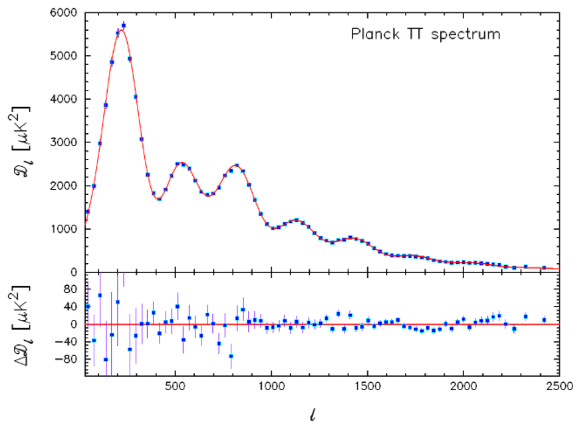
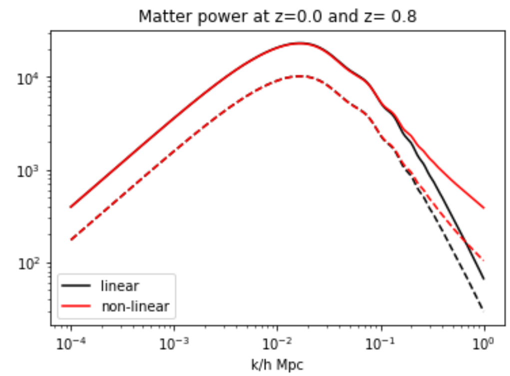
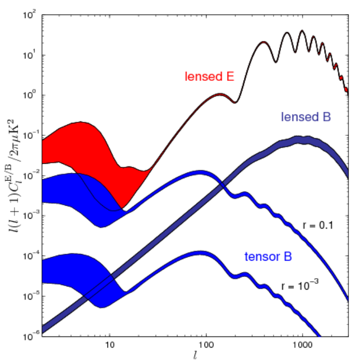
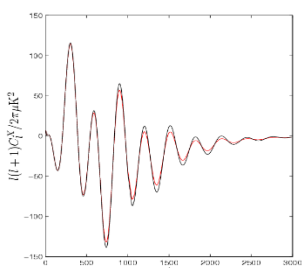
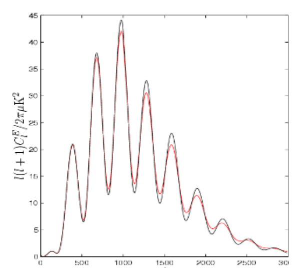
CMB Codes

