Chapter 2 Computational Cosmology

2.1 Introduction

The past few decades have seen huge advances in our understanding of the formation and evolution of cosmic structure. Recent galaxy surveys (see e.g. Colless et al. (2001); York et al. (2000)) have allowed us to quantify the structure of the local Universe and theoretical models have allowed us to explain how these structures grew from the first few moments of the Universe's existence to the present day. Due to the huge timescales involved in the formation of large scale structure, computer simulation has become one of the most important methods for the investigation of the formation and evolution of the largest scale cosmic structure. In order to perform reliable simulations is important that we understand the relative strengths and weaknesses of different simulation algorithms. In adition it is also important that we have a reliable way of generating the initial conditions for large scale structure simulations. This chapter is concerned with all of these things and is split broadly into two sections.

In the first section we introduce and describe the most common cosmological simulation techniques before performing a series of tests of the reliability of simulation codes by carrying out a comparison of the results obtained by two different simulation techniques against one of the standard cosmological tests, the Santa Barbara cluster comparison test (Frenk et al. (1999))

In the second half of this chapter we discuss the techniques by which cosmological initial conditions can be generated and introduce a code for the generation of high resolution cosmological initial conditions in which additional small scale power can be added to large scale modes. We use this code to present a detailed study of the effects of mass resolution on a set of simulated galactic haloes.

2.2 Simulation Techniques

In cosmological simulation the matter in the universe is usually modelled in two seperate components: firstly a collisionless fluid that represents dark matter or stellar material and secondly a baryonic component (primarily hydrogen and helium). Both of these phases feel the gravitational force, the gas phase is in addition subject to hydrodynamic forces. In order to turn the problem of the solution of the hydrodynamics equations into one that can be represented numerically we need to discretize the fluid properties that we wish to calculate. Broadly speaking hydrodynamic simulation codes may be broken down into two classes 'Eulerian' (discretize space and represent gas properties on a grid) and 'Lagrangian' (discretize mass and represent fluid properties in individual particles).

Hydrodynamic forces may be evaluated in a large number of different ways. The most commonly used Lagrangian codes employ the Smoothed Particle Hydrodynamics (SPH) (Gingold and Monaghan, 1977; Monaghan, 1992) method, and the most common Eulerian codes use the Piecewise Parabolic Method (PPM), first developed by Colella and Woodward (1984) (see also Woodward and Colella (1984)). Lagrangian simulation codes have been implemented in a cosmological framework by various authors, first on a static grid (Cen (1992)) and later on a deformable grid allowing a spatially variable mass resolution Bryan et al. (1995); Fryxell et al. (2000); Teyssier (2002); Quilis (2004). Although not discussed further in this thesis it is worth noting that other hydrodynamic methods do exist, including HPM, and the smoothed lagrangian hydrodynamics codes due to Gnedin (1995).

Most simulation codes calculate the gravitational forces (through solution of the Poisson equation) in one of a limited number of ways: either through the use of fast Fourier transforms on grids (PM methods). see e.g. (Klypin and Shandarin, 1983; White et al., 1983; Efstathiou and Eastwood, 1981); hierarchical trees (Barnes and Hut (1986); Jernigan and Porter (1989)); mesh relaxation methods (Brandt (1977)), or direct summation of the pairwise forces (see, e.g. (Aarseth et al., 1979; Frenk et al., 1983)). Each of these methods has its own advantages and disadvantages. PM codes are extremely fast but spatial resolution is limited by the size of the spatial grid. Other methods are more computationally expensive but allow forces to be resolved on much smaller scales. Hybrid codes, for example P^{3}M (Efstathiou and Eastwood, 1981; Couchman et al., 1995; Wadsley and Bond, 1997) or Tree-PM (Wadsley et al., 2004; Springel, 2005) can combine the best features of each of these schemes. As in the case of the hydrodynamic calculation we will discuss only the most commonly used methods of calculating the gravitational force, neglecting interesting but rarely used techniques such as the approach due to Widrow and Kaiser (1993) of reformulating the Schroedinger equation to describe the dynamics of collisionless matter.

In the remainder of this section we will discuss in more detail a few of the methods by which the gravitational and hydrodynamic forces may be calculated for arbitrary distributions of matter.

2.2.1 The Gravity Calculation

For any continuous density field $\rho(\mathbf{r})$, we can calculate the associated gravitational potential via the Poisson equation

$$\nabla^2\Phi(\mathbf{r}) = 4\pi G\rho(\mathbf{r}), \tag{2.1}$$

where Φ represents the gravitational potential and G is Newton's gravitational constant. The gradient of the potential as obtained from 2.1 can then be obtained to calculate the gravitational acceleration at any point. In cosmological comoving coordinates the Poisson equation becomes (Peebles (1981))

$$\nabla^2\Phi(\mathbf{r}) = 4\pi G a^2[\rho - \bar{\rho}]\,. \tag{2.2}$$

General relatavistic effects can be neglected in cosmology due to the fact that the Newtonian limit is applicable on scales smaller than the Hubble length (c/H) and larger than the Schwartzchild radii of any collapsed objects. In order to approximate a continuous field computationally we need to discretise the phase space distribution of matter, $f(x, y, z, v_x, v_y, v_z)$. This is usually achieved by subsampling the true phase-space distribution and representing it with enough discrete particles to capture the important features of the underlying phase space distribution.

In this section we will discuss four different ways of calculating the gravitational force: The direct summation, using Fourier transforms on a regular grid, using a hierarchical tree and finally using iterative methods on a (possibly irregular) grid. Finally these four techniques are compared and contrasted.

Particle Particle Methods

The most conceptually simple way of calculating the gravitational force on a point mass due to a system of other point particles is the particle-particle (PP) direct summation method. This method, used in the first ever simulations of cosmic structure formation (Aarseth (1963); Hénon (1964), works by explicity summing the gravitational force exerted on each particle by each other particle directly via Newton's law of gravitation

$$\vec{F} = \frac{GMm}{(r^2 + \epsilon^2)^{3/2}}\vec{r}, \tag{2.3}$$

where ϵ represents a gravitational softening term, which acts to flatten off the gravitational force at small radii ($r < \epsilon$) and so diminish the effect of two-body relaxation.

PP codes solve exactly for the gravitational force on each particle, the disadvantage of these codes is that the number of computations to calculate the gravitational force on N particles scales as N^2, so these codes are unfeasible for use in even moderately large simulations.

Particle Mesh Methods

One method of cutting down the number of operations needed to calculate the gravitational force is to calculate potentials on a regular mesh and then interpolate the forces back to the particles.

The process of using a mesh to calculate the gravitational forces on a series of discrete mass elements consists of three seperate steps:

1. Mapping the discrete particle masses on to a uniform grid
2. Solution of equation 2.1 to obtain the potential on the grid
3. Differencing of the potential on the uniform grid at the location of each particle to obtain the gravitational force.

In this section we will discuss each of these processes in turn. For the sake of simplicity we will assume that we have a cubic volume of side length L, which contains N_p discrete particles. The mass of the ith particle is m_i. The mass distribution at any given three dimensional coordinate, $\mathbf{r}$ may then be described by

$$m(\mathbf{r}) = \sum_{i=0}^{N_p} m_i \delta(\mathbf{r} - \mathbf{r}_i)\,. \tag{2.4}$$

If we place a uniform cubic grid over the volume then using equation 2.4 we can calculate the mass density at a gridpoint with coordinate $\mathbf{r}_p$ by

$$\rho(\mathbf{r}_p) = \frac{1}{\Delta^3} \sum_{i=0}^{N_p} m_i W(\mathbf{r}_i - \mathbf{r}_p)\,, \tag{2.5}$$

where Δ represents the grid spacing and W is a function that represents the charge assignment scheme that is in use. In order to simplify the mathematics we will demonstrate a number of charge assignment functions in one-dimension but note that multidimensional versions of the assignment functions can be obtained by multiplying together the corresponding one-dimensional functions for each component. The simplest solution is to assign all of a particle's mass to its nearest grid point (NGP), mathematically this is represented by

$$W(\mathbf{x}) = \begin{cases} 1 & \text{if } |x| < \frac{\Delta}{2}; \\ 0 & \text{otherwise}\,, \end{cases}$$

where $x = x_i - x_p$. The NGP scheme introduces lots of noise on small ($< \Delta$) scales, where the assigned charges will flip from 0 to their maximum value as particles cross grid boundaries. A more accurate solution is obtained by using the cloud in cell (CIC) scheme, which spreads a particle's mass over the nearest two grid points in each direction. This scheme may be described

$$W(\mathbf{x}) = \begin{cases} \left(1 - \frac{|x|}{\Delta}\right) & \text{if } |x| < \Delta \\ 0 & \text{otherwise}\,. \end{cases}$$

This is more computationally expensive than the NGP scheme but the resulting mappings are more accurate. More complex still is the triangular shaped cloud (TSC) scheme, where each particle's mass is spread over even more grid points and the resulting density is even less noisy. TSC is described by

$$W(\mathbf{x}) = \begin{cases} \frac{3}{4} - \left(\frac{|x|}{\Delta}\right)^2 & \text{if } |x| \le \frac{\Delta}{2}; \\ \frac{1}{2}\left(\frac{3}{2} - \frac{|x|}{\Delta}\right)^2 & \text{if } \frac{\Delta}{2} < |x| \le \frac{3\Delta}{2}; \\ 0 & \text{otherwise}\,, \end{cases}$$

where the values of the coefficients are determined by requiring that the function and its derivative are continuous and that it integrates to unity. Figure 2.1 shows the difference between the three mass assignment schemes in one dimension. The choice of a charge assignment scheme, therefore, depends upon a number of factors. High order schemes are more computationally expensive to calculate, but reduce the amount of noise in the

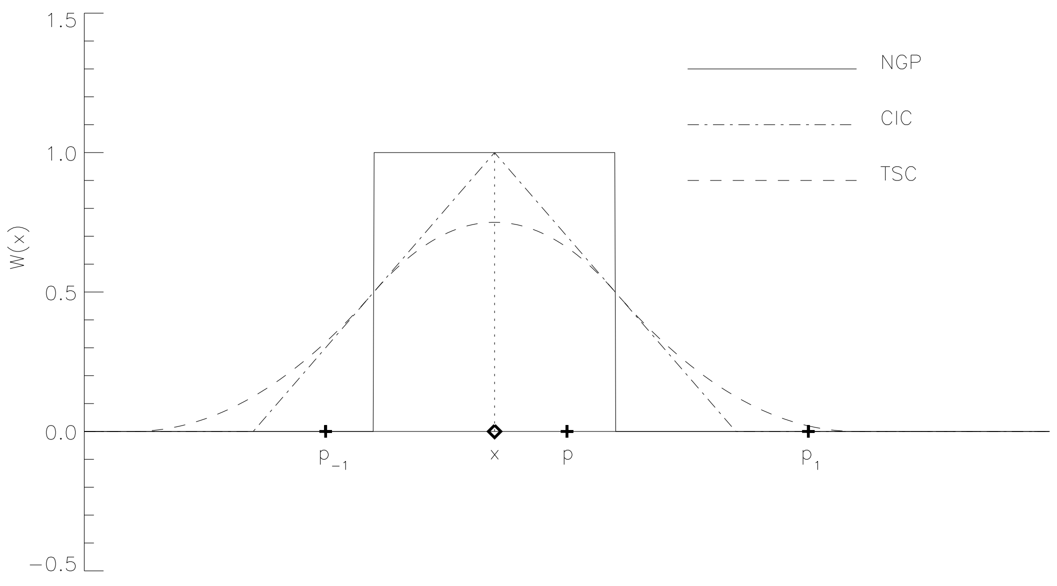


Figure 2.1: The weighting functions for three mass assignment schemes in one dimension. The amount of mass that is assigned to each gridpoint is proportional to the value of the assignment function at the gridpoint. Considering a particle at point x with neighbouring grid points p_{-1}, p and p_1 the NGP method assigns all mass to gridpoint p, CIC assigns to the nearest two points (p and p_1). TSC assigns mass to the nearest three points.

resulting solution. Low order schemes require few operations to calculate, but increase the amount of noise in the result.

The second step in computing the gravitational force is to solve equation 2.1 on the uniform grid. With the advent of rapid methods for performing Fourier transforms in the 1960s (Cooley and Tukey, 1965), fast Fourier transform (FFT) methods, the most computationally efficient method of solving this equation became to work in k-space.

To begin we need to note that the Poisson equation is linear, that is, the potential at any mesh point (p, q, r) can always be written as the sum of contributions from all other source points (p', q', r'), or

$$\phi_{p,q,r} = \sum \mathfrak{G}_{p-p',q-q',r-r'} \rho_{p',q',r'} \, . \tag{2.6}$$

Equation 2.6 expresses the potential as the convolution of the source distribution ρ with some function that describes the interaction, usually called the Green's function, $\mathfrak{G}$.

Now, via the convolution theorem, the statement that (see e.g. Boas (1983))

the Fourier transform of a convolution is equal to the piecewise products of the Fourier transforms of the quantities involved

we can rewrite equation 2.6 in the form

$$\hat{\phi}_{k,l,m} = \hat{\mathfrak{G}}_{k,l,m}\hat{\rho}_{k,l,m}\,. \tag{2.7}$$

In order to solve the Poisson equation we therefore need to identify the form of the Green's function that describes the interaction. To do this we note that the Greens function for a particular differential equation is defined to be its solution under the influence of a delta function (Boas (1983)). That is:

$$\nabla^2\mathfrak{G}(t,t') = \delta(t'-t)\,, \tag{2.8}$$

where $\delta(t-t')$ represents a Dirac delta function centered at t'. Using the Green's function we can then try to find a solution to the Poisson equation for a real density field by adding up the contributions from point sources at each gridpoint. The Green's function in this case is described by(Hockney and Eastwood (1988))

$$\hat{\mathfrak{G}}_{p,q,r} = -1/\Big[\pi(p^2+q^2+r^2)\Big]\,. \tag{2.9}$$

The form of the Green's function changes when we define the density only on a grid. The Green's function of the seven-point finite-difference approximation to the laplacian is

$$\hat{\mathfrak{G}}_{p,q,r} = -\frac{\pi}{\Delta^2}\frac{1}{\Bigg(\Big[\sin^2(\pi p\Delta)+\sin^2(\pi q\Delta)+\sin^2(\pi r\Delta)\Bigg)}\,. \tag{2.10}$$

See appendix B for details on how this form of the Green's function is derived. Substituting equation 2.10 into equation 2.7 allows us to obtain the potential at every point in space by Fourier transforming the density field, multiplying with $\mathfrak{G}_{p,q,r}$ and then taking an inverse Fourier transform. The gradient of the potential at the position of each mass element may then by approximated by finite differencing of the potential

$$\left.\frac{\partial\phi}{\partial x}\right|_{i,j,k} = \frac{1}{2\Delta}(\phi_{i+1,j,k}+\phi_{i-1,j,k}-2\phi_{i,j,k})\,, \tag{2.11}$$

which is accurate to first order. The spatial resolution available through FFT methods is limited by the spatial size of the FFT grid (as demonstrated in figure 2.2). To obtain gravitational forces on scales below that of the FFT grid we need to apply additional methods.

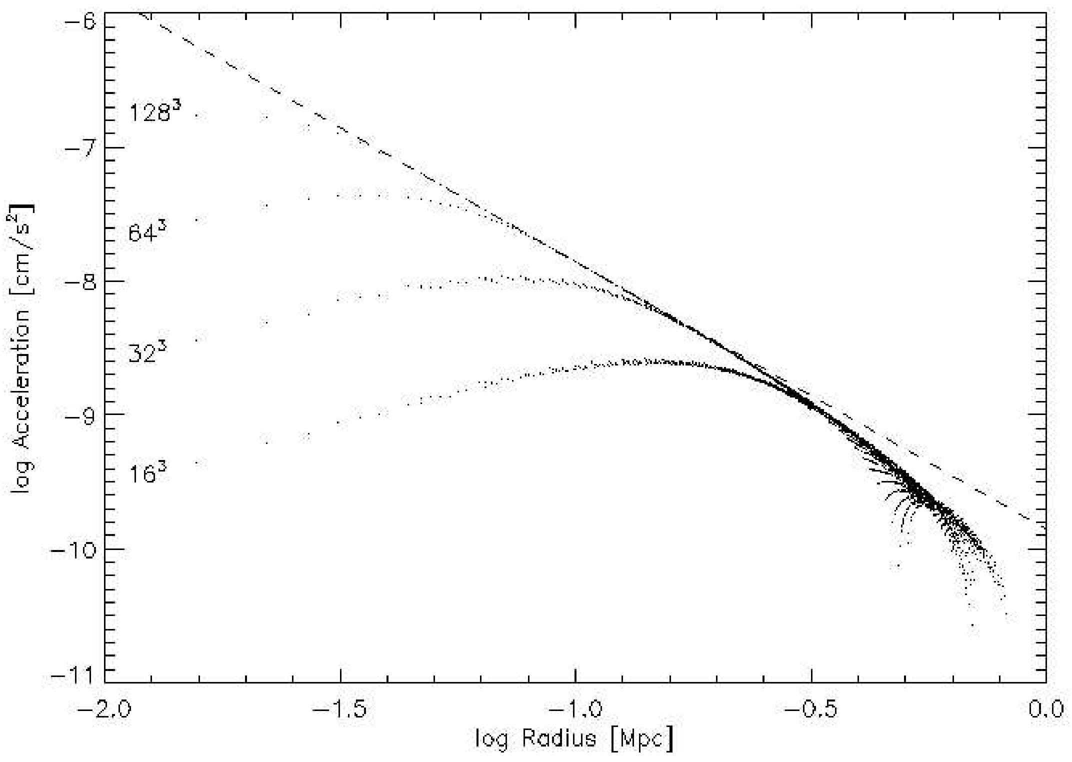


Figure 2.2: Gravitational acceleration of a series of test masses ($m = 0$) due to a single point mass in the centre of a periodic box. The dashed line shows the potential from a point mass, ignoring the periodic images of the massive particle (GM/r^2). Each series of black points represents the gravitational acceleration of a single particle as performed with a different number of PM cells (as marked on the graph). The deviations from a pure $1/r^2$ law at high radii are due to the gravitational influence of the periodic images of the point mass. The deviations at low radii are due to the finite PM grid size. The higher the resolution of the PM mesh, the more accurate the gravitational force at small scales.

