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.

École Rodolphe Clédassou

Fundamental layout of Einstein-Boltzmann solvers

Homogeneous background
$H(a), \rho_i(a), D(a), \ldots$

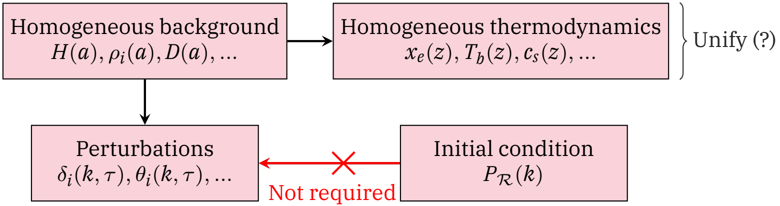
$\rightarrow$

Homogeneous thermodynamics
$x_e(z), T_b(z), c_s(z), \ldots$

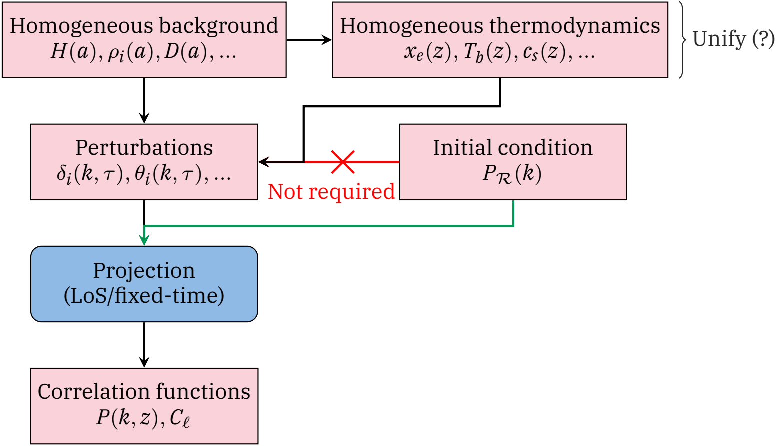
Fundamental layout of Einstein-Boltzmann solvers

$$\left.\boxed{\begin{array}{c}\text{Homogeneous background}\\ H(a),\rho_i(a),D(a),\ldots\end{array}} \longrightarrow \boxed{\begin{array}{c}\text{Homogeneous thermodynamics}\\ x_e(z),T_b(z),c_s(z),\ldots\end{array}}\right\}\text{Unify (?)}$$

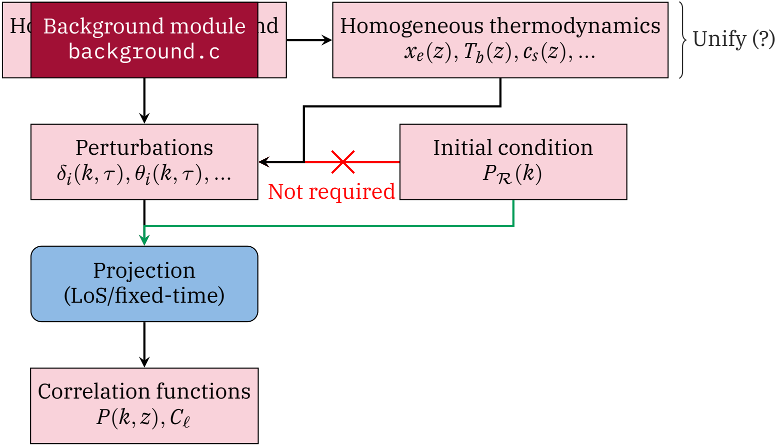
Fundamental layout of Einstein-Boltzmann solvers



