Lecture VI: Nonlinear & Transfer

Julien Lesgourgues

EPFL & CERN

London, 15.05.2014

Where are we?

```
int main() {
 # done:
 input_init(pfc,ppr,pba,pth,ppt,ptr,ppm,psp,pnl,ple,pop);
 background_init(ppr,pba);
 thermodynamics_init(ppr,pba,pth);
 perturb_init(ppr,pba,pth,ppt);
 primordial_init(ppr,ppt,ppm);
 # to be done:
 nonlinear_init(ppr,pba,pth,ppt,ppm,pnl);
 transfer_init(ppr,pba,pth,ppt,pnl,ptr);
 spectra_init(ppr,pba,ppt,ppm,pnl,ptr,psp);
 lensing_init(ppr,ppt,psp,pnl,ple);
 output_init(pba,pth,ppt,ppm,ptr,psp,pnl,ple,pop);
}
```

The nonlinear module

`source/nonlinear.c`

The nonlinear module

The goal is to

- compute factors $R^{NL}(k,\tau) = \delta_m^{NL}/\delta_m^L$,
- store them in `pnl->nl_corr_density[...]`,
- in the transfer and spectra module in charge of C_l's and $P(k)$, multiply everywhere $S_i(k,\tau)$ with $R^{NL}(k,\tau)$.

The nonlinear module

The goal is to

- compute factors $R^{NL}(k,\tau) = \delta_m^{NL}/\delta_m^L$,
- store them in `pnl->nl_corr_density[...]`,
- in the transfer and spectra module in charge of C_l's and $P(k)$, multiply everywhere $S_i(k,\tau)$ with $R^{NL}(k,\tau)$.

Old module featured HALOFIT, one-loop perturbation theory, Time Renormalisation Group.

The nonlinear module

The goal is to

- compute factors $R^{NL}(k,\tau) = \delta_m^{NL}/\delta_m^L$,
- store them in `pnl->nl_corr_density[...]`,
- in the transfer and spectra module in charge of C_l's and $P(k)$, multiply everywhere $S_i(k,\tau)$ with $R^{NL}(k,\tau)$.

Old module featured HALOFIT, one-loop perturbation theory, Time Renormalisation Group.

New module only features HALOFIT so far (including Takahashi et al. `arxiv:1208.2701`, extended to massive neutrinos by S. Bird in 2014, following the method of Bird et al. 2011).

The nonlinear module

The goal is to

- compute factors $R^{NL}(k,\tau) = \delta_m^{NL}/\delta_m^L$,
- store them in `pnl->nl_corr_density[...]`,
- in the transfer and spectra module in charge of C_l's and $P(k)$, multiply everywhere $S_i(k,\tau)$ with $R^{NL}(k,\tau)$.

Old module featured HALOFIT, one-loop perturbation theory, Time Renormalisation Group.

New module only features HALOFIT so far (including Takahashi et al. `arxiv:1208.2701`, extended to massive neutrinos by S. Bird in 2014, following the method of Bird et al. 2011). In project:

- other fitting formulas,
- restore one-loop PT,
- linking with recent codes of Scoccimarro et al., Bernardeau et al., Pietroni et al. featuring various renormalisation approaches.

External functions in nonlinear

- `nonlinear_k_nl_at_z()`: returns k_{NL} at given z
- `nonlinear_init()`
- `nonlinear_free()`

External functions in nonlinear

- `nonlinear_k_nl_at_z()`: returns k_{NL} at given z
- `nonlinear_init()`
- `nonlinear_free()`

Input parameter:

```
non linear = halofit (or blank)
```

External functions in nonlinear

- `nonlinear_k_nl_at_z()`: returns k_{NL} at given z
- `nonlinear_init()`
- `nonlinear_free()`

Input parameter:

```
non linear = halofit (or blank)
```

For HALOFIT,

- `nonlinear_pk_l()` computes linear spectrum $P_L(k,\tau)$ given primordial spectrum and $\delta_m(k,\tau)$ (works only for adiabatic IC).

External functions in nonlinear

- `nonlinear_k_nl_at_z()`: returns k_{NL} at given z
- `nonlinear_init()`
- `nonlinear_free()`

Input parameter:

```
non linear = halofit (or blank)
```

For HALOFIT,

- `nonlinear_pk_l()` computes linear spectrum $P_L(k,\tau)$ given primordial spectrum and $\delta_m(k,\tau)$ (works only for adiabatic IC).
- `nonlinear_halofit()` computes non-linear spectrum $P_{NL}(k,\tau)$ given linear spectrum.