Tree Methods

As an alternative to Fourier techniques are the so-called tree methods. In these schemes, the particles are arranged in a hierarchy of groups. In order to cut down on the number of computer operations when compared to a PP code, distant particles in the tree are then treated in groups. The process by which a tree is constructed and then used to calculate the gravitational force is discussed in this section.

There are a multitude of different ways of organizing the particles in a volume into a hierarchy, including the Barnes and Hut octree (Barnes and Hut (1986)), trees based on nearest-neighbour pairings (Jernigan and Porter (1989)) and binary trees. The methods by which two different types of tree-structure can be unambiguously created is discussed here.

Constructing a Barnes-Hut tree: In this scheme, starting with the entire computational domain, the volume is recursively partitioned into a sequence of cubes. Each time a cube is found to contain more than one particle it is split into eight equal volume children. This process is demonstrated for two-dimensions in figure 2.3.

Constructing a Jernigan-Porter tree: (Jernigan (1985)) Given a list of N particles the two which are closest together in physical space are joined together to form a node. These two particles are replaced by a single centre of mass node. This process is repeated indefinitely until only one centre of mass node remains.

The detailed shape of the hierarchical tree affects only the storage requirements and efficiency of the gravitational calculation, the actual mathematical details are independent of which tree construction method is used, so the following presentation of techniques for calculating the gravitational force given a hierarchical tree structure is independent of precisely which tree is used.

The gravitational force on a single particle is calculated via the following method: Starting from the largest node we apply the criterion

$$r > \frac{l}{\theta}, \tag{2.12}$$

where r is the distance between the particle we are considering and the current node of the tree. l is the spatial extent of the node and θ is an accuracy parameter. If this inequality is satisfied then the tree node is distant enough that the tree walk along this branch may be terminated and the force from this node added to the total force on the particle. If, on the other hand, the inequality is not satisfied, the node is opened up and its child nodes are evaluated. This process is repeated recursively until every particle

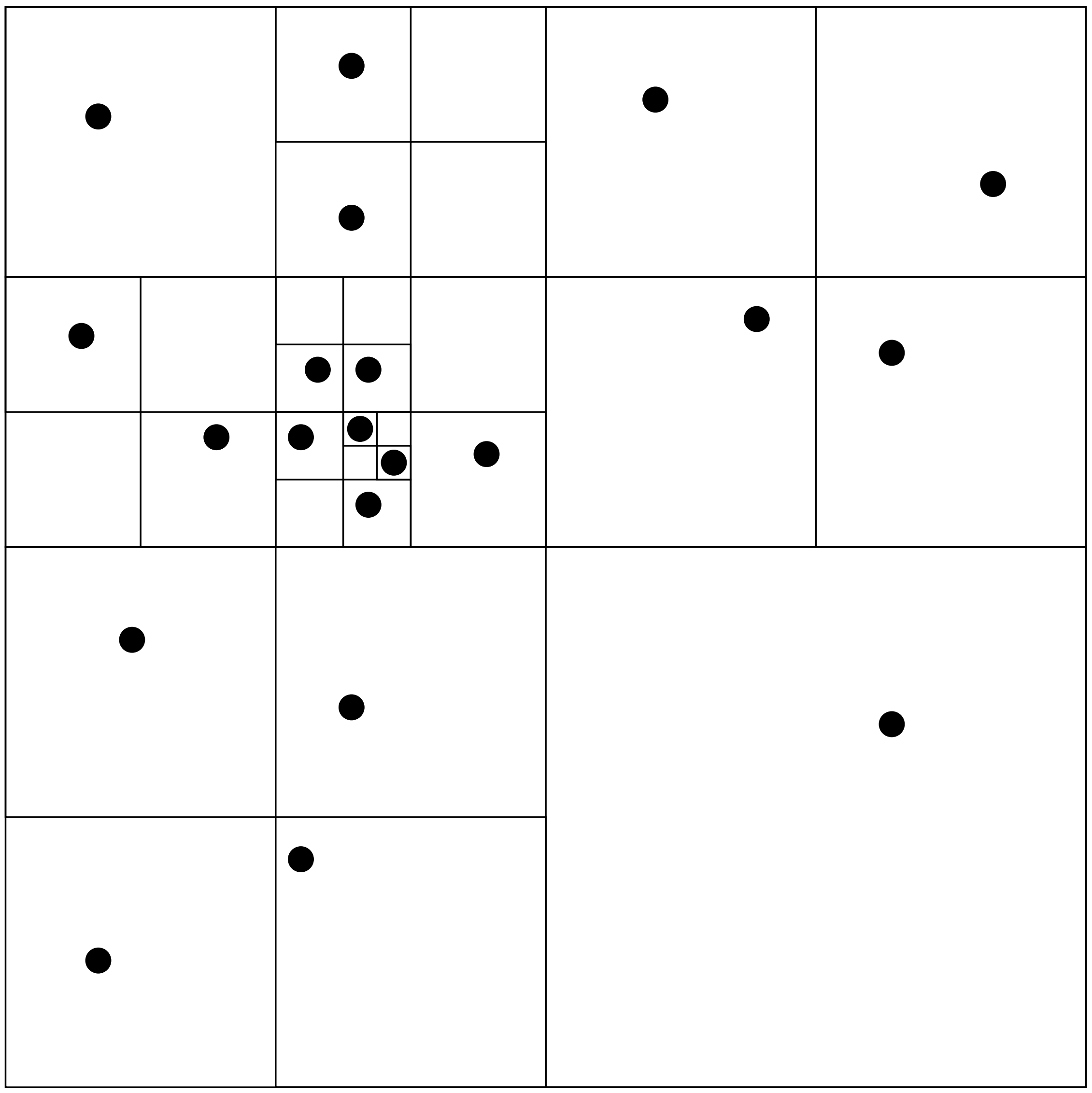


Figure 2.3: Simple depiction of a two dimensional version of an octree, the quadtree. Each black circle represents a single particle and each square represents a node on the tree. The quadtree is constructed by starting with the entire computational volume contained within one node and then recursively splitting each node into four equal pieces if it contains more than one particle. Here we see six different levels of the tree.

has been allowed to contribute to the gravitational force.

The force on a particle from any node can be calculated to first order by

$$\vec{F} = \frac{GMm}{(\mathbf{r}^2 + \epsilon^2)^{3/2}} \vec{r}, \tag{2.13}$$

where r is the distance between the centre of mass of the node and the particle. M and m are the mass of the node and the particle respectively and ϵ is the Plummer softening of the gravitational potential. More accurate forces can be obtained at the cost of more computations per node by including higher order corrections to the gravitational force.

Consider the distribution of matter inside any node of volume $\mathfrak{V}$, its effect on the potential and acceleration field outside $\mathfrak{V}$ is given by

$$\phi(r) = \int_{\mathfrak{V}} \mathfrak{G}(\mathbf{r} - \mathbf{x}) \rho(\mathbf{x}) d^3x \,, \tag{2.14}$$

where ρ is the mass density of matter in the volume and $\mathfrak{G}$ is the Green's function of the interaction. In the case of gravity we use the Green's function of the Laplacian, equation 2.9. The analytic form of the high order corrections for the gravitational force has been described in detail by Salmon and Warren (1994), who derived the analytic forms of the high order corrections all the way up to the hexidecapole.

It is worth noting that by this method we obtain only an approximation to the true force. However, the discrepancy between the calculated force and the actual force obtained by direct summation may be made arbitrarily small by tuning the tree opening criterion, θ (how deeply through the tree we walk when doing the calculation), and altering the order to which the gravitational potential is calculated for each cell (how accurately we evaluate the acceleration due to each node).

There is currently disagreement between various authors about the most efficient way to evaluate a tree force. In the original treecodes of Springel et al. (2001) the potential was expanded out to the quadrupole term, however in the more recent versions Springel (2005), it was found to be more efficient to use only the monopole term, and to walk deeper into the tree. In contrast the treecode of Wadsley et al. (2004) used the hexadecopole moment for each node, and found it more computationally efficient to use relatively few tree nodes. Hernquist (1987) and Barnes and Hut (1989) both find that the use of the quadrupole moment may increase the efficiency of the tree calculation.

Mesh Relaxation Methods

As an alternative to using Fourier methods for solving the Poisson equation on a grid, iterative methods may be used. These methods are of great use in eulerian codes (e.g.

FLASH Fryxell et al. (2000)) and have definite advantages when used in conjunction with adaptive mesh refinement (AMR) (see section 2.2.2), in which gravity needs to be solved on a grid that is not necessarily uniform in space.

The solution of the Poisson equation on a mesh via relaxation method proceeds in three steps

1. Map the discrete particles to the mesh
2. Iterate to find the correct gravitational potential
3. Difference the potential and map back to the particles to obtain the gravitational force.

The first of the three steps proceeds in an identical manner to that discussed in section 2.2.1. The particles are mapped on to the grid using usually either the NGP, CIC or TSC schemes. Considering a two-dimensional grid (grid spacing Δ), we can write that

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}\approx\frac{\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}}{\Delta^2}+\frac{\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}}{\Delta^2}=\rho_{i,j}\,. \tag{2.15}$$

This equation can be rearranged into a form we can work with iteratively

$$\phi_{i,j}\approx\frac{1}{4}\left(\phi_{i-1,j}+\phi_{i+1,j}+\phi_{i,j-1}+\phi_{i,j+1}+\Delta^2\rho_{i,j}\right). \tag{2.16}$$

Equation 2.16 forms the basis for the Jacobi method of solving for the gravitational potential on a grid. The full method is as follows: Take a guess at the gravitational potential, usually this takes the form of the gravitational potential calculated on the last timestep, or if no potential is known it can be zeroed out. Now loop over every gridpoint applying equation 2.16 to obtain an improved estimate of the gravitational potential at that coordinate. Keep on iterating until the solution has converged to the correct potential. It should be noted that it is hard to define exactly what is meant by convergence. One commonly used measure of the error in the potential is the residual, defined as

$$R(\mathbf{x})=\nabla^2\phi(\mathbf{x})-\nabla^2\phi(\mathbf{x})=\rho(\mathbf{x})-\nabla^2\phi(\mathbf{x})\,, \tag{2.17}$$

where ϕ represents our current best estimate of the gravitational potential, as obtained by iteration. When $\sum R(\mathbf{x})^2$ drops below some critical tolerance parameter then the potential is said to be converged.

One improvement to the Jacobi method is to use Gauss-Seidel iteration. Computationally Gauss-Seidel iteration proceeds in exactly the same way as Jacobi iteration, except the values are updated as soon as they are calculated. It can be shown (Hockney and Eastwood (1988)) that this leads to a large reduction in the number of computer operations needed to converge the solution.

An additional improvement comes about when we consider the parallelisation of the iteration problem. As presented so far, the standard Gauss-Seidel scheme cannot be run easily in parallel since each calculation depends upon the results obtained from the previous one. One way to get around this is to use the so-called 'red-black' ordering of points. This ordering is demonstrated in figure 2.4. The advantage of red-black ordering with regards to parallelisation is that the calculation as performed on each black point is independent of what happens to every other black point. Therefore in a massively parallel code, each processor can calculate the potential on some subset of the black points, before communicating once, and then working independently again on the red points.

The major disadvantage of relaxation techniques is that due to the fact that each gridpoint interacts only with its immediate neighbour every timestep that large scale perturbations take very many iterations to travel the length of the grid, and convergence is very slow. This problem will be addressed when discussing hybrid codes in section 2.2.1.

Comparison of Methods of Solving the Poisson Equation

So far in this section we have described four different methods of solving the Poisson equation for a system of discrete particles. Each of these methods has its own advantages and disadvantages. In this section we will compare each of the different methods and introduce some examples of hybrid codes, which combine the best features of two or more computational methods.

Particle-Particle Codes: PP codes have the advantage that they are conceptually the simplest codes to understand, and they obtain the gravitational force on any particle to within machine precision. The big problem with PP codes is that as the number of particles N increases, the number of calculations increases as $\sim O(N^2)$, making PP codes computationally prohibitive for large numbers of particles. Specialised hardwave, for example GRAPE (Sugimoto et al. (1990)) can be employed to allow the rapid evaluation of the gravitational sums, but PP codes are still limited in their application to large simulations.

Particle-Mesh Codes: PM codes solve exactly for the gravitational force on a series

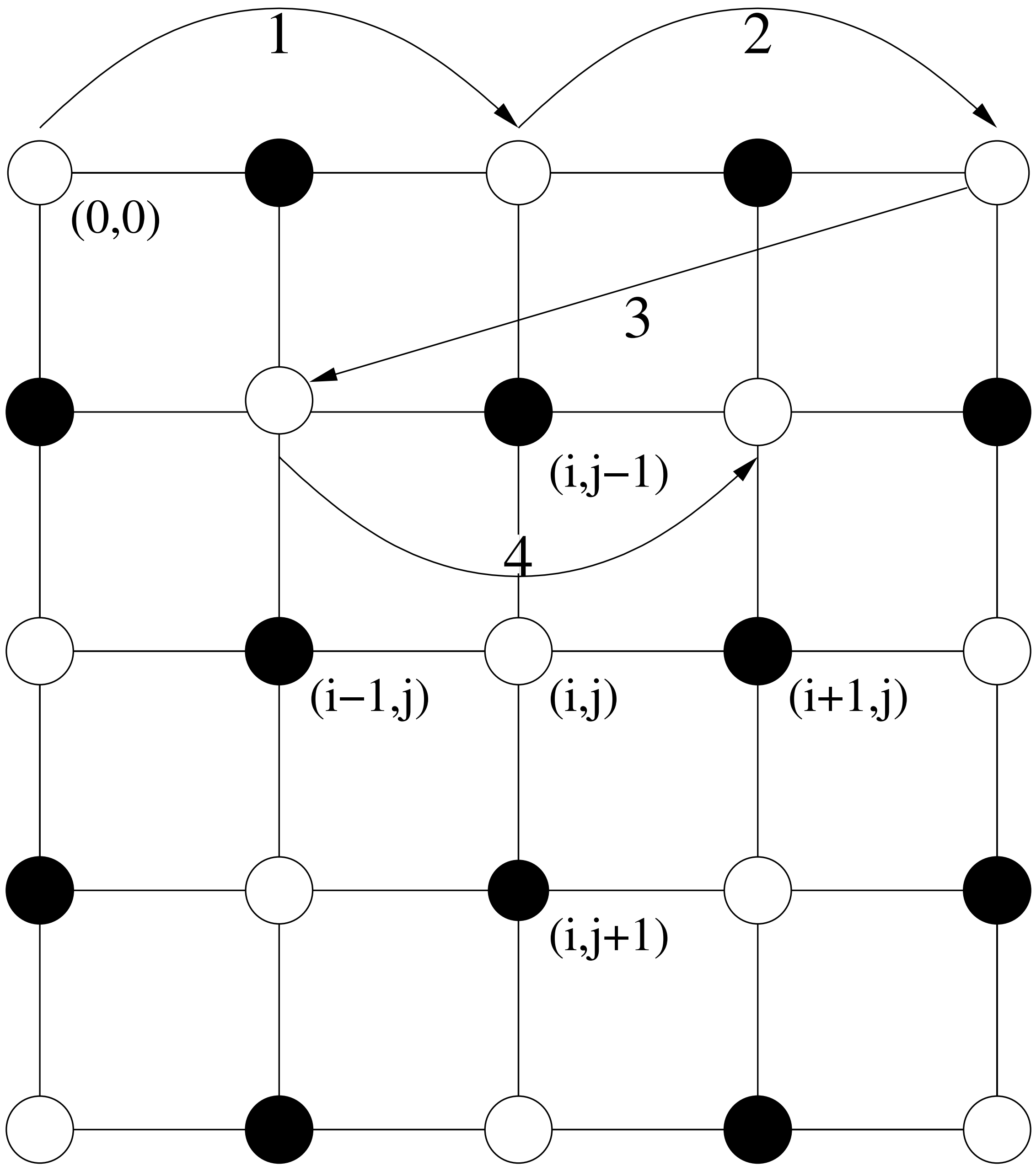


Figure 2.4: Red-Black ordering. Each circle represents a point on the computational grid, specified by two indices i and j. When iterating to solve the gravitational potential, we first loop over all white points (as labelled by arrows 1,2,3,4), updating the solution as we go. Then we do a second loop over all black points, using the updated values of the white points to obtain the new solution.

of gridpoints, the only errors introduced are due to the interpolation from particles to gridpoints and back. The major drawback of a PM code is that the gravitational force cannot be represented below the scale of a couple of grid spacings, and a hard limit is placed on the size of a PM grid by memory considerations. To achieve a factor of two in the grid spacing, the total number of gridpoints must be increased by a factor of 8. The number of calculations needed for a PM code scales as $O(N_g \log_2 N_g)$ (Hockney and Eastwood (1988)), where N_g represents the number of gridpoints in the simulation domain, making the PM calculation one of the most efficient methods of obtaining a gravitational potential.

