This repository contains codes for doing N-Body simulation using particle-pair and particle mesh methods.

3D Nbody Simulation

This is simple N body simulation without any implemantation of cosmological models.

Instructions

To run the code, you need to have installed:

- Python 3.8
- Numpy
- Numba
- Matplolib

Run with:

```
$ python -m nbody3d.nbody
```

Working

The code is a simple implementation of particle - pair method, where force is computed for each pair and then summed over for each such pair. After that using the Leapfrog integration we calculate the postion and velocity, and iterate over again.

There are three main functions required for doing the simualation.

```
acceleration(pos, mass, G, softening)
```

This function calculates acceleration from the particle-pair method.
Suppose that the position of a particle of interest j be x_j and then there are N such particles. Then we calulate the relative vector between each such pair i.e $\vec{r_{ji}} = \vec{x_j} - \vec{x_i}$ where $i = \{1, 2, \ldots, N\}$. Then using Newton's Law of gravitation we calculate acceleration for each such pair and thus the net accleration becomes

$$\mathbf{a}_j = G \sum_{i \neq j} m_i \frac{r_{ji}}{|r_{ji}|^3}$$

we use the softening parameter so that the relative distance never becomes zero.
This function returns a $N \times 3$ matrix conatining the accleration vector i.e $[a_x,\ a_y,\ a_z]$ of each particle.

```
energy(pos, vel, mass, G)
```

This function calculates the energy of the total system.
The `vel` parameter here is a $N \times 3$ matrix that contains the velocity vector of each particle, $[v_x,\ v_y,\ v_z]$ so for the particle i its easy to calculate the magnitude as $v_i = \sqrt{v_{i,x}^2 + v_{i,y}^2 + v_{i,z}^2}$ Thus we can calulate the kinetic energy of each particle as $\frac{mv_j^2}{2}$.

Then we calculate Gravitational Potential Energy between each pair, for that we need the relative distance between each particle which we do the same way as before. Then the gravitational potential energy for particle i and j will be $-G\frac{m_i m_j}{r_{ij}}$.

We sum the kinetic energy for each particle and potential energy for each such pair. Thus the total energy is:

$$E_{tot} = \sum_{i=1}^{N} \frac{m_i v_i^2}{2} \quad - \sum_{1 \leq i \leq j \leq N} G\frac{m_i m_j}{r_{ij}}$$

For a good simulation, the total, energy should be near 0 throughout the simulation and should remain more or less constant.

```
main(N, tEnd, dt, softening=0.1, energyplot=False, plotRealTime=True)
```

This funtion is the main function which runs the simulation. The parameters are self-explanatory and are described in the the `nbody.py` also.

Don't keep the Number of particles too High, as this code is vectorized, so it have a drawback of large memory requirement.
This function initializes the inital grid and perturbation. If you don't want the perturbations, then you can distribute the paricles randomly as well, by *un-commenting* this line :

```
pos = np.random.randn(N, 3)
```

Note that the grid form is a cube, so make sure that the N you chose is a cube of some integer.

Credits

This code implementation is based on the code implemented by Phillip Mocaz

2D Nbody Simulation

This simulation will incorporate Friedmann's Equation and will be a more realistics simulation

Maths required

Deriving all the thing in great detail won't be possible, however I'll try to write reasonable explations for all the steps. For detailed text one should refer to Barbara Ryden's book "Introduction to Cosmology" [@Ryden2017] and Andrew Liddle's book "Introduction to Modern Cosmology" [@liddle2015introduction]

Since we are considering the universe to be dominated by the darkmatter, the dynamics is governed by the hydrodynamic equations:

$$\begin{aligned}\partial_t(r) + \nabla_r(\rho\vec{u}) &= 0\\ \partial_t\vec{u} + (\vec{u}\cdot\nabla_r)\vec{u} &= -\nabla_r\phi\\ \nabla_r^2\Phi &= 4\pi G\rho\end{aligned}$$

Now, the comoving coordinate is defined by:

$$\vec{r} = a(t)\vec{x}$$

where $\vec{r}$ is the real distance and $\vec{x}$ is the co-moving coordinate, and $a(t)$ is called the scale factor of the universe. So on moving from real to comoving coordinates, and considering a densitity perturbation

$$\delta + 1 = \frac{\rho}{\rho_b}$$

the above equations will change. The equation that we are interested in is the modififed poisson's equation.

$$\nabla\phi = 4\pi G a^2(\rho - \rho_b) = 4\pi G a^2(\delta\rho_b)$$

where

$$\Phi = \underbrace{\frac{2}{3}\pi G\rho_b a^2 x^2}_{\substack{\text{background potential} \\ \text{of homogenous universe}}} + \underbrace{\phi}_{\text{peculier potential}}$$

Now the First Friedmann's Equation is

$$\mathcal{H}^2 = \frac{8\pi G}{3}\rho_b a^2 - k$$

So for the flat universe ($k = 0$) we can write the poisson equation as:

$$\Delta\phi = 4\pi G a^2(\delta\rho_b) = \frac{3}{2}\Omega_m \mathcal{H}^2 \delta$$

Where Ω_i is a dimensionless quantity defined by $\Omega_i = \dfrac{\rho_i}{\rho_{i,critical}}$

Simulation

The code is structured in the following manner:
- ***Cosmology*** : This is for getting the Hubble's Constant and other pramters such as a, $\dot{a}$, and $\Omega_i\,'s$
- ***Mass deposition*** : Particle mesh codes need densiy as a function of the mesh rather the particle mass.
- ***Interpolation*** : We need to interpolate back grid quantities onto the particle coordinates.
- ***Integrator*** : The evoultion is described by a Hamiltonian System of equation. We will use leap-

frog integrator which is a genric solrver for such system.

- ***Poisson Solver*** : For solving the Poisson equation governing the evoulution of density.
- ***Intitalization*** : The inital perturbation is given by *Zeldovich Apporximation*
- ***Main*** : For stiching everything together.