Updates on the **forward modeling** of **$f\sigma_8$** with **ZTF**

Aurélien Valade,

B. Racine, D. Fouchez, J. Bautista,

D. Rosseli, F. Feinstein, B. Sanchez & B. Carreres

CPPM, Marseille

ZTF Cosmo Barcelona, 11/12/2024

Who is $f\sigma_8$?

→ → Linear theory! ← ←

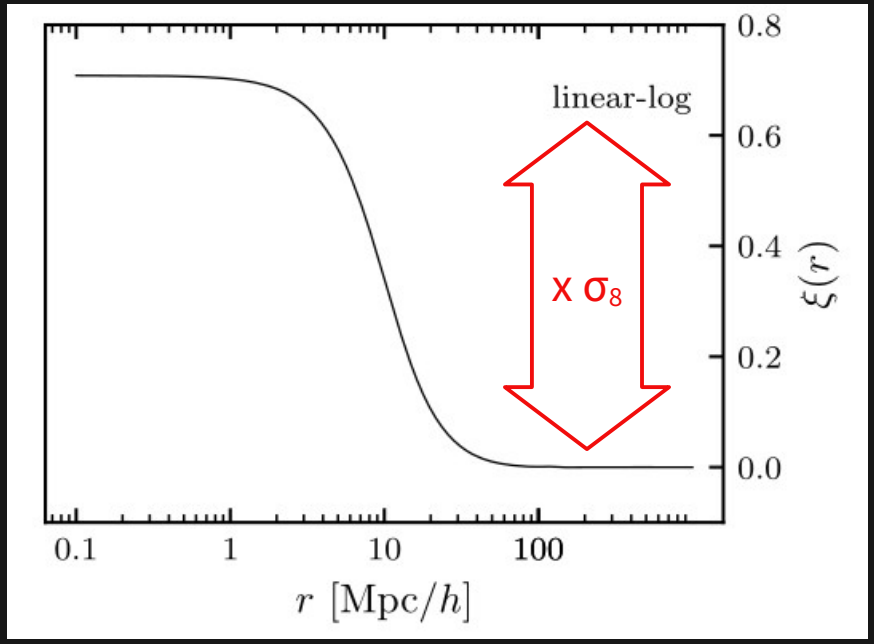
$$\vec{\nabla} \cdot \vec{v} = -H_0 (f\sigma_8) \delta_{\sigma_8 = 1}$$

Who is $f\sigma_8$?

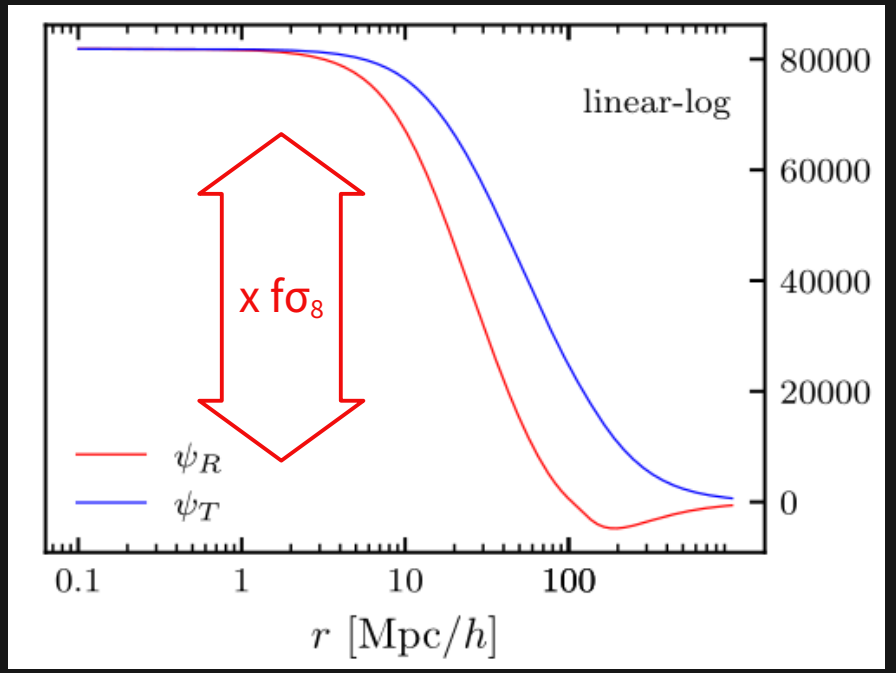
→ → **Linear theory!** ← ←

$$\vec{\nabla} \cdot \vec{v} = -H_0 (f\sigma_8) \delta_{\sigma_8 = 1}$$

Density correlation function



Velocity correlation functions

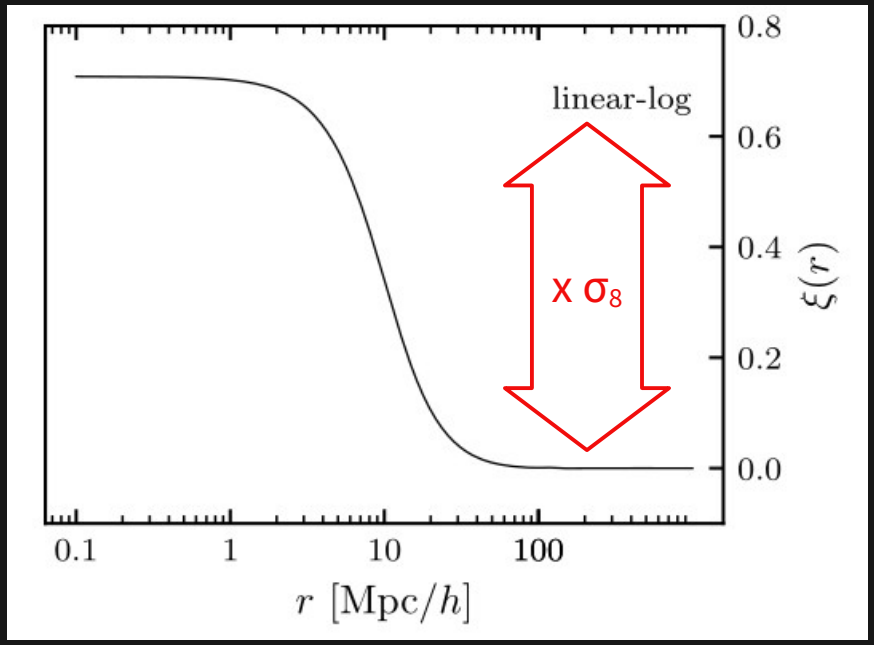


Who is $f\sigma_8$?

→ → **Linear theory!** ← ←

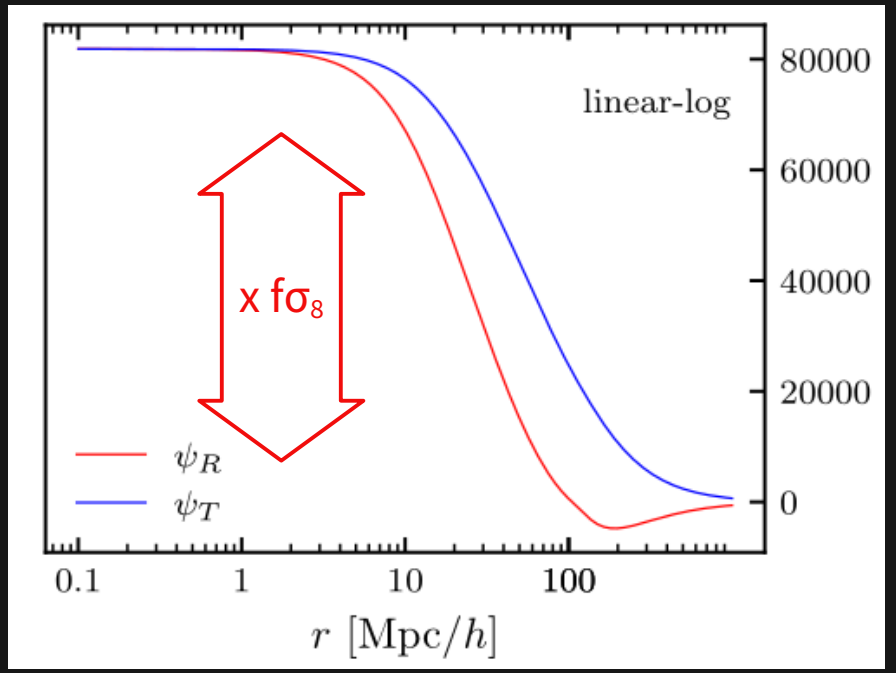
$$\vec{\nabla} \cdot \vec{v} = -H_0 (f\sigma_8) \delta_{\sigma_8 = 1}$$