Tree Codes: Tree codes solve for the gravitational potential at the point of each particle to within any desired accuracy. The number of calculations required to evaluate the gravitational potential in a system of N particles scales as $O(\mathrm{N}\log N)$ (Springel (2005)). Although not as efficient as a PM code, tree codes do not suffer from the resolution limit imposed by a finite grid size.

Mesh Relaxation Codes: Mesh relaxation techniques (e.g. Gauss-Seidel iteration) can solve the Poisson equation to within any required tolerance. On a uniform grid, pure Gauss-Seidel iteration is unworkable due to the large number of iterations required to communicate information over very many gridpoints. These techniques become more useful in combination with adaptive grids. In these codes the large scale perturbations are first solved on a coarse grid and then sampled up to a finer grid, with the Gauss-Seidel iterations used only to perform corrections on small scales.

Hybrid Techniques: Various authors have either extended these techniques or combined two or more of them in order to make use of the best features of each method. The large drawback to using a PM code is that the forces on small scales are strongly supressed. The simplest extension to a plain PM code is to explicitly sum the forces from nearby particles (the PP calculation), and use the mesh to calculate long range forces. This is a P^{3}M code (Hockney and Eastwood (1988); Efstathiou and Eastwood (1981)). Efstathiou et al. (1985) compared the relative accuracy of P^{3}M and PM codes in a cosmological context. P^{3}M codes can resolve the gravitational force down to any required scale but the number of operations needed to calculate the gravitational force scales as $O(N_n \log(N))$ (Springel (2005)), where N represents the number of particles in the simulation and N_n represents the average number of neighbours (defined as particles close enough to be calculated using the PP bit of code). If the particle resolution is significantly higher than the resolution of the PM grid then the PP calculation begins to dominate and the calculation slows down. A further extension to the P^{3}M code is to adaptively place high resolution PM grids over high density parts of the volume Adaptive-P^{3}M (Couchman et al. (1995)). This cuts down on the amount of work that the PP calculation has to do inside the refined regions. Care must be taken, however, as the memory requirements for the storage of many high resolution PM grids are large.

The numerical technique that is currently most commonly employed in cosmological simulation is the Tree-PM (Xu (1995)) method. In a similar way to the P^{3}M method, the short range part of the gravitational interaction is replaced with a tree code. This simulation technique is very computationally efficient, allows the gravitational force to be resolved to arbitrarily small scales and as such has become the technique of choice for most of the current generation of lagrangian codes.

2.2.2 The Hydrodynamic Calculation

In order to describe the behaviour of an ideal gas we need to define both the equations of motion and the equation of state. The equation of state of an ideal gas is given by

$$p = A\rho^{\gamma} \,, \tag{2.18}$$

where γ is the ratio of the specific heats. There are then three equations that describe the behaviour of the gas. The continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \,, \tag{2.19}$$

the Euler equation ,

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}^2) + \nabla p = \rho \mathbf{g} \,, \tag{2.20}$$

and the energy evolution equation:

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [(\rho E + p)\mathbf{v}] = \rho \mathbf{v} \cdot \mathbf{g} \,. \tag{2.21}$$

Here p represents the gas pressure, ρ represents its density and γ is the adiabatic exponent. A is assumed to be constant in both space and time and is related to the specific entropy of the gas. v and g represent the velocity and gravitational acceleration vectors and E is the internal energy per unit mass of the gas. These equations also need to be coupled to the Poisson equation as described in section 2.2.1.

In the remainder of this section we will discuss how these equations can be modelled both by a system of particles and by a regular mesh.

Particle Methods

The most popular Lagrangian (mass quantised) hydrodynamics method is Smoothed Particle Hydrodynamics (SPH; (Gingold and Monaghan, 1977; Monaghan, 1992)) in which the fluid is represented by some number of discrete point particles, and the fluid properties at any point can be calculated by taking the weighted mean of the properties of nearby particles.

The main advantages of SPH are that it does not need any sort of spatial grid in order to calculate spatial derivatives and that by its lagrangian nature areas of interest (usually high density regions) are well represented due to the flow of particles into that area. This is in contrast to grid based hydrodynamic methods where great effort must be spent in setting up grids so that areas of interest are simulated in high detail (see section 2.2.2).

The essentials of the SPH method can be expressed with two concepts. Firstly we need to assume that the phase space distribution of a gas can be adequately represented by some discrete distribution of particles. Secondly the properties of the gas fluid at any point can be estimated by taking a weighted average of those properties over the surrounding particles.

The value of some hydrodynamical quantity, A, can be calculated at any point in space via the following equation

$$\widetilde{A}(\mathbf{r}) = \int A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}', h)\mathbf{dr}', \tag{2.22}$$

where the integration is over all space. A represents some property of the fluid, with $\widetilde{A}$ representing an estimated value at coordinate $\mathbf{r}$. $W(r, h)$ is an appropriately chosen smoothing function, the SPH kernel, which must obey two properties. Firstly it must be normalised,

$$\int W(\mathbf{r}-\mathbf{r}', h)\mathbf{dr}' = 1\,, \tag{2.23}$$

secondly, in the limit that we have an infinite number of particles (and therefore h tends to zero), the kernel must tend toward becoming a delta function

$$\lim_{h\to 0} W(\mathbf{r}-\mathbf{r}', h) = \delta(\mathbf{r}-\mathbf{r}')\,. \tag{2.24}$$

The original kernel chosen by ((Gingold and Monaghan, 1977)) was a Gaussian,

$$W(x, h) = \frac{1}{h\sqrt{\pi}} e^{-(x^2/h^2)}\,, \tag{2.25}$$

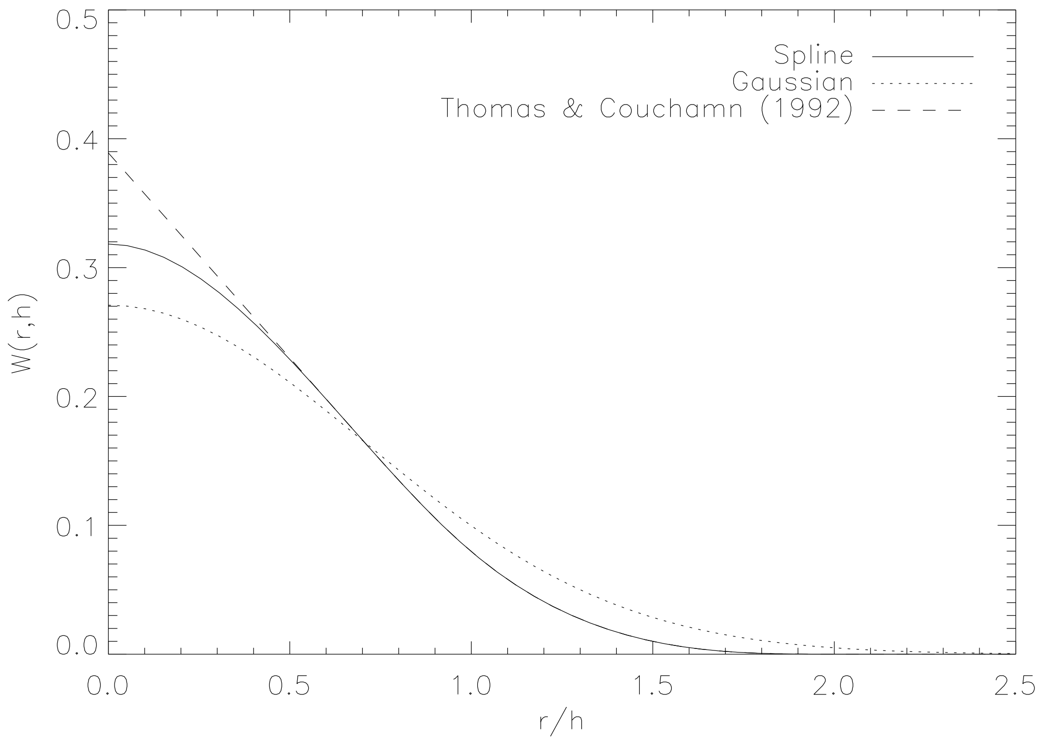


Figure 2.5: A comparison of SPH kernels as used in the literature. The spline kernel falls off to precisely zero ar $r/h = 2.0$, the Gaussian kernel is small but nonzero at high radii. The Thomas & Couchman (1992) kernel is identical to the spline kernel at $r/h > 0.5$, but is modified such that it is linear at smaller radii

an improvement to the Gaussian kernel comes in the form of a spline kernel

$$W(\mathbf{r}, h) = \frac{1}{\pi h^3} \begin{cases} 1 - \frac{3}{2}\frac{r}{h}^2 + \frac{3}{4}\frac{r}{h}^3 & \text{if } 0 \leq \frac{r}{h} < 1; \\ \frac{1}{4}(2 - \frac{r}{h})^3 & \text{if } 1 \leq \frac{r}{h} \leq 2; \\ 0 & \text{otherwise}. \end{cases}$$

The spline kernel has a continuous second derivative, the dominant error term in the integral interpolant falls off as h^2, and for $r > 2h$ the kernel is precisely zero. Figure 2.5 shows three different SPH kernels used in the literature.

When working with a finite number of particles the integral in equation 2.22 is approximated by a summation

$$\widetilde{A}(\mathbf{r}) = \sum_i m_i \frac{A_i}{\rho_i} W(\mathbf{r} - \mathbf{r_i}, h). \tag{2.26}$$

Where i represents a unique particle label for each particle in the simulation. It is important to note that although the summation in equation 2.26 is formally done over all particles, in reality the choice of $W(r,h)$ will mean that only those particles within a distance $2h$ of the point we are considering are important in the calculation. Derivatives of the interpolant can be obtained by ordinary differentiation; no need for finite differencing or grids. For example ∇A may be written

$$\nabla\widetilde{A}(\mathbf{r}) = \sum_i \frac{A_i}{\rho_i}\nabla W(\mathbf{r}-\mathbf{r_i},h)\,. \tag{2.27}$$

Using these ideas the equations of motion of an ideal gas may be obtained. Consider an ideal gas with no heat sinks or inputs, decoupled from the gravitational force. The momentum (or Euler) equation is then described by

$$f = -\frac{1}{\rho}\nabla p + \nabla\phi_G = \frac{\partial v}{\partial t} + (v\cdot\nabla)v\,. \tag{2.28}$$

For the purposes of demonstrating how SPH represents the equations of motion of an ideal gas we will consider rewriting the $\nabla P/\rho$ term in a form suitable for integration with an SPH code. The simplest approach is to use the fundamental equation of SPH (equation 2.26) along with equation 2.27 and the identity

$$\nabla P = [\nabla(\rho P) - P\nabla\rho]/\rho\,, \tag{2.29}$$

to write

$$\rho_a\nabla P_a = \sum_i m_i(P_a - P_i)\nabla_i W(r_i - r_a, h)\,, \tag{2.30}$$

where $\nabla_i W(r_i - r_a, h)$ represents the gradient of $W(r_i - r_a)$ taken with respect to the coordinates of particle a. However, with this equation of motion, linear and angular momentum are not conserved exactly (Monaghan (1992)). It is therefore better to make the equation symmetric between a and i by using the identity

$$\frac{\nabla P}{\rho} = \nabla(\frac{P}{\rho}) + \frac{P}{\rho^2}\nabla(\rho)\,, \tag{2.31}$$

we can write that

$$\frac{\nabla_a P_a}{\rho_a} \approx \sum_i \frac{m_i}{\rho_i^2}P_i\nabla_i W(r_i - r_a,\bar{h}) + \frac{P_a}{\rho_a^2}\sum_i m_i\nabla_i W(r_i - r_a,\bar{h})\,;$$

$$\frac{\nabla_a P_a}{\rho_a} = \sum_i m_i\Big(\frac{P_a}{\rho_a^2} + \frac{P_i}{\rho_i^2}\Big)\nabla_i W(r_a - r_i,\bar{h})\,, \tag{2.32}$$

where $W(r_a - r_i,\bar{h})$ is a symmetrized kernel (Hernquist and Katz (1989)), which ensures that the force between two particles is precisely symmetric.

$$W(r_a - r_i,\bar{h}) = \frac{1}{2}|W(r_i - r_j,h_i) + W(r_i - r_j,h_j)|\,. \tag{2.33}$$

Equations 2.32 and 2.33 provide the basis for reconstructing the equations of motion of an ideal gas in a form suitable for use in SPH. Using these techniques the momentum of particle i can be integrated forward in time using (Springel and Hernquist (2002))

$$\frac{dv_i}{dt} = -\sum_a m_a \Big(\frac{P_a}{\rho_a^2} + \frac{P_i}{\rho_i^2} + \Pi_{ia}\Big) \nabla_a W(r_a - r_i, \bar{h}) \,. \tag{2.34}$$

Where Π_{ia} represents an artificial viscosity force between particles i and a. The artificial viscosity is introduced in order to damp the unphysical oscillations that occur after strong shocks, and also has the effect of allowing shocks to occur. The artificial viscosity term was originally introduced by Monaghan and Gingold (1984) and took the following form, known as the Monaghan-Gingold tensor

$$\Pi_{ij} = \begin{cases} -\frac{\alpha c_{ij}\mu_{ij} + \beta\mu_{ij}^2}{\rho_{ij}}, & \text{if } (r_i - r_j)\cdot(v_i - v_j) < 0; \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{ij} = \frac{h_{ij}(r_i - r_j)\cdot(v_i - v_j)}{r_{ij}^2 + \eta^2}, \tag{2.35}$$

where c_{ij}, h_{ij} and ρ_{ij} are the arithmetic means of the sound speed, smoothing length and density of particles i and j. α and β are free parameters that are used to tune the strength of the artificial viscosity term. Steinmetz (1996) found that values of $\alpha = 0.5$ and $\beta = 1$ worked well. For problems with strong shocks, these values were increased to $\alpha = 1$ and $\beta = 2$, but it is noted that an increased artificial viscosity has the effect of smearing out shocks.

It was noted (Hernquist and Katz (1989); Katz and Gunn (1991)) that this form of the artificial viscosity does not vanish in pure shear flows ($\nabla \times v \neq 0$, $\nabla \cdot v = 0$), and may lead to unphysical angular momentum transport in the formation of disks, particularly in the case of low resolution galaxy simulations (Steinmetz (1996)). This problem can be almost completely prevented by using the form of the articifial viscosity due to Balsara (1995)

$$\bar{\Pi}_{ia} = \Pi_{ia}(f_i + f_a) \,, \tag{2.36}$$

where the function f is an order-of-magnitude estimate of the irrotational fraction of the flow. Described by

$$f_i = \frac{|\nabla \cdot v_i|}{|\nabla \cdot v_i| + |\nabla \times v_i| + 0.0001 c_i/h_i} \tag{2.37}$$

The final term in the denominator prevents divergences. In the case of a purely compressive flow ($\nabla \times v = 0$, $\nabla \cdot v \neq 0$) $f = 1$, and the artificial viscosity is identical to the Monaghan-Gingold viscosity. In a purely rotational flow ($\nabla \times v = 0$, $\nabla \cdot v \neq 0$) the viscosity is completely supressed. Steinmetz (1996) found that in a simple simulation of a gas disk containing 280 particles and using the Monaghan-Gingold artificial viscosity the half-angular-momentum velocity grew by a factor of 2 to 3 within 3Gyr. Using the modified artificial viscosity the half-angular-momentum radius varied by less than 10% over a Hubble time.