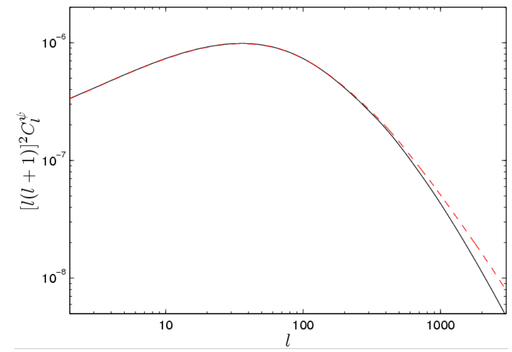










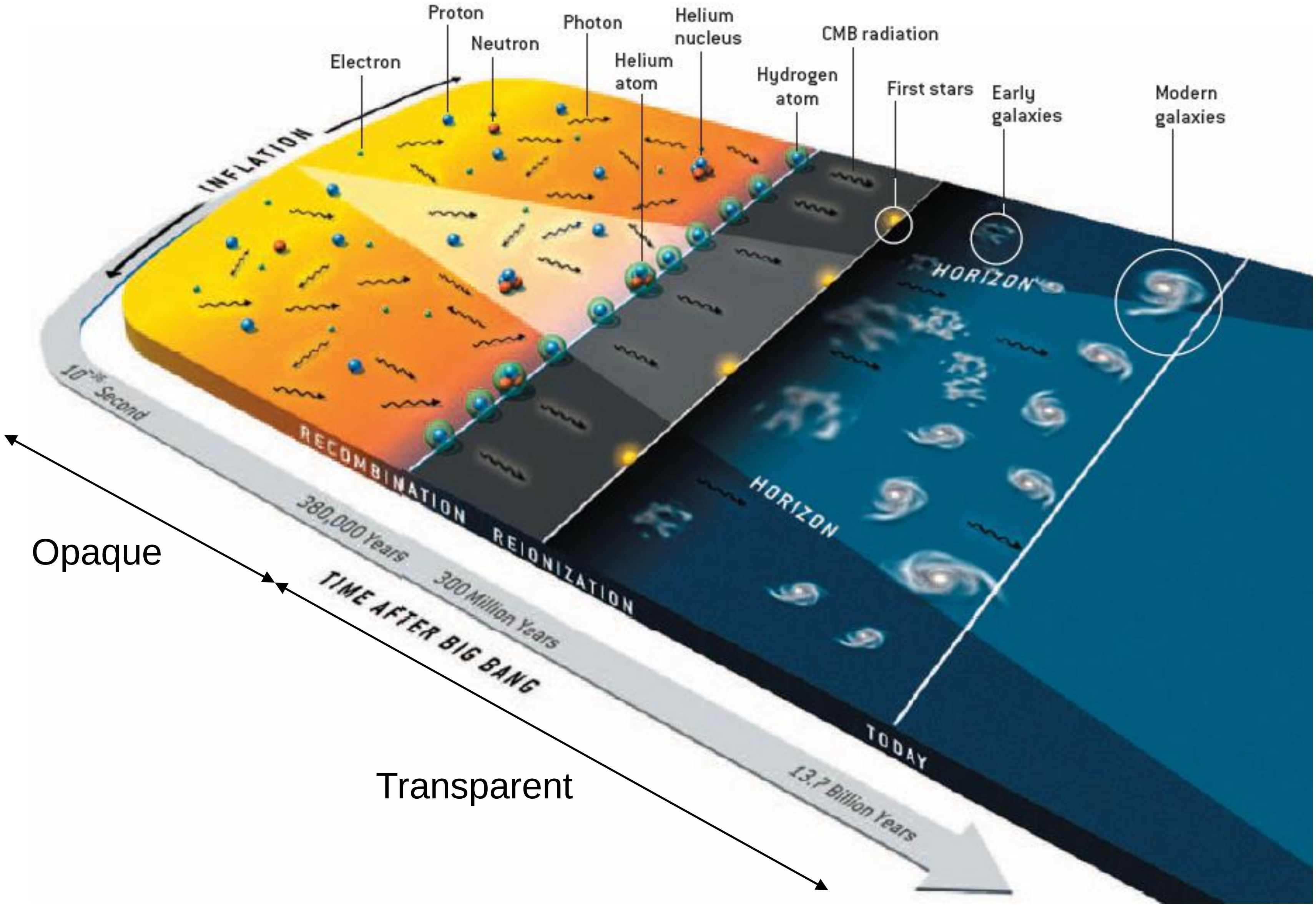


UNIVERSITY OF SUSSEX

Antony Lewis
https://cosmologist.info/

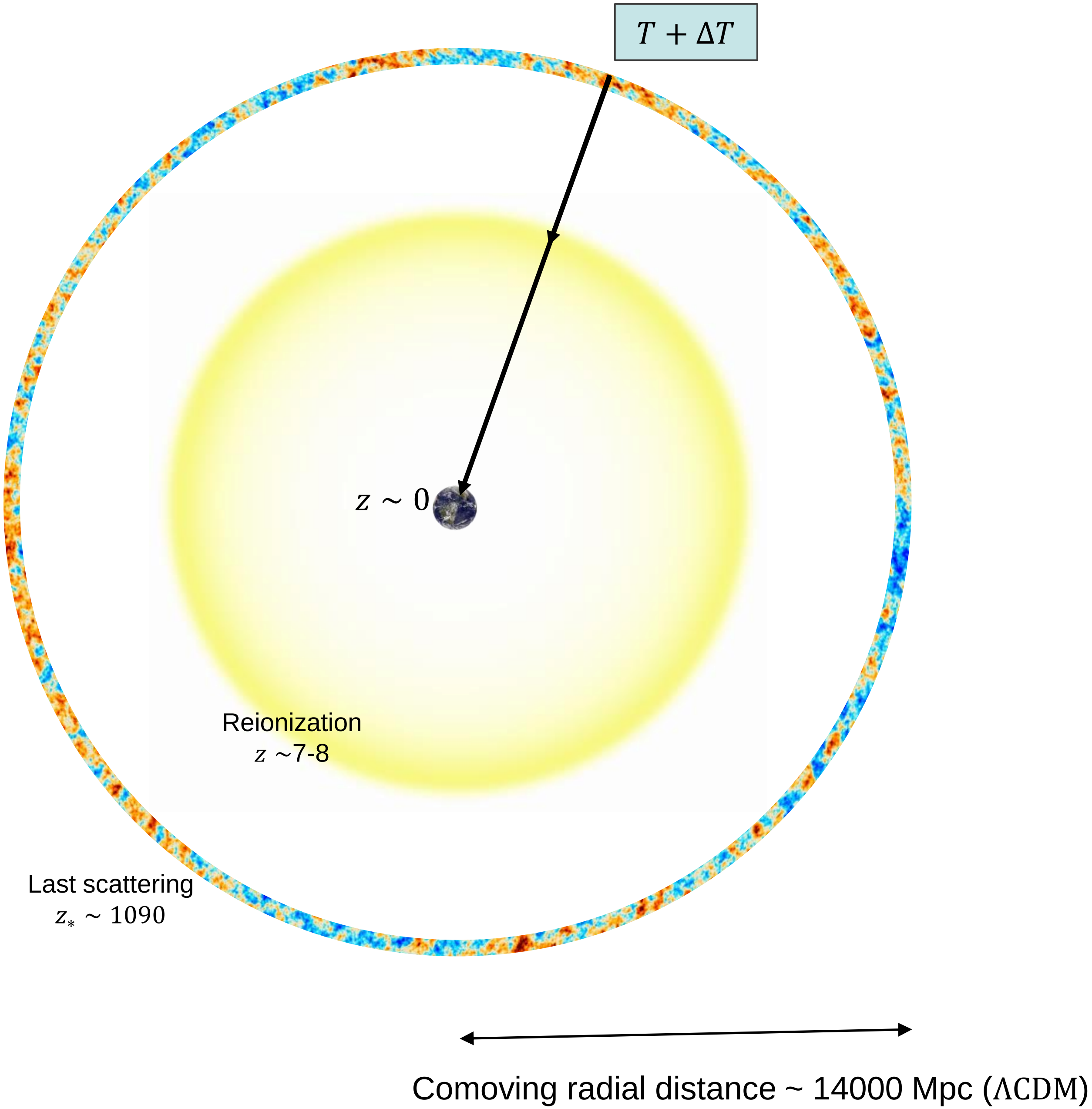
UKRI Science and Technology Facilities Council

Evolution in the standard cosmology



Hu & White, Sci. Am., 290 44 (2004)

