CMBquick 1. Documentation

This is the doc file CMBquickDoc1.nb of version 0.0.3 of `CMBquick1``. Last update on 22 September 2013..

■ Preambule

Author

Cyril Pitrou

Institute of Astrophysics of Paris (IAP, France)

`cyril.pitrou@ens-lyon.org`

`http://www2.iap.fr/users/pitrou/cmbquick.htm`

Intro

`CMBquick1`` `is a` *Mathematica* Package which provides tools to compute linear perturbations solutions in cosmology.

It is an independent implementation from CMBFAST or CAMB (it uses a different gauge, different time steps, different k-spacing).

Since computations in *Mathematica* are slower than in low level compiled code like Fortran or C, it is unavoidably slower.

However, the purpose of this package is not the speed of computation, but rather to have a code easy to

understand, and thus to modify. Additionaly, its functional design enables the user to compute linear perturbations solutions interactively.

This enables to check, plot, output and understand all the steps of the computation in a straightforward way, and is thus a pedagogical tool for linear perturbations.

The main features provided are

1.Background integration (Friedmann equation, Recombination history),

2.First order transfer functions for all geometrical and physical variables, including perturbed recombination,

3.Power spectra of matter and gravitational potentials,

4.CMB multipoles for polarisation and temperature, with the possibility of adding lensing, 5.Support of tensor modes, massive neutrinos, but not curvature.

See `History[]` and `ToBeDone[]` for more information about versions and future developments

`CMBquick2`` is an extension of `CMBquick1`` for second-order perturbation theory. See the corresponding documentation file.

However it is still in "alpha" version and is incomplete.

Loading the package

Loading the package is straightforward once it has been installed :

You just have to evaluate '<< CMBquick/CMBquick1.m;'

This loads all the definition, but does not start any computation.

We can also check how much memory space

it takes to store all these definitions, and the time it takes to load them (less than 1 second!):

```
In[1]:= MemBefore = MemoryInUse[];
        << CMBquick/CMBquick1.m; // Timing // First

        General::obspkg :
          PlotLegends` is now obsolete. The legacy version being loaded
             may conflict with current Mathematica functionality.
             See the Compatibility Guide for updating information.

           -----------------------------------------------------------------------------
             -------

           These packages come with ABSOLUTELY
             NO WARRANTY; for details type Disclaimer[].
           This is free software, and you are welcome to
             redistribute it under certain conditions.
           See the General Public License for details.

           -----------------------------------------------------------------------------
             -------

           Package CMBquick`CMBquick1`  version 0.0.3, {2014, 3, 19}

           CopyRight (C) 2009-2012, Cyril Pitrou, under the General Public License.

           -----------------------------------------------------------------------------
             -------

           Please send me your comments and
             don't hesitate to report the bugs and mistakes

           at cyril.pitrou@ens-lyon.org

           For information, browse also the
             Add-Ons in the Documentation Center of the Help.

           You can also find worked examples in the
             'Examples' subdirectory of your CMBquick installation.

           You can also download the pdf of the documentation on the  CMBquick  page.

           For information on versions and
             future development, type 'ToBeDone[]' and 'History[]'

           -----------------------------------------------------------------------------
             -------

           A file with interpolation
              functions for the Bessel functions has been found in
           /Users/pitrou/Library/Mathematica/Applications/CMBquick/Data/jls.dat
           It will be used if you load it with 'LoadAndGenerateBesselsBinary[lmax]'
           unless you erase it and recreate it.
           ------------------------------------------------------------------------
              -------------

Out[2]= 1.459687

In[3]:= MemoryInUse[] - MemBefore

Out[3]= 22 555 104
```

GPL

```
In[4]:= Disclaimer[]
```

```
These are points 11 and 12 of the General Public License:

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
  FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT
  WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER
  PARTIES PROVIDE THE PROGRAM `AS IS´ WITHOUT WARRANTY OF ANY KIND,
  EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
  PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF
  THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU
  ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
  WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
  REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR
  DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL
  DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM
  (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED
  INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF
  THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER
  OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
```

Useful definitions for this notebook.

A (CMB)quick overview

■ The main parameters

Numerical parameters

The computation of the multipoles C_l' s is based on the line of sight approach. See Hu & White for instance for more details on the method.

We first solve the first order system of coupled equation (Einstein-Boltzmann system) for a given set of k which spans from the variable `kmin` to `kmax` using a number `Nk` of Fourier modes k.

We solve first the first few moments of the Boltzmann hierarchy in order to obtain the expression of the emitting sources, and use an integral solution on these emitting sources in order to obtain all the multipoles Cl's.

All Fourier modes are in units of k_{eq}, that is the Fourier mode at equivalence. (between radiation and matter).

```
In[18]:= Grid[{{"kmin", kmin}, {"kmax", kmax}, {"Nk", Nk}}, Frame → All]
```

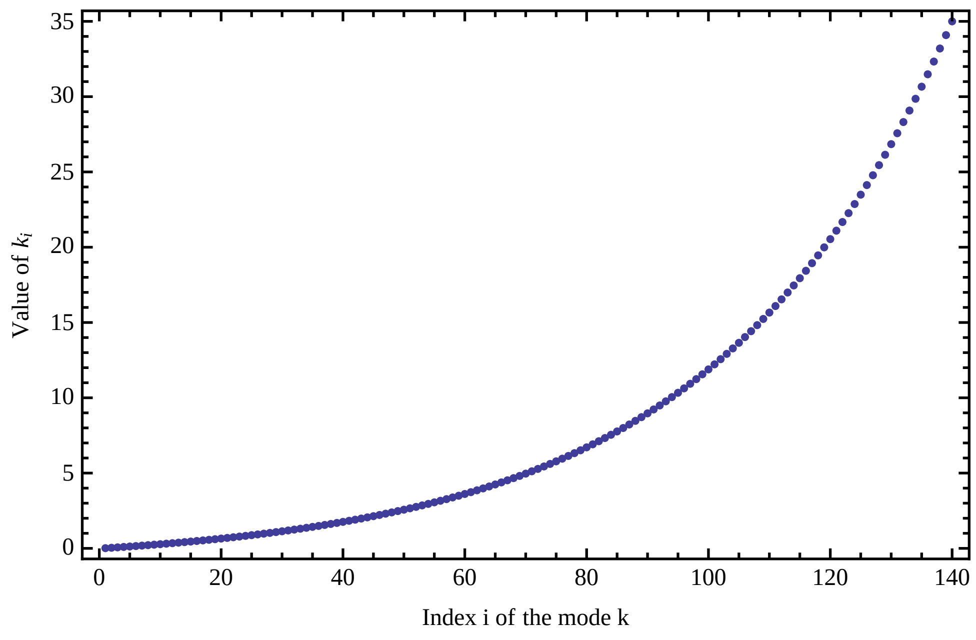
Out[18]=

kmin	0.0133333
kmax	35
Nk	140

Points in this k sampling are logarithmically spaced and the list of points is stored in ListK. By doing this we sample more the small modes. This is especially required when dealing with reionization.

```
In[19]:= ListPlot[ListK, Frame → True,
          FrameLabel → {"Index i of the mode k", "Value of ki"}]
```

Out[19]=



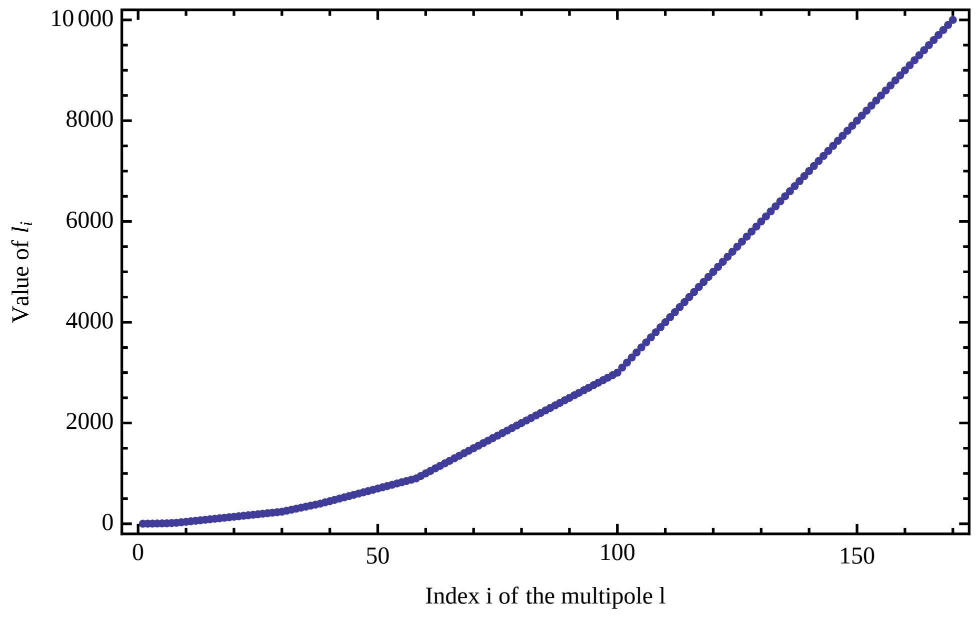
The sources are solved by integrating numerically the Coupled system of Einstein - Boltzmann equation. See the relevant section for more details.

The multipoles are computed from the line of sight approach. We compute then the mutlipoles C_l s only for a sub sample of the possible l's. This list is stored in `ListlUsedinBesselFunction`.

We can see that the low l are more sampled

```
In[20]:= ListPlot[ListlUsedinBesselFunction, Frame → True,
          FrameLabel → {"Index i of the multipole l", "Value of li"}]
```

Out[20]=



Physical parameters and cosmology

The physical parameters of the model are all contained in a list of parameters, that we call a ' Cosmological Parameters List'. Its generic name is thus '`cpl`'.

The list "CPL" contains the cosmological parameters reported in the best fit of WMAP5 (see their Table).

The cpl is passed as an argument to nearly everything, since all the quantities of our cosmology depend on the cosmological parameters of the model.

In this form CPL is thus just a list of the parameters with no explanations.

```
In[21]:=  CPL

Out[21]=  {2.726, 0.719, 3.046, 0.1326, 0.02273, 2.41, 0.963, 0.087, 1., {}, 0, 1}
```

It can be reshaped in a human readable form, with additional parameters of the cosmology which can be derived from it :

```
In[22]:=  Cosmology[CPL, BackgroundParameters]
```

Out[22]=

Variable	Value	Units	Comment
Ω_{b0}	0.043969		Abundance of baryons
Ω_{c0}	0.21253		Abundance of CDM
Ω_{r0}	0.000080966		Abundance of radiation (massless ν's and photons)
$\Omega_{\Lambda 0}$	0.74342		Abundance of Λ
Ω_K	0		Abundance of curvature
T_0	2.726	K	Temperature of CMB
N_ν	3.046		Number of massless neutrinos
h	0.719		Reduced Hubble constant
τ_{rei}	0.087		Optical depth of reionization
n_s	0.963		Scalar perturbations spectral index
k_{eq}	0.0096685	Mpc^{-1}	k at equivalence time
z_{rei}	11.045		Redshift at reionization
z_{eq}	3167.		Redshift at equivalence
z_{LSS}	1059.8		Redshift at $\tau - \tau_{rei} = \ln(2)$
z_{dec}	1089.		Redshift at max of visibility function $(\tau' e^{-\tau})$
z_*	1089.6		Redshift at $\tau - \tau_{rei} = 1$
$d_A(z_*)$	14108.	Mpc	Angular distance at z_*
$d_A(z_{eq})$	14272.	Mpc	Angular distance at equivalence
D_H	4169.58	Mpc	Hubble distance today
t_0	13.6849	Gyears	Age of the Universe
t_*	380 300.	years	Age of universe at z_*
$r_{hor}(z_{dec})$	285.66	Mpc	Radius of horizon at z_{dec}
η_0	14394.	Mpc	Conformal time today

If we had set the option 'PerturbationParameters' in the command above,
then we would also obtain information about primordial perturbations and σ_8

In order to derive the parameters of the cosmology out of the list of parameters CPL, it was required to

-Integrate the Friedmann equation

-Integrate the recombination equation to obtain the recombination history and thus the optical properties of the universe

■ Background equations

Friedmann equations

The basic parameter of our time integration is not the conformal time but the the reduced scale factor $y = a/a_{eq}$. The value of y today is y0.

The radiation era is this defined by $y < 1$,
and the matter era is defined by $y > 1$. The value of this reduced scale factor today is :

```
In[23]:= y0[CPL]

Out[23]= 3168.
```

The conformal time is then inferred by integrating the Friedmann equation.

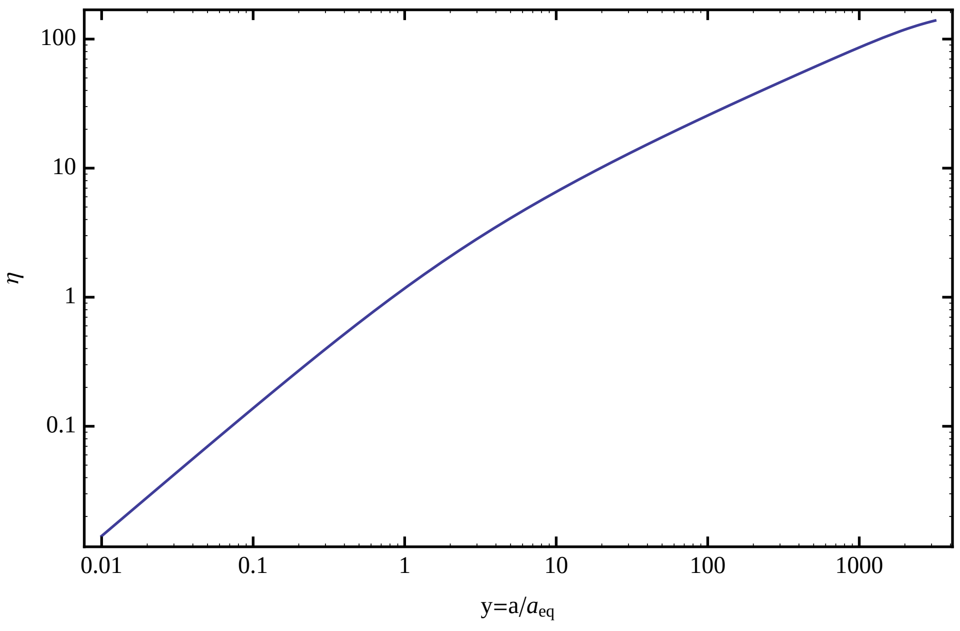
This leads to the function η(y) whose name is 'etaOFy' and its inverse y(η) whose name is 'yOFeta'. Note that all converting function will be of the form 'aOFb'

The conformal time today is thus, in units of ${k_{eq}}^{-1}$

```
In[24]:= etaOFy[CPL]@y0[CPL]

Out[24]= 139.165
```

And we can check that in the radiation era ($y < 1$) $\eta \sim a$ whereas in the matter era ($y > 1$) $\eta \sim a^{1/2}$,
and finally in the era dominated by Λ we have $\Delta\eta \sim -1/a$.

```
In[25]:= LogLogPlot[etaOFy[CPL][y], {y, 0.01, y0[CPL]},
          Frame → True, FrameLabel → {"y=a/a_eq", "η"}]

Out[25]=
```



Other variables are then available. The redshift ' z', and also the scale factor normalized to 1 when T = 13.6 eV which is named ' x'.

The temperature today in eV is thus

```
In[26]:= 13.6 / xOFy[CPL][y0[CPL]]

Out[26]= 0.000234812
```

Recombination

In order to solve for the first order Boltzmann equation, we need to infer the fraction of free electron throughout the history of the universe. Free electrons recombine successively the Helium I I in Helium I, then the Helium I in Helium0.

Then the remaining free electrons are to recombine the H+ in neutral hydrogen.

The first part about Helium is nearly at equilibrium, and we can rely on the Saha equation. However for Hydrogen recombination, the process is mostly out of equilibrium and we need to solve the Boltzmann equation for the recombination/Ionization number density. More details can be found in the founding paper of Peebles and also in Ma & Bertschinger 1995. A comprehensive review is also exposed in Senatore et. al

However these approaches are insufficent as they rely on a simplified two - states atom of hydrogen. They can be tweaked with a fudge factor to describe rather accurately what would be obtained by consider a many-states atom of hydrogen, and this is the approach taken in the widely used algorithm RECFAST. For Helium recombination, we allow the user either the RECFAST algorithm or the Saha equilibrium.

The variable 'x' is used in the computation of recombination since it is the right variable to control the change of behaviour of the integrator. Indeed in CAMB and CMBFAST, the variable is the redihift z, and the user has to ensure that the redshift used in the code for the change of behaviour of the differential equation solver are correct. Indeed the HeliumII ecombination is performed through the Saha equilibrium and then the HeI and H recombination are solved dynamically. There is thus a time where the solver needs to go from one method to the other method.

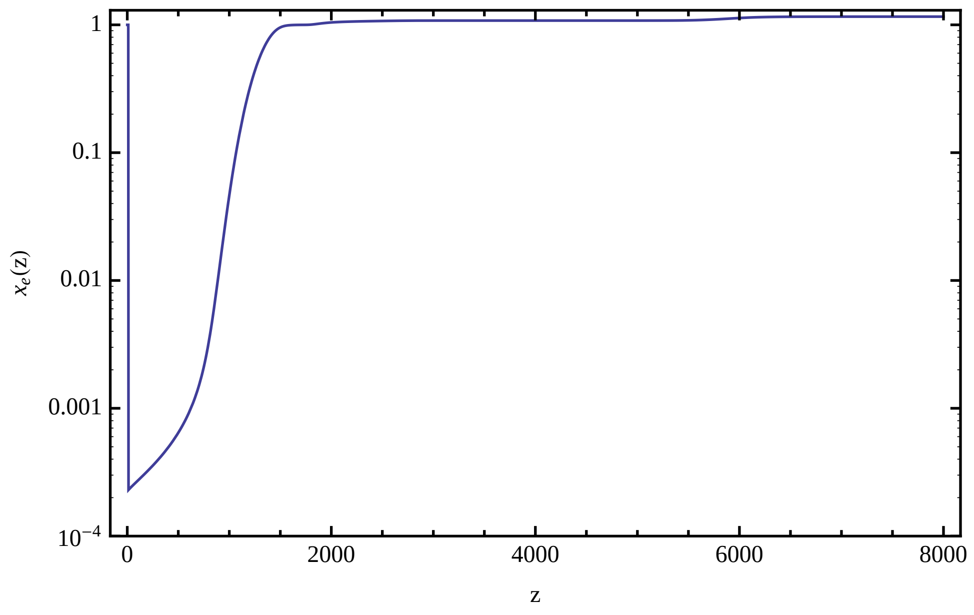
Our implementation is thus more model independent, and this is the reason why we introduced the variable 'x'.

The result of the numerical integration of the Boltzmann equation of recombination leads to the free electrons fraction which is 'xe'. We can see the bump of Helium II-> I around z-5500, then the starting point of of He I->0 at z ~ 2500.

Then the recombination of hydrogen starting around z ~ 1500. Finally the universe is reionized around z~11.

```
In[27]:= LogPlot[xe[CPL][z], {z, 0, 8000},
          PlotRange → {0.0001, 1.3}, Frame → True, FrameLabel → {"z", "xe(z)"}]
```

Out[27]=



This can then be used to infer the interaction rate dτ/dy [name is 'dτyM'], and then by integrating it in y we obtain the optical depth τ as a function of y [name is 'τyM'].

We can finally infer the visbility function $\tau' \; e^{-\tau}$ [name is 'VoftM']. In all these functions, the suffix stands for 'Memorized', which means that values are remembered:

```
In[28]:= Grid[{{"First computation", First@AbsoluteTiming[VoftM[CPL][2]]},
           {"Second computation", First@Timing[VoftM[CPL][2]]}}, Frame → All]
```

Out[28]=

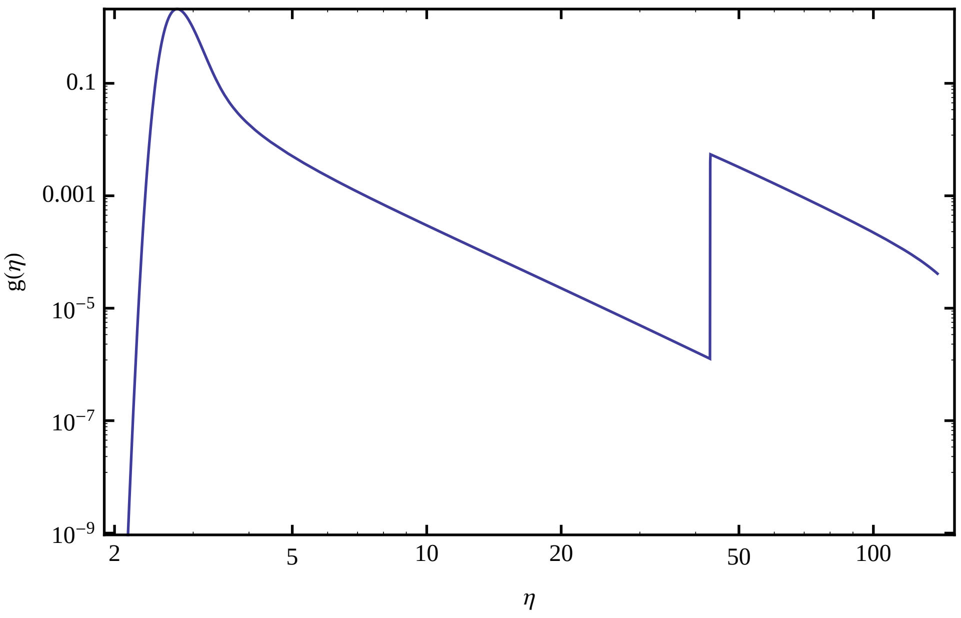
First computation	0.010279
Second computation	0.000387

Note that even when values are memorized,
the access time is still $\sim 10^{-5}$ s. This is the main limitation of *Mathematica* in terms of computation speed ...

The visibility function through the Universe history looks like :

```
In[29]:= LogLogPlot[VoftM[CPL][t], {t, etaOFy[CPL][2], η0[CPL]},
          Frame → True, FrameLabel -> {"η", "g(η)"}, PlotRange → {0, 10}]
```

```
Out[29]=
```



Physical units

In order to convert modes whose value is in units of k_{eq} in Mpc^{-1}, we use the function `RescaleMp` whose syntax is

```
In[30]:= ? RescaleMp
```

```
Number of Mpc in 1/keq for a given
    'Cosmological Parameters List'. Syntax is 'RescaleMp[cpl]'
```

The mode ktest in Mpc^{-1} is thus given by

```
In[31]:= ktest = 5;
         ktest / RescaleMp[CPL]

Out[32]= 0.0483425
```

And the time today can be translated into a distance in Mpc by

```
In[33]:= η0[CPL] RescaleMp[CPL]

Out[33]= 14 393.6
```

First order equations

Equations solver

The system of first order equations is solved by the function ' `Sys1`' for which the syntax is `Sys1[cpl][k]`

In order to obtain information on any command, just type.

```
In[34]:= ?Sys1
```

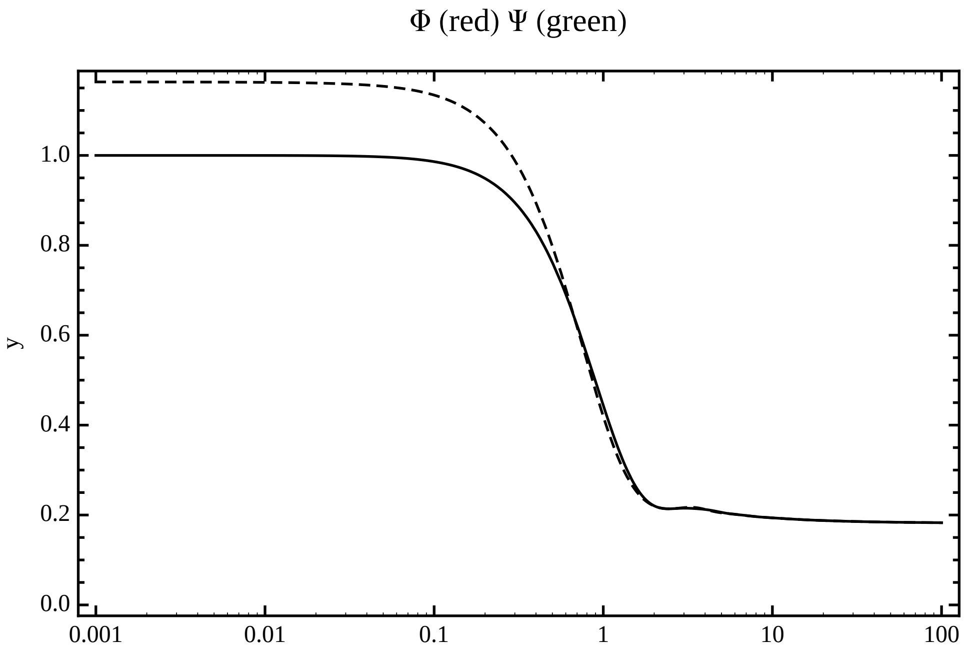
```
First order differential system. Syntax is 'Sys1[cpl][k]'
```

```
In[35]:= First@Timing[Sys1[CPL][0.3]]
         First@Timing[Sys1[CPL][3]]

Out[35]= 9.130481

Out[36]= 1.067867
```

It is automatically called when functions which are solved in the system are asked for somewhere. For instance if we want to plot the transfer function of the gravitational potentials Φ and Ψ for a given mode k, then it will be called and the result obtained would be set in Φ and Ψ. In general functions are calculated only when necessary, and not before.

```
In[37]:= MyLogLinearPlotBandW[{Φ1[CPL][ktest][y], Ψ1[CPL][ktest][y]},
          {y, 0.001, 100}, FrameLabel → {"y"}, PlotLabel → "Φ (red) Ψ (green)"]
```

