Applied Runge-Kutta-Munthe-Kaas Integration for the Quaternion Kinematics

Aksel Sveier*, Alexander M. Sjøberg †, and Olav Egeland‡
Norwegian University of Science and Technology, 7491 Trondheim, Norway

I. Introduction

The attitude of a body-fixed frame relative to the inertial frame can be represented by a 4-parameter unit quaternion. Attitude representations were surveyed in [1] and the unit quaternion was identified as the recommended parameterization in simulations, as it is easy to normalize, it has few parameters compared to the rotation matrix, and it has a linear kinematic equation of motion. The unit quaternion is commonly used in attitude estimation [2], and is the preferred parameterization in spacecraft attitude control as it has no singular configurations, as opposed to three-parameter representations based on Euler angles and Rodrigues parameters.

To achieve accurate integration of the attitude kinematics over long time intervals, the choice of integration method is important. The unit quaternions form a Lie group under quaternion multiplication and the Lie group integrator of Crouch and Grossmann [3] was compared against classical Runge-Kutta (RK) integration with normalization in [4]. For large time steps it was found that the Crouch-Grossmann (CG) method had clear advantages in terms of accuracy and preservation of the quaternion norm compared with the classical RK method.

The integration scheme of Runge-Kutta-Munthe-Kaas (RKMK) [5] is another type of Lie group integrator that performs the integration based on the differential kinematic equation of the Lie algebra of the Lie group. One advantage of this method is that standard Butcher coefficients [6] of the classical RK methods can be directly applied, while the CG method has additional constraints that have to be accounted for when obtaining the Butcher coefficients.

In this note we formulate the RKMK method for quaternions and perform a comparison to the previously proposed CG methods in a simulation study. In [4], the CG algorithm was applied for quaternions with a matrix Lie group formulation, for both the quaternion kinematics and the quaternion multiplication. This requires the computation of the matrix exponential of 4×4 matrices instead of the quaternion exponential. In the update, this also require multiplication of 4×4 matrices, which requires 64 scalar multiplications and 48 scalar additions, whereas the quaternion product requires 16 multiplications and 12 additions [7]. Since computational cost plays an important role in the choice of integrator, we formulate both the CG and RKMK algorithms in terms of the quaternion product and quaternion exponential to reduce the computational burden of the algorithms. Furthermore, we provide operational counts of the explicit CG and RKMK methods, both in terms of quaternion operations and scalar operations, to enable comparison of their computational cost.

The note is organized as follows: Section II provides the necessary preliminaries, including a Lie group description of quaternions and the quaternion kinematics. Section III introduces the numerical integration methods, which includes the Runge-Kutta method, the Crouch-Grossmann method, and the Runge-Kutta-Munthe-Kaas method. Section IV provides a quantitative description of the computational cost of the techniques covered in Section III. Section V describes the case study and presents the results. Section VI concludes the note.

II. Preliminaries

A. The Lie group SU(2)

A quaternion $\boldsymbol{q}$ can be written as a sum of a scalar $\eta \in \mathbb{R}$ and a three-dimensional vector $\boldsymbol{\sigma} \in \mathbb{R}^3$. This is written

$$\boldsymbol{q} = \eta + \boldsymbol{\sigma} \in \mathbb{H}. \tag{1}$$

where $\mathbb{H}$ is the set of quaternions [8]. The addition and subtraction of two quaternions $\boldsymbol{q}_1 = \eta_1 + \boldsymbol{\sigma}_1$ and $\boldsymbol{q}_2 = \eta_2 + \boldsymbol{\sigma}_2$ is component-wise and given by

$$\boldsymbol{q}_1 \pm \boldsymbol{q}_2 = \eta_1 \pm \eta_2 + \boldsymbol{\sigma}_1 \pm \boldsymbol{\sigma}_2, \tag{2}$$

while the quaternion product is given by

$$\boldsymbol{q}_1 \circ \boldsymbol{q}_2 = \eta_1\eta_2 - \boldsymbol{\sigma}_1 \cdot \boldsymbol{\sigma}_2 + \eta_1\boldsymbol{\sigma}_2 + \eta_2\boldsymbol{\sigma}_1 + \boldsymbol{\sigma}_1 \times \boldsymbol{\sigma}_2. \tag{3}$$

A vector $\boldsymbol{v} \in \mathbb{R}^3$ can be treated as a quaternion with zero scalar part. The quaternion product is then

$$\boldsymbol{q} \circ \boldsymbol{v} = -\boldsymbol{\sigma} \cdot \boldsymbol{v} + \eta\boldsymbol{v} + \boldsymbol{\sigma} \times \boldsymbol{v}. \tag{4}$$

The norm of a quaternion is $\|\boldsymbol{q}\|^2 = \eta^2 + \boldsymbol{\sigma} \cdot \boldsymbol{\sigma}$. The conjugated quaternion is given as $\boldsymbol{q}^* = \eta - \boldsymbol{\sigma}$, which

*Ph.D. Candidate, Department of Mechanical and Industrial Engineering, aksel.sveier@ntnu.no.

†Ph.D. Candidate, Department of Mechanical and Industrial Engineering, alexander.m.sjoberg@ntnu.no.

‡Professor, Department of Mechanical and Industrial Engineering, olav.egeland@ntnu.no.

gives $\boldsymbol{q} \circ \boldsymbol{q}^* = \|\boldsymbol{q}\|^2$. The inverse quaternion $\boldsymbol{q}^{-1}$ satisfies $\boldsymbol{q} \circ \boldsymbol{q}^{-1} = 1$, and it follows that $\boldsymbol{q}^{-1} = \boldsymbol{q}^*/(\|\boldsymbol{q}\|^2)$.

The special unitary group $SU(2)$ is isomorphic to the set of unit quaternions

$$\boldsymbol{q} \in \mathbb{H}^u = \{\boldsymbol{q} \in \mathbb{H} : \|\boldsymbol{q}\|^2 = 1\} \tag{5}$$

and forms a Lie group under quaternion multiplication. A unit quaternion $\boldsymbol{q} \in \mathbb{H}^u$ can be written in terms of an angle θ and a unit vector $\boldsymbol{k}$ as

$$\boldsymbol{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\boldsymbol{k} \in \mathbb{H}^u, \tag{6}$$

where $\eta = \cos\frac{\theta}{2}$ and $\boldsymbol{\sigma} = \sin\frac{\theta}{2}\boldsymbol{k}$. The rotation matrix $\boldsymbol{R} \in SO(3)$ describing the rotation by an angle θ about $\boldsymbol{k}$ is given by $\boldsymbol{R} = \boldsymbol{I} + \sin\theta\hat{\boldsymbol{k}} + (1-\cos\theta)\hat{\boldsymbol{k}}^2$, where $\hat{\boldsymbol{a}}$ is the skew-symmetric form of a vector $\boldsymbol{a}$, so that $\hat{\boldsymbol{a}}\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$. Then it is straightforward to verify that

$$\boldsymbol{R}\boldsymbol{a} = \boldsymbol{q} \circ \boldsymbol{a} \circ \boldsymbol{q}^*, \tag{7}$$

which means that the quaternion $\boldsymbol{q} \in \mathbb{H}^u$ represents the same rotation as the rotation matrix $\boldsymbol{R} \in SO(3)$.

