JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 34, No. 4, July–August 2011

Time-Optimal Reorientation of a Spacecraft Using an Inverse Dynamics Optimization Method

George A. Boyarko,[*] Marcello Romano,[†] and Oleg A. Yakimenko[‡]
Naval Postgraduate School, Monterey, California 93943-5107

DOI: 10.2514/1.49449

This paper proposes a rapid attitude trajectory generation method for satellite reorientation, which is suitable for both rest-to-rest and track-to-track maneuvers. The problem is first formulated and solved using academic software readily used for generating optimal guidance trajectories offline. Then, the problem is reformulated using a Bezier polynomial parametrization of the quaternion trajectory, which naturally satisfies the unit quaternion norm constraint. The parameters of the quaternion trajectory polynomial and of the speed profile are varied to arrive at a quasi-optimal solution that is both feasible and exactly matches the endpoint conditions specified in the problem. The reduction in the number of varied parameters due to the predetermined structure of the trajectory leads to a fast computational speed as well as a solution that satisfies the end constraints at each iteration. The paper includes numerical simulation results obtained for several cases.

I. Introduction

THE optimal satellite reorientation problem is of general interest to many in the field of aerospace engineering. The available literature on this subject is extensive (see for instance [1–5]). Many civilian and military space missions need to have agile attitude maneuver capability. For instance, TacSat-3 was intended to demonstrate responsive delivery of information to operational users [6]. Because of the satellite's low Earth orbit, the timeline for tasking, slewing and disseminating data is greatly reduced. Other challenges include the fact that the tasking can be modified at any moment up to a short period of time after the ground station starts uploading the tasking, as well as the idea that TacSat must autonomously slew to the target, collect and process the data, and then downlink the data directly to the customer who is not collocated with the ground station. Finally, current real-time feedback controls are not optimized for minimum time [2]. The preceding challenges lend themselves to the need for the ability to rapidly generate feasible trajectories that are optimized for minimum time.

The goal of the present paper is to provide a method to determine a feasible attitude trajectory solution that meets endpoint requirements and dynamic constraints while performing a good overall maneuver relative to a given performance index. Also, the method should work for any boundary conditions including nonrest to nonrest (track-to-track) maneuvers. The major requirement is that the method must provide a feasible real-time solution as opposed to offline computations even if it requires some sacrifices in terms of optimality.

The existing techniques include using so-called pseudospectral methods. These methods can provide an incredibly accurate solution to an optimal control problem, but may require extended periods of time to converge, or, in the case of a smaller number of nodes, converge to a suboptimal solution [7–9]. In addition, the optimal solution does not have an analytical representation, which may pose problems when trying to implement it using a feedforward scheme of suggested control commands [7–9]. If the numerical solution is afterwards approximated with some analytical function and/or the controls are smoothed, it may lose some optimality and the smoothed solution may not satisfy the boundary conditions.

The authors pursue another approach exploiting the general idea of the direct optimization methods of calculus of variations together with an inverse dynamics approach. In particular, polynomials are used as basis functions to generate trajectories in quaternion space that can be traversed according to a computed analytic speed profile. Instead of expressing trajectories in the time domain, an abstract argument is introduced: this allows the trajectory to be formulated so as the states and their derivatives are intrinsically and exactly satisfied at the endpoints. This results in decoupling space and time, allowing the speed over the trajectory to be varied to satisfy problem constraints. The resulting quasi-optimal trajectory solution can be generated (and updated) rapidly because of the reduction in the number of varied parameters due to the restriction on the trajectory structure by specifying a polynomial basis. Although this method lacks some flexibility due to the predefined structure of the solution, it provides a feasible solution that satisfies the endpoint constraints at every iteration, even when the initial conditions change due to disturbances or delays. Specific applications include scenarios where derivative conditions on beginning and ending states need to be met, such as tracking missions, docking missions, and other missions where a simple eigenaxis slew may be unacceptable.

It should be noted that the use of inverse dynamics to optimize the rotational motion of a satellite has been evaluated by other authors as well. In [10,11] Euler angles are used, which suffer from well-known kinematic singularities, and no attempt to decouple the time and space domains is made. Furthermore, in [12]a linearized blending of two spins in quaternion space is used.

The present paper proposes a novel approach resulting in a robust and fast computational technique. The paper is organized as follows. Section II describes the problem to be solved as well as the dynamic and kinematic models. Section III demonstrates obtaining a solution using an academic solver employing a pseudospectral method. Section IV introduces the Inverse Dynamics in the Virtual Domain (IDVD) approach and develops a complete computational scheme using Bezier curves to parametrize quaternion trajectories, followed by Sec. V presenting sample solutions. Finally, Sec. VI compares the results of optimization obtained with both methods.

II. Spacecraft Model and Attitude Trajectory Optimization Problem

The rotational dynamics of the spacecraft can be described by Euler's rotational equations of motion. Written in the body-fixed principal axes this results in the vector equation [13,14]

Received 18 February 2010; revision received 20 September 2010; accepted for publication 20 September 2010.

[*]Department of Mechanical and Aerospace Engineering; gaboyark@nps.edu. Member AIAA.

[†]Department of Mechanical and Aerospace Engineering; mromano@nps.edu. Associate Fellow AIAA.

[‡]Department of Mechanical and Aerospace Engineering; oayakime@nps.edu. Associate Fellow AIAA.

$$\mathbf{I}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} = \mathbf{T} \tag{1}$$

which expands into the scalar equations:

$$\dot{\omega}_x \triangleq \alpha_x = \frac{(I_{yy} - I_{zz})\omega_y\omega_z + T_x}{I_{xx}}$$
$$\dot{\omega}_y \triangleq \alpha_y = \frac{(I_{zz} - I_{xx})\omega_x\omega_z + T_y}{I_{yy}}$$
$$\dot{\omega}_z \triangleq \alpha_z = \frac{(I_{xx} - I_{yy})\omega_y\omega_x + T_z}{I_{zz}} \tag{2}$$

In Eqs. (2) and (3) $\mathbf{I} = \mathrm{diag}([I_{xx}, I_{yy}, I_{zz}])$ is the inertia matrix (along the principal axes), $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ is the vector of angular velocities, and $\mathbf{T} = [T_x, T_y, T_z]^T$ is the vector of torques (bounded controls).

In turn, rotational kinematics can be described using quaternion $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$ as: [13,14]

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\Omega} \tag{3}$$

where the symbol $\otimes$ is used to indicate the quaternion multiplication operation, and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z, 0]^T$.

The problem in question is to find the slew trajectory (quaternion time history) for a satellite subject to specific constraints that minimizes the time to complete the maneuver t_f. This is expressed as minimizing the performance index:

$$J = \int_0^{t_f} \mathrm{d}t \tag{4}$$

while reorienting a satellite from the initial conditions $\boldsymbol{\omega}_0$, $\mathbf{q}_0$ to the final conditions $\boldsymbol{\omega}_f$, $\mathbf{q}_f$ for a system (2) and (3), subject to constraints on controls

$$\mathbf{T}_{\min} \le \mathbf{T} \le \mathbf{T}_{\max} \tag{5}$$

Bilimoria and Wie [3] formulated this problem for a rest-to-rest maneuver and presented the solution obtained by using an indirect method. They showed that, in general, for the case of a symmetric body with bounds on each torque component, it results in a non-eigenaxis maneuver. In addition to that, the following section presents a more general solution obtained offline to be used along with that of [3] as a reference for the proposed online solution obtained using a direct method exploiting the inverse dynamics of Eqs. (2) and (3).

To this end, Table 1 shows two different test cases examined in this paper. Test Case 1, representing an idealized rather than real spacecraft, was taken directly from [3], while another was chosen to illustrate a more general scenario, when a spacecraft is not necessarily symmetric.

In terms of the endpoint conditions the paper considers two basic scenarios assuming $\phi = 90°$ and $\phi = 180°$ slew maneuvers about the z-axis (so that $\mathbf{q}_0 = [0, 0, 0, 1]^T$ and $\mathbf{q}_f = [0, 0, \sin\frac{1}{2}\phi, \cos\frac{1}{2}\phi]^T$) with zero and nonzero normalized body rates at the endpoints ($\boldsymbol{\omega}_0 = \boldsymbol{\omega}_f = \mathbf{0}_{3\times1}$ and $\boldsymbol{\omega}_0 = -\boldsymbol{\omega}_f \neq \mathbf{0}_{3\times1}$. Finally, for the normalized states, the constraints (5) take the form $-\mathbf{1}_{3\times1} \le \mathbf{T} \le \mathbf{1}_{3\times1}$.

All computations were carried out on a 2.33 GHz Dell Precision M90 desktop computer with an Intel T7600 processor and 1 Gb of RAM. As the optimization engine, SNOPT [15], the Gauss Pseudospectral Optimization Software (GPOPS) [16], and MATLAB [17] `fmincon` function (IDVD) were used. For the sake of completeness

Table 1 Description of the test cases

Case	Normalized inertia matrix
Case 1	$\mathbf{I} = \mathrm{diag}([1, 1, 1])$
Case 2	$\mathbf{I} = \mathrm{diag}([3, 1, 2])$

and repeatability it should also be noted that while the IDVD solution was obtained in the purely interpretative environment of MATLAB, SNOPT used a library of optimized executable files and, therefore, was much more computationally effective.

III. Solving the Problem Using the Gauss Pseudospectral Method

Before proceeding with the proposed online solution, consider the problem formulated in the previous section using one of the prominent pseudospectral (collocation) methods. The goal is to have some reference solutions and also to see if the solutions obtained using this approach can be reliably used for online optimization.

It should be noted that the Bilimoria and Wie [3] solution has been previously reobtained [18] by using the commercial software package DIDO [19].

The GPOPS package was chosen for the present work because of its open source nature and free availability [16].

