N-Body Simulations using the Particle-Mesh Method

Daniel Wray

University of Nottingham

December 14, 2018

Abstract

N-body simulations give us the ability to model the universe on scales that would be otherwise impossible. Here I describe and implement the Particle-Mesh method of simulating motion under the influence of gravity. The accuracy of the model is tested: comparing the results of a top hat collapse to the analytical solution, and the effect of periodic boundaries is negated by adding buffer particles to the corners of the simulation. Model universes are simulated to analyse how the formation of structure changes with particles of different masses. It is found that when particle mass is increased, the size of the clusters decreases, and the sharing of particles between the clusters increases.

1 Introduction

The physics of Newtonian gravity is incredibly simple, and can be described by the instantly recognisable law of gravitation. However, when trying to evolve this system in time, a differential equation is formed that is very difficult to solve. This is because the total force being the sum of contributions made by every particle in the system. Relating the total force to the acceleration of a particle this differential equation is written,

$$\ddot{\vec{r_j}} = -G\sum_{i\neq j}^{N} \frac{m_i}{|\vec{r_i} - \vec{r_j}|^2}\hat{r}, \tag{1}$$

where N is the total number of particles in the system. Notice that for N particles this will result in a system of N coupled non-linear second order differential equations. Not only are these equations difficult to solve, other than $N = 0, 1$ which are trivial, $N = 2$ is the only exactly solvable case [1]. The complexity of these systems hindered progress in the field until around 1980, when the advent of computers meant that numerical methods could be used to solve systems with upwards of 100 particles. This was done using the first and most straight forward approach, the Particle-Particle (PP) method. The PP method computes the acceleration of each particle using equation (1), and then particles are propagated forward using an integration scheme such as the Runge Kutta 4th Order, or a symplectic method such as Verlet Integration. This method is inefficient with $O(N^2)$ operations per timestep[4]. Due to this large number of operations, the PP method can only, at least with reasonable speed, simulate a system of less than 1000 particles. For system sizes within this range the PP method is the fastest, however for much larger systems a new more efficient method had to be developed. The Particle-Mesh (PM) method tackles the problem from a slightly different angle by considering the potentials of the system and solving the Poisson equation [3],

$$\nabla^2\phi = 4\pi G\rho, \tag{2}$$

where ϕ is the gravitational potential energy and ρ is the mass density. The efficiency of the PM method is $O(N \log N)$ meaning it can run simulations for particle numbers in the millions. There is a small catch, in that the potential and mass density fields have to be discretised. For a computer to solve equation (2) the fields must be approximated as distributions on a grid. As a result,

[1] There exists solutions for $N = 3$ but these are edge cases that require very specific initial conditions, see [2]

this method suffers inaccuracies on short length scales (relative to the grid size) and is most accurate over large distances, which gives PM applications in the evolution of the universe and the formation of large structures. This method is convenient, as it naturally incorporates periodic boundary conditions, allowing effectively infinite universes to be simulated. It was famously used[2] to prove that the universe must be dominated by a cold dark matter - ruling out neutrinos as a dark matter particle candidate [1].

This project aims to describe the theory behind a PM implementation, then investigate its accuracy by comparing the results of a top hat collapse to the analytical solution. Using the simulation, various model universes will be analysed, to observe the effect particle masses have on the formation of large structures.

2 The Particle-Mesh Method

This method considers the Poisson equation, (2). If we had known the potential and were trying to find the mass density this would be a rather easy equation to solve, just taking the second derivative. However, since we are trying to solve for the potential, this requires a very computationally expensive operation called a convolution. Directly solving using a convolution would be slow, therefore a mathematical 'trick' is used. Convolutions are multiplications in Fourier space. Taking the Fourier transform of (2), the Poisson equation becomes an algebra problem and the potential can be solved for very easily. Then by taking the inverse and transforming back into position space the potential has been solved for, with the most costly operation being a Fourier transform[7]. An outline of the entire PM method could be given as so,

1. Find the mass density.

2. Fourier transform the field.

3. Solve in momentum space.

4. Inverse Fourier transform back to position space.

5. Interpolate the forces onto the particles.

[2]In actuality a slightly modified method, P^3M, that has increased accuracy at short length scales and close encounter interactions was used splitting PM into 5 distinct sections. An implementation of PM would follow this structure closely with slight deviations by user preference.

