The Astrophysical Journal, 932:124 (24pp), 2022 June 20 https://doi.org/10.3847/1538-4357/ac6dd2
© 2022. The Author(s). Published by the American Astronomical Society.
OPEN ACCESS

GENGA. II. GPU Planetary N-body Simulations with Non-Newtonian Forces and High Number of Particles

Simon L. Grimm[1], Joachim G. Stadel[2], Ramon Brasser[3], Matthias M. M. Meier[4,5], and Christoph Mordasini[6]
[1] University of Bern, Center for Space and Habitability, Gesellschaftsstrasse 6, CH-3012, Bern, Switzerland; simon.l.grimm@unibe.ch
[2] University of Zürich, Institute for Computational Science, Winterthurerstrasse 190, CH-8057, Zürich, Switzerland
[3] Origins Research Institute, Centre for Astronomy and Earth Sciences, Konkoly Thege Miklos St 15-17, H-1121 Budapest, Hungary
[4] Naturmuseum St.Gallen, Rorschacher Strasse 263, 9016 St.Gallen, Switzerland; matthias.meier@naturmuseumsg.ch
[5] ETH Zurich, Institute of Geochemistry and Petrology, Clausiusstrasse 25, 8092 Zurich, Switzerland
[6] University of Bern, Physikalisches Institut, Gesellschaftsstrasse 6, CH-3012, Bern, Switzerland
Received 2022 January 24; revised 2022 May 1; accepted 2022 May 6; published 2022 June 23

Abstract

We present recent updates and improvements of the graphical processing unit (GPU) N-body code GENGA. Modern state-of-the-art simulations of planet formation require the use of a very high number of particles to accurately resolve planetary growth and to quantify the effect of dynamical friction. At present the practical upper limit is in the range of 30,000–60,000 fully interactive particles; possibly a little more on the latest GPU devices. While the original hybrid symplectic integration method has difficulties to scale up to these numbers, we have improved the integration method by (i) introducing higher level changeover functions and (ii) code improvements to better use the most recent GPU hardware efficiently for such large simulations. We added treatments of non-Newtonian forces such as general relativity, tidal interaction, rotational deformation, the Yarkovsky effect, and Poynting–Robertson drag, as well as a new model to treat virtual collisions of small bodies in the solar system. We added new tools to GENGA, such as semi-active test particles that feel more massive bodies but not each other, a more accurate collision handling and a real-time openGL visualization. We present example simulations, including a 1.5 billion year terrestrial planet formation simulation that initially started with 65,536 particles, a 3.5 billion year simulation without gas giants starting with 32,768 particles, the evolution of asteroid fragments in the solar system, and the planetesimal accretion of a growing Jupiter simulation. GENGA runs on modern NVIDIA and AMD GPUs.

Unified Astronomy Thesaurus concepts: Celestial mechanics (211); N-body simulations (1083); Planet formation (1241); Solar system formation (1530); GPU computing (1969); Computational methods (1965); Asteroid dynamics (2210); N-body problem (1082)

1. Introduction

GENGA is an N-body code optimized for simulating planet formation and planetary system evolution. It runs on graphical processing units (GPUs) to speed up the integration time. On average, a high-end GPU has a similar amount of computing power as a 128-core CPU, but with less than half of the power consumption (Portegies Zwart 2020). Since the release of the first version of GENGA (Grimm & Stadel 2014), available computational resources have improved dramatically in performance, but especially GPU performance. On the other hand, the requirements of state-of-the-art simulations in planet formation have also increased dramatically. Current hardware allows for 30,000–60,000 fully interactive planetesimals to be simulated for 10 Myr in about 60 days. Examples of these high-end simulations are given in Quarles & Kaib (2019), Clement et al. (2020) and Woo et al. (2021). Since the parallel programming environment in the CUDA language has evolved, it is also necessary to reconsider some of the parallelization techniques used in GENGA and to optimize or adapt them further to suit modern hardware.

In addition to a good performance, a modern N-body code for planet formation or long-term planetary system evolution must also include non-Newtonian forces. General relativity corrections, tidal forces and rotational deformation forces are necessary to simulate the long-term behavior of compact exoplanetary systems. In simulations including small bodies, the Yarkovsky effect and eventually Poynting–Robertson drag must be implemented to calculate volatile material between different parts of the planetary system more realistically.

Examples of N-body CPU codes including such additional forces are Mercury-T (Bolmont et al. 2015) or Posidonius (Blanco-Cuaresma & Bolmont 2017). Codes that used CPU parallel N-body methods are REBOUND (Rein & Liu 2012) and PKDGRAV3 (Potter et al. 2017), while GPUs are used in the codes QYMSYM (Moore & Quillen 2010), GLISSE (Zhang & Gladman 2022), or Hiperion.[7]

GENGA uses a hybrid symplectic integration method, which is able to maintain a good long-term energy conservation while it is also able to resolve close encounters accurately. This is possible by using a smooth changeover function that moves the gravitational forces between close-encounter pairs from the symplectic integrator to a direct N-body solver. By increasing the number of particles in the simulations, it has become evident that this integration method does not scale up well. Since the number of close encounters can grow very fast with the number of particles used, the efficiency of the parallelization can be affected. This fact makes evident the need to

[7] https://www.konkoly.hu/staff/regaly/research/hiperion.html

reconsider the hybrid symplectic integration method for large N simulations. In this work, we introduce higher level changeover functions and we demonstrate how this new method is able to perform simulations with up to ∼60,000 fully interactive planetesimals.

In addition to non-Newtonian forces and an improved integration method, we add a variety of new options and tools to GENGA, which we describe below. This paper is structured as follows:

In Section 2, we summarize the theory of the hybrid symplectic integrator and explain why the integration method needs to be improved in order to handle a large number of close encounters. We continue with a detailed description of newly added non-Newtonian forces. These are general relativity corrections in Section 3, tidal forces and rotational deformation forces in Section 4. Both of these sections contain a testing and comparison subsection. In Section 5, we introduce new forces and models for small-body simulations. These are the Yarkovsky effect in Section 5.1, Poynting–Robertson drag in Section 5.4, a collisional break-up model for meteoroid dynamics of the solar system in Section 5.6 and a model for collisional induced rotation-rate reset in Section 5.7. The latter two models are useful for example when the transport of small ejecta material in the solar system is studied. We present an example simulation of ejecta material from the asteroid 6 Hebe in Section 5.8.

After introducing new forces and models we continue with a detailed description of improvements of the integration method in Section 6. These include more efficient implementations of the Bulirsch–Stoer integration kernels needed for the close-encounter handling presented in Section 6.1, and the introduction of a higher level changeover function shown in Section 6.3. In the Sections 6.4 and 6.5, we present two long-term example simulations of terrestrial planet formation with and without gas giants and a gas disk. For both simulations, the results are shown for different numbers of initial particles, ranging from 2048 to 65,536.

In Section 7, we describe improvements in the collision and encounter handling. These include the option of a more precise collision resolution, the option for reporting and tracing back collisions at a time before they really collide, and the option to report all close-encounter events during a simulation. New tools in GENGA are described in Section 8. These include a new semi-active test particle mode in Section 8.1 and the option of using predefined coordinates, mass, or radii tracks in Section 8.2.

Finally, the performance of GENGA is analyzed in Section 9.

GENGA is open source and available at https://bitbucket.org/sigrimm/genga. The repository also contains a detailed user documentation and a tutorial where GENGA can be tested online.

2. Theory of The Hybrid Symplecic Integrator

A symplectic integration method (e.g., Wisdom & Holman 1991) is able to integrate planetary orbits over billions of years without a drift in energy and angular momentum. Since the method is restricted to use a fixed time step, it is difficult to handle close encounters between bodies where the time step of the involved bodies must be reduced substantially. It also presents a numerical challenge if the orbital periods in the system vary over several orders of magnitude, since the orbit of the innermost bodies determine the overall time step. The hybrid symplectic method developed by Chambers (1999) uses a smooth changeover function to transfer the calculation of close encounters from the symplectic to a direct N-body integrator. We also use a Bulirsch–Stoer (Stoer & Bulirsch 2002; Press et al. 2007) method to resolve the close-encounter part, but other adaptive time step N-body methods could be used here as well, such as a Hermite integrator (Makino & Aarseth 1992) or embedded Runge–Kutta schemes (Hairer et al. 1993), even if these may cause artificial precession in the orbits. The transition between the symplectic part and the direct N-body part must be applied smoothly enough to prevent the accumulation of large errors in the total energy. Therefore, a critical radius must be defined to set a threshold between the close-encounter regime and the normal integration regime. It must be chosen to be large enough to ensure a smooth transition, but small enough to avoid spending too much simulation time in the encounter phase. In the following section, we repeat the most important parts of the hybrid symplectic integrator. For more details, we refer to Chambers (1999) or Grimm & Stadel (2014).

By using democratic coordinates (heliocentric positions and barycentric velocities), the Hamiltonian system of a planetary system can be split into three parts $H_{\rm A}$, $H_{\rm B}$, and $H_{\rm C}$, which correspond to the Keplerian part, the interaction part, and the Sun part of the Hamiltonian system, respectively. These are given by

$$H = H_{\rm A} + H_{\rm B} + H_{\rm C}, \tag{1}$$

with

$$H_{\rm A} = \sum_{i=1}^{N}\left(\frac{p_i^2}{2m_i} - \frac{Gm_i m_\star}{r_{i\star}}\right) - \sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{Gm_i m_j}{r_{ij}}[1 - K(r_{ij})], \tag{2}$$

$$H_{\rm B} = -\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{Gm_i m_j}{r_{ij}}K(r_{ij}), \tag{3}$$

and

$$H_{\rm C} = \frac{1}{2m_\star}\left(\sum_{i=1}^{N}\boldsymbol{p}_i\right)^2, \tag{4}$$

where the symbol $\star$ refers to the central mass and $K(r_{ij})$ is a smooth changeover function ranging from 0 to 1 (see top panel of Figure 9). The distance between the particles i and j is r_{ij}.

The limit where the changeover functions is applied is defined by a critical radius $r_{\rm crit}$ of a particle i, which is computed as

$$r_{{\rm crit},i} = \max(n1 \cdot R_{{\rm H},i},\, n2 \cdot dt \cdot v_i). \tag{5}$$

The critical radius depends on two terms. The first depends on the Hill radius $R_{\rm H}$ while the second contains the time step dt and the Kepler velocity v of the particle i. The two parameters $n1$ and $n2$ are constants. Experimentation has found that typical values that yield a good compromise between accuracy and computation time are $n_1 = 3$ and $n_2 = 0.4$ (Chambers 1999).

The integrator needs to search for close-encounter pairs at each time step and to sort them into independent close-encounter groups. These groups are then integrated with the Bulirsch–Stoer direct N-body method (Grimm & Stadel 2014).

Ideally, the close-encounter groups consist of only a single pair of bodies, but it can happen that bodies have multiple close-encounter pairs, which need to be linked together in a bigger close-encounter group. In the worst case scenario, all bodies are in a close encounter with some neighboring bodies, and all of them are linked together into a single giant close-encounter group. This scenario is likely to happen when the particle number density is increased during high-*N* simulations. In that case, the original hybrid symplectic method becomes very inefficient and must be improved or replaced with something else. We describe this in Section 6.3.

2.1. High-N Simulations and Large Close-encounter Groups

The critical radius $r_{\mathrm{crit},i}$ depends directly on the velocity of the particles, which introduces a difficulty when running high-resolution simulations with a large number of small planetesimals. While the first term in the critical radius decreases as the Hill radius decreases, the second term does not depend on the particle radii or masses. This means that, by increasing the particle surface or volume density of the simulations, the critical radii of the planetesimals do not change much and more and more particles are considered to be in a mutual close-encounter regime. Ultimately this can lead to a situation whereby every particle is in a close encounter with its neighbor, and therefore all particles are linked together into one very large close-encounter group, as illustrated in Figure 7. This behavior is already described in Grimm & Stadel (2014, Section 5).

The original version of GENGA only permitted close-encounter group sizes of at most 128 particles because of memory limitations in the first Bulirsch–Stoer implementation of Grimm & Stadel (2014). In Section 6, we present a new implementation without an upper limit on the close-encounter group sizes. However, the performance of these very large close-encounter groups is less efficient than small groups and this situation should still be avoided if possible. In Section 6.3, we present a way to reduce the size of close-encounter groups even for high-resolution simulations.

2.2. Non-Newtonian Forces

Celestial mechanics contains a variety of different forces. Besides the dominant Newtonian gravitational force, thermal forces, tides, or general relativity effects can be important. While Grimm & Stadel (2014) already contains a model for a gas disk, we add in this work the most important non-Newtonian forces to GENGA. Other forces can be added individually. The user can enable or disable all these forces in the GENGA parameter file.

In general, these additional forces can depend on both the positions and the velocities of the bodies, which would violate the symplectic structure of the integrator because these forces would not be separable in the Hamiltonian system (Saha & Tremaine 1994). Applying such a force in a usual kick operation would lead to an error in the energy which should be avoided (Saha & Tremaine 1994). A simple solution to this problem is to calculate the force with an implicit midpoint method because it is also symplectic (Saha et al. 1997).

In Sections 3–5, we describe the newly added forces in detail. These are general relativity corrections, tidal forces, rotational deformation, the Yarkovsky effect, and Poynting–Robertson drag. These forces are expressed in heliocentric coordinates. That means that before the forces can be applied in the integration, barycentric velocities must be converted to heliocentric coordinates and back after the non-Newtonian force step, which causes computational overhead.

2.3. The Implicit Midpoint Method

A conservative implementation of a general force *f*, which is not separable in the Hamiltonian system, can be achieved by the implicit midpoint method, which is given by (Hairer et al. 1993)

$$\boldsymbol{v}_i^{t+dt} = \boldsymbol{v}_i^t + dt \cdot \boldsymbol{a}_f\left(\boldsymbol{r}_i^t, \frac{\boldsymbol{v}_i^t + \boldsymbol{v}_i^{t+dt}}{2}\right). \tag{6}$$

Here $\boldsymbol{a}_f$ is the acceleration caused by the force *f*. Equation (6) needs to be iterated until the difference between $\boldsymbol{v}_i^{t+dt}$ and $\boldsymbol{v}_i^t$ is smaller than a defined tolerance value. In practice, we find that fewer than three iterations are needed. We use the implicit midpoint method in GENGA to implement non-Newtonian forces that are not separable in the Hamiltonian.

3. General Relativity

General relativity (GR) causes a perihelion shift of all bodies, orbiting a central mass as well as an increase in the mean motion (Saha & Tremaine 1994). Due to energy conservation the semimajor axis is not affected and remains constant. While the effect of GR is usually small for solar system formation simulations, it can play an important role for objects close to the central star such as compact systems of exoplanets. This is especially true for long-term stability analysis of planetary systems.

GR can be approximated by post-Newtonian corrections of the form (Kidder 1995; Mardling & Lin 2002; Fabrycky 2010):

$$\boldsymbol{a}_{\mathrm{PN}} = -\frac{G(M_\star + m_i)}{r_i^2 c^2} \cdot \left\{-2(2-\eta)\dot{r}_i \boldsymbol{v}_i + \left[(1+3\eta)\boldsymbol{v}_i \cdot \boldsymbol{v}_i - \frac{3}{2}\eta \dot{r}_i^2 - 2(2+\eta)\frac{G(M_\star + m_i)}{r_i}\right]\hat{\boldsymbol{r}}\right\}, \tag{7}$$

where *c* is the speed of light and η is defined as:

$$\eta = \frac{M_\star m_i}{(M_\star + m_i)^2}. \tag{8}$$

A similar description is given in Sitarski (1983) or at a higher order in Moyer (2003, Equations (4)–(26)). Since Equation (7) depends on the positions and the velocities, the GR corrections must by applied with the implicit midpoint method described above in Section 2.3.

3.1. GR Hamiltonian Splitting

As an alternative to the previous formulation, Saha & Tremaine (1994) provide a description of general relativity corrections by using a Hamiltonian splitting approach. The GR (or also called post-Newtonian) Hamiltonian equation takes the form given by

$$H_{\mathrm{PN}} = \sum_i \left(\alpha_i H_{\mathrm{Kep},i}^2 + \frac{\beta_i}{r_i^2} + \gamma_i p_i^4\right), \tag{9}$$

with the quantities

$$
\begin{aligned}
\alpha_i &= \frac{3}{2m_i c^2},\\
\beta_i &= \frac{-\mu_i^2 m_i}{c^2},\\
\gamma_i &= -\frac{1}{2m_i^3 c^2},\\
\mu_i &= G(M_\star + m_i),
\end{aligned}
\tag{10}
$$

and the speed of light c.

Following Saha & Tremaine (1994) the term $\gamma_i p_i^4$ in the Hamiltonian equation can be expressed as

$$
d\boldsymbol{x}_i = -dt \cdot 2\frac{v_i^2}{c^2}\boldsymbol{v}_i, \tag{11}
$$

with the time step dt. This term can be combined with the $H_{\rm C}$ term in Equation (4). The β term in the Hamiltonian equation can be expressed as

$$
d\boldsymbol{v}_i = -dt \cdot 2\frac{\mu^2}{c^2 r_i^4}\boldsymbol{r}_i, \tag{12}
$$

and can be combined with the $H_{\rm B}$ term in Equation (3). However, care must be taken that the changeover function $K(r_{ij})$ does not affect this term.

Following Saha & Tremaine (1994) yet again, the α term can be implemented through a time step modification in the particle drift operation (Equation (2)):

$$
dt_i{}' = dt \cdot \left(1 - \frac{3}{2}\frac{\mu}{a_i c^2}\right). \tag{13}
$$

This time step modification must be applied in both the FG function of the Kepler solver (Grimm & Stadel 2014) as well as the Bulirsch–Stoer direct solver for close encounters. It is important to only modify the acceleration terms between the central mass and the particles, $\boldsymbol{a}_{i\star}$, and *not* between two particles, $\boldsymbol{a}_{ij}$.

