THE JOURNAL OF NAVIGATION (2017), **70**, 165–183. © The Royal Institute of Navigation 2016
doi:10.1017/S037346331600045X

Strapdown Inertial Navigation Algorithms Based on Lie Group

Jun Mao, Junxiang Lian and Xiaoping Hu

(*Department of Automatic Control, College of Mechatronics and Automation, National University of Defense Technology*)
(E-mail: maojun12@nudt.edu.cn)

This paper presents a framework for a strapdown Inertial Navigation System (INS) algorithm design by using Lie group and Lie algebra. The general way to solve Lie group differential equations is introduced. Investigations reveal that this general Lie group method provides a simpler unified way to solve differential equations involving direction cosine matrix, quaternion and dual quaternion, which are widely used in INS algorithm design. Furthermore, we also present a new INS algorithm based on the Special Euclidean group $SE(3)$ under the guidelines of Lie group method. Analyses show that $SE(3)$ algorithm has the same accuracy as a dual quaternion algorithm, this is also justified by numerical simulations. Though the $SE(3)$ algorithm has no improvements in accuracy, the general Lie group method used in the design process shows its brevity and uniformity.

KEYWORDS

1. Inertial Navigation System. 2. Lie Group. 3. Lie Algebra.

Submitted: 2 December 2015. Accepted: 6 June 2016. First published online: 28 July 2016.

1. INTRODUCTION. A typical Strapdown Inertial Navigation System (SINS) algorithm is a process of calculating the attitude/velocity/position in the navigation frame by integration of the angular rate and total acceleration. Total acceleration is calculated as the sum of gravitational acceleration and specific force acceleration which is produced by non-gravitational forces. Modern attitude integration adopts a two-speed approach: a high-speed calculation where an exact rotation vector differential equation is integrated to account for the non-commutative errors caused by multi-axis angular motion and a moderate-speed calculation in which an analytically exact equation is used to map the rotation vector (provided from the high-speed integration) to attitude changes. A two-speed approach can be also used for velocity/position integration in dynamic environments that parallel the two-speed attitude integration approach. The high speed velocity/position algorithms are designed to account for errors caused by angular and linear oscillation (Savage, 1998b), and the moderate algorithms are used to update velocity/position. Thus, the navigation algorithms are divided into two parts: high-speed circle and moderate-speed circle.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

However, unlike the attitude algorithm which utilises a rotation vector to measure the exact attitude change, velocity/position integration algorithms lack a parallel exact analytical formula, thus being typically limited to first-order accuracy under manoeuvring conditions (Savage, 1998b). To overcome the limitation, Wu et al. (2005) developed the Dual Quaternion (DQ) algorithm which combined rotation and translation as a whole. The geometry algebra INS algorithm (Wu and Wang, 2012) is similar to the DQ algorithm. Later, Savage (2006b) proposed the Velocity Translation Vector (VTV) and Position Translation Vector (PTV) which achieved exact solutions to rigid body translation. Comparisons between DQ and VTV show that dual quaternion has the same velocity translation part as Savage's VTV (Wu, 2006). For the position translation part, Savage's PTV can be reduced to the double integral of specific force acceleration when angular-rate/specific-force are constant, while the DQ deduced PTV has no such character (Wu and Xiao, 2011; Savage, 2006a).

For all the algorithms presented (Savage, 1998a; Savage, 1998b; Savage, 2006b; Wu et al., 2005; Wu and Wang, 2012), the key point is to solve the differential equations. Bortz (1971) acquired the solution to differential equations on a Direction Cosine Matrix (DCM) by analysing the geometry relationships and a series of complicated algebra operations. Savage's translation vectors are carried out in a similar way to Bortz's in two separated processes, thus evolving many more algebra operations. Compared to the derivations of VTV and PTV, the DQ derivation is simpler as it combines rotation and translation as a whole. The screw vector in DQ has a similar form to a rotation vector, and the solutions to DQ differential equations are obtained in a similar way as Bortz's method. However, the screw vector is a six-dimensional dual vector consisting of rotation vector and translation vector, leading to more complicated geometry analysis and algebra operations than Bortz's method. Though researchers (Nazaroff, 1979; Jiang, 1991) tried to simplify Bortz's derivation, their methods can hardly extend to other underlying manifolds used in INS algorithm design, such as DQ or geometry algebra. Careful observations find that DCM, quaternion, DQ and geometry algebra used in INS algorithm design are Lie subgroups. This character inspired us to apply the Lie group method to achieve a simpler and unified way to obtain solutions on these Lie subgroups.

Before our work, Bar-Itzhack (1989) found that a general rotation in N dimensional space can be obtained by the exponential of a skew Hermitian matrix; unfortunately there lacked a way to obtain that skew Hermitian matrix. Later, when using the Lie group theory, we came to understand that Bar-Itzhack's work is a primitive implementation of the Lie group method to rigid body rotations. When taking a Lie group method we find that not only rotations, but also rigid body translations have a general form, and the Lie group method gives a way to obtain that general solution. To justify this, we implemented the Lie group method to DCM, quaternion and DQ to replace the existing derivations with a much simpler unified method. We also developed a special Euclidean group-based algorithm as an example to show how the Lie group method can be applied in INS. As the special Euclidean group $SE(3)$ is homeomorphous to DQ, the new algorithm is supposed to have the same accuracy as the DQ algorithm.

In benefitting from the Lie group method, we can have a modern way to re-recognise the navigation differential equations. Various Lie group integrators have been developed to obtain numerical solutions to Lie group differential equations (Iserles et al., 2000; Celledoni and Owren, 2003), however the existing Lie group integrators are

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

trying to implement a Runge-Kutta-like method on manifolds, and this is not the case for inertial navigation systems which require the algorithms to achieve minimal errors under coning/sculling/scorning environments. Previous attitude algorithms and DQ algorithms adopted a two-speed approach (Wu et al., 2005; Ignagni, 2003). In our paper, we will develop a similar two-speed numerical algorithm to integrate $SE(3)$ group differential equations.

In Section 2 we introduce the general way to solve differential equations on Lie groups, and apply this method to some useful Lie groups used in inertial navigation. Section 3 develops a new INS algorithm based on $SE(3)$, and Section 4 conducts a series of simulations to justify the new algorithm. The concluding remarks are in Section 5.

2. DIFFERENTIAL EQUATIONS ON LIE GROUPS.

2.1. *General Differential Equations On Lie Groups.* A Lie group $\mathfrak{G}$ is a differential manifold, for which the product $\mathfrak{G} \times \mathfrak{G} \to \mathfrak{G}$ is a differential mapping (Jonghoon and Chung, 2005). The Lie algebra $\mathfrak{g}$ of the Lie group $\mathfrak{G}$ is defined as the tangent space at the identity, equipped with the Lie bracket. The Lie bracket is a bilinear, skew-symmetric mapping $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, which also has the Jacobi's identity. The Lie bracket of two Lie algebras is also called the commutator. For two commute Lie algebra F and L, their Lie brackets are $[F, L] = [L, F] = 0$[1].

The differential equations on group $\mathfrak{G}$ can be seen as the tangent vector field on $\mathfrak{G}$, and it can be expressed as the left translation of Lie algebra, that is

$$\dot{y}(t) = y(t)A(t) \tag{1}$$

where $y \in \mathfrak{G}$, $A \in \mathfrak{g}$ and $\dot{y}$ the tangent vector at y.

According to the Lie group theory (Iserles et al., 2000), the solution of the above Lie group differential equation is given by the exponential map $\exp : \mathfrak{g} \to \mathfrak{G}$ of a Lie algebra $C(t)$

$$y(t) = y(0)\exp(C(t)) \tag{2}$$

To compute the Lie algebra $C(t)$, we will introduce some important knowledge first. We start this with the so called Baker-Campbell-Hausdorff theorem (BCH), which says, for sufficiently small $t \geq 0$ we have

$$\exp(tF)\exp(tL) = \exp(tH) \tag{3}$$

where F, L, H are Lie algebras. In the above $H = bch(F, L)$ can be constructed from iterated commutators of F and L. The first few terms are

$$H = F + L + \frac{1}{2}t[F, L] + \frac{1}{12}t^2([F, [F, L] + [L, [L, F]]]) + \mathrm{O}(t^3) \tag{4}$$

Higher-order terms can be obtained by recursion (Varadarajan, 1984). We can see from Equation (4), when F and L commute, H is their sum. However, when two Lie algebras fail to commute, there exist some other terms denoted by their commutators, these terms essentially caused the so called non-commutativity phenomenon.

