N-BODY SIMULATIONS II
Codes, Integrators

Mexican School of
Computer Simulations
Mexico City

Octavio Valenzuela (IA-UNAM)

Lecture I Overview

- Why N-body? Decision? Examples, Examples…
- General Structures of Nbody Codes: IC´s, Densisty-Force Calculation, Time Integration, done!???
- Colisionality
- Testing Convergence
- More Examples
- More Examples
- Final Advice

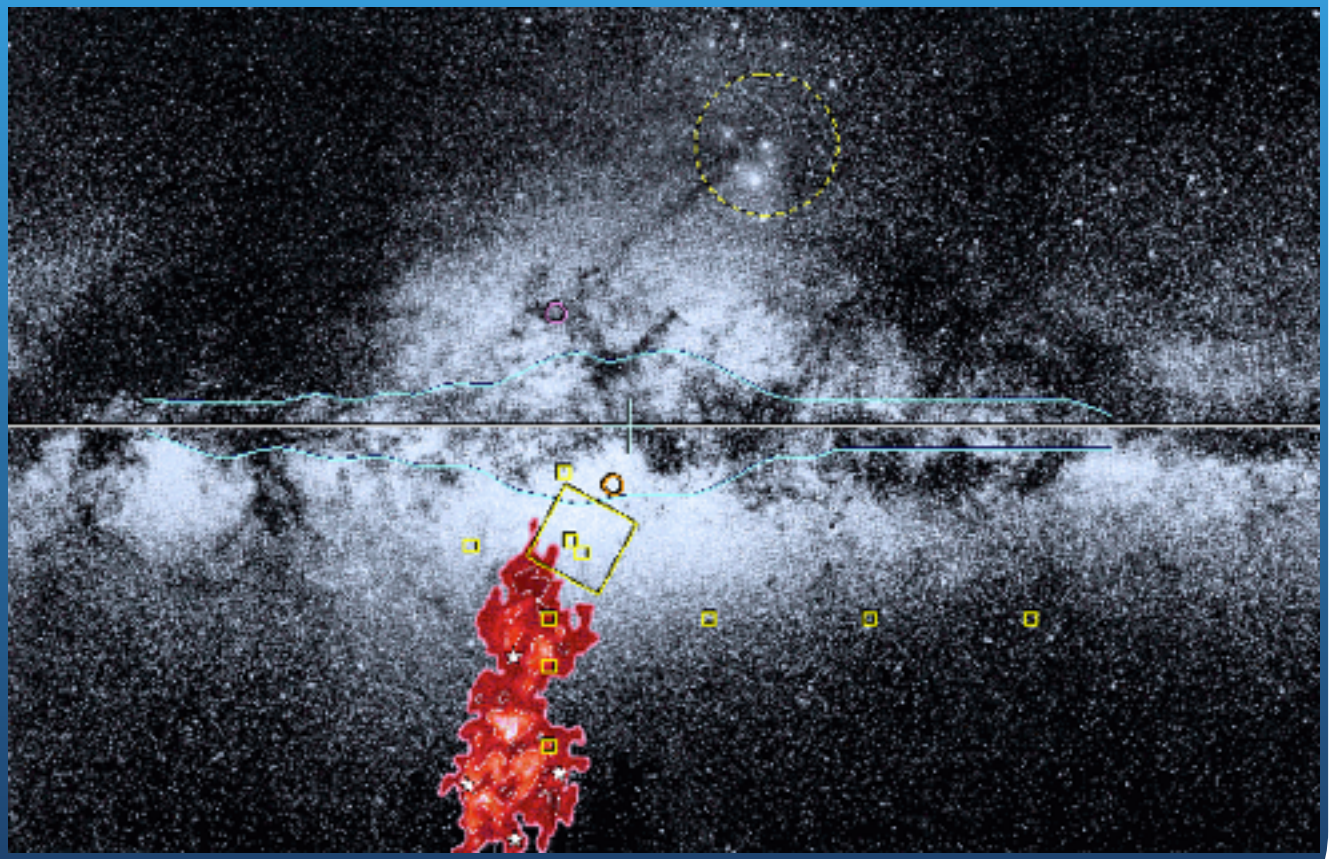
Reasons to use N-body Simulations

- 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)

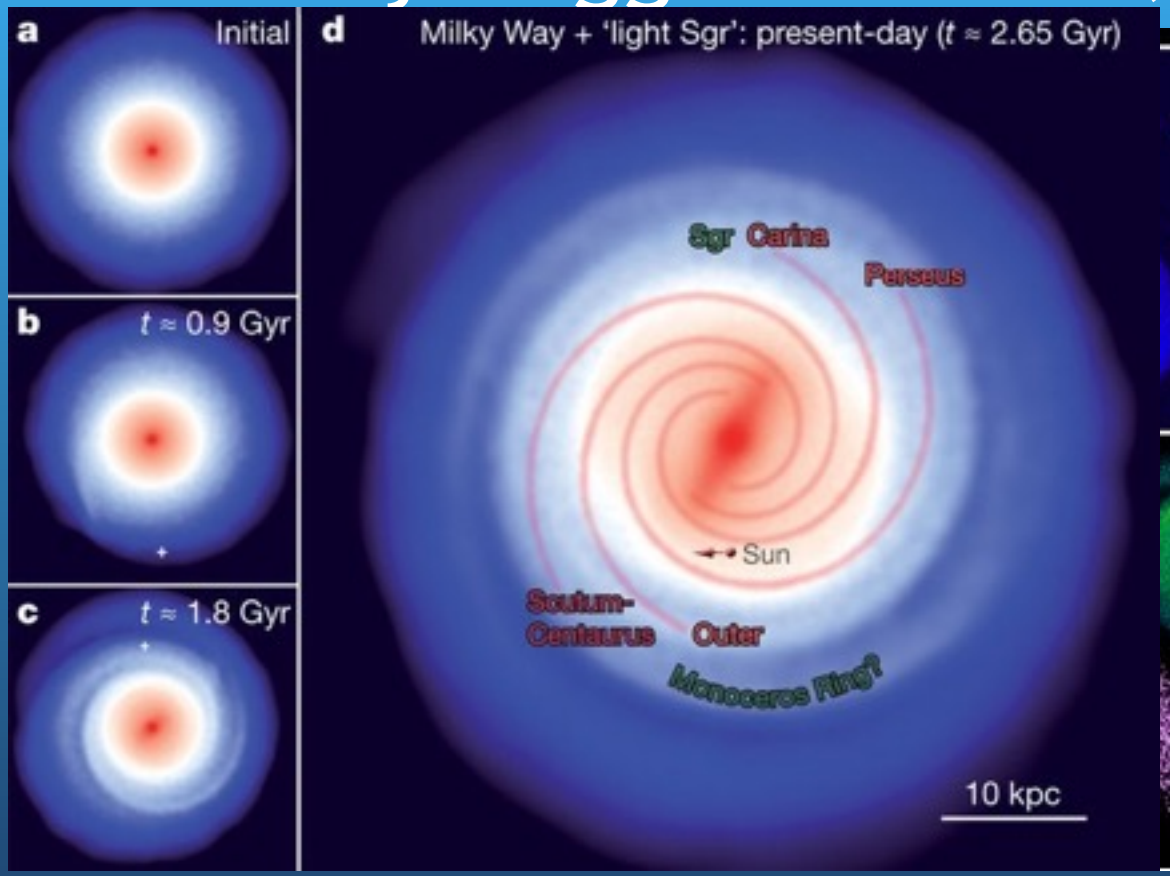
The Sagittarius satellite being disrupted by an encounter with the Milky Way

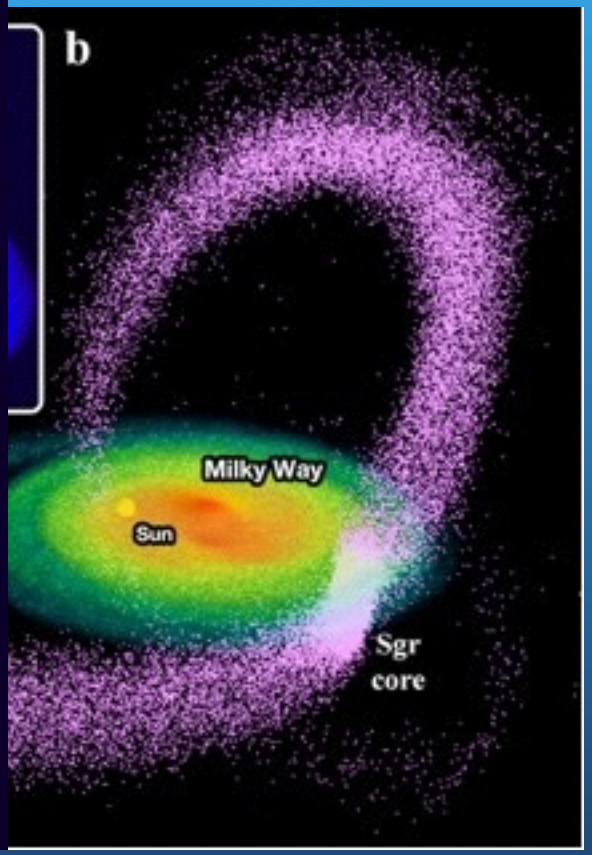
Ibata 1994



Tidal Stripping and Stirring

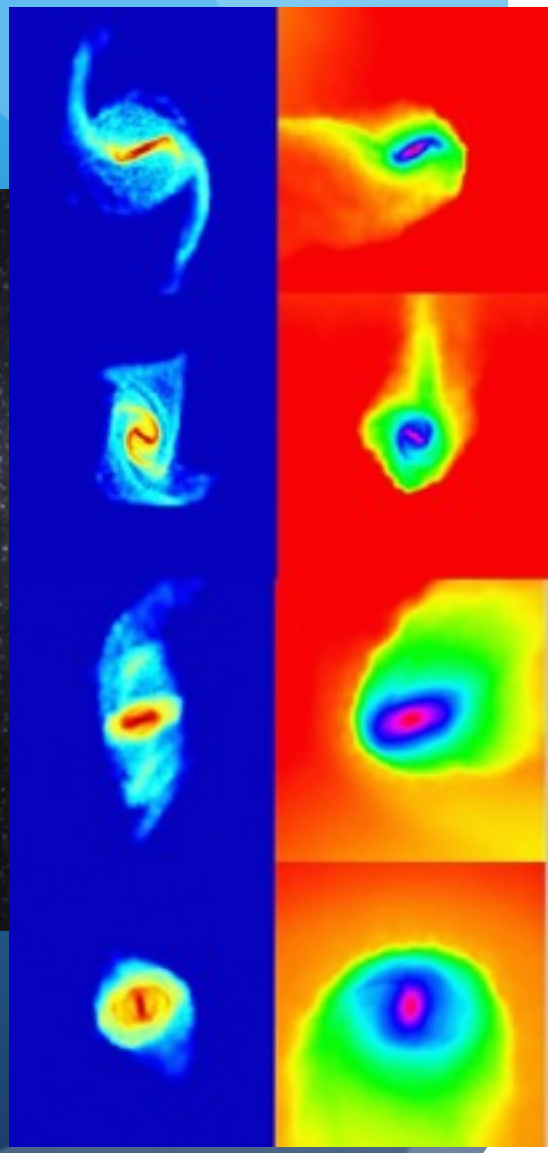
Spiral Arms and disk rings secularly or tidally triggered? Purcell, Bullock 2011





Law, Majewsky

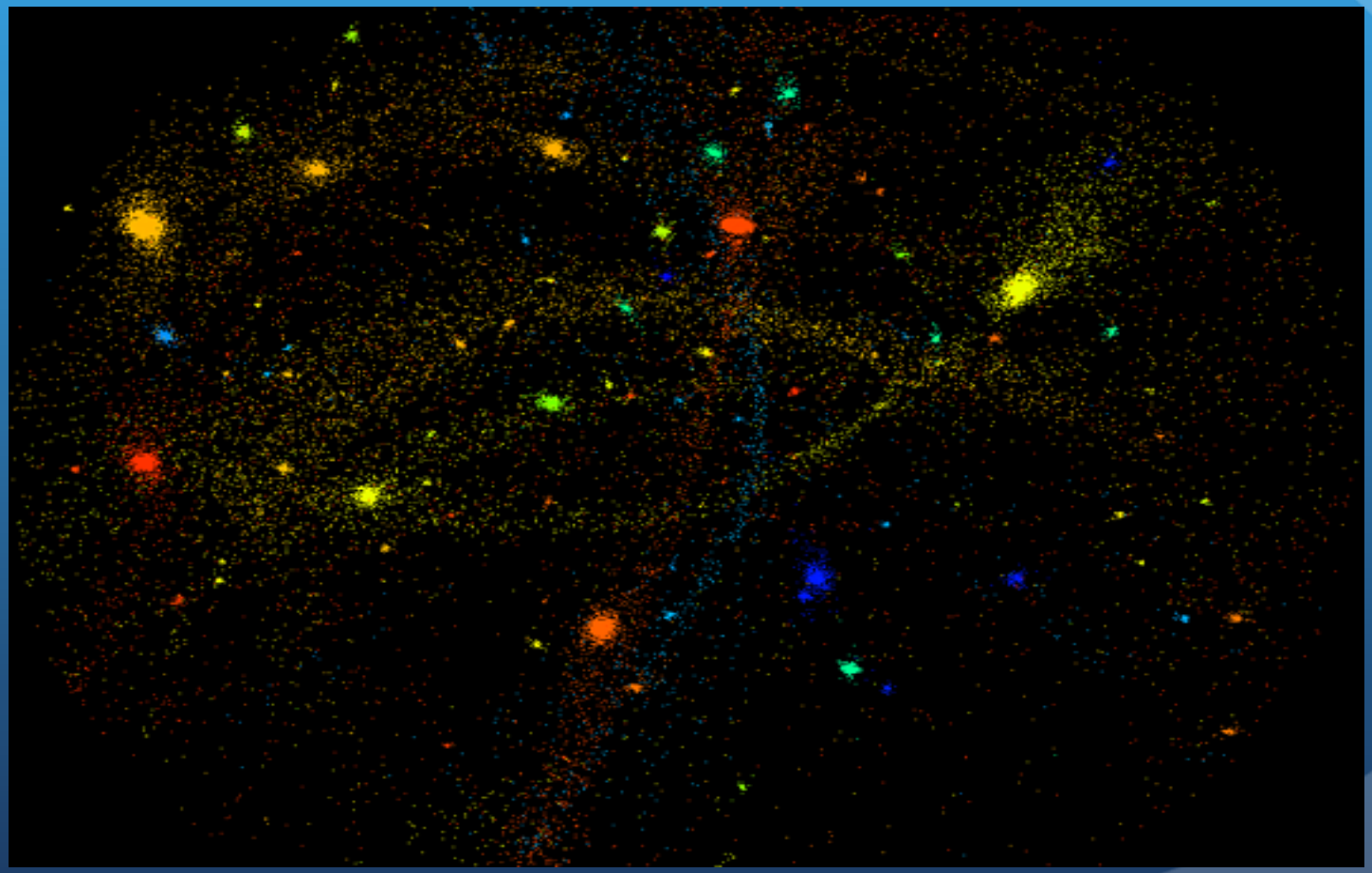
What about M31? Dynamical state of its satellites? MW ones?



Tidal stirring

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

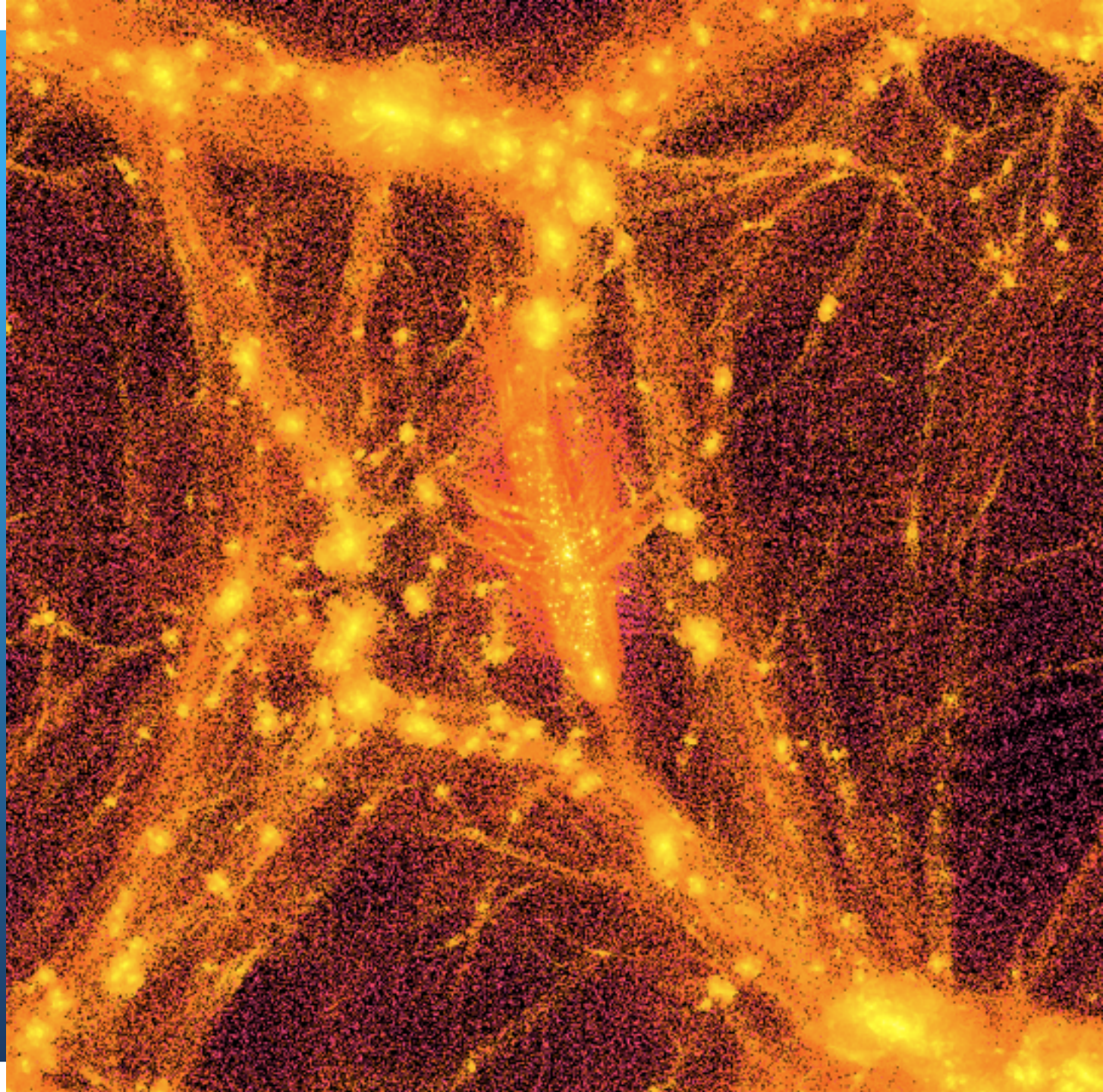
Halo Archeology

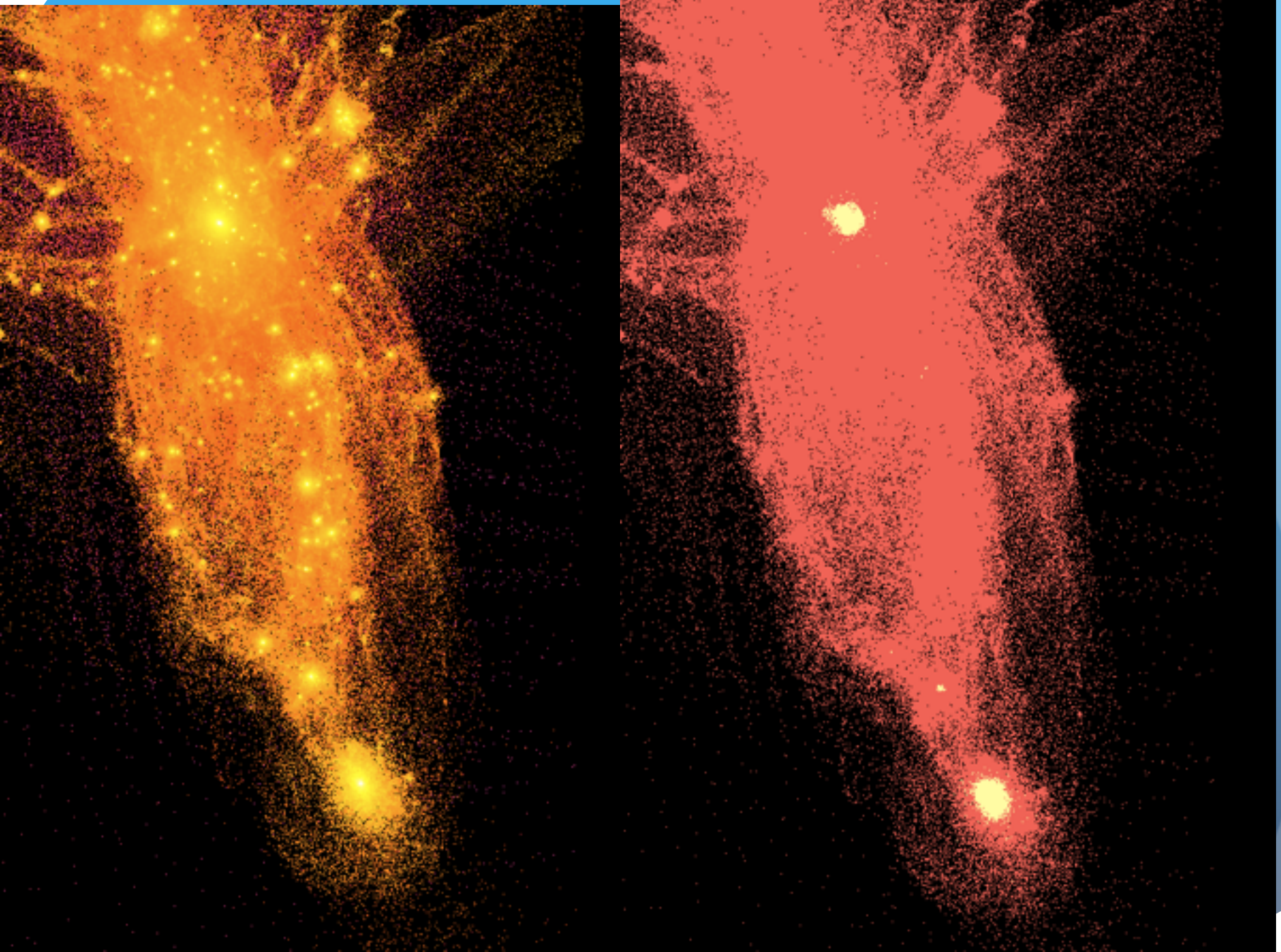


Galaxy - Galaxy interactions (HST)

