

CLASS

Cosmological Linear Anisotropy Solving System



Markus Mosbech

Institute for Theoretical Particle Physics and Cosmology, RWTH Aachen
University

Les Karellis, France, 17-30 Aug 2025

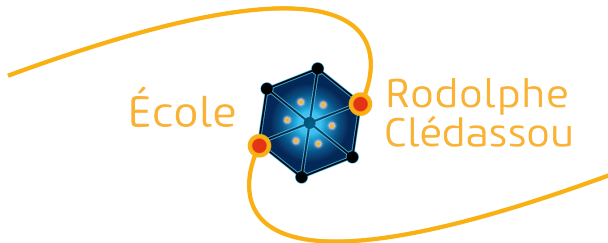
These slides available at https://github.com/MarkMos/class_lecture
Visit <http://class-code.net/> for more info!

class in Les Karellis

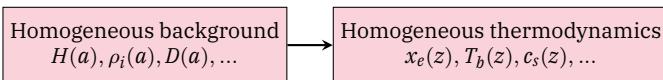
What to expect in these lectures:

- Basics: Why use **class**?
- Usage: Installation
- Usage: Python Interface
- Basics: Existing Species
- Basics: Module Overview
- Theory: What is **class** based upon?
- Coding: Implementing features

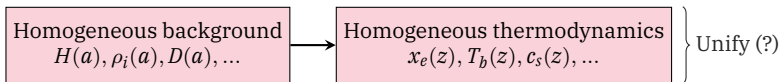
We will learn **how to use class** and **which models** can be run with it.



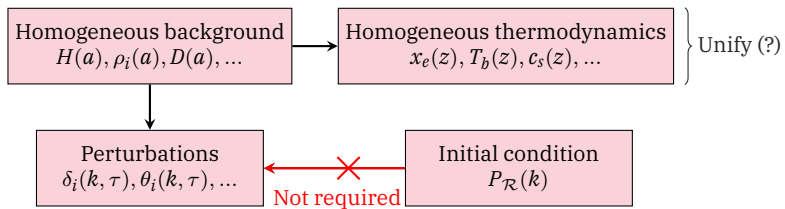
Fundamental layout of Einstein-Boltzmann solvers



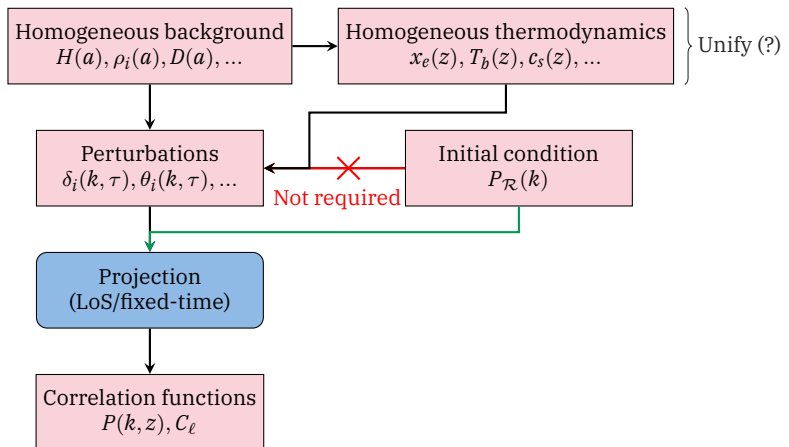
Fundamental layout of Einstein-Boltzmann solvers



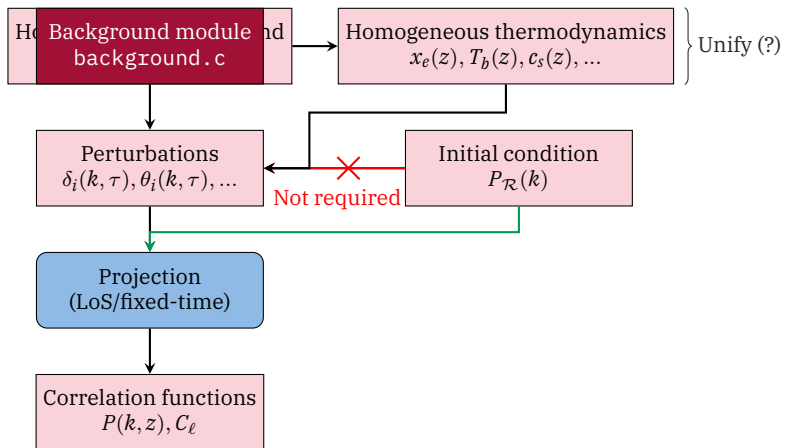
Fundamental layout of Einstein-Boltzmann solvers



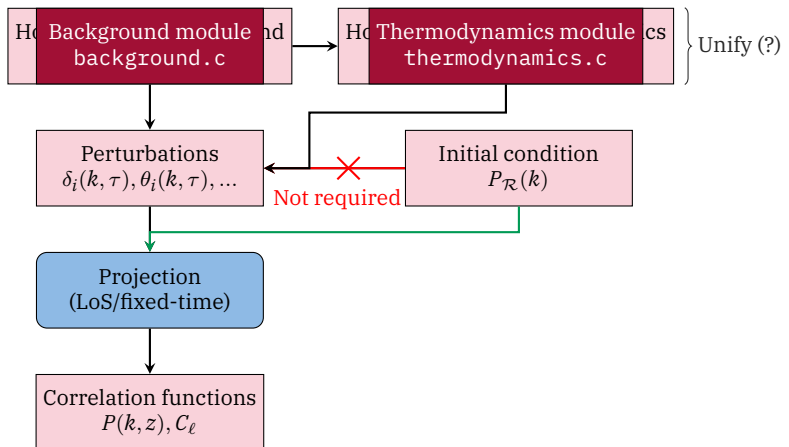
Fundamental layout of Einstein-Boltzmann solvers



