THE GRAVITATIONAL N-BODY PROBLEM

1. The N-body problem

1.1. **Governing equations.** Newton's law of gravitation in two dimensions states that the force exerted on particle i by particle j is given by

$$\mathbf{f}_{ij} = -\frac{Gm_i m_j}{r_{ij}^3}\mathbf{r}_{ij} = -\frac{Gm_i m_j}{r_{ij}^2}\hat{\mathbf{r}}_{ij}\,,$$

where G is the gravitational constant, m_i and m_j are the masses of the particles, r_{ij} is the distance between the particles, and $\mathbf{r}_{ij}$ is the vector that gives the position of particle i relative to particle j. $\hat{\mathbf{r}}_{ij}$ is the normalized distance vector. If $\mathbf{e}_x$ and $\mathbf{e}_y$ are unit vectors in the x and y directions, respectively, then

$$\mathbf{r}_{ij} = (x_i - x_j)\,\mathbf{e}_x + (y_i - y_j)\,\mathbf{e}_y\,,$$

so that

$$r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$$

and

$$\hat{\mathbf{r}}_{ij} = \mathbf{r}_{ij}/r_{ij}\,.$$

Given a distribution of N particles, a straight-forward calculation of the force exerted on particle i by the other $N-1$ particles is given by

$$\mathbf{F}_i = -Gm_i \sum_{j=0, j\neq i}^{N-1} \frac{m_j}{r_{ij}^2}\hat{\mathbf{r}}_{ij}.$$

Note that by using another formula for the pairwise forces, other types of particle systems governed by Newtonian mechanics can be modeled, e.g in electrodynamics and molecular dynamics.

There is a built-in instability in the given formulation when $r_{ij} \ll 1$. To deal with this, we introduce a slightly modified force that corresponds to so-called Plummer spheres as follows:

$$\mathbf{F}_i = -Gm_i \sum_{j=0, j\neq i}^{N-1} \frac{m_j}{\left(r_{ij} + \epsilon_0\right)^3}\mathbf{r}_{ij}.$$

where ϵ_0 is a small number (we will use 10^{-3}). This effectively caps the maximum force between two particles and acts as a smoother for the simulation.

Date: August 10, 2022.

Using the *symplectic Euler*[1] time integration method, the velocity $\mathbf{u}_i$ and position $\mathbf{x}_i$ of particle i can then be updated with

$$\begin{aligned} \mathbf{a}_i^n &= \frac{\mathbf{F}_i^n}{m_i}\,, \\ \mathbf{u}_i^{n+1} &= \mathbf{u}_i^n + \Delta t \mathbf{a}_i^n\,, \\ \mathbf{x}_i^{n+1} &= \mathbf{x}_i^n + \Delta t \mathbf{u}_i^{n+1}\,, \end{aligned}$$

where Δt is the time step size and $\mathbf{a}_i$ is the acceleration of particle i. Note that this straightforward solution, which we will use in this assignment, becomes computationally expensive for large values of N, since the total number of operations required to compute the forces on all N particles at each time step grows as $\mathcal{O}(N^2)$. There are algorithms that can be used to improve the scaling, but in this assignment we stick with the straightforward $\mathcal{O}(N^2)$ solution.

2. Problem Setting

In this assignment you will implement a Python code that calculates the evolution of N particles in a gravitational simulation. You will calculate the motion of an initial set of particles that approximates the evolution of a galaxy.

The simulation is done in two spatial dimensions, so the position of each particle will be given by two coordinates (x and y). There will be N particles with positions in a $L \times W$ dimensionless domain (Use $L = W = 1$ in your simulations). This means that the x and y values will be between 0 and 1.

The particle masses and intitial positions and velocities of the particles will be read from a file when the program starts, then simulated for a given number of timesteps, and in the end the final masses, positions and velocities will be written to a results file. (The masses will of course be the same as in the beginning, but they will be written to the results file anyway so that it can be used as input for another simulation later.)

Since fewer bodies have less mass with which to stick together, we want gravity to scale inversely with the number of bodies. Set $G = 100/N$, $\epsilon_0 = 10^{-3}$ and timestep $\Delta t = 10^{-5}$.

3. Files included with the assignment

Download the `Nbody.tar.gz` file from Studium and unpack it. Inside it you find three directories:

- `compare_gal_files` : a C-program that can be compiled into a separate executable that can be used to compare different simulation results; you can use this to check the correctness of your results.
- `input_data` : a set of input files that you can use when testing your program.

[1] The symplectic Euler method is a version of the explicit Euler scheme which is the most basic example of a partitioned Runge-Kutta method (PRK). For the standard Euler scheme, the formula for $\mathbf{x}_i^{n+1}$ would read $\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t \mathbf{u}_i^n$. The symplectic Euler scheme preserves global properties of the gravitational system (e.g. total energy), while the standard Euler does not.

- `ref_output_data` : a set of reference output files that you can use to check that your results are correct. The `compare_gal_files` program can be used to compare your result files to the reference output files.

File format used for initial configuration (and storage of result): The binary file format used will simply consist of a sequence of `double` numbers giving the mass, position, velocity, and "brightness" of each particle. The "brightness" is a number that is not directly used in our simulations, but that should be kept and stored together with each particle in the result file, so that the result file has the same format as the input file. Both mass and "brightness" for each particle should be the same in the result file as in the input file. The order of the numbers in the binary file format is as follows:

```
particle 0 position x
particle 0 position y
particle 0 mass
particle 0 velocity x
particle 0 velocity y
particle 0 brightness
particle 1 position x
particle 1 position y
particle 1 mass
particle 1 velocity x
particle 1 velocity y
particle 1 brightness
...
```

So, the total file size will be `N*6*sizeof(double)`. Note that the same format should be used for the output file with the final result from the simulation. This means that a result of a previous simulation can later be used as starting point for a new simulation.

