Included with the assignment is incomplete C++ code for simulating gravitational n-body problems. It currently has code to integrate the equations with a simple Verlet method, a fixed time step, and the brute force $O(n^2)$ evaluation of gravitational forces. By default n bodies are placed randomly in an ellipse, with a velocity that would match circular orbits in the two body case.

(1) Replace the Verlet integrator with classic 4th order Runge-Kutta. Define a simple test case, with n reasonably small, the initial conditions fixed, and a particular end-time. Verify numerically that you achieve 4th order convergence by running with some large Δt, then $\frac{1}{2}\Delta t$, then $\frac{1}{4}\Delta t$, then $\frac{1}{8}\Delta t$, etc. and assuming that the smallest timestep is close to exact to estimate the error for each step size.

(2) Compare the efficiency of 4th order Runge-Kutta to 4th order Adams-Bashforth: implement AB4 and, for your test case, determine a step size for AB4 that leads to roughly the same error as RK4 at Δt. Which is faster?

(3) The code also provides a class which constructs a kd-tree suitable for Barnes-Hut. However, it is missing the actual fast recursive gravity approximation: fill this in. Verify that the Barnes-Hut accelerations are accurate compared to the brute force accelerations.

(4) (pure theory) Determine the monotonicity condition for the following implicit second order Runge-Kutta method:

$$\begin{aligned} y_{n+1/2} &= y_n + \frac{1}{2}\Delta t f(y_{n+1}, t_{n+1}) \\ y_{n+1} &= y_n + \Delta t f(y_{n+1/2}, t_{n+1/2}) \end{aligned}$$

That is, when you plug in the test equation with λ a real negative number, how small must Δt be to ensure the numerical solution monotonically decays to zero?