An important detail is that the formulation above requires the use of pseudovelocities. Saha & Tremaine (1994) recommend to carry out the entire integration in pseudovelocities and only convert back to real velocities at integration output times. However, GENGA allows the use of other non-Newtonian forces that can also depend on the velocities. We cannot make use of this approach and instead we are required to compute all the GR terms and other forces in real velocities rather than in pseudovelocities. Therefore, at each time step, we convert from real velocities to pseudovelocities and back, causing a slight computational overhead. Only the Sun-kick and the drift operation is done in pseudovelocities. Since the conversion in Equation (13) can be done for each particle individually, the parallelization of it is trivial. However, this approach requires two additional CUDA kernel calls, which reduces the performance of small simulations because these are not dominated by the N^2 kick operations or by close encounters.

3.2. Comparison and Test of the GR Effect

A good object to test the general relativity effect is evolution of the asteroid (1566) Icarus (Shapiro et al. 1971; Sitarski 1992). It has an orbital eccentricity of 0.83 so that it ventures closer than 0.2 au to the Sun. In order to compute the evolution of the orbit of Icarus, we integrate it together with the 25 most massive objects of the solar system. All data are taken from the JPL HORIZON system.[8] We compare our integration with the measured orbit of Icarus. The results are shown in Figure 1. One can clearly see how the GR corrections improve the orbital positions during the first few orbital periods. Comparing the Hamiltonian splitting approach with the implicit midpoint method leads to no significant difference. The integration precision can be improved when the 25 perturber objects are not integrated, but rather interpolated from the known ephemeris. The reason for this is that in reality other forces and effects occur, which we do not include here and which can influence the positions of the perturbers slightly. The remaining difference in the orbital position after applying the GR corrections is most likely caused by other non-gravitational effects (Greenberg et al. 2017), and small deviations in the initial conditions increase after each revolution.

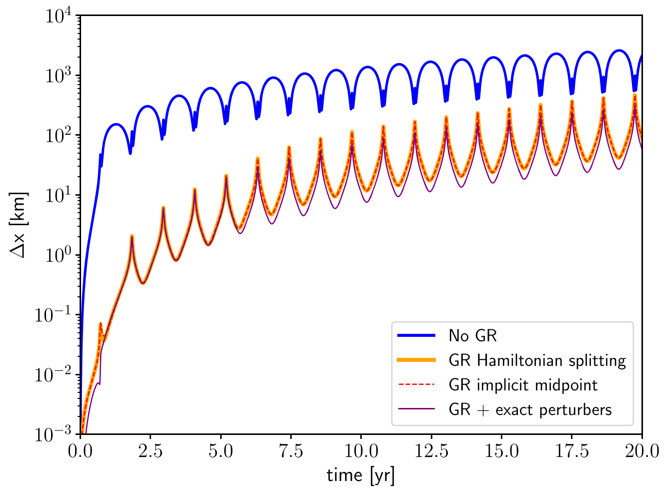

Figure 1. Difference in position of the asteroid (1566) Icarus between the integration and the measured orbit. The Asteroid Icarus is integrated together with the 25 most massive objects of the solar system. When GR corrections are not considered (blue line) then the position after two orbital periods differs by more than 100 km. With GR effects enabled, the difference is less than 2 km. There is not significant difference between the Hamiltonian splitting approach (orange line) and the implicit midpoint method (red dashed line). When the position of the 25 perturbers is not integrated, but interpolated from their measured positions, then the difference in position is reduced further (purple line).

The GR effect calculation scales linearly with the number of particles N, so that the performance of large simulations ($N \gtrsim 4096$) is only affected marginally. Simulations with $N \approx 1024$ are slowed down by $\sim$1%, while small simulations ($N \lesssim 128$) are affected more. A simulation with 16 fully interacting bodies can take up to 1.5 times longer with the implicit midpoint method, and more than twice as long with the Hamiltonian splitting method. The user can choose between the two implementations with a setting in the GENGA parameter file. We recommend using the implicit midpoint method, since it gives a better performance for similar accuracy.

4. Tidal Forces and Rotational Deformation

In this section, we describe the implementation of the tidal force and the rotational deformation force. Both of these forces

[8] https://ssd.jpl.nasa.gov/horizons.cgi

depend on the positions and the velocities, therefore they are calculated with the implicit midpoint method, described in Section 2.3. All equations are given in heliocentric coordinates.

4.1. Tidal Forces

For implementing the tidal effects, we follow the weak friction tidal model described in Hut (1981) and Bolmont et al. (2015). In this model, the tidal bulge of a primary body with radius R, induced by a companion body of mass m, is described by two point masses at the surface of the primary body with mass (Hut 1981)

$$\mu \approx \frac{1}{2} k_{2,i} m R^3 [r(t-\tau)]^{-3}, \tag{14}$$

where $r(t-\tau)$ is the distance between the two bodies at time $t-\tau$, and τ is a constant time lag, which models the tidal dissipation. The quantity $k_{2,i}$ is the potential Love number of degree 2 of the body i. Introducing the quantities

$$D_{\mathrm{to},i} = 3G\frac{m_\star^2 R_i^5}{r_i^7} k_{2,i} \tag{15}$$

$$D_{\mathrm{to},\star} = 3G\frac{m_i^2 R_\star^5}{r_i^7} k_{2,\star} \tag{16}$$

leads to Equation (5) in Bolmont et al. (2015), which is given by

$$P_{\mathrm{to},i} = D_{\mathrm{to},i}\tau_i \tag{17}$$

$$P_{\mathrm{to},\star} = D_{\mathrm{to},\star}\tau_\star. \tag{18}$$

The radial part of the tidal force can be written as

$$F_{\mathrm{Tr}} = -(D_{\mathrm{to},\star} + D_{\mathrm{to},i}) - 3\frac{\dot{r}_i}{r_i}(P_{\mathrm{to},\star} + P_{\mathrm{to},i}), \tag{19}$$

where r_i is the distance between body i and the central mass $\star$, and $\dot{r}_i = \frac{dr_i}{dt}$.

The total tidal force can be written as Equation (6) in Bolmont et al. (2015), where we corrected a typo in the first line (also corrected in Bolmont et al. 2020), which is given by

$$\begin{aligned} \boldsymbol{F}_T &= F_{\mathrm{Tr}}\boldsymbol{e}_{ri} \\ &+ P_{\mathrm{to},i}(\boldsymbol{\Omega}_i \times \boldsymbol{e}_{ri}) + P_{t0,\star}(\boldsymbol{\Omega}_\star \times \boldsymbol{e}_{ri}) \\ &- (P_{\mathrm{to},i} + P_{\mathrm{to},\star})(\dot{\boldsymbol{\theta}}_i \times \boldsymbol{e}_{ri}), \end{aligned} \tag{20}$$

where $\boldsymbol{e}_{ri}$ is the radial unit vector, and $\boldsymbol{\Omega}_\star$ is the rotational angular velocity vector of the star, $\boldsymbol{\Omega}_i$ the rotational angular velocity vector of body i, and $\dot{\boldsymbol{\theta}}_i$ describes the orbital velocity of the body i. A figure explaining the tidal force model is given in Hut (1981) and in Bolmont et al. (2015).

Rewriting the last term as

$$\begin{aligned} (\dot{\boldsymbol{\theta}}_i \times \boldsymbol{e}_{ri}) &= \frac{1}{r_i^3}(\boldsymbol{r}_i \times \boldsymbol{v}_i) \times \boldsymbol{r}_i \\ &= \frac{\boldsymbol{v}_i}{r_i} - \frac{(\boldsymbol{r}_i \cdot \boldsymbol{v}_i)\boldsymbol{r}_i}{r_i^3} \\ &= \frac{\boldsymbol{v}_i}{r_i} - \frac{\dot{r}_i}{r_i}\boldsymbol{e}_{ri}, \end{aligned} \tag{21}$$

leads to

$$\begin{aligned} \boldsymbol{F}_T = &\left[-(D_{\mathrm{to},\star} + D_{\mathrm{to},i}) - 2\frac{\dot{r}_i}{r_i}(P_{\mathrm{to},\star} + P_{\mathrm{to},i})\right]\boldsymbol{e}_{ri} \\ &+ P_{\mathrm{to},i}(\boldsymbol{\Omega}_i \times \boldsymbol{e}_{ri}) + P_{\mathrm{to},\star}(\boldsymbol{\Omega}_\star \times \boldsymbol{e}_{ri}) \\ &- (P_{\mathrm{to},i} + P_{\mathrm{to},\star})\frac{\boldsymbol{v}_i}{r_i}. \end{aligned} \tag{22}$$

Note that there are different models of the tidal forces in existence other than the constant time lag description that we use. For more details see, e.g., Efroimsky & Lainey (2007), Leconte et al. (2010), or Auclair-Desrotour et al. (2014). Some of these models can also be treated in a secular evolution model. In Section 4.3, we compare our implementation to three secular evolution models. Leconte et al. (2010) describe how the time lag τ can be converted into the tidal quality factor Q. While there is no simple relation for the general case, the case of a non-synchronized circular orbit with $Q \gg 1$ can be converted as (Leconte et al. 2010)

$$Q = \frac{1}{2\tau|\omega - n|}. \tag{23}$$

4.2. Tidal Torque

In addition to the acceleration on the particles the tidal force generates a torque, which changes the spin angular momentum of the particles and of the central star. This tidal torque is given as (Hut 1981)

$$\boldsymbol{N}_T = \boldsymbol{r} \times \boldsymbol{F}_{T\theta}, \tag{24}$$

where $\boldsymbol{F}_{T\theta}$ is the transverse component of the tidal force. With the relation

$$\boldsymbol{r} \times (\boldsymbol{\Omega} \times \boldsymbol{e}_r) - \boldsymbol{r} \times \frac{\boldsymbol{v}}{r} = \boldsymbol{\Omega} r - (\boldsymbol{r}\cdot\boldsymbol{\Omega})\boldsymbol{e}_r - \boldsymbol{e}_r \times \boldsymbol{v}, \tag{25}$$

the tidal torque can be written as

$$\boldsymbol{N}_{\mathrm{T}i} = P_{\mathrm{to},i}[\boldsymbol{\Omega}_i r_i - (\boldsymbol{r}_i \cdot \boldsymbol{\Omega}_i)\boldsymbol{e}_{ri} - \boldsymbol{e}_{ri} \times \boldsymbol{v}_i] \tag{26}$$

$$\boldsymbol{N}_{T\star} = P_{\mathrm{to},\star}[\boldsymbol{\Omega}_\star ri - (\boldsymbol{r}_i \cdot \boldsymbol{\Omega}_\star)\boldsymbol{e}_{ri} - \boldsymbol{e}_{ri} \times \boldsymbol{v}_i]. \tag{27}$$

In heliocentric coordinates the spin evolution takes the form (Bolmont et al. 2015)

$$\frac{d}{dt}(I_\star \boldsymbol{\Omega}_\star) = -\sum_{j=1}^{N} \frac{m_\star}{m_\star + m_j}\boldsymbol{N}_{T\star} \tag{28}$$

and

$$\frac{d}{dt}(I_i \boldsymbol{\Omega}_i) = -\frac{m_\star}{m_\star + m_i}\boldsymbol{N}_{\mathrm{T}i}, \tag{29}$$

where I is the moment of inertia.

We integrate the tidal spin evolution with the implicit midpoint method described in Section 2.3 so that the total angular momentum of the system is conserved and we accurately model the evolution of the semimajor axis. Since every particle is affecting the spin of the central star, an additional parallel reduction step is necessary to sum up all contributions synchronously.[9]

Note that our current implementation respects only the tidal effect from the central star to the other bodies, and vice versa. It

[9] With a parallel reduction method, arrays of length N can be summed up in $\log(N)$ steps. Depending on the size N, this can be performed within a thread block, or must be distributed across different thread blocks and kernel calls.

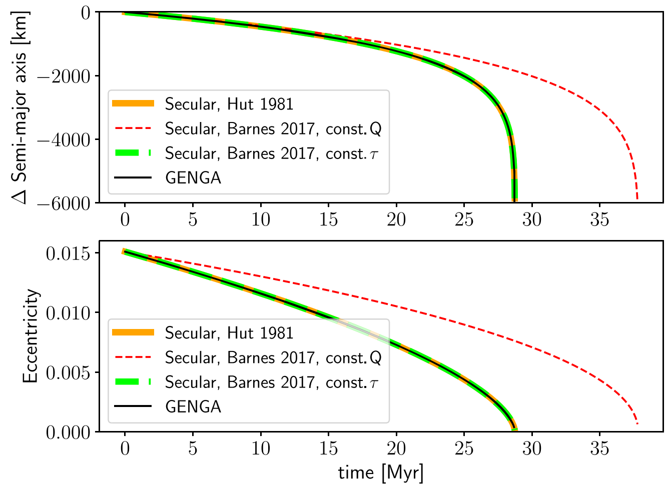

Figure 2. Comparison of GENGA against secular evolution models on the example of the Mars-Phobos system. The models correspond to Hut (1981) (orange solid line), Barnes (2017), const. Q model (red dashed line), and Barnes (2017), const. τ model (green dashed line). The evolution of the semimajor axis in GENGA agrees well with the constant τ scenario in the secular evolution models. The evolution of the eccentricity starts with the same slope, but then gets damped more rapidly in GENGA. This plot reproduces the results of Figure 2 from Efroimsky & Lainey (2007).

does not include tidal effects between other bodies, such as tidal effects between multiple planets. This feature is already implemented in other codes (e.g., Posidonius Blanco-Cuaresma & Bolmont 2017), and we plan to eventually include it in a future version of GENGA.

4.3. Test of the Tidal Forces

To test the implementation of the tidal forces, we repeat the calculations of the Mars-Phobos system described in Efroimsky & Lainey (2007) and Auclair-Desrotour et al. (2014). As initial values, we use the same parameters as given in Table 1 in Auclair-Desrotour et al. (2014). Additionally, we use an eccentricity of 0.0151 for Phobos. We compare our results against three secular evolution models. These are[10]

1. Hut (1981, Equations (9) and (10)) This uses a constant τ value.
2. Barnes (2017), constant Q value (also called constant-phase-lag-model, CPL). It is similar to Efroimsky & Lainey (2007, Equation (30)) (Kaula model).
3. Barnes (2017), Correia (2009), constant τ value (also called constant-time-lag-model, CTL). It is similar to Efroimsky & Lainey (2007, Equation (36)) (Singer-Mignard model).

The results are shown in Figure 2. The evolution of the semimajor axis and the eccentricity from GENGA agree well with the constant τ secular evolution models, as is expected because GENGA also uses a constant τ model. However, in GENGA the tidal model is not implemented as a secular evolution model. While we use a time step of 0.1 day in GENGA, the secular evolution models can use a time step of 100–1000 days.

[10] All three secular evolution models consider only the tidal bulge on the primary body, caused from the orbiting body, and not vice versa. A model which contains both directions is given in Correia (2009).

4.4. Rotational Deformation

When a viscous body is rotating, its shape is transformed into a symmetric oblate ellipsoid. The potential energy of a system containing two oblate bodies 0 and 1 is given in Moyer (1971, Equation (158)), Moyer (2003) or (Correia et al. 2011) as

$$U = -\frac{Gm_0m_1}{r}\left[1 - \sum_{i=0,1}\sum_{n=1}^{\infty} J_n\left(\frac{R_i}{r}\right)^n P_n(\hat{r}\cdot\hat{\Omega}_i)\right], \tag{30}$$

with the mass m, the physical radius R, the distance between the bodies r, the rotational angular velocity Ω and the Legendre polynomial of degree n, $P_n(x)$. In GENGA we truncate the order n to 2 (Correia et al. 2011) and we compute the J_2 parameter purely via rotational deformation as (Correia et al. 2011; Bolmont et al. 2015)

$$J_{2,i} = k_{2f,i}\frac{\Omega_i^2 R_i^3}{3Gm_i} \tag{31}$$

and

$$J_{2,\star} = k_{2f,\star}\frac{\Omega_\star^2 R_\star^3}{3Gm_\star}, \tag{32}$$

where $k_{2f,i}$ is the second potential Love number (fluid Love number) of body i.

We follow the description in Bolmont et al. (2015) and define the following quantities:[11]

$$C_\star = \frac{1}{2}Gm_i m_\star J_{2,\star}R_\star^2 \tag{33}$$

and

$$C_i = \frac{1}{2}Gm_i m_\star J_{2,i}R_i^2. \tag{34}$$

With these definitions the force due to the rotational deformation is given as (Bolmont et al. 2015)

$$\begin{aligned}\boldsymbol{F_R} = \Bigg\{&-\frac{3}{r_i^5}(C_\star + C_i)\\ &+\frac{15}{r_i^7}\left[C_\star\frac{(\boldsymbol{r}_i\cdot\boldsymbol{\Omega}_\star)^2}{\boldsymbol{\Omega}_\star^2} + C_i\frac{(\boldsymbol{r}_i\cdot\boldsymbol{\Omega}_i)^2}{\boldsymbol{\Omega}_i^2}\right]\Bigg\}\boldsymbol{r}_i\\ &-\frac{6}{r_i^5}\left(C_\star\frac{\boldsymbol{r}_i\cdot\boldsymbol{\Omega}_\star}{\boldsymbol{\Omega}_\star^2}\boldsymbol{\Omega}_\star + C_i\frac{\boldsymbol{r}_i\cdot\boldsymbol{\Omega}_i}{\boldsymbol{\Omega}_i^2}\boldsymbol{\Omega}_i\right).\end{aligned} \tag{35}$$

Similar to the tidal force, the rotational deformation generates a torque on the bodies. This rotational deformation torque is given as (Hut 1981)

$$\boldsymbol{N_R} = \boldsymbol{r}\times\boldsymbol{F}_{R\theta}, \tag{36}$$

where once again $\boldsymbol{F}_{R\theta}$ is the transverse component of the rotational deformation force. Following Bolmont et al. (2015) the torque can be written as

$$\boldsymbol{N}_{R\star} = -\frac{6}{r_i^5}C_\star\frac{\boldsymbol{r}_i\cdot\boldsymbol{\Omega}_\star}{\boldsymbol{\Omega}_\star^2}(\boldsymbol{r}_i\times\boldsymbol{\Omega}_\star) \tag{37}$$

[11] In Bolmont et al. (2015) $C_\star$ and C_i are reversed.

and

$$\boldsymbol{N}_{Ri} = -\frac{6}{r_i^5} C_i \frac{\boldsymbol{r}_i \cdot \boldsymbol{\Omega}_i}{\boldsymbol{\Omega}_i^2} (\boldsymbol{r}_i \times \boldsymbol{\Omega}_i). \tag{38}$$

The rotational deformation torque is implemented the same way as the tidal torque described in Section 4.1.

