Acta Mech 224, 3091–3109 (2013)
DOI 10.1007/s00707-013-0914-2

F. Zhao · B. G. M. van Wachem

A novel Quaternion integration approach for describing the behaviour of non-spherical particles

Received: 17 November 2012 / Revised: 6 March 2013 / Published online: 13 July 2013
© The Author(s) 2013. This article is published with open access at Springerlink.com

Abstract There are three main frameworks to describe the orientation and rotation of non-spherical particles: Euler angles, rotation matrices and unit Quaternions. Of these methods, the latter seems the most attractive for describing the behaviour of non-spherical particles. However, there are a number of drawbacks when using unit Quaternions: the necessity of applying rotation matrices in conjunction to facilitate the transformation from body space to world space, and the algorithm integrating the Quaternion should inherently conserve the length of the Quaternion. Both drawbacks are addressed in this paper. The present paper derives a new framework to transform vectors and tensors by unit Quaternions, and the requirement of explicitly using rotation matrices is removed altogether. This means that the algorithm derived in this paper can describe the rotation of a non-spherical particle with four parameters only. Moreover, this paper introduces a novel corrector-predictor method to integrate unit Quaternions, which inherently conserves the length of the Quaternion. The novel framework and method are compared to a number of other methods put forward in the literature. All the integration methods are discussed, scrutinised and compared to each other by comparing the results of four test cases, involving a single falling particle, nine falling and interacting particles, a 2D prescribed torque on a sphere and a 3D prescribed torque on a non-spherical particle. Moreover, a convergence study is presented, comparing the rate of convergence of the various methods. All the test cases show a significant improvement of the new framework put forward in this paper over existing algorithms. Moreover, the new method requires less computational memory and fewer operations, due to the complete omission of the rotation matrix in the algorithm.

1 Introduction

Understanding the behaviour of rigid particles is important for many industrial processes and phenomena occurring in nature. It is estimated that over 70 % of chemical processes involve small particles at some point. Moreover, particles play a large role in natural phenomena, such as avalanches, sediment transport and erosion, to name just a few. Because of this importance, the field of modelling the behaviour of large number particles is well established. In 1979, Cundall et al. [10] proposed the distinct element method (DEM), which models the behaviour of individual particles by solving Newton's second law in a Lagrangian framework. This method has proved to be highly useful and is used in hundreds of research papers. Moreover, any numerical simulation of multi-body dynamics, especially those involving free body rotations, require accurate integration of the rigid body orientation equations.

F. Zhao · B. G. M. van Wachem (✉)
Division of Thermofluids, Department of Mechanical Engineering,
Imperial College London, Exhibition Road, London SW7 2AZ, UK
E-mail: berend.van.wachem@gmail.com
Tel.: +44-20-75947030
Fax: +44-20-75945702

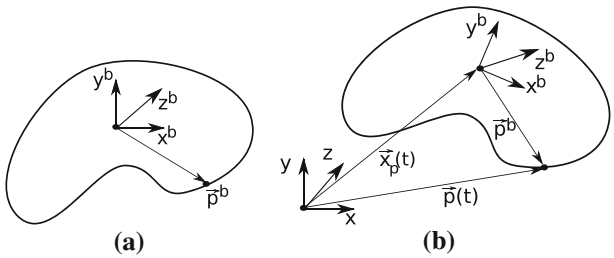

Fig. 1 The relation between body space (**a**) and world space (**b**). The fixed axes of body space, x_o, y_o and z_o are indicated in both figures. The position of a fixed point in body space, p_o, is transformed to world space, $p(t)$ (**a**) body space (**b**) world space

However, most of the simulations with DEM consider purely spherical particles. Although this may be an acceptable approximation for a number of applications, sometimes this is not the case. Modelling the behaviour of non-spherical particles can be done by: using a mathematical description of the surface (e.g. [11]), a finite element approach, describing a mesh of the surface (e.g. [28]) or by building a non-spherical particle from spheres (e.g. [27,35]). In all of these methods, the dynamic equations of the particles and the subsequent integration of these equations are significantly more complex than for spherical particles. The present paper focuses on developing a novel fast, efficient and accurate method to formulate and integrate the equations of rotation for non-spherical particles.

The translation and rotation of a rigid particle can be determined in two Cartesian coordinate frameworks: body space and world space. In body space, the origin of the Cartesian coordinates is fixed on the particle mass centre and the axis of the coordinates rotate along with the particle. This is often referred to as the Lagrangian framework. In world space, the coordinates are fixed in the origin of the initial Cartesian reference framework. This is often referred to as the Eulerian framework. These frameworks are depicted in Fig. 1. Although the translational motion of the particle can easily be converted between the two types of spaces, the rotational motion of the non-spherical particle is determined by a more complex rotational operator. There are various ways to formulate this operator. Among the numerous methods describing the rotation operator, the most commonly used frameworks are Euler angles, rotation matrices and unit Quaternions. All of these methods have some kind of limitation, which will be discussed in this paper.

In the literature, there are a number of papers describing the rotation of a rigid body in the framework of Hamilton dynamic systems (e.g [8,7,18,25,29]). Kosenko [25] reports a complex algorithm to represent the rotation of a free rigid body in the framework of Quaternions. Moreover, the Euler dynamic governing equations are also determined by Quaternion groups in the Hamiltonian framework. However, this algorithm is limited to the Euler case (i.e. free body dynamics), referring to a framework without the application of an external torque. For free body dynamics in Hamiltonian systems, the discrete Moser–Veselov (DMV) algorithm [31] is a very accurate numerical integrator, exactly conserving the kinetic energy and angular momentum, to update the rotation matrix, R. Hairer and Vilmart [18] and MacLaurgh and Zanna [29] have further improved the DMV algorithm by modifying the momentum of inertia at each integration and time transformation, respectively. Both improved DMV methods avoid singularity problems and increase the accuracy of the algorithms, to make them suitable for long-time integration. Moreover, Hairer and Vilmart [18] use unit Quaternions to represent the rotation matrix and transform the DMV method into the framework of Quaternions to simplify the implementation. However, these improved DMV algorithms are limited to free body dynamics. To account for the full dynamic rigid body problem, thus including the application of a torque working on the body, Celledoni et al. [8] propose a Stormer/Verlet splitting method to divide the rotation motion into two parts: the free rigid body kinetic part and the torque part, both in the Hamilton system. The free rigid body problem consists of Euler dynamic equations and a differential equation for updating the rotation matrix. The Euler equations can be determined by an exact method using the Jacobi elliptic function, and the approximate update of the rotation matrix or corresponding Quaternion is introduced for free body dynamics in [7]. The torque part is determined by a differential function involving the rotation matrix. This method is very expensive, and with the current computer power not suitable for simulations with a large number of rigid bodies.

For a fully dynamic system with a large number of rigid bodies under application of an external torque, the algorithms, which are mentioned above, are not feasible due to their very high computational cost. Moreover, lower-order algorithms for the fast integration of the rotation matrix are inaccurate, leading to inevitable singularity problems for long-time simulations. Moreover, Euler angles suffer from the so-called Gimbal lock problem when representing 3D rotation (e.g. [15]). To avoid these numerical difficulties, unit Quaternions are

commonly adopted to describe, at least part of, the rotation operator. Unit Quaternions, sometimes referred to as Euler parameters, are well known to represent rotation without singularity problems in molecular dynamics modelling and rigid particle modelling (e.g. [1,15,23,35]). The problem of the rotation matrix becoming singular occurs during the time integration of the rotation matrix. To prevent this, at least the time integration of the rotation operator is done by representing the rotation by a unit Quaternion. After the time integration, most approaches found in the literature convert the Quaternion to a rotation matrix (e.g. [1,6,23,30]). This rotation matrix is then used to transform the vector and tensor variables between the different frameworks, such as the vector $\boldsymbol{p}(t)$ in Fig. 1.

In the present work, unit Quaternions are applied not only to represent the time integration of the rotation operator, but also to replace rotation matrices completely and transform vector and tensor variables between different coordinate systems directly. Hence, there is no requirement for a rotation matrix in this new method. This saves both computational memory and effort. Accordingly, a novel method is derived to describe the transformation of vectors and tensors between frameworks based solely on Quaternions. The outline of the paper is as follows: section two briefly reviews rotation operators in general properties of Quaternions, and a novel model is derived to transform tensors between different coordinate frameworks. In addition, the equations relating the rotation matrix to the unit Quaternion are introduced. In section three, several algorithms which have been put forward in the literature to integrate unit Quaternions are scrutinised and discussed and a novel integration method is put forward. In Sect. 4, the algorithms are compared to each other by applying them to four different test cases, and the outcomes of this are discussed. Conclusions are drawn in the final section.