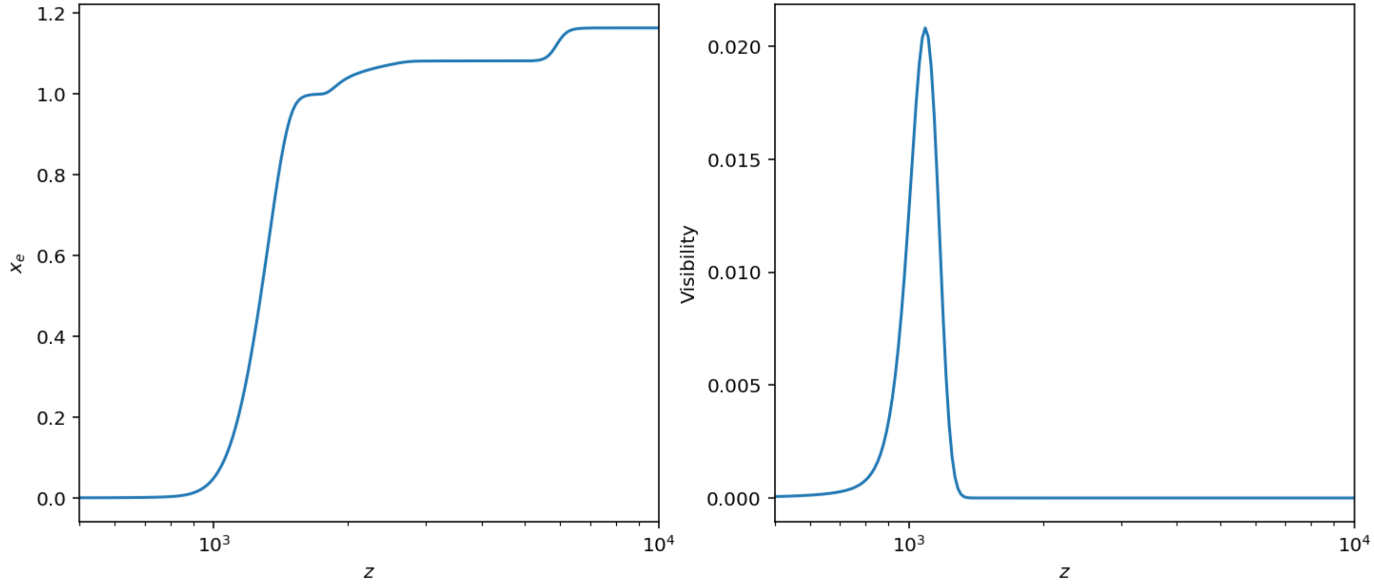
In comoving distance



Background Codes

Ionization/thermal history $(x_e(z), T_b(z))$

$x_e = \mathrm{n_e}/\mathrm{n_H}$ ionization fraction



RECFAST (Seager et al)

https://www.astro.ubc.ca/people/scott/recfast.html

fast/simple approximate x_e/T_b-only code

RECFAST three-level atom model

$$\frac{dx_{\rm p}}{dz} = \left(x_{\rm e}x_{\rm p}n_{\rm H}\alpha_{\rm H} - \beta_{\rm H}(1-x_{\rm p}){\rm e}^{-h\nu_{H2s}/kT_{\rm M}}\right) \times \frac{(1+K_{\rm H}\Lambda_{\rm H}n_{\rm H}(1-x_{\rm p}))}{H(z)(1+z)(1+K_{\rm H}(\Lambda_{\rm H}+\beta_{\rm H})n_{\rm H}(1-x_{\rm p}))}, \qquad (1)$$

$$\frac{dx_{\rm HeII}}{dz} = \left(x_{\rm HeII}x_{\rm e}n_{\rm H}\alpha_{\rm HeI} - \beta_{\rm HeI}(f_{\rm He}-x_{\rm HeII}){\rm e}^{-h\nu_{HeI2^1s}/kT_{\rm M}}\right) \times \frac{(1+K_{\rm HeI}\Lambda_{\rm He}n_{\rm H}(f_{\rm He}-x_{\rm HeII}){\rm e}^{-h\nu_{ps}/kT_{\rm M}}))}{H(z)(1+z)\left(1+K_{\rm HeI}(\Lambda_{\rm He}+\beta_{\rm HeI})n_{\rm H}(f_{\rm He}-x_{\rm HeII}){\rm e}^{-h\nu_{ps}/kT_{\rm M}}\right)}, \qquad (2)$$

$$\alpha_{\rm H} = F10^{-19}\frac{at^b}{1+ct^d}\,{\rm m}^3{\rm s}^{-1}, \qquad (3)$$

$$\alpha_{\rm HeI} = q\left[\sqrt{\frac{T_{\rm M}}{T_2}}\left(1+\sqrt{\frac{T_{\rm M}}{T_2}}\right)^{1-p}\left(1+\sqrt{\frac{T_{\rm M}}{T_1}}\right)^{1+p}\right]^{-1}{\rm m}^3{\rm s}^{-1}, \qquad (4)$$

$$\frac{dT_{\rm M}}{dz} = \frac{8\sigma_{\rm T}a_{\rm R}T_{\rm R}^4}{3H(z)(1+z)m_{\rm e}c}\,\frac{x_{\rm e}}{1+f_{\rm He}+x_{\rm e}}\,(T_{\rm M}-T_{\rm R}) + \frac{2T_{\rm M}}{(1+z)}.$$