Unpacking the `Nbody.tar.gz` file gives a directory called `input_data` with various input files that you can use. The smallest ones with just a few particles are good to use to verify that your forces and timestepping are working properly.
`circles_N_2.gal` : two stars with equal mass moving in circles.
`circles_N_4.gal` : four stars with equal mass moving in circles.
`sun_and_planet_N_2.gal` : one heavy particle and one lighter particle orbiting the heavy one, like a planet around a sun.
`sun_and_planets_N_3.gal` : sun and two planets.
`sun_and_planets_N_4.gal` : sun and three planets.

For the larger cases that are more interesting, the input files have the following form:

`ellipse_N_01500.gal`

where the number, 1500 in this case, is the number of stars N.

Those initial distributions have an elliptic shape similar to Figure 3.1, and their evolution may lead to (for example) something like Figure 3.2.

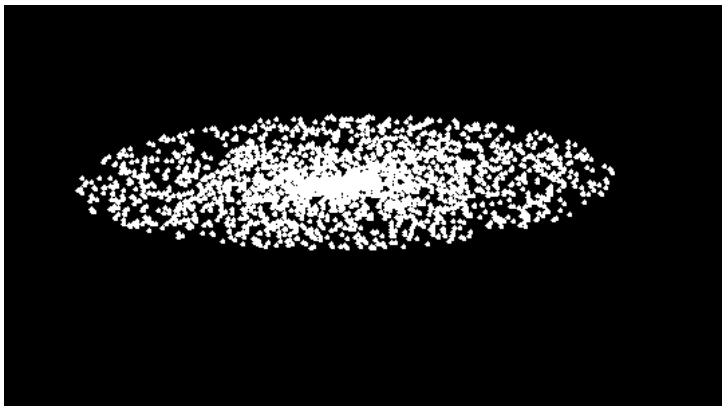

FIGURE 3.1. Example of an initial distribution of a galaxy with 2500 stars

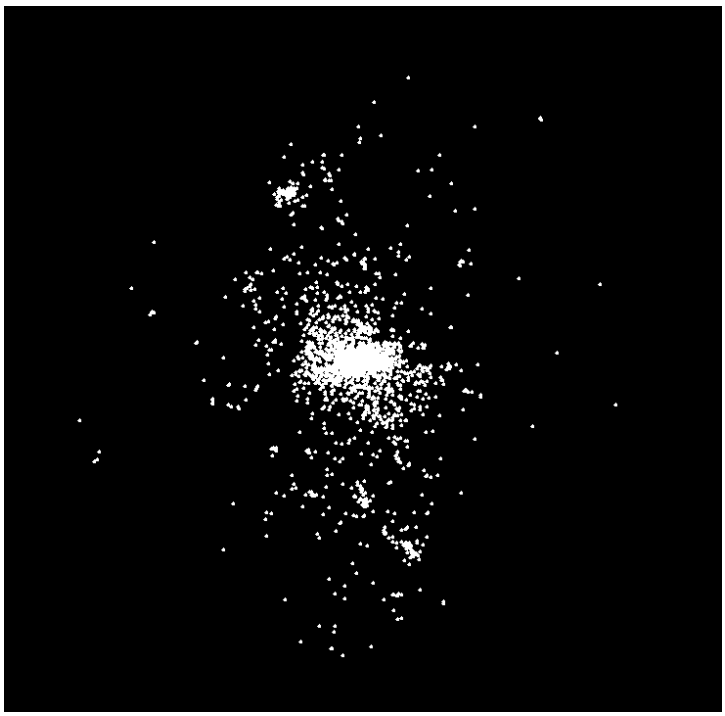

FIGURE 3.2. Example of a possible distribution of a galaxy with 2500 stars after some time (this picture is for a different initial distribution, you should not expect to get precisely this result)

3.1. **Note: simulations will always be unstable for long times.** The small test cases with for example particles moving in circles, or like a sun with a few planets, are good for verifying that the force computation and time stepping is working properly. However, even when the computations are done correctly this kind of simulations will become unstable when running for a long time since small

errors (there is always some effect of rounding errors) will accumulate over time and sooner or later this will always be seen.

So, when looking at the small simulations of a few planets etc. you should mostly focus on the initial behavior, for example check that the particles move in circles or orbit the sun in the expected way a few times, but if looking at it for longer times it is not strange that the simulations "go crazy", that is to be expected.

3.2. **Check your own results by comparing to reference output data.** To make it easier for you to check your own results, the `Nbody.tar.gz` file contains in the `ref_output_data` directory a few reference galaxy distributions in the following files:
`ellipse_N_00010_after200steps.gal`
`ellipse_N_00100_after200steps.gal`
`ellipse_N_00500_after200steps.gal`
`ellipse_N_01000_after200steps.gal`
`ellipse_N_02000_after200steps.gal`
and a small program `compare_gal_files.c` for comparison of galaxy files.

As the filenames suggest, those files contain galaxy distributions after 200 timesteps. So, for example, if you run your own simulation for 200 timesteps starting from `ellipse_N_01000.gal` your result should be close to the reference result in `ellipse_N_01000_after200steps.gal`.

The `compare_gal_files.c` program outputs the maximum difference between particle positions as `pos_maxdiff`. Since the coordinates for particle positions are supposed to be between 0 and 1, for one timestep the errors in positions should be small, on the level of rounding errors for the floating-point precision used. Errors will then of course accumulate during the simulation, but since the reference output data are for relatively short simulations, only 200 timesteps, error accumulation should not be too bad. Errors should still be within a few orders of magnitude from the size of rounding errors.