[1] As Lie bracket is skew-symmetric, we have $[F, L] = -[L, F]$, so if $[F, L] = [L, F]$, then $[F, L] = [L, F] = 0$.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

At this point, the Lie algebra $C(t)$ in Equation (2) can be regarded as the *bch* of $A(t)$ over the time interval. As $A(t)$ is a time varying Lie algebra, we need to carry out the differential equation of $C(t)$ by using the differential of the exponential map. The differential of the exponential mapping is defined as the "left trivialised" tangent of the exponential map, that is, as function $d\exp : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ such that

$$\frac{d}{dt}\exp(C(t)) = \exp(C(t))d\exp_{-C(t)}\left(\dot{C}(t)\right) \tag{5}$$

By differential of Equation (2), we have

$$\begin{aligned} y(0)\exp(C(t))d\exp_{-C(t)}\left(\dot{C}(t)\right) = y(t)A(t) &\Leftrightarrow y(t)d\exp_{-C(t)}\left(\dot{C}(t)\right) \\ &= y(t)A(t) \Leftrightarrow \dot{C}(t) = d\exp^{-1}_{-C(t)}(A(t)) \end{aligned} \tag{6}$$

The above equation can be regarded as a kind of 'infinitesimal BCH generator', it shows how $C(t)$ interacts with the time varying Lie algebra $A(t)$ to generate the current *bch*. $d\exp^{-1}$ is the inverse of $d\exp$, and has the explicit form as

$$\begin{aligned} d\exp^{-1}_{C}(A) &= A - \frac{1}{2}\left[C, A\right] + \frac{1}{12}\left[C, \left[C, A\right]\right] + \cdots \\ &= \sum_{j=0}^{\infty}\frac{B_j}{j!}ad^j_C A \end{aligned} \tag{7}$$

where B_j is the j^{th} Bernoulli number, and $ad_C A$ is the adjoint operator satisfying $ad_C A = [C, A]$. When the *bch* Lie algebra $C(t)$ commute with $A(t)$, all the Lie brackets in Equation (7) will be reduced to zero, and thus $C(t)$ became the integral of $A(t)$.

In summary, the solution of Equation (1) can be generally given by the following lemma.

Lemma 1: For small $t \geq 0$, the solution of Equation (1) is given by

$$y(t) = y(0)\exp(C(t)) \tag{8}$$

where $C \in \mathfrak{g}$ satisfies the differential equation

$$\dot{C}(t) = d\exp^{-1}_{-C(t)}(A(t)) \quad C(0) = 0 \tag{9}$$

In the following part, we will apply Lemma 1 to DCM and DQ to show that the previous independent works of Bortz (1971) and Wu et al. (2005) can be simplified in the same framework by using Lemma 1. We also introduce the rigid body transformation matrix $se(3)$ which is equivalent to DQ but with the usual matrix and vector representation.

2.2. *Differential Equations On so(3).* The special orthogonal group $so(3)$ is defined as

$$so(3) = \left\{\boldsymbol{C} \in \mathbb{R}^{3\times 3} | \boldsymbol{C}\boldsymbol{C}' = I, \det \boldsymbol{C} = 1\right\}$$

where det is the matrix determinant. This group is a subgroup of matrix Lie group with the usual matrix product as the group operator. In strapdown INS algorithms, this group is used as the DCM.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

Suppose the DCM of frame O respect to frame N is denoted as $\boldsymbol{C}_{NO}$, its differential equation is given as

$$\dot{\boldsymbol{C}}_{NO} = \boldsymbol{C}_{NO}\lceil\boldsymbol{\omega}_{NO}^{O}\rceil \tag{10}$$

where $\boldsymbol{\omega}_{NO}^{O}$ is the angular rate of frame O relative to frame N, coordinated in frame O. $\lceil\boldsymbol{\omega}_{NO}^{O}\rceil$ is the matrix representation of $\boldsymbol{\omega}_{NO}^{O} = (\omega_x; \omega_y; \omega_z)$, which has the form

$$\lceil\boldsymbol{\omega}_{NO}^{O}\rceil = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

By comparing to Equation (1), one can find that $\lceil\boldsymbol{\omega}_{NO}^{O}\rceil$ is indeed the Lie algebra of $\boldsymbol{C}_{NO}$, as it is also isomorphic to $\boldsymbol{\omega}_{NO}^{O} \in \mathbb{R}^3$, we also say $\boldsymbol{\omega}_{NO}^{O}$ is the Lie algebra of $\boldsymbol{C}_{NO}$.

The Lie algebra of $so(3)$ is denoted as $\mathfrak{so}(3)$, the Lie bracket of two Lie algebra $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2 \in \mathfrak{so}(3)$ is the matrix commutator given as

$$[\lceil\boldsymbol{\omega}_1\rceil, \lceil\boldsymbol{\omega}_2\rceil] = \lceil\boldsymbol{\omega}_1\rceil\lceil\boldsymbol{\omega}_2\rceil - \lceil\boldsymbol{\omega}_2\rceil\lceil\boldsymbol{\omega}_1\rceil \tag{11}$$

The adjoint operator of $\boldsymbol{\omega}_{NO}^{O}$ is $\lceil\boldsymbol{\omega}_{NO}^{O}\rceil$.

By supplying Lemma 1, the solution of Equation (10) can be given by the exponential map of Lie algebra $\boldsymbol{\phi} \in \mathfrak{so}(3)$ as

$$\boldsymbol{C}_{NO}(t) = \boldsymbol{C}_{NO}(0)\exp(\boldsymbol{\phi}) \tag{12}$$

where $\boldsymbol{\phi} \in \mathfrak{so}(3)$ satisfies the differential equation

$$\dot{\boldsymbol{\phi}} = d\exp_{-\boldsymbol{\phi}}^{-1}\boldsymbol{\omega}_{NO}^{O} \tag{13}$$

For a matrix Lie group, the exponential map is defined as the matrix exponential (Murray et al., 1994). That is

$$\exp(\boldsymbol{\phi}) = \exp m(\lceil\boldsymbol{\phi}\rceil) = \sum_{j=0}^{\infty}\frac{\lceil\boldsymbol{\phi}\rceil^j}{j!} \tag{14}$$

Upon series evaluation, it is analytically given (Jonghoon and Chung, 2005), for $\boldsymbol{\phi} = \boldsymbol{0}$

$$\exp(\boldsymbol{\phi}) = \boldsymbol{I} \tag{15}$$

and for nonzero $\boldsymbol{\phi}$

$$\Omega = \exp(\boldsymbol{\phi}) = \boldsymbol{I} + \frac{\sin(\phi)}{\phi}\lceil\boldsymbol{\phi}\rceil + \frac{\sin^2\left(\frac{\phi}{2}\right)}{\frac{\phi^2}{2}}\lceil\boldsymbol{\phi}\rceil^2 \tag{16}$$

Note that $2\sin^2(\phi/2) = 1 - \cos(\phi)$, the above equation becomes the same as the map between rotation vector and DCM.

$d\exp_{-\boldsymbol{\phi}}^{-1}$ in Equation (13) can be computed by Equation (7) , and it also has a closed-form as

$$d\exp_{-\boldsymbol{\phi}}^{-1} = \boldsymbol{I} + \frac{1}{2}\lceil\boldsymbol{\phi}\rceil - \frac{\phi\cot\left(\frac{\phi}{2}\right) - 2}{2\phi^2}\lceil\boldsymbol{\phi}\rceil^2 \tag{17}$$

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

where ϕ is the norm of vector ϕ. When $\phi = \mathbf{0}$, $d\exp^{-1}_{-\boldsymbol{\phi}} = \boldsymbol{I}$,

Apply the closed-form to Equation (13) and note that $\cot\left(\frac{\phi}{2}\right) = \frac{\sin\phi}{1-\cos\phi}$, we have

$$\dot{\boldsymbol{\phi}} = \left[\boldsymbol{I} + \frac{1}{2}\lceil\boldsymbol{\phi}\rceil + \frac{1}{\phi^2}\left(1 - \frac{\phi\sin\phi}{2(1-\cos\phi)}\right)\lceil\boldsymbol{\phi}\rceil^2\right]\boldsymbol{\omega}^O_{NO} \tag{18}$$

This is the same as the differential equation of rotation vector.

Equations (16) and (18) were developed by Bortz (1971) in a lengthy geometry method, and later simplified by several other researchers (Nazaroff, 1979; Jiang, 1991). The Bortz equations achieved the exact expression of non-commutativity and show how previous rotations interact with the current angular velocity to obtain one final rotation. When using the Lie group method, the angular velocity can be regarded as the Lie algebra of the $so(3)$ group, thus the Baker-Campbell-Hausdorff theorem can be applied to describe how finite rotations can combine into one single rotation. The rotation vector obtained by Bortz is exactly the *bch* of angular velocity in Lie group theory, and can be computed according to Equation (6).

In the above, we show how Lemma 1 can be used to obtain the same outcome as Bortz's work in a simpler mathematical way. The following part also extends in the same way to obtain solutions on other manifolds used in INS algorithms.

2.3. *Differential Equations On* $SE(3)$. For a point $\boldsymbol{q} \in R^3$ in frame O, its position vector in the reference frame N can be expressed as

$$\boldsymbol{q}^N = \boldsymbol{p}^N + \boldsymbol{C}_{NO}\boldsymbol{q}^O \tag{19}$$

where $\boldsymbol{p}^N$ is the coordinates of the origin of frame O relative to frame N, the superscripts is the corresponding measured frame. See Figure 1.

