ISIMA 2011 Project Report

Planetary Dynamics in Collisional Particle Disks

Zhao Sun[1], Eugene Chiang[2], Yoram Lithwick[3]

[1] Purple Mountain Observatory, Chinese Academy of Sciences, China

[2] University of California, Berkeley, USA

[3] Northwestern University, USA

Abstract

A simple, fast algorithm which simulates collisions between inelastic particles in an optically thin disk orbiting a central mass is implemented to the N-body simulation code MERCURY. The hybrid symplectic integrator is used to simulate a moonlet in the Saturn ring scenario, and produced a propeller structure around the moonlet which opens a partial gap in the ring.

1. Introduction

Collisional particle disks evolvements in the presence of planets is an interesting subject and there are many questions related to it. For example, how do Type I/II migration rates vary with planet eccentricity? How do gap widths and lengths vary with planet mass? For planets that do not open gaps or open only partial gaps, what torques are exerted by co-orbital disk material on the planet? The last two questions have important implications on Type III migration and on Saturn rings.

In Saturn’s rings, small moonlets produce partial gaps, as physical collisions and gravitational interactions between ring particles diffuse particles back into the gap downstream of the moonlet. Because of Keplerian shear, density perturbations excited at a moonlet’s position drift toward greater longitudes inside the moonlet’s orbit and toward lesser longitudes outside, forming a pair of features dubbed a “propeller” for its S-like shape. One of the propellers orbital longitude deviated from a strictly Keplerian solution (Tiscareno et al. 2010), which implies semimajor axis variations of the underlying moonlet. There are several explanations, Crida et al.

(2010) and Rein & Papaloizou (2010) investigated semimajor axis variations due to stochastic torques exerted by self-gravitating wakes of ring particles (Salo 1995). Tiscareno et al. (2010) suggested that the moonlet may librates within a resonance established by another larger moon. Pan & Chiang (2010) proposed that propellers like it are participating in a new type of co-orbital resonance with nearby ring material. They interpreted the non-Keplerian motions of propeller moonlets as back reactions of their perturbed disks on the moonlets, however, they have not considered how the motions of the moonlet feed back into shaping the gap. To investigate this issue further, we would like to use a more realistic model which simulates the dynamics of the system including all the interactions between the moonlet and the disk particles.

In this work, we imply the collision algorithm developed by Lithwick & Chiang (2007), hereafter L07, to the commonly used N-body simulation code MERCURY (Chambers & Migliorini 1997). The algorithm simulates collisions between inelastic particles in an optically thin disk orbiting a central mass. It is simple to implement and adds negligible running time to existing N-body codes. In L07 they used SWIFT subroutine package (Levison & Duncan 1994) to integrate the gravitational equations for the motions of the Sun and massless test particles with the Wisdom-Holman mapping method (Wisdom & Holman 1991). However, they test the algorithm without planet, and SWIFT can only handle massless test particles in the disk which can only feel the gravitational influence of the massive bodies but do not affect each other or the massive bodies. In this work, we use hybrid symplectic integrator (Chambers 1999) of the MERCURY package with small bodies as disk particles which can have mass and gravitationally affect and are affected by the massive bodies such as moonlets.

2. The collision algorithm

All bodies are evolved in two dimensions: out-of-plane velocities and coordinates are always identically zero. The subroutine simulates collisions between test particles in a disk with vertical optical depth

$$\tau \sim N_{\mathrm{tp}} \frac{s^2}{\bar{r}\Delta} < 1, \tag{1}$$

where $N_{\rm tp}$ is the number of test particles, s is their size, and $\bar{r}$ and Δ are are, respectively, the mean orbital radius and the radial width of the annulus that the particles occupy. In collisional particle disks, collisions tend to isotropize the velocity distribution. The collision time is $t_{\rm col} \sim 1/\left(n_v s^2 u\right)$, where u is the one-dimensional random speed and n_v is the volumetric number density, which is related to τ via $\tau \sim n_v s^2 u t_{\rm orb}$. Therefore

$$t_{\rm col} \sim \frac{t_{\rm orb}}{\tau}. \qquad (2)$$

The collision time is longer than the orbital time by the *u*-independent factor $1/\tau$.