The Lie algebra $su(2)$ of $SU(2)$ is isomorphic to $\mathbb{R}^3$ and the maps between $su(2)$ and $SU(2)$ are given by the exponential and logarithmic maps [9]

$$\boldsymbol{q} = \exp_q\left(\frac{\theta}{2}\boldsymbol{k}\right), \quad \frac{\theta}{2}\boldsymbol{k} = \log_q \boldsymbol{q}. \tag{8}$$

Using $\phi = \theta/2$ the quaternion exponential function is defined by

$$\exp_q(\phi\boldsymbol{k}) = 1 + \phi\boldsymbol{k} + \frac{(\phi\boldsymbol{k})^2}{2!} + \frac{(\phi\boldsymbol{k})^3}{3!} + \dots, \tag{9}$$

where $(\cdot)^n$ denotes the quaternion product of order n. By using $\boldsymbol{k}^2 = \boldsymbol{k} \circ \boldsymbol{k} = -\boldsymbol{k} \cdot \boldsymbol{k} = -1$ it can be seen that

$$\begin{aligned}\exp_q(\phi\boldsymbol{k}) &= \left(1 - \frac{\phi^2}{2!} + \frac{\phi^4}{4!}\dots\right) + \left(\phi - \frac{\phi^3}{3!} + \frac{\phi^5}{5!}\dots\right)\boldsymbol{k} \\ &= \cos\phi + \sin\phi\boldsymbol{k}.\end{aligned} \tag{10}$$

If a logarithm $\boldsymbol{u} \in \mathbb{R}^3$ is given, then the exponential can be computed by

$$\exp_q \boldsymbol{u} = \cos\|\boldsymbol{u}\| + \mathrm{sinc}\|\boldsymbol{u}\|\boldsymbol{u}, \tag{11}$$

where $\mathrm{sinc}x = \sin x/x$. The Taylor series expansion for $\mathrm{sinc}x$ around $x = 0$ is

$$\mathrm{sinc}x = \frac{\sin x}{x} = \sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!} \tag{12}$$

and shows that $\mathrm{sinc}x$ is well defined and smooth for all x. The logarithmic map of $\boldsymbol{q} = \eta + \boldsymbol{\sigma} \in \mathbb{H}^u$ is computed as

$$\log_q \boldsymbol{q} = \frac{\arcsin\|\boldsymbol{\sigma}\|}{\|\boldsymbol{\sigma}\|}\boldsymbol{\sigma}. \tag{13}$$

This is a smooth mapping for all unit quaternions, since $\|\boldsymbol{\sigma}\| \in [0,1]$ and $\arcsin x/x$ is well defined and smooth for all x, which can be seen from the Taylor series expansion

$$\frac{\arcsin x}{x} = 1 + \frac{x^2}{6} + \frac{3x^4}{40} + \frac{5x^6}{112}\dots. \tag{14}$$

The kinematic differential equation for a unit quaternion $\boldsymbol{q}_{IB} \in \mathbb{H}^u$ describing the orientation of the body frame B relative to the inertial frame I is given by

$$\dot{\boldsymbol{q}}_{IB} = \frac{1}{2}\omega^I \circ \boldsymbol{q}_{IB} = \frac{1}{2}\boldsymbol{q}_{IB} \circ \omega^B. \tag{15}$$

Here, $\omega \in \mathbb{R}^3$ is the angular velocity given in the indicated frame and can be treated as a quaternion where the scalar component is zero. All algorithms considered in this note assumes the angular velocity to be given in the body frame and we will in the following discard the superscript and subscripts. It is noted that by changing the order of quaternion multiplication, the corresponding algorithms with the angular velocity given in the inertial frame can be obtained.

B. Inverse Right and Left Jacobian for Quaternions

Consider the kinematic differential equation of the unit quaternion

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \circ \omega. \tag{16}$$

The logarithm of the quaternion is $\boldsymbol{u} = \log_q \boldsymbol{q} = \frac{\theta}{2}\boldsymbol{k}$. The corresponding rotation matrix $\boldsymbol{R} \in SO(3)$ has the logarithm $\boldsymbol{v} = \theta\boldsymbol{k}$, which satisfies the kinematic differential equation [10]

$$\dot{\boldsymbol{v}} = \boldsymbol{\Psi}_{R,r}^{-1}(\boldsymbol{v})\omega, \tag{17}$$

where $\boldsymbol{\Psi}_{R,r}^{-1}(\boldsymbol{v})$ is the inverse right Jacobian of the logarithm $\boldsymbol{v}$. Since $\boldsymbol{v} = 2\boldsymbol{u}$ the corresponding expression for the quaternion logarithm is

$$\dot{\boldsymbol{u}} = \boldsymbol{\Psi}_{q,r}^{-1}(\boldsymbol{u})\omega, \tag{18}$$

where $\boldsymbol{\Psi}_{q,r}^{-1}(\boldsymbol{u}) = \frac{1}{2}\boldsymbol{\Psi}_{R,r}^{-1}(2\boldsymbol{u})$ is the inverse right Jacobian of the logarithm $\boldsymbol{u}$. The inverse right Jacobian $\boldsymbol{\Psi}_{R,r}^{-1}(\boldsymbol{v})$ has the following closed form solution [10, 11]

$$\boldsymbol{\Psi}_{R,r}^{-1}(\boldsymbol{v}) = \boldsymbol{I} + \frac{1}{2}\hat{\boldsymbol{v}} + \frac{1 - \frac{\|\boldsymbol{v}\|}{2}\cot\frac{\|\boldsymbol{v}\|}{2}}{\|\boldsymbol{v}\|^2}\hat{\boldsymbol{v}}^2, \tag{19}$$

where $\boldsymbol{I}$ denotes the identity matrix of appropriate dimensions. It follows that the inverse right Jacobian $\boldsymbol{\Psi}_{q,r}^{-1}(\boldsymbol{u})$ is given as

$$\boldsymbol{\Psi}_{q,r}^{-1}(\boldsymbol{u}) = \frac{1}{2}\left(\boldsymbol{I} + \hat{\boldsymbol{u}} + \frac{1 - \|\boldsymbol{u}\|\cot\|\boldsymbol{u}\|}{\|\boldsymbol{u}\|^2}\hat{\boldsymbol{u}}^2\right). \tag{20}$$

If the angular velocity is defined in the inertial frame, then $\dot{\boldsymbol{u}} = \boldsymbol{\Psi}_{q,l}^{-1}(\boldsymbol{u})\omega$ where $\boldsymbol{\Psi}_{q,l}^{-1}(\boldsymbol{u})$ is the inverse left Jacobian for $\boldsymbol{u}$. The closed form of the inverse left Jacobian can be obtained from the closed form of the inverse right Jacobian by applying the property $\boldsymbol{\Psi}_{q,l}^{-1}(\boldsymbol{u}) = [\boldsymbol{\Psi}_{q,r}^{-1}(\boldsymbol{u})]^T$ [10], such that

$$\boldsymbol{\Psi}_{q,l}^{-1}(\boldsymbol{u}) = \frac{1}{2}\left(\boldsymbol{I} - \hat{\boldsymbol{u}} + \frac{1 - \|\boldsymbol{u}\|\cot\|\boldsymbol{u}\|}{\|\boldsymbol{u}\|^2}\hat{\boldsymbol{u}}^2\right). \tag{21}$$

For details about the right and left Jacobians, see [10–12].

