Lecture 9: The Thermodynamics module

DAY IV : Thursday 30th October

09:30-10:15	CLASS	The thermodynamics module.	JL
10:15-11:00	MontePython	Internal structure of the code.	BA

Coffee

11:30-12:15	CLASS	The perturbation module.	JL

Lunch

13:30-14:15	CLASS	Playing with perturbations.	JL
14:15-15:00	General	Advanced ODE solvers. ndf15.	TT

Tea

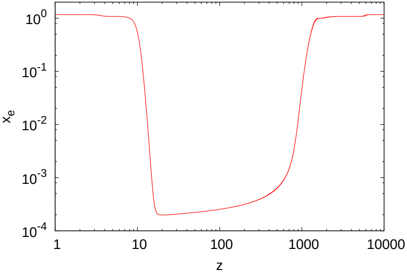
15:45-16:30	Optional	Lecturers will answer questions	

Thermal history

The module must:

- solve recombination and reionisation, to compute the free electron fraction

  $x_e = n_{\rm free\ electrons}/n_p$ (not counting protons in He nuclei, so x_e can be > 1)



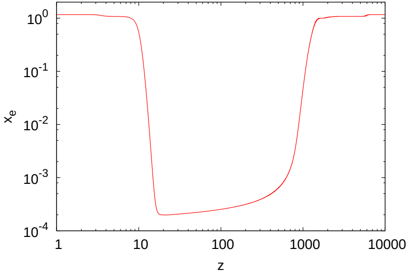
[observe the first/second He recombination, the H recombination (around $T \sim 0.3$ eV, not 13.6 eV!), the H reionisation, the first He reionisation]

Thermal history

The module must:

- solve recombination and reionisation, to compute the free electron fraction

  $x_e = n_{\text{free electrons}}/n_p$ (not counting protons in He nuclei, so x_e can be > 1)



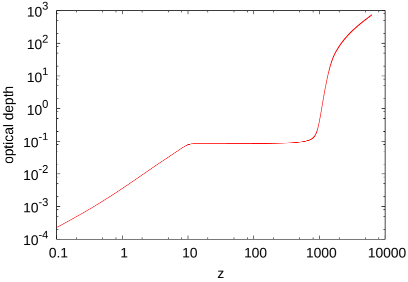
  [observe the first/second He recombination, the H recombination (around $T \sim 0.3$ eV, not 13.6 eV!), the H reionisation, the first He reionisation]

- Thomson scattering rate $\kappa' = \sigma_T a n_p x_e$: universe becomes transparent when $\kappa' < H$, i.e. at recombination

Thermal history

The module must:

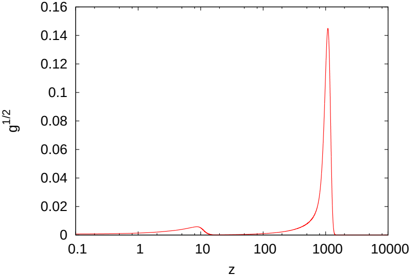
- integrate $\kappa' = \sigma_T a n_p x_e$ to get the optical depth of the cosmic fog, $\kappa(\tau) = \int_\tau^{\tau_0} \kappa' d\tau$



Thermal history

The module must:

- infer the visibility function $g(\tau) = \kappa' e^{-\kappa} =$ probability that last interaction took place at τ



Recombination

$$\mathrm{H}^+ + \mathrm{e}^- \longleftrightarrow \mathrm{H} + \gamma$$

- accurate simulation extremely involved (many excited levels contribute)

Recombination

$$\mathrm{H}^{+} + \mathrm{e}^{-} \longleftrightarrow \mathrm{H} + \gamma$$

- accurate simulation extremely involved (many excited levels contribute)
- RECFAST: Peebles recombination (two levels) with fudge factors

Recombination

$$\mathrm{H}^+ + \mathrm{e}^- \longleftrightarrow \mathrm{H} + \gamma$$

- accurate simulation extremely involved (many excited levels contribute)
- RECFAST: Peebles recombination (two levels) with fudge factors
- also HyRec, CosmoRec. All agree with RECAST v1.6 at precision level of Planck

Recombination

$$\mathrm{H}^+ + \mathrm{e}^- \longleftrightarrow \mathrm{H} + \gamma$$

- accurate simulation extremely involved (many excited levels contribute)
- RECFAST: Peebles recombination (two levels) with fudge factors
- also HyRec, CosmoRec. All agree with RECAST v1.6 at precision level of Planck
- CLASS user can switch between RECFAST 1.6 (coded within `source/thermodynamics.c`) and HyRec (separate code in `hyrec/`) using `recombination = RECFAST` or `recombination = HyRec`.

