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

Cosmoligical N-body code GLAM.

Anatoly Klypin[1⋆] and Francisco Prada[2]

[1] *Astronomy Department, New Mexico State University, Las Cruces, NM, USA*
[2] *Instituto de Astrofísica de Andalucía (CSIC), Glorieta de la Astronomía, E-18080 Granada, Spain*

14 September 2017

ABSTRACT

Here we provide description of the GLAM code: a parallel version of the Particle-Mesh cosmological code, that provides us with a tool to quickly generate a large number of N-body cosmological simulations with a reasonable speed and acceptable resolution. We call our code Parallel Particle-Mesh GLAM (PPM-GLAM), which is the core of the GLAM (GaLAxy Mocks) pipeline for massive production of galaxy catalogs. PPM-GLAM generates the density field, including peculiar velocities, for a particular cosmological model and initial conditions.

Key words: cosmology: Large scale structure - dark matter - galaxies: halos - methods: numerical

1 INTRODUCTION

Requirements for the generation of many thousands of high-quality simulations are extreme. Existing codes such as GADGET, RAMSES, or ART are powerful for large high-resolution simulations, but they are not fast enough for medium quality large-number of realisations required for analysis and interpretation of large galaxy surveys. New types of codes (e.g., White et al. 2014; Tassev et al. 2013; Feng et al. 2016) are being developed for this purpose. Here, we present the performance results of our N-body Parallel Particle-Mesh GLAM code (PPM-GLAM), which is the core of the GLAM (GaLAxy Mocks) pipeline for the massive production of large galaxy catalogs. PPM-GLAM generates the density field, including peculiar velocities, for a particular cosmological model and initial conditions.

There are a number of advantages of cosmological Particle-Mesh (PM) codes (Klypin & Shandarin 1983; Hockney & Eastwood 1988; Klypin & Holtzman 1997) that make them useful on their own to generating a large number of galaxy mocks (e.g., QPM, COLA, FastPM; White et al. 2014; Tassev et al. 2013; Feng et al. 2016), or as a component of more complex hybrid TREE-PM (e.g., Gadget2, HACC; Springel 2005; Habib et al. 2014) and Adaptive-Mesh-Refinement codes (e.g. ART, RAMSES, ENZO; Kravtsov et al. 1997; Teyssier 2002; Bryan et al. 2014). The main advantage of PM codes is that they are the fastest codes available and they can provide – after careful tunig of parameters – accurate enough simulations for many projects.

In Section 2 we discuss the main features of our PPM-GLAM simulation code and give estimates of the memory, time-stepping and cpu required to make simulations with the code. Performance of the GLAM code is described in Section 3 Main algorithms for density assignment, Poisson solving, and particles advance can be found in Section 4. Parallelization is discussed in Section 5. Section 6 provides tests for the effects of mass and force resolutions, and the effects of time-stepping.

2 PARALLEL PARTICLE-MESH GLAM CODE

Here, we discuss the main features of the PPM-GLAM code and provide the motivation for the selection of appropriate numerical parameters.

The code uses a regularly spaced three-dimensional mesh of size $N_{\rm g}^3$ that covers the cubic domain L^3 of a simulation box. The size of a cell $\Delta x = L/N_{\rm g}$ and the mass of each particle $m_{\rm p}$ define the force and mass resolution respectively:

$$m_{\rm p} = \Omega_m\,\rho_{\rm cr,0}\left[\frac{L}{N_{\rm p}}\right]^3 = \quad (1)$$

$$= 8.517\times10^{10}\left[\frac{\Omega_m}{0.30}\right]\left[\frac{L/h^{-1}{\rm Gpc}}{N_{\rm p}/1000}\right]^3 h^{-1}M_\odot, \quad (2)$$

$$\Delta x = \left[\frac{L/h^{-1}{\rm Gpc}}{N_{\rm g}/1000}\right]h^{-1}{\rm Mpc}, \quad (3)$$

where $N_{\rm p}^3$ is the number of particles and $\rho_{\rm cr,0}$ is the critical density of the universe at present.

PPM-GLAM solves the Poisson equation for the gravi-

⋆ E-mail: aklypin@nmsu.edu

tational potential in a periodical cube using a Fast Fourier Transformation (FFT) algorithm. The dark matter density field used in the Poisson equation is obtained with the Cloud-In-Cell (CIC) scheme using the positions of dark matter particles. Once the gravitational potential is obtained, it is numerically differentiated and interpolated to the position of each particle. Then, particle positions and velocities are advanced in time using the second order leap-frog scheme. The time-step is increased periodically as discussed in Appendix A. Thus, a standard PM code has three steps that are repeated many times until the system reached its final moment of evolution: (1) Obtain the density field on a 3D-mesh that covers the computational volume, (2) Solve the Poisson equation, and (3) Advance particles to the next moment of time.

The computational cost of a single PPM-GLAM simulation depends on the number of time-steps $N_{\rm s}$, the size of the 3D-mesh $N_{\rm g}^3$, and the adopted number of particles $N_{\rm p}^3$. The CPU required to solve the Poisson equation is mostly determined by the cost of performing a single 1D-FFT. We incorporate all numerical factors into one coefficient and write the CPU for the Poisson solver as $AN_{\rm g}^3$. The costs of density assignment and particle displacement (including potential differentiation) scale proportionally to $N_{\rm p}^3$. In total, the CPU time $T_{\rm tot}$ required for a single PPM-GLAM run is:

$$T_{\rm tot} = N_{\rm s}\left[A{N_{\rm g}}^3 + (B+C){N_{\rm p}}^3\right], \tag{4}$$

where B and C are the coefficients for scaling the CPU estimate for particle displacements and density assignment. These numerical factors were estimated for different processors currently used for N-body simulations and are given in Table 1. For a typical simulation with parameters $N_{\rm g} = 2400$, $N_{\rm p} = N_{\rm g}/2$ the CPU per time-step is ~ 0.5 hours and wall-clock time per step $\sim 1-3$ minutes. The total cost of 1000 PPM-GLAM realizations with $N_{\rm s} = 150$ is 75K CPU hours, which is a modest allocation even for a small computational cluster or a supercomputer center.