Figures 1–8 show the minimum-time solutions for a 180° slew of a satellite obtained by applying GPOPS. Specifically, Figs. 1 and 2 present time histories of all states and controls for the solution that involves 100 nodes, which results in $(100 - 2) \times 10 = 980$ variable parameters. Figure 3 depicts the three-dimensional (3-D) representation of the solution in inertial space, clearly showing that it is not an eigenaxis maneuver, with an inclination of the z_b axis during rotation in the $x_b y_b$ direction. This solution compares with the solution presented in [3] fairly well. The final calculated maneuver time, t_f, was found to be 3.243 s. However, it took almost two hours of CPU time to obtain this solution. Another observation is that because of the nature of the system (2), the optimal control has a bang–bang structure (Fig. 2). That results in the maximum magnitude of the angular acceleration at the boundary points (for Case 1 angular accelerations are simply equal to the corresponding controls). Also, if we are to update a trajectory while the satellite performs this rotation (to accommodate possible disturbances and unmodeled dynamics), it would cause discontinuities of angular acceleration (sudden jumps in controls). Increasing the order of the system to account for the boundary conditions on angular accelerations will

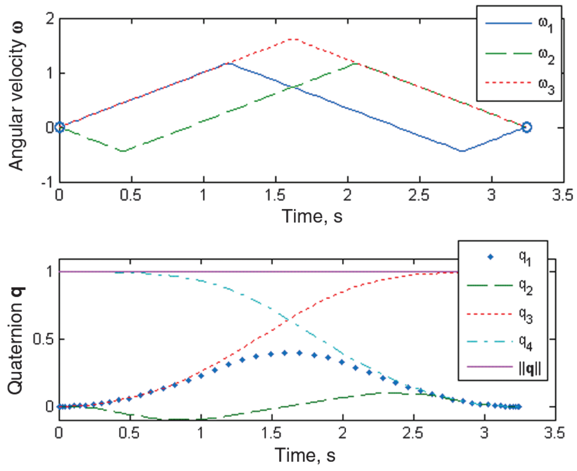

Fig. 1 Time histories of the states for Case 1 (100 Nodes GPOPS Solution).

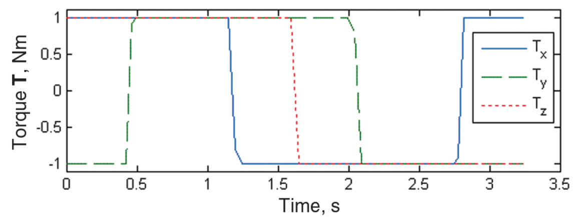

Fig. 2 Time history of the controlling torques for Case 1 (100 Nodes GPOPS Solution).

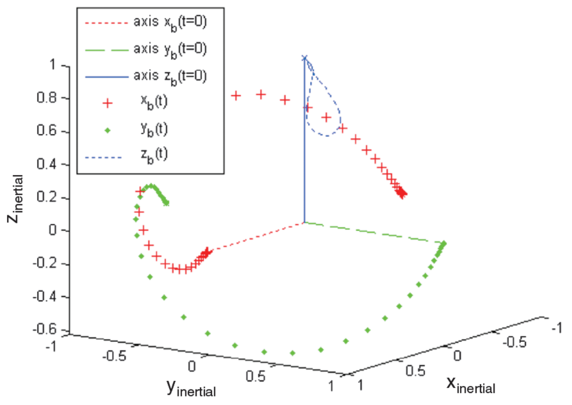

Fig. 3 The 3D representation of the solution for Case 1 (100 Nodes GPOPS Solution).

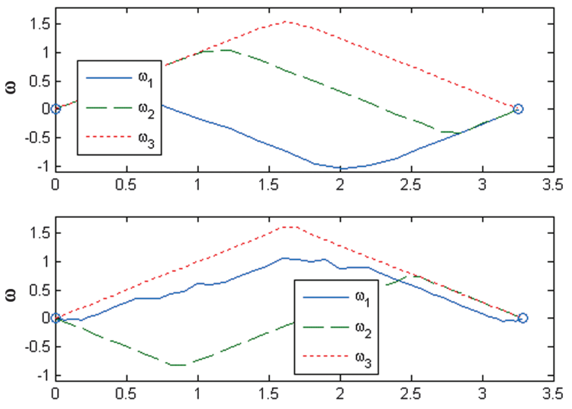

Fig. 4 Case 1 comparison of time histories of angular velocity components obtained for 25 nodes (top) and 50 nodes (bottom) for GPOPS solution.

obviously cause a slight degradation of the performance index and a further increase of the required CPU time to obtain a solution. Hence, although in this case GPOPS does produce a valid solution, it is not practical and cannot be used for online computations.

As pointed out in [20], reducing the number of nodes may lead to a more robust (in terms of computational time) result, therefore an attempt was made to obtain a solution of the same problem using a smaller number of nodes. These GPOPS solutions are shown in Figs. 4–6.

It results that, if the number of nodes is reduced, GPOPS converges to different solutions (it is indeed a well-known fact that different solutions exist [4]). To this end, Fig. 4 shows time histories of the angular velocity components for the 25 and 50-node solution (involving 230 and 480 varied parameters, respectively). Obviously, they are different from the 100-node solution in Fig. 2. While a 25-node solution is simply symmetrical with respect to the 100-node

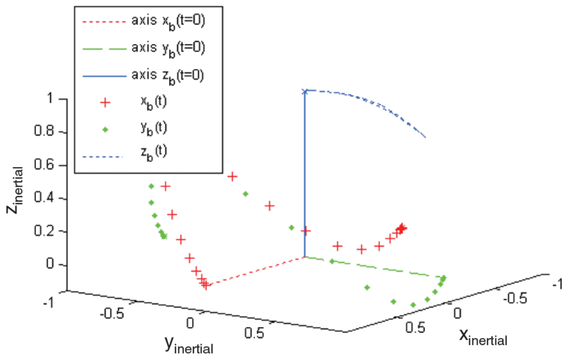

Fig. 5 The 3D representation of the 25-node GPOPS solution for Case 1.

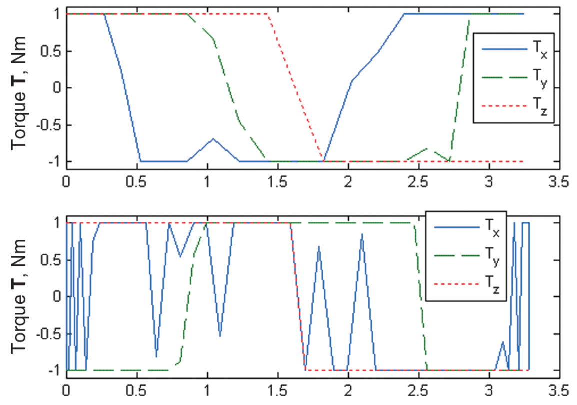

Fig. 6 Case 1 comparison of time histories of torques, obtained for 25 nodes (top) and 50 nodes (bottom) for GPOPS solution.

solution, as can be seen by comparing Figs. 3 and 5 showing an inclination of the z_b axis during rotation in the $-x_b y_b$ direction, and represents another equally optimal solution out of possible four solutions, a 50-node solution appears not to be valid (optimal) at all (Fig. 6).

As expected, decreasing the number of nodes leads to a substantial decrease in the computational time, but as shown above the method could produce a nonvalid solution. Also, even if it produces valid time histories for control torques, it may not be trackable by the inner-loop controllers. These give two more reasons why the solutions obtained using pseudospectral methods may not be used in a real-time feedforward control scheme.

For Case 2, the nonsymmetrical inertia with the bounds on individual control torques, the solution is slightly different (Figs. 7 and 8). The overall characteristic of this and other solutions involving different sets of the boundary conditions will be presented in Sec. V, but the general tendency is the same: it requires at least a hundred nodes to produce a valid and feasible offline solution. Yet, GPOPS presents a good and easy-to-use tool to produce reference trajectories that can be used for comparison with solutions obtained using other approaches, like the Inverse Dynamics in the Virtual Domain that will be introduced here below.

IV. Inverse Dynamics in the Virtual Domain Approach

One of the two main ideas of the Inverse Dynamics in the Virtual Domain (IDVD) method is exploiting the differential flatness property of the equations of motion [20–23]. In the considered problem, this relates to the fact that all the state and control variables can be expressed as functions of the output variable or its time derivatives, which in this case is the quaternion itself:

$$\boldsymbol{\omega} = f_1(\mathbf{q}, \dot{\mathbf{q}}), \qquad \mathbf{T} = f_2(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{6}$$

Another aspect of IDVD involves handling computations in the virtual domain, allowing space and time decoupling. As consequence, a trajectory can be computed while manipulating the speed at which this trajectory is followed.

The following presents a novel parameterization for the output variable, i.e., the quaternion, and develops a step-by-step computational routine (see also the pseudo code in the Appendix).

A. Quaternion Trajectory Parameterization

To parameterize the problem, the output trajectory structure is imposed by using some combination of basis functions. The standard approach would be to choose some combination of polynomials or trigonometric functions for the output variables [20–22]. While this may be straightforward when dealing with state variables in translational space, it may become more challenging when dealing with attitude representation.

Even though using quaternions is an advantageous method to express attitude because of the absence of singularities, choosing

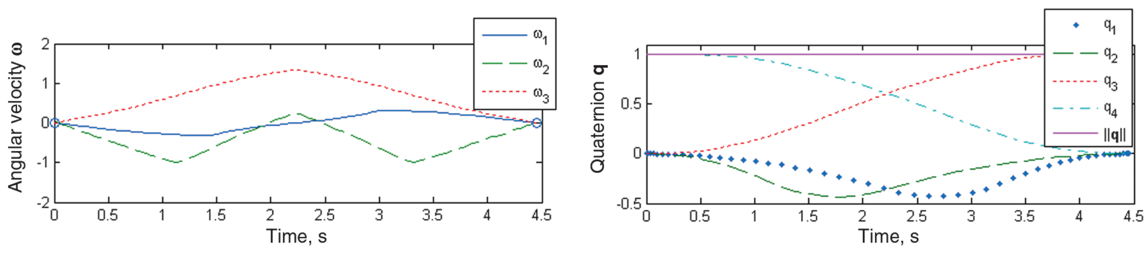

Fig. 7 Time histories of the states for Case 2 (100 Nodes GPOPS Solution).