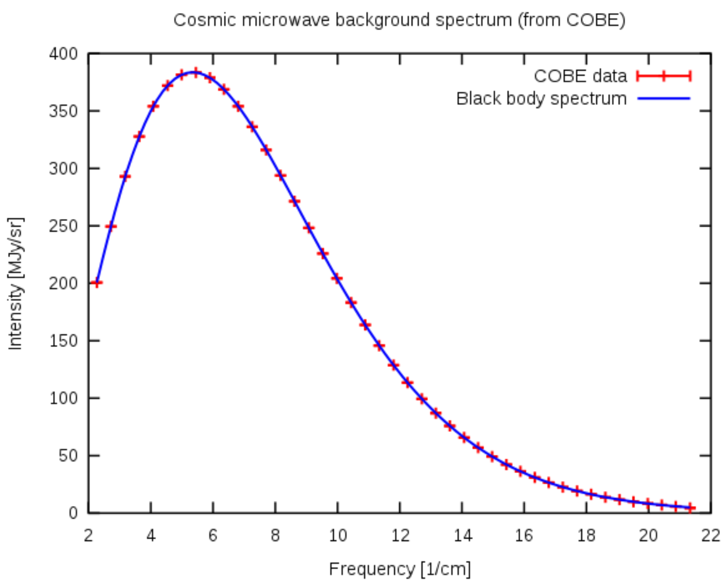
astro-ph/9909275

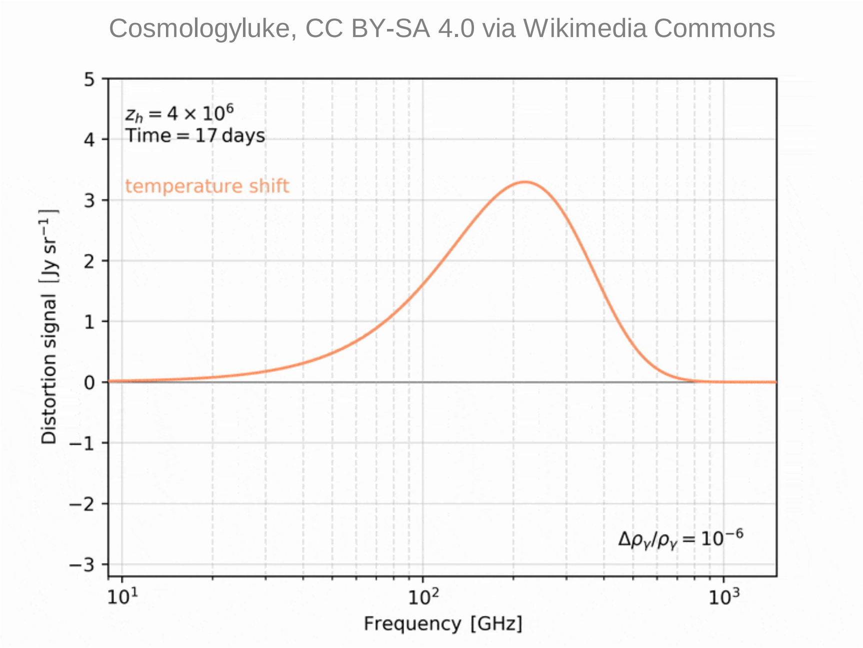
Full distribution function + multi-level atom + smart approximations

CosmoRec (Chluba et al)
http://www.jb.man.ac.uk/~jchluba/Science/CosmoRec/Welcome.html
HyRec (Ali-Haimoud, Hirata et al)
https://github.com/nanoomlee/HYREC-2

Beyond blackbody: the full (monopole) spectral distortion spectrum

- μ, y and other mixed distortions from energy injection
- spectral features from recombination

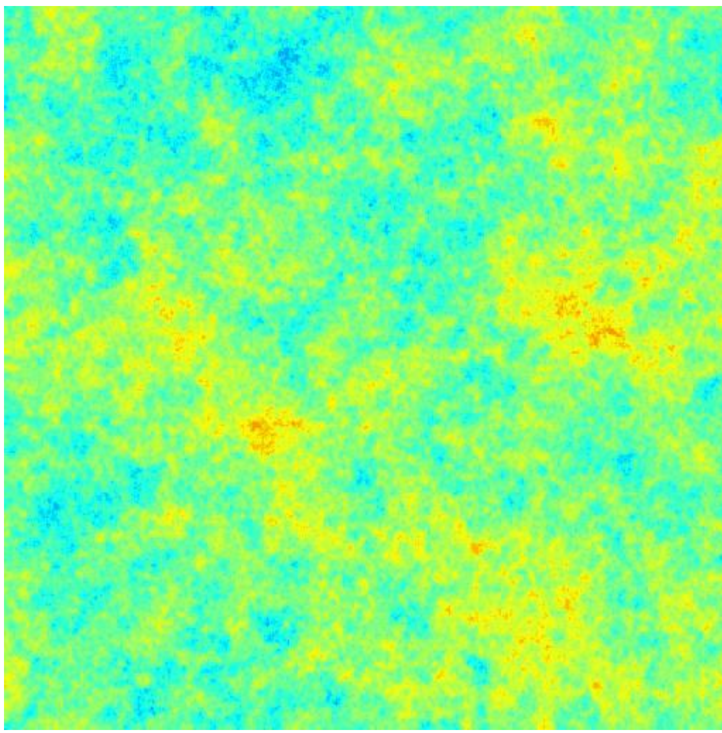




Anisotropies - CMB power spectra (C_l, temperature and polarization)

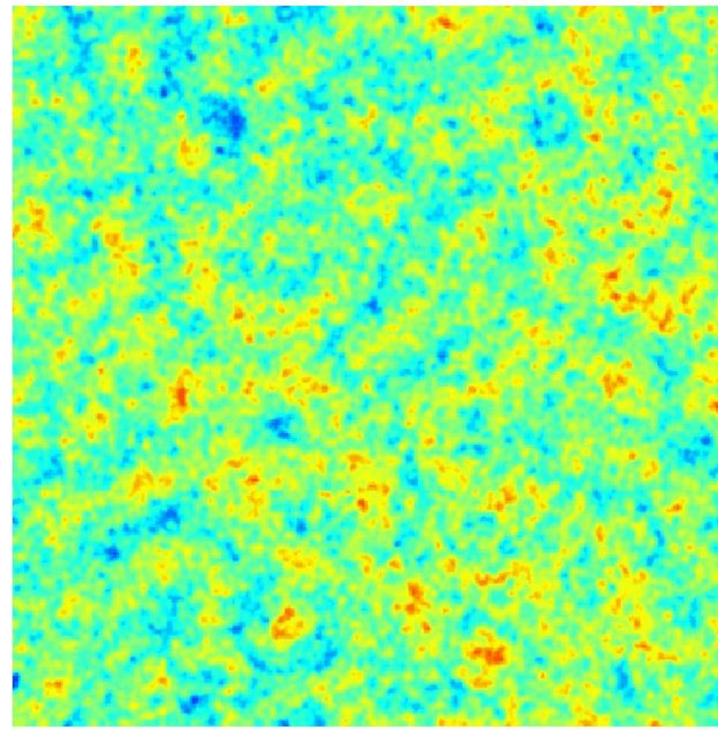
End of inflation $P_R(k) \approx A_s(k/k_0)^{n_s-1}$,

