Zupan, E., Saje, M. 2011. Integrating rotation from angular velocity. *Advances in Engineering Software* 42: 723-733, DOI: 10.1016/j.advengsoft.2011.05.010.

Integrating rotation from angular velocity

Eva Zupan [1] **and Miran Saje** [2]

University of Ljubljana, Faculty of Civil and Geodetic Engineering, Jamova 2, SI-1115 Ljubljana, Slovenia

Abstract

The integration of the rotation from a given angular velocity is often required in practice. The present paper explores how the choice of the parametrization of rotation, when employed in conjuction with different numerical time-integration schemes, effects the accuracy and the computational efficiency. Three rotation parametrizations – the rotational vector, the Argyris tangential vector and the rotational quaternion – are combined with three different numerical time-integration schemes, including classical explicit Runge-Kutta method and the novel midpoint rule proposed here. The key result of the study is the assessment of the integration errors of various parametrization–integration method combinations. In order to assess the errors, we choose a time-dependent function corresponding to a rotational vector, and derive the related exact time-dependent angular velocity. This is then employed in the numerical solution as the data. The resulting numerically integrated approximate rotations are compared with the analytical solution. A novel global solution error norm for discrete solutions given by a set of values at chosen time-points is employed. Several characteristic angular velocity functions, resulting in small, finite and fast oscillating rotations are studied.

1 Introduction

The integration of rotation from the given angular velocity is often required in practice. The application areas include such diverse fields as navigation in aerospace technologies, robotics, animating rotations in computer graphics,

[1] E-mail address: eva.zupan@fgg.uni-lj.si
[2] Corresponding author. E-mail address: miran.saje@fgg.uni-lj.si

describing curvatures with rotations in structural mechanics, and classical dynamics of rigid bodies. For example, in a strapdown inertial navigation system [1,6,15,26,27], the gyroscopes measure the angular velocity of the space vehicle. In order to obtain the global position of the vehicle, the coordinate transformation matrix between the body-fixed and the spatial coordinate systems is needed. This requires the real-time integration of the angular velocity-rotation differential equation. An accurate, yet computationally economical solution is thus vital in designing a strapdown navigation system. A conceptually similar application is also typical in a real-time optical tracking of human body motion employed in robotics and computer graphics [11,24,32]. A further example can be taken from structural mechanics, where in a curvature-based finite-element formulation, one needs to integrate the rotation from the given curvature field [38,39].

The choice of finite rotation parameters is often discussed in literature, see e.g. [2,3,18]. To the best knowlege of the authors, none of them, however, has analysed the combined influence of different rotation parameters and time integrators.

As the differential equation under consideration is strongly non-linear, an analytical solution is not possible for angular velocities other than those being constant in time, and hence a numerical method has to be applied. The primary objective of the present paper is to study how the choice of the parametrization of rotation, when employed in conjunction with different numerical time-integration schemes, effects the accuracy and the computational efficiency in numerical integration of rotations from the given time-dependent angular velocity. We compare three rotation parametrizations, i.e. the rotational vector, see, e.g. [3,7,22,34,38], the tangential vector introduced by Argyris [2] and the rotational quaternion [40], and three different numerical time-integration schemes, namely, two alternative midpoint rules (one from [33], while the other appears to be new) and the classical fifth-order Runge-Kutta method.

The key result is the assessment of the integration errors of the above listed parametrization–integration method combinations. In analyzing the errors, we define the time-dependent function corresponding to a rotational vector, and determine the related exact time-dependent angular velocity function. This one is then employed as the angular velocity in the numerical solution. The resulting numerically integrated approximate rotations are finally compared with the analytical solution. Three characteristic types of angular velocity functions are studied: (i) the angular velocity, representing small rotations in a relatively long time interval; (ii) the angular velocity, representing moderate, yet well oscillating rotations; and (iii) the angular velocity is such that the rotations are both large and highly oscillating.

In the rotation matrix error analysis, the drift, skew and scale errors are often

employed as the local error measures [23,26]. Although these error measures carry on some more physical or geometrical meaning, the large number of scalar components that need to be compared makes these measures less appropriate for assessing the global error. Therefore we choose the L^2 function norms of the rotational vector components as an appropriate mathematical norm that measures the solution deviations globally.

2 Rotation and angular velocity

From the mathematical point of view the rotation is a unitary operator $R : I\!R^3 \to I\!R^3$ on the (Hilbert) vector space $I\!R^3$ whose action $R : \vec{a} \mapsto R\vec{a}$ can be written as

$$\vec{a}_1 = R\vec{a} \tag{1}$$

and $RR^* = R^*R = I$, for R^* being the Hilbert space adjoint of R [31, p. 312]. The properties of the rotation operator are well described in, e.g. [2,3,12].

If rotation R changes with time t, we denote it by $R(t)$. The compositum of the derivative of rotation with respect to time t and the adjoint operator R^* defines the skew-hermitian *angular velocity operator*

$$\Omega = \dot{R}R^* = -\Omega^*. \tag{2}$$

This operator plays a fundamental role in dynamics of deformable and rigid bodies. When $\Omega(t)$ is a given function of time, Eq. (2) represents the differential equation for the rotation in terms of the angular velocity. Our objective here is to integrate Eq. (2) for $R(t)$ numerically.

Remark 1 *The same differential equation is encountered not only in dynamics, but also in statics of spatial beams. There, the role of Ω is played by the skew-hermitian curvature operator $\mathcal{K}$ dependent on the arc-length of the beam, s. When the curvature is the basic unknown of the beam formulation, as in Zupan and Saje [38,39], rotation $R(s)$ has to be determined from $\mathcal{K} = R'R^*$ with $\mathcal{K}(s)$ being a given function of s.*

It is convenient to introduce a spatial coordinate system consisting of a fixed but otherwise arbitrary *reference point* $\mathcal{O}$, and a *fixed orthonormal basis* $\mathcal{B}_g = \left(\vec{g}_1, \vec{g}_2, \vec{g}_3\right)$. Then the operators R, R^* and Ω can be represented by the components of the rotation matrix $\boldsymbol{R}$, its transposed matrix $\boldsymbol{R}^T$ and the angular velocity matrix $\boldsymbol{\Omega}_g$, respectively. The rotation matrix is a proper orthogonal matrix $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$ with $\det \boldsymbol{R} = +1$, and $\boldsymbol{\Omega}_g$ is a skew-symmetric

matrix

$$\boldsymbol{\Omega}_g = \boldsymbol{S}\left(\boldsymbol{\omega}_g\right) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} = -\boldsymbol{\Omega}_g^T. \tag{3}$$

The components of the matrix $\boldsymbol{\Omega}_g$ are expressed with the components of the *axial vector* $\vec{\omega}$, $\boldsymbol{\omega}_g = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T$, of the operator Ω. In kinematics the axial vector $\vec{\omega}$ is known as the *angular velocity vector*.

Rotation $R\left(t\right)$ maps the fixed basis into another orthonormal vector basis $\mathcal{B}_G = \left(\vec{G}_1\left(t\right), \vec{G}_2\left(t\right), \vec{G}_3\left(t\right)\right)$, usually attached to the centroid of a moving body, and is therefore called the *body basis*: $\vec{G}_i\left(t\right) = R\left(t\right)\vec{g}_i$.

The angular velocity matrix can be expressed in either of the two bases as

$$\boldsymbol{\Omega}_g = \dot{\boldsymbol{R}}_g \boldsymbol{R}^T, \tag{4}$$

$$\boldsymbol{\Omega}_G = \boldsymbol{R}^T \dot{\boldsymbol{R}}_G. \tag{5}$$

Here, the indices $\left[...\right]_g$ and $\left[...\right]_G$ stand for the fixed and the body basis, respectively. Note that the components of the matrix $\boldsymbol{R}$ are identical in the two frames, i.e. $\boldsymbol{R}_g = \boldsymbol{R}_G$. That is why the indices are omitted.

3 Parametrization of rotations

The rotation is fully described by the 3×3 proper orthogonal matrix $\boldsymbol{R}$. Such an orthogonal matrix is uniquely represented by three independent scalar parameters, yet it is not singularity free nor bijective. There exists several parametrizations based on three or more scalars. Here we discuss only the parametrizations by the rotational vector, the tangential vector and the rotational quaternion. The common ground of these three parametrizations is the notion of the rotation as a movement of a rigid body by an angle ϑ about the axis of rotation, specified by a unit vector $\vec{n}$.

3.1 Rotational vector

The rotational vector, $\vec{\vartheta}$, is widely used in finite-element formulations of geometrically exact three-dimensional beam and shell theories ([9,20,38,34], and

many others). The length of the rotational vector, $\vartheta = \left|\vec{\vartheta}\right|$, equals to the angle of rotation about the unit vector, $\vec{n}$, establishing its direction, $\vec{\vartheta} = \vartheta\vec{n}$. Let $\boldsymbol{\vartheta}_g$ denote the one-column matrix of $\vec{\vartheta}$ with respect to the fixed basis. When the rotation matrix is represented by the rotational vector, it takes the form

$$\boldsymbol{R}\left(\boldsymbol{\vartheta}_g\right) = \boldsymbol{I} + \frac{\sin\vartheta}{\vartheta}\boldsymbol{\Theta}_g + \frac{1-\cos\vartheta}{\vartheta^2}\,\boldsymbol{\Theta}_g^2, \tag{6}$$

attributed to Rodrigues. Skew-symmetric matrix $\boldsymbol{\Theta}_g$ is composed from the components of the rotational vector as $\boldsymbol{S}\left(\boldsymbol{\vartheta}_g\right)$ described in Eq. (3). Hence, $\boldsymbol{\vartheta}_g$ is the axial vector of $\boldsymbol{\Theta}_g$.

