The N-body Simulation

The dynamical equations of the density field are nonlinear. The N-body simulation is used to compute the evolution of the density contrast. The equation of motion for each particle depend on solving for the gravitational field due to all other particle in the expanding universe.

1 The Basic Equations

The Hubble parameter and the deceleration parameter of the backgroung universe are respectively given by

$$H(a) = H_0\sqrt{\Omega_{m0}a^{-3} + \Omega_{\Lambda 0}} = H_0 E(a) \tag{1}$$

and

$$q = \frac{[\Omega_{m0}a^{-3} - 2\Omega_{\Lambda 0}]}{2E^2(a)}. \tag{2}$$

We are interested in following the time evolution of the density contrast

$$\delta(\mathbf{x}, t) = \frac{\rho(\mathbf{x}, t) - \bar{\rho}(t)}{\bar{\rho}(t)}\,. \tag{3}$$

1.1 Fourier Transform

We consider a comoving volume $V = l_1 \times l_2 \times l_3$ with periodic boundary conditions *i.e.* $\delta(x) = \delta(x + l)$ and so on. We use the Fourier representation

$$\Delta(\mathbf{k}) = \int_{\in cube} d^3x \exp\left(-i\mathbf{k}\cdot\mathbf{x}\right)\delta(\mathbf{x}) \tag{4}$$

where $\Delta(\mathbf{k})$ is the Fourier transform of $\delta(\mathbf{x})$, and

$$\delta(\mathbf{x}) = \frac{1}{V}\sum_{-\infty}^{\infty} \exp\left(i\mathbf{k}\cdot\mathbf{x}\right)\Delta(\mathbf{k}). \tag{5}$$

The periodicity in real space makes Fourier space discrete, *i.e.* $\mathbf{k}$ takes values

$$\mathbf{k} = 2\pi\left(\frac{n_1}{l_1}\hat{\mathbf{i}} + \frac{n_2}{l_2}\hat{\mathbf{j}} + \frac{n_3}{l_3}\hat{\mathbf{k}}\right) \tag{6}$$

with

$$n_1, n_2, n_3 = 0, \pm 1, \pm 2, \pm 3, ... \tag{7}$$

1.2 Power Spectrum

The power spectrum $P(k)$ is defined through

$$\langle \Delta(\mathbf{k})\Delta^*(\mathbf{k}')\rangle = V\delta_{\mathbf{k},\mathbf{k}'}P(k) \tag{8}$$

where $\langle ....\rangle$ represents an ensemble average over different realizations of the random variables $\Delta(\mathbf{k})$. The power spectrum $P(\mathbf{k})$ has been assumed to be isotropic (direction independent) *i.e.* $P(\mathbf{k}) = P(k)$. The present day value of σ_8 (see Appendix C) is used to normalize the amplitude of power spectrum.

1.3 Initial Conditions

The initial conditions are set at some initial time t_i when $\delta(\mathbf{x})$ is in the linear regime.

1.3.1 Gaussian Random Field

We assume that the initial density fluctuations $\delta(\mathbf{x})$ are a Gaussian random field whereby

$$\Delta(\mathbf{k}) = \sqrt{\frac{VP(k)}{2}}[a(\mathbf{k}) + ib(\mathbf{k})] \tag{9}$$

where $a(\mathbf{k})$ and $b(\mathbf{k})$ are two real valued, independent Gaussian random variables of unit variance which satisfy

$$\langle a(\mathbf{k})a(\mathbf{k}')\rangle = \langle b(\mathbf{k})b(\mathbf{k}')\rangle = \delta_{\mathbf{k},\mathbf{k}'} \tag{10}$$

1.3.2 Growing Mode

We assume that $\delta(\mathbf{x}, t)$ is initially in the growing mode whereby

$$\delta(\mathbf{x}, t) = \frac{D(t)}{D(t_i)}\delta(\mathbf{x}, t_i) \tag{11}$$

and we have the peculiar velocity

$$\mathbf{v}_p(\mathbf{x}) = -aH(a)f\left[\nabla\nabla^{-2}\delta(\mathbf{x})\right]. \tag{12}$$

We calculate the growing mode using

$$D(a) = E(a)\int_0^a \frac{da'}{\left[a'E(a')\right]^3}, \tag{13}$$

and

$$f(a) = \frac{a}{D}\frac{dD}{da} = \frac{1}{a^2E^2D} - (1+q). \tag{14}$$

1.3.3 The Initial Power Spectrum

The cosmological model combined with theory and observations predict the present day linear power spectrum $P(k, t_0)$. This is the power spectrum that we would have if linear theory were valid at all length-scales in the present epoch. We use this to calculate the initial power spectrum at t_i which we use in equation (11)

$$P(k, t_i) = \left[\frac{D(t_i)}{D(t_o)}\right]^2 P(k, t_0) \tag{15}$$

1.3.4 The Zel'dovich approximation

The initial particle positions are calculated using the Zel'dovich Approximation (ZA)

$$\mathbf{x} \rightarrow \mathbf{x} - \nabla\nabla^{-2}\delta(\mathbf{x}). \tag{16}$$

This initial particle distribution corresponds to the initial density fluctuation field $\delta(\mathbf{x}, t_i)$ provided that $|\delta| \ll 1$. We have the initial velocities $\mathbf{v} = a\mathbf{v}_p$ (defined later) as

$$\mathbf{v} \rightarrow -aHf\nabla\nabla^{-2}\delta(\mathbf{x}). \tag{17}$$

Where $f = d(lnD)/d(lna)$.