Memory is another critical factor that should be considered when selecting the parameters of our simulations. PPM-GLAM uses only one 3D-mesh for storing both density and gravitational potential, and only one set of particle coordinates and velocities. Thus, for single precision variables the total required memory $M_{\rm tot}$ is:

$$M_{\rm tot} = 4N_{\rm g}^3 + 24N_{\rm p}^3 \text{ Bytes}, \tag{5}$$

$$= 29.8\left(\frac{N_{\rm g}}{2000}\right)^3 + 22.3\left(\frac{N_{\rm p}}{1000}\right)^3 \text{GB}, \tag{6}$$

$$= 52\left(\frac{N_{\rm p}}{1000}\right)^3 \text{GB, for} \quad N_{\rm g} = 2N_{\rm p}. \tag{7}$$

The number of time-steps $N_{\rm s}$ is proportional to the computational costs of the simulations. This is why reducing the number of steps is important for producing a large set of realisations. White et al. (2014) and Koda et al. (2016) use just ~ 10 time-steps for their QPM and COLA simulations. Feng et al. (2016) and Izard et al. (2015) advocate using $N_{\rm s} \approx 40$ steps for Fast-PM and ICE-COLA. The question still remains: what optimal number of time-steps should be adopted? However, there is no answer to this question without specifying the required force resolution, and without specifying how the simulations will be used to generate mock galaxies.

Below we provide a detailed discussion on the effects of time-stepping. We argue that for the stability and accuracy of the integration of the dark matter particle trajectories inside dense (quasi-) virialised objects, such as clusters of galaxies, the time-step Δt must be smaller enough to satisfy the constraints given by eqs. (31) and (33). For example, FastPM simulations with 40 time-steps and force resolution of $\Delta x = 0.2\,h^{-1}$Mpc (see Feng et al. 2016) do not satisfy these conditions and would require 2-2.5 times more time-steps. However, a small number of time-steps manifests itself not in the power spectrum (though, some decline in $P(k)$ happens at $k \sim 1\,h^{-1}$Mpc). Its effect is mostly observed in a significantly reduced fraction of volume with large overdensities and random velocities, which potentially introduces undesirable scale-dependent bias.

Because our main goal is to produce simulations with the minimum corrections to the local density and peculiar velocities, we use $N_{\rm s} \approx 100-200$ time-steps in our PPM-GLAM simulations. This number of steps also removes the need to split particle displacements into quasi-linear ones and the deviations from quasi-linear predictions. Thus, in this way we greatly reduce the complexity of the code and increase its speed, while also substantially reduce the memory requirements.

Finally, we estimate the computational resources – CPU time and computer memory – required to run a large set of PPM-GLAM simulations for different combinations of box size L, accuracy of the power spectrum and mesh size $N_{\rm g}$. There are many factors that define the optimal selection of computational parameters, including the number of time-steps, the number of realisations, the effects of super-sample waves and the limitations on the available computer memory. The estimates for (i) the wave-number $k_{1\%}$ at witch the error in $P(k)$ reaches the level of 1%:

$$k_{1\%} = \frac{0.3}{\Delta x} = \frac{0.3N_{\rm g}}{L}, \tag{8}$$

where the box size L is given in units of h^{-1}Mpc. Lines of constant $k_{1\%}$ are shown as dotted (blue) lines in Figure 1. The larger the value of $k_{1\%}$, better is the performance reached at smaller scales. (ii) to estimate the number of time-steps $N_{\rm s}$ required for a given simulation, we assume that on average the particles should not move more than 1/3 of a cell. This includes fast-moving particles in clusters of galaxies. We assume that the rms 3D-velocity for the particles in galaxy clusters is $v \approx 2000\,{\rm km\,sec}^{-1}$. Estimating the number of steps as $a/\Delta a = N_{\rm s}$ and using eq. (33) with $\beta = 1/3$, we find that the number of time-steps is

$$N_{\rm s} \approx \frac{60N_{\rm g}}{L}, \tag{9}$$

where the box size is given in units of h^{-1}Mpc.

Thus, the total amount of CPU-hours required for producing N_r simulations with box size L and mesh size $N_{\rm g}$ is

$$t_{\rm tot} = N_r N_{\rm s} {N_{\rm g}}^3 t_1 = 2.4\times10^{-9} N_r N_g^4 L^{-1}, \tag{10}$$

where t_1 is the CPU-hours required per time-step. Here we use the timings provided in the first raw of Table 1. It is

Table 1. Timing of the PPM-GLAM code for different computational systems. The columns give: (1) number of particles N_p, (2) number of grid cells N_g, (3) processor type and number of cores, (4) the total wall-clock time per step in minutes, (5) wall-clock time for the Poisson solver in minutes, (6) advancing particles timing in minutes and (7) density assignment timing in minutes. Columns (8–10) give the parameters A, B, C for CPU time per cell and per particle in eq. (4) in units of 10^{-8}. Other columns provide: (11) CPU time per step in minutes, (12) CPU time per step per particle in 10^{-6} seconds, (13) CPU time in hours for a un with 150 time-steps.

(1) N_p	(2) N_g	(3) Processor cores	(4) Total min	(5) Poisson min	(6) Particles min	(7) Density min	(8) A	(9) B	(10) C	(11) CPU step	(12) CPU particle	(13) CPU run
1200^3	2400^3	Intel E5-2680v4 2.4GHz 2x14	1.20	1.02	0.07	0.10	12.4	6.8	9.8	33.6	1.17	84
1200^3	2400^3	Intel E5-2680v3 2.5GHz 2x12	1.44	1.22	0.08	0.14	15.0	6.7	11.7	34.5	1.20	86
500^3	1000^3	Intel E5-2680v4 2.4GHz 2x14	0.075	0.062	0.0039	0.0088	10.4	5.2	11.8	2.1	1.01	5.2
1000^3	2000^3	Intel E5-2680v4 2.4GHz 2x14	0.65	0.55	0.037	0.057	11.5	6.2	9.6	18.2	1.09	45.5
1300^3	2600^3	AMD 6174 2.4GHz 4x12	2.23	1.72	0.21	0.30	28.2	27.5	39.3	107	2.92	267
1600^3	3200^3	AMD 6376 2.3GHz 4x16	2.44	2.11	0.11	0.22	24.7	10.3	20.6	156	2.28	390

also convenient to use the total volume covered by all set of realisations, i.e. $V = N_r L^3$. Using the expressions for the total volume and CPU-hours, V and $t_{\rm tot}$, we can write a relation between the required box size L and grid $N_{\rm g}$ to run many simulations covering a volume V under the condition that the total CPU is $t_{\rm tot}$:

$$L = \left(\frac{2.4 \times 10^{-9} V}{t_{\rm tot}} \right)^{1/4} N_{\rm g}. \qquad (11)$$

In Figure 1 we plot lines of $L(N_{\rm g})$ for a somewhat arbitrary value of $V = 5000(h^{-1}{\rm Gpc})^3$ and for three different CPU times of $10^5, 10^6, 10^7$CPU-hours. Additional constraints are coming from the SSC modes that limit the box size, which we assume to be bigger than $L = 500\,h^{-1}$Mpc. The number of realisations should be large (thousands) regardless of the total covered volume. This tends to limit the box size to lower values, or, equivalently, to increase proportionally the CPU time. All theses limitations are shown in Figure 1. They indicate that currently the selection is limited to the parameters inside the oval drawn in the plot.

