Workshop on Tensions in Cosmology

Corfu 11.09.2022

Boltzmann solvers in the era of cosmological tensions: symbolic implementation of extensions in PyCosmo

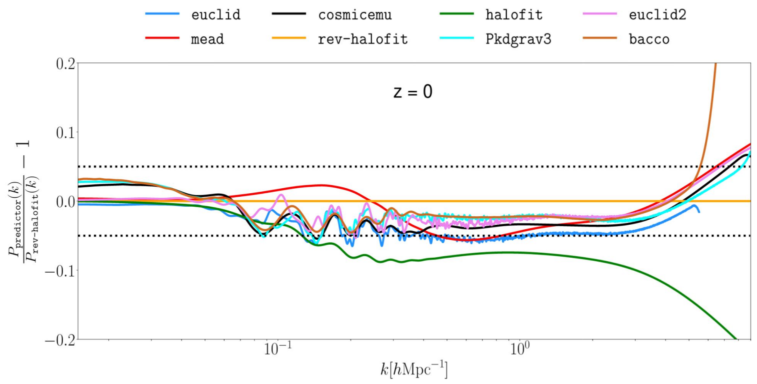
Beatrice Moser (moserb@phys.ethz.ch)

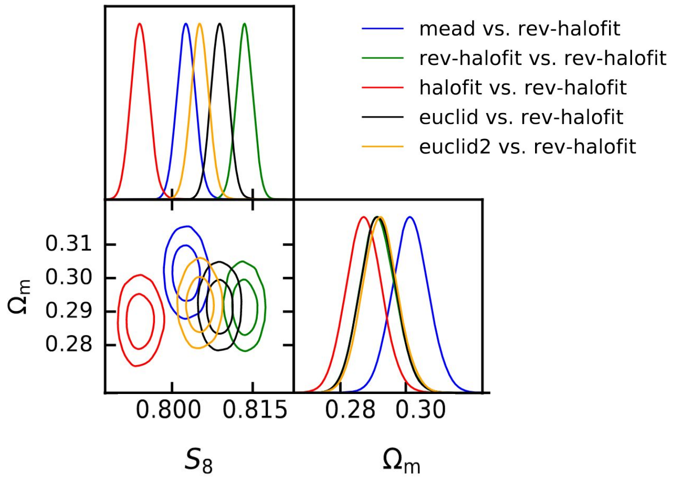
IPA ETH zürich

In collaboration with: Christiane S. Lorenz, Uwe Schmitt, Alexandre Refregier, Janis Fluri, Raphael Sgier, Federica Tarsitano, Lavinia Heisenberg

Theory codes and cosmological tensions

❖ Theoretical uncertainties can impact cosmological constraints from upcoming surveys (e.g. Stage IV weak lensing surveys) ⇒ understand and minimize approximations



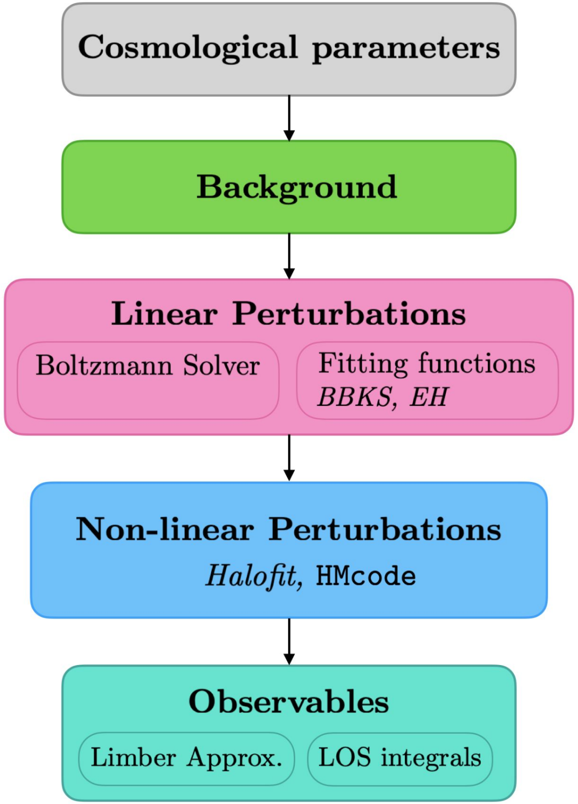


Figures from Tan et al., 2022, arXiv:2207.03598

❖ Tensions ⇒ necessity to implement new models to test extensions of ΛCDM

Presented in: Refregier et al., 2017, arXiv:1708.05177
Code Comparisons: Tarsitano et al., 2021, arXiv:2005.00543

