

Numerical Simulations of Gravitational Lensing

Jan Hartlap

Argelander-Institut für Astronomie
Universität Bonn

DUEL Summer School
Paris, August 2009



Argelander-
Institut
für
Astronomie

Overview

- ▶ *N*-body simulations
 - ▶ the particle-mesh code;
basic techniques
 - ▶ improvements
 - ▶ setting up a simulation
 - ▶ single-halo simulations
- ▶ Ray-tracing simulations for strong lensing
- ▶ Ray-tracing simulations for weak lensing

Basic equations

Evolution of the phase space distribution function $f(\mathbf{r}, \mathbf{u}, t)$:

$$\text{Vlasov eqn. : } \frac{df}{dt} = \frac{\partial f}{\partial t} + \mathbf{u} \frac{\partial f}{\partial \mathbf{r}} - \nabla \Phi \frac{\partial f}{\partial \mathbf{u}} = 0 ,$$

$$\text{Poisson eqn. : } \nabla^2 \Phi = 4\pi G \rho(\mathbf{r}, t) ,$$

$$\text{where } \rho(\mathbf{r}, t) = m_{\text{DM}} \int d^3u f(\mathbf{r}, \mathbf{u}, t) .$$

Vlasov eqn. is 6-dimensional!

How to solve this?

On a grid?

- ▶ Memory consumption $\propto N_g^6$
→ need 4 GB RAM for $N_g = 32$,
256 GB for $N_g = 64$
→ impossible!
- ▶ Resolution $\approx L_{\text{box}}/N_g$
→ far too low

The Monte-Carlo approach

Idea:

- ▶ Divide phase space into N_p cells
- ▶ Assign the mass in each cell to a particle
- ▶ Each of these particles obeys the characteristic eqns. / eqns. of motion:

$$\frac{dr}{dt} = \mathbf{u}, \quad \frac{du}{dt} = -\nabla\Phi$$

(set of charact. eqns. equivalent to Vlasov eqn.)

- ▶ Estimate f from the density of particles in phase space

Bonus:

Resolution automatically adapts to density!

The algorithm

First: Setup initial density field (periodic boundary conditions), derive velocity field from this.

Then: evolve system over many small timesteps

For each step:

- ▶ obtain a density estimate from the particle distribution
- ▶ compute the gravitational potential
- ▶ compute force on each particle
- ▶ advance particles, update velocities
- ▶ produce some useful output

Initial conditions: Gaussian random fields

Gaussian random fields

- ▶ Fourier modes are statistically independent
- ▶ PDF of each Fourier mode is Gaussian with variance given by power spectrum:

$$\langle \tilde{\delta}(\mathbf{k}) \tilde{\delta}(\mathbf{k}') \rangle = \left(\frac{2\pi}{\Delta k} \right)^3 \delta_{\mathbf{k}, \mathbf{k}'} P(|\mathbf{k}|), \quad \text{with } \Delta k = \frac{2\pi}{L_{\text{box}}}$$

Implementation

- ▶ for each Fourier mode (pixel), draw two Gaussian random numbers:

$$\text{Re } \tilde{\delta}(\mathbf{k}), \text{ Im } \tilde{\delta}(\mathbf{k}) \sim \mathcal{N} \left[0, L_{\text{box}}^3 P_{\text{lin}}(k, t_i) / 2 \right]$$

- ▶ transform to real space (use Fast Fourier Transform FFT)

Initial conditions: Zeldovich approximation (I)

- ▶ initial density field at $t = t_i$ is Gaussian
- ▶ can use linear theory to perturb uniform particle distribution

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

$$\mathbf{v} = \dot{D}_+(t_i) \mathbf{S}(\mathbf{q}) ,$$

\mathbf{x} , \mathbf{v} : comoving pos./vel.;

D_+ : lin. growth factor

\mathbf{S} : displacement field

- ▶ \mathbf{q} is a fixed label for each particle, here: initial coordinates
- ▶ information about evolution of density field in mapping $\mathbf{x}(\mathbf{q})$

$$\tilde{\mathbf{S}}(\mathbf{k}) = \frac{i\mathbf{k} \tilde{\delta}(\mathbf{k})}{D_+(t_i) |\mathbf{k}|^2}$$

Initial conditions: Zeldovich approximation (II)

- ▶ Choose initial scale factor a_i , timestep Δa :
usually $z_i \approx 50 \dots 100$
all modes of interest linear!
- ▶ Start particles on a uniform distribution: $\rightarrow \mathbf{q}$
 - ▶ uniformly random (shot noise)
 - ▶ grid (might cause artifacts)
 - ▶ glass (start with uniformly random distr., then evolve under “anti-gravity”)
- ▶ Obtain realization of $\tilde{\delta}(\mathbf{k}, a_i)$ on a grid
- ▶ Compute $\tilde{\mathbf{S}}_{\mathbf{k}} = i\mathbf{k} \delta(\mathbf{k}) / D_+(t_i) |\mathbf{k}|^2$ for $\mathbf{k} \neq 0$
- ▶ Displace particles, assign initial velocities

Basic steps of a PM code

Main loop of a Particle Mesh code:

- ▶ Assign particles to grid ($\mathcal{O}[N_p]$)
- ▶ Use FFT to compute gravitational potential ($\mathcal{O}[N_g^3 \log(N_g^3)]$)
→ periodic boundary conditions!
- ▶ Use finite differencing (numerical derivatives) to get forces on particles ($\mathcal{O}[N_p]$)
- ▶ Advance particles, e.g. using leapfrog scheme ($\mathcal{O}[N_p]$)

Mass assignment (I)

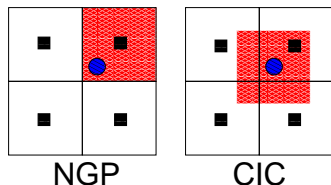
- ▶ Give particles a shape $S(x)$
- ▶ To mesh cell at \hat{x}_m , assign fraction of mass in this cell (1D example):

$$\rho(\hat{x}_m) = \frac{m_p}{\Delta} \sum_{i=1}^{N_p} W(x_i - \hat{x}_m),$$

where

$$W(x_i - \hat{x}_m) = \int_{\hat{x}_m - \Delta/2}^{\hat{x}_m + \Delta/2} dx S(x - x_i)$$

Mass assignment (II): assignment schemes



NGP scheme	$S(\mathbf{x}) \propto \delta_D(\mathbf{x} - \mathbf{x}_i)$	(1 mesh point)
CIC scheme	$S(\mathbf{x}) \propto \Pi(\mathbf{x} - \mathbf{x}_i)$	(8 mesh points)
TSC scheme	$S(\mathbf{x}) \propto \Lambda(\mathbf{x} - \mathbf{x}_i)$	(27 mesh points)
...

Higher order schemes yield smoother density and force but become very costly!

