SPIRAL: An Efficient Algorithm for the Integration of the Equation of Rotational Motion

Carlos Andrés del Valle[a,b,*], Vasileios Angelidakis[a,c], Sudeshna Roy[a], José Daniel Muñoz[b], Thorsten Pöschel[a]

[a] *Institute for Multiscale Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstrasse 3, 91058 Erlangen, Germany*
[b] *Departamento de Física, Universidad Nacional de Colombia, Carrera 45 No. 26-85, Edificio Uriel Gutiérrez, Bogotá D.C., Colombia*
[c] *School of Natural and Built Environment, Queen's University Belfast, David Keir Building, Stranmillis Road, BT9 5AG Belfast, United Kingdom*

Abstract

We introduce Spiral, a third-order integration algorithm for the rotational motion of extended bodies. It requires only one force calculation per time step, does not require quaternion normalization at each time step, and can be formulated for both leapfrog and synchronous integration schemes, making it compatible with many particle simulation codes. The stability and precision of Spiral exceed those of state-of-the-art algorithms currently used in popular DEM codes such as Yade, MercuryDPM, LIGGGHTS, PFC, and more, at only slightly higher computational cost. Also, beyond DEM, we see potential applications in all numerical simulations that involve the 3D rotation of extended bodies.

Keywords: numerical integration of the equation of rotational motion, particle simulation, discrete element method (DEM), molecular dynamics (MD)

1. Introduction

The efficient and robust integration of the equations of rotational motion is a fundamental requirement for the simulation of many-particle systems. Developers of popular programs for the Discrete Element Method (DEM), see Table 1, and Molecular Dynamics (MD), such as LAMMPS [45] and GROMACS [2] face the problem of compromising between simple algorithms that require small time steps to maintain stability and more efficient algorithms such as Runge-Kutta-4 [25], that can handle larger time steps, but at the cost of a higher number of force calculations per time step, which can be prohibitively expensive for larger systems.

When using quaternions to describe the particles' orientations in numerical algorithms, care must be taken to preserve their unitarity [24, 19]. Unfortunately, many popular al-

*Corresponding author.
E-mail address: cdelv@unal.edu.co

Table 1: Popular algorithms for integrating the equation of rotational motion and DEM software that uses them. For a detailed discussion, see Appendix B.

algorithm	DEM codes using it			
Direct Euler [7]	MFix[42]	EDEM[3]	BlazeDEMGPU[13]	MUSEN[10]
Velocity Verlet[52]	LIGGGHTS[22]	GranOO[4]	EDEM[3]	MercuryDPM[54]
Fincham [11]	Yade[40]	Esys_Particle [1]	WooDEM[55]	
Buss [8]	PFC7[18]			
Johnson et al.[21]	PFC7[18]			
PFC4 [17, 30]	MercuryDPM[54, 30]			

gorithms do not conserve the quaternion norm, thus requiring normalization at each time step. Table 1 provides a list of algorithms utilized in widespread DEM codes to integrate the rotational motions of particles. Among those, Buss' algorithm [8] stands out as it is the only one that does not require subsequent quaternion normalization. All algorithms listed in Table 1 provide second-order accuracy per time step, despite having better algorithms at our disposal for decades. For instance, Omelyan [29] proposed a third-order algorithm that shows remarkable energy conservation performance, a critical factor in MD simulations. Neto and Bellucci [28] proposed an algorithm that slightly outperforms Omelyan's in terms of energy conservation, albeit at the cost of higher complexity.

We introduce an algorithm for integrating the equations of rotational motion of extended bodies that offers several advantages: It does not need quaternion normalization and requires only one force calculation per time step. It provides third-order accuracy and excellent numerical stability, including energy conservation, allowing for larger time steps and, thus, reducing overall computational costs. The algorithm can be adapted to different code architectures and simulation frameworks, including leapfrog[1] [16] and non-leapfrog variants, making it flexible and compatible with most simulation setups.

In addition to mechanical many-body systems, numerical integration of the equation of rotational motion is used in many fields, including video game engines like Vulkan [53], Unity [49], and Unreal engine [50], animation and computer graphics [23, 20, 14, 41], robotics [47, 6], sensors [36, 43, 51, 36], navigation systems [56, 9], autonomous vehicles [35, 31, 38], and control systems [44, 27]. Remarkably, many control systems employ variational methods that result in a formulation similar to the one presented here, but they are much more complex [35, 31, 36] and generally provide only second-order accuracy.

We compared the new integration algorithm against the previously mentioned ones, assessing accuracy, stability, performance, and energy conservation. We find excellent results across all the evaluation criteria, highlighting its potential as an improved general method for integrating equations for rotational motion.

[1]Leapfrog is a popular integration scheme where velocities are shifted half a time step with respect to positions. The positions jump one-time step using the shifted velocity, while the velocities jump one time step using the forces computed from the positions. The algorithm features third-order accuracy in each time step [16].

2. Spiral - an algorithm for the numerical integration of the equation of rotational motion

The third-order SPIRAL[2] algorithm applies to leapfrog and non-leapfrog architectures. Here, we describe the leapfrog version. The non-leapfrog version is presented in Appendix A. We describe the 3D orientation of a rigid body at time t by unit quaternions q(t) [15]. Quaternions are an extension of the imaginary numbers where the imaginary part is a three-dimensional vector[3]. For a great introduction to quaternion algebra applied to rotations, see [19] . To derive an expression for $\mathrm{q(t+\Delta t)}$, we start with the quaternion's time derivative [24, 19],

$$\dot{\mathrm{q}} = \frac{\mathrm{dq}}{\mathrm{dt}} = \frac{1}{2}\omega(\mathrm{t})\mathrm{q}\,, \tag{1}$$

where $\omega = \{0, \omega_\mathrm{x}, \omega_\mathrm{y}, \omega_\mathrm{z}\}$ is a pure imaginary quaternion representing the angular velocity $\vec{\omega} = \{\omega_\mathrm{x}, \omega_\mathrm{y}, \omega_\mathrm{z}\}$ in the reference frame of the rotating body. We assume that $\vec{\omega}$ depends only explicitly on time and solve the differential equation, (1) by separation of variables

$$\int_\mathrm{t}^{\mathrm{t}+\Delta\mathrm{t}} \frac{\mathrm{dq}}{\mathrm{q}} = \frac{1}{2}\int_\mathrm{t}^{\mathrm{t}+\Delta\mathrm{t}} \omega(\mathrm{t}')\mathrm{dt}'\,, \tag{2}$$

to obtain

$$\mathrm{q(t+\Delta t) = q(t)}\ \exp\left(\frac{1}{2}\int_\mathrm{t}^{\mathrm{t}+\Delta\mathrm{t}} \omega(\mathrm{t}')\,\mathrm{dt}'\right)\,. \tag{3}$$

We expand the unknown $\omega(\mathrm{t}')$ in a Taylor series around a,

$$\omega(\mathrm{t}') = \omega(\mathrm{a}) + \dot{\omega}(\mathrm{a})(\mathrm{t}'-\mathrm{a}) + \frac{1}{2}\ddot{\omega}(\mathrm{a})(\mathrm{t}'-\mathrm{a})^2 + \mathcal{O}\left[(\mathrm{t}'-\mathrm{a})^3\right]\,. \tag{4}$$

Take $\mathrm{a} = \mathrm{t} + \frac{\Delta\mathrm{t}}{2}$ (leapfrog assumption), and insert it into Eq. (3), to obtain

$$\mathrm{q(t+\Delta t) = q(t)}\ \exp\left[\frac{\Delta\mathrm{t}}{2}\omega\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right) + \mathcal{O}(\Delta\mathrm{t}^3)\right]\,. \tag{5}$$

The exponential of a purely imaginary quaternion $\theta\hat{\mathrm{u}}$, where θ is a scalar and $\hat{\mathrm{u}} = \{0, \mathrm{u_x}, \mathrm{u_y}, \mathrm{u_z}\}$, a unit quaternion, reads [19]

$$\mathrm{e}^{\theta\hat{\mathrm{u}}} = \cos\theta + \vec{\mathrm{u}}\sin\theta\,, \tag{6}$$

with the unit vector $\vec{\mathrm{u}} = \{\mathrm{u_x}, \mathrm{u_y}, \mathrm{u_z}\}$. Then

$$\mathrm{q(t+\Delta t) = q(t)}\left(\cos\theta_1 + \frac{\vec{\omega}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right)}{\left|\vec{\omega}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right)\right|}\sin\theta_1\right)\,,\quad \text{with}\quad \theta_1 \equiv \frac{\Delta\mathrm{t}}{2}\left|\vec{\omega}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right)\right|\,. \tag{7}$$

[2]SPIRAL is an acronym for "stable particle rotation integration algorithm".

[3]A quaternion $\mathrm{q} = \{\mathrm{q_r}, \mathrm{q_x}, \mathrm{q_y}, \mathrm{q_z}\} = \mathrm{q_r} + \mathrm{q_x}\hat{\imath} + \mathrm{q_y}\hat{\jmath} + \mathrm{q_z}\hat{\mathrm{k}}$ is a four-dimensional object with a real component $\mathrm{q_r}$ and three imaginary ones forming a vector, $\vec{\mathrm{q}} = \mathrm{q_x}\hat{\imath} + \mathrm{q_y}\hat{\jmath} + \mathrm{q_z}\hat{\mathrm{k}}$, where the basis $\hat{\imath}, \hat{\jmath}, \hat{\mathrm{k}}$ obeys the algebra $\hat{\imath}\cdot\hat{\imath} = \hat{\jmath}\cdot\hat{\jmath} = \hat{\mathrm{k}}\cdot\hat{\mathrm{k}} = -1$, $\hat{\imath}\cdot\hat{\jmath} = -\hat{\jmath}\cdot\hat{\imath} = \hat{\mathrm{k}}$, $\hat{\jmath}\cdot\hat{\mathrm{k}} = -\hat{\mathrm{k}}\cdot\hat{\jmath} = \hat{\imath}$ and $\hat{\mathrm{k}}\cdot\hat{\imath} = -\hat{\imath}\cdot\hat{\mathrm{k}} = \hat{\jmath}$, combining the essential property of imaginary numbers and the cross product.

The shown transformations preserve the norm of the initial quaternion, thus eliminating the need for subsequent normalization. The approach is similar to the one by Buss [8], but Eq. 7 achieves a higher-order precision. Also, Seelen et al. [37] propose a predictor-corrector version of the Zhao and Van Wachem scheme [57] by using a similar formulation, but achieving only second-order accuracy as well.

The expression for $q(t + \Delta t)$ is independent of the method chosen to compute $\vec{\omega}$. To compute the angular velocity, we need to solve Euler's equations of rotational motion [12],

$$\begin{aligned}\dot{\omega}_x &= \frac{M_x}{I_x} + \omega_y\,\omega_z \frac{I_y - I_z}{I_x}\,,\\ \dot{\omega}_y &= \frac{M_y}{I_y} + \omega_z\,\omega_x \frac{I_z - I_x}{I_y}\,,\\ \dot{\omega}_z &= \frac{M_z}{I_z} + \omega_x\,\omega_y \frac{I_x - I_y}{I_z}\,,\end{aligned} \tag{8}$$

where $\vec{M}(t)$ is the torque acting in the body's principal axis reference frame and the body's principal moments of inertia are I_x, I_y, and I_z.

