A parallel-GPU code for asteroid aggregation problems with angular particles

Fabio Ferrari [1,2]★ Michèle Lavagna[1] and Emmanuel Blazquez[1]

[1]*Department of Aerospace Science and Technology, Politecnico di Milano, Via La Masa 34, Milan I-20156, Italy*
[2]*Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Dr, Pasadena, CA 91109, USA*

ABSTRACT

The paper presents a numerical implementation of the gravitational N-body problem with contact interactions between non-spherically shaped bodies. The work builds up on a previous implementation of the code and extends its capabilities. The number of bodies handled is significantly increased through the use of a CUDA/GPU-parallel octree structure. The implementation of the code is discussed and its performance is compared against direct N^2 integration. The code features both smooth (force-based) and non-smooth (impulse-based) methods, as well as a visco-elastic non-smooth method, to handle contact interaction between bodies. The numerical problem of simulating 'rubble-pile' asteroid gravitational aggregation processes is addressed. We discuss the features of the problem and derive criteria to set up the numerical simulation from the dynamical constraints of the combined gravitational–collisional problem. Examples of asteroid aggregation scenarios that could benefit from such implementation are finally presented.

Key words: gravitation – methods: numerical – minor planets, asteroids: general – planets and satellites: dynamical evolution and stability – planets and satellites: formation.

1 INTRODUCTION

The numerical resolution of the N-body problem is one of the most popular problems in high-performance computing. The intrinsic $O(N^2)$ complexity of the problem has challenged many astronomers and computer scientists since the beginning of the computing era. N-body algorithms have evolved side by side with the increase of computing power and availability of new resources, such as Graphic Processing Units (GPUs). State-of-the-art implementations include parallel N-body tree codes (Stadel 2001; Morbidelli 2002; Richardson et al. 2002a), hybrid codes (Aarseth 1999, 2001; Wang et al. 2015), adaptive algorithms of optimal orders (Pruett, Rudmin & Lacy 2003), systolic algorithms (Dorband, Hemsendorf & Merritt 2003), or more generally symplectic codes (Wisdom & Holman 1991; Duncan, Levison & Lee 1998; Chambers 1999). In order to be able to solve granular dynamics problems, the capability of gravitational N-body codes needs to be extended to include contact interaction between non-point-like bodies. Two main classes of methods are commonly used to deal with contact interactions: hard- and soft-body methods. The first considers impulsive contacts between non-deformable particles (Alder & Wainwright 1959; Jean & Moreau 1987). This method is numerically very stable and allows us to simulate effectively the dynamics of hundreds

of thousands of bodies. In more recent years, the alternative class of soft-body methods has been developed (Cundall & Strack 1979). Based on a force-driven approach, soft-body methods are more suitable than hard-body ones for the simulation of smooth dynamics. These could be used to complement some deficiencies of hard-body models. For example, due to the impulsive nature of contact exchanged, hard-body methods are not adequate to reproduce phenomena such as wave propagation in granular media (Gilardi & Sharf 2002). However, the numerical solution of soft-body methods is more unstable and often requires extremely small time-steps to adequately reproduce elastic forces at contact. Both hard- and soft-body models have their own advantages and shortcomings and their use shall be carefully weighted upon the application scenario. As a general rule, hard-body methods qualify to simulate non-smooth dynamics, and vice versa: Soft-body method is suitable for smooth problems. However, whether a problem should be classified in either one category is a controversial field of debate: both the models rely on sets of non-directly measurable parameters, which often offer poor physical interpretation when compared to real-world scenarios (Dubois, Acary & Jean 2018). To date, both hard- and soft-body methods have been successfully used to simulate a wide variety of planetary science scenarios: Richardson et al. (2002b) used their hard-body model to simulate gravitational aggregation of rubble-pile asteroids and planetary ring dynamics (Porco et al. 2008); Wada, Senshu & Matsui (2006) used a soft-body approach to study cratering impacts under constant gravity field, and more recently,

★ E-mail: fabio1.ferrari@polimi.it

soft-sphere methods with mutual gravity between bodies have been used to simulate rubble-pile scenarios (Sánchez & Scheeres 2011; Schwartz, Richardson & Michel 2012; Tancredi et al. 2012). Another key aspect for a realistic evaluation of contact interactions concerns the shape of interacting bodies. All the aforementioned codes are limited to spherically shaped bodies. The use of spherical shapes is now considered to be a very relevant simplification for problems where contact between bodies happens. The use of spheres allows us to drastically reduce the computational burden, but can heavily affect the realism of the model when solving for the contact interaction between bodies (Michel, Benz & Richardson 2004). As such, the determination of the final shape of the aggregates and fragments could not be addressed (Richardson et al. 2009). Few attempts have been made using hard-body polyhedral grains together with video game engines to solve for contact interactions (Movshovitz, Asphaug & Korycansky 2012). Although polyhedral bodies provide a better modelling, compared to spheres, of some physical phenomena (e.g. related to interlocking between bodies), the computational accuracy provided by game engines is very low (single precision) and not able to satisfy basic energy or angular momentum conservation requirements. Recently, Ferrari et al. (2017) implemented the *N*-body aggregation problem with non-spherical shapes and non-smooth contacts, using the modules of Chrono::Engine (C::E) (Tasora et al. 2016; Tasora 2019), an open software optimized for granular and multibody engineering problems and able to handle contacts and collisions of a large number of complex-shaped bodies. The results obtained were very promising and satisfactorily accurate in terms of energy and angular momentum conservation. In their work, Ferrari et al. (2017) made use of direct *N*-body simulations to solve for gravitational dynamics, and only few thousands of bodies could be handled by the simulator.

The work presented here builds upon the work by Ferrari et al. (2017) and discusses the implementation of a parallel CUDA-GPU octree structure to extend the capabilities of the code to a higher number of complex-shaped bodies. As its parent code, the implementation is based on a rework of the C::E multiphysics simulation engine to simulate collisions between non-spherical bodies and integrate the dynamics of the problem. The code features both non-smooth (impulse-based) and smooth (force-based) methods to handle contact interactions. Also, a compliant non-smooth method is featured and proposed for the first time for planetary science application, in the attempt to joint the advantages of numerical stability provided by its impulse-based formulation, together with its ability of simulating compliance at contact level. Section 2 discusses the general architecture of the code and presents methods available to solve gravitational and collisional dynamics. The performance of the code is discussed in Section 3, where the results of validation/test problems are reported. The applicability to asteroid aggregation problems is eventually discussed through simulation examples. We address the problem of rubble-pile asteroid aggregation processes in Section 4. The physics of gravitational and collisional processes involved are reviewed and their numerical modelling is discussed. We derive a set of qualitative and quantitative requirements from dynamical and numerical constraints of the gravitational–collisional problem.

2 NUMERICAL IMPLEMENTATION

The code is built using a modular approach, taking advantage of the object-oriented C++ architecture of C::E library. Modules represent interchangeable units of software, easily accessible by the user. Each module contains methods and routines to execute specific tasks. The main modules developed and/or directly retrieved by C::E libraries are as follows:

    (i) gravity
    (ii) contact
    (iii) rigid-body dynamics
    (iv) body creation
    (v) numerical solvers
    (vi) data input/output
    (vii) visual interface
    (viii) post-processing

Modules (ii), (v), and (vii) are directly adapted from C::E libraries, modules (iii) and (iv) are based on the C::E libraries and further extended with additional methods and routines, and modules (i), (vi), and (viii) are fully developed by the authors. Modules (vi) and (vii) provide routines that allow the user to interface with the code at a higher level than software implementation level. The visual interface is based on Irrlicht library (Gebhardt 2016) and can be set to highlight desired geometrical or simulation features (forces, contacts), as well as to display real-time simulation outputs. The data input/output interface is straightforward: Every user-tunable parameter of the model can be assigned via an input text file that is parsed and acquired by the code, while output data are typically saved into dedicated text files. This section presents a general overview of the numerical methods available in each module, focusing on those relevant to the gravitational aggregation problem.

2.1 Gravitational dynamics

Module (i) encloses methods and routines used to compute the gravity forces acting on each body. These are based on the evaluation of gravity field produced by point-mass or shape-based (e.g. polyhedron and ellipsoid) gravity sources. To simulate the problem of gravitational aggregation, the classical *N*-body problem with point-mass sources is considered. This is a common and reasonable assumption: For a high number of commensurate interacting bodies, the use of shape-based mutual gravity would impact dramatically on the computational effort and would not provide relevant benefits to the accuracy of results, as discussed in Section 4.1. As detailed below, module (i) includes both direct N^2 integration and a parallel-GPU octree method.

2.1.1 The N-body problem

The equations of motion of the *N*-body problem are typically written by using Newton's law to compute the gravitational interactions between *N* masses:

$$m_i \ddot{\boldsymbol{R}}_i = G \sum_{j=1, j \neq i}^{N} \frac{m_i m_j}{||\boldsymbol{R}_{ij}||^3} \boldsymbol{R}_{ij} \qquad \forall i = 1 : N, \tag{1}$$

where $\boldsymbol{R}_i$ represents the position vector of the *i*-th body in an inertial frame, m_i its mass, G the universal gravitational constant, and $\boldsymbol{R}_{ij} = \boldsymbol{R}_j - \boldsymbol{R}_i$. A method implementing all *N*-to-*N* interactions is commonly referred as direct N^2 or particle–particle method (Hockney & Eastwood 1988; Hut & Makino 2010). Such a method is very simple to implement and provides exact computation of the motion of the *N*-bodies without any physical approximation. The accuracy of the result depends solely on the numerical errors of the solver. The simple dynamics of direct N^2 method make
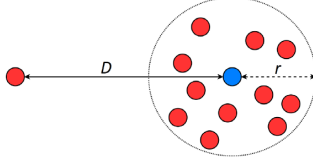
Figure 1. Schematics of BH clustering criteria.

it very easy to implement and parallelize. However, they are extremely inefficient from a computational point of view: $N(N-1)$ interactions must be computed for each body involved in the simulation and this leads to a time complexity of $O(N^2)$. A direct consequence is that such codes are limited in terms of the maximum number of bodies to be handled. The bottlenecks are simulation time that becomes prohibitively long, and hardware capabilities with computer running out of memory, both effects occurring very rapidly as N increases. The direct N^2 method, even with advanced parallelism on processing units, is not suitable for efficient asteroid aggregation simulations that can involve more than few thousands of bodies.