The transformation can form a special Euclidean group $SE(3)$, as a 4×4 matrix of the form

$$\boldsymbol{Y}_{NO} = \begin{bmatrix} \boldsymbol{C}_{NO} & \boldsymbol{p}^N \\ 0 & 1 \end{bmatrix} \in SE(3) \tag{20}$$

$SE(3)$ is also a subgroup of a matrix Lie group, with the unit determinant as the group property. The differential equation can be given as

$$\begin{bmatrix} \dot{\boldsymbol{C}}_{NO} & \dot{\boldsymbol{p}}^N \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{NO} & \boldsymbol{p}^N \\ 0 & 1 \end{bmatrix}\begin{bmatrix} \lceil\boldsymbol{\omega}^O_{NO}\rceil & \boldsymbol{v}^O_N \\ 0 & 0 \end{bmatrix}$$
$$\Leftrightarrow \dot{\boldsymbol{Y}}_{NO} = \boldsymbol{Y}_{NO}\lceil\boldsymbol{V}^O_{NO}\rceil \tag{21}$$

where $\lceil\boldsymbol{V}_{NO}\rceil$ is the matrix representation of the vector $\boldsymbol{V}_{NO} = (\boldsymbol{v}^O_N; \boldsymbol{\omega}^O_{NO}) \in \mathbb{R}^6$, and $\boldsymbol{v}^O_N$ is the velocity with respect to frame N coordinated in frame O. By comparing to Equation (1), we can see $\lceil\boldsymbol{V}_{NO}\rceil$ belongs to the Lie algebra[2] of $SE(3)$, denoted by $\mathfrak{se}(3)$. Just as the Lie bracket of $\mathfrak{so}(3)$, the Lie bracket on $\mathfrak{se}(3)$ is also the matrix

[2] As discussed earlier, the vector $\boldsymbol{V}$ can also treated as a Lie algebra.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

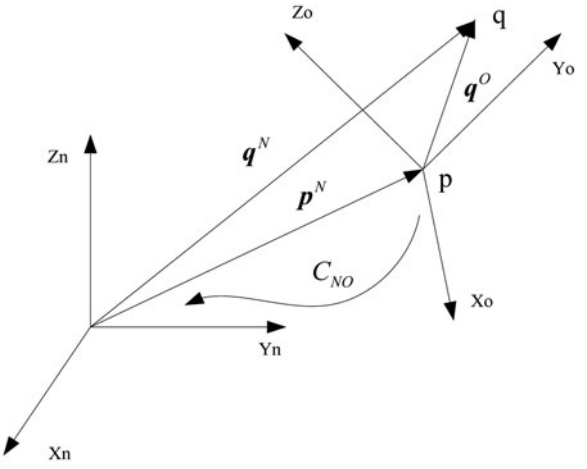


Figure 1. General motion of frame O respect to frame N.

commutator given as

$$[\lceil \boldsymbol{V}_1 \rceil, \lceil \boldsymbol{V}_2 \rceil] = \lceil \boldsymbol{V}_1 \rceil \lceil \boldsymbol{V}_2 \rceil - \lceil \boldsymbol{V}_2 \rceil \lceil \boldsymbol{V}_1 \rceil \tag{22}$$

Making use of the adjoint operator defined by

$$ad_{\boldsymbol{V}} = \begin{bmatrix} \lceil \boldsymbol{\omega} \rceil & \lceil \boldsymbol{v} \rceil \\ 0 & \lceil \boldsymbol{\omega} \rceil \end{bmatrix} \tag{23}$$

The Lie bracket is readily obtained as $[\lceil \boldsymbol{V}_1 \rceil, \lceil \boldsymbol{V}_2 \rceil] = \lceil ad_{V1} \boldsymbol{V}_2 \rceil$

Following Lemma 1, the solution to Equation (21) is the exponential of some Lie algebra $\boldsymbol{\varphi} \in \mathfrak{se}(3)$, given as

$$\boldsymbol{Y}(t) = \boldsymbol{Y}(0) \exp(\boldsymbol{\varphi}) \tag{24}$$

where $\boldsymbol{\varphi} \in \mathfrak{se}(3)$ satisfies the differential equation

$$\dot{\boldsymbol{\varphi}}(t) = d \exp^{-1}_{-\boldsymbol{\varphi}} \boldsymbol{V}^O_{NO} \tag{25}$$

Similar to the exponential map on $\mathfrak{so}(3)$, the exponential of $\boldsymbol{\varphi} \in \mathfrak{se}(3)$ is originally given as

$$\exp(\boldsymbol{\varphi}) = \exp m(\lceil \boldsymbol{\varphi} \rceil) = \sum_{j=0}^{\infty} \frac{\lceil \boldsymbol{\varphi} \rceil^j}{j!}$$

By using Rodrigues' formula, the close form of the exponential map of $\boldsymbol{\varphi} = (\boldsymbol{\lambda}; \boldsymbol{\phi}) \in \mathfrak{se}(3)$ is (Jonghoon and Chung, 2005), for $\boldsymbol{\phi} = \mathbf{0}$

$$\exp(\boldsymbol{\varphi}) = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{\lambda} \\ 0 & 1 \end{bmatrix} \tag{26}$$

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

for nonzero $\boldsymbol{\phi}$

$$\exp(\boldsymbol{\varphi}) = \begin{bmatrix} \boldsymbol{\Omega} & \frac{1}{\phi^2}((\boldsymbol{I} - \Omega)(\boldsymbol{\phi} \times \boldsymbol{\lambda}) + \boldsymbol{\phi}\boldsymbol{\phi}'\boldsymbol{\lambda}) \\ 0 & 1 \end{bmatrix} \tag{27}$$

where $\boldsymbol{\Omega} = \exp(\boldsymbol{\phi}) = \exp m(\lceil \boldsymbol{\phi} \rceil) \in so(3)$

$d\exp^{-1}_{\boldsymbol{\varphi}}$ can be computed by Equation (7), however, we will introduce the close form of it for efficient computation, before this, we firstly define some parameters, they are

$$s = \frac{\sin\frac{\phi}{2}}{\frac{\phi}{2}} \qquad c = \cos\left(\frac{\phi}{2}\right) \tag{28}$$

They are used to compute

$$\beta = s^2 \quad \text{and} \quad \gamma = \frac{c}{s} \tag{29}$$

For Lie algebra $\boldsymbol{\varphi} = (\boldsymbol{\lambda}; \boldsymbol{\phi}) \in \mathfrak{se}(3)$ with nonzero $\boldsymbol{\phi}$, the inverse of the differential of the exponential map are presented as a 6×6 matrix

$$d\exp^{-1}_{\boldsymbol{\varphi}} = \begin{bmatrix} d\exp^{-1}_{\boldsymbol{\phi}} & \boldsymbol{D} + \frac{1}{2}\lceil \boldsymbol{\lambda} \rceil \\ 0 & d\exp^{-1}_{\boldsymbol{\phi}} \end{bmatrix} \tag{30}$$

where

$$\boldsymbol{D} = \frac{1-\gamma}{\|\phi\|^2}\lceil \boldsymbol{\lambda}, \boldsymbol{\phi} \rceil + \frac{\frac{1}{\beta} + \gamma - 2}{\|\phi\|^4}(\boldsymbol{\phi}'\boldsymbol{\lambda})\lceil \boldsymbol{\phi} \rceil^2$$

$\boldsymbol{\phi}'$ is the transpose of $\boldsymbol{\phi}$, and $\lceil \boldsymbol{\lambda}, \boldsymbol{\phi} \rceil = \lceil \boldsymbol{\lambda} \rceil \lceil \boldsymbol{\phi} \rceil + \lceil \boldsymbol{\phi} \rceil \lceil \boldsymbol{\lambda} \rceil$

If $\boldsymbol{\phi} = \boldsymbol{0}$, then $\boldsymbol{D} = \boldsymbol{0}$. When $\sin(\phi/2) = 0$ and $\boldsymbol{\phi} \neq \boldsymbol{0}$, the close-form becomes singular, at this time Equation (7) can be used to approximate it.

From the last section, we have seen that Lie algebra $\boldsymbol{\phi}$ is indeed the rotation vector given by Bortz. Here we will investigate the translation part in $\boldsymbol{\varphi}$, that is $\boldsymbol{\lambda}$. As presented in Equation (20), the vector in the last column represents the position vector of $\exp(\boldsymbol{\varphi})$ with respect to $\boldsymbol{Y}(0)$, denoted by $\Delta\boldsymbol{r}$

$$\Delta\boldsymbol{r} = \frac{1}{\phi^2}((\boldsymbol{I} - \Omega)\lceil \boldsymbol{\phi} \rceil \boldsymbol{\lambda} + \boldsymbol{\phi}\boldsymbol{\phi}'\boldsymbol{\lambda}) \tag{31}$$

Take the same function of ϕ defined in Savage, (2006b) as

$$f_1(\phi) = \frac{\sin\phi}{\phi} \qquad f_2(\phi) = \frac{1-\cos\phi}{\phi^2} \quad \text{and} \quad f_3(\phi) = \frac{1 - f_1(\phi)}{\phi^2}$$

Then take Equation (16) into Equation (31)

$$\Delta\boldsymbol{r} = \frac{1}{\phi^2}\left[\left(I - I - f_1(\phi)\lceil \boldsymbol{\phi} \rceil - f_2(\phi)\lceil \boldsymbol{\phi} \rceil^2\right)\lceil \boldsymbol{\phi} \rceil + \boldsymbol{\phi}\boldsymbol{\phi}'\right]\boldsymbol{\lambda} \tag{32}$$

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

Note that $\frac{\boldsymbol{\phi}\boldsymbol{\phi}'}{\phi^2} = \frac{\lceil\boldsymbol{\phi}\rceil\lceil\boldsymbol{\phi}\rceil}{\phi^2} + \boldsymbol{I}$ and $\lceil\boldsymbol{\phi}\rceil^3 = -\phi^2\lceil\boldsymbol{\phi}\rceil$, the translation can be given as

$$\Delta\boldsymbol{r} = \left[\boldsymbol{I} + f_2(\phi)\lceil\boldsymbol{\phi}\rceil + f_3(\phi)\lceil\boldsymbol{\phi}\rceil^2\right]\boldsymbol{\lambda} \tag{33}$$

Comparing to the translation part in DQ (Wu, 2006), $\boldsymbol{\lambda}$ is indeed the same as the dual part of the screw vector.