For optimal results, accurate angular velocity calculation is crucial and non-trivial, given the highly nonlinear expressions for angular velocity. We employ a Strong-Stability Preserving Runge-Kutta-3 (SSPRK3) scheme [39], which is highly stable and requires fewer operations than Runge-Kutta-4 [25] and the iterative algorithm used by Omelyan [29]. This way, we avoid performing the approximation done by the Omelyan algorithm in the angular velocity calculation, see Appendix B. Moreover, inspired by [21], we compute the torque $\vec{M}(t)$ only once per time step, i.e. assuming the torque constant during the time step. Using SSPRK3, we update the angular velocity through

$$\vec{\omega}\left(t + \frac{\Delta t}{2}\right) = \vec{\omega}\left(t - \frac{\Delta t}{2}\right) + \frac{1}{6}\left(\vec{K}_1 + \vec{K}_2 + 4\vec{K}_3\right)\,, \tag{9}$$

with

$$\begin{aligned}\vec{K}_1 &= \Delta t\,\dot{\vec{\omega}}\left(\vec{\omega}, \vec{M}(t)\right)\,,\\ \vec{K}_2 &= \Delta t\,\dot{\vec{\omega}}\left(\vec{\omega} + \vec{K}_1, \vec{M}(t)\right)\,,\\ \vec{K}_3 &= \Delta t\,\dot{\vec{\omega}}\left(\vec{\omega} + \frac{1}{4}\left(\vec{K}_1 + \vec{K}_2\right), \vec{M}(t)\right)\,,\end{aligned} \tag{10}$$

where the angular acceleration $\dot{\vec{\omega}}$ is given by Eq. (8). This SSPRK3 integration scheme for the angular velocity is an excellent choice for non-spherical particles; nevertheless, for spherical ones, we suggest using a normal leapfrog [16] as the non-linearities vanish. Incurring in performance improvements without losing precision.

Using the leapfrog scheme, $\vec{\omega}$ is updated before q. Therefore, it is necessary to compute the angular velocity back by half a step before starting the simulation. This can be done with Eq. (9) with the argument $-\frac{\Delta t}{2}$ instead of Δt.

3. Validation against an analytical solution

Consider a rigid body with principal moments of inertia $\mathrm{I_x}, \mathrm{I_y}, \mathrm{I_z}$. For the special case $\mathrm{I_x} \neq \mathrm{I_y} = \mathrm{I_z}$ we can solve the set of nonlinear differential equations, Eq. (8) analytically, provided a constant torque $\vec{\mathrm{M}} = \{\mathrm{M_x}, 0, 0\}$ with $\mathrm{M_x} \neq 0$ and $\omega_\mathrm{x}(\mathrm{t}) \neq 0$ at all times. The solution reads (see Appendix C for the derivation)

$$\begin{aligned}
\omega_\mathrm{x}(\mathrm{t}) &= \sqrt{\tau} = \omega_\mathrm{x}^0 + \frac{\mathrm{M_x}}{\mathrm{I_x}}\mathrm{t}, \\
\omega_\mathrm{y}(\tau) &= \mathrm{k_1} \cos(\Omega\tau) + \mathrm{k_2} \sin(\Omega\tau) , \\
\omega_\mathrm{z}(\tau) &= \eta\Omega \left(\mathrm{k_2} \cos(\Omega\tau) - \mathrm{k_1} \sin(\Omega\tau)\right) ,
\end{aligned} \tag{11}$$

with the initial angular velocity $\vec{\omega}(0) = \{\omega_\mathrm{x}^0, \omega_\mathrm{y}^0, \omega_\mathrm{z}^0\}$ and

$$\begin{aligned}
\Omega^2 &\equiv -\left(\frac{\mathrm{I_x}}{2\mathrm{M_x}}\right)^2 \frac{(\mathrm{I_x}-\mathrm{I_y})(\mathrm{I_z}-\mathrm{I_x})}{\mathrm{I_z}\mathrm{I_y}} , \\
\eta &\equiv \frac{\mathrm{I_y}}{\mathrm{I_z}-\mathrm{I_x}} \frac{2\,\mathrm{M_x}}{\mathrm{I_x}} , \\
\mathrm{k_1} &\equiv \omega_\mathrm{y}^0 \cos\left(\Omega\left(\omega_\mathrm{x}^0\right)^2\right) - \frac{\omega_\mathrm{z}^0}{\Omega\eta} \sin\left(\Omega\left(\omega_\mathrm{x}^0\right)^2\right) , \\
\mathrm{k_2} &\equiv \omega_\mathrm{y}^0 \sin\left(\Omega\left(\omega_\mathrm{x}^0\right)^2\right) + \frac{\omega_\mathrm{z}^0}{\Omega\eta} \cos\left(\Omega\left(\omega_\mathrm{x}^0\right)^2\right) .
\end{aligned} \tag{12}$$

Note that this solution does not apply for $\mathrm{M_x} = 0$ (for this case, the solution is also given in Appendix C). Further analytically solvable special cases of the rigid body equation can be found in [26, 33, 48].

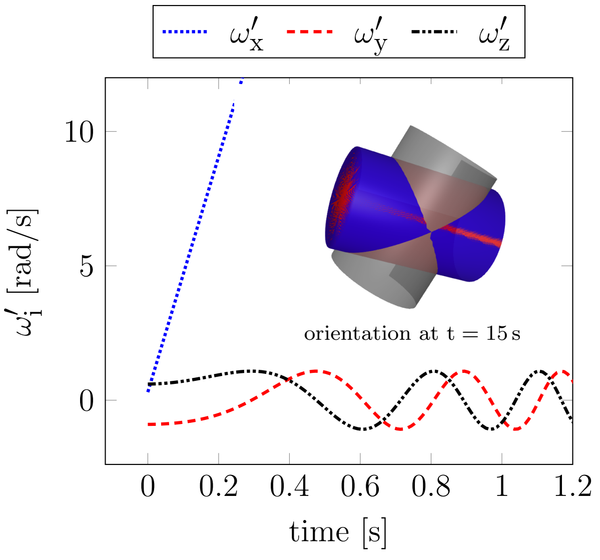
For subsequent comparison, we use the above analytical expression of the body's angular velocity (which we call $\vec{\omega}'$) to compute its orientation $\mathrm{q}'(\mathrm{t})$ by numerical integration of (1) using the highly accurate *DifferentialEquations.jl* [32]. The orientation $\mathrm{q}'(\mathrm{t})$ will serve as a reference to quantify the accuracy of $\mathrm{q}(\mathrm{t})$ as delivered by our algorithm compared to other algorithms from the literature.

To compute the error between the analytical solution and the solution from the different integration algorithms, we define the error metric:

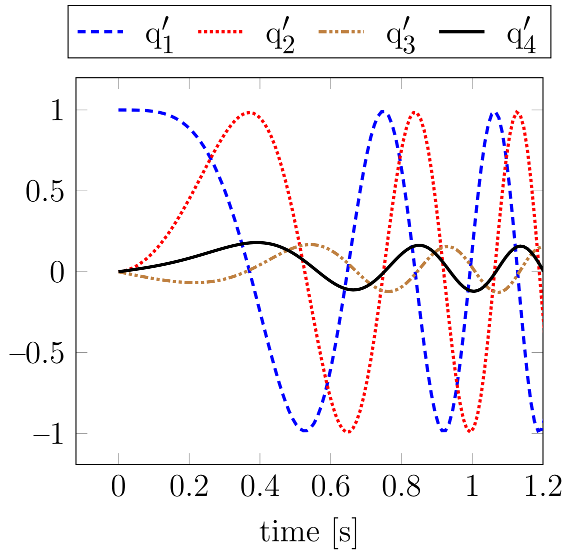
$$\left\|\vec{\mathrm{v}} - \vec{\mathrm{v}}'\right\| := \frac{\sum_\mathrm{i} |\mathrm{v_i} - \mathrm{v}'_\mathrm{i}|}{\sum_\mathrm{i} |\mathrm{v}'_\mathrm{i}|}. \tag{13}$$

This metric is the vector's relative $\mathrm{L_1}$ norm. We use the $\mathrm{L_1}$ norm instead of $\mathrm{L_2}$ because $\mathrm{L_2}$ measures the drift of the norm and the error in the direction; in contrast, the $\mathrm{L_1}$ norm gives more weight to the error in individual components of the vector.

The analytic solution, Eq. (11), describes an object with two identical principal moments of inertia, e.g., a cylinder, subjected to a constant torque. Here, we consider a steel cylinder of radius 5 cm, height 15 cm, and density $7750\,\mathrm{kg/m^3}$. Then, the corresponding tensor of inertia in the cylinder's principal axis frame is $\vec{\mathrm{I}} = \{\mathrm{I_x}, \mathrm{I_y}, \mathrm{I_z}\} = \{0.0114, 0.0228, 0.0228\}\,\mathrm{kg\,m^2}$. We further assume the cylinder's initial angular velocity, $\vec{\omega_0} = \{0.3, -0.9, 0.6\}\,\mathrm{rad/s}$, its initial orientation $\mathrm{q_0} = \{1, 0, 0, 0\}$, and the time-independend torque, $\mathrm{M_x} = 0.5\,\mathrm{N\,m}$. Fig. 1 shows the analytical solution for the specified parameters.



(a) components of the angular velocity



(b) quaternion components

Figure 1: 3D rotation of a cylinder subjected to a constant torque: (a) analytical solution for the angular velocity $\vec{\omega}'$, (b) highly accurate numerical reference solution for the orientation $\mathrm{q}'(\mathrm{t})$. For the detailed set of parameters, see the text. The inset to (a) shows the cylinder's orientation at $\mathrm{t} = 15$ s for the reference solution (red), our proposed Spiral algorithm (blue, nearly coinciding with red), and the direct Euler algorithm (grey).

The precision of various integration algorithms is depicted in Fig. 2. It shows the error as defined in Eq. (13) at $\mathrm{t} = 1\,\mathrm{s}$ between the numerically obtained values, q and $\vec{\omega}$, and the (semi-)analytical results, q' and $\vec{\omega}'$, respectively, as functions of the time step $\Delta\mathrm{t}$. We see that Spiral outperforms all other integration schemes studied over the entire range of time steps, some of them by several orders of magnitude. Only the Runge-Kutta-4-scheme [25] performs better; however, this algorithm requires four torque computations per time step compared to one for Spiral and, therefore, Runge-Kutta-4 is not suitable for large-scale simulations. More details on the compared algorithms can be found in Appendix B.

While Fig. 2 provides valuable insights into the accuracy of the integration algorithms, it does not explicitly demonstrate their numerical stability. To assess stability, Fig. 3 shows the accumulated relative error as a function of the total simulation time. The time step is chosen $\Delta\mathrm{t} = 10^{-5}\,\mathrm{s}$.

