The Astronomical Journal, 151:64 (6pp), 2016 March doi:10.3847/0004-6256/151/3/64

A MULTIRATE VARIABLE-TIMESTEP ALGORITHM FOR *N*-BODY SOLAR SYSTEM SIMULATIONS WITH COLLISIONS

P. W. Sharp[1] and W. I. Newman[2]
[1] Department of Mathematics, University of Auckland, Private Bag 92019, Auckland, New Zealand; sharp@math.auckland.ac.nz
[2] Departments of Earth, Planetary, & Space Sciences, Physics & Astronomy, and Mathematics, University of California, Los Angeles, CA 90095, USA; win@ucla.edu
Received 2015 September 15; accepted 2016 January 12; published 2016 February 18

ABSTRACT

We present and analyze the performance of a new algorithm for performing accurate simulations of the solar system when collisions between massive bodies and test particles are permitted. The orbital motion of all bodies at all times is integrated using a high-order variable-timestep explicit Runge–Kutta Nyström (ERKN) method. The variation in the timestep ensures that the orbital motion of test particles on eccentric orbits or close to the Sun is calculated accurately. The test particles are divided into groups and each group is integrated using a different sequence of timesteps, giving a multirate algorithm. The ERKN method uses a high-order continuous approximation to the position and velocity when checking for collisions across a step. We give a summary of the extensive testing of our algorithm. In our largest simulation—that of the Sun, the planets Earth to Neptune and 100,000 test particles over 100 million years—the relative error in the energy after 100 million years was of the order of 10^{-11}.

Key words: celestial mechanics – methods: numerical – minor planets, asteroids: individual (MSC 2010: 65L06, 85-08, 70F10)

1. INTRODUCTION

State-of-the-art algorithms for simulations of the solar system when collisions between test particles and massive bodies are permitted either use fixed timesteps or a combination of fixed and varying timesteps. The algorithm in Grazier et al. (2005) uses a small fixed timestep during close encounters and a larger fixed timestep otherwise. The algorithms in the integrators `Mercury` (Chambers 1999), `Swifter` (Kaufmann 2005), `SyMBA` (Duncan et al. 1998; Levison & Duncan 2000; Levison et al. 2011), `QYMSYM` (Moore & Quillen 2011), and `GENGA` (Grimm & Stadel 2014) use a varying timestep or an equivalent during close encounters and a fixed timestep otherwise.

We present and test an algorithm that uses a high-order variable-timestep explicit Runge–Kutta Nyström (ERKN) pair for the integration of all bodies at all times. Our algorithm is intended for accurate simulations near limiting precision.

Using the same integration formula for all bodies at all times means that the implementation of the algorithm is noticeably simpler than existing algorithms. No heuristic is needed for switching to and from the scheme for handling close encounters. There are fewer fine tuning constants in the algorithm with a value that must be fixed, and fewer input values to the algorithm the user must specify. These features make it easier to perform a thorough investigation of the algorithm's performance.

Of greater importance is the use of a variable timestep. When the timestep is fixed, the accumulated error in the position and velocity of a test particle can increase rapidly with time if the eccentricity is above some specified threshold. In addition, a fixed-timestep method cannot accurately integrate the orbit of a test particle close to the Sun unless the timestep is very small.

To illustrate the difficulty with eccentric orbits, we performed 200,000 simulations, each of the Sun, the Jovian planets, and a different test particle. One half of the test particles had an initial eccentricity of 0.5, the other half 0.8. Each simulation was performed three times, once each with the variable-timestep order 12 ERKN pair of Dormand et al. (1987), the order 12 formulae from the pair used as a fixed-timestep method, and the fixed-timestep order 13 Störmer method of Grazier et al. (2005). The fixed-timestep ERKN method used a timestep that was the average of the timesteps when the same particle was integrated using the ERKN pair. As recommended in Grazier et al. (2005), we used a timestep of four days for the Störmer method.

Figure 1 contains smoothed graphs of the L_2 norm $\|E\|_2$ of the error in the position of the test particles after 30,000 days as a function of the initial a. The L_2 norm of the n-vector $\boldsymbol{w}$ is defined as

$$\|\boldsymbol{w}\|_2 \equiv \left[\sum_{i=1}^{n} w_i^2\right]^{1/2}. \tag{1}$$

The graphs were smoothed using the MATLAB function `filter`. The error for each particle was calculated as the difference between the numerical solution and an accurate reference solution obtained from an integration in quadruple precision. All particles within an activity sphere of a planet at the end of one or more integration steps were excluded from the graphs.

