

class.c

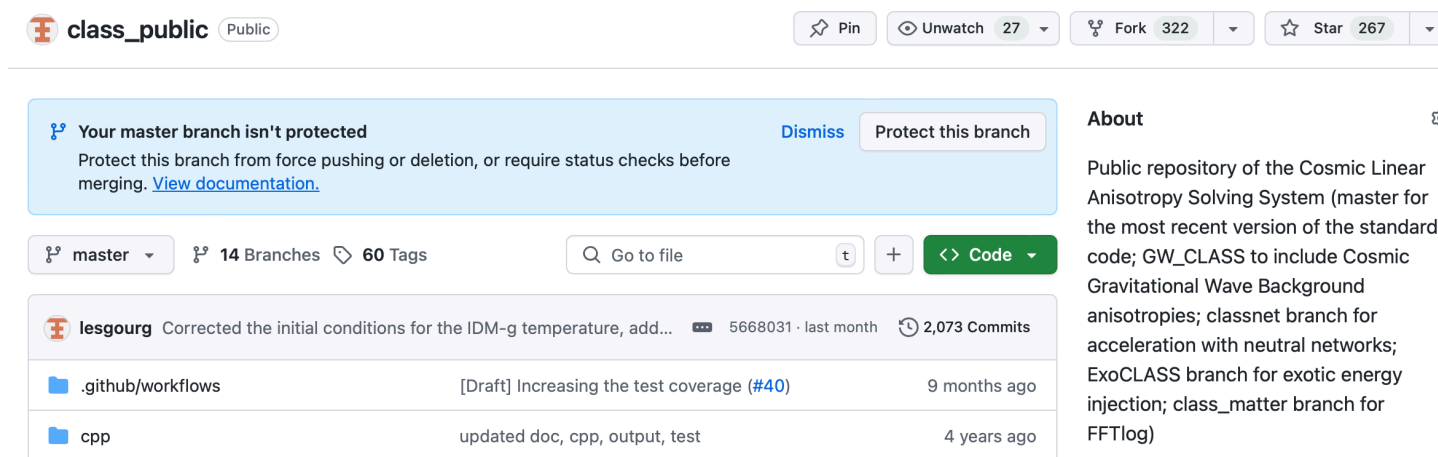


000004e0 0a 0a 45 72 72 6f 72 20 72 75 6e 6e 69 6e 67 20 00000620 69 6e 20 6c 65 6e 73 69 6e 67 5f 69 6e 69 74 20
000004f0 69 6e 70 75 74 5f 69 6e 69 74 5f 66 72 6f 6d 5f 00000630 0a 3d 3e 25 73 0a 00 0a 0a 45 72 72 6f 72 20 69
00000500 61 72 67 75 6d 65 6e 74 73 20 0a 3d 3e 25 73 0a 00000640 6e 20 6f 75 74 70 75 74 5f 69 6e 69 74 20 0a 3d
00000510 00 00 00 00 00 00 00 00 0a 0a 45 72 72 6f 72 20 00000650 3e 25 73 0a 00 00 00 00 0a 0a 45 72 72 6f 72 20
00000520 72 75 6e 6e 69 6e 67 20 62 61 63 6b 67 72 6f 75 00000660 69 6e 20 6c 65 6e 73 69 6e 67 5f 66 72 65 65 20
00000530 6e 64 5f 69 6e 69 74 20 0a 3d 3e 25 73 0a 00 00 00000670 69 6e 20 73 70 65 63 74 72 61 5f 66 72 65 65 20
00000540 0a 0a 45 72 72 6f 72 20 69 6e 20 74 68 65 72 6d 00000680 0a 3d 3e 25 73 0a 00 00 0a 0a 45 72 72 6f 72 20
00000550 6f 64 79 6e 61 6d 69 63 73 5f 69 6e 69 74 20 0a 00000690 69 6e 20 74 72 61 6e 73 66 65 72 5f 66 72 65 65
00000560 3d 3e 25 73 0a 00 00 00 0a 0a 45 72 72 6f 72 20 000006a0 20 0a 3d 3e 25 73 0a 00 0a 0a 45 72 72 6f 72 20
00000570 69 6e 20 70 65 72 74 75 72 62 5f 69 6e 69 74 20 000006b0 69 6e 20 6e 6f 6e 6c 69 6e 65 61 72 5f 66 72 65
00000580 0a 3d 3e 25 73 0a 00 00 0a 0a 45 72 72 6f 72 20 000006c0 65 20 0a 3d 3e 25 73 0a 00 00 00 00 00 00 00 00
00000590 69 6e 20 70 72 69 6d 6f 72 64 69 61 6c 5f 69 6e 000006d0 0a 0a 45 72 72 6f 72 20 69 6e 20 70 72 69 6d 6f
000005a0 69 74 20 0a 3d 3e 25 73 0a 00 00 00 00 00 00 00 000006e0 72 64 69 61 6c 5f 66 72 65 65 20 0a 3d 3e 25 73
000005b0 0a 0a 45 72 72 6f 72 20 69 6e 20 6e 6f 6e 6c 69 000006f0 0a 00 00 00 00 00 00 00 0a 0a 45 72 72 6f 72 20
000005c0 6e 65 61 72 5f 69 6e 69 74 20 0a 3d 3e 25 73 0a 00000700 69 6e 20 70 65 72 74 75 72 62 5f 66 72 65 65 20
000005d0 00 00 00 00 00 00 00 00 0a 0a 45 72 72 6f 72 20 00000710 0a 3d 3e 25 73 0a 00 00 0a 0a 45 72 72 6f 72 20
000005e0 69 6e 20 74 72 61 6e 73 66 65 72 5f 69 6e 69 74 00000720 69 6e 20 74 68 65 72 6d 6f 64 79 6e 61 6d 69 63
000005f0 20 0a 3d 3e 25 73 0a 00 0a 0a 45 72 72 6f 72 20 00000730 73 5f 66 72 65 65 20 0a 3d 3e 25 73 0a 00 00 00
00000600 69 6e 20 73 70 65 63 74 72 61 5f 69 6e 69 74 20 00000740 0a 0a 45 72 72 6f 72 20 69 6e 20 62 61 63 6b 67
00000610 0a 3d 3e 25 73 0a 00 00 0a 0a 45 72 72 6f 72 20 00000750 72 6f 75 6e 64 5f 66 72 65 65 20 0a 3d 3e 25 73

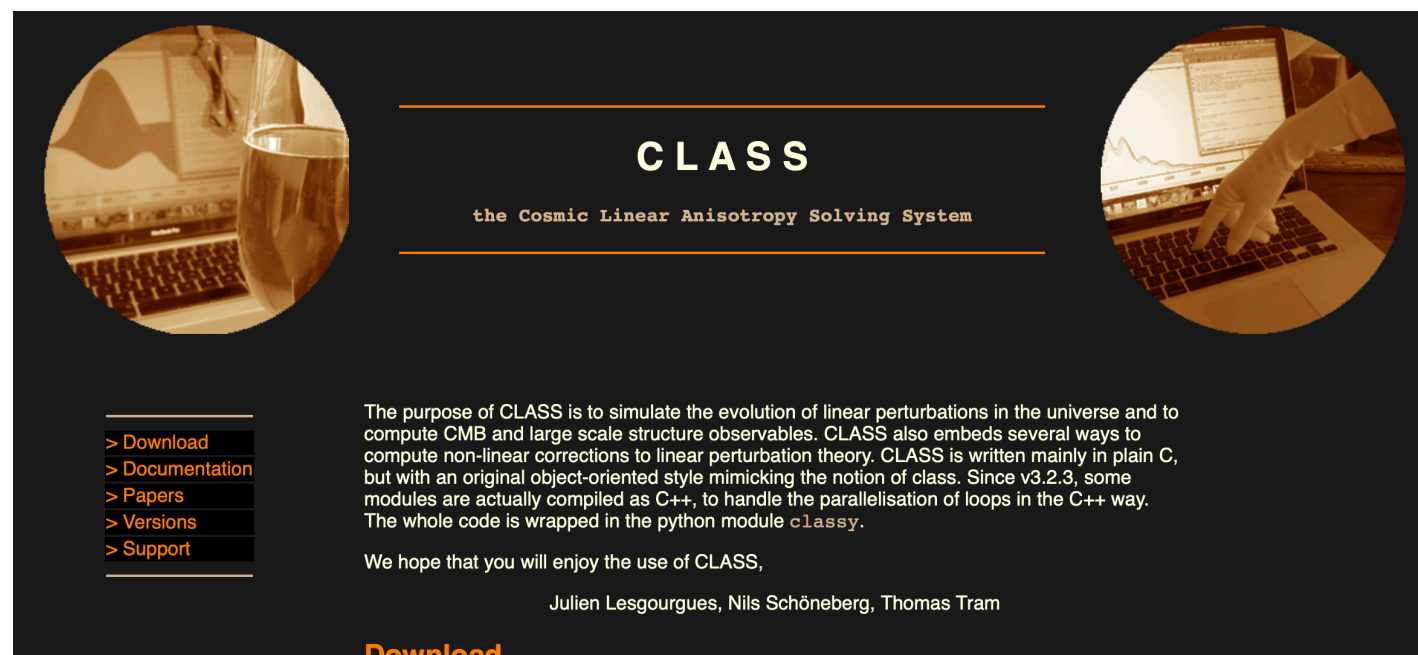
JL, Thomas Tram, Nils Schöneberg (+community)