1.4 Particle Dynamics

The action for a unit mass particle moving under gravity can be written as

$$S = \int_{t_1}^{t_2} \left[\frac{1}{2}\left(a^2\dot{\mathbf{x}}^2 - \ddot{a}a\mathbf{x}^2\right) - \psi\right] dt \tag{18}$$

where $\mathbf{x}$ is the comoving coordinate, $a(t)$ is the scale factor, $\psi(\mathbf{x}, t)$ is the gravitational potential

$$\nabla^2\psi = 4\pi G a^2 \rho(\mathbf{x}). \tag{19}$$

and ρ is the matter density. It is convenient to use $a(t)$ as parameter instead of t, whereby $dt = \frac{da}{\dot{a}}$ and using $d\mathbf{x}/da = \mathbf{x}'$ we have

$$S = \int_{a_1}^{a_2} \left[\frac{1}{2}\dot{a}a^2\mathbf{x}'^2 - \phi\right] da \tag{20}$$

where

$$\nabla^2\phi = \frac{4\pi G a}{H(a)}\bar{\rho}\delta(\mathbf{x}, t). \tag{21}$$

We have final equations of motion in terms of $\mathbf{v}$ which is the momentum conjugate to $\mathbf{x}$ as

$$\frac{d\mathbf{x}}{da} = \frac{\mathbf{v}}{a^3 H(a)} \tag{22}$$

$$\frac{d\mathbf{v}}{da} = -\nabla\phi(\mathbf{x}) \tag{23}$$

and

$$\nabla^2\phi = \frac{3}{2}\frac{H_0^2\Omega_{mo}}{a^2H(a)}\delta(\mathbf{x},t). \tag{24}$$

For detail calculation see Appendix A (equations (60), (61) and (62)).

1.5 Redshift Space Distortion

At the end of the particle distribution is mapped to redshift space using

$$\mathbf{s} = \mathbf{x} + \hat{\mathbf{n}}\frac{\hat{\mathbf{n}}\cdot\mathbf{v}_p}{aH(a)} \tag{25}$$

where $\hat{\mathbf{n}}$ is a unit vector along the line of sight from a distant observer and $\mathbf{v}_p = \mathbf{v}/a$ is the peculiar velocity. This is equivalent to the mapping

$$\mathbf{s} = \mathbf{x} + \hat{\mathbf{n}}\frac{\hat{\mathbf{n}}\cdot\mathbf{v}}{a^2H(a)} \tag{26}$$

The power spectrum in redshift space $P^s(\mathbf{k})$ depends on the direction of $\mathbf{k}$ through $\mu = \mathbf{k}.\hat{\mathbf{n}}/k$ which is the cosine of the angle between the wave vector $\mathbf{k}$ and line of sight $\hat{\mathbf{n}}$. It is convenient to quantify this anisotropy in terms of angular momenta $P_l^s(k)$ as

$$P^s(\mu,k) = \sum_l \mathcal{P}_l(\mu)P_l^s(k), \tag{27}$$

where

$$P_l^s(k) = (2l+1)\int \mathcal{P}_l(\mu)P^s(\mu,k)\frac{d\Omega_k}{4\pi}, \tag{28}$$

and $\mathcal{P}_l(\mu)$ are the Legendre polynomials.

2 Particle Mesh(PM) Code Implementation

We represent the simulation volume V using a $N1\times N2\times N3$ grid of spacing L *i.e.* $(l_1, l_2, l_3) = L \times (N_1, N_2, N_3)$, and use $\mathbf{m} = (m_1, m_2, m_3)$ to denote different points on the grid. We then have the comoving vector $\mathbf{x_m} = \mathbf{m}L$ for different points on the grid.

We use MM particles to represent the mass distribution inside the simulation volume. The program allows the possibility of populating every NF site with a particle are each incremented by NF when placing the particles, and we have

$$MM = \frac{N1 \times N2 \times N3}{(NF)^3} \,. \tag{29}$$

It is convenient to consider the particle trajectories with reference to the grid using the dimensionless variables

$$\mathbf{X} = \mathbf{x}/L \qquad \text{and} \qquad \mathbf{V} = \mathbf{v}/(LH_0) \tag{30}$$

2.1 Fourier Transform on Grid

The Fourier Transform on the grid is implemented through discrete Fourier transform (DFT) using C subroutine library FFTW (version 3.3.1). Given an array $A_\mathbf{m}$ (in real space), FFTW performs the operation

$$B_\mathbf{n} = \sum_\mathbf{m} \exp\left[-2\pi i\left(\frac{n_1 m_1}{N_1} + \frac{n_2 m_2}{N_2} + \frac{n_3 m_3}{N_3}\right)\right] A_\mathbf{m} \tag{31}$$

and returns the array $B_\mathbf{n}$, where the variables n_1, n_2, n_3 run over the range $0 \le n_1 < N1$ *etc.* on the grid. We interpret this as

$$B_\mathbf{n} = \sum_\mathbf{m} \exp(-i\mathbf{k_n}\cdot\mathbf{x_m}) A_\mathbf{m} \equiv [\mathrm{FT}(A)]_\mathbf{n} \,, \tag{32}$$

where the wave vector $\mathbf{k_n}$ has components

$$\mathbf{k_n} = 2\pi\left(\frac{n_1}{l_1}, \frac{n_2}{l_2}, \frac{n_3}{l_3}\right). \tag{33}$$