Common choice: CIC, sometimes TSC

The Poisson solver

Poisson-Eqn.:

$$\nabla_{\mathbf{x}}^2 \phi = \frac{3H_0^2 \Omega_m}{2a} \delta(\mathbf{x})$$

- ▶ transform δ to Fourier space (FFT)
- ▶ Get FT of potential using

$$\tilde{\phi}(\mathbf{k}) = \frac{3H_0^2 \Omega_m}{2a} \frac{1}{|\mathbf{k}|^2} \tilde{\delta}(\mathbf{k})$$

- ▶ transform back to real space
→ potential $\phi(i, j, h)$ on a grid

Particle accelerations

- ▶ perform finite differencing to get accelerations:

$$g_1(i, j, h) = -\nabla_1\phi \approx -\frac{\phi(i+1, j, h) - \phi(i-1, j, h)}{2\Delta}$$

$$g_2(i, j, h) = -\nabla_2\phi \approx -\frac{\phi(i, j+1, h) - \phi(i, j-1, h)}{2\Delta}$$

$$g_3(i, j, h) = -\nabla_3\phi \approx -\frac{\phi(i, j, h+1) - \phi(i, j, h-1)}{2\Delta}$$

- ▶ interpolate accelerations onto particle positions

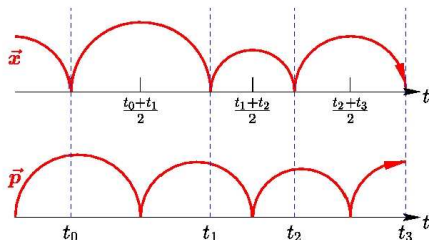
To avoid self-forces, always choose the same interpolation scheme as for the mass assignment!

Time integration: the leapfrog

$$\begin{aligned}\mathbf{v}(a + \Delta a/2) &= \mathbf{v}(a - \Delta a/2) + f_1(a) \mathbf{g}(a) \Delta a , \\ \mathbf{x}(a + \Delta a) &= \mathbf{x}(a) + f_2(a + \Delta a/2) \mathbf{v}(a + \Delta a/2) \Delta a .\end{aligned}$$

(f_1, f_2 are functions of a and cosmology,
arising from transition to comoving coordinates and $t \rightarrow a$).

- ▶ accurate to 2nd order
- ▶ computationally cheap
- ▶ time-symmetric
(symplectic)



(Note: most codes use adaptive timesteps)

Assessing the PM algorithm

Pros:

- ▶ Fast
- ▶ Relatively simple to implement

Cons:

- ▶ Resolution set by mesh spacing
 - higher resolution only by increasing N_g
 - out of memory
- ▶ Force is anisotropic on scales of mesh spacing

Particle-Particle Particle Mesh (P3M) algorithms

Idea:

- ▶ Split forces into long- and short range component
- ▶ Compute long range part using mesh and FFT (like PM)
- ▶ Obtain short range contribution by direct summation over neighbouring particles

Advantages:

- ▶ Sub-grid resolution, much higher dynamic range
- ▶ Mesh can be coarser than PM mesh
- ▶ Force anisotropies due to the mesh can be suppressed:
make force split at $r_s \gtrsim 2\Delta$

The force split (I)

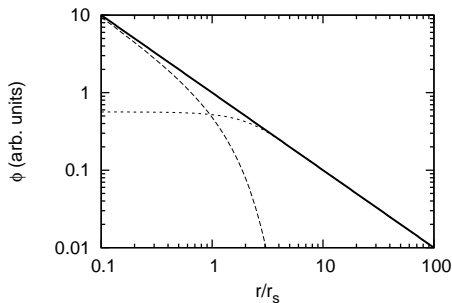
- ▶ Split potential of single particle into $\phi = \phi^{\text{long}} + \phi^{\text{short}}$, with

$$\phi^{\text{long}}(r) = -\frac{Gm}{r} \operatorname{erf}\left(\frac{r}{2r_s}\right), \quad \phi^{\text{s}}(r) = -\frac{Gm}{r} \operatorname{erfc}\left(\frac{r}{2r_s}\right)$$

- ▶ Long range part can be computed in Fourier space:

$$\phi_{\mathbf{k}}^{\text{long}} = \phi_{\mathbf{k}} \exp(-k^2 r_s^2)$$

- ▶ Finite differencing yields contribution to particle force



The force split (II)

The short range contribution is computed by direct summation ($r_i \equiv |\mathbf{x} - \mathbf{x}_i|$):

$$\phi^{\text{short}}(\mathbf{x}) = -Gm \sum_{i=1}^{N_p} \frac{\text{erfc}\left(\frac{r_i}{2r_s}\right)}{r_i}$$

More convenient: short range contribution to the force:

$$\mathbf{F}^{\text{short}}(\mathbf{x}) = -Gm \sum_{i=1}^{N_p} \frac{(\mathbf{x} - \mathbf{x}_i)}{r_i} \left[\text{erfc}\left(\frac{r_i}{2r_s}\right) + \frac{r_i}{r_s \sqrt{\pi}} \exp\left(-\frac{r_i^2}{4r_s^2}\right) \right]$$

The force split (III)

- ▶ Short range force is negligible for $\approx 4r_s$
→ sum up the short range contributions from all neighbours inside a sphere with $r = 4r_s$
- ▶ Add this to the force from the long-range mesh part
- ▶ Apply leapfrog as before

This is nice, but slow:

When clustering starts, simulation stalls!

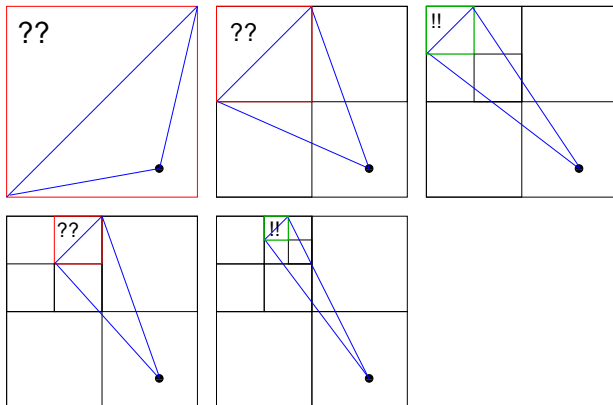
($\approx 10^{4-6}$ particles in an intermediate halo $\Rightarrow \approx 10^{8-12}$ force summations!)