2.1.2 Hierarchical treecode algorithm

Algorithms based on tree data structures rely on more dynamic and adaptive computations that allow for a significant reduction of time complexity up to $O(N\log(N))$. The Barnes–Hut (also referenced as BH in the following) algorithm (Barnes & Hut 1986) groups particles using a hierarchy of cube structures. A node in the algorithm corresponds to a cube in physical space. Because of the use of octrees, each node has eight child nodes obtained by a simple homogeneous spatial subdivision performed along the three principal axes of the system. The tree is therefore built by recursive subdivision until each node of the tree contains zero or one particle. The structure is adaptive, implying that the size of the tree is not fixed but comes as a result of the repartition of the particles in the 3D space. The data structure can grow naturally to more levels in regions where the particle density is high. The octree is rebuilt at each time-step[1] of the dynamic simulation and consists of bodies stored at terminal nodes, called leaf nodes, and intermediary internal nodes that behave as clusters of particles. The algorithm uses the newly obtained adaptive octree data structure to compute the centre of mass of the cubes involved in the simulation and the forces applied to each body: body-to-body and body-to-node interactions are considered.

2.1.3 Clustering

The algorithm relies on the idea that the force generated by a cluster of bodies can be approximated by treating the cluster as a single body. The accuracy of the approximations depends on the distance D of the cluster from the body and the radius r of the cluster of particles, as shown in Fig. 1. We can therefore define the accuracy through $\theta = r/D$, a user-tunable parameter used to set the error of the method. The position of centre of mass and total mass of the node is computed for each leaf and internal node of the BH tree.

[1]See Section 4.1 for more details on the choice of simulation time-step(s).

For a given internal node $\mathcal{N}_{\text{int}}$:

$$\begin{cases} X_{\mathcal{N}_{\text{int}}} = \frac{1}{m_{\mathcal{N}_{\text{int}}}} \sum_{i=1}^{8} X_{\text{child}[i]} m_{\text{child}[i]} \\[2mm] Y_{\mathcal{N}_{\text{int}}} = \frac{1}{m_{\mathcal{N}_{\text{int}}}} \sum_{i=1}^{8} Y_{\text{child}[i]} m_{\text{child}[i]} \\[2mm] Z_{\mathcal{N}_{\text{int}}} = \frac{1}{m_{\mathcal{N}_{\text{int}}}} \sum_{i=1}^{8} Z_{\text{child}[i]} m_{\text{child}[i]} \\[2mm] m_{\mathcal{N}_{\text{int}}} = \sum_{i=1}^{8} m_{\text{child}[i]} \end{cases} \quad (2)$$

where $(X_{\mathcal{N}_{\text{int}}}, Y_{\mathcal{N}_{\text{int}}}, Z_{\mathcal{N}_{\text{int}}})$ are the coordinates of the position vector $\boldsymbol{R}_{\mathcal{N}_{\text{int}}}$ along the three principal axes of the system, while $m_{\text{child}[i]}$ and $(X_{\text{child}[i]}, Y_{\text{child}[i]}, Z_{\text{child}[i]})$ are, respectively, the mass and the coordinates of the position vector along the principal axes of the i-th child node of $\mathcal{N}_{\text{int}}$.

2.1.4 Force computation

After the octree is built and all nodes contain the required information, parameter θ is computed for each body-node pair. $(\mathcal{B}, \mathcal{N})$:

$$\theta_{\mathcal{B},\mathcal{N}} = \frac{\text{Radius of } \mathcal{N}}{R_{\mathcal{N}} - R_{\mathcal{B}}}. \quad (3)$$

This is compared with the value θ chosen by the user to set the accuracy and performance of the force estimation algorithm. A more detailed discussion on the effects of the user-tunable parameter θ on both computational time and accuracy of the algorithm is provided in Section 4.1. For each body $\mathcal{B}$, the tree is traversed from its root downwards along multiple branches and checks are performed at each encounter with a node $\mathcal{N}$, in order to compute force contributions:

(i) If $\mathcal{N}$ is a leaf node, the force contribution is evaluated as a classical body-to-body interaction:

$$F_{\mathcal{N} \to \mathcal{B}} = G m_{\mathcal{B}} m_{\mathcal{N}} \left(\frac{X_{\mathcal{N}} - X_{\mathcal{B}}}{r^3_{\mathcal{N}-\mathcal{B}}}, \frac{Y_{\mathcal{N}} - Y_{\mathcal{B}}}{r^3_{\mathcal{N}-\mathcal{B}}}, \frac{Z_{\mathcal{N}} - Z_{\mathcal{B}}}{r^3_{\mathcal{N}-\mathcal{B}}} \right). \quad (4)$$

(ii) If $\mathcal{N}$ is an internal node and $\theta_{\mathcal{B},\mathcal{N}} < \theta$, the traversal is interrupted at this node and the force contribution is evaluated:

$$F_{\mathcal{N} \to \mathcal{B}} = G m_{\mathcal{B}} m_{\mathcal{N}} \left(\frac{X_{\mathcal{N}} - X_{\mathcal{B}}}{r^3}, \frac{Y_{\mathcal{N}} - Y_{\mathcal{B}}}{r^3}, \frac{Z_{\mathcal{N}} - Z_{\mathcal{B}}}{r^3} \right), \quad (5)$$

where r is the distance from the particle to the centre of mass of the cube softened according to a parameter ϵ:

$$r = \sqrt{(X_{\mathcal{N}} - X_{\mathcal{B}})^2 + (Y_{\mathcal{N}} - Y_{\mathcal{B}})^2 + (Z_{\mathcal{N}} - Z_{\mathcal{B}})^2 + \epsilon^2}. \quad (6)$$

ϵ is called softening length and, as its name suggests, ensures the 'softening' of gravitational forces when $\boldsymbol{R}_{\mathcal{B}} \approx \boldsymbol{R}_{\mathcal{N}}$, to avoid very high values of speed and acceleration that the integrator could not handle. Studies show that, without opting for a time-dependent adaptive softening parameter, ϵ must be a factor of at least twice the minimum distance between a body and the attractive cluster involved (Rodionov & Sotnikova 2005). In the case of finite-size body interaction, with contact and collisions, the softening parameter is not needed, since no overlap between the centre of mass of a body and a node could occur.

(iii) If $\mathcal{N}$ is an internal node and $\theta_{\mathcal{B},\mathcal{N}} > \theta$, the traversal continues along all eight child nodes of $\mathcal{N}$.

2.1.5 CUDA implementation of GPU-parallel BH

The implementation of the BH algorithm on a GPU using CUDA language is inspired by the work of Burtscher and Pingali (Burtscher & Pingali 2011). The physical domain is divided into subdomains and the bodies are grouped following an octree structure. Similar to what was done by Burtscher & Pingali (2011), the numerical tasks to follow the BH algorithm have been divided among five kernels, which are written to minimize memory accesses. The tasks are executed sequentially on the GPU as follows:

(i) compute bounding box around all bodies (root of the octree);
(ii) build the octree;
(iii) compute mass properties of each node;
(iv) sort bodies by distance (to speed up force evaluation);
(v) compute forces.

Each body is assigned to a thread of the GPU, which traverses the octree to compute all force contributions acting on the body. Thanks to sorting, neighbour bodies require the traversal of a similar subpart of the octree, whereas distant bodies may require completely different traversals.

2.2 Rigid-body dynamics and contact interactions

Modules (iii) and (iv) deal with the creation of bodies and the definition of their inertial and surface properties. The latter depend on the contact method the user choose to use, among those available in (ii). The following paragraphs recall briefly the main features of the code concerning body handling and contact interactions, focusing on the parameters relevant to each method. References are provided for a more complete discussion of each method.

2.2.1 Creation of bodies and their properties

The shape of each body can be selected by the user among common geometries (sphere, box, cone, ...) or directly input by the user as a triangulated mesh. The shape can also be generated as the convex envelope of a randomly created cloud of points. When dealing with a large number of bodies, methods exist to select their spatial distribution or pattern (regular grid, randomly uniform, Poisson-disc sampling, ...) and bounding domain (common geometry or fitted in a triangulated mesh). Each body possesses 6 degrees of freedom, including both translational and rotational 3D motion. Random distribution routines can be used to set physical properties (size, density, ...) and dynamical state (position, velocity, angular position, angular velocity) of bodies as well. Finally, the surface properties of each body can be defined. These are directly correlated to the contact method in use. Surface/contact parameters are discussed below both for smooth and for non-smooth methods. More details can be found in the documentation of Tasora (2019).

2.2.2 Collision detection

Collision detection is performed into two steps: a *broad* phase and a *narrow* phase. During the broad phase, pairs of bodies whose geometrical boundaries are close enough are identified. The bounding volume of a body is estimated based on its velocity and integration time-step. If the bounding volume of two bodies overlaps, the narrow phase is performed and contact points are precisely found using a GJK algorithm (Tasora & Anitescu 2010; Ferrari et al. 2017).

