Geometric Integration of Quaternions

Michael S. Andrle*, John L. Crassidis†
University at Buffalo, State University of New York, Amherst, NY, 14260-4400

This paper employs a geometric integration algorithm to propagate the quaternion kinematics in order to preserve the unit norm. While many studies have focused on this aspect specifically while considering other systems, this work has the additional objective of studying result accuracy. Many applications including space object tracking and asteroid cataloguing require state propagation over longer time intervals where only sparse observation data is available. It is during these intervals that the quaternion norm drifts and accuracy decreases as a result of error accumulation. Quaternion trajectories obtained using third and fourth order Crouch-Grossman Lie group methods are compared with those calculated using the classical third and fourth order Runge-Kutta algorithms using different time steps. Results show that the use of the Crouch-Grossman Lie group method better preserves the quaternion unit norm for the larger time steps considered. It is also found that the fourth order Crouch-Grossman algorithm is more accurate than its Runge-Kutta counterpart except for the smallest time step used.

I. Introduction

The group of unit quaternions is often used in practice to parameterize the attitude of an object or reference frame with respect to another, for example the attitude of a space vehicle, an orbiting satellite, or an asteroid with respect to an inertial reference frame. In navigation or tracking applications, dynamic models are combined with available observation data into an estimator in order to improve state knowledge as well as to identify unknown parameters. One important component of sequential state estimation is to propagate the system state in the absence of observation data. Quaternions lead to difficulties during this propagation stage since classical numerical integration methods are unable to preserve the unit norm. Additionally, brute-force normalizations may introduce errors that accumulate over longer propagation periods.

Quaternion norm preservation and accuracy is important in many areas; one prevailing subject is space situational awareness. The U.S. Air Force is collecting data in order to catalogue Earth orbiting satellites, objects and debris. Accurate tracking of all Earth-orbiting objects is important since their presence can be extremely hazardous to current space missions as well as disruptive for any future mission. One method currently used to estimate the attitude of such objects makes use of light curve data.[1] This concept has also been improved upon in order to estimate the object's orbit trajectory and determine the object's shape.[2] However, data is sparse as a result of the number of sensors available and the large amount of objects in orbit.[2] As a result, an object's attitude may often need to be propagated very accurately over long time intervals before an update is possible.

Asteroid tracking poses the same difficulties because of its similarity to the aforementioned problem of cataloguing Earth-orbiting objects. Light curve data is also being used to estimate the shape and state of asteroids,[3,4] although observations are again infrequent and limited. Accurate integration algorithms are required in order to provide reasonable state estimates when new observation data becomes available.

Classical numerical integration methods may introduce large inaccuracies due to their inherent inability to preserve quaternion group properties. In practice, a brute-force normalization is performed at each time step to maintain the quaternion unit norm. This may however introduce additional errors in accuracy. Propagation errors due to the lack of quaternion norm preservation or the accumulation of errors from brute-force normalizations may affect the attitude estimate during an extended time interval without observation.

*Graduate Student, Department of Mechanical & Aerospace Engineering. Email: msandrle@buffalo.edu.
†Professor, Department of Mechanical & Aerospace Engineering. Email: johnc@buffalo.edu, Associate Fellow AIAA.

The objective is to employ a Lie group method developed by Crouch and Grossman[5] in order to integrate a quaternion parameterized attitude. Its ability to preserve the unit norm will be compared with that of the classical Runge-Kutta (RK) method. While it is expected that the attitude state will better remain on its manifold using a Lie group method, this work is equally concerned with determining what improvement in accuracy might be obtained. The accuracy of quaternion estimates obtained using RK methods is compared with those obtained using the Crouch-Grossman (CG) method. Additionally, a RK method with brute-force quaternion normalizations at each time step will be considered. This will permit the determination of any impact these operations have on result accuracy and more importantly the comparison of the new approach with that typically employed in practice.

The paper is organized as follows. In Section II, the Lie group structure of quaternions is described. Section III recalls classical RK algorithms, which at present are extensively employed in space applications to propagate attitude state knowledge. This section also provides the CG algorithm whose capabilities and performance this work explores relative to classical RK methods. The fundamental differences between these two approaches, which are very important in application, are also discussed. Finally in Section IV, simulations are conducted to highlight any advantage of each, particularly in the accuracy of the results over extended propagation intervals. Simulation parameters are defined such that a closed-form solution is available as a basis for evaluation.

II. Quaternions as a Lie Group

The set of unit quaternions given by the unit sphere S^3 in $\mathbb{R}^4$ forms a group under quaternion multiplication. This group is a *Lie group* of 3 dimensions and is isomorphic to the *special unitary group* SU(2).[6] The Lie algebra of SU(2), which is written as $\mathfrak{su}(2)$, is isomorphic to $\mathbb{R}^3$. Each quaternion is expressed by its real vector representation $\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T$. Accordingly, each element of $\mathfrak{su}(2)$, denoted by $\boldsymbol{\omega} \in \mathbb{R}^3$, is identified with its real matrix representation

$$\frac{1}{2}\Omega(\boldsymbol{\omega}) \equiv \frac{1}{2}\begin{bmatrix} -[\boldsymbol{\omega}\times] & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \in \mathfrak{su}(2) \tag{1}$$

where the cross product matrix is given by

$$[\boldsymbol{\omega}\times] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2}$$

In this form, the quaternion kinematics are[7]

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q} \tag{3}$$

which is a differential equation on SU(2).