2 Rotation and Quaternions

2.1 Rotation dynamic equations

The equations of motion describing non-spherical rigid particles consist of translational and rotational components. The position of a particle can be represented equally simple in world space and in body space, but for the orientation of a particle, the rotational equations are significantly more complex in world space than body space. Therefore, the most common and convenient way is to compute rotational properties of particles in body space and, if required, transform them into world space. The governing equations of rotational motion compose of angular momentum equations and the differential equation of rotation operators. Firstly, the angular momentum $\boldsymbol{L}^b$ is defined by

$$\boldsymbol{L}^b = \mathsf{I}^b \boldsymbol{\omega}^b, \tag{1}$$

where the second-order tensor I^b is the constant moment of inertia in body space, and $\boldsymbol{\omega^b}$ represents the angular velocity of a particle. The superscript b means the variables are in the body space framework. For a particle undergoing the effect of an external torque, the torque, $\boldsymbol{\tau^b}$, is determined by

$$\boldsymbol{\tau}^b = \dot{\boldsymbol{L}}^b + \boldsymbol{\omega}^b \times \boldsymbol{L}^b, \tag{2}$$

where the time derivative of angular momentum is given as

$$\dot{\boldsymbol{L}}^b = \dot{\mathsf{I}}^b \boldsymbol{\omega}^b + \mathsf{I}^b \dot{\boldsymbol{\omega}}^b. \tag{3}$$

Because the moment of inertia tensor does not change in body space for a rigid particle, I^b is a constant and the first term on the right-hand side of the above equation is equal to zero. The angular acceleration can then be expressed as

$$\dot{\boldsymbol{\omega}}^b = \mathsf{I}^{b^{-1}} (\boldsymbol{\tau}^b - \boldsymbol{\omega}^b \times \mathsf{I}^b \boldsymbol{\omega}^b). \tag{4}$$

On the other hand, the differential equation of a rotation operator Q is defined by:

$$\dot{Q} = f(Q) Q, \tag{5}$$

where $f(Q)$ is a function involving the operator Q.

2.2 Rotation operators

There are three commonly used frameworks to describe rotation: Euler angles, rotation matrices and unit Quaternions, all of which have advantages and drawbacks (e.g. [14,16]). In addition, they can be translated from one framework to another. The details about the performance and issues of these rotation frameworks are also discussed in [12,14].

2.2.1 Euler angles

With the orientation of a solid particle fixed in body space, Euler angles (ϕ, θ, ψ) represent three composed axis rotations mapping the particle in body space to the world space. These angles involve a combination of sine and cosine functions, which are nonlinear. In addition, there are twelve ways to represent Euler angles [12]. Although only three independent quantities denote this operator, there are several drawbacks of Euler angles, such as the Gimbal lock problem, which gives rise to the loss of one degree of freedom when two of three axes are rotating into parallel configuration. This operator can only work well in applications involving one or two dimensional rotation only. For a general 3D rotation framework, the Gimbal lock problem will occur and the method is mostly not suitable.

2.2.2 Rotation matrix

In dynamics, a rotation matrix is a 3×3 orthogonal matrix performing a rotational motion in three dimensions with a determinant of 1. A vector $\boldsymbol{v}$ transforms from one coordinate system to another by application of the rotation matrix R as:

$$\boldsymbol{v}' = \mathsf{R}\boldsymbol{v}, \tag{6}$$

where the vector $\boldsymbol{v}'$ represents the rotated counterpart of the vector $\boldsymbol{v}$. A second-order tensor k is transformed by the rotation matrix as

$$\mathsf{k}' = \mathsf{R}\,\mathsf{k}\,\mathsf{R}^T. \tag{7}$$

A rotation matrix is by definition an orthogonal matrix, the columns of which are of unit length. During the integration or differentiation of the rotation matrix, six constraints are required, as the three degrees of freedom associated with the rotation are described by nine components. There are three constraints for maintaining the unit length of the rotation matrix columns and a further three constraints for keeping them orthogonal to each other. When these six constraints are not met implicitly by the numerical integration or differentiation, singularity problems may arise, making the required inversion difficult or impossible. In order to avoid these problems, unit Quaternions can be applied instead.

2.2.3 Unit Quaternions

Due to the absence of singularity and Gimbal lock problems, unit Quaternions are increasingly popular to represent rotation. General Quaternions do not only change the orientation of a vector, but also scale the length of a vector. Therefore, the equation for representing rotation cannot be a simple Quaternion multiplication, as the length of the vector will, generally, change. To represent rotations by Quaternions, the length of the Quaternions must be exactly unity. Rotation without scaling is performed by unit Quaternions, see e.g. [13,20],

2.3 Quaternions

Quaternions were first introduced by Sir Hamilton [17,19] in the nineteenth century and have been widely used to represent rotation for modelling dynamic systems in the past decades. They are expressed in a complex number system, consisting of a scalar part and a vector part. Hence, there are a total of 4 unknowns. In dynamics, the physical meaning of a Quaternion is to scale the length and change the orientation of a vector [22]. A Quaternion is defined by:

$$q = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}, \tag{8}$$

where q_0, q_1, q_2 and q_3 are real numbers, and $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ are unit vectors directed along to the x, y and z axis, respectively. Quaternions can also be written as a real number and a vector:

$$q = [q_0,\ \boldsymbol{q}]. \tag{9}$$

Three useful operations of a Quaternion itself can be defined: conjugation, norm and inverse. The conjugate of a Quaternion is defined as

$$q^* = q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k}, \tag{10}$$

the norm of a Quaternion is determined by

$$\|q\| = \sqrt{\|q\|^2} = \sqrt{qq^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}, \tag{11}$$

and the inverse of a Quaternion is given as

$$q^{-1} = \frac{q^*}{\|q\|}. \tag{12}$$

A unit Quaternion is a Quaternion of norm 1; then, the inverse Quaternion is equal to the conjugate Quaternion:

$$\|q\| = \sqrt{\|q\|^2} = 1 \ \rightarrow \ q^* = q^{-1}. \tag{13}$$

2.3.1 Multiplication of Quaternions

The multiplication between two Quaternions represents the subsequent application of each Quaternion. This product is often referred to as the Grassman product [3,23]. In vector representation, the product of Quaternions p and q is given as Quaternion t:

$$t = pq = [p_0q_0 - \boldsymbol{pq},\ p_0\boldsymbol{q} + q_0\boldsymbol{p} + \boldsymbol{p} \times \boldsymbol{q}] \tag{14}$$

The equation consists of vector dot and cross products. Due to the anti-commutative property of the cross product, the multiplication of Quaternions is not commutative. Quaternion multiplication can also be represented by matrix multiplication:

$$t = \mathsf{Q}(p)q = \begin{pmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & p_3 & -p_2 \\ p_2 & -p_3 & p_0 & p_1 \\ p_3 & p_2 & -p_1 & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}. \tag{15}$$

More detail about Quaternion algebra is introduced in [2,22,26].

2.4 Rotation by unit Quaternion

A vector $\boldsymbol{s}$ rotated by a pair of unit Quaternions is defined by:

$$\boldsymbol{s}' = q\boldsymbol{s}q^{-1}, \tag{16}$$

where q is a unit Quaternion, q^{-1} represents the conjugation of q, and the vector $\boldsymbol{s}$ is interpreted as a Quaternion as $s = [0,\ \boldsymbol{s}]$ with the scalar part equal to zero. The unit Quaternion q can be directly expressed in a form containing the vector around which the rotation takes place and the angle of the rotation [5,23]:

$$q = \cos\frac{\alpha}{2} + \sin\frac{\alpha}{2}\hat{\boldsymbol{q}}, \tag{17}$$

where $\hat{\boldsymbol{q}}$ is the normalised vector around which the rotation takes place and the angle α indicates the rotational angle. In the unit Quaternion q, the coefficients q_0, q_1, q_2 and q_3 are sometimes referred to as Euler parameters (e.g. [5]), which are not independent of each other, as they must always satisfy

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1. \tag{18}$$

Many integration algorithms do not inherently respect this constraint and explicitly re-normalise the Quaternion after the algorithms are applied, by defining the corrected Quaternion as

$$\hat{q} = \frac{q}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}}. \tag{19}$$

This is, however, not the same as inherently embedding the unit length of the Quaternion, as expressed by Eq. (18), into the algorithm itself. Applying Eq. (19) modifies the relation between the four parameters of the Quaternion, therefore modifying the rotation it represents.

