The Astronomical Journal, 127:1218–1226, 2004 February

FAST FORCE ALGORITHMS AND SOLAR SYSTEM INTEGRATIONS

Paul Wiegert,[1] Douglas McNeil, and Martin Duncan
Department of Physics, Queen's University, Kingston, ON K7L 3N6, Canada

and

Hal Levison
Department of Space Studies, Southwest Research Institute, Suite 400,
1050 Walnut Street, Boulder, CO 80302
Received 2003 May 15; accepted 2003 October 16

ABSTRACT

Symplectic integrators have been the backbone of much theoretical solar system research over the past decade. As implemented, they involve the direct computation of the distances between each pair of N particles, a process whose effort grows as $O(N^2)$. A variety of fast [that is, with effort growing more slowly than $O(N^2)$] but approximate force calculation methods have been developed in other areas of research. Several of these algorithms are examined here, and their speed and accuracy are compared with traditional methods, with an eye toward their suitability for solar system research in particular. We find that approximate force algorithms can provide, in some situations, a suitable alternative to traditional ones, with break-even in terms of computation time at particle numbers as low as a few hundred, and often with only modest increases in the short-term error. Though empirically stable on the systems tested here, the effect of approximate methods on the phase-space manifold of the problem remains a concern.

Key words: celestial mechanics — methods: *n*-body simulations — solar system: general

1. INTRODUCTION

Although symplectic integration techniques had been used in solar system integrations in the past (Gladman & Duncan 1990), the field was revolutionized by the methods of Wisdom & Holman (1992; see also Kinoshita, Yoshida, & Nakai 1991; Saha & Tremaine 1992). In these methods, the particles' Hamiltonian is split into different parts and the coordinates are advanced under each part independently, typically by leapfrog but sometimes by other kernels.

The Hamiltonian may be split in a number of ways. The original implementation of Wisdom & Holman involved a splitting into a purely Kepler term plus a single interaction Hamiltonian. Later algorithms, such as SyMBA (Duncan, Levison, & Lee 1998) and Mercury (Chambers 1999), split it into a Kepler, a linear drift, and an interaction term. Regardless of the splitting, the computational cost of the "noninteraction" (i.e., Kepler or Kepler plus drift) Hamiltonian(s) grows linearly with the number of bodies N.

The interaction Hamiltonian incorporates the mutual gravitational perturbations of the bodies in the system. This process, as it is usually implemented, requires explicit determination of the interparticle distances for all N particles, an $O(N^2)$ computation. As N increases, this portion of the calculation can become prohibitively expensive. As many questions of solar system research can best be addressed by large-N simulations (e.g., planet formation, ring dynamics, asteroid and Kuiper belt evolution) and many researchers are beginning to do such experiments, it seems timely to consider how this computational expense might be reduced.

Many algorithms have been developed, both within and outside the astronomical community, to provide less expensive determinations of the interparticle forces for large N-body systems. These invariably involve some loss of force accuracy. Cosmologists and galactic dynamicists have constructed algorithms able to handle millions of particles; however, they are interested in following the particles for far fewer dynamical times than are typical of solar system research. For example, our Galaxy, with an age of 10 Gyr and a rotation period of 250 Myr, has only existed for 40 dynamical times, whereas the solar system, at roughly half that age, has been through thousands of millions.

Thus, solar system simulations require that error propagation be minimal over very many time steps, the size of which is often constrained by the smallest orbit being simulated. For example, a simulation of terrestrial planet formation might ideally follow 10^5 particles for 10^8 yr. Assuming the particle disk has an inner edge at Mercury and the integrator takes 20 steps per orbit there, the task could be completed in, say, 1 month of dedicated computer time only if each time step could take 0.3 ms. However, currently such a calculation would take approximately half an hour per time step with traditional symplectic integrators and current CPUs. Not all solar system processes of interest require such long integration times. However, computing time remains a constraint on many problems, particularly those that involve the modeling of large numbers of particles.

Though time constraints are of much practical concern, more important is the reliability of the integration performed. Integration errors can be classified into different types, including truncation error (introduced by the integration algorithm, owing to its imperfect representation of the physical problem) and round-off (introduced by the finite precision of computer arithmetic). Here we are most interested in the errors introduced by the use of approximations to the forces/accelerations at each time step, and their effect on symplectic integration schemes. We note however that even direct $O(N^2)$ methods are approximations themselves, and good at best to the number of bits used to represent real numbers, typically 64 (i.e., "REAL*8" in Fortran or "double" in C).

[1] Current address: Department of Physics and Astronomy, University of Western Ontario, London, ON N6A 3K7, Canada; pwiegert@uwo.ca.

What constitutes unacceptable error varies with the type and purpose of the simulation in question. In solar system dynamics, the interest is usually in the reproduction of the gross statistical behavior of a system rather than the detailed path of each of the particles. Given the frequent presence of chaos in simulations of the solar system, its subsystems, or precursors, such a statistical approach is often the best that can be hoped for, even given hypothetical error-free integration methods.

We will consider only dissipationless evolution here. The relative error in the energy and angular momentum thus serve as practical measures of the integration error. The energy and angular momentum can be computed directly from the positions and velocities of the initial and final states. The error in the angular momentum quoted is the size of the vector difference between the start and end values (i.e., $|\boldsymbol{L}_{\rm end} - \boldsymbol{L}_{\rm start}|$, not $|\boldsymbol{L}_{\rm end}| - |\boldsymbol{L}_{\rm start}|$). Methods that conserve these quantities to high accuracy are generally to be preferred; however, even very small or zero errors in E and $\boldsymbol{L}$ do not guarantee the correct evolution of the bodies in question. Nevertheless, they do provide a practical and commonly used metric for measuring an integration algorithm's performance, and we will use them here, bearing the previous caveat in mind.

Here the suitability of a number of approximate algorithms will be examined in the particular context of solar system integrations and symplectic integrators, and our approach will be an experimental one. Three principal questions are of interest: What is the error of the different force calculation methods? What is their relative speed? What is their suitability for solar system simulations in different regimes?

Section 2 outlines the system and algorithms tested here, § 3 presents the results, and conclusions are drawn in § 4.

2. METHODS

2.1. *Direct Methods*

There are a number of codes suitable for the integration of solar system problems that have been made available to the astronomical community by their creators. In particular, Swift (Levison & Duncan 1994) and its subset SyMBA (Duncan et al. 1998) and Mercury (Chambers & Migliorini 1997; Chambers 1999) are widely used. The latter two both incorporate the ability to handle close encounters between particles symplectically. These packages all compute the interaction Hamiltonian through a direct $O(N^2)$ calculation. As Mercury's speed scales the same way in our tests (both analytically and from our experiments), we present results for SyMBA only, for the sake of simplicity.

