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

Visit `http://class-code.net/` for more info!

`class` in Les Karellis

What to expect in this *advanced* lecture:

Theory:	What is `class` based upon?
Coding:	Structure of `class`
Coding:	Essential rules and conventions
Coding:	Implementing features (C and python)
Coding:	Using MontePython/Cobaya with `class`

We will learn the theory behind `class` and the fundamental rules of its code base.

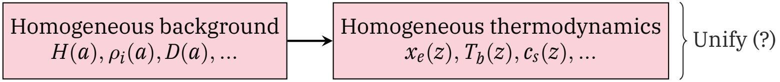
class Theory

1. Fundamental layout of Einstein-Boltzmann solvers
2. Essential steps for each module
3. A few details for each of these steps

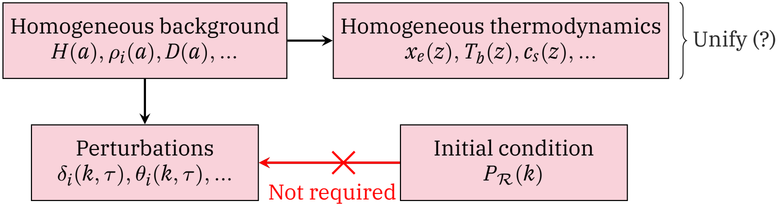
Fundamental layout of Einstein-Boltzmann solvers

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{Homogeneous background}\\ H(a), \rho_i(a), D(a), \ldots\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Homogeneous thermodynamics}\\ x_e(z), T_b(z), c_s(z), \ldots\end{array}}
\end{tikzcd}$$

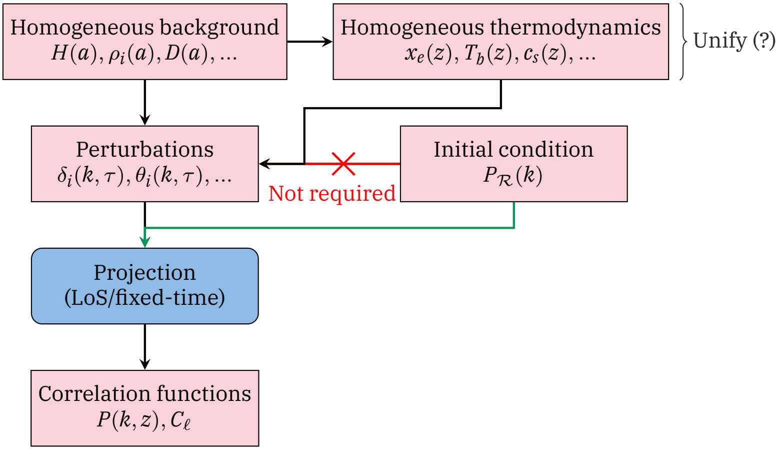
Fundamental layout of Einstein-Boltzmann solvers


Homogeneous background
$H(a), \rho_i(a), D(a), \ldots$
Homogeneous thermodynamics
$x_e(z), T_b(z), c_s(z), \ldots$
Unify (?)

Fundamental layout of Einstein-Boltzmann solvers

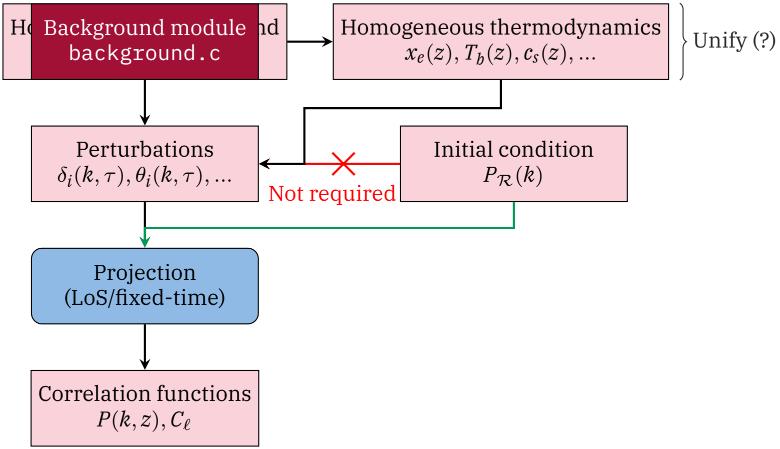

Homogeneous background
$H(a), \rho_i(a), D(a), \dots$
Homogeneous thermodynamics
$x_e(z), T_b(z), c_s(z), \dots$
Unify (?)
Perturbations
$\delta_i(k,\tau), \theta_i(k,\tau), \dots$
Initial condition
$P_{\mathcal{R}}(k)$
Not required

