ASTR415: Problem Set #6

Curran D. Muhlberger
University of Maryland
(Dated: May 17, 2007)

Using existing implementations of the leapfrog and Runge-Kutta methods for solving coupled ordinary differential equations, several N-body systems were explored both quantitatively and qualitatively. First, a symmetric two-body problem was solved using both integrators to compare the results with the known analytic solution. Then, several many-body systems were generated and integrated with both solvers to explore the effects of softening.

In the case of the two-body problem, the leapfrog method did a better job of conserving energy than the Runge-Kutta method and resulted in phase diagrams that more closely resembled the exact solution. For the many-body systems, however, the leapfrog method produced strange results (such as the particles arranging themselves into a column) and failed to conserve energy. For $N = 500$ with the chosen initial conditions, a timestep of $h = 0.001$ was found to be necessary for the Runge-Kutta method to conserve energy over 100 time units (100000 steps). Finaly, the scaling of the PP method was explored and found to be less than the theoretical scaling of $O(N^2)$.

I. INTEGRATION OF THE TWO-BODY PROBLEM

The two-body problem can be reduced to a one-body problem where a particle of the reduced mass $\mu = \frac{m_1 m_2}{m_1+m_2}$ orbits a particle of the total mass $M = m_1 + m_2$ and whose position vector $\mathbf{r}$ corresponds to the displacement of the two masses from one another, $\mathbf{r}_1 - \mathbf{r}_2$. Since the one-body problem has an analytical solution for the geometry of the orbits, the two-body problem does as well. This makes it suitable for testing the N-body code, since the numerical results from the integrator can be compared with exact solutions.

Consider the case where the two bodies are both of unit mass, so $m_1 = m_2 = 1$. Choose units such that $G = 1$. The orbits are such that the particles are separated by unit distance at apoapsis. Choosing this as the initial configuration, at $t = 0$, $\mathbf{r}_1 = \langle -1/2, 0, 0\rangle$ and $\mathbf{r}_2 = \langle 1/2, 0, 0\rangle$. The initial velocities determine the eccentricity e of the orbit through the relations

$$\frac{1}{a} = \frac{2}{r} - \frac{v^2}{GM}$$

$$r_a = (1 + e)a$$

where $r_a = 1$ is the particles' separation at apoapsis. Once v is chosen, the initial velocity conditions become $\mathbf{v}_2 = \langle 0, -v/2, 0\rangle$ and $\mathbf{v}_2 = \langle 0, v/2, 0\rangle$.

For $e = 0.5$, $v = 1$. The orbits of both particles in such a system are plotted in figure 1, as integrated using both the leapfrog and Runge-Kutta methods. As the period of these orbits is 2.4 time units, each orbit requires 48 time steps using a step size of $h = 0.05$. Therefore, about 5000 steps are required to achieve 100 complete orbits. Reporting the results of the integration every 5 steps yields 1000 points with which to analyze the orbit over time.

From this data, the relative radial velocity v_r of the particles at each point in time can be computed and plotted against their separation distance r, producing the phase diagram shown in figure 2. Furthermore, the energy can be calculated according to

$$E = \frac{1}{2}v_1^2 + \frac{1}{2}v_2^2 - \frac{1}{r}$$

The energy of the system as calculated by both integrators is plotted against time in figure 3. Looking at both of these plots, it is clear that the Leapfrog method does a much better job at conserving energy than the Runge-Kutta method. However, the Leapfrog method is very poor at maintaining the positions of the particles, and they tend to precess rapidly, unlike the orbits produced by the Runge-Kutta method.

For $e = 0.9$, $v = \sqrt{1/5}$. The orbits of the particles in this system are plotted in figure 4 for both integrators. With an orbital period of 1.7 time units, each orbit requires 565 time steps using a step size of $h = 0.003$. Therefore, approximately 57000 steps are required to achieve 100 complete orbits, and an output frequency of once every 57 steps yields 1000 data points along these orbits.

The phase diagram for this system is shown in figure 5. Neither integrator can accurately reproduce the exact solution near periapsis, but the leapfrog method does a far better job at apoapsis than the Runge-Kutta method. The energy of the system is plotted against time in figure 6. Here the superior ability of the leapfrog method to conserve energy over the Runge-Kutta method is even more pronounced.