It is important to recall the exponential operator, $\exp(\cdot)$, which here maps as

$$\exp : \mathfrak{su}(2) \to \mathrm{SU}(2) \tag{4}$$

For example, in the case of constant $\boldsymbol{\omega}$ the solution to Eq. (3) can be written as

$$\begin{aligned} \mathbf{q}(t) &= \exp\left(t\frac{1}{2}\Omega(\boldsymbol{\omega})\right)\mathbf{q}(0) \\ &= \mathbf{q}_\omega \otimes \mathbf{q}(0) \end{aligned} \tag{5}$$

where $\mathbf{q}_\omega$ is the quaternion corresponding to a rotation about the axis $\boldsymbol{\omega}/\|\boldsymbol{\omega}\|$ through the angle $t\|\boldsymbol{\omega}\|$ and $\otimes$ denotes quaternion multiplication. Solutions to Eq. (3) are known for some more general cases.[8] In all cases however, returning to Eq. (3), the tangent space at $\mathbf{q}$ is of the form $\frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q}$ and the solution of Eq. (3) is such that $\mathbf{q}(t)$ belongs to S^3 for all t. This is the group property that should be preserved throughout numerical integration.

III. Numerical Integration

In this section, the numerical algorithms used to integrate Eq. (3) are introduced. The first is the Runge-Kutta (RK) algorithm and the second is the Crouch-Grossman (CG) Lie group approach. In Section IV, results of integrating Eq. (3) using both methods will be compared.

A. Classical Runge-Kutta Integration

From the quaternion kinematics Eq. (3), let

$$\mathbf{f}(\mathbf{q}, t) = \frac{1}{2}\Omega(\boldsymbol{\omega}(t))\mathbf{q} \tag{6}$$

With this definition, the classical s-stage Runge-Kutta integration algorithm applied to Eq. (3) is given by[9]

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}^{(i)} \tag{7a}$$

$$\mathbf{k}^{(i)} = \mathbf{f}(\mathbf{q}^{(i)}, t_k + c_i \Delta t) \tag{7b}$$

$$\mathbf{q}^{(i)} = \mathbf{q}_k + \Delta t \sum_{j=1}^{i-1} a_{ij} \mathbf{k}^{(j)} \tag{7c}$$

where Δt is the time step and $\{a_{ij}\}_{1\leq j<i\leq s}$, $\{b_i\}_{1\leq i\leq s}$, and $\{c_i\}_{1\leq i\leq s}$ are prescribed coefficients. The primary disadvantage of the RK integration method is that the integration of a given $\mathbf{q}_k \in S^3$ produces the approximation $\mathbf{q}_{k+1}$ which does not belong to S^3. This is easily observed in Eq. (7a). As mentioned earlier, one possible remedy is to project $\mathbf{q}_{k+1}$ onto the unit sphere at each time step by brute-force normalization. At each time step, the following update would be performed:

$$\mathbf{q}_{k+1} \to \mathbf{q}_{k+1}/\|\mathbf{q}_{k+1}\| \tag{8}$$

Clearly, Eq. (8) ensures a normalized quaternion to within machine precision. However, the accuracy of this approach will be studied later.

Table 1. Butcher Table

c_1	a_{11}	a_{12}	$\cdots$	a_{1s}
c_2	a_{21}	a_{22}	$\cdots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\cdots$	a_{ss}
	b_1	b_2	$\cdots$	b_s

As is well-known, a particular RK method is determined by the choice of s and determining the coefficients in such a way as to achieve a desired order of accuracy (the number of stages bounds reachable orders of accuracy). For the RK algorithm, when these *order conditions* are imposed, the coefficients are typically under-determined and many satisfactory solution sets exist. Third and fourth order RK algorithms are considered here. Their coefficients can be conveniently listed in a Butcher table as depicted in Table 1. The coefficients for the three stage, third order Runge-Kutta (RK3) employed in this work are listed in Table 2 and those for the four stage, fourth order Runge-Kutta (RK4) are in Table 3. The brute-force normalized third and fourth order Runge-Kutta methods (RK3n and RK4n, respectively) are the same as the RK3 and RK4 though at each time step the normalization in Eq. (8) is performed.

B. Crouch-Grossman Lie Group Method

As observed above, the RK method updates $\mathbf{q}$ in such a way that it evolves on $\mathbb{R}^4$. Since $\mathbf{q}$ is not arbitrarily defined in $\mathbb{R}^4$, but only on a smooth manifold contained within $\mathbb{R}^4$, namely the unit 3-sphere S^3, one may

Table 2. 3-Stage, Third Order Runge-Kutta (RK3)

0			
1/2	1/2		
1	−1	2	
	1/6	2/3	1/6

Table 3. 4-Stage, Fourth Order Runge-Kutta (RK4)

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

argue that the update of $\mathbf{q}$ should not be made unconstrained in $\mathbb{R}^4$. The idea behind the Crouch-Grossman method suggests that this update instead be performed by the exponential map of the associated Lie algebra.