\Rightarrow use tree-code to compute short range force (TreePM)

Tree codes: force computation

- ▶ For each particle: recursively walk the tree
- ▶ Use node as superparticle if opening angle criterion fulfilled, e.g.:

$$(\text{node diam.}) / \text{distance} < \theta_{\text{oa}}$$



AMR potential solver

In AMR, meshes are highly irregular

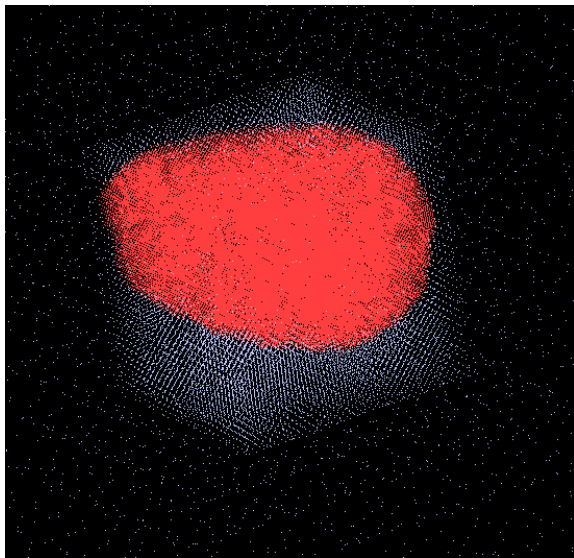
⇒ can't use FFT!

- ▶ get potential on coarse zero-level mesh using FFT
- ▶ now solve on the first refinement level:
- ▶ use potential from level 0 to get a first guess for level 1 potential
- ▶ solve for potential using *mesh relaxation* (iterative method)
- ▶ proceed likewise for the higher levels
- ▶ advance particles (nearly) as usual

Resimulating a halo

- ▶ run a simulation in cosmological volume ($\approx 50h^{-1}$ Mpc)
- ▶ identify interesting halo
- ▶ identify region from which halo formed at initial z (“amoeba”-shaped)
- ▶ sample mass distr. in this region with low-mass particles
- ▶ add small scale-modes in small periodic box around halo (use Zeldovich approx.)
- ▶ represent mass distribution outside high-res region with higher-mass particles
- ▶ evolve using tree code, tree-PM, AMR, etc.

Resimulating a halo

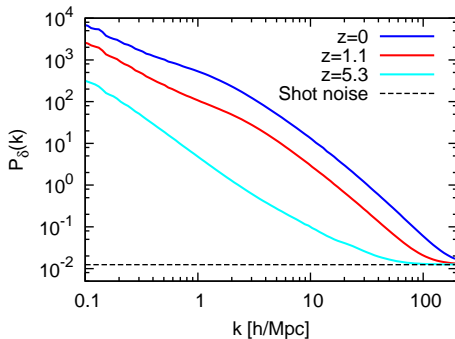


(Power et al. 2003)

Discreteness effects

- ▶ sample continuous density field with finite number of particles
⇒ sampling fluctuations
- ▶ model: random, Poisson-sampling of density field:

$$P_{\text{shot}}(k) = \frac{L_{\text{box}}^3}{N_p}$$



Force softening

- ▶ discreteness effects $\Rightarrow f$ not smooth, but “grainy”
- ▶ stochastic acceleration by close encounters
 \Rightarrow relaxation time decreases
- ▶ make system integrable with low order time integration scheme

\Rightarrow smooth the (short-range) potential, e.g.:

$$\phi(\mathbf{x}) = -Gm \frac{\operatorname{erfc}\left(\frac{r}{2r_s}\right)}{\sqrt{\mathbf{x}^2 + \epsilon^2}}$$

(ϵ : softening length)

- ▶ ϵ small: forces very noisy, system becomes collisional
- ▶ ϵ large: biased estimate of force, non-Newtonian

Choosing the softening length

- ▶ Trade-off between noisiness and bias of force:
⇒ can find optimal ϵ
(but depends on density field under consideration)
- ▶ Fit formulae $\epsilon(N_p)$ in Merrit (1996), *AJ* 111, 462;
Athanassoula et al. (2000), MNRAS 314, 475

Rule of thumb (for LSS simulations):

$$\epsilon \sim (\text{mean interparticle distance}) / (30 \dots 50)$$

- ▶ Convergence study for realistic halo:
Power et al. (2003), MNRAS 338, 14

Note:

Choice of ϵ affects time-step (and thus also run-time):
resolve close encounters!

Choosing the right kind of simulation

Applications of PM

- ▶ Large-scale structure studies (e.g. BAO)
- ▶ Weak lensing peak statistics
- ▶ LSS mass reconstruction

Need better resolution for:

- ▶ Cosmic shear on small/intermediate scales
- ▶ Higher-order statistics (3pt-cf, etc.)
- ▶ Simulating single halos

References

N-body techniques

- ▶ Hockney, R. & Eastwood, J.: “Computer simulation using particles”(McGraw-Hill)
- ▶ A. Klypin: Writing a PM code
<http://astro.uchicago.edu/~andrey/talks/PM/talk.pdf>
- ▶ Review (E. Bertschinger):
http://nedwww.ipac.caltech.edu/level5/March03/Bertschinger/Bert_contents.html
- ▶ Tree codes: Barnes & Hut (1986), Nature 324, 446
- ▶ Numerical techniques: Efstathiou et al. (1985), ApJS 57, 241
- ▶ ART: Kravtsov et al. (1997), ApJS 111, 73
- ▶ HYDRA: Couchman (1991), ApJ 368, 23

References

Public codes

- ▶ **PM code by Klypin & Holtzman:**
<http://astro.nmsu.edu/~aklypin/PM/pmcode/index.html>
- ▶ **GADGET-2 (V. Springel):**
<http://www.mpa-garching.mpg.de/gadget/>
- ▶ **AMIGA (A. Knebe)**
<http://popia.ft.uam.es/AMIGA/>
- ▶ **List of public codes (also initial condition generators):**
<http://www.itp.uzh.ch/astrosim/code/doku.php?id=home:code:nbody>
- ▶ **A number of tools (halo finding/visualization)**
<http://www-hpcc.astro.washington.edu/tools/tools.html>
- ▶ **IFRIT (N. Gnedin, visualization):**
<http://sites.google.com/site/ifrithome/>

References

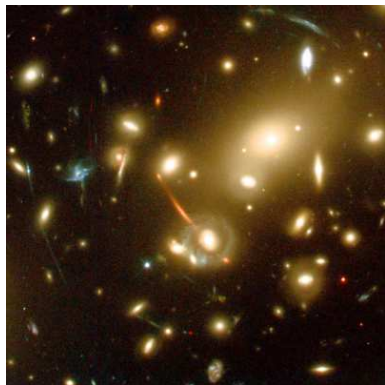
Useful tools

- ▶ Fastest Fourier Transform in the West:
<http://www.fftw.org>
- ▶ GNU Scientific Library:
<http://www.gnu.org/software/gsl/>
- ▶ Numerical Recipes:
<http://www.nr.com/oldverswitcher.html>

Semi-analytical models of galaxy formation

- ▶ Kauffmann et al. (1999), MNRAS 303, 188
- ▶ De Lucia & Blaizot (2007), MNRAS 375, 2
- ▶ Cole et al. (2000), MNRAS 319, 168
- ▶ Bower et al. (2006), MNRAS 370, 645
- ▶ ... and references therein