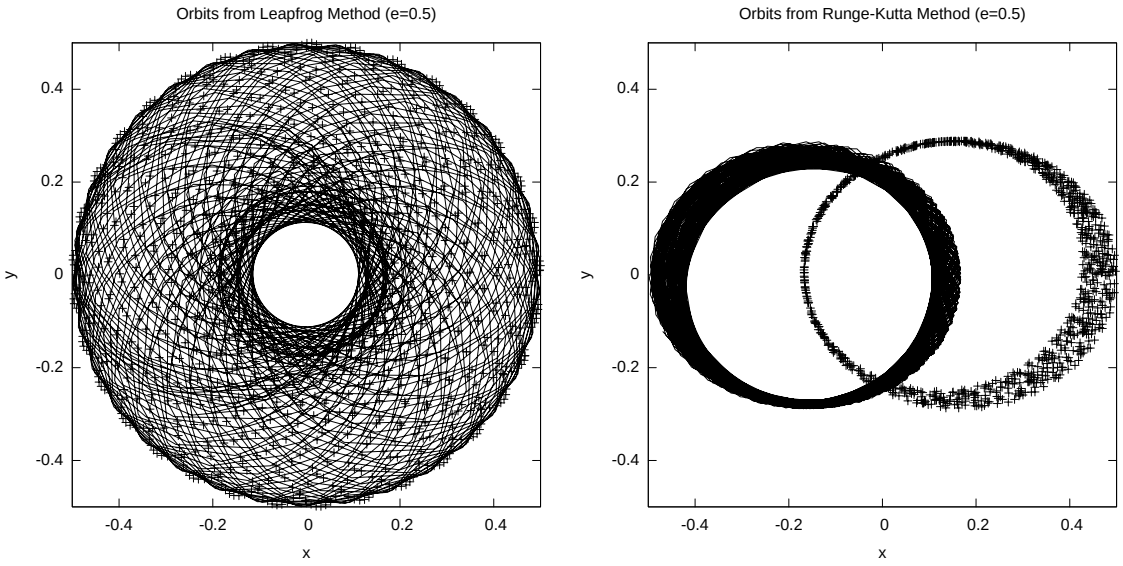

FIG. 1: Plots of the orbits for $e = 0.5$. The orbit of one particle is plotted with lines while the orbit of the other is plotted with crosses.

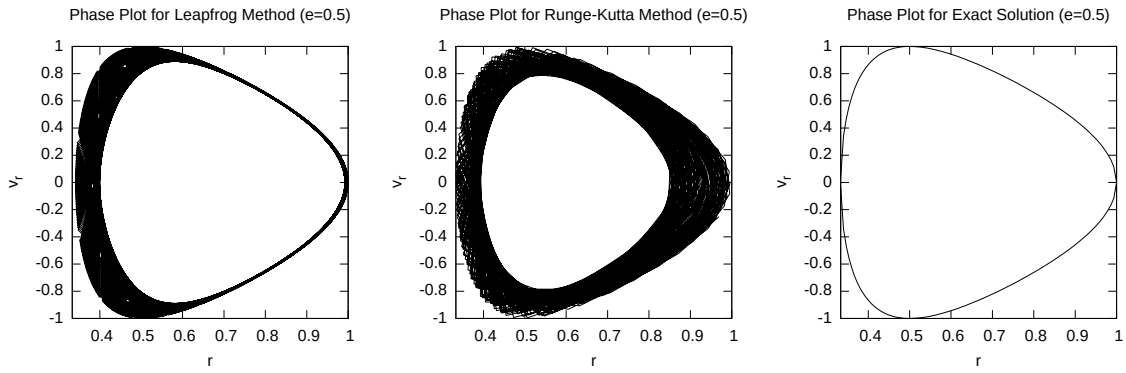

FIG. 2: Phase diagram for $e = 0.5$.

A. Solving Kepler's Equation for Radial Velocity

In figures 2 and 5, a phase diagram for the exact solution is given. Kepler's equation has no analytic solution expressing r as a function of t, but it does have an analytic solution for $\dot{r}$ in terms of r. This solution can be found as follows. Kepler's equation states that

$$M = E - e\sin(E)$$

where M is the mean anomaly, defined as $M = 2\pi t/P$, and E is the eccentric anomaly, defined as $\cos^{-1}(\frac{1-r/a}{e})$. Differentiating both sides with respect to t and solving for $\dot{r}$ yields