In most research papers dealing with the numerical integration of non-spherical particles with Quaternions, the time integration of the rotation operator is addressed by employing unit Quaternions. After the time integration itself, the corresponding rotation matrices are determined from the Quaternion and used to determine the orientation of the particles and transform vector and tensor properties between body space and world space by Eqs. (6) and (7). This requires an inverse relationship between rotation matrices and unit Quaternions. The rotation matrix corresponding to the unit Quaternion is given by the Quaternion components as

$$\mathsf{R} = \begin{pmatrix} 1 - 2(q_2^2 + q_3^2) & 2q_1q_2 - 2q_0q_3 & 2q_0q_2 + 2q_1q_3 \\ 2q_1q_2 + 2q_0q_3 & 1 - 2(q_1^2 + q_3^2) & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_0q_1 + 2q_2q_3 & 1 - 2(q_1^2 + q_2^2) \end{pmatrix}. \tag{20}$$

Equation (6), expressing the transformation of a vector by a rotation matrix, is equivalent to Eq. (16),

$$\boldsymbol{v}' = q\,\boldsymbol{v}\,q^{-1} = \mathsf{R}\boldsymbol{v}. \tag{21}$$

Some second-order tensor variables are also required to transform between the two different coordinate systems, such as the inertia tensor and resistance tensors [30]. These tensors can be transformed by applying Eq. (7). However, there exists no equivalent equation using unit Quaternions to perform the same transformation in the literature. All methods presented so far determine the rotation matrix from the Quaternion and apply the rotation matrix subsequently, as expressed in Eq. (7) to compute the tensors in the rotated framework.

In this paper, a new model is derived for determining a tensor in the rotated framework directly by unit Quaternions without the necessity of determining the corresponding rotation matrix. To determine a second-order tensor in the rotated framework, using Eq. (7),

$$\mathsf{I}' = \mathsf{R}\,\mathsf{I}\,\mathsf{R}^T,$$

where the tensors I' and I can be considered as three sequential column vectors $\boldsymbol{I}_1$, $\boldsymbol{I}_2$ and $\boldsymbol{I}_3$ as:

$$\mathsf{I} = \left(\begin{pmatrix} I_{11} \\ I_{21} \\ I_{31} \end{pmatrix} \begin{pmatrix} I_{12} \\ I_{22} \\ I_{32} \end{pmatrix} \begin{pmatrix} I_{13} \\ I_{23} \\ I_{33} \end{pmatrix} \right) = \begin{pmatrix} \boldsymbol{I}_1 & \boldsymbol{I}_2 & \boldsymbol{I}_3 \end{pmatrix}.$$

A tensor I'' can be expressed as

$$\mathsf{I}'' = \mathsf{R}\,\mathsf{I}, \tag{22}$$

where I'' can be considered as transforming the three sequential column vectors $(\boldsymbol{I}_1\ \boldsymbol{I}_2\ \boldsymbol{I}_3)$ by a rotation matrix, so a corresponding unit Quaternion can replace the rotation matrix,

$$\begin{aligned} \boldsymbol{I}_1'' &= q\boldsymbol{I}_1q^{-1}, \\ \boldsymbol{I}_2'' &= q\boldsymbol{I}_2q^{-1}, \\ \boldsymbol{I}_3'' &= q\boldsymbol{I}_3q^{-1}. \end{aligned} \tag{23}$$

Combining these three equations, a new expression is determined as

$$\mathsf{I}'' = q\mathsf{I}q^{-1}, \tag{24}$$

which is equivalent to Eq. (22). Equation (7) can be interpreted as the tensor I'', as defined in Eq. (22):

$$\mathsf{I}' = \mathsf{R}\,\mathsf{I}\,\mathsf{R}^T = \mathsf{I}''R^T. \tag{25}$$

The transpose of the tensor I' is represented by

$$\mathsf{I}'^T = \mathsf{R}\,\mathsf{I}''^T, \tag{26}$$

in which a unit Quaternion q can replace the rotation matrix by applying Eq. (24),

$$\mathsf{I}'^T = q\mathsf{I}''^T q^{-1}, \tag{27}$$

and given $\mathsf{I}''^T = (q\mathsf{I}q^{-1})^T$,

$$\mathsf{I}'^T = q(q\mathsf{I}q^{-1})^T q^{-1}. \tag{28}$$

Finally, the transformation of second-order tensors by unit Quaternions is expressed as

$$\mathsf{I}' = (q(q\mathsf{I}q^{-1})^T q^{-1})^T. \tag{29}$$

Following the above analysis, unit Quaternions can be used to transform vector properties during rotation, but also to transform tensor properties directly. Accordingly, rotation matrices can be completely replaced by corresponding unit Quaternions only, and the rotation matrix is no longer required. This will save a significant amount of computer memory (4 instead of 9 floating point numbers per particle), and increase the accuracy introduced by round-off errors, as fewer operations are required.

3 Numerical integration of unit Quaternions

The time derivative of a unit Quaternion q has a very simple form. It is determined by the angular velocity vector $\boldsymbol{\omega}$ and the Quaternion itself, see for instance [4,9,34]:

$$\dot{q} = \frac{1}{2}\omega q = \begin{pmatrix} 0 & \frac{-\omega_x}{2} & \frac{-\omega_y}{2} & \frac{\omega_z}{2} \\ \frac{\omega_x}{2} & 0 & \frac{\omega_z}{2} & \frac{-\omega_y}{2} \\ \frac{\omega_y}{2} & \frac{-\omega_z}{2} & 0 & \frac{\omega_x}{2} \\ \frac{\omega_z}{2} & \frac{\omega_y}{2} & \frac{-\omega_x}{2} & 0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}, \tag{30}$$

where the angular velocity ω is considered as a Quaternion, $[0, \boldsymbol{\omega}]$. In the past decades, several algorithms have been put forward to integrate unit Quaternions to represent rotation. Some algorithms are based on Taylor expansion of the unit Quaternion (e.g. [1,15]). In these algorithms, addition and subtraction operators are required to add or subtract the derivatives of the Quaternion. However, if the sum of additions and subtractions is not exactly of zero length, an increase in the length of the integrated Quaternion is obtained. Therefore, to preserve the constraint of a unit Quaternion when applying such a method, the new Quaternion requires re-normalisation after each integration time step. The re-normalisation procedure does not only enforce the constraint concerning unit length, but it also affects the relationship between the four Euler parameters. In order to integrate a unit Quaternion without the application of addition or subtraction operators between Quaternion derivatives, a patent was filed by [36]. This patent describes an algorithm to update a unit Quaternion. A similar methodology is also derived by, e.g., [5,23,34], and is described further below.

3.1 Previously proposed methods based on Taylor polynomial expansion

In many other research papers, such as [1,4,15,24,30,33,35], the integration of a unit Quaternion is solved based on Taylor series expansion, and Eq. (30), or an equivalent form, is adopted. The Euler method is the most simple algorithm, whereas higher-order methods based on the Taylor series expansion have also been put forward and are discussed below.

3.1.1 Euler method

The Euler method is the most simple method for integrating unit Quaternions. The Quaternion at the new time step, q_{n+1}, is approximated by

$$q_{n+1} = q_n + \frac{1}{2}\omega_n q_n \delta t + O(\delta t^2). \tag{31}$$

Here, q_{n+1} is determined by a first-order Taylor expansion. Due to the increment $\frac{1}{2}\omega_n q_n \delta t$ added to the unit Quaternion q_n, the length of q_{n+1} will be different from unity. In order to preserve the unity constraint, the Quaternion q_{n+1} requires re-normalisation at each time step:

$$\hat{q}_{n+1} = \frac{q_{n+1}}{\|q_{n+1}\|}. \tag{32}$$

However, the re-normalisation procedure gives rise to numerical errors, adding to the relatively large truncation errors of the first-order Euler method. To decrease the errors in the Euler method, several algorithms which are also based on Taylor series expansion are discussed below, such as the leap-frog method [1,35], the second-order Taylor series expansion, the second-order Adams–Bashforth method [30], the Runge–Kutta method [32] and the scalar factor method introduced in [24].