PyCosmo



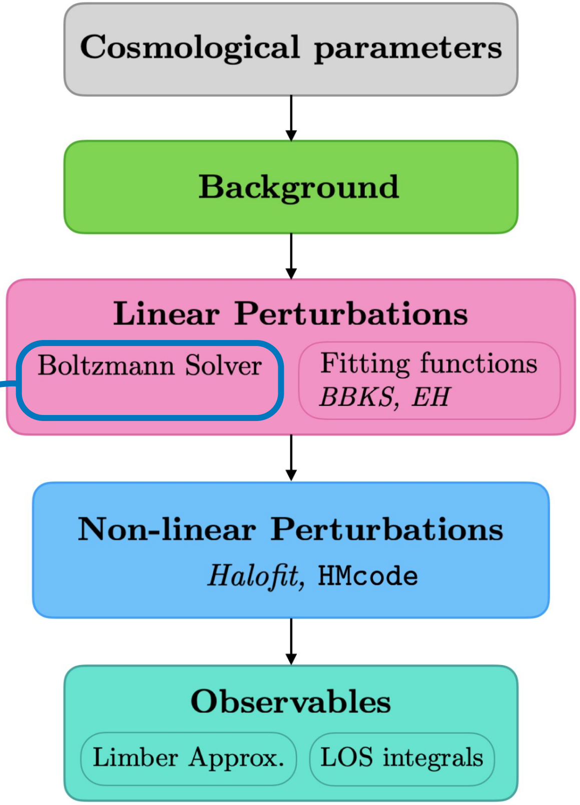
Credit: Tarsitano et al., 2021, arXiv:2005.00543

Boltzmann solver

PyCosmo

- Solves the Einstein - Boltzmann ODE system numerically
- Most widely used are `CLASS` and `CAMB`
- `PyCosmo` approach:
  - `Sympy` symbolic equations translated to fast C/C++ code by `sympy2c`
  - Easily extensible to new models and fast
  - Reduce number of approximations
  - Easily accessible through the PyCosmo HUB https://pycosmohub.com

Boltzmann solver extensions: Moser et al., 2022, arXiv:2112.08395
`sympy2c`: Schmitt et al., 2022, arXiv:2203.11945



Credit: Tarsitano et al., 2021, arXiv:2005.00543

Model ⇒ Result

$$\frac{d\delta_b}{dlna} = -\frac{k}{aH}u_b - 3\frac{d\Phi}{dlna}$$

SymPy

```
a = Symbol("a")
k = Symbol("k")
Phi = Symbol("Phi")
delta_b = Symbol("delta_b")
u_b = Symbol("u_b")

ddelta_b_dlan = -k / (a * H)
                 * u_b
                 - 3 * dPhi_dlan
```

