CrossMark

The Journal of Navigation (2021), **74**:4 763–781
doi:10.1017/S0373463321000023

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Attitude reconstruction from strap-down rate gyros using power series

Habib Ghanbarpourasl

Mechatronics Department, University of Turkish Aeronautical Association, Ankara, Turkey.
E-mail: habib.ghanbarpour@gmail.com

Received: 02 April 2020; **Accepted:** 10 January 2021; **First published online:** 4 March 2021

Keywords: attitude reconstruction, time varying linear systems, quaternion update, quaternions' numerical integration, coning error compensation

Abstract

This paper introduces a power series based method for attitude reconstruction from triad orthogonal strap-down gyros. The method is implemented and validated using quaternions and direction cosine matrix in single and double precision implementation forms. It is supposed that data from gyros are sampled with high frequency and a fitted polynomial is used for an analytical description of the angular velocity vector. The method is compared with the well-known Taylor series approach, and the stability of the coefficients' norm in higher-order terms for both methods is analysed. It is shown that the norm of quaternions' derivatives in the Taylor series is bigger than the equivalent terms coefficients in the power series. In the proposed method, more terms can be used in the power series before the saturation of the coefficients and the error of the proposed method is less than that for other methods. The numerical results show that the application of the proposed method with quaternions performs better than other methods. The method is robust with respect to the noise of the sensors and has a low computational load compared with other methods.

1. Introduction

There are two types of implementation of inertial navigation systems: stable platform and strap-down systems. The stabilised platforms have complex electro-mechanical components for stabilising gyroscopes and accelerometers, but the navigation algorithms of such systems are not complex. On the contrary, the strap-down systems have simple hardware, but they use complex algorithms. These algorithms suffer from the numerical solution of navigation differential equations and have been developed over decades. The development of strap-down systems was delayed up to the 1960s, when digital technology emerged and computers were developed (Jekeli, 2012). The critical point in developing the strap-down systems was reconstructing the attitude from the output of three orthogonal rate gyroscopes (Minor, 1965; McKern, 1968; Garg et al., 1978). From 1950 to 1960, two types of algorithms were used. The first was the first-order algorithm with high-speed updating (10–20 kHz) and the second used higher-order algorithms with low frequency updates (50–100 Hz) (Savage, 1998a, 1998b). Algorithms that sampling data from sensors with high frequency and update the attitude with low frequency are called two-speed algorithms. The two-speed algorithms had moderate accuracy and high speeds with Picard-type recursive integration, which was introduced by Savage (1966). With the progress of digital computers, Wilcox (1967) and McKern (1968) introduced third-order algorithms.

The concept of rotation vector was first used by Jordan (1969) and Bortz (1971). Gilmore (1980) developed an algorithm that used the Bortz technique with high frequency updating of rotation vector, and subsequent updating of quaternions with low frequency. Miller (1983) used second-degree polynomials

© The Royal Institute of Navigation 2021

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

for the angular velocity with three data and solved the differential equations of rotation vector and then quaternions are updated. Lee followed Miller's work with more data and prediction of coning motion (Lee et al., 1990). Later, Miller's work was optimised and improved by Ignagni (1990, 1996) and Mussoff (Musoff and Murphy, 1995). Savage published two important papers in 1998. After reviewing other works in the first paper, he focused on the attitude integration algorithms based on two-speed updating (Savage, 1998a), and in the second one, he investigated the differential equations of velocity and position and introduced an analytical method for sculling motion (Savage, 1998b). At that time, there was some focus on the sculling algorithms (Ignagni, 1998; Savage, 1998a, 1998b). Ignagni developed a class of sculling algorithms for compensation and optimisation of angular velocity and showed the duality of sculling and coning motions (Ignagni, 1998). Roscoe then showed that these motions were convertible to each other with a simple formula (Roscoe, 2001).

In recent years, another round of optimisation of the attitude-integrating algorithms has been started (Savage, 2010, 2017; Song et al., 2013). This need is due to the recent advancements in military projectiles, which have high-speed rotations, while algorithms still need to be developed for cold atom inference gyroscopes (Wu and Yan, 2019). Some recent publications used Picard iteration with fitted angular velocity polynomial to solve the attitude integration problem with different attitude parameters (Yan et al., 2017; Wu, 2018; Wu et al., 2018). Using a Chebyshev polynomial for fitted angular velocity and quaternion or rotation vector has shown better results (Wu and Yan, 2019). Wu has further applied Taylor series for attitude reconstruction (Wu and Litmanovich, 2019). Using the Taylor series for a solution of any differential equation has difficulties, such as the need to calculate the higher-order derivatives of a parameter (Hairer et al., 1993); however, in the case of attitude differential equation, the derivative of the quaternion/direction cosine matrix can be calculated using the Leibniz rule (Wu and Litmanovich, 2019). Another example of such a method has been presented with low-order expansion for attitude and angular velocity series (Miller, 1983). Most recent works, which are based on the Taylor series (Yan et al., 2017), use direction cosine matrix (DCM) for parameterisation of attitude.

This paper introduces a power series based solution for the attitude differential equation. It investigates the advantages and disadvantages of Taylor's polynomial and power series for the reconstruction of attitude from angular velocity when it is fitted by a polynomial. Two methods of attitude representation are examined – quaternion and DCM – because in these cases, the differential equations are pseudo-linear (multiplication of two variables) and one can simply calculate the successive derivatives of attitude parameters. In the case of rotation vector and Euler angles, the differential equations are nonlinear and the calculation of successive derivatives is impossible. The algorithms are implemented in single and double precision forms to find which way is best for real applications because in real applications there are limited bits for representation of variables and the round off errors cause errors in the integrating process. When the norm of the coefficients of Taylor or power series is beyond the range of variables, other terms are ignored. The paper is organised as follows. Section 2 briefly discusses quaternions and DCM and those differential equations. Section 3 is devoted to solving and analysing the differential equation of quaternion and DCM using power series, while Section 4 introduces Taylor series based on the solution for those differential equations. Section 5 evaluates performance of methods via coning motions. Finally, a brief summary is given in Section 6.

2. Quaternion

A quaternion is a composition of four elements, which are used for the representation of attitude between two frames. Sometimes it is represented by a 4×1 unit vector (noncommutative) as given below (Jekeli, 2012):

$$\boldsymbol{q} = \begin{bmatrix} s \\ \boldsymbol{V} \end{bmatrix} \tag{1}$$

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

In this representation method, s is a scalar, and $\boldsymbol{V}$ is a 3×1 vector. The conjugate of a quaternion is defined as:

$$\boldsymbol{q}^* = \begin{bmatrix} s \\ -\boldsymbol{V} \end{bmatrix} \tag{2}$$

The multiplication of the two quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ is represented by $\boldsymbol{q}_1 \otimes \boldsymbol{q}_2$, and in the matrix multiplication form, it is written as:

$$\boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = \boldsymbol{q}_2 \odot \boldsymbol{q}_1 = [\boldsymbol{q}_1]_L \boldsymbol{q}_2 = [\boldsymbol{q}_2]_R \boldsymbol{q}_1 \tag{3}$$

where $[\boldsymbol{q}_1]_L$ and $[\boldsymbol{q}_2]_R$ are 4×4 matrices, as shown below:

$$[\boldsymbol{q}_1]_L = \begin{bmatrix} s_1 & -\boldsymbol{V}_1^T \\ \boldsymbol{V}_1 & s_1\boldsymbol{I} + [\boldsymbol{V}_1\times] \end{bmatrix}, \quad [\boldsymbol{q}_2]_R = \begin{bmatrix} s_2 & -\boldsymbol{V}_2^T \\ \boldsymbol{V}_2 & s_2\boldsymbol{I} - [\boldsymbol{V}_2\times] \end{bmatrix} \tag{4}$$

where $[\boldsymbol{V}\times]$ is skew-symmetric 3×3 and is named cross product matrix of vector $\boldsymbol{V}$. The indices i and b are used for representation of inertial and body frames. The angular velocity of the body frame relative to the inertial frame, which is expressed in the body frame, is denoted by $\boldsymbol{\omega}^b$. For the representation of a vector as a quaternion, we use the notation $\boldsymbol{\omega}^{b'} = \left[\begin{smallmatrix} 0 \\ \boldsymbol{\omega}^b \end{smallmatrix}\right]$. The projection of that vector onto the inertial frame using quaternion is represented by (Jekeli, 2012):

$$\boldsymbol{\omega}^{i'} = \boldsymbol{q} \otimes \boldsymbol{\omega}^{b'} \otimes \boldsymbol{q}^* = \begin{bmatrix} 0 \\ \boldsymbol{C}_b^i \boldsymbol{\omega}^b \end{bmatrix} \tag{5}$$

where $\boldsymbol{C}_b^i$ is the rotation matrix from b-frame to i-frame while the rotation matrix is a function of the quaternion, as given below (Jekeli, 2012):

$$\begin{bmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{C}_b^i \end{bmatrix} = [\boldsymbol{q}]_L [\boldsymbol{q}^*]_R \tag{6}$$

