Mon. Not. R. Astron. Soc. **000**, 000–000 (0000) Printed 14 September 2017 (MN LATEX style file v2.2)

Methods for cosmological N-body simulations

A. Klypin

Astronomy Department, New Mexico State University, Las Cruces, NM, USA

14 September 2017

ABSTRACT

Cosmological N-body simulations play an important role in modern cosmology by providing vital information regarding the evolution of the dark matter: its clustering and motion, about properties of dark matter halos. The simulations are instrumental for the transition of the theoretical cosmology from an inspiring but speculative part of astronomy to the modern precision cosmology. In spite of more than 50 years of development, N-body methods are still a thriving field with invention of more powerful methods providing more accurate theoretical predictions. Here we review different numerical methods (PM, TREE, AMR) and ideas used in the field. Results for the evolution of the dark matter distribution function and power spectrum are briefly introduced.

1 INTRODUCTION

Dark matter is an important component of the Universe. All observational evidence indicates that it dominates dynamics of normal and dwarf galaxies, clusters and groups of galaxies. At high redshifts it provided the force that drove the formation of first galaxies and quasars. The observed large filaments and giant voids all can be understood and explained if we combine the dynamics of the dark matter with the predictions of the inflation model on the spectrum of primordial fluctuations.

The dark matter is likely made of particles that other than the gravity force do not couple with the other matter (such as normal gas, which for some reason in cosmology is called "baryons". Leptons, do not take an offense - you do not weigh much here). There may be some channel of interactions between dark matter particles resulting in annihilation and production of normal particles. However, even if present (no observational evidence so far), this channel is weak and the dark matter is (mostly) preserved over the evolution of the Universe.

How this dark matter evolves and how it forms different structures and objects was an active field of research for a very long time. The first (somewhat) realistic N-body simulation – collapse of a cloud of 300 self-interacting particles – was done by P.J.E. Peebles Peebles (1970). Remember that at that time of the dawn of cosmology, there was no dark matter, the hot gas x-ray gas in clusters had not been yet discovered (it was discovered in 1971), there were no voids or superclusters. So the first N-body simulation had indicated that the force of gravity alone may be responsible for the formation of clusters of galaxies, which was a big step forward. It also discovered a problem – the density profile in the model was not right: too steep. The solution for this problem was continuous mass accretion on the forming cluster instead of a one-time event of collapse (Gunn & Gott 1972).

With the development of computer hardware and new numerical algorithms N-body simulations became more realistic. Klypin & Shandarin Klypin & Shandarin (1983) made the first 3D Particle-Mesh (PM) simulation with 32,768 particles and realistic initial conditions (nearly the same technique as used at present). The model demonstrated that the large scale structure of the Universe should be a net of clusters of galaxies connected with filaments. The model even got a name "the chicken Universe" from one of the plots in the paper, which looked like a chicken. Davies et al. Davis et al. (1985) used Particle-Particle-Particle-Mesh (P^3M) code developed by Hockney & Eastwood to run 32^3 particles with high (at that time) resolution to show that galaxies ("light") should not follow the dark matter ("mass"). That was a very important idea. In their own words: "... kind of bias to be expected if bright galaxies form only at relatively high peaks of the linear density distribution".

From that moment the simulations took off. Larger and larger numbers of particles were used as new codes and new computers became available. For some time it looked almost like a sport: whose simulation has more "muscle". The pace has slowed down in recent years mostly because it became more difficult to analyze the simulations and to make the results accessible to the larger community.

Development of numerical methods was crucial for advances in N-body simulations. At the beginning direct summation technique was used to run the simulations (Peebles 1970; White 1976; Aarseth et al. 1979). At that time – slower processors, no parallel computing – it was difficult to make simulations with more than just a few thousand particles. The main motivation at that time was to develop new computational methods. The number of operations in the direct summation method scales as $\propto N^2$, where N is the number of particles. So, one quickly ran out of available cpu. However, now the situation is different: processors are much faster and the number of cores on a workstation can be sig-

nificant. A simulation with $N = 10^5 - 10^6$ is relatively fast (from few hours to few days). Such simulations can be very useful for testing different ideas and for small runs. It is also very easy to modify the code because everything is very transparent. For example, one can add external tidal force or modify the law of gravity. It is also a great tool for training students: a simple parallel pair-wise summation code can be written in few hours.

Particle-Mesh method (Klypin & Shandarin 1983; Hockney & Eastwood 1988; Klypin & Holtzman 1997) was a big step forward with cpu scaling $\propto N$. However, it requires a large 3D mesh for computation of the gravitational potential. The size of a cell in that mesh defines the force resolution, and, if one needs better resolution, the number of cells should be increased. As the result, one may run out of available computer memory. Still, the PM method is very fast and is easy to implement. It is a part of all more sophisticated and fast codes.

Hybrid codes Particle-Particle-Particle-Mesh (P^3M) (Efstathiou et al. 1985; Hockney & Eastwood 1988) and Adaptive P^3M (Couchman 1991) were popular for some time, but they were superceded by either Adaptive Mesh Refinement codes (Kravtsov et al. 1997; Teyssier 2002; Bryan et al. 1995, 2014) or by TREE codes (Appel 1985; Barnes & Hut 1986; Stadel 2001; Salmon & Warren 1994; Springel et al. 2001). Older review of N-bodymethods can be found in Dolag et al. (2008).

2 COSMOLOGICAL N-BODY PROBLEM: MAIN EQUATIONS

In order to derive equations for the cosmological N-body problem, one can start with the equations of general relativity and derive equations of motion of self-gravitating nonrelativistic particles in the expanding Universe. For the case of nonrelativistic matter and the weak-field limit, we simply arrive at the Newtonian equations. There are some limitations with this approach: we cannot treat relativistic particles and we neglect time needed for gravitational perturbations to travel from one point to another effectively treating changes in the gravitational potential as instantaneous. However, these effects are not significant for most applications: velocities are typically well below relativistic and effects of the finite time of gravitational perturbations are small.

We start with definitions. Proper $\mathbf{r}$ and comoving coordinates $\mathbf{x}$ are related:

$$\mathbf{r}(\mathbf{x}, t) = a(t)\mathbf{x}(t), \tag{1}$$

where $a(t)$ is the expansion factor. Differentiating eq.(1) over time, we get velocities:

$$\mathbf{v}(\mathbf{x}, t) \equiv \dot{\mathbf{r}} = a\dot{\mathbf{x}} + \dot{a}\mathbf{x} = H\mathbf{r} + \mathbf{v}_{\rm pec}. \tag{2}$$

Here $\mathbf{v}_{\rm pec} = a\dot{\mathbf{x}}$ is the peculiar velocity and $H = \dot{a}/a$ is the Hubble constant. It is also useful to introduce the specific momentum defined as $\mathbf{p} \equiv \mathbf{a^2}\dot{\mathbf{x}} = \mathbf{a}\mathbf{v}_{\rm pec}$.

In cosmology we deal with a rather specific case of the N-body problem. Here discreteness of matter can be neglected. In general this is not the case with the two-body effects gradually accumulating over time. Systems studied in cosmology such as the nonlinear evolution of dark matter clustering do not suffer from the two-body scattering and can be treated using the collisionless Boltzmann equation paired with the Poisson equation for the gravitational potential. In the comoving coordinates the Boltzmann equation describing the evolution of the distribution function $f(\mathbf{x}, \mathbf{p}, \mathbf{t})$ can be written as:

$$\frac{\partial f}{\partial t} + \mathbf{x}\frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \nabla\phi\frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \mathbf{0}, \tag{3}$$

where peculiar gravitational potential $\phi(\mathbf{x})$ is related with the normal gravitational potential Φ as $\Phi = 2\pi G\rho_b r^2/3 + \phi$ where the first term is the potential of the background (constant over space) density field ρ_b and the second term is the deviation from the background. Changing coordinates from proper $\mathbf{r}$ to comoving $\mathbf{x}$ we can write the Poisson equation as:

$$\nabla^2\phi = 4\pi G a^2(\rho(\mathbf{x}) - \rho_{\mathbf{b}}) = \mathbf{4\pi G}\frac{\mathbf{\Omega_0}\rho_{\rm cr,0}}{\mathbf{a}}\delta_{\rm dm}(\mathbf{x}, \mathbf{t}). \tag{4}$$

Here $\delta_{\rm dm} \equiv (\rho_{\rm dm}(\mathbf{x}, \mathbf{t}) - \langle\rho_{\rm dm}\rangle)/\langle\rho_{\rm dm}\rangle$ is the dark matter density contrast. Factors Ω_0 and $\rho_{\rm cr,0}$ are the average matter (dark matter plus baryons) density in the units of the critical density and the critical density all taken at the present moment $a = 1$.

Note that the right hand side (r.h.s) of eq.(4) may have a positive or *negative* sign. This is unusual considering that in a normal Poisson equation the density is always positive. The negative sign of the density term in eq.(4) happens in locations where the density is below the average density of the Universe. While there are no real negative densities in the Poisson equation, the regions with the negative r.h.s. of eq.(4) in comoving coordinates act as if there are. For example, in these regions the peculiar gravitational acceleration points away from the center of an underdense region resulting in matter being pushed away from the center. This explains why over time voids (large underdense regions) observed in the large-scale distribution of the dark matter become bigger and more spherical.