Simulations of cluster strong lensing



Arc statistics

- ▶ number of arcs by cluster lenses sensitive probe of cosmological model
 - ▶ number of massive objects
 - ▶ internal structure
 - ▶ discrepancy between observation and prediction (too few arcs predicted)
- ⇒ detailed simulations needed

Simulations of cluster strong lensing

Basic simulation steps

- ▶ run N -body simulation of a single cluster
- ▶ project cluster onto lens plane
- ▶ compute deflection angles
- ▶ populate source plane(s)
- ▶ shoot light rays starting at observer
- ▶ apply lens equation
- ▶ identify images
- ▶ do statistics

Lens plane construction

Projection

- ▶ project N -body particles onto 2D grid $\rightarrow \Sigma(\xi)$
(e.g. using CIC or TSC)
- ▶ can obtain 3 realizations by projecting along the three axes
- ▶ scale with Σ_{crit} appropriate for sources at redshift z_s
 $\rightarrow \kappa$
- ▶ possibly smoothing to reduce discreteness noise
(e.g. Gaussian kernel)

Computing the deflection angles

$$\alpha(\boldsymbol{\theta}) = \frac{1}{\pi} \int d^2\theta' \kappa(\boldsymbol{\theta}') \frac{\boldsymbol{\theta} - \boldsymbol{\theta}'}{|\boldsymbol{\theta} - \boldsymbol{\theta}'|^2}$$

Three ways to compute this:

(1) Direct summation

$$\alpha(\boldsymbol{\theta}_{ij}) \approx \Delta\theta^2 \sum_{kl} \kappa(\boldsymbol{\theta}_{kl}) \frac{\boldsymbol{\theta}_{ij} - \boldsymbol{\theta}_{kl}}{|\boldsymbol{\theta}_{ij} - \boldsymbol{\theta}_{kl}|^2}$$

slow if too many grid points ($\mathcal{O}(N^2 \times N^2)$)

(2) Compute sum using tree code

Computing the deflection angles

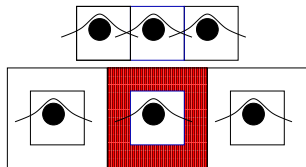
(3) Fast Fourier Transform (FFT):

Since $\alpha = \nabla\psi$ and $\nabla^2\psi = 2\kappa$:

$\tilde{\alpha}_i = -i l_j \tilde{\psi}$ and $-\ell^2 \tilde{\psi} = 2\tilde{\kappa}$

$$\Rightarrow \tilde{\alpha}_i(\ell) = \frac{2i l_j}{\ell^2} \tilde{\kappa}(\ell) \quad \text{for } \ell \neq 0$$

- ▶ fast ($\mathcal{O}(N^2 \log N^2)$)
- ▶ need significant zero-padding to reduce influence of periodic replicas of cluster



Ray-tracing

- ▶ shoot light rays starting from observer
- ▶ rays form grid on the lens (image) plane
- ▶ interpolate deflection angle onto light rays
- ▶ compute source plane position of rays using lens equation:

$$\beta(\theta) = \theta - \alpha(\theta)$$

⇒ “mapping table”

- ▶ can compute Jacobian using finite differencing, e.g.:

$$\mathcal{A}_{12}(\theta) = \frac{\partial \beta_1(\theta)}{\partial \theta_2} \approx \frac{\beta_1(\theta_1, \theta_2 + \Delta\theta) - \beta_1(\theta_1, \theta_2 - \Delta\theta)}{2\Delta\theta}$$

Finding images

- ▶ sample source plane uniformly with galaxies (e.g. elliptical)
- ▶ for each galaxy, find rays ending in it (using mapping table)
- ▶ identify images: connected pixels in image plane belonging to same source

Adaptive source grids for arc statistics:

- ▶ identify critical curves (change in sign of $\det \mathcal{A}$)
- ▶ find caustics using mapping table
- ▶ artificially increase source density around caustics by some factor
- ▶ for predictions of arc abundance, downweight these images by the same factor

Sources with redshift distribution

- ▶ so far: single source redshift
- ▶ can rescale α for different source redshifts:

$$\alpha'(\theta, z'_s) = \alpha(\theta, z_s) \frac{D_s(z_s) / D_{ds}(z_d, z_s)}{D_s(z'_s) / D_{ds}(z_d, z'_s)}$$

- ▶ in principle: do this for each source, find images, etc.
- ▶ more practical: bin sources in redshift

References

Simulations / Predictions

- ▶ Bartelmann & Weiss (1994), A&A 287, 1
- ▶ Bartelmann et al. (1998), A&A 330, 1
- ▶ Meneghetti et al. (2000), MNRAS 314, 338
- ▶ Meneghetti et al. (2008), A&A 482, 403
- ▶ Wambsganss et al. (2008), ApJ 676, 753
- ▶ Oguri et al. (2003), ApJ 599, 7
- ▶ Dalal et al (2004), ApJ 609, 50
- ▶ Li et al. (2005) ApJ, 635,795
- ▶ Wu & Chiueh (2006), ApJ 639,695

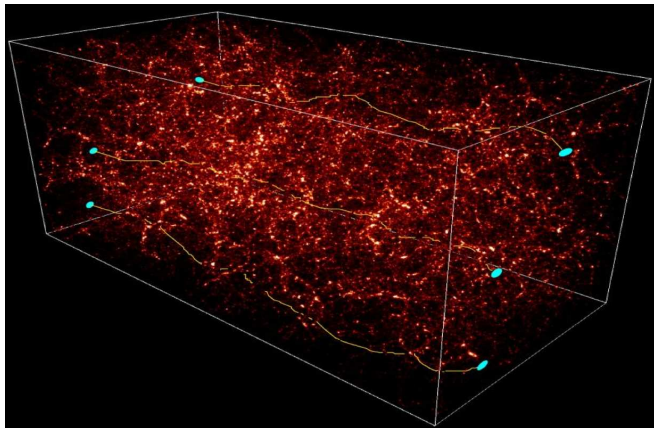
References

Observations

- ▶ Luppino et al. (1999), A&AS, 136, 117
- ▶ Zaritsky & Gonzalez (2003) ApJ 584, 691
- ▶ Gladders et al. (2003), ApJ 593, 48

Simulating weak lensing

- ▶ so far: single lens plane / dominant lens
- ▶ need multiple lens planes for simulating lensing by LSS



Uses of ray-tracing simulations

- ▶ test approximations
- ▶ Predictions: power spectra, n -point statistics
- ▶ covariance matrices of WL statistics
for inference of cosmological parameters
- ▶ realistic simulations of galaxy-galaxy lensing and cluster
weak lensing
- ▶ peak statistics
- ▶ study systematic effects
- ▶ ...