Density correlation function



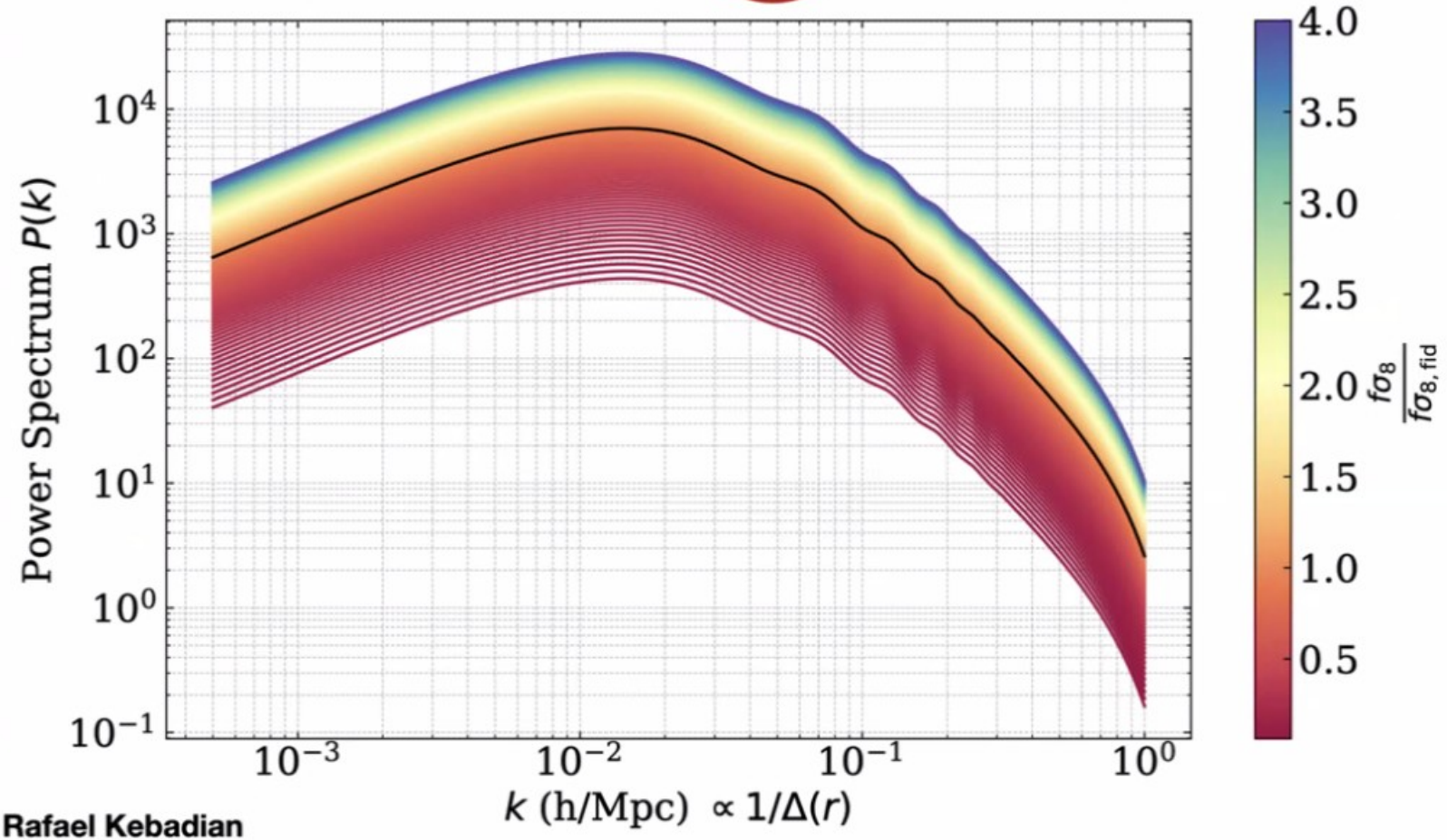
→ **Shape of the functions fixed**

Velocity correlation functions



→ **degeneracy between f and σ_8**

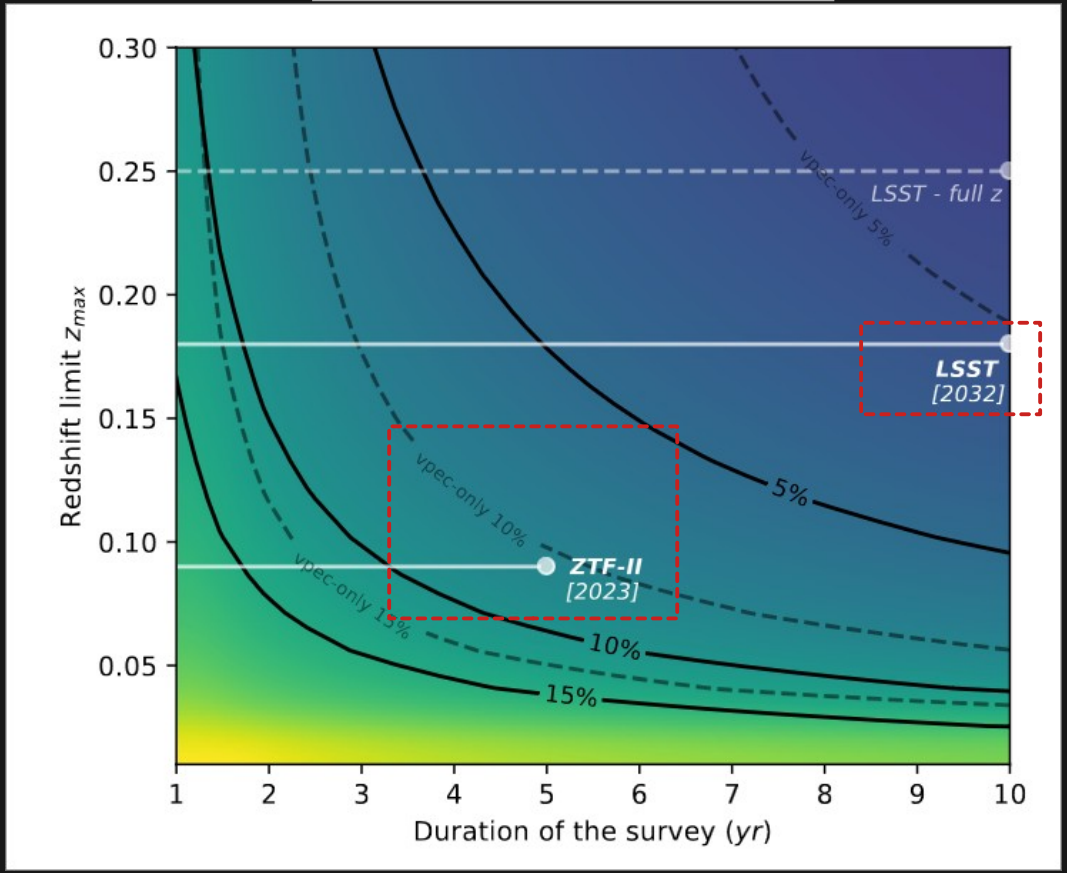
$$C_{ij}^{vv} = \frac{H_0^2 \left(f\sigma_8\right)^2}{2\pi^2 \left(f\sigma_8\right)^2_{\text{fid}}} \int_0^{+\infty} f_{\text{fid}}^2 P_{\theta\theta}(k) D_u^2(k) W_{ij}(k; r_i, r_j)\, \mathrm{d}k$$



Rafael Kebadian

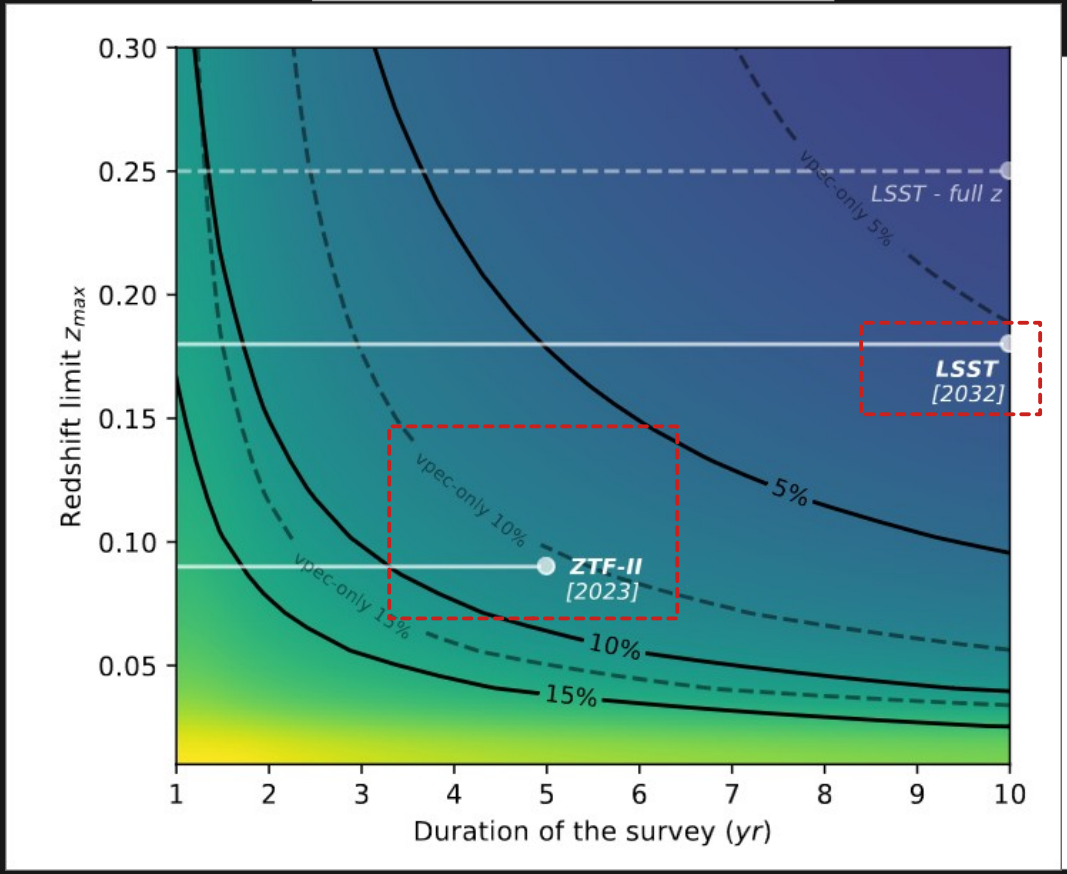
Prospects on $f\sigma_8$

Fisher Matrix (Graziani 2020)



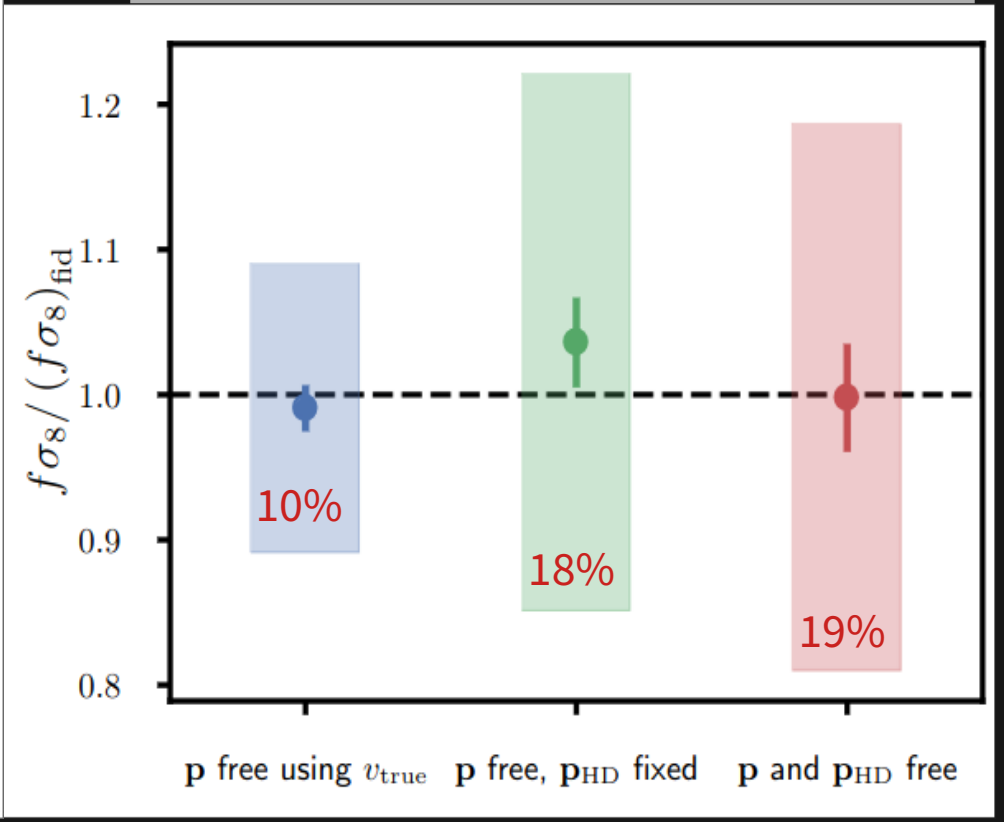
Prospects on $f\sigma_8$

Fisher Matrix (Graziani 2020)



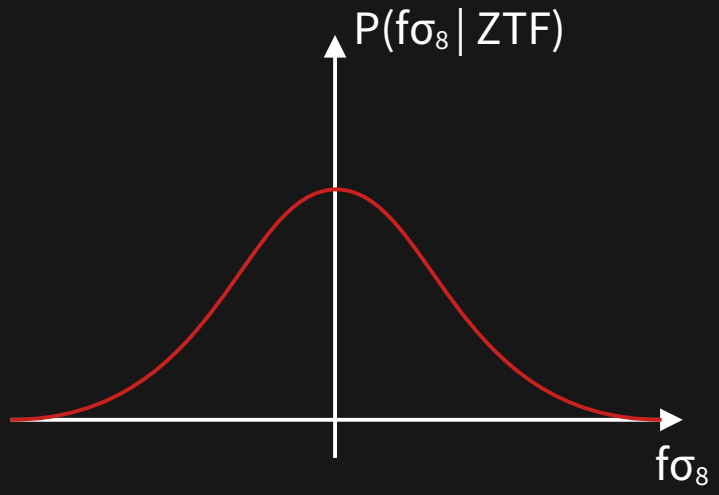
Likelihood maximization (Carreres 2023)

- 6 years
- $z < 0.06$ (complete sample)