Last scattering surface

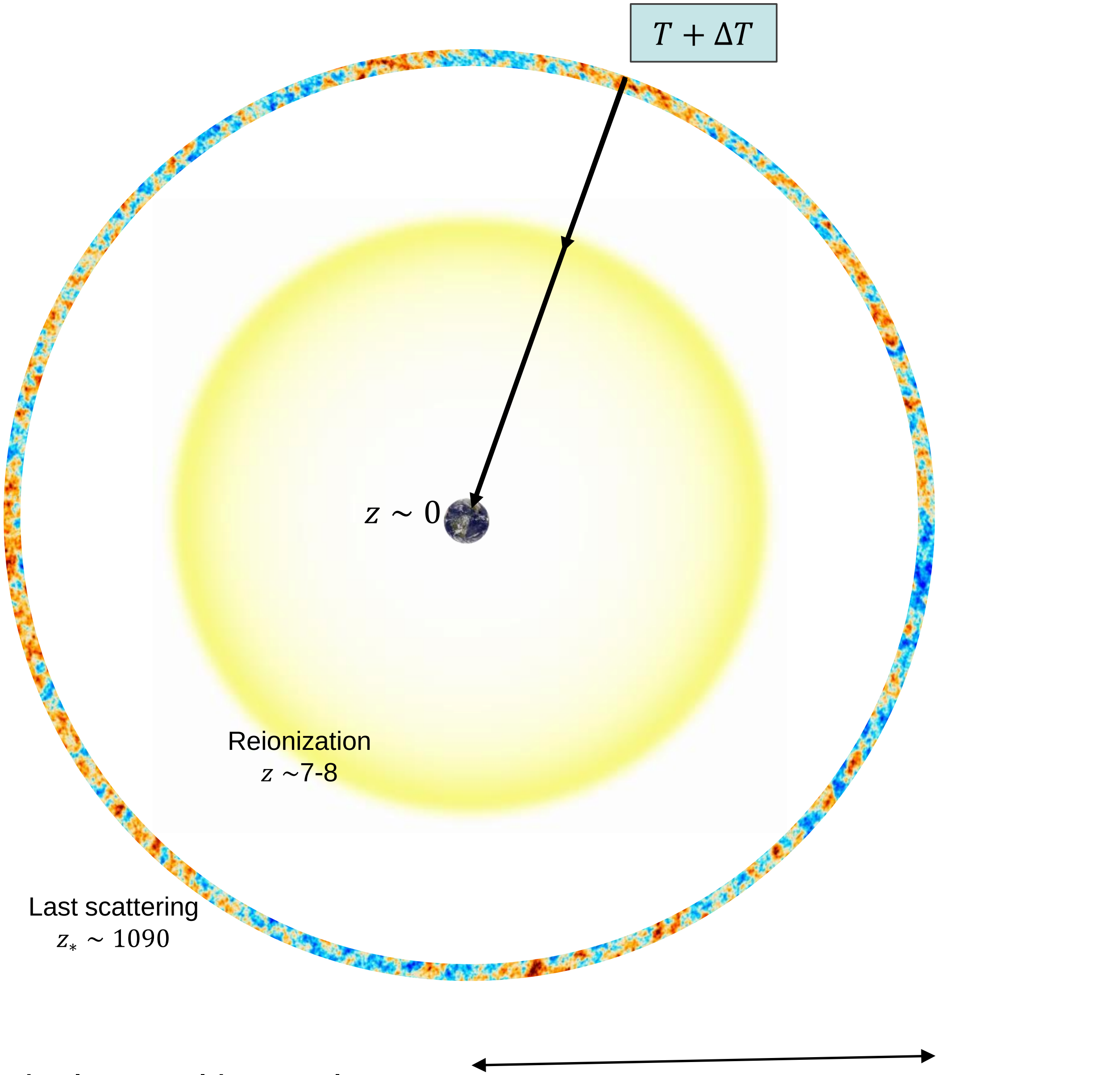


gravity+
pressure+
diffusion

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



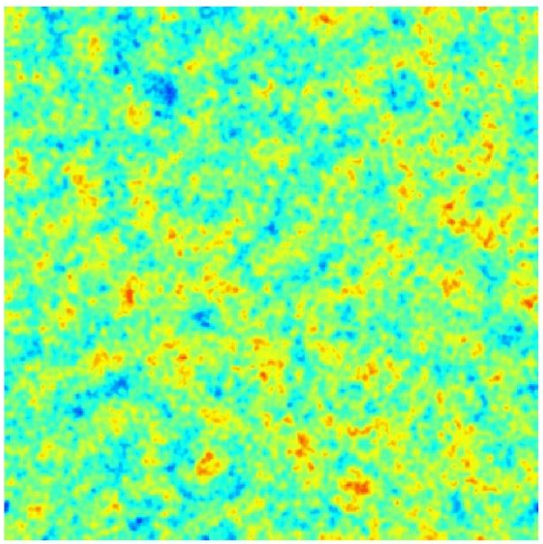
In comoving distance



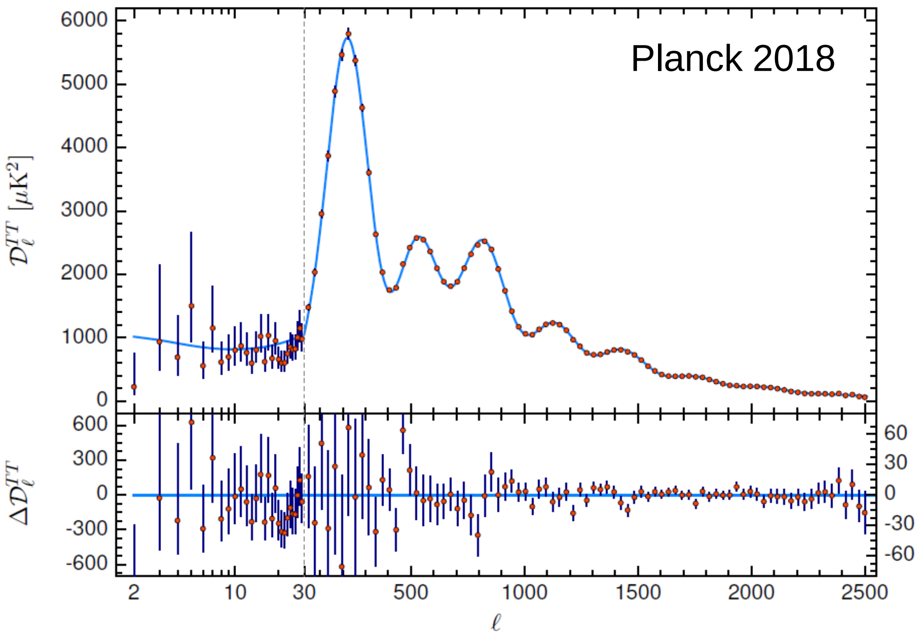
Angular projection spherical Bessel integral
+ integral over source visibility

