CODES IC´s Cosmological Nbody Simulations

ISYA 2018, Colombia, Socorro

Octavio Valenzuela

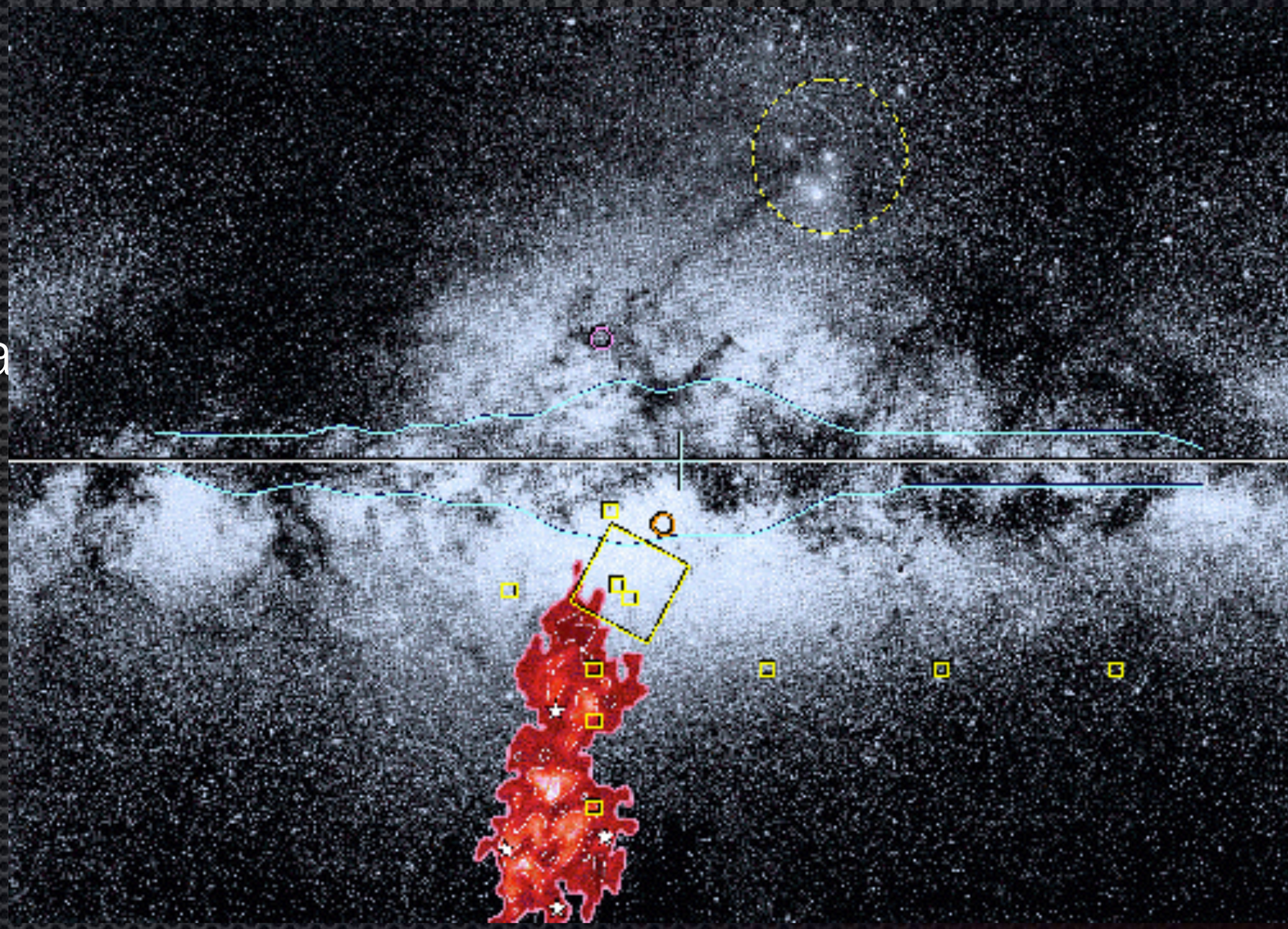
Reasons to use N-body Simulations

Exploring the role on non-linear dynamical evolution, hard to do in analytical way for more than 2 bodies

Studying non-equillibrium, transient processes in a selfconsistent way.

The Galactic Bulge and Sagittarius (Dynamical Feedback in action)

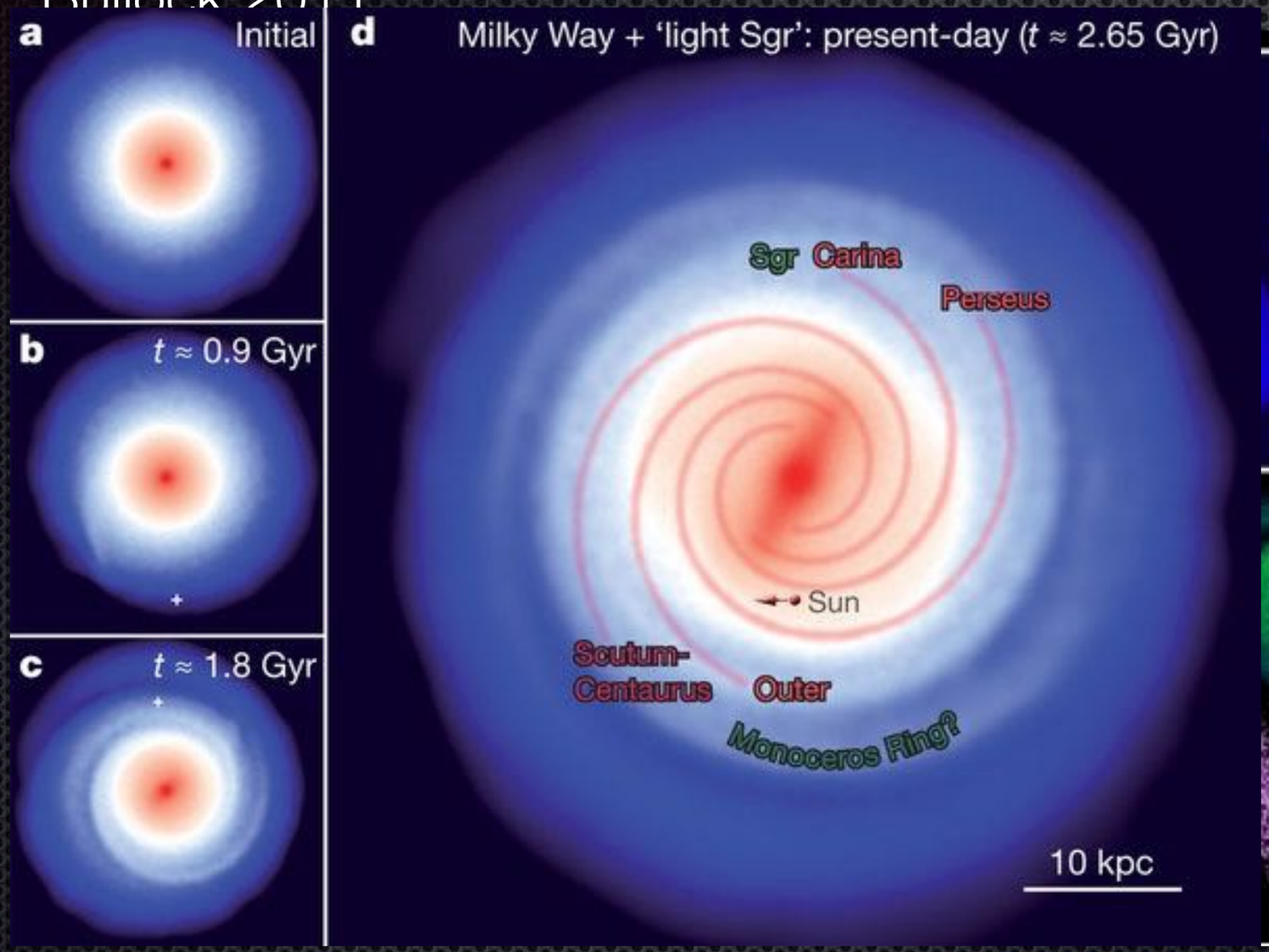
- The Sagittarius
- satellite
- being disrupted by a
- with the
- Milky Way

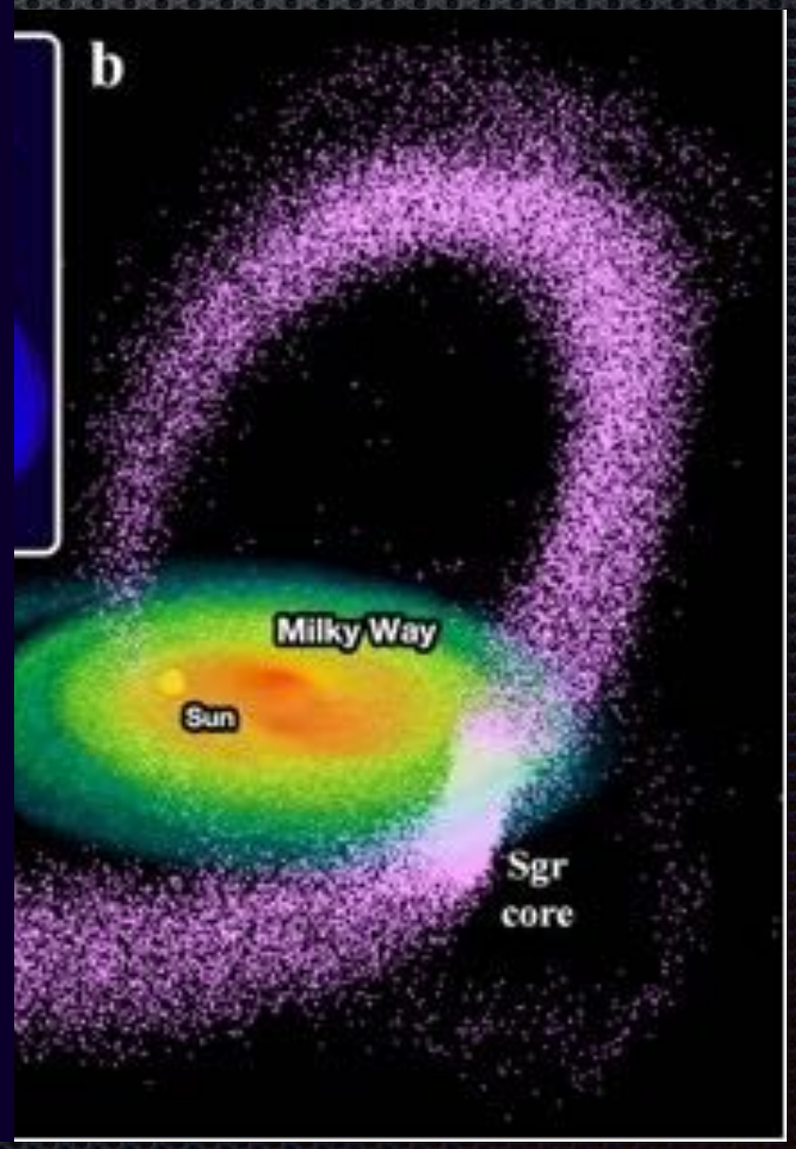


- Ibata 1994

l Stripping and Stirring

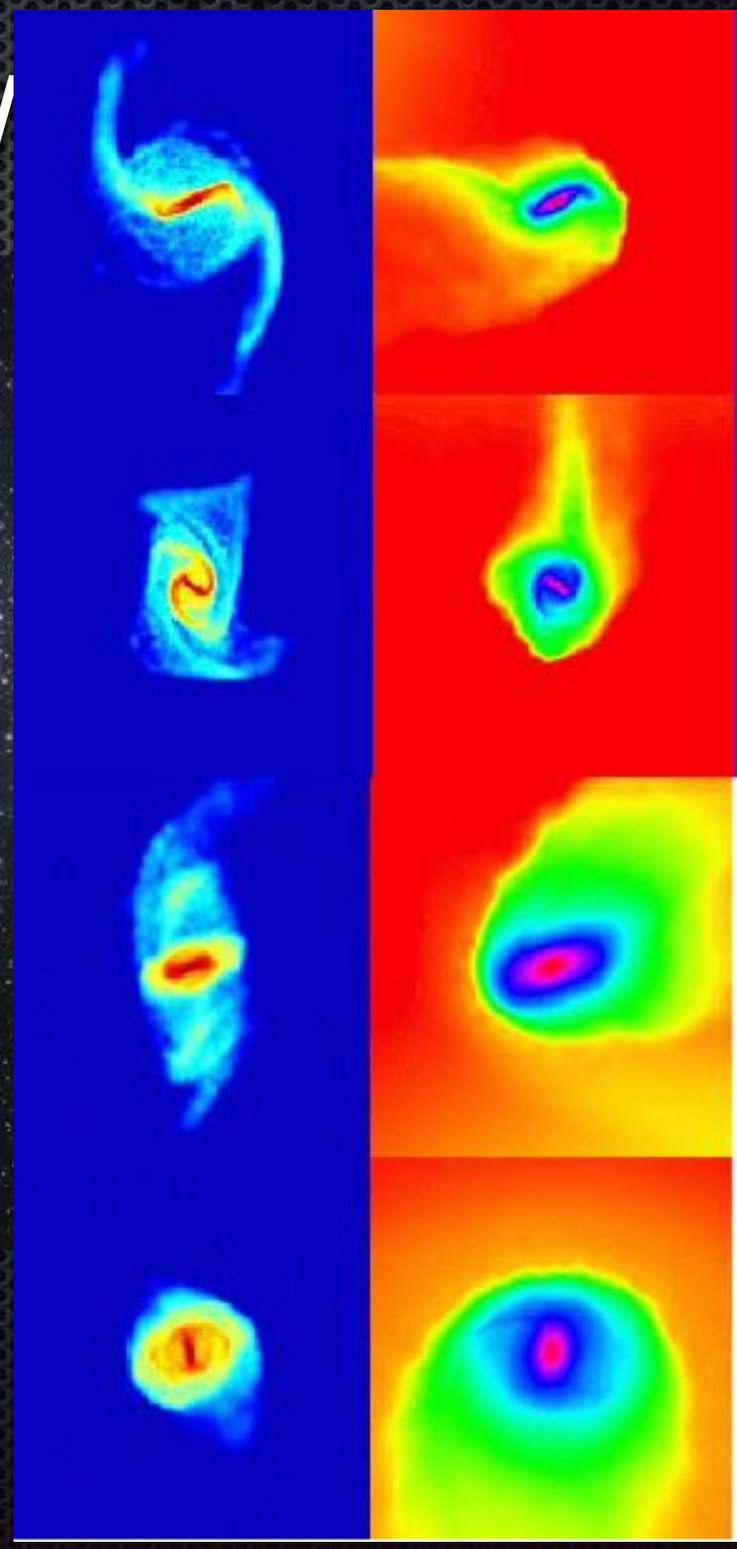
Spiral Arms and disk rings secularly or tidally triggered? Purcell, Bullock 2011





Law, Majewsky

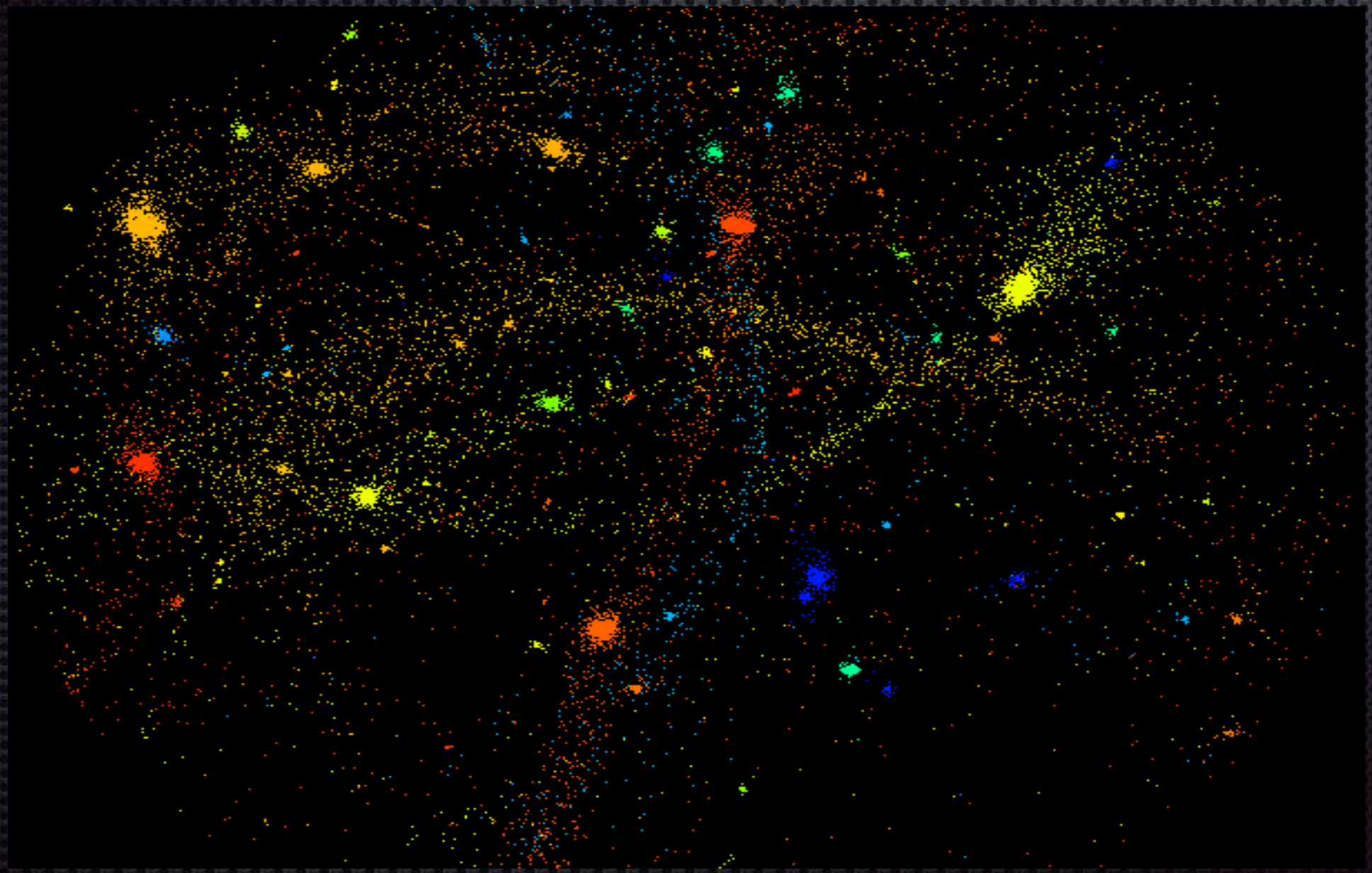
What about M31? Dynamical state of its satellites? MW

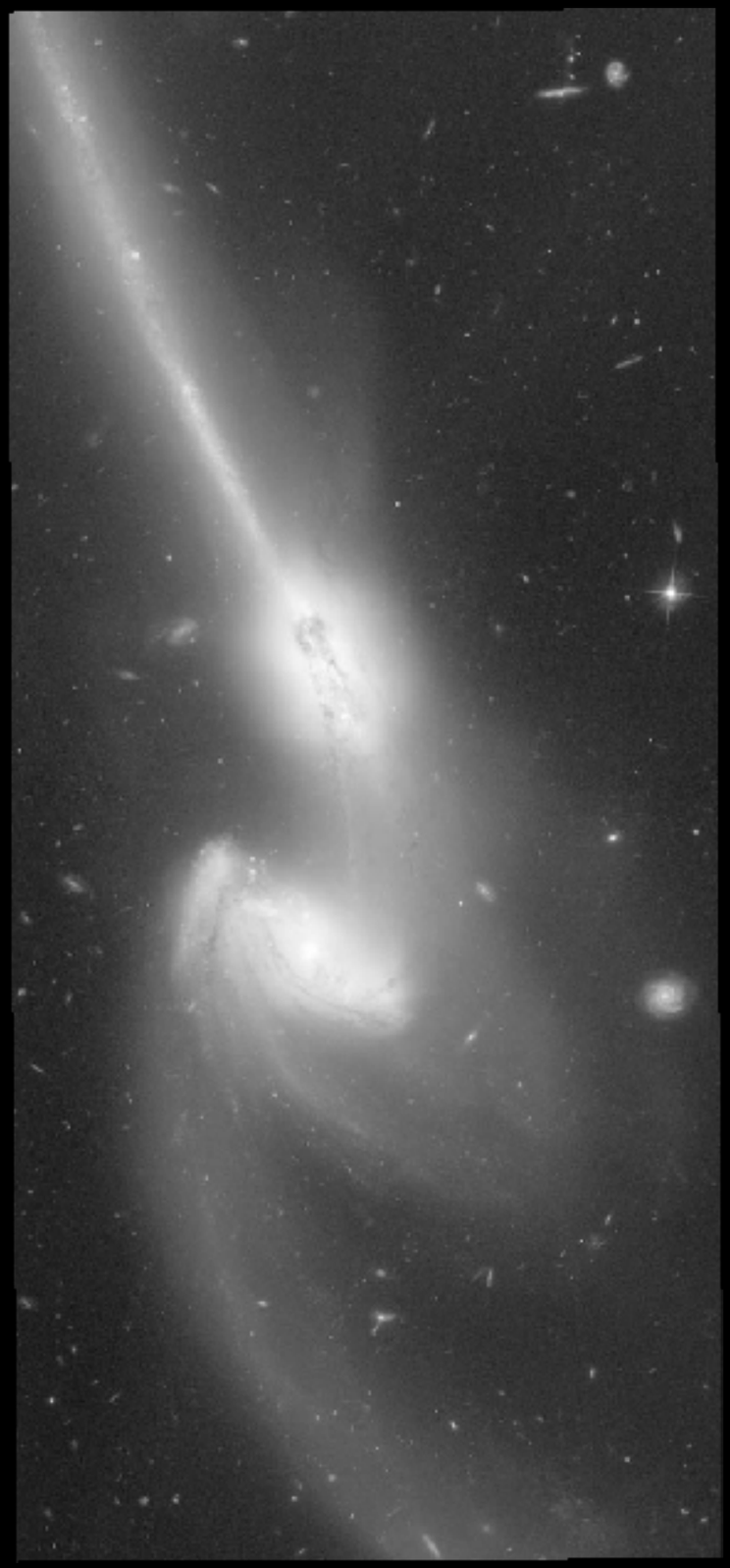


Biasing interpretation of observations?
From disk? Migrations. Bulge. From satellites?

Tidal stirring

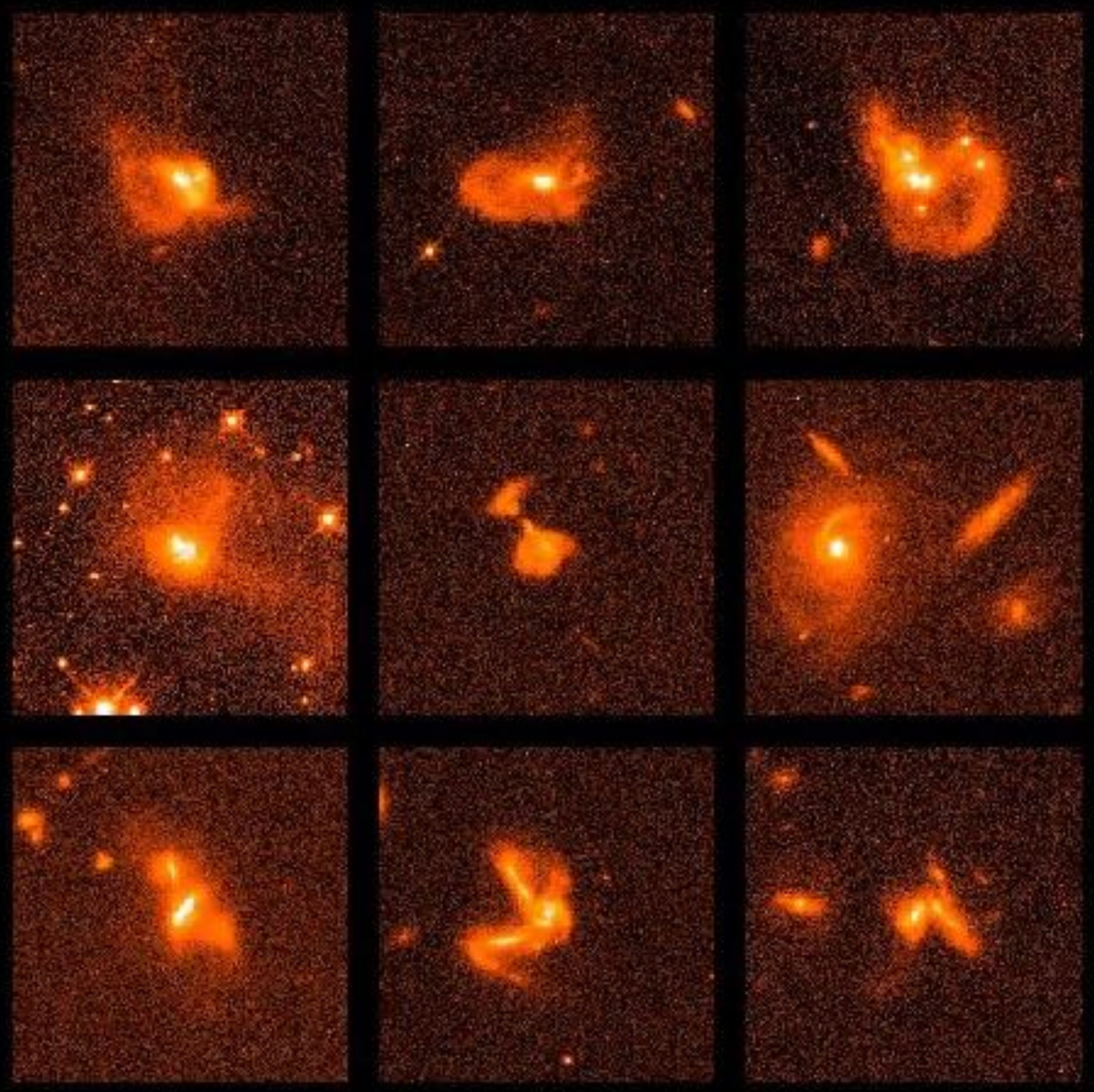
Halo Archeology



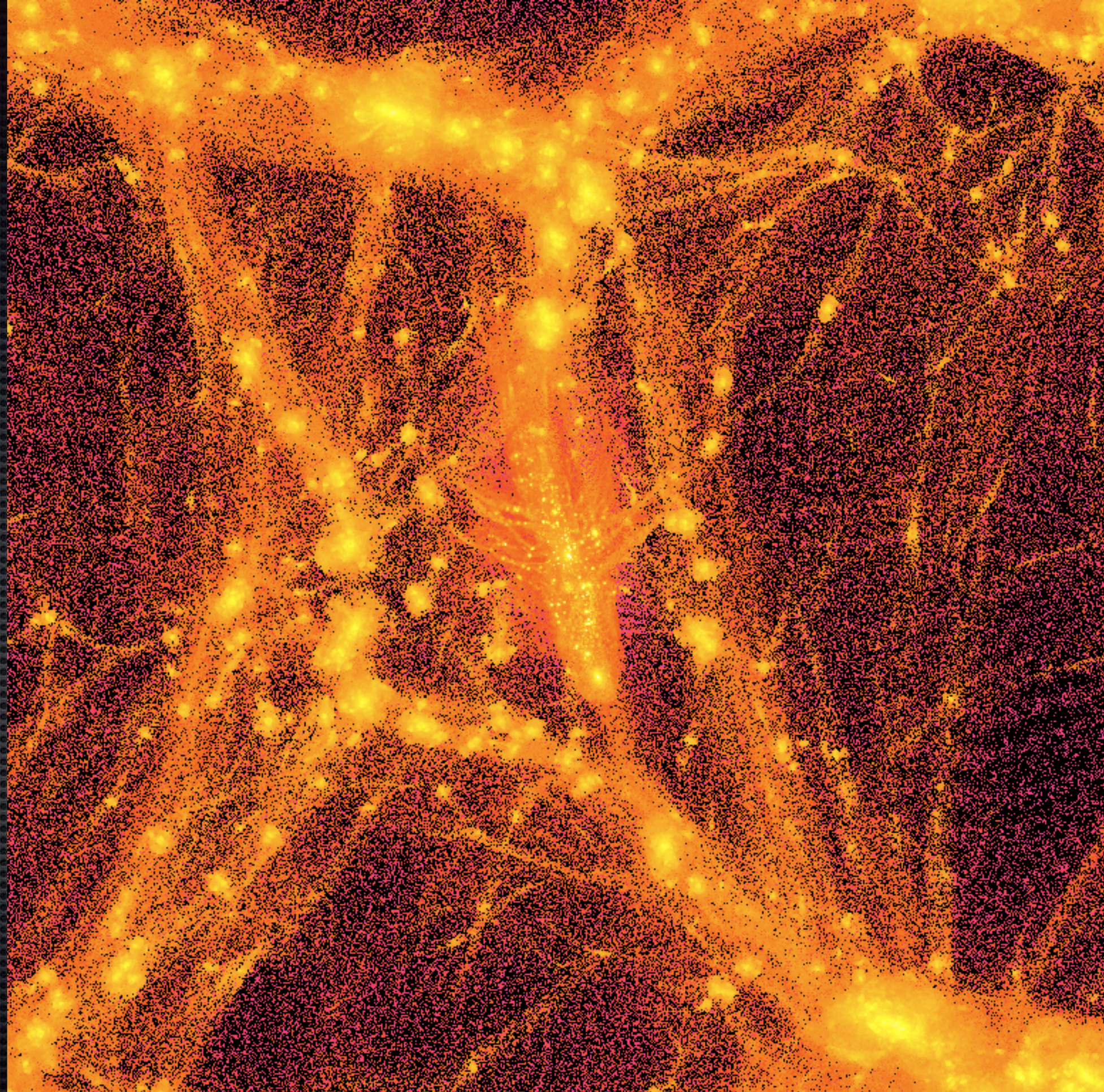


Galaxy - Galaxy interactions (HST)