Fundamental layout of Einstein-Boltzmann solvers

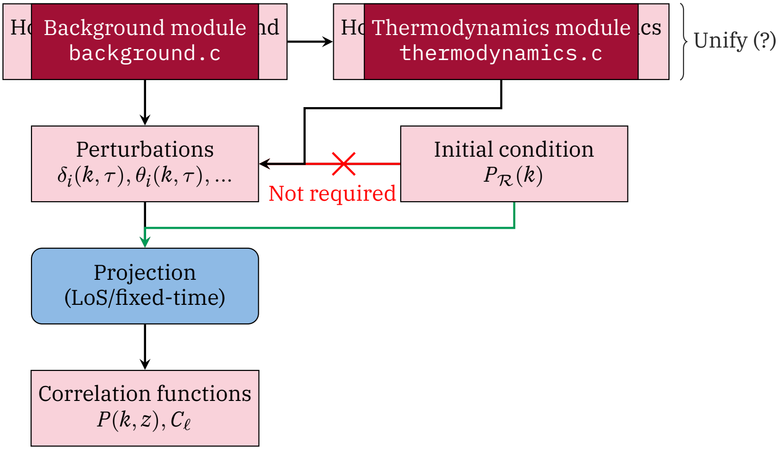


Homogeneous background $H(a), \rho_i(a), D(a), \dots$
Homogeneous thermodynamics $x_e(z), T_b(z), c_s(z), \dots$
Unify (?)
Perturbations $\delta_i(k,\tau), \theta_i(k,\tau), \dots$
Not required
Initial condition $P_{\mathcal{R}}(k)$
Projection (LoS/fixed-time)
Correlation functions $P(k,z), C_\ell$

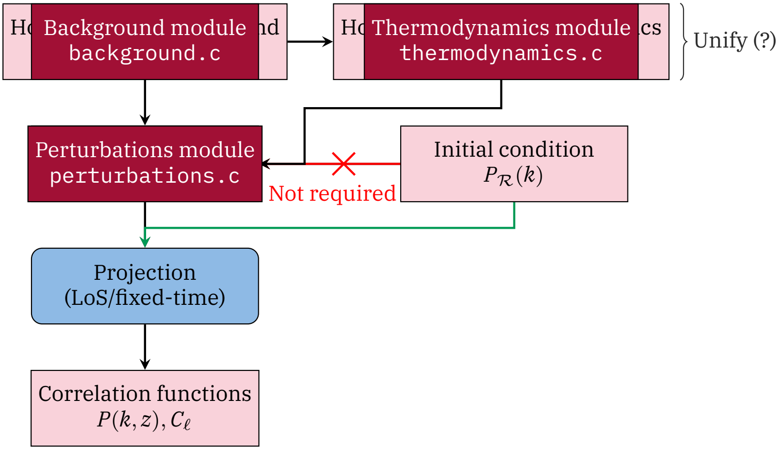
Fundamental layout of Einstein-Boltzmann solvers



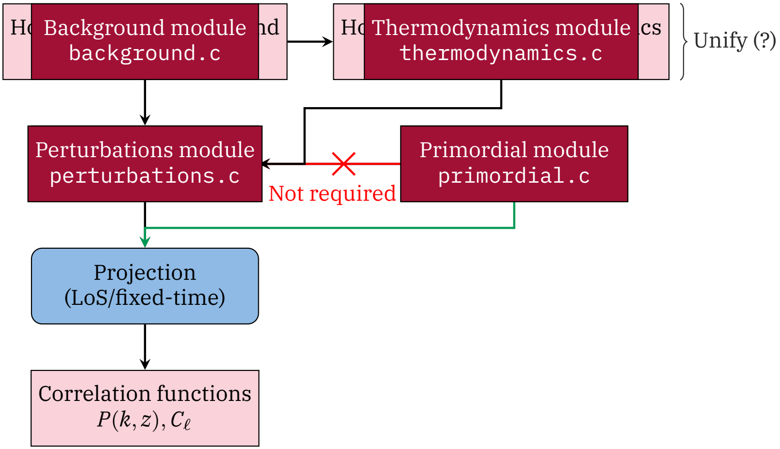
Fundamental layout of Einstein-Boltzmann solvers



