Planning with Attitude
Brian E Jackson¹, Kevin Tracy¹, and Zachary Manchester¹

Abstract—Planning trajectories for floating-base robotic systems that experience large attitude changes is challenging due to the nontrivial group structure of 3D rotations. This paper introduces a powerful and accessible approach for optimization-based planning on the space of rotations using only standard linear algebra and vector calculus. We demonstrate the effectiveness of the approach by adapting Newton's method to solve the canonical Wahba's problem, and modify the trajectory optimization solver ALTRO to plan directly on the space of unit quaternions, achieving superior convergence on problems involving significant changes in attitude.

Index Terms—Motion and Path Planning, Optimization and Optimal Control, Computational Geometry, Underactuated Robots, Motion Control,

I. INTRODUCTION

Many robotic systems—including quadrotors, airplanes, satellites, autonomous underwater vehicles, and quadrupeds—can perform arbitrarily large three-dimensional translations and rotations as part of their normal operation. While representing translations is straightforward and intuitive, effectively representing the nontrivial group structure of 3D rotations has been a topic of study for many decades. Although we can intuitively deduce that rotations are three-dimensional, a globally non-singular three-parameter representation of the space of rotations does not exist [1]. As a result, when parameterizing rotations, we must either a) choose a three-parameter representation and deal with singularities and discontinuities, or b) choose a higher-dimensional representation and deal with constraints between the parameters. While simply representing attitude is nontrivial, generating and tracking motion plans for floating-base systems is an even more challenging problem.

Early work on control problems involving the rotation group dates back to the 1970s, with extensions of linear control theory to spheres [2] and $SO(3)$ [3]. Effective attitude tracking controllers have been developed for satellites [4], quadrotors [5–10], and a 3D inverted pendulum [11] using various methods for calculating three-parameter attitude errors.

More recently, these ideas have been extended to trajectory generation [12], sample-based motion planning [13], [14], and optimal control. Approaches to optimal control with attitude states include analytical methods applied to satellites [15], discrete mechanics [16–18], a combination of sampling-based planning and constrained trajectory optimization for satellite formations [19, 20], projection operators [21], or more general theory for optimization on manifolds [22]. Nearly all of these methods rely heavily on principles from differential geometry and Lie group theory; however, despite these works, many recent papers in the robotics community continue to naively apply standard methods for motion planning and control with no regard for the group structure of rigid body motion.

In this paper, we make a departure from previous approaches to geometric planning and control that rely heavily on ideas and notation from differential geometry, and instead use only basic mathematical tools from linear algebra and vector calculus that should be familiar to most roboticists. In Sec. III, we introduce an approach to quaternion differential calculus similar to [23, 24], but significantly simpler and more general, enabling straightforward adaptation of existing algorithms to systems with quaternion states. For concreteness, we then apply our method to the canonical Wahba’s problem [25] in Sec. IV, and demonstrate superior convergence to approaches that fail to properly account for the group structure. In Sec. V, we extend these ideas to the problem of trajectory optimization, and detail modifications to ALTRO, a state-of-the-art constrained trajectory optimization solver, and demonstrate performance gains on several benchmark problems. With the modifications presented in this paper, ALTRO explicitly leverages both the structure of the trajectory optimization problem as well as the group structure of 3D rotations, making it uniquely well-suited to solving challenging problems with near real-time performance.

In summary, our contributions include:

• A unified approach to quaternion differential calculus entirely based on standard linear algebra and vector calculus.
• Derivation of a Newton-based algorithm for nonlinear optimization directly on the space of unit quaternions.
• Implementation of a fast and efficient solver for trajectory optimization problems with attitude dynamics and nonlinear constraints that correctly accounts for the group structure of 3D rotations.

II. Background

We begin by defining some useful conventions and notation. Attitude is defined as the rotation from the robot’s body frame to the world frame. We also define gradients to be row vectors, that is, for $f(x) : \mathbb{R}^n \to \mathbb{R}$, $\frac{\partial f}{\partial x} \in \mathbb{R}^{1 \times n}$.
A. Unit Quaternions

We leverage the fact that quaternions are linear operators and that the space of quaternions $\mathbb{H}$ is isomorphic to $\mathbb{R}^4$ to explicitly represent—following the Hamilton convention—a quaternion $q \in \mathbb{H}$ as a standard vector $q \in \mathbb{R}^4 := [q_s \ q_v^T]^T$ where $q_s \in \mathbb{R}$ and $q_v \in \mathbb{R}^3$ are referred to as the scalar and vector parts of the quaternion, respectively. The space of unit quaternions, $\mathbb{S}^3 = \{q : ||q||_2 = 1\}$, is a double-cover of the rotation group $SO(3)$, since $q$ and $-q$ represent the same rotation.