The results presented in this work demonstrate that the Parallel Particle-Mesh GLAM code, at a very low computational cost, generates the accurate dark matter clustering statistics required for the production of thousands of mock galaxy catalogs. The next step, which will be presented in a future paper, will be to develop a bias scheme that will take as input the accurate density field provided by PPM-GLAM and produce those large galaxy catalogs in the context of the upcoming and existing large-scale galaxy surveys.

3 PERFORMANCE OF THE GLAM CODE

The PPM-GLAM code was tested on a variety of processors (both Intel and AMD) and computer platforms. Results of code timing are given in Table 1 for different hardware configurations and parameters of the simulations. As might have been expected, the Intel processors are about twice faster than AMD when timing results are scaled to CPU-hours per core. However, this is somewhat deceiving because the AMD processors provide about twice more cores. If this is taken into account, then the difference between AMD and Intel processors becomes smaller. For example, a single computational node with four AMD-6376 processors has the the same performance as a node with two Intel E5-2680v4 processors when rescaled to the same computational task.

Column 12 in Table 1 provides the CPU time scaled per individual particle. In that respect it is a measure of the efficiency of the parallelisation and performance of our PM code. It shows that within $\sim 20\%$ the code scales well for different number particles and mesh sizes.

The computational cost of a PPM-GLAM simulation depends on the number of time-steps N_s, the size of the 3D-mesh $N_{\rm g}^3$, and the adopted number of particles $N_{\rm p}^3$. The CPU required to solve the Poisson equation is mostly determined by the cost of performing a single 1D-FFT. Thus, it is proportional to $(N_g \log N_{\rm g})^3$. There are some additional costs (e.g., two 3D-matrix transpositions to align the mesh with the memory of individual computational processors), but those are relatively small.

The required memory for a simulation is given by eq. (7). We could have used double precision accuracy for the coordinates, as adopted in FastPM by Feng et al. (2016), but our estimates show that the loss of coordinates accuracy at the edge of the simulation box are practically negligible. For example, for an extreme configuration of a $1000\,h^{-1}$Gpc simulation box with $N_{\rm g} = 3000$ mesh particles moving for 13 Gyrs with a constant drift velocity of $500\,{\rm km\,sec^{-1}}$, and with additional random velocity of $1000\,{\rm km\,sec^{-1}}$, will have an error of just $2 \times 10^{-4} h^{-1}$Mpc. This is very small uncertainty as compared with the simulation cell size of $0.33\,h^{-1}$Mpc.

While the CPU speed and RAM memory estimates are very favorable for a very large number of medium-resolution PPM-GLAM simulations, equations (4-7) clearly indicate that increasing either the resolution or losing resolution for some code parameter configurations can have serious repercussions. For example, increasing the force resolution $\Delta x = L/N_{\rm g}$ by a factor of two, increases the computational CPU cost eight times, i.e. a very large factor. Thus, the pa-

