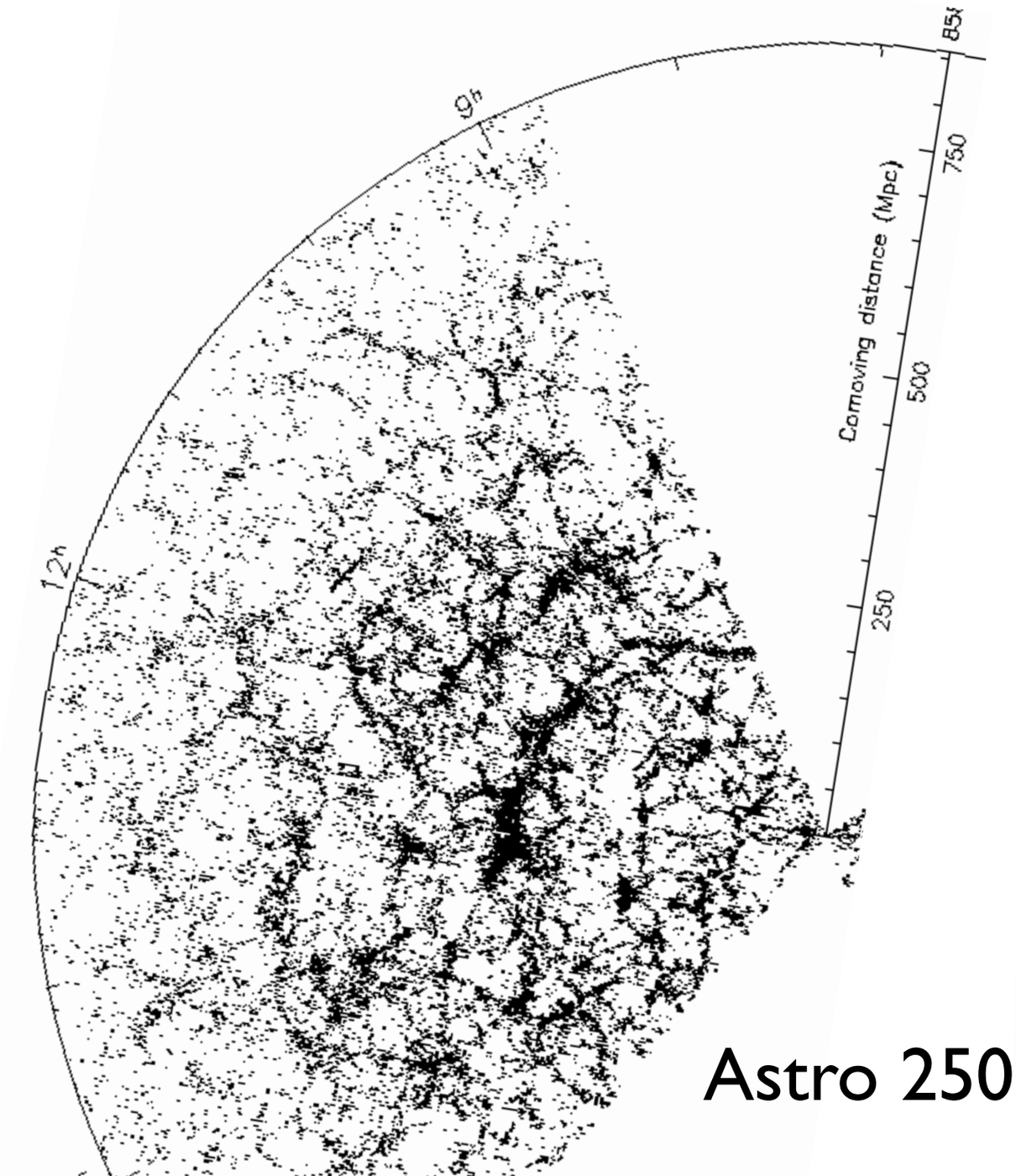
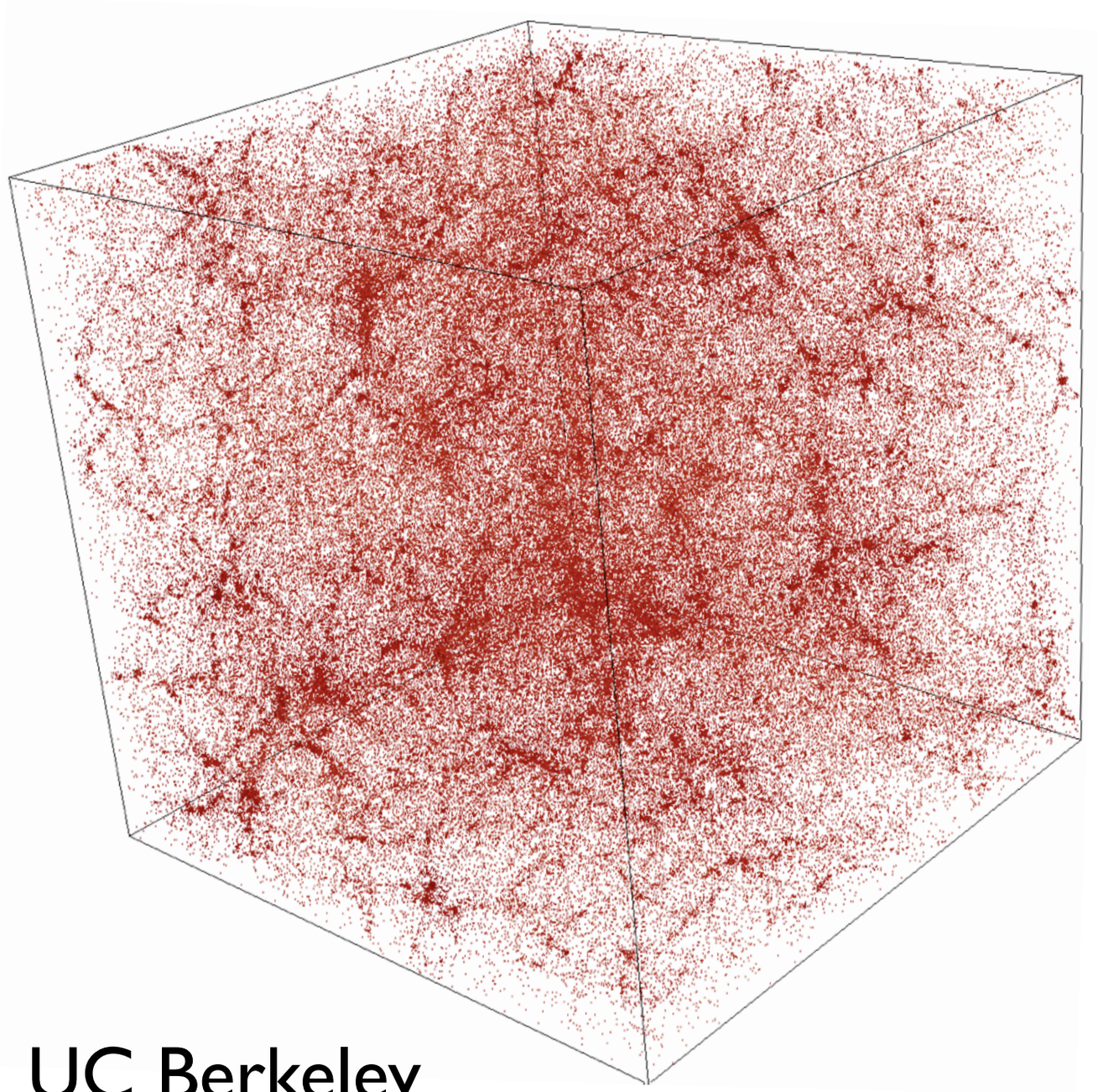
Cosmology with N-body (+ Gadget)

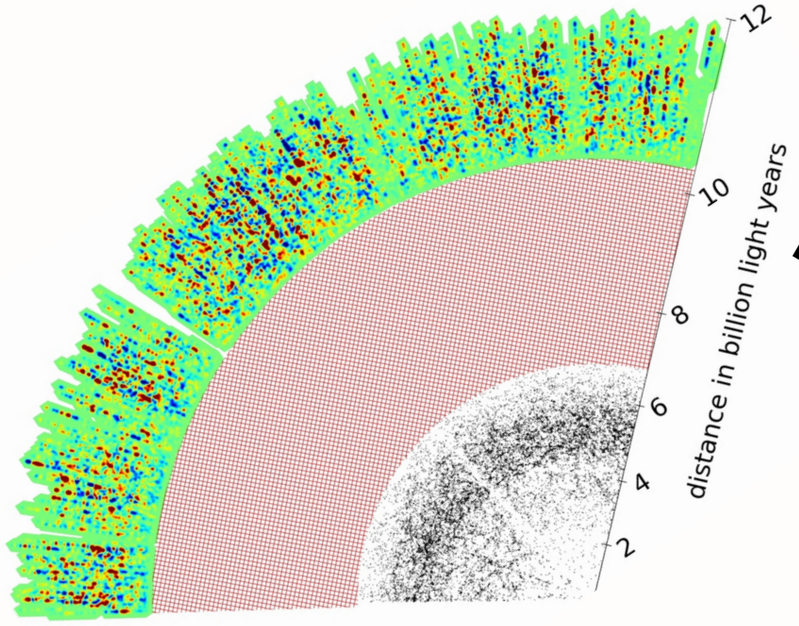
Zarija Lukić (LBL)