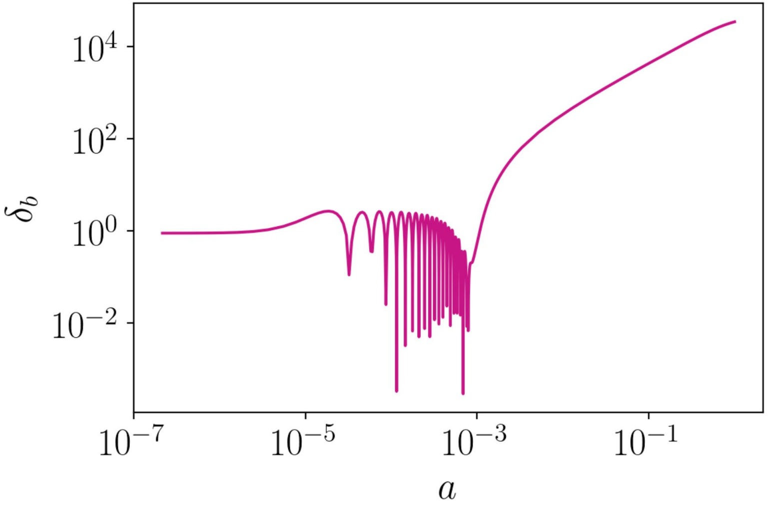
Model ⇒ Result

$$\frac{d\delta_b}{d\ln a} = -\frac{k}{aH}u_b - 3\frac{d\Phi}{d\ln a}$$

SymPy

```
a = Symbol("a")
k = Symbol("k")
Phi = Symbol("Phi")
delta_b = Symbol("delta_b")
u_b = Symbol("u_b")

ddelta_b_dlan = -k / (a * H)
                * u_b
                - 3 * dPhi_dlan
```

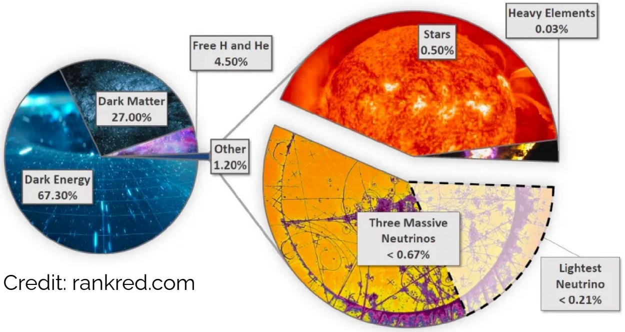


`sympy2c`

- ❖ Generates C/C++ code used from Python as extension module
- ❖ Optimization using permutations and splits
- ❖ Use LSODA solver

Extensions: Dark energy with a constant equation of state and Massive Neutrinos

- Dark energy with a constant equation of state $w_{\rm de} = p_{\rm de}/\rho_{\rm de} \neq -1$ is a minimal extension to ΛCDM ⇒ adds two DE perturbation equations
- Massive neutrinos:
  - Involve numerical integration already at background level
  - Hierarchy of multipoles + momentum dependence