C. Introductory Example: Euler's method

Suppose that the quaternion at time step t_k is $\boldsymbol{q}_k$, and that the angular velocity is ω_k. Let the time step be h so that the next time step is $t_{k+1} = t_k + h$. Then the quaternion at time t_{k+1} can be evaluated with Euler's method for (16), which gives the additive update

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \frac{h}{2}(\boldsymbol{q}_k \circ \omega_k). \tag{22}$$

This should be combined with a renormalization $\boldsymbol{q}_{k+1} := \boldsymbol{q}_{k+1}/\|\boldsymbol{q}_{k+1}\|$ to ensure that the result is a unit quaternion.

A modification to Euler's method is to use the multiplicative update

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k \circ \exp_q\left(\frac{h\omega_k}{2}\right). \tag{23}$$

This ensures that result is a unit quaternion and it will become apparent in Sect. III.C that this is equivalent to the Crouch-Grossmann method of order 1.

A further development is to find the local change in attitude described by the logarithm $\Delta\boldsymbol{u}$ and then perform the multiplicative update according to the exponential mapping of the change. Using Euler's method on the dynamics of the logarithm (18) gives the update equation

$$\Delta\boldsymbol{u}_{k+1} = \Delta\boldsymbol{u}_k + h\boldsymbol{\Psi}_{q,r}^{-1}(\Delta\boldsymbol{u}_k)\omega_k. \tag{24}$$

Since the change in attitude is considered, the term $\Delta\boldsymbol{u}_k$ is by definition reset to $\boldsymbol{0}$ at each new time step. The propagated logarithm is injected into the state as

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k \circ \exp_q\left(\Delta\boldsymbol{u}_{k+1}\right). \tag{25}$$

When (24) is used to update the change, then this corresponds to the RKMK method of order 1, which is seen to be equivalent to (22) since $\boldsymbol{\Psi}_{q,r}^{-1}(\Delta\boldsymbol{u}_k = \boldsymbol{0}) = (1/2)\boldsymbol{I}$.

III. Numerical Integration Methods

In this section we present Lie group integration methods formulated in for quaternions to perform the integration of the quaternion kinematic differential equation in (16). We start by reviewing the classical Runge-Kutta (RK) methods, which forms the basis for the Lie group method of Runge-Kutta-Munthe-Kaas (RKMK). We also review the Lie group approach of Crouch-Grossman (CG) formulated in terms of the quaternion product.

A. The RK Algorithm

The RK method with s stages for the differential equation

$$\dot{\boldsymbol{y}} = f(\boldsymbol{y}, t) \tag{26}$$

over a time step h from $\boldsymbol{y}_k$ to $\boldsymbol{y}_{k+1}$, is given by

$$\left.\begin{aligned} \boldsymbol{y}^{(i)} &= \boldsymbol{y}_k + \textstyle\sum_{j=1}^{s} a_{i,j}\boldsymbol{k}^{(j)} \\ \boldsymbol{k}^{(i)} &= hf(\boldsymbol{y}^{(i)}, t_k + c_i h), \end{aligned}\right\} i = 1, \ldots, s, \tag{27}$$
$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \sum_{j=1}^{s} b_j\boldsymbol{k}^{(i)}.$$

Each stage of the explicit RK method is related to a set of parameters $a_{i,j}$, b_i and c_i, which are arranged in a Butcher table on the form given in Table 1. Note that the empty positions are defined as zero.

Table 1 The explicit Butcher table

c_1					
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
$\vdots$	$\vdots$		$\ddots$		
c_s	$a_{s,1}$	$a_{s,2}$	$\ldots$	$a_{s,s-1}$	
	b_1	b_2	$\ldots$	b_{s-1}	b_s

When the quaternion kinematic differential equation (16) is considered, an additional renormalization $\boldsymbol{q}_{k+1} := \boldsymbol{q}_{k+1}/||\boldsymbol{q}_{k+1}||$ must be added to (27) to ensure that the result is a unit quaternion. This approach was studied in [4] and showed similar performance to the CG method when small time steps were considered.

The Butcher tables for the 3-stage RK method of order 3 (RK3), the 4-stage RK method of order 4 (RK4) and the 6-stage RK method of order 5 (RK5) are given Table 2 [6].

B. The RKMK Algorithm on SU(2)

The RKMK algorithm [5] is based on the integration of (18). This means that the method performs the integration on the Lie algebra of the Lie group being considered, which in this case is $su(2)$. We formulate the algorithm in terms of quaternions, quaternion multiplication, quaternion exponential and the inverse right Jacobian for the quaternion logarithm, which means that the angular velocity is

Table 2 The Butcher tables for explicit Runge-Kutta methods

(a) 3-stage RK3

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	-1	2	
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$

(b) 4-stage RK4

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

(c) 6-stage RK5

0						
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$				
$\frac{1}{2}$	0	0	$\frac{1}{2}$			
$\frac{3}{4}$	$\frac{3}{16}$	$-\frac{3}{8}$	$\frac{3}{8}$	$\frac{9}{16}$		
1	$-\frac{3}{7}$	$\frac{8}{7}$	$\frac{6}{7}$	$-\frac{12}{7}$	$\frac{8}{7}$	
	$\frac{7}{90}$	0	$\frac{32}{90}$	$\frac{12}{90}$	$\frac{32}{90}$	$\frac{7}{90}$

considered in the body frame. The Butcher parameters of the RKMK method are obtained from the Butcher tables for standard RK methods as given in Table 2. The RKMK algorithm with s stages is given as

$$
\left.
\begin{aligned}
\Theta^{(i)} &= \textstyle\sum_{j=1}^{s} a_{ij}\boldsymbol{F}^{(j)} \\
\boldsymbol{\theta}^{(i)} &= h\boldsymbol{\omega}\left(\boldsymbol{q}_k \circ \exp_q\left(\Theta^{(i)}\right), t_k + c_i h\right) \\
\boldsymbol{F}^{(i)} &= \boldsymbol{\Psi}_{q,r}^{-1}(\Theta^{(i)})\boldsymbol{\theta}^{(i)}
\end{aligned}
\right\} i = 1, \dots, s,
$$
$$
\begin{aligned}
\Theta &= \sum_{j=1}^{s} b_j \boldsymbol{F}^{(j)} \\
\boldsymbol{q}_{k+1} &= \boldsymbol{q}_k \circ \exp_q(\Theta).
\end{aligned}
\tag{28}
$$

