Mon. Not. R. Astron. Soc. **346,** 924–932 (2003)

A tree code for planetesimal dynamics: comparison with a hybrid direct code

Adrián Brunini★ and Héctor R. Viturro

Facultad de Ciencias Astronómicas y Geofísicas, Universidad Nacional de La Plata – IALP, CONICET, Paseo del Bosque s/n (1900) La Plata, Argentina

Accepted 2003 August 22. Received 2003 July 15; in original form 2002 November 26

ABSTRACT

We present a tree code for simulations of collisional systems dominated by a central mass. We describe the implementation of the code and the results of some test runs with which the performance of the code was tested. A comparison between the behaviour of the tree code and a direct hybrid integrator is also presented. The main result is that tree codes can be useful in numerical simulations of planetary accretion, especially during intermediate stages, where possible runaway accretion and dynamical friction lead to a population with a few large bodies in low-eccentricity and low-inclination orbits embedded in a large swarm of small planetesimals in rather excited orbits. Some strategies to improve the performance of the code are also discussed.

Key words: accretion, accretion discs – methods: numerical – celestial mechanics – Solar system: general.

1 INTRODUCTION

Direct N-body numerical simulation of planetesimal dynamics in Keplerian discs is at present a subject of growing interest (Kokubo & Ida 1996, 1998, 2000; Chambers & Wetherill 1998; Levison, Lissauer & Duncan 1998; Brunini & Fernández 1999). The development of sophisticated and efficient algorithms, like the fully symplectic maps of Wisdom & Holman (1991) and Duncan, Levison & Lee (1998) and the hybrid integrators of Chambers (1999) and Brunini & Melita (2002), has made it possible to perform numerical simulations for periods comparable to the age of a mature planetary system. However, the systems simulated so far contain only a relatively modest number of planetesimals N, several orders of magnitude smaller than the number of planetesimals in nature. This limitation is in part due to the way interparticle forces are evaluated. In most codes, the planetesimal–planetesimal interaction is evaluated directly. This process scales as N^2, explaining why simulations including more than a few thousand particles are, for the moment, impracticable on current low-cost workstations. The correct modelling of important collective processes, like dynamical friction and density waves (Cionco & Brunini 2002), requires simulations to be performed with a larger number of planetesimals. As the number of particles grows, so does the computational time involved, and fast and efficient algorithms become vital.

Tree codes are among the most efficient algorithms to compute the gravitational interaction of systems of massive numbers of interacting particles, and they are now widely used in simulations involving collisionless systems, such as stellar systems, galactic dynamics and cosmological simulations. However, their capabilities have not yet been exploited in the field of planetary accretion simulations, where collisions are important.

Richardson (1993) developed a tree code for planetesimal dynamics, especially designed to study the dynamical evolution of planetary rings. It includes the sliding box strategy of Wisdom & Tremaine (1988), which is not adequate to study extended and non-homogeneous planetesimal systems.

Richardson et al. (2000) reported a new tree code, which is an adaptation of a collisionless code, and thus not originally designed to cope with the kind of problems in which we are interested. This code can run in parallel on several processors, making it capable of coping with a very large number of particles ($N \sim 10^6$). The integration of the equations of motion is carried out using the well-known leapfrog algorithm, which has the advantage of being symplectic and of second order, although only one evaluation of the interparticle force is needed per integration step. Close encounters are not treated in any special way, but temporal adaptivity ensures their accurate treatment. Independent time-steps h for each particle are computed in the usual manner (Duncan et al. 1998; Brunini & Melita 2002):

★E-mail: abrunini@fcaglp.unlp.edu.ar

$h \propto r^{3/2}$,

where r is the distance to the particle that contributes the largest acceleration (including the Sun). For each particle, the nearest-neighbour search is performed according to the algorithm of Bentley & Friedman (1979). For 8–32 of the nearest neighbours of each particle, a collision is assumed if the relative motion extrapolated from the position at the beginning of each time-step, and with constant velocity, predicts it.

In the present paper, we present a new tree code specially designed for planetesimal dynamics that accurately predicts and integrates close encounters. Our main objective is not to present a better code, but to perform a careful comparison against an N-body hybrid code developed by one of us (Brunini & Melita 2002), in order to assess the potential advantages of this kind of algorithm on modest computers.