Grid Methods

The basis of most grid based hydrodynamics codes is to split the simulation volume into a discrete set of gridpoints, for which the gas properties are known. The evolution of this system can then be evolved by solving for the behaviour of a gas at the boundaries between each of these cells. This problem may be described in the following way,

$$\mathbf{u}(x,0) = \begin{cases} u_{\mathrm{L}}^0 & \text{for } x < 0\,, \\ u_{\mathrm{R}}^0 & \text{for } x \geq 0\,, \end{cases}$$

where subscripts denote spatial positions and superscripts represent time. Such a problem is called a *Riemann problem* and represents a discontinuity in the gas properties. The usual way of incorporating the Riemann problem into the numerical solution is to take (u_i^n, u_{i+1}^n), for each i in turn, that is to treat the boundary between each grid cell in the simulation as a separate Riemann problem, which are then thought of as providing information about the solution in each interval $(i, i+1)$. The precise workings of the Riemann solver are beyond the scope of this thesis but include random choice methods (Glimm (1965); Chorin (1976)) and Newton iteration (Colella and Glaz (1985); van Leer (1979)).

Godunov (1959) assumed that the initial data could be replaced by a piecewise constant set of states with discontinuities at $x_{i-1/2}$ and $x_{i+1/2}$, this simplified problem can then be solved exactly. Godunov then replaced the exact solution with a new set of piecewise constant approximation, preserving the integral properties of the exact solution. The first major extension to this approach was made by van Leer (van Leer (1979)), who approximated the data by piecewise linear segments, allowing discontinuities between the segments. This approach required the solution to a more complex problem but raises the accuracy of the resulting solution substantially.

The piecewise parabolic mesh (PPM) method (Colella and Woodward (1984); Woodward and Colella (1984)) extends this concept to higher orders and uses parabolic solutions to the Riemann problem, allowing for much more accurate solutions. PPM codes are now used in a wide variety of current mesh codes (e.g. Fryxell et al. (2000); Bryan et al. (1995)) due to the high order accuracy of the scheme and the fact that PPM can resolve strong shocks much more readily than the lower order piecewise codes.

Uniform mesh codes run into the same problems as pure PM gravity codes, the spatial resolution of the resulting solution is limited by the size of the largest uniform mesh that can fit in memory. For this reason adaptive mesh refinement (AMR) codes have been developed (e.g. Fryxell et al. (2000)). Here the spatial resolution of the grid is allowed to vary with position. This is demonstrated in figure 2.6, which depicts the density field around a simulated galaxy cluster (see section 2.3.1 for details of the simulation).

In most AMR codes, the actual structure of the adaptive mesh takes one of two forms: *block structured*, in which the computational volume is filled with identical 'blocks' that may be split apart or combined together to change the spatial resolution at a point., or *structured grids* in which rectangular grids of different sizes and resolutions are placed in regions of interest. Figure 2.6 shows an example of a block structured code (FLASH). Each square in the image represents $8 \times 8 \times 8$ grid points. It is clear that the regions of interest (the centre of the galaxy cluster) are simulated with greater precision than the surrounding gas. One challenge in writing a block structured AMR code is to find some measure of when it is necessary to refine a block up to a higher refinement level. FLASH uses the Löhner (1987) estimator. In one dimension if we have some discrete function u, defined at gridpoints seperated by an amount Δ then we can approximate the second derivative as:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} + u_{i-1} - 2u_i}{\Delta^2} \,. \tag{2.38}$$

The Löhner estimator, which is a modified version of the second derivative, normalised by the average of the gradient over one computational cell, is then given by

$$L = \frac{|u_{i+1} - 2u_i + u_{i-1}|}{|u_{i+1} - u_i| + |u_i - u_{i-1}| \epsilon \Big[|u_{i+1} - 2|u_i| + |u_{i-1}| \Big]} \,, \tag{2.39}$$

where u_i is the refinement test variable's value in the ith cell. The last term in the denominator of this expression acts as a filter, preventing the refinement of small ripples. By default the constant ϵ is given a value of 0.01, although this can be tuned in the code. In addition to this refinement criterion it was found that forcing refinement in regions of high density provided very good results.

At the expense of more complex coding, AMR codes allow Eulerian hydrodynamics codes to compete with Lagrangian codes, both in terms of spatial resolution and the computational resources required to evolve a system (see e.g. O'Shea et al. (2005) for a recent comparison)

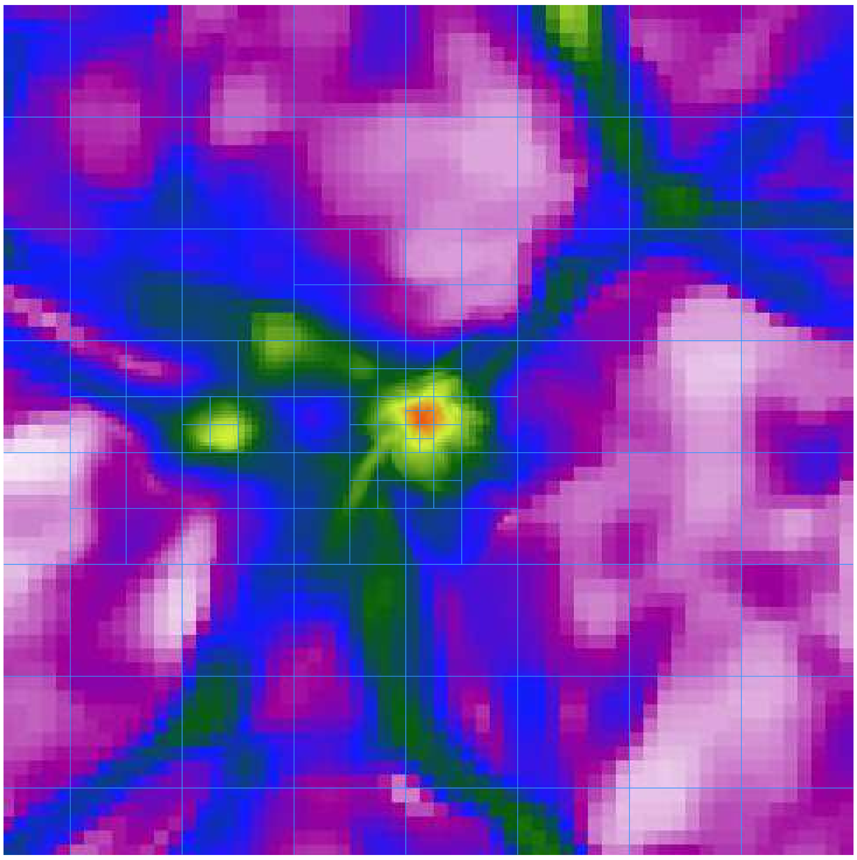


Figure 2.6: Baryonic density slice around a simulated galaxy cluster at redshift 0. The image is 20Mpc across. Blue lines represent the location of blocks in the adaptive AMR mesh, each square contains 8×8 gridpoints. It is clear that the mesh has a higher spatial resolution in the higher density regions so the properties of the cluster are simulated more exactly than those of the surrounding low density gas. The blocks in this image were refined using the local gas density to flag when it becomes necessary to increase the level of refinement.

2.2.3 Time Stepping

The final numerical detail we will consider is the role of timestepping in cosmological simulation.

The naive choice for a system of timestepping is just to update the velocity and position at the start of each timestep and then make linear increments over the length of each timestep. This may be denoted by

$$r_{i+1} = r_i + v_i \Delta t \,, \tag{2.40}$$

$$v_{i+1} = v_i + a_i \Delta t \,, \tag{2.41}$$

where r, v and a are positions, velocities and accelerations respectively. The subscript i represents the value at the beginning of timestep i, and $i+1$ represents its value at the end. This timestepping method is only first-order accurate. A simple improvement on this simple timestepping scheme is the so called *leapfrog* scheme (see e.g. Hockney and Eastwood (1988)), which is second-order accurate, but still only requires one costly force evaluation per timestep

$$r_{i+\frac{1}{2}} = r_i + \frac{1}{2} v_i \Delta t \,, \tag{2.42}$$

$$v_{i+1} = v_i + a_i t \,, \tag{2.43}$$

$$r_{i+1} = r_{i+\frac{1}{2}} + \frac{1}{2} v_{n+1} \Delta t \,, \tag{2.44}$$

More complex timestepping algorithms are discussed by Quinn et al. (1997). Other than the time integration method we need to be careful with the length of timesteps. If timesteps are too short then the simulation will take an unfeasable amount of time to run. If they are too long then the system being simulated may become unstable

For baryonic matter the fundamental timestep limiting factor is given by the Courant-Friedrich-Lewy (CFL; Courant et al. (1928)) condition, which qualititatively states that the timestep must be less than the time for 'some significant action' to occur, and preferably considerably less. For baryonic matter this is usually set using the time it takes for a 'signal', say, a sound-wave to propagate across one resolution element, or

$$\Delta t_{\mathrm{cfl}} = \frac{H \; \mathrm{cfl}}{c} = \frac{H\rho \; \mathrm{cfl}}{\gamma p} \,, \tag{2.45}$$

where cfl is a parameter, set significantly lower than unity in order to guarantee stability, c is the local sound speed. H is the size of a single resolution element corresponding to either the smoothing length of a particle or the size of a grid cell depending on the type of simulation code in use. Other timesteps used in simulations include limits controlled by the radiative cooling rate of a gas and the maximum particle displacements per timestep.

2.3 Code Validation

In sections 2.2.1 and 2.2.2 we discussed a large number of different simulation algorithms, all of which make different assumptions and calculate forces in widely differing ways. It is reasonable, therefore, to ask how similar the behaviour of the different codes is in a physically relevant situation. In this section we investigate the accuracy of results produced by two different simulation codes (FLASH Fryxell et al. (2000) and Gadget Springel (2005)), which operate using two very different algorithms (PPM on an adaptive grid & Mesh relaxation in FLASH versus SPH and Tree-PM in Gadget).

The full physics of galaxy formation and evolution is a very complex problem (see chapter 3), and is not yet fully understood. For this reason we restrict our studies to physics that is relatively well understood: gravitational dynamics and adiabatic[1] gas dynamics. Although vastly simplified, this model has immediate relevance to the simulation of various astronomical objects including the hot component of X-Ray clusters.

Inclusion of more complex physical processes including cooling and heating, metal production, supernova feedback and reionization is still something that differs wildly between different authors and a comprehensive comparison between different author's implementations of additional physics still remains to be done.

Various code comparison projects have been undertaken in the literature. O'Shea et al. (2005) compare two codes, simulating dark matter and adiabatic gas in a large cosmological volume. The most comprehensive comparison project undertaken so far is probably the Santa Barbara (SB) comparison project of Frenk et al. (1999). The initial conditions for the SB cluster are publically available[2], making the SB cluster an ideal way to verify the accuracy of our simulation codes.

2.3.1 The Santa Barbara Test

The SB cluster (Frenk et al. (1999)) represents the formation of an X-Ray cluster in a CDM universe, and was originally simulated independently by 12 different groups. The codes compared in the paper span the full range of techniques discussed in this chapter and include both parallel and serial simulation codes, and have resolution lengths from 5kpc all the way up to 960kpc.

[1]Although technically these simulations do contain non-adiabatic shocks and should perhaps better be termed 'non-radiative'simulations, we continue to follow the convention found in the Santa Barbara paper and refer to them as adiabatic simulations

[2]http://star-www.dur.ac.uk/~csf/clusdata

The cosmology of choice for the SB study used the parameters $h_{100} = 0.5$, $\sigma_8 = 0.9$, $\Omega = 1.0$, $\Omega_b = 0.1$. The size of the simulated volume at redshift 0 was 64Mpc. The power spectrum for the simulation was obtained from Bardeen et al. (1986). The initial conditions were made available in the form of 256^3 grids of linear theory displacements in the x, y, and z directions (see 2.4), and each author in the original SB paper used these displacement files to generate initial conditions in their own format. The recommended starting redshift for each simulation is 20.

Simulation Details

We ran the SB cluster initial conditions with both Gadget and FLASH. The initial conditions were subsampled down to a 64^3 mesh, then one particle was placed in the centre of each gridcell. The minimum allowed adaptive gridcell size allowed in FLASH is 0.125Mpc, the gravitational softening allowed in Gadget is 0.1Mpc. The minimum SPH smoothing length is constrained to be a minimum of 0.1 times the gravitational softening.

At redshift zero the centre of mass coordinates of the haloes agreed very well, [33.308, 34.45, 33.53] in FLASH versus [33.418, 34.36, 33.26] in Gadget. The dark matter properties at redshift zero lie in all cases within the range of scatter investigated in the SB paper. The dark matter density profiles are shown in figure 2.7. Outside of the resolution length, each density profile agrees very well. Inside of one resolution length the results become very uncertain. The same behaviour was found with the dark matter velocity dispersion profiles (figure 2.8). Plots of the positions of individual dark matter particles in each halo show that although the overall position of the halo remains unchanged between simulations, the locations of the very non-linear small scale structure is significantly different between the two runs. Once again this is in good agreement with the differences between different codes found in the original SB paper.

In common with the SB paper we find that when looking at the baryonic component of the matter in the halo agreement between the different codes is less good. The gas density profile in the halo (figure 2.10) is in good agreement down to roughly the resolution length of each code. The AMR codes, which specify gas properties on a grid have a constant density below the grid spacing.

The temperature profile of the galaxy clusters are plotted in figure 2.10. Additionally plotted in this graph are the results from the code of Bryan and Norman (1995), an AMR code that solves the hydrodynamic equations in a similar way to FLASH. Once again up

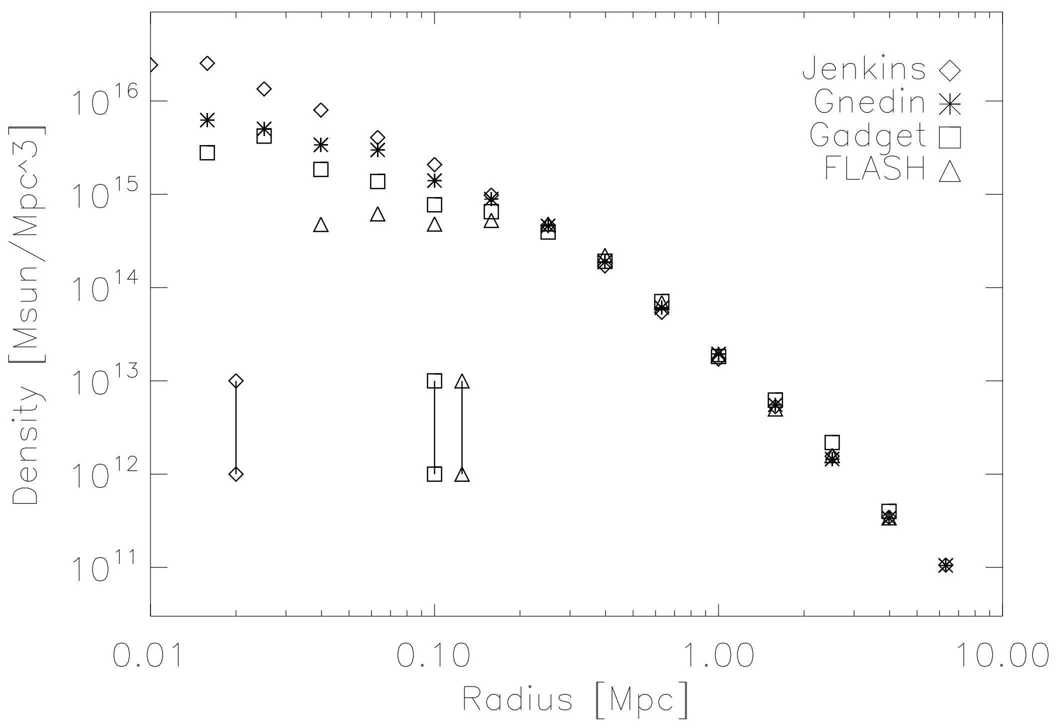


Figure 2.7: Dark matter density profile of the z=0 halo from both the FLASH and Gadget simulations alongside the results of Jenkins and Gnedin from the Santa Barbara paper. The three vertical lines represent the resolution limit in each of the simulations. The FLASH resolution limit is set to the size of the smallest cells in the cluster. Flash obtains a constant density on scales below the grid spacing. The Gadget softening was chosen to match closely with the FLASH grid spacing. The softening of Jenkins is 20kpc, the softening of the Gnedin simulation is equal to the softening of the Gadget simulation.