The differential equation of quaternion for attitude dynamics is (Savage, 1998a, 1998b):

$$\dot{\boldsymbol{q}} = \frac{1}{2} \boldsymbol{q} \otimes \boldsymbol{\omega}^{b'} \tag{7}$$

And the equivalent equation for the rotation matrix is (Savage, 1998a, 1998b):

$$\dot{\boldsymbol{C}}_b^i = \boldsymbol{C}_b^i [\boldsymbol{\omega}^b\times] \tag{8}$$

The quaternion as a function of rotation vector is (Jekeli, 2012):

$$\boldsymbol{q} = \begin{bmatrix} \cos(\varphi/2) \\ \sin(\varphi/2)\hat{\boldsymbol{n}} \end{bmatrix} \tag{9}$$

where φ is rotation angle, and $\hat{\boldsymbol{n}}$ is a unit vector that shows the direction of rotation vector.

In real applications, the angular velocity is measured with high sampling frequency as much as possible and an approximated polynomial is fitted to it. In some cases, the output of sensors is incremental angles (integral of the angular velocity is measured in the incremental time) (Wu and Yan, 2019). Fitting a polynomial on high frequency sampled data has some benefits because it is able to reduce the white noise of sensors and describes the angular velocity as a continuous function of time. Suppose the given polynomial describes the angular velocity between times zero and t (Wu and Yan, 2019).

$$\boldsymbol{\omega}^b = 2(\boldsymbol{w}_0 + \boldsymbol{w}_1 t + \ldots + \boldsymbol{w}_n t^n) \tag{10}$$

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

where t is time and $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_n$ are the coefficient vectors of the fitted polynomial. When there are more than $m \geq n+1$ data, the least square (LS) estimation method helps us to calculate the unknown parameters $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_n$ as given below (Wu and Litmanovich, 2019; Wu and Yan, 2019):

$$\begin{bmatrix} \omega^b(t_0)^T \\ \omega^b(t_1)^T \\ \cdots \\ \omega^b(t_m)^T \end{bmatrix} = 2 \begin{bmatrix} 1 & t_0 & \ldots & t_0^n \\ 1 & t_1 & \ldots & t_1^n \\ \ldots & \ldots & \ldots & \ldots \\ 1 & t_m & \ldots & t_m^n \end{bmatrix} \begin{bmatrix} w_0^T \\ w_1^T \\ \ldots \\ w_n^T \end{bmatrix} \tag{11}$$

where, $\boldsymbol{\omega}^b(t_0), \boldsymbol{\omega}^b(t_1), \ldots, \boldsymbol{\omega}^b(t_m)$ are sample data in times $t_0, t_1, \ldots, t_m$. When n is high, the coefficient matrix in Equation (11) will be singular, then QR-factorisation or singular value decomposition are useful methods to solve this problem. When there are correlations between gyros' noises, the weighted LS of errors as a goal function is more efficient. In that condition, the weighting is selected as an inverse of the covariance matrix. Here we avoid discussing this issue because it has been discussed in many classic texts.

3. Attitude reconstruction using power series

Using notation Equation (4) in Equation (7), the differential equation of quaternions in the matrix form is:

$$\dot{\boldsymbol{q}} = [0.5\omega^{b'}]_R \boldsymbol{q} \tag{12}$$

Substituting Equation (10) in Equation (12), we obtain the following results:

$$\dot{\boldsymbol{q}} = \boldsymbol{\Omega}(t)\boldsymbol{q}, \boldsymbol{\Omega}(t) = (\boldsymbol{\Omega}_0 + \boldsymbol{\Omega}_1 t + \ldots + \boldsymbol{\Omega}_n t^n), \boldsymbol{q}(0) = \boldsymbol{q}_0 \tag{13}$$

where n is the degree of angular velocity's polynomial and the matrices $\boldsymbol{\Omega}_0, \boldsymbol{\Omega}_1, \ldots, \boldsymbol{\Omega}_n$ are constant and known matrices with 4×4 dimensions, and they are defined as follows:

$$\boldsymbol{\Omega}_i = [\boldsymbol{w'}_i]_R, i = 0, 1, \ldots, n \tag{14}$$

Now, we examine a power series solution for Equation (13) with a degree of the infinite as:

$$\boldsymbol{q}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \ldots + \mathbf{a}_i t^i + \ldots \tag{15}$$

where $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_i, \ldots$ are constant and unknown vectors with the dimensions 4×1. After substituting Equation (15) in Equation (13) and rearranging the coefficient t^i, $(i = 0, 1, \ldots)$, the following equations are extracted:

$$\mathbf{a}_i = \begin{vmatrix} \frac{1}{i}\left(\sum_{j=0}^{i-1} \boldsymbol{\Omega}_j \mathbf{a}_{i-j-1}\right), & 1 \leq i \leq n \\ \frac{1}{i}\left(\sum_{j=0}^{n} \boldsymbol{\Omega}_j \mathbf{a}_{i-j-1}\right), & i > n \end{vmatrix} \tag{16}$$

In the above equation, one can determine the coefficient vectors $\mathbf{a}_i$, recursively. The constant vector $\mathbf{a}_0$ is equal to initial condition $\boldsymbol{q}_0$. When $i \leq n$, by using the first n Equation (16), the coefficients

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

$\mathbf{a}_1, \ldots, \mathbf{a}_n$ are determined. When $i > n$, the following sequence can be extracted:

$$\begin{bmatrix} \mathbf{a}_{i-n} \\ \mathbf{a}_{i-n+1} \\ \mathbf{a}_{i-n+2} \\ \ldots \\ \mathbf{a}_{i-1} \\ \mathbf{a}_i \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \ldots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \mathbf{0} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{I} \\ \frac{1}{i}\boldsymbol{\Omega}_n & \frac{1}{i}\boldsymbol{\Omega}_{n-1} & \frac{1}{i}\boldsymbol{\Omega}_{n-2} & \ldots & \frac{1}{i}\boldsymbol{\Omega}_1 & \frac{1}{i}\boldsymbol{\Omega}_0 \end{bmatrix} \begin{bmatrix} \mathbf{a}_{i-n-1} \\ \mathbf{a}_{i-n} \\ \mathbf{a}_{i-n+1} \\ \ldots \\ \mathbf{a}_{i-2} \\ \mathbf{a}_{i-1} \end{bmatrix} \tag{17}$$

The above system is a linear iteration varying discrete system and the state vector of the system is $\boldsymbol{\chi}_i = [\boldsymbol{a}_{i-n}^T, \ldots, \boldsymbol{a}_i^T]^T$ which contains $n+1$ coefficient vectors of the quaternions' power series. If the system matrix of the above equation is represented by $\mathbf{M}_{i-1}$, then Equation (17) is summarised by:

$$\boldsymbol{\chi}_i = \mathbf{M}_{i-1}\boldsymbol{\chi}_{i-1},\ \boldsymbol{\chi}_n = [\boldsymbol{a}_0^T, \ldots, \boldsymbol{a}_n^T]^T, i > n \tag{18}$$

All blocks inside the matrix $\mathbf{M}_{i-1}$ are 4×4 matrices. The dimension of the matrix $\mathbf{M}_{i-1}$ is $4(n+1) \times 4(n+1)$. Again, we write Equation (15) as a sequence:

$$\tilde{\boldsymbol{q}}_i = \tilde{\boldsymbol{q}}_{i-1} + \mathbf{a}_i t^i, \tilde{\boldsymbol{q}}_n = \boldsymbol{a}_0 + \ldots \boldsymbol{a}_n t^n, i > n \tag{19}$$

Here $\tilde{\boldsymbol{q}}_i$ is an approximation of $\boldsymbol{q}(t)$ up to the ith term. Substituting $\mathbf{a}_i$ from the last equation of (17) into Equation (19), one can write:

$$\tilde{\boldsymbol{q}}_i = \tilde{\boldsymbol{q}}_{i-1} + \boldsymbol{H}_{i-1}\boldsymbol{\chi}_{i-1}, \boldsymbol{H}_{i-1} = t^i \begin{bmatrix} \frac{1}{i}\boldsymbol{\Omega}_n & \ldots & \frac{1}{i}\boldsymbol{\Omega}_0 \end{bmatrix}, i > n \tag{20}$$

where $\tilde{\boldsymbol{q}}_i$ is the approximation of quaternion by power series when the truncation order is i. It is obvious that for a fixed time $t < 1$ and initial condition $\tilde{\boldsymbol{q}}_n$, sequence 20 will converge when:

$$\lim_{i \to \infty} ||\boldsymbol{H}_{i-1}\boldsymbol{\chi}_{i-1}|| = 0 \tag{21}$$

Equation (21) shows that when i approaches infinity, $\boldsymbol{H}_\infty$ will approach zero matrix because $\lim_{i \to \infty} {}^{t^i}/_{i} = 0$. Furthermore, the norm of $\boldsymbol{\chi}_\infty$ is zero too (see theorem 1), then the stability of Equation (20) will be guaranteed.