When $\boldsymbol{R}\left(\boldsymbol{\vartheta}_g\right)$ is inserted into Eq. (4), we obtain Eq. (2) expressed in terms of the rotational vector with respect to the fixed basis:

$$\dot{\boldsymbol{\vartheta}}_g = \boldsymbol{T}^{-1}\left(\boldsymbol{\vartheta}_g\right)\boldsymbol{\omega}_g, \tag{7}$$

where

$$\boldsymbol{T}^{-1}\left(\boldsymbol{\vartheta}_g\right) = \boldsymbol{I} - \frac{1}{2}\,\boldsymbol{\Theta}_g + \frac{1}{\vartheta^2}\left(1 - \frac{\vartheta}{2\tan\frac{\vartheta}{2}}\right)\boldsymbol{\Theta}_g^2 \tag{8}$$

is the inverse of the matrix operator $\boldsymbol{T}_g = \boldsymbol{T}\left(\boldsymbol{\vartheta}_g\right) = \boldsymbol{I} + \frac{1-\cos\vartheta}{\vartheta^2}\,\boldsymbol{\Theta}_g + \frac{\vartheta-\sin\vartheta}{\vartheta^3}\,\boldsymbol{\Theta}_g^2$. The differential equation (7) has a singularity problems at $\vartheta = 2k\pi$, $k \in \mathbb{Z}\backslash\{0\}$, because $\frac{2k\pi}{\tan k\pi}$ and $\frac{1}{2k\pi\cdot\tan k\pi}$ tend to infinity. At these critical points the solution of the equation still exists as the modified form of the same equation, $\boldsymbol{\omega}_g = \boldsymbol{T}\left(\boldsymbol{\vartheta}_g\right)\dot{\boldsymbol{\vartheta}}_g$, has only one removable singularity at $\vartheta = 0$, due to $\lim\limits_{\vartheta\to 0}\frac{1-\cos\vartheta}{\vartheta^2} = \frac{1}{2}$ and $\lim\limits_{\vartheta\to 0}\frac{\vartheta-\sin\vartheta}{\vartheta^3} = \frac{1}{6}$. We can still expect numerical problems whenever angle of rotation ϑ approaches the values $2k\pi$, $k \in \mathbb{Z}\backslash\{0\}$.

3.2 Tangential vector

Parametrization of rotation by the tangential vector $\vec{\psi} = \tan\frac{\vartheta}{2}\vec{n}$ was suggested by Argyris [2]. The vector $\vec{\psi}$ has the norm $\left|\vec{\psi}\right| = \psi = \tan\frac{\vartheta}{2}$ and the direction $\vec{n}$. Note that $\boldsymbol{\psi}$ tends to infinity for $\vartheta = \pi + 2k\pi$, $k \in \mathbb{Z}$. Therefore the tangential vector can be used for parametrization of rotation only for the angles ϑ between two consecutive values $\pi + 2k\pi$ and $\pi + 2\left(k+1\right)\pi$. Let $\boldsymbol{\psi}_g$ denote the one-column matrix representation of $\vec{\psi}$ with respect to the fixed basis. The related rotation matrix is derived by the help of Eq. (6) and reads

$$\boldsymbol{A}\left(\boldsymbol{\psi}_g\right) = \boldsymbol{I} + \frac{2}{1+\psi^2}\left(\boldsymbol{\Psi}_g + \boldsymbol{\Psi}_g^2\right), \tag{9}$$

Table 1
Formulae for calculating $\boldsymbol{\psi}$, $\dot{\boldsymbol{\psi}}$, $\widehat{\boldsymbol{q}}$ and $\dot{\widehat{\boldsymbol{q}}}$ from $\boldsymbol{\vartheta}$.

$\vartheta = \sqrt{\vartheta_1^2 + \vartheta_2^2 + \vartheta_3^2}$	$\boldsymbol{n} = \frac{\boldsymbol{\vartheta}}{\vartheta}$
$\dot{\vartheta} = \frac{\dot{\boldsymbol{\vartheta}} \cdot \boldsymbol{\vartheta}}{\vartheta}$	$\dot{\boldsymbol{n}} = \frac{\dot{\boldsymbol{\vartheta}}}{\vartheta} - \frac{\boldsymbol{\vartheta} \dot{\vartheta}}{\vartheta^2}$
$\boldsymbol{\psi} = \tan \frac{\vartheta}{2} \boldsymbol{n}$	$\dot{\boldsymbol{\psi}} = \frac{\dot{\vartheta}}{\cos^2 \frac{\vartheta}{2}} \boldsymbol{n} + \tan \frac{\vartheta}{2} \dot{\boldsymbol{n}}$
$\widehat{\boldsymbol{q}} = \begin{bmatrix} \cos \frac{\vartheta}{2} \\ \boldsymbol{n} \sin \frac{\vartheta}{2} \end{bmatrix}$	$\dot{\widehat{\boldsymbol{q}}} = \begin{bmatrix} -\frac{\dot{\vartheta}}{2} \sin \frac{\vartheta}{2} \\ \dot{\boldsymbol{n}} \sin \frac{\vartheta}{2} + \boldsymbol{n} \frac{\dot{\vartheta}}{2} \cos \frac{\vartheta}{2} \end{bmatrix}$

where $\boldsymbol{\Psi}_g = \boldsymbol{S}\left(\boldsymbol{\psi}_g\right)$ is a skew-symmetric matrix whose axial vector is $\boldsymbol{\psi}_g$. It is composed from the components of the axial vector as described in Eq. (3).

Inserting Eq. (9) and its time derivative into Eq. (4) gives the differential equation (2) in terms of the tangential vector as

$$\dot{\boldsymbol{\psi}}_g = \boldsymbol{U}^{-1}\left(\boldsymbol{\psi}_g\right) \boldsymbol{\omega}_g, \tag{10}$$

$$\boldsymbol{U}^{-1}\left(\boldsymbol{\psi}_g\right) = \frac{1+\psi^2}{2} \boldsymbol{I} + \frac{1}{2}\left(-\boldsymbol{\Psi}_g + \boldsymbol{\Psi}_g^2\right). \tag{11}$$

As described above the solution $\boldsymbol{\psi}$ can be computed only for angles in range $\vartheta \in (\pi + 2k\pi, \pi + 2(k+1)\pi)$, for some $k \in \mathbb{Z}$. Note that the computation of the matrices $\boldsymbol{A}_g$ and $\boldsymbol{U}_g^{-1}$ does not require evaluating trigonometric functions, which somewhat speeds up the computations.

In the numerical implementation, we need formulae to compute $\boldsymbol{\psi}_g$ and $\dot{\boldsymbol{\psi}}_g$ from a given rotational vector. They are displayed in Table 1.

3.3 Rotational quaternion

Quaternions have been recently often discussed in relation with the structural mechanics and classical dynamics of rigid and deformable bodies, see e.g. [17,27,29,32,40]. Therefore only fundamentals need to be presented here. Systematic descriptions can be found in mathematical textbooks [28] or [37]. Quaternion can be defined as a formal sum of a scalar and a vector: $\widehat{a} = a_0 + \vec{a}$. For arbitrary quaternions $\widehat{a} = a_0 + \vec{a}$ and $\widehat{b} = b_0 + \vec{b}$ ($a_0, b_0 \in I\!R$, $\vec{a}, \vec{b} \in I\!R^3$), and a scalar $\lambda \in I\!R$, the following operations are defined :

i. sum: $\widehat{a} + \widehat{b} := (a_0 + b_0) + \left(\vec{a} + \vec{b}\right)$,
ii. multiplication by a scalar: $\lambda \widehat{a} := \lambda a_0 + \lambda \vec{a}$,
iii. quaternion multiplication: $\widehat{a} \circ \widehat{b} := \left(a_0 b_0 - \vec{a} \cdot \vec{b}\right) + \left(b_0 \vec{a} + a_0 \vec{b} + \vec{a} \times \vec{b}\right)$.

These operations make the set of quaternions $I\!H$ an algebra. The quaternion multiplication (iii) is not a commutative operation as it contains the cross-vector product. Note that set $I\!H$ including only the first two of the above operations is also a four-dimensional linear space over $I\!R$ and is thus isomorphic to a linear space $I\!R^4$.

We furthermore define the quaternion $\widehat{a}^* = a_0 - \overrightarrow{a}$ conjugated to $\widehat{a}$, with a property $\left(\widehat{a} \circ \widehat{b}\right)^* = \widehat{b}^* \circ \widehat{a}^*$. The *quaternion norm* is defined as $|\widehat{a}| = \sqrt{\widehat{a} \circ \widehat{a}^*} = \sqrt{a_0^2 + \left|\overrightarrow{a}\right|^2}$. If, for a quaternion $\widehat{a}$, it holds $\widehat{a} = -\widehat{a}^*$, then it must be of the form $\widehat{a} = 0 + \overrightarrow{a}$, and is called a *pure quaternion*. Such a quaternion can be uniquely identified with its vector part and the linear spaces of pure quaternions and $I\!R^3$ are isomorphic. Any quaternion $\widehat{a} = a_0 + \overrightarrow{a}$ can also be written in an alternative *polar form* $\widehat{a} = |\widehat{a}| \left(\cos\theta + \overrightarrow{a}_n \sin\theta\right)$, where $\overrightarrow{a}_n = \frac{\overrightarrow{a}}{\left|\overrightarrow{a}\right|}$ is a unit vector; angle θ can be extracted from the quaternion using $\cos\theta = \frac{a_0}{|\widehat{a}|}$ and $\sin\theta = \frac{\left|\overrightarrow{a}\right|}{|\widehat{a}|}$. In order to present quaternions as members of the linear space in the componential form, we have to introduce a basis. Instead of choosing an arbitrary orthonormal basis of the four dimensional vector space, we rather expand the previously chosen bases of the three-dimensional linear space; elements of bases $\mathcal{B}_g$ and $\mathcal{B}_G$ are extended into pure quaternions $\widehat{g}_i = 0 + \overrightarrow{g}_i$, and $\widehat{G}_i = 0 + \overrightarrow{G}_i$ and the fourth base quaternion is added as $\widehat{g}_0 = \widehat{G}_0 = 1 + \overrightarrow{0}$. This construction clearly results in an orthonormal basis of the 4-dimensional quaternion space. Then any quaternion can be represented with respect to either the fixed or the body basis, represented as a one-column matrices $\widehat{\boldsymbol{a}}_g = \begin{bmatrix} a_{g0} & a_{g1} & a_{g2} & a_{g3} \end{bmatrix}^T$ and $\widehat{\boldsymbol{a}}_G = \begin{bmatrix} a_{G0} & a_{G1} & a_{G2} & a_{G3} \end{bmatrix}^T$, respectively.