High redshift Galaxies

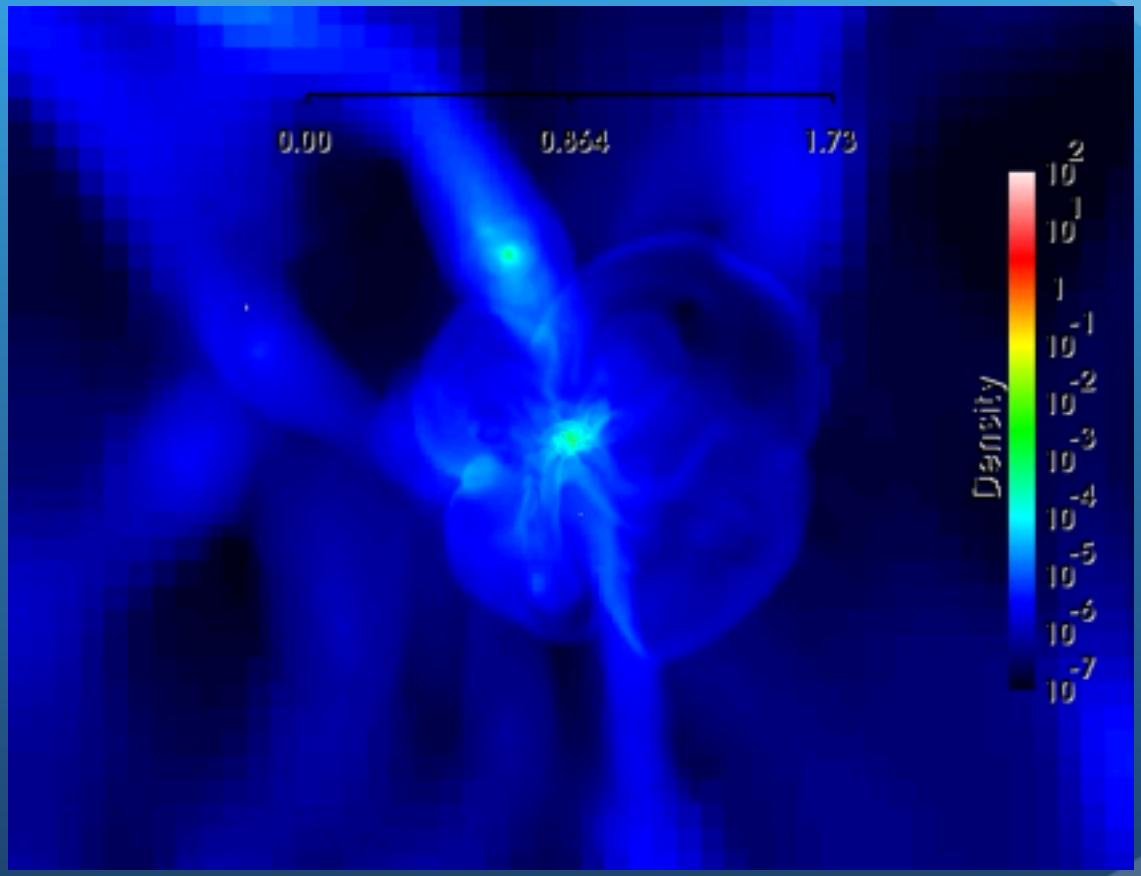
Dark Matter
Galaxy
Environment





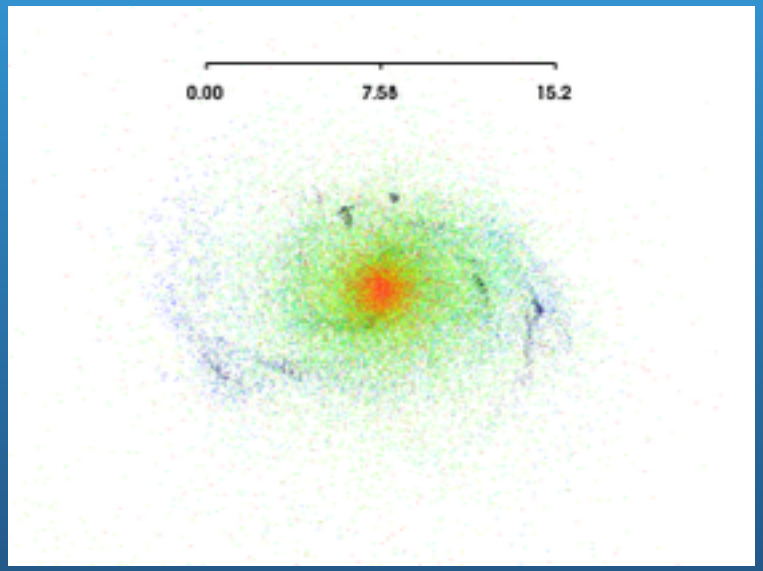
Galaxy at z=1

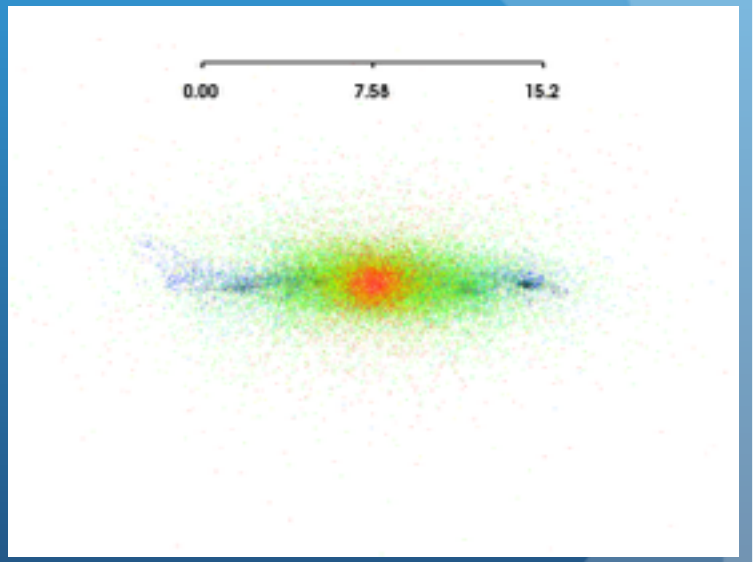
Gonzalez-Samaniego, Valenzuela, Colin

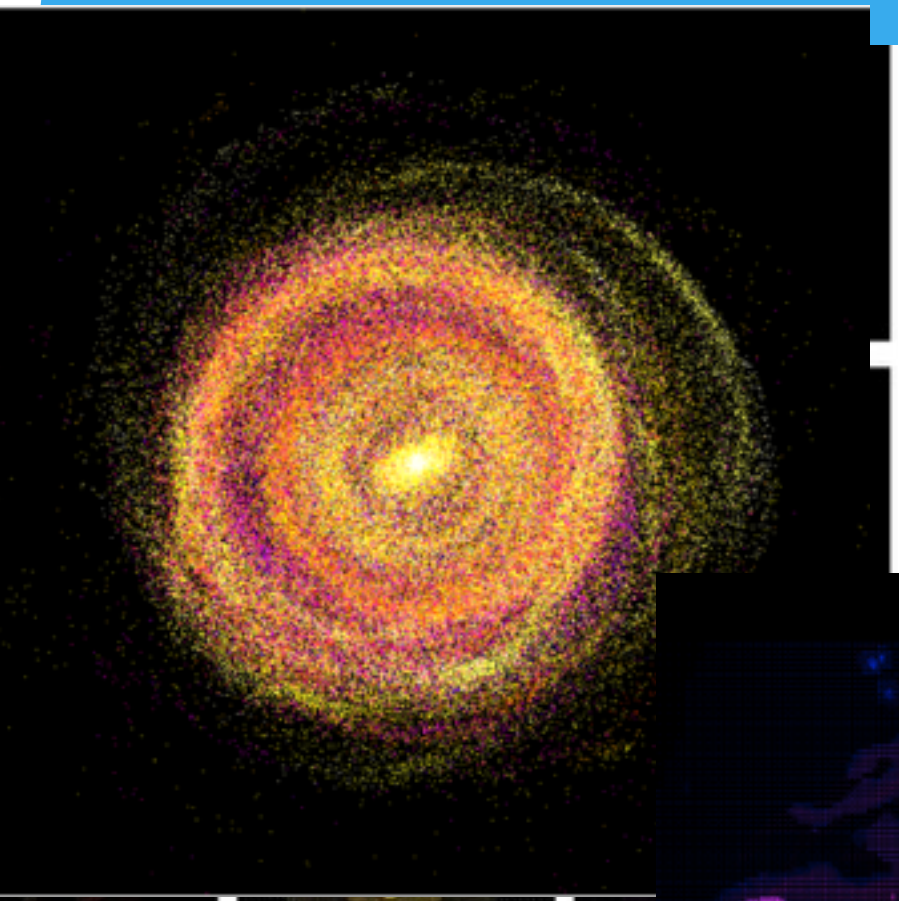


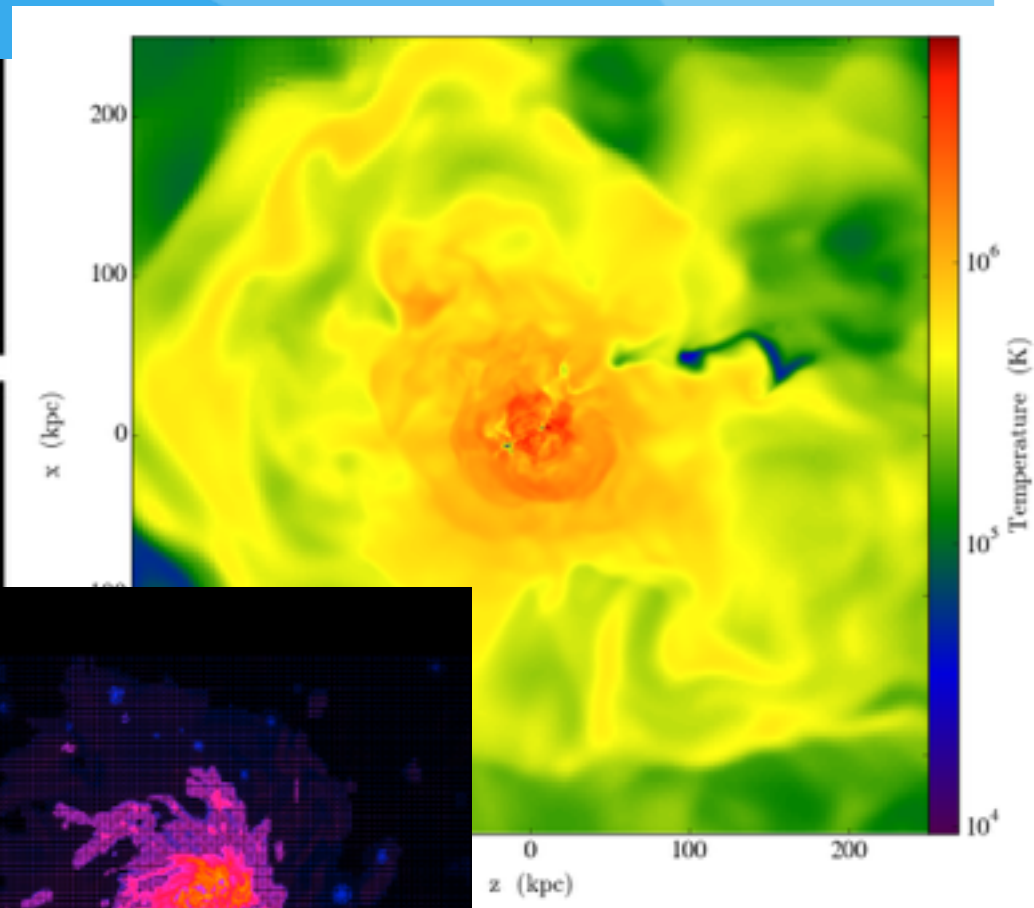
width = 111 kpc

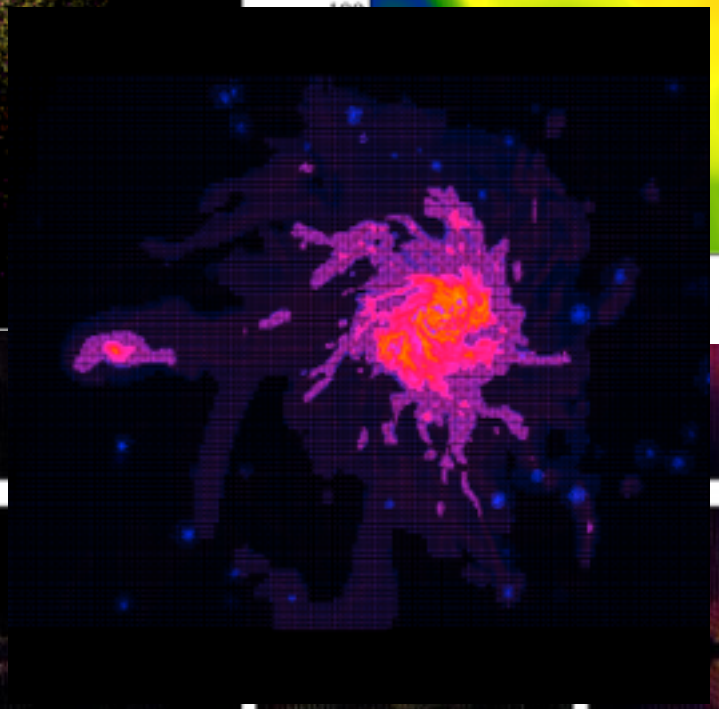
Stellar Component (1e6 DM particles inside Rvir) Z=0

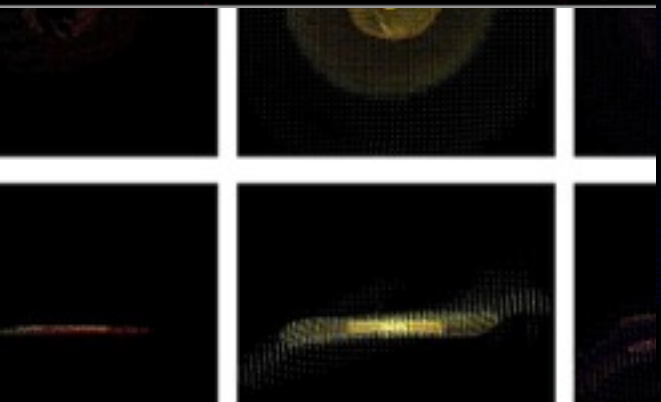


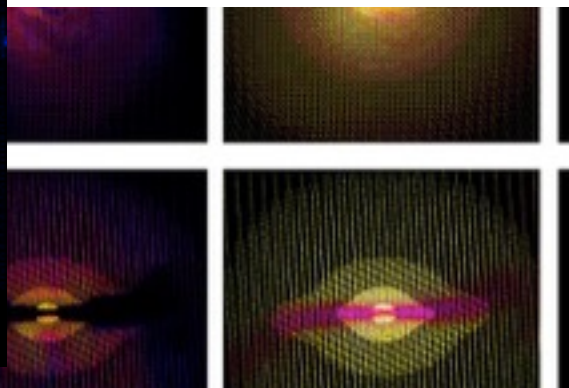


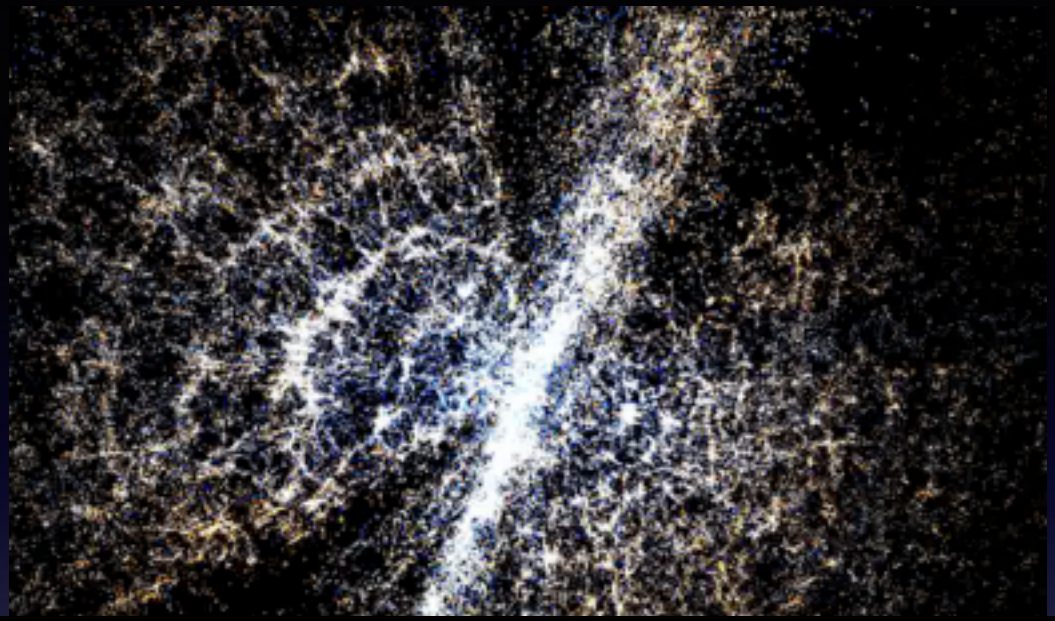



200
100
0
x (kpc)
0
100
200
z (kpc)
10^6
10^5
10^4
Temperature (K)









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:

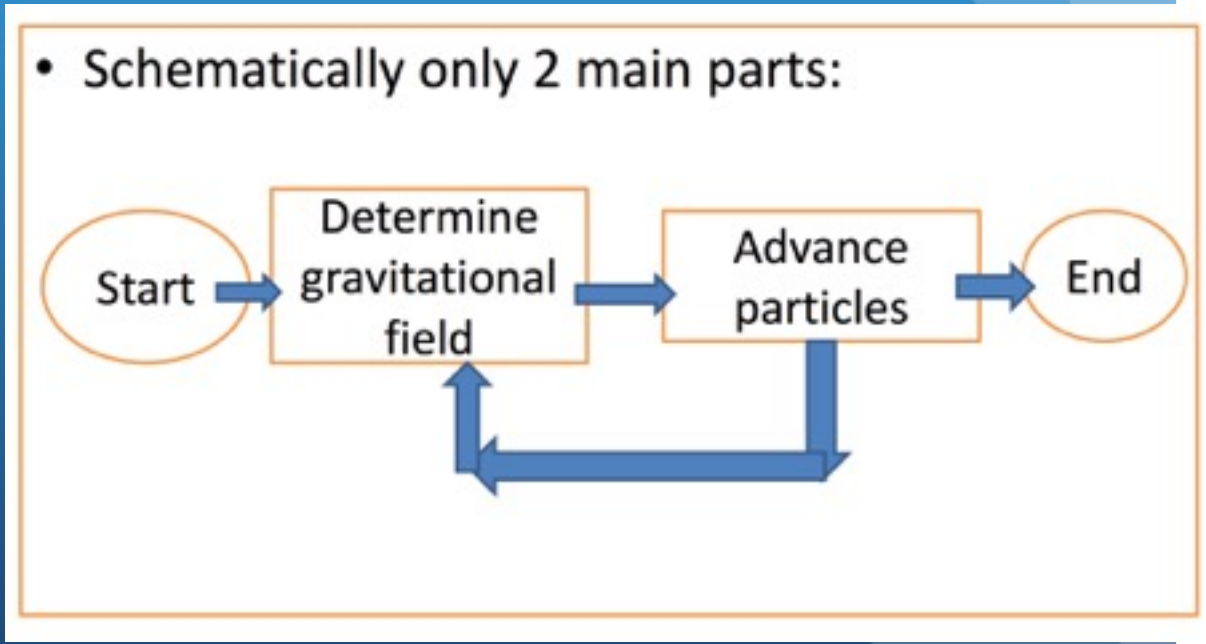
- Easier than analytic solution
- The only robust solution
- The most realistic approach

- 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

General Structure of N-body codes

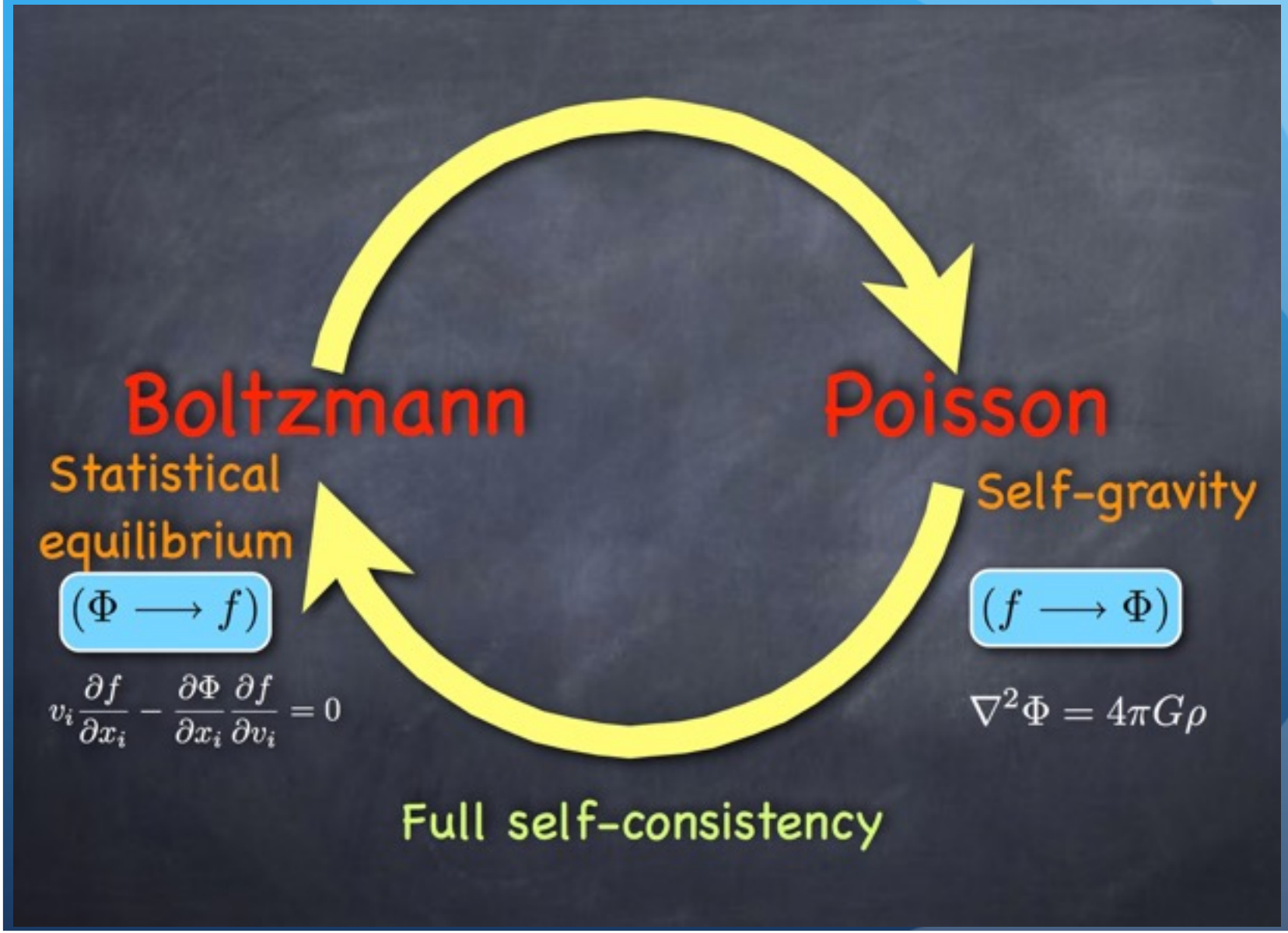
- Density/Force Calculation: direct summation, boundary conditions problem
- Time Integration



Initial Conditions: The Art

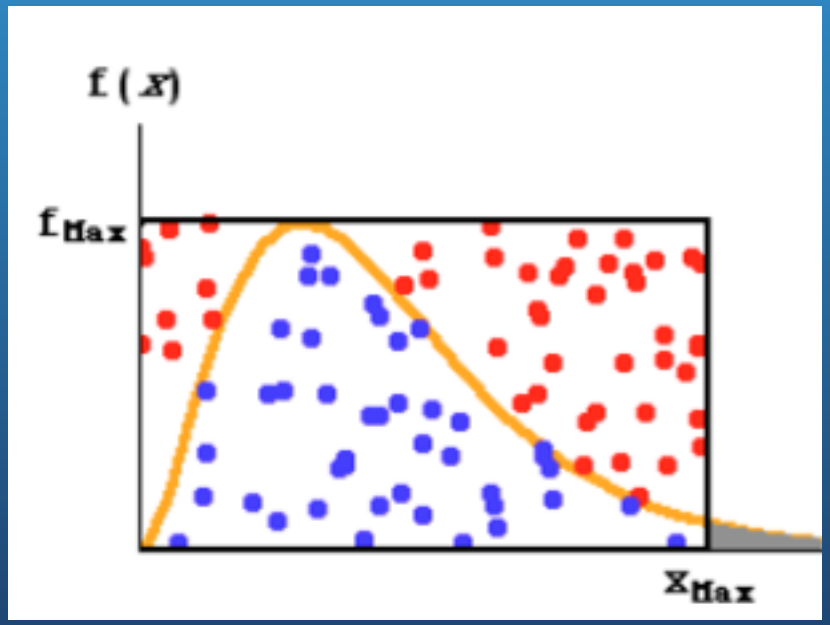
- Isolated
- Galaxies, halos, etc

Ideally we would like...



Method I: Phase Space Function

- Sample CBE solution



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

Random sampling?

Quiet Start (Sellwood)

Inhibits scatter

1st project: Spherical System

- Use a random number generator (e.g. rand3)
- Generates particles in a homogeneous cubic box with side L=2R centered at the origin
- Reject particles outside radius=R
- Measure radial number density of particles
- assign vx=vy=vz=0
- m=cte=1
- Evolve the system

Tipsy Ascii Format

- http://www-hpcc.astro.washington.edu/tools/tipsy/tipsy.html

```
ntotal, ngas, nstar

ndimensions

time

mass(i), i = 1 to ntotal

x_position(i), i = 1 to ntotal

y_position(i), i = 1 to ntotal

z_position(i), i = 1 to ntotal

x_velocity(i), i = 1 to ntotal

y_velocity(i), i = 1 to ntotal

z_velocity(i), i = 1 to ntotal

grav_softening_length_dark_particles(i), i =  1  to
ndark

grav_softening_length_star_particles(i), i =  1  to
nstar

density(i), i = 1 to ngas

temperature(i), i = 1 to ngas

sph_smoothing_length(i), i = 1 to ngas,

metals_gas(i), i = 1 to ngas,

metals_star(i), i = 1 to nstar,

formation_time(i), i = 1 to nstar,

potential_energy(i), i = 1 to ntotal,
```

Isochronal Disk

This is great but… are disk like that? Halo? Bulge?

$$\phi = \frac{GM}{b + r_b}, {r_b}^2 = R^2 + b^2$$

$$\Sigma = Mb/2(ln((R + r_b)/b)) - R/r_b)$$

profiles are given by Pichon & Lynden Bell (1996)[21](Fig PLB 5). Finally Miyamoto's distribution i

$$f_M(\varepsilon, h) = \frac{(2m+3)}{2\pi^2}(-\varepsilon)^{2+2m}\ {}_2F_1\left(-m, -2-2m, \frac{1}{2}, \frac{-h^2}{2\varepsilon}\right),$$

where ${}_2F_1$ is a terminating Hypergeometric function of the second kind.

Choosing particles from a DF

- Most workers select particles randomly:
  - Generate a candidate particle with coordinates ($\boldsymbol{x}$,$\boldsymbol{v}$) in allowed ranges
  - Compute the value of $f(E,L)$ at this point
  - Generate another random variable $0 < \alpha < f_{max}$
  - Accept this particle only if $\alpha < f(E,L)$
  - Continue until enough particles are accepted

Equilibrium halo containing a disk

- Hernquist (1993) uses the Jeans equations
- Raha et al. (1991), Kuijken & Dubinski (1995) , Debattista & S (2000) iterative approach
- Holley-Bockelmann *et al*. (2005) use Eddington inversion
- Syer & Tremaine (1996) suggest a made-to-measure approach
- All these methods are much better than the rough (and computationally expensive) approach of growing the disk inside a halo ← caution

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

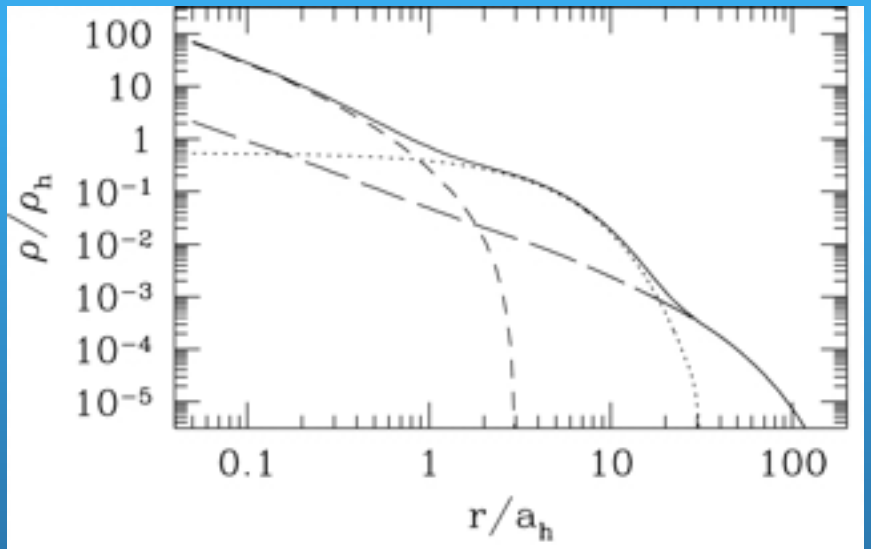
Because
<- Halo disk
interaction

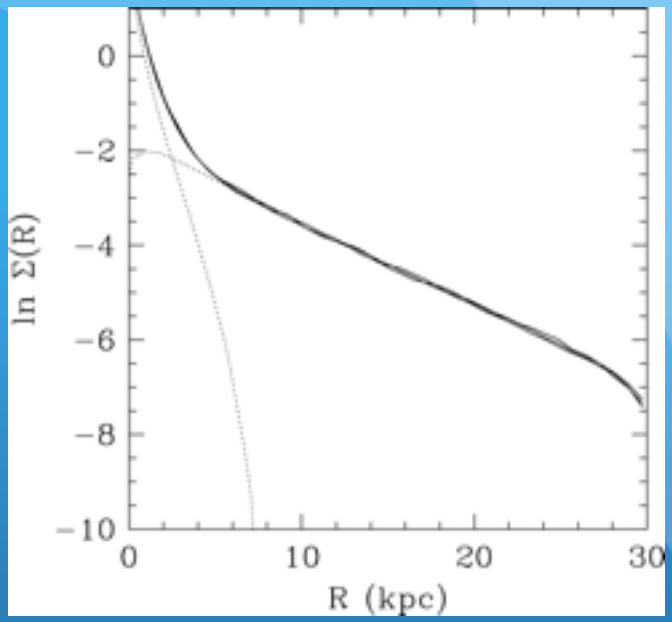
But the final stage is not always what we wanted
Sometimes not converging

