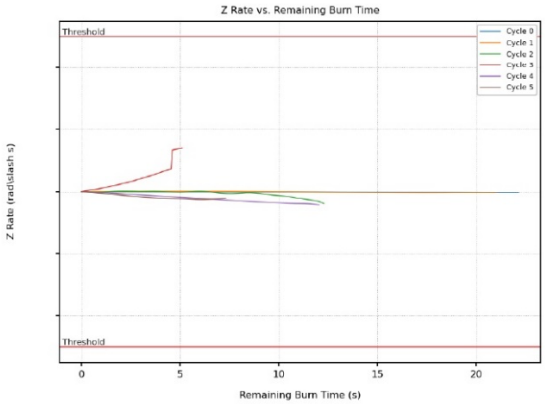
maximum limit at the final boundary condition. This is outlined with the red limit for the former, and the green limit for the latter. The results show that the upper maximum limit was not tested on any DQG cycle, however the maximum limit on the final tilt angle was tested on three DQG cycles. The maximum glide slope constraint was enforced as an upper limit throughout the entire trajectory, and the solution remained within the limit.



a) Yaw Rate vs. Remaining Burn Time

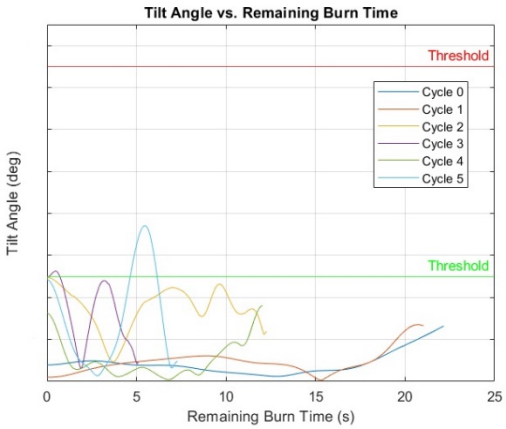


b) Pitch Rate vs. Remaining Burn Time

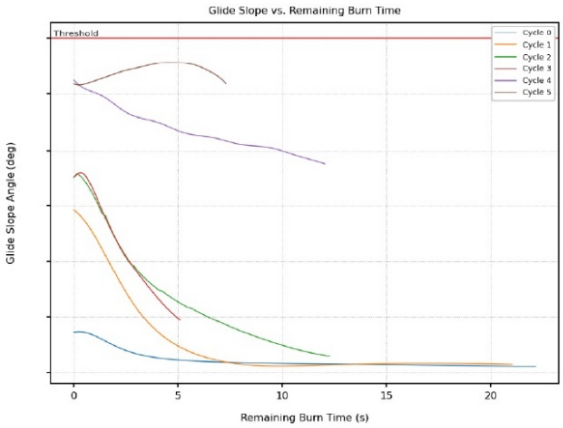


c) Roll Rate vs. Remaining Burn Time

Figure 11: DQG Constrained Angular Rates



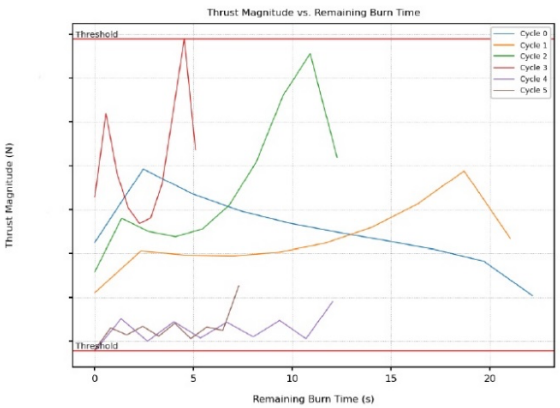
a) Tilt Angle vs. Remaining Burn Time



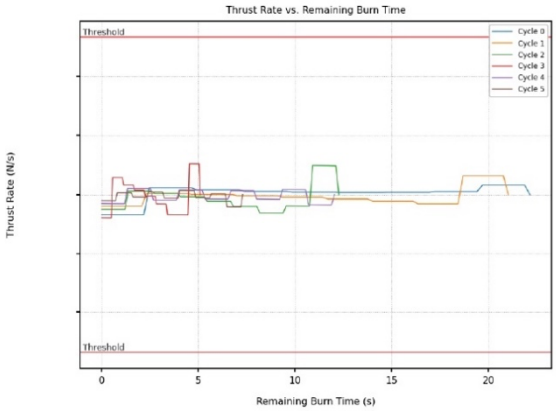
b) Glide Slope vs. Remaining Burn Time

Figure 12: DQG Constrained Tilt & Glide Slope Angles