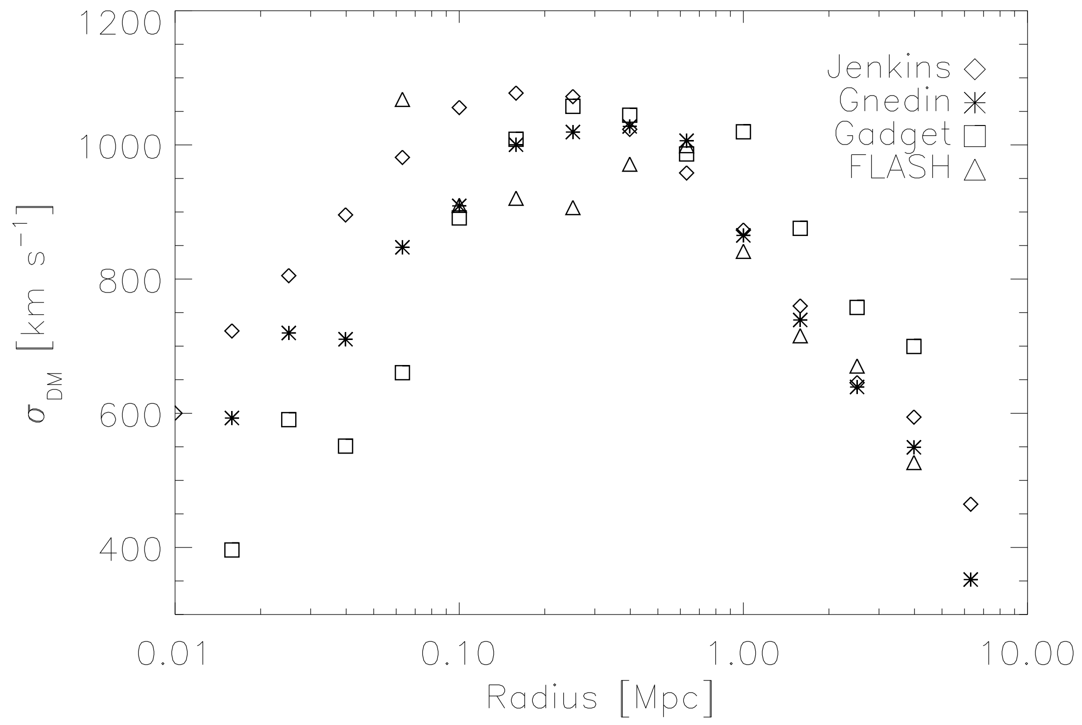


Figure 2.8: Dark matter velocity dispersion profile of the z=0 halo from both the FLASH and Gadget simulations alongside the results of Jenkins and Gnedin from the Santa Barbara paper. The quantity shown in the plot is the one-dimensional velocity dispersion, calculated from the three dimensional velocity dispersion by $\sigma_{1d} = \sigma_{3d}/\sqrt{3}$. In common with Frenk et. al. (1999) we find that agreement between all codes is to within 20% outside of the resolution limit of the code.

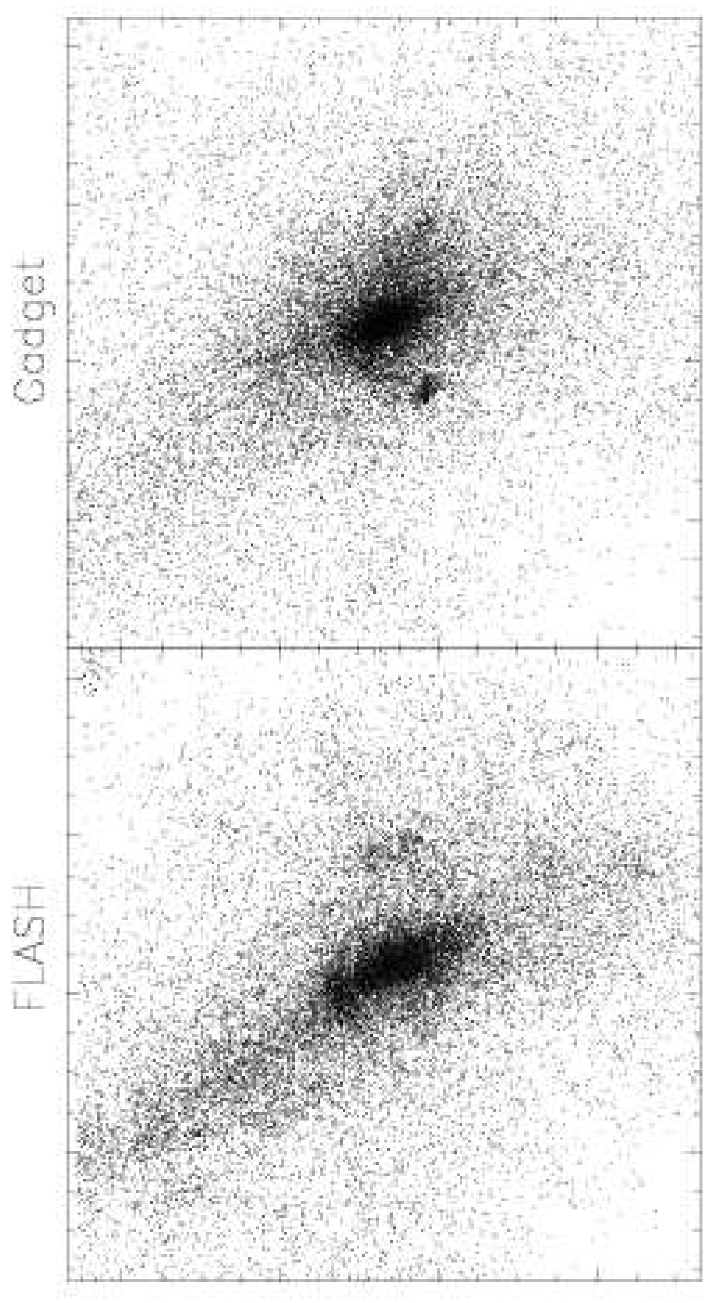


Figure 2.9: Dark matter particles in an 8Mpc cube centered on the centre of mass of each halo. The shapes of the haloes between the FLASH and Gadget runs differ by a small amount, but are both well within the spread of shapes observed in the original Santa Barbara study.

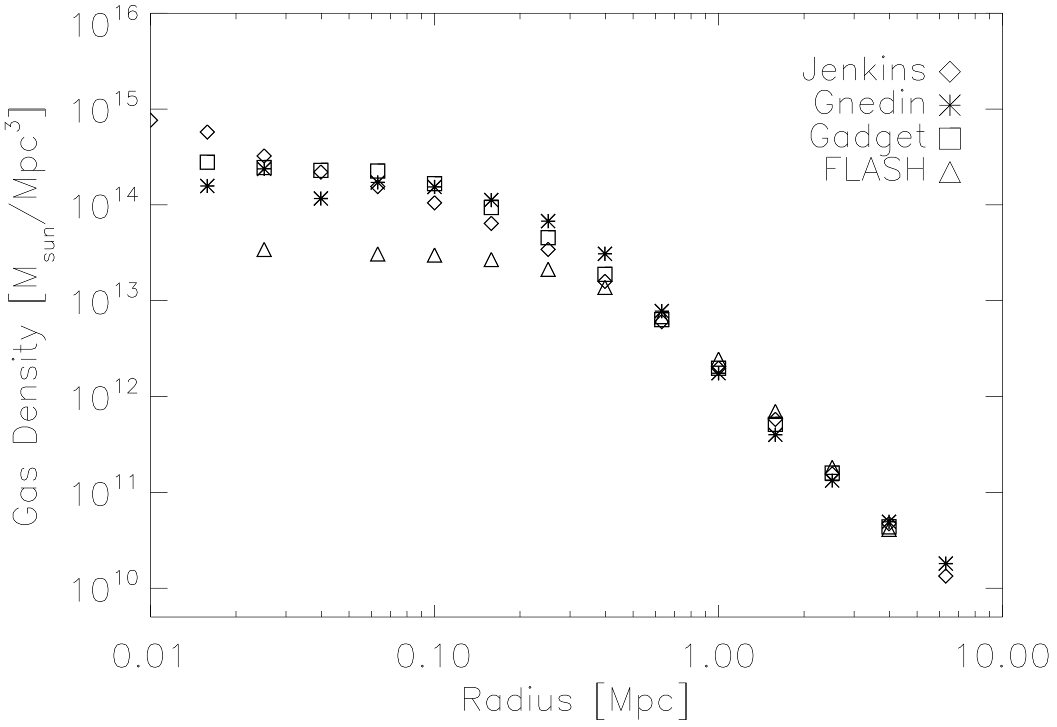


Figure 2.10: Gas density profile of the SB cluster from various runs. Agreement outside of the resolution length is good between all codes. Inside of one resolution length the density of the AMR run flattens off due to the finite cellsize.

to the resolution length of the simulation the results are very similar.

These results suggest that both codes can reliably simulate a complex, non-linear situation and despite the wildly different numerical effects and methods applied in each code the overall results are remarkably similar. This suggests that both codes are solving both the gravitational evolution and hydrodynamics accurately in this fully cosmological scenario.

2.4 Initial Conditions

The generation of correct and accurate initial conditions for cosmological simulation is of crucial importance. The problem is one of representing a nearly scale-invariant Gaussian random field with a given power spectrum using a discrete set of particles. In this section we introduce methods of generating the initial, uniform distributions of particles, before discussing methods of imposing a given power spectrum on this particle distribution,

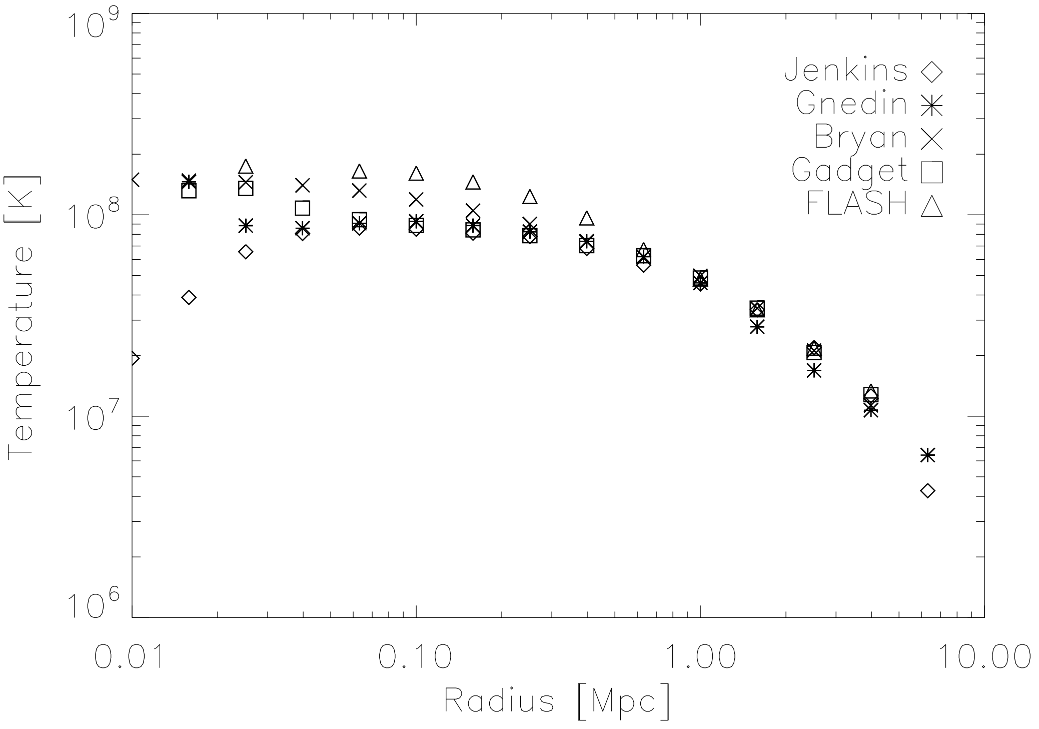


Figure 2.11: Temperature profile of the SB cluster. Agreement at the outskirts of the cluster is good. Additionally plotted in this graph are the results from the code of Bryan & Norman (1995), an adaptive mesh refinement code similar to FLASH. It appears that the temperatures near the centre of the halo diverge depending upon the simulation method used. It should be noted, however that the cellsize in the FLASH simulation is 0.125Mpc so we cannot trust results below this radius.

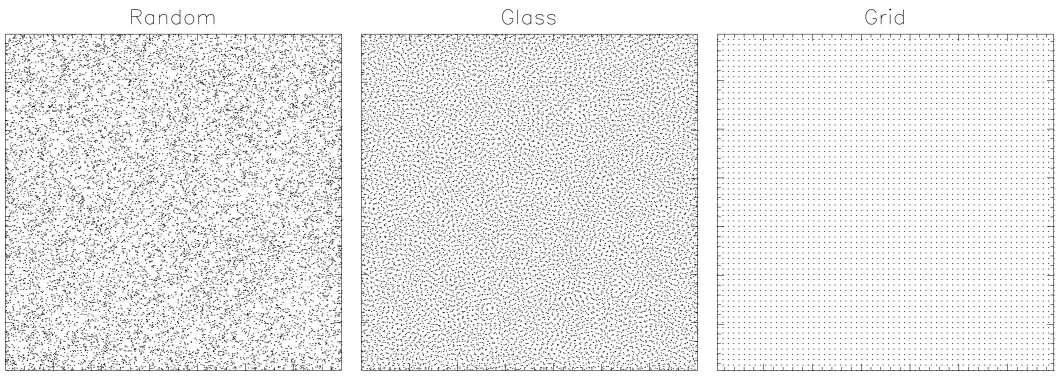


Figure 2.12: Different types of pre-initial particle distribution. Each panel shows particles from a thin slice from the pre-initial conditions of a 262144 particle simulation. The random distribution is unsuitable for any sort of simulation, both the grid and glass distributions are used by various authors.

and then discussing techniques for resimulation of individual objects. A code developed using the techniques in this section is used in section 2.5 to quantify the numerical effects of mass resolution on simulated galaxy haloes.

2.4.1 Pre-initial Conditions

Before generating cosmological initial conditions it is important that we are able to create a set of 'pre-initial conditions'; which represent the state of the system before the cosmological perturbations are imposed. This has been investigated in detail by White (1994), who discussed three choices for initial particle positions: randomly placed particles, particles on a grid and particles in a *glass* distribution. An example of each of these three configurations is shown in figure 2.12

The first type of particle distribution, random placing of particles suffers strongly from Poisson noise and initially has a white noise power spectrum. If a simulation filled with randomly distributed particles is evolved, non-linear objects will form, even if other fluctuations are not imposed.

The usual approach to get around this is to place particles on a grid, the problem with this is that the initial grid structure introduces strongly preferred directions along each axis on the scale of a few grid spacings. Simulations with very little small scale power (e.g. hot dark matter simulations) will therefore develop spurious features along the grid axis.

The optimal solution, as first proposed by White (1994) is to take an initially random distribution of particles and evolve it as an Einstein-de Sitter universe for many expansion factors, with the sign of the gravitational force reversed, making it repulsive. This has the effect of smoothing out the particle distribution and removing all large scale perturbations and preferred directions.

2.4.2 Generation of Uniform Volume

As mentioned at the beginning of section 2.4 the problem we wish to solve is that of representing a primordial spectrum of density fluctuations using a discrete ensemble of particles. As discussed in section 1.1.2 in the inflationary scenario these density fluctuations are thought to take the form of a Gaussian random field. This is defined as a field where in real space each data element is chosen from a Gaussian probability distribution. Equivalently we can represent this in k-space by choosing for each k-value normally distributed amplitudes (with variance equal to $P(k)$) and random phases.

The properties of a Gaussian random field may be described completely by its power spectrum, $P(k)$, which in the case of most inflation models is of the form

$$P(k) = Ak^{n_s} \, , \tag{2.46}$$

where A and n_s are the amplitude and slope of the primordial power spectrum. Cosmological initial conditions are generated at significantly lower redshift than the epoch of inflation and so we have to take into account their early linear evolution. This is usually accomplished by multiplying the initial fluctuation spectrum by a transfer function representing the later evolution of the fluctuations

$$P(k) = Ak^{n_s} |T(k,t)|^2 \, , \tag{2.47}$$

Transfer functions have been published by many authors, Holtzman (1989) provides a listing of early examples and describes how such transfer functions are calculated. These analytic examples have been superseded by numerical codes that compute matter transfer functions accurately (e.g. Ma and Bertschinger (1995); Seljak and Zaldarriaga (1996)). For the rest of this chapter we will assume that reliable transfer functions have been provided. We now introduce the methods by which initial conditions can be generated by two different methods; working in k-space and real-space.

In k-space

Our starting point is to describe the density at a given space and time, using the density contrast, defined as

$$\delta(x,t) = \frac{\rho(x,t) - \bar{\rho}(t)}{\bar{\rho}(t)}\,, \tag{2.48}$$

where $\bar{\rho}(t)$ is the mean density of the universe at time t. If we assume that the simulation volume we are interested in is periodic then we may obtain the k-space representation of the density field via a Fourier transform,

$$\delta(k) = \frac{1}{(2\pi)^3}\int e^{-i\mathbf{k}.\mathbf{x}}\delta(x)d^3x\,. \tag{2.49}$$

In k-space we may measure the power spectrum, $P(k)$, of the Gaussian random field by noting that

$$P(k) = \delta(k)\delta^*(k)\,. \tag{2.50}$$

Where $*$ represents the complex conjugate. In order to generate a random field with a desired power spectrum we initially specify the k-space density fluctuation field, $\delta(k)$. The k-space fluctuation is a complex value with random phase and a gaussian distribution of amplitudes. We pick a random amplitude for each k-vector from a distribution with variance $P(k)$. These random numbers are the k-space representation of the density field, the real space initial density field may be then recovered by an inverse Fourier transform

$$\delta(x) = \int e^{i\mathbf{k}.\mathbf{x}}\delta(k)d^3k\,. \tag{2.51}$$

This provides a conceptually simple way of generating random fields with any desired power spectrum, but it does have some drawbacks, notably that only a discrete number of wavevectors can fit a whole number of wavelengths across the simulation volume. These wavevectors are given by

$$k = \frac{2n\pi}{L}\,, \tag{2.52}$$

where L is the size of the simulation volume. This suggests that the power spectrum is only sampled at a series of discrete values, and so any features in the power spectrum that lie between these points are not modelled. This effect is particularly serious at small values of k, where only very few wavelengths fit across the box. Secondly the finite size of the simulation volume means that wavelengths larger than that of the simulation box are not included in the simulations. Gelb and Bertschinger (1994) showed that in a CDM cosmology a simulation volume of size 50Mpc would underestimate σ_8 by over

40% due to the truncation of the power spectrum at long wavelengths. Simulations of the Lymanα forest are particularly strongly affected by the loss of this large scale power (see Rauch et al. (1997)). We shall see in later sections how some of these difficulties may be overcome.