It is however necessary to fold (n_1, n_2, n_3) beyond $(\frac{N_1}{2}, \frac{N_2}{2}, \frac{N_3}{2})$ to obtain the negative wave vectors *i.e.* we use

$$n_1 \to n_1 - N_1 \qquad \text{for} \qquad n_1 > \frac{N_1}{2} \;\; \text{etc.} \tag{34}$$

The Backward Fourier Transform (BTF) in FFTW performs the operation

$$A_\mathbf{m} = \sum_\mathbf{n} \exp\left[2\pi i\left(\frac{n_1 m_1}{N_1} + \frac{n_2 m_2}{N_2} + \frac{n_3 m_3}{N_3}\right)\right] B_\mathbf{n} \equiv [\mathrm{BFT}(B)]_\mathbf{m} \,. \tag{35}$$

The expressions for Fourier transform stated above are unnormalized. We then have

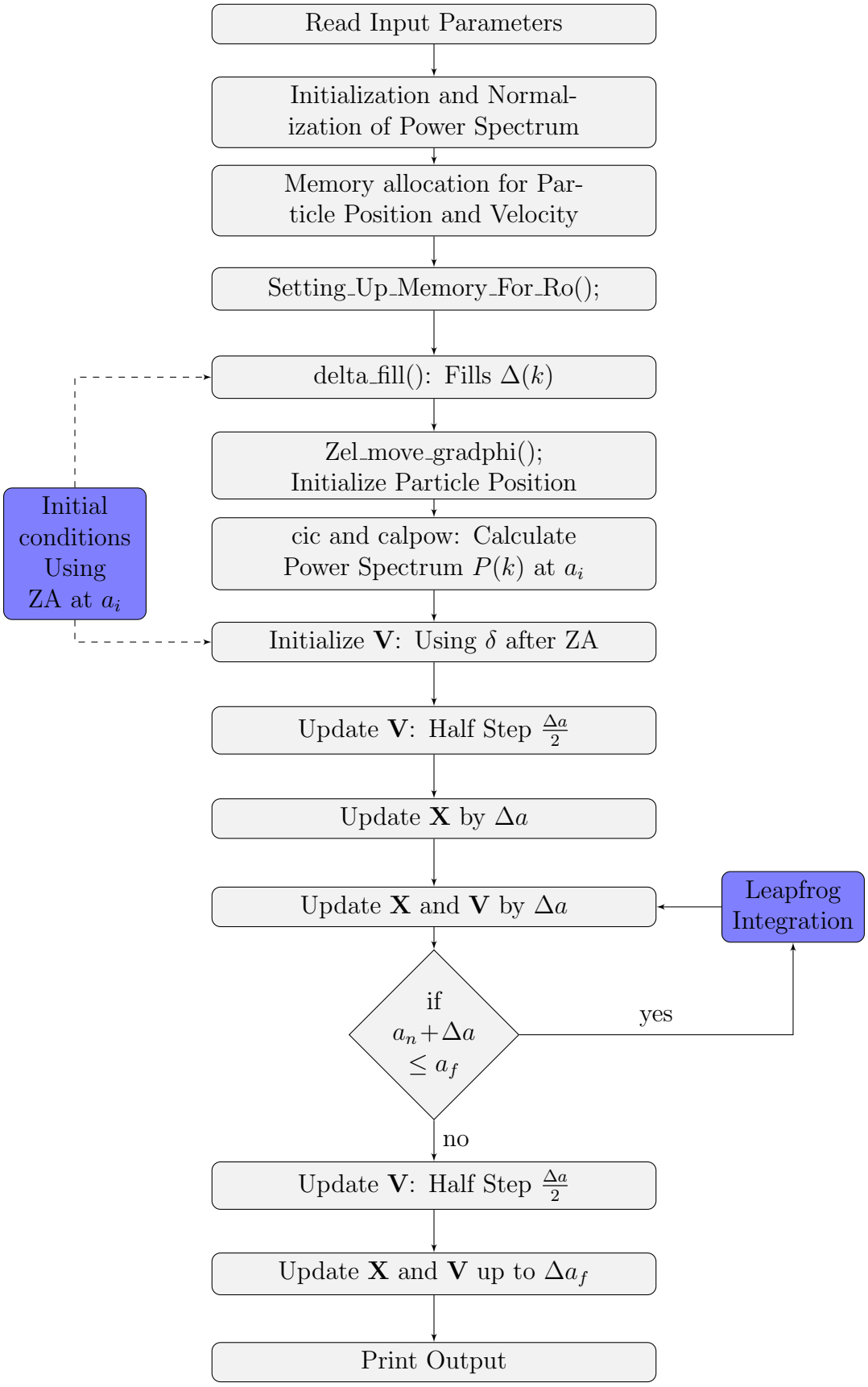
$$\Delta(\mathbf{k_n}) = L^3[\mathrm{FT}(\delta)]_\mathbf{n} \tag{36}$$

and

$$\delta(\mathbf{x_m}) = V^{-1}[\mathrm{BFT}(\Delta)]_\mathbf{m} \tag{37}$$

corresponding to equation(4) and (5) respectively. Here $\mathbf{x_m}$ and $\mathbf{k_n}$ are defined on the grid.

