



Dartmouth

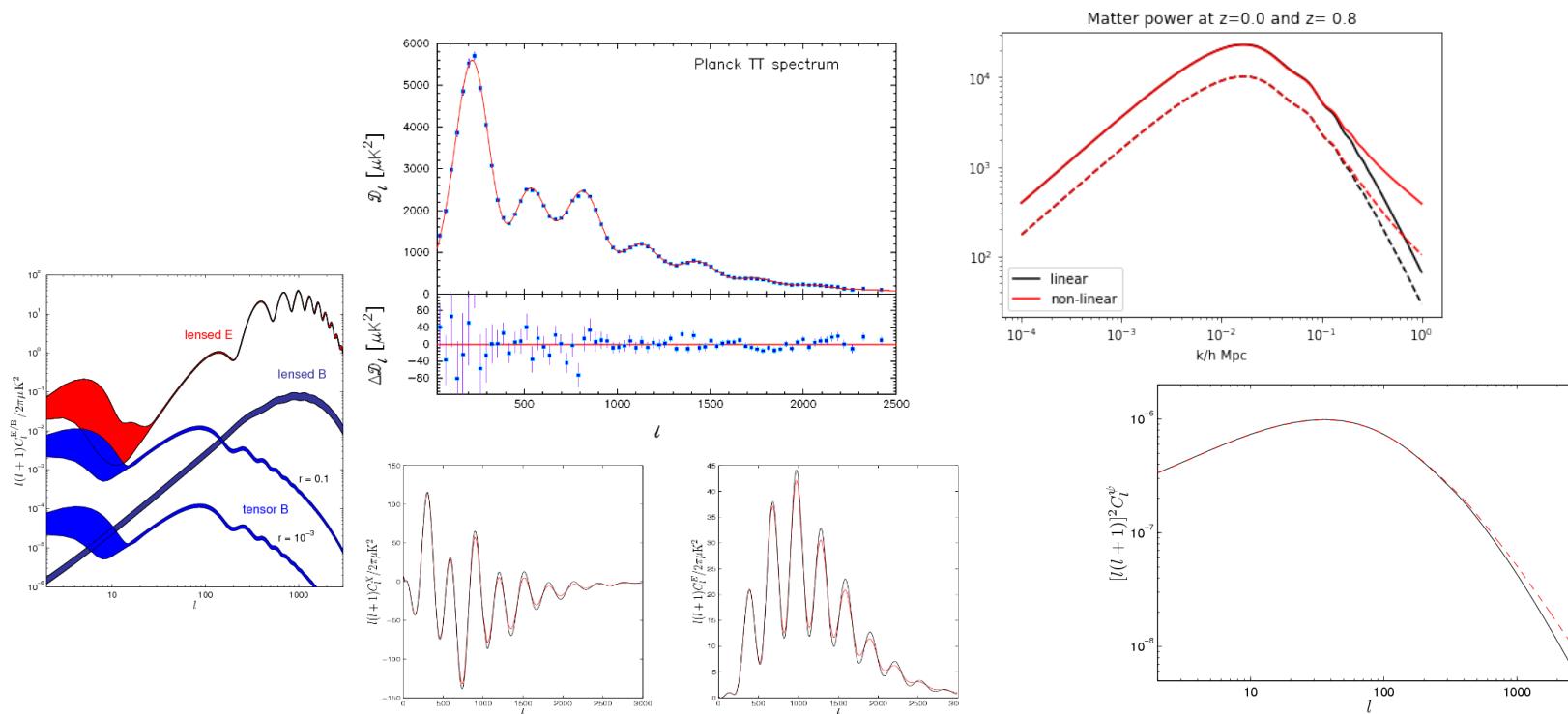


TRIUMF

The lecture will begin shortly. Please mute your microphone until you are ready to speak.

Introduction to CAMB

Code for Anisotropies in the Microwave Background



References

- <http://camb.info/> (main site, register for mailing list and download)
<http://github.com/cmbant/CAMB/> (latest source, fixes and other branches)
- <http://camb.readthedocs.io/en/latest/> (Python CAMB docs, “pip install camb”)
- Help and discussion, and searchable store of previous question answers:
<http://cosmocoffee.info/viewforum.php?f=11>
- Python example notebooks (made with [Jupyter notebook](#))
<http://camb.readthedocs.io/en/latest/CAMBdemo.html>
<http://camb.readthedocs.io/en/latest/ScalEqs.html> (equations and symbolics)
- CAMB notes: <http://cosmologist.info/notes/CAMB.pdf>
(equations and definitions)
- Other lectures/introductions:
http://icg.port.ac.uk/~jschewts/cantata/CAMB/CAMB_lecture.pdf
http://camb.info/Work_with_CAMB_V13_for_AL.pdf

Requirements

Local installation

Python 2.7 or 3.4+

Fortran 90 (gfortran 4.9+ or ifort)

Virtual machines

(Linux with all requirements pre-installed)

<http://cosmologist.info/CosmoBox/>

On Windows can install (unmodified) Python wrapper using “pip install camb” without a working gfortran installation. Otherwise get 64-bit gfortran (<https://gcc.gnu.org/wiki/GFortranBinaries>) or use virtual machine.