Given a pre-initial distribution of particles and an associated density field the final step in calculating the initial particle distribution is to perturb the initial particle positions to give them the the same power spectrum as the density field generated above. This procedure is calculated using the Zel'Dovich approximation (Zel'Dovich (1970))

$$\mathbf{r}(t, \mathbf{q}) = a(t)[\mathbf{q} - b(t)s(\mathbf{q})] , \tag{2.53}$$

where $\mathbf{q}$ represents the unperturbed position of each particle, $\mathbf{r}(t, q)$ is the perturbed position of a particle initially at position $\mathbf{q}$, $b(t)$ is the growth rate of linear density fluctuations in the expanding universe and $s(q)$ may be written as the gradient of a potential, $s(q) = \nabla\Phi(q)$ (Shandarin and Zeldovich (1989)), given suitable normalisation.

The form of the growth rate for density perturbations depends upon the cosmology we are using, and are calculated by rewriting the continuity, Euler and Poisson equations in terms of the density contrast (Peebles (1981)), and then assuming that in the early universe both δ and $\dot{\delta}$ are small. In the case of an Einstein-de Sitter universe ($\Lambda = 0$, $\Omega_0 = 1$, dust-dominated) then two solutions are permitted; a growing mode with $\delta \propto t^{2/3} \propto a$ and a decaying mode $\delta \propto 1/t \propto a^{-3/2}$. In the case of universes with lower matter densities or a cosmological constant the solution is more complex but some situations still lend themselves to analytic solution (see e.g. White (1994))

Once positions have been assigned, it is necessary to generate initial velocities for the particles. At times corresponding to the start of the simulation the peculiar velocity at a point can be derived immediately from equation 2.53

$$\dot{r}(t, q) = -\dot{b}(t)s(q) . \tag{2.54}$$

The full procedure for generating a uniform set of initial conditions is therefore to

- Generate a random field in k-space with variance $P(k)$ at each k vector
- take the inverse fourier transform of this field, to obtain a uniform mesh of densities with the required power spectrum
- generate a 'pre-initial' distribution of particles

- displace the particles using the Zel'Dovich (1970) approximation, set velocities using equation 2.54

In the following sections we will examine how initial conditions may be generated by a different method and then how non-uniform initial conditions may be created.

In Real Space

It was pointed out by Salmon (1996) that any Gaussian random field sampled on a lattice can be written as the convolution of white noise with a function that we will call the transfer function, or

$$\delta(\mathbf{x}) = \int T'(k) N(k) e^{i\mathbf{k}.\mathbf{x}} d^3k \,, \tag{2.55}$$

Where $T'(k)$ represents the transfer function. Note that the transfer function defined above is not the same as that defined in section 2.4.2. $N(k)$ is Gaussian white noise with autocorrelation

$$\xi(r) = \langle N(x) N(x+r) \rangle = \delta_D(0) \,, \tag{2.56}$$

where δ_D represents a Dirac delta function. The transfer function may be shown to be related to the desired power spectrum of our density field by (Salmon (1996))

$$T'(k) = [P(k)]^{1/2} \,. \tag{2.57}$$

Thus we may construct a random field with any desired power spectrum via the following steps:

1. Generate an uncorrelated gaussian white noise sample, $N(x)$, for each real space gridpoint
2. Take the Fourier transform of this white noise
3. Multiply the noise by the transfer function as given previously
4. Inverse Fourier transform this result to obtain the real space density contrast field

This method is very similar to the k-space approach to generating Gaussian random fields, but with the addition of an extra Fourier transform (step 1). This makes the generation of single level Gaussian random fields more computationally expensive than the k-space approach , but is absolutely critical when the algorithm is extended to multiple levels.

2.4.3 Generation of 'Zoomed' Initial Conditions

As discussed in section 2.4.2 in order to obtain the correct clustering statistics of haloes in a cosmological setting we require the use of large simulation volumes. However, in order to simulate an individual galaxy in a realistic manner we require a spatial resolution better than 1kpc and a mass resolution better than $10^6 M_\odot$. This consideration has led to many authors performing 'zoomed' simulations, where most of the particles are concentrated in regions of interest, and other regions are represented only by relatively few particles. Generating initial conditions of this form provides additional challenges on top of those required to generate a uniform volume. In this section we will discuss how zoomed initial conditions can be generated, both in real and k-space.

In k-space

To make random fields on multiple levels of refinements an approach followed by many workers is that due to Katz et al. (1994). Here the Gaussian random field is first generated at the coarsest level, then again at a higher resolution ensuring that all of the random numbers chosen for long wavelength power match up exactly with the coarse data. These density fields are then overlaid onto each other and the high resolution data is then thrown away in the regions where it is not needed.

This approach has the advantage that it is simple to code, but the maximum resolution of any region in the simulation is limited by the largest FFT that can be performed as in order to generate even a small region at high resolution the high resolution data for the entire simulation volume is generated by direct FFT and then most of it is thrown away. This fact limits the maximum resolution of any initial conditions generated by this method on today's computers to around 2048^3 (holding a single 2048^3 array of single precision real numbers in memory takes 32Gb of RAM). For this reason alternative approaches to the generation of Gaussian random fields have been implemented, we will now discuss one of these.

In Real Space

Pen (1997) and Bertschinger (2001) extended the real-space techniques of Salmon (1996) to multiple levels of refinement. In this section we will discuss the generation of initial conditions on two levels of refinement, as represented in figure 2.13. In figure 2.13 the coarse level has M vertices, the subvolume has $M_s = 3$ vertices and refinement factor,

$r = 2$. We can then describe any gridpoint using two indices, m for the coarse grid and n for the refined grid. The first step in generating a refined set of initial conditions is to create an additional sample of white noise on the refined grid, N_1, which maintains the same large-wavelength structure as the white noise on the coarse grid, N_0. This may be accomplished by (Hoffman and Ribak (1991)) the process

$$N(m,n) = N_1(m,n) + N_0(m) - \bar{N}_1(m)\,. \tag{2.58}$$

This just represents that each coarse gridpoint has its value propagated to its associated r^3 refined points, then a high frequency correction $N_1 - \bar{N}_1$ is added. Given this white noise we now need to calculate the corresponding density field, which is considered to be split up into long and short-wavelength components,

$$\delta(m,n) = \delta_0'(m,n) + \delta_1(m,n)\,, \tag{2.59}$$

where $\delta_0'(m,n)$ represents the contribution of the coarse density field, $\delta_0(m)$, to the high resolution density field. $\delta_1(m,n)$ is the additional short wavelength contribution from the refined region.

The *short-wavelength contribution* may be calculated simply from

$$\delta_1(m,n) = [N_1(m,n) - \bar{N}_1(m)] \star T'\,, \tag{2.60}$$

where $\star$ represents a convolution. It was shown by Bertschinger (2001) that it is possible to carry out this convolution using only $(rM_s)^3$ gridpoints and as such the full $(rM)^3$ calculation is avoided.

The *long-wavelength contribution* could, in principle, be evaluated by resampling the coarse grid at every point up to the full resolution of the refined region (i.e. an entire r^3M^3 points), and then explicitly carrying out the convolution over the entire volume. This is impractical and runs into exactly the problem we are trying to avoid through working in real space as we would be forced to carry out convolutions over the entire $(rM)^3$ volume. One solution to this problem is to write $\delta_0'(m,n)$ as a convolution of the coarse density field with some filtering function:

$$\delta_0'(m,n) = \sum_{m',n'} \delta_0(m')W(m-m', n-n')\,. \tag{2.61}$$

This equation has a simple interpretation. The coarse grid density is spread to the fine grid by mapping each value of $\delta_0(m)$ to r^3 grid points. This leads to aliasing effects due to each bunch of r^3 gridpoints having exactly the same value, and in k-space this leads to

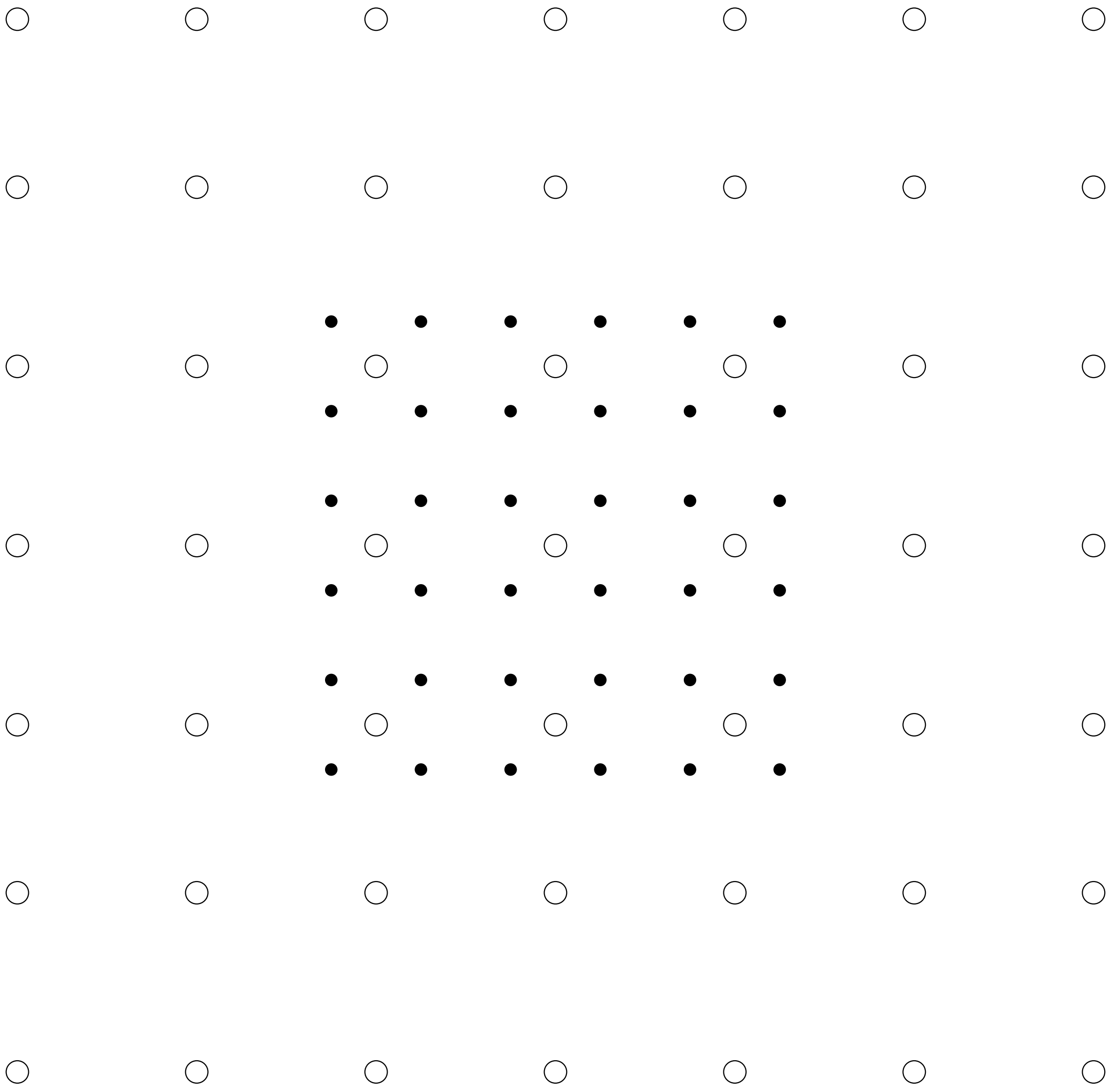


Figure 2.13: Simple depiction of a nested, hierarchical grid. The hollow points represent a coarse grid, with size $M = 7$, the filled points represent a refined region with refinement factor, $r = 2$ and the size of the refined region is $M_s = 3$

high frequency distortions, thus just setting $\delta'(m, n) = \delta_0(m)$ leads to an incorrect power spectrum. Bertschinger (2001) showed that this problem can be avoided through the use of an *aliasing filter*, which for certain forms of $W(m, n)$ is exactly equivalent to carrying out the full convolution.

Figure 2.14 shows a typical CDM overdensity field with two different levels of refinement generated by the methods described above. The refined density fields in each of the panels was generated without resorting to carrying out the full $(rM)^3$ calculation, and as such grids at much higher levels of refinement can be generated without prohibitive memory and computational costs. This section contains a simplified sketch of the method by which the real-space refinement technique works, more detail is contained in Pen (1997) and Bertschinger (2001)

2.4.4 An Initial Conditions Toolkit

Generating a zoomed set of initial conditions consists of very many separate and distinct steps, we need to: generate the transfer function and initial particle distribution. Create density and velocity fields for each different mass resolution. Finally the particles on each level of refinement must be displaced according to the Zel'Dovich (1970) approximation.

In order to facilitate this process an interactive toolkit has been developed, which allows interactive selection of cosmological parameters and refinement positions (figure 2.15).

In addition to generating cosmological density fields using the real space techniques of Bertschinger (2001) the package facilitates the easy resimulation of haloes. Two features of the code that make this process easy are the method of particle refinement, and the way in which noise samples are generated so that their large scale properties remain the same regardless of the resolution of the sample. The methods by which each of these processes is carried out is detailed in the following section

Particle Refinement on Individual Haloes

When creating the initial particle distributions for the simulation of an individual halo it is important that we both minimize the number of high resolution particles in the simulation (in order that it remains computationally cheap to run), and ensure that the halo itself remains uncontaminated by low resolution particles. These two constraints are sometimes difficult to satisfy at the same time. We will now describe the process by which our code marks out individual haloes and ensures they remain uncontaminated until redshift zero. Figure 2.16 shows the steps required to identify and recursively refine a region of a simulation. Each panel represents a different part of the process

top-left: A low resolution simulation is run, all particles that are within the halo at redshift zero are marked, and their positions in the low resolution initial conditions are noted. This defines the region of the early universe that will collapse into the halo at redshift zero

