Implementation of a Six Degree of Freedom Precision Lunar Landing Algorithm Using Dual Quaternion Representation

Lloyd D. Strohl III[*] Andrew W. Berning Jr.[†], and Stefan Bieniawski[‡]
Blue Origin, Kent, WA 98032, USA

Javier A. Doll[§], Matthew P. Fritz[¶]
The Charles Stark Draper Laboratory, Inc, Cambridge, MA 02139, USA

Stephanie White[‖]
Odyssey Space Research, Houston, TX 77058

John M. Carson III[**] and Behçet Açikmeşe[††]
NASA

In this study, a powered descent guidance algorithm using a unit dual quaternion representation of the vehicle dynamics is implemented in a high-fidelity simulation and on representative flight hardware. This Dual-Quaternion Guidance (DQG) algorithm is applied to the precision lunar landing problem which levies complex constraints upon the trajectory, including state triggered attitude constraints to enable terrain-relative navigation and hazard detection as well as real-time requirements for landing site re-designation. The investigation explores DQGs usefulness as a mission design tool as well as a real-time guidance algorithm and defines real-time performance requirements for the hazard detection and avoidance (HDA) re-targeting phase of precision lunar landing. The experiment is presented in two parts. First, DQG is implemented within a high-fidelity Monte Carlo simulation to tune the algorithm's parameters for the simulated vehicle, to refine the mission design, and to develop guidance update timing requirements to perform the HDA maneuver. DQG generates trajectories online for the divert which are tracked by the vehicle's inner-loop controllers to the targeted landing site. Second, DQG is run on representative hardware to demonstrate real-time operation through a divert maneuver. These results allow for rapid, flexible, optimal mission design satisfying complex constraints, and for the definition of real-time performance requirements for the HDA operations inherent in precision lunar landing. The HDA divert maneuver is found to require guidance trajectory updates in less than three seconds. DQG is found to be too slow to meet this update timing on the descent and landing computer (DLC) in its current implementation. DQG running on alternative hardware can meet the update rate requirement. Algorithm implementation improvements are also recommended which are expected to speed up computation sufficiently to meet requirements on the DLC.

[*]GNC Engineer, LStrohl@blueorigin.com
[†]GNC Engineer, ABerning@blueorigin.com
[‡]Principal Technologist, Autonomy and GNC, SBieniawski@blueorigin.com
[§]Member Technical Staff, jdoll@draper.com
[¶]Member Technical Staff, mfritz@draper.com
[‖]Software Engineer,stephanie.white@odysseysr.com
[**]Technical Integration Manager for Precision Landing - STMD, AIAA Associate Fellow Lifetime, John.M.Carson@nasa.gov
[††]IPA from University of Washington, AIAA Associate Fellow, behcet@uw.edu

I. Nomenclature

$6-DoF$	=	Six degrees of freedom
$BSOCP$	=	Behçet SOCP
$CONOPs$	=	Concept of Operations
$CPRS$	=	Custom Parser
DLC	=	Descent and Landing Computer
DQG	=	Dual Quaternion Guidance
HDA	=	Hazard Detection and Avoidance
LTV	=	Linear Time-Varying
PDI	=	Powered Descent Initiation
PTR	=	Penalized Trust Region
SCP	=	Sequential Convex Programming
$SCvx$	=	Successive Convexification
$SOCP$	=	Second-Order Cone Program
TRN	=	Terrain Relative Navigation

II. Introduction

The precision lunar landing problem involves several relevant and distinct phases for powered descent:

1) Powered Descent Initiation
2) Sensor Visibility
3) Hazard Detection and Avoidance
4) Vertical Descent
5) Landing

The sensor visibility and Hazard Detection and Avoidance (HDA) phases levy non-convex constraints upon the landing trajectory, which must be flexible to re-targeting on-board. Terrain relative navigation (TRN) and hazard detection sensors only function under certain conditions, such as vehicle altitude and ground-track velocity as well as site-relative vehicle attitude for sensing angles. These coupled navigation and guidance constraints apply to a six-degree-of-freedom (6-DoF) landing trajectory which must either be carefully satisfied by a-priori mission design and simple on-board guidance, or through solution of the full constrained 6-DoF powered descent guidance problem in real-time. HDA operations constrain solution time of the landing trajectory to the order of seconds. Over the past few years, solution of the constrained 6-DoF powered descent guidance problem has converged upon sequential convex programming (SCP) methods including penalized trust region (PTR) and successive convexification (SCvx).

The authors of [1] applied a PTR-based powered descent guidance algorithm to the 6-DoF constrained landing problem and showed real-time capability for in-plane powered descent guidance with independent thrust and torque inputs. This simplified problem allowed direct comparison with a known optimal solution. The problem formulation consisted of 7 state variables, 2 control variables, free final time, state constraints, and thrust and torque box constraints. The constraints on state variables were chosen to be inactive during the trial. The lunar DQG formulation in the current study is significantly larger and contains many more non-convexities due particularly to the presence of state-triggered constraints, as detailed in Section III.A.1. The authors demonstrated real-time solution with <1 % suboptimality on the order of 250ms with a 3.2 GHz Intel Core i5 processor for a discretization of 20 nodes.

In [2], the authors performed open-loop Monte Carlo testing with dispersed initial mass, position, and velocity using a DQG formulation enforcing a state-triggered line-of-sight (LOS) constraint. The problem was again smaller than the problem presented in the current study. Runtime analysis was performed on a 3.2 GHz Intel Core i5 processor, and showed a mean solution time of 0.75s. This was a more modern processor than the descent and landing computer (DLC) evaluated in this study, but showed promising performance on a problem with significant non-convexities which approached the complexity that is presented here.

In [3] the authors present a formulation of DQG flown open-loop on a terrestrial suborbital rocket flight. Their formulation was adapted to the lunar landing problem presented here, and enforced many of the same constraints but did not enforce state-triggered LOS constraints. The terrestrial flight was less dynamic than a lunar precision landing, being largely vertical and much shorter, so that state-triggered constraints to satisfy sensor visibility were unnecessary. The terrestrial DQG formulation used 10 discretization nodes, and employed a custom solver rather than the generic

Behçet Second Order Cone Program (BSOCP) implemented for this study, developed and provided by University of Washington [4]. The authors achieved a three second update rate in flight in the DLC.

This study demonstrates that DQG, an algorithm employing SCP methods, can be implemented as flight code to generate an on-board solution of the full constrained 6-DoF powered descent guidance problem for precision lunar landing through closed-loop simulation and performance bench-marking on representative hardware. The performance requirements for this landing problem are driven by the HDA phase as the final re-targeting problem requires rapid computation of the divert trajectory.

III. Methods

To demonstrate implementability of DQG for on-board guidance, we address the problems of generation of constrained reachability sets and of computation timing. A set of states from which DQG is capable of solving for safe and precise landing trajectories is generated through Monte Carlo simulation, and its robustness to these conditions which might be expected in a real lunar mission is demonstrated. Computation timing requirements levied by precision lunar landing problem are explored and DQG's performance evaluated against these requirements.

A. DQG formulation

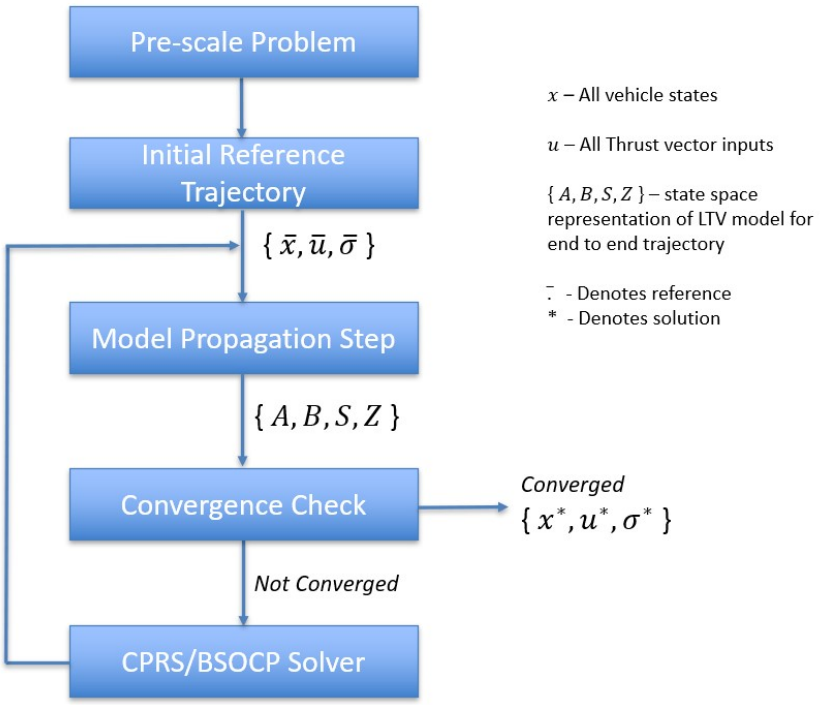
In order to achieve safe and precise landing utilizing hazard avoidance technology, an advanced guidance algorithm is required to reliably produce a trajectory that adheres to imposed constraints and is capable of doing so in a propellant efficient manner. These constraints may vary to ensure that hazard detection sensor suites are effective, that the vehicle is capable of flying the trajectory, and that the vehicle will reach its destination safely and accurately. The DQG algorithm aims to solve this challenging problem. The work builds on previous research into DQG performed by the University of Washington in partnership with NASA [2].