Again, we see that Spiral outperforms the other algorithms, except for Runge-Kutta-4 [25], which uses four torque calculations per time step. In contrast to Spiral, all other integration algorithms in the plot require quaternion normalization after each time step, except for the scheme by Buss [8]. We have noticed that the Omelyan algorithm [29], albeit not requiring quaternion normalization due to its mathematical formulation, tends to

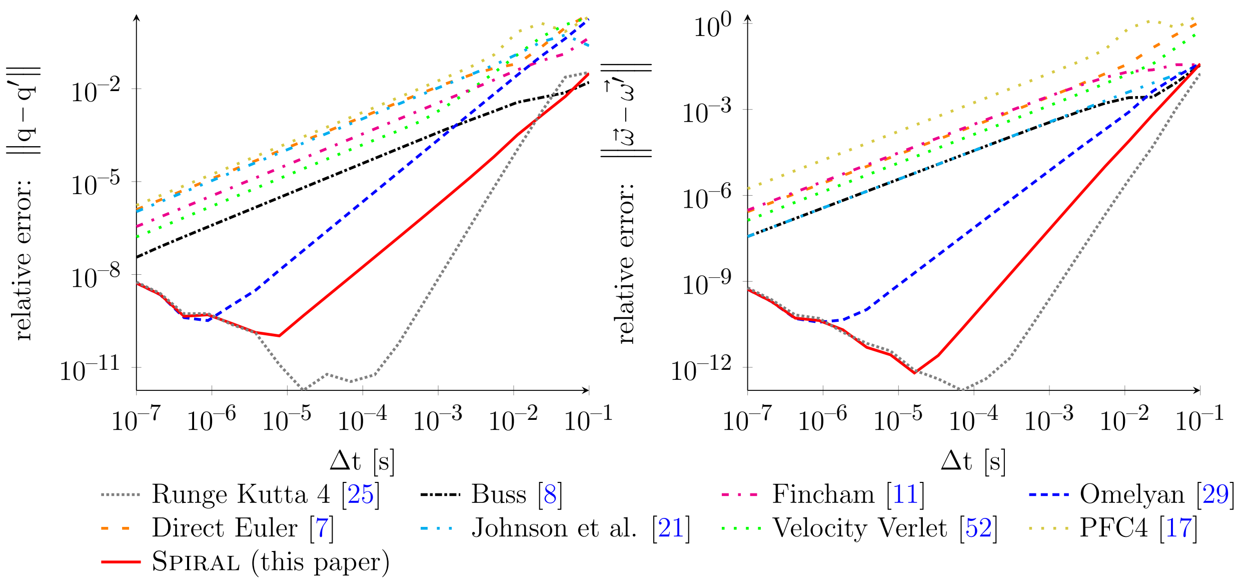


Figure 2: Comparison of various algorithms with the analytical solution at time $t = 1\,s$: (left) relative error of the predicted quaternion with respect to the semi-analytical solution as a function of the time step, (right) relative error of the angular velocity with respect to the analytical solution as a function of the time step. The errors were computed using Eq. (13).

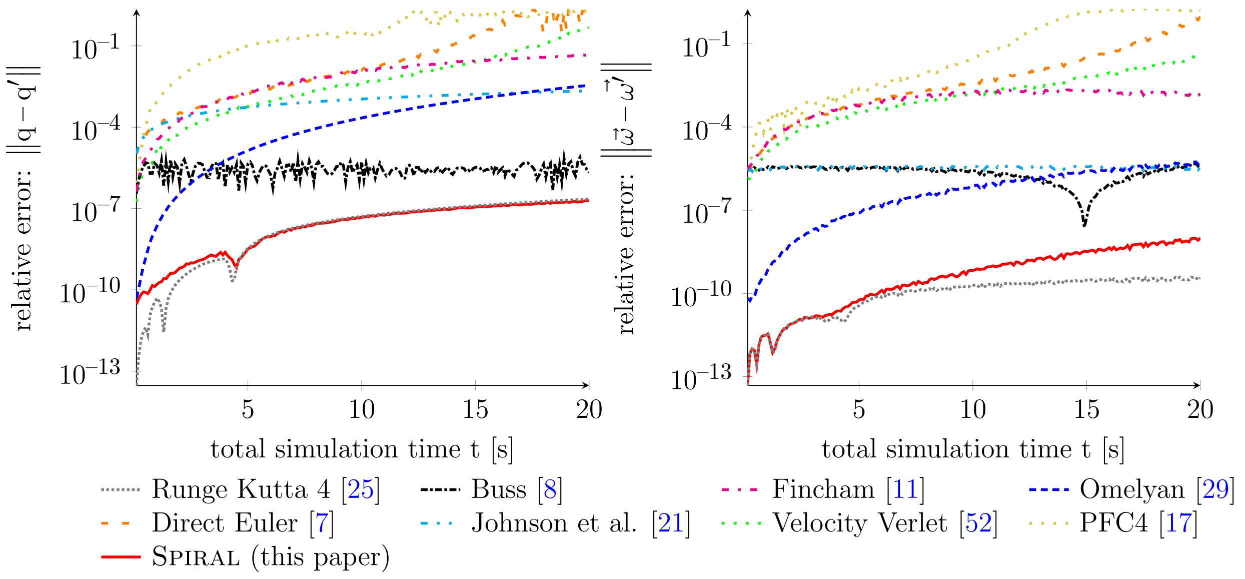


Figure 3: Comparison of various algorithms with the analytical solution; evolution of the error. The time step is $\Delta t = 10^{-5}$ s: (left) relative error of the predicted quaternion with respect to the semi-analytical solution as a function of total simulation time, (right) relative error of the angular velocity with respect to the analytical solution as a function of the time step. The error was computed according to Eq. (13).

accumulate significant numerical errors over time. Therefore, periodic normalization of the quaternions is advised.

4. Performance analysis

Fig. 4 shows the computer time of the algorithms for one time step, relative to the time employed by the Direct Euler algorithm when normalizing the quaternion [7].

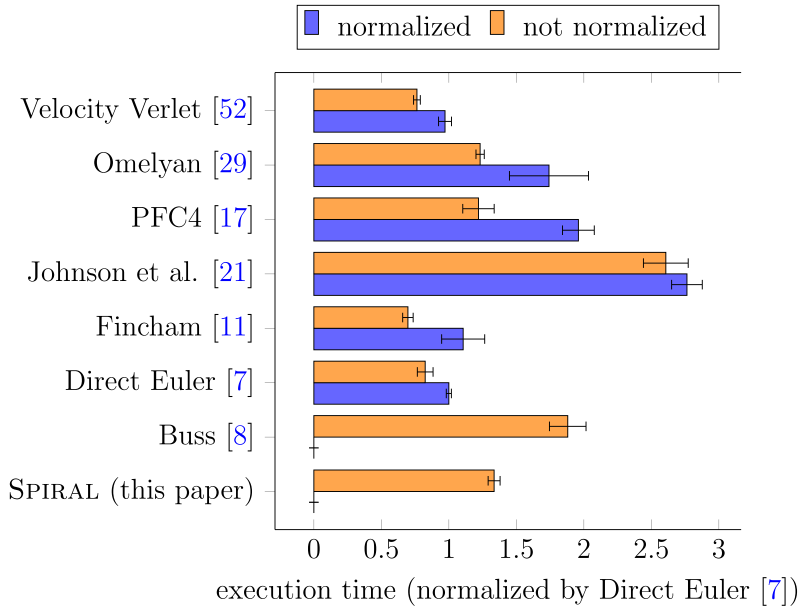


Figure 4: Execution time per time step relative to the time required by the Direct Euler algorithm [7]. Results with and without quaternion normalization. Note that SPIRAL does not require quaternion normalization.

The performance metrics were determined using BENCHMARKTOOLS.JL [5]. We see that quaternion normalization involves significant performance degradation. Therefore, it is a clear advantage that SPIRAL does not require normalization.

Although SPIRAL takes about 30 % more time to perform a single time step compared to the Direct Euler method [7], it ultimately allows for faster simulations. This is because SPIRAL requires fewer steps to achieve the same level of accuracy as other methods, as quantified in the previous Section. For a fair evaluation of the algorithms' efficiency, we determine the appropriate time step that achieves a given error by using the data from Fig. 2, with the bracketing algorithm from ROOTS.JL [34]. Measuring the total CPU time needed for each algorithm to integrate a particle's motion for a period of 1 s with a predefined total error, we can asses their relative speedups. Fig. 5 shows the speedup of each algorithm as the time spent with Direct Euler [7] divided by the time spent by that algorithm to reach the same target error after 1 s. The time step required for a list of specified target errors is reported in Tab. 2. Again, SPIRAL reveals much better than all other algorithms, around 10 times faster than Omelyan [29] and more than 100 times faster than Velocity Verlet [52]. Observe that all algorithms except Omelyan and SPIRAL are of the same order of accuracy as Direct Euler and, therefore, scale the same as Direct Euler when decreasing the target error.

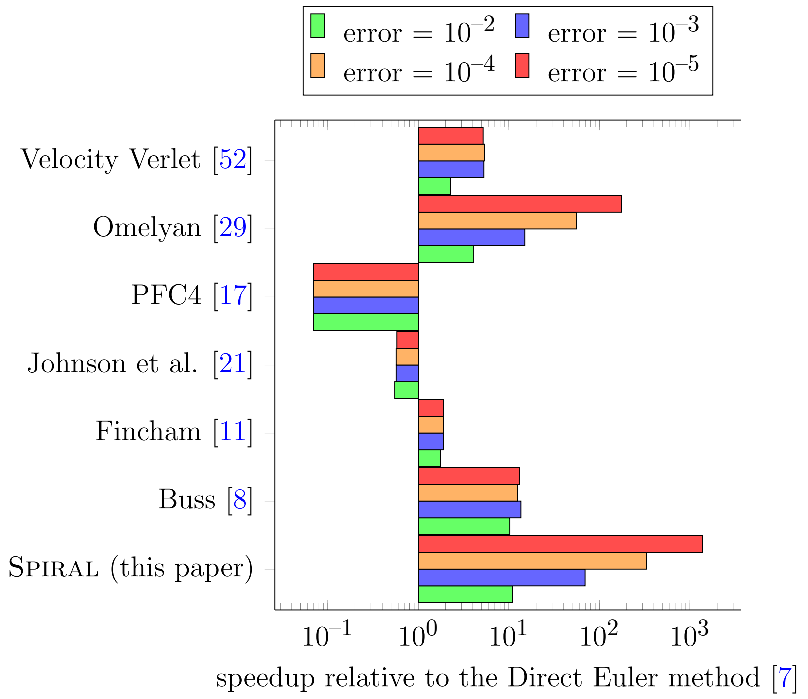


Figure 5: Speedup of several algorithms relative to the Direct Euler method. The speedup is computed by measuring the Computer time needed to simulate 1 s of the system specified in Fig. 1 with a predefined target error of $(\left\|\mathrm{q}-\mathrm{q}'\right\| + \left\|\vec{\omega}-\vec{\omega}'\right\|)/2$ at the end of the simulation. The shown speedup is the time required by the Direct Euler scheme [7] divided by the time required by each algorithm, and is shown for four values of the target errors: 10^{-2} (green), 10^{-3} (blue), 10^{-4} (orange) and 10^{-5} (red). The time step each algorithm requires to achieve the different target errors is provided in Tab. 2. The errors were computed by using Eq. (13).

