JOURNAL OF AIRCRAFT
Vol. 38, No. 4, July–August 2001

Review of Attitude Representations Used for Aircraft Kinematics

W. F. Phillips* and C. E. Hailey†
Utah State University, Logan, Utah 84322-4130
and
G. A. Gebert‡
Sverdrup Technology, Inc., Eglin Air Force Base, Florida 32542

A detailed survey is presented of the literature on attitude representation dating from the early work of Euler and Hamilton to recent publications in fields such as navigation and control. The scope is limited to the development of the aircraft kinematic transformation equations in terms of four different attitude representations, including the well-known Euler angles, the Euler-axis rotation parameters, the direction cosines, and the Euler–Rodrigues quaternion. The emphasis is directed at the application of the quaternion formulation to aircraft kinematics. Results are presented that reinforce observations that the quaternion formulation, typically implemented to eliminate singularities associated with the Euler angle formulation, is far superior to the other commonly used formulations based on computational efficiency alone. A development of quaternion constraints necessary to independently constrain roll, pitch, yaw, bank angle, elevation angle, and/or azimuth angle is presented. For verification of simulation codes, a general closed-form solution to the quaternion formulation, for the case of constant rotation, is also presented. Additionally, a discussion is provided of numerical integration methods and numerical errors for the quaternion formulation. This discussion is especially important for simulations that may still utilize a common error reduction scheme originally developed for analog computers.

Nomenclature

$\boldsymbol{A}$	=	arbitrary quaternion
$\boldsymbol{B}$	=	arbitrary quaternion
$[\boldsymbol{C}]$	=	direction-cosine matrix
$\boldsymbol{E}$	=	Euler axis vector
E_x	=	x component of Euler axis vector
E_y	=	y component of Euler axis vector
E_z	=	z component of Euler axis vector
$\boldsymbol{e}$	=	Euler–Rodrigues quaternion
$\boldsymbol{e}_r$	=	renormalized Euler–Rodrigues quaternion
e_x	=	x component of Euler–Rodrigues quaternion
e_y	=	y component of Euler–Rodrigues quaternion
e_z	=	z component of Euler–Rodrigues quaternion
e_0	=	scalar component of Euler–Rodrigues quaternion
$\dot{\boldsymbol{e}}$	=	time derivative of Euler–Rodrigues quaternion
$\boldsymbol{e}^*$	=	conjugate of Euler–Rodrigues quaternion
g	=	acceleration of gravity
$[\boldsymbol{I}]$	=	identity matrix
$\boldsymbol{i}_x$	=	unit vector in the x direction, Earth-fixed coordinates
$\boldsymbol{i}_{x_b}$	=	unit vector in the x_b direction, body-fixed coordinates
$\boldsymbol{i}_y$	=	unit vector in the y direction, Earth-fixed coordinates
$\boldsymbol{i}_{y_b}$	=	unit vector in the y_b direction, body-fixed coordinates
$\boldsymbol{i}_z$	=	unit vector in the z direction, Earth-fixed coordinates
$\boldsymbol{i}_{z_b}$	=	unit vector in the z_b direction, body-fixed coordinates
k	=	gain coefficient
$[\boldsymbol{M}]$	=	matrix in the quaternion rate equation
p	=	rolling rate, body-fixed coordinates
$\boldsymbol{Q}$	=	arbitrary quaternion
Q_x	=	x component of arbitrary quaternion
Q_y	=	y component of arbitrary quaternion
Q_z	=	z component of arbitrary quaternion
Q_0	=	scalar component of arbitrary quaternion
$\boldsymbol{Q}^*$	=	conjugate of arbitrary quaternion
q	=	pitching rate, body-fixed coordinates
r	=	yawing rate, body-fixed coordinates
$\boldsymbol{T}$	=	temporary quaternion
T_x	=	x component of temporary quaternion
T_y	=	y component of temporary quaternion
T_z	=	z component of temporary quaternion
T_0	=	scalar component of temporary quaternion
t	=	time
u	=	axial or x_b component of airspeed, body-fixed coordinates
$\boldsymbol{V}$	=	airspeed vector
$\boldsymbol{V}_w$	=	wind vector
V_{wx}	=	x component of wind, Earth-fixed coordinates
V_{wy}	=	y component of wind, Earth-fixed coordinates
V_{wz}	=	z component of wind, Earth-fixed coordinates
v	=	sideslip or y_b component of airspeed, body-fixed coordinates
$\boldsymbol{v}$	=	arbitrary vector
v_x	=	x component of arbitrary vector, inertial coordinates
v_{x_b}	=	x component of arbitrary vector, body-fixed coordinates
v_y	=	y component of arbitrary vector, inertial coordinates
v_{y_b}	=	y component of arbitrary vector, body-fixed coordinates
v_z	=	z component of arbitrary vector, inertial coordinates
v_{z_b}	=	z component of arbitrary vector, body-fixed coordinates
w	=	normal or z_b component of airspeed, body-fixed coordinates
x	=	x coordinate, inertial coordinates
x_b	=	x coordinate, body-fixed coordinates
y	=	y coordinate, inertial coordinates
y_b	=	y coordinate, body-fixed coordinates
z	=	z coordinate, inertial coordinates
z_b	=	z coordinate, body-fixed coordinates
δt	=	time step
ε	=	orthogonality error
Θ	=	total rotation angle
θ	=	Euler elevation angle or pitch attitude
ϕ	=	Euler bank angle or roll attitude
φ	=	general Euler angle
ψ	=	Euler azimuth angle or heading
$\boldsymbol{\omega}$	=	angular velocity vector

Presented as Paper 2000-4302 at the AIAA 39th Aerospace Sciences Meeting and Exhibit, Reno, NV, 10–13 January 2000; received 7 September 2000; revision received 21 January 2001; accepted for publication 24 January 2001. Copyright © 2001 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Professor, Mechanical and Aerospace Engineering Department. Member AIAA.

†Associate Professor, Mechanical and Aerospace Engineering Department. Associate Fellow AIAA.

‡Technical Fellow, Aerodynamics. Senior Member AIAA.

Introduction

THE numerical simulation of motion with six degrees of freedom (six DOF) is important in many areas of modern technology, ranging from computer animated film making to the development of aircraft and spacecraft flight simulators. For these varied applications, there is a need to describe both position and orientation in either an inertial coordinate system or a noninertial coordinate system. With specific application to aircraft flight, one common method for describing attitude is through the use of Euler angles. The Euler angle formulation contains a singularity known as gimbal lock. As a result, alternative methods for relating noninertial coordinates to inertial coordinates have been developed. One such formulation employs quaternion algebra to remove the singularity.

Several mechanics textbooks introduce quaternions as part of the treatment of generalized rigid-body motion.[1,2] Quaternions are applied to a diverse set of rigid-body motion problems ranging from simulations of nanotechnology,[3] underwater tethers,[4] vehicle navigation,[5,6] robotics,[7–11] molecular dynamics,[12] and electromagnetic fields[13] to computer visualization and animation.[14–16] In the early 1990s, Chou[17] published a review article on quaternion kinematics for the robotics community. Also published in the early 1990s, Shuster[18] provides a review of vector/matrix algebra and attitude representations used by the spacecraft community including Euler angles, direction cosines, axis-azimuth representations, Euler–Rodrigues symmetric parameters, Rodrigues or Gibbs parameters, and Cayley–Klein parameters.

What is lacking is a current, thorough review of attitude kinematics directed primarily toward the aircraft simulation and modeling community. To support this point of view, a survey of recent and classic introductory atmospheric flight mechanic textbooks shows no[19–31] or very limited[32,33] discussion of quaternions, despite the wide use of the quaternion formulation in simulating aircraft motion.[34,35] The need for a review of attitude kinematics for the aircraft community is underscored by the use of quaternion integration schemes employed on today's digital computers, which were developed originally for analog simulations in the 1950s and early 1960s. An early application of quaternions to analog computer simulation of aircraft and missile motion was developed by Robinson[36,37] and popularized by Mitchell and Rogers.[38] Because these analog computer simulations were inherently first order, the raw quaternion formulation did not work well. When such simulations were run, the magnitude of the quaternion would grow quite rapidly with time. Because the formulation requires the magnitude of the quaternion to remain unity, these large errors would render the simulations almost useless. To remedy this problem, techniques were developed to reduce this computer error. One error reduction scheme, due to Corbett and Wright,[39] worked quite well when used with the analog computers of that era. Its use is no longer necessary with the higher-order integration techniques available on today's digital computers. However, the Corbett–Wright correction is still presented as necessary in the development of quaternion kinematics in recently published aeronautics textbooks.[33,40]

In simulating six-DOF aircraft motion, Newton's second law is most conveniently written in terms of a noninertial coordinate system fixed to the moving aircraft. Position and orientation are most conveniently described in terms of an Earth-fixed coordinate system, which for all practical purposes can be considered to be an inertial coordinate system. Here the inertial or Earth-fixed coordinate system will be designated x, y, and z and the noninertial or body-fixed coordinate system will be designated x_b, y_b, and z_b. The orientations of these two coordinate systems relative to the Earth and the aircraft are shown in Fig. 1. The position of the aircraft is specified by the location of the origin of the noninertial frame with respect to the inertial frame, and the orientation of the aircraft is specified in terms of the orientation of the noninertial frame relative to the inertial frame. To develop a complete six-DOF formulation, some means of transforming arbitrary vectors between noninertial coordinates and inertial coordinates is required.

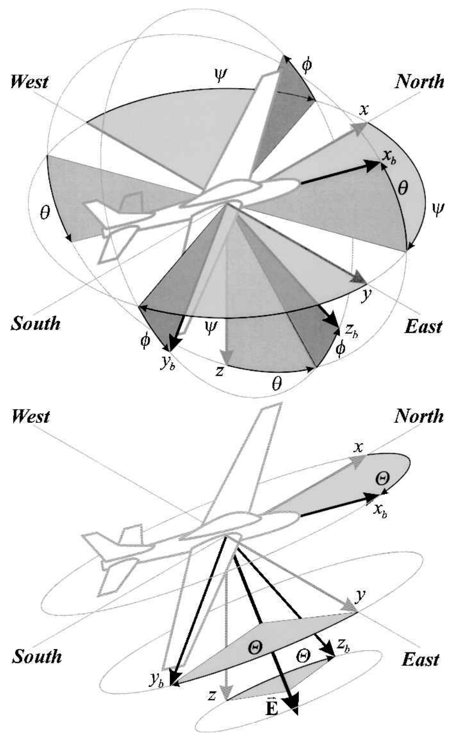

Fig. 1 Comparison between the Euler angle rotations and the Euler axis rotation.

Euler Angle Formulation

The orientation of the noninertial reference frame relative to the inertial reference frame can be described in terms of three consecutive rotations through three body-referenced Euler angles. There are 12 possible ways to define 3 independent body-referenced Euler angles. When started from the inertial reference frame, three consecutive rotations are performed, each about one of the three body-referenced axes, to arrive at the final noninertial reference frame. The only restriction required to provide three DOF is that no two consecutive rotations can be about the same axis. With this restriction, there are six symmetric sets of Euler angles,

$$\varphi_x \to \varphi_y \to \varphi_x, \qquad \varphi_x \to \varphi_z \to \varphi_x, \qquad \varphi_y \to \varphi_x \to \varphi_y$$

$$\varphi_y \to \varphi_z \to \varphi_y, \qquad \varphi_z \to \varphi_x \to \varphi_z, \qquad \varphi_z \to \varphi_y \to \varphi_z$$

and six asymmetric sets,

$$\varphi_x \to \varphi_y \to \varphi_z, \qquad \varphi_x \to \varphi_z \to \varphi_y, \qquad \varphi_y \to \varphi_x \to \varphi_z$$

$$\varphi_y \to \varphi_z \to \varphi_x, \qquad \varphi_z \to \varphi_x \to \varphi_y, \qquad \varphi_z \to \varphi_y \to \varphi_x$$

Any of these 12 Euler angle sets can be used for attitude representation. For example, orbital mechanics and the quantum theory of angular motion use primarily the $\varphi_z \to \varphi_x \to \varphi_z$ formulation.[41–43] The rotation matrix for each of the 12 sets of Euler angles are presented by Tandon.[44] Graphical methods to illustrate the Euler angle transformations are also available.[45–47]

The aeronautics community commonly uses the last of the rotation sequences, $\varphi_z \to \varphi_y \to \varphi_x$. These three Euler angles, commonly written ψ, θ, and ϕ, are the azimuth angle or heading, the elevation angle or pitch attitude, and the bank angle or roll attitude, respectively. For this particular set of Euler angles, the orientation of the body-fixed frame, x_b, y_b, and z_b, relative to the Earth-fixed frame, x, y, and z, is described by performing the three consecutive rotations in the specific order as follows. First, the Earth-fixed frame is rotated about the z axis through an angle ψ. Next, the revolved reference frame is rotated about the new y axis through an angle θ. Finally, the revolved reference frame is rotated about the new x axis through an angle ϕ, to arrive at the final body-fixed reference frame. This set of Euler angles is shown in Fig. 1. Readers should be cautioned that other rotation sequences have been associated with the aeronautics community, for example, $\varphi_z \to \varphi_x \to \varphi_y$ (Ref. 48).

With the usual Euler angle definitions, consider an arbitrary vector $\boldsymbol{v}$ having components v_x, v_y, and v_z in the inertial coordinate system and having components v_{x_b}, v_{y_b}, and v_{z_b} in the noninertial coordinate system. When the notation $S_\chi = \sin(\chi)$ and $C_\chi = \cos(\chi)$ is used, the transformation of this vector from inertial coordinates to noninertial coordinates is given by[29,44,49,50]

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix} \begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & C_\theta \end{bmatrix} \begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{1}$$

Because Newton's second law is to be written in terms of the noninertial coordinate system, the gravitational vector must be expressed in noninertial coordinates. Applying Eq. (1) to the gravitational vector produces

$$\begin{Bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b} \end{Bmatrix} = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ g \end{Bmatrix} = g \begin{Bmatrix} -S_\theta \\ S_\phi C_\theta \\ C_\phi C_\theta \end{Bmatrix} \tag{2}$$

The inverse of the transformation matrix in Eq. (1) is its transpose, making

$$\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} \tag{3}$$

and from Eq. (3), the ground speed is related to the airspeed by

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} + \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{4}$$

where the dot over a variable indicates the usual derivative with respect to time. The two vectors on the right-hand side of Eq. (4) are, respectively, the wind vector written in Earth-fixed coordinates,

$$\boldsymbol{V}_w = V_{wx}\boldsymbol{i}_x + V_{wy}\boldsymbol{i}_y + V_{wz}\boldsymbol{i}_y \tag{5}$$

and the airspeed vector, which is written in body-fixed coordinates,

$$\boldsymbol{V} = u\boldsymbol{i}_{x_b} + v\boldsymbol{i}_{y_b} + w\boldsymbol{i}_{z_b} \tag{6}$$

The relationship between the noninertial angular rates and the time rate-of-change of the Euler angles is (see Refs. 23 and 29)

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & S_\phi S_\theta / C_\theta & C_\phi S_\theta / C_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{7}$$

where the vector on the far right-hand side of Eq. (7) is the angular velocity vector written in body-fixed coordinates,

$$\boldsymbol{\omega} = p\boldsymbol{i}_{x_b} + q\boldsymbol{i}_{y_b} + r\boldsymbol{i}_{z_b} \tag{8}$$

Equations (4) and (7) together provide the kinematic transformation equations in terms of Euler angles, which allow us to update the position and orientation of the aircraft with time. The gimbal lock singularity is seen in the last two terms of the first and last rows of Eq. (7). When the Euler elevation angle θ is ± 90 deg, these four terms go to infinity, and the Euler angle integration becomes indeterminate. Combining Euler angle sequences can sometimes yield a singularity that is difficult to observe from a cursory inspection of the equations. Junkins and Shuster[51] propose a scheme involving spherical trigonometric relations to provide a clearer picture of the singularities. Alternate spacecraft-attitude dynamics models have been proposed that eliminate the Euler angle singularity (see Refs. 52 and 53), although the aircraft community has not adopted them.

Direction Cosine Formulation

The direction-cosine matrix can be formed from any of the symmetric or asymmetric Euler angle sets (see Ref. 54). For example, the matrix on the right side of Eq. (1) is a direction-cosine matrix. One way to avoid the singularity in Eq. (7) is to treat the nine components of this matrix as a fundamental description of orientation. The elements of this matrix are called the direction cosines. If these nine direction cosines are known, the components of an arbitrary vector in body-fixed coordinates are quite simply related to the components of the same vector in inertial coordinates through the definition of the direction-cosine matrix $[\boldsymbol{C}]$,

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} \equiv \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{9}$$

and the gravitational vector, expressed in body-fixed coordinates, is

$$\begin{Bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b} \end{Bmatrix} = g \begin{Bmatrix} C_{13} \\ C_{23} \\ C_{33} \end{Bmatrix} \tag{10}$$

The inverse of the direction-cosine matrix is simply its transpose, so that the ground speed is related to the airspeed by

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} + \begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{11}$$

When the body-fixed reference frame is undergoing rotation, the elements of the direction-cosine matrix are functions of time. To complete the kinematic formulation in terms of direction cosines, a set of differential equations relating the temporal derivatives of the nine elements of the direction-cosine matrix to the body-fixed angular rates is required. These nine equations, known as Poisson's kinematic equations, can be written in matrix form as (see Refs. 53 and 54)

$$\begin{bmatrix} \dot{C}_{11} & \dot{C}_{12} & \dot{C}_{13} \\ \dot{C}_{21} & \dot{C}_{22} & \dot{C}_{23} \\ \dot{C}_{31} & \dot{C}_{32} & \dot{C}_{33} \end{bmatrix} = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{12}$$

Note that there is no standard convention for numbering the direction cosines. Here, the numbering in Eq. (9) has been defined relative to the forward transformation, from Earth-fixed coordinates to body-fixed coordinates. This is the most commonly used notation.[18,54–56] However, other authors have numbered the direction cosines relative to the inverse transformation, from body-fixed coordinates to Earth-fixed coordinates.[33] One notation is simply the transpose of the other. Nevertheless, the reader should be careful to observe which notation is being used when reviewing a publication or simulation code.