We capture this behavior with two-dimensional simulations in which all bodies have zero inclination. For every time step *dt*, a two-dimensional square grid is built, with each grid element having dimensions $s_{\rm grid} \times s_{\rm grid}$; here, $s_{\rm grid}$ can be thought of as the size of a particle. If two test particles fall in the same grid cell, and if their relative speed is negative (i.e., if they are approaching each other), then they collide with each other with probability $P_{\rm col} = dt/t_{\rm orb} \ll 1$, where $t_{\rm orb}$ is the orbital time at the collision point. A random number generator is used to determine whether or not they actually collide.

To see that this algorithm gives the same collision time as equation (2) (where τ is given by eq. [1] with $s = s_{\rm grid}$), it is instructive to consider first a simpler algorithm that also yields the correct collision time. In this simpler algorithm, one waits for a time interval of $t_{\rm orb}$ (instead of *dt*) before finding which particles fall in the same grid cell. Then two particles that do fall in the same grid cell, and have converging velocities, collide with probability $P_{\rm col} = 1$. Since the probability that a given particle lies in a cell occupied by a second particle is τ, the collision time is $t_{\rm orb}/\tau$, as required. Turning now to the algorithm that we actually use, since we apply this algorithm every time interval *dt* (and not $t_{\rm orb}$), we must correspondingly reduce the probability of a collision by $dt/t_{\rm orb}$ in order to maintain the collision time at the value given by equation (2).

Although carried out in only two dimensions, we emphasize that our algorithm models three-dimensional disks in which collisions maintain inclinations that are comparable to the random eccentricities. A truly two-dimensional disk is not realistic, because collisions invariably generate out-of-plane velocities. But if one could somehow prevent the generation of out-of-plane velocities, the collision time in such a disk would be $\sim s/(u\tau)$, which differs from equation (2) by the factor $s/ut_{\rm orb}$. Since our algorithm satisfies equation (2), it does not model truly two-dimensional disks.

As will be shown below, collisions drive the random speed of the particles to $u \geq s_{\rm grid}/t_{\rm orb}$. Hence, if two particles fall in the same grid cell at one time, they will usually fall in separate grid cells after one orbital period. Since collisions can potentially occur every time step, it might be thought that the same two particles can collide many times in succession—a behavior that we consider undesirable. But this behavior is avoided by the requirement that particles must be approaching each other for a collision to occur; immediately after they collide, their relative velocities reverse signs, and they are no longer candidates for a collision pair.

One of the main advantages of our algorithm is that the time step is not restricted by the Courant condition. In a naive bruteforce algorithm, one must restrict $dt \ll s/u$ in order to ensure that any two particles that fall within a distance s of each other collide. This restriction on *dt* can be very cumbersome when $u \gg s/t_{\rm orb}$, as it will be whenever planets stir up the eccentricities. We avoid the Courant condition by treating the vertical dimension statistically: when two particles fall within the same two-dimensional grid cell, they need only collide a small fraction of the time because their vertical positions will, in general, differ. With our algorithm, we may choose *dt* to be as large as is allowed by the integrator, which is typically a significant fraction of the orbital time.

If two particles have been selected for a collision, i.e., if they lie in the same grid cell, are approaching each other, and are selected by the random number generator, then their velocities are updated as though the bodies were frictionless spheres whose surfaces touch (e.g., Trulsen 1971): the component of the relative velocity vector that lies parallel to the axis connecting the two particles is reversed in sign (from a converging velocity to a diverging one) and multiplied by the coefficient of restitution ε, i.e., in obvious notation,

$$u'_{{\rm rel},\|} = -\varepsilon u_{{\rm rel},\|}. \tag{3}$$

Neither the perpendicular component of the relative velocity vector, nor the velocity of the center of mass of the two colliders, nor the positions of the colliders are changed by the collision. A collision does not alter the sum of the angular momenta of both colliding bodies; hence the collision algorithm exactly conserves total angular momentum. Note that a collision between two particles separated by distance *d* changes the velocities of the particles as though each were a smooth sphere with radius *d*/2. Since *d* changes from collision to collision, the particles' sizes are effectively changing; they are only approximately $s_{\rm grid}$.