Table 2: The time step Δt required to achieve the average target error $\left(\|q - q'\| + \left\|\vec{\omega} - \vec{\omega'}\right\|\right)/2$ after 1 s of simulation time for the problem described in Sec. 3. The error was computed using Eq. (13).

	target error			
algorithm	10^{-5}	10^{-4}	10^{-3}	10^{-2}
SPIRAL (this paper)	3.044e-03	8.372e-03	2.329e-02	6.167e-02
Omelyan [29]	2.943e-04	9.319e-04	2.952e-03	9.400e-03
Buss [8]	2.650e-05	2.663e-04	2.890e-03	5.529e-02
Velocity Verlet [52]	6.789e-06	6.852e-05	6.779e-04	3.527e-03
Fincham [11]	3.069e-06	3.071e-05	3.091e-04	3.295e-03
Direct Euler [7]	1.324e-06	1.325e-05	1.333e-04	1.421e-03
Johnson et al. [21]	1.801e-06	1.801e-05	1.804e-04	1.824e-03
PFC4 [17, 30]	1.662e-07	1.662e-06	1.663e-05	1.671e-04

5. Energy conservation of a many-particle system

To evaluate the performance of SPIRAL in a many-body environment, we simulated non-spherical particles in a box. The contacts between the particles and the box and between the particles themselves are described by the conservative Hertz law, ensuring energy conservation. For the translations we used the leapfrog algorithm to integrate Newton's equations of motion, which has the same accuracy order as SPIRAL. Our focus was on studying energy conservation in a dynamic system of chess pieces, as illustrated in Fig. 6.

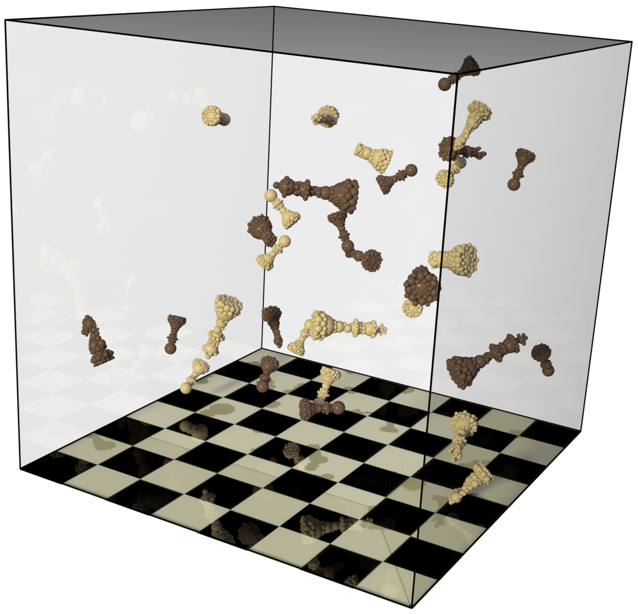


Figure 6: Snapshot of a simulation of non-spherical particles (chess pieces modeled as multi-spheres) bouncing in a box. The elastic Hertz contact force describes particle-particle and particle-wall collisions.

The sizes of the chess pieces follow FIDE recommendations (king: 9.5 cm, queen 8.5 cm, bishop 7 cm, knight 6 cm, rook 5.5 cm, pawn 5 cm). Material density of the pieces is $3200\,\mathrm{kg/m^3}$, the Young modulus, 60 GPa, and the Poisson ratio, 0.25. We used YADE [40] as a simulation framework. Initially, the positions of the objects were random, such that the particles were not in contact. Their initial velocity was 1 m/s in a random direction. Their initial angular velocity was π rad/s, also in a random direction.

Fig. 7 shows the relative change of energy in a total simulation time of 1.5 s as a function of the time step size for three integration algorithms: Fincham, Omelyan, and SPIRAL. To facilitate comparison, the dashed line in the figure represents the Rayleigh time step $\Delta \mathrm{t_c}$, which is the time taken for a shear wave to propagate through a solid particle [46]. It is common to use $\Delta \mathrm{t_c}$ as a criterion to select a time step $\Delta \mathrm{t} \sim 0.1\Delta \mathrm{t_c}$ to $0.3\Delta \mathrm{t_c}$ when looking for simulation stability. We see that, even if the time step employed by SPIRAL or Omelyan is greater than the Rayleigh time step, it still conserves energy more effectively than a second-order algorithm (Fincham's algorithm) when using a much smaller time step.

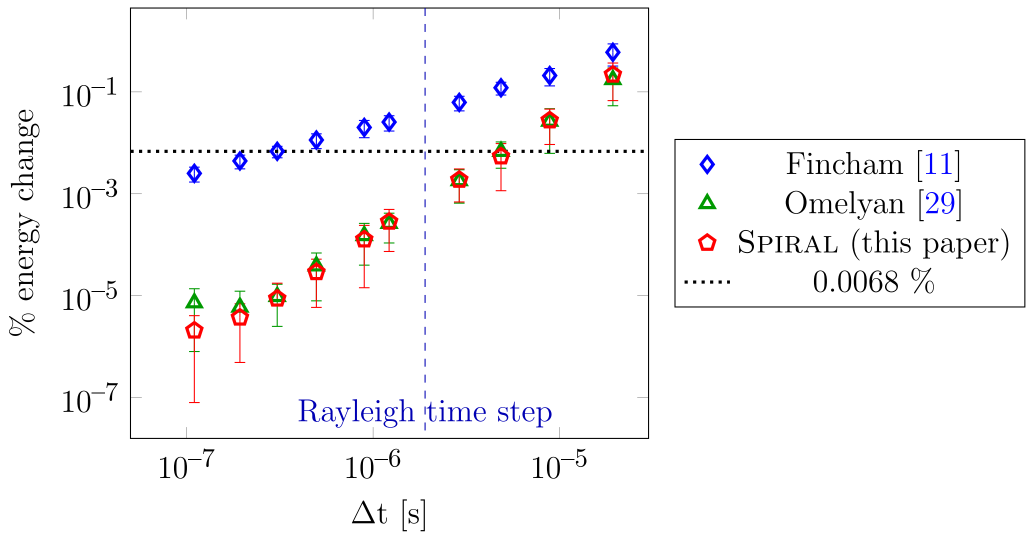


Figure 7: The relative deviation of the total energy from its initial value, plotted against the integration time step, for a simulation time of 1.5 s. The results are an average of 50 random initial conditions. The Rayleigh time step is indicated by the dashed line.

We also see that SPIRAL conserves energy on the same level as the integration scheme by Omelyan [29], which is known to be very stable. An approximated energy change of 0.0068% (the horizontal dotted line in Fig. 7) is reached with a $\Delta t = 4.85 \cdot 10^{-6}$ for Omelyan and SPIRAL algorithms (which are both third order), much larger than $\Delta t = 3.1 \cdot 10^{-7}$ required for Fincham's algorithm (which is second order only). Nevertheless, SPIRAL takes less CPU time (19 s, in comparison with 23 s for Omelyan and 304 s for Fincham, running on eight cores in a laptop with 32 Gb of DDR5 RAM clocked at 6400 MT/s and an AMD Ryzen 9 6900HS Creator Edition CPU). The time difference is because SPIRAL does not need an iterative process at each time step, as Omelyan does, improving the overall efficiency.

6. Conclusions

We introduced SPIRAL, a highly accurate and efficient third-order integration algorithm for rigid body rotational motion. The algorithm displays excellent numerical stability. We provide both leapfrog and non-leapfrog variants for easy adaptation to simulation frameworks. Our algorithm features remarkable accuracy and stability while incurring minimal performance penalties compared to other algorithms used in popular DEM software; see Tab. 1. By construction, our algorithm preserves the norm of the quaternion, thus removing the need for regular quaternion normalization in each time step. Also, SPIRAL is designed to integrate the non-linear Euler's equations, unlike most algorithms in Tab. 1, thus achieving greater accuracy and stability. The stability and preservation of the quaternion norm also enable the use of larger time steps, which improves simulation performance while achieving the same level of accuracy as other integration schemes. In addition, SPIRAL employs a modified third-order Strong-Stability Preserving Runge-Kutta (SSPKR3) scheme to update the angular velocity, which computes the torques only once per time step. When combined, those integration schemes give Spiral an outstanding performance: 10 times faster than Omelyan [29] and 100 times faster than Velocity Verlet [52] to reach the same target error.

We tested the algorithm stability in two scenarios: first, by looking at the accumulation of errors against time for a single rotor with analytical solution (the same we used for studying accuracy) and, second, by testing for energy conservation on a realistic DEM scenario of non-spherical particles interacting with a conservative Hertz law. In the first test, our algorithm was superior to all other algorithms tested and, in the second one, it was on par with the Omelyan algorithm [29], which is known to reach excellent energy conservation.

As part of this work, we integrated Spiral algorithm into two major DEM simulation codes. Now, Spiral is an integral part of the Yade [40] and MercuryDPM [54] distributions, ensuring that our algorithm's benefits are accessible to a broad scientific community.

7. Acknowledgements

CAV thanks the Institute for Multiscale Simulation (MSS) at the Friedrich-Alexander-Universität Erlangen-Nürnberg for funding a visitor's grant. JDM was supported by the Visiting Professor Program, funded by the Bavarian State Ministry of Science and the Arts. SR thanks the Humboldt Research Foundation for granting the 'Humboldt Research Fellowship' and the State of Bavaria for granting a Habilitation Fellowship (FFL). This work was funded by the German Science Foundation (DFG) through grant PO 472/42-1 (SALTED). The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High-Performance Computing Center (NHRFAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). The work was also supported by the Interdisciplinary Center for Nanostructured Films (IZNF), the Competence Unit for Scientific Computing (CSC), and the Interdisciplinary Center for Functional Particle Systems (FPS) at Friedrich-Alexander-Universität Erlangen-Nürnberg.

8. Data availability

The Spiral algorithm was implemented in two major DEM simulation systems Yade and MercuryDPM. The implementation of Spiral as tested in this paper and the corresponding test scripts are available at `https://github.com/cdelv/AlgorithmsForRotationalMotion`.

9. Conflict of interest declaration

We declare we have no competing interests.

References

[1] Abe S, D Place, & P Mora 2004 'A parallel implementation of the lattice solid model for the simulation of rock mechanics and earthquake dynamics' *Pure and Applied Geophysics* 161:2265–2277 doi:10.1007/s00024-004-2562-x.

[2] Abraham MJ, T Murtola, R Schulz, S Páll, JC Smith, B Hess, & E Lindahl 2015 'GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers' *SoftwareX* 1:19–25 doi:10.1016/j.softx.2015.06.001.

[3] Altair 2023 'EDEM - discrete element method (DEM) software' URL https://altair.com/edem/ accessed: 21-06-2023.

[4] Andre D, J Charles, & I Iordanoff 2015 *3D Discrete Element Workbench for Highly Dynamic Thermo-mechanical Analysis: GranOO (Discrete Element Model and Simulation of Continuous Materials Behavior)* Discrete Element Model and Simulation of Continuous Materials Behavior Wiley.

[5] Banchmark 2023 'Benchmarktools.jl' URL https://juliaci.github.io/BenchmarkTools.jl/stable/ accessed: 17-03-2023.

[6] Beeson P & B Ames 2015 'TRAC-IK: An open-source library for improved solving of generic inverse kinematics' in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* IEEE Press p. 928–935 doi:10.1109/HUMANOIDS.2015.7363472.

[7] Boyce W, R DiPrima, & D Meade 2017 *Elementary Differential Equations and Boundary Value Problems* Wiley.