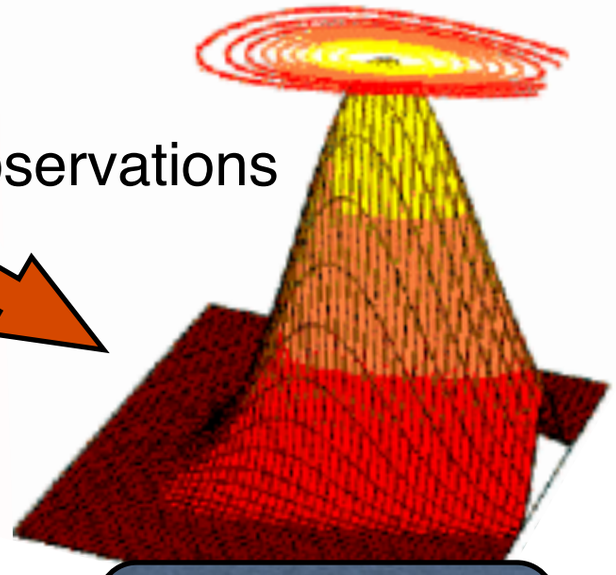


Astro 250, UC Berkeley

Explaining the universe



Observations

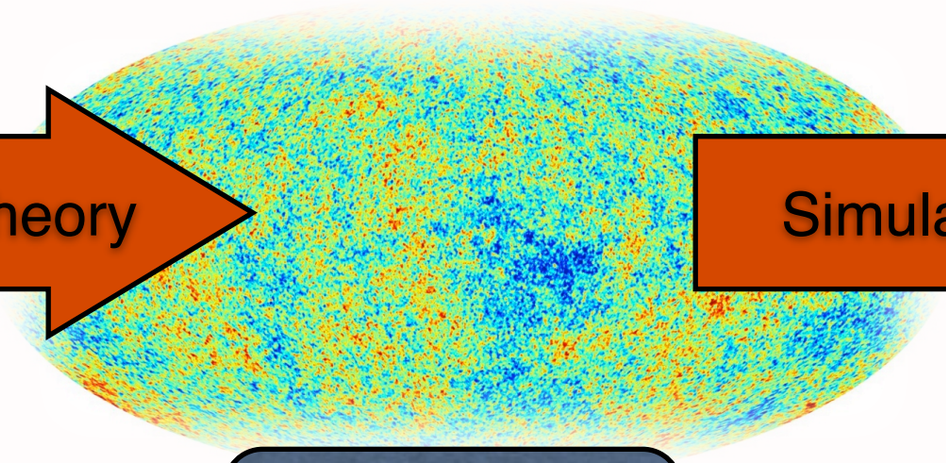


Likelihood analysis

Theoretical predictions

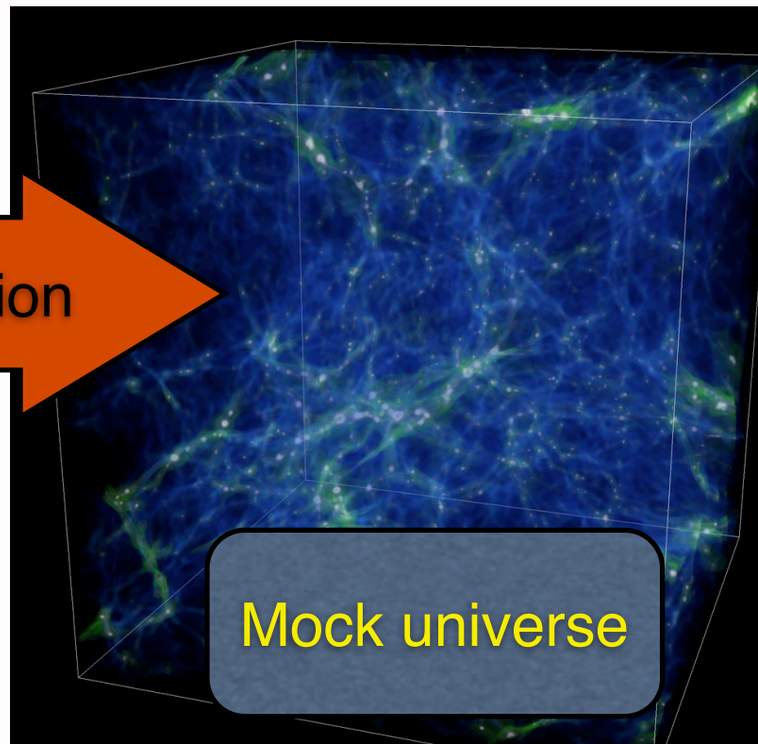
Cosmology model

Linear theory



Initial conditions

Simulation



Mock universe

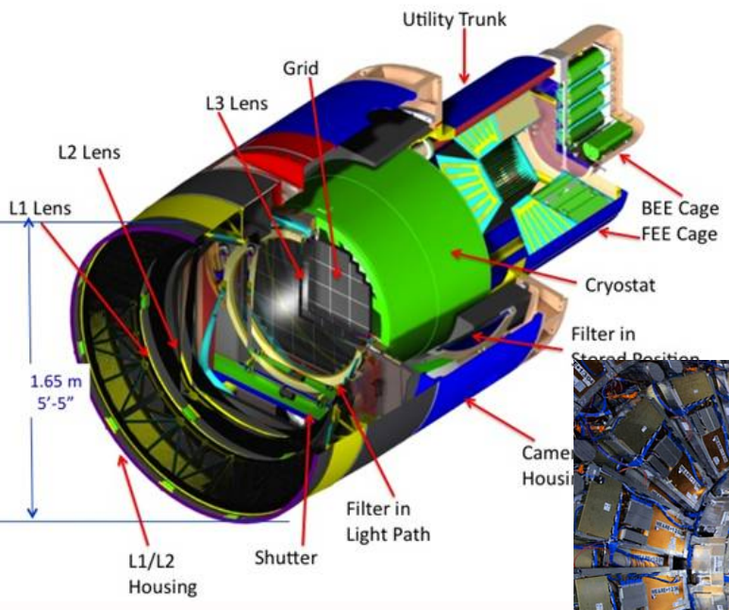
Computational discovery

BERKELEY LAB

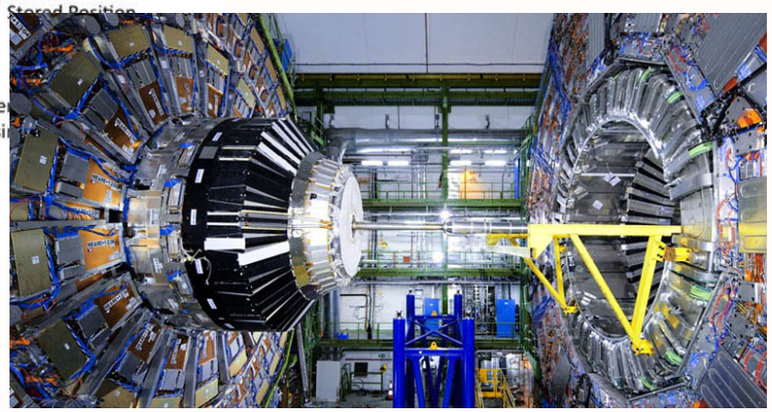
Telescopes, accelerators

Simulation codes, computers

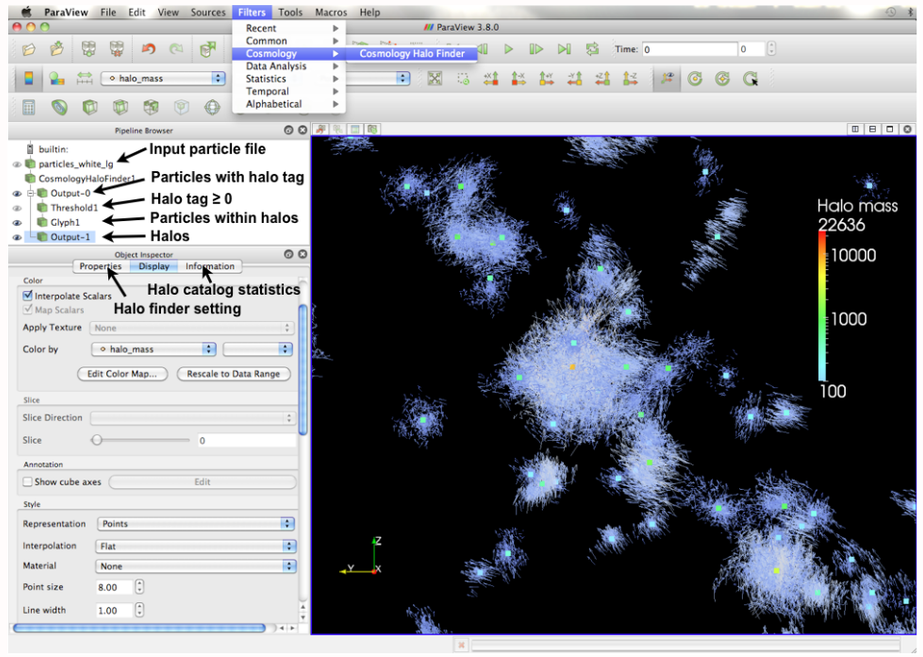

```
int
main (int argc, char* argv[])
{
    BoxLib::Initialize(argc, argv);


    // save the inputs file name for later
    if (argc > 1) {
      if (!strchr(argv[1], '=')) {
        inputs_name = argv[1];
      }
    }
    BL_PROFILE_REGION_START("main()");
    BL_PROFILE_VAR("main()", pmain);

    //
    // Don't start timing until all CPUs are ready to go.
    //
    ParallelDescriptor::Barrier("Starting main.");

    BL_COMM_PROFILE_NAMETAG("main TOP");

#ifdef BL_USE_MPI
    const int MPI_IntraGroup_Broadcast_Rank = ParallelDescriptor::IOProcessor() ? MPI_ROOT : MPI_PROC_NULL;
    int nSidecarProcsFromParmParse(-3);
    Nyx::nSidecarProcs = 0;
    int prevSidecarProcs(0);
    int sidecarSignal(NyxHaloFinderSignal);
    int resizeSidecars(false);  // ---- instead of bool for bcast
#endif
```