$$\dot{r} = \frac{\sqrt{-2(r-1)(re+r+e-1)}}{r}$$

When $e = 0.5$ this simplifies to

$$\frac{\sqrt{-3r^2+4r-1}}{r}$$

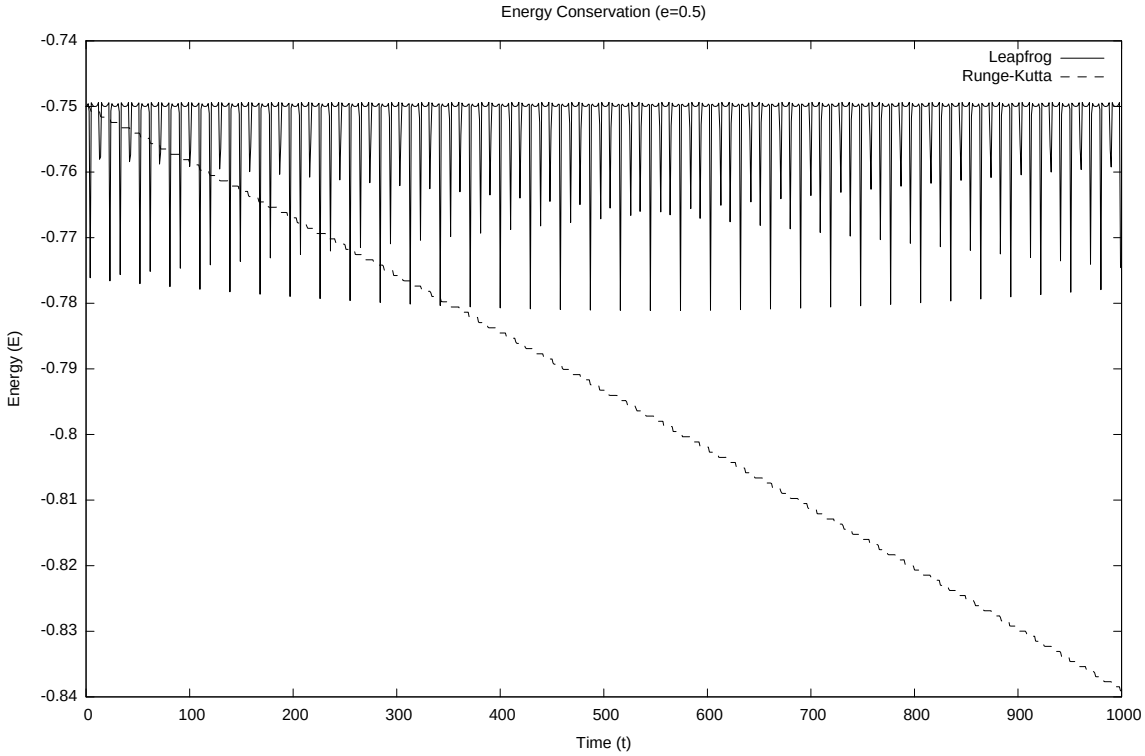

FIG. 3: Energy versus time for $e = 0.5$..