Out[37]=

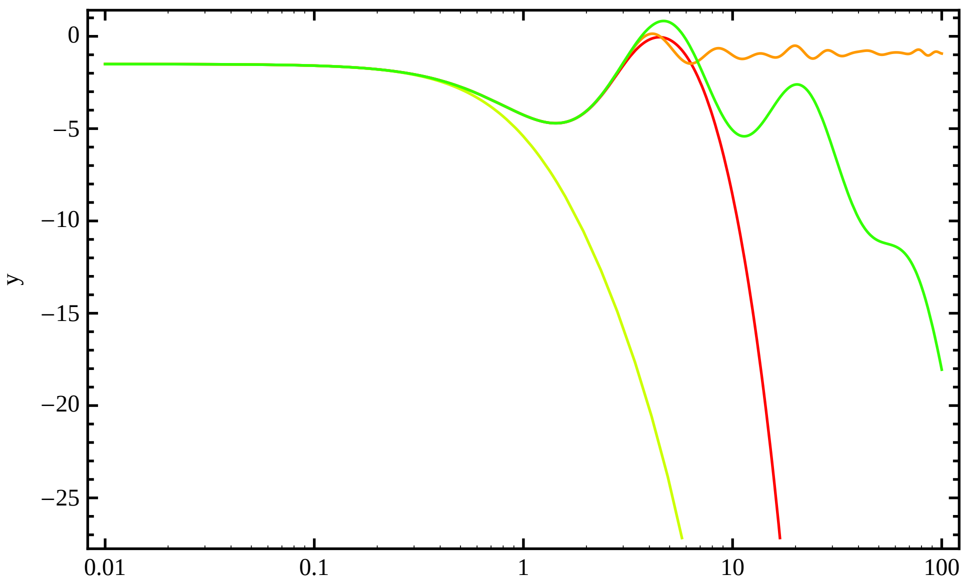


We can also infer the transfer function of the cold dark matter energy density, the radiation energy density, baryon energy density, and the tight-coupled approximation.

```
In[38]:= ktest = 3;
         MyLogLinearPlotColors[
          {B0o1[CPL][ktest][y], 3 / 4 Ro1[CPL][0][ktest][y], C0o1[CPL][ktest][y],
           3 / 4 (1 + 4 / 3 R[CPL][y]) / (1 + R[CPL][y]) F0o1[CPL][ktest][y]}, {y, 0.01, 100},
          FrameLabel → {"y"}, PlotLabel → "δb (red)  3/4δr (green) and δc (blue)"]
```

Out[39]=

δ_b (red) $3/4\delta_r$ (green) and δ_c (blue)

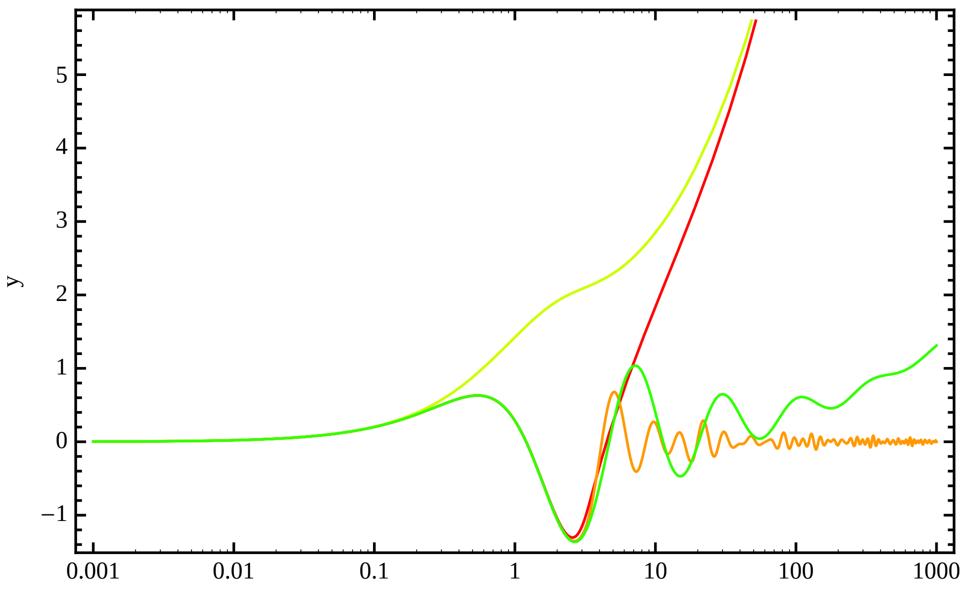


The same can be done for the velocities:

```
In[40]:= MyLogLinearPlotColors[{B1o1[CPL][ktest][y], Ro1[CPL][1][ktest][y] / 4,
           C1o1[CPL][ktest][y], F1o1[CPL][ktest][y]}, {y, 0.001, 1000},
          FrameLabel → {"y"}, PlotLabel → "Vb (red)  Vr (green) and Vc (blue)"]
```

Out[40]=

V_b (red) V_r (green) and V_c (blue)



Memory Management

I did not worry too much about memory management. As mentionned earlier, when you compute things, most of the time they are stored. This is because most computations are time consuming and it can save a lot of time to have values stored. The functions whose stored values are the largest in terms of memory usage, are stored in a list. It is then possible to erase this list, which is associated with a given cosmological parameters list (`cpl`) by typing for instance

```
In[41]:= CleanGarbage[CPL]

           14 026 600 Bytes have been gained.
```

The multipoles C_l

Choosing the effects

A "list of effect" is passed as an argument to all mutlipole-related functions. It is a list of Boolean values which tells whether or not an effect should be considered. The structure is

```
In[42]:= Grid[{{"LSS", "Early effects", "Late effects", "Sachs-Wolfe",
            "Integrated Sach-Wolfe", "Doppler", "Quadrupole (l=2 sources)"}}, Frame → All]
```

Out[42]=

LSS	Early effects	Late effects	Sachs-Wolfe	Integrated Sach-Wolfe	Doppler	Quadrupole (l=2 sources)

The default list is `EL` and has all the effects:

```
In[43]:= EL

Out[43]= {True, True, True, True, True, True, True}
```

By allowing to select only certain effects, we can understand the role played by each one of them.

Spherical Bessel functions

The sources can then be computed out of the transfer functions. By integration over time with the correct spherical Bessel functions, we obtain the multipoles Θ_{lm} and E_{lm} (and B_{lm} for tensor modes) of each Fourier mode.

Then the CMB multipoles C_l are inferred by an integration of the type ' $|\Theta_{\mathrm{lm}}|^2$ x primordial power spectrum' over k.

We are free to choose the effects that we want to include in the resulting CMB multipoles. In that sense CMBquick is a good tool to explore the physical effects.

The list of effects is stored in a list which is a ' Effects list'. It is essentially similar to a 'Cosmological Parameters List'. The typical name is ' `EL`' (for all effects included).

We compute then the multipoles C_l for the list between `lmin` and `lmax`

```
In[44]:= lmin = 2;
         lmax = 2500;
```

The Bessel functions which were stored are then loaded up to a maximum multipole. If the maximum multipole is larger than the maximum multipole up to which the values have been stored, then CMBquick generates the Bessel functions and extends its list of precalculated Bessel functions. This part is time consuming since there is no approximation scheme in the computation of the Bessel function. Since this is to be done only once, it should not be a problem.

If the precomputed Bessel functions are not loaded, CMBquick will compute the necessary Bessel functions, without approximation, and this is very time-consuming (~24 hours on a dual-Core 2.4 GHz CPU).

Note that in version 0.0.3 this should be reduced to a few seconds using a faster and more efficient algorithm and this incovenient will disappear.

You should not forget to include this command in any notebook in which you use CMBquick. Preferably you could put it at the beginning.

```
In[46]:= LoadAndGenerateBesselsBinary[lmax];

           Previously, Bessel functions were computed up to lmax=2500

           No need to compute other Bessel functions
```

This beginning of the list for which we are going to compute the multipoles is

```
In[47]:= Listls = Select[ListlUsedinBesselFunction, # ≤ lmax &]

Out[47]= {2, 3, 4, 6, 8, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140,
          150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 260, 280, 300, 320, 340,
          360, 380, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700,
          725, 750, 775, 800, 825, 850, 875, 900, 950, 1000, 1050, 1100, 1150, 1200,
          1250, 1300, 1350, 1400, 1450, 1500, 1550, 1600, 1650, 1700, 1750, 1800, 1850,
          1900, 1950, 2000, 2050, 2100, 2150, 2200, 2250, 2300, 2350, 2400, 2450, 2500}
```

Precomputing the CMB sources for line of sight integration

We precompute the sources. This means precomputing the Boltzmann hierarchy, possibly with parallelization if the option $ParalellizeBool is set to True (default value).

But also we precompute the source multipoles. We can follow the splitting between the kernels by setting $Verbose=True.

Note that parallelization does not work well for *Mathematica* 7, so I recommend to set here $ParalellizeBool=False in that case.

```
In[48]:= $Verbose = False;(* In case you want a lot
          of useless oupput during intermediary computations *)
         $ParallelizeBool = False; (* Put True if you have enough RAM.*)

In[50]:= PreComputeSourcesMultipoles[CPL, EL, 0] // AbsoluteTiming

           We compute The k-dependent Boltzmann Hierarchy.
             If $ParallelizeBool=True, this is parallelized over CPUs.

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization

Out[50]= {130.166900, Null}

In[53]:=
```

Avoid annoying message for later.

```
In[51]:= ParallelEvaluate[Off[InterpolatingFunction::dmval];];
         Off[InterpolatingFunction::dmval];

In[53]:=
```

We optimize the use of the memory because Parallelization is very Memory consuming.

```
In[53]:= Share[]
         ParallelEvaluate[Share[]]

Out[53]= 42 232

Out[54]= {2 424 440, 2 424 440, 2 424 440, 2 424 440}
```

Unlensed spectra

The function `Cl` computes the Cl' s, the function `ClRed` also mutliplies it by $T_0^2 l\,(l+1)/(2\,\pi)\,10^{12}$. It requires the '`cpl`' (the 'cosmological parameters list'), the '`el`' (the 'effects list"), the method (`FastFullSky` or `FullSky` or one of the three FlatSky methods which are `FlatSky1 FlatSky2` or `FlatSky3`), the correlation (`TT TE` or `EE`), the type of perturbations ('`Scalars`' or '`Tensors`') and the choice of the lensing method (Currently only either `UnLensed` or `Lensed`) to compute the Cl's.

```
In[55]:= ?ClRed
         ?Cl
```

Same thing as Cl but multiplied by $T_0{}^2 l(l+1)/(2\pi)$ and expressed in μK^2.

"C_l computed for the given 'cosmological parameters list', and for a choice of 'effects list' in the CMB, and for a method. \nSyntax is thus Cl[cpl,el,method,correlation,type_of_perturbations,lensingBoolean]. \nThe methods can be 'FastFullSky' (The best one) 'FullSky' 'FlatSky1'(k_{orth} r=l) 'FlatSky2' (k_{orth} r=l+1/2) or 'FlatSky3' (The best of the lowest order flat sky methods) and 'FlatSkyCorrected' (k_{orth} r=$\sqrt{l(l+1)}$). \nFlatSkyCorrected implements the first correction to the flat sky expansion ('FlatSky3' method). \nThe correlation can be TT TE EE (or BB if relevant). The type_of_perturbations can be either 'Scalars' or 'Tensors'. lensingBoolean is either 'Lensed' or 'UnLensed'"

We store and interpolate the unlensed multipoles. We print the time required to understand what limits the computational time. Indeed, in CMBquick most of the time is lost in the integrals over the sources, and not in the numerical resolution of the sources from their dynamical equations. This is because accessing variables in *Mathematica* is always a limiting factor. As a result CMBquick is approximately 10 times slower than CAMB.

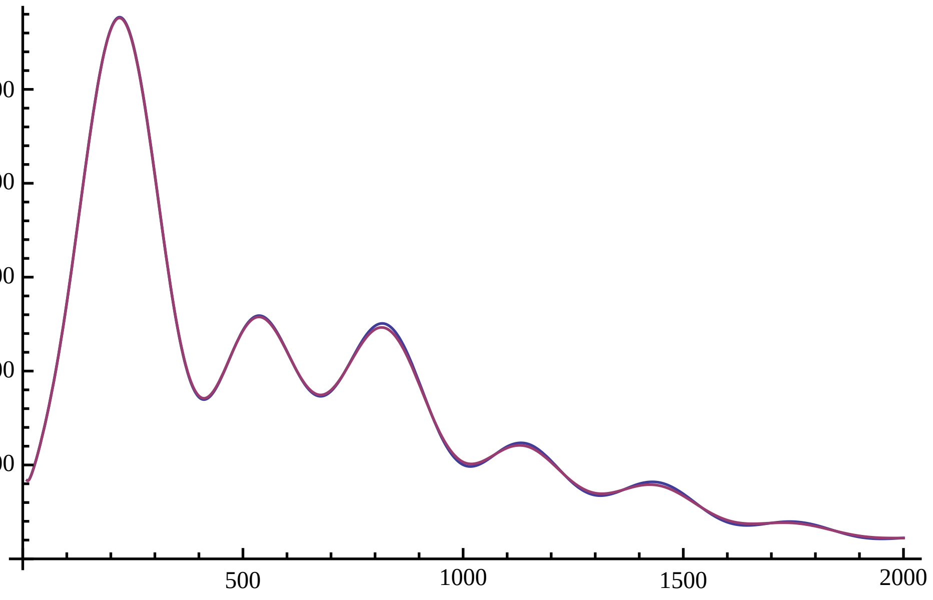
```
In[57]:= First@AbsoluteTiming[ClTEfullI = Interpolation[
             MyParallelize@Table[If[$Verbose, Print["Kernel ", $KernelID, " l= ", l]];
               {l, ClRed[CPL, EL, FastFullSky, TE, Scalars, UnLensed][l]},
               {l, Listls}], Method → "Spline"]]
         First@AbsoluteTiming[
           ClTTfullI = Interpolation[
             MyParallelize@Table[If[$Verbose, Print["Kernel ", $KernelID, " l= ", l]];
               {l, ClRed[CPL, EL, FastFullSky, TT, Scalars, UnLensed][l]},
               {l, Listls}], Method → "Spline"]]
         First@AbsoluteTiming[ClEEfullI = Interpolation[
             MyParallelize@Table[If[$Verbose, Print["Kernel ", $KernelID, " l= ", l]];
               {l, ClRed[CPL, EL, FastFullSky, EE, Scalars, UnLensed][l]},
               {l, Listls}], Method → "Spline"]]

Out[57]= 105.772391

Out[58]= 0.003863

Out[59]= 0.004390
```

Lensed spectra

The computation of the `TT` and `EE` is faster when `Cl` (or `ClRed`) is called with the `TE` correlation, it also computes the `TT` and `EE` for nearly no additional cost.

We can then also store the lensed multipoles. This implements the (simple...) flat sky harmonic method of Hu 2000.

```
In[60]:= (*$Verbose=True;*)
         First@AbsoluteTiming[ClTElensedfullI =
            Interpolation[Table[{l, ClRed[CPL, EL, FastFullSky, TE, Scalars, Lensed][l]},
              {l, Listls}], Method → "Spline"]]
         First@AbsoluteTiming[ClTTlensedfullI = Interpolation[
             Table[{l, ClRed[CPL, EL, FastFullSky, TT, Scalars, Lensed][l]}, {l, Listls}],
             Method → "Spline"]]
         First@AbsoluteTiming[ClEElensedfullI = Interpolation[
             Table[{l, ClRed[CPL, EL, FastFullSky, EE, Scalars, Lensed][l]}, {l, Listls}],
             Method → "Spline"]]

Out[60]= 45.792506

Out[61]= 15.402053

Out[62]= 16.939180

In[63]:= Plot[{ClTTfullI[l], ClTTlensedfullI[l]}, {l, 10, 2000}]

Out[63]=
```



Loading the CAMB-WMAP5 reference files

The output of the high resolution CAMB integration, are stored in two files, one for the unlensed result, and one for the lensed results.

The parameters used are the reference parameters of WMAP5 which are stored in ' CPL'

The general function to load and interpolate the TT correlation in a given CAMB file

```
In[64]:= LoadCAMB[Name_String] := Module[{file, datafile, lmaxfile},
           file = Import[StringJoin[CMBquick`CMBquick1`$CMBquickDataDirectory, Name]];
           datafile[ll_] := file[[ll - 1, 2]];
           lmaxfile = Min[lmax, Length[file]];
           Interpolation[Table[{ll, datafile[ll]}, {ll, lmin, lmaxfile}]]
          ]

In[65]:= ClCamb = LoadCAMB["/CAMB_WMAP_unlensed.dat"];
         ClCambL = LoadCAMB["/CAMB_WMAP_lensed.dat"];
```

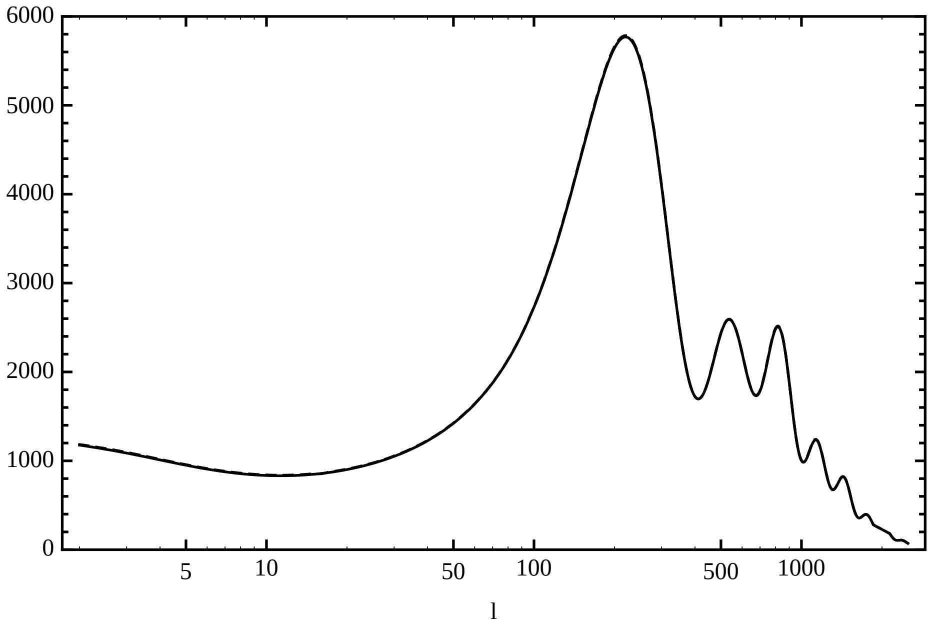
Plots

We plot the out put of CMBquick (continuous line) and CAMB (dashed line)

```
In[67]:= ClTTplot = MyLogLinearPlotBandW[
           {ClTTfullI[l], ClCamb[l]}, {l, lmin, lmax}, FrameLabel → {"l", ""},
           PlotLabel -> "(T0)^2 l(l+1) CTT /(2π)  10^-12 K^2 unlensed. CMBquick vs CAMB",
           PlotRange → {0, 6000}]

         ClTTplotlensed = MyLogLinearPlotBandW[
           {ClTTlensedfullI[l], ClCambL[l]}, {l, lmin, lmax}, FrameLabel → {"l", ""},
           PlotLabel -> "(T0)^2 l(l+1) CTT /(2π)  10^-12 K^2 lensed. CMBquick vs CAMB",
           PlotRange → {0, 6000}]
```

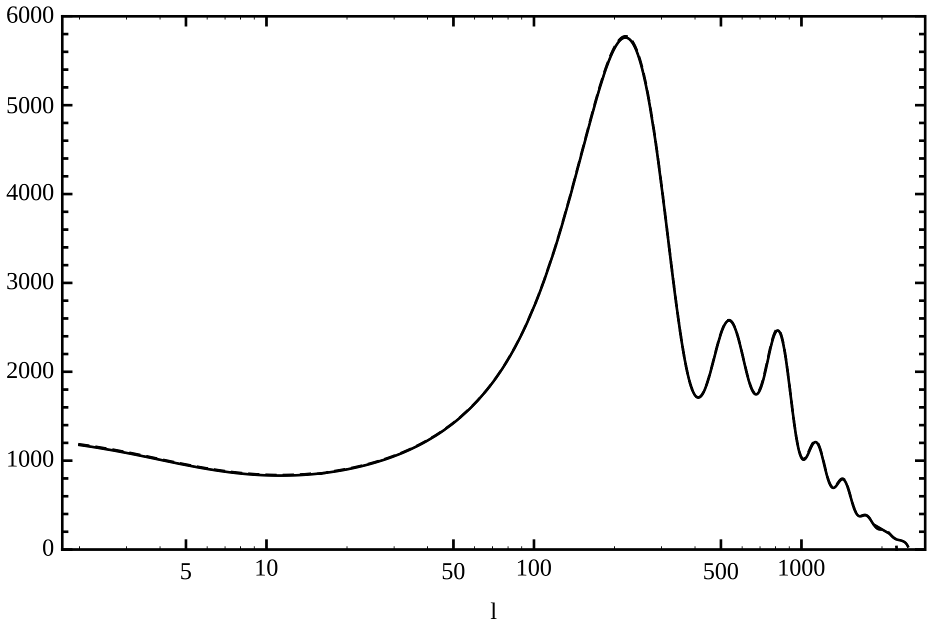
$(T_0)^2 l(l+1)\ C_{TT}\ /(2\pi)\ \ 10^{-12} K^2$ unlensed. CMBquick vs CAMB

Out[67]=



$(T_0)^2 l(l+1)\ C_{TT}\ /(2\pi)\ \ 10^{-12} K^2$ lensed. CMBquick vs CAMB

Out[68]=

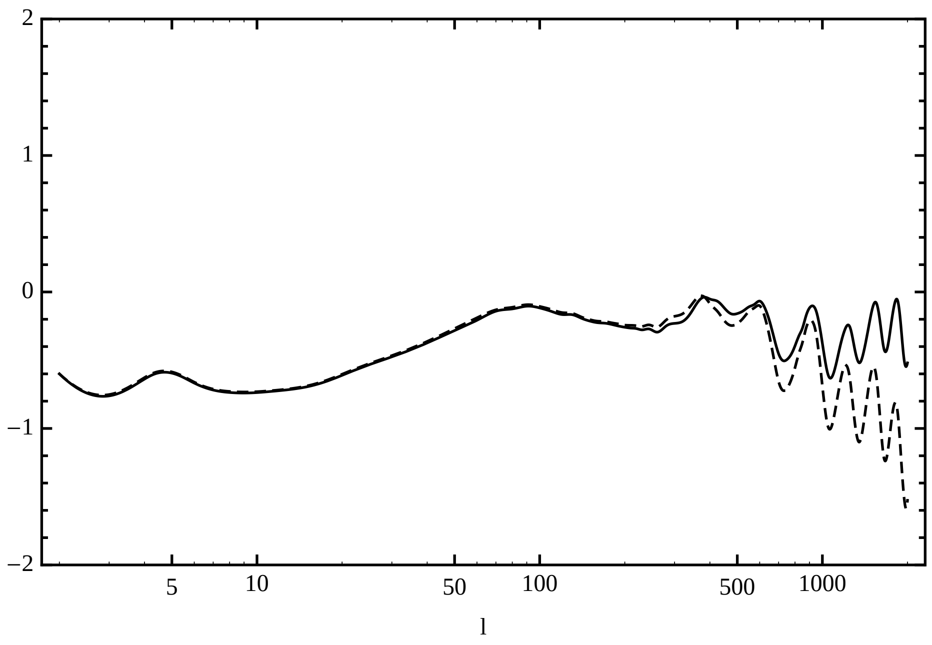


We can check the error between the two packages

```
In[69]:= ClfullErrorI = Interpolation[Table[{l,
              100 * (ClRed[CPL, EL, FastFullSky, TT, Scalars, UnLensed][l] / ClCamb[l] - 1)},
            {l, Listls}], Method → "Spline"];
         ClfulllensedErrorI = Interpolation[Table[
            {l, 100 * (ClRed[CPL, EL, FastFullSky, TT, Scalars, Lensed][l] / ClCambL[l] - 1)},
            {l, Listls}], Method → "Spline"];

         MyLogLinearPlotBandW[{ClfullErrorI[l], ClfulllensedErrorI[l]},
          {l, lmin, 2000}, FrameLabel → {"l", ""},
          PlotLabel -> "Error on C_TT  w.r.t. CAMB, in % for lensed and unlensed.",
          PlotRange → {-2, 2}]
```

Error on C_{TT} w.r.t. CAMB, in % for lensed and unlensed.