Building a ray-tracing simulation

The multiple-lens-plane algorithm

- ▶ Construct matter distribution along line of sight from N -body simulation(s)
- ▶ Discretization: divide into slices (≈ 100 Mpc thick) and project onto lens planes (3D \rightarrow 2D)
Deflections occur only at the lens planes
- ▶ Compute deflection angles and its derivatives for each lens plane
- ▶ Shoot light rays, starting from observer
- ▶ Recursion formulae yield ray positions on planes, Jacobian to each plane along rays

Assumptions & approximations

- ▶ density field does not evolve significantly during light travel time through slice
- ▶ geometrical weights (f_K , $a(w)$, etc.) are \approx constant within slice
- ▶ plane-parallel projection \rightarrow angles w.r.t. line of sight small

Computing the deflection angles

Notation

- ▶ $f_K^{(k)} \equiv f_K(w^{(k)})$ is the comoving angular diameter distance to the k -th lens plane
- ▶ $f_K^{(k,l)} \equiv f_K(w^{(l)} - w^{(k)})$
- ▶ $\beta^{(k)}$: angular coordinate on k -th lens plane
- ▶ $X^{(k)}$ is X on the k -th lens plane
- ▶ $w_{L,U}$ are the lower, upper slice boundaries

Computing the deflection angles

Reminder:

Eqn. of geodesic deviation:

$$\frac{d^2 \mathbf{x}}{dw^2} + K \mathbf{x} = -\frac{2}{c^2} [\nabla_{\perp} \phi[\mathbf{x}(\boldsymbol{\theta}, w), w]].$$

Solution:

$$\mathbf{x}(\boldsymbol{\theta}, w) = f_K(w) \boldsymbol{\theta} - \frac{2}{c^2} \int_0^w dw' f_K(w - w') [\nabla_{\perp} \phi[\mathbf{x}(\boldsymbol{\theta}, w'), w']]$$

Infinitesimal deflection angle:

$$d\alpha = \frac{\mathbf{x}(w + dw) - \mathbf{x}(w)}{f_K(dw)} = -\frac{2}{c^2} \nabla_{\perp} \phi(\mathbf{x})$$

Computing the deflection angles

For each LP: sum up all infinitesimal deflections along the z-direction:

$$\alpha^{(k)}(\beta^{(k)}) = \int_{w_L^{(k)}}^{w_U^{(k)}} dw \frac{2}{c^2} \nabla_{\perp} \phi(\mathbf{x})$$

Define deflection potential accordingly:

$$\psi^{(k)}(\beta^{(k)}) \equiv \frac{2}{c^2} \frac{1}{f_K^{(k)}} \int_{w_L^{(k)}}^{w_U^{(k)}} dw \phi(\mathbf{x})$$

so that

$$\alpha^{(k)}(\beta^{(k)}) = \nabla_{\theta} \psi^{(k)}(\beta^{(k)})$$

Computing the deflection angles

Define the projected density contrast $\sigma^{(k)}$ on the lens planes such that

$$\nabla_{\theta}^2 \psi^{(k)} = 2\sigma^{(k)}$$

Thus (using $\nabla_{\theta}^2 = f_K^{(k)2} \nabla_{\perp}^2$):

$$\begin{aligned}\sigma^{(k)} &= \frac{1}{c^2} f_K^{(k)} \int_{w_L^{(k)}}^{w_U^{(k)}} dw \nabla_{\perp}^2 \phi(\mathbf{x}) \\ &= \frac{1}{c^2} f_K^{(k)} \int_{w_L^{(k)}}^{w_U^{(k)}} dw \left[\nabla^2 \phi(\mathbf{x}) - \underbrace{\nabla_w^2 \phi(\mathbf{x})}_{\approx 0} \right]\end{aligned}$$

Computing the deflection angles

Using the Poisson-eq.

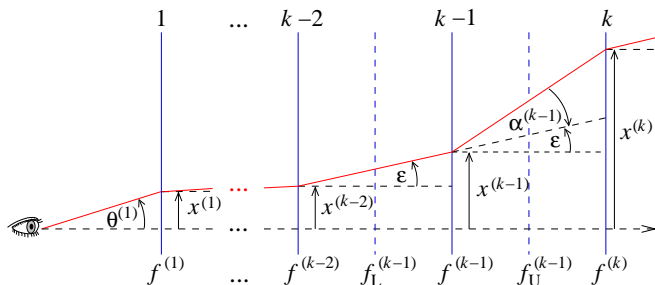
$$\nabla^2 \phi = \frac{3H_0^2 \Omega_m}{2a} \delta$$

we have

$$\sigma^{(k)}(\beta^{(k)}) = \frac{3H_0^2 \Omega_m}{2c^2} \frac{f_K^{(k)}}{a^{(k)}} \int_{w_L^{(k)}}^{w_U^{(k)}} dw' \delta(\beta^{(k)}, w')$$

⇒ relation between the deflection angles and the matter distribution projected onto lens planes

Ray-tracing



$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + f_K^{(k-1,k)} \left[\epsilon - \alpha^{(k-1)}(\mathbf{x}^{(k-1)}) \right], \text{ where}$$

$$\epsilon = \frac{\mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)}}{f_K^{(k-2,k-1)}}.$$

Ray-tracing

Using $\mathbf{x}^{(k)} = f_K^{(k)} \boldsymbol{\beta}^{(k)} \rightarrow$ angular coordinates:

$$\boxed{\boldsymbol{\beta}^{(k)}} = \frac{1}{f_K^{(k)}} \left[F^{(k)} f_K^{(k-1)} \boxed{\boldsymbol{\beta}^{(k-1)}} - f_K^{(k-1,k)} \boxed{\boldsymbol{\alpha}^{(k-1)}(\boldsymbol{\beta}^{(k-1)})} - G^{(k)} f_K^{(k-2)} \boxed{\boldsymbol{\beta}^{(k-2)}} \right],$$

where

$$F^{(k)} = \left(1 + \frac{f_K^{(k-1,k)}}{f_K^{(k-2,k-1)}} \right) \quad \text{and} \quad G^{(k)} = \frac{f_K^{(k-1,k)}}{f_K^{(k-2,k-1)}}$$

$\Rightarrow \boldsymbol{\beta}^{(k)}$ recursively given by the previous two positions and the deflection angle at previous lens plane

The Jacobian matrix

Jacobian matrix for lens mapping from image plane (θ) to k -th lens plane:

$$\mathcal{A}^{(k)}(\theta) = \frac{\partial \beta^{(k)}}{\partial \theta}$$

Differentiate recursion relation for $\beta^{(k)}$:

$$\begin{aligned} \mathcal{A}^{(k)}(\theta) = & F^{(k)} \frac{f_K^{(k-1)}}{f_K^{(k)}} \mathcal{A}^{(k-1)}(\theta) \\ & - \frac{f_K^{(k-1,k)}}{f_K^{(k)}} \mathcal{U}^{(k-1)}(\beta^{(k-1)}) \mathcal{A}^{(k-1)}(\theta) \\ & - G^{(k)} \frac{f_K^{(k-2)}}{f_K^{(k)}} \mathcal{A}^{(k-2)}(\theta) . \end{aligned}$$