Recombination

$$\mathsf{H}^+ + \mathsf{e}^- \longleftrightarrow \mathsf{H} + \gamma$$

- accurate simulation extremely involved (many excited levels contribute)
- RECFAST: Peebles recombination (two levels) with fudge factors
- also HyRec, CosmoRec. All agree with RECAST v1.6 at precision level of Planck
- CLASS user can switch between RECFAST 1.6 (coded within `source/thermodynamics.c`) and HyRec (separate code in `hyrec/`) using `recombination = RECFAST` or `recombination = HyRec`.
- RECFAST integrates $\frac{d}{dz}\{x_{\rm H}, x_{\rm He}, T_b\}$. HyRec is more sophisticated.

Recombination

$$\mathrm{H}^+ + \mathrm{e}^- \longleftrightarrow \mathrm{H} + \gamma$$

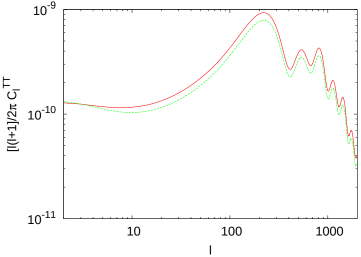
- accurate simulation extremely involved (many excited levels contribute)
- RECFAST: Peebles recombination (two levels) with fudge factors
- also HyRec, CosmoRec. All agree with RECAST v1.6 at precision level of Planck
- CLASS user can switch between RECFAST 1.6 (coded within `source/thermodynamics.c`) and HyRec (separate code in `hyrec/`) using `recombination = RECFAST` or `recombination = HyRec`.
- RECFAST integrates $\frac{d}{dz}\{x_{\mathrm{H}}, x_{\mathrm{He}}, T_b\}$. HyRec is more sophisticated.
- In both cases, CLASS needs to keep in memory an interpolation table for just $\{x_e(z), T_b(z)\}$.

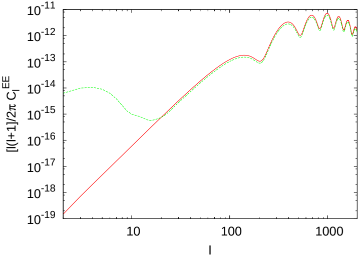
Recombination

- Recombination needs one more cosmological parameter: the primordial Helium fraction $Y_{\rm He}$.
- User can fix it to given value (e.g. `Y_He = 0.25`) or to `Y_He = BBN`. Then the value is infered from an interpolation table computed with a BBN code (Parthenope), for each given value of $N_{\rm eff}$, ω_b (assumes $\mu_{\nu_e} = 0$, easy to generalise).
- BBN interpolation table located in separate directory, in `bbn/bbn.dat`

Reionization

- reionisation very uncertain. Can be probed directly by looking at IGM (Lyman-α, ...) but with large uncertainties.
- CMB probes mainly an integrated quantity, $\tau_{\rm reio} = \int_{\tau_*}^{\tau_0} \kappa' d\tau$, close to 0.09. Gives suppression of C_l's at large l due to rescattering.
- small-l CMB (T and even better E) gives information on history (i.e. on $x_e(z)$, through $\kappa'(z)$).





Reionization

- Since we don't have a compelling model we don't solve for reionisation (like we do for recombination).

Reionization

- Since we don't have a compelling model we don't solve for reionisation (like we do for recombination).
- We impose by hand a functional form for $x_e(z)$ at small redshift, depending on a few free parameters.

Reionization

- Since we don't have a compelling model we don't solve for reionisation (like we do for recombination).
- We impose by hand a functional form for $x_e(z)$ at small redshift, depending on a few free parameters.
- Module in charge of

Reionization

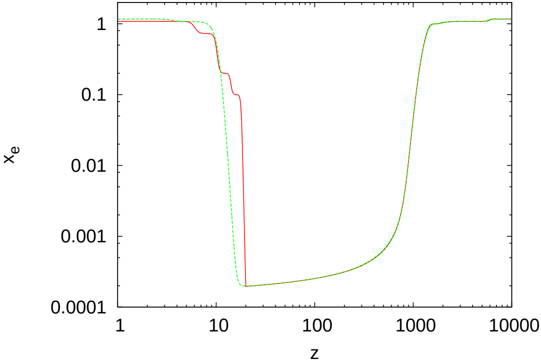
- Since we don't have a compelling model we don't solve for reionisation (like we do for recombination).
- We impose by hand a functional form for $x_e(z)$ at small redshift, depending on a few free parameters.
- Module in charge of
  1. applying the function chosen by the user (`reio_parametrization =`, and extra parameters ...)