There are only three DOF associated with orientation. However, there are nine components in the direction-cosine matrix. Thus, these nine components cannot be independent. There must be six levels of redundancy associated with this formulation. It can be shown that rigid-body rotation is a proper orthogonal transformation.[58] Therefore, the direction-cosine matrix is proper orthogonal and the inverse of the direction-cosine matrix must be its transpose,[57,58] $[\boldsymbol{C}]^{-1} = [\boldsymbol{C}]^T$. This requires the constraints

$$C_{11}^2 + C_{21}^2 + C_{31}^2 = 1, \qquad C_{12}^2 + C_{22}^2 + C_{32}^2 = 1$$

$$C_{13}^2 + C_{23}^2 + C_{33}^2 = 1, \qquad C_{11}C_{12} + C_{21}C_{22} + C_{31}C_{32} = 0$$

$$C_{11}C_{13} + C_{21}C_{23} + C_{31}C_{33} = 0$$

$$C_{12}C_{13} + C_{22}C_{23} + C_{32}C_{33} = 0 \tag{13}$$

These six constraints are usually called the redundancy relations. Mathematically, Eq. (12) preserves the orthogonality of the direction-cosine matrix. However, errors associated with integrating Eq. (12) numerically can cause degradation in the orthogonality of the matrix. Furthermore, these orthogonality errors can build up with time during the simulation. A method for avoiding this error buildup with the direction cosine formulation was first developed by Corbett and Wright[39] and is still commonly used today.

The direction cosine formulation contains no singularities and is frequently used by the aeronautics community[59] to avoid the possibility of gimbal lock. However, numerical integration of Eq. (12) is excessively time consuming, and a severe computational penalty is paid for its use.

Euler Axis Formulation

The orientation of the noninertial reference frame relative to the inertial reference frame can be described in terms of a single rotation through an angle Θ, about a particular axis $\boldsymbol{E}$, which is commonly called the Euler axis or the eigenaxis.[60] A comparison between this Euler axis rotation and the Euler angle rotations that are commonly used by the aircraft community is shown in Fig. 1.

The Euler axis rotation[60] gives rise to a four component description of orientation. The four components in this description are the total rotation angle Θ and the three components of a vector directed along the Euler axis, E_x, E_y, and E_z. Because these four parameters describe an orientation having only three DOF, there must be some redundancy in the Euler axis description as well. In fact, by describing orientation in this manner, a fourth mathematical DOF has been introduced. Clearly, the vector describing the orientation of the Euler axis could be of any arbitrary length. To remove this additional mathematical DOF, a constraint must be applied, fixing the magnitude of the Euler axis vector. Whereas this constraint is somewhat arbitrary, the usual and most obvious solution is to require the Euler axis vector to be of unit magnitude,

$$E_x^2 + E_y^2 + E_z^2 \equiv 1 \tag{14}$$

During the Euler axis rotation the orientation of the Euler axis is invariant. Thus, the vector $\boldsymbol{E}$ directed along the Euler axis has the same components in both inertial and noninertial coordinates,

$$\begin{Bmatrix} E_x \\ E_y \\ E_z \end{Bmatrix} = \begin{Bmatrix} E_{xb} \\ E_{yb} \\ E_{zb} \end{Bmatrix} \tag{15}$$

The components of an arbitrary vector $\boldsymbol{v}$ in body-fixed coordinates are related to the components of the same vector in Eath-fixed coordinates through what is commonly called Euler's formula[61]

$$\begin{Bmatrix} v_{xb} \\ v_{yb} \\ v_{zb} \end{Bmatrix} = \begin{bmatrix} E_{xx} + C_\Theta & E_{xy} + E_z S_\Theta & E_{xz} - E_y S_\Theta \\ E_{xy} - E_z S_\Theta & E_{yy} + C_\Theta & E_{yz} + E_x S_\Theta \\ E_{xz} + E_y S_\Theta & E_{yz} - E_x S_\Theta & E_{zz} + C_\Theta \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{16}$$

where $E_{ij} = E_i E_j (1 - C_\Theta)$. Since the early work of Euler, a number of papers have been written on the derivation of Eq. (16) (see Refs. 62–68).

From Eq. (16), the gravitational vector expressed in noninertial coordinates is

$$\begin{Bmatrix} g_{xb} \\ g_{yb} \\ g_{zb} \end{Bmatrix} = g \begin{Bmatrix} E_{xz} - E_y S_\Theta \\ E_{yz} + E_x S_\Theta \\ E_{zz} + C_\Theta \end{Bmatrix} \tag{17}$$

The inverse of the transformation matrix in Eq. (16) is obtained by simply rotating through the negative of the total rotation angle that is used in the forward transformation,[55]

$$\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} = \begin{bmatrix} E_{xx} + C_\Theta & E_{xy} - E_z S_\Theta & E_{xz} + E_y S_\Theta \\ E_{xy} + E_z S_\Theta & E_{yy} + C_\Theta & E_{yz} - E_x S_\Theta \\ E_{xz} - E_y S_\Theta & E_{yz} + E_x S_\Theta & E_{zz} + C_\Theta \end{bmatrix} \begin{Bmatrix} v_{xb} \\ v_{yb} \\ v_{zb} \end{Bmatrix} \tag{18}$$

Thus, the ground speed can be related to the airspeed through

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} + \begin{bmatrix} E_{xx} + C_\Theta & E_{xy} - E_z S_\Theta & E_{xz} + E_y S_\Theta \\ E_{xy} + E_z S_\Theta & E_{yy} + C_\Theta & E_{yz} - E_x S_\Theta \\ E_{xz} - E_y S_\Theta & E_{yz} + E_x S_\Theta & E_{zz} + C_\Theta \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{19}$$

The relationship between the noninertial angular rates and the time rate-of-change of the Euler axis rotation parameters is given by

$$\begin{Bmatrix} \dot{\Theta} \\ \dot{E}_x \\ \dot{E}_y \\ \dot{E}_z \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} 2E_x & 2E_y & 2E_z \\ E'_{xx} + C/S & E'_{xy} - E_z & E'_{xz} + E_y \\ E'_{xy} + E_z & E'_{yy} + C/S & E'_{yz} - E_x \\ E'_{xz} - E_y & E'_{yz} + E_x & E'_{zz} + C/S \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{20}$$

where $E'_{ij} = -E_i E_j C/S$, $S = \sin(\Theta/2)$, and $C = \cos(\Theta/2)$. Equations (19) and (20) are the kinematic transformation equations in terms of the Euler axis rotation parameters. Notice that Eq. (20) also has a singularity. When the total rotation angle Θ is zero, the integration of these equations is indeterminate.

Euler–Rodrigues Quaternion Formulation

The Euler[69]–Rodrigues[70] formulation is related to the Euler axis formulation through a simple change of variables. The four parameters in the Euler axis description of orientation are used to define four different parameters, which are somewhat more convenient. These four new parameters are defined as

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \equiv \begin{Bmatrix} \cos(\Theta/2) \\ E_x \sin(\Theta/2) \\ E_y \sin(\Theta/2) \\ E_z \sin(\Theta/2) \end{Bmatrix} \tag{21}$$

These four parameters are known as the Euler[69]–Rodrigues[70] symmetric parameters or the quaternion of finite rotation (see Ref. 71). The Euler–Rodrigues formulation for rigid-body kinematics was developed well before 1844, when the well-known mathematician, Hamilton, first developed the quaternion and the detailed theory of a noncommutative algebraic system known as quaternion algebra.[72–75] Much has been written about the life of Hamilton (see Refs. 75–77) and his moment of enlightenment concerning quaternion formulas while crossing a bridge with his wife. Hamilton, in a letter to his son, noted he could not resist the impulse to carve the fundamental quaternion formula into the stone on the bridge. However, it is clear that Hamilton did not develop quaternion algebra as means of describing rotational transformations.

The brief review of the history of the Euler–Rodrigues formulation that is presented here is taken primarily from Altmann,[78,79] Shuster,[18] and Cheng and Gupta.[80] In 1758, an early work of Euler[81] showed that any differential movement of a rigid body can be expressed as a translation and a rotation about some specific axis. Euler's theorem on the motion of a rigid body,[60] as well as Euler's formula[61] were both published in 1775. The first publication of the derivation of the Euler angles[82] was in 1862. Although the work was published posthumously, the date is probably in error because Euler died in 1783.

Whether Euler knew of the Euler–Rodrigues symmetric parameters is a subject of debate. In 1770, Euler[69] developed four symmetric parameters for orthogonal transformations (without the use of half angles). Roberson[83] and Jacobi[84] argue that Euler had presented a rotation matrix in terms of the so-called Euler–Rodrigues parameters. Shuster18 notes that the symmetric parameters developed by Euler in the 1770 paper contained sign errors and formed an improper orthogonal matrix. Note that Euler viewed a matrix as a table. The matrix as a mathematical object would not evolve until vector space was studied by Grassman,[85,86] Gibbs,[87] and Heaviside.[88] As a side note, the chief properties of vectors were worked out in Hamilton's investigation of quaternion algebra (see Ref. 89).

In 1840, four years before Hamilton began his algebraic study of quaternions, Rodrigues[70] published his work on the Euler–Rodrigues symmetric parameters, the rules for the compositions, and a geometrical construction for combining two rotations. Unlike Hamilton, whose accomplishments are well documented, the only published factual article on Rodrigues did not appear until 1980 (see Ref. 90). Earlier historians have invented a collaborator of Rodrigues by the name of Olinde (see Refs. 91 and 92) or have mistaken his name as Rodrigue (see Ref. 93) or Rodriques (see Ref. 94). Additional historical details can be found by Kline,[95] Van der Waerden,[96] Crowe,[97] and McDuffee.[98]

There is no universal agreement on the choice of indices for the Euler-Rodrigues symmetric parameters. Most authors use the indices 1–4 or 0–3. Some authors have chosen indices 1–3 for the vector components while using 4 as the scalar index,[18,99,100] whereas others have used 1 for the scalar index and 2–4 for the vector component.[101] The scalar index has also been chosen as 0 with indices 1–3 denoting the vector components.[15,17,38,102–107] There is not even universal agreement on the order of the vector components. Whereas most authors assign the x, y, and z components in ascending order, at least two authors have used 4 to denote the x component, 3 for the y component, and 2 for the z component.[34,37] To avoid confusion in the present paper, the vector components are explicitly labeled using the subscripts x, y, and z and a 0 subscript is used to denote the scalar component.

Because the four parameters defined by Eq. (21) uniquely describe an orientation having only three DOF, these four parameters must be related in some way. This relation is easily seen by squaring the four components of Eq. (21) and adding them together. This gives

$$e_0^2 + e_x^2 + e_y^2 + e_z^2 = \cos^2(\Theta/2) + \left(E_x^2 + E_y^2 + E_z^2\right)\sin^2(\Theta/2) \tag{22}$$

Because the Euler axis vector $\boldsymbol{E}$ is a unit vector and $\cos^2(\chi) + \sin^2(\chi) = 1$, it follows that

$$e_0^2 + e_x^2 + e_y^2 + e_z^2 = 1 \tag{23}$$

The transformation in Eq. (16) can be written in terms of half the total rotation angle by applying the trigonometric identities, $\sin(\chi) = 2\sin(\chi/2)\cos(\chi/2)$, $\cos(\chi) = \cos^2(\chi/2) - \sin^2(\chi/2)$, and $1 - \cos(\chi) = 2\sin^2(\chi/2)$. Thus, when the notation $S = \sin(\chi/2)$, $C = \cos(\chi/2)$, and $E_{ij} = 2E_iE_jS^2$ is used, Eq. (16) can be rewritten as

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} = \begin{bmatrix} E_{xx} + C^2 - S^2 & E_{xy} + 2E_zSC & E_{xz} - 2E_ySC \\ E_{xy} - 2E_zSC & E_{yy} + C^2 - S^2 & E_{yz} - 2E_xSC \\ E_{xz} + 2E_ySC & E_{yz} - 2E_xSC & E_{zz} + C^2 - S^2 \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{24}$$

Now, when Eq. (21) is applied and it is recognized that $E_{ij} = 2e_ie_j$ and $S^2 = (E_x^2 + E_y^2 + E_z^2)S^2 = e_x^2 + e_y^2 + e_z^2$, Eq. (24) becomes

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} = \begin{bmatrix} e_x^2 + e_0^2 - e_y^2 - e_z^2 & 2(e_xe_y + e_ze_0) & 2(e_xe_z + e_ye_0) \\ 2(e_xe_y - e_ze_0) & e_y^2 + e_0^2 - e_x^2 - e_z^2 & 2(e_ye_z + e_xe_0) \\ 2(e_xe_z - e_ye_0) & 2(e_ye_z - e_xe_0) & e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{bmatrix} \times \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{25}$$

The transformation equation given by Eq. (25) can be used to obtain the gravitational acceleration vector in body-fixed coordinates,

$$\begin{Bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b} \end{Bmatrix} = g \begin{Bmatrix} 2(e_xe_z - e_ye_0) \\ 2(e_ye_z + e_xe_0) \\ e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{Bmatrix} \tag{26}$$

The inverse of the transformation matrix in Eq. (25) is

$$\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} = \begin{bmatrix} e_x^2 + e_0^2 - e_y^2 - e_z^2 & 2(e_xe_y - e_ze_0) & 2(e_xe_z + e_ye_0) \\ 2(e_xe_y + e_ze_0) & e_y^2 + e_0^2 - e_x^2 - e_z^2 & 2(e_ye_z - e_xe_0) \\ 2(e_xe_z - e_ye_0) & 2(e_ye_z + e_xe_0) & e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{bmatrix} \times \begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} \tag{27}$$

Thus, the ground speed is related to the airspeed by

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} + \begin{bmatrix} e_x^2 + e_0^2 - e_y^2 - e_z^2 & 2(e_xe_y - e_ze_0) & 2(e_xe_z + e_ye_0) \\ 2(e_xe_y + e_ze_0) & e_y^2 + e_0^2 - e_x^2 - e_z^2 & 2(e_ye_z - e_xe_0) \\ 2(e_xe_z - e_ye_0) & 2(e_ye_z + e_xe_0) & e_z^2 + e_0^2 - e_x^2 - e_y^2 \end{bmatrix} \times \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{28}$$

The time rate-of-change of the Euler–Rodrigues symmetric parameters can be related to the time rate of change of the Euler axis rotation parameters. Differentiating Eq. (21) produces

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \begin{Bmatrix} -\sin(\Theta/2) \\ E_x\cos(\Theta/2) \\ E_y\cos(\Theta/2) \\ E_z\cos(\Theta/2) \end{Bmatrix} \frac{\dot{\Theta}}{2} + \begin{Bmatrix} 0 \\ \dot{E}_x\sin(\Theta/2) \\ \dot{E}_y\sin(\Theta/2) \\ \dot{E}_z\sin(\Theta/2) \end{Bmatrix} \tag{29}$$

When Eq. (20) is used to express the time rate of change of the Euler axis rotation parameters in terms of the noninertial angular rates, Eq. (29) can be written as

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} -E_xS & -E_yS & -E_zS \\ C & -E_zS & E_yS \\ E_zS & C & -E_xS \\ -E_yS & E_xS & C \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{30}$$

or after applying Eq. (21)

$$\begin{Bmatrix} \dot e_0 \\ \dot e_x \\ \dot e_y \\ \dot e_z \end{Bmatrix} = \frac{1}{2}\begin{bmatrix} -e_x & -e_y & -e_z \\ e_0 & -e_z & e_y \\ e_z & e_0 & -e_x \\ -e_y & e_x & e_0 \end{bmatrix}\begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{31}$$

Because Eq. (31) is linear in both the noninertial angular rates and the Euler–Rodrigues symmetric parameters, it can also be written as[108,109]

$$\begin{Bmatrix} \dot e_0 \\ \dot e_x \\ \dot e_y \\ \dot e_z \end{Bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \tag{32}$$

Equations (28) and (31) or (32) provide the kinematic transformation equations in terms of the Euler–Rodrigues symmetric parameters.

Robinson[37] detailed the advantages of the Euler–Rodrigues quaternion formulation over the Euler angle or the direction cosine formulation almost 50 years ago. Most of his observations, obtained on analog computers, are still valid in the digital world. When the Euler elevation angle θ is ± 90 deg, the Euler angle integration becomes indeterminate. Despite the singularity, the Euler angle formulation is widely used because the three Euler angles have the simple interpretation of heading, elevation angle, and bank angle.[110,111] The Euler–Rodrigues formulation is free of singularities; however, the physical interpretation of the quaternion is much less intuitive than that associated with the Euler angles. In a mathematical study, Stuelpnagel[101] considered parameterization of a general three-parameter rotation group, four-parameter rotation group, and five- and six-parameter groups. He proves that the three-parameter rotation group leads to nonlinear kinematic equations and that the Euler–Rodrigues symmetric parameters represent the smallest number of parameters (four) with linear kinematic equations. Errors associated with numerical integration of the kinematic equations for attitude have been characterized for both the quaternion and the direction cosine parameterizations, and the superiority of the quaternion parameterization is well documented.[104,111–116] Lovren and Piper[117] show that, for the isolated case of classical coning motion, integration errors are similar for the direction-cosine parameterization and the quaternion parameterization. Another advantage of the Euler–Rodrigues formulation is the application of Kalman filtering to quaternion estimation.[118–121] For the reasons stated earlier, the spacecraft community often utilizes the Euler–Rodrigues formulation for attitude control systems.[104,122–141]

Perhaps the greatest advantage of the quaternion formulation over either the Euler angle formulation or the direction-cosine formulation is increased computational speed. Numerical integration of the nine component direction-cosine formulation is excessively time consuming when compared with the four component Euler–Rodrigues formulation. The trigonometric functions in the Euler angle transformation matrix make simulations that use this nonlinear formulation much more computationally intensive than those using the linear equations of the quaternion formulation. Furthermore, the computational advantage of the quaternion formulation can be extended even further through use of Hamilton's quaternion algebra. While the Euler–Rodrigues formulation was originally developed before quaternion algebra was conceived, the most computationally efficient algorithms are in fact based on this mathematical rule set, first introduced by Hamilton.[72] To demonstrate this computational advantage, a brief introduction to quaternion algebra is presented here.

Quaternion Algebra

A general quaternion $\boldsymbol{Q}$ is defined as

$$\boldsymbol{Q} \equiv Q_0 + Q_x\boldsymbol{i}_x + Q_y\boldsymbol{i}_y + Q_y\boldsymbol{i}_z \tag{33}$$

where Q_0, Q_x, Q_y, and Q_z are scalars and $\boldsymbol{i}_x$, $\boldsymbol{i}_y$, and $\boldsymbol{i}_z$ are unit vectors in the Cartesian x, y, and z directions, respectively. A quaternion has both vector and scalar properties. Vectors and scalars can be thought of as special cases of the more general quaternion. A scalar is a quaternion with Q_x, Q_y, and Q_z equal to zero, and a vector is a quaternion where Q_0 equals zero.