Out[71]=

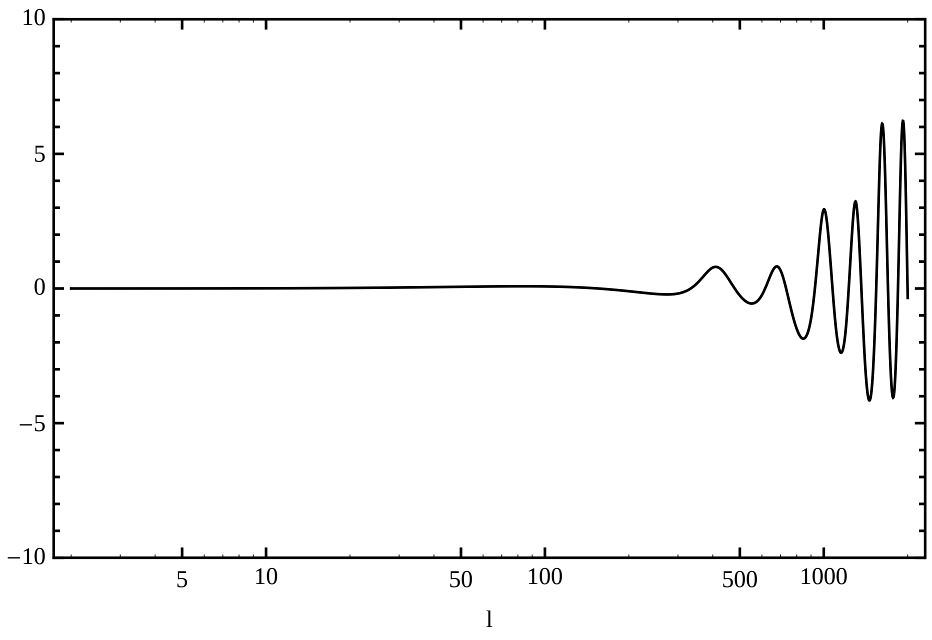


```
In[72]:= (* Jouer sur ksmall et NumberstepsLate. Rest is fine I think. *)
```

And we can also check the effect of lensing

```
In[73]:= ClfullDiffwithlensingI = Interpolation[
            Table[{l, 100 * (ClRed[CPL, EL, FastFullSky, TT, Scalars, Lensed][l] /
                    ClRed[CPL, EL, FastFullSky, TT, Scalars, UnLensed][l] - 1)},
             {l, Listls}], Method → "Spline"];

         MyLogLinearPlotBandW[{ClfullDiffwithlensingI[l]},
          {l, lmin, 2000}, FrameLabel → {"l", ""},
          PlotLabel -> "% of difference between lensed and unlensed C_TT",
          PlotRange → {-10, 10}]

Out[74]=
```

% of difference between lensed and unlensed C_{TT}



Separating the effects

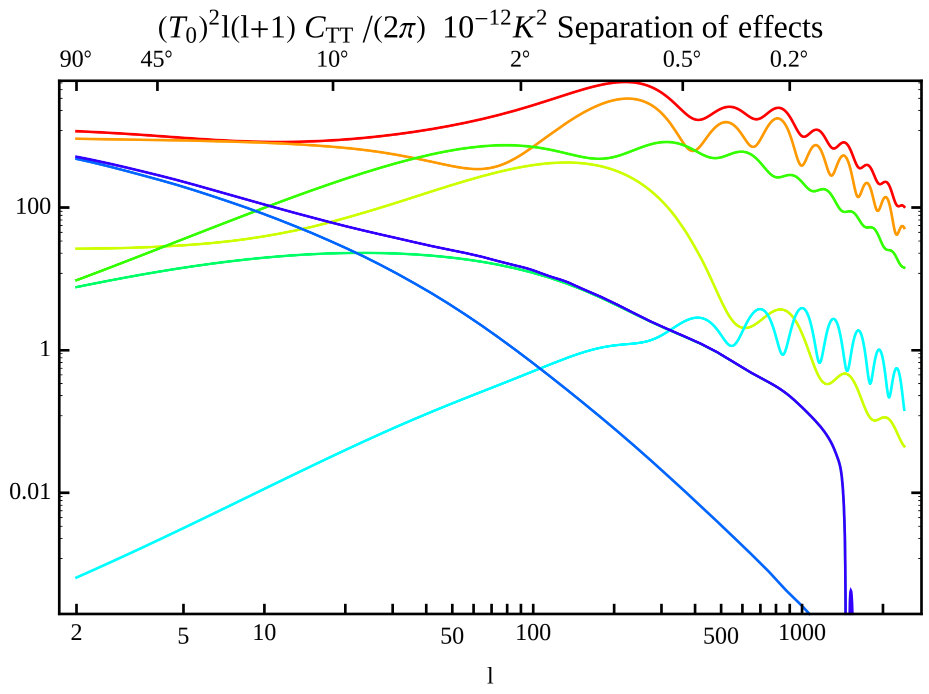
The 'Effects List' allows the user to choose the physical effects at play in the CMB. Of course the real CMB includes all effects, but this enables to understand the order of magnitude of the different contribution.

See the relevant section further in this documentation for details about the "Effects List".

```
In[75]:= ELSW = {True, False, False, True, False, False, False};
         ELEarly = {False, True, False, False, True, False, False};
         ELDoppler = {True, False, False, False, False, True, False};
         ELLate = {False, False, True, False, True, False, False};
         ELQuadrupole = {True, False, False, False, False, False, True};
         ELRei = {False, False, True, False, False, True, False};
         ELAllLate = {False, False, True, True, True, True, True};
```

```
In[82]:= MyLogLogPlotColors[{ClRedI[CPL, EL, FastFullSky, TT, Scalars, UnLensed][l],
           ClRedI[CPL, ELSW, FastFullSky, TT, Scalars, UnLensed][l],
           ClRedI[CPL, ELEarly, FastFullSky, TT, Scalars, UnLensed][l],
           ClRedI[CPL, ELDoppler, FastFullSky, TT, Scalars, UnLensed][l],
           ClRedI[CPL, ELRei, FastFullSky, TT, Scalars, UnLensed][l],
           ClRedI[CPL, ELQuadrupole, FastFullSky, TT, Scalars, UnLensed][l],
           ClRedI[CPL, ELLate, FastFullSky, TT, Scalars, UnLensed][l],
           ClRedI[CPL, ELAllLate, FastFullSky, TT, Scalars, UnLensed][l]},
          {l, lmin, lmax - 100}, FrameLabel → {"l", ""},
          PlotLabel -> "(T0)^2 l(l+1) CTT /(2π)  10^-12 K^2 Separation of effects",
          PlotRange → {0.0002, 6000}, FrameTicks → MyTicks]

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization

           We now compute the Source multipoles (Slm) computation for temperature

           We now compute the Source multipoles (Slm) computation for polarization
```

Out[82]=

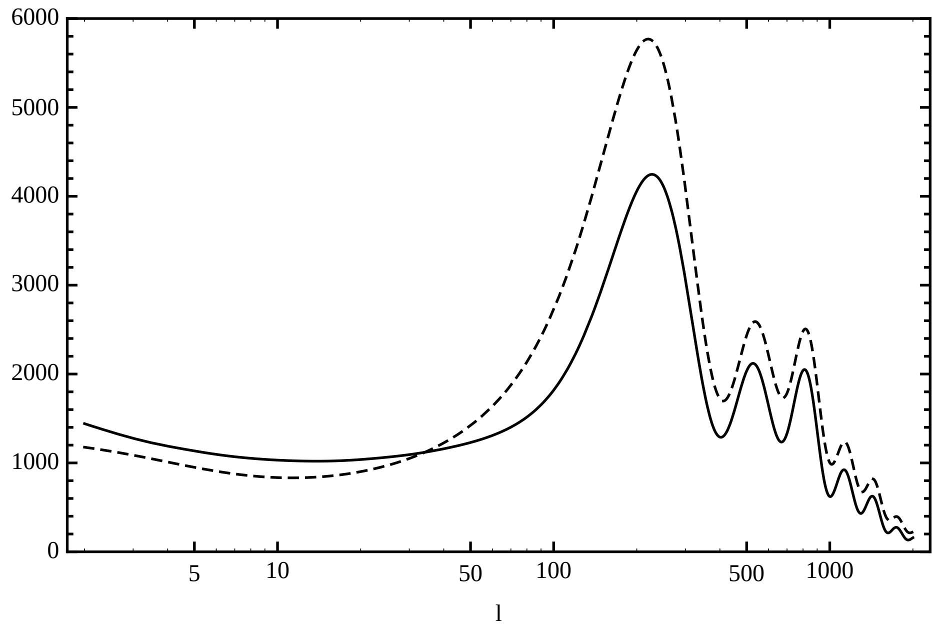


Note that they do not add quadratically. Only total Scalars and total Tensors perturbations add quadratically since there are no cross-corrrelations between them.

```
In[83]:= MyLogLinearPlotBandW[{ClRedI[CPL, ELSW, FastFullSky, TT, Scalars, UnLensed][l] +
            ClRedI[CPL, ELEarly, FastFullSky, TT, Scalars, UnLensed][l] +
            ClRedI[CPL, ELDoppler, FastFullSky, TT, Scalars, UnLensed][l] +
            ClRedI[CPL, ELLate, FastFullSky, TT, Scalars, UnLensed][l] +
            ClRedI[CPL, ELQuadrupole, FastFullSky, TT, Scalars, UnLensed][l],
          ClRedI[CPL, EL, FastFullSky, TT, Scalars, UnLensed][l]},
         {l, lmin, 2000}, FrameLabel → {"l", ""},
         PlotLabel -> "Cl  Individual effects added, and total result",
         PlotRange → {0, 6000}]
```

Out[83]=

C_l Individual effects added, and total result



```
In[84]:= CleanGarbage[CPL];

          75 852 680 Bytes have been gained.
```

Tensor modes

We just have to specify that the type of perturbations is '`Tensors`'.

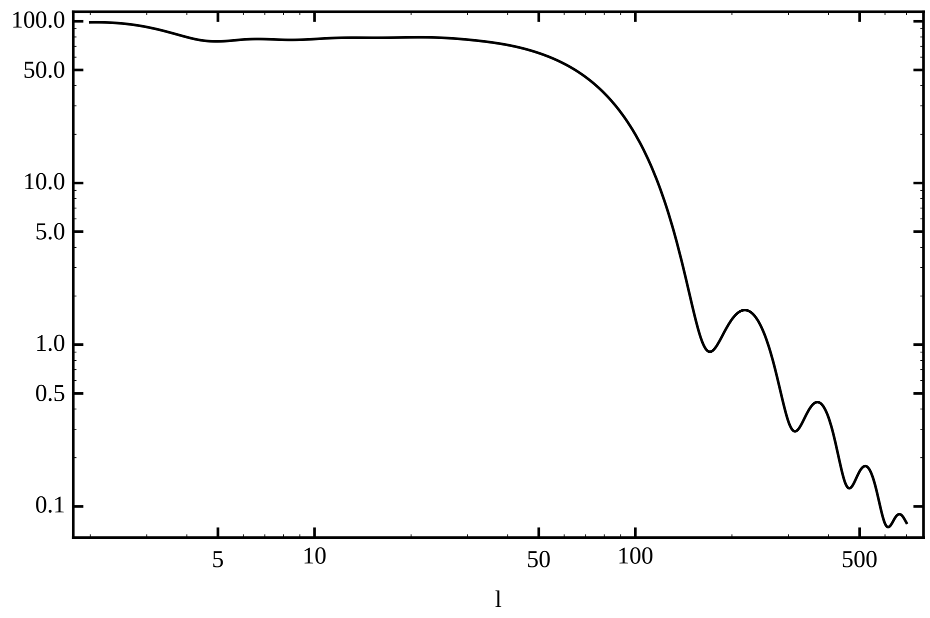
We also have to take a set a Cosmological Parameters which has a non zero tensor to scalar ratio (we take '`CPLGW`'. Type '`Cosmology[CPLGW,PerturbationParameters]`' to obtain the details of this model')

```
In[85]:= MyLogLogPlotBandW[{ ClRedI[CPLGW, EL, FastFullSky, TT, Tensors, UnLensed][l]},
          {l, lmin, Min[lmax, 700]}, FrameLabel → {"l", ""},
          PlotLabel -> "(T0)^2 l(l+1)  CTT /(2π)  10^-12 K^2 unlensed. Tensor modes"(*,
          PlotRange→{0,6000}*)]
```

```
We compute The k-dependent Boltzmann Hierarchy.
  If $ParallelizeBool=True, this is parallelized over CPUs.

We now compute the Source multipoles (Slm) computation for temperature

We now compute the Source multipoles (Slm) computation for polarization
```

$(T_0)^2$l(l+1) C_{TT} /(2π) $10^{-12}K^2$ unlensed. Tensor modes

Out[85]=

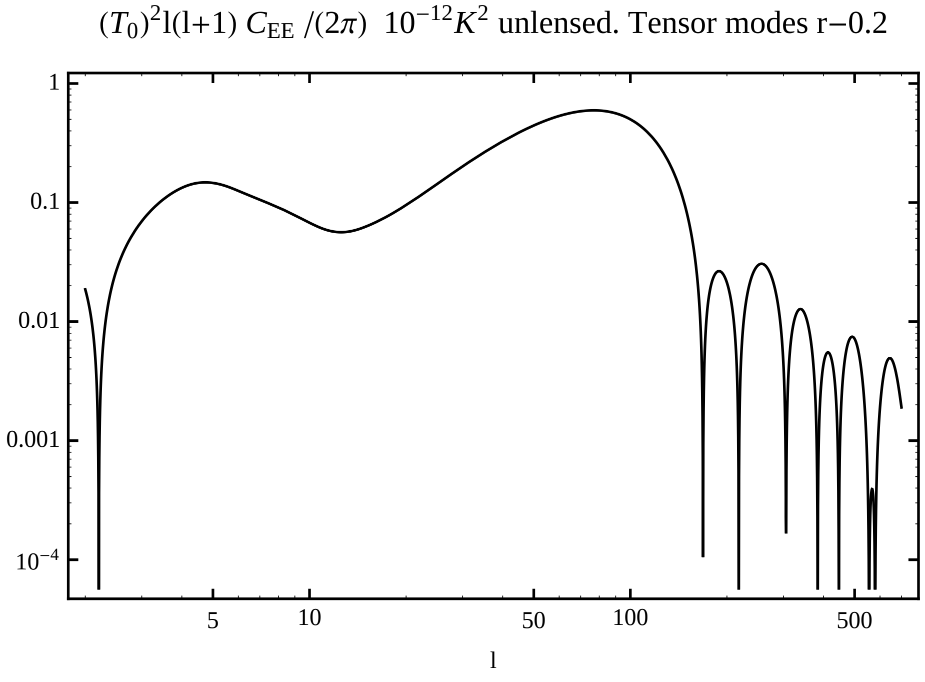


We can even see the bump of reionization at low l' s.

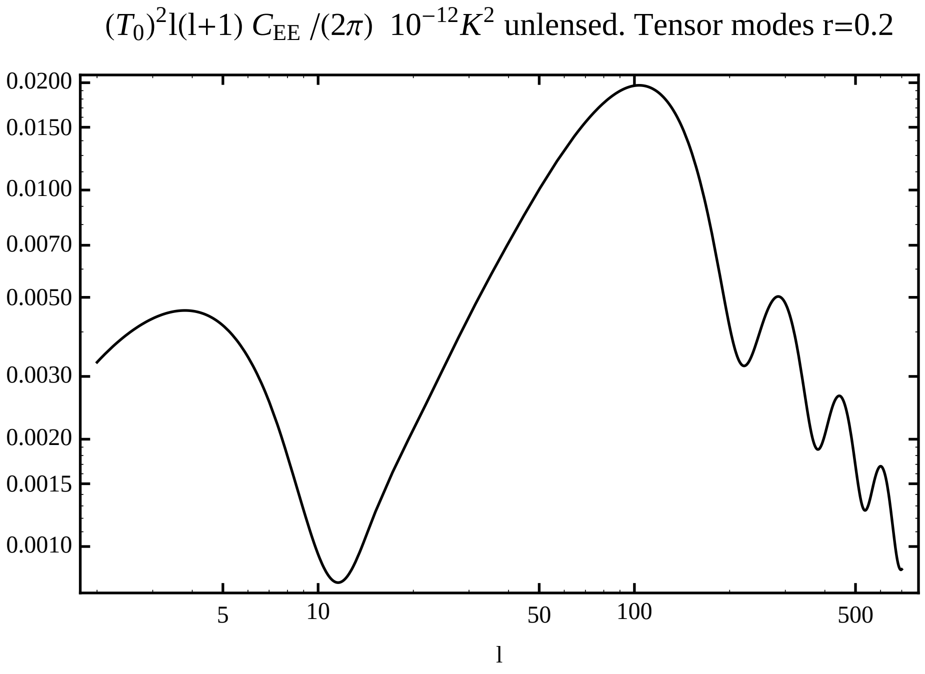
We plot also the TE and EE signals

```
In[86]:= MyLogLogPlotBandW[{ Abs@ClRedI[CPLGW, EL, FastFullSky, TE, Tensors, UnLensed][l]},
          {l, lmin, Min[lmax, 700]}, FrameLabel → {"l", ""},
          PlotLabel -> "(T0)^2 l(l+1) CEE /(2π)  10^-12 K^2 unlensed. Tensor modes r-0.2"(*,
          PlotRange→{0,6000}*)]

         MyLogLogPlotBandW[{ ClRedI[CPLGW, EL, FastFullSky, EE, Tensors, UnLensed][l]},
          {l, lmin, Min[lmax, 700]}, FrameLabel → {"l", ""},
          PlotLabel -> "(T0)^2 l(l+1) CEE /(2π)  10^-12 K^2 unlensed. Tensor modes r=0.2"(*,
          PlotRange→{0,6000}*)]
```

Out[86]=



Out[87]=

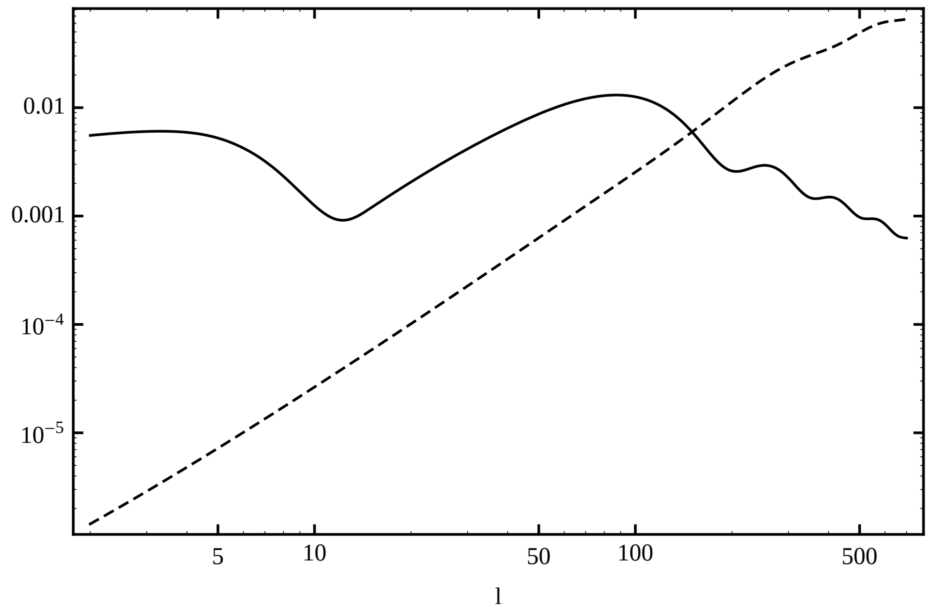


Even more important signe BICEP results in 2014, the BB signal from tensor modes as compared to the one from lensing. Here r=0.2 as found in BICEP.

```
In[88]:= MyLogLogPlotBandW[{ClRedI[CPLGW, EL, FastFullSky, BB, Tensors, UnLensed][l],
           ClRedI[CPL, EL, FastFullSky, BB, Scalars, Lensed][l]},
          {l, lmin, Min[lmax, 700]}, FrameLabel → {"l", ""},
          PlotLabel -> "(T0)^2 l(l+1) C_BB /(2π)  10^-12 K^2  Tensors(r=0.2) and Lensing"(*,
          PlotRange→{0,6000}*)]
```

$(T_0)^2$l(l+1) C_{BB} /(2π) $10^{-12}K^2$ Tensors(r=0.2) and Lensing

Out[88]=



We clean all the values stored about this model since we should not waste memory:

```
In[89]:= CleanGarbage[CPLGW];

           66 529 512 Bytes have been gained.
```

■ Plotting experimental data

We can superimpose the most recent data sets. (Don't blame me if the list is not exhaustive ... things evolve too fast).

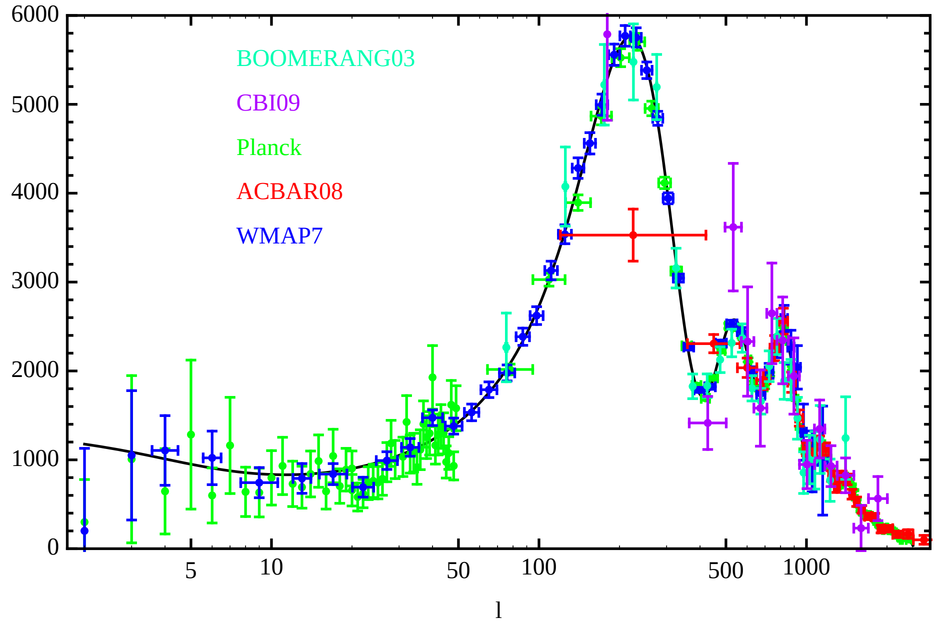
The plots with only data points (and error bars) are stored in the variables 'DataPlotCCaxeXaxeY" where CC is the correlation type (TT TE or EE), and where axeX and axeY are the scaling of these axes (Lin for linear and Log for Log scale).

```
In[90]:= ClTTplotLensed = MyLogLinearPlotBandW[
            {ClTTlensedfullI[l]}, {l, lmin, lmax}, FrameLabel → {"l", ""},
            PlotLabel -> "(T0)^2 l(l+1) C_TT /(2π)  10^-12 K^2 lensed. ", PlotRange → {0, 6000}];
```

```
In[91]:= Show[ClTTplotLensed, DataPlotTTLogLin]
```

Out[91]=

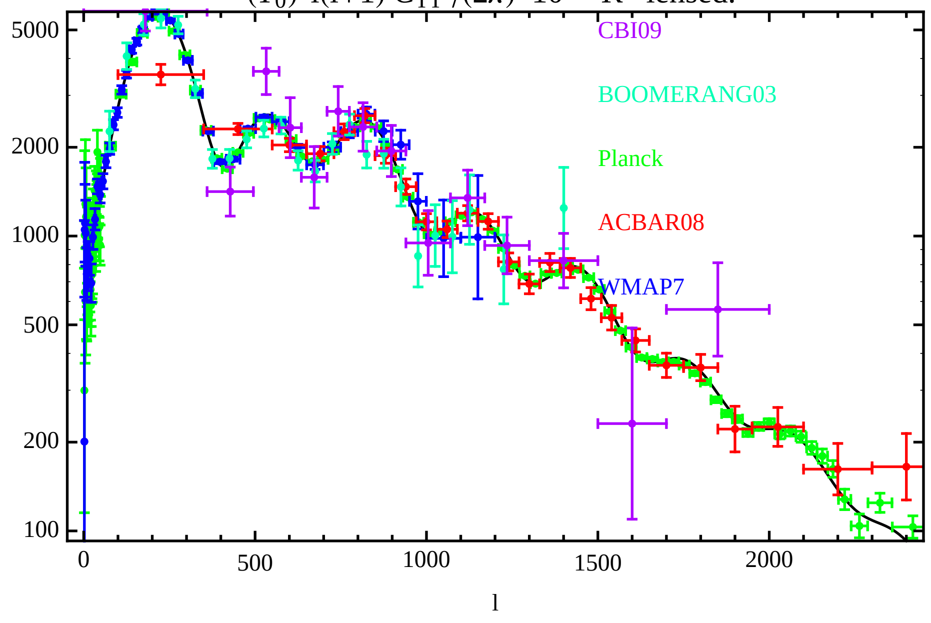
$(T_0)^2 l(l+1)\ C_{TT}\ /(2\pi)\ \ 10^{-12} K^2$ lensed.



```
In[92]:= ClTTplotLensed = MyLogPlotBandW[
            {ClTTlensedfullI[l]}, {l, lmin, lmax - 100}, FrameLabel → {"l", ""},
            PlotLabel -> "(T₀)²l(l+1) C_TT /(2π)  10⁻¹²K² lensed. ", PlotRange → {0, 6000}];
         Show[ClTTplotLensed, DataPlotTTLinLog]
```

Out[93]=

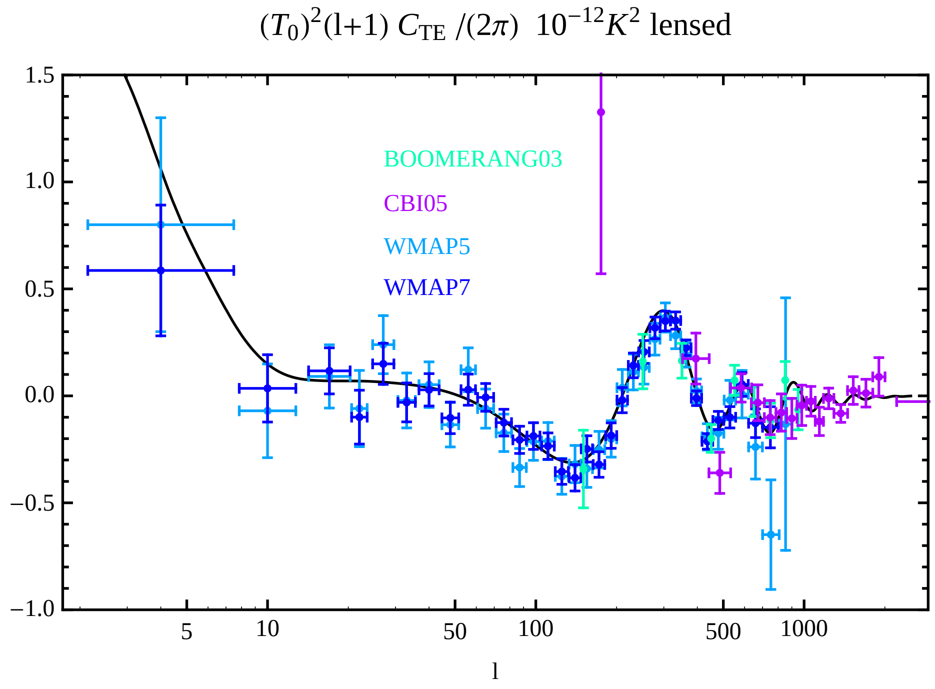
$(T_0)^2 l(l+1)\ C_{TT}\ /(2\pi)\ \ 10^{-12} K^2$ lensed.