Theorem 1. *For a fixed initial condition $\boldsymbol{\chi}_n$, the sequence in Equation (17) is asymptotically stable, and it will converge to zero when the number of iterations approaches infinity.*

Proof. The iteration varying linear system (18) is termed asymptotically stable with equilibrium point zero, if for any bounded initial condition $\boldsymbol{\chi}_n$, we have $\lim_{i \to \infty} ||\boldsymbol{\chi}_i|| = ||\boldsymbol{\chi}_\infty|| = 0$ then $\boldsymbol{\chi}_\infty = \lim_{i \to \infty} \boldsymbol{\chi}_i = 0$. Using Equation (18) one can write,

$$\boldsymbol{\chi}_\infty = (\mathbf{M}_\infty \ldots \mathbf{M}_\infty \ldots \mathbf{M}_{n+1}\mathbf{M}_n)\boldsymbol{\chi}_n \tag{22}$$

For asymptotical stability, it needs $||\mathbf{M}_\infty \ldots \mathbf{M}_\infty \ldots \mathbf{M}_{n+1}\mathbf{M}_n|| \to 0$ as $i \to \infty$. Here we consider norm two $||.||_2$ of a matrix, the maximum singular value has to be zero when the number of iterations approaches infinity. When the norm of $\mathbf{M}_\infty$ is zero (see Appendix A), then the norm of $\boldsymbol{\chi}_\infty$ for any initial condition $\boldsymbol{\chi}_n$ is zero, consequently $\boldsymbol{\chi}_\infty = 0$. □

As a result, the state Equation (18) is stable and it will converge to zero for any bounded initial condition. It is important to mention that the maximum value of $||\boldsymbol{\chi}_i||$ can be large but it will remain limited in this case. It can be easily checked that Equation (20) satisfies Equation (13) when i approaches infinity, $(d/dt)\tilde{\boldsymbol{q}}_\infty(t) = \boldsymbol{\Omega}(t)\tilde{\boldsymbol{q}}_\infty(t)$ when there is no error in the initial condition $\boldsymbol{\chi}_n$, that means $\tilde{\boldsymbol{q}}_\infty$

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

converges to true quaternion. It means $\mathbf{a}_0$ is known and other $n-1$ coefficients $\mathbf{a}_1, \ldots, \mathbf{a}_{n-1}$ are calculated by the first equation of Equation (16) recursively. In the same way, we consider a power series for DCM as given below:

$$\boldsymbol{C}_b^i = \boldsymbol{C}_0 + \boldsymbol{C}_1 t + \ldots + \boldsymbol{C}_i t^i + \ldots \tag{23}$$

where $\boldsymbol{C}_0$ is the rotation matrix at time zero and $\boldsymbol{C}_i, i = 1, 2, \ldots$ are constant matrices. Substituting Equation (23) into Equation (8), and comparing equivalent terms, it is possible to determine the coefficient matrices sequentially, which gives the following results:

$$\boldsymbol{C}_i = \begin{cases} \frac{1}{i}\left(\sum_{j=0}^{i-1} \mathbf{C}_{i-j-1}\mathbf{W}_j\right), & 1 \le i \le n \\ \frac{1}{i}\left(\sum_{j=0}^{n} \mathbf{C}_{i-j-1}\mathbf{W}_j\right), & i > n \end{cases} \tag{24}$$

where $\mathbf{W}_j = \frac{1}{2}[\boldsymbol{w}_0\times], i = 1, \ldots, n$ are constant matrices, and the coefficient matrices $\boldsymbol{C}_i, i = 1, 2, \ldots$ are determined by using Equation (24) recursively. In the above equation, when i goes to infinity, $\boldsymbol{C}_i$ will go to zero and Equation (24) will converge into a true DCM. Here, we are avoiding the repetition of the same results that are derived for quaternion expansion.

4. Attitude reconstruction using Taylor series

The well-known Taylor series has been used for approximation of an infinitely differentiable function for centuries. Some researchers have used this method for reconstructing quaternion form rate gyro output. Unfortunately, the Taylor series requires calculating the derivative of the quaternions infinite times, but due to the structure of quaternion dynamics, we can derive an explicit method for this purpose. Here we consider the Taylor expansion of the quaternion as a function of time as (Hairer et al., 1993; Kunovský, 2015):

$$\boldsymbol{q}(t) = \boldsymbol{q}_0 + \sum_{i=1}^{\infty} \frac{t^i}{i!} \boldsymbol{q}^{(i)}|_{t=0} \tag{25}$$

where $\boldsymbol{q}^{(i)}|_{t=0}$ is the ith derivative of quaternion that evaluated at time zero. Using Equation (13) and the Leibniz formula we can write (Hairer et al., 1993):

$$\boldsymbol{q}^{(i)} = \sum_{j=0}^{i-1} \boldsymbol{C}_j^{i-1} \boldsymbol{\Omega}^{(j)}|_{t=0} \boldsymbol{q}^{(i-j-1)}|_{t=0} \tag{26}$$

where $\boldsymbol{C}_j^{i-1} = (((i-1)!)/(j!(i-j-1)!))$ denotes the number of j combinations of $i-1$ elements and $\boldsymbol{\Omega}^{(j)}|_{t=0}$ represents the jth derivative of $\boldsymbol{\Omega}(t)$ at time zero. Using the power series Equation (10), we can calculate the derivatives of angular velocity as:

$$\begin{cases} \boldsymbol{\Omega}^{(j)}|_{t=0} = j!\boldsymbol{\Omega}_j \, j \le n \\ \boldsymbol{\Omega}^{(j)}|_{t=0} = 0 \, j > n \end{cases} \tag{27}$$

Substituting Equation (27) into Equation (26), we can write,

$$\boldsymbol{q}^{(i)}|_{t=0} = \begin{cases} \sum_{j=0}^{i-1} P_j^{i-1} \boldsymbol{\Omega}_j \boldsymbol{q}^{(i-j-1)}|_{t=0} & 1 \le i \le n \\ \sum_{j=0}^{n} P_j^{i-1} \boldsymbol{\Omega}_j \boldsymbol{q}^{(i-j-1)}|_{t=0} & i < n \end{cases} \tag{28}$$

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

where $P_j^{i-1} = (((i-1)!)/((i-j-1)!))$ is the number of permutations of $i-1$ objects taken j at a time. Equation (28) is used to calculate all the derivatives of quaternion recursively at time zero. For simplicity, we will neglect $|_{t=0}$ in $q^{(i)}|_{t=0}$, as all the derivatives are evaluated at time zero. Then Equation (28) becomes,

$$\boldsymbol{q}^{(i)} = \begin{vmatrix} \sum_{j=0}^{i-1} \boldsymbol{P}_j^{i-1} \boldsymbol{\Omega}_j \boldsymbol{q}^{(i-1-j)} & 1 \le i \le n \\ \sum_{j=0}^{n} \boldsymbol{P}_j^{i-1} \boldsymbol{\Omega}_j \boldsymbol{q}^{(i-1-j)} & i < n \end{vmatrix} \tag{29}$$

Rearranging Equation (29) in the recursive form, we obtain:

$$\begin{gathered} \boldsymbol{\Upsilon}_i = \boldsymbol{\mathcal{N}}_{i-1} \boldsymbol{\Upsilon}_{i-1}, \; \boldsymbol{\Upsilon}_i = \begin{bmatrix} (\boldsymbol{q}^{(i-n)})^T & \dots & (\boldsymbol{q}^{(i)})^T \end{bmatrix}^T \\ \boldsymbol{\mathcal{N}}_{i-1} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 & \boldsymbol{I} \\ \boldsymbol{P}_n^{i-1} \boldsymbol{\Omega}_n & \boldsymbol{P}_{n-1}^{i-1} \boldsymbol{\Omega}_{n-1} & \boldsymbol{P}_{n-2}^{i-1} \boldsymbol{\Omega}_{n-2} & \dots & \boldsymbol{P}_1^{i-1} \boldsymbol{\Omega}_1 & \boldsymbol{P}_0^{i-1} \boldsymbol{\Omega}_0 \end{bmatrix} \end{gathered} \tag{30}$$