Quaternions multiplication is defined as

$$ q_2 \otimes q_1 = L(q_2)q_1 = R(q_1)q_2 \quad (1) $$

where $L(q)$ and $R(q)$ are orthonormal matrices defined as

$$ L(q) := \begin{bmatrix} q_s & -q_v^T \\ q_v & q_s I + [q_v]_\times \end{bmatrix} \quad (2) $$

$$ R(q) := \begin{bmatrix} q_s & -q_v^T \\ q_v & q_s I - [q_v]_\times \end{bmatrix}, \quad (3) $$

and $[x]_\times$ is the skew-symmetric matrix operator

$$ [x]_\times := \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \quad (4) $$

The inverse of a unit quaternion $q^{-1}$, giving the opposite rotation, is equal to its conjugate $q^*$, which is simply the same quaternion with a negated vector part:

$$ q^* = T q := \begin{bmatrix} 1 \\ -I_3 \end{bmatrix} q. \quad (5) $$

The following identities, which are easily derived from (2)–(3), are extremely useful:

$$ L(T q) = L(q)^T = L(q)^{-1} \quad (6) $$

$$ R(T q) = R(q)^T = R(q)^{-1}. \quad (7) $$

We will sometimes find it helpful to create a quaternion with zero scalar part from a vector $r \in \mathbb{R}^3$. We denote this operation as,

$$ \hat{r} = H r \equiv \begin{bmatrix} 0 \\ I_3 \end{bmatrix} r. \quad (8) $$

Unit quaternions rotate a vector through the operation $\hat{r'} = q \otimes \hat{r} \otimes q^*$. This can be equivalently expressed using matrix multiplication as

$$ r' = H^T L(q)R(q)^T H r = A(q)r, \quad (9) $$

where $A(q)$ is the rotation matrix in terms of the elements of the quaternion $[27]$.

B. Rigid Body Dynamics

For clarity, we will restrict our attention to rigid bodies moving freely in 3D space. That is, we consider systems with dynamics of the following form:

$$ x = \begin{bmatrix} r \\ q \\ v \\ \omega \end{bmatrix}, \quad \dot{x} = \begin{bmatrix} \frac{1}{2} q \otimes \dot{\omega} = \frac{1}{2} L(q) H \omega \\ \frac{1}{m} W_F(x, u) \\ J^{-1}(B_T(x, u) - \omega \times J \omega) \end{bmatrix} \quad (10) $$

where $x$ and $u$ are the state and control vectors, $r \in \mathbb{R}^3$ is the position, $q \in \mathbb{S}^3$ is the attitude, $v \in \mathbb{R}^3$ is the linear velocity, and $\omega \in \mathbb{R}^3$ is the angular velocity. $m \in \mathbb{R}$ is the mass, $J \in \mathbb{R}^{3 \times 3}$ is the inertia matrix, $W_F(x, u) \in \mathbb{R}^3$ are the forces in the world frame, and $B_T(x, u)$ are the moments in the body frame.

III. Quaternion Differential Calculus

We now present a simple but powerful method for taking derivatives of functions involving quaternions based on the notation and linear algebraic operations outlined in Sec. II-A.

Derivatives consider the effect an infinitesimal perturbation to the input has on an infinitesimal perturbation to the output. For vector spaces, the composition of the perturbation with the nominal value is simple addition and the infinitesimal perturbation lives in the same space as the original vector. For unit quaternions, however, neither of these are true; instead, they compose according to (11), and infinitesimal unit quaternions are (to first order) confined to a 3-dimensional plane tangent to $\mathbb{S}^3$ (see Fig. 1).

The fact that differential unit quaternions are three-dimensional should make intuitive sense: Rotations are inherently three-dimensional and differential rotations should live in the same space as angular velocities, i.e. $\mathbb{R}^3$.