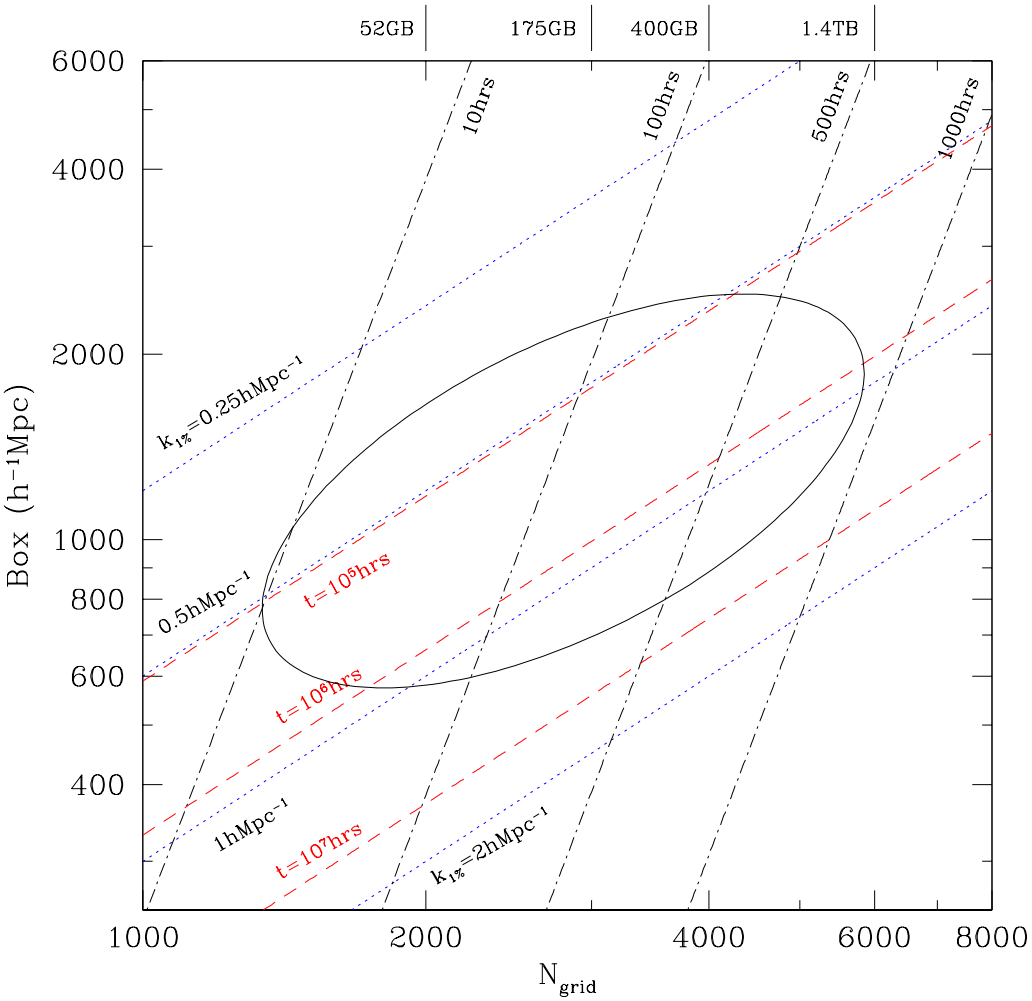

Figure 1. Dependance of the different numerical parameters of the PPM-GLAM simulations on box- and mesh-size. The vertical lines at the top-axis of the plot show the computer memory required for a simulation with mesh-size $N_{\rm grid}$ and number of particles $N_{\rm p} = N_{\rm grid}/2$. Dot-dashed lines correspond to the number of CPU-hours needed to make a single realisation with the given combination of mesh-size (and particles) and number of steps as defined by eq. (9). Diagonal dotted (blue) lines show constant values of $k_{1\%}$ (the wave-number at which the error in $P(k)$ reaches the level of 1%). In order to achieve better resolution than a selected value of $k_{1\%}$, the simulation parameters (box- and mesh-size) should be set to those values located below the corresponding $k_{1\%}$ dotted line. Dashed (red) lines are lines of constant CPU time (in hours) required to make a set of PPM-GLAM simulations with the cumulative volume of 5000 $(h^{-1}{\rm Gpc})^3$. In order to avoid large Super Sample Covariance (SSC) defects, the simulations should have large enough box-size $L \gtrsim 500\,h^{-1}$Mpc. The requirements to have a large number of realisations $N_{\rm r}$, for a given CPU time and accuracy, tend to reduce the simulation box-size. Overall, these different constraints tend to limit the selection of computational parameters to the oval area indicated in the plot.

rameters of the simulations should be selected very carefully. Loss of resolution may happen as a side-effect of a modification in algorithms that at first sight seems reasonable.

For example, the QPM code (White et al. 2014) uses the Green functions given in eq. (16) instead of the more advanced eq. (18) (Hockney & Eastwood 1988) adopted in PPM-GLAM. Our tests show that this change alone reduces the force resolution by about 20 percent, which seems like a small loss, but not for a cosmological PM code. In order to recover the loss, one would need to increase the CPU and memory by a factor 1.7. Because PM codes tend to run at the limit of available computer memory, this factor represents a serious disadvantage. One may also think of improving the PM code, for example, by increasing the order of the gravitational potential interpolation scheme (QPM; White et al. 2014) or by replacing numerical differentiation by obtaining acceleration in the Fourier-space (COLA; Tassev et al. 2013). Yet, higher-order schemes will effectively reduce the resolution, and when compared at the same resolution, these modifications only slow down the code without gaining numerical accuracy.

One may try to avoid numerical differentiation of the gravitational potential by solving the acceleration in Fourier-space, as done in the COLA and FastPM codes (Tassev et al. 2013; Feng et al. 2016). Potentially, that strategy could increase the resolution, but it is not clear whether

this procedure actually is beneficial[1]. However, the computational cost of such a modification is very substantial. It requires doubling the memory (additional 3D-mesh for accelerations) and also doubling the CPU time (3 inverse FFTs instead of just one in our PPM-GLAM code).

4 ALGORITHMS

In general, a cosmological PM code consists of three steps to evolve the particles: (1) Using the particle positions $\mathbf{r}$ to obtain the density $\rho_{i,j,k}$ at the nodes of an homogenous 3D-mesh that covers the computational domain, (2) Solve the Poisson equation on the mesh, and (3) advance the particles to a new moment of time.

4.1 Density field:

We start with the calculation of the density field produced by $N_{\rm p}^3$ particles on the $N_{\rm g}^3$ nodes of the mesh. In order to assign the particle density 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 the product $S(x)S(y)S(z)$. Two choices for S are adopted: Cloud-In-Cell (CIC) and Triangular Shaped Cloud (TSC). Here we will use the CIC scheme, i.e.

$$CIC: S(x) = \frac{1}{\Delta x}\begin{cases}1, & |x| < \Delta x/2\\ 0, & \text{otherwise}\end{cases}. \qquad (12)$$

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 $\mathbf{x_i}$. The weight function is $w(x) = \int_{x_i-\Delta/2}^{x_i+\Delta/2} S(x_p - x')dx'$:

$$CIC: w(x) = \begin{cases}1 - |x|/\Delta x, & |x| < \Delta x\\ 0, & \text{otherwise}\end{cases}. \qquad (13)$$

Although these relations given in eqs.(12–13) look somewhat complicated, in reality they require very few operations in the code. For the CIC scheme a particle contributes to the 8 nearest cells. If the coordinates are scaled to be from 0 to $N_{\rm g}$, where $N_{\rm g}$ is the size of the grid in each direction, then taking an integer part of each particle coordinate with center (x, y, z) - in Fortran: $i = INT(x)$... - gives the lower bottom grid cell (i, j, k). Then, the distance of the particle from that cell center is $dx = x - i, dy = y - j, dz = z - k$.

4.2 Gravitational potential

Having the density field $\rho_{i,j,k}$, we can estimate the gravitational potential by solving the Poisson equation, which for clarity we simply write as

$$\nabla^2\phi = 4\pi G\rho(\mathbf{x}). \qquad (14)$$

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

$$\tilde{\phi}_{i,j,k} = 4\pi G\tilde{\rho}_{i,j,k}G(\mathbf{k}), \qquad (15)$$

and then do the inverse FFT to find out 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 – no additional RAM memory is required.

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 decomposition of the gravitational potential in the computational volume, and then to differentiate the Fourier harmonics analytically. This gives

$$G_0(\mathbf{k}) = -\frac{\mathbf{1}}{\mathbf{k_x^2 + k_y^2 + k_z^2}} = -\left(\frac{\mathbf{L}}{\mathbf{2\pi}}\right)^{\mathbf{2}}\frac{\mathbf{1}}{\mathbf{i^2 + j^2 + k^2}}. \qquad (16)$$

A better way of solving the Poisson equation that we use in our PPM-GLAM code 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:

$$\begin{aligned}\nabla^2\phi &= \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}\\ &\approx [\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}\\ &+ \phi_{i,j+1,k} - 2\phi_{i,j,k} + \phi_{i,j-1,k}\\ &+ \phi_{,j,k+1} - 2\phi_{i,j,k} + \phi_{i,j,k-1}]/\Delta x^2.\end{aligned} \qquad (17)$$

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 the 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:

$$G_1(\mathbf{k}) = \frac{\mathbf{\Delta x^2}}{\mathbf{2}}\left[\cos\left(\frac{\mathbf{2\pi i}}{\mathbf{N_g}}\right) + \cos\left(\frac{\mathbf{2\pi j}}{\mathbf{N_g}}\right) + \cos\left(\frac{\mathbf{2\pi k}}{\mathbf{N_g}}\right) - \mathbf{3}\right]^{\mathbf{-1}}. \qquad (18)$$

For small (i, j, k), eq.(18) gives the same results as eq.(15). However, when (i, j, k) is close to $N_{\rm g}$, the finite-difference scheme G_1 provides less suppression for high-frequency harmonics and thus gives a stronger and more accurate force at distances closer to the grid spacing Δx. Hockney & Eastwood (1988) argue that this happens because the finite-difference approximation partially compensates the dumping of short waves that are related with the density assignment.