In the above equation, when $i > n$, it is possible to calculate the time derivatives of the quaternion in a recursive form. The system matrix $\mathbf{N}_{i-1}$ is an iteration varying matrix and it is used to calculate the ith derivative of quaternion at time zero from $n+1$ last derivatives. Again, Equation (25) can be written in the recursive form:

$$\begin{gathered} \bar{\boldsymbol{q}}_i(t) = \bar{\boldsymbol{q}}_{i-1}(t) + \boldsymbol{L}_{i-1} \boldsymbol{\Upsilon}_{i-1} \\ \boldsymbol{L}_{i-1} = t^i \begin{bmatrix} F_n^{i-1} \boldsymbol{\Omega}_n & \dots & F_0^{i-1} \boldsymbol{\Omega}_0 \end{bmatrix}^T, i > n \end{gathered} \tag{31}$$

where $\bar{\boldsymbol{q}}_i(t)$ is an approximation of quaternion at time t, truncated up to i iterations, and $F_n^{i-1} = ((1)/(i(i-n-1)!))$ is a constant scalar. For stability analysis of the system presented by Equation (30), matrices $\boldsymbol{\Omega}_0, \dots, \boldsymbol{\Omega}_n$ are constant and skew-symmetric matrices, when i approaches infinity, all the permutation coefficients will go to infinity and then the norm of $\mathbf{N}_\infty$ will become infinite. Consequently, Equation (30) is unstable, because when one element of a matrix goes to infinity, the norm of the matrix approaches infinity too. Then, contrary to the time series method, the norm of the quaternion's derivative for higher degrees grows unboundedly. It can be easily checked that Equation (31) satisfies Equation (13) when i approaches infinity, then $(d/dt)\bar{\boldsymbol{q}}_\infty(t) = \boldsymbol{\Omega}(t)\bar{\boldsymbol{q}}_\infty(t)$ and that means $\bar{\boldsymbol{q}}_\infty$ will converge to true quaternion. Although the $\bar{\boldsymbol{q}}_\infty$ approaches true quaternion, the norm of $\boldsymbol{\Upsilon}_{i-1}$ grows unboundedly in Equation (31) but the norm of $\boldsymbol{L}_{i-1}$ approaches zero. This issue will generate a problem in real implementations.

Again, like quaternions, we consider the Taylor series of rotation matrix as given below:

$$\boldsymbol{C}(t) = \boldsymbol{C}_0 + \sum_{i=1}^{\infty} \frac{t^i}{i!} \boldsymbol{C}^{(i)}|_{t=0} \tag{32}$$

where $\boldsymbol{C}^{(i)}|_{t=0}$ is ith derivative of the rotation matrix at time zero. Using the Leibniz formula:

$$\boldsymbol{C}^{(i)} = \sum_{j=0}^{i-1} C_j^{i-1} \boldsymbol{C}^{(i-j-1)}|_{t=0} [\boldsymbol{\omega}^{(j)}|_{t=0} \times] \tag{33}$$

where $\boldsymbol{C}^{(i-j-1)}|_{t=0}$ is the $(i-j-1)$th derivative of the rotation matrix at time zero and $[\boldsymbol{\omega}^{(j)}|_{t=0} \times]$ is a skew-symmetric matrix of the jth derivative of angular velocity at time zero.

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

The error of ith order Taylor series of Equation (25) is (Hairer et al., 1993; Wu and Yan, 2019).

$$\begin{aligned}\bar{\boldsymbol{q}}_i(t) &= \boldsymbol{q}_0 + \sum_{j=1}^{i} \frac{t^j}{j!}\boldsymbol{q}^{(j)}|_{t=0} + \boldsymbol{R}(\tau),\\ \boldsymbol{R}_i(\tau) &= \frac{\tau^{i+1}}{(i+1)!}\boldsymbol{q}^{(i+1)}|_{t=0}, \tau = [0t]\end{aligned} \tag{34}$$

where $\boldsymbol{R}_i(\tau)$ is the remainder terms after the ith term, which is the approximation error of the Taylor polynomial and, τ is between zero and t. The Taylor theorem describes the asymptomatic behaviour of the remainder term, then (Hairer et al., 1993):

$$\lim_{i\to\infty} \boldsymbol{R}_i(\tau) = \boldsymbol{0} \tag{35}$$

Comparing Equation (31) with Equation (34), we can write:

$$\boldsymbol{R}_i(\tau) = \tau^i \left[F_n^{i-1}\boldsymbol{\Omega}_n \ \ldots \ F_0^{i-1}\boldsymbol{\Omega}_0 \right] \boldsymbol{\Upsilon}_{i-1} \tag{36}$$

Furthermore, it is obvious that:

$$\lim_{i\to\infty} F_j^{i-1} = 0, j = 0, \ldots, n \tag{37}$$

Using Equation (35), the limit of all terms of Equation (36) has to be zero.

$$\lim_{i\to\infty} F_j^{i-1}\boldsymbol{\Omega}_j \boldsymbol{q}^{(i-j-1)} = 0, j = 0, \ldots, n \tag{38}$$

where $\boldsymbol{\Omega}_j$ is a constant matrix and then the limit of $F_j^{i-1}q^{(i-j-1)}$ for any j has to be zero, but the third term in Equation (40) goes to infinity while the first term goes to zero, when the i approaches infinity. Then the speed of F_j^{i-1} becomes zero, which is faster than the norm of $q^{(i-j-1)}$ going to infinity. Unfortunately, for the Taylor approximation we need to calculate the derivatives of the quaternion at time t. In the case of quaternions of higher-order derivatives, there is unbounded growth due to the instability of Equation (30), when the angular velocity is non-zero. We can say that the implementation of the Taylor series is internally unstable and perhaps there is numerical degeneracy.

If we continue with the Taylor series with more terms to reduce the remainder term i in Equation (34), finally the one element of $q^{(i)}$ will be bigger than the β or smaller than the $-\beta$. Here β is the largest value and $-\beta$ has the lowest value specified to any variable. For example, when using double precision data type, the largest value is $1.79769(10^{308})$, but for single precision data it is $3.4028235(10^{38})$ (Muller et al., 2018). When $q^{(i)} > \beta$, this condition can generate more errors and it needs to cut the Taylor series up to $i-1$ terms. A simple way to control this degeneration is to consider i terms of Equation (31), in which the norm of $q^{(i)}$is lower than the β, before adding ith term to the Taylor series.

5. Simulation study

To study the performances of algorithms and compare the results with recently published algorithms, we consider the classic coning motion that has been mentioned in the literature (Wu, 2018; Wu et al., 2018; Wu and Yan, 2019). The angular velocity of the classical coning motion is, $\boldsymbol{\omega}_{\text{con}} = \omega_0\left[-2\sin^2\frac{\alpha}{2} \ -\sin\alpha\sin\omega_0 t \ \sin\alpha\cos\omega_0 t \right]^T$, and the error-less quaternion for coning motion is (Wu, 2018; Wu et al., 2018; Wu and Yan, 2019), $\boldsymbol{q}_{\text{con}} = \left[\cos\frac{\alpha}{2} \ 0 \ \sin\frac{\alpha}{2}\cos\omega_0 t \ \sin\frac{\alpha}{2}\sin\omega_0 t \right]^T$. Here α is the coning angle, and $\omega_0 = 2\pi f_0$ is the angular velocity of the coning motion, while f_0 is the coning frequency in Hz. When the true quaternion is $\boldsymbol{q}$, and $\hat{\boldsymbol{q}}$ is the calculated quaternion, the following equations will be used for the calculation of the error of the rotation angle. Here, we use the last three