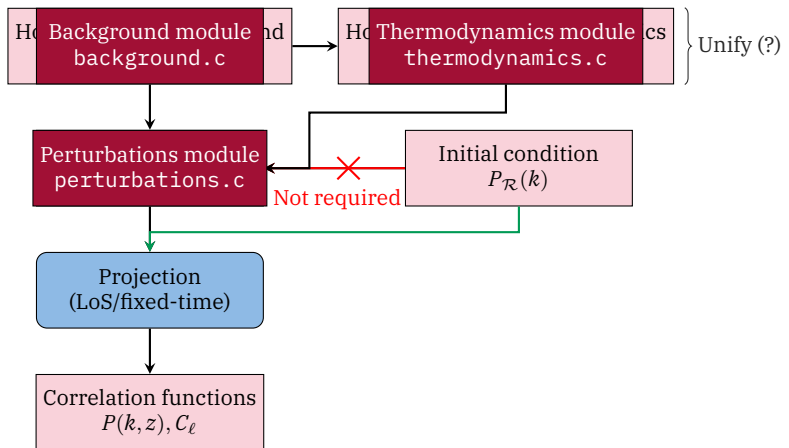
Fundamental layout of Einstein-Boltzmann solvers



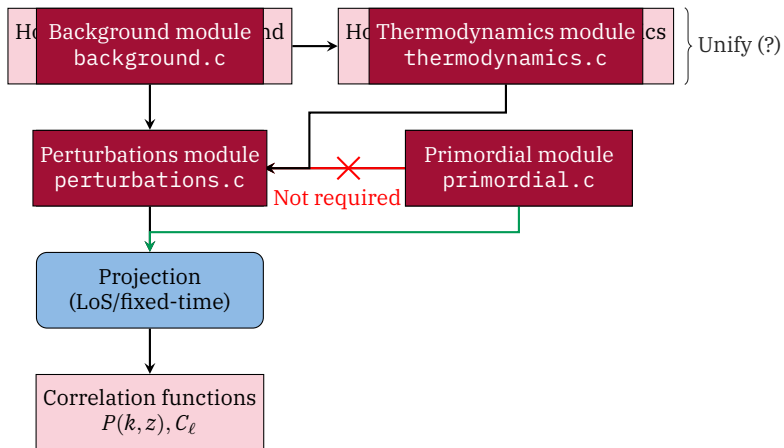
Fundamental layout of Einstein-Boltzmann solvers



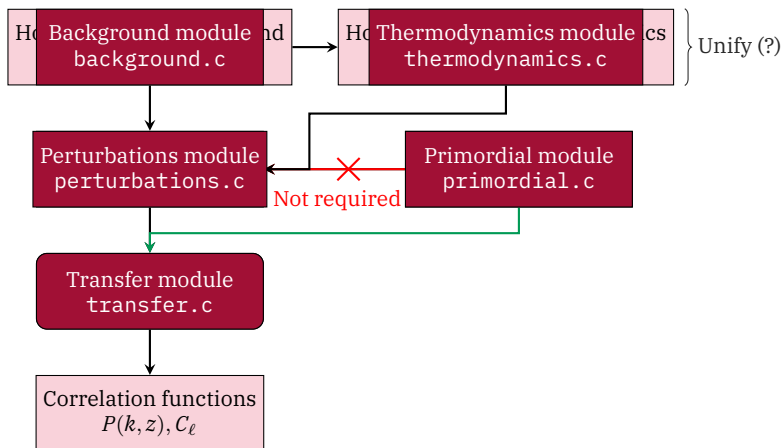
Fundamental layout of Einstein-Boltzmann solvers



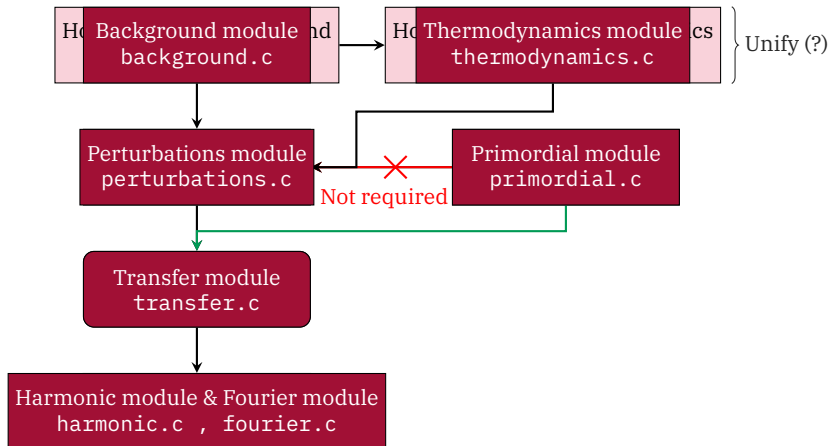
Fundamental layout of Einstein-Boltzmann solvers



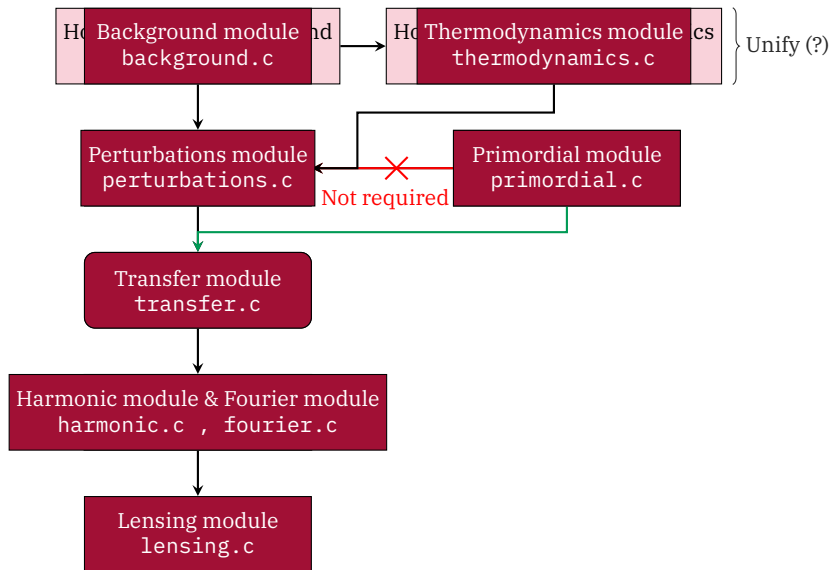
Fundamental layout of Einstein-Boltzmann solvers



