# Gravity on a Mesh

Mitchell Karmen, Christina Liu, Carla Salguero

Computational Physics Fall 2019

## 1 Overview

Calculating gravity on a mesh allows us to simulate the gravitational forces on many particles without exhaustive computing. The particle-mesh technique enables us to approximate the distribution of matter using a large set of particles in 3-dimensions. We use this mesh to derive a gravitational potential for many particles quickly. The manual [1] guides us to start the simulation by an initial time step with some initial conditions for the particles. We then repeatedly find the potential on the mesh to evolve the forces over time.

A main goal of this project is to solve the following Poisson Equation:

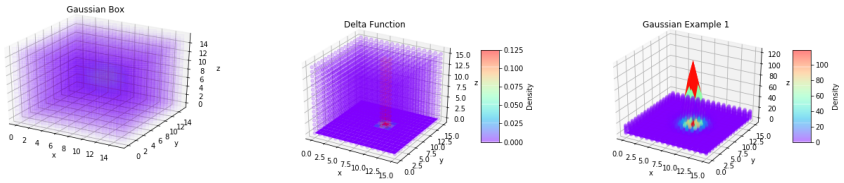
$$\boxed{\nabla^2\phi(\vec{x}) = 4\pi G\rho(\vec{x})} \tag{1}$$

This equation relates matter density to gravitational potential surrounding that matter. In order to approximate the density field, we count the number of particles in each cell of a rectangular mesh. This requires us to create a 3D array that is made up of cells for which we calculate densities. Calculating densities in this larger-scale way allows us to dramatically reduce the number of operations we need to solve Poisson's equation. We see that Poisson's equation gives us gravitational potential, a scalar field. The gradient of the potential provides the acceleration on each cell. After this is done, the motion of the particles in the cell is found, and their positions are updated. The process continues with new, and slightly altered density fields.

## 2 Re-scaling the problem

### 2.1 Goals

To avoid working with very large numbers and risking round-off error and expensive computation, we re-scale the problem. We define a new set of unitless variables that pertain to position, mass, velocity, and time. In order to make it unitless, we need to scale out the characteristic length, time, and total mass scale. Hence, we set G equal to 1.




(a) 3D density field for a Gaussian distribution of particles

(b) 2D cross section of a Dirac delta density field

(c) 2D cross section of the Gaussian density field

Figure 1: Visualizations of different density fields for different mass distributions

### 2.2 Procedure: Writing the Program

## 3 Particle positions and density field

### 3.1 Goals

We create a 16 by 16 by 16 grid that contains $16^3$ particles. In order to do this, we distribute a sample of particles in 3D space to produce a given density field. Specifically, we wrote a function that is able to distribute those particles in terms of a multi-variate Gaussian. This function has a chosen center, semi-major axis a, and axis ratios ($\frac{b}{a}$ and $\frac{c}{a}$). We create a 3 by $N$ array of particles, where $N$ is the number of particles, and we save the x, y, and z component of each particle's location. We chose the Gaussian to be centered around the center of our cube, and the variance to be very small.

After this is done, we define the density field based on our particular array of particle locations in 3D. We assume each particle to have a finite volume, and add a portion of its mass to each cell that the particle overlaps based on how much they overlap. This way, we can sum over the masses of all of the particles for the total density distribution. Finally, we write a function that evaluates the density field and make some plots to show that it has worked as expected.

To check that we got the density field function correctly, we made a 16 by 16 by 16 grid full of zeros and then plotted the density field on a color map. This is on a rainbow colormap, where redder indicates denser. As seen in Figure 1(a), this is a multi-variate Gaussian, as it is denser in the middle.

Additionally, found it useful to visualize the density field more clearly by plotting 2D cross sections of it. Included in Figure 1(b) and Figure 1(c) is the cross-section of a delta-function density field, and the aforementioned Gaussian distribution (with a surface representing the density distribution above).