The shear matrix

$\mathcal{U}^{(k)}$ is the shear matrix:

$$\mathcal{U}_{ij}^{(k)} = \frac{\partial \alpha_i^{(k)}(\boldsymbol{\beta}^{(k)})}{\partial \beta_j^{(k)}} = \frac{\partial^2 \psi^{(k)}(\boldsymbol{\beta}^{(k)})}{\partial \beta_i^{(k)} \partial \beta_j^{(k)}}.$$

Together with

$$\boldsymbol{\alpha}^{(k)}(\boldsymbol{\beta}^{(k)}) = \nabla_{\theta} \psi^{(k)}(\boldsymbol{\beta}^{(k)})$$

$$\nabla_{\theta}^2 \psi^{(k)} = 2\boldsymbol{\sigma}^{(k)}$$

and

$$\boldsymbol{\sigma}^{(k)}(\boldsymbol{\beta}^{(k)}) = \frac{3H_0^2 \Omega_m}{2c^2} \frac{f_K^{(k)}}{a^{(k)}} \int_{w_L^{(k)}}^{w_U^{(k)}} dw' \delta(\boldsymbol{\beta}^{(k)}, w')$$

we have all necessary quantities for the ray-tracing.

Getting lensing quantities

- ▶ algorithm yields Jacobian from observer to each lens plane
- ▶ \mathcal{A} in general not symmetric (matrix products $\mathcal{U} \times \mathcal{A}$)
- ▶ two ways to define lensing quantities:

$$\mathcal{A} = \begin{pmatrix} 1 - \kappa' - \gamma'_1 & -\gamma'_2 - \varpi' \\ -\gamma'_2 + \varpi' & 1 - \kappa' + \gamma'_1 \end{pmatrix}$$

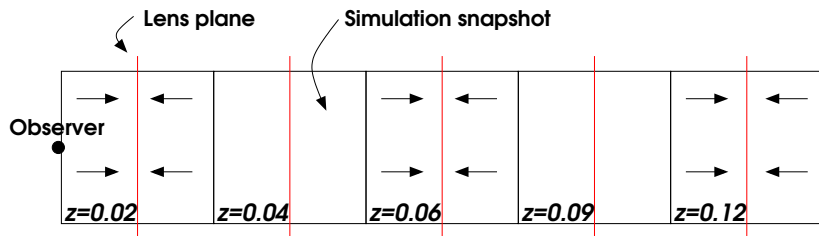
$$\mathcal{A} = \begin{pmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} 1 - \kappa - \gamma_1 & -\gamma_2 \\ -\gamma_2 & 1 - \kappa + \gamma_1 \end{pmatrix} .$$

Note: $\kappa' \approx \kappa$ and $\gamma' \approx \gamma$ in the weak lensing regime

Practical implementation: lens planes

Lens plane construction for small boxes

- ▶ Project complete snapshot onto a lens plane
- ▶ Avoid repetition of structure along LOS by random rotations, translations



Practical implementation: lens planes

Lens plane construction for small boxes: assumptions

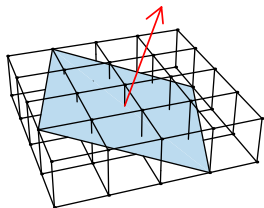
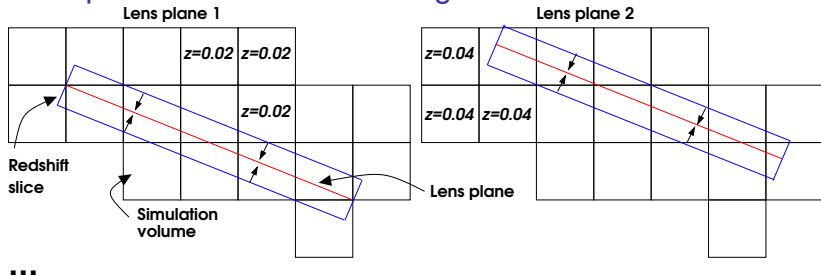
- ▶ No significant evolution of density field in snapshot
- ▶ $f_K, a(w) \approx \text{const.}$ within snapshot

Boxes larger than ≈ 200 Mpc

- ▶ Cannot project whole box: assumptions fail
→ large discretization error
- ▶ Simply subdivide box and proceed as before?
No, random transformations destroy large-scale correlations

Practical implementation: lens planes

Lens plane construction for large boxes



2

- ▶ LOS not \perp to box boundaries
- ▶ Clever choice of angle:
 - ▶ Lens planes still periodic
 - ▶ Avoid repetition of structures along LOS

Practical implementation: lens planes

Smoothing

- ▶ N -body particles represent a phase-space cell
- ▶ distance to nearest neighbour related to local density

⇒ need some smoothing on the lens plane

- ▶ simplest: Gaussian with fixed width
- ▶ better: adaptive smoothing
e.g. SPH-like particle shape:

$$\Sigma_p(\mathbf{x}) \propto \left(1 - \frac{|\mathbf{x} - \mathbf{x}_p|^2}{r_p^2} \right)^2$$

r_p : distance to 64th nearest neighbour

Practical implementation: lens planes

Since lens planes are periodic: go to Fourier space!

Then: $\nabla \rightarrow -i\ell$

$$\tilde{\psi}^{(k)}(\ell) = -2 \frac{\tilde{\sigma}^{(k)}(\ell)}{|\ell|^2}$$

$$\tilde{\alpha}^{(k)}(\ell) = -i\ell \tilde{\psi}^{(k)}(\ell) = 2i \tilde{\sigma}^{(k)}(\ell) \frac{\ell}{|\ell|^2}$$

$$\tilde{\mathcal{U}}_{ij}^{(k)} = -\ell_i \ell_j \tilde{\psi}^{(k)}(\ell) = \frac{2\ell_i \ell_j}{|\ell|^2} \tilde{\sigma}^{(k)}(\ell)$$

Note: sometimes faster to use finite differencing to get α, \mathcal{U}

Practical implementation: large lens planes (I)

In tilted-los-approach:

lens planes considerably bigger than box face!