We observe from Figure 1 that $\|E\|_2$ for the variable-timestep ERKN pair E12VS varies little with a, and has similar values for $e = 0.5$ and $e = 0.8$. The error also varies little with a for the fixed-timestep Störmer method S13FS when $e = 0.5$. For $e = 0.8$, the graph of $\|E\|_2$ has a distinct elbow at $a \approx 8$. For larger a, $\|E\|_2$ varies little with a; for smaller a, $\|E\|_2$ increases rapidly as a decreases. E12FS fares worse than S13FS and has an elbow for $e = 0.5$ and 0.8.

We begin in Section 2 with the definition of the equations of motion, ERKN pairs, and their continuous approximations. We also give a summary of the scheme for the selection of the timestep for ERKN pairs. In Section 3, we present our multirate

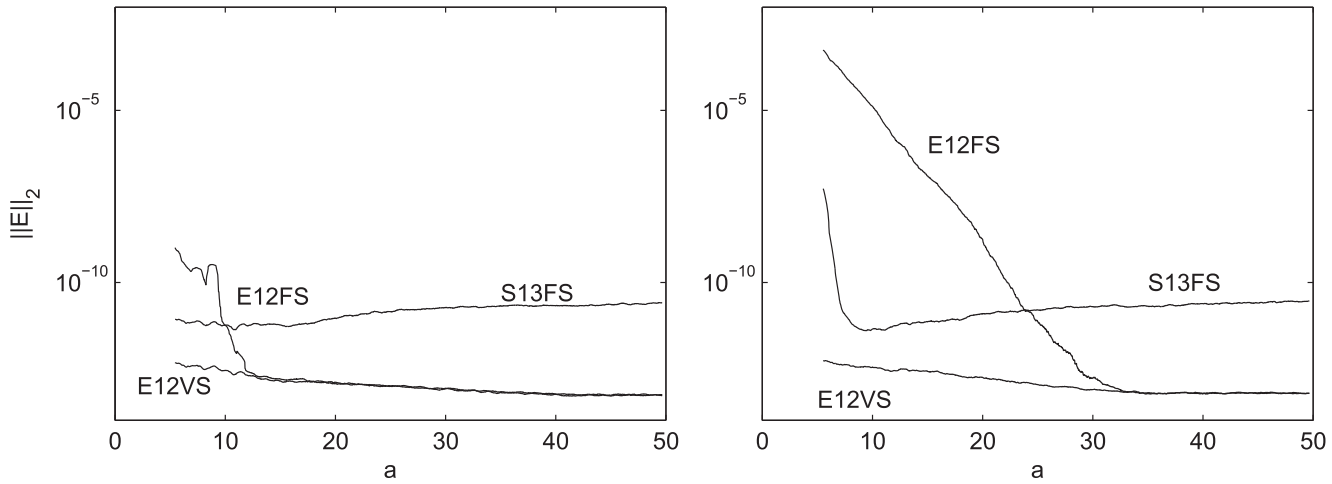

Figure 1. L_2 norm of the error in the position of the test particles after 30,000 days as a function of the initial a for initial eccentricities of $e = 0.5$ (left) and $e = 0.8$ (right). E12VS—the order 12 variable-timestep ERKN pair, E12FS—the order 12 fixed-timestep ERKN method, S13FS—the order 13 fixed-timestep Störmer method.

algorithm and follow this in Section 4 with a discussion of suitable input values to the algorithm. We give a summary of extensive numerical testing of our algorithm in Section 5 and end in Section 6 with a discussion of our results.

2. BACKGROUND

Let $\boldsymbol{R}_i(t)$, $i = 1,\ldots, N_p$ and $\boldsymbol{r}_j(t)$, $j = 1,\ldots, N_t$, be the position of the ith massive body and jth test particle, respectively, at time t, where N_p is the number of massive bodies and N_t is the number of test particles. The equations of motion for all $N_p + N_t$ bodies can be written as

$$\ddot{\boldsymbol{R}}_i = \sum_{k=1,k\neq i}^{N_p} \frac{\mu_k(\boldsymbol{R}_k - \boldsymbol{R}_i)}{\|\boldsymbol{R}_k - \boldsymbol{R}_i\|_2^3}, \quad i = 1,\ldots,N_p, \tag{2}$$

$$\ddot{\boldsymbol{r}}_j = \sum_{k=1}^{N_p} \frac{\mu_k(\boldsymbol{R}_k - \boldsymbol{r}_j)}{\|\boldsymbol{R}_k - \boldsymbol{r}_j\|_2^3}, \quad j = 1,\ldots,N_t, \tag{3}$$

where the dot operator denotes differentiation with respect to t and μ_l is G times the mass of the lth massive body.