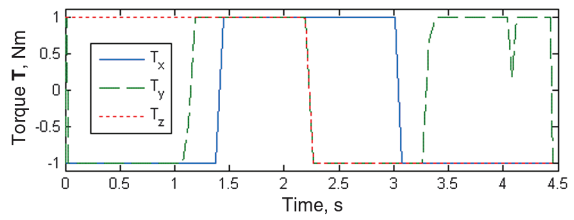

Fig. 8 Time history of the controlling torques for Case 2 (100 Nodes GPOPS Solution).

basis functions becomes more challenging because a nonlinear unit norm condition needs to be preserved across the quaternion history [23]. For this reason, a specific expression for the quaternion was chosen according to the work of Kim et al. [24]. In particular, the quaternion time history is expressed as a product of exponential maps containing a constant parameter multiplied by a Bezier polynomial of degree n, as follows:

$$\mathbf{q}(\tau) = \mathbf{q}_0 \otimes \prod_{i=1}^{n} \exp(\tilde{\boldsymbol{\omega}}_I \tilde{\beta}_{i,n}(\tau)) \tag{7}$$

where the symbol Π is used to indicate the quaternion product of sequence and

$$\tilde{\beta}_{i,n}(\tau) = \sum_{j=i}^{n} \binom{n}{j} (1-\tau)^{n-j} \tau^j, \quad \text{and} \quad \tilde{\boldsymbol{\omega}}_i = \log(\tilde{\mathbf{q}}_{i-1}^{-1} \otimes \tilde{\mathbf{q}}_i) \tag{8}$$

where $\tilde{\mathbf{q}}_i$ are constant quaternions that act as control points, and the following definitions apply for the exponential and logarithmic maps [24]

$$\mathbf{q} = \exp(\mathbf{v}) = \begin{cases} \left[\sin\left(\frac{|\mathbf{v}|}{2}\right) \frac{\mathbf{v}}{|\mathbf{v}|} \quad \cos\left(\frac{|\mathbf{v}|}{2}\right) \right]^T & |\mathbf{v}| \neq 0 \\ [0 \quad 0 \quad 0 \quad 1]^T & |\mathbf{v}| = 0 \end{cases}$$

$$\mathbf{v} = \log(\mathbf{q}) \tag{9}$$

$$= \begin{cases} 2\left(\frac{\arccos(q_4)}{\sqrt{q_1^2+q_2^2+q_3^2}}\right)[q_1 \quad q_2 \quad q_3]^T & \sqrt{q_1^2+q_2^2+q_3^2} \neq 0 \\ [0 \quad 0 \quad 0] & \sqrt{q_1^2+q_2^2+q_3^2} = 0 \end{cases}$$

The exponential map in Eq. (9) can be interpreted as a mapping from the vector $\mathbf{v}$ into a unit quaternion, where the vector $\mathbf{v}$ is a vector having the direction of the Euler's axis and the magnitude equal to the Euler's angle of the resulting unit quaternion $\mathbf{q}$. The map becomes one to one if the domain of the exponential is limited so that $|\mathbf{v}| < 2\pi$ [24]. In particular, $\exp(\mathbf{v})$ gives the quaternion $\mathbf{q}$ corresponding to the orientation obtained, from the initial orientation $[0 \quad 0 \quad 0 \quad 1]^T$, by rotating of an angle $|\mathbf{v}|$ around the fixed direction $\frac{\mathbf{v}}{|\mathbf{v}|}$. This is also the direction of the angular velocity if the particular case of rotation about a fixed axis is considered. On the other hand, the logarithm map in Eq. (9), gives the vector having the direction of the Euler's axis and the magnitude equal to the Euler's angle of the orientation from $[0 \quad 0 \quad 0 \quad 1]^T$ to $\mathbf{q}$. In general, given any two unit quaternions, $\mathbf{q}_1$ and $\mathbf{q}_2$, it is possible to express an interpolating curve $\gamma_{q_1 q_2} \in S^3$, parametric in τ, which connects $\mathbf{q}_1$ to $\mathbf{q}_2$ along the geodesic on the four dimensional sphere (i.e., through an eigenaxis rotation). A general expression for this curve is $\gamma_{q_1 q_2} = \mathbf{q_1} \otimes \exp[f(\tau) \log(\mathbf{q}_1^{-1} \otimes \mathbf{q}_2)]$, with $0 \leq f(\tau) \leq 1$. If f $f(\tau) = \tau$, the curve gives a linear interpolation in the parameter τ. Equation (7) uses instead an interpolation based on Bezier polynomials.

In summary, by using Eq. (7), the quaternion $\mathbf{q}(\tau)$ is obtained by the quaternion product of $n+1$ quaternions ($\mathbf{q}_0, \exp(\tilde{\boldsymbol{\omega}}_1 \tilde{\beta}_{1,n}(\tau)), \ldots, \exp(\tilde{\boldsymbol{\omega}}_n \tilde{\beta}_{n,n}(\tau))$):, i.e., the orientation corresponding to $\mathbf{q}(\tau)$ is assumed as a sequence of n rotations (instantaneous in τ), starting from the orientation corresponding to $\mathbf{q}_0$, with the ith rotation being an eigenaxis rotation of amplitude ($\tilde{\beta}_{i,n}(\tau)|\tilde{\boldsymbol{\omega}}_i|$) about the axis $\frac{\tilde{\boldsymbol{\omega}}_i}{|\tilde{\boldsymbol{\omega}}_i|}$. Since $|\exp(\mathbf{v})| = 1 \ \forall \, \mathbf{v}$, as defined in Eq. (9), the resulting quaternion of Eq. (7) verifies the unit norm constraint for any τ.

It is important to note that in Eqs. (7) and (8) $\tau \in [0; 1]$ is an abstract argument that is used instead of time. This is a key point of the IDVD method and, in this particular case, allows to exploit interesting attributes of the Bezier polynomials and define properties at the boundaries.

For example, the analytic expressions of the fifth-order Bezier polynomials, $\tilde{\beta}_{i,5}(\tau)$, are as follows:

$$\begin{aligned} \tilde{\beta}_{1,5}(\tau) &= \tau^5 - 5\tau^4 + 10\tau^3 - 10\tau^2 + 5\tau \\ \tilde{\beta}_{2,5}(\tau) &= -4\tau^5 + 15\tau^4 - 20\tau^3 + 10\tau^2 \\ \tilde{\beta}_{3,5}(\tau) &= 6\tau^5 - 15\tau^4 + 10\tau^3 \\ \tilde{\beta}_{4,5}(\tau) &= -4\tau^5 + 5\tau^4, \qquad \tilde{\beta}_{5,5}(\tau) = \tau^5 \end{aligned} \tag{10}$$

These expressions have the favorable properties:

$$\tilde{\beta}_{i,5}(0) = 0 \quad \text{and} \quad \tilde{\beta}_{i,5}(1) = 1, \quad \text{for } i = 1, \ldots, 5 \tag{11}$$

$$\begin{aligned} \tilde{\beta}'_{1,5}(0) = \tilde{\beta}'_{5,5}(1) = 5, \qquad \tilde{\beta}'_{1,5}(1) = \tilde{\beta}'_{5,5}(0) = 0 \\ \tilde{\beta}'_{i,5}(0) = \tilde{\beta}'_{i,5}(1) = 0, \quad \text{for } i = 2, 3, 4 \end{aligned} \tag{12}$$

$$\begin{aligned} \tilde{\beta}''_{1,5}(0) = \tilde{\beta}''_{5,5}(1) = 20, \qquad \tilde{\beta}''_{1,5}(1) = \tilde{\beta}''_{5,5}(0) = 0 \\ \tilde{\beta}''_{2,5}(0) = \tilde{\beta}''_{4,5}(1) = -20, \qquad \tilde{\beta}''_{2,5}(1) = \tilde{\beta}''_{4,5}(0) = 0 \\ \tilde{\beta}''_{3,5}(0) = \tilde{\beta}''_{3,5}(1) = 0 \end{aligned} \tag{13}$$

which fix the value of the polynomials and its derivatives at the endpoints. The prevailing idea is that $\mathbf{q}(\tau = 0) = \tilde{\mathbf{q}}_0$ and $\mathbf{q}(\tau = 1) = \tilde{\mathbf{q}}_5$, where we can define

$$\tilde{\mathbf{q}}_0 \triangleq \mathbf{q}_0 \quad \text{and} \quad \tilde{\mathbf{q}}_5 \triangleq \mathbf{q}(t_f) = \mathbf{q}_f \tag{14}$$

The parametrization of the quaternion trajectory, given by Eq. (7), results in a convenient calculation of derivatives with respect to the virtual domain argument τ. The results for the first derivative for a third-order Bezier polynomial were presented in [24]. In general, the first-order derivative with respect to the argument τ is given by:

$$\mathbf{q}'(\tau)=\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\tau}(\tau)=\mathbf{q}_0\otimes\begin{bmatrix}\exp(\tilde{\boldsymbol{\omega}}_1\tilde{\beta}_{1,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_1\tilde{\beta}'_{1,n}(\tau)\otimes\exp(\tilde{\boldsymbol{\omega}}_2\tilde{\beta}_{2,n}(\tau))\otimes\ldots\otimes\exp(\tilde{\boldsymbol{\omega}}_n\tilde{\beta}_{n,n}(\tau))+\\ \exp(\tilde{\boldsymbol{\omega}}_1\tilde{\beta}_{1,n}(\tau))\otimes\exp(\tilde{\boldsymbol{\omega}}_2\tilde{\beta}_{2,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_2\tilde{\beta}'_{2,n}(\tau)\otimes\ldots\otimes\exp(\tilde{\boldsymbol{\omega}}_n\tilde{\beta}_{n,n}(\tau))+\ldots+\\ \exp(\tilde{\boldsymbol{\omega}}_1\tilde{\beta}_{1,n}(\tau))\otimes\exp(\tilde{\boldsymbol{\omega}}_2\tilde{\beta}_{2,n}(\tau))\otimes\ldots\otimes\exp(\tilde{\boldsymbol{\omega}}_n\tilde{\beta}_{n,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_n\tilde{\beta}'_{n,n}(\tau)\end{bmatrix}\tag{15}$$

where $\tilde{\boldsymbol{\Omega}}_i=[\tilde{\omega}_{ix},\tilde{\omega}_{iy},\tilde{\omega}_{iz},0]^T$. Note that collecting the terms in Eq. (15) into $\mathbf{q}'(\tau)=\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\tau}(\tau)=\mathbf{q}_0\otimes\sum_{j=1}^{n}(\tilde{\boldsymbol{\Omega}}_j\tilde{\beta}'_{j,n}(\tau))\otimes\prod_{i=1}^{n}\exp(\tilde{\boldsymbol{\omega}}_i\tilde{\beta}_{i,n}(\tau))$ would lead to an incorrect expression, since the quaternion product is not commutative.

