A particle-mesh n-body algorithm was programmed to compute the n-body solutions for the following scenarios:

1) Single Particle
2) Two Particles
3) Large-N Particles: Periodic and Non-Periodic using
    a) Leapfrog Step Algorithm
    b) RK4 Step Algorithm

All of the code can be found on GitHub in the Code folder. The script that discusses the properties and functions at hand is ***particleProperties.py***

Within each of these scenarios, we measure and compare the Kinetic, Potential, and Mechanical Energies at each time-step for the varying step algorithms…

This particle-mesh algorithm finds the mass density using a reference in each grid, as demonstrated in the code. A similar method is used to find the force on each grid.

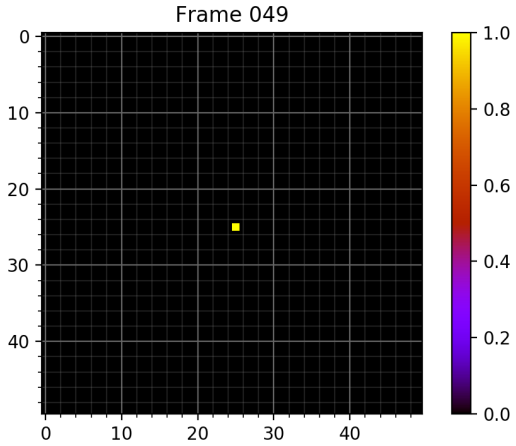
For the potential of our system, I employ the solution to the Poisson equation.

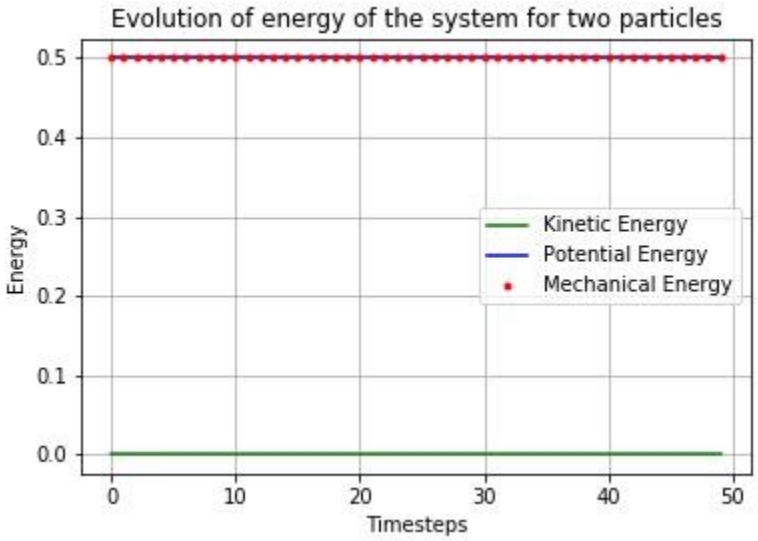
As for the forces, that is simply given by $F = -\nabla\phi$ where $\nabla^2\phi = 4\pi G\rho$ and a central difference scheme solves this. The particle-mesh nature of this problem results in particles in one grid not exerting forces on each other. This becomes more evident for large particle counts and low grid/mesh sizes as the particle per grid density increases.

In this project, I utilize both the leapfrog and RK4 methods to iterate through steps. The leapfrog method, also known as the forward Euler method, is a simple numerical method for solving ordinary differential equations (ODEs) that uses a forward difference formula for the derivative, rather than a central difference formula like the more widely known Runge-Kutta (RK4) method. This means that the leapfrog method approximates the derivative at the beginning of a time step, rather than the middle like RK4. This can make the leapfrog method more stable for some ODEs, but it can also be less accurate than RK4 in general.

In the context of N-body solutions, we will compare these two methods by their energy-conservation efficiencies.