The paper is organized as follows: Section 2 describes the main features of the code. Numerical tests and simulations are presented in Section 3. The last section is devoted to conclusions.

2 DESCRIPTION OF THE CODE

We have adopted the basic algorithm of the tree code described by Barnes & Hut (1986, 1989), Hernquist (1987), Barnes (1995) and Pfalzner & Gibbon (1996). In what follows, we will describe the main features of the code.

2.1 Computing accelerations

The first step in computing the acceleration of a particle in a tree code is the construction of the tree structure. Following previous authors, we have adopted the octal tree scheme. The construction itself begins by identifying the largest cell (the root) of the tree with a cube containing the entire system of particles. This cell is subdivided into eight equal subcells; these subcells are in turn recursively subdivided into eight cells each, and so on until the ending cells (the leaves) contain either one or no particle.

Next, total masses, centre-of-mass coordinates, critical radii and quadrupole moments are calculated for each cell, by recursively descending the tree. A fundamental aspect in this step is the use of proper recursive formulae for computing quadrupole moments (see Hernquist 1987; Pfalzner & Gibbon 1996).

The last step in computing the acceleration of a particle starts traversing the tree from the root to the leaves. A cell will contribute as a whole to this acceleration if it is far enough from the particle. Otherwise, the cell is discarded and the process is repeated with its *children*. The walk ends when all subdivided cells contain either one or no particle. To decide whether or not a cell must be subdivided, a multipole acceptance criterion (MAC) is used (Salmon & Warren 1994). The original MAC (Barnes & Hut 1986) is based on the aperture angle θ, i.e. the ratio between the size of the cell and its distance from the particle being accelerated: a cell is subdivided whenever θ is greater than a certain threshold. We have incorporated in our code the modified Barnes multipole acceptance criterion (Barnes 1993, 1994).

2.2 Time-step integration

In most tree codes, the second-order symplectic leapfrog algorithm is the common choice for integrating the equations of motion, and we adopt this scheme. The approach used to integrate the equations of motion is as follows.

The first step in the procedure is to find those particles satisfying the *close encounter condition*. In this application, the distance used in searching nearest neighbours is proportional to the Hill's radius of each particle, which we have defined as

$$R_{\rm H} = r_{\rm p} \left(\frac{m_{\rm p}}{3\,{\rm M}_\odot} \right)^{1/3}, \tag{1}$$

where $r_{\rm p}$ is the heliocentric distance of the particle and $m_{\rm p}$ its mass. All those pairs closer than three times this distance were considered in the close encounter condition. The selection of this distance has been made taking some considerations into account: the most important one is that larger values of the factor before $R_{\rm H}$ make transitions between heliocentric and planetocentric motion more frequent, producing large jumps in the energy of the system. (These transitions, in the way the code works, destroy the symplectic nature of the leapfrog, degrading the conservation of energy.) On the other hand, if this factor is too large, the computational cost to integrate close encounters will be more than can be justified by the accuracy of the overall integration (Chambers 1999). Using a too small value, close encounters will not be integrated accurately. There is also the fact that this was the common choice by other authors (Levison & Duncan 1994; Duncan et al. 1998; Chambers 1999; Brunini & Melita 2002). As we are interested in comparing the performance of the tree code with that of Brunini & Melita (2002), it seems to be the best choice.

As part of the process of integration of close encounters, the integration step $t_{\rm e}$ used to update the position and velocity of these particles is determined. This time-step is a fraction of the global time-step $t_{\rm g}$, with which the whole system is updated. The step $t_{\rm e}$ is chosen as follows:

(a) For each pair i, j in a close encounter situation, we compute the quantity

$$t_{ij} = t_{\rm g} \left(\frac{r_{ij}}{r_{\rm CM}} \right)^{p}, \tag{2}$$

where $r_{\rm CM}$ is the distance from the centre of mass of both particles to the central mass, r_{ij} is the mutual separation and p is a constant.

If $t_{\rm min} = \min\{t_{ij}\}$, we choose the largest integer $n_{\rm e}$ such that $n_{\rm e} t_{\rm min} \leqslant t_{\rm g}$, and the integration of the close encounters proceeds as follows: the particles in close encounter, are advanced $n_{\rm e}$ steps up to completing a time interval $t_{\rm g}$.