Main numerical code is in Fortran, wrapped by Python module. Two main branches on github:

master: uses standard Fortran 90 and will compile on most F90 compilers

devel: is Fortran 2003 (uses classes) and requires gfortran 6+ or ifort 14+

Both branches are maintained. *devel* is a somewhat modernized update that allows things like run-time switching of the dark energy model (also supported in python wrapper). Intended to become the default once gfortran 6 becomes more widespread.

Examples here use the *master* (default) version.

Alternative/cross-check: CLASS <http://class-code.net/>

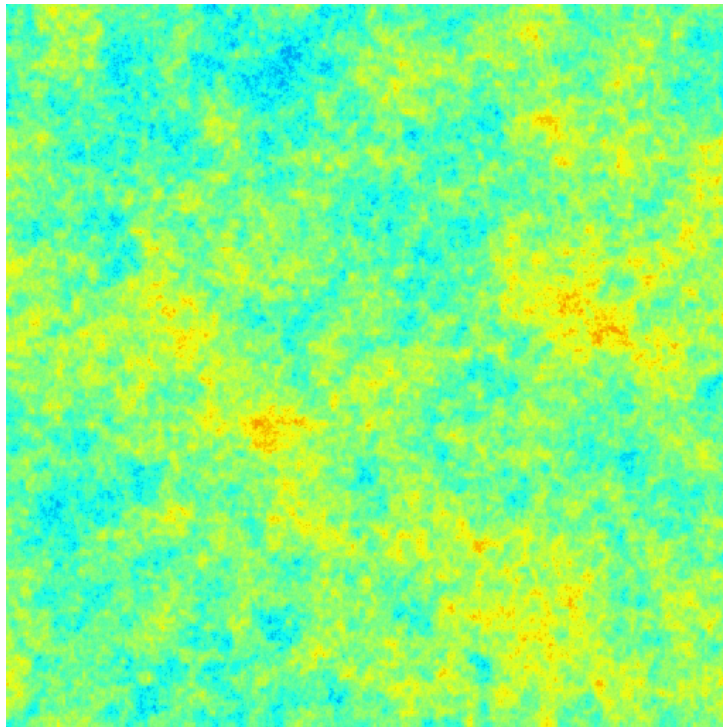
CMB power spectra (C_l , temperature and polarization)

10^{-5} perturbations \Rightarrow Linear theory predictions very accurate

\Rightarrow Fourier modes of perturbations evolve independently

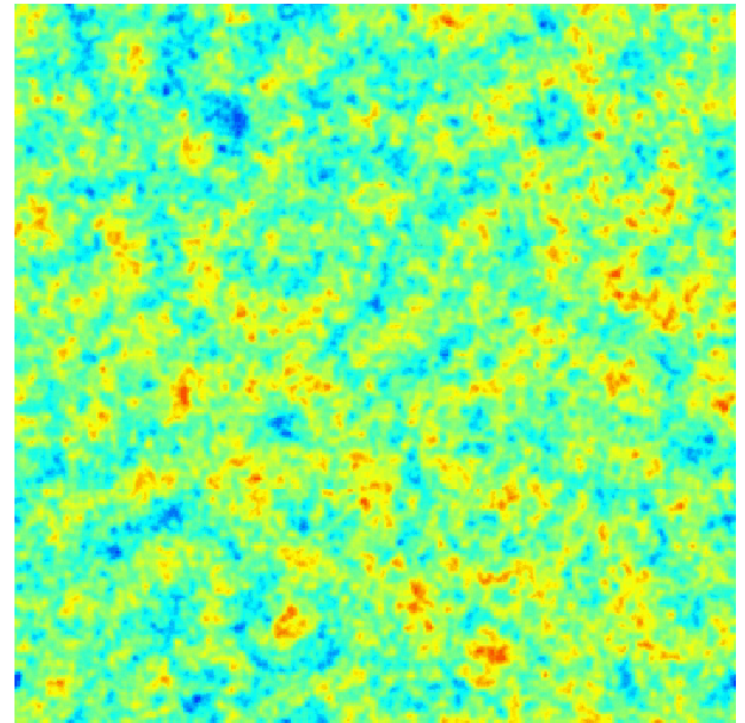
End of inflation (A_s, n_s, n_{run})