The algorithm has now been completely described, aside from how the code finds which pairs of particles lie in the same grid cell. To find colliding pairs, the code first determines in which grid cell each particle lies. A grid cell is labeled by two integers representing its location along the *x*- and *y*-axes. Second, the code sorts the grid cells that contain test particles with the heapsort algorithm (Press et al. 1992). The sorted occupied grid cells are then checked to see if the same grid cell is repeated for two different particles. The step that takes the most time in the entire collision algorithm is the heapsort, which requires $\sim N_{\rm tp} \ln N_{\rm tp}$ operations.

A ring diffuses in the time that it takes a particle to random walk across its width. This random walk has a step-size equal to the epicyclic excursion of a particle ($\sim s_{\rm grid}$) and a time per random step of $t_{\rm col}$. Thus, to diffuse the width of the ring Δ takes a time

$$t_{diff} \sim t_{col} \left(\frac{\Delta}{s_{grid}} \right)^2 \gg t_{col}, \tag{4}$$

where the inequality holds when $\Delta \gg s_{\rm grid}$; otherwise, $t_{\rm diff} \sim t_{\rm col}$ until $\Delta \sim s_{\rm grid}$. Since $t_{col} \propto 1/n \propto \Delta$, where $n \equiv dN_{tp}/dr$ is the number of particles per radial distance, a ring expands as $\Delta \propto t^{1/3}$.

We used a hybrid symplectic/Bulirsch-Stoer integrator instead of a traditional mixed-variable symplectic integrator. The integration scheme for the second-order hybrid integrator is as follows.

(i) The coordinates remain fixed. Each body receives an acceleration owing to the other bodies (but not the Sun), weighted by a factor K, lasting for *dt*/2.

(ii) The momenta remain fixed, and each body shifts position by an amount $t\sum_i p_i/2M_*$.

(iii) Bodies not in an encounter move on a Keplerian orbit about the Sun for *dt*. For bodies in an encounter, the Kepler terms, and the close encounter terms weighted by (1-K), are integrated numerically for *dt*.

(iv) As step (ii).

(v) As step (i).

3. Simulations and early results

Firstly, we used circular rings of particles without any planets as a comparison test to the origin paper L07. The parameters of the simulation include the coefficient of restitution (ε), the size of a grid element (s_{grid}), the number of test particles (N_{tp}), and the initial orbital elements of the test particles. For the test run, the mass of the particles are set to 0. The central body's mass is 1 $M_{\odot}$ .We simulate narrow rings with mean radius $\overline{r} = 1\ \mathrm{AU}$ and radial width $\Delta \ll \overline{r}$, and choose *dt* =0.18 yr for the integration time step.

Because of the steep dependence of the diffusion timescale on the width of the ring, $t \propto \Delta^3$, it takes a long time to simulate even a modest increase in Δ . Simulation parameters must be chosen judiciously. We fix $\varepsilon = 0.3$, $N_{\mathrm{tp}} = 10^4$ and $\overline{r} = 1\ \mathrm{AU}$, and seek the optimal values for s_{grid} and $\Delta_0 \equiv \Delta\big|_{t=0}$. To simulate as large an increase in Δ as possible, the simulation should begin with as narrow a ring as possible. For a fixed s_{grid} , the narrowest ring that is not optically thick has unity optical depth $\Delta_0 \sim N_{\mathrm{tp}} \left(s_{\mathrm{grid}}\right)^2 \big/ 2\pi\overline{r}$. The fastest timescale is obtained with the smallest Δ_0 . But we must have $\Delta_0 \geq s_{\mathrm{grid}}$, so the optimal values are $\Delta_0 = s_{\mathrm{grid}} = 2\pi\overline{r} \big/ N_{\mathrm{tp}}$. Rounding up, we set $\Delta_0 = s_{\mathrm{grid}} = 10^{-3}\ \mathrm{AU}$. Initially, the ring particles were uniformly distributed in a ring with edges at $1 \pm (5\times10^{-4})\ \mathrm{AU}$. We plot a histograms of the ring particles number counts vs. radius at *t* = 0, 1000 and 100 000 yr in Fig. 1, and it is in excellent agreement with Fig. 4 of L07, which shows the density evolution in a diffusing ring.