In each of these n_e steps the program computes the interaction only over those particles in a close encounter condition. The program makes no distinction between the particles that are in a close encounter and the rest of the system, and always applies the same scheme based on the tree code previously described.

It is important to remark that the force on each one of the particles in a close encounter condition includes the contribution of all the particles of the system. Moreover, the particles that are not in the close encounter remain static. This means that the tree is rebuilt at each of these n_e steps. This is necessary because the particles that are in close encounters are moving relative to those that remain static.

(b) After the program completes these n_e steps, it computes the interaction over the rest of the particles that were not in a close encounter condition, updating their positions and velocities by applying a leapfrog step of amplitude t_g.

This process is repeated up to the end of the time interval, always beginning with the search to determine those particles in a *close encounter condition*.

We have experimented with different values of the exponent p in equation (2). As in Brunini & Melita (2002), we have decided to use $p = 1$, in contrast to the usual choice of $p = 3/2$ (Levison et al. 1998; Richardson et al. 2000). It is worth noting that with larger values of p we would obtain more accuracy in the integration of close encounters. However, we have observed that for $p = 3/2$ the number n_e increases by a factor of almost 100, making the runs very time-consuming in our modest computational devices, but without an evident improvement in energy conservation, which is limited by other factors such as the degree of the multipole expansion in the whole integration and the aperture angle θ.

As for the case of the choice of the factor of 3 before R_H, it is important to mention that Brunini & Melita (2002) have also used $p = 1$, and in order to make our results directly comparable we maintain the same criterion.

As can be seen in the numerical experiments shown in the next section (see Fig. 1), the accuracy in the integration of close encounters is satisfactory with this choice of parameters. We have observed that for $p = 1$ the number of small steps t_e is within the range 1 to 2000, with a typical value of 400. It is worth noting that even choosing a large value of p the global error may be reduced by reducing the overall time-step t_g.

A fundamental aspect in this kind of problem is the nearest-neighbour searching method used to construct the list of particles that are to be considered in a close encounter.

We used the tree structure again. The search method is a straightforward modification of the one already outlined for computing the acceleration of a particle. In this respect we follow the same ideas described in Hernquist & Katz (1989).

2.3 Accretion algorithm

The method employed to search for collisions is similar to that used to search for particles satisfying the *close encounter condition*, but now the search is restricted to the list of particles that are in a close encounter.

The distance used in searching for nearest neighbours is now the sum of the geometrical radii of the particles. If a pair of particles are in contact, we replace them by a new particle, placed at the centre of mass of the pair and conserving the linear impulse. This scheme is applied after each integration step t_e.

2.4 Some details of implementation

The leapfrog routine deserves a little attention. The following is a pseudo-code skeleton of the leapfrog loop of the program.

```
int leapfrog(void)
    {
    time = initialtime;
    while( time <= finaltime )
      {
      /*
      * Search neighbors for each particle whose distances are less than
      * a value proportional to the sum of the respective Hill radii.
      * Also determine "ne" and "te".
      */
      search_encounters();
      if(they_are_encounters)
        {
        /*
        * Integrate only those particles that are in encounter using
        * "ne" steps of "te" amplitude.
        */
        for(i=0 ; i<ne ; i++)
```

```
      {
      /*
      * Update pos. and vel. according to leap-frog scheme
      */
      update_vel(te/2); /* update velocities 1/2 step */
      update_pos(te); /* update positions 1 step */
      create_octtree(); /* rebuild tree */
      calculate_phi(); /* compute potential and acceleration */
      update_vel(te/2); /* update velocities 1/2 step */
      /*-----------------------------------------------------------*/
      /*
      * After each step, search particles for agglomeration.
      * The search is made only over the subset of particles that
      * are in encounter.
      */
      accrete_particles();
      /*-----------------------------------------------------------------*/
      }
    }
  /*
  * Here "update_vel/pos" must actualize only the rest of particles.
  * It is, those not satisfying the encounter condition.
  */
  update_vel(deltatime/2); /* update velocities 1/2 step */
  update_pos(deltatime); /* update positions 1 step */
  create_octtree(); /* rebuild tree */
  calculate_phi(); /* compute potential and acceleration */
  update_vel(deltatime/2); /* update velocities 1/2 step */
  /*-------------------------------------------------------------------*/
  /*
  * After each big step, search particles for agglomeration, again.
  */
  accrete_particles();
  /*-------------------------------------------------------------------*/
  time+=deltatime;
  /*-------------------------------------------------------------------*/
  }
return;
}
```