5. Forces and Models for Small Bodies

Simulating small bodies of the solar system (or other planetary systems) often requires the integration of a very high number of particles. There are more than 1 million asteroids known in the asteroid belt (https://ssd.jpl.nasa.gov/), and taking into account also the (unseen) smaller fragments would lead to immense numbers, exceeding current computational capabilities. Still, the presence of all these small bodies is important for the dynamics of meteoroids: collisions can lead to both break-up events and changes in meteoroid rotation rate. Together with the Yarkovsky effect and Poynting–Robertson drag, collisions between small bodies can influence the migration rate of asteroids and thus generate impact events on the Earth or other planets. An alternative to simulating such collisions with an *N*-body integrator is to include a probabilistic collision model, which uses an average probability that a given small body would collide with another small body (Farinella et al. 1998). Note that the presented probabilistic collision model is different from the real collision handling of GENGA and is applied only to test particles. Collisions between massive bodies or between a massive body and a test particle can still occur in the usual way (see Section 7.1). We do not include the YORP effect in this work, although it can also have an important contribution (Rubincam 2000; Vokrouhlický et al. 2015). While the Yarkovsky effect only alters the orbital elements of small radiated bodies, the YORP effect would also change the rotation rate and the obliquity of the body. Contrary to the Yarkovsky effect, the YORP effect depends on the shape of the object, which makes it harder to apply for real solar system asteroids. In this section, we describe first the implementation of the Yarkovsky effect and the Poynting–Robertson drag, followed by a description of a model for collisional break-up events and rotation changes of small bodies. At the end of this section, we present an example simulation including these effects.

5.1. Yarkovsky Effect

The Yarkovsky effect is caused by thermal radiation forces, acting on small rotating objects (Öpik 1951). Typically, the Yarkovsky effect is important for asteroids or meteoroids with a diameter smaller than ∼40 km (Bottke et al. 2006). It causes a migration in the semimajor axis and is especially important to deliver material into mean-motion or secular resonances with the giant planets, where the affected material undergoes further dynamical effects such as eccentricity pumping (e.g., Brož 2006).

The Yarkovsky effect emerges from the anisotropic re-emission of thermal radiation from the heated surface of the rotating object, which leads to an orbital acceleration. The effect can be split into two different types, the diurnal Yarkovsky effect and the seasonal Yarkovsky effect. The diurnal effect is the strongest when the spin axis is perpendicular to the orbital plane and can cause both inward migration and outward migration depending on the spin direction. The seasonal effect is the strongest when the spin axis of the object lies in the orbital plane, and leads to an inward orbital migration. The physical parameters and their values that go into computing the Yarkovsky effect are summarized in Table 1, and a more detailed description of the Yarkovsky effect is given in, e.g., Farinella et al. (1998), Vokrouhlicky (1998), Vokrouhlický & Farinella (1999), Vokrouhlický et al. (2000), Bottke et al. (2000, 2006), or Brož (2006).

Table 1
Parameters and Initial Conditions for the Yarkovsky Effect Test

Semimajor axis	a	2 au
Eccentricity	e	0
Inclination	i	0°
Material density	ρ	3500.0 kg m^{-3}
Physical radius	R	0.1–100 m
Rotation frequency	ω	5 hr
Obliquity	γ_{Seasonal}	90°
Obliquity	γ_{Diurnal}	0
Thermal conductivity	K	2.65 W m^{-1} K^{-1}
Specific heat capacity	C	680 J kg^{-1} K^{-1}
Thermal emissivity	ϵ	1.0
Asteroid's Bond albedo	A	0.0
Solar constant	S	1367 W m^{-2}.

Following Vokrouhlicky (1998), we define the radiation force factor

$$\Phi = \frac{\pi R^2 F}{mc}, \tag{39}$$

with the physical radius R, the mass m, the speed of light c, and the Solar radiation flux F (in flux units) at the heliocentric distance d given by

$$F = \frac{S}{d^2}. \tag{40}$$

Here S is the Solar constant in W m^{-2} and d is the time averaged heliocentric distance in au, i.e.,

$$d = a(1 + \tfrac{1}{2}e^2), \tag{41}$$

with a the semimajor axis and e the orbital eccentricity.

Following Vokrouhlicky (1998) and Vokrouhlický & Farinella (1999) further, we define the thermal inertia Γ as

$$\Gamma = \sqrt{\rho C K}, \tag{42}$$

which depends on the material density ρ, the specific heat capacity C, and the thermal conductivity K. We also define the thermal parameter

$$\Theta_{\text{seasonal}} = \frac{\Gamma \sqrt{n}}{\epsilon \sigma T_\star^3}, \tag{43}$$

where σ is the Stefan–Boltzmann constant, $T_\star$ the subsolar surface temperature, and n is the orbital mean motion. The subsolar surface temperature is set by

$$T_\star^4 = \frac{(1 - A)F}{\epsilon \sigma}, \tag{44}$$

with the Bond albedo A and the thermal emissivity ϵ. Following the derivation in Vokrouhlický & Farinella (1999, Appendix B), the acceleration of the seasonal Yarkovsky effect can then be written as

$$\boldsymbol{a}_{\text{seasonal}} = \frac{4}{9}\frac{(1-A)\Phi}{(1+\lambda)} \times \sum_{k\geqslant 1} G_k[s_P\alpha_k\cos(knt+\delta_k) + s_Q\beta_k\sin(knt+\delta_k)]\boldsymbol{s}, \tag{45}$$

with the spin vector $\boldsymbol{s}$ and the quantities λ, α, β, $(G_k\cos\delta_k)$, $(G_k\sin\delta_k)$, s_P, and s_Q as defined in Vokrouhlický & Farinella (1999, Appendix B). In the current implementation in GENGA, we only use $k = 1$ in the summation of Equation (45).

Similarly, we follow Vokrouhlický et al. (2000) for the diurnal Yarkovsky effect and define

$$\Theta_{\text{diurnal}} = \frac{\Gamma\sqrt{\omega}}{\epsilon\sigma T_\star^3}, \tag{46}$$

with the rotational frequency ω. The diurnal effect can then be written as

$$\boldsymbol{a}_{\text{diurnal}} = \frac{4}{9}\frac{(1-A)\Phi}{(1+\lambda)}G[\sin\delta + \cos\delta\boldsymbol{s}\times]\frac{\boldsymbol{r}\times\boldsymbol{s}}{r}. \tag{47}$$

The quantity δ is a thermal lag angle (Vokrouhlický & Farinella 1999). After having computed the total Yarkovsky acceleration

$$\boldsymbol{a}_Y = \boldsymbol{a}_{\text{seasonal}} + \boldsymbol{a}_{\text{diurnal}},$$

the new velocity of the body is updated with a velocity kick of the time step dt as

$$\boldsymbol{v}^{n+1} = \boldsymbol{v}^n + \boldsymbol{a}_Y dt.$$

5.2. Orbit-averaged Yarkovsky Effect

As an alternative to calculating the acceleration of the Yarkovsky effect at every time step, an orbit-averaged drift in the semimajor axis can be applied.

For circular orbits and by linearizing the heat conductivity, the orbit-averaged change in the semimajor axis a can be written following Vokrouhlický & Farinella (1999) and Vokrouhlický et al. (2000) as

$$\left(\frac{da}{dt}\right)_{\text{diurnal}} = -\frac{8}{9}\frac{(1-A)\Phi}{n}\frac{G\sin\delta}{1+\lambda}\cos\gamma \tag{48}$$

and

$$\left(\frac{da}{dt}\right)_{\text{seasonal}} = \frac{4}{9}\frac{(1-A)\Phi}{n}\sum_{k\geqslant 1}\frac{G_k\sin\delta_k}{k}\chi_k\bar{\chi}_k, \tag{49}$$

where γ is the spin axis obliquity, and the quantities G, δ, λ, and χ are defined in Vokrouhlický & Farinella (1999) and Vokrouhlický et al. (2000). Similar formulations are derived in Farinella et al. (1998), Vokrouhlický & Farinella (1999). A more precise model of the Yarkovsky effect is described in Vokrouhlický et al. (2000) or Vokrouhlický & Farinella (1998), and involves the numerical integration of the heat diffusion equation. For simplicity, we do not use this model in the current implementation of GENGA. The user can enable the direct Yarkovky effect or the orbit-averaged Yarkovsky effect in the GENGA parameter file. Applying the averaged Equations (48) and (49) requires a conversion between Cartesian coordinates to Keplerian elements and back, which causes computational overhead.

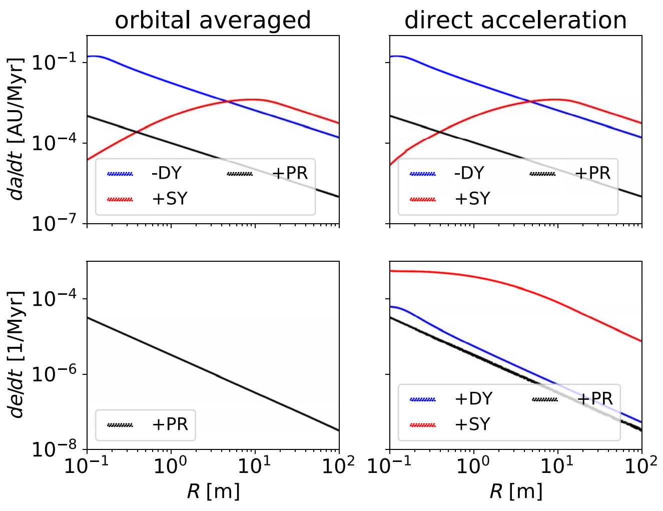

Figure 3. Test of the seasonal and diurnal Yarkovsky effect and the Poynting–Robertson drag for the values of Table 1. (For the Poynting–Robertson drag, we use $e = 0.05$ instead of 0.) Shown are both methods, the orbital-averaged drift method da/dt (de/dt) (left panels) and the direct acceleration $\boldsymbol{a}_Y$, $\boldsymbol{a}_{\text{PR}}$ (right panels). Note that the orbital-averaged drift method for the Yarkovsky effect only affects the semimajor axis and not the eccentricity. This plot reproduces the result from Bottke et al. (2000, Figure 4) for the Yarkovsky effect.

5.3. Test of the Yarkovsky Effect

We test the Yarkovsky effect with the same initial conditions as given in Farinella et al. (1998, Section 4) and Bottke et al. (2000, Section 4.1), which are summarized in Table 1. We generate 1000 test particles with a physical radius between 0.1 and 100 m and integrate them for 10,000 yr with both methods: the direct and the orbital-averaged method. The direct acceleration method is simpler to apply into the N-body integrator and is in practice also faster than the orbital-averaged drift method because it does not require the conversion to orbital elements and back again. Therefore we recommend to use the direct acceleration method in simulations that include the Yarkovsky effect. The results are shown in Figure 3 reproduce the results from Bottke et al. (2000, Figure 4). Note that there is a factor of 2 difference between the results of Bottke et al. (2000, Figure 4) and Farinella et al. (1998, Figure 1).

Another example including the Yarkovsky effect is presented in Section 5.8.

5.4. Poynting–Robertson Drag

When a body is so small that its orbital momentum becomes comparable to the total momentum caused by solar photons (typically below the centimeter to millimeter range) it encounters, it becomes subject to Poynting–Robertson drag, for which many different derivations are known, e.g., Robertson (1937), Burns et al. (1979, 2014).

Following Burns et al. (1979), the change in the semimajor axis a and the eccentricity e due to this force can be written as

$$\frac{da}{dt} = -\frac{\eta}{a}Q_{\text{pr}}\frac{(2+3e^2)}{(1-e^2)^{3/2}} \tag{50}$$

and

$$\frac{de}{dt} = -\frac{5}{2}\frac{\eta}{a^2}Q_{\text{pr}}\frac{e}{(1-e^2)^{1/2}}, \tag{51}$$

with

$$\eta = \frac{S_0 r_0^2 A}{mc^2} \tag{52}$$

where A is the cross section of the particle, S_0 is the Solar constant at 1 au, and $r_0 = 1$ au. The radiation pressure coefficient $Q_{\rm pr}$ is assumed to be 1 for purely absorbing particles.

Implementing the Equations (50) and (51) in an N-body integrator requires the transformation from Cartesian coordinates to Keplerian elements and back to Cartesian coordinates, after the terms (50) and (51) have been applied. These conversations would require a substantial amount of the total computing time. A more efficient approach is to apply the velocity change directly as (Burns et al. 1979)

$$\boldsymbol{a}_{\rm PR} = \frac{d\boldsymbol{v}}{dt} = \frac{\eta c}{r^2} Q_{\rm pr}\left[\left(1 - \frac{\dot{r}}{c}\right)\hat{r} - \frac{\boldsymbol{v}}{c}\right]. \tag{53}$$

Since the acceleration in Equation (53) depends on the velocity, we use a few implicit midpoint iterations to perform the velocity kick operation in a symplectic way. Typically, less than three iterations in the implicit midpoint method are needed to converge to machine precision.

5.5. Test of the Poynting–Robertson Drag

We repeat the same test as in Section 5.3 for the Poynting–Robertson drag, with the exception that here we use an eccentricity of 0.05 instead of 0 the get non-zero values for de/dt. The results are shown in Figure 3 together with the Yarkovky effect, and show that the Poynting–Robertson drag is important only for small particles with a radius <1 m.

For another test of the Poynting–Robertson drag we follow Kortenkamp (2013), who uses numerical simulations to follow the orbital evolution of dust particles in the solar system. The Poynting–Robertson drag causes the semimajor axis to decay, and the particle drifts toward the Sun until it eventually reaches a resonance condition with a planet. Dust particles can then be trapped in this resonance condition for many tens of thousands of years. Similar to Kortenkamp (2013), we analyze the orbital evolution of a 28 μm sized dust particle in the presence of all plants of the solar system. In Figure 4, the evolution of the semimajor axis and of the eccentricity of a dust particle are depicted. After 3000 yr, the dust particle gets trapped in orbital resonance with the Earth, specifically the 2:3, and the eccentricity begins to increase until it reaches a value of ∼0.25 and the particle leaves the resonance condition again after 90,000 yr.

5.6. Collisional Break-up Model

The asteroid belt of the solar system contains a very large number of objects. This means that particles within the asteroid belt will eventually collide with some others. It is computationally not feasible to include all asteroids in every simulation so that it is desirable to describe the asteroid belt by a probabilistic collision model. Such a model uses the probability that an object would collide with another object in the asteroid belt and uses random numbers to generate virtual collision events. These events can either lead to a reset of the rotation rate of a given object or destroy the object and replace it with new generated fragment particles. In the following, we describe the details of the collisional break-up model, which generates new fragment particles on the fly, and the rotation reset model. We emphasize that the collisional break-up model and the rotation-rate reset model are tailored to the asteroid belt between Mars and Jupiter.

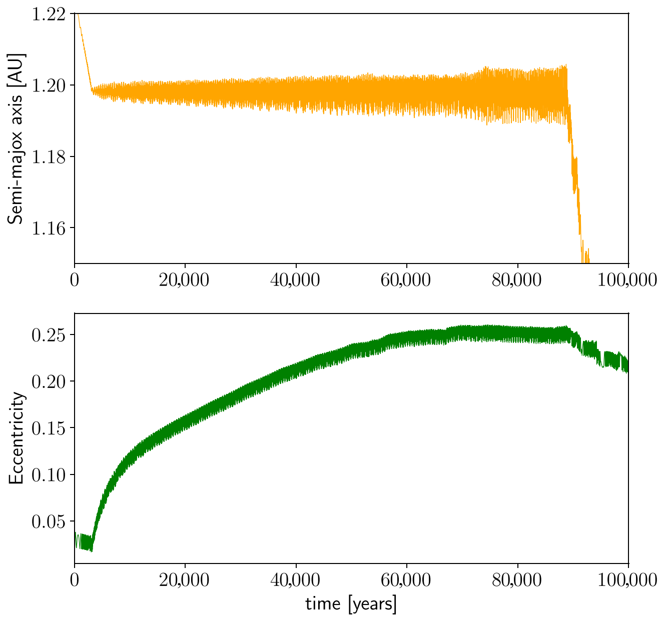

Figure 4. Orbital evolution of a 28 μm sized dust particle under the influence of Poynting–Robertson drag and the presence of the solar system planets. The dust particle drifts toward the Sun and gets trapped in a resonance condition around the Earth. After 90,000 yr, the eccentricity gets large enough that the dust particle can leave the orbital trap again.

Following Farinella et al. (1998), we use a collisional lifetime of

$$\tau_{\rm col} = 20\sqrt{\frac{R}{1\ {\rm m}}}\ {\rm Myr}, \tag{54}$$

which is equivalent to a break-up probability of $1/\tau_{\rm col}$ per year. When a small body breaks up, its mass is replaced by a series of fragments with radii between the radius of the original object $R_{\rm o}$, and $R_{\rm m} = 0.01$ m. The minimum radius $R_{\rm m}$ is a compromise between run time and accuracy. Having a smaller value for $R_{\rm m}$ would generate much more particles, which slows down the integration speed. Setting it to a larger value would generate less particles. The value of $R_{\rm m}$ can be set as a user parameter. We use the following distribution to generate the radii of the new fragments $R_{\rm f}$, which is given by

$$R_{\rm f} = \left[\left(R_0^{(n+1)} - R_{\rm m}^{(n+1)}\right)\cdot y + R_{\rm m}^{(n+1)}\right]^{1/(n+1)}, \tag{55}$$

with $n = -3/2$ and a uniform generated random number $y \in (0, 1)$. The mass of each generated fragment is subtracted from the remaining mass of the original body. The first fragment to exceed the remaining mass gets the remaining mass.