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

Table 1. *Simulation parameters.*

	Parameter	Value
1	Coning angle	$1 \cdot 50$ degrees
2	Angular velocity of coning motion	10, 30 Hz
3	Attitude updating frequency	100 Hz
4	Sampling frequency	1000 Hz
5	Degree of angular velocities' fitted polynomial	5, 6, 8
6	Number of iterations	5, 15, 25, 35

elements of the quaternion because the first element of the error quaternion $\boldsymbol{q}_e$ is close to one, and the cosine function is not sensitive at zero, then

$$\varepsilon = 2\sin^{-1}(\boldsymbol{q}_e(2:4)), \boldsymbol{q}_e = [\boldsymbol{q}^* \otimes \hat{\boldsymbol{q}}] \tag{39}$$

where $\boldsymbol{q}_e(2:4)$ is vector part of the error's quaternion. When we use the DCM, the attitude error can be calculated as follows:

$$\varepsilon = \cos^{-1}[0.5(\text{trace}(\boldsymbol{C}_e) - 1)], \boldsymbol{C}_e = \boldsymbol{C}\hat{\boldsymbol{C}}^T \tag{40}$$

Again, due to the non-sensitivity of cosine function in zero, Equation (40) is not effective. Instead, we can approximate the error of rotation angle from $\boldsymbol{C}_e$ as:

$$\boldsymbol{S} = \boldsymbol{C}_e - \boldsymbol{I}, \quad \boldsymbol{S} \cong [\boldsymbol{s}\times], \quad \varepsilon = \boldsymbol{s} \tag{41}$$

where, when the rotation angle is closer to zero, the matrix $\boldsymbol{S}$ is almost a skew-symmetric matrix. Now, we will compare the performances of the four methods as follows:

- QuatPow: Quaternion approximation using power series,
- QuatTay: Quaternion approximation using Taylor series,
- DCMPow: DCM approximation using power series, and
- DCMTay: DCM approximation using Taylor series.

We used single and double precision implementations for implementing the above four methods. For any method in real applications, the processor has limited bits for implementation of any variable type. For example, any single precision number is between $1.1755(10^{-38})$ and $3.4028(10^{+38})$; in case of double precision, it is between $2.2251(10^{-308})$ and $1.7977(10^{308})$; therefore, the results for single and double precisions will be different (Muller et al., 2018).

All the simulations are done with ideal gyros and the sampling frequency of rate gyros is 1000 Hz and we suppose that the attitude updating frequency is 100 Hz (two-speed algorithm). Actually, for updating quaternion, there are 11 data then the maximum degree of the polynomial for angular velocity is 10. There are two parameters for coning motion, we are considering $f_0 = 30$ Hz, and $\alpha = 50$ degrees for the first simulation. Simulation parameters are represented in Table 1.

When f_0 is high, we need very high sampling frequencies for attitude reconstruction. The true angular velocity is presented in Figure 1 when $t \in [0, 0.05]$ sec. We know that the amplitude of angular velocity is a function of coning motion's frequency.

Here N and N_{it} stand for degree of angular velocities' fitted polynomial and number terms of series (number of iterations), respectively. Figure 2 shows the error between the fitted and real angular velocities when N is 6 and 8. It is obvious that when N is high, the error of angular velocity is low. In this simulation, when N is 8, the order of error is $3.3(10^{-7})$rad/sec and when N is 6, the error of fitted angular velocity is $2.4(10^{-4})$rad/sec. Here, we set N at 8.

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

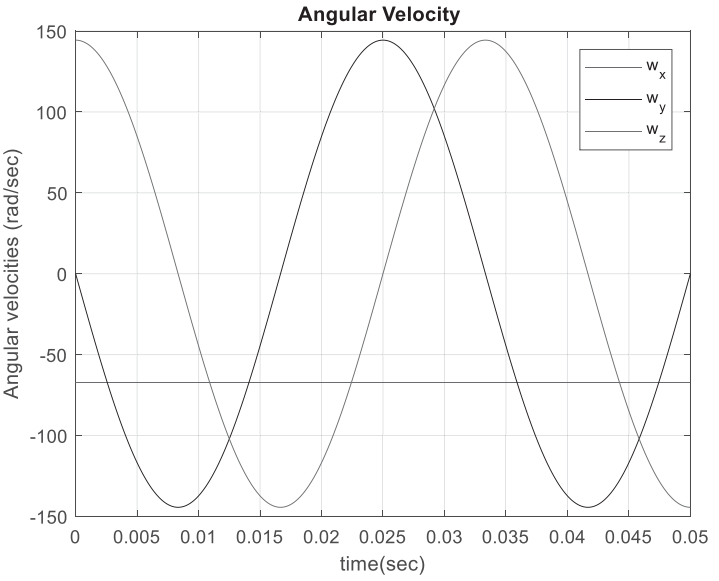


Figure 1. *Angular velocity in* $t \in [0, 0.05]$*sec.*

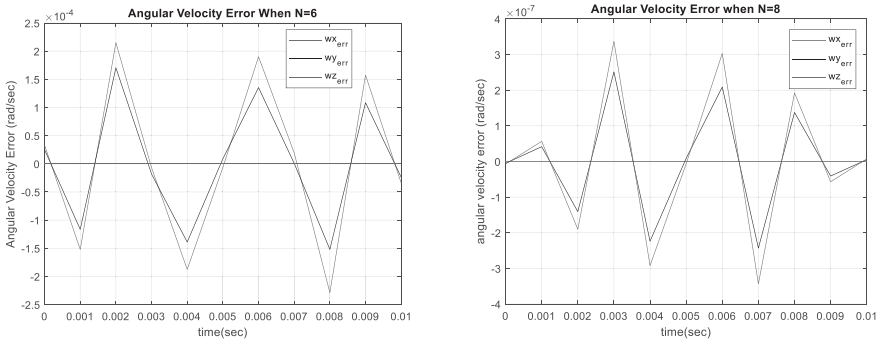


Figure 2. *Error of fitted angular velocity with degrees of* $N = 6$ *and* $N = 8$*.*

Figure 3 shows the single and double precision algorithm implementation when the fitted polynomial degree is 8, and iteration numbers are 5, 15, 25 and 35. Here the blue colour shows the double precision implementation and red colour stands for single precision implementation. Obviously, when the number of iterations increases, the errors in both implementations will reduce. The double precision implementation always shows better accuracy than the single precision for all methods. Furthermore, when the number of iterations is low, the single and double precision implementations have the same performance, but when the numbers of iterations are high, two implementation methods have different performance.

Figure 4 compares the effects of degrees of the fitted polynomial in the error of rotation. The red colour stands for $N = 8$, and the blue colour is for $N = 5$. It is obvious that when the error of angular velocity is low (N is 8), the error of all the methods in double or single forms is lower than the case in which $N = 5$. It was obvious because using high degrees of fitted polynomial has fewer errors for angular velocity and then the calculated attitude has low error.

Figure 5 shows that in the double implementation form, for all methods, when the number of iterations approaches infinity, finally, all the algorithms have the same errors, but in limited iterations, the quaternion's performance is always better than the DCM. In high iterations, both quaternion and