2.1 Mass Density Distribution

Since the purpose of the PM method is to be efficiently ran on a computer, approximations must be made that are unavoidable due to the nature of computer systems. The mass density in reality is a continuous scalar field, however, for use within PM it must be approximated as a distribution. This requires laying a mesh over the top of the particle system, and summing the contributions made by every particle that lies within one grid cube, then dividing by the volume of each cube for the density. The required resolution of this mesh is a fine balance between the accuracy and speed desired by the user. The most efficient values would be powers of 2, which shall be explained further in §2.2. In my implementation of the algorithm I use a grid size of 64^3, this is low enough for real time simulations and high enough for a sufficient level of accuracy[4].

Another decision which must be made when forming the mass density distribution is the mass assignment scheme. This method is one which the algorithm uses to assign the mass to the grid cubes. The method used in my implementation, and by far the simplest, is the Nearest Grid Point (NGP) scheme[7]. The scheme takes the position of the particle and finds the cube that the particle is contained within, then assigns all of the mass of the particle into that grid cube. This simplistic scheme has issues, see §5, which motivated the development of alternative schemes such as Cloud in Cell (CIC)[4]. This smooths the distribution, by assigning a fraction of mass to the neighbouring grid cubes. This can been seen much clearer visually in Figure (1). An important caveat is that the same scheme used to assign the masses to the mesh must also be used to interpolate the force on the particle. If different methods are used, momentum is no longer conserved and simulation accuracy is lost[4].

2.2 Solving in Fourier Space

To find the potential distribution we solve in Fourier space. My implementation uses a Fast Fourier Trans-

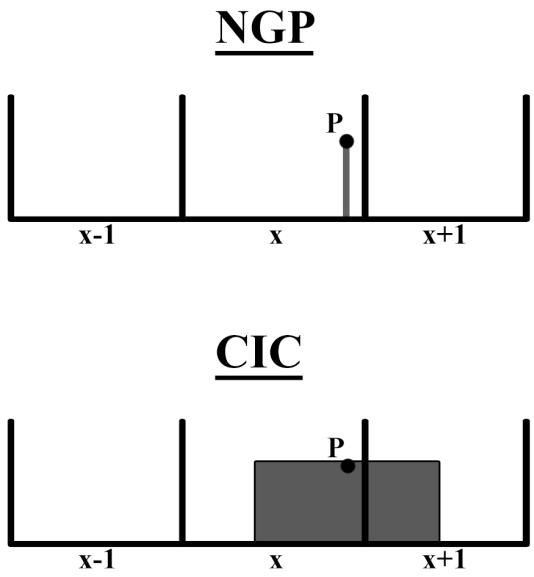

Figure 1: The division of mass into grid points for the NGP and CIC schemes. Notice that in NGP although P is very close to the edge of x all the mass is added to x, whereas in CIC this mass is shared with $x+1$ making the mass density distribution more accurately represent the locations of masses.

form (FFT) routine. A small problem is that the Poisson equation, (2), is continuous in nature and acts on a field. When using a distribution, finite difference methods must be used. In the 1D case this gives

$$\frac{\phi(x+1)-2\phi(x)+\phi(x-1)}{(\Delta x)^2}=4\pi G\rho(x). \quad (3)$$

Taking the Fourier transform of this,

$$\frac{(e^{ik\Delta x}-2+e^{-ik\Delta x})}{(\Delta x)^2}\Phi(k)=4\pi GP(k), \quad (4)$$