For the velocity distribution of the fragments, we use a scaling value

$$u = y \cdot m_{\rm f}^{-1/6}, \tag{56}$$

with the mass of the fragment $m_{\rm f}$ and a random number $y \in (0.8, 1.2)$ (Nakamura & Fujiwara 1991). A constant velocity budget of 31 m s^{-1} (based on $V_{\rm budget} = (0.3/N)^{0.5} \times V_{\rm impactor}$, and $N = 8000$; see Wiegert 2015 for details) is distributed to the fragments in proportion to the scaling value u. A randomly distributed velocity vector with the given length sets the new orbit of the particle. Typical values of these created fragment velocities are of the order of $\sim$0.1–1 m s^{-1}.

In GENGA, fragments with a radius smaller than $R_{\rm remove} = 0.01$ m are removed to prevent having too many particles in the simulation. The value of $R_{\rm remove}$ can be set as a user parameter. The spin axis direction is generated randomly following a two-dimensional bivariate normal distribution over the solid angle. For the rotation rate, we take

$$\omega_{\rm f} = \mathrm{random}\left(\frac{1}{36 R_{\rm f}}, \frac{1}{R_{\rm f}}\right), \tag{57}$$

which reproduces the range of rotation rates typically observed for Near Earth Asteroids in the meter (Beech & Brown 2000) to kilometer (Farinella et al. 1998) size range.

5.7. Collisional Reset Model of Rotation Rate and Obliquity

For each object at each time step, we calculate the probability that its rotation was reset during the last time step by a virtual collision with another particle as a function of its radius R and previous rotation rate. The rotation will be changed if the impactor has an angular momentum at the collision time similar to the rotational angular momentum of the target object, such that the radius of the projectile is given as (Farinella et al. 1998)

$$r_{\rm projectile} = \left(\frac{2\sqrt{2} R \omega \rho_{\rm target}}{5 \rho_{\rm projectile} V}\right)^{1/3} R \tag{58}$$

which, assuming $\rho_{\rm target} = \rho_{\rm projecile}$, simplifies to

$$r_{\rm projectile} = \left(\frac{2\sqrt{2}\,\omega}{5V}\right)^{1/3} R^{4/3}. \tag{59}$$

Here V is the typical collisional velocity in the asteroid belt ($\sim$5 km s^{-1}). The probability of a rotation-rate reset through collision with a large enough projectile is given by

$$\frac{1}{\tau_{\rm rot}} = 3 \times 10^5 P_i R^2 r_{\rm projectile}^{-5/2}\ \mathrm{s}^{-1} \tag{60}$$

with P_i being the intrinsic collisional probability for the asteroid belt, 2.85×10^{-18} km^2 yr^{-1} (Farinella et al. 1998). Using this number, and replacing $r_{\rm projectile}$ in Equation (60) with the expression in Equation (59) results in a final probability of

$$\frac{1}{\tau_{\rm rot}} = 10^{-18} R^{-4/3} \left[\frac{2\sqrt{2}\,\omega}{5V}\right]^{-5/6}\ \mathrm{s}^{-1} \tag{61}$$

where all values are in SI units. If the rotation rate is reset, then both the obliquity and the new rotation rate are randomized as described in Section 5.6.

5.8. Example Simulation of Debris Particles

To test the new functions and forces for small bodies we simulate the transport of ejecta material from the asteroid 6 Hebe to the Earth, in a similar fashion to that described in Bottke et al. (2000) and Brož (2006). The asteroid 6 Hebe is thought to be a parent body of the H chondrites (Gaffey et al. 1993), one of the most abundant types of meteorites falling to Earth today. It is located close to mean-motion resonances with Jupiter (mainly the 7:2 and the 3:1 mean-motion resonances), which makes the transport of small material very efficient. The ejecta material migrates via the Yarkovsky effect into the resonance, where the eccentricity is increased and the particles can reach an Earth-crossing orbit and some particles will hit the surface of the Earth or also other inner planets (Bottke et al. 2000). As initial conditions of our simulation, we use the coordinates of the solar system planets and the asteroid 6 Hebe 50 Myr ago, which we obtain from a backward integration of today's positions and velocities. For the backward integration, we include the solar system's planets and Pluto, following the idea of Bottke et al. (2000), where the planets Venus to Neptune are included. To obtain a more precise integration, also the largest asteroids of the solar system could be included. We note that also the precise time of the break-up event can influence the dynamics of the ejecta particles. Following Bottke et al. (2000), we chose 50 Myr. At the location of 6 Hebe, we generate 39,574 test particles with random spins and radii following a power-law distribution with slope $n = -3/2$ between $r_0 = 0.01$ m and $r_1 = 50$ m. From there, we simulate the system forward in time with and keep track of all encounter and collision events with the planets. In Figure 5 are shown four simulations, all starting with the same initial conditions. The first simulation (left column) contains only gravity and no additional forces or models. Since the ejecta particles have an initial velocity, some of them are located already in resonant conditions with Jupiter. There, their eccentricities are increased and the particles migrate toward the inner solar system, where the inner planets damp the eccentricities again and the particles can reach Earth-crossing orbits or even collide with a planet. After about 10 Myr, the bulk of the ejecta particles has reached the inner solar system.

The second column of Figure 5 includes the Yarkovsky effect and Poynting–Robertson drag. With these two effects, debris particles can constantly drift into the resonances with Jupiter and lead to many more particles reaching the inner solar system over a protracted time period.

The third column of Figure 5 adds the rotational reset model, described in Section 5.7. The result looks very similar to the second column. Differences would only occur at a later time, where the rotation reset model leads to a random walk in the semimajor axis of the debris particles.

The fourth column of Figure 5 includes also the fragmentation model, described in Section 5.6. Since the number of particles grows exponentially with this model, we start with only 10% of the particles as initial conditions. In order to speed up the calculation, we stop the simulation after 6 Myr manually, remove 90% of the particles (i.e., we reduce the number of particles from 126,564 to 12,656) and restart the simulation again from there. Additionally, the fragmentation model constantly removes all new generated particles with a radius of less than 0.01 m. The fragmentation model leads to many more smaller particles, which can increase the migration rate of the particles into the resonances but also increases the computation time dramatically.

In Figure 6, we show all encounter and collision events with the Earth. The aim of this figure is to display the different collision rates and collision timeline for the different forces that are applied. The green bars display encounters with the Earth within 50 Earth radii, while the blue histograms show collisions with the Earth. It is clear that, in the gravity-only simulation, the number of encounters peaks between 5 and 10 Myr and

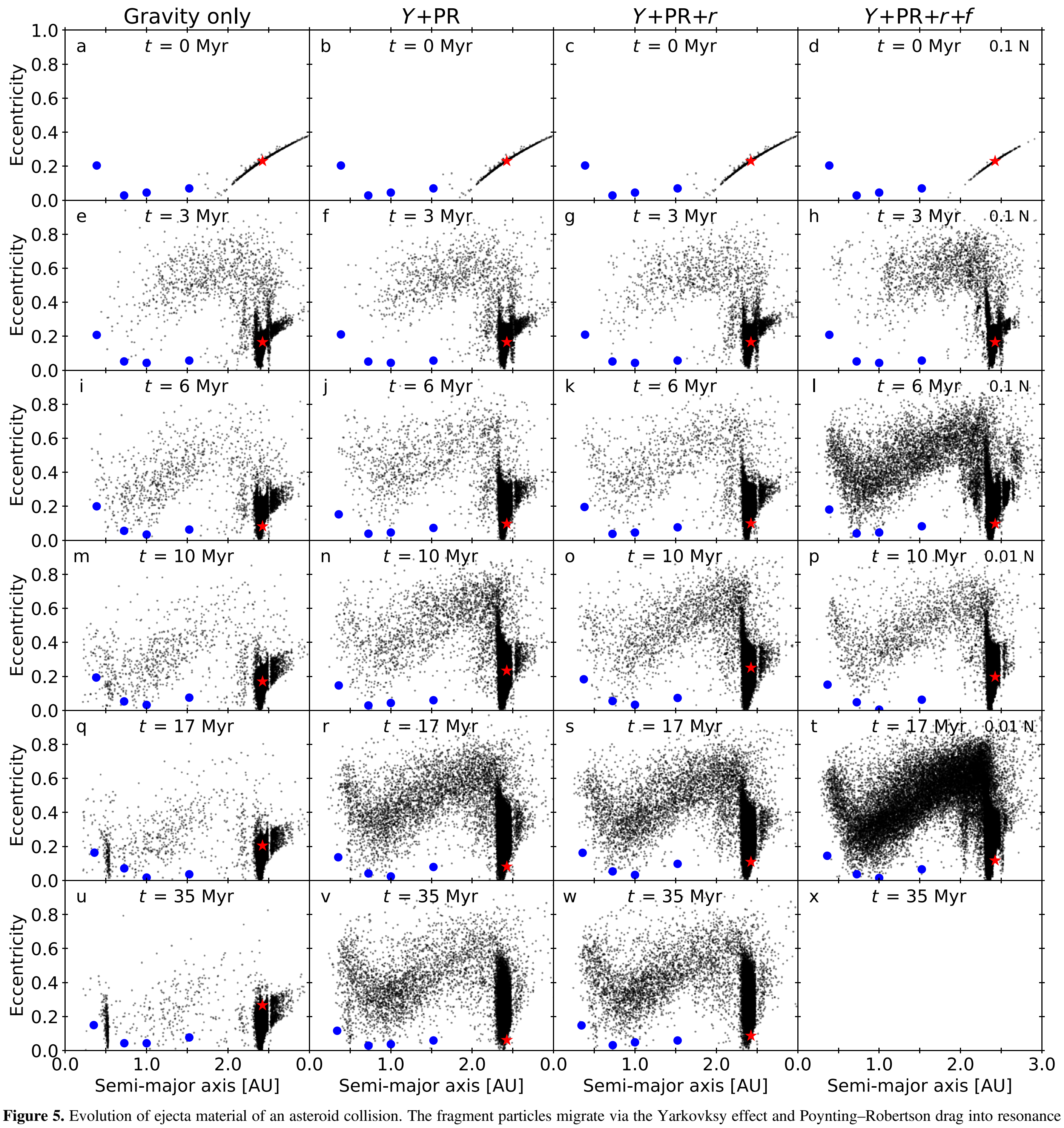

Figure 5. Evolution of ejecta material of an asteroid collision. The fragment particles migrate via the Yarkovksy effect and Poynting–Robertson drag into resonance conditions with Jupiter, where the eccentricities of the particles get increased. Later, the eccentricities get damped from the inner planets and finally they can reach Earth-crossing orbits and even hit the surface of the Earth. The left column shows the evolution of the fragment particles with only the gravitational force included, the second column includes the Yarkovsky effect and Poynting–Robertson grad (Y + PR). The third column includes the rotational reset model (Y + PR + r), and the fourth column includes the fragmentation model (Y + PR + r + f). The simulation in the fourth column starts with 10 times fewer particles as the others (indicated with "0.1 N"), and after 6 Myr, the number of particles is again reduced manually by a factor of 10 (indicated with "0.01 N") to limit the total number of particles. The blue dots show the positions of the inner planets and the red star indicates the position of the 6 Hebe parent body. The last sub-panel of the Y + PR + r + f simulation (sub-panel (x)) is empty because this simulation only reached 17 Myr.

then slowly declines. In the cases with Y + PR and Y + PR + r the number of encounters increases and remains roughly steady between 20 and 35 Myr. In the bottom panel the number of encounters and collisions increases rapidly until 17 Myr, after which the simulation is stopped.

6. Improvements in the Close-encounter Integration and High-resolution Simulations

In the following section, we describe the updates to the close-encounter integration routines. First, it includes new parallelization

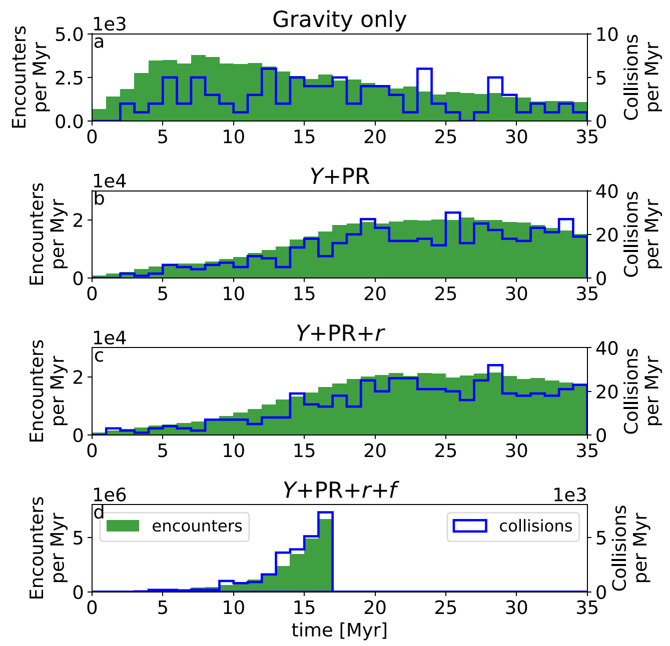

Figure 6. Close-encounter and collision events of the ejecta material with the Earth for the same simulations shown in Figure 5. Close encounters in this figure are encounter events with a closest distance to the Earth of less than 50 Earth radii. The data of the $Y + \mathrm{PR} + r + f$ simulation (panel (d)) is scaled up to the original number of initial particles.

schemes for the Bulirsch–Stoer kernels, which allow us to integrate much larger close-encounter groups. Second, we introduce a hierarchical method that uses multiple changeover functions, because it is able to integrate large close-encounter groups much more efficiently. We demonstrate how this new method can increase the computational speed significantly.

6.1. The Bulirsch–Stoer Direct Integration Kernels

In GENGA, close encounters are integrated with a Bulirsch–Stoer method. All particles in a given close-encounter group need to be integrated synchronously. To achieve a good performance, it is necessary to always use the fastest available memory on the GPU.[12] While small close-encounter groups can be integrated within a single thread block and benefit from fast shared memory, larger groups must be distributed across multiple thread blocks and are more affected by a slower global memory access and kernel overhead time. Depending on the close-encounter group size n, different parallelization methods are needed. In the following, we describe four different methods to cover the range from $n = 2$ to $n \gg 1000$ and summarize the parallelization parameters used in Table 2. GENGA sorts all close-encounter pairs into independent groups, which can be integrated in parallel. Different groups of the same size can be easily launched in parallel by using multiple thread blocks. To launch different versions of the kernels, different CUDA streams can be used.[13] Table 2 indicates the stream index, the number of threads per thread block and the number of thread blocks per close-encounter group of the different Bulirsch–Stoer kernel implementations.

Table 2
Parameters for the Close-Encounter Integration Parallelization

Group Size	Kernel Name	Stream Index	Threads per Block	Thread Blocks
2	BS-B	0	4	1
3-4	BS-B	1	16	1
5-8	BS-B	2	64	1
9-16	BS-B	3	256	1
17-32	BS-B	4	256	1
33-64	BS-A	5	64	1
65-128	BS-A	6	128	1
129-256	BS-A	7	256	1
255-512	BS-A or BS-M	8	256	1 or 4
513-1024	BS-M	9	256	4
1025-2048	BS-M	10	256	8
…	BS-M	…	256	…

The different close-encounter kernels are BS-B, BS-A, and BS-M, which are described in detail below. The BS-M kernels cannot use more than 256 threads per thread block, because of a hardware limitation in the number of registers.

6.1.1. The BS-B Kernel

Grimm & Stadel (2014) describes an implementation of the Bulirsch–Stoer method (BS-B) by using shared memory and a full n^2 parallelization, where n is the size of the close-encounter group. This implementation is very efficient for small values of n, but it is not applicable for large n because of limitations of the shared memory size on the GPUs and the number of threads per thread block. Due to hardware limitations, we can use this kernel only for groups up to a size of 32 bodies. The BS-B kernel is still identical to the description in Grimm & Stadel (2014).

6.1.2. The BS-A Kernel

The BS-A kernel does not use an n^2 neighbor accessing scheme, but it uses a list of all close-encounter partners, and all involved particles iterate through that list. Since not all particles must have the same number of close-encounter pairs, this implementation can lead to thread divergences,[14] which affects the occupancy of the kernel. Reading and iterating the close-encounter pairs list requires additional computing time. The BS-A kernel is less efficient than the BS-B kernel for small values of n, and since the BS-B kernel is not scalable to $n > 32$, it is the best alternative for groups sizes up to 256 due to limitations on shared GPU memory. Even larger close-encounter groups would need too much shared memory and another routine is required.

[12] GPUs contain different memory types with different sizes and access speeds. The fastest and most limited are the local registers, followed by the shared memory. The largest and slowest memory type is the global memory. Differences in access times between these memory types can easily be a factor of 100 or more. The CUDA programming model is organized in threads, warps, thread blocks, and grids. The smallest units are the threads; they can be executed in parallel by the CUDA cores. A collection of 32 (or 64) consecutive threads is called a warp. All threads in a warp must perform the same operation. They can communicate via warp shuffle functions with registers or via shared memory. A thread block can contain up to 1024 threads. Threads in a thread block can communicate via shared memory, they cannot communicate with threads in other thread blocks. Communication between thread blocks is only possible via the CPU and different kernel calls and by using global memory.

[13] A CUDA stream is a sequence of GPU operations, e.g., kernel calls or memory transfers. Different CUDA streams can be executed concurrently in parallel.

[14] A thread divergence happens if some threads in a warp should execute a different operation than others. Since this is not possible, the two operations are serialized and performed one after the other. In that case, the number of active threads in the warp is reduced, which is called a lower occupancy. The occupancy depends on the number of bodies and the initial conditions. For a simulation with 8192 particles, we measured an occupancy of about 75%.

