ASTRON 449, Winter 2019 – Problem Set 4

Due Thu Feb. 28, in class.

REGULAR PROBLEMS:

1. Distance to globular clusters. A self-gravitating system made of $N = 5 \times 10^5$ stars is measured to have a velocity dispersion $\sigma = 10$ km s^{-1} and an angular half-light radius $\theta = 3$ arcmin. Use this information to estimate the distance to the system. For your estimate, you may assume that the system consists of identical stars, each of mass $m = 1$ M$_\odot$.

2. Negative heat capacity of self-gravitating systems. Consider a self-gravitating system of N stars each of mass m and define the temperature T of the system at position $\mathbf{x}$ in analogy with gases:

$$\frac{1}{2}m\overline{v^2} = \frac{3}{2}k_{\rm B}T. \tag{1}$$

The mass-weighted temperature is then

$$\overline{T} \equiv \frac{\int d^3\mathbf{x}\rho(\mathbf{x})T}{\int d^3\mathbf{x}\rho(\mathbf{x})} \tag{2}$$

and we define the heat capacity of the system as

$$C \equiv \frac{dE}{d\overline{T}}, \tag{3}$$

where E is the total (gravitational potential+kinetic) energy of the system. For a stationary (virialized) system, show that $C = -(3/2)Nk_{\rm B}$, and thus that the heat capacity is negative. This implies that the *temperature of the system increases as it loses energy*, a generic property of bound, finite systems in which dominant forces are gravitational.

3. Formation of stellar binaries. Estimate the rate at which binary stars form through 3-body encounters in a self-gravitating cluster of N identical stars, each of mass m. The radius of the cluster is R. Assume that the condition for this to happen is that a star experiences a collision in which its velocity is changed by order unity, $\delta v/v \sim 1$, and that there is a third star nearby.[1] Follow the following steps:

i) Consider first one star. What is the impact parameter b_{90} of a collision that produces $\delta v/v \sim 1$? Evaluate the time scale between such encounters and express it in terms of the crossing time $t_{\rm cross}$ of the cluster.

ii) In what fraction of those collisions is there (by chance) a third star within $\approx b_{90}$ of the strong

[1] When only two point masses experience a strong encounter, they scatter elastically. When there is a third star nearby, one of the three stars can absorb energy by being flung out to infinity while the two other stars become gravitational bound to one another.

binary collision? Evaluate the time scale for a strong 3-body collision of this type, t_3, and express it in terms of $t_{\rm cross}$.

iii) Now consider the star cluster as whole. What is the time interval between strong 3-body collisions, t_3(sys)? Express t_3(sys) in terms of $t_{\rm cross}$ and compare t_3(sys) to the 2-body relaxation time of the cluster.

COMPUTATIONAL PROBLEMS:

Reminder concerning units: Treat Newton's constant G as a variable whose value can be modified in the code. By default, we work in dimensionless units and set $G = 1$.

For this problem set, you will develop a particle-mesh (PM) simulation code that will allow you to evolve the $N-$body problem for much larger N than is practical with direct summation.

We noted in class that the PM method has a computational cost $O(N \log N)$. In this case, N is the total number of nodes on the computational grid, which is often chosen to be the same as the number of particles (e.g., in cosmological simulations). In 3D, the number grid of nodes scales with the cubic power of the number of grid points along each dimension and is therefore usually very large (e.g., $512^3 \approx 134$M).

To keep the computational and memory requirements low enough that simulations can be quickly evolved on a laptop, we will restrict particle motions to one or two dimensions. If you wish, you will be able to straightforwardly adapt your code to evolve fully 3D problems.

Refer to BT2 section 2.9.3b for several important details on how to implement the PM method with vacuum boundary conditions (this means that we assume that there is no matter other than the particles we input). You may also refer to the slides on numerical methods presented in class.

C1. Computing the potential using Fourier methods. First, you will focus on a simple problem to practice using the FFT method to compute the gravitational potential due to a given mass distribution. You do not need to construct a full program including command line input and ASCII output for this problem, but only the code that you need to produce the plot in part b).

a) For a potential softening kernel S, the smoothed gravitational potential is given by

$$\Phi(\mathbf{x}) = G \int d^3\mathbf{x}' S(\mathbf{x} - \mathbf{x}')\rho(\mathbf{x}'). \tag{4}$$

Starting from this, derive an equation for $\hat{\Phi}(\mathbf{k})$ in terms of $\hat{S}(\mathbf{k})$ and $\hat{\rho}(\mathbf{k})$, where the "hat" notation indicates the Fourier transform.

b) To practice using this result, consider the 1D case with $G = 1$. The potential $\Phi(x)$ generated

by a point mass $M = 1$ at $x = 0$ is then simply

$$\Phi_{\rm point}(x) = -\frac{1}{x}. \tag{5}$$

This will be our analytic reference solution

Consider now the case of a small but finite-size density distribution of the same total mass centered on the origin,

$$\rho(x) = 5 \quad \text{for } |x| < 0.1; \quad 0 \text{ otherwise}, \tag{6}$$

and a Plummer softening kernel

$$S(r) = \frac{-1}{\sqrt{r^2 + \epsilon^2}} \tag{7}$$

with $\epsilon = 0.1$. For $|x| \gg 0.1$, the finite-size and softening effects become small and we expect the gravitational potential to become increasingly close to that generated by the central point mass.