Now we discuss a unit quaternion. Assume that $\widehat{q} = q_0 + \overrightarrow{q}$ is a unit quaternion, $|\widehat{q}| = 1$, with the polar representation being $\widehat{q} = \cos\vartheta + \overrightarrow{n}\sin\vartheta$. Then we introduce two linear transformations, defined alternatively by the left and the right multiplication:

$$\phi_L\left(\widehat{q}\right) : \widehat{x} \longmapsto \widehat{q} \circ \widehat{x} \tag{12}$$

$$\phi_R\left(\widehat{q}\right) : \widehat{x} \longmapsto \widehat{x} \circ \widehat{q}. \tag{13}$$

The transformations $\phi_L(\widehat{q})$ and $\phi_R(\widehat{q})$ are represented in the matrix notation spanned by the fixed basis:

$$\boldsymbol{\phi}_L(\widehat{\boldsymbol{q}}) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \qquad \boldsymbol{\phi}_R(\widehat{\boldsymbol{q}}) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix}. \tag{14}$$

It is easy to show that the above matrices fulfil the following relations:

$$\det \boldsymbol{\phi}_L(\widehat{\boldsymbol{q}}) = 1 \qquad \boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})\,\boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})^T = \boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})^T\,\boldsymbol{\phi}_L(\widehat{\boldsymbol{q}}) = \boldsymbol{I} \tag{15}$$

$$\det \boldsymbol{\phi}_R(\widehat{\boldsymbol{q}}) = 1 \qquad \boldsymbol{\phi}_R(\widehat{\boldsymbol{q}})\,\boldsymbol{\phi}_R(\widehat{\boldsymbol{q}})^T = \boldsymbol{\phi}_R(\widehat{\boldsymbol{q}})^T\,\boldsymbol{\phi}_R(\widehat{\boldsymbol{q}}) = \boldsymbol{I}. \tag{16}$$

Any 4×4 matrix satisfying properties listed in Eqs. (15) or (16) is an element of the special orthogonal group $SO(4)$. Therefore the operators defined in Eqs. (12) and (13) are both rotations in $I\!R^4$ and both rotate by angle ϑ [37].

Rotating a pure quaternion by only $\boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})$ or $\boldsymbol{\phi}_R(\widehat{\boldsymbol{q}})$ does not result in a pure quaternion. A compositum does, however; i.e. if $\widehat{\boldsymbol{a}}$ is a pure quaternion, then

$$\widehat{\boldsymbol{a}}_1 = \boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})\,\boldsymbol{\phi}_R(\widehat{\boldsymbol{q}}^*)\,\widehat{\boldsymbol{a}} = \widehat{\boldsymbol{q}} \circ \widehat{\boldsymbol{a}} \circ \widehat{\boldsymbol{q}}^*$$

is also a pure quaternion. Because $\boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})$ and $\boldsymbol{\phi}_R(\widehat{\boldsymbol{q}}^*)$ each rotates by the same angle, their compositum, $\boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})\,\boldsymbol{\phi}_R(\widehat{\boldsymbol{q}}^*)$, results in the double angle rotation. Thus the subsequent application of the unit quaternion

$$\widehat{q} = \cos\frac{\vartheta}{2} + \vec{n}\sin\frac{\vartheta}{2} \tag{17}$$

on an arbitrary pure quaternion $\widehat{a} = 0 + \vec{a}$

$$\widehat{a}_1 = \widehat{q} \circ \widehat{a} \circ \widehat{q}^* \tag{18}$$

results in a pure quaternion $\widehat{a}_1 = 0 + \vec{a}_1$ rotated by the angle ϑ. $\widehat{q}$ in (17) is called the *rotational quaternion*, ϑ is the full angle of rotation and $\vec{n}$ is the unit vector specifying the axis of rotation.

The double quaternion product on $\widehat{a}$ in Eq. (18) can be replaced by the action of the matrix $\boldsymbol{Q}(\widehat{\boldsymbol{q}}) = \boldsymbol{\phi}_L(\widehat{\boldsymbol{q}})\,\boldsymbol{\phi}_R(\widehat{\boldsymbol{q}}^*)$ on $\widehat{a}$. In terms of parameter $\widehat{\boldsymbol{q}}$, this 4×4 rotation matrix reads

$$\boldsymbol{Q}(\widehat{\boldsymbol{q}}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2q_0^2 + 2q_1^2 - 1 & 2q_1q_2 - 2q_0q_3 & 2q_0q_2 + 2q_1q_3 \\ 0 & 2q_1q_2 + 2q_0q_3 & 2q_0^2 + 2q_2^2 - 1 & -2q_0q_1 + 2q_2q_3 \\ 0 & -2q_0q_2 + 2q_1q_3 & 2q_0q_1 + 2q_2q_3 & 2q_0^2 + 2q_3^2 - 1 \end{bmatrix}.$$

Note that its 3×3 submatrix is identical to the rotation matrix $\boldsymbol{R}$. For further details see [37] or [40].

In terms of quaternions, the rotation–angular velocity differential equation (2) takes the form [37]

$$\widehat{\boldsymbol{\omega}}_g = 2\dot{\widehat{\boldsymbol{q}}} \circ \widehat{\boldsymbol{q}}^*, \text{ or} \tag{19}$$

$$\widehat{\boldsymbol{\omega}}_G = 2\widehat{\boldsymbol{q}}^* \circ \dot{\widehat{\boldsymbol{q}}}. \tag{20}$$

Here the quaternion $\widehat{\boldsymbol{\omega}}$ is a pure quaternion $\widehat{\boldsymbol{\omega}} = 0 + \boldsymbol{\omega}$ and $\boldsymbol{\omega}$ is the angular velocity vector.

The formulae for the computation of $\widehat{\boldsymbol{q}}$ and $\dot{\widehat{\boldsymbol{q}}}$ from the given rotational vector are displayed in Table 1.

4 Solution algorithms

Eq. (2) represents the system of the first-order differential equations in time. In structural mechanics it is common to use a midpoint-rule type of time integrators or a version of the Newmark algorithms, see, e.g. [20,30,33,34,35], while the Runge-Kutta methods are used only occasionally [7,8].

In discussing the accuracy and efficiency of the time-integration scheme as a function of the rotation parametrization, we will limit our analysis to the following time-integration schemes:

(1) The midpoint rule proposed by Simo, Tarnow and Doblare [33], where the averaged rotation matrix in the time step is used; this scheme will be referred to as 'MP-R'.
(2) The midpoint rule with the averaged quaternion parameter, here marked by 'MP-q'.
(3) The explicit fifth-order Runge-Kutta method *ode45* as implemented in the commercial program *Matlab* [25].

The first and the third of the above integration methods are well known. The MP-R method was first presented by Simo et al. [33] and later on used in the analysis of the spatial beam structures by several other authors, e.g. [19,21]. The *ode45* method is one of the most often used explicit integration methods implemented in the Matlab environment.

We assume that the rotation matrix $\boldsymbol{R}(t_n)$ is parametrized by $\boldsymbol{\vartheta}(t_n)$ and given at time t_n. To obtain the rotation matrix at time t_{n+1}, we require that Eq. (5) is satisfied at *mid-time* $t_m = \frac{t_n+t_{n+1}}{2}$. Following Simo et al. [33] the midpoint values of $\boldsymbol{R}_m$, $\boldsymbol{\Omega}_m$ and $\dot{\boldsymbol{R}}_m$ are evaluated by the midpoint rule as:

$$\begin{aligned}
\boldsymbol{R}_m &= \frac{\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1})) + \boldsymbol{R}(\boldsymbol{\vartheta}(t_n))}{2}, \\
\dot{\boldsymbol{R}}_m &= \frac{\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1})) - \boldsymbol{R}(\boldsymbol{\vartheta}(t_n))}{dt}, \\
\boldsymbol{\Omega}_m &= \frac{\boldsymbol{\Omega}_G(t_{n+1}) + \boldsymbol{\Omega}_G(t_n)}{2}.
\end{aligned} \tag{21}$$

$dt = t_{n+1} - t_n$ is the given time increment. Here $\boldsymbol{R}$ is assumed to be parametrized by $\boldsymbol{\vartheta}$, yet formally equal expressions hold, if $\boldsymbol{R}$ were parametrized by $\boldsymbol{\psi}$ or $\widehat{\boldsymbol{q}}$. By inserting (21) into (5) we get the algebraic equation

$$\frac{\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1})) - \boldsymbol{R}(\boldsymbol{\vartheta}(t_n))}{dt} = \left(\frac{\boldsymbol{R}(\boldsymbol{\vartheta}(t_n)) + \boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1}))}{2}\right)\left(\frac{\boldsymbol{\Omega}_G(t_n) + \boldsymbol{\Omega}_G(t_{n+1})}{2}\right)$$

with the exact solution for $\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1}))$ being

$$\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1})) = \boldsymbol{R}(\boldsymbol{\vartheta}(t_n))\left(\boldsymbol{I} + \frac{dt}{2}\boldsymbol{\Omega}_m\right)\left(\boldsymbol{I} - \frac{dt}{2}\boldsymbol{\Omega}_m\right)^{-1}.$$

Here (see [33])

$$\left(\boldsymbol{I} + \frac{dt}{2}\boldsymbol{\Omega}_m\right)\left(\boldsymbol{I} - \frac{dt}{2}\boldsymbol{\Omega}_m\right)^{-1} = \boldsymbol{I} + \frac{2}{1 + \frac{1}{2}\left\|dt\,\boldsymbol{\omega}_m\right\|^2}\left(\frac{dt}{2}\boldsymbol{\Omega}_m + \frac{(dt)^2}{4}\boldsymbol{\Omega}_m^2\right). \tag{22}$$

After $\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1}))$ has been obtained, $\boldsymbol{\vartheta}(t_{n+1})$ is extracted from $\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1}))$ by Spurrier's algorithm [36]. Although the rotation matrix $\boldsymbol{R}_m$ is approximated by the average value of $\boldsymbol{R}(\boldsymbol{\vartheta}(t_{n+1}))$ and $\boldsymbol{R}(\boldsymbol{\vartheta}(t_n))$, and is therefore not orthogonal, the rotation matrix $\boldsymbol{R}$ at new time t_{n+1} fulfils the orthogonality condition.