2.2.3 Contact method: smooth contacts

The Smooth Contact Method (SMC) implemented in C::E (Fleischmann et al. 2015) is a force-based soft-body DEM method. The equations of motion are formulated as a system of Differential Algebraic Equations (DAEs), namely a system of Ordinary Differential Equations (ODEs) to reproduce the dynamics and Algebraic Equations (AEs) for the kinematic constraints. Forces are exchanged between bodies at contact using a two-way normal-tangent spring-dashpot system. The constitutive model of the spring-dashpot system can be selected between the simple Hooke and the more realistic Hertzian model. Relevant parameters of the SMC are coefficients of friction (static, dynamic, spinning), cohesion (as an attractive force at contact points), stiffness, and damping. As any soft-body DEM, it is suitable for smooth problems with no discontinuities but requires very small time-steps for cases with high stiffness to avoid numerical instability.

2.2.4 Contact method: non-smooth contacts

The Non-Smooth Contact (NSC) method implemented in C::E (Anitescu & Tasora 2010; Tasora & Anitescu 2010; Tasora & Anitescu 2011) is an impulse-based hard-body method. Unlike the SMC, the equations of motion are formulated as Differential Variational Inequalities (DVIs) and require the solution of a Cone Complementarity Problem (CCP) at each time-step. The NSC shares with the SMC parameters defining friction (static, dynamic, and spinning) and adhesion. The handling of contact interaction is, however, much simpler and solely relies on the restitution coefficient, defined as the ratio between velocity after and before the collision. Due to its impulse-momentum formulation, the NSC is best suited for problems with discontinuities or with nearly rigid contacts (high stiffness).
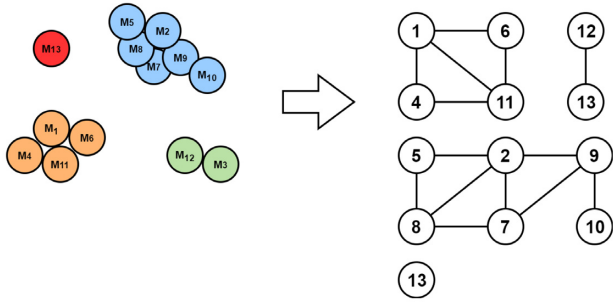
2.2.5 Contact method: non-smooth contacts with compliance

In the context of non-smooth dynamics, a method is available to simulate contacts with compliance and damping (Tasora et al. 2013). As for the NSC method, the equations of motion are formulated as DVI and are based on an impulse-momentum formulation. However, compliance and damping are enforced at the constraint level and the method is suitable not only for non-smooth problems but also to simulate the elastic and smooth behaviour typical of soft-body DEM models. Also, since it does not rely on the solution of a DAE, but its solution is found after solving a CCP-based DVI, this method does not require the very small time-step required by soft-body DEM. Its parameters are analogous to that of SMC: friction (static, dynamic, and spinning), cohesion, stiffness, and damping.

2.3 Numerical solvers

As discussed in the previous section, when using the SMC approach the dynamics are written as a system of DAE, and when using the NSC approach the dynamics are written as a DVI problem. From the numerical point of view, DAEs require to numerically integrate ODEs, while DVIs require the solution of an optimization problem at each time-step. Despite the radical difference in approaching the solution, both approaches require a time stepper and a non-linear solver. A large variety of time steppers and solvers is available in C::E (Tasora 2017): the most interesting for gravitational–granular dynamics applications include symplectic methods (semi-implicit Euler, leapfrog) and Runge–Kutta methods (RK45, explicit Euler,

Figure 2. Example of global contact graph construction.

implicit Euler, trapezoidal, Heun), to be complemented with either iterative or direct solvers. In particular, symplectic methods are very suitable for gravitational dynamics problem, and semi-implicit Euler method provides the best performance among them, when paired with an iterative solver (Anitescu & Tasora 2010; Mangoni, Tasora & Garziera 2018). This is suitable and effective to solve both DAE and DVI problems.

2.4 Post-processing: aggregate identification

Module (viii) contains routines for the post-simulation analysis of results. We provide here one example that applies to the case of rubble-pile aggregation simulations. One of the tasks to be performed after such simulations is to identify the aggregate(s) of bodies and determine their properties such as shape, mass, inertia, and void fraction (porosity).

2.4.1 Aggregate identification

The single or multiple aggregates obtained are identified through a method based on graph theory (Wilson 1996). First, the graph $\mathscr{G}$ is initialized to represent the whole system. Bodies are vertices of $\mathscr{G}$ and the size of the vertex set $E(\mathscr{G})$ is N (number of bodies). Edges of $\mathscr{G}$ represents physical contacts between objects: Two bodies M_i and M_j, represented by vertices P_i and P_j of $\mathscr{G}$, respectively, are in contact if and only if there is an edge $\{P_i, P_j\}$ linking P_i to P_j. At the end of the simulation, a contact container is filled with pairs ID_i, ID_j corresponding to pairs of bodies in contact with each other. The graph $\mathscr{G}$ is filled with all existing contacts, drawn as edges in the graph. A schematic example is shown in Fig. 2.

Aggregates obtained correspond to the connected subgraphs of $\mathscr{G}$. Single bodies that do not belong to any aggregate (e.g. because they have escaped from the re-aggregation processes) are represented by isolated vertices. The number of aggregates is defined as the number of connected components of $\mathscr{G}$ with a size greater than one, that is the number of the subset made of the connected subgraphs of $\mathscr{G}$ with more than one vertex. A Depth First Search algorithm (Wilson 1996) is used to provide an effective way of traversing simple graphs. The output is a classification of the edges and a spanning tree that can be used for finding the connected components of a graph. Using such an algorithm results in a complexity of $O(N + E)$ (where N and E are, respectively, the number of vertices and edges of $\mathscr{G}$) for the differentiation process.

2.4.2 Shape of the aggregate

The problem of defining the shape of a body made up of several discrete elements is not trivial and has a non-unique solution: Infinite surfaces exist to envelope a given set of points. After identifying the aggregate and its children bodies, we address the problem of defining its overall shape by means of an alpha-shape algorithm (Edelsbrunner & Mücke 1994). This computes the surface by enveloping a given set of points: in our case, all vertices of children bodies. Unlike the simpler convex hull envelope, the alpha-shape surface is, in general, non-convex. The working principle of the alpha-shape algorithm can be visually represented as follows. We start from a cloud of points (vertices of children bodies). Let us imagine to roll a sphere of radius R_α over the cloud of points. The alpha-shape envelope is the surface created by the sphere as it rolls over the cloud of points. In the limiting case where $R_\alpha = \infty$, the surface enveloping the cloud would be a convex hull. For a finite value of R_α, non-convex envelopes can be obtained. Typically, R_α should be of the same order of magnitude of the mean distance between vertices. As mentioned, the problem has a non-unique solution and the choice of R_α makes the process arbitrary. This is affecting in a relevant manner the computation of global properties of the aggregate such as volume, void fraction (porosity), and shape-related quantities (elongation, axial ratios, ...). To solve for this arbitrariness, we define an admissible range of enveloping surfaces, within a range of limiting cases. In general, two limiting surfaces can be defined: (*i*) the minimum volume surface, which is found using the minimum value of R_α such to have no surfaces delimiting cavities inside the aggregate, and (*ii*) the maximum volume surface, which is the convex hull of the aggregate, obtained with $R_\alpha = \infty$. Global properties are then computed considering the uncertainty on their values due to the arbitrary definition of the surface, and range between values computed using minimum and maximum volume surfaces.
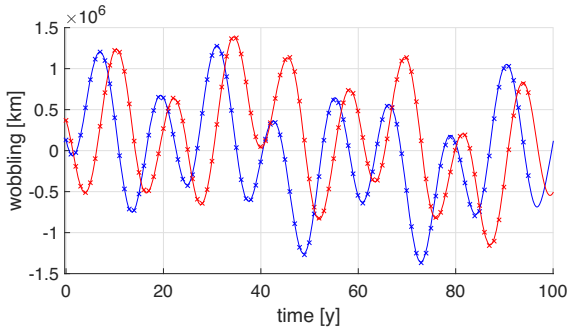
3 VALIDATION SCENARIOS

Original modules of C::E dealing with the modelling of granular dynamics have been widely tested and validated against benchmark and experimental tests [see the 'Validation Studies' section in Tasora (2019), Heyn (2013), and others]. The results of further test simulations are shown here to validate the gravity modules and their integration into the overall implementation. In particular, the accuracy of direct N^2 and octree gravity models is assessed and their performance is compared in terms of computational time. Conservation of energy, and linear and angular momentum, is checked for the purely gravitational case and for the coupled gravity–collision problem.

3.1 Direct *N*-body integration

The first set of tests concerns the accuracy of force computation for the direct N^2 method. Unlike the octree method, direct integration considers all gravitational interactions with no physical approximation. We performed several simulations of direct integration between mutually interacting bodies and in particular:

(i) conic solutions of the two-body problem are reproduced as a function of orbital energy;
(ii) orbits of planets in the Solar system are propagated for 2000 yr;
(iii) Halo and Lyapunov orbits around collinear libration points of the Sun–Earth and Earth–Moon circular restricted three-body problems (CR3BP) are reproduced.