For quaternion multiplication, a procedure called the quaternion product is defined. Because a vector is a special case of a quaternion, care should be taken to distinguish the quaternion product from either the dot product or the cross product. Here, the quaternion product will be indicated using the operator $\otimes$. The quaternion products of the usual Cartesian unit vectors are defined according to the following rule set:

$$\begin{aligned} &\boldsymbol{i}_x \otimes \boldsymbol{i}_x \equiv -1, && \boldsymbol{i}_x \otimes \boldsymbol{i}_y \equiv \boldsymbol{i}_z, && \boldsymbol{i}_x \otimes \boldsymbol{i}_z \equiv -\boldsymbol{i}_y \\ &\boldsymbol{i}_y \otimes \boldsymbol{i}_x \equiv -\boldsymbol{i}_z, && \boldsymbol{i}_y \otimes \boldsymbol{i}_y \equiv -1, && \boldsymbol{i}_y \otimes \boldsymbol{i}_z \equiv \boldsymbol{i}_x \\ &\boldsymbol{i}_z \otimes \boldsymbol{i}_x \equiv \boldsymbol{i}_y, && \boldsymbol{i}_z \otimes \boldsymbol{i}_y \equiv -\boldsymbol{i}_x, && \boldsymbol{i}_y \otimes \boldsymbol{i}_z \equiv -1 \end{aligned} \tag{34}$$

The quaternion product simply follows the distributive law. Thus, the quaternion product of one quaternion $\boldsymbol{A}$ with another quaternion $\boldsymbol{B}$ is[142]

$$\begin{aligned} \boldsymbol{A} \otimes \boldsymbol{B} &= (A_0 + A_x\boldsymbol{i}_x + A_y\boldsymbol{i}_y + A_z\boldsymbol{i}_z) \otimes (B_0 + B_x\boldsymbol{i}_x + B_y\boldsymbol{i}_y + B_z\boldsymbol{i}_z) \\ &= (A_0B_0 - A_xB_x - A_yB_y - A_zB_z) + (A_0B_x + A_xB_0 \\ &\quad + A_yB_z - A_zB_y)\boldsymbol{i}_x + (A_0B_y - A_xB_z + A_yB_0 + A_zB_x)\boldsymbol{i}_y \\ &\quad + (A_0B_z + A_xB_y - A_yB_x + A_zB_0)\boldsymbol{i}_z \end{aligned} \tag{35}$$

The quaternion products defined in Eq. (34) are nearly like cross products, following the right-hand rule. However, special treatment is given to the quaternion product of a unit vector with itself. Also note that, in general, $\boldsymbol{A} \otimes \boldsymbol{B} \neq \boldsymbol{B} \otimes \boldsymbol{A}$ and so the quaternion product is not commutative. Also notice that the quaternion product reduces to simple multiplication for the special case when either operand is a scalar. However, it does not reduce to either the dot product or the cross product when both operands are vectors. In general, the quaternion product of two simple vectors is a four-component quaternion, equal to the negative of the dot product added to the cross product,

$$\begin{aligned} \boldsymbol{A} \otimes \boldsymbol{B} &= -(A_xB_x + A_yB_y + A_zB_z) + (A_yB_z - A_zB_y)\boldsymbol{i}_x \\ &\quad + (A_zB_x - A_xB_z)\boldsymbol{i}_y + (A_xB_y - A_yB_x)\boldsymbol{i}_z = -\boldsymbol{A}\cdot\boldsymbol{B} + \boldsymbol{A}\times\boldsymbol{B} \end{aligned} \tag{36}$$

Quaternions not only have properties of scalars and vectors, but quaternion algebra also has similarities to complex algebra. The magnitude of a quaternion is defined similar to that of a complex number or a vector,

$$|\boldsymbol{Q}| \equiv \sqrt{Q_0^2 + Q_x^2 + Q_y^2 + Q_z^2} \tag{37}$$

Also, a conjugate of a quaternion is defined as

$$\boldsymbol{Q}^* \equiv Q_0 - Q_x\boldsymbol{i}_x - Q_y\boldsymbol{i}_y - Q_z\boldsymbol{i}_z \tag{38}$$

where the asterisk indicates a quaternion conjugate. Using Eq. (38) with Eq. (35) produces

$$\boldsymbol{Q} \otimes \boldsymbol{Q}^* = Q_0^2 + Q_x^2 + Q_y^2 + Q_z^2 = |\boldsymbol{Q}|^2 \tag{39}$$

Thus, similar to a complex variable, the quaternion product of a quaternion with its conjugate generates a scalar equal to the square of the magnitude of the quaternion.

The four Euler–Rodrigues symmetric parameters can be thought of as the four components of a particular unit quaternion $\boldsymbol{e}$, having components (e_0, e_x, e_y, e_z), which uniquely specifies the orientation of the noninertial coordinate system relative to the inertial coordinate system. Furthermore, the rotational transformation given by Eq. (25)

can also be expressed in terms of quaternion products. Let $\boldsymbol{v}$ be an arbitrary vector having quaternion components $(0, v_{x_b}, v_{y_b}, v_{z_b})$ in the noninertial coordinate system and having quaternion components $(0, v_x, v_y, v_z)$ in the inertial coordinate system. The components of $\boldsymbol{v}$ in noninertial coordinates are related to the components of $\boldsymbol{v}$ in inertial coordinates through the quaternion transformation[17]

$$\boldsymbol{v}_b = \boldsymbol{e}^* \otimes (\boldsymbol{v} \otimes \boldsymbol{e}) \tag{40}$$

When Eq. (40) is expanded using Eq. (35), the orthogonal transformation can be written as a two-step process using a temporary quaternion $\boldsymbol{T}$,

$$\boldsymbol{T} = \boldsymbol{v} \otimes \boldsymbol{e} = (-v_x e_x - v_y e_y - v_z e_z) + (v_x e_0 + v_y e_z - v_z e_y)\boldsymbol{i}_x$$
$$+ (-v_x e_z + v_y e_0 + v_z e_z)\boldsymbol{i}_y + (v_x e_y + v_y e_x + v_z e_0)\boldsymbol{i}_z \tag{41}$$

$$\boldsymbol{v}_b = \boldsymbol{e}^* \otimes \boldsymbol{T} = (e_0 T_x - e_x T_0 - e_y T_z + e_z T_y)\boldsymbol{i}_{x_b} + (e_0 T_y + e_x T_z$$
$$- e_y T_0 - e_z T_x)\boldsymbol{i}_{y_b} + (e_0 T_z - e_x T_y + e_y T_x - e_z T_0)\boldsymbol{i}_{z_b} \tag{42}$$

From Eq. (40) and the definitions given in Eqs. (21) and (38), note that the conjugate of an Euler–Rodrigues quaternion represents the inverse rotation.

The advantage of Eqs. (41) and (42) over Eq. (25) is a matter of reduced computation time. The rotational transformation as expressed in Eq. (25) requires 39 multiplications and 21 additions. On the other hand, computing the same transformation from Eqs. (41) and (42) requires only 24 multiplications and 17 additions. This translates to a significant computational saving. For typical modern computers, the transformation computed from Eq. (25) requires about 50% greater computation time than does that computed from Eqs. (41) and (42). This can be quite significant in a flight simulator, which requires a very large number of such transformations for the visual display.[143]

For aircraft that are not all-attitude vehicles, such as transports, helicopters, and roll-stabilized missiles, Euler angles are often used to compute attitude because the singularity is not encountered (see Ref. 144). However, the computational advantage of the quaternion transformation is even more impressive when compared with the Euler angle transformation given by Eq. (1). Even when the trigonometric functions in Eq. (1) are evaluated only once, the transformation computed from Eq. (1) requires about 11 times as long to evaluate as the quaternion transformation computed from Eqs. (41) and (42). Thus, even ignoring the singularity in the Euler angle formulation, the quaternion formulation is far superior to the Euler angle formulation, based on computational efficiency alone.

In the mid-1980s, algorithms were developed, which further reduced the computation time for quaternion multiplication on the hardware available at that time. For example, Dvornychenko,[145] based on the work of Winograd,[146,147] presented two algorithms that were claimed to reduce computation time. The first of these algorithms requires 11 multiplications and 19 additions. The second requires 10 multiplications and 26 additions. This compares to 16 multiplications and 12 additions for the conventional quaternion multiplication described in Eq. (35). The hardware available at that time required something on the order of five instruction cycles for multiplication and only one cycle for addition. Thus, trading five multiplications for seven additions would result in a significant reduction in computation time, for the hardware of that era. However, typical modern hardware requires only one instruction cycle for multiplication and one instruction cycle for addition. Thus, the algorithms that were developed in the 1980s to speed up quaternion transformations will actually slow down the transformations when used with today's hardware. As late as 1993, Schuter[18] has claimed that these algorithms will speed up the transformations. For simulations run on modern hardware, these old algorithms should be replaced with the more straightforward algorithm specified by Eq. (35).

The system of differential equations, which governs the change in the transformation quaternion with time, can also be written in terms of a quaternion product.[100,148] This differential system, specified by either Eq. (31) or Eq. (32), can be written as

$$\dot{\boldsymbol{e}} = \tfrac{1}{2}\boldsymbol{e} \otimes \boldsymbol{\omega} \tag{43}$$

where $\boldsymbol{\omega}$ is the angular velocity vector in body-fixed coordinates, as defined by Eq. (8). Writing the quaternion differential equation in this form provides no computational saving over the matrix form given by Eq. (32). From Eq. (35), it is observed that the quaternion product in Eq. (43) requires 16 multiplications and 12 additions. This is exactly equivalent to that required for the matrix multiplication in Eq. (32). However, the matrix multiplication in Eq. (31) requires only 12 multiplications and 8 additions. Thus, the temporal derivative of the transformation quaternion should always be computed from Eq. (31).

Aircraft simulations often deal with more than two coordinate systems. For example, a single simulation could use an inertial coordinate system fixed to the center of the Earth, an Earth-fixed coordinate system at the aircraft's local latitude and longitude, an atmosphere-fixed coordinate system, and an aircraft body-fixed coordinate system. In such simulations, it is convenient to be able to combine efficiently a succession of coordinate rotations into a single transformation. This is, of course, readily accomplished when using a direction-cosine transformation by simply multiplying the direction-cosine matrices for each successive rotation. For example, if $[\boldsymbol{C}]_{1-2}$ is the direction cosine matrix for the transformation from coordinate system 1 to coordinate system 2 and $[\boldsymbol{C}]_{2-3}$ is the direction cosine matrix for the transformation from coordinate system 2 to coordinate system 3, we can write

$$[\boldsymbol{C}]_{1-3} = [\boldsymbol{C}]_{2-3}[\boldsymbol{C}]_{1-2} \tag{44}$$

where $[\boldsymbol{C}]_{1-3}$ is the direction cosine matrix for the transformation from coordinate system 1 to coordinate system 3. Because matrix multiplication is not commutative, the right-to-left order of the multiplication is important.

In an analogous manner, a succession of coordinate rotations can be combined into a single Euler–Rodrigues quaternion by making use of the quaternion product defined in Eq. (35). If $\boldsymbol{e}_{1-2}$ is the quaternion for the transformation from coordinate system 1 to coordinate system 2 and $\boldsymbol{e}_{2-3}$ is the quaternion for the transformation from coordinate system 2 to coordinate system 3, then the Euler–Rodrigues quaternion for the transformation from coordinate system 1 to coordinate system 3 is given by

$$\boldsymbol{e}_{1-3} = \boldsymbol{e}_{1-2} \otimes \boldsymbol{e}_{2-3} \tag{45}$$

Here again the order of multiplication is important because the quaternion product is not commutative. Notice that this quaternion product requires a left-to-right order, which contrasts with the right-to-left order required for direction cosine matrix multiplication. The relation given by Eq. (45) is not directly obvious from examination of Eq. (40). Whittaker[1] demonstrates the validity of this result.

There is some confusion in the literature regarding the order of multiplication required when using the Euler–Rodrigues quaternion for a succession of coordinate rotations. For example, Dvornychenko[145] presents a right-to-left ordering of the of the quaternion product, which is opposite to that shown in Eq. (45). This is because Dvornychenko has also used an unconventional quaternion definition, which has the vector components defined as the negative of those defined in Eq. (21). The positive signs used in Eq. (21) correspond to the conventional right-hand Euler–Rodrigues quaternion, which is based on a right-hand rotation about the Euler axis. The negative signs used by Dvornychenko[146] result in a left-hand quaternion, which is not normally used. When the right-hand Euler–Rodrigues quaternion is used, successive coordinate rotations must be expressed using the left-to-right quaternion product that is shown in Eq. (45).

Relationships Between the Quaternion and other Attitude Descriptors

The physical interpretation of the Euler–Rodrigues quaternion is much less intuitive than that associated with the Euler angles. For this reason it is convenient to be able to relate the quaternion to the Euler angles. Such a relation can be obtained by combining Eqs. (1) and (25). These two equations require

$$\begin{bmatrix} e_x^2+e_0^2-e_y^2-e_z^2 & 2(e_xe_y+e_ze_0) & 2(e_xe_z-e_ye_0) \\ 2(e_xe_y-e_ze_0) & e_y^2+e_0^2-e_x^2-e_z^2 & 2(e_ye_z+e_xe_0) \\ 2(e_xe_z+e_ye_0) & 2(e_ye_z-e_xe_0) & e_z^2+e_0^2-e_x^2-e_y^2 \end{bmatrix}$$

$$= \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta C_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix} \tag{46}$$

This matrix equation provides nine scalar equations relating the four components of the Euler–Rodrigues quaternion $\boldsymbol{e}$ to the three Euler angles, ϕ, θ, and ψ. However, the nine components of the matrices on both sides of Eq. (46) each describe a rotation having only three DOF. Thus, there are six levels of redundancy in Eq. (46). Additionally, Eq. (23) must also be satisfied, which simply requires that $\boldsymbol{e}$ be a unit quaternion. The three DOF in Eq. (46) combined with the requirement in Eq. (23) provide exactly the four DOF needed to solve for the four components of the quaternion $\boldsymbol{e}$.

When the diagonal components of Eq. (46) are combined with the requirement for a unit quaternion expressed in Eq. (23), a four-by-four system of algebraic equations for the squares of the quaternion components is obtained,

$$\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \begin{Bmatrix} C_\theta C_\psi \\ S_\phi S_\theta S_\psi + C_\phi C_\psi \\ C_\phi C_\theta \\ 1 \end{Bmatrix} \tag{47}$$

This system is readily solved by direct elimination to give

$$\begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \frac{1}{4} \begin{Bmatrix} 1 + C_\theta C_\psi + S_\phi S_\theta S_\psi + C_\phi C_\theta + C_\phi C_\psi \\ 1 - C_\phi C_\theta - S_\phi S_\theta S_\psi - C_\phi C_\psi + C_\theta C_\psi \\ 1 - C_\theta C_\psi - C_\phi C_\theta + S_\phi S_\theta S_\psi + C_\phi C_\psi \\ 1 - C_\theta C_\psi + C_\phi C_\theta - S_\phi S_\theta S_\psi - C_\phi C_\psi \end{Bmatrix} \tag{48}$$

Applying the half angle identities, Eq. (48) is written as

$$\begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \begin{Bmatrix} (C_{\phi/2}C_{\theta/2}C_{\psi/2} + S_{\phi/2}S_{\theta/2}S_{\psi/2})^2 \\ (S_{\phi/2}C_{\theta/2}C_{\psi/2} - C_{\phi/2}S_{\theta/2}S_{\psi/2})^2 \\ (C_{\phi/2}S_{\theta/2}C_{\psi/2} + S_{\phi/2}C_{\theta/2}S_{\psi/2})^2 \\ (S_{\phi/2}S_{\theta/2}C_{\psi/2} + C_{\phi/2}C_{\theta/2}S_{\psi/2})^2 \end{Bmatrix} \tag{49}$$

Each of these component equations has two possible solutions, and so the signs are indeterminate at this point. The off-diagonal components of Eq. (46) may be extracted to produce

$$\begin{bmatrix} 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & -2 & 2 & 0 & 0 \\ 0 & -2 & 0 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 & 2 \\ -2 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \begin{Bmatrix} e_0e_x \\ e_0e_y \\ e_0e_z \\ e_xe_y \\ e_xe_z \\ e_ye_z \end{Bmatrix} = \begin{Bmatrix} C_\theta C_\psi \\ S_\phi S_\theta S_\psi - C_\phi S_\psi \\ -S_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi \\ S_\phi C_\theta \\ C_\phi S_\theta S_\psi - S_\phi C_\psi \end{Bmatrix} \tag{50}$$

This simple algebraic system is easily solved by adding and subtracting appropriate pairs of equations. Replacing the first and second equations with their sum and their difference and doing likewise with the other two pairs of equations results in

$$\begin{Bmatrix} e_0e_x \\ e_0e_y \\ e_0e_z \\ e_xe_y \\ e_xe_z \\ e_ye_z \end{Bmatrix} = \frac{1}{4} \begin{Bmatrix} S_\phi C_\theta - C_\phi S_\theta S_\psi + S_\phi C_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi + S_\theta \\ C_\theta S_\psi - S_\phi S_\theta C_\psi + C_\phi S_\psi \\ C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi - S_\theta \\ S_\phi C_\theta + C_\phi S_\theta S_\psi - S_\phi C_\psi \end{Bmatrix} \tag{51}$$

When Eq. (49) is used in Eq. (51) and the half-angle identities are applied, the off-diagonal components of Eq. (46) reduce to

$$\begin{Bmatrix} s_0s_x(S_\phi C_\theta - C_\phi S_\theta S_\psi + S_\phi C_\psi) \\ s_0s_y(C_\phi S_\theta C_\psi + S_\phi S_\psi + S_\theta) \\ s_0s_z(-C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi) \\ s_xs_y(C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi) \\ s_xs_z(-C_\phi S_\theta C_\psi - S_\phi S_\psi + S_\theta) \\ s_ys_z(-S_\phi C_\theta + C_\phi S_\theta S_\psi + S_\phi C_\psi) \end{Bmatrix}$$

$$= \begin{Bmatrix} S_\phi C_\theta - C_\phi S_\theta S_\psi + S_\phi C_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi + S_\theta \\ C_\theta S_\psi - S_\phi S_\theta C_\psi + C_\phi S_\psi \\ C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi - S_\theta \\ S_\phi C_\theta + C_\phi S_\theta S_\psi - S_\phi C_\psi \end{Bmatrix} \tag{52}$$

where s_0, s_x, s_y, and s_z are the unknown signs of the quaternion components from Eq. (49). Thus, the off-diagonal components of Eq. (46) provide only three additional pieces of information, $s_0s_x = 1$, $s_0s_y = 1$, and $s_0s_z = -1$. When these signs are used with Eq. (49), only two possible solutions to Eq. (46) exist, which are

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \pm \begin{Bmatrix} C_{\phi/2}C_{\theta/2}C_{\psi/2} + S_{\phi/2}S_{\theta/2}S_{\psi/2} \\ S_{\phi/2}C_{\theta/2}C_{\psi/2} - C_{\phi/2}S_{\theta/2}S_{\psi/2} \\ C_{\phi/2}S_{\theta/2}C_{\psi/2} + S_{\phi/2}C_{\theta/2}S_{\psi/2} \\ C_{\phi/2}C_{\theta/2}S_{\psi/2} - S_{\phi/2}S_{\theta/2}C_{\psi/2} \end{Bmatrix} \tag{53}$$

Both of these solutions are valid. Obviously, any orientation of one coordinate system relative to another can be described in terms of two different right-hand rotations. For example, a right-hand rotation of 90 deg about the positive x axis is equivalent to a right-hand rotation of 270 deg about the negative x axis. The two solutions expressed in Eq. (53) represent these two equivalent rotations. Usually the positive sign is selected.

