CLASS

the Cosmological Linear Anisotropy Solving System[1]

Julien Lesgourgues
TTK, RWTH Aachen University

U. di Padova, 15-16.11.2021

[1] code developed by Julien Lesgourgues & Thomas Tram plus many others...

class physical content and overall organization

1. The 10 modules and their generic organization
2. Review of modules with emphasis on `background`, `thermodynamics`, `perturbations`, (`transfer`)

The 10 class modules

Executing class means going once through the sequence of modules:

```
 1. input.c              # parse/make sense of input parameters
                         # (advanced logic)
 2. background.c.        # homogeneous background
 3. thermodynamics.c.    # ionisation history, scattering rate
 4. perturbations.c.     # evolution of linear perturbations
                         # in Fourier space
 5. primordial.c.        # primordial spectrum, inflation
 6. fourier.c            # 2-point statistics in Fourier space:
                         # P(k), P_NL(k), sigma8...
 7. transfer.c.          # conversion from Fourier to harmonic
                         # space (line-of-sight integral)
 8. harmonic.c.          # 2-point stat. in harmonic space: C_l
 9. lensing.c            # CMB lensing
 10. distorsions.c       # CMB spectral distorsions
(+ 11. output.c)         # (print output in files)
```

Overall structure of class

In CLASS, what is a module?

- a file include/xxx.h containing some declarations
- a file source/xxx.c containing some functions
- each module is a associated with a structure xx, containing all what *other* modules need to know, and nothing else
- some fields in this structure are filled in the input.c module (input parameters relevant for this module)
- all other fields are filled by a function xxx_init(...)
- "executing a module" ≡ calling xxx_init(...)

In include/background.h: localise struct background
In source/background.c: localise background_init()

Overall structure of class

List of structures associated to modules:

module	structure	ab.	*	main content

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$,
and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

List of structures associated to modules:

module	structure	ab.	*	main content
input.c	precision	pr	ppr	all precision parameters

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of `class`

List of structures associated to modules:

module	structure	ab.	*	main content
`input.c`	`precision`	`pr`	`ppr`	all precision parameters
`background.c`	`background`	`ba`	`pba`	background quantities as funct. of τ

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of `class`

List of structures associated to modules:

module	structure	ab.	*	main content
`input.c`	`precision`	`pr`	`ppr`	all precision parameters
`background.c`	`background`	`ba`	`pba`	background quantities as funct. of τ
`thermodynamics.c`	`thermodynamics`	`th`	`pth`	thermo. quantities as funct. of z

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$,
and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of class

List of structures associated to modules:

module	structure	ab.	*	main content
input.c	precision	pr	ppr	all precision parameters
background.c	background	ba	pba	background quantities as funct. of τ
thermodynamics.c	thermodynamics	th	pth	thermo. quantities as funct. of z
perturbations.c	perturbations	pt	ppt	source (or transfer) functions $S^i(k,t)$

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of class

List of structures associated to modules:

module	structure	ab.	*	main content
input.c	precision	pr	ppr	all precision parameters
background.c	background	ba	pba	background quantities as funct. of τ
thermodynamics.c	thermodynamics	th	pth	thermo. quantities as funct. of z
perturbations.c	perturbations	pt	ppt	source (or transfer) functions $S^i(k,t)$
primordial.c	primordial	pm	ppm	primordial spectra $\mathcal{P}_\mathcal{R}(k)$,...

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of class

List of structures associated to modules:

module	structure	ab.	*	main content
input.c	precision	pr	ppr	all precision parameters
background.c	background	ba	pba	background quantities as funct. of τ
thermodynamics.c	thermodynamics	th	pth	thermo. quantities as funct. of z
perturbations.c	perturbations	pt	ppt	source (or transfer) functions $S^i(k,t)$
primordial.c	primordial	pm	ppm	primordial spectra $\mathcal{P}_\mathcal{R}(k)$,...
fourier.c	fourier	fo	pfo	2-point statistics (Fourier) $P(k,z)$,...

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0-\tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of class

List of structures associated to modules:

module	structure	ab.	*	main content
input.c	precision	pr	ppr	all precision parameters
background.c	background	ba	pba	background quantities as funct. of τ
thermodynamics.c	thermodynamics	th	pth	thermo. quantities as funct. of z
perturbations.c	perturbations	pt	ppt	source (or transfer) functions $S^i(k,t)$
primordial.c	primordial	pm	ppm	primordial spectra $\mathcal{P}_{\mathcal{R}}(k)$,...
fourier.c	fourier	fo	pfo	2-point statistics (Fourier) $P(k,z)$,...
transfer.c	transfer	tr	ptr	harmonic transfer functions $\Delta_l^i(k)$

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of `class`

List of structures associated to modules:

module	structure	ab.	*	main content
input.c	precision	pr	ppr	all precision parameters
background.c	background	ba	pba	background quantities as funct. of τ
thermodynamics.c	thermodynamics	th	pth	thermo. quantities as funct. of z
perturbations.c	perturbations	pt	ppt	source (or transfer) functions $S^i(k,t)$
primordial.c	primordial	pm	ppm	primordial spectra $\mathcal{P}_\mathcal{R}(k)$,...
fourier.c	fourier	fo	pfo	2-point statistics (Fourier) $P(k,z)$,...
transfer.c	transfer	tr	ptr	harmonic transfer functions $\Delta_l^i(k)$
harmonic.c	harmonic	hr	phr	2-point statistics (harmonic) C_ℓ's

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of `class`

List of structures associated to modules:

module	structure	ab.	*	main content
`input.c`	`precision`	`pr`	`ppr`	all precision parameters
`background.c`	`background`	`ba`	`pba`	background quantities as funct. of τ
`thermodynamics.c`	`thermodynamics`	`th`	`pth`	thermo. quantities as funct. of z
`perturbations.c`	`perturbations`	`pt`	`ppt`	source (or transfer) functions $S^i(k,t)$
`primordial.c`	`primordial`	`pm`	`ppm`	primordial spectra $\mathcal{P}_\mathcal{R}(k)$,...
`fourier.c`	`fourier`	`fo`	`pfo`	2-point statistics (Fourier) $P(k,z)$,...
`transfer.c`	`transfer`	`tr`	`ptr`	harmonic transfer functions $\Delta_l^i(k)$
`harmonic.c`	`harmonic`	`hr`	`phr`	2-point statistics (harmonic) C_ℓ's
`lensing.c`	`lensing`	`le`	`ple`	lensed CMB C_ℓ's