Where they code lives

- CLASS defined as a python module on PiPY repository
- https://github.com/lesgourg/class_public/ contains master branch (and other public branches)



- <http://class-code.net/> contains latest class_public-x.y.z.tar.gz + info, links to docs, wiki, articles

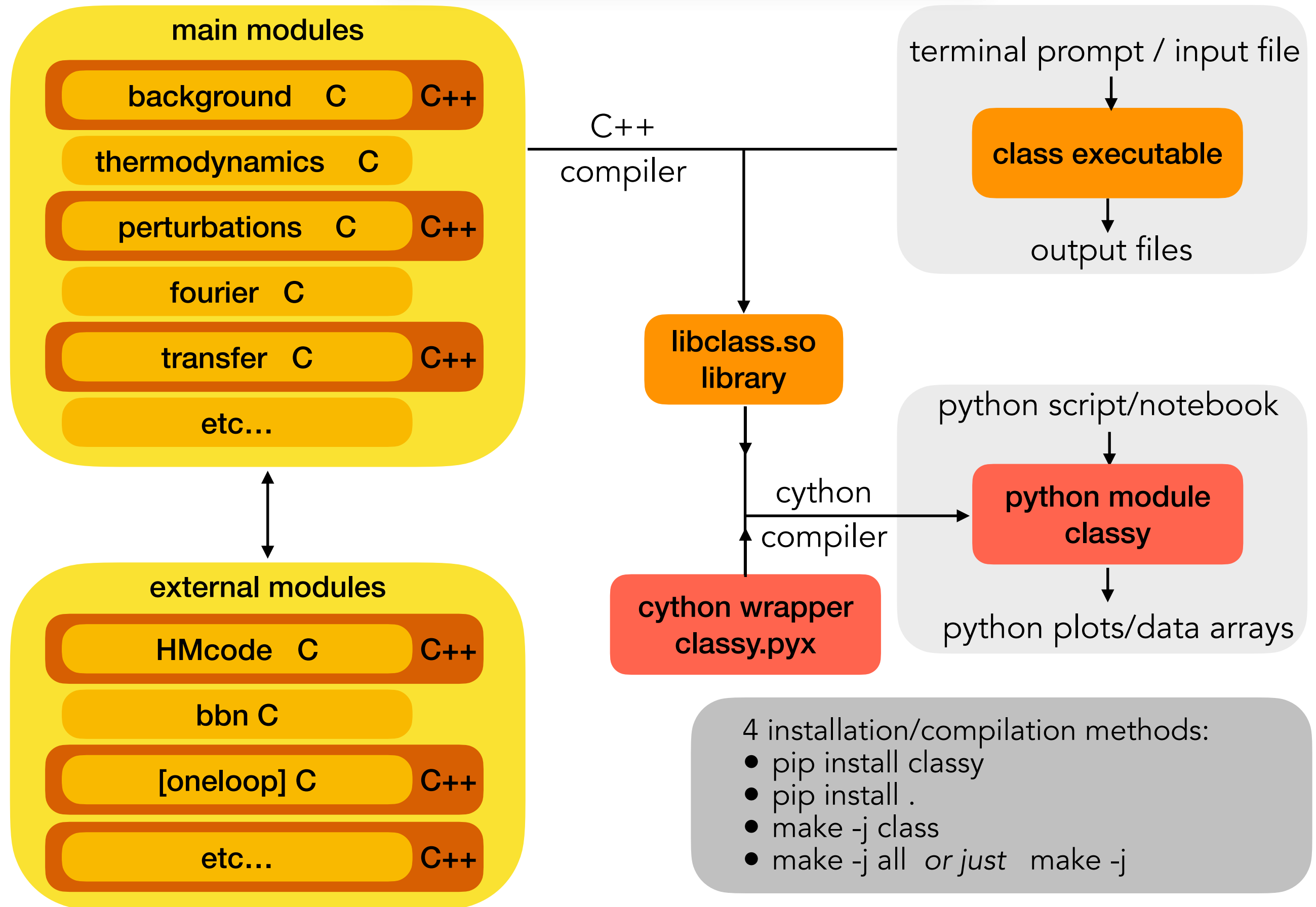


Where the code is documented

class is documented in various places:

- The file **explanatory.ini** lists all input parameters.
(downloadable from GitHub page if classy installed via pip)
- **NEW!** Documentation of the **classy** wrapper at <https://class-code.readthedocs.io>.
- **NEW!** class LLM agent at <https://classclapp.streamlit.app/>.
- **Updated!** Python docstrings in the **classy** wrapper.
- class manual in repository
- Online documentation at https://github.com/lesgourg/class_public/wiki
- Old course notes linked on <http://class-code.net> and <https://schoeneberg.github.io/>

Overall architecture of CLASS

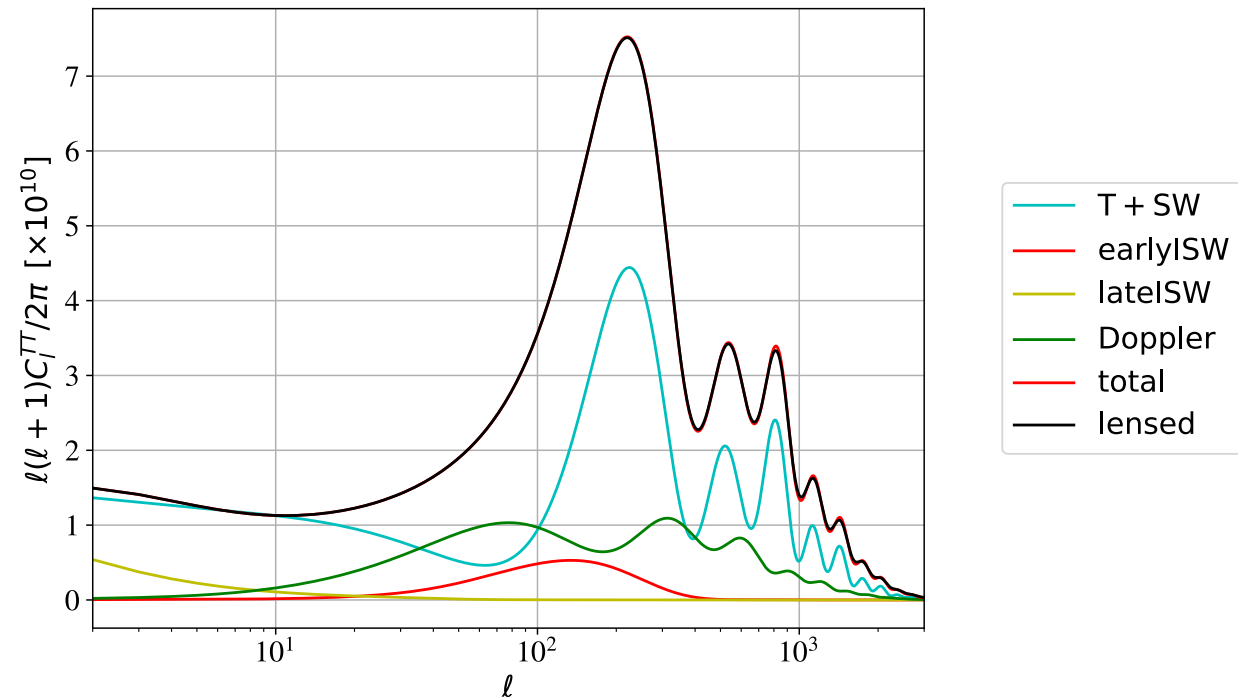


Basic runs

- Run with one of :
 - `./class default.ini`
 - `python scripts/warmup.py`
 - `jupyter notebook notebooks/warmup.ipynb`
- Syntax for all **input parameters** detailed in **explanatory.ini**
- Input file: `omega_b = 0.02238`
`output = tCl, pCl`
- python: `xxx.set({'omega_b': '0.02238', 'output': 'tCl, pCl'})`
- demo with **notebooks/warmup.ipynb**