[1] We compare the errors in the power spectra at $k = 0.3h\text{Mpc}^{-1}$ shown in Figure 2 with the FastPM results Feng et al. (2016) (see their Figure 2). For simulations with the same force resolution of $0.34h^{-1}$Mpc, PPM-GLAM performs more accurately in spite of the fact that FastPM used Fourier-space to avoid the numerical differentiation of the gravitational potential.

4.3 Time-stepping

We write the particle equations of motions and the Poisson equation, using the particle momenta $\boldsymbol{p} \equiv a^2\dot{\boldsymbol{x}}$, as follows

$$\frac{d\mathbf{x}}{da} = \mathbf{u}, \qquad \mathbf{u} \equiv \frac{\mathbf{p}}{a^3 H}, \quad \boldsymbol{p} \equiv a^2\dot{\boldsymbol{x}}, \tag{19}$$

$$\frac{d\mathbf{p}}{da} = \mathbf{g}, \qquad \mathbf{g} \equiv -\frac{\nabla\phi}{aH}, \tag{20}$$

$$\nabla^2\phi = \frac{3}{2}\frac{H_0^2\Omega_0\delta_{\rm dm}}{a}, \tag{21}$$

$$H^2 = H_0^2\left(\frac{\Omega_0}{a^3} + \Omega_{\Lambda,0}\right), \quad \Omega_0 + \Omega_{\Lambda,0} = 1. \tag{22}$$

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

Because we start the simulations at a relatively high redshift $z_i \approx 100$, and because the number of time-steps $N_s \approx 100-200$ is not large, the time-stepping scheme should be carefully selected and tuned. In the sense of the accuracy of the time-stepping algorithms, there are two regimes: (1) when fluctuations grow nearly linearly at large redshifts and when the expansion factor a may change substantially over a single time-step, and (2) later moments when fluctuations are in the non-linear regime with a changing very little over a time-step. Both regimes present a challenge for accurate simulations with a small number of time-steps. There are different possibilities in handling these challenges.

At the linear stage of evolution the main concern and the main test is the linear growth of fluctuations. To address this problem COLA (Tassev et al. 2013, 2015) splits the changes in coordinates into a second-order perturbation term (estimated by a separate algorithm), and a residual, which is integrated using N-body methods. Instead, QPM (White et al. 2014) uses a logarithmic time-step (constant in $\Delta a/a$). COLA's time-stepping is a good idea (but very expensive) for the quasi-linear regime. However, at the very nonlinear stages of evolution, when the second order perturbation approximation is bound to be not valid, the splitting of the coordinate advances into two terms that cannot produce any benefits, and thus, it seems to be just a waste of CPU. At this stage a constant-step leap-frog scheme is preferred: it is time-symmetric, second-order accurate and hamiltonian preserving approximation.

Motivated by these considerations, we select a time-stepping scheme which uses a constant time-step at low redshifts $z < z_{\rm limit}$, but periodically increases the time-step at large redshifts $z > z_{\rm limit}$. The parameter $z_{\rm limit}$ defines the transition from early quisi-linear to late non-linear regimes. With a resolution of $\Delta x = (0.3 - 0.5)h^{-1}$Mpc in our simulations, some halos may start to collapse and virialise at $z < z_{\rm limit}$. This is the stage when we switch the time-stepping to the normal leap-frog scheme with a constant time-step. For our simulations we select $z_{\rm limit} = 3$.

(i) Early stages of evolution $z > z_{\rm limit}$. It is important to estimate how the terms $\boldsymbol{u}$ and $\boldsymbol{g}$, in the right-hand-sides of equations (19-20), evolve with the expansion parameter a at the linear regime. Because there are terms with large powers of a, one may be concerned with the accuracy of the integration of quickly evolving terms. However, when one considers all the terms, the situation is much less alarming. Indeed, in the linear regime the peculiar gravitational potential ϕ does not change with a, and along the particle trajectory $\mathbf{g}(a) \propto a^{1/2}$, leading to $\mathbf{p} \propto a^{3/2}$ and $\mathbf{u} \propto a^0$(constant). This means that there are no quickly evolving terms in the equations of motions. This slow evolution of the $\mathbf{u}$ and $\mathbf{g}$ terms allows one to periodically increase the time-step without substantial loss of accuracy. We do it by testing the magnitude of $\Delta a/a$. If this ratio falls below a specified value $(\Delta a/a)_{\rm min}$ (typically $3 - 5 \times 10^{-2}$), the time-step Δa is increased by factor 3/2.

We can write the time-stepping scheme using a sequence of kick K and drift D operators, which are defined as advances of particle momenta $\mathbf{p}$ and particle coordinates $\mathbf{x}$ from moment a to moment $a + \Delta a$:

$$K(\Delta a, a, \tilde{a}) \quad : \quad \mathbf{p}(a + \Delta a) = \mathbf{p}(a) + \mathbf{g}(\tilde{a})\Delta a, \tag{23}$$

$$D(\Delta a, a, \tilde{a}) \quad : \quad \mathbf{x}(a + \Delta a) = \mathbf{x}(a) + \mathbf{u}(\tilde{a})\Delta a, \tag{24}$$

where $\tilde{a}$ is the moment at which either $\mathbf{u}$ or $\mathbf{g}$ are estimated.

If we start with particle momenta at the time $a_{-1/2} = a_0 - \Delta a/2$ (a half time-step behind the coordinates defined at a_0), and use the notation $a_m = a_0 + m\Delta a$, the standard leap-frog scheme can be written as the following sequence of kick and drift operators:

$$K(\Delta a, a_{-1/2}, a_0)D(\Delta a, a_0, a_{1/2}) \tag{25}$$

$$K(\Delta a, a_{1/2}, a_1)D(\Delta a, a_1, a_{3/2}) \tag{26}$$

$$K(\Delta a, a_{3/2}, a_2)D(\Delta a, a_2, a_{5/2})\ldots \tag{27}$$

When at some moment a_0 we need to increase the time-step by factor 3/2, we do it by making a stronger kick and then by modifying the time-step to the new value of $\Delta a' = 3\Delta a/2$:

$$K(5\Delta a/4, a_{-1/2}, a_0)D(3\Delta a/2, a_0, a_{3/4})\ldots \tag{28}$$

After applying the first pair of kick-drift operands, the normal setup of the leap-frog scheme is restored with the particles momenta behind the coordinates by a half of the new time-step. The code continues the integration of the trajectories with a constant time-step until the moment when $\Delta a/a$ becomes smaller than the minimum value. The time-step is increased again by the factor 3/2, and the process continues.

