Lecture VII: Spectra, Lensing, Output

Julien Lesgourgues

EPFL & CERN

Geneva, 03.03.2014

The spectra module

`source/spectra.c`

The function spectra_init()

The function `spectra_init()` calls (at most) three functions:

- `spectra_pk` computes the linear and non-linear matter power spectrum

$$P_L(k,z) = \left(\delta_m(k,\tau(z))\right)^2 \mathcal{P}(k)$$

  or in the case of several initial conditions,

$$P_L(k,z) = \sum_{ij} \delta_m^i(k,\tau(z))\, \delta_m^j(k,\tau(z)) \mathcal{P}_{ij}(k)$$

  (same for $P_{NL}(k,z)$ with extra factor R_{NL}^2). Result stored in `psp->ln_pk` and `psp->ln_pk_nl`.

- `spectra_sigma()` computes mean variance in sphere of radius R. By default, code calls it to get $\sigma_8(z=0)$ and stores it in `psp->sigma8`

The function spectra_init()

- `spectra_cl` computes the harmonic power spectra (formulae below holds in flat space)

$$C_l^{XY} = 4\pi \sum_{ij} \int \frac{dk}{k} \Delta_l^X(k) \Delta_l^Y(k) \mathcal{P}(k)$$

or in the case of several initial conditions,

$$C_l^{XY} = 4\pi \sum_{ij} \int \frac{dk}{k} \frac{1}{2} \left[\Delta_l^{iX}(k) \Delta_l^{jY}(k) + \Delta_l^{iX}(k) \Delta_l^{iY}(k) \right] \mathcal{P}_{ij}(k)$$

for $XY \in \{TT, TE, EE, BB, PP, TP, EP, N_i N_j, TN_i, PN_i, L_i L_j, TL_i, N_i L_j\}$ where $P \equiv$ CMB lensing, $N_i \equiv$ galaxy number count in i-th bin, and $L_i \equiv$ galaxy lensing in i-th bin.
Calculation takes place only for few values of l, later result can be interpolated at any l.

The result is stored in `psp->cl`.

External functions in spectra

Functions called later by `output` module, but you might need to use them directly when embedding CLASS in your own code or when writing your own `main` function.

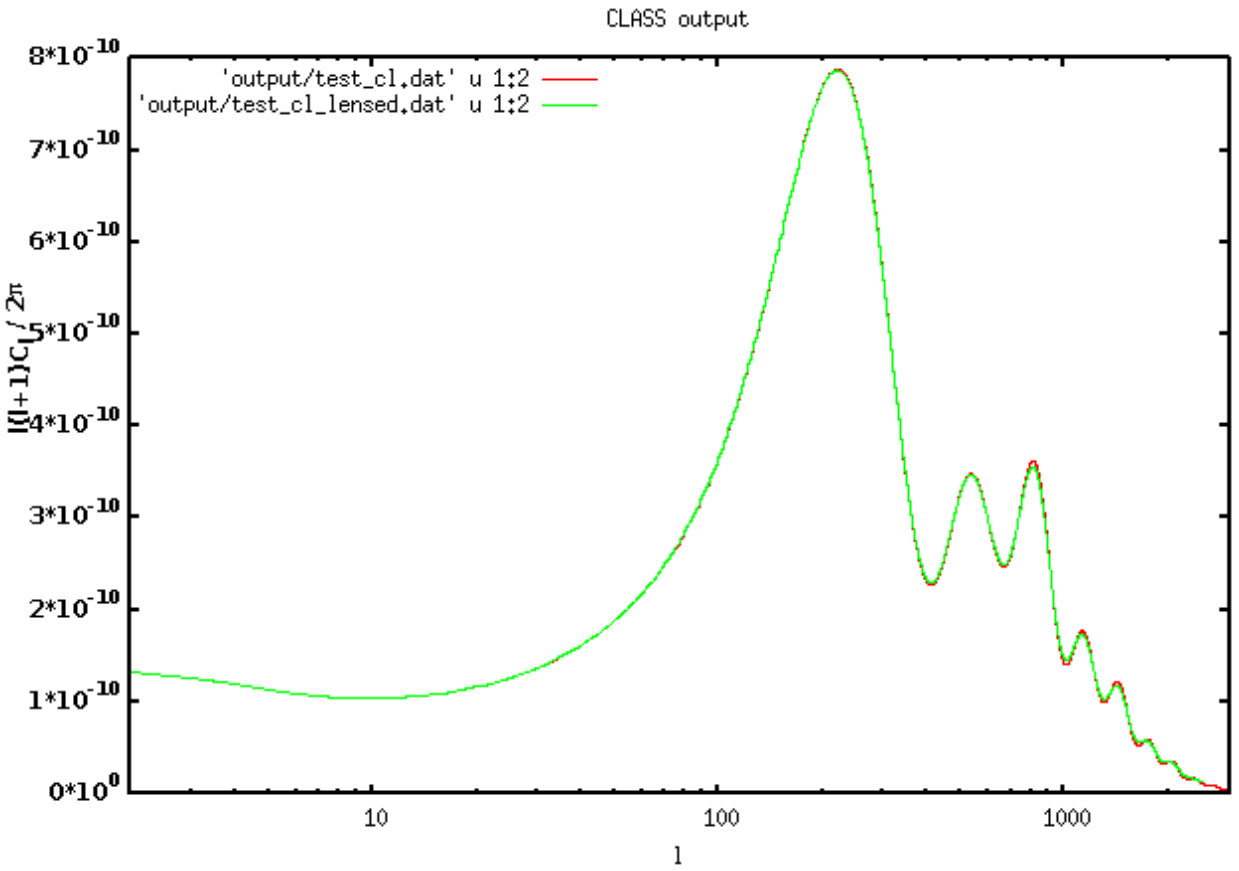
- `spectra_cl_at_l()` returns C_l's (either linear or non-linear) at a given l. Result can be a big array because returns result for each type, each i.c., each mode, and the total for each type.
- `spectra_pk_at_z()` returns $P_L(k,z)$ at given z for all k's.
- `spectra_pk_at_k_and_z()` returns $P_L(k,z)$ at given z and k.
- `spectra_pk_nl_at_z()` returns $P_{NL}(k,z)$ at given z for all k's.
- `spectra_pk_nl_at_k_and_z()` returns $P_{NL}(k,z)$ at given z and k.
- `spectra_sigma()` returns σ_R, i.e. the variance of matter fluctuations in a sphere of radius R, like the usual σ_8.
- `spectra_bandpower(l_1,l_2)` returns the bandpower $\sum_{l_1<l<l_2}(2l+1)C_l$