DQG uses unit dual-quaternions to represent combined vehicle translation and rotation states. This allows expression of some dynamics and constraints in convex forms more easily, while other constraints must be approximated and solved successively using SCvx.

DQG has been flown in an open-loop configuration on Blue Origin's New Shepard as a secondary payload under the Safe and Precision Landing Integrated Capabilities Evolution (SPLICE) [5] project at NASA. This previous implementation focused on providing guidance solutions for a terrestrial flight to demonstrate its capability in a flight configuration. Lunar trajectories require more time and energy to exhaust in order to achieve safe landing requirements, thus these trajectories tend to be much larger. They also tend to have more dynamic attitude profiles since fuel efficient lunar trajectories contain a large horizontal velocity component. Improvements to the algorithm have been made following the work done under SPLICE in order to meet stricter constraints for a lunar landing scenario. The vehicle constraints have been tailored for the Blue Moon lander, and they include a suite of visibility constraints that have not been flown on New Shepard.

1. Dual Quaternion Guidance Architecture

This section details the overall operation of the guidance algorithm. During execution, DQG is provided a navigation state update, estimated vehicle mass, and targeting commands. The algorithm then approximates the non-convex guidance problem with a series of convex sub-problems which are solved successively by a Second Order Cone Program (SOCP), and the iterative solutions converge to solve the original non-convex problem. The operational flow of the application is highlighted in Figure 1 where CPRS refers to a custom parser developed by University of Washington and detailed in [2].



Pre-scale Problem
Initial Reference Trajectory
$\{\bar{x}, \bar{u}, \bar{\sigma}\}$
Model Propagation Step
$\{A, B, S, Z\}$
Convergence Check
Converged
$\{x^*, u^*, \sigma^*\}$
Not Converged
CPRS/BSOCP Solver
x – All vehicle states
u – All Thrust vector inputs
$\{A, B, S, Z\}$ – state space representation of LTV model for end to end trajectory
$\bar{\cdot}$ - Denotes reference
$*$ - Denotes solution

Fig. 1 Dual Quaternion Guidance Application Architecture

Pre-scaling the Problem Before any computations are completed within the algorithm, the entire problem is scaled down from real world values in order to assist the SOCP with solving the problem within a reasonable number of iterations. These scale factors are determined by first defining a set of scale factors for handling time, mass, and position inputs and parameters. From there the corresponding scale factors for velocity, acceleration, force, torque, and vehicle inertia can be derived. The selection of the initial scale factor set is not trivial and has a direct effect on the convex optimization problem's conditioning, thus the selection of these initial scale factors must be chosen wisely. Analysis can be performed to determine a scale factor set that minimizes the problem conditioning.

Initial Reference Trajectory Generation The DQG algorithm requires an initial reference trajectory in order to complete model propagation. The algorithm currently determines an initial reference trajectory by linearly interpolating vehicle mass, attitude, position, angular rate, and velocity from the initial provided state to the final desired state. The attitude profile can be interpolated using spherical linear interpolation techniques for trajectories with more dynamic attitude profiles. The initial thrust trajectory is chosen to be equivalent to the opposite of the gravitational force imposed on the vehicle. The initial torque trajectory is computed to counter the moment induced by the initial thrust trajectory. This process generates a set of discrete nodes of state information from the current state to the final desired state, where the resolution of these nodes is a design choice. This initial guess is not a dynamically feasible trajectory, but the successive convexification process shapes this initial guess trajectory into a solution that is dynamically feasible and meets imposed constraints.

Model Propagation The equations of motion for a 6-DoF rocket problem are continuous, nonlinear, and time varying. In order to use the successive convexification architecture, these equations of motion must be transformed into a discrete linear time-varying (LTV) set. To accomplish this, the equations of motion must be linearized about a reference trajectory and then discretized to generate a state-space representation of the vehicle dynamics. A state space model representation of the vehicle dynamics is derived for each provided reference trajectory node, creating a set of linear time-invariant models at each node. This is done using a Runge-Kutta propagation method and discretization techniques.

Convergence Check This phase of the algorithm determines whether the acquired trajectory and thrust solution is dynamically feasible and converged on an optimal cost function solution. Dynamic feasibility is determined by evaluating the defects between each of the propagated nodes. If these defects meet a specified tolerance, then the reference trajectory is considered dynamically feasible. Cost function optimality is determined by measuring the differences between the solver's last trajectory state solution and the reference trajectory that the solution was linearized about. If the difference meets a specified tolerance, the solution is considered optimal.

Convex Optimization If the provided trajectory has not converged, then the reference trajectory and set of dynamic models generated during propagation are used to set up the convex optimization problem. The equality and inequality constraints are defined at application initialization, which utilizes the generated models, reference trajectory, and parameters needed to define the constraints. These values are updated as the dynamic models and reference trajectories change during execution.

A parser translates these constraints into a form usable by the convex optimization solver or the SOCP. The SOCP uses a primal-dual Interior Pointing Method to determine an optimal solution to the provided convex problem and the resultant solution is used to update the reference trajectory before cycling through the iterative process until a converged solution is found.

Enforced Constraints This section summarizes each of the vehicle and trajectory constraints that have been implemented into the DQG application for the Blue Moon reference trajectory. The constraints are broken into the following categories: 1) boundary condition constraints, 2) final condition constraints, 3) control constraints, and 4) visibility constraints. Boundary and final conditions constraints are used to constrain the overall trajectory. The control constraints are used to enforce vehicle specific hardware constraints that cannot be violated. Visibility constraints correspond to sensor and crew visibility requirements that must be met to ensure mission success. The enforced constraints are summarized in the following bulleted list.

1) Initial Condition Constraints
    - Equality constraint on initial vehicle mass
    - Equality constraint on initial vehicle attitude
    - Equality constraint on initial vehicle angular rate
    - Equality constraint on initial vehicle velocity
    - Equality constraint on initial vehicle altitude
    - Initial East & North initial position constrained to within a maximum distance
2) Final Condition Constraints
    - Final vehicle tilt angle
    - Equality constraint on final position
    - Final mass constrained to be greater than or equal to the vehicle dry mass
    - Final vertical velocity
    - Final horizontal velocity
    - Equality constraint on final vehicle angular rate
    - Final thrust magnitude equivalent to force of gravity
3) State Constraints
    - Maximum vehicle tilt angle constraint
    - Maximum vehicle angular rate per body frame axis
    - Minimum vehicle altitude constraint
4) Control Constraints
    - Maximum thrust magnitude
    - Minimum thrust magnitude
    - Maximum torque magnitude per body frame axis
    - Maximum main engine gimbal deflection
    - Maximum main engine thrust rate
    - Maximum main engine gimbal deflection rate
5) Visibility Constraints
    - Crew window maximum line of sight
    - HD sensor maximum line of sight
    - Maximum glide slope

B. Dynamics Model

As described in Section III.A.1, DQG propagates a dynamics model between optimization nodes. This model is formulated in a local East-North-Up reference frame centered at the landing target, and uses a uniform gravity model. For the New Shepard Tipping Point flights, uniform gravity was a valid approximation as the nominal trajectory was largely vertical. For lunar landing the nominal trajectory crosses a significant arc of the lunar surface, and a central body gravity model is necessary for trajectory optimization at powered descent initiation.

This study focuses its analysis on the HDA phase and divert as the most stressing case for DQG's capabilities. DQG's trajectory generation is limited to an area near enough to the landing site for a non-spherical gravity model to be valid. DQG is initially called prior to HDA at 8km slant range, then again for the HDA divert analysis.

C. Hazard Detection and Avoidance

HDA for the precision lunar landing problem encompasses several distinct phases of the landing process. An example scenario's concept of operations (CONOPs) with representative state values is shown in Figure 2.

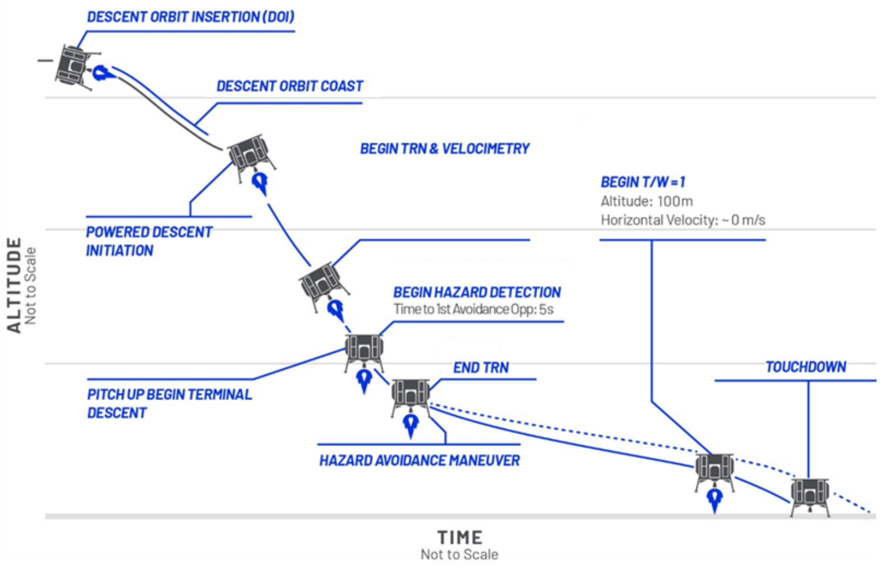

DESCENT ORBIT INSERTION (DOI)
DESCENT ORBIT COAST
BEGIN TRN & VELOCIMETRY
BEGIN T/W = 1
Altitude: 100m
Horizontal Velocity: ~0 m/s
POWERED DESCENT INITIATION
ALTITUDE
Not to Scale
BEGIN HAZARD DETECTION
Time to 1st Avoidance Opp: 5s
END TRN
TOUCHDOWN
PITCH UP BEGIN TERMINAL DESCENT
HAZARD AVOIDANCE MANEUVER
TIME
Not to Scale

Fig. 2 HDA CONOPs

The landing vehicle is first put into a descent orbit with a relatively small insertion burn at apolune on the opposite side of the moon to the landing site. The vehicle coasts to perilune, then ignites its landing engines for powered descent initiation (PDI). For a large initial fraction of its descent burn, the vehicle's attitude is largely horizontal to the surface. During this phase the terrain relative navigation (TRN) sensors have a chance to survey the surface and remove accumulated drift from the inertial navigation solution. Guidance and control is responsible for aligning the view angle of these sensors with the surface during this initial phase.