SyMBA, together with the nonsymplectic RADAU integrator (Everhart 1985, as implemented by the Mercury package) and what we will call the "standard W-H" code (i.e., Wisdom-Holman style, with forces computed directly and without symplectic handling of close encounters) provide the benchmarks against which the performance of the other algorithms will be compared.

Our "standard W-H" code is a traditional Kepler-plus-interaction leapfrog integrator in Jacobi coordinates, as proposed by Wisdom & Holman (1992), written in C. By replacing the subroutine that calculates the effects of the interaction Hamiltonian, the same code also serves as a "driver" for those algorithms that compute approximate forces but are not themselves integrators. Thus the driver allows the approximate methods to be tested within a traditional symplectic integrator.

The standard code and the driver, being in the original Wisdom-Holman style, do not handle close encounters well. Since the large-N systems of interest are likely to involve significant numbers of close approaches between particles, the traditional remedy of softening was applied to prevent spurious heating. This allows comparisons of these simpler methods with the ones that handle close approaches symplectically. The latter do not have any softening applied and thus receive a somewhat different and more demanding test. For the softening kernel, a potential $\Phi = -GM/(r^2 + \epsilon^2)^{1/2}$, or Plummer model, was used, with a fixed $\epsilon = 0.01$ AU. This value was chosen because it is larger than the Hill sphere of all the particles in the disks considered here. This softening was also used in all the approximate methods described below except for FALCON, which uses an adaptive softening (meaning that the softening is mass-dependent in order to reduce the absolute force error; the Ferrers sphere F_0 is used here [see Dehnen 2001, 2002 for more details]).

2.2. *Approximate Methods*

Direct methods are accurate to near machine precision. Most other algorithms suffer some degradation in the accelerations due to the approximations used. With the choice of a sufficiently tight tolerance, the exact nature of which may vary from algorithm to algorithm, the accuracy of approximate schemes can generally be varied, even up to machine precision, though this usually results in an effective reversion to a direct, and hence $O(N^2)$, calculation.

There are a wide variety of gravity simulation codes based on approximate methods currently available. Many of these have been developed for cosmology or galactic dynamics. Some also include smoothed particle hydrodynamics (SPH), a technique that allows the effects of gas to be modeled. Such methods are likely to be useful when considering the very early solar system, when gas played a significant role. However, most are not of sufficiently high accuracy in their force calculations to be likely to be of use for long-term solar system dynamics, and a complete treatment of them will not be attempted here.

Mesh methods construct a grid in the simulation space, possibly on finer scales where particles are densest. Particles are lumped together at the nearest nodes for the purposes of determining the potential, which can be done very efficiently through the use of the fast Fourier transform. The resulting potential is valid only at the grid points, but it can be interpolated to provide an approximation of the force on each particle. Though very fast (Sellwood 1997), they work best on systems with periodic boundary conditions because of their use of Fourier transforms. As there are other methods more suited to isolated systems such as the solar system, we will not examine mesh methods here.

2.2.1. *Tree Codes*

Early tree codes were based on hierarchies of body pairs (Appel 1985; Jernigan & Porter 1989); however, more recent techniques instead break up the simulation space itself into individual regions. The now-standard tree code (Barnes & Hut 1986) breaks up the simulation space (the "root" cell) into a hierarchy of cubic cells, each cell being subdivided into eight daughter cells until each contains only some small number of particles, often just one. At each level of decomposition, the cell's mass and center-of-mass position are computed. Higher order multipole moments of the cell's mass distribution may also be calculated. Once the tree is constructed, the force on each particle is determined by examining the largest eight cells (those just below the root cell) in turn. The ratio of each cell's length l to the distance D between the cell's center of mass and the particle is considered. If l/D is less than some chosen θ, the force on the particle due to the cell and all its contents is approximated by a single body of the cell's total mass located at its center of mass (plus higher order moments, if any). If, rather, $l/D > \theta$ (i.e., the cell is too close), the same test is applied to the cell's internal daughter cells recursively down through the tree.

With a single particle in each cell, it is apparent that the forces between nearby particles will end up being computed directly, while those in cells that are sufficiently far away are replaced for the purposes of the force computation by a single particle at their center of mass. This allows the computation to be reduced to $O(N \log N)$ operations (Barnes & Hut 1986). The choice of θ (the "opening angle," usually taken to be of order unity) determines which particles are far enough for the center-of-mass approximation to be used, allowing the accuracy of the calculation to be varied. Tree codes have already been used for planetesimal dynamics simulations (Richardson et al. 2000).

The code tested here is "Partree," an MPI (Message Passing Interface) parallel implementation of a tree code in C due to Dubinski (1996). The serial algorithm is only slightly modified from the Barnes-Hut model, and the parallelization technique is based on that of Salmon (1991). In these tests, instead of the driver discussed in § 2, we used an implementation of SyMBA modified to incorporate forces from Partree. The code was run on a single CPU using only the democratic heliocentric integrator without close-encounter handling to ensure a fair comparison.

2.2.2. *The Fast Multipole Method*

The fast multipole method (FMM), sometimes called the fast multipole algorithm (FMA), computes the forces between well-separated particles by means of spherical multipole expansions (Greengard & Rohklin 1987; Cheng, Greengard, & Rohklin 1999). Consider a group of $N/2$ particles. To compute the multipole expansion of their gravitational field around one of these particles requires $O(NM/2)$ operations, where M is the number of multipole terms used. Given a second group of $N/2$ particles sufficiently far away that the multipole expansion is valid, the evaluation of the multipole expansion over these points requires another $O(NM/2)$, for a total of $O(NM)$, operations. Thus, the far field can in principle be computed with $O(N)$ effort.

In practice, things are more complicated, as the sorting of particles into well-separated groups, as well as the final application of all their multipole expansions, must be done. In particular, Greengard & Rohklin (1987) developed a method of translating all the distant multipole expansions to a local origin near the particle in question, allowing these far-field contributions to be combined and thus more efficiently computed. In practice, FMM implementations are often found to scale as $O(N \log N)$ (Cappuzo-Dolcetta & Miocchi 1998).

An implementation of FMM written in C is the Distributed Parallel Multipole Tree Algorithm (DPMTA; Rankin 1999). This code was developed within the molecular dynamics community but is easily adapted to astronomical systems. It computes the force due to nearby particles directly, by constructing a tree and using the traditional opening-angle criterion, and uses the FMM algorithm to compute the far-field forces.

Its force accuracy can be tuned over a wide range (from a few digits to full machine precision) by adjusting the number of multipole terms and, thus, can in principle produce exact (at least within round-off) forces as an $O(N)$ process. The force error of most other approximate algorithms can only be improved by decreasing the opening angle [i.e., increasing the distance out to which direct $O(N^2)$ force computation is used]. This makes the DPMTA method particularly attractive. If the number of operations it requires really does grow more slowly that N^2, there must be some N for which it can provide machine-precision forces faster than the direct method—though this limit proves to be currently of little practical use (see § 3.3).