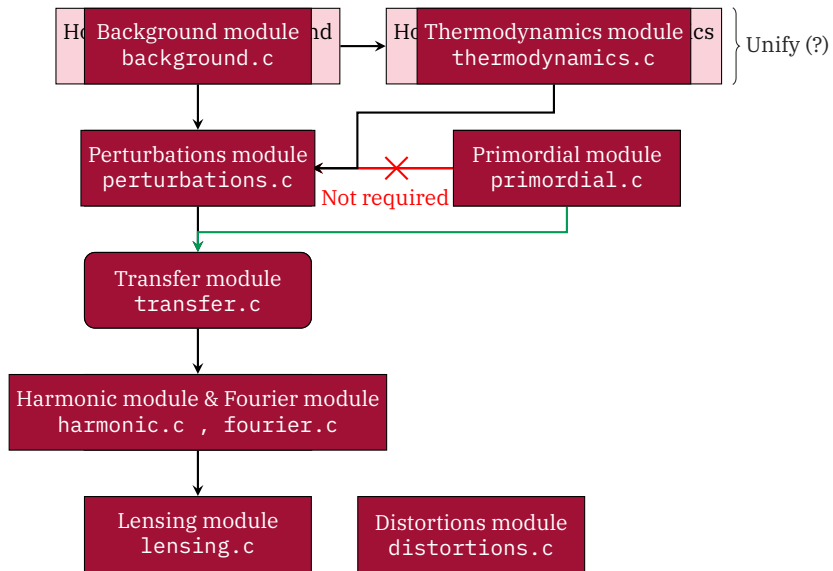
Fundamental layout of Einstein-Boltzmann solvers



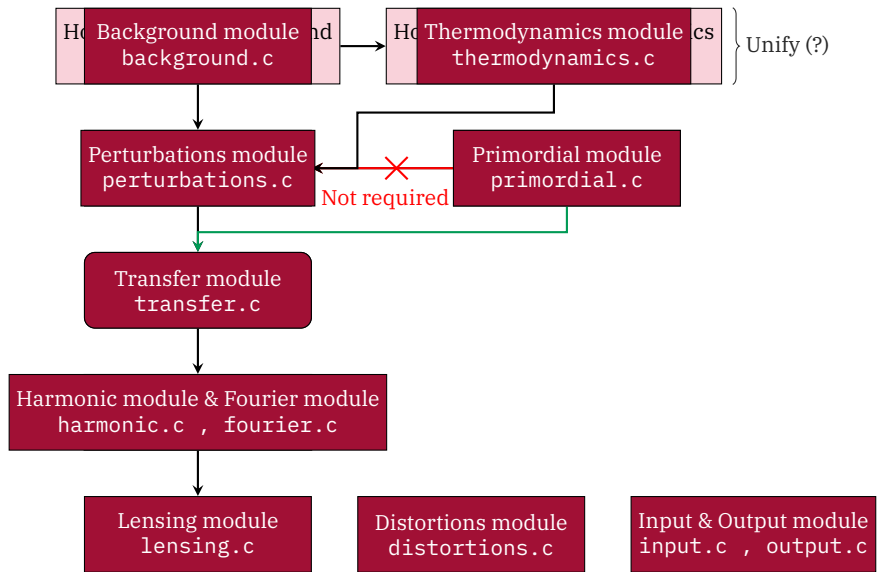
Fundamental layout of Einstein-Boltzmann solvers



Fundamental layout of Einstein-Boltzmann solvers



Fundamental layout of Einstein-Boltzmann solvers

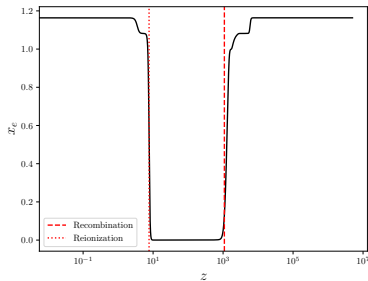


Let's make a journey through each module!

Essentials 3: Thermodynamics

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

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



Then $x_e(z) \rightarrow \kappa'(z)$ (Thomson scattering rate)

$\rightarrow \kappa(z)$ (Optical depth)

$\rightarrow \exp(-\kappa(z))$ (factor for Integrated Sachs-Wolfe effect)

$\rightarrow g(z)$ (visibility function for Sachs-Wolfe effect)

$\rightarrow g'(z)$ (factor for Doppler effect)

The Thermodynamics Module

Simplest model of **recombination** is the **Saha equation**.

It is well known that a non-relativistic ($T \ll m$) species in thermal equilibrium obeys

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi} \right)^{3/2} e^{-m/T} \quad (1)$$

Thus we find using **complete thermal equilibrium** with $\mu_{\text{ionized}} + \mu_e = \mu_{\text{rec}}$ that

$$\frac{n_e n_{\text{ionized}}}{n_{\text{rec}}} \approx \left(\frac{m_e T}{2\pi} \right)^{3/2} e^{-E_{\text{bind}}/T} \times \underbrace{\left[e^{\mu_{\text{ionized}} + \mu_e - \mu_{\text{rec}}} \left(\frac{g_e g_{\text{ionized}}}{g_{\text{rec}}} \right) \left(\frac{m_{\text{ionized}}}{m_{\text{rec}}} \right)^{3/2} \right]}_{\approx 1}$$

This gives

$$\frac{x_e^2}{1 - x_e} \approx \left(\frac{1.1 \cdot 10^{-10}}{n_{\text{H},0}/T_{\text{cmb},0}^3} \right) \left(\frac{\text{eV}}{T} \right)^{3/2} \exp(39.9 - 13.6 \frac{\text{eV}}{T}) \quad (2)$$

and thus recombination at $T \approx \frac{13.6\text{eV}}{39.9} \approx 0.34\text{eV} \rightarrow z \approx 1400$.

The Thermodynamics Module

Simplest model of **recombination** is the **Saha equation**.