The rules (21)–(22) are also applied for parameter $\vec{\psi}$, only that $\boldsymbol{R}(\boldsymbol{\vartheta}(t))$ is replaced with $\boldsymbol{A}(\boldsymbol{\psi}(t))$. Similarly, if we use parameter $\widehat{q}$, $\boldsymbol{R}(\boldsymbol{\vartheta}(t))$ is replaced with the 3×3 submatrix of $\boldsymbol{Q}(\widehat{q}(t))$. Hence, these schemes are algorithmically equivalent, yet in numerical calculations, they can behave differently.

Remark 2 *The midpoint values of the rotation matrix may, alternatively, be computed from the average values of parameters* $\vec{\vartheta}$ *and* $\vec{\psi}$*. Such a proposi-*

tion would result in a different midpoint rotation matrix, yielding non-linear algebraic equations which cannot be solved in a closed analytical form.

The midpoint method is a generalizated Euler method and gives the second-order accurate solution. The method has a local truncation error $O\left(dt^3\right)$ [10].

4.2 MP-q

As pointed out above, the midpoint rule that employs averaged parameters $\overrightarrow{\vartheta}$ or $\overrightarrow{\psi}$, results in the non-linear algebraic equations which must be solved iteratively. In contrast, such an alternative midpoint rule when applied in the quaternion equation (19) yields linear algebraic equations and is, thus, a computationally efficient numerical scheme. The midpoint values of $\widehat{\boldsymbol{q}}$, $\boldsymbol{\omega}$ and $\dot{\widehat{\boldsymbol{q}}}$ are evaluated as

$$\begin{aligned}
\widehat{\boldsymbol{q}}_m &= \frac{\widehat{\boldsymbol{q}}\left(t_{n+1}\right) + \widehat{\boldsymbol{q}}\left(t_n\right)}{2}, \\
\boldsymbol{\omega}_m &= \frac{\boldsymbol{\omega}_G\left(t_{n+1}\right) + \boldsymbol{\omega}_G\left(t_n\right)}{2}, \\
\dot{\widehat{\boldsymbol{q}}}_m &= \frac{\widehat{\boldsymbol{q}}\left(t_{n+1}\right) - \widehat{\boldsymbol{q}}\left(t_n\right)}{dt}.
\end{aligned} \tag{23}$$

Consequently, the midpoint angular velocity quaternion is $\widehat{\boldsymbol{\omega}}_m = \begin{bmatrix} 0 \\ \boldsymbol{\omega}_m \end{bmatrix}$. Multiplying Eq. (20) by $\frac{1}{2}\widehat{\boldsymbol{q}}$ gives

$$\dot{\widehat{\boldsymbol{q}}} = \frac{1}{2}\widehat{\boldsymbol{\omega}}_G \circ \widehat{\boldsymbol{q}}. \tag{24}$$

Eq. (24) is required to be satisfied at time t_m. After inserting Eq. (23) into Eq. (24), we obtain the system of linear equations for $\widehat{\boldsymbol{q}}\left(t_{n+1}\right)$

$$\frac{\widehat{\boldsymbol{q}}\left(t_{n+1}\right) - \widehat{\boldsymbol{q}}\left(t_n\right)}{dt} = \frac{1}{2}\frac{\widehat{\boldsymbol{\omega}}_G\left(t_{n+1}\right) + \widehat{\boldsymbol{\omega}}_G\left(t_n\right)}{2} \circ \frac{\widehat{\boldsymbol{q}}\left(t_{n+1}\right) + \widehat{\boldsymbol{q}}\left(t_n\right)}{2},$$

which is conveniently written in the matrix form

$$\widehat{\boldsymbol{q}}\left(t_{n+1}\right) = \boldsymbol{A}\,\widehat{\boldsymbol{b}}, \tag{25}$$

where

$$\boldsymbol{A} = \frac{4\,dt^2}{16 + dt^2\,|\boldsymbol{\omega}_m|^2} \begin{bmatrix} \frac{4}{dt} & -\omega_{m1} & -\omega_{m2} & -\omega_{m3} \\ \omega_{m1} & \frac{4}{dt} & -\omega_{m3} & \omega_{m2} \\ \omega_{m2} & \omega_{m3} & \frac{4}{dt} & -\omega_{m1} \\ \omega_{m3} & -\omega_{m2} & \omega_{m1} & \frac{4}{dt} \end{bmatrix} \in I\!R^{4,4}$$

$$\widehat{\boldsymbol{b}} = \begin{bmatrix} \frac{1}{dt} q_0\,(t_n) - \frac{1}{4}\boldsymbol{q}\,(t_n) \cdot \boldsymbol{\omega}_m \\ \frac{1}{dt}\boldsymbol{q}\,(t_n) + \frac{1}{4}\,(q_0\,(t_n)\,\boldsymbol{\omega}_m + \boldsymbol{\omega}_m \times \boldsymbol{q}\,(t_n)) \end{bmatrix} \in I\!R^{4,1}$$

and $\boldsymbol{\omega}_m = [\omega_{m1}, \omega_{m2}, \omega_{m3}]^T$. Thus $\widehat{\boldsymbol{q}}\,(t_{n+1})$ can be obtained directly from the given $\widehat{\boldsymbol{q}}\,(t_n)$, $\boldsymbol{\omega}_G\,(t_n)$, $\boldsymbol{\omega}_G\,(t_{n+1})$, and dt by operating a matrix on a vector. Note that $\widehat{\boldsymbol{q}}\,(t_{n+1})$ obtained from Eq. (25) is not the rotational quaternion. Finally, the computed $\widehat{\boldsymbol{q}}\,(t_{n+1})$ is normed to reduce the scale and the screw errors of rotation [26].

The matrix $\boldsymbol{A}$ has the following properties: (i) it is a normal matrix, $\boldsymbol{A}\,\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A}$; (ii) its determinant is positive

$$\det \boldsymbol{A} = \frac{256 dt^4}{\left(16 + dt^2\,|\boldsymbol{\omega}_m|^2\right)^2} > 0;$$

it has double conjugated eigenvalues

$$\lambda_{1,2,3,4} = \frac{4\,dt\left(4 - dt\sqrt{-|\boldsymbol{\omega}_m|^2}\right)}{16 + dt^2\,|\boldsymbol{\omega}_m|^2}$$

whose absolute values are equal to $\frac{4\,dt}{\sqrt{16+dt^2|\boldsymbol{\omega}_m|^2}} < dt$. Therefore the method is stable for $dt \leq 1$, see [4, pp 806–808]. It has to be pointed out that this integration method does not require the evaluation of a rotation matrix at any time.

4.3 Runge-Kutta methods

The Runge-Kutta methods are widely used and often implemented in commercial computer programs. The methods are well documented in standard textbooks of numerical analysis (e.g. [14] and [16]).

A typical Runge-Kutta method assumes the solution of the differential equa-

tion $\dot{y}(t) = f(t, y(t))$ at time t_{n+1} in the form

$$y(t_{n+1}) = y(t_n) + \sum_{i=1}^{s} \gamma_i k_i \quad \text{with} \quad k_i = dt\, f\left(t_n + dt\, \alpha_i, y(t_n) + \sum_{j=1}^{i} \beta_{ij} k_j\right), \tag{26}$$

where s is the number of stages of the method, $dt = t_{n+1} - t_n$ is the time step, and coefficients α_i, β_{ij}, γ_i specify the particular method. The order of the method is derived from the comparison between the approximation (26) and the exact Taylor series expansion.

In what follows we will employ a time-step adapting fifth-order explicit Runge-Kutta method *ode45,* implemented in the commercial program *Matlab* [25]. It is a seven-stage explicit Runge-Kutta formula-based routine suitable for exact integration, see Dormand and Prince [13], which stems from the work of Fehlberg. The details on Fehlberg's approach can be found in [10]. The Butcher array of the method reads [13]:

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	−56/15	32/9				
8/9	19372/6561	−25360/2187	−212/729				
1	9017/3168	−355/33	46732/5247	49/176	−5103/18656		
1	35/384	0	500/1113	125/192	−2187/6784	11/84	
	5179/57600	0	7571/16695	393/640	−92097/339200	187/2100	1/40
	35/384	0	500/1113	125/192	−2187/6784	11/84	0

The last two lines give the fourth-order and the fifth-order accurate solution. The method was constructed such to provide the minimal local truncation error of the fifth-order solution $O(dt^6)$, while the difference between the fourth and the fifth-order solutions is sufficiently large to enable the step-size control. The Matlab environment enables the error control for both the absolute and the relative local errors, which adapts the time step accordingly.

The method can easily be applied for the solution of Eq. (2) in conjunction with any of the three parametrizations, employing Eqs. (7), (10) and (19), respectively. For the sake of brevity of notation, we will indicate the choice of the parametrization as *ode45*(ϑ), *ode45*(ψ), and *ode45*(q).

5 Numerical tests

The tests of accuracy, stability and efficiency of the above described numerical methods when combined with different parametrizations of rotations are performed for characteristic numerical examples. The first example presents a small planar rotation in a relatively large time interval. The second example includes moderate rotations with oscillations of different amplitudes and frequencies. The third example includes unlimited highly oscillating spatial rotations.

All computations have been performed in the Matlab 7.8 environment on the PC with Intel Core2 processor which runs at 2.66GHz.