### 3.2 Cloud-In-Cell Approach

The cloud in cell approach (CIC) lets us visualize the particles as cubes in 3-dimensions that have uniform density and they are the size of one grid cell.

The way our program implements the CIC approach [4] is by defining how it's used. Our cube is in 3D, so we define an x, y, and z value for each particle's position. The density contribution for each particle will be described as $x_p$,$y_p$,$z_p$. When a particular cubical cell contains a particle, it will use the following notation:

i = [$x_p$] j= [$y_p$] k=[$z_p$]

We wrote a helper function, truncate, [5] which is used in CIC to help us find which cell the particle belongs to. We use the floor function to allow for either the value to be less than or equal to itself. By setting delta x to be 1, we can then say that the center of the cell is when $x_c$, $y_c$, $z_c$ = i,j,k. We do this in the program by setting the center of the cell a distance 0.5 more than the values of i,j,and k.

Now, the particle $x_p$,$y_p$,$z_p$ has the possibility to continue to the densities of either the parent cell, which is labeled i,j,k or, since it is a 3D cube, to seven other neighboring cells. The distance of where the particle is located minus the center of the cell is labeled dx and tx is defined as the size of the cell minus the distance dx.

Then, by taking to account that there are many variations of how a particle can interact with 8 cells, the following relationship is made for all 8 possibilities:

$$\begin{aligned}
\rho_{i,j,k} &= \rho_{i,j,k} + m_p t_x t_y t_z; \;\; \rho_{i+1,j,k} = \rho_{i+1,j,k} + m_p d_x t_y t_z; \\
\rho_{i,j+1,k} &= \rho_{i,j+1,k} + m_p t_x d_y t_z; \;\; \rho_{i+1,j+1,k} = \rho_{i+1,j+1,k} + m_p d_x d_y t_z; \\
\rho_{i,j,k+1} &= \rho_{i,j,k+1} + m_p t_x t_y d_z; \;\; \rho_{i+1,j,k+1} = \rho_{i+1,j,k+1} + m_p d_x t_y d_z; \\
\rho_{i,j+1,k+1} &= \rho_{i,j+1,k+1} + m_p t_x d_y d_z; \;\; \rho_{i+1,j+1,k+1} = \rho_{i+1,j+1,k+1} + m_p d_x d_y d_z
\end{aligned} \tag{2}$$

By having the function return $\rho$, we are able to make a density grid.

### 3.3 Visualizing Cross Section

Now we are able to make random points for a 16 x 16 x 16 grid and use the CIC function to create values for $\rho$. We want to verify that the densest volume is the center of our Gaussian, because that is where many particles overlap. We are able to visualize the cross section via scatter plot or surface. We created a scatter plot that is the size of the cell and creates a colored legend that indicates the density of the particles, as shown in Figure 2.

The second way to visualize this is by creating a surface embedded in 3D, whose height depends on the density of a 2D cross section. That is seen in Figure 3.

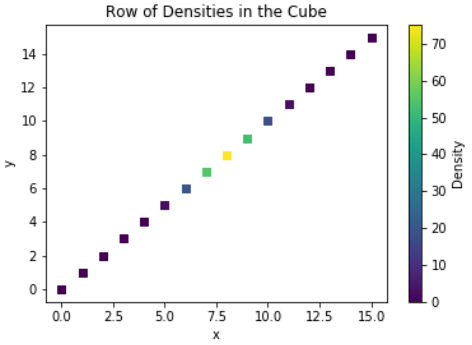



Figure 2: Scatter plot with color based on density

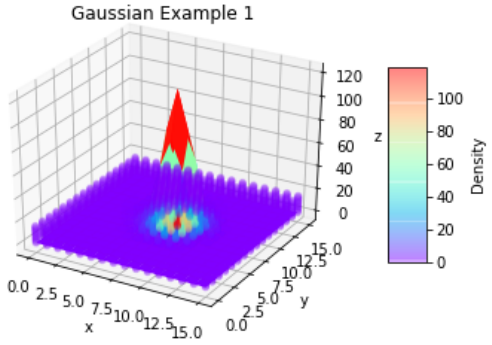



Figure 3: Surface height based on density

Creating an image can allow us to physically see how these particles behave and find a way to articulate it. When we implemented 3-dimensions to the program, we saw that we get a closer look to the center of the cube, there are more interactions taking place.