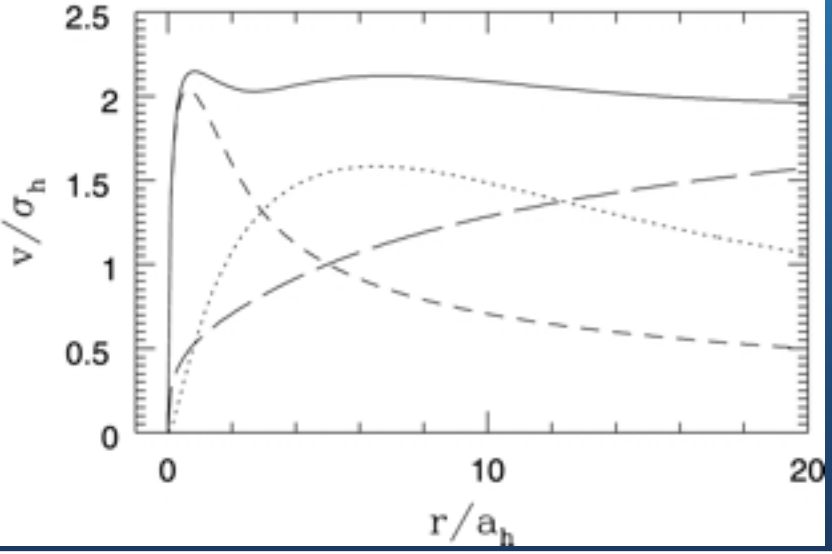
Starts assuming a spherical disk from the point of view of the halo

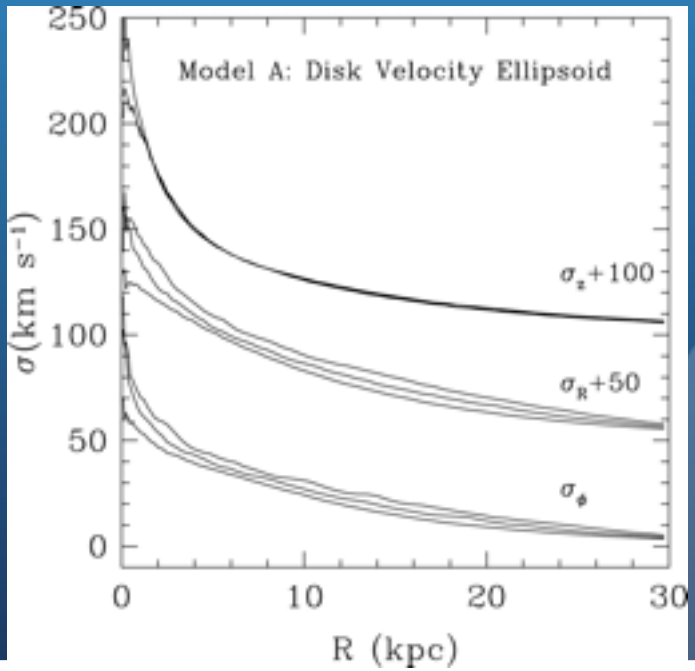
Excelent for controlled experiments

Implementation
Widrow Dubinsky









CBE Moments: Hernquist 1993

- Flexible regarding density structure
- Accuracy depends on the approximation order
- Problems if not epicyclic approximation is satisfied

$$\sigma_R^2(R) = Ae^{-\frac{R}{R_d}}.$$

$$\sigma_R(R_{\rm ref}) = Q\frac{3.36G\Sigma(R_{\rm ref})}{\kappa(R_{\rm ref})},$$

$$V_\phi^2 = V_c^2 - \sigma_R^2\left(\frac{2R}{R_d} + \frac{\kappa^2}{4\Omega^2} - 1\right),$$

$$\sigma_\phi^2 = \sigma_R^2\frac{\kappa^2}{4\Omega^2}.$$

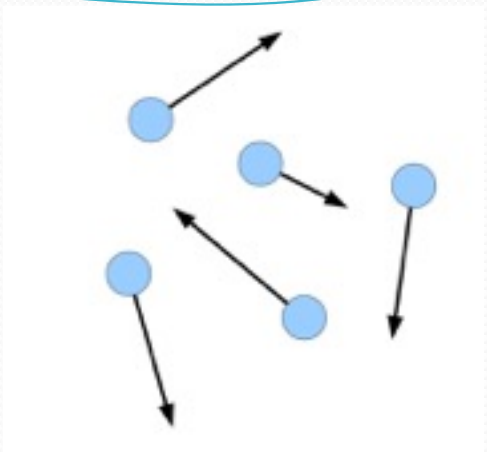
$$\sigma_z^2(R) = \pi G z_0 \Sigma(R),$$

Careful with subtle effects
Good for many applications

CODES: Which one is the best code?

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

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: $G = M_T = a_s = 1$

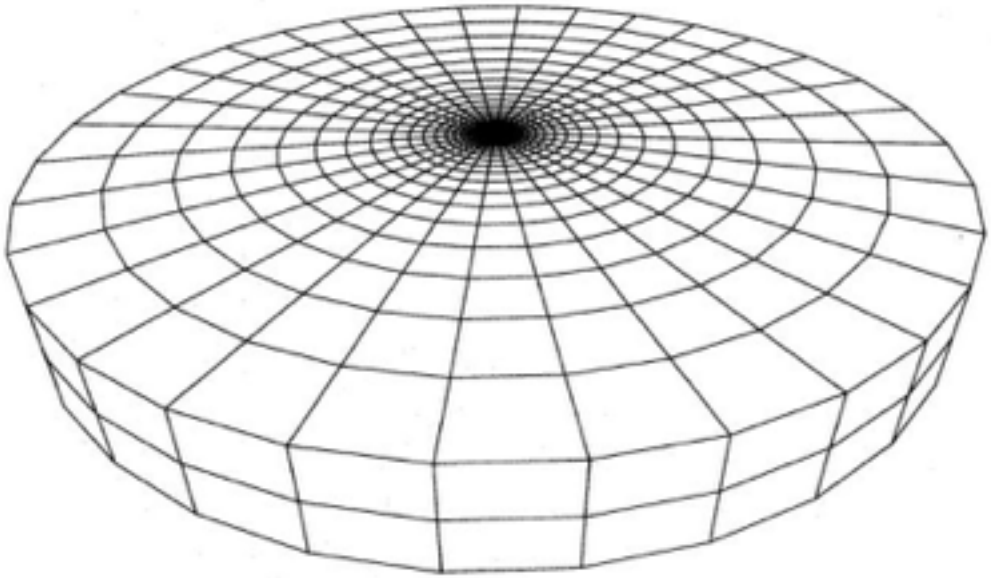
PP Codes

- 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?

- Fixed Shape, GRIDS, Orthogonal Base

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

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

Accuracy decreases

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

The Particle-Mesh Approach

•The idea is that Poisson's equation may be thought as a convolution in real space of the density field and a Green's function g(x).

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

Example for vacuum boundaries:

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

•In Fourier space, this convolution is simply a product between the transforms of the two functions, one of which is fixed and need to be computed/tabulated only once!

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

→ **Solve the potential in these steps:**

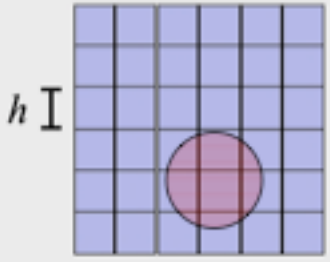
(1) FFT forward of the density field
(2) Multiplication with the Green's function
(3) FFT backwards to obtain potential

The four steps of the PM algorithm

(a) Density assignment
(b) Computation of the potential
(c) Determination of the force field
(d) Assignment of forces to particles

The Particle-Mesh Approach: Density Assignment

- Assuming that particles have a finite "shape" in 3D, we assign to each cartesian cell the fraction of the mass that falls into it.



- The overlap is given by:

$$W(\mathbf{x_m} - \mathbf{x}_i) = \int_{\mathbf{x_m}-\frac{h}{2}}^{\mathbf{x_m}+\frac{h}{2}} S(\mathbf{x}' - \mathbf{x}_i)\,\mathrm{d}\mathbf{x}' = \int \Pi\left(\frac{\mathbf{x}' - \mathbf{x_m}}{h}\right) S(\mathbf{x}' - \mathbf{x}_i)\,\mathrm{d}\mathbf{x}'$$

The assignment function is hence the convolution:

$$W(\mathbf{x}) = \Pi\left(\frac{\mathbf{x}}{h}\right) \star S(\mathbf{x}) \qquad \text{where} \qquad \Pi(x) = \begin{cases} 1 & \text{for } |x| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

- And the density of the mesh is a sum over all particles:

$$\rho(\mathbf{x_m}) = \frac{1}{h^3} \sum_{i=1}^{N} m_i\, W(\mathbf{x}_i - \mathbf{x_m})$$

The Particle-Mesh Approach: Particle Shape Functions

•These are some of the most common choices for shape and mesh assignment functions:

Name	Shape function S(x)	# of cells involved	Properties of force
NGP Nearest grid point	$\delta(\mathbf{x})$	$1^3 = 1$	piecewise constant in cells
CIC Clouds in cells	 $\frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right) \star \delta(\mathbf{x})$	$2^3 = 8$	piecewise linear, continuous
TSC Triangular shaped clouds	 $\frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right) \star \frac{1}{h^3}\Pi\left(\frac{\mathbf{x}}{h}\right)$	$3^3 = 27$	continuous first derivative

•Note that it is advisable to use similar grid assignment and interpolation schemes when loading the mesh and when interpolating the potential to obtain the forces.

The Particle-Mesh Approach: Advantages and Shortcomings

- The main advantage of the PM scheme is its speed and simplicity.
  - The cost of all steps scale as either N or N*log(N). It is also easy to port these algorithms to massively parallel computers.
- The principal shortcomings are:
  - Its reliance on a mesh of a defined geometry is prone to introducing artifact due to the anisotropy of the grid.
  - The spatial resolution is limited to the mesh size. In 3D and on powerful workstations, 1024^3 or 2048^3 FFT are the current practical limit.
    - For a box of size 50 Mpc/h, this implies a spatial resolution not better than 24 kpc/h, larger than a typical galaxy!
- *An ideal method would combine the speed and simplicity of the PM scheme but increasing the limited dynamic range imposed by the mesh size.*

More PM caveats

- FFT is very efficient
- Regular grid wastes computation time is 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 solver using Fourier analysis

Use of Fast Fourier Transform to solve for the Poisson equation

Poor's man Poisson solver:

$$\frac{\partial^2 \Phi}{\partial x^2} = \rho \qquad -k^2 \tilde{\Phi}(k) = \tilde{\rho}(k) \qquad \tilde{G}(k) = -\frac{1}{k^2}$$

$$\frac{\partial \Phi}{\partial x} = -F \qquad -ik\tilde{\Phi}(k) = \tilde{F}(k) \qquad \tilde{D}(k) = -ik$$

Using finite difference approximations:

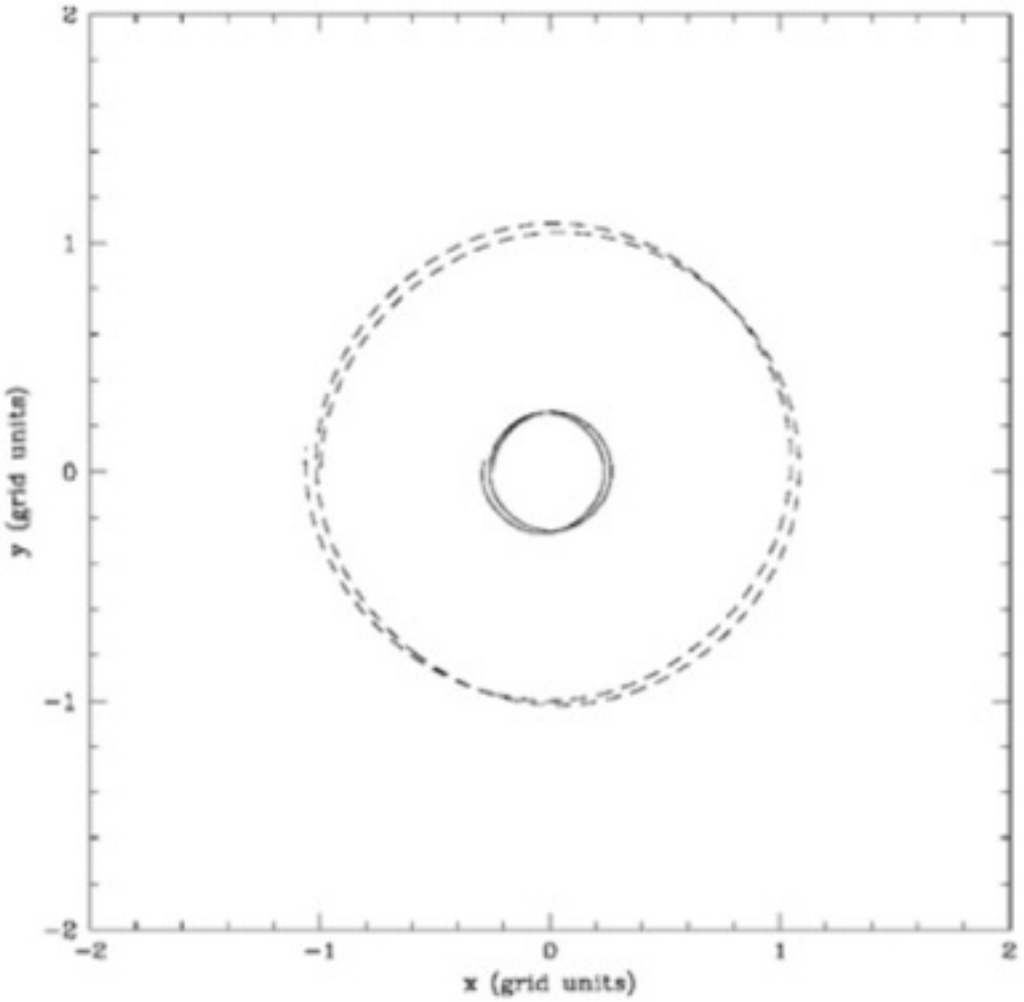
$$\Phi_{i+1} - 2\Phi_i + \Phi_{i-1} = \rho_i \Delta_x^2 \qquad \tilde{G}(k) = -\frac{\Delta x^2/4}{\sin(\frac{k\Delta x}{2})^2}$$

$$-(\Phi_{i+1} - \Phi_{i-1}) = F_i \Delta_x \qquad \tilde{D}(k) = -i\frac{\sin(k\Delta x)}{\Delta x}$$

Final force is given by:

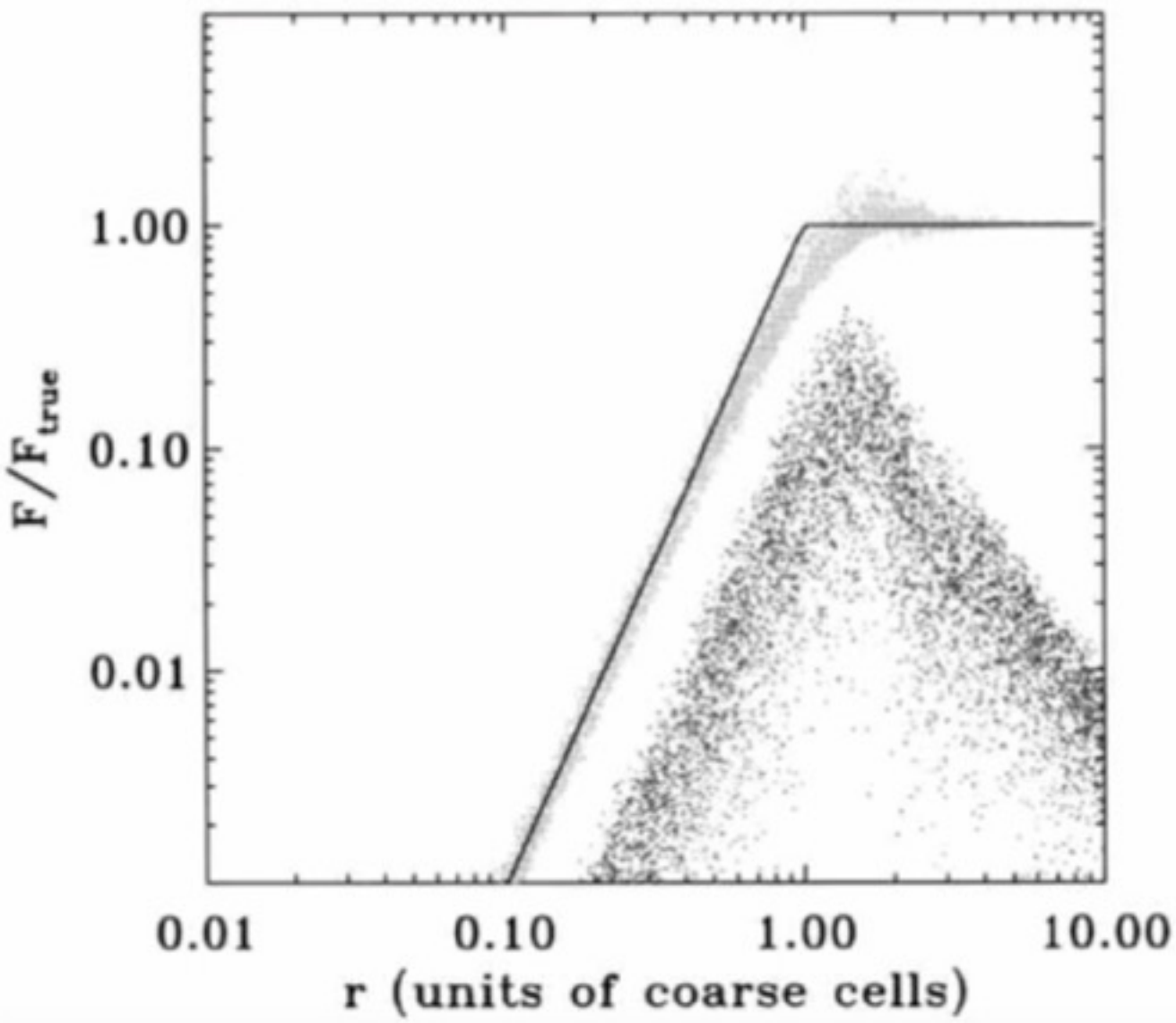
$$\tilde{F}(k) = -\frac{m_p^2}{\Delta x^2} \tilde{W}^F(k) \tilde{D}(k) \tilde{G}(k) \tilde{W}^\rho(k) \tilde{n}(k)$$

Force + integrator accuracy



examples of particle's trajectory

PM force accuracy



CIC

7-point Laplacian

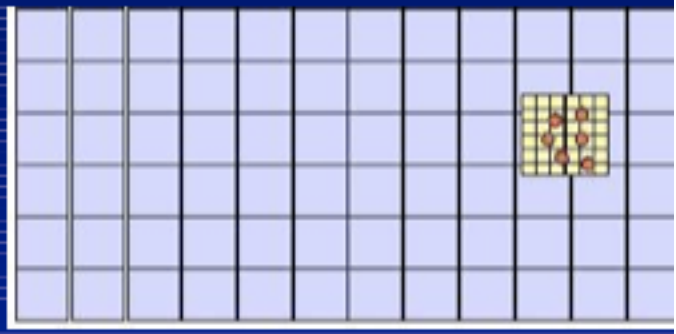
2 points gradient

CIC^{-1}

2nd Project: Download Klypin PMCode

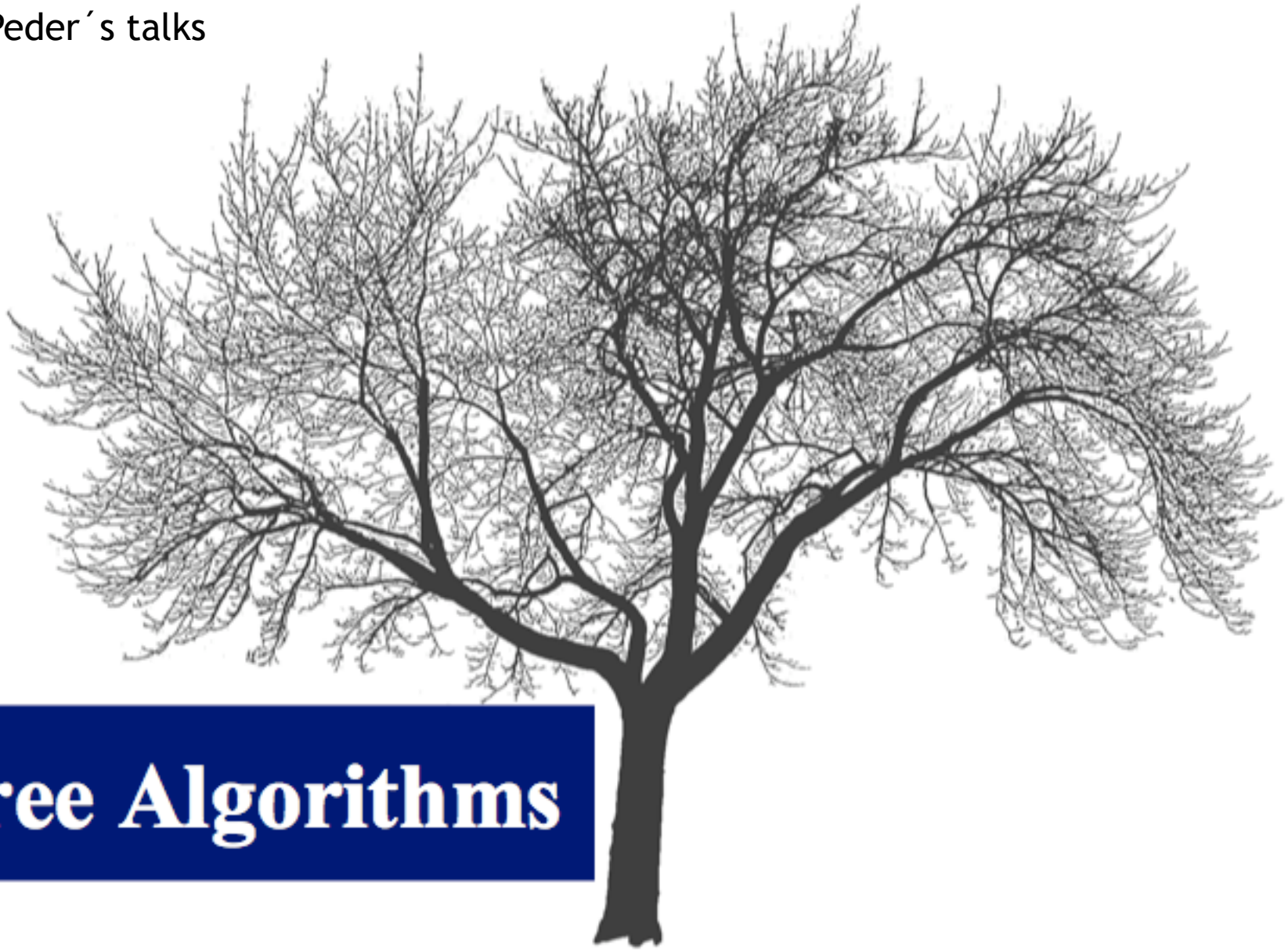
- Generate Initial Conditions for a LCDM Universe
- PMstartM, PMmodelsM
- Measure the Power Spectrum Empower
- Investigate effects of Box size, Grid size and Particle numbers in the IC Spectrum
- If you have time Evolve one of the models and investigate the same effects

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



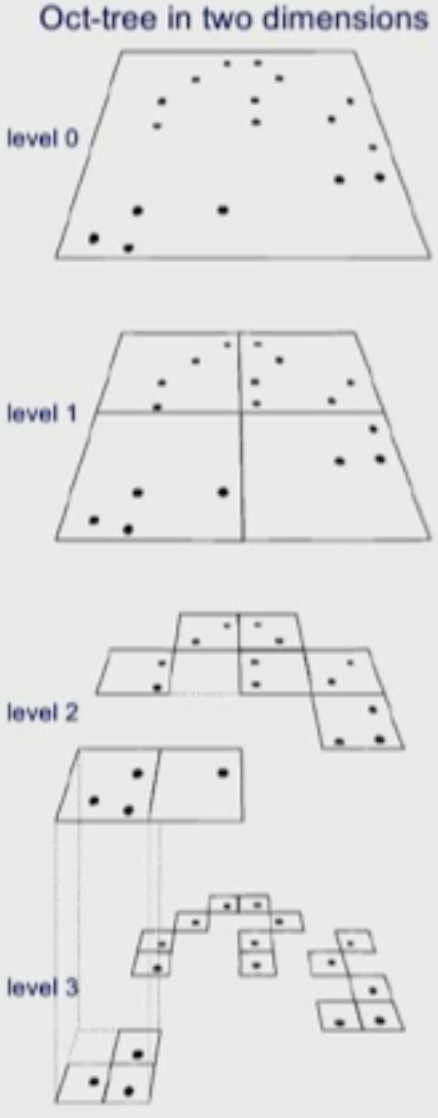
- This extends the PM method with a direct summation of short-range forces on the scale of the mesh cells.
  - GOOD: Increases substantially the spatial dynamic range
  - BAD: Highly-clustered states (when the short range force calculations dominate) are very costly to evolve
- Mesh-refinements may be 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

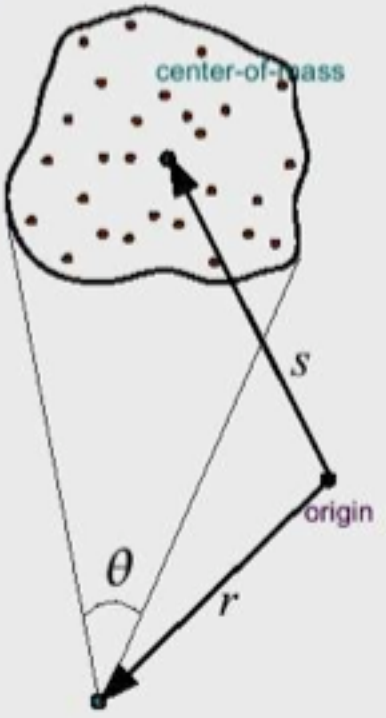


Tree Algorithms

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

dipole contribution vanishes outside each node

- And get:

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

Cosmological simulations with GADGET - Mozilla Firefox

Archivo Editar Ver Ir Marcadores Herramientas Ayuda

http://www.mpa-garching.mpg.de/gadget/

SUSE Linux | Entertainment | News | Internet Search | Reference | Maps and Directions | Shopping | People and Companies

GADGET - 2

A code for cosmological simulations of structure formation

General
- Description
- Features
- Authors and History
- Acknowledgments
- News

Software
- Download
- Requirements
- License
- Mailing List
- Change-Log
- Examples

