For this project I implemented a Particle-Mesh n-body solver. Instead of tracking the interaction of every pairs of particles and summing them up we look at the potential of the system and we solve the Poisson equation

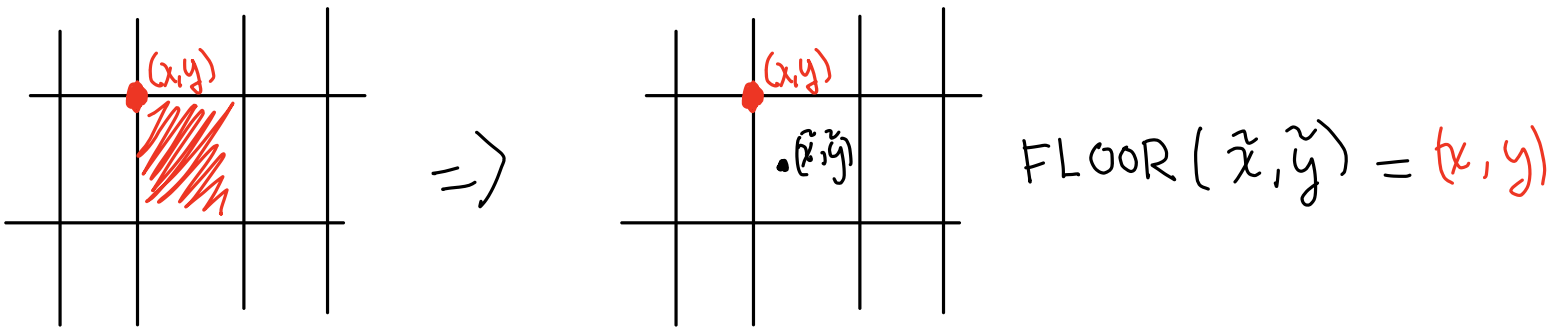
$$\nabla^2 \phi = 4\pi G \rho$$

To do that we discretize the space into a grid and compute the density in each grid square. Clearly that leads to some inaccuracies on short length scales. The good news though is that the our biggest operation runs O(N log N), so we get to use a LOT of particles.

Step 1: Finding the mass density

To find the mass density I used the Nearest Grid Point technique which simply associate each particle to its nearest grid point.

I dunno if the way I implemented is conventional. I labeled each grid square by their top left grid coordinate and floored each position to get the left top coordinate.



That its done to the whole position array all at once, then I looped over that floored array to increment the mass in the rho matrix at each position. Finally I divide rho by the area of a grid square.

Step 2: Solving the Poisson equation to get the potential

This simply amounts to taking the convolution of rho with a kernel function. More than one kernel function is possible, I just took the one used by Prof. Sievers. The convolution was obviously done doing a product in the Fourier space using FFTs. Note that for the non-periodic case the mesh is padded all around by half of its length in order to avoid wrap around.

Step3: Computing the forces

The force is given by

$$F = -\nabla \phi$$

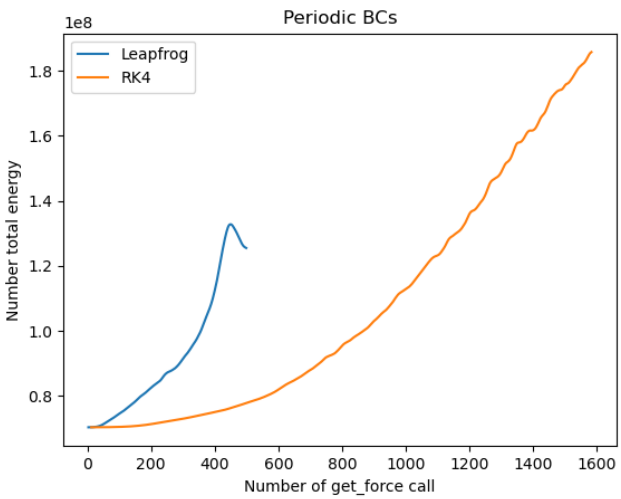
Which I solved using the central difference scheme. Just a forward difference would lead to particles acting on themselves.

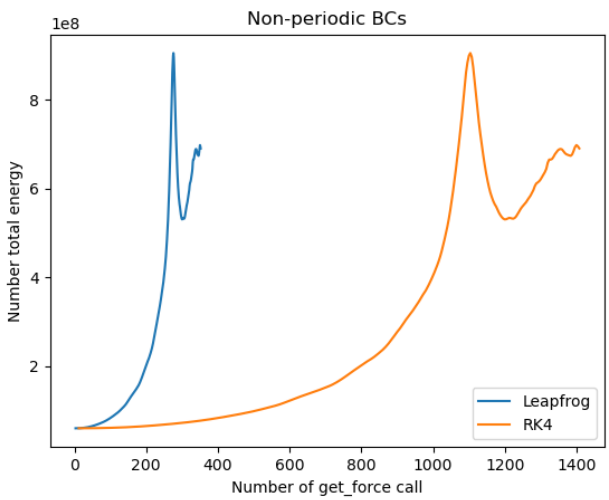
Step 4: Taking a step:

This was done in two different ways. First with a basic leap frog and secondly with a 4th order Runge-Kutta scheme. My Runge-Kutta scheme is essentially the same as Prof. Sievers. His implementation was just much better than everything I came up with.

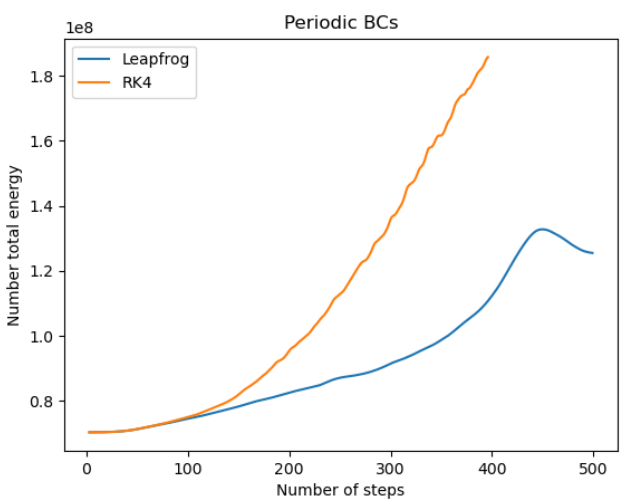
Results:

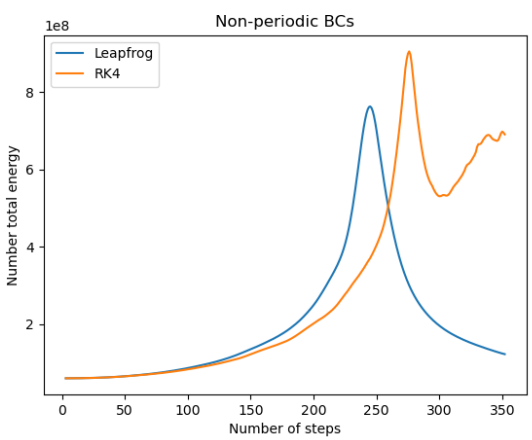
All the gifs are in the project folder of the repo. Here's I am going to present all the energy conservation stuff.





Those are my trend for energy conservation as a function of get force call. The Runge-Kutta does better at it





Those are the trends for energy conservations as a function of the number of steps. This time leapfrog does better. I read online that leapfrog is supposed to be symplectic so I am a little worried that the energy is not conserved for the leapfrog scheme.