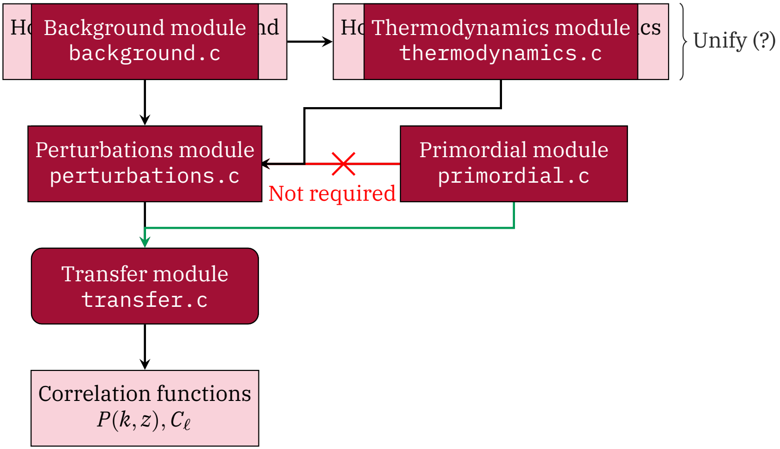
Fundamental layout of Einstein-Boltzmann solvers



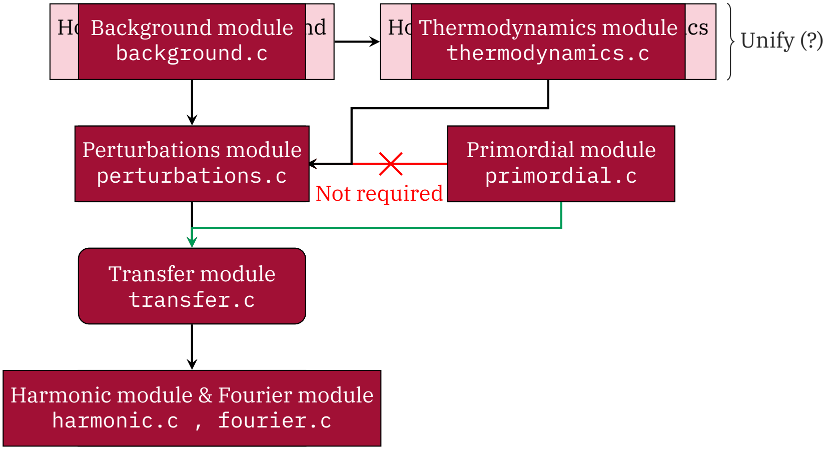
Fundamental layout of Einstein-Boltzmann solvers



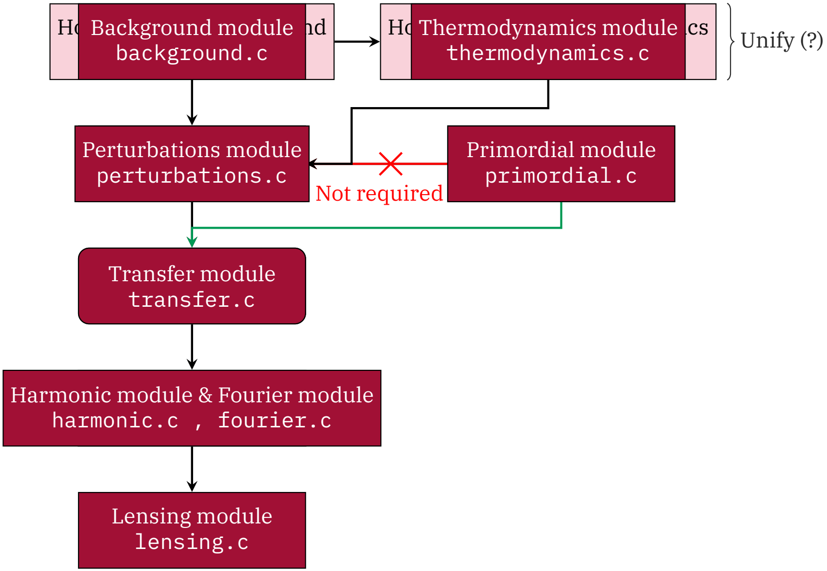
Fundamental layout of Einstein-Boltzmann solvers