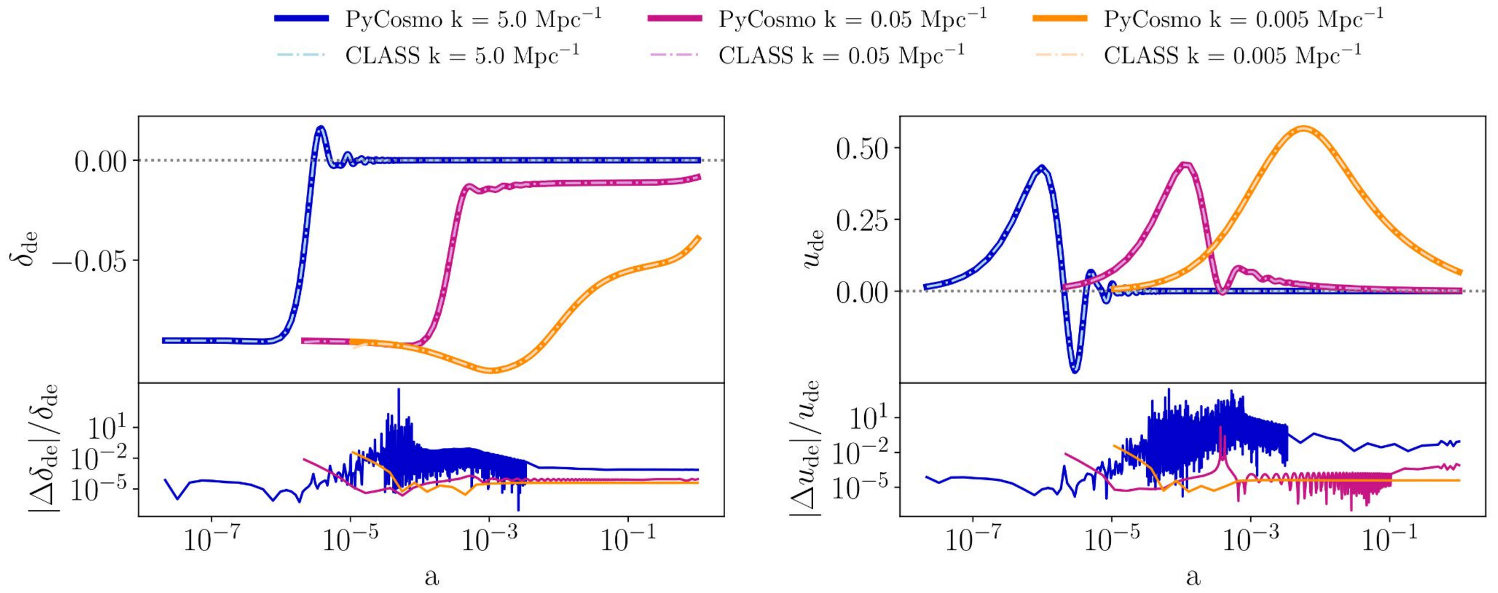


Credit: rankred.com

- Discretized momenta
- `sympy2c` handles numerical integration

Extensions: Dark energy with a constant equation of state and Massive Neutrinos

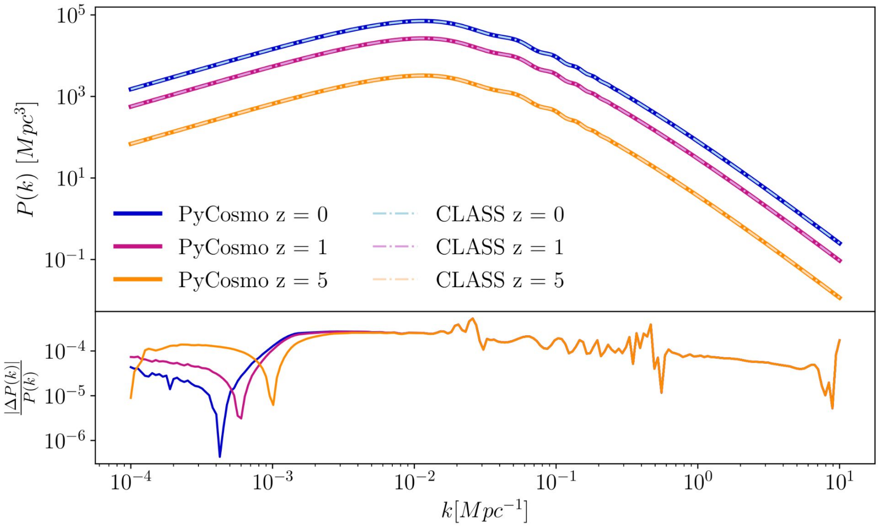
- Compare the results with `CLASS`
  - Evolution of fields at fixed k values



Dark energy model with $w_{de} = -0.9$

Extensions: Dark energy with a constant equation of state and Massive Neutrinos

- ❖ Compare the results with `CLASS`
  - ➢ Total matter power spectrum at fixed redshift z



Three degenerate massive neutrinos with $\Sigma m_\nu = 60\ \mathrm{meV}$

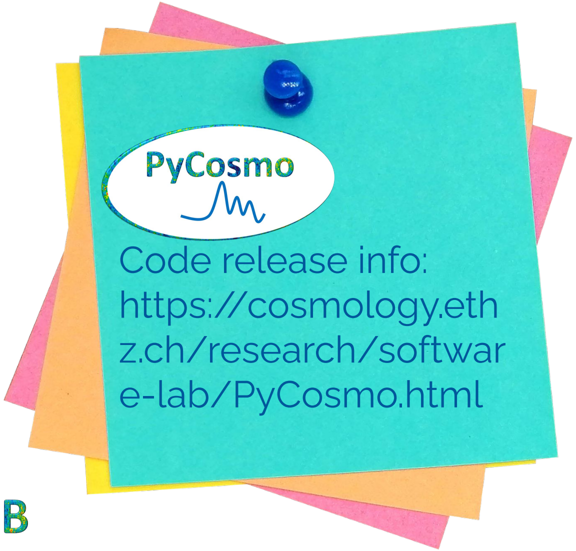
- ❖ Runtime similar to `CLASS` (depending on precision settings and models)