[8] Buss SR 2000 'Accurate and efficient simulation of rigid-body rotations' *Journal of Computational Physics* 164:377–406 doi:10.1006/jcph.2000.6602.

[9] Chen CW, JS Kouh, & JF Tsai 2013 'Modeling and simulation of an AUV simulator with guidance system' *IEEE Journal of Oceanic Engineering* 38(2):211–225 doi:10.1109/JOE.2012.2220236.

[10] Dosta M & V Skorych 2020 'MUSEN: An open-source framework for GPU-accelerated DEM simulations' *SoftwareX* 12:100618 doi:10.1016/j.softx.2020.100618.

[11] Fincham D 1992 'Leapfrog rotational algorithms' *Molecular Simulation* 8:165–178 doi:10.1080/08927029208022474.

[12] Goldstein H 2002 *Classical Mechanics* Pearson Education.

[13] Govender N, D Wilke, & S Kok 2016 'Blaze-DEMGPU: Modular high performance DEM framework for the GPU architecture' *SoftwareX* 5:62–66 doi:10.1016/j.softx.2016.04.004.

[14] Grassia FS 1998 'Practical parameterization of rotations using the exponential map' *Journal of Graphics Tools* 3:29–48 doi:10.1080/10867651.1998.10487493.

[15] Hamilton WR 1844 'LXXVIII. On quaternions; or on a new system of imaginaries in algebra' *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 25:489–495 doi:10.1080/14786444408645047.

[16] Hockney R & J Eastwood 1988 *Computer Simulation Using Particles* CRC Press doi: 10.1201/9780367806934.

[17] Itasca Consulting Group I 2008 '(2008a) PFC3D — particle flow code in 3 dimensions, version 4.0, user's manual.' Minneapolis: Itasca.

[18] Itasca Consulting Group I 2021 '(2021) PFC — particle flow code in 2 and 3 dimensions, version 7.0, documentation set of version 7.00.132' Minneapolis: Itasca.

[19] Jia YB 2017 'Quaternions: Com S 477/577 notes' Technical report Iowa State University URL `https://faculty.sites.iastate.edu/jia/files/inline-files/quaternion.pdf` accessed: 17-03-2023.

[20] Johnson MP 2003 *Exploiting Quaternions to Support Expressive Interactive Character Motion* Ph.D. thesis Massachusetts Institute of Technology.

[21] Johnson SM, JR Williams, & BK Cook 2008 'Quaternion-based rigid body rotation integration algorithms for use in particle methods' *International Journal for Numerical Methods in Engineering* 74:1303–1313 doi:10.1002/nme.2210.

[22] Kloss C, C Goniva, A Hager, S Amberger, & S Pirker 2012 'Models, algorithms and validation for opensource DEM and CFD–DEM' *Progress in Computational Fluid Dynamics, an International Journal* 12(2-3):140–152 doi:10.1504/PCFD.2012.047457.

[23] Kobilarov M, K Crane, & M Desbrun 2009 'Lie group integrators for animation and control of vehicles' *ACM Trans Graph* 28 doi:10.1145/1516522.1516527.

[24] Kuipers J 1999 *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality* Princeton paperbacks Princeton University Press.

[25] Kutta W 1901 'Beitrag zur näherungsweisen integration totaler differentialgleichungen' *Zeit Math Phys* 46:435–53.

[26] Longuski JM & P Tsiotras 1993 'Analytical solutions for a spinning rigid body subject to time-varying body-fixed torques, Part I: Constant Axial Torque' *Journal of Applied Mechanics* 60:970–975 doi:10.1115/1.2901010.

[27] Manchester ZR & MA Peck 2016 'Quaternion variational integrators for spacecraft dynamics' *Journal of Guidance, Control, and Dynamics* 39:69–76 doi:10.2514/1.G001176.

[28] Neto N & L Bellucci 2006 'A new algorithm for rigid body molecular dynamics' *Chemical Physics* 328:259–268 doi:10.1016/j.chemphys.2006.07.009.

[29] Omelyan IP 1998 'Algorithm for numerical integration of the rigid-body equations of motion' *Phys Rev E* 58:1169–1172 doi:10.1103/PhysRevE.58.1169.

[30] Ostanin I, V Angelidakis, T Plath, S Pourandi, A Thornton, & T Weinhart 2024 'Rigid clumps in the mercurydpm particle dynamics code' *Computer Physics Communications* 296:109034 doi:https://doi.org/10.1016/j.cpc.2023.109034.

[31] Pons A & F Cirak 2023 'Quaternion variational integration for inertial maneuvering in a biomimetic unmanned aerial vehicle' *ASME Letters in Dynamic Systems and Control* 3 doi:10.1115/1.4062685.

[32] Rackauckas C & Q Nie 2017 'Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia' *Journal of Open Research Software* doi:10.5334/jors.151.

[33] Romano M 2008 'Exact analytic solution for the rotation of a rigid body having spherical ellipsoid of inertia and subjected to a constant torque' *Celestial Mechanics and Dynamical Astronomy* 100:181–189 doi:10.1007/s10569-007-9112-7.

[34] Roots 2023 'Roots.jl' URL https://juliamath.github.io/Roots.jl/dev/ accessed: 17-03-2023.

[35] Rucker C 2018 'Integrating rotations using nonunit quaternions' *IEEE Robotics and Automation Letters* 3:2979–2986 doi:10.1109/LRA.2018.2849557.

[36] Sabatini AM 2005 'Quaternion-based strap-down integration method for applications of inertial sensing to gait analysis' *Medical and Biological Engineering and Computing* 43:94–101 doi:10.1007/BF02345128.

[37] Seelen LJH, JT Padding, & JAM Kuipers 2016 'Improved quaternion-based integration scheme for rigid body motion' *Acta Mechanica* 227:3381–3389 doi:10.1007/s00707-016-1670-x.

[38] Shah S, D Dey, C Lovett, & A Kapoor 2018 'AirSim: High-fidelity visual and physical simulation for autonomous vehicles' in M Hutter & R Siegwart (eds) *Field and Service Robotics* Cham: Springer International Publishing pp. 621–635.

[39] Shu CW & S Osher 1988 'Efficient implementation of essentially non-oscillatory shock-capturing schemes' *Journal of Computational Physics* 77:439–471 doi:10.1016/0021-9991(88)90177-5.

[40] Smilauer V, V Angelidakis, E Catalano, R Caulk, B Chareyre, W Chèvremont, S Dorofeenko, J Duriez, N Dyck, J Elias, B Er, A Eulitz, A Gladky, N Guo, C Jakob, F Kneib, J Kozicki, D Marzougui, R Maurin, C Modenese, G Pekmezi, L Scholtès, L Sibille, J Stransky, T Sweijen, K Thoeni, & C Yuan 2021 *Yade documentation* The Yade Project doi:10.5281/zenodo.5705394.

[41] Su J, C Schroeder, & R Fedkiw 2009 ‘Energy stability and fracture for frame rate rigid body simulations’ in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* New York, NY, USA: Association for Computing Machinery SCA ’09 p. 155–164 doi:10.1145/1599470.1599491.

[42] Syamlal M, W Rogers, & TJ O‘Brien 1993 ‘MFiX documentation theory guide’ Technical report USDOE Morgantown Energy Technology Center (METC), WV (United States)) doi:10.2172/10145548.

[43] Tadano S, R Takeda, & H Miyagawa 2013 ‘Three dimensional gait analysis using wearable acceleration and gyro sensors based on quaternion calculations’ *Sensors* 13:9321–9343 doi:10.3390/s130709321.

[44] Tayebi A & S McGilvray 2006 ‘Attitude stabilization of a VTOL quadrotor aircraft’ *IEEE Transactions on Control Systems Technology* 14:562–571 doi:10.1109/TCST.2006.872519.

[45] Thompson AP, HM Aktulga, R Berger, DS Bolintineanu, WM Brown, PS Crozier, PJ in ’t Veld, A Kohlmeyer, SG Moore, TD Nguyen, R Shan, MJ Stevens, J Tranchida, C Trott, & SJ Plimpton 2022 ‘LAMMPS - A flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales’ *Computer Physics Communications* 271:108171 doi:10.1016/j.cpc.2021.108171.

[46] Thornton C 2015 *Granular Dynamics, Contact Mechanics and Particle System Simulations: A DEM study* Particle Technology Series Springer International Publishing doi:10.1007/978-3-319-18711-2.

[47] Todorov E, T Erez, & Y Tassa 2012 ‘MuJoCo: A physics engine for model-based control’ in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 5026–5033 doi:10.1109/IROS.2012.6386109.

[48] Tsiotras P & JM Longuski 1996 ‘Analytic solution of Euler’s equations of motion for an asymmetric rigid body’ *Journal of Applied Mechanics* 63:149–155 doi:10.1115/1.2787190.

[49] unreal ‘Unity’ URL https://unity.com accessed: 28-07-2023.

[50] unreal ‘Unreal engine’ URL https://www.unrealengine.com accessed: 28-07-2023.

[51] Valenti RG, I Dryanovski, & J Xiao 2016 ‘A linear Kalman filter for MARG orientation estimation using the algebraic quaternion algorithm’ *IEEE Transactions on Instrumentation and Measurement* 65:467–481 doi:10.1109/TIM.2015.2498998.

[52] Verlet L 1967 ‘Computer ”experiments” on classical fluids. I. thermodynamical properties of Lennard-Jones molecules’ *Phys Rev* 159:98–103 doi:10.1103/PhysRev.159.98.

[53] vulkan 2023 ‘Vulkan’ URL https://www.vulkan.org/ accessed: 28-07-2023.

[54] Weinhart T, L Orefice, M Post, MP van Schrojenstein Lantman, IF Denissen, DR Tunuguntla, J Tsang, H Cheng, MY Shaheen, H Shi, P Rapino, E Grannonio, N Losacco, J Barbosa, L Jing, JE Alvarez Naranjo, S Roy, WK den Otter, & AR Thornton 2020 'Fast, flexible particle simulations — an introduction to MercuryDPM' *Computer Physics Communications* 249:107129 doi:10.1016/j.cpc.2019.107129.

[55] woodem 2023 'WooDEM' URL https://woodem.eu/ accessed: 21-06-2023.

[56] Wu Y, X Hu, D Hu, T Li, & J Lian 2005 'Strapdown inertial navigation system algorithms based on dual quaternions' *IEEE Transactions on Aerospace and Electronic Systems* 41:110–132 doi:10.1109/TAES.2005.1413751.

[57] Zhao F & BGM van Wachem 2013 'A novel quaternion integration approach for describing the behaviour of non-spherical particles' *Acta Mechanica* 224:3091–3109 doi:10.1007/s00707-013-0914-2.