3 PERFORMANCE OF THE CODE

To test the performance of the code, we have carried out a set of numerical experiments. All of them are based on the Kokubo & Ida (1996) runaway simulations. It is worth mentioning that Brunini & Melita (2002) have used the same test run, and thus our results should be directly comparable with those of Kokubo & Ida (1996). The first one consists in the integration of 2000 equal-mass planetesimals (mass 10^{24} g, bulk density 2 g cm^{-3}), distributed uniformly in a thin annulus of width $\Delta a = 0.1$ au centred at 1 au. The initial eccentricities and inclinations were Rayleigh distributed with $\sigma_e = 2\sigma_i = 4R_H/a$, where R_H is the mutual Hill radius.

In the first simulation, we deactivate the possibility of mergers, in order to test the global accuracy of the code, the accuracy in the computation of close encounters, and also the efficiency of detecting them.

We have performed a series of experiments using different time-steps t_g and aperture angle $\theta = 0.7$. The global results are summarized in Table 1, where we show the relative error in the total energy of the system after $t = 10$ yr. We can see that up to $t_g = 0.01$ yr the behaviour is satisfactory, but for larger time-steps the conservation of energy is degraded. In Fig. 1 we have plotted the data of the first two columns of Table 1. For small time-steps the error is proportional to $1/t_g$, revealing that it is dominated by the round-off errors (Gear 1971). For time-steps $t_g > 4 \times 10^{-3}$ yr the error starts to be dominated by the leapfrog integration scheme, and, as expected, it behaves like t_g^2. For larger time-steps the slope is much steeper (as steep as the fourth power). In this region the error is not dominated by the leapfrog scheme but for the computation of the interaction. As shown in Table 1, for $t_g \leqslant 0.01$ yr close encounters are detected successfully (the number of close

Table 1. Dependence of the global error with the time-step after $t = 10$ yr. The total number of detected close encounters after $t = 10$ yr normalized to the total number of close encounters detected for the run with $t_g = 0.001$ yr is also shown.

t_g (yr)	$\lvert\Delta E\rvert/\lvert E_0\rvert$ ($t = 10$ yr)	N_{enc}
0.0010	8.3×10^{-6}	1
0.0011	8.3×10^{-6}	0.98
0.0013	4.5×10^{-6}	0.98
0.0015	5.2×10^{-6}	1.00
0.0017	3.9×10^{-6}	1.00
0.0019	4.1×10^{-6}	0.97
0.0022	2.9×10^{-6}	0.97
0.0025	2.9×10^{-6}	0.98
0.0030	2.8×10^{-6}	0.99
0.0035	2.1×10^{-6}	1.01
0.0042	1.9×10^{-6}	1.00
0.0050	2.3×10^{-6}	0.98
0.0060	3.3×10^{-6}	0.98
0.0071	4.9×10^{-6}	0.99
0.0083	4.1×10^{-6}	0.98
0.0100	5.9×10^{-6}	0.98
0.0150	1.6×10^{-5}	0.94
0.0220	1.0×10^{-4}	0.87
0.0330	4.1×10^{-4}	0.67
0.0500	1.8×10^{-3}	0.39
0.0630	4.2×10^{-3}	0.28
0.0800	1.0×10^{-2}	0.23
0.1000	2.4×10^{-2}	0.20

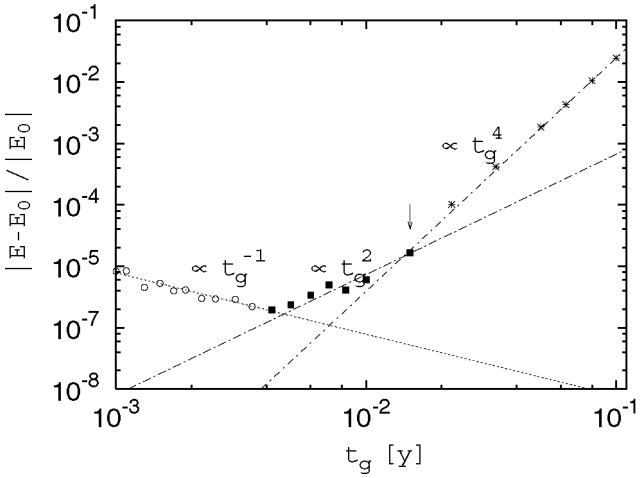

Figure 1. Relative error in the total energy of the system after $t = 10$ yr versus the time-step t_g. The arrow indicates the transition between successful and unsuccessful detection of close encounters. We have used different symbols for each one of the regimes: open circles where the error is dominated by round-off, squares where the error is dominated by the local truncation error of the leapfrog scheme, and stars where the error is dominated by the loss of close encounters.