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of `class`

List of structures associated to modules:

module	structure	ab.	*	main content
`input.c`	`precision`	`pr`	`ppr`	all precision parameters
`background.c`	`background`	`ba`	`pba`	background quantities as funct. of τ
`thermodynamics.c`	`thermodynamics`	`th`	`pth`	thermo. quantities as funct. of z
`perturbations.c`	`perturbations`	`pt`	`ppt`	source (or transfer) functions $S^i(k,t)$
`primordial.c`	`primordial`	`pm`	`ppm`	primordial spectra $\mathcal{P}_\mathcal{R}(k)$,...
`fourier.c`	`fourier`	`fo`	`pfo`	2-point statistics (Fourier) $P(k,z)$,...
`transfer.c`	`transfer`	`tr`	`ptr`	harmonic transfer functions $\Delta_l^i(k)$
`harmonic.c`	`harmonic`	`hr`	`phr`	2-point statistics (harmonic) C_ℓ's
`lensing.c`	`lensing`	`le`	`ple`	lensed CMB C_ℓ's
`distorsions.c`	`distorsions`	`sd`	`psd`	CMB spectral distorsions

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

List of structures associated to modules:

module	structure	ab.	*	main content
input.c	precision	pr	ppr	all precision parameters
background.c	background	ba	pba	background quantities as funct. of τ
thermodynamics.c	thermodynamics	th	pth	thermo. quantities as funct. of z
perturbations.c	perturbations	pt	ppt	source (or transfer) functions $S^i(k,t)$
primordial.c	primordial	pm	ppm	primordial spectra $\mathcal{P}_\mathcal{R}(k)$,...
fourier.c	fourier	fo	pfo	2-point statistics (Fourier) $P(k,z)$,...
transfer.c	transfer	tr	ptr	harmonic transfer functions $\Delta_l^i(k)$
harmonic.c	harmonic	hr	phr	2-point statistics (harmonic) C_ℓ's
lensing.c	lensing	le	ple	lensed CMB C_ℓ's
distorsions.c	distorsions	sd	psd	CMB spectral distorsions
output.c	output	op	pop	description of output format

In a flat universe, line-of-sight integrals read $\Delta_l^i(k) = \int d\tau S^i(k,\tau) j_l(k(\tau_0 - \tau))$, and harmonic spectra are given by $C_l^{ij} = 4\pi \int \frac{dk}{k} \mathcal{P}(k) \Delta_l^i(k) \Delta_l^j(k)$.

Overall structure of `class`

Each module contains:

- a function `xxx_init(...)` filling the structure `xx`
- a function `xxx_free(...)` freeing all the memory allocated to this structure
- some functions `xxx_external_1(...)`, ..., `xxx_external_n(...)` that can be called from other modules (e.g. to read correctly or interpolate the content of the structure `xx`)
- some functions `xxx_internal_1(...)`, ..., `xxx_internal_m(...)` that are called only inside the module, within `xxx_init(...)`

Overall structure of `class`

Each module contains:

- a function `xxx_init(...)` filling the structure `xx`
- a function `xxx_free(...)` freeing all the memory allocated to this structure
- some functions `xxx_external_1(...)`, ..., `xxx_external_n(...)` that can be called from other modules (e.g. to read correctly or interpolate the content of the structure `xx`)
- some functions `xxx_internal_1(...)`, ..., `xxx_internal_m(...)` that are called only inside the module, within `xxx_init(...)`

Following order always respected in `xxx.c`:

```
xxx_external_1(...)
...
xxx_external_n(...)
xxx_init(...)
xxx_free(...)
xxx_internal_1(...)
...
xxx_internal_m(...)
```

Overall structure of class

Each module contains:

- a function `xxx_init(...)` filling the structure `xx`
- a function `xxx_free(...)` freeing all the memory allocated to this structure
- some functions `xxx_external_1(...)`, ..., `xxx_external_n(...)` that can be called from other modules (e.g. to read correctly or interpolate the content of the structure `xx`)
- some functions `xxx_internal_1(...)`, ..., `xxx_internal_m(...)` that are called only inside the module, within `xxx_init(...)`

Following order always respected in `xxx.c`:

```
xxx_external_1(...)
...
xxx_external_n(...)
xxx_init(...)
xxx_free(...)
xxx_internal_1(...)
...
xxx_internal_m(...)
```

Remark: a module in the CLASS code is very similar to a "class" in C++. We enjoy the structure of C++ with the speed and readability of C.

Overall structure of class

The main() function of CLASS located in main/class.c could only contain:

```
int main() {
 input_init_..(..,ppr,pba,pth,ppt,ptr,ppm,phr,pfo,ple,psd,
     pop);
 background_init(ppr,pba);
 thermodynamics_init(ppr,pba,pth);
 perturbations_init(ppr,pba,pth,ppt);
 primordial_init(ppr,ppt,ppm);
 fourier_init(ppr,pba,pth,ppt,ppm,pfo);
 transfer_init(ppr,pba,pth,ppt,pfo,ptr);
 harmonic_init(ppr,pba,ppt,ppm,pfo,ptr,phr);
 lensing_init(ppr,ppt,phr,pfo,ple);
 distorsions_init(ppr,pba,pth,ppt,ppm,psd)
 output_init(pba,pth,ppt,ppm,ptr,phr,pfo,ple,psd,pop)
 /* all calculations done, free the structures */
 distorsions_free(psd);
 lensing_free(ple);
 harmonic_free(phr);
 transfer_free(ptr);
 fourier_free(pfo);
 primordial_free(ppm);
 perturbations_free(ppt);
 thermodynamics_free(pth);
 background_free(pba);
}
```

Background module

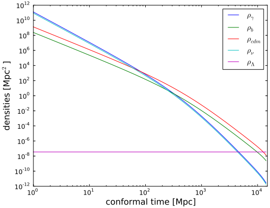
A. Background

• Get all background quantities as function of a time variable (`class v>3.0` $\rightarrow$ integration w.r.t. $\ln(a)$, but afterwards everything available as function of a, z, conformal time τ, proper time t)

• integration of Friedmann: $\frac{d\tau}{d\ln a} = \frac{1}{aH}$

• Gives mapping between $\tau \leftrightarrow a \leftrightarrow z \leftrightarrow t$

• Gives time evolution of all densities, pressures, Ω_m, Ω_r



• Gives time evolution of relevant cosmological distances and horizons, approximate (scale-independent) growth factors, varying fundamental constants...

Background module

Homogeneous units

Inside all modules *except thermodynamics*: everything in Mpc^n.

Examples:
- conformal time τ in Mpc, $H = \frac{a'}{a^2}$ in Mpc^{-1}
- $\rho_{\rm class} \equiv \frac{8\pi G}{3} \rho_{\rm physical}$ in Mpc^{-2}, such that $H = \left(\sum_i \rho_i - K/a^2 \right)^{1/2}$

Input/output can be in different units, then precised in comments of input/output files or in description of python functions.

Background module

Homogeneous units

Inside all modules *except thermodynamics*: everything in Mpc^n.