The truncation error for the variable step scheme can be found similarly to the way how it is done for the standard leap-frog scheme by eliminating the velocities from the scheme, and then by expanding the coordinates in the Taylor series around moment a_0. This gives $x_{3/2} - (5/2)x_0 + (3/2)x_{-1} = (15/8)g_0\Delta a^2$, and the truncation error ϵ at the moment of time-step increase a_0 is:

$$\epsilon = \frac{5}{16}\dot{g}_0\Delta a^3, \tag{29}$$

which should be compared with the truncation of the constant step leap-frog scheme:

$$\epsilon = \frac{1}{12}\ddot{g}_0\Delta a^4. \tag{30}$$

The truncation error at the moment of modifying the time-step is clearly larger than for the constant-step leapfrog, but it is still a third-order approximation. The reason for that is the selection of the numerical factor 5/4 in the kick operator (eq. 28), which kills the second-order error. These errors are only for a single time-step. The cumulative error for a large

number of steps depends on how single-steps errors accumulate. This typically results in scaling the force resolution $\epsilon \propto \Delta a^2$ for the constant time-step. Because there are only very few number of times when the time-step is increased in our code (typically 5-10 times for the total ~ 150 of steps), the final error is mostly dominated by the cumulative error of the constant-step kicks and drifts.

(2) Late stages of evolution $z < z_{\rm limit}$. As fluctuations evolve and become very nonlinear, halos start to form, merge and grow. At this stage the main concern is how accurately the code traces the evolution of dark matter in halos. The number of time-steps is an important factor defining the accuracy. However, the number of steps is just one of the factors: one cannot really find out the required number of steps without specifying the force resolution and without knowing the science application and requirements of the simulations.

Our goal is to generate PPM-GLAM simulations that reproduce the dark matter density and velocity fields with the resolution of up to $\sim 1/3 - 1/2\,h^{-1}$Mpc. Peculiar velocities are an integral part of the process implying that redshift distortions should be simulated, not added posteriorly using some analytical prescription. The force resolution and the magnitude of the peculiar velocities set stringent constraints on the allowed time-step.

The largest peculiar velocities $\sim 1000 - 3000\,{\rm km\,sec^{-1}}$ occur in clusters and large galaxy groups. The time-step should be small enough so that per time-step a particle should move by less than a fraction of a cell. Thus, for both stability and accuracy of the integration (Hockney & Eastwood 1988),

$$\beta \equiv \frac{v\Delta t}{\Delta R} \lesssim 1, \qquad (31)$$

were v is the particle velocity, Δt and ΔR are the time-step and the (proper) cell size. Assuming that the time-step is small, we can write $\Delta t = \Delta a/aH(a)$. If $\Delta x = \Delta R/a$ is the comoving cell size, then we can write β in the following form:

$$\beta = \frac{v}{a\Delta x}\frac{\Delta a}{aH(a)} = \frac{v}{\Delta x H_0}\frac{\Delta a}{a}\sqrt{\frac{a}{\Omega_0 + \Omega_\Lambda a^3}}. \qquad (32)$$

Scaling velocities and resolution to some characteristic values we finally write the condition for selecting the time-step as follows

$$\beta = 10\left[\frac{\Delta a}{a}\right]\left[\frac{v_{1000}}{\Delta x_{\rm Mpc}}\right]\sqrt{\frac{a}{\Omega_0 + \Omega_\Lambda a^3}} < 1, \qquad (33)$$

where v_{1000} is the peculiar velocity in units of $1000\,{\rm km\,sec^{-1}}$ and $\Delta x_{\rm Mpc}$ is the comoving cell size in units of h^{-1}Mpc.

This condition is difficult to satisfy if the number of steps is small. To make an estimate, let's assume that a PM code makes 40 time-steps using a constant-step leapfrog scheme (e.g. ICE-COLA and FastPM Izard et al. 2015; Feng et al. 2016). This implies that at $z \approx 0$ the time-step is about $\Delta a/a \approx 2.5 \times 10^{-2}$. Because we want the code to attain realistic velocities inside clusters of galaxies, we take $v = 2000\,{\rm km\,sec^{-1}}$. For typical force resolution of $\Delta x = 0.3\,h^{-1}$Mpc we find that $\beta = 1.7$. In other words, dark matter particles are moving too fast for this combination of peculiar velocity and resolution.

What happens if the time-step is too big? In this case large halos will not be as dense as they should be and random velocities are smaller in the central halo regions. This will be observed as a decline in the power spectrum of dark matter. For example, Feng et al. (2016) using FastPM find a decline of 4% in the power spectrum at $k = 1h^{-1}$Mpc for simulations with force resolution $\Delta x = 0.22\,h^{-1}$Mpc and 40 time-steps. However, the main concern and the main problem is that the defect depends on the local density and *rms* velocity. As such, it affects much more massive clusters, where velocities are large, than small halos with small *rms* velocities.

In our simulations the time-step at later moments becomes relatively small with a typical value of $\Delta a/a \approx (0.75 - 1) \times 10^{-2}$, which is sufficient even for fast particles in very massive clusters.

5 PARALLELIZATION

Parallelization of PPM-GLAM is done with OpenMP directives. Because OpenMP can be applied only to memory on a single computational node, this limits the number of particles and force resolution. This also makes the code faster because the code does not use slow communications across and iside computational nodes required for MPI parallelization. Using only OpenMP directives also makes the code simple and easy to modify. The later is very important because data analysis routines are still being modified and improved.

For solving the Poisson equation the PPM-GLAM uses FFT fortran-90 routines for the real-to-real transformations provided by publicly available code FFT5PACK (Swarztrauber 1984). This makes the code portable: no libraries should be installed. Using MKL routines provided by the Intel Fortran compilers may further improve the code performance.

Each OpenMP thread handles ${N_{\rm g}}^2$ 1-D FFT transformations. After sweeping the 3-D mesh in two directions the matrix is transposed to improve the data locality. Then the FFT is applied again twice: to complete the 3-D sweep and to start the inverse FFT. The matrix is transposed back, and the other two FFT sweeps are completed. OpenMP ATOMIC directives are applied for density assignments, which slows down the code, but allows it to run in parallel. The motion of particles is a naively parallel part of the code. Overall, the code uses only one 3-D matrix and requires three 3-D FFT passes.