Cameras, detectors



Analysis software

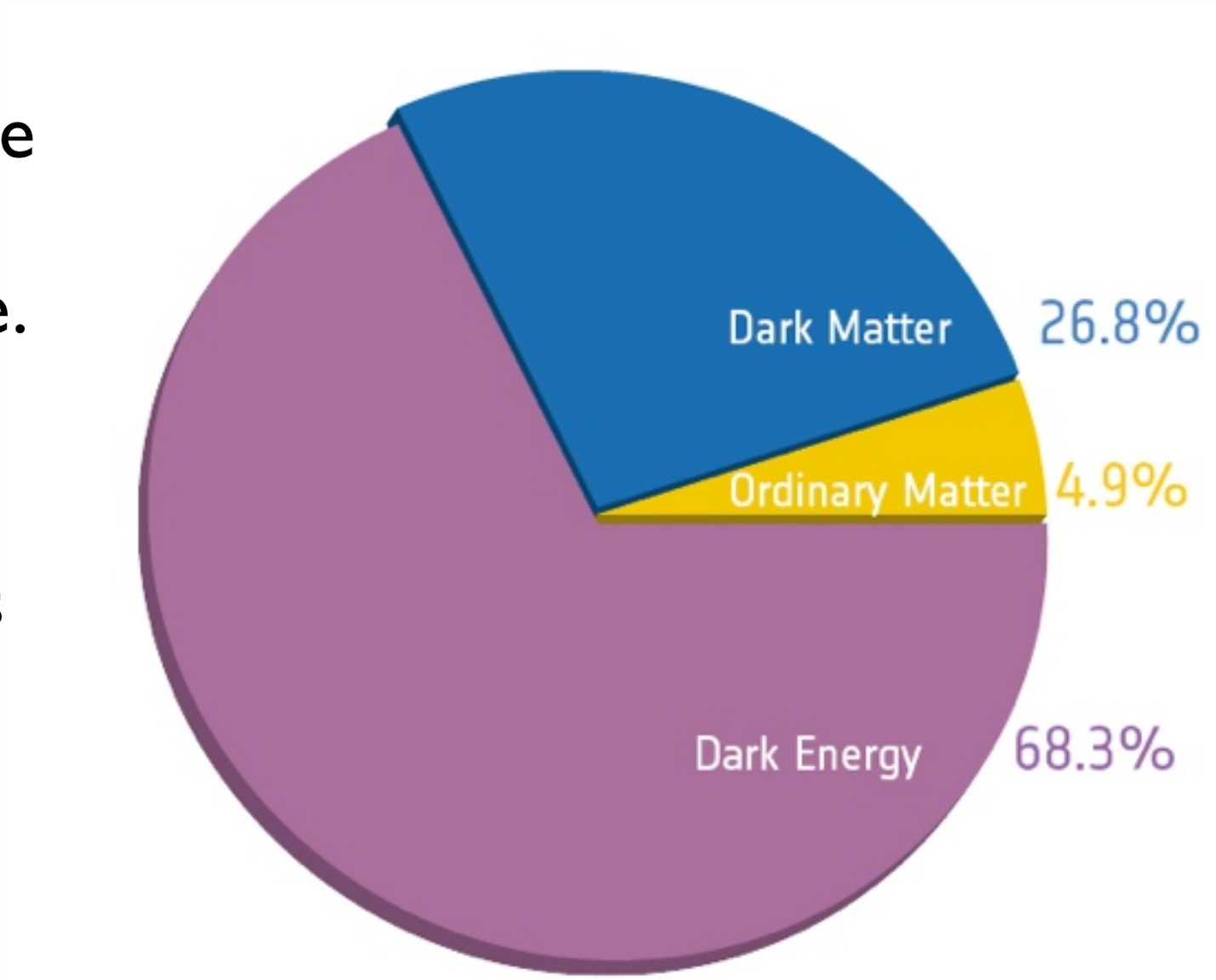


The cosmic pie

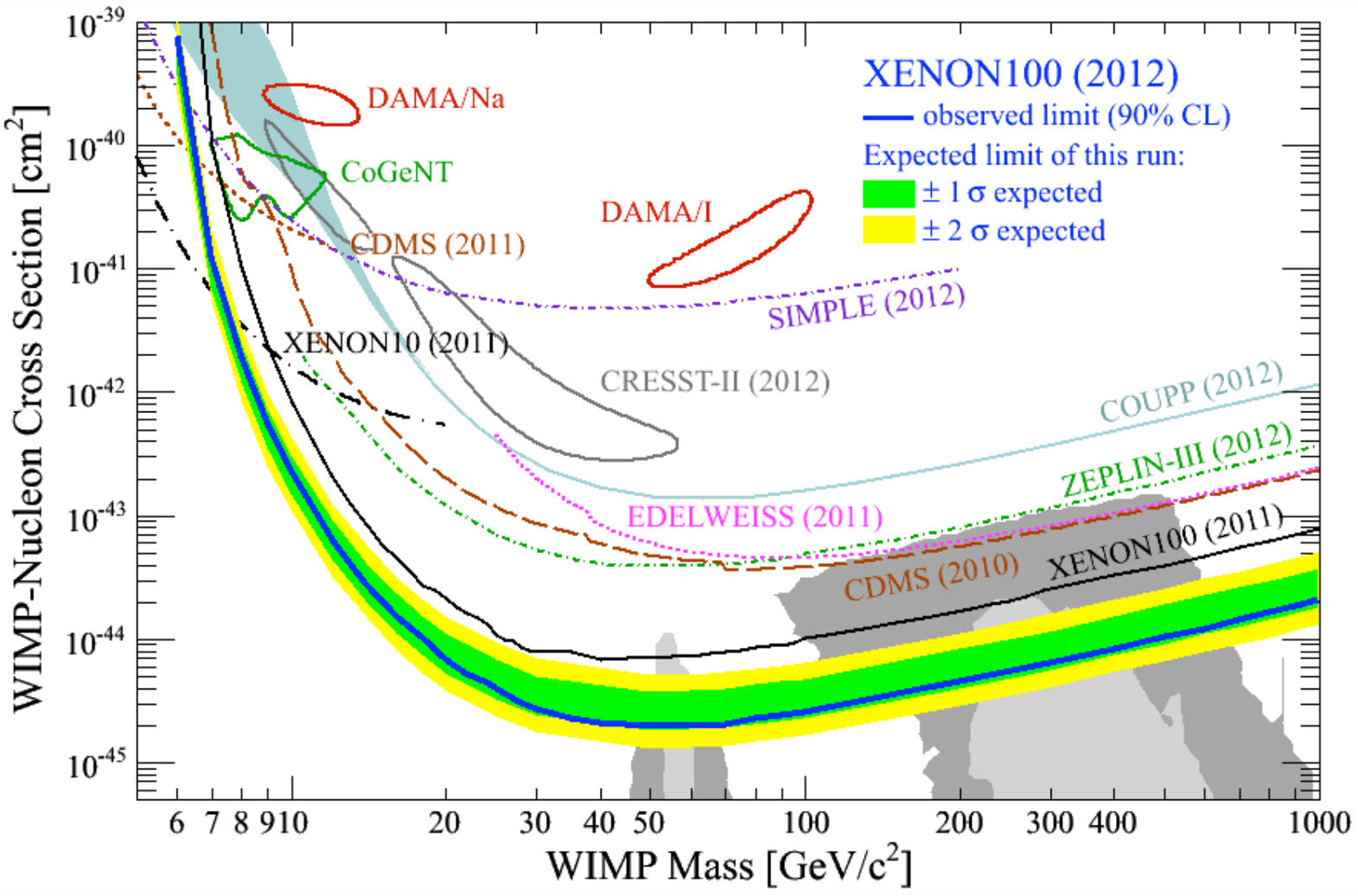
Dark energy is responsible for the accelerated expansion of the universe.

Dark matter has been detected indirectly by its gravitational influence.

Ordinary matter makes stars and galaxies…



Dark matter



$$\sigma \approx 10^{-42} cm^2$$

$$m_{DM} \approx 10^2 GeV \approx 10^{-22} g$$

$$\rho_{crit} \approx 10^{-30} \frac{g}{cm^3}$$

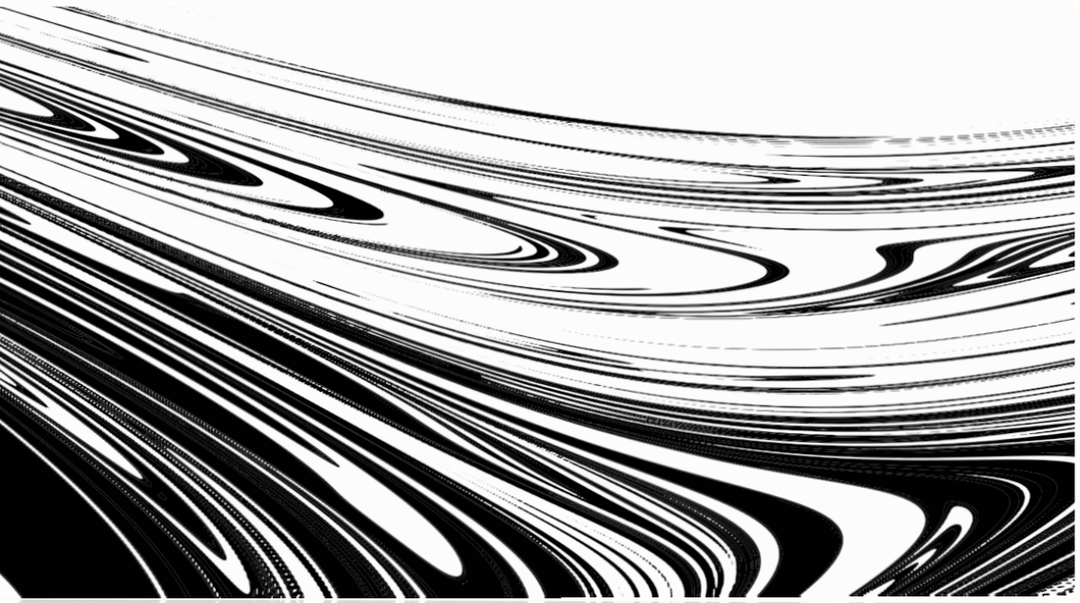
$$\rho_{crit} = \frac{N m_{DM}}{V} = n\, m_{DM}$$

$$n \approx 10^{-8} \frac{1}{cm^3}$$

$$\lambda = \frac{1}{n\sigma} \approx \frac{1}{10^{-8} 10^{-42}} cm = 10^{50} cm \approx 10^{30} Mpc$$