It is well known that a non-relativistic ($T \ll m$) species in thermal equilibrium obeys

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi} \right)^{3/2} e^{-m/T} \quad (1)$$

Thus we find using **complete thermal equilibrium** with $\mu_{\text{ionized}} + \mu_e = \mu_{\text{rec}}$ that

$$\frac{n_e n_{\text{ionized}}}{n_{\text{rec}}} \approx \left(\frac{m_e T}{2\pi} \right)^{3/2} e^{-E_{\text{bind}}/T} \times \underbrace{\left[e^{\mu_{\text{ionized}} + \mu_e - \mu_{\text{rec}}} \left(\frac{g_e g_{\text{ionized}}}{g_{\text{rec}}} \right) \left(\frac{m_{\text{ionized}}}{m_{\text{rec}}} \right)^{3/2} \right]}_{\approx 1}$$

This gives

$$\frac{x_e^2}{1 - x_e} \approx \left(\frac{1.1 \cdot 10^{-10}}{n_{\text{H},0}/T_{\text{cmb},0}^3} \right) \left(\frac{\text{eV}}{T} \right)^{3/2} \exp(39.9 - 13.6 \frac{\text{eV}}{T}) \quad (2)$$

and thus recombination at $T \approx \frac{13.6\text{eV}}{39.9} \approx 0.34\text{eV} \rightarrow z \approx 1400$. This is of course wrong...

The Thermodynamics Module

Simplest model of **recombination** is the **Saha equation**.

It is well known that a non-relativistic ($T \ll m$) species in thermal equilibrium obeys

$$n(\mu, T) \approx g e^{\mu/T} \left(\frac{mT}{2\pi} \right)^{3/2} e^{-m/T} \quad (1)$$

Thus we find using **complete thermal equilibrium** with $\mu_{\text{ionized}} + \mu_e = \mu_{\text{rec}}$ that

$$\frac{n_e n_{\text{ionized}}}{n_{\text{rec}}} \approx \left(\frac{m_e T}{2\pi} \right)^{3/2} e^{-E_{\text{bind}}/T} \times \underbrace{\left[e^{\mu_{\text{ionized}} + \mu_e - \mu_{\text{rec}}} \left(\frac{g_e g_{\text{ionized}}}{g_{\text{rec}}} \right) \left(\frac{m_{\text{ionized}}}{m_{\text{rec}}} \right)^{3/2} \right]}_{\approx 1}$$

This gives

$$\frac{x_e^2}{1 - x_e} \approx \left(\frac{1.1 \cdot 10^{-10}}{n_{\text{H},0}/T_{\text{cmb},0}^3} \right) \left(\frac{\text{eV}}{T} \right)^{3/2} \exp(39.9 - 13.6 \frac{\text{eV}}{T}) \quad (2)$$

and thus recombination at $T \approx \frac{13.6\text{eV}}{39.9} \approx 0.34\text{eV} \rightarrow z \approx 1400$. This is of course wrong...

recombination is a non-equilibrium process

The Thermodynamics Module

The **effective multi-level atom** is the basis for recombination codes.

1s 2s 2p 3s 3p 3d ... ionized \rightarrow 1s 2s 2p ionized

The Thermodynamics Module

The **effective multi-level atom** is the basis for recombination codes.

$1s\ 2s\ 2p\ 3s\ 3p\ 3d\ \dots$ ionized $\rightarrow 1s\ 2s\ 2p$ ionized

Reason: Intermediate transitions ($4p \rightarrow 3s$) or ($3s \rightarrow 2p$) are comparatively instant. Why? Direct transition $2s \rightarrow 1s$ is forbidden, and $2p \rightarrow 1s$ is immediately reversed by $1s \rightarrow 2p$. The medium is **optically thick** during recombination.

The Thermodynamics Module

The **effective multi-level atom** is the basis for recombination codes.

1s 2s 2p 3s 3p 3d ... ionized \rightarrow 1s 2s 2p ionized

Reason: Intermediate transitions (4p \rightarrow 3s) or (3s \rightarrow 2p) are comparatively instant. Why? Direct transition 2s \rightarrow 1s is forbidden, and 2p \rightarrow 1s is immediately reversed by 1s \rightarrow 2p. The medium is **optically thick** during recombination.

Instead, focus on 2p \rightarrow 1s with subsequent redshifting of photon to escape reabsorption (slow) or 2s \rightarrow 1s with two-photon decay (slow).

Peeble's equation

$$\dot{x}_e \approx f_{\text{photo-ion}}(T)x_{\text{rec}} - f_{\text{rec}}(T)x_e x_{\text{ionized}} \quad (3)$$

Solved numerically, basis of **recfast**

The Thermodynamics Module

recfast only resolves $2s \approx 2p$

The Thermodynamics Module

recfast only resolves $2s \approx 2p$

Improvement: **HyRec** with EMLA resolves $2s, 2p$. Even more, can do $2s, 2p, 3s, \dots$ with *effective* rates.

Fullest code to date: **CosmoRec** does full numerical computation (iteratively).
Comparatively slow, but highest achievable accuracy

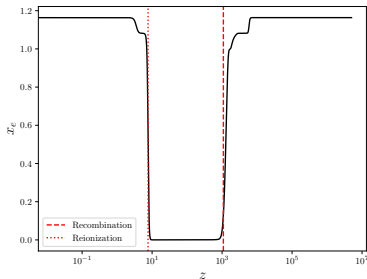
The Thermodynamics Module

recfast only resolves $2s \approx 2p$

Improvement: **HyRec** with EMLA resolves $2s, 2p$. Even more, can do $2s, 2p, 3s, \dots$ with *effective* rates.

Fulllest code to date: **CosmoRec** does full numerical computation (iteratively).
Comparatively slow, but highest achievable accuracy

Further complication: Helium (higher elements don't contribute)



The Thermodynamics Module

User can choose to model approximate recombination and get $x_e(z)$, $T_b(z)$ from:

- **RECFAST** (Wong, Moss & Scott 2008)
- **HyRec-2** (Y. Ali-Haïmoud, N. Lee)
- Possibly soon? **CosmoRec** (J. Chluba)

The Thermodynamics Module

User can choose to model approximate recombination and get $x_e(z)$, $T_b(z)$ from:

- **RECFAST** (Wong, Moss & Scott 2008)
- **HyRec-2** (Y. Ali-Haïmoud, N. Lee)
- Possibly soon? **CosmoRec** (J. Chluba)

Recombination needs one more cosmological parameter: the **primordial Helium fraction** Y_{He} .