6 EFFECTS OF TIME-STEPPING AND FORCE-RESOLUTION

6.1 Effects of time-stepping

To estimate effects of finite time-step, we run a series of simulations that start with the same initial conditions at $z_{\rm init} = 100$, have the same force resolution $\Delta x = 0.33\,h^{-1}$Mpc, and differ only by the time-stepping parameters. Specifically, we run four realisations with box size $1h^{-1}$Gpc, $N_{\rm p} = 1000$ and $N_{\rm g} = 3000$. The number of time-steps was changing almost by a factor of two from one simulation to another with $N_{\rm s} = 34, 68, 147, 302$. The two runs with $N_{\rm s} = 34, 68$ have the time-step $\Delta a/a \approx 0.15, 0.06$ all the time, while the other two runs have $\Delta a/a$ at $z > 3$ limited to $\Delta a/a \approx 0.036, 0.015$ for $N_{\rm s} = 147, 302$ correspondingly, and a constant Δa at later moments. At $z = 0$ they had $\Delta a/a \approx 0.014, 0.006$ for

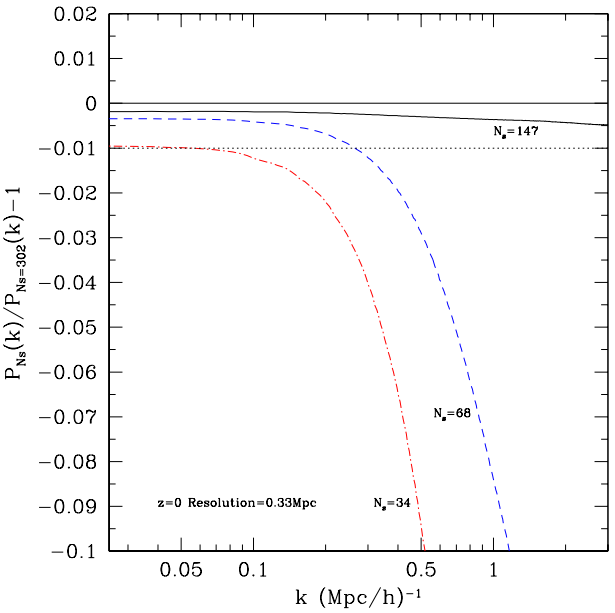

Figure 2. Effects of the number of time-steps on the amplitude and convergence of the power spectrum. The simulations have the same initial conditions, number of particles $N_{\rm p} = 1000$, box size $1h^{-1}$Gpc and force resolution $\Delta x = 0.33h^{-1}$Mpc. The only difference between $P(k)$ for various realisations, relative to the power spectrum of the simulation with the largest number of time-steps $N_{\rm s} = 302$, is the adopted number of time-steps, which is indicated in the plots. Results clearly converge when the number of steps increases and becomes $\gtrsim 100$ with very little difference between simulations with $N_{\rm s} = 147$ and $N_{\rm s} = 302$.

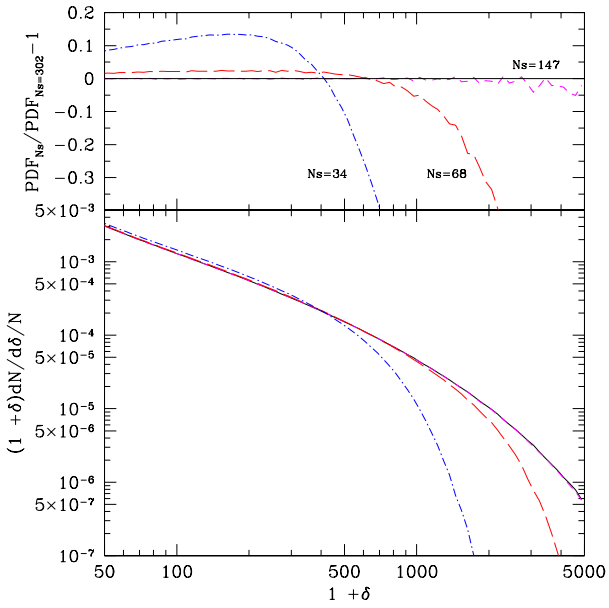

Figure 3. The same as in Figure 2 but for the convergence of the density distribution function. Results clearly converge when the number of time-steps increases and becomes $\gtrsim 100$. However, a smaller number of steps results in a dramatic suppression of the number of high-density regions where DM particles move very fast, which is observed as an artificial scale-dependent bias.

$N_{\rm s} = 147, 302$ respectively. For comparison, a run with a constant Δa, initial $z_{\rm init} = 39$, and $N_{\rm s} = 40$ has $\Delta a/a = 0.024$ at $z = 0$ and $\Delta a/a \approx 0.1$ at $z = 3$.

Figure 2 shows results for the power spectrum of fluctuations relative to the power spectrum of the simulation with the largest number of time-steps $N_{\rm s} = 302$. There are clearly significant errors in the simulations with the smaller number of steps. Note that the errors are small at long waves which indicates that even a small number of steps is sufficient for tracking the linear growth of fluctuations. However, the errors dramatically increase at small scales because the code cannot keep particles with large velocities inside dense regions. When the number of steps increases the accuracy also improves very substantially, we clearly see converge of the power spectrum when the number of steps increases and becomes $\gtrsim 100$.

The power spectrum may give somewhat too optimistic impression. After all, even 34 time-steps give an error in $P(k)$ of only 3% at $k \sim 0.3\,h{\rm Mpc}^{-1}$. The problem is that the error is much larger if we look at dense regions. We study this effect by analyzing the density distribution function of dark matter PDF, i.e. the fraction of volume occupied by cells with a given overdensity δ. In order to do that we find the density in each cell of the 3000^3 mesh, and count the number of cells in a given density range $(\delta, \delta+\Delta\delta)$. Figure 3, bottom panel, shows the PDF for those simulations with different number of time-steps. At low densities, the PDF is relatively insensitive to the number of steps, and this is why the errors in $P(k)$ were also reasonable at long-waves. The situation is quite different at large densities: relative errors are very large for densities $\delta > 1000$, see top panel in Figure 3. The plot also shows a clear convergence for the simulations with the larger number of steps with very little difference between $N_{\rm s} = 147$ and $N_{\rm s} = 302$.