DPMTA version 3.1 is tested here, which can be run either serially or in parallel under either MPI or PVM (Parallel Virtual Machine). It does not accept a softening length but was modified (a simple task) to do so for these tests. This was the only modification made to any of the codes tested here.

DPMTA is not an integrator, but a routine that computes the interparticle forces, and it was tested with the driver (described in § 2.1). An opening angle $\theta = 0.5$ and four multipole terms were used. DPMTA performs fastest when the tree contains 10–50 particles per cell, so the maximum depth of the tree must be varied from 4 at small N to 6 at $N = 10^5$ to maintain this. The test systems (to be discussed in § 2.4) were in an overall root cell 5 AU in extent in the x-, y-, and z-directions.

FALCON [Force Algorithm with Complexity $O(N)$; Dehnen 2000] is similar to DPMTA but differs from it in that it uses Cartesian multipole expansions to compute the far-field forces, significantly speeding computation. Also, because of the symmetry of the Taylor expansion in Cartesian coordinates, along with an implementation that avoids asymmetry between gravity sources and sinks, this algorithm implicitly conserves linear momentum. However, FALCON uses a fixed expansion (up to the octupole), unlike DPMTA, which allows an arbitrary number of multipole terms.

An adaptive opening angle of 0.5 (see Dehnen 2000 for more details) was used. The maximum allowed tree depth was 50, and cells in the tree were split when they contained more than eight particles. The C version of the code released in 2002 October was tested here.

Table 1 outlines the features implemented by each of the methods.

2.3. *Processor and Compiler*

All simulations were run serially and in double precision on one of a cluster of 20 identical 600 MHz DEC Alpha EV56's with 64 megabytes of RAM and 256 megabytes of swap space, running Red Hat Linux 7.1. Fortran codes (SyMBA, RADAU via the Mercury package) were compiled with the Compaq Fortran 90 compiler X1.1.1-1684; C codes (Partree and the W-H code, as well as DPMTA, FALCON, and their driver) were compiled with the Compaq C compiler, version 6.2-506. All codes were compiled with the same optimizations. Clearly, some codes may perform somewhat better or worse with other processor-compiler combinations; our choice is intended simply to provide a representative straw-man system as a basis for comparison.

2.4. *Simulated Systems*

The test system used in these simulations is a 2.67 $M_\oplus$ disk of material spread with surface density $\propto r^{-1.5}$ between 0.3 and 2 AU around the Sun. The disk particles, all of equal mass, have their initial eccentricity e and inclination i chosen randomly from uniform distributions such that $e \in [0, 0.01]$ and $i \in [0^\circ, 0\overset{\circ}{.}5]$. The other angular elements were chosen

TABLE 1
Algorithms Tested

Method	Softening?	Fixed *dt*?	Approaches?	Tree?	FMM?
DPMTA	Yes	Yes	No	Yes	Yes
FALCON	Yes	Yes	No	Yes	Yes
Partree	Yes	Yes	No	Yes	No
RADAU	No	No	Yes	No	No
Standard W-H	Yes	Yes	No	No	No
SyMBA	No	No	Yes	No	No

Note.—A comparison of the different methods, and whether they include softening, have a fixed time step, handle close approaches in detail, and compute the far field using aspects of tree and/or FMM methods.

randomly from the uniform distribution $[0, 2\pi)$. This disk is based on those of Chambers (2001), who used similar disks of 153–158 bodies to study terrestrial planet formation. The number and masses of particles in the disk used here are varied, but the total disk mass is constant for all simulations.

Though chosen to model terrestrial planet formation, cold solid-particle disks are relevant to other areas of solar system research, for example, the Kuiper and asteroid belts, giant planet formation, extrasolar planet formation, and planetary rings. Realistic systems may involve the presence of larger bodies (e.g., protoplanetary cores, gas giant planets, satellites) within or near an otherwise smooth disk. However, approximate force algorithms tend to do less well when the mass distribution is highly asymmetric. Since the number of larger particles (n) will be smaller than the number of small ones ($\sim N$), it makes sense to handle large bodies separately by a direct calculation, which will only be $O(nN)$. There are any number of ways of doing so, including the use of massless test particles. Only the most demanding case will be examined here, that in which all mutual particle interactions are to be considered.

Our purpose here is not to conduct large-scale simulations per se, so the length of our integrations is limited in order to efficiently examine a variety of codes and particle numbers. For the first suite of simulations, designed to determine the scaling of the computing time with N, the total number of particles varies from 10^2 to 10^5 but the total simulation time is only 100 years. The time step is chosen to be 6 days, giving a minimal 10 steps per orbit for the innermost particles, and a total of 6088 steps overall.

The second suite of simulations are longer and examine energy and angular momentum conservation for the different fast approximate algorithms over time. These simulations run 100 times longer (10^4 yr) with 10^3 particles each. Such integrations are too short to be used in themselves to study either planet formation (a process that takes tens or hundreds of millions of years) or the error-induced diffusion of the Hamiltonian system away from its manifold. Rather, they are designed purely to examine as much as is practical the accumulation of errors for the various algorithms employed. This work is intended to provide an introductory look at the performance of these methods, not a definitive statement as to their suitability for all large-N systems under all conditions.

2.5. *Parallel Computing*

Parallel computing, with its potential for significant speed advantages, is likely to become important in numerical studies of the solar system, though it has yet to make much of an impact. Of the algorithms tested here only DPMTA and Partree, neither of which was designed for solar system studies, have parallel implementations. Only the serial versions were used in these tests.

Parallel codes are typically more sensitive to the details of the system they are being run on. For example, communication time may dominate over processing time on clusters with slow networks. As a result, timing parallel codes is less straightforward than for serial codes and will be left for future work.