Last scattering surface



gravity+
pressure+
diffusion

$(\Omega_b h^2,$
 $\Omega_c h^2 \dots)$



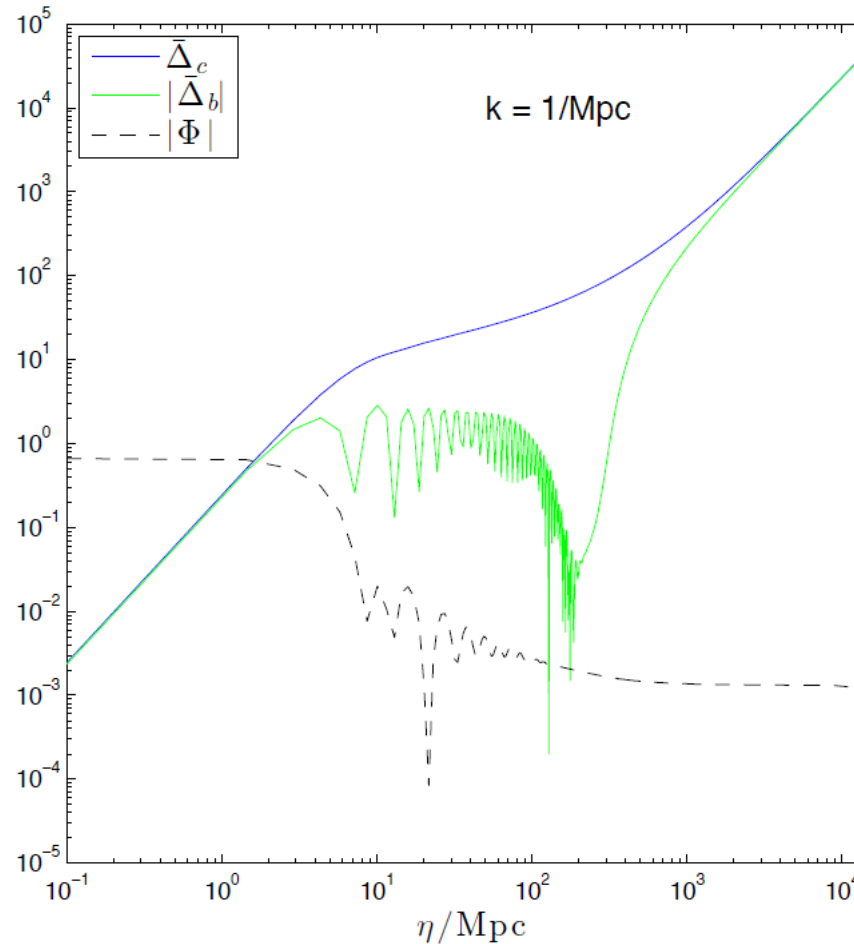
Linear, but:

relativistic (GR), Boltzmann hierarchies for relativistic species (γ, ν), photon polarization

To calculate power spectrum from statistically homogeneous perturbations do *not* need to evolve realisations (unlike in large-scale structure simulations)

$$\text{Linearity: } X(\mathbf{k}, \eta) = X(\mathbf{k}, 0)T(k, \eta)$$

- only need to evolve $T(k, \eta)$, tells you how *all* perturbations with same $|\mathbf{k}|$ evolve



Transfer functions
for each perturbation:

e.g.