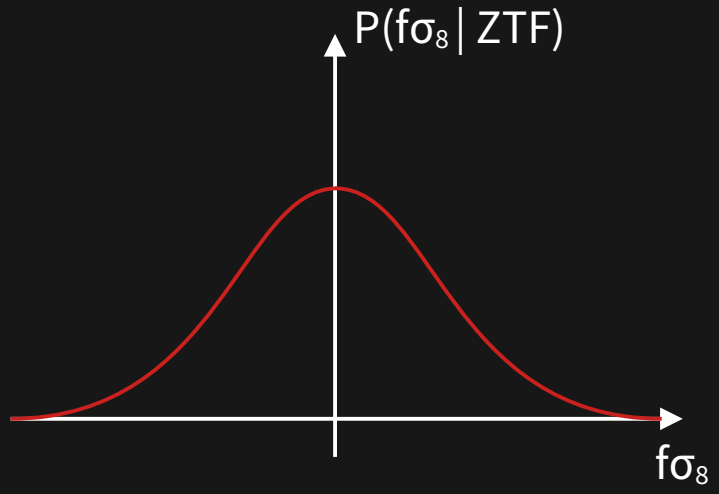
Forward modeling

$$P(f\sigma_8, \ldots | \mathrm{ZTF})$$



Forward modeling

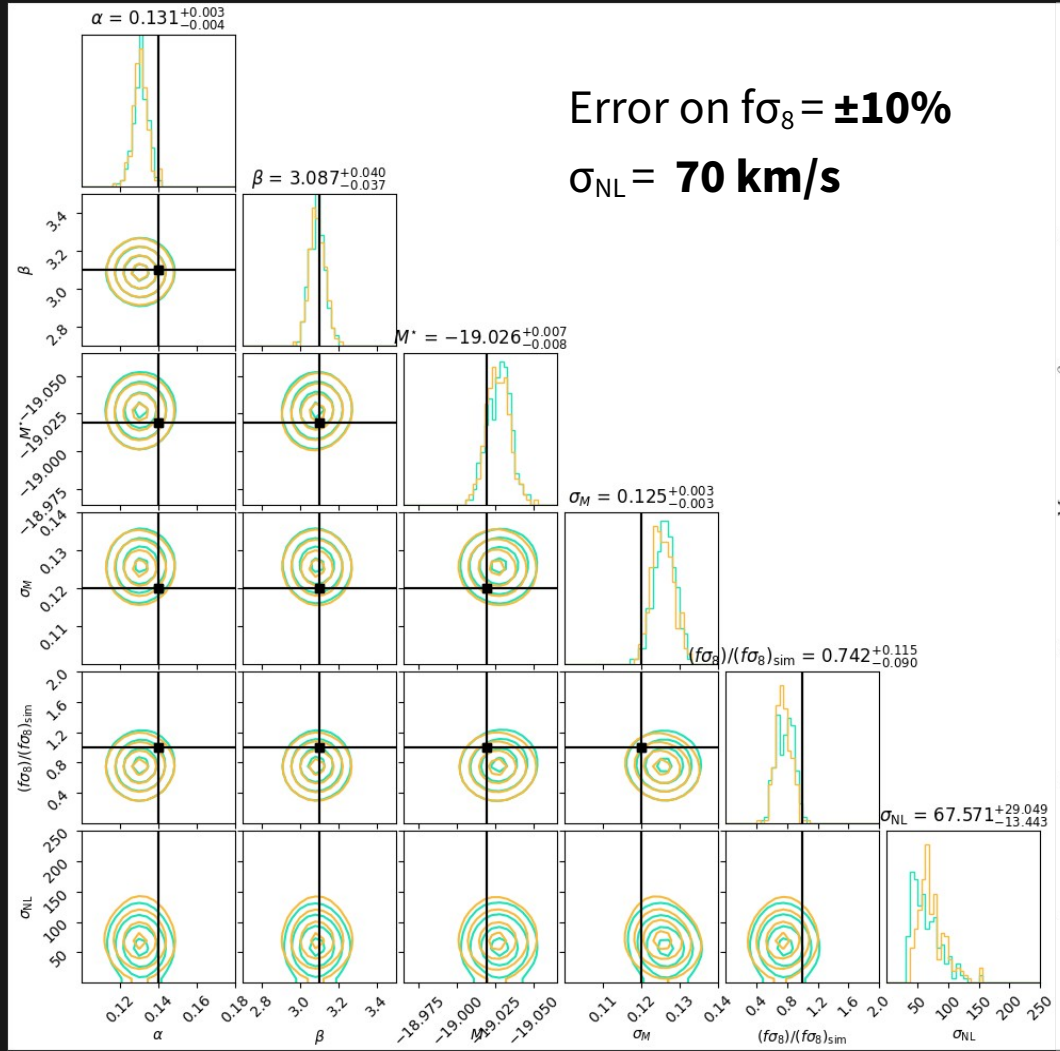
$$P(f\sigma_8, \ldots | \mathrm{ZTF}) = P(\mathrm{ZTF} | f\sigma_8, \ldots) P(f\sigma_8, \ldots)$$



What is HAMLET and what does it do?

Demo Hamlet Movie

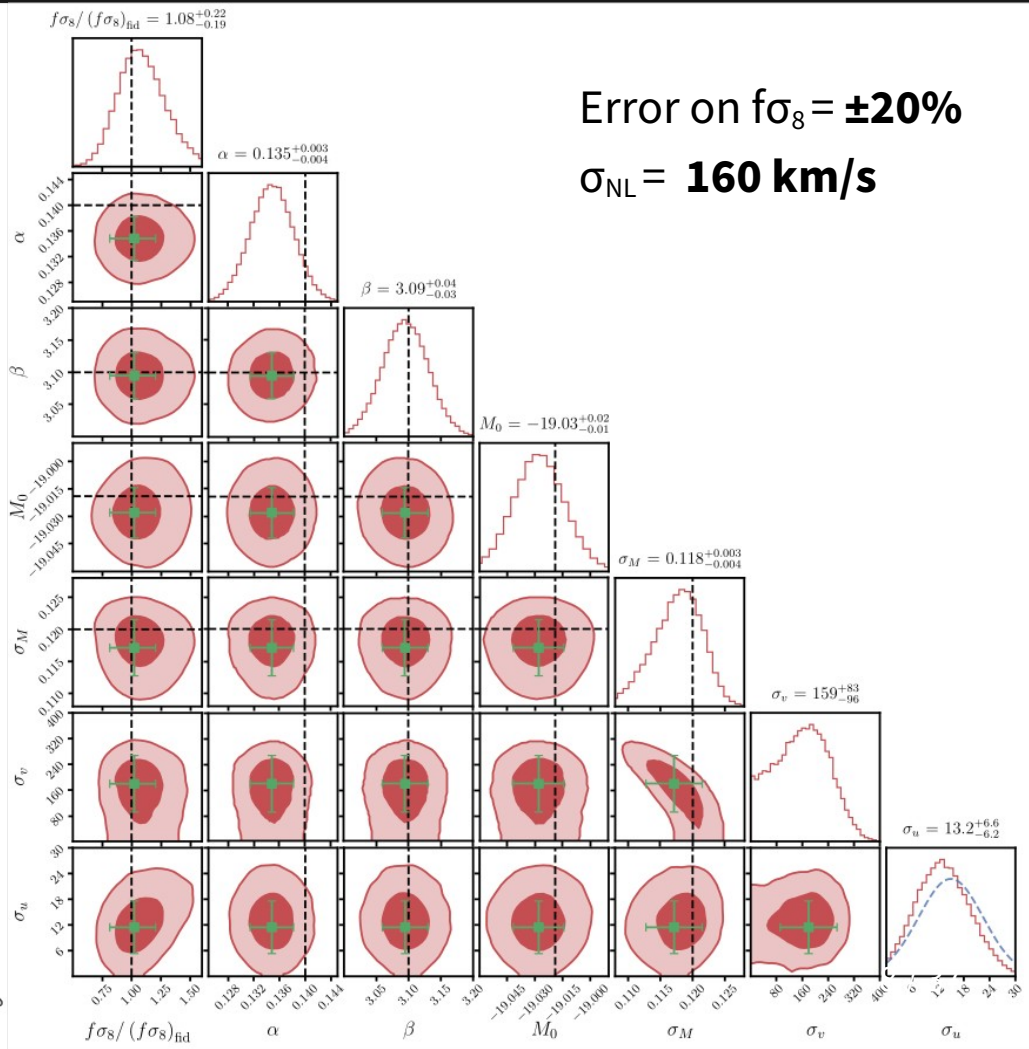
This work (1 mock)



Error on $f\sigma_8$ = **±10%**

σ_{NL} = **70 km/s**

Carreres 2023 (1 mock)

