



GLASS

GENERATOR FOR LARGE SCALE STRUCTURE

Nicolas Tessore (UCL)

German Centre for Cosmological Lensing
10 Mar 2023

GLASS

GLASS is a Python library to simulate a past light cone. It randomly samples the matter field and forward-models everything else, such as e.g. the weak lensing and galaxy fields. One particular application is to create catalogues for photometric galaxy surveys.

GLASS

GLASS is a Python library to simulate a past light cone. It randomly samples the matter field and forward-models everything else, such as e.g. the weak lensing and galaxy fields. One particular application is to create catalogues for photometric galaxy surveys.



[arXiv:2302.01942](https://arxiv.org/abs/2302.01942)



[glass-dev/glass](https://github.com/glass-dev/glass)



glass.readthedocs.io

OUTLINE

Methodology

Problems in current simulations and how we solved them in *GLASS*. Useful to anyone doing this kind of simulation.

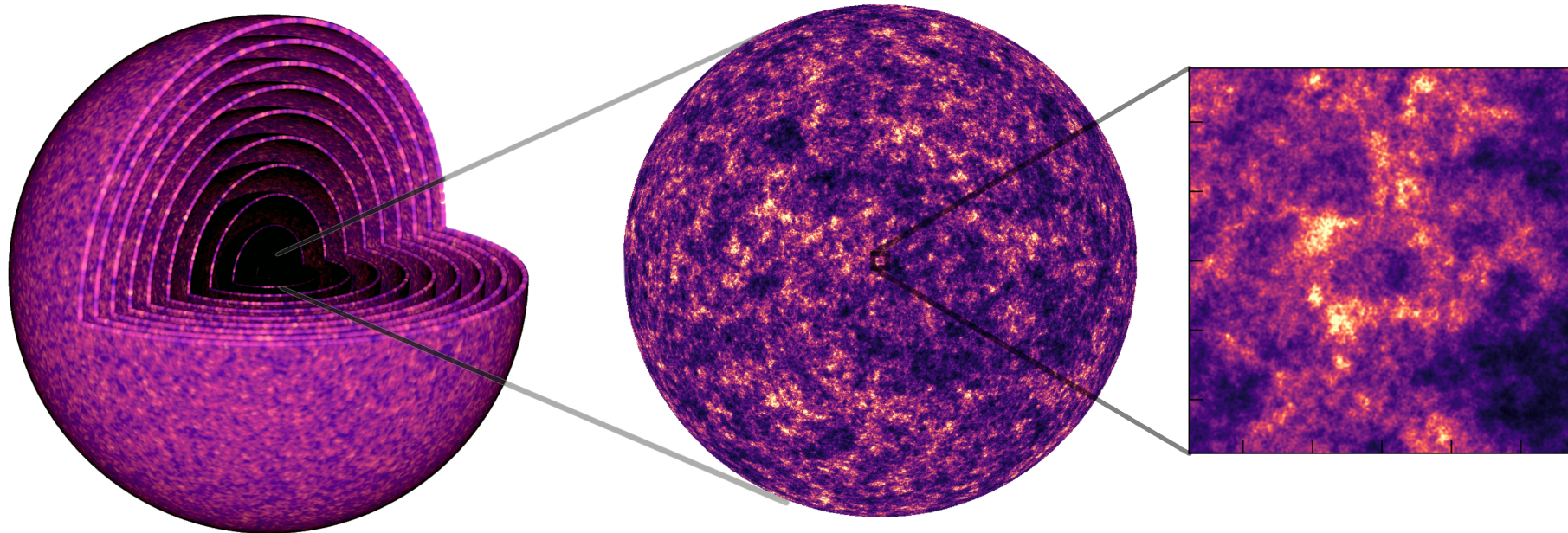
Implementation

Specific example of how to use *GLASS*: Simulating a galaxy survey.