$$\Delta_c = \delta\rho_c/\rho_c$$

$$\Delta_b = \delta\rho_b/\rho_b$$

Φ = gravitational potential

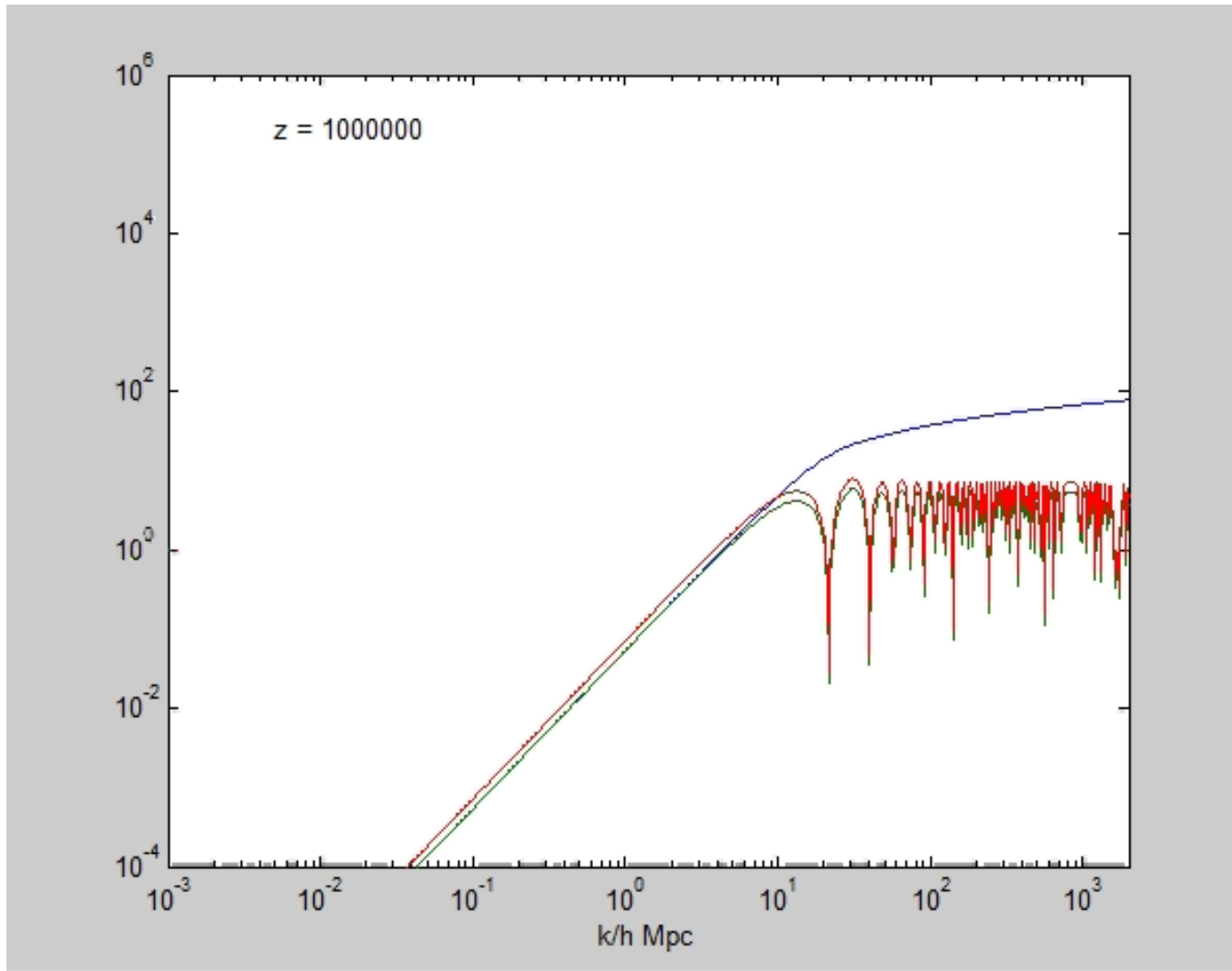
Note: CAMB integrates
transfer functions for variables
defined in the CDM
frame/synchronous gauge.

Outside horizon
No causal contact

Acoustic oscillations
in radiation fluid

Growth of structure

Just need to evolve 1D grid of k values for each species/multipole



Photon, Baryon and dark matter transfer functions

Integration of ~50 differential equations (Boltzmann hierarchy in angular modes l for each species)

- for each decoupled wavenumber mode (few hundred k)
- for each type of perturbation (density, vorticity, gravitational wave) if needed
- from conformal time η when modes well outside horizon (series solution) until after recombination or today

Photon temperature ($0 \leq l \leq 2 - 30$)

$$\underbrace{I'_l + \frac{k}{2l+1} \left[\beta_{l+1}^m \frac{(l+1)^2 - m^2}{l+1} I_{l+1} - l I_{l-1} \right]}_{\text{Coupling of distribution function multipoles}} = \underbrace{-S n_e \sigma_T \left(I_l - \delta_{l0} I_0 - \frac{4}{3} \delta_{l1} v - \frac{2}{15} \zeta \delta_{l2} \right)}_{\substack{\text{Thomson scattering} \\ \gamma + e^- \rightarrow \gamma + e^-}} + \underbrace{\frac{8}{15} k \sigma \delta_{l2} - 4 h' \delta_{l0} - \frac{4}{3} k A \delta_{l1}}_{\text{Gravitational coupling}}$$