Documentation
- Code Paper
- Users Guide
- Code Reference

Publications
- Scientific Papers

Features

look Marios´s
and Peder´s talks

- 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

Aplicaciones Lugares Escritorio lun 29 de oct, 23:05

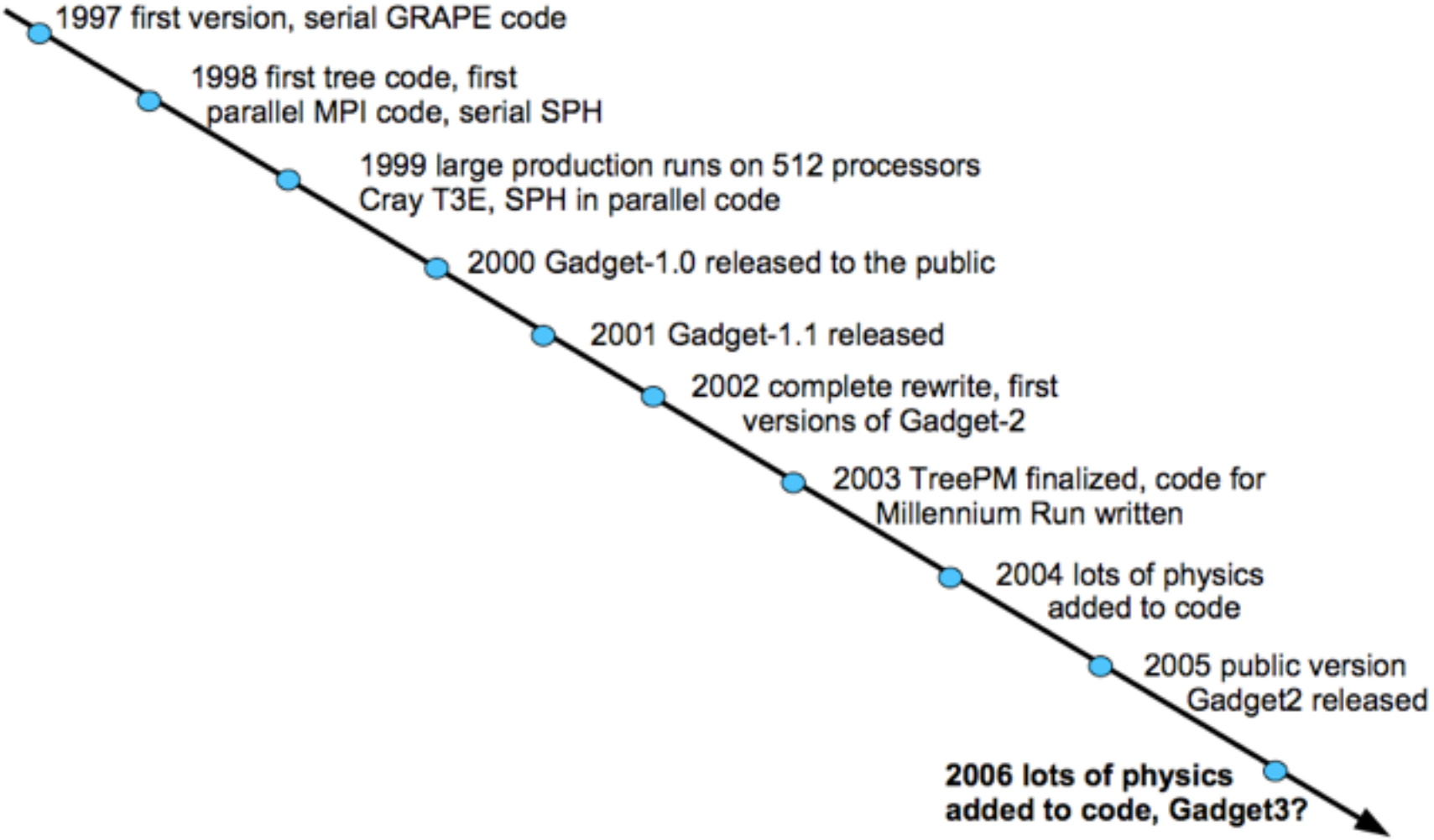
GADGET is a versatile TreeSPH N-body code for cosmological applications

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

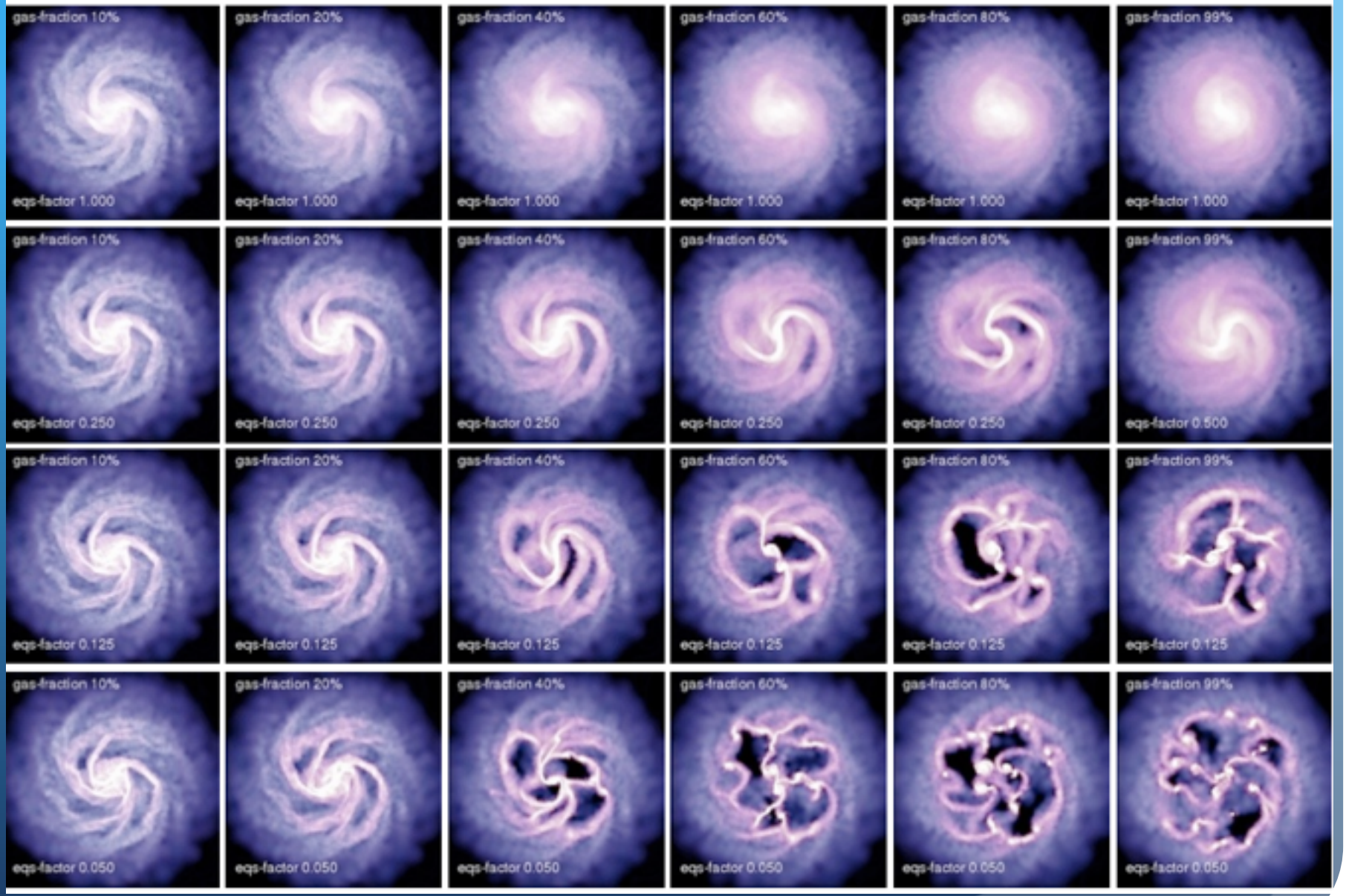
GADGET has evolved over the years and is in a process of continuous change

MAJOR EVENTS IN GADGET'S DEVELOPMENT



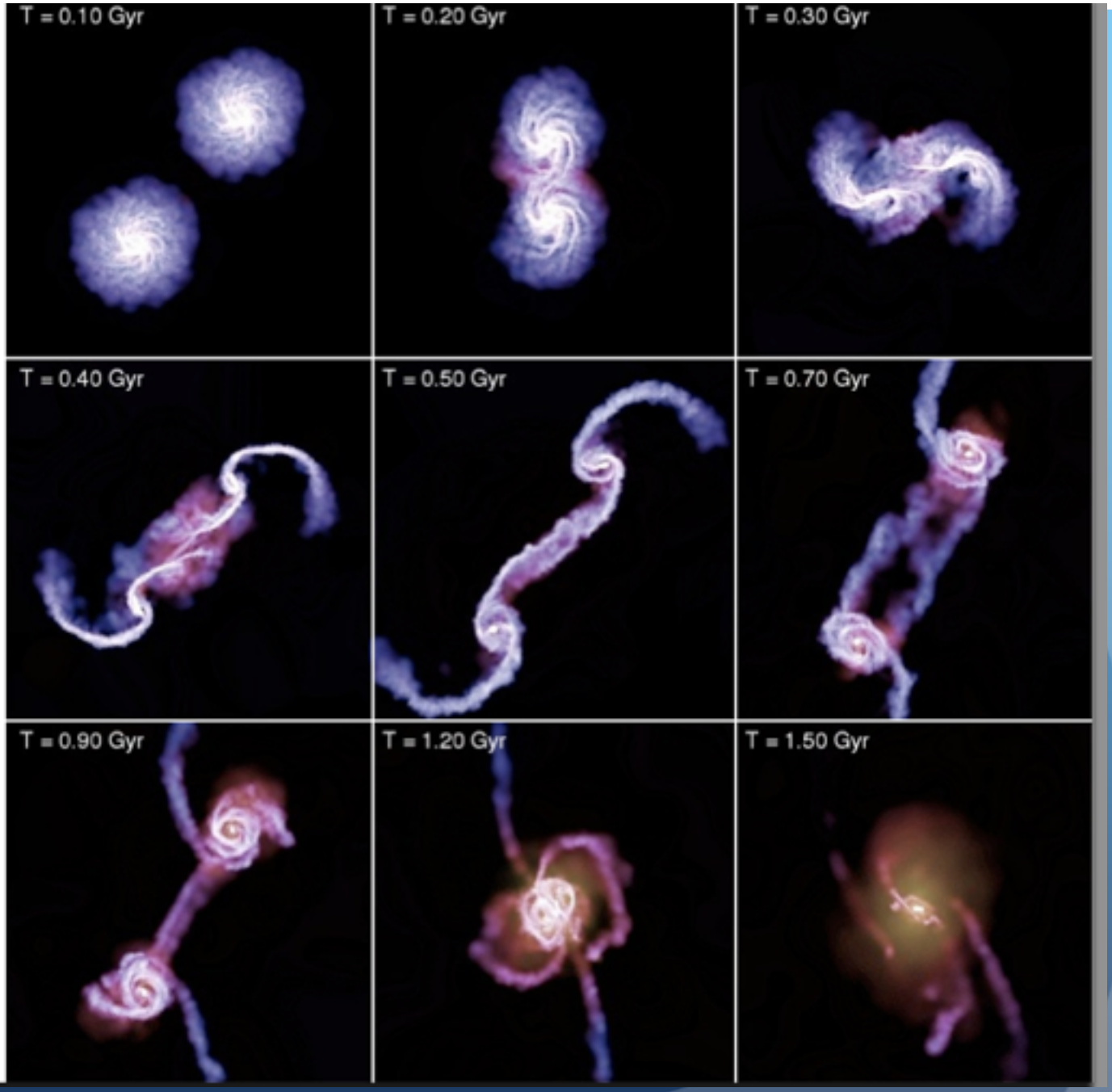
The multiphase-model allows stable disk galaxies even for very high gas surface densities

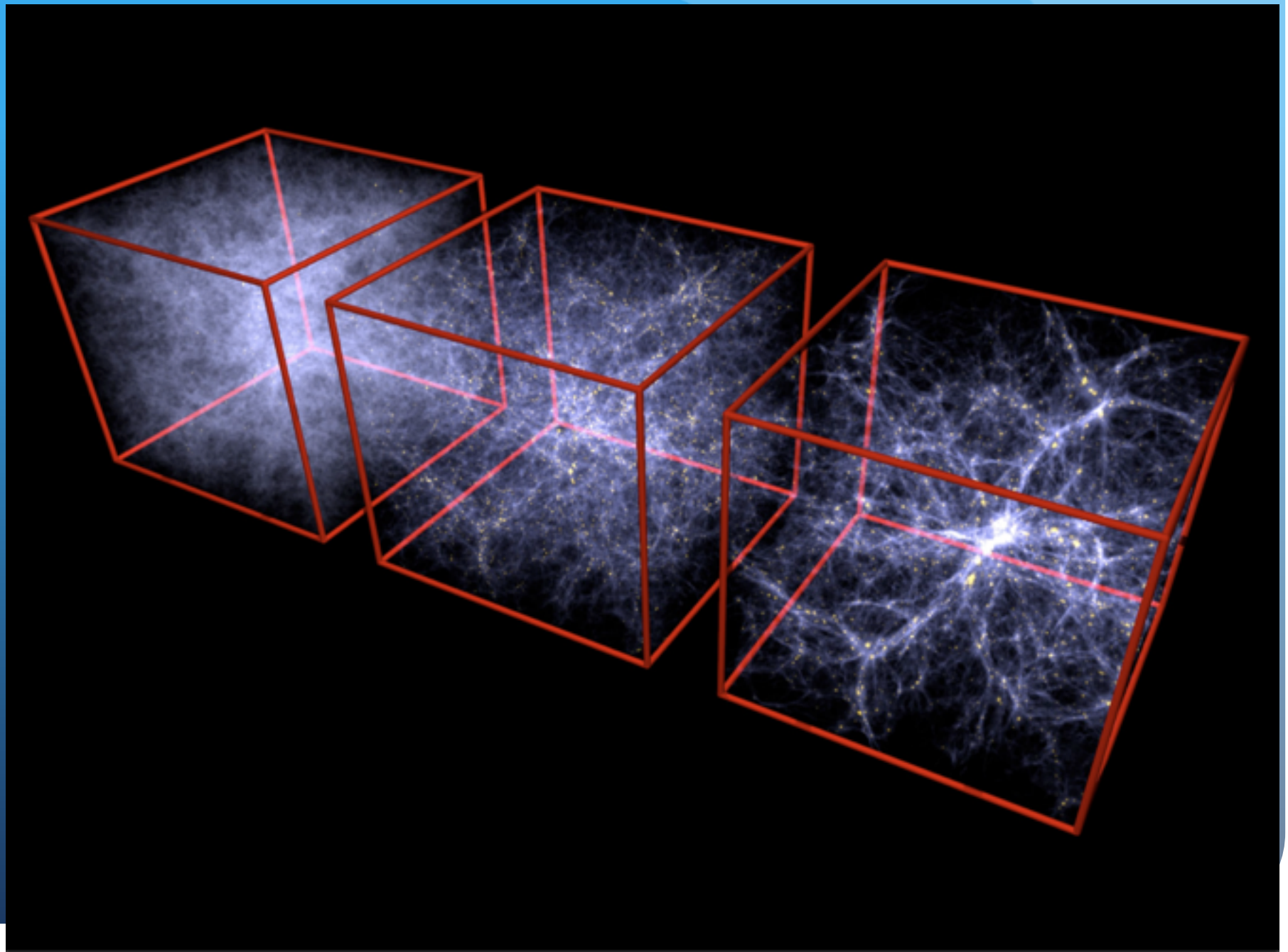
STABILITY OF DISKS AS A FUNCTION OF GAS FRACTION AND EQUATION OF STATE



In major-mergers between two disk galaxies, tidal torques extract angular momentum from cold gas, providing fuel for nuclear starbursts

TIME EVOLUTION OF A PROGRADE MAJOR MERGER

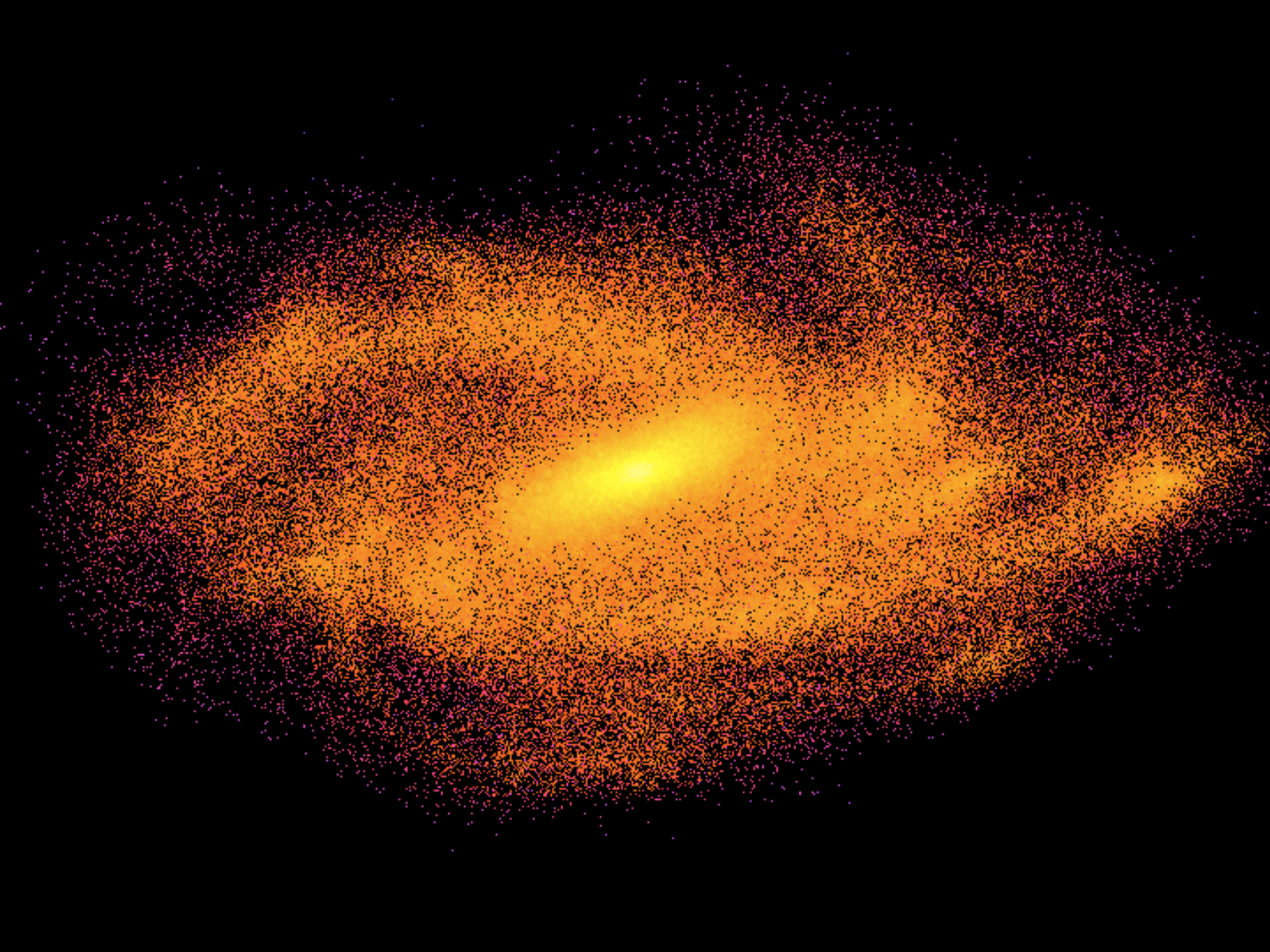




Gasoline

- 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 PhD Thesis



The quest for higher resolution…

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

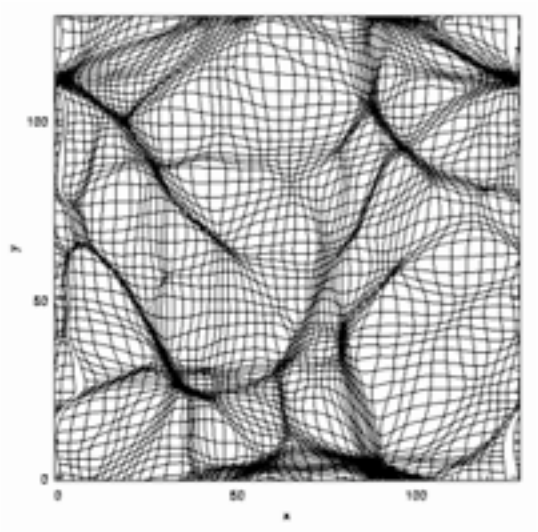
- an additional shrinking can be due to expansion if integrating on a grid fixed in comoving coordinates

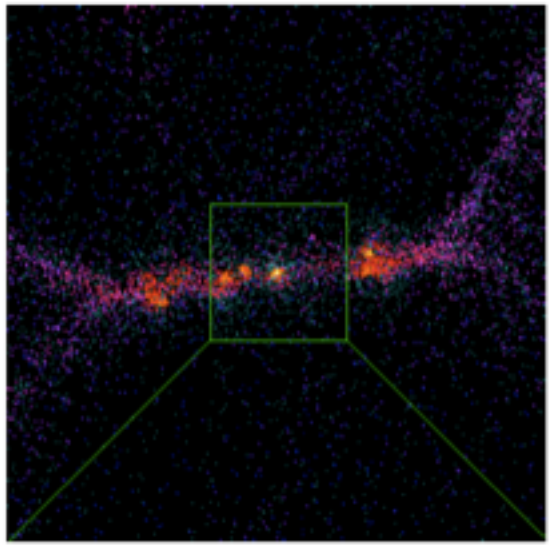
- fixed uniform grid methods are therefore not suitable to study halo structure in the highly nonlinear regime or galaxy formation

Lagrangian Approach

- Smoothed Lagrangian Hydrodynamics (SLH)
  [Gnedin 1996; Pen 1997]

- Smoothed Particle Hydrodynamics (SPH)





Alternative approach is AMR…

In numerical analysis, central to any Eulerian method is the manner in which it discretizes the continuous domain of interest into a grid of many individual elements. This grid may be *static*, established once and for all at the beginning of the computation, or it may be *dynamic*, tracking the features of the result as the computation progresses. If the computation has features which one wants to track which are much smaller than the overall scale of the problem, and which move in time, then one must either include many more static grids to cover the region of interest, or adopt a dynamic scheme.

The advantages of a dynamic gridding scheme are:

1. Increased computational savings over a static grid approach.
2. Increased storage savings over a static grid approach.
3. Complete control of grid resolution, compared to the fixed resolution of a static grid approach, or the Lagrangian-based adaptivity of smoothed particle hydrodynamics.

http://en.wikipedia.org/wiki/Adaptive_mesh_refinement

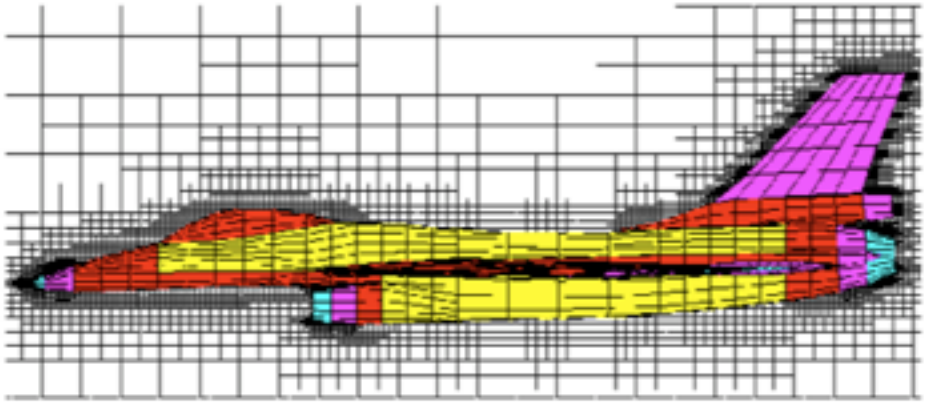
The AMR Approach

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

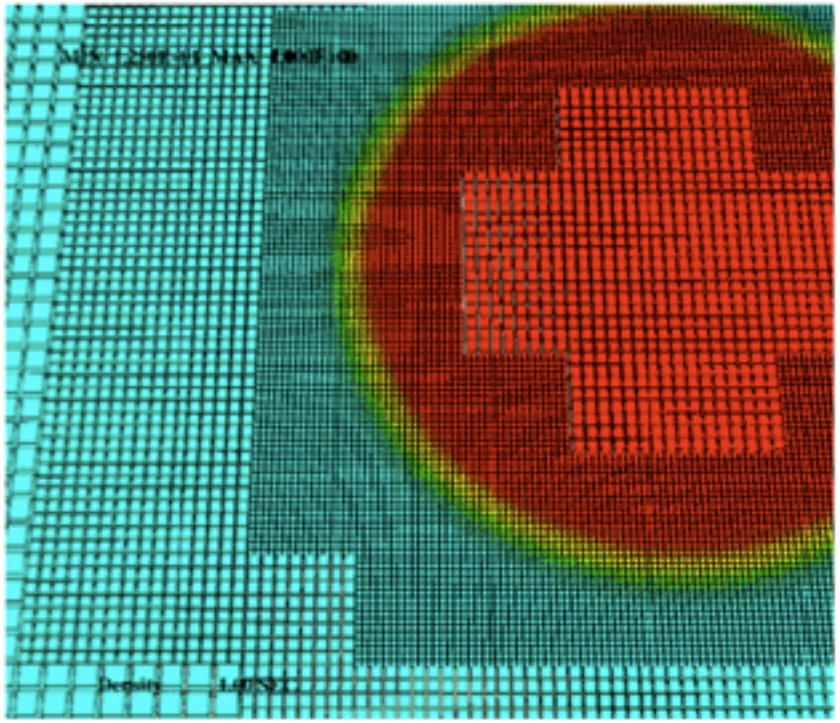
- The *Adaptive Mesh Refinement* (AMR) methods increase the dynamic range of grid-based numerical algorithms beyond the limits imposed by existing hardware.