High redshift Galaxies

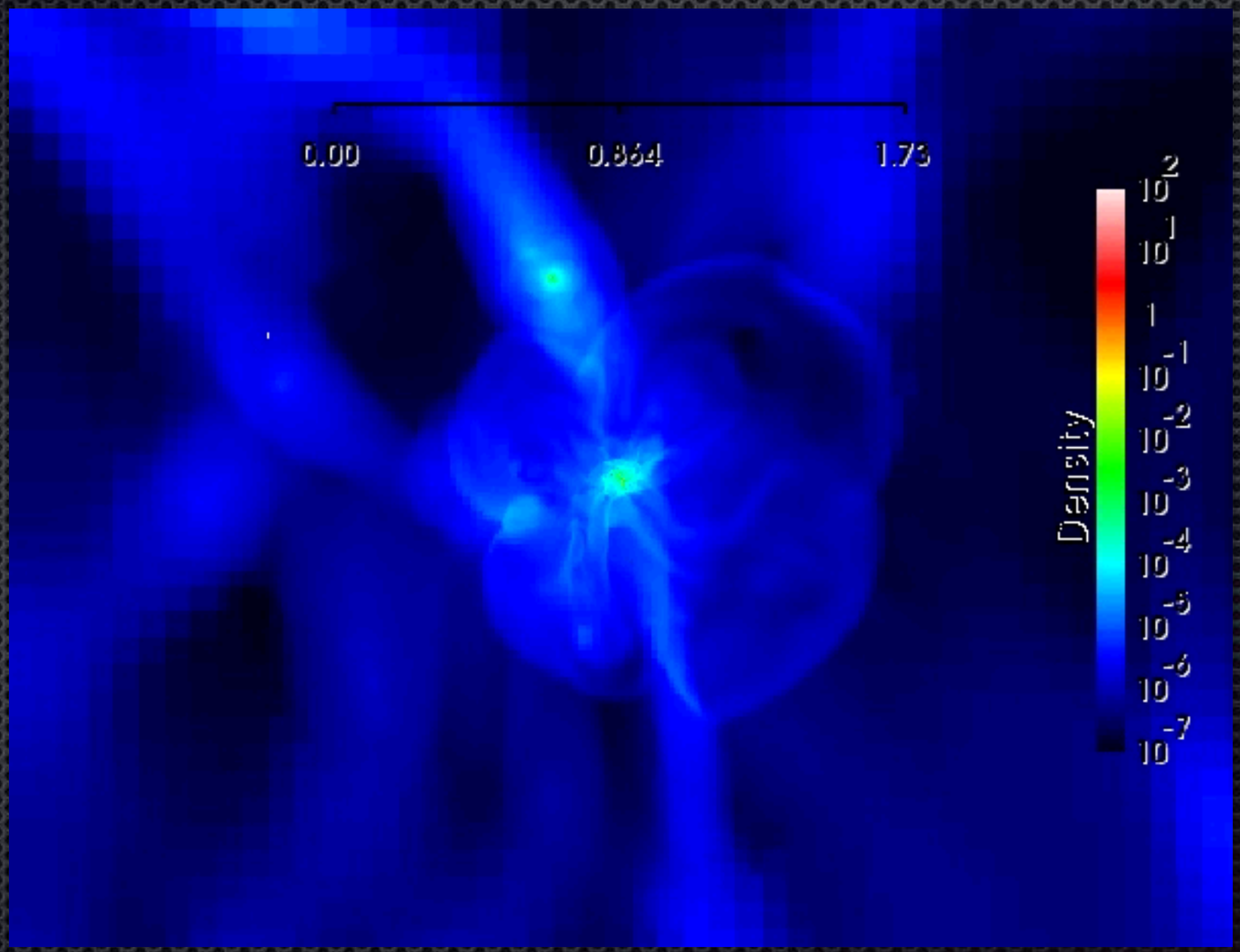


Dark Matter
Galaxy
Environment

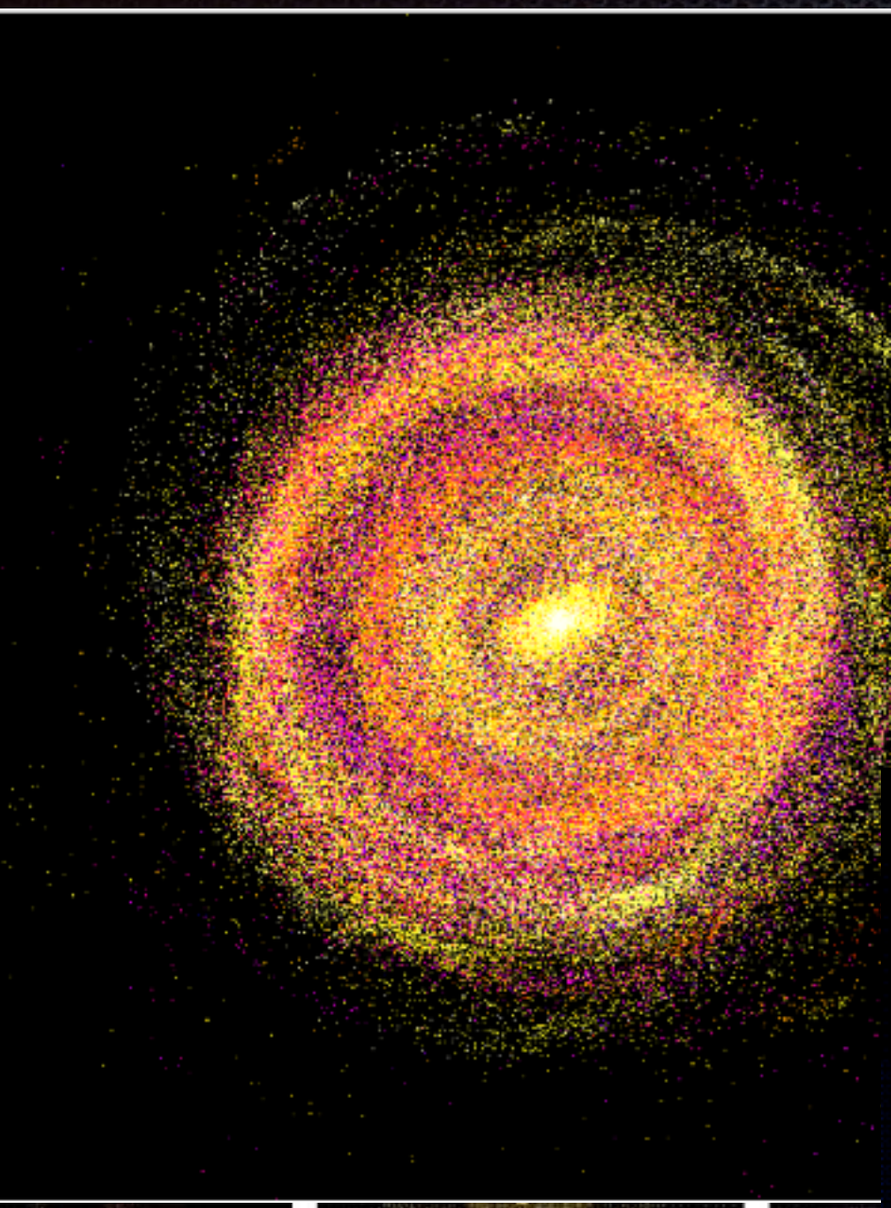


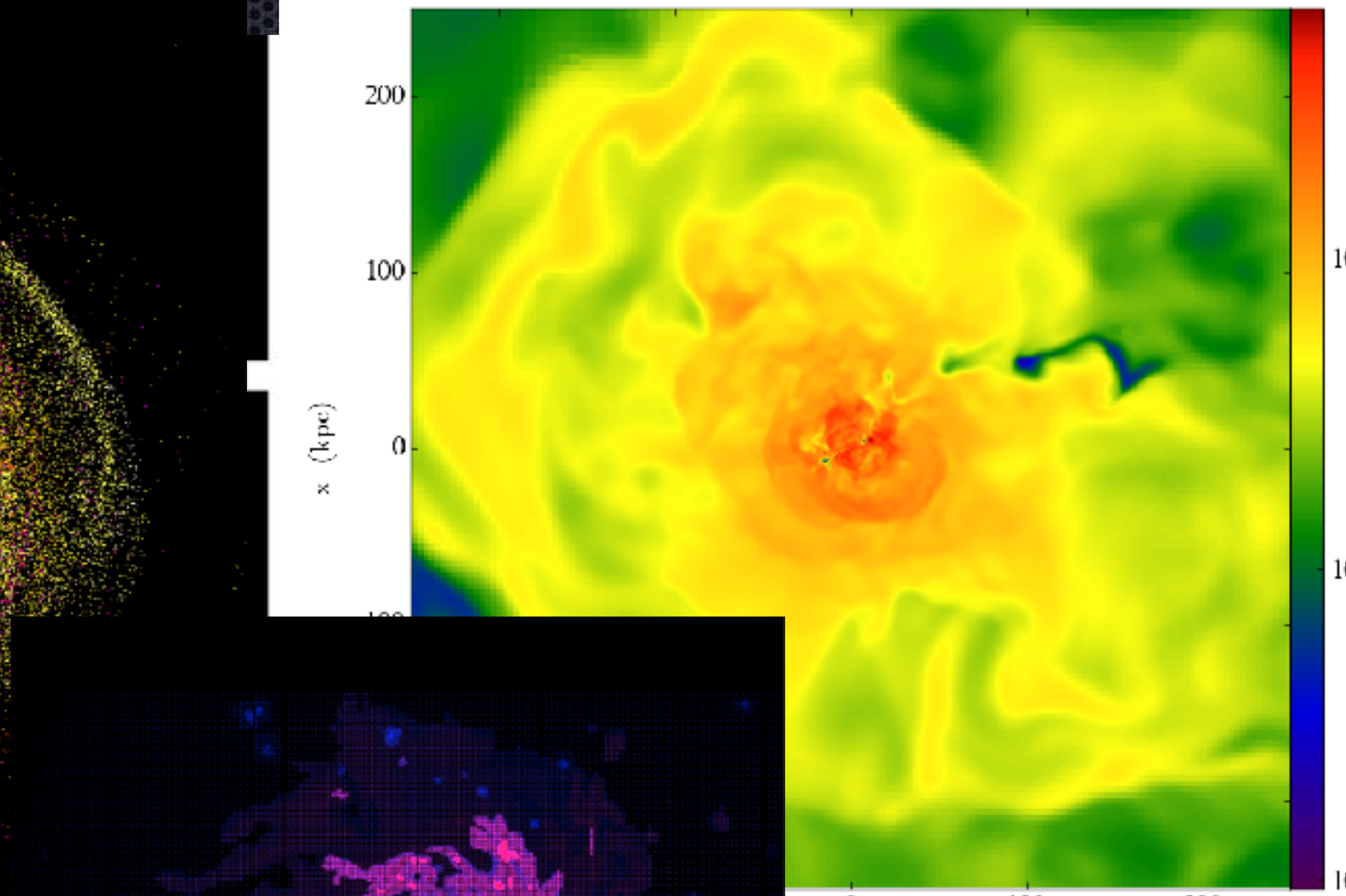
Galaxy at z=1

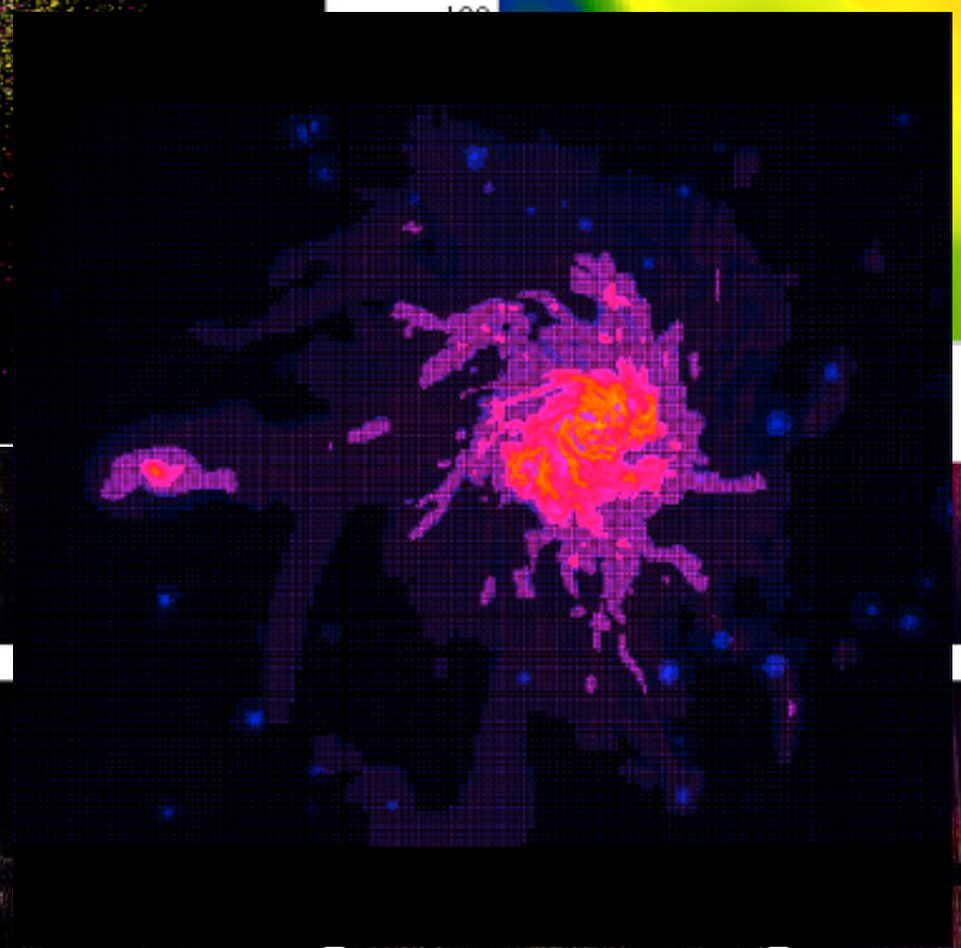
Gonzalez-Samaniego, Valenzuela, Colin

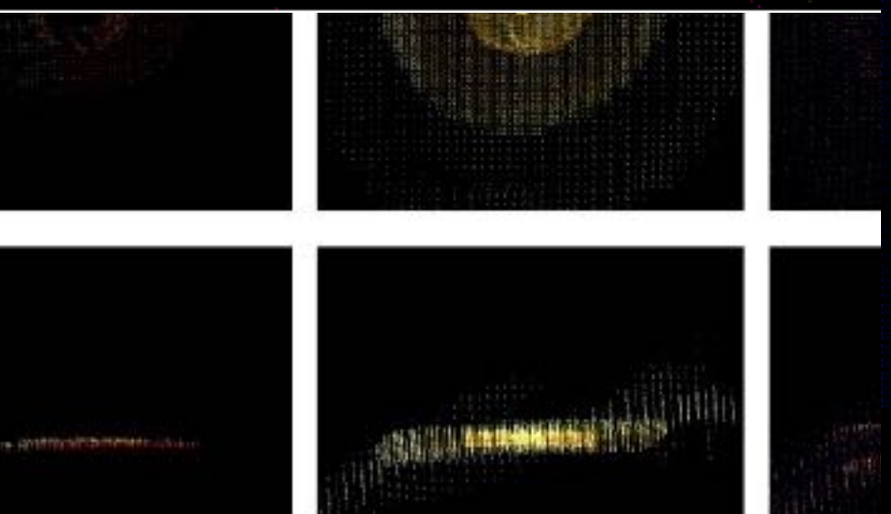


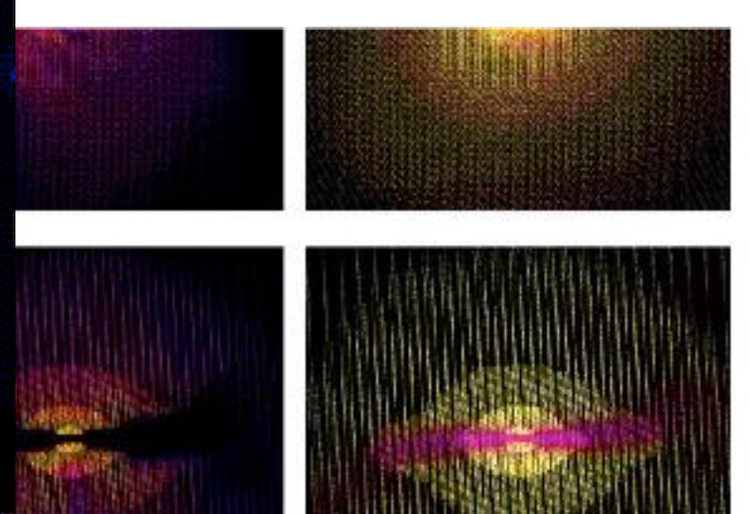
width = 111 kpc

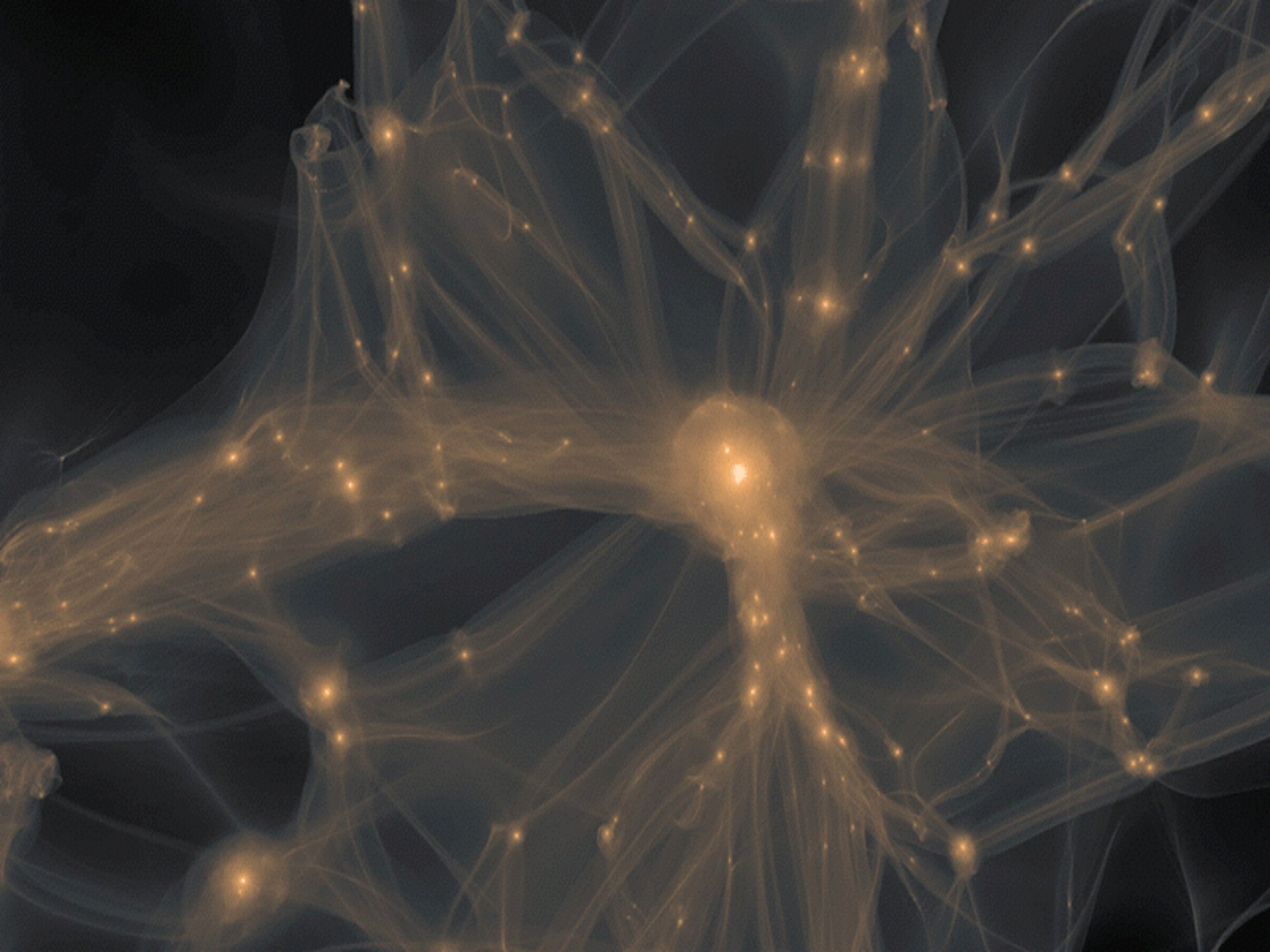


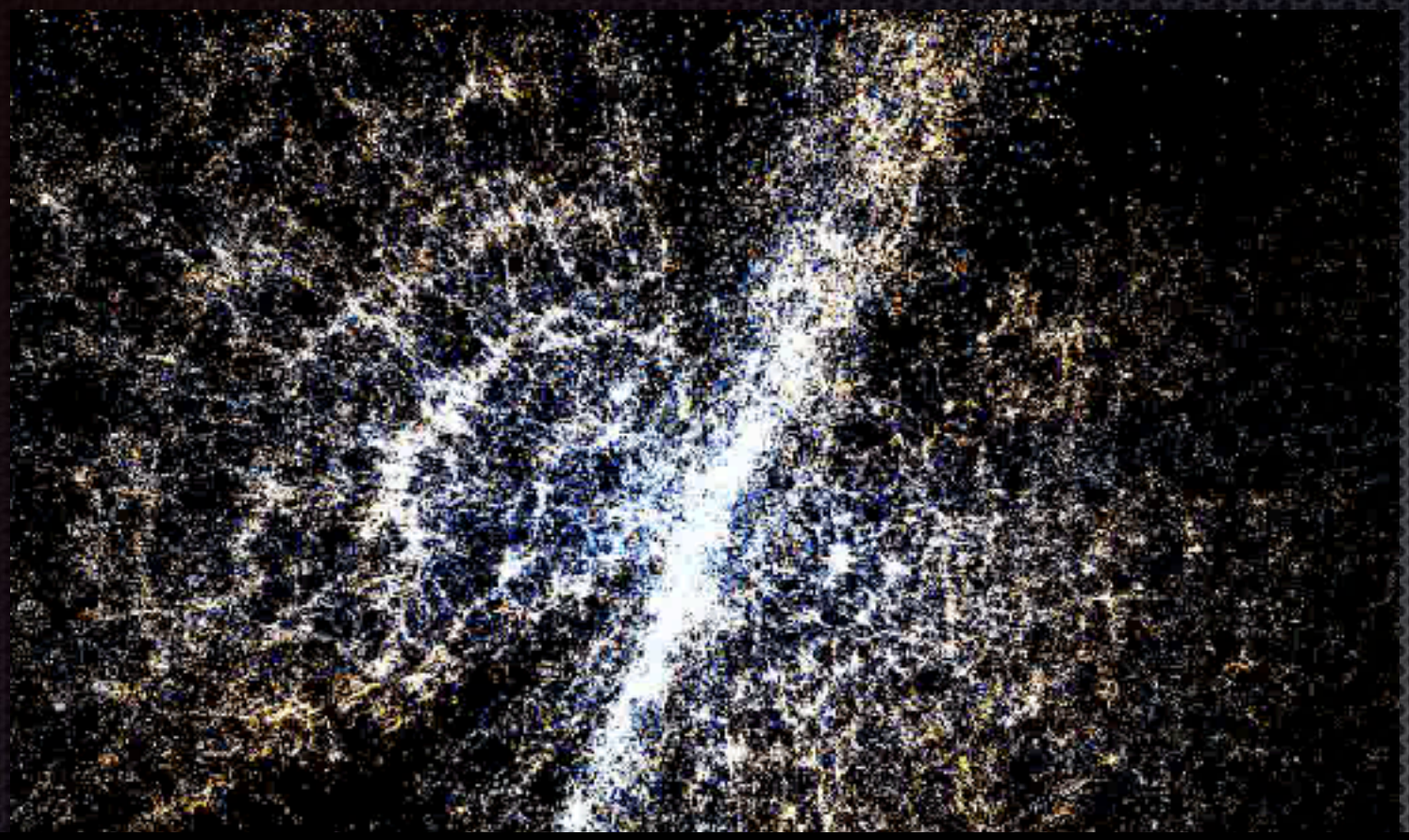

200
100
0
x (kpc)
10^6
10^5
10^4
Temperature (K)
0
100
200
z (kpc)











Cosmic Large Scale Structure growth is a complex multi scale process

Transient process are quite slow so they bias our interpretation

Galaxies are the tracers however they are biased with respect the average density

Avoid these reasons:

- Easier than analytic solution
- The only robust solution
- The most realistic approach

- Nbody simulations are just another approach to dynamical problems
- Very useful, but you have to test, and test the robustness of results
- Compare if possible against analytic solutions
- Convergence studies of solutions is your guide most of the time, because frequently there is not an analytical solution

You have a powerful toy, be careful

Kinetic description in terms of Vlasov-Poisson

Density of particles in phase space

= distribution function $f(\mathbf{x}, \mathbf{v}, t)$

The most general way to handle the problem however very expensive at least in memory

Evolution governed by **Boltzmann equation**

$$\frac{\partial f}{\partial t} + \frac{\mathbf{v}}{a^2} \cdot \boldsymbol{\nabla}_{\mathbf{x}} f - \boldsymbol{\nabla}_{\mathbf{x}} \phi \cdot \boldsymbol{\nabla}_{\mathbf{v}} f = C[f]$$

- $\frac{\partial f}{\partial t}$: velocities advect in configuration space
- $\frac{\mathbf{v}}{a^2} \cdot \boldsymbol{\nabla}_{\mathbf{x}} f$: velocities advect in configuration space
- $\boldsymbol{\nabla}_{\mathbf{x}} \phi \cdot \boldsymbol{\nabla}_{\mathbf{v}} f$: grav. forces advect in momentum space
- $C[f]$: particle-particle interactions = **0**

+ Poisson equation for grav. potential

$$\boldsymbol{\nabla}^2_{\mathbf{v}} \phi = \frac{4\pi G}{a^3} \int (f - \bar{\rho}) \, \mathrm{d}^n v$$

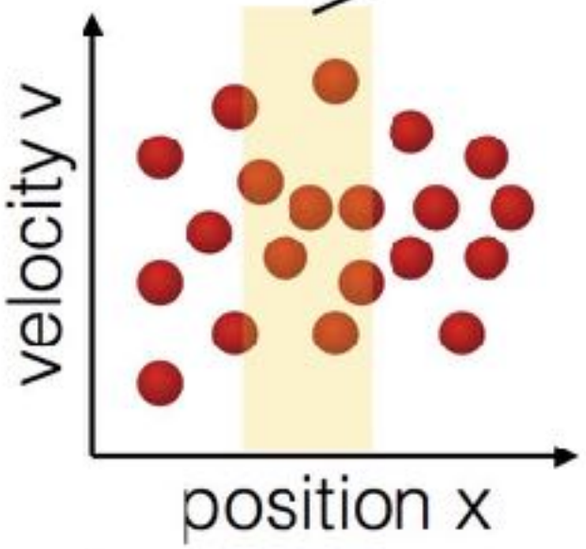
for C[f]=0: Vlasov-Poisson

Kinetic description of dark matter

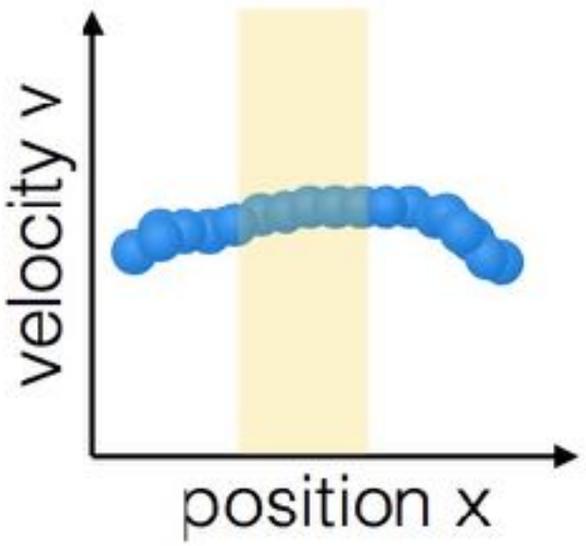
Density of particles in phase space given by

= distribution function $f(\mathbf{x}, \mathbf{v}, t)$

hot case



cold case



phase space of n+n dim:

generally,
f is truly 2n-dimensional

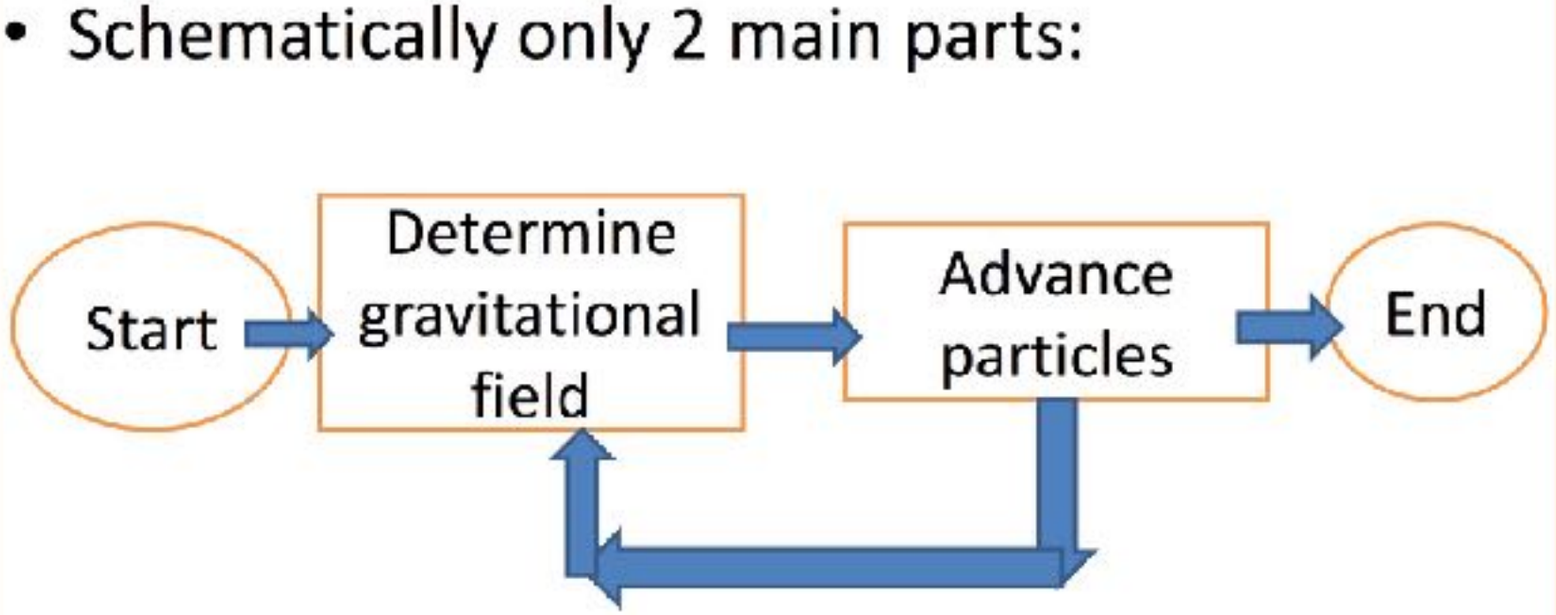
in cold limit,
f is only n-dimensional
= **monokinetic**

You can also use moments of Boltzman equation and solve hydro-like equations

General Structure of N-body codes

- Density/Force Cal
  conditions problen
- Time Integration

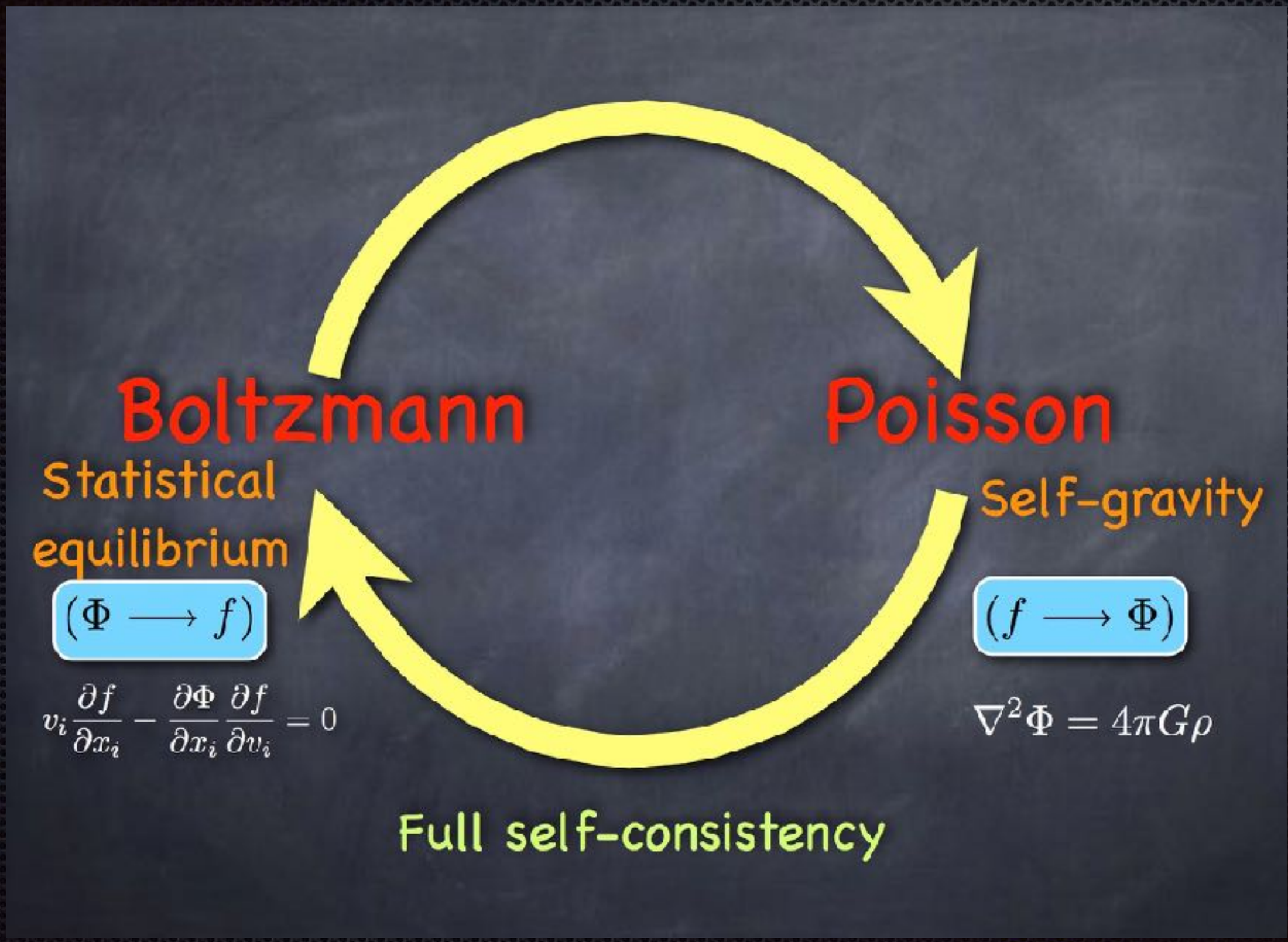
- Schematically only 2 main parts:



Initial Conditions: The Art

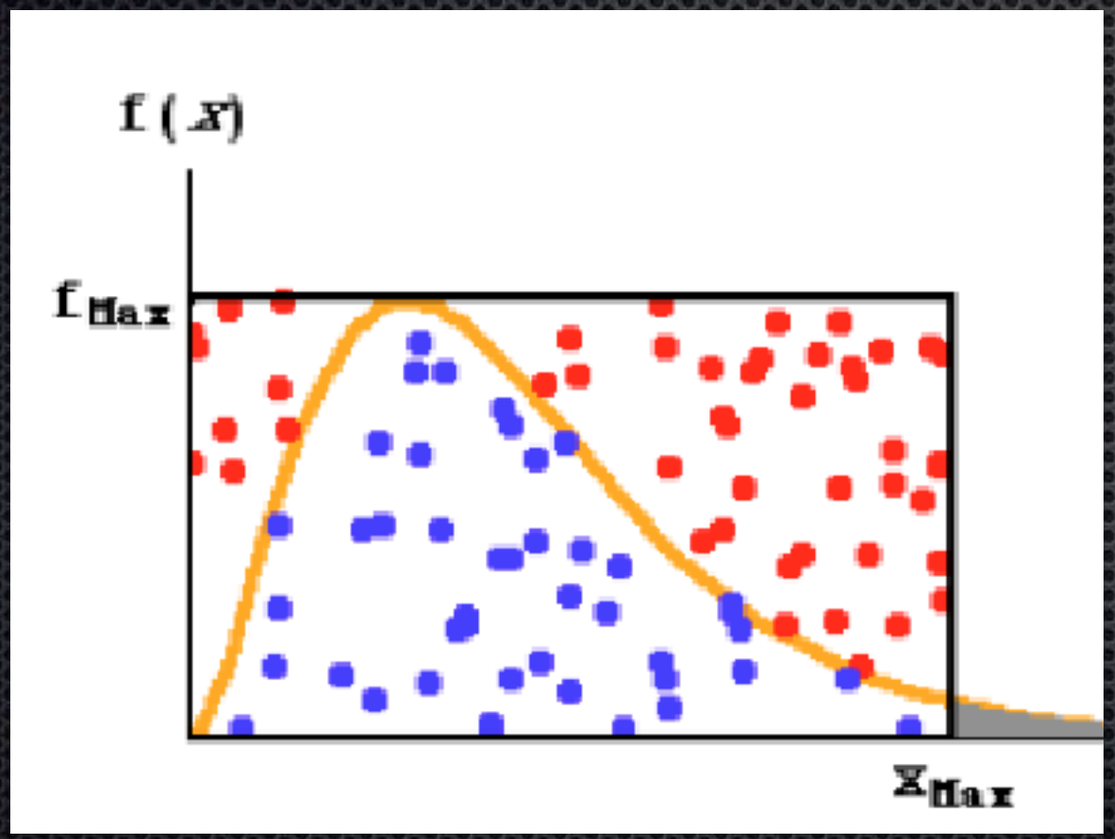
- Isolated
- Galaxies, halos, etc

Ideally we would like...for Isolated dynamical systems



Method I: Phase Space Function

- Fully Self-consistent
- But…PSF available only for idealized symmetric systems



Random sampling?