External functions in nonlinear

- `nonlinear_k_nl_at_z()`: returns k_{NL} at given z
- `nonlinear_init()`
- `nonlinear_free()`

Input parameter:

```
non linear = halofit (or blank)
```

For HALOFIT,

- `nonlinear_pk_l()` computes linear spectrum $P_L(k,\tau)$ given primordial spectrum and $\delta_m(k,\tau)$ (works only for adiabatic IC).
- `nonlinear_halofit()` computes non-linear spectrum $P_{NL}(k,\tau)$ given linear spectrum.
- `nonlinear_init()` stores the ratio at all times and scales.

External functions in nonlinear

- `nonlinear_k_nl_at_z()`: returns k_{NL} at given z
- `nonlinear_init()`
- `nonlinear_free()`

Input parameter:

```
non linear = halofit (or blank)
```

For HALOFIT,

- `nonlinear_pk_l()` computes linear spectrum $P_L(k,\tau)$ given primordial spectrum and $\delta_m(k,\tau)$ (works only for adiabatic IC).
- `nonlinear_halofit()` computes non-linear spectrum $P_{NL}(k,\tau)$ given linear spectrum.
- `nonlinear_init()` stores the ratio at all times and scales.
- for small τ: code detects that non-linear scale is outside k range required by the user or the code. Then $R^{NL}(k,\tau) = 1$.

The transfer module

`source/transfer.c`

The transfer module

The goal is to compute harmonic transfer functions by performing several integrals of the type

$$\Delta_l^X(q) = \int d\tau \;\; S_X(k(q),\tau) \;\; \phi_l^X(q,(\tau_0-\tau))$$

for each mode, initial conditions, and several types of source functions. In flat space $k = q$.

Calculation done for few values of l (controlled by precision parameters). C_l's are interpolated later.

The transfer module

$$\Delta_l^X(q) = \int d\tau \;\; S_X(k,\tau) \;\; \phi_l^X(q,(\tau_0-\tau))$$

In flat space and for CMB transfer functions:

mode	type	source	Bessel
scalar	temperature T0	S_T^0	$j_l(x), x \equiv k(\tau_0-\tau)$
	temperature T1	S_T^1	j_l'
	temperature T2	S_T^2	$\frac{1}{2}(3j_l'' + j_l)$
	polarisation E	S_P	$(...)\frac{j_l}{x^2}$
tensor	temperature T2	S_T^2	$(...)\frac{j_l''}{x^2}$
	polarisation E	S_P	$\frac{1}{4}\left(-j_l + j_l'' + 2\frac{j_l}{x^2} + 4\frac{j_l'}{x}\right)$
	polarisation B	S_P	$\frac{1}{2}\left(j_l' + 2\frac{j_l}{x}\right)$
vector	...	...	...

Much more complicated in non-flat space, and involving hyperspherical Bessel functions, different for each q.

The transfer module

$$\Delta_l^X(q) = \int d\tau \;\; S_X(k,\tau) \;\; \phi_l^X(q,(\tau_0-\tau))$$

In flat space and for LSS transfer functions:

mode	type	source	bessel
scalar	density in bin i	$W_i(\tau)\delta_m(\tau,k)+...$	j_l
	CMB lensing	$-\tilde{W}_l(\tau,\tau_{\rm rec})(\phi(\tau,k)+\psi(\tau,k))$	j_l
	(with	$\tilde{W}_l = \frac{(\tau-\tau_{\rm rec})}{(\tau_0-\tau)/(\tau_0-\tau_{\rm rec})}$)	
	lensing in bin i	$-\int d\tau' W_i(\tau')\tilde{W}_l(\tau,\tau')(\phi(\tau,k)+\psi(\tau,k))$	j_l

Choose window functions $W_i(z)$ with input parameters:

```
selection=gaussian (dirac, tophat)
selection_mean = 1.,2.,3.
selection_width = 0.5
non_diagonal=1
```

Look also at options in explanatory.ini for global selection function
`dNdz_selection=analytic (filename)`, `dNdz_evolution`, `bias`.

Density source functions

In CLASS v≥2.1, the galaxy number count C_l's refer to

$$\Delta(\mathbf{n},z) = D_g + \Phi + \Psi + \frac{1}{\mathcal{H}}\left[\Phi' + \partial_r(\mathbf{V}\cdot\mathbf{n})\right] + \left(\frac{\mathcal{H}'}{\mathcal{H}^2} + \frac{2}{r_S\mathcal{H}}\right)\left(\Psi + \mathbf{V}\cdot\mathbf{n} + \int_0^{r_S} dr(\Phi' + \Psi')\right) + \frac{1}{r_S}\int_0^{r_S} dr\left[2 - \frac{r_S - r}{r}\Delta_\Omega\right](\Phi + \Psi).$$

(gauge-independent density, redshift space distorsions, Doppler effect, lensing, gravitational corrections). However, by default, only the first term will be computed.