where the capital letters represent the Fourier transform of a function. As previously stated, now in Fourier space this equation becomes algebraic, and the potential can be solved for. We find,

$$\Phi(k)=-\frac{4\pi G}{K^2}P(k). \quad (5)$$

with the definition [3],

$$K=\frac{\sin(\pi k\Delta x)}{\pi\Delta x}. \quad (6)$$

It can also be seen that in the limiting case,

$$\lim_{\Delta x\to 0}\frac{\sin(\pi k\Delta x)}{\pi\Delta x}=k. \quad (7)$$

That is, as the spacing between the grid cubes becomes infinitely small, the finite difference method becomes the continuous case, validating the approximation when made for small grid spacing.
Equation (5) is the 1D case, my implementation in 3D solves this equation extended into all 3 degrees of freedom,

$$\Phi(k,l,m)=-\frac{4\pi G}{K^2+L^2+M^2}P(k,l,m). \quad (8)$$

where k, l and m are the wave vectors in 3D Fourier space. All that remains is to take the inverse Fourier transform of (8) to find the potential in position space. In my implementation I chose again to utilise the FFT algorithm (more accurately, an iFFT at this stage) for efficiency. Efficiency becomes very important in 3D, as for a single distribution the FFT must be ran for each individual row and column etc, which vastly increases the operation count. The FFT has maximum efficiency when the number of grid points is a power of 2. Since the bulk of this simulation relies on an FFT, simulations that have a mesh with 2^n points in each dimension will vastly decrease computation time.

2.3 Interpolating Forces

It was stated in §2.1 that the scheme used to assign mass to the mass density distribution must also be used to calculate the force on a particle, in order to conserve momentum. This only requires that the same fraction of potential

[3] This is sometimes defined as $K = k\,\mathrm{sinc}\,(\pi k\Delta x)$, however I found in my implementation that the sinc function was more computationally expensive than a well optimised sin function with a lookup-table. Therefore, I preferred the definition in equation (6) for efficiency.

from each grid cube summed is the same as the fraction of mass distributed. For NGP, as used in my implementation, all the mass is assigned to a single grid cube, so this fractioning of potential can be ignored.
Thus far, we have found the potential at every point on the mesh, to move the particles forward the force must be found. We can use the relationship,

$$\vec{F} = -\nabla\phi. \tag{9}$$

and then for the acceleration,

$$\vec{a} = -\frac{1}{m}\nabla\phi. \tag{10}$$

A similar problem to that in §2.2 arises here as equation (10) is valid for a field and our potential is a distribution. Finite difference methods must but employed again to approximate the derivative. A naive approach would be to use,

$$a_x = -\frac{(\phi_{x+1} - \phi_x)}{m\Delta x}, \tag{11}$$

in each dimension. Here, the subscript denotes which grid cube the value of the potential should be taken from. However, this leads to the problem of self-acting forces. A self-acting force is caused, which causes the particle to feel the force of its own presence. (11) uses grid cube x which is the same grid cube that the particles own mass contributes to, meaning that if there was only one particle in the simulation the particle would still feel a force. The solution to this is rather simple. Instead of the grid cube that contains the particle being used the cubes either side are used. Due to the isotropy of the gravitational force, the force caused by the particle itself is exactly cancelled out and only contributions made by other particles remain. Mathematically this is written[7],

$$a_x = -\frac{(\phi_{x-1} - \phi_{x+1})}{2m\Delta x} \tag{12}$$

Using this, the acceleration in all directions can be correctly determined and used to update the particle position. The integration scheme my implementation uses is a very simple first order solver, which assumes constant acceleration and velocity between each time-step. This is sufficient, as the goal of the PM method is to simulate large scale motion. This integration scheme can be written,

$$\begin{aligned} \vec{v}_{t+1} &= \vec{v}_t + \vec{a}\Delta t, \\ \vec{r}_{t+1} &= \vec{r}_t + \vec{v}_{t+1}\Delta t \end{aligned} \tag{13}$$

Note that the velocity is updated before the position on each loop. Once the positions of the particles have been updated the entire process must be repeated again for every time-step into the future[3].