At a relatively low altitude the hazard detection sensors are able to resolve landing hazards and the hazard avoidance system can provide an adjusted landing target. The guidance is therefore responsible for ensuring that the HDA system has visibility and sufficient time to compute a safe landing target. Guidance brings the potential landing field into view of the HDA sensors by adjusting the vehicle's attitude, no longer constrained by TRN visibility.

HDA takes some time to survey the landing site and compute a safe landing target. In this study, this time is simulated as five seconds. The last moment the HDA system might provide a new target is at 500m altitude from the initial site. This timing means that the guidance system will receive a new target five seconds after the vehicle has crossed 500m altitude.

The vehicle diverts to a new target for the start of vertical descent before descending with minimal horizontal deviation.

Precision lunar landing therefore levies complex state-triggered constraints upon the descent trajectory. Guidance must produce a trajectory which satisfies attitude constraints during the TRN phase, different attitude constraints during the HDA divert phase, and fixed vehicle control constraints throughout the entire landing sequence as listed in Section III.A.1.

The trajectory prior to the HDA event must be planned to allow the HDA sensors to survey the landing site. Divert trajectory generation after the HDA event must occur fast enough to ensure safe landing subject to vehicle control constraints. SCP methods may provide a means to satisfy these constraints, if they can be feasibly employed.

D. 6-DoF simulation

A 6-DoF simulation of a lunar landing mission to the south pole is developed as a representative case for the precision lunar landing problem. This simulation enables open and closed-loop Monte Carlo analysis of DQG performance within a realistic HDA context. The simulation is developed in a Matlab/Simulink environment. The top-level simulation runs at a fixed discrete rate of 100Hz to support simulation with software in the loop.

The simulation models N-body gravity, sensor uncertainty, non-linear 6-DoF vehicle dynamics, and a typical set of thrust actuators including a main engine and reaction control system thrusters. The vehicle's inner-loop controllers track a reference guidance command for the vehicle's 6-DoF state.

The simulation begins prior to the TRN sensing phase. DQG is provided with an initial 6-DoF vehicle state and vehicle mass properties and it generates a nominal trajectory to the start of vertical descent satisfying the complex constraints levied by the vehicle and the HDA CONOPs. The vehicle tracks DQGs initial trajectory through the HDA event.

The HDA CONOPs are simulated using a simple slant range trigger and a time delay. The time delay, less than five seconds, is representative of the time for an HDA system to scan the landing zone and locate a safe landing site. Once selected, the new targeted vertical descent location and a command to compute a re-targeting trajectory are passed as an input to DQG.

Figure 3 shows a timeline of the simulated events during powered descent, with colored lines representing the relative duration of each phase. The vehicle's engine lights and flies a simple braking guidance until it is close enough to the landing site for DQG's uniform gravity model to apply. DQG is then asked for a guidance solution and once its computation time has elapsed, the vehicle guidance switches to DQG. From the beginning of the visibility phase to a short time prior to HDA, the vehicle's terrain relative sensors must be pointed at the ground. The current implementation of DQG is not configured to handle this constraint explicitly, but the roll angle is otherwise unconstrained during this phase of flight under nominal conditions so the TRN pointing constraint is of low priority for optimization. At 500 m slant range the HDA system begins computing a new landing location. After five seconds it sends the new landing location to DQG and requests a divert. DQG computes a divert trajectory solution and after its computation time has elapsed the vehicle attempts to follow the new trajectory. The simulation ends once DQG's time-to-go reaches zero.

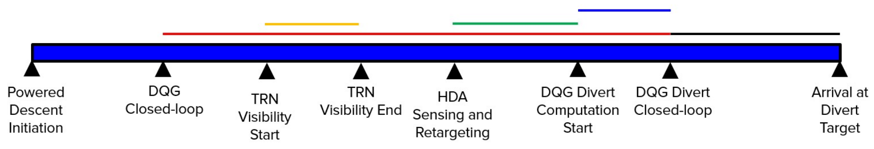


Fig. 3 Simulated Events Timeline

DQG computes a new trajectory to the re-targeted site which is not subject to the TRN and HDA state triggered constraints. The vehicle tracks this new trajectory to the beginning of vertical descent, then switches to a simpler guidance solution for the vertical descent phase. DQG's performance is thus measured against the targeted state, rather than the final landing state.

Figure 4 shows a simulated vehicle groundtrack through the HDA and Divert events. The dotted line denotes a 50m lateral divert range from the original target, centered at the origin. Between HDA and Divert the HDA system computes a new target in five seconds, leaving very little time for DQG to provide its divert trajectory.

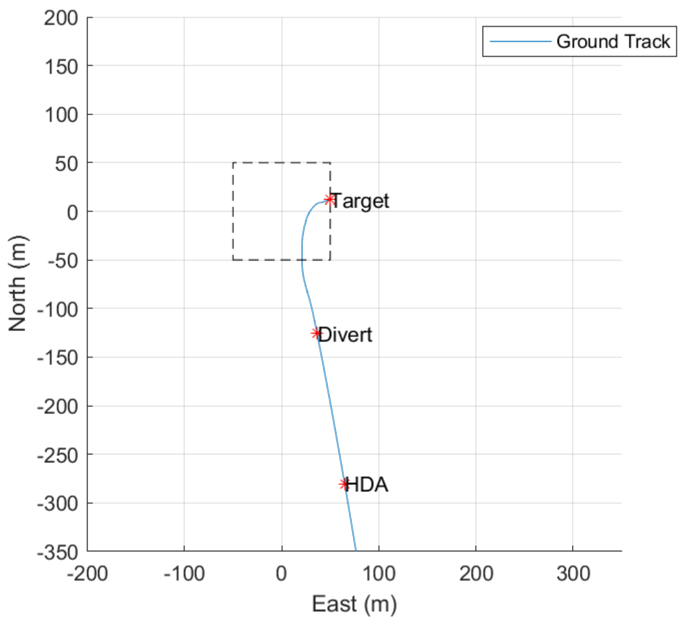


Fig. 4 Divert Groundtrack

1. Monte-Carlo

6-DoF simulation enables Monte Carlo testing for robustness of the guidance algorithm. Robustness testing and generation of constrained reachability sets is performed using the NASA TRICK simulation, developed under the NASA SPLICE program. Dispersed parameters include initial position, velocity, and angular rate as detailed in Table 1. These dispersions are used for open-loop robustness testing of DQG which allows for tuning and refinement of the algorithm. Monte Carlo testing is performed with 1000 runs.

DQG, and convex programming methods in general, require well chosen initial conditions for guaranteed convergence. Mission planning must ensure that the vehicle arrives at PDI in a state from which DQG is able to generate a safe and precise trajectory solution. This is true of all on-board guidance algorithms, but for convex programming methods it can be difficult or impossible to know the space of feasible initial conditions.

The NASA TRICK simulation is used to generate a constrained reachability set of initial conditions at the HDA event by dispersing the initial vehicle state and the targeted divert position and generating trajectories with DQG. The initial states are again dispersed according to Table 1 about the nominal trajectory, which is generated by DQG from an earlier state of powered descent. This simulates calling DQG on-board for retargeting. The divert position is dispersed up to 100m radially from the central target. The simulated vehicle then follows the trajectory in the 6-DoF simulation to confirm feasibility and performance.

Parameter	**Uniform Dispersion**
Position	±100 m per axis
Velocity	±5 m/s per axis
Angular Velocity	±1.15 deg/s per axis

Table 1 Divert Initial Condition Dispersion

2. Software-in-the-loop

Closed-loop testing is performed with both the NASA TRICK simulation as well as a high-fidelity simulation developed by Blue Origin in Simulink. The DQG software used in this application was originally developed as C code as part of the NASA SPLICE project. This software is called from Simulink through an s-function wrapper within the simulated vehicle's flight manager. Its generated trajectories are interpolated on the solved burn time to provide an instantaneous guidance command, which is followed by the vehicle's inner-loop controllers during closed-loop testing. Both open and closed-loop testing are performed.

In open-loop testing the trajectories generated by DQG are examined independently from the vehicle model's ability to track them. To determine the set of states from which DQG is capable of generating feasible trajectories for the HDA re-targeting phase, the vehicle state from the nominal trajectory following HDA re-targeting is sampled for initial conditions passed to DQG. These conditions are swept across the altitude and velocity axes to find the limits at which DQG can no longer generate feasible trajectories to satisfy the precision lunar landing constraints, providing a reachability set for trajectory design. The vehicle's state must belong to this set at the beginning of the re-targeting maneuver.