2.2 Flowchart



2.3 Inputs

We read the input parameters for our simulation from the file *input.nbody_comp*. The inputs for our code are given below

```
seed   Nbin                          |  -50012  10
vhh    vomegam    vomegalam   vnn    |  0.6704  0.3183  0.6817  0.9619
vomegab    sigma_8_present           |  0.04902  0.8347
N1  N2  N3  MM  LL                   |  2144  2144  2144  1231925248  0.07
aa_initial   NF    oflag             |  0.008   2   2
delta_aa                             |  0.004
Noutput  pk_flag                     |  5   1
nz                                   |  25.0  15.0  10.0  9.0  8.0
```

Where
seed: for the random number generation,
Nbin: number of bin for the power spectrum calculation,
vhh: current value of Hubble parameter (h) in $(Km/sec)Mpc^{-1}$ unit,
vomegam: present value of the total matter density (baryons+CDM) parameter (Ω_{m0}),
vomegalam: current value of cosmological constant or dark energy density parameter ($\Omega_{\Lambda 0}$),
vnn: spectral index of primordial power spectrum (n_s),
vomegab: present-day baryon matter density parameter (Ω_{b0}),
sigma _ 8_ present: amplitude of the power spectrum (σ_8) on the scale $8h^{-1}Mpc$ at present,
N1, N2, N3: number of grid points along x-axis, y-axis and z-axis,
MM: total number of particle,
LL: spacing between grid points in Mpc unit,
NF: number of filling, for every grid point ($NF = 1$) or for alternating grid point ($NF = 2$),
oflag: flag for writing final output in a specific unit ($h^{-1}Kpc$),
vaa: initial scale factor (a_i),
delta_aa: step size for the iteration (Δa),
Noutput: number of nbody outputs,
pk_flag: flag for power spectrum calculation,
nz: redshifts

All the cosmological parameter values are taken from *Planck* data with *WMAP* polarization data (Planck 2013 results. XVI. Cosmological parameters - Planck Collaboration (Ade, P.A.R. et al.) arXiv:1303.5076 [astro-ph.CO]).

2.4 Initialization and Normalization of Power Spectrum

The function TFset_parameters(vomegam*vhh*vhh,vomegab/vomegam,tcmb) (defined in the tf_fit.c) initializes the parameters need for power spectrum calculation. That function implement the fitting formula in *Eisenstein & Hu (1997)*, by calculating the transfer function for an arbitrary CDM+baryon universe. The output of that function nothing but it set many global variables used in TFfit_onek() function. Later, the function float Pk(float k) (see the powerspec.c program) return the power spectrum for a given k-mode.

The present day value of σ_8 (see Appendix C) is used to normalize the amplitude of power spectrum.

We calculate power spectrum at any time t_i (or a_i) given the present day power spectrum ($P(k, a_0)$) using the linear theory by the following formula (from equation (15))

$$P(k, a_i) = \pi^2 vol \cdot D^2(a_i) P(k, a_0) \tag{38}$$

where $D(a)$ is the growing mode, *vol* is the volume of our simulation box.

2.5 Setting Up Memory

In the void Setting_Up_Memory_For_Ro(float vaa) function we allocate memory for the density at each grid points ro = allocate_fftwf_3d(N1, N2, N3+2), where the function *allocate_fftwf_3d()* defined in *allotarrays.c*. Beside that we define the fftw-plans for forward and backward Fourier transform.

```
fftwf_plan_dft_r2c_3d(N1,N2,N3,&ro[0][0][0],(fftwf_complex*)&ro[0][0][0])
fftwf_plan_dft_c2r_3d(N1,N2,N3,(fftwf_complex*)&ro[0][0][0],&ro[0][0][0])
```

We also set the number of threads for parallelization

```
omp_set_num_threads(omp_get_max_threads());
```

and calculate some constants

```
norml=1./((1.0*N1)*(1.0*N2)*(1.0*N3));
Cx=2.*pi/(N1*LL);  Cy=2.*pi/(N2*LL);   Cz=2.*pi/(N3*LL);
Lcube=powf(LL,3.);
vol=Lcube/norml;
CC=pi*pi*vol*powf(Df(av),2.);
rho_b_inv=1.0/(norml*MM);
```

2.6 delta_fill: Fill $\Delta(\mathbf{k})$'s

The function void delta_fill(long* seed) generate the initial density contrast $\Delta(\mathbf{k_n})$ in Fourier space. We only fill one half of the box, since $\delta(\mathbf{x})$ is a real quantity and for real-to-complex Fourier transform another half of the box is just the complex conjugate of the other half. We use standard float gasdev(long*) function (from numerical recipes in c) to generate Gaussian random variable. The only input of delta_fill() is the seed for pseudo random number generation. The amplitudes of $\Delta(\mathbf{k_n})$ *i.e.* $\sqrt{\frac{VP(k_n)}{2}}$ are calculated using another function float p(long i,long j,long k).