CMB power spectrum

Perturbations at last scattering



arXiv:1807.06209



Assume initial power spectrum + model parameters

Calculate theory C_l

test with data

Calculation of theoretical perturbation evolution

Perturbations $O(10^{-5})$

Simple linearized equations are very accurate (except small scales)

Can use real or Fourier space

Fourier modes evolve independently: simple to calculate accurately

Physics Ingredients

•Thomson scattering (non-relativistic electron-photon scattering)
  - tightly coupled before recombination: ‘tight-coupling’ approximation
  (baryons follow electrons because of very strong e-m coupling)
•Background recombination physics (Saha/full multi-level calculation)
•Linearized General Relativity
•Boltzmann equation (how angular distribution function evolves with scattering)

CMB power spectrum C_l

- Theory: Linear physics + Gaussian primordial fluctuations

$$a_{lm} = \int d\Omega\ \Delta T\ Y_{lm}^{*}$$

Theory prediction $C_l = \langle | a_{lm} |^2 \rangle$

- variance (average over all possible sky realizations)
- statistical isotropy implies independent of *m*

Initial power spectrum
+ cosmological parameters
+ background evolution

linearized GR
+ Boltzmann equations
→

C_L
+ lensed C_L
+ other linear late-time observables

Boltzmann Codes

CMB anisotropies + linear $P_m(k,t)$, lensing etc.

- Cosmics (Ma & Bertshinger) – slow brute force
  Fortran 77, not maintained

- CMBfast (Seljak & Zaldarriaga) – Fast line-of-sight method + spline tricks
  Fortran 77/90, not maintained

- CAMB (Lewis & Challinor)
  http://camb.info/
  http://github.com/cmbant/CAMB
  http://camb.readthedocs.io/en/latest/ ("pip install camb", "conda install camb")
  OOP Fortran 2008 and Python 3.6+

- CMBEasy (Doran)
  OOP C++, not maintained

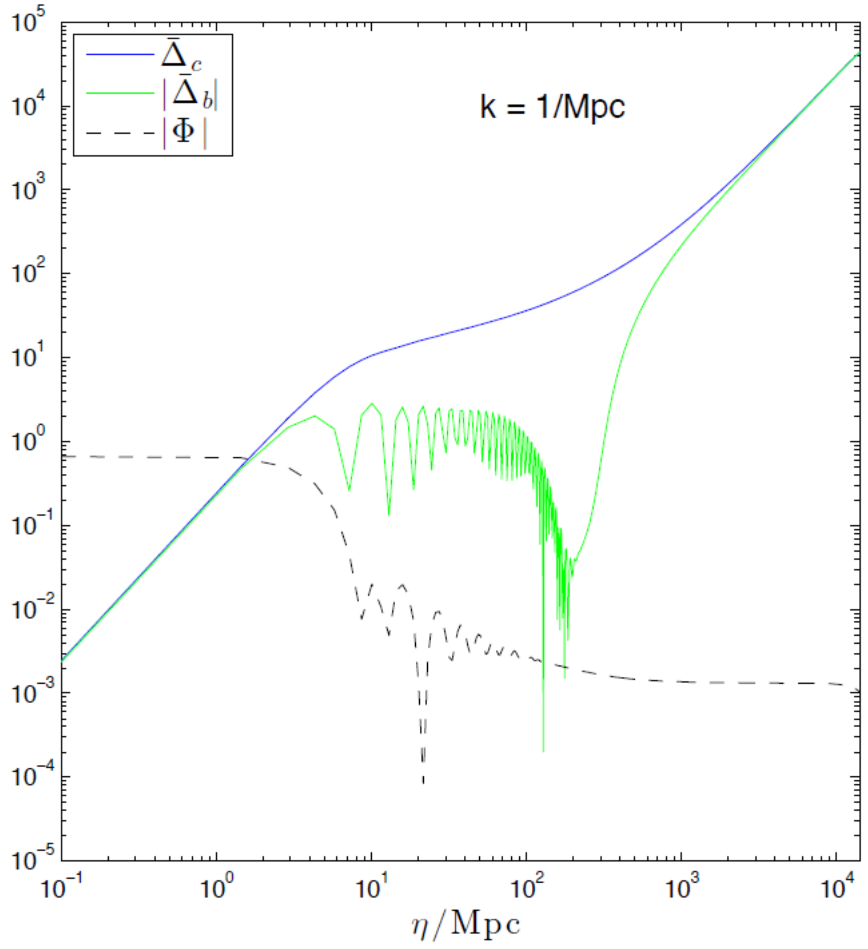
- CLASS (Lesgourges, Tram, Blas et al)
  http://class-code.net/
  https://github.com/lesgourg/class_public
  https://lesgourg.github.io/class_public/class_public-3.1.0/doc/manual/html/index.html
  Plain C and Python

+ various codes in development/not very widely used/specialist purpose (PyCosmo, CMBAns, Bolt.jl, …)
+ various extensions to CAMB or CLASS