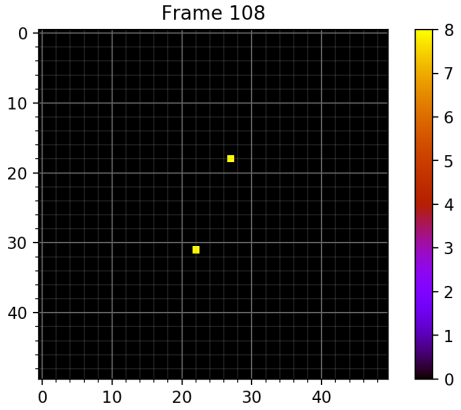
Single-Body Problem:

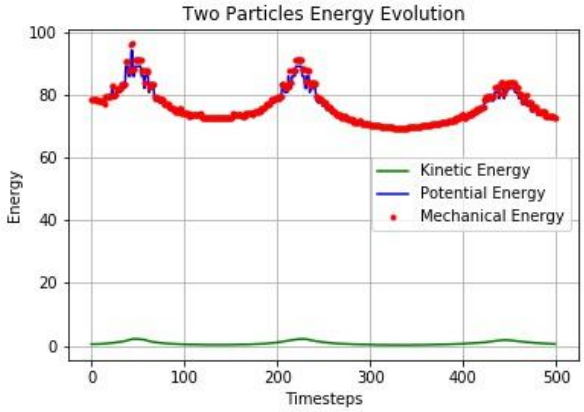




Observe in the ***SingleBody.gif*** video that the particle remains at rest. This is reinforced by the energy evolution plot for this system as the kinetic energy remains zero. Computing the potential energy for the single-body isn't intuitively correct, so the key takeaway from this plot is the zero KE.

Two-Body Problem:





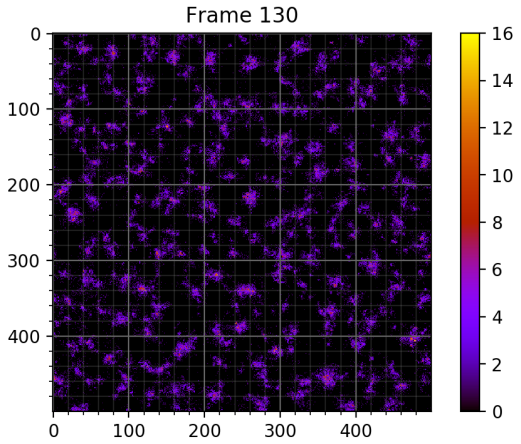
Please observe ***TwoBodies.gif*** to see the two bodies **orbiting each other for a reasonable amount of steps**. At 'longer' timescales (not included in the gif), the orbits eventually fade away with the particles flying off. This is likely due to there being a low-N count which isn't optimal in the central difference scheme for the Poisson equation. This can even be noticed in our energy plot with the peak Mech Energies decreasing over timesteps.

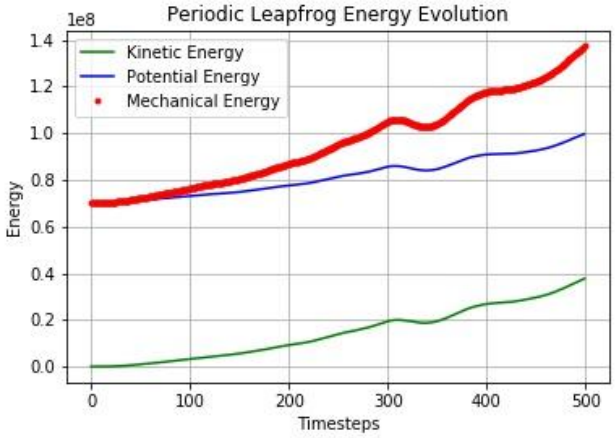
We note that the average energy is relatively constant over our timescale. The fact that Mech E isn't constant - however- is highly unintuitive, as this process is periodic. A reasonable explanation doesn't come to mind directly. Likely this comes from low-step counts, we may expect it to equalize over thousands of steps.

When experimenting with larger grid sizes, I noticed that the drift decreased with more orbits in our step domain. This is likely due to the fact that we are using a particle-mesh algorithm and averaged forces existing in non-corresponding cells push our particles.