Now, reconsider the numerical integration of

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q} \tag{9}$$

using instead this Lie group method. The s-stage Crouch-Grossman algorithm is given by[5,10]

$$\mathbf{q}_{k+1} = \exp(\Delta t\, b_s K^{(s)}) \cdots \exp(\Delta t\, b_1 K^{(1)})\mathbf{q}_k \tag{10a}$$

$$K^{(i)} = \frac{1}{2}\Omega(\mathbf{q}^{(i)}, t_k + c_i\Delta t) = \frac{1}{2}\Omega(\boldsymbol{\omega}(t_k + c_i\Delta t)) \tag{10b}$$

$$\mathbf{q}^{(i)} = \exp(\Delta t\, a_{i,i-1} K^{(i-1)}) \cdots \exp(\Delta t\, a_{i,1} K^{(1)})\mathbf{q}_k \tag{10c}$$

Clearly, both $\Delta t\, a_{i,j} K^{(i)}$ and $\Delta t\, b_i K^{(j)}$, for $1 \leq i \leq s$ and $1 \leq j < i$, are elements of the Lie algebra $\mathfrak{su}(2)$. Since $\text{trace}(\Omega) = 0$ it follows from $\det[\exp(\Omega)] = \exp[\text{trace}(\Omega)]$ that $\det[\exp(\Omega)] = 1$. It can also be verified that $\exp(\Omega)\exp(\Omega)^T = \exp(\Omega)^T\exp(\Omega) = I_{4\times 4}$ and thus not only is $\exp(\Omega)$ an orthogonal matrix, but it belongs to SO(4) because of its unit determinant. As a result, since the product of matrices belonging to SO(4) also belongs to SO(4), the algorithm in Eqs. (10a)-(10c) preserves the quaternion unit length and therefore yields a $\mathbf{q}_{k+1}$ that belongs to S^3. In Eq. (10a), $\mathbf{q}_{k+1}$ can be interpreted as being obtained from $\mathbf{q}_k$ by a product of incremental steps along the unit hypersphere. In fact, each increment is equivalent to quaternion multiplication under which the set of unit quaternions is closed. This method preserves the group aspects of quaternions.

At this point, it is useful to note that since the Lie algebra in Eq. (1) does not depend on the state $\mathbf{q}$, the intermediate $\mathbf{q}^{(i)}$ need not be calculated and therefore Eq. (10b) is not needed. Additionally, the matrix exponentials in Eqs. (10a) and (10c) have a closed-form solution given by[7]

$$\exp(\Delta t\, b_j K^{(j)}) = I_{4\times 4}\cos\left(\frac{1}{2}\Delta t\, b_j \|\boldsymbol{\omega}_k^{(j)}\|\right) + \Omega(\boldsymbol{\omega}_k^{(j)})\frac{1}{\|\boldsymbol{\omega}_k^{(j)}\|}\sin\left(\frac{1}{2}\Delta t\, b_j \|\boldsymbol{\omega}_k^{(j)}\|\right) \tag{11}$$

where $\boldsymbol{\omega}_k^{(j)} = \boldsymbol{\omega}(t_k + c_j\Delta t)$. Therefore, no approximation or truncation need be performed in their calculation. The solution in Eq. (11) is obtainable only because the system coefficients are "frozen" at their current values; in other words, Eq. (11) assumes a zero-order hold on system parameters including the angular velocity. Therefore, it is undesirable to apply Eq. (11) directly to Eq. (3) since $\boldsymbol{\omega}$ is time-dependent while a numerical integration algorithm such as the RK or CG approach is better suited as it takes this variation into account.

The order conditions for CG algorithms are more complicated than those of classical RK algorithms. These conditions include those encountered for RK methods as well as additional highly nonlinear conditions.[11] While CG methods enjoy the advantage that their coefficients also construct valid RK methods of

the same order, which can be implemented for additional state variables such as $\boldsymbol{\omega}$, these coefficients are in fact difficult to obtain and much research is dedicated to this objective which is outside the scope of this work.[11,12]

A third order formulation, denoted by CG3, can be achieved using three stages.[5,13] The coefficients for this case[10] are provided in Table 4. Given $\mathbf{q}_k$ and using the CG3 as an example, $\mathbf{q}_{k+1}$ is obtained through

Table 4. 3-Stage CG3

0			
3/4	3/4		
17/24	119/216	17/108	
	13/51	−2/3	24/17

the following calculations:

$$K^{(1)} = \frac{1}{2}\Omega(\boldsymbol{\omega}(t_k)) \tag{12a}$$

$$K^{(2)} = \frac{1}{2}\Omega(\boldsymbol{\omega}(t_k + \frac{3}{4}\Delta t)) \tag{12b}$$

$$K^{(3)} = \frac{1}{2}\Omega(\boldsymbol{\omega}(t_k + \frac{17}{24}\Delta t)) \tag{12c}$$

$$\mathbf{q}_{k+1} = \exp\left(\frac{24}{17}\Delta t\, K^{(3)}\right) \exp\left(-\frac{2}{3}\Delta t\, K^{(2)}\right) \exp\left(\frac{13}{51}\Delta t\, K^{(1)}\right) \mathbf{q}_k \tag{12d}$$

The state-independent Lie algebra is determined for each of the 3 stages in Eqs. (12a)-(12c) and the integrated state $\mathbf{q}_{k+1}$ is determined by a weighted application of their exponential maps using Eq. (12d). Coefficients for a fourth order algorithm, here denoted by CG4, have been determined by Jackiewicz *et. al* using least-squares minimization.[13] These coefficients are provided in Table 5. Note that the fourth order CG algorithm requires five stages ($s = 5$). The third and fourth order CG algorithms consisting of Eq. (10) with the respective coefficients listed in Tables 4 and 5 are implemented in the following section and compared with the above RK methods.

