Writing a particle-mesh N-Body code

Benedikt Diemer, ASTR 321, 03/11/2011

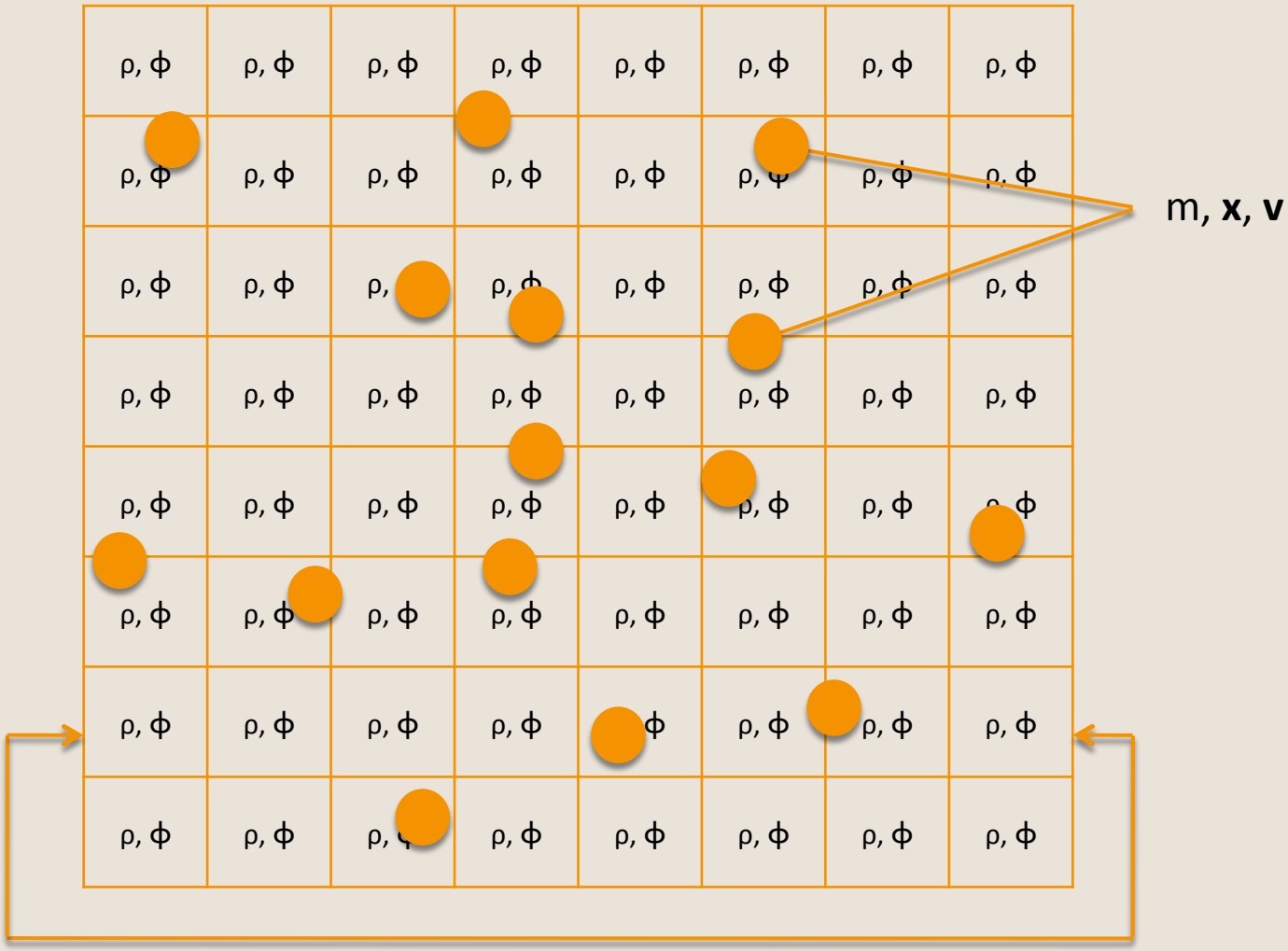
Part I

Introduction

Goal

- Compute gravitational interaction between large number of particles
- Direct summation of forces is $O(N^2)$ – Millennium run had 10^{10} particles, leaving 10^{20} interactions per timestep!
- PM: Particles keep track of the moving masses (Lagrangian), mesh of the fields (Eulerian)
- Credit for this particular implementation: Anatoly Klypin, Andrey Kravtsov