The same thing with TE and EE correlations

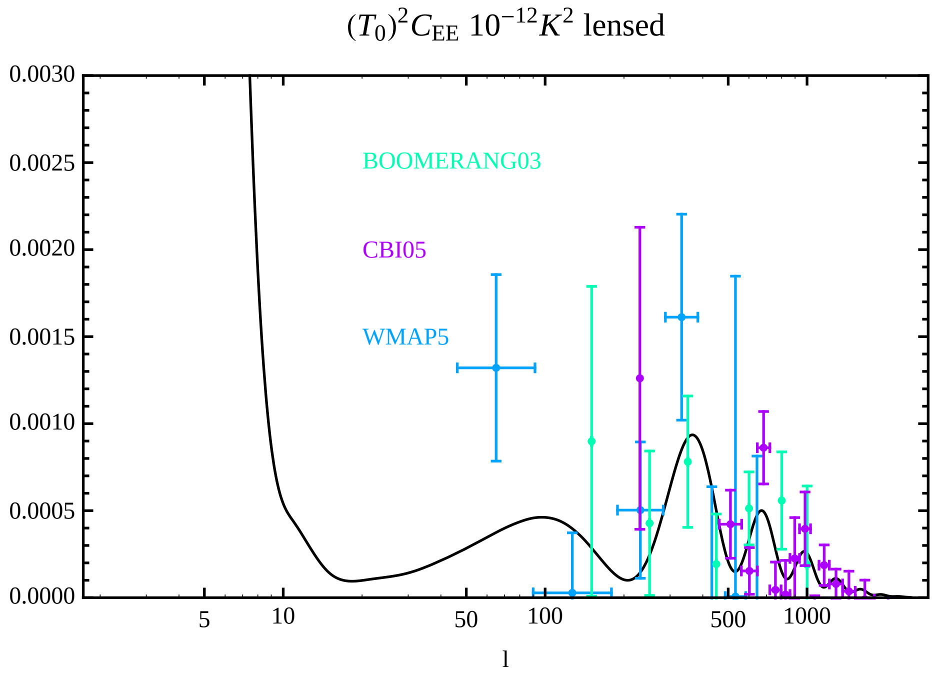
```
In[94]:= ClTEplot = MyLogLinearPlotBandW[
           {ClTElensedfullI[l] / l}, {l, lmin, lmax}, FrameLabel → {"l", ""},
           PlotLabel -> "(T0)^2 (l+1)  CTE  /(2π)   10^-12 K^2  lensed", PlotRange → {-1, 1.5}];

         Show[ClTEplot, DataPlotTELogLin]
```

Out[95]=



```
In[96]:= ClEEplot = MyLogLinearPlotBandW[
           {ClEElensedfullI[l] / l / (l + 1) 2 Pi}, {l, lmin, lmax}, FrameLabel → {"l", ""},
           PlotLabel -> "(T0)^2 CEE  10^-12 K^2  lensed", PlotRange → {0, 0.003}];

         Show[ClEEplot, DataPlotEELogLin]
```

Out[97]=



■ The power spectrum

The function PΦI interpolates the power spectrum of Φ
at a given y. (we recall that $y = a/a_{eq}$). The syntax is thus `PΦI[cpl][y][k]`.
Note that it is defined by $< \Phi(k)\,\Phi(p) > \; = \delta(k+p)\, 2\pi^2\, P(k)/k^3$

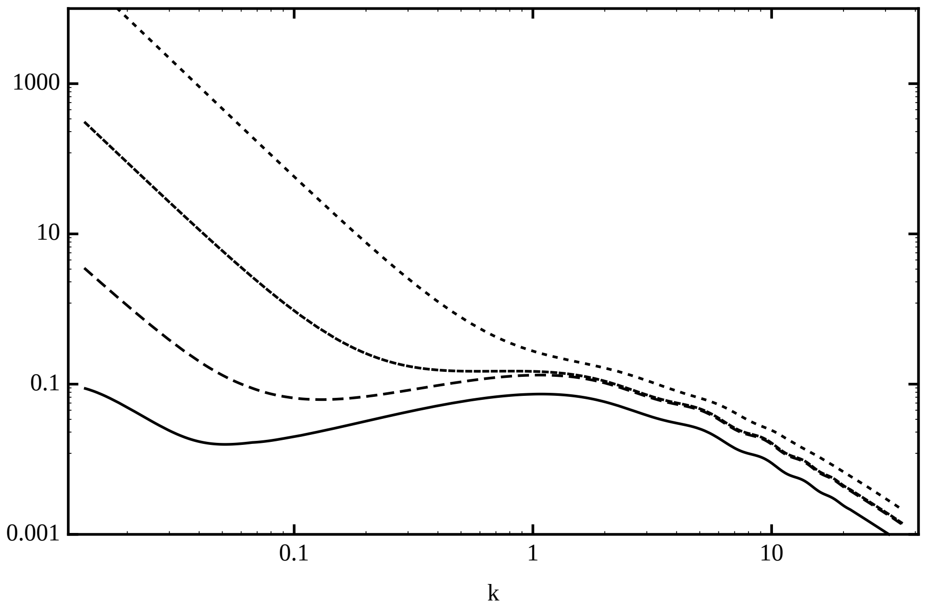
The function PDI does the same thing for the matter density power spectrum.
We can thus follow the evolution in time of the spectra

The matter energy density δ power spectra as a function of time

```
In[98]:= MyLogLogPlotBandW[
          {PDI[CPL][y0[CPL]][k] 2 Pi^2 / k^3, PDI[CPL][y0[CPL] / 10][k] 2 Pi^2 / k^3 10^2,
           PDI[CPL][y0[CPL] / 100][k] 2 Pi^2 / k^3 100^2,
           PDI[CPL][y0[CPL] / 1000][k] 2 Pi^2 / k^3 1000^2},
          {k, kmin, kmax}, FrameLabel → {"k", ""},
          PlotLabel → "(y0/y)^2Pδ(y,k)   for   y=y0,y0/10,y0/100,y0/1000",
          PlotRange -> {0.001, 10000}]

           We compute The k-dependent Boltzmann Hierarchy.
             If $ParallelizeBool=True, this is parallelized over CPUs.
```

$(y_0/y)^2 P_\delta(\mathrm{y,k})$ for $y=y_0, y_0/10, y_0/100, y_0/1000$

Out[98]=

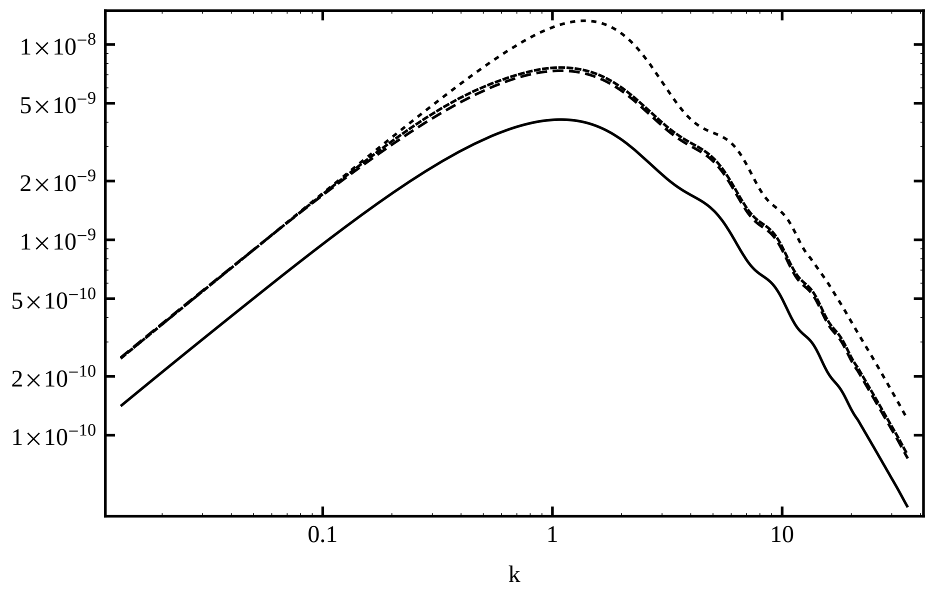


The potential Φ power spectra as a function of time

```
In[99]:= MyLogLogPlotBandW[
           {PΦI[CPL][y0[CPL]][k] 2 Pi^2 k, PΦI[CPL][y0[CPL] / 10][k] 2 Pi^2 k,
            PΦI[CPL][y0[CPL] / 100][k] 2 Pi^2 k, PΦI[CPL][y0[CPL] / 1000][k] 2 Pi^2 k},
           {k, kmin, kmax}, FrameLabel → {"k", ""},
           PlotLabel → " k^4 P_Φ(y,k)    for    y=y_0,y_0/10,y_0/100,y_0/1000"]
```

$k^4\ P_\Phi$(y,k) for y=y_0,y_0/10,y_0/100,y_0/1000

Out[99]=



```
In[100]:=
     CleanGarbage[CPL];

        2 095 392 Bytes have been gained.
```

The power spectrum of Number counts

We get more information about the power spectra for number counts

```
In[101]:=
     ? PΔn*
```

▼ CMBquick`CMBquick1`

PΔnI	PΔnNewtI	PΔnSyncI

The default list of parameters for number counts is

```
In[102]:=
      CPLNC

Out[102]=
     {1., 0.5, 0.05, 0}
```

We define another one

```
In[103]:=
      CPLNC2 = {2, 1, 0.1, 0};
```

Which means bias = 1. $z_{\text{middle}} = 0.5$ σ_z=0.05 and $f_{\text{NL}} = 0$.

The power spectrum of b δ_c where b is the bias is given by P△nNewtI. The integration is performed from the time of reionization so we cheat with this parameter to make sure it is not too far away.

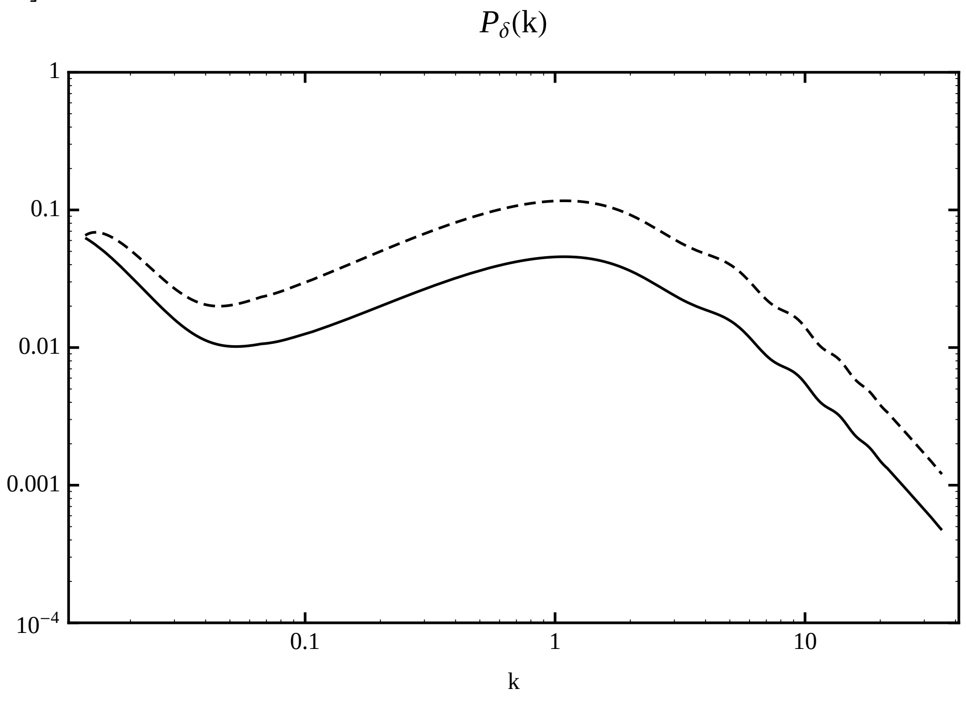
```
In[104]:=
      CPLDn = CPL;
      CPLDn[[8]] = 0.0165;
```

With this cosmology reionizationis around z = 3 so this is not too far. With this cosmology now we can plot the spectra on 3D slices.

```
In[106]:=
      MyLogLogPlotBandW[
       {P△nNewtI[CPLDn, CPLNC][k] 2 Pi^2 / k^3, P△nNewtI[CPLDn, CPLNC2][k] 2 Pi^2 / k^3},
       {k, kmin, kmax}, FrameLabel → {"k", ""},
       PlotLabel → "Pδ(k)", PlotRange -> {0.0001, 1}]

          We compute The k-dependent Boltzmann Hierarchy.
            If $ParallelizeBool=True, this is parallelized over CPUs.
```

The power spectrum of ' b δ_c + 3 H / k (b − 1) V_c ' where b is the bias is given by P△nI

```
In[100]:=
      MyLogLogPlotBandW[
       {P△nI[CPLDn, CPLNC][k] 2 Pi^2 / k^3, P△nI[CPLDn, CPLNC2][k] 2 Pi^2 / k^3},
       {k, kmin, kmax}, FrameLabel → {"k", ""},
       PlotLabel → "Pδ(k)", PlotRange -> {0.0001, 1}]

Out[100]=
```



The related Cl' s taking into account all GR corrections can then be computed (see arXiv:1106.3999).

Indeed we only observe Cl's so this is much more related to observables than the 3D spectra.

```
In[101]:=
      ? ClDD
```

```
Correlation multipole for Cluster number counts.
     Syntax is ClDD[cpl,elnc,cplnc][l]. See also arXiv:1106.3999.
```

Here ELNC is the list of GR effect to be included. And the default contains all effect (f_{NL} contribution, redshift space distortion, magnification effect, redshift effect, distance effect, geometrical effects, and two gauge effects from the definition of the bias).

```
In[102]:=
      ELNC

Out[102]=
      {True, True, True, True, True, True, True, True}
```

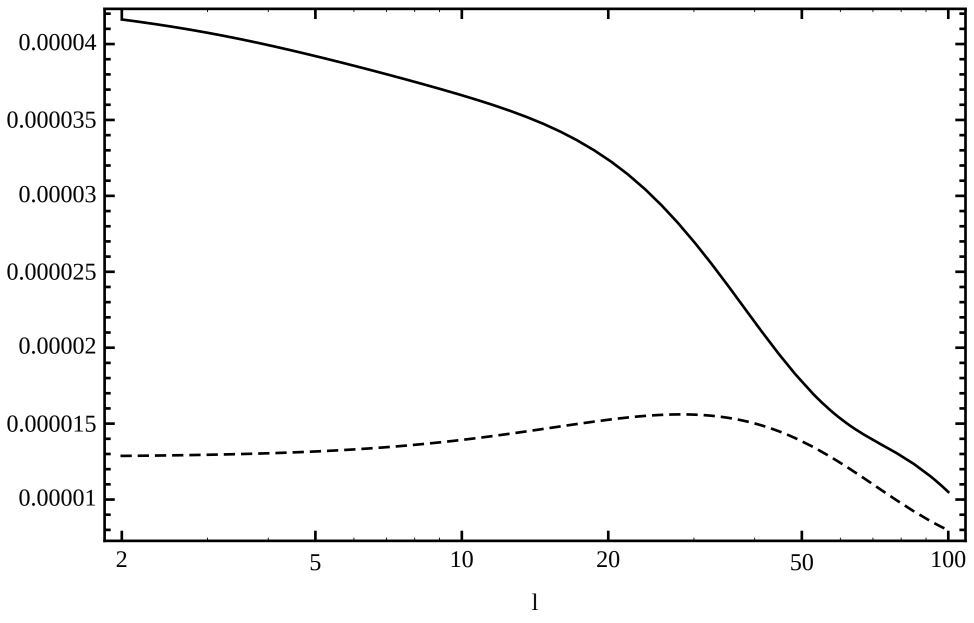
We then compute the Cl's for number counts in two different cases (galaxies are located at two different redshift positions)

```
In[103]:=
      PreComputeSourceskspace[CPLDn, 0];
      (* We parallelize the Boltzmann hierarchy for the Fourier space sampling*)

In[104]:=
      ClDDI = Interpolation[MyParallelize@Table[{l, ClDD[CPLDn, ELNC, CPLNC][l]},
           {l, Select[Listls, # ≤ 100 &]}], Method → "Spline"]
      ClDDI2 = Interpolation[MyParallelize@Table[{l, ClDD[CPLDn, ELNC, CPLNC2][l]},
           {l, Select[Listls, # ≤ 100 &]}], Method → "Spline"]

         We proceed using parallelization. Consider setting
           $ParallelizeBool=False if your system does not have enough memory.
          If your system freezes because of Memory Shortage,
           consider evaluating 'CloseKernels[]'.

Out[104]=
      InterpolatingFunction[{{2., 100.}}, <>]

Out[105]=
      InterpolatingFunction[{{2., 100.}}, <>]
```

```
In[106]:=
      MyLogLinearPlotBandW[{ClDDI[l], ClDDI2[l]}, {l, lmin, 100}, FrameLabel → {"l", ""},
       PlotLabel -> "Cl for Number counts. GR effects included."]
```

```
Out[106]=
```

C_l for Number counts. GR effects included.



We can also test the cross - correlation between CMB (late ISW and reionization only) and galaxy number counts.

```
In[107]:=
      ?ClγD
```

Cross−Correlation multipole between CMB (Only Late ISW and reionization) with cluster number counts. Syntax is ClγD[cpl,elnc,cplnc][l], where elnc and cplnc are the list of effects and the parameters list for galaxy number counts.

In a first case we will consider the redshift-space distortion and in the second case we will ignore only that effect, and we then compare the curves.

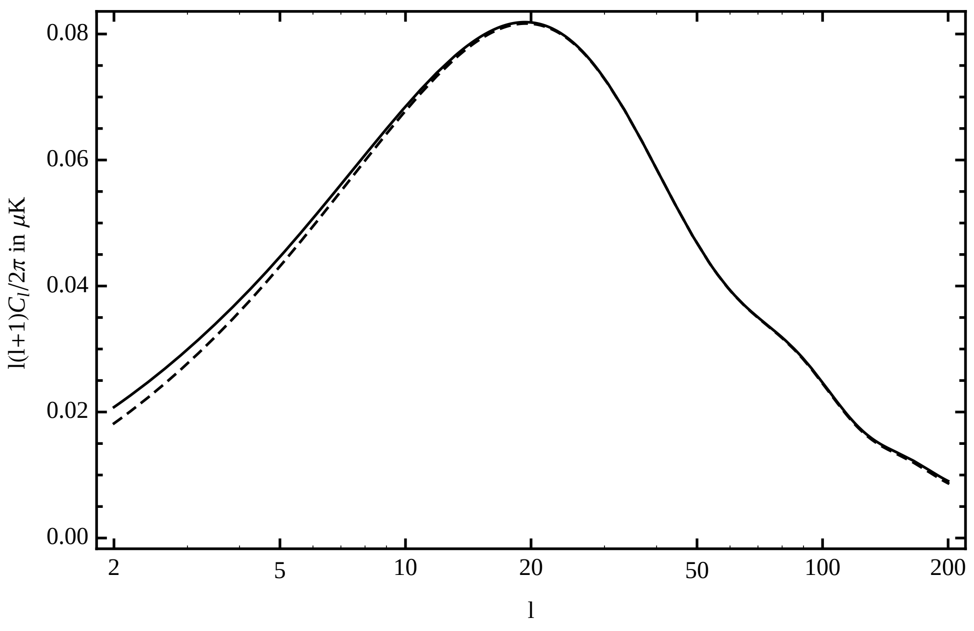
```
In[108]:=
      ELNCnoRSD = {True, False, True, True, True, True, True, True};
```

```
In[109]:=
      (* We precompute the sources for CMB temperature with
       parallelization. Only the Late time effects are necessary.*)
      (* For the moment we do not precompute the sources for number counts *)
      PreComputeSourcesMultipoles[CPLDn, ELLate, 0] // AbsoluteTiming

         We now compute the Source multipoles (Slm) computation for temperature

         We now compute the Source multipoles (Slm) computation for polarization
```

```
Out[109]=
      {15.618506, Null}
```

```
In[110]:=
     ClγDI = Interpolation[
       MyParallelize@Table[{l, l (l + 1) / (2 Pi) * 2.73 * 10^6 ClγD[CPLDn, ELNC, CPLNC][l]},
         {l, Select[Listls, # ≤ 200 &]}], Method → "Spline"]
     ClγDI2 = Interpolation[MyParallelize@
        Table[{l, l (l + 1) / (2 Pi) * 2.73 * 10^6 ClγD[CPLDn, ELNCnoRSD, CPLNC][l]},
         {l, Select[Listls, # ≤ 200 &]}], Method → "Spline"]

        We proceed using parallelization. Consider setting
          $ParallelizeBool=False if your system does not have enough memory.
         If your system freezes because of Memory Shortage,
          consider evaluating 'CloseKernels[]'.

Out[110]=
     InterpolatingFunction[{{2., 200.}}, <>]

Out[111]=
     InterpolatingFunction[{{2., 200.}}, <>]

In[112]:=
     MyLogLinearPlotBandW[{ClγDI[l], ClγDI2[l]},
      {l, lmin, 200}, FrameLabel → {"l", "l(l+1)Cl/2π in μK "},
      PlotLabel -> "(CMB x Number counts). With and without RSD."]

Out[112]=
```

(CMB x Number counts). With and without RSD.



```
In[113]:=
     CleanGarbage[CPLDn]

        155 262 848 Bytes have been gained.

(kernel 74)  Kernel 74   86 926 432 Bytes have been gained.

(kernel 73)  Kernel 73   86 877 952 Bytes have been gained.

(kernel 75)  Kernel 75   86 861 120 Bytes have been gained.

(kernel 76)  Kernel 76   87 154 000 Bytes have been gained.
```

Obtaining information

You can obtain an interactive list of all functions publicly defined in CMBquick1 with the command `?"CMBquick`CMBquick1`*"`

You have to click on the name within the table below to obtain information on each variable.

```
In[114]:=
      ?"CMBquick`CMBquick1`*"
```

▼ CMBquick`CMBquick1`

Accuracyet⋱ alimit	deltXI	jl	R1tco1	y0	Ωνm0
Adiabatic	Disclaimer	jllpml	RECFAST	yBLSS	$Accuracy1
AgeUniver⋱ se	Dist	kmax	Rei	yBrei	$Adaptativ⋱ ePrecisio⋱ n1
As2WMAP	DistLSS	kmin	Reionizatio⋱ nFractio⋱ nWMAP	ye	$Approxim⋱ ationLate⋱ Time
B0o1	dPNmo1	kn	RescaleMp	yEearly	$Boundst
B0tc2o1	drhoNmo1	koffset	Ro1	yELSS	$Boundsta⋱ ndstep
B0tco1	dVNmo1	korth	SAHA	yErei	$CMBquic⋱ kDataDir⋱ ectory
B1o1	dηearly	ksmall	SamplePoi⋱ nts	yHe	$CMBquic⋱ kDirector⋱ y
B1tc2o1	dηlate	Late	SamplePoi⋱ ntsLate	yi1	$CMBquic⋱ kDocDire⋱ ctory
B1tco1	dηLSS	Late$	SamplePoi⋱ ntsRei	yLSS	$Doppler
Backgroun⋱ dParame⋱ ters	dηrei	Lensed	Scalars	yOFeta	$DopplerB⋱ ool
BB	Early	Linear	Simpson	yOFetaM	$earlyISW
Bispectrum	Early$	ListK	SizeStepk	yOFx	$ExistFile
bL	EE	ListlUsedin⋱ BesselF⋱ unction	Slm	yOFz	$Integrate⋱ TC
bL1L2L3	EL	Listt	SlmA	zLSSInsta⋱ ntaneous	$Integratio⋱ nMethod
bLLL	ELLate	LoadAndG⋱ enerateB⋱ esselsBi⋱ nary	Slml	zOFeta	$Integratio⋱ nPoints1
blml	Elm	lofangle	Slmkl	zOFy	$interpjlMe⋱ thod
bNL	elml	LSS	SpectralIn⋱ dex	ζ1	$ISWBool
bPartielL1⋱ L2L3	ELNC	LSS$	SpectrumA⋱ mplitude	η0	$Kdelk

C0o1	Eo1	MethodRecombinationHe	Stc2o1	ηBLSS	$lateISW
C1o1	etaOFy	MyParallelize	Stco1	ηBrei	$ListLK
CDMIsocurvature	etaOFyM	Myprecision	SubListlUsedinBesselFunction	ηEearly	$Lkdek
Cheatwcs2	etaOFz	MyTicks	Sys1	ηELSS	$Lkmax
Cl	F0o1	MyUnitStep	Sys1Early	ηErei	$Lkmin
ClDD	F0tc2o1	Nk	Sys1Late	ηLSS	$LmaxBessels
CleanGarbage	F0tco1	Nneutrinos	Sys1M2	Θlm	$LoadBeginningStoreFile
Cll	F1o1	No1	SysTC1	σ8	$LoadTabs
CLl	F1tc2o1	NoReionization	T0	τpyM	$LoadTabsAndPreambule
ClRed	F1tco1	Npointbispectre	T0WMAP	τpyMnoRei	$LSSDirac
ClRedl	FastFullSky	Nstepkflat	T0$	τrei	$MakeName
ClγD	FlatSky1	nsWMAP	TE	τreiWMAP	$MyIntegral
CMBquick1	FlatSky2	NνWMAP	Tensors	τrei$	$MyIntegral2D
ConstantTransferFunction	FlatSky3	PDI	ThinShellCorrected	τyM	$MyIntegral2DN
Cosmology	FlatSkyCorrected	PerturbationParameters	ThinShellCorrected1surR2	Φ1	$MyIntegralN
CPL	FlatSkyFast2	PhotonsCoupled	ThinShellCorrectedDP	Φ1l	$OR
CPL7	Fracbtom	PiNmo1	ThinShellCorrectedDws	Φp1	$ParallelizeBool
CPLGW	Fracγtor	PiNmo1$	To1	Φp1A	$Pos
CPLNC	FudgeH	PreComputeSourceskspace	ToBeDone	Φp1l	$PrecisionLensing
CPLNoRei	FudgeHe	PreComputeSourcesMultipoles	Tpo1	Ψ1	$Quadrupole
CPLSI	FullSky	ProjFunction	Transfer1l	Ψp1	$QuasiInstantaneousLSS
	GetFileFromCMBquickWeb				$RefineFac

CPLν	uickWeb⋱site	PSΦ	Trapeze	Ωb	$RefineEar⋱ly
cs2	GetjlsFile	PSΦC	TT	Ωb0	$Reionizati⋱onModel
cs2tot	gz	PSΦk2	UnLensed	Ωb0h2WM⋱AP	$Simpson
CteMagnet⋱ic	H	PΔnl	UnloadBes⋱sels	Ωc	$Steplk
Cubic	Hcosmicy	PΔnNewtl	VeryFastF⋱ullSky	Ωc0	$Stiffness⋱Switchin⋱g
DataPlotE⋱ELinLin	Hermit	PΔnSyncl	VoftM	Ωm	$SW
DataPlotE⋱ELogLin	History	PΦl	Vtc2o1	Ωm0	$Trapeze
DataPlotT⋱ELinLin	HprsurH2	Ql	Vtco1	Ωm0h2WM⋱AP	$Verbose
DataPlotT⋱ELogLin	hred	Qll	w	Ωr	$Version
DataPlotT⋱TLinLin	Hub	R	wtot	Ωr0	$WidthPeri⋱od
DataPlotT⋱TLogLin	hWMAP	R0tc2o1	xe	ΩΛ	
DataPlotT⋱TLogLog	InitialCondi⋱tions	R0tco1	xeH	ΩΛ0	
DeltX	Instantane⋱ous	R1tc2o1	xOFy	Ωνm	

CMBquick1 in details

■ History

You can check the history of the different versions by typing :

```
In[115]:=
     History[]

            -0.0.3   (2012) Parallelization
          implemented. Works better with at least Mathematica 8.0
                    -Type 'ToBeDone[]' to have an idea of
          what I have in mind for the next improvements

            -0.0.2  (05/02/2010)
                    -Tensor modes have been implemented. The
          Cosmological Parameters List (cpl) has now two extra parameters,
                     which are the tensor to scalar
          ratio, and the tensor spectral index
                     (n_T=1 for scale-invariant power-spectrum).
                    -Massive neutrinos are now allowed, for scalar-type
          perturbations (they are ignored for tensor-type perturbations).
                     It is using the brute force method of sampling on a grid the
          possible momenta q of the neutrinos distribution. Very time-consuming.
                     l_max for the massive neutrinos hierarchy
          is 8. It is enough for CMB, but corresponds to the
          low resolution of CAMB for the matter power spectrum
                     and thus is inadequqte for the precise
          computation of σ8, though it goves a rather good estimate.
                    -isocurvature initial conditions are allowed.
          Very basic implementation with just CDM isocurvature mode.

            -0.0.1  (10/01/10)
                    -First order weak lensing
          implemented in the harmonic approach of astro-ph/0001303.
                    -Compilation of certain functions
          in order to fasten the computation.
                    -Addition of plots of the most recent data, that
          could be superimposed to the actual plots of CMBquick1.
```

Notation and Conventions

■ The parameters

The cosmological parameters

The cosmological parameters are given as arguments to nearly all functions.