The differential equation of $\boldsymbol{\lambda}$ is

$$\dot{\boldsymbol{\lambda}} = \left(\boldsymbol{D} + \frac{1}{2}\boldsymbol{\lambda}\right)\boldsymbol{\omega}_{NO}^{O} + d\exp_{-\boldsymbol{\phi}}^{-1}\boldsymbol{v}_N^O \tag{34}$$

In fact, the above equation is also the same as the differential equation of the dual part in DQ (See Appendix A for details). As the real part of DQ is also the same as Bortz' rotation vector, the $SE(3)$ group and DQ are analytically the same.

Several papers have discussed the differences of the translations in DQ and Savage's translations vectors (Wu, 2006; Wu and Xiao, 2011; Savage, 2006a). As $\boldsymbol{\lambda}$ is equivalent to the translation part of DQ, the comparisons of $\boldsymbol{\lambda}$ and translation vectors are similar to these works. In brief, $\boldsymbol{\lambda}$ is the same as a velocity transition vector, however, $\boldsymbol{\lambda}$ can hardly deduce a PTV that can reduce to a double integral of specific force acceleration when angular-rate/specific-forces are constant.

2.4. *Other Lie Groups Used In INS.* Applications of the general Lie group method in two useful matrix Lie group $SO(3)$ and $SE(3)$ are presented. However, as we can see, this general method is not constrained to a matrix Lie group. Any Lie group differential equation satisfying Equation (1) can be solved in this way. In fact, quaternion and dual quaternion are also subgroups of Lie group, with the unit norm as their group identity and quaternion/dual quaternion product as the group operator. Their differential equations are[3]

$$\dot{\boldsymbol{q}} = \boldsymbol{q}\left(\frac{1}{2}\boldsymbol{\omega}_{NO}^{O}\right) \quad \text{and} \quad \dot{\hat{\boldsymbol{q}}} = \hat{\boldsymbol{q}}\left(\frac{1}{2}\hat{\boldsymbol{w}}_{NO}^{O}\right) \tag{35}$$

where $\boldsymbol{q}$ is quaternion, $\hat{\boldsymbol{q}}$ is dual quaternion and $\hat{\boldsymbol{w}}_{NO}^{O}$ is the corresponding dual vector.

These differential equations are in exactly the same form as Equation (1). Furthermore, the dual/angular vector product also satisfies the Lie bracket property. Thus, the general Lie group method can also apply to quaternion and dual quaternion (see Appendix B).

The existing solutions of DCM and dual quaternion are re-derived by using the Lie group method, we also introduced the $SE(3)$ group for rigid body dynamics. Traditionally, the maps between kinematic vectors and quaternion/ dual quaternion/ DCM are given by analysing the geometry relationships between them which involve a number of algebraic manipulations. However, this can be greatly simplified when taking a modern treatment of the differential equations as they are Lie group equations. Thus, the map between kinematic vectors and quaternion/ dual quaternion/ DCM/ $SE(3)$ is indeed the exponential map. The differential of the kinematic vector can be given by a single step by using a general differential of the exponential map.

[3] Here, the vector $\boldsymbol{\omega}_{NO}^{O}$ is a vector with 4 elements, which the first one is zero, $\boldsymbol{\omega}_{NO}^{O} = \begin{bmatrix}0; & \boldsymbol{\omega}_{NO}^{O}\end{bmatrix}$

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

3. STRAPDOWN INERTIAL NAVIGATION ALGORITHMS BASED ON $SE(3)$.

3.1. *Navigation Differential Equations.* The general Lie group method and its applications in rigid body dynamics are introduced in the last section. In this section, we will design an INS algorithm based on this unified Lie group method in $SE(3)$ form. This new algorithm acts as an example of a Lie group-based INS algorithm, and as a comparison with the previous DQ algorithm which is designed in a primitive geometrical way.

Before introducing the new algorithm, we define three coordinate frames, they are body frame B, Earth frame E and inertial frame I. Frame B is aligned with the strapdown inertial sensors axis. Frame E has its origin at centre of Earth, and one axis parallel to Earth's polar axis, the other axis fixed to the Earth and parallel to the equatorial plane. Inertial frame I is defined as the Earth frame at time instant $t = 0$.

As we know, the velocity update algorithm in the INS system is a process of integrating the total acceleration. The total acceleration is the sum of gravitational acceleration and the specific force acceleration, in the inertial frame it can be expressed as

$$\ddot{\boldsymbol{r}}^I = \boldsymbol{g}^I + \boldsymbol{s}^I \tag{36}$$

where $\boldsymbol{r}$ is the position vector of the body frame, $\boldsymbol{g}$ is the gravitational acceleration, $\boldsymbol{s}$ is the specific force acceleration. Superscripts mean they are coordinated in frame I.

Velocity and position are the integrations of Equation (36), they are given as

$$\dot{\boldsymbol{r}}^I = \dot{\boldsymbol{r}}^I(0) + \int_0^t \left(\boldsymbol{g}^I + \boldsymbol{s}^I\right) d\tau$$
$$\boldsymbol{r}^I = \boldsymbol{r}^I(0) + \dot{\boldsymbol{r}}^I(0)t + \int_0^t \left[\int_0^{\tau_1} \left(\boldsymbol{g}^I + \boldsymbol{s}^I\right) d\tau_2\right] d\tau_1.$$

In a typical strapdown INS, the specific force acceleration is measured in the body frame, and the gravitational acceleration is calculated in the Earth frame or the local frame. To avoid unnecessary coordinate transformation, it is more convenient to treat the gravitational acceleration and the specific force acceleration separately.

Define the thrust velocity $\boldsymbol{v}_t$ as the integration of the specific force acceleration, i.e.

$$\boldsymbol{v}_t^I = \int_0^t \boldsymbol{s}^I d\tau. \tag{37}$$

Similarly, define the gravitational velocity $\boldsymbol{v}_g$ as the sum of initial velocity and integration of gravitational acceleration, i.e.

$$\boldsymbol{v}_g^I = \dot{\boldsymbol{r}}^I(0) + \int_0^t \boldsymbol{g}^I d\tau. \tag{38}$$

Thus, the velocity can be written as $\dot{\boldsymbol{r}}^I = \boldsymbol{v}_t^I + \boldsymbol{v}_g^I$. Since the velocity of the vehicle can be decomposed into the thrust velocity $\boldsymbol{v}_t$ and gravitational velocity $\boldsymbol{v}_g$, one can calculate them in two independent frames parallel to the body frame and the Earth frame. To do this, the velocity frames T and G are introduced (Wu et al., 2005).

For thrust velocity frame T, its axes are aligned with those of body frame B, vector from Earth's centre to its origin is exactly thrust velocity. For gravitational velocity frame G, its axes are aligned with Earth frame E, the vector from Earth's centre to its origin is exactly gravitational velocity.

Firstly, consider the inertial frame I and the thrust velocity frame T. Note that, it is the thrust velocity that represents the position vector of frame T, as a consequence, the derivative of the translation is not the velocity but the specific acceleration.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

Suppose a group element $\boldsymbol{Y}_{IT} \in SE(3)$ characterises the general displacement of frame T. Since frame T is parallel to the body frame, we have $\boldsymbol{\omega}_{IT}^{T} = \boldsymbol{\omega}_{IB}^{B}$ and $\boldsymbol{C}_{IT} = \boldsymbol{C}_{IB}$. Then, the Lie algebra of $\boldsymbol{Y}_{IT} \in SE(3)$ can be written as $\boldsymbol{\Lambda}_{IT}^{T} = (\boldsymbol{s}^{T}; \boldsymbol{\omega}_{IT}^{T}) = (\boldsymbol{s}^{B}; \boldsymbol{\omega}_{IB}^{B}) \in \mathfrak{se}(3)$.