METHODOLOGY

SPHERICAL SIMULATIONS

Simulate cosmic fields as observed on the sphere



High angular resolution, low radial resolution

(matches photometric galaxy surveys)

ANGULAR DISCRETISATION

Angular statistics for Stage 4 survey: $\ell_{\max} \approx 5000$

≈ 2 arcmin angular resolution

≈ 2 Mpc ($z = 1$), ≈ 0.25 Mpc ($z = 0.1$)

At the same time: coherent **full sky** simulations

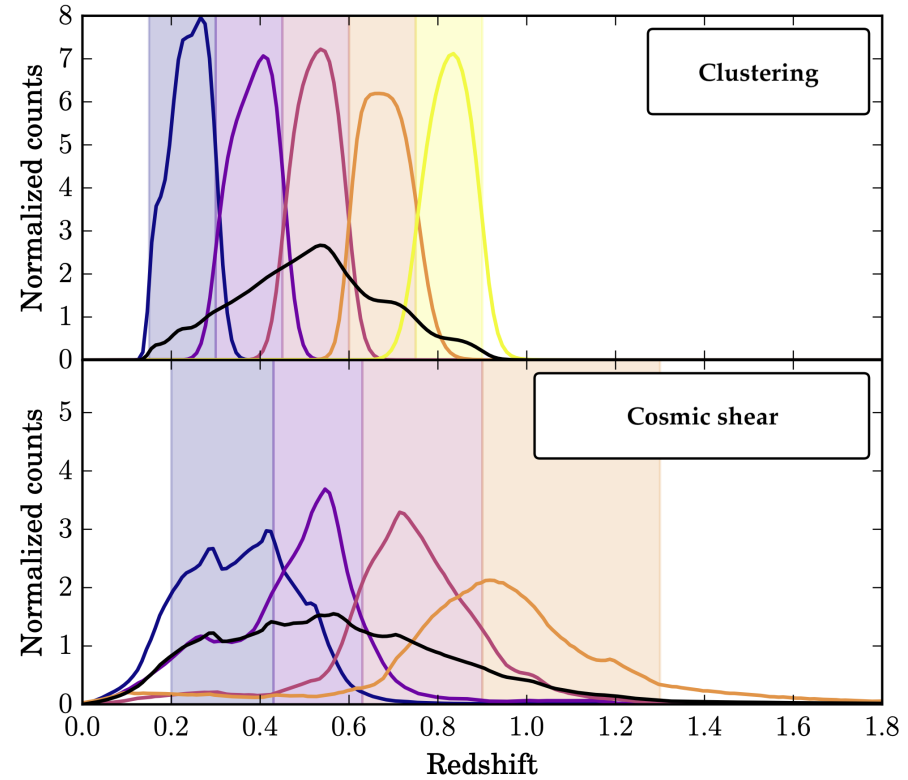
≈ 7 Gpc ($z = 1$), ≈ 0.85 Gpc ($z = 0.1$)

RADIAL DISCRETISATION

Two approaches to split the simulation in the radial direction
(Xavier et al. 2016)

EFFECTIVE FIELDS

Generate the effective random field for each observable,
e.g. angular clustering and cosmic shear in all tomographic bins