Now that we have an understanding of how these particles look like, we must solve Poisson's equation for the gravitational potential that they produce.

# 4 Solving Poisson's equation

## 4.1 The Fourier Method

We don't need to actually solve the PDE in order to find solutions to this equation in time. We can take Poisson's equation:

$$\vec{\nabla}^2\phi(\vec{x}) = 4\pi\rho(\vec{x}) \tag{3}$$

and Discrete Fourier transform each side to write a more numerically efficient solver. If we write Poisson's equation in finite difference form:

$$\rho_j = \frac{\phi_{j+1} - 2\phi_j + \phi_{j+1}}{\Delta\phi^2} \tag{4}$$

Where $\Delta\phi$ in our case is one. We can DFT this to see that:

$$\frac{1}{N}\sum_{n=0}^{N-1} \tilde{\phi_n} e^{-2\pi ijn}[e^{(-2\pi in)/N} - 2 + e^{(-2\pi in)/N}] = \Delta\phi^2 \frac{1}{N}\sum_{n=0}^{N-1} \tilde{\rho_n} e^{(-2\pi ijn)/N} \tag{5}$$

It is evident that a possible solution of this equation is that each term in the sum is equal. Moreover, Fourier transforms are unique. Therefore, the unique solution is that each individual term is equal:

$$\tilde{\phi_n}[e^{(-2\pi in)/N} - 2 + e^{(-2\pi in)/N}] = \Delta\phi\tilde{\rho_n} \tag{6}$$

Which we can rearrange and use trigonometric identities to find the relationship between the Fourier transforms of $\rho$ and $\phi$:

$$\tilde{\phi_n} = \frac{\Delta\phi^2\tilde{\rho_n}}{2[\cos 2n/N - 1]} \tag{7}$$

Finally, if we imagine the continuous limit where $N \to \infty$, we Taylor expand the cosine:

$$\tilde{\phi_n} = \frac{\Delta\phi^2\tilde{\rho_n}}{2[1 - \frac{1}{2}(2\pi n/N)^2 - 1]} \tag{8}$$

and because $k = 2\pi n/\delta\phi N$,

$$\tilde{\phi_n} = \frac{\tilde{\rho_n}}{-k_n^2} \tag{9}$$

The $-1/k^2$ is a Green's function, which we will implement based on $r$ in the coming steps.

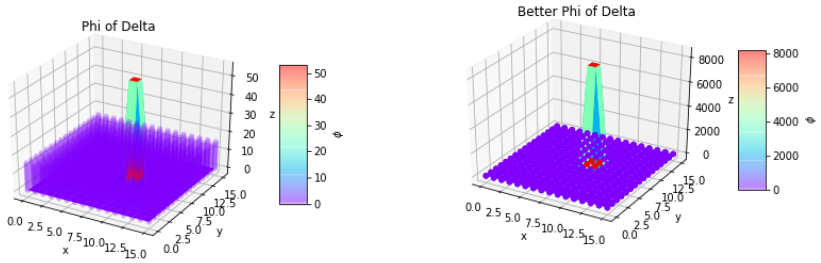
## 4.2 Goals

After solving Poisson's equation, we need to test the case of the point source at the center of the box, which is our delta function density field. As seen in Figure 4(a), the result looks like a Green's function [3] response to a delta function source. We want to check that this is the Green's function we expect from a delta function source.

We, in fact, don't get the expected response from an isolated mass, but from a periodic delta distribution. This is because the FFT assumes a period of $N$, the size of our box. This is most easily seen in the potential off-center cells in Figure 4(a).