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

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

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

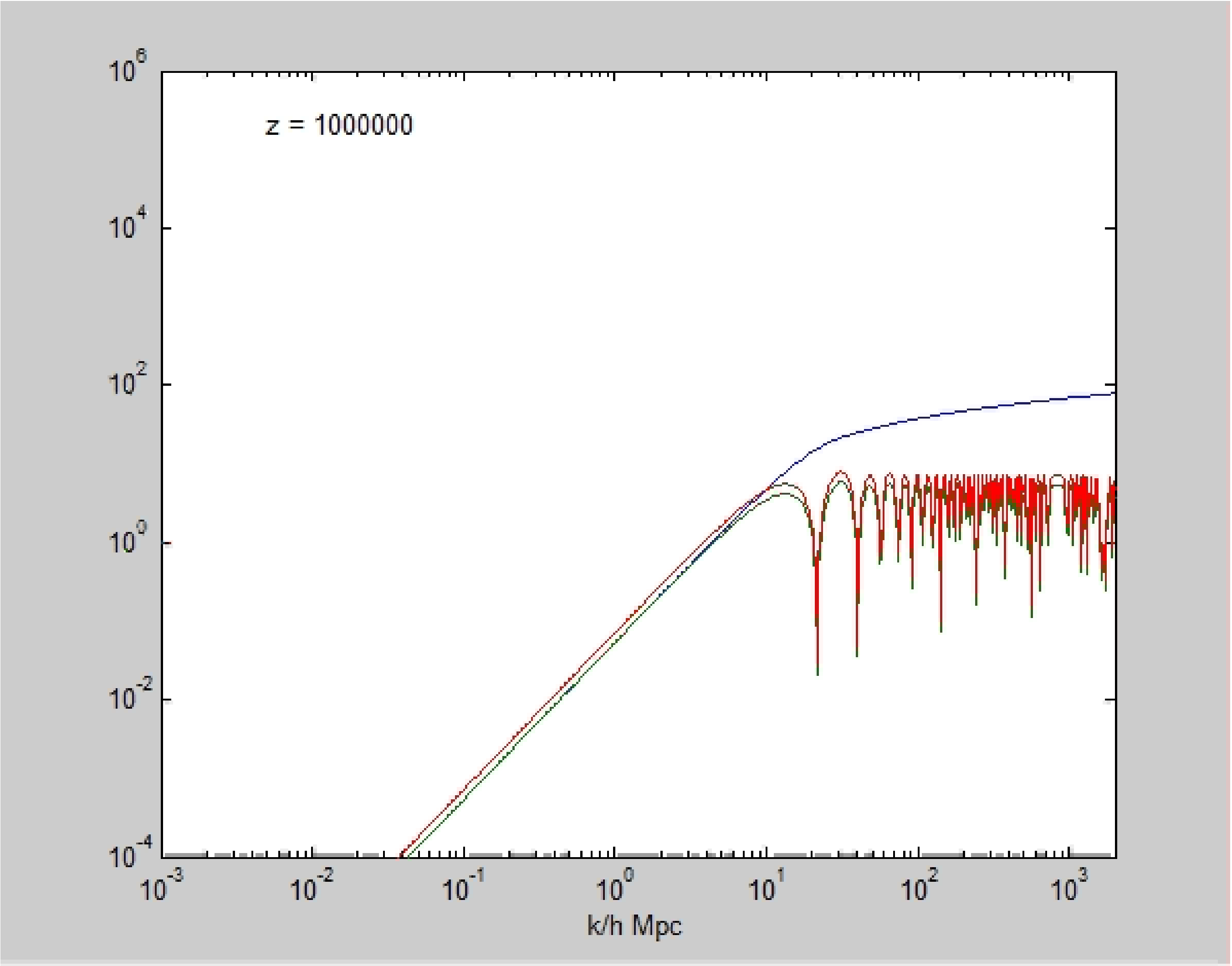
Note: CAMB integrates transfer functions for variables defined in the CDM frame/synchronous gauge.

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 ($0 \le l \le 2-30$)

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

Photon polarization

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

$$\mathcal{B}_l^{m\pm\prime} + k\left[\beta_{l+1}^m \frac{(l+3)(l-1)}{(l+1)^3}\frac{(l+1)^2 - m^2}{(2l+1)}\mathcal{B}_{l+1}^{m\pm} - \frac{l}{2l+1}\mathcal{B}_{l-1}^{m\pm} + \frac{2m}{l(l+1)}\sqrt{\beta_0^m}\mathcal{E}_l^{m\mp}\right] = 0. \qquad (15)$$

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

+ Baryons, dark matter, dark energy, …

+ Gravitational perturbations

Equations are evolved (e.g. 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 $T, E, B\ or\ \phi$ (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: e.g. using HALOFIT approximate model)

What can we learn from the CMB?

- Initial conditions
  What types of perturbations, power spectra, distribution function (Gaussian?);
  ⇒ constrain inflation parameters or alternatives.

- What and how much stuff
  Matter densities (Ω_b, Ω_{cdm});; neutrino mass; dark matter mass/interactions/decay
  (details of peak shapes, amount of small scale damping)

- Geometry and topology
  global curvature Ω_K of universe; topology
  (angular size of perturbations; repeated patterns in the sky)

- Evolution
  Expansion rate as function of time,
  dark energy evolution: w = pressure/density (with CMB lensing/external data);
  reionization; Hubble constant H_0 (assuming model)

- Astrophysics
  S-Z effect (clusters), foregrounds, etc. Tests of growth/modified gravity etc.

Conclusions

- CMB background now well modelled, 2+ codes available for spectral distortions + ionization history.

  The future? (only weak distortion constraints currently)

- Multiple CMB anisotropy codes mature, fast (~1s per model)

  - Linear CMB power spectra (trivial to change inflation $P_0(k)$)
  - CMB lensing power spectrum and lensed C_l
  - Matter power spectrum
  - galaxy counts/galaxy lensing 21cm C_l etc. – in linear theory
  - approximate models for non-linear corrections

Existing body of highly-constraining CMB data (Planck+ACT/SPT/Bicep)
New CMB data coming on small scales, B-mode polarization etc.
Explosion of large-scale structure data (LSST, DESI, Euclid, etc)