5.1 Comparison measures

Firstly we introduce some further notations and the comparison measures needed for the assessment of the methods.

NOTATIONS.

- *Exact rotational vector.*
  We choose analytical functions for the components of the rotational vector $\boldsymbol{\vartheta}(t) = [\vartheta_1(t), \vartheta_2(t), \vartheta_3(t)]$ and compute a $N \times 3$ matrix $[\boldsymbol{\vartheta}(t_i)]_{t_i \in T} = [\boldsymbol{\vartheta}_1(t_i), \boldsymbol{\vartheta}_2(t_i), \boldsymbol{\vartheta}_3(t_i)]_{t_i \in T}$ consisting of the values of $\boldsymbol{\vartheta}(t)$ at chosen times $t_i \in T$. Using Eq. (2) and $\boldsymbol{\vartheta}(t)$ we derive the $N \times 3$ matrix of angular velocity vectors $[\boldsymbol{\omega}(t_i)]_{t_i \in T} = [\boldsymbol{\omega}_1(t_i), \boldsymbol{\omega}_2(t_i), \boldsymbol{\omega}_3(t_i)]_{t_i \in T}$. These functions and their point values will be termed *exact functions* and *exact values*. Because the computed results of the rotational vector are saw-toothed (constrained to an interval $\left[-\frac{\pi}{2}, \frac{3\pi}{2}\right]$, see Sec. 5.4), the *constrained exact rotational vector values* are also computed for comparisons and are denoted by $\left[\overline{\boldsymbol{\vartheta}}(t_i)\right]_{t_i \in T}$. They are obtained from $[\boldsymbol{\vartheta}(t_i)]_{t_i \in T}$ by first calculating the rotation matrix from the Rodrigues formula (6) and then extracting $\overline{\boldsymbol{\vartheta}}$ by the Spurrier algorithm [36]. This algorithm is computationally stable and does not introduce the error bigger than the round-off error of the computer.
- *Exact tangential vector and quaternion.*
  Exact tangential vector and rotational quaternion are computed directly from the exact components of the rotational vector using the formulae from Table 1.
- *Computed (approximate) rotational vectors and quaternions.*
  Computed results of the rotational quantities carry exact notations of the particular parametrization, ϑ, ψ, or q, and method, $MP-R$, $MP-q$, or

ode45.

COMPARISON MEASURES. The results of the numerical solution are given as a discrete set of values at time-points in interval $[0, t_N]$. A small translation with respect to time in the case of a highly increasing solution can represent a relatively small error of the solution on the whole, and the introduced error measure should be able to consider such an error properly. This is achieved by introducing the global L^2 function norm [31]

$$\|f\|_2 = \sqrt{\int_a^b |f|^2 \, dx}.$$

Function $|f|^2$ must be integrable on the interval $[a, b]$, i.e. $f \in L^2([a, b])$. The L^2 norm of the difference between the two functions $f, g \in L^2([a, b])$ is then given by

$$\|f - g\|_2 = \sqrt{\int_a^b |f - g|^2 \, dx}.$$

For our present objective, which is to discuss the global error of the solution, the combination of the absolute and the relative L^2 norm,

$$RL^2 = \frac{\|f - g\|_2}{\max\{1, \|f\|_2\}},$$

is found more appropriate. Here, f denotes the exact function and g its approximation. Note that a constant difference between the solution and the exact function during the time interval induces a linearly increasing RL^2 norm. To make the discrete set of resulting values a square integrable function, we assume the linear interpolation over each time increment and use the trapezoidal integration rule.

In what follows we will see that the RL^2 norm is in some examples limited to one. This happens only when two conditions are satisfied: (i) the magnitude of the norm of the analytical solution is larger than one, i.e. $\max\{1, \|f\|_2\} = \max \|f\|_2$, and (ii) the numerical result g is underestimated so that $\|f - g\|_2 \approx \|f\|_2$. Hence, when RL^2 approaches value one, the results are no longer reliable.

5.2 Example 1: Bounded planar rotation

In our first numerical example, the rotational vector is given by

$$\boldsymbol{\vartheta}(t) = \left[\sin^2(2t), 0, \cos(2t)\right].$$

Here the norm of the rotational vector is bounded, $|\boldsymbol{\vartheta}| = \vartheta \leq 1$. Fig. 1 shows the time variations of the components of the three rotational parametrizations

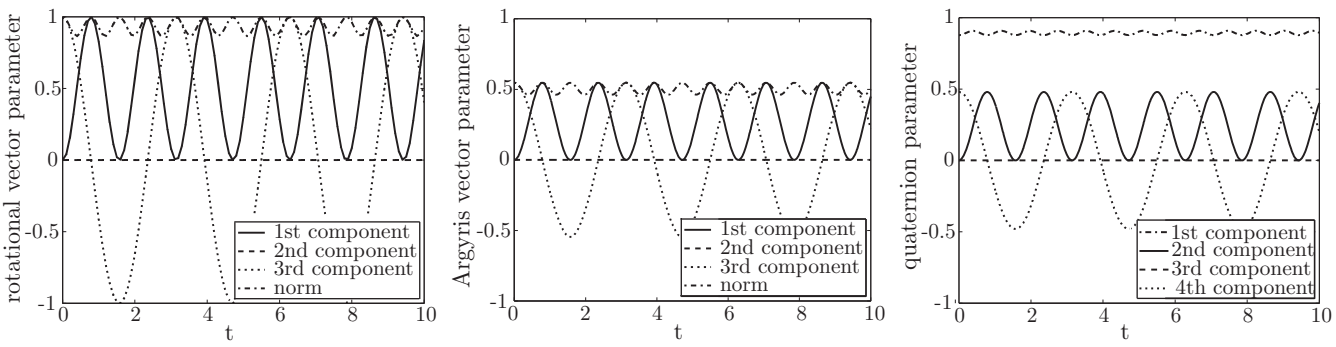


Figure 1. Example 1: Graphs of the exact rotation component functions and vector norms for different parametrizations.

and related vector norms. The quaternion norm is 1 and is therefore not displayed in the figure. We assume a relatively large time interval $[0, 100]$. We employ method *ode45*, and prescribe the local relative error to be less than 10^{-10}. The number of adaptive time steps ranges from 2805 to 3061 for various parametrizations. The constant time step for midpoint integrators is taken to be 0.033, resulting in 3031 steps.

The comparisons of the RL^2 error norms on the complete time interval for different parametrizations and the associated computational times are presented in Table 2. Small differences in error between different parametrizations when *ode45* method is applied are due to the different number of time steps. The rotational vector needs the largest number of time steps to satisfy the local error demand. The accuracy of the MP-R methods when applied to all three parametrizations is comparable. The error for the MP-q method is about one half of the MP-R methods. The most effective methods in terms of the computational time appears to be MP-R(ψ). The growth of the RL^2 error norms with time are displayed in Fig. 2 for the component $k = 1$, where the errors appear to be the largest. Note that the RL^2 error grows nearly linearly with time for any method and parametrization, which indicates the nearly constant spacing between exact and approximative rotation parameters with time.

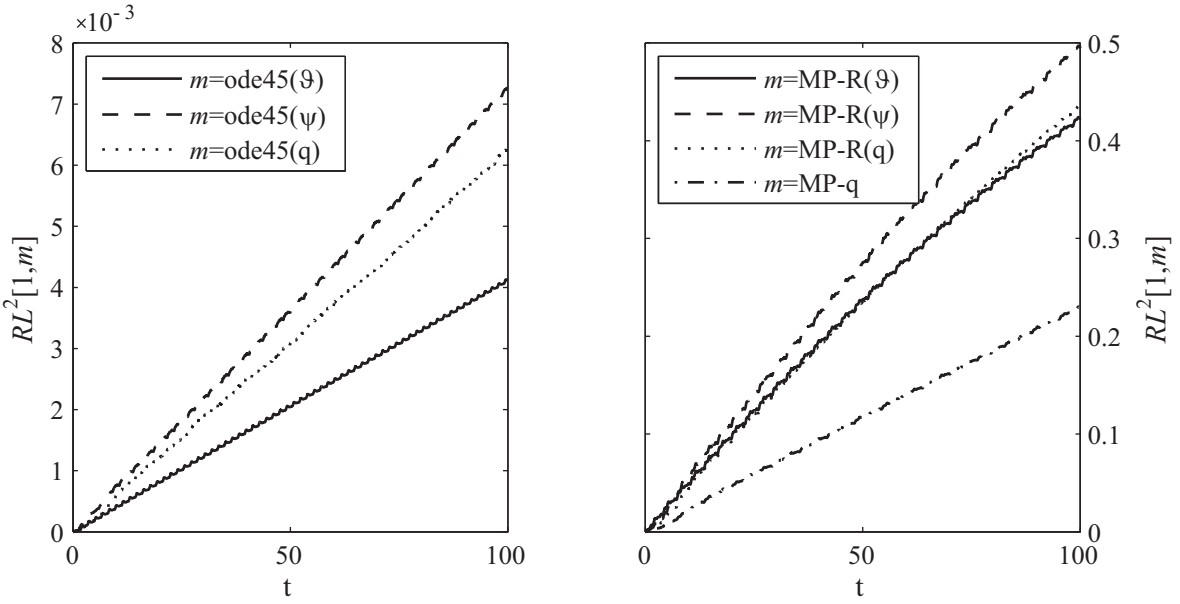
As observed from Table 2, the computational times and the number of time steps of the high-order Runge-Kutta *ode45* and midpoint methods are very similar, while the RL^2 errors differ substantially. It is found instructive to compare the midpoint methods to the low-order Runge-Kutta method all having a comparable theoretical accuracy. Such a method implemented in Matlab environment is the 4-stage Bogacki-Shampine 2(3) Runge-Kutta method, denoted as *ode23*, with a little lower turnication error as the midpoint method [5]. The comparisons are displayed in Table 2, see column ODE23. The local relative error is taken to be less than 10^{-4}. The results show that *ode23* requires about two to three times longer computational time for roughly equal number of time steps as applied in midpoint methods, but its accuracy is better. This can be atributed to the slightly higher order of the method and to

Table 2
Example 1: Comparisons of errors, numbers of time steps, and computational times.

method m	ODE45			ODE23			MP-R			MP-q
parameter α	ϑ	ψ	q	ϑ	ψ	q	ϑ	ψ	q	q
time steps	3061	2805	2869	3318	3402	3184	constant: 3031			
comp. time [s]	0.86	0.63	0.67	1.95	1.51	1.55	0.78	0.57	0.87	0.90
$RL^2\,[0,m]$	–	–	0.003	–	–	0.02	–	–	0.05	0.03
$RL^2\,[1,m]$	0.004	0.007	0.006	0.01	0.02	0.07	0.4	0.5	0.4	0.2
$RL^2\,[2,m]$	0.003	0.003	0.001	0.5	0.1	0.2	0.8	0.9	0.8	0.5
$RL^2\,[3,m]$	0.0002	0.005	0.002	0.01	0.01	0.06	0.2	0.2	0.2	0.07

the adaptive time-stepping implemented in *ode23*.