Photon polarization

$$\begin{aligned}
 \mathcal{E}_l^{m\pm'} + k \left[\beta_{l+1}^m \frac{(l+3)(l-1)}{(l+1)^3} \frac{(l+1)^2 - m^2}{(2l+1)} \mathcal{E}_{l+1}^{m\pm} - \frac{l}{2l+1} \mathcal{E}_{l-1}^{m\pm} - \frac{2m}{l(l+1)} \sqrt{\beta_0^m} \mathcal{B}_l^{m\mp} \right] &= -S n_e \sigma_T \left(\mathcal{E}_l^{m\pm} - \frac{2}{15} \zeta^{m\pm} \delta_{l2} \right) \\
 \mathcal{B}_l^{m\pm'} + k \left[\beta_{l+1}^m \frac{(l+3)(l-1)}{(l+1)^3} \frac{(l+1)^2 - m^2}{(2l+1)} \mathcal{B}_{l+1}^{m\pm} - \frac{l}{2l+1} \mathcal{B}_{l-1}^{m\pm} + \frac{2m}{l(l+1)} \sqrt{\beta_0^m} \mathcal{E}_l^{m\mp} \right] &= 0.
 \end{aligned} \tag{15}$$

+ neutrinos (no scattering, but massive so energy dependent)

+ Baryons, dark matter, dark energy, ...

+ Gravitational perturbations

Equations are evolved (using Runge-Kutta) and used to compute “sources” $S(\eta, k)$ for each observable which are integrated over time to get the l -space “transfers”:

$$T_{X,l}(k) = \int_0^{\eta_0} j_l(k(\eta_0 - \eta)) S_X(\eta, k)$$

X is T, E, B or ϕ (CMB lensing potential). S_X depends on lowest moments of the distribution functions, e.g. monopole, dipole, quadrupole etc.

Finally adding up all the modes integrated against primordial curvature perturbation power spectrum $P_R(k)$ gives the angular power spectra

$$C_l^{XY} \propto \int d \ln k \ P_R(k) \ T_{X,l}(k) T_{Y,l}(k)$$

The matter power spectrum is simpler

(though not directly observable: function of comoving k at constant time slice)

$$P(k, z) \propto P_R(k) T_\Delta(k, \eta(z))$$

(+ non-linear corrections: in CAMB fit using HALOFIT approximate model)

Δ is usually defined as the total synchronous gauge matter density perturbation.

Using CAMB: Python wrapper

Installation:

Unmodified default code: “pip install camb”

From source: in pycamb folder “python setup.py install”

See sample notebook: <http://camb.readthedocs.io/en/latest/CAMBdemo.html>

Load the module

```
import camb
from camb import model, initialpower
```

Make a **CAMBparams** object instance and set parameters you want

```
#Set up a new set of parameters for CAMB
pars = camb.CAMBparams()
#This function sets up CosmoMC-Like settings, with one massive neutrino and helium set using BBN consistency
pars.set_cosmology(H0=67.5, ombh2=0.022, omch2=0.122, mnu=0.06, omk=0, tau=0.06)
pars.InitPower.set_params(ns=0.965, r=0)
```

Set l_{\max} you want. *lens_potential_accuracy=1* gets non-linear lensing potential.

```
pars.set_for_lmax(2000, lens_potential_accuracy=1)
```

Actually do the calculation and get a results object:

```
results = camb.get_results(pars)
```

Then get things you want from the results object:

```
#get dictionary of CAMB power spectra  
powers = results.get_cmb_power_spectra(pars)  
for name in powers: print name
```

```
total  
lens_potential  
lensed_scalar  
unlensed_scalar  
unlensed_total  
tensor
```

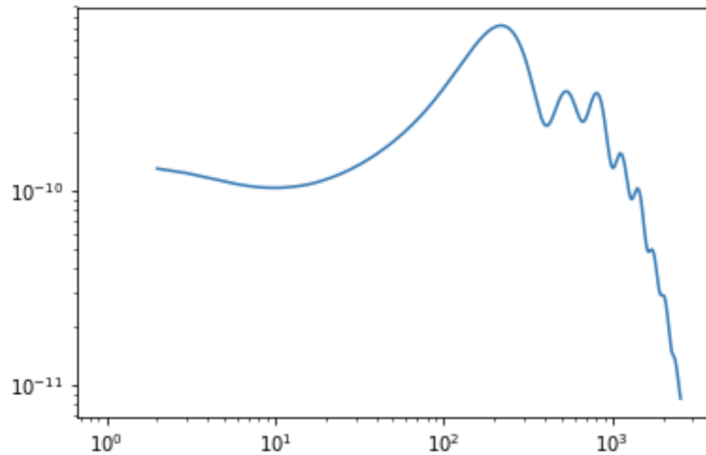
For example the total (lensed scalar+tensor) CMB power spectra are

```
totCL=powers['total']
```

totCL[L, i] is C_L for i=0 (TT), 1 (EE), 2 (BB), 3 (BB)