There are many possible three-parameter representations for small rotations in the literature. Many authors use the exponential map $[8, 12, 18, 21, 22, 28, 29]$, while others have used the Cayley map (also known as Rodrigues parameters) $[16, 17]$. Modified Rodrigues Parameters (MRPs) $[80]$, or the vector part of the quaternion $[8]$. We choose Rodrigues parameters $[24]$ because they are computationally efficient and do not inherit the sign ambiguity associated with unit quaternions. The mapping between a vector of Rodrigues parameters $\phi \in \mathbb{R}^3$ and a unit quaternion $q$ is known as the Cayley map:

$$ q = \varphi(\phi) = \frac{1}{\sqrt{1 + ||\phi||^2}} \begin{bmatrix} 1 \\ \phi \end{bmatrix}. \quad (11) $$

Fig. 1. When linearizing about a point $q$ on an sphere $\mathbb{S}^{n-1}$ in $n$-dimensional space, the tangent space $T$ is a plane living in $\mathbb{R}^{n-1}$, illustrated here with $n = 3$. Therefore, when linearizing about a unit quaternion $q \in \mathbb{S}^3$, the space of differential rotations lives in $\mathbb{R}^3$. 


We will also make use of the inverse Cayley map:

\[ \phi = \varphi^{-1}(q) = \frac{q_0}{q_s}. \]  

(12)

A. Jacobian of Vector-Valued Functions

When taking derivatives with respect to quaternions, we must take into account both the composition rule and the nonlinear mapping between the space of unit quaternions and our chosen three-parameter error representation.

Let \( \phi \in \mathbb{R}^3 \) be a differential rotation applied to a function with quaternion inputs \( y = h(q) : \mathbb{S}^3 \rightarrow \mathbb{R}^p \), such that

\[ y + \delta y = h(L(q)\varphi(\phi)) \approx h(q) + \nabla h(q)\phi. \]  

(13)

Note that we chose to represent \( \phi \) in the body frame, consistent with the standard definition of angular velocity, and therefore it is applied to \( q \) through right (rather than left) multiplication. We can calculate the Jacobian \( \nabla h(q) \in \mathbb{R}^{p \times 3} \) by differentiating (13) with respect to \( \phi \), evaluated at \( \phi = 0 \):

\[ \nabla h(q) = \frac{\partial h}{\partial q} L(q)H := \frac{\partial h}{\partial q} G(q) = \frac{\partial h}{\partial q} \left[ -q_o^T \left[ q_s I_3 + [q_s \times] \right] \right] \]  

(14)

where \( G(q) \in \mathbb{R}^{4 \times 3} \) is the attitude Jacobian, which essentially becomes a “conversion factor” allowing us to apply results from standard vector calculus to the space of unit quaternions. This form is particularly useful in practice since \( \partial h/\partial q \in \mathbb{R}^{p \times 4} \) can be obtained using finite differences or automatic differentiation. As an aside, although we have used Rodrigues parameters, \( G(q) \) is actually the same (up to a constant scalar factor) for any choice of three-parameter attitude representation.

B. Hessian of Scalar-Valued Functions

If the output of \( h \) is a scalar (\( p = 1 \)), then we can find its Hessian by taking the Jacobian of (14) with respect to \( \phi \) using the product rule, again evaluated at \( \phi = 0 \):

\[ \nabla^2 h(q) = G(q)^T \frac{\partial^2 h}{\partial q^2} G(q) - I_3 \frac{\partial h}{\partial q}, \]  

(15)

where the second term comes from the second derivative of \( \varphi(\phi) \). Similar to \( G(q) \), this expression is the same (up to a constant scalar factor) for any choice of three-parameter attitude representation.

C. Jacobian of Quaternion-Valued Functions

We now consider the case of a function that maps unit quaternions to unit quaternions, \( q' = f(q) : \mathbb{S}^3 \rightarrow \mathbb{S}^3 \). Here we must also consider the non-trivial effect of a differential rotation applied to the output, i.e.:

\[ L(q')\varphi(\phi') = f(L(q)\varphi(\phi)). \]  

(16)

Solving (16) for \( \phi' \) we find,

\[ \phi' = \varphi^{-1}(L(q')^T f(L(q)\varphi(\phi))) \approx \nabla f(q) \phi. \]  

(17)

Finally, the desired Jacobian is obtained by taking the derivative of (17) with respect to \( \phi \):

\[ \nabla f(q) = H^T L(q')^T \frac{\partial f}{\partial q} L(q)H = G(q)^T \frac{\partial f}{\partial q} G(q). \]  

(18)

Once again, (18) holds (up to a constant) for any three-parameter attitude representation.