When compared to the expected analytical (two-body problem) or numerical solution (*N*-body problem and CR3BP), the code is able to reproduce *exactly* the result, where the term *exactly* (here and

Figure 3. Validation scenario: in-plane (*x* and *y* components in the ecliptic J2000 plane) wobbling of the Sun around the barycentre of the Solar system during a 100-yr time span. A comparison between JPL's Horizons ephemerides[2] (solid line) and numerical integration (cross markers) is shown.

from now on) implies to within round-off accuracy or numerical integration truncation errors. As mentioned in Section 2.3, the semi-implicit Euler method is a very effective choice to solve gravitational dynamics, thanks to its symplectic nature that guarantees conservation of energy within the system. Energy and linear and angular momentum are conserved *exactly* for all cases, consistently with the nature of the direct N^2 formulation that introduces no approximation errors in the model. The successful outcome of the above tests guarantees that gravitational forces are computed correctly for each body. Fig. 3 shows an example related to one of the aforementioned scenarios. The expected in-plane wobbling of the Sun around the barycentre of the Solar system (data from JPL's Horizons ephemerides, solid line) is compared, and it matches well with the outcome of a direct N^2 integration of the motion of the Sun and planets during a 100-yr time span (cross markers).
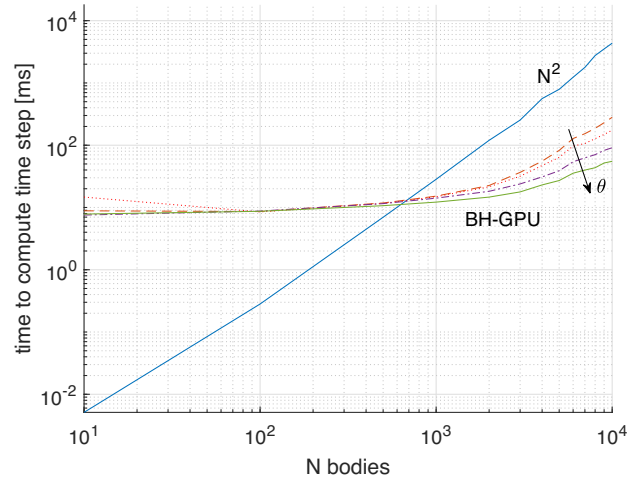
3.2 GPU-parallel octree

We evaluate here the performance of the GPU-parallel implementation and compare it to direct N^2 integration in terms of computational time and accuracy. Results presented here were obtained using a medium-range laptop with

(i) Intel(R) Core(TM) i7-6500U CPU @3.1Ghz;
(ii) Nvidia Geforce 940M (384 CUDA cores).

Although certainly dependent on hardware architecture, the results shown here identify trends and behaviours that are not dependent on hardware, but only on the algorithms used. To assess the performance of algorithms, two different sets of simulations are performed. For each simulation, bodies are created at time zero in random positions within a cubic domain and with a uniform distribution. Collision detection and interaction are disabled during the entire duration of the simulation, such that bodies only experience mutual gravity. The results of validation scenarios are in agreement with previous assessments concerning tree codes (Barnes & Hut 1986; Hernquist 1987).
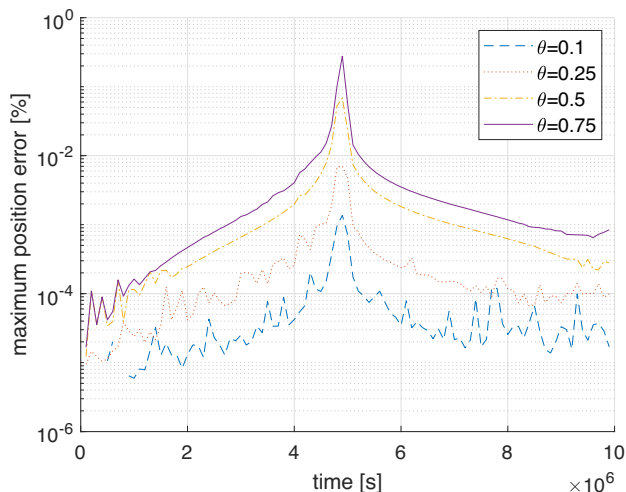
3.2.1 Computational time

A first set of simulations is arranged to compare the computational time required by each algorithm. Simulations are performed for

Figure 4. Time to compute one integration time-step as a function of the number of bodies in the simulation. Results are shown for direct N^2 integration versus BH-GPU-parallel algorithm ($\theta = 0.1, 0.25, 0.5,$ and 0.75).

several scenarios, including N ranging from 10 up to 10^4, and using both direct N^2 integration and BH-GPU algorithm with different levels of accuracy ($\theta = 0.1, 0.25, 0.5,$ and 0.75). We compare the time required by the code to compute one integration time-step. In principle, results can be compared after computing only one time-step. However, to avoid any bias and for the sake of a more accurate profiling, we integrate the dynamics forward for several time-steps and take the average time required to compute one of them. Also, we want the dynamics to be steady such to not interfere with the evaluation of performance. To this goal, we use a very short time-step (10 s) and a total duration of the simulation of 10^4 s. Results are shown in Fig. 4 using a logarithmic scale. As expected, the cost of direct integration increases with N^2. As discussed in Section 2.1, the BH-GPU algorithm executes a sequence of kernels and the largest amount of computational time is shared between the creation of the octree and the evaluation of the gravitational body–body or body–cluster interactions. Both octree and gravity evaluation depend on the number of bodies in the system, but to very different extents. Fig. 4 shows clearly that for a low number of bodies, the computational time required is dominated by the octree creation phase, which is independent from the choice of the accuracy parameter θ and increases very slowly with N. Conversely, for a high number of bodies, the time expense is dominated by the evaluation of the gravitational interactions, which instead depends on θ. When comparing direct N^2 with BH-GPU, it clearly appears that for a low number of bodies (approximately $N < 700$, the same for which θ is not affecting the time expense of BH-GPU), the process of building the octree makes the BH-GPU algorithm much more expensive than direct N^2 integration. Conversely, for a high number of bodies, the BH-GPU is much faster than direct N^2. For $N = 1000$, the BH-GPU turns to be nearly two times faster, and for $N = 10^4$, it is more than one order of magnitude faster. For a further increased number of bodies, direct N^2 simulations become practically unfeasible and BH-GPU becomes the only feasible option. As discussed, BH-GPU algorithm can be tuned in accuracy: This affects the computational time as shown for different θ values used in Fig. 4. In this high-N regime, the computational time of BH-GPU ranges from $O(N^2)$ (with $\theta = 0$, i.e. no clustering of bodies, equivalent to direct integration) up to $O(N \log (N))$ as θ increases.
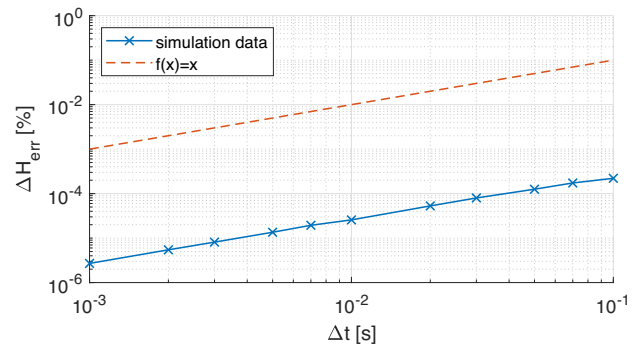
Figure 5. Maximum relative position error in time. The error is computed for simulations of 1000 bodies with BH-GPU-parallel algorithm ($\theta = 0.1, 0.25, 0.5,$ and 0.75), and it is relative to a simulation with direct N^2 integration.

3.2.2 Accuracy

A second set of simulations is used to assess the accuracy of the BH-GPU algorithm. In this case, we wait for the bodies to complete a full orbit around the barycentre of the system and come back to their initial state at rest. Since collisions are disabled, all bodies are undisturbed in their gravity-driven path and the system exhibits a nearly perfect energy-conserved pulsating contracting–dilating behaviour. This simulation time frame gives us a commensurate estimate of typical time duration of aggregation phenomena and provides us with every possible gravity-related condition, i.e. both, when bodies are far away and when they are close to each other. Compared to the first set of simulations, we use a larger time-step and we integrate forward the dynamics for a much longer time. Results are shown here for the case of $N = 1000$ and for $\theta = 0.1$, 0.25, 0.5, and 0.75. A simulation using direct N^2 integration is also performed and used as a real-world result to assess the accuracy of the BH-GPU algorithm. At each time-step, the position of each body in the system is compared to its corresponding real-world one and the relative error is evaluated as percentage of the latter

$$e_i(t) = \frac{\|\boldsymbol{r}_{i_{\mathrm{BH}}}(t) - \boldsymbol{r}_{i_{N^2}}(t)\|}{\|\boldsymbol{r}_{i_{N^2}}(t)\|} 100, \tag{7}$$

where $\boldsymbol{r}_{i_{\mathrm{BH}}}(t)$ and $\boldsymbol{r}_{i_{N^2}}(t)$ are the positions of body i at time t when using, respectively, BH-GPU and N^2 algorithms. Fig. 5 shows the maximum position error among bodies in the system at each time-step of the simulation. As expected, the error increases monotonically with θ, which is a direct mean and user-tunable parameter to select the desired level of accuracy. Few more features are worthy for a discussion. The error is shown to be time dependent and exhibits a peak at half-simulation time, i.e. at the end of contraction phase. This is expected, as the relative position error depends on the magnitude of accelerations acting on the bodies. The higher the acceleration, the higher the relative error when evaluating body–body versus body–cluster interactions. Accordingly, the error is shown to be higher in the middle part of the simulation, where bodies are closer to each other, and it increases during contraction and decreases as they get further to each other. It is interesting to look at the worst case depicted in Fig. 5. At the end of contraction phase,

Figure 6. Error on total angular momentum of the system after one simulation time-step, with both gravity and contact interactions. Error is shown in percentage, normalized to the initial value of angular momentum. The linear trend of the error as a function of time-step duration is highlighted.

the error between the two extremes $\theta = 0.1$ and 0.75 is of about two orders of magnitude. However, the maximum relative error obtained using $\theta = 0.75$ is still quite small, of the order of 0.3 per cent. Also, it is worth noting that this simulation provides us with an extremely conservative estimate of the accuracy. This is because real-world scenarios include the dissipative effects introduced by collisions and often entail only a part of the contraction phase simulated here. For example, if collisions were enabled in the scenario considered here, they would become very relevant starting at approximately $2.5 \cdot 10^6$ s (*transient phase*) and the bodies would never get as close to each other as they are in the $[3.5, 6.5] \cdot 10^6$ s range due to their physical extent (*aggregate phase*). They will therefore experience lower gravitational acceleration actions and position errors due to BH-GPU clustering: surely lower than the worst-case 0.3 per cent value found here.