Similarly, the second-order derivative is calculated starting from the expression of Eq. (15), by applying the chain rule as follows (only the derivative of the first element within the brackets of Eq. (15) is completed below):

$$\mathbf{q}''(\tau)=\frac{\mathrm{d}\mathbf{q}'}{\mathrm{d}\tau}(\tau)=\mathbf{q}_0\otimes\begin{bmatrix}\exp(\tilde{\boldsymbol{\omega}}_1\tilde{\beta}_{1,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_1\tilde{\beta}'_{1,n}(\tau)\otimes\tilde{\boldsymbol{\Omega}}_1\tilde{\beta}'_{1,n}(\tau)\otimes\exp(\tilde{\boldsymbol{\omega}}_2\tilde{\beta}_{2,n}(\tau))\otimes\ldots\otimes\exp(\tilde{\boldsymbol{\omega}}_n\tilde{\beta}_{n,n}(\tau))+\\ \exp(\tilde{\boldsymbol{\omega}}_1\tilde{\beta}_{1,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_1\tilde{\beta}''_{1,n}(\tau)\otimes\exp(\tilde{\boldsymbol{\omega}}_2\tilde{\beta}_{2,n}(\tau))\otimes\ldots\otimes\exp(\tilde{\boldsymbol{\omega}}_n\tilde{\beta}_{n,n}(\tau))+\\ \exp(\tilde{\boldsymbol{\omega}}_1\tilde{\beta}_{1,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_1\tilde{\beta}'_{1,n}(\tau)\otimes\exp(\tilde{\boldsymbol{\omega}}_2\tilde{\beta}_{2,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_2\tilde{\beta}'_{2,n}(\tau)\otimes\ldots\otimes\exp(\tilde{\boldsymbol{\omega}}_n\tilde{\beta}_{n,n}(\tau))+\\ \exp(\tilde{\boldsymbol{\omega}}_1\tilde{\beta}_{1,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_1\tilde{\beta}'_{1,n}(\tau)\otimes\exp(\tilde{\boldsymbol{\omega}}_2\tilde{\beta}_{2,n}(\tau))\otimes\ldots\otimes\exp(\tilde{\boldsymbol{\omega}}_n\tilde{\beta}_{n,n}(\tau))\otimes\tilde{\boldsymbol{\Omega}}_n\tilde{\beta}'_{n,n}(\tau)+\ldots\end{bmatrix}\tag{16}$$

The reason to expand the polynomial from a third-order (as shown in Kim et al. [24]) to a fifth-order is that it is desired to fix the first- and second-order derivatives of the quaternion function at the endpoints. In particular, for the case of fifth-order Bezier polynomial, by applying Eqs. (10–13), the derivative values at the endpoints can be calculated with the simple expression:

$$\left.\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\tau}\right|_{\tau=0}=5\mathbf{q}_0\otimes\tilde{\boldsymbol{\Omega}}_1,\qquad \left.\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\tau}\right|_{\tau=1}=5\mathbf{q}_f\otimes\tilde{\boldsymbol{\Omega}}_5\tag{17}$$

Note that in Eq. (17), the first-order derivative of the quaternion describes how the vector parameters $\tilde{\boldsymbol{\omega}}_1$ and $\tilde{\boldsymbol{\omega}}_5$ are related to the angular velocity at the endpoints. In a similar fashion, it yields

$$\begin{aligned}\left.\frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}\tau^2}\right|_{\tau=0}&=-20\mathbf{q}_0\otimes\tilde{\boldsymbol{\Omega}}_1+25\mathbf{q}_0\otimes\tilde{\boldsymbol{\Omega}}_1^2+20\mathbf{q}_0\otimes\tilde{\boldsymbol{\Omega}}_2\\ \left.\frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}\tau^2}\right|_{\tau=1}&=-20\tilde{\mathbf{q}}_4\otimes\tilde{\boldsymbol{\Omega}}_4\otimes\tilde{\mathbf{q}}_4^{-1}\otimes\mathbf{q}_f+20\mathbf{q}_f\otimes\tilde{\boldsymbol{\Omega}}_5+25\mathbf{q}_f\otimes\tilde{\boldsymbol{\Omega}}_5^2\end{aligned}\tag{18}$$

B. Mapping from the Virtual to Time Domain

Now that the trajectory is parameterized as a function of the virtual domain variable τ, a mapping must be employed to convert this trajectory into a time dependent one. To do this, a speed factor $\lambda>0$ is defined that maps the points of the trajectory from the virtual domain to the time domain:

$$\lambda(\tau)=\frac{\mathrm{d}\tau}{\mathrm{d}t},\qquad t(\tau)=\int_0^\tau\frac{\mathrm{d}\tau}{\lambda(\tau)}\tag{19}$$

The more complex is the structure of $\lambda(\tau)$, the more flexibility is provided in the definition of the trajectory. However, for this application, we restrict $\lambda(\tau)$ to a positive value function that contains a reduced number of varied parameters as follows:

$$\lambda(\tau)=a+b\tau^2+c(1-\tau)^2+d(1-(1-\tau)^2)+e(1-\tau^2)\tag{20}$$

The analytical integral of Eq. (19) not only provides computational efficiency and an accurate integration to the minimum-time performance index, but also provides a continuous mapping from the virtual domain to the time domain. Alternatively stated, once completed the inversion of the dynamics (see below), a continuous control history is available, whose resolution does not suffer from a limited number of node points.

Although a speed factor of the form in Eq. (20) does not allow matching the optimal minimum-time solutions precisely, varying the parameters contained within $\lambda(\tau)$ (a, b, c, d, and e) still allows sufficient variation of the speed along the trajectory defined by Eq. (7) to produce feasible and easy to track solutions.

C. Inverting the Dynamics

As a result of the mapping from virtual to time domain, the expression for the derivatives of $\mathbf{q}$ with respect to time is:

$$\begin{aligned}&\mathbf{q}(t(\tau))=\mathbf{q}(\tau),\qquad \dot{\mathbf{q}}(t(\tau))=\frac{\mathrm{d}\mathbf{q}(t)}{\mathrm{d}t}=\frac{\mathrm{d}\mathbf{q}(\tau)}{\mathrm{d}\tau}\lambda\\ &\ddot{\mathbf{q}}(t(\tau))=\frac{\mathrm{d}^2\mathbf{q}(t)}{\mathrm{d}t^2}=\frac{\mathrm{d}^2\mathbf{q}(\tau)}{\mathrm{d}\tau^2}\lambda^2+\frac{\mathrm{d}\mathbf{q}(\tau)}{\mathrm{d}\tau}\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}\lambda\end{aligned}\tag{21}$$

Note that if the trajectory in the virtual domain $\mathbf{q}(\tau)$ is specified along with the speed trajectory $\lambda(\tau)$, the resulting trajectory of $\mathbf{q}(t)$ as well as its derivatives can be analytically expressed and mapped to the time domain.

Inverting the kinematic Eqs. (3) and differentiating the obtained expression, results in the following expressions of the angular velocity and angular acceleration:

$$\begin{aligned}&\boldsymbol{\Omega}(t)=2\mathbf{q}^{-1}(t)\otimes\dot{\mathbf{q}}(t),\\ &\boldsymbol{\alpha}(t)=2\mathbf{q}^{-1}(t)\otimes\ddot{\mathbf{q}}(t)-\mathbf{q}^{-1}(t)\otimes\dot{\mathbf{q}}(t)\otimes\boldsymbol{\Omega}(t)=2\mathbf{q}^{-1}(t)\otimes\ddot{\mathbf{q}}(t)\\ &\quad-\frac{1}{2}\boldsymbol{\Omega}(t)\otimes\boldsymbol{\Omega}(t)=2\mathbf{q}^{-1}(t)\otimes\ddot{\mathbf{q}}(t)+\frac{1}{2}\boldsymbol{\omega}^2(t)1\end{aligned}\tag{22}$$

where $\boldsymbol{\alpha}=[\alpha_x,\alpha_y,\alpha_z,0]^T$, $\omega^2(t)=|\boldsymbol{\omega}(t)|^2$, and $1=[0,0,0,1]^T$. The first equation of Eq. (22) is obtained from Eq. (3), while the second one is obtained by taking the time derivative of Eq. (3) and solving for $\boldsymbol{\alpha}(t)$. Furthermore, it is immediate to get the reciprocal $\mathbf{q}^{-1}(t)$ from the quaternion $\mathbf{q}(t)$ by considering that for a unit quaternion the reciprocal is equal to the conjugate $\mathbf{q}^*(t)$.

The torque history needed to follow such a trajectory is calculated by inverting Eq. (2):

$$\begin{aligned}T_x(t)&=\alpha_x(t)+(I_{zz}-I_{yy})\omega_y(t)\omega_z(t),\\ T_y(t)&=\alpha_y(t)+(I_{xx}-I_{zz})\omega_x(t)\omega_z(t),\\ T_z(t)&=\alpha_z(t)+(I_{yy}-I_{xx})\omega_y(t)\omega_x(t)\end{aligned}\tag{23}$$

D. Matching Initial Conditions

From the preceding equations, a quaternion history can be developed based on the Bezier polynomial that satisfies predefined quaternion values as well as angular velocities and angular accelerations at the endpoints. In particular, this section explains how the value of the parameters defining the Bezier polynomial expansion is computed. The desired angular velocity and acceleration dictate the quaternion derivatives at the endpoints to be as follows:

$$\begin{aligned}\dot{\mathbf{q}}(0) &= \frac{1}{2}\mathbf{q}(0)\otimes\boldsymbol{\Omega}(0)\\ \dot{\mathbf{q}}(t_f) &= \frac{1}{2}\mathbf{q}(t_f)\otimes\boldsymbol{\Omega}(t_f)\\ \ddot{\mathbf{q}}(0) &= \frac{1}{2}\mathbf{q}(0)\otimes\left[\alpha(0)-\frac{1}{2}\omega^2(0)\right]\\ \ddot{\mathbf{q}}(t_f) &= \frac{1}{2}\mathbf{q}(t_f)\otimes\left[\alpha(t_f)-\frac{1}{2}\omega^2(t_f)\right]\end{aligned}\tag{24}$$

Based on the properties of the fifth-order Bezier polynomial, the coefficients of the quaternion expression are then calculated by:

$$\tilde{\boldsymbol{\Omega}}_1 = \frac{\boldsymbol{\Omega}(0)}{10\lambda(0)}\tag{25}$$

$$\tilde{\boldsymbol{\Omega}}_5 = \frac{\boldsymbol{\Omega}_f}{10\lambda(1)}\tag{26}$$

$$\tilde{\boldsymbol{\Omega}}_2 = \frac{\mathbf{q}_0^{-1}\otimes[\frac{\ddot{\mathbf{q}}(0)}{\lambda^2(0)}-\frac{\dot{\mathbf{q}}(0)}{\lambda^2(0)}\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}|_{\tau=0}]+20\tilde{\boldsymbol{\Omega}}_1-25\tilde{\boldsymbol{\Omega}}_1^2}{20}\tag{27}$$

$$\tilde{\boldsymbol{\Omega}}_4 = \frac{\tilde{\mathbf{q}}_4^{-1}\otimes(-[\frac{\ddot{\mathbf{q}}(t_f)}{\lambda^2(1)}-\frac{\dot{\mathbf{q}}(t_f)}{\lambda^2(1)}\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}|_{\tau=1}]+20\mathbf{q}_f\otimes\tilde{\boldsymbol{\Omega}}_1+25\mathbf{q}_f\otimes\tilde{\boldsymbol{\Omega}}_1^2)\otimes\mathbf{q}_f^{-1}\otimes\tilde{\mathbf{q}}_4}{20}\tag{28}$$