The re-targeting event requires more rapid computation of a new trajectory by an on-board guidance algorithm than the initial nominal trajectory due to the much smaller remaining time to execute a divert maneuver. Additional time spent generating a divert trajectory while following the previous nominal trajectory directly reduces potential divert capability. There is therefore a maximum allowable trajectory generation time for a given initial and final state while satisfying the precision landing constraints.

Closed-loop testing is performed to refine and validate the constraints imposed upon DQG's trajectories. The simulated vehicle is flown through powered descent following DQG's guidance commands, and its resulting trajectory is checked against the mission constraints. The vehicle's control system translates DQG's 6-DoF guidance commands into actuator commands, and the actuator controllers drive physical models of the actuators.

DQG provides a full 6-DoF guidance command but the plant will not exactly match DQG's model due to many sources of error including higher-order gravitational effects, propellant slosh, control actuator errors, and navigation error (though navigation error is excluded from this analysis). The inner-loop controllers compute tracking error between DQG's 6-DoF trajectory and the 6-DoF navigation states and apply linear feedback.

DQG's computation time is simulated artificially to investigate the closed-loop timing performance requirements using a Stateflow sequencer, which triggers DQG computation. The DQG software generates a new trajectory within one simulation frame, but it does not become available to the flight software until the specified discrete simulation frames (ticks) have passed. This architecture, rather than directly modeling DQG's computation time with a fixed discrete simulation rate, allows changing the simulated computation time without code recompilation.

3. Hardware-in-the-Loop

DQG is implemented on a representative DLC for performance testing and real-time implementability. The DLC has been developed through the SPLICE program, and its hardware specifications are detailed in [6]. DQGs computation timing is evaluated open-loop on the DLC using initial conditions drawn from the previously defined reachability set. For open-loop testing a simple test driver in a Linux environment calls DQGs execution function with defined inputs.

IV. Results

A. Reachability

Provided an initial condition sampled from the nominal trajectory after a simulated HDA event as described in Section III.D.2, DQG is able to generate feasible trajectories for a wide range of divert targets. Figure 5 shows generated divert trajectories for different targets uniformly distributed across a square centered on the nominal target. These represent a divert capability, not a landing accuracy.

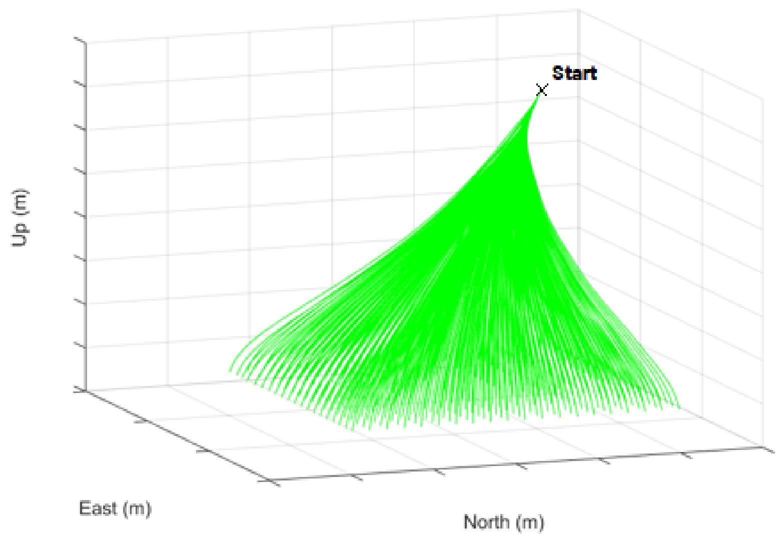


Fig. 5 Divert Reachability

This extensive divert feasibility suggests that DQG is robust to a large divert envelope under nominal conditions. Figure 6 shows DQG's estimated final mass estimate contour for this sweep. The vehicle enters from the South. The off-center maximum suggests that the original trajectory design was propellant sub-optimal, and might be improved by a small change to the approach trajectory. With a central gravity model, DQG could be active at powered descent initiation and may be able to make these improvements to the trajectory as well as other changes to ease satisfaction of the complex LOS constraints.

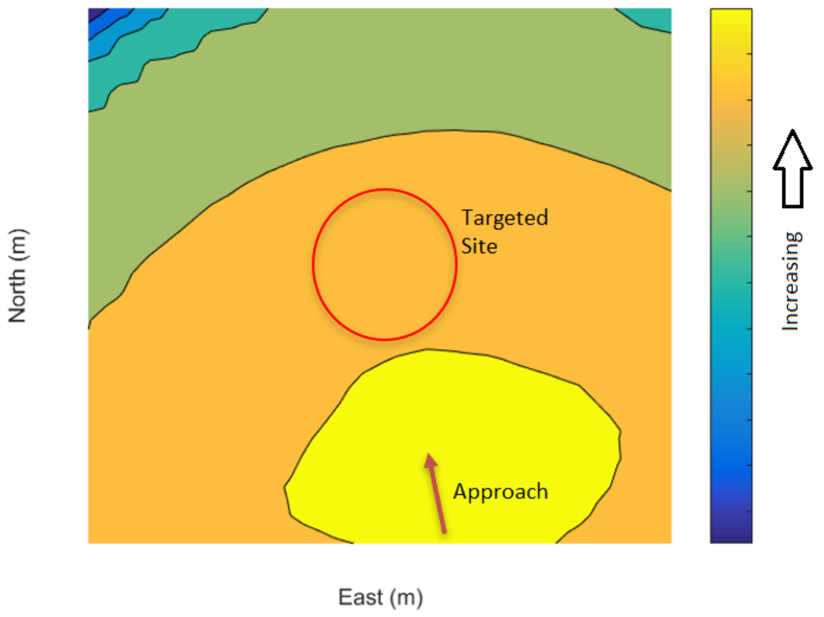


Fig. 6 Divert Final Mass

Figure 7 shows 1000 closed-loop Monte Carlo runs with dispersed initial and final conditions as listed in Table 1. These states represent a constrained reachability set; from the sampled initial conditions, DQG can be expected to provide feasible safe and precise landing trajectories anywhere in the dispersed divert region. Again, these represent divert capability from the dispersed initial conditions rather than landing accuracy. Some of these trajectories have non-monotonic altitude due to the initial condition dispersion, particularly in velocity and angular rate. In reality, conditions at the HDA are unlikely to result in such drastic maneuvers but these outliers demonstrate robustness of the algorithm and the vehicle.

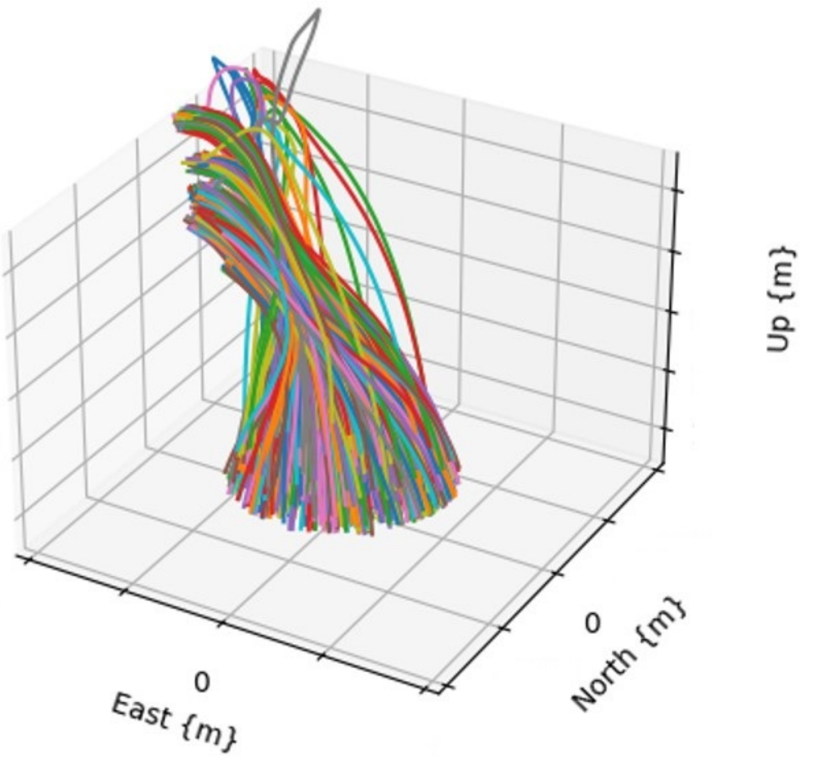


Fig. 7 Reachability Set

B. HDA Requirements

In the closed-loop simulation, the vehicle was flown on a nominal trajectory until the HDA-triggered divert. In this simulation, the vehicle crossed 500m slant range to the nominal target at 852 seconds, and a new landing target was provided to DQG five seconds later at 857 seconds.