Figure 2. Example 1: Graphs of the RL^2 error of the 1st component of the rotational vector for different parametrizations; left for method *ode45*, and right for the midpoint methods.



In general, all the above described methods turn out to be appropriate for such a simple plane rotation. As expected the tangential vector shows the fastest performance. Among the midpoint rules MP-q seems to be the most accurate.

5.3 Example 2: Oscillating moderate rotation

Here we attempt to solve system (2) for two different rotational vectors, whose components experience oscillations of very different frequencies and ampli-

tudes, both for the planar

$$\boldsymbol{\vartheta}\,(t) = \left[\sin^2(2t), 0, \sin\,(t) + 0.08 \cdot \cos(100t)\right], \quad t \in [0, 10] \tag{27}$$

and the three-dimensional case

$$\boldsymbol{\vartheta}\,(t) = \left[\sin^2(2t), \cos\,(t) + 0.08 \cdot \sin(100t), \sin\,(t) + 0.08 \cdot \cos(100t)\right], \quad t \in [0, 100]\,. \tag{28}$$

The applied time intervals of the solution are $[0, 10]$ and $[0, 100]$, respectively. Fig. 3 shows how the components and the vector norms change with time for the rotational vector given in (27). The local relative error in the method *ode45* was taken to be equal or less than 10^{-10}.

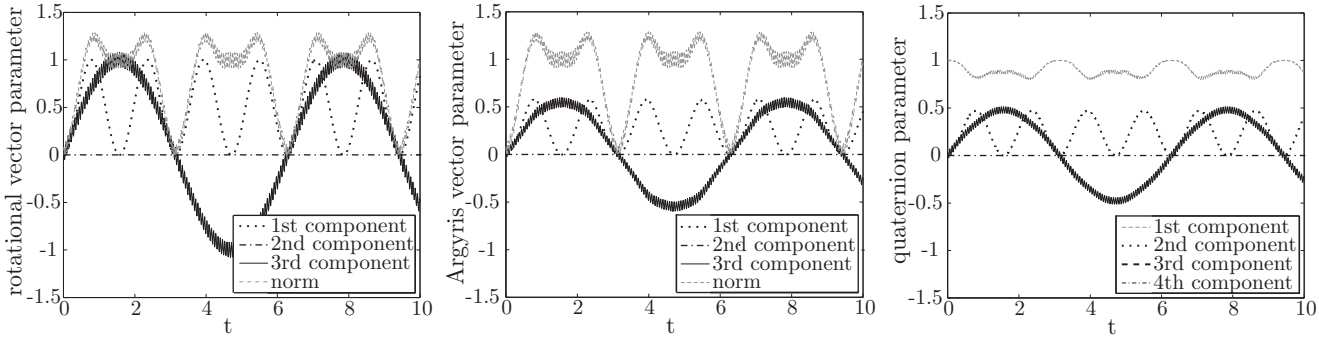


Figure 3. Example 2, Eq. (27): Graphs of the exact rotation component functions and vector norms for different parametrizations.

For the integration of Eq. (2) in the time interval $[0, 10]$, the *ode45* method required from 3825 to 4585 time steps depending on the parametrization; a large number of time steps well indicates that the problem is more demanding. The time step size varies between 0.00086 and 0.0031. The constant time step used in the midpoint methods is 0.0025, which corresponds to 4000 time steps.

Table 3
Example 2, Eq. (27): Comparisons of errors, numbers of time steps and computational times.

method m	ODE45			MP-R			MP-q
parameter α	ϑ	ψ	q	ϑ	ψ	q	q
time steps	4585	4145	3825	constant: 4000			
comp. time [s]	1.64	0.91	0.90	0.92	0.63	0.70	0.71
$RL^2\,[0, m]$	-	-	0.0029	-	-	0.030	0.029
$RL^2\,[1, m]$	0.00005	0.0015	0.0008	0.003	0.023	0.010	0.009
$RL^2\,[2, m]$	0.0001	0.00004	0.00006	0.016	0.009	0.009	0.008
$RL^2\,[3, m]$	0.009	0.014	0.020	0.242	0.256	0.238	0.234

Table 3 shows the comparisons of the resulting accuracy and the computational times for $\boldsymbol{\vartheta}(t)$ from Eq. (27). For the same local accuracy demands, *ode45* method, when combined with rotational quaternion, needs the minimum number of time steps and results in the fastest performance. Small differences in global accuracy are due to different numbers of time steps.

The errors of the midpoint methods are roughly at least one order larger than those of the ode methods, yet very similar for all midpoint methods used. This is also clearly displayed in Fig. 4, which shows the evolution of the RL^2 error norms with time for the biggest error component. In contrast to Example 1, the errors of the MP-R(q) and MP-q are nearly equal.

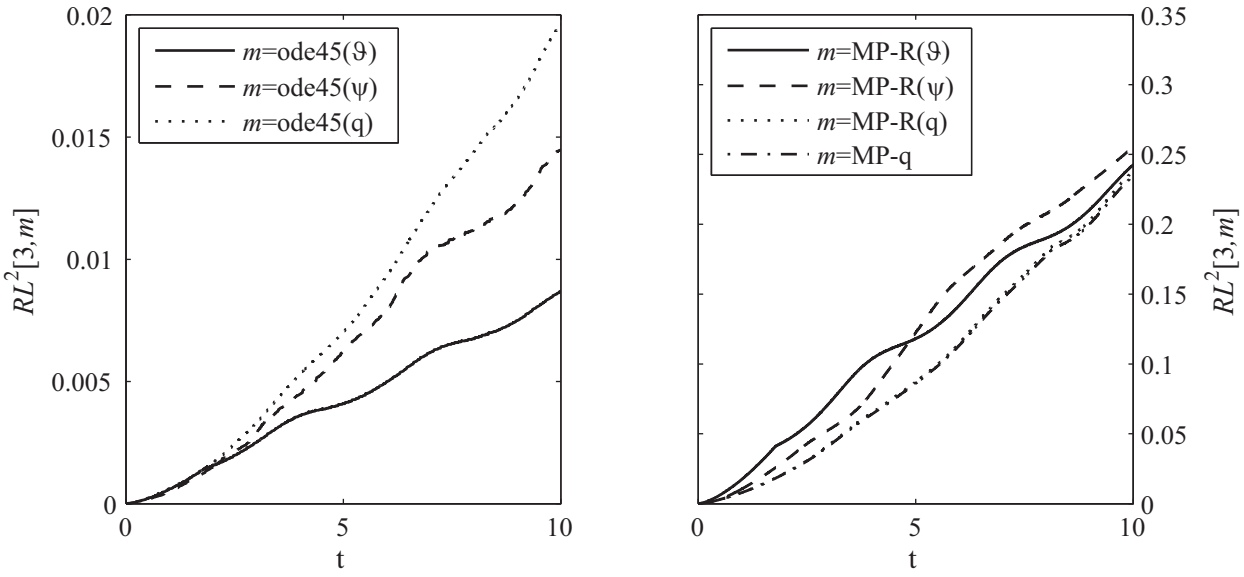


Figure 4. Example 2, Eq. (27): Graphs of the RL^2 error of the 3rd component of the rotational vector for different parametrizations; left for method *ode45*, and right for the midpoint methods.

Similar conclusions can be stated for the fully spatial example, Eq. (28), in the long-range analysis (Fig. 5), where about 10-times larger numbers of time steps are used (48397, 44637 and 41457 for methods *ode45*(ϑ), *ode45*(ψ) and *ode45*(q), respectively, and 50000 equal time steps for all midpoint rules). We can again observe a nearly linear RL^2 error, which indicates constant spacing between exact and approximative rotation parameters with time for the *ode*45 integrators, while for the midpoint integrators RL^2, errors are clearly growing and before $t = 50$ the errors are of the same order as the result. This clearly indicates that three-dimensional case of Example 2 is too demanding for the second-order midpoint integrators.