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

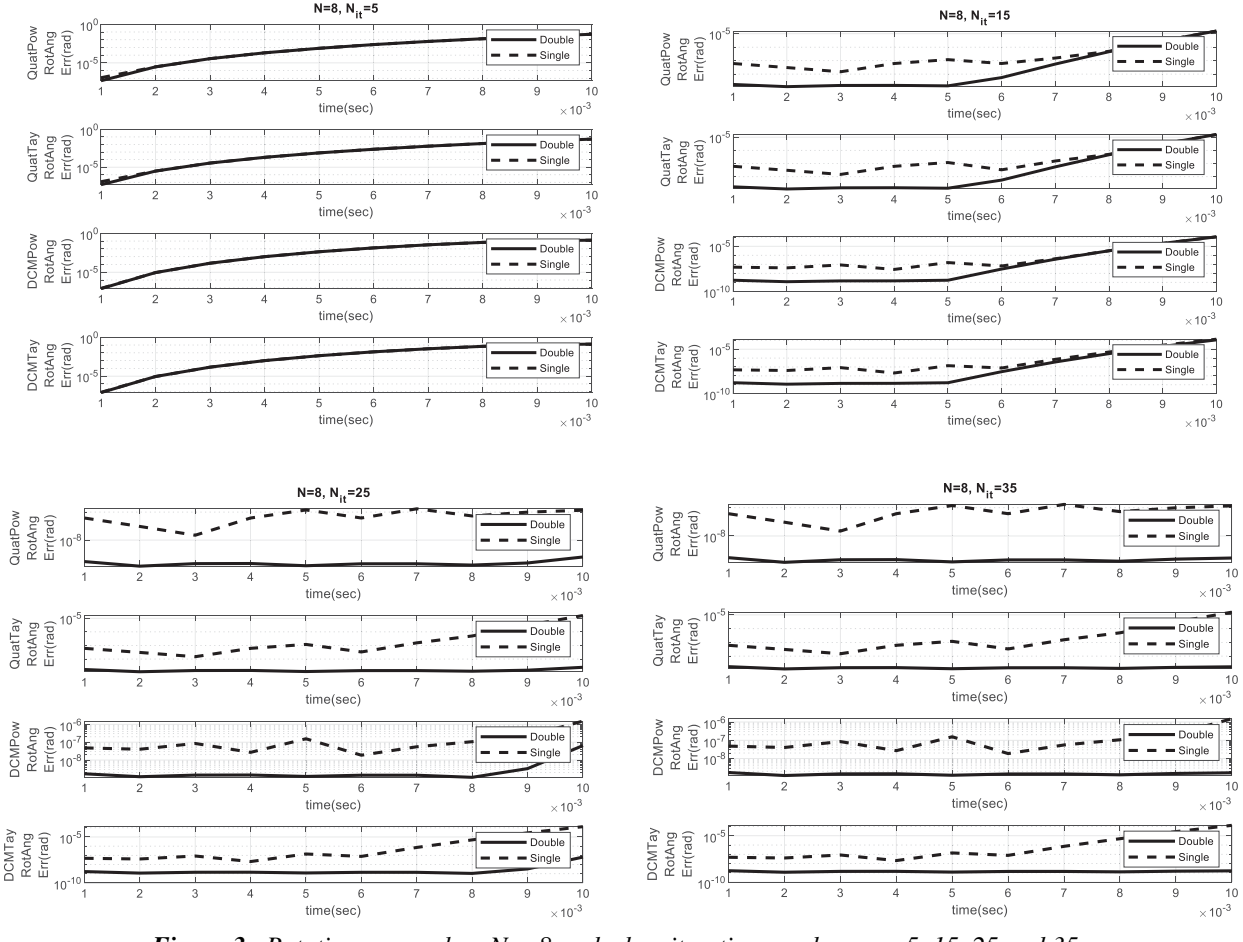


Figure 3. *Rotation error when N = 8 and when iteration numbers are 5, 15, 25 and 35..*

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

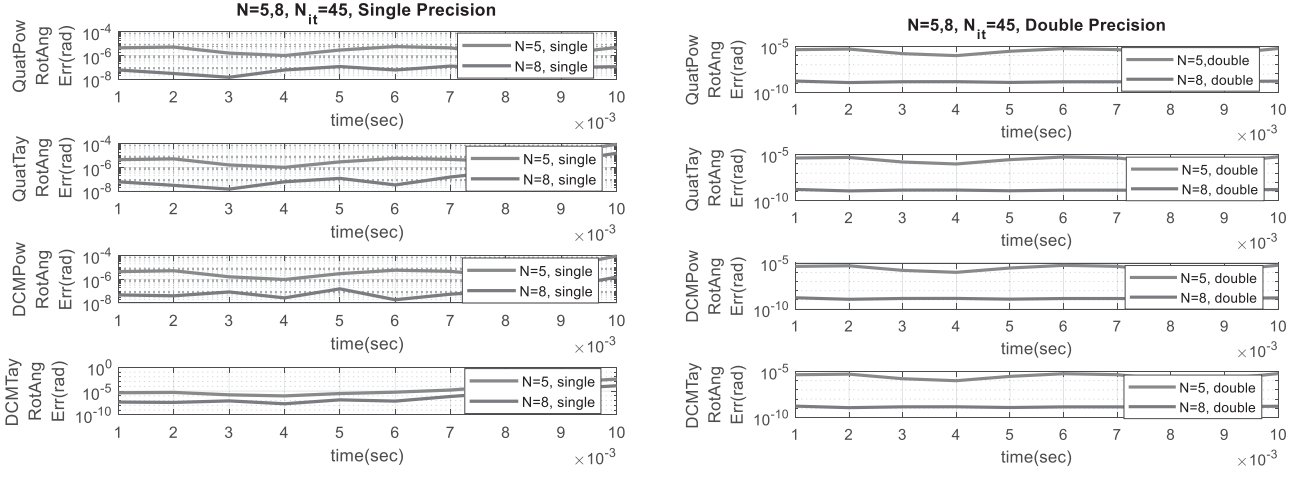


Figure 4. *Rotation error when N is 5, 8 and,* $N_{it} = 45$ *in single and double precision implementation forms.*

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

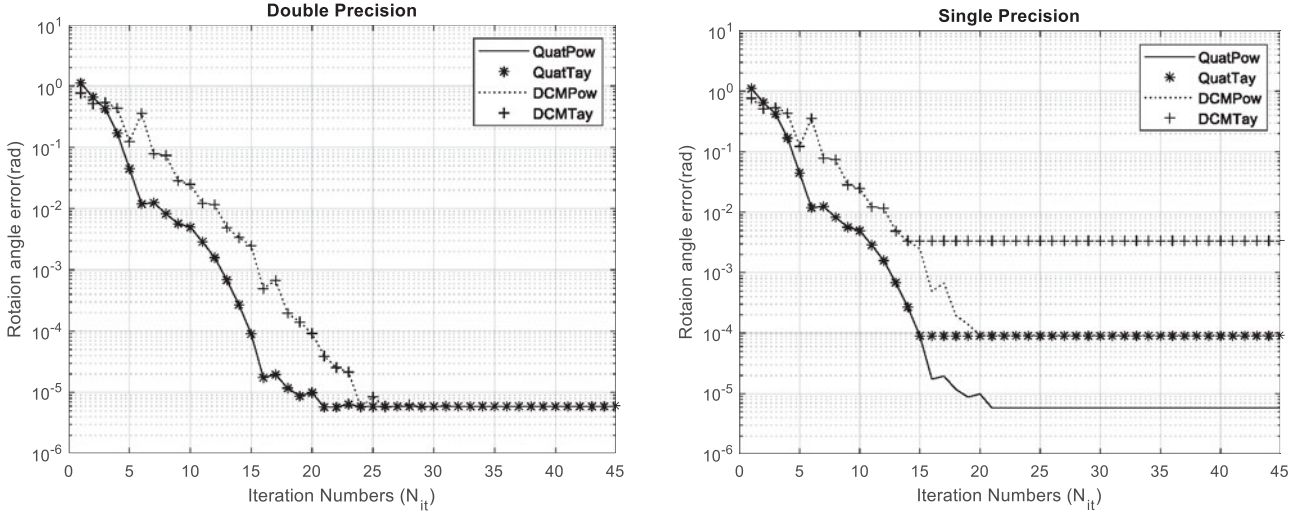


Figure 5. *Maximum error of double precision implementation at time $t = 0.01$ s when $N = 8$.*

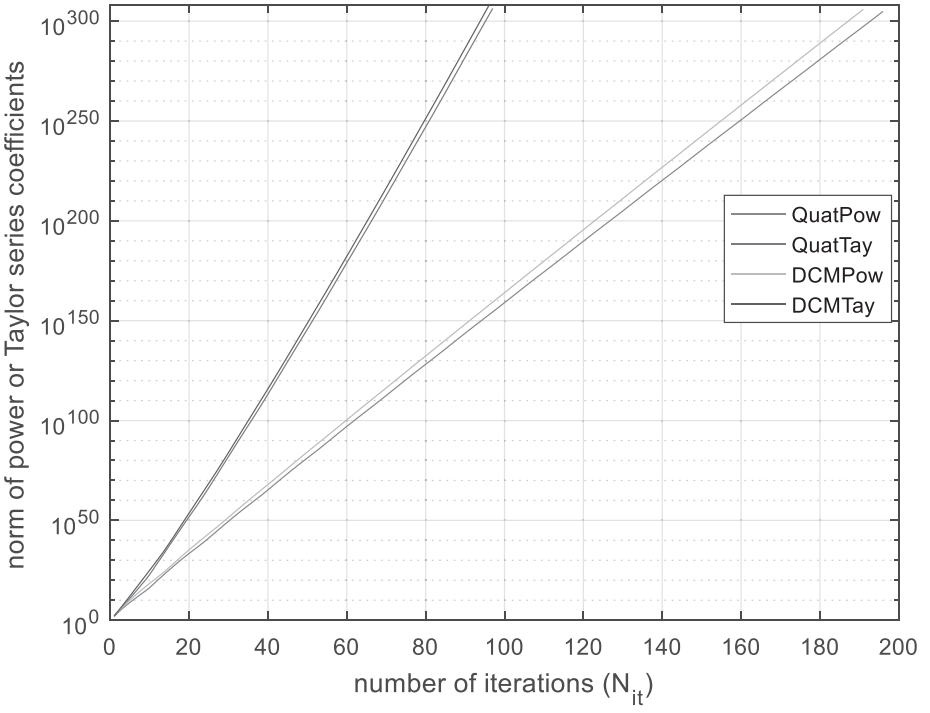


Figure 6. *The rotation angle error when N = 8.*

DCM methods, the Taylor and power series results are close to each other. In the second figure, we can see that the single precision implementations, the QuatPow has the best performance when the number of iterations increases. The power series always shows better results than the Taylor series when the number of iterations is large enough.