The inverse of Eq. (53) is obtained from Eq. (46) as well. This is rather straightforward and yields[105]

$$\begin{Bmatrix} \phi \\ \theta \\ \psi \end{Bmatrix} = \begin{Bmatrix} \operatorname{atan2}\left[2(e_0e_x + e_ye_z),\ \left(e_0^2 + e_z^2 - e_x^2 - e_y^2\right)\right] \\ \operatorname{asin}[2(e_0e_y - e_xe_z)] \\ \operatorname{atan2}\left[2(e_0e_z + e_xe_y),\ \left(e_0^2 + e_x^2 - e_y^2 - e_z^2\right)\right] \end{Bmatrix} \tag{54}$$

The function atan 2 in Eq. (54) is a two-argument arctangent that returns a result in the proper quadrant, such as the atan 2 intrinsic provided in FORTRAN and C. The two-argument function is not needed for the elevation angle θ because this angle is defined only in the range from $-\pi/2$ to $\pi/2$.

The Euler–Rodrigues quaternion is also related to the total rotation angle and the Euler axis components through the definition of the Euler–Rodrigues symmetric parameters,

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \equiv \begin{Bmatrix} \cos(\Theta/2) \\ E_x \sin(\Theta/2) \\ E_y \sin(\Theta/2) \\ E_z \sin(\Theta/2) \end{Bmatrix} \tag{55}$$

The inverse of Eq. (55) is readily found to be

$$\begin{Bmatrix} \Theta \\ E_x \\ E_y \\ E_z \end{Bmatrix} = \begin{Bmatrix} 2\cos^{-1}(e_0) \\ e_x/\sin(\Theta/2) \\ e_y/\sin(\Theta/2) \\ e_z/\sin(\Theta/2) \end{Bmatrix} \tag{56}$$

Thus, from knowledge of the four components of the Euler–Rodrigues quaternion, the orientation of the noninertial reference frame with respect to the inertial reference frame can be expressed as a single rotation about a known axis through a known angle. For nonzero rotation angles, the three vector components of this unit quaternion compose a vector directed along the Euler axis. The scalar component of this quaternion is equal to the cosine of one-half the angle of rotation about this axis, in a direction defined by the right-hand rule. If the scalar component of this quaternion is ± 1, the orientation of the Euler axis is indeterminate. However, a scalar component of ± 1 would indicate a total rotation angle of zero, making knowledge of the Euler axis unnecessary. It is important to recognize that, like the Euler axis vector $\boldsymbol{E}$, the four components of the unit quaternion $\boldsymbol{e}$ are the same in both the inertial and the noninertial reference frames. Thus, a reference frame need not be specified when referring to the components of this particular quaternion.

During the 1970s the spacecraft community utilized both the direction cosines and the Euler–Rodrigues quaternion to describe attitude. For example, in space shuttle steering algorithms the command attitude was computed as a direction-cosine matrix, whereas the attitude error was more conveniently described as a quaternion.[102] The quaternion was used to describe the attitude error because of its simple physical interpretation. The error axis coincides with the vector part of the error quaternion, and the total error is readily determined from the scalar part. To compute the error quaternion, the command-attitude quaternion was first extracted from the direction-cosine matrix. Several other space shuttle flight algorithms also required the determination of an attitude quaternion from a direction-cosines matrix. Consequently, a number of papers have been written on extracting the quaternion from the direction cosines.[103,149–152] The most efficient method was first published in 1978 by Shepperd[104] and has been restated by Shuster and Natanson.[99] The aircraft and missile community is still using both the direction cosines[59,153] and the Euler–Rodrigues quaternion (see Refs. 35 and 154–156) as attitude descriptors for aircraft flight simulation.

The direction cosines can be expressed in terms of the four components of the Euler–Rodrigues quaternion by equating the left-hand side of Eqs. (9) to the left-hand side of Eqs. (25):

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} = \begin{bmatrix} e_x^2+e_0^2-e_y^2-e_z^2 & 2(e_xe_y+e_ze_0) & 2(e_xe_z-e_ye_0) \\ 2(e_xe_y-e_ze_0) & e_y^2+e_0^2-e_x^2-e_z^2 & 2(e_ye_z+e_xe_0) \\ 2(e_xe_z+e_ye_0) & 2(e_ye_z-e_xe_0) & e_z^2+e_0^2-e_x^2-e_y^2 \end{bmatrix} \tag{57}$$

Combining the diagonal components of Eq. (57) with Eq. (23) provides a four-by-four system of equations that is readily solved to relate the squares of the quaternion components to the diagonal components of the direction-cosine matrix,

$$\begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \frac{1}{4}\begin{Bmatrix} 1+C_{11}+C_{22}+C_{33} \\ 1+C_{11}-C_{22}-C_{33} \\ 1-C_{11}+C_{22}-C_{33} \\ 1-C_{11}-C_{22}+C_{33} \end{Bmatrix} \tag{58}$$

The off-diagonal components of Eq. (57) can be rearranged to give

$$\begin{Bmatrix} e_0e_x \\ e_0e_y \\ e_0e_z \\ e_xe_y \\ e_xe_z \\ e_ye_z \end{Bmatrix} = \frac{1}{4}\begin{Bmatrix} C_{23}-C_{32} \\ C_{31}-C_{13} \\ C_{12}-C_{21} \\ C_{12}+C_{21} \\ C_{31}+C_{13} \\ C_{23}+C_{32} \end{Bmatrix} \tag{59}$$

The components of the quaternion can be computed, without singularities, from the direction cosines by using Eqs. (58) and (59). This is done by first finding the quaternion component of greatest magnitude from Eq. (58),

$$e_{\max}^2 = \max\left(e_0^2, e_x^2, e_y^2, e_z^2\right) \tag{60}$$

Once the component of largest magnitude has been determined, the quaternion is computed from one of the following algorithms:

$$\text{if } \left(e_0^2 = e_{\max}^2\right), \qquad e_0 = \pm\sqrt{1+C_{11}+C_{22}+C_{33}}\Big/2$$
$$e_x = (C_{23}-C_{32})/4e_0, \qquad e_y = (C_{31}-C_{13})/4e_0$$
$$e_z = (C_{12}-C_{21})/4e_0 \tag{61}$$

$$\text{if } \left(e_x^2 = e_{\max}^2\right), \qquad e_x = \pm\sqrt{1+C_{11}-C_{22}-C_{33}}\Big/2$$
$$e_0 = (C_{23}-C_{32})/4e_x, \qquad e_y = (C_{12}+C_{21})/4e_x$$
$$e_z = (C_{31}+C_{13})/4e_x \tag{62}$$

$$\text{if } \left(e_y^2 = e_{\max}^2\right), \qquad e_y = \pm\sqrt{1-C_{11}+C_{22}-C_{33}}\Big/2$$
$$e_0 = (C_{31}-C_{13})/4e_y, \qquad e_x = (C_{12}+C_{21})/4e_y$$
$$e_z = (C_{23}+C_{32})/4e_y \tag{63}$$

$$\text{if } \left(e_z^2 = e_{\max}^2\right), \qquad e_z = \pm\sqrt{1-C_{11}-C_{22}+C_{33}}\Big/2$$
$$e_0 = (C_{12}-C_{21})/4e_z, \qquad e_x = (C_{31}+C_{13})/4e_z$$
$$e_y = (C_{23}+C_{32})/4e_z \tag{64}$$

Notice that, as was the case with Eq. (46), there are two possible quaternions that will satisfy Eq. (57). These are the same two equivalent quaternions that were discussed following Eq. (53).

Applying Rotational Constraints to the Quaternion Formulation

It is occasionally desirable to simulate aircraft motion with one or more of the rotational DOF constrained. This is normally accomplished using the Euler angle formulation. However, because the quaternion transformation is more than an order of magnitude faster than the Euler angle transformation, the ability to apply rotational constraints to the quaternion formulation may be of some interest.

When applying rotational constraints, the coordinate system in which the constraint will be applied must be specified. For example, a roll constraint is very different from a bank angle constraint. The roll can be constrained very simply, in either the Euler angle formulation or the quaternion formulation, by simply replacing the x_b component of the angular momentum equation with the roll constraint

$$p = 0 \tag{65}$$

However, note that constraining the roll does not constrain the bank angle. All three Euler angles can still take any possible value with the roll completely constrained.

To constrain the bank angle, using the quaternion formulation, Eq. (54) is employed. If the bank angle is to remain zero, Eq. (54) requires

$$e_0 e_x + e_y e_z = 0 \tag{66}$$

and

$$e_x \dot{e}_0 + e_0 \dot{e}_x + e_z \dot{e}_y + e_y \dot{e}_z = 0 \tag{67}$$

Applying Eq. (32) to Eq. (67) produces

$$e_x(-pe_x - qe_y - re_z) + e_0(pe_0 + re_y - qe_z)$$
$$+ e_z(qe_0 - re_x + pe_z) + e_y(re_0 + qe_x - pe_y) = 0 \tag{68}$$

This can be simplified to yield the quaternion bank angle constraint

$$\left(e_0^2 + e_z^2 - e_x^2 - e_y^2\right)p + 2(e_0 e_y - e_x e_z)r = 0 \tag{69}$$

Thus, to constrain the bank angle, the roll and the yaw must be coordinated according to Eq. (69).

If the roll is to be constrained as well as the bank angle, then Eq. (69) requires

$$2(e_0 e_y - e_x e_z)r = 0 \tag{70}$$

From Eq. (54), this is equivalent to

$$\sin(\theta)r = 0 \tag{71}$$

Here it is observed that, if both the roll and the bank angle are to be constrained, then either the yaw or the elevation angle must also be constrained. With any nonzero elevation angle, any amount of yaw will produce a bank angle. This is not a function of the quaternion formulation. It is a simple kinematic fact.

The relationship between the body-fixed angular rates and the Euler angles, which was demonstrated here by using the roll and bank angle, is true in general. None of the Euler angles can be constrained by constraining only one of the body-fixed angular rates. The bank angle can be changed using only pitch and yaw, the elevation angle can be changed using only roll and yaw, and the azimuth angle can be changed using only roll and pitch. Any motion having only one of the rotational DOF constrained, in body-fixed coordinates, allows for all three DOF in orientation.

Similarly, the other rotational DOF can be constrained using the pitch constraint

$$q = 0 \tag{72}$$

the quaternion elevation angle constraint

$$\left(e_0^2 + e_z^2 - e_x^2 - e_y^2\right)q - 2(e_0 e_x + e_y e_z)r = 0 \tag{73}$$

the yaw constraint

$$r = 0 \tag{74}$$

and the quaternion azimuth angle constraint

$$2(e_0 e_x - e_y e_z)q + \left(e_0^2 + e_y^2 - e_x^2 - e_z^2\right)r = 0 \tag{75}$$

Motion with only one rotational DOF, in body-fixed coordinates, can also be simulated using the quaternion formulation. Constraining any two rotational DOF in body-fixed coordinates will constrain two DOF in orientation. For pure rolling motion,

$$q = r = 0 \tag{76}$$

Using these constraints in Eq. (32) gives

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -p & 0 & 0 \\ p & 0 & 0 & 0 \\ 0 & 0 & 0 & p \\ 0 & 0 & -p & 0 \end{bmatrix}\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \frac{p}{2}\begin{Bmatrix} -e_x \\ e_0 \\ e_z \\ -e_y \end{Bmatrix} \tag{77}$$

If the elevation angle and the azimuth angle are both initially zero, from Eq. (53), the initial condition is

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix}_{t=0} = \begin{Bmatrix} \cos(\phi_0/2) \\ \sin(\phi_0/2) \\ 0 \\ 0 \end{Bmatrix} \tag{78}$$

When Eq. (78) is used with Eq. (77), note that both e_y and e_z remain zero during this motion. Thus, pure rolling motion can be simulated using the constraints

$$q = r = e_y = e_z = 0 \tag{79}$$

Similarly, pure pitching motion could be simulated by applying the quaternion constraints

$$p = r = e_x = e_z = 0 \tag{80}$$

and pure yawing motion results from the constraints

$$p = q = e_x = e_y = 0 \tag{81}$$

Closed-Form Quaternion Solution for Constant Rotation

It is possible to obtain a closed-form solution to the quaternion formulation for the case of constant rotation. Although such a condition would frequently occur in spacecraft applications, it almost never occurs in aircraft applications. The angular rates experienced by a moving aircraft depend on the aerodynamic moments acting on the craft and are almost always changing with time. In this case, closed-form solutions to the quaternion formulation are very difficult or impossible to obtain, for all but the most trivial conditions.[157] Nevertheless, closed-form analytic solutions to the quaternion formulation for the case of constant angular rates provide an excellent mechanism for verifying[158,159] the numerical algorithms used to integrate the quaternion formulation. For such verification, it is useful to have a general solution that allows for rotation about any or all of the body-fixed axes. The development of such a solution, from Eq. (32), is rather straightforward.

Consider a rigid body that is rotating at a constant angular velocity. The differential equations governing the change in the components of the quaternion $\boldsymbol{e}$ with time are given by Eq. (32). This system of differential equations can be written in matrix notation as

$$\dot{\boldsymbol{e}} - [\boldsymbol{M}]\boldsymbol{e} = 0 \tag{82}$$

where

$$[\boldsymbol{M}] = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \tag{83}$$

When the matrix $[\boldsymbol{M}]$ is constant, the solution to Eq. (82) is

$$\boldsymbol{e}(t) = [\exp([\boldsymbol{M}]t)]\boldsymbol{e}(0) \tag{84}$$

where the matrix exponential $[\exp([\boldsymbol{M}]t)]$ is computed from the matrix series definition

$$[\exp([\boldsymbol{M}]t)] \equiv [\boldsymbol{I}] + [\boldsymbol{M}]t + [\boldsymbol{M}][\boldsymbol{M}]t^2/2! + \cdots \tag{85}$$

Here $\boldsymbol{e}(0)$ is the initial value of the quaternion at time $t = 0$, and $[\boldsymbol{I}]$ is the identity matrix. If the matrix $[\boldsymbol{M}]$ were completely arbitrary, the matrix exponential would need to be evaluated by summing a large number of terms in the infinite series. However, in this particular case we can take advantage of a special property of the matrix that is defined in Eq. (83).

By direct multiplication it is readily shown from Eq. (83) that

$$[\boldsymbol{M}][\boldsymbol{M}] = -\tfrac{1}{4}\omega^2[\boldsymbol{I}] \tag{86}$$

where $\omega^2 = p^2 + q^2 + r^2$. When use is made of Eq. (86), the infinite series in Eq. (85) can be rearranged as

$$[\exp([\boldsymbol{M}]t)] \equiv [\boldsymbol{I}]\left(1 - \frac{(\omega t/2)^2}{2!} + \frac{(\omega t/2)^4}{4!} + \cdots\right)$$

$$+ \frac{2}{\omega}[\boldsymbol{M}]\left(\frac{\omega t}{2} - \frac{(\omega t/2)^3}{3!} + \frac{(\omega t/2)^5}{5!} + \cdots\right)$$

or

$$[\exp([\boldsymbol{M}]t)] \equiv [\boldsymbol{I}]\cos(\omega t/2) + (2/\omega)[\boldsymbol{M}]\sin(\omega t/2) \tag{87}$$

Using Eq. (87) in Eq. (84) yields the general closed-form solution for constant rotation

$$\boldsymbol{e}(t) = [[\boldsymbol{I}]\cos(\omega t/2) + (2/\omega)[\boldsymbol{M}]\sin(\omega t/2)]\boldsymbol{e}(0) \tag{88}$$

For the special initial condition where all three Euler angles are zero,

$$\begin{Bmatrix} e_0(0) \\ e_x(0) \\ e_y(0) \\ e_z(0) \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{89}$$

the solution given by Eq. (88) reduces to the rather obvious result

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \begin{Bmatrix} \cos(\omega t/2) \\ (p/\omega)\sin(\omega t/2) \\ (q/\omega)\sin(\omega t/2) \\ (r/\omega)\sin(\omega t/2) \end{Bmatrix} \tag{90}$$

Equations (88) and (90) provide an excellent mechanism for testing most numerical algorithms used to integrate the quaternion formulation.

Numerical Integration of the Quaternion Formulation

When integrating Eq. (32) numerically, the zeros on the diagonal can cause some problems. Integration of this system can result in very large numerical errors if a first-order method is used. For this reason, a higher-order method, such as fourth-order Runge–Kutta or fourth-order Adams–Bashforth–Moulton (see Refs. 160 and 161) should always be used to integrate the quaternion formulation. Because most modern numerical codes use such methods, this is not at all restrictive for simulations run on modern digital computers.