Dust model

- Collisionless fluid evolving under self-gravity



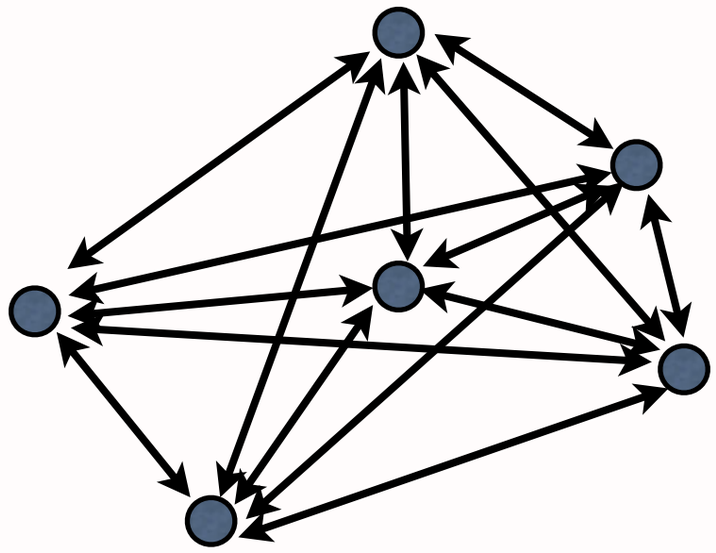
$$\frac{\partial f}{\partial t} + \frac{1}{ma^2}\mathbf{p}\cdot\nabla f - m\nabla\phi\cdot\frac{\partial f}{\partial \mathbf{p}} = 0$$

$$\nabla^2\phi = \frac{4\pi G}{a}(\rho_{tot} - \rho_0)$$

- Solve as N-body problem

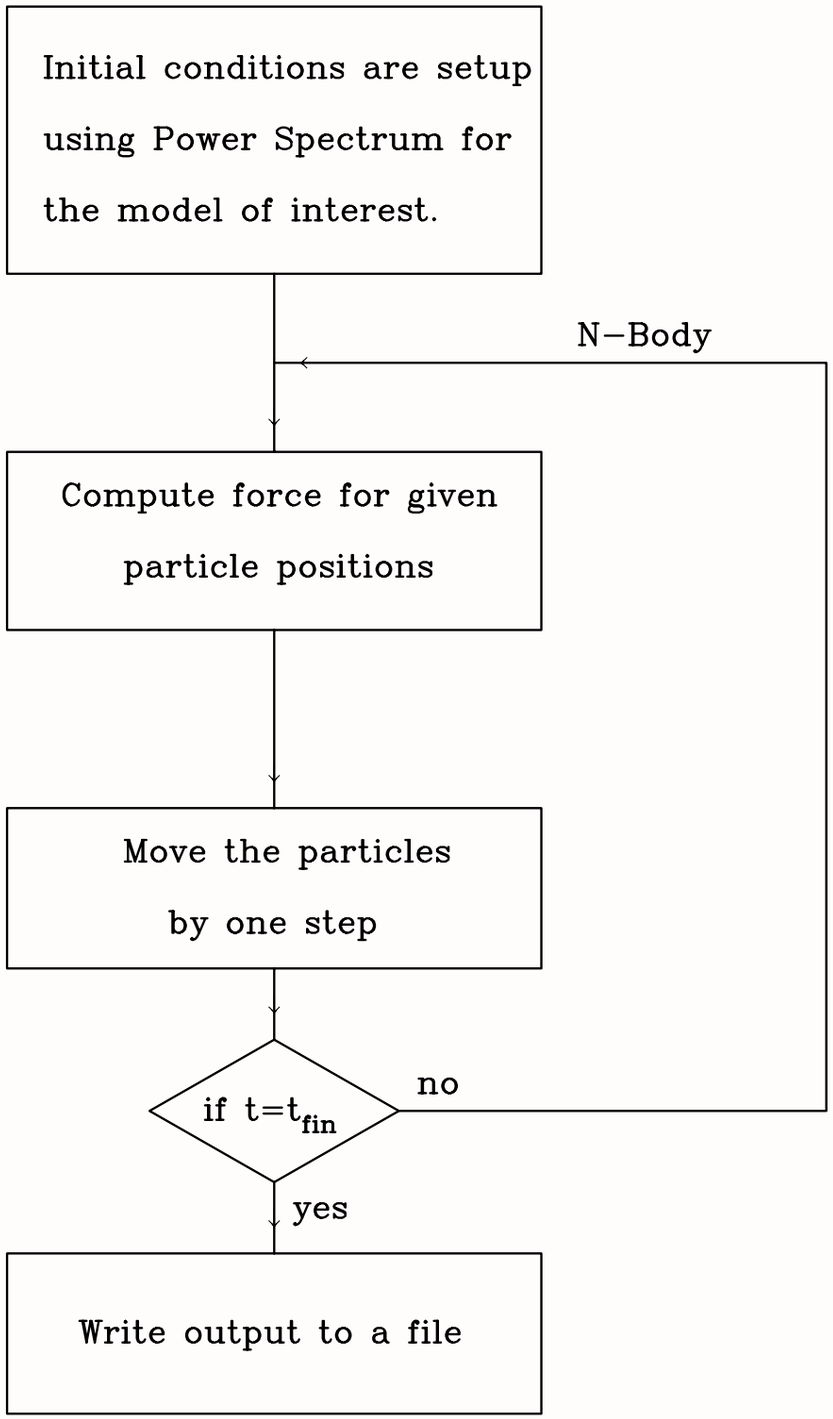
$$\frac{d\mathbf{x}_i}{dt} = \frac{1}{a}\mathbf{u}_i$$

$$\frac{d(a\mathbf{u}_i)}{dt} = \mathbf{g}_i$$



$O(N^2)$ scaling

N-body algorithm



Two basic modules:

1. Compute the force field for a given configuration of particles

2. Move the particles in this force field

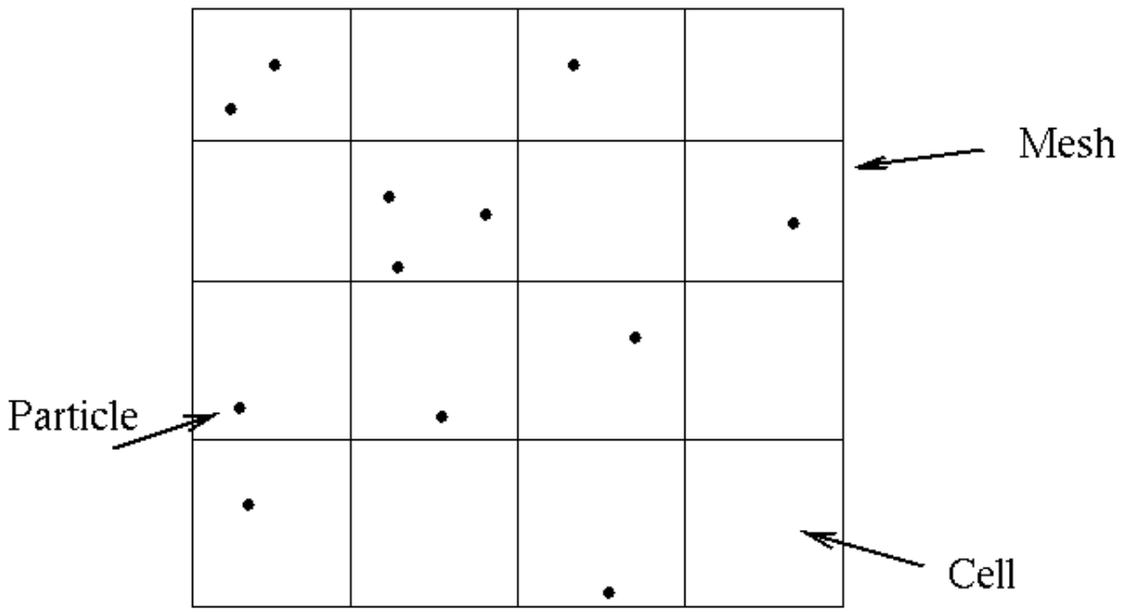
In addition, we have to setup the initial conditions and write the output

Particle-Mesh (PM)

- Particle mass is deposited on a grid to get density
- Poisson equation is solved on a grid
- Forces are interpolated back on the particles

Advantages:

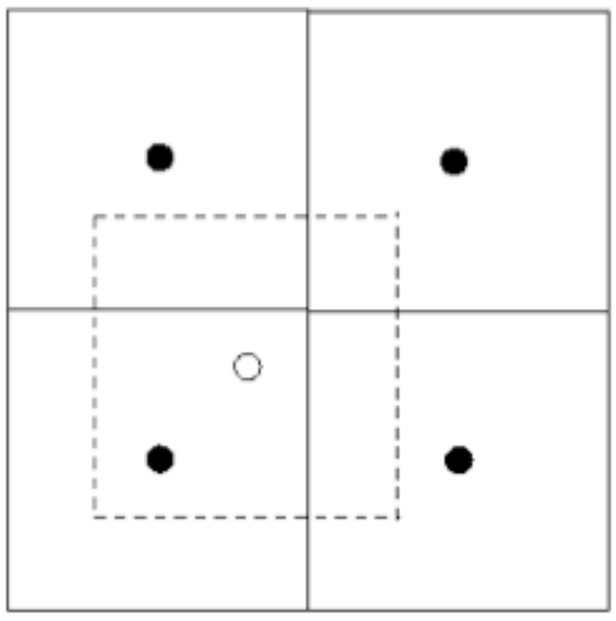
1. Exceptionally fast
2. Good parallel scaling
3. Easy implementation of periodic boundaries
4. Same efficiency regardless of the particle distribution



Deposition

- Particle mass is deposited on a grid to get density
- Forces are interpolated back on the particles

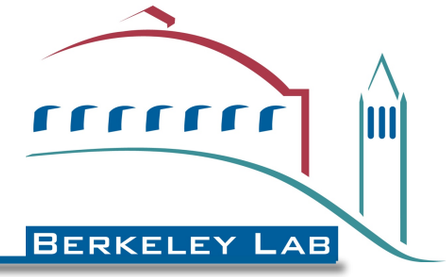
For example, Cloud in Cell (CIC):



$$\rho_i = \frac{1}{\Delta x} \sum_{p=1}^{N_p} m_p W^{\rho}(x_p - x_i)$$

$$F(x_p) = m_p \sum W^{F}(x_p - x_i) F_i$$

Poisson on a grid

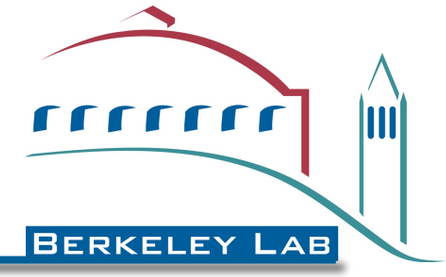


- Many ways to solve Poisson equation on a grid:
  - If we deal with an uni-grid, spectral solver is the fastest
  - In AMR setting, multigrid is the best bet

$$\Phi(\mathbf{x}) = \int g(\mathbf{x} - \mathbf{x}')\, \rho(\mathbf{x})\, \mathrm{d}\mathbf{x}'$$

$$\hat{\Phi}(\mathbf{k}) = \hat{g}(\mathbf{k}) \cdot \hat{\rho}(\mathbf{k})$$

Poisson on a grid



- Many ways to solve Poisson equation on a grid:
  - If we deal with an uni-grid, spectral solver is the fastest
  - In AMR setting, multigrid is the best bet

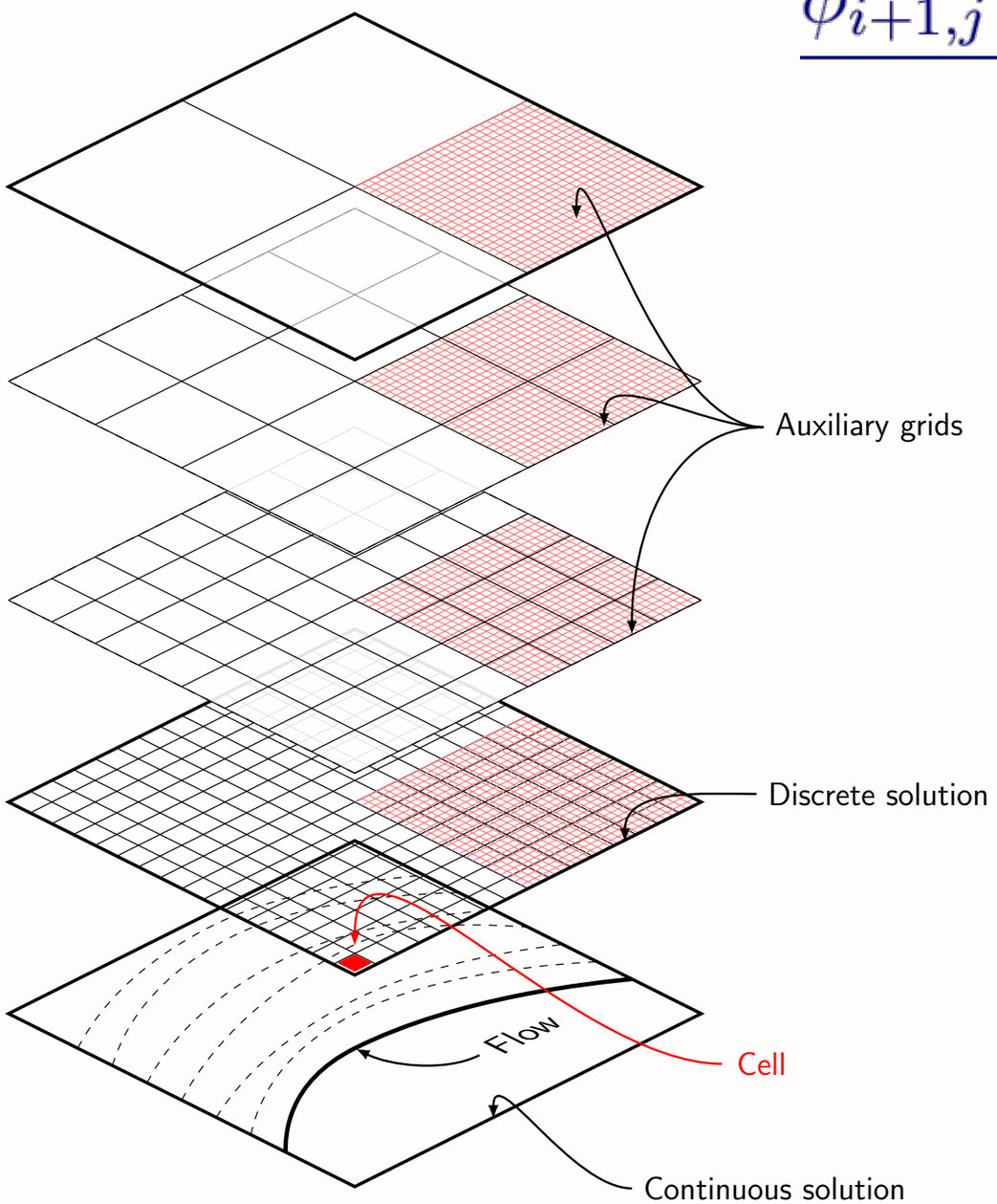
$$\nabla^2\phi = \frac{4\pi G}{a}(\rho_{tot} - \rho_0)$$

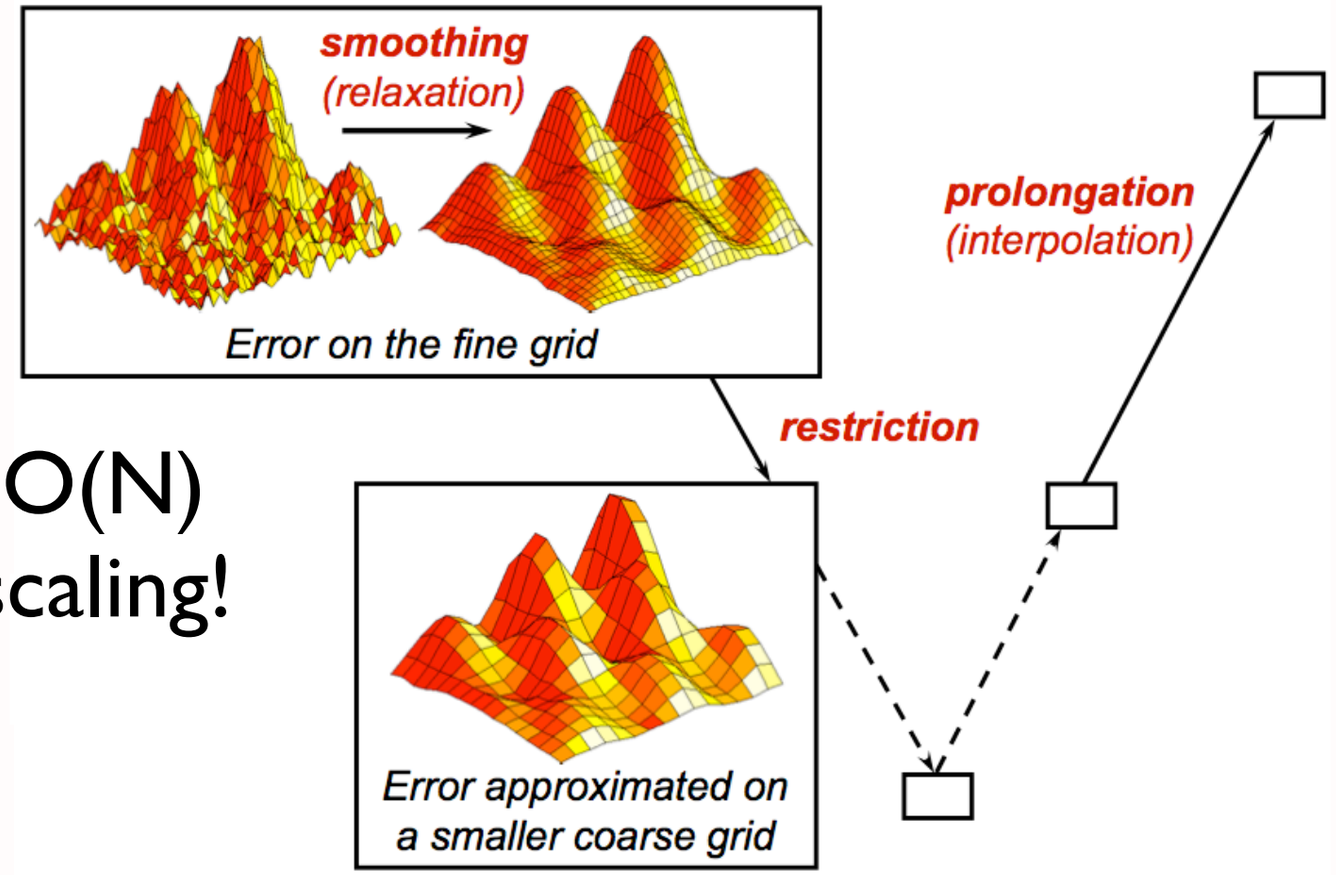
$$\frac{\phi_{i+1,j} + \phi_{i,j+1} - 4\phi_{i,j} + \phi_{i-1,j} + \phi_{i,j-1}}{\Delta x^2} = \rho_{i,j}$$