Figure 6 shows that the norm of the derivatives of the Taylor polynomial and the norm of power series' coefficients increase when the number of iterations increases. It is clear that, when the iteration number approaches infinity, in the ideal case, there is no error in the calculated attitude. It is also observable that the power series always has a low norm and then we can consider more terms for decreasing the error. In addition, the norm of coefficients of DCM is lower than the quaternion coefficients' norm. In the quaternion form, we can take more terms into the model, so we will get better performance. The one cycle of computational burden of algorithms are 1, 1·02, 1·09, 1·10 for QuatPow, QuatTay, DCMPow, DCMTay in relation to the computaion burden for QuatPow method.

Figure 7 represents the error of rotation angle with two fitted angular velocities' polynomials using different algorithms in single and double precision forms for a long time (100 s). All the subplots show that the double precision implementations have low errors, but in power series, in single precision implementation form, the error of rotation angle has the lowest value, which is closer to double implementation method and the worst case is using Taylor series of DCM.

Figure 8 shows the results for single and double precision implementations when the coning parameters are $f_0 = 10$ Hz and $\alpha = 1$ degree, and $N_{it} = 50$. Here, the number of iterations is high enough for implementation and increasing the N_{it} will have the same results. Figure 8 represents the situation in which the frequency and magnitude of the angular velocity for coning motion are low, the performances of algorithms are closer to each other, and double precision implementation always gives better results than the single precision. Figure 9 shows the 1,000 time Monte-Carlo simulations for single and double implementaions for algorithms when the standard deviation (std) of white noise of gyros are 0·1 deg/hour. All algorithms are using $N = 8$ and $N_{it} = 50$, $f_0 = 10$ Hz, $\alpha = 1$ deg.

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

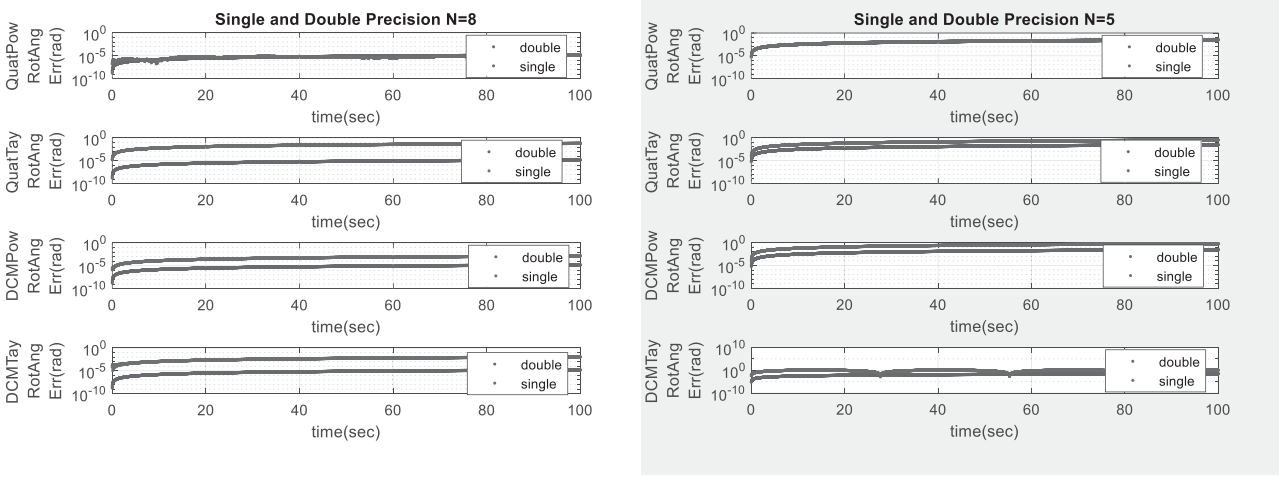


Figure 7. *The rotation angle error when N = 8 and N = 5.*

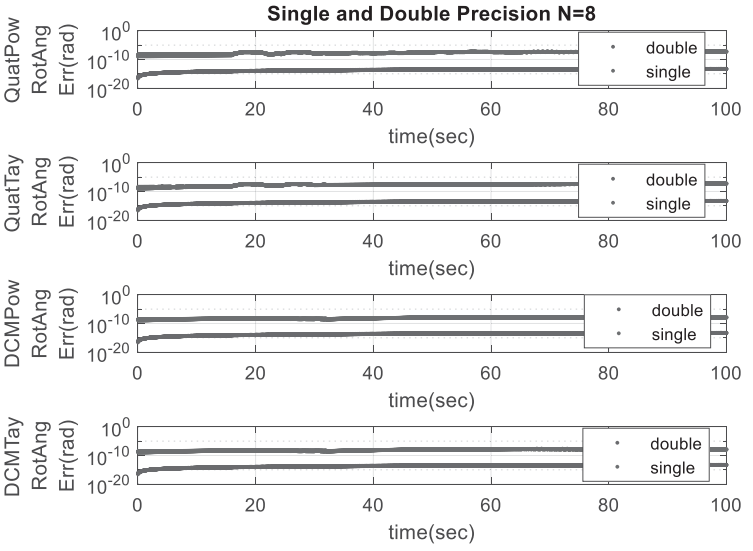


Figure 8. *The rotation angle error when* $N = 8$, $f_0 = 10\,Hz, \alpha = 1\,deg, N_{it} = 50$.

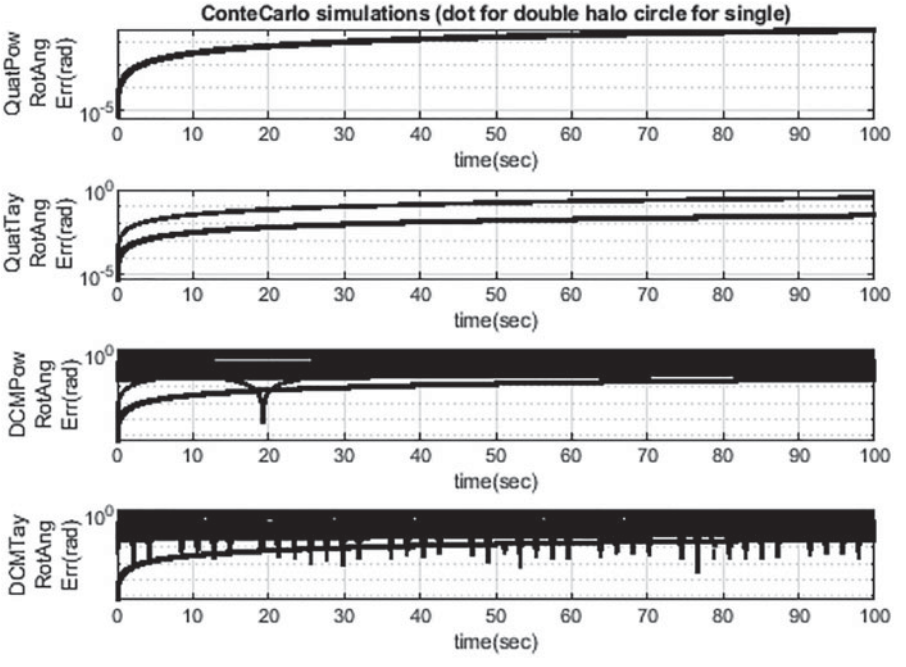


Figure 9. *Monte-Carlo simulations when* $N = 8$ *and* $N_{it} = 50$.

Table 2 shows that in all methods the mean and std of error for double precision implementation are lower than the single precision implementation. The mean and std of error for DCM are both bigger than the quaternion method and thus we can say the DCM method is more sensitive to sensors' noises. In addition, mean and std of error in QuatPow are lower than all methods for both single and double precision implementations. Then we can say that the QuatPow method is more robust than the others with respect to the sensors' noises.