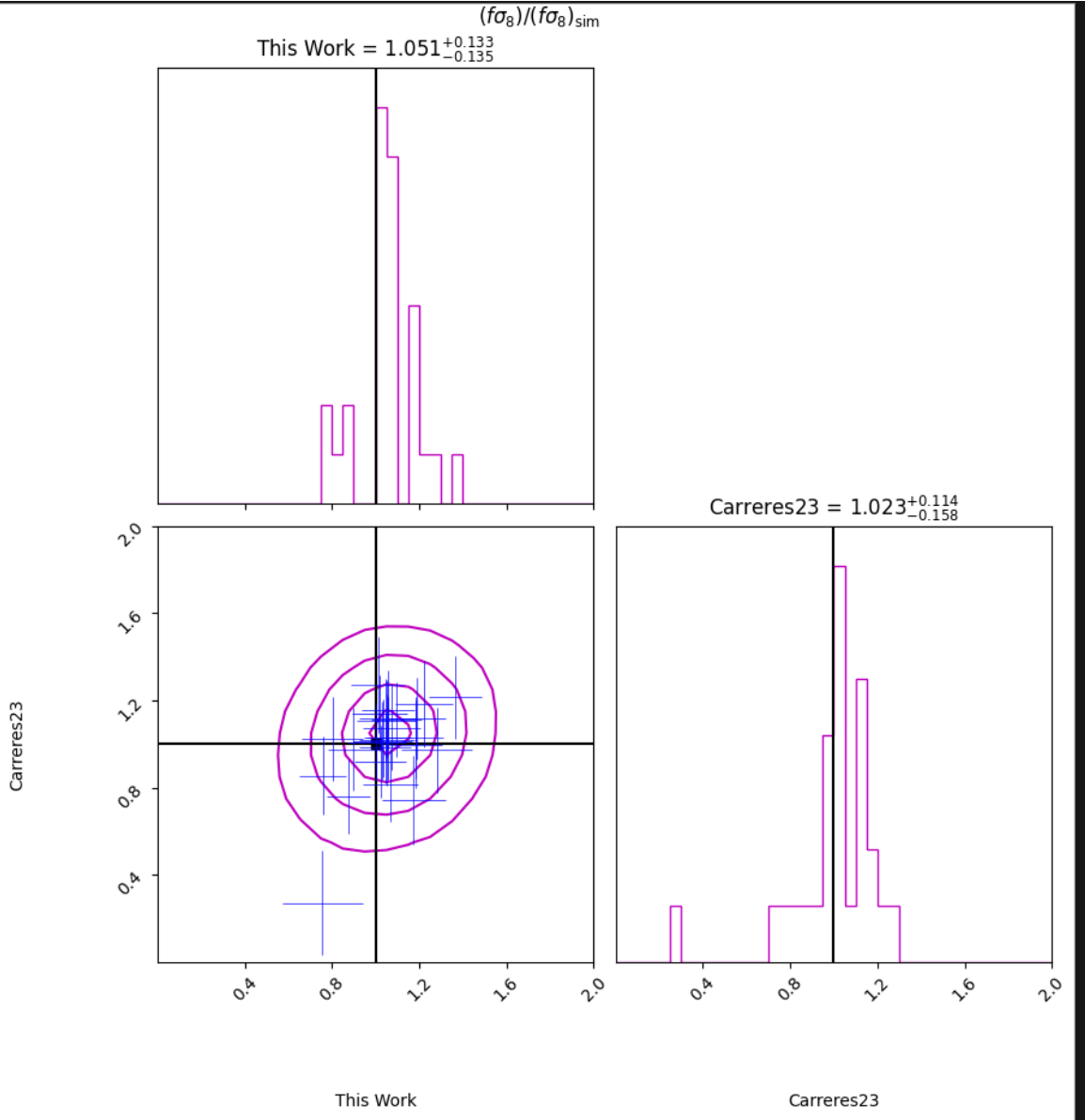


Error on $f\sigma_8$ = **±20%**

σ_{NL} = **160 km/s**

Values $(f\sigma_8)$ / $(f\sigma_8)_{sim}$

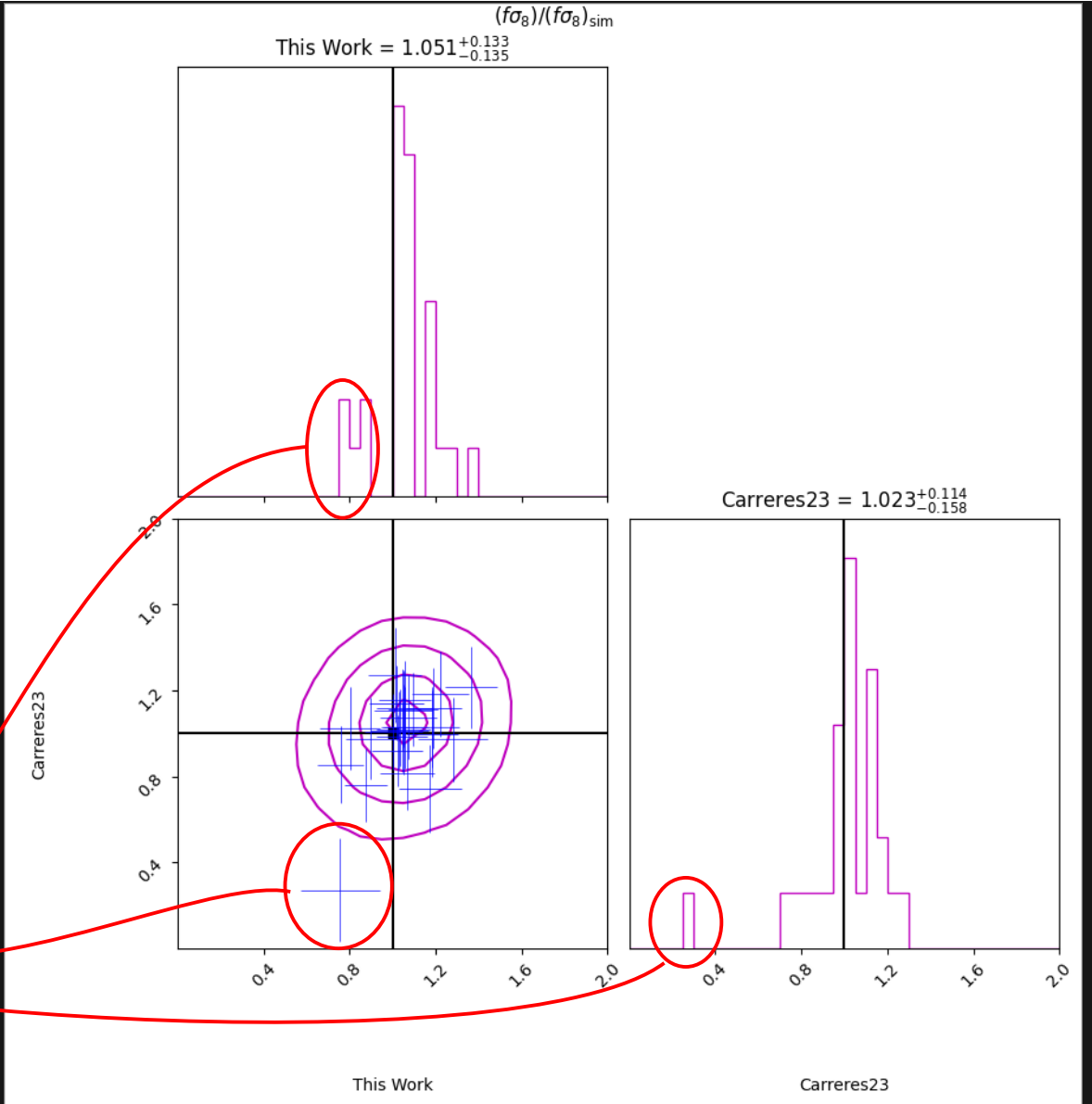
$$\rho_{\mathrm{C23}\leftrightarrow\mathrm{TW}} = 0.5\ (0.33)$$



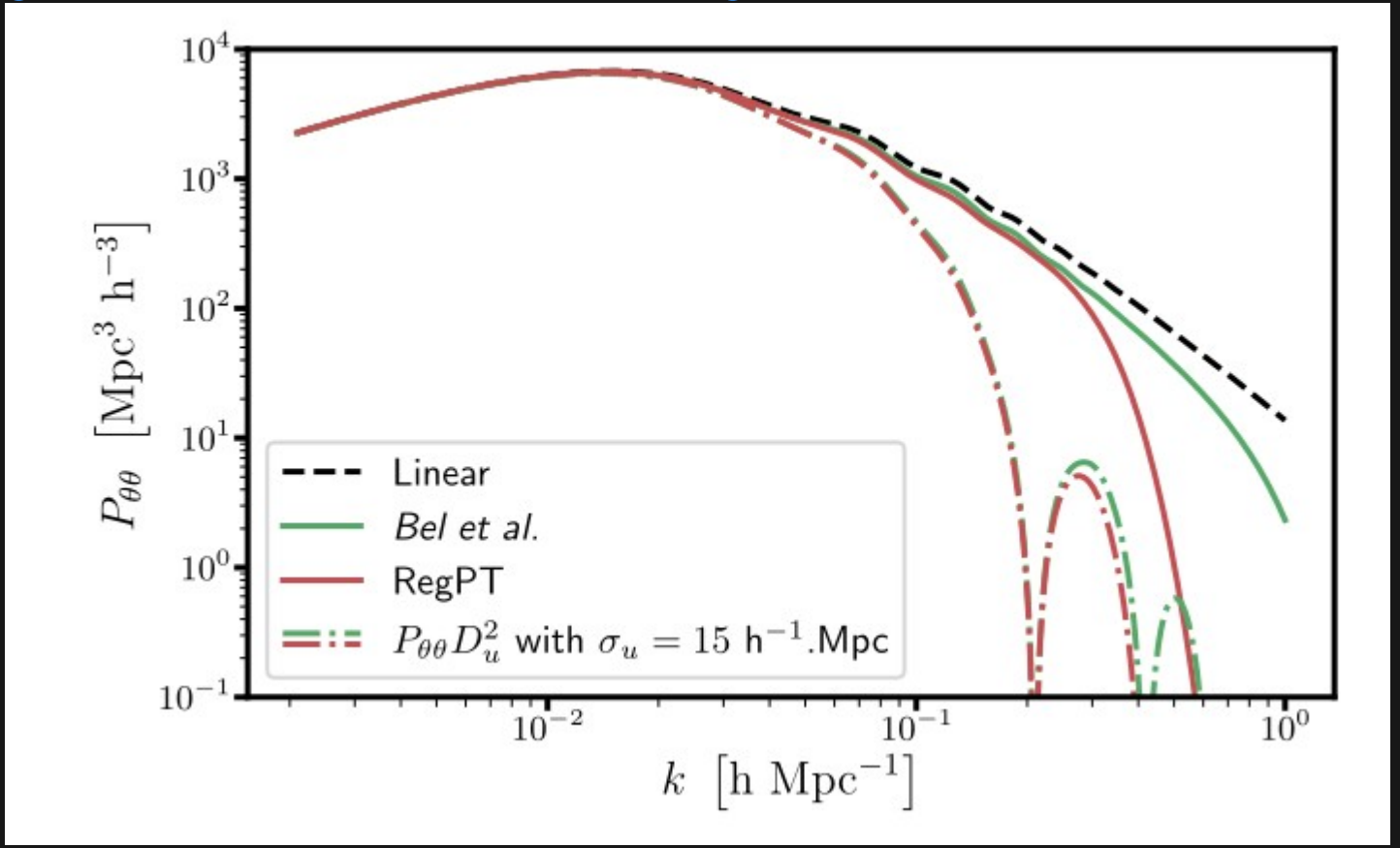
Values $(f\sigma_8)$ / $(f\sigma_8)_{sim}$

$$\rho_{\mathrm{C23}\leftrightarrow\mathrm{TW}} = 0.5 \ (0.33)$$

Remove outlier (number 18)



Go beyond linear theory?



Carreres 2023

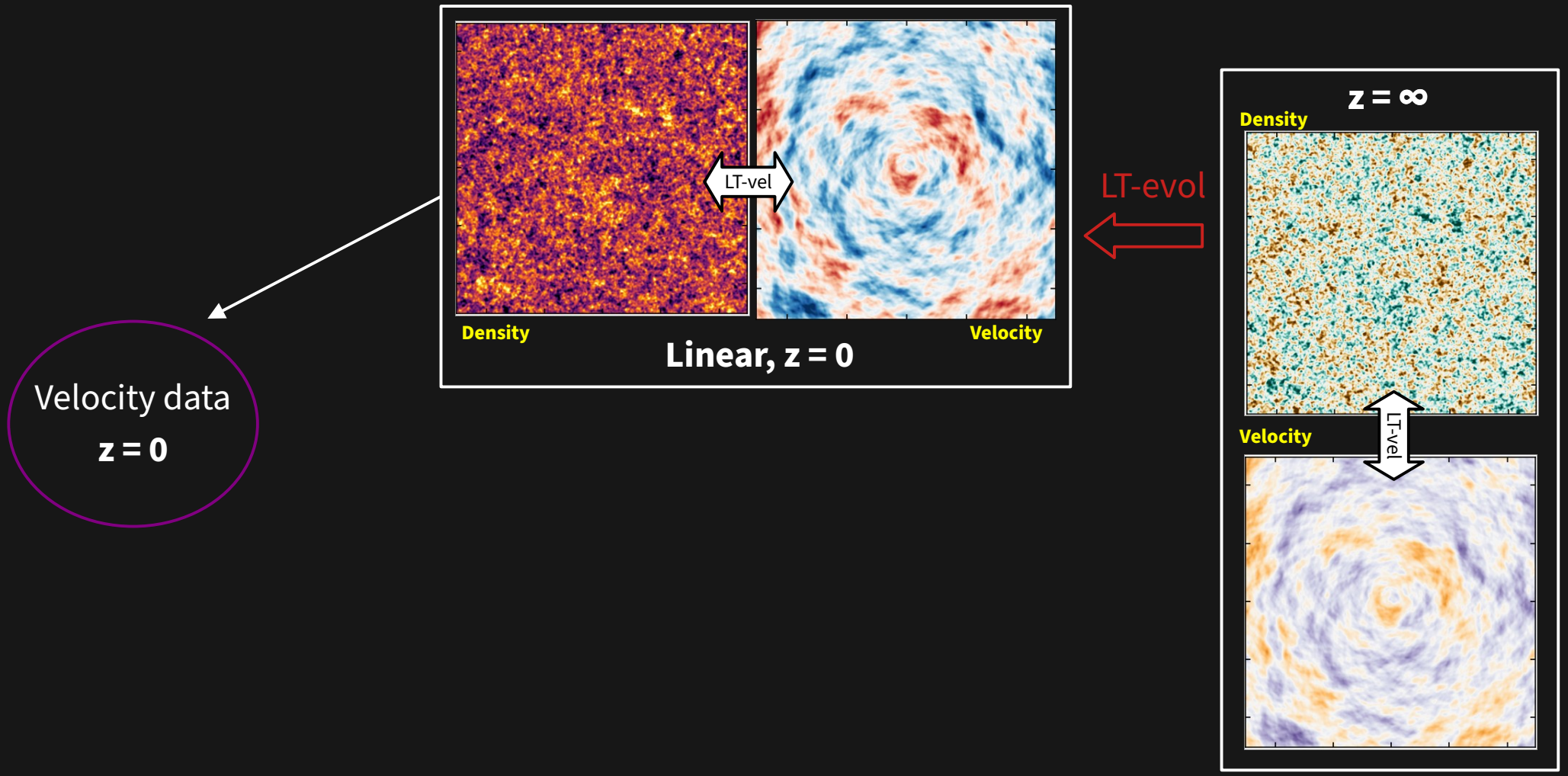
Linear Theory

Frozen stucture, amplification with time

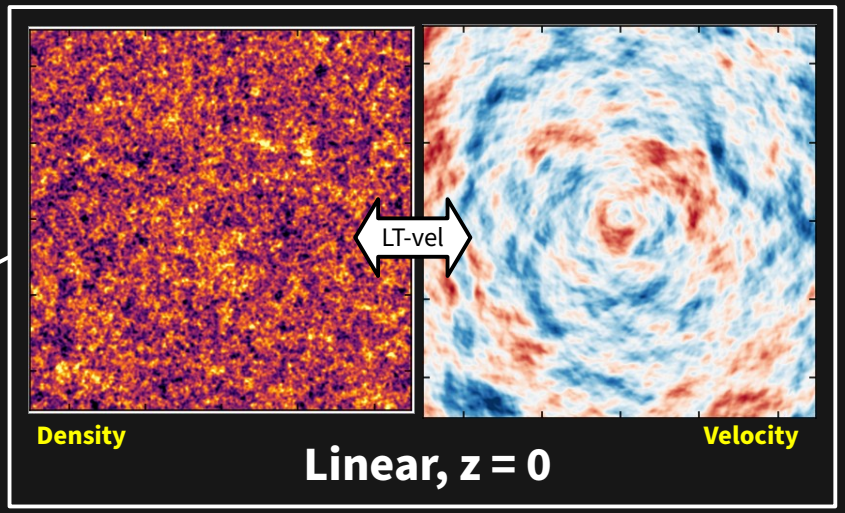
$$\delta(\vec{r}, a) = D_+(a)\delta(\vec{r}) \qquad \text{(LT-evol)}$$

$$\vec{\nabla} \cdot \vec{v}(\vec{r}) = -H_0 f \delta(\vec{r}) \qquad \text{(LT-vel)}$$