Table 5. 5-Stage CG4[13]

a_{21}	=	0.8177227988124852	b_1	=	0.1370831520630755
a_{31}	=	0.3199876375476427	b_2	=	−0.0183698531564020
a_{32}	=	0.0659864263556022	b_3	=	0.7397813985370780
a_{41}	=	0.9214417194464946	b_4	=	−0.1907142565505889
a_{42}	=	0.4997857776773573	b_5	=	0.3322195591068374
a_{43}	=	−1.0969984448371582	c_1	=	0.0
a_{51}	=	0.3552358559023322	c_2	=	0.8177227988124852
a_{52}	=	0.2390958372307326	c_3	=	0.3859740639032449
a_{53}	=	1.3918565724203246	c_4	=	0.3242290522866937
a_{54}	=	−1.1092979392113565	c_5	=	0.8768903263420429

IV. Approach Comparison

The torque-free motion of some rigid body is considered as a test case. The system state consists of its attitude, which is parameterized by $\mathbf{q}$, and its angular rate, $\boldsymbol{\omega}$. The state equations are then the quaternion

kinematics and the angular dynamics:

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q} \tag{13a}$$

$$J\dot{\boldsymbol{\omega}} = -[\boldsymbol{\omega}\times]J\boldsymbol{\omega} \tag{13b}$$

Despite considering the torque-free case in which the angular rate too belongs to a Lie group, the angular rate is not integrated using the geometric approach. This is because most applications do not conserve angular momentum. In practice, the angular dynamics are typically

$$J\dot{\boldsymbol{\omega}} = -[\boldsymbol{\omega}\times]J\boldsymbol{\omega} + \tau(\mathbf{q}, \boldsymbol{\omega}) \tag{14}$$

which requires a less restrictive approach. Therefore, when the CG approach is employed, the angular rate $\boldsymbol{\omega}$ is integrated using the associated RK algorithm consisting of Eqs. (7a)-(7c) with the CG algorithm coefficients. For example, for the s-stage CG approach, we have the following algorithm

$$\boldsymbol{\omega}^{(1)} = \boldsymbol{\omega}_k \tag{15a}$$

$$\mathbf{k}^{(1)} = -J^{-1}[\boldsymbol{\omega}^{(1)}\times]J\boldsymbol{\omega}^{(1)} \tag{15b}$$

$$K^{(1)} = \frac{1}{2}\Omega(\boldsymbol{\omega}^{(1)}) \tag{15c}$$

$$\vdots$$

$$\boldsymbol{\omega}^{(i)} = \boldsymbol{\omega}_k + \sum_{j=1}^{s-1} a_{ij}\Delta t\, \mathbf{k}^{(j)} \tag{15d}$$

$$\mathbf{k}^{(i)} = -J^{-1}[\boldsymbol{\omega}^{(i)}\times]J\boldsymbol{\omega}^{(i)} \tag{15e}$$

$$K^{(i)} = \frac{1}{2}\Omega(\boldsymbol{\omega}^{(i)}) \tag{15f}$$

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}^{(i)} \tag{15g}$$

$$\mathbf{q}_{k+1} = \exp(\Delta t\, b_s K^{(s)}) \cdots \exp(\Delta t\, b_1 K^{(1)})\mathbf{q}_k \tag{15h}$$

which employs the appropriate CG coefficients. This "mixed" integration scheme is valid since the CG coefficients, as highlighted earlier, along with Eqs. (7a)-(7c) belong to the family of RK algorithms.[5] Unlike in Eq. (6) where the appearance of $\boldsymbol{\omega}$ lead to non-autonomous kinematics in $\mathbf{q}$, the dynamic system in Eq. (13) is autonomous and the time variable does not appear explicitly in the integration algorithms.

The RK algorithms are applied directly to the state

$$\mathbf{x} = \begin{bmatrix}\mathbf{q}^T & \boldsymbol{\omega}^T\end{bmatrix}^T \tag{16}$$

with all appearances of the state $\mathbf{q}$ in Eq. (7) replaced by the appended state $\mathbf{x}$ defined in Eq. (16) and the function $\mathbf{f}$ in Eq. (6) is augmented to

$$\mathbf{f}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}) = \begin{bmatrix}\frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q} \\ -J^{-1}[\boldsymbol{\omega}\times]J\boldsymbol{\omega}\end{bmatrix} \tag{17}$$

A. Simulation Precisions

For simulations, the inertia tensor J, the initial quaternion $\mathbf{q}_0 = \mathbf{q}(0)$ and the initial angular rate $\boldsymbol{\omega}_0 = \boldsymbol{\omega}(0)$ are taken to be

$$J = \mathrm{diag}(200, 200, 100) \equiv \mathrm{diag}(J_T, J_T, J_3) \tag{18a}$$

$$\mathbf{q}_0 = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T \tag{18b}$$

$$\boldsymbol{\omega}_0 = \begin{bmatrix}0.05 & 0 & 0.01\end{bmatrix}^T \tag{18c}$$