Historically, aircraft simulations using the quaternion formulation were first commonly run using analog computers. Because these analog simulations were inherently first order, the raw quaternion formulation did not work well. When such simulations were run, the magnitude of the quaternion would grow quite rapidly with time. Because the formulation requires the magnitude of the quaternion to remain unity to maintain orthogonality in the attitude transformation, these large orthogonality errors would render the simulations almost useless. To remedy this problem, techniques were developed to reduce this orthogonality error. One such method, developed by Robinson[36,37] and popularized by Mitchell and Rogers,[38] is based on the method first suggested by Corbett and Wright[39] for maintaining orthogonality in the direction-cosine matrix. With this method, the kinematic equations in Eq. (32) were modified to include error reduction terms on the diagonal,

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2}\begin{bmatrix} k\varepsilon & -p & -q & -r \\ p & k\varepsilon & r & -q \\ q & -r & k\varepsilon & p \\ r & q & -p & k\varepsilon \end{bmatrix}\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \tag{91}$$

where ε is the orthogonality error defined as

$$\varepsilon = 1 - \left(e_0^2 + e_x^2 + e_y^2 + e_z^2\right) \tag{92}$$

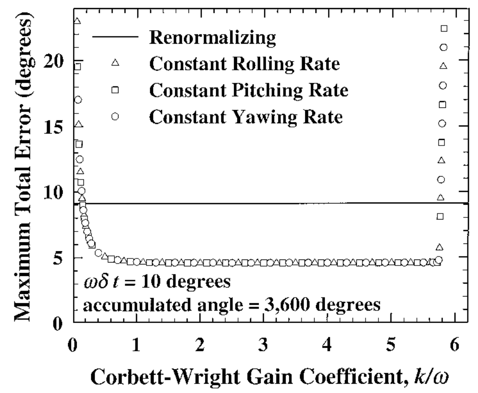

Fig. 2 Effect of changing the gain coefficient on the accuracy of a first-order Euler algorithm using Corbett–Wright orthogonality control.

and k is a gain coefficient, which Mitchell and Rogers[38] said should be "set to a very high value." This error reduction scheme, which shall be called Corbett–Wright orthogonality control,[39] worked quite well when used with the analog computers of that era.

Although analog computers are no longer widely available, it is possible to demonstrate approximately how Corbett–Wright[39] orthogonality control worked with an analog computer by integrating Eq. (91) using a first-order Euler integration scheme. Results of such an integration are shown in Fig. 2. For Fig. 2, the total error was computed as the difference between the numerical result and the exact solution from Eq. (88). Notice that, as the value of the Corbett–Wright[39] gain coefficient approaches zero, which is the case of no orthogonality control, the total error becomes very large. For gain coefficients larger than about 0.5ω, the error is reduced to a more or less acceptable level. With the exception of the increase in error on the right, the results shown in Fig. 2 are very similar to the results reported by Mitchell and Rogers[38] for a typical analog computer. Unlike the first-order digital integration, analog simulations were well behaved for very large values of k. The abrupt increase in error that is seen in Fig. 2 for k/ω greater than about 5.7 is a result of the well-known Euler instability that is associated with any first-order numerical integration. For Eq. (91), Fang and Zimmerman[162] have shown that this first-order instability occurs whenever the product of the gain coefficient and the time step is greater than unity,

$$k\delta t > 1$$

When digital computers first became fast enough to replace analog computers for running such simulations, (Corbett–Wright[39]), orthogonality control was often used in the digital solution algorithms. Furthermore, this orthogonality control is still widely used for aircraft flight simulation today.[33,40] It is, however, not necessary. When a modern digital computer is used to integrate Eq. (32) with any of the prevalent fourth-order numerical methods, the orthogonality error is extremely small. Using the Corbett–Wright[39] orthogonality control scheme with these modern numerical algorithms increases the computation time but does little to improve the accuracy of the simulation.

Even though the orthogonality error for modern numerical algorithms is very small, it can accumulate. Thus, if a simulation is to be run for long periods of time using large time steps, the quaternion should be renormalized periodically. The quaternion may be renormalized by dividing each component by the magnitude,[154,163,164]

$$\boldsymbol{e}_r = \boldsymbol{e}/|\boldsymbol{e}| = \boldsymbol{e}\Big/\sqrt{e_0^2 + e_x^2 + e_y^2 + e_z^2} \tag{93}$$

where the subscript r indicates the renormalized quaternion. As can be seen in Fig. 2, renormalization is not as accurate as Corbett–Wright[39] orthogonality control for first-order integration. However, a different result is obtained for higher-order integration. If the time steps used with fourth-order Runge–Kutta are of a size to produce about 10 deg of revolution per time step, it requires about 3×10^6 iterations to degrade the magnitude of the quaternion by 1%. Because a very long renormalization period can be used, periodic

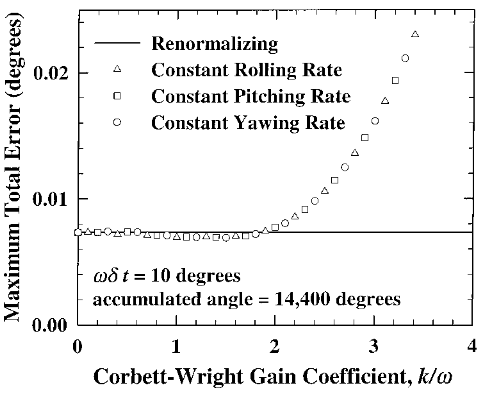

Fig. 3 Effect of changing the gain coefficient on the accuracy of a fourth-order Runge–Kutta algorithm using Corbett–Wright orthogonality control.

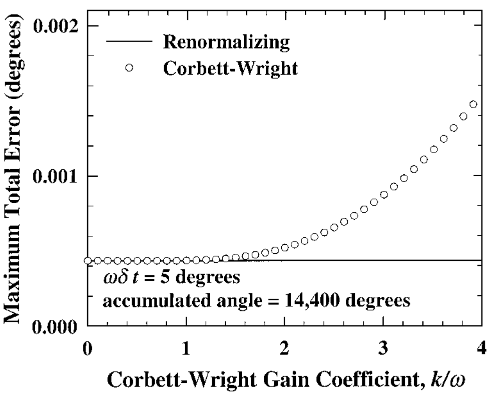

Fig. 5 Effect of reducing the step size on the accuracy of a fourth-order Runge–Kutta algorithm using Corbett–Wright orthogonality control.

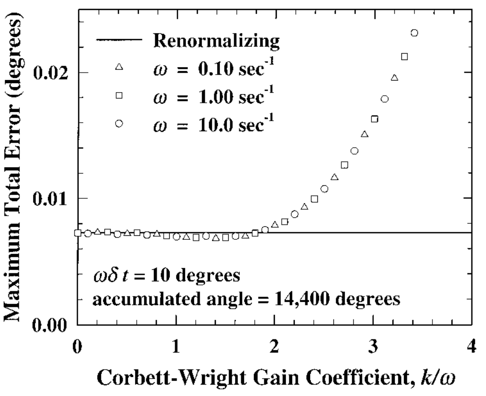

Fig. 4 Effect of changing the angular rate on the accuracy of a fourth-order Runge–Kutta algorithm using Corbett–Wright orthogonality control.

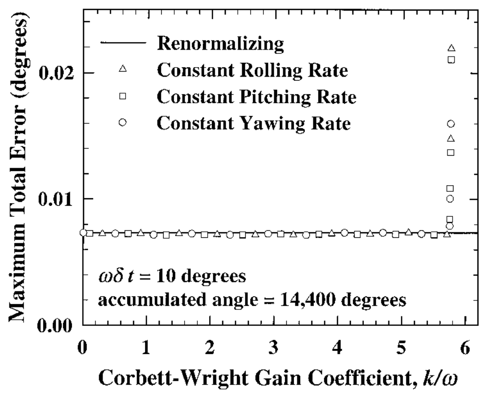

Fig. 6 Effect of changing the gain coefficient on the accuracy of the Fang and Zimmerman algorithm using Corbett–Wright orthogonality control.

renormalization of the quaternion is much faster than using the Corbett–Wright orthogonality control scheme. However, if computation time is not a concern, Corbett–Wright orthogonality control will eliminate the growth of the orthogonality error for simulations using large time steps. If the time steps are small enough to produce about 5 deg of rotation or less per time step, no orthogonality control whatsoever is required with fourth-order Runge–Kutta integration.

If the Corbett–Wright[39] orthogonality control scheme is to be used for large time-step simulations, it should be remembered that this only eliminates the growth of the orthogonality error. It does not eliminate the drift error, which for modern algorithms can be much larger than the orthogonality error. In fact, when a large gain coefficient is used with fourth-order integration and Corbett–Wright orthogonality control, the drift error is increased by more than the orthogonality error is reduced, which actually increases the net error for the simulation. This is shown in Fig. 3. In contrast with the older analog simulations that used a very large gain coefficient, when Corbett–Wright orthogonality control is used with a fourth-order digital integration, a gain coefficient larger than about 2ω should not be used. As can be seen in Fig. 4, this is truly independent of the angular rate. Furthermore, as is seen in Fig. 5, decreasing the time step does not eliminate this problem. For such fourth-order integration, a gain coefficient equal to the magnitude of the angular velocity will effectively eliminate the growth of the orthogonality error associated with large time steps without adversely affecting the drift error. However, for aircraft flight simulations, the angular rates are not typically known a priori. Furthermore, the angular rate is typically changing with time, and a very low angular rate requires a very low gain coefficient for accurate simulation. Thus, to avoid increasing the total error, a variable gain coefficient equal to the angular rate should be used,

$$k = \omega = \sqrt{p^2 + q^2 + r^2} \tag{94}$$

The implementation of Eq. (94) adds substantially to the computational burden of Corbett–Wright[39] orthogonality control. Thus, for greatest computational efficiency, it is much preferred to use periodic renormalization of the quaternion over any use of the Corbett–Wright orthogonality control scheme.

In 1969 Fang and Zimmerman[162] proposed a pseudo-fourth-order method for integrating Eq. (91). With conventional fourth-order Runge–Kutta, the time rate of change of the quaternion is computed from Eq. (91) four times for each time step. With the Fang and Zimmerman algorithm, this same fourth-order procedure is used to integrate Eq. (91), except that ε is held constant through all four computations in each time step. The orthogonality error is updated only once at the beginning of each full time step.

The Fang and Zimmerman[162] algorithm exhibits features of both first-order and fourth-order numerical integration. Although both the angular velocity and the quaternion on the right-hand side of Eq. (91) are estimated to fourth-order accuracy, ε is only estimated to first-order accuracy. Results of integrating Eq. (91) using the Fang and Zimmerman algorithm are shown in Fig. 6. Notice that the accuracy of the method, for small values of the gain coefficient, is comparable to that for full fourth-order integration. However, the method also exhibits an abrupt first-order instability when the gain coefficient is greater than one divided by the time step $k > 1/\delta t$.

Note from Figs. 3–6 that Corbett–Wright[39] orthogonality control offers no significant reduction in simulation error for modern numerical integration algorithms. On the other hand, if the gain coefficient is not judicially chosen, Corbett–Wright orthogonality control can significantly increase the simulation error. Periodic renormalization of the quaternion does not present this problem and is much more computationally efficient.

The computation time required for periodic renormalization of the quaternion can be reduced even further by recognizing that the orthogonality error is always very small for modern fourth-order integration algorithms. Equation (93) can be written as

$$\boldsymbol{e}_r = \boldsymbol{e}\Big/\sqrt{e_0^2 + e_x^2 + e_y^2 + e_z^2} = \boldsymbol{e}/\sqrt{1-\varepsilon} \tag{95}$$

where ε is the error in the square of the quaternion magnitude, as given by Eq. (92). Because ε is very small, Eq. (95) can be closely approximated as[162]

$$\boldsymbol{e}_r \cong \boldsymbol{e}\left(1 + \tfrac{1}{2}\varepsilon\right) \tag{96}$$

or after applying Eq. (92)

$$\boldsymbol{e}_r \cong \boldsymbol{e}\left[1.5 - 0.5\left(e_0^2 + e_x^2 + e_y^2 + e_z^2\right)\right] \tag{97}$$

Periodic application of Eq. (97) is sufficient to eliminate the growth of the orthogonality error for all time.

Renormalizing the quaternion with Eq. (97) requires nine multiplications and four additions. Using Eq. (93), on the other hand, requires eight multiplications, three additions, one square root, and one division. Typical modern hardware requires 1 instruction cycle for addition, 1 instruction cycle for multiplication, something on the order of 4 instruction cycles for division, and anywhere from 4 to 30 instruction cycles for the square root operation, depending on the processor and the compiler. Thus, renormalizing with Eq. (93) requires anywhere from about 50 to 250% greater computation time than using Eq. (97), on typical modern hardware. Furthermore, even if future developments reduce the time required for division and square root to one instruction cycle each, Eq. (97) will be, at worst, computationally equivalent to Eq. (93).

Although periodic renormalization of the quaternion using Eq. (97) provides a computationally efficient means for controlling the orthogonality error, it does not control the drift error. In fact, for higher-order algorithms, it does nothing whatsoever to reduce the total error. In Figs. 3–6, the points that correspond to a Corbett–Wright[39] gain coefficient of zero were obtained using no orthogonality control at all. In each case, the total error obtained with no orthogonality control was the same as that obtained when the quaternion was renormalized after each time step. From this, one may be tempted to conclude that there is no value in orthogonality control. This is not exactly true. If the orthogonality error is controlled, a small amount of drift error does not adversely affect a flight simulator because it is constantly being corrected with virtually imperceptible pilot input. The pilot's perception of the drift error is similar to that caused by an infinitesimal change in the aerodynamics of the aircraft. The orthogonality error, on the other hand, has no counterpart in the physical world and cannot be compensated for by the pilot. When the quaternion magnitude deviates from unity, transformations obtained from Eq. (25) or Eq. (40) become nonorthogonal, and vector length is not preserved through the transformation. Because rotation is an orthogonal transformation, large orthogonality errors can cause physically unrealistic distortion in the kinematic transformations. For large time-step simulations with no orthogonality control, these transformation distortions will continue to increase with time, unaffected by pilot input.

To demonstrate the effects of both orthogonality error and drift error on the visual display, Fig. 7 shows a three-dimensional image displayed after four different quaternion simulations. Each of the four simulations was carried out using fourth-order Adams–Bashforth–Moulton numerical integration. In each case, all aspects of the integration, except the error control, were the same. All four images are viewed from exactly the same position and with exactly the same perspective. The differences in the images result entirely from numerical integration errors in the transformation quaternion.

The image in Fig. 7a is essentially error free. The orthogonality error was removed using periodic renormalization, and the drift error was eliminated with pilot input. The darker image in Fig. 7b was generated using periodic renormalization without pilot input and thus contains no orthogonality error but about 17 deg of drift error. To emphasize the error, this image has been overlaid on an error-free image in light gray. The integration used to obtain the darker image

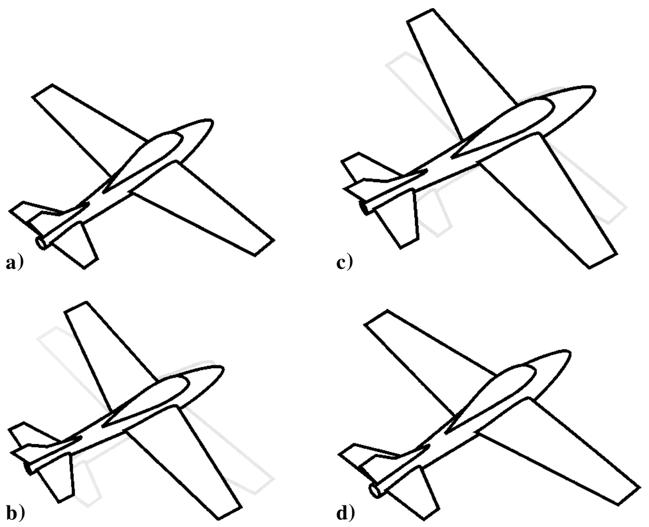

Fig. 7 Effects of drift error and orthogonality error on the visual display after 2×10^6 iterations using fourth-order Adams–Bashforth–Moulton with a) periodic renormalization, no orthogonality error, and drift error eliminated by pilot; b) periodic renormalization, no orthogonality error, and 17-deg drift error; c) no orthogonality control, 8% orthogonality error, and 17-deg drift error; d) no orthogonality control, 8% orthogonality error, and drift error eliminated by pilot.

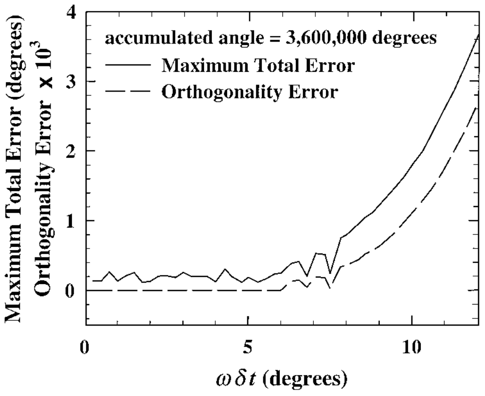

Fig. 8 Effect of step size on the accuracy of a fourth-order Runge–Kutta algorithm with no orthogonality control, using single-precision computations.

in Fig. 7c had no orthogonality control or pilot input and produced about 8% orthogonality error and about 17 deg of drift error. Again, the light gray image in Fig. 7c is error free. When Figs. 7b and 7c are compared, it can be seen that orthogonality error produces a scaling distortion in the visual display. As is seen in Fig. 7d, pilot input eliminates the drift error but does nothing to eliminate the scaling distortion caused by the orthogonality error. It can readily be shown from Eq. (40) that, when orthogonality error is present, all objects in the visual display are scaled by a factor of $|\boldsymbol{e}|^2$.

Because the orthogonality error is only a small fraction of the total error, reducing the size of the time step and/or increasing the order of the numerical integration algorithm are the only effective ways to reduce total error. The effects of time-step size on the orthogonality error and the total error, for a fourth-order Runge–Kutta algorithm with no orthogonality control, are shown in Figs. 8 and 9, for both single- and double-precision computations. Figures 8 and 9 show a common misconception concerning the origin of the orthogonality error, which is often attributed to the roundoff error associated with the finite word size used for numerical computation.[165] As can be seen by comparing Fig. 8 with Fig. 9, neither the orthogonality error nor the total error is significantly affected by the computation precision for a large step size. Furthermore, for a step size of less than about 5 deg, the single-precision orthogonality error is exactly zero, whereas the double-precision orthogonality error is very small

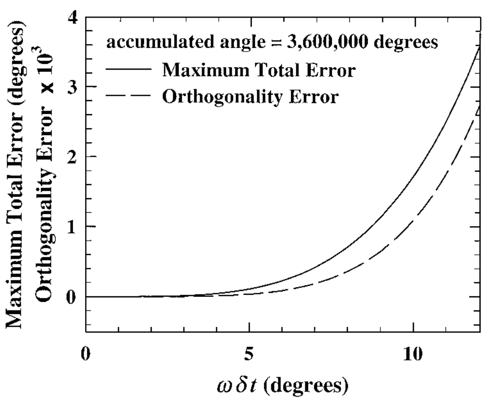

Fig. 9 Effect of step size on the accuracy of a fourth-order Runge–Kutta algorithm with no orthogonality control, using double-precision computations.