In particular, Eqs. (25) and (26) are obtained from Eqs. (17) and (21). Furthermore, Eqs. (27) and (28) are obtained from Eqs. (18) and (21). Finally, the complementary parameters $\tilde{\mathbf{q}}_i$ are defined as:

$$\tilde{\mathbf{q}}_1 = \mathbf{q}_0\otimes\exp(\tilde{\omega}_1),\qquad \tilde{\mathbf{q}}_4 = \mathbf{q}_f\otimes\exp(\tilde{\omega}_5)^{-1}\tag{29}$$

and

$$\tilde{\mathbf{q}}_2 = \tilde{\mathbf{q}}_1\otimes\exp(\tilde{\omega}_2),\qquad \tilde{\mathbf{q}}_3 = \tilde{\mathbf{q}}_4\otimes\exp(\tilde{\omega}_4)^{-1}\tag{30}$$

Finally,

$$\tilde{\boldsymbol{\omega}}_3 = \log(\tilde{\mathbf{q}}_2^{-1}\otimes\tilde{\mathbf{q}}_3)\tag{31}$$

The resulting benefit of this laborious formulation is that the attitude trajectory history of the quaternion $\mathbf{q}$ that satisfies the unit norm constraint can be specified using a reduced set of parameters. These parameters are the initial and final conditions on the quaternion itself as well as the values of the angular velocity and angular acceleration at those endpoints.

E. Increasing the Polynomial Order

More flexibility in the trajectory can be obtained by increasing the order of the Bezier polynomial used in the basis function. For the case of a seventh-order polynomial, the same structure as Eq. (7) is employed with $n = 7$. This leads to the introduction of $\tilde{\mathbf{q}}_6$, $\tilde{\mathbf{q}}_7$, $\tilde{\boldsymbol{\omega}}_6$, and $\tilde{\boldsymbol{\omega}}_7$, which must be defined to be consistent with previous definitions from Eqs. (7) and (8). If the values of orientation and angular velocity at the endpoints are set, endpoint conditions of angular jerk as well as angular acceleration can be used as varied parameters. Setting a specified (low) value for the initial and final jerk can be critical for slewing maneuvers of flexible spacecraft, to avoid excitation of structural modes.

To do this, the third-order derivative of the output vector is calculated as:

$$\begin{aligned}\mathbf{q} &= \frac{\mathrm{d}^3\mathbf{q}}{\mathrm{d}\tau^3}\lambda^3+\frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}\tau^2}2\lambda\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}\lambda+\frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}\tau^2}\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}\lambda^2+\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\tau}\frac{\mathrm{d}^2\lambda}{\mathrm{d}\tau^2}\lambda^2\\ &\quad+\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}\tau}\left(\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}\right)^2\lambda\end{aligned}\tag{32}$$

The new expressions for the virtual derivatives (taken with respect to the virtual argument τ) are also recalculated and are analogous to Eqs. (16) and (17), but with the addition of a third derivative to accommodate the change between a fifth- and seventh-order polynomial:

$$\mathbf{q}'|_{\tau=0} = 7\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_1,\qquad \mathbf{q}'|_{\tau=1} = 7\tilde{\mathbf{q}}_7\otimes\tilde{\boldsymbol{\Omega}}_7\tag{33}$$

$$\begin{aligned}\mathbf{q}''|_{\tau=0} &= 49\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_1^2+42\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_2-42\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_1\\ \mathbf{q}''|_{\tau=1} &= 42\tilde{\mathbf{q}}_7\otimes\tilde{\boldsymbol{\Omega}}_7+49\tilde{\mathbf{q}}_7\otimes\tilde{\boldsymbol{\Omega}}_7^2-42\tilde{\boldsymbol{\Omega}}_6\otimes\tilde{\mathbf{q}}_7\end{aligned}\tag{34}$$

$$\begin{aligned}\mathbf{q}'''|_{\tau=0} &= 343\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_1^3-882\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_1^2-420\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_2\\ &\quad+210\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_1+210\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_3+882\tilde{\mathbf{q}}_0\tilde{\boldsymbol{\Omega}}_1\otimes\tilde{\boldsymbol{\Omega}}_2\\ \mathbf{q}'''|_{\tau=1} &= 343\tilde{\mathbf{q}}_7\otimes\tilde{\boldsymbol{\Omega}}_7^3-882\tilde{\mathbf{q}}_6\otimes\tilde{\boldsymbol{\Omega}}_6\otimes\tilde{\boldsymbol{\Omega}}_7\otimes\tilde{\mathbf{q}}_6^{-1}\otimes\tilde{\mathbf{q}}_7\\ &\quad+210\tilde{\boldsymbol{\Omega}}_5\otimes\tilde{\mathbf{q}}_7+210\tilde{\mathbf{q}}_7\otimes\tilde{\boldsymbol{\Omega}}_7+882\tilde{\mathbf{q}}_7\otimes\tilde{\boldsymbol{\Omega}}_7^2\\ &\quad+882\tilde{\mathbf{q}}_0\otimes\tilde{\boldsymbol{\Omega}}_1\otimes\tilde{\boldsymbol{\Omega}}_2-420\tilde{\boldsymbol{\Omega}}_6\otimes\tilde{\mathbf{q}}_7\end{aligned}\tag{35}$$

New values for the constants that fix the initial conditions of the quaternion trajectory can be calculated similarly to the fifth-order polynomial case, except that an extra step needs to be taken to accommodate the third-order derivative of $\mathbf{q}$.

V. Solving the Problem Using the IDVD Method

This section presents the results of the IDVD method using two different parameterizations to obtain the minimum-time solutions of the problem posed in Sec. II. As discussed in the previous section, as opposed to hundreds of varied parameters as in the case with the pseudospectral methods, the list of parameters to be optimized using IDVD includes only 11 variables, in the case of fifth-order quaternion parametrization, and 17 variables in the case of seventh-order quaternion parametrization. In particular, in case of fifth-order parametrization, the optimization variables are the boundary values of all three components of the angular acceleration plus the five coefficients defining the virtual speed profile [see Eq. (20)], while the quaternion and angular velocity at the beginning and end of the slewing are given data. Increasing the order of quaternion approximation polynomials from 5 to 7 allows to assign also the boundary conditions for angular acceleration and varying the boundary values of angular jerks (note that, with IDVD to satisfy the higher-order derivatives at the boundary points there is no need to introduce new equations of motion). However, in what follows, for the seventh-order polynomial both the angular jerk and angular acceleration will be varied (i.e., 12 optimization parameters in addition to the five parameters of the speed profile, for a total of 17 parameters). This is done to show the improvement of the performance index to match that of the GPOPS solution.

During the optimization process the constraint is imposed that the resulting control must obey Eq. (5) and $\lambda(\tau) > 0$. Initial guesses for the angular acceleration and jerk were taken to be equal to zero, initial guess for the zero-order coefficient of $\lambda(\tau)$ is $a = \frac{1}{2\phi}$; initial guess of all of the other parameters of $\lambda(\tau)$ is zero. The initial guess value of a is an approximate estimate of the inverse of the time it takes to change attitude by an angle ϕ with the maximum angular rate. This choice of initial guesses contributes to have a feasible initial solution.

A. IDVD Solutions Using Fifth-Order Bezier Polynomial

Figures 9–11 present the results obtained applying the IDVD with a quaternion based on a fifth-order Bezier polynomial (compare it

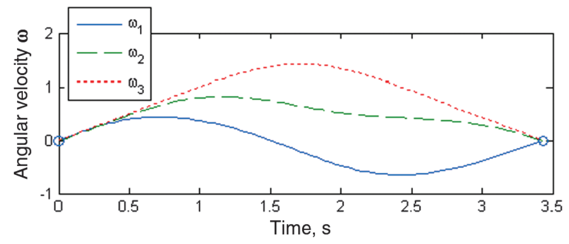
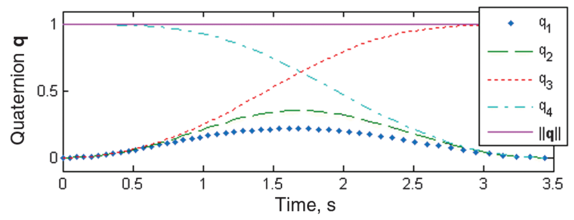

Fig. 9 Time histories of the states for Case 1 (fifth-order Bezier polynomials IDVD solution).

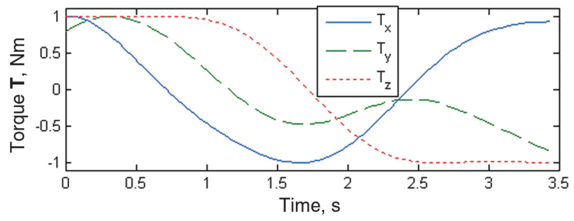

Fig. 10 Time history of controlling torques for Case 1 (fifth-order Bezier polynomials IDVD solution).

with the GPOPS solution presented in Figs. 1–3). The solution was run using 100 points (although, as opposed to GPOPS, it would make no difference running it for a larger or smaller number of points), and resulted in a slightly higher value of t_f, but it took significantly less time to converge (to be discussed further, in Sec. V). Figure 12 shows the virtual speed factor, the key element in matching the virtual and time domains.

The major difference compared with the GPOPS solution is that the controls do not have a bang–bang nature. Again, this was done choosing the proper quaternion parameterization. When implemented in the real-time controller, these controls may be easier to track. Also, having different controlling torques at the endpoints, means having different angular accelerations. While for the GPOPS solution the initial and terminal angular accelerations are at the mercy of the optimization routine, using IDVD allows matching them with the current accelerations, making the control algorithm more robust.

The resulting solution for the same scenario using just 25 nodes is shown in Figs. 13–15. As in the case of GPOPS, it also converges to another equally optimal solution.