Conclusion

Boltzmann solver extensions: Moser et al., 2022, arXiv:2112.08395
`sympy2c`: Schmitt et al., 2022, arXiv:2203.11945

- `PyCosmo` Boltzmann solver uses `sympy2c` to translate `Sympy` symbolic expressions to optimized C/C++ code
- **Easily extensible to new models!**
- Possible extensions:
  - Quintessence
  - Early dark energy
  - Dark matter / neutrino models

  → **Joel Mayor**
- Publicly available
- Can be used interactively on the PyCosmo HUB



See **Silvan Fischbacher** at 17.10 in PSA for a `PyCosmo` application!

Thank you!

Beatrice Moser (moserb@phys.ethz.ch)

Additional slides

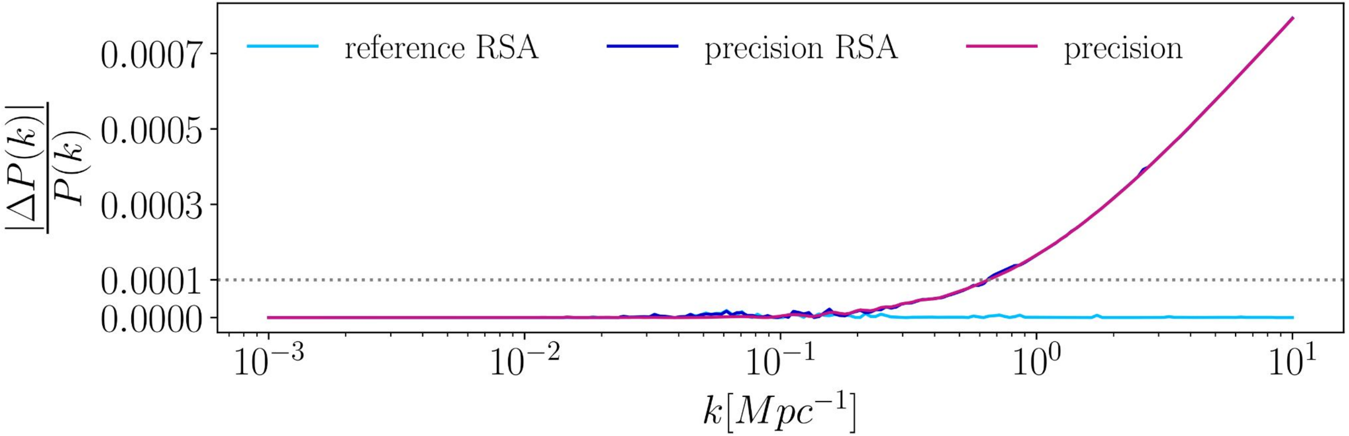
Runtime

Model	k_{max}	*Speed* settings, time [s]			*Precision* settings, time [s]		
		`PyCosmo`	`PyCosmo RSA`	`CLASS`	`PyCosmo`	`PyCosmo RSA`	`CLASS`
ΛCDM	1 Mpc^{-1}	1.26	0.23	0.42	3.80	1.05	2.02
ΛCDM	10 Mpc^{-1}	8.80	0.44	0.80	20.5	2.20	5.28
wCDM	1 Mpc^{-1}	1.32	0.65	1.29	3.84	1.55	2.86
wCDM	10 Mpc^{-1}	9.08	0.82	4.91	20.93	2.72	10.18
degenerate M_ν	1 Mpc^{-1}	54.54	29.04	10.19	237.26	154.93	105.87
degenerate M_ν	10 Mpc^{-1}	357.24	98.52	13.78	1337.32	471.22	417.95

Table 1: Best execution time from three executions on a full Euler VI node.

