

## Computational Physics Project / Gravity on a mesh

This project focuses on solving Poisson's equation on a uniform mesh. The problem you will solve here involves a technique known as *particle-mesh* which is often used in the context of simulating the formation of structure in the universe and in other contexts.

The version of Poisson's equation we need to solve is:

$$\nabla^2 \phi(\vec{x}) = 4\pi G \rho(\vec{x}) \quad (1)$$

In order to follow how structure forms due to gravity, the particle-mesh technique approximates the distribution of matter using a large set of particles in 3-dimensions. The simulation starts at some initial time step with some initial conditions for the particles. The density field is approximated by counting the number of particles in each cell of a rectangular mesh. This yields a 3D array containing the density field. Then, Poisson's equation is solved for the gravitational potential. The acceleration on each particle can then be calculated as the gradient of the potential. The velocity of the particle is updated, and its position is updated according to the velocity. Then the process is repeated with the new (slightly different) density field. The basics of this method were described thoroughly in a classic book by Hockney & Eastwood (1988), *Computer Simulation Using Particles*.

### 1. Rescaling the problem

First it is important to scale certain factors out of the problem for convenience:

- Define a new unitless set of variables for position, mass, velocity, and time, by scaling out the length scale, time scale, total mass scale, and  $G$ . Set the size of the cubic region we are considering to unity, and  $G = 1$ , in these new units. There should be a single combination of these overall scale values that characterizes the expression. You have the freedom to choose to set this combination to unity. You should perform your numerical analysis in these variables; your numerical solutions can then be scaled to different total mass and lengths by keeping this combination fixed.

### 2. Particle positions and density field

The first thing you will need to do is to define a set of particle positions and infer a density field. We will use the simplest *cloud-in-cell* approach to this problem. You should use  $32^3$  particles and a  $32 \times 32 \times 32$  grid, so that things run fast.

- First, you need to distribute a bunch of particles in the 3D space to represent a given density field. Write a function to distribute particles according to a multi-variate Gaussian with a chosen center, chosen semimajor axis  $a$ , and chosen axis ratios ( $b/a$  and  $c/a$ ). It is fine if you let the principal axes of the Gaussian be aligned with the coordinate system you are working in.
- Second you need to define a density field based on the particles. To do so, imagine each particle is a small cubicle “cloud” with sides equal to the size of a grid cell. Then the contribution of each particle to the density in each cell is just the fractional overlap of the particle and the cell. Write a function that evaluates the density field and make some plots to show that it is working as expected.

### 3. Solving Poisson’s equation

Then you need to solve Poisson’s equation:

$$\nabla^2 \phi(\vec{x}) = 4\pi\rho(\vec{x}) \quad (2)$$

This is a partial differential equation (PDE), but you do not need to wait until we learn about PDE solutions in order to solve this equation. You can use Fourier methods to perform the solution — i.e. use the fact that the Fourier transform of the above equation is:

$$k^2 \tilde{\phi}(\vec{k}) = 4\pi\tilde{\rho}(\vec{k}) \quad (3)$$

However! You cannot use precisely this formula with a discrete Fourier Transform. Instead, there is a discrete version of this formula.

- Write the derivation of and implement the discrete version of the Fourier solution described above. Test the case of a point source at the center of the box. The result is a Green’s function response of the method to a delta function source. Show that its potential is not what you would expect from an isolated delta function source, but that it is what you would expect with a periodically repeated delta function source.
- To simulate an isolated distribution of mass, you have to use a special trick. First, you need to “isolate” the mass distribution with a moat of zero density, by expanding the grid size by a factor of two in each dimension. Second, you need to limit the range of the Green’s function, so that the periodic isolated regions cannot influence each other. To do the latter, you need to create the Green’s function response you want numerically:

$$g(\vec{r}) = \frac{1}{r} = \frac{1}{\sqrt{x^2 + y^2 + z^2}} \quad (4)$$

for  $|x| < 1$ ,  $|y| < 1$ ,  $|z| < 1$ , 0 otherwise, and  $g(\vec{0}) = 1$ . Then instead of  $k^{-2}$ , use the discrete Fourier Transform of this Green’s function. Hockney & Eastwood, around pages 212–213,

explain this. Implement this method and show that it works as expected for a delta function source.

- Use your implementation to calculate the potential for several cases with different widths and axis ratios of the Gaussian. Test a spherically symmetric case against the analytic expectation.

#### 4. Integrating the equations in time

Now you have initial positions, the resulting density field, and the potential resulting from that density field. You can now integrate the system forward in time. The equations for each particle are:

$$\begin{aligned}\dot{\vec{v}} &= -\vec{\nabla}\phi \\ \dot{\vec{x}} &= \vec{v}\end{aligned}\tag{5}$$

- Implement the Verlet integrator I describe in the notes for ODEs, for the particle-mesh case. Between each integer step, update the positions according to the velocities. Between each half-integer step, update the velocities according to the gradient of the potential at the location of the particle; use a first-order method to calculate the gradient.
- Run the particle mesh for several initial Gaussian distributions of particles at rest. Make some animations showing what the particles do.