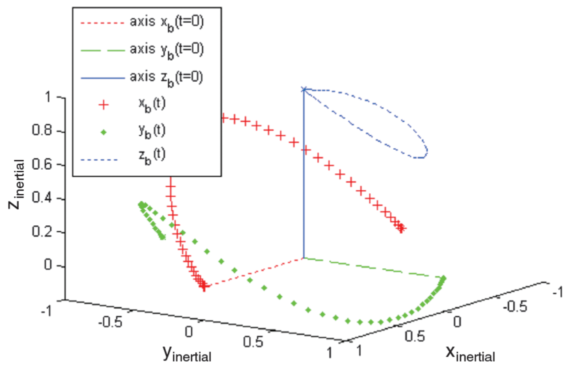

Fig. 11 Principal axis outline of 180° slew for Case 1 (fifth-order Bezier polynomials IDVD solution).

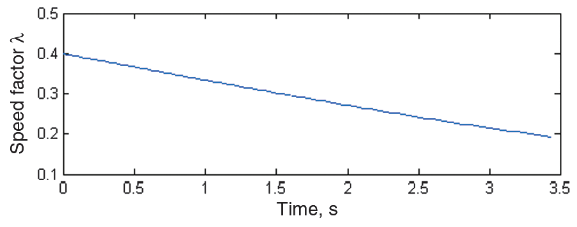
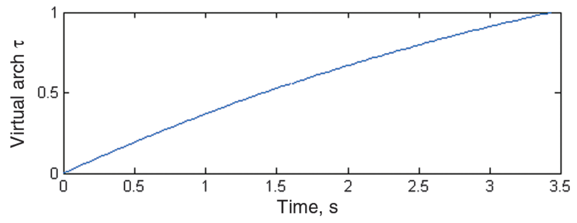

Fig. 12 Mapping the virtual and time domains for the Case 1 solution (fifth-order Bezier polynomials IDVD solution).

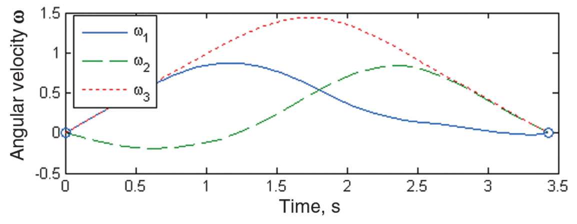
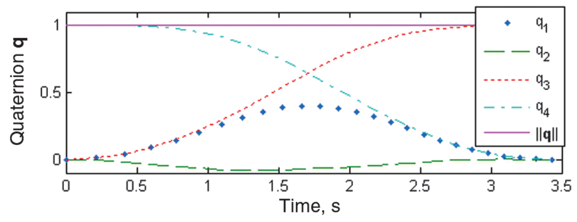

Fig. 13 Time histories of the states for Case 1 with 25 nodes (fifth-order Bezier polynomials IDVD solution).

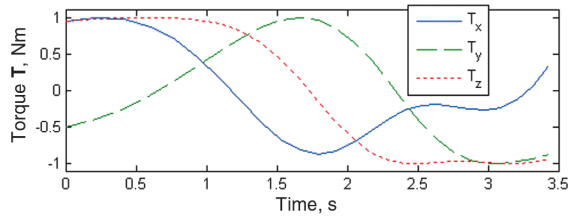

Fig. 14 Time history of controlling torques for Case 1 using 25 nodes (fifth-order Bezier polynomials IDVD solution).

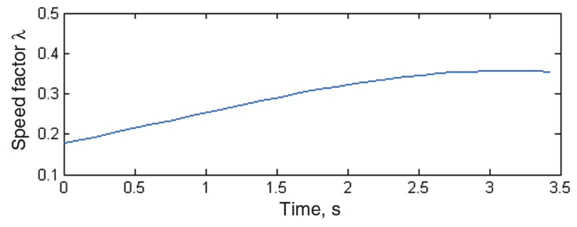
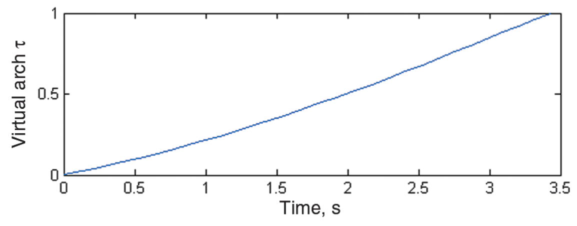

Fig. 15 Mapping the virtual and time domains for the Case 1 solution using 25 nodes (fifth-order Bezier polynomials IDVD solution).

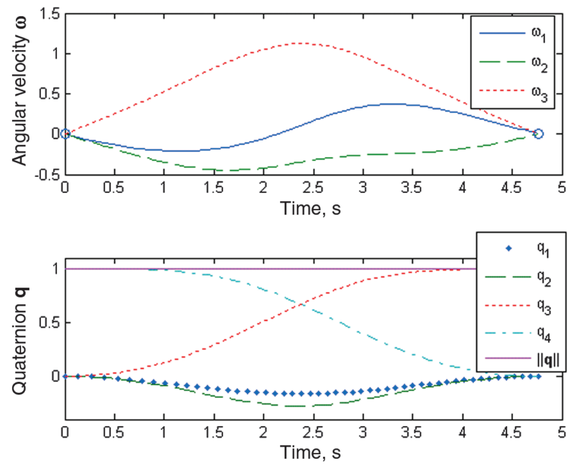
Fig. 16 Time histories of the states for Case 2 (fifth-order Bezier polynomials IDVD solution).

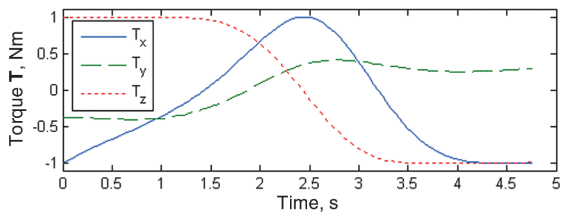
Fig. 17 Time history of controlling torques for Case 2 (fifth-order Bezier polynomials IDVD solution).

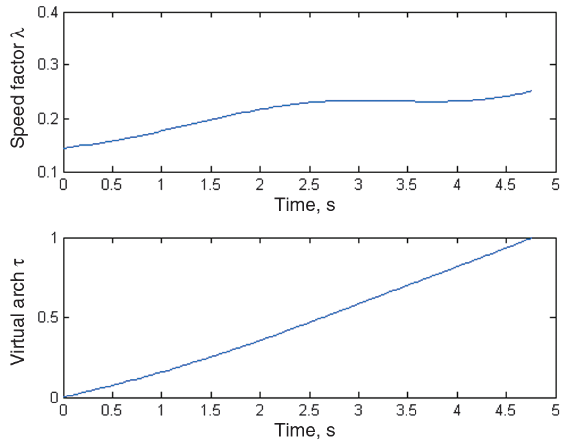
Fig. 18 Mapping the virtual and time domains for the Case 2 solution (fifth-order Bezier polynomials IDVD solution).

With any number of nodes the IDVD solution results in a smooth control history, readily available to be fed forward to the tracking.

As in the case of the GPOPS solution for Case 2, the nonsymmetrical inertia matrix causes certain changes as compared with the symmetric matrix solution. The 100-node IDVD solution in this case results in 4.767 s maneuver and requires about a minute to compute (see Figs. 16–18).

B. IDVD Solutions Using Seventh-Order Bezier Polynomial

To compare the results obtained using different polynomial orders, Figs. 19–21 present the solution of the same problem using a quaternion based on a seventh-order Bezier polynomial with angular acceleration and jerk varied at both ends at the trajectory. As shown, this brings a solution closer to that of GPOPS but doubles the computational time required to converge. The following section addresses this issue in more detail.

VI. IDVD vs GPOPS Results Comparison

This section presents a comparison of the results obtained using the IDVD method with those of the GPOPS method. It disregards the fact that the results obtained with GPOPS for low number of nodes are infeasible, but rather concentrates on the computational advantages the IDVD approach has for any number of intermediate points (nodes in the case of GPOPS). To start with, Tables 2 and 3 summarize the 180° rest-to-rest slew maneuver solutions for symmetric and asymmetric inertia matrix, obtained using GPOPS and IDVD as discussed in Secs. III and V.

In these tables, all results are compared versus the eigenaxis maneuver solution. First, it is shown that the true optimal solution, obtained offline in [3], which is not an eigenaxis rotation, provides about 8.5 and 11% improvement of the performance index, the maneuver time t_f, for Case 1 and Case 2, respectively. It would be appropriate to exploit this economy solution onboard, but unfortunately it cannot be produced in real time and hence the need for other methods still arises.

As seen from Tables 2 and 3 the GPOPS solution converging to one of the equally optimal solutions, assures about the same gain in the performance index as in the truly optimal one. But again it takes too much computational time to be implemented onboard. Specifically, the 100-node solution that does converge and assures a smooth controls history, takes about an hour to converge.

As discussed in the previous section the IDVD solution has a much more robust performance, allowing computing the same type of maneuvers just in a few seconds as opposed to hours (using an executable optimization library the IDVD method produces solutions in fractions of a second [22]). Of course, some of the optimality (performance index value) has to be sacrificed. On the positive side, the solution is always feasible and smooth for any number of computational points, and can be brought closer to the GPOPS solution (in

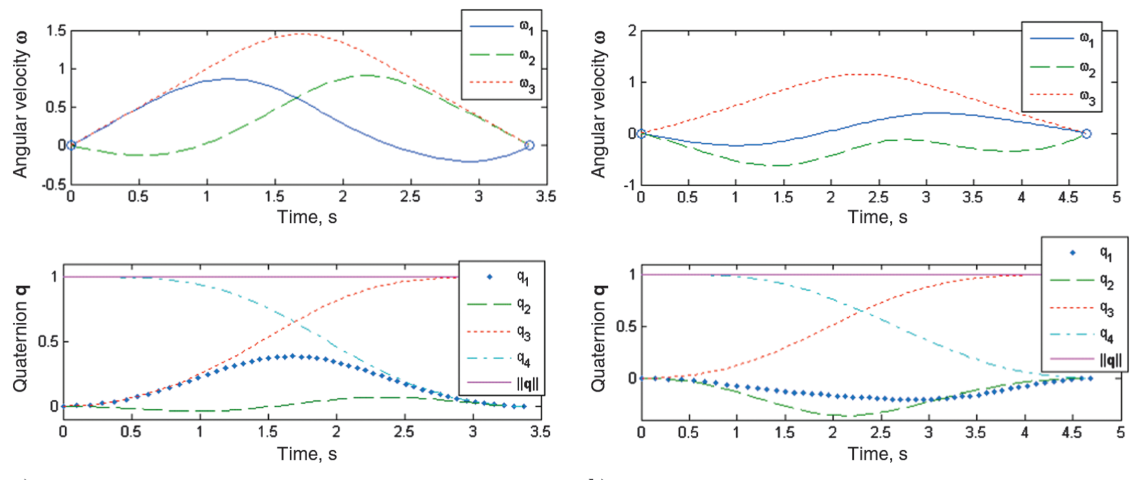
a) b)

Fig. 19 Time histories of the states for a) Case 1 and b) Case 2 with 100 nodes (seventh-order Bezier polynomials IDVD solution).