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

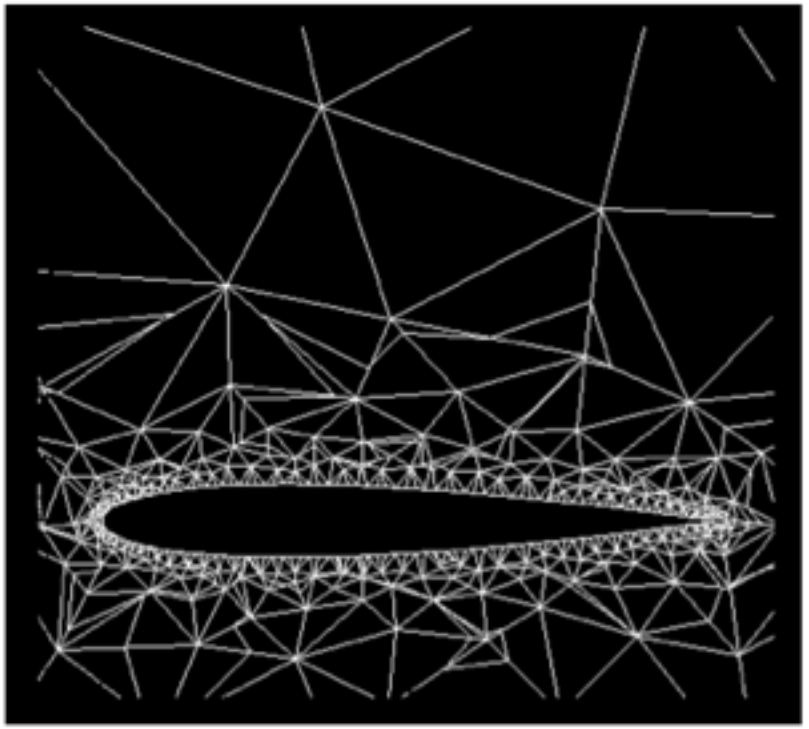
- Now gaining popularity in astrophysics and cosmology



Structured vs unstructured AMR



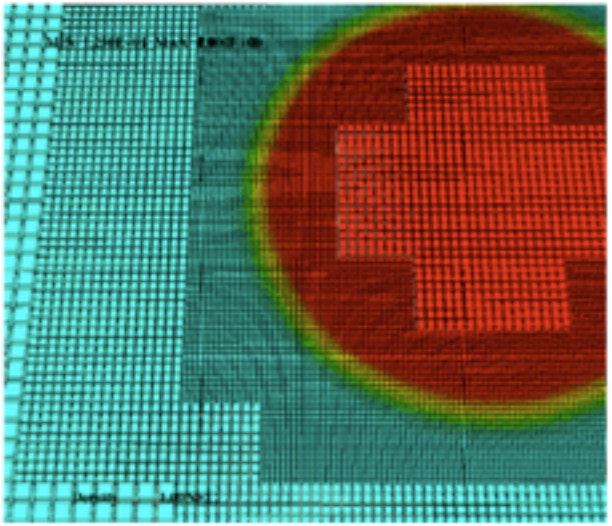
Structured: hierarchy of rectangular grids or irregularly shaped meshes of cubic cells

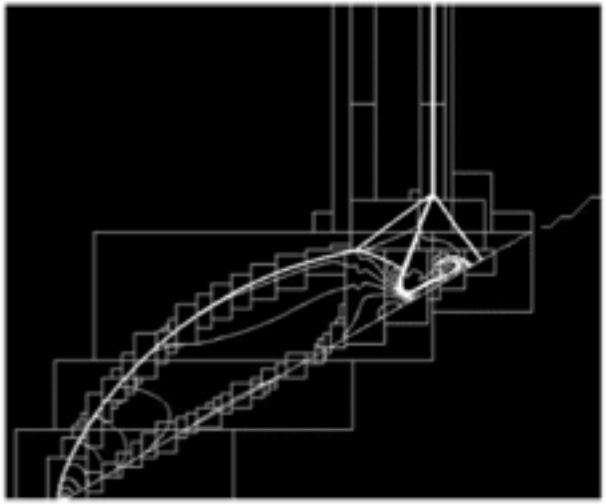


Unstructured: highly flexible refinement meshes, efficient for cases of complicated region geometry and boundaries; more sophisticated data structures and algorithms

Arepo uses something
Like this one

Structured AMR



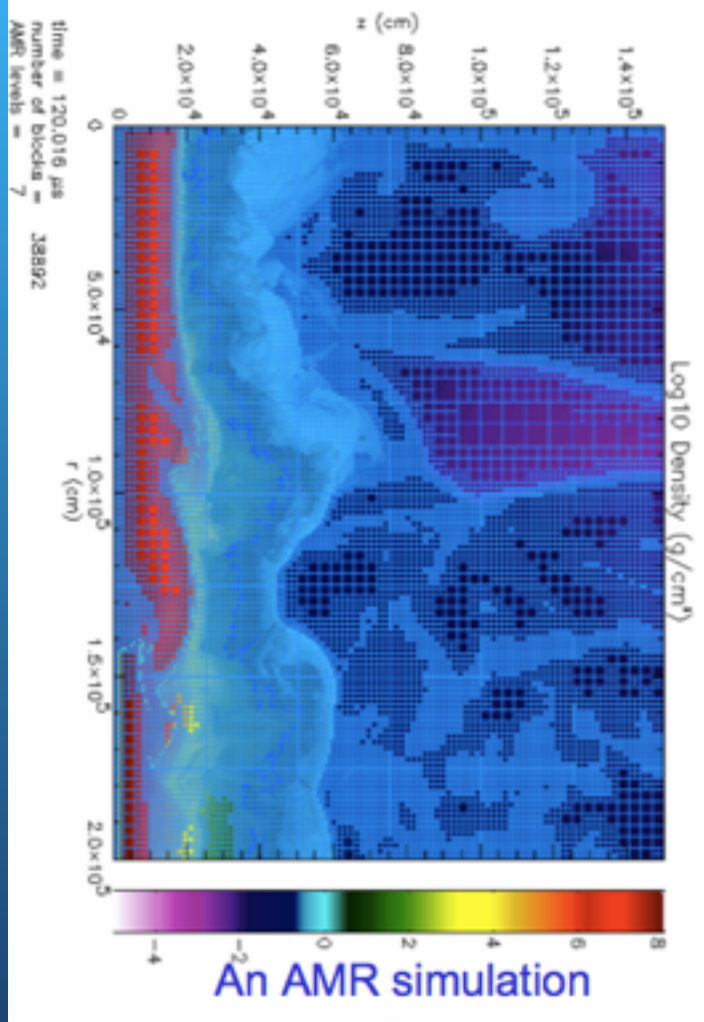


- Is most widely used in astrophysics, as there is no particular motivation to use the unstructured meshes. The finite difference techniques are simpler and often more stable and accurate with cubic cells

- First methods developed by Marsha Berger, Joseph Oliger, and Phillip Colella in the early 1980s

- The first algorithms used rectangular grids organized in hierarchy of meshes with levels of the hierarchy defined by the grid cell size (resolution)

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^{3}M (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 and Galactic AMR codes

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

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 code, numerical techniques, and analysis

In Alexander Knebe website

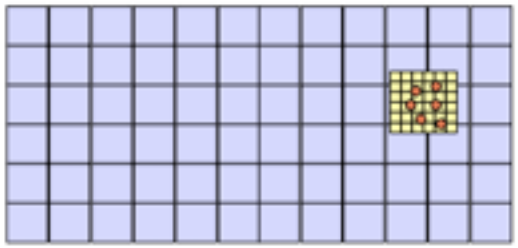
AP3M: HYDRA

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.

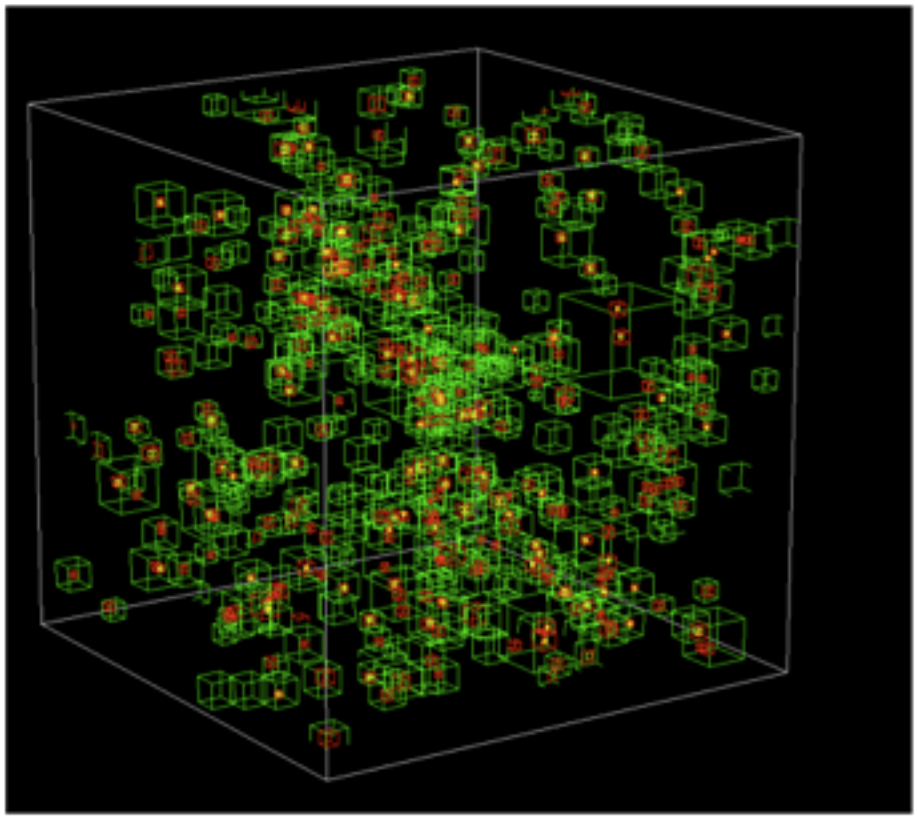
In AP^3M, mesh-refinements are placed on clustered regions



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)



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

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

- 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

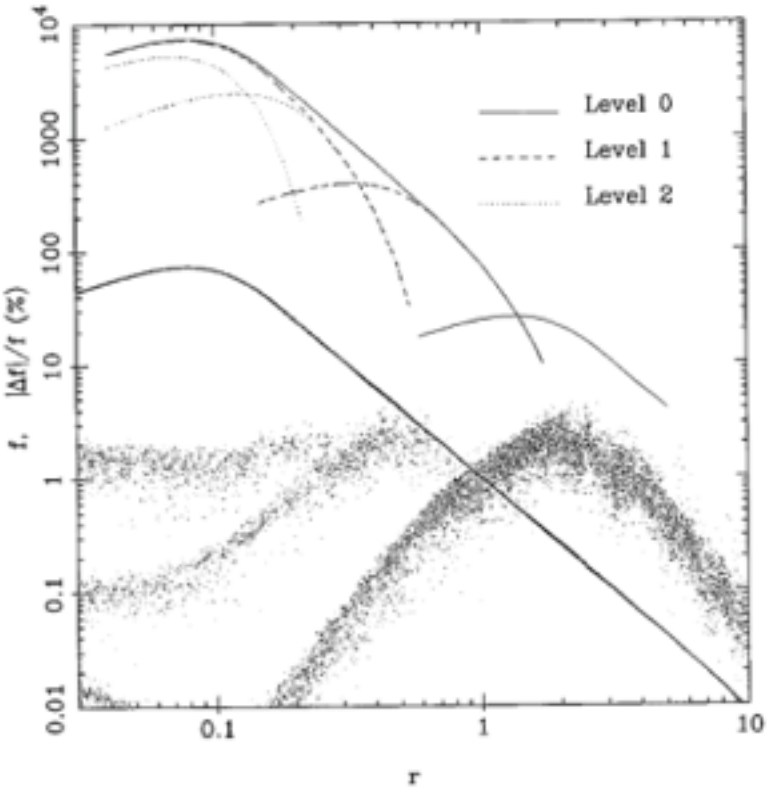


FIG. 1.—The pairwise force as a function of radius calculated on a 32^3 periodic mesh with two levels of refinement. The heavy continuous line is the calculated force, and the dots indicate the modulus of the error in the force. The curves at the top of the plot indicate the components of the force (×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

- 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*

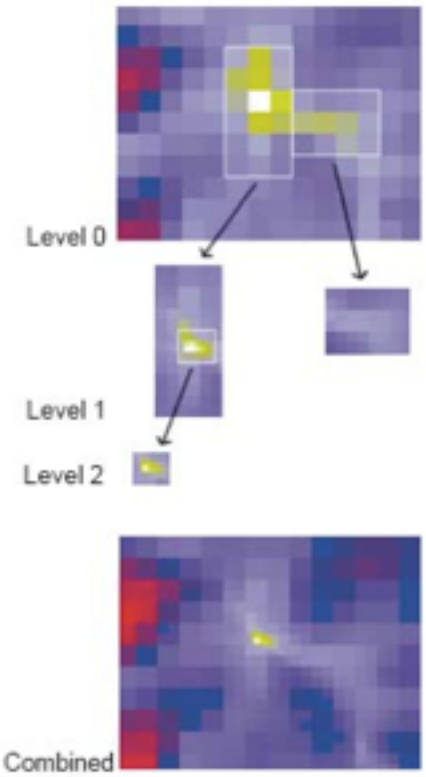
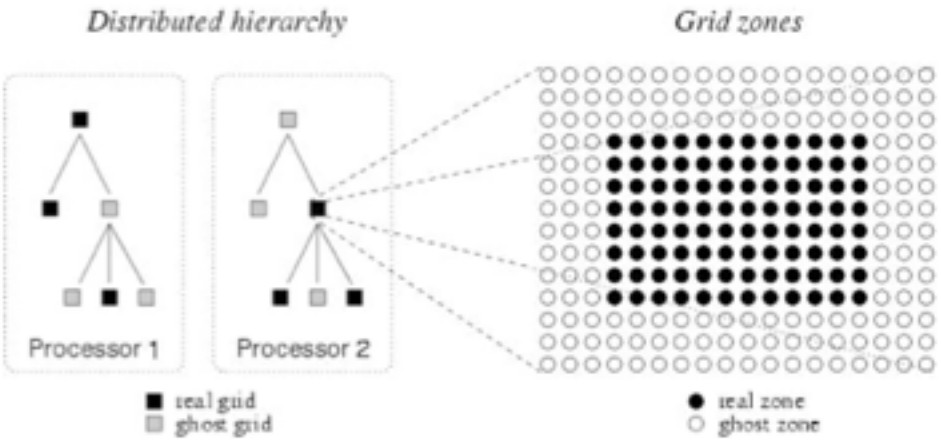
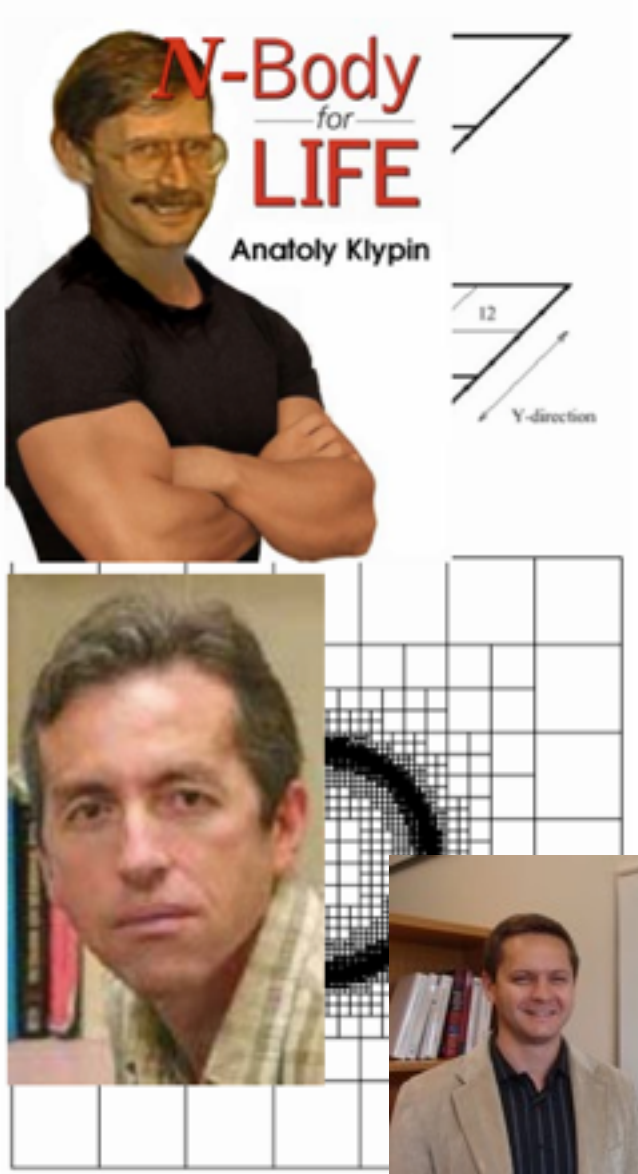


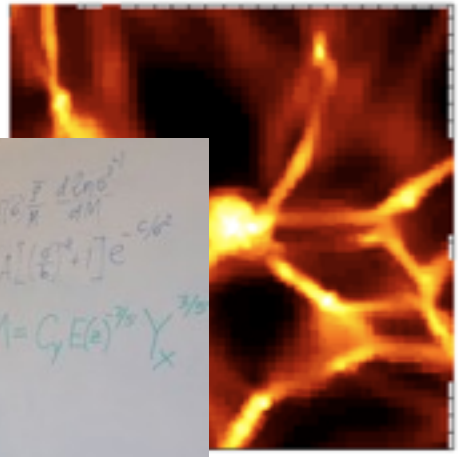
Figure 1. A 2D adaptive mesh refinement example showing a three-level, four-grid hierarchy.

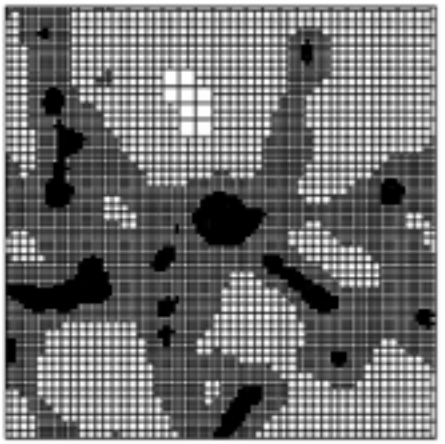


Adaptive Refinement Tree (ART) code



- 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.

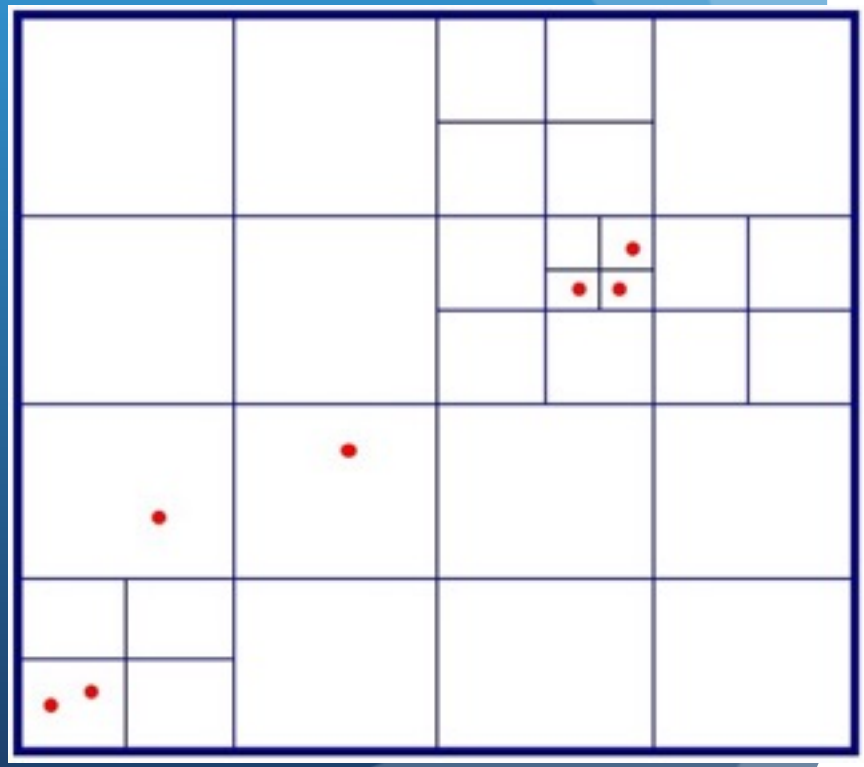




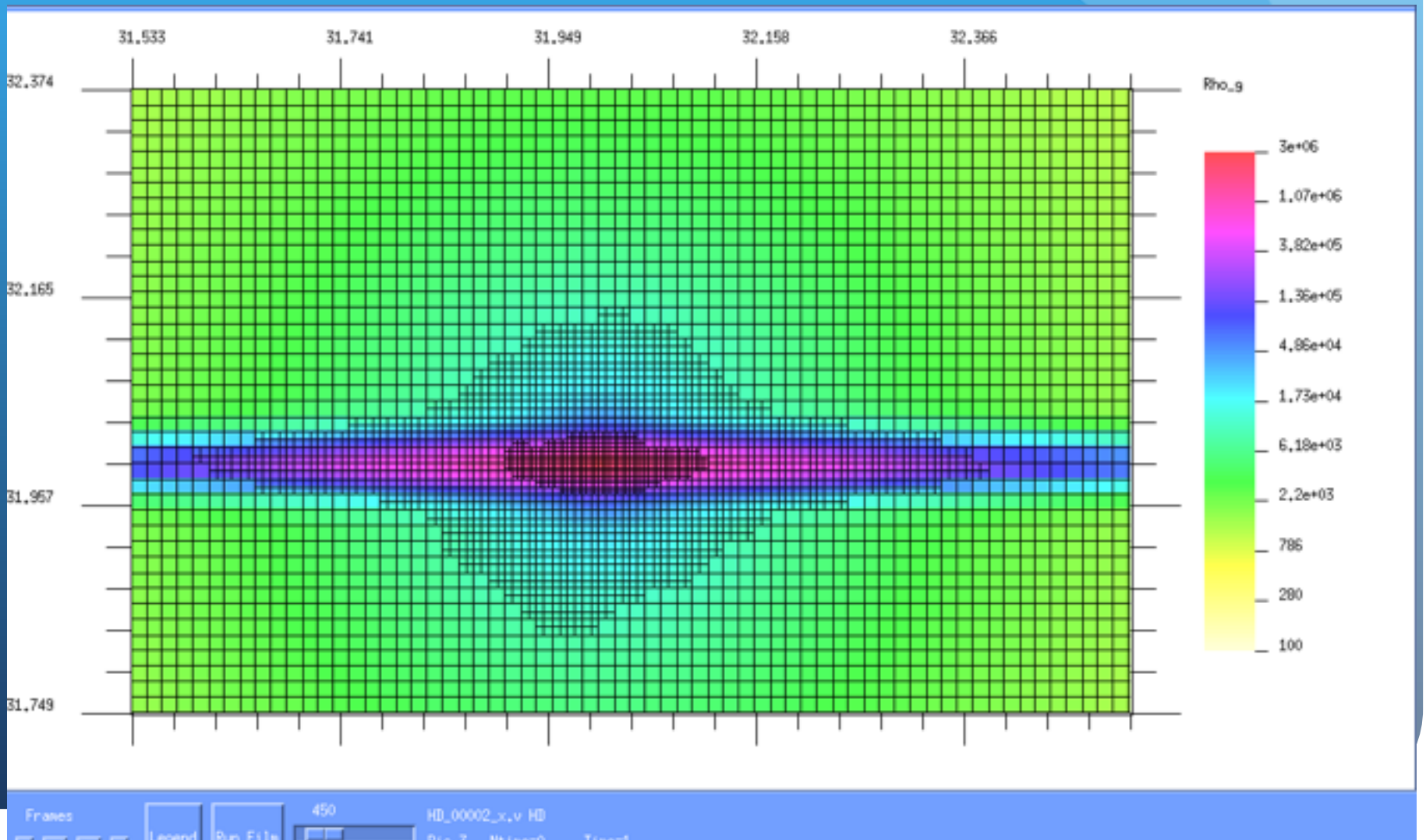
ART Code
(Adaptive Refinement Tree)
(Kravtsov et.al. 1997)

Equations of characterisics for the Vlasov equation coupled with the Poisson equation take the form of "N-body problem":

$$\frac{d\mathbf{p}}{da} = -\frac{\nabla\phi}{\dot{a}}$$
$$\frac{d\mathbf{x}}{da} = \frac{\mathbf{p}}{\dot{a}a^2}$$
$$\nabla^2\phi = 4\pi G a^2 \Omega_{\rm dm}\delta_{\rm dm}\rho_b$$
$$\dot{a} = \frac{H_0}{\sqrt{a}}\sqrt{\Omega_0 + \Omega_{\rm curv,0}\cdot a + \Omega_{\Lambda,0}\cdot a^3}$$



Hydro ART


31.533
31.741
31.949
32.158
32.366
32.374
32.165
31.957
31.749
Rho_g
3e+06
1.07e+06
3.82e+05
1.36e+05
4.86e+04
1.73e+04
6.18e+03
2.2e+03
786
280
100
Frames
450
HD_00002_x.v HD

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

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

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

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

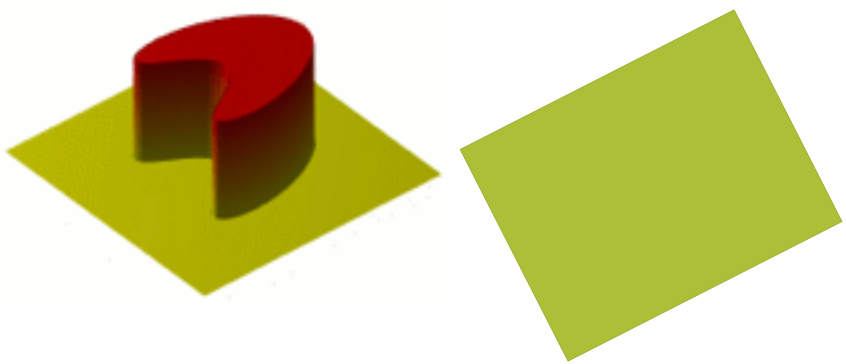
- red-black Gauss-Seidel relaxation scheme with successive over-relaxation (SOR) further speeds up the convergence (Hockney & Eastwood 1988; also, Numerical Recipes)

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

Problema de optimización: Técnica de relajación.

Ec. de Calor.

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

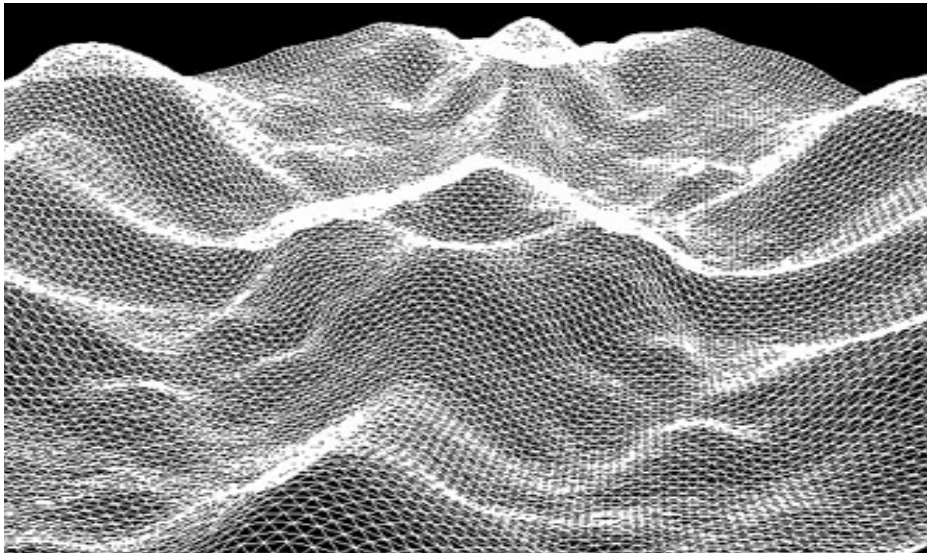


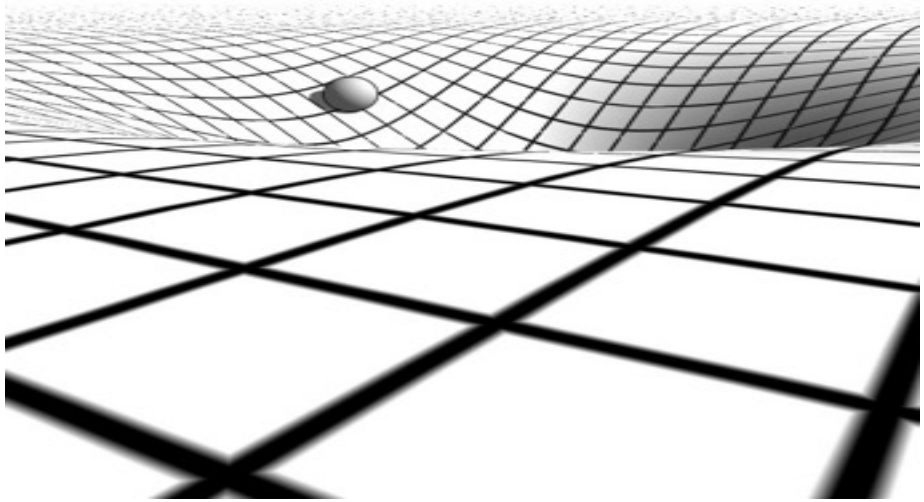
Ec. de Difusión.