Examples:
- conformal time τ in Mpc, $H = \frac{a'}{a^2}$ in Mpc^{-1}
- $\rho_{\rm class} \equiv \frac{8\pi G}{3} \rho_{\rm physical}$ in Mpc^{-2}, such that $H = \left(\sum_i \rho_i - K/a^2\right)^{1/2}$

Input/output can be in different units, then precised in comments of input/output files or in description of python functions.

New in `class v>3.0`: a_0 absorbed everywhere

All quantities that should normally scale with some power of a_0^n are renormalised by a_0^{-n}, in order to be independent of a_0.

Examples:
- a in the code stands for a/a_0 in reality
- τ in the code stands for $a_0 \tau c$ in Mpc
- any prime in the code stands for $\frac{1}{a_0 c} \frac{d}{d\tau}$ in Mpc^{-1}
- r_x stands for any comoving radius times a_0

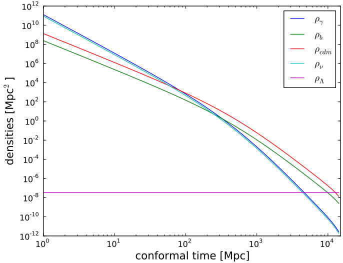
Background module

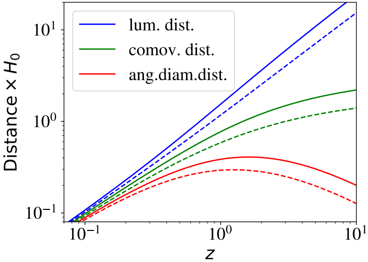
Retrieving background information when running C code from command line:

```
./class myinput.ini
```

1. with `background_verbose=1` or more, gives age, conformal age, $N_{\rm eff}$, $z_{\rm eq}$...
2. with `write_background=yes`, gives a table `output/myinput_background.dat` with many columns, at least:

1:z	2:proper time [Gyr]	3:conf. time [Mpc]
4:H [1/Mpc]	5:comov. dist.	6:ang.diam.dist.
7:lum. dist.	8:comov.snd.hrz.	9:(.)rho_g
10:(.)rho_b	11:(.)rho_cdm	12:(.)rho_lambda
13:(.)rho_ur	14:(.)rho_crit	15:(.)rho_tot
16:(.)p_tot	17:(.)p_tot_prime	
18:gr.fac. D	19:gr.fac. f	





Background module

Retrieving background information through the python wrapper in a script/notebook:

1. with function `background=xxx.get_background()`: get a dictionary identical to previous table:
   `dict_keys(['(.)rho_crit', 'lum. dist.', '(.)rho_b', 'H [1/Mpc]', 'conf. time [Mpc]', 'comov.snd.hrz.', '(.)rho_g', '(.)rho_lambda', 'comov. dist.', '(.)rho_cdm', 'ang.diam.dist.', 'proper time [Gyr]', 'gr.fac. D', 'gr.fac. f', 'z', '(.)rho_ur'])`
   (see example in `notebooks/distances.ipynb` or `scripts/distances.py`)
2. with `parameters=xxx.get_current_derived_parameters([..,..,...])`: get list of requested arguments, including:
   `'h', 'H0', 'Omega_Lambda', 'Omega0_fld', 'age', 'conformal_age', 'm_ncdm_in_eV', 'm_ncdm_tot', 'Neff', 'Omega_m', 'omega_m', ...`
   (see example in `notebooks/distances.ipynb` or `scripts/distances.py`)
3. additional specific functions to retrieve background quantities:
   `.Hubble(z), .angular_distance(z), .luminosity_distance(z), .scale_independent_growth_factor(z), .scale_independent_growth_factor_f(z),`
   (see example in `notebooks/warmup.ipynb` or `scripts/warmup.py`)

Background module

Classification of variables in `background` module:

In general, three types of parameters:

- $\{A\}$ which can be expressed directly at any given time, as a function of a or additional variables $\{B\}$.
- $\{B\}$, which need to be integrated w.r.t. $\ln(a)$ through 1st-order diff. eqs.
- $\{C\}$, which also need to be integrated w.r.t. $\ln(a)$ but are not used for $\{A\}$.

ΛCDM and many simple extensions:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ..., \}$ with e.g. $H(a) = \left(\sum_X \rho_x(a) - \frac{K}{a^2}\right)^{1/2}$
- $\{B\} = \varnothing$
- $\{C\} = \{\tau, t, r_s, \text{growth factors}\}$ with e.g. $\frac{d\tau}{d\ln a} = \frac{1}{aH}, \frac{dt}{d\ln a} = \frac{1}{H}, \frac{dr_s}{d\tau} = \frac{c_s^2}{aH}$

Background module

Classification of variables in `background` module:

In general, three types of parameters:

- $\{A\}$ which can be expressed directly at any given time, as a function of a or additional variables $\{B\}$.
- $\{B\}$, which need to be integrated w.r.t. $\ln(a)$ through 1st-order diff. eqs.
- $\{C\}$, which also need to be integrated w.r.t. $\ln(a)$ but are not used for $\{A\}$.

ΛCDM and many simple extensions:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ...,\}$ with e.g. $H(a) = \left(\sum_X \rho_x(a) - \frac{K}{a^2}\right)^{1/2}$
- $\{B\} = \varnothing$
- $\{C\} = \{\tau, t, r_s, \text{growth factors}\}$ with e.g. $\frac{d\tau}{d\ln a} = \frac{1}{aH}$, $\frac{dt}{d\ln a} = \frac{1}{H}$, $\frac{dr_s}{d\tau} = \frac{c_s^2}{aH}$

Exemple of DE/DM/DR fluid:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ..., w_{\rm fld}(a), \rho_{\rm fld}(\rho_{\rm fld})\}$
- $\{B\} = \{\rho_{\rm fld}\}$ with $\frac{d\rho_{\rm fld}}{d\ln a} = -3(1 + w_{\rm fld}(a))\rho_{\rm fld}$

Background module

Classification of variables in `background` module:

In general, three types of parameters:

- $\{A\}$ which can be expressed directly at any given time, as a function of a or additional variables $\{B\}$.
- $\{B\}$, which need to be integrated w.r.t. $\ln(a)$ through 1st-order diff. eqs.
- $\{C\}$, which also need to be integrated w.r.t. $\ln(a)$ but are not used for $\{A\}$.