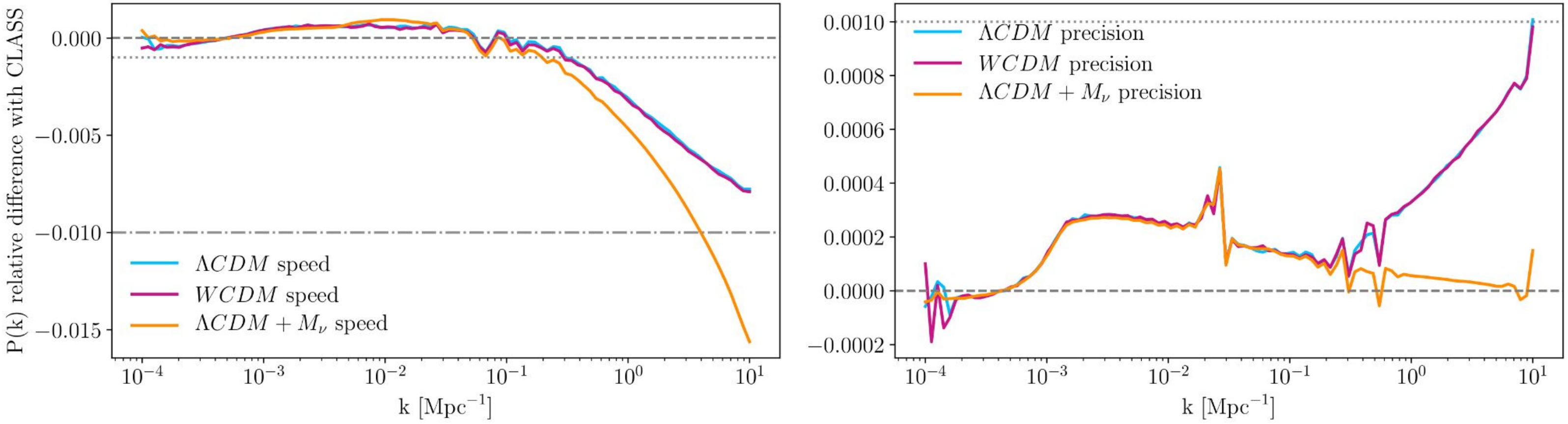
Extensions: Radiation Streaming Approximation

- After decoupling, photons and massless neutrinos free stream in external gravitational fields
- RSA reduces size of ODE system ⇒ speed-up
- `sympy2c` handles dynamical switch between two different ODE systems
- Induced error negligible



Agreement and Performance

- Agreement with `CLASS` at different precision settings for all models



- Runtime similar to `CLASS` when using RSA (depending on precision settings and models)

Precision settings:

- Speed:
  - CLASS: cl_permille.pre (RSA+UFA+TCA+ncdmFA)
  - PyCosmo: l_max=l_max_mnu=17, rtol=atol=mnu_relerr=10^{-5}
- Precision:
  - CLASS: pk_ref.pre (RSA+UFA+TCA)
  - PyCosmo: l_max=l_max_mnu=50 rtol=atol=mnu_relerr=10^{-6}