3. RESULTS

3.1. *Speed*

Since CPU time scales simply with the length of the simulation (unlike the error), short, 100 yr, simulations are sufficient to compare the speed of the algorithms. The time taken for these simulations as a function of the number of particles N is shown in Figure 1. The simulations with the largest N are impractical to rerun multiple times. However, the shorter ones (up to $N = 10^3$) were run 10 times for each of the algorithms, and the median time is shown. The exception was Partree, for which the standard deviation was determined from the timing of subsamples (in this case, the timing of individual time steps) within a single integration run. The execution times have standard deviations of typically 1%–3%, though sometimes

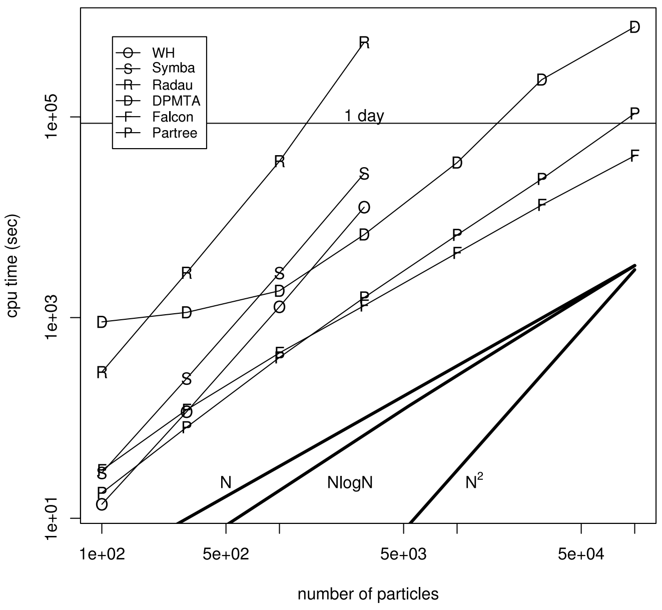

Fig. 1.—CPU time used as a function of the particle number N for the various algorithms tested for 100 yr simulations. The heavy lines indicate the slopes expected of methods with the indicated dependencies on N.

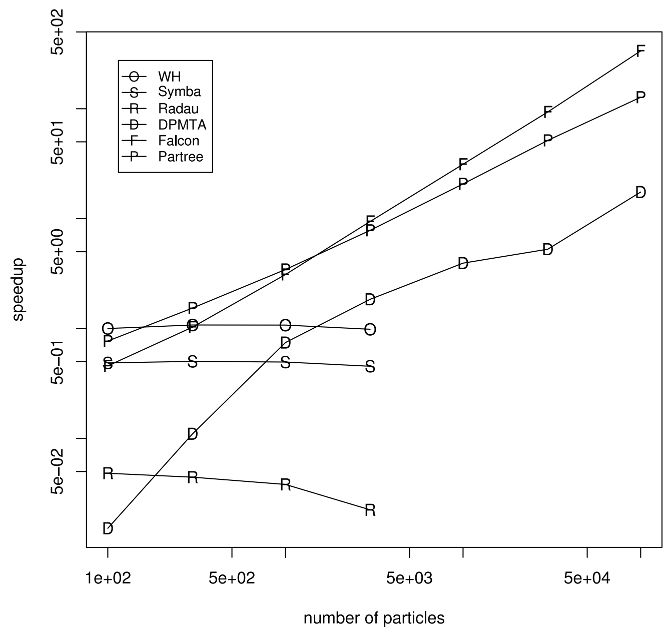

Fig. 2.—Ratio of the execution time for the standard W-H method to that for the other algorithms, or the "speedup." Execution times beyond $N = 3000$ for the W-H method are extrapolated from low N-values.

larger. These variations are due to overhead associated with the multiuser environment of the computer cluster, which was shared with other users during our runs.

Figure 2 shows the "speedup," the CPU time relative to one that scales strictly as N^2, and normalized to unity for the standard method at $N = 10^2$. All the direct codes show the expected near-$O(N^2)$ growth in the computing time as the particle number is increased.

SyMBA takes about twice as long as standard W-H, reflecting the extra overhead involved in handling encounters symplectically. SyMBA and RADAU handle a fundamentally harder problem than the other algorithms, for which close encounters have been softened away. RADAU is the slowest by about a factor of 20, as expected of this high-order nonsymplectic method.

FALCON is the fastest algorithm, with a speedup of 330 at $N = 10^5$. Partree has a speedup of 125, and DPMTA 17, at $N = 10^5$. Clearly, this affords these methods a significant advantage over direct methods at large N. Even at relatively small N, FALCON and Partree are as fast as direct methods. The break-even point for the approximate methods is $N \sim 10^2$–10^3. The CPU time per step (in milliseconds) and the CPU time (in days) required for a million-year simulation at $N = 10^2$ and at $N = 10^3$ are in Tables 2 and 3. The relative speeds of these algorithms vary quite a lot even going from $N = 10^2$ to $N = 10^3$, owing to their different sensitivities to N. Though obviously specific to the computer used here, the results provide insight into the current limits on solar system integrations at large N.

The slopes X of the lines in Figure 1 are given in Table 4, indicating that the computational effort, in this implementation rather than in theory, grows as N^X. The slopes are consistent with expectations, with a few minor variations. RADAU does worse than N^2, but it takes a variable time step that one expects to decrease, and hence increase the computation time, as the particle density increases.

DPMTA varies substantially from linear in Figure 1, being very sensitive to the maximum allowed tree depth (Rankin 1999). This method does have an execution time that, in tests at low particle numbers, grows slowly with N. However, this behavior does not persist to large N, where its slope becomes comparable to that of the tree methods.

The FALCON algorithm both is the fastest overall and shows the slowest growth with N, even at $N = 10^5$, growing only slightly faster than linear. Though all the approximate methods outperform direct ones at large N, FALCON has the shortest execution time by a significant margin, a factor of 2 or more.

3.2. *Accuracy*

Figures 3 and 4 show the relative error in the total energy E and the angular momentum $\boldsymbol{L}$ at the end of the 100 yr simulations. More precisely, the average value of the error over the final 10 yr is shown, in order to mitigate the effects of benign energy oscillations on the result.

The approximate codes are all competitive with the direct codes in terms of conservation of energy. The nonsymplectic,

TABLE 2

Simulation Results: $N = 10^2$

Method	CPU per Step (ms)	$t_{10^6\ \mathrm{yr}}$ (days)
Standard W-H	2.26 ± 0.02	1.60 ± 0.01
Partree	2.95 ± 0.44	2.08 ± 0.31
SyMBA	4.65 ± 0.03	3.28 ± 0.02
FALCON	4.95 ± 0.15	3.48 ± 0.11
RADAU	47.0 ± 5.1	33.1 ± 3.6
DPMTA	149 ± 2.7	104.9 ± 1.9

Note.—The median CPU time per step for the 100-particle disk simulations, as well as the extrapolated time to complete a 10^6 yr integration. Algorithms are ranked from fastest to slowest.

TABLE 3

Simulation Results: $N = 10^3$

Method	CPU per Step (ms)	$t_{10^6\ \mathrm{yr}}$ (days)
Partree	65.8 ± 8.0	46 ± 5.6
FALCON	73.0 ± 3.6	51 ± 2.5
Standard W-H	210.6 ± 3.2	148 ± 2.2
DPMTA	303.8 ± 4.9	214 ± 3.4
SyMBA	456.6 ± 20	321 ± 14
RADAU	5935 ± 50	4182 ± 35

Note.—The median CPU time per step for the 1000-particle disk simulations, as well as the extrapolated time to complete a 10^6 yr integration. Algorithms are ranked from fastest to slowest. Note that the ranking has changed appreciably from Table 2.