3.3 Collisions

A set of simulations is performed to test conservation laws during collisions and contact interactions between bodies. In this case, we simulated the settling dynamics of an already formed self-gravitating aggregate with ~1000 convex-hull bodies. The dynamics are propagated forward for one time-step and contacts are solved using both smooth and non-smooth methods. The difference of physical and dynamical quantities is monitored. Several simulations are performed using the SMC method and different time-steps, ranging between 10^{-3} and 10^{-1} s. The same is done using the NSC method that is able to handle higher time-steps, between 10^{-1} to 10 s. A more detailed discussion on time-step selection criteria is reported in Section 4.1. Unlike the purely gravitational case, energy and angular and linear momentum are not conserved *exactly* and collisions introduce approximation errors in the models. Since the semi-implicit Euler method is a first-order integrator, the error is expected to depend linearly on the time-step. This is confirmed by our simulations, as shown in Fig. 6, which refer to the SMC case. It shows the angular momentum error in percentage, normalized to its initial value, for different time-steps. As expected, a clear linear trend is identified for the error, as it scales with the time-step.

4 THE ASTEROID AGGREGATION PROBLEM

This section discusses how rubble-pile scenarios are simulated using the code described above (Section 2). We discuss the appropriate

choice of methods to use, how to tune their parameters, and how to set up the numerical simulation.

4.1 Addressing the problem

To properly address the problem of simulating rubble-pile dynamics, we study the fundamental interactions that occur between bodies during gravitational aggregation phenomena. These can be associated with phases of the gravitational aggregation process:

(i) *Gravitational phase*: The dynamics of the bodies is only driven by mutual gravity and they do not experience contact interactions.

(ii) *Transient phase*: This phase begins as bodies start to collide with each other and form small aggregates of few bodies each. This phase is characterized by an extremely chaotic motion of the bodies that are simultaneously subjected to both gravity and random collisions. From a macroscopic point of view, the overall dynamics of the system is dominated by mutual gravity between bodies and newly forming aggregates. However, at particle level, the dynamics is dominated by collisions, which creates impulsive (non-smooth contact) or quasi-impulsive (smooth) forces on a much faster scale than gravitational one. The combined effects at system and body level, and the dissipative nature of collisions, lead eventually the system to a steady-state condition.

(iii) *Aggregate phase*: After transient phase, the system reaches a steady-state equilibrium where bodies are either clustered in one or more aggregates, or dispersed.

The most challenging phase to be simulated is certainly the transient phase, where both gravitational and collision dynamics play a major role. Since gravity and collisions act at different scales, we explore the capability and suitability of the code by studying such dynamics separately. This leads to meaningful and general conclusions on how to numerically simulate such interactions, also applicable to the fully coupled problem.

4.1.1 Gravitational dynamics

In this early stage of the aggregation process, bodies are not in contact with each other and only interact gravitationally. As discussed in Section 3, the choice of the most suitable method depends on the number of bodies involved: direct *N*-body for few bodies, and octree for a large number of bodies with a hardware-dependent threshold typically between 1000 and 10 000 bodies. In order to correctly simulate the *N*-body dynamics, the integration time-step Δt_g must be consistent to the characteristic time T_g of the problem and must sample the dynamics at least twice per T_g. In particular, the following must be satisfied (Ferrari et al. 2017):

$$\Delta t_g < \frac{T_g}{2} = \frac{1}{2\sqrt{G\rho}}, \tag{8}$$

where G is the universal gravitational constant and ρ is the material density of the bodies. For typical values of asteroid material density ranging between 1 and 4 g cm^{-3} (Richardson et al. 2002b), maximum time-steps are of the order of thousands of seconds.

4.1.2 Collisional dynamics

As done for gravitational dynamics, we study here the fundamental features related to collisional interactions between bodies. We analyse the different phases and numerical tasks involved when

solving for collisions and derive numerical constraints. From the numerical and algorithmic point of view, collisional dynamics requires the solution of two different problems: collision detection and collision output determination.

The collision detection algorithm operates at each time-step to identify contacts between body pairs. To ensure proper collision detection, the time-step must be adequate: If the time-step is too high, a collision could be missed. To quantify the dynamical constraint on the time-step, we consider here the case of a collision between two spheres of radius *R*. In particular, we consider the limiting case when a collision is not detected. It is easy to assess that, for the case of a direct collision occurring along the line connecting the centres of the spheres, the distance travelled during the time-step Δt_d must be lower than 4*R*, and then

$$\Delta t_d < \frac{4R}{v}, \tag{9}$$

where v is the relative velocity between bodies. Equation (9) can be generalized for the case of grazing collisions (Sánchez & Scheeres 2011):

$$\Delta t_d < \frac{2\sqrt{4R\delta - \delta^2}}{v}, \tag{10}$$

where δ is the maximum overlap allowed between bodies: the smaller the overlap allowed, the smaller the time-step required and the more precise the detection algorithm. Note that the case of direct collision in (9) can be retrieved from (10) when $\delta = 2R$ (complete overlapping). When considering a system of *N*-bodies, v is the maximum relative velocity in the system and 2*R* is the sum of the radii of the two smallest particles. Also, although (10) refers to a simplified case with spheres, the same relation can be conservatively used with convex hulls of any shape, where *R* represents their minimum characteristic size.

The second task is to solve for the actual collisional dynamics when bodies are in contact, in order to compute the collision output. This task is performed by the contact method selected, as discussed in Section 2.2. Hard- and soft-body methods deal with a very different modelling of the physics at contact. Hard-body methods model instantaneous collisions and deal with non-smooth dynamics based on impulse-momentum formulation. Since the collision is instantaneous, it does not make sense to speak of characteristic time involved in the dynamical process and then to derive a requirement on the time-step from that. However, the time-step still plays a major role and has to be selected properly to ensure the stability of the numerical solver. Hence, unlike gravity, collision detection, and, as discussed below, soft-body methods, the choice of the time-steps for hard-body methods is not driven by dynamical constraints, but by the properties of the numerical solver. We consider here the case of the semi-implicit Euler method. Looking at its stability region (Niiranen 1999), we can derive a constraint on the time-step to be used, as a function of the roots of the characteristic equation. However, our problem involves many bodies and requires the solution of a CCP at each time-step: The roots of the characteristic equation of such a problem cannot be easily determined. Alternatively, an empirical but very effective method is to determine a proper time-step by comparison against known benchmark problems or known behaviour of the system. For example, when benchmark problems are not available, a very convenient way is to tune the time-step according to a desired level of accuracy in terms of the conservation of total angular momentum of the system H_{tot}. As discussed in Section 3, the error depends linearly on the time-step. This linear dependence provides us with a

very effective means to estimate and select properly the time-step. Further examples on how the time-step can be set are provided in C::E technical documentation and validation studies (Tasora 2019), with reference to non-smooth (NSC) and non-smooth visco-elastic methods.

On the other hand, unlike hard-body ones, soft-body methods directly rely on the physical modelling of collisional dynamics. The collision occurs in a finite time and foresees the visco-elastic behaviour of the bodies at contact. Collisions are modelled by means of a spring-dashpot system at the contact point. We can therefore identify a characteristic time related to the fundamental frequency of the spring-dashpot system ω_k (Tsuji, Kawaguchi & Tanaka 1993):

$$T_k = \frac{2\pi}{\omega_k} = 2\pi\sqrt{\frac{m_r}{k}}. \tag{11}$$

When considering a system of N-bodies, ω_k is the highest frequency in the system, i.e. made of the combination of highest stiffness k and lowest reduced mass m_r.[3] Accordingly, the time-step must be at least two times smaller than the characteristic time

$$\Delta t_k < \frac{T_k}{2} = \frac{\pi}{\omega_k}. \tag{12}$$

Practical applications typically make use of smaller time-steps, ranging from $0.1T_k$ to T_k/π (Sánchez & Scheeres 2011), or even smaller than $0.01T_k$ (Herrmann & Luding 1998).