3.1.2 Leap-frog method

The leap-frog, or mid-point, method for the time integration of unit Quaternions is outlined by Walton and Braun [35]; unit Quaternions and angular velocities at the mid-point between two adjacent time steps are evaluated in this algorithm. It is expressed as

$$q_{n+1} = q_n + \frac{1}{2}\omega_{n+\frac{1}{2}} q_{n+\frac{1}{2}} \delta t, \tag{33}$$

where the angular velocity $\boldsymbol{\omega}$, expressed above as Quaternion ω, at time level $n + \frac{1}{2}$ is explicitly determined by the leap-frog method. On the other hand, $q_{n+\frac{1}{2}}$ is simply expressed by:

$$q_{n+\frac{1}{2}} = \frac{q_n + q_{n+1}}{2}. \tag{34}$$

Due to the appearance of q_{n+1} on both sides in Eq. (33), this leap-frog algorithm is significantly more complex and requires more variables and operations than the Euler method. However, because of the application of Quaternion addition operators, the algorithm still causes the length of the integrated Quaternion, q_{n+1}, to exceed unity. Therefore, the Quaternions need to be re-normalised at each time step.

In this paper, this algorithm for Quaternion integration is reformulated and written it in terms of Quaternion multiplication only. This novel formulation prevents the application of addition or subtraction to the Quaternion. To illustrate the algorithm, two new Quaternions (β and $\tilde{q}$) are defined as

$$\tilde{q} = f\beta\beta, \qquad \beta = [1,\ \beta_x, \beta_y,\ \beta_z], \tag{35}$$

where the components of β are given by

$$\begin{aligned} \beta_x &= \frac{\delta t}{4}\omega_x^{n+\frac{1}{2}}, \\ \beta_y &= \frac{\delta t}{4}\omega_y^{n+\frac{1}{2}}, \\ \beta_z &= \frac{\delta t}{4}\omega_z^{n+\frac{1}{2}}. \end{aligned} \tag{36}$$

The scale factor f is defined as

$$f = \frac{\|\beta\|^2}{A}, \tag{37}$$

where A is determined by the components of β,

$$A = 1 + 2\beta_x^2 + 2\beta_y^2 + 2\beta_z^2. \tag{38}$$

The unit Quaternion at the next time level is determined by

$$q_{n+1} = \tilde{q} q_n. \tag{39}$$

The unit Quaternion q_{n+1} at the new time level also requires re-normalisation at each time step. The above algorithm is the Quaternion counterpart of Eq. (33), and the results are very similar. Furthermore, the relation between the unit Quaternions q_n and q_{n+1} is not linear. Therefore, Eq. (34), approximating $q_{n+\frac{1}{2}}$ in a linear fashion, is not appropriate and inaccurate. Some higher-order methods based on the same idea are proposed by other researchers, where multiple evaluations of the Quaternion between time levels n and $n+1$ are required, such as Runge–Kutta or Adams–Bashforth methods [30]. However, there is in principle no direct physical meaning to the addition or subtraction of Quaternions. Moreover, the higher-order methods based on this principle cannot prevent the length of the Quaternion deviating from unity; the more addition or subtraction operators in higher-order methods can even influence the relationship between q_n and q_{n+1}. Hence, all these algorithms require re-normalisation at each time step, giving rise to numerical errors.

3.1.3 Scalar factor method

Kleppmann [24] concludes that the Euler method and other higher-order methods as outlined above do not properly integrate unit Quaternions and that the required re-normalisation procedure introduces significant errors. In [24], a scalar factor is introduced in the derivative equation directly,

$$\dot{q}_n = f \omega_n q_n, \tag{40}$$

where f is defined as

$$f(\delta t, \|\omega\|) = \frac{1}{\|\omega\| \delta t} \tan\left(\frac{\|\omega\| \delta t}{2}\right). \tag{41}$$

The unit Quaternion q_{n+1} at the next time step is then determined by the Euler method as

$$q_{n+1} = q_n + \dot{q}_n \delta t, \tag{42}$$

where δt represents the size of the time step. Writing out $\dot{q}_n \delta t$ in the above equation gives

$$\dot{q}_n \delta t = f \omega q \delta t = \tan(\|\delta q\|) \frac{\delta q}{\|\delta q\|}, \tag{43}$$

where δq is

$$\delta q = \frac{\delta t}{2} \omega q. \tag{44}$$

Finally, after the re-normalisation procedure, q_{n+1} is expressed as

$$q_{n+1} = \left[q_n + \tan(\|\delta q\|) \frac{\delta q}{\|\delta q\|} \right] \cos\left(\|\delta q\|\right). \tag{45}$$

As there are discontinuities in the tan function, there is a possibility that numerical instabilities occur when $\|\delta q\|$ is equal to $\frac{\pi}{2}$ during a single time step.

3.2 Direct multiplication method

In order to derive a better integration method of unit Quaternions for representing rotation, a patent was developed by Whitemore [36], and an analogous derivation is presented in [5,23,34]. In these methods, the Quaternion multiplication replaces the addition operator in the integral equation altogether. The derivative of unit Quaternion q is introduced by Eq. (30),

$$\dot{q} = \frac{1}{2}\omega q,$$

and an exponential map is introduced, given by

$$T(\omega, \delta t) = \exp(\Omega \delta t), \tag{46}$$

where

$$\Omega = \begin{pmatrix} 0 & \frac{-\omega_x}{2} & \frac{-\omega_y}{2} & \frac{\omega_z}{2} \\ \frac{\omega_x}{2} & 0 & \frac{\omega_z}{2} & \frac{-\omega_y}{2} \\ \frac{\omega_y}{2} & \frac{-\omega_z}{2} & 0 & \frac{\omega_x}{2} \\ \frac{\omega_z}{2} & \frac{\omega_y}{2} & \frac{-\omega_x}{2} & 0 \end{pmatrix}.$$

The Quaternion at the next time level, q_{n+1}, is represented as

$$q_{n+1} = T(\Omega, \delta t) q_n. \tag{47}$$

The exponential term in the above equation is expanded in a Maclaurin series and a simplified form is obtained,

$$T(\omega, \delta t) = \left(\cos \frac{\|\boldsymbol{\omega}\|}{2} \mathsf{I} + \frac{2}{\|\boldsymbol{\omega}\|} \sin \frac{\|\boldsymbol{\omega}\|}{2} \Omega \right) \delta t, \tag{48}$$

where I is the fourth-order identity matrix, and the vector $\boldsymbol{\omega}$ is the angular velocity. The matrix T in the above equation can be also represented as a unit Quaternion,

$$\tilde{q} = \left[\cos \frac{\|\boldsymbol{\omega}\| \delta t}{2}, \sin \frac{\|\boldsymbol{\omega}\| \delta t}{2} \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|} \right]. \tag{49}$$

Finally, the unit Quaternion at time level $n + 1$ is expressed as

$$q_{n+1} = \tilde{q}_n q_n. \tag{50}$$

In this method, there is no necessity for an addition or subtraction operator. An exponential map is employed to approximate the increment of a unit Quaternion. Theoretically, the multiplication between unit Quaternions can preserve their unit length.

A similar method as outlined above can also be put forward, by starting from the sequential rotation as given by Eq. (16),

$$\begin{aligned} \boldsymbol{s}_1 &= \tilde{q}_0 \boldsymbol{s}_0 \tilde{q}_0^{-1}, \\ \boldsymbol{s}_2 &= \tilde{q}_1 \boldsymbol{s}_1 \tilde{q}_1^{-1}, \\ \boldsymbol{s}_3 &= \tilde{q}_2 \boldsymbol{s}_2 \tilde{q}_2^{-1}, \\ &\cdots\cdots \\ \boldsymbol{s}_{n+1} &= \tilde{q}_n \boldsymbol{s}_n \tilde{q}_n^{-1}, \end{aligned}$$

where the unit Quaternion $\tilde{q}_i$ $(i = 1, 2, 3 \ldots n)$ represents rotation within a time step and is defined analogously to Eq. (49),

$$\tilde{q}_n = \left[\cos \frac{\|\boldsymbol{\omega}_n\| \delta t}{2}, \sin \frac{\|\boldsymbol{\omega}_n\| \delta t}{2} \frac{\boldsymbol{\omega}_n}{\|\boldsymbol{\omega}_n\|} \right],$$

where the rotation angle α in a time step is determined by the length of the angular velocity at that time step and the time step as $\|\boldsymbol{\omega}_n\|\delta t$, and the direction of rotation is the same as the direction of the angular velocity $\boldsymbol{\omega}_n$. The unit Quaternion q_n, which represents a vector rotated from original position at $t = 0$ to time level n, is defined by

$$q_n = \prod_{i=1}^{n} \tilde{q}_{n-i}. \tag{51}$$

Finally, the unit Quaternion for the next time level q_{n+1} is described by

$$q_{n+1} = \tilde{q}_n q_n, \tag{52}$$

which is the same as defined in Eq. (50).