Multigrid

- Deposit mass on a grid, and solve linear system:

$$\frac{\phi_{i+1,j} + \phi_{i,j+1} - 4\phi_{i,j} + \phi_{i-1,j} + \phi_{i,j-1}}{\Delta x^2} = \rho_{i,j}$$

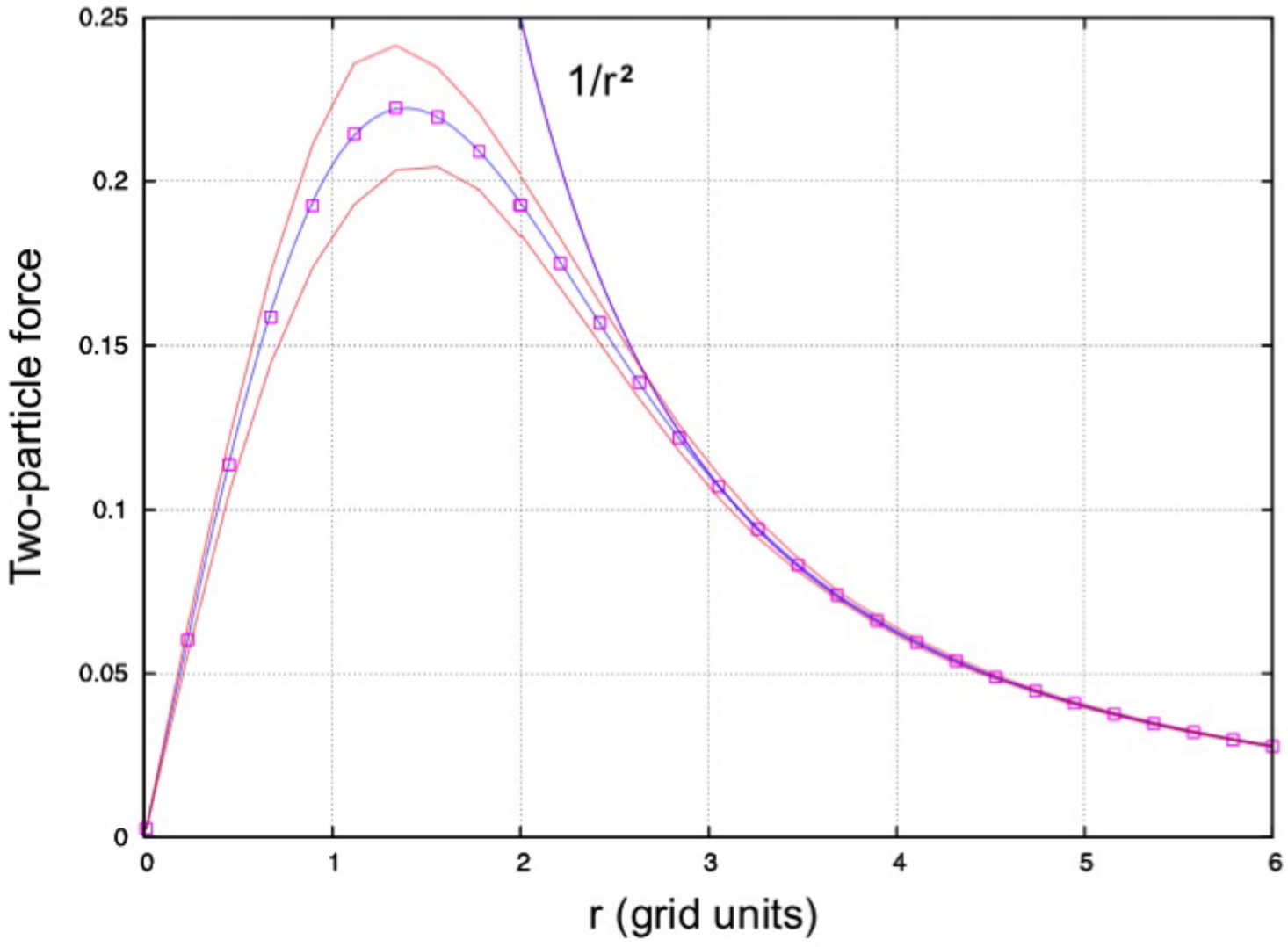




O(N) scaling!

Excellent engine for non-linear Poisson equation

PM accuracy

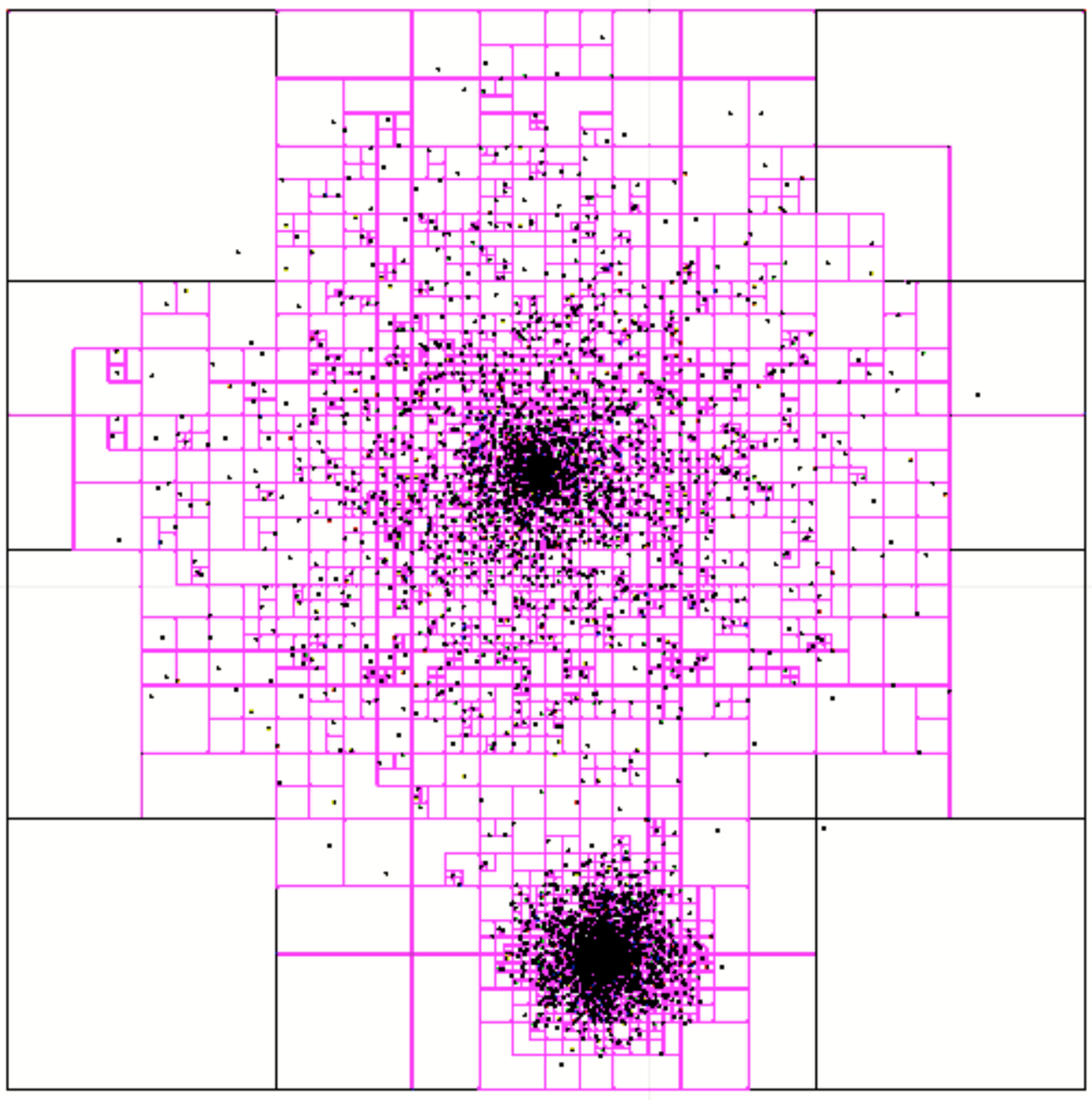


Force anisotropy noise is shown by the bracketing orange lines representing the 1-σ deviations.

Resolution is a serious drawback. Without adaptive refinements, memory requirements goes as N_g^3.