The rate equation of $\boldsymbol{Y}_{IT}$ can be acquired by using Equation (21), as the form

$$\begin{bmatrix} \dot{\boldsymbol{C}}_{IT} & \dot{\boldsymbol{v}}_{t}^{I} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{IT} & \boldsymbol{v}_{t}^{I} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lceil \boldsymbol{\omega}_{IT}^{T} \rceil & \boldsymbol{s}^{T} \\ 0 & 0 \end{bmatrix}$$
$$\dot{\boldsymbol{Y}}_{IT} = \boldsymbol{Y}_{IT} \lceil \boldsymbol{\Lambda}_{IT}^{T} \rceil \tag{39}$$

The direction cosine matrix $\boldsymbol{C}_{IT}$ and thrust velocity $\boldsymbol{v}_{t}^{I}$ can be derived from the solution of the above equation.

Secondly, consider the inertial frame I and the gravitational velocity frame G. Note that it is the gravitational velocity that represents the position vector of frame G, so the derivative of the translation is not the velocity but the gravitational acceleration.

Let $\boldsymbol{Y}_{IG} \in SE(3)$ describe the general displacement of frame G, the Lie algebra of $\boldsymbol{Y}_{IG} \in SE(3)$ can be expressed as $\boldsymbol{\Lambda}_{IG}^{G} = (\boldsymbol{g}^{G}; \boldsymbol{\omega}_{IG}^{G}) = (\boldsymbol{g}^{E}; \boldsymbol{\omega}_{IE}^{E}) \in \mathfrak{se}(3)$. The differential equation is

$$\begin{bmatrix} \dot{\boldsymbol{C}}_{IG} & \dot{\boldsymbol{v}}_{g}^{I} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{IG} & \boldsymbol{v}_{g}^{I} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lceil \boldsymbol{\omega}_{IG}^{G} \rceil & \boldsymbol{g}^{G} \\ 0 & 0 \end{bmatrix}$$
$$\dot{\boldsymbol{Y}}_{IG} = \boldsymbol{Y}_{IG} \lceil \boldsymbol{\Lambda}_{IG}^{G} \rceil \tag{40}$$

The direction cosine matrix $\boldsymbol{C}_{IG}$ and gravitational velocity $\boldsymbol{v}_{g}^{I}$ can be derived from the solution of the above equation.

To obtain position, we define a position frame U, with its axes aligned with those of the Earth frame; the vector from Earth's centre to its origin is exactly the position vector of the vehicle. By definition, we have $\boldsymbol{\omega}_{IU}^{U} = \boldsymbol{\omega}_{IE}^{E}$ and $\boldsymbol{v}_{I}^{U} = \boldsymbol{C}_{UI}\boldsymbol{v}_{I}^{I} = \boldsymbol{C}_{EI}(\boldsymbol{v}_{t}^{I} + \boldsymbol{v}_{g}^{I})$. Suppose a group element $\boldsymbol{Y}_{IU} \in SE(3)$ shows the displacement of frame U, and the corresponding Lie algebra is $\boldsymbol{V}_{IU}^{U} = (\boldsymbol{v}_{I}^{U}; \boldsymbol{\omega}_{IU}^{U}) \in \mathfrak{se}(3)$.

The rate equation of $\boldsymbol{Y}_{IU}$ is

$$\begin{bmatrix} \dot{\boldsymbol{C}}_{IU} & \dot{\boldsymbol{r}}^{I} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{C}_{IU} & \boldsymbol{r}^{I} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \lceil \boldsymbol{\omega}_{IU}^{U} \rceil & \boldsymbol{v}_{I}^{U} \\ 0 & 0 \end{bmatrix}$$
$$\dot{\boldsymbol{Y}}_{IU} = \boldsymbol{Y}_{IU} \lceil \boldsymbol{V}_{IU}^{U} \rceil \tag{41}$$

By solving the above equation, the position vector $\boldsymbol{r}^{I}$ can be obtained.

So far, the navigation equations are presented in the form of $SE(3)$ and $\mathfrak{se}(3)$. The navigation parameters (attitude, velocity and position) in frame I can be acquired by solving Equations (39), (40) and (41) and they can be expressed in any other navigation frames, i.e., the parameters in Earth frame are

$$\boldsymbol{C}_{EB} = \boldsymbol{C}_{EI}\boldsymbol{C}_{IB} = (\boldsymbol{C}_{IG})'\boldsymbol{C}_{IT}$$
$$\boldsymbol{v}_{E}^{E} = \boldsymbol{v}_{I}^{E} - \boldsymbol{\omega}_{IE}^{E} \times \boldsymbol{r}^{E} = \boldsymbol{v}_{t}^{E} + \boldsymbol{v}_{g}^{E} - \boldsymbol{\omega}_{IE}^{E} \times \boldsymbol{r}^{E}$$
$$\boldsymbol{r}^{E} = \boldsymbol{C}_{I}^{E}\boldsymbol{r}^{I} = (\boldsymbol{C}_{G}^{I})'\boldsymbol{r}^{I}$$

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

The fundamental mechanism of the strapdown INS algorithm based on $SE(3)$ and $\mathfrak{se}(3)$ is presented in Figure 2. The navigation frame is chosen as the Earth frame.

3.2. *Numerical Integration Algorithms.* Though various numerical integration methods have been developed to solve Lie group differential equations, the integration method in INS is, somehow, different. The existing methods impose a Runge-Kutta-like method on manifolds (Iserles et al., 2000). However, in INS we need to develop numerical algorithms to achieve minimal errors under coning/sculling environments. In this section, a new update algorithm is developed to integrate the $SE(3)$ group rate equations. The new algorithm is structured similar to the traditional two-speed approach (Ignagni, 2003).

Note that all the rate Equations (39), (40) and (41) have the same form as

$$\dot{\boldsymbol{Y}} = \boldsymbol{Y}\lceil\boldsymbol{\Lambda}\rceil \quad \text{or} \quad \dot{\boldsymbol{Y}} = \boldsymbol{Y}\lceil\boldsymbol{V}\rceil$$

where superscripts and subscripts are neglected for brevity.

We will take the differential equation of the velocity frame as the general form for further discussion. The Lie algebra $\boldsymbol{\Lambda}$ is generally expressed as $\boldsymbol{\Lambda} = (\boldsymbol{a}; \boldsymbol{\omega}) \in \mathfrak{se}(3)$ in this section. As shown in Equation (24), the solution of the differential equation is the exponential of a Lie algebra and suppose it is expressed as $\boldsymbol{\Phi} = (\boldsymbol{\eta}; \boldsymbol{\phi}) \in \mathfrak{se}(3)$.

The differential equation of the Lie algebra $\boldsymbol{\Phi} = (\boldsymbol{\eta}; \boldsymbol{\phi}) \in \mathfrak{se}(3)$ is

$$\dot{\boldsymbol{\Phi}}(t) = d\exp^{-1}_{-\boldsymbol{\Phi}} \boldsymbol{\Lambda} \tag{42}$$

Taking $d\exp^{-1}_{-\boldsymbol{\Phi}}$ into Equation (42), and adopting a second-order accuracy, that is adopting the first two terms in Equation (7) , we have[4]

$$\dot{\boldsymbol{\Phi}} = \boldsymbol{\Lambda} + \frac{1}{2}[\boldsymbol{\Phi}, \boldsymbol{\Lambda}] \tag{43}$$

Let $\boldsymbol{\Phi} \in \mathfrak{se}(3)$ be small enough, Equation (43) can be approximated by

$$\dot{\boldsymbol{\Phi}} = \boldsymbol{\Lambda} + \frac{1}{2}[\boldsymbol{\alpha}, \boldsymbol{\Lambda}] \quad \text{where} \quad \boldsymbol{\alpha} = \int_{t_k}^{t} \boldsymbol{\Lambda} dt \tag{44}$$

Integrating Equation (44) during a minor interval, the Lie algebra $\Delta\boldsymbol{\Phi}$ is

$$\Delta\boldsymbol{\Phi} = \hat{\boldsymbol{\alpha}} + \frac{1}{2}\int_{t_k}^{t_{k+1}} [\hat{\boldsymbol{\alpha}}, \boldsymbol{\Lambda}] dt \tag{45}$$

where $\hat{\boldsymbol{\alpha}} = (\Delta\boldsymbol{v}; \Delta\boldsymbol{\theta})$ is the accumulated incremental of Lie algebra $\boldsymbol{\Lambda}$ during the current minor interval. Using $\Delta\boldsymbol{\Phi}_s = (\Delta\boldsymbol{\eta}_s; \Delta\boldsymbol{\phi}_s)$ to denote the integrating term, we can investigate the components separately. The integration term in Equation (45) is equivalent to the following formulae

$$\Delta\boldsymbol{\phi}_s = \frac{1}{2}\int_{t_k}^{t_{k+1}} [\Delta\boldsymbol{\theta} \times \boldsymbol{\omega}] dt \quad \text{and} \quad \Delta\boldsymbol{\eta}_s = \frac{1}{2}\int_{t_k}^{t_{k+1}} [\Delta\boldsymbol{\theta} \times \boldsymbol{a} + \Delta\boldsymbol{v} \times \boldsymbol{\omega}] dt \tag{46}$$

$\Delta\boldsymbol{\phi}_s$ and $\Delta\boldsymbol{\eta}_s$ are exactly the same as the coning and the sculling in the conventional algorithm (Savage, 1998a; Savage, 1998b). As $\Delta\boldsymbol{\Phi}_s$ combined the two terms as a whole, it is named as a screw correction.