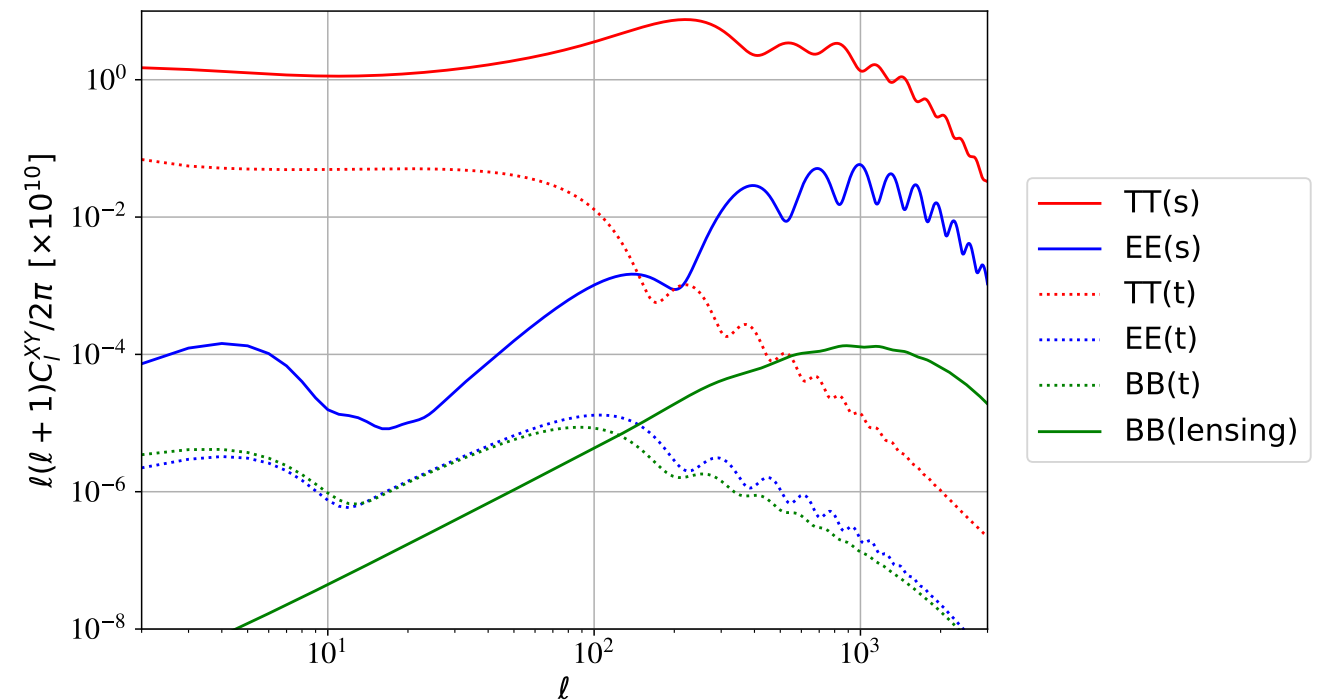
Example of output from other notebooks in notebooks/