ΛCDM and many simple extensions:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ...,\}$ with e.g. $H(a) = \left(\sum_X \rho_x(a) - \frac{K}{a^2}\right)^{1/2}$
- $\{B\} = \varnothing$
- $\{C\} = \{\tau, t, r_s, \text{growth factors}\}$ with e.g. $\frac{d\tau}{d\ln a} = \frac{1}{aH}$, $\frac{dt}{d\ln a} = \frac{1}{H}$, $\frac{dr_s}{d\tau} = \frac{c_s^2}{aH}$

Exemple of DE/DM/DR fluid:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ..., w_{\rm fld}(a), \rho_{\rm fld}(\rho_{\rm fld})\}$
- $\{B\} = \{\rho_{\rm fld}\}$ with $\frac{d\rho_{\rm fld}}{d\ln a} = -3(1 + w_{\rm fld}(a))\rho_{\rm fld}$

Exemple of extended cosmology with quintessence ϕ:

- $\{A\} = \{\rho_i, p_i, H, ..., V(\phi), \rho_\phi(\phi, \phi')\}$ with e.g. $\rho_\phi(\phi, \phi') = \frac{1}{2}(\phi')^2 + V(\phi)$
- $\{B\} = \{\phi, \phi'\}$ with $\frac{d\phi}{d\ln a} = \frac{\phi'}{aH}$, $\frac{d\phi'}{d\ln a} = -2\phi' - \frac{a}{H}V(\phi)$

Background module

Classification of variables in `background` module:

In general, three types of parameters:

- $\{A\}$ which can be expressed directly at any given time, as a function of a or additional variables $\{B\}$.
- $\{B\}$, which need to be integrated w.r.t. $\ln(a)$ through 1st-order diff. eqs.
- $\{C\}$, which also need to be integrated w.r.t. $\ln(a)$ but are not used for $\{A\}$.

ΛCDM and many simple extensions:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ..., \}$ with e.g. $H(a) = \left(\sum_X \rho_x(a) - \frac{K}{a^2}\right)^{1/2}$
- $\{B\} = \varnothing$
- $\{C\} = \{\tau, t, r_s, \text{growth factors}\}$ with e.g. $\frac{d\tau}{d\ln a} = \frac{1}{aH}, \frac{dt}{d\ln a} = \frac{1}{H}, \frac{dr_s}{d\tau} = \frac{c_s^2}{aH}$

Exemple of DE/DM/DR fluid:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ..., w_{\rm fld}(a), \rho_{\rm fld}(\rho_{\rm fld})\}$
- $\{B\} = \{\rho_{\rm fld}\}$ with $\frac{d\rho_{\rm fld}}{d\ln a} = -3(1 + w_{\rm fld}(a))\rho_{\rm fld}$

Exemple of extended cosmology with quintessence ϕ:

- $\{A\} = \{\rho_i, p_i, H, ..., V(\phi), \rho_\phi(\phi, \phi')\}$ with e.g. $\rho_\phi(\phi, \phi') = \frac{1}{2}(\phi')^2 + V(\phi)$
- $\{B\} = \{\phi, \phi'\}$ with $\frac{d\phi}{d\ln a} = \frac{\phi'}{aH}, \frac{d\phi'}{d\ln a} = -2\phi' - \frac{a}{H}V(\phi)$

Also Cold Dark Matter decaying into Dark Radiation...

- $\{A\} = \{\rho_i, p_i, H, ..., \rho_{\rm dcdm}, \rho_{\rm dr}\}$
- $\{B\} = \{\rho_{\rm dcdm}, \rho_{\rm dr}\}$ with $\frac{d\rho_{\rm dcdm}}{d\ln a} = -3\rho_{\rm dcdm} - \frac{a}{H}\Gamma(a)\,\rho_{\rm dcdm}$

Background module

External functions in background module:

```
background_at_z(....)  # interpolates all background
                       # quantities {A,B,C} at given z
background_at_tau(...) # interpolates all background
                       # quantities {A,B,C} at given tau
background_tau_of_z(...) # conversion tau(z)
background_z_of_tau(...) # conversion z(tau)
background_functions(...) # direct analytic expression
                          # of {A} given a,{B}
background_w_fld(...)         # direct analytic expression
                              # of w(a) for fluid
background_varconst_of_z(...) # direct analytic expression
                              # of alpha(a), ...
```

Background module

Internal functions in background module with technical role:

```
# common to all modules
background_init(...)
background_free(...)
background_free_noinput(...)
background_free_input(...)
background_indices(...)

# solves ODE d{B,C}/dlna=...
background_solve(...) # calls generic_evolver(...)
background_sources(...) # technical for generic_evolver(...)
background_timescale(...) # technical for        ''

# extract data from pba->background_table
# for output in file (with write_background)
# or through wrapper (with .get_background())
background_output_titles(...) # write header
background_output_data(...)   # extract one row of values
```

Background module

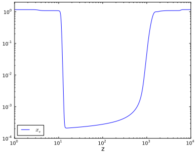
Internal functions in background module with the physics (in addition to background_functions(...), _w_fld(...), _varconst_of_z(...):

```
# for ncdm species with psd (massive nu's, WDM, ...)
background_ncdm_distribution(...) # defines actual psd f(q)
background_ncdm_test_function(...)
background_ncdm_init(...)
background_ncdm_momenta(...)
background_ncdm_M_from_Omega(...)

background_checks(...) # input consistency checks

# for ODE: d{B,C}/dlna=...
background_initial_conditions(...) # ICs
background_derivs(...) # actual differential equations
                       # (calls background_function(), ...)

 background_find_equality(...) # get tau_eq, z_eq

 # detailed summary of cosmo. params if input_verbose>1
 background_output_budget(...)

 # for scalar field (quintessence): potential, ...
 V_scf(...), dV_scf(...), ddV_scf(...), Q_scf(...)
```

B. Thermodynamics

Get all thermodynamics quantities as a function of a time variable (class $\rightarrow$ redhsift z) after integrating differential equations like recombination equations:

$$\frac{dx_e}{dz} = \text{excitation, ionization}$$
$$\frac{dT_b}{dz} = \text{expansion, heating}$$



Then $x_e(z) \rightarrow \kappa'(z)$ (Thomson scattering rate)and its higher derivatives
$\rightarrow \kappa(z)$ (Optical depth) and its exponential
$\rightarrow g(z)$ (visibility function for Sachs-Wolfe effect) and its derivative
$\rightarrow \tau_d(z)$ (baryon drag optical depth)
$\rightarrow r_d(z)$ (approximate photon comoving damping scale)
while $T_b(z) \rightarrow w_b(z)$ (baryon e.o.s parameter)
$\rightarrow c_b^2(z)$ (baryon sound speed) and its derivatives
Plus possibly: exotic scattering rates, optical depth, visibility, temperature, sound speed in Dark Sector

Thermodynamics module

Essential steps:

1. solve ODE for $\frac{dx_\mathrm{H}}{dz} = ...$, $\frac{dx_\mathrm{He}}{dz} = ...$, $\frac{dT_b}{dz} = ...$ (plus possibly Dark Sector quantities)
   - $\frac{dT_b}{dz}$ always computed inside the module and integrated over time. ODE system contains at least $T_b(z)$.
   - in general $\frac{dx_\mathrm{H}}{dz}$ and $\frac{dx_\mathrm{He}}{dz}$ computed at each z by an external code: either HyRec2020 in `external/HyRec2020` (Ali-Haimoud & Lee, default), or RecFastCLASS in `external/RecfastCLASS` (Recfast v1.5 authors + Meinert, Schoeneberg). They are part of the ODE system, and integrated internally by CLASS.
   - at very high redshift, their value is imposed by some approximations. They are removed from the ODE system.
   - at each step, compute possible contribution of exotic energy injection (DM annihilation/decay, PBH accretion/evaporation) described in 1910.04619 and coded in `external/heating/injection.c`; add it internally e.g. to $\frac{dT_b}{dz}$ or pass it to HyRec2020/RecFastCLASS.
   - at very low redshift, contribution of reionization added to the solution of the ODE.
2. infer $x_e = x_\mathrm{H} + \frac{n_\mathrm{He}}{n_\mathrm{H}} x_\mathrm{He}$
3. infer all other variables $\kappa'(z)$, $g(z)$, etc.

Thermodynamics module

Primordial helium fraction Y_{He} can be:

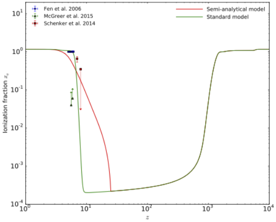
- passed in input by user
- (default:) inferred from $(\omega_b, N_{\mathrm{eff}})$ using standard BBN interpolation table produced by PArthENoPE v1.2 and stored in `external/bbn/sBBN_2017.dat`

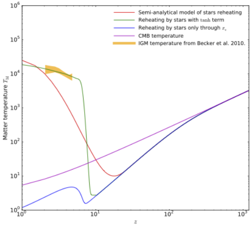
Thermodynamics module

Retrieving thermodynamics information when running C code from command line:
`./class myinput.ini`

1. with `thermodynamics_verbose=1` or more, gives $Y_{\rm He}$, plus characteristic redshifts $z_{\rm rec}$ (recombination from max. visibilty function), z_* (recombination $\kappa = 1$), z_d (baryon drag), $z_{\rm reio}$ (reionization) and the value of several quantities at this time...
2. with `write_thermodynamics=yes`, gives a table `output/myinput_thermodynamics.dat` with many columns, at least:

1:z	2:conf. time [Mpc]	3:x_e
4:kappa' [Mpc^-1]	5:exp(-kappa)	6:g [Mpc^-1]
7:Tb [K]	8:dTb [K]	9:w_b
10:c_b^2	11:tau_d	





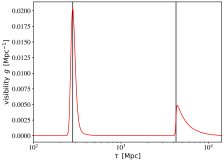
Thermodynamics module

Retrieving thermodynamics information through the python wrapper in a script/notebook:

1. with function `thermodynamics=xxx.get_thermodynamics()`: get a dictionary identical to previous table:

```
dict_keys(['x_e', 'g [Mpc^-1]', 'conf. time
 [Mpc]', "kappa' [Mpc^-1]", 'tau_d', 'Tb [K
]', 'c_b^2', 'exp(-kappa)', 'z'])
```

(see example in `notebooks/thermo.ipynb` or `scripts/thermo.py`)



2. with `parameters=xxx.get_current_derived_parameters([..,...,...])`: get list of requested arguments, including: `'YHe'`, `'tau_reio'`, `'z_reio'`, `'z_rec'`, `'tau_rec'`, `'rs_rec'`, `'rs_rec_h'`, `'ds_rec'`, `'ds_rec_h'`, `'ra_rec'`, `'ra_rec_h'`, `'da_rec'`, `'da_rec_h'`, `'100*theta_s'`, `'z_star'`, `'tau_star'`, `'rs_star'`, `'rs_star_h'`,`'ds_star'`,`'ds_star_h'`, `'ra_star'`, `'ra_star_h'`, `'da_star'`, `'da_star_h'`, `'100*theta_star'`, `'z_d'`, `'tau_d'`, `'ds_d'`, `'ds_d_h'`, `'rs_d'`, `'rs_d_h'`,
(see example in `notebooks/thermo.ipynb` or `scripts/thermo.py`)

3. additional specific functions to retrieve background quantities: `.ionisation_fraction(z)`, `.baryon_temperature(z)`

Thermodynamics module

Important functions in thermodynamics:

```
# external
thermodynamics_at_z(...) # all quantities at z

# common to all modules
thermodynamics_init(...)
thermodynamics_free(...)

# solves ODE dTb/dlna=..., etc.
thermodynamics_solve(...) # calls generic_evolver(...)
thermodynamics_derivs(...) # ODEs for Tb and maybe others.
                           # calls HyRec2020/RecFastCLASS
thermodynamics_ionization_fractions(...) # approximations
 # for recombination, may superseed HyRec2020/RecFastCLASS
thermodynamics_reionization_function(...) # reionization

# extract data from pth->thermodynamics_table
# for output in file (with write_thermodynamics)
# or through wrapper (with .get_thermodynamics())
thermodynamics_output_titles(...) # write header
thermodynamics_output_data(...)   # extract one row
```

C. Perturbations

- Find all perturbations ($\delta_X(\tau, k)$, $\phi(\tau, k)$, ...) by integrating ODEs for each independent wavenumber k, each mode (scalar/vector[1]/tensor), each initial condition (adiabatic/isocurvature):
  - Boltzmann (non-perfect fluids: photon temperature/polarization, massless/massive neutrino temperature)
  - Continuity + Euler (perfect fluid: baryons, hypothetical (DE/DM/DR) fluid) or approximatively pressureless species: (CDM)
  - linearized Einstein equations (one = differential equation, others = constraint equations)

  Perturbations normalized to conventional initial condition (`class` $\rightarrow$ curvature $\mathcal{R}(\vec{k}) = 1$ for scalars with adiabatic I.C.), in reality: transfer functions.

Equations follow literally notations of Ma & Bertschinger 1996, `astro-ph/9506072`

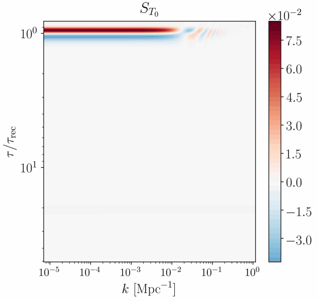
Multi-gauge code: everything coded in newtonian gauge or synchronous gauge. Option: output everything in N-body gauge. Structure ready for more gauges.

[1] in `class` $\rightarrow$ vector perturbation equations present just in case, but never used: no implemented scenario where vectors are relevant, no vector I.C. and observables.