To remedy this, we must create an isolated distribution of mass. This is most easily done by replacing our $k^{-2}$ with a Green's function, and imposing boundary conditions on that function.

**Steps to create an isolated distribution of mass:**




(a) Phi of delta function

(b) A better phi of delta function using Green's theorem

Figure 4: Comparing the phi we get using the two different methods

1. Isolate the mass distribution with a boundary zero density cells and expand the grid size by a factor of 2 in each dimension.
2. Then, limit the range of Green's function so that those regions that are isolated do not influence each other. We can do this by setting the Green's function equal to zero outside the inner cube.

After implementing these changes, we get Figure 4(b), which we can see is a lot flatter than Figure 4(a). This shows that we eliminated the problem of the FFT assuming a period of N. We can see that Figure 4(a) is not exactly flat at the bottom, indicating some periodic changes, while Figure 4(b) is constant.

### 4.3 Visualizing $\phi$ in 3D

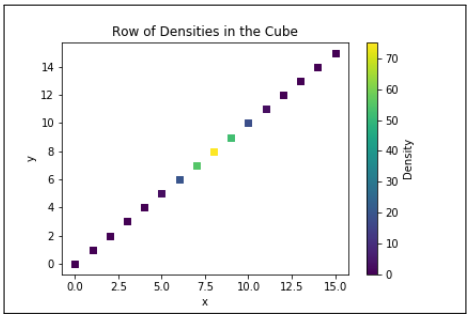
As the following equation of $\phi$ shows,

$$\boxed{k^2\tilde{\phi}(\vec{k}) = 4\pi\tilde{\rho}(\vec{k})} \tag{10}$$

we need to obtain $\phi$ in terms of $k^2$. This means we have to first compute the N-dimensional discrete Fourier Transform of $\rho$ and then compute the N-dimensional inverse discrete Fourier Transform of the values to the right, including $k^2$.

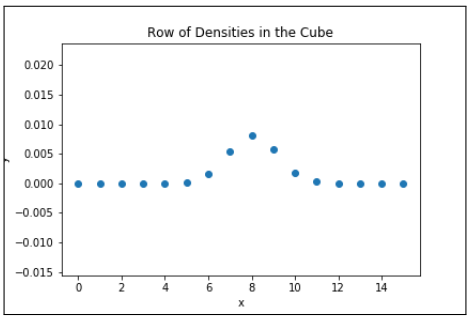
### 4.4 The Cross Section of $\phi$

Similar to what we did to be able to understand how the particles can overlap, we want to see how this relates to $\phi$. We want to visualize the overlapping that occurs with this value. We get a similar graph:




(11)

Another helpful visualization that led us to believe that our program ran correctly is by plotting a scatter plot whose y axis is the values of $\phi$ and the x axis correlates to the interval on the cube.

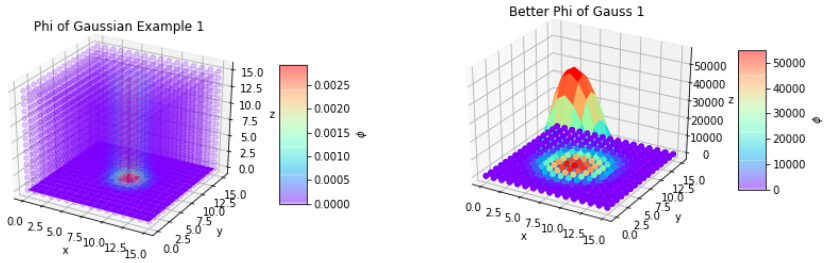



(12)

## 4.5 Green's Function

We define both a Green's function and a grid to visualize Green's function. With this, we then pad 2 rows in each dimension. This allows us to simulate an isolated mass density. By using Green's theorem, we are able to get $\phi$. Just like we did for understanding the particles with respect to density and the value of $\phi$, we will do the same for $\phi$, but by using Green's theorem.