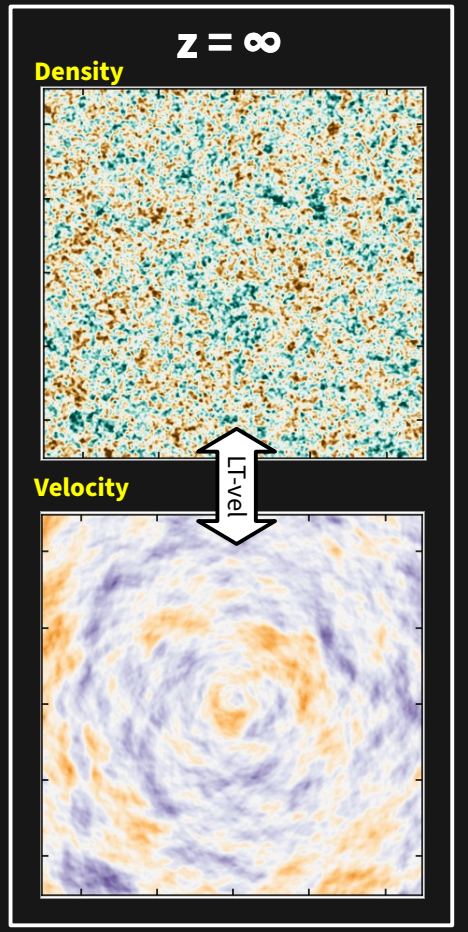
Velocity-density fields conservation-like equation

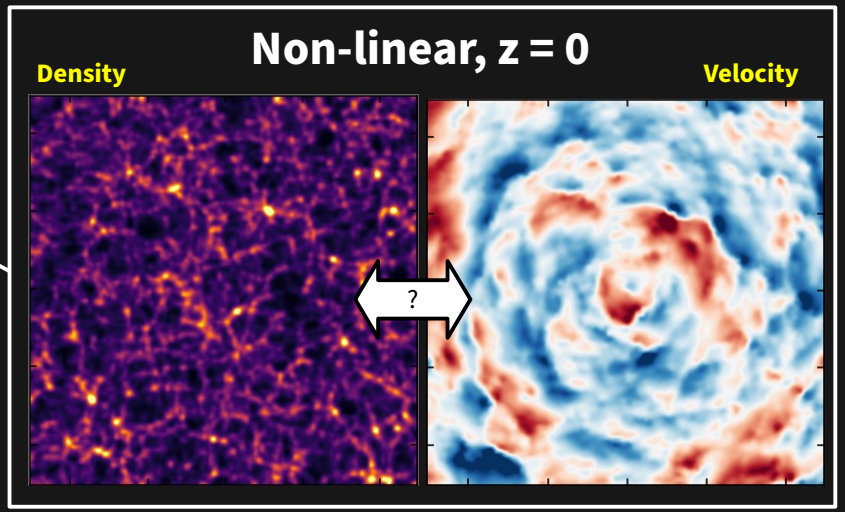

Velocity data
z = 0
Density
LT-vel
Velocity
Linear, z = 0
LT-evol
z = ∞
Density
LT-vel
Velocity

Velocity data
z = 0



LT-evol





N-body
LPT

Non-linear code requirements

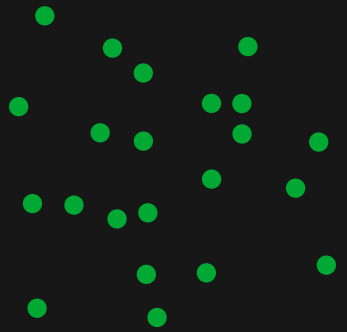
- **Cheap** → evaluated about 1 000 000 times
- **Differentiable** → for the inference algorithm
- **Low resolution** → limit number of free parameters
- **GPU compatible** → inference speed-up about 1000

particle mesh with derivatives

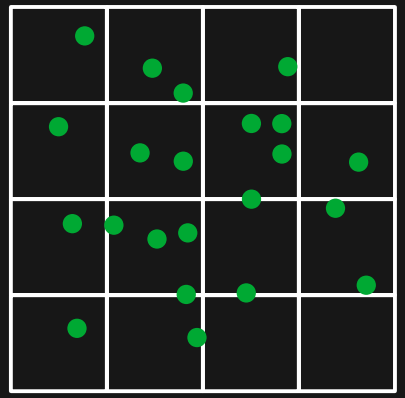
`pmwd` is a differentiable cosmological particle-mesh forward model. The C_2 symmetry of the name symbolizes the reversibility of the model, which helps to dramatically reduce the memory cost when used with the adjoint method. Based on `JAX`, pmwd is fully differentiable, and is highly performant on GPUs.

Particle mesh

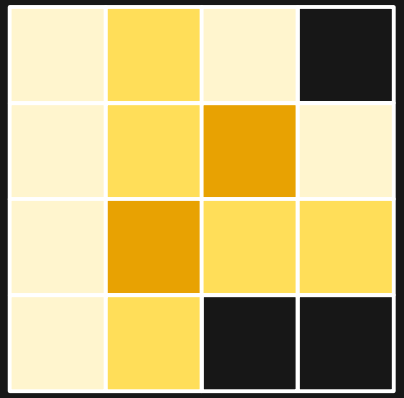
Particles



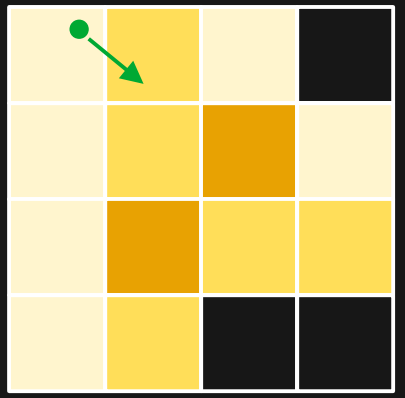
Lattice



Cloud in Cell (CiC)