cltt_terms

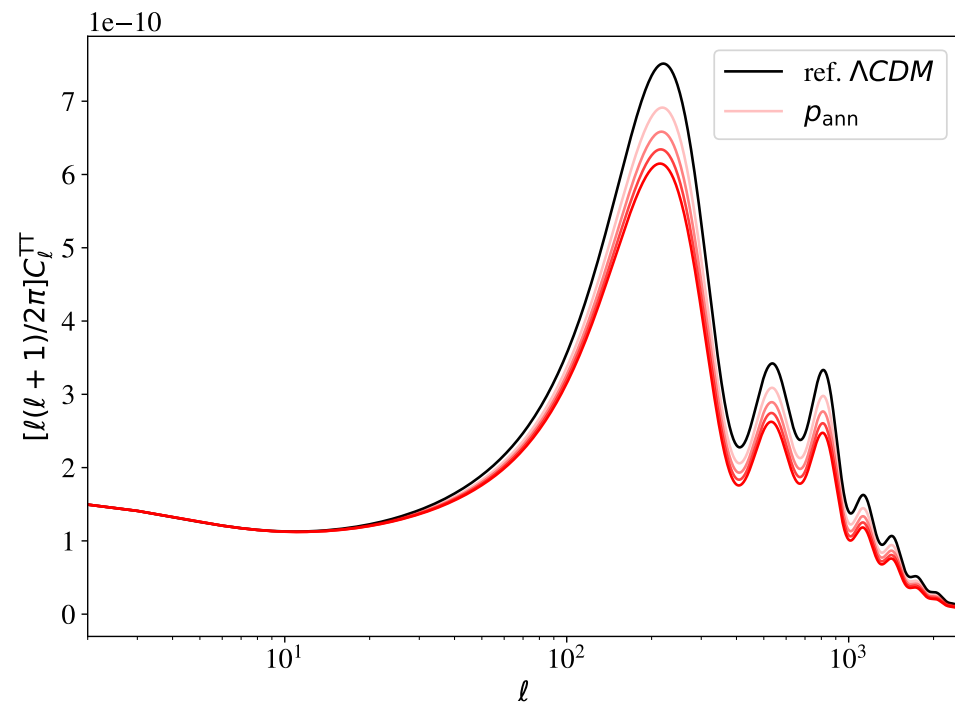


cl_ST

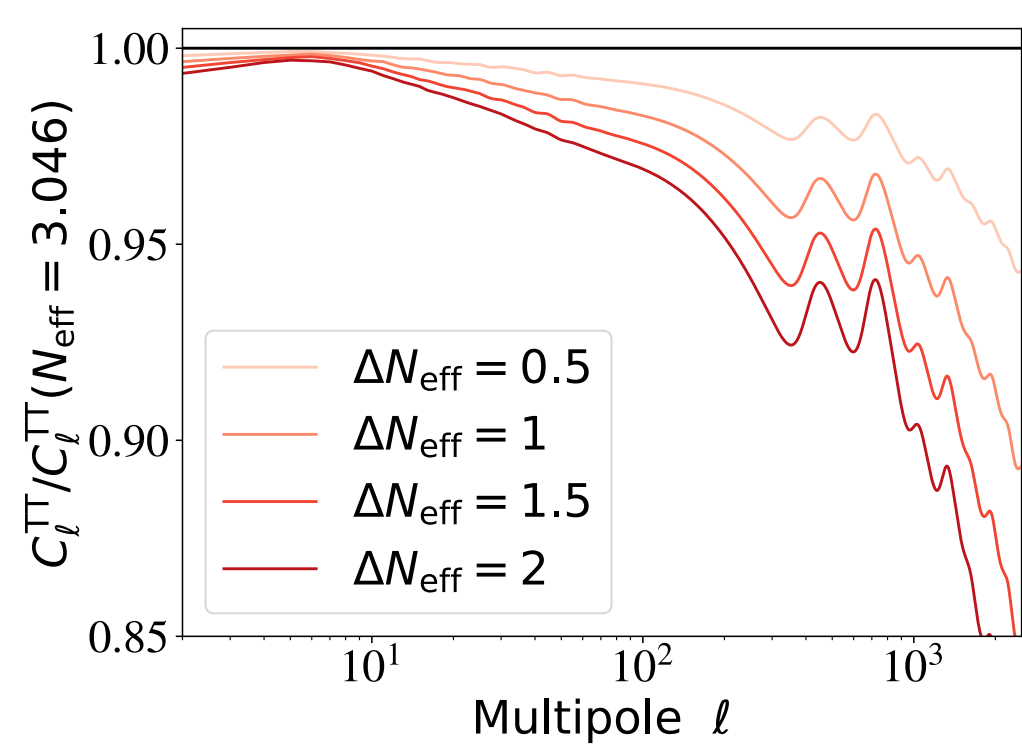
$r = 0.1$



varying_pann

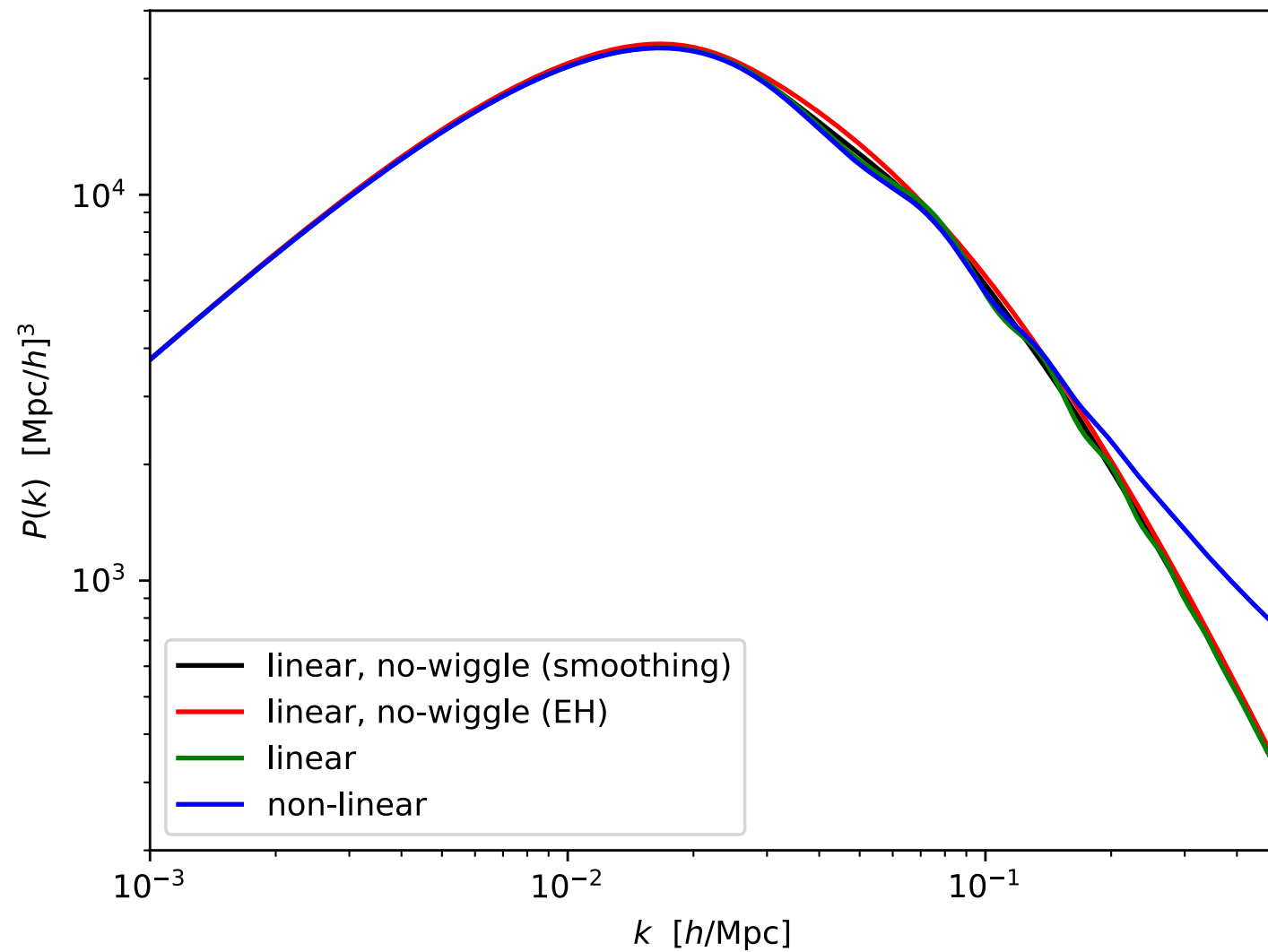


varying_neff

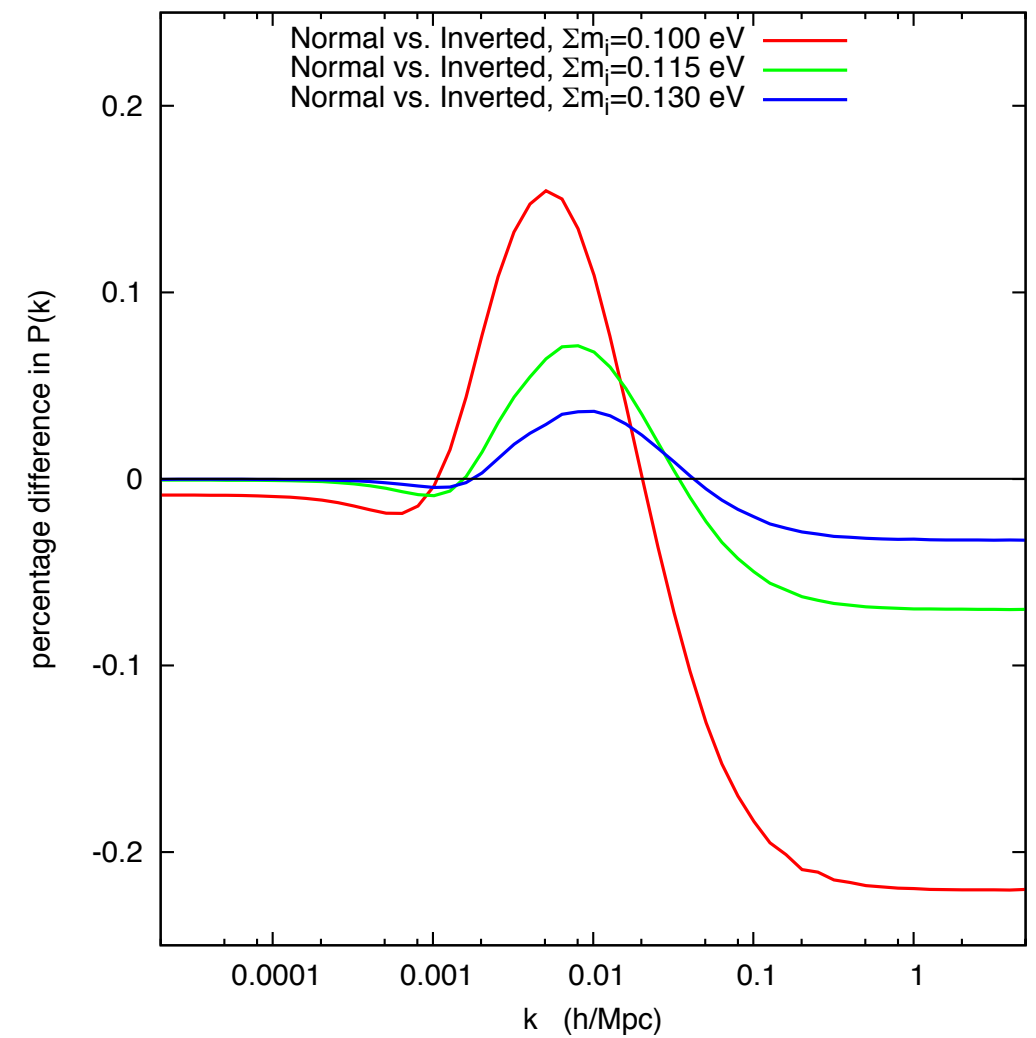


Example of output from other notebooks in notebooks/

test_hmcode

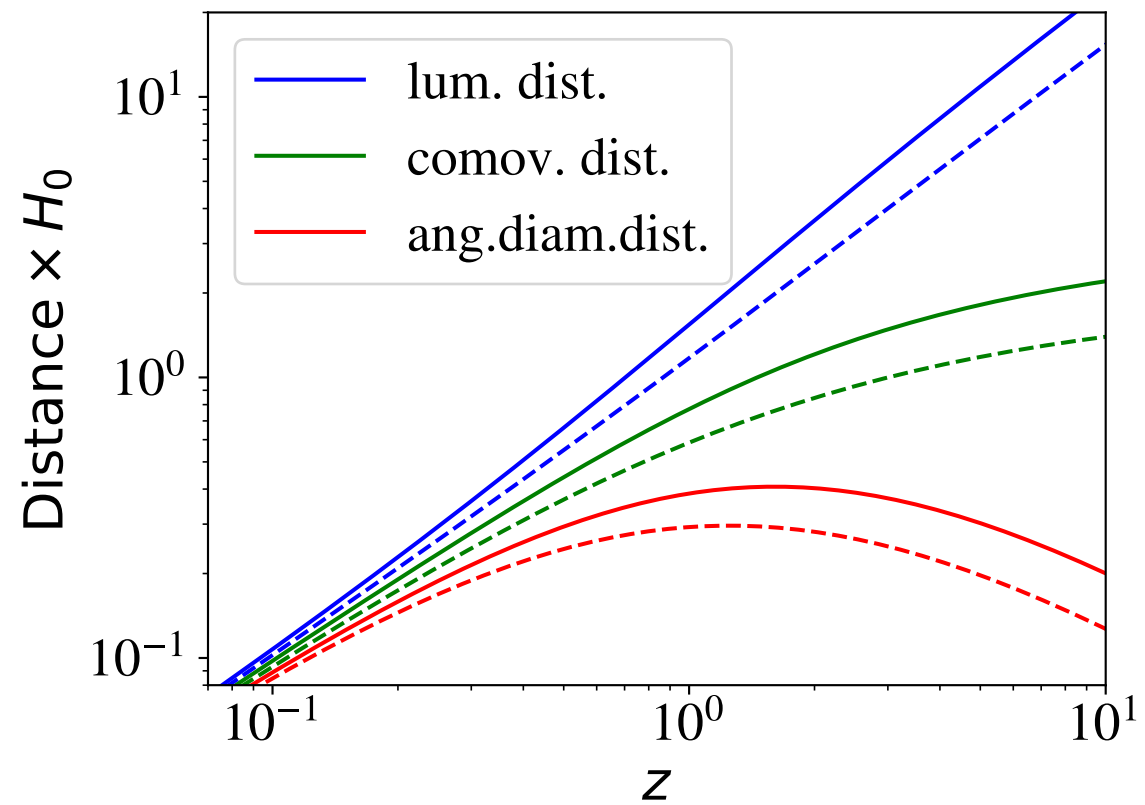


neutrino_hierarchy

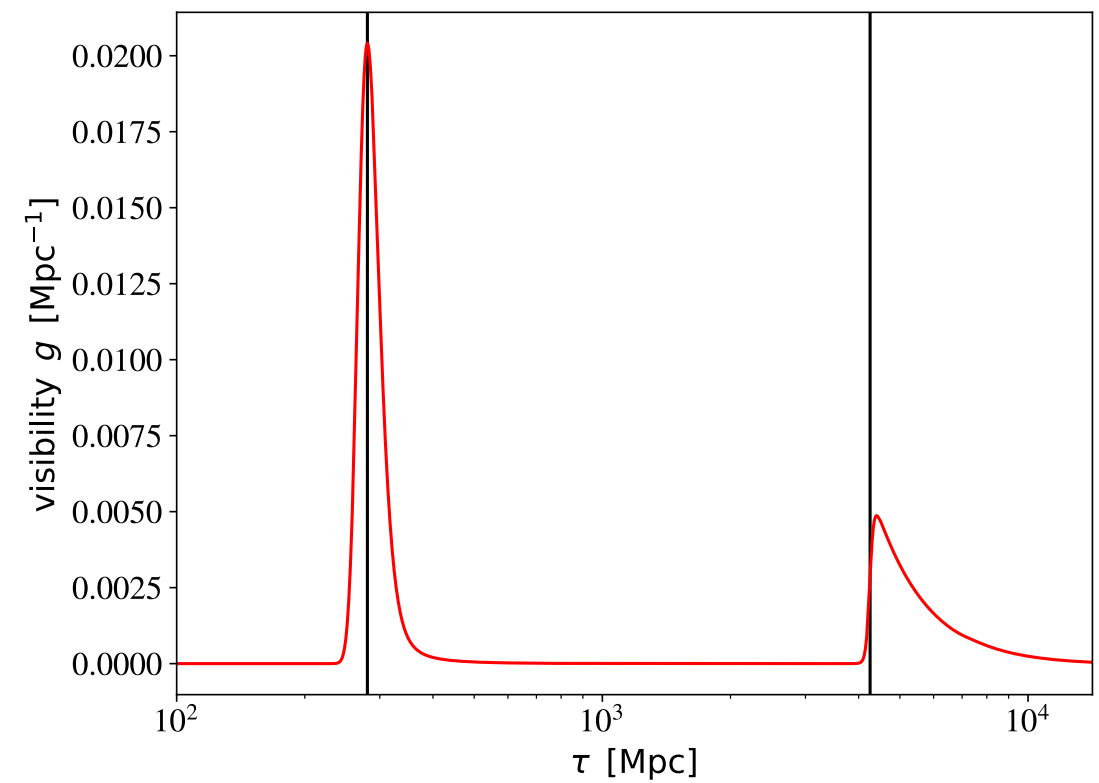


Example of output from other notebooks in notebooks/

distances

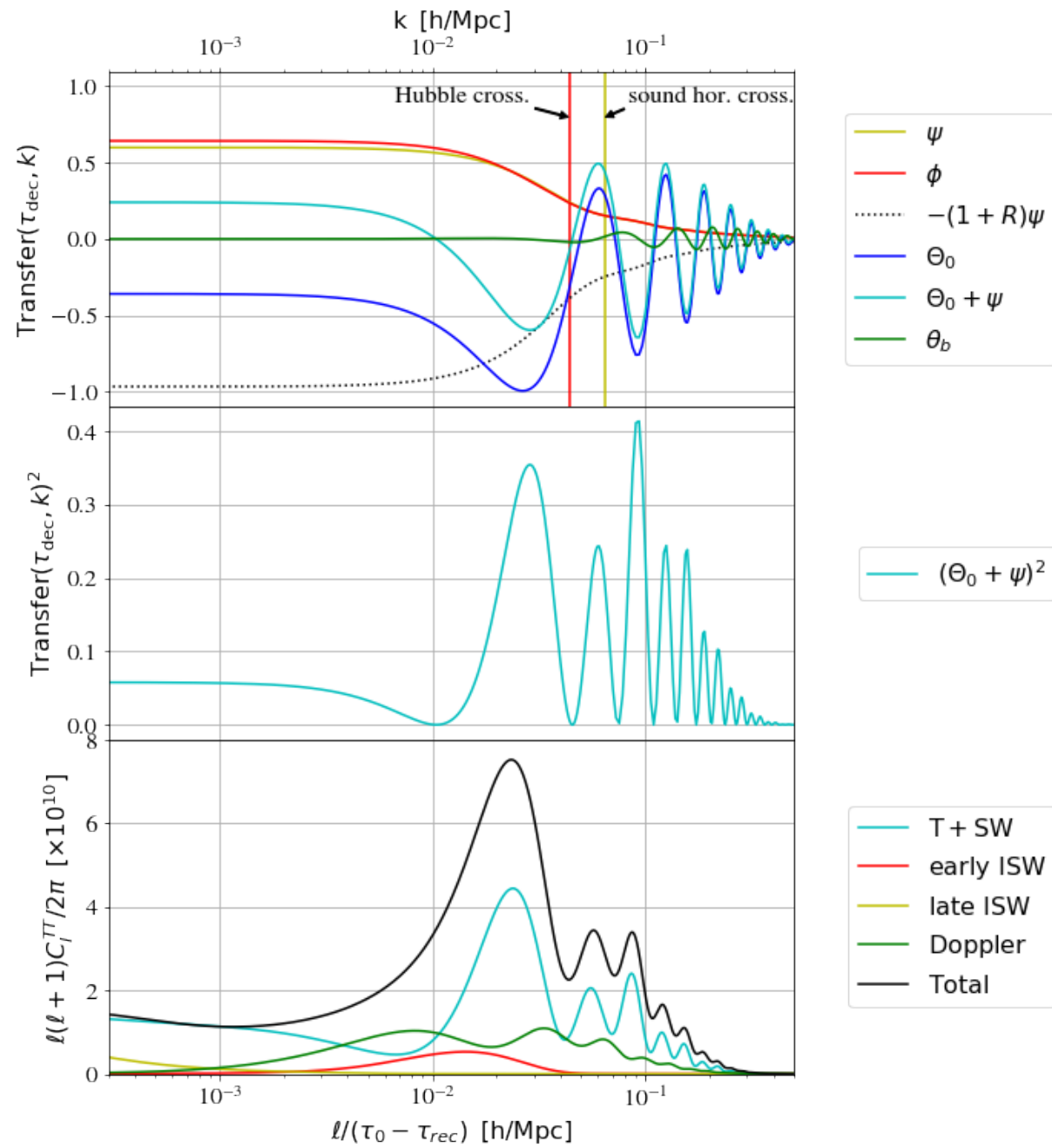


thermo

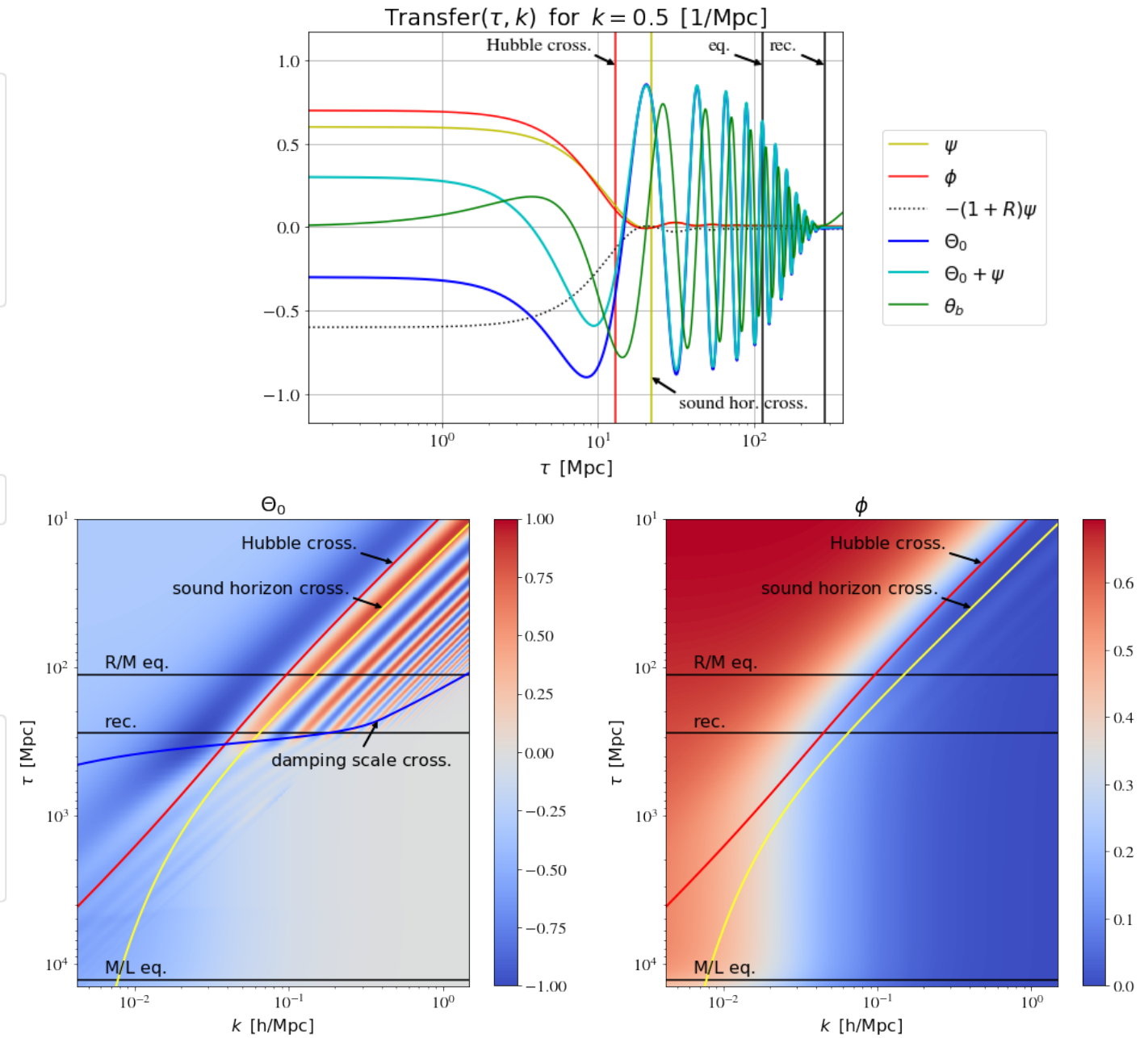


Example of output from other notebooks in notebooks/

one_time



one_k



many_k

Content of the modules

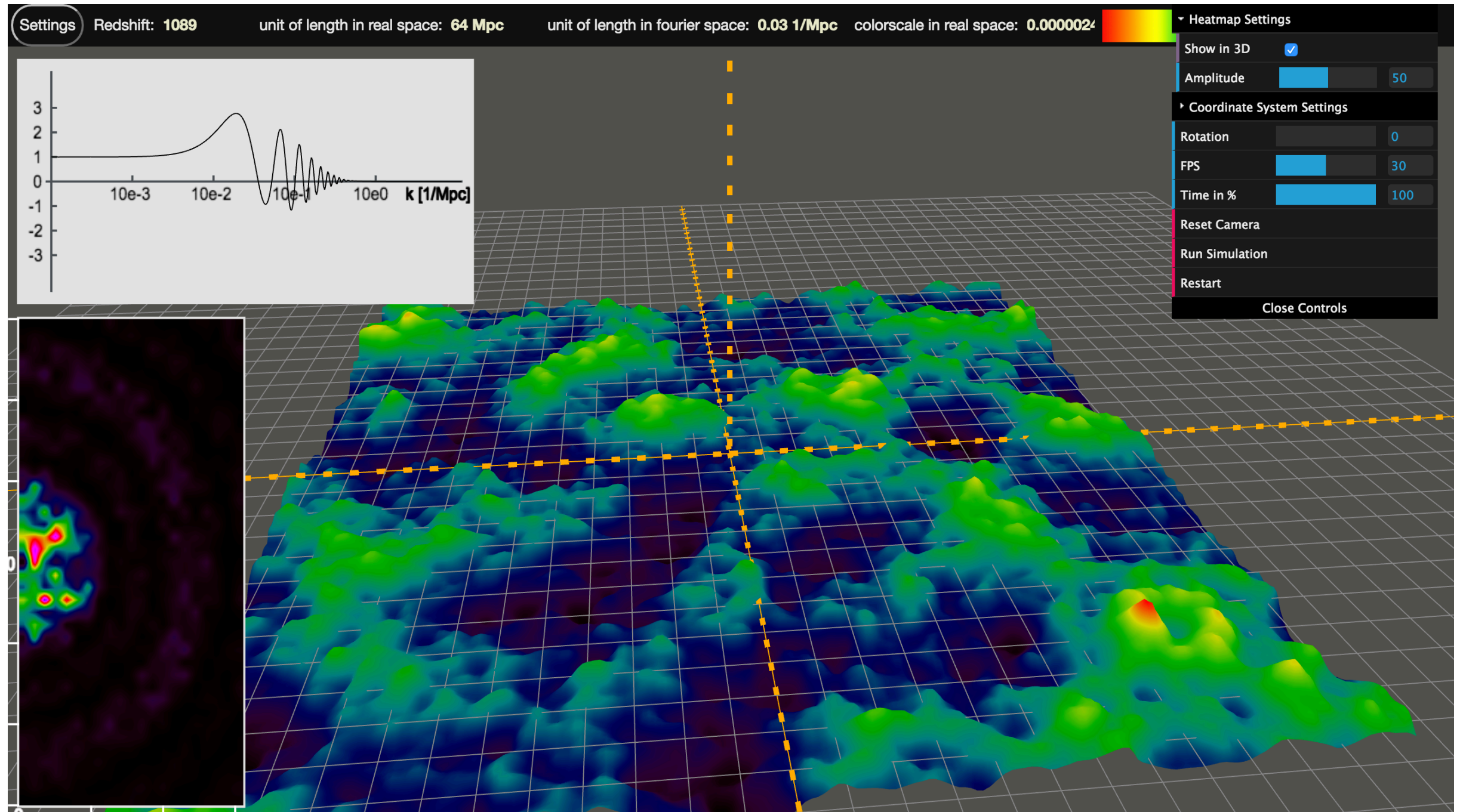
Main modules

1. input
2. background e.g. $d_a(z)$
3. thermodynamics e.g. $g(\tau)$
4. perturbations e.g. $\delta_c(\tau, k)$
5. primordial e.g. $\mathcal{P}_{\mathcal{R}}(k)$
6. fourier e.g. $P_m(z, k)$
7. transfer e.g. $\Delta_\ell(\tau, k)$
8. harmonic e.g. $C_\ell^{\text{TT}, \text{unlensed}}$
9. lensing e.g. $C_\ell^{\text{TT}, \text{lensed}}$
10. distortions e.g. $y, \mu, \dots$

External modules (codes/data)

- bbn
- HyRec2020
- RecfastCLASS
- Halofit
- HMcode
- heating
- distortions
- external_Pk
- RealSpaceInterface

RealSpaceInterface



CLAPP

- described in [arXiv:2508.05728](https://arxiv.org/abs/2508.05728)