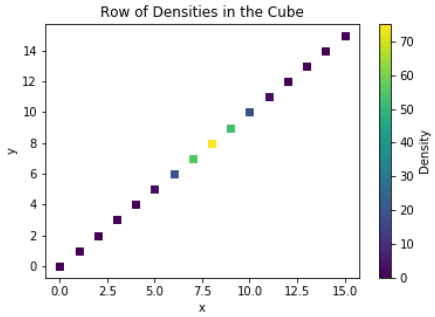
The cross section of $\phi$ is seen below:




(a) Phi of Gaussian function

(b) A better phi of Gaussian function using Green's theorem

Figure 5: Comparing the phi we get using the two different methods




(13)

We also create a definition that creates a grid for green's function for both Gaussian and delta. By correctly defining our global variables N, delta, $\rho$, and $\phi$, we can nicely get what looks like a delta density distribution.

Now, let's look at the Gaussian distribution and see how it behaves. Figure 5 shows the difference in the two methods of computing $\phi$.

Now, when we use the definition we created to create a grid for green's function and use a padded $\rho$ value, we get a nice visualization of how it behaves in Figure 5(b).

Now, we can use this implementation to calculate the potential for several cases with different widths and axis ratios of the Gaussian. Changing the different widths and axis ratios yields different phis as seen in Figure 6 and Figure 7.