Note that for explicit methods, $\Theta^{(1)} = \mathbf{0}$ in the first iteration. This implies that the Jacobian in the computation of $\boldsymbol{F}^{(1)}$ evaluates to $1/2\boldsymbol{I}$, resulting in $\boldsymbol{F}^{(1)} = 1/2\boldsymbol{\theta}^{(1)}$. Consequently, the explicit RKMK with s stages requires the computation of $s-1$ inverse right Jacobians. It also requires s quaternion exponential operations and s quaternion multiplication operations when the angular velocity dynamics depend on the orientation. This reduces to 1 quaternion exponential operation and 1 quaternion multiplication operation when the angular velocity dynamics are independent of the orientation. In terms of memory it can be seen that only the $\boldsymbol{F}^{(i)}$ vector needs to be stored from each stage, resulting in $3s$ units of memory requirement. Counts of quaternion and matrix operations for the 3-stage RKMK method of order 3 (RKMK3), the 4-stage RKMK method of order 4 (RKMK4) and the 6-stage RKMK method of order 5 (RKMK5) are shown in Table 3.

C. The CG Algorithm

The integration method of Crouch and Grossman [3] performs the update in each stage such that the intermediate results are on the Lie group, as opposed to the RKMK method where the intermediate results are in terms of the logarithm. We state the algorithm in terms of the quaternion product and quaternion exponential, where the angular velocity is given in the body frame. The CG algorithm with s stages for the differential equation in (15) is given as

$$
\left.
\begin{aligned}
\boldsymbol{q}^{(i)} &= \boldsymbol{q}_k \circ \exp_q\left(a_{i,i-1}\boldsymbol{F}^{(i-1)}\right) \\
&\qquad \circ \dots \circ \exp_q\left(a_{i,1}\boldsymbol{F}^{(1)}\right) \\
\boldsymbol{F}^{(i)} &= h\tfrac{1}{2}\boldsymbol{\omega}\left(\boldsymbol{q}^{(i)}, t_k + c_i h\right)
\end{aligned}
\right\} i = 1, \dots, s,
$$
$$
\boldsymbol{q}_{k+1} = \boldsymbol{q}_k \circ \exp_q\left(b_1\boldsymbol{F}^{(1)}\right) \circ \dots \circ \exp_q\left(b_s\boldsymbol{F}^{(s)}\right). \tag{29}
$$

This approach preserves the Lie group conditions for the quaternions. The Butcher coefficients and the order of the methods are not trivial to obtain. Moreover, minimization techniques must be applied to determine the Butcher coefficients needed to use a higher order CG algorithm. The CG algorithm of order 4 (CG4) requires $s = 5$ stages and the Butcher table for this algorithm was determined in [13] and is given in Table 4 together with the Butcher table for the 3-stage CG algorithm of order 3 (CG3). For the 9-stage CG algorithm of order 5 (CG5), the reader is referred to [13] for a complete table of Butcher coefficients.

It can be seen from (29) that the number of quaternion exponential operations is $\sum_{i=1}^{s} i$, and the number of quaternion multiplications is also $\sum_{i=1}^{s} i$. If the angular velocity dynamics is independent of orientation, the number of quaternion exponential and quaternion multiplication operations reduces to s. Similar to the RKMK algorithm, only the $\boldsymbol{F}^{(i)}$ vectors needs to be stored for each stage, resulting in $3s$ units of memory requirement. The counts of quaternion and matrix operations for the explicit CG3, CG4 and CG5 algorithms are shown in Table 3

IV. Computational Cost and Approximate Solutions

To determine the computational cost of the considered integrators we provide counts for the number of mathe-

Table 3 Quaternion and matrix operations for integrator methods. ($\omega = \omega(\boldsymbol{q}, t) \mid \omega = \omega(t)$)

	Stages	$\exp_q(\cdot)$	$\mathbb{H}^u \circ \mathbb{H}^u$	$\Psi^{-1}_{q,r}(\cdot)$	$\mathbb{R} \pm \mathbb{R}$	$\mathbb{R}^3 \pm \mathbb{R}^3$	$\mathbb{R} \times \mathbb{R}$	$\mathbb{R} \times \mathbb{R}^3$	$\mathbb{R}^{3\times3} \times \mathbb{R}^3$
CG3	3	6 | 3	6 | 3	0	2	0	3	9 | 6	0
CG4	5	15 | 5	15 | 5	0	4	0	5	20 | 10	0
CG5	9	45 | 9	45 | 9	0	8	0	9	54 | 18	0
RKMK3	3	3 | 1	3 | 1	2	2	3	2	6	2
RKMK4	4	4 | 1	4 | 1	3	3	3	3	7	3
RKMK5	6	6 | 1	6 | 1	5	5	12	5	19	5

Table 4 Butcher tables for Crouch-Grossman methods

(a) 3-stage CG3

$$\begin{array}{c|ccc} 0 & & & \\ \frac{3}{4} & \frac{3}{4} & & \\ \frac{17}{24} & \frac{119}{216} & \frac{17}{108} & \\ \hline & \frac{13}{51} & -\frac{2}{3} & \frac{24}{17} \end{array}$$

(b) 5-stage CG4

$$\begin{aligned}
a_{21} &= 0.8177227988124852 & a_{31} &= 0.3199876375476427 \\
a_{32} &= 0.0659864263556022, & a_{41} &= 0.9214417194464946 \\
a_{42} &= 0.4997857776773573 & a_{43} &= -1.0969984448371582 \\
a_{51} &= 0.3552358559023322 & a_{52} &= 0.2390958372307326 \\
a_{53} &= 1.3918565724203246 & a_{54} &= -1.1092979392113465 \\
b_1 &= 0.1370831520630755 & c_1 &= 0.0 \\
b_2 &= -0.0183698531564020 & c_2 &= 0.8177227988124852 \\
b_3 &= 0.7397813985370780 & c_3 &= 0.3859740639032449 \\
b_4 &= -0.1907142565505889 & c_4 &= 0.3242290522866937 \\
b_5 &= 0.3322195591068374 & c_5 &= 0.8768903263420429
\end{aligned}$$

matical operations needed in their implementation. This gives a more deterministic measure than trying to time the integrators in a computer program, as the run-time of the integrators may vary depending on the implementation and the system specifications. The operation counts for the exponential function and inverse right Jacobian are presented in detail, and we show how the inverse right Jacobian can be approximated by its Taylor series expansion to reduce the number of operations. In the simulation section we will also show that this approximation has minimal effect on the resulting accuracy for the simulation case considered.

The mathematical operations are grouped into scalar additions and subtractions (A), scalar multiplications (M), scalar divisions (D), scalar square root operations (Sqrt) and trigonometric function calls (F).