2.7 Zel_move_gradphi()

The particles are first placed on the grid points. The initial particle positions are then displaced from the grid. The displacements are generated using Zel'dovich Approximation (ZA) equations (16) and (17). For ZA we need to calculate $-\nabla\phi = -\nabla\nabla^{-2}\delta(\mathbf{x})$, which we compute in Fourier space. We generate the displaced particle positions using

$$\mathbf{X} \rightarrow \mathbf{X} + L^{-1}\left[\mathrm{BFT}\left(\frac{i\mathbf{k}\Delta}{k^2}\right)\right] \tag{39}$$

and velocity

$$\mathbf{V} \rightarrow a^2 E f L^{-1}\left[\mathrm{BFT}\left(\frac{i\mathbf{k}\Delta}{k^2}\right)\right] \tag{40}$$

In the function void grad_phi(int ix) we calculate each (ix component i.e. x, y or z) component of $\frac{i\mathbf{k}\Delta}{k^2}$ and take the back Fourier transform. For that purpose we allocate an extra grid va[][][] of same order as ro[][][] to save the ixth component of $\nabla\phi$. At the start of this function, ro[][][] contains delta(k) and at end va[][][] contains the ixth component of $-\nabla\nabla^{-2}\delta(\mathbf{x})$

The void Zel_move_gradphi(float av,float **rra,float **vva) function takes scale factor, position array and velocity array as input. In this function for each component (x, y and z) we call *void grad_phi(int)* to get ith component of $-\nabla\nabla^{-2}\delta(\mathbf{x})$. We use those values to displaced the position and velocity according to equation (39) and (40). We impose periodic boundary condition to ensure that all the particle are inside the box.

2.8 cic()

After displacement, the particles are no longer on the grid points. We only define the densities on the grid points. The Cloud-in-Cell (CIC) algorithm is used to calculate density $(1+\delta)$ on the grid points from a given particle distribution. We first clear out the whole density array ro[ ][ ][ ]. The function void cic(float **rra) takes particle positions as input. Cloud-in-Cell (CIC) calculates contribution from the particle position to the eight corners of a cell. The formula for the weight to one of the corner $\mathbf{a}_{i,j,k}$ of a cell containing the particle at $\mathbf{x}$ is given by

$$W_{i,j,k} = \prod_m (1 - |a_m - x_m|), \tag{41}$$

where $m = 0, 1, 2$. Adding these weights from eight adjacent cells we get the value at that grid point.

2.9 Dynamical Iteration

The equations corresponding to the particle dynamics from equations (22), (23) and (24) become

$$\frac{d\mathbf{V}}{da} = \frac{3}{2}\frac{\Omega_{m0}}{a^2 L E(a)}[-\nabla\nabla^{-2}\delta] = f(\mathbf{X}, a) \tag{42}$$

and

$$\frac{d\mathbf{X}}{da} = \frac{\mathbf{V}}{a^3 E(a)} = g(\mathbf{V}, a) \tag{43}$$

(see Appendix B for more details).

For the iteration of positions and velocities we use leapfrog integration method. Initially we make positions($\mathbf{x}$) half step ahead of velocities($\mathbf{v}$). In leapfrog integration method the equation for updating position and velocity are

$$\mathbf{V}_{i+1/2} = \mathbf{V}_{i-1/2} + f(\mathbf{X}_i, a_i) \cdot \Delta a \tag{44}$$

$$\mathbf{X}_{i+1} = \mathbf{X}_i + g(\mathbf{V}_{i+1/2}, a_{i+1/2}) \cdot \Delta a \tag{45}$$

Where i to denote the i^{th} iteration and Δa is the step size.

2.9.1 Update_v()

This function is used for three operation

(a) Initialize $\mathbf{V}$ after ZA (when $\Delta a \not> 0$)

(b) Update $\mathbf{V}$ half Step $\frac{\Delta a}{2}$ for Leapfrog integration

(c) Update $\mathbf{V}$ (when $\Delta a > 0$) for N-body dynamic iteration

The void Update_v(float av,float delta_aa,float **rra,float **vva) function takes scale factor, scale factor step size, position array and velocity array as input. Here we first call *cic()* to calculate $\delta(\mathbf{x})$ on grid from particle distribution. After that we do forward Fourier transform of $\delta(\mathbf{x})$ to get $\Delta(\mathbf{k})$ at each grid point.

We calculate the potential(ϕ) *i.e.* $\nabla^{-2}\Delta(\mathbf{k})$ in Fourier space because in Fourier space this Laplacian inverse operation became multiplication with some factor. We do this by calling the function void Get_phi(int f_flag), in which we use the following multiplication factor for Laplacian inverse operation (see appendix B for more details)

$$\nabla^{-2} \Rightarrow -\frac{L^2}{4}\left[\sum_m sin^2\left(\frac{Lk_m}{2}\right)\right]^{-1} , \tag{46}$$

where $m = 0, 1, 2$. Next we do backward Fourier transform of $\nabla^{-2}\Delta(\mathbf{k})$ to get $\phi(x)$ on the grid. We compute the differentiation *i.e.* $\nabla\phi(x)$ (for each component) using the following expression

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] . \tag{47}$$

which gives the acceleration on the grid in real space.

Now we want the acceleration at the particle position, for that we calculate the weight similarly as CIC (see $ 3.7) but get the acceleration using the reverse process of CIC *i.e.* adding these eight weight from eight corner of a cell to the particle position. After that we just update the velocities according to equation (42) and (44) to get new velocities at some scale factor $a_i + \triangle a$.