Quiet Start (Sellwood), cosmological neutrinos??

Inhibits scatter

Prendergast & Tomer (1970)

- This method works for axisymmetric (flattened) potentials and an adopted form for $f(E,L_z)$
- Iterate
  1. guess an initial $\rho(r,z)$
  2. solve for Φ (*e.g.* using Poisson solver)
  3. compute a new $\rho(r,z) = \int f\, d^3\boldsymbol{v}$
  4. complete steps 2 & 3 until changes in ρ are small (8 – 10 iterations)
- Adaptable and yields a good equilibrium

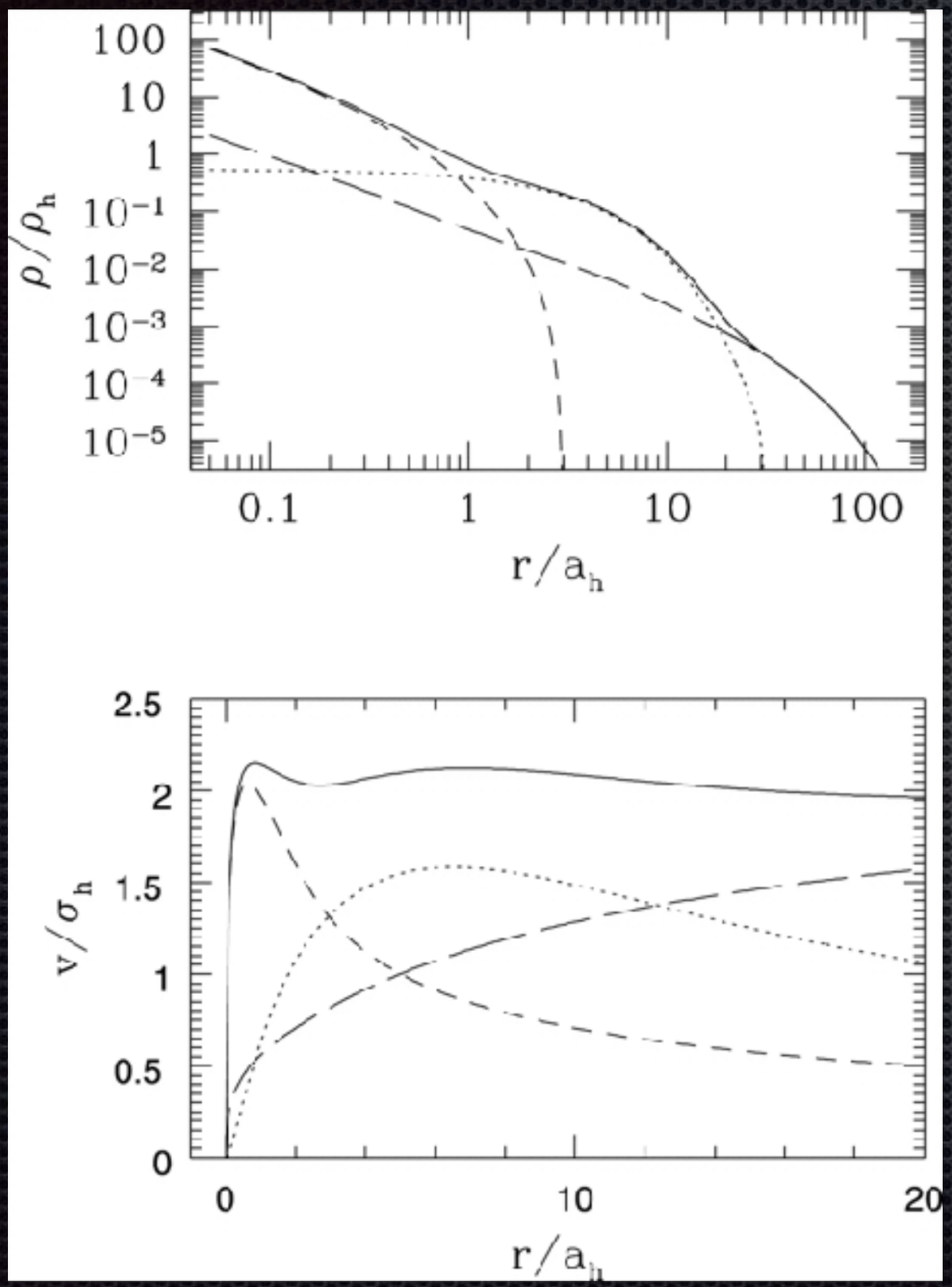
Because
Halo disk
<- interaction

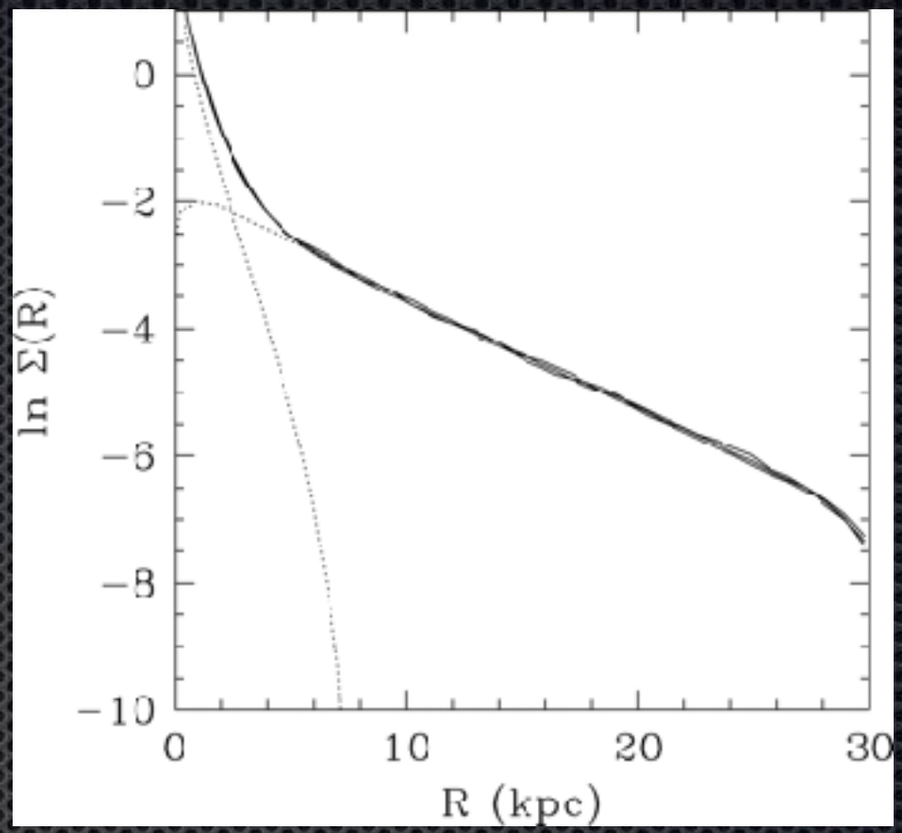
But the final stage is not always what we wanted
Sometimes not converging

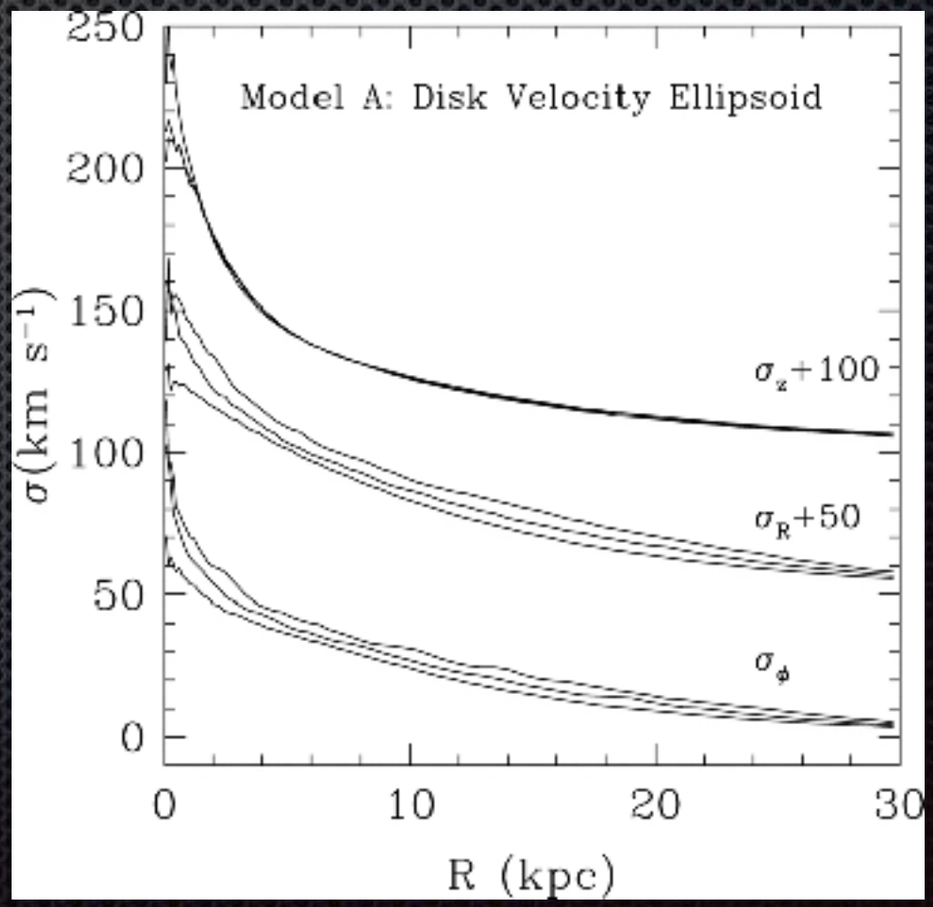
Starts assuming a spherical disk from the point of view of the halo

Excelent for controlled experiments

Implementation
Widrow Dubinsky for MW/M31







Cosmological

GENERATING INITIAL CONDITIONS

SUPERIMPOSING DENSITY PERTURBATIONS

$$\delta(\vec{x},t) = \frac{\rho(\vec{x},t) - \bar{\rho}(t)}{\bar{\rho}(t)}$$

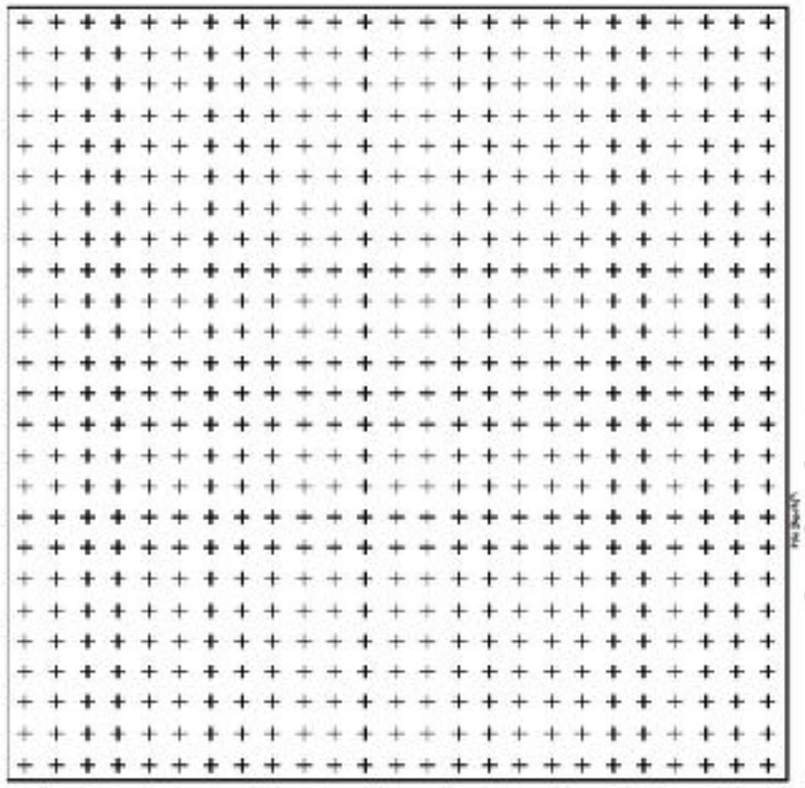
$$\delta(\vec{x}) = \sum \hat{\delta}(\vec{k}) e^{-i\vec{k}\cdot\vec{x}}$$

FFT

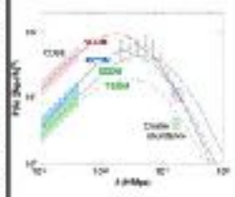
$$P(k) = \left\langle \left| \hat{\delta}(\vec{k}) \right|^2 \right\rangle_{|\vec{k}|=k}$$

RMS density deviation at scale 2Pi/k

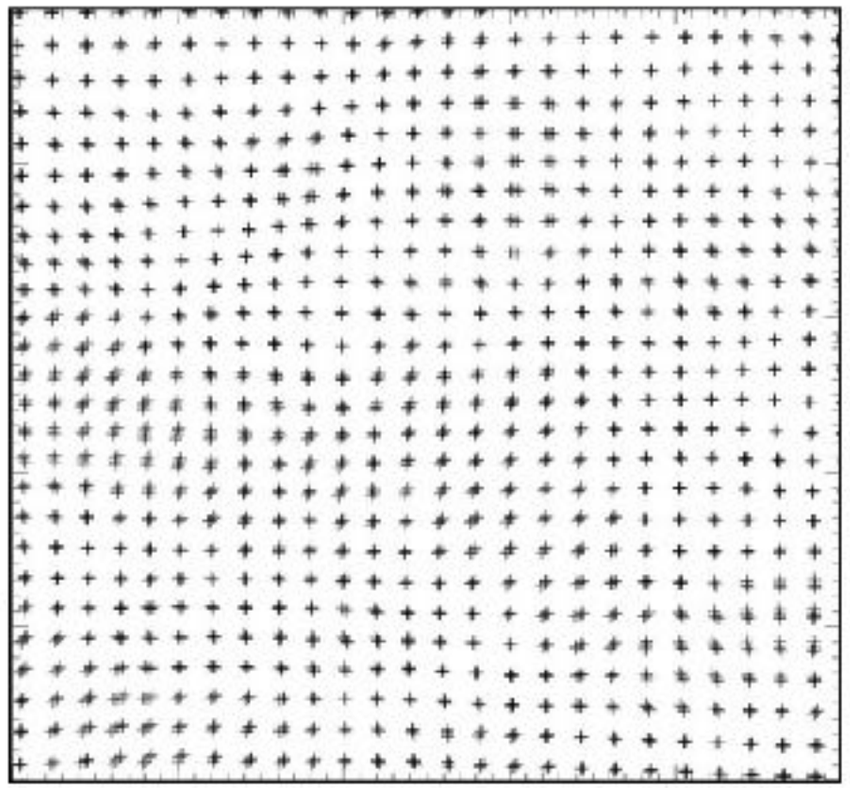
homogeneous & isotropic



$P(k)$



initial conditions



$$\delta(\vec{x}) = \sum \hat{\delta}(\vec{k}) e^{-i\vec{k}\cdot\vec{x}} \qquad P(k) = \left\langle \left| \hat{\delta}(\vec{k}) \right|^2 \right\rangle_{|\vec{k}|=k}$$

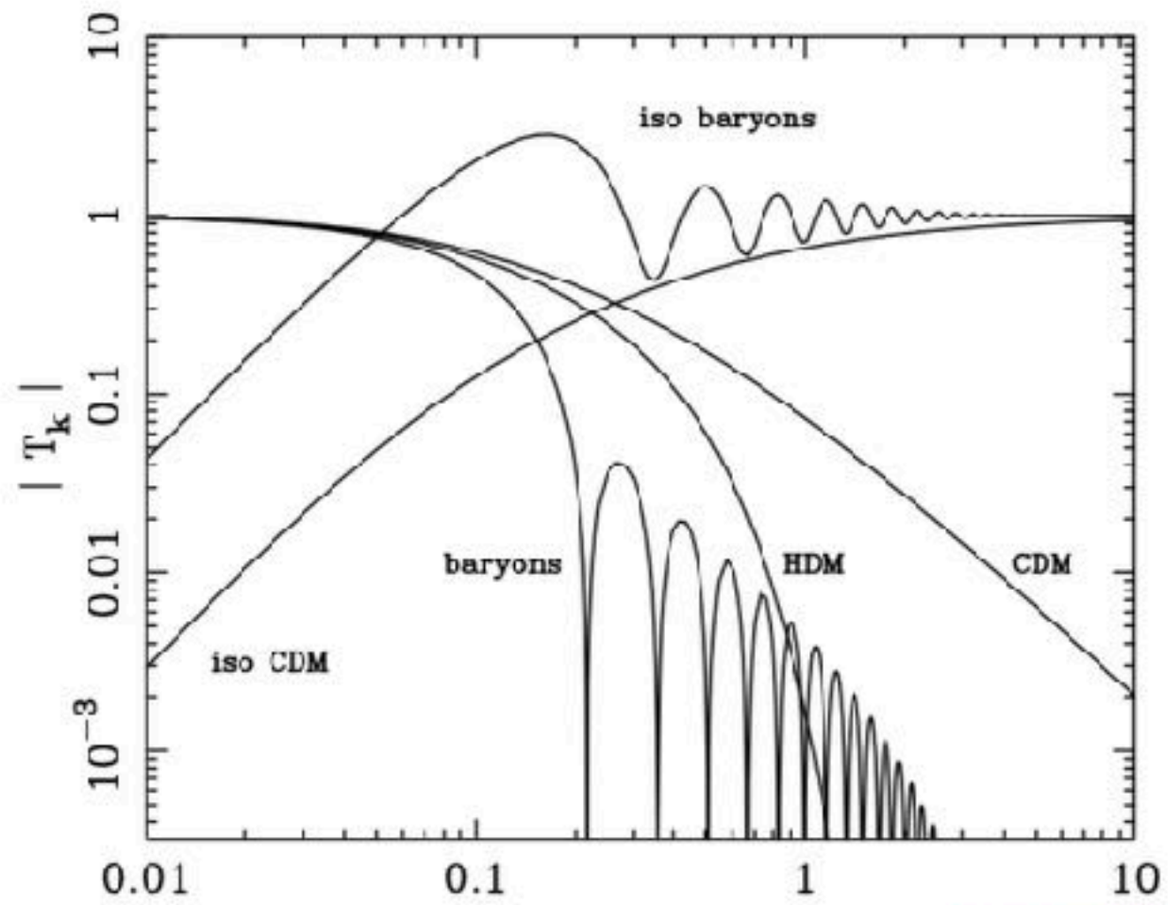
- inflation theory

$$P_i(k) = Ak^n \qquad \text{with } n = 1 \text{ (Harrision - Zeldovich spectrum)}$$

- transferring $P(k)$ across recombination

$$P(k) = T^2(k) P_i(k)$$

$$\delta(\vec{x}) = \sum \hat{\delta}(\vec{k}) e^{-i\vec{k}\cdot\vec{x}} \qquad P(k) = \left\langle \left| \hat{\delta}(\vec{k}) \right|^2 \right\rangle_{|\vec{k}|=k}$$



captures all the complicated physics due to the coupling of radiation and matter

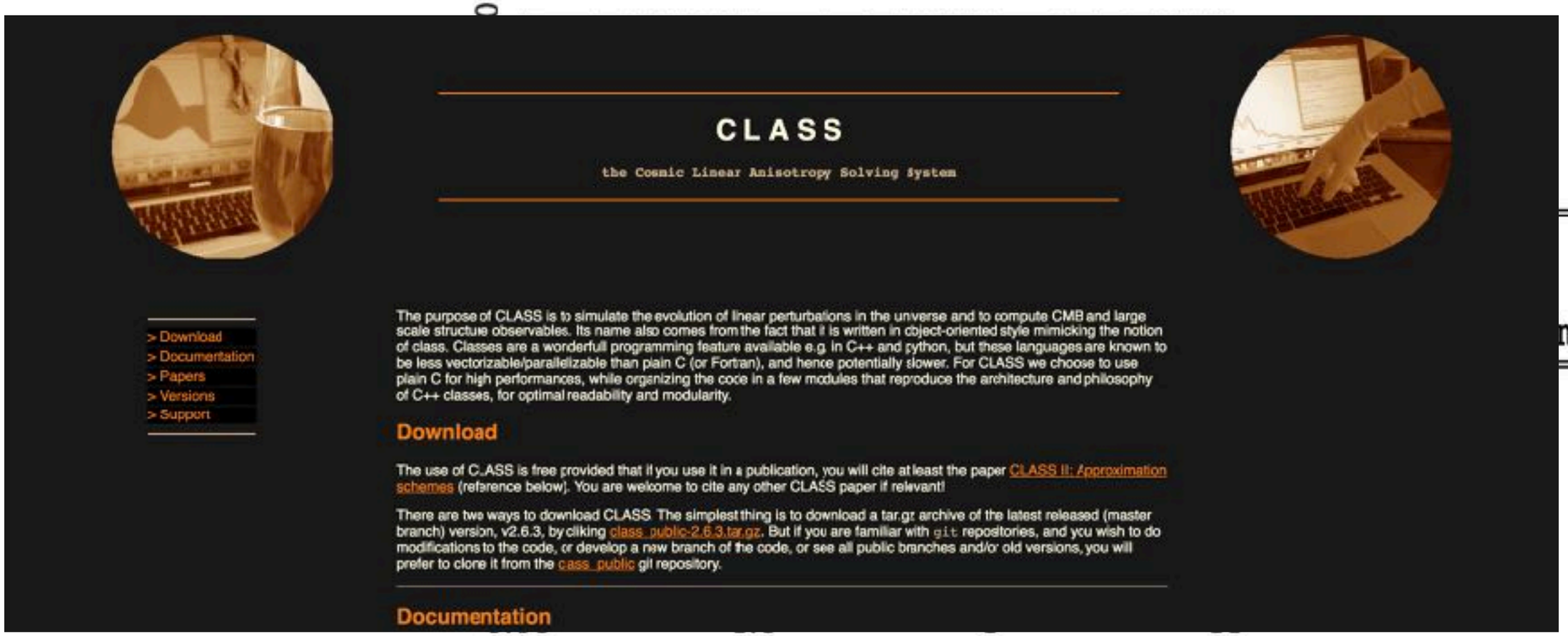
$$P(k) = T^2(k) P_i(k)$$

GENERATING INITIAL CONDITIONS

SUPERIMPOSING DENSITY PERTURBATIONS

$$\delta(\vec{x}) = \sum \hat{\delta}(\vec{k}) e^{-i\vec{k}\cdot\vec{x}}$$

$$P(k) = \left\langle \left| \hat{\delta}(\vec{k}) \right|^2 \right\rangle_{|\vec{k}|=k}$$



here goes half of
CDM, WDM,SIDM,
SFDM, massive Neutrinos
Mod Gravity, contrasted with CMB

$$\mathrm{k} \;/\; \Omega_{\mathrm{m}} \mathrm{h}^2 \;\mathrm{Mpc}^{-1}$$

$$P(k) = T^2(k) P_i(k)$$

$$\delta(\vec{x}) = \sum \hat{\delta}(\vec{k}) e^{-i\vec{k}\cdot\vec{x}}$$

FFT

$$P(k) = \left\langle \left| \hat{\delta}(\vec{k}) \right|^2 \right\rangle_{|\vec{k}|=k}$$

RMS deviation at scale 2Pi/k

- remember linear perturbation theory*:

Is a continuous media, colisionless plasma, Boltzmann moments

density contrast:

$$\frac{\partial \delta}{\partial t} + \frac{1}{a}\nabla \cdot \vec{u} = 0$$

mass conservation

peculiar velocity field:

$$\frac{\partial \vec{u}}{\partial t} + \frac{\dot{a}}{a}\vec{u} = -\frac{1}{a}\nabla\Phi$$

momentum conservation

$$\Delta\Phi = 4\pi G a^2 \bar{\rho}\delta$$

Poisson's equation

$$\delta(\vec{x}) = \sum \hat{\delta}(\vec{k}) e^{-i\vec{k}\cdot\vec{x}} \qquad P(k) = \left\langle \left| \hat{\delta}(\vec{k}) \right|^2 \right\rangle_{|\vec{k}|=k}$$

- remember linear perturbation theory*:

$$\frac{\partial^2 \delta}{\partial t^2} + 2\frac{\dot{a}}{a}\frac{\partial \delta}{\partial t} - 4\pi G\bar{\rho}\delta = 0$$ evolution of matter density

$$\delta(x,t) = D(t)\delta_0(x)$$

$$\boxed{P(k) = D^2(t)P_0(k)}$$

- Zel'dovich approximation:

$$\vec{x}(t) = \vec{q} + D(t)\vec{S}(\vec{q})$$

Ansatz based upon the idea to

displace particles from their initial positions on a regular mesh

- Zel'dovich approximation:

$$\boxed{\vec{x}(t) = \vec{q} + D(t)\vec{S}(\vec{q})}$$

$$D(t) = \frac{\delta(\vec{x},t)}{\delta_0(\vec{x},t_0)}$$

$$\frac{\partial^2 D}{\partial t^2} + 2\frac{\dot{a}}{a}\frac{\partial D}{\partial t} - 4\pi G\bar{\rho}D = 0$$

general solution:

$$D(t) = \frac{5}{2}\Omega_0 \frac{\dot{a}}{a}\int_{t_0}^{t}\frac{1}{\dot{a}^2}dt'$$

- Zel'dovich approximation:

$$\vec{x}(t) = \vec{q} + D(t)\vec{S}(\vec{q})$$

- in practice...

Mesh in your HPC project

$\vec{q}$ = regular grid, i.e. $q_{k,l,m}$

$$D = \frac{5}{2}\Omega_0 \frac{\dot{a}}{a}\int_{t_0}^{t} \frac{1}{\dot{a}^2} dt'$$

$$\vec{S}(\vec{q}) = -\nabla\Psi(\vec{q})$$

$$\Psi(\vec{q}) = FFT^{-1}(\hat{\Psi}(\vec{k}))$$

$$\hat{\Psi} = \hat{\delta}_0(k)\frac{1}{k^2}$$

$$\hat{\delta}_0(k) = \sqrt{P_0(k)}R_{\vec{k}}\, e^{i\varphi_{\vec{k}}}$$

$$R_{\vec{k}} e^{i\varphi_{\vec{k}}} = R_1 + iR_2$$

$$R_1, R_2 = \text{Gauss}(0,1)$$

$D(t)$: determines the initial redshift of the simulation

$S(q)$: determines the direction of displacement

- Zel'dovich approximation:

  - positions

$$\boxed{\vec{x}(t) = \vec{q} + D(t)\vec{S}(\vec{q})}$$

  - velocities

$$\boxed{\dot{\vec{x}}(t) = \dot{D}(t)\vec{S}(\vec{q})}$$

- 2nd order Lagrangian perturbation theory

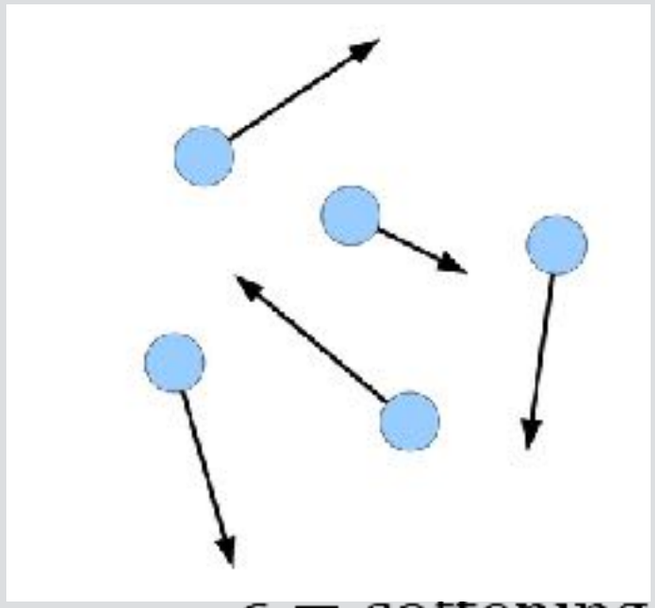
$$\vec{x}(t) = \vec{q} + D(a)S(\vec{q}) - D^{(2)}S^{(2)}(\vec{q})$$

Higher accuracy IC´s, why??

CODES: Which one is the best code?

- Adaptive: Tree codes, Adaptive Mesh Refinement (AMR)
- Particle-Particle
- Fixed Shape, GRIDS, Orthogonal Base

PP-Code



- Ecuación a resolver es Newton:

$$\ddot{\vec{r}}_i = \dot{\vec{v}}_i = \vec{a}_i = \sum_{j=1}^{N} \frac{G m_j \vec{r}_{ij}}{\left(r_{ij}{}^2 + \epsilon^2\right)^{3/2}} \quad , \qquad \epsilon = \text{softening}$$

- Con ϵ se suprime el efecto e importancia de encuentros cercanos (i.e. evita la colisionalidad).
- Los resultados de la simulación son confiables para $\sim 2\epsilon$ (Athanassoula et al. 2000).
- Las unidades usadas son tales que: $G = M_T = a_s = 1$

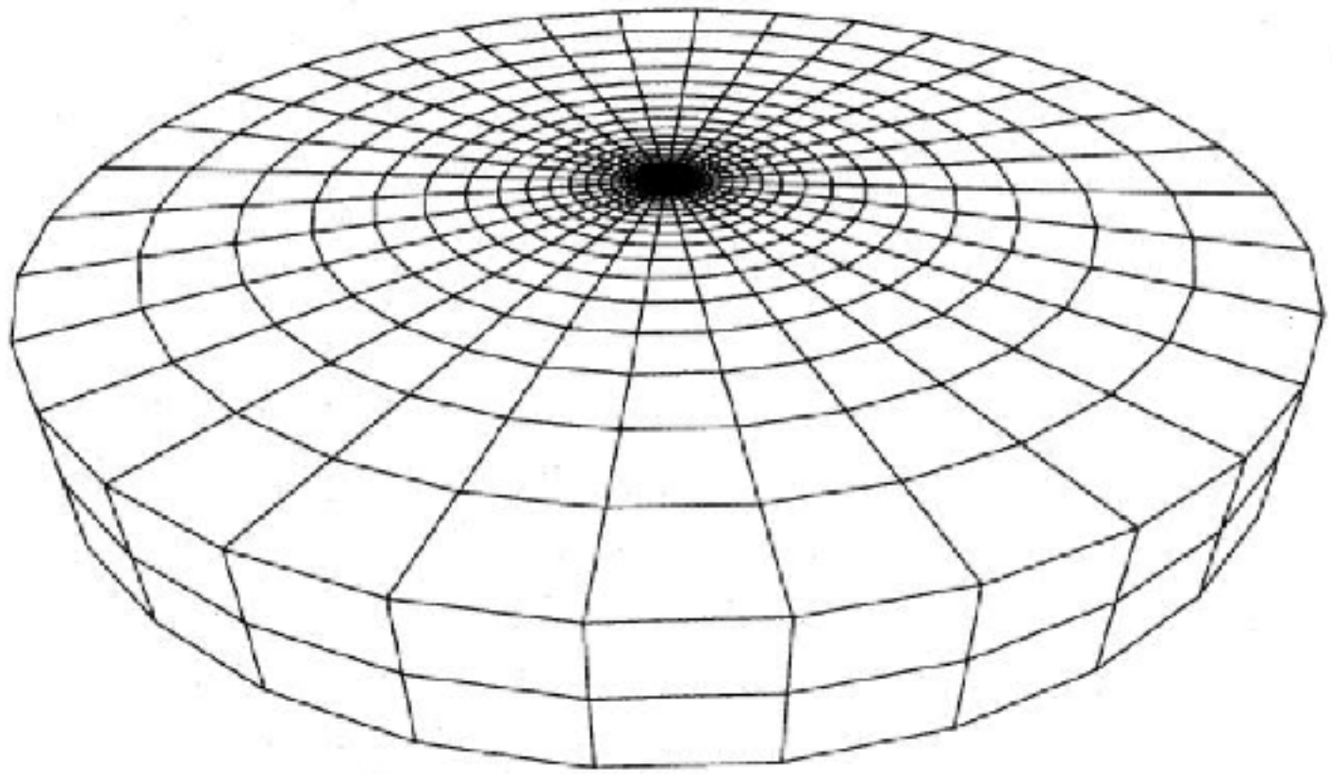
PP Codes

- Challenge: Time of calculation Scales as N^2 (harder go further than 1e6 particles)
- Can be used for collision less systems (galaxies, LSS) but they are not that efficient
- Very Useful for high density systems (clusters, galaxy nucleus, binaries decay)
-

Criteria

- Accuracy: Geometry, capture of relevant processes, good handling of numerical effects, quality of integration
- Efficiency: Speed, with the right accuracy

Grids



- By far the most efficient methods to solve for the gravitational field use grids
  - Cartesian, cylindrical polar, or spherical
  - polar grids have additional advantages

Shape may imply disadvantages when systems evolves quickly in shape

CODES: Which one is the best?