Equations (2) and (3) combined with initial conditions form the initial value problem

$$\ddot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0, \ \dot{\boldsymbol{y}}(0) = \dot{\boldsymbol{y}}_0. \tag{4}$$

An explicit ERKN pair to integrate Equation (4) calculates the approximations $\boldsymbol{y}_i$, $\dot{\boldsymbol{y}}_i$, $\widehat{\boldsymbol{y}}_i$, and $\widehat{\dot{\boldsymbol{y}}}_i$ on each integration step. The values $\boldsymbol{y}_i$ and $\dot{\boldsymbol{y}}_i$ are order p approximations to $\boldsymbol{y}(t_i)$, and $\dot{\boldsymbol{y}}(t_i)$, respectively, and $\widehat{\boldsymbol{y}}_i$ and $\widehat{\dot{\boldsymbol{y}}}_i$ are order $q < p$ approximations to $\boldsymbol{y}(t_i)$ and $\dot{\boldsymbol{y}}(t_i)$, respectively. The four approximations are calculated from the updated formulae

$$\begin{aligned} \boldsymbol{y}_l &= \boldsymbol{y}_{l-1} + h\dot{\boldsymbol{y}}_{l-1} + h^2\sum_{j=1}^{s} b_j \boldsymbol{f}_j, \\ \dot{\boldsymbol{y}}_l &= \dot{\boldsymbol{y}}_{l-1} + h\sum_{j=1}^{s} b'_j \boldsymbol{f}_j, \end{aligned} \tag{5}$$

$$\begin{aligned} \widehat{\boldsymbol{y}}_l &= \boldsymbol{y}_{l-1} + h\dot{\boldsymbol{y}}_{l-1} + h^2\sum_{j=1}^{s} \widehat{b}_j \boldsymbol{f}_j, \\ \widehat{\dot{\boldsymbol{y}}}_l &= \dot{\boldsymbol{y}}_{l-1} + h\sum_{j=1}^{s} \widehat{b'}_j \boldsymbol{f}_j, \end{aligned} \tag{6}$$

where $h = t_l - t_{l-1}$ is the timestep, $\boldsymbol{f}_1 = \boldsymbol{f}(t_{l-1}, y_{l-1})$ and

$$\boldsymbol{f}_j = \boldsymbol{f}\left(t_{l-1} + c_j h, \boldsymbol{y}_{l-1} + c_j h\dot{\boldsymbol{y}}_{l-1} + h^2\sum_{k=1}^{j-1} a_{jk}\boldsymbol{f}_k\right),$$
$$j = 2,\ldots,s.$$

The integers s, p, q and the coefficients a, b, b', $\widehat{b}$, $\widehat{b'}$, c are chosen so the formulae in the pair have the required order and the pair has desirable properties. We refer to p as the order of the pair.

On each step with an ERKN pair, the timestep h is chosen so that the norm of the estimated local error satisfies the local error test. The local error in $\boldsymbol{y}(t_l)$ and $\dot{\boldsymbol{y}}(t_l)$ is estimated as $\widehat{\boldsymbol{y}}_l - \boldsymbol{y}_l$ and $\widehat{\dot{\boldsymbol{y}}}_l - \dot{\boldsymbol{y}}_l$ respectively. A step is accepted if the local error test $\text{le} \leqslant \text{TOL}$ is satisfied where $\text{le} = \max\{\|\widehat{\boldsymbol{y}}_l - \boldsymbol{y}_l\|_\infty, \|\widehat{\dot{\boldsymbol{y}}}_l - \dot{\boldsymbol{y}}_l\|_\infty\}$, and $\text{TOL} > 0$ is the local error tolerance supplied by the user. The norm $\|\cdot\|_\infty$ is the L_∞ norm and is defined for the n-vector $\boldsymbol{w}$ as

$$\|\boldsymbol{w}\|_\infty \equiv \max_{1\leqslant i\leqslant n}\{|w_i|\}. \tag{7}$$

Other norms such as the L_2 defined in Equation (1) can be employed. We use the L_∞ norm because it is less sensitive to decreases in the number of components of $\boldsymbol{y}$ and $\dot{\boldsymbol{y}}$ when test particles are removed from a simulation.

During an integration, the timestep h_{new} for a new step is calculated from the timestep h for the step just taken using the formula

$$h_{\text{new}} = \beta\left[\frac{\text{TOL}}{\text{le}}\right]^{1/(q+1)} h, \tag{8}$$

where $0 < \beta < 1$. The constant β acts as a safety factor that ensures most steps are accepted. We had $\beta = 0.9$ in all of our testing, other values could have been used. Formula Equation (8) is employed on all attempted steps except the first for

which the user supplies the timestep since the local error estimate is not available. On the last step of an interval, $h_{\rm new}$ is adjusted so that the integrator does not step past the right end point of the interval.