Appendix A. Non-leapfrog variant of Spiral

To derive a non-leapfrog variant of Spiral, we follow the steps in Sec. 2. With $\mathrm{a} = \mathrm{t}$ in Eq. (4) we obtain

$$\mathrm{q}(\mathrm{t}+\Delta\mathrm{t}) = \mathrm{q}(\mathrm{t})\ \exp\left[\frac{\Delta\mathrm{t}}{2}\omega(\mathrm{t}) + \frac{\Delta\mathrm{t}^2}{4}\dot{\omega}(\mathrm{t}) + \mathcal{O}(\Delta\mathrm{t}^3)\right] , \tag{A.1}$$

which is the non-leapfrog version of Eq. (5). Using Eq. (6) we obtain

$$\mathrm{q}(\mathrm{t}+\Delta\mathrm{t}) = \mathrm{q}(\mathrm{t}) \left(\cos\theta_2 + \frac{\vec{\omega}\,(\mathrm{t})}{|\vec{\omega}\,(\mathrm{t})|}\sin\theta_2\right)\left(\cos\theta_3 + \frac{\dot{\vec{\omega}}\,(\mathrm{t})}{\left|\dot{\vec{\omega}}\,(\mathrm{t})\right|}\sin\theta_3\right) \tag{A.2}$$

with

$$\theta_2 = \frac{\Delta\mathrm{t}}{2}\left|\vec{\omega}\,(\mathrm{t})\right| , \qquad \theta_3 = \frac{\Delta\mathrm{t}^2}{4}\left|\dot{\vec{\omega}}\,(\mathrm{t})\right| . \tag{A.3}$$

The non-leapfrog variant does not need a backward step for initialization. At each time step, $\mathrm{q}(\mathrm{t})$ is updated before $\vec{\omega}(\mathrm{t})$. This order is necessary as the torque is required to compute $\dot{\vec{\omega}}$ with Eq. (8), which is only known at time t. Figs. A.8 and A.9 compare the leap-frog and non-leapfrog versions of Spiral with the scheme by Omelyan [29], which was the second best performing in Sec. 3, after Spiral. We observe that both versions of Spiral perform similarly in accuracy and error accumulation.

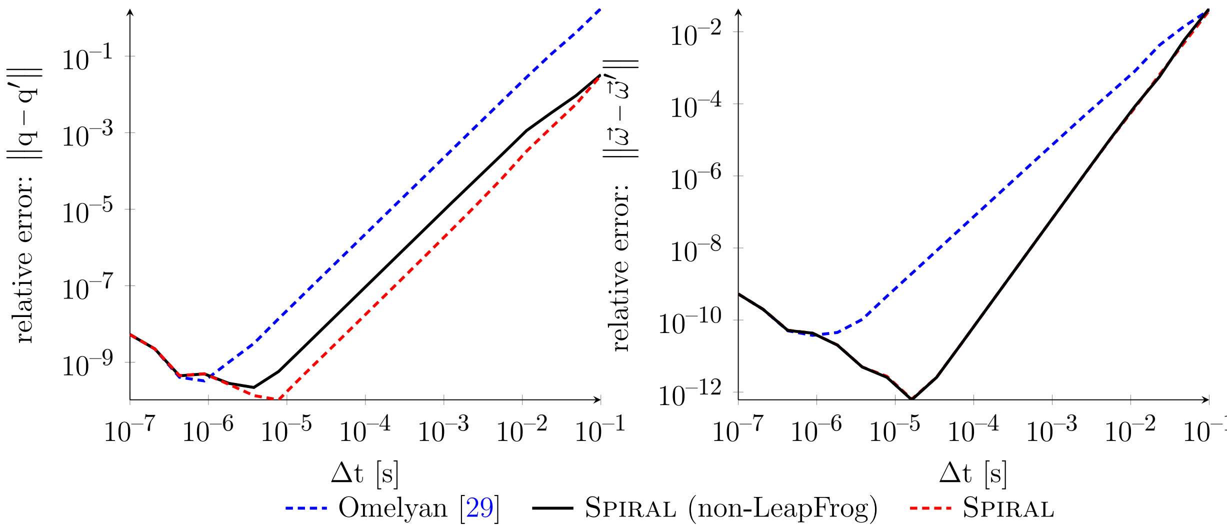


Figure A.8: Comparison of Spiral (leapfrog and non-leap frog variants) with the leapfrog scheme by Omelyan [29]: (left) relative error of the predicted quaternion with respect to the semi-analytical solution as a function of the time step, (right) relative error of the angular velocity with respect to the analytical solution as a function of the time step. The error was computed according to Eq. (13).

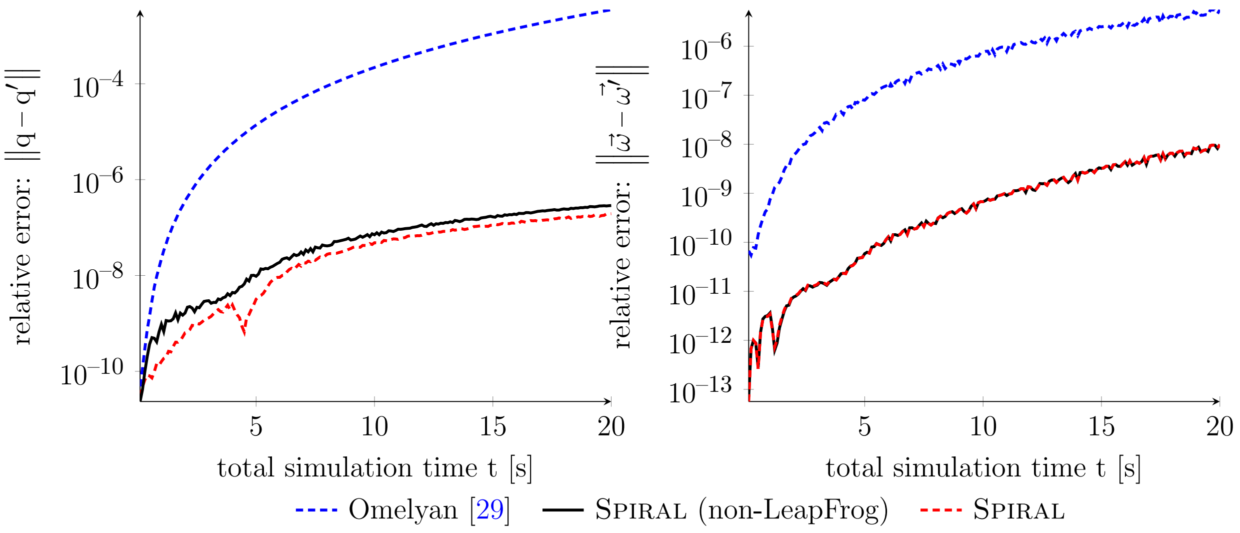


Figure A.9: Comparison of SPIRAL (leapfrog and non-leap frog variants) with the leapfrog scheme by Omelyan [29]; evolution of the error. The time step is $\Delta t = 10^{-5}$ s. (left) Relative error of the predicted quaternion with respect to the semi-analytical solution as a function of total simulation time. (right) Relative error of the angular velocity with respect to the analytical solution as a function of the total simulation time. The error was computed according to Eq. (13).

Appendix B. Rotation Algorithms

This Appendix provides details of the algorithms listed in Table 1 and used for comparison in Secs. 3, 4, 5, and Appendix A.

To prevent any confusion regarding the reference frames used in the algorithms, we establish the following notation: Quantities described in the principal axis reference frame of the rotating object appear like in the rest of this paper (the angular velocity $\vec{\omega}$, for instance). Variables in the lab frame have a superscript, $\vec{\omega}^{\mathrm{lab}}$. Furthermore, in the principal axis frame, the inertia tensor is represented as $\vec{\mathrm{I}}$, and it is treated mathematically as a vector because it is a diagonal matrix. In the lab frame, the inertia tensor is denoted as J and is a 3x3 symmetrical matrix. The quaternion representing the orientation is the same for both reference frames.

The angular velocity derivative, as described in Eq. 8, is formulated in the principal axis reference frame. This derivative is a function of the inertia tensor, the angular velocity, and the torque. Similarly, the quaternion derivative in Eq. 1 is also formulated in the principal axis frame. To calculate the derivative in the lab frame, you can use the equation $\dot{\mathrm{q}} = \frac{1}{2}\mathrm{q}\omega$. It is important to recognize that this equation, along with Eq. 1, requires the angular velocity to be converted into an imaginary quaternion. Alternatively, the quaternion derivative can be expressed by using matrix-vector multiplication. Consequently, you can represent $\dot{\mathrm{q}}$ as a function of the quaternion and the angular velocity vector [19]. For clarity, in the algorithm description, we show how to evaluate each derivative with brackets (for example, $\dot{\vec{\omega}}\left[\vec{\omega}, \vec{\mathrm{M}}\right]$).

Appendix B.1. Direct Euler

The second-order Direct Euler algorithm [7] updates the angular velocities and the quaternions through

$$\begin{aligned}\vec{\omega}(\mathrm{t}+\Delta\mathrm{t}) &= \vec{\omega}(\mathrm{t}) + \Delta\mathrm{t}\,\dot{\vec{\omega}}\left[\vec{\omega}(\mathrm{t}), \vec{\mathrm{M}}(\mathrm{t})\right] \\ \mathrm{q}(\mathrm{t}+\Delta\mathrm{t}) &= \mathrm{q}(\mathrm{t}) + \Delta\mathrm{t}\,\dot{\mathrm{q}}\left[\mathrm{q}(\mathrm{t}), \vec{\omega}(\mathrm{t}+\Delta\mathrm{t})\right] .\end{aligned} \tag{B.1}$$

This algorithm requires normalization of the quaternions in each time step. All quantities are in the principal axis frame.

Appendix B.2. Velocity Verlet

The second-order Velocity Verlet algorithm [52] updates the angular velocities and the quaternions through

$$\begin{aligned}\vec{\omega}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right) &= \vec{\omega}(\mathrm{t}) + \frac{\Delta\mathrm{t}}{2}\,\dot{\vec{\omega}}\left[\vec{\omega}(\mathrm{t}), \vec{\mathrm{M}}(\mathrm{t})\right] \\ \mathrm{q}(\mathrm{t}+\Delta\mathrm{t}) &= \mathrm{q}(\mathrm{t}) + \Delta\mathrm{t}\,\dot{\mathrm{q}}\left[\mathrm{q}(\mathrm{t}), \vec{\omega}(\mathrm{t}+\Delta\mathrm{t})\right] .\end{aligned} \tag{B.2}$$

Compute the torques, $\vec{\mathrm{M}}(\mathrm{t}+\Delta\mathrm{t})$, and then

$$\vec{\omega}(\mathrm{t}+\Delta\mathrm{t}) = \vec{\omega}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right) + \frac{\Delta\mathrm{t}}{2}\dot{\vec{\omega}}\left[\vec{\omega}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right), \vec{\mathrm{M}}(\mathrm{t}+\Delta\mathrm{t})\right] . \tag{B.3}$$

This algorithm requires normalization of the quaternions in each time step. All quantities are in the principal axis frame.