encounters varies by less than 3 per cent among the different runs), but for $t_g > 0.01$ yr (the point marked with the arrow in Fig. 1) the code is not able to detect all the close encounters and the interaction is poorly modelled. This behaviour is probably due to the strategy we have implemented to detect them. A predictive strategy such as the one used by Richardson et al. (2000) will be tested in the future.

As expected, for $t_g > 0.01$ yr the number of close encounters not detected behaves as t_g^2 (see Table 1), explaining the fact that in this region the error behaves as t_g^4, because the integration of the close encounters was also performed by means of the leapfrog scheme. It is worth noting that, losing even only few close encounters, energy conservation may be degraded.

After some experimentation, in all the numerical experiments presented in what follows, we have decided to use a time-step $t_g = 0.0025$ yr.

We have also experimented with the aperture angle θ. In Table 2 we present the main results of the simulations with different aperture angles. As can be seen, the multipole expansion is good even for large values of the aperture angle. We have decided to use the standard value

Table 2. Relative error in the total energy computed after 1000 integration steps (we used $t_g = 0.0025$ yr), for different aperture angles.

θ (rad)	$\Delta E/E_0$
0.3	-3.8×10^{-8}
0.4	-6.2×10^{-8}
0.5	-1.3×10^{-7}
0.6	-2.9×10^{-7}
0.7	-5.1×10^{-7}
0.8	-2.5×10^{-6}
0.9	-4.1×10^{-6}
1.0	-8.7×10^{-6}

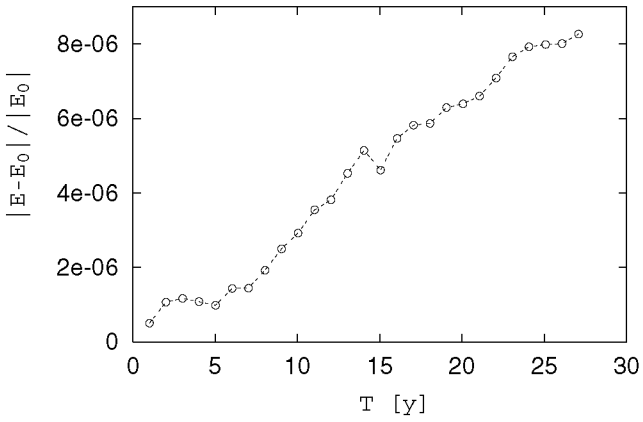

Figure 2. Temporal evolution of the relative error in the total energy of the system.

$\theta = 0.7$, which is the one used in almost all the simulations reported in the literature. It is worth noting that the CPU time is 10 times longer for $\theta = 0.3$ than for $\theta = 0.7$.

For these choices of t_g and θ, we have analysed the temporal behaviour of the total energy of the system. If the method were fully symplectic, the relative error of the total energy should not show secular trends. However, as can be seen in Fig. 2, it grows almost linearly with time. Nevertheless, the secular rate is $\sim 3.3 \times 10^{-8}$ yr^{-1}, and we consider this behaviour as satisfactory.

Thus, we have performed a new simulation, this time activating the possibility of mergers. In this new simulation, the radii of the planetesimals were artificially enhanced by a factor of 5 (as done by Kokubo & Ida 1996). We have performed the same simulation using the tree code (TC) and also our hybrid code (HC), using the same initial conditions, with $N = 2000$ and $N = 4000$, following the integration up to $t = 20\,000$ yr. Brunini & Melita (2002) have tested their HC with this same numerical experiment. In the runs with the HC presented in this paper we have used a time-step of 0.005 yr (i.e. the same time-step as in Brunini & Melita 2002), ensuring that the results reported here are fully consistent with those of Kokubo & Ida (1996).