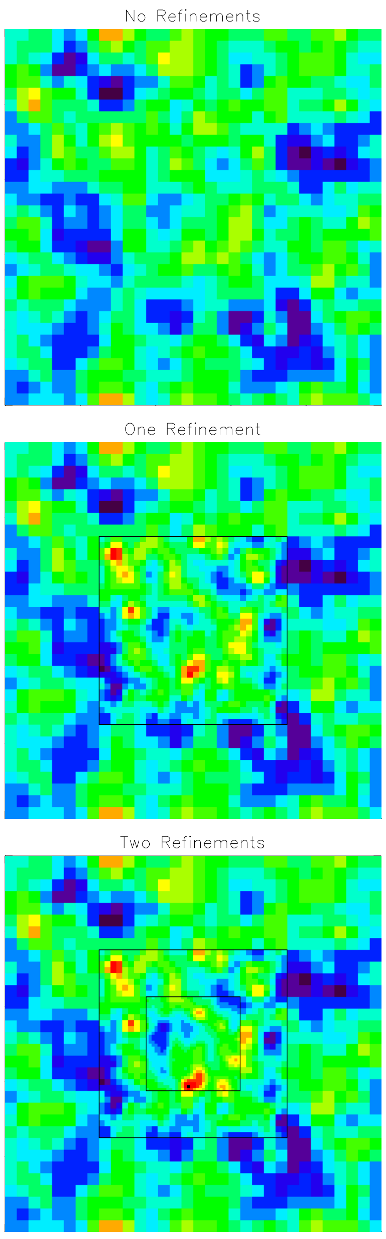
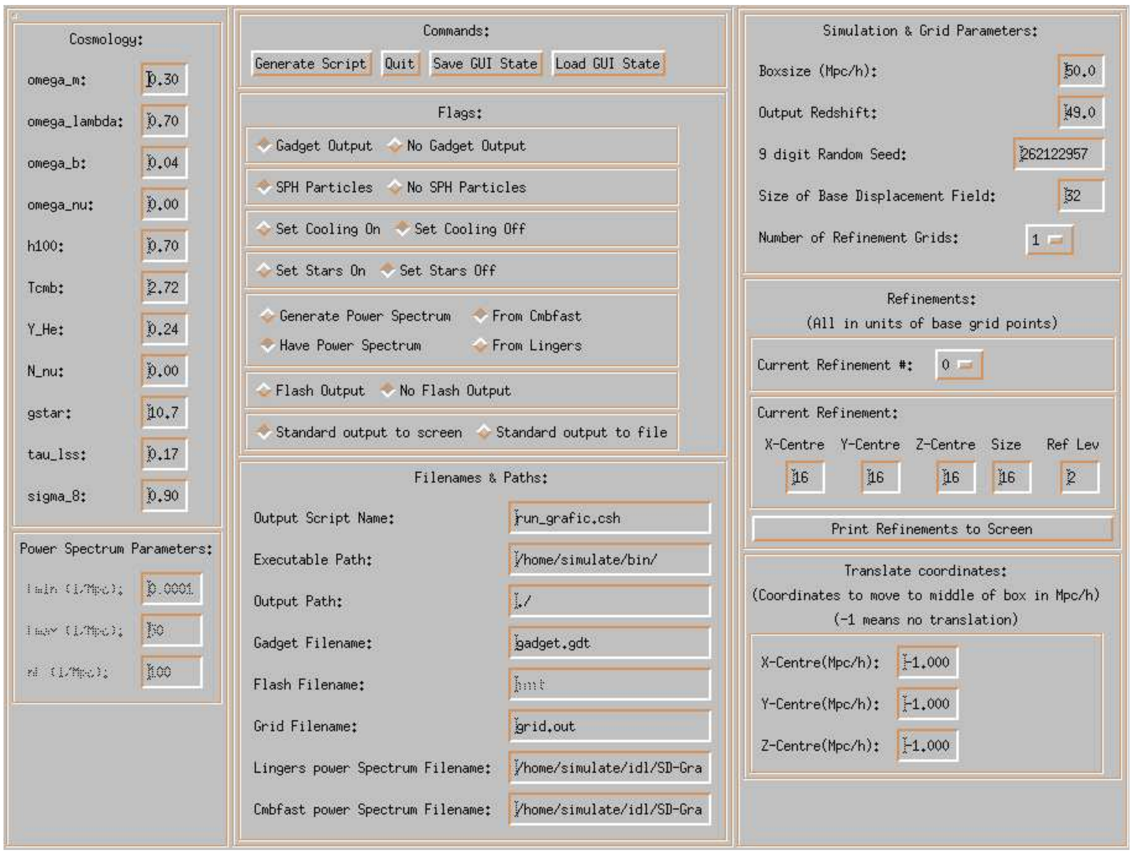


Figure 2.14: Slice through a typical overdensity field for a CDM set of initial conditions. The upper panel shows an unrefined density field with $M = 32$. The central plot depicts the same field with an additional subgrid ($r = 2$, $M_s = 16$), and the lowest grid shows a second, nested refinement ($r = 4$, $M_s = 8$). Note that in each case the large scale features of the field agree well, but additional small-scale power is added to the simulation volume.



Cosmology:

omega_m:	0.30
omega_lambda:	0.70
omega_b:	0.04
omega_nu:	0.00
h100:	0.70
Tcmb:	2.72
Y_He:	0.24
N_nu:	0.00
gstar:	10.7
tau_lss:	0.17
sigma_8:	0.90

Power Spectrum Parameters:

kmin (1/Mpc):	0.0001
kmax (1/Mpc):	50
nk (1/Mpc):	100

Commands:

Generate Script · Quit · Save GUI State · Load GUI State

Flags:

Gadget Output · No Gadget Output

SPH Particles · No SPH Particles

Set Cooling On · Set Cooling Off

Set Stars On · Set Stars Off

Generate Power Spectrum · From Cmbfast

Have Power Spectrum · From Lingers

Flash Output · No Flash Output

Standard output to screen · Standard output to file

Filenames & Paths:

Output Script Name:	run_grafic.csh
Executable Path:	/home/simulate/bin/
Output Path:	./
Gadget Filename:	gadget.gdt
Flash Filename:	hmt
Grid Filename:	grid.out
Lingers power Spectrum Filename:	/home/simulate/idl/SD-Gra
Cmbfast power Spectrum Filename:	/home/simulate/idl/SD-Gra

Simulation & Grid Parameters:

Boxsize (Mpc/h):	50.0
Output Redshift:	49.0
9 digit Random Seed:	262122957
Size of Base Displacement Field:	32
Number of Refinement Grids:	1

Refinements:
(All in units of base grid points)

Current Refinement #: 0

Current Refinement:

X-Centre	Y-Centre	Z-Centre	Size	Ref Lev
16	16	16	16	2

Print Refinements to Screen

Translate coordinates:
(Coordinates to move to middle of box in Mpc/h)
(-1 means no translation)

X-Centre(Mpc/h):	-1.000
Y-Centre(Mpc/h):	-1.000
Z-Centre(Mpc/h):	-1.000

Figure 2.15: Grapical User Interface for the initial conditions toolkit, allowing for simple, interactive input and design of cosmological initial conditions.

top-right: It is important that particles that end up in the halo are completely surrounded by other high-resolution particles, in order that their properties are self consistently calculated and no spurious numerical effects are introduced. The first step of ensuring this is the case is to map the low-resolution halo particles to a three dimensional grid, note that there are holes in the grid, and some particles are disconnected from the rest of the region.

bottom-left: The grid is convolved a number of times with the three-dimensional version of the following kernel

$$\mathbf{X} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

This has the effect of filling in holes in the marked region, and expands the region so that previously unlinked regions are now linked.

bottom-right: All low resolution particles lying within the expanded grid are marked and then split into 8 lower mass particles. Particles that interact with the halo should now be part of one contiguous region, and should be surrounded by other particles of the same mass. Figure 2.17 shows an example of this process being carried out recursively. At redshift zero The resulting halo is uncontaminated by high resolution particles out to a few times r_{200}, and there are a minimal number of particles in the initial conditions. If, at any point, it is found that the halo is contaminated, it is easy to mark additional particles, and so extend the high resolution region, until the halo does not become contaminated by redshift zero.

Resolution Independent White Noise

One disadvantage of generating noise samples in real-space is that it is very difficult to increase the resolution of a uniform grid whilst maintaining the same large scale power. In order that our initial conditions toolkit can, for a given random seed, *always* reproduce the same large scale structure we require a more sophisticated method of generating white noise samples. We choose to work in k-space and generate numbers such that the lowest k-vectors are always filled in first with zero-mean gaussian deviates. The noise sample is then obtained by performing an inverse FFT on this field. If we require the same noise field at a different resolution this process is repeated, with higher frequency modes incorporated, the underlying low frequency structure will, however, remain unchanged.

Putting together the real-space refinement techniques with a simple, interactive initial condition design package and the halo resimulation techniques discussed in this section allows us to rapidly and easily construct catalogues of resimulated haloes in any cosmology, and to scale up and down the mass resolution of any one of these haloes without affecting the large scale properties of the simulation. In the next section we use this code to probe the effects of mass resolution on the properties of simulated haloes.

2.5 Resolution Studies

The effects of numerical resolution on the properties of simulated galaxies is numerically complex, hard to quantify, especially in the presence of complex physical processes for instance SN feedback and radiative cooling. In this section we attempt to quantify the effects of particle number on the properties of individual MW sized haloes.

Early galaxy formation simulations (e.g. Navarro and White (1994)) reported a huge loss of angular momentum in galaxies, leading to the formation of galaxies strongly dominated by a central concentration of cold gas. Governato et al. (2004) and Kaufmann et al. (2006) have shown that the angular momentum problem is due to poor mass and spatial resolution, which may lead to significant angular momentum loss in baryonic disks embedded in dark matter haloes.

Additionally Governato et al. (2006) found that as the number of particles in the halo approached several million, the resulting galaxy became increasingly less centrally concentrated. It is clear that a thorough understanding of the effects of resolution on the properties of simulated galaxies is a non-trivial problem but is necessary in order to fully understand galaxy formation.

The work described in this section represents preliminary work in this direction, comparing a set of MW sized haloes generated using the initial conditions code of section 2.4.4 at a variety of mass resolutions.

2.5.1 The Simulation Set

In order to identify suitable candidates for resimulation a uniform volume of size 50Mpc containing 192^3 dark matter particles was evolved from redshift 99 to redshift zero (using $\Omega_0 = 0.3$, $\Omega_b = 0.044$, $h_{100} = 0.7$, $\sigma_8 = 0.9$). From this volume MW mass haloes ($\sim 10^{12} M_\odot$) were selected and resimulated at a variety of mass resolutions. Table 2.5.1 contains details of the mass resolutions in different simulations. The highest mass reso-

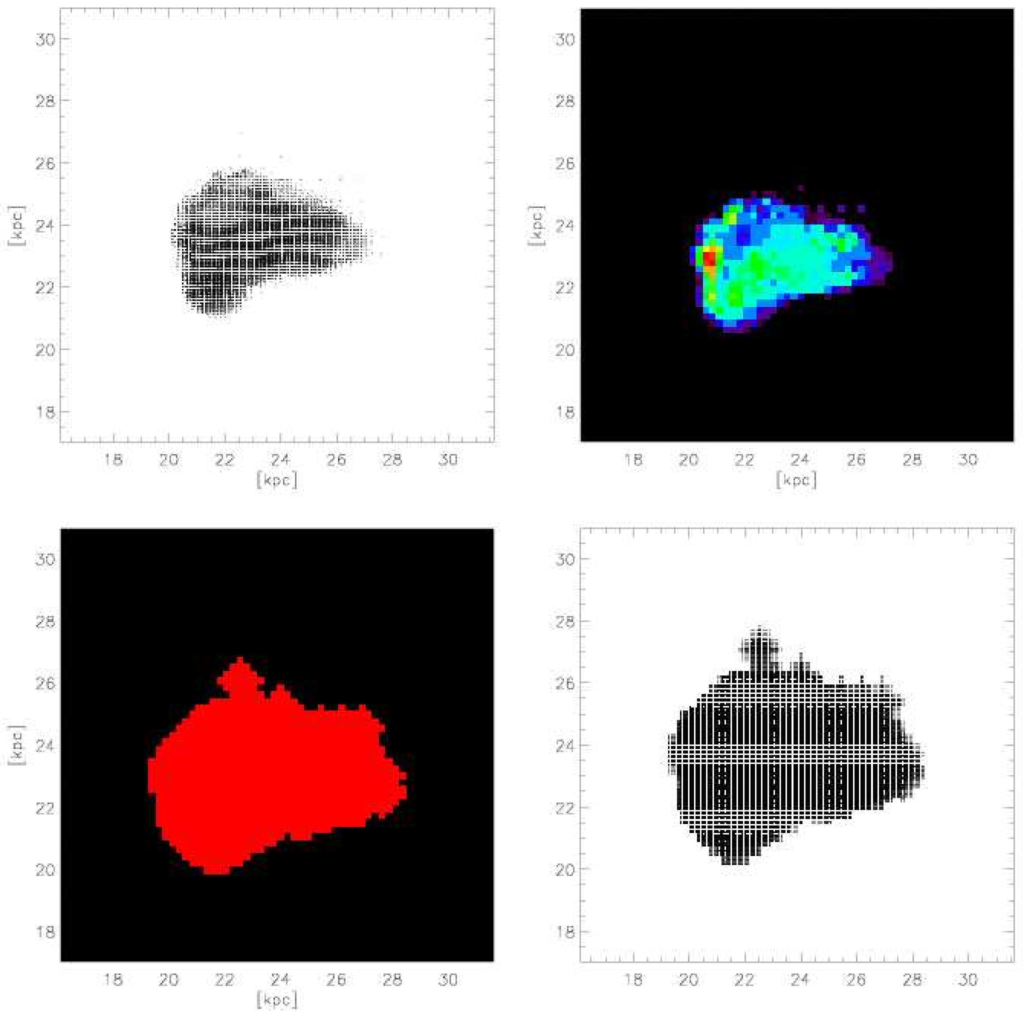


Figure 2.16: The process by which individual haloes are marked out in a zoomed simulation. The process is described in detail in the text and will be summarized here. *top-left:* a low resolution simulation is run, and the particles that end up in the halo are traced back to the initial conditions. *top-right:* these particles are mapped to a grid. *bottom-left:* this grid is then convolved with a kernel (equation 2.4.4), which ensures that the halo remains uncontaminated. *bottom-right:* particles in the marked region are split into eight lower mass particles.

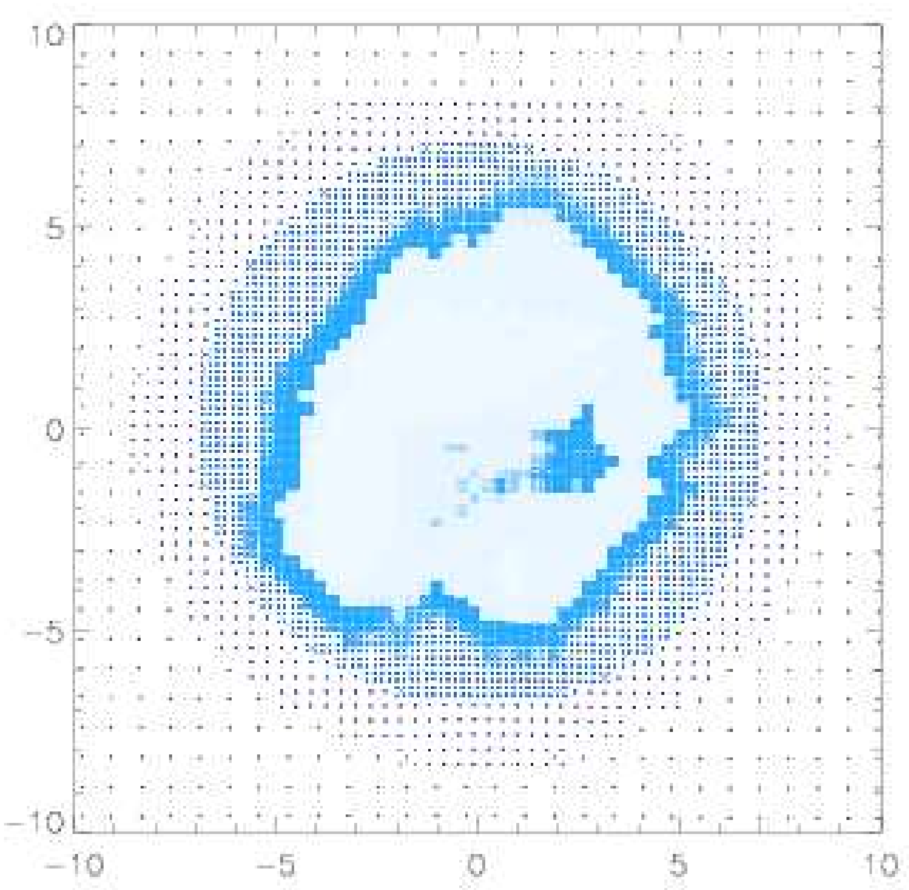


Figure 2.17: Thin slice through the initial ($z = 99$) particle distribution in a zoomed galaxy simulation. The darker the colour of a point the more massive the particle it represents. The ratio of masses between the largest and smallest particles in this simulation is ~ 5000

Name	Mass Resolution ($M_\odot$)	Gravitational Softening (kpc)
Uniform	7.21×10^8	9.11
X0X2	9.01×10^7	4.61
X0X3	2.67×10^7	3.03
X0X4	1.12×10^7	2.27
X0X5	5.76×10^6	1.82
X0X6	3.34×10^6	1.52

Table 2.1: Details of the initial particle masses in each of the simulations. The naming convention followed is that the numbers represent one-dimensional spatial resolutions ($\propto N_{part}^{1/3}$), as a multiple of the resolution used in the uniform simulation. The mass represents the smallest dark matter particle mass in the refined region.

lution haloes have a dark matter mass resolution of $2.84 \times 10^6 M_\odot$, corresponding to of order a few million dark matter particles within the virial radius of the halo. The lowest mass resolution simulations contain only a few tens of thousands of particles within the virial radius of the halo. The gravitational softening length of each simulation was chosen to be consistent with that used in Reed et al. (2005a), scaled with the linear spatial resolution ($\propto N^{1/3}$) of our simulations.

Each simulation was run with dark matter only, again with dark matter and adiabatic gas and then finally with dark matter, and full physics (star formation, feedback, radiative cooling, as described in Stinson et al. (2006)). In this section we discuss only the dark matter simulations.