Leapfrog Iteration Problem:

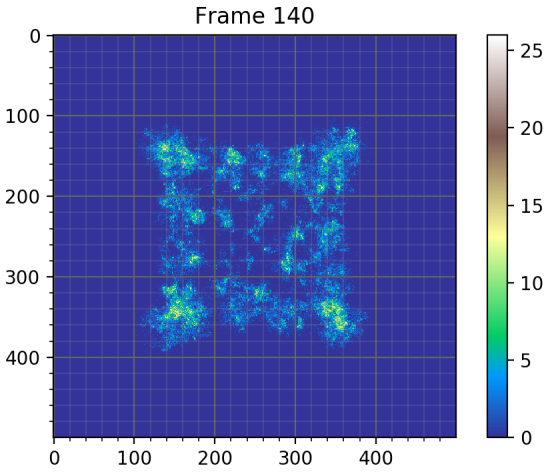
(a) Periodic Leapfrog

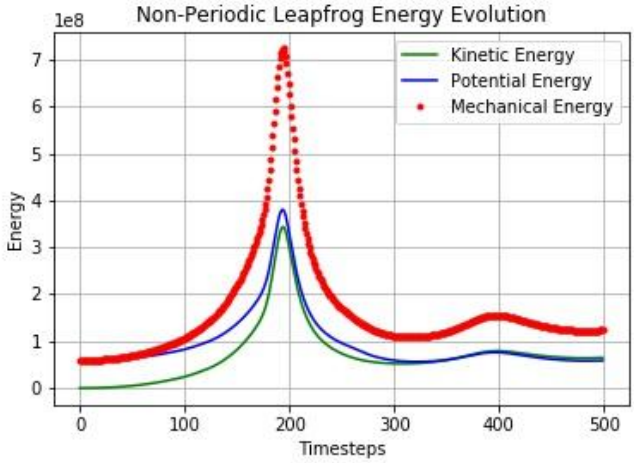




Please watch ***PeriodicLeapfrogCompressed.mp4*** to see the process in action. In all of the Large-N simulations, we run at 100,000 particles using the same random seed. We see in the periodic case that clusters instantly begin forming throughout the grid.

(b) <u>Non-Periodic Leapfrog</u>





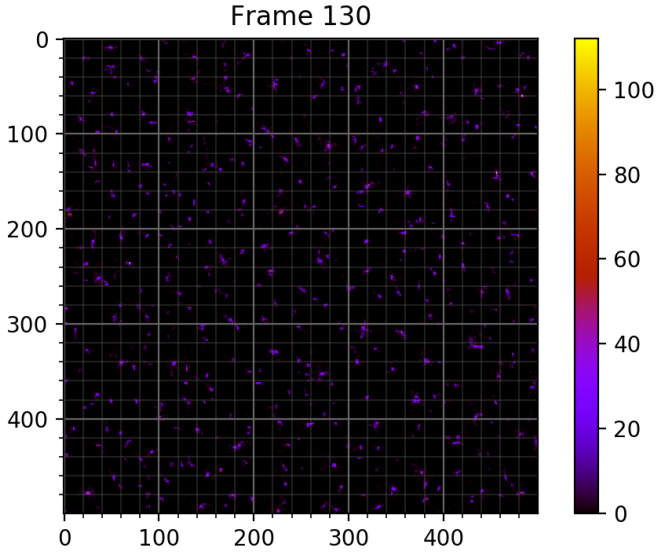
Please watch ***NonPeriodicLeapfrogCompressed.mp4*** to see the process in action. In the non-periodic case, however, we note that since forces are felt centrally towards the middle of the grid, particles collapse inwards.