- Fix it ($Y_{\text{He}} = 0.25$)
- Get it from BBN ($Y_{\text{He}} = \text{BBN}$). **class** has interpolation table pre-pcomputed with a **BBN code** (**Parthenope**), for each given value of N_{eff} , ω_b (assumes $\mu_{\nu_e} = 0$, easy to generalize).
- BBN interpolation table located in separate directory (in `external/bbn/sBBN_2017.dat`, update inbound)

The Thermodynamics Module

For reionization:

- tanh with complicated argument (like **CAMB**)
- multi-tanh
- half tanh
- from file (either linear or tanh)

The Thermodynamics Module

For reionization:

- tanh with complicated argument (like **CAMB**)
- multi-tanh
- half tanh
- from file (either linear or tanh)

Mini-shooting to find z_{reio} for given $\tau_{\text{reio}} = \kappa_{\text{reio}}$.

Optical depth $\kappa(z)$ = inverse number of expected interactions $\Rightarrow \kappa'(z) = an_H x_e \sigma_T$

The Thermodynamics Module

We also include

- Energy injection (increases ionization, heats T_b)
This can cause changes in scattering $\kappa(z)$ and thus be observable with CMB
- Time-dependent fundamental constants \rightarrow Causes shift in recombination due to fundamental dependencies such as
$$E_{\text{binding}} = \frac{1}{2}\alpha^2 m_e = 13.6\text{eV} (137\alpha)^2 \left(\frac{m_e}{511\text{keV}}\right)$$
We remind ourselves $1 + z_{\text{rec}} = T_{\text{rec}}/T_{\text{cmb}} \approx \frac{E_{\text{binding}}}{12.57\text{meV}}$
- Computation of useful quantities $z_{\text{rec}}, z_{\text{drag}}, z_*, D_A(z_{\text{rec}}), r_s(z_{\text{drag}}), \dots$

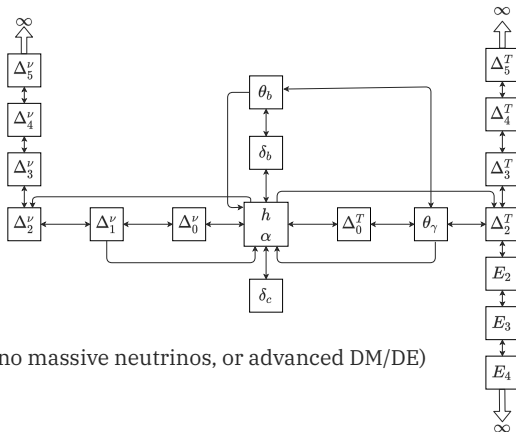
Thermodynamics exercise

Let's check this with an exercise!

Download the jupyter notebook [Exercise_thermodynamics_to_fill.ipynb](#) and follow the steps to plot the properties computed by the Thermodynamics module.

Essentials 4: Perturbations

- Find all perturbations ($\delta_X(\tau, k)$, $\phi(\tau, k)$, ...) by integrating ODEs for each independent wavenumber k , each mode (scalar/tensor), each initial condition (adiabatic/isocurvature)



Minimal set! (no massive neutrinos, or advanced DM/DE)

The Perturbations Module

- Find all perturbations ($\delta_X(\tau, k)$, $\phi(\tau, k)$, ...) by integrating ODEs for each independent wavenumber k , each mode (scalar/tensor), each initial condition (adiabatic/isocurvature):
 - Boltzmann
 - Continuity + Euler
 - linearized Einstein equations (one = ODE, others = constraint equations)

Linear perturbations \Rightarrow perturbations normalized to initial condition
(**class** \rightarrow curvature $\mathcal{R} = 1$ for scalar with adiabatic I.C.)

The Perturbations Module

Einstein Equations

$$k^2\phi + 3\mathcal{H}(\phi' + \mathcal{H}\psi) = -4\pi Ga^2\delta\rho \quad (4)$$

$$k^2(\phi' + \mathcal{H}\psi) = 4\pi Ga^2(\rho + P)\theta\phi'' + \mathcal{H} \quad (5)$$

$$(\psi' + \phi') + (2\mathcal{H}' + \mathcal{H}^2)\psi + \frac{1}{3}k^2(\phi - \psi) = 4\pi Ga^2\delta P \quad (6)$$

$$k^2(\phi - \psi) = 12\pi Ga^2(\rho + P)\sigma \quad (7)$$

and Boltzmann equations

$$\frac{dF_0^{(\gamma)}}{d\eta} + kF_1^{(\gamma)} = 4\phi' \quad (8)$$

$$\frac{dF_1^{(\gamma)}}{d\eta} - \frac{k}{3} [F_0^{(\gamma)} - 2F_2^{(\gamma)}] = \frac{4k}{3}\psi + \underbrace{\Gamma_{\gamma,b}}_{\text{from thermodynamics}} [F_1^{(b)} - F_1^{(\gamma)}] \quad (9)$$

$$\frac{dF_2^{(\gamma)}}{d\eta} - \frac{k}{5} [2F_1^{(\gamma)} - 3F_3^{(\gamma)}] = \underbrace{\Gamma_{\gamma,b}}_{\text{from thermodynamics}} [-F_2^{(\gamma)} + \Pi^{\text{pol}}/10] \quad (10)$$

$$\frac{dF_\ell^{(\gamma)}}{d\eta} - \frac{k}{2\ell+1} [\ell F_{\ell-1}^{(\gamma)} - (\ell+1)F_{\ell+1}^{(\gamma)}] = 0 \quad (\text{infinite hierarchy}) \quad (11)$$

The Perturbations Module

4 Einstein Equations (only one dynamical)

1 $\ell_{\max}^{(\gamma)}$ photon temperature hierarchy

1 $\ell_{\max}^{(\gamma)}$ photon polarization hierarchy (or 2 $\ell_{\max}^{(\gamma)}$)

2 baryon (density, velocity)

1/2 cdm equations (density?, velocity)

Either

a) 1 $\ell_{\max}^{(\text{dr})}$ massless neutrino hierarchy

b) 1 $\ell_{\max}^{(\text{dr})}$ massless neutrino hierarchy

+ $N_{\text{ncdm}} \cdot \ell_{\max}^{(\text{ncdm})} \cdot N_q$ massive neutrino hierarchies

= Too many equations for simple solvers!