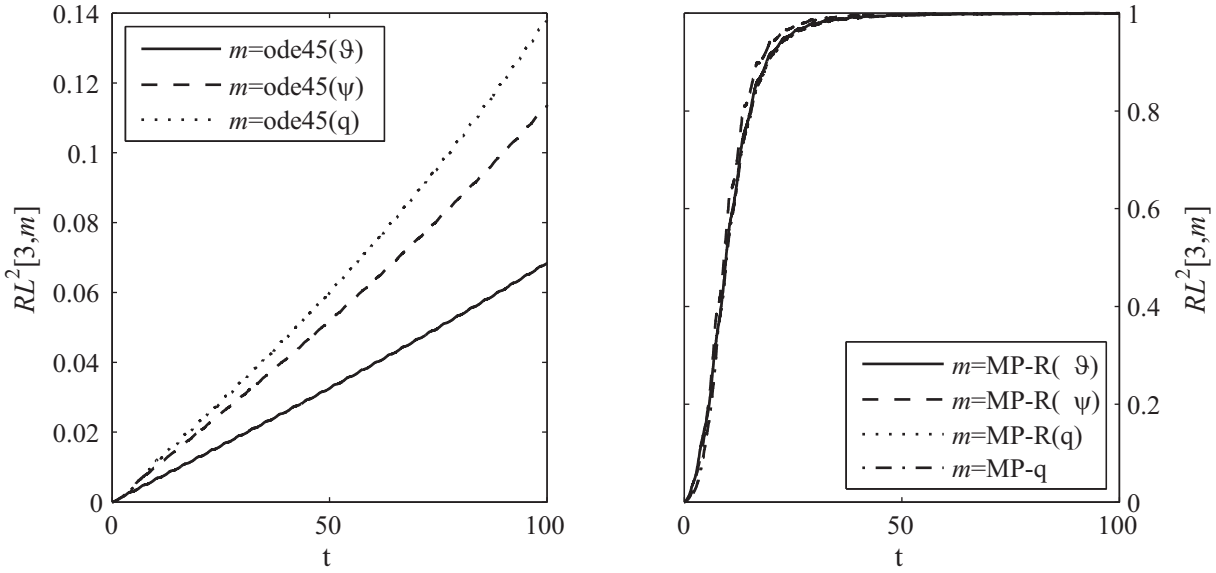


Figure 5. Example 2, Eq. (28): Graphs of the RL^2 error norm of the 3rd component of the rotational vector for different parametrizations; left for method *ode45*, and right for the midpoint methods.

5.4 Example 3: large rotation

This example explores the integration of large rotations. The numerical experimentations show that the crossing of value 2π does not necessarily appear to be a problem, if the rotational vector parametrization is employed. If, e.g. $\boldsymbol{\vartheta}(t) = [0, 0, 100t + 0.01]$, and the time interval $[0, 10]$, the angle of rotation ϑ crosses the multiple of 2π exactly 159 times. Yet solving Eq. (2) presents no trouble. Note also that the MP-R(ϑ) integrator gives us the restricted results for the rotational vector (see Fig. 6). Similar conclusions hold for the spatial

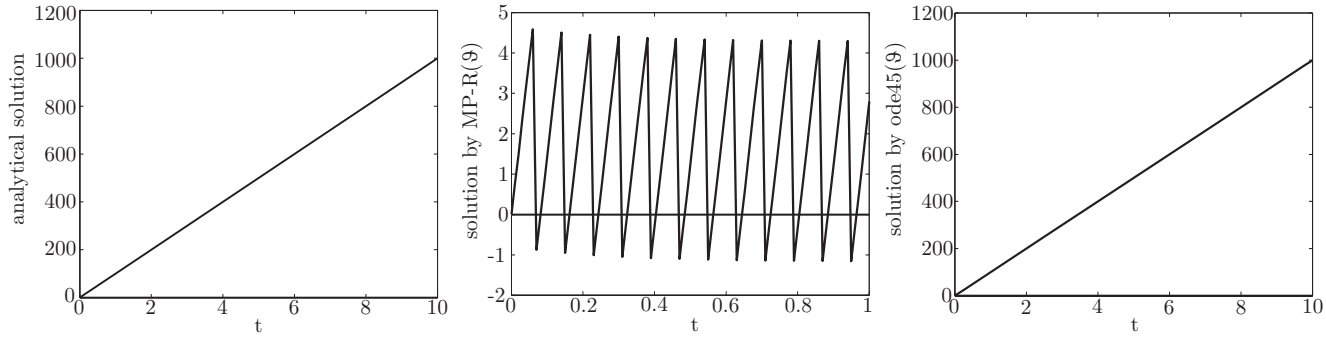


Figure 6. The rotational vector component functions for $\boldsymbol{\vartheta} = [0, 0, 100t + 0.01]$.

rotation $\boldsymbol{\vartheta}(t) = [t + 1, 2t, 3t]$ and the time interval $[0, 10]$. In this test the angle of rotation crosses the multiple of 2π five times. In both examples *ode45*(ψ) fails to obtain the solution. The second graph in Fig. 6 indicates that the angles $-\frac{\pi}{2}$ and $\frac{3\pi}{2}$ represents the limits of the restriction interval for the results com-

puted by method MP-R(ϑ). *These limits are not related to the critical points of the operator defined in Eq. (8).* The restricted results for the rotational vector are compared to the restricted exact values $\left[\overline{\boldsymbol{\vartheta}}\left(t_i\right)\right]_{t_i \in T}$ mentioned earlier.

An example in which *ode45*(ϑ) fails, because the time-step size has to be reduced below the smallest value allowed, is (Fig. 7)

$$\boldsymbol{\vartheta}\left(t\right) = \left[2 + \sin^2(2t), t, 5t^3 - 4t\right]. \tag{29}$$

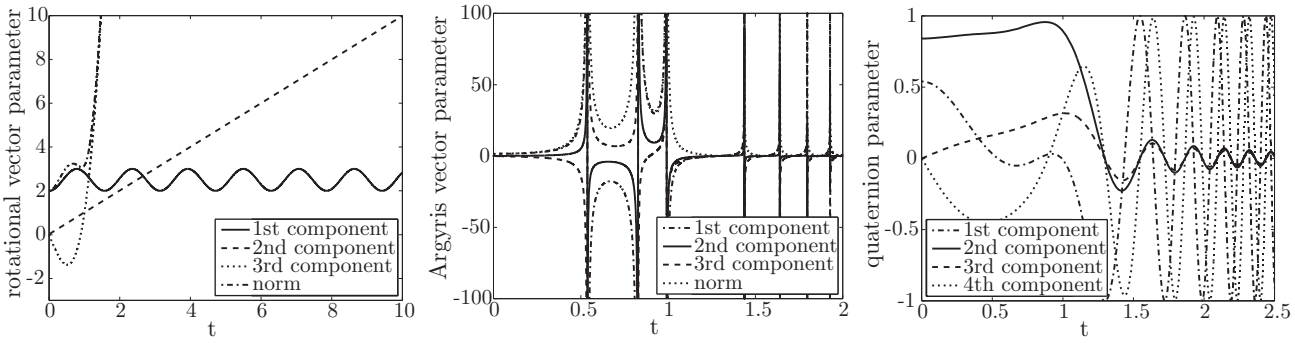


Figure 7. Example 3, Eq. (29): Graphs of the exact rotation component functions and vector norms for different parametrizations.

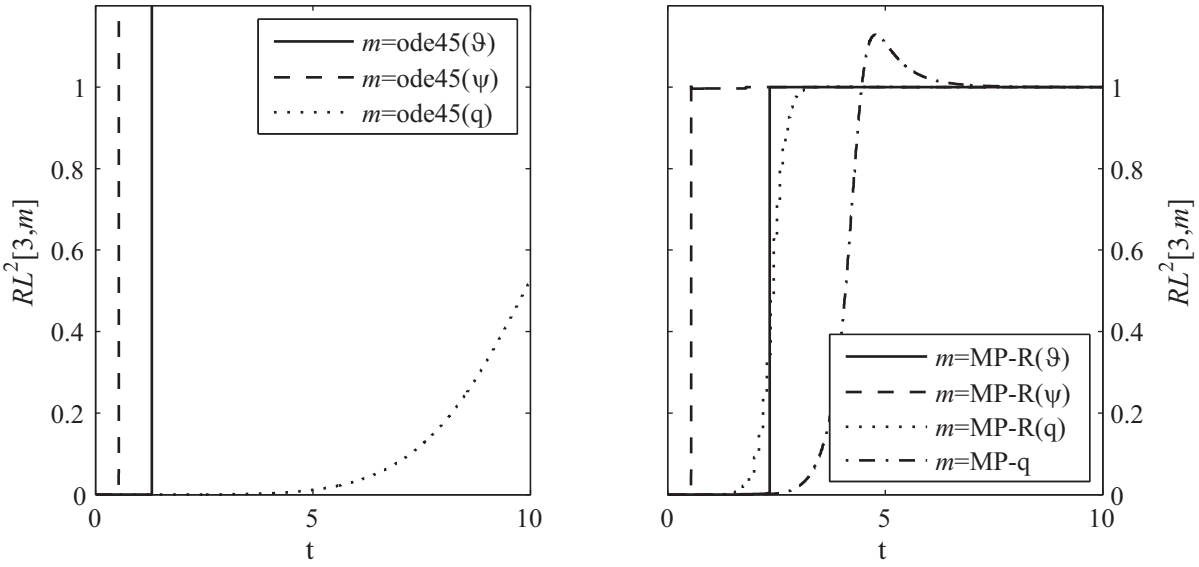


Figure 8. Example 3, Eq. (29): Graphs of the RL^2 error norm of the 3rd component for different parametrizations; left for method *ode45* and right for the midpoint methods.

The early failure is experienced for vector parametrizations (for the rotational vector parametrization, at $t \approx 1.296609$, and for the Argyris vector

parametrization, at $t \approx 0.5375769$) but not for the quaternion parametrization. Here the advantage of the quaternion parametrization is its restriction to unity, as well as the absence of any singularity. The method *ode45*(q) required 45613 time steps on the whole time interval $[0, 10]$ and the time step size ranges between $2.9 \cdot 10^{-5}$ and 0.026. The constant time step used in the midpoint methods is 0.0002, which corresponds to 50 000 time steps.