- Cylindrical, spherical
- Coupling cylindrical + spherical, something else?

- Caution, good geometric choice for the beginning
- What happens if it evolves?

Accuracy decreases

- Fixed Shape, GRIDS, Orthogonal Base

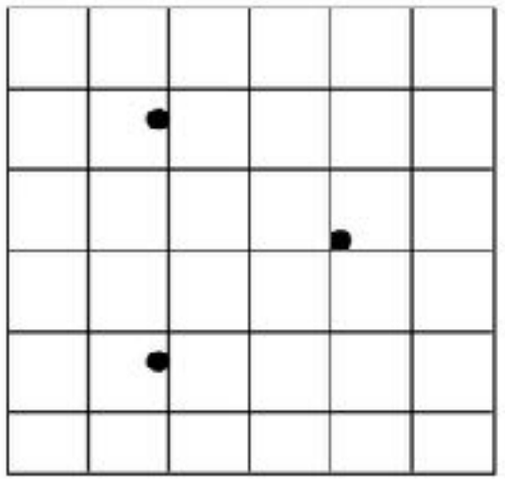
Modern codes currently used many techniques

- PP-Regularization (analytical 2 body solution for close encounters: binary formation and destruction)
- PM (PM + Perturbative solution)
- Tree´s
- Hybrid´s
- AP3M(AMR-fixed + pp)
- Tree-GRAPE (Tree-pp)
- Tree-GPU ()
- AMR
- Hybrid´s

- numerically integrate Poisson's equation

$$\Delta\Phi(\vec{g}_{k,l,m}) = 4\pi G\rho(\vec{g}_{k,l,m})$$

$$\vec{F}(\vec{g}_{k,l,m}) = -m\nabla\Phi(\vec{g}_{k,l,m})$$



1. calculate mass density on grid $\vec{x}_i \rightarrow \rho(\vec{g}_{k,l,m})$
2. solve Poisson's equation on grid $\Phi(\vec{g}_{k,l,m})$
3. differentiate potential to get forces $\vec{F}(\vec{g}_{k,l,m})$
4. interpolate forces back to particles $\vec{F}(\vec{g}_{k,l,m}) \rightarrow \vec{F}(\vec{x}_i)$

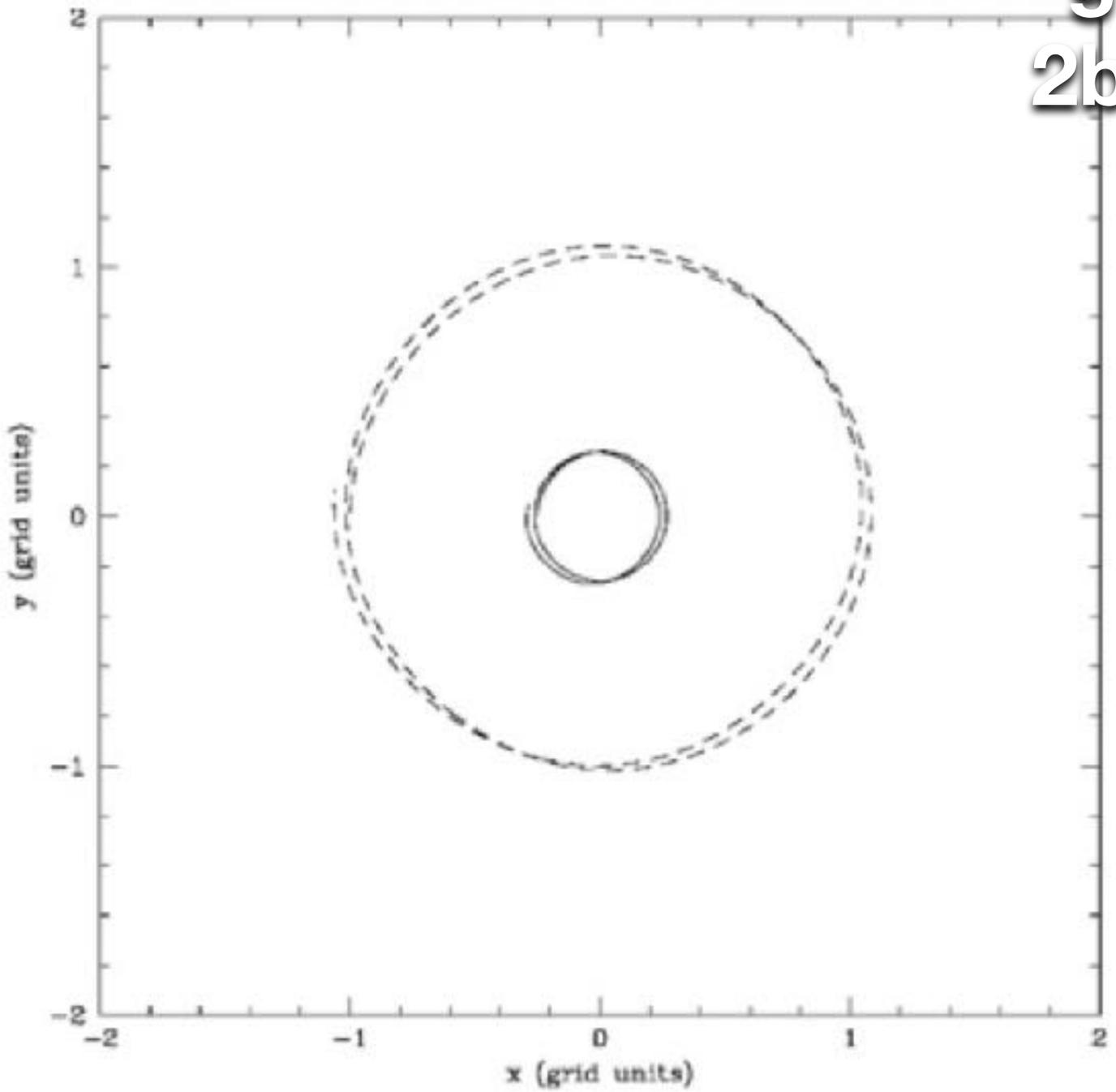
sounds like a waste of time and computer resources,
but **exceptionally fast** in practice

SomePM caveats

- FFT is very efficient
- Regular grid wastes computation time, if the dynamical range is large (from galaxies to the universe, from stars-to galaxies)
- Nested PM grids?
- Aliasing effects
- Careful that the Grid-Box-N of particles do not trigger artificial structure (**your project)**

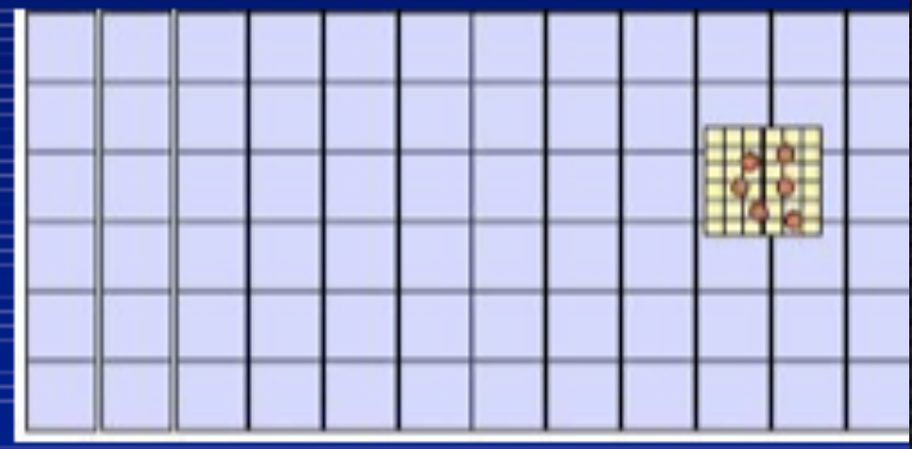
Force + integrator accuracy

good old
2body test



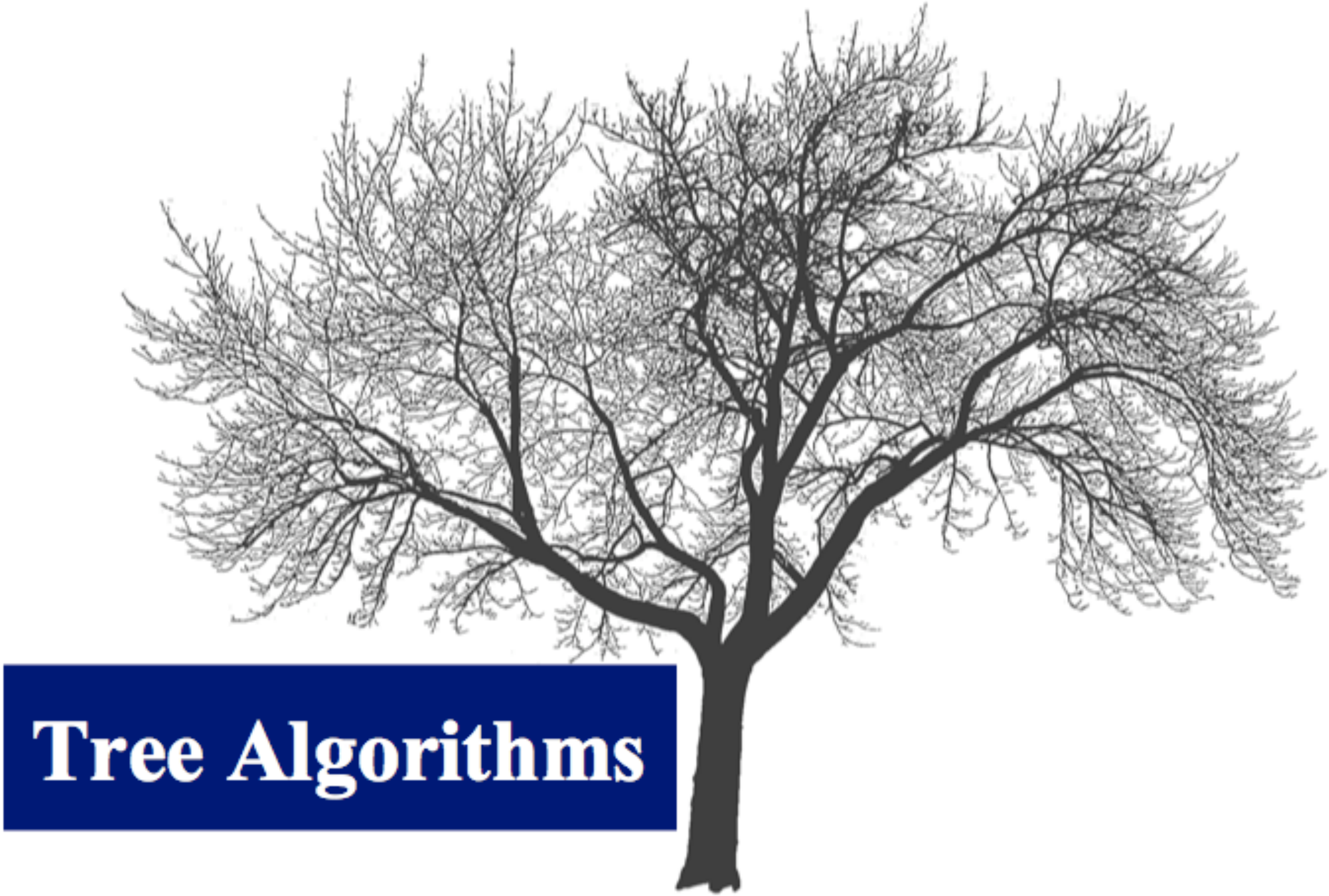
examples of particle's trajectory

The Particle-Particle Particle-Mesh (P^3M) Scheme:



- This extends the PM method with a direct summation of short-range forces on the scale of the mesh cells.
  - GOOD: Increases substantially the spatial dynamic range
  - BAD: Highly-clustered states (when the short range force calculations dominate) are very costly to evolve
- Mesh-refinements may be placed on clustered regions (as in the AP^3M and Hydra codes developed by Hugh Couchman et al) but this suffers from high complexity (which makes it very difficult to parallelize) and ambiguities on the optimal choice of refinement placement.

look Marios´s
and Peder´s talks



Tree Algorithms

Tree algorithms

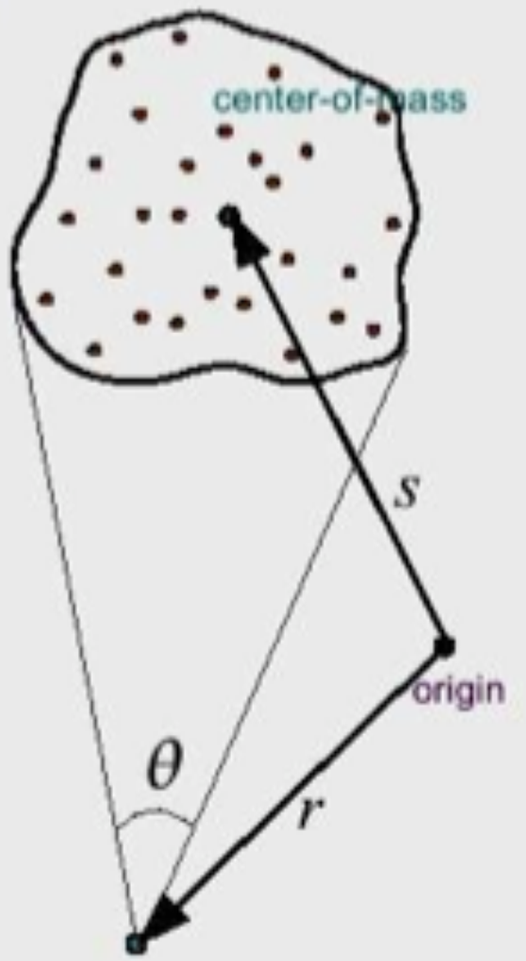
- The idea here is to use the fact that the contribution to the potential of far-away masses is given by the first few terms of a multipole expansion.

$$\Phi(\mathbf{r}) = -G \sum_i \frac{m_i}{|\mathbf{r} - \mathbf{x}_i|}$$

We expand:

$$\frac{1}{|\mathbf{r} - \mathbf{x}_i|} = \frac{1}{|(\mathbf{r} - \mathbf{s}) - (\mathbf{x}_i - \mathbf{s})|}$$

for $|\mathbf{x}_i - \mathbf{s}| \ll |\mathbf{r} - \mathbf{s}|$ $\mathbf{y} \equiv \mathbf{r} - \mathbf{s}$



- And get:

dipole contribution vanishes outside each node

$$\frac{1}{|\mathbf{y} + \mathbf{s} - \mathbf{x}_i|} = \frac{1}{|\mathbf{y}|} - \frac{\mathbf{y} \cdot (\mathbf{s} - \mathbf{x}_i)}{|\mathbf{y}|^3} + \frac{1}{2} \frac{\mathbf{y}^T \left[3(\mathbf{s} - \mathbf{x}_i)(\mathbf{s} - \mathbf{x}_i)^T - \mathbf{I}(\mathbf{s} - \mathbf{x}_i)^2 \right] \mathbf{y}}{|\mathbf{y}|^5} + \ldots$$

GADGET - 2

A code for cosmological simulations of structure formation

look Juan Carlos Classes and projects

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.
- Smoothed particle hydrodynamics with fully adaptive smoothing lengths and a novel entropy conserving formulation of SPH.
- Signal-velocity parameterisation of the artificial viscosity, as suggested by Monaghan.
- Individual timesteps for all particles. In the TreePM scheme, long-range and short-range forces are integrated with different timesteps.
- Work-load balanced domain decomposition and dynamic tree updates.
- Efficient cell-opening criteria for the gravitational tree-walk.
- Support for parallel I/O and a number of different output formats, including the HDF5 format

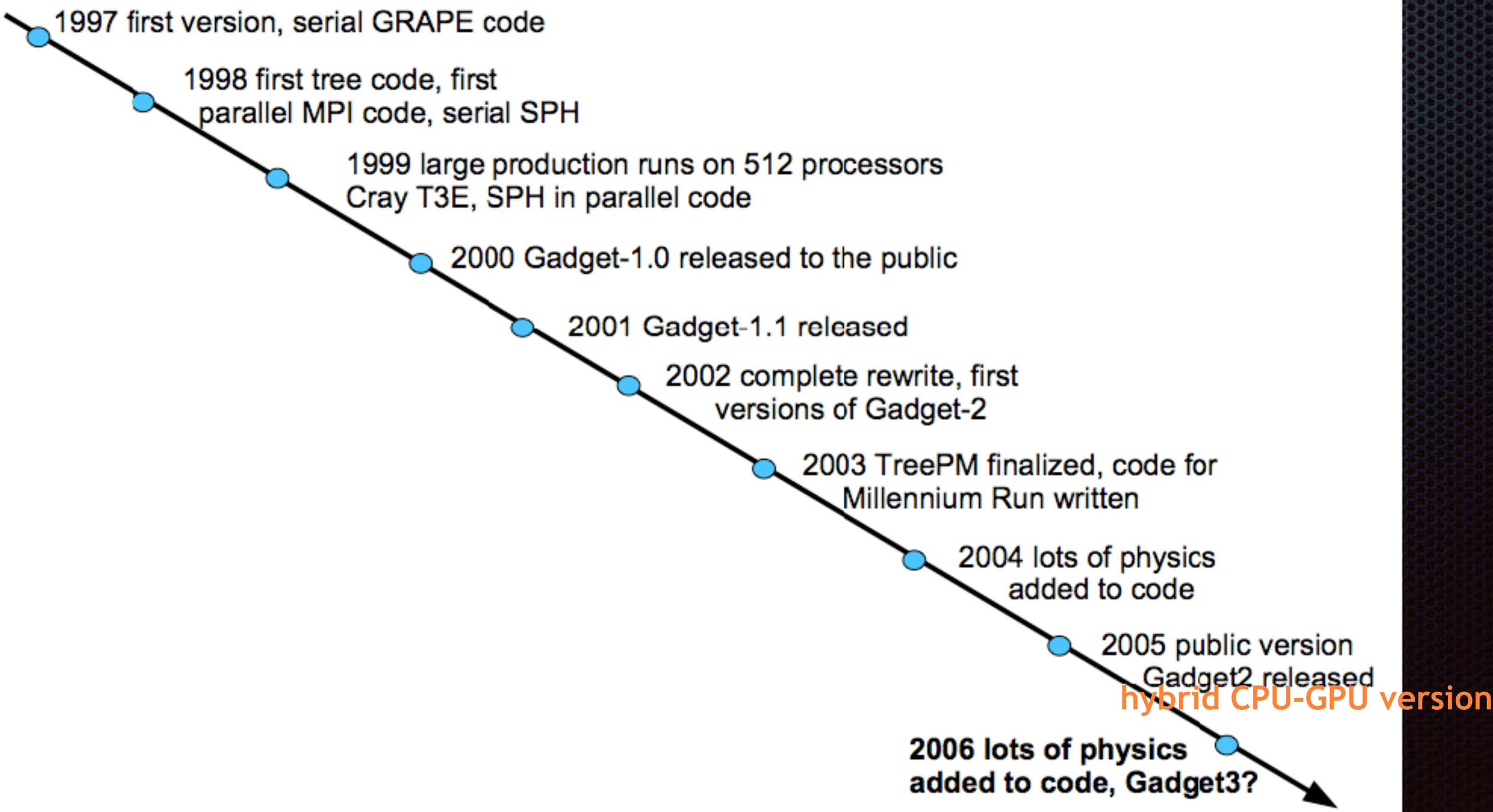
GADGET is a versatile TreeSPH N-body code for cosmological applications

PRINCIPLE CHARACTERISTICS OF GADGET

- Gravity solver based on a TREE or TreePM algorithm
- Hydrodynamics is followed by means of SPH
- Timesteps can be individual and adaptive
- Code is parallelized with MPI for distributed memory architectures
- Code is written in C and is highly portable
- A basic version of the code is publicly available

GADGET has evolved over the years and is in a process of continuous change

MAJOR EVENTS IN GADGET'S DEVELOPMENT



Gasoline

- N-body Solver (KD-Tree Method, PKDGRAV) and Smoothed Particle Hydrodynamics
- Physics: Gravity, Hydrodynamics, Atomic Chemistry (Radiative Heating, Cooling)
- Subgrid Physics: Star Formation, Supernova Feedback, Planetesimal Collisions

Wadsley, Stadel & Quinn 2003, Stadel Ph
Governato et al 2005, Nature

Likely the future in paralelization efficiency. CHARM++, P-Cello code

5. ChANga (Charm N-body GrAvity solver)

http://librarian.phys.washington.edu/astro/index.php/Research:ChaNGa
http://www-hpcc.astro.washington.edu/tools/changa.html

– Barnes-Hut tree code (where leaves can be > 1 particle and there is multipole expansion for far away particles)

– Softening not as Plummer but cubic spline

– In gravity only mode seems to work well for > 128k cores

– Multi-stepping for timesteps allowed (which does not mean it is smart to use it)

– Excellent work-load balance (= integrations and particles are well distributed between the used CPUs) thanks to
CHARM++ runtime system
http://charm.cs.uiuc.edu/software

The quest for higher resolution…

- as perturbations collapse, their size shrinks by orders of magnitude as matter collapses to overdensities in excess of 1e6…

- an additional shrinking can be due to expansion if integrating on a grid fixed in comoving coordinates

- fixed uniform grid methods are therefore not suitable to study halo structure in the highly nonlinear regime or galaxy formation

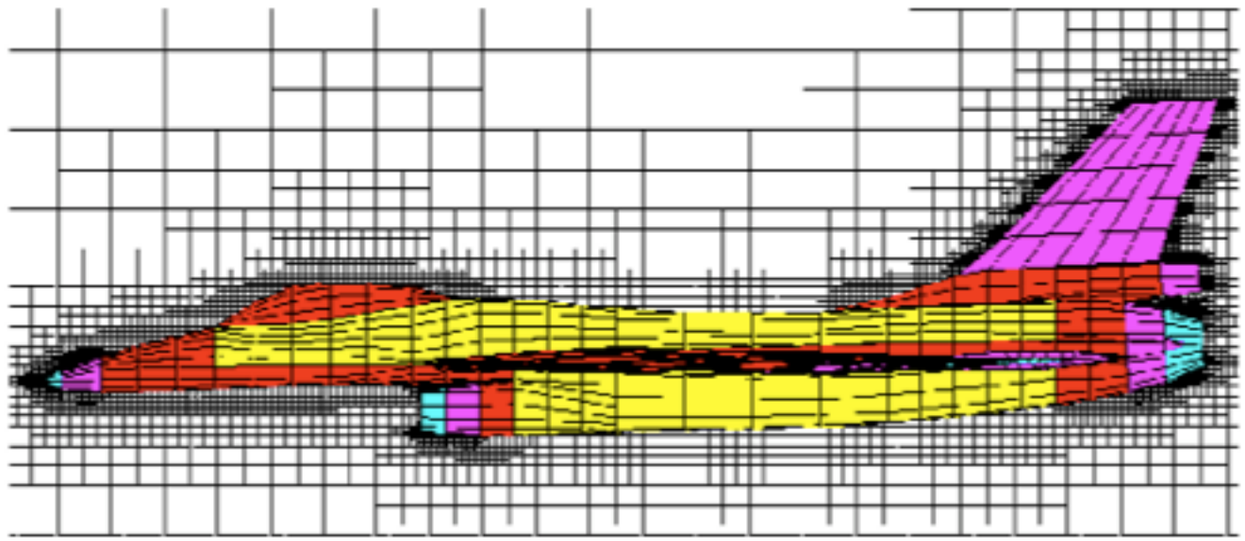
The AMR Approach

- Efficient, reliable finite element methods *for uniform grids* have been developed for solving the Poisson and gasdynamics equations.

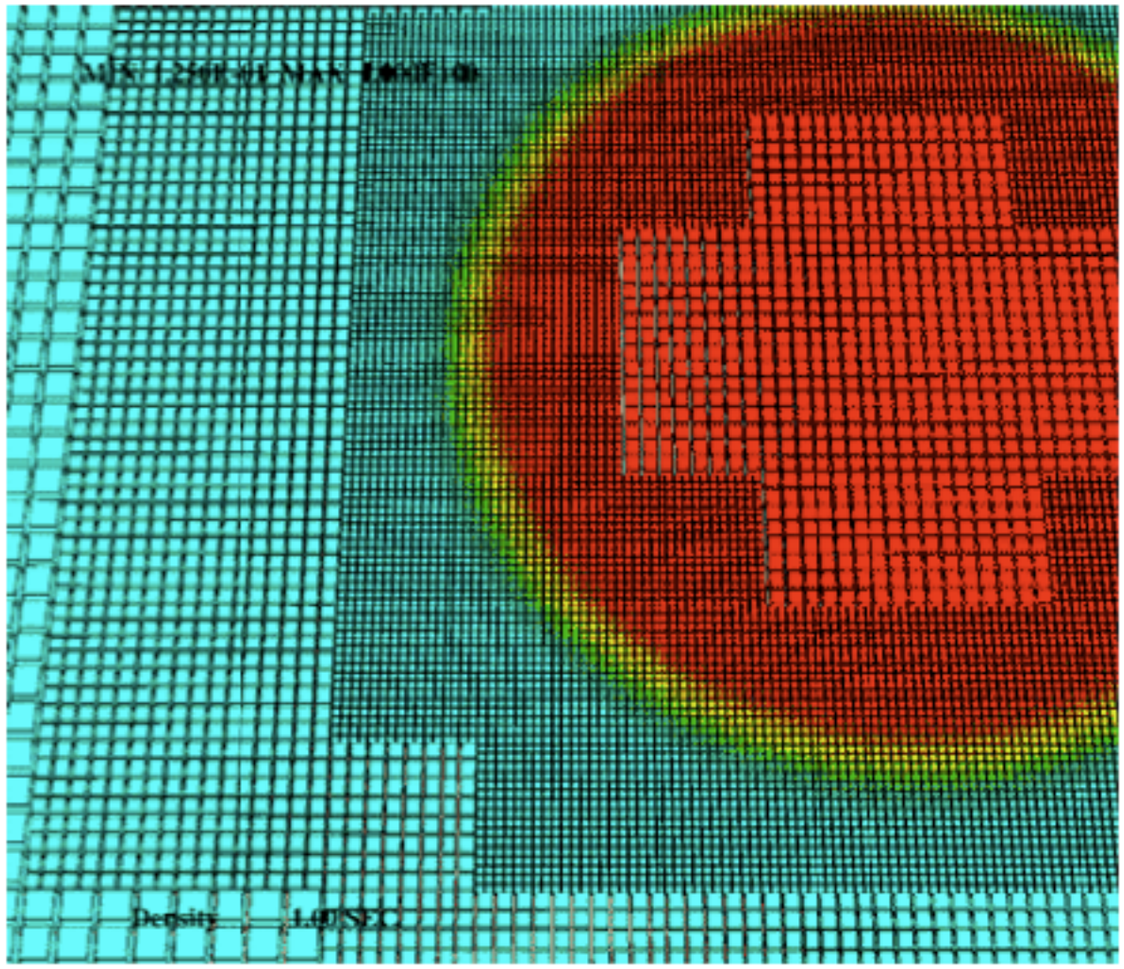
- The *Adaptive Mesh Refinement* (AMR) methods increase the dynamic range of grid-based numerical algorithms beyond the limits imposed by existing hardware.

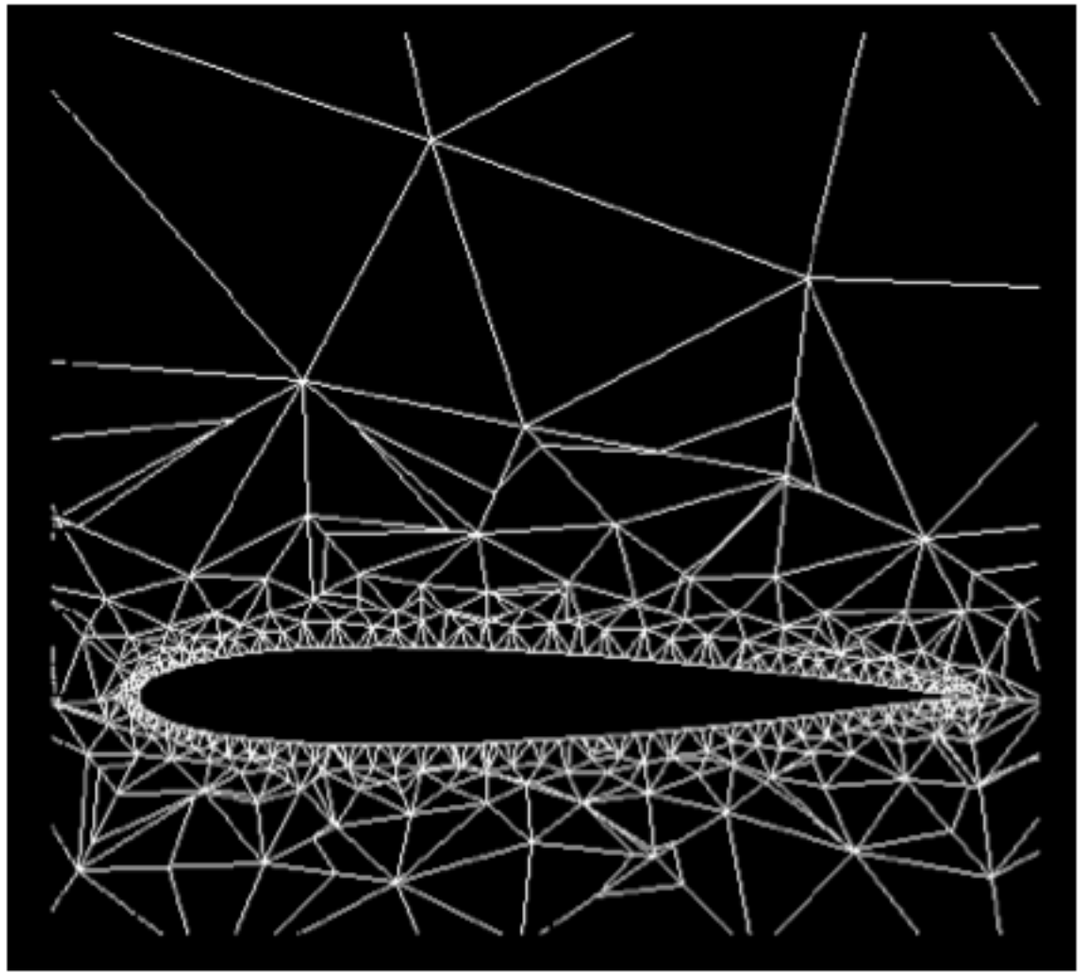
- The methods have numerous applications in different fields of physics, engineering, etc.