Reionization

- Since we don't have a compelling model we don't solve for reionisation (like we do for recombination).
- We impose by hand a functional form for $x_e(z)$ at small redshift, depending on a few free parameters.
- Module in charge of
  1. applying the function chosen by the user (`reio_parametrization =`, and extra parameters ...)
  2. ensuring automatically a smooth transition at some $z \sim 40$ between the solution for $x_e(z)$ computed by the recombination code, and the requested reionisation function.

Reionization

- if `reio_parametrization = reio_camb`, $x_e(z)$ has a $tanh$-shaped step, centered on $z_{\rm reio}$, and matched to the correct value corresponding to freeze-out after recombination. User free to pass either `z_reio = ...` or `tau_reio = ...`. Codes find the missing one automatically, stores it in `pth` (and indicates it in output if `thermodynamics_verbose` > 0).

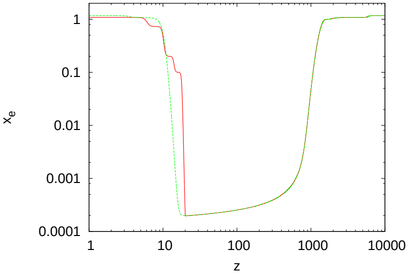


Reionization models

- instead, if `reio_parametrization = reio_bins_tanh`, code assumes a binned reionisation history, with smooth $tanh$ steps between bin centers. User passes e.g.

```
binned_reio_num = 3
binned_reio_z = 8,12,16
binned_reio_xe = 0.7,0.2,0.1
binned_reio_step_sharpness = 0.3
```

- then `tau_reio` cannot be passed in input, but calculated, stored and given in output.



Quantities stored in thermodynamics_table

The table `pth->thermodynamics_table[index_z*pth->th_size+pba->index_th]` has indices:

`index_th_xe`	ionization fraction	x_e
`index_th_dkappa`	Thomson scattering rate	κ' (units Mpc^{-1})
`index_th_tau_d`	Baryon drag optical depth	$\int_\tau^{\tau_0} \frac{4\rho_\gamma}{3\rho_b}\kappa' d\tau$
`index_th_exp_m_kappa`	exp. of (photon) optical depth	$e^{-\kappa}$ with $\kappa = \int_\tau^{\tau_0} \kappa' d\tau$
`index_th_g`	visibility function	$g = \kappa' e^{-\kappa}$
`index_th_Tb`	baryon temperature	T_b given by RECFAST
`index_th_cb2`	squared baryon sound speed	$c_b^2 = \frac{k_B}{\mu} T_b \left(1 - \frac{1}{3}\frac{d\ln T_b}{d\ln a}\right)$
`index_th_rate`	max. variation rate	(for sampling the sources)

(plus extra indices for other derivatives: κ'', κ''', g', g'', $(c_b^2)'$, $(c_b^2)''$).

Look in `include/thermodynamics.h`

External functions in thermodynamics.c

- thermodynamics_at_z(pba,pth,z,...,pvecthermo): interpolates in thermodynamics table (stored in pth) at a given z, returns a vector pvecthermo.

Other useful quantities stored in pth

- thermodynamics_init() finds $z_{\rm rec}$ numerically, by searching for the maximum of the visibility function.

Other useful quantities stored in pth

- thermodynamics_init() finds $z_{\rm rec}$ numerically, by searching for the maximum of the visibility function.
- it stores the related quantities pth->z_rec, pth->tau_rec, pth->rs_rec, pth->ds_rec, pth->ra_rec, pth->da_rec.

Other useful quantities stored in pth

- `thermodynamics_init()` finds $z_{\rm rec}$ numerically, by searching for the maximum of the visibility function.
- it stores the related quantities `pth->z_rec`, `pth->tau_rec`, `pth->rs_rec`, `pth->ds_rec`, `pth->ra_rec`, `pth->da_rec`.
- These quantitites play a crucial role in choosing the sampling of the sources in k-space, because oscillation phase given by $\cos\left(2\pi \frac{d_s(z_{\rm rec})}{\lambda(z_{\rm rec})}\right)$. Also used to print out $\theta_s \equiv \frac{d_s(z_{\rm rec})}{d_a(z_{\rm rec})} = \frac{r_s(z_{\rm rec})}{r_a(z_{\rm rec})}$.

Other useful quantities stored in pth

- thermodynamics_init() finds $z_{\rm rec}$ numerically, by searching for the maximum of the visibility function.
- it stores the related quantities pth->z_rec, pth->tau_rec, pth->rs_rec, pth->ds_rec, pth->ra_rec, pth->da_rec.
- These quantitites play a crucial role in choosing the sampling of the sources in k-space, because oscillation phase given by $\cos\left(2\pi\frac{d_s(z_{\rm rec})}{\lambda(z_{\rm rec})}\right)$. Also used to print out $\theta_s \equiv \frac{d_s(z_{\rm rec})}{d_a(z_{\rm rec})} = \frac{r_s(z_{\rm rec})}{r_a(z_{\rm rec})}$.
- also finds the baryon drag time z_d numerically, such that the baryon optical depth at z_d is one.