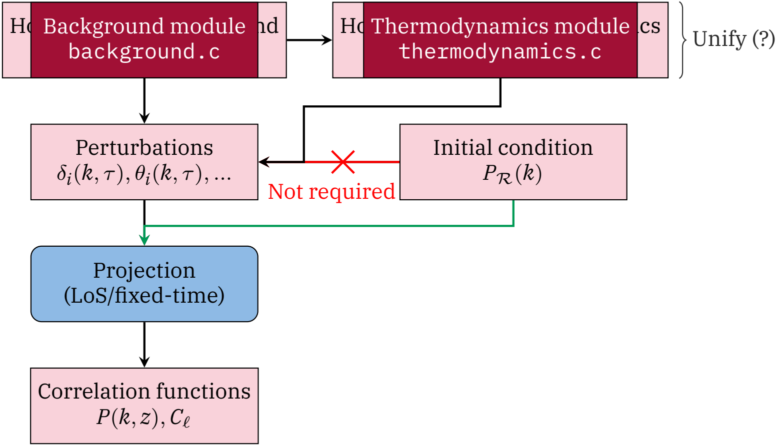
Fundamental layout of Einstein-Boltzmann solvers



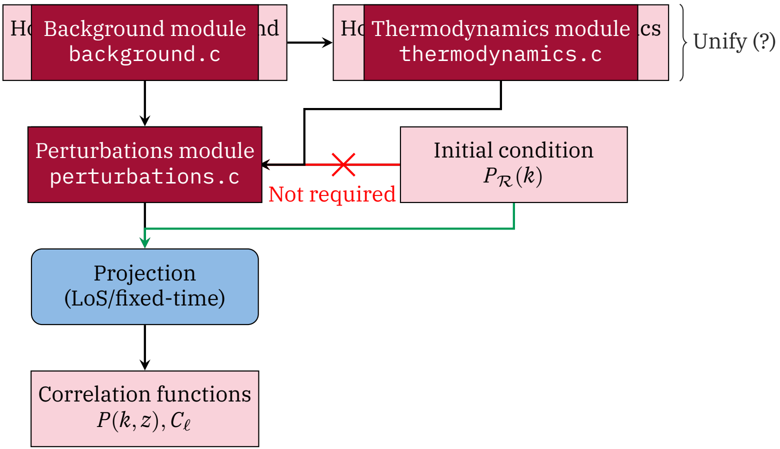
Fundamental layout of Einstein-Boltzmann solvers



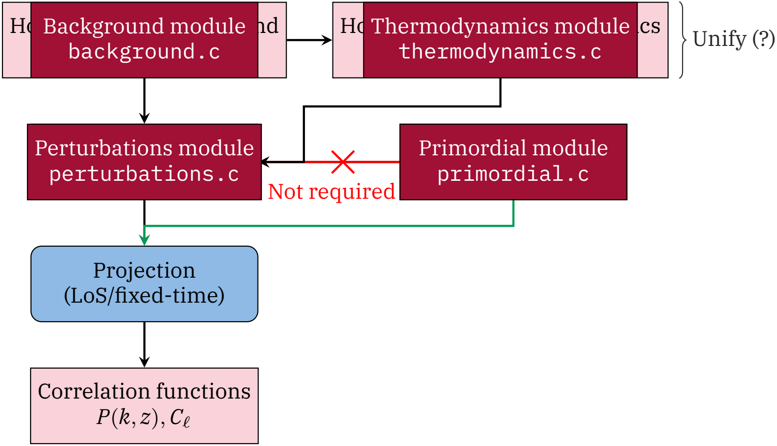
Fundamental layout of Einstein-Boltzmann solvers



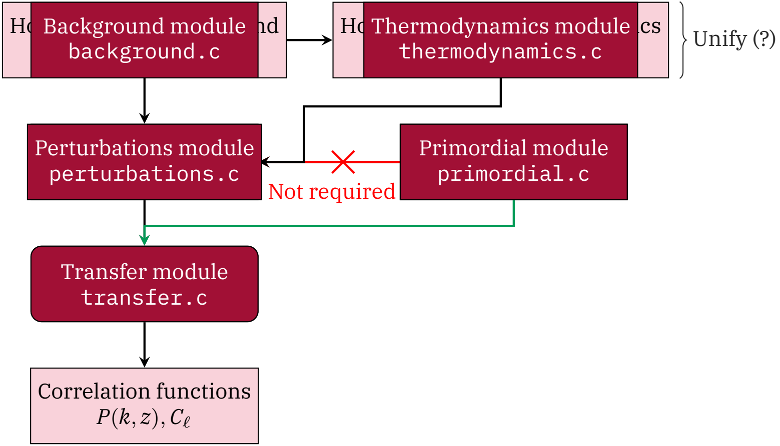
Fundamental layout of Einstein-Boltzmann solvers