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

Ec. de Boltzmann.

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





Relaxation solvers for the Poisson equation

Solve the linear system $\Delta_{ij}\Phi_j = \rho_i$ with arbitrary mesh geometry.

Simplest scheme: the Jacobi or Gauss-Seidel method (in 2D).

$$\phi_{i,j}^{n+1} = \frac{1}{4}\left(\phi_{i+1,j}^{n} + \phi_{i-1,j}^{n} + \phi_{i,j+1}^{n} + \phi_{i,j-1}^{n}\right) - \frac{1}{4}\rho_{i,j}$$

Converge very slowly for long wavelength and large grids.

Very sensitive to the initial guess.

Faster convergence is obtained for Gauss-Seidel "over-relaxation" method with red-black ordering.

$$\phi_{i,j}^{n+1} = \omega\phi_{i,j}^{n} + (1-\omega)\phi_{i,j}^{n+1} \quad \text{with} \quad 1 < \omega < 2$$

Fastest convergence for $\omega \simeq \dfrac{2}{1+\alpha\frac{\pi}{N}}$

Similar performance with the Conjugate Gradient method. For a NxN grid: exact convergence in N^2 iterations,

In practice, order N iterations are necessary to reach the level of truncation errors `epsilon=1e-3 to 1e-4`

Multigrid solver for the Poisson equation

Use coarse-grid sampling to speed-up convergence at large scale.

Proposed by Brandt (1973) to solve elliptic problems.

Use smoothing properties of Jacobi and Gauss-Seidel scheme to reduce high-frequency modes in the error.

Use coarsening to reduce low-frequency modes at a faster rate.

Reduce the cost of relaxation solvers from N^2 to N lnN or even N.

Briggs, W.L., “A Multigrid Tutorial”, SIAM Monograph, (2000)

Two-grid scheme

On the fine grid, define the residual $r_\ell^n = \Delta_\ell \Phi_\ell^n - \rho_\ell$

and the error $e_\ell^n = \Phi_\ell^n - \Phi_\ell^\infty$

$$\Delta_\ell e_\ell^n = r_\ell^n$$

1- Perform a few J/GS iterations (smoothing).

2- Restrict the residual to the coarse grid:

$$r_\ell^n \longrightarrow r_{\ell-1}^0$$

3- Solve for the coarse grid system:

$$\Delta_{\ell-1} e_{\ell-1} = r_{\ell-1}$$

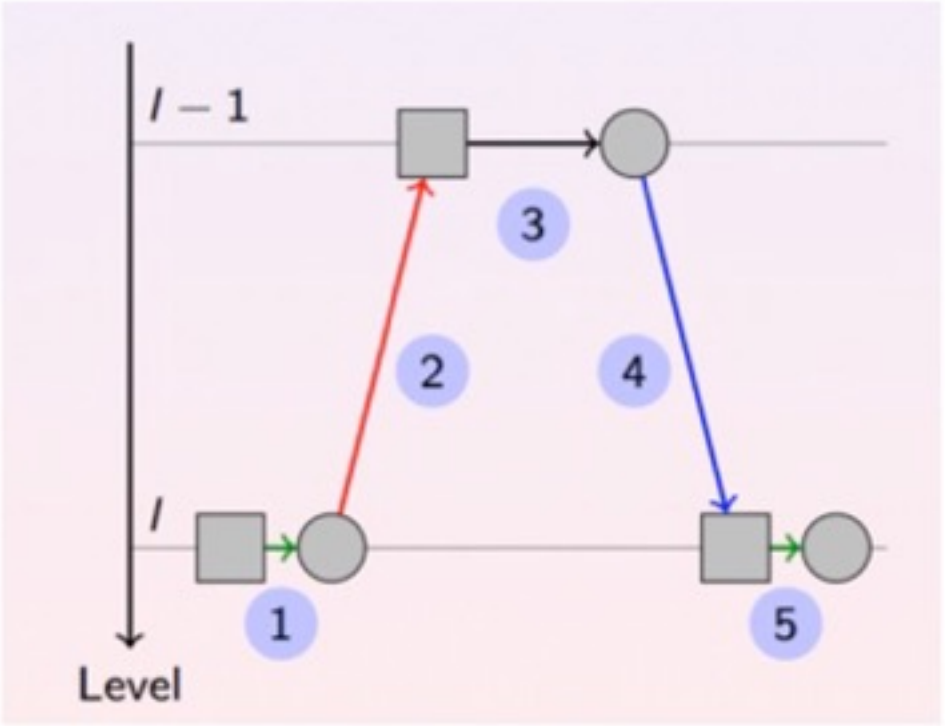
4- Prolong back the error to the fine grid:

$$e_{\ell-1}^\infty \longrightarrow e_\ell^{n+1}$$

5- Correct the fine grid solution:

$$\Phi_\ell^{n+1} = \Phi_\ell^n + e_\ell^{n+1}$$

and perform a few J/GS iteration.



Multigrid scheme

Recursively apply the 2-grid scheme. Solve for the exact solution only at the coarsest level.

Iterate one or twice before going to the finer level.

Common multigrid schedules

1 iteration per level
("V-cycle")

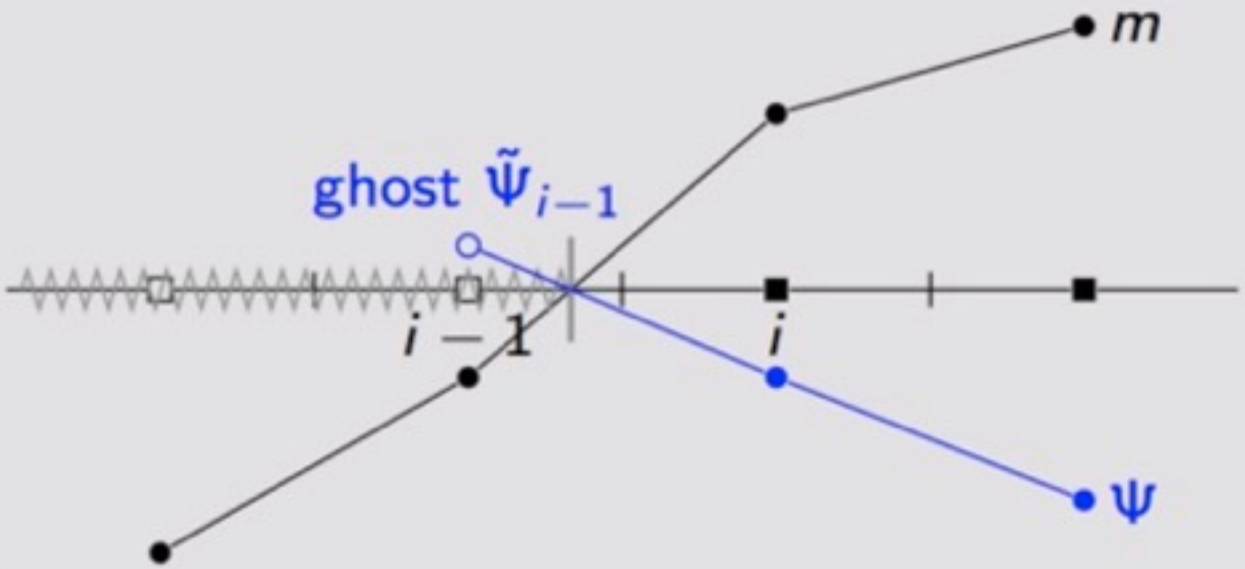
2 iterations per level
("W-cycle")

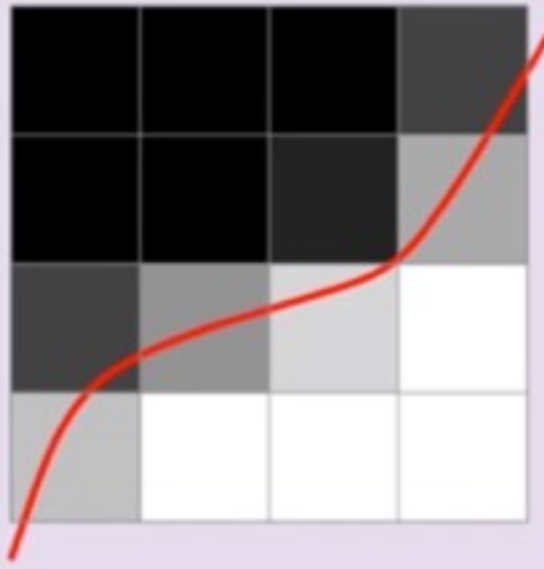
Converge in very few iterations, independently of grid size.

Quasi-insensitive to the quality of the initial guess.

Boundary-capturing technique

On each level, the boundary is defined as the zero-level set of a domain-fitted function (distance to the interface or volume fraction.



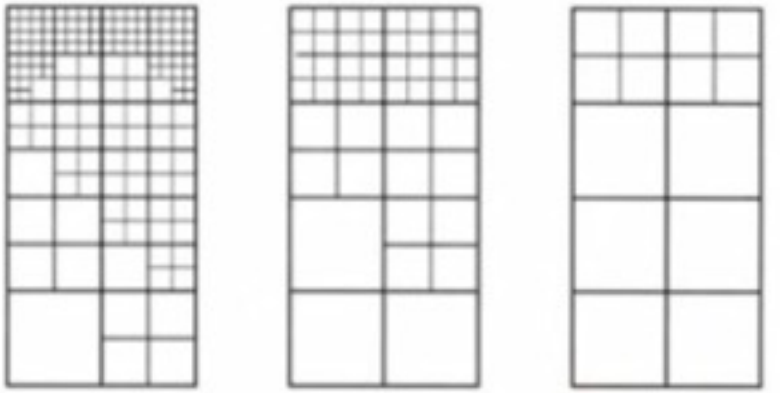


Boundary condition is enforced by linear extrapolation: second order boundary reconstruction as in Gibou (2002).

Multigrid for AMR

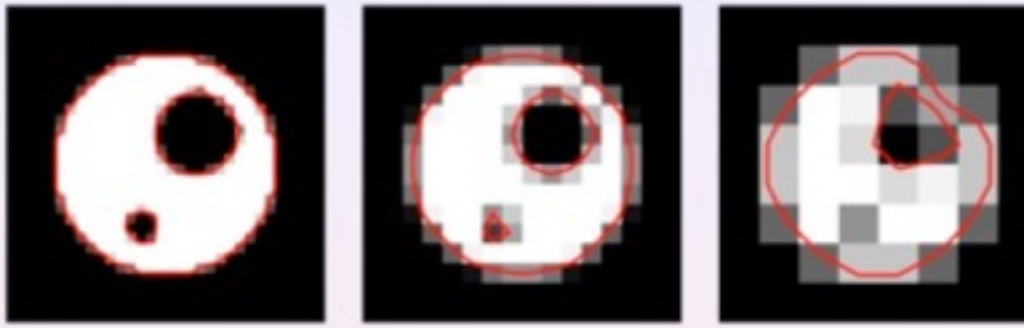
Two-way coupling: one needs to define a ensemble of AMR grids, each AMR grid corresponds to a level in the multigrid hierarchy.

Berger, M., "An adaptive multigrid method for the euler equations", Springer Berlin



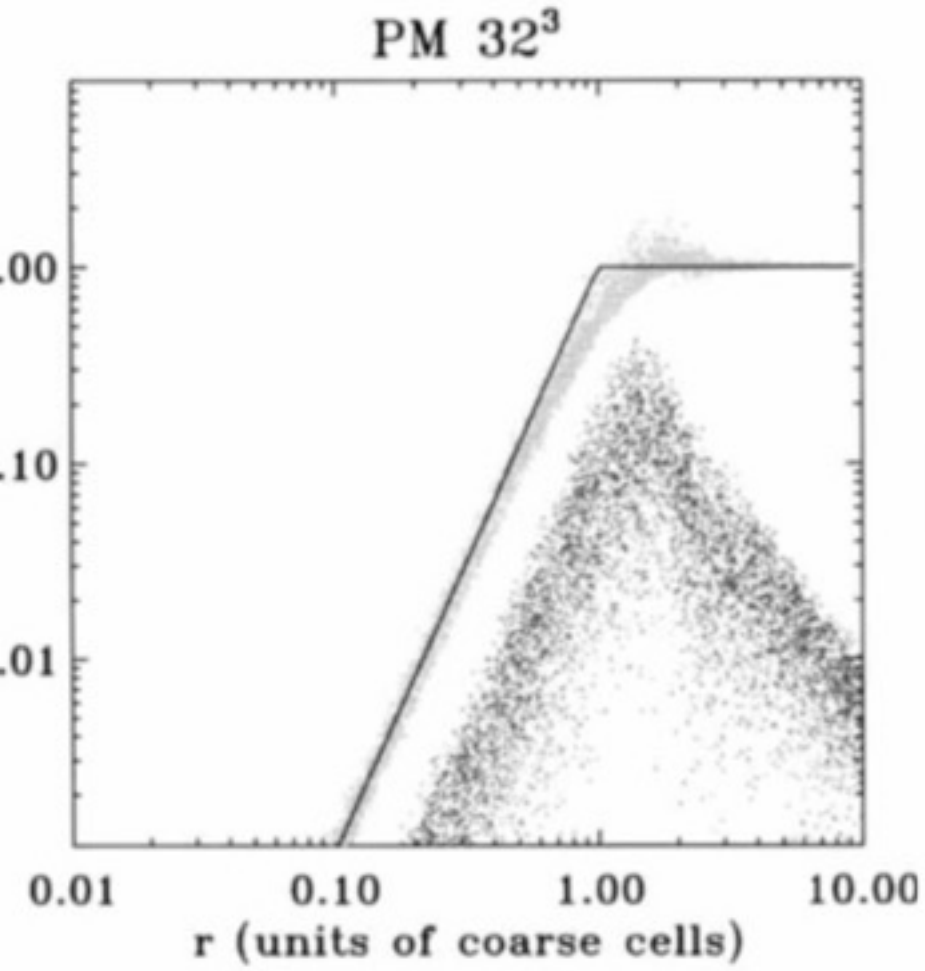
Level 3 Level 2 Level 1

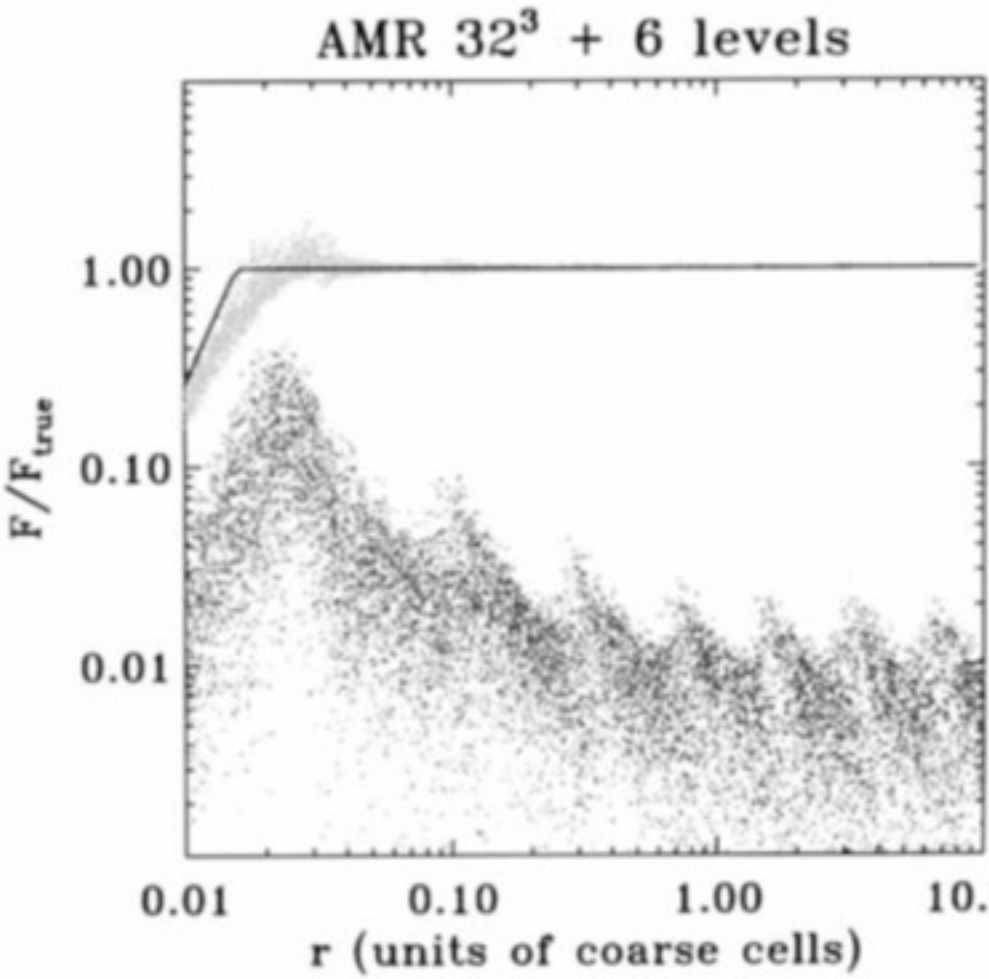
One-way coupling: for each AMR level, one needs to design a multigrid scheme for arbitrary-shaped boundary conditions.



Level 3 Level 2 Level 1

PM-AMR force accuracy





- Using force shaping and small scale PP corrections, the Particle Mesh method can reach very high resolution and accuracy. The AP3M method is the ultimate N-body solver for this family. Not well for arbitrary mesh geometry.

- Question: constant force softening versus adaptive force softening ?

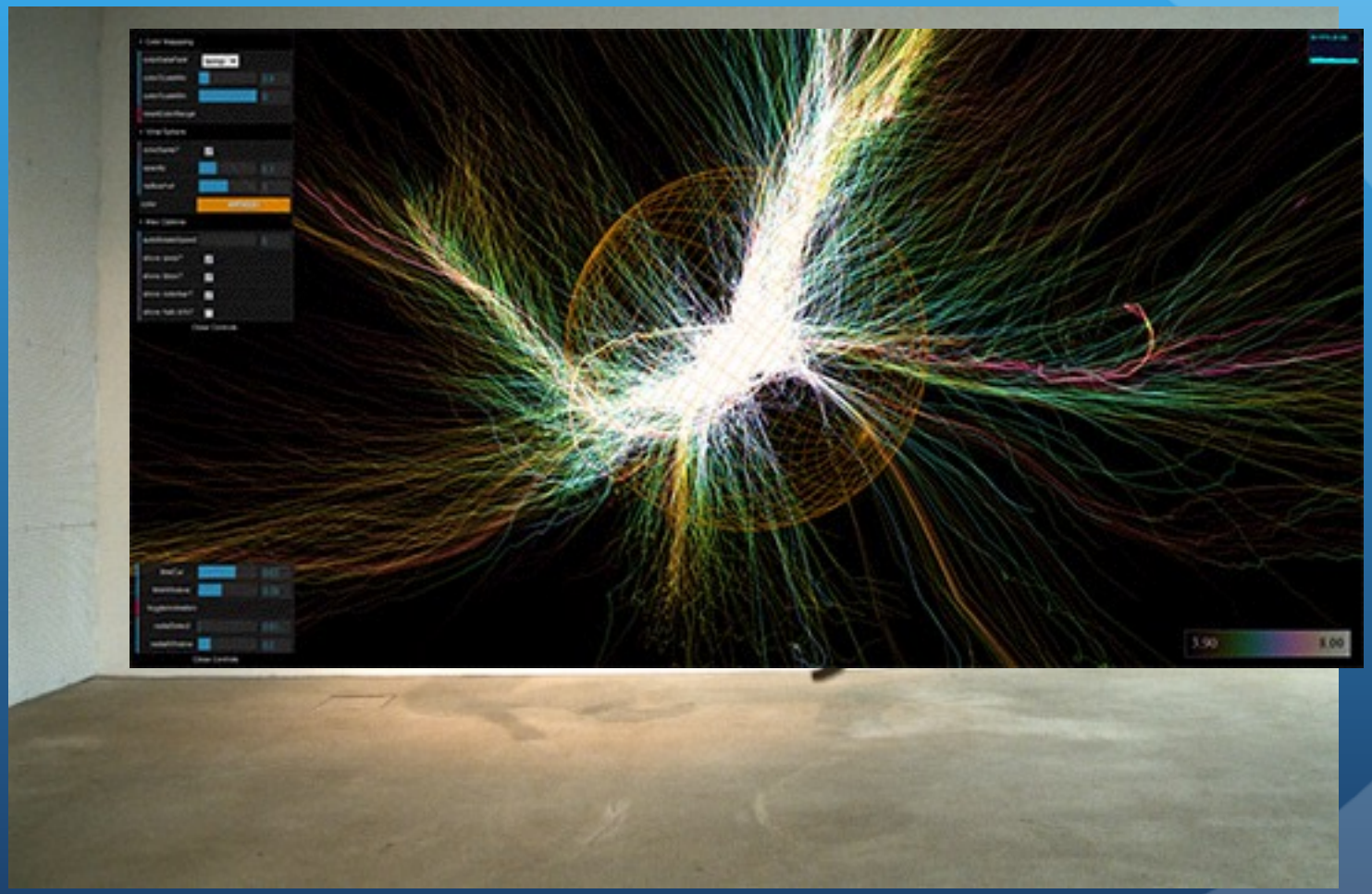
- PM-AMR has proven to be a powerful alternative.

- Two distinct approaches for AMR: one-way versus two-way coupling.

- PM-AMR reaches a similar accuracy than other codes (P3M or Tree), although force discontinuities can be an issue.

- PM-AMR is a very efficient method when used in conjunction with AMR-based multigrid solvers.

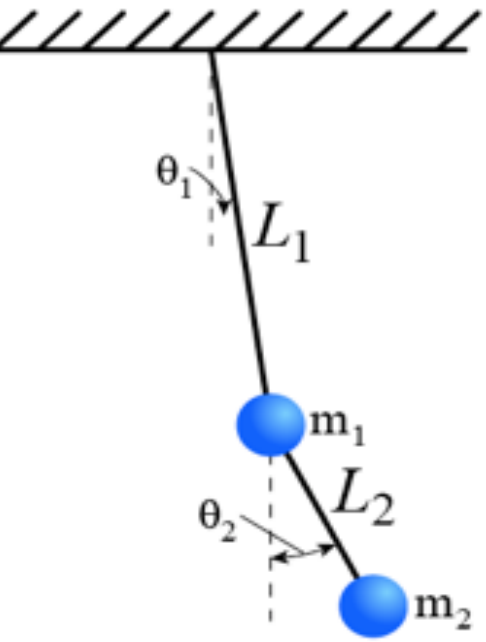
Move. Integration Time Step



Movimiento/Trayectorias de las partículas

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

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



Motion/ Particles Trejectories

$\ddot{X}$

- $\ddot{X} = F(x)/m = a(x)$ (Initial Condition problem)

$\dot{X}$

- $\dot{X} = V$

$\dot{V}$

- $\dot{V} = a(x)$

How to proceed?

- A classic problem in Ordinary Diferential Equations
- Several Methods

Time integration methods

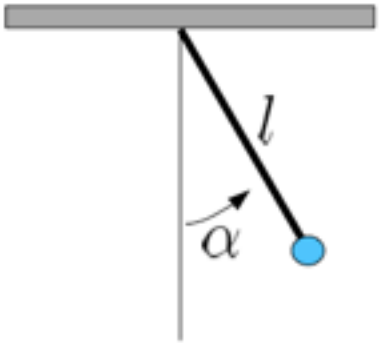
Want to numerically integrate an **ordinary differential equation** (ODE)

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

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}

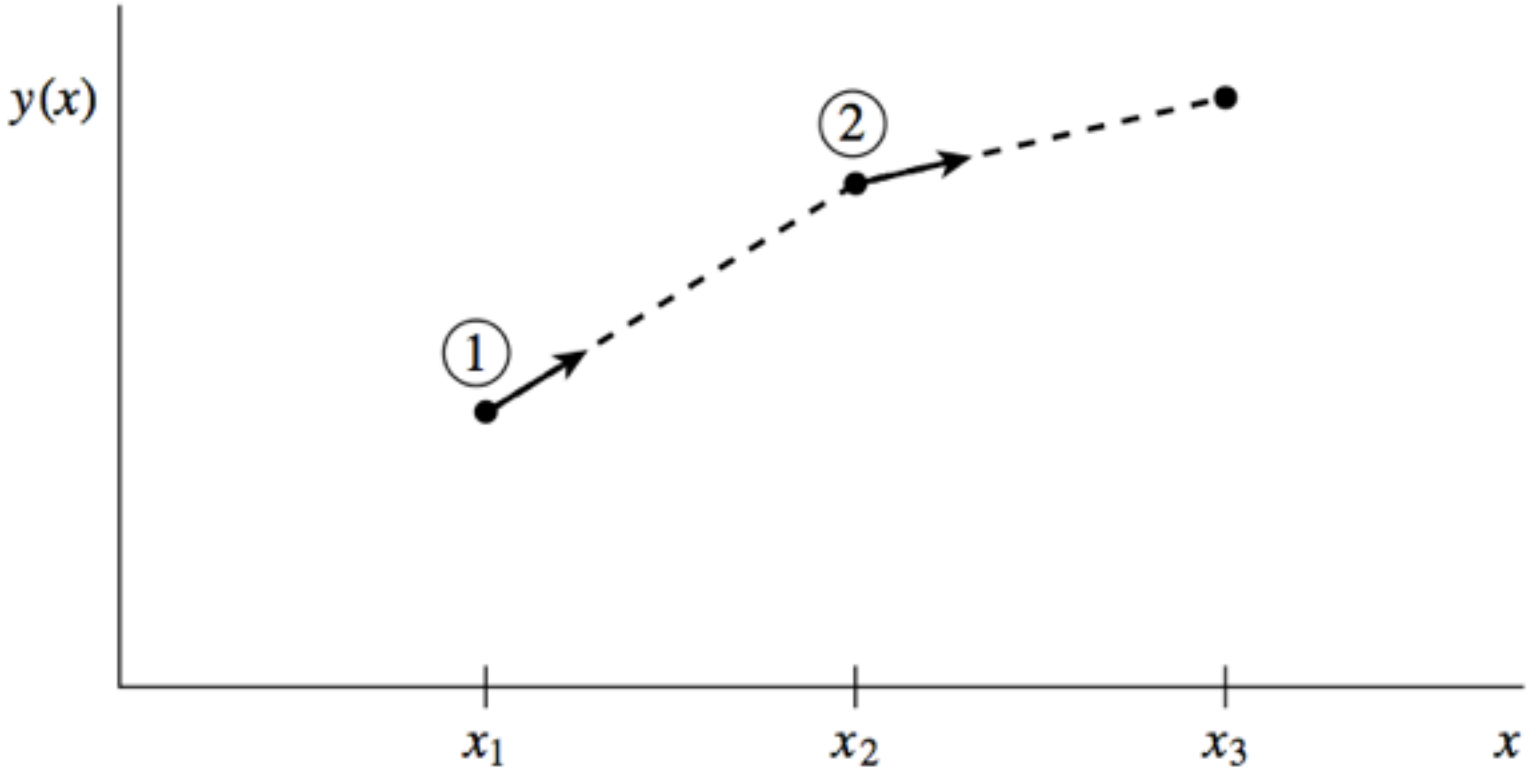


Figure 16.1.1. Euler's method. In this simplest (and least accurate) method for integrating an OD
he derivative at the starting point of each interval is extrapolated to find the next function value. T
nethod has first-order accuracy.