The collisionless Boltzmann equation eq.(3) is a linear first order partial differential equation in the 7-dimensional space $(\mathbf{x}, \mathbf{p}, \mathbf{t})$. It has a formal solution in the form of characteristics: a set of curves that cover the whole space. The characteristics do not intersect and do not touch each other. Along each characteristic the value of the distribution function is preserved. In other words, if at some initial moment t_i we have coordinate $\mathbf{x_i}$, momentum $\mathbf{p_i}$, and phase-space density f_i, then at any later moment t along the characteristic we have $f(\mathbf{x}, \mathbf{p}, \mathbf{t}) = \mathbf{f_i}(\mathbf{x_i}, \mathbf{p_i}, \mathbf{t_i})$. Equations of the characteristics, the Poisson equation, and the Friedmann equation can be written as follows:

$$\frac{d\mathbf{x}}{da} = \frac{\mathbf{p}}{a^3 H}, \qquad \frac{d\mathbf{p}}{da} = -\frac{\nabla\phi}{aH}, \tag{5}$$

$$\nabla^2\phi = \frac{3}{2}\frac{H_0^2\Omega_0\delta_{\rm dm}}{a}, \tag{6}$$

$$H^2 = H_0^2\left(\frac{\Omega_0}{a^3} + \Omega_{\Lambda,0}\right), \quad \Omega_0 + \Omega_{\Lambda,0} = 1. \tag{7}$$

Here we specifically assumed a flat cosmological model with the cosmological constant characterized by the density parameter $\Omega_{\Lambda,0}$ at redshift $z = 0$.

There are numerical factors in eqs.(5–6) that obscure the fact that the equations of characteristics are nothing but the equations of motion of particles under the force of gravity. These equations are almost the equations of the N-body problem in the comoving coordinates. However, there are differences. Characteristics cover the whole phase-space which we cannot do in simulations that use a finite number of particles. Instead, we approximate the phase-space by placing particles at some positions and giving them initial momenta.

How exactly we place the particles depends on the problem to be solved. For example, if a large simulation volume is expected to be resolved everywhere with the same accuracy, then particles should be nearly homogeneously distributed initially and have the same mass. If instead a small region should be resolved with a higher resolution than its environment, than we place lots of small particles in the region and cover the rest of the volume with few large particles.

Because we intend to produce an approximate solution for the continuous distribution of matter in space as described by the Boltzmann-Poisson equations, we may not even think that we solve the N-body problem – an ensemble of point masses moving under the force of gravity. For example, at the initial moment the volume of a simulation may be covered by many small non-overlapping cubes (not points). Then each cube is treated as a massive particle with some size, mass, and momentum. So, instead of N point masses we have N small cubes. This is definitely a better approximation for the reality. Indeed, these types of approximations are used in many simulations. For example, in Particle-Mesh (PM) simulations dark matter particles are small cubes with constant density and size. In Adaptive Mesh Refinement (AMR) codes particles are also cubes with the size of the cube decreasing in regions with better force resolution.

The last clarification is related to the baryons. In order to treat the baryons properly, we need to include equations of hydrodynamics and add gas density to the Poisson equation. We clearly do not do it in N-body simulations. Still, we cannot ignore baryons. They constitute a significant fraction of mass in the Universe. If we neglect baryons, there will be numerous defects. For example, the growth rate of fluctuations even on large scales will be wrong and virial masses will not be correct. In cosmological N-body simulations we assume that all the mass – dark matter and baryons – is in particles and each particle represents both dark matter and baryons with the ratio of the two being equal to the cosmological average ratio.

3 SIMPLE N-BODY PROBLEM: PAIR-WISE SUMMATION

We start discussion of numerical techniques with a very simple case: forces are estimated by summing up all contributions from all particles and with every particle moving with the same time-step. The computational cost is dominated by the force calculations that scale as N^2, where N is the number of particles in the simulation. Because of the steep scaling, the computational cost of a simulation starts to be prohibitively too large for $N \gtrsim 10^6$. However, simulations with a few hundred thousand particles are fast, and there are numerous interesting cases that can be addressed with $N < 10^6$ particles. Examples include major-mergers of dark matter halos, collisions of two elliptical galaxies, and tidal stripping and destruction of a dwarf spheroidal satellite galaxy moving in the potential of the Milky Way galaxy. In these cases it is convenient to use proper, not comoving coordinates.

The problem that we try to solve numerically is the following. For given coordinates $\mathbf{r}_{\rm init}$ and velocities $\mathbf{v}_{\rm init}$ of N massive particles at moment $t = t_{\rm init}$ find their velocities $\mathbf{v}$ and coordinates $\mathbf{r}$ at the next moment $t = t_{\rm next}$ assuming that the particles interact only through the Newtonian force of gravity. If $\mathbf{r_i}$ and m_i are the coordinates and masses of the particles, then the equations of motion are:

$$\frac{d^2\mathbf{r_i}}{dt^2} = -G \sum_{j=1, i\neq j}^{N} \frac{m_j(\mathbf{r_i} - \mathbf{r_j})}{|\mathbf{r_i} - \mathbf{r_j}|^{\mathbf{3}}}, \qquad (8)$$

where G is the gravitational constant. Two steps should be taken before we start solving equations (8) numerically.

First, we introduce force softening: we make the force weaker ("softer") at small distances to avoid very large accelerations, when two particles collide or come very close to each other. This makes numerical integration schemes stable. Another reason for softening the force at small distances is that in cosmological environments, when one deals with galaxies, clusters of galaxies, or the large-scale structure, effects of close collisions between individual particles are very small and can be neglected. In other words, the force acting on a particle is dominated by the cumulative contribution of all particles, not by a few close individual companions.

There are different ways of introducing the force softening. For mesh-based codes, the softening is defined by the size of cell elements. For TREE codes the softening is introduced by assuming a particular kernel, and it is different for different implementations. The simplest and often used method is called the Plummer softening. It replaces the distance between particles $\Delta r_{ij} = |\mathbf{r_i} - \mathbf{r_j}|$ in eq. (8) with the expression $(\Delta r_{ij}^2 + \epsilon^2)^{1/2}$, where ϵ is the softening parameter.

Second, we need to introduce new variables to avoid dealing with too large or too small physical units of a real problem. This can be done in a number of ways. For mesh-based codes, the size of the largest resolution element and the Hubble velocity across the element give scales of distance and velocity. Here we use more traditional scalings. Suppose M and R are scales of mass and distances. These can be defined by a particular physical problem. For example, for simulations of an isolated galaxy M and R can be the total mass and the initial radius. It really does not matter what M and R are. The scale of time t_0 is chosen as $t_0 = (GM/R^3)^{-1/2}$. Using M, R, and t_0 we can change the physical variables $\mathbf{r_i}$, $\mathbf{v_i}$, m_i into dimensionless variables using the following relations:

$$\mathbf{r_i} = \tilde{\mathbf{r}}_\mathbf{i}\,\mathbf{R}, \quad \mathbf{v_i} = \tilde{\mathbf{v}}_\mathbf{i}\,\frac{\mathbf{R}}{\mathbf{t_0}}, \quad \mathbf{m_i} = \tilde{\mathbf{m}}_\mathbf{i}\,\mathbf{M}, \quad \mathbf{t} = \tilde{\mathbf{t}}\,\mathbf{t_0}. \qquad (9)$$

We now change the variables in eq. (8) and use the Plummer softening:

$$\tilde{\mathbf{g}}_\mathbf{i} = -\sum_{j=1}^{N} \frac{\tilde{m}_j(\tilde{\mathbf{r}}_\mathbf{i} - \tilde{\mathbf{r}}_\mathbf{j})}{\left(\Delta\tilde{r}_{ij}^2 + \tilde{\epsilon}^2\right)^{3/2}}, \qquad \frac{d\tilde{\mathbf{v}}_\mathbf{i}}{d\tilde{t}} = \tilde{\mathbf{g}}_\mathbf{i}, \qquad \frac{d\tilde{\mathbf{r}}_\mathbf{i}}{d\tilde{t}} = \tilde{\mathbf{v}}_\mathbf{i}, \qquad (10)$$

where $\tilde{\mathbf{g}}_\mathbf{i}$ is the the dimensionless gravitational acceleration.

Note that these equations look exactly as eq. (8), if we formally set $G = 1$ and $\epsilon = 0$.

All numerical algorithms for solving these equations include three steps, which are repeated many times:

- find acceleration $g(r)$
- update velocity $v = v + \Delta v(g)$
- update coordinates $r = r + \Delta r(v)$

Here is a simple fragment of a Fortran-90 code that does it using direct summation of accelerations:

```
Program Simple
   .... (set parameters)
   .... (read data)
   Do                    ! Main loop of integration
      Call Acceleration  ! find accelerations
      v = v+g*dt         ! update velocities
      X = X+v*dt         ! update coordinates
      t = t +dt          ! update time
      If(t> t_end)exit   ! stop when time is up
   End do
end Program Simple

Subroutine Acceleration ! find accelerations
    g = 0.      ! set accelerations to zero
    Do i=1,N    ! for each particle i
    Do j=1,N    !      add contributions
      g(:,i)=g(:,i)+m(j)*(X(:,j)-X(:,i))/ &
             sqrt(SUM((X(:,j)-X(:,i))**2+eps2)**3)
    EndDo
    EndDo
end Subroutine Acceleration
```

In this code we extensively use Fortran-90 feature of vector operations. For example, the statement $V = V + g \times dt$ means "do it for every element" of arrays $V(i, j)$ and $g(i, j)$. There are simple ways of speeding up the code. Particles can be assigned into groups according to their accelerations with each group having their own time step. In this case particles with large accelerations update their coordinates and accelerations more often while particles in low density (and acceleration) regions move with large time step, thus reducing the cost of their treatment. Calculations of the acceleration can be easily parallelized using OpenMp directives. These optimizations can speed up the code by hundreds of times making it a useful tool for simple simulations.