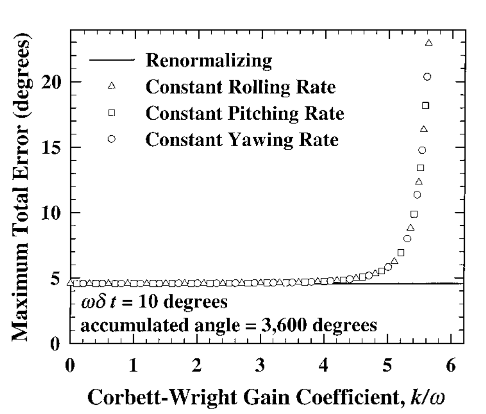

Fig. 10 Effect of changing the gain coefficient on the accuracy of a second-order Runge–Kutta algorithm using Corbett–Wright orthogonality control.

but finite. This is because orthogonality error does not result from the roundoff error associated with word size and computation precision. It is rather the result of errors associated with the order of the integration algorithm. This numerical integration error is called truncation error because it results from truncating the Taylor series for differentiation to a finite number of terms.

In an aircraft flight simulation, computing the time derivatives of the translational and angular velocity components requires computation of the aerodynamic forces and moments. Depending on the method used, these computations can be quite time consuming. Because a digital simulation using the fourth-order Runge–Kutta algorithm requires evaluating these time derivatives four times for each time step, first-order and second-order integration methods have occasionally been used in an attempt to reduce computation time. As shall be demonstrated, this is not a particularly good choice.

Results from a second-order Runge–Kutta integration with Corbett–Wright[39] orthogonality control are shown in Fig. 10. Notice that, as was the case for fourth-order integration, Corbett–Wright orthogonality control offers no reduction in total simulation error for this second-order integration. Furthermore, for the time step used, the accuracy of this second-order simulation is not significantly improved over that of the first-order simulation shown in Fig. 2. At first thought, this may seem counterintuitive.

From the theory of numerical methods we know that, in the limit as the step size approaches zero, the global error for the first-order integration should be linearly proportional to the step size, whereas the global error for the second-order integration should be proportional to the step size squared. For very small step size, this is confirmed in Fig. 11. However, for larger step size, the global error results primarily from the higher-order terms, which have been neglected in both the first- and second-order algorithms. Thus, for larger time steps, the second-order algorithm offers little increase in accuracy over the first-order algorithm. This can be seen in Fig. 12.

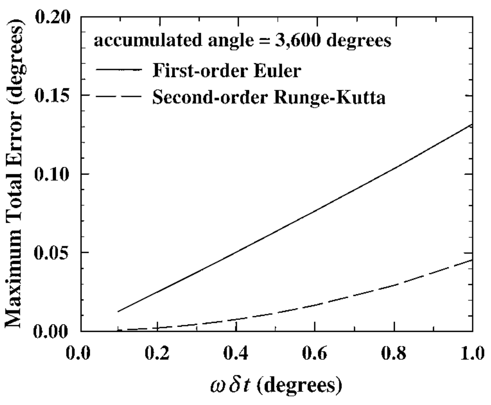

Fig. 11 Comparison between first-order Euler integration and second-order Runge–Kutta integration for small time steps.

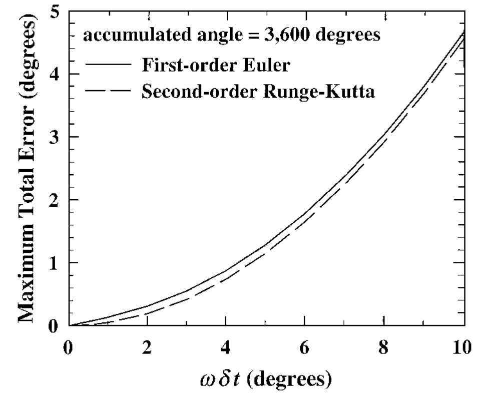

Fig. 12 Comparison between first-order Euler integration and second-order Runge–Kutta integration for large time steps.

When digital computers first became practical for aircraft flight simulation, new integration algorithms were developed in an attempt to reduce computation time without sacrificing accuracy. In 1963, Pope[166] proposed one such algorithm, which was called the exponential method. This algorithm was first used to integrate the quaternion formulation at the NASA Langley Research Center in 1973.[167] In the aircraft community, the method is commonly known as the local linearization method. Because the method has been used in aircraft flight simulation for many years but is not commonly used elsewhere, a development of this algorithm will be presented here.

The differential equations governing the change in the components of the quaternion $\boldsymbol{e}$ with time are given by Eq. (32). This system of differential equations written in matrix notation is

$$\dot{\boldsymbol{e}} = [\boldsymbol{M}]\boldsymbol{e} \tag{98}$$

where

$$[\boldsymbol{M}] = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \tag{99}$$

The right-hand side of Eq. (98) can be expanded in a Taylor series about $t = t_i$ and $\boldsymbol{e} = \boldsymbol{e}_i$

$$\begin{aligned} \dot{\boldsymbol{e}} &= [\boldsymbol{M}]_i\boldsymbol{e}_i + [\boldsymbol{M}]_i(\boldsymbol{e} - \boldsymbol{e}_i) + [\dot{\boldsymbol{M}}]_i\boldsymbol{e}_i(t - t_i) \\ &\quad + (1/2!)\left\{2[\dot{\boldsymbol{M}}]_i(\boldsymbol{e} - \boldsymbol{e}_i)(t - t_i) + [\ddot{\boldsymbol{M}}]_i\boldsymbol{e}_i(t - t_i)^2\right\} \\ &\quad + (1/3!)\left\{3[\ddot{\boldsymbol{M}}]_i(\boldsymbol{e} - \boldsymbol{e}_i)(t - t_i)^2 + [\dddot{\boldsymbol{M}}]_i\boldsymbol{e}_i(t - t_i)^3\right\} + \cdots \end{aligned} \tag{100}$$

where the subscript i indicates evaluation at time $t = t_i$.

Equation (100) forms the basis for the local linearization algorithms that have been used to integrate the quaternion formulation. These algorithms are based on the method proposed by Pope,[166] which was originally called the exponential method. Barker et al.[167] first applied this method to the quaternion formulation using a first-order approximation to Eq. (100),

$$\dot{\boldsymbol{e}} \cong [\boldsymbol{M}]_i \boldsymbol{e}_i + [\boldsymbol{M}]_i(\boldsymbol{e} - \boldsymbol{e}_i) + [\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i (t - t_i) \tag{101}$$

In an attempt to improve the local linearization algorithm, Yen and Cook[168] used the approximation

$$\dot{\boldsymbol{e}} \cong [\boldsymbol{M}]_i \boldsymbol{e}_i + [\boldsymbol{M}]_i(\boldsymbol{e} - \boldsymbol{e}_i) + [\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i (t - t_i) + [\boldsymbol{M}]_i[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i (t - t_i)^2 \tag{102}$$

Equation (102) includes an approximation for the second term on the right-hand side of Eq. (100). Within the second term, this approximation implies

$$(\boldsymbol{e} - \boldsymbol{e}_i) \cong \dot{\boldsymbol{e}}_i (t - t_i) = [\boldsymbol{M}]_i \boldsymbol{e}_i (t - t_i)$$

and

$$[\ddot{\boldsymbol{M}}] \cong 0$$

Because Eq. (102) completely ignores that portion of the second-order term from Eq. (100) that contains $[\ddot{\boldsymbol{M}}]$, in general, this equation should not be expected to give higher-order accuracy than Eq. (101). This was confirmed by the numerical results of Yen and Cook.[168]

From Eqs. (99) and (100), we see that a full second-order approximation for the right-hand side of Eq. (100) would require knowledge of the second derivative of the angular rate vector. Although the first derivative of the angular rate vector is always available from the equations of motion, its second derivative is not directly available in a six-DOF flight simulation. For this reason, a complete second-order approximation to the right-hand side of Eq. (100) has not been used. Because the approximation given by Eq. (102) is no more accurate than that given by Eq. (101), only the local linearization algorithm based on Eq. (101) will be presented here.

The local linearization algorithm of Barker et al.[167] has been used in aircraft flight simulation for many years. This algorithm is based on the closed-form solution to Eq. (101),

$$\begin{aligned}\boldsymbol{e}_{i+1} = {} & [\exp([\boldsymbol{M}]_i \delta t)]\boldsymbol{e}_i + \{[\exp([\boldsymbol{M}]_i \delta t)] - [\boldsymbol{I}] \\ & - [\boldsymbol{M}]_i \delta t\}[\boldsymbol{M}]_i^{-1}[\boldsymbol{M}]_i^{-1}[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i \end{aligned} \tag{103}$$

where $\delta t = t_{i+1} - t_i$. Applying Eq. (87) to replace the matrix exponential in Eq. (103) gives

$$\begin{aligned}\boldsymbol{e}_{i+1} = {} & \cos(\omega_i \delta t/2)\boldsymbol{e}_i + (2/\omega_i)\sin(\omega_i \delta t/2)[\boldsymbol{M}]_i \boldsymbol{e}_i \\ & + [\cos(\omega_i \delta t/2) - 1][\boldsymbol{M}]_i^{-1}[\boldsymbol{M}]_i^{-1}[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i + [(2/\omega_1)\sin(\omega_i \delta t/2) \\ & - \delta t][\boldsymbol{M}]_i[\boldsymbol{M}]_i^{-1}[\boldsymbol{M}]_i^{-1}[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i \end{aligned} \tag{104}$$

From Eq. (99), it is readily shown that the inverse of $[\boldsymbol{M}]$ is

$$[\boldsymbol{M}]^{-1} = -(4/\omega^2)[\boldsymbol{M}] \tag{105}$$

and

$$[\boldsymbol{M}]^{-1}[\boldsymbol{M}]^{-1} = -(4/\omega^2)[\boldsymbol{M}][\boldsymbol{M}]^{-1} = -(4/\omega^2)[\boldsymbol{I}] \tag{106}$$

Thus, when Eq. (106) is used in Eq. (104), the local linearization algorithm can be written as

$$\begin{aligned}\boldsymbol{e}_{i+1} = {} & \cos(\omega_i \delta t/2)\boldsymbol{e}_i + (2/\omega_i)\sin(\omega_i \delta t/2)[\boldsymbol{M}]_i \boldsymbol{e}_i \\ & + \left(4\big/\omega_i^2\right)[1 - \cos(\omega_i \delta t/2)][\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i \\ & + \left(4\big/\omega_i^2\right)[\delta t - (2/\omega_i)\sin(\omega_i \delta t/2)][\boldsymbol{M}]_i[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i \end{aligned} \tag{107}$$

Notice that, for the special case where $[\boldsymbol{M}]$ is constant, Eq. (107) reduces to the exact solution given by Eq. (88).

Also note that Eq. (107) is numerically indeterminate for the special case when ω_i is zero. However, when the series expansion is used for the sine and the cosine, it is readily shown that when ω_i is zero Eq. (107) reduces to

$$\boldsymbol{e}_{i+1} \underset{\omega_i \to 0}{=} \boldsymbol{e}_i + \delta t[\boldsymbol{M}]_i \boldsymbol{e}_i + (\delta t^2/2)[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i + (\delta t^3/6)[\boldsymbol{M}]_i[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i \tag{108}$$

Furthermore, from Eq. (99) we see that when ω_i is zero, all 16 components of $[\boldsymbol{M}]$ are also zero. Thus, even though Eq. (108) is not numerically indeterminate, it can be simplified to give

$$\boldsymbol{e}_{i+1} \underset{\omega_i \to 0}{=} \boldsymbol{e}_i + (\delta t^2/2)[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i \tag{109}$$

Equations (107) and (109) provide the foundation for the local linearization algorithm.

The order of the truncation error associated with the local linearization algorithm can be evaluated by using the Taylor series expansion, which is defined in Eq. (85), for the matrix exponential in Eq. (103). This gives

$$\begin{aligned}\boldsymbol{e}_{i+1} = {} & \left\{[\boldsymbol{I}] + [\boldsymbol{M}]_i \delta t + [\boldsymbol{M}]_i[\boldsymbol{M}]_i \delta t^2\big/2! + \cdots\right\}\boldsymbol{e}_i \\ & + \left\{[\boldsymbol{M}]_i[\boldsymbol{M}]_i \delta t^2\big/2! + \cdots\right\}[\boldsymbol{M}]_i^{-1}[\boldsymbol{M}]_i^{-1}[\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i\end{aligned}$$

or

$$\begin{aligned}\boldsymbol{e}_{i+1} = {} & \boldsymbol{e}_i + [\boldsymbol{M}]_i \boldsymbol{e}_i \delta t + \{([\boldsymbol{M}]_i[\boldsymbol{M}]_i + [\dot{\boldsymbol{M}}]_i)/2!\}\boldsymbol{e}_i \delta t^2 \\ & + \{([\boldsymbol{M}]_i[\boldsymbol{M}]_i[\boldsymbol{M}]_i + [\boldsymbol{M}]_i[\dot{\boldsymbol{M}}]_i)/3!\}\boldsymbol{e}_i \delta t^3 + \cdots\end{aligned} \tag{110}$$

The Taylor series expansion of the exact solution is

$$\boldsymbol{e}_{i+1} = \boldsymbol{e}_i + \dot{\boldsymbol{e}}_i \delta t + (\ddot{\boldsymbol{e}}_i/2!)\delta t^2 + (\dddot{\boldsymbol{e}}_i/3!)\delta t^3 + \cdots \tag{111}$$

From Eq. (98) we have

$$\dot{\boldsymbol{e}}_i = [\boldsymbol{M}]_i \boldsymbol{e}_i \tag{112}$$

$$\ddot{\boldsymbol{e}}_i = [\boldsymbol{M}]_i \dot{\boldsymbol{e}}_i + [\dot{\boldsymbol{M}}]_i \boldsymbol{e}_i = \{[\boldsymbol{M}]_i[\boldsymbol{M}]_i + [\dot{\boldsymbol{M}}]_i\}\boldsymbol{e}_i \tag{113}$$

$$\begin{aligned}\dddot{\boldsymbol{e}}_i = {} & [\boldsymbol{M}]_i \ddot{\boldsymbol{e}}_i + 2[\dot{\boldsymbol{M}}]_i \dot{\boldsymbol{e}}_i + [\ddot{\boldsymbol{M}}]_i \boldsymbol{e}_i = \{[\boldsymbol{M}]_i[\boldsymbol{M}]_i[\boldsymbol{M}]_i + [\boldsymbol{M}]_i[\dot{\boldsymbol{M}}]_i \\ & + 2[\dot{\boldsymbol{M}}]_i[\boldsymbol{M}]_i + [\ddot{\boldsymbol{M}}]_i\}\boldsymbol{e}_i\end{aligned} \tag{114}$$

Using Eqs. (112–114) in Eq. (111) gives

$$\begin{aligned}\boldsymbol{e}_{i+1} = {} & \boldsymbol{e}_i + [\boldsymbol{M}]_i \boldsymbol{e}_i \delta t + \{([\boldsymbol{M}]_i[\boldsymbol{M}]_i + [\dot{\boldsymbol{M}}]_i)/2!\}\boldsymbol{e}_i \delta t^2 \\ & + \{([\boldsymbol{M}]_i[\boldsymbol{M}]_i[\boldsymbol{M}]_i + [\boldsymbol{M}]_i[\dot{\boldsymbol{M}}]_i \\ & + 2[\dot{\boldsymbol{M}}]_i[\boldsymbol{M}]_i + [\ddot{\boldsymbol{M}}]_i)/3!\}\boldsymbol{e}_i \delta t^3 + \cdots\end{aligned} \tag{115}$$

When the exact expansion from Eq. (115) is compared with the expansion of the local linearization algorithm that is given by Eq. (110), the local truncation error for this algorithm is found to be

$$R_i = \{(2[\dot{\boldsymbol{M}}]_i[\boldsymbol{M}]_i + [\ddot{\boldsymbol{M}}]_i)/3!\}\boldsymbol{e}_i \delta t^3 + \mathcal{O}(\delta t^4) + \cdots \tag{116}$$

Thus, the method produces third-order truncation error with components proportional to both the first and second derivatives of the angular rate vector. It does not provide an exact solution, except for the special case of constant rotation. Because the truncation error is of order δt^3, the local linearization method provides second-order accuracy in a single-point, single-step, integration algorithm.

A more widely known integration method, which was also developed to reduce computation time over that required for fourth-order Runge–Kutta, is fourth-order Adams–Bashforth–Moulton (see Ref. 161). This commonly used numerical integration scheme is a four-point, two-step, backward-difference method. Because four starting values are required, the method must be started using a single-point method such as fourth-order Runge–Kutta. However, after the first three time steps, this method requires evaluating the

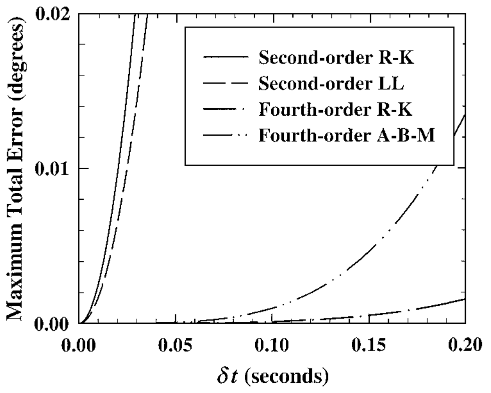

Fig. 13 Comparison between second-order Runge–Kutta, second-order local linearization, fourth-order Runge–Kutta, and fourth-order Adams–Bashforth–Moulton integration.

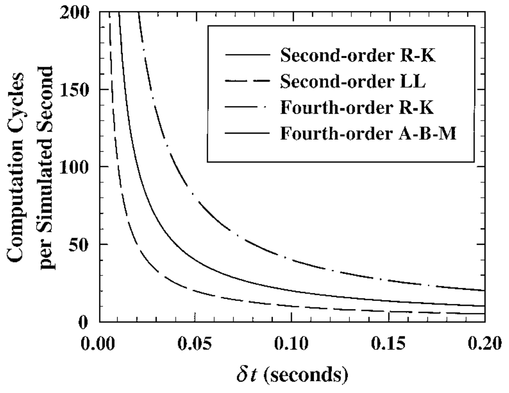

Fig. 14 Computation cycles per simulated second as a function of step size for four numerical integration algorithms.