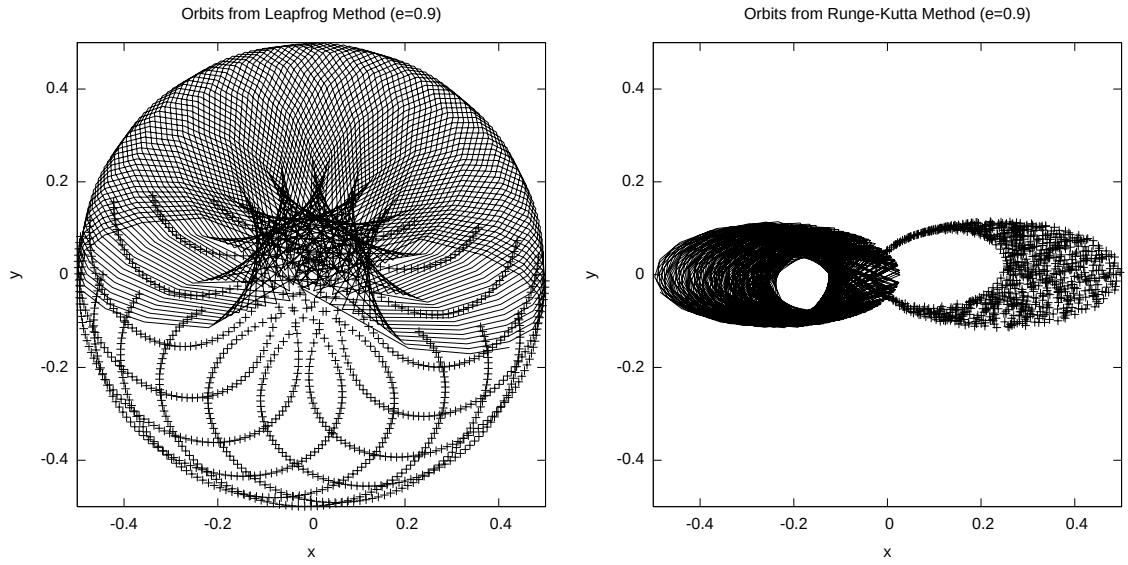

FIG. 4: Plots of the orbits for for $e = 0.9$. The orbit of one particle is plotted with lines while the orbit of the other is plotted with crosses.

and when $e = 0.9$, it simplifies to

$$\frac{\sqrt{-\frac{19}{5}r^2 + 4r - \frac{1}{5}}}{r}$$

These equations were used to make the phase diagrams for the exact solutions in order to compare them with those produced by the integrated solutions.

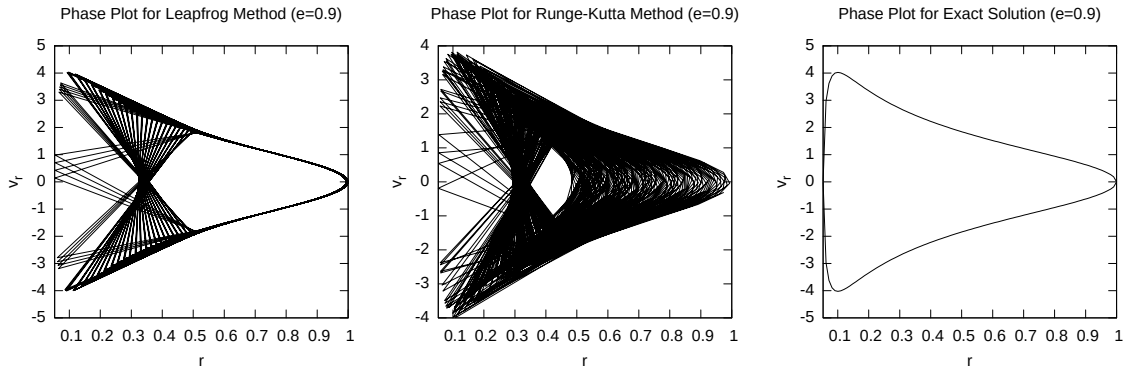

FIG. 5: Phase diagram for $e = 0.9$..