1. Closed-loop tracking

The ability of the simulated vehicle to track DQG's position trajectory through the HDA divert event is shown in Figures 8 through 10. Here, DQG's simulated computation time is one second, three seconds, and five seconds respectively. DQG is only called once for the divert, as for computation times larger than 1 second the subsequent updates often struggle to converge for such short trajectories. A single update also illustrates the errors resulting from the computation delay. The quantity plotted is tracking error, calculated as in Equation 1 for each axis where $scale_{pos}$ is the maximum tracking error experienced in the one second case. The results are given in an Up-East-North coordinate frame centered on the initially targeted landing site.

$$error = \frac{pos_{DQG} - pos_{Nav}}{scale_{pos}} \tag{1}$$

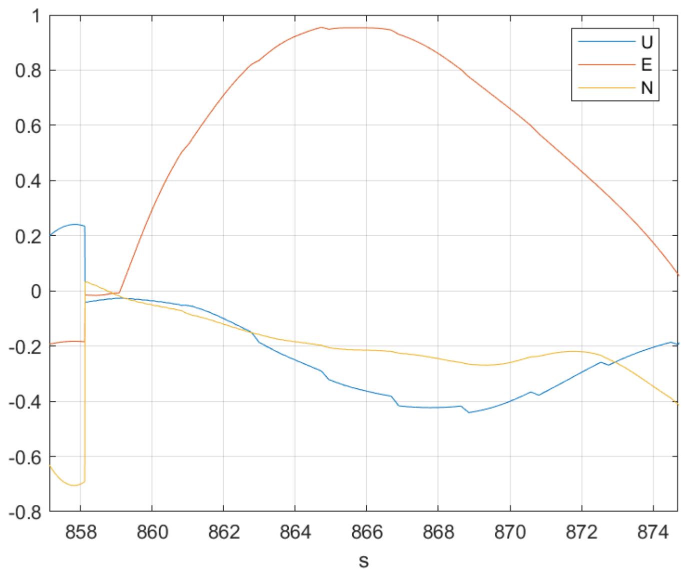


Fig. 8   Pos Err: 1s Timing

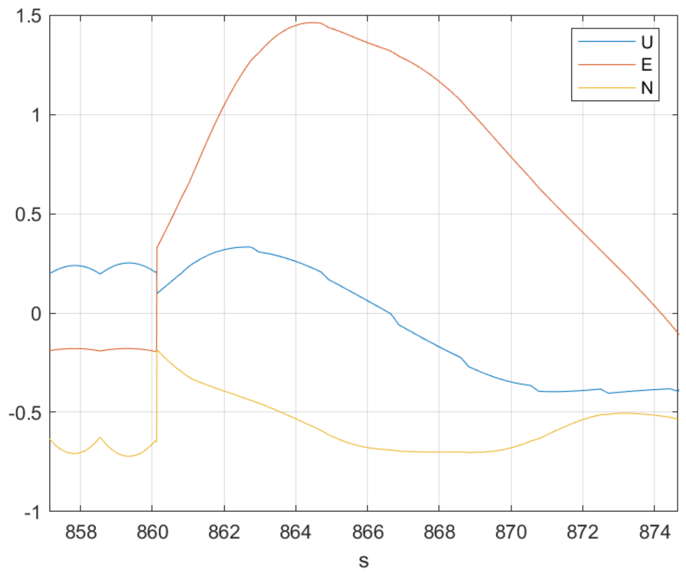


Fig. 9   Pos Err: 3s Timing

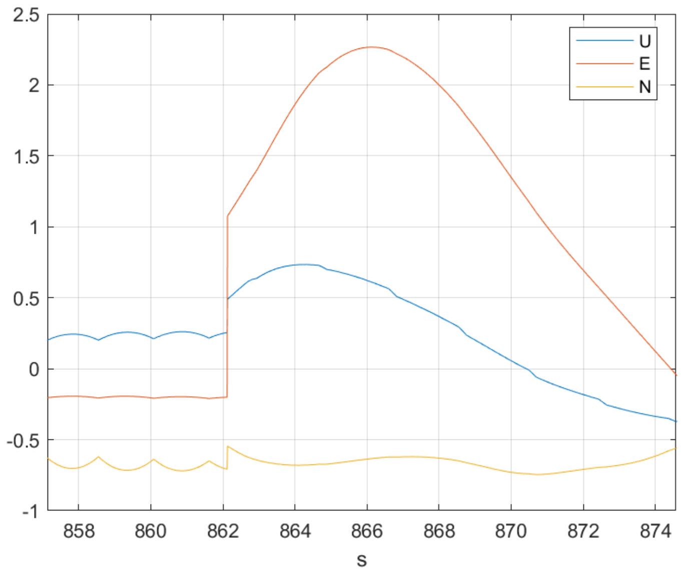


Fig. 10 Pos Err: 5s Timing

Each of the position tracking error figures shows an existing tracking error prior to generation of the divert solution. This error, as well as its oscillation, are primarily due to propellant slosh in this high fidelity simulation, a real phenomenon that can also levy complex constraints on a powered descent trajectory but which is not explicitly addressed in this analysis. Figure 8 shows this error dropping to nearly zero at 858 seconds, when DQG provided its updated guidance solution. Figures 9 and 10 do not show as much improvement. Were DQG's computation instantaneous, tracking would be expected to be initially perfect because DQG resets its initial guidance command to the vehicle's exact navigation state. However, due to the simulated delay DQG's initial guidance command lags the navigation state. This effect is exacerbated by increasing computation time.

The position tracking error plots also show increasing error beyond the initial lag, particularly in the east direction. This is due to initial error in the velocity state primarily by the same lag. Figures 11 through 13 show the velocity tracking error, again computed as in Equation 2 for each axis with $scale_{vel}$ equal to the maximum velocity tracking error for the one second case.

$$error = \frac{vel_{DQG} - vel_{Nav}}{scale_{vel}} \tag{2}$$

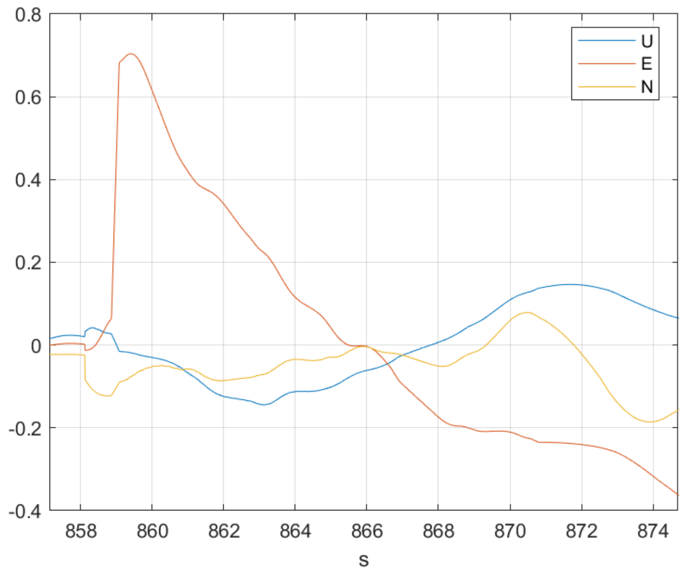


Fig. 11 Vel Err: 1s Computation

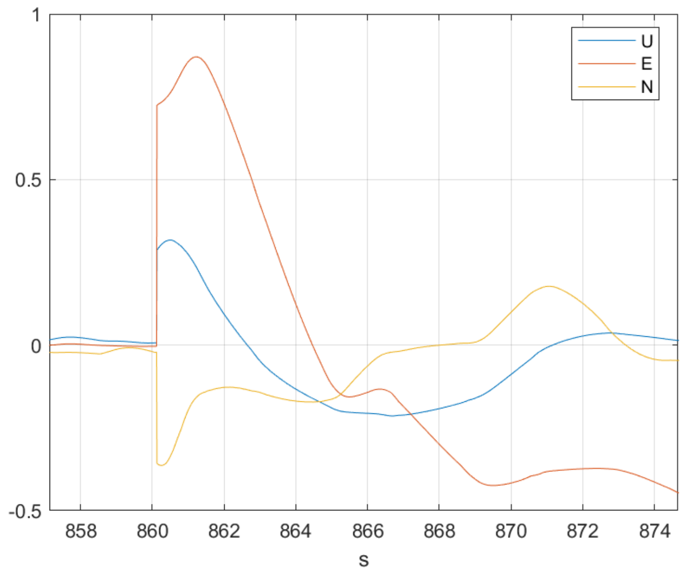


Fig. 12 Vel Err: 3s Computation

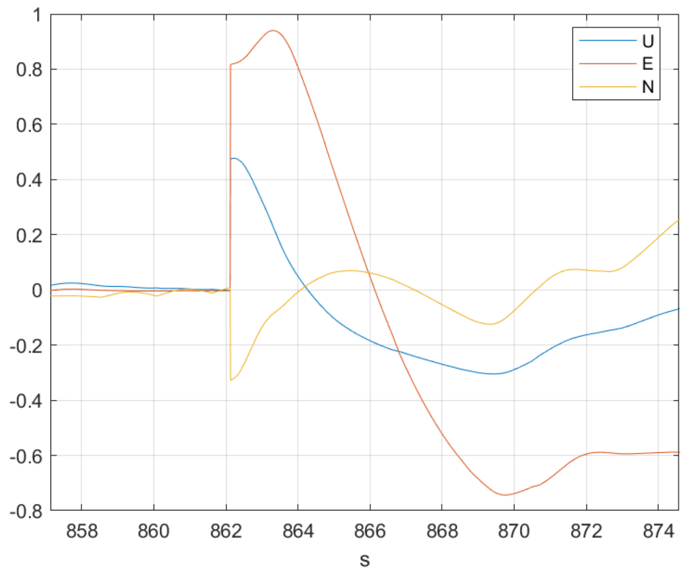


Fig. 13 Vel Err: 5s Computation

These results do not reveal a hard computation timing requirement for DQG, but they do show a clear trend of poor tracking during the divert with increasing computation time as shown in Figure 14. The final position and velocity error requirements for a safe precision landing imply a computation time requirement of less than three seconds based upon this trend. Tracking error plots and trends for simulated timing beyond five seconds are given in the Appendix, and they follow this trend of increasing error.