IV. Modifying Newton’s Method

Newton’s method uses derivative information about a function to iteratively approximate its roots. For unconstrained systems, this method is highly effective, and can exhibit quadratic convergence rates. For constrained systems, the updates can be projected back onto the feasible set at each iteration, but without the same convergence guarantees.

In this section, we will leverage the quaternion calculus results introduced in the previous section to modify Newton’s method so that it implicitly accounts for the quaternion unit-norm constraint. Unlike the projection approach, this modified form of Newton’s method retains the fast convergence rates associated with the unconstrained method. We will demonstrate this behavior on Wahba’s Problem, a least-squares attitude estimation problem \([25],[26]\).
Algorithm 1 Multiplicative Gauss-Newton Method

1: \( k = 0 \)
2: while \( \|\phi\| > \) tolerance do
3: \( \nabla r = \frac{\partial r(q_k)}{\partial q} G(q_k) \) \( \triangleright \) Compute Jacobian
4: \( \phi = -\left(\nabla r^T \nabla r\right)^{-1} \nabla r^T r(q_k) \) \( \triangleright \) Compute update step
5: \( q_{k+1} = L(q_k)\varphi(\phi) \) \( \triangleright \) Apply update step
6: \( k = k + 1 \)
7: end while

B. Results

Figure 3 compares the multiplicative Gauss-Newton method with a naïve Newton’s method in which the quaternion is simply projected back onto the unit sphere at every iteration. The naïve method makes progress initially, but quickly stalls. By correctly handling the group structure of unit quaternions, the multiplicative method is able to maintain the fast convergence rates typical of Newton’s method. By comparing our method with the global solution obtained from a singular-value decomposition [31], we see that our method recovers the globally optimal solution within a small number of iterations.

V. Trajectory Optimization for Rigid Bodies

Here we outline the modifications to the ALTRO solver to solve trajectory optimization problems for rigid bodies, which extends easily to arbitrary systems whose state is in \( \mathbb{R}^n \times \mathbb{S}^2 \). We consider trajectory optimization problems of the form,

\[
\begin{align*}
\text{minimize} & \quad \ell_f(x_N) + \sum_{k=1}^{N-1} \ell_k(x_k, u_k) \\
\text{subject to} & \quad x_{k+1} = f(x_k, u_k), \quad g_k(x_k, u_k) \leq 0, \quad h_k(x_k, u_k) = 0,
\end{align*}
\]

where \( x \) and \( u \) are the state and control vectors as described in Sec. II-B and \( f \) are the dynamics as defined in [10], \( \ell_k \) is a general nonlinear cost function at a single time step, \( N \) is the number of time steps, and \( g_k, h_k \) are general nonlinear inequality and equality constraints.

ALTRO combines techniques from both differential dynamic programming (DDP) and direct transcription methods to achieve high performance on challenging constrained nonlinear trajectory optimization problems. Like most methods for nonlinear optimization, ALTRO iteratively approximates the nonlinear functions \( f, \ell, g, \) and \( h \) with their first or second-order Taylor series expansions. Leveraging the methods from Sec. III, we adapt the algorithm to optimize directly on the error state \( \delta x \in \mathbb{R}^{12} \):}

\[
\delta x_{k+1} = A_k \delta x_k + B_k \delta u_k,
\]

where

\[
A_k = E(\bar{x}_{k+1})^T \frac{\partial f}{\partial x}|_{\bar{x}_k, \bar{u}_k} E(\bar{x}_k), \quad B_k = E(\bar{x}_{k+1})^T \frac{\partial f}{\partial u}|_{\bar{x}_k, \bar{u}_k},
\]

and \( E(x_k) \in \mathbb{R}^{13 \times 12} \) is the error-state Jacobian:

\[
E(x) = \begin{bmatrix} I_3 & G(q) & I_3 \\ \end{bmatrix}.
\]

By applying (14) and (15) to the nonlinear cost functions \( \ell \) and (18) to the nonlinear constraint functions \( g \) and \( h \), we can calculate the second-order expansion of the cost function:

\[
\delta \ell(x, u) \approx \frac{1}{2} \delta x^T \ell_{xx} \delta x + \frac{1}{2} \delta u^T \ell_{uu} \delta u + \delta u^T \ell_{ux} \delta x + \ell_u^T \delta x^T + \ell_u^T \delta u.
\]