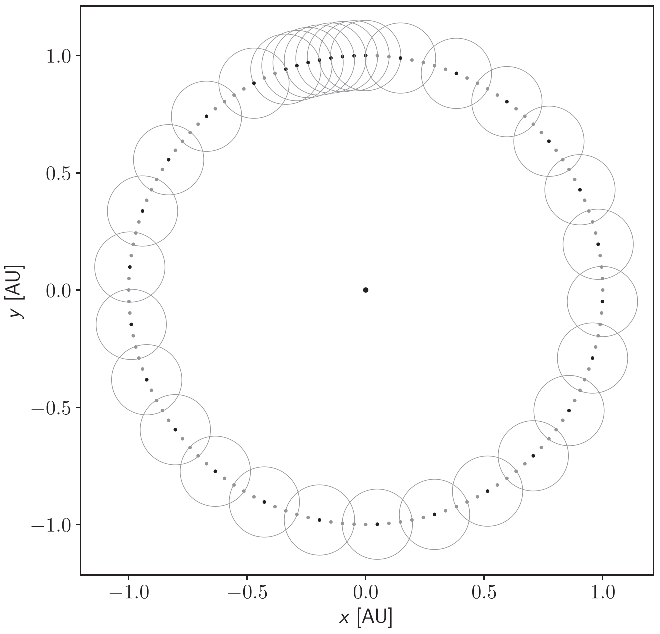

Figure 7. Initial conditions for the close-encounter performance test. N equal-mass particles are regularly positioned in a ring around a central mass. The critical radii of the particles is chosen such that every particle is in a close encounter with eight other neighboring particles. Since the critical radii overlap, all particles are grouped together into a single close-encounter group. The figure shows an example with $N = 128$. The circles show the critical radii of the particles, but for clarity of the figure, we do not show them for all particles.

6.1.3. The BS-M Function

For $n > 256$, the integration cannot be parallelized further within only a single thread block, but instead the close-encounter groups must be distributed over multiple thread blocks. This also requires that the sub-steps of the Bulirsch–Stoer integration are controlled from the host (i.e., the CPU) and therefore this involves many different kernel calls and communication between CPU and GPU over the motherboard and access to global memory. It is clear that this method suffers from kernel overhead time and global memory access which slow the computation down due to communication bottlenecks. On the positive side, this method can be applied to all group sizes.

6.2. Close-encounter Performance Test

To test the performance of the different implementations as a function of group sizes, we position N particles regularly spaced on a ring around a central mass as shown in Figure 7. We set the critical radii of the particles in a way that every particle is in a close encounter with eight neighboring particles (for $N < 16$ we use all available neighbors). In this way, all N particles are grouped together into a single close-encounter group. Figure 8 shows the execution time per time step of the different methods, depending on the number of bodies N. The used initial conditions are very synthetic and, in real simulations, there are certainly situations where the performance of the different implementations can differ, but it demonstrates well the importance of fast memory usage and kernel overhead. This test shows also that integrating large close-encounter groups is the performance bottleneck in simulations with many particles. In practice, large close-encounter groups ($\gtrsim$128) should be avoided if possible; in Section 6.3, we present a way to achieve that. In the latest

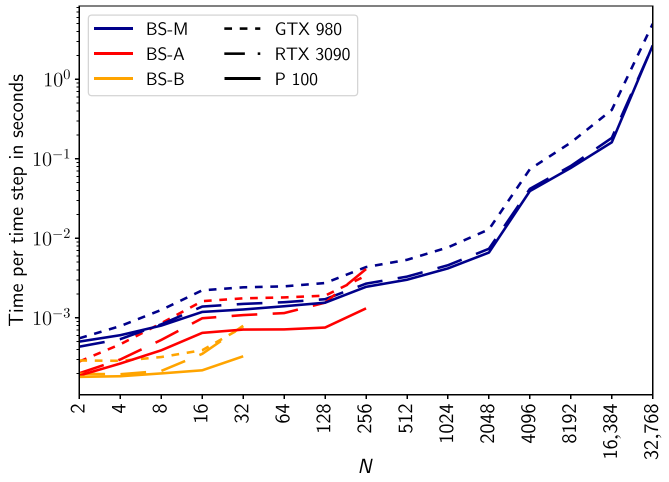

Figure 8. Performance of different parallelization implementations of the Bulirsch–Stoer method used for the close-encounter integration. The used initial conditions for this speed test are shown in Figure 7. The BS-B method benefits from the fast data communication through shared memory, but it is not scalable to more than 32 particles. For larger group sizes, more and more global memory must be used, which leads to a performance penalty. The steep ascend from $N = 2$ to $N = 16$ for the BS-A and BS-M methods comes from the fact that there are not enough neighbor particles available for eight close-encounter partners. Shown is the data for three different GPU types, a GTX 980, an RTX 3090, and a Tesla P100.

version of GENGA, we use the BS-B, BS-A, and BS-M methods for close-encounter group sizes, as indicated in Table 2.

6.3. Higher Level Changeover Functions

The original hybrid symplectic integrator described in Section 2 consists of two levels: the basic level (level 0), which is the symplectic integrator with a fixed step size wherein bodies revolve around the central mass, and the Bulirsch–Stoer method (level 1), which is used for close encounters between bodies other than the central mass. A changeover function smoothly switches between the two levels. This method can be extended by introducing a number of intermediate levels between the fully symplectic regime and the Bulirsch–Stoer regime: instead of switching directly to the Bulirsch–Stoer method another symplectic integration step with a reduced time step can be applied. This is desirable because the Bulirsch–Stoer steps are computationally expensive and should be used as a last resort. With the introduction of an additional level, a second changeover function can then smoothly switch between the new additional shorter symplectic step and the Bulirsch–Stoer method. In this way, the second argument in the critical radius (Equation (5)), $n_2 \cdot v \cdot dt$, gets reduced in the higher levels because of the smaller time step. The Bulirsch–Stoer method is called much later in the close-encounter phase, and is therefore used much less frequently. This method can reduce the close-encounter group sizes requiring the Bulirsch–Stoer integration saving valuable computation time. In practice, there can be multiple such intermediate levels. This new formalism lies in between of the original hybrid symplectic integrator and the “multiple time step symplectic algorithm” as implemented in the SyMBA code (Duncan et al. 1998). While the described method in this work also uses multiple symplectic time steps, unlike SyMBA, the deepest level is still calculated with a direct N-body method. The reduction factor on the time step of the intermediate levels

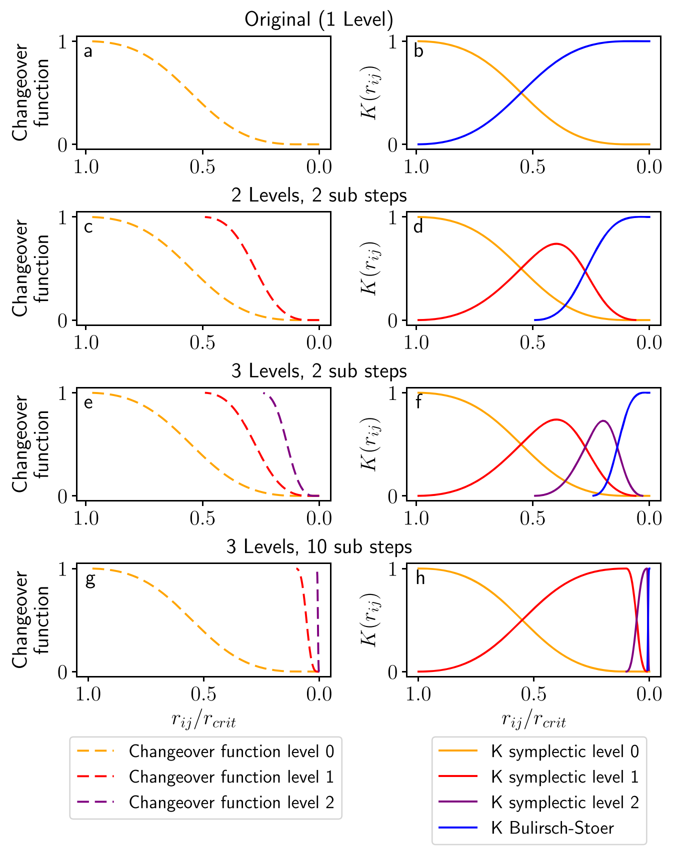

Figure 9. Panels (a) and (b): original hybrid symplectic method, a changeover function switches smoothly from K symplectic to K Bulirsch–Stoer. Panels (c) and (d): a second changeover function is included. The basic symplectic step is divided in the first level into two sub-steps. The second level is the Bulirsch–Stoer method. Panels (e) and (f): two additional changeover functions are included for a three-level scheme with two sub-steps in each level. Panels (g) and (h): two additional changeover functions are included for a three-level scheme with 10 sub-steps in each level.

can be set freely by the user. Common options are two, four, or 10, but any other number is also possible. Note that in SyMBA, a factor of three is used by default. Since the changeover function is acting in the range of $1.0 < r_{ij}/r_{\rm crit} < 0.1$[15] the choice of splitting a level into 10 sub-steps leads to a special case where no more than two levels are active at the same time. Figure 9 shows four examples of different numbers of symplectic levels and sub-steps per level. The panels (a) and (b) are identical to the original hybrid symplectic integrator, with a single changeover function $K(r_{ij})$. In this case we define

$$\begin{aligned} K_{\rm symplectic} &= K(r_{ij}) \\ K_{\rm Bulirsch-Stoer} &= 1 - K(r_{ij}). \end{aligned} \tag{62}$$

The panels (c) and (d) show an example with two levels and each level is split into two sub-steps. In this case we define

$$\begin{aligned} K_{\rm symplectic\ 1} &= K(r_{ij}, dt) \\ K_{\rm symplectic\ 2} &= (1 - K(r_{ij}, dt)) \cdot K(r_{ij}, dt/2) \\ K_{\rm Bulirsch-Stoer} &= (1 - K(r_{ij}, dt)) \cdot (1 - K(r_{ij}, dt/2)), \end{aligned} \tag{63}$$

with the base time step dt.

Panels (e) and (f) show three levels with two sub-steps, and panels (g) and (h) show three levels with 10 sub-steps. In all examples, the Bulirsch–Stoer method is only called at the blue lines.

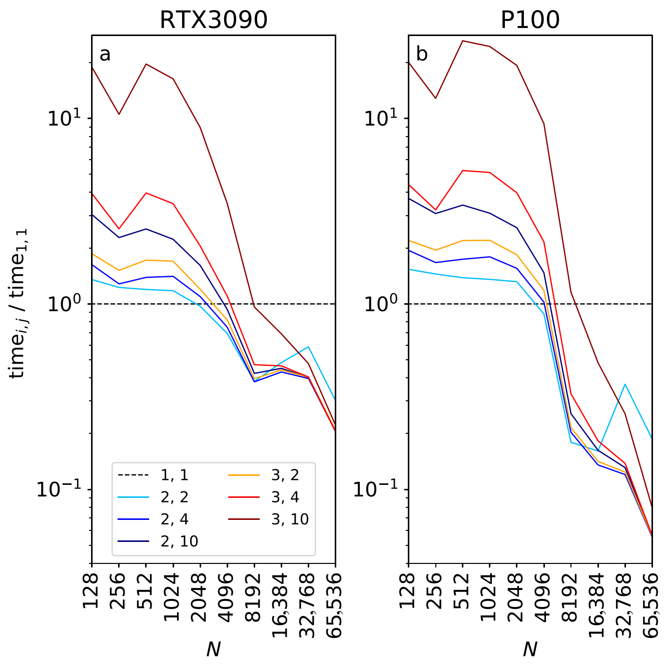

Figure 10. Performance test of higher symplectic levels. The shown time is normalized to the time of the original hybrid symplectic integration method (1, 1). The legend indicates the number of symplectic levels, followed by the number of sub-steps per level. The initial condition is a disk of N planetesimals, distributed between 0.5 and 4 au, with a total mass of 5 $M_\oplus$. For a large number of bodies, using more symplectic levels can speed up the integration significantly. The comparison between different GPU types indicates that every GPU has a different optimal choice of parameters. For $N < 4096$ the choice of 1,1 is generally the fastest.

An important detail is that we still use the same critical radius definition (Equation (5)) as in the original algorithm. The only difference is that the time step is reduced in the higher levels, which leads to a smaller critical radius. In principle, the algorithm could also be used to reduce the Hill radius part of Equation (5), as the SyMBA method does (Duncan et al. 1998), but this is not done in the current version of GENGA. The reason is that, inside the Hill sphere of a body, the trajectory of a third object is no longer dominated by the central body, but is rather described by three-body dynamics. Therefore, using a Kepler drift about the central body to advance the orbit is not very effective. A more detailed description of the algorithm is given in Appendix A.

Figure 10 shows a performance comparison of different symplectic levels for an increasing number of planetesimals and for two different GPUs. The legend in the left panel indicates the number of levels and the number of sub-steps per level. The time is normalized to the original hybrid symplectic integrator (black dashed line). The initial conditions for the simulations are disks of N planetesimals with a total mass of 5 $M_\oplus$ orbiting the Sun between 0.5 and 4 au. The results show that using more symplectic levels can speed up the simulation, especially at high N. The results also show that there is not a unique best choice for all GPU types, and for every GPU an individual best choice must be found. However, both GPUs show that the symplectic levels only speed up the simulations when $N \gtrsim 4096$.

In practice, a good balance between the number of levels and the number of sub-steps must be found. Using a higher number of levels requires more memory to store all close-encounter pairs of each level, and using a higher number of sub-steps

15 The factors 1.0 and 0.1 are defined by the form of the changeover function in Chambers (1999, Equation (10)).

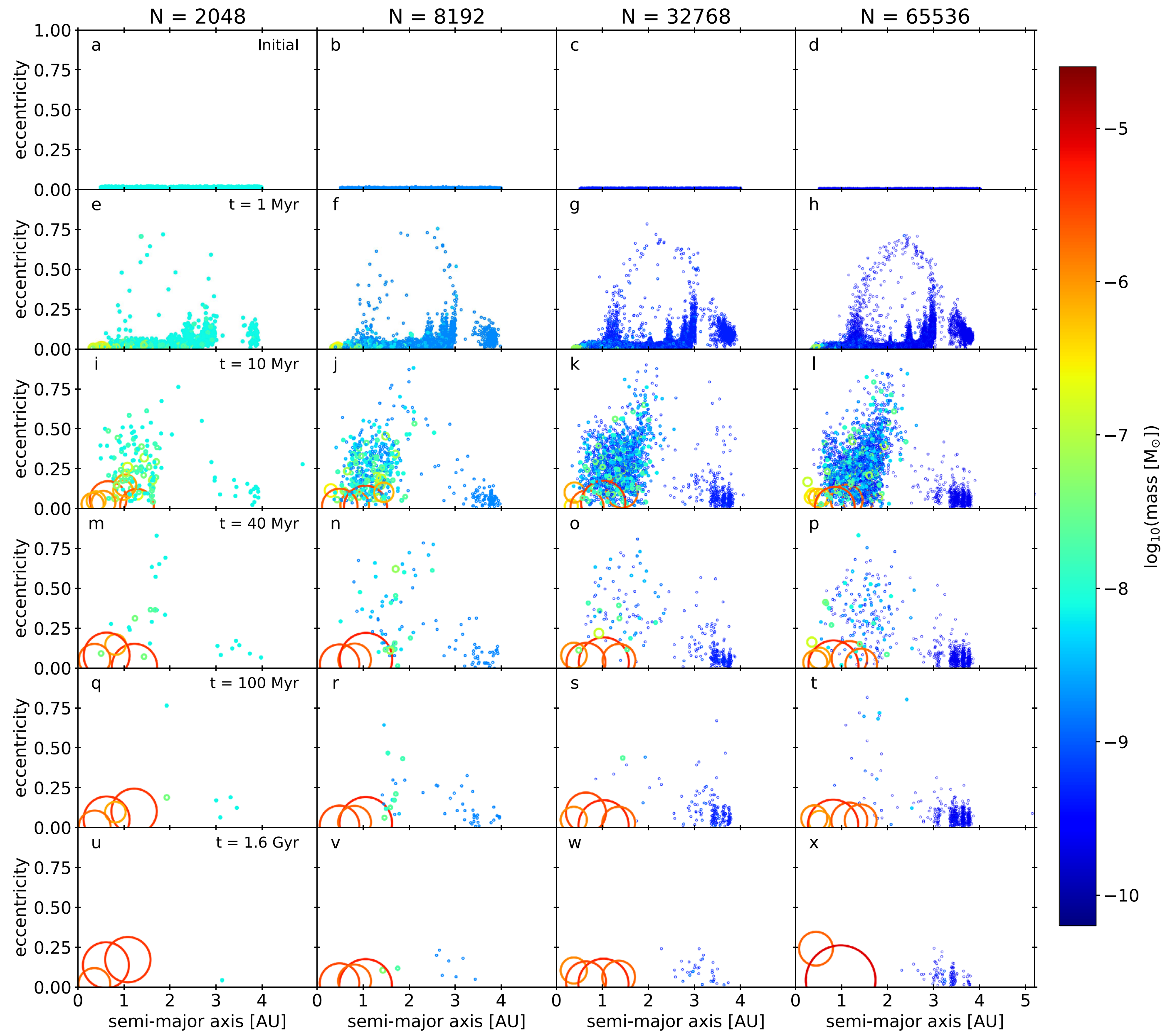

Figure 11. Terrestrial planet formation starting with N planetesimals with a total mass of 5 $M_\oplus$ and including a gas disk, Jupiter, and Saturn. Sweeping secular resonances from Jupiter and Saturn scatter planetesimals to the inner part of the disk, where terrestrial planets can form within 20–30 Myr. The final configuration of the formed planets is dependent on stochastic processes and can be different in each individual simulation. By including more and smaller planetesimals, the structure of an asteroid belt becomes visible. To achieve realistic masses in the asteroid belt, still much more and smaller particles must be used. The color and the size of the dots both represent the mass of the planetesimals. The gas giants are not shown in this Figure.

increases the amount of kernel calls needed and therefore can increase the total kernel overhead time. The fastest configuration will depend on the GPU device used. Therefore, we recommend testing different configurations before running long-term simulations in order to find the best option. In a future version of GENGA, this process could also be automated by a self-tuning routine at the beginning of the simulation.