(also tight coupling == stiff equations)

(also sparse system)

The Perturbations Module

ODE Solver (customized for Einstein-Boltzmann equations)

- Stiff system require implicit method like backward Euler or more advanced:
→ find \mathbf{y}_{n+1} as a solution of $\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{y}'(\mathbf{y}_{n+1})\delta t$
- Still fast: Newton method with Jacobian recycling
- Robustness requires δt to be adaptive time step
- Source function required at predefined t_i : on-the-fly interpolation
- System is sparse: some algebra gives big speed up (sparse LU decomposition)

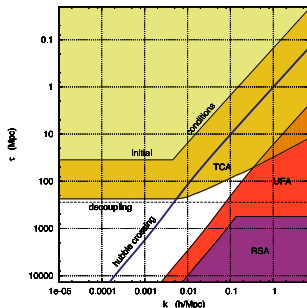
Everything gathered in ndf15 by T. Tram (CLASS II 2011).
TCA could even be removed!

The Perturbations Module

ODE approximations (papers : CLASS II & CLASS IV 2011)

Idea of these approximations: Reduce number of evolved equations.

- Tight Coupling Approximation for baryons and γ at 2nd order \rightarrow Suppresses shear & higher moments whenever $k\tau_{b\gamma} \ll 1$
- Ultrarelativistic Fluid Approximation (for massless ν , also one for massive ones): truncated Boltzmann, 3 equations \rightarrow Suppresses higher order moments whenever $k\tau \gg 1$
- Radiation Streaming Approximation (for photons and massless ν): test particles, 0 equations \rightarrow Only follow oscillation-averaged evolution when $k\tau \gg \ell$ (+ for photons $k\tau_{b\gamma} \gg 1$)

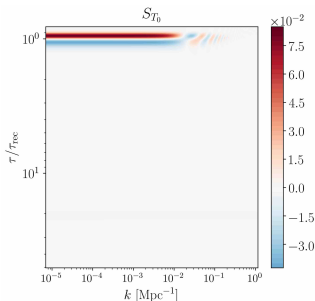


The Perturbations Module

Source functions

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

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



The Perturbations Module

Photon hierarchies

Two approaches to polarization in Boltzmann hierarchy:

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

The Perturbations Module

Photon hierarchies

Two approaches to polarization in Boltzmann hierarchy:

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

CMBFAST: first in flat space, second in curved space

The Perturbations Module

Photon hierarchies

Two approaches to polarization in Boltzmann hierarchy:

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

CMBFAST: first in flat space, second in curved space

CAMB: always second case

The Perturbations Module

Photon hierarchies

Two approaches to polarization in Boltzmann hierarchy:

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

CMBFAST: first in flat space, second in curved space

CAMB: always second case

class: first case by default, thanks to new analytic results in curved space
(T. Tram & JL, JCAP 2013 [arXiv:1305.3261], approximation!)

User can select second case

Perturbations exercise

Let's check this with an exercise!

Download the jupyter notebook [Exercise_perturbations_to_fill.ipynb](#) and follow the steps to plot the properties computed by the Perturbations module.

Essentials 5: Primordial

Initial conditions for scalars (adiabatic, isocurvature) and tensors. Linear theory
 \Leftrightarrow Gaussian independent Fourier modes \Leftrightarrow only power spectrum required

- analytic: 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

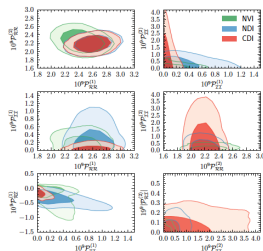


Fig. 22. Two dimensional distributions for power in isocurvature modes, using *Planck*+WP data.

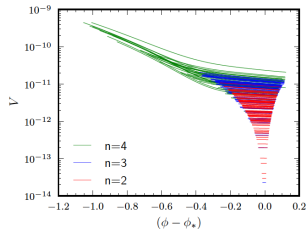


Fig. 14. Observable range of the best-fitting inflaton potentials, when $V(\phi)$ is Taylor expanded at the n th order around the pivot value ϕ_* , in natural units (where $\sqrt{8\pi}M_{\text{pl}} = 1$), assuming a flat prior on ϵ_V , η_V , ξ_V^2 , and ϖ_V^3 , and using *Planck*+WP data.

The Primordial module

Primordial spectra : modes

P_k_ini type =	modes =	ic =
analytic_Pk inflation_V inflation_H inflation_V_end	at least one: s,t s,t s,t s,t	at least one: ad,bi,cdi,nid,niv ad ad ad
external_Pk	at least one: s,t	ad

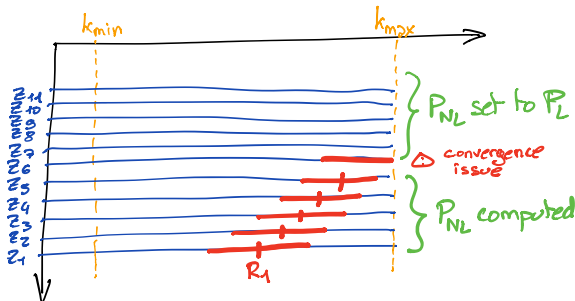
Essentials 6: Fourier

- 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 also non-linear correction factor $(R^{NL}(k, z))^2 = P_m^{NL}(k, z)/P_m(k, z)$ for e.g., CMB lensing, cosmic shear, number count C_ℓ 's

The Fourier Module

How to emulate non-linear evolution with a halo model?