To summarize, two different constraints on the simulation time-step can be derived after the analysis of the collisional process. The first one is expressed by equation (10). It concerns the capability of the code to properly detect collisions and does not depend on the contact method in use. The second one depends on the contact method in use. In case of impulsive contacts, it depends on the stability of the numerical solver and can be determined empirically. In case of smooth dynamics, it depends on the dynamics of contact as expressed in equation (12). Both requirements shall be satisfied to properly simulate collisional processes. This duality is easily resolved by enforcing the most stringent requirement, i.e. by choosing the lowest time-step required. As a general rule, collision detection will be the bottleneck for high-velocity impacts and vice versa, and an accurate collision output will be the driving criteria in case of low-velocity collisions. In the attempt to quantify in a more rigorous way this general criterion, we investigate possible links between equations (10) and (12). Indeed, while the two effects appear not to be directly coupled for hard-body methods, some considerations can be made for the case of soft-body methods, where both (10) and (12) are built upon the dynamics of the problem. In particular, it is reasonable to assume that the maximum overlap allowed during collision detection δ must be consistently smaller than the maximum Δx contraction of the spring-dashpot system. If this is not satisfied, the visco-elastic behaviour at contact will not be accurately resolved. The maximum contraction Δx of a spring-mass system can be computed based on energy conservation consideration, by equating the total energy right before contraction (only kinetic energy, no contraction) to the total energy at maximum contraction (only potential energy, zero velocity):

$$\frac{1}{2}m_r v^2 = \frac{1}{2}k\Delta x_s^2 \tag{13}$$

from which is easy to find that

$$\Delta x_s = \frac{v}{\omega_k}. \tag{14}$$

In the case of a spring-dashpot system, the total energy is no longer conserved and a fraction of the initial kinetic energy is dissipated. The maximum contraction will be lower in this case, reduced by a factor of $\xi < 1$

$$\Delta x_{sd} = \xi\frac{v}{\omega_k}. \tag{15}$$

Now we define the maximum overlap allowed during collision detection δ as a small fraction of Δx_{sd}:

$$\delta = \eta\xi\frac{v}{\omega_k} \tag{16}$$

with $\eta < <1$, or equivalently

$$\delta = \varepsilon\frac{v}{\omega_k} \tag{17}$$

with $\varepsilon = \eta\xi < <1$. We can now substitute (17) into (10) and get

$$\Delta t_d < \frac{2\sqrt{4R\varepsilon\frac{v}{\omega_k} - \varepsilon^2\frac{v^2}{\omega_k^2}}}{v}. \tag{18}$$

This relation correlates the two time-steps involved in the numerical simulation of collision processes. In particular, we compare (18) with (12). The collision detection time-step Δt_d and the stiffness time-step Δt_k are equal if

$$\frac{2\sqrt{4R\varepsilon\frac{v}{\omega_k} - \varepsilon^2\frac{v^2}{\omega_k^2}}}{v} = \frac{\pi}{\omega_k}, \tag{19}$$

which can be rewritten as

$$4R\varepsilon\frac{v}{\omega_k} - \varepsilon^2\frac{v^2}{\omega_k^2} = \frac{\pi^2}{4}\frac{v^2}{\omega_k^2} \tag{20}$$

and finally, after trivial algebraic manipulation, as

$$v = \frac{4R\varepsilon}{\frac{\pi^2}{4} + \varepsilon^2}\omega_k. \tag{21}$$

This provides us with the quantification of the general criterion stated above. In particular, when selecting the time-step, collision detection will be the driving criterion when $\Delta t_d < \Delta t_k$, i.e. when

$$v > \frac{4R\varepsilon}{\frac{\pi^2}{4} + \varepsilon^2}\omega_k \tag{22}$$

and vice versa. Equation (22) gives us a useful condition to evaluate the tightest constraint through a simple comparison between maximum velocity and highest frequency in the system. Also, it correlates them with the accuracy of collision detection through ε. As expected, the problem of missing grazing collisions is an issue for high velocities and vice versa. More in detail, the break even point of the criterion is driven by the behaviour $v \sim \varepsilon\omega_k$, where ε is very small and ω_k is typically very high. Note that the dependence on R shown in (22) might be misleading, since R appears at the denominator of ω_k as well. In particular, the mass is proportional to R^3 and then the right-hand side of equation (22) is proportional to $\sim 1/\sqrt{R}$.

The last point to discuss is about the choice of the contact method (SMC versus NSC versus NSC with compliance) to be used. As mentioned in Section 2.2, SMCs are naturally suited for smooth problems (low-velocity contacts), NSC for non-smooth problems (high-velocity impacts) while in principle, the NSC method with compliance can be used for both. A more detailed comparison between the aforementioned methods, including simulation of different scenarios, evaluation of performance, and criteria for selection of parameters, is reported in C::E technical documentation

---

[3]m_r is the reduced mass of the two least massive bodies in the system.

(Tasora 2019). Studies to assess the accuracy of such methods are also reported. Unlike gravity, where the exact dynamics of the system are known and can be computed, contact methods do not model in a comprehensive fashion the physics involved in collisional processes. While the results of gravity algorithms can be validated using an analytical exact formulation, the only way to assess the accuracy of contact methods is through experimental test. As mentioned in Section 3, the documentation of C::E (Tasora 2019) reports the results of several benchmark scenarios, including cone penetration test, standard triaxial test, and direct shear test. Further analytical and experimental validation scenarios are reported in Heyn (2013).

4.1.3 The gravity–collision problem

The full gravitational aggregation problem features both gravitational and collisional dynamics and, in principle, both of them constrain the choice of the simulation time-step. However, gravity and collisions act on very different time-scales and the characteristic time of gravity is typically several orders of magnitude higher than that of collisions. Also, their dynamics are not coupled. For these reasons, it is not required to perform the simulation using only one time-step [which would be the smallest one, i.e. the one related to collisions $\Delta t_c = \min(\Delta t_k, \Delta t_d)$]. Instead, as done in Sánchez & Scheeres (2012), the aggregation scenario can be simulated using two time-steps: the gravity time-step Δt_g, determined using (8), and the collision time-step Δt_c, determined using either (10) or (12). In particular, we advance the simulation of Δt_c to properly solve collisional dynamics, but we evaluate gravity (through N^2 interactions or by building the octree) every Δt_g. This allows for consistent savings in terms of computational time, without jeopardizing the accuracy of results.

In this context, a more accurate modelling of the local gravity interactions between fragments would not be beneficial towards a more realistic representation of the granular *N*-body problem. Past astrodynamics studies have clearly shown that the mutual gravitational dynamics between objects of irregular shape, represented using shape-based (e.g. polyhedron) models, are very different from the case of point-mass source interaction (e.g. Fahnestock & Scheeres 2006; Ferrari & Lavagna 2018). However, the granular *N*-body problem is very different from that of astrodynamics applications, which consider purely gravitational motion with no contact and collision interactions. In our case, due to the presence of both gravity and contact interactions, acting at different time-scales, a noticeable effect of deviations due to enhanced gravity modelling would appear after a time when many contact/collision interactions have already occurred. In this case, the chaotic nature of the contact/collision interactions represents a much higher source of uncertainty compared to the accuracy of local gravity models. Because of this, all authors identify the better modelling of contact/collision interactions as the way to enhance the realism of simulations (e.g. Michel et al. 2004; Richardson et al. 2009; Movshovitz et al. 2012, and others) and agree to using octrees, at the cost of a lower accuracy of local gravity computations. Unlike astrodynamics applications, the goal here is not to accurately reproduce the coupled and orbital-attitude dynamics of each fragments, but rather to reproduce the global behaviour of the system. Relevant parameters concerning gravity are indeed global ones, such as bulk density and total mass of the system, and lower fidelity local models of gravity (as in octrees) are widely used.

4.2 Features of the problem

As discussed, the study of the dynamics of self-gravitating aggregates is a very complex problem and deals with challenges arising from the coupled and chaotic interactions occurring within the granular media. The study of asteroid shapes and rubble-pile dynamics has its fundamentals in the continuum theory, extended by Holsapple (2001, 2004, 2007, 2010) to the study of rubble-pile objects. However, self-gravitating granular systems are not strengthless fluid bodies and, for example, can spin faster than a perfect fluid before shedding mass (Richardson, Elankumaran & Sanderson 2005). Even for cohesionless aggregates, many effects can contribute to providing strength to the aggregate. These include surface interaction phenomena, such as friction, and other effects related to the geometry of contact interactions between particles such as interlocking (Sánchez & Scheeres 2012). To summarize all these effects, the strength of a rubble pile can be quantified based on its effective angle of friction, which results from particle shapes, size distributions, packing, and surface friction (Sánchez & Scheeres 2012). Holsapple (2010) showed that the angle of friction is very important to establish the domain of admissible equilibrium shapes, and that spins at limiting conditions become larger for a higher angle of friction, implying a larger strength of the aggregate. Accordingly, Sńchez & Scheeres (2016) observe that the angle of friction inhibits deformation. In Zhang et al. (2017), geometric interlocking is the main source of shear strength for crystal structures, and higher values can be achieved for random packing and increased size heterogeneity of fragments. Among geometrical effects, extreme relevance is given by the angularity of non-spherical particles. As mentioned, spheres behave very differently from angular particles, for which the contact dynamic problem is very different. Due to their non-smooth shape, angular particles can foster interlocking and between particles. In particular, spheres have only one contact point per pair, while non-spherical objects can have several. This has a great impact on all aspects related to contact dynamics and to the exchange of interactions/forces at contact points. For example, this affects how the values of parameters such as surface friction and restitution coefficients act on a global scale, since they both act at contact points. This means that their effect can be greatly amplified depending on relative geometry. In practice, friction and restitution act on a larger scale and contact interactions result more dissipative with respect to the case of spheres. This is confirmed by experiments showing that restitution appears to be much lower for irregular objects (Hartmann 1978) due to the loss of centre-of-mass kinetic energy to rotational energy post-collision (Korycansky & Asphaug 2006). In practice, angular shapes can increase the strength (Richardson, Elankumaran & Sanderson 2005; Jiang, Shen & Wang 2015; Zhang et al. 2018) of the aggregate and, through a lower restitution coefficient, increase the likelihood of the formation of a stable aggregate during a gravitational re-accumulation process (Walsh, Richardson & Michel 2008). The resolution of the model, in terms of the number of bodies in the rubble-pile aggregate, is also very relevant. Richardson et al. (2005) show that coarse configurations consisting of a small number of larger particles are more resistant to tidal disruption or reshaping than fine configurations with many smaller particles, observing that the strength of the aggregate depends on the number of particles as well. A major role in the granular dynamics processes is played by friction. Sánchez & Scheeres (2012) found that friction increases substantially the limiting spin rate. For the case of angular bodies, where the overall effect of friction is higher (many contact points), the expected limiting spin rate would be even higher. Sánchez &

Scheeres (2012) also observe that the addition of surface friction makes the aggregate more stable. They conclude that friction might modify the square root dependence between critical spin of a granular aggregate and bulk density. The same is found by Zhang et al. (2017), who conclude that higher interparticle friction can keep a spinning rubble pile stable at a higher spin rate. Furthermore, Pravec & Harris (2007) show how maximum spin rate increases (substantially) with friction. To summarize, the chaotic dependence between the dynamical history, initial shape/configuration of the particles, contact/surface parameters, and properties of the final aggregate (Zhang et al. 2017) gives a clear picture of the complexity of the problem and the unpredictability of its dynamics.