2.9.2 Update_x()

When we have velocities(**v**), we can easily update the positions(**x**) using that velocities according to equation (43) and (45). Similarly as void Update_v() function the void Update_x(float aa,float delta_aa,float **rra,float **vva) function also takes scale factor, scale factor step size, position array and velocity array as input.

2.10 calpow()

We calculate the power spectrum using the function void calpow(int f_flag,int Nbin,double* power, double* powerk, double* kmode,long *no) from a given density distribution. The calpow() function takes f_flag, Nbin and density array ro[ ][ ][ ] as input. When the f_flag = 0, the density is in Fourier space and when it is 1, the density is in real space. We use this flag to convert the density in Fourier space. Then it checks whether the density is in Fourier space or not; if not then it do the Fourier transform to convert the density in Fourier space. We divide the full **k**-range into 'Nbin' number of bins (logarithmic) and check for all the **k**-modes in which bin it lies. Finally it gives average $P(k)$, k and number of modes for each bin. We match our calculated power spectrum with the fitting formula given by *Eisenstein & Hu (1997).*

2.11 write_output()

At the desired redshift, we print x, y and z components of position and velocity for each particle in the *output.nbody* file. We use the function int write_output(char *fname,long int seed,int output_flag,float **rra,float **vva,float vaa) to write the output in a binary file (gadget format). In this function, we take file name and output_flag as inputs. Output_flag is used to specify the units of position and velocity. When otput_flag =1 it writes position and velocity in grid units, otherwise position in $kpch^{-1}$ unit and velocity in km/sec unit.

First it writes the parameters from io_header header1, which is defined in 'nbody.h'. After that it writes the whole position array (rra[][][]) and it's corresponding velocity array (vva[][][]) at the desired redshift (vaa).

References

Efstathiou, G., Davis, M., Frenk, C., & White, S.D.M. 1985, ApJS, 57, 241.
Hockney, R.W., & Eastwood, J.W. 1981, Computer Simulations using Particles,Mc Graw-Hill.

3 Appendices

Appendix A

The physical coordinate

$$\mathbf{r} = a\mathbf{x} \tag{48}$$

$\mathbf{x}$ is comoving coordinate. Then velocity

$$\mathbf{u} = \dot{\mathbf{r}} = \dot{a}\mathbf{x} + a\dot{\mathbf{x}} \tag{49}$$

Peculiar velocity: Deviation from Hubble flow

$$\mathbf{v}_p = \mathbf{u} - \dot{a}\mathbf{x} = \mathbf{u} - H_0\mathbf{r} \tag{50}$$

The Lagrangian of a unit mass particle (using equivalence principle) under a gravitational potential $\psi(r)$

$$L = \frac{1}{2}\dot{\mathbf{r}}^2 - \psi(\mathbf{r}) \tag{51}$$

Considering the expanding coordinate system as well as the peculiar velocity

$$\begin{aligned} L &= \frac{1}{2}(\dot{a}\mathbf{x} + a\dot{\mathbf{x}})^2 - \psi(\mathbf{x}) \\ &= \frac{1}{2}(\dot{a}^2\mathbf{x}^2 + a^2\dot{\mathbf{x}}^2 + 2a\dot{a}\mathbf{x}\dot{\mathbf{x}}) - \psi(\mathbf{x}) \\ &= \frac{1}{2}a^2\dot{\mathbf{x}}^2 - \frac{1}{2}a\ddot{a}\mathbf{x}^2 - \psi(\mathbf{x}) + (\frac{1}{2}\dot{a}^2\mathbf{x}^2 + \frac{1}{2}a\ddot{a}\mathbf{x}^2 + a\dot{a}\mathbf{x}\dot{\mathbf{x}}) \\ &= \frac{1}{2}a^2\dot{\mathbf{x}}^2 - \frac{1}{2}a\ddot{a}\mathbf{x}^2 - \psi(\mathbf{x}) + \frac{d}{dt}(\frac{1}{2}a\dot{a}\mathbf{x}^2) \end{aligned} \tag{52}$$

Now, we know L invariant under addition or subtraction of any total derivative of coordinate or velocity w.r.t time.

$$\begin{aligned} L &= \frac{1}{2}a^2\dot{\mathbf{x}}^2 - \frac{1}{2}a\ddot{a}\mathbf{x}^2 - \psi(\mathbf{x}) \\ &= \frac{1}{2}a^2\dot{\mathbf{x}}^2 - \phi(\mathbf{x}) \end{aligned} \tag{53}$$

where

$$\phi(\mathbf{x}) = \psi(\mathbf{x}) + \frac{1}{2}a\ddot{a}\mathbf{x}^2 \tag{54}$$