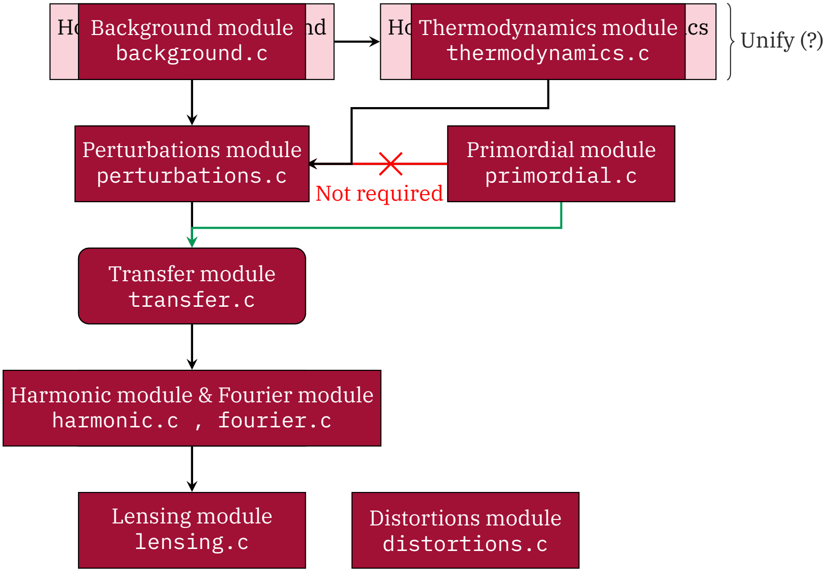
Fundamental layout of Einstein-Boltzmann solvers



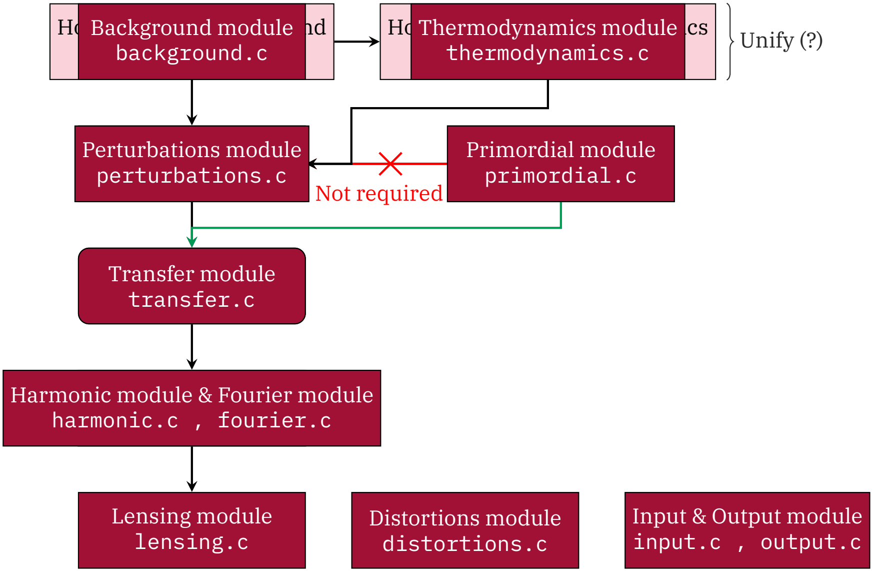
Fundamental layout of Einstein-Boltzmann solvers



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!