E.g. 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
```

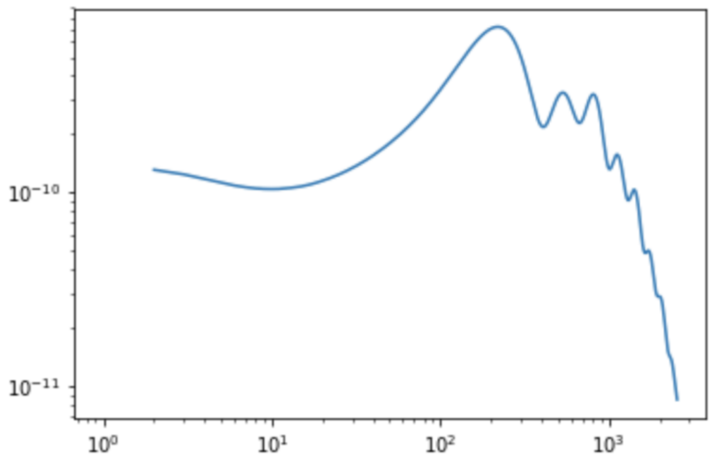
For example the total (lensed scalar+tensor) CMB power spectra are

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

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

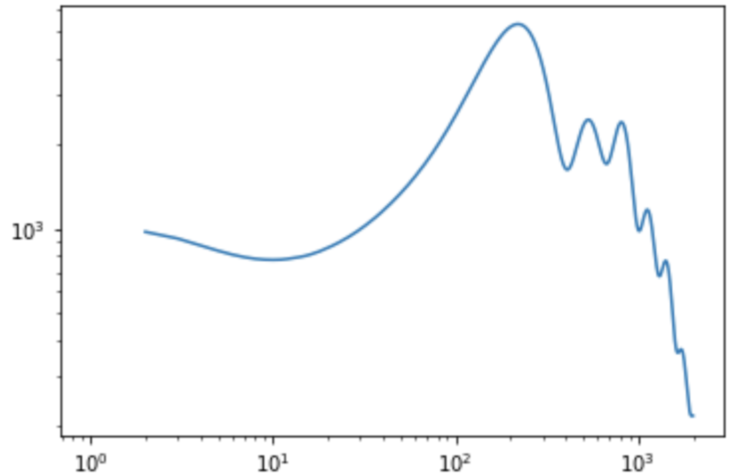
C_L result array by default are $\frac{l(l+1)C_l}{2\pi}$ in dimensionless units (i.e. $\Delta T/T$)

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



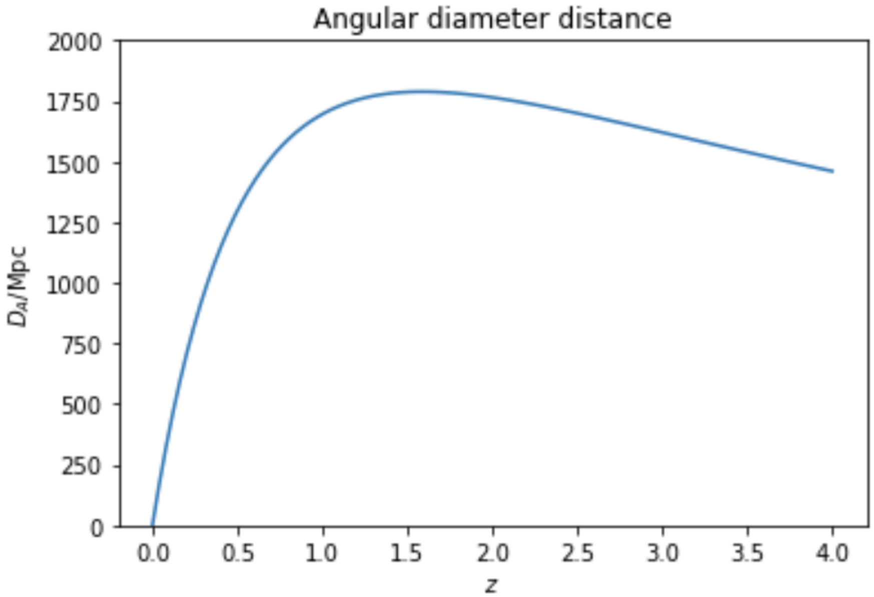

```
lensed_scalar =results.get_lensed_scalar_cls(lmax=2000, CMB_unit = 'muK')
plt.loglog(np.arange(2001), lensed_scalar[:,0]);
```

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, derived 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 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('contraints',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'

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

'contraints'

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

'var_subs'

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

'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'

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

'delta_eqs'

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

...

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)
```

$$\left[\frac{d}{dt}\Delta_r(t) = -k\,\mathrm{q_r}(t) + 4\frac{d}{dt}\Phi_N(t), \quad \frac{d}{dt}\Delta_g(t) = -k\,\mathrm{q_g}(t) + 4\frac{d}{dt}\Phi_N(t),\right.$$

$$\frac{d}{dt}\Delta_b(t) = -\frac{1}{\rho_b(t)}\left(\left(k\,\mathrm{v_b}(t) - 3\frac{d}{dt}\Phi_N(t)\right)(\mathrm{p_b}(t) + \rho_b(t)) + 3\left(\mathrm{c_{sb}^2}(t)\rho_b(t) - \mathrm{p_b}(t)\right)\Delta_b(t)H(t)\right),$$

$$\frac{d}{dt}\Delta_c(t) = -k\,\mathrm{v_c}(t) + 3\frac{d}{dt}\Phi_N(t),$$

...

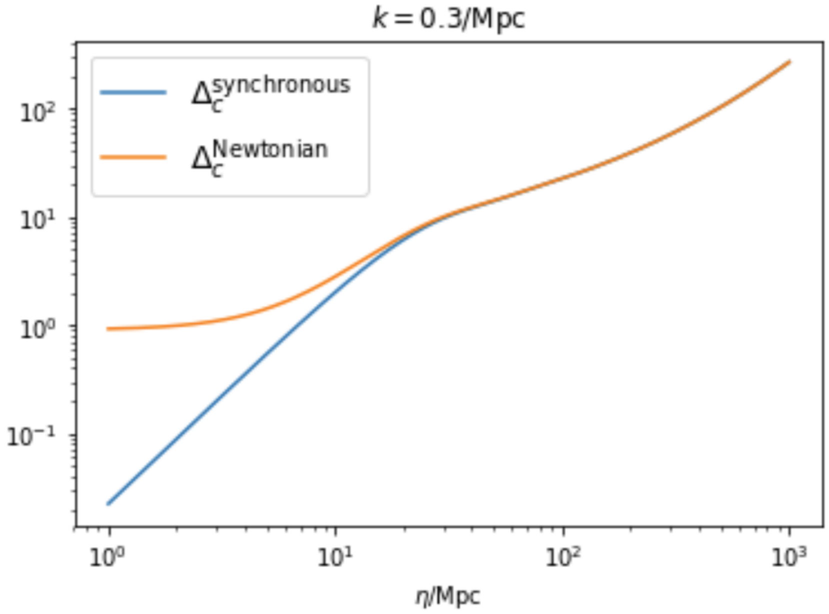
camb_fortran can convert symbolic expressions into CAMB-variable Fortran source code

```
#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 + &
```

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);
```



See further symbolic examples at
http://camb.readthedocs.io/en/latest/ScalEqs.html