TABLE 4

Fits to $N = 10^2$–10^5 Simulation Results

Method	Slope (X)	Intercept (Y)
FALCON	1.0342 ± 0.0199	−0.5087 ± 0.0723
DPMTA	1.0529 ± 0.1132	+0.4560 ± 0.4109
Partree	1.2509 ± 0.0155	−1.1942 ± 0.0562
Standard W-H	2.0041 ± 0.0223	−2.8863 ± 0.0624
SyMBA	2.0195 ± 0.0169	−2.5992 ± 0.0473
RADAU	2.2143 ± 0.0662	−2.0130 ± 0.1850

Note.—Least-squares fitted slopes and intercepts to the CPU times plotted in Fig. 1, so that $\log t \sim X \log N + Y$.

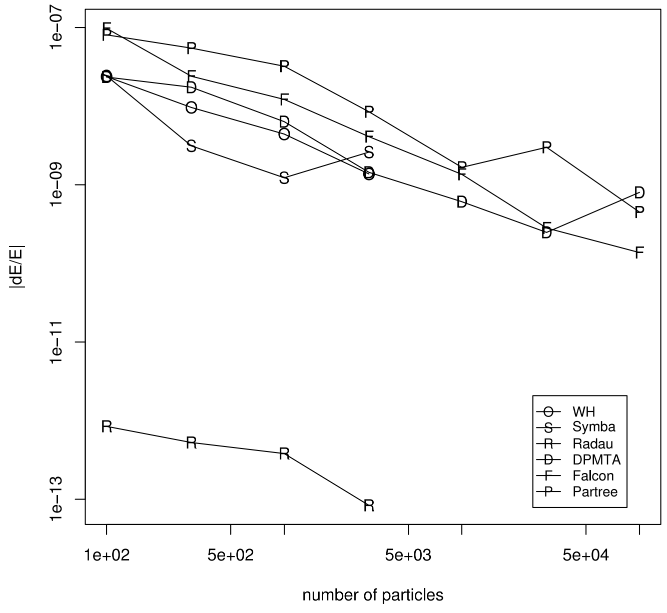

FIG. 3.—Relative error in the energy at the end of the short (100 yr) simulations as a function of N.

variable–step-size RADAU algorithm outperforms all others by about 3 orders of magnitude or more. However, the real test of energy conservation will come in the longer runs. Symplectic methods have appreciable but bounded long-term energy errors; this may be disrupted in the long term by inaccuracies in the force calculations.

The approximate methods perform poorly in terms of conservation of angular momentum $\boldsymbol{L}$, relative to direct ones. This is not simply an effect of softening, as the standard W-H code also shows very good angular momentum conservation. Any symplectic algorithm should, by construction, conserve angular momentum to machine precision. FALCON, though it conserves L less well than direct methods, does better than other approximate methods.

We note here again that the SyMBA and RADAU codes undergo a more rigorous test, having to deal with close approaches in detail (i.e., no softening): for example, they handle about 1200 close approaches over 100 years for the 3000-particle simulations.

The relative error is generally seen to decrease with increasing N. This effect can be related to the summation of larger numbers of smaller errors, with a consequent reduction in their total. A full treatment is complex, but the Plummer softening used here produces behavior that closely matches the $N^{-0.73}$ to $N^{-0.77}$ dependence found for the energy error by Athanassoula et al. (2000) and Dehnen (2001), respectively. FALCON's adaptive softening should allow a slightly sharper decrease ($\sim N^{-0.8}$). This is a small difference, but Figure 3 does show a somewhat larger overall drop in FALCON's energy error than the other methods'.

A plot of error in E versus that in L for the all simulations performed is presented in Figure 5. Each point represents one of the simulations with N in the range 10^2–10^5. Though obtained after simulations of only 100 years, the plot provides a general feel for the capabilities of these algorithms. Their accuracy will be examined in more detail with the longer term simulations presented below.

3.2.1. *Long Integrations*

Figures 6 and 7 show the error evolution over 10^4 yr for a 1000-particle disk for the different methods, with the exception of RADAU, which because of its slowness was only run for 2000 years of simulated time. The runs with standard W-H and FALCON were extended to 10^5 and 10^6 yr, respectively, to better observe the long-term error evolution.

The approximate methods show error growth comparable to that of traditional methods over the entire simulation. The energy errors of the simulations performed with DPMTA and FALCON match those of the direct methods quite closely. However, we note that symplectic integrators display an inherent oscillation in the energy, which can be reduced through the use of symplectic correctors (Saha & Tremaine 1992; Wisdom, Holman, & Touma 1996). This effect (which is not

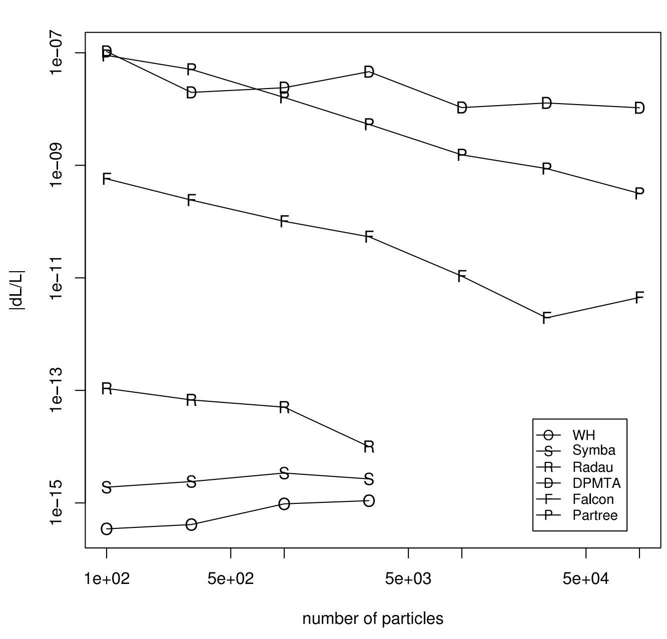

FIG. 4.—Same as Fig 3, but for the angular momentum

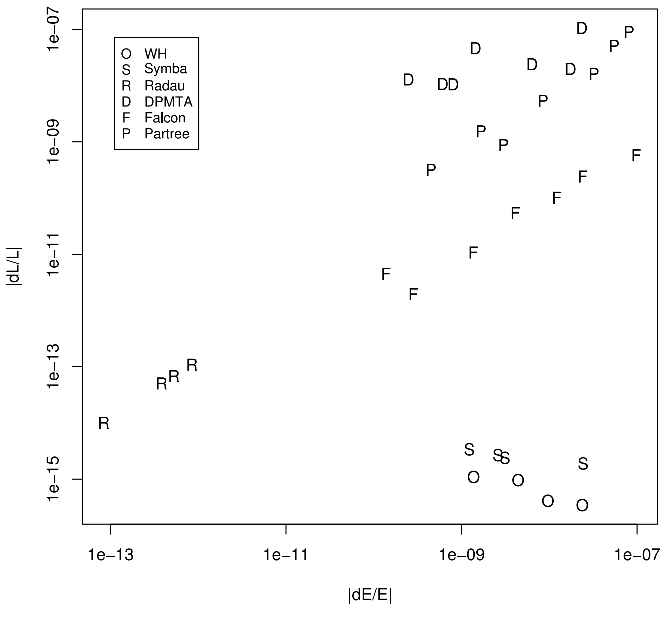

FIG. 5.—Relative energy and angular momentum errors for the short simulations. Each point, indicated by a letter, represents one value of N.