Show this by using the PM method in 1D to numerically compute the softened gravitational potential $\Phi_{\rm fs}$ for the finite-size mass distribution over the interval $x \in [-5,\ 5]$. Choose a number of grid points sufficiently large that your solution is well resolved throughout the domain.

(Recall from the slides presented in class that you can use the functions `numpy.fft.rfftn` and `numpy.fft.irfftn` to compute the FFT and inverse FFT of Python arrays containing real numbers. Note that you will need to "double" your arrays to use the FFT method to compute the potential over a desired domain. BT2 section 2.9.3b explains how to do this and you can follow their instructions.)

c) Plot the finite-size mass distribution, the corresponding potential computed using the PM method compared to the analytic solution for the point-mass case, and the potential softening kernel. For the softening kernel, plot the values over the "doubled" domain used for the FFT. See Figure 1 for how to arrange the panels.

C2. Full PM code. You will now write a full simulation code to evolve the phase-space coordinates of a set of particles using the PM method. Your program should have the same specifications as the direct summation code that you wrote in PS3, with the following modifications:

- The accelerations should be computed using the PM method instead of direct summation.[2]
- The code should be 2D instead of 3D, with N_x grid points along each axis, so that you only need to compute FFTs on a 2D grid containing N_x^2 cells. The value of N_x should be specified on the command line as the argument `Nx` in the example below.

[2]An efficient way to do this is to start from your direct summation code and use a function call to evaluate particle accelerations given the state of the system. You can then replace the direct summation calculation with a PM calculation to evaluate accelerations.

- The size of the (square) domain along each axis should be specified on the command line by the argument `gridsize`. The particle coordinates should be such that the domain is centered on $(x,\ y) = (0, 0)$, i.e. each axis should run from $-$`gridsize`$/2$ to $+$`gridsize`$/2$.
- Use the CIC mass assignment scheme to deposit mass on the grid. Recall that you must use a consistent interpolation scheme to interpolate the potential between grid nodes in order to ensure conservation of linear momentum.
- As in PS3, you need only implement the leapfrog integrator and the computations should use a Plummer softening length specified on the command line.[3]

The program should be runnable using a command of the form

```
python pm2d.py input_data integr t dt eps gridsize Nx dt_out output_base,
```

where all arguments not defined above are as in PS3. The ASCII outputs should also be in the same format as in PS3, except that the particle distribution snapshots should only include x and y coordinates:

```
1 m1 x1 y1 vx1 vy1
2 m2 x2 y2 vx2 vy2
...
N mN xN yN vxN vyN.
```

b) Download the initial conditions `input_binary_2d.dat` from the course website. These correspond to an extreme-mass-ratio binary problem, in which the first (central) mass has `m1=1` and the second mass has `m2=0.000001`. To a good approximation, this problem corresponds to a test particle orbiting in the Keplerian potential generated by the first particle. The initial position and velocity of the second particle were chosen so that its orbit should closely match the circular orbits that you computed in PS1 and PS2. Evolve these ICs for a full orbital period using your PM code with the following parameters:

```
python pm2d.py input_binary_2d.dat 6.3 0.1 0.1 5.0 512 0.1 outputs/binary_test_pm2d
```

c) Produce a plot similar to the one in PS3 (for the direct summation problem) which shows the particle positions at $t = 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6$ as well as diagnostics of energy and angular momentum conservation vs. time. If your PM code is correct, you should find that the light particle follows a nearly circular orbit and that both the total energy and angular momentum of the system are conserved.

You will use your PM code to evolve more interesting systems of large-N particles in the next problem set, so it is important that you complete this problem and validate your code using the

[3] In general, you will want to use a softening length larger than ≈ 3 cells, otherwise the softening length will not be resolved by the grid.

above test problem.

To upload: Your plots for problems C1 and C2; example global output and final particle data files for the test problem in C2; copies of your Python codes; and answers to the questions above.

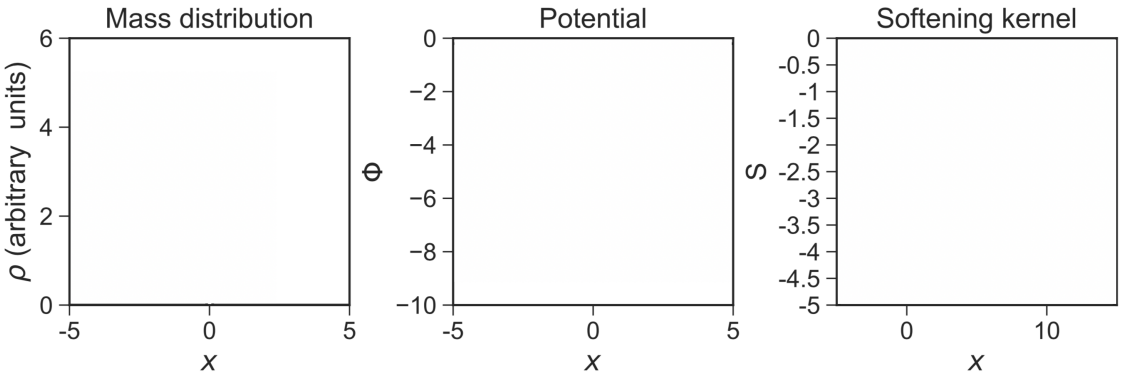

Fig. 1.— Example multi-panel plot for the 1D FFT problem in C1.