CLAPP: The CLASS LLM Agent for Pair Programming

S. Casas,^{1,2} C. Fidler,¹ B. Bolliet,^{3,4} F. Villaescusa-Navarro,^{5,6} and J. Lesgourgues¹

<https://classclapp.streamlit.app/> or <https://github.com/santiagocasas/clapp>

Share ☆ ↗ 🔊 ⋮

🔑 API & Assistants

OpenAI API Key

..... 🔍

Password to encrypt/decrypt API key

..... 🔍

Choose LLM model 🧠

gpt-4o ▾

Response Mode

☒ Fast Mode

⚡ Quick responses with good quality (recommended for most uses)

🚀 Initialize with Selected Model

🔧 CLASS Setup

Install CLASS to enable code execution and plotting capabilities

📦 Install CLASS

🧪 Test CLASS



🗨️ Hello! I'm here to assist you with the CLASS code, a cosmological tool that specializes in solving the Einstein-Boltzmann equations. My role is to help you understand the key components of the CLASS code and guide you through specific tasks such as setting up cosmological parameters, understanding differential equation-solving related to cosmology, and interpreting results. Whether you need help with implementation details or clarifying concepts in cosmology, feel free to ask!

Type your prompt here... ➤

CLAPP

- described in [arXiv:2508.05728](https://arxiv.org/abs/2508.05728)

CLAPP: The CLASS LLM Agent for Pair Programming

S. Casas,^{1,2} C. Fidler,¹ B. Bolliet,^{3,4} F. Villaescusa-Navarro,^{5,6} and J. Lesgourgues¹