6.4. Example I: High-resolution Terrestrial Planet Formation

We use the described higher order changeover mechanism to test the formation of the terrestrial planets via planetesimal collisions, with different particle resolutions. We use a disk of N equal-mass planetesimals, distributed between 0.5 and 4 au with a total mass of 5 $M_\oplus$, with the number of planetesimals varying between 2048 and 65,536. We include the gas giants Jupiter and Saturn on their current orbits and use a gas disk to simulate planetesimal migration. The sweeping secular resonances from the gas giants and the gas disk start to scatter the planetesimals between ∼2 and ∼3.5 au to the inner part of the system, where terrestrial planets can form within 20–30 Myr. After ∼100 Myr, the terrestrial planets have accreated most of the surrounding planetesimals but, in some cases, they can still collide with each other. Snapshots of the simulations are shown in Figure 11. The mechanism of the sweeping secular resonances and more details of the used gas disk are given in Woo et al. (2021). The final configuration of the formed terrestrial planets are affected by the chaotic nature of the N-body problem and therefore depend on

Table 3
Execution Time of Example I in Months on a Tesla P100 GPU[a]

N	$<10^6$ yr	10^6–10^7 yr	10^7–10^8 yr	10^8–10^9 yr
2048	0.02	0.14	0.36	1.47
4096	0.03	0.20	0.38	2.22
8192	0.10	0.31	0.48	2.14
16,384	0.65	0.68	0.57	2.86
32,768	0.90	1.22	0.88	3.07
65,536	2.76	2.60	1.30	3.78

Note.
[a] The execution time shown is the total run time of the simulations. The computer cluster used had a wall time limit of 24 hr for each job. When the wall time limit was reached, GENGA automatically saved the vectors at the final time step before it is terminated. Therefore, continuing a simulation on such a cluster takes only ∼5–30 s per restart. Restarting simulations periodically can improve the performance slightly because the self-tuning step is executed more often (see Section 8.3) and each segment of the simulation will run faster as a result.

stochastic processes (e.g., in which order floating point numbers are processed (see Appendix B.3)) so that the observed differences in the figure are not only caused by using different particle sizes. However one can see clearly that, by using an increased mass resolution, the structure of the asteroid belt becomes visible. In this test case, we include the gas giants already at the beginning of the simulation. Inserting them at a later time, or by using a growth track for their masses, would affect the location and structure of the remaining asteroid belt. Also, the initial distribution of the planetesimals strongly affects the final distribution of the asteroid belt. The execution time on a Tesla P100 GPU of this example is listed in Table 3.

In order to find the best configuration for the number of levels and the number of sub-steps per level, we run each configuration for 1000 time steps before the simulation starts, and choose the fastest option. We repeat this procedure periodically as the number of particles gets reduced.

6.5. Example II: High-resolution Planet Formation without Gas Giants

We repeat the planet formation simulations of the previous section without any gas giants and without a gas disk. Not including the gas giants in the system has a huge impact on the planetary formation process. The planetesimals are not exposed to large mean motion or secular resonances and therefore their eccentricities remain very low for a long time. This also means that the influence of a gas disk is less pronounced and has only little effect on the final configuration. This is the expected result because the formation process takes longer than the lifetime of the gas disk; we confirm this in test simulations. Therefore, we do not include the gas disk in this section and the simulation is only using pure Newtonian gravity. In Figure 12, we display the result of four simulations with a different number of planetesimals. The total amount of mass is always 5 $M_\oplus$. Without the gas giants, the planet formation process is much slower and growth is inside out. The formed planetary embryos or planets scatter the surrounding planetesimals to a larger semimajor axis and a more eccentric orbit. With this mechanism, planets can also be formed outside of the original planetesimal disk. The figure shows that smaller planetesimals are scattered outside faster than larger planetesimals, and that planet formation continues over a longer time. After 3.5 Gyr, planet formation is still not finished in the outer part of the disk. The execution time on an Tesla P100 GPU of this example is listed in Table 4. The execution time is larger than in the example I, because more planetesimals remain in the simulation.

7. Improved Collision and Encounter Options

In Section 6, we describe an improved method to handle a large number of close encounters numerically in an N-body integrator. We also describe updates to the collision handling between particles and new diagnostic tools for keeping track of physical close encounters.

7.1. Collision Handling

A collision between two bodies happens when their physical radii overlap ($r_{ij} < R_i + R_j$), and the two bodies are merged into a single body. The current version of GENGA treats collisions as perfectly inelastic mergers by conserving linear momentum. Physically, a part of the potential and kinetic energy is transformed into internal energy that will be dissipated as heat. GENGA keeps track of this internal energy, such that the overall energy of the system is conserved. Angular momentum is conserved by a transformation into the spin of the new body. The position and velocity of the new body are given by

$$\boldsymbol{x}_{\text{new}} = \frac{\boldsymbol{x}_i m_i + \boldsymbol{x}_j m_j}{m_i + m_j} \tag{64}$$

and

$$\boldsymbol{v}_{\text{new}} = \frac{\boldsymbol{v}_i m_i + \boldsymbol{v}_j m_j}{m_i + m_j}, \tag{65}$$

where i and j indicate the indices of the two colliding bodies. The spin $\boldsymbol{S}$ of the new body is calculated as

$$\boldsymbol{S}_{\text{new}} = \boldsymbol{S}_i + \boldsymbol{S}_j + \boldsymbol{L}_{ij}, \tag{66}$$

with

$$\boldsymbol{L}_{ij} = \frac{m_i m_j}{m_i + m_j}(\boldsymbol{r}_{ij} \times \boldsymbol{v}_{ij}) \tag{67}$$

being the orbital angular momentum. The new radius R is set by conserving the mass and by mixing the densities of the two bodies

$$R_{\text{new}} = (R_i^3 + R_j^3)^{1/3} \tag{68}$$

and the new mass is just the sum of the masses

$$m_{\text{new}} = m_i + m_j. \tag{69}$$

In order to keep track of the energy conservation, we add the lost kinetic and potential energy from collisions, ejections, and that caused by gas drag into a quantity U. This quantity includes the energy loss from all particles together and not just from single particles. The lost energy U is not directly related to a physical quantity, however it contains the change of the inner energy of all particles and the spin energy caused at collision.

When two particles i and j collide, then U is increased by

$$\Delta U = \frac{1}{2}\frac{m_i m_j}{m_i + m_j} v_{ij}^2 - G\frac{m_i m_j}{r_{ij}}. \tag{70}$$

The index of the new merged body is set according to

1. either the index of the more massive body or
2. if both bodies have an equal mass, then the lower index of the bodies i and j.

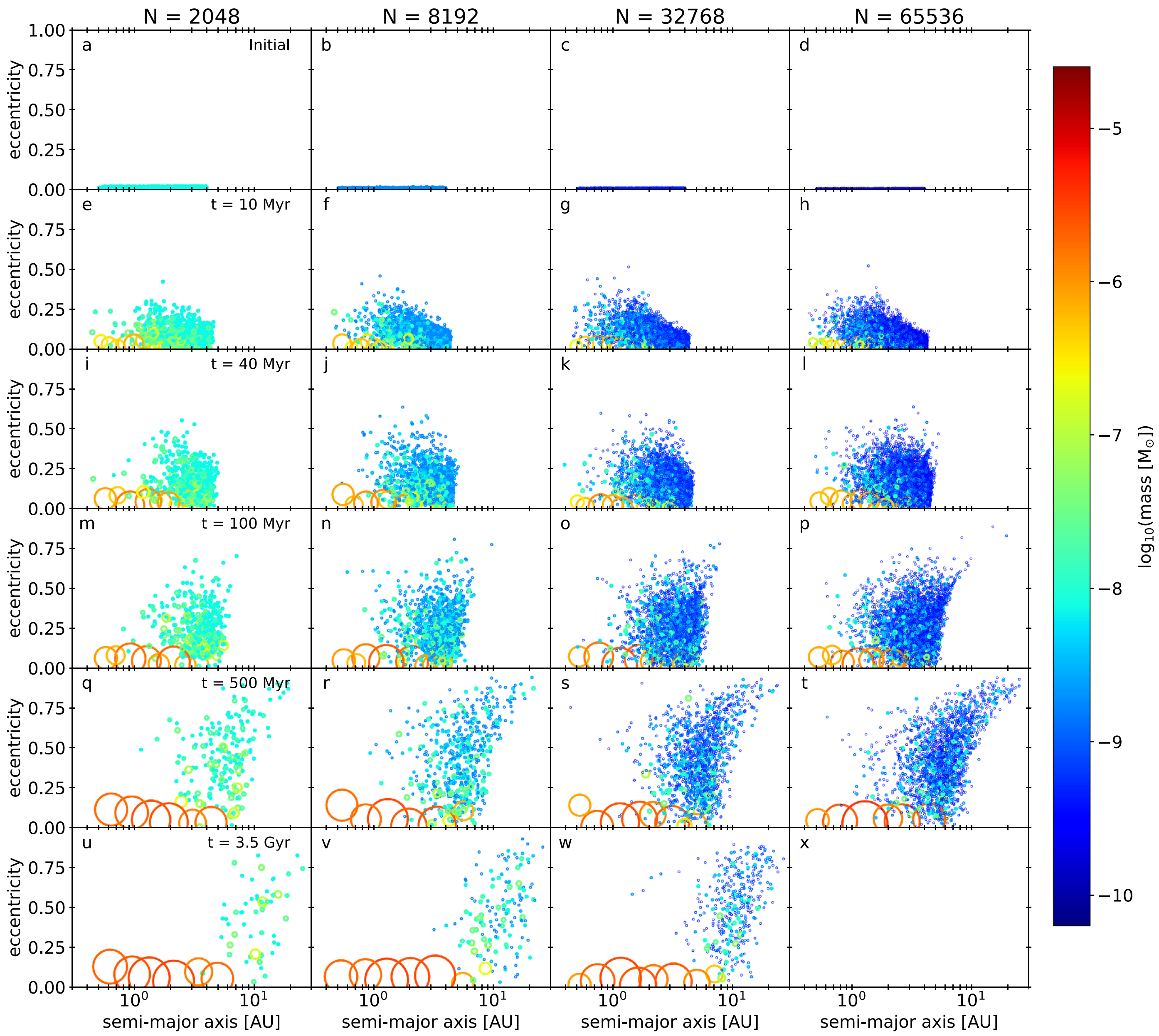

Figure 12. Planet formation starting with N planetesimals with a total mass of 5 $M_{\oplus}$ and without any gas giants, which significantly slows the planet formation process (Quintana & Lissauer 2014). Planetary embryos are first formed at the inner disk, where they scatter the planetesimals to a larger semimajor axis and more eccentric orbits. At the outer edge, planet formation still continues after more than 3 Gyr. The snapshots at 100 Myr (panels (m) to (p)) and 500 Myr (panels (q) to (t)) show that smaller planetesimals can be scattered away earlier than more massive planetesimals. The snapshot at 3.5 Gyr (panels (u) to (w)) indicate that, for smaller planetesimals, the planet formation process at the outer edge of the disk is not yet finished after 3.5 Gyr. Panel (x) is empty because this simulation reached only 500 Myr. The color and the size of the dots both represent the mass of the planetesimals.

Table 4
Execution Time of Example II in Months on a Tesla P100 GPU

N	$<10^6$ yr	10^6–10^7 yr	10^7–10^8 yr	10^8–10^9 yr
2048	0.02	0.12	0.55	2.14
4096	0.04	0.23	0.87	2.32
8192	0.09	0.45	1.54	3.57
16,384	0.22	0.91	2.68	5.04
32,768	0.68	2.32	4.93	7.82
65,536	2.41	7.22	9.51	…

7.2. Collision Precision

The collision process is resolved during the Bulirsch–Stoer direct integration of the bodies in a close encounter with discrete time steps. Therefore, a collision is generally not detected at the exact collision time, but rather when the two particles already overlap by a small amount. In some situations, it can be necessary to compute the collision time more precisely and to extract the coordinates of the two bodies at the exact collision time. GENGA offers the option to control the collision precision by a user parameter called “Collision Precision.” This parameter sets the tolerance of the collision detection and is set in units of a radius fraction, i.e.,

$$\frac{(R_i + R_j) - r_{ij}}{R_i + R_j} < \text{tolerance}. \qquad (71)$$

It is also possible to define if collisions should be reported with slightly overlapping positions, or if positions should be

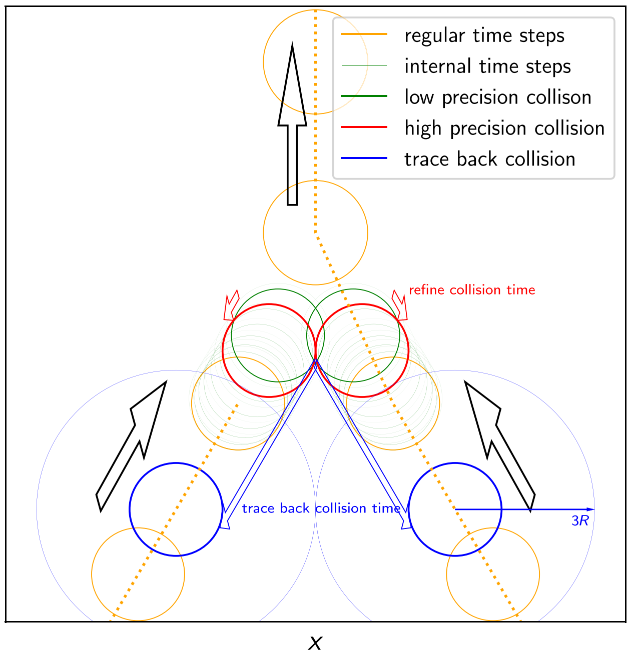

Figure 13. Example of two colliding bodies. The orange circles show the regular time steps of the integration, the thin green circles the internal reduced time steps during the Bulirsch–Stoer close-encounter method. The precision of the resolved collision time can be set by a user parameter. The green color indicates a low precision, the red color indicates a high precision. Collisions can also be traced back to a time before the actual collision happens, where the two bodies are still separated by an increased radius. In blue color is shown such a back-traced collision at a position where the distance between the two bodies corresponds to three times the sum of their radii.

reported shortly before the exact collision time. The latter option is needed when a bouncing collision model is implemented.

An example of a collision is shown in Figure 13. The orange circles indicate the positions and sizes of the bodies at the regular time steps. During a close-encounter phase, the time steps are refined within the Bulirsch–Stoer method, shown in thin green color. When a low precision is used, then the collision is reported at the time step when the two bodies start to overlap. When a high collision precision is used, then the contact point is refined until the tolerance is reached.

7.3. Trace Back Collisions

For certain applications, it is necessary to extract the coordinates of two colliding bodies at an earlier time, when the two particles are still separated by two or three times their radii. When this is the case, collisions are resolved with an external method, e.g., a model that resolves the inner structure of the bodies. GENGA has the option to back trace collisions to a time before the actual collision happens. In order to do that, we save the current time step where the collision is detected and apply a backward time step. During this backward time step, we increase the radii of the two affected bodies by the desired fraction, and we report the new detected collision positions. After that, GENGA can continue at the saved time step as usual. When multiple collisions happen at the same time step, we have to resolve each of them independently, because the increased radii of two bodies can affect the dynamics of other particles as well. An example of a back-traced collision is shown in Figure 13 in blue color.

7.4. Stop at Collision Time

When an external code is used to resolve collisions in more detail (e.g., Timpe et al. 2020), it is necessary to stop the entire simulation at the time of the first detected collision, or at the earliest back-traced collision time. It is important that all other particles, which are not involved in the collision or in a close encounter, are also integrated backward in time self consistently. GENGA supports this option to stop a simulation and creates a separated collision output file containing all bodies at the time of the first collision. This file can then be used to resolve the collision externally and to create a new initial condition file to continue the *N*-body integration with GENGA until the next collision is detected.

7.5. Report Close Encounters

GENGA supports the option to store the coordinates of all detected close encounters in a file. In this context, a close encounter is not defined as the usual numerical close encounter (Equation (5)), but rather when the separation r_{ij} between the bodies i and j is smaller than the sum of the increased physical radii:

$$r_{ij} < f\,(R_i + R_j),$$

where R_i and R_j are the physical radii of the two bodies, and the factor f a user parameter. An encounter event is reported at the moment of closest approach, which typically happens only once per orbit. In order to find all such close encounters, the critical radius r_{crit} (Equation (5)) is automatically increased if necessary. An example of an application where this option can be used is shown in Section 5.8.

7.6. Stop at Encounters

It is possible to stop a simulation automatically when an encounter between two bodies occurs. In this context, a close encounter is defined when the separation r_{ij} between the bodies i and j is smaller than the sum of the increased Hill radii:

$$r_{ij} < g(R_{\mathrm{H},i} + R_{\mathrm{H},j}),$$

where $R_{\mathrm{H},i}$ and $R_{\mathrm{H},j}$ are the Hill radii of the two bodies, and the factor g is a user parameter.

8. New Tools and Options in GENGA

In the following section, we describe additional improvements and updates of GENGA since Grimm & Stadel (2014). These updates are (1) a new semi-active test particle mode, (2) a function to use predefined masses, (3) radii or orbital elements tables, (4) a self-tuning performance optimization procedure and (5) the support for AMD GPUs. More tools and functions are listed in Appendix B, which includes a real-time visualization tool using openGL, a method to create exact reproducible results, an option for creating a-e and a-i grids of a simulation, and an option for a GPU buffer for outputs.

8.1. New Semi-active Test Particles Mode

We add a test particle mode for semi-active bodies. In this mode, test particles can interact with large bodies, but they do not interact with other test particles. This mode can be used when simulating small particles whose total gravitational potential influences the orbits of larger planets as used in

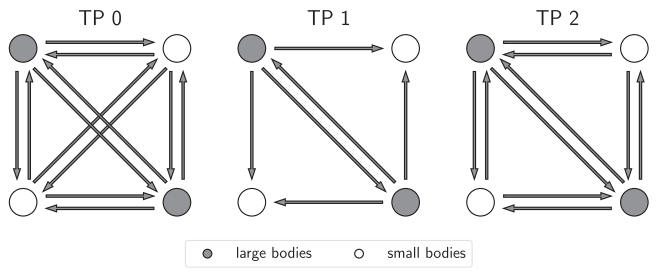

Figure 14. Schematic of the different test particle modes (TP). The left panel shows the full interactive mode, where each particle interacts with all the other particles. The middle panel shows the test particle mode 1, where only large particles interact with other particles. The right panel shows the test particle mode 2 for semi-active particles, where small particles do not interact with other small particles but with other large particles. The arrow in the Figure indicates all gravitational force terms that are calculated.

Quarles & Kaib (2019), for example. Since test particles in the semi-active mode can also influence large particles, the close-encounter groups must be calculated the same way as in the fully interactive mode. When simulating planet formation by merging small planetesimals, then the test particle mode should not be used, because it would prevent the collisions of planetesimals and therefore the formation of larger bodies through the merger of smaller ones. An exception could be made when the collisions of small particles are treated statistically. Test particles do not have to be massless, the user can specify a mass threshold for treating particles as test particles or as fully interactive particles.