In that sense, CMBquick1 is not procedural, but functional. Values are going to be computed only if you ask for and exactly when you ask for, and not before. It is thus interactive.

Only the parameters which are not likely to be modified are given a fixed value (which can be anyway modified by the user)

A set of parameters list are precomputed. These are in order

-WMAP5

-WMAP7

-WMAP5 but with one of neutrinos having a mass (.1eV)

-WMAP5 but with a tensor to scalar ratio of 1, and a scale-invariant power spectrum of the primordial tensor modes

-WMAP5 but without reionization

The values of the parameters are

```
In[116]:=
     Grid[{CPL, CPL7, CPLν, CPLGW, CPLNoRei}, Frame → All]
```

```
Out[116]=
```

2.726	0.719	3.046	0.1326	0.02273	2.41	0.963	0.087	1.	{}	0	1
2.726	0.714	3.046	0.1334	0.0227	2.38	0.969	0.086	1.	{}	0	1
2.726	0.719	2.046	0.1326	0.02273	2.41	0.963	0.087	1.	{1}	0	1
2.726	0.719	3.046	0.1326	0.02273	2.41	0.963	0.087	1.	{}	1	1
2.726	0.719	3.046	0.1326	0.02273	2.41	1	0	1	{}	0	1

We see that the order of parameters is

{ T_0, h, Number of massless ν's, $\Omega_{m0}\,h^2$, $\Omega_{b0}\,h^2$, $A_s{}^2$, n_s, τ_{rei} , Reionization Fraction, {List of masses of massive ν's}, r (tensor to scalar ratio), tensor spectral index }

You can of course choose whatever list of cosmological parameters. CMBquick1 has not been tested with fancy parameters, so you will probably obtain an error if you ask too much.

However, since recombination is computed with physical parameters (x) rather than redshift, it should be robust since the switch between Saha and dynamical recombination

is depending on the temperature, and not on the redshift.

I would be glad if you could send me then the notebook which produced the error, so that I can ensure the error does not appear again.

The numerical parameters of the k-spacing

The main parameters of the numerical integrations are:

`kmin` and `kmax` which are the boundaries of the k-space interpolations. `Nk` is the number of points in the k-space.

```
In[117]:=
     kmin
     kmax
     Nk

Out[117]=
     0.0133333

Out[118]=
     35

Out[119]=
     140
```

The modes are spaced logarithmically in that range, whit a refinement of the k < 1 region. This is determined by the variable '`koffset`'. Broadly speaking, it determines the k which separates the highly sampled region in k-space to the normally/poorly sampled region.

The list of modes k used is then stored in the variable `ListK`

```
In[120]:=
     ListK

Out[120]=
     {0.0133333, 0.0396984, 0.0667494, 0.0945042, 0.122981, 0.152199, 0.182177, 0.212935,
      0.244493, 0.276873, 0.310095, 0.344181, 0.379154, 0.415037, 0.451854, 0.489628,
      0.528386, 0.568151, 0.608952, 0.650814, 0.693765, 0.737833, 0.783048, 0.82944,
      0.877039, 0.925876, 0.975983, 1.02739, 1.08014, 1.13427, 1.18979, 1.24677,
      1.30523, 1.3652, 1.42674, 1.48988, 1.55466, 1.62113, 1.68933, 1.7593, 1.83109,
      1.90475, 1.98033, 2.05787, 2.13743, 2.21906, 2.30281, 2.38875, 2.47692,
      2.56738, 2.66019, 2.75543, 2.85314, 2.95339, 3.05625, 3.16178, 3.27006,
      3.38116, 3.49515, 3.61211, 3.73211, 3.85523, 3.98155, 4.11116, 4.24415,
      4.38059, 4.52058, 4.66422, 4.81159, 4.9628, 5.11794, 5.27711, 5.44043, 5.608,
      5.77993, 5.95633, 6.13732, 6.32302, 6.51355, 6.70904, 6.90961, 7.11541,
      7.32656, 7.5432, 7.76547, 7.99354, 8.22753, 8.46761, 8.71394, 8.96668, 9.226,
      9.49206, 9.76504, 10.0451, 10.3325, 10.6274, 10.9299, 11.2403, 11.5587,
      11.8855, 12.2207, 12.5647, 12.9177, 13.2798, 13.6513, 14.0325, 14.4236,
      14.8249, 15.2366, 15.6591, 16.0925, 16.5372, 16.9935, 17.4617, 17.942,
      18.4349, 18.9405, 19.4593, 19.9917, 20.5378, 21.0982, 21.6731, 22.2631,
      22.8683, 23.4893, 24.1265, 24.7802, 25.451, 26.1392, 26.8453, 27.5698,
      28.3131, 29.0758, 29.8583, 30.6612, 31.485, 32.3302, 33.1973, 34.0871, 35.}
```

The full set of equations with radiation is integrated only when k <= `ksmall`. Indeed for larger modes, radiation provides only oscillations which are unobservable, even during reionization as they average out.

For modes larger than `ksmall`, the integration is performed exactly until `yEearly` (see further), and then it switches to a system without radiation.

```
In[121]:=
     ksmall

Out[121]=
     3
```

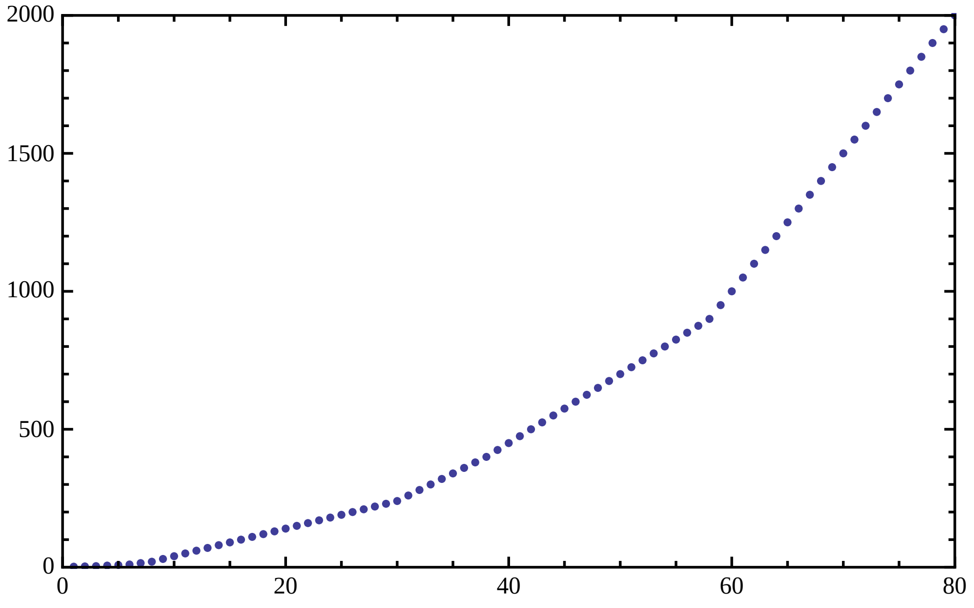
The numerical parameters of the Cls

The list of l' s for which the Cl's are computed is stored in '`ListlUsedinBesselFunction`'. This plot summarizes the first 80 values.

Beyond the l's are sampled every Δl=25.

```
In[122]:=
      ListPlot[ListlUsedinBesselFunction,
       PlotRange → {{0, 80}, {0, ListlUsedinBesselFunction[[80]]}}, Frame → True]
```

```
Out[122]=
```



Numerical parameters of the time integration

The numerical integration is performed in $y = a/a_{eq}$. Th e starting point is given by `yi1`

```
In[123]:=
      yi1
```

```
Out[123]=
      0.0001
```

Then for small modes (smaller than ksmall), the integration is done through the function `Sys1`, which integrates up to y_0. For large modes, the integration is split in two parts.

First 'Sys1Early' integrates up to a given y_{end} which is taken to be `'yEearly'` (see further for the definition). Then `'Sys1Late'` integrates the rest of the time up to y_0 dropping the radiation and neutrinos moments, but keeping massive neutrinos if relevant.

The accuracy of the numerical integration is controlled by the variable `$Accuracy1`. The Numerical integration will be performed in order to give a result valid at $10^{(-\$Accuracy1)}$.

```
In[124]:=
      $Accuracy1
```

```
Out[124]=
      5
```

However if the option ' `$AdaptativePrecision1`' is set to True, this is overwritten and CMBquick adapts the numerical precision depending on the mode k integrated.

It refines at low k (higher accuracy), in order to ensure tight coupling numerically. This option is not necessary for CMB (and that's why the default value is False).

It is only interesting if you are interested in the transfer function in great details...

Numerical parameters of the line of sight integration

The number of points used to perform the numerical integral in the line-of-sight approach, is controlled by the number of steps used in the time integration, and the number of steps used in the k integration.

-The Number of steps used in the time integration is controlled by '`SamplePoints`' (For Last Scattering effects and Early type effects) and '`SamplePointsLate`' (For late effects and reionization).

```
In[125]:=
      SamplePoints
      SamplePointsLate

Out[125]=
      100

Out[126]=
      250
```

-The step size used in the k-integrals is given by '`SizeStepk`'

```
In[127]:=
      SizeStepk

Out[127]=
      0.0133333
```

Overall precision

It is possible to modify most of the numerical parameters together, just by modifying the value of '`Myprecision`'.

This will act on most of the numerical parameters. If you set `MyPrecision` to 1, you will obtain the default precision, but you can increase it to 3, in order to obtain high accuracy

at a cost of longer computations of course.

```
In[128]:=
      Myprecision

Out[128]=
      1
```

■ Naming of Functions

Species in the universe

There are several types of species. First we have the radiation type species, which are neutrinos and photons

Then we have baryons and cold dark matter. They are referred to by the letters `N, R, B, C`.

The suffix o1 (like 'order 1') is then added to all related quantities, in order to differentiate these quantities in CMBquick2.

For Baryons we have for instance

```
In[129]:=
      ? "CMBquick`CMBquick1`N*o1*"
      ? "CMBquick`CMBquick1`R*o1*"
```

```
Neutrinos moments at first order. Syntax is No1[cpl][L][k][y]
```

```
▼ CMBquick`CMBquick1`
R0tc2o1    R0tco1    R1tc2o1    R1tco1    Ro1
```

And for matter (click to get information on a given function)

```
In[131]:=
      ? "CMBquick`CMBquick1`B*o1*"
      ? "CMBquick`CMBquick1`C*o1*"
```

```
▼ CMBquick`CMBquick1`
B0o1    B0tc2o1    B0tco1    B1o1    B1tc2o1    B1tco1

▼ CMBquick`CMBquick1`
C0o1                C1o1
```

`B0o1` is the monopole which indentifies with the energy density, and `B1o1` is the dipole, which identifies with -kV where V is the scalar velocity.

The same thing for cold dark matter. We must then specify the `CPL`, the Fourier mode (which k"), and the time (which y?).

The syntax is thus, for instance

B0o1[CPL][1][1]

Again you can learn about the syntax just by typing '`?B0o1`' in that case. This is important since you can get information on any variable by that mean.

```
In[133]:=
      ? B0o1
```

```
Baryons monopole (zeroth moment). Syntax is B0o1[cpl][k][y]
```

For radiation, that is neutrinos and photons, the moments go up to $l_{max} = 8$,
and the number of the multipole is an argument. The name for photons brightness perturbation is thus 'Ro1'
and then we must specify the CPL, the moment (which l?) the mode (which k),
and which y (which time?). The syntax is thus for instance
`Ro1[CPL][0][1][1]`

```
In[134]:=
      ? Ro1
```

```
Radiation moments at first order. Syntax is Ro1[cpl][L][k][y]
```

Geometrical quantities

We have two scalar perturbations, which are `Φ1` and `Ψ1`, and then tensor perturbations '`To1`'. They depend on the `CPL`, the Fourier mode and the y. The value of the gravitational potential today for $k = k_{\mathrm{eq}}$ is thus

```
Φ1[CPL][1]@y0[CPL]
```

```
In[135]:=
      ? "CMBquick`CMBquick1`Φ*1"
```

▼ CMBquick`CMBquick1`

Φ1	Φp1

Conversion functions

We have several functions to descrive the time in the Universe.

We can either use the reduced scale factor $y = a/a_{\mathrm{eq}}$ or the conformal time η, or the redshift z,
or finally the other reduced scale factor x, which is normalized to unity when $T_{\mathrm{CMB}} = 13.6\,\mathrm{eV}$.

Conversion functions are provided, and are of the form aOFb. They depend always from the cosmology (except the relation between z and y), and their syntax is thus

```
a=aOFb[cpl][b]
```

The suffix `M` can also be added sometimes. It stands for 'Memorize'. In that case, once a value has been computed, it is stored and the next access to this value is faster.

```
In[136]:=
      ? "CMBquick`CMBquick1`*OF*"
```

▼ CMBquick`CMBquick1`

eta⋱ O⋱ Fy	eta⋱ O⋱ Fy⋱ M	eta⋱ O⋱ Fz	xOFy	yOF⋱ et⋱ a	yOF⋱ et⋱ a⋱ M	yOFx	yOFz	zOF⋱ et⋱ a	zOFy

We can for instance check the redshift at equivalence (by definition y=1)

```
In[137]:=
      zOFy[CPL][1]

Out[137]=
      3167.
```

Numerical tools

Numerical integrals

Input and output

Some low level functions are written to output and input the Bessel functions.

They are also used to output (and input) the results of the numerical integration for the sources.

You should not need these functions.

Background functions

Friedmann equations

Hubble function

The Hubble function is given by '`H[cpl][y]`'. Unless there are massive neutrinos, its expression is analytic since we know exactly with which power of y the different types of energy densities decrease $(1/y^3$ for cold dark matter and baryons, $1/y^4$ for neutrinos and photons).

```
In[145]:=
        ? "CMBquick`CMBquick1`H"
```

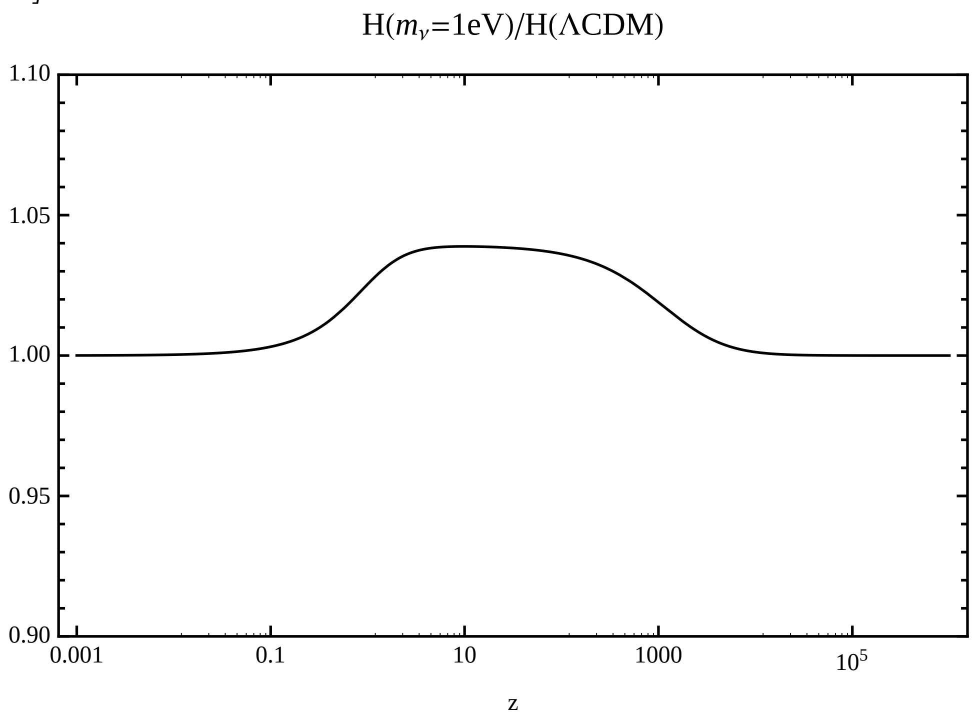
"Hubble function H(y) in units of k_{eq}. Syntax is 'H[cpl][y]'"

We can for instance see how a massive neutrino (1eV) affects the Hubble function, by comparing the Hubble function to the case where there are only massless neutrinos (plotting their ratio).

Since the massive neutrinos shifts the notion of "equivalence" between radiation and matter, this change the conversion of k_{eq} into Mpc. Since the conformal time Hubble function `H` is expressed in units

of k_{eq} we need to take that into account by defining:

```
In[146]:=
        HMpc[cpl_][y_] := H[cpl][y] / RescaleMp[cpl];

In[147]:=
        MyLogLinearPlotBandW[{(HMpc[CPLν][yOFz[CPLν][z]]) / (HMpc[CPL][yOFz[CPL][z]])},
         {z, 10^-3, 10^6}, PlotRange -> {0.9, 1.1},
         PlotLabel -> "H(mν=1eV)/H(ΛCDM)", FrameLabel -> {"z", ""}]

Out[147]=
```



At late time, the cosmological constant dominates and the Hubble functions are the same.

At early time, the massive neutrinos are relativistic and thus there is no difference with the purely massless case.

At intermediary times, the massive neutrinos contribute to massive matter and the Hubble function is thus larger.

Abundancies

The abundances are then derived, and are also functions of y

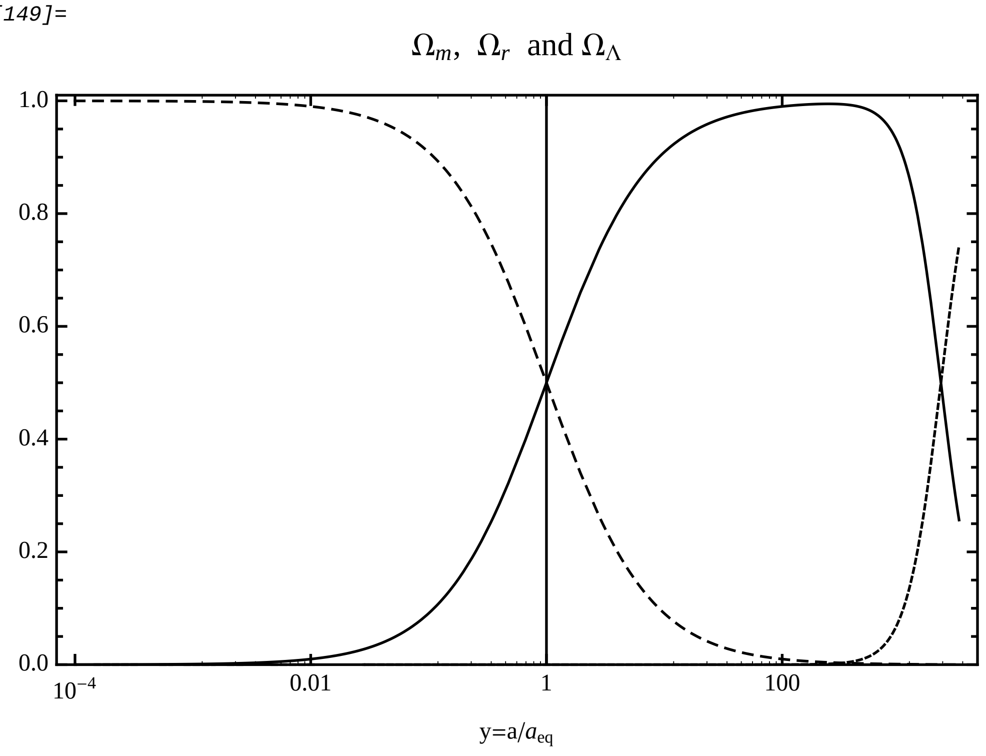
```
In[148]:=
      ? "CMBquick`CMBquick1`Ω*"
```

▼ CMBquick`CMBquick1`

Ωb	Ω⋱b⋱0	Ω⋱b⋱0⋱h⋱2⋱W⋱M⋱A⋱P	Ωc	Ω⋱c⋱0	Ω⋱m	Ω⋱m⋱0	Ω⋱m⋱0⋱h⋱2⋱W⋱M⋱A⋱P	Ωr	Ω⋱r⋱0	Ω⋱Λ	Ω⋱Λ⋱0	Ω⋱ν⋱m	Ω⋱ν⋱m⋱0

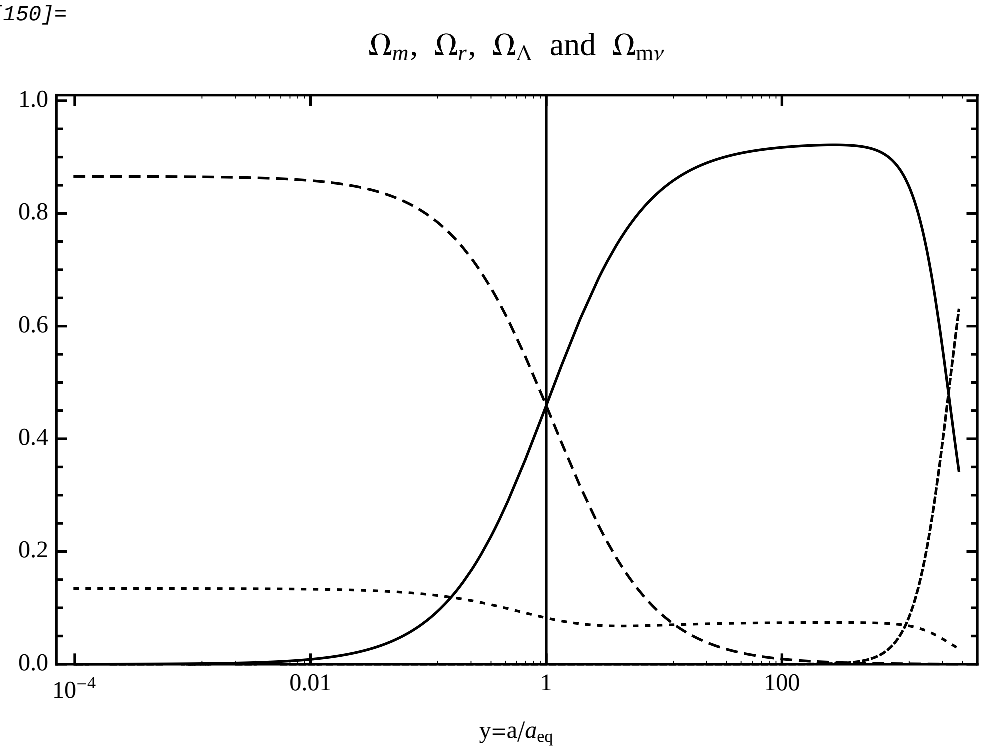
We can check that equivalence at y=1 is indeed the equivalence:

```
In[149]:=
      MyLogLinearPlotBandW[{Ωm[CPL][u], Ωr[CPL][u], ΩΛ[CPL][u]},
       {u, 0.0001, y0[CPL]}, FrameLabel -> {"y=a/a_eq", ""},
       PlotLabel -> "Ωm,  Ωr  and ΩΛ", PlotRange -> {0, 1.01}, GridLines → {{1}, {}}]
Out[149]=
```



Or for a model with one massive neutrino :

```
In[150]:=
      MyLogLinearPlotBandW[{Ωm[CPLν][u], Ωr[CPLν][u], ΩΛ[CPLν][u], Ωνm[CPLν][u]},
       {u, 0.0001, y0[CPL]}, FrameLabel -> {"y=a/a_eq", ""},
       PlotLabel -> "Ωm,  Ωr,  ΩΛ  and  Ωmν", PlotRange -> {0, 1.01}, GridLines → {{1}, {}}]

Out[150]=
```



We can check that the y of equivalence is indeed 1 numerically, up to the machine precision, with the simple function:

```
In[151]:=
      yeq[cpl_] :=
        First@Select[u /. Solve[Ωm[cpl][u] == Ωr[cpl][u], u], (Im[#] == 0) && (# > 0) &];

In[152]:=
      yeq[CPL]
      yeq[CPLν]

Out[152]=
      1.

Out[153]=
      1.
```

We also notice that the massive neutrinos are not counted as matter nor as radiation, though they go from one regime to another.