<https://classclapp.streamlit.app/> or <https://github.com/santiagocasas/clapp>

Example Questions:

1. Can you show me how to plot in python the temperature C_{ells} for CMB using classy?
2. Can you show me a python code to plot the ratio between the lensing C_{ells} when using two nonlinear models: hmcode and halofit?
3. Give me a python code that compares all nonlinear matter power spectrum methods available in classy and shows the ratio against halofit
4. Give me the growth rate as a function of z , between 0 and 3 for a LCDM cosmology
5. Show me the energy density of dark energy as a function of z , for a DESI-like cosmology with $w_0=-0.7$ and $w_a=-0.8$?

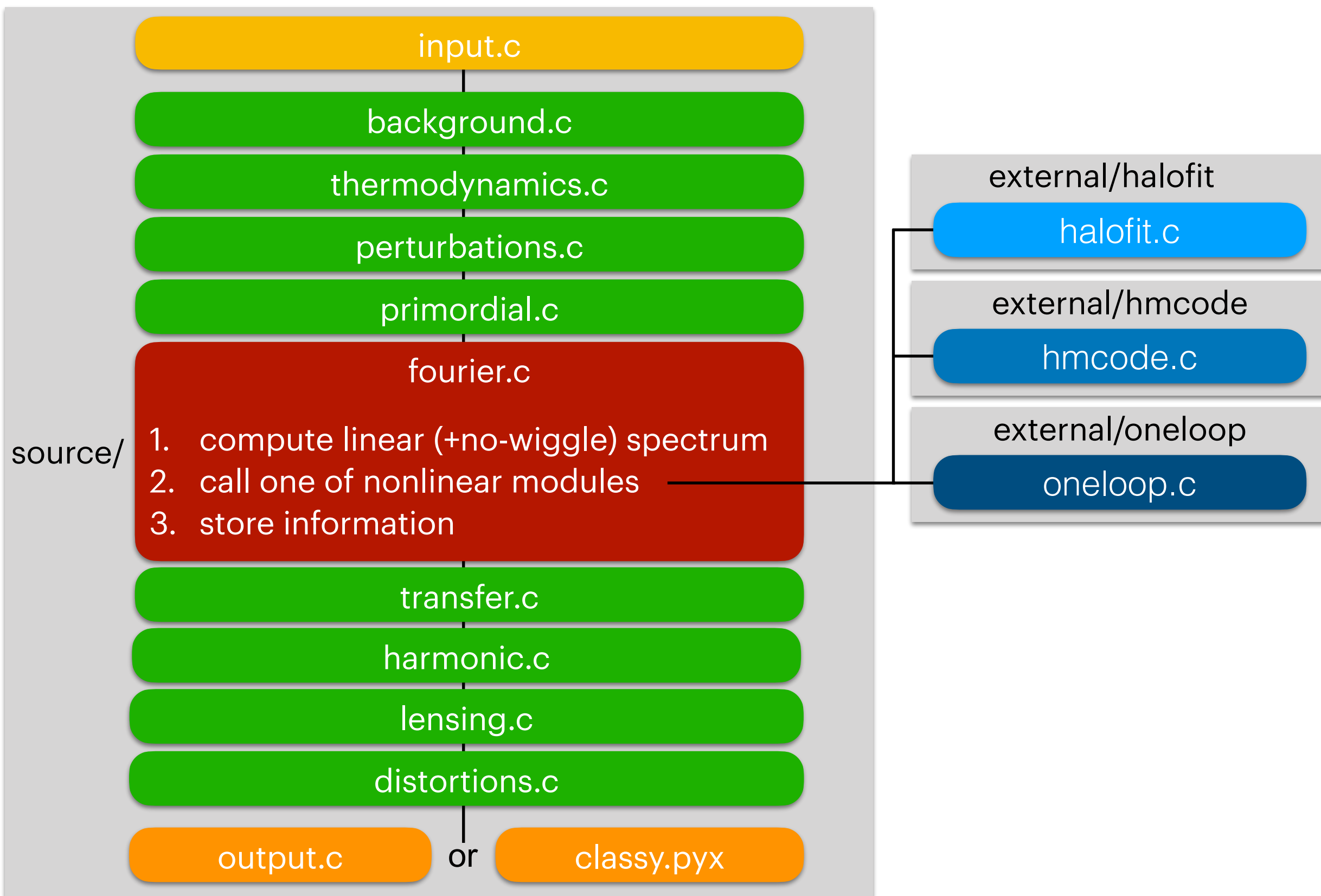
Range of application

- Lots of **cosmologies** within FLRW (incl. non-standard DM models, neutrinos, DE models)
- Lots of **observables** (incl. density Cl's, shear Cl's w/o Limber, HMcode 2020)
- Separate public **branches**: **GW_CLASS** for GW anisotropies, **ExoCLASS** for interface with particle physics codes, **class_matter** for FFTlog instead of line-of-sight...
- CLASS **interfaced** from MontePython, Cobaya, Cosmosis, CosmoPower, OLE <https://github.com/svenguenter/OLE> (see [arXiv:2503.13183](https://arxiv.org/abs/2503.13183), Günther et al.)