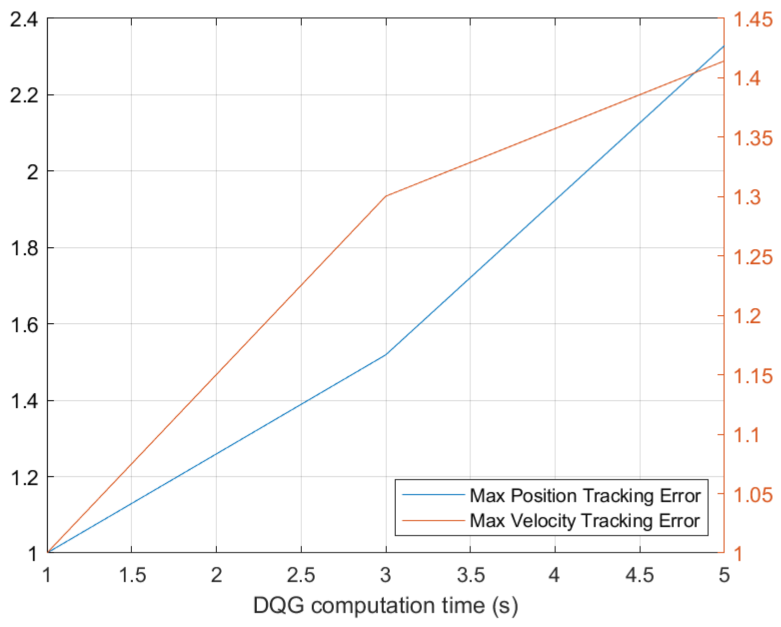


Fig. 14 Maximum Tracking Error vs. Computation Time

These results also reveal an important consideration for future algorithm development. DQG does not take into account its own computation time, but if its timing could be known or specified reliably then DQG or the flight manager could propagate a previous guidance solution through that time, providing an estimated initial condition corresponding to the expected vehicle state at the end of computation. The guidance solution would much more closely match the vehicle state upon completion of the update cycle. The methods in Section IV.C provide a means of determining a maximum required computation time, which could be a fixed number of cycles on target hardware and would allow precise knowledge of the delay time for this propagation.

C. Computation time

DQG's computation time depends upon the problem to be solved since SCP is an iterative solution method. Figures 15 and 16 show the distributions of DQG computation timing while solving a dispersed set of diverts from the HDA scenario while running on the DLC.

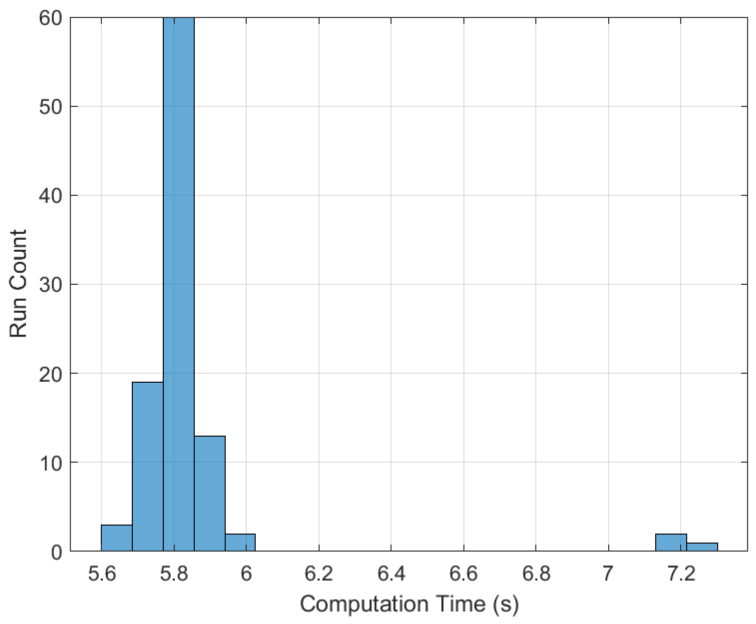


Fig. 15 Computation Time Distribution: 10 Nodes

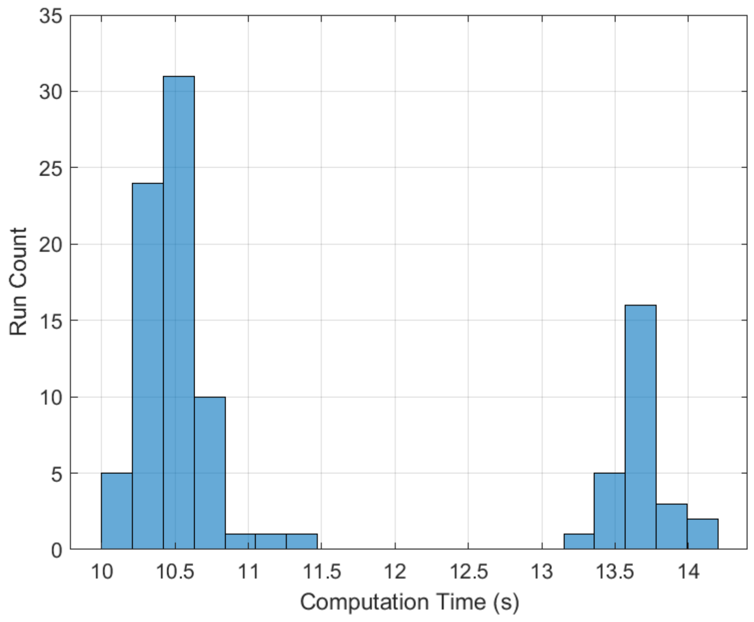


Fig. 16 Computation Time Distribution: 20 Nodes

These distributions show two important characteristics of the timing performance. First, They are bimodal, with the smaller set of larger computation times corresponding to an additional successive convexification iteration required by the problem. Certain scenarios, while potentially feasible, may pose a harder problem for DQG to solve and a resulting longer computation time.

Second, they show that the timing performance is strongly dependent upon the optimal control problem's discretization. With 20 optimization nodes, DQG achieves a mean update time of 11.36 seconds on the DLC. With 10 optimization nodes, the mean is cut nearly in half to 5.85 seconds. This discretization is chosen to ensure validity of DQG's dynamics model as well as to enforce constraints along the trajectory. Figure 17 shows two open-loop trajectories from 8000m slant range, generated by the 20 node and 10 node DQG variants. The 3D trajectories only differ slightly between the variants. Figure 18 shows the sensor LOS at each node of the DQG trajectories vs. slant range. The state-triggered LOS constraints are shown in the bottom left corner of the plot, as they become active between 1000m and 500m slant range. Prior to 1000m slant range, the LOS angle is not constrained. Between 1000m and 500m slant range the LOS angle is constrained to be less than 35 degrees.

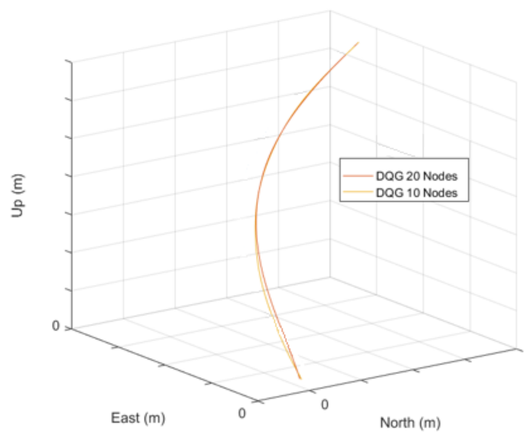


Fig. 17 Comparison of Trajectories Between Variants

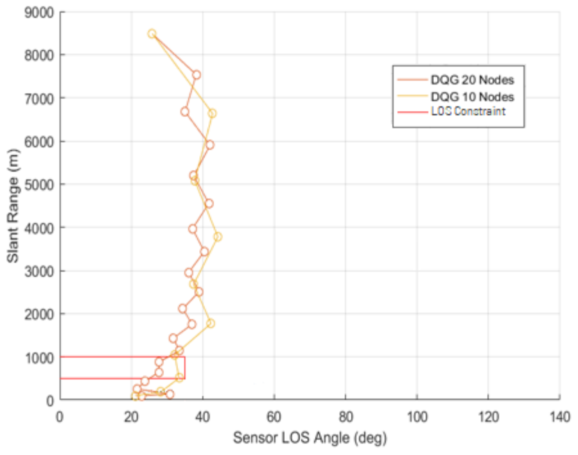


Fig. 18 Comparison of Line-of-sight Angle Between Variants

The 20 Node trajectory has several optimization nodes within the constrained zone, while the 10 Node trajectory only has one node within the zone. Since DQG generates the entire trajectory subject to that node's constraint, the 10 Node trajectory looks similar to the 20 Node trajectory. If the discretization were any coarser there may have been no optimization nodes subject to this constraint. However, periodically updating DQG throughout the landing phase has the effect of increasing the node resolution as the vehicle approaches these complex HDA constraints. As a result, 10 nodes are found sufficient for the precision lunar landing problem.