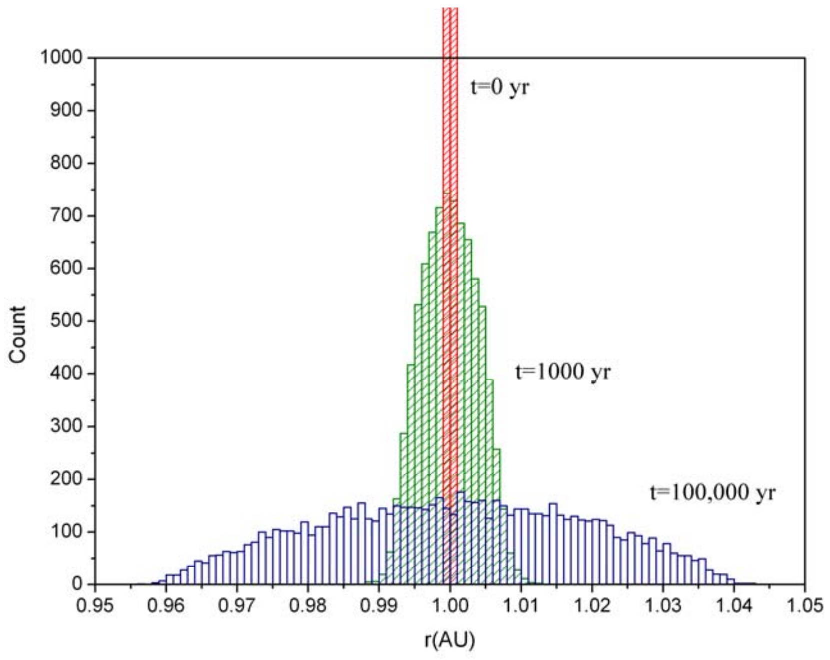

Fig.1. Density evolution in a diffusing ring. Histograms of the ring particles number counts are shown at t = 0, 1000 and 100 000 yr from the test simulation.

Secondly, as the test run showed that the implementation of the algorithm to MERCURY was successful, we put a big body inside the ring as a moonlet to simulate the Saturn ring scenario. The moonlet has a mass of $M_{\text{moon}} = 3\times10^{-6} M_{\odot}$, and the initial position of the moonlet is at $x_0 = 1\ \text{AU}$, $y_0 = 0\ \text{AU}$ in Cartesian Coordinates, therefore the Hill Radius of the moonlet is

$$R_{\text{H}} = r\left(\frac{M_{\text{moon}}}{3M_{\odot}}\right)^{1/3} = 10^{-2}\ \text{AU}. \quad (5)$$

The radius of the moonlet was set to $R_{\text{moon}} = 10^{-3}\ \text{AU}$, same as the s_{grid}. In order for the ring to not diffuse too much during the time of the run which is about 50 yr, and wide enough for the moonlet to be imbedded in, but also not to be too optically thin to keep a good resolution, we set $\Delta_0 = 10^{-1}\ \text{AU}$ with the ring edges at $1 \pm (5\times10^{-2})\ \text{AU}$, which is about 10 times of R_{H}, the ring particles are still massless, and all the other parameters are the same as the test run. Fig. 2, 3, 4 and 5 shows positions of the moonlet and all the ring particles at 0, 1, 10 and 50 yr accordingly. Fig. 3 shows the moonlet created a partial gap in the disk with a propeller structure around it, but as the integration continues, the partial gap spreads out and finally forms a full gap, as shown in Fig.4 and 5. We think this is due to the initial setting of the parameters that the viscosity of the disk is not big enough comparing to the real Saturn ring. We also did a simulation that gives all the ring particles the same mass $M_{\mathrm{p}} = 3\times10^{-16} M_{\odot}$, and kept all the other parameters the same as the last run, and the results look the same.

