PHYS-512: Final project

Vadym Bidula

14 Dec 2021

Dependencies

- `numpy`, `scipy`, `matplotlib`, `tqdm`
- `ffmpeg` - program suite for video handling. Used within the python code via `os` to compile set of `.png` images to `.mp4` video.

1 Overview

This project implements 2D N-body simulation based on the convolution of particles density with Green's function using FFT. The essential steps are:

1. Calculate kernel (Green's function) - a potential of a single particle on the specified grid. For non-periodic boundary conditions generated kernel is 2 times larger (because of padding the density matrix with zeros in that case).
2. Set initial conditions. It includes position, velocities and masses of particles as well, as other parameters, such as size of the grid, softening parameter, boundary conditions.
3. Calculate particle density over grid. This is done by projecting positions of particles on the 2 dimensional grid. All the following calculations will be done using density of a cell (sum of masses of particles inside its borders). For non-periodic boundary conditions all the particles outside of the grid are removed (energy will not be conserved in that case).
4. Calculate gravitational potential over the grid. This is given by convolution of particle density and kernel (element-wise multiplication in Fourier space). For non-periodic boundary conditions the density matrix is padded with zeros ($n \times n \to 2n \times 2n$) to make sure the potential will not be cyclic.
5. Gradient of potential is calculated using two-point derivatives in x and y directions separately. In non-periodic case it is assumed that potential out of grid border is always equal to zero.

6. Update position and velocities for specific time step.
7. Repeat 3-7.

2 Single particle

The simplest application of `Particles`. Code is located in `single_particle.py`. Parameters used:

Program output saved to `videos/one_particle.mp4`. As expected, the particle remains at the same spot, energy is fully preserved.

3 Two particles

Assume our particles are at distance d from each other and have the same mass m. The gravitational force is then:

$$F_g = \frac{Gm^2}{d^2} \tag{3.1}$$

Centripetal force is given by:

$$F_c = m\omega^2 r = m\omega^2 \frac{d}{2} \tag{3.2}$$

For the system to be stable these two forces must be equal

$$m\omega^2 \frac{d}{2} = \frac{Gm^2}{d^2} \tag{3.3}$$

$$\omega^2 = \frac{2Gm}{d^3} \tag{3.4}$$

Since $\omega = \frac{V}{r}$

$$V^2 = \frac{2Gm}{d^3}\frac{d^2}{4} \tag{3.5}$$

Assuming $m = 1$, $G = 1$ (used in the program):

$$V = \sqrt{\frac{1}{2d}} \tag{3.6}$$

This expression is used to set initial velocities of particles. Code is located in `two_particles.py`. Parameters used:

- $N_{grid} = 64$
- $dt = 3$

- $d = 20$ (distance as number of grid cells)

Program output saved to `videos/two_particles.mp4`. Particles in the video wiggle a bit because these frames represent a discrete grid projection of two particles, not their actual coordinates. Energy preserves at the level of 0.1%.

4 Leapfrog

Scripts for periodic and non-periodic boundaries are located in `leapfrog_periodic.py` and `leapfrog_nonperiodic.py` correspondingly. Program output saved to `videos/leapfrog_periodic.mp4` and `videos/leapfrog_nonperiodic.mp4`.

Parameters used:

- $N_{part} = 10^6$
- $N_{grid} = 1024$
- $dt = 0.02$

Energy conservation depends on the time step chosen. The time step $dt = 0.02$ was chosen by trying different time steps and is a good compromise between speed of computation and accuracy. For both periodic and non-periodic boundary conditions the maximum change in energy was about 0.03%. For comparison, choosing 10 times bigger time step increases change in energy significantly (a few percent) and creates an effect of 'snowballing', when a change in total energy leads to even greater change.

5 RK4

Code for periodic and non-periodic boundaries is located in `rk4_periodic.py` and `rk4_nonperiodic.py` correspondingly. Program output saved to `videos/rk4_periodic.mp4` and `videos/rk4_nonperiodic.mp4`.

Parameters used:

- $N_{part} = 10^6$
- $N_{grid} = 1024$
- $dt = 0.08$

RK4 method does 4x calls to get forces per step, so to obtain the "fixed computational work per unit time" I made the time step for rk4 4 times bigger. The same initial conditions were used (including random seed). As a result, the rk4 method preserves energy much worse than leapfrog in case of periodic boundary conditions (maximum change in energy for rk4 is about 64%) or does not conserve it at all in non-periodic case ($|\Delta E| \approx 510\%$). Making time step smaller slightly improves results. For comparison, setting time step equal to that used for a leapfrog, the total change in energy becomes ~ 10 times smaller, but still, this is much worse than using leapfrog at the fixed computational work.