4 MOVING PARTICLES: TIME-STEPPING ALGORITHMS

Numerical integration of equations of motion are relatively simple as compared with the other part of the N-body problem – the force calculations. Still, a wrong choice of parameters or an integrator can make a substantial impact on the accuracy of the final solution and on cpu time. To make arguments more transparent, we write equations of motion in proper coordinates and assume that the gravitational acceleration can be estimated for every particle. In this case the equations of motion for each particle are simply:

$$\frac{d\mathbf{v(t)}}{dt} = \mathbf{g(x)}, \qquad \frac{\mathbf{dx(t)}}{\mathbf{dt}} = \mathbf{v(t)}. \tag{11}$$

Along particle trajectory acceleration can be considered as a function of time $\mathbf{g(x(t))}$. If we know coordinates $\mathbf{x_0}$ and velocities $\mathbf{v_0}$ at some initial moment t_0, then eqs. (11) can be integrated from $t = t_0$ to $t_1 = t_0 + dt$:

$$\mathbf{x_1} = \mathbf{x_0} + \int_{\mathbf{t_0}}^{\mathbf{t_1}} \mathbf{v(t)dt}, \qquad \mathbf{v_1} = \mathbf{v_0} + \int_{\mathbf{t_0}}^{\mathbf{t_1}} \mathbf{g(t)dt}. \tag{12}$$

We now expand $\mathbf{v(t)}$ and $\mathbf{g(t)}$ in the Taylor series around t_0 and substitute those into eqs. (12) to obtain different approximations for $\mathbf{x_1}$ and $\mathbf{v_1}$. If we keep only the first two terms, we get the first order Euler approximation: $\mathbf{x_1} = \mathbf{x_0} + \mathbf{v_0 dt} + \epsilon$, where $\epsilon \approx g_0 dt^2/2 \propto O(dt^2)$ and $\mathbf{v_1} = \mathbf{v_0} + \mathbf{g_0 dt} + \epsilon$, $\epsilon \propto O(dt^2)$. Accuracy and convergence of the Euler integrator are low, and it is never used for real simulations. One may think that adding $g_0 dt^2/2$ term to displacements may increase the accuracy, but it really does not because velocities are still of the first order. In the next iteration the first-order velocity makes the displacement also of the first order. However, we may dramatically improve the accuracy by re-arranging terms in the Taylor expansion in order to kill some high order terms.

Suppose initial velocity is given not at the moment t_0, but a half timestep earlier at $t_{-1/2} = t_0 - dt/2$. Using coordinates at t_0 we find acceleration $\mathbf{g_0(t_0)}$. We now advance velocity one step forward from $t_{-1/2}$ to $t_{1/2} = t_{-1/2} + dt$. Note that when we do it, we use acceleration at the middle of the time step, not on the left boundary of the time step as in the Euler integrator. We then advance coordinates to moment $t_1 = t_0 + dt$ using the new value of velocity. As the result, the scheme of integration is:

$$\mathbf{v_{1/2}} = \mathbf{v_{-1/2}} + \mathbf{g_0 dt}, \qquad \mathbf{x_1} = \mathbf{x_0} + \mathbf{v_{1/2} dt}. \tag{13}$$

In order to find the accuracy of this approximation, we first eliminate velocities from eqs. (13): $\mathbf{x_1 - 2x_0 + x_{-1} = g_0 dt}$. There is an error in this integrator, which we can find by using the Taylor expansion for $\mathbf{x_{\pm 1}}$ up to the fourth order term. This gives:

$$\mathbf{x_1 - 2x_0 + x_{-1} = g_0 dt} + \epsilon, \tag{14}$$

where the error of the approximation is

$$\epsilon = \frac{1}{12}\frac{d^2 g}{dt^2} dt^4. \tag{15}$$

Here the second time derivative of the acceleration is estimated at $t = t_0$. This is a dramatic improvement as compared with the Euler integrator: the error is proportional to dt^4 and, as a bonus, there is a small factor 1/12.

In astronomy the integrator is called the leap-frog because velocities are "jumping over" coordinates and then coordinates are "jumping over" velocities. Figure 1 shows the sequence of advances of coordinates and velocities for the Euler and leap-frog integrators. Besides being more accurate than the Euler integrator, the leap-frog integrator has two more serious advantages. It is time reversible: if we change the direction of time, flip the direction of velocities and repeat all the steps in the reverse direction, we will arrive at the same initial conditions from which we started (neglecting the rounding errors). This preserves one of the basic properties of the Newtonian equations of motion: time reversibility. Another property is the Hamiltonian structure of the equations of motion. Because we solve the equations only approximately, we introduce errors, that in general may be

non-Hamiltonian. Those non-Hamiltonian errors in practice result in a gradual change in the total energy of the numerical solution. The leap-frog integrator has a very good property in that its errors are Hamiltonian. In other words, the numerical solution provided by the leap-frog integration has Hamiltonian structure, but its Hamiltonian is slightly different from the Hamiltonian of the exact solution. Numerical integrators of this type (preserving Hamiltonian structure) are called symplectic. So, a constant-step leap-frog integrator is symplectic. An indication that a integrator is symplectic is the lack of a long-term drift in the energy.

One disadvantage of the leap-frog is that velocities and coordinates are defined at different moments of time. It is convenient to split the integrator into smaller steps that allow for synchronization of time moments and are also easier to modify when the time-step changes. An algorithm of integration of trajectories can be written as a sequence of operators, which advance particle positions (called drifts) and change velocities (called kicks). Let $K(dt)$ be an operator (kick) that advances velocities by time dt. Applying the operator simply means $K(dt) : \mathbf{v} = \mathbf{v} + \mathbf{g}\mathbf{dt}$. Similarly, the drift operator is $D(dt) : \mathbf{x} = \mathbf{x} + \mathbf{v}\mathbf{dt}$. We also need to specify the moment when the gravitational acceleration is calculated and the moment when the decision is made to change the time-step. So, we use G and S operators to indicate those two moments. For example, a simple constant-step leap-frog integrator can be written as sequence of $GK(dt)D(dt)GK(dt)D(dt)$....

Using the K and D operators we can also write the leap-frog integrator which starts with $\mathbf{x}$ and $\mathbf{v}$ defined at the same moment of time and ends at $t + dt$ moment:

$$KDK: \quad K(dt/2)D(dt)GK(dt/2)S. \tag{16}$$

New accelerations are estimated after advancing coordinates, and the change in the time-step dt is made at the end of each time-step. The sequence of actions for the KDK integrator is illustrated in the top panel of Figure 1.

Changing the time-step may be necessary when particles experience a vast range of accelerations, which is typically the case in high-resolution cosmological simulations. However, changing the time-step results in breaking symmetries and reducing the accuracy of the leap-frog integrator. It becomes not time reversible and it loses its ability to preserve the energy. There are some ways to restore these properties, but they are complicated and never used in cosmology.

We illustrate the accuracy and the long-term behavior of different integrators by applying them to a simple yet realistic case of the particle motion in a spherical system with density $\rho \propto r^{-2}$ and gravitational potential $\phi = \ln(r)$. This is a good approximation for the density of dark matter halos with the NFW profile around the characteristic "core" radius. We select an eccentric orbit with the ratio of apo- to pericenter 10:1. This is somewhat larger than the typical ratio of 5:1 in the equilibrium NFW profile, but not unusual. Duration of integration is motivated by how many orbits a star or a dark matter particle orbiting the center of the Milky Way galaxy makes over the age of the Universe. It takes the Sun $\sim 3 \times 10^8$ yrs to make one period. Thus, we get a total of ~ 30 periods of rotation. Assuming a flat rotation curve, a star with radius of 1 kpc will make 300 orbits over the age of the Universe. The number of periods for a star or a dark matter particle is not much different in a dwarf galaxy: velocities are smaller, but so are the distances. Motivated by these numbers, we run tests for few hundred periods. Accuracy of integration also depends on the number of time-steps, which we assume to be 10^5 – a realistic estimate for simulations such as Bolshoi. In our tests we use 500 time-steps for one orbital period.

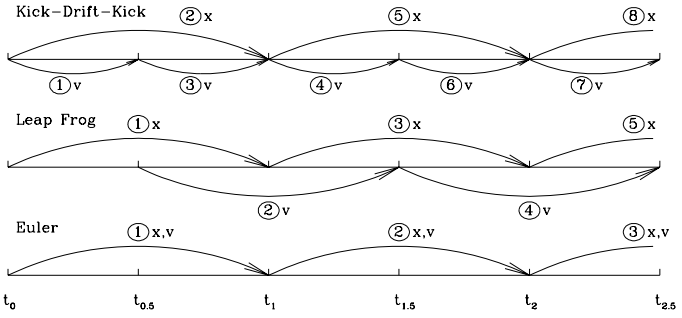

Figure 1. Different schemes for numerical integration of equations of motion. Numbers in circles indicate the sequence of steps in calculating changes in coordinates and velocities with letter following the number showing which parameter – coordinates x or velocities v – is modified. Gravitational acceleration is recalculated after each advance in coordinates.

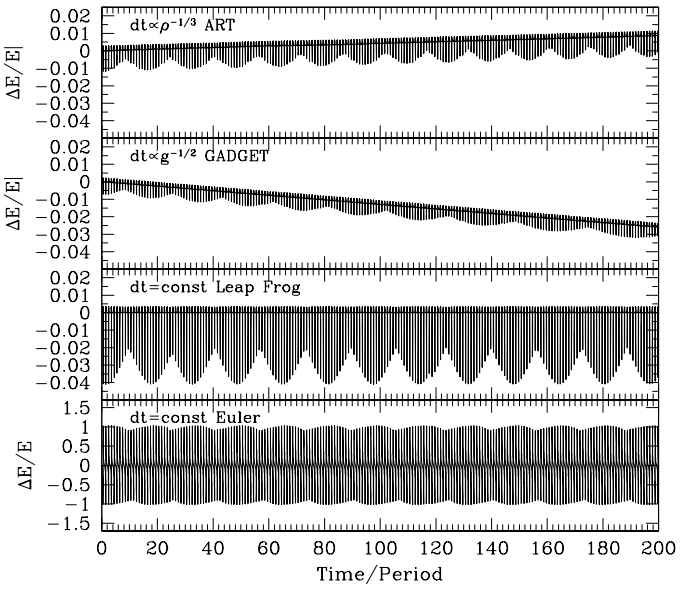

Figure 2. Accuracy of energy conservation for a particle orbiting the center of isothermal density profile $\rho \propto r^{-2}$ in an eccentric orbit with apo- to pericenter ratio 10:1. Trajectories were followed with different integrators, each integrator using 500 time-steps per orbital period. The Euler scheme gives the worst accuracy (note the change in the y-axis). The leap-frog with a constant time-step shows no long-term energy drift, but errors are large as compared with codes with variable time-steps. Errors are smaller for variable time-step integrators, but they also show a linear trend with time.