[4] Here $[\boldsymbol{\Phi}, \boldsymbol{\Lambda}]$ is the corresponding Lie algebra in vector representation.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

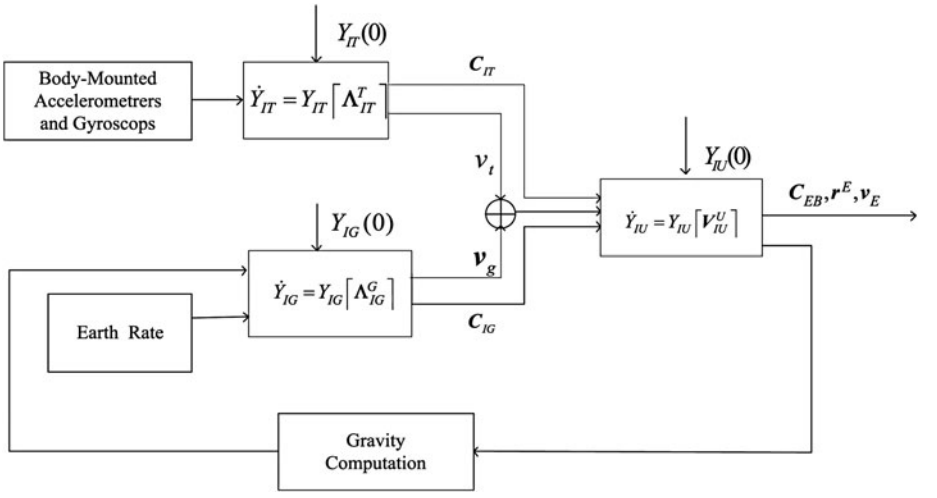


Figure 2. The fundamental mechanism of the strapdown INS algorithm based on Lie group and Lie algebra.

Similar to the calculation of coning and sculling, the general screw correction can be approximated by the sum of all possible Lie brackets formed from N sub-minor sensor-data intervals, making up a minor computational interval. An approximation can be formulated as

$$\Delta\boldsymbol{\Phi}_s = \sum_{i=1}^{N}\sum_{j>i}^{N} K_{ij}\left[\widehat{\boldsymbol{\alpha}}_i, \widehat{\boldsymbol{\alpha}}_j\right] \tag{47}$$

where $\widehat{\boldsymbol{\alpha}}_i$ is the incremental Lie algebra over the i^{th} sub-minor interval; The optimal coefficients K_{ij} are exactly the same as those in conventional algorithm (Ignagni, 2003). For example when $N = 2$, the coefficient $K_{12} = 2/3$.

After the screw vector $\boldsymbol{\Phi}$ is calculated, the numerical solution to the group rate equation can be given as

$$\boldsymbol{Y}(t + \Delta T) = \boldsymbol{Y}(t)\exp(\boldsymbol{\Phi}(\Delta T)) \tag{48}$$

where $\boldsymbol{Y}(t)$ is the previous Lie group and $\boldsymbol{Y}(t + \Delta T)$ is the current one, $\exp(\boldsymbol{\Phi}(\Delta T))$ is the updating Lie group during the major interval.

Since the exponential map contains trigonometric functions which are very hard to compute in real time, approximated truncation of trigonometric series are adopted to simplify it. For example, a fourth-order approximation is given as

$$\sin(\phi) = \phi + \frac{\phi^3}{3!} \quad \cos(\phi) = 1 - \frac{\phi^2}{2!} + \frac{\phi^4}{4!}$$

This approximation is equivalent to adopting the corresponding terms in the series formula given in Section 2.

4. SIMULATIONS.

In Section 2 we demonstrated that $SE(3)$ group is analytically equivalent to dual quaternion. In this section we will show that the new $SE(3)$ based

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

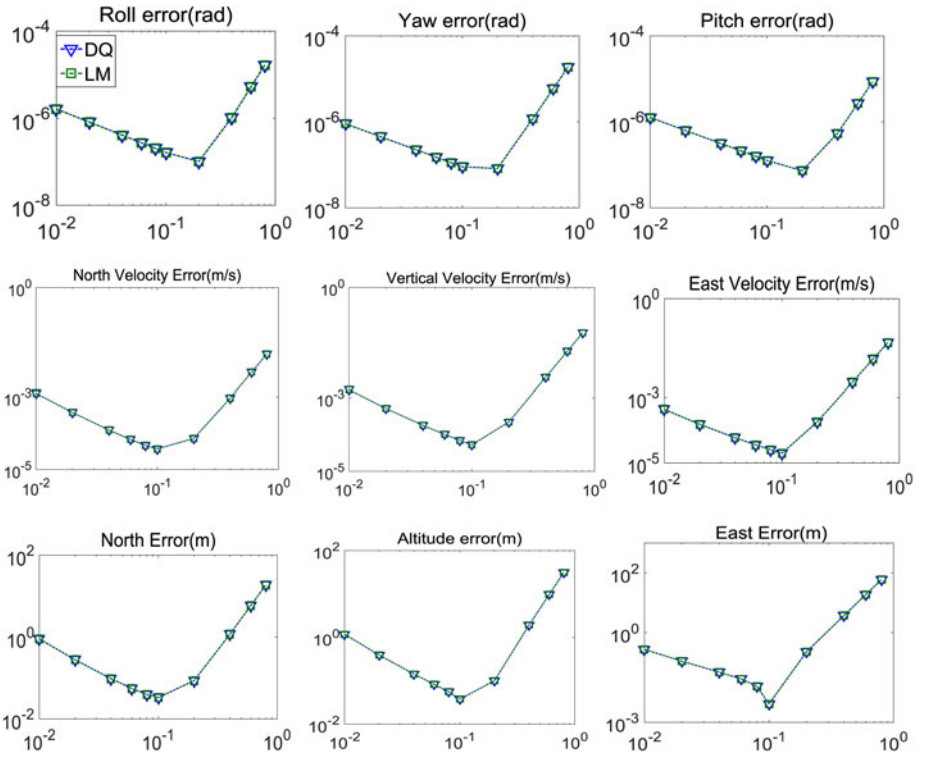


Figure 3. (With ideal inertial sensors) MAEs(log-log plot) in Euler angles/velocity/position as function of input signal frequency.

INS algorithm achieves the same accuracy as DQ numerically. A variety of numerical simulations were conducted by using Matlab2010. The local-level frame is defined with x directing north and y directing upward vertically. In the simulations, we use an ideal trace generator which feeds on two inputs: the local frame acceleration and the Euler angle rate, to produce gyro/accelerometer outputs as well as the vehicle's ideal navigation parameters, namely, attitude angles, velocity vector and position. Algorithm errors are computed according to these ideal navigation parameters.

The inputs of the trace generator are configured as 20 $\sin(2\pi\, ft)(\mathrm{m/s}^2)$ for every channel of ground acceleration and $2\pi f \cos(2\pi ft)$(rad/s) for every channel of Euler angle rate. The sampling frequency of gyro/accelerometer is 100Hz. At the beginning, the vehicle stands still on the ground of the Earth at latitude 30°, longitude 110° and altitude 0 m, and the Euler angles are 0°. Gravitation is calculated in the local-level frame and then transformed to the Earth frame for the sake of consistency (Chatfield, 2013; Wei and Schwarz, 1990). The two-sample algorithm is adopted and a fourth-order truncation of trigonometric series is used in the simulations. The DQ algorithm is conducted by Simulink modules and the new $se(3)$ algorithm is conducted by m-files. In order to make comparisons, the simulation results are transformed from the Earth frame to the local level frame, the transformation algorithm is based on Wu et al. (2003). The simulations are conducted under ideal conditions.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

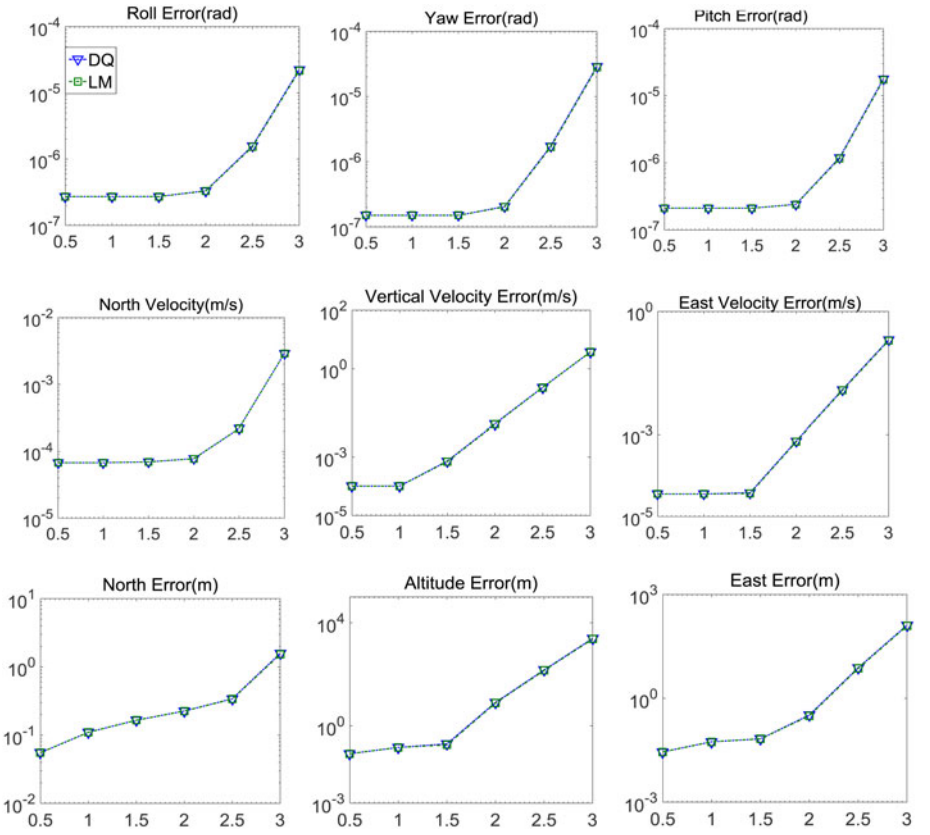


Figure 4. (With ideal inertial sensors) MAEs(log plot) in Euler angles/velocity/position as function of time.