3 Accuracy of the Model

Before any science is done with the simulation, it is important to check the method and it's implementation produce physically accurate results. The easiest way of verifying the accuracy is to run a simulation of a problem with a known solution and compare the results. The first check I made was to verify that the potential around a single particle produced the characteristic k/r curve. This can be done by taking a 2D slice out of the potential distribution that crosses the particle. The result can be seen in Figure (2). Note that unlike the PP method PM naturally avoids singularities when $r = 0$. This isn't for free, however, and can cause inaccuracies as a result, see §5.

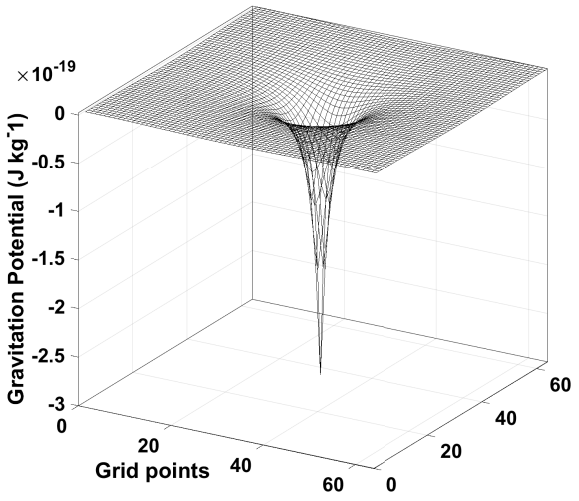

Figure 2: The potential distribution caused by a single particle of unit mass. This is a slice through the 3D distribution with one coordinate held at grid point 31 (of 64).

Once my implementation of PM gave physical results for a single particle it was time to check the accuracy of

the simulation with multiple particles. One many particle system with a known solution is the top hat collapse. The name refers to a 'top hat' shape made by the mass density distribution, however in reality this system looks like a spherical cluster of particles. The analytical solution can be found by solving Newtons law of gravitation,

$$\frac{d^2r}{dt^2} = \frac{GM}{r^2}, \tag{14}$$

directly using integration. The solution to (14) is a parametric curve that depends on the total energy of the system[9]. The curve predicts that the top hat will initially expand before collapsing back down, however, by setting the initial velocity of the particles to zero this expansion can be ignored. Figure (3) shows the simulations of this system. At first sight this simulation seems underwhelming, there is a large difference between the results and the analytical model. The error here is caused by the periodic boundary conditions of the system. The parts of the top hat closest to the edges feel more force as they are closer to the other side of the top hat periodically. A fix is to add buffer particles in the corners of the system. These particles add a small extra force to the rest of the top hat slowing down the collapse, so that the force is even in all directions. This small decrease is enough to cause the improvements seen in Figure (4). The agreement of these curves justifies the accuracy of the PM method in this context and more specifically my implementation of the method.

The data points and associated uncertainties in these simulations were calculated by repeating the simulation 10^3 times, and calculating the mean position and the error on the mean. The error increasing with time I suspect is due to the inaccuracies of the previous time-step being compounded.

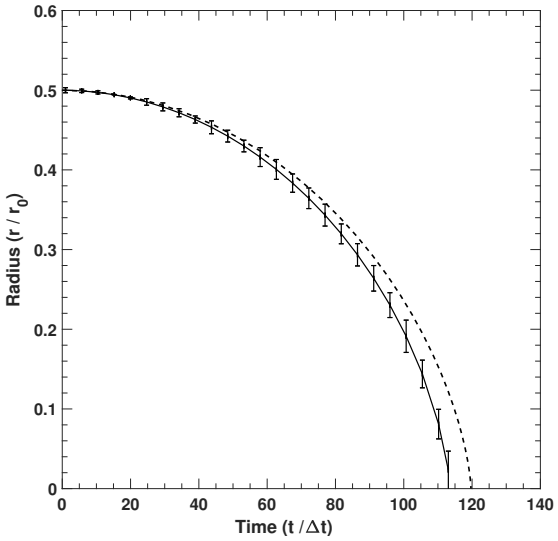

Figure 3: The collapse of a spherical top hat's radius (as a fraction of the simulation space) against the number of time-steps. The dashed line represents the analytical solution.