The vehicle control constraints also performed as expected. Figure 13 shows the provided DQG thrust magnitude and thrust rates as a function of remaining burn time. These are the thrust commands generated by DQG to actuate the vehicle to the desired trajectory. The algorithm enforced a maximum and minimum bound for both thrust magnitude and thrust rate. The results show that the provided commands did maintain these set limits. Similarly, the main engine gimbal deflection and deflection rate were also constrained and are shown in Figure 14 followed by the generated torque commands are shown in Figure 15.

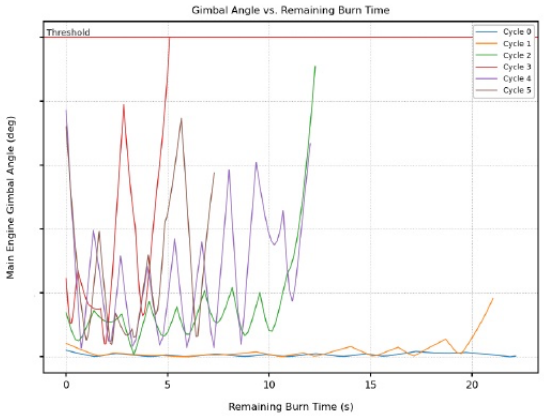


a) Thrust Magnitude vs. Remaining Burn Time

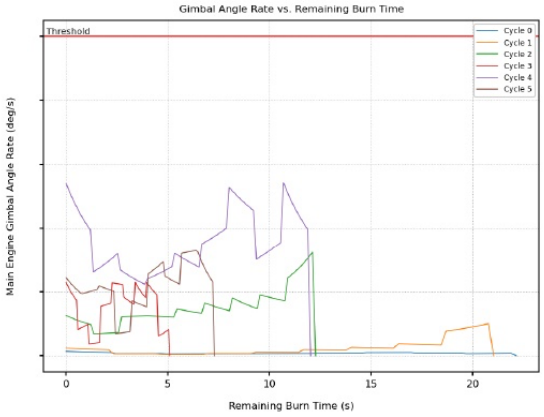


c) Thrust Rate vs. Remaining Burn Time

Figure 13: DQG Constrained Thrust Magnitude & Thrust Rate



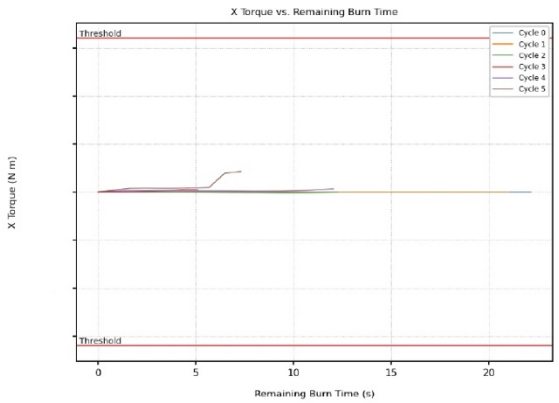
a) Gimbal Deflection vs. Remaining Burn Time



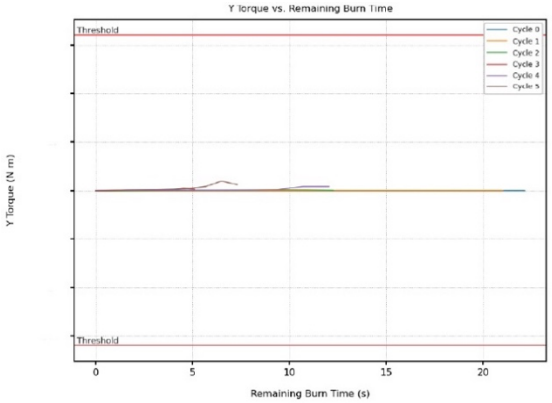
d) Gimbal Deflection Rate vs. Remaining Burn Time