A. Operation counts for the quaternion exponential

The closed form solution is computed as given in (11). The operation $||\boldsymbol{u}|| = \sqrt{\boldsymbol{u}^T\boldsymbol{u}}$ is precomputed and requires 1 square root, 3 multiplications, 2 additions and 1 unit of memory for temporary storage. The result $\boldsymbol{q} = \cos||\boldsymbol{u}|| + \operatorname{sinc}(||\boldsymbol{u}||)\boldsymbol{u}$ requires 2 function calls and 3 multiplications. The total cost for the quaternion exponential is thus 2 F, 1 Sqrt, 6 M and 2 A.

B. Operation counts for the inverse Right Jacobian

The inverse right Jacobian given in (20) can be approximated about the origin by using the following 3rd order Taylor expansion

$$\boldsymbol{\Psi}^{-1}_{q,r}(\boldsymbol{u}) = \frac{1}{2}\left(\boldsymbol{I} + \hat{\boldsymbol{u}} + \left(\frac{1}{3} + \frac{||\boldsymbol{u}||^2}{45}\right)\hat{\boldsymbol{u}}^2 + O(||\cdot||^3)\right). \tag{30}$$

A similar approximation is found in [14], but for $\boldsymbol{\Psi}^{-1}_{R,r}$ as a solution to the Bortz equation for small rotations [15]. This expression eliminates the need of a function call, i.e. the trigonometric cotangent function, as well as a square-root operation if the higher order terms are neglected. By comparing the approximation in (30) with the closed form in (20) it can be seen that they only differ in the expression for the scalar that is multiplied with the squared skew-symmetric matrix. By defining $\gamma_1 = (1 - ||\boldsymbol{u}|| \cot ||\boldsymbol{u}||)/||\boldsymbol{u}||^2$ and $\gamma_2 = 1/3 + ||\boldsymbol{u}||^2/45$ the approximation error between the exact and approximated inverse right Jacobian can be analyzed. The approximation error of γ_2 is plotted in Fig. 1 for an increasing value of $||\boldsymbol{u}||$.

To determine the operation counts for the two versions of the inverse right Jacobian, we start by evaluating the expression

$$\frac{1}{2}\left(\boldsymbol{I} + \hat{\boldsymbol{u}} + \gamma_i \hat{\boldsymbol{u}}^2\right). \tag{31}$$

We find that $\hat{\boldsymbol{u}}$ requires no computation, the squared skew symmetric matrix $\hat{\boldsymbol{u}}^2$ requires 6 unique multiplications, 3 unique additions and 6 units of memory for temporary storage. The term $\gamma_i \hat{\boldsymbol{u}}^2$ then requires 6 multiplications. Finally, we find that the computation of (31) requires 9 additions and 3 multiplications, resulting in a total cost of of 15 M and 12 A, plus the computational cost of γ_1 and γ_2.

First, we evaluate the computation of γ_1. Computing and storing $||\boldsymbol{u}||^2 = \boldsymbol{u}^T\boldsymbol{u}$ requires 3 multiplications, 2 additions, and 1 unit of memory. The computation of $||\boldsymbol{u}|| = \sqrt{||\boldsymbol{u}||^2}$ requires 1 square-root operation and 1 unit of memory. The term $1 - ||\boldsymbol{u}|| \cot(||\boldsymbol{u}||)$ requires 1 function call, 1 addition, 1 multiplication and 1 unit of memory. Finally, $\gamma_1 = (1 - ||\boldsymbol{u}|| \cot(||\boldsymbol{u}||))/||\boldsymbol{u}||^2$, which requires 1 division. This leave the total computational cost of γ_1 as 1 F, 1 Sqrt, 1 D, 4 M and 3 A. For the evaluation of γ_2, we use that $||\boldsymbol{u}||^2$ require 3 multiplications, 2 additions, and 1 unit of memory. Then, $\gamma_2 = 1/3 + ||\boldsymbol{u}||^2/45$ which require 1 multiplication and 1 addition, leaving the total computational cost of γ_2 as 4 M and 3 A. The operation counts for (20) and (30) are summarized in Table 5 for ease of comparison.

Table 5 Operation counts for the exact and the approximate inverse right Jacobian.

Method	F	Sqrt	D	M	A
Closed form (γ_1)	1	1	1	19	15
Taylor 3rd order (γ_2)	0	0	0	19	15

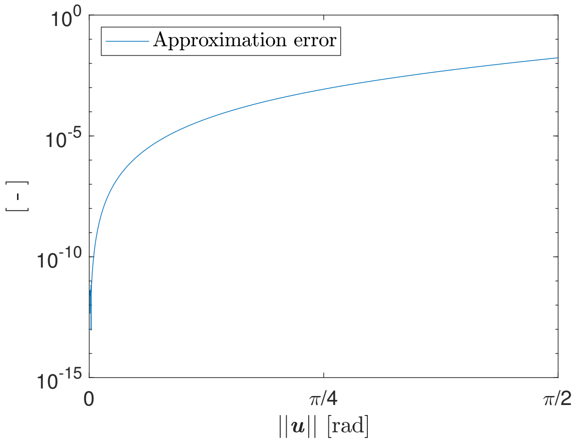


Fig. 1 Logarithmic plot of the error between the exact and approximated inverse right Jacobian.

C. Total operation counts

In addition to quaternion exponential, quaternion multiplication (16 M and 12 A) and right inverse Jacobians, the RKMK and CG methods require scalar additions (1 A), vector additions (3 A), scalar multiplications (1 M), scalar-vector multiplications (3 M) and matrix-vector multiplications (9 M and 6 A). The number of these operations for each of the considered methods are listed in Table 3. Note that we have taken into account the Butcher coefficients when these operations are counted, i.e. multiplication with 0 and ± 1 are not counted. Moreover, the time step h is considered to be multiplied with the factor $\frac{1}{2}$ from the inverse right Jacobian in (28), such that a scalar multiplication is counted instead of a scalar-vector multiplication.

Since we now have determined the operation counts for each of the operations listed in Table 3 we can determine the total computational cost of each of the methods. Table 6 shows the computational cost when the angular velocity is a function of orientation while Table 7 shows the computational cost when the angular velocity is independent of orientation. The values enclosed by parenthesis denotes the computational cost when the approximation of the inverse right Jacobian is considered.

When the angular velocity is a function of orientation it can be seen that the RKMK methods require less function calls, square roots and multiplications than the CG methods of equivalent order. Interestingly, the RKMK method of order 5 requires fewer function calls, square roots and multiplications than the CG method of order 4. Since divisions and additions are cheaper to compute than square roots and trigonometric function calls, it can be argued that RKMK5 is less computational costly than CG4.