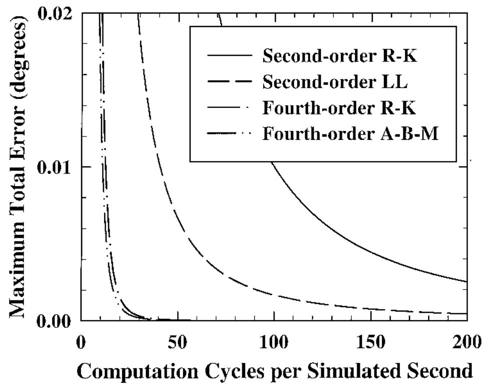

Fig. 15 Simulation error as a function of computation cycles per simulated second for four numerical integration algorithms.

time derivatives only twice for each time step. This provides fourth-order accuracy similar to that provided by fourth-order Runge–Kutta, but at a computational cost similar to that required for second-order Runge–Kutta. Because the development of the fourth-order Adams–Bashforth–Moulton algorithm is presented in most undergraduate textbooks on numerical methods, it will not be repeated here.

In general, simulation error can be reduced by either reducing the size of the time step or by increasing the order of the integration method. Both require greater computation time. Figure 13 shows how total error varies with step size for four different integration algorithms, second-order Runge–Kutta, second-order local linearization, fourth-order Runge–Kutta, and fourth-order Adams–Bashforth–Moulton. The results shown in Fig. 13 were obtained from numerical integration of the quaternion formulation for the special case of a sinusoidal rotation vector,

$$\begin{Bmatrix} p \\ q \\ r \end{Bmatrix} = \begin{Bmatrix} \omega_{ax}\sin(\omega_{fx}t) \\ \omega_{ay}\sin(\omega_{fy}t) \\ \omega_{az}\sin(\omega_{fz}t) \end{Bmatrix}$$

The error was defined relative to a fourth-order Runge–Kutta solution using very small time steps. As expected, the fourth-order algorithms produce less error for a given time step than do the second-order algorithms. However, the higher-order algorithms also require a greater number of computation cycles for each time step. For this reason, it may not be directly obvious which of the four methods will result in the fastest computation for a given level of accuracy.

The second-order local linearization algorithm requires only one cycle of computation for the aerodynamic forces and moments at each time-step. The second-order Runge–Kutta algorithm and the fourth-order Adams–Bashforth–Moulton algorithm each requires two computation cycles per time step, whereas the fourth-order Runge–Kutta algorithm requires four. For the same simulations that were used to produce Fig. 13, the number of computation cycles per simulated second are shown as a function of step size in Fig. 14.

Because computation time depends on the method of integration as well as the step size, Fig. 13 does not present a fair comparison of the error produced by the four algorithms. Because computation time is proportional to the number of computation cycles per simulated second, a better comparison is shown in Fig. 15. In Fig. 15, the total simulation error is plotted as a function of the number of computation cycles per simulated second. From Fig. 15, it is seen that fourth-order Adams–Bashforth–Moulton produces the smallest error for a given level of computation. Furthermore, it can be seen that a simulation using fourth-order Runge–Kutta is only slightly less efficient than one using fourth-order Adams–Bashforth–Moulton. In addition, Fig. 15 shows that the second-order algorithms are much less efficient than either of the fourth-order algorithms. These low-order algorithms should not be used, unless for some other reason very small time steps are required. The maximum total error shown in Figs. 13 and 15 was computed using

$$\omega_{ax} = \omega_{ay} = \omega_{az} = \omega_{fx} = \omega_{fy} = \omega_{fz} = 1.0\,\text{rad/s}$$

The absolute magnitude of the error shown in Figs. 13 and 15 depends on the angular rates. However, the relative comparison between the errors realized by the four different numerical methods is independent of these angular rates.

Conclusions

Among some fraction of the aircraft community, the quaternion formulation has gained the somewhat undeserved reputation of being hard to understand. This reputation has more to do with the scattered nature of the literature on this topic, particularly in aircraft journals and textbooks, than it does with the raw complexity of the quaternion formulation itself. Information from over 160 references, pertinent to attitude kinematics, was assembled in an attempt to provide a clearer understanding of the quaternion formulation. To this end, a systematic development of the kinematic transformation equations in terms of Euler angles, direction cosines, the Euler axis rotation parameters, and the Euler–Rodrigues quaternion was presented. The kinematic equations, including the gravitational vector and the effects of wind were presented for these four attitude representations. The singularities in the both the Euler angle formulation and the Euler axis formulation were discussed. Because the physical interpretation of the Euler–Rodrigues quaternion is much less intuitive than that associated with Euler angles, relationships between the quaternion and the other attitude descriptors were presented.

To avoid the perceived complexity of the quaternion formulation, the aircraft community has often implemented either the direction-cosine formulation or the Euler angle formulation for attitude description in aircraft flight simulators. This has been done at significant computational cost. In addition to eliminating the singularity in the Euler angle formulation, the quaternion formulation is far superior to either the Euler angle formulation or the direction-cosine formulation, based on computational efficiency alone. Numerical integration of the nine component direction-cosine formulation requires more than double the computation time needed for the four-component quaternion formulation. Utilizing quaternion algebra, it was shown that the Euler angle transformation requires about 11 times as long to evaluate as the quaternion transformation. Consequently, even in cases where the aircraft is not an all-attitude vehicle and the gimbal lock singularity is not encountered, the quaternion formulation provides important computational savings that should be seriously considered.

To take full advantage of the speed and accuracy of the Euler–Rodrigues quaternion formulation, it is necessary to give some consideration to the numerical method used to integrate the kinematic equations. Early aircraft flight simulations were run on analog computers that were inherently first order. Numerical integration of the quaternion rate equations with a first-order method results in very large errors. Early analog programmers observed a similar behavior and employed an error reduction scheme, which in this paper is called Corbett–Wright[39] orthogonality control. This orthogonality control is still found in some aircraft simulation codes that are in use today. Integration with any of today's prevalent fourth-order numerical methods produces very little orthogonality error. Using the Corbett–Wright orthogonality control scheme with modern numerical algorithms increases the computational time but does little to improve the accuracy of the simulation.

Even though the orthogonality error for modern numerical algorithms is very small, it can accumulate. Periodic renormalization of the quaternion eliminates this error. The Corbett–Wright[39] orthogonality control scheme will also eliminate the growth of the orthogonality error, but it does not eliminate the drift error. In some cases, for certain values of the Corbett–Wright gain coefficient, the drift error is increased by more than the orthogonality error is reduced, which actually increases the total error for the simulation. Periodic renormalization of the quaternion provides a computationally efficient means for controlling orthogonality error without increasing the drift error, but it does not eliminate drift error. If the orthogonality error is controlled, a small amount of drift error does not adversely affect a flight simulator because it is constantly being corrected with virtually imperceptible pilot input. The pilot's perception of the drift error is similar to that caused by an infinitesimal change in the aerodynamics of the aircraft. The orthogonality error, on the other hand, has no counterpart in the physical world and cannot be compensated for by the pilot. Reduction in size of the time step and/or increasing the order of the integration are the only effective ways to reduce total error.

Acknowledgments

The authors would like to thank Stephen D. Derry at the NASA Langley Research Center for reviewing this paper and providing some very helpful suggestions. The authors also appreciate the editorial and bibliographic efforts of Joan P. Smith.

References

[1]Whittaker, E. T., *A Treatise on the Analytical Dynamics of Particles and Rigid Bodies*, 4th ed., Cambridge Univ. Press, Cambridge, England, U.K., 1937, pp. 1–14.

[2]Goldstein, H., *Classical Mechanics*, 2nd ed., Addison Wesley Longmann, Reading, MA, 1981, pp. 148–158.

[3]Nakano, A., "Rigid-Body-Based Multiple Time Scale Molecular Dynamics Simulation of Nanophase Materials," *International Journal of High Performance Computing Applications*, Vol. 13, No. 2, 1999, pp. 154–162.

[4]Hover, F. S., "Simulation of Massless Tethers," *Ocean Engineering*, Vol. 24, No. 8, 1997, pp. 765–783.

[5]Smith, R., Frost, A., and Probert, P., "A Sensor System for the Navigation of an Underwater Vehicle," *International Journal of Robotics Research*, Vol. 18, No. 7, 1999, pp. 697–710.

[6]Fjellstad, O. E., and Fossen, T. I., "Position and Attitude Tracking of AUV's: A Quaternion Feedback Approach," *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 4, 1994, pp. 512–518.

[7]Chou, J. C. K., and Kamel, M., "Finding the Position and Orientation of a Sensor on a Robot Manipulator Using Quaternions," *International Journal of Robotics Research*, Vol. 10, No. 3, 1991, pp. 240–254.

[8]Wee, L. G., Walker, M. W., and McClamroch, N. H., "An Articulated-Body Model for a Free-Flying Robot and Its Use for Adaptive Motion Control," *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 2, 1997, pp. 264–277.

[9]Dornaika, F., and Horaud, R., "Simultaneous Robot-World and Hand-Eye Calibration," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 4, 1998, pp. 617–622.

[10]Caccavale, F., Natale, C., Siciliano, B., and Villani, L., "Six-DOF Impedance Control Based on Angle/Axis Representations," *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 2, 1999, pp. 289–300.

[11]Daniilidis, K., "Hand-Eye Calibrations Using Dual Quaternions," *International Journal of Robotics Research*, Vol. 18, No. 3, 1999, pp. 286–298.

[12]Omelyan, I. P., "On the Numerical Integration of Motion for Rigid Polyatomics: the Modified Quaternion Approach," *Computers in Physics*, Vol. 12, No. 1, 1998, pp. 97–103.

[13]Kuipers, J. B., "SPASYN an Electromagnetic Relative Position and Orientation Tracking System," *IEEE Transactions on Instrumentation and Measurements*, Vol. 29, No. 4, 1980, pp. 462–471.

[14]Shoemake, K., "Animating Rotation with Quaternion Curves," *Computer Graphics*, Vol. 19, No. 3, 1985, pp. 245–254.

[15]Hanson, A. J., and Ma, H., "Quaternion Frame Approach to Streamline Visualization," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1., No. 2, 1995, pp. 164–174.

[16]Hart, J. C., Francis, G. K., and Kauffman, L. H., "Visualizing Quaternion Rotation," *ACM Transactions on Graphics*, Vol. 13, No. 3, 1994, pp. 256–276.

[17]Chou, J. C. K., "Quaternion Kinematics and Dynamic Differential Equations," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, 1992, pp. 53–64.

[18]Shuster, M. D., "A Survey of Attitude Representations," *Journal of the Astronautical Sciences*, Vol. 14, No. 4, 1993, pp. 439–517.

[19]Perkins, C. D., and Hage, R. E., *Airplane Performance Stability and Control*, Wiley, New York, 1949, pp. 374–474.

[20]Duncan, W. J., *The Principles of Control and Stability of Aircraft*, Cambridge Univ. Press, Cambridge, England, U.K., 1952, pp. 61–92.

[21]Babister, A. W., *Aircraft Stability and Control*, Oxford Univ. Press, Oxford, England, U.K., 1961, pp. 1–700.

[22]Kolk, W. R., *Modern Flight Dynamics*, Prentice–Hall, Englewood Cliffs, NJ, 1961, pp. 22–40.

[23]Etkin, B., *Dynamics of Atmospheric Flight*, Wiley, New York, 1972, pp. 121–154.

[24]McRuer, D. T., Ashkenas, I. L., and Graham, D., *Aircraft Dynamics and Automatic Control*, Princeton Univ. Press, Princeton, NJ, 1973, pp. 203–295.

[25]Babister, A. W., *Aircraft Dynamic Stability and Response*, Pergamon, Oxford, England, U.K., 1980, pp. 11–28.

[26]McCormick, B. W., *Aerodynamics, Aeronautics, and Flight Mechanics*, 2nd ed., Wiley, New York, 1995, pp. 538–582.

[27]Etkin, B., and Reid, L. D., *Dynamics of Flight: Stability and Control*, 3rd ed., Wiley, New York, 1996, pp. 93–128.

[28]Cook, M. V., *Flight Dynamics Principles*, Wiley, New York, 1997, pp. 11–29.

[29]Nelson, R. C., *Flight Stability and Automatic Control*, 2nd ed., McGraw–Hill, New York, 1998, pp. 96–130.

[30]Roskam, J., *Airplane Flight Dynamics and Automatic Flight Control*, Design, Analysis and Research Corp., Lawrence, KS, 1998, pp. 3–34.

[31]Schmidt, L. V., *Introduction to Aircraft Flight Dynamics*, AIAA, Reston, VA, 1998, pp. 93–118.

[32]Abzug, M. J., *Computational Flight Dynamics*, AIAA, Reston, VA, 1998, pp. 5–7.

[33]Pamadi, B. N., *Performance, Stability, Dynamics, and Control of Airplanes*, AIAA, Reston, VA, 1998, pp. 344–347.

[34]Abzug, M. J., and Larrabee, E. E., *Airplane Stability and Control: A History of Technologies that Made Aviation Possible*, Cambridge Univ. Press, Cambridge, England, U.K., 1997, pp. 251–259.

[35]Jackson, E. B., "Manual for a Workstation-Based Generic Flight Simulation Program (LaRCsim) Version 1.4," NASA TM-110164, April 1995.

[36]Robinson, A. C., "On the Use of Quaternions in the Simulation of Rigid Body Motion," *Proceedings of the Simulation Council Session of the Eastern Joint Computer Conference*, MacMillan, New York, 1958, pp. 1–18.

[37]Robinson, A. C., "On the Use of Quaternions in the Simulation of Rigid Body Motion," Rept. WADC TR 58-17, Wright–Patterson Air Force Base, OH, Dec. 1958.

[38]Mitchell, E. E. L., and Rogers, A. E., "Quaternion Parameters in the Simulation of a Spinning Rigid Body," *Simulation*, Vol. 4, No. 6, 1965, pp. 390–396.

[39]Corbett, J. P., and Wright, F. B., "Stabilization of Computer Circuits," Rept. WADC TR 57-425, edited by E. Hochfeld, Wright–Patterson Air Force Base, OH, 1957.

[40]Rolfe, J. M., and Staples, K. J., *Flight Simulation*, Cambridge Univ.

Press, Cambridge, England, U.K., 1988, pp. 380–440.

[41]Kaplan, M. H., *Modern Spacecraft Dynamics and Control*, Wiley, New York, 1976, pp. 1–415.

[42]Messiah, A., *Quantum Mechanics*, North–Holland, Amsterdam, Vol. 1, 1961, pp. 1–504.

[43]Messiah, A., *Quantum Mechanics*, North–Holland, Amsterdam, Vol. 2, 1962, pp. 505–1136.

[44]Tandon, G. K., "Coordinate Transformations," *Spacecraft Attitude Determination and Control*, edited by J. R. Wertz, Kluwer Academic, Dordrecht, The Netherlands, 1978, pp. 760–766.

[45]Pio, R. L., "Symbolic Representation of Coordinate Transformation," *IEEE Transactions on Aerospace and Navigational Electronics*, Vol. 11, June 1964, pp. 128–134.

[46]Pio, R. L., "Euler Angle Transformations," *IEEE Transactions on Automatic Control*, Vol. 11, No. 4, 1966, pp. 707–715.

[47]Bar-Itzhack, I. Y., "The Program: An Essay Tool for Angular Position and Rate Computations," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 519–529.

[48]Mayer, A., "Rotations and Their Algebra," *SIAM Review*, Vol. 2, No. 2, 1960, pp. 77–122.

[49]Hughes, P. C., *Spacecraft Attitude Dynamics*, Wiley, New York, 1986, pp. 1–564.

[50]Kane, T. R., Likins, P. W., and Levinson, D. A., *Spacecraft Dynamics*, McGraw–Hill, New York, 1983, pp. 1–436.

[51]Junkins, J. L., and Shuster, M. D., "The Geometry of the Euler Angles," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 531–543.

[52]Markley, F. L., "New Dynamic Variables for Momentum-Bias Spacecraft," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 557–567.

[53]Broucke, R. A., "On the Use of Poincare Surfaces of Section in Rigid Body Motion," *Journal of Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 593–601.

[54]Kolve, D. I., "Describing an Attitude," *Advances in the Astronautical Sciences, Proceedings of the 16th AAS Rocky Mountain Guidance and Control Conference*, Vol. 81, Univelt, San Diego, CA, 1993, pp. 289–303.

[55]Markley, F. L., "Attitude Dynamics," *Spacecraft Attitude Determination and Control*, edited by J. R. Wertz, Kluwer Academic, Dordrecht, The Netherlands, 1978, pp. 510–521.

[56]Sabroff, A., Farrenkopf, R., Frew, A., and Gran, M., "Investigation of the Acquisition Problem in Satellite Attitude Control," Air Force Flight Dynamics Lab., TR-65-115, TRW, Redondo Beach, CA, Dec. 1965.

[57]Sidi, M. J., *Spacecraft Dynamics and Control,* Cambridge Univ. Press, Cambridge, England, U.K., 1997, pp. 318–327.

[58]Satake, I., *Linear Algebra*, Marcel Dekker, New York, 1975, pp. 198–208.

[59]Bryan, T. R., Lewis, D. G., and Perry, D. D., "Software User's Manual for TRAP 3.1," TAG 92-03, Battelle, Columbus, OH, March 1993.

[60]Euler, L., "Formulae Generales pro Translatione Quacunque Corporum Rigidorum," *Novi Commentari Academiae Scientiarum Imperialis Petropolitanae*, Vol. 20, 1775, pp. 189–207.

[61]Euler, L., "Nova Methodus Motum Corporum Rigidorum Determinandi," *Novi Commentari Academiae Scientiarum Imperialis Petropolitanae*, Vol. 20, 1775, pp. 208–238.

[62]Beatty, M. F., "Vector Analysis of Finite Rigid Rotations," *Journal of Applied Mechanics*, Vol. 44, No. 3, 1977, pp. 501, 502.

[63]Kozik, T. J., "Finite Rotations and Associated Displacement Vectors," *Journal of Applied Mechanics*, Vol. 43, No. 4, 1976, pp. 698, 699.

[64]Mathews, J., "Coordinate-Free Rotation Formalism," *American Journal of Physics*, Vol. 44, No. 12, 1976, p. 1210.

[65]Pearlman, N., "Vector Representation of Rigid-Body Rotations," *American Journal of Physics*, Vol. 35, No. 12, 1967, pp. 1164–1167.

[66]Beatty, M. F., "Kinematics of Finite, Rigid-Body Displacements," *American Journal of Physics*, Vol. 35, No. 10, 1967, pp. 949–954.