The system and initial conditions described by Eqs. (13) and (18) describe torque-free motion of an axisymmetric rigid body.[14] A closed-form solution for both $\mathbf{q}(t)$ and $\boldsymbol{\omega}(t)$ is readily available for this case.[8] The angular rate is given by[8, 14]

$$\omega_1(t) = \omega_{01}\cos(\omega_n t) + \omega_{02}\sin(\omega_n t) \tag{19a}$$

$$\omega_2(t) = \omega_{02}\cos(\omega_n t) - \omega_{01}\sin(\omega_n t) \tag{19b}$$

$$\omega_3(t) = \omega_{03} \tag{19c}$$

where the ω_{0i} are the components of $\boldsymbol{\omega}_0$ and the body nutation rate is

$$\omega_n = \omega_{03}(J_T - J_3)/J_T \tag{20}$$

The solution in Eqs. (19) and (20) is easily obtained since the third component becomes decoupled and the first two components reduce to a simple harmonic oscillator. The quaternion solution can now be expressed as

$$\mathbf{q}(t) = \mathbf{y}(t) \otimes \mathbf{q}_0 \tag{21}$$

where $\mathbf{y}(t)$ is given by[8]

$$\mathbf{y}(t) = \begin{bmatrix} h_{01}\cos(\alpha)\sin(\beta) + h_{02}\sin(\alpha)\sin(\beta) \\ h_{02}\cos(\alpha)\sin(\beta) - h_{01}\sin(\alpha)\sin(\beta) \\ h_{03}\cos(\alpha)\sin(\beta) + \sin(\alpha)\cos(\beta) \\ \cos(\alpha)\cos(\beta) - h_{03}\sin(\alpha)\sin(\beta) \end{bmatrix} \tag{22}$$

with the definitions

$$\alpha = \frac{1}{2}\omega_n t \tag{23a}$$

$$\beta = \frac{1}{2}\omega_i t \tag{23b}$$

$$\mathbf{h}_0 = \mathbf{H}_0/\|\mathbf{H}_0\| \tag{23c}$$

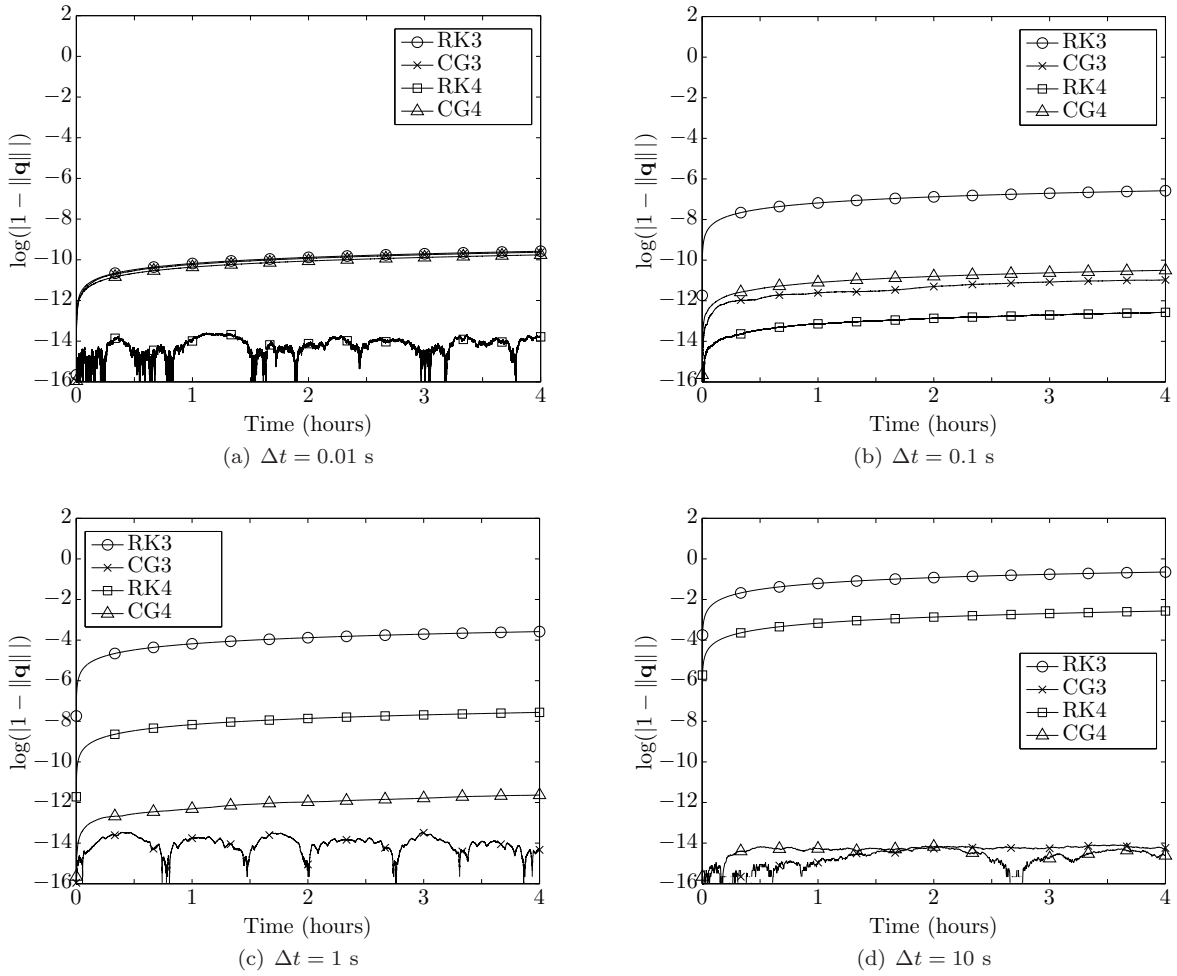
In Eq. (23b), $\omega_i = H/J_T$ is the inertial nutation rate and in (23c), $\mathbf{H}_0 = J\boldsymbol{\omega}_0$ is the initial angular momentum vector. The quaternion solution in Eqs. (21)-(23) can be obtained by applying Floquet theory. Since the system coefficients for the $\mathbf{q}$ state kinematics in Eq. (13a) are P-periodic, where $P = 2\pi/\omega_n$, a fundamental matrix solution can be found using its Floquet representation.[15] This in turn allows explicit calculation of the state transition matrix[16] and we obtain Eqs. (21)-(23).