Figure 2 shows the results obtained with different integrators. The Euler scheme was by far the worst. A constant step leap-frog integrator is clearly much better: errors are much smaller and they do not grow with time; just as expected for a time reversible symplectic integrator. However, the errors are still large. The largest error occurs at

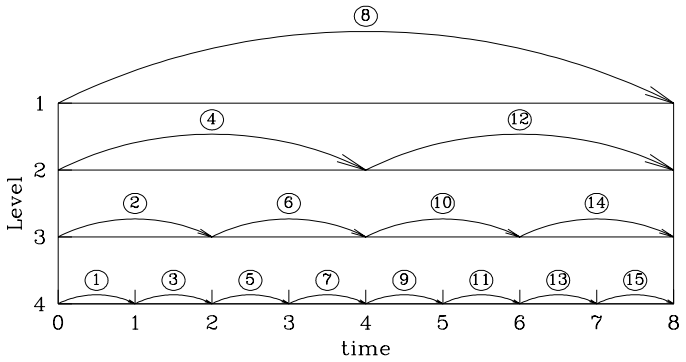

Figure 3. Time stepping scheme for multilevel resolution codes. In this case a four-level hierarchy of steps is chosen. Numbers in circles indicate the order of moving particles at different levels.

the smallest radius where the acceleration is the largest. Using an integrator with smaller time-step at small radius improves the accuracy as demonstrated by two variable time-step integrators used for the test.

Conditions for changing the time-step are different in different codes. For example, in the ART (Kravtsov et al. 1997) and RAMSES (Teyssier 2002) codes the time-step decreases by factor 2 when the number of particles exceeds some specified level (typically 2-6 particles). A cell that exceeds this level is split into eight smaller cells resulting in the drop by 2^3 times of the number of particles in a cell. The time-step also is decreased twice. This prescription gives scaling of the time-step with the local density ρ as $dt \propto \rho^{-1/3}$. Zemp et al. Zemp et al. (2007) advocate a scheme with scaling of $dt \propto \rho^{-1/2}$. The GADGET (Springel 2005) and Pkdgrav code use a scaling with the gravitational acceleration $dt \propto g^{-1/2}$, which for $\rho \propto r^{-2}$ gives $dt \propto \rho^{-1/4}$.

Results for two variable time-step integrators are presented in two top panels of Figure 2. The first uses the GADGET prescription $dt \propto g^{-1/2}$ and the time-step was allowed to change at the end of each time-step. Note that in real GADGET runs the time-step changes only by factor 2 when needed. The second variable time-step integrator uses the ART and RAMSES prescription $dt \propto \rho^{-1/3}$. In our particular case the density changes ten times along the trajectory. So, the time-step changes only once when a particle moves from apocenter to pericenter and once on the way out. The radius of the time-step jump was arbitrarily chosen to be 1/3 of the apocenter radius. Results clearly show improvement in the accuracy, but also indicate that errors show systematic drift with time.

Most of high-resolution N-body codes have particles moving with time-steps that differ by a power of 2 from one group of particles to another. The order of advancing different groups and the order of calculation of the gravity force depends on the particular type of code and implementation. Grouping of particles according to force resolution comes naturally in the Adaptive-Mesh-Refinement codes where a particle is assigned to the highest resolution cell that contains the particle. So, the particle takes the attributes of the cell: its size defines the resolution and the time-step. In TREE codes the grouping can be done by particular adopted conditions for the time-step refinement. Figure 3 gives an example of a sequence of steps in a four-level hierarchy of time-steps used in some AMR codes. In this case we chose a design that attempts to make steps time symmetric. Quinn et al.Quinn et al. (1997) gives examples of stepping diagrams for a TREE code. A different time-stepping sequence is used in ENZO code. For example, see Figure 2 in Bryan et al. (2014).

More detailed discussions of time-stepping in N-body codes can be found in Saha & Tremaine (1992); Quinn et al. (1997); Springel (2005); Binney & Tremaine (2008).

5 PARTICLE-MESH CODES

There are a number of advantages of Particle-Mesh (PM) codes (Klypin & Shandarin 1983; Hockney & Eastwood 1988; Klypin & Holtzman 1997) that make them useful on their own Klypin & Shandarin (1983); White et al. (2014); Izard et al. (2015); Feng et al. (2016) or as a component of more complex hybrid TREE-PM codes Springel (2005); Habib et al. (2014). Cosmological PM codes are the fastest codes available and they are simple. A PM code solves the Poisson equation eq.(4) using a regularly spaced three-dimensional mesh that covers the cubic domain of a simulation. We start with the calculation of the density field on the nodes of the mesh and then proceed with solving the Poisson equation. Once that is done, the gravitational potential is differentiated to produce acceleration and particles are advanced by one time-step.

In order to assign density of particles to the 3D mesh, we introduce a particle shape (Hockney & Eastwood 1988). If $S(x)$ is the density at distance x from the particle and Δx is the cell size, then the density at distance (x, y, z) is a product $S(x)S(y)S(z)$. Two choices for S are used - Cloud In Cell (CIC) and Triangular Shaped Cloud (TSC):

$$CIC: S(x) = \frac{1}{\Delta x}\begin{cases}1, & |x| < \Delta x/2\\ 0, & \text{otherwise}\end{cases} \tag{17}$$

$$TSC: S(x) = \frac{1}{\Delta x}\begin{cases}1 - |x|/\Delta x, & |x| < \Delta x\\ 0, & \text{otherwise}\end{cases} \tag{18}$$

The fraction of particle mass assigned to a cell is just a product of three weight functions $w(x)w(y)w(z)$, where $\mathbf{r} = \mathbf{r_p} - \mathbf{x_i}$ is the distance between particles with coordinates $\mathbf{x_p}$ and cell center $\mathbf{x_i}$, and the weight function is $w(x) = \int_{x_i-\Delta/2}^{x_i+\Delta/2} S(x_p - x')dx'$:

$$CIC: w(x) = \begin{cases}1 - |x|/\Delta x, & |x| < \Delta x\\ 0, & \text{otherwise}\end{cases}, \tag{19}$$

$$TSC: w(x) = \begin{cases}\frac{3}{4} - |x|^2/\Delta x^2, & |x| < \Delta x/2\\ \frac{1}{2}\left(\frac{3}{2} - |x|/\Delta x\right)^2, & \Delta x/2 < |x| < 3\Delta x/2\\ 0, & \text{otherwise}\end{cases} \tag{20}$$

Although these relations eqs.(17–20) look somewhat complicated, in reality they require very few operations in a code. For the CIC scheme a particle contributes to the 8 nearest cells. If coordinates are scaled to be from 0 to $N_{\rm grid}$, where $N_{\rm grid}$ is the size of the grid in each direction, then taking an integer part of each coordinate of particle with center (x, y, z) (in Fortran: $i = INT(x)$...) gives the lower

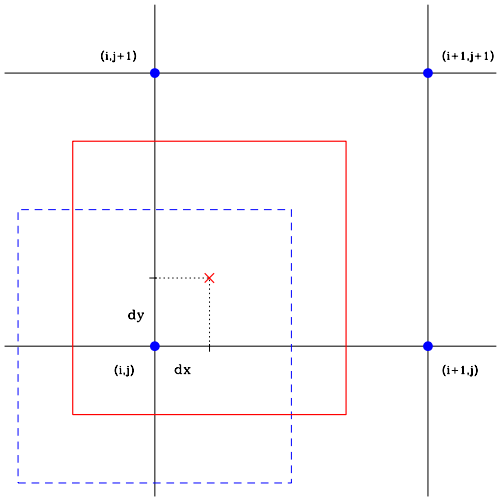

Figure 4. Example of the Cloud-In-Cell density assignment in two dimensions. Centers of mesh cells are shown with large blue circles. Blue dashed square presents boundaries of the cell with coordinates (i, j). Particle center shown with red cross has coordinates (dx, dy) and its boundaries are shown as red box. Area of intersection of the red and blue boxes is the mass that the particle contributes to the cell (i, j). All four cells indicated in the plot receive a contribution from the particle.

bottom grid cell (i, j, k). See Figure 4 for an example in 2D. Then the distance of the particle from that cell center is $dx = x - i, dy = y - j, dz = z - k$. The contributions of the particle to density ρ are:

$$\begin{cases} \rho_{i,j,k} & = & \rho_{i,j,k} + (1-dx)(1-dy)(1-dz) \\ \rho_{i+1,j,k} & = & \rho_{i+1,j,k} + dx(1-dy)(1-dz) \\ & \dots & \\ \rho_{i+1,j+1,k+1} & = & \rho_{i+1,j+1,k+1} + dxdydz \end{cases} \quad (21)$$

Having the density field $\rho_{i,j,k}$, we can estimate the gravitational potential by solving the Poisson equation.

To make the algorithm more transparent, we write the Poisson equation as $\nabla^2\phi = 4\pi G\rho$. We select the computational volume to be a cube of size L^3, which is periodically replicated to mimic the Universe. Coordinates of particles are in the limits $0 - L$: if a particle happens to move over a boundary of the cube, it appears on the other size the cube. The computational domain is covered by a cubic mesh of size $N^3_{\rm grid}$.The mesh is used to store the density field $\rho_{i,j,k}$. The algorithm can be written such a way that the same mesh is used to store the gravitational potential $\phi_{i,j,k}$. No additional storage is required.