Appendix B.3. Fincham Leapfrog

The second-order Fincham Leapfrog algorithm [11] uses angular momenta instead of angular velocities. It includes a mid-step interpolation of the quaternion to increase the stability. The update from t to $\mathrm{t}+\Delta\mathrm{t}$ is performed in two steps. First, the angular momenta are updated as in Leapfrog schemes,

$$\vec{\mathrm{L}}^{\mathrm{lab}}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right)=\vec{\mathrm{L}}^{\mathrm{lab}}\left(\mathrm{t}-\frac{\Delta\mathrm{t}}{2}\right)+\Delta\mathrm{t}\,\vec{\mathrm{M}}^{\mathrm{lab}}(\mathrm{t})\,. \tag{B.4}$$

In this algorithm, the torque and angular momentum are in the lab reference frame. Then, we update the quaternion with a predictor-corrector scheme. Here, we need the derivative of the quaternion at different angular velocities,

$$\begin{aligned}\mathrm{q_a}&=\mathrm{q}+\frac{\Delta\mathrm{t}}{2}\,\dot{\mathrm{q}}\left[\mathrm{q}(\mathrm{t}),\vec{\omega}(\mathrm{t})\right]\\ \mathrm{q}(\mathrm{t}+\Delta\mathrm{t})&=\mathrm{q}+\Delta\mathrm{t}\,\dot{\mathrm{q}}\left[\mathrm{q_a},\vec{\omega}(\mathrm{t})\right].\end{aligned} \tag{B.5}$$

The angular velocities in the rotating body-fixed frame can be calculated as

$$\begin{aligned}\vec{\omega}(\mathrm{t})&=\hat{\mathrm{A}}\left[\vec{\mathrm{L}}\left(\mathrm{t}-\frac{\Delta\mathrm{t}}{2}\right)+\frac{\Delta\mathrm{t}}{2}\,\vec{\mathrm{M}}(\mathrm{t})\right]\\ \vec{\omega}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right)&=\hat{\mathrm{A}}\,\vec{\mathrm{L}}\left(\mathrm{t}+\frac{\Delta\mathrm{t}}{2}\right),\end{aligned} \tag{B.6}$$

where $\hat{\mathrm{A}}$ is the rotation matrix to translate from lab-fixed to body-fixed coordinates. The matrix can be calculated from the quaternion components. This algorithm requires normalization of the quaternion.

Appendix B.4. Buss Algorithm

Unlike the above-described algorithms, the second-order Buss algorithm [8] updates the inertia tensor and uses the quantities in the lab frame. Before the update, we compute

$$\begin{aligned}\mathrm{J}(\mathrm{t})&=\hat{\mathrm{A}}^{-1}(\mathrm{t})\,\vec{\mathrm{I}}\\ \vec{\omega}^{\mathrm{lab}}(\mathrm{t})&=\mathrm{J}^{-1}(\mathrm{t})\,\vec{\mathrm{L}}^{\mathrm{lab}}(\mathrm{t})\\ \dot{\vec{\omega}}^{\mathrm{lab}}(\mathrm{t})&=\mathrm{J}^{-1}(\mathrm{t})\left[\vec{\mathrm{M}}^{\mathrm{lab}}(\mathrm{t})-\vec{\omega}^{\mathrm{lab}}(\mathrm{t})\times\vec{\mathrm{L}}^{\mathrm{lab}}(\mathrm{t})\right].\end{aligned} \tag{B.7}$$

Here, $\hat{\mathrm{A}}$ is the rotation matrix for transforming lab-fixed into body-fixed coordinates, and J is the inertia tensor matrix in the lab frame, calculated from the inertia tensor in the principal axis frame $\vec{\mathrm{I}}$. The rotation matrix can be computed from the quaternion components.

The update proceeds in two steps. First, let us compute

$$\begin{aligned}
\bar{\vec{\omega}} &= \vec{\omega}^{\mathrm{lab}}(\mathrm{t}) + \frac{\Delta \mathrm{t}}{2}\,\dot{\vec{\omega}}^{\mathrm{lab}}(\mathrm{t}) + \frac{\Delta \mathrm{t}^2}{12}\left[\dot{\vec{\omega}}^{\mathrm{lab}}(\mathrm{t}) \times \vec{\omega}^{\mathrm{lab}}(\mathrm{t})\right] \\
\theta &= \left|\bar{\vec{\omega}}\right|\,\Delta \mathrm{t}
\end{aligned} \tag{B.8}$$

and then

$$\begin{aligned}
\mathrm{q}(\mathrm{t}+\Delta \mathrm{t}) &= \left[\cos\frac{\theta}{2} + \frac{\bar{\vec{\omega}}}{|\bar{\vec{\omega}}|}\sin\frac{\theta}{2}\right]\mathrm{q}(\mathrm{t}) \\
\vec{\mathrm{L}}^{\mathrm{lab}}(\mathrm{t}+\Delta \mathrm{t}) &= \vec{\mathrm{L}}^{\mathrm{lab}}(\mathrm{t}) + \Delta \mathrm{t}\,\vec{\mathrm{M}}^{\mathrm{lab}}(\mathrm{t})\,.
\end{aligned} \tag{B.9}$$

The version of the Buss algorithm presented here is the same as the one in PFC7 documentation [18]. Buss' original version [8] uses rotation matrices instead of quaternions.

Appendix B.5. Omelyan Algorithm

Omelyan's third-order Advance Leapfrog algorithm [29] proceeds as follows:

$$\begin{aligned}
\vec{\omega}\left(\mathrm{t}+\frac{\Delta \mathrm{t}}{2}\right) &= \vec{\omega}\left(\mathrm{t}-\frac{\Delta \mathrm{t}}{2}\right) + \Delta \mathrm{t}\,\dot{\vec{\omega}}\left[\vec{\omega}(\mathrm{t}), \vec{\mathrm{M}}(\mathrm{t})\right] \\
\mathrm{q}(\mathrm{t}+\Delta \mathrm{t}) &= \frac{\left(1-\frac{\Delta \mathrm{t}^2}{16}\left|\vec{\omega}\left(\mathrm{t}+\frac{\Delta \mathrm{t}}{2}\right)\right|^2\right)\mathbb{1} + \Delta \mathrm{t}\,\dot{\mathrm{q}}\left[\mathrm{q}(\mathrm{t}), \vec{\omega}(\mathrm{t}+\frac{\Delta \mathrm{t}}{2})\right]}{1+\frac{\Delta \mathrm{t}^2}{16}\left|\vec{\omega}\left(\mathrm{t}+\frac{\Delta \mathrm{t}}{2}\right)\right|^2}\,\mathrm{q}(\mathrm{t})\,,
\end{aligned} \tag{B.10}$$

where $\mathbb{1}$ is the identity quaternion. Eq. (B.10) uses the derivative of the angular velocity at time t. Comparison with Eq. (8) reveals that the computation of $\dot{\vec{\omega}}(\mathrm{t})$ would require the angular velocity at time t, which is yet unknown. To solve this problem, Omelyan proposed the approximation

$$\omega_\alpha(\mathrm{t})\,\omega_\beta(\mathrm{t}) \approx \frac{1}{2}\left[\omega_\alpha\left(\mathrm{t}-\frac{\Delta \mathrm{t}}{2}\right)\omega_\beta\left(\mathrm{t}-\frac{\Delta \mathrm{t}}{2}\right) + \omega_\alpha\left(\mathrm{t}+\frac{\Delta \mathrm{t}}{2}\right)\omega_\beta\left(\mathrm{t}+\frac{\Delta \mathrm{t}}{2}\right)\right]. \tag{B.11}$$

This approximation turns Eqs. (B.10) into a system of nonlinear equations that can be efficiently solved by iteration, with the initial guess

$$\vec{\omega}^{(0)}(\mathrm{t}) = \vec{\omega}\left(\mathrm{t}-\frac{\Delta \mathrm{t}}{2}\right). \tag{B.12}$$

Experience shows that - for a sufficiently small time step - about three iterations are enough to achieve convergence. In this paper, we always used three iterations.

By construction, this algorithm preserves the norm of the quaternions. As a result, renormalization shouldn't be necessary; however, we found that the algorithm accumulates significant numerical errors over time and can become unstable. We suggest sporadically normalizing the quaternion when using this algorithm.

Appendix B.6. Johnson (2008) Algorithm

Johnson's algorithm [21] is based on a Runge-Kutta-4 scheme for the update of the quaternion:

$$q(t+\Delta t) = q(t) + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) , \tag{B.13}$$

with

$$\begin{aligned} k_1 &= \dot{q}\left[q(t),\ \vec{\omega}(t)\right] \\ k_2 &= \dot{q}\left[q(t) + \Delta t\,\frac{k_1}{2},\ \vec{\omega}(t)\right] \\ k_3 &= \dot{q}\left[q(t) + \Delta t\,\frac{k_2}{2},\ \vec{\omega}(t)\right] \\ k_4 &= \dot{q}\left[q(t) + \Delta t\,k_3,\ \vec{\omega}(t)\right] . \end{aligned} \tag{B.14}$$

Next, the angular momentum is updated as

$$\vec{L}^{\mathrm{lab}}(t+\Delta t) = \vec{L}^{\mathrm{lab}}(t) + \Delta t \vec{M}^{\mathrm{lab}}(t) . \tag{B.15}$$

Note that Johnson suggests using a second-order predictor-corrector algorithm for evolving angular momentum [21]. This would require two force calculations and two quaternion updates per time step; nevertheless, for the tests in Sec. 3 the algorithm wouldn't have benefited from that predictor-corrector scheme, because the torque is given and constant. This algorithm requires the quaternion to be normalized.

Appendix B.7. PFC4/ MercuryDPM

The algorithm used by MercuryDPM [54] for non-spherical particles is described in the PFC4 manual [17] and [30]. It uses rotation matrices to represent the particle orientation. We are not aware of the algorithm's name or its original reference. The current version of PFC [18] uses either the Buss [8] or the Johnson algorithm [21]. In PFC4 all the quantities are in the lab reference frame.

The algorithm proceeds as follows: First, the angular velocity and momentum are iteratively updated until convergence as

$$\begin{aligned} \vec{L}_n^{\mathrm{lab}} &= J\,\vec{\omega}_n^{\mathrm{lab}} \\ \vec{\omega}_{n+1}^{\mathrm{lab}} &= \vec{\omega}_0^{\mathrm{lab}} + \Delta t\,J^{-1}\left(\vec{M}^{\mathrm{lab}} - \vec{\omega}_n^{\mathrm{lab}} \times \vec{L}_n^{\mathrm{lab}}\right) . \end{aligned} \tag{B.16}$$

Then, the orientation is updated as

$$\hat{A}(t+\Delta t) = \hat{A}(t) + \Delta t\,\dot{\hat{A}}(t) , \tag{B.17}$$

where $\hat{A}$ is the rotation matrix representing the particle's orientation. Its derivative reads

$$\dot{\hat{A}} = \varepsilon_{ijk}\,\omega_j^{\mathrm{lab}}\,\hat{A}_{mk} , \tag{B.18}$$

where ε_{ijk} is the Levi-Civita symbol. Einstein's summation notation applies. In this paper, we used three iterations for convergence, as in MercuryDPM.