The lensing module

`source/lensing.c`

The function lensing_init()

Given the unlensed CMB spectra $C_l^{TT,TE,EE,BB}$ and the spectrum of the lensing potential C_l^{PP}, the goal is to compute the lensed spectra $\tilde{C}_l^{TT,TE,EE,BB}$.



The function lensing_init()

- follows the all-sky method of A. Challinor and A. Lewis.
- implemented in CLASS by S. Prunet.
- differs from CAMB only through numerical implementation (quadrature weights, etc.), not methodology. Results agree very well.
- module is switched on/off with `lensing = yes, no`. But it also requires at least `output= tCl, lCl` or `output= pCl, lCl` or `output= tCl, pCl, lCl` to output lensed spectra.

At the end of `lensing_init()`, `ple->cl_lens` contains a replica of tables in `psp->cl`, excepted that $C_l^{TT,TE,EE,BB}$ are replaced by their lensed counterpart.

External function: `lensing_cl_at_l()`, works like `spectra_cl_at_l()`, but returns only total lensed spectra (individual lensed spectra make no sense).

Called by `output` module or by external code. Contains really observable quantities: called by `classy.pyx` and Monte Python.

The output module

source/output.c

The output module

Called in the last place by `main/class.c` to write all requested output in files. Only writing, no physics, no manipulation of tables stored in other modules. Uses external interpolation functions of other modules. If CLASS embedded in another code, same information is obtained by directly calling such functions.

Exemple of wrappers in `test/test_loops.c`, `python/classy.pyx`, `cpp/ClassEngine.cc`

The future

- quintessence, decaying dark matter, all models in which background needs to be called with a shooting method at the beginning...
- coupled species: examples of CLASS implementation in the literature (Wilkinson et al.); unify and incorporate in master branch...
- semi-analytic methods for non-linear structure formation
- second-order CLASS... already exists! SONG by Guido Pettinari. Some of it will be progressively incorporated in CLASS, e.g. tree-level bispectrum
- more about modified gravity. Example of complicated model implementation in Audren, Blas, JL, Sibiryakov 2013. General parametrisation (Horndeski)
- automatic documentation
- automatic generator of `classy` wrapper
- more detailed python wrapper of all existing external functions: CLASS will become a python module with library of functions

CLASS Bibliography

- CLASS I: Overview, by J. Lesgourgues, arXiv:1104.2932 [astro-ph.IM] *General presentation and documentation*
- CLASS II: Approximation schemes, by D. Blas, J. Lesgourgues, T. Tram, arXiv:1104.2933 [astro-ph.CO], JCAP 1107 (2011) 034 *2nd order TCA, UFA, RSA, ndf15*
- CLASS III: Comparison with CAMB for LambdaCDM, by J. Lesgourgues, arXiv:1104.2934 [astro-ph.CO]
- CLASS IV: Efficient implementation of non-cold relics, by J. Lesgourgues, T. Tram, arXiv:1104.2935 [astro-ph.CO], JCAP 1109 (2011) 032
- Optimal polarisation equations in FLRW universes, by Thomas Tram, Julien Lesgourgues, JCAP 1310 (2013) 002 *Boltzmann hierarchies in curved spacetime*
- Fast and accurate CMB computations in non-flat FLRW universes, by Julien Lesgourgues, Thomas Tram, arXiv:1312.2697 [astro-ph.CO] *computation of hyperspherical bessel functions, useful formulas in curved space-time, approximations specific to curved space-time*
- The CLASSgal code for Relativistic Cosmological Large Scale Structure, by E. Di Dio, F. Montanari, J. Lesgourgues and R. Durrer, JCAP 1311 (2013) 044