When the angular velocity is independent of orientation RKMK3 requires fewer function calls than CG3 and the same amount of square root operations, while CG3 requires fewer divisions, multiplications and additions. For RKMK4 and CG4 it can be seen that RKMK4 requires fewer function calls, square roots and fewer multiplications than CG4 and it can be argued that the total computational cost of RKMK4 is lower than for CG4. The RKMK5 method requires 3 less trigonometric function calls than CG4, however, the amount of multiplications and additions is significantly higher for RKMK5.

V. Simulations and Results

In order to verify the RKMK methods and compare the results with the CG methods, we use the same simulation example as in [4], using the following dynamic system

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \circ \boldsymbol{\omega} \tag{32}$$

$$\dot{\boldsymbol{\omega}} = -\boldsymbol{J}^{-1}\hat{\boldsymbol{\omega}}\boldsymbol{J}\boldsymbol{\omega}, \tag{33}$$

Table 6 Total computational cost for integrator methods for $\omega = \omega(\boldsymbol{q}, t)$.

	F	Sqrt	D	M	A
CG3	12	6	0	162	86
CG4	30	15	0	395	214
CG5	90	45	0	1161	638
RKMK3	8 (6)	5 (3)	2 (0)	142	95
RKMK4	11 (8)	7 (4)	3 (0)	196	131
RKMK5	17 (12)	11 (6)	5 (0)	334	230

Table 7 Total computational cost for integrator methods for $\omega = \omega(t)$.

	F	Sqrt	D	M	A
CG3	6	3	0	87	44
CG4	10	5	0	145	74
CG5	18	9	0	261	134
RKMK3	4 (2)	3 (1)	2 (0)	98	67
RKMK4	5 (2)	4 (1)	3 (0)	130	89
RKMK5	7 (2)	6 (1)	5 (0)	224	160

with the inertial tensor and initial conditions

$$\boldsymbol{J} = \mathrm{diag}(J_t, J_t, J_a) = \mathrm{diag}(200, 200, 100) \tag{34}$$

$$\boldsymbol{q}(t_0) = [1 \;\; 0 \;\; 0 \;\; 0]^T \tag{35}$$

$$\boldsymbol{\omega}(t_0) = [0.05 \;\; 0 \;\; 0.01]^T, \tag{36}$$

where $\boldsymbol{\omega}(t) = [\omega_x(t) \quad \omega_y(t) \quad \omega_z(t)]^T$. Conveniently, a closed form solution for the angular velocity is given by

$$\boldsymbol{\omega}(t) = \begin{bmatrix} \omega_x(t_0)\cos(\omega_n t) + \omega_x(t_0)\sin(\omega_n t) \\ \omega_y(t_0)\cos(\omega_n t) - \omega_x(t_0)\sin(\omega_n t) \\ \omega_z(t_0) \end{bmatrix}, \tag{37}$$

where

$$\omega_n = \omega_z(t_0)\frac{J_t - J_a}{J_t}. \tag{38}$$

Because the angular velocity has a closed form solution, the quaternion at time t is given by

$$\boldsymbol{q}(t) = \boldsymbol{q}(t_0) \circ \boldsymbol{y}(t), \tag{39}$$

where

$$\boldsymbol{y}(t) = \cos\alpha\cos\beta - h_z(t_0)\sin\alpha\sin\beta + \begin{bmatrix} h_x(t_0)\cos\alpha\sin\beta + h_y(t_0)\sin\alpha\sin\beta \\ h_y(t_0)\cos\alpha\sin\beta - h_x(t_0)\sin\alpha\sin\beta \\ h_z(t_0)\cos\alpha\sin\beta + \sin\alpha\sin\beta \end{bmatrix} \tag{40}$$

and

$$\alpha = \frac{\omega_n t}{2} \tag{41}$$

$$\beta = \frac{\omega_i t}{2} \tag{42}$$

$$\boldsymbol{h}(t_0) = \frac{\boldsymbol{H}(t_0)}{||\boldsymbol{H}(t_0)||}, \tag{43}$$

where $\boldsymbol{H}(t_0) = \boldsymbol{J}\boldsymbol{\omega}(t_0)$ and $\omega_i = ||\boldsymbol{H}(t_0)||/J_t$.

In the simulations of the numerical methods, the angular velocity is integrated using the standard RK algorithm in (27) with the same order and Butcher coefficients as in the CG and RKMK algorithms. This is valid since the Butcher coefficients of both the CG and the RKMK algorithms give valid RK algorithms of the same order. This mixed integration scheme is shown in detail in [4].

The error quaternion $\delta\boldsymbol{q}(t_k) = \delta\eta(t_k) + \delta\boldsymbol{\sigma}(t_k)$ between the closed form solution $\boldsymbol{q}(t_k)$ and the integrator result $\tilde{\boldsymbol{q}}(t_k)$ at time $t_k = kh$ was calculated as

$$\delta\boldsymbol{q}(t_k) = \tilde{\boldsymbol{q}}(t_k) \circ \boldsymbol{q}(t_k)^*. \tag{44}$$

Similar to [4], the small angle approximation is used

$$2\delta\boldsymbol{\sigma} \approx \begin{bmatrix} \delta\phi & \delta\vartheta & \delta\varphi \end{bmatrix}, \tag{45}$$