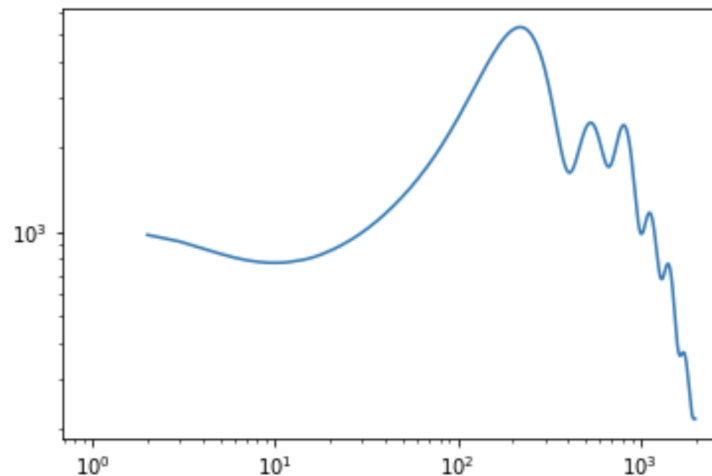
C_L result array by default are $\frac{l(l+1)C_l}{2\pi}$ in dimensionless units (i.e. $\Delta T/T$)

```
plt.loglog(np.arange(totCL.shape[0]), totCL[:,0]);
```



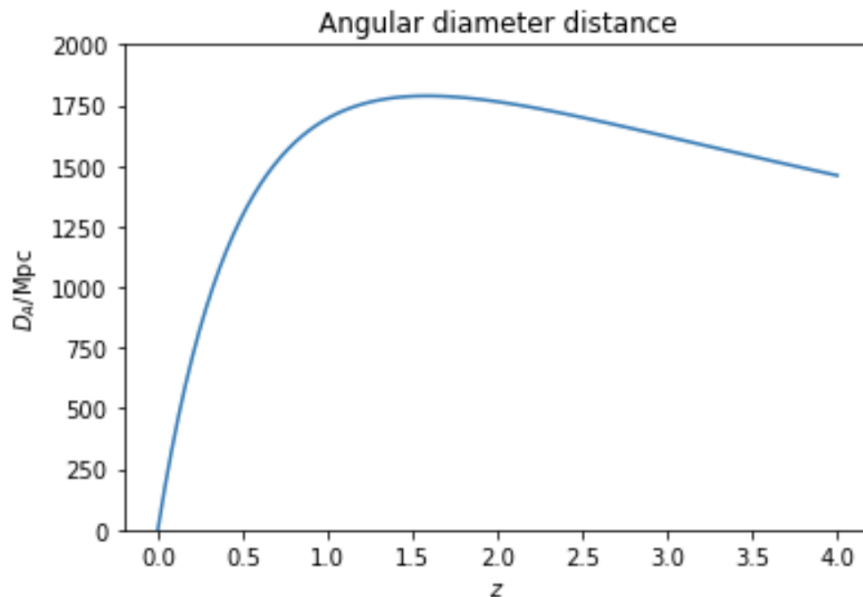
```
lensed_scalar = results.get_lensed_scalar_cls(lmax=2000, CMB_unit = 'muK')  
plt.loglog(np.arange(2001), lensed_scalar[:,0]);
```

Can use other functions
to pull spectrum of interest
and change units.
E.g. in μK^2 units.



Similarly you can get the matter power spectrum, lensing potential spectrum, background functions, derives parameters and more.

```
z = np.linspace(0,4,100)
DA = results.angular_diameter_distance(z)
plt.plot(z, DA)
plt.xlabel('$z$')
plt.ylabel(r'$D_A$ /\rm{Mpc}$')
plt.title('Angular diameter distance')
plt.ylim([0,2000]);
```



See the [example notebook](#) for further examples

camb.symbolic

Python module using **sympy** giving most of the symbolic equations being used, relations between variables, functions to change between gauges and write camb code.

```
from camb.symbolic import *
sympy.init_printing()
print 'CAMB: ', camb.__version__, 'Sympy: ', sympy.__version__
```

CAMB: 0.1.5 Sympy: 1.0

```
display('background_eqs',background_eqs)
display('constraints',constraints)
display('var_subs',var_subs)
display('q_sub',q_sub)
display('pert_eqs',pert_eqs)
display('total_eqs',total_eqs)
#can use tot_eqs as combination of total_eqs + pert_eqs + background_eqs
```

'background_eqs'

$$\left[\frac{d}{dt}a(t) = H(t)a(t), \quad \frac{d}{dt}H(t) = -\frac{\kappa}{6}(3P(t) + \rho(t))a^2(t), \quad \frac{d}{dt}\text{exptau}(t) = \text{visibility}(t) \right]$$