Remark 3 *The analyses using ode45*(ϑ) *and ode45*(ψ) *also fail, if we simplify the rotational vector by imposing* $\vartheta_1(t) = 0$ *or* $\vartheta_2(t) = 0$ *in Eq. (29). If we put* $\vartheta_1(t) = \vartheta_2(t) = 0$*, only method ode45*$(\psi)$ *fails, however.*

Remark 4 *Ibrahimbegović and Mikdad [20] noticed that the direct implementation of the Newmark scheme on the total rotational vector results in inability of the method to continue the computation, when the norm of the rotational vector approaches* 2π*. This is consistent with the present numerical results, showing that the computed value for* ϑ *at the critical time* $t_i = 1.296580...$ *is* $6.28265...$ *i.e.* $0.999\,91... \cdot 2\pi$*.*

As observed from Fig. 8, only the method *ode45*(q) gives sufficiently accurate results and an economic computational time. None of the midpoint methods is capable to obtain sufficiently accurate solution in the whole time interval. The midpoint method when combined with the quaternion parametrization, MP-q, seems to be most successful, however. Those using the vector parametrization experience a sudden blow up of the error; for the Argyris vector parametrization, the blow up occurs roughly at the first critical point, and for the rotational vector parametrization, around the second critical point. Among all the midpoint methods, the MP-q method appears to be the most accurate and its result acceptable for $t < 3$. When LR^2 norm is close to one, results become completely unreliable.

Regarding the performance of the ode methods near failure, the failure of *ode45*(ϑ) is indicated by a rapid decrease of the time step. This takes place as soon as the rotational vector norm becomes roughly a 2π-multiple. At an instant of time, when the rotational vector norm is very close to 2π, a time step, smaller than the machine precision is needed to satisfy the local error demand; consequently, the singularity is nearly met and the failure of the method *ode45*(ϑ) cannot be avoided. The reason for a sudden error blow up of the MP-R(ϑ) method is similar. Here, the time step is constant, but again rather small, so that there exists a high probability that one of the time steps will be close enough to induce a near-singularity of the system matrix. Because we do not control demands for the local error in the MP solvers, the solver continues with the calculations, yet the results are useless, henceforth.

As we have already mentioned, solvers using the tangential vector parametrization of rotation are effective only for rotations with angle of rotation lying

within the time interval $(\pi + 2k\pi, \pi + 2(k+1)\pi)$, for some $k \in \mathbb{Z}$. That is why the failure of the *ode45*(ψ) cannot be avoided, when the norm of the Argyris tangential vector becomes large resulting in the angle of rotation of about $\pi + 2n\pi$. The numerical failure of MP-R(ψ) does not appear, but results are nonetheless useless.

6 Conclusions

The differential equation relating rotation and angular velocity has been numerically integrated for the rotations when the angular velocity is a given function of time. We have studied the effect of the choice of the parametrization of rotation, when employed in conjunction with different numerical time-integration schemes, on the accuracy and the computational efficiency. We have combined the rotational vector, the tangential vector or the rotational quaternion with either one of the two alternative midpoint methods or with the Runge-Kutta method.

By employing various angular velocities, representing small, moderate and highly oscillating rotations as input, we have found out that the quaternion parametrization of rotations combined with the newly developed midpoint integration outperforms in accuracy the classical one developed in [33] at roughly the same computational demands. We have also found out that the numerical instability has not emerged with any of the quaternion-based integration methods if compared to rotational and tangential vector parametrizations.

While in the rotation matrix error analysis, the drift, skew and scale errors are often employed as the local error measures [23,26], we have here used the L^2 norm of the rotational vector and quaternion components which is a well defined mathematical norm appropriate as the measure of the global error of the solution on the whole time interval.

Acknowledgment

This work was supported by the Slovenian Research Agency through the research programme P2-0260. The support is gratefully acknowledged.

References

[1] M. S. Ahmed, D. V. Ćuk, *Comparison of different computation methods for strapdown inertial navigation systems*, Scientific-Tehnical Review **55** (2005) 22–29.

[2] J. Argyris, *An excursion into large rotations*, Comput. Methods Appl. Mech. Engrg. **32** (1982) 85–155.

[3] S. N. Atluri, A. Cazzani, *Rotations in computational solid mechanic*s, Arch. Comput. Methods Engrg. **2** (1995) 49–138.

[4] K. J. Bathe, *Finite Element Procedures*, Prentice-Hall International, Inc., 1996.

[5] P. Bogacki, F. Shampine, *A 3(2) pair of Runge-Kutta formulas*, Appl. Math. Letters **2** (1989) 1–9.

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

[7] C. L. Bottasso, M. Borri, *Integrating finite rotations*, Comput. Methods Appl. Mech. Engrg. **58** (1986) 79–116.

[8] C.L. Bottasso, *A new look at finite elements in time: a variational interpretation of Runge-Kutta methods,* Appl. Numer. Math. **25** (1997) 355–368.

[9] B. Brank, J. Korelc, A. Ibrahimbegović, *Dynamics and time-stepping for elastic shells undergoing finite rotations,* Computers & Structures **81** (2003) 1193–1210.

[10] J.C. Butcher, *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*, John Wiley&Sons, Chichester – New York – Brisbane – Toronto – Singapore, 1987.

[11] S. B. Choe, J. J. Faraway, *Modeling head and hand orientation during motion using quaternions*, Society of Automotive Engineers (2004) 1–7.

[12] M. A. Crisfield, *Non-linear Finite Element Analysis of Solids and Structures, Volume 2, Advanced Topics*, John Wiley & Sons, Chichester–New York–Weinheim–Brisbane–Singapore–Toronto, 1997.

[13] J. R. Dormand, P. J. Prince, *A family of embedded Runge-Kutta formulae*, J. Comp. Appl. Math. **6** (1980) 19–26.

[14] G. A. Evans, *Practical Numerical Analysis*, John Wiley&Sons, Chichester – New York – Brisbane – Toronto – Singapore, 1995.

[15] B. Friendland, *Analysis of strapdown navigation using quaternions*, IEEE Transactions on Aerospace and Electronic Systems **14** (1978) 764–768.

[16] C. F. Gerald, P. O. Wheatley, *Applied Numerical Analysis*: *fifth edition*, Addison-Wesley publishing company, 1994.

[17] S. Ghosh, D. Roy, *Consistent quaternion interpolation for objective finite element approximation of geometrically exact beam,* Comput. Methods Appl. Mech. Engrg. **198** (2008) 555–571.

[18] A. Ibrahimbegović, *On the choice of finite rotation parameters,* Comput. Methods Appl. Mech. Engrg. **149** (1997) 49–71.

[19] A. Ibrahimbegović, S. Mamouri, *Energy conserving/decaying implicit time-stepping scheme for nonlinear dynamics of three-dimensional beams undergoing finite rotations,* Comput. Methods Appl. Mech. Engrg. **191** (2002) 4241–4258.

[20] A. Ibrahimbegović, M. al Mikdad, *Finite rotations in dynamics of beams and implicit time-stepping schemes*, Int. J. Numer. Methods Engrg. **41** (1998) 781–814.

[21] G. Jelenić, M.A. Crisfield, *Dynamic analysis of 3D beams with joints in presence of large rotations,* Comput. Methods Appl. Mech. Engrg. **190** (2001) 4195–4230.

[22] G. Jelenić, M. Saje, *A kinematically exact space finite strain beam model–finite element formulation by generalized virtual work principle,* Comput. Methods Appl. Mech. Engrg. **120** (1995) 131–161.

[23] Y. F. Jiang, Y. P. Lin, *Error analysis of quaternion transformations*, IEEE Transactions on Aerospace and Electronic Systems **27** (1991) 634–639.

[24] S. M. Johnson, J. R. Williams, B. K. Cook, *Quaternion-based rigid body rotation integration algorithms for use in particle methods*, Int. J. Numer. Meth. Engrg. (2007), doi: 10.1002/nme.2210.

[25] The MathWorks, Inc. *MATLAB, Using Matlab*, Natick, 1999.

[26] R. E. Mortensen, *Strapdown guidance error analysis*, IEEE Transaction on Aerospace and Electronic Systems **10**, 451–457, 1974.

[27] W. F. Phillips, C. E. Hailey, G. A. Gebert, *A review of attitude kinematics for aircraft flight simulation*, Modeling and Simulation Technologies Conference, 14-17 August 2000, Denver, Colorado.

[28] I. R. Poreous, *Clifford Algebras and the Classical Groups*, Cambridge University Press 1995.

[29] I. Romero, *The interpolation of rotations and its application to finite element models of geometrically exact rods,* Comput. Mech. **34** (2004) 121–133.

[30] I. Romero, F. Armero, *An objective finite element approximation of the kinematics of geometrically exact rods and its use in the formulation of an energy-momentum conserving scheme in dynamics,* Int. J. Numer. Methods Eng. **54** (2002) 1683–1716.

[31] W. Rudin, *Real and Complex Analysis*, *Third edition*, McGraw-Hill Book Company, 1987.

[32] K. Shoemake, *Animating rotation with quaternion curves*, ACCM Siggraph. **19** (1985) 245–254.

[33] J. C. Simo, N. Tarnow, M. Doblare, *Non-linear dynamics of three-dimensional rods: exact energy and momentum* conserving algorithms, Int. J. Numer. Methods Engng. **38** (1995) 1431–1473.

[34] J. C. Simo, L. Vu-Quoc, *On the dynamics in space of rods undergoing large motions - a geometrically exact aproach*, Comput. Methods Appl. Mech. Engrg. **66** (1988) 125–161.

[35] J. C. Simo, K. K. Wong, *Uconditionaly stable algorithms for rigid body dynamics that exactly preserve energy and momentum*, Int. J. Numer. Methods Engng. **31** (1991) 19–52.

[36] R. A. Spurrier, *Comment on "Singularity-free extraction of a quaternion from a direction-cosine matrix"*, J. Spacecraft **15** (1978) 255.

[37] J. P. Ward, *Quaternions and Cayley Numbers*, Kluwer academic Publishers, Dordrecht/Boston/London, 1997.

[38] D. Zupan, M. Saje, *Finite-element formulation of geometrically exact three-dimensional beam theories based on interpolation of strain measures*, Comput. Methods Appl. Mech. Engrg. **192** (2003) 5209–5248.

[39] D. Zupan, M. Saje, *The three-dimensional beam theory: Finite element formulation based on curvature*, Comput. Struct. **81** (2003) 1875–1888.

[40] E. Zupan, M. Saje, D. Zupan, *The quaternion-based three-dimensional beam theory,* Comput. Methods Appl. Mech. Engrg. **198** (2009) 3944–3956.