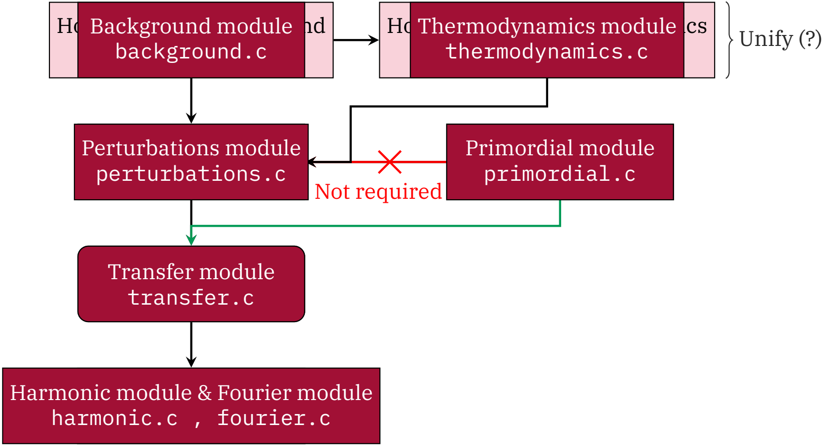
Fundamental layout of Einstein-Boltzmann solvers



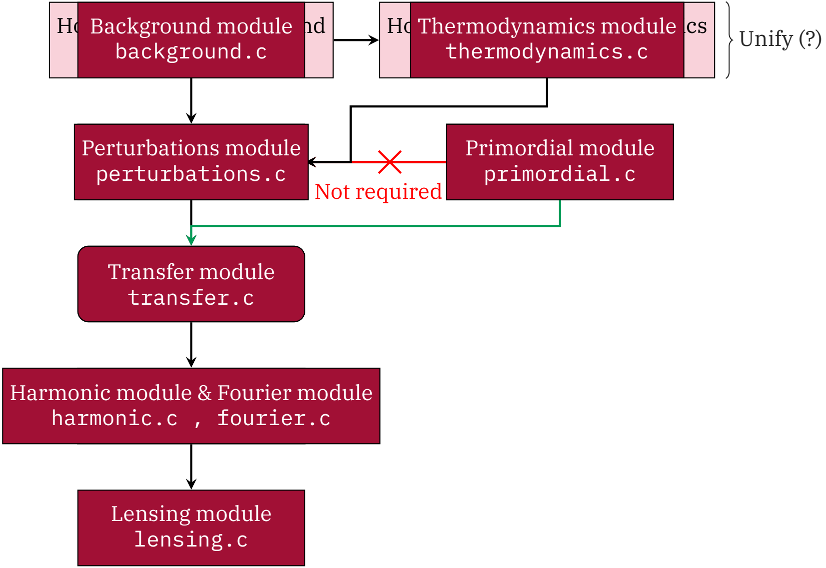
Fundamental layout of Einstein-Boltzmann solvers



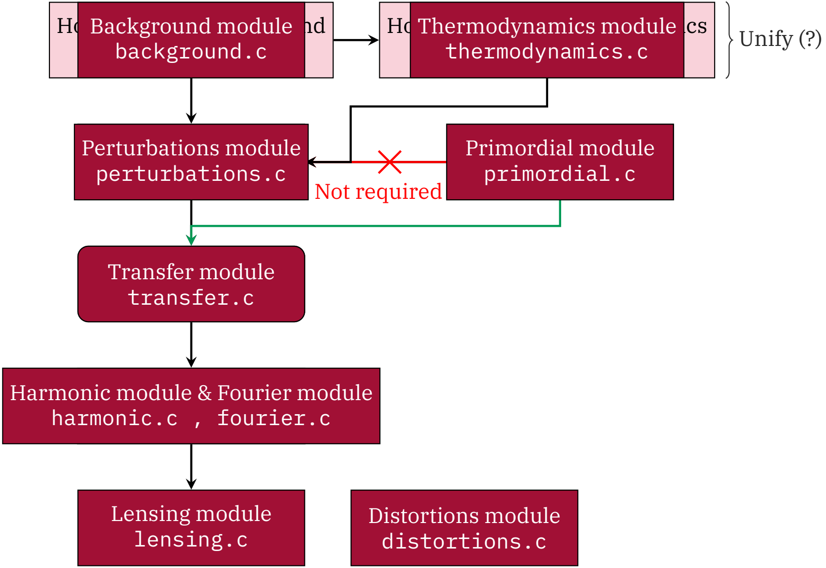
Fundamental layout of Einstein-Boltzmann solvers