The equivalence between radiation and matter is thus defined with photons and massless neutrinos versus cold dark matter and baryons

A similar method based on intersection of functions can lead to the redshift at which the cosmological constant starts to dominate the energy balance:

```
In[154]:=
      zΛ[cpl_] := zOFy[cpl]@First@
          Select[u /. Solve[Ωm[cpl][u] == ΩΛ[cpl][u], u], (Im[#] == 0) && (Re[#] > 0) &];
```

```
In[155]:=
      zΛ[CPL]

Out[155]=
      0.42577
```

And of course with one massive neutrino (using `CPLν`), there is more massive matter and this redshift is slightly smaller ...

```
In[156]:=
      zΛ[CPLν]

Out[156]=
      0.412504
```

Physical time

It is possible to know for any y the physical time it corresponds to, with the function `AgeUniverse`. Its syntax is `AgeUniverse[cpl][y]`.

For instance the age of the universe at equivalence or at the middle of the Last Scattering Surface and today (see further for definitions) can be inferred by evaluating

```
In[157]:=
      AgeUniverse[CPL][1]
      AgeUniverse[CPL][yLSS[CPL]]
      AgeUniverse[CPL][y0[CPL]]

Out[157]=
      58 809.1

Out[158]=
      398 523.

Out[159]=
      1.36849 × 10^10
```

■ Recombination

Ionization fraction

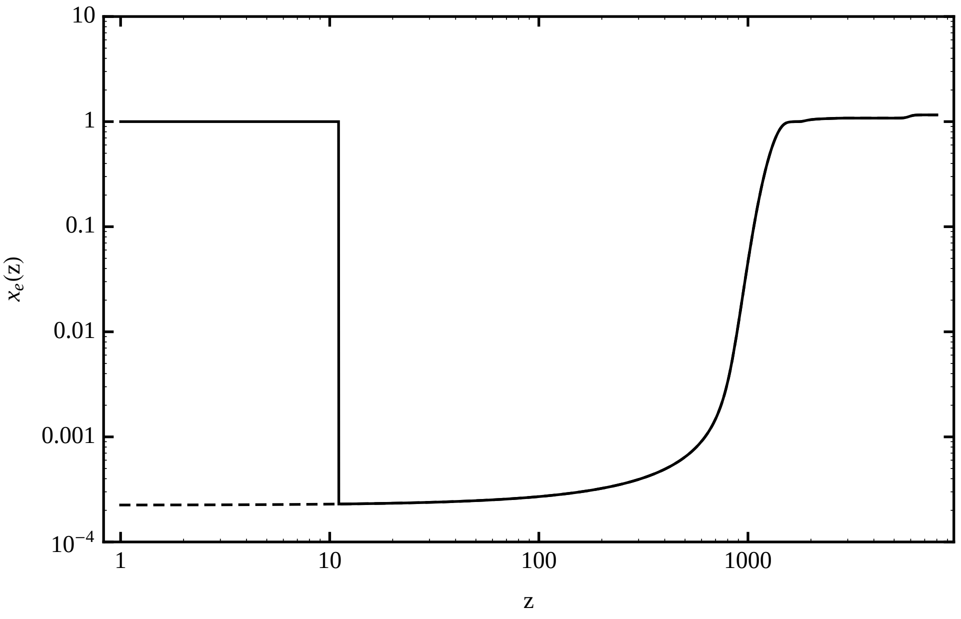
The fraction of free electrons is given by '`xe`'

When we plot it, we see the Helium ionization bumps, and `xe`>1 at early time. This is based on RECFAST algorithm.

If we take a set of parameters with no reionization (the dashed curve), there is no jump at low redshift, as expected...

```
In[160]:=
     MyLogLogPlotBandW[{xe[CPL][z], xe[CPLNoRei][z]}, {z, 1, 8000},
      PlotRange → {0.0001, 10}, Frame → True, FrameLabel → {"z", "xe(z)"}]
```

```
Out[160]=
```



If we wish to use Saha equilibrium for the HeI recombination (and not just for HeII recombination as in RECFAST), then we can set '`MethodRecombinationHe`' to '`SAHA`' instead of its default value which is:

```
In[161]:=
     MethodRecombinationHe
```

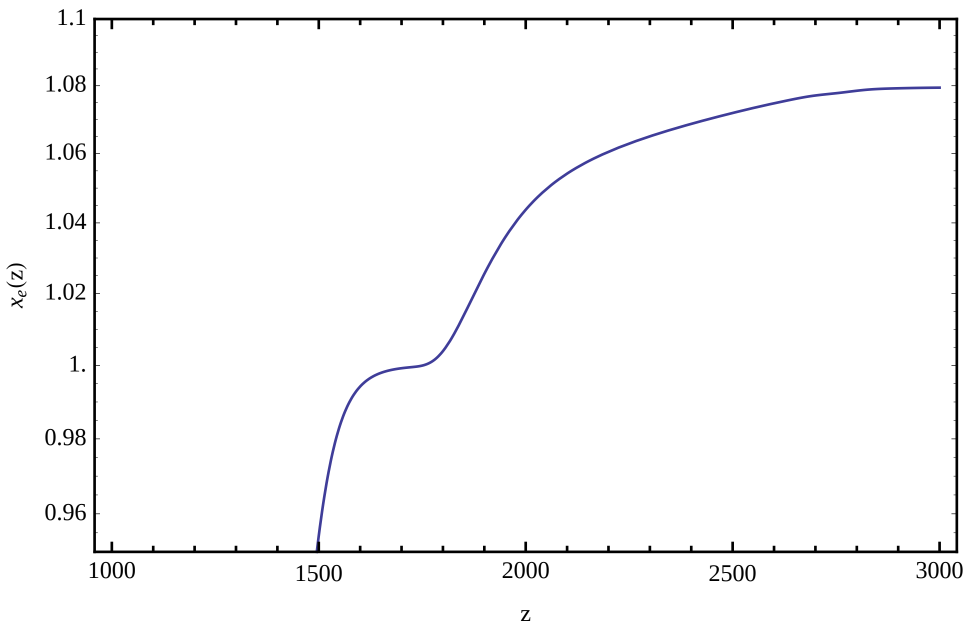
```
Out[161]=
     RECFAST
```

As you can notice, this is not a parameter included in the 'cosmological parameters list', and thus you would need to set this option before any evaluation in order to have an effect (since objects already computed are stored).

This choice might change in the future if it appears that the users want to have a good control on the recombination method. For the moment, we will consider that RECFAST is the best recombination method.

We can see that our implementation is similar to Figure 2 of this recent RECFAST paper (Wong et. al. 2007):

```
In[162]:=
      LogPlot[xe[CPL][z], {z, 1000, 3000},
       PlotRange → {0.95, 1.1}, Frame → True, FrameLabel → {"z", "xe(z)"}]

Out[162]=
```



For the moment, the implementationof RECFAST is a simplified one, where the temperature of matter is equal to the temperature of radiation.

The fudge factors of the RECFAST algorithm are

```
In[163]:=
      ? Fudge*
```

▼ CMBquick`CMBquick1`

FudgeH	FudgeHe

And the default values are the recommended ones:

```
In[164]:=
      FudgeH
      FudgeHe

Out[164]=
      1.14

Out[165]=
      0.86
```

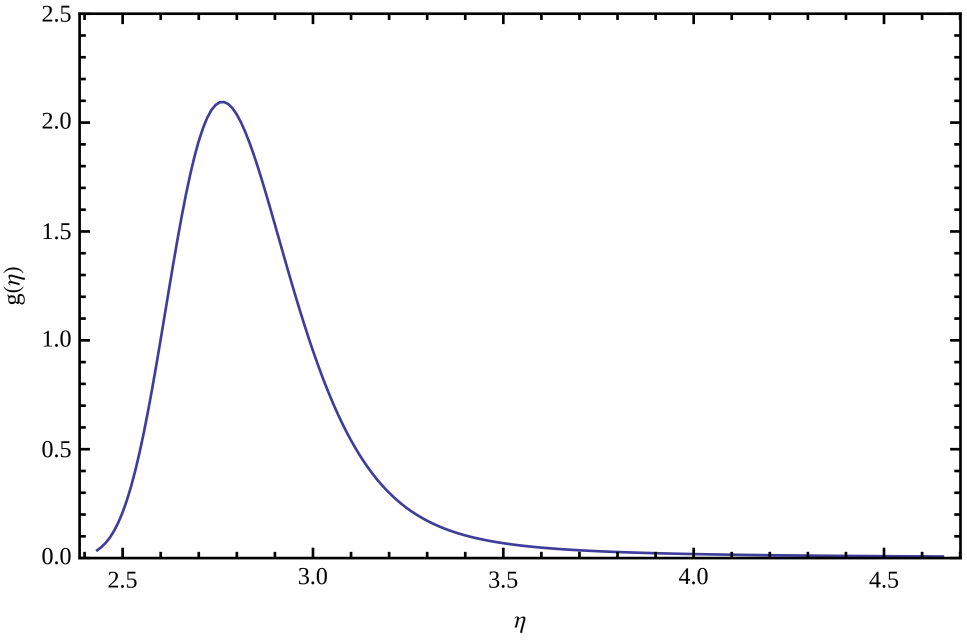
Visibility function at LSS and reionization

The visibility function is provided by `VoftM`. Syntax is '`VoftM[cpl][`η`]`'. M stands for 'Memorize', and values calculated once will be accessed very quickly. '`oft`' stand for 'it is a function of (conformal) time'.

This function has the conformal time as argument, since it is used mostly in the line of sight integral where the integration is performed on η.

```
In[166]:=
     ListPlot[Table[{t, VoftM[CPL][t]}, {t, ηBLSS[CPL], ηELSS[CPL], dηLSS[CPL] / 2}],
      PlotRange → {0, 2.5}, Joined → True, Frame → True,
      Axes → False, FrameLabel -> {"η", "g(η)"}]
```

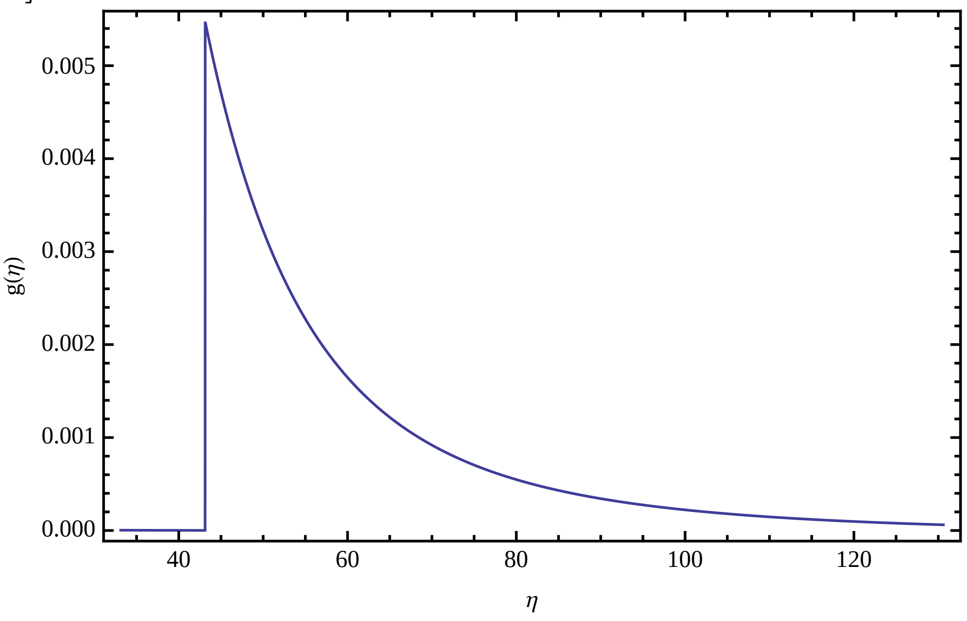
```
Out[166]=
```



And the bump at reionization. Our model is instant reionization so this is sharp. The user is free to modify the model in the source code of CMBquick1 (It is a function of $a = y/y_0 = (1+z)^{-1}$).

```
In[167]:=
     ListPlot[Table[{t, VoftM[CPL][t]}, {t, ηBrei[CPL] - 10, ηErei[CPL], dηLSS[CPL] / 2}],
      Joined → True, Frame → True, FrameLabel -> {"η", "g(η)"}, Axes → False]
```

```
Out[167]=
```



Specific times

In the previous subsection, we see that use was made of some specific times.

These are in order

*Last scattering surface

-"Beginning of Last Scattering Surface (LSS)" -> η`BLSS`

-"Middle of the LSS" -> η`LSS`

-"End of LSS" -> η`ELSS`

*Early effects

-"End of early effects (up to when the early ISW is negligible, or up to the beginning of reionization if reionization included)" -> η`Eearly`

* Reionization and late effects

-"Beginning of reionization (similar to ηEearly if reionization)" -> η`Brei`

-"End of reionization (when the effects of reionization are completely negligible)" -> η`Erei`

-"Time today" -> η`0`

```
In[168]:=
        ?"CMBquick`CMBquick1`η*"
```

▼ CMBquick`CMBquick1`

ηO	ηBLSS	ηBrei	ηEearly	ηELSS	ηErei	ηLSS

These specific times exist as well for the y parameter, and you only need to replace η by y in the previous name

This specific times enable to define periods which are

Period	from	to	Name
–LSS	ηBLSS	ηELSS	'LSS'
–Early effects	ηELSS	ηEearly	'Early'
–Late effects	ηEearly	η0	'Late'

A period for reionization is also defined, but it is a subperiod of the Late effects period. It is not used.

Distances

It is also possible to extract the (comoving) distance from any time η, that is more precisely the angular distance, with the function `Dist`.

It is just given by η`0-`η since only flat universes are allowed for the moment.

```
In[169]:=
        ?"CMBquick`CMBquick1`Dist*"
```

▼ CMBquick`CMBquick1`

Dist	DistLSS

For instance the angular distance of the Last Scattering Surface (LSS) is given in units of $k_{eq}{}^{-1}$

```
In[170]:=
      Dist[CPL]@ηLSS[CPL]

Out[170]=
      136.346
```

And it can be converted in Mpc with `RescaleMp`

```
In[171]:=
      % RescaleMp[CPL]

Out[171]=
      14102.1
```

Time steps

For each period, we define a timestep, and the names are:

```
In[172]:=
      ? "CMBquick`CMBquick1`dη*"
```

▼ CMBquick`CMBquick1`

dηearly	dηlate	dηLSS	dηrei

These time steps are used in the time integration on the sources.

They are calculated such that the number of points in the `LSS` and `Early` periods is '`SamplePoints`'

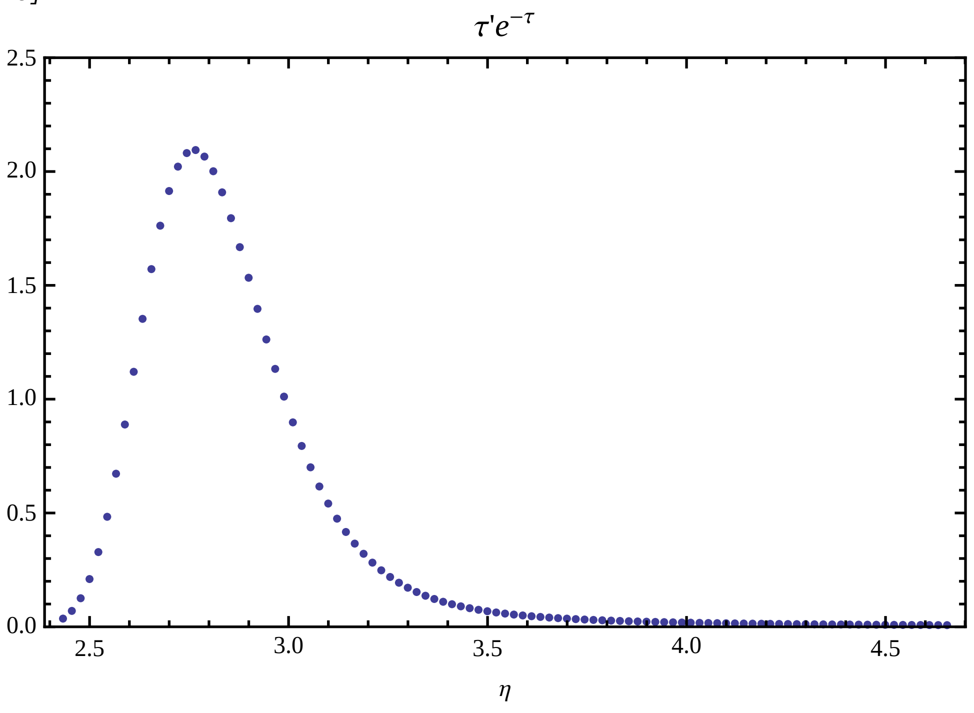
and such that the number of points in the Late effects period is '`SamplePointsLate`'

```
In[173]:=
      SamplePoints
      SamplePointsLate

Out[173]=
      100

Out[174]=
      250
```

The list of corresponding points can be obtained from '`Listt[cpl,period]`'

For instance we can plots the sampling points of the visibility function in the LSS

```
In[175]:=
     ListPlot[Table[{t, VoftM[CPL][t]}, {t, Listt[CPL, LSS]}],
      PlotRange → {0, 2.5}, Frame → True, FrameLabel -> {"η", ""}, PlotLabel -> "τ'e^-τ"]

Out[175]=
```



Optical depth

The optical depths is given as a function of y

```
In[176]:=
     ?"CMBquick`CMBquick1`τ*y*"
```

▼ CMBquick`CMBquick1`

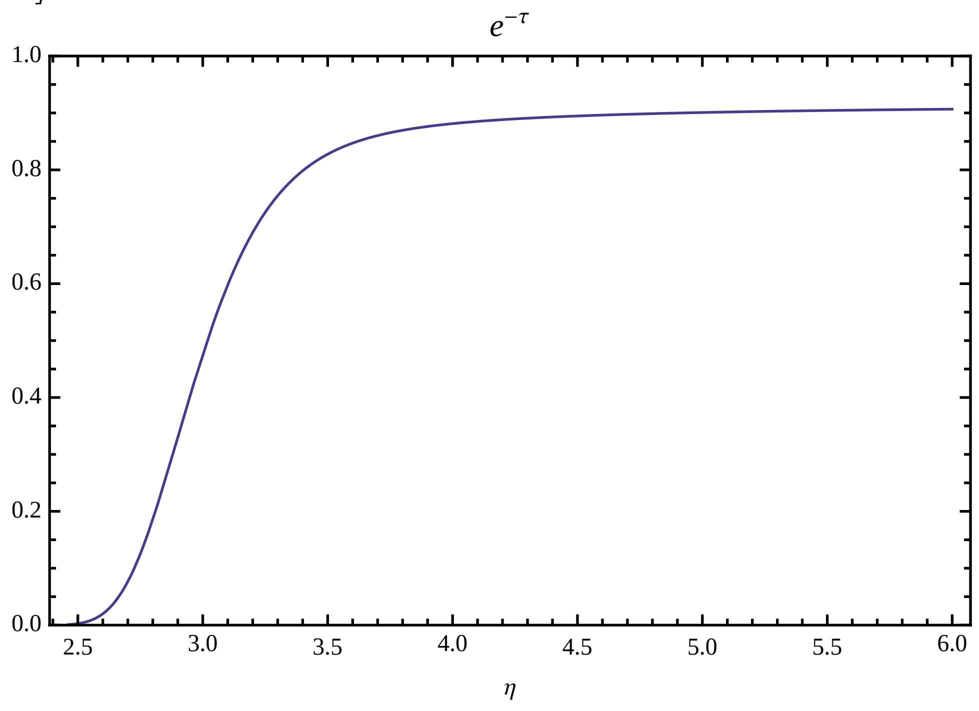
τpyM	τpyMnoRei	τyM

We see that we have a function for τ (`τyM`) and one for $d\tau/dy$ (`τpyM`). Again, the suffix `M` means that values are memorized once there are computed once. This can increase the speed but can also fill the memory.

'`y`' means that it is a function of `y` and `'p'` stands for prime.

We can plot $e^{-\tau}$during the LSS period:

```
In[177]:=
     Plot[Exp[-τyM[CPL][y]], {y, yBLSS[CPL], yELSS[CPL]}, PlotRange → {0, 1},
      Frame → True, Axes → False, FrameLabel -> {"η", ""}, PlotLabel -> "e^-τ"]

Out[177]=
```



It does not converge to 1 at the end of the LSS, since there is reionization later

Cosmological parameters on the background space-time

You can obtain information on a specific model only on the background spacetime, by typing

```
In[178]:=
     Cosmology[CPL, BackgroundParameters]

Out[178]=
```

Variable	Value	Units	Comment
Ω_{b0}	0.043969		Abundance of baryons
Ω_{c0}	0.21253		Abundance of CDM
Ω_{r0}	0.000080966		Abundance of radiation (massless ν's and photons)
$\Omega_{\Lambda 0}$	0.74342		Abundance of Λ
Ω_K	0		Abundance of curvature
T_0	2.726	K	Temperature of CMB
N_ν	3.046		Number of massless neutrinos
h	0.719		Reduced Hubble constant
τ_{rei}	0.087		Optical depth of reionization
n_s	0.963		Scalar perturbations spectral index
k_{eq}	0.0096685	Mpc^{-1}	k at equivalence time
z_{rei}	11.045		Redshift at reionization
z_{eq}	3167.		Redshift at equivalence
z_{LSS}	1059.8		Redshift at $\tau-\tau_{rei}=\ln(2)$
z_{dec}	1089.		Redshift at max of visibility function $(\tau' e^{-\tau})$
z_*	1089.6		Redshift at $\tau-\tau_{rei}=1$
$d_A(z_*)$	14108.	Mpc	Angular distance at z_*
$d_A(z_{eq})$	14272.	Mpc	Angular distance at equivalence
D_H	4169.58	Mpc	Hubble distance today
t_0	13.6849	Gyears	Age of the Universe
t_*	380 300.	years	Age of universe at z_*
$r_{hor}(z_{dec})$	285.66	Mpc	Radius of horizon at z_{dec}
η_0	14394.	Mpc	Conformal time today