→ plane too big for memory at full resolution

Multi-mesh approach:

- ▶ split deflection potential into long- and short range components
- ▶ coarse mesh for full lens plane with long-range part
- ▶ cover plane with several high-res meshes for short-range part
- ▶ compute short-range meshes only for parts of plane where there are light rays

Practical implementation: large lens planes (II)

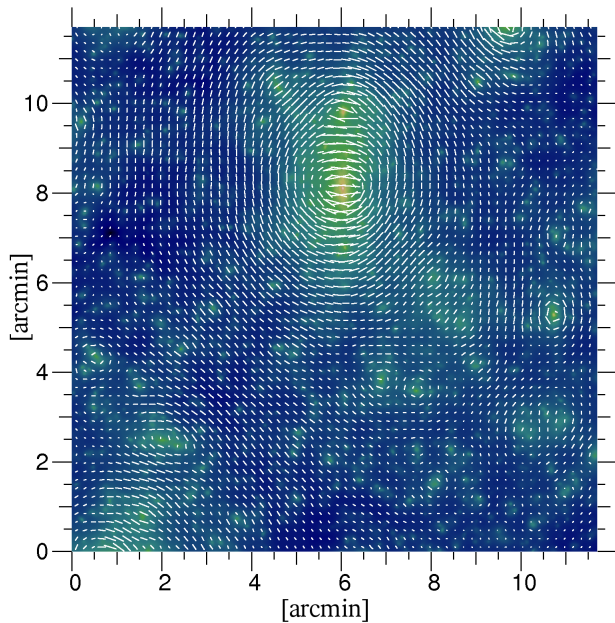
Coarse mesh:

- ▶ assign particles to mesh
- ▶ compute potential in Fourier space
- ▶ long range part: $\tilde{\psi}_1(\ell) = \tilde{\psi}(\ell) \exp\left(-\theta_{\text{split}}^2 \ell^2\right)$,
where $\theta_{\text{split}} = x_{\text{split}}/f_K^{(k)}$, where x_{split} is the comoving splitting length

Fine mesh:

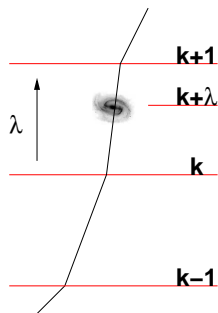
- ▶ fine mesh resolution: half of softening length
- ▶ $\tilde{\psi}_s(\ell) = \tilde{\psi}(\ell) \left[1 - \exp\left(-\theta_{\text{split}}^2 \ell^2\right)\right]$
- ▶ mesh slightly larger than necessary to avoid wrap-around effects when using FFT

Example



Making mock galaxy catalogues

- ▶ randomly sample image plane with galaxies
- ▶ assign redshifts to galaxies according to some $p(z)$
- ▶ linearly interpolate Jacobian onto galaxy positions in θ
- ▶ interpolate Jacobian in redshift:
 - ▶ let $\lambda \in [0, 1]$ and $f_K^{(k)} \leq f_K^{(k+\lambda)} \leq f_K^{(k+1)}$
 - ▶ write down recursion relation for $\mathcal{A}^{(k+\lambda)}$ and $\mathcal{A}^{(k+1)}$ and subtract them:



$$\mathcal{A}^{(k+\lambda)}(\theta) = \frac{f_K^{(k,k+\lambda)}}{f_K^{(k+\lambda)}} \frac{f_K^{(k+1)}}{f_K^{(k,k+1)}} \mathcal{A}^{(k+1)}(\theta) + \frac{f_K^{(k)}}{f_K^{(k+\lambda)}} \left(1 - \frac{f_K^{(k,k+\lambda)}}{f_K^{(k,k+1)}} \right) \mathcal{A}^{(k)}(\theta)$$

Including galaxies: SAMs

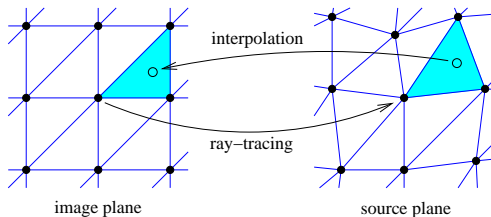
Semi-analytic models (SAMs) yield galaxy positions β_g on the lens planes

Task:

- ▶ get lensed positions $\theta_g(\beta_g)$
in contrast to ray-tracing, where we computed $\beta^{(k)}(\theta)$
- ▶ get Jacobians at galaxy position (linear interpolation)
- ▶ get lensed magnitudes:

$$m_X^{(\text{lensed})} = m_X^{(\text{unlensed})} - 2.5 \log(\mu(\theta_g)) ,$$

Including galaxies: SAMs



Solution:

- ▶ during ray-tracing, store ray positions on each lens plane
- ▶ partition image plane grid into triangles
- ▶ backtrace triangles to source plane using $\beta^{(k)}(\theta)$ from RT
- ▶ find all galaxies within given triangle on source plane
- ▶ get image positions, Jacobians for each galaxy by linear interpolation from the three light rays at the corners
- ▶ galaxy covered by multiple triangles: multiple images

Ray-tracing summary

- ▶ Construct redshift slices
- ▶ Project onto lens planes:
 - ▶ small boxes: full box + random transformations
 - ▶ large boxes: tilted los + different ray positions on first plane
- ▶ Compute defl. angles, shear matrices in Fourier space
- ▶ Shoot light rays, get positions and Jacobians by recursion
- ▶ Create mock galaxy catalogue:
 - ▶ randomly sample image plane with galaxies of different redshift
 - ▶ get lensed positions, magnitudes of SAM galaxies

Resolution issues

Lens planes

- ▶ maximum resolution of N -body sim. in densest regions $\approx \epsilon$
 \Rightarrow choose $\Delta x_{LP} \approx \epsilon$
- ▶ smoothing on lens planes is important to suppress shot noise (in particular from nearby planes)

Resolution issues

$$\kappa(\theta) \propto \int dw' \frac{f_K(w_s - w') f_K(w')}{a(w') f_K(w_s)} \delta[f_K(w') \theta, w']$$

Angular resolution

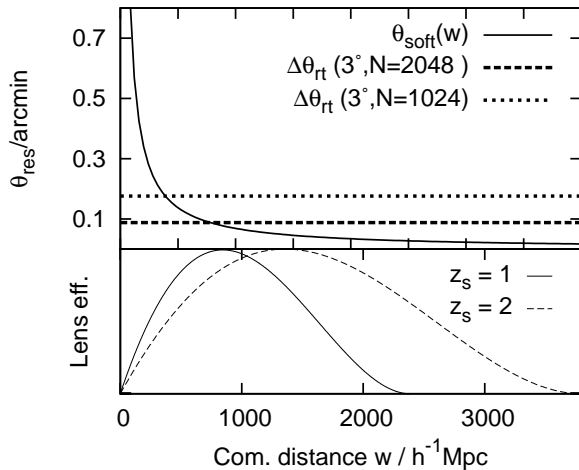
- ▶ max. angular resolution \approx resolution of lens planes at peak of lens efficiency

$$w_{\max} = \max_w \frac{D_+[a(w)] f_K(w_s - w) f_K(w)}{a(w) f_K(w_s)}$$

\Rightarrow choose $\Delta\theta_{\text{rt}} \lesssim \theta_{\text{soft}}(w_{\max})$