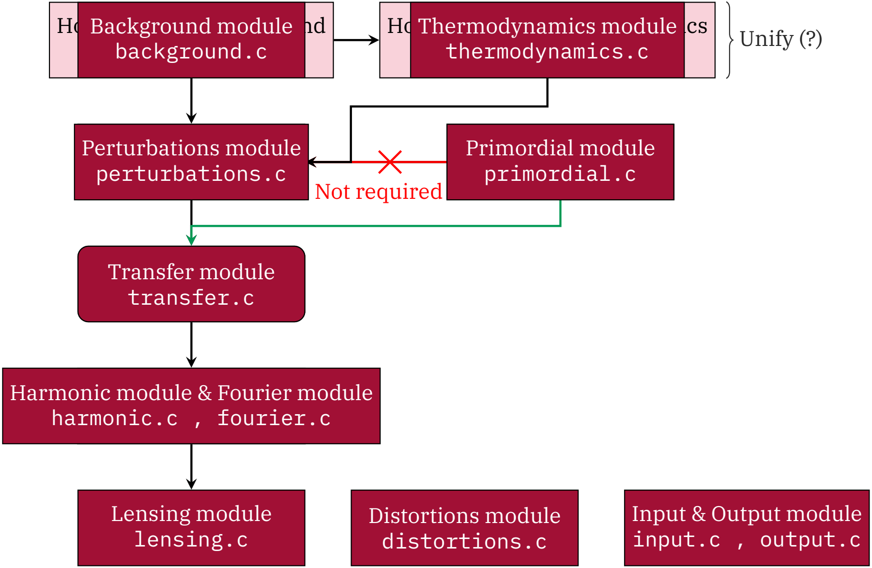
Fundamental layout of Einstein-Boltzmann solvers



Fundamental layout of Einstein-Boltzmann solvers



Fundamental layout of Einstein-Boltzmann solvers



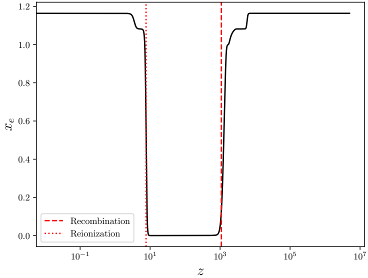
Essential steps in Einstein-Boltzmann solver

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

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

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

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

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

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

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

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

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

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

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

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

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

The Thermodynamics Module

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

1s 2s 2p 3s 3p 3d ... ionized → 1s 2s 2p ionized

Reason: Intermediate transitions (4p→3s) or (3s→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

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

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 $2s, 2p, 3s, \ldots$ with *effective* rates.

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

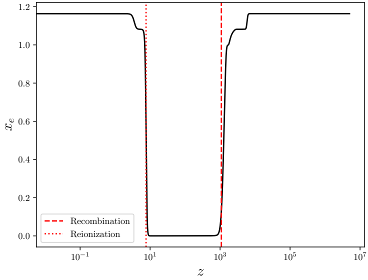
The Thermodynamics Module

recfast only resolves $2s \approx 2p$

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

Fullest 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_He = 0.25`)
- Get it from BBN (`Y_He = 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 $z_{\rm reio}$ for given $\tau_{\rm reio} = \kappa_{\rm reio}$.
Optical depth $\kappa(z)$ = inverse number of expected interactions $\Rightarrow \kappa'(z) = a n_H x_e \sigma_T$