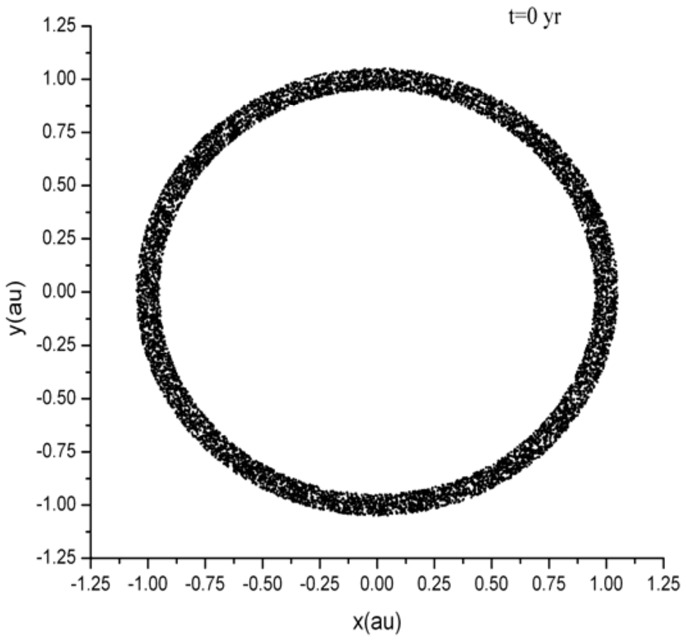

Fig.2. Positions of the moonlet and all the ring particles at *t*=0 yr.

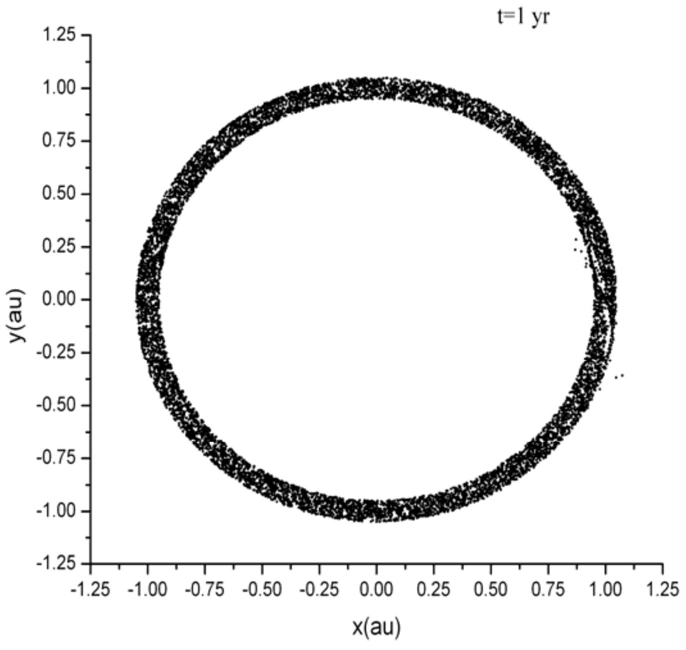

Fig.3. Positions of the moonlet and all the ring particles at *t*=1 yr.