- Now gaining popularity in astrophysics and cosmology



Structured vs unstructured AMR



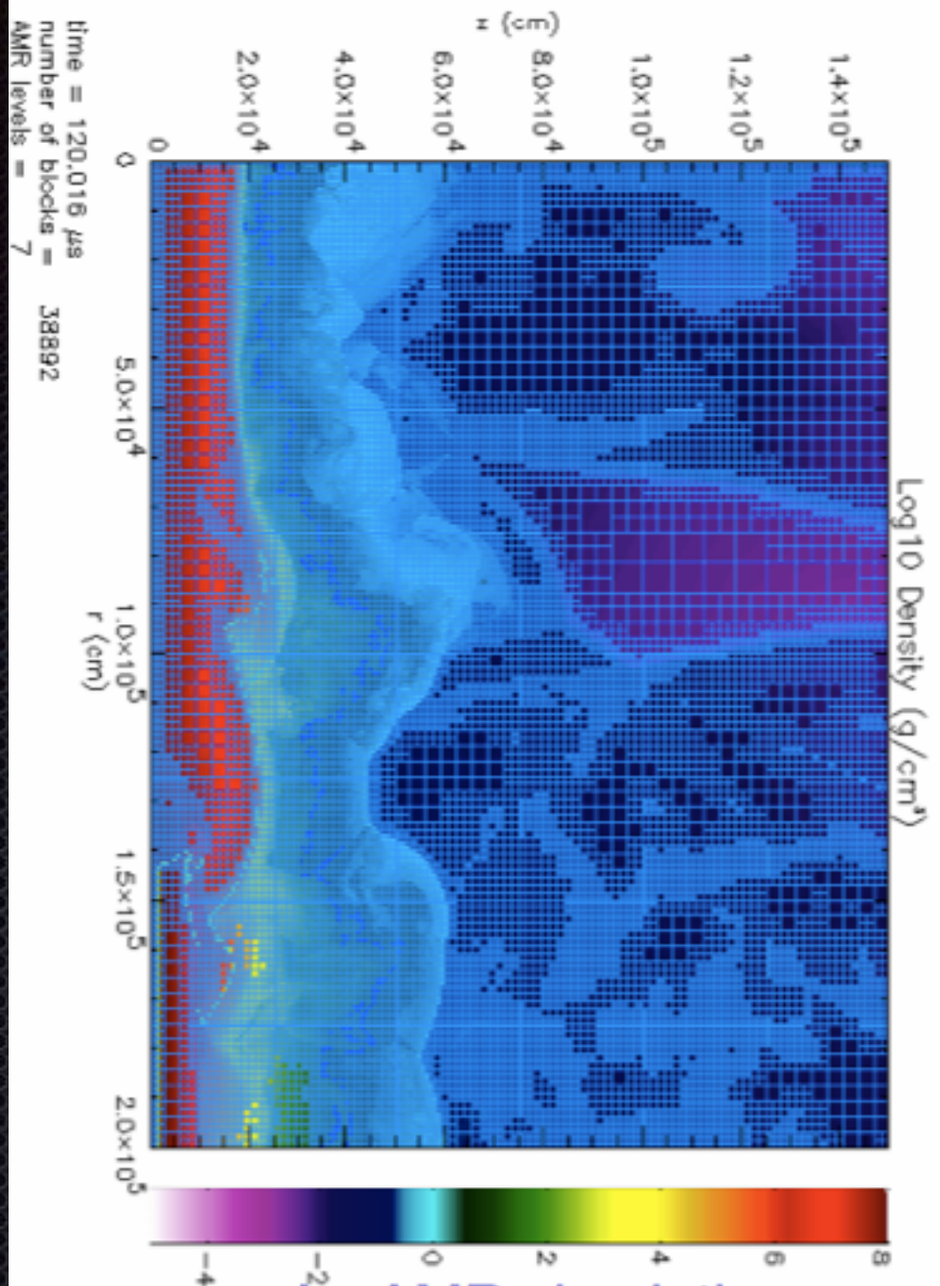


Structured: hierarchy of rectangular grids or irregularly shaped meshes of cubic cells

Unstructured: highly flexible refinement meshes, efficient for cases of complicated region geometry and boundaries; more sophisticated data structures and algorithms

Arepo uses something
Like this one

Refining cell by cell



An AMR simulation with the Flash code

- Taken to its limit, one can think of tree of individual cells or small groups of cells (quads – 4 cells in 2D, octs – 8 cells in 3D)
- In this case, the refinement can be controlled on the level of individual cells, which allows meshes with complicated geometries to match complicated features in the systems (shocks, filaments etc.)
- This approach is used in ART, FLASH, AMIGA, RAMSES

Cosmological and Galatic AMR codes

- AP^3M (N-body)

  H. Couchman 1991

  publicly available: *http://coho.mcmaster.ca/hydra/ap3m/ap3m.html*

- ENZO (N-body + gasdynamics)

  G. Bryan & M. Norman 1996

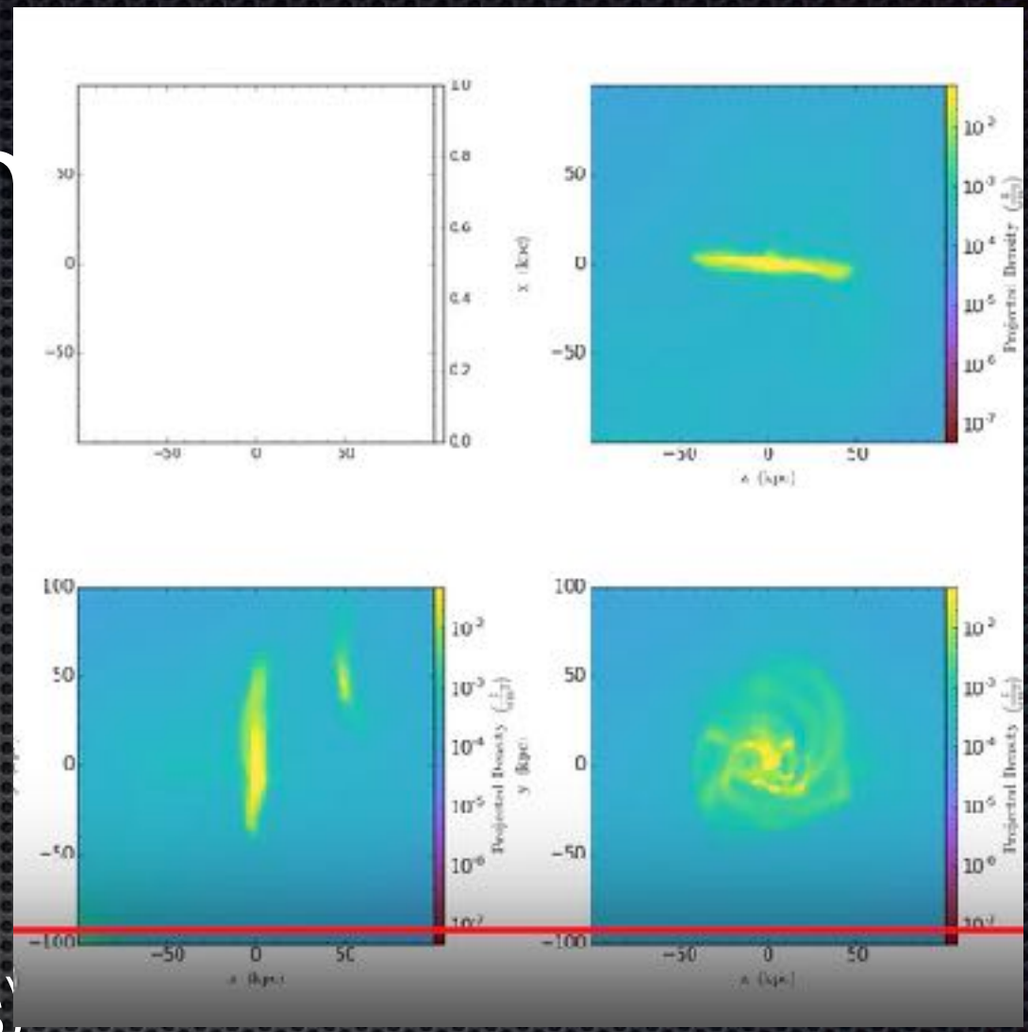
  publicly available: *cosmos.ucsd.edu/enzo/*

- Adaptive Refinement Tree (N-body + gasdynamics)

  Kravtsov, Klypin & Khokhlov 1997; Kravtsov 1999;

  Kravtsov, Klypin & Hoffman 2002

Cosmological an AMR codes



Includes SN,AGN, MHD, MG
MPI paralellization

RAMSES (N-body + gas dynamics)
R. Tesyer (http://irfu.cea.fr/Projets/Site_ramses/RAMSES.html)

This morning simulation…

http://popia.ft.uam.es/AMIGA/

MLAPM/AMIGA

- A publicly available AMR N-body code that uses PM and multigrid approaches to solve the Poisson equation and integrate particle trajectories

- Detailed description of the publicly available

In Alexander Knebe website

this morning after the break

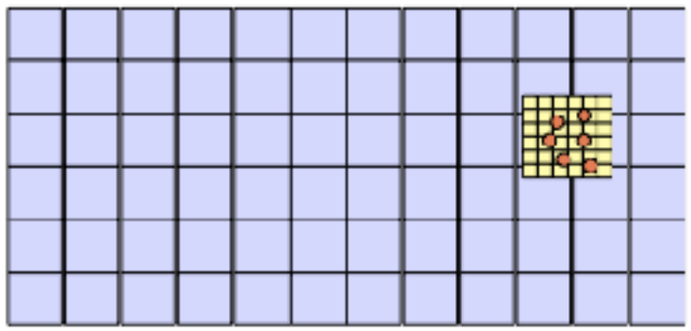
AP3M: HYDRA

Particle-Particle PM schemes (P^3M)

Idea: Supplement the PM force with a direct summation short-range force at the scale of the mesh cells. The particles in cells are linked together by a chaining list.

Offers much higher dynamic range, but becomes slow when clustering sets in.

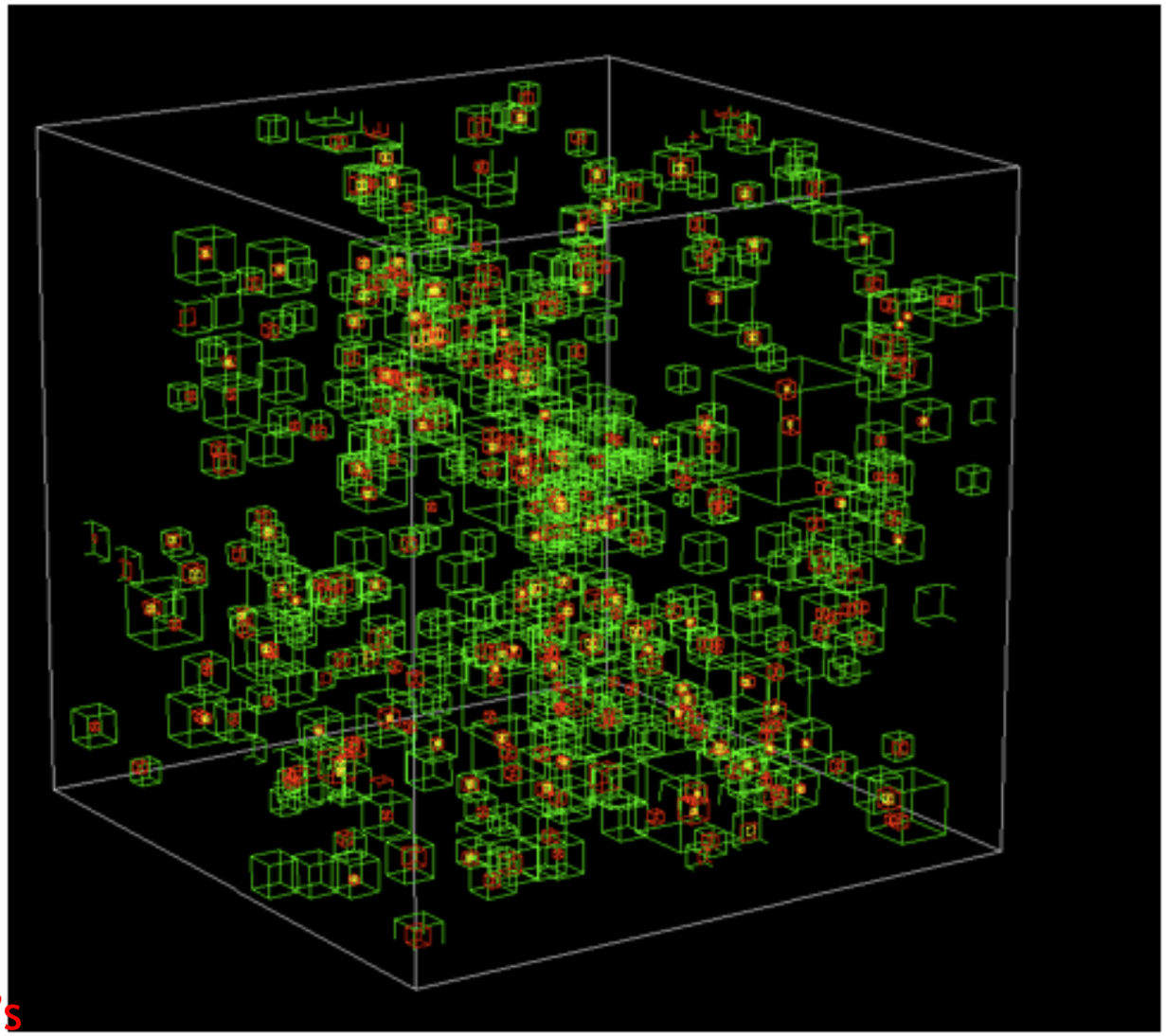
In AP^3M, mesh-refinements are placed on clustered regions



Can avoid clustering slow-down, but has higher complexity and ambiguities in mesh placement

Codes that use AP^3M: **HYDRA** (Couchman)

AP^{3}M code (Adaptive P^{3}M)



GPU's
May give
A new chance to this
Kind of code

subgrids introduced adaptively in an AP^{3}M simulation around collapsing halos

- P^{3}M algorithm achieves higher resolution by complementing low-res PM force, with higher resolution small-separation force calculated via direct PP Calculation (hence the name: PPPM or P^{3}M)

- as matter collapses into halos PP part of the calculation carries larger and larger burden. Simulation slows down or stalls.

- Idea behind AP^{3}M: reduce PP load by using PM force calculation on subgrids introduced in high particle density regions

Critical for any adaptive Nbody method: avoid jumps from one accuracy level to another

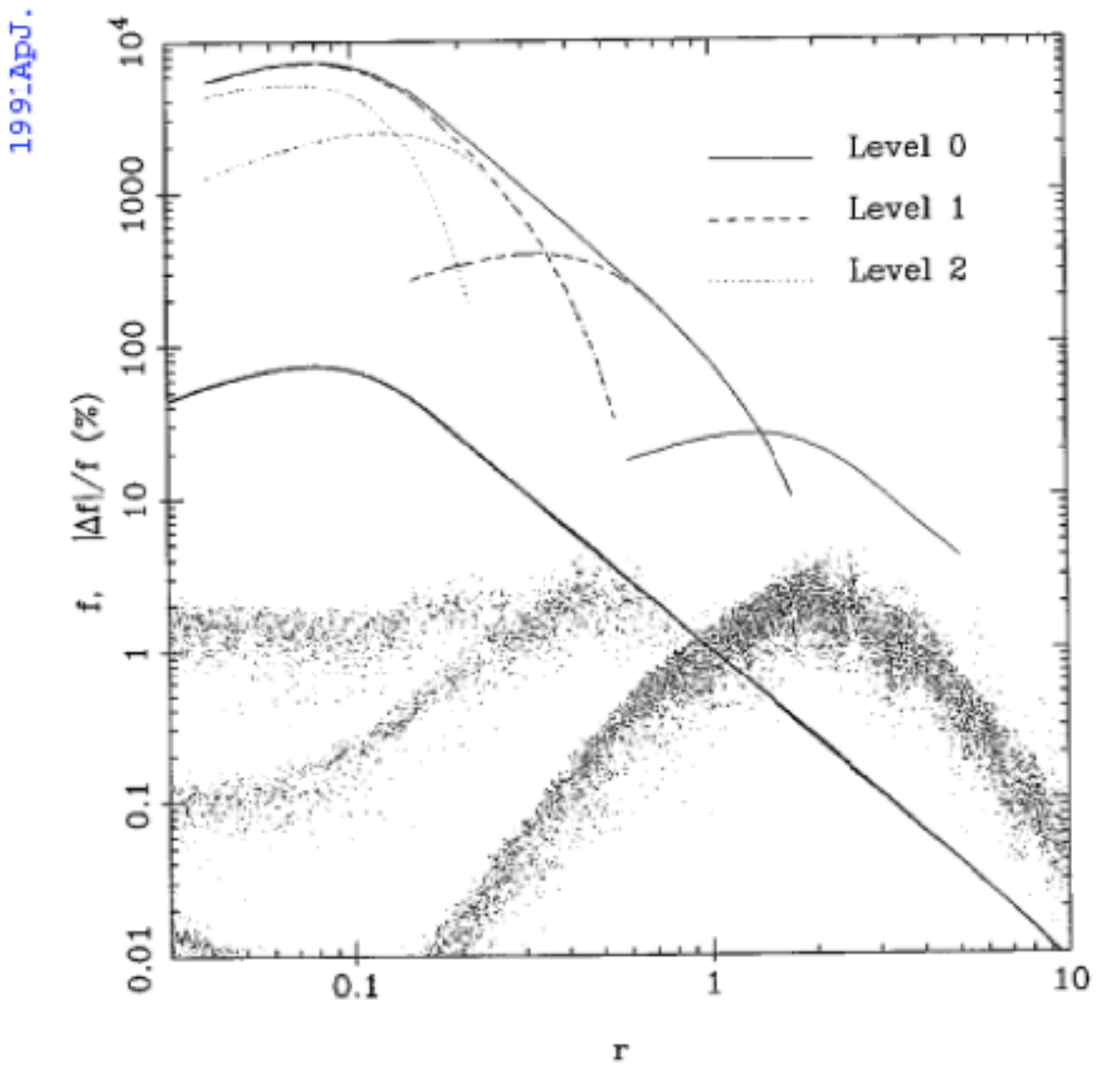


FIG. 1.—The pairwise force as a function of radius calculated on a 32^3 periodic mesh with two levels of refinement. The heavy continuous line is the calculated force, and the dots indicate the modulus of the error in the force. The curves at the top of the plot indicate the components of the force (× 100) calculated from the base mesh (level 0), the refinements, and the corresponding direct sums.

Couchman 1991

- Gravity force in the AP^{3}M code is stitched together from different grids and PP calculation at the smallest separations

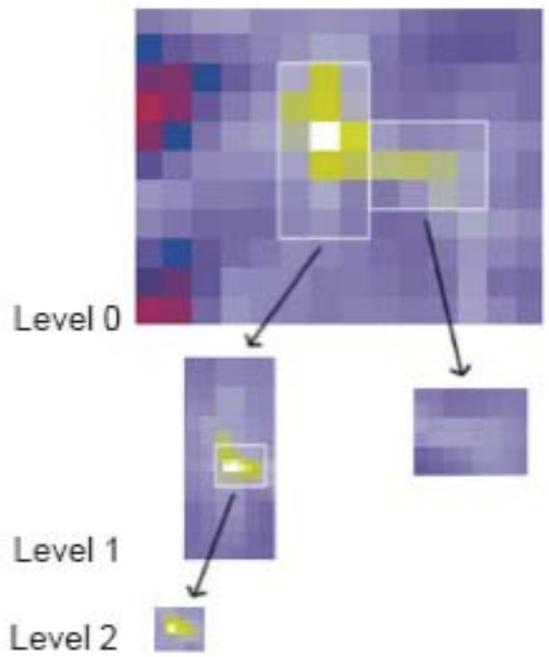
- The same gravity solver (albeit w/o PP part) is used in another AMR code – Enzo

refinement criteria AMR

opening angle for Tree

Precursor: Zeus
Descendent: P-Cello (ask me)

Enzo



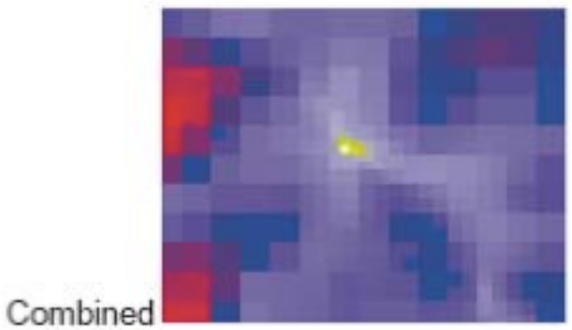


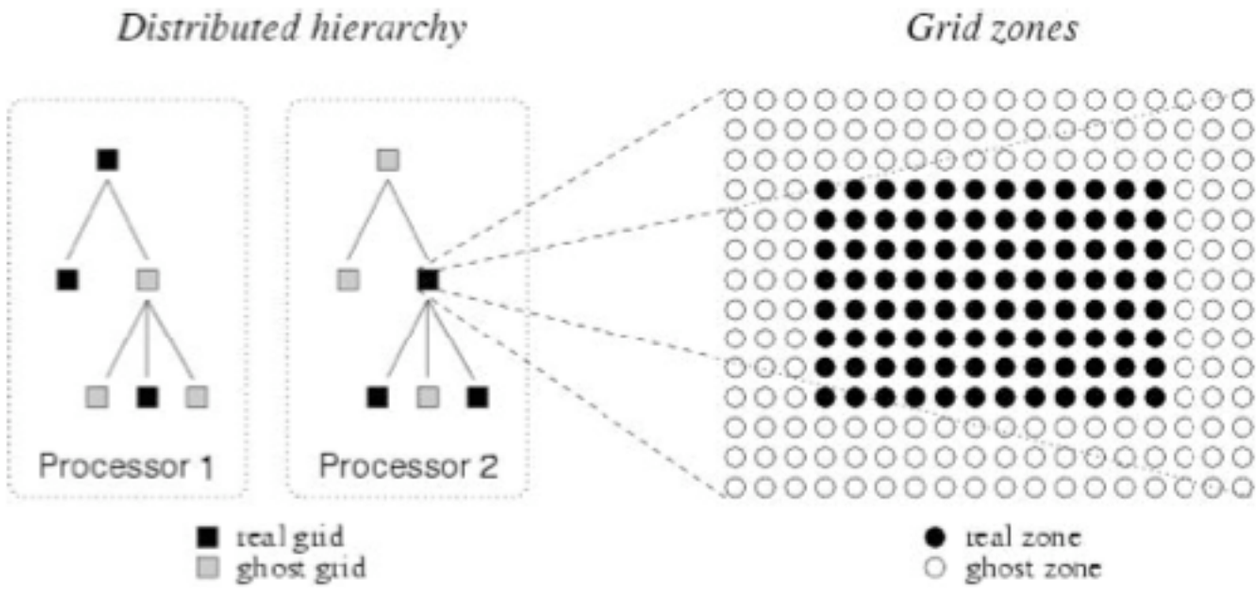
Figure 1. A 2D adaptive mesh refinement example showing a three-level, four-grid hierarchy.

- The first AMR N-body + Eulerian gasdynamics code in cosmology

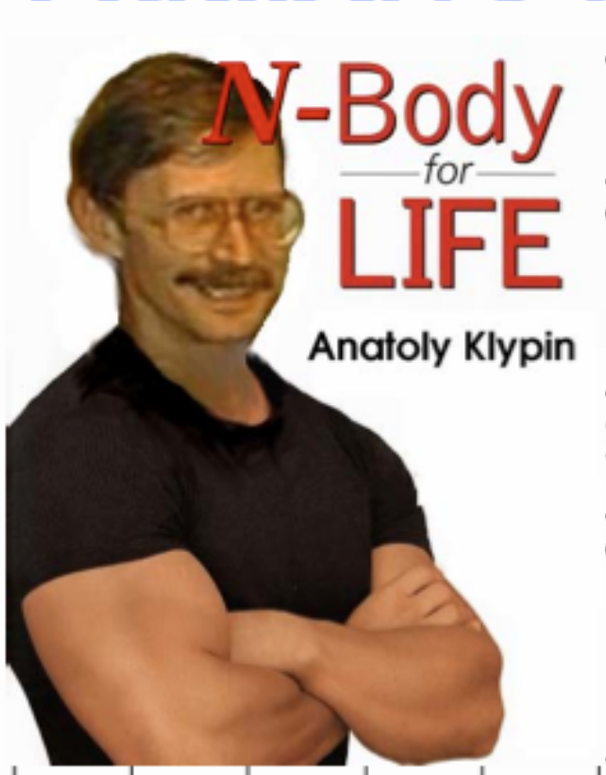
- Initially developed by Greg Bryan & Michael Norman in 1996-1998 at UIUC and NCSA. Currently is developed and maintained by M. Norman's group at UC San Diego.

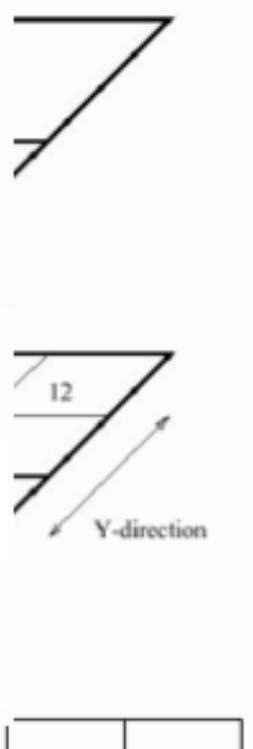
Refines cell blocks not only cells

- *publicly available (code and documentation: cosmos.ucsd.edu/enzo*

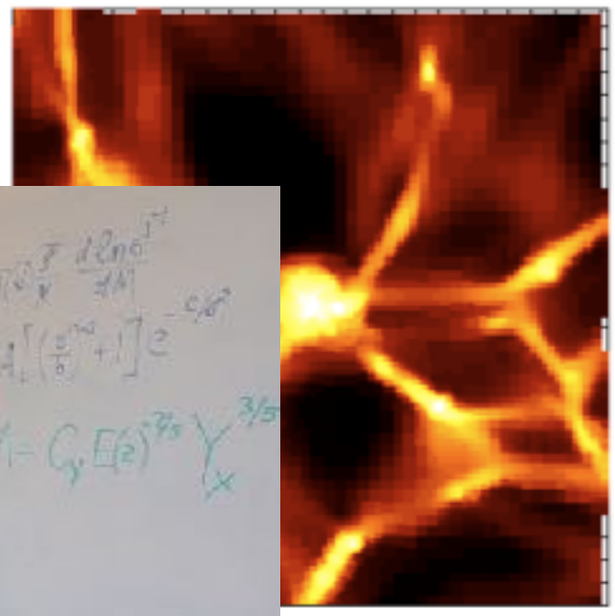


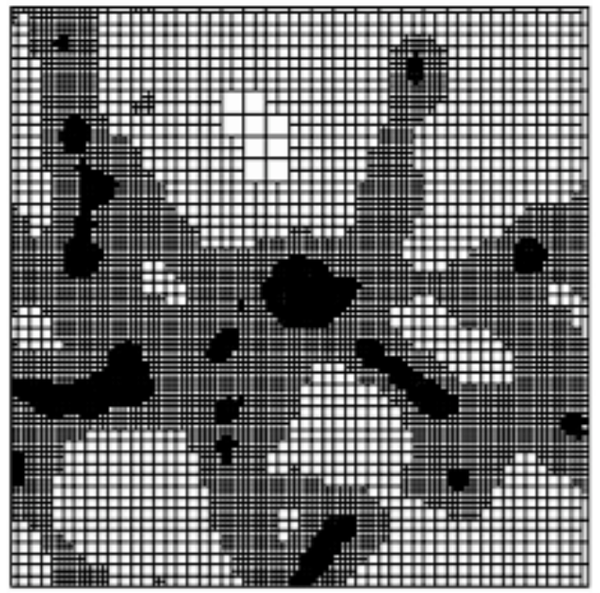
Adaptive Refinement Tree (ART) code



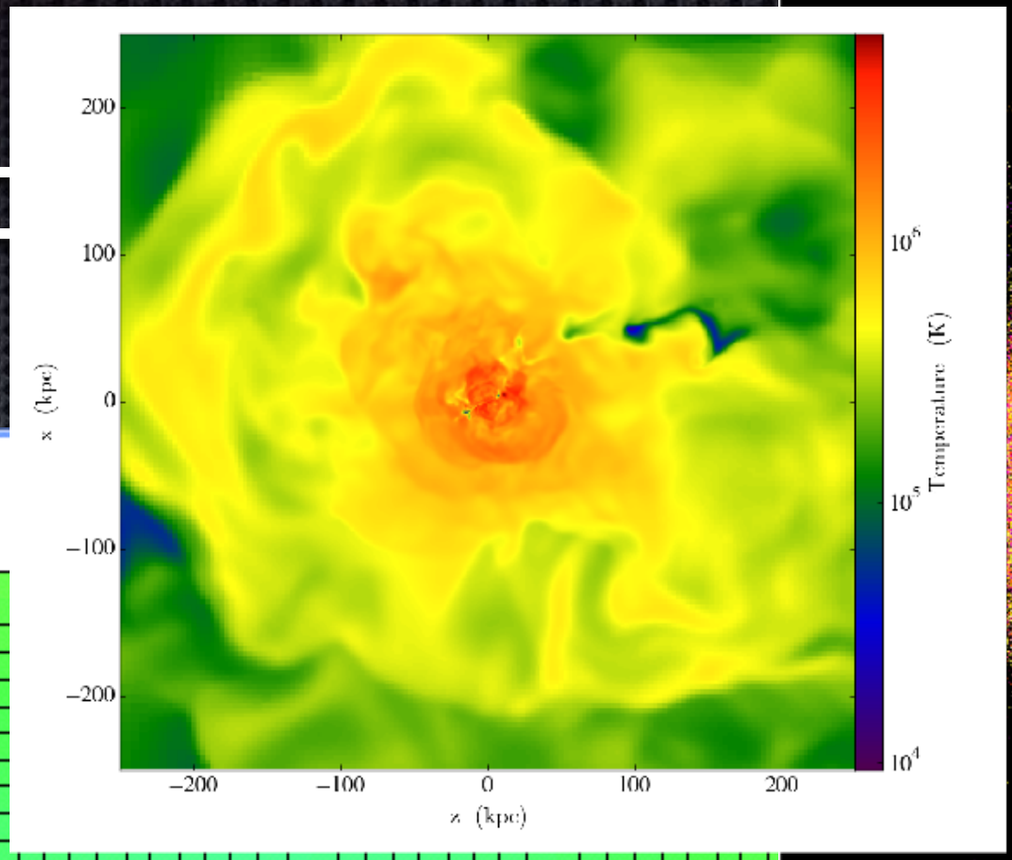


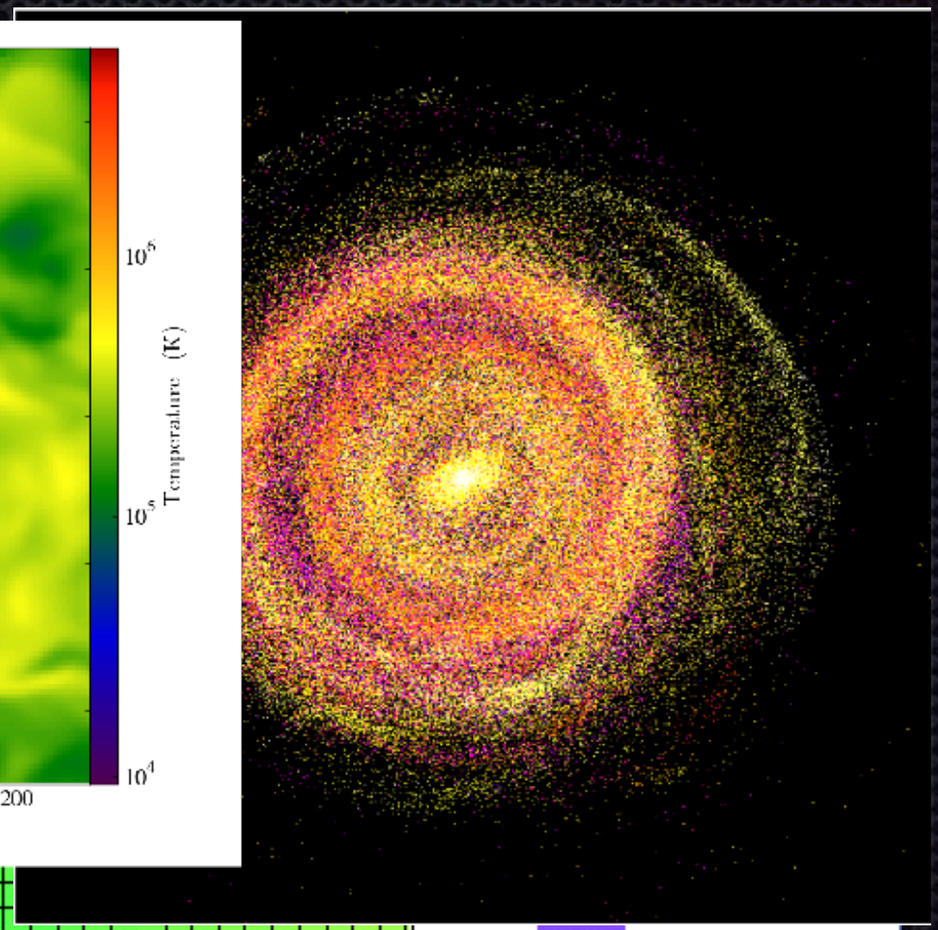
- The ART code *refines (and derefines) mesh cells individually*.
- We use a fully-threaded oct tree data structure (hence, the ART name) to support the refinement mesh hierarchy. The cost is only 2.5 storage words per cell. [Khokhlov 1998]
- This allows for flexible adaptive refinement structure that can be easily modified. The meshes can effectively match the complex geometry of filaments, sheets, and clumps in a cosmological simulation.