4 Formation of Structure

Taking advantage of the periodic boundaries, a universe that is effectively infinite in size can be simulated. After a large number of time-steps this 'effective' infinity is no longer valid - as the clumps of mass begin to cluster[8]. However, for the shorter time scales of structure formation the simulation is still valid.

The first simulation I ran was for 32^3 particles which can be seen in Figure (5), where the formation of small clusters can be clearly observed. To create a larger contrast with the background, this simulation was ran again with 64^3 particles. The result is displayed in Figure (6), with the clusters being much more noticeable. This simulation was also left to run for many more time-steps resulting in a large cluster, which demonstrates the instability previously mentioned. The final result of this simulation can be seen in Figure (7). A simulation very similar to the previous was then conducted, with the mass of the particles increased by a factor of 10^4. The result had much more structure than previously seen, but these were much smaller in size, with some clusters forming strands of particles between neighbouring clusters. The strands presumably appear due to the wider potential-well caused by the higher mass clusters, and the shorter inter-cluster distances, making it easier for the clusters to share particles. The result of this simulation is in Figure (8).[4]

[4] All the simulations that produced the figures in this section used 'computer units'[6]. I chose to do this as the structure was the focus not a numerical output.

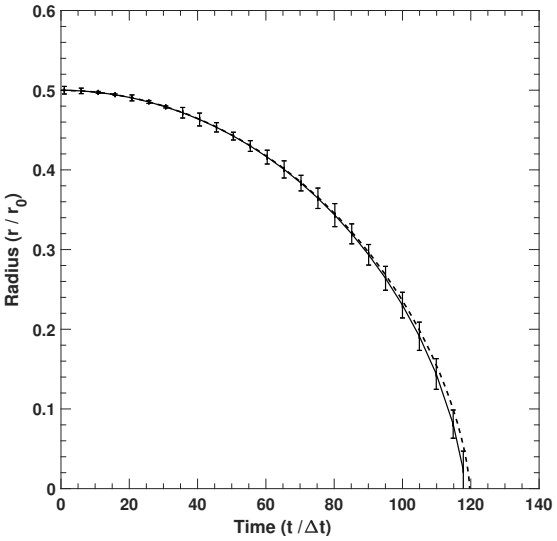

Figure 4: The collapse of a spherical top hat with buffer particles in the outer corners to cancel the effects of the periodic boundaries. The dashed line represents the analytical solution.

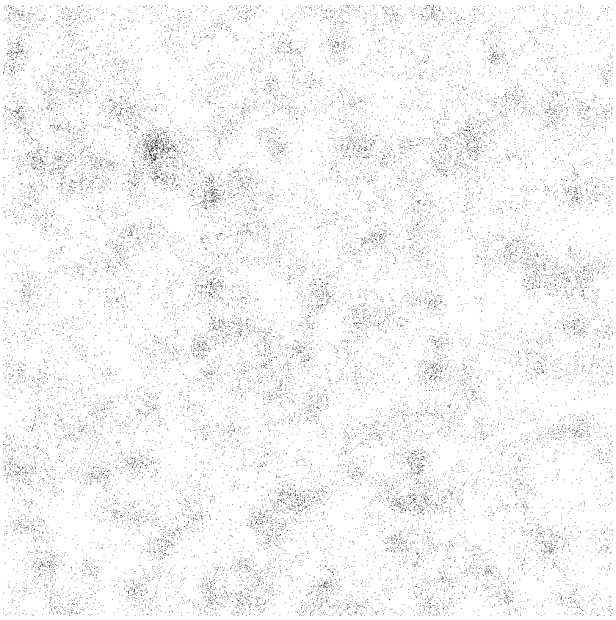

Figure 5: A simulation of 32^3 particles. Faint structures can be observed.