Essential steps in Einstein-Boltzmann solver

Module 1. Input

Read in input files, take care of *shooting*.

```
h = 0.7
#H0 = 70
Omega_m = 0.3
#omega_m = 0.14
sigma8=0.8
```

Special care for equivalent/unknown parameters

Input management in `class`

Terminal | **Python wrapper**

Terminal:

file `xxx.ini`

↓

`input_init(...)`

(parser)

↓

Python wrapper:

`.set(...)`

↓

`struct` `file_content fc;` (all parameter names/values stored as arrays of strings)

↓

`input_read_from_file(...)`

↓

`input_read_parameters(...)`

(assign all default values + interprete input + update some parameters)

↓

Only *relevant* parameters get stored in the structures of each module

Input management in class

For indirect parameters, use shooting method

Repeated calls of `input_read_parameters(...)`, class executions, from `input_read_from_file(...)` until shooting target is met.

Input management in class

For indirect parameters, use shooting method

Repeated calls of `input_read_parameters(...)`, class executions, from `input_read_from_file(...)` until shooting target is met.

Example:
How would you code the input parameter θ_s?
Use approximate formula $\rightarrow$ inflexible, inaccurate

Input management in class

For indirect parameters, use shooting method

Repeated calls of `input_read_parameters(...)`, `class` executions, from `input_read_from_file(...)` until shooting target is met.

Example:
How would you code the input parameter θ_s?
Use approximate formula $\rightarrow$ inflexible, inaccurate

Try out a few values and narrow down (Example: User wants $100\theta_s = 1.04325$)

h	$100\theta_s$
0.7	1.0522492086422521
0.65	1.0270326366580724
0.68215616173	1.0437999980620178
0.68110138476	1.0432819283581667
0.68103637942	1.0432499363679562
0.68103650871	1.0432499921072458
0.68103652701	1.0432500079710365
...	...

Input management in `class`

For indirect parameters, use shooting method

Repeated calls of `input_read_parameters(...)`, `class` executions, from `input_read_from_file(...)` until shooting target is met.

Example:
How would you code the input parameter θ_s?
Use approximate formula $\rightarrow$ inflexible, inaccurate

Try out a few values and narrow down (Example: User wants $100\theta_s = 1.04325$)

h	$100\theta_s$
0.7	1.0522492086422521
0.65	1.0270326366580724
0.68215616173	1.0437999980620178
0.68110138476	1.0432819283581667
0.68103637942	1.0432499363679562
0.68103650871	1.0432499921072458
0.68103652701	1.0432500079710365
...	...

In practice, use more sophisticated Ridder's method / Newton's method

Input management in class

For shooting parameters, establish mapping between *target parameter*, *unknown parameter* and *level*. Currently:

target parameter	unknown parameter	level
$100 \times \theta_s$	h	thermodynamics
Ω_dcdm	$\rho^\mathrm{ini}_\mathrm{dcdm}$	background
~~σ_8~~	~~A_s~~	~~spectra~~
...	...	...

... plus a few others (alternative parametrizations of decaying CDM, quintessence parameters).

This is what is used e.g. in models of early dark energy!

If you need to add such parameters: see how it is done e.g. for `100*theta_s` and replicate the structure!

Special exception $\tau_\mathrm{reio} \leftrightarrow z_\mathrm{reio}$ only concerns reionization and is done independently in `thermodynamics.c`
New Special exception: $\sigma_8 \leftrightarrow A_s$ can be very simply analytically re-scaled (multiplicative property), therefore done independently in `input.c`

Input management in class

Budget equation:

$$\sum_X \Omega_X = 1 + \Omega_k$$

To avoid over-constraining the input, one of the last three (`Omega_Lambda`, `Omega_fld`, `Omega_scf`) must be left unspecified and class will assign it using budget equation.

Possibly more advanced in the future

- default: `Omega_Lambda` is automatically adjusted
- if you pass `Omega_Lambda`, `Omega_fld` is automatically adjusted
- if you pass `Omega_Lambda` and `Omega_fld`: `Omega_scf` is automatically adjusted (if you allow, by setting to -1)

This allows whatever combination.
E.g. to get Λ plus a DE fluid:
`Omega_Lambda=0.2`, `Omega_scf=0` or `Omega_fld=0.3`, `Omega_scf=0`

Helpful output by setting background verbose `>= 2`

Essential steps in Einstein-Boltzmann solver

Module 2. Background

Get all background quantities as function of a scale factor a.