Our algorithm requires the continuous approximations $\boldsymbol{y}_l(t)$ and $\dot{\boldsymbol{y}}_l(t)$ to $\boldsymbol{y}(t)$ and $\dot{\boldsymbol{y}}(t)$, respectively, for $t \in [t_{l-1}, t_l]$. These approximations, commonly called interpolants, are of order $p^\star \leqslant p$ and defined as

$$\begin{aligned}\boldsymbol{y}_l(\tau) &= \boldsymbol{y}_{l-1} + \tau h \dot{\boldsymbol{y}}_{l-1} + h^2 \sum_{j=1}^{s^\star} \beta_j(\tau) \boldsymbol{f}_j,\\ \dot{\boldsymbol{y}}_l(\tau) &= \dot{\boldsymbol{y}}_{l-1} + h \sum_{j=1}^{s^\star} \gamma_j(\tau) \boldsymbol{f}_j,\end{aligned} \tag{9}$$

where $\tau = (t - t_{l-1})/h$, $s^\star \geqslant s$, and $\gamma_j(\tau)$ are not the derivatives of $\beta_j(\tau)$. As the notation implies, $\boldsymbol{f}_j$, $j = 1,\ldots,s$, are the same as the corresponding $\boldsymbol{f}_j$ in Equations (5) and (6). If $s^\star > s$, the values $\boldsymbol{f}_j$, $j = s+1,\ldots,s^\star$, must be calculated. Since the $\boldsymbol{f}_j$ are independent of τ, these extra values of $\boldsymbol{f}_j$ must be found just once per step. The interpolant can then be evaluated for any number of τ without calculating more $\boldsymbol{f}$ values.

Many ERKN pairs have been published. Dormand et al. (1987) presented a new order 12 pair ($p = 12$). They tested the pair and other high-order pairs on *Kepler*'s two-body problem with orbital eccentricities in the range of 0.1–0.9. Dormand et al. (1987) concluded that their new pair, which we denote by DEP12, was the most efficient pair near limiting precision. Sharp et al. (2013) presented new order 10 ($p = 10$) and 12 pairs. They tested the new pairs and DEP12 on four N-body models from solar system dynamics, as well as *Kepler*'s two-body problem with orbital eccentricities of 0.1, 0.5, and 0.9, and the PLEI N-body problem of Hairer et al. (1987). Sharp et al. (2013) concluded that their order 12 pair was on average 10% more efficient than DEP12 and that their order 10 pair could be more efficient than their order 12 pair on problems with rapid changes in the solution. Despite the greater efficiency of these new pairs, we used DEP12 because interpolants are not available for the new pairs.

The pair DEP12 has over 250 coefficients. Some of these coefficients require more than 100 digits to represent, and the complete set of coefficients spreads over at least five pages. The coefficients are available online as part of the integrator `rknint` (Brankin et al. 1989), see algorithm 670 at http://netlib.org/toms/.

3. ALGORITHM

In this section, we first give an overview of our algorithm and then discuss the details.

3.1. Overview

We begin a simulation by dividing the N_t test particles into N_g groups of equal size. If N_g does not divide N_t exactly, the first $N_g - 1$ groups are made as large as possible and of equal size; the N_gth group contains the remaining test particles. The selection of N_g is discussed later.

Next we attempt to integrate the massive bodies and the test particles in the first group from $t = 0$ to $t = \Delta t$ where Δt is specified by the user and is far greater than the typical timestep. The timestep h on each step of the integration is chosen so the local error test is satisfied. The timestep for the final step on the interval is further constrained to ensure $t = \Delta t$ at the end of the final step for the interval. After each step is completed, we remove those test particles in the group that have been ejected from the solar system or have collided with a massive body. The attempt to integrate the group to $t = \Delta t$ is stopped if all of the test particles in the group have been removed or an error condition has been raised in the integrator.

Once the first group has been processed, we attempt to integrate the second group from $t = 0$ to $t = \Delta t$. This includes a re-integration of the massive bodies since the orbits of the test particles depend on the position of the massive bodies. As with the first group, after each integration step, we remove test particles in the group that have been ejected from the solar system or have collided with a massive body. The remaining groups are treated in a similar way. After the last group has been integrated to $t = \Delta t$, we integrate all groups with at least one test particle one group at a time from $t = \Delta t$ to $t = 2\Delta t$. The simulation continues until either all test particles have been removed, the final value t_f of t is reached, or an error has occurred.

Each of the N_g groups of test particles is integrated independently of the other groups. This means each group will have its own sequence of timesteps, making our algorithm multirate. The different sequences of timesteps will lead to a divergence of the position of the same massive body in different groups. During a simulation we calculate the difference between the position of the massive bodies for the ith group, $i > 1$, and the first group, and use linear least squares to fit the power law αt^β to the norm of the difference. If the power law indicates that the difference is large enough to eliminate all resident information, we stop the simulation.

Even though different timestep sequences are used for different groups, the timestep for all steps and all groups is chosen so that the norm of the local error estimate is bounded above by the same fixed value (the local error tolerance). Hence, the divergence of the position for non-chaotic motion is slow. For the simulations in Section 5 below, a representative power law for the norm of the difference in a planet's position is $10^{-18.4} t^{1.67}$, where t is in days and the difference is in au. This norm after 10 million and 100 million years is 0.0037 au and 0.17 au, respectively, small compared to the typical semimajor axis of test particles.