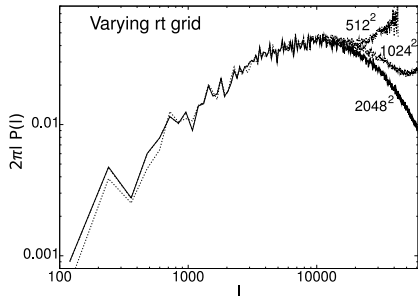
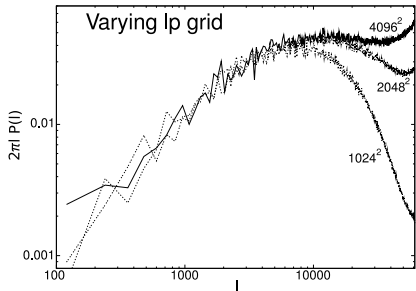
- ▶ resolution for nearby lens planes dominated by grid on lens planes/ force softening, but downweighted by lens efficiency
- ▶ resolution for distant planes dominated by $\Delta\theta_{\text{rt}}$

Resolution issues



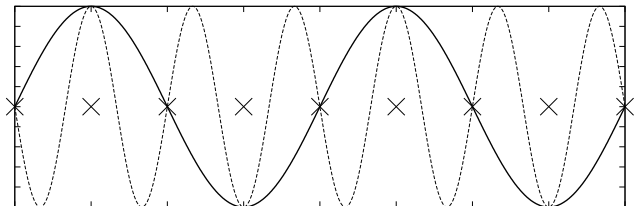
(flat Universe, $\epsilon = 20 h^{-1} \text{kpc}$)

Effect on the power spectrum



- ▶ bigger pixels on LP: more smoothing, less shot noise
- ▶ fewer light rays: undersample lens planes → aliasing

Aliasing



If matter distribution on lens plane has power on scales smaller than grid spacing of light rays:

(see also: sampling theorem)

power will be moved (“aliased”) into the representable range

⇒ choose grid spacing small enough!

How many realizations can you get?

- ▶ most signal from peak of lensing efficiency (at w_{\max})
- ▶ there: diameter of light cone $d_{\max} = \omega f_K(w_{\max})$,
(ω : diameter of field of view)

$$\Rightarrow N_{\text{real}} \lesssim 3 \left(\frac{L_{\text{box}}}{d_{\max}} \right)^2$$

- ▶ factor “3”: three “quasi-independent” directions
 - ▶ small box: random transformations
 - ▶ large box: choose three orthogonal lines of sight

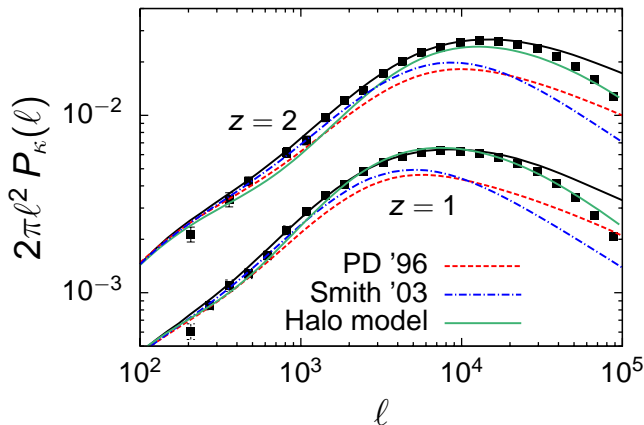
Example:

- ▶ $L_{\text{box}} = 500 h^{-1}$ Mpc
- ▶ $\omega = 4^\circ$
- ▶ $z_{\text{source}} = 1$

$$\Rightarrow N_{\text{real}} \lesssim 3 \times 63$$

Results from ray-tracing: power spectrum

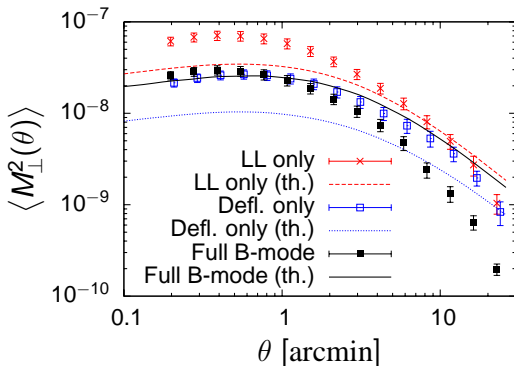
Comparison of power spectrum from ray-tracing through Millennium Simulation to theoretical predictions



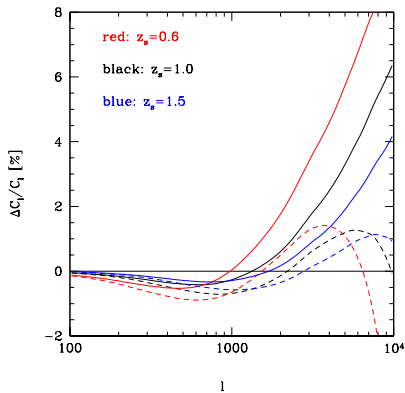
Results from ray-tracing

B-modes

- ▶ switch off either deflection angles or non-linear coupling between lens planes (LL, matrix products)
- ▶ measure B-mode
- ▶ compare to theoretical predictions

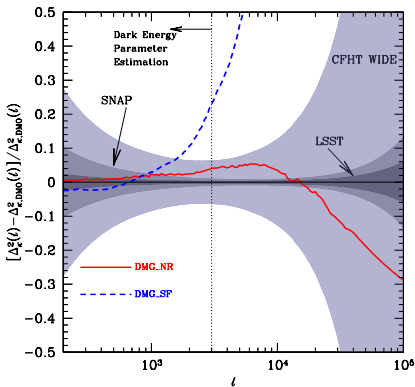


Importance of baryons



(Jing et al. 2006, ApJ 640, 119)

dashed: non-rad. baryons
solid: rad. cooling/heating,
 star formation, SN feedback



(Rudd et al. 2008, ApJ 672, 19)

red: non-rad. baryons,
blue: rad. cooling/heating,
 star formation, SN feedback,
 sources at $z = 1$

References

- ▶ Derivation of the multiple-lens-plane-*alg.*:
Seitz et al (1994), CQGra 11, 2345
- ▶ Numerical issues, convergence:
Premadi et al. (1998), ApJ 493, 10
- ▶ Influential paper:
Jain et al. (2000), ApJ 530, 547
- ▶ 3D algorithm:
Barber et al. (2000), MNRAS 319, 267
- ▶ Variant of the algorithm (tiling technique):
Hamana & Mellier (2001), MNRAS 327, 169
- ▶ White & Vale (2003), APh 22, 19
- ▶ Lensed Mock Map Facility:
Forero-Romero et al. (2007), MNRAS 379, 1507

References

- ▶ Ray-tracing through Millennium Simulation, including SAM:
Hilbert et al. (2009), A&A 499, 31
- ▶ Large set of simulations:
Sato et al. (2009), ApJ 701, 945
- ▶ Sim. for weak lensing peak statistics:
Dietrich & Hartlap (2009), astro-ph:0906.3512

Full-sky/CMB lensing:

- ▶ Das & Bode (2008), ApJ 682, 1
- ▶ Fosalba et al. (2008), MNRAS 391, 435
- ▶ Teyssier et al. (2009), A&A 497, 335