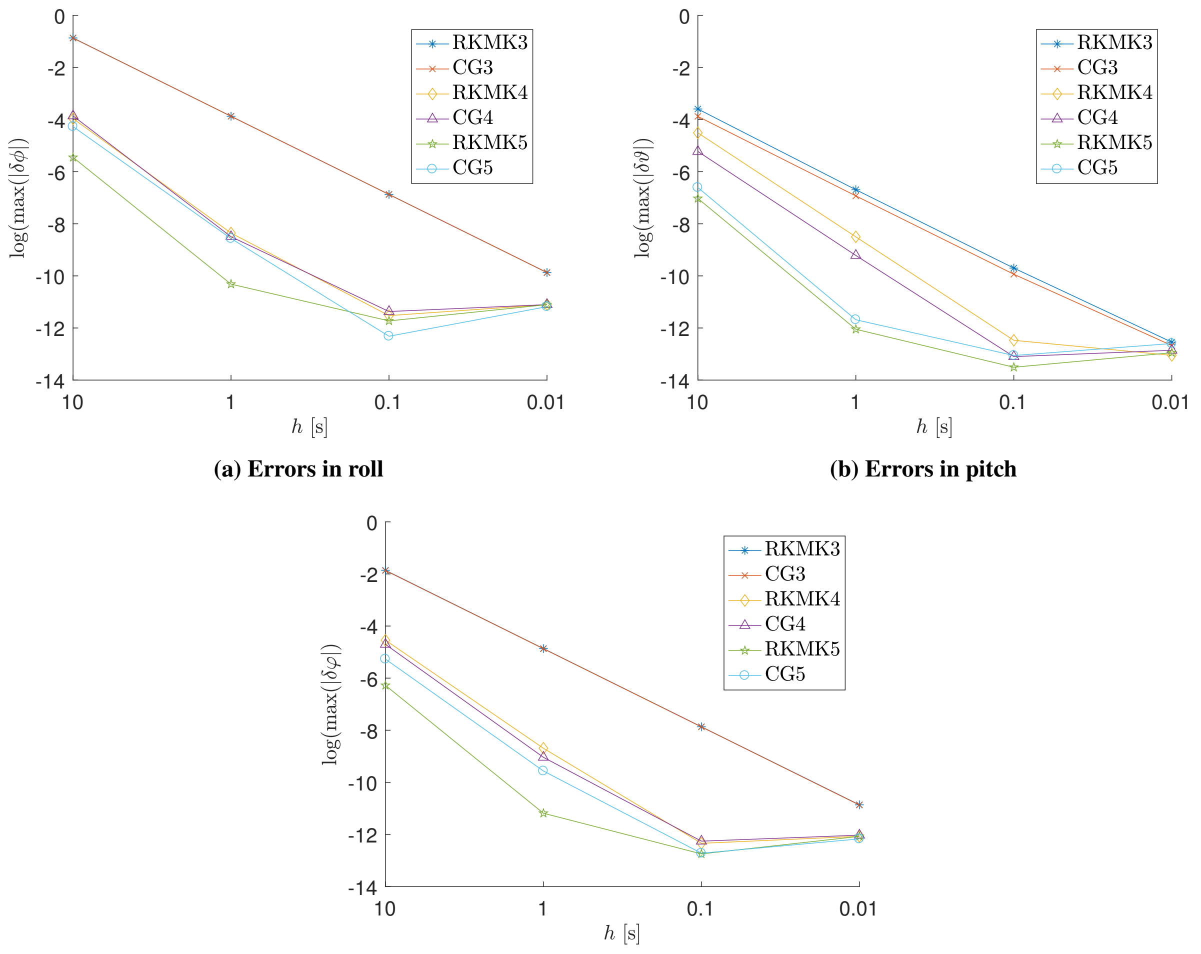
where the components are approximations of the roll, pitch and yaw of a rotation. The error metric is the maximum value of error in roll, pitch and yaw during a 4-hour simulation.

A. Simulation with exact inverse right Jacobian

In this simulation, we employed the exact form of the inverse right Jacobian, i.e. (20), for the RKMK methods. The quaternion exponential was computed as stated in (11) for all methods and the trigonometric functions were computed using built-in MatLab functions. The errors for the integrators are plotted in Fig. 2 for varying time steps.

It can be seen from Fig. 2 that the errors for RKMK3 and CG3 are overlapping for roll and yaw, while CG3 performs slightly better in pitch. However, it must be acknowledged that the errors in pitch are approximately 2 orders of magnitude lower than the errors in roll and yaw, thus the total error difference between RKMK3 and CG3 is small. The same can be observed for RKMK4 and CG4. RKMK5 have smaller error than CG5, which can be explained by the accumulation of round-off errors for CG5. This can become a significant source of error as CG5 has $s = 9$ stages and 54 Butcher coefficients, which are all irrational values truncated to 16 digits [13].

It is also noted that for the smallest time steps considered, i.e. $h = 0.01$ s, the error for the higher order methods increase compared to time step $h = 0.1$ s due to the accumulation of machine and round-off errors. The



(a) Errors in roll

(b) Errors in pitch

(c) Errors in yaw

Fig. 2 Maximum Euler angle simulation errors for different time steps.

accuracy for the 4th and 5th order methods converge for smaller time steps, thus there is no advantage of using higher order methods for small time steps. .

B. Simulation with approximated inverse right Jacobian

We repeated the simulations in Sect. V.A using the 3rd order Taylor approximation of the inverse right Jacobian, i.e. (30), for the RKMK methods. The quaternion exponential and the trigonometric function calls were computed in the same manner as in Sect. V.A for all methods. Since the increase in errors when using the approximated inverse right Jacobian was small it was not possible to distinguish the results from the plots in Fig. 2. Therefore, we have listed the increase in logarithmic errors when using the approximated inverse right Jacobian in Table 8. Note that when the time step was smaller than $h = 1$ s no increase in errors occurred. The table shows that the increase in errors are larger when higher time steps are used. This can be supported by the fact that the argument of the inverse right Jacobian is larger for large time steps. The higher order methods also suffer a larger increase in error than the lower order methods, due to the already high precision of the higher order methods.

C. Unit quaternion norm preserving properties

Both RKMK and CG are by definition norm preserving numerical solvers due to the exponential updates. However, due to machine precision and the accumulation of round off errors the unit norm of the quaternions may not be preserved. The norm error for the 4-hour simulations are plotted in Fig. 3. In general it is expected that the RKMK methods will have better norm preserving properties than the CG method of equivalent order due to the lower number of quaternion exponential function calls. We do however observe two inconsistencies in the results. The first is seen

Table 8 Increase in the logarithmic errors for the RKMK methods when using approximated inverse right Jacobian.

		$h = 10$s	$h = 1$s
RKMK3	$\log(\delta\phi)$	1.41×10^{-7}	2.52×10^{-11}
	$\log(\delta\vartheta)$	7.88×10^{-7}	2.37×10^{-8}
	$\log(\delta\varphi)$	1.53×10^{-7}	4.73×10^{-10}
RKMK4	$\log(\delta\phi)$	3.54×10^{-6}	4.71×10^{-7}
	$\log(\delta\vartheta)$	4.99×10^{-6}	3.91×10^{-7}
	$\log(\delta\varphi)$	3.58×10^{-7}	2.04×10^{-6}
RKMK5	$\log(\delta\phi)$	1.36×10^{-3}	2.90×10^{-6}
	$\log(\delta\vartheta)$	9.56×10^{-3}	4.47×10^{-4}
	$\log(\delta\varphi)$	3.94×10^{-3}	1.79×10^{-4}

for the smallest time step considered in Fig. 3a, where the CG3 method have a smaller norm error than RKMK3. The second inconsistency is that the CG5 method has a similar or smaller norm error than CG4 throughout the simulations. However, the results shows that the RKMK methods have similar or better norm preserving properties than the CG methods of equivalent order, except for the case already discussed.

VI. Conclusion

The Lie group integrators of Crouch–Grossman (CG) and Runge–Kutta–Munthe-Kaas (RKMK) have been formulated in this paper in terms of quaternion multiplication, quaternion exponential, and the quaternion Jacobian, and their application for the integration of the quaternion kinematics has been successfully demonstrated. The simulation results showed that the accuracy level of CG methods can be achieved at a decreased computational cost with RKMK methods. Furthermore, operational counts have been provided for the methods considered to enable a quantitative comparison of their computational cost. A third-order approximation of the inverse right Jacobian has also been provided, and it has been shown that the computational cost of the RKMK method could be reduced with little loss in accuracy.

Acknowledgments

The research presented in this paper was funded by the Norwegian Research Council under Project Number 237896, SFI Offshore Mechatronics.

References

[1] Shuster, M. D., “Survey of Attitude Representations,” *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 439–517.