3.2. Details

We use up to three conditions that a test particle must satisfy before it is regarded as ejected from the solar system. These conditions include that the heliocentric distance of the test particle is at least $R_{\rm ej}$, the test particle is unbound from the solar system, and it is on an outward bound trajectory.

After establishing that a test particle has not been ejected, we check if the test particle collided with the ith massive body, $i = 1,\ldots,N_p$, using the steps described below. In the description, $d_{ij}^2(t)$ is the square of the distance between the ith massive body and the jth particle at time t, and $R_{c,i}^2$ is square of the collision sphere radius for the ith massive body. We use d_{ij}^2 and not d_{ij} to reduce the number of square roots. There are at least two choices for the collision radius: the physical radius or the radius of the gravitational cross-section of Safronov & Zvjagina (1969).

We first check if the test particle is inside the collision sphere for the ith massive body at the end of the step i.e., if $d_{ij}^2(t_l) \leqslant R_{c,i}^2$. If it is, we remove the particle and go to the next particle. If the inequality is not satisfied, we check if the particle would have hit the ith massive body during the step. To do this, we first check if the inequalities

$$\frac{d\,[d_{ij}^2(t_{l-1})]}{dt} \equiv 2(\boldsymbol{R}_i(t_{l-1}) - \boldsymbol{r}_j(t_{l-1})) \cdot (\dot{\boldsymbol{R}}_i(t_{l-1}) - \dot{\boldsymbol{r}}_j(t_{l-1})) < 0, \quad (10)$$

$$\frac{d\,[d_{ij}^2(t_l)]}{dt} \equiv 2(\boldsymbol{R}_i(t_l) - \boldsymbol{r}_j(t_l)) \cdot (\dot{\boldsymbol{R}}_i(t_l) - \dot{\boldsymbol{r}}_j(t_l)) > 0, \quad (11)$$

are both satisfied. If they are not, the distance between the test particle and the massive body did not have a local minimum on the integration step, and we move on to the next massive body or test particle.

If the inequalities, Equations (10) and (11), are satisfied, we check the inequality

$$d_{ij}^2(t_{l-1}) + h\frac{d\,[d_{ij}^2(t_{l-1})]}{dt} \leqslant R_{c,i}^2. \quad (12)$$

The left side of Equation (12) is a lower bound on the square of the distance between the particle and the ith massive body when the distance has a local minimum on the step.

If Equation (12) is not satisfied, the simulation moves on to the next massive body or test particle. If Equation (12) is satisfied, we find the minimum distance between the massive body and the test particle on the step by solving

$$\frac{d\,[d_{ij}^2(t)]}{dt} = 0, \quad (13)$$

using an iterative method. The test particle is removed if the square of the minimum distance is no larger than $R_{c,i}^2$. To evaluate $d\,[d_{ij}^2(t)]/dt$, we use the interpolants Equation (9) to approximate the components of the position and velocity of the massive body and then form $d\,[d_{ij}^2(t)]/dt$. If $s^\star > s$, which it is for our algorithm, we calculate $\boldsymbol{f}_j$, $j = s+1, \ldots, s^\star$, at the start of the iterations for the current test particle.

We considered many possible iterative schemes and chose Krogh's `zero`. This method starts with a bracket on the root sought, and then uses linear interpolation for the first iteration and quadratic interpolation for subsequent iterations. The method employs heuristics that reduce the number of function evaluations. Fortran and C implementations of `zero` are available at http://netlib.org/math/index.html as part of the `math77` library. A user guide for `zero` is available at http://netlib.org/math/docpdf/ch08-01.pdf. Page three of the user guide lists the total number of function evaluations used by `zero` and 12 other methods on the 15 test problems of Alefeld et al. (1995). The best of the 12 other methods uses 26%–28% more function evaluations than `zero` for convergence tolerances of 10^{-7}, 10^{-10}, 10^{-15}, and 0 ("find as accurate a solution as possible"). We used `zero` with a convergence tolerance of zero in our algorithm.

If `zero` converges, we remove the test particle if it collided with the massive body and move on to the next test particle. We also move on to the next test particle if there was no collision. The routine `zero` has never failed to converge in our test runs. Nevertheless, failure is possible and if it occurs, we print an error message along with the position of the test particle and massive body, and go to the next test particle.

In addition to performing the integrations and checks for ejection and collisions, our algorithm does bookkeeping. This includes maintaining a count on the number of test particles left in the simulation and each group at time t, and updating arrays that hold information about the test particles. We also record information about the state of the integration for each group.

4. INPUT VALUES

The user-specified inputs to our algorithm are $p^\star$, ΔT, h_0, TOL, and N_g. We now discuss suitable values for the the first four inputs. The selection of a suitable value for N_g is discussed in the next section.