For the ten node variant of DQG, the DLC is able to compute fully constrained 6-DoF trajectories in under 6 seconds. This is slower than the update time required for HDA.

In [4] the authors found that a custom solver generated for the specific application using BSOCP can reduce computation time by half for small problems relative to BSOCP. The ten node variant of DQG presented here consists of the solution variable counts in Table 2. The authors of [4] would not consider this a small problem and find that problems of this size may not realize much benefit from a custom solver. However, the DQG formulation in this study uses nearly an order of magnitude fewer second-order cones. In [3], the authors employed such a custom solver on a very similar formulation of DQG and realized an update rate on the DLC in flight of three seconds. If this reduction were realized for the lunar landing problem, DQG would meet the three second timing target on the DLC.

Solution Variables	**1891**
Linear Cones	860
Second-Order Cones	45

Table 2 Solution Variable Count

D. Hardware Dependent Timing Performance

With 10 optimization nodes, DQG running on the DLC can generate trajectory solutions in an average of 5.85 seconds as shown in Figure 15. Figure 19 shows the same timing test results for an Intel Xeon W-3245 CPU @ 3.20GHz.

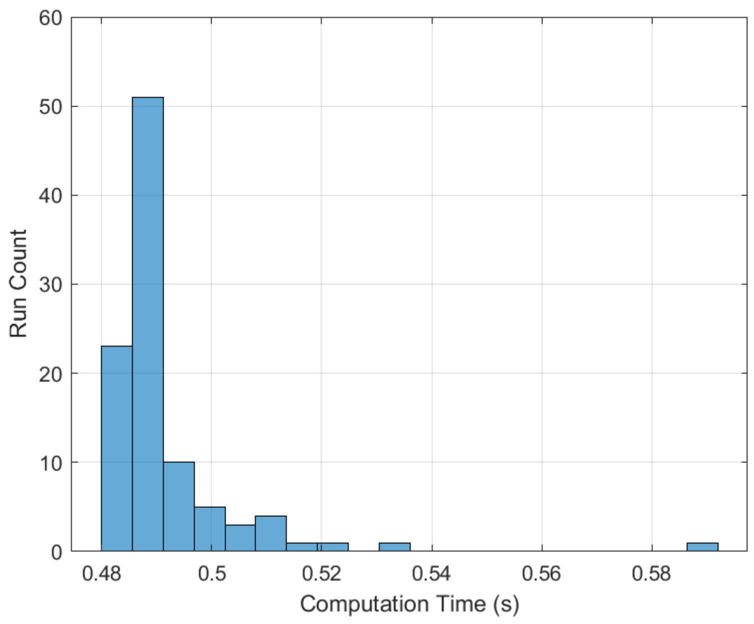


Fig. 19 Desktop Computation Time Distribution: 10 Nodes

DQG achieves a mean update time of 0.49 seconds and a maximum of 0.59 seconds on the Intel Desktop. While space flight hardware is subject to very different constraints than desktop simulation machines, selection of flight hardware heavily impacts performance and there may be significant improvements made through hardware changes. The order of magnitude improvement demonstrated in this simple comparison is very promising for the feasibility of on-board implementation. These results provide a rough benchmark of achievable performance on more modern processors, similar to the results from [2]. Further study on hardware selection and possible improvements is presented in [7].

V. Conclusions / Future Work

The full 6-DoF precision lunar landing problem can be solved on-line using SCP methods in real-time, but DQG does not do so fast enough to satisfy the HDA CONOPs on the DLC in its current implementation. Its computation time on more powerful hardware is well within the requirement. The algorithm is also powerful mission design tool which can be used to rapidly generate complex constrained trajectories.

A constrained reachability set is developed for validation of the DQG's robustness and for use in mission design as a boundary condition constraint for previous mission phases.

A few improvements are identified that may be made to the current implementation of the DQG algorithm to support precision lunar landing in future work.

A. Central Gravity Model

DQG is currently limited to generating trajectories near the intended landing site, due primarily to a uniform gravity model. This assumption fails as the curvature of the lunar surface becomes significant. With a central gravity model, DQG could optimize the entire landing trajectory and extract additional mass savings and robustness to complex constraints.

B. Computation Time Propagation

The computation time required for DQG to produce a trajectory is significant enough that by the time a trajectory is available it is out of date, leading to tracking error and possible failure to meet safe landing constraints. This issue is common to any onboard guidance algorithm implementation, but DQG's computation time is on the order of the demonstrated timing requirement for precision lunar landing. DQG does not currently attempt to account for this delay.

It could instead propagate the provided initial condition forward by a known delay using its internal dynamic models and a previous guidance solution. Once computation is complete, the trajectory would be much more closely aligned with the vehicle's state.

C. Custom Solver

To achieve a three second update timing on the DLC, a custom SOCP solver of the kind developed in [4] may be necessary. Future work is planned to implement this custom solver for DQG.

VI. Acknowledgments

The work presented in this paper was completed under the public-private partnership between Blue Origin and NASA through the Precision Lunar Landing System Announcement of Collaborative Opportunity started in 2019.

This work was co-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. All other rights are reserved by the copyright owner.

References

[1] 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, p. 0844.

[2] Reynolds, T. P., Szmuk, M., Malyuta, D., Mesbahi, M., Açkmee, B., and Carson, J. M., "Dual Quaternion-Based Powered Descent Guidance with State-Triggered Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 9, 2020, p. 15841599. https://doi.org/10.2514/1.g004536, URL http://dx.doi.org/10.2514/1.G004536.

[3] Fritz, M., Doll, J., Ward, K., Mendeck, G., Sostaric, R., Pedrotty, S., Kuhl, C., Açkmee, B., Bieniawski, S., III, L. S., and Jr., A. B., "Post-Flight Performance Analysis of Navigation and Advanced Guidance Algorithms on a Terrestrial Suborbital Rocket Flight," *AIAA Scitech 2022 Forum*, 2022.

[4] Dueri, D., and Acikmese, B., "Automated Custom Code Generation for Embedded, Real-time Second Order Cone Programming," *IF AC Proceedings*, , No. 47, 2014, pp. 1605–1612.

[5] 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. https://doi.org/10.2514/6.2021-0256.

[6] Rutishauser, D., and Tse, T., "Hardware-in-the-Loop Testing for Suborbital Flights of the Safe and Precise Landing Integrated Capabilities Evolution (SPLICE) Project Technologies," *AIAA Scitech 2020 Forum*, 2020. URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-0367.

[7] Rutishauser, D., Ramadorai, R., Prothro, J., Fleming, T., and Fidelman, P., "NASA and Blue Origin Collaborative Assessment of Precision Landing Algorithms and Computing," *AIAA Scitech 2021 Forum*, 2021. URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0377.

Appendices

A. Tracking Error

Figures 20 through 25 show tracking error plots as in Section IV.B, with computation times from 7.5s to 15s. At these extended computation times, the tracking error is so large from the effect of the delay that the vehicle is effectively flying to the final targeted state under PID control, with error saturation on the position. DQG's trajectory is lost in the error, and these computation times are likely infeasible.