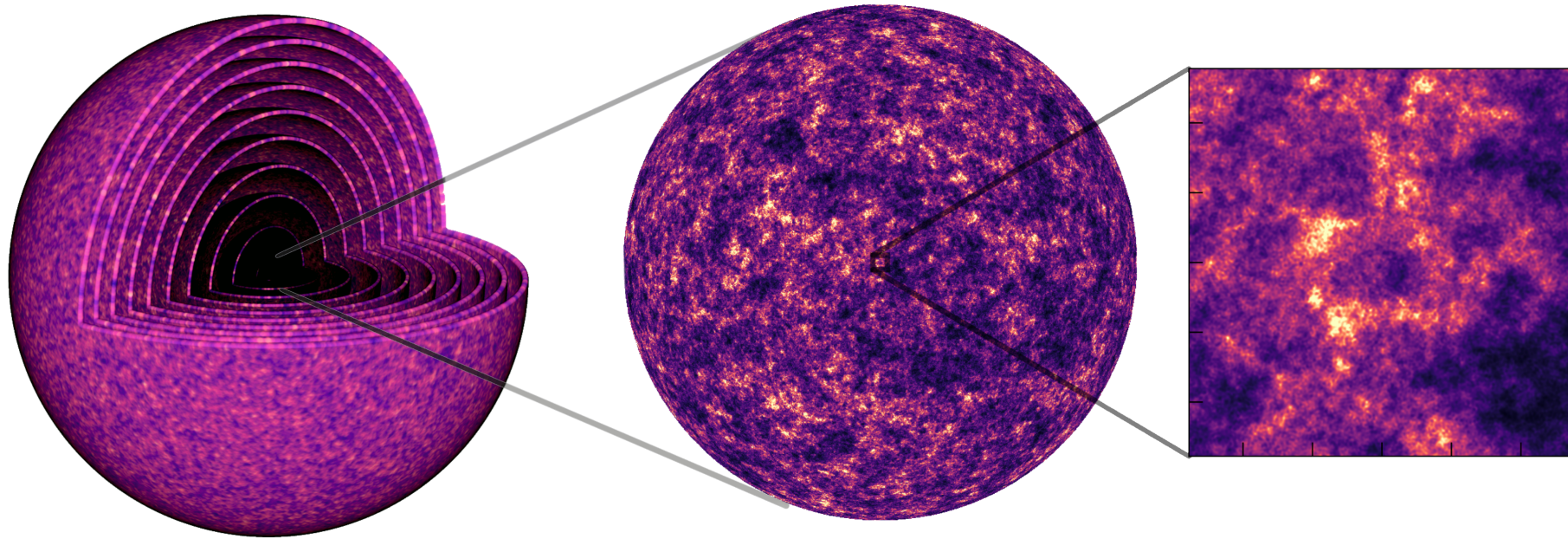


(DES Y1, image credit arXiv:1810.02499)

Must **prescribe** distributions of all fields and two-point statistics of all field combinations!

FORWARD MODELLING

Discretise the **entire** past light cone into projected matter fields