An interpolant of order $p^\star$ introduces an error of $O(h^{p^\star+1})$, which we call the interpolation error. We want this error to be insignificant compared to the error already present in the numerical solution at the end of the step. The standard values for $p^\star$ when solving general initial value ordinary differential equations is $p-1$ or p, see, for example, Baker et al. (1996). For either value of $p^\star$, the accumulated error in $\boldsymbol{y}_l$ and $\dot{\boldsymbol{y}}_l$ after many timesteps will be significantly larger than the interpolation error. Near the start of the integration, the accumulated error would not have had time to grow significantly with t. If $p^\star = p-1$, the interpolant error would then be significant. We therefore use $p^\star = p$. This choice means more CPU time is required to form and evaluate the interpolant than for $p^\star = p-1$. This extra work is a small fraction of the total work for a simulation because as we demonstrate in the next section that the interpolant is needed on a small percentage of the timesteps. The coefficients of the interpolant with $p^\star = p$ are available online in the file http://www.math.auckland.ac.nz/~sharp/rkncoeff.dat. This file is that referred to by Baker et al. (1996).

There are a wide range of acceptable values for ΔT. The only restriction is that it must be sufficiently large so that forcing the integrator to step exactly to ΔT, $2\Delta T$, $3\Delta T$, ..., has little effect on the integration. In our simulations, we typically use 1000 or 10,000 years for ΔT.

The scheme for selecting the timestep is very effective in adjusting the timestep to its optimal value in just a few steps. Hence the CPU time required to complete a long simulation is insensitive to the initial timestep h_0 and the same value can be used for all long simulations.

For the simulations we are interested in, there is a limited range of values for the local error tolerance TOL. The tolerance must be chosen so that the integrations are being done at or near limiting precision. Too large a value would mean the truncation error dominated the round-off error and we were not getting as much accuracy as we could. Too small a value would mean that the round-off error was dominating the truncation error. We experimented with TOL $= 10^{-i}$, $i = 12, 13, 14, 15$ and found that TOL $= 10^{-13}$ was a good compromise.

The main source of round-off error in the propagated solution at limiting precision is the addition in the update formula, Equations (5) and (6), that adds the h terms to $\boldsymbol{y}_{i-1}$ and $\dot{\boldsymbol{y}}_{i-1}$ to get the values at the end of a step. We used the standard technique of compensated summation, see, for example, Section 4.3 in Higham (2002), to reduce the round-off error and found no noticeable reduction in the total error on long simulations.

Table 1
Normalized CPU Times for the Simulations of the Sun, the Planets Earth through Neptune, and 1000 Test Particles Initially in the Jupiter–Saturn Zone

t_f (years)	N_g					
	1	2	5	10	20	50
10^3	1.40	1.17	1.03	1.00	1.03	1.16
10^4	1.38	1.10	1.02	1.00	1.03	1.17
10^5	1.51	1.15	1.03	1.00	1.01	1.12
10^6	1.44	1.11	1.00	1.03	1.04	1.24
10^7	1.00	1.03	1.00	1.08	–	–
10^8	1.00	1.13	1.34	1.82	–	–

Note. An entry of "–" means we stopped the simulation before completion because it was clear that the normalized time would be too large to change our conclusions.

5. NUMERICAL TESTS

In this section, we present a summary of our extensive testing of our multirate algorithm. In all tests, the massive bodies were the Sun and the planets, Earth through to Neptune. The test particles were randomly distributed in the Jupiter–Saturn zone at the start of a simulation. The units of time and distance were days and astronomical units respectively.

The aim of our first two simulations was to assess the performance of the scheme for varying the timestep. Both simulations had 250 test particles. The first simulation was over 100,000 years and the second over 100 million years.

During the simulation of 100,000 years, the timestep varied by four orders of magnitude from 3.13×10^{-3} days (approximately five minutes) to 35.4 days and the average was 23.9 days. Five test particles collided with Jupiter and one with Saturn. The timestep had a local minimum at each collision and the global minimum was one of these minima. The results for the simulation of 100 million years agreed well with those for the simulation of 100,000 years. In particular, the timestep varied from 1.96×10^{-3} to 36.1 days and the average was 26.1 days.