Firstly, the input frequency is set to cover a list of discrete frequencies from 0·01 to 1Hz, i.e. $f = 0{\cdot}01, 0{\cdot}02, 0{\cdot}04, 0{\cdot}06, 0{\cdot}08, 0{\cdot}1, 0{\cdot}2, 0{\cdot}4, 0{\cdot}6, 0{\cdot}8$Hz. The runtime is set to be 1800 seconds. The Maximum Absolute Errors (MAE) are shown in Figure 3 as functions of the input signal frequency. Simulations show that the $SE(3)$ algorithm is similar to the DQ algorithm in accuracy and turning frequency.

Simulations are also conducted to test the long range performance of the algorithms. The input signal frequency is set to 0·06Hz, runtime is set to 0·5 hour, 1 hour, 1·5 hours, 2 hours, 2·5 hours, and 3 hours respectively. Results are shown in Figure 4. Simulations show that after 2 hours of navigation, errors grow faster in time. This is mainly due to the accumulated errors of the INS. We also notice that velocity and position errors in the vertical channel are obviously larger than east and north errors. This is mainly due to the inherent instability of the vertical channel, not the effects of the form of equations and integration methods.

Though the $SE(3)$ algorithm shows no improvement in accuracy, the general Lie group method used in the design process shows its brevity and uniformity. Also $SE(3)$ and $\mathfrak{se}(3)$ are much more direct than DQ, and also keep the mathematical prerequisites to minimum as they are usual matrix and vector operations. However, the

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

new algorithm is not as efficient as the DQ algorithm in computation. But their gap in computation cost is not large, and modern computer technology enables the new algorithm to be viable for real time applications (see Appendix C for a detailed computation cost comparison).

5. CONCLUSIONS. This paper presents a unified method for INS algorithm design by using Lie group and Lie algebra. This method can be applied to Lie subgroups used in INS algorithms including the special orthogonal group $SO(3)$ and the special Euclidean group $SE(3)$, even quaternion and dual quaternion can be treated in this framework. By using this method, the solutions of the differential equations on these Lie subgroups can be easily obtained. Comparing to the previous lengthy geometrical methods, the Lie group method is a simpler unified one.

Based on this method, we developed a new INS algorithm by using $SE(3)$. The principle of inertial navigation is simplified as three group differential equations. We also formulated a new numerical integration algorithm to solve these group equations. Analytical and numerical comparisons show that the new algorithm achieves the same accuracy as the existing DQ algorithm. Though the $SE(3)$ algorithm has no improvement in accuracy, the general Lie group method used in the design process shows its brevity and uniformity. Also the vector and matrix representation of $SE(3)$ and $\mathfrak{se}(3)$ give us a more direct way to treat the rigid body motion than DQ. However, the computation cost of $SE(3)$ algorithm is a little bit larger than DQ, but modern computer technology enables the $SE(3)$ algorithm to be viable for real time applications.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (No.61503403, No.61573371). The authors would like to thank editors and anonymous reviewers for their insightful comments and constructive suggestions.

REFERENCES

Bar-Itzhack, I.Y. (1989). Extension of Euler's Theorem to N-Dimensional Spaces. *IEEE Transactions on Aerospace and Electronic Systems*, **25**, 903–909.

Bortz, J.E. (1971). A New Mathmatical Formulation for Strapdown Inertial Navigation. *IEEE Transactions on Aerospace and Electronic Systems*, **7**, 61–66.

Celledoni, E. and Owren, B. (2003). Lie Group Methods for Rigid Body Dynamics and Time Integration on Manifolds. *Computer Methods in Applied Mechanics and Engineering*, **192**, 421–438.

Chatfield, A.B. (2013). *Fundamentals of High Accuracy Inertial Navigation*. American Institute of Aeronautics and Astronautics.

Ignagni, M.B. (2003). Optimal Strapdown Attitude Integration Algorithms. *Journal of Guidance*, **13**, 363–369.

Iserles, A., Munthe-Kass, H.Z., Norsett, S.P. and Zanna, A. (2000). *Lie Group Methods*. Acta Numerica.

Jiang, Y.F. (1991). On the Rotation Vector Differential Equation. *IEEE Transactions on Aerospace and Electronic Systems*, **27**, 181–183.

Jonghoon, P. and Chung, W.K. (2005). Geometric Integration on Euclidean Group With Application to Articulated Multibody Systems. *IEEE Transactions on Robotics*, **21**, 850–862.

Murray, R.M., Li., Z and Sastry, S.S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc.

Nazaroff, G.J. (1979). The Orientation Vector Differential Equation. *Journal of Guidance and Control*, **2**, 351–352.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

Savage, P.G. (1998a). Strapdown Inertial Navigation Integration Algorithm Design, Part 1: Attitude Algorithms. *Journal of Guidance, Control, and Dynamics*, **21**, 19–28.
Savage, P.G. (1998b). Strapdown Inertial Navigation Integration Algorithm Design, Part 2: Velocity and Position Algorithms. *Journal of Guidance, Control, and Dynamics*, **21**, 208–221.
Savage, P.G. (2006a) Reply by the Author to Y.Wu et al., *Journal of Guidance, Control and Dynamics*, **29**, 1485–1486.
Savage, P.G. (2006b). A Unified Mathematical Framework for Strapdown Algorithm Design. *Journal of Guidance, Control, and Dynamics*, **29**, 237–249.
Varadarajan, V.S. (1984). *Lie Group, Lie Algebra, and Their Representations*. Springer.
Wei, M. and Schwarz, K.P. (1990). A Strapdown Inertial Algorithm using an Earth-fixed Cartesian Frame. *Journal of The Institution of Navigation*, **37**, 153–176.
Wu, D. and Wang, Z.Z. (2012). Strapdown Inertial Navigation System Algorithms Based on Geometric Algebra. *Advances in Applied Clifford Algebra*, **22**, 1151–1167.
Wu, Y.X. (2006). Comment on " A Unified Mathematical Framework for Strapdown Algorithm Design". *Journal of Guidance, Control, and ,Dynamics*, **29**, 1482–1483.
Wu, Y.X., Hu, X.P., Hu, D.W., Li, T. and Lian, J. (2005). Strapdown Inertial Navigation System Algorithms Based on Dual Quaternion. *IEEE Transactions on Aerospace and Electronic Systems*, **41**, 110–132.
Wu, Y.X., Wang, P., Hu, X.P. (2003). Algorithm of Earth-Centered Earth-fixed Coordinates to Geodetic Coordinates. *IEEE Transactions on Aerospace and Electronic Systems*, **39**, 1457–1461.
Wu, Y.X., Xiao, Z.X. (2011) On Position Translation Vector. *Proccedings of AIAA Guidance, Navigation and Control Conference*, Minnesota, USA

APPENDIX A. THE DIFFERENTIAL EQUATION OF $\boldsymbol{\lambda}$

In Equation (34), we have

$$\boldsymbol{D} = \frac{1-\gamma}{\|\boldsymbol{\phi}\|^2}\lceil\boldsymbol{\lambda},\boldsymbol{\phi}\rceil + \frac{\frac{1}{\beta}+\gamma-2}{\|\boldsymbol{\phi}\|^4}(\boldsymbol{\phi}'\boldsymbol{\lambda})\lceil\boldsymbol{\phi}\rceil^2 \qquad \lceil\boldsymbol{\lambda},\boldsymbol{\phi}\rceil = \lceil\boldsymbol{\lambda}\rceil\lceil\boldsymbol{\phi}\rceil + \lceil\boldsymbol{\phi}\rceil\lceil\boldsymbol{\lambda}\rceil$$

$$\gamma = \frac{\phi\sin\phi}{2(1-\cos\phi)} \qquad \frac{1}{\beta} = \frac{\phi^2}{2(1-\cos\phi)}$$

$$d\exp^{-1}_{-\boldsymbol{\phi}} = \boldsymbol{I} + \frac{1}{2}\lceil\boldsymbol{\phi}\rceil + \frac{1}{\phi^2}\left(1 - \frac{\phi\sin\phi}{2(1-\cos\phi)}\right)\lceil\boldsymbol{\phi}\rceil^2$$