Tree method

- Another way to avoid direct summation of forces (N^2) is to use Tree algorithm (Nlog(N) complexity).



For example, Barnes-Hut:

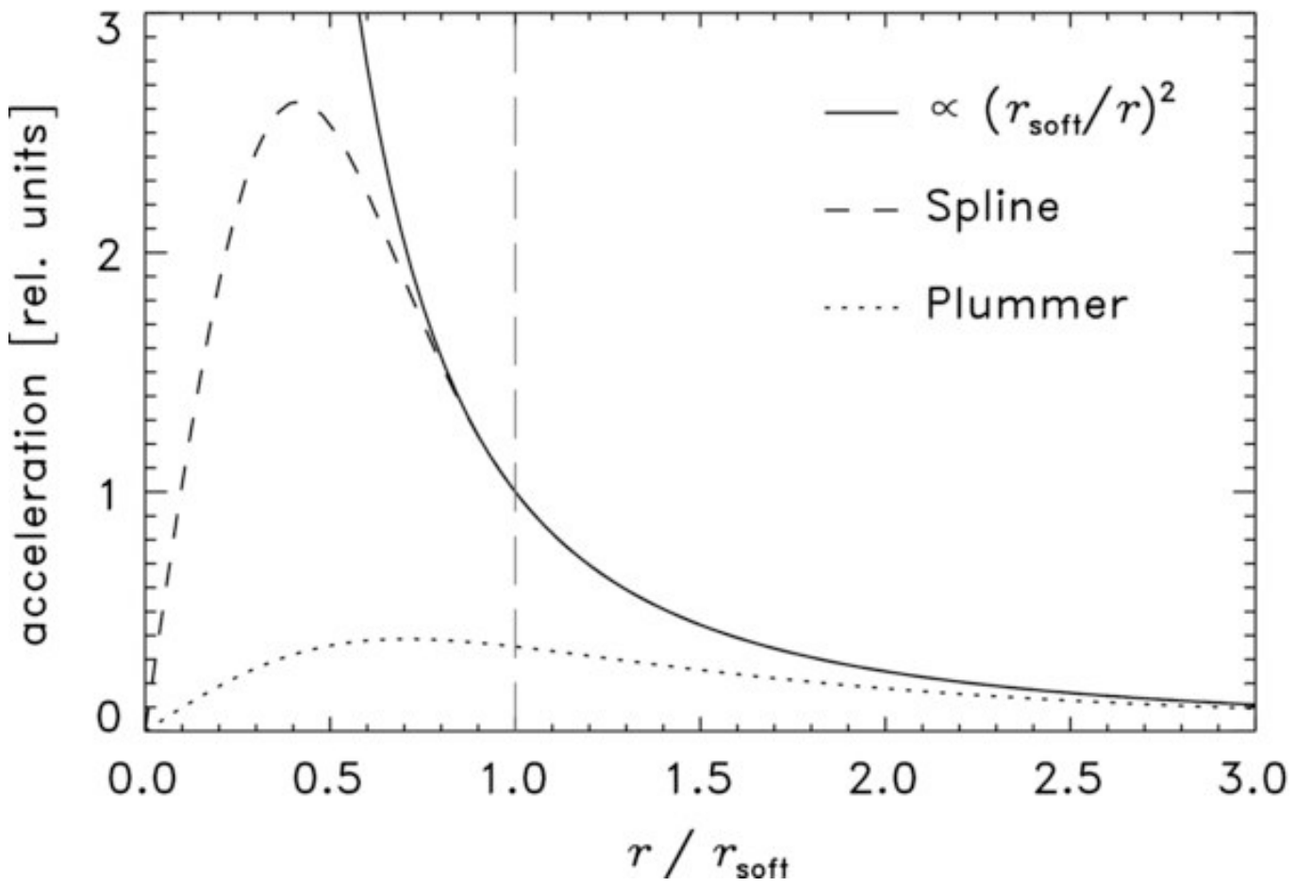
- Assemble all particles on an oct tree.
- "Leafs" contain 0 or 1 particle.
- Nodes store total mass and center of mass info.
- Traverse the tree, if the center of mass is far enough, use it to calculate the force.

Softening

- Prevents particle scattering and formation of tight binary systems.
- Makes two-body relaxation time large.
- Speeds-up time integration.

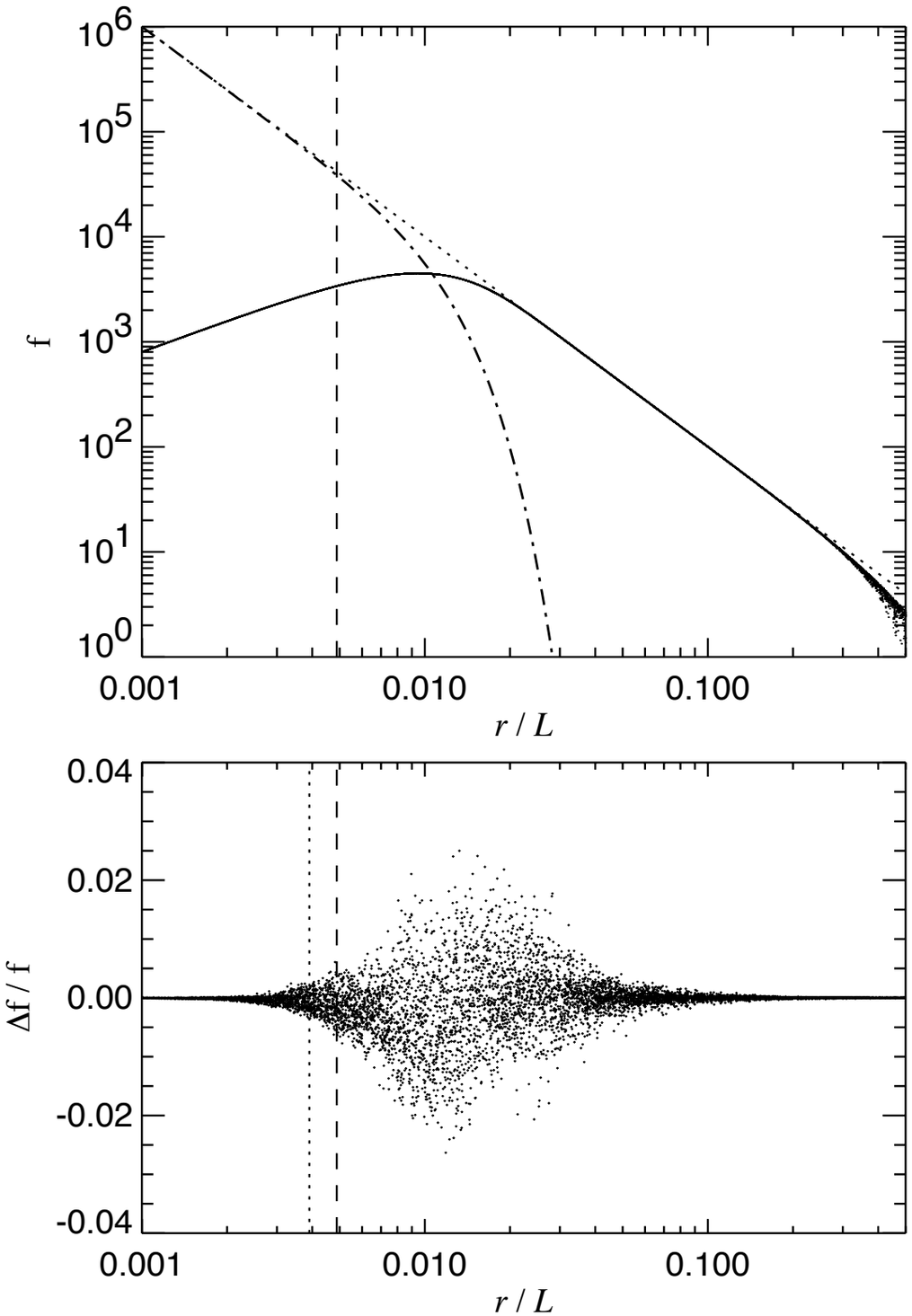
$$\ddot{\mathbf{x}}_i = -\nabla_i \Phi(\mathbf{x}_i)$$

$$\Phi(\mathbf{x}) = -G \sum_{j=1}^{N} \frac{m_j}{[(\mathbf{x}-\mathbf{x}_j)^2 + \epsilon^2]^{1/2}}$$