Observables are forward-modelled from initial matter simulation, which does not depend on or even know about eventual data

This is how *GLASS* works

PROBLEM: NUMBER OF SHELLS

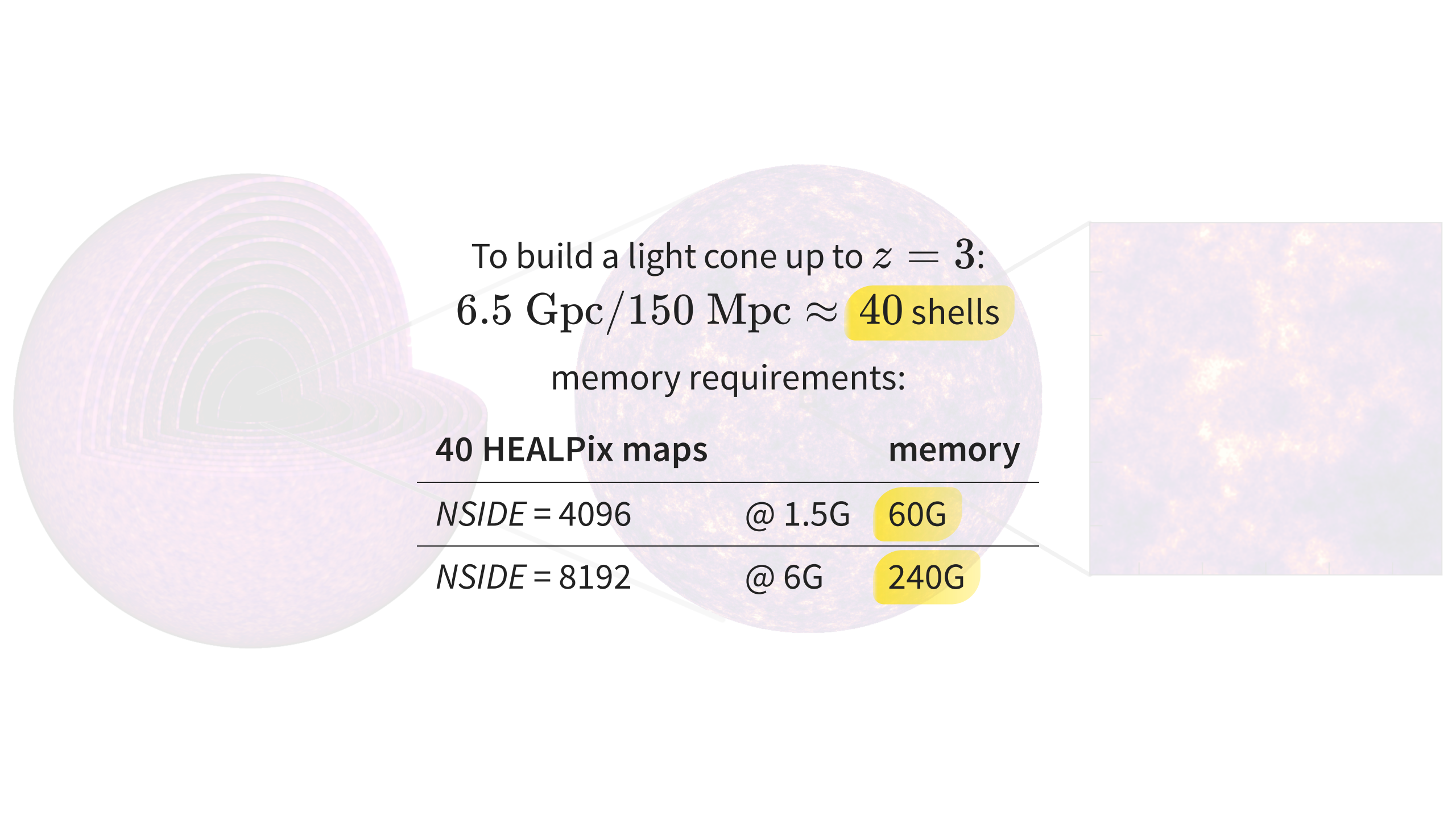
Some observables (e.g. galaxies and weak lensing) need to compute **integrals** over the matter fields