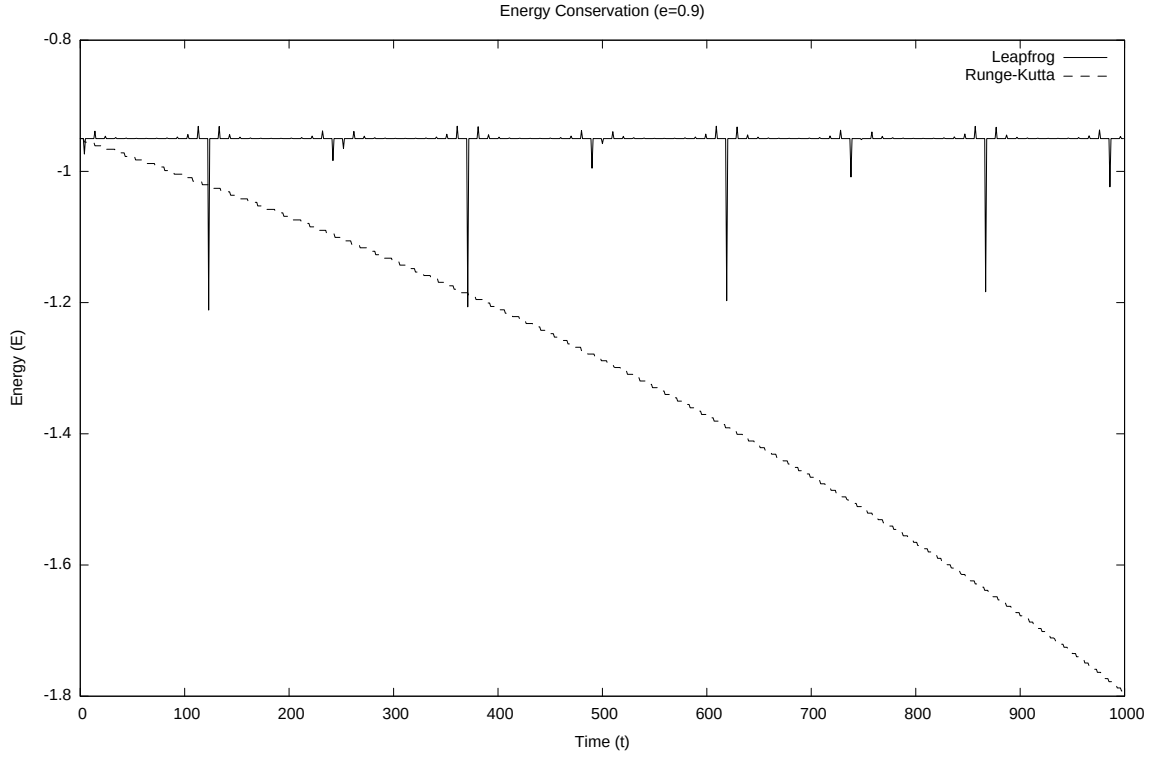

FIG. 6: Energy versus time for $e = 0.9$.

II. THE N-BODY PROBLEM

A. Initial Conditions

Several simulations were done using $N = 500$ and $N = 1000$ points distributed in a sphere with radius 100. The masses were distributed according to a Rayleigh distribution centered at either 1 or 2. The system was given a positive angular momentum along the z-axis by choosing initial velocities in the $x-y$ plane as if each particle were in a circular orbit about a particle of unit mass at the origin. The z velocities were chosen uniformly as small deviations from 0.

For the first two, low-precision, runs ($N = 1000$), which determined the effects of softening, the mass distribution was centered at 1 mass unit and the initial velocities were not scaled. The softening parameter was first set to 0 and then to 0.2, and results were obtained for both integrators. For subsequent, high-precision, runs ($N = 500$), the mass distribution was centered at 2 mass units, and the initial velocities were scaled by a factor of 10. The initial conditions used in the latter case are shown in figure 7.

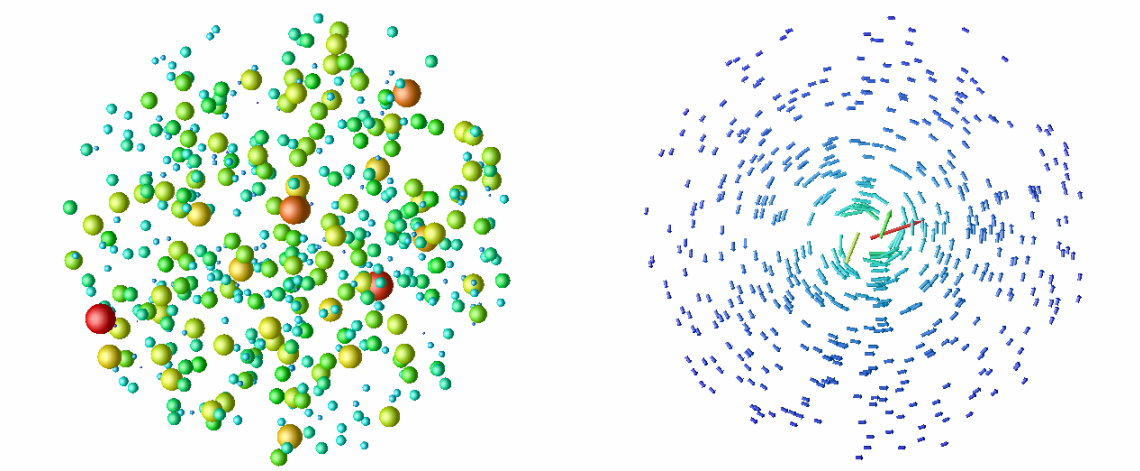

FIG. 7: Initial conditions used for the high-precision simulations ($N = 500$). On the left are the initial positions of the bodies with the glyphs colored and scaled according to mass. On the right are the initial velocities.

B. Qualitative Results

Using the Runge-Kutta method with a timestep of $h = 0.1$ and no softening, particles quickly begin to leave the system at high velocities due to close encounters, and the results of the simulation are clearly non-physical. Adding a softening parameter of 0.2 greatly reduces this effect, and the system quickly collapses in on itself before rebounding and sending some particles into a loose external shell while the others formed a dense interior core with a relatively high characteristic velocity.

Integrating a smaller system with a smaller timestep produced similar results, except that, as a result of the greater vorticity in the initial conditions, the particles first formed an annulus before collapsing in on themselves. In both cases the leapfrog method appeared to give unphysical results, with the particles forming a loose column. However, with the smaller timestep, both methods give similar results for early times.