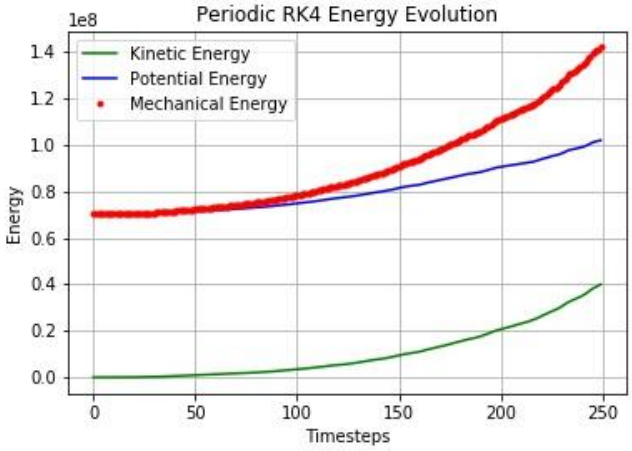
<u>Energy Comparison (P vs NP):</u>

We note here that the Periodic evolution is much more stable and horizontal than the non-periodic case. We note a massive spike in all types of energy roughly at the time of the ultimate collapse (roughly 200 steps in). In longer times, the formed ball cluster results in little dampened oscillations as observed in the tail of the non-periodic curve. Thusly, it is quite clear that energy is much better conserved in the periodic energy evolution (range of 0.7 to 1.4 in energy) than in the non-periodic one (range of 1 to 7 in energy). Is that still the case for RK4?

RK4 Iteration Problem:

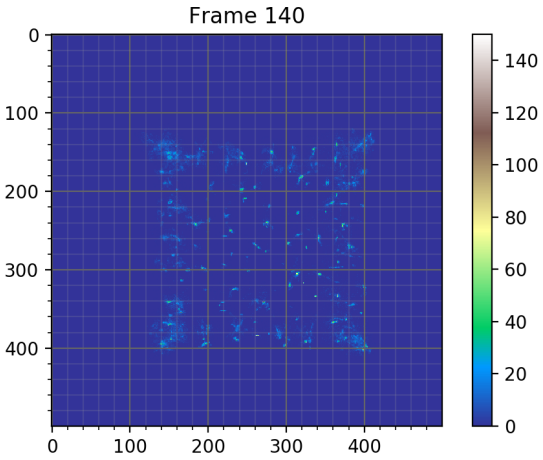
(a) Periodic RK4

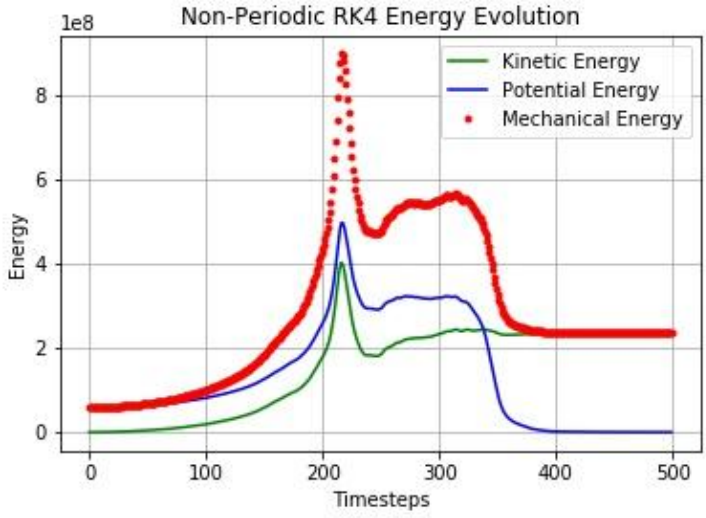




Please watch ***PeriodicRK4.mp4*** to see the process in action. In this part, we performed the same simulation as the leapfrog iterations (with an increased time constant 'dt' due to weird bugs - but keeping the same particle count and seed). Not entirely sure however why the clusters are less visible in RK4 (either due to higher densities in cells resulting in them not being as voluminous) .

(b) <u>Non-Periodic RK4</u>





Please watch ***NonPeriodicRK4.mp4*** to see the process in action. The behaviours for long timescales are rather odd. It can be observed that the cluster drifts to the bottom-right of the grid for no perceivable reason. It is rather interesting to note however that when this occurs, the mechanical energy rapidly stabilizes (at around 380 timesteps). Something weird is occurring during the second bump (between 230 and 380 timesteps). We can also note that the clusters are less visible indicating higher cell densities. This may interact with our energy conservation process resulting in this odd behaviour.

Most importantly - yes, Periodic RK4 conserves energy better than non-periodic RK4; this is identical for both iteration methods.

I sadly was not feeling awesome enough for the 3D iterations, however, after finishing this project I realize it wouldn't be too complicated to integrate (with some more computational power required).