The different test particles modes of GENGA are illustrated in Figure 14 and can be summarized as follows:

1. Fully interactive mode (TP 0).
   All bodies interact with all the other bodies. This is the default.
2. Test particles mode (TP 1).
   Test particles do not interact gravitationally with other particles, except when they collide with large bodies, then their momentum is added to the final momentum of the new body.
3. Semi-active test particle mode (TP 2).
   Test particles do not interact with other test particles. They interact with large bodies the same way as in the fully interactive mode.

The performance of the test particles modes is described in more detail in Section 9.1.

8.2. Usage of Predefined Coordinates, Mass, or Radius Tables

The set-elements option in GENGA permits users to modify the orbital parameters, mass, or radius of bodies, according to a predefined data file. This can, for example, be used to fix the position of a planet to a certain location and make chaotic evolution more or less reproduceable, or to use a pre-calculated mass–radius evolution table of a planet and let the planet grow with time. When Keplerian elements are provided, the code converts the Cartesian coordinates during the simulation into Keplerian elements, modifies them according to the table, and converts them back into Cartesian coordinates. At each time step, GENGA interpolates temporally between given values in the table by using a cubic interpolation scheme. An application using a mass and radius table is presented in Section 8.2.1.

8.2.1. Example: Planetesimal Accretion and Satellite Capture by Jupiter

In this example, we simulate the planetesimal accretion of a forming Jupiter-like planet (Pollack et al. 1996; Alibert et al. 2005). We use a pre-calculated table with the planet mass and planetesimal capture radius of the growing gas giant, depicted in the top panel of Figure 15. This data was obtained in a similar way as described in Mordasini et al. (2012), i.e., with a classical 1D giant planet formation model based on the core accretion paradigm akin to Pollack et al. (1996). The planetesimal capture radius is calculated as in Inaba & Ikoma (2003). This means that this model combines the solution of the planetary interior structure equations to get the gas accretion rate with the accretion of planetesimals based on a rate equation to calculate the core accretion rate (Pollack et al. 1996; Alibert et al. 2005). The growing gas giant is fixed at a semimajor axis of 5.2 au and reaches its final mass and radius after about 3.5 Myr. We embed the gas giant into a disk of 100,000 planetesimals, distributed uniformly between 2 and 8 au. The planetesimals are treated as massless test particles. At the beginning, the gas giant accretes nearby planetesimals and it starts to excite the eccentricities of planetesimals located in mean-motion resonances and those that get within a few Hill radii. When the excited planetesimals reach a distance of 1000 au to the Sun, they are removed from the simulation and reported as ejected particles. The middle panel of Figure 15 shows the time evolution of the semimajor axis and eccentricities of the planetesimals. The color in the middle panel indicates the original semimajor axis of the particles. After the gas giant has reached its final mass, it has cleared the planetesimal disk between 4.5 and 6.5 au, which roughly corresponds to the theoretical feeding zone. Since the mass of the gas giant increases with time, it is possible that some planetesimals are captured as satellites. In our application, we have observed 11 captured satellites from the inner edge of the feeding zone. Satellite captures from the outer edge of the feeding zone can also happen, but are more rare. The bottom panel of Figure 15 shows all observed satellite capture events together with all planetesimal accretion and ejection events. At the beginning of the simulation, when the mass of the gas giant is still small, it collects all surrounding planetesimals within the feeding zone. But while the planet grows in mass, it is not able to accrete more objects, but instead the planet starts to scatter them away until they leave the solar system or collide with other planets (Levison et al. 2010).

The simulation was run for four million years. The final number of accreted, ejected, and remaining planetesimals is 28,483 (28%), 4915 (5%), and 66,602 (67%), out of the initially 100,000 (100%). This simulation does not include the effect of gas drag on the planetesimals. However, tests indicate that the gas drag can be very important, and that the structure of the gas disk around Jupiter can change the number of accreted and ejected significantly. In this paper, this simulation mainly serves as an illustration of the possibility to use predefined mass and radius tables in GENGA. We note, however, that there is an ongoing discussion in the literature about the efficiency of planetesimal accretion by forming giant planets (see Podolak et al. 2020 and references therein).

It is also interesting to note that our observed satellite capture events could by applied to exoplanetary systems as well. For instance, two potential exomoon candidates are discussed (Kepler 1625b-i Teachey & Kipping 2018 and Kepler 1708b-i

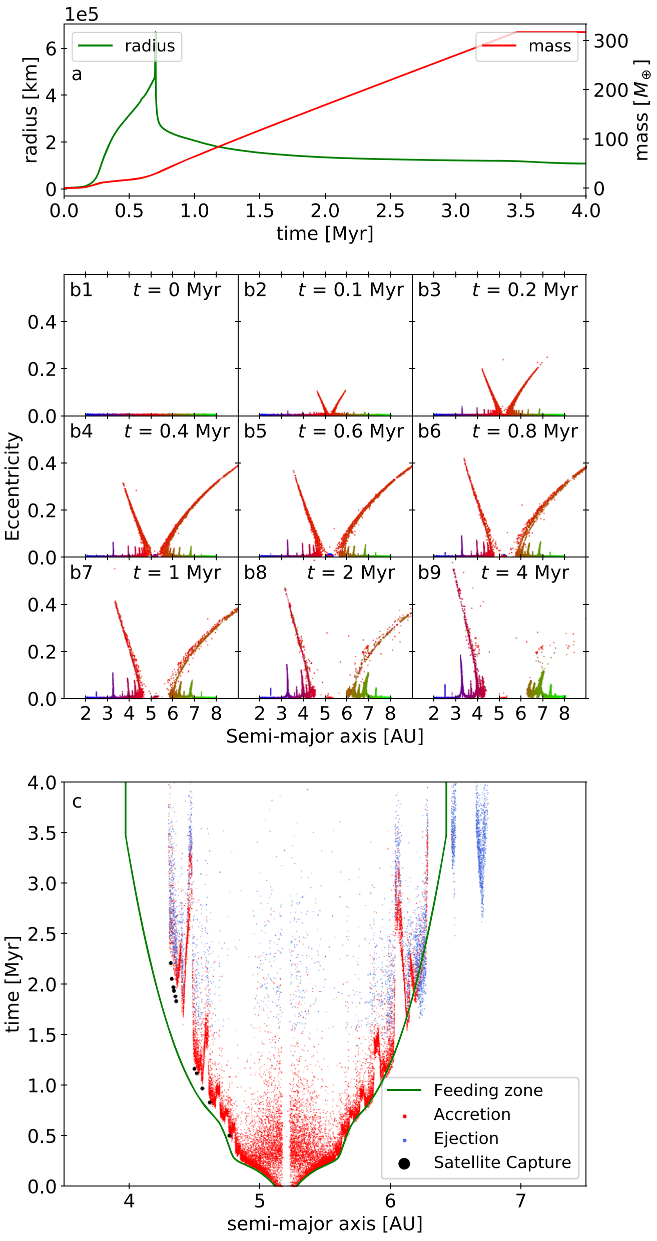

Figure 15. Example of a growing Jupiter simulation by using a predefined mass and radius table. The top panel shows the time evolution of the mass and planetesimal capture radius of a Jupiter-like planet. The middle panel shows the evolution of a planetesimal disk, distributed between 2 and 8 au, where the color indicates the original distance to the Sun. One can see how the gas giant removes all particles from the mean-motion resonance locations. The bottom panel shows the theoretical feeding zone of the giant planet in green color and all accretion and ejection events of the simulation. The x-axis indicates the original semimajor axis of the planetesimals at the beginning of the simulation. The black dots indicate satellite capture events, where some planetesimals evolve from a circumsolar orbit into a circumplanetary orbit.

Kipping et al. 2022), which could have formed in a similar scenario (Hamers & Portegies Zwart 2018).

8.3. Use Self-tuning Kernel Parameters

All GPU kernels need to be configured with a specific number of threads per thread block and a specific number of thread blocks, and the performance of the kernel can depend on this choice. Furthermore the best choice of kernel parameters can also depend on the initial conditions and the specific GPU. Therefore it is impossible to know beforehand which choice of kernel parameters leads to the best performance. To find the best choice ensuring the simulation runs as fast as possible, we implement a self-tuning routine in GENGA at the beginning of the integration. This routine tests different configurations and then adopts the best kernel parameters. The self-tuning routine can be disabled by a user parameter if needed but it is not recommended. By not using the self-tuning and using non-optimal kernel parameters, the performance penalty can be less than 1% in some cases or more than 100% in other cases, depending on the GPU type and the used initial conditions.

8.4. Support for AMD GPUs

In the last few years, AMD has developed an equivalent programming language to CUDA called HIP. Today, HIP supports most of the functionalities of CUDA, but not everything. An important difference to CUDA is that the warp size is not fixed to 32: it can be either 32 or 64. We updated GENGA such that it can run with different warp sizes, and we developed a translation tool to port GENGA from CUDA to HIP. We have successfully tested the HIP GENGA version on an AMD Radeon VII and an AMD Instinct MI100 GPU. While the results of simulations agree between the NVIDIA and the AMD GPUs, the performance of small simulations on AMD cards is not yet satisfying. The AMD GPUs seem to suffer much more from long kernel overhead time. This can probably be reduced by rearranging and combining different kernels into fewer kernel calls, but that is subject to future work.

9. Performance

We measure the performance of the code by integrating a planetesimal disk with N bodies, distributed between 0.5 and 4 au and a total mass of 5 $M_\oplus$. The number of close encounters increases with the number of bodies. For less than 256 bodies, no close encounters occur in this test. We use the best choice of the number of symplectic levels and sub-steps for each GPU type (see Section 6.3). We use an NVIDIA GTX 980, NVIDIA GTX 1080, NVIDIA RTX 3090, and an AMD Radeon VII GPU, all installed in desktop machines. We further test the NVIDIA Tesla K20, NVIDIA Tesla P100, NVIDIA Tesla A100, and an AMD Instinct MI100 installed on HPC server systems in Switzerland, Norway, Finland, and Slovenia. Finally, we also test an NVIDIA Quadro T2000 GPU from a notebook machine. While the Tesla, Instinct, and the Radeon VII cards support fast double-precision computing, typically half the speed of single precision, the other GPUs only have an artificially reduced double-precision performance, typically only 1.5%–4% (1/64 to 1/24) of the single precision performance.

The top panel of Figure 16 shows the time for 1000 time steps as a function of the number of bodies for different GPUs. Since the Tesla GPUs support fast double-precision calculations, they are significantly faster for high-N simulations. For low N, the RTX 3090 card can benefit from more CUDA cores and a high clock speed of close to 2 GHz. The AMD GPUs show a good performance for large simulations, but they suffer from much more kernel overhead time than the NVIDIA GPUs for small simulations. The bottom panel of Figure 16 shows the performance of the most important kernels, measured on the RTX 3090 GPU. For low N, the FG (Kepler drift) kernel dominates the execution time, and also the total kernel

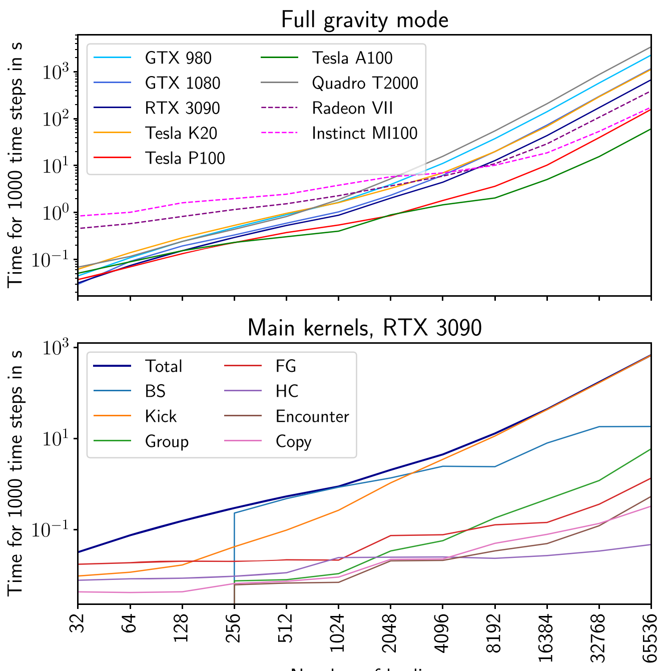

Figure 16. Performance of the full gravity mode, for a set of N massive bodies distributed in a disk between 0.5 and 4 au and a total mass of 5 $M_\oplus$. The GTX, RTX, and Radeon cards are installed in desktop machines, the Tesla and Instinct cards in a computer cluster and the Quadro card is integrated in a notebook machine. The GTX, RTX, Tesla, and Quadro cards are NVIDIA GPUS, the Radeon and Instinct cards are AMD GPUs. The bottom panel shows the performance of the main kernels: BS (Bulirsch–Stoer routine for close encounters), Kick (gravity calculation), group (close-encounter pairs grouping), FG (Kepler orbit solver), HC (Sun kick), encounter (close-encounter check), and copy (data transfer for close-encounter information). The difference between the shown kernels and the total time is caused mostly by the kernel overhead time.

overhead time and data transfer is important. For an intermediate number of bodies ($256 < N < 4096$), the Bulirsch–Stoer integration dominates the full integration, while for high N (>4096), the kick operation takes up most of the time. This result demonstrates clearly how the code can use different kernel optimizations, depending on the number of bodies. Note that the performance of the code also depends on the initial conditions, and especially how many close encounters occur.

9.1. Test Particles

To measure the performance of the test particle mode, we set up a particle disk between 0.5 and 4 au with zero inclination. The disk contains 16 massive particles which perturb the test particles as well as each other. This setup also leads to close encounters between test particles and the massive bodies. We test the performance of the test particle modes 1 and 2 on the same GPUs as the full self-gravity tests.

In Figure 17, we show the measured performance as a function of the particle number. For a high number of particles ($N > 16{,}384$), the Tesla P100 and Tesla A100 are clearly the fastest option, while for a low number of particles ($N < 1024$), the RTX 3090 is faster in this setup. The AMD GPUs show a good performance for high-N simulations, but they suffer from much more kernel overhead time than the NVIDIA GPUs for low-N simulations. In the bottom panel of Figure 17, we once again plot the performance of the main kernels. For a high

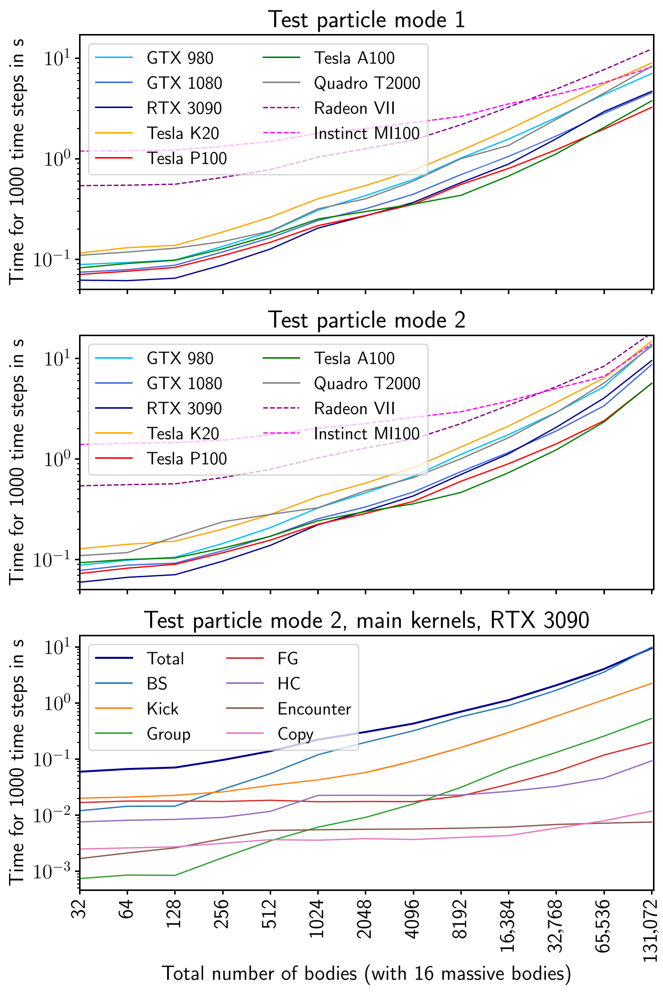

Figure 17. Performance of the test particle modes 1 and 2, for a setup with 16 massive bodies and (N-16) test particles, embedded in a disk between 0.5 and 4 au. The GTX, RTX and Radeon cards are installed in desktop machines, the Tesla and Instinct cards in a computer cluster and the Quadro card is integrated in a notebook machine. The GTX, RTX, Tesla, and Quadro cards are NVIDIA GPUS, the Radeon and Instinct cards are AMD GPUs. The bottom panel shows the performance of the main kernels: BS (Bulirsch–Stoer routine for close encounters), Kick (gravity calculation), group (close-encounter pairs grouping), FG (Kepler orbit solver), HC (Sun kick), encounter (close-encounter check), and copy (data transfer for close-encounter information). The difference between the shown kernels and the total time is caused mostly by the kernel overhead time.

number of particles, the Bulirsch–Stoer routine from the close-encounter phase clearly dominates the run time, followed by the kick operation. For a small number of particles, the kick operation and the FG routine for the Keplerian orbit calculation are comparable in run time. Also the total kernel overhead time is important for a low number of particles.

10. Discussion and Conclusions

We updated the GPU N-body code GENGA from its first invocation (Grimm & Stadel 2014) to be able to integrate high-resolution planet formation simulations by introducing a hierarchical close-encounter method with multiple changeover functions. We improved the performance of individual kernels by optimizing the parallelization and memory usage, and by adding a kernel self-tuning mechanism that finds the best kernel parameters for different GPU types. We added several non-Newtonian forces and a model for small-body collisions for meteorite dynamics. We included the option of using semi-active test particles, different options of setting the collision precision handling and added other tools like a real-time visualization with openGL.