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

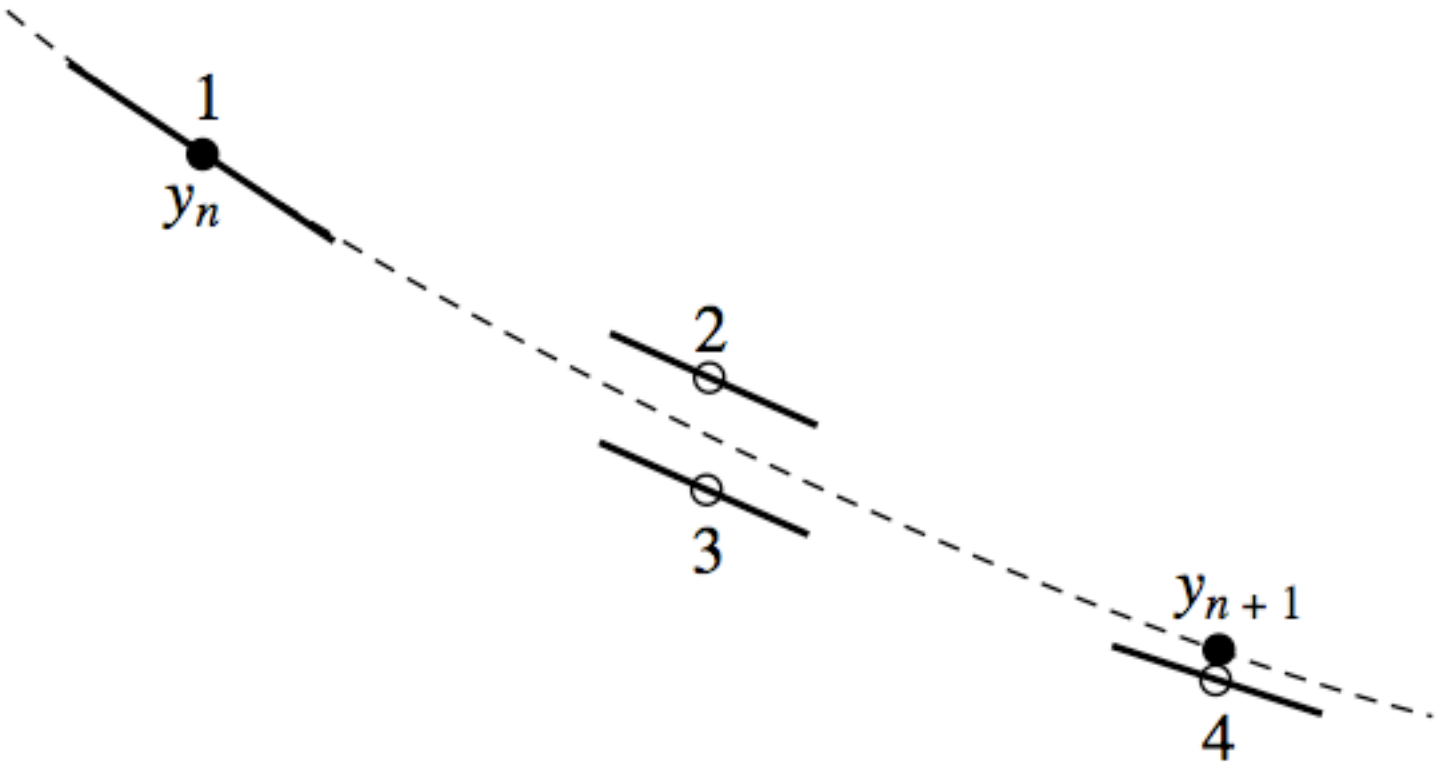
whole class of integration methods

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.

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



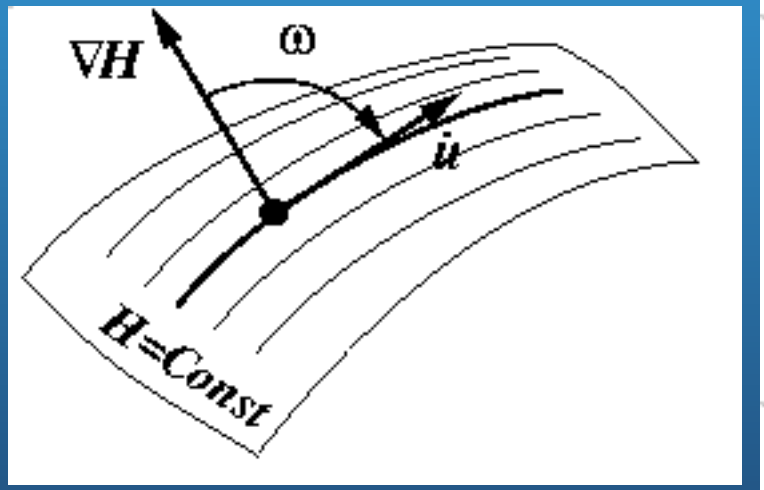
1.3. Fourth-order Runge-Kutta method. In each step the derivative is evaluate
e initial point, twice at trial midpoints, and once at a trial endpoint. From these de
tion value (shown as a filled dot) is calculated. (See text for details.)

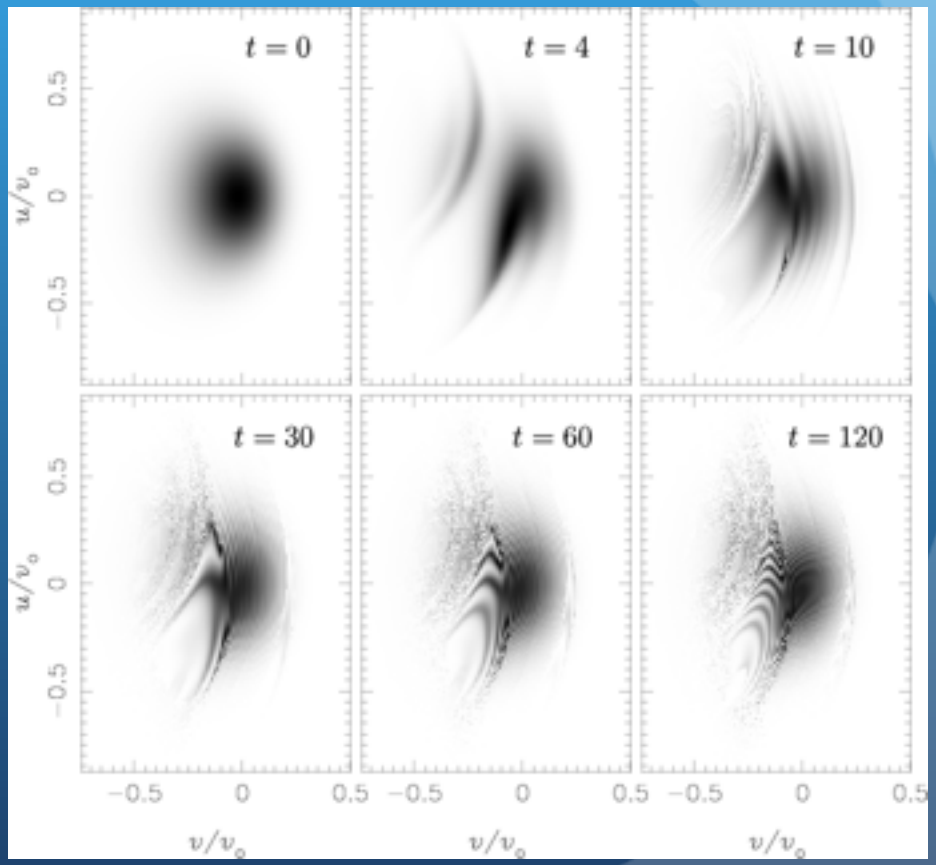
Which one shall we use?

- 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,

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

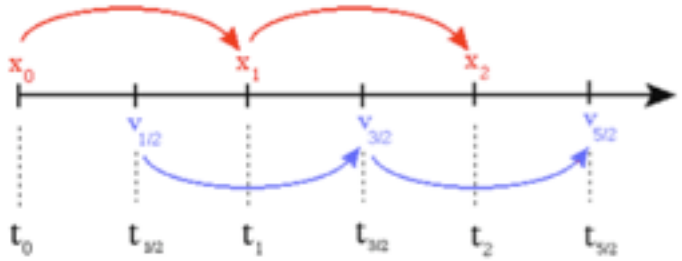
With this definition, we can write down a statement of the Leapfrog scheme that advances x_i to x_{i+1} and $v_{i+1/2}$ to $v_{i+3/2}$:

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

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

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

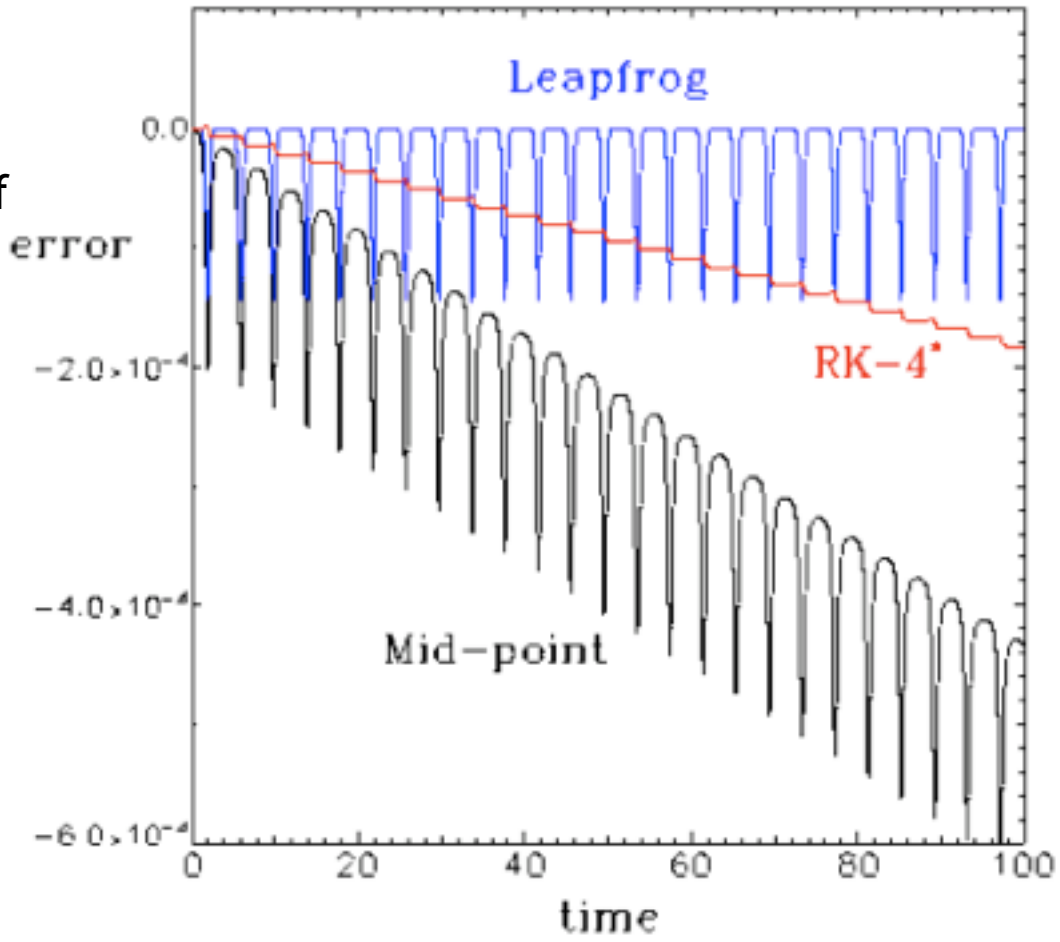
that this scheme does indeed give second-order accuracy in *x*. In fact, it is formally equivalent to the Mid-point or second-order predictor-corrector methods.



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

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

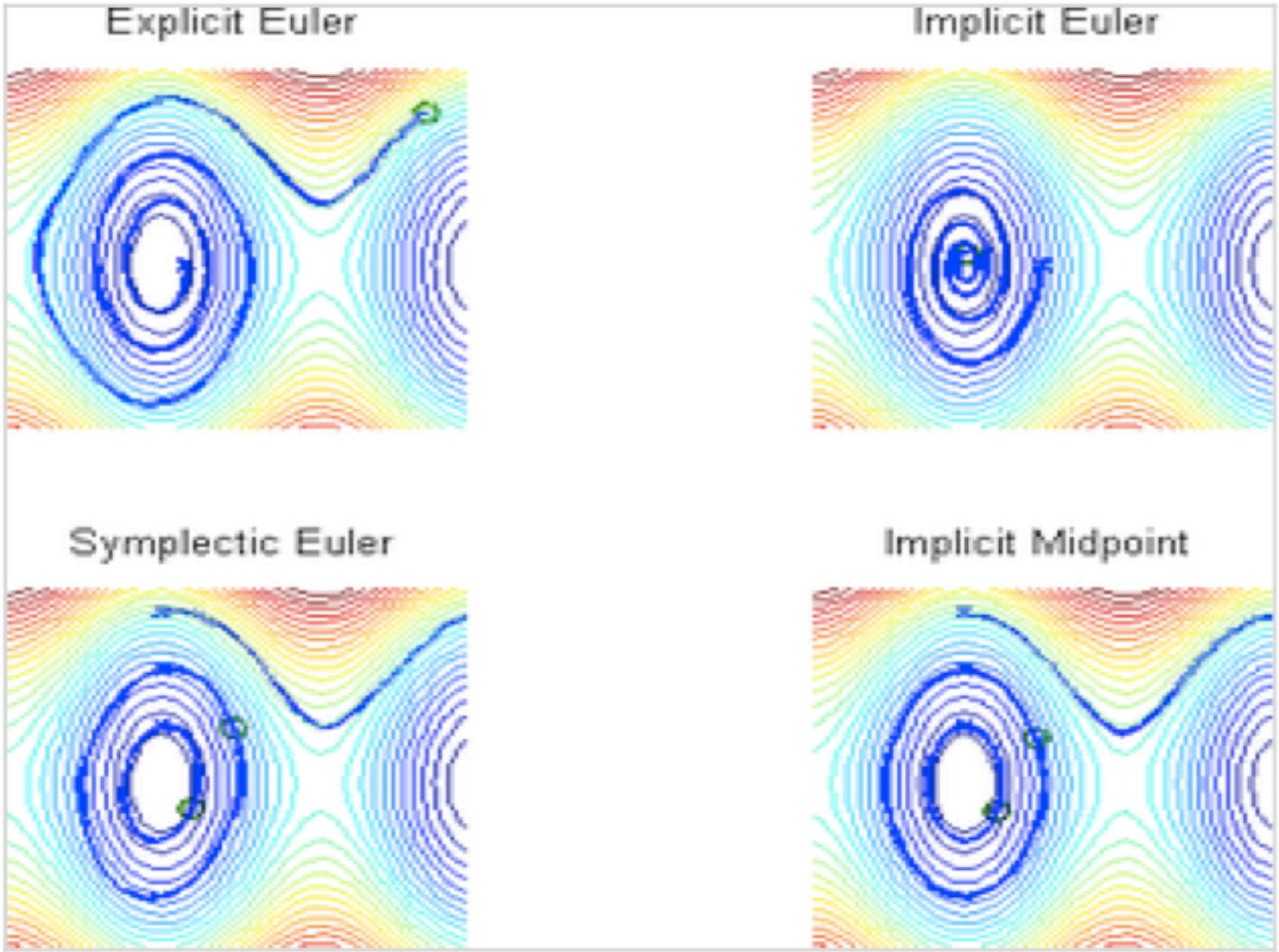
which gives Hamilton's equations

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

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.)

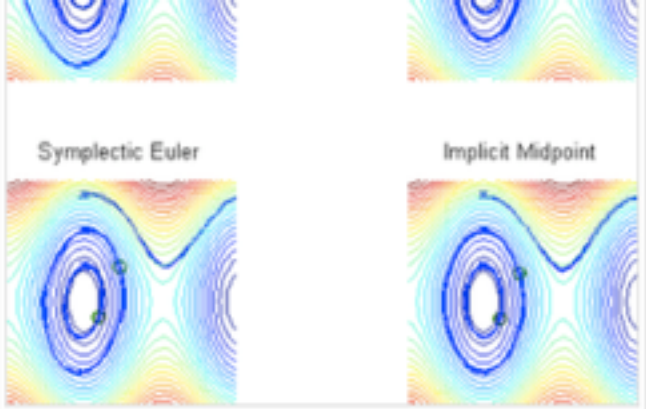


Simple pendulum: trajectories

Recall that the exact flow φ_t of a Hamiltonian system with one degree of freedom is area-preserving, in the sense that

$$\det \frac{\partial \phi_t}{\partial (q_0, p_0)} = 1 \text{ for all } t.$$

This formula is easily verified by hand. For our pendulum example we see that the numerical flow $\Phi_{\mathrm{eE},h} : z_k \mapsto z_{k+1}$ of the explicit Euler method is **not** area-preserving; viz.,



Simple pendulum: trajectories

$$\det \frac{\partial}{\partial (q_0, p_0)} \Phi_{\mathrm{eE},h}(z_0) = \begin{vmatrix} 1 & h \\ -h\cos q_0 & 1 \end{vmatrix} = 1 + h^2 \cos q_0 .$$

A similar calculation can be carried out for the implicit Euler method, where the determinant is

$$\det \frac{\partial}{\partial (q_0, p_0)} \Phi_{\mathrm{iE},h}(z_0) = (1 + h^2 \cos q_1)^{-1}.$$

However, the symplectic Euler method **is** area-preserving:

$$\begin{pmatrix} 1 & -h \\ 0 & 1 \end{pmatrix} \frac{\partial}{\partial (q_0, p_0)} \Phi_{\mathrm{sE},h}(z_0) = \begin{pmatrix} 1 & 0 \\ -h\cos q_0 & 1 \end{pmatrix},$$

thus $\det(\partial \Phi_{\mathrm{sE},h} / \partial (q_0, p_0)) = 1$. The implicit midpoint rule has similar geometric properties.

To summarize: the pendulum example shows that, besides the explicit and implicit Euler methods not being good choices of method to solve the problem, the symplectic Euler method and implicit midpoint rule agree well with the exact flow of the system, with the midpoint rule agreeing more closely. Furthermore, these latter two methods are area-preserving, just as the exact flow is; they are two examples of geometric (in fact, symplectic) integrators.

Previous discussion is accurate if all particles share the same time step

This is inefficient for large dynamical range simulations

Approximate solutions (Quinn et al)

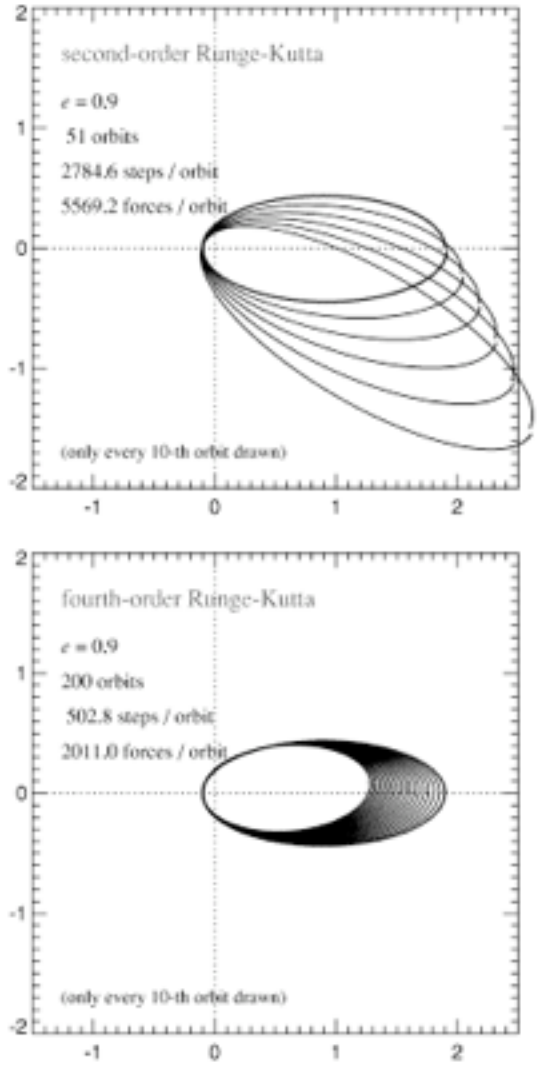
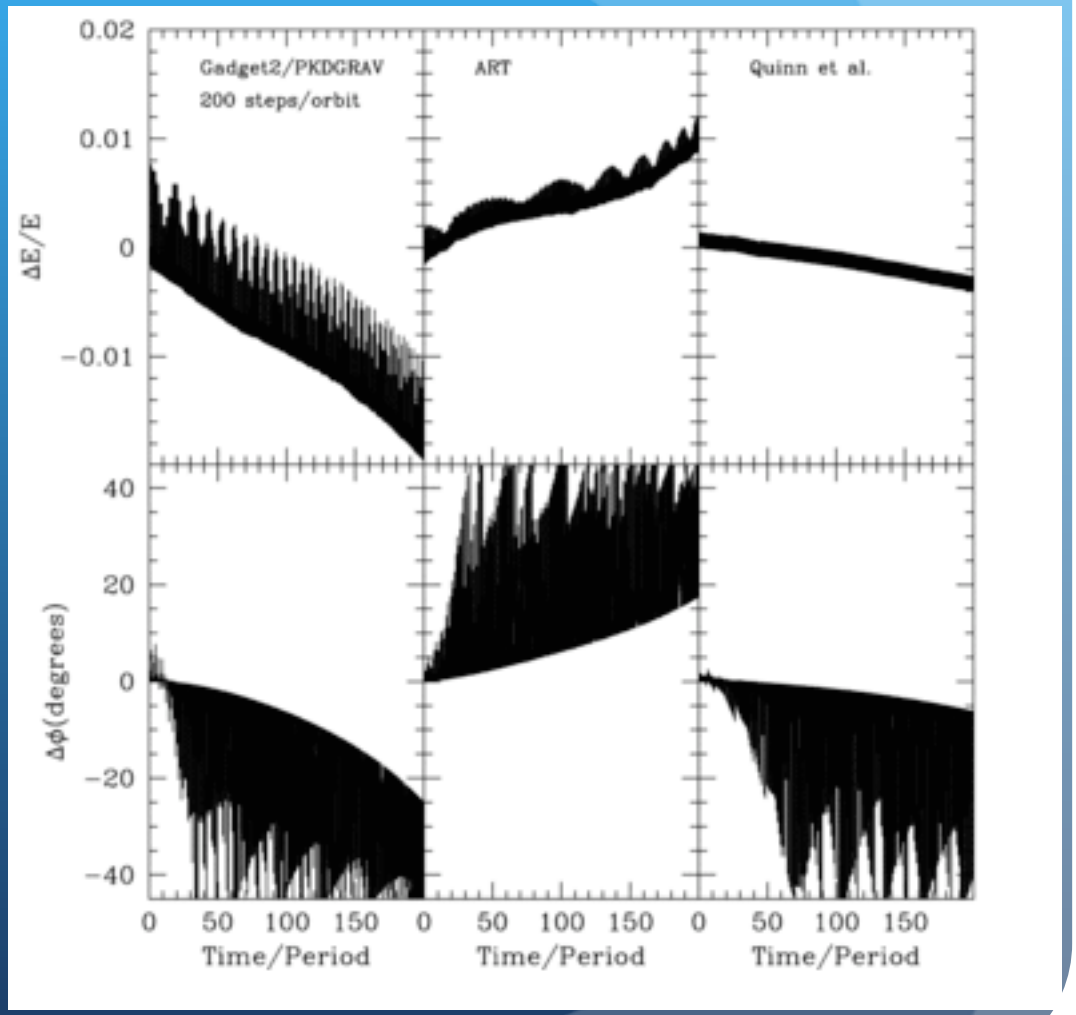


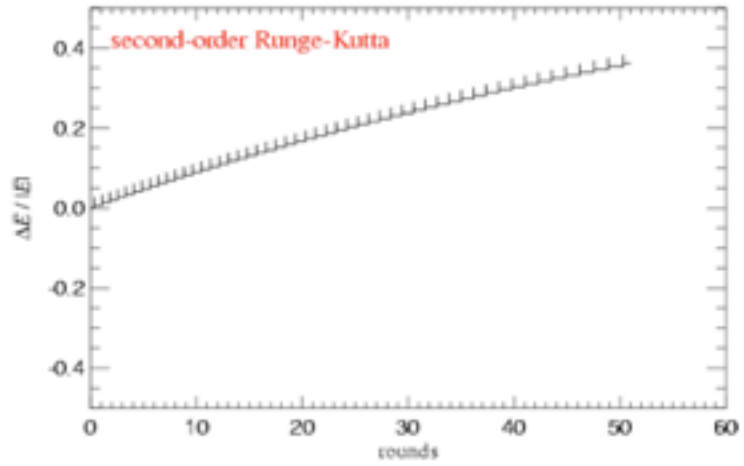
Figure 4. A Kepler problem of high eccentricity evolved with different simple time integration schemes, using an equal time-step in all cases. Even

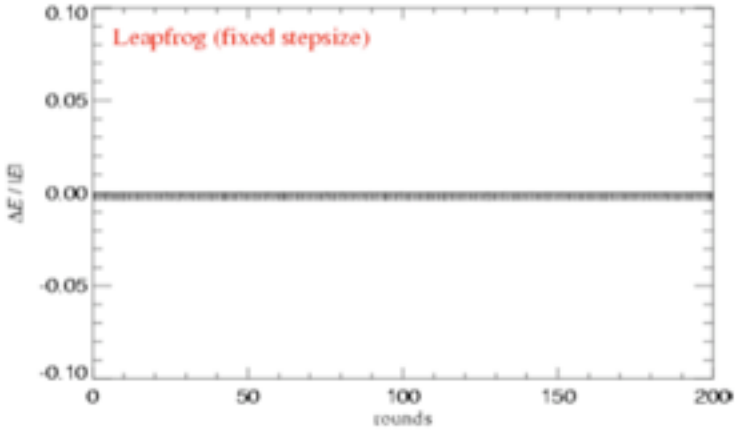
Klypin, Valenzuela, Colin, Quinn 08

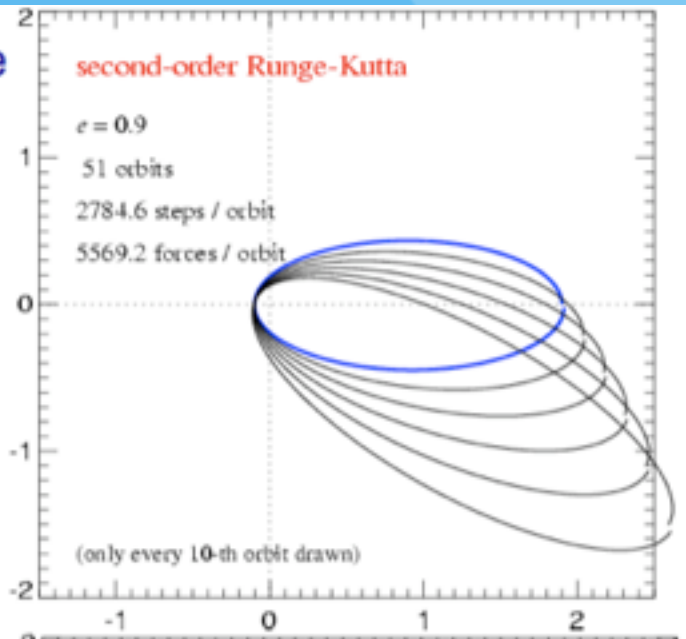


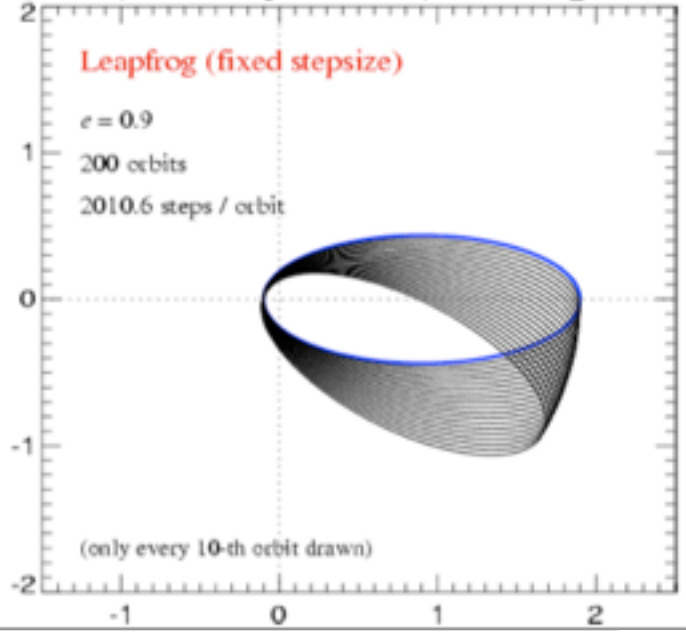
When compared with an integrator of the same order, the leapfrog is highly superior