'constraints'

$$\left[Kf_1k^3\phi(t) + \frac{k\kappa}{2}(Kf_1\Pi(t) + \delta(t))a^2(t) + \frac{3\kappa}{2}H(t)a^2(t)q(t), \quad k^2\eta(t) + 2kH(t)z(t) - \kappa a^2(t)\delta(t), \quad \frac{2k^2}{3a^2(t)}(-Kf_1\sigma(t) + z(t)) + \kappa q(t), \right. \\ \left. -\frac{k}{3}z(t) + A(t)H(t) + \dot{h}(t) \right]$$

'var_subs'

$$\left\{ \dot{h}(t) : \frac{1}{6H(t)}(-k^2\eta(t) + \kappa a^2(t)\delta(t) - 6A(t)H^2(t)), \quad \phi(t) : -\frac{\kappa a^2(t)}{2Kf_1k^3}(Kf_1k\Pi(t) + k\delta(t) + 3H(t)q(t)), \right. \\ \left. \sigma(t) : \frac{1}{2Kf_1k^2H(t)}(k(-k^2\eta(t) + \kappa a^2(t)\delta(t)) + 3\kappa H(t)a^2(t)q(t)), \quad z(t) : \frac{1}{2kH(t)}(-k^2\eta(t) + \kappa a^2(t)\delta(t)) \right\}$$

'q_sub'

CAMB uses covariant perturbation variables in the 3+1 formulation.

`camb.symbolic` defines equations that are valid in any gauge. They can be written in specific gauges by specific choices of restrictions on variables.

```
#Fluid components
display('density_eqs',density_eqs)
display('delta_eqs', delta_eqs)
display('vel_eqs',vel_eqs)
#can use component_eqs as combination of density_eqs + delta_eqs + vel_eqs
```

'density_eqs'

$$\left[\frac{d}{dt} \rho_b(t) = -3 (p_b(t) + \rho_b(t)) H(t), \quad \frac{d}{dt} \rho_c(t) = -3 H(t) \rho_c(t), \quad \frac{d}{dt} \rho_g(t) = -4 H(t) \rho_g(t), \quad \frac{d}{dt} \rho_r(t) = -4 H(t) \rho_r(t), \right. \\ \left. \frac{d}{dt} \rho_\nu(t) = -3 (p_\nu(t) + \rho_\nu(t)) H(t), \quad \frac{d}{dt} \rho_{de}(t) = -3 (w_{de}(t) + 1) H(t) \rho_{de}(t) \right]$$

'delta_eqs'

$$\left[\frac{d}{dt} \Delta_r(t) = -k q_r(t) - 4\dot{h}(t), \quad \frac{d}{dt} \Delta_g(t) = -k q_g(t) - 4\dot{h}(t), \right. \\ \left. \frac{d}{dt} \Delta_b(t) = \left(k v_b(t) + 3\dot{h}(t) \right) \left(-\frac{p_b(t)}{\rho_b(t)} - 1 \right) + \left(-3 c_{sb}^2(t) + \frac{3 p_b(t)}{\rho_b(t)} \right) \Delta_b(t) H(t), \quad \frac{d}{dt} \Delta_c(t) = -k v_c(t) - 3\dot{h}(t), \right. \\ \left. \dots \right]$$

...

You can easily convert between Newtonian gauge, synchronous/CDM frame results, and other possible frame choices.

```
#e.g. can check we recover standard Newtonian gauge equations
#(note all equations above are valid in any frame)
newtonian_gauge(delta_eqs)
```

$$\begin{aligned} \left[\frac{d}{dt} \Delta_r(t) = -k_{\text{qr}}(t) + 4 \frac{d}{dt} \Phi_N(t), \quad \frac{d}{dt} \Delta_g(t) = -k_{\text{qg}}(t) + 4 \frac{d}{dt} \Phi_N(t), \right. \\ \left. \frac{d}{dt} \Delta_b(t) = -\frac{1}{\rho_b(t)} \left(\left(k_{\text{vb}}(t) - 3 \frac{d}{dt} \Phi_N(t) \right) (p_b(t) + \rho_b(t)) + 3 (c_{\text{sb}}^2(t) \rho_b(t) - p_b(t)) \Delta_b(t) H(t) \right), \right. \\ \left. \frac{d}{dt} \Delta_c(t) = -k_{\text{vc}}(t) + 3 \frac{d}{dt} \Phi_N(t), \right. \end{aligned}$$

camb_fortran can
convert symbolic
expressions into
CAMB-variable
Fortran source code