B. Results

Third and fourth order RK and CG algorithms are used to integrate Eqs. (13) and (18). These algorithms were provided in Section III. The RK algorithms use the coefficients in Tables 2 or 3 and the CG algorithms follow Eq. (15) and use the coefficients in Tables 4 or 5. The norm errors throughout a four-hour time interval are shown in Figure 1 for four different time steps; respectively, $\Delta t = 0.01\,\text{s}$, $0.1\,\text{s}$, $1\,\text{s}$ and $10\,\text{s}$.

Figures 1(a) and 1(b) show that with time-steps of $\Delta t = 0.01\,\text{s}$ and $0.1\,\text{s}$, the RK4 better preserves the quaternion norm over the CG4 by at least 4 orders of magnitude and by approximately 2 orders of magnitude, respectively. For $\Delta t = 0.01\,\text{s}$ in Figure 1(a), the improvement is by "at least" 4 orders of magnitude since machine precision is reached. For these particular simulations, the floating point spacing from one is 2^{-52} which limits the precision to which the norm error can be calculated. However, the CG3 performs comparably with the RK3 algorithm in Figure 1(a) and better than it for $\Delta t = 0.1\,\text{s}$ as is clear in Figure 1(b).

For the larger time-steps of $\Delta t = 1\,\text{s}$ and $10\,\text{s}$, respectively Figures 1(c) and 1(d), significant improvement is seen with both the third and fourth order CG algorithms. In particular, there is a 4 order of magnitude improvement between the RK4 and CG4 algorithms for $\Delta t = 1\,\text{s}$ and over 10 orders of magnitude when $\Delta t = 10\,\text{s}$. Between the CG3 and RK3 algorithms, there is more than 10 orders of magnitude improvement for $\Delta t = 1\,\text{s}$ and more than 12 for $\Delta t = 10\,\text{s}$. It may appear peculiar, especially for $\Delta t = 0.1\,\text{s}$ and $1\,\text{s}$, that the CG3 out performs the CG4 algorithm in Figure 1. However, it is by construction that the CG algorithm maintains the quaternion norm, not by its order of accuracy. The CG4 suffers from considerable more machine precision and round-off error accumulation than the CG3 primarily since the CG4 has five stages compared with three stages for the CG3.



(a) $\Delta t = 0.01$ s (b) $\Delta t = 0.1$ s

(c) $\Delta t = 1$ s (d) $\Delta t = 10$ s

Figure 1. Quaternion Norm Errors Over Time, $t \in [0, 4]$ Hours Using the Third and Fourth Order RK and CG Algorithms.

In general, results from Figure 1 show that as the time step is sufficiently decreased, the classical RK methods maintain the quaternion norm at least as well as the CG algorithms. Though this is observed in practice, it is important to acknowledge that accumulated machine and round-off error plays an important role. The added calculations for the matrix exponentials in Eq. (11) and in the case of the CG4, additional stages, causes error to accumulate more rapidly.

Now the accuracy of the RK and CG algorithms is considered. Using the closed-form solution for Eqs. (13) and (18), the true error in using each algorithm can be computed.[8] Define first the following error quaternion:

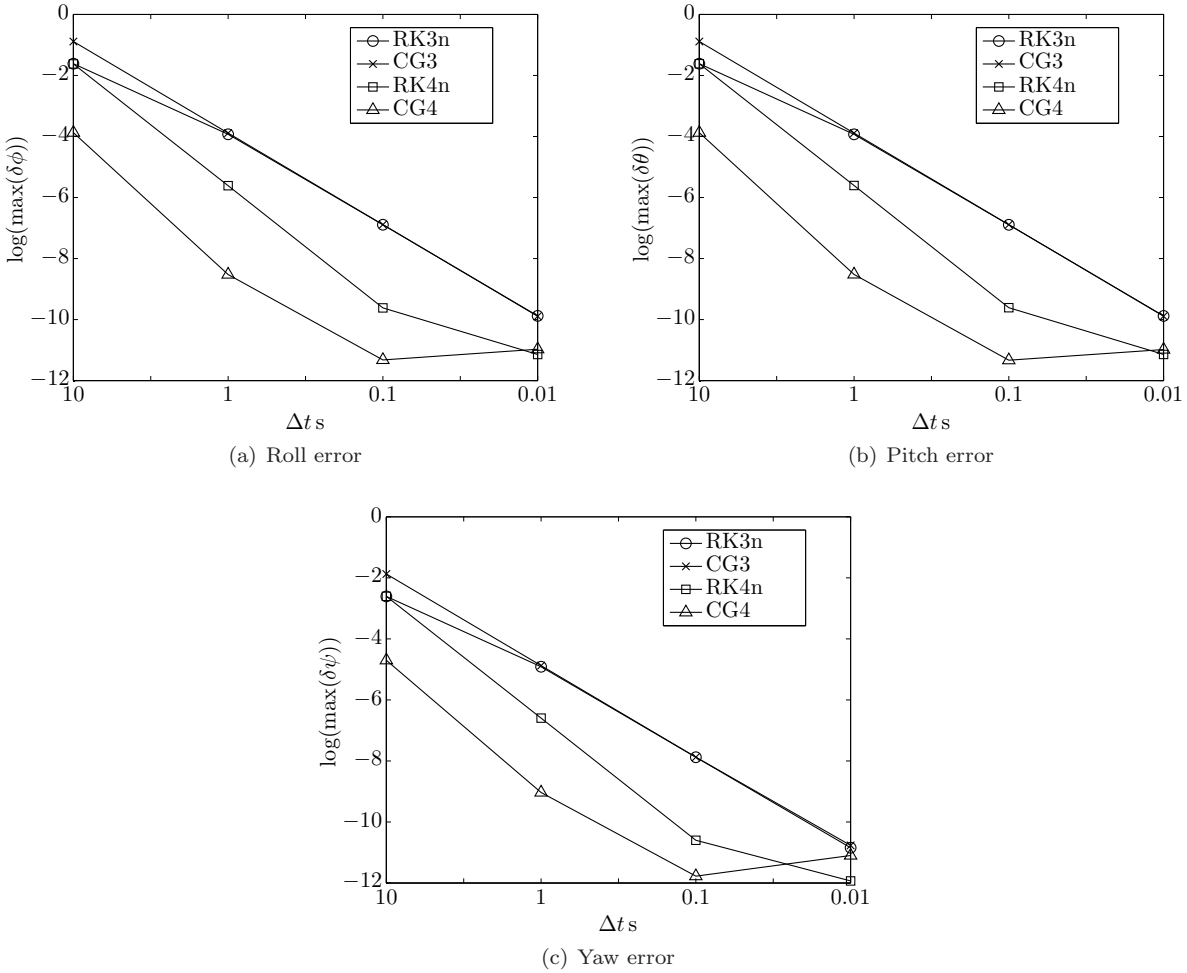
$$\boldsymbol{\delta}\mathbf{q}_k = \mathbf{q}_k \otimes \bar{\mathbf{q}}_k^{-1} \tag{24}$$

where $\mathbf{q}_k$ is the quaternion obtained by numerical integration and $\bar{\mathbf{q}}_k$ is the true quaternion at time $t = k\,\Delta t$ obtained from the closed-form solution in Eqs. (21)-(23). Defining $\boldsymbol{\delta}\mathbf{q} = \begin{bmatrix}\boldsymbol{\delta\varrho}^T & \delta q_4\end{bmatrix}^T$, consider also the small angle approximation: $\boldsymbol{\delta\varrho} \approx \frac{1}{2}\boldsymbol{\delta\alpha}$ where

$$\boldsymbol{\delta\alpha} = \begin{bmatrix}\delta\phi & \delta\theta & \delta\psi\end{bmatrix}^T \tag{25}$$

is the roll, pitch, and yaw error vector for any rotation sequence.[7] The maximum value of each roll, pitch, and yaw angle component obtained during the 4-hour simulation is used as a measurement of error.

No significant difference is observed in the accuracy between the RK3 and RK3n or between the RK4 and RK4n algorithms. This validates the brute-force quaternion normalization that is used in practice, for example in the quaternion-based extended Kalman filter for attitude determination.[7] Therefore, accuracy results are shown only for the RK3n and RK4n approaches and not for their standard counterparts.



(a) Roll error

(b) Pitch error

(c) Yaw error

Figure 2. Max Euler Angle Errors Using the RK3n, CG3, RK4n and CG4 Algorithms.

The maximum angle errors obtained during the 4-hour simulation time interval are shown as a function of Δt for each algorithm in Figure 2. Figures 2(a), 2(b), and 2(c) show respectively the maximum roll, pitch, and yaw angle errors. The advantage in using the CG4 method is that approximately 2 orders of accuracy improvement in the attitude angle errors is observed for all considered time-steps greater than 0.01 s. The CG3 formulation employed yields slightly larger angle errors than the RK3 for $\Delta t = 10$ s but it obtains approximately the same accuracy for the smaller time-steps.