C. Energy Conservation

The total energy of the system should be given by

$$E = \sum_{i=1}^{N} \left(\frac{1}{2} m_i v_i^2 - \sum_{j=i+1}^{N} \frac{m_i m_j}{r_{ij}} \right)$$

This represents the sum of the kinetic energy of each of the particles and the potential each particle contributes to those counted after it. Surprisingly, the leapfrog integrator was unable to conserve the energy of the system over 100 time units even for time steps as small as $h = 0.001$ (see figure 8). For large time steps, the Runge-Kutta integrator exhibited even larger deviations in energy, but as the time step was reduced, the energy remained stable until near the end of the simulation (see figure 9).

Given the leapfrog method's ability to conserve energy for smaller systems and the characteristic shape of its energy curves in all four simulations, it is possible that an error exists in either the energy computation or simulation code. However, the Runge-Kutta method seems to hold the energy steady for small time steps. For these initial conditions and on this time scale, a time step of $h = 0.001$ appears to be necessary to achieve meaningful results. Unfortunately, even for as few as $N = 500$ points, this takes several hours to compute on a modern processor.

D. Program Scalability

The PP method for solving the N-body problem is an $O(N^2)$ algorithm. However, the code appears to scale at an order smaller than 2. Random initial conditions were generated for N between 64 and 1024 in powers of 2, and the