$$S = \int \delta(z) dz$$

These integrals must be **approximated** using the discretised matter shells

$$S \approx \sum_i w_i \delta_i$$

For **per cent-level accuracy** in clustering and cosmic shear, we need shells of ≈ 150 Mpc comoving



To build a light cone up to $z = 3$:
 $6.5 \text{ Gpc}/150 \text{ Mpc} \approx 40 \text{ shells}$

memory requirements:

40 HEALPix maps

memory

$NSIDE = 4096$

@ 1.5G

60G

$NSIDE = 8192$

@ 6G

240G

HINT 1

Multivariate Gaussian random variables can be sampled **iteratively**

Add one more Gaussian random variable to n existing ones:

$$\Sigma_{n+1} = \begin{pmatrix} \Sigma_n & \mathbf{c}_n \\ \mathbf{c}_n^\top & \sigma_{n+1}^2 \end{pmatrix}$$

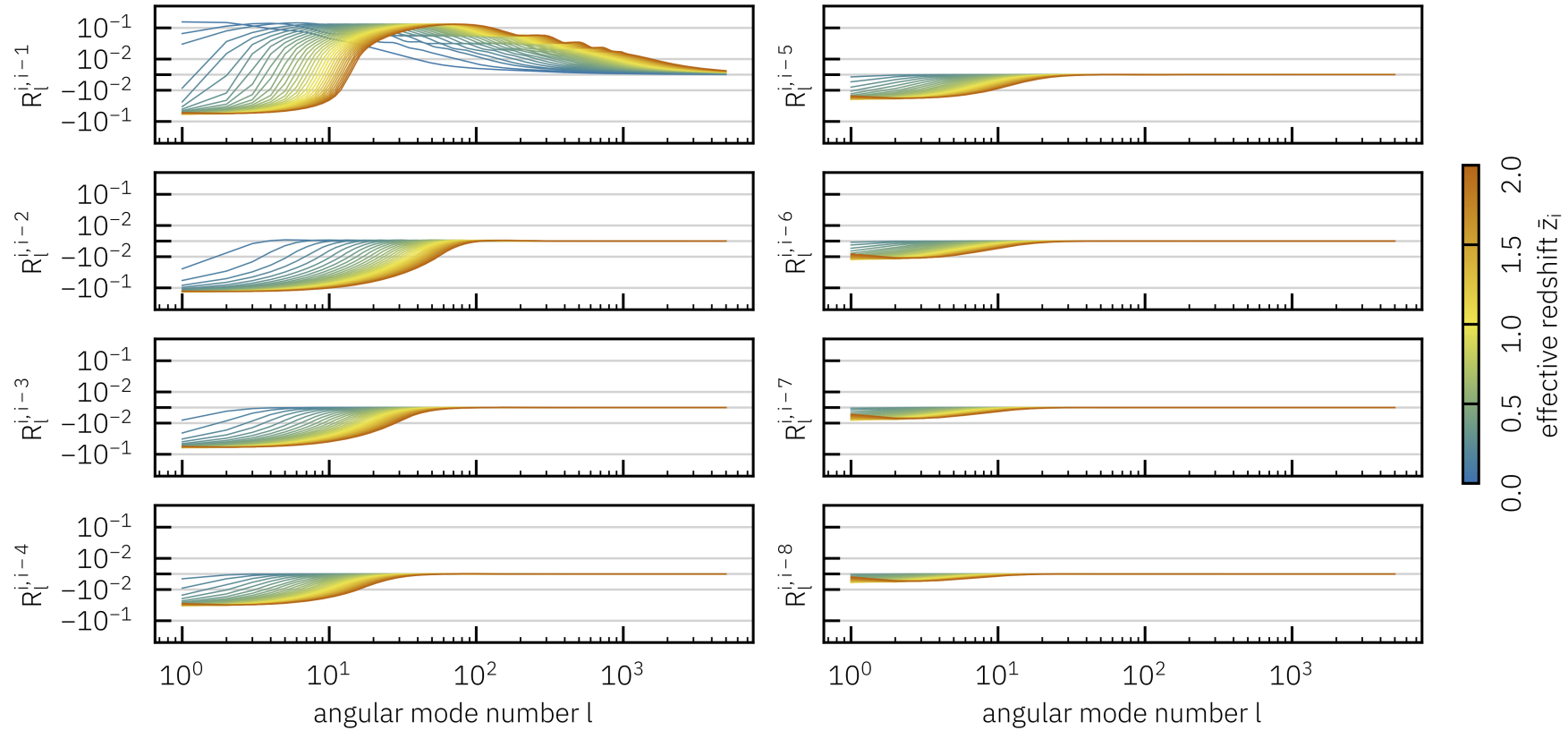
Sample new Gaussian variate **independently** from the conditional distribution:

$$\tilde{\mu}_{n+1} = \mu_{n+1} + \mathbf{c}_n^\top \Sigma_n^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_n)$$

$$\tilde{\sigma}_{n+1}^2 = \sigma_{n+1}^2 - \mathbf{c}_n^\top \Sigma_n^{-1} \mathbf{c}_n$$

HINT 2

Matter is not correlated much beyond a few hundred Mpc



$$R_l^{ij} = \frac{C_l^{ij}}{\sqrt{C_l^{ii} C_l^{jj}}} \text{ is the correlation coefficient for shells } i \text{ and } j$$

SOLUTION

We can sample each new Gaussian random field **iteratively** from the conditional distribution of earlier shells

(imprints all cross-correlations between shells)

After a few iterations, we can **forget about past shells**

(constant memory use, no matter how many shells there are in total)

We **transform** the Gaussian random fields to a more suitable distribution, e.g. **lognormal**

PROBLEM: ACCURATE LOGNORMAL TRANSFORMS

Lognormal transformation:

$$Y = \exp(X) - 1$$

Need to simulate Gaussian random field X so that the statistics of observed lognormal field Y come out right

TWO-POINT STATISTICS IN REAL SPACE

Angular correlation function relation for lognormal transformation:

$$C(\theta) = \exp\{G(\theta)\} - 1$$

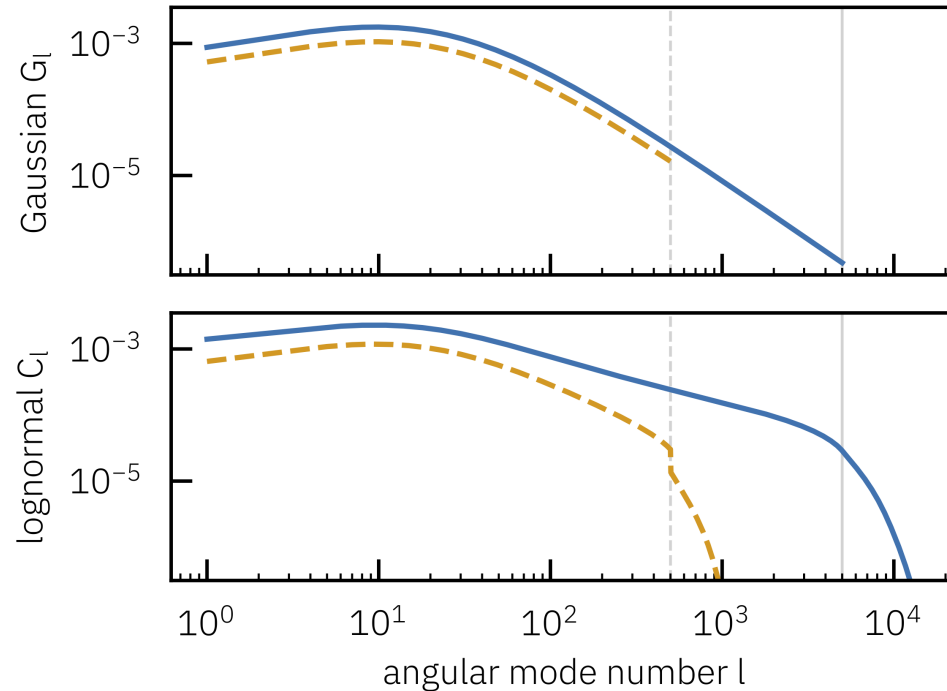
$$G(\theta) = \ln\{1 + C(\theta)\}$$

A pointwise transform $Y = f(X)$ in real space always results in a simple relation $C(\theta) = C(G(\theta))$ for the correlation function

Unfortunately, neither theory nor simulation operate in real space

TWO-POINT STATISTICS IN HARMONIC SPACE

Transformation is non-local in harmonic space and **mixes all scales**



- Band-limited lognormal spectra cannot be obtained from band-limited Gaussian spectra, and vice versa
- Theory always gives us band-limited lognormal spectra
- Simulation always requires band-limited Gaussian spectra

PROBLEM

Given a band-limited lognormal spectrum, it is generally **impossible** to construct an exact band-limited Gaussian spectrum

Necessary approximations to make it work are either **not accurate enough** or require going to **much higher resolution**

In short: $C_l \rightarrow G_l$ does not work!

SOLUTION

The true angular matter power spectrum is not band-limited; it is only our theory that stops at some point

The band-limited Gaussian spectrum that we require for simulation does not produce a band-limited lognormal field — **so what?!**

Solve for the Gaussian spectrum that reproduces the known modes of the lognormal spectrum: $G_l \rightarrow C_l$

For *GLASS*, we wrote a simple Gauss–Newton solver; it converges in a few steps but there is lots of room for improvement

PROBLEM: ITERATIVE LENSING

Convergence at redshift z requires all matter at lower redshift:

$$\kappa(z) = \frac{3\Omega_m}{2} \int_0^z \delta(z') \frac{x_M(z') x_M(z', z)}{x_M(z)} \frac{1+z'}{E(z')} dz'$$

Naive approximation by discrete sum would require us to
keep all shells at lower redshift in memory:

$$\kappa_i \approx \sum_{j \leq i} w_{ij} \delta_j$$

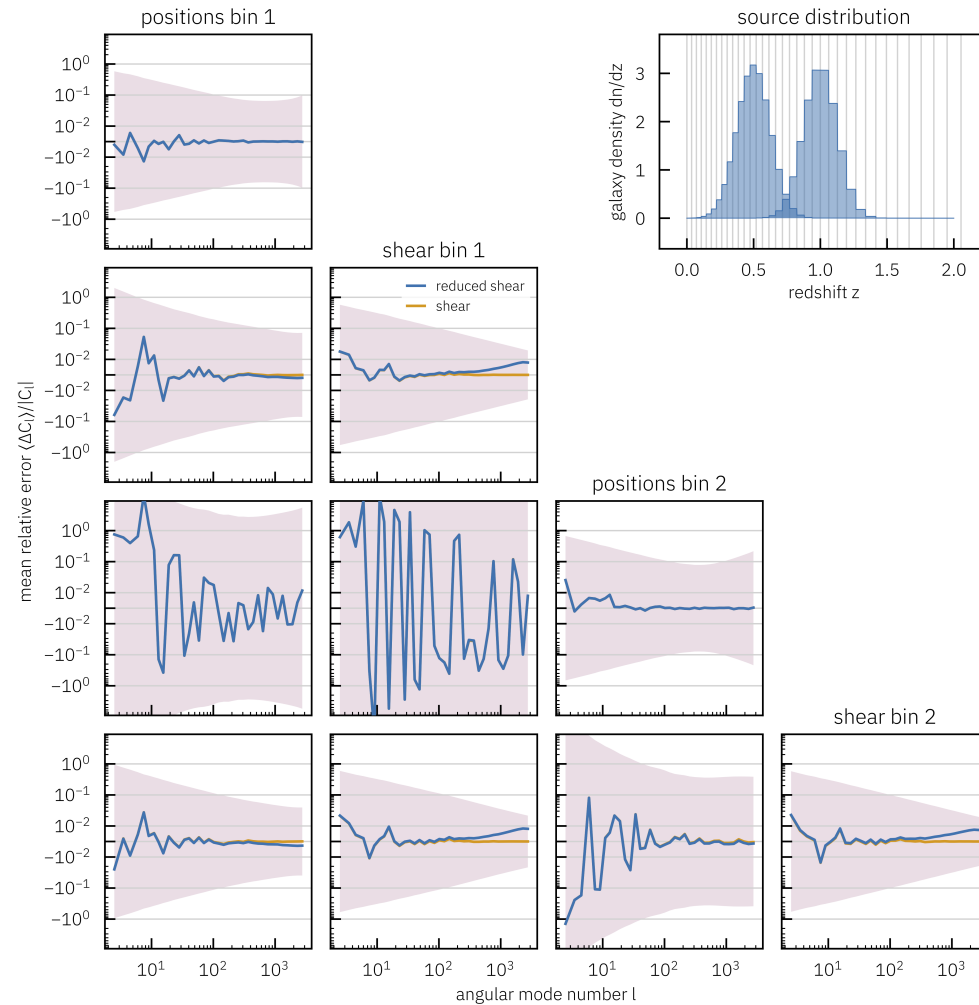
SOLUTION

Apply the iterative lensing technique from
multi-plane strong lensing

$$\kappa_i = \dots \kappa_{i-1} + \dots \kappa_{i-2} + \dots \delta_{i-1}$$

We only need to keep **two convergence maps** in memory to
compute weak lensing **iteratively**

RESULTS



Better than per cent-level accuracy is achievable!

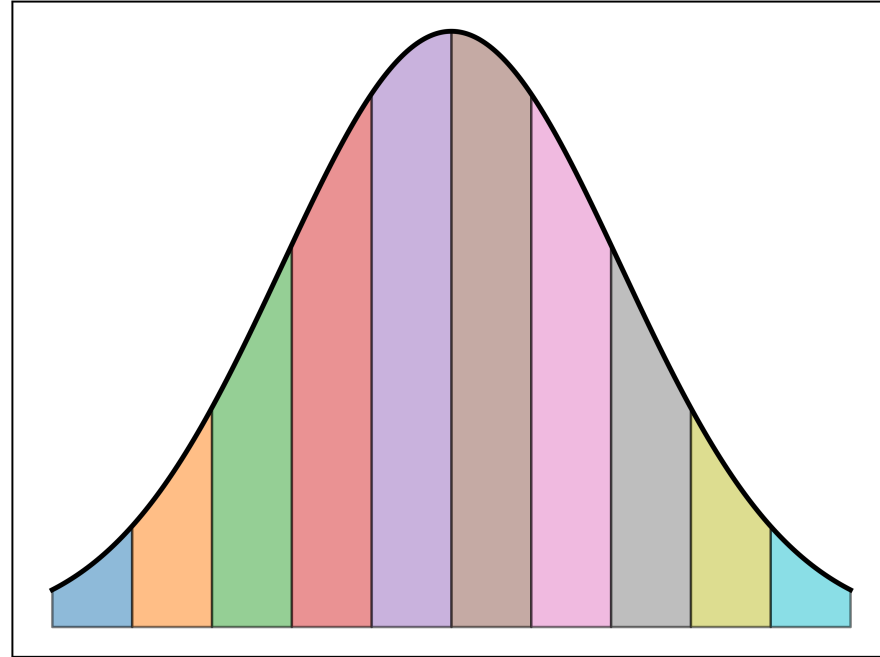
BONUS PROBLEM: ACCURATE THEORY (UNSOLVED)