Total matter power spectrum

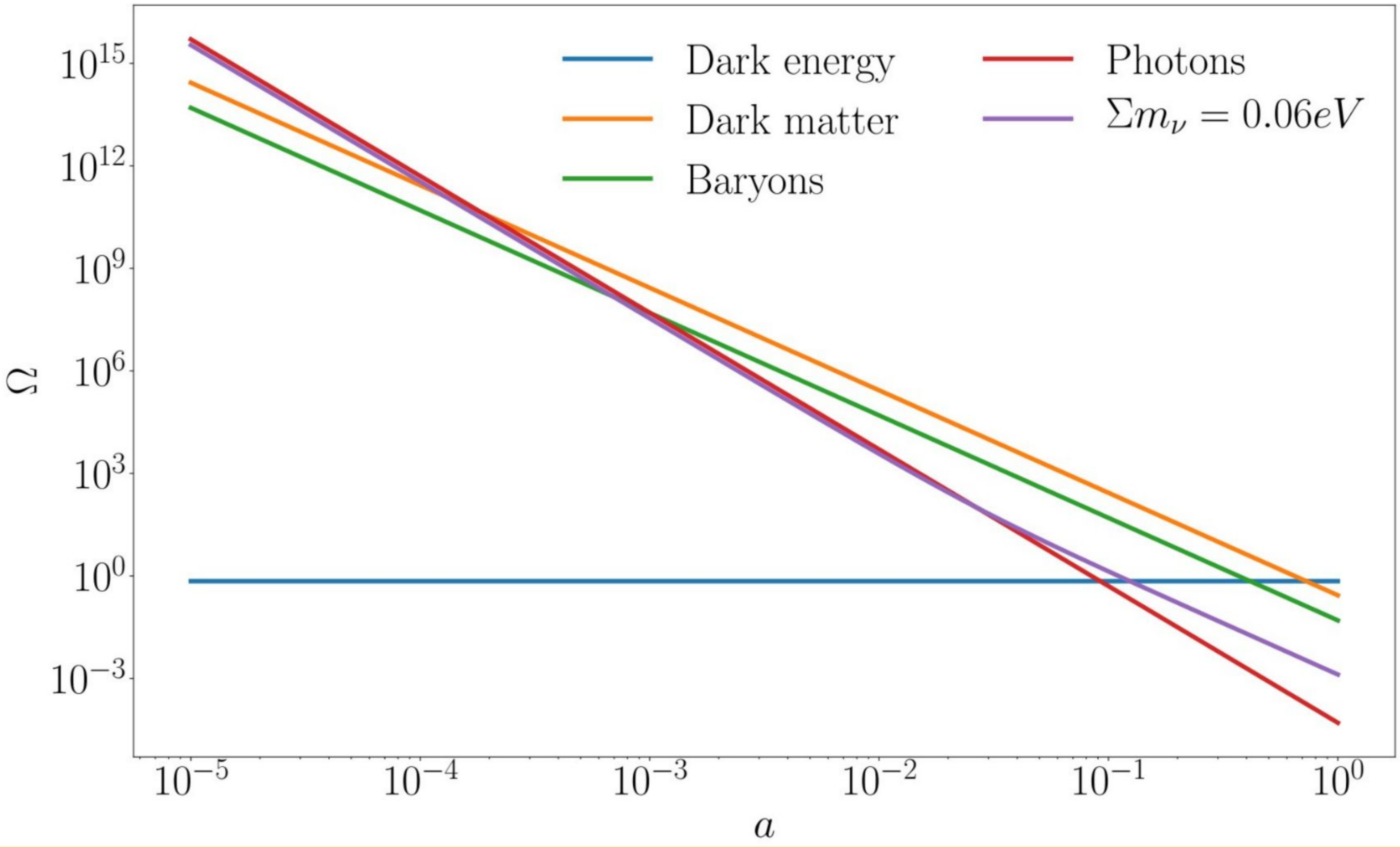
- We compute the the gauge invariant real space matter power spectrum accounting for real space matter fluctuations and volume distortions

$$P(k,a) = 2\pi^2 \mathcal{A}_s \frac{k^{n_s-4}}{k_p^{n_s-1}} \delta_{m,tot}(k,a)^2$$

where

$$\delta_{m,tot} = \frac{\delta\rho_{m,tot}}{\bar{\rho}_{m,tot}} + 3\frac{aH}{k^2}\theta_{m,tot} = \frac{\Omega_{dm}\delta + \Omega_b\delta_b + \Omega_{\nu,m}\delta_{\nu,m}}{\Omega_{m,tot}} + 3\frac{aH}{k}\frac{\Omega_{dm}u + \Omega_b u_b + (\Omega_{\nu,m} + P_{\nu,m})u_{\nu,m}}{\Omega_{m,tot} + P_{m,tot}}$$

(Yoo et al., 2009; Yoo, 2010; Bonvin & Durrer, 2011; Challinor & Lewis, 2011)



$$\Omega_m = 0.321 \quad \Omega_b = 0.05 \quad \Omega_\kappa = 0 \quad N_\nu = 0 \quad N_{\nu,m} = 3 \quad m_\nu = 0.02eV/c^2$$

Suppression of the PS on small scales