We start with applying a 3D Fast Fourier Transformation (FFT) to the density field. That gives us Fourier components on a grid of the same size as the density field $\tilde{\rho}_{\mathbf{k}}$, where $\mathbf{k}$ is a vector with integer components in the range $0, 1, \dots, N_{\rm grid} - 1$. Now we multiply harmonics $\tilde{\rho}_{i,j,k}$ by the Green functions $G(\mathbf{k})$ to obtain amplitudes of Fourier harmonics of the gravitational potential ϕ:

$$\tilde{\phi}_{\mathbf{k}} = 4\pi G\tilde{\rho}_{\mathbf{k}} G(\mathbf{k}), \quad (22)$$

and then do the inverse FFT to find the gravitational potential $\phi_{i,j,k}$. Note that all these operations can be organized in such a way that only one 3D mesh is used.

The simplest, but not the best, method to derive the Green functions is to consider $\phi_{i,j,k}$ and $\rho_{i,j,k}$ as amplitudes of the Fourier components of the gravitational potential in the computational volume and then to differentiate the Fourier harmonics analytically. This gives

$$G_0(\mathbf{k}) = -\frac{\mathbf{1}}{\mathbf{k_x^2 + k_y^2 + k_z^2}} = -\left(\frac{\mathbf{L}}{\mathbf{2\pi}}\right)^{\mathbf{2}} \frac{\mathbf{1}}{\mathbf{i^2 + j^2 + k^2}}, \quad (23)$$

where $(k_x, k_y, k_z) = (2\pi/L)(i, j, k)$ are components of the wave-vector in physical units. A better way of solving the Poisson equation is to start with the finite-difference approximation of the Laplacian ∇^2. Here we use a the second order Taylor expansion for the spacial derivatives:

$$\begin{aligned} \nabla^2\phi &= \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} \\ &\approx [\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k} \\ &+ \phi_{i,j+1,k} - 2\phi_{i,j,k} + \phi_{i,j-1,k} \\ &+ \phi_{,j,k+1} - 2\phi_{i,j,k} + \phi_{i,j,k-1}]/\Delta x^2. \end{aligned} \quad (24)$$

This approximation leads to a large system of linear algebraic equations: $A\phi = 4\pi G\rho$, where ρ is the vector on the right hand side, ϕ is the solution, and A is the matrix of coefficients. All of its diagonal components are equal to -6, and all 6 nearest off-diagonal components are 1. The solution of this matrix equation can be found by applying the Fourier Transformation. This provides another approximation for the Green functions:

$$G_1(\mathbf{k}) = \frac{\mathbf{\Delta x^2}}{\mathbf{2}} \times \left[\cos\left(\frac{2\pi i}{N_{\rm grid}}\right) + \cos\left(\frac{2\pi j}{N_{\rm grid}}\right) + \cos\left(\frac{2\pi k}{N_{\rm grid}}\right) - 3\right]^{-1}. \quad (25)$$

For small (i, j, k) eq.(26) gives the same results as eq.(22). However, at (i, j, k) close to $N_{\rm grid}$ the finite-difference scheme G_1 provides less suppression for high-frequency harmonics and thus gives a stronger and more accurate force at distances close to the grid spacing Δx. Hockney & Eastwood (1988) argue that this happens because the finite-difference approximation partially compensates dumping of short waves related with the density assignment.

The computer memory puts constraints on the PM method because the method requires a large 3-dimensional mesh of size $N^3_{\rm grid}$ while the force resolution increases only as the first power of $N_{\rm grid}$: $\Delta x = L/N_{\rm grid}$, where L is length of the computational box. As we start to increase the resolution, we quickly run of the computer memory.

6 ADAPTIVE MESH REFINEMENT CODES

We can improve the PM method by increasing the resolution only where it is needed: by placing additional small-size elements – cubic cells – only in regions where there are many particles and where the resolution should be larger. Codes that use this idea are called the Adaptive Mesh Refinement (AMR) codes because they recursively increase the resolution constructing a hierarchy of cubic cells with smaller

and smaller elements in dense regions while keeping only large cells in regions that do not require high resolution. There are two ways of doing this: by splitting every element of the mesh, that has many particles, into 8 twice smaller boxes (Khokhlov 1998) or by placing a new rectangular block of cells to cover the whole high density region (Berger & Colella 1989). ART (Kravtsov et al. 1997; Gottloeber & Klypin 2008) and RAMSES (Teyssier 2002) codes use the first method while ENZO (Bryan et al. 1995) uses the second. Here we will mostly concentrate our attention to the method of Khokhlov (1998), which is frequently used in cosmological N-body simulations. Here we mostly follow algorithm and presentation of Kravtsov et al. (1997).

Cells are treated as individual units which are organized in refinement trees. Each tree has a root a cell belonging to a base cubic grid that covers the entire computational volume. If the root is refined (split) it has eight children (smaller nonoverlapping cubic cells residing in its volume), which can be refined in their turn, and so on. Cells of a given refinement level are organized in linked lists and form a refinement mesh. The tree data structures make mesh storage and access in memory logical and simple, while linked lists allow for efficient mesh structure traversals.

The tree ends with unsplit cells, which are called leaves. This structure is called an octal rooted tree, and is the construct used in TREE codes. We use fully threaded trees, in which cells are connected with each other on all levels. In addition, cells that belong to different trees are connected to each other across tree boundaries. All cells can be considered as belonging to a single threaded tree with a root being the entire computational domain and the base grid being one of the tree levels. The tree structure is supported through a set of pointers. Each cell has a pointer to its parent and a pointer to its first child. In addition, cells have pointers to the six adjacent cells (these make the tree fully threaded) so that information about a cells neighbors is easily accessible.

An elementary refinement process creates eight new cubic cells of equal volume (children) inside a parent cell. When the parent is refined, it is checked if all six neighbors are of the same level as the parent. If there are coarser neighbors (of smaller level than the parent), those neighbors are also split. If a neighbor in its turn has coarser neighbors, the neighbors neighbors are also split, and so forth. We thus build a refinement structure that obeys a rule allowing no neighbor cells with level difference greater than 1.

Once the refinement structure is build, we can solve the Poisson equation. On the zero (lowest) level all the volume is covered with a constant-size grid, and the Poisson equation is solved with the FFT method described in Section 5. The zero-level solution is used on the first refinement level either as an initial guess for the potential of a split cell or as a boundary condition for cells that are not split. After the Poisson equation is solved, the process repeats on the next level.

On each non-zero-level the Poisson equation is solved using iterative relaxation method (Hockney & Eastwood 1988; Kravtsov et al. 1997). We write the Poisson equation

$$\nabla^2\phi = \rho \tag{26}$$

as a diffusion equation

$$\frac{\partial\phi}{\partial\tau} = \nabla^2\phi - \rho. \tag{27}$$

As the fictitious time τ increases, the initial guess for ϕ approaches (relaxes to) an equilibrium solution, which is the solution of eq.(26). The finite-difference form of eq.(27) is:

$$\phi^{n+1}_{i,j,k} = \phi^n_{i,j,k} + \frac{\Delta\tau}{\Delta^2 x}\left(\sum_{l=1}^{6}\phi^n_l - 6\phi^n_{i,j,k}\right) - \rho_{i,j,k}\Delta\tau, \tag{28}$$

where the summation is over cell's 6 neighbors, Δx is the cell-size at the current level, and $\Delta\tau$ is fictitious time-step. For stability reasons $\Delta\tau \leqslant \Delta^2 x/6$. By selecting the maximum allowed time-step, we write the iteration scheme as:

$$\phi^{n+1}_{i,j,k} = \frac{1}{6}\sum_{l=1}^{6}\phi^n_l - \frac{\Delta^2 x}{6}\rho_{i,j,k}. \tag{29}$$

The convergence of the relaxation method can be improved in two ways. First, we split all the cells into "black" and "red" such that every "black" cell has only "red" neighbors and vice versa. (One can think about a 3d chess board). Each iteration is split into two phases: find and replace ϕ only for all "red" cells, and then only for "black" ones. Second, one can use successive overrelaxation (SOR) technique (Hockney & Eastwood 1988).

7 TREE AND TREE-PM CODES

TREE codes Appel (1985); Barnes & Hut (1986); Stadel (2001); Salmon & Warren (1994); Springel et al. (2001) use different ideas to estimate the force of gravity. Instead of solving the Poisson equation on a mesh as PM and AMR codes do, the TREE codes split particles into groups of different size and replace the force from individual particles in the group with a single multipole force of the whole group. The larger is the distance from a particle, the bigger is the allowed size of the particle group. Modern variants of the TREE algorithm are typically hybrid codes with the long-range force treated by a PM algorithm and the short-range handled by a TREE code (Xu 1995; Bagla 2002; Springel 2005; Warren 2013; Habib et al. 2014). Thus, there are four components of a TREE code: (1) Grouping algorithm, (2) Multipole expansion, (3) Condition for selecting size of the group (opening angle condition), and (4) Splitting the long- and short-distance forces.

Grouping. The oct tree algorithm is typically used in many TREE codes Springel (2005); Salmon & Warren (1994); Warren (2013). If the number of particles in a cell exceeds a specified threshold, it is split into 8 small cubic cells. Example of the oct tree is shown in Figure 5. Binary KD trees were used by Stadel (2001). In this method boundaries of rectangular cells are defined by the position of medians of coordinates of particles along each alternating direction. In some cases cells are quite elongated in KD tree algorithm. This can be mitigated by modifying the grouping algorithm. Gafton & Rosswog (2011) proposed the Recursive Coordinate Bisection (RCB) algorithm that splits cells at the center of mass with the direction of the bisecting plane being perpendicular to the direction of the maximum cell size. Indeed, Figure fig:TREEgroup indicates that cells are less elongated in the case of the RCB algorithm, which is used by Habib et al. (2014).