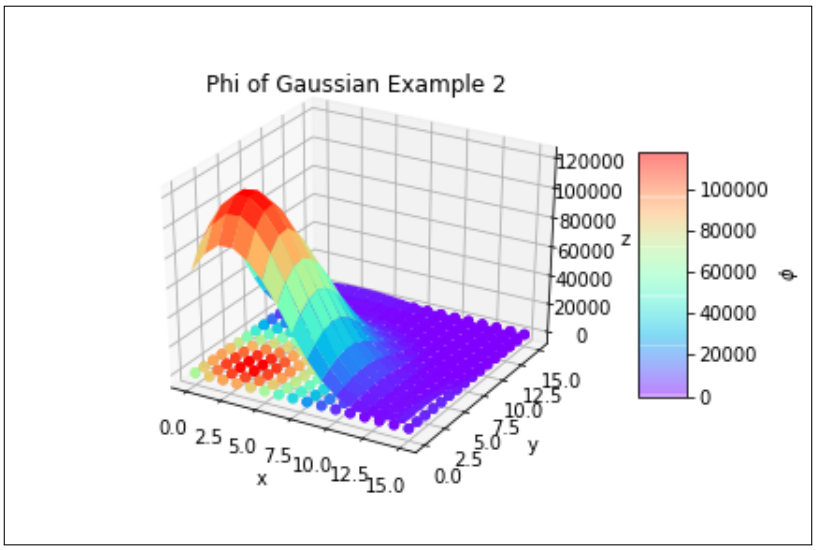



Figure 6: Phi of Gaussian with $\mu = 6$ and $\sigma = 2$

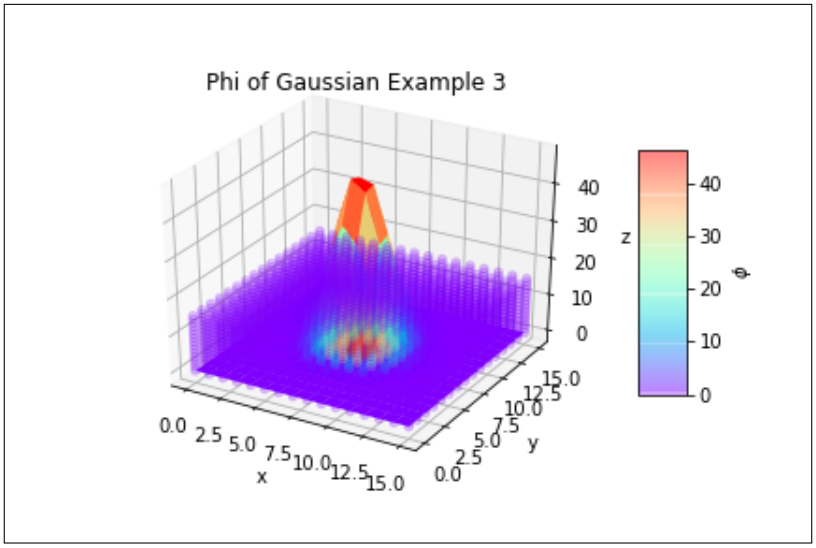



Figure 7: Phi of Gaussian with $\mu = 8$ and $\sigma = \frac{2}{3}$

# 5 Integrating the equations in time

## 5.1 Goals

We now want to see how this system evolves in time. To do this, we use the positions of our particles at each step to calculate gravitational potential. We can use the potential to get accelerations as seen below, and use these accelerations to update the positions of the particles for some finite time step $dt$. We repeat this process, and watch the system change.

$$\boxed{\begin{aligned} \dot{\vec{v}} &= -\vec{\nabla}\phi \\ \dot{\vec{x}} &= \vec{v} \end{aligned}} \tag{14}$$

After calculating accelerations from $\phi$, we use a Verlet integrator in order to update positions. Verlet integration is useful in a physical simulation like this because the steps are time-reversible, and that it approximately conserves energy (it follows a path of constant area in phase space). It is worth noting that we do not follow the exact Verlet integration steps in the course notes, as seen below.

## 5.2 Integration Technique

It is possible to write the equations of the Verlet integrator to cancel out the velocity term in all steps but the initial. Our first step is:

$$\vec{x}_1 = \vec{x}_0 + \vec{v}_0\,\Delta t + \frac{1}{2}\vec{A}(\vec{x}_0)\,\Delta t^2 \tag{15}$$

And our following steps are:

$$\vec{x}_{n+1} = 2\vec{x}_n - \vec{x}_{n-1} + \vec{A}(\vec{x}_n)\,\Delta t^2 \tag{16}$$

For each step, $\vec{A}$ is calculated from $\phi$, which depends on $x$. We want to see how the positions of the particles change as time progresses.

The best way to visualize this is through an animation. Gifs of simulations of 64 and 4096 particles are attached.

# 6 Conclusion

In all, we created a particle mesh in a cube, that accurately simulates the interaction of many particles quickly. PM simulations are commmonly used in cosmology to simulate structure formation, and test dark matter models. It would be interesting to scale up our code and involve effects of general relativity to test how our simulations replicate real systems.

# 7 References


[1] Blanton, Michael. "Gravity on a Mesh". *https://blanton144.github.io/computational/pdf/project-gravity-mesh.pdf.*

[2] Fenics Project. "Fundamentals: Solving the Poisson equation". *https://fenicsproject.org/pub/tutorial/sphinx*
*IwAR28dOWDEFilrfAHO6IFKb6hoF5x9svZmk2wVFbSFIrjo2CPILYlhWL−*
*Smk.*

*[3] "Green's Function". https://en.wikipedia.org/wiki/Green%27s_function.*

*[4] Kravtsov, Andrey. "Writing a Particle Mesh Code". http://background.uchicago.edu/ whu/Courses/Ast321-*
*11/pm.pdf?fbclid=IwAR2cDZu5Cx8SF-natWufadoqTLi08mMc3CiJJSExSu9bdH324jnc5RhKfFE.*

*[5] "Open-Source FFT-Based Poisson's Equation Solver for Computing With Accelerators". https://www.researchgate.net/publication/330405662_PittPack_Open-*
*Source-Based$_P$oisson's_Equation_Solver_for_Computing$_W$ith$_A$ccelerators?fbclid =*
*IwAR2Lwu9IArVj0T6UkMUXzxzqSTi8F1ZI0KvmmYWL8uG7bXLPZ4ZraWd−*
*xek.*

*[6] "Verlet Integration". https://en.wikipedia.org/wiki/Verlet_integration.*