[67]Beatty, M. F., "Vector Representation of Rigid Body Rotation," *American Journal of Physics*, Vol. 31, No. 2, 1963, pp. 134, 135.

[68]Grubin, C., "Vector Representation of Rigid Body Rotation," *American Journal of Physics*, Vol. 30, No. 6, 1962, pp. 416, 417.

[69]Euler, L., "Problema Algebraicum ob Affectiones Prorsus Singulares Memorabile," *Novi Commentari Academiae Scientiarum Imperialis Petropolitanae*, Vol. 15, 1770, pp. 75–106.

[70]Rodrigues, O., "Des Lois Géométriques qui Régissent les Déplacements d'un Système Solide dans l'espace, et de la Variation des Coordonneés Provenant de ses Déplacements Considereés Indépendamment des Causes qui Peuvent les Produire," *Journal des Mathematiques Pures et Appliquees*, Vol. 5, 1840, pp. 380–440.

[71]Cayley, A., "On Certain Results Relating to Quaternions," *Philosophical Magazine*, Vol. 26, No. 1, 1845, pp. 141–145.

[72]Hamilton, W. R., "On Quaternions: Or a New System of Imaginaries in Algebra," *Philosophical Magazine*, Vol. 25, No. 3, 1844, pp. 489–495.

[73]Hamilton, W. R., *Lectures on Quaternions*, Hodges and Smith, Dublin, 1853, pp. 1–736.

[74]Hamilton, W. R., *Elements of Quaternions*, Longmans, Green and Co., London, 1866, pp. 1–762.

[75]Graves, R. P., *Life of Sir William Rowan Hamilton*, Arno, New York, 1975, Vol. 1, pp. 1–689, Vol. 2, pp. 1–719, Vol. 3, pp. 1–673.

[76]Hankins, T. L., *Sir William Rowan Hamilton*, Johns Hopkins Univ. Press, Baltimore, MD, 1980, pp. 1–474.

[77]O'Donnell, S., *William Rowan Hamilton, Portrait of a Prodigy*, Boole Press, Dublin, 1983, pp. 1–224.

[78]Altmann, S. L., *Rotations, Quaternions, and Double Groups*, Oxford Univ. Press, Oxford, England, U.K., 1986, pp. 1–317.

[79]Altmann, S. L., "Hamilton, Rodrigues, and the Quaternion Scandal," *Mathematics Magazine*, Vol. 62, No. 5, 1989, pp. 291–307.

[80]Cheng, H., and Gupta, K. C., "An Historical Note on Finite Rotations," *Journal of Applied Mathematics*, Vol. 56, No. 2, 1989, pp. 139–145.

[81]Euler, L., "Du Mouvement de Rotation des Corps Solides Autour d'un Axe Variable," *Memoires de l'académiei des sciences de Berlin*, Vol. 14, 1758, pp. 154–193.

[82]Euler, L., "De Motu Corporum Circa Punctum Fixum Mobilium," *Commentatio 825 Indicis Enestroemiani, Opera Posthuma*, Vol. 2, 1862, pp. 43–62.

[83]Roberson, R. E., "Kinematic Equations for Bodies Whose Rotation is Described by the Euler–Rodrigues Parameters," *AIAA Journal*, Vol. 6, No. 5, 1968, pp. 916, 917.

[84]Jacobi, D. G. J., "Bemerkungen zu einer Abhandlung Eulers über die Orthogonale Substitution," *C. G. J. Jacobis Gesammelte Werke*, Vol. 3, Chelsea, New York, 1969, pp. 601–609.

[85]Grassman, H. G., *Die Lineare Ausdehnungslehre, Ein Neuer Zweig der Mathematik Dargestellt und durch Anwendungen auf die übrigen Zweige der Mathematik, wie auch auf die Statik, Mechanik, die Lehre vom Magnetismus und die Krystallonomie erläutert*, O. Wigand, Leipzig, Germany, 1878, pp. 1–301.

[86]Grassman, H. G., *Die Ausdehnungslehre: vollständig und in strenger Form bearbeitet*, T. C. F. Enslin, Berlin, 1862, pp. 1–388.

[87]Gibbs, J. W., *Vector Analysis*, edited by E. B. Wilson, Dover, New York, 1901, pp. 1–436.

[88]Heaviside, O., *Electromagnetic Theory*, Vol. 1, Dover, New York, 1950, pp. 1–386.

[89]Boyer, C. B., and Merzbach, U. C., *A History of Mathematics*, 2nd ed., Wiley, New York, 1989, pp. 656–762.

[90]Gray, J. J., "Olinde Rodrigues' Paper of 1840 on Transformation Groups," *Archive for the History of Exact Sciences*, Vol. 21, No. 4, 1980, pp. 375–385.

[91]Cartan, E., *Leçons sur la Theorie des Spineurs. I. Les Spineurs de l'espace a Trois Dimensions*, Hermann, Paris, 1938, pp. 57–63.

[92]Temple, G., *Cartesian Tensors. An Introduction*, Methuen, London, 1960, pp. 68–92.

[93]Booth, A. J., *Saint-Simon and Saint-Simonism, A Chapter in the History of Socialism in France*, Longmans, Green, Reader and Dyer, London, 1871, pp. 1–262.

[94]Wilson, E., *To the Finland Station. A Study in the Writing and Acting of History*, Secker and Warburg, London, 1941, pp. 100–509.

[95]Kline, M., *Mathematical Thoughts from Ancient to Modern Times*, Oxford Univ. Press, Oxford, England, U.K., 1972, pp. 1–1238.

[96]Van Der Waerden, B. L., *A History of Algebra*, Springer-Verlag, Heidelberg, Germany, 1985, pp. 1–271.

[97]Crowe, M. J., *A History of Vector Analysis*, Dover, New York, 1967, pp. 1–270.

[98]McDuffee, C. C., *The Theory of Matrices*, Chelsea, New York, 1946, pp. 1–110.

[99]Shuster, M. D., and Natanson, G. A., "Quaternion Computation from a Geometric Point of View," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 545–556.

[100]Vathsal, S., "Derivation of the Relative Quaternion Differential Equations," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 5, 1991, pp. 1061–1064.

[101]Stuelpnagel, J., "On the Parameterization of the Three-Dimensional Rotation Group," *SIAM Review*, Vol. 6, No. 4, 1964, pp. 422–430.

[102]Klumpp, A. R., "Singularity-free Extraction of a Quaternion from a Direction-Cosine Matrix," *Journal of Spacecraft and Rockets*, Vol. 13, No. 12, 1976, pp. 754, 755.

[103]Spurrier, R. A., "Comment on 'Singularity-Free Extraction of a Quaternion from a Direction-Cosine Matrix," *Journal of Spacecraft and Rockets*, Vol. 15, No. 4, 1978, pp. 255, 256.

[104]Shepperd, S. W., "Quaternion from Rotation Matrix," *Journal of Guidance and Control*, Vol. 1, No. 3, 1978, pp. 223, 224.

[105]Jiang, Y. F., and Lin, Y. P., "Error Analysis of Quaternion Transformation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27,

No. 4, 1991, pp. 634–638.

[106]Aramanovitch, L., "Spacecraft Orientation Based on Space Object Observations by Means of Quaternion Algebra," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 4, 1995, pp. 859–866.

[107]Kuipers, J. B., *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*, Princeton Univ. Press, Princeton, NJ, 1999, pp. 155–168.

[108]Mayo, R. A., "Relative Quaternion State Transition Relation," *Journal of Guidance and Control*, Vol. 2, No. 1, 1979, pp. 44–48.

[109]Stevens, B. L., and Lewis, F. L., *Aircraft Control and Simulation*, Wiley, New York, 1992, pp. 39–44.

[110]McFarland, R. E., "A Standard Kinematic Model for Flight Simulation at NASA-Ames," NASA CR-2497, Jan. 1975.

[111]Jenkins, P. N., "Missile Dynamics Equations for Guidance and Control Modeling and Analysis," Rept. RG-84-17, Redstone Arsenal, AL, April 1984.

[112]Wilcox, J. C., "A New Algorithm for Strapdown Inertial Navigators," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 3, No. 5, 1967, pp. 796–802.

[113]Mckern, R. A., "A Study of Transformation Algorithms for Use in a Digital Computer," M.S. Thesis T-493, Massachusetts Inst. of Technology, Cambridge, MA, 1968.

[114]Bortz, J. E., "A New Mathematical Formulation for Strapdown Inertial Navigation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 7, No. 1, 1971, pp. 61–66.

[115]Mortensen, R. E., "Strapdown Guidance Error Analysis," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 10, No. 5, 1974, pp. 451–457.

[116]Friedland, B., "Analysis of Strapdown Navigation Using Quaternions," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 14, No. 5, 1989, pp. 764–768.

[117]Lovren, N., and Pieper, J. K., "Error Analysis of Direction Cosines and Quaternion Parameter Techniques for Aircraft Attitude Determination," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 3, 1998, pp. 983–989.

[118]Lefferts, E. J., Markley, F. L., and Shuster, M. D., "Kalman Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, 1982, pp. 417–429.

[119]Bar-Itzhack, I. Y., and Oshman, Y., "Attitude Determination from Vector Observations: Quaternion Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 21, No. 1, 1985, pp. 128–136.

[120]Shuster, M. D., "Quaternion on the Kalman Filter," *Advances in the Astronautical Sciences*, Vol. 85, No. 1, 1993, pp. 16–19.

[121]Creamer, G., DeLaHunt, P., and Gates, S., "Attitude Determination and Control of Clementine During Lunar Mapping," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 3, 1996, pp. 505–511.

[122]Ickes, B. F., "A New Method for Performing Digital Control System Attitude Computations Using Quaternions," *AIAA Journal*, Vol. 8, No. 1, 1970, pp. 13–17.

[123]Grubin, C., "Attitude Determination for a Strapdown Inertial System Using the Euler-Axis/Angle and Quaternion Parameters," AIAA Paper 73-900, 1973.

[124]Morton, H. S., Jr., Junkins, J. L., and Blanton, J. N., "Analytical Solutions for Euler Parameters," *Celestial Mechanics*, Vol. 10, No. 3, 1974, pp. 287–301.

[125]Morton, H. S., Jr., "Hamiltonian and Lagrangian Formulations of Rigid Body Rotational Dynamics Based on the Euler Parameters," *Journal of Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 569–591.

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

[127]Vadali, S. R., Kraige, L. G., and Junkins, J. L., "New Results on the Optimal Spacecraft Attitude Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 7, No. 3, 1984, pp. 378–380.

[128]Vathsal, S., "Optimal Control of Quaternion Propagation Errors in Spacecraft Navigation," *Journal of Guidance, Control, and Dynamics*, Vol. 9, No. 3, 1986, pp. 382–385.

[129]Wie, B., Weiss, H., and Arapostathis, A., "Quaternion Feedback Regulator for Spacecraft Eigenaxis Rotations," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 3, 1989, pp. 375–380.

[130]Chatterji, G. B., and Tahk, M., "Quaternion Formulation for Boost Phase Attitude Estimation, Guidance and Control of Exoatmospheric Interceptors," *Proceedings of the 1989 American Control Conference*, Inst. of Electrical and Electronics Engineers, New York, 1989, pp. 1561–1566.

[131]Katz, A., "Special Rotation Vectors: A Means for Transmitting Quaternions in Three Components," *Journal of Aircraft*, Vol. 30, No. 1, 1993, pp. 148–150.

[132]Lo, S. C., and Y. P. Chen, "Smooth Sliding Mode Control for Spacecraft Attitude Tracking Maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 6, 1995, pp. 1345–1349.

[133]Joshi, S. M., Kelkar, A. G., and Went, J. T. Y., "Robust Attitude Stabilization of Spacecraft Using Nonlinear Quaternion Feedback," *IEEE Transactions on Automatic Control*, Vol. 40, No. 10, 1995, pp. 1800–1803.

[134]Crassidis, J. L., and Markely, F. L., "Sliding Mode Control Using Modified Rodrigues Parameters," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 6, 1996, pp. 1381–1383.

[135]Chung, D., Lee, J. G., and Park, C. G., "Strapdown INS Error Model for Multiposition Alignment," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 32, No. 4, 1996, pp. 1362–1366.

[136]Bar-Itzhack, I. Y., "REQUEST: A Recursive QUEST Algorithm for Sequential Attitude Determination," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 5, 1996, pp. 1034–1038.

[137]Kelkar, A. G., and Joshi, S. M., "Global Stabilization of Flexible Multibody Spacecraft Using Quaternion-Based Nonlinear Control Law," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 5, 1996, pp. 1186–1188.

[138]Lee, K. L., Lee, J. G., Roh, Y. K., and Park, C. G., "Modeling Quaternion Errors in SDINS: Computer Frame Approach," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 1, 1998, pp. 289–299.

[139]Reynolds, R. G., "Quaternion Parameterization and a Simple Algorithm for Global Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 4, 1998, pp. 669–671.

[140]Gupta, S., "Linear Quaternion Equations with Application to Spacecraft Attitude Propagation," *IEEE Aerospace Applications Conference Proceedings*, Vol. 1, Inst. of Electrical and Electronics Engineers, New York, 1998, pp. 69–76.

[141]Yu, M. J., Lee, J. G., and Park, H. W., "Comparison of SDINS Inflight Alignment Using Equivalent Error Models," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 35, No. 3, 1999, pp. 1046–1054.

[142]Fallon, L., "Quaternions," *Spacecraft Attitude Determination and Control*, edited by J. R. Wertz, Kluwer Academic, Dordrecht, The Netherlands 1978, pp. 758, 759.

[143]Jones, D. W., "Quaternions Quickly Transform Coordinates Without Error Buildup," *Electrical Design News*, Vol. 40, No. 5, 1995, pp. 95–100.

[144]Bar-Itzhack, I. Y., and Idan, M., "Recursive Attitude Determination from Vector Observations: Euler Angle Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 2, 1987, pp. 152–157.

[145]Dvornychenko, V. N., "The Number of Multiplications Required to Chain Coordinate Transformations," *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 1, 1985, pp. 157–159.

[146]Winograd, S., "On the Number of Multiplications Necessary to Compute Certain Functions," *Communications on Pure and Applied Mathematics*, Vol. 23, No. 2, 1970, pp. 165–179.

[147]Winograd, S., "Some Bilinear Forms Whose Multiplicative Complexity Depends on the Field of Constants," *Mathematical Systems Theory*, Vol. 10, No. 2, 1977, pp. 169–180.

[148]Wehage, R. A., "Quaternions and Euler Parameters—A Brief Exposition," *Computer-Aided Analysis and Optimization of Mechanical System Dynamics*, edited by E. J. Haug, NATO ASI Series, Vol. F9, Springer-Verlag, Heidelberg, Germany, 1984, pp. 147–180.

[149]Bar-Itzhack, I. Y., "Iterative Computation of Initial Quaternion," *Journal of Spacecraft and Rockets*, Vol. 7, No. 3, 1970, pp. 367–369.

[150]Grubin, C., "Derivation of the Quaternion Scheme via the Euler Axis and Angle," *Journal of Spacecraft and Rockets*, Vol. 7, No. 10, 1970, pp. 1261–1263.

[151]Klumpp, A. R., "Reply to Comment on 'Singularity Extraction of a Quaternion from a Direction-Cosine Matrix," *Journal of Spacecraft and Rockets*, Vol. 15, No. 4, 1978, pp. 256–256.

[152]Grubin, C., "Quaternion Singularity Revisited," *Journal of Guidance and Control*, Vol. 2, No. 3, 1979, pp. 255, 256.

[153]Salguero, D., "Trajectory Analysis and Optimization Software," Rept. SAND99-0811, Sandia National Labs., Albuquerque, NM, May 1999.

[154]Prewitt, N. C., Belk, D. M., and Maple, R. C., "Multiple Body Trajectory Calculations Using the Beggar Code," AIAA Paper 96-3384, July 1996.

[155]Thome, W. J., "Quaternion Utilization in a Missile Model," U.S. Air Force Inst. of Technology, Rept. AFIT-GST-MA-80M-3, Wright–Patterson AFB, OH, March 1980.

[156]Kendra, M. J., and Bonito, N. A., "Coordinate System Conversion Software User's Guide for Attitude, Quaternion, M50 and Related Applications," Rept. GL-TR-90-0327, Geophysics Lab., Air Force Systems Command, United States Air Force, Hanscom Air Force Base, MA, Nov. 1990.

[157]Hankey, W. L., Miller, L. E., and Scherr, S. J., "Use of Quaternions in Flight Mechanics," U.S. Air Force Wright Aeronautical Labs., Rept. AFWAL-TR-84-3045, Wright–Patterson AFB, OH, March 1984.

[158]"Guide for the Verification and Validation of Computational Fluid Dynamics Simulations," AIAA, G-077-1998, Jan. 1998.

[159]Roache, P. J., *Verification and Validation in Computational Sciences*

and Engineering, Hermosa, Albuquerque, NM, 1998, pp. 1–648.

[160]Gerald, C. F., and Wheatley, P. O., *Applied Numerical Analysis*, 6th ed., Addison Wesley Longman, Reading, MA, 1999, pp. 451–477.

[161]Hoffman, J. D., *Numerical Methods for Engineers and Scientist*, McGraw–Hill, New York, 1992, pp. 225–266.

[162]Fang, A. C., and Zimmerman, B. G., "Digital Simulation of Rotational Kinematics," NASA TN D-5302, Oct. 1969.

[163]Bar-Itzhack, I. Y., "Optimum Normalization of a Computed Quaternion of Rotation," *IEEE Transactions of Aerospace and Electronic Systems*, Vol. 7, No. 2, 1971, pp. 401, 402.

[164]Katz, A., *Computational Rigid Vehicle Dynamics*, Krieger, Malabar, FL, 1997, pp. 29–38.

[165]"Atmospheric and Space Flight Vehicle Coordinate Systems," American National Standards Inst./AIAA, Rept. R-004-1992, Feb. 1992.

[166]Pope, D. A., "An Exponential Method of Numerical Integration of Ordinary Differential Equations," *Communications of the ACM*, Vol. 6, No. 8, 1963, pp. 491–493.

[167]Barker, L. E., Bowles, R. L., and Williams, L. H., "Development and Application of a Local Linearization Algorithm for the Integration of Quaternion Rate Equations in Real-Time Flight Simulation Problems," NASA TN D-7347, Dec. 1973.

[168]Yen, K., and Cook, G., "Improved Local Linearization Algorithm for Solving the Quaternion Equations," *Journal of Guidance and Control*, Vol. 3, No. 5, 1980, pp. 468–471.