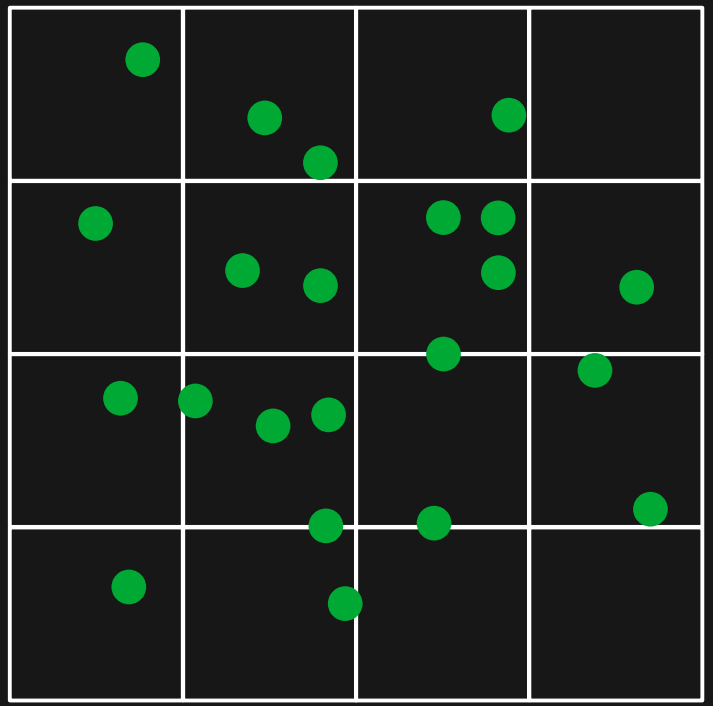
Gravitational potential



(+) fast, smoothly differentiable

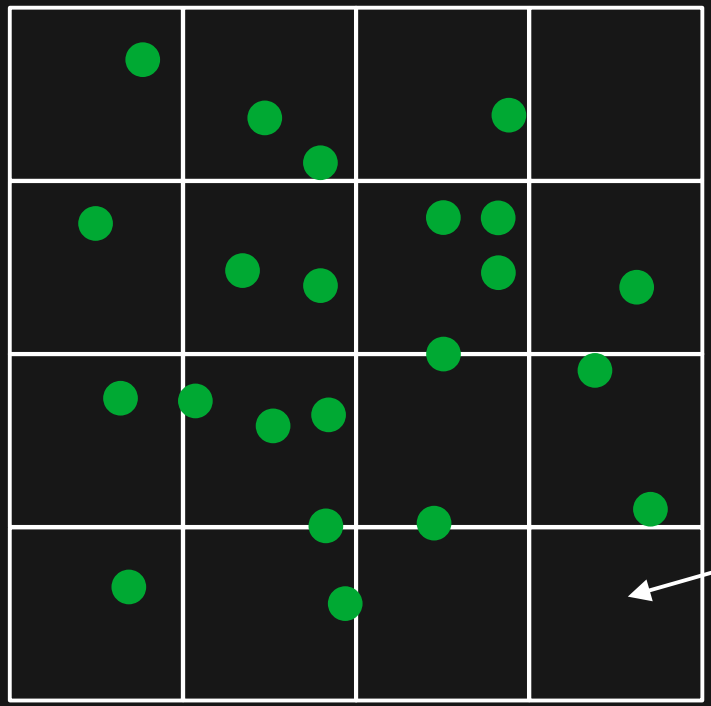
(--) inaccurate below cell size

Computing the velocity field



$\rho = \text{n particles in cell}$

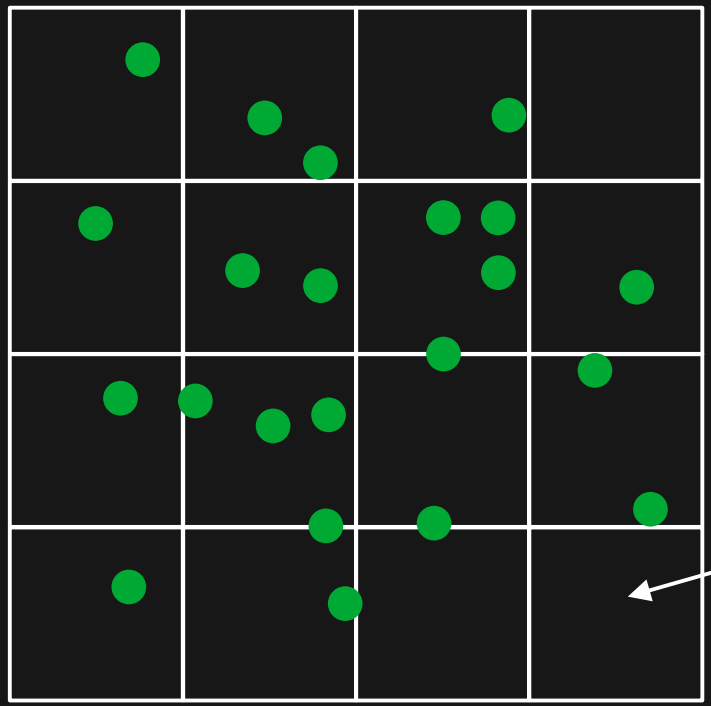
Computing the velocity field



$\rho =$ n particles in cell

$\rho < 1$ particle / cell

Computing the velocity field

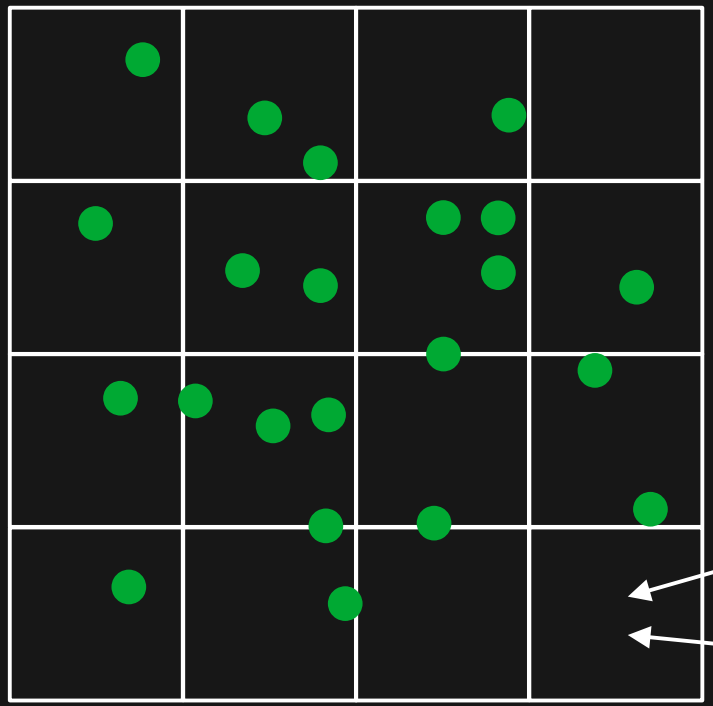


$$\rho = \text{n particles in cell}$$

$$\vec{v} = \frac{\text{sum velocities in cell}}{\text{n particles in cell}}$$

$\rho < 1$ particle / cell

Computing the velocity field



$$\rho = \text{n particles in cell}$$

$$\vec{v} = \frac{\text{sum velocities in cell}}{\text{n particles in cell}}$$

$\rho < 1$ particle / cell

$$\vec{v} = \frac{0}{0} = \text{ill defined}$$

+ non-differentiable!

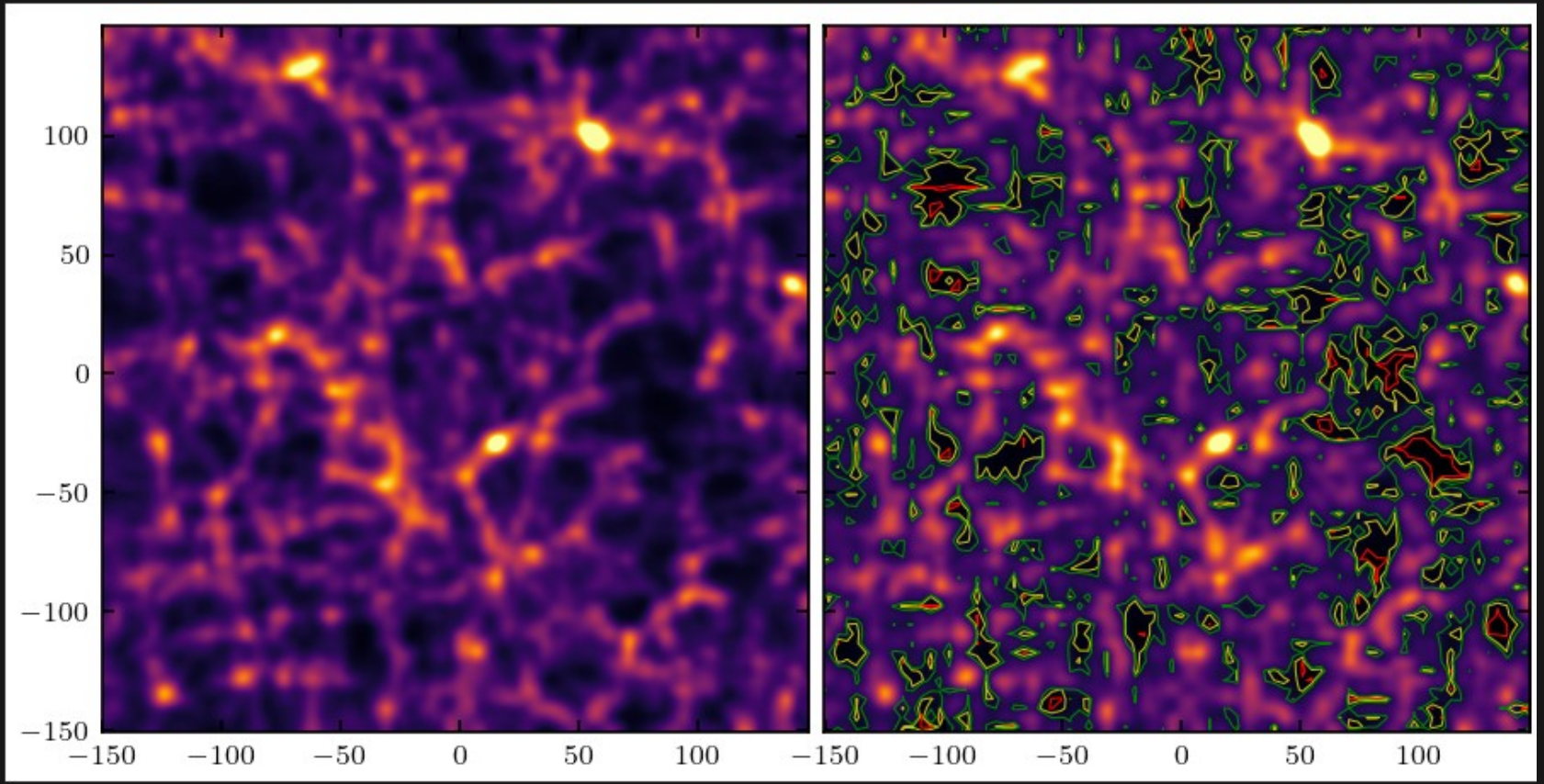
Empty cells

Cells with less than

2 particles 1 particle 0.1 particle

256^3 particles, 128^3 cells

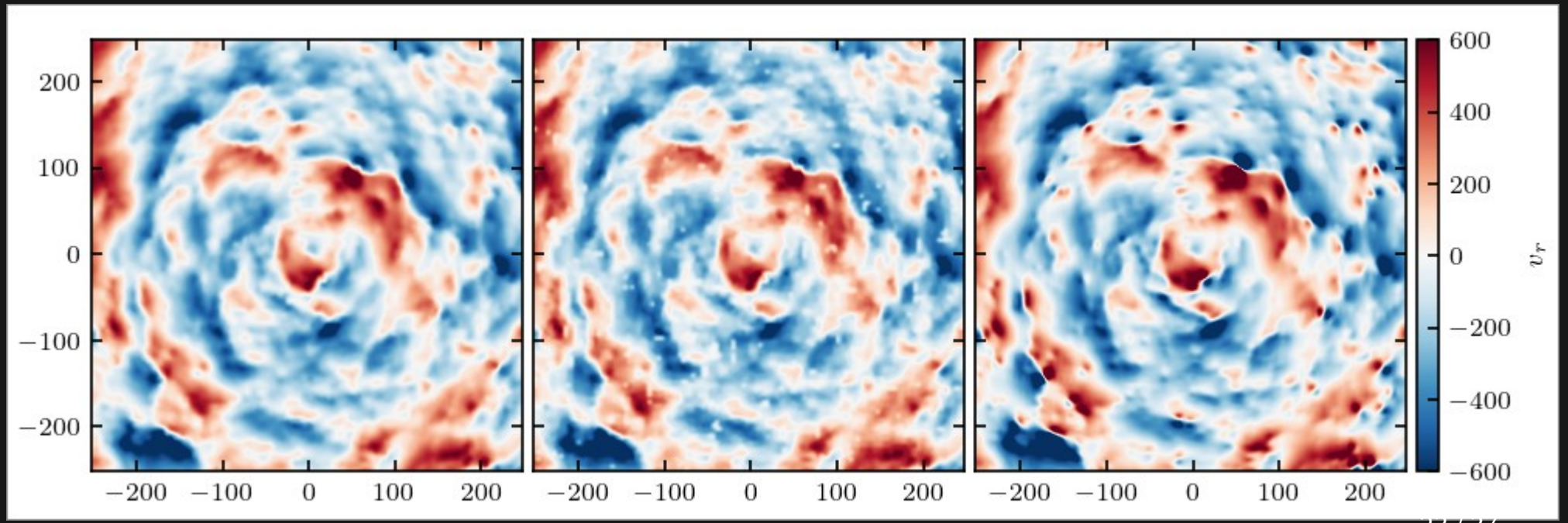
128^3 particles, 128^3 cells



Use linear theory

$$\cdot\vec{v}(\vec{r}) = -H_0 f \delta(\vec{r})$$

High-res (reference) Low-res Low-res with v(δ) ← linear theory



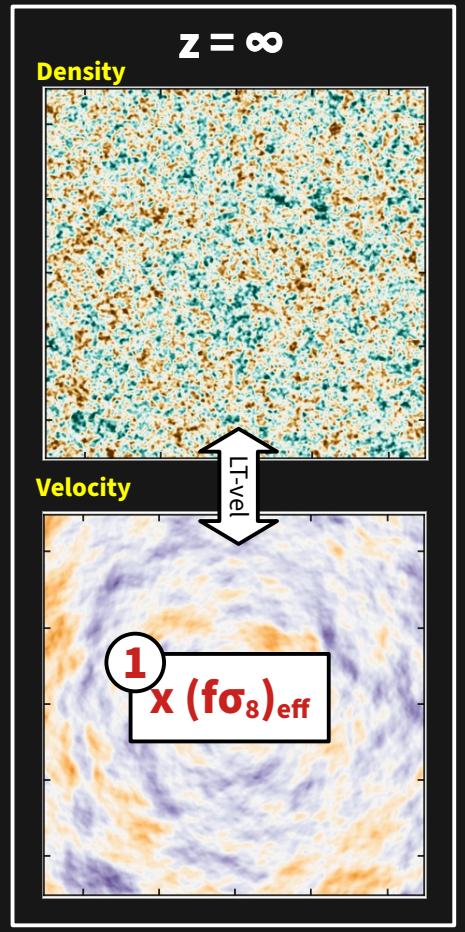
Example of run on Cosmicflows 4

demo non linear

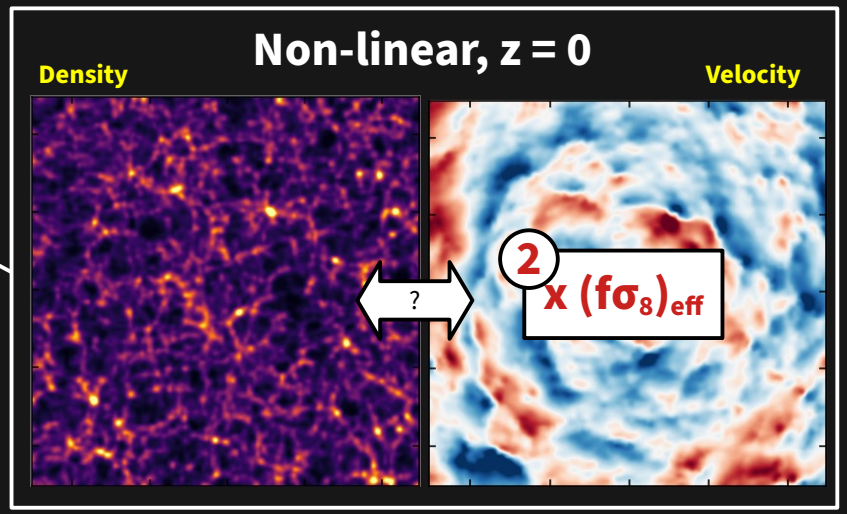
What about $f\sigma_8$?