Particle-mesh scheme



Periodic boundary conditions

Procedure

Step	Order
Compute density field from particles	N_p
Solve for potential field	$N_c \log_2(N_c)$
Take gradient of potential as acceleration	N_p
Advance particle positions and velocities	N_p
Order of all operations	$\mathbf{N_p + N_c \log_2(N_c)}$

- Field operations dominate
- Say this algorithm is 1000 times slower than direct summation; for 256^3 (16.8 m) particles,

  $N^2 / 1000\ N \log_2(N) = 700$ times faster!

Part II

Algorithms

Step 0: Dimensionless variables

- We can define dimensionless variables in which

$$r_0 = \frac{L_{\rm box}}{N_g}; \quad N_g^3 = \text{total number of grid cells}$$

$$\begin{aligned} t_0 &\equiv H_0^{-1}, \\ v_0 &\equiv \frac{r_0}{t_0}, \\ \rho_0 &\equiv \frac{3H_0^2}{8\pi G}\Omega_{m,0}, \\ \phi_0 &\equiv \frac{r_0^2}{t_0^2} = v_0^2. \end{aligned}$$

$$\tilde{\mathbf{x}} \equiv a^{-1}\frac{\mathbf{r}}{r_0}, \quad \tilde{\mathbf{p}} \equiv a\ \frac{\mathbf{v}}{v_0}, \quad \tilde{\phi} \equiv \frac{\phi}{\phi_0}, \quad \tilde{\rho} = a^3\frac{\rho}{\rho_0}$$

$$\phi = \Phi + 1/2\, a\ddot{a}\,(r/a)^2 = \Phi + \frac{H_0^2}{2}\left(\Omega_{\Lambda,0} - \frac{1}{2}a^{-3}\Omega_{m,0}\right) r^2$$

Step 0: Dimensionless variables

$$\tilde{\nabla}^2\tilde{\phi} = \frac{3}{2}\frac{\Omega_0}{a}\tilde{\delta},$$

$$\frac{d\tilde{\mathbf{p}}}{da} = -f(a)\tilde{\nabla}\tilde{\phi}, \quad \frac{d\tilde{\mathbf{x}}}{da} = f(a)\frac{\tilde{\mathbf{p}}}{a^2}.$$

$$\tilde{\delta} = \tilde{\rho} - 1$$

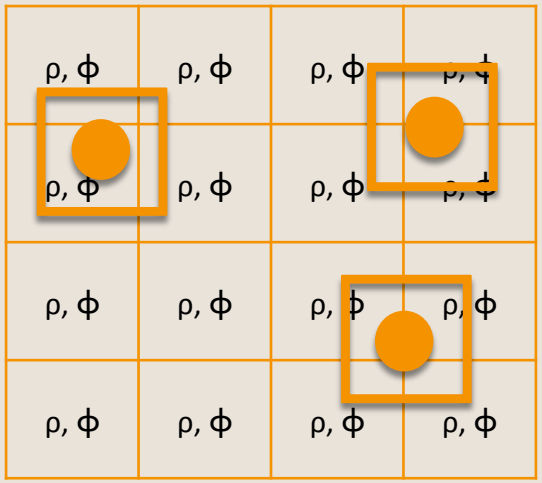
$$f(a) \equiv H_0/\dot{a} = \left[a^{-1}\left(\Omega_{m,0} + \Omega_{k,0}a + \Omega_{\Lambda,0}a^3\right)\right]^{-1/2}$$

- The only physical quantities are a and Ω_x!

  ➔ we treat a as the time variable
- After the simulation, positions and velocities can be converted back into Mpc, km / s etc.