6.2 Effects of force resolution

Figures 1 and 2 in the main text show how the power spectrum converges as the force and mass resolution increase. Here we present results of some additional tests. In order to study the effects of the force resolution we run the simulations with the same number of time-steps $N_{\rm s} = 136$ and number of particles $N_p = 1000^3$, and change the force resolution from $\Delta x = 0.25\,h^{-1}$Mpc to $1\,h^{-1}$Mpc by running the same initial conditions using different mesh-sizes. The smallest mesh has the same number of grid points as the the number of particles $N_g = N_p = 1000$. We then run other simulations with $N_g = 2000^3, 3000^3$, and 4000^3 meshes. We also run an additional simulation with $N_g = 2000^3$ but with twice larger time-steps ($N_{\rm s} = 270$). Figure 4 presents the ratio at $z = 0$ of the power spectrum $P(k)$ in each simulation to that with the highest resolution $N_{\rm g} = 4000$.

Figure 5 shows the evolution of the power spectra (scaled by k^2 to reduce the dynamical range) in these simulations. Results indicate $\sim 1\%$ convergence for $k \lesssim 1\,h{\rm Mpc}^{-1}$. At smaller scales the error increases, but it is still $\sim 20-30\%$ even at $k \approx (3-5)\,h{\rm Mpc}^{-1}$, which is also consistent with what we find from the comparison with the MultiDark simulations in Figure 1 (left panel).

The evolution of the power spectra presented in Figure 5 demonstrates significant suppression of fluctuations for the

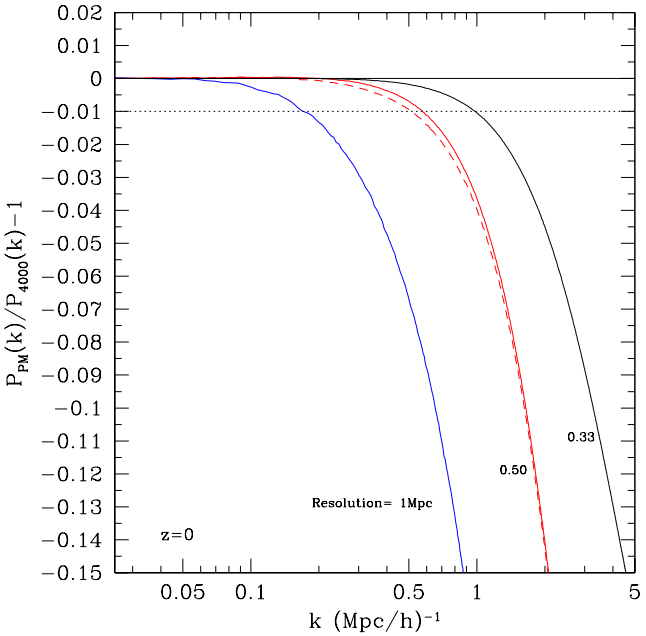

Figure 4. Effects of force resolution on the power spectrum $P(k)$ at $z = 0$ of a series of simulations with the same number of particles $N_p = 1000^3$ and computational box $L = 1000h^{-1}$Mpc. We run these simulations for grid sizes of N_g =1000, 2000, 3000, and 4000 with the force resolution ranging from $\Delta x = 0.25h^{-1}$Mpc to $1h^{-1}$Mpc. The plot shows the ratio of the power spectrum $P(k)$ in each simulation to that set-up with the highest resolution run $N_g = 4000$. The dashed-curve is for a simulation with twice larger number of time-steps. With a resolution of $\Delta x = 0.5h^{-1}$Mpc the number of steps $N_s \approx 100$ was sufficient.

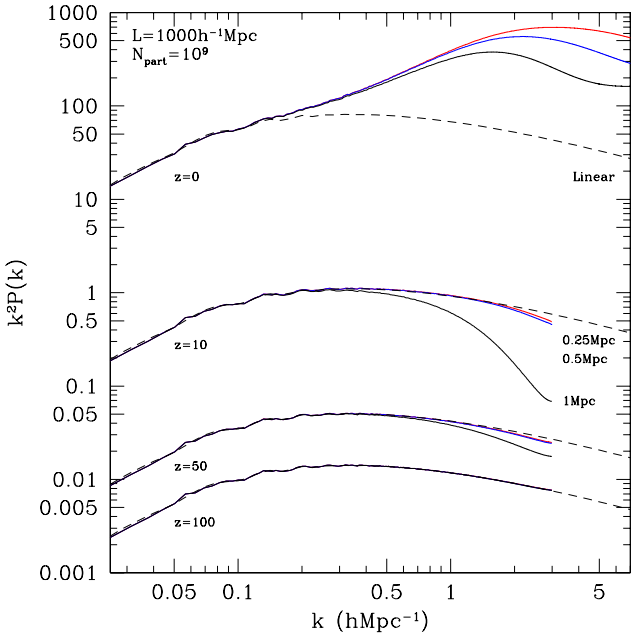

Figure 5. Evolution of the power spectrum with redshift for the various simulations. Better force resolution increases the power spectrum, but there are clear indications of convergence at a fixed wavenumber. The simulation with the number of particles equal to the mesh-size (labeled 1 Mpc in the plot) shows disproportionally large suppression of fluctuations at initial stages of evolution.

simulation with the same number of particles and mesh cells $N_g = N_p = 1000$.

REFERENCES

Bryan G. L. et al., 2014, Rev.Astrn.Astrophys., 211, 19
Feng Y., Chu M.-Y., Seljak U., 2016, ArXiv e-prints
Habib S. et al., 2014, ArXiv e-prints
Hockney R. W., Eastwood J. W., 1988, Computer simulation using particles
Izard A., Crocce M., Fosalba P., 2015, ArXiv e-prints
Klypin A., Holtzman J., 1997, ArXiv Astrophysics e-prints
Klypin A. A., Shandarin S. F., 1983, MNRAS, 204, 891
Koda J., Blake C., Beutler F., Kazin E., Marin F., 2016, MNRAS, 459, 2118
Kravtsov A. V., Klypin A. A., Khokhlov A. M., 1997, Rev.Astrn.Astrophys., 111, 73
Springel V., 2005, MNRAS, 364, 1105
Swarztrauber P., 1984, Parallel Computing, 1, 45
Tassev S., Eisenstein D. J., Wandelt B. D., Zaldarriaga M., 2015, ArXiv e-prints
Tassev S., Zaldarriaga M., Eisenstein D. J., 2013, Journal of Cosmology and Astroparticle Physics, 6, 036
Teyssier R., 2002, Astr.Astrophy., 385, 337
White M., Tinker J. L., McBride C. K., 2014, MNRAS, 437, 2594