Velocity data

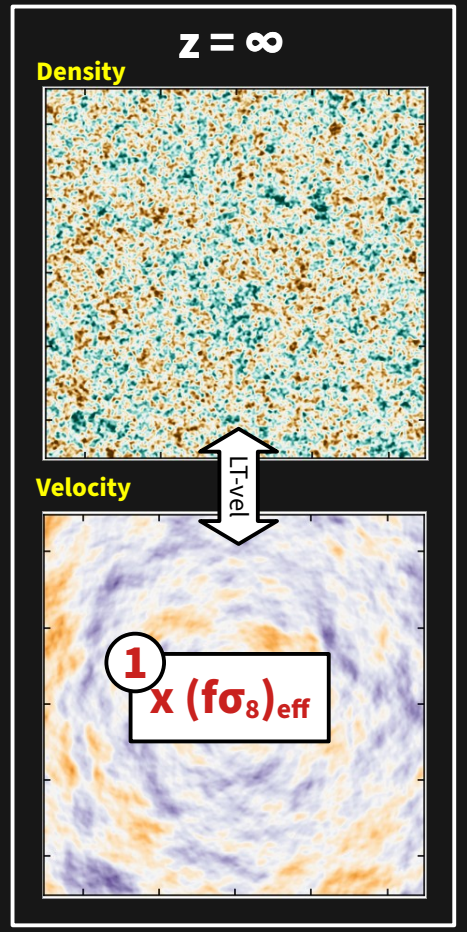
z = 0

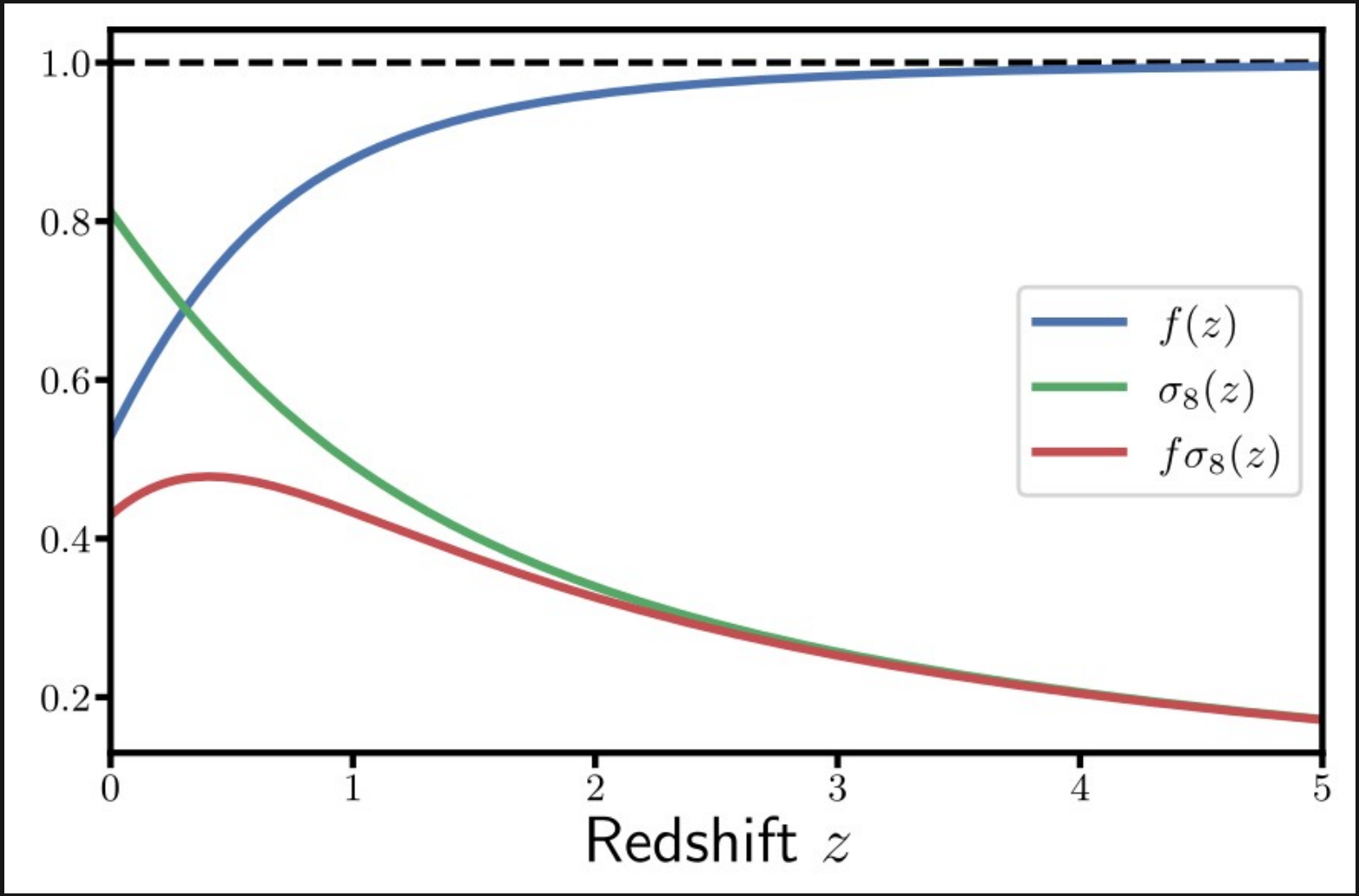


Velocity data
z = 0



N-body
LPT

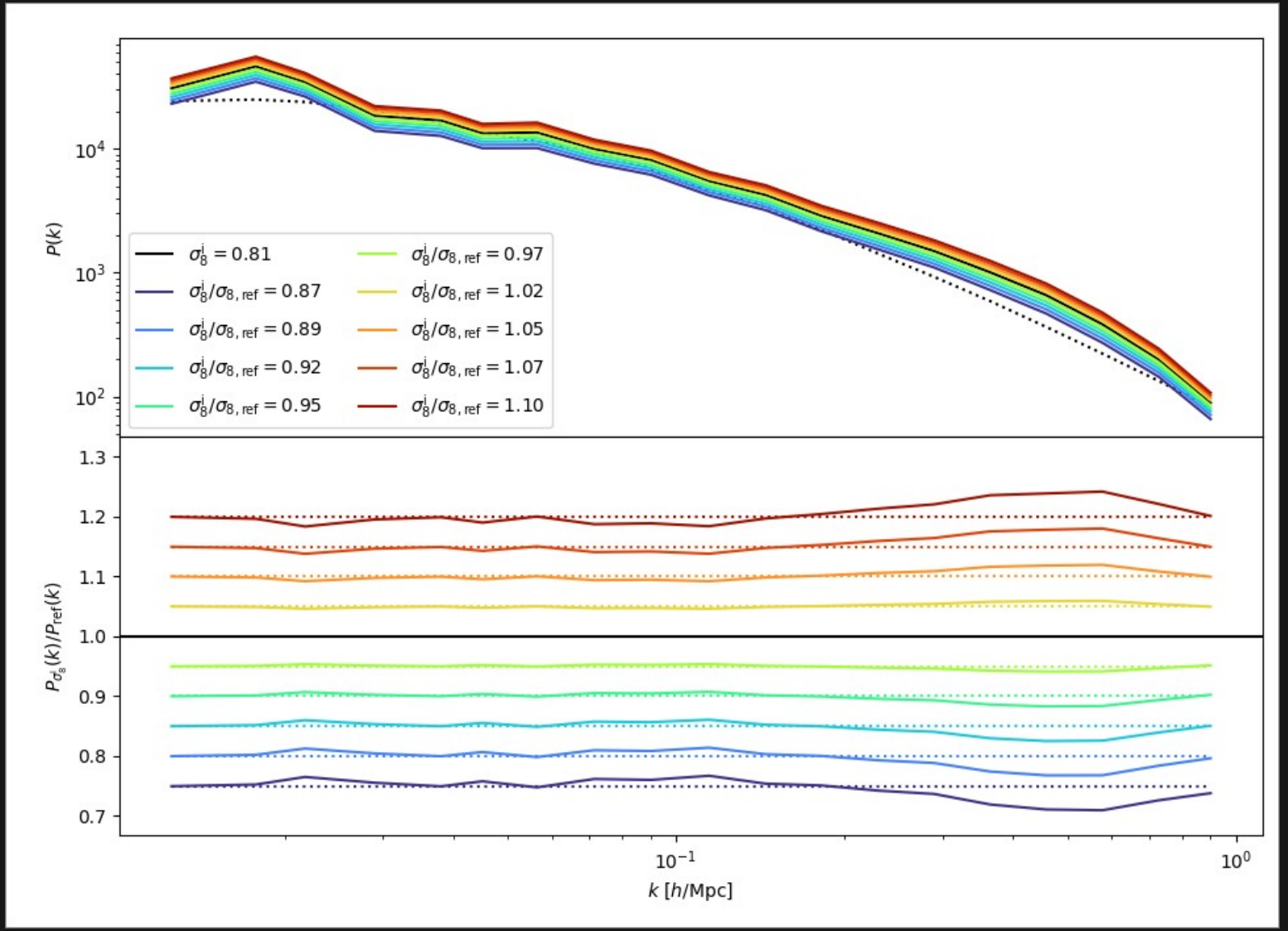




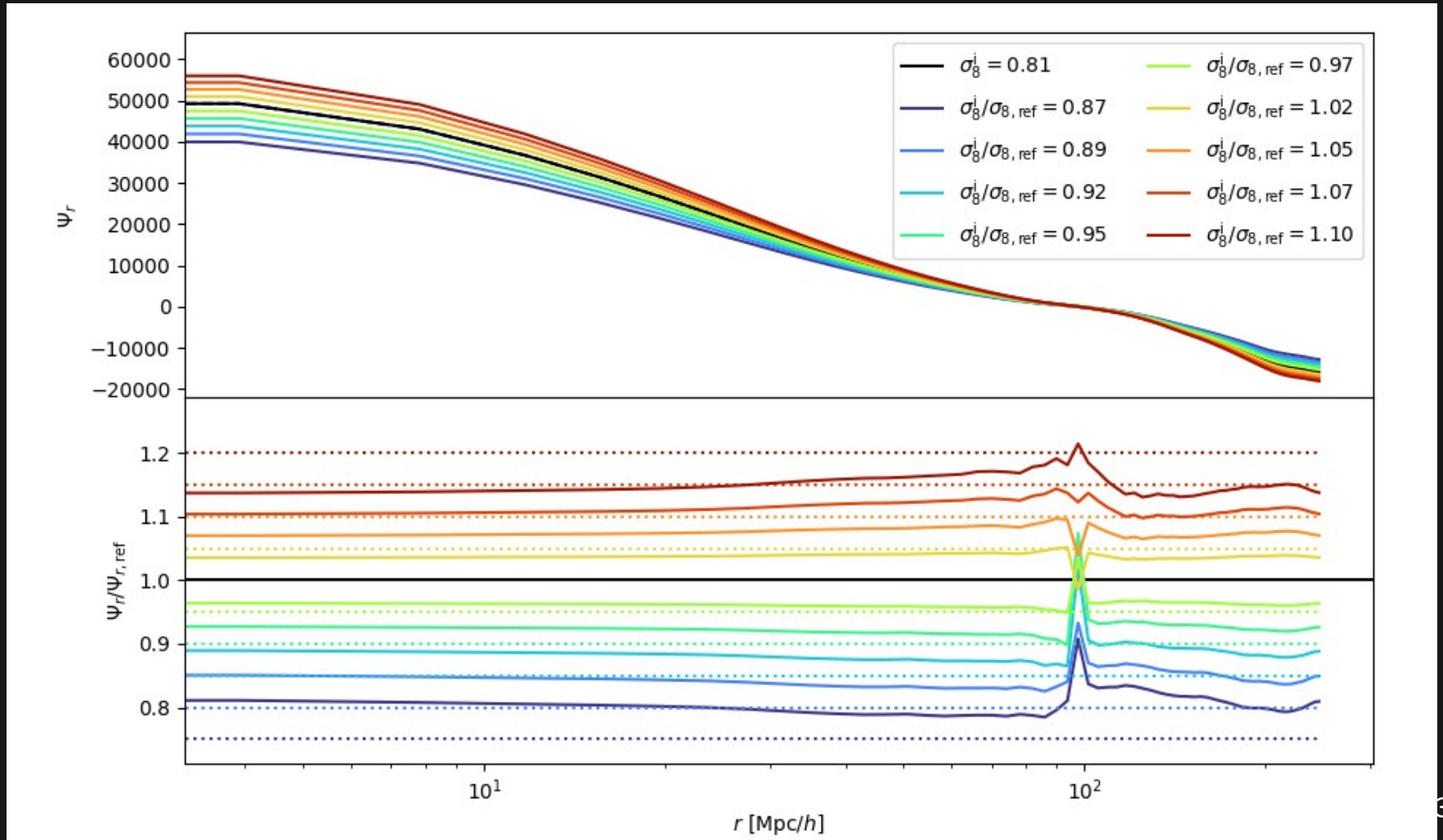
Still linear theory!