Appendix C. A special solution of Euler's equations of motion

Appendix C.1. The case of constant driving, $\mathrm{M_x} \neq 0$

We consider a rigid body with principal moments of inertia $\mathrm{I_x} \neq \mathrm{I_y} = \mathrm{I_z}$, driven by a constant torque $\vec{\mathrm{M}} = \{\mathrm{M_x}, 0, 0\}$ with $\mathrm{M_x} \neq 0$. Under these assumptions, Euler's rigid body equations of motion in the principal axis frame, Eq. (8), reduces to

$$\begin{aligned} \mathrm{I_x}\,\dot{\omega}_\mathrm{x} &= \mathrm{M_x} & \text{(C.1)}\\ \mathrm{I_y}\,\dot{\omega}_\mathrm{y} &= \omega_\mathrm{z}\,\omega_\mathrm{x}\,(\mathrm{I_z} - \mathrm{I_x}) & \text{(C.2)}\\ \mathrm{I_z}\,\dot{\omega}_\mathrm{z} &= \omega_\mathrm{x}\,\omega_\mathrm{y}\,\left(\mathrm{I_x} - \mathrm{I_y}\right)\,. & \text{(C.3)} \end{aligned}$$

The solution of Eq. (C.1) is trivial,

$$\omega_\mathrm{x}(\mathrm{t}) = \omega_\mathrm{x}^0 + \frac{\mathrm{M_x}}{\mathrm{I_x}}\,\mathrm{t}\,, \tag{C.4}$$

with $\omega_\mathrm{x}^0 \equiv \omega_\mathrm{x}(0)$. To solve Eqs. (C.2, C.3), take the time derivative of Eq. (C.2),

$$\frac{\mathrm{I_y}}{\mathrm{I_z} - \mathrm{I_x}}\,\ddot{\omega}_\mathrm{y} = \dot{\omega}_\mathrm{z}\omega_\mathrm{x} + \omega_\mathrm{z}\dot{\omega}_\mathrm{x}\,, \tag{C.5}$$

and substitute $\dot{\omega}_\mathrm{z}$ obtained from this equation into Eq. C.3, to obtain

$$\frac{\mathrm{I_z}}{\mathrm{I_x} - \mathrm{I_y}}\left(\frac{\mathrm{I_y}}{\mathrm{I_z} - \mathrm{I_x}}\,\frac{1}{\omega_\mathrm{x}}\,\ddot{\omega}_\mathrm{y} - \frac{\dot{\omega}_\mathrm{x}}{\omega_\mathrm{x}}\,\omega_\mathrm{z}\right) = \omega_\mathrm{x}\,\omega_\mathrm{y}\,. \tag{C.6}$$

We also substitute ω_z as obtained from Eq. (C.2) into Eq. (C.6), to obtain

$$\frac{\mathrm{I_z}}{\mathrm{I_x} - \mathrm{I_y}}\left(\frac{\mathrm{I_y}}{\mathrm{I_z} - \mathrm{I_x}}\,\frac{1}{\omega_\mathrm{x}}\ddot{\omega}_\mathrm{y} - \frac{\dot{\omega}_\mathrm{x}}{\omega_\mathrm{x}}\,\frac{\mathrm{I_y}}{\mathrm{I_z} - \mathrm{I_x}}\,\frac{1}{\omega_\mathrm{x}}\dot{\omega}_\mathrm{y}\right) = \omega_\mathrm{x}\omega_\mathrm{y}\,. \tag{C.7}$$

Equation (C.7) is true if $\omega_\mathrm{x} \neq 0$. It can be written conveniently as

$$\ddot{\omega}_\mathrm{y} - \frac{\dot{\omega}_\mathrm{x}}{\omega_\mathrm{x}}\,\dot{\omega}_\mathrm{y} = \mathrm{B}\,\omega_\mathrm{x}^2\,\omega_\mathrm{y} \tag{C.8}$$

with

$$\mathrm{B} \equiv \frac{(\mathrm{I_x} - \mathrm{I_y})(\mathrm{I_z} - \mathrm{I_x})}{\mathrm{I_z}\mathrm{I_y}}\,. \tag{C.9}$$

According to our initial assumption, $\mathrm{I_x} \neq \mathrm{I_y} = \mathrm{I_z}$, we define $\mathrm{I_*} \equiv \mathrm{I_y} = \mathrm{I_z}$ and, thus,

$$\mathrm{B} = -\left(\frac{\mathrm{I_x} - \mathrm{I_*}}{\mathrm{I_*}}\right)^2 < 0\,. \tag{C.10}$$

Since $\omega_x(t)$ is an explicit function of time (Eq. (C.4)), we can use ω_x as a time scale and replace derivatives d/dt by $d/d\omega_x$. We obtain

$$\begin{aligned} \frac{d\omega_y}{dt} &= \frac{d\omega_y}{d\omega_x}\frac{d\omega_x}{dt} = \frac{M_x}{I_x}\frac{d\omega_y}{d\omega_x} \\ \frac{d^2\omega_y}{dt^2} &= \left(\frac{M_x}{I_x}\right)^2 \frac{d^2\omega_y}{d\omega_x^2}\,. \end{aligned} \tag{C.11}$$

Equation (C.8) reads, then,

$$\frac{d^2\omega_y}{d\omega_x^2} - \frac{1}{\omega_x}\frac{d\omega_y}{d\omega_x} = \left(\frac{I_x}{M_x}\right)^2 B\,\omega_x^2\,\omega_y\,, \tag{C.12}$$

provided $M_x \neq 0$. Substituting $\tau \equiv \omega_x^2$ yields

$$\begin{aligned} \frac{d\omega_y}{d\omega_x} &= \frac{d\omega_y}{d\tau}\frac{d\tau}{d\omega_x} = 2\,\omega_x\frac{d\omega_y}{d\tau} \\ \frac{d^2\omega_y}{d\omega_x^2} &= 2\frac{d\omega_y}{d\tau} + 2\,\omega_x\frac{d}{d\omega_x}\frac{d\omega_y}{d\tau} = 2\frac{d\omega_y}{d\tau} + 4\tau\frac{d^2\omega_y}{d\tau^2}\,, \end{aligned} \tag{C.13}$$

and Eq. (C.12) becomes

$$\frac{d^2\omega_y}{d\tau^2} = \left(\frac{I_x}{2M_x}\right)^2 B\,\omega_y\,. \tag{C.14}$$

The last expression is the equation of motion for a harmonic oscillator, which we recast as

$$\frac{d^2\omega_y}{d\tau^2} = -\Omega^2\,\omega_y\,, \tag{C.15}$$

with

$$\Omega^2 \equiv -\left(\frac{I_x}{2M_x}\right)^2 B\,. \tag{C.16}$$

(Recall that $B < 0$, see Eq. (C.10)). Its solution reads

$$\omega_y(\tau) = k_1\cos(\Omega\tau) + k_2\sin(\Omega\tau)\,, \tag{C.17}$$

whose coefficients k_1 and k_2 shall be determined from the initial conditions. We derive a corresponding equation from Eq. (C.2) by using Eqs. (C.11,C.13,C.14), obtaining

$$\omega_z = \frac{I_y}{I_z - I_x}\frac{1}{\omega_x}\frac{d\omega_y}{dt} = \frac{I_y}{I_z - I_x}\frac{1}{\omega_x}\frac{M_x}{I_x}\frac{d\omega_y}{d\omega_x} = \frac{I_y}{I_z - I_x}\,2\,\frac{M_x}{I_x}\frac{d\omega_y}{d\tau} = \eta\,\frac{d\omega_y}{d\tau}\,, \tag{C.18}$$

with

$$\eta \equiv \frac{I_y}{I_z - I_x}\frac{2\,M_x}{I_x}\,. \tag{C.19}$$

Thus, from Eq. (C.17) we obtain

$$\omega_{\mathrm{z}}(\tau) = \eta\Omega\left(\mathrm{k}_2\cos(\,\Omega\tau\,) - \mathrm{k}_1\sin(\,\Omega\tau\,)\right)\,. \tag{C.20}$$

The set of Eqs. (C.4, C.17, C.20) form the complete solution of Euler's Equations (C.1,C.2,C.3). For t = 0, we find $\tau = \left(\omega_{\mathrm{x}}^0\right)^2$, according to its definition. With this result on hand, we obtain $\omega_{\mathrm{y}}^0 \equiv \omega_{\mathrm{y}}(\mathrm{t}=0)$ and $\omega_{\mathrm{z}}^0 \equiv \omega_{\mathrm{z}}(\mathrm{t}=0)$ from Eqs. (C.17,C.20) as

$$\begin{aligned}
\omega_{\mathrm{y}}^0 &= \mathrm{k}_1\,\cos\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right) + \mathrm{k}_2\,\sin\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right)\\
\omega_{\mathrm{z}}^0 &= \eta\Omega\left[\mathrm{k}_2\,\cos\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right) - \mathrm{k}_1\,\sin\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right)\right]\,.
\end{aligned} \tag{C.21}$$

By solving the linear set of Equations (C.21) we determine the parameters k_1 and k_2 from the initial velocities $\left(\omega_{\mathrm{x}}^0, \omega_{\mathrm{y}}^0, \omega_{\mathrm{z}}^0\right)$. With the definitions of Ω and η (Eqs. (C.16) and (C.19)) we get

$$\begin{aligned}
\mathrm{k}_1 &= \omega_{\mathrm{y}}^0\cos\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right) - \frac{\omega_{\mathrm{z}}^0}{\Omega\eta}\sin\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right)\\
\mathrm{k}_2 &= \omega_{\mathrm{y}}^0\sin\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right) + \frac{\omega_{\mathrm{z}}^0}{\Omega\eta}\cos\left(\Omega\left(\omega_{\mathrm{x}}^0\right)^2\right),
\end{aligned} \tag{C.22}$$

which concludes the analytical solution of Eqs. (C.1,C.2,C.3).

Appendix C.2. The undriven rotator, $\mathrm{M}_{\mathrm{x}} = 0$

For this special case, we can see from (C.1) that

$$\omega_{\mathrm{x}}(\mathrm{t}) = \omega_{\mathrm{x}}^0\,. \tag{C.23}$$

By inserting $\dot{\omega}_{\mathrm{z}}$ from Eq. (C.3) into the time derivative of Eq. (C.2), we obtain

$$\ddot{\omega_{\mathrm{y}}} = \frac{\mathrm{I}_{\mathrm{z}} - \mathrm{I}_{\mathrm{x}}}{\mathrm{I}_{\mathrm{y}}}\omega_{\mathrm{x}}^0\dot{\omega}_{\mathrm{z}} = \left(\omega_{\mathrm{x}}^0\right)^2\frac{\mathrm{I}_{\mathrm{z}} - \mathrm{I}_{\mathrm{x}}}{\mathrm{I}_{\mathrm{y}}}\frac{\mathrm{I}_{\mathrm{x}} - \mathrm{I}_{\mathrm{y}}}{\mathrm{I}_{\mathrm{z}}}\,\omega_{\mathrm{y}}\,. \tag{C.24}$$

This is again a harmonic oscillator equation,

$$\ddot{\omega_{\mathrm{y}}} = -\Gamma^2\,\omega_{\mathrm{y}}\,, \tag{C.25}$$

with $\Gamma > 0$ for $\mathrm{I}_{\mathrm{x}} \neq \mathrm{I}_{\mathrm{y}} = \mathrm{I}_{\mathrm{z}}$. Using its formal solution for $\omega_{\mathrm{y}}(\mathrm{t})$, we can solve the equation

$$\omega_{\mathrm{z}}(\mathrm{t}) = \frac{\mathrm{I}_{\mathrm{y}}}{\mathrm{I}_{\mathrm{z}} - \mathrm{I}_{\mathrm{x}}}\frac{1}{\omega_{\mathrm{x}}^0}\dot{\omega_{\mathrm{y}}} = \zeta\dot{\omega_{\mathrm{y}}} \tag{C.26}$$

and, thereafter, the set (C.23,C.25, C.26) can be solved in the same way as shown in the preceding subsection.