Combining the above observations, the simulations suggest that there is no advantage in using the CG algorithms considered for $\Delta t = 0.01$ s. The CG4 performs worse than the RK4 in terms of both norm preservation and accuracy. While the performance of the CG3 algorithm is about the same as that of the RK3, both in terms of norm preservation and accuracy, the added computational cost is a clear disadvantage. However, for the larger time steps considered, the CG3 has improved norm preservation over the RK3 although without any accuracy improvement. Although, with $\Delta t = 10$ s, CG3 accuracy becomes poorer than that of the RK3. Thus, despite the fact that the third order Crouch-Grossman algorithm showed significant improvement in norm preservation over the classical Runge-Kutta approach for larger time steps, a brute-force normalized Runge-Kutta can achieve this without accuracy degradation. On the other hand, for larger time steps, the CG4 significantly improved accuracy over the RK4, but it improved norm preservation

only in the cases of $\Delta t = 1\,\mathrm{s}$ and $10\,\mathrm{s}$.

V. Conclusion

The application of Crouch-Grossman algorithms to the integration of quaternions was successfully demonstrated. Results obtained using both third and fourth order Crouch-Grossman algorithms were compared with those of the third and fourth order Runge-Kutta methods. Additionally, a Runge-Kutta algorithm with brute-force normalization was employed to determine the effect of projection on result accuracy.

It was found that Crouch-Grossman algorithms did not always outperform classical Runge-Kutta methods as might be expected. In particular, for the smallest time step considered, the third order Crouch-Grossman algorithm yielded no advantages over its Runge-Kutta counterpart to justify the added computational cost; further, the fourth order Crouch-Grossman algorithm performed worse than the fourth order Runge-Kutta method both with respect to norm preservation and accuracy. Additionally, augmenting a Runge-Kutta method did not noticeably affect accuracy. This result validates what is done in practice and serves also to detract from the Crouch-Grossman approach.

A clear advantage was observed in using the fourth order Crouch-Grossman algorithm. For larger time steps, significant improvements in accuracy were observed and the quaternion norm was preserved to within high precision. In the case of the largest time step considered, the quaternion norm was preserved to within machine precision. As a result of these findings, the fourth order Crouch-Grossman algorithm employed in this work has potential in applications that would benefit from larger integration time-steps and where accuracy is important. Propagations over days or weeks can be achieved more rapidly when larger time steps are permitted.

References

[1]Jah, M. and Madler, R., "Satellite Characterization: Angles and Light Curve Data Fusion for Spacecraft State and Parameter Estimation," *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference*, Vol. 49, Sept. 2007, Paper E49.

[2]Linares, R., Crassidis, J. L., Jah, M. K., and Kim, H., "Astrometric and Photometric Data Fusion for Resident Space Object Orbit, Attitude, and Shape Determination Via Multiple-Model Adaptive Estimation," *AIAA Guidance, Navigation, and Control Conference*, Aug. 2010, AIAA-2010-8341.

[3]Kaasalainen, M. and Torppa, J., "Optimization Methods for Asteriod Lightcurve Inversion I: Shape Determination," *Icarus*, Vol. 153, No. 4, Jan. 2001, pp. 24–36.

[4]Kaasalainen, M. and Torppa, J., "Optimization Methods for Asteriod Lightcurve Inversion II: The Complete Inverse Problem," *Icarus*, Vol. 153, No. 4, Jan. 2001, pp. 37–51.

[5]Crouch, P. and Grossman, R., "Numerical Integration of Ordinary Differential Equations on Manifolds," *Journal of Nonlinear Science*, Vol. 3, 1993, pp. 1–33.

[6]Marsden, J. E. and Ratiu, T. S., *Introduction to Mechanics and Symmetry*, Springer, New York, NY, 1999, pp. 302–309.

[7]Crassidis, J. L. and Junkins, J. L., *Optimal Estimation of Dynamic Systems*, CRC Press, Boca Raton, FL, 2nd ed., 2012, pp. 451–460, 611–614.

[8]Markley, F. L., "Attitude Dynamics," *Spacecraft Attitude Determination and Control*, edited by J. R. Wertz, Kluwer Academic Publishers, Dordrecht, Netherlands, 1978, pp. 523–531.

[9]Lapidus, L. and Seinfeld, J. H., *Numerical Solution of Ordinary Differential Equations*, Academic Press, Inc., New York, NY, 1971, pp. 39–50.

[10]Park, J., "Geometric Integration on Euclidean Group with Application to Articulated Multibody Systems," *IEEE Transactions on Robotics*, Vol. 21, No. 5, 2005, pp. 850–863.

[11]Marthinsen, A. and Owren, B., "A Note on the Construction of Crouch-Grossman Methods," *BIT*, Vol. 41, No. 1, 2001, pp. 207–214.

[12]Owren, B. and Marthinsen, A., "Runge-Kutta Methods Adapted to Manifolds and Based on Rigid Frames," *BIT*, Vol. 35, No. 1, 1999, pp. 116–142.

[13]Jackiewicz, Z., Marthinsen, A., and Owren, B., "Construction of Runge-Kutta Methods of Crouch-Grossman Type of High Order," *Advanced Computational Mathematics*, Vol. 13, No. 4, 2000, pp. 405–415.

[14]Thomson, W., *Introduction to Space Dynamics*, Dover, New York, NY, 1961, pp. 113–116.

[15]Hale, J. and Koçak, H., *Dynamics and Bifurcations*, Springer-Verlag, New York, NY, 1991, pp. 256–263.

[16]Chen, C.-T., *Linear System Theory and Design*, Holt, Rinehart, and Winston, New York, NY, 1970, pp. 134–141.