This also gives mapping $a \leftrightarrow z \leftrightarrow t \leftrightarrow$ conf.time

Details of the steps in Einstein-Boltzmann solvers

Let’s formalize problem!

Three types of parameters:

- $\{A\}$ are analytical functions of scale factor and $\{B\}$ quantities.
- $\{B\}$ need to be integrated over, and are used to compute $\{A\}$
- $\{C\}$ also need to be integrated over, but are not used to compute $\{A\}$.

Details of the steps in Einstein-Boltzmann solvers

Let's formalize problem!

Three types of parameters:

- $\{A\}$ are analytical functions of scale factor and $\{B\}$ quantities.
- $\{B\}$ need to be integrated over, and are used to compute $\{A\}$
- $\{C\}$ also need to be integrated over, but are not used to compute $\{A\}$.

ΛCDM and many simple extensions:

- $\{A\} = \{\rho_i(a), p_i(a), H(a), ...,\}$ with e.g. $H(a) = \left(\sum_X \rho_X(a) - \frac{K}{a^2}\right)^{1/2}$
- $\{B\} = \{\}$ (eliminated since `v3.0`)
- $\{C\} = \{t, \tau, r_s, D, f\}$ with e.g. $\frac{dt}{da} = 1/H(a)$, $\frac{dr_s}{da} = c_s(a)/(a \cdot H(a))$

Details of the steps in Einstein-Boltzmann solvers

Let's formalize problem!

Three types of parameters:

- $\{A\}$ are analytical functions of scale factor and $\{B\}$ quantities.
- $\{B\}$ need to be integrated over, and are used to compute $\{A\}$
- $\{C\}$ also need to be integrated over, but are not used to compute $\{A\}$.

Example of DE/DM/DR fluid:

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

Details of the steps in Einstein-Boltzmann solvers

Let's formalize problem!

Three types of parameters:

- $\{A\}$ are analytical functions of scale factor and $\{B\}$ quantities.
- $\{B\}$ need to be integrated over, and are used to compute $\{A\}$
- $\{C\}$ also need to be integrated over, but are not used to compute $\{A\}$.

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 $\frac{d\phi}{da} = \phi'/[aH(a)]$, $\frac{d\phi'}{da} = -2\phi' - aV(\phi)/H(a)$

Details of the steps in Einstein-Boltzmann solvers

Let's formalize problem!

Three types of parameters:

- $\{A\}$ are analytical functions of scale factor and $\{B\}$ quantities.
- $\{B\}$ need to be integrated over, and are used to compute $\{A\}$
- $\{C\}$ also need to be integrated over, but are not used to compute $\{A\}$.

Also Cold Dark Matter decaying into Dark Radiation...

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

Details of the steps in Einstein-Boltzmann solvers

Small details:

- Quantities as $D_A(z), D_L(z), r_s, t_\mathrm{age}$ can be derived after all A,B,C are computed
- Takes care of NCDM integration of phase-space distribution
- Useful checks & output
- $\rightarrow$ Budget equation output at verbosity level 2

Essential steps in Einstein-Boltzmann solver

Module 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)

Details of the steps in Einstein-Boltzmann solvers

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

Details of the steps in Einstein-Boltzmann solvers

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

Details of the steps in Einstein-Boltzmann solvers

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

Details of the steps in Einstein-Boltzmann solvers

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

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

Details of the steps in Einstein-Boltzmann solvers

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.

Details of the steps in Einstein-Boltzmann solvers

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

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

Solved numerically, basis of recfast

Details of the steps in Einstein-Boltzmann solvers

recfast only resolves $2s \approx 2p$

Details of the steps in Einstein-Boltzmann solvers

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

Details of the steps in Einstein-Boltzmann solvers

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)

Details of the steps in Einstein-Boltzmann solvers

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)

Details of the steps in Einstein-Boltzmann solvers

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)

Details of the steps in Einstein-Boltzmann solvers

For reionization:

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

Details of the steps in Einstein-Boltzmann solvers

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 $\Rightarrow \kappa'(z) = a n_H x_e \sigma_T$

Details of the steps in Einstein-Boltzmann solvers

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