Other useful quantities stored in pth

- thermodynamics_init() finds $z_{\rm rec}$ numerically, by searching for the maximum of the visibility function.
- it stores the related quantities pth->z_rec, pth->tau_rec, pth->rs_rec, pth->ds_rec, pth->ra_rec, pth->da_rec.
- These quantitites play a crucial role in choosing the sampling of the sources in k-space, because oscillation phase given by $\cos\left(2\pi \frac{d_s(z_{\rm rec})}{\lambda(z_{\rm rec})}\right)$. Also used to print out $\theta_s \equiv \frac{d_s(z_{\rm rec})}{d_a(z_{\rm rec})} = \frac{r_s(z_{\rm rec})}{r_a(z_{\rm rec})}$.
- also finds the baryon drag time z_d numerically, such that the baryon optical depth at z_d is one.
- it stores the related quantities pth->z_d, pth->tau_d, pth->ds_d, pth->rs_d (the latter gives the phase of the BAOs in large scale structure).

Is RECFAST identical in CLASS and CAMB?

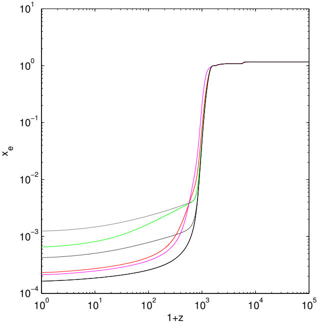
Two differences:

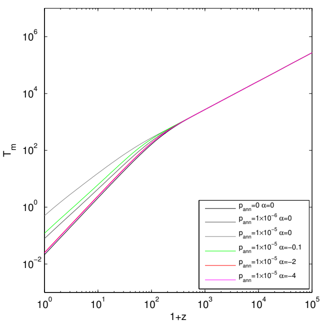
- RECFAST solution slightly smoothed around points where solution is not derivable. Just useful for testing the limit of high accuracy / small stepsize in RECFAST.

Is RECFAST identical in CLASS and CAMB?

Two differences:

- RECFAST solution slightly smoothed around points where solution is not derivable. Just useful for testing the limit of high accuracy / small stepsize in RECFAST.
- several input parameters allow to play with a DM annihilation effect, as described in Giesen et al. 2012. Effect on x_e and T_b, with signatures on CMB.





Printing the thermal history

1. Execute e.g. `./class myinput.ini` including in the input file:

```
write thermodynamics = yes
root = output/toto_
```

Printing the thermal history

1. Execute e.g. `./class myinput.ini` including in the input file:

```
write thermodynamics = yes
root = output/toto_
```

Printing the thermal history

1. Execute e.g. `./class myinput.ini` including in the input file:

   ```
   write thermodynamics = yes
   root = output/toto_
   ```

   The output module will also write a file `output/toto_thermodynamics.dat` with an explicit header:

   ```
   # Table of selected thermodynamics quantitites
   # The following notation is used in column titles:
   # x_e = electron ionisation fraction
   # -kappa = optical depth
   # kappa’ = Thomson scattering rate, prime denotes conformal time
   derivatives
   # g = kappa’ e^-kappa = visibility function
   # Tb = baryon temperature
   # c_b^2 = baryon sound speed squared
   # tau_d = baryon drag optical depth
   #z conf. time [Mpc] x_e kappa’ [Mpc^-1] exp(-kappa)g [Mpc^-1] Tb [K
   ] c_b^2 tau_d
   ```

Printing the thermal history

1. Execute e.g. `./class myinput.ini` including in the input file:

   ```
   write thermodynamics = yes
   root = output/toto_
   ```

   The output module will also write a file `output/toto_thermodynamics.dat` with an explicit header:

   ```
   # Table of selected thermodynamics quantitites
   # The following notation is used in column titles:
   # x_e = electron ionisation fraction
   # -kappa = optical depth
   # kappa' = Thomson scattering rate, prime denotes conformal time
   derivatives
   # g = kappa' e^-kappa = visibility function
   # Tb = baryon temperature
   # c_b^2 = baryon sound speed squared
   # tau_d = baryon drag optical depth
   #z conf. time [Mpc] x_e kappa' [Mpc^-1] exp(-kappa)g [Mpc^-1] Tb [K
   ] c_b^2 tau_d
   ```

2. use python, classy, and `thermo = cosmo.get_thermodynamics()`