Figure 14: DQG Constrained Gimbal Deflection & Gimbal Deflection Rate

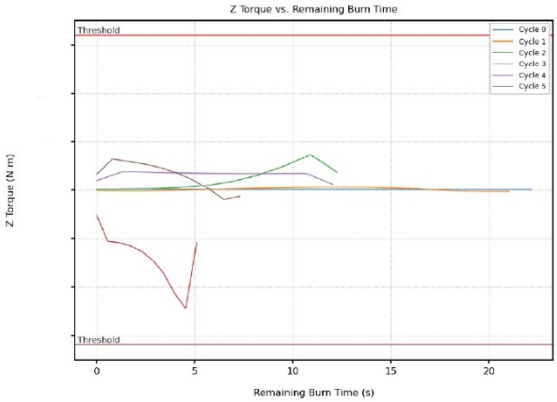
It is clear that all of the enforced control constraints were properly enforced by the DQG algorithm. The thrust minimum and maximum magnitude limits were both tested in these results, but the DQG algorithm did not allow the solution to break these limitations. This can also be observed in the gimbal angle deflection results. The yaw & pitch axes torque commands were tame compared to the roll axis torque commands. This was expected since the main engine has little to no dynamic effect on controlling the roll rates experienced by the vehicle. DQG heavily utilized the roll torque commanding capability to null the roll rates experienced on cycle 3.



a) Yaw Torque vs. Remaining Burn Time



b) Pitch Torque vs. Remaining Burn Time



c) Roll Torque vs. Remaining Burn Time

Figure 15: DQG Constrained Torque Commands

VII. Conclusion / Future Work

The navigation algorithm performed as expected during flight and in post-flight playback analysis. Noise visible in navigation error, specifically the attitude error near landing, is expected to be improved upon receipt of a best-estimate trajectory from the launch vehicle provider. Although computationally expensive, DQG successfully solved for multiple optimal solutions both during flight and in post-flight playback analysis within the execution time allotted. All generated solutions were within the designed constraint limitations.

Future work is required to incorporate ground state information as part of the navigation update application as well as expand the navigation algorithm to incorporate inputs from additional landing sensors. Computational loading needs to be further reduced to increase the rate at which the DQG application can execute on the DLC EDU, eventually targeting of 0.5Hz execution rate. In addition, an outer closed loop control style architecture needs to be developed to allow for intermediate updates to the DQG generated guidance commands to provide higher rate commands for closed loop flights. Future terrestrial test flights will focus on the fusion of additional landing sensors and added algorithmic capability to both navigation and DQG algorithms with a goal of closed-loop feedback of the guidance commands for use by the launch vehicle (closed loop flight test).

VIII. Acknowledgments

The work presented in this paper funded through the NASA Johnson Space Center as part of technology advancement and demonstration for the SPLICE project in conjunction with the NASA Tipping Point Program. Flight testing was made possible by Blue Origin, LLC based in Kent, WA. The authors would also like to acknowledge the support and expertise provided by the graduate students at University of Washington under Dr. Behçet Açikmeşe as well as Texas A&M professor Dr. Kyle DeMars and his students.

IX. References

[1] R. R. Sostaric, J. M. Carson, S. M. Pedrotty and et al., "The SPLICE Project: Safe and Precise Landing Technology Development and Testing," in *AIAA SciTech Forum*, Virtual Event, 2021.

[2] J. M. Carson, E. Robertson, N. Trawny and F. Amzajerdian, "Flight Testing ALHAT Precision Landing Technologies Integrated Onboard the Morpheus Rocket Vehicle," in *AIAA SPACE 2015 Conference and Exposition, AIAA SPACE Forum, (AIAA 2015-4417)* , Pasadena, California, 31 Aug-2 Sep 2015.

[3] J. B. Olansen, S. R. Munday and J. L. Devolites, "Project Morpheus: Lander Technology Development," in *AIAA SPACE 2014 Conference and Exposition, AIAA SPACE Forum, (AIAA 2014-4314*, San Diego, CA, August 4-7, 2014.