Tree is truncated once a cell reaches a specified minimum number of particles. In this case the cell called a

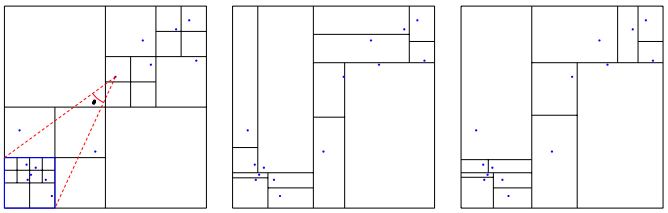

Figure 5. Examples of particle grouping algorithms for TREE codes. *Left panel:* the oct tree for 14 particles presented by blue circles. If the number of particles in a cell exceeds a specified threshold (in this case one particle), it is split into 8 small cubic cells (4 cells in 2D). Red dashed lines show opening angle θ for a particles close to the centre and for a cell indicated by a thick blue square. *Middle panel:* binary KD tree for the same set of particles. Boundaries of rectangular cells are defined by position of medians along each alternating direction. In some cases cells are quite elongated. *Right panel:* Recursive Coordinate Bisection tree. Cells are split at the center of mass with the direction of the bisecting plane being perpendicular to the direction of the maximum cell size. Cells are less elongated than in the case of KD trees.

leaf. The number of particles in a leaf can be as low as one. However, it can be significantly larger (Wadsley et al. 2004; Habib et al. 2014). If a leaf has more than one particle, then forces between particles in the cell are estimated using pair-wise summation. This can be faster than building more levels of the TREE hierarchy.

Multipole expansion. A number of physical quantities is collected for each cell that are used for force estimates. GADGET-2 code (Springel 2005) stores the mass and the center of mass of all particles in a given cell. Multipole expansion up to hexadecapole is used in PKDGRAV (Stadel 2001). Quadrupole expansion was used by Springel et al. (2001) and Dubinski et al. (2004). There is no rule what order of expansion to select. Low orders are faster to calculate and less memory is needed to store the information. At the same time, higher orders of expansion may allow one to use larger opening angles resulting in faster overal calculations. Grouping algorithm may also affect the selection of the expansion. The bisection trees can produce elongated cells implying that a higher order of mass expansion may be needed to maintain force accuracy.

Cell opening condition. Once the TREE is constructed and all information regarding mass distribution in each cell is stored, we start to find the forces by looping through all leaves and for each leaf by walking along the TREE down from the largest cells. Each cell of size l is tested whether angle $\theta \approx l/d$ at which it is seen by particles in the leaf at distance d is too large. If θ exceeds a specified threshold, the force contributions are not taken from the cell itself. We "open" the cell meaning that we descend to children of the cell and test them regarding their opening angles. Once the opening angle is small enough, the force contribution from the cell is accepted, and the algorithm proceeds to the next top-level cell.

Particular implementation of the cell-opening condition changes from code to code. In GADGET-2 the force is accepted if

$$\theta = \frac{l}{d} \leqslant \sqrt{\alpha g/[GM/d^2]}, \tag{30}$$

where g is the particle acceleration from the previous time-step, d is the distance from the particle to the cell of mass M and linear size l. Here α is a tolerance parameter defining the error of the force. There is an additional condition that each coordinate distance of the particle and geometrical cell center should be small:

$$|d_i - c_i| \leqslant 0.6l, i = 1, 2, 3 \tag{31}$$

where d_i and c_i are coordinates of the particle and the cell center. GASOLINE (Wadsley et al. 2004) uses opening condition:

$$\theta = \frac{2B_{\max}}{\sqrt{3}d} \leqslant \alpha, \tag{32}$$

where $B_{\max}$ is the maximum distance between the cell center of mass and a particle in the cell, d is the distance from the particle, for which the force is estimated, to the cell center of mass, and α is the tolerance parameter.

Splitting the long- and short-distance forces. In order to advance particles from one moment of time to another we must estimate the total force of gravity acting on each particle. We can split the total force into a smoothly varying part handled by the PM method and a short range force estimated by the TREE code. For example, we imagine that a point-size particle – a delta function in space – with mass m and position $\mathbf{r_i}$ is split into two components: (1) a sphere S of constant density and radius r_s , and (2) the point mass m minus the sphere with mass m. Schematically we can write the total density as

$$\begin{aligned} \rho(\mathbf{r}) &= S(\mathbf{r} - \mathbf{r_i}; \mathbf{r_s}, \mathbf{m}) + [\mathbf{m}\delta(\mathbf{r} - \mathbf{r_i}) - \mathbf{S}(\mathbf{r} - \mathbf{r_i}, \mathbf{r_s}, \mathbf{m})] \\ &\equiv \rho^{PM} + \rho^{\mathrm{TREE}}. \end{aligned} \tag{33}$$

If we open the brackets and collect all the terms, we get just the original point mass. Note that the second term in the r.h.s, ρ^{TREE}, has the total mass equal to zero. So, it does not produce a force at distances $r > r_s$. It can be estimated by a TREE algorithm which is simplified in this case by the fact that we should not look for force contributions from particles and cells which distance is larger than r_s. In other words, the *walk* over the tree includes only a local search. This dramatically speeds up the TREE part of the code. The first term ρ^{PM} represents the smooth component of the density distribution and can be efficiently handled by the PM algorithm. This splitting algorithm also simplifies the situation with periodical boundary conditions, which is a complication for pure TREE codes.

In practice, the sphere S may not have a constant density, and the point mass should be replaced by softened density profile. Hockney & Eastwood (1988) present details of force splitting used in historically important P^3M code. Here we follow the prescription for the for splitting in GADGET-2. Another example of force splitting is given by Habib et al. (2014). The gravitational potential ϕ in GADGET-2 is split in Fourier space into two components $\phi_k = \phi_k^{PM} + \phi_k^{TREE}$, where the long-distance part ϕ_k^{PM} is obtained with the PM code that has additional filter r_s:

$$\phi_k^{PM} = \phi_k \exp(-k^2 r_s^2). \tag{34}$$

The scale of the filter is larger by the factor 1–3 than the PM cell-size. The short range part of the gravitational potential

is estimated in the real space:

$$\phi_k^{\rm TREE}(\mathbf{x}) = -\mathbf{G}\sum_{\mathbf{i}} \frac{\mathbf{m_i}}{|\mathbf{x}-\mathbf{r_i}|}{\rm erfc}\left(\frac{|\mathbf{x}-\mathbf{r_i}|}{\mathbf{2r_s}}\right), \qquad (35)$$

where the summation is taken over all particles and cells that can contribute to the short-range force at $\mathbf{x}$.

8 EVOLUTION OF THE DARK MATTER DENSITY AND THE POWER SPECTRUM

We review some results of N-body simulations. It is nearly impossible to even mention all important results and to cite all relevant publications – there are too many of them. The goal is to present the main qualitative results and trends of few basic properties of the distribution and evolution of the dark matter: the density distribution function and the power spectrum. Results presented below have been known before. They are reproduced using the publicly available MultiDark and Bolshoi simulations (Riebe et al. 2013; Klypin et al. 2011, 2016) and simulations done usingthe PM code of (Klypin & Prada 2016) All simulations use the Planck cosmological parameters (Planck Collaboration et al. 2013).

8.1 Dark matter density

At very high resdhsifts and on very large scales fluctuations grow close to the predictions of the linear theory. As the amplitude of fluctuations increases, they enter the regime of non-linear evolution. The transition to the non-linear regime is complicated and can be roughly split into two stages: weakly and strongly non-linear.

We start with the evolution of the probability density function (PDF), which tells us what fraction of the volume is occupied by regions with a given density. We randomly place in space a cube of size Δx and measure the mass M inside it and its average density: $\rho = M/\Delta x^3$. What is the probability $p(\rho)d\rho$ that the volume element has density ρ? This quantity has a long history in cosmology (Coles & Jones 1991; Kofman et al. 1994; Lam & Sheth 2008). We will discuss PDF for dark matter, but instead we also could analyze, for example the distribution function of galaxies. That leads us to the statistics called cell counts: how many cells have N objects (Hubble 1936; White 1979; Sheth et al. 1994; Marinoni et al. 2005).

In the linear regime the distribution function is a Gaussian:

$$p_{\rm lin}(\rho) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\delta^2}{2\sigma^2}\right), \qquad (36)$$

where $\delta \equiv \rho/\rho_m - 1$ is the density contrast, ρ_m is the average density, and σ^2 is the dispersion of δ. As the fluctuations grow, they become nonlinear, and $p(\rho)$ starts to show deviations from the Gaussian distribution. Figure 6 shows the evolution of PDF as measured with cells of size $\Delta x = 2.2h^{-1}$Mpc . At redshift $z = 20$ the fluctuations are almost in the linear regime and $p(\rho)$ is well approximated by the Gaussion distribution eq. (36). Still, the fit is far from perfect. For example, the peak of $p(\rho)$ is at $\rho < \rho_m$, and there is an excess of cells with large densities. This happens because the fluctuations just start to deviate from the linear growth.

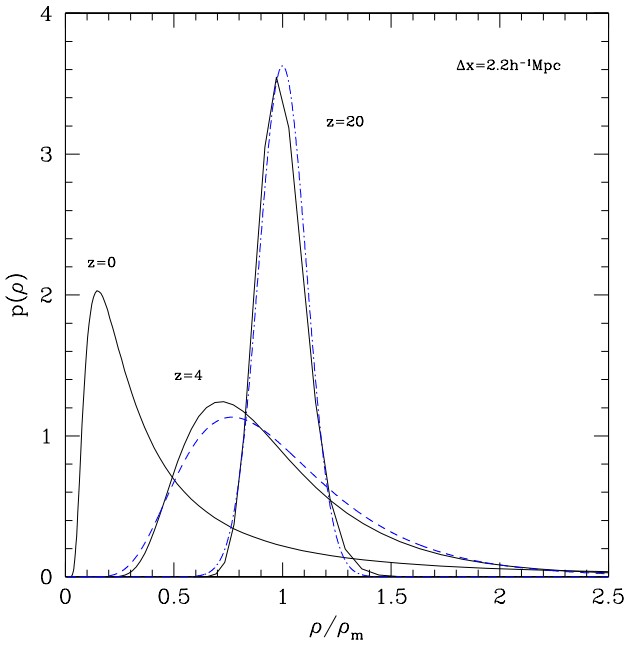

Figure 6. Evolution of the density distribution function with the redshift. The PDF was estimated using cells with size $2.2h^{-1}$Mpc. At $z = 20$ the fluctuations at this scale are still almost in the linear regime with the Gaussian distribution (dot-dashed curve) providing a good fit. By $z = 4$ stronger non-linear effects result in a very skewed distribution with the maximum of $p(\rho)$ shifting to low density and significant enhansment at large densities. At this stage of evolution $p(\rho)$ can be approximated by the log-normal distribution (dashed curve). However, it starts to badly fail for later states of evolution.