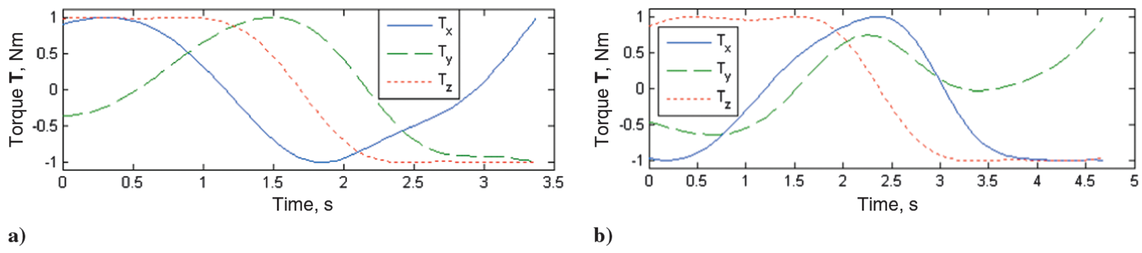

Fig. 20 Time history of controlling torques for a) Case 1 and b) Case 2 with 100 nodes (seventh-order Bezier polynomials IDVD solution).

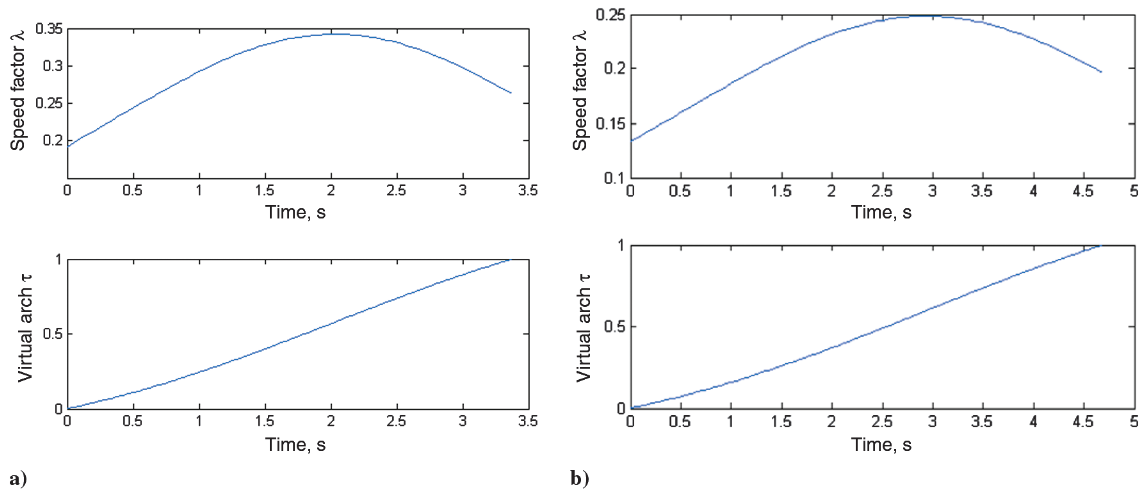

Fig. 21 Mapping the virtual and time domains for a) Case 1 and b) Case 2 solutions with 100 nodes (seventh-order Bezier polynomials IDVD solution).

terms of the value of a performance index) by increasing the number of varied parameters (the order of the quaternion approximation polynomial). Furthermore, IDVD has an analytic representation of the solution, which allows the number of nodes generated, possible for better tracking performance, to be increased without complex interpolation schemes or recalculating the entire solution.

In conclusion, the GPOPS candidate solution should use no less than 100 nodes, and about 10% gain in the performance index "costs"

Table 2 The 180° rest-to-rest slew maneuver about the z-axis, symmetric inertia (Case 1)

Trajectory generation method	Nodes	Computational time, s	Cost (t_f)	% Improvement
Eigenaxis	——	——	3.5449	~0%
Optimal (Bilimoria/Wie [3])	——	——	3.2431	8.51%
GPOPS	25	15.5	3.2573	8.11%
GPOPS	50	200.5	3.2859	7.31%
GPOPS	100	5962.8	3.2430	8.52%
IDVD fifth-order	25	3.7	3.4289	3.27%
IDVD fifth-order	50	4.8	3.4373	3.04%
IDVD fifth-order	100	10.0	3.4382	3.01%
IDVD seventh-order	25	44.0	3.3756	4.78%
IDVD seventh-order	50	69.8	3.3769	4.74%
IDVD seventh-order	100	121.2	3.3776	4.72%

Table 3 The 180° rest-to-rest slew maneuver about the z-axis, asymmetric inertia (Case 2)

Trajectory generation method	Nodes	Computational time, s	Cost (t_f)	% Improvement
Eigenaxis	——	~0	5.0133	~0%
GPOPS	25	22.2	4.4609	11.02%
GPOPS	50	520.1	4.4499	11.24%
GPOPS	100	1893.9	4.4528	11.18%
IDVD fifth-order	25	4.8	4.7857	4.54%
IDVD fifth-order	50	7.6	4.7861	4.53%
IDVD fifth-order	100	9.5	4.7864	4.53%
IDVD seventh-order	25	24.3	4.6768	6.71%
IDVD seventh-order	50	41.2	4.6846	6.56%
IDVD seventh-order	100	80.4	4.6869	6.51%

Table 4 The 90° rest-to-rest slew maneuver about the z-axis, symmetric inertia (Case 1)

Trajectory generation method	Nodes	Computational time, s	Cost (t_f)	% Improvement
Eigenaxis	——	——	2.5066	~0%
GPOPS	25	20.7	2.4336	2.91%
GPOPS	50	120.0	2.4332	2.93%
GPOPS	100	236.2[a]	2.4296	3.07%
IDVD fifth-order	25	5.8	2.5654	−2.34%
IDVD fifth-order	50	7.2	2.5666	−2.39%
IDVD fifth-order	100	9.4	2.5671	−2.41%
IDVD seventh-order	25	39.7	2.5043	0.09%
IDVD seventh-order	50	58.9	2.5058	0.03%
IDVD seventh-order	100	77 3	2.5058	0.03%

[a]Solved recursively using previous 50 node solution as initial guess.

Table 5 The 90° rest-to-rest slew maneuver about the z-axis, asymmetric inertia (case 2)

Trajectory generation method	Nodes	Computational time, s	Cost (t_f)	% Improvement
Eigenaxis	——	~0	3.5449	——
GPOPS	25	17.4	3.4450	2.82%
GPOPS	50	166.0	3.4408	2.94%
GPOPS	100	1432.1	3.4430	2.87%
IDVD fifth-order	25	5.9	3.6277	−2.33%
IDVD fifth-order	50	9.8	3.6307	−2.42%
IDVD fifth-order	100	17.3	3.6316	−2.44%
IDVD seventh-order	25	41.5	3.5373	0.21%
IDVD seventh-order	50	80.4	3.5389	0.17%
IDVD seventh-order	100	158.8	3.5394	0.16%

Table 6 The 90° maneuver for symmetric inertia (case 1) and nonzero boundary rates

Trajectory generation method	Nodes	Computational time, s	Cost (t_f)	% Improvement with respect to GPOPS 100 solution
GPOPS	25	48.3	2.4028	−0.07%
GPOPS	50	333.9	2.4016	−0.02%
GPOPS	100	1182.6	2.4011	0.00%
IDVD fifth-order	25	5.9	2.5437	−5.94%
IDVD fifth-order	50	5.6	2.5450	−5.99%
IDVD fifth-order	100	6.1	2.5451	−6.00%
IDVD seventh-order	25	27.9	2.4885	−3.64%
IDVD seventh-order	50	41.5	2.4895	−3.68%
IDVD seventh-order	100	90.6	2.4896	−3.69%

an order of an hour of CPU time. As discussed in Sec. III, this solution features a bang–bang control, i.e., does not account for controllers' dynamics, and therefore can still not be used onboard as is. On the other hand, the always-feasible and ready-to-go IDVD solution (employing as low as say 25 computational points) can be produced much faster, but surrenders up to two-thirds of its gain as compared with that of the GPOPS solution (about one half for the seventh-order approximation).

Tables 4 and 5 present similar data for the 90° rest-to-rest slew maneuver. While GPOPS provides about 3% gain compared with a simple eigenaxis slew solution, the IDVD method has almost no advantage or may be even worse if using a fifth-order quaternion approximation. All major conclusions, however, remain the same.

Table 6 compares GPOPS and IDVD solutions for one of such cases, when $\boldsymbol{\omega}_0 = -\boldsymbol{\omega}_f = \frac{1}{10}\mathbf{1}_{3\times 1}$ (other sets of nonzero boundary conditions were explored as well, and proved to maintain the same trends). In this table all results are compared against a valid 100-node GPOPS solution. As seen, the GPOPS solutions with a lesser number of nodes produce somewhat infeasible solutions, meaning that they cannot be implemented in the control scheme explicitly. The IDVD solutions may yield to GPOPS as much as about 4% with respect to the performance index, but again are produced much faster. Furthermore, while the 90 and 180° rest-to-rest slew maneuvers with zero boundary rates feature multiple equally optimal solutions, so that both GPOPS and IDVD solutions converge to different solutions, when changing the number of nodes (GPOPS)/computational points (IDVD), in the case of nonzero boundary rates they all converge to the same solution as illustrated in Fig. 22.

It should be noted that in practice, the direct methods would likely be used also in situations where the end-conditions (angular rates, accelerations) of the slew are specified and not equal to zero. For this case no simple eigenaxis slew solution exists and therefore any solution produced online would be good.