3.3 Newly proposed predictor-corrector direct multiplication method

The novel method put forward in this paper approximates the angular velocity with a basic Lie–Euler method, or called predictor-corrector method, which is outlined in Allen and Tildesley [1]. However, the Quaternion integration method in [1] is directly based on Taylor series, so that

$$q_{n+\frac{1}{2}} = q_n + \frac{1}{2}\dot{q}_n \delta t, \tag{53}$$

$$q_{n+1} = q_n + \dot{q}_{n+\frac{1}{2}} \delta t. \tag{54}$$

As mentioned previously in this section, the addition and subtraction operators appearing in Quaternion integration equations are physically meaningless and can give rise to numerical errors. In this paper, we propose a new algorithm, the predictor-corrector direct multiplication (PCDM) method, which is not based on Taylor series, but applies the direct multiplication algorithm. Firstly, the variables that describe the rational motion of a particle are transformed into body space from world space at current time level n:

$$\boldsymbol{\omega}_n^b = q_n^{-1} \boldsymbol{\omega}_n q_n, \tag{55}$$

$$\boldsymbol{\tau}_n^b = q_n^{-1} \boldsymbol{\tau}_n q_n. \tag{56}$$

The angular velocity expressed in body space at the mid-point of the next time level, $\boldsymbol{\omega}^b_{n+\frac{1}{2}}$, and at a quarter of the next time level, $\boldsymbol{\omega^b}_{n+\frac{1}{4}}$, are determined by

$$\begin{aligned} \boldsymbol{\omega}^b_{n+\frac{1}{4}} &= \boldsymbol{\omega}^b_n + \frac{1}{4}\dot{\boldsymbol{\omega}}^b_n \delta t, \\ \boldsymbol{\omega}^b_{n+\frac{1}{2}} &= \boldsymbol{\omega}^b_n + \frac{1}{2}\dot{\boldsymbol{\omega}}^b_n \delta t, \end{aligned} \tag{57}$$

where the angular acceleration $\dot{\boldsymbol{\omega}}^b_n$ is determined by Eq. (4). In Eq. (49), the application of $\boldsymbol{\omega}$ represents the spatial angular velocity of a particle within a time step in world space. In most algorithms, the angular velocity is defined at the mid-point in world space, $\boldsymbol{\omega}_{n+\frac{1}{2}}$, which is transformed to world space by application of the unit Quaternion q_n, as the value of $q_{n+\frac{1}{2}}$ is unknown at that time level. Hence, the time level of the Quaternion is not respected in these algorithms. To decrease the error caused by the mixing of time levels, this paper uses the predictor-corrector method to approximate the angular velocity at the mid-point. Firstly, the predicted angular velocity at a quarter at next time level in world space, $\boldsymbol{\omega}_{n+\frac{1}{4}}$, is approximated based on the unit Quaternion q_n:

$$\boldsymbol{\omega}_{n+\frac{1}{4}} = q_n \boldsymbol{\omega}^b_{n+\frac{1}{4}} q_n^{-1}. \tag{58}$$

Then, a *prediction* of the unit Quaternion at the half time interval, $q'_{n+\frac{1}{2}}$, is determined by the velocity $\boldsymbol{\omega}_{n+\frac{1}{4}}$. The prime on the variable emphasises that it concerns the initial prediction of the variable, not its final value:

$$q'_{n+\frac{1}{2}} = \left[\cos\frac{\|\boldsymbol{\omega}_{n+\frac{1}{4}}\|\delta t}{4}, \sin\frac{\|\boldsymbol{\omega}_{n+\frac{1}{4}}\|\delta t}{4} \frac{\boldsymbol{\omega}_{n+\frac{1}{4}}}{\|\boldsymbol{\omega}_{n+\frac{1}{4}}\|} \right] q_n. \tag{59}$$

Using this predicted unit Quaternion $q'_{n+\frac{1}{2}}$, the angular acceleration $\dot{\boldsymbol{\omega}}^b_{n+\frac{1}{2}}$ is determined by application of Eq. (4), and the angular velocity $\boldsymbol{\omega}_{n+\frac{1}{2}}$ at mid-point of next time level in world space is determined by

$$\boldsymbol{\omega}_{n+\frac{1}{2}} = q'_{n+\frac{1}{2}} \boldsymbol{\omega}^b_{n+\frac{1}{2}} q'^{-1}_{n+\frac{1}{2}}. \tag{60}$$

Then, the corrected unit Quaternion q_{n+1} at the new time level is

$$q_{n+1} = \left[\cos \frac{\|\boldsymbol{\omega}_{n+\frac{1}{2}}\| \delta t}{2}, \sin \frac{\|\boldsymbol{\omega}_{n+\frac{1}{2}}\| \delta t}{2} \frac{\boldsymbol{\omega}_{n+\frac{1}{2}}}{\|\boldsymbol{\omega}_{n+\frac{1}{2}}\|} \right] q_n. \tag{61}$$

Finally, the angular velocity in body space at the new time level can be determined and transformed to the angular velocity in world space,

$$\boldsymbol{\omega}^b_{n+1} = \boldsymbol{\omega}^b_n + \dot{\boldsymbol{\omega}}^b_{n+\frac{1}{2}} \delta t, \tag{62}$$

$$\boldsymbol{\omega}_{n+1} = q_{n+1} \boldsymbol{\omega}^b_{n+1} q^{-1}_{n+1}. \tag{63}$$

The method as outlined above presents a consistent and accurate predictor-corrector direct multiplication (PCDM) method to determine the unit Quaternion representing the orientation of a non-spherical particle and its angular velocity. Moreover, this method does not use a rotation matrix and does not mix time levels inconsistently in its final correction. In the next section, the different methods as discussed and derived above will be compared to each other in a number of realistic conditions.

4 Comparison of methods and discussion

The various methods that have been discussed in this paper in Sect. 3.1 are compared with the novel PCDM method as outlined in Sect. 3.3, Eqs. (55) to (63). The comparison is done considering two criteria: energy conservation and the rate of convergence. Each of these criteria is used to analyse four different test cases, each representing a realistic problem involving the behaviour of non-spherical particles.

In the modelling of non-spherical particles, both translation and rotation need to be considered. The translation of the particles is determined by a velocity Verlet scheme [1] for all methods and test cases.

4.1 Energy conservation

One of the most important features of a numerical framework for solving the equations governing the behaviour of particles is that it conserves total energy. If the behaviour of non-spherical particles is determined in a gravity field without any source of dissipation, the total mechanical energy (E_{tot}) of the particles in this system is represented by

$$E_{\text{tot}} = \sum_{i=1}^{N} (m_i g \delta h_i + \frac{1}{2} m_i \boldsymbol{v}_i^2 + \frac{1}{2} \boldsymbol{\omega} \cdot \mathsf{I}_i \boldsymbol{\omega}_i) = \text{constant}, \tag{64}$$

where N is the total number of particles, and δh_i represents the height of the mass centre of the ith particle.

All the collisions are determined with zero coefficient of friction, meaning there are no kinetic energy losses caused by friction. Moreover, the value of the coefficient of restitution is unity, meaning there is no translational kinetic energy loss during a collision between particles themselves or particles and boundary walls. Therefore, a collision does not change the total kinetic energy. The error in energy conservation can then be expressed as

$$E_{\text{error}} = \left| \frac{E^n_{\text{tot}} - E^0_{\text{tot}}}{E^0_{\text{tot}}} \right| \times 100\,\%, \tag{65}$$

where E^n_{tot} represents the total energy of the particles after the nth integration time step. E^0_{tot} represents the initial total energy of the particles.