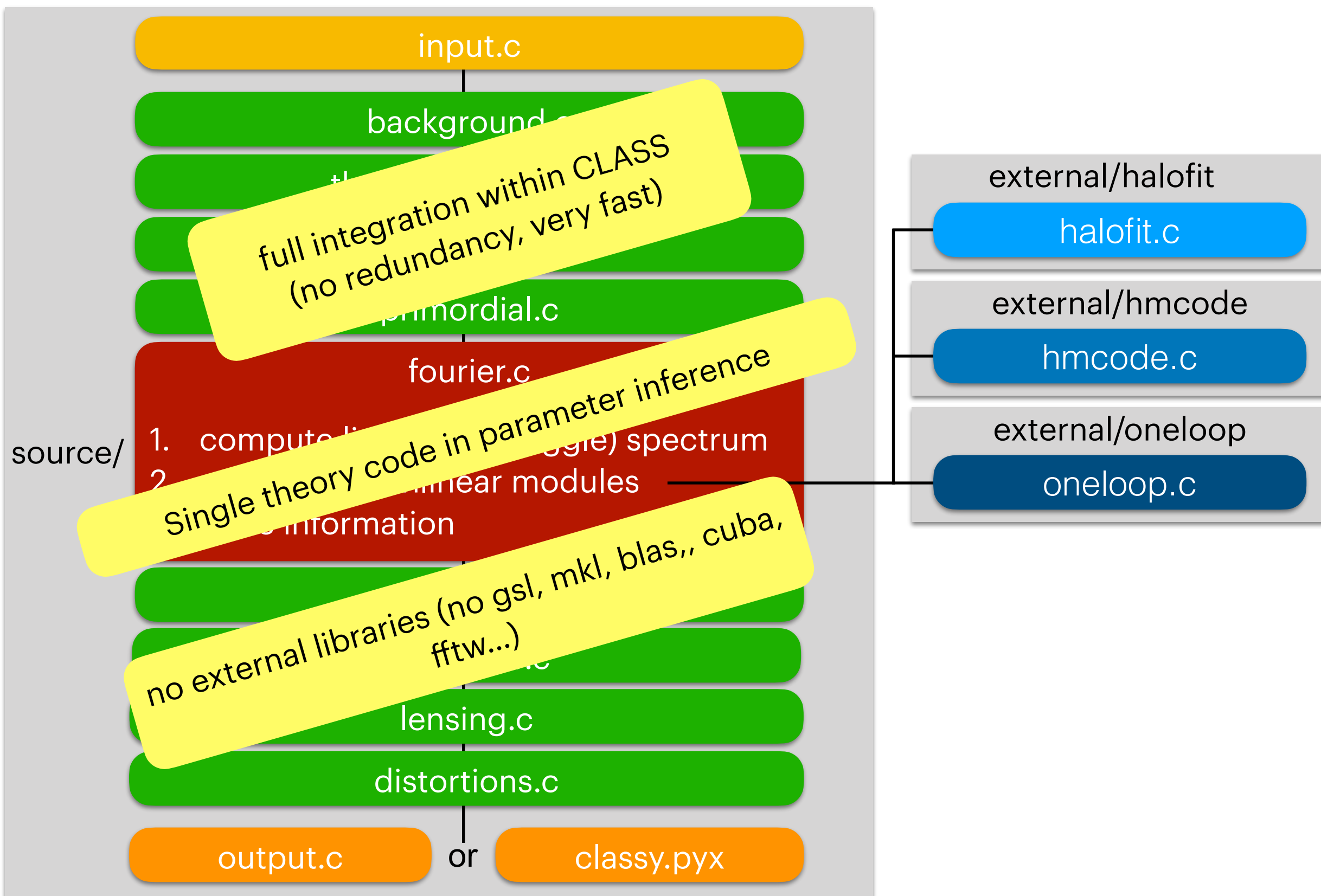
Range of application

- Lots of **cosmologies** within FLRW (incl. non-standard DM models, neutrinos, DE models)
- Lots of **observables** (incl. density Cl's, shear Cl's w/o Limber, HMcode 2020)
- Separate public **branches**: **GW_CLASS** for GW anisotropies, **ExoCLASS** for interface with particle physics codes, **class_matter** for FFTlog instead of line-of-sight...
- CLASS **interfaced** from MontePython, Cobaya, Cosmosis, CosmoPower, OLE <https://github.com/svenguenter/OLE> (see [arXiv:2503.13183](https://arxiv.org/abs/2503.13183), Günther et al.)
- Coming soon:
 - **3.4**: New architecture for **classy.py**, DM interacting with massive neutrinos, vector modes...
 - **3.5**: One-loop matter power spectrum and bispectrum of **EFTofLSS**

Overall structure of CLASS v3.5



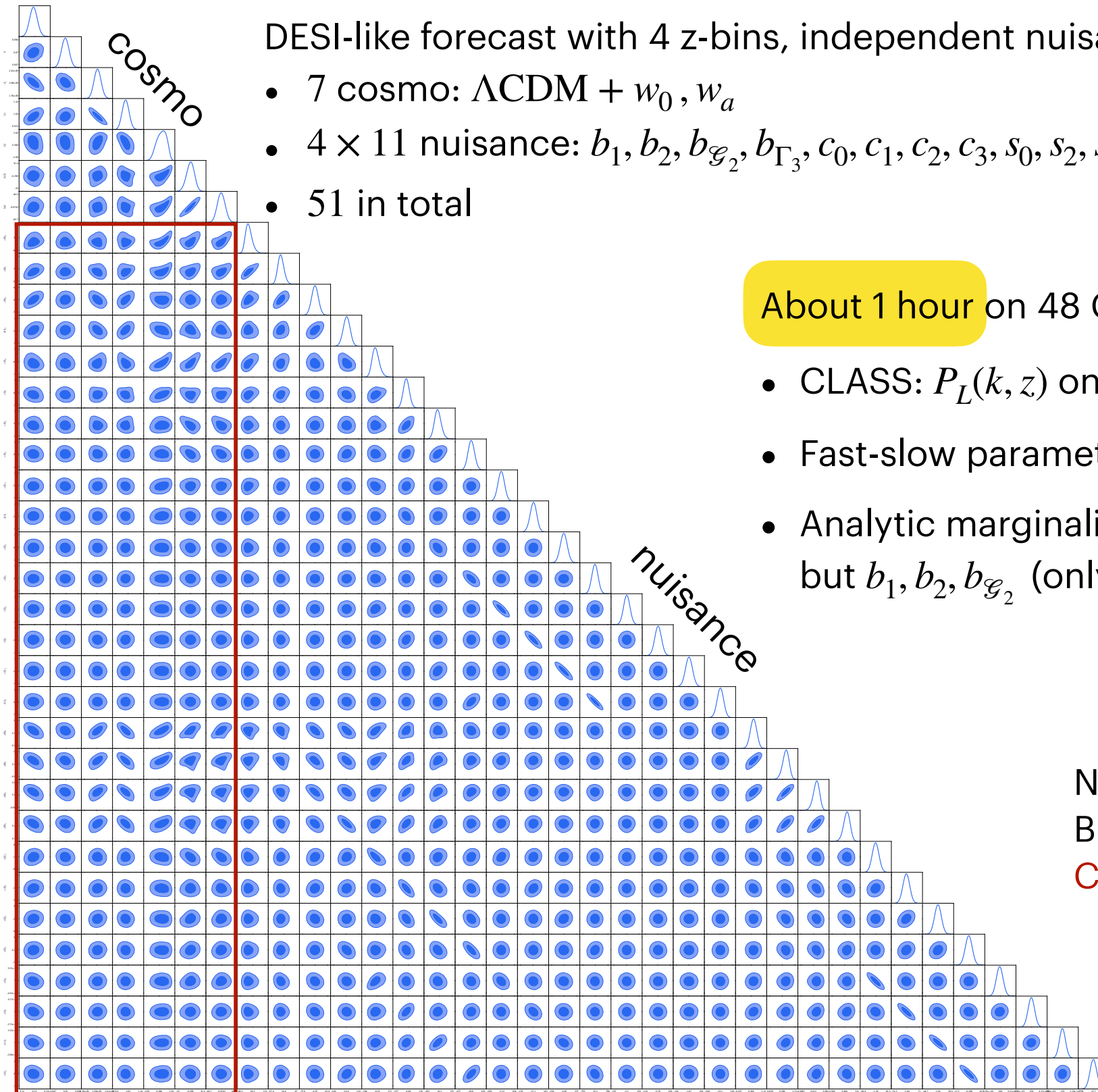
Overall structure of CLASS v3.5



What happens in the oneloop module ?