The Thermodynamics Module

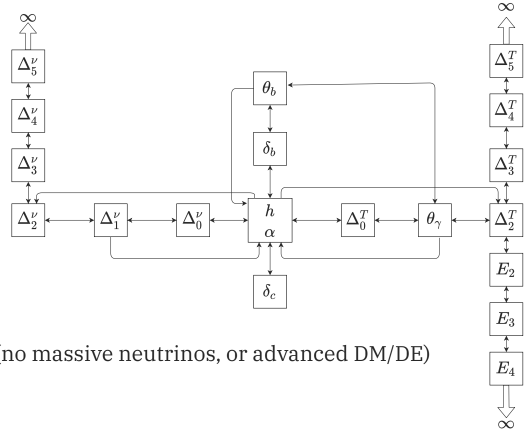
We also include

- 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 $z_{\text{rec}}, z_{\text{drag}}, z_*, D_A(z_{\text{rec}}), r_s(z_{\text{drag}}), \ldots$

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.

- 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.)

Einstein Equations

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

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

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

$$k^2(\phi - \psi) = 12\pi G a^2(\rho + P)\sigma \tag{7}$$

and Boltzmann equations

$$\frac{\mathrm{d}F_0^{(\gamma)}}{\mathrm{d}\eta} + kF_1^{(\gamma)} = 4\phi' \tag{8}$$

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

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

$$\frac{\mathrm{d}F_\ell^{(\gamma)}}{\mathrm{d}\eta} - \frac{k}{2\ell+1}\left[\ell F_{\ell-1}^{(\gamma)-(\ell+1)F_{\ell+1}^{(\gamma)}}\right] = 0 \qquad \text{(infinite hierarchy)} \tag{11}$$

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 $\ell_{\max}^{(\mathrm{dr})}$ massless neutrino hierarchy
b) 1 $\ell_{\max}^{(\mathrm{dr})}$ massless neutrino hierarchy
$+ N_{\mathrm{ncdm}} \cdot \ell_{\max}^{(\mathrm{ncdm})} \cdot N_q$ massive neutrino hierarchies

= Too many equations for simple solvers!
(also tight coupling == stiff equations)
(also sparse system)

The Perturbations Module

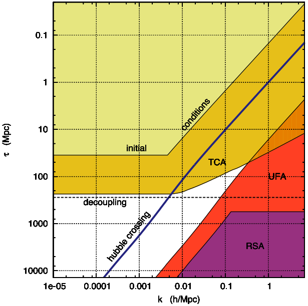
ODE Solver (customized for Einstein-Boltzmann equations)

- Stiff system require implicit method like backward Euler or more advanced:
  $\rightarrow$ find y_{n+1} as a solution of $\mathbf{y_{n+1}} = y_n + y'(\mathbf{y_{n+1}})\delta t$
- 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!

ODE approximations (papers : CLASS II & CLASS IV 2011)
Idea of these approximations: Reduce number of evolved equations.

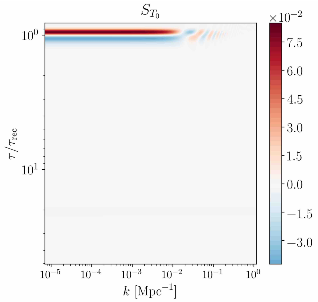
- Tight Coupling Approximation for baryons and γ at 2nd order $\rightarrow$ Suppresses shear & higher moments whenever $k\tau_{b\gamma} \ll 1$
- Ultrarelativistic Fluid Approximation (for massless ν, also one for massive ones): truncated Boltzmann, 3 equations $\rightarrow$ Suppresses higher order moments 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 $k\tau_{b\gamma} \gg 1$)



Source functions

- Keep memory not of everything, but anything useful for final calculation of observables:
  - raw transfer function ($\delta_m(\tau, k) \to P_m(k, z)$)
  - non-trivial combinations (photon, baryon, metric, thermodynamical functions $\to$ 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):
  $(F_\ell, G_\ell) \to (S_T, S_P) \to (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $2\ell_{\mathrm{max}}$ equations, only flat!
- Hu & White 1997 (TAM):
  $(\Theta_\ell, E_\ell, B_\ell) \to (S_T, S_E, S_B) \to (\Delta_\ell^T, \Delta_\ell^E, \Delta_\ell^B)$: $3\ell_{\mathrm{max}}$ equations!