Table 1 The properties of cylindrical fibres and the boundary wall of the box

Fibre	D [m]	E [Pa]	ν [−]	μ [−]	e [−]
	0.01	5.0×10^7	0.35	0.0	1.0
Walls	Dimension [m^3]	E [Pa]	ν	μ	e
	$1 \times 1 \times 1$	5.0×10^7	0.23	0.0	1.0

The diameter (D), Young's modulus (E), Poisson's ratio (ν), coefficient of friction (μ) and coefficient of restitution (e)

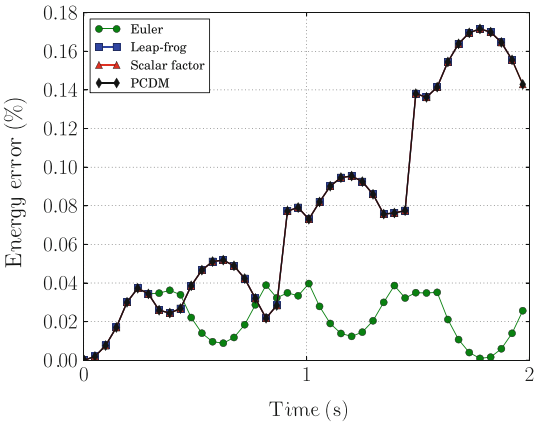

Fig. 2 The error in energy resulting from the four different rotational integration algorithms for test case 1: a single falling fibre

4.1.1 Comparison with three test cases

The first two test cases used to compare the integration methods consider a cylindrical fibre with an aspect ratio, defined as $r = \frac{b}{D}$, of 3, and a density and a volume of 1.1×10^3 kg/m^{3}and 1.961×10^{-6}m^3, respectively. The simulations of the first two test cases are carried out in a computational domain of a unit cube ($1 \times 1 \times 1$). The boundaries of this box are considered as frictionless, rigid walls. The properties of the fibre and the computational domain are presented in Table 1. In the domain, forces and torques acting on each particle are caused by gravity (body force) and by collisions between particles themselves or by the particle and boundary walls. A Hertzian collision model is applied to approximate the forces and torques acting on the particle during a collision.

First test case: single falling fibre. In the first test, a single elongated fibre, initially placed exactly in the middle of the box with a 30 degree angle between the principle axis of the fibre and the plane of the bottom wall, falls down under the effect of gravity. Due to its initial angle, a torque acts on the particle during the first collision with the bottom wall and the particle starts to rotate.

The results of the four different numerical methods as previously discussed in this paper are compared to each other in Fig. 2, showing the total energy error as a function of time for the various integration methods. These methods are the Euler method, the leap-frog method, the scalar factor method and the newly proposed PCDM method.

In Fig. 2, the errors of the different methods are shown for this test case over two seconds of simulation with a fixed time step of $\Delta t = 1.0^{-7}$s.

The Euler method predicts the minimum error in kinetic energy, very close to zero and non-increasing, whereas the errors in energy produced by all the other integration algorithms are small, typically less than 0.2 %, but are larger and increasing. Also, the errors produced by the other 3 integration algorithms almost overlap each other. Although the errors from other algorithms are somewhat bigger than those from the Euler method, all the algorithms can be considered acceptable for this case.

Second test case: nine falling fibres. A second test case comprises of 9 fibres in the same box, falling under gravity. In order to increase the number of collisions within the total period of two seconds, all the nine particles are given an initial velocity of $(1.0, \ -1.0, \ 0.0)$. Their initial positions are distributed evenly on a vertical plane halfway through the box. The angle between the principle axis of the fibre and the plane of the bottom wall is

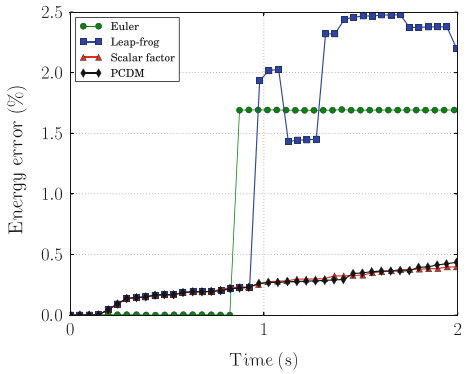

Fig. 3 The error in total energy resulting from the four different rotational integration algorithms in the case of 9 falling fibres

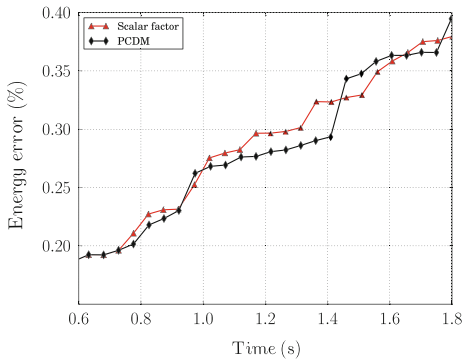

Fig. 4 The error in total energy resulting from the scalar factor method and the novel PCDM method in the case of 9 falling fibres

75 degrees. Compared to the previous test case, which only involves particle–wall collisions, also numerous particle–particle collisions occur.

Figure 3 shows the errors of energy conservation for this case, comparing the four integration algorithms as outlined previously. The errors for only the scalar factor method and the newly proposed PCDM method are shown in Fig. 4. For this second test case, the Euler and leap-frog methods become unstable and the absolute errors rapidly exceed 1.5 and 2 %, respectively. On the other hand, the errors produced by the scalar factor algorithm and the novel PCDM method are steady and very close to each other. The difference in the plot between the two algorithms indicates that the particles have different trajectories and orientations. Figure 5 shows the snapshot of one particle trajectory and the locations of the fibres for: (a) the novel PCDM method and (b) the scalar factor method. This figure shows the trajectories predicted by both methods as well as the dynamics of the trajectories are significantly different. As very small errors produced in each time step will dramatically change the trajectory and orientation of each particle, different algorithms cannot produce exactly the same trajectories over time. This phenomenon is referred to as the Lyapunov instability [21]. From this second test case, it is not possible to distinguish between the scalar factor method and the PCDM method. Although the trajectories and final positions differ significantly, the energy conservation error is comparable for both methods, and the case is too complex to compare with an analytical solution.

Third test case: prescribed one-dimensional torque on a particle The third test case considers the rotation of a particle by prescribing its torque. The particle considered is a sphere with a diameter and density of 2 m and 1,100 kg/m^3, respectively. To precisely follow the rotation of this particle, a unit vector, initially $x = (1.0,\ 0.0,\ 0.0)$, is projected onto the sphere. The function prescribing the torque is given by $\tau = (0.0,\ A\ \exp(Ct),\ 0.0)$, in which A and C are 1×10^5 and 1, respectively, and t represents time. This test case has only been performed with the scalar factor method and the PCDM method, as all other methods do not achieve convergence. This test case can be evaluated analytically, by integrating the torque function with respect to time. A direct comparison between the different algorithms can thus be made. The results of these two methods are presented in Table 2, which shows the orientation of the unit vector after one second

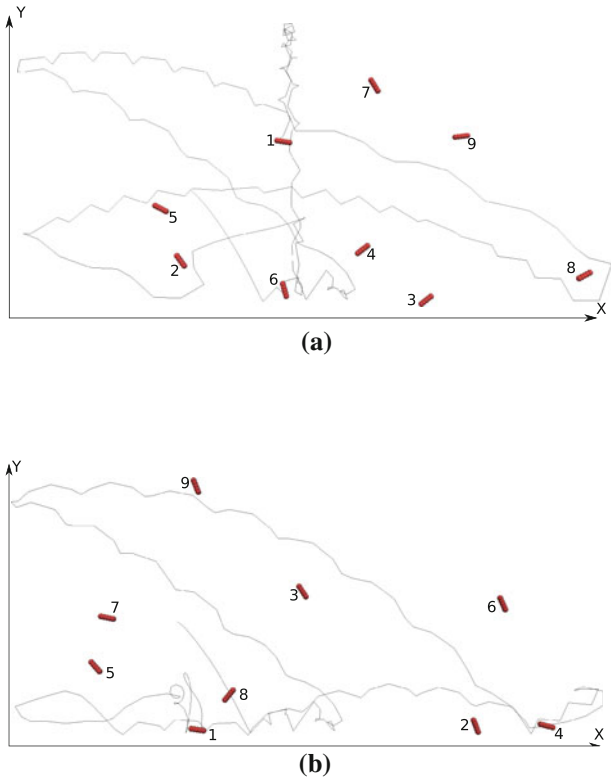

Fig. 5 Different particle trajectories as predicted by: **a** the novel PCDM method, and **b** the scalar factor method for the case of 9 falling fibres **a** one particle trajectory and nine particle positions at the end of 2 second simulation by the novel PCDM method. **b** one particle trajectory and nine particle positions at the end of 2 second simulation by scalar factor method