It is worth noting that, for the case $N = 4000$, the mass of each body is half the mass used in the simulation with $N = 2000$, so the total mass of the system is conserved. The comparison of the results obtained with both codes is shown in Figs 3 to 5. Fig. 3 displays the evolution of the total number of bodies in the simulations with $N = 2000$. For the HC the final number of particles after $t = 2 \times 10^4$ yr is 780, whereas for the TC this number is 820. The evolution of the rms eccentricity and inclination of the system is also shown in Fig. 4 for the same N. The TC reproduces the expected evolution, where σ_e is always near $2\sigma_i$. Also for comparison we show the temporal evolution of these quantities in our simulation with the HC. Finally, we observe in Fig. 5 the evolution of the mass of the largest body, and the mean mass of the system for the simulation with $N = 4000$. The two simulations behave almost identically. The classical behaviour of runaway accretion is evident. In the simulation with $N = 2000$, the final masses of the largest body of both simulations differ by a factor of 2 due to a stochastic event at the end of the HC simulation.

It is interesting to examine how the code behaves with the number of particles. We have computed the CPU time used per year of evolution. Fig. 6 shows the evolution of the CPU time against the total number of bodies for the simulation with $N = 2000$. The numbers in this figure are averages over time intervals of 50 yr. Also shown is the behaviour of the HC. As can be seen, the TC, even using a smaller step size (t_g for the HC is 0.005 yr), is faster than the HC. However, at the beginning of the simulation, and until the total number of bodies drops below $N \sim 1500$, it behaves like (or even worse than!) a particle–particle method, with an exponent $q \sim 2.99$. When less than 1500 bodies remain in the system, the TC behaves as N^q with $q \sim 1.44$, i.e. within the expected indices for a collisionless TC (Jernigan & Porter 1989).

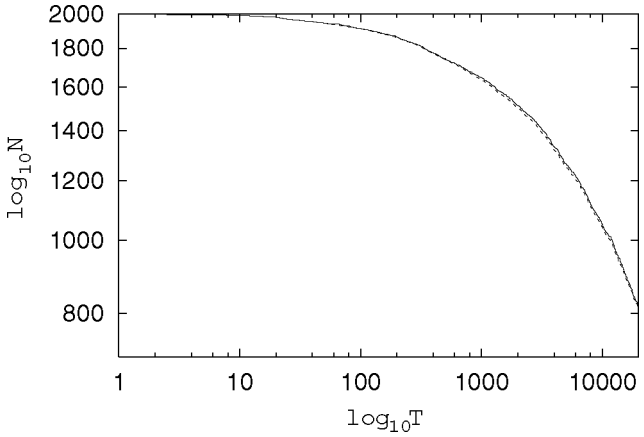

Figure 3. Evolution of the number of bodies for the simulation with 2000 equal-mass bodies: solid line, TC; dashed line, HC.

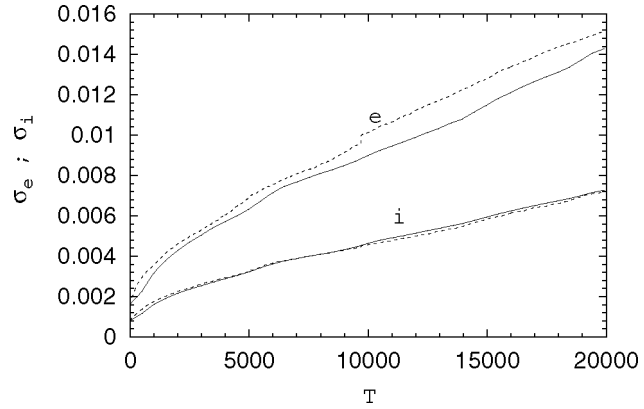

Figure 4. Evolution of the rms eccentricity and inclination for the simulation with 2000 equal-mass bodies: dashed line, TC; solid line, HC.