Step 1: Density field

- Cloud in cell algorithm: distribute particle mass evenly in a space the size of one cell



Step 2: Potential from density

- Poisson's equation, discretized and interpolated reads

$$\begin{aligned}\tilde{\nabla}^2\tilde{\phi} &\approx \tilde{\phi}_{i-1,j,k}+\tilde{\phi}_{i+1,j,k}+\tilde{\phi}_{i,j-1,k}+\tilde{\phi}_{i,j+1,k}+\tilde{\phi}_{i,j,k-1}+\tilde{\phi}_{i,j,k+1}-6\tilde{\phi}_{i,j,k}= \\ &= \frac{3}{2}\frac{\Omega_0}{a}(\tilde{\rho}_{i,j,k}-1).\end{aligned}$$

- These N_c^3 (!) equations are solved in Fourier space by

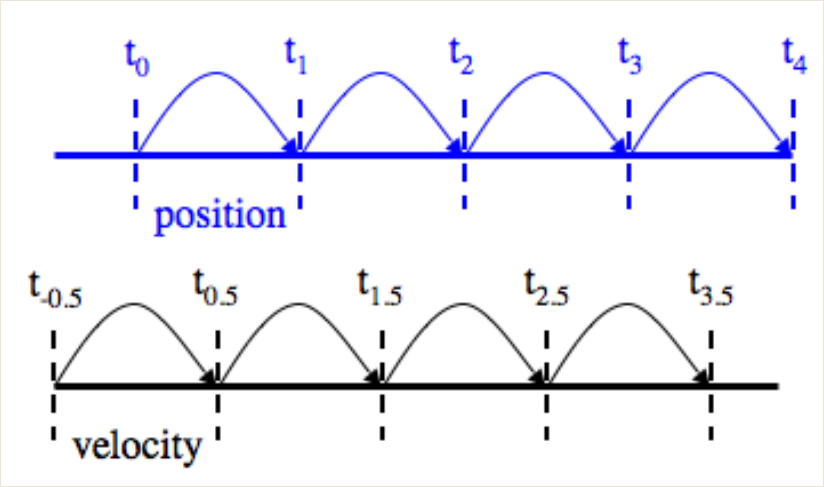
$$\hat{\phi}_{lmn} = G(\mathbf{k}_{lmn})\hat{\rho}_{lmn},$$

- where

$$G(\mathbf{k}) = -\frac{3\Omega_0}{8a}\left[\sin^2\left(\frac{k_x}{2}\right)+\sin^2\left(\frac{k_y}{2}\right)+\sin^2\left(\frac{k_z}{2}\right)\right]^{-1}$$

Step 3: Advance *x*

- Use simple second-order ("leapfrog") integration scheme

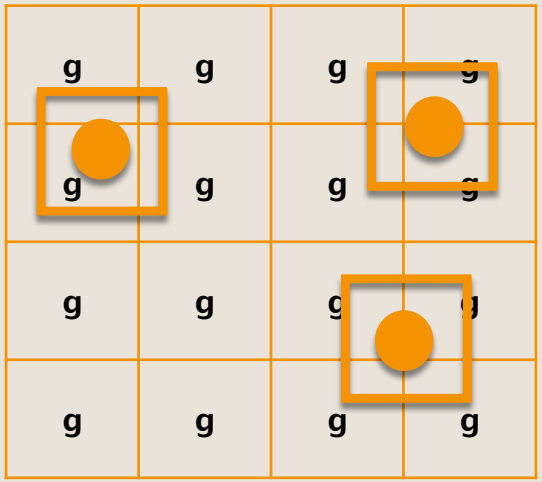


Step 4: Advance *v*

- Take acceleration as linear gradient of potential field

$$\tilde{\mathbf{g}}_n = -\tilde{\nabla}\tilde{\phi}_n$$

- Interpolate acceleration from neighbouring cells
- Use the SAME interpolation scheme as for density assignment to avoid self-forces