Table 2 The orientation of a unit vector $x = (1.0,\ 0.0,\ 0.0)$ after a second simulation with a nonlinear torque by scalar factor method and the novel PCDM method

	Position at time $t = 1.0$	Position error
Theoretical result	$(2.934253 \times 10^{-1},\ 0.0,\ -9.55982 \times 10^{-1})$	0.0
Scalar factor method	$(2.871152 \times 10^{-1},\ 0.0,\ -9.578961 \times 10^{-1})$	2.15 %
PCDM method	$(2.915775 \times 10^{-1},\ 0.0,\ -9.565472 \times 10^{-1})$	0.63 %

of physical time with a constant time step $\triangle t = 1 \times 10^{-4}$s, whereas the error in angle with time is shown in Fig. 6. The results from the newly proposed PCDM method are significantly closer to the analytical result than the results obtained from the scalar factor method. Figure 6 shows the error in the angle of the unit vector as a function of time, comparing the analytical result with the PCDM method and the scalar factor method. This figure clearly shows the effect of the re-normalisation which is required by the scalar factor method; when re-normalisation is applied, the error in angle increases rapidly.

4.2 Rate of convergence

A very important aspect of a numerical integration method is the rate of convergence. For many types of engineering problems, millions of particles will be studied and a favourable integration algorithm should quickly converge as the time step decreases. In this section, the rate of convergence of the most common methods is analysed.

A numerical integration algorithm for unit Quaternions will never diverge, as unit Quaternions are by definition of unit length. In other words, the error of the rotational angles cannot diverge, and the range of angle errors is from 0 to π rad. Hence, this makes the analysis somewhat more complicated.

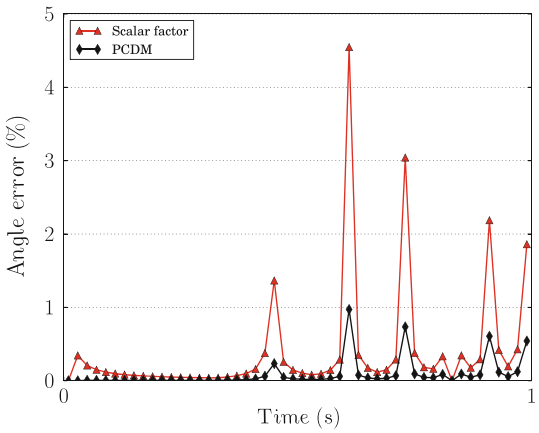

Fig. 6 The error in the prediction of the unit vector as a function of time for the scalar factor method and the PCDM method

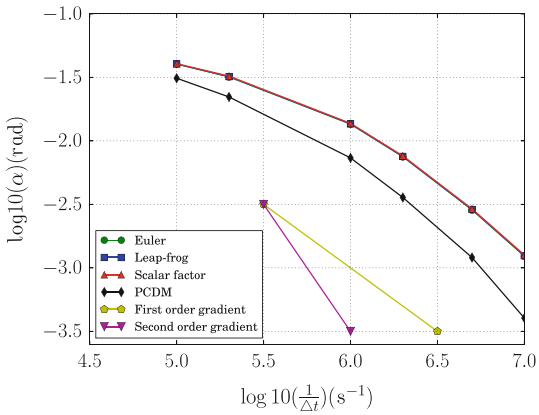

Fig. 7 The angle error of the unit vector as a function of the inverse of time step $\frac{1}{\Delta t}$ (log–log plot)

4.2.1 Comparison with a test case

Fourth test case: prescribed three-dimensional torque on a particle To compare the rate of convergence, the test case of a fibre, with an initial orientation of $\boldsymbol{B} = (1.0,\ 0.0,\ 0.0)$, is placed in a torque field. A small nonlinear torque, $\boldsymbol{\tau} = (A\ \sin(200\pi t),\ C\ \exp(3.0 + 24t),\ C\ 5^{52t})$, in which A and C are 5.0×10^{-4} and 1.0×10^{-5}, respectively, is applied on the fibre to force it to rotate.

The various Quaternion integration methods are used to evaluate the evolution of the rotation of the fibre around its axis up to $T = 0.1$ s. The error of the orientation at time step n of the fibre is determined by evaluating

$$\alpha_n^{\text{error}} = \| \arccos(\boldsymbol{b}_n \cdot \boldsymbol{B}) - \arccos(\boldsymbol{B}_n \cdot \boldsymbol{B}) \|, \tag{66}$$

where $\boldsymbol{b}_n$ in world space is the unit vector projection of $\boldsymbol{B}_n$ as determined by the numerical methods described in Sect. 3, and $\boldsymbol{B}_n$ considered as the analytical result of the unit vector in world space at time step n. The total rotational angle error is then summed as

$$\alpha = \frac{\Delta t}{T} \sum_{n=1}^{n_{\text{final}}} \alpha_n^{\text{error}}, \tag{67}$$

where n_{final} represents the total number of time steps required to achieve time T. The time step, Δt, is varied between 1.0×10^{-5} and 1.0×10^{-7}, as the Euler method does not converge for a larger time step. The effect of the time step on the total error is shown on a log-log scale in Fig. 7.

In this figure, also the slopes of first- and second-order rate of convergence are indicated. The error in the prediction of the orientation of the PCDM method is always significantly less compared to the other methods. Because the Euler, leap-frog and scalar function methods are derived from the Taylor expansion, they have a

very similar rate of convergence, which is of first order for small time steps. This is because they all inhibit the error of the re-normalisation of the Quaternion, irrespective of the order of the Taylor series considered. With a time step larger than $\triangle t = 1.0 \times 10^{-6}$, the rate of convergence is less than 1.

The rate of convergence of the PCDM method is significantly larger than that of the other methods, even for large time steps and approaching second-order rate of convergence as the time step decreases.

4.3 Discussion

For the simplest test case considered, a single falling fibre, all four methods show acceptable results. For all other test cases, the Euler method and leap-frog method show a rapid increase in mass conservation error in time. Although the scalar factor method always shows a stable solution, the error in prediction for the two test cases involving a prescribed torque on a particle are larger than with the newly proposed predictor-corrector method. Moreover, the study of the rate of convergence in the fourth test case shows that the scalar factor method shows a significantly slower rate of convergence than the predictor-corrector method. Moreover, there are no significant differences in computational cost for the methods. The novel PCDM method is more accurate and stable than the other methods for any time step considered in the four test cases.

5 Conclusions

The modelling of the dynamics of non-spherical particles is significantly more complex than the modelling of spherical particles. The difference arises from the requirement of determining the orientation and rotation of a non-spherical particle. For spherical particles, this is done by the application of a vector only. For non-spherical particles, several frameworks are available to describe the orientation: Euler angles, rotation matrices and unit Quaternions. The application of both Euler angles and rotation matrices suffers from serious drawbacks, concerning lack of stability and uniqueness and the occurrence of singularities. Therefore, unit Quaternions seem to be the most attractive framework and are used widely within the modelling of the dynamics of non-spherical particles.

However, there are a number of potential drawbacks when using unit Quaternions to represent the orientation and rotation of non-spherical particles. The first drawback is that almost all research papers so far employ both unit Quaternions and rotation matrices to determine the rotational behaviour of a non-spherical particle. This results in an increased requirement of computer memory and possible inaccuracies which are introduced by the frequent conversion from the unit Quaternion to a rotation matrix and back.