5 Shortcomings

Overall, the PM method isn't perfect, with various shortcomings that I would like to highlight. For instance, the method does not take into account close-encounter interactions, rendering it almost useless at small length scales, or in any system where the particles are likely to collide. This is because the equation that the algorithm solves is not exactly equation (14) but rather a slightly modified version[4],

$$\frac{d^2r}{dt^2} = \frac{GM}{r^2 + \eta}, \tag{15}$$

where η is a smoothing constant. It's clear that for $r^2 \gg \eta$ the algorithm will make a great approximation of (14) but as they become comparable the results will differ.

A related issue is the selection of the mass assignment scheme. CIC does a much better job of smoothing the mass density distribution than NGP, but if the scheme chosen uses too much smoothing the value of η changes causing the inaccuracies[4]. I chose to use NGP in my implementation, due to its simplicity, as well as slightly better results at shorter lengths scales, however, the scheme has some problems that extend further than the issue of smoothing. Since all the particles within a grid cube will experience an equal acceleration, at the edges of a grid cubes, lines of particles can coagulate as they all merge with the neighbouring cube at once. In CIC because the relative distance within the grid cube is taken into account, accelerations are not equal and this does not occur.

6 Summary

My implementation of the PM method successfully produced physical simulations of particles acting under the influence of gravity. It was able to recreate the potential-well produced by a single mass and model a top hat collapse of multiple thousands of particles. The model, as expected, deviated when performing the top hat collapse due to interactions with itself through the periodic boundaries. This was then accounted for by adding buffer particles that would slow the over accelerated sections of the top hat. This gives a curve which closely agrees with the

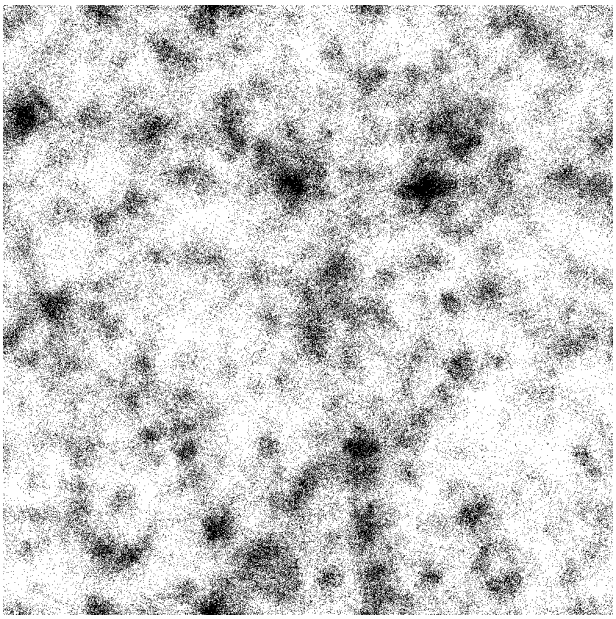

Figure 6: This simulation has the same settings as Figure (5) only it was ran with 64^3 particles. The structures can now be seen with much higher contrast to the background.