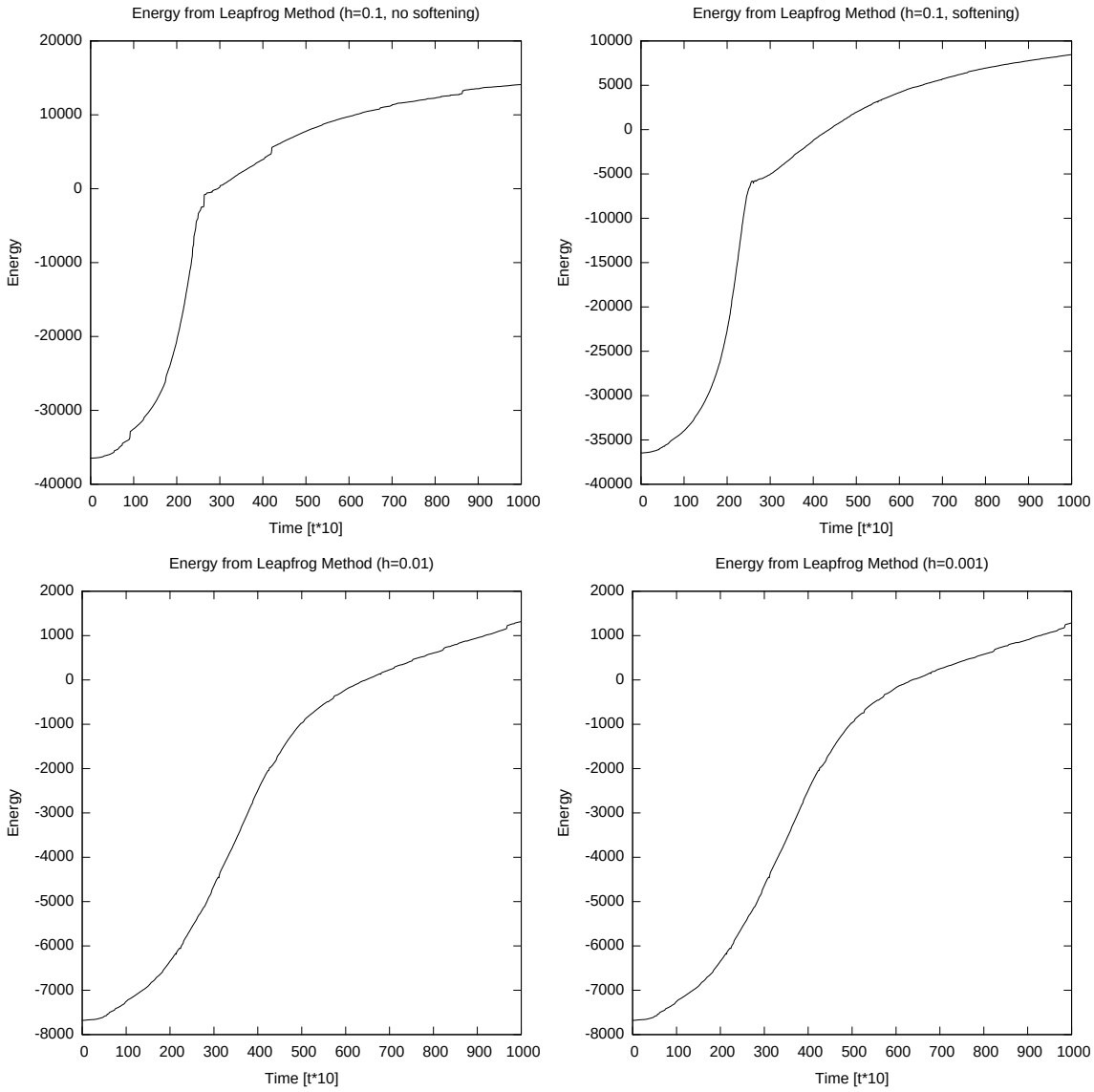

FIG. 8: Conservation of energy by the leapfrog integrator for moderate N.

time needed to integrate the system for 100 steps was recorded for both the leapfrog method and the Runge-Kutta method on two different computers. The results are plotted in figure 10.

On both machines, the leapfrog method scaled slightly better than the Runge-Kutta method. On the AMD Athlon64, the leapfrog method scaled as $O(N^{1.6})$, while the Runge-Kutta method scaled as $O(N^{1.9})$. On the Intel Core2 Duo, the leapfrog method scaled as $O(N^{1.3})$, while the Runge-Kutta method scaled as $O(N^{1.6})$. For both methods the Intel Core2 Duo performed significantly better (by a factor of 2) despite the code being single-threaded.

These scaling exponents are difficult to explain, as the nested sum present in every function evaluation is clearly an $O(N^2)$ operation. Furthermore, as the application is written in Java, SIMD instructions are not being used to accelerate the loops. In some situations it is possible for overhead to create the illusion of a smaller scaling coefficient, but that is not the case here, as the slopes of the curves are steady for large N. If the problem were to grow beyond the cache size of the processor, one would expect the overall slope to be artificially greater than the true value, not less. The low scaling exponents for this program are at present a mystery.