Perturbation module

- Keep memory not of everything, but anything useful for final calculation of observables:
  - raw transfer functions ($\delta_x(\tau, k)$, $\theta_x(\tau, k)$, metric)
  - linear combinations like $\delta_m(\tau, k) \rightarrow P_m(k, z)$
  - additional non-trivial combinations (photon, baryon, metric, thermodynamical functions) $\rightarrow$ CMB source functions $S_{T_i}(k, \tau)$, $S_P(k, \tau)$

All these are called *source functions* in `class`



Perturbation module

Two approaches to polarization in Boltzmann hierarchy:

- Ma & Bertschinger 1994:
  $(F_\ell, G_\ell) \rightarrow (S_T, S_P) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $2\ell_{\max}$ equations!
- Hu & White 1997:
  $(\Theta_\ell, E_\ell, B_\ell) \rightarrow (S_T, S_E, S_B) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $3\ell_{\max}$ equations!

Perturbation module

Two approaches to polarization in Boltzmann hierarchy:

- Ma & Bertschinger 1994:
  $(F_\ell, G_\ell) \rightarrow (S_T, S_P) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $2\ell_{\rm max}$ equations!
- Hu & White 1997:
  $(\Theta_\ell, E_\ell, B_\ell) \rightarrow (S_T, S_E, S_B) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $3\ell_{\rm max}$ equations!

CMBFAST: first in flat space, second in curved space

Perturbation module

Two approaches to polarization in Boltzmann hierarchy:

- Ma & Bertschinger 1994:
  $(F_\ell, G_\ell) \rightarrow (S_T, S_P) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $2\ell_{\rm max}$ equations!
- Hu & White 1997:
  $(\Theta_\ell, E_\ell, B_\ell) \rightarrow (S_T, S_E, S_B) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $3\ell_{\rm max}$ equations!

CMBFAST: first in flat space, second in curved space

CAMB: always second case

Perturbation module

Two approaches to polarization in Boltzmann hierarchy:

- Ma & Bertschinger 1994:
  $(F_\ell, G_\ell) \rightarrow (S_T, S_P) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $2\ell_{\rm max}$ equations!
- Hu & White 1997:
  $(\Theta_\ell, E_\ell, B_\ell) \rightarrow (S_T, S_E, S_B) \rightarrow (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $3\ell_{\rm max}$ equations!

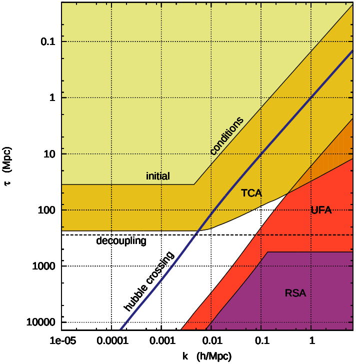
CMBFAST: first in flat space, second in curved space

CAMB: always second case

CLASS: always first case, thanks to new analytic results in curved space (T. Tram & JL, JCAP 2013 [arXiv:1305.3261]; Pitrou, Pereira & JL, Phys.Rev.D 2020 [arXiv:2005.12119])

Perturbation module

The approximation scheme (CLASS II & CLASS IV 2011)



- Tight Coupling Approximation for baryons and γ at 2nd order
- Ultrarelativistic Fluid Approximation (for massless ν, also one for massive ones): truncated Boltzmann, 3 equations
- Radiation Streaming Approximation (for photons and massless ν): test particles, 0 equations

Perturbation module

Like in background and thermodynamics, use of generic_evolver(...) which may point at:

- rkck: 4th-order adaptive-step Runge-Kutta
- ndf15 (default): an ODE solver customized for Einstein-Boltzmann solvers:
  - Stiff system require implicit method like backward Euler or more advanced:
    $\rightarrow$ find y_{n+1} as a solution of $y_{n+1} = y_n + y'(y_{n+1})\delta t$
  - Should still be fast: Newton method with Jacobian recycling
  - Robustness requires δt to be determined automatically (adaptive time step)
  - Source function required at predefined t_i: integrator must interpolate on-the-fly at these valkues
  - System is sparse: some algebra gives big speed up (sparse LU decomposition)

Everything gathered in ndf15 by T. Tram (CLASS II 2011).
TCA could even be removed!

Perturbation module

Retrieving information on transfer/source functions when running C code from command line: `./class myinput.ini`

1. with `output=...,dTk,...`: gives density transfer functions at selected times for each species in output file `output/myinput_tk(_z0).dat`. Many columns, at least:
   `1:k (h/Mpc)   2:d_g   3:d_b   4:d_cdm   5:d_ur   6:d_tot   7:phi   8:psi`
2. with `output=...,vTk,...`: adds velocity transfer functions at selected times to output file `output/myinput_tk.dat`:
   `9:t_g   10:t_b   11:t_cdm   12:t_tot`
3. with `k_output_values = 0.01, 0.1, ...`: gives time evolution of selected modes in output file `output/myinput_perturbations_k0_s.dat`, etc. Many columns:
   `1:tau [Mpc]   2:a   3:delta_g   4:theta_g   5:shear_g`   etc.

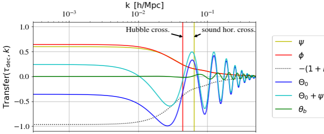
Perturbation module

Retrieving background information through the python wrapper in a script/notebook:

1. with function `transfers=xxx.get_transfers()`: get a dictionary of transfer functions at selected times:

```
dict_keys(['phi', 'psi', 't_cdm', 't_b', '
d_tot', 't_g', 'd_ur', 'd_cdm', 'd_b', '
t_tot', 't_ur', 'd_g', 'k (h/Mpc)'])
```

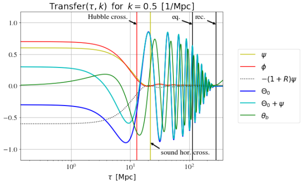
(see example in `notebooks/one_time.ipynb` or `scripts/one_time.py`)



2. with `parameters=xxx.get_perturbations()`: get a dictionary of transfer functions for the wavenumbers selected with the input parameter 'k_output_values':'0.001, 0.01,0.1':

```
dict_keys(['a', 'theta_g', 'phi', 'pol0_g',
 'theta_b', 'theta_ur', 'shear_ur', '
shear_g', 'tau [Mpc]', 'theta_cdm', '
delta_ur', 'psi', 'pol2_g', 'delta_g', '
delta_cdm', 'pol1_g', 'delta_b'])
```

(see example in `notebooks/one_k.ipynb` or `scripts/one_k.py`)