[2] Crassidis, J. L., Markley, F. L., and Cheng, Y., “Survey of Nonlinear Attitude Estimation Methods,” *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 1, 2007, pp. 12–28. doi:10.2514/1.22452.

[3] Crouch, P. E., and Grossman, R., “Numerical Integration of Ordinary Differential Equations on Manifolds,” *Journal of Nonlinear Science*, Vol. 3, No. 1, 1993, pp. 1–33. doi:10.1007/BF02429858.

[4] Andrle, M. S., and Crassidis, J. L., “Geometric Integration of Quaternions,” *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 6, 2013, pp. 1762–1767. doi:10.2514/1.58558.

[5] Iserles, A., Munthe-Kaas, H., Nørsett, S., and Zanna, A., “Lie-group Methods,” *Acta Numerica*, 2005, pp. 36–40. doi:10.1017/S0962492900002154.

[6] Butcher, J. C., and Goodwin, N., *Numerical Methods for Ordinary Differential Equations*, Wiley Online Library, 2008, Vol. 2, Chap. 23, pp. 93–99. doi:10.1002/9780470753767.

[7] Eberly, D., “Rotation Representations and Performance Issues,” *Magic Software Inc., Chapel Hill, NC*, 2002.

[8] Angulo, J., “Riemannian L p Averaging on Lie Group of Nonzero Quaternions,” *Advances in Applied Clifford Algebras*, Vol. 24, No. 2, 2014, pp. 355–382. doi:10.1007/s00006-013-0432-2.

[9] Huynh, D. Q., “Metrics for 3D Rotations: Comparison and Analysis,” *Journal of Mathematical Imaging and Vision*, Vol. 35, No. 2, 2009, pp. 155–164. doi:10.1007/s10851-009-0161-2.

[10] Chirikjian, G. S., *Stochastic Models, Information Theory, and Lie Groups, Volume 2*, Birkhäuser, 2012, Chap. 10, pp. 29, 39–41.

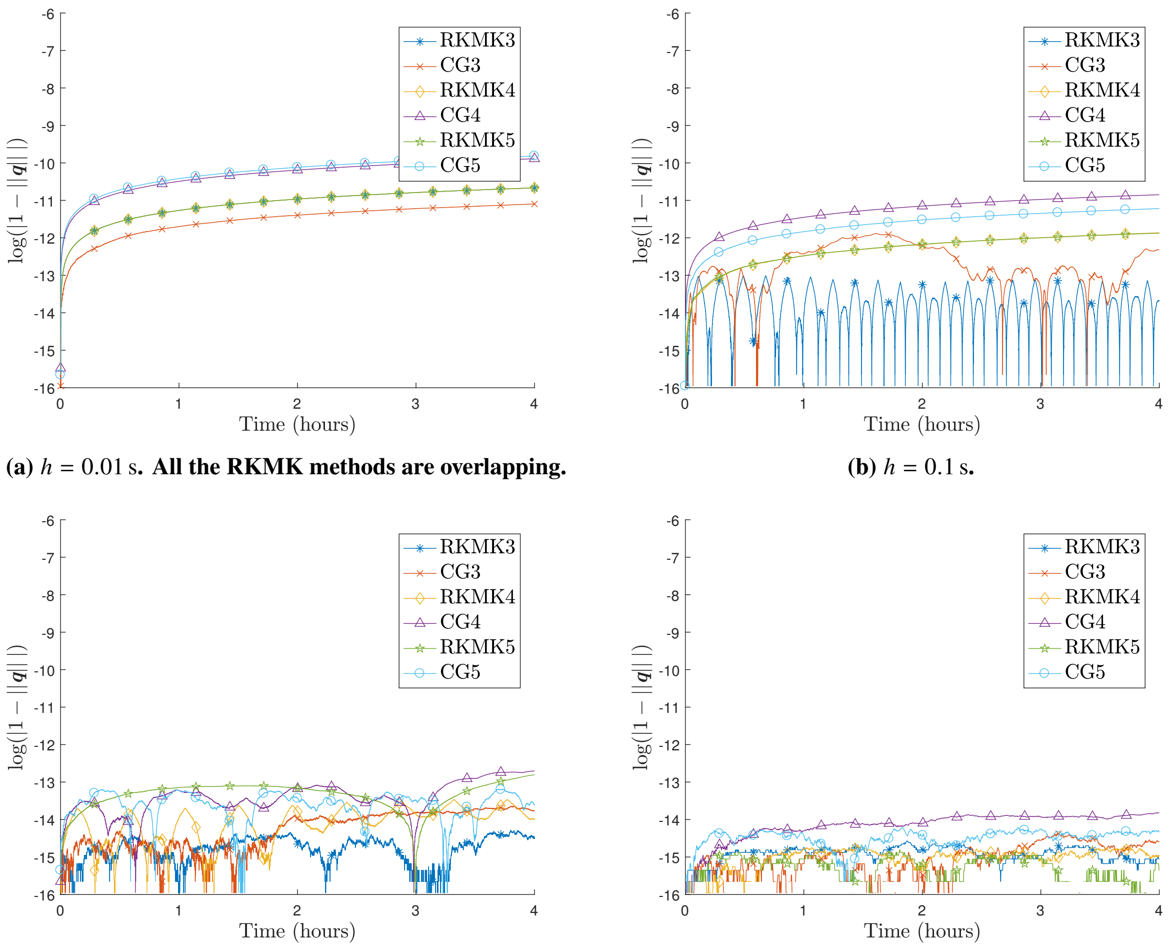
[11] Bullo, F., and Murray, R. M., “Proportional Derivative (PD) Control on the Euclidean Group,” CDS Technical Report 95-010, California Institute of Technology, 1995.

[12] Barfoot, T. D., and Furgale, P. T., “Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems,” *IEEE Trans. Robotics*, Vol. 30, No. 3, 2014, pp. 679–693. doi:10.1109/TRO.2014.2298059.

[13] Jackiewicz, Z., Marthinsen, A., and Owren, B., “Construction of Runge–Kutta Methods of Crouch–Grossman Type of High Order,” *Advances in Computational Mathematics*, Vol. 13, No. 4, 2000, pp. 405–415. doi:10.1023/A:1016645730465.

[14] Pittelkau, M. E., “Rotation Vector in Attitude Estimation,” *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 6, 2003, pp. 855–860. doi:10.2514/2.6929.

[15] Bortz, J. E., “A New Mathematical Formulation for Strapdown Inertial Navigation,” *IEEE Trans. Aerospace and Electronic Systems*, Vol. AES-7, No. 1, 1970, pp. 61–66. doi:10.1109/TAES.1971.310252.



(a) h = 0.01 s**. All the RKMK methods are overlapping.**

(b) h = 0.1 s**.**

(c) h = 1 s**.**

(d) h = 10 s**.**

Fig. 3 Quaternion norm errors for the considered CG and RKMK methods over a 4-hour simulation.