With the described updates and optimizations, the performance of the new GENGA version can be up to two orders of magnitudes faster than the original methods reported in Grimm & Stadel (2014), depending on the initial conditions used. In the example simulations, we presented a planet formation simulation with a gas disk that ran longer than 1.5 billion years. And we presented a planet formation simulation without gas giants performed over 3.5 billion years that shows how planets are also formed outside of the initial planetesimal distribution.

In this paper, we did not discuss the gas-disk model and the multi-simulation mode of GENGA as well as its built-in TTV tool used in Grimm et al. (2018) for estimating the masses of the TRAPPIST-1 exoplanets. These topics will be improved and reported on in the future, as well as new tools or forces that will be be added to the present version of GENGA.

With the presented updates, it is now possible to run simulations with more than 30,000 fully interactive planetesimals in a feasible time. The added forces allow us to study the dynamics of small bodies in the solar system, or to study the formation and subsequent evolution of compact exoplanetary systems where tidal forces play an important role. It is also important that the field of high-performance computing and GPU computing evolves quickly and that simulation software must be constantly adapted and evolved to be able to use the most recent hardware efficiently. With this work, we aimed to provide a fast and user-friendly code for the study of terrestrial planet formation and evolution of the solar system and other exoplanetary systems.

Calculations were performed on UBELIX (http://www.id.unibe.ch/hpc), the HPC cluster at the University of Bern.

This work has been carried out within the framework of the NCCR PlanetS supported by the Swiss National Science Foundation. C.M. acknowledges the support from the Swiss National Science Foundation under grant 200021_204847 “PlanetsInTime”.

We thank Jørgen Nordmoen for running performance tests on the NVIDIA Tesla A100 and the AMD Instinct MI100 GPUs at the University of Oslo.

Appendix A
Multi-level Close-encounter Algorithm

The basic hybrid symplectic integrator can be written with the following scheme:

```
Rcrit(dt)
Kick (dt/2), close encounter pre-check
HC(dt/2)
FG(dt)
if(close encounter candidates):
  close encounter detection
  if(close encounters):
   grouping
   Bulirsch--Stoer(dt)
HC(dt/2)
Kick(dt/2)
```

where `Kick` corresponds to Equation (3), i.e., the interaction part of the Hamiltonian system, `HC` to Equation (4), i.e., the democratic momentum summation, and `FG` to Equation (2), i.e., the Kepler part of the Hamiltonian system. The parameter `dt` is the time step. For more details on these routines we refer to Grimm & Stadel (2014).

To include the additional symplectic levels into the integrator, we replace the Bulirsch–Stoer call with a recursive function `SEnc` (Symplectic Encounters) and use the parameter `SLevels` to indicate the number of symplectic levels. We have

```
Rcrit(dt)
Kick(dt/2), close encounter pre-check
HC(dt/2)
FG(dt)
if(close encounter candidates):
close encounter detection
if(SLevels > 1):
if(close encounters):
SEnc(dt, 1.0, 0)
else:
if(close encounters):
grouping
Bulirsch--Stoer(dt)
HC(dt/2)
Kick(dt/2)
```

And the function `SEnc` consists of:

```
SEnc(dt, ds, SLevel):
  groupS
  SLevel+ = 1
  ds *= substeps
  for(int s = 0; s < substeps; s++):
   RcritS()
   KickS(dt/ds / 2 )
   FGS(dt/ds)
   close encounter detection
   if(close encounters):
    if(SLevel < SLevels -1):
    SEnc(dt, ds, SLevel)
   else:    grouping
    Bulirsch--Stoer(dt/ds)
  KickS(dt/ds / 2 )
```

where `substeps` is the number of sub-steps for each level, and $1/ds$ is the reduction factor of the time step. The `groupS` function extracts all relevant close-encounter pairs from the upper symplectic level. It uses a scan function to perform a parallel stream compaction operation to generate the new close-encounter lists. The functions `RcritS`, `KickS`, and `FGS` are similar to the original functions, but are applied only to the relevant bodies in each symplectic level.

Appendix B
Additional New Tools in GENGA

In this appendix, we list new tools in GENGA, which are not described in the main text.

B.1. Real-time Visualization with OpenGL

The Nvidia CUDA drivers offer the option to use openGL to visualize data on the GPU directly on the screen. Since no data transfer from the GPU to the CPU is needed, the visualization can be done very efficiently. We provide such an openGL interface for GENGA where a simulation can be projected in

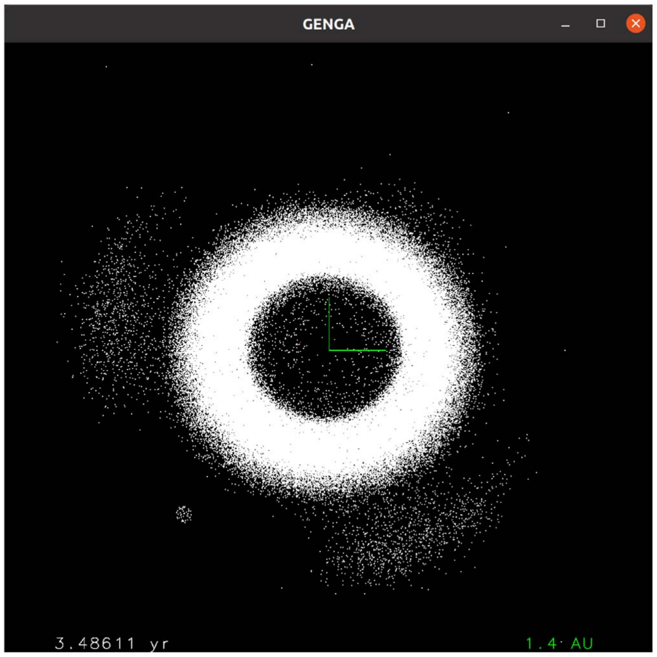

Figure 18. Screenshot of the GENGA real-time visualization tool using openGL illustrating the inner part of the solar system, including Jupiter and the asteroid belt. The Trojans are nicely visible on the Lagrange points of Jupiter.

real time to the screen. Basic user interactions are possible, like zooming into the simulation or moving and rotating the viewing position. Having the possibility to experience the dynamics of a simulation in real time can help to build up a deeper intuition on the physics. Furthermore, short time dynamical effects such as satellite capture events can be observed in a simple and intuitive way. The only requirement to use the openGL interoperability is that the GPU must be connected to a monitor. A screenshot of the real-time visualization is shown in Figure 18.

B.2. Use Calendar File for Irregular Output Times

When outputs are needed at irregular times, then GENGA offers an option to specify a calendar file, containing all desired output times. When the integration has reached an irregular time step, then the time step is reduced to the desired length and the output data is produced. After that, the reduced time step is reverted to get back to the last regular time step and the integration is continued as usual. In this way, no additional energy error is added.

B.3. Exact Reproducible Results

Usually, the outcome of a parallel numerical calculation is not exactly reproducible. The reason is that calculations are done with a finite precision and that the outcome depends on the order of the operations. In general, we have $a + b + c \neq a + c + b$. Often the planetary N-body problem is chaotic, and therefore tiny differences in the calculation can lead to a different result on individual bodies. Even the number of formed planets can vary from simulation to simulation. A detailed study about this effect on GPUs is given in Hoffmann et al. (2017).

However it is possible to force GENGA to exactly reproduce a given outcome. This is possible because on the GPU, summations of arrays are performed with parallel reduction sums. These parallel reduction sums have a well-defined order in which the terms are calculated, and the result is therefore always reproducible if the structure of the parallel reduction sum remains the same. The structure can, however, change by using different kernel parameters (see Section 8.3), or by using a different GPU type.

The only place that has not a fixed order in GENGA is the creation of the close-encounter pair lists. By introducing an additional sorting step on the close-encounter pairs lists, this order can be fixed and the result of a given initial condition is always the same. It is important to note that this does not mean that the results are "true"; they still suffer from small round-off errors, just that this error is now always the same as before. Since the additional sorting step also introduces a performance penalty, it is not recommended to use this mode of GENGA for production runs. It is mostly useful to check if a GPU works correctly and it helps to eliminate memory leaks in a code. The reproducible option is only supposed to work on the same type of GPU, using different GPU types can still change the order of operations in the summation parts.

To enable the exact reproducible outcome mode in GENGA, the "Serial Grouping" option must be used.

B.4. The a-e and a-i Grid Option

The a-e and a-e grids are two-dimensional histograms. They count the number of particles in a given time span for each cell in a "semimajor axis versus eccentricity" or a "semimajor axis versus inclination" grid. The grids are calculated on the fly directly on the GPU, and they are particularly useful to visualize short time dynamics of a system without having to write a lot of output files. The grids are updated during the FG step. Since multiple bodies in parallel could contribute to the same a-e or a-i cell count, we use an atomicAdd[16] operation to make sure that every body is counted correctly. This usage of the CUDA atomicAdd functions could potentially slow down the simulation when a lot of particles contribute to the same cell, but they also reduce the needed amount of memory. The a-e and a-i grids are stored in the GPU memory. Only at given intervals, the data is transferred back to the CPU and written to a file. In Figure 19 is shown an example of the a-e grid. It can help to visualize the dynamics of a system, and makes short time movements of the particles visible.

B.5. GPU Output Buffer

When the state of a simulation needs to be written to a file, then first the data of all the bodies needs to be transferred back from the GPU to the CPU via the PCI-Express bus on the motherboard. When this data transfer is requested very frequently, then it can create a bottleneck of the entire run time. In order to increase the data transfer rate, we implement a GPU buffer that stores the output data temporarily on the GPU and then moves a bigger amount of data together, which can improve the performance. The size of this buffer can be set by the user in the GENGA parameter file. The GPU output buffer is especially useful in the multi-simulation mode when outputs are written at an interval of less than 10 time steps.

[16] A CUDA atomic function performs its operation without interference from other threads and is therefore thread safe.

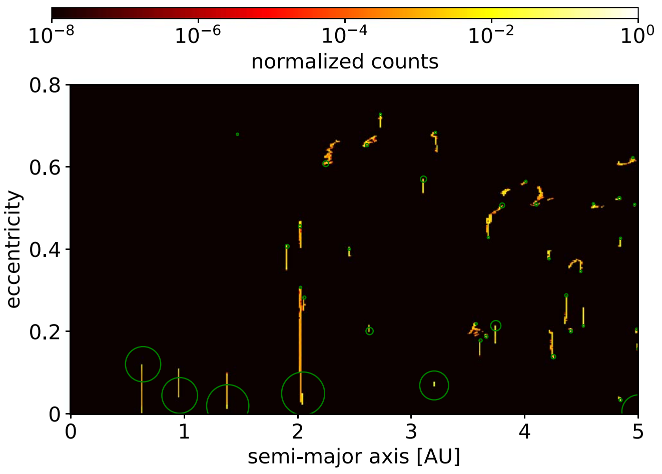

Figure 19. The a-e grid as an example of planet formation. The green circles show a snapshot of the simulation where the semimajor axis and eccentricity represent a single moment in time. The size of the circles corresponds to the masses of the bodies. The color map on top of the green circles shows the a-e grid, where the positions of the bodies of all time steps of a given interval are included. The a-e grid is a two-dimensional histogram and allows to visualize short scale dynamics without using a lot of additional memory.

ORCID iDs

Simon L. Grimm https://orcid.org/0000-0002-0632-4407
Joachim G. Stadel https://orcid.org/0000-0001-7565-8622
Ramon Brasser https://orcid.org/0000-0001-6331-2165
Matthias M. M. Meier https://orcid.org/0000-0002-7179-4173
Christoph Mordasini https://orcid.org/0000-0002-1013-2811

References

Alibert, Y., Mordasini, C., Benz, W., & Winisdoerffer, C. 2005, A&A, 434, 343
Auclair-Desrotour, P., Le Poncin-Lafitte, C., & Mathis, S. 2014, A&A, 561, L7
Barnes, R. 2017, CeMDA, 129, 509
Beech, M., & Brown, P. 2000, P&SS, 48, 925
Blanco-Cuaresma, S., & Bolmont, E. 2017, in EWASS 4, Star-planet interactions (EWASS-SS4-2017)
Bolmont, E., Demory, B. O., Blanco-Cuaresma, S., et al. 2020, A&A, 635, A117
Bolmont, E., Raymond, S. N., Leconte, J., Hersant, F., & Correia, A. C. M. 2015, A&A, 583, A116
Bottke, W. F., Rubincam, D. P., & Burns, J. A. 2000, Icar, 145, 301
Bottke, W. F., Vokrouhlický, D., Rubincam, D. P., & Nesvorný, D. 2006, AREPS, 34, 157
Brož, M. 2006, PhD Thesis, Charles Univ. https://sirrah.troja.mff.cuni.cz/~mira/mp/phdth/
Burns, J. A., Lamy, P. L., & Soter, S. 1979, Icar, 40, 1
Burns, J. A., Lamy, P. L., & Soter, S. 2014, Icar, 232, 263
Chambers, J. E. 1999, MNRAS, 304, 793
Clement, M. S., Kaib, N. A., & Chambers, J. E. 2020, AJ, 1, 18
Correia, A. C. M. 2009, ApJL, 704, L1
Correia, A. C. M., Laskar, J., Farago, F., & Boué, G. 2011, CeMDA, 111, 105
Duncan, M. J., Levison, H. F., & Lee, M. H. 1998, AJ, 116, 2067
Efroimsky, M., & Lainey, V. 2007, JGRE, 112, E12003
Fabrycky, D. C. 2010, arXiv:1006.3834
Farinella, P., Vokrouhlický, D., & Hartmann, W. K. 1998, Icar, 132, 378
Gaffey, M. J., Burbine, T. H., & Binzel, R. P. 1993, Metic, 28, 161
Greenberg, A. H., Margot, J.-L., Verma, A. K., et al. 2017, AJ, 153, 108
Grimm, S. L., Demory, B.-O., Gillon, M., et al. 2018, A&A, 613, A68
Grimm, S. L., & Stadel, J. G. 2014, ApJ, 796, 23
Hairer, E., Nørsett, S. P., & Wanner, G. 1993, Solving Ordinary Differential Equations I: Nonstiff Problems (2nd Revised. ed.; Berlin: Springer)
Hamers, A. S., & Portegies Zwart, S. F. 2018, ApJL, 869, L27
Hoffmann, V., Grimm, S. L., Moore, B., & Stadel, J. 2017, MNRAS, 465, 2170
Hut, P. 1981, A&A, 99, 126
Inaba, S., & Ikoma, M. 2003, A&A, 410, 711
Kidder, L. E. 1995, PhRvD, 52, 821
Kipping, D., Bryson, S., Burke, C., et al. 2022, NatAs, 6, 367
Kortenkamp, S. J. 2013, Icar, 226, 1550
Leconte, J., Chabrier, G., Baraffe, I., & Levrard, B. 2010, A&A, 516, A64
Levison, H. F., Thommes, E., & Duncan, M. J. 2010, AJ, 139, 1297
Makino, J., & Aarseth, S. J. 1992, PASJ, 44, 141
Mardling, R. A., & Lin, D. N. C. 2002, ApJ, 573, 829
Moore, A., & Quillen, A. 2010, NewA, 16, 445
Mordasini, C., Alibert, Y., Klahr, H., & Henning, T. 2012, A&A, 547, A111
Moyer, T. 1971, Mathematical Formulation of the Double-Precision-Orbit-Determination-Program (DPODP), JPL Technical Report, JPL-TR-32-1527, Jet Propulsion Laboratory, California Institute of Technology
Moyer, T. D. 2003, Formulation for Observed and Computed Values of Deep Space Network Data Types for Navigation (New York, NY: Wiley-Interscience)
Nakamura, A., & Fujiwara, A. 1991, Icar, 92, 132
Öpik, E. J. 1951, PRIAA, 54, 165
Podolak, M., Haghighipour, N., Bodenheimer, P., Helled, R., & Podolak, E. 2020, ApJ, 899, 45
Pollack, J. B., Hubickyj, O., Bodenheimer, P., et al. 1996, Icar, 124, 62
Portegies Zwart, S. 2020, NatAs, 4, 819
Potter, D., Stadel, J., & Teyssier, R. 2017, ComAC, 4, 2
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 2007, Numerical Recipes 3rd Edition: The Art of Scientific Computing (3rd ed.; Cambridge: Cambridge Univ. Press)
Quarles, B., & Kaib, N. 2019, AJ, 157, 67
Quintana, E. V., & Lissauer, J. J. 2014, ApJ, 786, 33
Rein, H., & Liu, S. F. 2012, A&A, 537, A128
Robertson, H. P. 1937, MNRAS, 97, 423
Rubincam, D. P. 2000, Icar, 148, 2
Saha, P., Stadel, J., & Tremaine, S. 1997, AJ, 114, 409
Saha, P., & Tremaine, S. 1994, AJ, 108, 1962
Shapiro, I. I., Smith, W. B., Ash, M. E., & Herrick, S. 1971, AJ, 76, 588
Sitarski, G. 1983, AcA, 33, 295
Sitarski, G. 1992, AJ, 104, 1226
Stoer, J., & Bulirsch, R. 2002, Introduction to Numerical Analysis (3rd ed.; New York, NY: Springer),
Teachey, A., & Kipping, D. M. 2018, SciA, 4, eaav1784
Timpe, M., Han Veiga, M., Knabenhans, M., Stadel, J., & Marelli, S. 2020, ComAC, 7, 2
Vokrouhlicky, D. 1998, A&A, 338, 353
Vokrouhlický, D., Bottke, W. F., Chesley, S. R., Scheeres, D. J., & Statler, T. S. 2015, in The Yarkovsky and YORP Effects, ed. P. Michel, F. E. DeMeo, & W. F. Bottke (Tucson, AZ: Univ. Arizona Press), 509
Vokrouhlický, D., & Farinella, P. 1998, AJ, 116, 2032
Vokrouhlický, D., & Farinella, P. 1999, AJ, 118, 3049
Vokrouhlický, D., Milani, A., & Chesley, S. R. 2000, Icar, 148, 118
Wiegert, P. A. 2015, Icar, 252, 22
Wisdom, J., & Holman, M. 1991, AJ, 102, 1528
Woo, J. M. Y., Grimm, S., Brasser, R., & Stadel, J. 2021, Icar, 359, 114305
Zhang, K., & Gladman, B. J. 2022, NewA, 90, 101659