INTEGRATING THE KEPLER PROBLEM



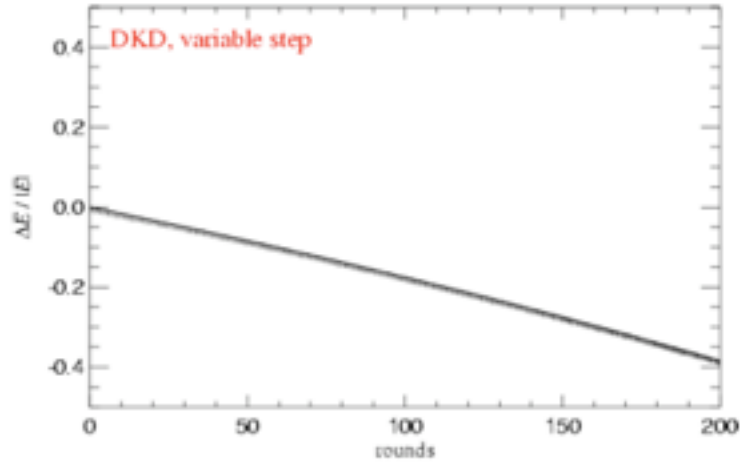


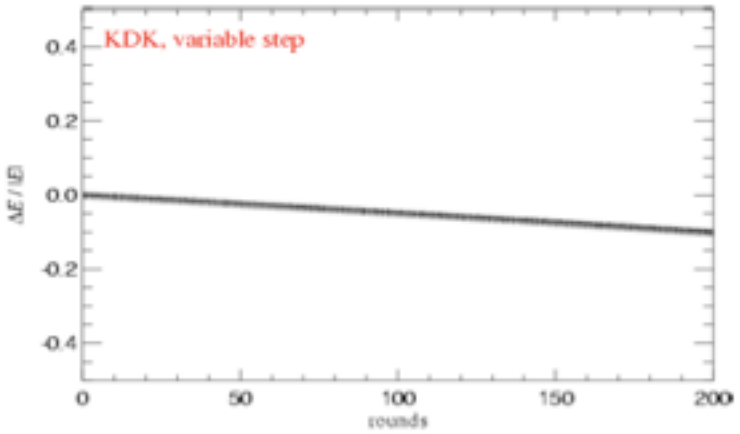




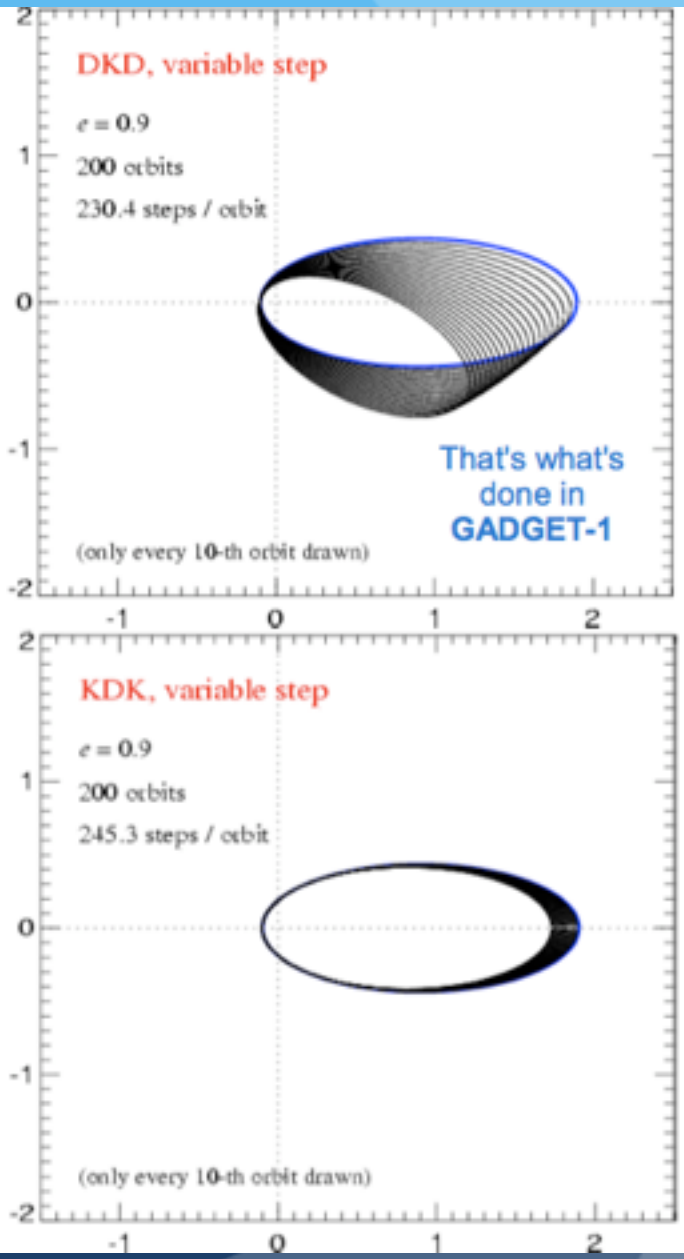
When an adaptive timestep is used, much of the symplectic advantage is lost

INTEGRATING THE KEPLER PROBLEM





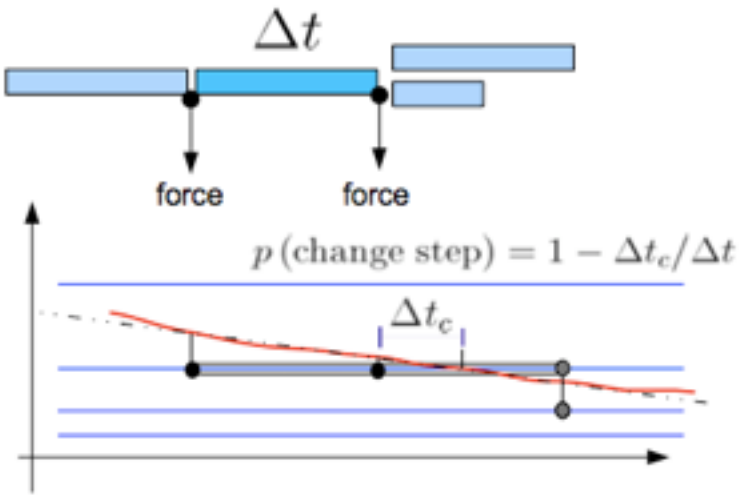
→ Going to KDK reduces the error by a factor 4, at the same cost !

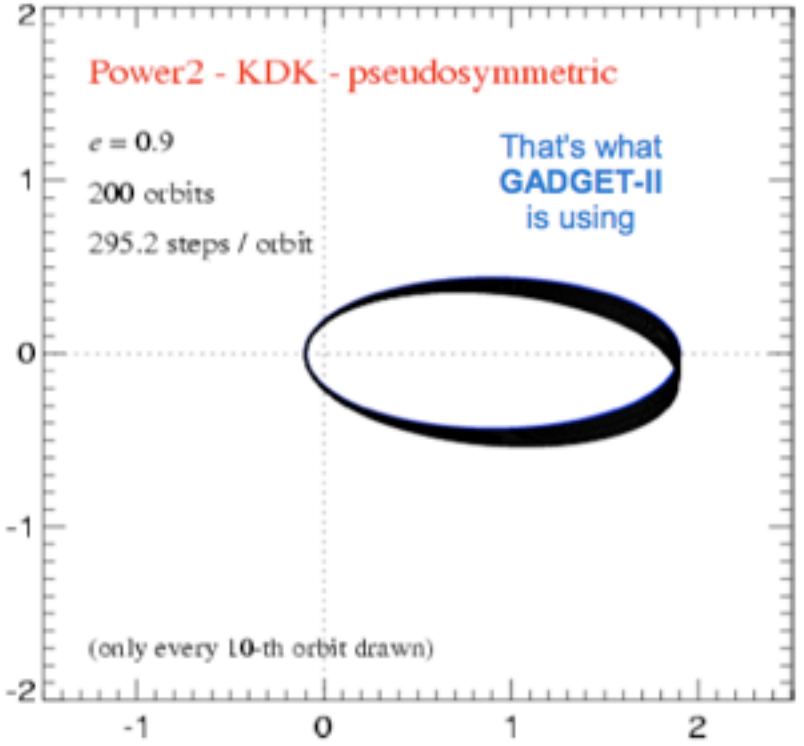


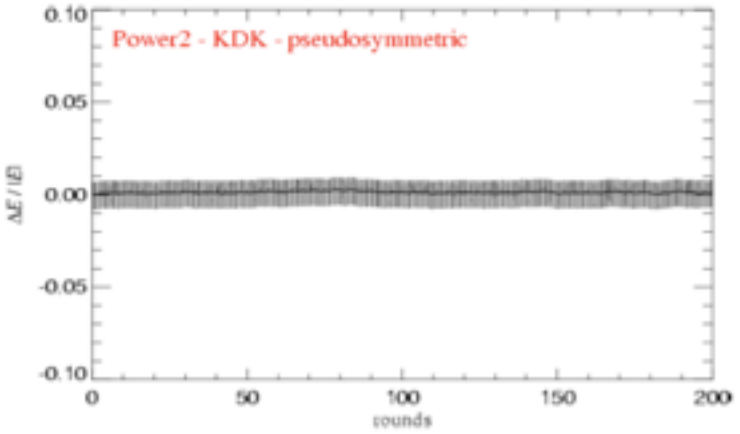
Pseudo-symmetric behaviour can be obtained by making the evolution of the expectation value of the numerical Hamiltonian time reversible

INTEGRATING THE KEPLER PROBLEM

KDK scheme



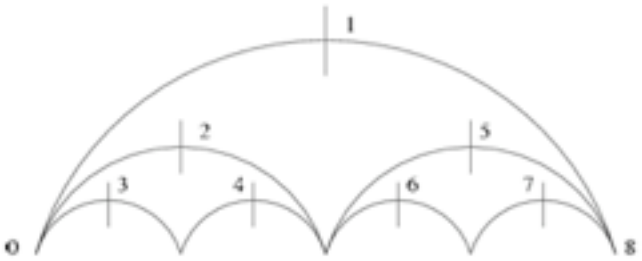




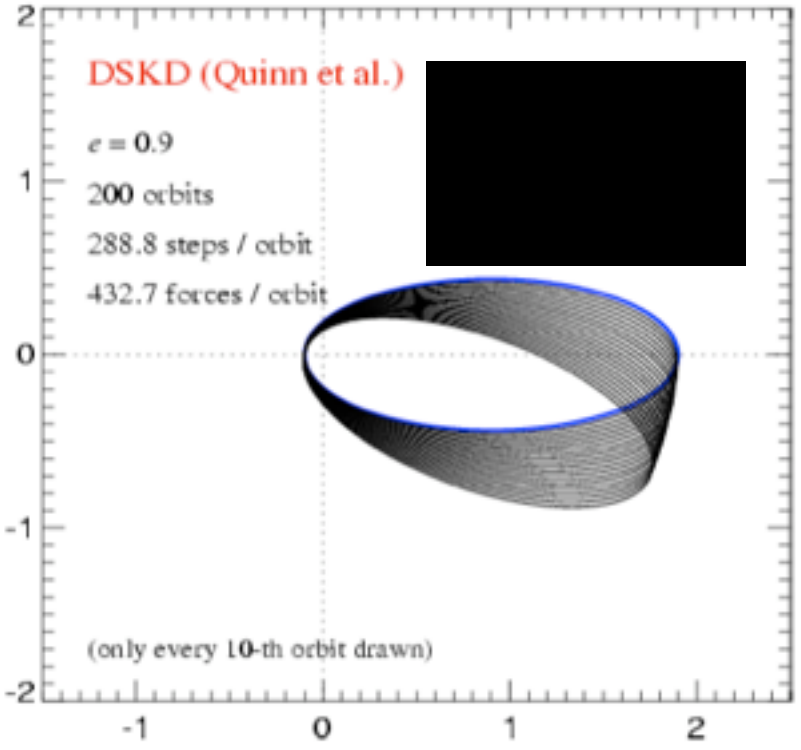
Gives the best result at a given number of force evaluations.

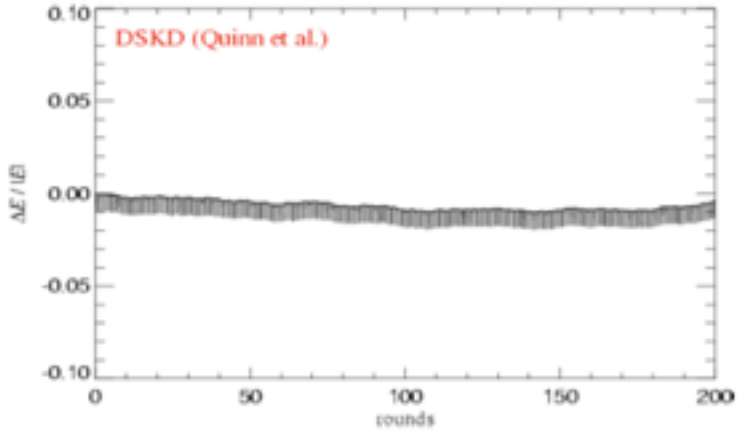
Symmetric behaviour can be obtained by using an implicit timestep criterion that depends on the end of the timestep

INTEGRATING THE KEPLER PROBLEM



Quinn et al. (1997)





- Force evaluations have to be thrown away in this scheme
- reversibility is only approximatively given
- Requires back-wards drift of system - difficult to combine with SPH

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?

A better leap frog

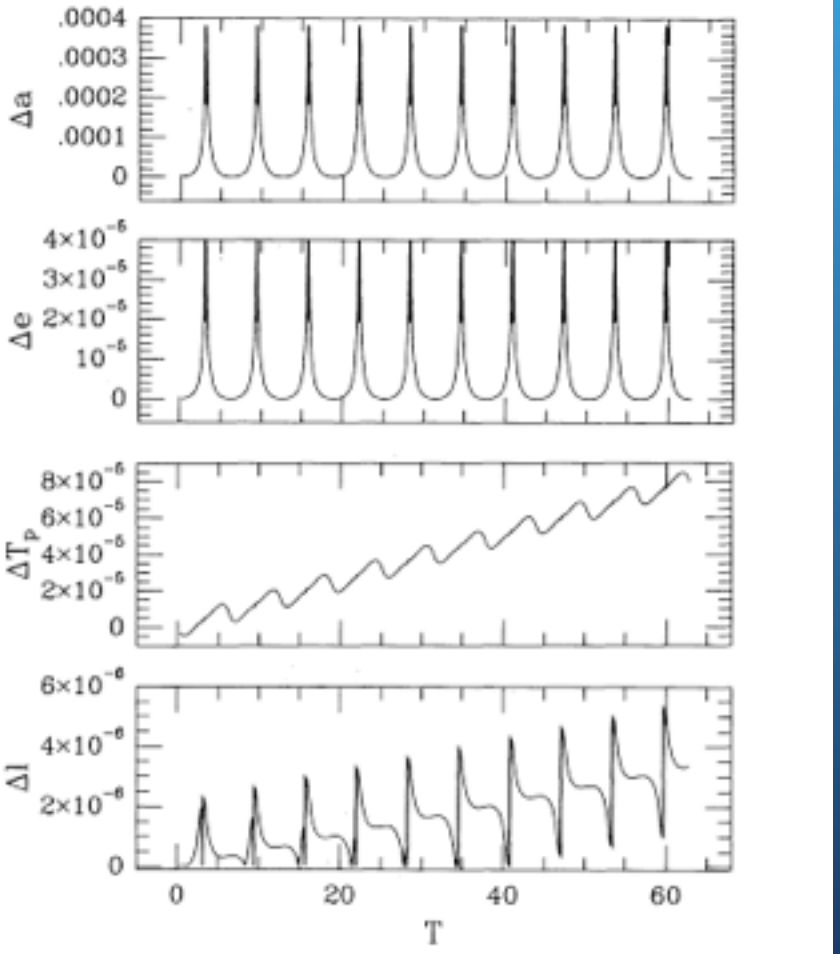


FIG. 1.—Variable-time step time-symmetric leapfrog integration (eq. [2])

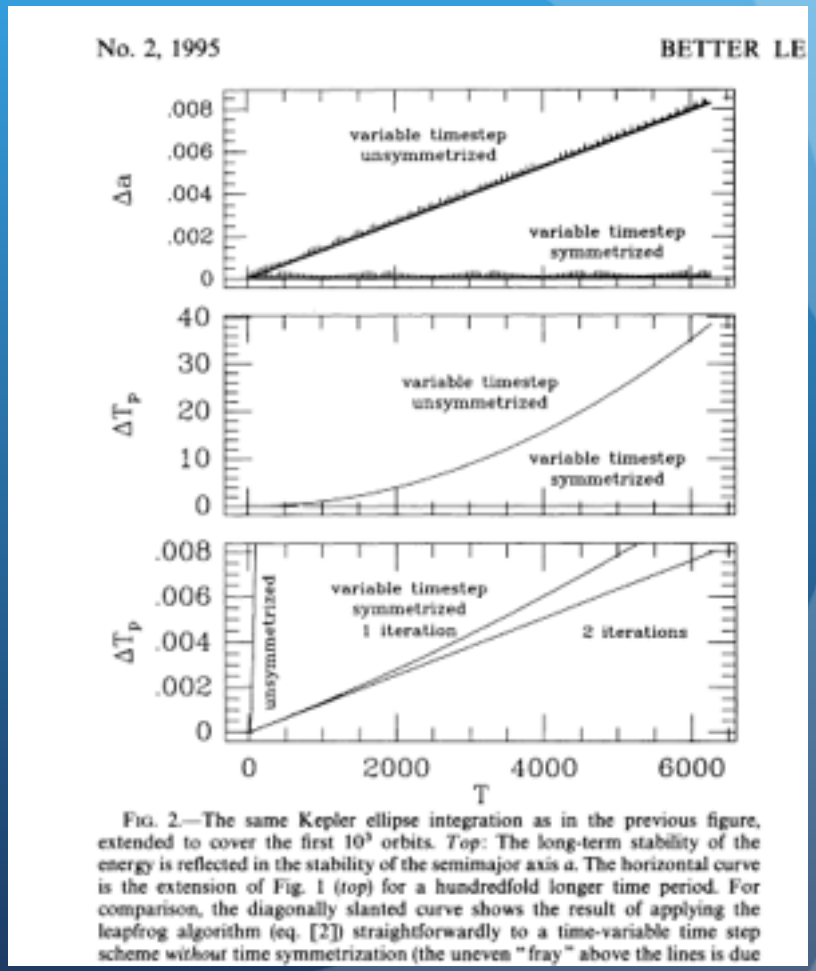


FIG. 2.—The same Kepler ellipse integration as in the previous figure, extended to cover the first 10^5 orbits. *Top*: The long-term stability of the energy is reflected in the stability of the semimajor axis *a*. The horizontal curve is the extension of Fig. 1 (*top*) for a hundredfold longer time period. For comparison, the diagonally slanted curve shows the result of applying the leapfrog algorithm (eq. [2]) straightforwardly to a time-variable time step scheme *without* time symmetrization (the uneven "fray" above the lines is due

N-body Integrators with Individual Time Steps from Hierarchical Splitting

Federico I. Pelupessy[a], Jürgen Jänes[b,c], Simon Portegies Zwart[a]

[a] *Leiden Observatory, Leiden University, PO Box 9513, 2300 RA, Leiden, The Netherlands*

[b] *Faculty of Science, University of Amsterdam, PO Box 94216, 1090GE, Amsterdam, The Netherlands*

[c] *Cambridge Computational Biology Institute, Department of Applied Mathematics and Theoretical Physics, Centre for Mathematical Sciences, Wilberforce Road, Cambridge CB3 0WA, United Kingdom*

Abstract

We review the implementation of individual particle time-stepping for N-body dynamics. We present a class of integrators derived from second order Hamiltonian splitting. In contrast to the usual implementation of individual time-stepping, these integrators are momentum conserving and show excellent energy conservation in conjunction with a symmetrized time step criterion. We use an explicit but approximate formula for the time symmetrization that is compatible with the use of individual time steps. No iterative scheme is necessary. We implement these ideas in the HUAYNO[1] code and present tests of the integrators and show that the presented integration schemes shows good energy conservation, with little or no systematic drift, while conserving momentum and angular momentum to machine precision for long term integrations.

Keywords: Stellar dynamics; Methods: numerical, N-body

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

Anyone?

The Vlassov-Poisson equation

Collisionless limit of the Boltzmann equation:

$$\frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial}{\partial t} f(\mathbf{x}, \mathbf{p}, t) + \frac{\mathbf{p}}{ma^2} \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{p}, t) - m \nabla_{\mathbf{x}} . \Phi(\mathbf{x}) \frac{\partial}{\partial \mathbf{p}} f(\mathbf{x}, \mathbf{p}, t) = 0$$

Liouville theorem: number of particles in conserved in phase-space

Gravitational acceleration is given by the Poisson equation

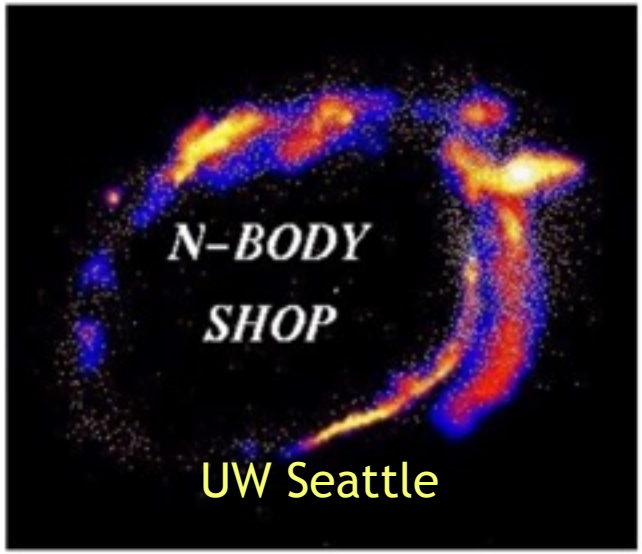
$$\Delta \Phi(\mathbf{x}) = \frac{4\pi G m}{a} \left(\int f(\mathbf{x}, \mathbf{p}, t) \mathrm{d}^3 \mathbf{p} - \overline{n} \right),$$

3 solution strategies:

Boltzmann-Poisson Solver???

- pure fluid on a 6D grid
- pure N body using direct or Tree force computations
- mixture of the 2: the Particle-Mesh method

Life beyond MPI? CHRAM++



Thomas Quinn
Graeme Lufkin
Joachim Stadel
James Wadsley

PPL PARALLEL PROGRAMMING LAB
DEPT OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS

Laxmikant Kale
Filippo Gioachin
Pritish Jetley
Celso Mendes
Amit Sharma
Lukasz Wesolowski
Edgar Solomonik

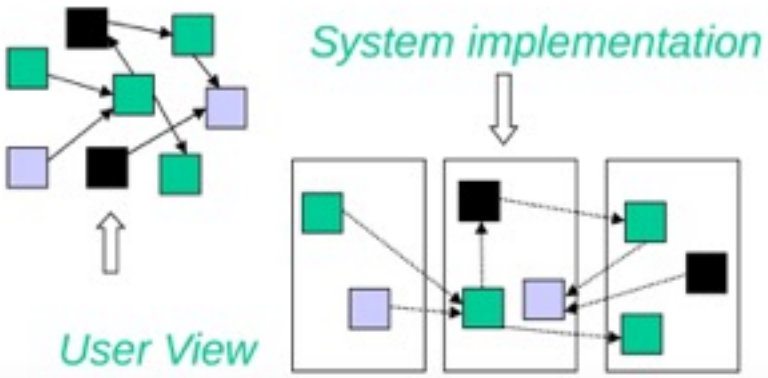
Beyond MPI

Charm++: Migratable Objects

Programmer: [Over] decomposition into virtual processors

Runtime: Assigns VPs to processors

Enables *adaptive runtime strategies*



Benefits

- Software engineering
  - Number of virtual processors can be independently controlled
  - Separate VPs for different modules
- Message driven execution
  - Adaptive overlap of communication
- Dynamic mapping
  - Heterogeneous clusters
    - Vacate, adjust to speed, share
  - Automatic checkpointing
  - Change set of processors used
  - Automatic dynamic load balancing
  - Communication optimization

All for now

- 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
- 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??


512g RAM
64 cores
AMD
twin node
2x128G RAM
XEON
48 cores
96 threads
Conectado
por fibra optica
40gbts

1.**DESI -México (ICN, IA, IF,CINVESTAV, UG, ININ)**

224 cores-320 threads

-(2) Nodes AMD+ 64 Cores 1U **512GB** RAM 1866 Mhz (**128 Cores**)

-(2) Nodes 1U Twin, Xeon E5-2680v3 2.5 GHz,

128GB RAM DDR4 2133 Mhz, c/u

512 GB RAM !! (**96 Cores (192 Threads)** 9.6 GT/s

- Storage (Extension to an existing Lustre system at ICN, dedicated to DESI
72 TB (12 x 6TB) at 12 Gb/s. Able to grow. Connection to ICN 1Gps
+

Hybrid system:
328 +12(master) = 360 cores
6 GPU cards (3GB RAM GDDR5, 448 CUDA cores 1.15 GHz **)**
Storage 30Tbytes at Raid 6, growing to ~200 Tbytes.

Total= 584 physical cores (700 threads, mostly to data intensive work). May work as 220 cores + Lustre system dedicated to DESI or combined with the existing cluster

Sharing Infiniband switch at 40Gbts