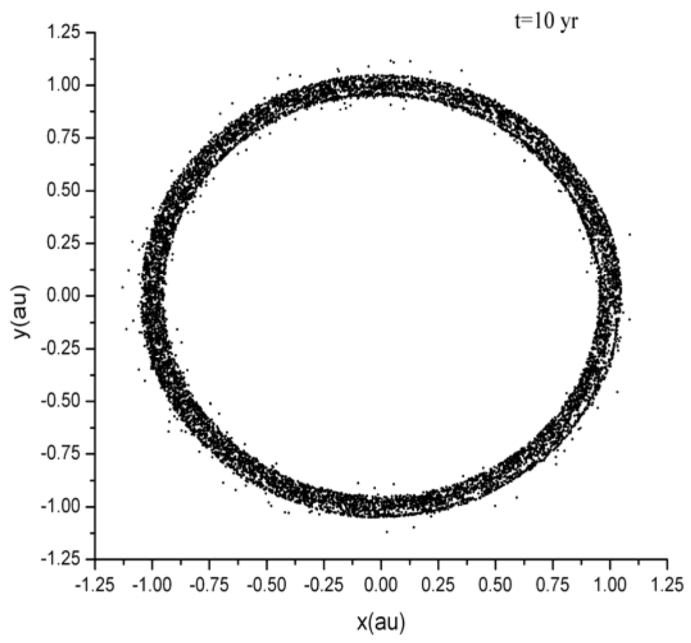

Fig.4. Positions of the moonlet and all the ring particles at *t*=10 yr.

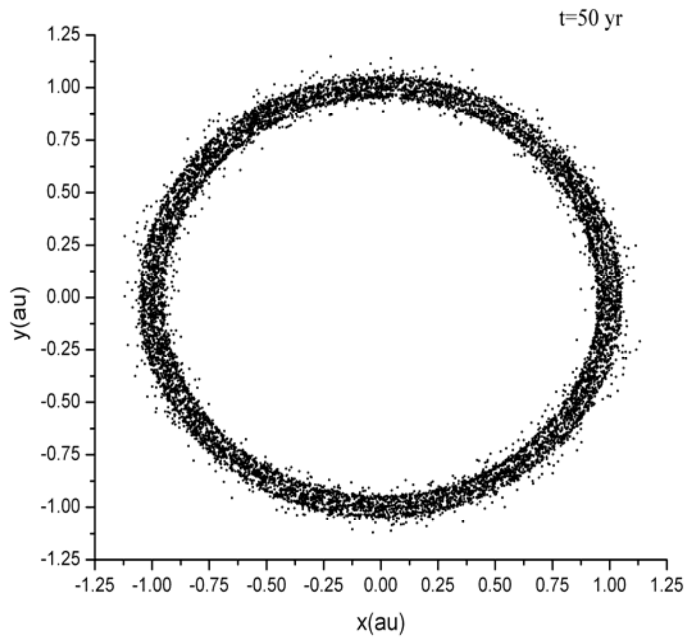

Fig.4. Positions of the moonlet and all the ring particles at *t*=50 yr.

4. Summary and outlook

In this work, we imply a simple, fast algorithm developed in L07, which numerically evolves disks of inelastically colliding particles surrounding a central star, to the N-body simulation code MERCURY. We use hybrid symplectic integrator to simulate a moonlet in the Saturn ring scenario, and produced a propeller structure around the moonlet which opens a partial gap in the ring. However, the initial settings of the parameters need to be adjusted in order to fit the real disk viscosity in order to maintain the partial gap.

The next steps of the work would be to find out the best combinations of the free parameters, namely $\bar{r}$, Δ, s_{grid} and μ; to apply the code on the propellers in Saturn's rings and compare with the observations, and to optimize the code, maybe parallelize it.

5. Acknowledgements

We thank the organizers of the 2011 ISIMA held in Peking University for providing this great opportunity of collaborative research.

References

Chambers J. E., Migliorini F., 1997, BAAS, 29, 1024

Chambers J. E. 1999, MNRAS, 304, 793

Crida, A., Papaloizou, J. C. B., Rein, H., Charnoz, S., & Salmon, J. 2010, AJ, 140, 944

Lithwick Y. & Chiang E. 2007, ApJ, 656, 524

Levison, H. F., & Duncan, M. J. 1994, Icarus, 108, 18

Pan F. & Chiang E. 2010, ApJL, 722, L178

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, Numerical Recipes in FORTRAN: The Art of Scientific Computing (2nd ed.; Cambridge: Univ. Press)

Rein, H., & Papaloizou, J. C. B. 2010, arXiv:1006.1643

Salo, H. 1995, Icarus, 117, 287

Tiscareno, M. S., et al. 2010, ApJ, 718, L92

Trulsen, J. 1971, Ap&SS, 12, 329

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