4.3 Simulation examples

A set of simulations is performed here to extend results presented by Ferrari et al. (2017) to an increased number of bodies and to verify the correlations between parameters of the simulation, initial conditions of the N-bodies, and aggregation outcome. To simulate realistic aggregation scenarios, it is important to carefully select the physical properties of the N-bodies and their initial dynamics. Initial conditions play a crucial role in the formation of the aggregates and their properties. The initial dynamical state includes the position and velocity v_0 of the centre of mass of each body, as well as the angular position and spin rate ω_0 of each body. Typical asteroid aggregation scenarios consider these bodies as fragments created from a parent larger body after, for example, a collision event (Michel et al. 2001; Campo Bagatin et al. 2018). In such a case, the fragments could share a common residual angular momentum due to spinning motion of the parent body. Parameter Ω is introduced to simulate such effect, and it is defined as the initial residual angular velocity of the system of particles: When $\Omega \neq 0$, the fragments have a non-zero orbital velocity around the centre of mass of the system. In order to compare and extend the results in Ferrari et al. (2017), similar scenarios are set up, using non-smooth dynamics and a time-step of 10 s. In our simulations, bodies are convex hulls whose shape is created as the geometrical envelope of randomly generated points. In particular, we use 16 points, and corresponding convex hulls have, on average, 10 vertices. Surface friction is based on a simple Coulomb model with a coefficient of 0.6. To foster the formation of aggregates, restitution is set to zero (perfectly inelastic collisions). Results presented here refer to simulations with 10^4 bodies, with an average characteristic size of each fragment of 3 km and maximum initial distance between bodies of the order of 100 km. The physical properties of fragments are chosen among typical values of objects belonging to the main asteroid belt or near-Earth asteroids population. The material density is set to 3 g cm^{-3}, a common choice for asteroid aggregation simulations (Sánchez & Scheeres 2011), suitable to produce values of bulk density between 1.2 and 2.5 g cm^{-3}, typical of the asteroid population (Richardson et al. 2002b). At initial time, the angular and centre of mass position of bodies are randomly generated with a uniform spatial distribution, as well as the directions of their linear and angular velocity vectors.

An example of aggregation sequence is shown in Fig. 7, for a simplified case with 10^3 bodies (to facilitate the visualization of images). While not in contact, the motion of the N-bodies is driven solely by their mutual attraction. When fragments start to interact at closer range, few small bodies are scattered away due to collision mechanisms. Depending on initial conditions, one, multiple, or no stable aggregates are eventually formed. Aggregation is not observed when relative velocities between bodies are too high and in particular when parameters v_0, ω_0, and Ω are above certain threshold
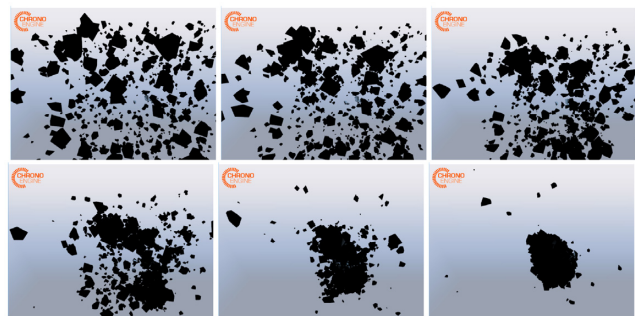

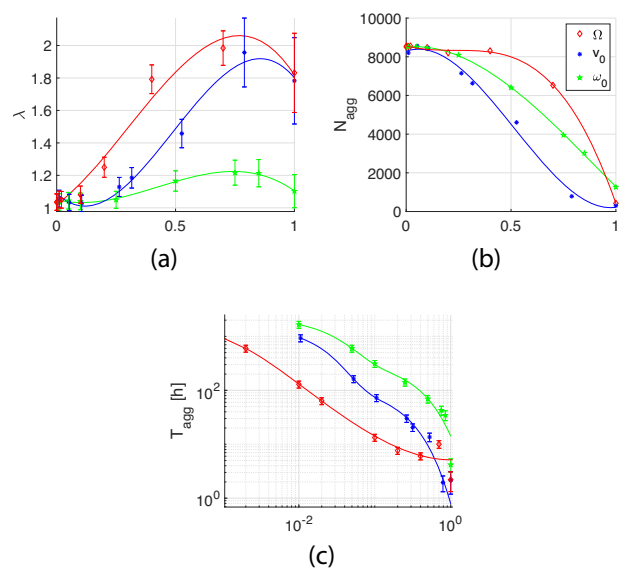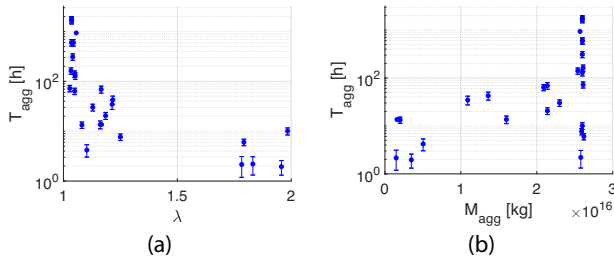**Figure 7.** Aggregation sequence example with 1000 bodies.

Figure 8. Properties of the final aggregate as functions of normalized initial conditions: (a) inertial elongation, (b) the number of fragments in the main aggregate, and (c) rotation period of the main aggregate. Uncertainties due to the definition of the final aggregate's surface are highlighted.

values, above which no aggregation is possible. As for the case under study, these values are identified approximately with $v_0 = 95$ m s^{-1}, $\omega_0 = 10^{-1}$ rad s^{-1}, and $\Omega = 5 \cdot 10^{-4}$ rad s^{-1}. After reaching a steady state, aggregates are considered as single asteroids. Different kinds of aggregates have resulted from the simulation performed.

Fig. 8 shows the properties of the largest aggregate formed after the numerical simulation, as a function of the initial dynamics of the system. In particular, it shows (a) the inertial elongation λ, defined as the ratio between maximum and minimum moment of inertia of the aggregate (Ferrari et al. 2017), (b) the number of fragments in the aggregate N_{agg}, and (c) its rotation period T_{agg}, as a function of initial conditions Ω (red diamonds), v_0 (blue asterisks), and ω_0 (green stars). The values of Ω, v_0, and ω_0 are shown on the abscissa normalized to their maximum values, so that each parameter ranges from 0 to 1. Polynomial (for λ and N_{agg}) and exponential (for T_{agg}) fitting functions are used to highlight the trend of data distribution.

The measure of the global properties and geometry of the final aggregate is affected by the uncertainty on the computation of its enveloping surface. As discussed in Section 2.4, quantities such as volume, porosity, bulk density, and size of the aggregate are better defined within a range of values computed between a minimum volume surface and a maximum volume surface. As a

Figure 9. Rotation period of the final aggregate as functions of (a) inertial elongation and (b) mass of the aggregate. Uncertainties due to the definition of the final aggregate's surface are highlighted.

consequence, the inertial elongation λ and the angular velocity (or equivalently the rotation period T_{agg}) of the aggregate are affected by the uncertainties of this measurement and should also be defined within a range of values. In particular, the angular velocity of the aggregate ω_{agg} is measured after computing its total angular momentum H_{agg} and inertia matrix I_{agg}, as $\omega_{agg} = I_{agg}^{-1} H_{agg}$, as in Richardson et al. (2005). The computation of the inertia tensor of the aggregate (and its principal inertia axes) relies on the definition of the enveloping surface. The uncertainty of this measurement depends on the size of the single particle with respect to the total aggregate. Hence, the accuracy depends on the number of particles in the aggregate and it is higher for smaller aggregates (those that spin faster in our simulations). Richardson et al. (2005) estimate that the error in axis measurement could be as large as on particle radius along each axis, with errors of about 10 per cent for the case of 1000 bodies. When propagated to the computation of angular velocity, they estimate errors larger than 20 per cent. Similarly, we estimated the error in the computed values of λ and T_{agg}, based on the number of bodies in the final aggregate, and show their uncertainties around the computed mean value in Figs 8 and 9. In our case, the smaller aggregates are the fastest spinning ones: They are made of tens or few hundreds of bodies, meaning potentially very large measurement errors. In other words, the resolution of our smaller aggregates is too coarse and does not allow us to estimate accurately their spin rate, elongation, porosity, and bulk density. Also, due to their low number of bodies, it is rather inaccurate to use the term rubble pile when referring to such small aggregates. To a closer inspection, the structure of these aggregates is closer to a monolith rather than a rubble pile: They are made of one or few big boulders and tens of smaller particles resting on their surface. This structure is by no means a typical self-gravitating aggregate and it is rather closer to the 'test particle on a rigid sphere' case reported in Weidenschilling (1981) and Richardson et al. (2000), where critical spin may be estimated using the simple balance between gravitational and centrifugal forces (Hestroffer et al. 2019). This is indeed the result we find for them. The spin rate for aggregates with many more particles is comparable to what has been found by other authors within the error prescribed by the uncertainties mentioned above. Among these, our fastest spinning aggregates have spin rates of the order of 4.2 ± 0.9 h.