The Perturbations Module

Photon hierarchies

Two approaches to polarization in Boltzmann hierarchy:

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

CMBFAST: first in flat space, second in curved space

CAMB: always second case

`class`: 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

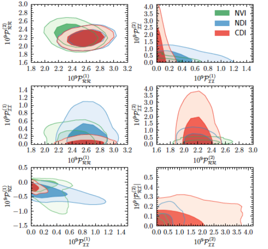


Fig. 22. Two dimensional distributions for power in isocurvature modes, using *Planck*+WP data.

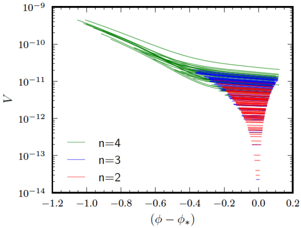


Fig. 14. Observable range of the best-fitting inflaton potentials, when $V(\phi)$ is Taylor expanded at the nth order around the pivot value ϕ_*, in natural units (where $\sqrt{8\pi}M_{\rm pl} = 1$), assuming a flat prior on ϵ_V, η_V, ξ_V^2, and ϖ_V^3, and using *Planck*+WP data.

The Primordial module

Primordial spectra : modes

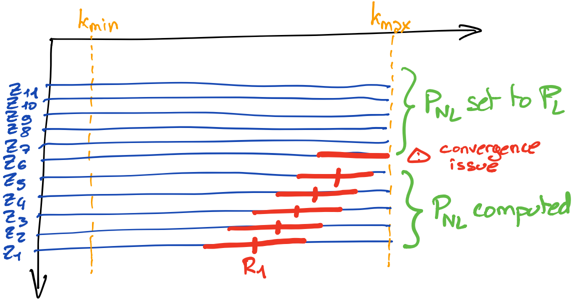
P_k_ini type =	modes =	ic =
analytic_Pk	at least one: s,t	at least one: ad,bi,cdi,nid,niv
inflation_V	s,t	ad
inflation_H	s,t	ad
inflation_V_end	s,t	ad
external_Pk	at least one: s,t	ad

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 $\left(R^{NL}(k,z)\right)^2 = P_m^{NL}(k,z)/P_m(k,z)$ for e.g., CMB lensing, cosmic shear, number count C_ℓ's

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

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



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

The Fourier Module

Halofit relies on simple similarity solution Ansatz:

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

with $y = k/k_{\mathrm{nl}} = kR_{\mathrm{NL}}(z)$.
Parameters calibrated to fit (early) simulations reasonably well.

$\Delta^2_{\mathrm{nl}} \approx \Delta^2_{2-\mathrm{halo}} + \Delta^2_{1-\mathrm{halo}}$ with $\Delta^2_{2-\mathrm{halo}} \approx \frac{(1+\Delta^2_{\mathrm{lin}})^\beta}{(1+\alpha\Delta^2_{\mathrm{lin}})} \cdot \exp\left(-y/4 - y^2/8\right)$ with $\Delta^2_{\mathrm{lin}} = P_{\mathrm{lin}}(k,z)\frac{k^3}{2\pi^2}$

Summary: Simple analytical fitting formula

The Fourier Module

HMcode has a more complicated halo model:

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

$\delta(k,\tau), \theta(k,\tau)$ (perturbations)

$\downarrow$

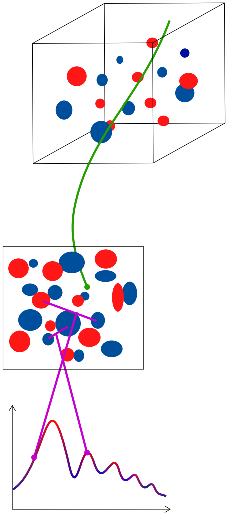
$S^{\text{cmb}}(k,\tau), S^m(k,\tau)$ (source functions)

