Writing a particle-mesh N-Body code
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**

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
<tr>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
<td>$\rho, \phi$</td>
</tr>
</tbody>
</table>

Periodic boundary conditions

$m, x, v$
### Procedure

<table>
<thead>
<tr>
<th>Step</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute density field from particles</td>
<td>$N_p$</td>
</tr>
<tr>
<td>Solve for potential field</td>
<td>$N_c \log_2(N_c)$</td>
</tr>
<tr>
<td>Take gradient of potential as acceleration</td>
<td>$N_p$</td>
</tr>
<tr>
<td>Advance particle positions and velocities</td>
<td>$N_p$</td>
</tr>
</tbody>
</table>

**Order of all operations**

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

$$N^2 / 1000 \, N \log_2(N) = 700 \text{ times faster!}$$
Algorithms
Step 0: Dimensionless variables

- We can define dimensionless variables in which

\[
\begin{align*}
    r_0 &= \frac{L_{\text{box}}}{N_g}; \\
    N_g^3 &= \text{total number of grid cells} \\
    t_0 &= H^{-1}_0, \\
    v_0 &= \frac{r_0}{t_0}, \\
    \rho_0 &= \frac{3H_0^2}{8\pi G}\Omega_{m,0}, \\
    \phi_0 &= \frac{r_0^2}{t_0^2} = v_0^2. \\
\end{align*}
\]

\[
\begin{align*}
    \tilde{x} &= a^{-1} \frac{\mathbf{r}}{r_0}, \\
    \tilde{p} &= a \frac{\mathbf{v}}{v_0}, \\
    \tilde{\phi} &= \frac{\phi}{\phi_0}, \\
    \tilde{\rho} &= a^3 \frac{\rho}{\rho_0}, \\
    \phi &= \Phi + \frac{1}{2} a \ddot{a} \left( \frac{r}{a} \right)^2 = \Phi + \frac{H_0^2}{2} \left( \Omega_{\Lambda,0} - \frac{1}{2} a^{-3} \Omega_{m,0} \right) r^2.
\end{align*}
\]
Step 0: Dimensionless variables

\[ \tilde{\nabla}^2 \tilde{\phi} = \frac{3 \Omega_0}{2} \tilde{\delta}, \]

\[ \frac{d\tilde{P}}{da} = -f(a)\tilde{\nabla}\tilde{\phi}, \quad \frac{d\tilde{x}}{da} = f(a)\frac{\tilde{P}}{a^2}. \]

\[ \tilde{\delta} = \tilde{\rho} - 1 \]

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

- The only physical quantities are \( a \) and \( \Omega_x \)!
  - 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

\[
\tilde{\nabla}^2 \tilde{\phi} \approx \tilde{\phi}_{i-1,j,k} + \tilde{\phi}_{i+1,j,k} + \tilde{\phi}_{i,j-1,k} + \tilde{\phi}_{i,j+1,k} + \tilde{\phi}_{i,j,k-1} + \tilde{\phi}_{i,j,k+1} - 6\tilde{\phi}_{i,j,k} = \\
= \frac{3\Omega_0}{2a}(\tilde{\rho}_{i,j,k} - 1).
\]

- These \(N^3\) equations are solved in Fourier space by

\[
\hat{\phi}_{lmn} = G(k_{lmn})\hat{\rho}_{lmn}.
\]

- where

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

- Use simple second-order ("leapfrog") integration scheme
Step 4: Advance \( \nu \)

- Take acceleration as linear gradient of potential field
  \[ \tilde{g}_n = -\nabla \phi_n \]

- 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

\[ x(t) = q + D_+(t)S(q), \]

- 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?

- 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 spectrum $P(k)$

$$S(q) = \alpha \sum_{k_{x,y,z}=-k_{max}}^{k_{max}} ik \, c_k \exp (ik \cdot q)$$

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

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

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