The final observation is that apparently there is no need to use as complex of an approximation for the speed factor $\lambda(\tau)$ as that of Eq. (18). By looking at Figs. 12, 15, 18, and 21 it appears that the number of coefficients (varied parameters) can be easily reduced to 3 (which would result in even faster convergence, but at the cost of a slightly larger performance index):

$$\lambda(\tau) = a + b\tau + c\tau^2 \tag{36}$$

In this case the optimization routine should assure the following inequalities hold:

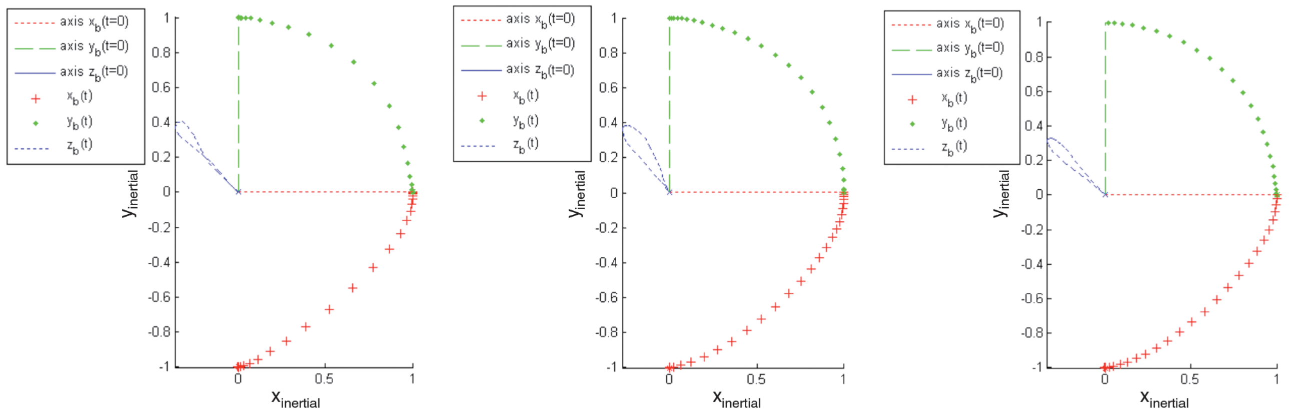

Fig. 22 Projections of the 90° rotation maneuver for a) GPOPS, b) fifth-order, and c) seventh-order approximation solutions.

$$a > 0, \qquad a + b + c > 0, \\ 4ac - 2b(1 + b) > 0, \quad \text{if } c > 0 \quad \text{and} \quad -2c < b < 0 \tag{37}$$

VII. Conclusions

The inverse dynamics optimization method with the novel quaternion approximation functions proposed in this paper allows computing feasible solutions fast enough to be used onboard satellites for online computation of slew maneuvers. Moreover, because of the smooth control histories it can be implemented in the control schemes involving a feedforwad loop. Compared with the true time-optimal solutions the inverse dynamics trajectories do not have a bang–bang control, which results in a slightly worse performance index. However, smooth controls benefit other mission preferences of having desired rates at the endpoints. In addition, it is a definite advantage in rapidly changing acquisition or tracking scenarios and when the slewing spacecraft possesses low-frequency flexible modes. The formulation presented in this paper is currently being applied to situations where the attitude is coupled with other dynamics such as translational motion in rendezvous and docking applications. In this case, a simple eigenaxis slew would not meet mission criteria such as matching rotational motion.

Appendix

In this Appendix, the pseudocode is provided for the IDVD optimization algorithm introduced in Sec. IV. The case of using a fifth-order Bezier quaternion parametrization is presented here; the procedure is analogous for the seventh-order Bezier parametrization case.

Given the following data, where $\mathbf{q}(0)$ and $\mathbf{q}(t_f)$ are the start and end orientations; $\boldsymbol{\omega}(0)$ and $\boldsymbol{\omega}(t_f)$ are the start and end angular velocities;

$\mathbf{I} = \mathrm{diag}([I_{xx}, I_{yy}, I_{zz}])$ is the inertia matrix; and $\mathbf{T}_{\min} \le \mathbf{T} \le \mathbf{T}_{\max}$ are the constraints on torques:

Step 1) The nonlinear programming solver sets new iteration values of the following optimization variables (for the first iteration, initial guess values are used): $\dot{\omega}_1(0), \dot{\omega}_2(0), \dot{\omega}_3(0)$ are components of angular acceleration at start; $\dot{\omega}_1(t_f)$, $\dot{\omega}_2(t_f)$, and $\dot{\omega}_3(t_f)$ are components of angular acceleration at end; and $a > 0$, $b > 0$, $c > 0$, $d > 0$, and $e > 0$ are defining $\lambda(\tau)$ [see Eq. (20)].

Step 2) Evaluate $\lambda(\tau)$ and $\lambda'(\tau)$ at a desired set of evaluation points τ (e.g., at 100 points within the interval $0 \leftrightarrow 1$) by using Eq. (20).

Step 3) Evaluate

$$t(\tau_i) = \int_0^{\tau_i} \frac{\mathrm{d}\tau}{\lambda(\tau)}$$

at the evaluation points and evaluate cost function

$$J = t_f = t(1) = \int_0^1 \frac{\mathrm{d}\tau}{\lambda(\tau)}$$

Step 4) By using Eqs. (24) and (31), compute the constant parameters of the fifth-order Bezier quaternion trajectory that satisfies the set boundary conditions (see given data), the current value of the optimization parameters (from step 1), and the boundary values on $\lambda(\tau)$ and $\lambda'(\tau)$ (from step 4).

Step 5) By using Eqs. (7), (15), and (16), compute the values of $\mathbf{q}(\tau)$, $\mathbf{q}'(\tau)$, and $\mathbf{q}''(\tau)$ at the evaluation points.

Step 6) By using Eq. (21), perform the transformation from the virtual domain into the time domain to obtain $\mathbf{q}(t(\tau))$, $\dot{\mathbf{q}}(t(\tau))$, and $\ddot{\mathbf{q}}(t(\tau))$ at the evaluation points.

Step 7) By using Eqs. (22) and (23), invert the dynamics and compute the values of angular velocity, angular accelerations, and needed torque components at the evaluation points.

Step 8) First, decide whether the constraints on the torque components are satisfied. If yes, go to step 9. If no, go to step 1.

Step 9) Next, decide whether the evaluated cost function value (see step 3) is the minimum value (within set accuracy). If yes, then end. If no, go to step 1.

References

[1] Vadali, S. R., and Junkins, J. L., "Optimal Open-Loop and Stable Feedback Control of Rigid Spacecraft Attitude Maneuvers," *Journal of the Astronautical Sciences*, Vol. 32, No. 2, April–June 1984, pp. 105–122.

[2] Junkins, J. L., and Turner, J. D., *Optimal Spacecraft Rotational Maneuvers*, Elsevier, New York, 1986, pp. 171–222.

[3] Bilimoria, K. D., and Wie, B., "Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 3, 1993, pp. 446–452. doi:10.2514/3.21030

[4] Bai, X., and Junkins, J. L., "New Results for Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, 2009, pp. 1071–1076. doi:10.2514/1.43097

[5] Melton, R. G., "Constrained Time-Optimal Slewing Maneuvers for Rigid Spacecraft," *Advances in the Astronautical Sciences*, AAS Paper 09-309; also, Univelt, Inc., San Diego, CA, Vol. 135, 2010, pp. 107–126.

[6] Davis, T., and Straight, S., "Development of the Tactical Satellite 3 for Responsive Space Missions," *Proceedings of the 4th Responsive Space Conference*, AIAA 2006-4003, April 2006.

[7] Yakimenko, O., Xu, Y., and Basset, G., "Computing Short-Time Aircraft Maneuvers Using Direct Methods," *Journal of Computer and Systems Sciences International*, Vol. 49, No. 3, 2010, pp. 145–176.

[8] Boyarko, G., Yakimenko, O., and Romano, M., "Formulation and Analysis of Matching Points of Interest in Two-Spacecraft for Optimal Rendezvous," *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, AIAA 2009-5669, Aug. 2009.

[9] Boyarko, G., Yakimenko, O., and Romano, M., "Modeling and Optimization of a Spacecraft Maneuvering with Respect to a Tumbling Object," *Proceedings of the AAS/AIAA Astrodynamics Specialist

Conference, AAS 09-316, Aug. 2009.

[10] McInnes, C. R., "Satellite Attitude Slew Maneuver using Inverse Control," *The Aeronautical Journal*, Vol. 102, May 1998, pp. 259–265.

[11] Louembet, C., Cazaurang, F., Zolghardi, A., Charbonnel, C., and Pittet, C., "Design of Algorithms for Satellite Slew Manoeuver by Flatness and Collocation," *Proceedings of the American Control Conference*, IEEE Publications, Piscataway, NJ, July 2007.

[12] Tanygin, S., and Woodburn, J., "Optimal Switching Between Targets Using Rate-Limited Slews," *AAS/AIAA Astrodynamics Specialists Conference*, AAS Paper 05-331, Aug. 2005.

[13] Wie, B., *Space Vehicle Dynamics and Control*, AIAA Education Series, AIAA, Reston, VA, 1998, pp. 326–327.

[14] Greenwood, D. T., *Principles of Dynamics*, Prentice–Hall, Upper Saddle River, NJ, 1987, pp. 99–103.

[15] Gill, P. E, Murray, W., and Saunders, A., User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming, Univ. of California, 92093-0112, San Diego, CA, 2006.

[16] Rao, A. V., Benson, D. A., Darby, C. L., Patterson, M. A., Francolin, C., and Huntington, G. T., "Algorithm 902: GPOPS, A Matlab Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, April–June 2009, pp. 1–39. doi:10.1145/1731022.1731032

[17] Matlab Version 2009a Users Manual, The Mathworks, Natick, MA, 2009.

[18] Fleming, A., "Real-time Optimal Slew Maneuver Design and Control," Astronautical Engineer's Thesis, U.S. Naval Postgraduate School, 2004.

[19] Ross, I. M., User's Manual for DIDO: A MATLAB Application Package for Solving Optimal Control Problems, Tomlab Optimization, Sweden, Feb 2004.

[20] Yakimenko, O., "Direct Method for Rapid Prototyping of Near Optimal Aircraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, 2000, pp. 865–875. doi:10.2514/2.4616

[21] Cowling, I. D., Yakimenko, O., Whidborne, J. F., and Cooke, A. K., "A Prototype of an Autonomous Controller for a Quadrotor UAV," *Proceedings of the European Control Conference*, ECC, Budapest, July 2007.

[22] Yakimenko, O., and Slegers, N., "Using Direct Methods for Terminal Guidance of Autonomous Aerial Delivery Systems," *Proceedings of the European Control Conference*, ECC, Budapest, Aug. 2009.

[23] Milam, M., "Real-Time Optimal Trajectory Generation for Constrained Dynamical Systems," Doctoral Thesis, California Inst. of Technology, 2003.

[24] Kim, M., Kim, M., and Shin, S., "A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives," *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, 1995, pp. 369–376.