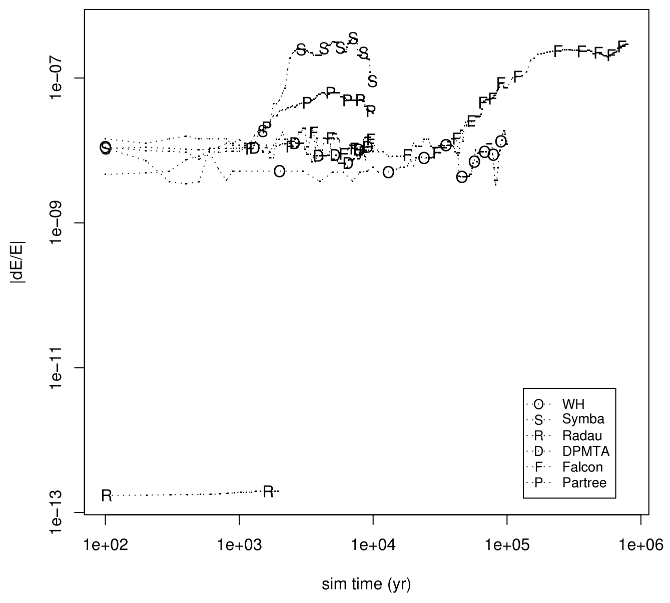

Fig. 6.—Relative error in the energy over the long (10^4 yr or more) simulations of the $N = 10^3$ disk. The data have been boxcar-smoothed for clarity and to bring to light the underlying trends.

usually called the "error," this term being reserved for any secular trend in the average value of the energy) sets a limit on the usefulness of the energy as a measure of accuracy. The value of $|dE/E|$ for the W-H code effectively measures the size of this oscillation. Thus, two similar curves on this plot do not necessarily indicate that the algorithms conserve energy "equally well," only that any secular energy drift these methods might suffer is much less than the energy oscillation. Nonetheless, the similarity of the $|dE/E|$ curve to that from the direct method is a necessary, if not sufficient, condition for an approximate force algorithm to be considered for use.

As for angular momentum (Fig. 7), the traditional methods substantially outperform the approximate ones. Orders of magnitude separate even the best fast methods from the $O(N^2)$ ones. Despite this fact, the error growth curves are well behaved. In particular, their slopes are otherwise often close to those of the standard methods. Thus, though certainly outperformed accuracy-wise by the traditional methods in both E and L, the methods show a well-behaved error growth, which provides some confidence in their reliability, if only over a restricted span.

3.3. *Machine Precision*

DPMTA is the only approximate algorithm tested here that allows the force errors to be reduced nearly to round-off without reverting to an $O(N^2)$ calculation. All the algorithms allow the opening angle θ to be reduced severely, but this simply results in the forces' being computed by the direct method over large regions of the space (rather than just locally), effectively reverting to an $O(N^2)$ process.

However, DPMTA allows an arbitrary number of multipole terms to be used in the force calculation (FALCON always employs a fixed number of terms, up to the octupole) while remaining an $O(N)$ [or at least faster than $O(N^2)$; see § 2.2.2] process. FMM may actually do better than the direct method in terms of accuracy, as the latter involves the superposition of large numbers of partially canceling forces of varying directions and magnitudes, with an associated accumulation of round-off errors.

We investigate the effect of increasing the number of multipole terms M on the accuracy of the result provided by DPMTA. Our earlier tests only used $M = 4$ and yet were much slower than other approximate methods. As a result, we expect that the code will run even more slowly with more multipole terms. Nevertheless, the possibility of reaching machine accuracy with an algorithm for which the number of computations grows more slowly than $O(N^2)$ provides a tantalizing prospect.

In order to get directly at the force accuracies, we compute the force (or more precisely, the acceleration) errors. We take the direct N^2 calculation as our reference and examine the variation as a function of M for a single "kick" step for the $N = 10^3$, 10^4, and 10^5 disks. The number of digits of accuracy of the DPMTA acceleration is plotted in Figure 8 for opening angles $\theta = 0.3$, 0.5, and 0.7. Note that this value is based on the mean acceleration error over all particles and must be interpreted with some care. The distribution of errors is essentially bimodal (local field done to machine accuracy, far field done approximately by FMM), and an average may not characterize it particularly well. However, particles with very small net interparticle-induced accelerations necessarily have large relative errors, and more conservative metrics such as the worst case provide little information on the algorithm's real performance. Note that only a limited number of simulations were performed at $N = 10^5$, as the memory required for the deeper tree and large number of multipoles exceeded that of our test machine.

For a given θ, the force error is relatively insensitive to N and saturates at machine precision at the top of the graph. The smaller opening angles achieve better accuracy for a given number of multipoles, since they compute a relatively large region directly, using the multipole expansion only for the most distant regions of the disk. The two smallest disks achieve very close to the same accuracy for a given M, whereas the $N = 10^5$ disk does slightly poorer. This is likely the result of the additional tree levels (six in total) needed at $N = 10^5$, as opposed to the four used for both the smaller disks (§ 2.2.2). The addition of these deeper levels, needed to maintain the speed of the

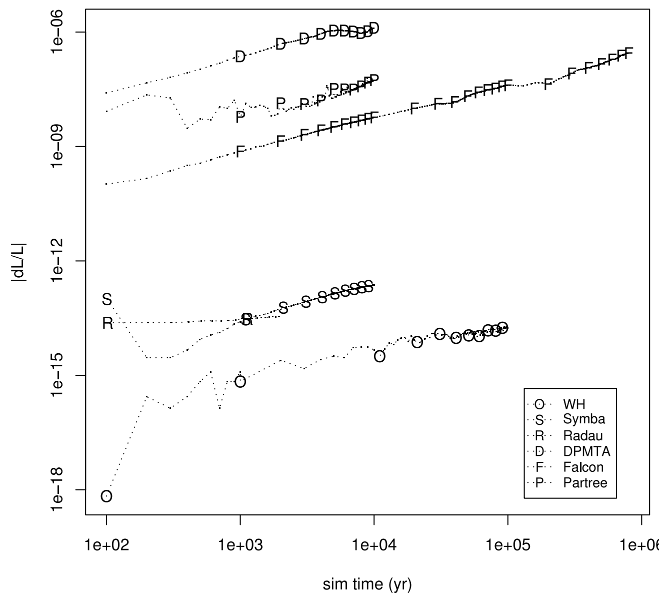

Fig. 7.—Relative error in the angular momentum over the long (10^4 yr or more) simulations of the $N = 10^3$ disk.