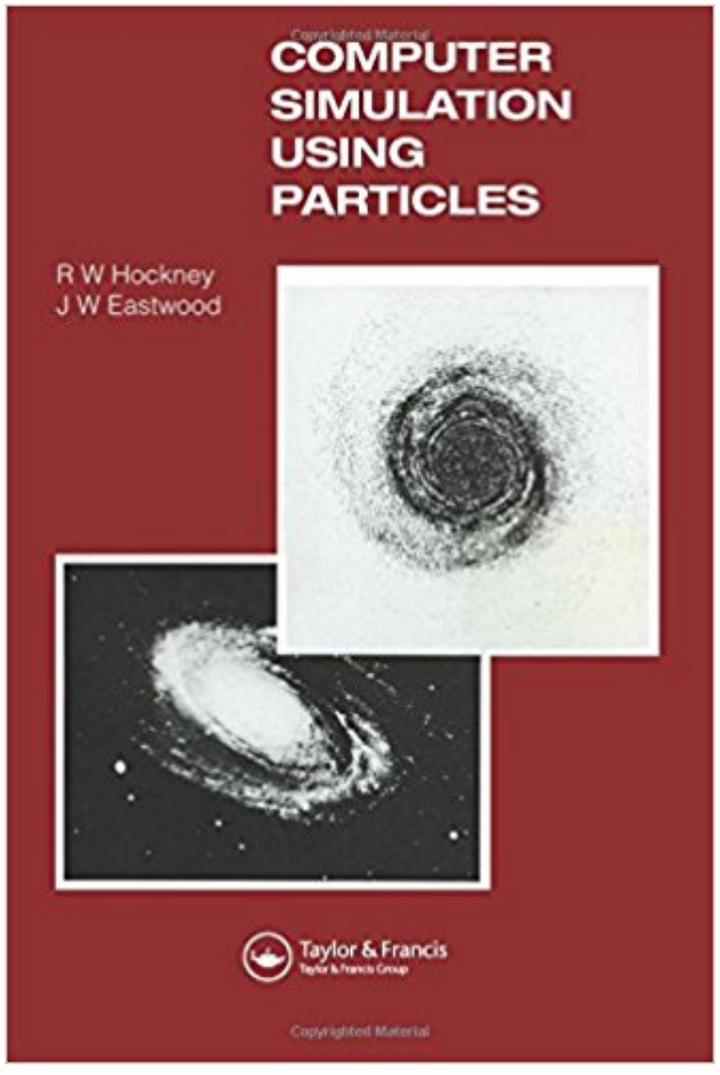


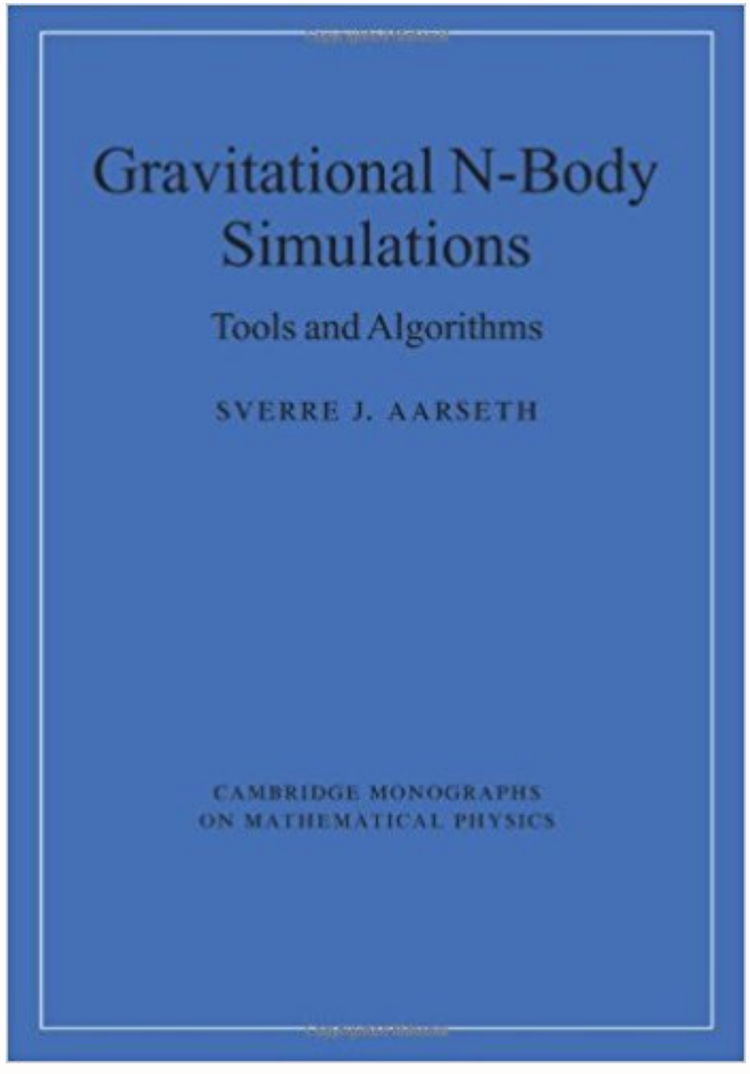
Gadget-2 (TreePM)

- Spectral solve for the long-range force
- Tree method for the short-range force
- Individual and adaptive timesteps
- Leapfrog time integration
- 1D grid decomposition
- Force matching is tricky



Resources



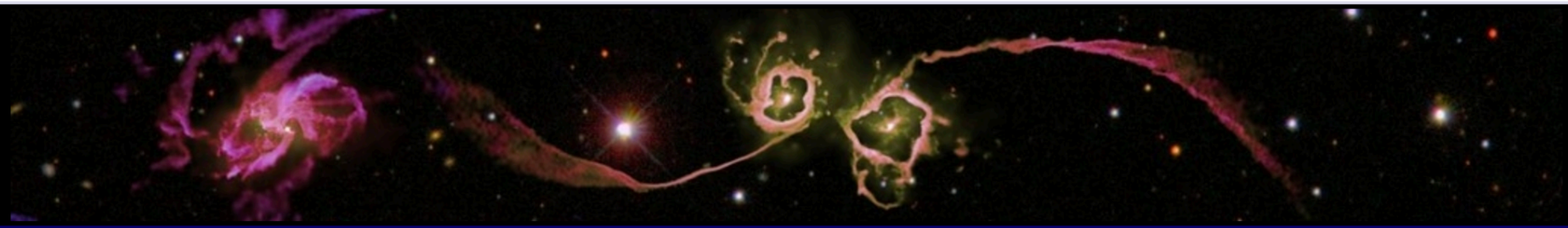


- Many code & review papers
- Google is your friend too

Hands on Gadget-2

BERKELEY LAB

www.gadgetcode.org



GADGET - 2

A code for cosmological simulations of structure formation

Description

GADGET is a freely available code for cosmological N-body/SPH simulations on massively parallel computers with distributed memory. **GADGET** uses an explicit communication model that is implemented with the standardized MPI communication interface. The code can be run on essentially all supercomputer systems presently in use, including clusters of workstations or individual PCs.

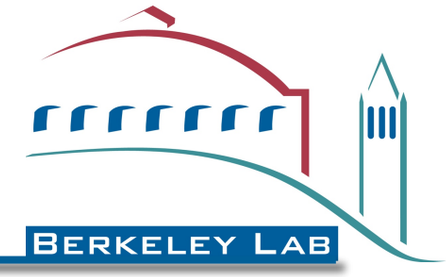
GADGET computes gravitational forces with a hierarchical tree algorithm (optionally in combination with a particle-mesh scheme for long-range gravitational forces) and represents fluids by means of smoothed particle hydrodynamics (SPH). The code can be used for studies of isolated systems, or for simulations that include the cosmological expansion of space, both with or without periodic boundary conditions. In all these types of simulations, **GADGET** follows the evolution of a self-gravitating collisionless N-body system, and allows gas dynamics to be optionally included. Both the force computation and the time stepping of **GADGET** are fully adaptive, with a dynamic range which is, in principle, unlimited.

GADGET can therefore be used to address a wide array of astrophysically interesting problems, ranging from colliding and merging galaxies, to the formation of large-scale structure in the Universe. With the inclusion of additional physical processes such as radiative cooling and heating, **GADGET** can also be used to study the dynamics of the gaseous intergalactic medium, or to address star formation and its regulation by feedback processes.

Features

- Hierarchical multipole expansion (based on a geometrical oct-tree) for gravitational forces.
- Optional TreePM method, where the tree is used for short-range gravitational forces only while long-range forces are computed with a FFT-based particle-mesh (PM) scheme. A second PM layer can be placed on a high-resolution region in 'zoom'-simulations.
- Periodic boundary conditions, either by means of the Ewald summation technique or based on the FFT algorithm used in the TreePM scheme. Simulations that only follow gas dynamics without self-gravity can be run in periodic boxes with arbitrary aspect ratios, and also in 2D, if desired.

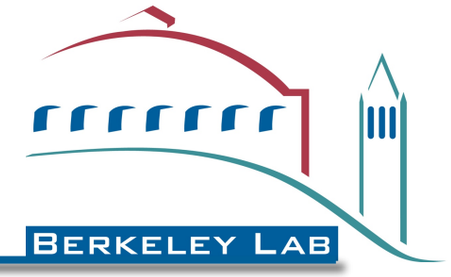
Gadget requirements



- Compiler suite (e.g. gnu)
- Message Passing Interface (MPI) library (Open MPI, MPICH)
- GNU scientific library (GSL)
- Parallel Fast Fourier Transform library FFTW, version 2
- Optional: HDF5
- Gadget-2 code

Easier than it looks

NERSC



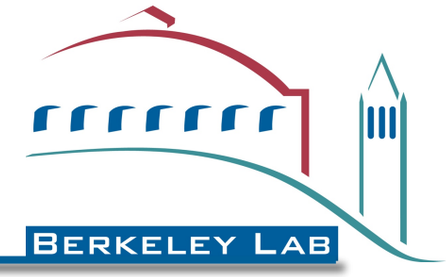
- module swap PrgEnv-intel PrgEnv-gnu (optional)
- module load gsl fftw/2.1.5.7
- cray-mpich is the default, and we'll keep that

- cd some place
- wget http://www.mpa-garching.mpg.de/gadget/gadget-2.0.7.tar.gz
- tar -zxf gadget-2.0.7.tar.gz
- cd Gadget-2.0.7/Gadget2

Makefile

- Defines compiler, flags, and some standard items, but also:
    - boundary conditions
    - gravitational softening options
    - PM grid size (not a runtime parameter!)
- Note that at the end of the file you have explanations for all of the gadget compile-time options

Parameter file



- Like any non-trivial code, Gadget-2 has some runtime parameters, like:
    - I/O options
    - System of units
    - Simulation parameters (cosmology, box-size...)
    - Numerical parameters (softening, tree opening...)
    - Other stuff...

Run

- sbatch script_name
- sqs to see job's status in the queue
- analyze data after the run is done