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

Table 2. *Monte-Carlo simulation results after 100 s.*

	Method	Double or single	Mean of error (rad)	std of error (rad)
1	QuatPow	double	0·0299784	0·0000002
		single	0·0337641	0·0000065
2	QuatTay	double	0·0299784	0·0000002
		single	0·3357337	0·0000067
3	DCMPow	double	0·7644496	0·4370067
		single	0·8051695	0·4530733
4	DCMTay	double	0·7644496	0·4370067
		single	0·7884743	0·4515462

6. Conclusion

In this research, four methods of attitude reconstruction from gyro data have been implemented. The methods were based on power series and Taylor series, and they were implemented using quaternion or DCM for representing attitude. All the methods have been implemented in single and double precision forms. Results show that when we use higher degrees for fitting polynomials for angular velocity, the errors of angular velocity will reduce and later, the errors of attitude will reduce as well. When the angular velocities are polynomial, the attitude differential equation has the time series based analytical solutions. Here, we considered two power series with infinite sentences. The first one was a power series while the second one was a Taylor polynomial. For assuring higher accuracy, we need to consider more terms in Taylor and power series polynomials. Unfortunately, the coefficients of series have large values in large iteration numbers. When we use the Taylor polynomial, we need to calculate the derivatives of the quaternion that have very large norms; consequently, the double and single precision implementations do not give the same results. When we compare the same terms of the Taylor series and the power series, the norms of power series coefficients are lower than the derivatives of the quaternion. Because of this issue, we can use more terms in the power series since, in that case, the accuracy of the algorithm with power series will be better than the Taylor series. When we use the quaternion method instead of DCM for representation of attitude, we should sum up several matrices (in DCM methods) or vectors (in quaternion methods) that have errors due to limited number of bits. It has been shown that the computational burden of QuatPow is lower than the all method and the proposed method has good properties regarding noise of sensors. Hence the QuatPow method is clearly the best method from all points of view.

APPENDIX A

The system matrix of state Equation (8) has the following form:

$$\mathbf{M}_{i-1} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{I} \\ \frac{1}{i}\boldsymbol{\Omega}_{\boldsymbol{n}} & \frac{1}{i}\boldsymbol{\Omega}_{\boldsymbol{n-1}} & \frac{1}{i}\boldsymbol{\Omega}_{\boldsymbol{n-2}} & \dots & \frac{1}{i}\boldsymbol{\Omega}_{\boldsymbol{1}} & \frac{1}{i}\boldsymbol{\Omega}_{\boldsymbol{0}} \end{bmatrix} \tag{A1}$$

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

When i approaches infinity, the last line of that matrix will approach zero matrix.

$$\mathbf{M}_{\infty} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A2}$$

The state transition matrix from n to ∞ is represented by $\boldsymbol{\Phi}_{\infty,n}$ and defined as

$$\chi_{\infty} = \boldsymbol{\Phi}_{\infty,n}\chi_n, \boldsymbol{\Phi}_{\infty,n} = \mathbf{M}_{\infty} \dots \mathbf{M}_{\infty} \dots \mathbf{M}_{n+1}\mathbf{M}_n \tag{A3}$$

The multiplication of $\mathbf{M}_{\infty}\mathbf{M}_{\infty}$ is

$$\mathbf{M}_{\infty}\mathbf{M}_{\infty} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A4}$$

Then after n times multiplication of $\mathbf{M}_{\infty}$ the state transition matrix $\boldsymbol{\Phi}_{\infty,n}$ will approach zero, so the norm of $\boldsymbol{\Phi}_{\infty,n}$ will be zero.

References

Bortz, J. E. (1971). A new mathematical formulation for strapdown inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems*, **7**(1), 61–66.

Garg, S. C., Morrow, L. D. and Mamen, R. (1978). Strapdown navigation technology: A literature survey. *Journal of Guidance and Control*, **1**(3), 161–172.

Gilmore, J. P. (1980). Modular strapdown guidance unit with embedded microprocessors. *Journal of Guidance and Control*, **3**(1), 3–10.

Hairer, E., Nørsett, S. P. and Wanner, G. (1993). Solving ordinary differential equations I. Nonstiff problems, Volume 8 of Springer Series in Computational Mathematics.

Ignagni, M. B. (1990). Optimal strapdown attitude integration algorithms. *Journal of Guidance, Control, and Dynamics*, **13**(2), 363–369.

Ignagni, M. B. (1996). Efficient class of optimized coning compensation algorithms. *Journal of Guidance, Control, and Dynamics*, **19**(2), 424–429.

Ignagni, M. B. (1998). Duality of optimal strapdown sculling and coning compensation algorithms. *Navigation*, **45**(2), 85–95.

Jekeli, C. (2012). *Inertial Navigation Systems with Geodetic Applications*. Walter de Gruyter, Berlin, New York.

Jordan, J. W. (1969). *An Accurate Strapdown Direction Cosine Algorithm*. National Aeronautics and Space Administration, Cambridge, MA, United States.

Kunovský, J. (2015). Modern Taylor Series Method. 2015 IEEE 13th International Scientific Conference on Informatics. IEEE, 1–8.

Lee, J. G., Yoon, Y. J., Mark, J. G. and Tazartes, D. A. (1990). Extension of strapdown attitude algorithm for high-frequency base motion. *Journal of Guidance, Control, and Dynamics*, **13**(4), 738–743.

McKern, R. A. (1968). A study of transformation algorithms for use in a digital computer. Doctoral dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Miller, R. B. (1983). A new strapdown attitude algorithm. *Journal of Guidance, Control, and Dynamics*, **6**(4), 287–291.

Minor, J. (1965). August. Low-Cost Strapdown Inertial Systems. *Guidance Control Conference*, 117–123.

Muller, J.-M., Brisebarre, N., De Dinechin, F., Jeannerod, C.-P., Lefevre, V., Melquiond, G., Revol, N., Stehlé, D. and Torres, S. (2018). Handbook of Floating-Point Arithmetic. Vol. 1. Boston, MA: Birkhäuser.

Musoff, H. and Murphy, J. H. (1995). Study of strapdown navigation attitude algorithms. *Journal of Guidance, Control, and Dynamics*, **18**(2), 287–290.

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press

Roscoe, K. M. (2001). Equivalency between strapdown inertial navigation coning and sculling integrals/algorithms. *Journal of Guidance, Control, and Dynamics*, **24**(2), 201–205.

Savage, P. (1966). A New Second-Order Solution for Strapped-Down Attitude Computation. Guidance and Control Conference, 1805.

Savage, P. (1998a). Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms. *Journal of Guidance, Control, and Dynamics*, **21**(1), 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**(2), 208–221.

Savage, P. G. (2010). Coning algorithm design by explicit frequency shaping. *Journal of Guidance, Control, and Dynamics*, **33**(4), 1123–1132.

Savage, P. (2017). *Down-summing Rotation Vectors for Strapdown Attitude Updating (SAI WBN-14019)*. Maple Plain, MN: Strapdown Associates, Inc. http://strapdownassociates.com/Rotation%20Vector%20Down_Summing.pdf.

Song, M., Wu, W. and Pan, X. (2013). Approach to recovering maneuver accuracy in classical coning algorithms. *Journal of Guidance, Control, and Dynamics*, **36**(6), 1872–1881.

Wilcox, J. C. (1967). A new algorithm for strapped-down inertial navigation. *IEEE Transactions on Aerospace and Electronic Systems*, **3**(5), 796–802.

Wu, Y. (2018). RodFIter: Attitude reconstruction from inertial measurement by functional iteration. *IEEE Transactions on Aerospace and Electronic Systems*, **54**(5), 2131–2142.

Wu, Y. and Litmanovich, Y. A. (2019). Strapdown attitude computation: functional iterative integration versus Taylor series expansion. *arXiv preprint arXiv:1909.09935*.

Wu, Y. and Yan, G. (2019). Attitude reconstruction from inertial measurements: QuatFIter and its comparison with RodFIter. *IEEE Transactions on Aerospace and Electronic Systems*, **55**(6), 3629–3639.

Wu, Y., Cai, Q. and Truong, T. K. (2018). Fast RodFIter for attitude reconstruction from inertial measurements. *IEEE Transactions on Aerospace and Electronic Systems*, **55**(1), 419–428.

Yan, G., Weng, J., Yang, X. K. and Qin, Y. Y. (2017). An accurate numerical solution for strapdown attitude algorithm based on Picard iteration. *Journal of Astronautics*, **38**(12), 1308–1313.

Cite this article: Ghanbarpourasl H (2021). Attitude reconstruction from strap-down rate gyros using power series. *The Journal of Navigation* **74**: 4, 763–781. https://doi.org/10.1017/S0373463321000023

https://doi.org/10.1017/S0373463321000023 Published online by Cambridge University Press