[4] J. M. Carson, C. I. Restrepo, C. R. Seubert, F. Amzajerdian, D. Pierrottet, S. Collins, T. O'Neal and R. Stelling, "Open-Loop Flight Testing of COBALT Navigation and Sensor Technologies for Precise Soft Landing," in *AIAA SPACE and Astronautics Forum and Exposition, AIAA SPACE Forum, (AIAA 2017-5287)*, Orlando, FL, 12 - 14 Sep 2017.

[5] M. P. Fritz, A. S. Olguin, K. W. Smith, R. Lovelace, R. Sostaric, S. Pedrotty, J. Estes, T. Tse and R. Garcia, "Operational Constraint Analysis of Terrain Relative Navigation for Landing Applications," in *AIAA ScitTech Forum*, Orlando, FL, 2020.

[6] L. D. Jaffe, "The Surveyor Lunar Landings," *Science,* vol. 164, no. 3881, pp. 775-789, 1969.

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

[8] C. N. D'Souza, "An Optimal Guidance Law for Planetary Landing," in *AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, 1997.

[9] B. A. Açıkmese and S. Ploen, "A Powered Descent Guidance Algorithm for Mars Pinpoint Landing," in *AIAA Guidance,Navigation, and Control Conference and Exhibit*, San Francisco, CA, 2005.

[10] S. Ploen, B. Açıkmese and A. Wolf, "A Comparison of Powered Descent Guidance Laws for Mars Pinpoint Landing," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Keystone, CO, 2006.

[11] S. R. Ploen and B. Açıkmese, "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control, and Dynamics,* vol. 30, no. 5, pp. 1353-1366, 2007.

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

[13] S. Steffes, "Development and Analysis of SHEFEX-2 Hybrid Navigation System Experiment," PhD Dissertation, Universitat Bremen, April 2013.

[14] R. Zanetti, K. J. DeMars and R. H. Bishop, "On Underweighting Nonlinear Measurements," *Journal of Guidance Control and Dynamics,* vol. 33, no. 5, pp. 1670-1674, 2010.

[15] C. D'Souza and R. Zanetti, "Information Formulation of the UDU Kalman Filter," *IEEE Transactions on the Aerospace and Electronic Systems,* 2019.

[16] C. Thornton, "Triangular Covariance Factorizations for Kalman Filtering," PhD thesis, University of California at Los Angeles, October 1976.

[17] G. J. Bierman, Factorization Methods for Discrete Sequential Estimation, New York: Dover Publications, 2006.

[18] K. C. Ward, C. Helmuth, G. S. Fritsch and K. J. DeMars, "Fusion of Multiple Terrain-Based Sensor for Descent-to-Landing Navigation," in *AIAA SciTech Forum*, Orlando, FL, 2020.

[19] T. Reynolds, Szmuk, M., D. Malyuta, M. Mesbahi, B. Açıkmeşe and J. M. Carson, "Dual Quaternion-Based Powered Descent Guidance with State-Triggered Constraints," *Journal of Guidance, Control, and Dynamics,* no. 43, pp. 1584-1599, 2020.

[20] T. Reynolds, D. Malyuta, M. Mesbahi, B. Acikmese and J. M. Carson, "A Real-Time Algorithm for Non-Convex Powered Descent Guidance," in *AIAA SciTech 2020 Forum*, Orlando, FL, 2020.

[21] D. Dueri, J. Zhang and B. Açikmese, "Automated Custom Code Generation for Embedded, Real-time Second Order Cone Programming," *IFAC Proceedings ,* no. 47, pp. 1605-1612, 2014.

[22] D. Rutishauser and T. Tse, "Hardware-in-the-Loop Testing for Suborbital Flights of the Safe and Precise Landing Integrated Capabilities Evolution (SPLICE) Project Technologies," in *AIAA SciTech Forum*, Orlando, FL, 2020.

[23] "ZYNQ UltraSCALE+," XILINX, [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html . [Accessed September 2021].

[24] "New Shepard Suborbital Launch Vehicle," Blue Origin, LLC, [Online]. Available: https://www.blueorigin.com/new-shepard/. [Accessed September 2021].