Taking the above equations into Equation (34), we have

$$\begin{aligned}\dot{\boldsymbol{\lambda}} &= \left(\boldsymbol{D} + \frac{1}{2}\boldsymbol{\lambda}\right)\boldsymbol{\omega}^O_{NO} + d\exp^{-1}_{-\boldsymbol{\phi}}\boldsymbol{v}^O_N \\ &= \boldsymbol{v}^O_N + \frac{1}{2}(\boldsymbol{\phi}\times\boldsymbol{\lambda} + \boldsymbol{\lambda}\times\boldsymbol{\phi}) + \frac{1}{\phi^2}\left(1 - \frac{\phi\sin\phi}{2(1-\cos\phi)}\right)\left[\boldsymbol{\phi}\times\left(\boldsymbol{\phi}\times\boldsymbol{v}^O_N\right)\right. \\ &\quad \left. + \boldsymbol{\phi}\times\left(\boldsymbol{\lambda}\times\boldsymbol{\omega}^O_{NO}\right) + \boldsymbol{\lambda}\times\left(\boldsymbol{\phi}\times\boldsymbol{\omega}^O_{NO}\right)\right] + \left[\frac{\phi+\sin\phi}{2\phi^3(1-\cos\phi)} - \frac{2}{\phi^4}\right](\boldsymbol{\phi}'\boldsymbol{\lambda})\boldsymbol{\phi}\times\left(\boldsymbol{\phi}\times\boldsymbol{\omega}^O_{NO}\right)\end{aligned}$$

This is exactly the same as the differential equation of the dual part in dual quaternion (Wu, 2006).

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

APPENDIX B. APPLYING THE LIE GROUP METHOD TO DQ. By applying Lemma 1, the solution to the dual quaternion differential Equation (35) can be given as

$$\hat{\boldsymbol{q}}(t) = \hat{\boldsymbol{q}}(0)\exp\left(\frac{\hat{\boldsymbol{\phi}}}{2}\right) \tag{A1}$$

where

$$\exp\left(\frac{\hat{\boldsymbol{\phi}}}{2}\right) = \sum_{j=0}^{\infty}\frac{\left(\frac{\hat{\boldsymbol{\phi}}\times}{2}\right)^n}{j!}$$

The differential equations of $\hat{\boldsymbol{\phi}}$ can be given by using Equation (7) as

$$\dot{\hat{\boldsymbol{\phi}}} = d\exp^{-1}_{\hat{\boldsymbol{\phi}}}\hat{\boldsymbol{\omega}} = \hat{\boldsymbol{\omega}} + \frac{1}{2}\hat{\boldsymbol{\phi}}\times\hat{\boldsymbol{\omega}} + \frac{1}{12}\hat{\boldsymbol{\phi}}\times\left(\hat{\boldsymbol{\phi}}\times\hat{\boldsymbol{\omega}}\right) + \ldots\ldots = \sum_{j=0}^{\infty}\frac{B_j}{j!}\left(\hat{\boldsymbol{\phi}}\times\right)^j\hat{\boldsymbol{\omega}} \tag{A2}$$

The resulting Equation (A1) and (A2) are exactly the same as Equations (43) and (35) in Wu et al. (2005). Comparing the two approaches in deriving the solution to the dual quaternion differential equation, the Lie group method is a much simpler one.

APPENDIX C. COMPARISON OF $SE(3)$ AND DQ ALGORITHM IN COMPUTATION COST. To make the computation cost comparison more clear, we split the $SE(3)$ algorithm into the following equations.

$$\boldsymbol{Y}(t+\Delta T) = \boldsymbol{Y}(t)\exp(\boldsymbol{\Phi}(\Delta T)) \tag{A3}$$

where

$$\exp(\boldsymbol{\Phi}(\Delta T)) = \begin{bmatrix}\Delta\boldsymbol{C} & \Delta\boldsymbol{v}\\ \boldsymbol{0} & 1\end{bmatrix} \tag{A4}$$

$$\Delta\boldsymbol{C} = \exp(\Delta\boldsymbol{\phi}) = \boldsymbol{I} + \frac{\sin(\Delta\phi)}{\Delta\phi}\lceil\Delta\boldsymbol{\phi}\rceil + \frac{1-\cos(\Delta\phi)}{\phi^2}\lceil\Delta\boldsymbol{\phi}\rceil^2 \tag{A5}$$

$$\Delta\boldsymbol{v} = \frac{1}{\phi^2}\left((\boldsymbol{I}-\Omega)(\boldsymbol{\phi}\times) + \boldsymbol{\phi}\boldsymbol{\phi}'\right)\boldsymbol{\eta} \tag{A6}$$

As we can see, the $SE(3)$ group are expressed by usual matrices and vectors, one can develop numerical algorithms for computer implementation directly based on Equations (A3)–(A6).

Dual quaternion consists of the real part and the dual part, and they are computed separately in numerical integration algorithms by using the following equations.

$$\begin{aligned}\boldsymbol{q}(t+\Delta T) &= \boldsymbol{q}(t)\circ\boldsymbol{q}(\Delta T)\\ \boldsymbol{q}'(t+\Delta T) &= \boldsymbol{q}(t)\circ\boldsymbol{q}'(\Delta T) + \boldsymbol{q}'(t)\circ\boldsymbol{q}(\Delta T)\end{aligned} \tag{A7}$$

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press

Table A1. Computation cost comparison of $SE(3)$ and DQ algorithm.

Algorithm	Equation	Scalar Addition	Scalar Multi	Trigonometric	Sqrt-Root
$SE(3)$ Algorithm	(A3)	48	64	0	0
	(A5)	39	51	2	1
	(A6)	26	26	0	0
	total	113	141	2	1
DQ Algorithm	(A7)	40	48	0	0
	(A8)	6	9	2	1
	(A9)	13	21	2	1
	(A10)	12	20	0	0
	total	71	98	4	2

where the real part $\boldsymbol{q}(\Delta T)$ is

$$\boldsymbol{q}(\Delta T) = \cos\left(\frac{\sigma}{2}\right) + \frac{\boldsymbol{\sigma}}{\sigma}\sin\left(\frac{\sigma}{2}\right) \tag{A8}$$

And the dual part $\boldsymbol{q}'(\Delta T)$ is

$$\boldsymbol{q}'(\Delta T) = -\frac{\sigma'}{2}\sin\left(\frac{\sigma}{2}\right) + \frac{\sigma\boldsymbol{\sigma}' - \sigma'\boldsymbol{\sigma}}{\sigma^2}\sin\left(\frac{\sigma}{2}\right) + \frac{\sigma'\boldsymbol{\sigma}}{2\sigma}\cos\left(\frac{\sigma}{2}\right) \tag{A9}$$

The resulting $\Delta\boldsymbol{v}$ is

$$\Delta\boldsymbol{v} = 2\boldsymbol{q}'(\Delta T)\circ\boldsymbol{q}^{*}(\Delta T) \tag{A10}$$

As discussed earlier $\boldsymbol{\phi}$ and $\boldsymbol{\sigma}$ are indeed the rotation vectors and calculated as $\Delta\boldsymbol{\phi} = \boldsymbol{\sigma} = \sum_{i=1}^{2}\Delta\boldsymbol{\theta}_i + \frac{2}{3}\Delta\boldsymbol{\theta}_1 \times \Delta\boldsymbol{\theta}_2$, the translation vectors $\Delta\boldsymbol{\eta}$, $\boldsymbol{\sigma}'$ are the same and calculated as $\Delta\boldsymbol{\eta} = \boldsymbol{\sigma}' = \frac{2}{3}(\Delta\boldsymbol{\theta}_1 \times \Delta\boldsymbol{v}_2 + \Delta\boldsymbol{v}_1 \times \Delta\boldsymbol{\theta}_2)$. We will only make computation cost comparisons of Equations (A3)–(A6) and Equations (A7)–(A10).

We evaluate the computation cost by the number of scalar additions, scalar multiplications, trigonometric functions and square root operations. For example, when given a vector $\boldsymbol{\omega} = (\omega_1; \omega_2; \omega_3)$, the norm is calculated as

$$\omega = \sqrt{\omega_1\omega_1 + \omega_2\omega_2 + \omega_3\omega_3}$$

which requires three scalar multiplications, two scalar additions and one square root operation. In this manner, the computation cost of $SE(3)$ algorithm and DQ are given by Table A1.

In the same computation sources, a single computation of $SE(3)$ algorithm takes 0·00023 seconds and DQ takes 0·00018 seconds.

Both the computation cost analysis and simulations show that DQ is more efficient than the $SE(3)$ algorithm, but their gap is not large. In fact, the difference between the two algorithms in computation cost stems from the different representation of the attitude part. In modern computer technology, most embedded systems are applicable to direction cosine matrix representation algorithms.

https://doi.org/10.1017/S037346331600045X Published online by Cambridge University Press