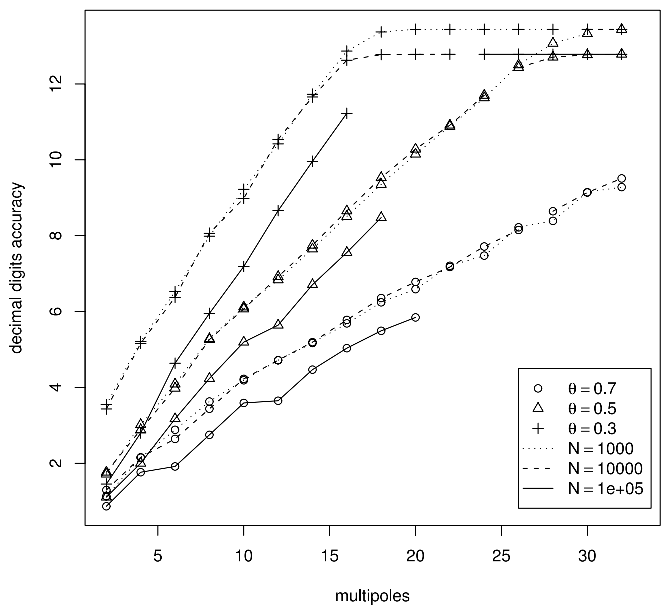

FIG. 8.—Average number of digits of accuracy in a given single force calculation for DPMTA for different opening angles θ and particle numbers N.

algorithm, results in the space's being decomposed into smaller cells, thus diminishing the region around each particle satisfying $l/D < \theta$ (i.e., diminishing the region in which direct force calculations are performed).

If the force error is near round-off, then we expect that DPMTA's accuracy will be comparable to those of the direct methods. This was confirmed by running some 100 yr simulations of the $N = 10^3$ disk at different θ-values (Fig. 9). The energy "error" levels off as a result of the small intrinsic energy oscillations inherent in the symplectic method, but the angular momentum error drops to round-off as the force errors do the same (Fig. 8), confirming that DPMTA is doing as well as the direct method.

Since equal force accuracies can be achieved for different combinations of θ and M, which is the fastest option? The answer depends on the value of N. DPMTA breaks the force calculation into an FMM part and a direct part, which scale differently with N. At lower θ, the direct calculation encompasses a larger volume, and the FMM calculation a smaller one, giving weight to the $O(N^2)$ term. Thus, we expect the CPU time to rise more sharply with N at smaller θ. In fact, tests show that while the CPU time required scales roughly as N for $\theta = 0.7$ and $\theta = 0.5$ (Table 4), it has increased to $\sim N^{1.5}$ at $\theta = 0.3$ for the disks tested here.

Figure 10 plots the CPU time required to generate forces with a given level of accuracy. For the smaller disk, smaller values of θ achieve the same accuracy faster because the multipole calculation, despite its good scaling with N, still takes the bulk of the time at $N = 10^3$. In this case, a smaller value of θ is to be preferred, but regardless of this value, DPMTA is slower than the direct method. For the larger disks, the differences between values of θ become less, though appreciable at higher accuracies. Thus, even at $N = 10^5$ DPMTA cannot yet produce double-precision forces faster than the direct method. Nevertheless, DPMTA continues to scale well and seems likely to surpass the direct method in the vicinity of $N = 10^6$.

Though FMM approaches hold great promise owing to their superior scaling with N, they remain too slow at the small particle numbers currently feasible to provide a good alternative to direct methods for high-accuracy applications, though with continuing improvements in algorithms and computer hardware, this gap will eventually close.

In a system where collisions are frequent and dissipation is important, errors in the handling of these processes are likely to exceed those in the integrator itself. In this case, one might be justified in using lower accuracy forces, with the accuracy parameters adjusted so that the error due to the integrator is less than or of order that in the other aspects of the simulation.

For large-N disks, DPMTA can provide reduced accuracy forces faster than the direct method. If, for example, only eight

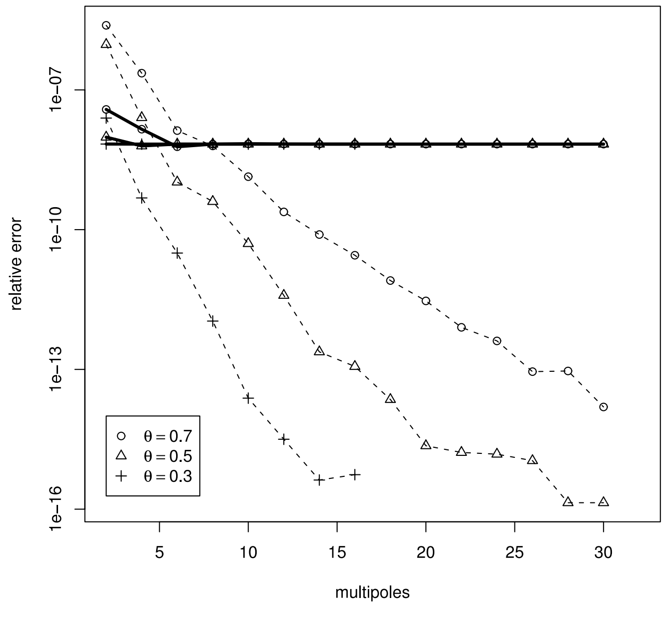

FIG. 9.—Relative error in E (*solid lines*) and L (*dashed lines*) after 100 yr as a function of M for the $N = 10^3$ disk with the DPMTA method. The energy error is flat as a result of the energy oscillations of the symplectic method, but the angular momentum errors go to round-off as the force errors (Fig. 8) do the same.