Part III

Testing a PM Code

Zeldovich approximation

- Separates spatial and temporal dependence of positions

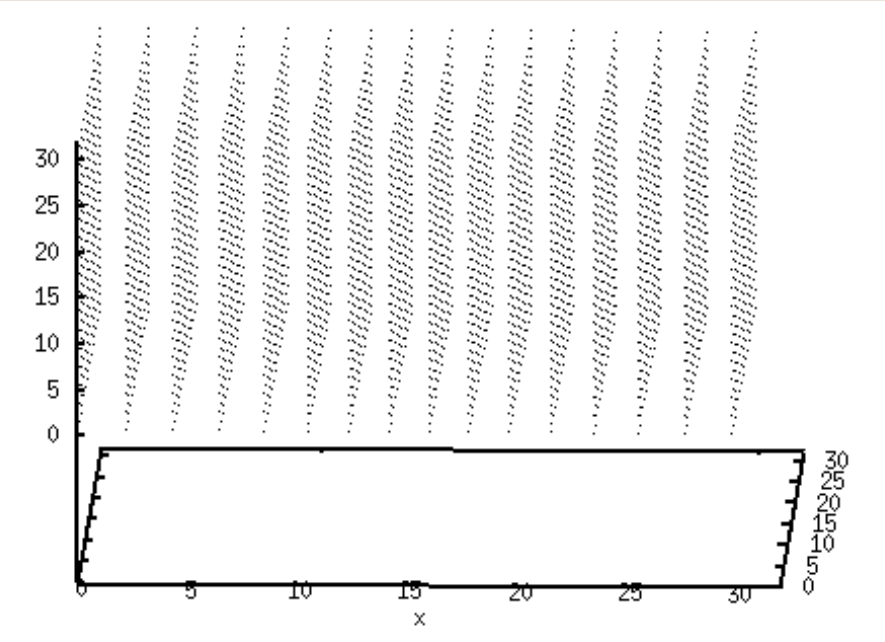
$$\mathbf{x}(t) = \mathbf{q} + D_+(t)\mathbf{S}(\mathbf{q}),$$

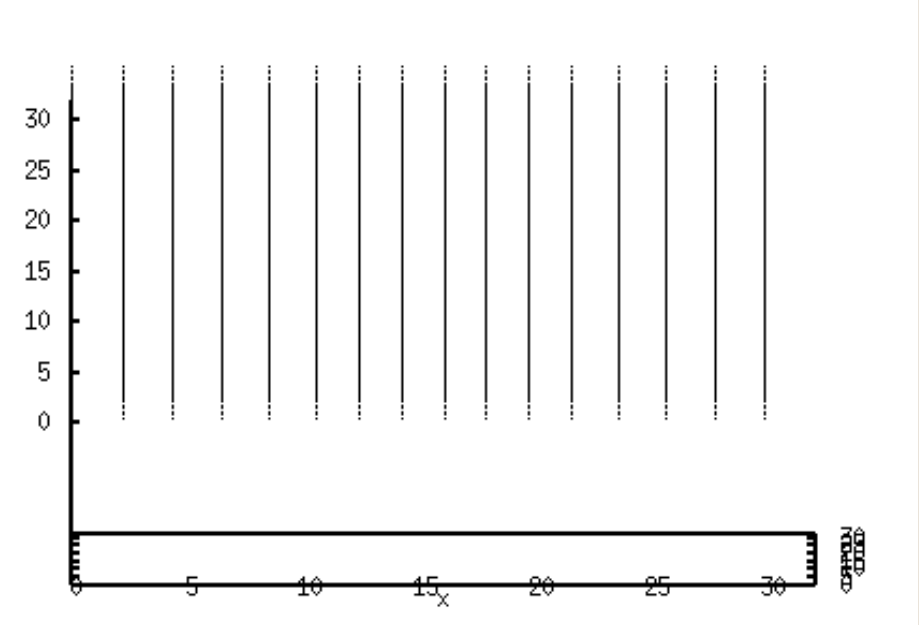
- Useful for code testing with a Zeldovich wave:

$$x(a) = q + D_+(a)A\sin(kq)$$

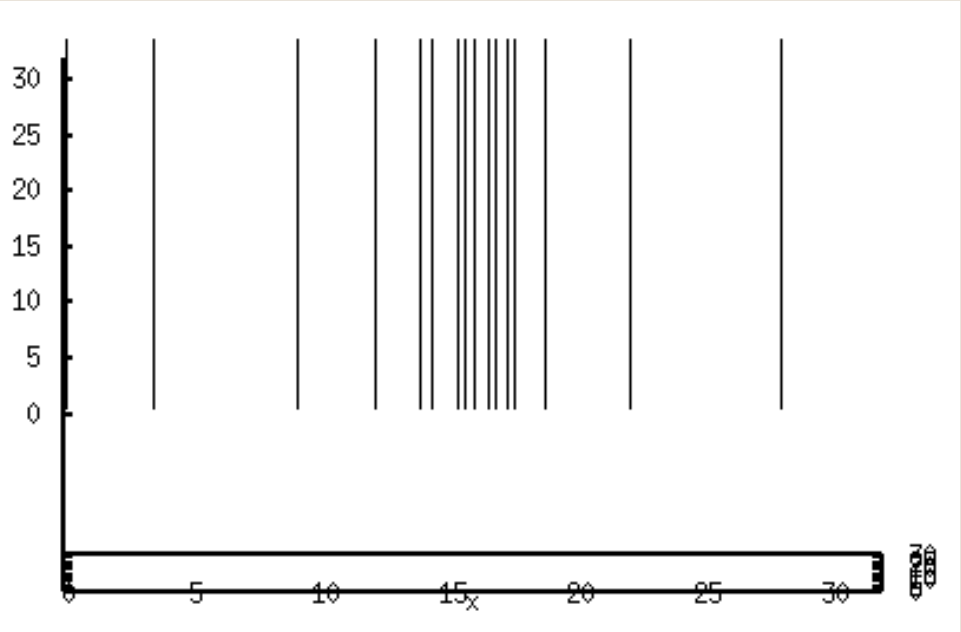
- In 1D, the solution is accurate until first crossing of trajectories
- D_+ is the growth function and depends on cosmology
- Simple case: $D_+ = a$ for $\Omega_m = 1$