2.5.2 The Basic Properties of Haloes

Some global properties of the haloes are plotted as a function of mass resolution in figure 2.19. For each halo the virial radius, r_{200}, defined as the radius at which the mean density contrast of the halo falls to 200 times the critical density of the universe was calculated and in addition each halo was fitted with the universal density profile due to Navarro et al. (1997)

$$\frac{\rho(r)}{\rho_c} = \frac{\delta_c}{\left(\frac{r}{r_s}\right)\left(1+\frac{r}{r_s}\right)^2} . \tag{2.62}$$

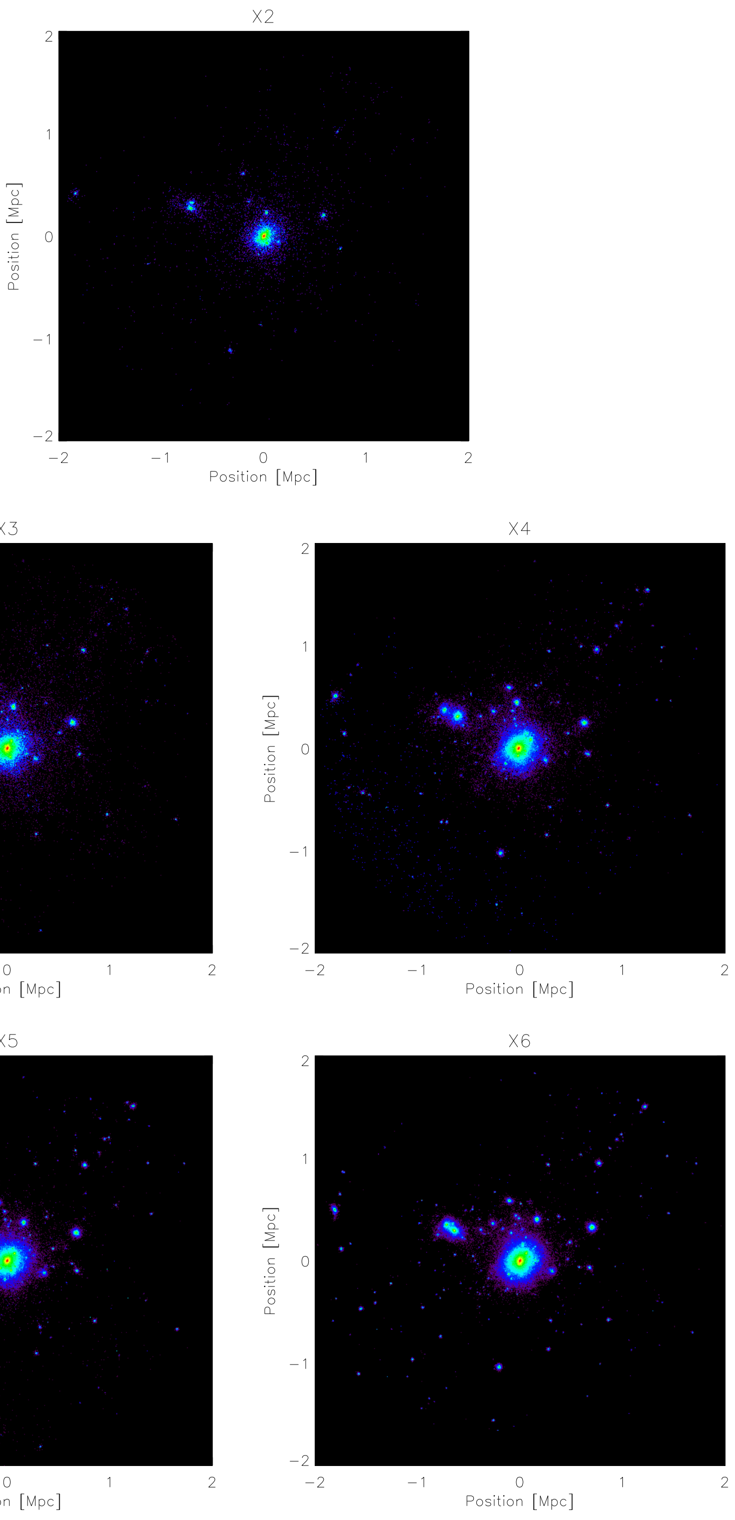


Figure 2.18: Dark matter particles in each of the haloes. It is clear that dark matter substructure in each of the simulations is in almost exactly the same place.

This profile fits the densities of dark matter haloes using two parameters, a central density, δ_c, and a scale radius, r_s. Additionally the *concentration* of a halo may be defined as the ratio of its virial radius to its scale radius. This number measures the shape of the halo (Łokas and Mamon (2001))

Figure 2.19 shows some of these parameters plotted as a function of the mass resolution of the simulation. One encouraging result is that the concentration parameters of the haloes remains approximately constant across all mass resolutions, indicating that the halo shape is independent of resolution. σ_{DM}, the three dimensional dark matter velocity dispersion decreases as the mass resolution in the halo is degraded. As would be expected the central overdensity of the halo, δ_c, decreases as the particle mass is increased. This is due to the effects of larger softening lengths.

Figure 2.20 shows the dark matter density profile of the halo for each mass resolution. The diagonal lines are power laws with slopes -3 and -1.5, as would be predicted from the universal halo fitting formula of Navarro et al. (1997). Figure 2.21 shows the three dimensional dark matter velocity dispersion of each halo, again agreement between the various simulations is very good, especially at large radii, where the features corresponding to individual pieces of substructure match up well.

Figure 2.22 shows the mean radial velocity as a function of radius for all of the different resolutions. It is clear that despite the vastly different refinement sizes and resolutions between the different simulations, the large scale flow velocities of the simulations are always recovered.

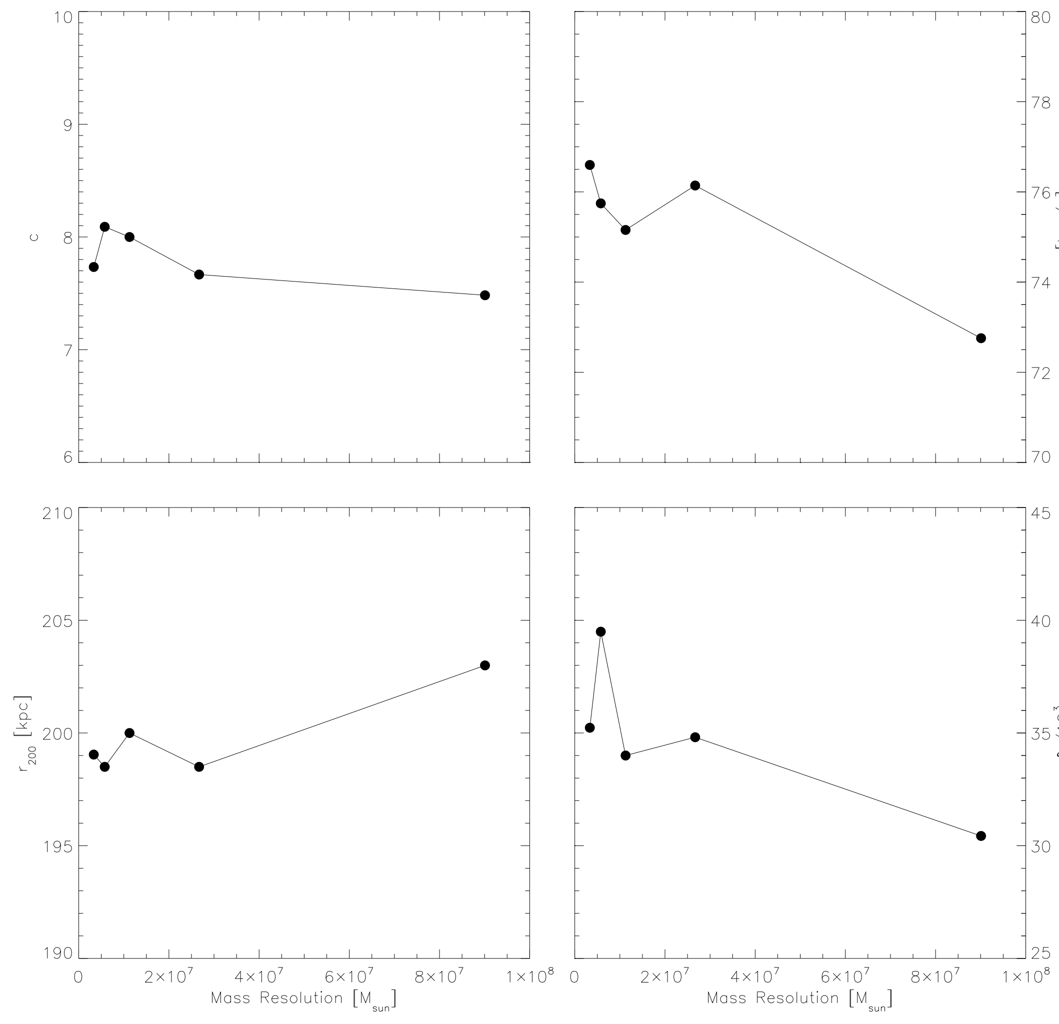


Figure 2.19: The global properties of the dark matter halo as a function of particle mass. *top-left* Halo concentration, *top-right* mean three dimensional velocity dispersion within r_{200}, *bottom-left* virial radius, *bottom-right* galaxy central overdensity as determined from a fit to the NFW profile.

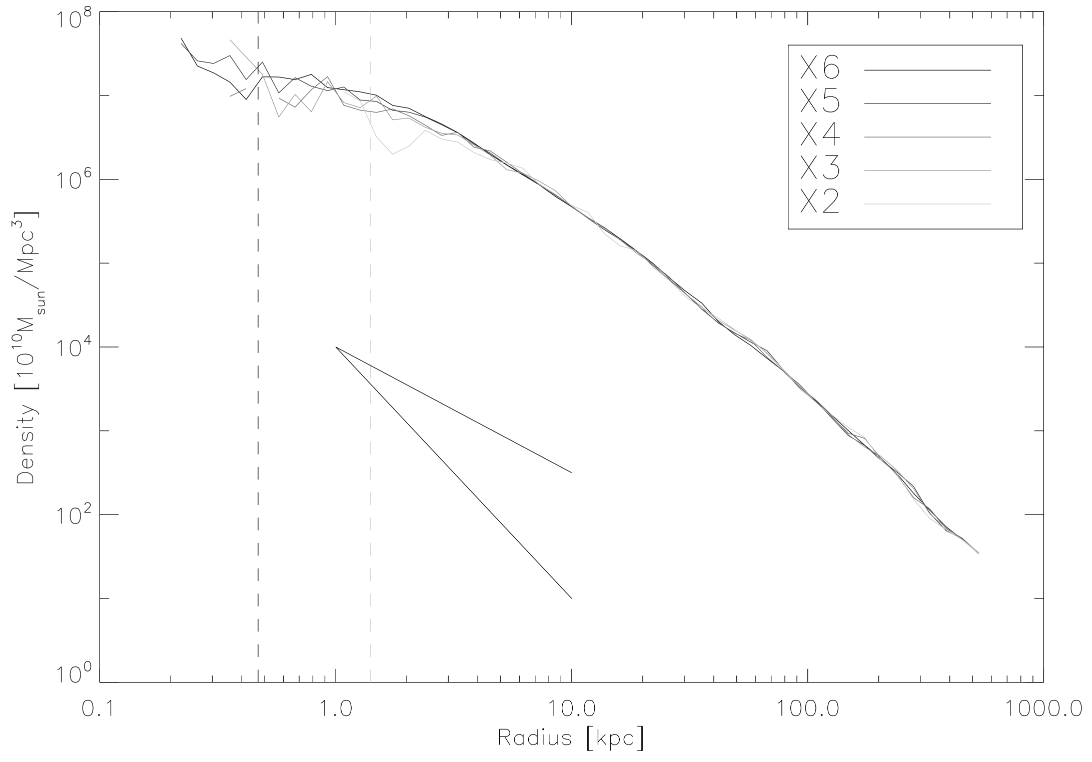


Figure 2.20: Dark matter density profiles. The vertical lines represent the softening length in simulations X2 and X6. The diagonal lines represent power laws with slopes -1 and -3, which are the inner and outer slopes prediced by the NFW density profile. The agreement between the different simulations is good up to the resolution length of each simulation.

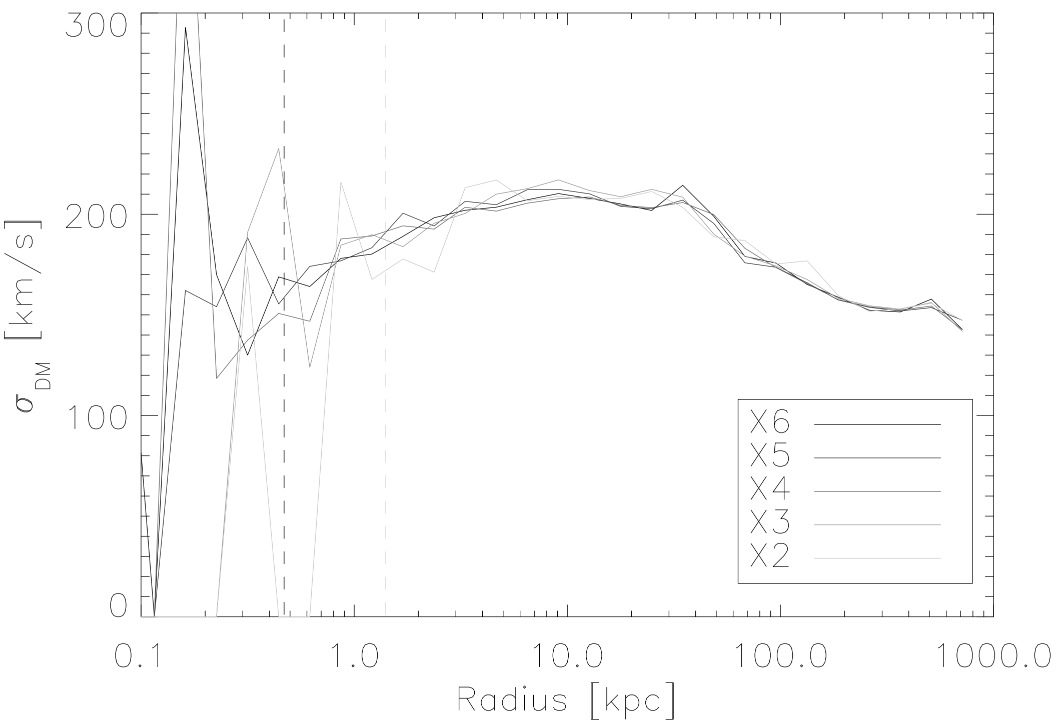


Figure 2.21: Dark matter velocity dispersion profiles, computed by taking the variance of all of the dark matter particles in a series of concentric shells. The vertical lines represent the softening lengths of simulations X2 and X6. The agreement between the different simulations is good up to the resolution length of each simulation.

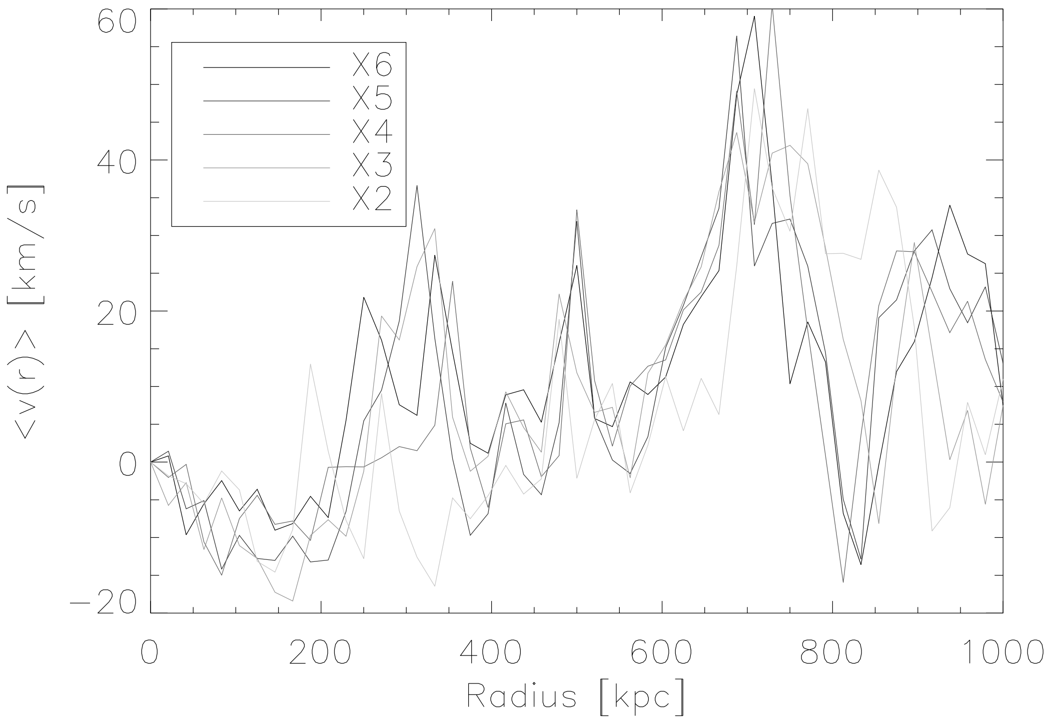


Figure 2.22: Dark matter infall velocity as a function of radius. It is clear that the large scale flow properties are preserved between the simulations, although the mean radial velocity is a very noisy statistic