Perturbation module

Important functions in perturbations:

```
# external
perturbations_sources_at_tau(...) # all sources at tau

# common to all modules
perturbations_init(...)
perturbations_free(...)

# solves ODE
perturbations_solve(...) # calls generic_evolver(...)
perturbations_derivs(...) # ODEs for all perturbations
perturbations_einstein(...) # linearised Einstein equations
perturbations_total_stress_energy(...) # delta T^mu^nu
perturbations_sources(...) # assembles output sources

# used only for k_output_value or get_perturbations()
perturbations_print_variables(...)

# extract data from ppt->sources
# for output in file (with dTk, vTk)
# or through wrapper (with .get_transfers())
perturbations_output_titles(...) # write header
perturbations_output_data(...)   # extract one row
```

D. Primordial spectra

Initial conditions for scalars (adiabatic, isocurvature) and tensors. Linear theory ⇔ Gaussian independent Fourier modes ⇔ only need primordial power spectra

- analytic mode: primordial power spectra as parametric functions (e.g. power-law)
- inflation mode: solve background+perturbation equation for single-field inflation and compute primordial scalar/tensor spectrum numerically

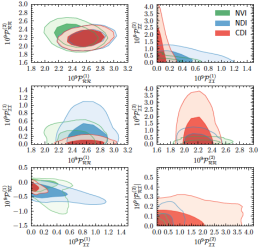


Fig. 22. Two dimensional distributions for power in isocurvature modes, using *Planck*+WP data.

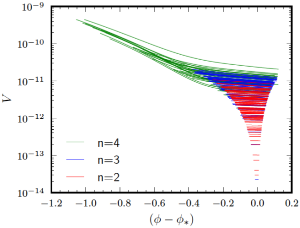
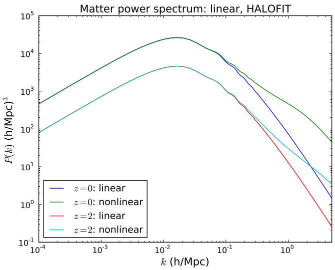


Fig. 14. Observable range of the best-fitting inflaton potentials, when $V(\phi)$ is Taylor expanded at the nth order around the pivot value ϕ_*, in natural units (where $\sqrt{8\pi}M_{\rm pl} = 1$), assuming a flat prior on ϵ_V, η_V, ξ^2_V, and ϖ^3_V, and using *Planck*+WP data.

Fourier module

E. Power spectra in Fourier space

- Linear matter power spectrum $P_m(k,z) \rightarrow$ integrated quantities $\sigma(R,z)$, $\sigma_8(z)$
- Linear baryon+CDM power spectrum $P_{cb}(k,z) \rightarrow$ integrated quantities $\sigma_{cb,8}(z)$
- Approximation for non-linear spectrum $P_m^{NL}(k,z)$ based on prescriptions like HALOFIT, HMCODE...
- Keep in memory non-linear correction factors like $R^{NL}(k,z) = \left(P_m^{NL}(k,z)/P_m(k,z)\right)^{1/2}$ for e.g. CMB lensing, cosmic shear, number count C_ℓ's



Transfer module

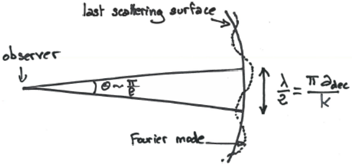
F. Transfer functions in harmonic space

CMB spectrum depends on $\Delta_\ell^X(k) = \ell$-th multipole of anisotropy of photon temperature and polarisation ($X \in \{T, E, B\}$) for each mode (scalar/tensor) and initial condition (adiabatic/isocurvature) today ($\tau = \tau_0$).

- In COSMICS: integrate equations for each k, ℓ, X, mode, I.C. until today.
- Since CMBFAST (Seljak & Zaldarriaga 1996): use "line-of-sight integral", more precisely and exact implicit solution of Boltzmann equation (here in flat space):

$$\Delta_\ell^X(k) = \int_\epsilon^{\tau_0} d\tau\ S^X(\tau, k)\ j_l(k(\tau_0 - \tau))$$

$S(\tau, k)$ only depends on thermodynamical functions, first few multipoles, baryons flux divergence and metric perturbations. Role of Bessel: projection from Fourier to harmonic space ($\theta\, d_a(z_{\rm rec}) = \frac{\lambda}{2}$ gives precisely $l = k(\tau_0 - \tau_{\rm rec})$):



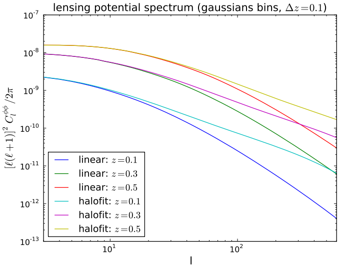
Curved space: spherical bessel functions → modified Bessel functions (hypergeometric)

F. Transfer functions in harmonic space

$$\Delta_\ell^X(k) = \int_\epsilon^{\tau_0} d\tau \; S^X(\tau, k) \; j_l(k(\tau_0 - \tau))$$

Applies not just to CMB $X \in \{T, E, B\}$ but also all LSS C_ℓ's (one X per type of observable and redshift bin).

- CMB lensing + cosmic shear: similar formulation, $S(\tau, k)$ depends on metric fluctuation and window function (intrinsic to lensing + source selection function)
- number count (galaxy clustering): $S(\tau, k)$ depends on baryon+CDM density fluctuation and selection function in each bin plus corrections from matter flux divergence and metric perturbations (RSD, Doppler, lensing, other GR effects)
- may include non-linear correction factors $R^{NL}(k, z)$



F. Transfer functions in harmonic space: compact source functions

Well known

$$\Delta_\ell(k) = \int_\epsilon^{\tau_0} d\tau \; S_T(\tau,k) \; j_\ell(k(\tau_0-\tau))$$

$$\text{with} \quad S_T(\tau,k) \equiv \underbrace{g\,(\Theta_0+\psi)}_{\text{SW}} + \underbrace{\left(g\,k^{-2}\theta_{\rm b}\right)'}_{\text{Doppler}} + \underbrace{e^{-\kappa}(\phi'+\psi')}_{\text{ISW}} + \text{polarisation}$$

comes from integration by part of:

$$\begin{aligned}\Delta_l(k) = \int_{\tau_{\rm ini}}^{\tau_0} d\tau \; \Big\{ & S_T^0(\tau,k) \; j_l(k(\tau_0-\tau)) \\ & + S_T^1(\tau,k) \; \frac{dj_l}{dx}(k(\tau_0-\tau)) \\ & + S_T^2(\tau,k) \; \frac{1}{2}\left[3\frac{d^2 j_l}{dx^2}(k(\tau_0-\tau)) + j_l(k(\tau_0-\tau))\right]\Big\}\end{aligned}$$

But $(S_T^1)'$, $(S_T^2)'$, $(S_T^2)''$ problematic! (Derivative of Einstein equation, massive neutrinos $\rightarrow$ finite differences...)