General trends of Figs 8 and 9 agree with expected behaviour and are coherent with what is discussed in Section 4.2. As expected, higher elongations are obtained for higher values of v_0, ω_0, and Ω. Maximum elongation is reached for values between 75 and 85 per cent of their limiting value. For higher values of initial conditions (>75–85 per cent), the simulations result in the formation of multiple aggregates of smaller size, which are more typically regularly shaped than larger ones. It is indeed observed that the

maxima in Fig. 8(a) represent thresholds after which the formation of more than one significant aggregate occurs. As confirmed in Fig. 8(b), the higher the velocities, the lower the number of bodies found in the main aggregate, which is smaller for higher values of v_0, ω_0, and Ω. Fig. 8(c) confirms a well-known result, with larger aggregates having a higher rotation period (lower spin rate) with respect to smaller ones. Fig. 9 shows the rotation period of the final aggregate as a function of (a) inertial elongation and (b) mass of the aggregate. As expected, fast rotators are more elongated and less massive, while slow rotators have a more regular rounded shape and are more massive.

5 CONCLUSION

The paper presents the numerical implementation of a code for coupled gravity–granular dynamics problems, and reports numerical methods available to solve gravitational and collisional dynamics. We build upon the work by Ferrari et al. (2017) and extend the capabilities of the code by introducing a parallel CUDA-GPU octree structure, to be able to evaluate mutual gravity for a higher number of bodies. Unlike existing *N*-body codes, the implementation presented here has the capability to handle non-spherically shaped bodies to a full double precision accuracy. Compared to classical sphere-based codes, this allows for a more realistic simulation of contacts between bodies and opens new opportunities and scenarios to be simulated. After presenting the results of test scenarios to validate newly implemented modules, we discuss the features and performance of numerical methods and derive requirements arising from the dynamics of asteroid aggregation scenarios. The problem of numerical simulation of gravitational aggregation is addressed: We provide guidelines and quantify criteria to properly set up such numerical simulations. We show examples of possible applications by performing numerical simulations of gravitational aggregation. The parameter space explored includes initial conditions of bodies in terms of relative velocity between them, spinning rate, and residual angular velocity of the system. The outcome of the analysis is discussed, showing the properties of the final self-gravitating aggregate after its stable formation in terms of inertial elongation, period of rotation, and the number of bodies in the aggregate.

REFERENCES

Aarseth S. J., 1999, Publ. Astron. Soc. Pac., 111, 1333
Aarseth S., 2001, New Astron., 6, 277
Alder B. J., Wainwright T. E., 1959, J. Chem. Phys., 31, 459
Anitescu M., Tasora A., 2010, Comput. Optim. Appl., 47, 207
Barnes J., Hut P., 1986, Nature, 324, 446
Burtscher M., Pingali K., 2011, in Hwu W. W., eds, GPU Computing Gems. Elsevier Inc, Boston, MA, p. 75
Campo Bagatin A., Alemañ R. A., Benavidez P. G., Richardson D. C., 2018, Icarus, 302, 343

Chambers J., 1999, MNRAS, 304, 793

Cundall P. A., Strack O. D. L., 1979, Géotechnique, 29, 47

Dorband E. N., Hemsendorf M., Merritt D., 2003, J. Comput. Phys., 185, 484

Dubois F., Acary V., Jean M., 2018, C. R. Méc., 346, 247

Duncan M., Levison H., Lee M., 1998, AJ, 116, 2067

Edelsbrunner H., Mücke E., 1994, Trans. Graph., 13, 43

Fahnestock E. G., Scheeres D. J., 2006, Celest. Mech. Dyn. Astron., 96, 317

Ferrari F., Lavagna M., 2018, Adv. Space Res., 62, 2245

Ferrari F., Tasora A., Masarati P., Lavagna M., 2017, Multibody Syst. Dyn., 39, 3

Fleischmann J., Serban R., Negrut D., Jayakumar P., 2015, J. Comput. Nonlinear Dyn., 11, 044502

Gebhardt N., 2016, IrrLicht Release 1.8.4. Available at: http://irrlicht.sourceforge.net

Gilardi G., Sharf I., 2002, Mech. Mach. Theory, 37, 1213

Hartmann W. K., 1978, Icarus, 33, 50

Hernquist L., 1987, ApJS, 64, 715

Herrmann H., Luding S., 1998, Contin. Mech. Therm., 10, 189

Hestroffer D. et al., 2019, A&AR, 27, 6

Heyn T., 2013, On the Modeling, Simulation, and Visualization of Many-body Dynamics Problems with Friction and Contact. University of Wisconsin-Madison, Madison, WI

Hockney H., Eastwood J., 1988, Computer Simulation Using Particles. CRC Press, Bristol, PA

Holsapple K., 2001, Icarus, 154, 432

Holsapple K. A., 2004, Icarus, 172, 272

Holsapple K. A., 2007, Icarus, 187, 500

Holsapple K. A., 2010, Icarus, 205, 430

Hut P., Makino J., 2010, Moving Stars Around. Maya Open Lab Project, The Art of Computational Science

Jean M., Moreau J. J., 1987, in Del Piero G., Maceri F., eds, Unilateral Problems in Structural Analysis — 2. Springer Vienna, Vienna, p. 151

Jiang M., Shen Z., Wang J., 2015, Comput. Geotech., 65, 147

Korycansky D., Asphaug E., 2006, Icarus, 181, 605

Mangoni D., Tasora A., Garziera R., 2018, Comput. Methods Appl. Mech. Eng., 330, 351

Michel P., Benz W., Tanga P., Richardson D., 2001, Science, 294, 1696

Michel P., Benz W., Richardson D., 2004, Planet. Space. Sci., 52, 1109

Morbidelli A., 2002, Annu. Rev. Earth Planet. Sci., 30, 89

Movshovitz N., Asphaug E., Korycansky D., 2012, ApJ, 759, 93

Niiranen J., 1999, Proc. Vol. 1, 6th. Int. Conf. Electrimacs' 99, Fast and Accurate Symmetric Euler Algorithm for Electromechanical Simulations (Note: The method became later known as 'Symplectic Euler'). Lisboa, Portugal, p. 71

Porco C. C., Weiss J. W., Richardson D. C., Dones L., Quinn T., Throop H., 2008, The AJ, 136, 2172

Pravec P., Harris A., 2007, Icarus, 190, 250

Pruett C., Rudmin J., Lacy J., 2003, J. Comput. Phys., 187, 298

Richardson D. C., Quinn T., Stadel J., Lake G., 2000, Icarus, 143, 45

Richardson D., Quinn T., Stadel J., Lake G., 2002a, Planet. Space. Sci., 30, 45

Richardson D., Leinhardt Z., Melosh H., Bottke W. Jr, Asphaug E., 2002b, in Bottke W. F. Jr, Cellino A., Paolicchi P., Binzel R. P., eds, Asteroids III, Vol. 160. Univ. Arizona Press, Tucson, AZ, p. 501

Richardson D. C., Elankumaran P., Sanderson R. E., 2005, Icarus, 173, 349

Richardson D. C., Michel P., Walsh K. J., Flynn K. W., 2009, Planet. Space. Sci., 57, 183

Rodionov S. A., Sotnikova N. Y., 2005, Astron. Rep., 49, 470

Sánchez P., Scheeres D. J., 2011, ApJ, 727, 120

Sánchez D. P., Scheeres D. J., 2012, Icarus, 218, 876

Schwartz S. R., Richardson D. C., Michel P., 2012, Granular Matter, 14, 363

Sńchez P., Scheeres D. J., 2016, Icarus, 271, 453

Stadel J., 2001, PhD thesis, Univ. Washington, Seattle

Tancredi G., Maciel A., Heredia L., Richeri P., Nesmachnow S., 2012, MNRAS, 420, 3368

Tasora A., 2017, Time Integration in Chrono::Engine. Available at: http://www.projectchrono.org/assets/white_papers/ChronoCore/integrator.pdf

Tasora A., 2019, Project Chrono. Available at: http://projectchrono.org/

Tasora A., Anitescu M., 2010, J. Comput. Nonlinear Dyn., 5, 031004

Tasora A., Anitescu M., 2011, Comput. Methods Appl. Mech. Eng., 200, 439

Tasora A., Anitescu M., Negrini S., Negrut D., 2013, Int. J. Non-Linear Mech., 53, 2

Tasora A. et al., 2016, in Kozubek T., Blaheta R., Šístek J., Rozložník M., Čermák M., eds, High Performance Computing in Science and Engineering. Springer Int. Publ., Cham, p. 19

Tsuji Y., Kawaguchi T., Tanaka T., 1993, Powder Technol., 77, 79

Wada K., Senshu H., Matsui T., 2006, Icarus, 180, 528

Walsh K. J., Richardson D. C., Michel P., 2008, Nature, 454, 188

Wang L., Spurzem R., Aarseth S., Nitadori K., Berczik P., Kouwenhoven M. B. N., Naab T., 2015, MNRAS, 450, 4070

Weidenschilling S., 1981, Icarus, 46, 124

Wilson R., 1996, Introduction to Graph Theory. Longman Group, Harlow, GB

Wisdom J., Holman M., 1991, AJ, 102, 1528

Zhang Y. et al., 2017, Icarus, 294, 98

Zhang Y., Richardson D. C., Barnouin O. S., Michel P., Schwartz S. R., Ballouz R.-L., 2018, ApJ, 857, 15

This paper has been typeset from a TEX/LATEX file prepared by the author.