Our next set of simulations assessed the dependence of the CPU time on the number of groups N_g. For a given t_f, the value of N_g that minimizes the CPU time is a compromise between the following two competing factors. Since the same timestep is used for all test particles in a group, increasing N_g will reduce the number of test particles integrated with a small timestep when a test particle undergoes a close encounter with a massive body. This reduces the CPU time for a simulation. This is countered, partly or wholly, by the increase in the number of times the massive bodies are integrated (they are integrated once for each group). To gain insight about the optimal value for N_g, we performed simulations of 1000 test particles over $t_f = 10^i$ years, $i = 3, 4, \ldots, 8$, for $N_g = 1, 2, 5, 10, 20, 50$. For each t_f, we recorded the CPU time for each N_g and found the minimum $t_{\mathrm{CPU,min}}$ of these times. We then normalized the CPU times for the given N_g by dividing by $t_{\mathrm{CPU,min}}$. For example, when $t_f = 10^3$ years, the CPU times were 41.3 s ($N_g = 1$), 34.4 s (2), 30.3 s (5), 29.5 s (10), 30.3 s (20), and 34.2 s (50). We have $t_{\mathrm{CPU,min}} = 29.5$ s and the normalized times 1.40, 1.17, 1.03, 1.00, 1.03, and 1.16 (2dp).

The normalized CPU times are listed in Table 1. An entry of "–" means we stopped the simulation before it reached t_f because it was clear that the ratio would be too large to change our conclusions. We observe from Table 1 that for small t_f, $N_g = 10$ is optimal, although $N_g = 5$ or 20 are almost as good. As t_f increases, the optimal N_g decreases and is 1 when $t_f = 10^8$ years. The optimal N_g decreased because many of the test particles had been ejected or removed and the integration of the massive bodies required a larger fraction of the CPU time. Decreasing N_g decreases this fraction. For simulations of a large number of test particles, the optimal N_g can be estimated from an initial integration of a small number of test particles, such as 1000 as we used.

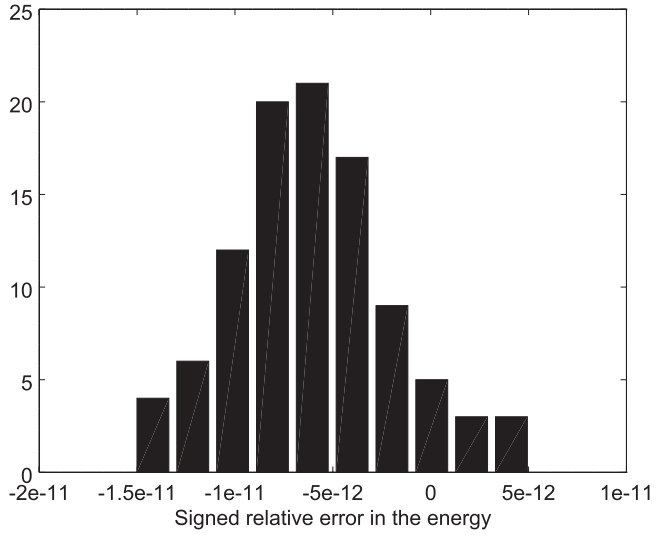

Figure 2. Histogram of the signed relative error in the energy at $t = 10^8$ years for 100 simulations each of 1000 test particles. Each data point is the signed relative error for one core.

Another aspect of using groups is deciding how to divide test particles into groups. One possibility is to use the initial orbits of the test particles to group those that are likely to have similar evolutionary histories. For example, test particles could be grouped according to their initial eccentricity and semimajor axis.

The last simulation we report on is of 100,000 test particles over 100 million years. We used 100 cores on the high-performance computer Pan at the University of Auckland's Centre for eResearch. Each core had 1000 test particles and a copy of the massive bodies. On each core, we used $N_g = 10$. We know from Table 1 that $N_g = 10$ is not optimal for $t_f = 10^8$ years. We used $N_g = 10$ because it provided a more thorough test of our algorithm than $N_g = 1$ did. For the same reason, we used the physical cross-section of the massive bodies and not their gravitational cross-section, and required the three conditions with $R_{\mathrm{ej}} = 50$ au stated in Section 3.2 to be satisfied before ejecting a test particle.

Across the 100 cores, the CPU time required to complete an integration varied from 2.58 to 3.42 days and had a mean of 3.05 days. The variation in CPU time was caused mostly by the differences in the orbital evolution and not by the timing uncertainty.

Figure 2 is a histogram of the signed relative error in the energy across the cores at the end of the simulation. We observe that the histogram is close to being symmetric. The mean of the signed relative error is -5.8×10^{-12}. We used linear least squares to fit the power law αt^{β} to the error for each core. The mean of β across the 100 cores was 0.68. Hence our method does not satisfy the optimal power law $t^{1/2}$ derived by Brouwer (1937). Despite this, the mean of the end point energy

error in our simulation is comparable to that of the Störmer method of Grazier et al. (2005), which does achieve the $t^{1/2}$ growth. We believe our method achieves comparable accuracy because the error on a single step is significantly smaller than that of the Störmer method.

By the end of the simulation, 94,093 test particles had been ejected from the solar system and 3591 had collided with a massive body. Of these collisions, 99.3% were detected at the end of a step. This percentage, being close to 100%, suggests the detection of collisions across a step could be omitted. Doing so would save little CPU time because the mean number of iterations performed by `zero` on each core was just 811.