F. Transfer functions in harmonic space: compact source functions

Example of temperature source function in CAMB:

```
!Maple fortran output - see scal_eqs.map
        ISW = (4.D0/3.D0*k*EV%Kf(1)*sigma+(-2.D0/3.D0*sigma
            -2.D0/3.D0*etak/adotoa)*k &
              -diff_rhopi/k**2-1.D0/adotoa*dgrho/3.D0+(3.D0*
                  gpres+5.D0*grho)*sigma/k/3.D0 &
              -2.D0/k*adotoa/EV%Kf(1)*etak)*expmmu(j)
!The rest, note y(9)->octg, yprime(9)->octgprime (octopoles)
    sources(1)= ISW +  ((-9.D0/160.D0*pig-27.D0/80.D0*ypol
        (2))/k**2*opac(j)+(11.D0/10.D0*sigma- &
    3.D0/8.D0*EV%Kf(2)*ypol(3)+vb-9.D0/80.D0*EV%Kf(2)*octg
        +3.D0/40.D0*qg)/k-(- &
    180.D0*ypolprime(2)-30.D0*pigdot)/k**2/160.D0)*dvis(j)
        +(-(9.D0*pigdot+ &
    54.D0*ypolprime(2))/k**2*opac(j)/160.D0+pig/16.D0+clxg
        /4.D0+3.D0/8.D0*ypol(2)+(- &
    21.D0/5.D0*adotoa*sigma-3.D0/8.D0*EV%Kf(2)*ypolprime(3)+
        vbdot+3.D0/40.D0*qgdot- &
    9.D0/80.D0*EV%Kf(2)*octgprime)/k+(-9.D0/160.D0*dopac(j)*
        pig-21.D0/10.D0*dgpi-27.D0/ &
    80.D0*dopac(j)*ypol(2))/k**2)*vis(j)+(3.D0/16.D0*ddvis(j
        )*pig+9.D0/ &
    8.D0*ddvis(j)*ypol(2))/k**2+21.D0/10.D0/k/EV%Kf(1)*vis(j
```

F. Transfer functions in harmonic space: compact source functions

So we should rather stick to

$$\Delta_l(k) = \int_{\tau_{\rm ini}}^{\tau_0} d\tau \, \Big\{ S_T^0(\tau,k) \; j_l(k(\tau_0-\tau)) \\ + S_T^1(\tau,k) \; \frac{dj_l}{dx}(k(\tau_0-\tau)) \\ + S_T^2(\tau,k) \; \frac{1}{2}\left[3\frac{d^2 j_l}{dx^2}(k(\tau_0-\tau)) + j_l(k(\tau_0-\tau))\right] \Big\}$$

CLASS v2.0 stores separately $S_T^0(\tau,k)$, $S_T^1(\tau,k)$, $S_T^2(\tau,k)$, and the transfer module will convolve them individually with respective bessel functions.

$$S_T^0 = g\left(\frac{\delta_g}{4}+\psi\right) + e^{-\kappa}(\phi'+\psi') \qquad S_T^1 = g\frac{\theta_b}{k} \qquad S_T^2 = \frac{g}{8}\left(G_0+G_2+F_2\right)$$

or

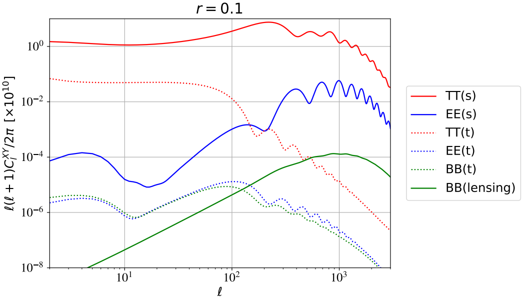
$$S_T^0 = g\left(\frac{\delta_g}{4}+\phi\right) + e^{-\kappa}2\phi' + g'\theta_b + g\theta_b' \qquad S_T^1 = e^{-\kappa}k(\psi-\phi) \qquad S_T^2 = \frac{g}{8}\left(G_0+G_2+F_2\right)$$

G. Harmonic power spectra (C_ℓ's)

Trivial:

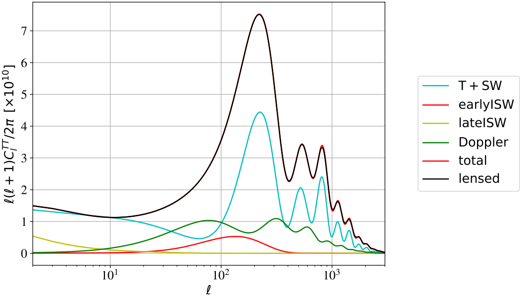
$$C_\ell^{XY} = \int \frac{dk}{k} \sum_{ij} \Delta_{\ell\,i}^X(k) \Delta_{\ell\,j}^Y(k) \mathcal{P}_{ij}(k)$$

with sum running over modes (scalar/tensor) and I.C. (adiabatic/isocurvature).



H. Lensed CMB C_ℓ's

- metric fluctuations (ϕ, ψ) $\rightarrow$ lensing potential source function $\rightarrow$ CMB lensing potential spectrum C_ℓ^{PP}
- several fluctuations $\rightarrow$ CMB source functions $\rightarrow$ unlensed CMB spectra $C_\ell^{TT,TE,EE,BB}$
- several quadratic sums over $C_{\ell_1}^{XY} C_{\ell_2}^{PP}$ $\rightarrow$ lensed CMB spectra $C_\ell^{TT,TE,EE,BB}$. Full-sky approach of Challinor & Lewis 2005.



Distorsion module

I. Spectral distorsions of CMB blackbody

Explained in *The synergy between CMB spectral distortions and anisotropies*, Lucca, Schöneberg, Hooper, Lesgourgues & Chluba, JCAP 2020 [arxiv:1910.04619].

Heating rates from `external/heating/injection.c` and `external/heating/noninjection.c` get processed with Jens Chluba's Green functions from `external/Greens_data.dat`, to get different components of spectral distorsions (μ, y, PCA of residuals).

Additional machinery to express results in the form potentially observable by FIRAS or PIXIE.

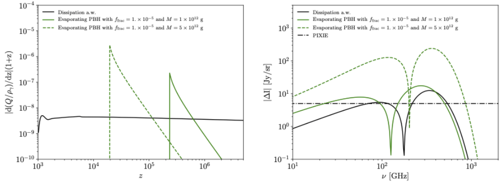


Figure 5. Heating rate (left panel) and SDs (right panel) caused by PBH evaporation (green line) The heating rate caused by the dissipation of acoustic waves (black line) is given as a reference. Once more, the dot-dashed line in the right panel represents the predicted PIXIE sensitivity.