Then the Hamiltonian of that unit mass

$$H = \frac{\mathbf{p}^2}{2a^2} + \phi(\mathbf{x}) \tag{55}$$

Now for homogeneous and isotropic universe

$$\ddot{a}\mathbf{x} = -\frac{4}{3}\pi r^3\bar{\rho}\frac{G}{r^2}\hat{\mathbf{r}} \tag{56}$$

where $\bar{\rho}$ is the average density of the universe. Then

$$\ddot{a} = -\frac{4}{3}\pi a\bar{\rho}G \tag{57}$$

$\psi(\mathbf{x})$, the gravitational potential satisfies the Poisson's equation

$$\nabla_r^2\psi(\mathbf{r}) = 4\pi G\rho \tag{58}$$

$$\nabla_x^2\psi(\mathbf{x}) = 4\pi G\rho a^2 \tag{59}$$

Now

$$\begin{aligned}
\nabla_x^2\phi(\mathbf{x}) &= \nabla_x^2\left[\psi(\mathbf{r}) + \frac{1}{2}a\ddot{a}x^2\right] \\
&= 4\pi G\rho(\mathbf{x})a^2 + \frac{1}{2}a\ddot{a} \times 6 \\
&= 4\pi\rho G a^2 + 3a\left(-\frac{4}{3}\pi\bar{\rho}Ga\right) \\
&= 4\pi G a^2\bar{\rho}\left(\frac{\rho}{\bar{\rho}} - 1\right) \\
\nabla_x^2\phi(\mathbf{x}) &= 4\pi G a^2\bar{\rho}\delta(\mathbf{x})
\end{aligned} \tag{60}$$

The Conjugate momentum, from equation (55)

$$\frac{d\mathbf{x}}{dt} = \frac{\partial H}{\partial \mathbf{v}} = \frac{\mathbf{v}}{a^2} \tag{61}$$

and force

$$\frac{d\mathbf{v}}{dt} = -\frac{\partial H}{\partial x} = -\frac{\partial\Phi(x)}{\partial x} = -\nabla\phi(\mathbf{x}) \tag{62}$$

The last three equations (60), (61) and (62) are our main result.

Appendix B

The simplest approximation for acceleration *i.e.* $-\nabla\nabla^{-2}\delta(\mathbf{x})$ is $BFT\left[\frac{i\mathbf{k}\Delta(\mathbf{k_n})}{k^2}\right]$. Which we use in Zel'dovich approximation, because at early time small k-modes *i.e.* large scale fluctuation are important. But at latter time of evolution when all k-mode are important we need to use better approximation. We know the Taylor series expansion of a function $f(x)$ around some small value h is given by

$$f(x+h) = f(x) + f'(x)\cdot h + \frac{1}{2}h^2 f''(x) + \dots \tag{63}$$

Using the calculus of finite differences, we can write from equation (63)

$$f'(x) = \frac{1}{2h}[f(x+h) - f(x-h)] \tag{64}$$

and

$$f''(x) = \frac{1}{h^2}[f(x+h) + f(x-h) - 2f(x)]\ . \tag{65}$$

let's take an example in $1D$

$$\frac{d^2}{dx^2}f(x) = g(x) \tag{66}$$

Then from equation (65)

$$\frac{1}{h^2}[f(x+h) + f(x-h) - 2f(x)] = g(x) \tag{67}$$

Taking Fourier transform of equation(67) and using shift theorem *i.e.* if $FT[f(x)] = F(k)$ then $FT[f(x-h)] = e^{-2\pi ikh}F(k)$ we get

$$\frac{1}{h^2}[e^{2\pi ikh}F(k) + e^{-2\pi ikh}F(k) - 2F(k)] = G(k) \tag{68}$$

$$\frac{2}{h^2}\left[\frac{e^{2\pi ikh} + e^{-2\pi ikh}}{2} - 1\right]F(k) = G(k) \tag{69}$$

$$\frac{2}{h^2}[cos(2\pi kh) - 1]F(k) = G(k) \tag{70}$$

$$F(k) = -\frac{h^2}{4}[sin^2(\pi kh)]^{-1}G(k) \tag{71}$$

That means in Fourier space $(\frac{d}{dx})^{-2} \Rightarrow -\frac{h^2}{4}[sin^2(\pi kh)]^{-1}$. We can extend this calculation in $3D$. Then in Fourier space with reference to the grid unit we can write

$$\nabla^{-2} \Rightarrow -\frac{L^2}{4}\left[\sum_m sin^2(Lk_m/2)\right]^{-1}, \tag{72}$$

where $m = 0, 1, 2$ and from equation (64)

$$\nabla \Rightarrow \frac{i}{L}\left[sin(Lk_x)\hat{x} + sin(Lk_y)\hat{y} + sin(Lk_z)\hat{z}\right] \tag{73}$$

First Method

In this method we first calculate the three components of acceleration on grid in Fourier space using equations (72) and (73). Then we take back Fourier transformation of these three components separately using 'Fastest Fourier Transform in the West' (FFTW) software to get acceleration in real space on grid. After that for each of three component we interpolate those value of acceleration to the particle position using a method called 'Cloud-in-Cell' CIC (see $ 3.7 for details).

Second Method

Here we first calculate the potential i.e. $\nabla^{-2}\Delta(\mathbf{k})$ on grid in Fourier space using equation (72). Then we take back Fourier transformation to get potential (ϕ) on grid in real space (using FFTW). Next we compute the differentiation using equation (64) in real space and calculating the weight (see $ 3.7 for details) we get the acceleration at the particle position.