Halofit or HMcode require non-linearity scale $R_{\text{NL}}(z)$ such that $\sigma(R_{\text{NL}}(z), z) = 1$.



To get $P^{\text{NL}}(k, z)$ at higher z one should increase k_{max} .

The Fourier Module

Halofit relies on simple similarity solution Ansatz:

$$\Delta_{1-\text{halo}}^2(k) = \underbrace{\frac{a_n y^{3f_2}}{(1 + b_n y^{f_2} + (f_3 c_n y)^{3-\gamma_n})}}_{\text{Original term, corrected with } f_2} \underbrace{\frac{1}{(1 + x_\mu y^{-1} + x_\nu y^{-2})(1 + 0.977 f_\nu)}}_{\text{Further corrections}}$$

with $y = k/k_{\text{nl}} = kR_{\text{NL}}(z)$.

Parameters calibrated to fit (early) simulations reasonably well.

$$\Delta_{\text{nl}}^2 \approx \Delta_{2-\text{halo}}^2 + \Delta_{1-\text{halo}}^2 \text{ with } \Delta_{2-\text{halo}}^2 \approx \frac{(1 + \Delta_{\text{lin}}^2)^\beta}{(1 + \alpha \Delta_{\text{lin}}^2)} \cdot \exp(-y/4 - y^2/8) \text{ with}$$

$$\Delta_{\text{lin}}^2 = P_{\text{lin}}(k, z) \frac{k^3}{2\pi^2}$$

Summary: Simple analytical fitting formula

The Fourier Module

HMcode has a more complicated halo model:

- $\Delta_{\text{nl}}^2 \approx \left(\left(\Delta_{2\text{-halo}}^2 \right)^\alpha + \left(\Delta_{1\text{-halo}}^2 \right)^\alpha \right)^{1/\alpha}$
- $P_{2\text{-halo}} = [P_{\text{lin}} + (f_{\text{dewiggle}} - 1.)P_{\text{BAO-wiggle}}] \times \{1 - f_{\text{damp}} \cdot (k/k_{\text{damp}})^{\alpha_{\text{damp}}} / [1 + (k/k_{\text{damp}})^{\alpha_{\text{damp}}}] \}$
- $P_{1\text{-halo}} = \int n(\nu k^\eta) g(\nu) dm$
- $\nu = \delta_c / \sigma(R(m))$
- $g(\nu) = A \cdot (1 + (q\nu^2)^{-p}) \exp(-q\nu^2/2)$ Sheth-Tormen HMF
- $n(x) = \text{NFW halo profile}$

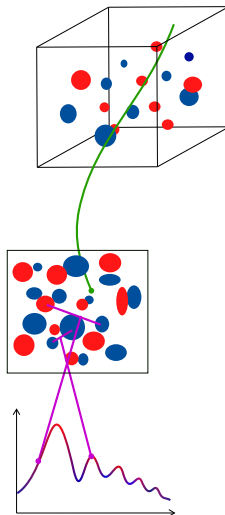
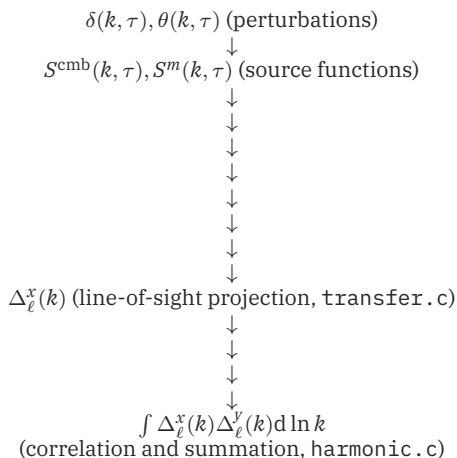
Summary: Physically motivated halo formula, reproduces well current simulations

Fourier exercise

Let's check this with an exercise!

Download the jupyter notebook [Exercise_fourier_to_fill.ipynb](#) and follow the steps to plot the properties computed by the Fourier module.

Essentials 7+8: Transfer & Harmonic



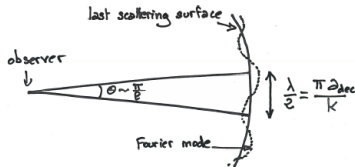
Transfer & Harmonic Modules

CMB spectrum depends on $\Delta_\ell^X(k) = \ell$ -th multipole of anisotropy of photon temperature and polarisation ($X \in \{T, E, B\}$) today ($\tau = \tau_0$).

Since **CMBFAST** (Seljak & Zaldarriaga 1996): use “line-of-sight integral”

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

$S(\tau, k)$ = source function from above. Role of Bessel: projection from Fourier to harmonic space ($\theta d_a(z_{\text{rec}}) = \frac{\lambda}{2}$ gives precisely $l = k(\tau_0 - \tau_{\text{rec}})$):



Curved space: spherical bessel functions \rightarrow modified Bessel functions (hypergeometric)

Transfer & Harmonic Modules

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

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: $S(\tau, k)$ involves broad window function)
- number count (galaxy clustering): $S(\tau, k)$ modeled fully relativistically (RSD, Doppler, lensing, other GR effects)
- may include non-linear correction factors $R^{NL}(k, z)$

Transfer & Harmonic Modules

Well known

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

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

comes from integration by part of:

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

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

Transfer & Harmonic Modules

So we should rather stick to

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

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.

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

or

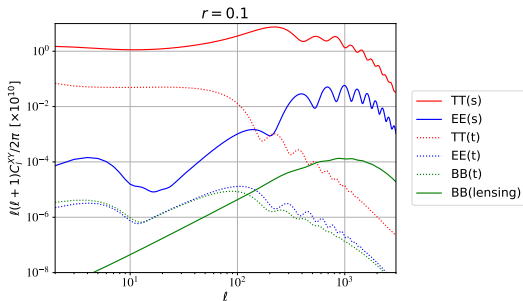
$$S_T^0 = g \left(\frac{\delta_g}{4} + \phi \right) + e^{-\kappa} 2\phi' + g'\theta_b + g\theta_b' \quad S_T^1 = e^{-\kappa} k(\psi - \phi) \quad S_T^2 = \frac{g}{8} (G_0 + G_2 + F_2)$$

Transfer & Harmonic Modules

Last step is (almost) 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).