Hydro AR


200
100
0
−100
−200
x (kpc)
−200
−100
0
100
200
z (kpc)
10^5
10^5
10^4
Temperature (K)



31,533
31,741
32,374
32,165
31,957
31,749
1,33e+05
4,85e+04
1,76e+04
6,43e+03
2,2e+03
700
280
100

Frames
Legend
Pur Film
450
HD_00002_x,v HD

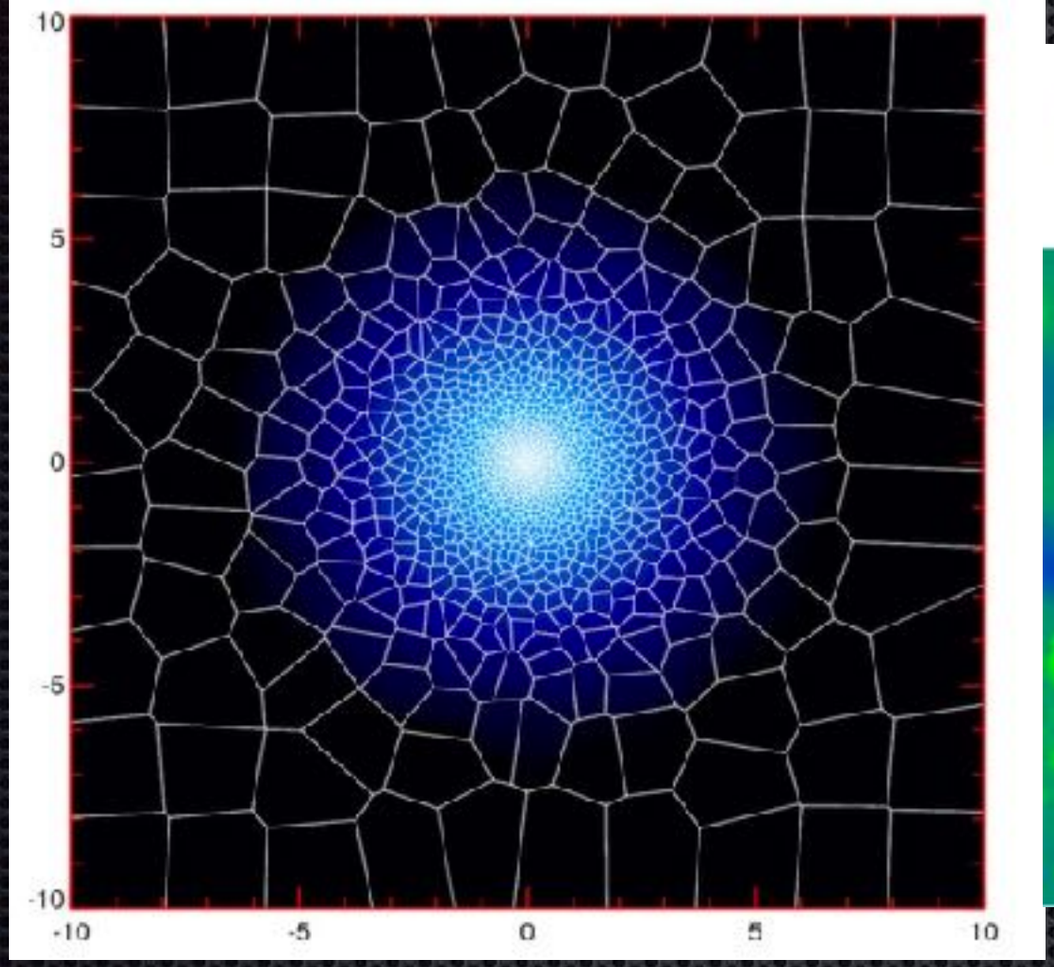
Paper arXiv:0901.4107

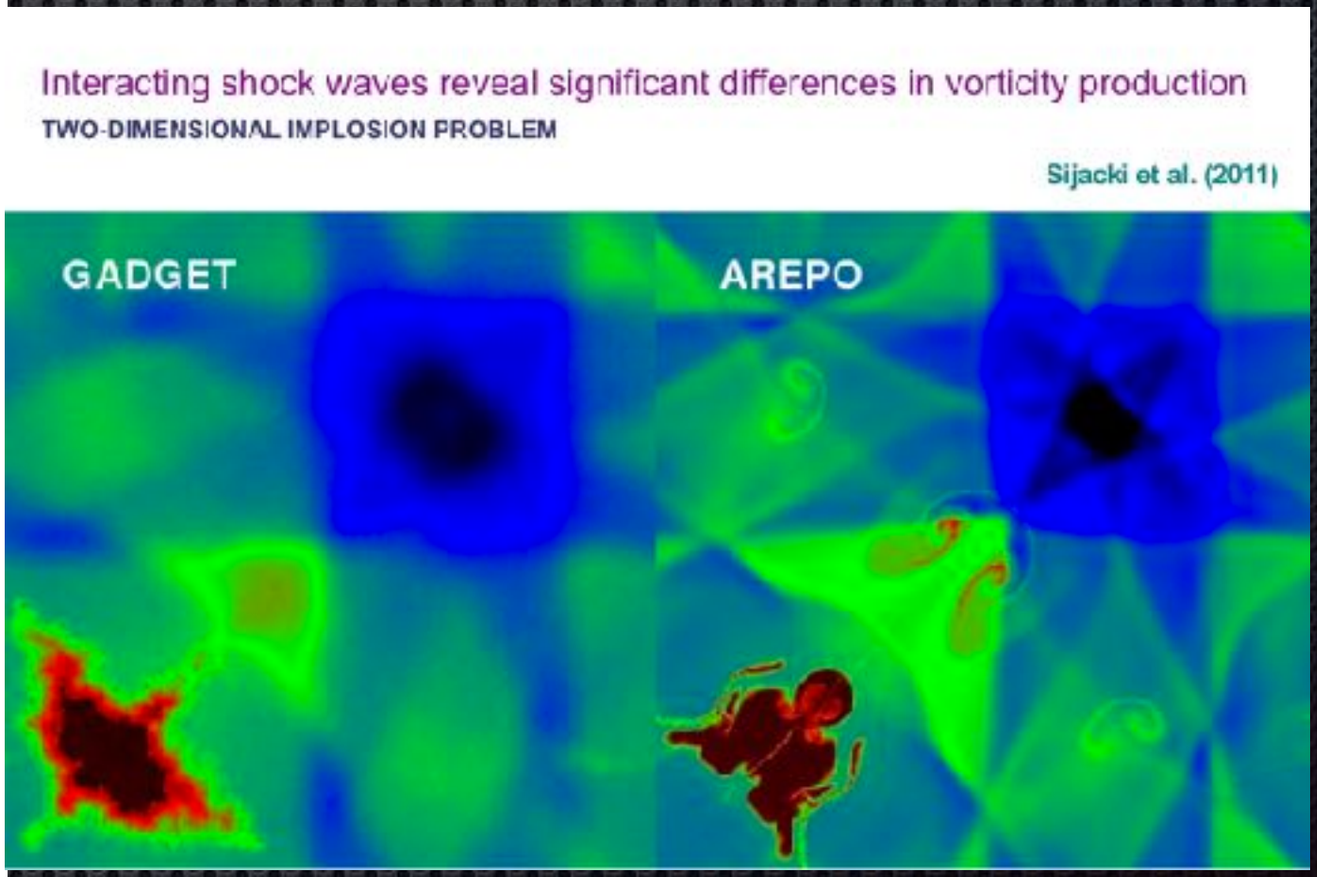
E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh

AREPO

Authors: Volker Springel (MPA)
Comments: accepted in MNRAS, 67 pages, 50 figures.

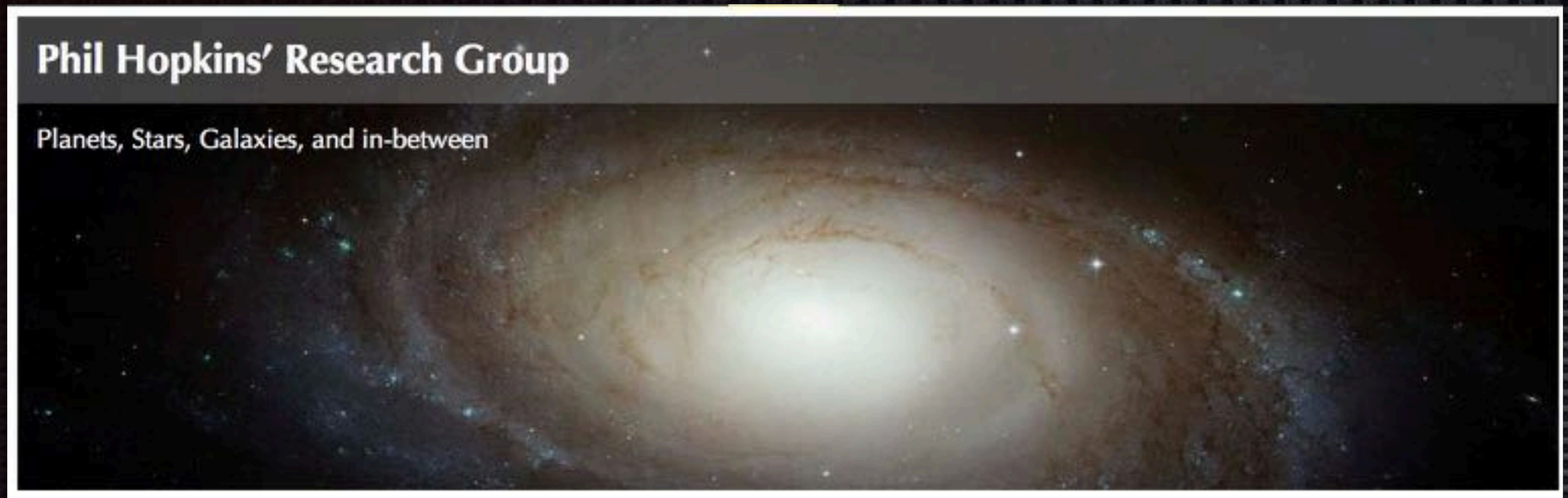
Hydrodynamic cosmological simulations at present usually employ either the Lagrangian smoothed particle hydrodynamics (SPH) technique, or Eulerian hydrodynamics on a Cartesian mesh with (optional) adaptive mesh refinement (AMR). Both of these methods have disadvantages that negatively impact their accuracy in certain situations, for example the suppression of fluid instabilities in the case of SPH, and the lack of Galilean-invariance and the presence o overmixing in the case of AMR. We here propose a novel scheme which largely eliminates these weaknesses. It is based on a moving unstructured mesh defined by the Voronoi tessellation of a set of discrete points. The mesh is use solve the hyperbolic conservation laws of ideal hydrodynamics with a finite volume approach, based on a second-order unsplit Godunov scheme with an exact Riemann solver. The mesh-generating points can in principle be moved arbitrarily. If they are chosen to be stationary, the scheme is equivalent to an ordinary Eulerian method with second order accuracy. If they instead move with the velocity of the local flow, one obtains a Lagrangian formulation of contin hydrodynamics that does not suffer from the mesh distortion limitations inherent in other mesh-based Lagrangian schemes. In this mode, our new method is fully Galilean-invariant, unlike ordinary Eulerian codes, a property that is of significant importance for cosmological simulations where highly supersonic bulk flows are common. In addition, the new scheme can adjust its spatial resolution automatically and continuously, and hence inherits the principal advanta of SPH for simulations of cosmological structure growth. The high accuracy of Eulerian methods in the treatment of shocks is also retained, while the treatment of contact discontinuities improves. We discuss how this approach is implemented in our new code AREPO, both in 2D and 3D, and is parallelized for distributed memory computers. We also discuss techniques for adaptive refinement or derefinement of the unstructured mesh. We introduce an individu time-step approach for finite volume hydrodynamics, and present a high-accuracy treatment of self-gravity for the gas that allows the new method to be seamlessly combined with a high-resolution treatment of collisionless dark matter use a suite of test problems to examine the performance of the new code and argue that the hydrodynamic moving-mesh scheme proposed here provides an attractive and competitive alternative to current SPH and Eulerian techniqu



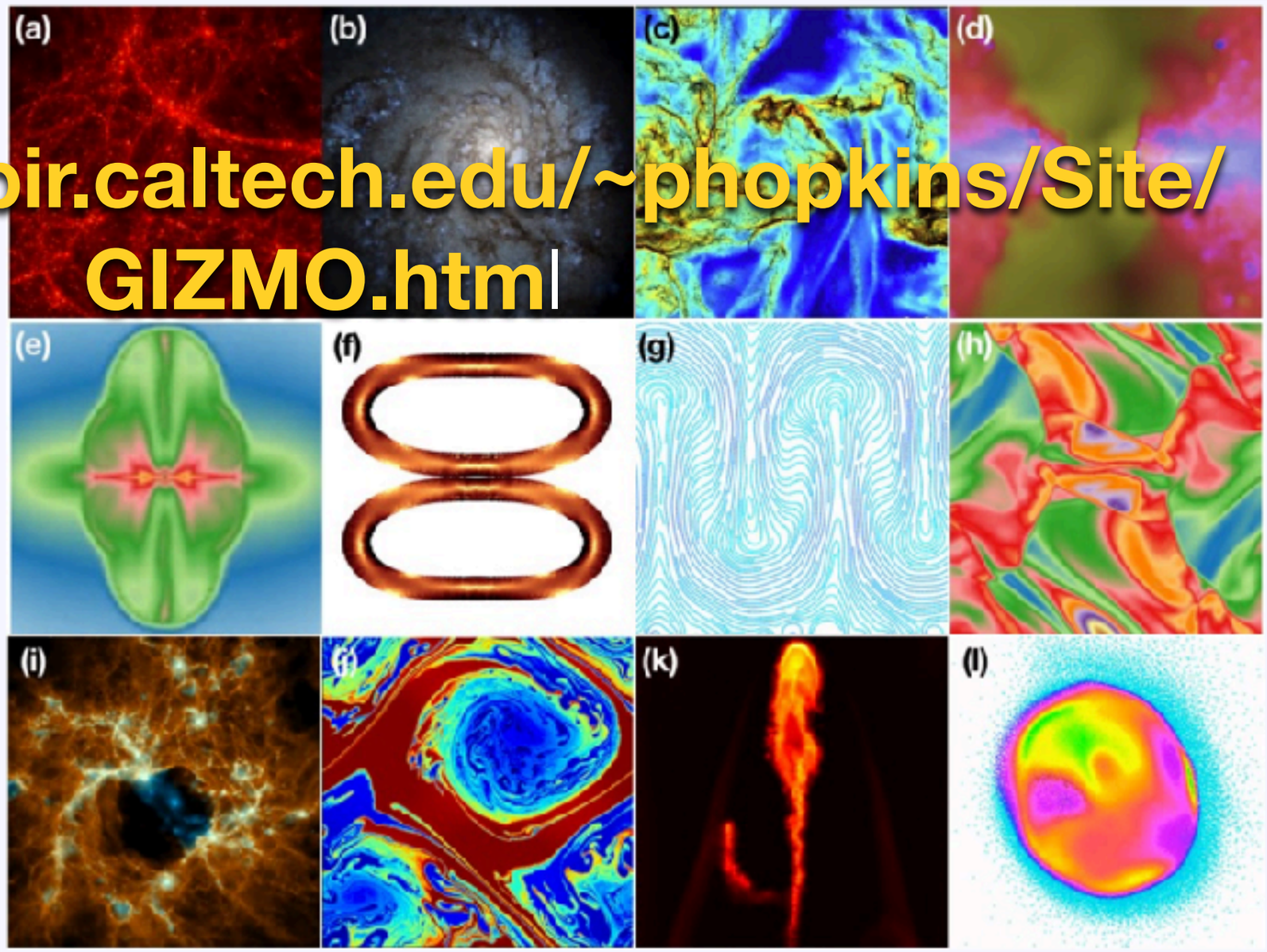


Phil Hopkins' Research Group

Planets, Stars, Galaxies, and in-between



GIZMO



http://www.tapir.caltech.edu/~phopkins/Site/GIZMO.html

Welcome!

Solvers

❑ Gravity solver:

FFT solver on the uniform (l=0) grid covering the entire volume
relaxation solver for l>0 [Gauss-Seidel + SOR + Chebyshev accel]
potential on child refinement cells is inherited from the parent cells
potential from the previous step is used as initial guess for the next step

❑ Eulerian gasdynamics solver:

2nd order Godunov solver with piecewise linear reconstruction
[van Leer 1979; Collela & Glaz 1985; Khokhlov 1998]

❑ **Particles:** are treated using standard particle -mesh methods
cloud in cell density assignment and force interpolation
2nd order leapfrog time integration, interpolation and loss of 2nd order accuracy at the refinement mesh interfaces…

❑ **Time integration** on each refinement level l is done with time step $dt_l = dt_0 / 2^l$, where dt_0 is the global time step on $l=0$, set so that the *CFL condition* is satisfied for cells on all levels

Relaxation

ART gravity solver

Relaxation

when you have a nonlinear poisson or can not use spectral method

Poisson equation

$$\nabla^2\phi = \rho \longrightarrow \frac{\partial\phi}{\partial\tau} = \nabla^2\phi - \rho. \quad \textit{relaxation equation}$$

relaxation iteration:

$$\phi_i^{n+1} = \phi_i^n + \frac{\Delta\tau}{\Delta^2}\left(\sum_{j=1}^{6}\phi_{Nb(j)}^n - 6\phi_i^n\right) - \rho_i\Delta\tau$$

- although in general relaxation is not a fast method, we have no choice with irregular meshes of ART…

- However, convergence of relaxation iteration is much much faster if we use potential solution from the previous step as initial guess to initialize the relaxation

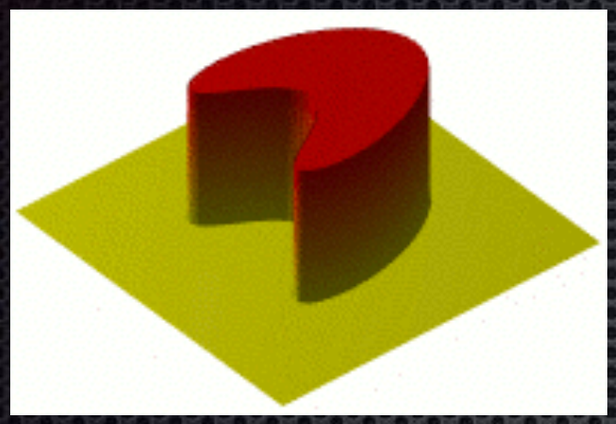
- red-black Gauss-Seidel relaxation scheme with successive over-relaxation (SOR) further speeds up the convergence (Hockney & Eastwood 1988; also, Numerical Recipes)

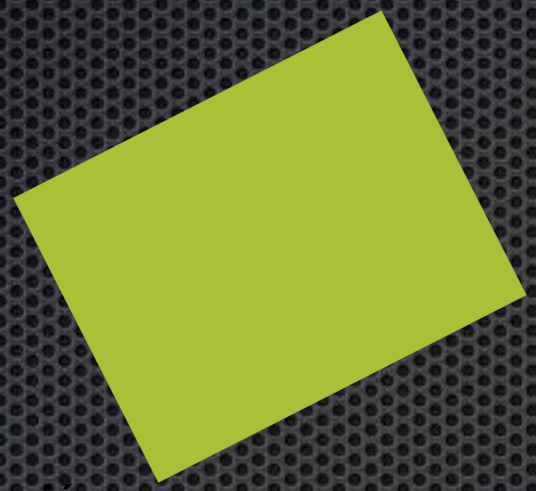
$$\phi_*^{n+1} = \omega\phi^{n+1} + (1-\omega)\phi^n,$$

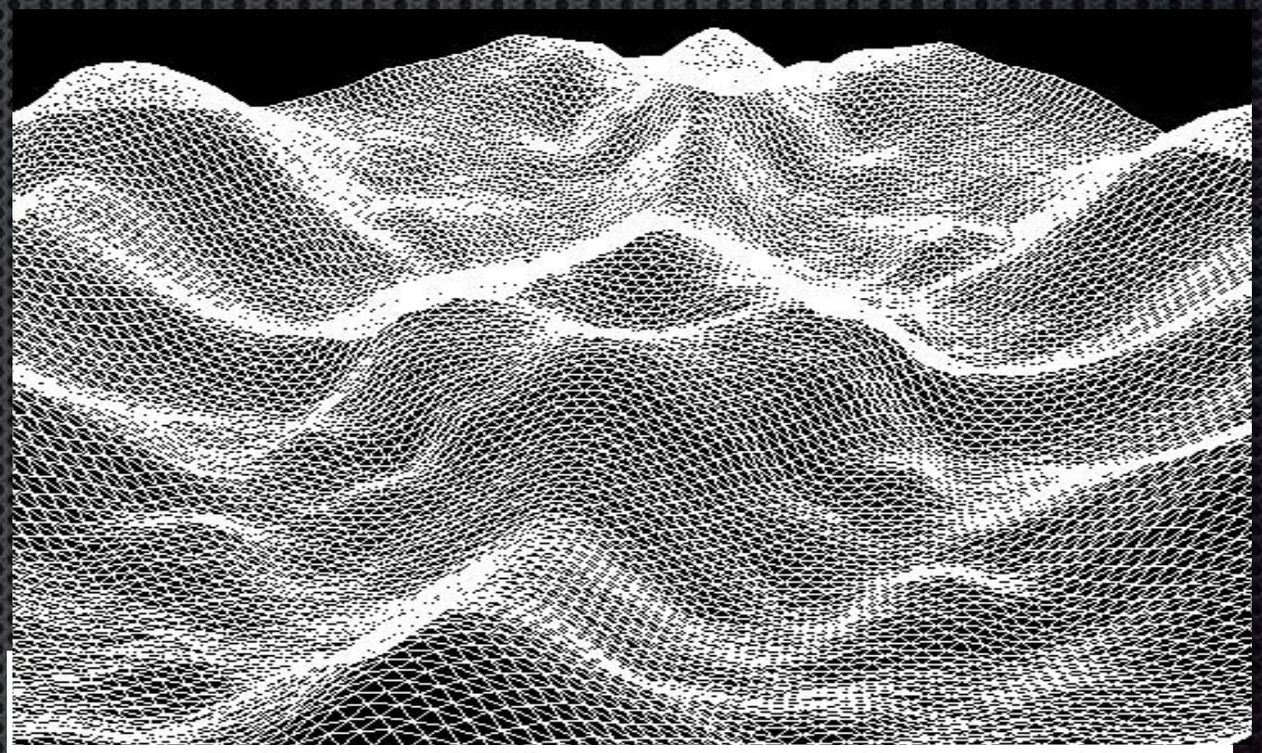
Problema de optimización: Técnica de relajación.

Ec. de Calor.

$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0$$

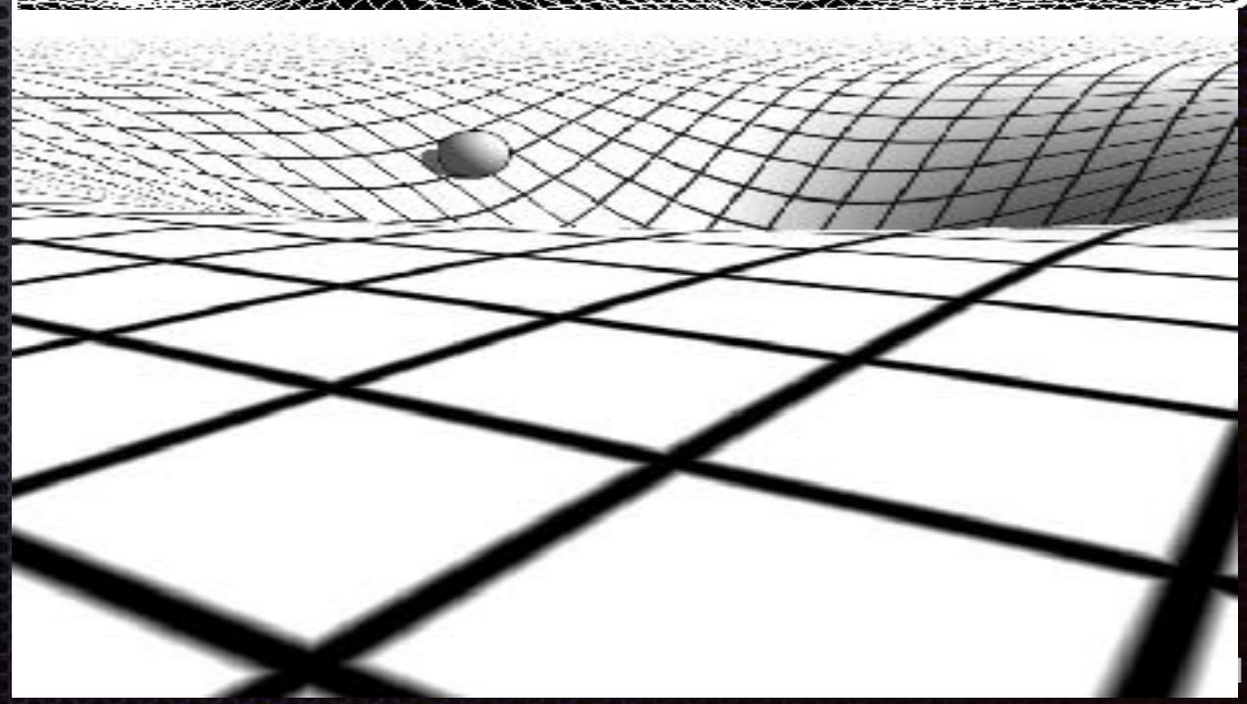






Ec. de Difusión.

$$\frac{\partial \psi(\mathbf{r},t)}{\partial t} = D\nabla^2 \psi(\mathbf{r},t)$$

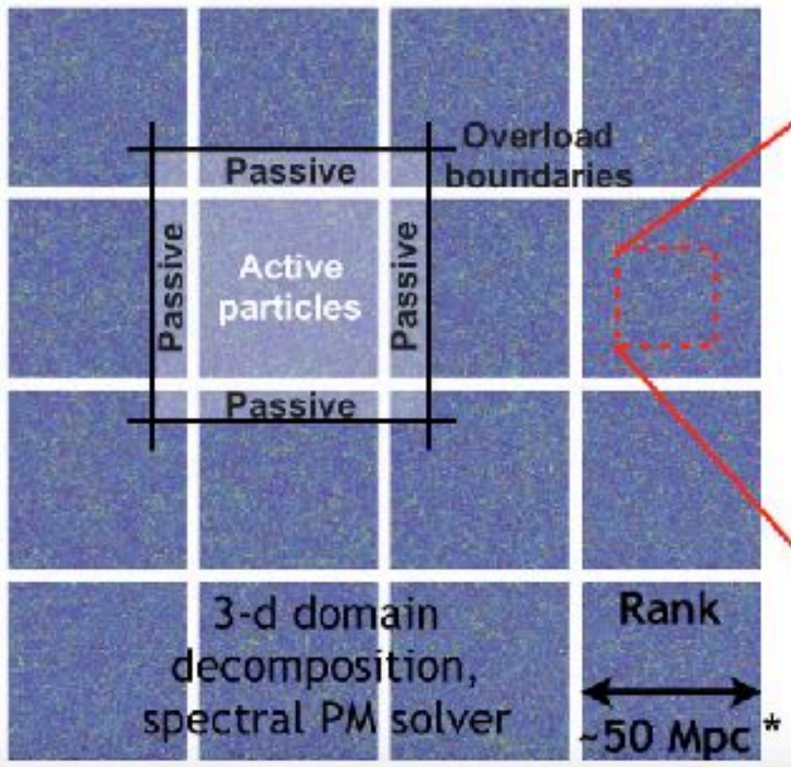


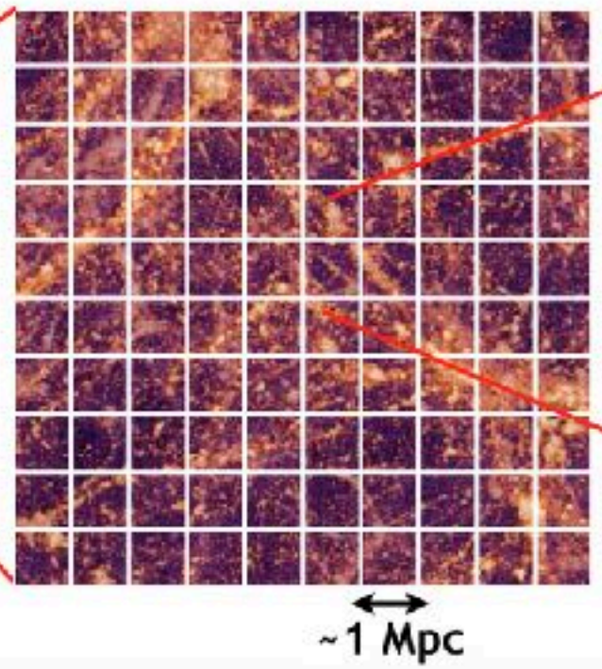
Ec. de Boltzmann.

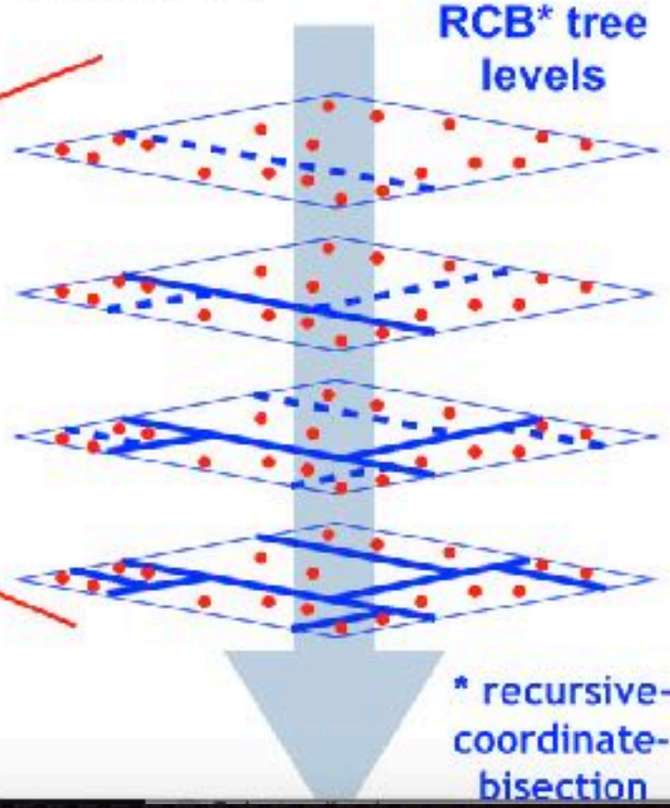
$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} - \nabla \Phi_T \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

HACC in a Nutshell

- **Long-range/short range force splitting:** *S. Habib et al. 2016, New Astronomy*
  - **Long-range:** Particle-Mesh solver, C/C++/MPI, **unchanged for different architectures,** FFT performance dictates scaling (custom pencil decomposed FFT) **Lanczos derivative**
  - **Short-range:** **Depending on node architecture** switch between tree and particle-particle algorithm; tree needs "thinking" (building, walking) but computationally less demanding (BG/Q, X86, KNL), PP easier but computationally more expensive (GPU)
- **Overload concept to allow for easy swap of short-range solver and minimization of communication (reassignment of passive/active in regular intervals)**
- **Adaptive time stepping, analysis on the fly, mixed precision, custom I/O, ...**

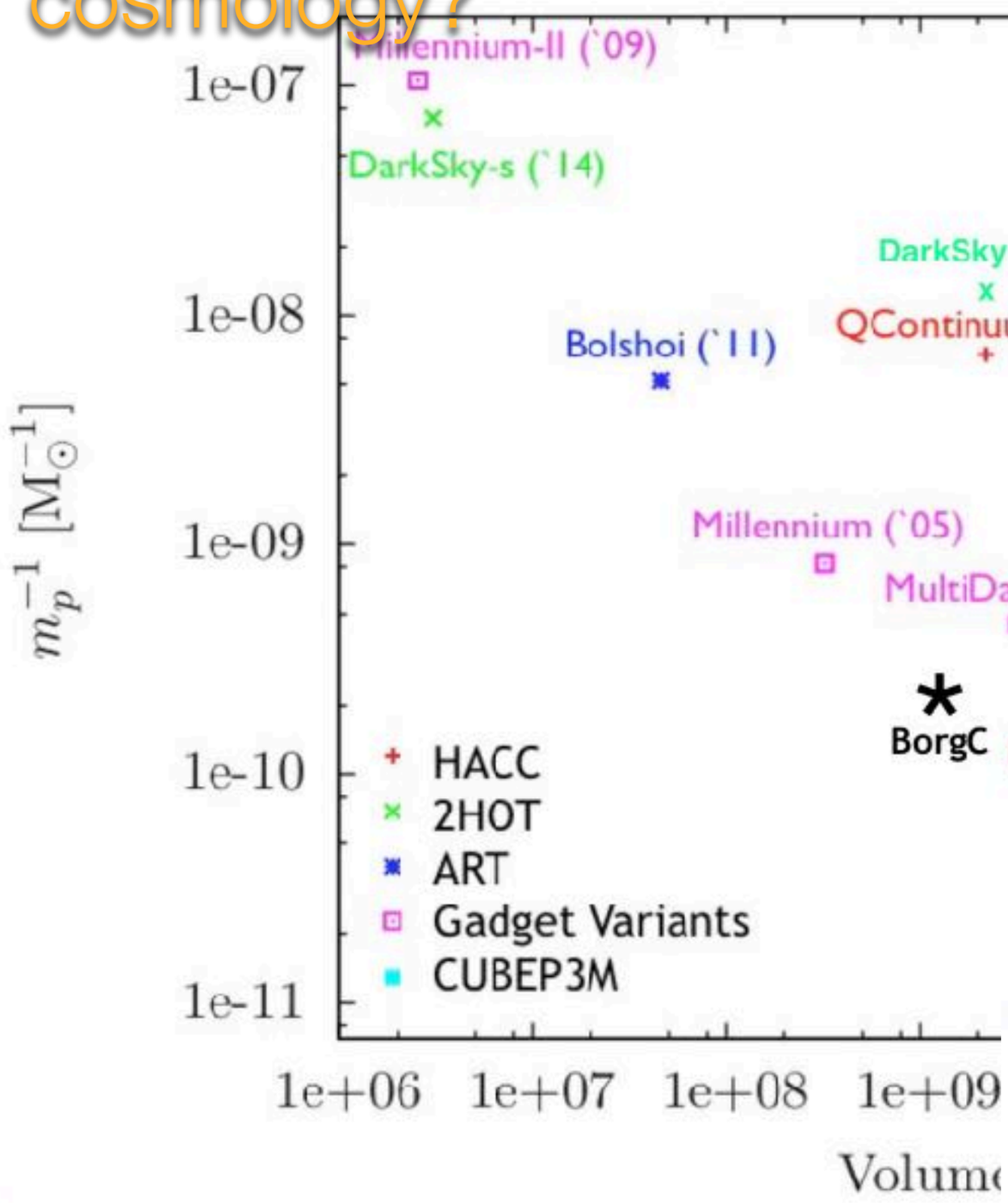


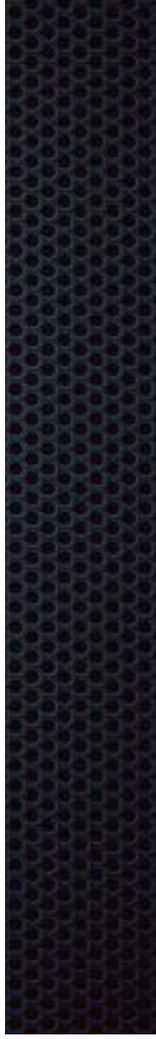




Current State-of-the-Art Gravity-Only Simulations

Alternative cosmology?