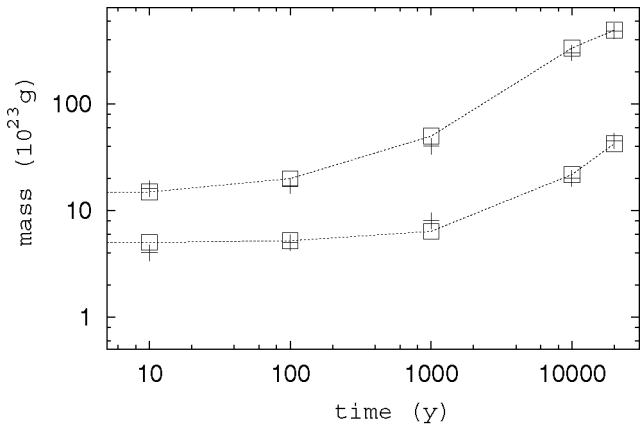

Figure 5. The mass of the runaway object in the simulation and the mean mass of the rest of the objects, except the largest one, for the simulation with 4000 equal-mass bodies: squares, TC; crosses, HC.

Fig. 7 displays the CPU time for the TC and for the HC for the simulation with $N = 4000$. In this case the exponent q at the beginning of the simulation is $q = 2.71$, and when the number of particles drops to $N \sim 1900$ the exponent reaches the value $q \sim 1.41$.

We have also performed a set of new numerical experiments, where the initial conditions are the same as those in the previous simulation with $N = 2000$, but the mass of the bodies was reduced by some factor, ranging from $1/8$ to 1. The exponent q at the beginning of each run and the mean q are shown in Table 3. Each simulation was stopped when $N = 1100$ objects remained in the system. As can be seen, the behaviour is within that expected. The code tends to behave as a non-collisional tree code when the mass of the system is reduced, and the interaction distance (i.e. $3R_{\rm H}$) is reduced accordingly.

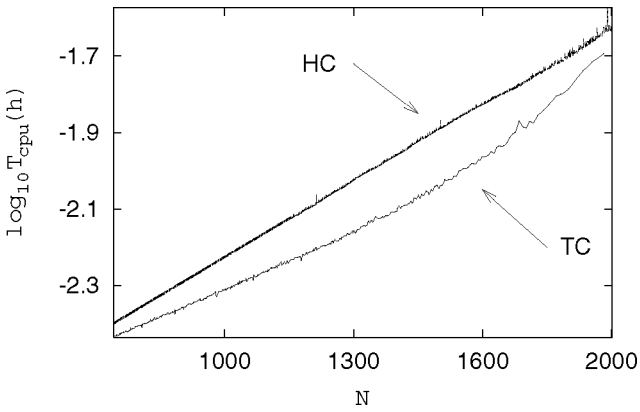

Figure 6. Evolution of the CPU time per year of evolution versus the number of objects, for the system of 2000 equal-mass objects.

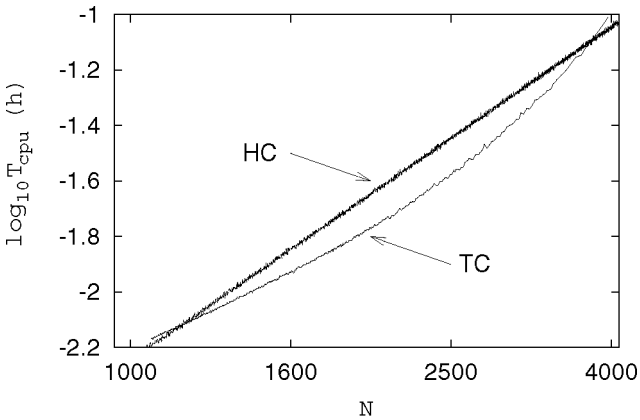

Figure 7. Evolution of the CPU time per year of evolution versus the number of objects, for the system of 4000 equal-mass objects.

Table 3. Scaling exponent for different total masses.

m/m_0	$\langle q \rangle$	q_0
1	2.02	3.04
0.5	1.82	2.11
0.25	1.69	1.98
0.125	1.58	1.63

It is worth noting that tree codes in general, based on multipole expansions of the interacting potential, are not appropriate to handle a small number of bodies, when the distribution of mass in the system is far from homogeneous, because the time-steps that should be used must be very small or the multipole expansions must be truncated at very high orders. We have simulated the evolution of a planetary system composed of the four giant planets of the Solar system, with their masses enhanced by a factor of 50, which is the classic test for hybrid codes. The behaviour of the TC was not satisfactory in this case, for the reasons already outlined.