These effects have been first developed in the branch CLASSgal, arXiv:1307.1459, `http://cosmology.unige.ch/tools/`, and then imported into the main code.

The transfer module

Here, a single source in the perturbation module (e.g. $\delta_m(\tau, k)$) leads to several sources in the transfer module (e.g. $W_i(\tau)\delta_m(\tau, k)$). Hence, the code considers that perturbation types and transfer types are two different things, with a correspondance

```
int index_tt_t0;          /* inferred from  index_tp_t0 */
int index_tt_t1;          /* inferred from  index_tp_t1 */
int index_tt_t2;          /* inferred from  index_tp_t2 */
int index_tt_e;           /* inferred from  index_tp_p  */
int index_tt_b;           /* inferred from  index_tp_p  */
int index_tt_lcmb;        /* from index_tp_phi_plus_psi */
int index_tt_density;     /* from index_tp_delta_m, ... */
....
int index_tt_lensing;     /* from index_tp_phi_plus_psi */
....

int * tt_size;
```

The function transfer_init()

- define dynamical indices with `transfer_indices_of_transfers()`.
- interpolate all sources along k with `transfer_perturbation_source_spline()`, k grid is finer in transfer module than in perturbation module.
- compute all flat Bessel functions
- loop over q (or k in flat space).
- : if non-flat, compute hyperspherical Bessel functions for this q
- : loop over modes, initial conditions, types
- : : translate `_tp_` in `_tt_` source, and eventually redefine time sampling
- : : loop over l
- : : : integrate transfer function, or approximate it by Limber, or by zero

$$\Delta_l^X(q) = \int d\tau \;\; S_X(k,\tau) \;\; \phi_l^X(q,(\tau_0-\tau))$$

Plotting a transfer function with test/test_transfer.c

```
input_init_from_arguments(argc, argv,&pr,&ba,&th,&pt,&tr,&pm
    ,&sp,&nl,&le,&op,errmsg);

background_init(&pr,&ba);
thermodynamics_init(&pr,&ba,&th);
perturb_init(&pr,&ba,&th,&pt);
transfer_init(&pr,&ba,&th,&pt,&tr);

/* choose a mode (scalar, tensor, ...) */
int index_mode=pt.index_md_scalars;

/* choose a type (temperature, polarization, grav. pot.,
    ...) */
int index_type=pt.index_tt_t0;

/* choose an initial condition (ad, bi, cdi, nid, niv, ...)
    */
int index_ic=pt.index_ic_ad;
```

Plotting a transfer function with test/test_transfer.c

```
output=fopen("output/test.trsf","w");

for (index_l=0; index_l<tr.l_size[index_mode]; index_l++) {
   for (index_q=0; index_q<tr.q_size; index_q++) {

      transfer = tr.transfer[index_mode]
          [((index_ic * tr.tt_size[index_mode] + index_type)
           * tr.l_size[index_mode] + index_l)
           * tr.q_size + index_q];

      fprintf(output,"%d %e %e %e\n",
              tr.l[index_l],
              tr.q[index_q],
              tr.k[index_mode][index_q],
              transfer);
   }
}
```

For instance you can type:

```
> make test_transfer
> ./test_transfer my_input.ini
```

Exercises

Exercise III

Compare the evolution of $\phi(k,\tau)$ and $\psi(k,\tau)$ for $k = 0.01, 0.1/\mathrm{Mpc}$. Check that they are not equal on super-Hubble scales. To understand why, plot $a^2(\bar{\rho}_\nu + \bar{p}_\nu)\sigma_\nu$ versus time and check that the results are consistent with the Einstein equation $k^2(\phi - \psi) = 12\pi G a^2(\bar{\rho} + \bar{p})\sigma_{\rm tot}$.

Exercise IV

Modify the first Einstein equation in the synchronous gauge like:

$$k^2\eta - \frac{1}{2}\frac{a'}{a}h' = -\mu(k,\tau)\ 4\pi G a^2 \bar{\rho}_{\rm tot}\delta_{\rm tot}\ .$$

Localise the above equation and implement, for instance, $\mu = 1 + a^3$. Print the evolution of ϕ and ψ in the standard and modified models, and conclude that the C_l^{TT}'s should be affected only through the late ISW effect. Get a confirmation by comparing directly the C_l's.

Exercise V

By doing small modifications of `perturb_sources()`, check whether the tensor temperature spectrum $C_l^{TT,\rm tens.}$ comes mainly from photon perturbations on the last scattering surface, or from an integrated Sachs-Wolfe effect.