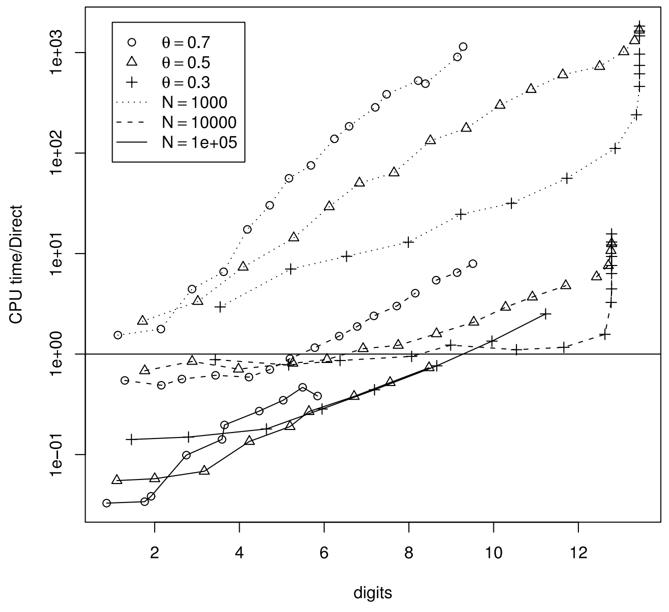

FIG. 10.—Fraction of CPU time, relative to a direct N^2 calculation, required to achieve a given mean force accuracy with DPMTA.

digits of accuracy (Fortran "REAL*4" or C "float") are deemed adequate, DPMTA breaks even with the direct method at $N = 10^4$ and does better at $N = 10^5$ (Fig. 10). DPMTA is expected to outperform the direct method by an even larger margin at lower accuracy and larger N. Thus, the tunable accuracy of FMM may make it an important tool for specific research niches, though one should be aware that other approximate methods may be competitive in these regimes.

4. CONCLUSIONS

Solar system integrations have to strike a balance between the accuracy of a simulation and the speed of the algorithm used. New approximate force algorithms can provide substantial increases in speed with a concomitant loss of accuracy. However, even $O(N^2)$ methods do not provide infinitely accurate forces: round-off ensures that. Nevertheless, computations performed in double precision are generally considered "good enough" for much solar system research.

It should be noted that if a real system of N' particles is modeled by a simulation of $N \ll N'$ bodies, the discreteness inherent in any N-body approach fundamentally limits the accuracy. It may be of limited usefulness to compute the force with extreme accuracy at discrete time intervals if the representation of the constituent population suffers inherent fluctuations produced by the (artificial) graininess of the simulated particle distribution.

On the other hand, one of the most compelling features of symplectic integrators is the belief that for a sufficiently small (and fixed) step size there usually exists a "surrogate Hamiltonian" for which a simulated particle's trajectory is an exact solution to within round-off. That is, the integrated system is the true solution of a Hamiltonian that differs from the true Hamiltonian by small terms, which often oscillate in time. Thus, a potentially fruitful approach would be to construct a Hamiltonian that is "sufficiently close" to the true interaction Hamiltonian and from which the derived forces can be quickly computed to machine accuracy using an order-N algorithm. Attempts to pursue this idea have met with limited success to date.

Approximate methods do have the advantage of allowing, through their input parameters, for the balance between speed and accuracy to be varied. Thus, these algorithms can be tuned to some extent to the problem being studied. Whether an approximate algorithm is appropriate for any given simulation is a question that can only be answered on a case-by-case basis. However, approximate force calculation methods can certainly provide substantial increases in speed, albeit at N larger than a few hundred. In some cases, only a modest increase in energy error, and a relatively larger one in angular momentum, results from the use of approximate algorithms. This larger error may make these methods unsuitable for some studies (e.g., long integrations of the planetary system). Methods such as the FMM algorithm allow for arbitrarily high accuracy to be achieved and are thus competitive with direct methods in terms of accuracy. They also scale very well with N and thus become more attractive at large particle numbers. However, their poor performance at the low values of N currently practicable make them currently unsuitable for solar system simulations, though this situation is likely to improve in the near future.

At large N, highly optimized algorithms such as FALCON provide for substantial speed increases with good accuracy, at least relative to other approximate methods. In certain regimes, the improved approximation to the actual particle distribution allowed by significantly larger values of N is likely to outweigh any losses due to reduced accuracy. Though this is a tantalizing possibility, the question "How accurate is accurate enough?" is not always easy to answer.

All currently feasible approximate methods, including FMM at below machine precision, show errors that, though not dramatically so, are larger than those of direct methods. As a result, the breaking of the symplecticity of the method becomes a serious concern. However, in environments where dissipation (e.g., due to collisions or drag) already plays some role, the loss of accuracy resulting from approximate force algorithms could well be dwarfed by uncertainty associated with these other processes. Though they must be used with some caution, approximate force algorithms are likely to provide a valuable tool for solar system research.

We would like to thank the authors of the various codes tested here for making the results of their efforts available to the wider community. This work has received ongoing support from the Natural Sciences and Engineering Research Council of Canada and NASA.

REFERENCES

Appel, A. 1985, SIAM J. Sci. Stat. Comput., 6, 85
Athanassoula, E., Fady, E., Lambert, J. C., & Bosma, A. 2000, MNRAS, 314, 475
Barnes, J., & Hut, P. 1986, Nature, 324, 446
Cappuzo-Dolcetta, R., & Miocchi, P. 1998, J. Comput. Phys., 143, 29
Chambers, J. E. 1999, MNRAS, 304, 793
———. 2001, Icarus, 152, 205
Chambers, J. E., & Migliorini, F. 1997, BAAS, 29, 1024
Cheng, H., Greengard, L., & Rohklin, V. 1999, J. Comput. Phys., 155, 468
Dehnen, W. 2000, ApJ, 536, L39
———. 2001, MNRAS, 324, 273
———. 2002, J. Comput. Phys., 179, 27
Dubinski, J. 1996, NewA, 1, 133
Duncan, M. J., Levison, H. F., & Lee, M. H. 1998, AJ, 116, 2067
Everhart, E. 1985, in IAU Colloq. 83, Dynamics of Comets, ed. A. Carusi & G. B. Valsecchi (Dordrecht: Reidel), 185
Gladman, B., & Duncan, M. 1990, AJ, 100, 1680
Greengard, L., & Rohklin, V. 1987, J. Comput. Phys., 73, 325
Jernigan, J. G., & Porter, D. H. 1989, ApJS, 71, 871
Kinoshita, H., Yoshida, H., & Nakai, H. 1991, Celest. Mech. Dyn. Astron., 50, 59
Levison, H. F., & Duncan, M. J. 1994, Icarus, 108, 18
Rankin, W. 1999, Ph.D. thesis, Duke Univ.
Richardson, D., Quinn, T., Stadel, J., & Lake, G. 2000, Icarus, 143, 45
Saha, P., & Tremaine, S. 1992, AJ, 104, 1633
Salmon, J. K. 1991, Ph.D. thesis, Caltech
Sellwood, J. A. 1997, in ASP Conf. Ser. 123, Computational Astrophysics, ed. D. A. Clarke & M. J. West (San Francisco: ASP), 215
Wisdom, J., & Holman, M. 1992, AJ, 102, 2022
Wisdom, J., Holman, M., & Touma, J. 1996, in Integration Algorithms and Classical Mechanics, ed. J. E. Marsden, W. G. Patrick, & W. F. Shadwick (Fields Inst. Commun. 10) (Providence: Am. Math. Soc.), 217