With these results, we can apply standard Newton and quasi-Newton techniques along the lines of Section IV. We can also calculate a second-order expansion of the “action-value function” \( Q(x, u) \) needed in DDP and LQR-based methods,

\[
\begin{align*}
Q_{xx} &= \ell_{xx} + A_k^T P_{k+1} A_k \\
Q_{uu} &= \ell_{uu} + B_k^T P_{k+1} B_k \\
Q_{ux} &= \ell_{ux} + B_k^T P_{k+1} A_k \\
Q_x &= \ell_x + A_k^T p_{k+1} \\
Q_u &= \ell_u + B_k^T p_{k+1},
\end{align*}
\]

from which we can calculate the quadratic expansion of the cost-to-go \( P_k \in \mathbb{R}^{12 \times 12}, p_k \in \mathbb{R}^{12} \), and optimal linear feedback gains \( K_k \in \mathbb{R}^{n \times 12} \) and feed-forward corrections.
$d_k \in \mathbb{R}^m$ by starting at the terminal state and performing a backward Riccati recursion as usual \cite{22, 33}.

During the “forward rollout” of these methods, the dynamics are simulated forward in time using updated control inputs:

$$u_k = \ddot{u}_k - d_k - K_k \delta x_k.$$  \hspace{1cm} (35)

where $\ddot{u}_k$ is the control value from the previous iteration, and $\delta x$ is computed using \cite{23}. For more details on the AL TRO algorithm, we refer the reader to \cite{32}.

A. Quaternion Cost Functions

In addition to the straight-forward modifications to the AL TRO algorithm itself, some care must be taken in designing cost functions that are well-suited to unit quaternions. We frequently minimize costs that penalize the geodesic distance between two unit quaternions \cite{34}, to work well in practice:

$$J_{\text{geo}} = (1 - |q^T g|).$$  \hspace{1cm} (36)

Its gradient and Hessian are,

$$\nabla J_{\text{geo}} = -\text{sign}(q^T g)q^T G(q)$$  \hspace{1cm} (37)

$$\nabla^2 J_{\text{geo}} = \text{sign}(q^T g)I_{3 \times 3} q.$$  \hspace{1cm} (38)

where sign denotes the signum function. This cost function is particularly useful for rotations since it eliminates the ambiguity arising from the quaternion double-cover of $SO(3)$.

VI. Experiments

In this section we present several trajectory optimization problems for systems that undergo large changes in attitude: an airplane barrel roll, a quadrotor flip, and a satellite with flexible solar panels that must slew to a new orientation while avoiding a keep-out zone. All problems are run using AL TRO, first without any of the modifications presented in the current paper, analogous to the naive Newton’s method in section IV, and labeled “naive”, and then using the modifications listed in Sec. V, and the geodesic cost function described in Sec. VA, labeled “modified”. All cost functions are of the following form:

$$\ell_{\text{naive}}(x, u, \bar{x}, Q, R) = \frac{1}{2} (x - \bar{x})^T Q (x - \bar{x}) + \frac{1}{2} u^T R u$$

$$\ell_{\text{modified}}(x, u, \bar{x}, Q, R) = \ell_{\text{naive}}(x, u, \bar{x}, \bar{Q}, R) + w (1 \pm \bar{q}^T q)$$  \hspace{1cm} (39)

$$w = \text{diag}(Q_r, Q_\theta, Q_v, Q_\omega),$$

where $\bar{x}$ is the reference state and $\bar{Q} = \text{diag}(Q_r, Q_\theta, Q_v, Q_\omega)$, with $Q_r, Q_\theta, Q_v, Q_\omega$ being the weights of $Q$ for position, and linear and angular velocity, respectively.

Timing results are summarized in Table II. All experiments are solved to a constraint satisfaction tolerance of $10^{-5}$ and discretized with a 4th order Runge-Kutta integrator. The results were run on a laptop computer with a 2.8 GHz i7-1165G7 processor with 16 GB of RAM. Code for all experiments is available on GitHub

A. Airplane Barrel Roll

A 360 degree barrel roll trajectory for a fixed-wing airplane was optimized. The airplane’s dynamics model consists of the a simple rigid body as defined in Section IV-B with forces and torques due to lift and drag fit from wind tunnel data \cite{34}. The airplane was tasked to do a barrel roll by constraining the terminal state to upside-down (see Fig. 3). To mitigate issues with integration error and drift in the magnitude of the quaternion, the following constraint function was used to enforce a terminal orientation of $\dot{q}$:

$$\frac{\dot{q}_v}{||q||} - q_v \text{sign} \left( \frac{\dot{q}^T q}{||q||} \right) = 0$$  \hspace{1cm} (41)

The solver was initialized with level flight trim conditions. The convergence of the different versions of AL TRO is compared in Fig. 4. As expected, the modified version achieves better convergence and faster solve times compared to the naive version since the expansions being provided to the algorithm more accurately capture the relationship between the attitude state and the goal and constraints. For this relatively simple problem, we gained a modest 31% improvement in runtime, despite the additional matrix multiplications when calculating the cost and constraint expansions.