We **need** faster and more accurate angular power spectrum codes

Anecdotally, we spent many times more time and effort in getting theory to the level of simulations than vice versa

Inaccurate theory leads to **inconsistencies** in the results, e.g. matter looks accurate, lensing looks accurate, but galaxies look off

CHALLENGE



$$C_l^{\text{tot}} = \sum_{i,j} C_l^{ij}$$

IMPLEMENTATION

GETTING STARTED

Installation:

```
$ pip install glass
```

For spectra from CAMB — but easy to use your own:

```
$ pip install glass-camb
```

Then jump right in with the examples:

glass.readthedocs.io/projects/examples/

A FIRST SIMULATION

Simulate the matter fields up to redshift 1 in 10 shells

A FIRST SIMULATION

```
1 # make a redshift grid with 10 shells up to redshift 1
2 zb = glass.shells.redshift_grid(0., 1., num=10.)
3
4 # make tophat windows for shells
5 ws = glass.shells.tophat_windows(zb)
6
7 # compute or load the angular power spectra for the shells
8 cls = ...
9
10 # compute Gaussian cls for lognormal fields for 3 correlated shells
11 gls = glass.fields.lognormal_gls(cls, nside=nside, lmax=lmax, ncorr=3)
12
13 # generator for lognormal matter fields
14 matter = glass.fields.generate_lognormal(gls, nside, ncorr=3)
15
16 # simulate each matter shell
17 for i, delta_i in enumerate(matter):
18
19     # work with each matter field delta_i
20     ...
```

ADD LENSING

ADD LENSING

```
13 # generator for lognormal matter fields
14 matter = glass.fields.generate_lognormal(gls, nside, ncorr=3)
15
16 # this will compute the convergence field iteratively
17 convergence = glass.lensing.MultiPlaneConvergence(cosmo)
18
19 # simulate each matter shell
20 for i, delta_i in enumerate(matter):
21
22     # add lensing plane from the window function of this shell
23     convergence.add_window(delta_i, ws[i])
24
25     # get convergence field
26     kappa_i = convergence.kappa
27
28     # compute shear field
29     gamm1_i, gamm2_i = glass.lensing.shear_from_convergence(kappa_i)
30
31     ...
```

ADD GALAXIES

ADD GALAXIES

```
# standard deviation in each component of galaxy ellipticity
sigma_e = 0.256

# use a fixed galaxy bias parameter here
galaxy_bias = 1.8

# galaxy number density per arcmin2, over all shells
n_arcmin2 = 0.1

# Smail redshift distribution with the given density
z = np.arange(0., 2., 0.01)
dndz = glass.observations.smail_nz(z, 1.0, 2.0, 1.5, norm=n_arcmin2)
```


SEE EXAMPLES FOR MORE!

EXTENDING *GLASS*

Ad hoc — just use your own code!

Extensions — there is first class support for extensions, which are on equal footing to the core library

We decided to have an open architecture where extensions are independent projects by independent people

If you want to create an extension: clone the template repository and go — it's all yours

But do consider to **get in touch** for coordination and help!

A FEW ONGOING DEVELOPMENTS

- Window functions for better accuracy with fewer shells
- Is constrained simulation of 3D clustering possible?
- Galaxies via halos (MSci project, Lea Harscouet, UCL)
- Accelerated and differentiable *GLASS*

CONCLUSIONS

GLASS ...

uses novel techniques to achieve better than per cent-level accuracy for simulated photometric galaxy surveys

is easy to use in your projects, both new and existing

is easy to extend and interface with your own code

has a lot of room for growth