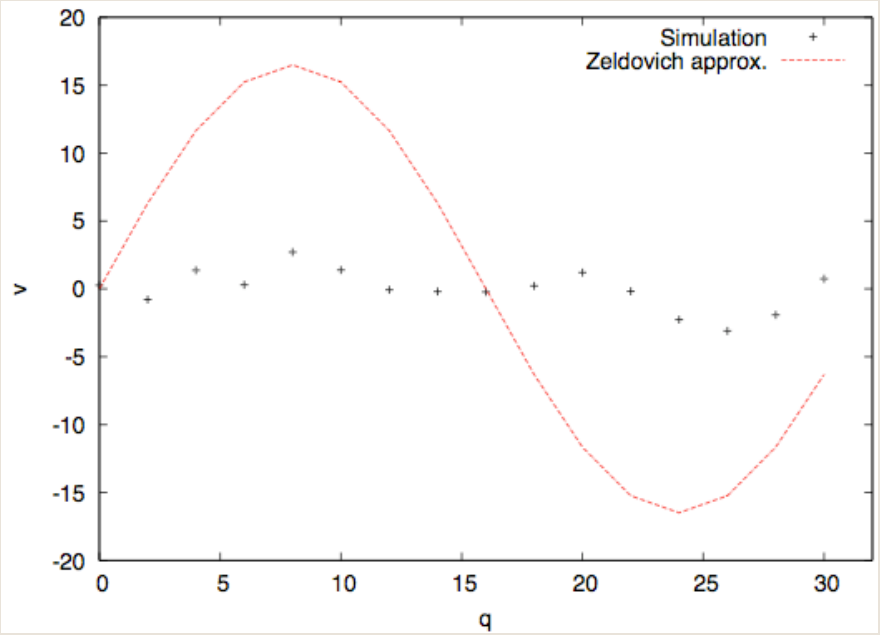
Zeldovich wave initial conditions



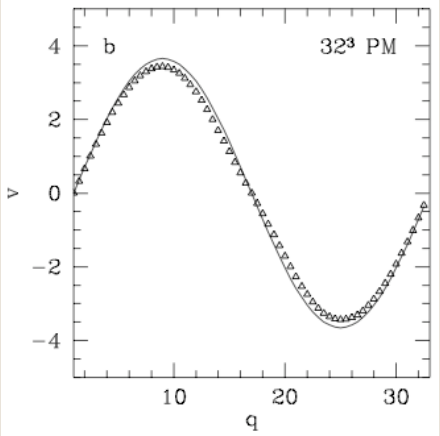


Zeldovich wave results

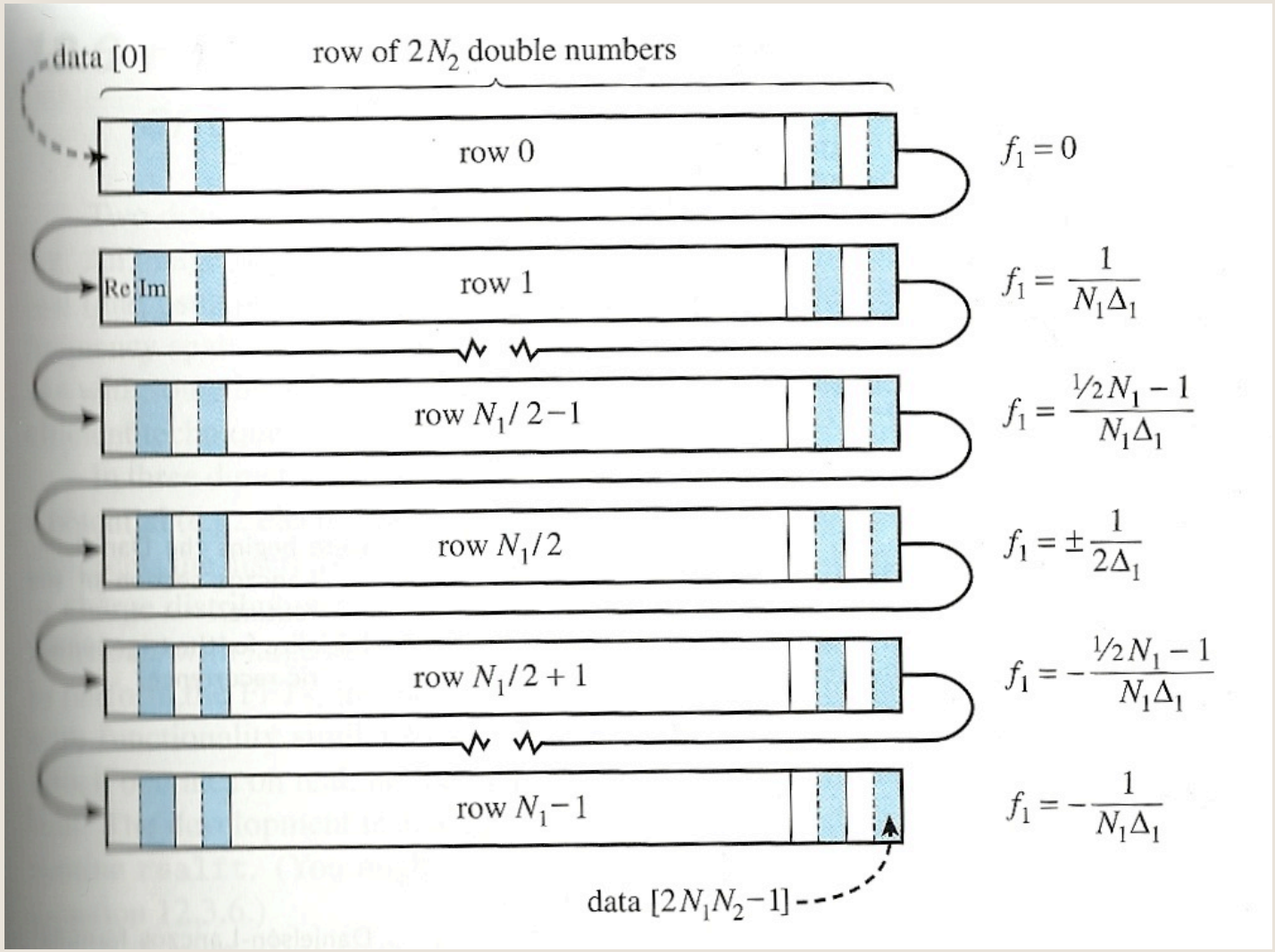




- Should look like this:



Where is the problem?