4 CONCLUSIONS

In this paper, the performance of a tree code specifically designed for planetesimal dynamics was compared against that of a direct hybrid code, which is becoming a standard to study the dynamical behaviour of planetesimal systems. The overall behaviour of the code was tested by reproducing some results already published. Regarding the main features of the code, the main results we have obtained are as follows.

(i) **Accuracy.** For our choice of parameters, energy conservation is as good as 10^{-8} per close encounter. This is due to the use of fixed step sizes together with a leapfrog algorithm to compute them.

(ii) **Speed.** At comparable accuracy, the code is, in general, faster than the hybrid code. However, the CPU time is strongly dependent on the degree of 'collisionality' of the system, determined by the distribution of mass. The code behaves like a particle–particle method if

the dynamics of the system is dominated by close encounters. The parameter determining the behaviour of the system is the total mass in planetesimals rather than the number of bodies, as shown by our simulations with $N = 2000$ and $N = 4000$ objects.

(iii) **Potential uses.** The tree code is potentially powerful to study the intermediate stages of planetesimal accumulation, where possible runaway accretion and dynamical friction lead to a population with a few large bodies in low-eccentricity and low-inclination orbits embedded in a swarm of small planetesimals in rather excited orbits. During the early stages, planetesimal systems are too collisional and the code behaves no faster than classical direct methods. Conversely, when the system is composed of a few interacting protoplanets, the multipole expansion degrades the accuracy of the results due to inhomogeneities in the distribution of mass in the system.

We plan to improve the code in several ways.

(1) To take advantage of the fact that the bodies evolve mostly under a central potential – a possible strategy in this direction would be to implement a hybrid algorithm, where the evolution under the central potential is computed with the analytical formulation of the two-body problem.

(2) To improve the computation of mergers, probably introducing the predictive strategy of Richardson et al. (2000).

(3) To avoid the rebuilding of the tree during the treatment of close encounters.

ACKNOWLEDGMENTS

We want to acknowledge the support by the Instituto Astrofísico de La Plata (CONICET).

REFERENCES

Barnes J. E., 1993, in Muñoz-Tuñón C., Sanchéz F., eds, The Formation and Evolution of Galaxies. Cambridge Univ. Press, Cambridge, p. 409
Barnes J. E., 1994, in Benz W., Barnes J., Müller E., Norman N., eds, Computational Astrophysics. Springer, Berlin
Barnes J. E., 1995, in Muñoz-Tuñón C., Sánchez F., eds, The Formation of Galaxies. Cambridge Univ. Press, Cambridge, p. 399
Barnes J. E., Hut P., 1986, Nat, 324, 446
Barnes J. E., Hut P., 1989, ApJS, 70, 389
Bentley J. L., Friedman J. H., 1979, Comput. Surv., 11, 397
Brunini A., Fernández J. A., 1999, Planet. Space Sci., 47, 591
Brunini A., Melita M. D., 2002, Icarus, 160, 32
Chambers J. E., 1999, MNRAS, 304, 793
Chambers J. E., Wetherill G. W., 1998, Icarus, 136, 304
Cionco R. G., Brunini A., 2002, MNRAS, 334, 77
Duncan M. J., Levison H. F., Lee M. H., 1998, AJ, 116, 2067
Gear C. W., 1971, Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall, Englewood Cliffs, NJ
Hernquist L., 1987, ApJS, 64, 715
Hernquist L., Katz N., 1989, ApJS, 70, 416
Jernigan J. G., Porter D. H., 1989, ApJ, 71, 871
Kokubo E., Ida S., 1996, Icarus, 123, 180
Kokubo E., Ida S., 1998, Icarus, 131, 171
Kokubo E., Ida S., 2000, Icarus, 143, 15
Levison H. F., Duncan M. J., 1994, Icarus, 108, 18
Levison H. F., Lissauer J. J., Duncan M. J., 1998, AJ, 116, 1998
Pfalzner S., Gibbon P., 1996, Many-Body Tree Methods in Physics. Cambridge Univ. Press, Cambridge
Richardson D. C., 1993, MNRAS, 261, 396
Richardson D. C., Quinn T., Stadel J., Lake G., 2000, Icarus, 143, 45
Salmon J. K., Warren M. S., 1994, J. Comput. Phys., 111, 136
Wisdom J., Holman M., 1991, AJ, 102, 1528
Wisdom J., Tremaine S., 1988, AJ, 95, 925

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