$\downarrow$
$\downarrow$
$\downarrow$
$\downarrow$
$\downarrow$
$\downarrow$
$\downarrow$
$\downarrow$

$\Delta_\ell^x(k)$ (line-of-sight projection, `transfer.c`)

$\downarrow$
$\downarrow$
$\downarrow$
$\downarrow$

$\int \Delta_\ell^x(k)\Delta_\ell^y(k)\mathrm{d}\ln k$
(correlation and summation, `harmonic.c`)



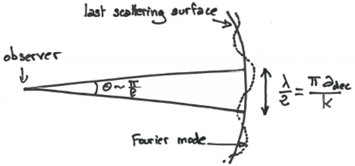
Transfer & Harmonic Modules

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 ($\theta\, d_a(z_{\rm rec}) = \frac{\lambda}{2}$ gives precisely $l = k(\tau_0 - \tau_{\rm rec})$):



Curved space: spherical bessel functions → 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

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

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

comes from integration by part of:

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

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

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.

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

or

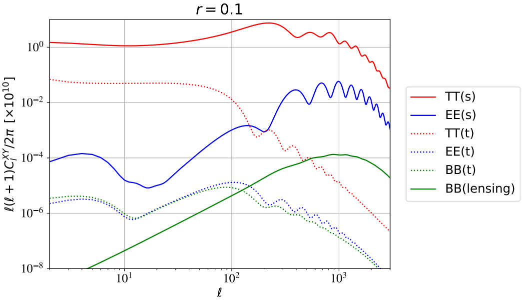
$$S_T^0 = g\left(\frac{\delta_g}{4} + \phi\right) + e^{-\kappa} 2\phi' + g'\theta_b + g\theta_b' \qquad S_T^1 = e^{-\kappa} k(\psi - \phi) \qquad S_T^2 = \frac{g}{8}\,(G_0 + G_2 + F_2)$$

Transfer & Harmonic Modules

Last step is (almost) trivial:

$$C_\ell^{XY} = \int \frac{dk}{k} \sum_{ij} \Delta^X_{\ell\, i}(k) \Delta^Y_{\ell\, j}(k) \mathcal{P}_{ij}(k)$$

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

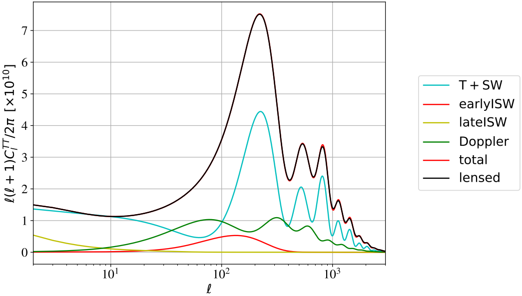


Harmonic exercise

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.



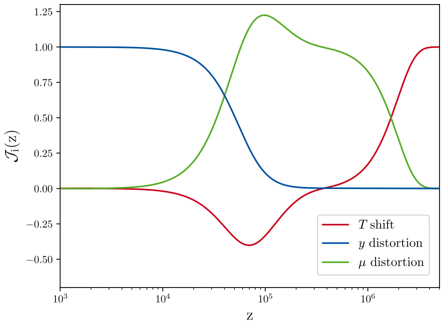
Essentials 10: Spectral Distortions

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

Simplified view:

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

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 occurences of `fld` in `include/*.h` and `source/*.c` (normally they are all within some "`if (has_fld){ ...}`" and you can search directly for occurences 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 occurences of `fld` in `include/*.h` and `source/*.c` (normally they are all within some "`if (has_fld){ ...}`" and you can search directly for occurences of `has_fld`)
4. duplicate these occurences

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 occurences of `fld` in `include/*.h` and `source/*.c` (normally they are all within some "`if (has_fld){ ...}`" and you can search directly for occurences of `has_fld`)
4. duplicate these occurences
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 occurences of `fld` in `include/*.h` and `source/*.c` (normally they are all within some “`if (has_fld){ ...}`” and you can search directly for occurences of `has_fld`)
4. duplicate these occurences
5. change `fld` into `earde`
6. change some equations to describe the specific properties of your feature