6. DISCUSSION

We presented a variable-timestep algorithm for long, accurate simulations of the solar system when collisions between test particles and massive bodies are permitted. Varying the timestep meant the orbits of test particles close to the Sun or on eccentric orbits were calculated accurately, thus avoiding two potential difficulties with fixed-timestep algorithms. This gives more assurance about the validity of conclusions drawn from the results of a simulation. Our test results showed that the error in the energy was very small.

Our algorithm has many applications. One such application, as we illustrated in the previous section, is an investigation of the collisions between the terrestrial planets and small bodies. Horner & Jones (2008, 2009) investigated how effective Jupiter is at preventing asteroids and centaurs from hitting Earth. To enhance the impact rate on Earth, Horner and Jones set Earth's radius at one million kilometers. In our simulations of the previous section, we used the realistic value of 6370 kilometers (the Safronov & Zvjagina 1969 radius is often significantly larger than this value) and found that just one test particle hit Earth over 100 million years. This is too small a number to make any sensible conclusion and far more test particles, perhaps 10^6, would be needed. This number is feasible with our algorithm.

Our algorithm can also be used to perform statistical tests on simulation results. For example, Grazier et al. (2014) performed simulations with 2000 test particles to investigate the delivery of volatiles to the outer asteroid belt. The simulations differed in the mass used for Jupiter and the zone in which the test particles started. Our algorithm is fast enough that multiple sets of 2000 test particles for each mass and zone could be simulated and the information from these simulations used to assess the conclusions could be drawn from the original sample.

Although not needed for the simulations of Section 5, our method will accurately integrate close encounters between massive bodies. This is so because the scheme to select the timestep in an integration does not distinguish between massive bodies and test particles. Our method does not permit collisions between massive bodies. We believe our method can be extended using the interpolants and the solver `zero` to permit these collision.

The authors thank the referee for comments and acknowledge Fred Krogh's help with the iterative method `zero`. W.I.N. acknowledges support from the UCLA Academic Senate for this collaboration. The authors acknowledge the contribution of the NeSI high-performance computing facilities and the staff at the Centre for eResearch at the University of Auckland. New Zealand's national facilities are provided by the New Zealand eScience Infrastructure (NeSI) and funded jointly by NeSI's collaborator institutions and through the Ministry of Business, Innovation and Employments Infrastructure programme. http://www.nesi.org.nz. The authors also acknowledge the use of the computational servers in the Department of Mathematics at the University of Auckland.

REFERENCES

Alefeld, G. E., Potra, F. A., & Shi, Yixun 1995, ACM Trans. Math. Softw., 21, 327
Baker, T. S., Dormand, J. R., Gilmore, J. P., & Prince, P. J. 1996, ApNM, 22, 51
Brankin, R. W., Gladwell, I., Dormand, J. R., Prince, P. J., & Seward, W. L. 1989, ACM Trans. Math. Softw., 15, 31
Brouwer, D. 1937, AJ, 46, 149
Chambers, J. E. 1999, MNRAS, 304, 793
Dormand, J. R., El-Mikkway, M. E. A., & Prince, P. J. 1987, IJNA, 7, 423
Duncan, M. J., Levison, H. F., & Lee, M. H. 1998, AJ, 116, 2067
Grazier, K. R., Castillo-Rogez, J. C., & Sharp, P. W. 2014, Icar, 232, 13
Grazier, K. R., Newman, W. I., Hyman, J. M., Sharp, P. W., & Goldstein, D. J. 2005, Proc. of 12th Computational Techniques and Applications Conf. 46 ed. R. May, & A. J. Roberts, C786
Grimm, S. L., & Stadel, J. G. 2014, ApJ, 796, 23
Hairer, E., Nørsett, S. P, & Wanner, G. 1987, Solving Ordinary Differential Equations. I: Nonstiff Problems (Springer: Berlin)
Higham, N. J. 2002, Accuracy and Stability of Numerical Algorithms (2nd ed.; Philadelphia: SIAM)
Horner, J., & Jones, B. W. 2008, IJAsB, 7, 251
Horner, J., & Jones, B. W. 2009, IJAsB, 8, 75
Kaufmann, D. E. 2005, Swifter, http://www.boulder.swri.edu/swifter/
Levison, H. F., & Duncan, M. J. 2000, AJ, 120, 2117
Levison, H. F., Morbidelli, A., Tsiganis, K., Nesvorný, D., & Gomes, R. 2011, AJ, 142, 152
Moore, A., & Quillen, A. C. 2011, NewA, 16, 445
Safronov, V. S., & Zvjagina, E. V. 1969, Icar, 10, 109
Sharp, P. W., Qureshi, M. A., & Grazier, K. R. 2013, NuAlg, 62, 133