- ▶ Step 1 : compute log-Fourier transform of 42 convolution kernels $K_{ij}^{(n)}$ for 42 loop integrals
 - Cosmology-independent: once per MCMC (then cached)
 - takes **0.6 s**
- ▶ Step 2 : compute log-Fourier transform of linear spectrum P_{lin} into coefficients c_i , then compute 42 loops $L^{(n)} = c_i K_{ij}^{(n)} c_j$ at these z_k
 - Cosmology-dependent, but independent of nuisance parameter: once per CLASS call = every time slow parameters are varied (Cholesky decomposition)
 - takes **0.07s** in rest of CLASS + **0.09 s** in one-loop module
- ▶ Step 3 : assemble $P_X^{\text{rsd}}(k, z, \mu)$ or $P_{X,\ell=0,2,4}^{\text{rsd}}(k, z)$ for $X \in \{\text{matter, tracers, cross}\}$ for requested {bias} {counter-terms} {stochastic terms} { z } { μ or ℓ }
 - Depends on nuisance parameter = every time fast parameter are varied (Cholesky decomposition). No need to go through perturbation module again
 - takes **0.003 s**

MontePython run



DESI-like forecast with 4 z-bins, independent nuisance parameters across bins

- 7 cosmo: Λ CDM + w_0, w_a
- 4×11 nuisance: $b_1, b_2, b_{\mathcal{G}_2}, b_{\Gamma_3}, c_0, c_1, c_2, c_3, s_0, s_2, s_4$
- 51 in total

About 1 hour on 48 CPUs (8 chains on 6 cores each)...

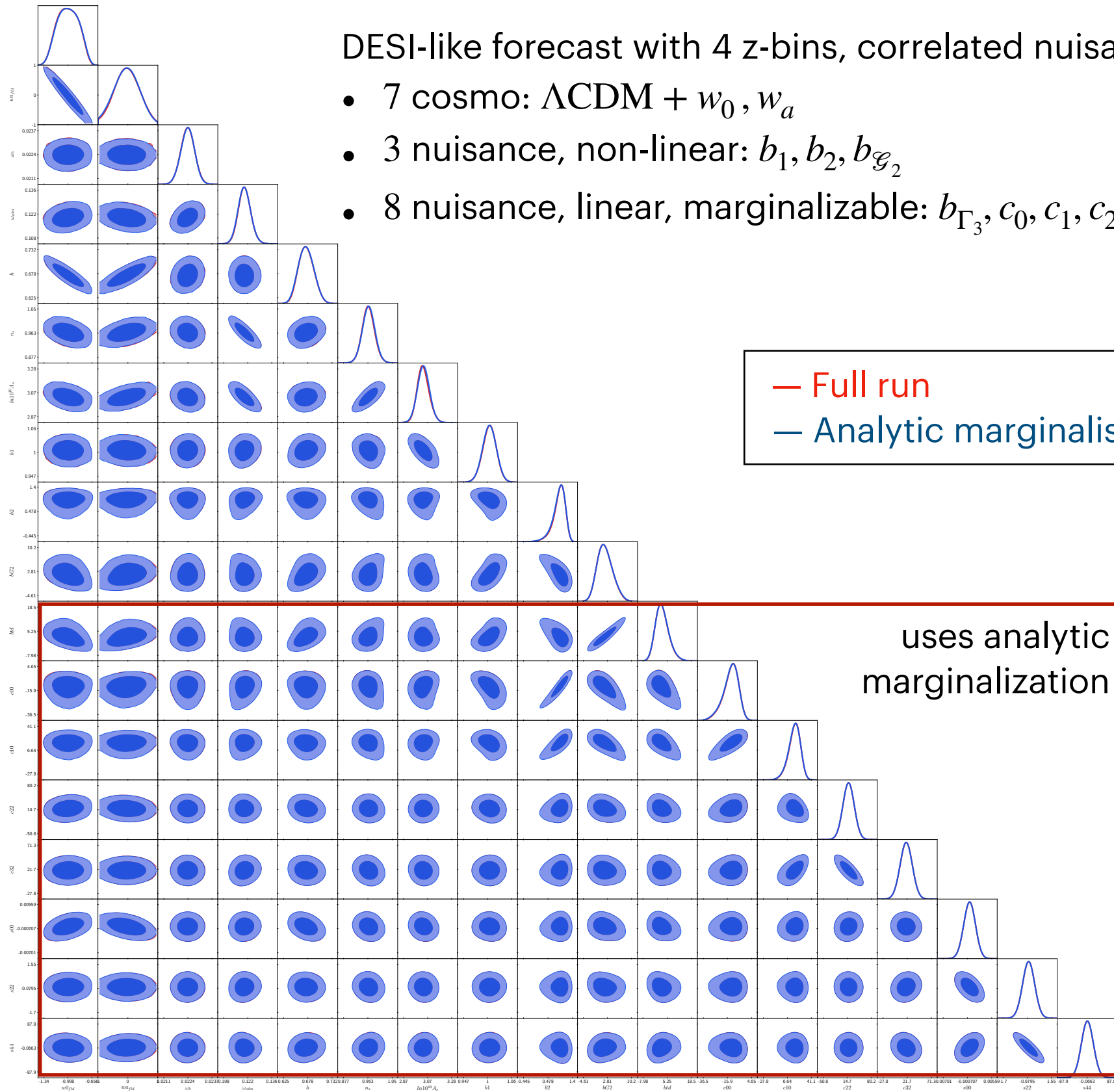
- CLASS: $P_L(k, z)$ only, OneLoop module adds 0.09s
- Fast-slow parameters (Cholesky)
- Analytic marginalisation over all nuisance param. but $b_1, b_2, b_{\mathcal{G}_2}$ (only 4×3 fast param.)

Need for emulator not obvious...
But compatible with **OLÉ**,
CONNECT, **CosmoPower**...

MontePython run

DESI-like forecast with 4 z-bins, correlated nuisance parameters across bins

- 7 cosmo: Λ CDM + w_0, w_a
- 3 nuisance, non-linear: $b_1, b_2, b_{\mathcal{E}_2}$
- 8 nuisance, linear, marginalizable: $b_{\Gamma_3}, c_0, c_1, c_2, c_3, s_0, s_2, s_4$



MontePython run

```
#-----Experiments to test (separated with commas)-----
```

```
data.experiments = ['bbn_shifted' , 'desi_elg_oneloop']
```

```
data.over_sampling = [1, 6]
```

```
#----- Parameter list -----
```

```
# Cosmological parameter list, names need to be understood by class
```

```
data.parameters['w0_fld'] = [-1., -2., 0., 1.2093e-01, 1., 'cosmo']
```

```
data.parameters['wa_fld'] = [ 0., -1., 1., 3.2132e-01, 1., 'cosmo']
```

```
data.parameters['omega_b'] = [ 0.022445, None, None, 3.6756e-04, 1., 'cosmo']
```

```
data.parameters['omega_cdm'] = [ 0.1200, None, None, 3.7164e-03, 1., 'cosmo']
```

```
data.parameters['h'] = [ 0.6736, None, None, 1.5357e-02, 1., 'cosmo']
```

```
data.parameters['n_s'] = [ 0.9649, None, None, 2.4577e-02, 1., 'cosmo']
```

```
data.parameters['ln10^{10}A_s'] = [ 3.044784, None, None, 5.1376e-02, 1., 'cosmo']
```

```
# Nuisance parameter list, same call, except the name does not have to be a class name
```

```
# - biases
```

```
data.parameters['b1'] = [1., None, None, 2.5482e-02, 1., 'nuisance']
```

```
data.parameters['b2'] = [1., None, None, 1.8695e-01, 1., 'nuisance']
```

```
data.parameters['bG2'] = [1., None, None, 1.6272e+00, 1., 'nuisance']
```

```
data.parameters['btd'] = [1., None, None, 2.6540e+00, 1., 'nuisance']
```

```
# - counterterms
```

```
data.parameters['c00'] = [-10., None, None, 7.7154e+00, 1., 'nuisance']
```

```
data.parameters['c10'] = [ 20., None, None, 1.4329e+01, 1., 'nuisance']
```

```
data.parameters['c22'] = [ 20., None, None, 2.3444e+01, 1., 'nuisance']
```

```
data.parameters['c32'] = [ 20., None, None, 1.8421e+01, 1., 'nuisance']
```

← Analytic marginalisation: nuisance -> derived_lkl

```
# - stochastic terms
```

```
data.parameters['s00'] = [0., None, None, 1.8107e-03, 1., 'nuisance']
```

```
data.parameters['s22'] = [0., None, None, 4.7421e-01, 1., 'nuisance']
```

```
data.parameters['s44'] = [0., None, None, 2.4973e+01, 1., 'nuisance']
```

```
# Fixed class parameters
```

```
data.cosmo_arguments['non_linear'] = 'oneloop'
```

```
data.cosmo_arguments['Omega_Lambda'] = 0.
```