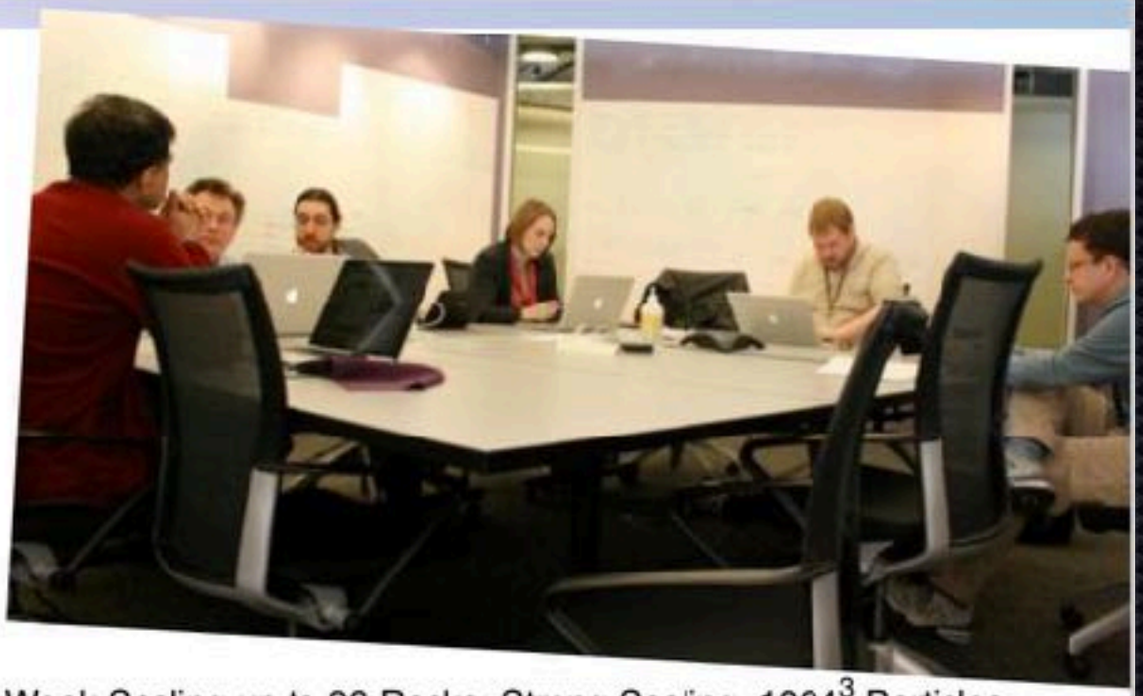




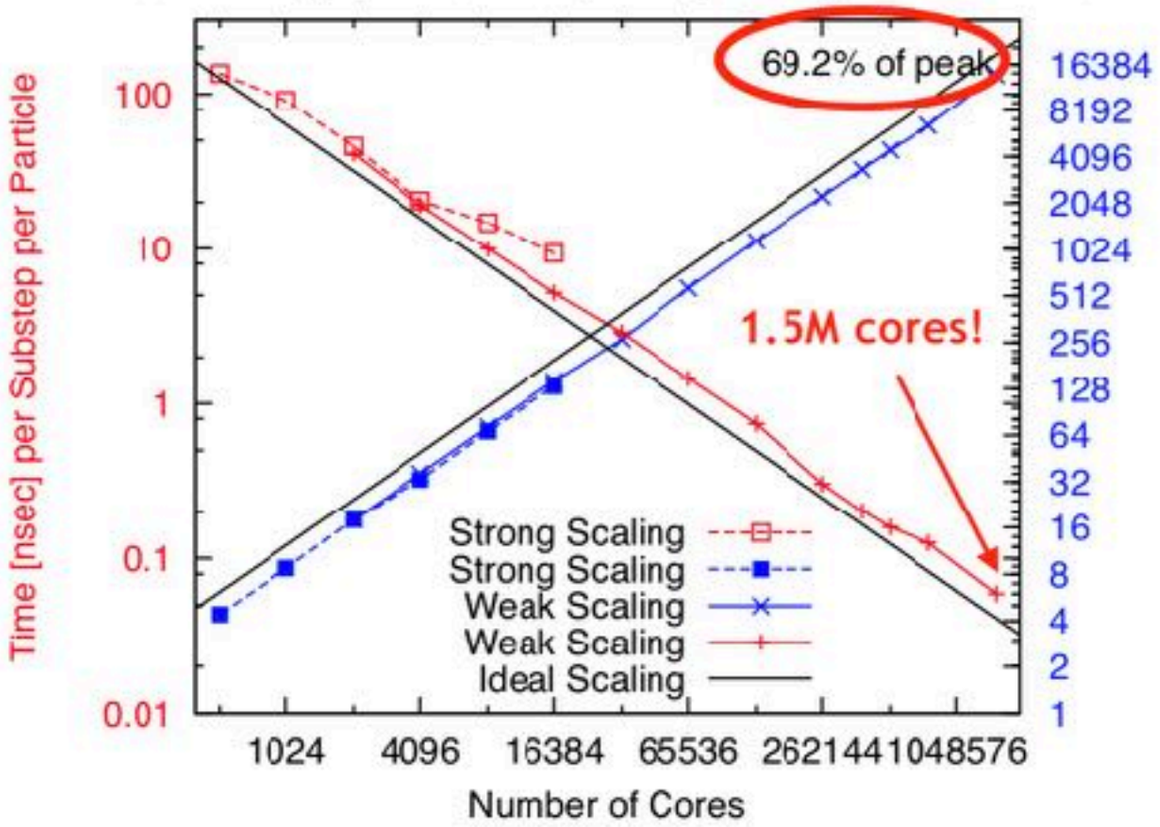
Current State-of-the

Nonlinear Structure Formation Simulations with HACC

- **HACC: Extreme-scale, particle-based framework for computational cosmology**
  - Very high levels of performance across multiple architectures (BG/Q, GPU, KNL,…)
  - Focus on absolute, not relative, performance; first production science code to break 10 PFlops sustained
- **Portability**
  - Uses multiple algorithms and implementations across architectures, yet 95% of code base remains unchanged
    - Programming model: C++/MPI + X (X=OpenMP/CUDA/OpenCL/assembly)
- **Science Targets**
  - Large-scale structure probes of cosmology
  - Exploration of effects of massive neutrinos, dynamical dark energy equation of state, …
  - Focus on large surveys



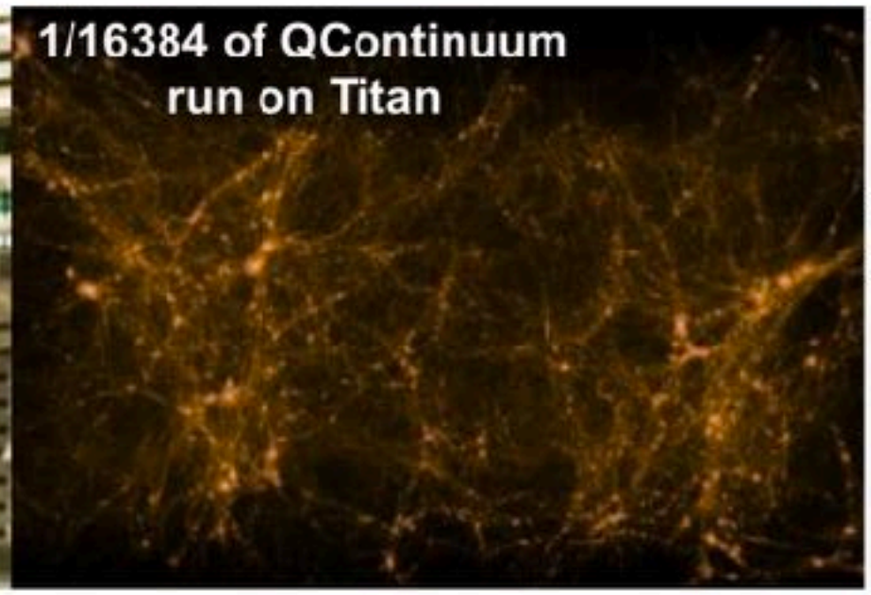
Weak Scaling up to 96 Racks; Strong Scaling, 1024^3 Particles

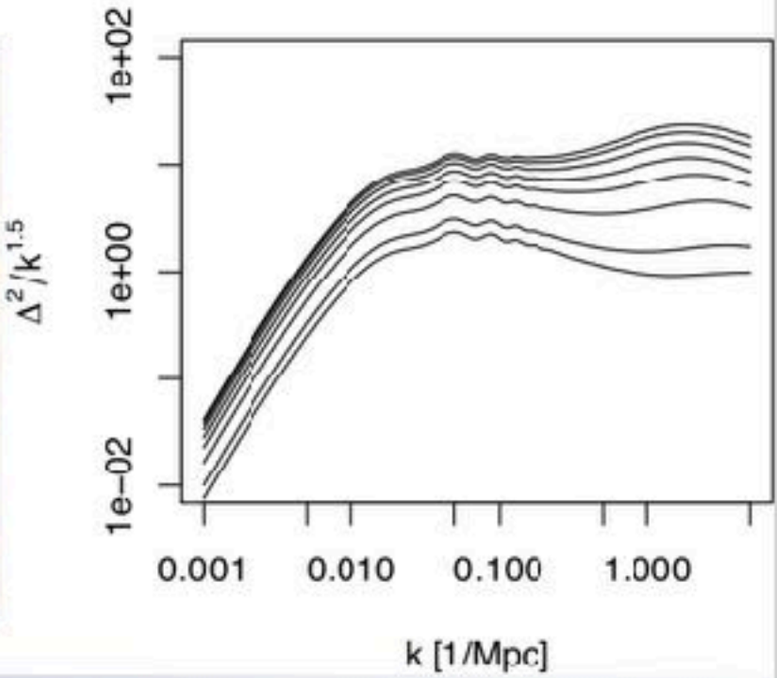


Habib et al. SC12, Habib et al. 2016

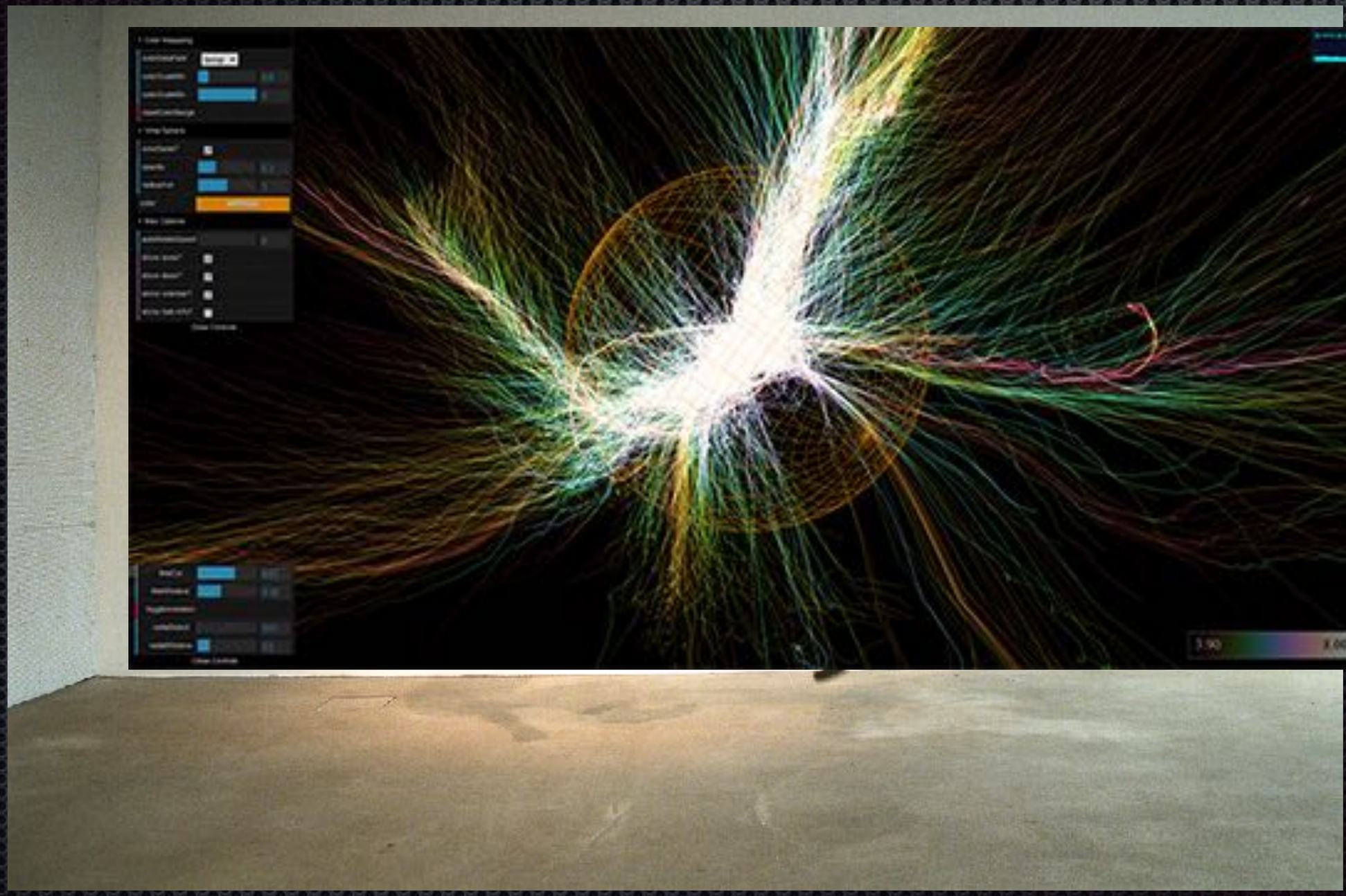
Simulation Challenges Now and in the Future

- **Computational Challenges:** Next-generation computing — opportunities and difficulties
  Exascale hardware/architecture challenge
- **Modeling Challenge:** Gravity + gasdynamics, subgrid models, galaxy/group/cluster modeling
  Scalable gravity + cosmological gasdynamics
- **Statistical Challenges:** Cosmology as a statistical inverse problem, emulators, machine learning
  Robust emulation

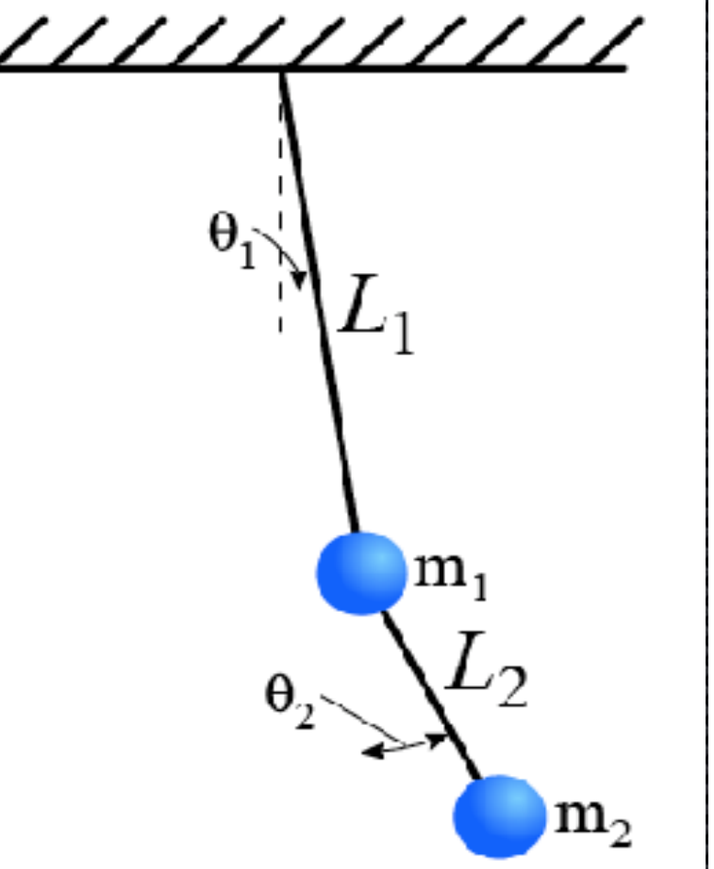




Move. Integration Time Step



Motion/particles trajectories



$$L = \frac{1}{6}m\ell^2\left[\dot{\theta}_2^2 + 4\dot{\theta}_1^2 + 3\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)\right] + \frac{1}{2}mg\ell\left(3\cos\theta_1 + \cos\theta_2\right).$$

There is only one conserved quantity (the energy), and no conserved momenta. The two momenta may be writte

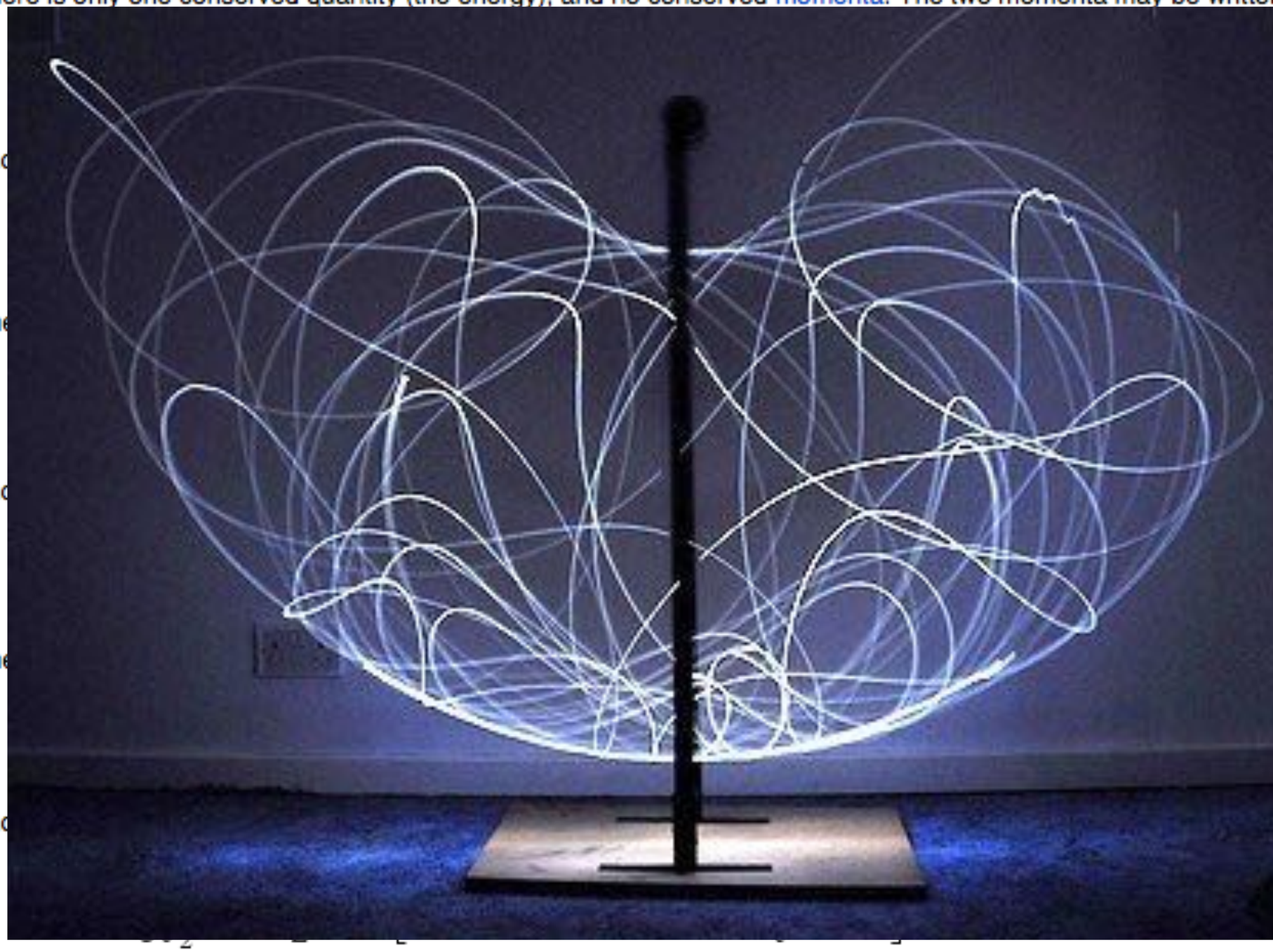
anc

The

anc

The

anc



Motion/ Particles Trejectories

..

- X = F(x)/m = a(x) (Initial Condition problem)

.

- X = V

.

- V = a(x)

How to proceed?

- A classic problem in Ordinary Diferential Equations
- Several Methods

Time integration methods

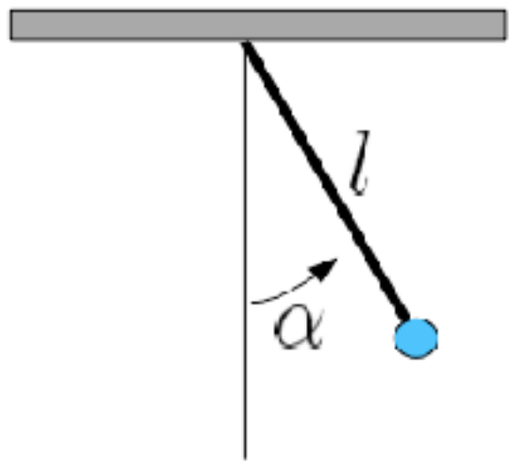
Want to numerically integrate an **ordinary differential equation** (ODE)

$$\dot{\mathbf{y}} = \int(\mathbf{y})$$

Note: $\mathbf{y}$ can be a vector

Example: Simple pendulum

$$\ddot{\alpha} = -\frac{g}{l}\sin\alpha$$



$$y_0 \equiv \alpha \quad y_1 \equiv \dot{\alpha}$$

$$\dot{\mathbf{y}} = f(y) = \begin{pmatrix} y_1 \\ -\frac{g}{l}\sin y_0 \end{pmatrix}$$

A numerical approximation to the ODE is a set of values $\{\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \ldots\}$ at times $\{t_0, t_1, t_2, \ldots\}$

There are many different ways for obtaining this.

Explicit Euler method

$$y_{n+1} = y_n + f(y_n)\Delta t$$

- Simplest of all
- Right hand-side depends only on things already non, **explicit method**
- The error in a single step is $O(\Delta t^2)$, but for the N steps needed for a finite time interval, the total error scales as $O(\Delta t)$!
- Never use this method, it's only **first order accurate**.

Implicit Euler method

$$y_{n+1} = y_n + f(y_{n+1})\Delta t$$

- **Excellent** stability properties
- Suitable for very stiff ODE
- Requires implicit solver for y_{n+1}

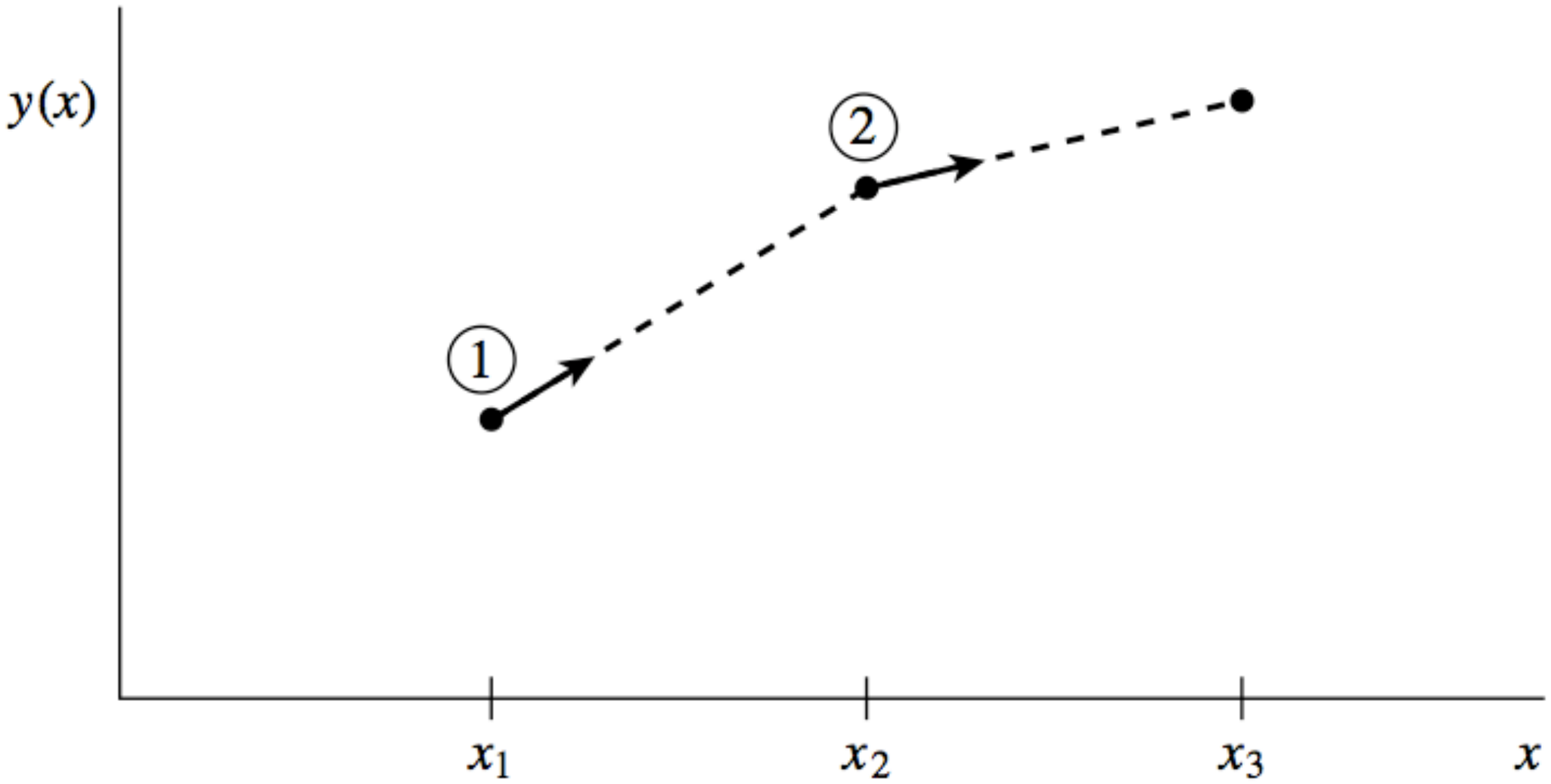


Figure 16.1.1. Euler's method. In this simplest (and least accurate) method for integrating an OD
he derivative at the starting point of each interval is extrapolated to find the next function value. Th
method has first-order accuracy.

Implicit mid-point rule

$$y_{n+1} = y_n + f\left(\frac{y_n + y_{n+1}}{2}\right)\Delta t$$

- **2nd order accurate**
- Time-symmetric, in fact **symplectic**
- But still implicit...

Runge-Kutta methods

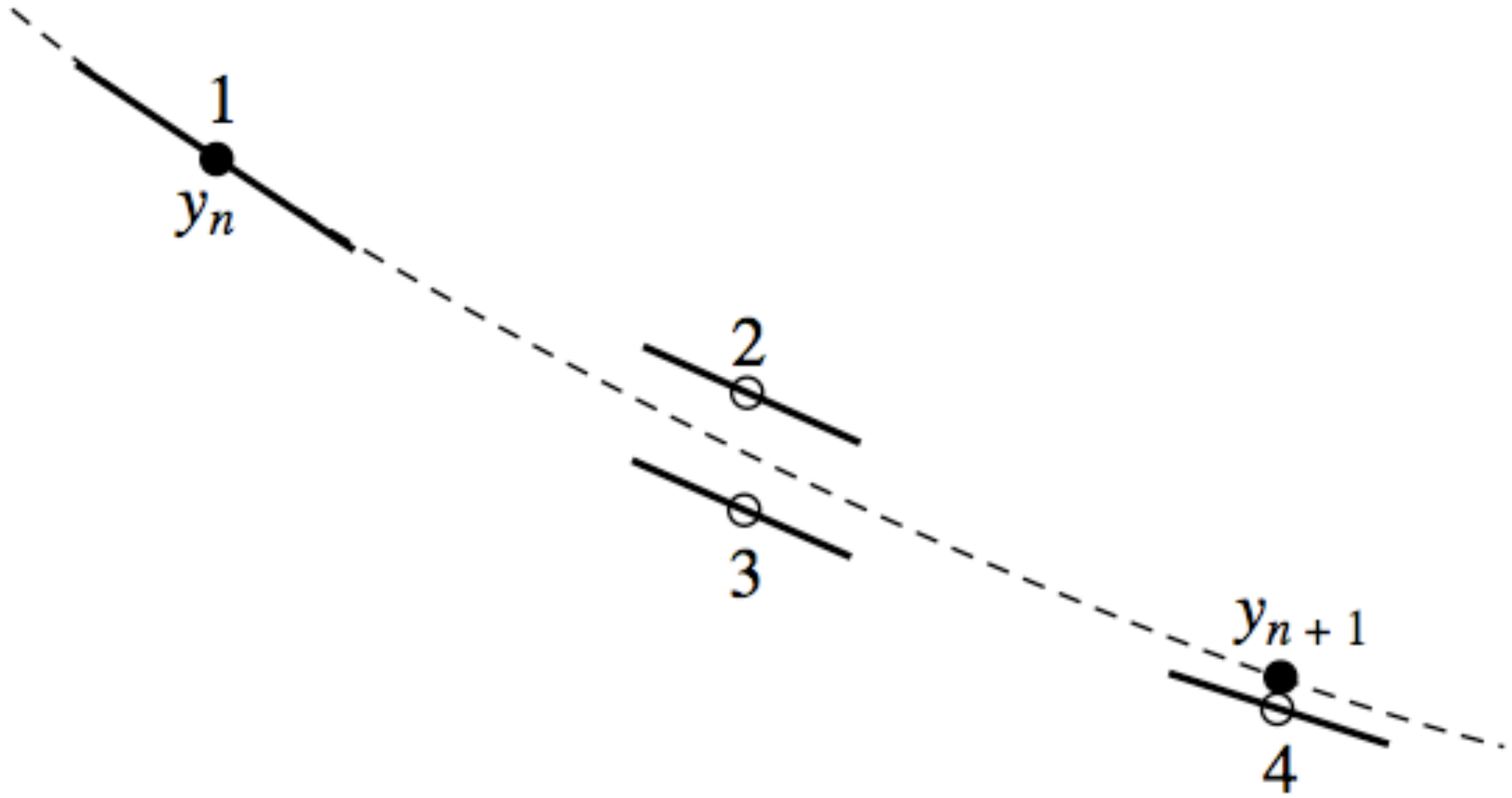
whole class of integration methods

2nd order accurate

$$\begin{aligned} k_1 &= f(y_n) \\ k_2 &= f(y_n + k_1\Delta t) \\ y_{n+1} &= y_n + \left(\frac{k_1 + k_2}{2}\right)\Delta t \end{aligned}$$

4th order accurate.

$$\begin{aligned} k_1 &= f(y_n, t_n) \\ k_2 &= f(y_n + k_1\Delta t/2, t_n + \Delta t/2) \\ k_3 &= f(y_n + k_2\Delta t/2, t_n + \Delta t/2) \\ k_4 &= f(y_n + k_3\Delta t/2, t_n + \Delta t) \\ y_{n+1} &= y_n + \left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\right)\Delta t \end{aligned}$$



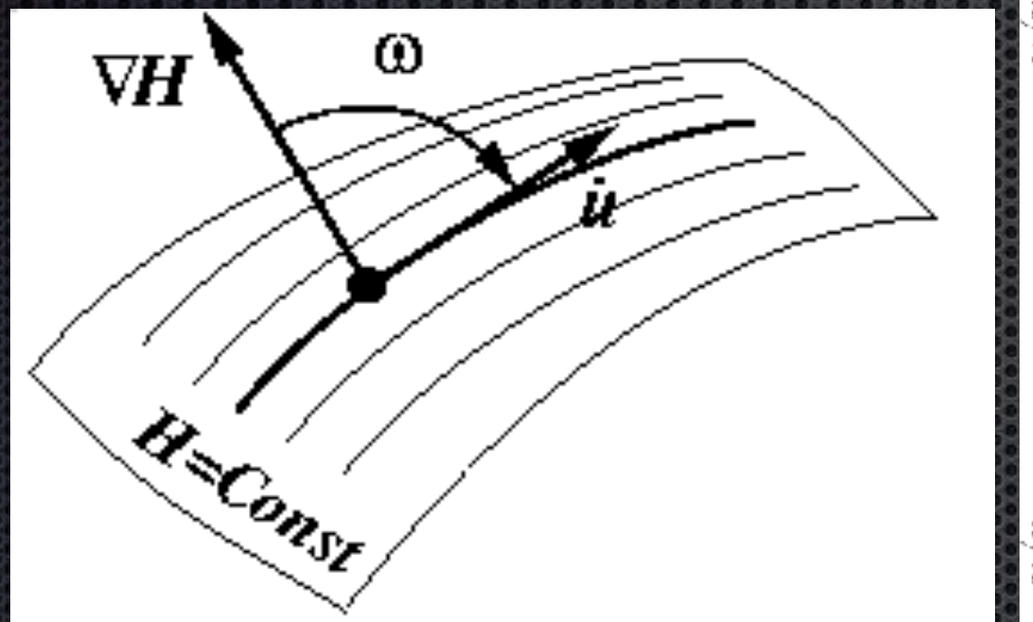
1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluate
e initial point, twice at trial midpoints, and once at a trial endpoint. From these de
ion value (shown as a filled dot) is calculated. (See text for details.)