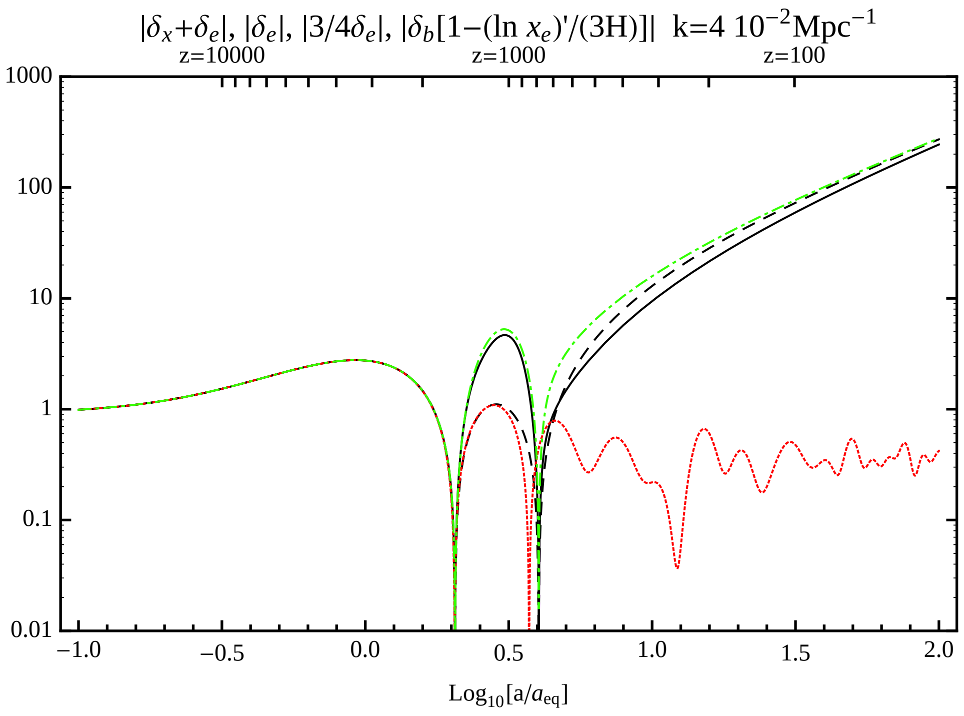
First order functions

■ Perturbed recombination

The method implemented is described in Senatore et. al. 2008

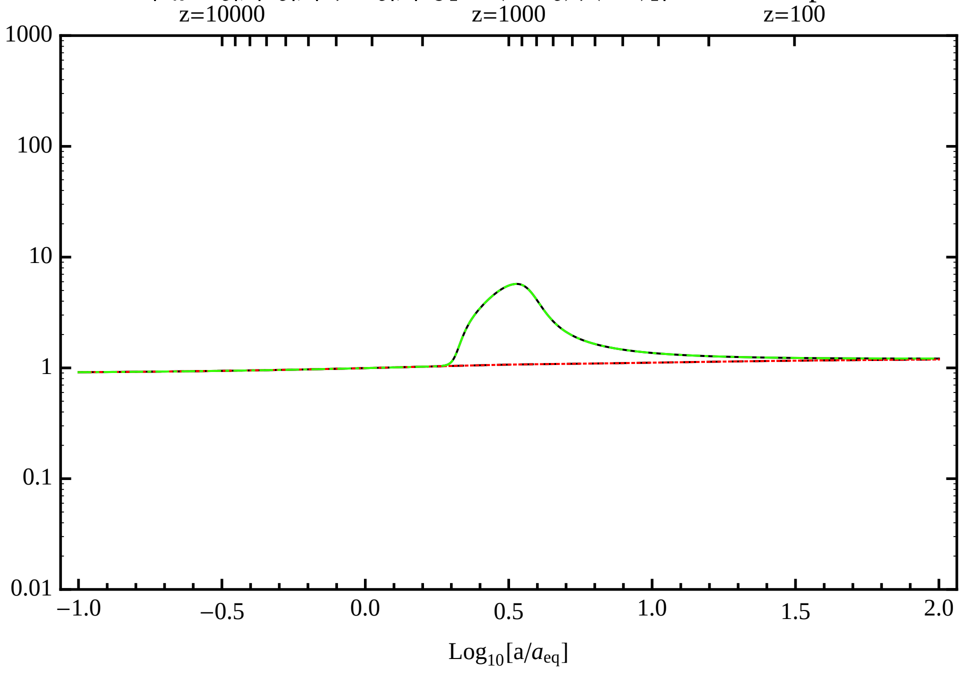
We reproduce their plots :

```
In[179]:=
     NormalizeaZeta = 3 / 5;
     ktest = 0.04 * RescaleMp[CPL]; (* 0.04 Mpc *)

     drec1 = LogPlot[{Abs[NormalizeaZeta DeltX[CPL][ktest][10^p] +
           NormalizeaZeta B0o1[CPL][ktest][10^p]],
         Abs[NormalizeaZeta B0o1[CPL][ktest][10^p]],
         Abs[NormalizeaZeta 3 / 4 Ro1[CPL][0][ktest][10^p]],
         Abs[NormalizeaZeta B0o1[CPL][ktest][10^p]
           (1 - 1 / 3 xeH[CPL] '[10^p] 10^p / xeH[CPL][10^p])]}, {p, -1, 2},
       PlotRange → {0.01, 1000}, PlotStyle → {{GrayLevel[0], Thickness[0.00225]},
         {Thickness[0.00225], Dashing[{0.013}], GrayLevel[0]},
         {Thickness[0.00225], Dashing[{0.002, 0.004}], Hue[0]},
         {Thickness[0.00225], Dashing[{0.013, 0.007, 0.002, 0.007}], Hue[0.3]}},
       PlotLabel → "|δx+δe|, |δe|, |3/4δe|, |δb[1-(ln xe)'/(3H)]|  k=4 10^-2Mpc^-1",
       FrameLabel → {"Log10[a/aeq]", ""}, Frame → True, FrameTicks → MyTicksZ[CPL]]

Out[181]=
```



```
In[182]:=
     ktest = 0.0001 * RescaleMp[CPL];

     drec2 = LogPlot[{Abs[NormalizeaZeta DeltX[CPL][ktest][10^p] +
           NormalizeaZeta B0o1[CPL][ktest][10^p]],
         Abs[NormalizeaZeta B0o1[CPL][ktest][10^p]],
         Abs[NormalizeaZeta 3 / 4 Ro1[CPL][0][ktest][10^p]],
         Abs[NormalizeaZeta B0o1[CPL][ktest][10^p]
           (1 - 1 / 3 xeH[CPL]'[10^p] 10^p / xeH[CPL][10^p])]}, {p, -1, 2},
       PlotRange → {0.01, 1000}, PlotStyle → {{GrayLevel[0], Thickness[0.00225]},
         {Thickness[0.00225], Dashing[{0.013}], GrayLevel[0]},
         {Thickness[0.00225], Dashing[{0.002, 0.004}], Hue[0]},
         {Thickness[0.00225], Dashing[{0.013, 0.007, 0.002, 0.007}], Hue[0.3]}},
       PlotLabel → "|δx+δe|, |δe|, |3/4δe|, |δb[1-(ln xe)'/(3H)]|  k=10^-4Mpc^-1",
       FrameLabel → {"Log10[a/aeq]", ""}, Frame → True, FrameTicks → MyTicksZ[CPL]]

Out[183]=
```

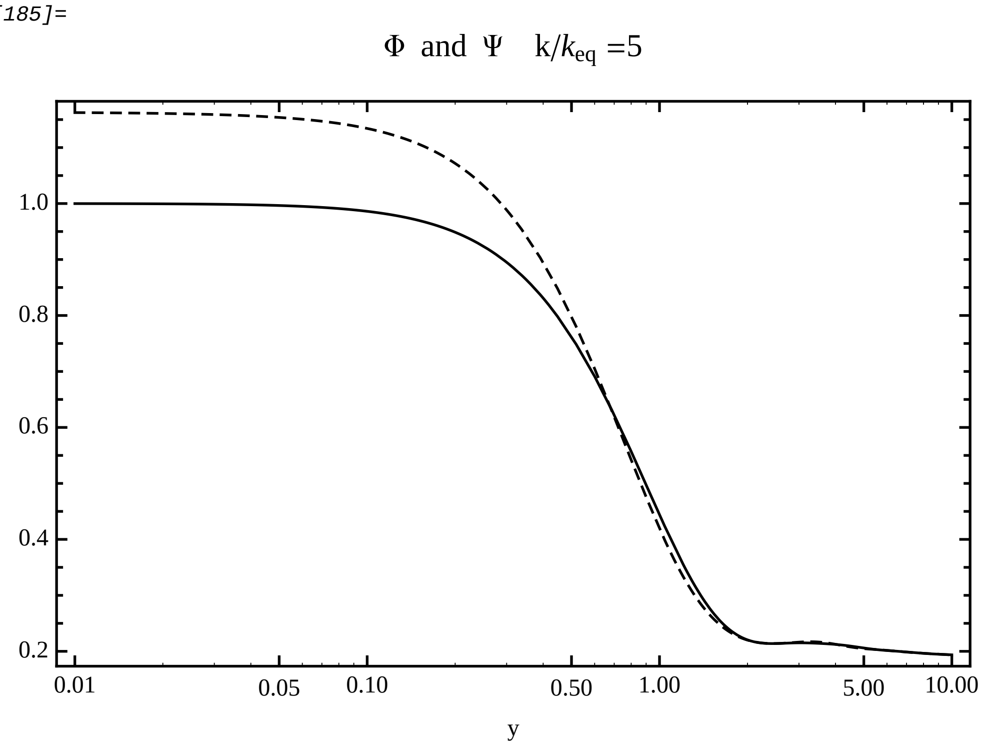
$|\delta_x+\delta_e|$, $|\delta_e|$, $|3/4\delta_e|$, $|\delta_b[1-(\ln x_e)'/(3H)]|$ k=10^{-4}Mpc^{-1}



■ First order numerical integration

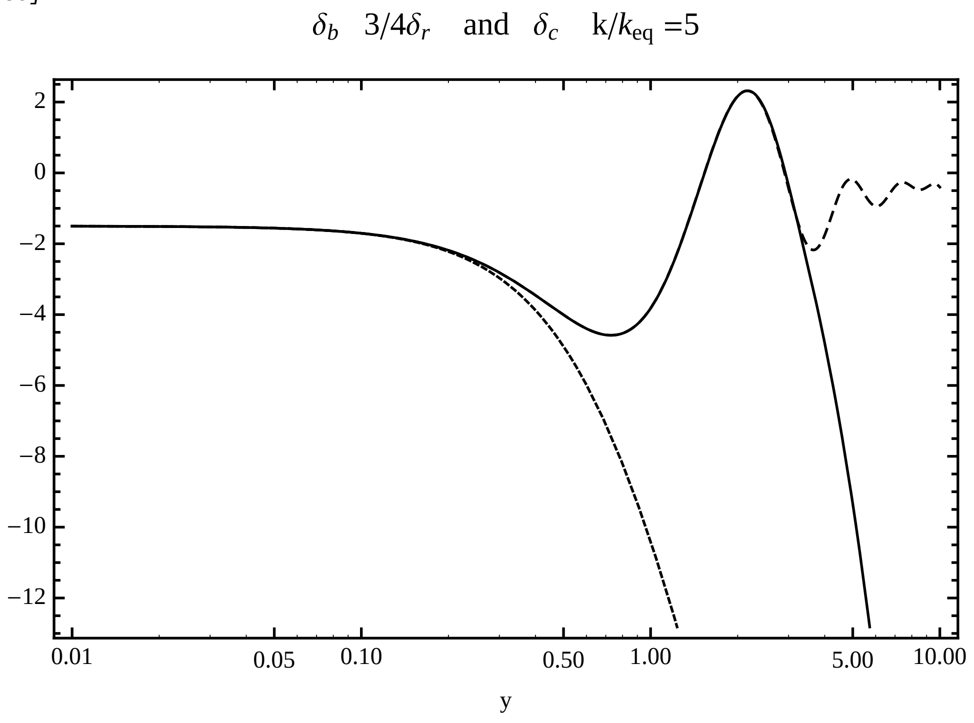
Potentials

```
In[184]:=
     kt = 5;
     MyLogLinearPlotBandW[{Φ1[CPL][kt][u], Ψ1[CPL][kt][u]}, {u, 0.01, 10},
      PlotLabel → "Φ  and  Ψ    k/k_eq  ="~StringJoin~ ToString[kt],
      FrameLabel → {"y", ""}]

Out[185]=
```



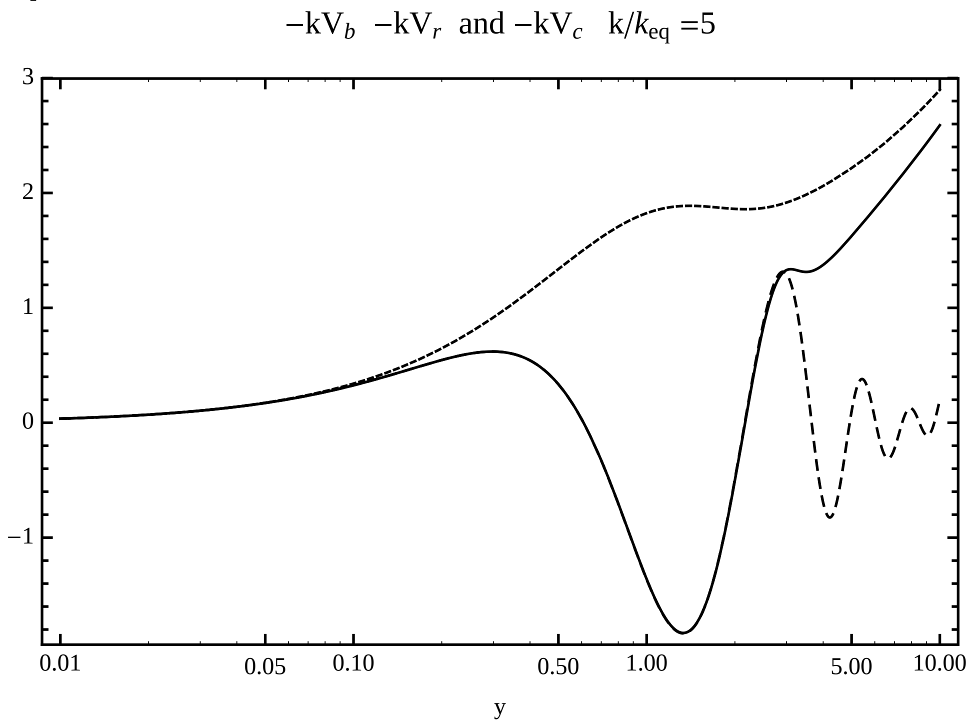
Densities

```
In[186]:=
      MyLogLinearPlotBandW[{B0o1[CPL][kt][u], 3 / 4 Ro1[CPL][0][kt][u], C0o1[CPL][kt][u]
         (*,3/4F0o1[CPL][kt][u](1+4/3*R[CPL][u])/(1+R[CPL][u])*)}, {u, 0.01, 10},
       PlotLabel → "δb   3/4δr    and   δc    k/keq  ="~StringJoin~ ToString[kt],
       FrameLabel → {"y", ""}]

Out[186]=
```



Velocities

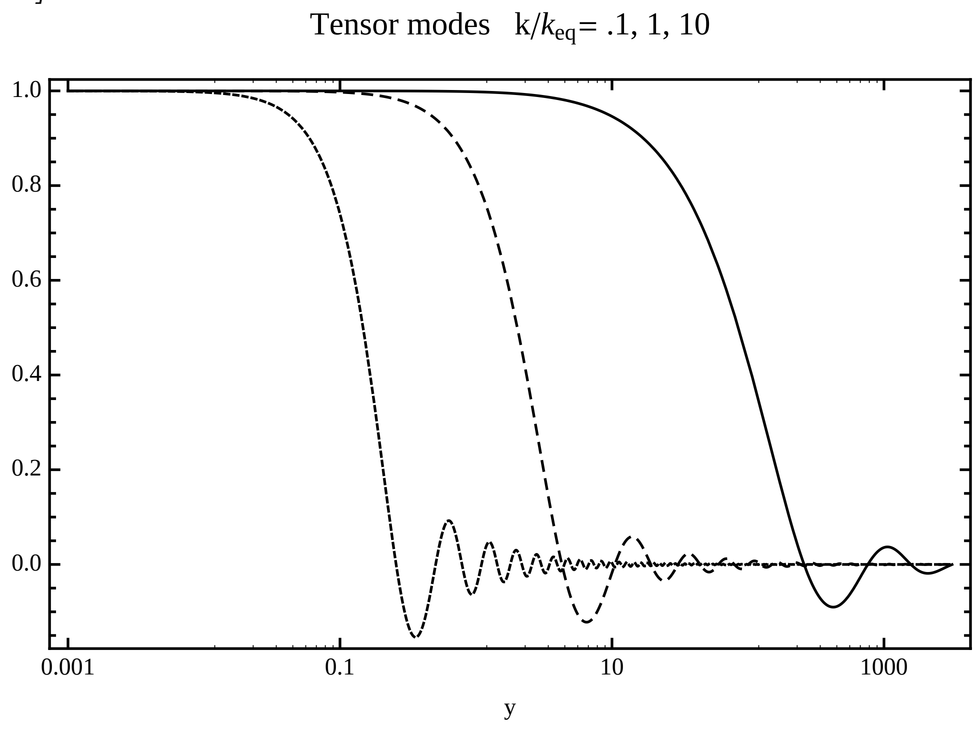
```
In[187]:=
    MyLogLinearPlotBandW[{B1o1[CPL][kt][u], Ro1[CPL][1][kt][u] / 4,
       C1o1[CPL][kt][u] (*,F1o1[CPL][kt][u]*)}, {u, 0.01, 10},
      PlotLabel → "-kV_b  -kV_r  and -kV_c   k/k_eq  ="~StringJoin~ ToString[kt],
      FrameLabel → {"y", ""}]
    On[InterpolatingFunction::"dmval"]
Out[187]=
```



Gravitational waves

You need to use a 'Cosmological Parameters List' which has a non zero tensor to scalar ratio, though what is calculated is only the transfer function of the gravitational waves. We use the default one which is `CPLGW.`

```
In[189]:=
      kt = 1;
      MyLogLinearPlotBandW[{To1[CPLGW][kt / 10][u],
        To1[CPLGW][kt][u], To1[CPLGW][kt * 10][u]}, {u, 0.001, y0[CPLGW]},
       PlotLabel → "Tensor modes   k/keq= .1, 1, 10", FrameLabel → {"y", ""}]

      InterpolatingFunction::dmval :
        Input value {−6.90745} lies outside the range of data in
            the interpolating function. Extrapolation will be used. ≫

      InterpolatingFunction::dmval :
        Input value {−6.90745} lies outside the range of data in
            the interpolating function. Extrapolation will be used. ≫

      InterpolatingFunction::dmval :
        Input value {−6.90745} lies outside the range of data in
            the interpolating function. Extrapolation will be used. ≫

      General::stop : Further output of InterpolatingFunction::dmval
            will be suppressed during this calculation. ≫

Out[190]=
```



```
In[191]:=
      CleanGarbage[CPLGW]

          19 509 320 Bytes have been gained.

(kernel 74) Kernel 74  -88 Bytes have been gained.

(kernel 73) Kernel 73  -88 Bytes have been gained.

(kernel 76) Kernel 76  -88 Bytes have been gained.

(kernel 75) Kernel 75  -112 Bytes have been gained.
```

■ Power spectrum

The gravitational potential power spectrum and the density fluctuations power spectrum are interpolated in k-space for a given y, using respectively the functions 'PΦI' and 'PDI'

```
In[192]:=
      ?P*I

          ▼ CMBquick`CMBquick1`
```

PDI	PΔnI	PΔnNewtI	PΔnSyncI	PΦI

Power spectrum at y = .01, 1, 100, y_0 $\left(\text{remember that } y = a / a_{\text{eq}}\right)$. We can observe how features coming from the damping of sub-Hubble modes during the radiation era disappear since the universe becomes (cold-dark-)matter dominated,

but leave a power spectrum damped on small scales, and that's why we have a change of slope around $k = k_{\text{eq}}$.

```
In[193]:=
     MyLogLogPlotColors[
      {k PΦI[CPL][.01][k], k PΦI[CPL][1][k], k PΦI[CPL][100][k], k PΦI[CPL][y0[CPL]][k]},
      {k, kmin, kmax}, FrameLabel -> {"k/k_eq", ""},
      PlotLabel -> "k P(Φ) at y=.01, 1, 100, y₀"]

        We compute The k-dependent Boltzmann Hierarchy.
          If $ParallelizeBool=True, this is parallelized over CPUs.

        Sources at small k computed

        Sources at large k computed for early times

        Sources at large k computed for late times with radiation truncation
```

InterpolatingFunction::dmval :
Input value {0.01} lies outside the range of data in the
interpolating function. Extrapolation will be used. ≫

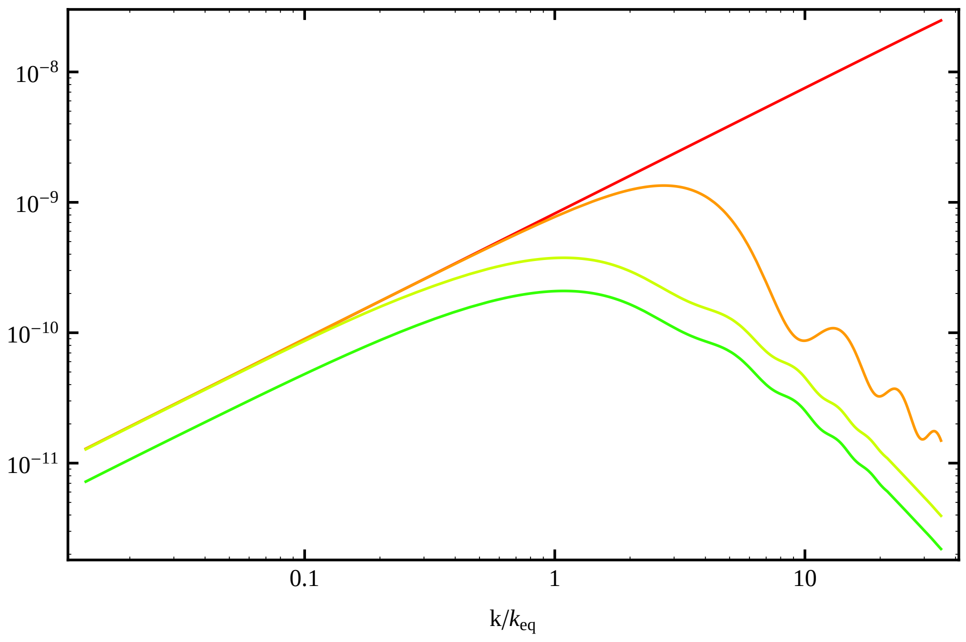
InterpolatingFunction::dmval :
Input value {0.01} lies outside the range of data in the
interpolating function. Extrapolation will be used. ≫

InterpolatingFunction::dmval :
Input value {0.01} lies outside the range of data in the
interpolating function. Extrapolation will be used. ≫

General::stop : Further output of InterpolatingFunction::dmval
will be suppressed during this calculation. ≫

Out[193]=

k P(Φ) at y=.01, 1, 100, y_0



■ Cosmological parameters on the perturbed space-time

You can obtain information on a specific model, including information on the perturbed space-time by typing

```
In[194]:=
     Cosmology[CPL, PerturbationParameters]
```

```
Out[194]=
```

Variable	Value	Units	Comment
Ω_{b0}	0.043969		Abundance of baryons
Ω_{c0}	0.21253		Abundance of CDM
Ω_{r0}	0.000080966		Abundance of radiation (massless ν's and photons)
$\Omega_{\Lambda 0}$	0.74342		Abundance of Λ
Ω_K	0		Abundance of curvature
T_0	2.726	K	Temperature of CMB
N_ν	3.046		Number of massless neutrinos
h	0.719		Reduced Hubble constant
τ_{rei}	0.087		Optical depth of reionization
n_s	0.963		Scalar perturbations spectral index
k_{eq}	0.0096685	Mpc^{-1}	k at equivalence time
z_{rei}	11.045		Redshift at reionization
z_{eq}	3167.		Redshift at equivalence
z_{LSS}	1059.8		Redshift at $\tau-\tau_{rei}=\ln(2)$
z_{dec}	1089.		Redshift at max of visibility function $(\tau' e^{-\tau})$
z_*	1089.6		Redshift at $\tau-\tau_{rei}=1$
$d_A(z_*)$	14108.	Mpc	Angular distance at z_*
$d_A(z_{eq})$	14272.	Mpc	Angular distance at equivalence
D_H	4169.58	Mpc	Hubble distance today
t_0	13.6849	Gyears	Age of the Universe
t_*	380 300.	years	Age of universe at z_*
$r_{hor}(z_{dec})$	285.66	Mpc	Radius of horizon at z_{dec}
η_0	14394.	Mpc	Conformal time today
$A_s{}^2$	2.41×10^{-9}		Primordial scalar perturbations amplitude at k=0.002 Mpc
n_S	0.963		Scalar spectral index
r	0		Tensor to Scalar ratio at k=0.002 Mpc
n_T	1		Tensor spectral index
σ_8	0.79443		Relies on an extrapolation of the matter power spectrum if $k_{max}<200k_{eq}$

We see that at the bottom, there is information about the primordial spectra and about σ_8 in addition to the
standard background information that one would obtain with the option 'BackgroundParameters'.
In order to obtain a precise number for σ_8 you need to set 'kmax' at least to 50.
Otherwise you would only obtain the result to slightly more than 0.1 %

C_l's

■ Initial Conditions

The default initial conditions are adiabatic. It is however possible to use CDM isocurvature initial conditions.

This is set in the parameter ' InitialConditions' which can take either the value 'Adiabatic' or 'CDMIsocurvature'

```
In[195]:=
      InitialConditions

Out[195]=
      Adiabatic
```

■ The Full sky approach

Loading the Bessel functions

In this section we compare the flat sky methods and the full sky method.

Two methods have been implemented in the Full Sky approach. One is referred to as '`FullSky`', and the other one as '`FastFullSky`'.

The first one is an old implementation and will probably disappear. Here we only describe the method '`FastFullSky`'

First we check that there is a file with precomputed Bessel functions. Otherwise we compute it (it takes one night ...) The same command does that.