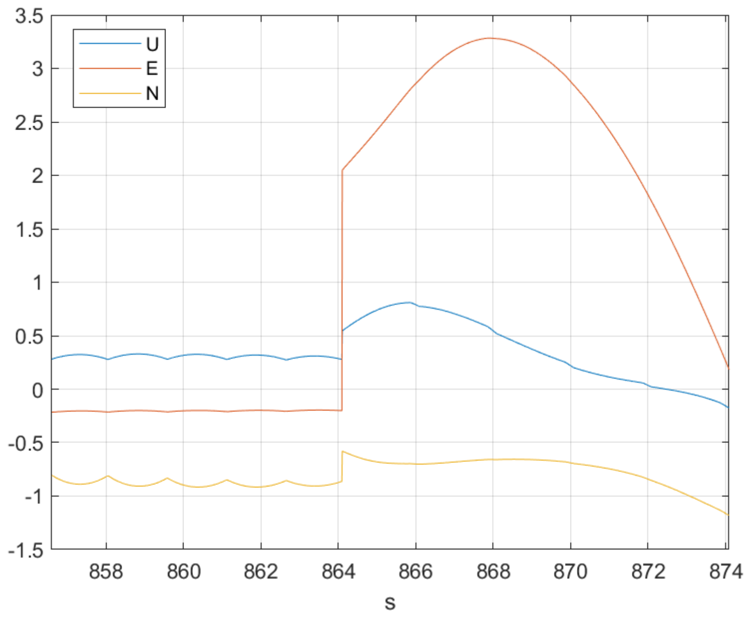


Fig. 20 Pos Err: 7.5s Timing

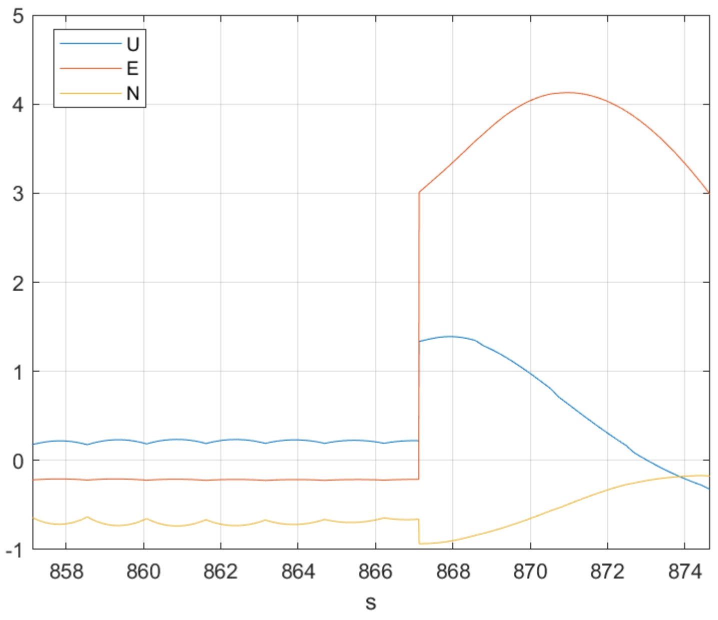


Fig. 21 Pos Err: 10s Timing

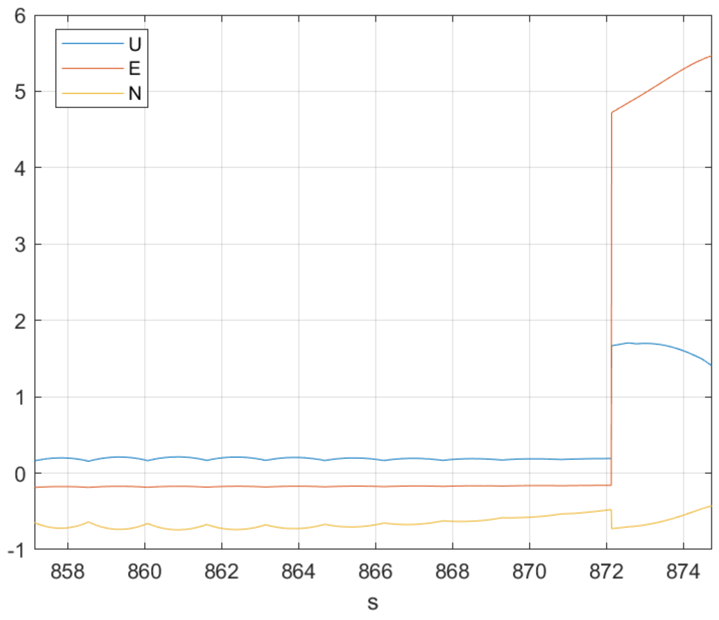


Fig. 22 Pos Err: 15s Timing

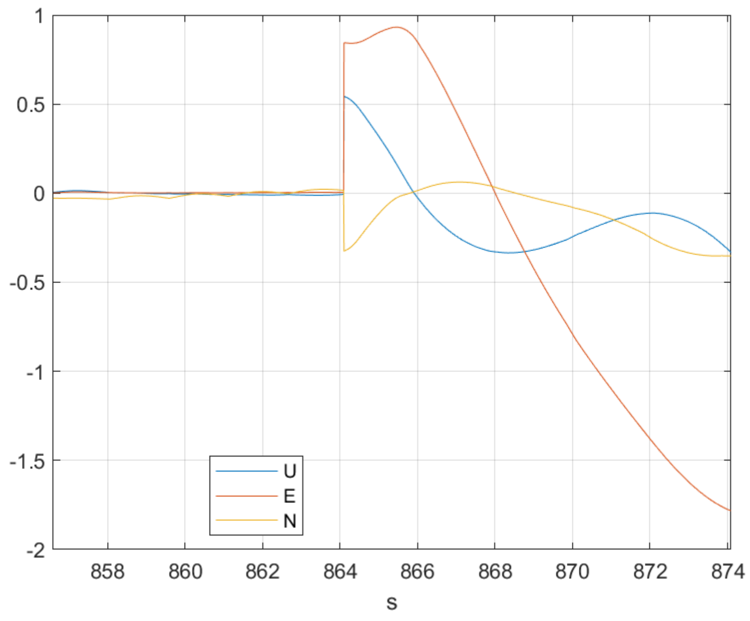


Fig. 23 Vel Err: 7.5s Timing

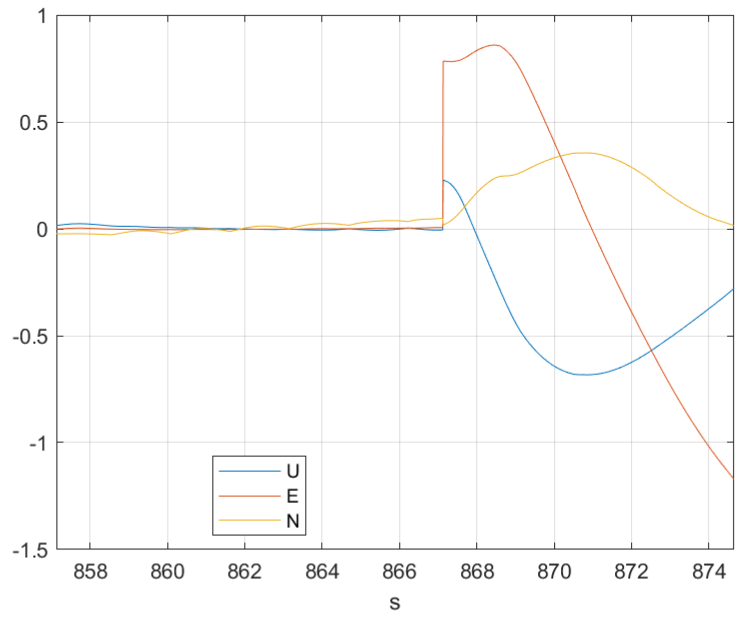


Fig. 24 Vel Err: 10s Timing

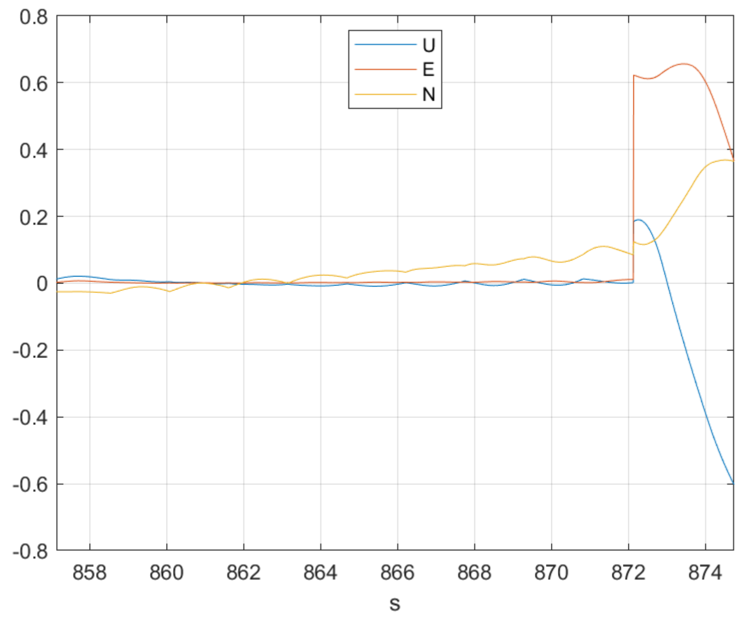


Fig. 25 Vel Err: 15s Timing

Figure 26 shows the maximum tracking error during the divert maneuver for all computation times up to 15 seconds. The position error is monotonic, but the velocity error appears to decrease between 7.5 and 15 seconds. This is because the 15 second update encompasses almost the entire divert maneuver. The vehicle's position by the time DQG's trajectory is available is nearly 50m from the divert target since it has almost arrived at the originally targeted site. Consequently, the vehicle's velocity is relatively low as would be expected near the end of a divert maneuver, so the error between DQG's velocity trajectory and the vehicle's true velocity is small.

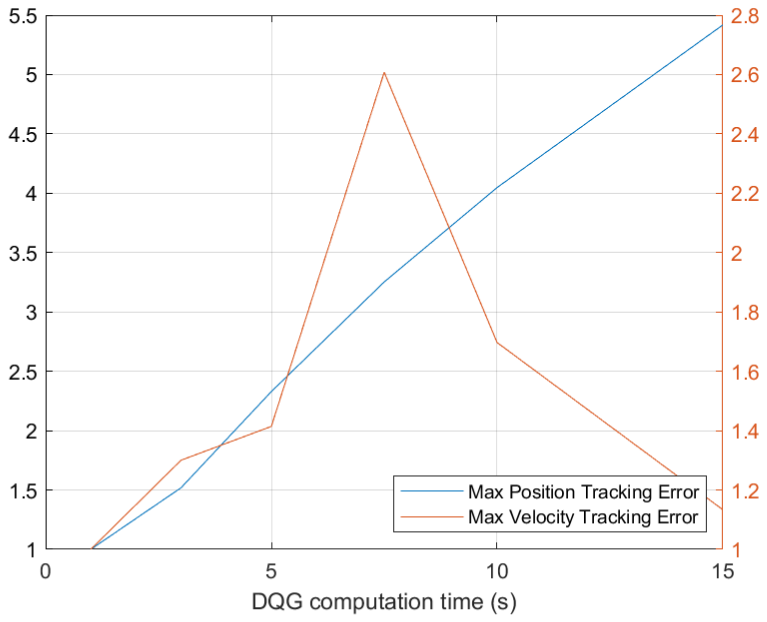


Fig. 26 Maximum Tracking Error vs. Computation Time