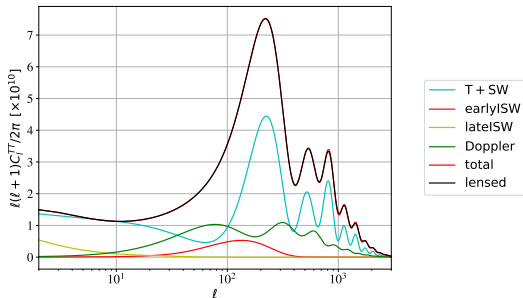
Harmonic exercise

Let's check this with an exercise!

Download the jupyter notebook [Exercise_harmonic_to_fill.ipynb](#) and follow the steps to plot the properties computed by the Harmonic module.

Essentials 9: Lensing

- metric fluctuations $(\phi, \psi) \rightarrow$ lensing potential source function \rightarrow CMB lensing potential spectrum C_ℓ^{PP}
- 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.



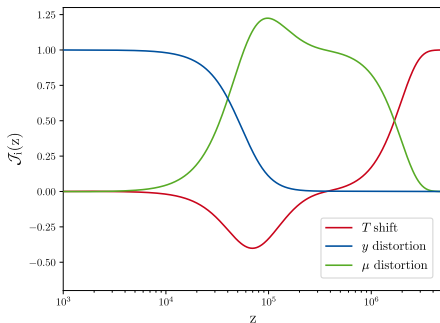
Essentials 10: Spectral Distortions

- Computations using CosmoTherm to derive thermalization Green's function
- Using Green's function to compute μ, y amplitudes

Simplified view:

$$a = \int \dot{Q} J_a(t) dt \quad (12)$$

with branching function $J_a(t)$.



Essentials 11: Output

Writes output files with correct headers and data.

If you are ever in doubt about `class` output units, check the headers of an output file.

Implementing new features

If you want to implement:

- a new species
- a new approximation scheme to simplify some equations in some regime
- a new mathematical description of an existing species (switching on more precise corrections, etc.)
- a new observable or output (new source function, new transfer function, new spectrum...)

the logic is always the same:

Implementing new features

If you want to implement:

- a new species
- a new approximation scheme to simplify some equations in some regime
- a new mathematical description of an existing species (switching on more precise corrections, etc.)
- a new observable or output (new source function, new transfer function, new spectrum...)

the logic is always the same:

- 1 define an acronym easy to search in the C files (e.g. for early dark energy: earde is good, ede is bad because it is inside “redefine”, “needed”, etc.)

Implementing new features

If you want to implement:

- a new species
- a new approximation scheme to simplify some equations in some regime
- a new mathematical description of an existing species (switching on more precise corrections, etc.)
- a new observable or output (new source function, new transfer function, new spectrum...)

the logic is always the same:

- 1 define an acronym easy to search in the C files (e.g. for early dark energy: earde is good, ede is bad because it is inside “redefine”, “needed”, etc.)
- 2 think of the feature closest to yours, and find its acronym (e.g. for fluid: fld)

Implementing new features

If you want to implement:

- a new species
- a new approximation scheme to simplify some equations in some regime
- a new mathematical description of an existing species (switching on more precise corrections, etc.)
- a new observable or output (new source function, new transfer function, new spectrum...)

the logic is always the same:

- 1 define an acronym easy to search in the C files (e.g. for early dark energy: earde is good, ede is bad because it is inside “redefine”, “needed”, etc.)
- 2 think of the feature closest to yours, and find its acronym (e.g. for fluid: fld)
- 3 grep for all occurrences of fld in include/*.h and source/*.c (normally they are all within some “**if** (has_fld){ ...}” and you can search directly for occurrences of has_fld)

Implementing new features

If you want to implement:

- a new species
- a new approximation scheme to simplify some equations in some regime
- a new mathematical description of an existing species (switching on more precise corrections, etc.)
- a new observable or output (new source function, new transfer function, new spectrum...)

the logic is always the same:

- 1 define an acronym easy to search in the C files (e.g. for early dark energy: earde is good, ede is bad because it is inside “redefine”, “needed”, etc.)
- 2 think of the feature closest to yours, and find its acronym (e.g. for fluid: fld)
- 3 grep for all occurrences of fld in include/*.h and source/*.c (normally they are all within some “**if** (has_fld){ ...}” and you can search directly for occurrences of has_fld)
- 4 duplicate these occurrences

Implementing new features

If you want to implement:

- a new species
- a new approximation scheme to simplify some equations in some regime
- a new mathematical description of an existing species (switching on more precise corrections, etc.)
- a new observable or output (new source function, new transfer function, new spectrum...)

the logic is always the same:

- 1 define an acronym easy to search in the C files (e.g. for early dark energy: earde is good, ede is bad because it is inside “redefine”, “needed”, etc.)
- 2 think of the feature closest to yours, and find its acronym (e.g. for fluid: fld)
- 3 grep for all occurrences of `fld` in `include/*.h` and `source/*.c` (normally they are all within some “`if (has_fld){ ... }`” and you can search directly for occurrences of `has_fld`)
- 4 duplicate these occurrences
- 5 change `fld` into `earde`

Implementing new features

If you want to implement:

- a new species
- a new approximation scheme to simplify some equations in some regime
- a new mathematical description of an existing species (switching on more precise corrections, etc.)
- a new observable or output (new source function, new transfer function, new spectrum...)

the logic is always the same:

- 1 define an acronym easy to search in the C files (e.g. for early dark energy: earde is good, ede is bad because it is inside “redefine”, “needed”, etc.)
- 2 think of the feature closest to yours, and find its acronym (e.g. for fluid: fld)
- 3 grep for all occurrences of `fld` in `include/*.h` and `source/*.c` (normally they are all within some “`if (has_fld){ ... }`” and you can search directly for occurrences of `has_fld`)
- 4 duplicate these occurrences
- 5 change `fld` into `earde`
- 6 change some equations to describe the specific properties of your feature