Where is the problem?

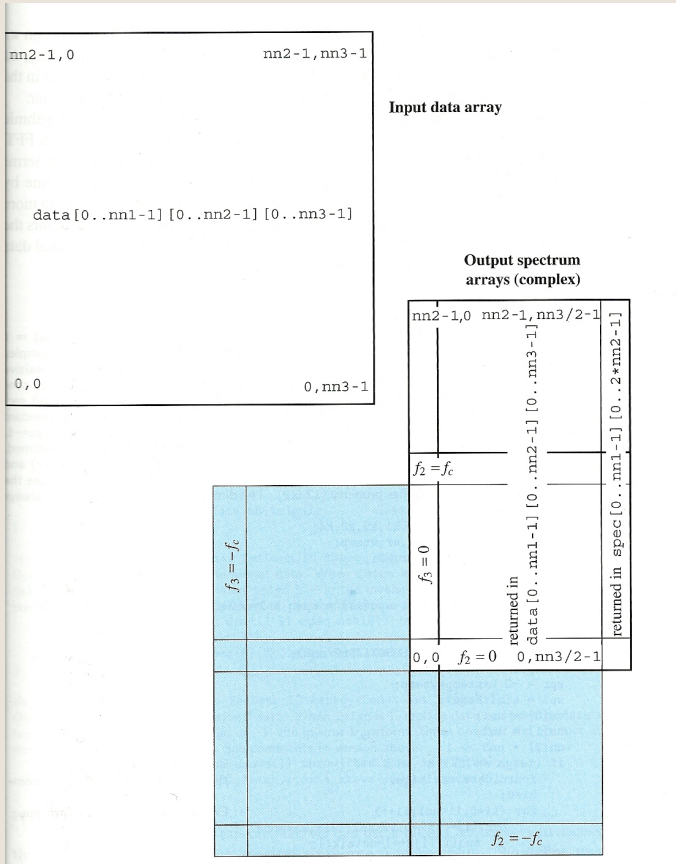


Figure 12.6.1. Input and output data arrangements for `rlft3`. All arrays shown are presumed to have a first (leftmost) dimension of range `[0..nn1-1]`, coming out of the page. The input data array is a real three-dimensional array `data[0..nn1-1][0..nn2-1][0..nn3-1]`. (For two-dimensional data, one sets `nn1 = 1`.) The output data can be viewed as a single complex array with dimensions `[0..nn1-1][0..nn2-1][0..nn3/2]` (cf. equation 12.6.3), corresponding to the frequency components f_1 and f_2 being stored in wraparound order, but only positive f_3 values being stored (others being obtainable by symmetry). The output data are actually returned mostly in the input array `data`, but partly stored in the real array `speq[0..nn1-1][0..2*nn2-1]`. See text for details.

- Very likely failed to multiply correct G(k) onto Fourier values
- Difficult to perform intermediate tests

Part IV

Cosmological simulations

Cosmological ICs

- Superimposed Zeldovich waves, randomized and weighted by power specrum P(k)

$$\mathbf{S}(\mathbf{q}) = \alpha \sum_{k_{x,y,z}=-k_{max}}^{k_{max}} i\mathbf{k}\, c_k \exp\left(i\mathbf{k}\cdot\mathbf{q}\right)$$

$$c_k = (a_k - ib_k)/2$$

$$a_k = \sqrt{P(k)}\,\frac{Gauss(0,1)}{k^2}, \quad b_k = \sqrt{P(k)}\,\frac{Gauss(0,1)}{k^2}.$$

$$\sigma^2 = P(k)/k^4$$

Conclusion

- Easiest way to implement a gravity solver
- Can be extended to P^3M (particle-particle interaction on small scales)
- Most modern N-Body simulations are done using tree codes (like GADGET)
- Planning to fix code using FFTW