By $z = 4$ the non-linearities become stronger, which is clearly demonstrated by a very skewed shape of the PDF: the maximum has shifted to even lower densities, and more mass migrated to larger densities. At this weakly non-linear regime the PDF can be approximated by the log-normal distribution (Coles & Jones 1991; Lam & Sheth 2008):

$$p_{\log}(\rho) = \frac{1}{\sqrt{2\pi\sigma^2}}\left(\frac{\rho}{\rho_m}\right)^{-1}\exp\left(-\frac{[\ln(\rho/\rho_m)+\sigma^2/2]^2}{2\sigma^2}\right), \qquad (37)$$

When the fluctuations become strongly nonlinear, the PDF develops a very long tail at large densities while its maximum shifts to even lower densities. Figure 7 shows $\rho p(\rho)$ at different moments. Here we use a small cell size of $\Delta x = 1.1h^{-1}$Mpc that also allows us to probe fluctuations with larger densities. At $z = 4$ the log-normal distribution still provides a fit for data around the maximum of PDF, but it fails in the wings. At $z = 0$ the log-normal distribution becomes nearly useless: it fails practically everywhere.

At this strongly non-linear regime the distribution function $p(\rho)$ develops a nearly power-law shape with an exponential decline:

$$p(\rho) \propto \rho^{-2}\exp(-\alpha\rho^{3/4}), \quad \rho > 10\rho_m. \qquad (38)$$

Figure 7 shows that this provides at very good approximation to the data.

Figure 8 shows the evolution of the dark matter power spectrum and demonstrates the three regimes of growth of

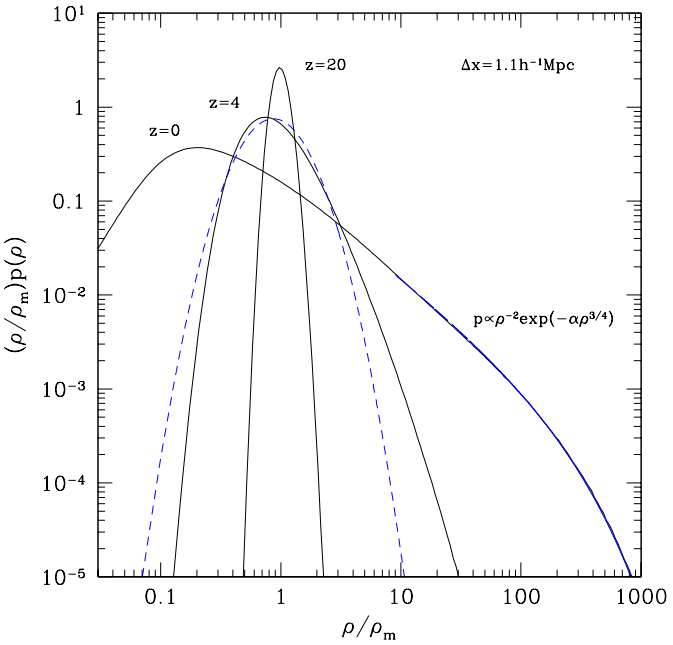

Figure 7. Detailed view of the density distribution function. PDF is scaled with the density and plotted on the logarithmic scale. A small cell of $1.1h^{-1}$Mpc is used. At $z = 4$ the log-normal distribution substantially deviates from the data at both low and large densities, but gives a sensible fit close to the maximum. At $z = 0$ the distribution function is so asymmetric that it cannot be even remotly approximated by the log-normal distribution. At large densities $p(\rho)$ is accurately approximated by a power law with an exponential decline.

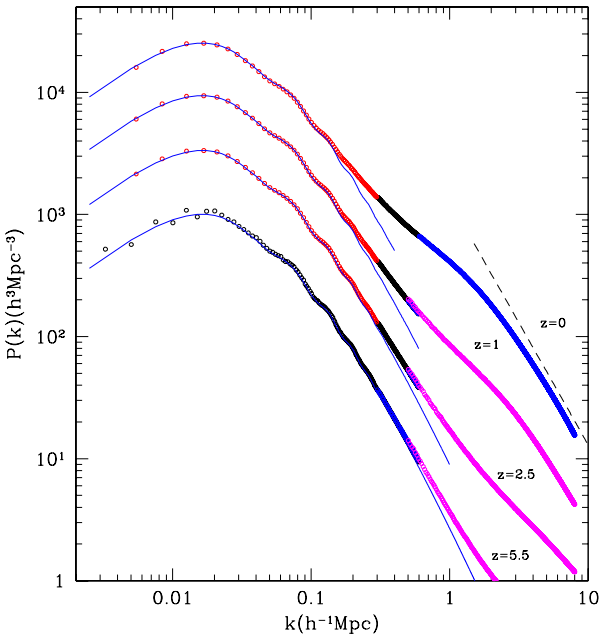

Figure 8. Evolution of the dark matter power spectrum. Simulations with different computational boxes and resolutions are shown by circles with different colors. Blue curves show predictions of the linear theory. The plot shows that at any redshift the power spectrum $P(k)$ has three regimes of growth: (1) linear growth on very long waves (small $k \lesssim 0.1\,h\mathrm{Mpc}^{-1}$) followed on larger wave-numbers by (2) the weakly non-linear regime where fluctuations grow much faster than predictions of the linear theory, and (3) strongly non-linear evolution at $k \gtrsim 1\,h\mathrm{Mpc}^{-1}$. In this regime the power spectrum gradually approaches power-law $P(k) \propto k^{-2}$ shown as the dashed line in the plot.

fluctuations (Peacock & Dodds 1994, 1996; Smith et al. 2003):

(1) On large scales (small k) fluctuations grow according to the predictions of the linear theory. Here the shape of the power spectrum $P(k, z)$ does not change, but its amplitude increases with time: $P(k, z) = D^2(z)P_{\rm lin}(k)$, where $D(z)$ is the linear growth factor normalized to be unity at present $D(z = 0) = 1$, and $P_{\rm lin}(k)$ is the linear power spectrum.

(2) On smaller scales (larger k) the fluctuations enter a weakly non-linear regime where the amplitude of fluctuations is still relatively small, but the fluctuations grow substantially faster than in the linear regime. The scale at which the fluctuations start to show a non-linear trend evolves with time. As time increases, the wavenumber of the transition k_{NL} becomes smaller and the amplitude $P(k_{NL})$ increases. The exact value of k_{NL} is somewhat arbitrary. If we choose the point at which $P(k)$ is, say 20% larger than the linear theory, than $k_{NL} \approx 0.2hMpc^{-1}$ at $z = 0$ and $k_{NL} \approx 1hMpc^{-1}$ at $z = 5.5$.

(3) On even smaller scales the fluctuations become strongly non-linear and enter regime of stable clustering. Contrary to naive expectations, the rate of the non-linear evolution is the fastest in the weakly non-linear regime. In strongly non-linear regime the dark matter is mostly in collapsed and nearly virialized halos. The halos still accrete mass and grow, but most of this mass stays in the outer halo regions. The inner regions the halos preserve their proper radius and mass (such the name "stable clustering"). Assuming that the number of pairs with a given proper separation r is preserved, the only effect, which is left, is the shrinking of halos in the comoving coordinates $x = r/a$. This leads to the increase in the power spectrum $P(k') \propto a^3$ and to the increase of the wavenumber $k' = ak$ (Davis & Peebles 1977; Smith et al. 2003). In the $P(k, z) - k$ plane, points start to drift to the right along k axis and upward along P axis as the fluctuations enter the stable clustering regime.

There are different ways to make analytical predictions for the non-linear evolution of the power spectrum. Hamilton et al.Hamilton et al. (1991) were the first to propose a physically motivated phenomenological model of mapping the linear correlation function $\xi_{\rm lin}(r) \propto a^2$ into the nonlinear function $\xi(r')$ by assuming a transition from the linear regime where $\xi = \xi_{\rm lin} \propto a^2$ and $r' = r$ to the regime of stable clustering where $\xi(r') = a\xi_{\rm lin} \propto a^3$ and $r^3 = (1 + \Delta^2_{NL})r'^3$, where Δ^2_{NL} is the non-linear estimate of the amplitude of fluctuations on the scale r'. Later Peacock & Dodds (1994, 1996); Smith et al. (2003) improved the model, which works reasonably well providing $\sim 10\%$ level of accuracy (Kravtsov & Klypin 1999; Heitmann et al. 2010). This may not be accurate enough for some tests. However, the model has an advantage that it provides a sensible approximation even at very large k were no other approximation works. It also gives qualitative explanation of the very non-linear regime.