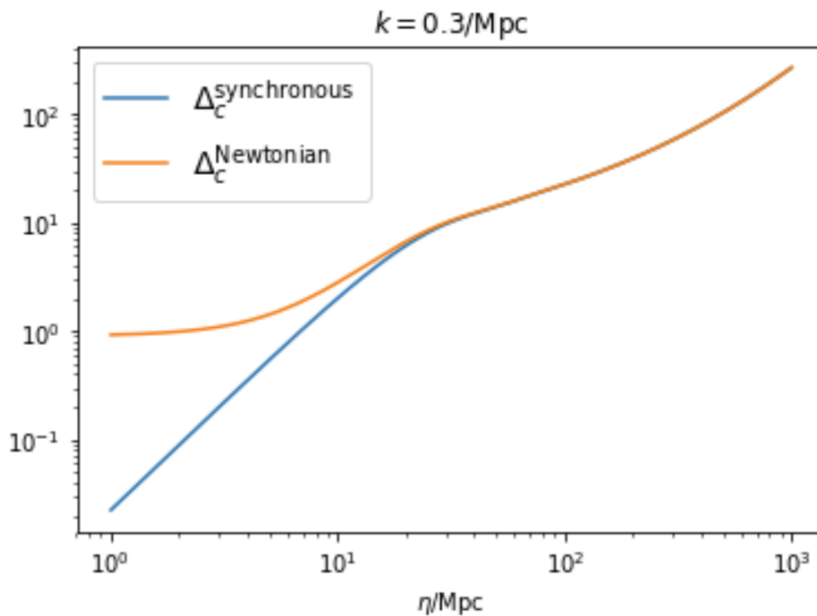
```
#These are definitions used in CAMB to get the various sources for the temperature
```

```
print camb_fortran(dphi, 'phidot')
print camb_fortran(dsigma, 'sigmadot')
print camb_fortran(diff(polter_t,t), 'polterdot')
print camb_fortran(polterddot, 'polterddot')
print camb_fortran(2*diff(phi,t)*exptau, 'ISW')
print camb_fortran(monopole_source, 'monopole_source')
print camb_fortran(doppler, 'doppler')
print camb_fortran(quadrupole_source, 'quadrupole_source')

phidot = (1.0d0/2.0d0)*(-adotoa*dgpi - 2*adotoa*k**2*phi + dgq*k &
    -diff_rhopi + k*sigma*(gpres + grho))/k**2
sigmadot = -adotoa*sigma - 1.0d0/2.0d0*dgpi/k + k*phi
polterdot = (1.0d0/10.0d0)*pigdot + (3.0d0/5.0d0)*Edot(2)
polterddot = -2.0d0/25.0d0*adotoa*dgq/(k*Kf(1)) - &
    4.0d0/75.0d0*adotoa*k*sigma - 4.0d0/75.0d0*dgpi - &
    2.0d0/75.0d0*dgrho/Kf(1) + dopacity*(-1.0d0/10.0d0*pig + &
    (7.0d0/10.0d0)*polter - 3.0d0/5.0d0*E(2)) &
    -3.0d0/50.0d0*k*octgdot*Kf(2) + (1.0d0/25.0d0)*k*qgdot - &
    1.0d0/5.0d0*k*Edot(3)*Kf(2) + opacity*(-1.0d0/10.0d0*pigdot + &
    (7.0d0/10.0d0)*polterdot - 3.0d0/5.0d0*Edot(2))
ISW = 2*exptau*phidot
monopole_source = (1.0d0/4.0d0)*visibility*(-4*etak + k*(clxg + &
    8*phi)*Kf(1))/(k*Kf(1))
doppler = (dvisibility*(sigma + vb) + visibility*(sigmadot + vbdot))/k
quadrupole_source = (5.0d0/8.0d0)*(3*ddvisibility*polter + &
    6*dvisibility*polterdot + visibility*(k**2*polter + &
    3*polterddot))/k**2
```

You can also pass symbolic expressions to some camb functions to get numerical results directly (behind the scenes it converts to Fortran, compiles, loads and then calls it from the main code)

```
data = camb.get_transfer_functions(pars)
#For example, this plots the Newtonian gauge density compared to the synchronous gauge one
import camb.symbolic as cs
Delta_c_N = cs.make_frame_invariant(cs.Delta_c, 'Newtonian')
ev=data.get_time_evolution(k, eta, ['delta_cdm',Delta_c_N])
plt.figure(figsize=(6,4))
plt.loglog(eta,ev[:,0])
plt.loglog(eta,ev[:,1])
plt.title(r'$k= %s/\rm{Mpc}$'%k)
plt.xlabel(r'$\eta/\rm{Mpc}$');
plt.legend([r'$\Delta_c^{\rm synchronous}$', r'$\Delta_c^{\rm Newtonian}$'], fontsize=14);
```



- CosmoMC

[http://icg.port.ac.uk/~jschewts/cantata/CAMB/
CosmoMC_lecture.pdf](http://icg.port.ac.uk/~jschewts/cantata/CAMB/CosmoMC_lecture.pdf)