Full lines → case (1)

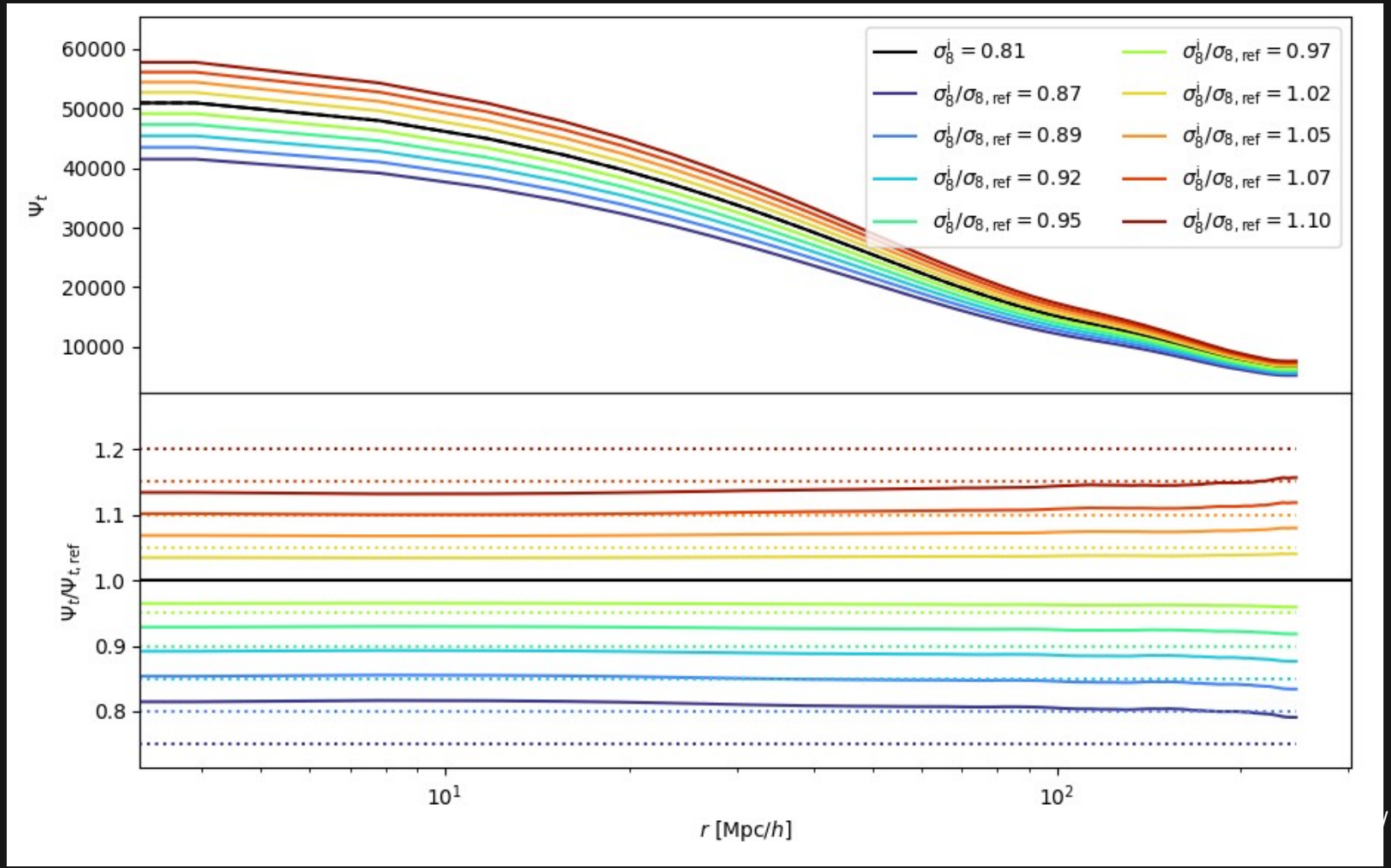
Dotted lines → case (2)



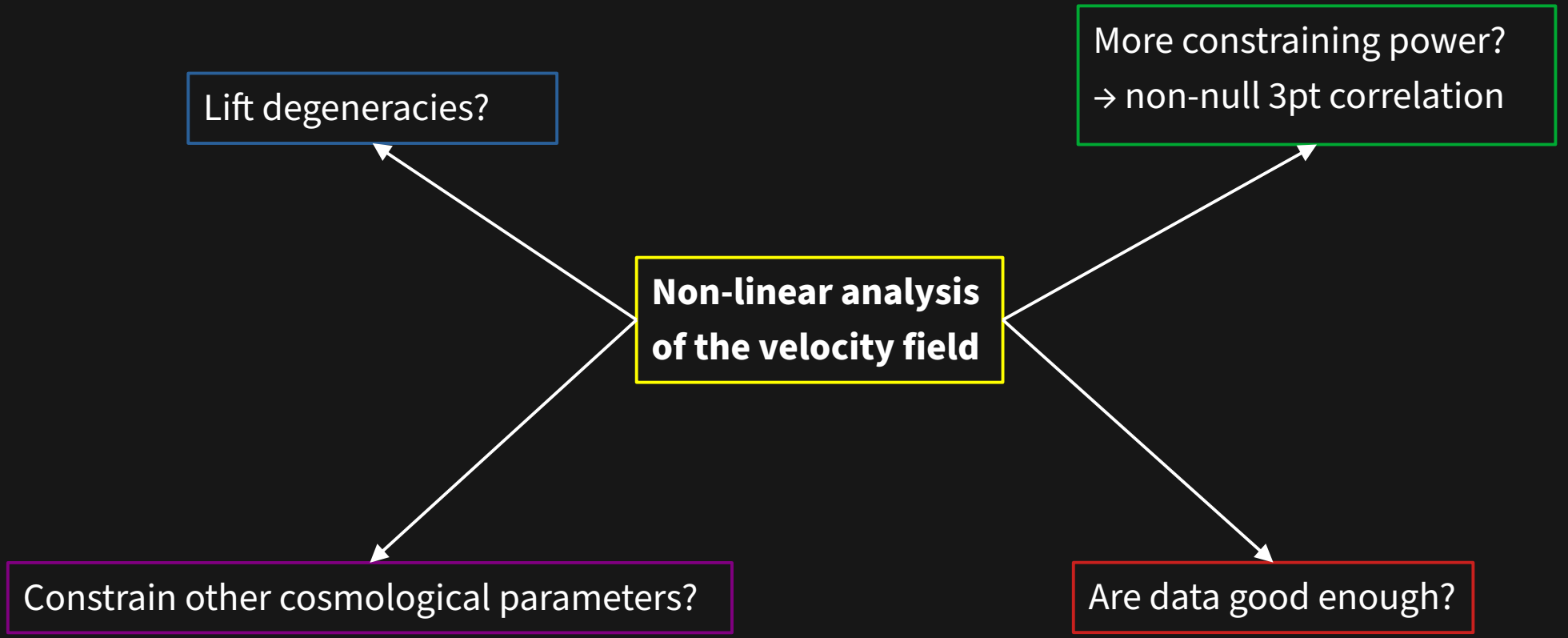
Velocity correlation function (radial)



Velocity correlation function (transversal)



Open questions



Skysurvey + pmwd = mocks from evolved random universes

1 → initialize (once)

```
EU = EvolvedUniverse(
    pmwd_cosmo_kwargs={},
    pmwd_conf_kwargs=dict(
        ptcl_grid_shape=(n_ptcl_per_side,) * 3,
        ptcl_spacing=L_box / n_ptcl_per_side,
        mesh_shape=mesh_shape,
        a_start=1 if just_2lpt else 1 / 64,
    ),
)
```

Skysurvey + pmwd = mocks from evolved random universes

1 → initialize (once)

```
EU = EvolvedUniverse(
    pmwd_cosmo_kwargs={},
    pmwd_conf_kwargs=dict(
        ptcl_grid_shape=(n_ptcl_per_side,) * 3,
        ptcl_spacing=L_box / n_ptcl_per_side,
        mesh_shape=mesh_shape,
        a_start=1 if just_2lpt else 1 / 64,
    ),
)
```

2 → run (on GPU)

```
ptcl = EU.run_PM(seed=0)
```

Skysurvey + pmwd = mocks from evolved random universes

1 → initialize (once)

```
EU = EvolvedUniverse(
    pmwd_cosmo_kwargs={},
    pmwd_conf_kwargs=dict(
        ptcl_grid_shape=(n_ptcl_per_side,) * 3,
        ptcl_spacing=L_box / n_ptcl_per_side,
        mesh_shape=mesh_shape,
        a_start=1 if just_2lpt else 1 / 64,
    ),
)
```

2 → run (on GPU)

```
ptcl = EU.run_PM(seed=0)
```

3 → build halos (slow...)

```
halos_xyz, halos_v_xyz, halos_mass = EU.build_halos(
    ptcl_xyz,
    ptcl_v_xyz,
    linking_length=halos_linking_length,
)
```

Skysurvey + pmwd = mocks from evolved random universes

1 → initialize (once)

```
EU = EvolvedUniverse(
    pmwd_cosmo_kwargs={},
    pmwd_conf_kwargs=dict(
        ptcl_grid_shape=(n_ptcl_per_side,) * 3,
        ptcl_spacing=L_box / n_ptcl_per_side,
        mesh_shape=mesh_shape,
        a_start=1 if just_2lpt else 1 / 64,
    ),
)
```

2 → run (on GPU)

```
ptcl = EU.run_PM(seed=0)
```

3 → build halos (slow...)

```
halos_xyz, halos_v_xyz, halos_mass = EU.build_halos(
    ptcl_xyz,
    ptcl_v_xyz,
    linking_length=halos_linking_length,
)
```

4 → observe

```
ra, dec, z_cos, v_r = EU.observe(
    halos_xyz,
    halos_v_xyz,
    obs_xyz=obs_xyz,
    obs_v_xyz=obs_v_xyz,
    box_coords="celestial",
)
```

Skysurvey + pmwd = mocks from evolved random universes

1 → initialize (once)

```
EU = EvolvedUniverse(
    pmwd_cosmo_kwargs={},
    pmwd_conf_kwargs=dict(
        ptcl_grid_shape=(n_ptcl_per_side,) * 3,
        ptcl_spacing=L_box / n_ptcl_per_side,
        mesh_shape=mesh_shape,
        a_start=1 if just_2lpt else 1 / 64,
    ),
)
```

2 → run (on GPU)

```
ptcl = EU.run_PM(seed=0)
```

3 → build halos (slow...)

```
halos_xyz, halos_v_xyz, halos_mass = EU.build_halos(
    ptcl_xyz,
    ptcl_v_xyz,
    linking_length=halos_linking_length,
)
```

4 → observe

```
ra, dec, z_cos, v_r = EU.observe(
    halos_xyz,
    halos_v_xyz,
    obs_xyz=obs_xyz,
    obs_v_xyz=obs_v_xyz,
    box_coords="celestial",
)
```

5 → quick overview

```
rho, v_xyz = EU.build_CiCs(ptcl, cic_log2_refine)
fig_rho, fig_v_r, fig_ps = EU.analyze_CiCs(rho, v_xyz)
fig_obs = EU.analyze_obs(ra, dec, z_cos, v_r)
```

Halos or particles?