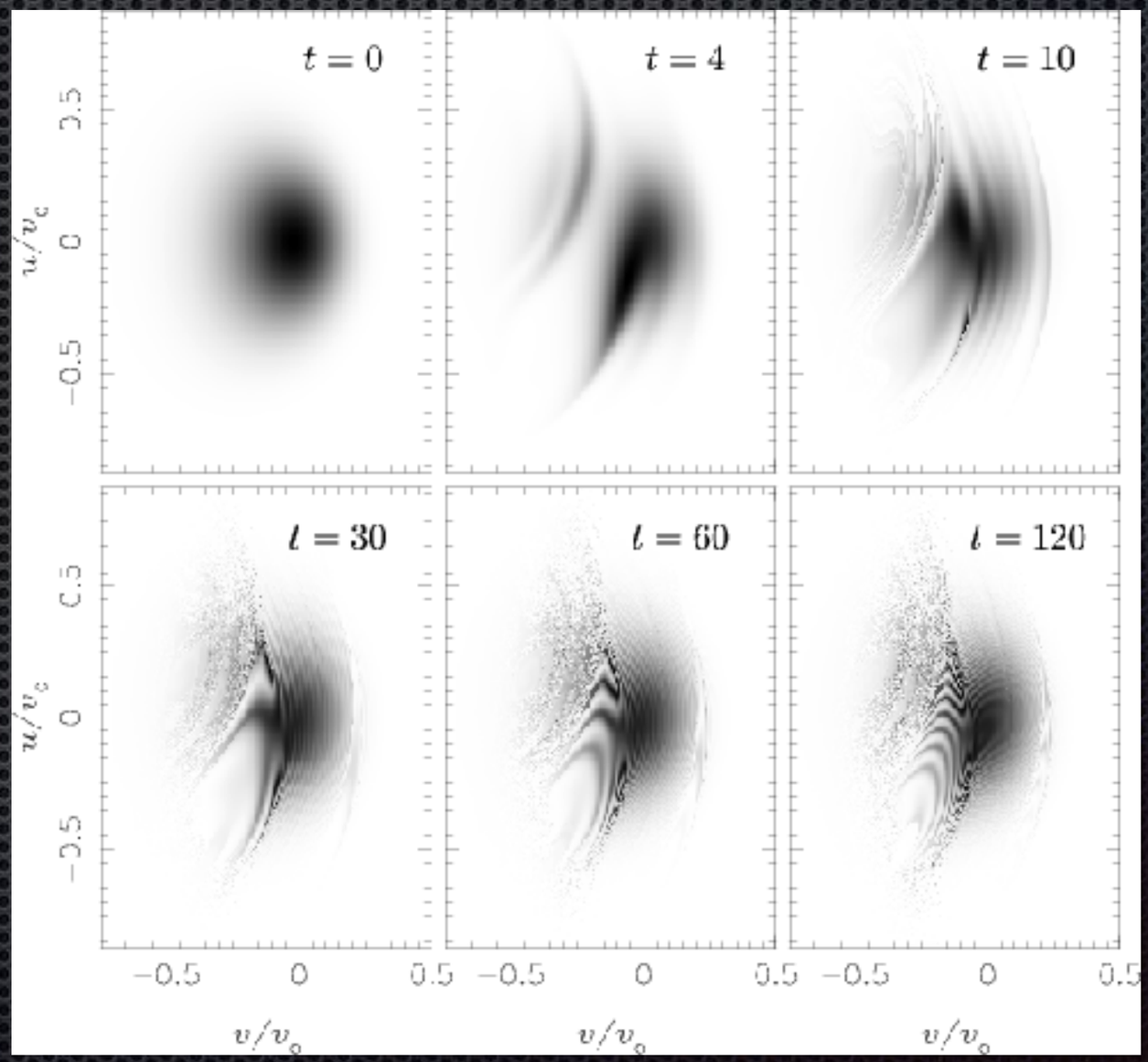
Which one shall we use?

- Euler? Tiny steps? Accuracy? Better avoid the explicit form. What about implicit one? Seems a good one?
- Runge-Kutta (order? 4th, higher?)
- Mid point
- Burlish-Stoher (prediction, correction at the end of interval, see N. Recipes)
- Commonly used and very good integrators

Are we done?

We try to represent this:
Evolution with a constant Hamiltonian (Volume in Phase Space, defined by Conserved Integrals of Motion or by Symmetries)





Most of the N-body codes use an scheme called Leap-Frog

- Why is that?
- Cheap in terms of computation
- Symplectic

The Leapfrog Integrator

For second-order problems like (4) in which f depends only on x, the second-order *Leapfrog* integration scheme is widely used. Its simplicity makes it an attractive alternative. However, it requires us to make a modification to the way we have been thinking about how -- and when -- our data are defined.

Up to now, we have assumed (quite reasonably) that all data are *synchronous* -- that is, all the components of the vector x_i are defined at the same time t_i. However, in second-order systems at least, it is often advantageous to define the velocities (*v = dx/dt*) at the *mid-points* of the intervals -- the velocities are said to be *staggered* with respect to the positions *x*. Setting aside for the moment how this is accomplished in practice, let us define, following our earlier convention,

$$v_{i+1/2} = v(t + \tfrac{1}{2}\delta t), \quad i = 0, 1, 2, \ldots \tag{23}$$

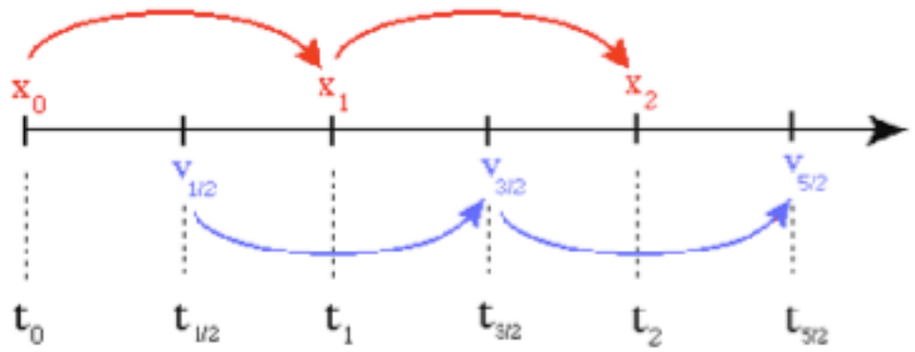
With this definition, we can write down a statement of the Leapfrog scheme that advances x_i to x_{i+1} and $v_{i+1/2}$ to $v_{i+3/2}$:

$$\begin{aligned} x_{i+1} &= x_i + v_{i+1/2}\,\delta t\,, \\ v_{i+3/2} &= v_{i+1/2} + f(x_{i+1})\,\delta t\,. \end{aligned} \tag{24}$$

It is depicted graphically below. Notice the *symmetry* between the ways *x* and *v* are advanced in time. You can easily verify by expanding out the Taylor series

$$v(t + \tfrac{1}{2}\delta t) = v(t) + \tfrac{1}{2}\delta t\, f(t) + O(\delta t^2) \tag{25}$$

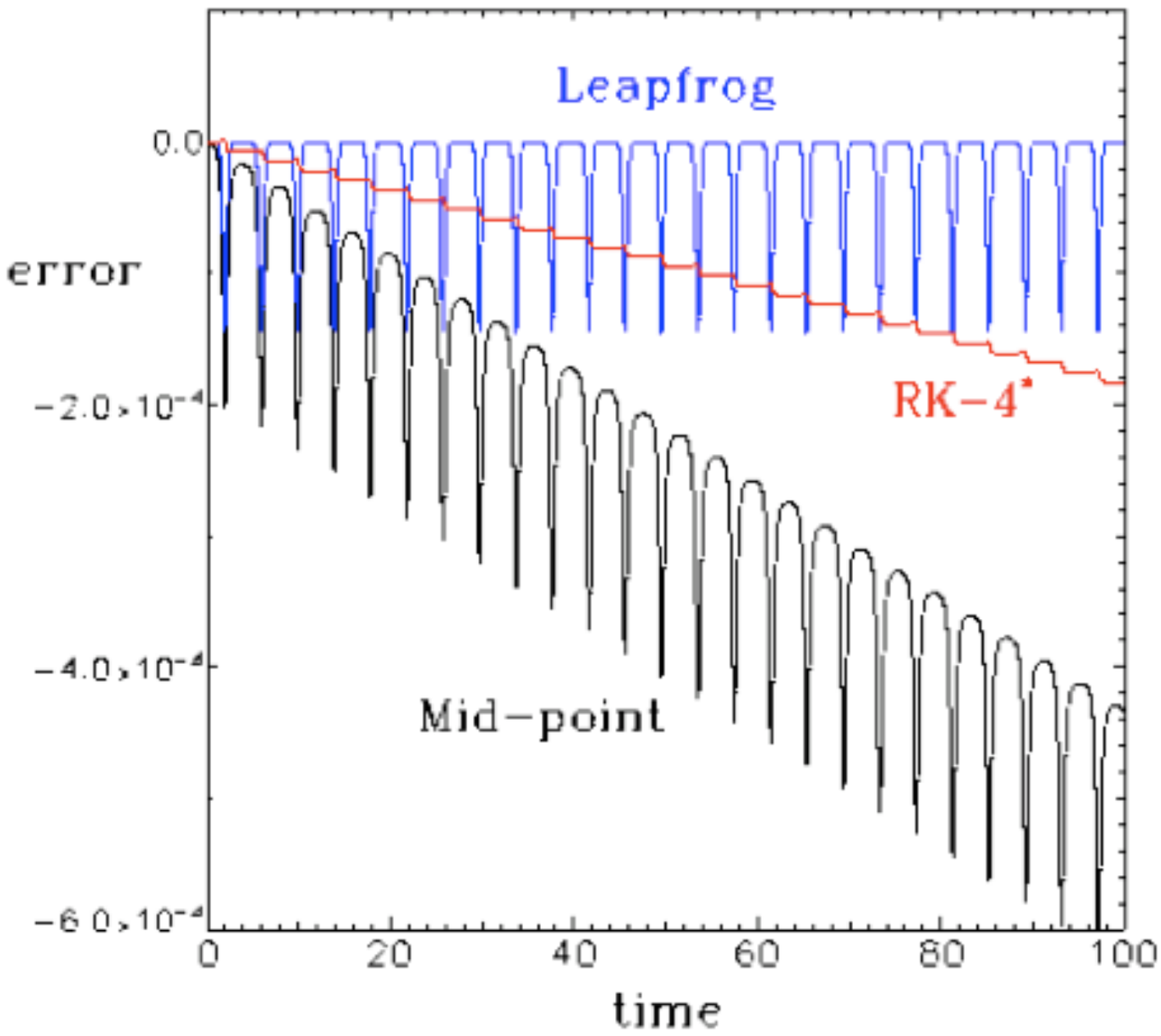
that this scheme does indeed give second-order accuracy in *x*. In fact, it is formally equivalent to the Mid-point or second-order predictor-corrector methods.



Of course, initial conditions are rarely specified at the staggered times required by the leapfrog scheme! Typically, we must use a so-called ``self-starting" scheme (like Euler, Mid-point or Runge-Kutta-4) to take the first half step and establish the value of $v_{1/2}$. The program leapfrog.c

by Runge-Kutta-4 is *systematic*, leading to a long-term drift in the orbital parameters, the energy error in the Leapfrog scheme has no such long-term trend. There *is* a periodic error over the course of an orbit, at the same level as the error in the Mid-point scheme, but the errors incurred over the outgoing portion of the orbit exactly cancel those produced on the incoming segment, so no net error results. The Leapfrog method is only second-order *accurate*, but it is very *stable*.

For collisional systems tracked by short period of time a high order non-symplectic integrator may be a good choice



In situations where we are interested in long-term small changes in the properties of a nearly periodic orbit, and where even small systematic errors would mask the true solution, time-reversible integrators such as the Leapfrog scheme are essential.

Time-Reversibility

You might well ask, ``Since we have to start off with one of the other schemes anyway, and since the Mid-point method is already very simple to program, why should I ever bother with the Leapfrog scheme?" The answer is that, unlike any of the other methods we have described, the Leapfrog integrator is *time reversible* -- and that property gives it some very important advantages.

To see the time reversibility explicitly , reconsider equation (24) and imagine that we wished to ``reverse our tracks" and step backward from $(t_{i+1}, x_{i+1}, v_{i+3/2})$ to $(t_i, x_i, v_{i+1/2})$. Applying the algorithm, we do the following:

$$\begin{aligned} v_{i+1/2} &= v_{i+3/2} + f(x_{i+1})(-\delta t), \\ x_i &= x_{i+1} + v_{i+1/2}(-\delta t), \end{aligned} \qquad (26)$$

But these are precisely the steps (in reverse) that we took to advance the system in the first place! In other words, if we use the Leapfrog scheme to integrate forward in time, then reverse the velocities (and the sign of the timestep) and use the same integrator to return to time $t = 0$, we will arrive *precisely* our starting point -- not approximately, as would be the case with the other integrators, but exactly, at least up to rounding error. Verify this for yourself by modifying `leapfrog.c` to reverse itself and integrate backwards after integrating forward for some interval -- 10 time units, say.

The Leapfrog scheme is time reversible because of the symmetric way in which it is defined. None of the earlier schemes have this property, because they all evaluate derivatives in an asymmetrical way. For example, in the Euler method, it is clear that the forward and backward steps would not cancel out precisely -- they use different derivatives, evaluated at different times. In the Mid-point method, which uses an estimate of the derivative at the center of the range, that estimate is still based on an extrapolation from the *left-hand* side of the interval. On time reversal, the corresponding estimate would be based on the derivative at the right-hand edge, and would *not* yield precisely the same result. The difference is small, but it is enough to prevent the scheme from

Symplectic integrators are designed for the numerical solution of Hamilton's equations, which read

$$\dot{p} = -\frac{\partial H}{\partial q} \quad \text{and} \quad \dot{q} = \frac{\partial H}{\partial p},$$

where q denotes the position coordinates, p the momentum coordinates, and H is the Hamiltonian (see Hamiltonian mechanics for more background).

The time evolution of Hamilton's equations is a symplectomorphism, meaning that it conserves the symplectic two-form $dp \wedge dq$. A numerical scheme is a symplectic integrator if it also conserves this two-form.

Symplectic integrators possess as a conserved quantity a Hamiltonian which is slightly perturbed from the original one. By virtue of these advantages, the SI scheme has been widely applied to the calculations of long-term evolution of chaotic Hamiltonian systems ranging from the Kepler problem to the classical and semi-classical simulations in molecular dynamics.

Most of the usual numerical methods, like the primitive Euler scheme and the classical Runge-Kutta scheme, are not symplectic integrators.

Splitting methods for separable Hamiltonians

A widely used class of symplectic integrators is formed by the splitting methods.

Assume that the Hamiltonian is separable, meaning that it can be written in the form

$$H(p,q) = T(p) + V(q). \qquad (1)$$

This happens frequently in Hamiltonian mechanics, with T being the kinetic energy and V the potential energy.

Then the equations of motion of a Hamiltonian system can be expressed as

$$\dot{z} = \{z, H(z)\} \qquad (2)$$

where $\{\cdot,\cdot\}$ is a Poisson bracket, and $z = (q,p)$. By using the notation $D_H = \{\cdot, H\}$, this can be re-expressed as

$$\dot{z} = D_H z.$$

The formal solution of this set of equations is given as

$$z(\tau) = \exp(\tau D_H) z(0). \qquad (3)$$

When the Hamiltonian has the form of eq. (1), the solution (3) is equivalent to

$$z(\tau) = \exp[\tau(D_T + D_V)] z(0). \qquad (4)$$

The SI scheme approximates the time-evolution operator $\exp[\tau(D_T + D_V)]$ in the formal solution (4) by a product of operators as

$$\exp[\tau(D_T + D_V)] = \prod_{i=1}^{k} \exp(c_i \tau D_T) \exp(d_i \tau D_V) + O(\tau^{n+1}), \tag{5}$$

where c_i and d_i are real numbers, and k is an integer, which is called the order of the integrator. Note that each of the operators $\exp(c_i \tau D_T)$ and $\exp(d_i \tau D_V)$ provides a symplectic map, so their product appearing in the right hand side of (5) also constitutes a symplectic map. In concrete terms, $\exp(c_i \tau D_T)$ gives the mapping

$$\begin{pmatrix} q \\ p \end{pmatrix} \mapsto \begin{pmatrix} q' \\ p' \end{pmatrix} = \begin{pmatrix} q + \tau c_i \frac{\partial T}{\partial p}(p) \\ p \end{pmatrix} \qquad \begin{pmatrix} q \\ p \end{pmatrix} \mapsto \begin{pmatrix} q \\ p - \tau d_i \frac{\partial V}{\partial q}(q) \end{pmatrix}.$$

The symplectic Euler method is the first-order integrator with $k = 1$ and coefficients

$$c_1 = d_1 = 1.$$

The Verlet method is the second-order integrator with $k = 2$ and coefficients

$$c_1 = c_2 = \tfrac{1}{2}, \qquad d_1 = 1, \qquad d_2 = 0.$$

A third order sympectic integrator (with $k = 3$) was discovered by Ronald Ruth in 1983. [1] One of the ma

$$\begin{aligned} c_1 &= \tfrac{2}{3}, & c_2 &= -\tfrac{2}{3}, & c_3 &= 1, \\ d_1 &= \tfrac{7}{24}, & d_2 &= \tfrac{3}{4}, & d_3 &= -\tfrac{1}{24}. \end{aligned}$$

A fourth order integrator (with $k = 4$) was also discovered by Ruth in 1983 and distributed privately to the

This fourth order integrator was published in 1990 by Forest and Ruth and also independently discovered

$$c_1 = c_4 = \frac{1}{2(2 - 2^{1/3})}, \quad c_2 = c_3 = \frac{1 - 2^{1/3}}{2(2 - 2^{1/3})},$$

$$d_1 = d_3 = \frac{1}{2 - 2^{1/3}}, \quad d_2 = -\frac{2^{1/3}}{2 - 2^{1/3}}, \quad d_4 = 0.$$

To determine these coefficients, the Baker–Campbell–Hausdorff formula can be used. Yoshida, in particu

Pendulum example

We can motivate the study of geometric integrators by considering the motion of a pendulum.

Assume that we have a pendulum whose bob has mass $m = 1$ and whose rod is massless of length $\ell = 1$. Take the acceleration due to gravity to be $g = 1$. Denote by $q(t)$ the angular displacement of the rod from the vertical, and by $p(t)$ the pendulum's momentum. The Hamiltonian of the system, the sum of its kinetic and potential energies, is

$$H(q,p) = T(p) + U(q) = \frac{1}{2}p^2 - \cos q\,,$$

which gives Hamilton's equations

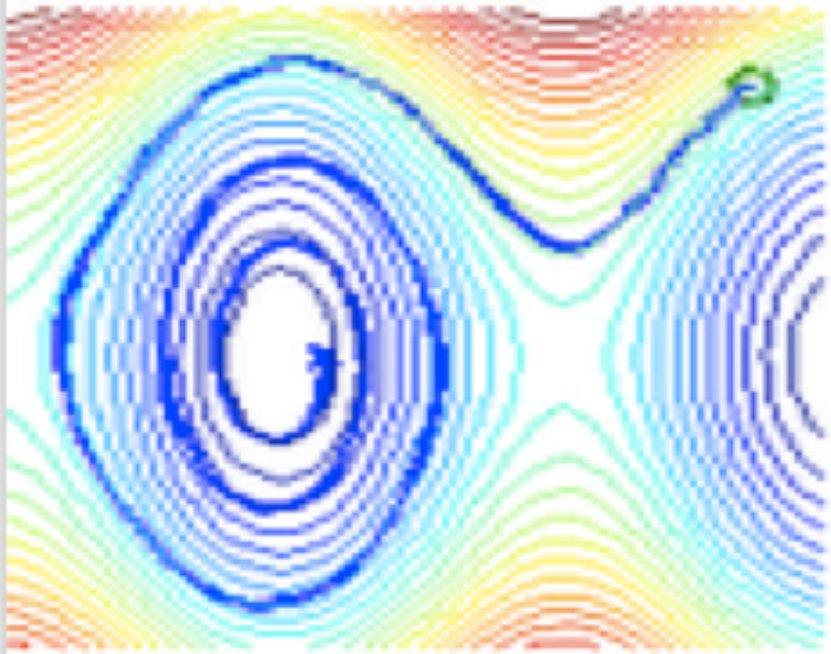
$$(\dot{q},\dot{p}) = (p, -\sin q)\,.$$

It is natural to take the configuration space Q of all q to be the unit circle $\mathbb{S}^1$, so that (q,p) lies on the cylinder $\mathbb{S}^1 \times \mathbb{R}$. However, we will take $(q,p) \in \mathbb{R}^2$, simply because (q,p)-space is then easier to plot. Define $z(t) = (q(t),p(t))^{\mathrm{T}}$ and $f(z) = (p, -\sin q)^{\mathrm{T}}$. Let us experiment by using some simple numerical methods to integrate this system. As usual, we select a constant step-size h and write $z_k := z(kh)$ for $k \geq 0$. We use the following methods.

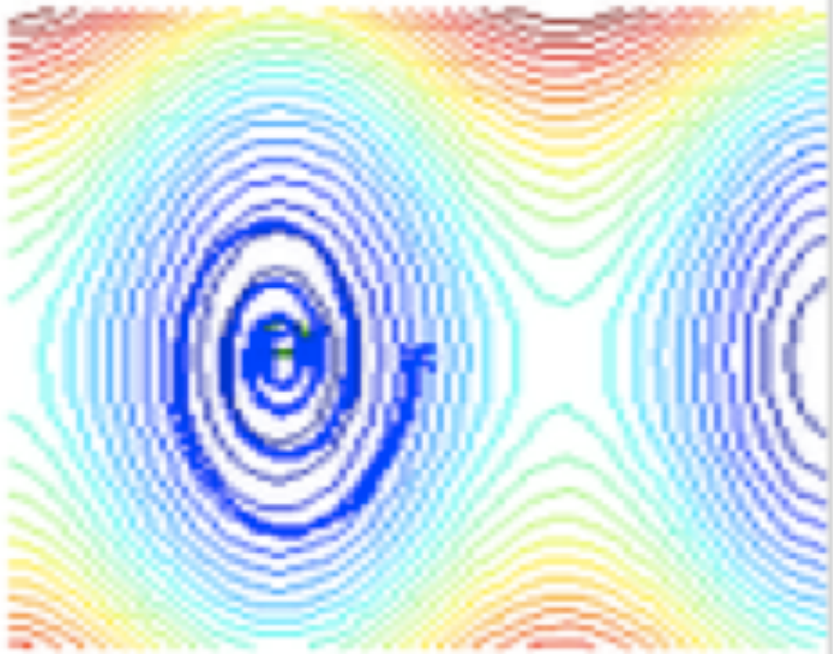
$z_{k+1} = z_k + hf(z_k)$ (explicit Euler),
$z_{k+1} = z_k + hf(z_{k+1})$ (implicit Euler),
$z_{k+1} = z_k + hf(q_k,p_{k+1})$ (symplectic Euler),
$z_{k+1} = z_k + hf((z_{k+1} + z_k)/2)$ (implicit midpoint rule).

(Note that the symplectic Euler method treats q by the explicit and p by the implicit Euler method.)

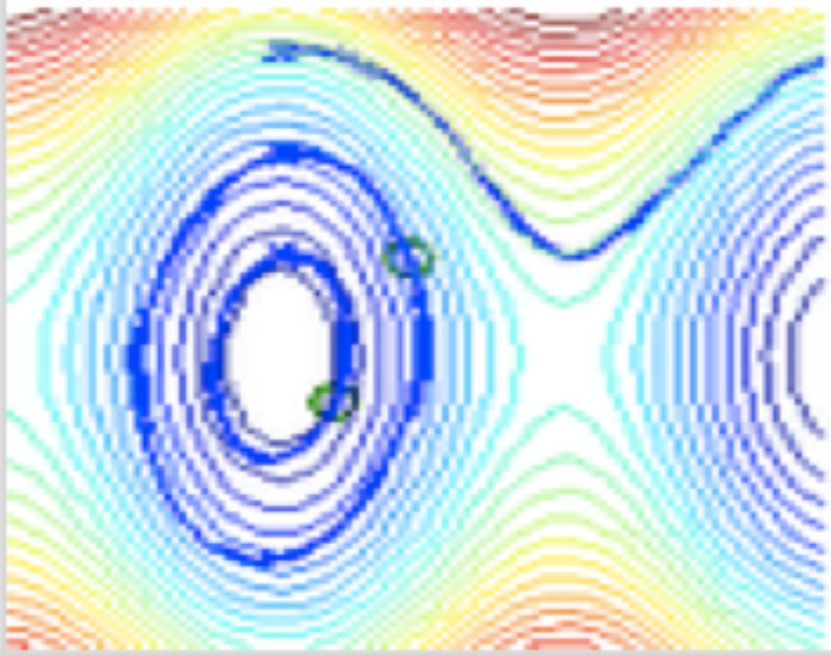
Explicit Euler



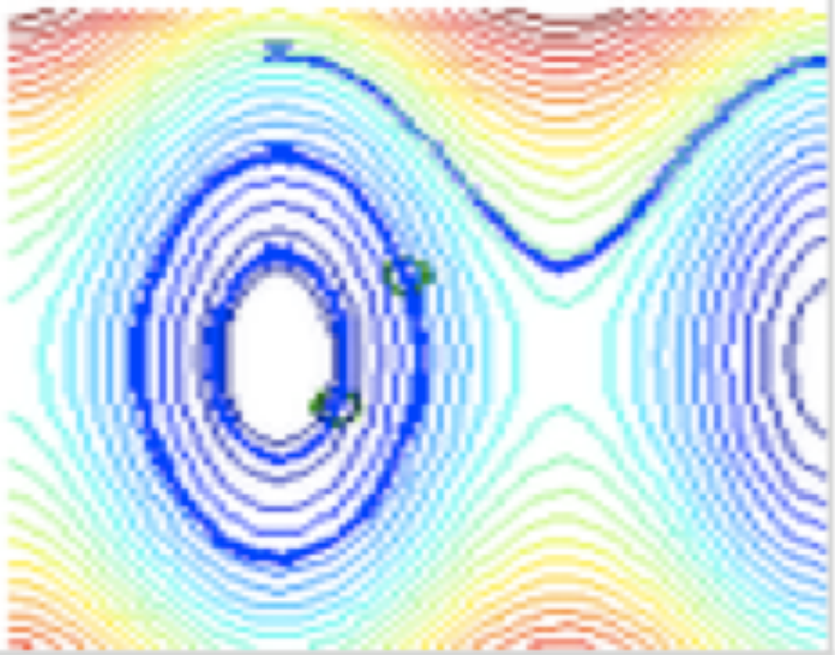
Implicit Euler



Symplectic Euler



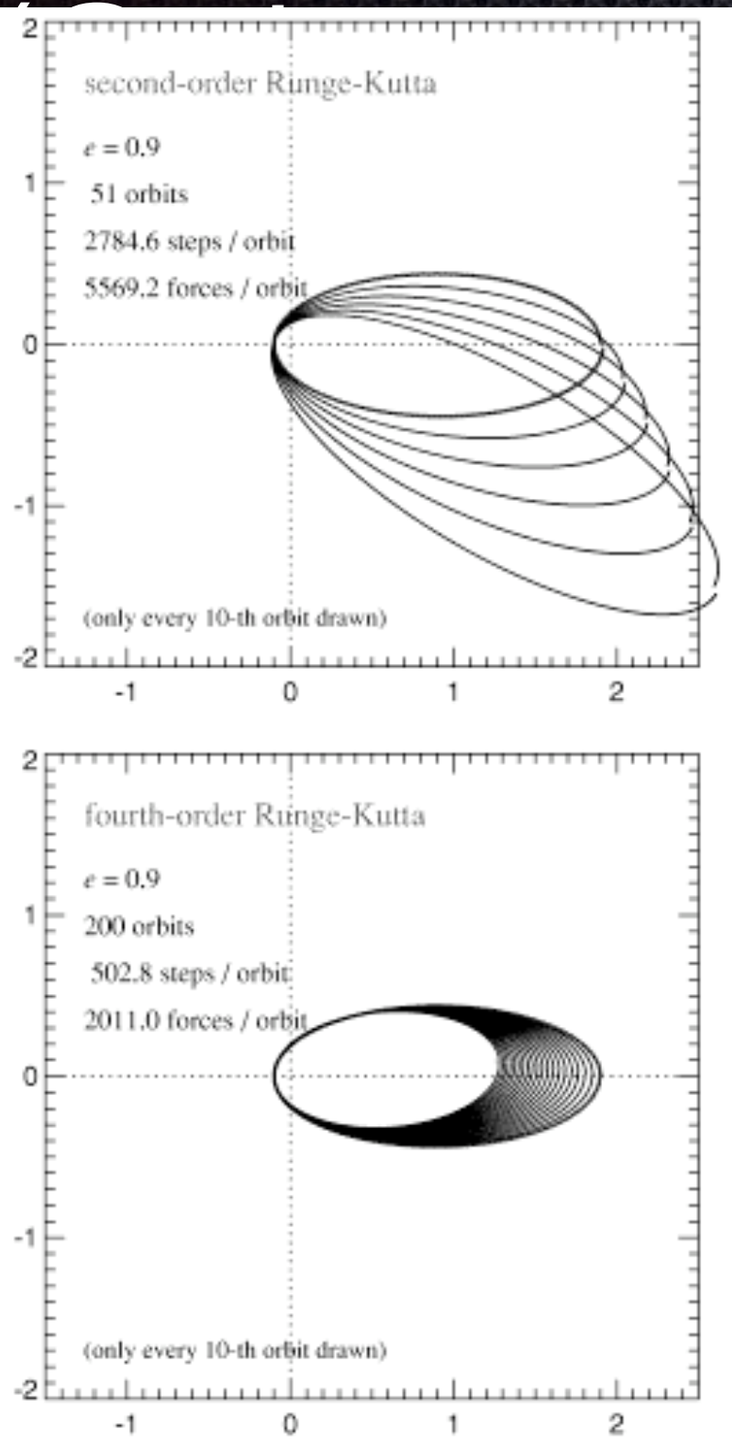
Implicit Midpoint



Simple pendulum: trajectories

Approximate solutions

Klypin, Valenzuela, Colin, Quinn 08



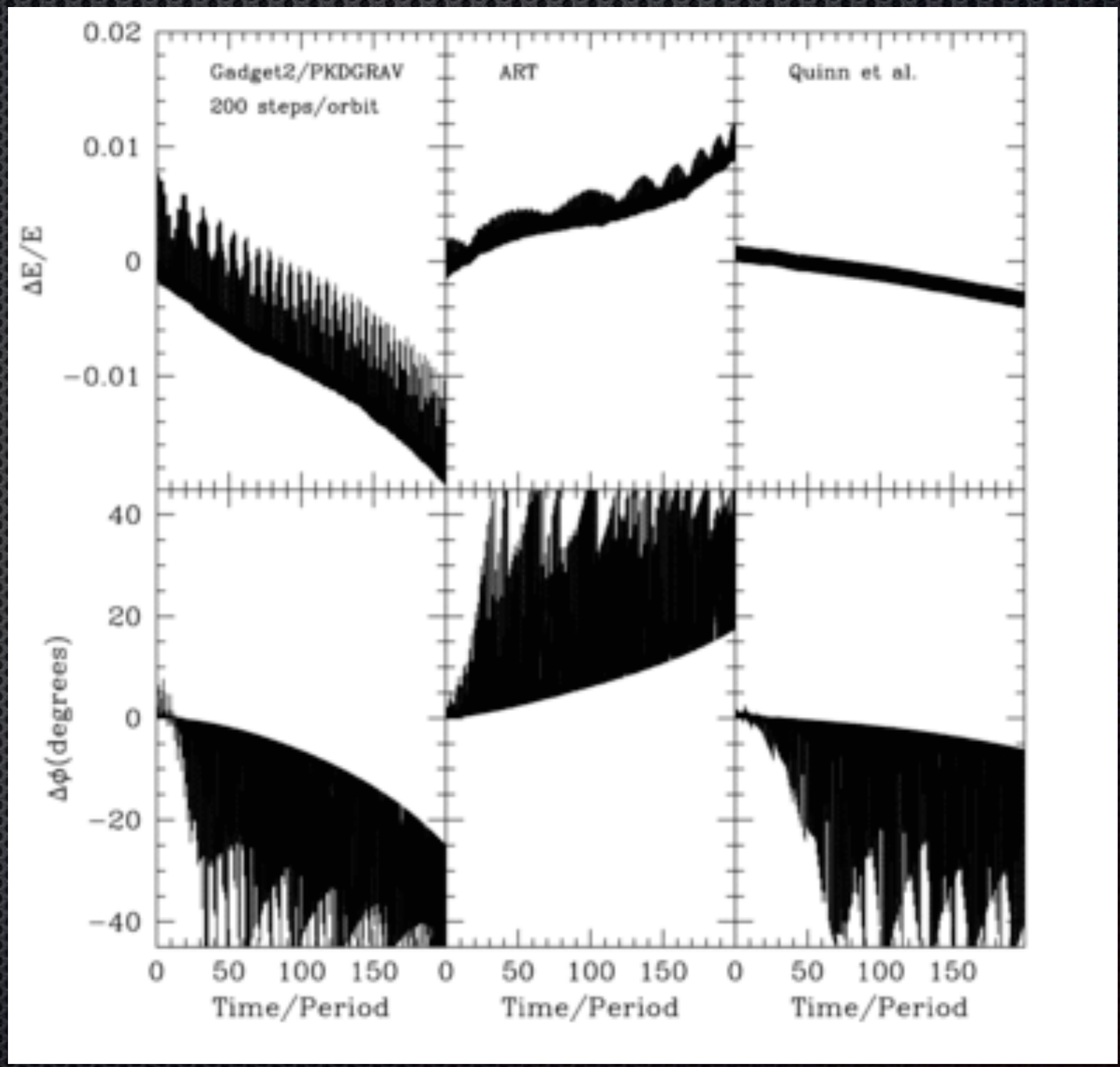


Figure 4. A Kepler problem of high eccentricity evolved with different simple time integration schemes, using an equal time-step in all cases. Even

A combination of a pseudo simplectic scheme and a time step prescription seems to have a slow departure from the Hamiltionian solution

- Can we do it better?

Hut, Makino, McMillan 1995

- Implicit: An iterative solution at each step (expensive, in the 90´s, now?)
- Worth trying
- ANY ONE?

Why shall we care?
Allows larger time steps without compromising accuracy?

Anyone?

Lessons

- Do not feel overwhelm: Code development and use is now a transdisciplinary group project.
- You are capable to learn the skills for your interest
- Advice: choose carefully the tool for the problem
- Test. Test and test afterwards
- You pretend to have a numerical dynamical experiment including some properties of the observed galaxy/universe not the actual universe inside the computer
- You are trying to test hypothesis not creating the universe ()but looks like
- Sometimes impossible to catch a numerical problem after the simulation has been run

- More eficiente Integrators
- More flexible and accurate Solvers
- More efficient parallelization
- New strategy Boltzmann solver??