Input

To study the relative merits and demerits of those two methods, We took density contrast as 3D Dirac delta function $\delta_D^3(\mathbf{x})$. We know that

$$\nabla^2\left(\frac{1}{r}\right) = -4\pi\delta_D^3(\mathbf{r}) \tag{74}$$

and

$$-\nabla\nabla^{-2}\delta_D^3(\mathbf{r}) = \frac{1}{4\pi}\nabla\left(\frac{1}{r}\right) = -\frac{1}{4\pi}\frac{\hat{\mathbf{r}}}{r^2}\ . \tag{75}$$

The density on grid is the cell average density. In this case the density on grid is

$$\rho_{(0,0,0)} = \frac{1}{L^3}\iiint_{-L/2}^{L/2}\delta_D^3(\mathbf{x})d^3x = \frac{1}{L^3}\ , \tag{76}$$

otherwise zero. For our convenience we have taken the density at origin is $4\pi L^{-3}$ and elsewhere zero.

Output

We compute force taking grid spacing 0.6 and 256^3 grid point. From equation (73), we can tell that the output should fall as square of the distance. We only took the z-axis for plotting. We have plotted $z^2 \cdot \nabla\nabla^{-2}\delta_D^3(z)$ with z ($= L \cdot z_m$). Figure 1 shows the outputs from those two methods.

Run Time

We have compute the times to run the functions for the two methods in the computer 10.34.32.23@vijaya, CTS, IIT Kharagpur. We have taken grid spacing 0.6 and six different box size i.e. 64^3, 128^3, 192^3, 256^2, 364^3, and 512^3. Figure 2 shows our results.

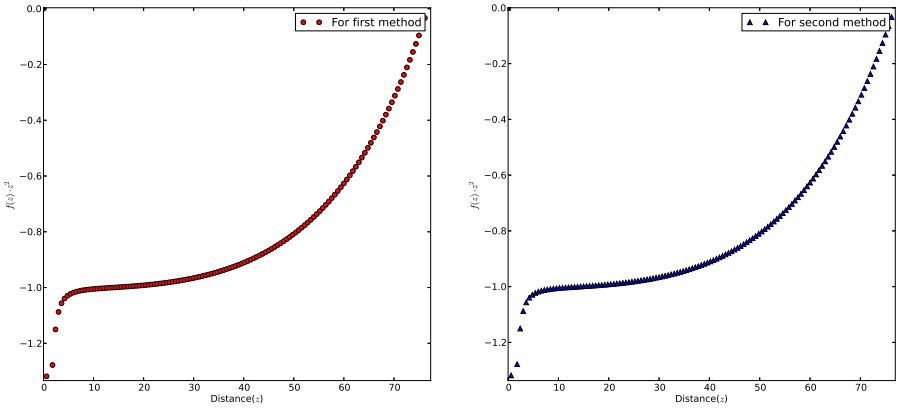


Figure 1: *Plot of* $z^2 \cdot \nabla\ \nabla^{-2}\delta_D^3(z)$ *as a function of z. Left: using first method. Right: using second method*

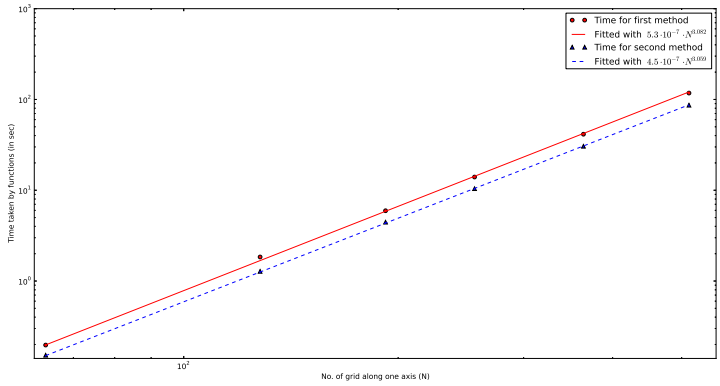


Figure 2: *The plot of the times required in two different method. Time required by first method shown by red dot, which fitted by* $5.3 \times 10^{-7} \cdot N^{3.082}$ *shown by red line. Time required by second method shown by blue pyramid, which is fitted by* $4.5 \times 10^{-7} \cdot N^{3.059}$ *shown by blue dash line*

Conclusions

For both the method we got same result but second method is more efficient than first one. So we use the second method for force calculation.

Appendix C

A common concept in the calculation of density field is that of filtering, where the density field is convolved with some window function f_r. then

$$\delta(x) \rightarrow \delta(x) * f_r(x) \tag{77}$$

$$\delta(x) = \frac{1}{V}\int d^3x'\delta(x')f(\vec{x}' - \vec{x}) \tag{78}$$

In Fourier space using convolution theorem we can write $\Delta(k) \rightarrow \Delta(k)f_k$, where f_k is the Fourier transform of f_r.Then the variance of the density field

$$\sigma^2 = \frac{1}{(2\pi)^3}\int P(k)|f_k|^2 d^3k \tag{79}$$

We use spherical Top-Hat window function $W_R(k)$ given by

$$W_R(k) = \frac{3}{(kR)^3}[sin(kR) - kRcos(kR)] \tag{80}$$

in real space

$$\begin{aligned} W_R(r) &= 1 \quad if \quad r \leq R \\ &= 0 \quad Otherwise \end{aligned} \tag{81}$$

σ_8 is the rms density fluctuation smooth with a spherical Top-Hat filter of radius $8h^{-1}Mpc$.

$$\sigma_8 = \left[\frac{1}{2\pi^2}\int P(k)|W_8(k)|^2 k^2 dk\right]^{1/2} \tag{82}$$