The second drawback concerns the conservation of length of a unit Quaternion. As a Quaternion describes both orientation and scaling, a unit Quaternion describing solely rotation must remain of constant, unit, length throughout all of the operations. Most methods put forward in the literature are based on a Taylor series expansion of the unit Quaternion and require addition, subtraction or scaling. This inherently leads to a change in length of the Quaternion during integration, and, consequently, re-normalisation is required. Although the re-normalisation restores the correct length of the Quaternion, it changes the relationship between the four parameters of which the Quaternion exists, thereby introducing a significant error. Applying higher-order methods, such as based on Runge–Kutta, does not prevent these errors from arising, and worsen rather than improve the overall accuracy of the method.

Both the drawbacks are addressed in this paper. The present work derives a new framework to transform vectors and tensors by unit Quaternions directly, and the necessity of rotation matrices is removed altogether. This means that the algorithm derived in this paper can describe the rotation of non-spherical particles using four parameters only, making it favourable for large-scale computations involving many particles.

To address the second drawback, a novel framework to integrate unit Quaternions is put forward in this paper, the predictor-corrector direct multiplication method. This novel algorithm avoids the use of subtraction or addition of Quaternions and uses multiplication of Quaternions only, so that re-normalisation is not required. The algorithm is based on a predictor-corrector method, so that the various time levels are not mixed.

In this paper, various numerical integration methods for Quaternions put forward in the literature are scrutinised and compared by applying each of them to four test cases: a single falling fibre, nine falling fibres, the 2D rotation of a vector projected on a sphere with a prescribed torque function and a 3D prescribed torque on a non-spherical fibre. In the first two test cases, the error in total energy is compared between the different methods, and in the third test case, the predicted orientation of the vector as a function of time is compared

to the analytical solution. In the last test case, in which the rate of convergence is studied, it is shown that the novel predictor-corrector direct multiplication method has a higher-order rate of convergence than other methods from the literature. The other methods never exceed a rate of convergence of 1, caused by the addition or subtraction of Quaternions and the subsequent necessity of re-normalisation. The novel predictor-corrector direct multiplication method put forward in this paper approaches a rate of convergence of 2. All the test cases presented in this paper show a significant improvement in accuracy of the algorithm put forward in this paper compared to other algorithms found in the literature.

Acknowledgements The authors are grateful for the fruitful discussions with Dr Otis Walton from www.grainflow.com. The authors are grateful to the Engineering and Physical Sciences Research Council (EPSRC) for their financial support (Grant No. EP/G049262/1).

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

1. Allen, M.P., Tildesley, D.J.: The Computer Simulation of Liquids, Vol. 42. Oxford University Press, Oxford (1989)
2. Altmann, S.L.: Rotations, Quaternions, and Double Groups, Vol. 3. Clarendon Press Oxford, England (1986)
3. Arribas, M., Elipe, A., Palacios, M.: Quaternions and the rotation of a rigid body. Celestial Mech. Dyn. Astr. **96**(3-4), 239–251 (2006). doi:10.1007/s10569-006-9037-6
4. Baraff, D.: An Introduction to Physically Based Modeling: Rigid Body Simulation I–Unconstrained Rigid Body Dynamics. SIGGRAPH Course Notes (1997)
5. Betsch, P., Siebert, R.: Rigid body dynamics in terms of Quaternions: Hamiltonian formulation and conserving numerical integration. Int. J. Numer. Methods Eng. **79**(4), 444–473 (2009)
6. Betsch, P., Steinmann, P.: Constrained integration of rigid body dynamics. Comput. Methods Appl. Mech. Eng. **191** (3–5), 467–488 (2001)
7. Celledoni, E., Fassò, F., Säfström, N., Zanna, A.: The exact computation of the free rigid body motion and its use in splitting methods. SIAM J. Sci. Comput. **30**, 2084–2112 (2008)
8. Celledoni, E., Säfström, N.: Efficient time-symmetric simulation of torqued rigid bodies using Jacobi elliptic functions. J. Phys. A. Math. General **39**(19), 5463–5478 (2006)
9. Chou, J.: Quaternion kinematic and dynamic differential equations. IEEE Trans. Robot. Autom. **8**(1), 53–64 (1992)
10. Cundall, P., Strack, O.D.L.: A discrete numerical model for granular assemblies. Géotechnique **29**(1), 47–65 (1979)
11. Delaney, G.W., Cleary, P.: The packing properties of superellipsoids. EPL (Europhys. Lett.) **89**(3), 34,002 (2010)
12. Diebel, J.: Representing attitude: Euler angles, unit Quaternions, and rotation vectors. Tech. rep., Stanford University, California, USA (2006)
13. Eberly, D.: Quaternion algebra and calculus. Magic Software, Inc. (2002)
14. Eberly, D.: Rotation representations and performance issues. Magic Software, Inc. (2002)
15. Evans, D., Murad, S.: Singularity free algorithm for molecular dynamics simulation of rigid polyatomics. Mol. Phys. **34**(2), 327–331 (1977)
16. Grassia, F.S.: Practical parameterization of rotations using the exponential map. J. Graphics Tools **3**, 1–13 (1998)
17. Gsponer, A., Hurni, J.P.: The physical heritage of Sir W.R. Hamilton. In: The Mathematical Heritage of Sir William Rowan Hamilton—commemorating the sesquicentennial of the invention of Quaternions, pp. 1–37. Trinity College, Dublin (1993)
18. Hairer, E., Vilmart, G.: Preprocessed discrete Moser Veselov algorithm for the full dynamics of a rigid body. J. Phys. A Math. General **39**(42), 13,225–13,235 (2006)
19. Hamilton, W.R.: On Quaternions; or on a new system of imaginaries in Algebra. Lond. Edinburgh Dublin Philos. Mag. J. Sci. (3d Series) **15-36**, 1–306 (1844)
20. Hoffmann, G.: Application of Quaternions. Tech. Rep. February 1978, Technische Universität Braunschweig (1978)
21. Hoover, W.G.: Lecture Notes in Physics–Molecular Dynamics. Springer, US (1986)
22. Ibanez, L.: Tutorial on Quaternions Part I. Insight Segmentation and Registration Toolkit (ITK) (2001)
23. Karney, C.F.F.: Quaternions in molecular modeling. J. Mol. Graph. Model. **25**(5), 595–604 (2007)
24. Kleppmann, M.: Simulation of colliding constrained rigid bodies. Tech. Rep. 683, University of Cambridge, UCAM-CL-TR-683, ISSN 1476-2986 (2007)
25. Kosenko, I.: Integration of the equations of a rotational motion of a rigid body in quaternion algebra. The Euler case. J. Appl. Math. Mech. **62**(2), 193–200 (1998)
26. Kuipers, J.B.: Quaternions and rotation sequences. Mater. Sci. Eng. A **271**(d), 322–333 (1999)
27. Langston, P.A., Al-Awamleh, M.A., Fraige, F.Y., Asmar, B.N.: Distinct element modelling of non-spherical frictionless particle flow. Chem. Eng. Sci. **59**(2), 425–435 (2004)
28. Latham, J., Munjiza, A.: The modelling of particle systems with real shapes. Philos. Trans. Roy. Soc. Lond. Ser. A. Math. Phys. Eng. Sci. **362**(1822), 1953 (2004)
29. Mclachlan, R.I., Zanna, A.: The discrete Moser-Veselov algorithm for the free rigid body, revisited. Tech. rep., University of Bergen (2003)
30. Mortensen, P., Andersson, H., Gillissen, J., Boersma, B.J.: Dynamics of prolate ellipsoidal particles in a turbulent channel flow. Phys. Fluids **20**(9), 093,302 (2008)
31. Moser, J., Veselov, A.P.: Discrete versions of some classical integrable systems and factorization of matrix polynomials. Commun. Math. Phys. **139**(2), 217–243 (1991)

32. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C, 2nd ed. Cambridge University Press, Cambridge (1992)
33. Qi, D.: Direct simulations of flexible cylindrical fiber suspensions in finite Reynolds number flows. J. Chem. Phys. **125**(11), 114,901 (2006)
34. Sabatini, A.M.: Quaternion-based strap-down integration method for applications of inertial sensing to gait analysis. Med. Biol. Eng. Comput. **43**(1), 94–101 (2005)
35. Walton, O.R., Braun, R.L.: Simulation of rotary-drum and repose tests for frictional spheres and rigid sphere clusters. DOE/NSF Workshop on Flow of Particulates (1993)
36. Whitmore, S.A., Hughes, L.: Calif: Closed-form Integrator for the Quaternion (Euler Angle) Kinematics Equations, US patent (2000)