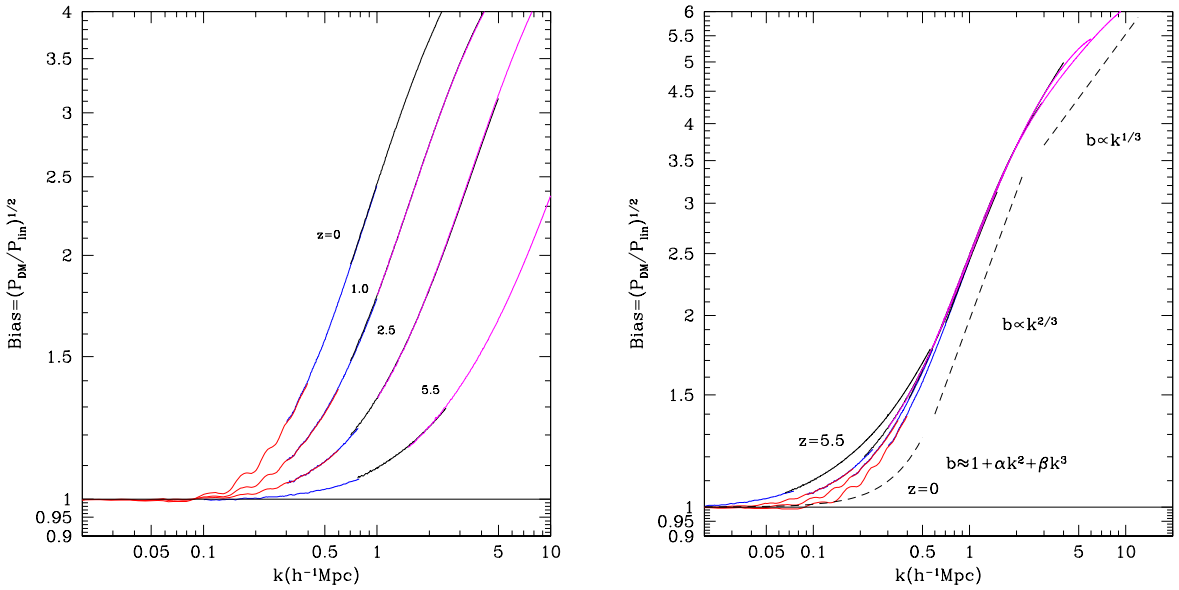

Figure 9. *Left panel:* Dependance of the bias parameter $b = \sqrt{P_{\rm DM}(k)/P_{\rm linear}(k)}$ on the wave-number and redshift. Wiggles at $k \sim (0.1-0.3)h^{-1}$Mpc are related with the smearing of the BAO oscillations by the non-linear interactions. *Right panel:* Bias parameter re-scaled to have the same k at $b = 2$. The plot illustrates self-similar growth of perturbations in the strongly non-linear regime $b \gtrsim 2$.

There is a different way of looking at the evolution of the power spectrum. Instead of plotting $P(k)$ at different redshifts, we can study the evolution of the ratio of the power spectrum P to the prediction of the linear theory $P_{\rm lin}$. This quantity is called the dark matter bias:

$$b^2(k;z) \equiv \frac{P(k;z)}{P_{\rm lin}(k;z)}. \qquad (39)$$

Left panel in Figure 9 shows the evolution of the bias $b(k;z)$ for the ΛCDM model with the Planck cosmological parameters. At any redshift there is a region at low wave-numbers k where $b \approx 1$. This is the domain of the linear growth of fluctuations. At larger k the bias factor starts to increase first as $k \approx 1 + \alpha(z)k^2$, where $\alpha(z)$ is a function of time. Then at larger k the bias factor deviates from this simple shape.

At $k \sim (0.1-0.3)h^{-1}$Mpc there are wiggles in the bias parameter that grow over time. Those wiggles are associated with the Baryonic Acoustics Oscillations (BAO, Eisenstein & Hu (1998); Eisenstein et al. (2005)). BAOs are related with the propagation of acoustic waves in the primordial plasma before the epoch of recombination $z \approx 1000$. During the recombination there is a sudden drop in the gas pressure and sound speed, which in turn effectively terminate the propagation of the acoustic waves. The characteristic scale of the acoustic horizon at the moment of recombination translates into a peak in the correlation function at $\sim 110h^{-1}$Mpc for the standart cosmology. (The exact value depends on cosmological parameters Ω_m, Ω_b, and h). The Fourier transform of the peak in the correlation function produces wiggles in the power spectrum of perturbations. As the perturbations enter non-linear regime, the peaks and throughs in $P_{\rm lin}(k)$ start to be smeared out. This is observed as appearance of wiggles in $b(k)$ in the weakly non-linear regime.

It is interesting to study the shape of the bias parameter at large wave-numbers. In order to clarify the situation we re-scale $b(k)$ functions at different redshifts to have the same value of k at $b \approx 2$. This is done by scaling k while keeping the bias parameter unchanged: $b(k\beta(z))$, where $\beta(z)$ is a factor that monotonically decreases with the redshift and $\beta(0) = 1$. The right panel in Figure 9 presents the rescaled bias parameter. It shows that the the bias parameter at large values $b \gtrsim 2$ evolves in a self-similar fashion: as fluctuations evolve the same shape of $b(k)$ simply shifts to smaller and smaller wave-numbers. In the limit of very large k the initial power spectrum scales as $P_{\rm lin}(k) \propto k^{-3}$ and in the strong non-linear regime $P(k) \propto k^{-2}$. So, the bias parameter should increase as $b \propto k^{1/2}$. At $k = (3-10)h^{-1}$Mpc (the largest k in Figure 9) the initial power spectrum is slightly shallower with $P_{\rm lin}(k) \propto k^{-2.6}$, which gives $b \propto k^{1/3}$, which is what we see in the simulations.

REFERENCES

Aarseth S. J., Turner E. L., Gott III J. R., 1979, ApJ, 228, 664

Appel A. W., 1985, SIAM Journal on Scientific and Statistical Computing, vol. 6, no. 1, January 1985, p. 85-103., 6, 85

Bagla J. S., 2002, Journal of Astrophysics and Astronomy, 23, 185

Barnes J., Hut P., 1986, Nature, 324, 446

Berger M. J., Colella P., 1989, Journal of Computational Physics, 82, 64

Binney J., Tremaine S., 2008, Galactic Dynamics: Second Edition. Princeton University Press

Bryan G. L. et al., 2014, ApJS, 211, 19

Bryan G. L., Norman M. L., Stone J. M., Cen R., Ostriker J. P., 1995, Computer Physics Communications, 89, 149

Coles P., Jones B., 1991, MNRAS, 248, 1
Couchman H. M. P., 1991, ApJL, 368, L23
Davis M., Efstathiou G., Frenk C. S., White S. D. M., 1985, ApJ, 292, 371
Davis M., Peebles P. J. E., 1977, ApJS, 34, 425
Dolag K., Borgani S., Schindler S., Diaferio A., Bykov A. M., 2008, Space Science Reviews, 134, 229
Dubinski J., Kim J., Park C., Humble R., 2004, NewA, 9, 111
Efstathiou G., Davis M., White S. D. M., Frenk C. S., 1985, ApJS, 57, 241
Eisenstein D. J., Hu W., 1998, ApJ, 496, 605
Eisenstein D. J. et al., 2005, ApJ, 633, 560
Feng Y., Chu M.-Y., Seljak U., 2016, ArXiv e-prints
Gafton E., Rosswog S., 2011, MNRAS, 418, 770
Gottloeber S., Klypin A., 2008, ArXiv e-prints
Gunn J. E., Gott III J. R., 1972, ApJ, 176, 1
Habib S. et al., 2014, ArXiv e-prints
Hamilton A. J. S., Kumar P., Lu E., Matthews A., 1991, ApJL, 374, L1
Heitmann K., White M., Wagner C., Habib S., Higdon D., 2010, ApJ, 715, 104
Hockney R. W., Eastwood J. W., 1988, Computer simulation using particles
Hubble E. P., 1936, Realm of the Nebulae
Izard A., Crocce M., Fosalba P., 2015, ArXiv e-prints
Khokhlov A., 1998, Journal of Computational Physics, 143, 519
Klypin A., Holtzman J., 1997, ArXiv Astrophysics e-prints
Klypin A., Yepes G., Gottlöber S., Prada F., Heß S., 2016, MNRAS, 457, 4340
Klypin A. A., Prada 2016, MNRAS, 0, 0
Klypin A. A., Shandarin S. F., 1983, MNRAS, 204, 891
Klypin A. A., Trujillo-Gomez S., Primack J., 2011, ApJ, 740, 102
Kofman L., Bertschinger E., Gelb J. M., Nusser A., Dekel A., 1994, ApJ, 420, 44
Kravtsov A. V., Klypin A. A., 1999, ApJ, 520, 437
Kravtsov A. V., Klypin A. A., Khokhlov A. M., 1997, ApJS, 111, 73
Lam T. Y., Sheth R. K., 2008, MNRAS, 386, 407
Marinoni C. et al., 2005, A&A, 442, 801
Peacock J. A., Dodds S. J., 1994, MNRAS, 267, 1020
Peacock J. A., Dodds S. J., 1996, MNRAS, 280, L19
Peebles P. J. E., 1970, AJ, 75, 13
Planck Collaboration et al., 2013, ArXiv e-prints
Quinn T., Katz N., Stadel J., Lake G., 1997, ArXiv Astrophysics e-prints
Riebe K. et al., 2013, Astronomische Nachrichten, 334, 691
Saha P., Tremaine S., 1992, AJ, 104, 1633
Salmon J. K., Warren M. S., 1994, Journal of Computational Physics, 111, 136
Sheth R. K., Mo H. J., Saslaw W. C., 1994, ApJ, 427, 562
Smith R. E. et al., 2003, MNRAS, 341, 1311
Springel V., 2005, MNRAS, 364, 1105
Springel V., Yoshida N., White S. D. M., 2001, NewA, 6, 79
Stadel J. G., 2001, PhD thesis, UNIVERSITY OF WASHINGTON
Teyssier R., 2002, A&A, 385, 337
Wadsley J. W., Stadel J., Quinn T., 2004, NewA, 9, 137
Warren M. S., 2013, ArXiv e-prints
White M., Tinker J. L., McBride C. K., 2014, MNRAS, 437, 2594
White S. D. M., 1976, MNRAS, 177, 717
White S. D. M., 1979, MNRAS, 186, 145
Xu G., 1995, ApJS, 98, 355
Zemp M., Stadel J., Moore B., Carollo C. M., 2007, MNRAS, 376, 273