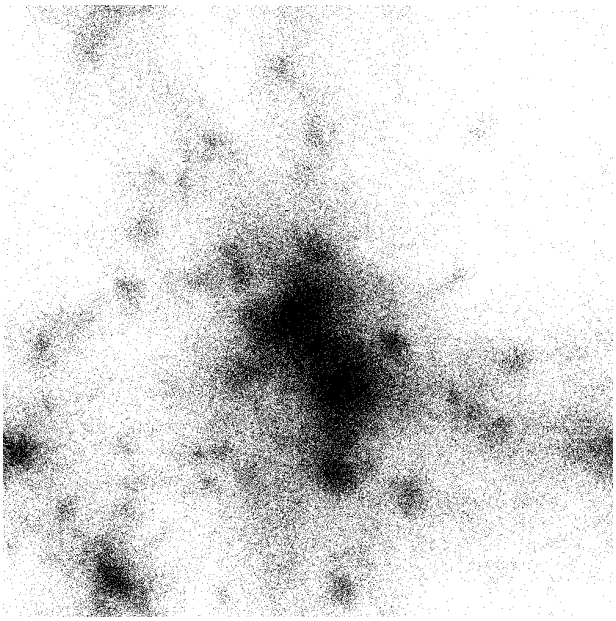

Figure 7: This is Figure (6) after an extended number of time-steps, demonstrating that eventually the structure becomes unstable.

analytical result, with an expected amount of deviation due to the inaccuracies associated with close-encounter interactions. Random distributions of particles were then evolved in time to investigate the structures that form, and the differences in these structures for different masses was analysed.

To reflect, I would have made two changes I would have made to my program. The first being the implementation of a CIC scheme. I underestimated the effect that the mass assignment scheme would have on the simulation accuracy. If had more time, I would have updated the program to use a scheme with more smoothing than NGP. The second would be to investigate the initial conditions of my simulations in more depth. From reading papers such as [1][5] I noticed the large amount of effort put into the initial conditions of the simulations. This, of course, makes perfect sense as the universe is not a random distribution of particles, as my simulations assumes. These subtle changes in the initial conditions would affect the overall evolution of the structures formed.

References

[1] M. Davis, G. Efstathiou, C. Frenk, and S. White. *The Evolution of Large-Scale Structure in a Universe Dominated by Cold Dark Matter*. 1984.

[2] J. Frank. *The Three-Body Problem*. 2006.

[3] R. Gonsalves. *Computational Physics I*. 2009.

[4] R. Hockney and J. Eastwood. *Computer simulation using particles*. 1988.

[5] A. Jenkins et al. *Evolution of Structure in Cold Dark Matter Universes*. 1998.

[6] A. Klypin and J. Holtzman. *Particle-Mesh code for cosmological simulations*. 1997.

[7] K. Lemmens. *An investigation, implementation and comparison of 3 important particle simulation techniques*. 1997.

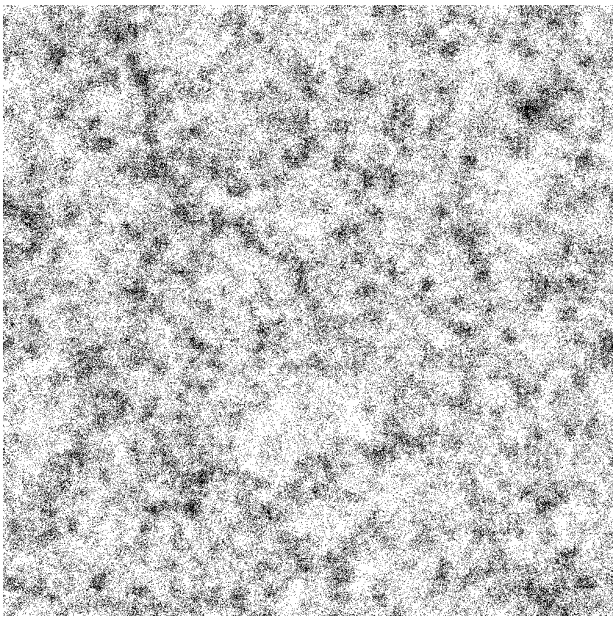

Figure 8: This simulation has 64^3 particles, which have a mass 10^4 times greater than those in the simulation of Figure (6). The structure formed here is noticeably smaller, however the evolution of strands between the clusters can be observed.

[8] R. Teyssier and O. Agertz. *Particle Mesh methods*. 2009.

[9] F. van den Bosch. *Non-Linear Structure Formation*. 2013.