Bessel Functions Generation and Loading:

```
In[196]:=
      LoadAndGenerateBesselsBinary[lmax]

         Previously, Bessel functions were computed up to lmax=2500

         No need to compute other Bessel functions
```

Choosing the effects

We consider only the SW effect, so our list of effects is (see further for details)

```
In[197]:=
      ELSW = {True, False, False, True, False, False, False};
```

We recall (see the overview) that the order of the values is

```
In[198]:=
      Grid[{{"LSS", "Early times", "Late Times", "Sachs-Wolfe",
         "Integrated Sachs-Wolfe", "Doppler", "Quadrupole (l=2 sources)"}}, Frame → All]
Out[198]=
```

LSS	Early times	Late Times	Sachs-Wolfe	Integrated Sachs-Wolfe	Doppler	Quadrupole (l=2 sources)

And the default value is `EL` which has all the effects.

```
In[199]:=
      EL

Out[199]=
      {True, True, True, True, True, True, True}
```

Computing the multipoles

The functions that we have are

```
In[200]:=
      ? "CMBquick`CMBquick1`Cl*"
```

▼ CMBquick`CMBquick1`

Cl	ClDD	CleanGarbage	ClI	ClRed	ClRedI	ClγD

We select a sublist of l values for which we want to compute the multipoles:

```
In[201]:=
      lmax = 1500;
      Listls = Select[ListlUsedinBesselFunction, # ≤ lmax &];
      lmin = First[ListlUsedinBesselFunction];
```

We can either build the interpolation of the multipoles ourselves

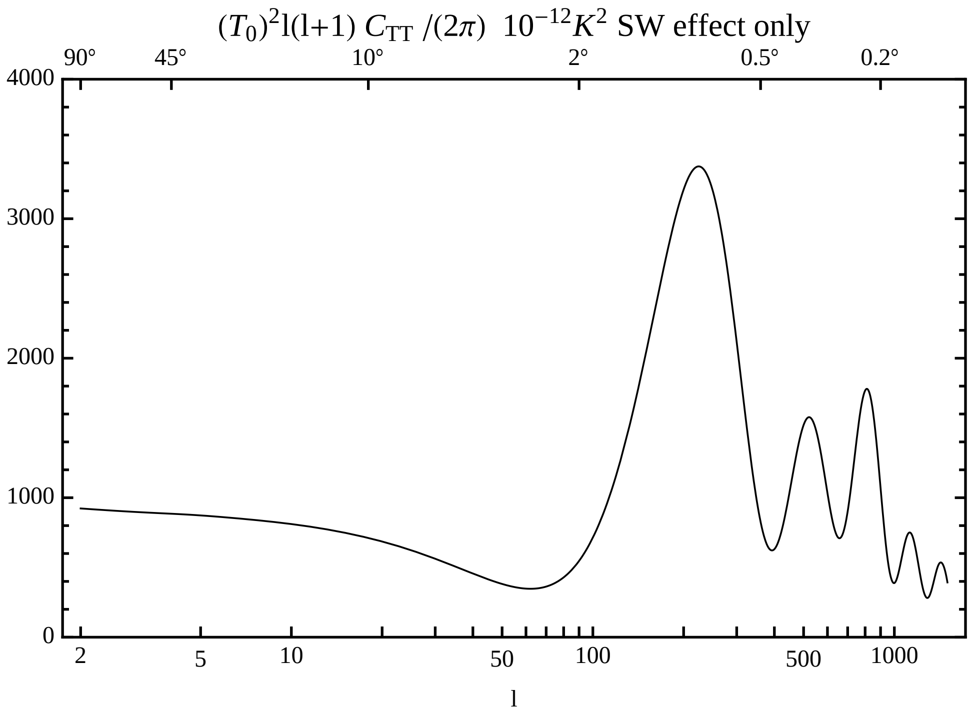
```
In[204]:=
      PreComputeSourcesMultipoles[CPL, ELSW, 0]
      ClfullI = Interpolation@MyParallelize@
          Table[{l, ClRed[CPL, ELSW, FastFullSky, TT, Scalars, UnLensed][l]}, {l, Listls}];

         We now compute the Source multipoles (Slm) computation for temperature

      InterpolatingFunction::dmval :
        Input value {2.47522} lies outside the range of data in
            the interpolating function. Extrapolation will be used. ≫

      InterpolatingFunction::dmval :
        Input value {2.50456} lies outside the range of data in
            the interpolating function. Extrapolation will be used. ≫

      InterpolatingFunction::dmval :
        Input value {2.53403} lies outside the range of data in
            the interpolating function. Extrapolation will be used. ≫

      General::stop : Further output of InterpolatingFunction::dmval
            will be suppressed during this calculation. ≫

         We now compute the Source multipoles (Slm) computation for polarization

         We proceed using parallelization. Consider setting
           $ParallelizeBool=False if your system does not have enough memory.
          If your system freezes because of Memory Shortage,
           consider evaluating 'CloseKernels[]'.
```

Or use CMBquick interpolated C_l's up to a given $l_{\max}$:

```
In[206]:=
      ClfullI2 = ClRedI[CPL, ELSW, FastFullSky, TT, Scalars, UnLensed, lmax];

         We proceed using parallelization. Consider setting
           $ParallelizeBool=False if your system does not have enough memory.
          If your system freezes because of Memory Shortage,
           consider evaluating 'CloseKernels[]'.
```

We plot the results. They don't look like usual CMB because we have only taken into account the SW effect.

```
In[207]:=
      MyLogLinearPlotBandW[{ClfullI2[l](*,ClTTsIanalytic[l]*)}, {l, lmin, lmax},
       PlotStyle → {{Thickness[0.002], GrayLevel[{0}]}}, FrameLabel → {"l", ""},
       PlotLabel -> "(T0)^2 l(l+1) CTT /(2π)  10^-12 K^2 SW effect only",
       FrameTicks → MyTicks, PlotRange → {0, 4000}]
Out[207]=
```



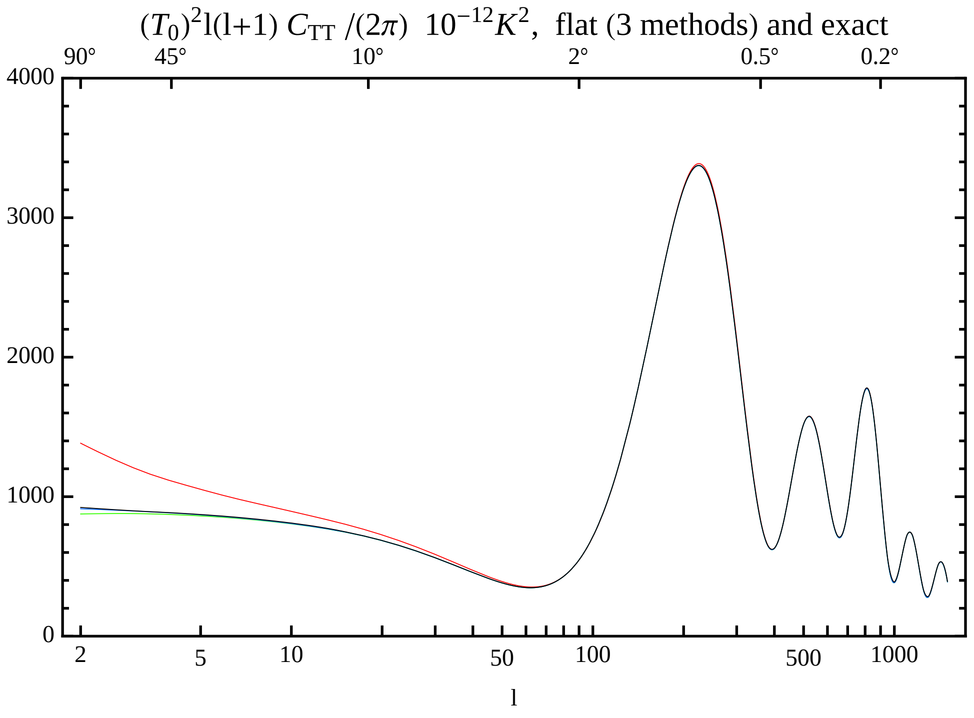
The flat sky approach

In this section we compare the three flat sky methods and the full sky method.

```
In[208]:=
      Clflat3I = Interpolation[
         MyParallelize@Table[{l, ClRed[CPL, ELSW, FlatSky3, TT, Scalars, UnLensed][l]},
           {l, Listls}], Method → "Spline"];
      Clflat2I = Interpolation[MyParallelize@
          Table[{l, ClRed[CPL, ELSW, FlatSky2, TT, Scalars, UnLensed][l]}, {l, Listls}],
         Method → "Spline"];
      Clflat1I = Interpolation[MyParallelize@
          Table[{l, ClRed[CPL, ELSW, FlatSky1, TT, Scalars, UnLensed][l]}, {l, Listls}],
         Method → "Spline"];

         We proceed using parallelization. Consider setting
           $ParallelizeBool=False if your system does not have enough memory.
          If your system freezes because of Memory Shortage,
           consider evaluating 'CloseKernels[]'.
```

Now we plot the results and the relative difference expressed in %

```
In[211]:=
      Off[InterpolatingFunction::"dmval"]
      LogLinearPlot[
       { Clflat1I[l], Clflat2I[l], Clflat3I[l], ClfullI[l](*,ClTTsIanalytic[l]*)},
       {l, lmin, lmax}, PlotStyle → {{Hue[0], Thickness[0.001]},
         {Hue[0.3], Thickness[0.001]}, {Hue[0.6], Thickness[0.001]},
         {Thickness[0.001], GrayLevel[{0}]}}, FrameLabel → {"l", ""},
       PlotLabel -> "(T₀)²l(l+1) C_TT /(2π)  10⁻¹²K²,  flat (3 methods) and exact",
       Frame → True, Axes -> False, FrameTicks → MyTicks, PlotRange → {0, 4000}]

Out[212]=
```



Extended features

■ C_l's generated from tensor modes

See the CMBquick overview above.

■ Massive neutrinos

This has been implemented only for scalar perturbations. You can test it by using the list of parameters

`CPLν`

which, we recall, includes one massive neutrino.

The resulting physical quantities associated with the neutrinos are (they contain in their name `Nm` for Neutrino Mass and `o1` for first order.

```
In[213]:=
     ?"CMBquick`CMBquick1`*Nmo1"
```

▼ CMBquick`CMBquick1`

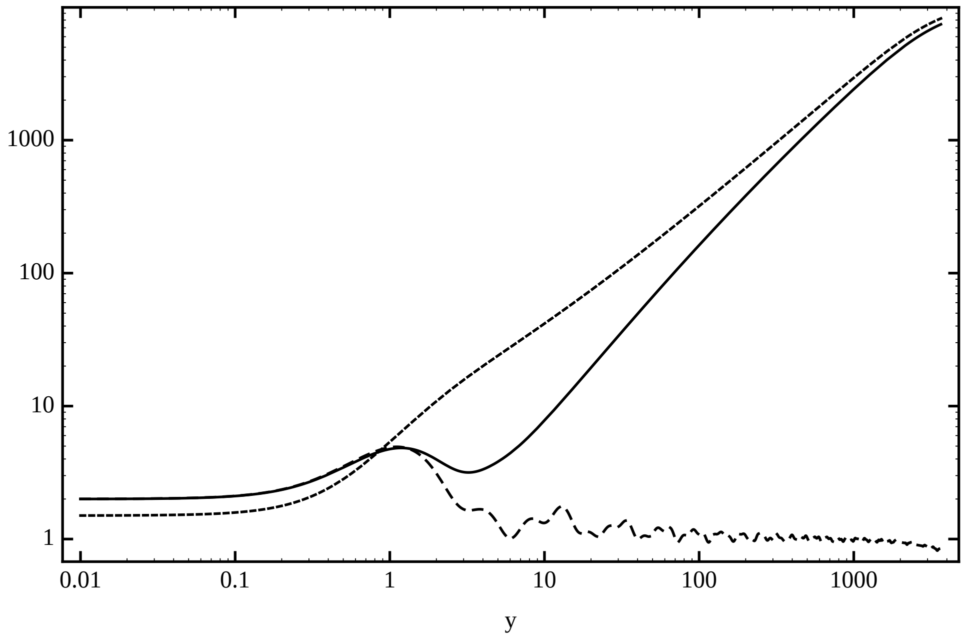
dPNmo1	drhoNmo1	dVNmo1	PiNmo1

And we can check that the massive neutrinos go from a relativistic behaviour to a massive behaviour.

For instance we can plot the densities of massive neutrinos, together with relativistic neutrinos and cdm.

```
In[214]:=
     kt = 3;
     MyLogLogPlotBandW[
      {Abs[drhoNmo1[CPLν][kt][y]], Abs[No1[CPLν][0][kt][y]], Abs[C0o1[CPLν][kt][y]]},
      {y, 0.01, y0[CPLν]}, FrameLabel → {"y", ""},
      PlotLabel → "ρν   ρc   and   ρmν     k/keq="~StringJoin~ToString[kt]]
Out[215]=
```

ρ_ν ρ_c and $\rho_{m\nu}$ k/k_{eq}=3



If we want details about the number and masses of the massive neutrinos in the model:

```
In[216]:=
     Cosmology[CPLν, PerturbationParameters] // AbsoluteTiming

        We compute The k-dependent Boltzmann Hierarchy.
          If $ParallelizeBool=True, this is parallelized over CPUs.

        Sources at small k computed

        Sources at large k computed for early times

        Sources at large k computed for late times with radiation truncation

Out[216]=
```

{1746.571135,

Variable	Value	Units	Comment
Ω_{b0}	0.043969		Abundance of baryons
Ω_{c0}	0.21253		Abundance of CDM
Ω_{r0}	0.000070097		Abundance of radiation (massless ν's a
$\Omega_{\nu m0}$	0.02057		Abundance of massive neutrin
m_ν's	{1}	eV	Masses of massive neutrinos
$\Omega_{\Lambda 0}$	0.72286		Abundance of Λ
Ω_K	0		Abundance of curvature
T_0	2.726	K	Temperature of CMB
N_ν	2.046		Number of massless neutrino
h	0.719		Reduced Hubble constant
τ_{rei}	0.087		Optical depth of reionizati
n_s	0.963		Scalar perturbations spectral i
k_{eq}	0.010846	Mpc^{-1}	k at equivalence time
z_{rei}	11.329		Redshift at reionization
z_{eq}	3658.2		Redshift at equivalence
z_{LSS}	1062.3		Redshift at $\tau-\tau_{rei}=\ln(2)$
z_{dec}	1091.3		Redshift at max of visibility functi
z_*	1092.1		Redshift at $\tau-\tau_{rei}=1$
$d_A(z_*)$	13693.	Mpc	Angular distance at z_*
$d_A(z_{eq})$	13868.	Mpc	Angular distance at equivalen
D_H	4169.58	Mpc	Hubble distance today
t_0	13.3993	Gyears	Age of the Universe
t_*	374324.	years	Age of universe at z_*
$r_{hor}(z_{dec})$	282.86	Mpc	Radius of horizon at z_{dec}
η_0	13975.	Mpc	Conformal time today
A_s^2	2.41×10^{-9}		Primordial scalar perturbations amplitude at k=0.
n_S	0.963		Scalar spectral index
r	0		Tensor to Scalar ratio at k=0.0
n_T	1		Tensor spectral index
σ_8	0.63977		Relies on an extrapolation of the matter power spectrum if k_{ma}

And indeed for m_ν =1eV, the transition between relativistic and non-relativistic behavior occurs when the temperature is around 1eV.

This corresponds to $x=\epsilon_0/1=13.6$ and thus it corresponds to a $y=a/a_{eq}$ given by

```
In[217]:=
     yOFx[CPLν][13.6 / 1]

Out[217]=
     0.859228
```

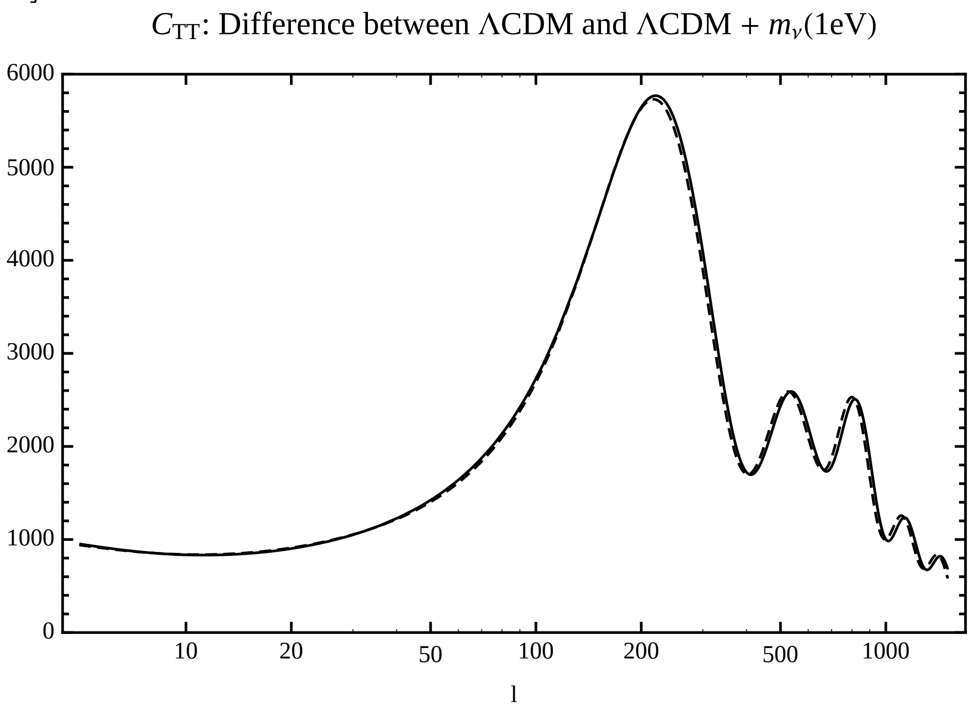
And this is what we observe on the plot, since around that time, massive neutrinos start to collapse and to behave differently from massless neutrinos.

We can compare the percentage of difference between this model with one massive neutrino, and the standard concordance model with (nearly) massless neutrinos.

```
In[218]:=
      PreComputeSourcesMultipoles[CPLν, ELLate, 0]

         We now compute the Source multipoles (Slm) computation for temperature

         We now compute the Source multipoles (Slm) computation for polarization
```

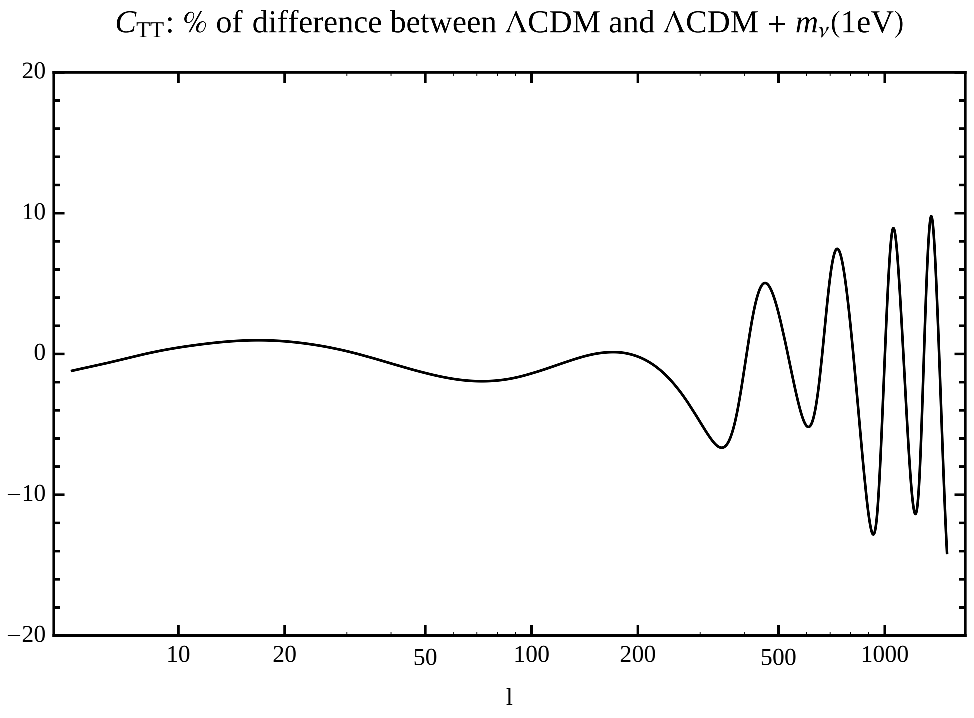
```
In[219]:=
      ClTTfullNeutrinoI =
        Interpolation[Table[{l, ClRed[CPLν, EL, FastFullSky, TT, Scalars, UnLensed][l]},
          {l, Listls}], Method → "Spline"];
      MyLogLinearPlotBandW[{ ClTTfullI[l], ClTTfullNeutrinoI[l]},
       {l, 5, lmax}, FrameLabel → {"l", ""},
       PlotLabel -> "CTT: Difference between ΛCDM and ΛCDM + mν(1eV)",
       PlotRange → {0, 6000}]

         We now compute the Source multipoles (Slm) computation for temperature
```

```
Out[220]=
```



We plot the difference in % in order to see the details of the difference

```
In[221]:=
      ClfullDiffMassiveNeutrinosI = Interpolation[
         Table[{l, 100 * (ClRed[CPLν, EL, FastFullSky, TT, Scalars, UnLensed][l] /
                ClRed[CPL, EL, FastFullSky, TT, Scalars, UnLensed][l] - 1)},
          {l, Listls}], Method → "Spline"];

      MyLogLinearPlotBandW[{ClfullDiffMassiveNeutrinosI[l]},
       {l, 5, lmax}, FrameLabel → {"l", ""},
       PlotLabel -> "C_TT: % of difference between ΛCDM and ΛCDM + m_ν(1eV)",
       PlotRange → {-20, 20}]
Out[222]=
```



```
In[223]:=
      CleanGarbage[CPLν]

         1 835 873 904 Bytes have been gained.
```

■ Primordial reduced bispectrum

Full Sky method for constant f_{NL}

We choose the range of l-values

```
In[224]:=
      lmin = 4;
      lmax = 2000;
      Listls = Select[ListlUsedinBesselFunction, # <= lmax &];
```

We choose to multiply the resulting reduced bispectra by this function in ordet to obtain a result of order unity

```
In[227]:=
      FactorMultiplicatif[l_] := l^2 * (l + 1)^2 / (2 Pi)^2 * 10^16;
```

The function in the case of equilateral bispectum is just bLLL. Otherwise the function is bL1L2L3

```
In[228]:=
      ? CMBquick`CMBquick1`bL*L*
```

▼ CMBquick`CMBquick1`

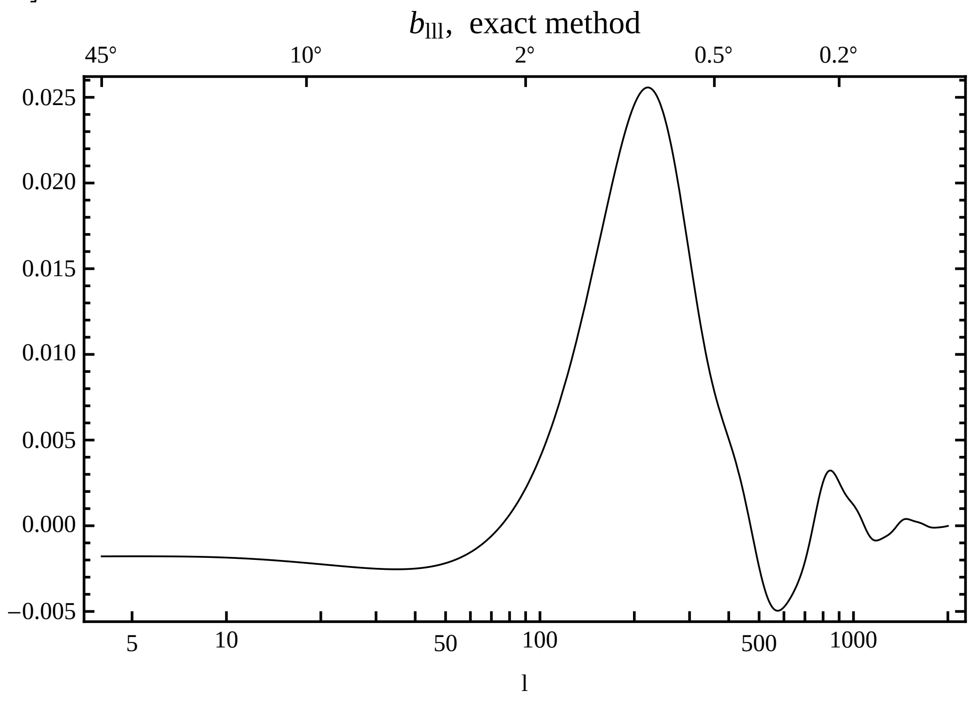
bL1L2L3	bLLL

And we will store the reduced bispectra in a list. So we define the rules:

```
In[229]:=
      Clear[bLLLlist, bLLLlistI]
      bLLLlist[cpl_, el_] :=
       (Table[{l, FactorMultiplicatif[l] bLLL[cpl, el][l]}, {l, Listls}])
      bLLLlistI[cpl_, el_] :=
       bLLLlistI[cpl, el] = Interpolation[bLLLlist[cpl, el], Method -> "Spline"]
```

We then plot the result (note that our sign convention differs from other authors so we multiply the overal result by -1)

```
In[232]:=
      MyLogLinearPlotBandW[{-bLLLlistI[CPL, EL][l]}, {l, lmin, lmax},
       PlotStyle → {{GrayLevel[0], Thickness[0.002]}}, FrameLabel → {"l", ""},
       PlotLabel -> "blll,  exact method", Frame → True, Axes -> False, FrameTicks → MyTicks]

         We now compute the Source multipoles (Slm) computation for temperature

Out[232]=
```



Garbage collection and memory management

This command gets rid of most of stored values for a given ' Cosmological Parameters List'.

This is not working completely, but it should enable you to get rid of most of lost memory and recover a "clean" working space.

We also close the subkernels in case of parallel computing.

```
In[233]:=
      CleanGarbage[CPL]
      CleanGarbage[CPLν]
      CloseKernels[]

         208 530 480 Bytes have been gained.

         424 Bytes have been gained.

Out[235]=
      {}

In[236]:=
      (*ParallelEvaluate[CleanGarbage[CPL];
        CleanGarbage[CPLν];]Useless*)
```

We can also unload the Bessel functions in order to clean additional space with the command

```
In[237]:=
      UnloadBessels[lmax];

         153 697 688 Bytes released from the precomputed Bessel functions.
```

Future Developments

Consult 'ToBeDone[]' for information on what remains to be done.

```
In[238]:=
      ToBeDone[]

            -I might make a break at that point and wait from feedback to
           see where to evolve the package. Several things are unresolved
            -Massive neutrinos in tensors modes (but who cares)
            -Vector mode (is it really relevant?)
            -Accurate lensing using correlation functions. This should be important.
            -Curvature. This is probably the most tricky one. Though
           the main issue is just in Hyperspherical Bessel functions
            -Follow matter temperature as in RECFAST instead
           of using just the radiation temperature.
            -Allow for other types of isocurvature initial conditions,
           and also for mixing of initial conditions types.
            -Put more realistic profiles of reionization.
           This is probably the next thing to do.
            -Write a function to do MCMC exploration.
           This is easy with Hastings' method.
             However, it requires a proper managment of memory in order
           to avoid leakage in the exploration of parameter space.
            -Allow more felixibility in the range of l values spanned.
            -Interpolate the k space logarithmically only
           at low k. The large k's should be linearly interpolated.
            -Allow the user to choose the maximum l used in the
           hierarchy. In particular in the massive ν hierarchy.
            -The equations of the differential Boltzmann-Einstein system should
           be built in a more transparent manner, with replacement rules.
            -Allow for more general background dark energy. This should be
           easy to implement since it only affects the Friedmann equation.
```

■ TIPE (Personal plots)