B. Quadrotor Flip

A 360 degree flip trajectory for a quadrotor was optimized with dynamics adapted from \cite{33}. To encourage
the flip, we specified a “waypoint” cost function of the following form:

$$\sum_{k \in N} \ell(x_k, u_k, \hat{x}, \hat{Q}, R) + \sum_{k \in W} \ell(x_k, u_k, \bar{x}_k, Q_w, R)$$

(42)

where $\hat{x}$, $\hat{Q}$ are the nominal state and state weight matrix, $Q_w$ is the weight matrix for the waypoints, and $W = \{20, 45, 51, 55, 75, 101\}$, $N = \{1 : 101\} \setminus W$. Four intermediary “waypoints” were used to encourage the quadrotor to reach angles of 90°, 180°, 270°, and 360° around an approximately circular arc. The last waypoint was used to encourage the quadrotor to move towards the final goal, and the first kept it above the floor before starting the loop. The solver was provided a dynamically infeasible initial trajectory that linearly interpolates between the initial and final states, rotating the quadrotor around the x-axis a full 360°.

Figure 5 shows snapshots of the trajectory as generated using ALTRO. To compare the convergence properties of the two methods, the optimal state and control trajectories were perturbed with random Gaussian white noise with a mean of 1 for position, linear velocity, and angular velocity, 0.1 for the controls, and 145 degrees for the orientation (about a random axis). As shown in Fig. 6, the modified method converges more reliably than the naïve method. It is also worth noting that this problem could not be solved using any three-parameter attitude representation, since it passes through the singularities at 90°, 180°, and 360° associated with Euler angles, Rodrigues parameters, and Modified Rodrigues Parameters, respectively.

C. Satellite Attitude Keep-Out

A spacecraft with flexible appendages was tasked to perform a 150 degree slew maneuver while ensuring that a body-mounted camera did not point within 40 degrees of a “keep-out zone” around the sun vector. The spacecraft dynamics are presented in detail in [36], and are based on equation (10) with the addition of six states to account for three flexible modes. Control torques are generated by four reaction wheels. A quadratic cost function penalizes error from the desired final attitude as well as displacement of the flexible modes. We enforce the camera keep-out zone with the following constraint,

$$(W_{r_{sun}})^T (W_A(q)^B B_{r_{cam}}) \leq \cos(40°),$$

(43)

where $B_{r_{cam}}$ is the camera line-of-sight unit vector in the body frame and $W_{r_{sun}}$ is the unit vector pointing to the sun in the world frame. The attitudes that satisfy this constraint comprise a non-convex set, with the constraint itself being nonlinear in $q$.

ALTRO is able to converge to a locally optimal trajectory for this problem without an initial guess (all controls were initialized to zero). The resulting attitude trajectory is depicted in Fig. 7. Without enforcing the camera constraint, the trajectory passes through the keep-out zone. As noted in Table I, the quaternion modifications did not result in a significant improvement over the naïve implementation of ALTRO for this problem, indicating that the computational benefits are problem dependent. We hypothesize that the more dynamic behaviors in the other examples benefit more from the quaternion modifications than the relatively slow-moving spacecraft.
state estimation for spacecraft with sparse measurements.

Acknowledgements

This work was supported by a NASA Early Career Faculty Award (Grant Number 80NSSC18K1503). This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded through JPL’s Strategic University Research Partnerships (SURP) program. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1656518. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References


[22] M. Watterson, S. Liu, K. Sun, T. Smith, and V. Kumar, “Trajectory optimization on manifolds with applications to SO(3) and $R^3 \times S^2$,” in Robotics: Science and Systems (RSS), Pittsburgh, PA, USA, Jun. 2018.


[31] F. L. Markley and D. Mortari, “How to estimate attitude from vector observations,” in Proceedings of...