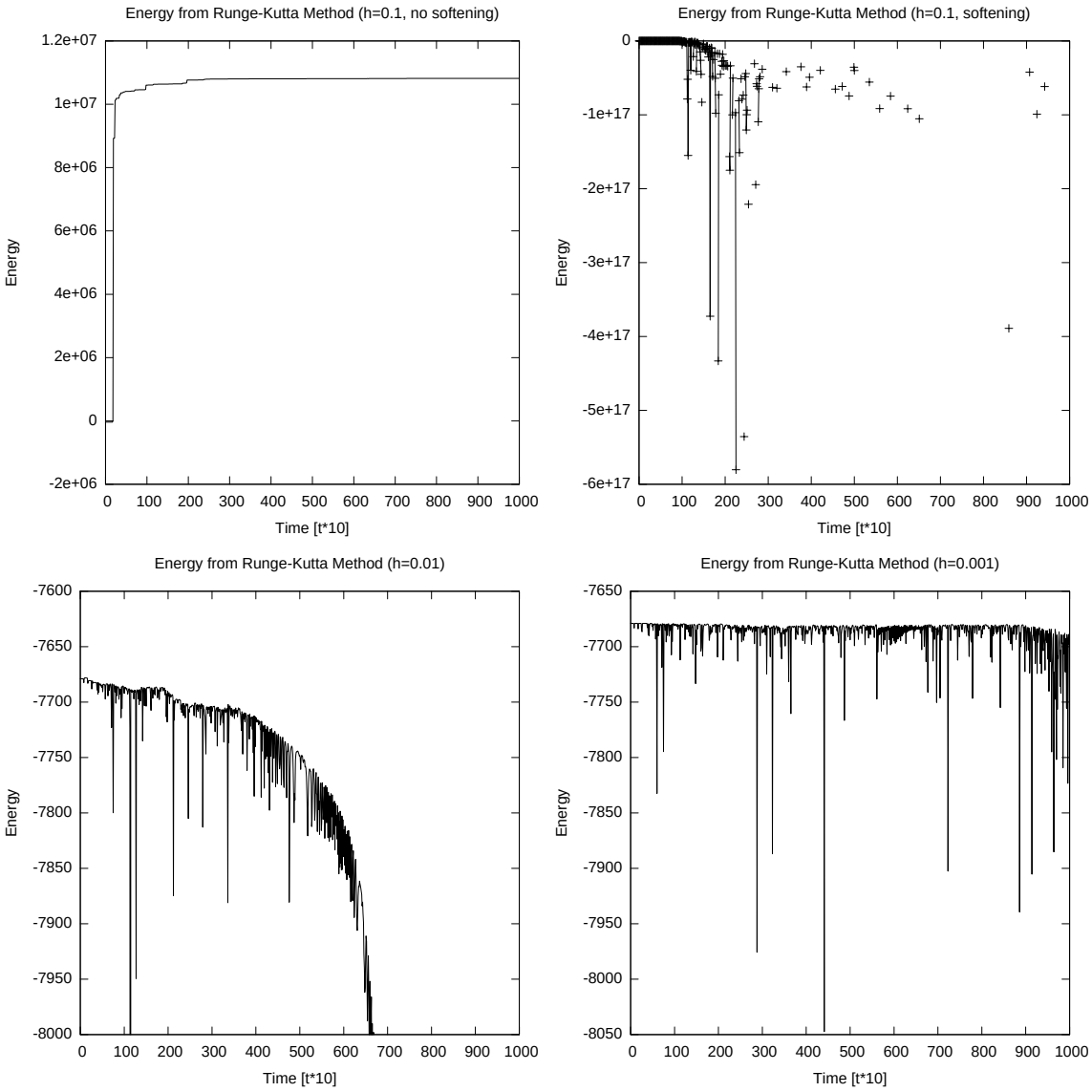

FIG. 9: Conservation of energy by the Runge-Kutta integrator for moderate N.

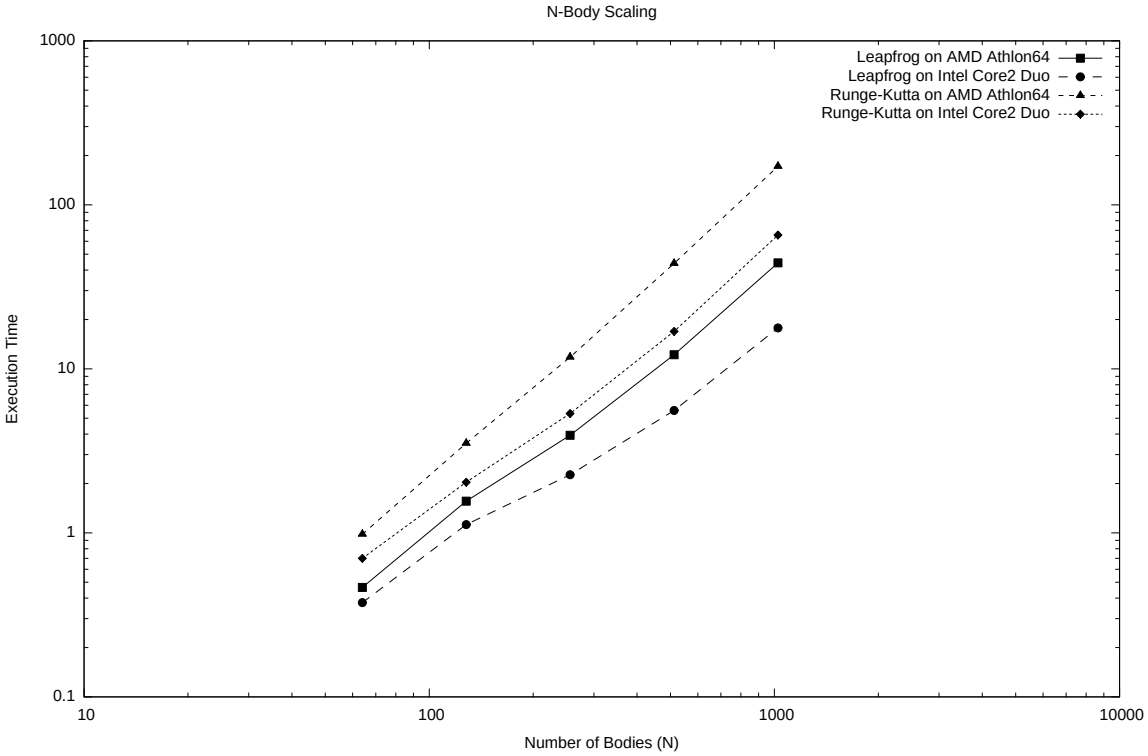

FIG. 10: Execution time versus N.